Development and benchmarking a novel scatter search algorithm for learning probabilistic graphical models in healthcare

John Threlfall

Cardiff University School of Mathematics
University of Cardiff

This dissertation is submitted for the degree of
Doctor of Philosophy

Cardiff University December 2022

I would like to dedicate this thesis to my loving parents ...

Acknowledgements

I want to acknowledge my supervisors, Professor Daniel Gartner and Professor Paul Harper, for their support and guidance during my PhD, which would not have been possible without their full support. To the funding body KESS2 and the KESS2 team who have provided funding and courses to help achieve this PhD.

Dr Christian Bannister encouraged me to start a PhD and has guided me through this PhD and worked with me to improve my scientific methods and how to guide a reader with scientific data in a clear manner.

Aneurin Bevan University Health Board has been the industry partner for this PhD and has provided support and resources.

A special thanks to Professor Paul Edwards at the Aneurin Bevan University Health Board, who helped to set up the ovarian cancer study and helped with data collection and domain knowledge. The following people on Paul's team are Dr Amber Summmerhayes, Nell Redman, Shannon Edwards and Jim Sweet.

In memory of Sian, who sadly passed away from ovarian cancer and who inspired her husband Jim, who has tirelessly fought to further research into earlier ovarian cancer diagnosis. It has been an honour to help in this endeavour.

Professor Rema Padman of Heinz College of Information Systems and Public Policy at Carnegie Mellon University has advised on most of the studies I have been part of and has helped improve my academic paper writing skills and scientific research methods. Thanks to her for allowing me to contribute to the No-Shows for Breast Cancer Follow-up Visits; An India Perspective study.

Dr Emily Williams and Dr Emma Aspland who help guide me through the first year of academic life and set me on a good path for the following years.

To my loving wife and family who have supported me throughout this journey.

Abstract

Healthcare data of small sizes are widespread, and the challenge of building accurate inference models is difficult. Many machine learning algorithms exist, but many are black boxes. Explainable models in healthcare are essential, so healthcare practitioners can understand the developed model and incorporate domain knowledge into the model. Probabilistic graphical models offer a visual way to represent relationships between data. Here we develop a new scatter search algorithm to learn Bayesian networks. This machine learning approach is applied to three case studies to understand the effectiveness in comparison with traditional machine learning techniques.

First, a new scatter search approach is presented to construct the structure of a Bayesian network. Statistical tests are used to build small Directed acyclic graphs combined in an iterative process to build up multiple larger graphs. Probability distributions are fitted as the graphs are built up. These graphs are then scored based on classification performance. Once no new solutions can be found, the algorithm finishes.

The first study looks at the effectiveness of the scatter search constructed Bayesian network against other machine learning algorithms in the same class. These algorithms are benchmarked against standard datasets from the UCI Machine Learning Repository, which has many published studies.

The second study assesses the effectiveness of the scatter search Bayesian network for classifying ovarian cancer patients. Multiple other machine learning algorithms were applied alongside the Bayesian network. All data from this study were collected by clinicians from the Aneurin Bevan University Health Board. The study concluded that machine-learning techniques could be applied to classify patients based on early indicators.

The third and final study looked into applying machine learning techniques to no-show breast cancer follow-up patients. Once again, the scatter search Bayesian network was used alongside other machine learning approaches. Socio-demographic and socio-economic factors involving low to middle-income families were used in this study with feature selection techniques to improve machine learning performance. It was found machine learning, when used with feature selection, could classify no-show patients with reasonable accuracy.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

AI	Artificial intelligence
BN	Bayesian network
CPD	Conditional probability distribution
CPU	Central processing unit
DAG	Directed acyclic graph
E	Expected value
G	Graph
MB	Markov blanket
PD	Probability distribution
P	Probability
SS	Scatter search
TS	Tabu search

Summary

Healthcare data of small sizes are widespread, and the challenge of building accurate inference models is difficult. Many machine learning algorithms exist, but many are black boxes. Explainable models in healthcare are essential, so healthcare practitioners can understand the developed model and incorporate domain knowledge into the model. Probabilistic graphical models offer a visual way to represent relationships between data. Here we develop a new scatter search algorithm to learn Bayesian networks. This machine learning approach is applied to three case studies to understand the effectiveness compared to traditional machine learning techniques. First, a new scatter search approach is presented to construct the structure of a Bayesian network. Statistical tests are used to build small Directed acyclic graphs combined in an iterative process to build up multiple larger graphs. These graphs are then scored based on classification performance. Once no new solutions can be found, the algorithm finishes. The first study looks at the effectiveness of the scatter search constructed Bayesian network against other machine learning algorithms in the same class. The second study assesses the effectiveness of the scatter search Bayesian network for classifying ovarian cancer patients. Multiple other machine learning algorithms were applied alongside the Bayesian network. All data from this study were collected by clinicians in the Aneurin Bevan University Health Board. The study concluded that machine-learning techniques could be applied to classify patients based on early indicators. The third and final study investigated applying machine learning techniques to no-show breast cancer follow-up patients. Once again, the scatter search Bayesian network was used alongside other machine learning approaches. Socio-demographic and socio-economic factors involving low to middle-income families were used in this study with feature selection techniques to improve machine learning performance. It was found machine learning, when used with feature selection, could classify no-show patients.

Chapter 1

Introduction

1.1 Motivations of this Work

Varying datasets exist in healthcare regarding the number of variables and sample size. The most difficult of these datasets to achieve meaningful results are those with a high number of variables and low sample size. These datasets are subject to the curse of dimensionality, a concept first introduced by Richard Bellman [17]. As the number of variables increases in a dataset, the number of samples needs to increase exponentially for statistical significance to be found [34]. Techniques such as Principal Component Analysis (PCA) can be used for dimensionality reduction for use with machine learning algorithms, but information loss can occur [71].

One of the problems with many machine learning (ML) techniques is the inability to understand how or why the ML model arrived at the answers produced. These models are called black box models and cannot incorporate domain knowledge from experts.

One such way to incorporate knowledge into black box models is to use techniques that fall under the domain of feature engineering. Features also known as predictor variables can be ranked in relation to an outcome variable as discussed in chapter x. Predictor variables can also be tested in relation to an outcome variable using statistical tests. Variables can also be selected based on scores or p-values reducing noise when training a machine learning model. Knowing which variables to include in the model requires domain knowledge of the dataset, for example when predicting the risk of ovarian cancer the variable hysterectomy may give good results. This variable indicating if a woman has had a hysterectomy may give good results in classification but only because the patient had a hysterectomy because of ovarian cancer. Once discussing this with people who have domain knowledge it comes apparent that this variable needs to be removed.

New features can also be created by combining variables through simple mathematical operations or using equations to combine one or more variables. In electronics this could be using the equation amps*volts to produce a new feature watts. Feature combinations may use more than two variables present in the dataset.

Transformations are another feature engineering technique. Scaling falls under this area and can be used to scale variables between 0 and 1, this is common practice for using datasets with neural networks.

Probabilistic graphical models offer a visual representation of the relationships between variables (attributes) that offer a visual aid in discussions with stakeholders. Probabilistic graphical models can be enriched with domain knowledge that stakeholders may have. This visual modelling process ensures everyone involved can contribute regardless of their modelling knowledge. Graphical models can still be used alongside other machine learning models forming ensembles to increase predictive capabilities.

Learning Probabilistic graphical models (Bayesian Networks) is a two-fold process. The first is learning the structure of a model (network), and the second is learning the parameters (probability distributions) of the model (network). In this thesis, the focus has been on learning the structure of the network using a newly developed constraint-based scatter search heuristic.

Research undertaken in [13] demonstrates that probabilistic graphical models (Bayesian Networks) can be used for feature selection, dimensionality reduction and prediction. Using a meta-heuristic search algorithm, the structure of a probabilistic graphical model can be found and reduced by taking the Markov blanket. These probabilistic graphical models were still able to produce good classification results.

Key research questions answered in this thesis are detailed below.

- How can the dimensionality of healthcare datasets be reduced for probabilistic graphical models?
- How can a compact structure of probabilistic graphical models be learned that contains only the essential variables?
- Can scatter search be applied when learning the structure of a probabilistic graphical model?
- Can the chi-squared test be used with scatter search to find the conditional independencies?
- How can these models be applied to classification models in healthcare?

- How do the learned probabilistic graphical models perform compared to other classification models?

To answer these research questions, three case studies have been undertaken. These case studies are as follows;

Study 1 focuses on benchmarking the scatter search along with other machine learning algorithms against well-known datasets from the UCI repository. This study gives a good understanding of how well the scatter search algorithm performs.

Study 2 is a study into the prediction of the risk of ovarian cancer from questionnaire data. The purpose and motivations behind this are to build a model that patients could use via a mobile app or GP to warn of a possible risk of ovarian cancer. Studies in the chapter show early detection and treatment are key to increasing survival rates.

Study 3 is a case study into the Indian breast cancer no-show dataset. Within this study, a model has been constructed to detect patients at risk of not attending breast follow-up appointments that may require intervention by a healthcare professional.

1.2 Main Contributions of the Thesis

During the development of this body of research, a new constraint-based scatter search approach was developed to learn the structure of a Bayesian network from data. The scatter search allows multiple solutions to be learned from data, with the final models being assessed based on metrics such as accuracy and confusion matrix metrics, although not limited to these metrics.

The scatter search Bayesian network algorithm has been applied to the ovarian cancer case study as well as the breast cancer no-show study. During the development of the algorithm, it was also applied to datasets from the machine learning repository for the first time.

Ovarian cancer suffers from late detection of the disease, reducing positive outcomes from treatment. It was hypothesized that the detection of ovarian cancer could be improved using questionnaire data from patients that showed early symptoms of the disease. Multiple machine learning algorithms, along with statistical methods were applied to the data. The outcome of the study found that a high accuracy with a high True Positive rate could be achieved. This research has developed models that could be implemented in an app or healthcare system to improve early ovarian cancer detection by referring those at risk for further testing.

The Breast Cancer no-show study attempts to build models to predict follow-up appointments and no-shows after a patient have had breast cancer treatment. A literature review

during the study found more than half of breast cancer deaths occur in low to middle-income countries. Monitoring patients for follow-up appointments is important to ensure they receive the correct aftercare treatment and monitor for recurrences of breast cancer. The data in the study were trained on multiple machine learning algorithms, including scatter search. It was found that models could achieve accuracies of up to 92.2%, a precision of 100 and a true positive rate of 96.1 although not necessarily in the same test iteration.

1.3 Overview

Chapter 2 focuses on the working of bayesian networks with an introduction to what they are and how they work. Construction of these networks with the three main areas is discussed, these areas are scoring, constraint and hybrid methods. Distribution fitting techniques such as Bayesian and Maximum likelihood are explained. The basic building blocks behind approximate and exact inference are covered in section 3.2.3. Markov blankets and metrics are discussed, with the final section covering search heuristics.

Chapter 3 reviews the current literature on machine learning and healthcare with a focus on classification. The search is grouped into four parts, group 1 feature selection and machine learning techniques, group 2 Bayesian networks and Markov blankets, group 3 search heuristics and group 4 applications in healthcare. Using the search methodology laid out in the PRISMA guidance and using a backward and forward search a systematic search was carried out using Scopus. During the search, 226 papers were found, filtered down to 59 relevant findings. Literature papers have been broken down into the country of origin, applications in healthcare, attribute selection and classification, bayesian network learning type, scoring methods, constraint-based methods and hybrid methods. Key books used during the development of the scatter search have been included. These books cover topics such as machine learning, python, statistics, search algorithms and data science. All books were located through Cardiff university library services, Google, Scopus, Springer, conferences and Amazon. In total 18 books were listed.

Chapter 4 introduces the newly developed constraint-based scatter search heuristic for learning the structure of the Bayesian network. The chapter examines related work with the scatter search algorithm and its origins. The methods used, and the operation of the algorithm applied to Bayesian networks are detailed.

Chapter 5 applies the scatter search to benchmarking datasets from the UCI machine learning repository. Using these benchmarking datasets, other machine learning algorithms are also applied to compare scatter search against them. The results of this study are then compared and discussed.

Chapter 6 is a study on ovarian cancer risk prediction from questionnaire data. This study aims to provide a quick and reliable way to model the prediction of ovarian cancer from possible early symptoms. This study uses multiple machine learning models, including the scatter search, to find the best possible models for prediction.

Chapter 7 is a study into predicting no-show follow-ups for breast cancer patients. Multiple machine learning models have been applied alongside the scatter search algorithm. This study differs from the last as attribution ranking selection techniques have been applied with groups of patients split by socioeconomic factors to understand the effects these can have on the models.

Chapter 8 is the summary looking back at the development of a new Bayesian network search algorithm called scatter search and the application of this algorithm deployed alongside other machine learning algorithms in three different studies.

Chapter 2

Bayesian Networks

2.1 Introduction

A Bayesian network is a graphical representation of probability distributions and the corresponding conditional probabilities. Graph nodes (vertices) represent probability distributions of attributes, and the edges in the graph represent conditional probabilities [166]. Bayesian networks must be a Directed acyclic graph (DAG) in that there must be no cycles in the network. This forms a graphical network with no directed cycles [88].

Bayesian networks offer a human graphical interpretable representation of relationships between attributes. These representations can complement expert domain knowledge and provide an easy way for modellers to work with domain experts to represent a system or problem properly.

There are many machine learning methods available to construct and fit the distributions for the network. Equally, there are many methods for evaluating the network's performance and fitness for purpose, but ultimately, the domain experts will assess the resulting network.

Bayesian networks find uses in classification, risk and continuous variable prediction applications.

2.2 Bayesian Networks

2.2.1 Network Construction

Bayesian networks need methods to construct the network and fit the distributions. Many ways have been developed over the years. Algorithms can be classified into three categories: scoring-based, constraint-based, and hybrid-based [158].

Scoring

Scoring-based algorithms use metrics scores such as BD (Bayesian Dirichlet), K2, MDL/BIC (Minimum description length/Bayesian Information Criterion) and AIC (Akaike Information Criterion).

Using a search algorithm such as TABU search, edges will be switched, and the overall network will be scored. The score will guide the search algorithm, and the algorithm will attempt to optimise the solution based on the given score.

The score given for the overall network is a goodness of fit statistic. Another way to look at it is the optimisation function will attempt to minimise or maximise the given statistic.

Constraint

Constraint-based methods use statistical tests such as chi-square, Kruskal Wallis test, t-test, ANOVA, MANOVA, Sign-test, etc. The type of statistical test to be used depends on the distribution type, continuous or discrete and if the distribution is parametric or non-parametric.

When building the graph structure, statistical tests test for conditional independence between nodes (attributes). When a p-value is less than 0.05, the child node has dependencies on the parent nodes. These dependencies form the edges of the network.

A search algorithm works through the nodes testing for dependencies strategically. As the algorithm progresses, the structure is built up, and a network is formed that complies with the Directed acyclic graph DAG need for a Bayesian network.

Hybrid

A hybrid approach uses both scoring and constraint-based methods to construct a graph. Many algorithms, such as TABU search, use a constraint-based system to build an initial graph. This graph is then improved upon by adding, switching and deleting edges through a guided search optimised on a network score such as K2.

2.2.2 Distribution Fitting

Once the network structure is found, the distributions represented by the nodes still need fitting. Distributions can be both parametric and non-parametric. Standard parametric distributions used are Normal Distribution, Student T Distribution, Exponential Distribution, Beta Distribution and Bernoulli Distribution. Multi-way tables can represent Non-parametric data.

Bayesian Estimation

The Bayesian estimator is used in Bayesian networks to fit the distributions of the graph. There are three principal distributions, the prior, the likelihood and the posterior. The prior and the likelihood distributions are used to form the parameters for the posterior.

The prior distribution is the prior belief of the posterior, and the likelihood is the frequentist approach formed from the data of what the posterior should be.

We multiply the prior values by the corresponding likelihood values to form the posterior distribution. We then obtain the marginal likelihood values by adding the values in the previous step. We then obtain the posterior by taking the values obtained in the first step and dividing them by the marginal values taken in the second step.

Maximum Likelihood

The maximum likelihood is a frequentist approach to estimating the distribution of data. Maximum likelihood attempts to from the optimal value for the mean and the standard deviation. Whilst iterating over possible data values for the mean and standard deviation, the algorithm finds the maximum value for both.

The drawback of this method compared to the Bayesian estimator is that prior assumptions are ignored and are not incorporated into the distribution.

2.2.3 Inference

The ability to perform inference on a Bayesian network model is essential for practical applications in the real world, such as classification and risk modelling—the ability to ask the models questions and receive approximate and exact answers. Some applications will need more than binary responses with a full probability distribution to be returned. The following examples will detail two standard methods.

Markov Chain Monte Carlo (MCMC)

The Markov chain Monte Carlo method comes in two parts. As the name would suggest, there are the Markov chain rules and then the distribution sampling using the Monte Carlo method.

Using the Markov chain rule, we only need the last distribution in the chain to sample from. In Figure 2.1, only X1 is required to sample from in order to generate an approximation of distribution Y1. The other variables in the chain are not relevant from sampling if we know the parameters for X1.

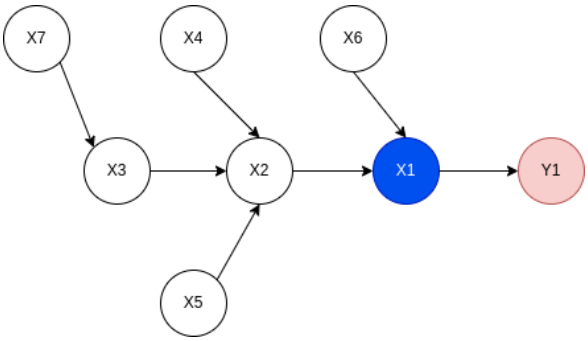


Fig. 2.1 Markov chain rule

Monte Carlo simulations are an approximate way of repeated random sampling to generate results of an unknown distribution through computational means to give a density estimation.

Variable Elimination

Variable elimination is a technique used to reduce the number of computations needed to calculate the probability chain rule across the Bayesian network. The algorithm works to group common factors in the network so these will only be computed once and stored for use in other iterations through the network. The algorithm's main functionality is to reduce the computational complexity.

Junction Tree

A junction tree, also known as a clique tree, can be used with a Bayesian network and is an example of exact inference. By turning the Bayesian networking into an undirected graph, we can set the actual observed values for the tree. These values need to be concerning the variable we want to know the values of given some evidence.

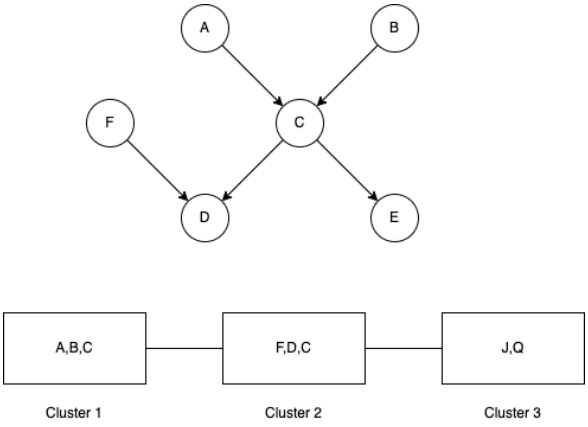


Fig. 2.2 Junction tree graph

Junction trees build on variable elimination by taking subgraphs of the main graph called clusters. The following three rules are needed for a junction tree to work; There can be only one path between clusters, as shown in figure 2.2, a cluster subset from the main graph must belong to a clique, and a node separated between clusters must also be in the separating cluster. Inference with the junction tree is done using the belief propagation algorithm.

2.3 Markov Blankets

Markov blankets are important in Bayesian networks as they can reduce the size of the network. An example is given X is the variable we want to predict; we only need the children of X and the parents of those children. This sub-selection of variables is called the Markov boundary and is a lot more computationally efficient when estimating distributions and performing inferences over the network.

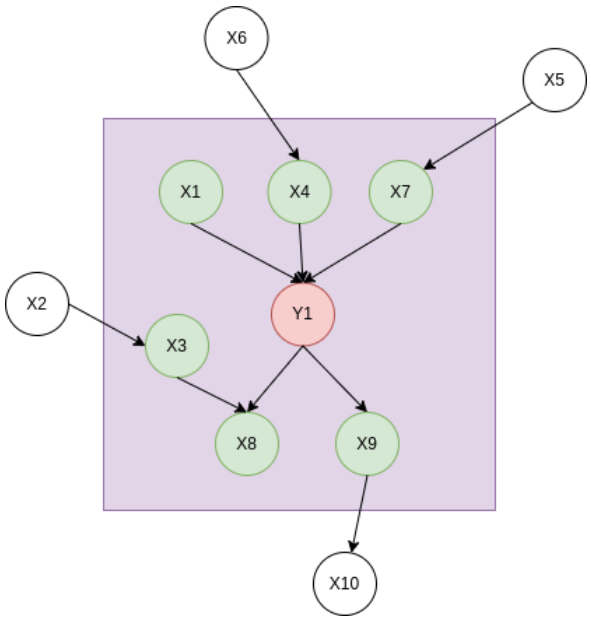


Fig. 2.3 Markov blanket

Figure 2.3 shows the Markov boundary of Y1. Everything inside the box is necessary for Y1, and everything outside of it can be discarded for the Bayesian network. Using this concept, a much more computationally efficient Bayesian network is created.

2.4 Metrics

2.4.1 Scoring

Scoring-based metrics score the network's overall performance and with the use of an optimisation algorithm attempt to improve upon the score given to the network. Many such algorithms exist and only four have been discussed here. These are the most common that can be found and have been used widely in research papers.

BD (Bayesian Dirichlet)

BDeu is an improvement on the BD scoring algorithm. The BD scoring algorithm takes the joint probability of the Bayesian network with the variable-parent combinations specified. BDeu further improves on this by calculating all of the needed parameters from the equivalent sample size and a prior distribution over the network. Improvements were also made to make the BDeu score equivalent for network structures that are the same. [114]

K2

The K2 metric is used to evaluate how well the Bayesian network fits the data and is a common Bayesian-based metric. The metric assumes prior uniform distributions on all possible initiations of parent attributes [25].

MDL/BIC (Minimum Description Length/Bayesian Information Criterion)

MDL is based on information theory, and the simplest model that best represents the problem is the best solution. MDL encodes the graph in two parts: the structure and the unexplained data. Encoding of the model is achieved through tables of conditional probabilities. The goal of the MDL score is to minimise it rather than maximise it, as with other algorithms. [114]

AIC (Akaike Information Criterion)

AIC is calculated from the number of variables used to build the model and the maximum likelihood estimate of the model. The scoring is based on the asymptotic behaviour of models with sufficiently large datasets. AIC favours more complex terms than MDL because of the lower penalty term.

2.4.2 Performace

Performance metrics are used to assess the Bayesian network performance whilst being used for inference. Common metrics such as accuracy, ROC, precision and recall are used for judging the fitness of a network. Bayesian networks can also be optimised based on these metrics.

Accuracy

Accuracy is the true values of true positive and true negative over the total number of samples inference has been performed on. This metric is very common but suffers in the case of imbalanced datasets and should be used in conjunction with other metrics to judge its reliability of it.

Receiver Operating Characteristic (ROC) and Area Under Curve ROC (AUC-ROC)

The receiver operating characteristic (ROC) metric looks at the relationship between the true-positive rate and the false-negative rate. The true positive is on the Y-axis, and the false positive is on X-axis. A score of 0.5 indicates the classifier is no better than predicting classes at random. Good scores on the Y-axis are 0.5+ with 1 being a perfect classification score. Scores dropping below 0.5 on the Y-axis indicate worse performance than a totally random score and a poorly performing classifier.

Recall

Recall also known as sensitivity, hit rate, or true positive rate (TPR), this metric is the true positives over the true positives plus the false negatives. The information provided by recall is the number of successful classifications over all the classifications performed.

Precision

Precision/positive predictive value (PPV) is defined as the true positives over the true positives plus the false negatives. Precision shows the proportion of the true positives with respect to the true positives and false negative classifications.

Confusion Matrix

A confusion matrix is a table that represents the True Negative (TN), False Negative (FN), True Positive (TP) and False Positive (FP). In classification, the predicted values vs actual values are represented in a table to understand the performance of the classifier performance.

A simple binary classification problem uses a 4x4 table but for multi-classification problems, this can be an NxN dimensional table.

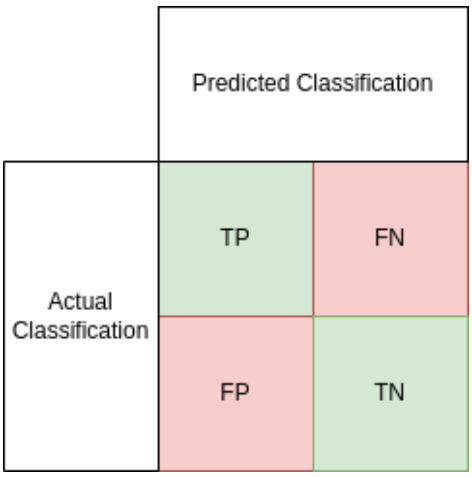


Fig. 2.4 Binary confusion matrix

Figure 2.4 shows a simple binary confusion matrix. Predicted classifications are compared against actual classifications.

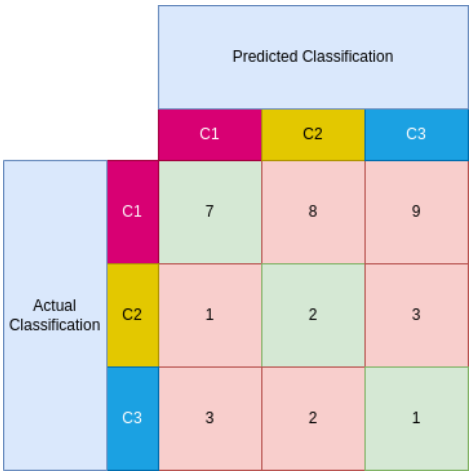


Fig. 2.5 Multi class confusion matrix

Figure 2.5 shows a multi-classification confusion matrix, just like a binary confusion matrix the actual vs predicted classes are compared and counted in the table.

True Negative (TN)

True negatives are defined as the number of correct classifications of negative values.

False Negative (FN)

False negatives are the number of incorrect classifications of false values.

True Positive (TP)

True positives are the number of correct classifications of a true value.

False Positive (FP)

False positives are the number of incorrect classifications of false values.

2.5 Search Heuristics for Probabilistic Graphical Models

Search heuristics are search procedures to find the optimal solution to a problem using different strategies. Many such strategies have been developed over the years for Bayesian networks, Three common types have been explained below. A review of current search strategies has been reviewed by [156] with a list of tools implementing these search strategies.

2.5.1 Hill Climbing

The Max-Min Hill-Climbing algorithm was first proposed by [171]. MMHC is a hybrid algorithm with two parts. The first is to find the parents and children of each attribute independently with an algorithm called Max-Min Parents and Children (MMPC), this algorithm finds the undirected graph. Conditional independence tests are used to find the children of the parents which reduces the search space.

Once all parents and children are found hill climbing is used to find the highest scoring graph. The hill-climbing starts with an empty graph and uses edge addition, deletion and edge reversal to find the largest increase in the score. When the score does not improve for x number of iterations the algorithm terminates with the best scoring graph [176].

2.5.2 Simulated Annealing

Simulated annealing (SA) is a probabilistic metaheuristic stochastic global search optimization algorithm that decided to move to the next state based on the current state [32]. Simulated annealing algorithms are based on the process of annealing metals.

The simulated annealing process has a key hyperparameter called temperature that is used to control the diversity of solutions and the movement within the search space. Once

the search begins the temperature is high with many diverse solutions being accepted. As the search continues the temperature variable is decrease limiting the search and diverse solutions on which to improve. When starting with a large temperature value the algorithm has the ability to freely move about the search space and find the best local minimum. As the temperature is brought down the algorithm is forced to converge to a minimum [99].

2.5.3 Tabu Search

Tabu search is a greedy search algorithm that uses adaptive memory techniques to prevent it from getting stuck in local minimums and stop it from visiting previous search spaces. Tabu search does allow diverse solutions (not the best solution) to prevent getting stuck in minimums.

Intensification is the process by which tabu search uses short-term memory to return to high-scoring solutions and localize and intensify the search. In order to diversify the search, long-term memory is used to allow lower-scoring solutions to be revisited.

Key research has been conducted in [12] that looks at the use of Tabu search for use with Bayesian networks to improve learning and classification models.

2.6 Summary

Bayesian networks offer a way to encode the probabilities of high dimensional datasets through the use of conditional independencies. Many strategies to learn Bayesian networks from data are available and can be broken down into three areas. These areas are constraint base, scoring based and hybrid methods, furthermore, these can be broken down into two learning areas, structured learning and parameter learning.

Inference with Bayesian network models can be done by exact or approximate inference. Exact inference can be achieved via variable elimination and junction trees although these are not the only methods of exact inference. Approximate methods such as Monte Carlo simulations can also be used to sample the distributions in the network to output values from the final distribution.

Many metrics are available to assess the performance or goodness of fit of the model. To assess the overall structure of the model, methods such as the BDeu, K2 metric and Minimum description length can be used, these methods are useful when performing structure learning. Metrics such as Accuracy, Receiver operating characteristic (ROC), Area under curve ROC (AUC-ROC), Recall, Precision, True Negative (TN), False Negative (FN), True Positive (TP),

False Positive (FP) can all be used to guide the fitting of the parameters as well as assess the networks overall performance in classification tasks.

In order to guide the search for the best structure and parameters for the model, search heuristics need to be applied. Many search heuristics exist and are covered by the literature review. The three covered here are Max-Min Hill climbing, Simulated Annealing (SA) and Tabu search. Each search has advantages and disadvantages, such search algorithms can be categorised by local and global search strategies as well as heuristics or metaheuristics. Metaheuristics can be used to guide the search for a metaheuristic to prevent getting trapped in local optima.

Chapter 3

A Review of Probabilistic Graphical Models

3.1 Introduction

In 2025, the amount of data which is generated annually worldwide is predicted to reach 175 zettabytes (Statista.de [53]). The availability of cheap storage volumes of any size and type which simplify the storage of any kind of data is just one explanation of this development. The massive storage of data forces organizations, such as health services, to structure their data in order to make it useful for medical decision-making.

Probabilistic graphical models offer a visual way to present and interpret relationships between attributes in datasets. These visual representations are useful when understanding the model and working with stakeholders, such stakeholders may offer valuable domain knowledge to further improve the graph without the need for machine learning knowledge themselves. Once the domain knowledge has been understood by the machine learning practitioner the model can be modified manually or rules added to the training procedure to incorporate the knowledge.

The aim of this literature review is to present state-of-the-art of research on probabilistic graphical models applied in healthcare. We set our focus on attribute selection, classification, as well as risk prediction tasks.

Our review highlights common themes such as scoring-based methods and the applications in which the algorithms were used along with the different techniques utilised. Also, our review provides a discussion of potential future research scope.

The remainder of the review is structured as follows: Section 2 introduces the methods used to identify the papers and discusses related literature reviews identified through this

search. Section 3 analyses and provides a classification of the results. Section 4 discusses gaps within the research, with Section 5 identifying areas for future research. Section 6 provides a conclusion. For the figures which discuss a classification result, a respective table within the Appendix has been included detailing the reference numbers for each paper. Table A7 within the Appendix provides a comprehensive list of the 62 papers including each classification category.

3.2 Search, Selection Criteria and Previous Literature Reviews

3.2.1 Search Criteria

In conducting this literature search we followed the recommendations and suggestions laid out in [179] and conducted a forward and backward search using the Scopus search engine. Scopus offers an advanced search engine and query language that also supports backwards and forward searching of research documents. An initial search was carried out using the query in Figure 3.1 with the returned documents being reviewed using the PRISMA process shown in Figure 3.2. Once the documents have been reviewed the final documents references are used to perform a backward-forward search laid out in Figure 3.3. A backward and forward search allows documents that have been referenced by the current articles or reference the current article to be retrieved. Doing a backward and forward search helps a researcher understand the foundations on which the research is built on and any further developments in the area that may have been achieved. New articles found in a backward and forward search may not have been found by the initial search but are important to understanding the theory behind a subject matter and to understand if new ideas the researcher has are original in nature. The search query can be broken down into four facets. These facets break down the key areas we are looking to address such as machine learning in healthcare and in particular Bayesian networks applied to healthcare modelling.

Facet 1 is for machine learning and classification, feature selection has also been included as Bayesian networks are often used to find the most relevant features and find how they depend on each other.

Facet 2 is looking at probabilistic graphical modelling with a focus on Bayesian networks with the use of Markov blankets to reduce the complexity of the models.

Facet 3 is search heuristics to build these models and find the dependencies.

Facet 4 looks for applications in a healthcare setting.

```
TITLE-ABS-KEY (("feature selection" OR classif$ OR "
    attribute selection" OR "structural learning" OR "machine
     learning") AND ("bayesian network$" OR "bayesian belief
    network$" OR "graphical model$" OR "markov blanket") AND
    ("hill climbing" OR "construction heuristic$" OR "
    improvement heuristic$" OR "2-opt" OR "k-opt" OR "path
    relinking" OR "genetic algorithm$" OR "ant colony" OR "
    tabu search" OR "scatter search" OR "differential
    evolution" OR "bee algorithm$" OR "particle swarm" OR "
    harmony search" OR "firefly algorithm$" OR "simulated
    annealing" OR "grow shrink algorithm" OR "TAN" OR "K2" OR
     "tree augmented naive bayes" OR "PC" OR "Peter and Clark
    " OR "incremental association" OR "IAMB" OR "MMPC" OR "
    Interleaved Incremental Association and Max-Min Parents
    and Children" OR "total conditioning" OR "grow shrink" OR
     "TS" OR "TCbw" OR "GSIMN" OR "DGSIMN" OR "K2ACO" OR "
    ChainACO" OR "ACO" OR "BAN$" OR "TABU" OR "CLT" OR "
    Simple Structure Learning" OR "RSMAX2" OR "K2rev" OR "
    K2opt" OR "greedy" OR "breadth first search" OR "BFS" OR
    "depth first search" OR "dfs" OR "Minimum Spanning Tree"
    OR "Random Walk" OR "k-Shortest Paths") AND ("disease" OR
     "patient" OR "not show" OR "diagnos*" OR "drg" OR "hrg"
    OR "length of stay" OR "readmission" OR "health care" OR
    "healthcare" OR "procedure$" OR "urgent*" OR "elective"
    OR "emergency" OR "clinical" OR "no show" OR "do not
    attend" OR "non-attendance" OR "cases"))
```

Fig. 3.1 Scopus search query

Figure 3.1 shows the Scopus query used to perform the literature search, each facet and separated by the keyword 'AND'.

3.2.2 Selection Criteria

Once the number of papers has been reduced to 226 papers we can now filter even further to find 17 relevant references. The PRISMA diagram in figure 3.2 shows the process of filtering the documents down.

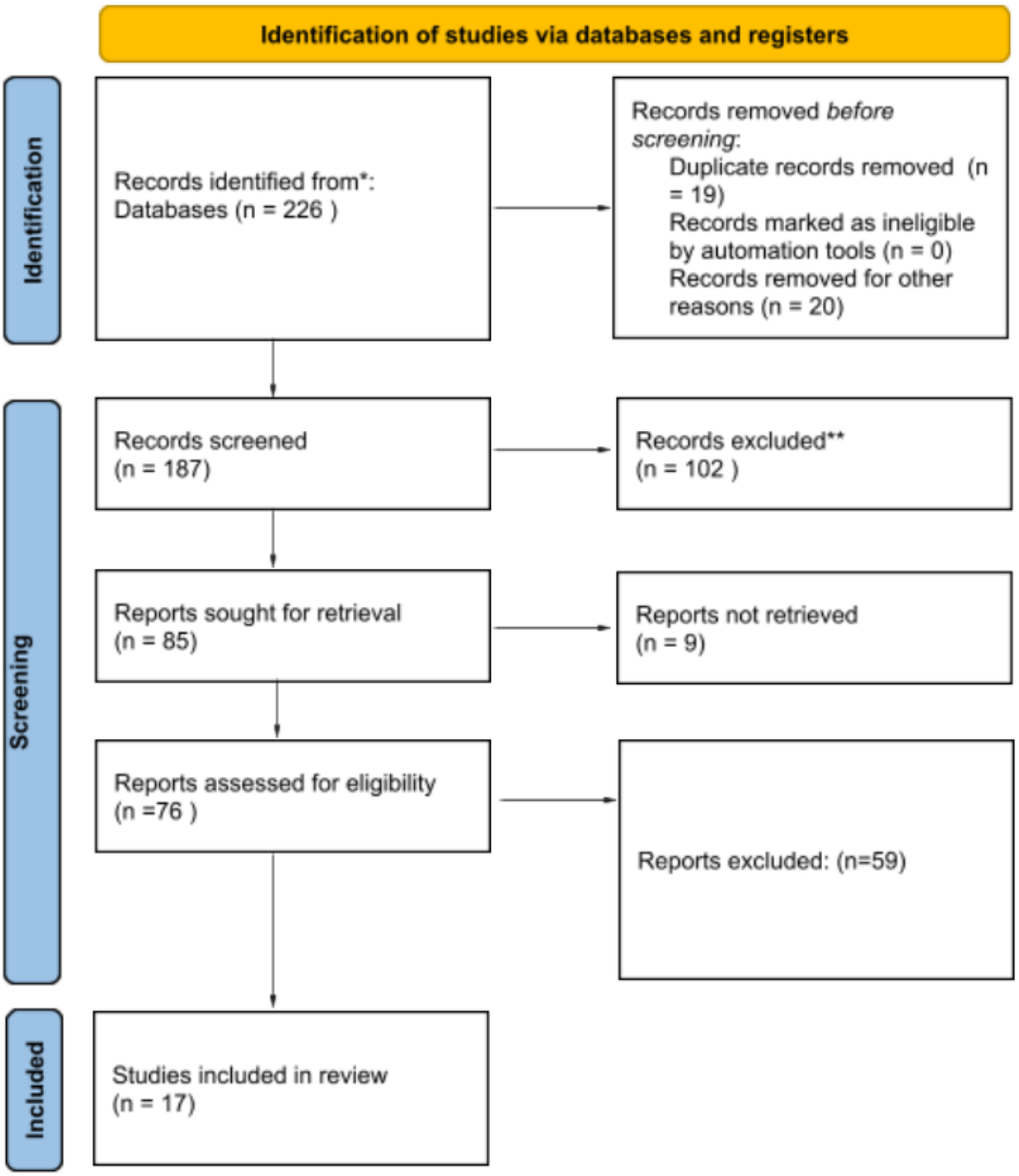


Fig. 3.2 PRISMA diagram

A forward search finds 98 references and a backward search returns 563 results, of these 4 from the forward search and 38 from the backwards search are now relevant. This brings the total number to 59.

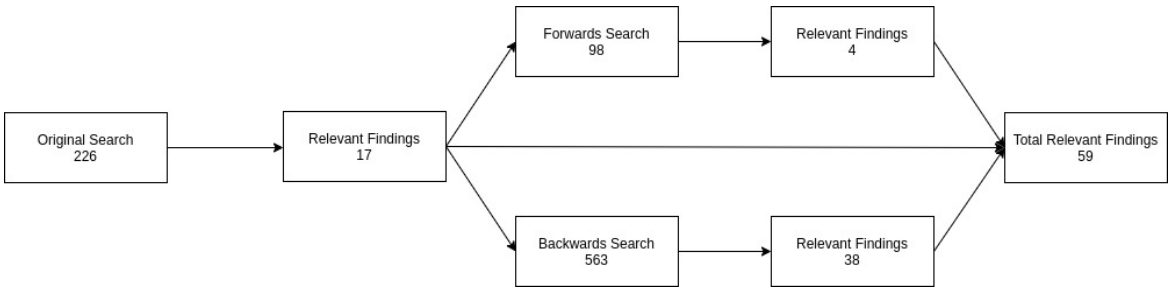


Fig. 3.3 Scopus filtered search

3.2.3 Previous Literature Reviews

There exists a vast amount of literature on Bayesian network modelling. From our findings, there are 4 relevant literature review articles that address Bayesian network applications in medicine. These reviews were found using a separate search for literature reviews using Google Scholar and Scopus and are treated separately from the other documents found. These literature reviews are covered separately as they put into context this review and why it exists. In what follows we will highlight each of the four reviews and summarize similarities and differences with our literature review.

The most comprehensive review is [102] which aims to bring together literature around Bayesian network (BN) modelling applied to healthcare including developing them. This review touches on four main themes, BNs not being used to their full potential, a generic process to train and deploy is still unavailable with many different ways to construct and train parameters of a BN, limitations in the literature of BNs in the way they are presented and finally the limitations of the impact BNs in practise and adoption.

Understanding when and where to apply BNs is important as there are many, [125] reviews of the current application areas in healthcare. The review looks at all areas of healthcare but identifies four common conditions in which BNs are applied, these are; cardiac, cancer, psychological and lung disorders. The authors conclude that it's not surprising 59% were in these four areas given "the notoriety of these conditions in the mainstream".

As more papers get published on the development of BNs within healthcare [101] finds the literature on the adoption in practice is limited. This review [101] looks into the chasm between research enthusiasm and clinical adoption. The key findings from this review are

that the published research lacks an overall BN development process and is difficult if not impossible to replicate results. Another key point is that if models are to be adopted these models need to be able to generalise in practice, something which has not been demonstrated in research papers. The final conclusion from this review is that despite immense interest and research into clinical decision making very few publications have been published on real-world implementations of this work. The author concludes that if the adoption of BNs is to be more widespread, more work is needed on the implementations of these BNs in the real world to assist clinical decision-making. The author states that the lack of published real-world implementations has led to duplicated efforts and allowed important research gaps to remain hidden.

The literature review we include here is [101], this review looks into what is preventing the adoption of BNs in medical decision-making. Key aspects that have been identified by the authors of this review are the benefits, barriers and facilitating factors in adopting BN-based systems in clinical practice. Groupings have been identified that prevent the implementation of BN systems, these break down as follows; lack of resources, clinical resistance, insufficient impact and model performance. It's been concluded that very few authors demonstrate an understanding of these barriers although many benefits were identified in the literature. The review also notes that there has been a clear gap between the development of BN models and the implementation of a useful decision support tool that can utilise those trained models. Conclusions are drawn that there has been an overemphasis on the technical aspects of algorithm development and modelling whilst important aspects such as usability, explainability and trust have been neglected. Suggestions have been made that the BN developers move away from the mindset of "why does it fail" towards "how to succeed". The review cites [138] "The key to this is getting the right information, to the right people, in the right format, through the right channels, at the right times to enhance decisions" as a way to combat this problem.

This review differs in looking at the construction methods of Bayesian networks in healthcare and the applications in classification tasks. One of the main areas has been risk prediction and classification of diseases in healthcare.

3.3 Classification of Literature and Summary Statistics

A total of 59 articles have been selected as part of this review of Bayesian network learning in the domain of healthcare. These articles cover structure learning using both statistical and score-based methods, other methods included are parameter and hybrid-based learning.

Articles outside of healthcare have also been included as these methods can also be applied to healthcare and may open up further research avenues.

Tables 1 and 2 show the year groupings of publications and the country of origin, the chosen articles span 1989 to 2021 and come from all over the world.

The application of Bayesian networks is covered in the section Applications in Healthcare and covers a variety of methods.

Feature selection is an important use and consideration when using Bayesian networks, the networks can perform the task of feature selection and can also benefit from features being filtered before the training begins.

Three main categories have been identified for the construction of the structure of Bayesian networks, these are constraint-based, scoring based and hybrid methods. Articles relating to these methods have been categorised in Table 3.3.

Search heuristics play a key role in exploring the search space of conditional dependencies, many such techniques have been developed over the years with varying results, each with its own merits. Table 3.3 lists the articles looking at these methods whilst [68] reviews a number of these methods. The ability to understand the performance of these networks whilst having a parameter on which to optimise the search solution is important, these are covered in the metrics selection.

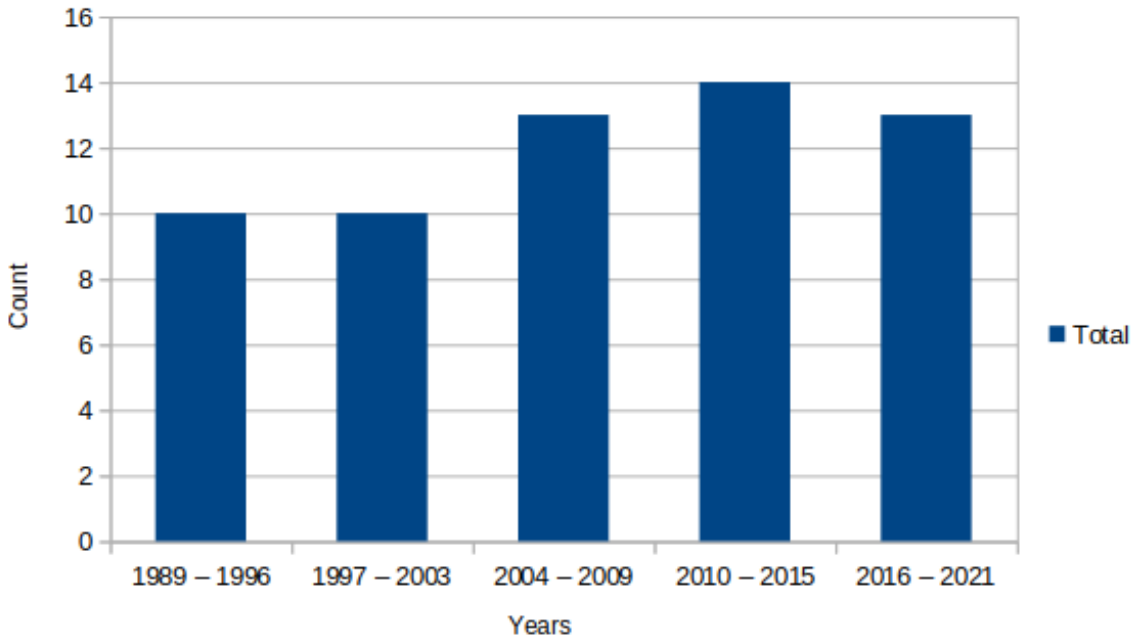


Fig. 3.4 Year of publication

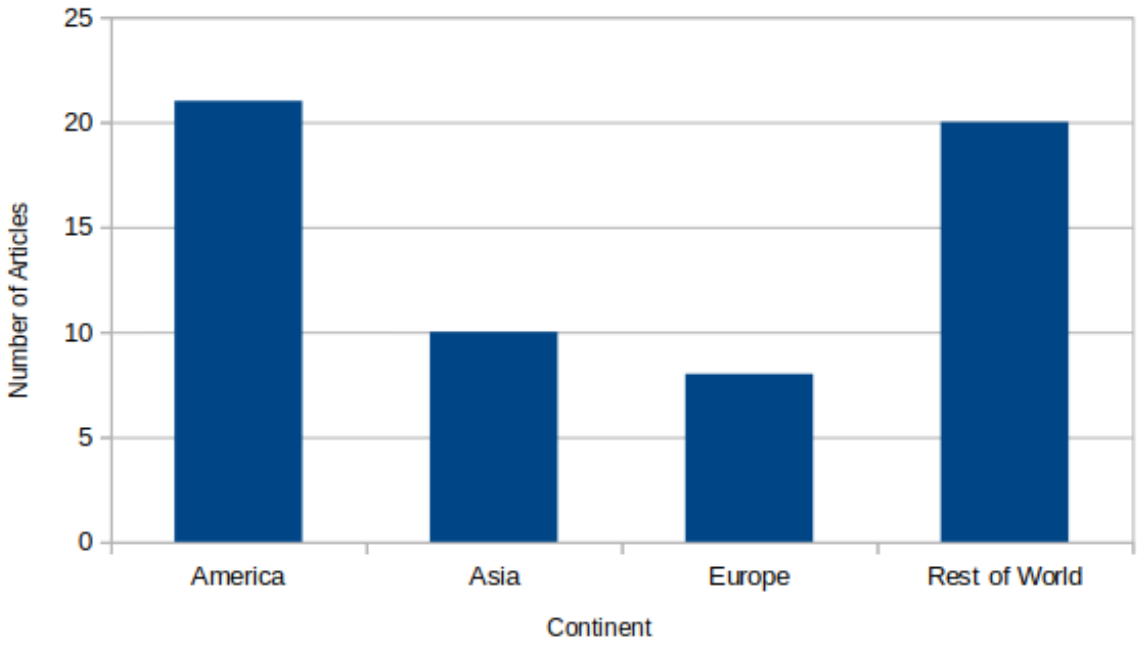


Fig. 3.5 Country of origin

3.4 Applications in Healthcare

There are many applications of Bayesian networks in healthcare this review focuses on Bayesian networks for classification in healthcare. Table 3.1 shows 10 of such journal articles or conference proceedings that were picked up in the Scopus search.

	Article	Total
Applications in health care	[69] [33] [1] [142] [133] [134] [148] [98] [11] [108]	10

Table 3.1 Healthcare articles

Calculating risk is important in health care, [69] [108] [11] focus on the use of the Bayesian network for risk prediction in healthcare. [69] looks into the use of audio samples with Bayesian networks to predict risks during children's activities, high accuracy rates of 97% have been achieved. The ability to link the relationships between disease and risk factors can be important [108] is an epidemiological study using Bayesian networks to model these conditional dependencies. Conducting an individual risk assessment in the healthcare environment has many uses, [11] assesses using Bayesian networks for decision models.

Imaging classification techniques for the detection and classification of optic nerve glaucoma have been developed by [1], these classification networks use a mixture of Self-Organizing Maps (SOMS) and Bayesian Networks. Scores of 0.86 for sensitivity, 0.8 for specificity and 0.913 AUROC have been reported.

Construction methods for Bayesian networks for epistasis detection are covered by [133], these methods include the x2 and G2 statistical tests for assessing the significance of epistatic interactions. Such tests are compared against Monte Carlo methods. Bayesian networks with the K2 search algorithm have been applied to the classification of lung cancer tumours by [148], and further improvements using hybrid feature selection techniques have also been applied.

Markov blankets offer a way to reduce the size of a network by finding the Markov boundary of a target node in a network. The research done by [142] to reduce feature subsets for high dimensional microarray cancer datasets shows better accuracy rates can be achieved by reducing the number of attributes for classification.

Large dimensional datasets can be problematic for machine learning algorithms, one solution to this presented in [134] looks to use an ensemble of Bayesian networks with a fuzzy clustering method to get around this. The paper presents a Bayesian fusion algorithm to utilize all sub-clusters learnt for better prediction across many attributes.

Research presented in [33] looks into Bayesian network methods to predict long-term health-related quality of life and comorbidity after bariatric surgery. Bayesian networks are compared with convolution neural networks (CNN) and multivariable logistic regression methods. It was found that Bayesian networks can outperform CNNs and multivariable logistic regression models. Different types of structure learning algorithms exist for Bayesian networks,[98] investigates the effectiveness of score-based, constraint-based and hybrid-based methods on demographic and health survey data.

3.5 Attribute Selection and Classification

Bayesian networks can benefit from feature selection to improve performance and also be used for feature selection by taking the Markov blanket of the predictor node in the network. In taking the Markov blanket only the relevant attributes are defined by the Markov boundary.

	Article	Total
Feature Selection	[142] [119] [134] [148] [14] [168]	7

Table 3.2 Feature selection articles

The approach taken by [142] [168] [134] [14] is the use of Bayesian networks for feature reduction, [142] [168] and [14] use Markov blankets to reduce the network size down to only the nodes needed to make classifications. A different approach is used by [134], subgraphs are created and used as an ensemble to reduce the dimensionality of the data.

Filter methods can be applied to pre-process data attributes so that only relevant data methods are used and thus save time by not training on redundant features. This is the approach employed by [119] and [148].

3.6 Bayesian Network Learning Type

There are three types of learning methods for building a Bayesian network structure: constraint-based, scoring-based, and hybrid methods [158]. Hybrid methods are a combination of constraint and scoring-based methods.

Constraint-based methods use statistical tests such as the chi-squared test to test for independence; A search strategy will use statistical tests to test pairs and groupings of variables for independence.

Scoring-based methods use scoring metrics such as BDeu, MDL, AIC and K2, to name a few. A search heuristic will find candidate solutions and assign them a score based on a scoring function. The search heuristic's objective function will either minimise or maximise the score based on the metric used.

Hybrid-based methods usually use a constraint-based method to find the initial set of graphs based on independence tests, reducing the number of solutions for scoring-based methods.

Search Heuristic	Construction Method	Articles	Total
K2	Constraint-based	[85]	1
	Scoring based	[44] [85] [38] [184]	4
	Hybrid	[1]	1
Scatter Search	Constraint-based		0
	Scoring based	[50]	1
	Hybrid		0
TABU Search	Constraint-based	[139]	1
	Scoring based	[134] [98] [159] [169]	4
	Hybrid	[119] and [14]	2
TAN	Constraint-based		0
	Scoring based	[69] [64] [35] [22]	4
	Hybrid		0
Greedy Search	Constraint-based		0
	Scoring based	[58] [36] [169]	3
	Hybrid		0
Hill Climbing	Constraint-based		
	Scoring based	[159] [129] [169]	3
	Hybrid		
Simulated Annealing	Constraint-based	[85] [68]	2
	Scoring based	[134] [159] [92]	3
	Hybrid		0
Ant Colony	Constraint-based		0
	Scoring based	[184]	1
	Hybrid		0
Grow Shrink Algorithm	Constraint-based	[159] [98] [85]	3
	Scoring based		
	Hybrid		
Others	Constraint-based	[133] [145] [142] [98] [159] [145] [85] [68]	8
	Scoring based	[134] [95] [184][30] [154] [65] [38] [129]	8
	Hybrid		

Table 3.3 Bayesian network learning type

3.6.1 Constraint-based Construction Methods

Markov blankets reduce the size of a Bayesian network by defining the Markov boundary. The Markov boundary contains only the relevant nodes related to the target node. The work done by [142] uses this principle to reduce the number of attributes in high dimensional datasets used in microarray cancer datasets. Testing the conditional independencies and selecting only the attributes restricted by a Markov boundary [142] creates an efficient Bayesian network with only the required attributes. In reducing the number of attributes [142] finds the efficiency and classification accuracy can be increased with microarray cancer datasets.

Constraint-based methods have not been as popular as scoring-based methods for searches conducted in healthcare with only 10 relevant papers vs 28 for scoring-based methods.

As with most constraint-based methods, statistical tests are used to evaluate the conditional independence between attributes, the most common method is the chi-squared method and is demonstrated in [68] [159] [134] [98] [133] [85] where it is used in the construction of the structure of the DAGs. The G-Test has been discussed in [98], whilst the B-statistic is covered by [133] with the Mote Carol permutation test being discussed in [159].

Markov blankets can be used to reduce the number of attributes required for classification, papers that have a focus on using constraint-based methods to construct these are [145] [142] [139] [68] and [159] In order to apply the statistical tests across all attributes in an efficient manner, a search strategy can be used. Strategies are applied to find the best combination of vertices and edges to represent the Bayesian network. Specific search algorithms can use both constraint-based methods or scoring base methods; here, we will be looking at constraint-based implementations.

The study undertaken by [98] compares constraint, scoring and hybrid methods for applications with the Demographic and Health survey data. The three constraint algorithms used in this study are Peter and Clark (PC), Incremental Association (IAMB) and Grow-Shrink (GS). The PC approach was slow and struggled with large datasets; whilst IAMB and GS performed better, they struggled with missing data values.

Learning Bayesian Networks with the bnlearn R package by [159] looks into both constraint and scoring-based BNs. These algorithms can also be used with parallel computing when used with the snow package, as shown in [159]. The constraint-based algorithms demonstrated are Grow-Shrink (GS), Incremental Association (IAMB), Fast Incremental Association, Interleaved Incremental Association and Max-Min Parents and Children (MMPC).

Numerous new Constraint-based algorithms have been developed to learn Bayesian networks and subsets of those networks known as Markov blankets. For example, a new algorithm called Total Conditioning (TC) and a variant of that called TCbw developed by [145] perform better than the PC algorithm with higher structural accuracy.

Another new Constraint-based algorithm proposed by [139] implements a TABU-based search using conditional statistical tests to improve on an initial graph. This algorithm restricts unfavourable moves of graph vertices to consider more favourable moves before trying less favourable moves again. This algorithm performs well on small datasets.

Four papers that came up in the search that focuses on constraint-based Bayesian networks in healthcare are [133] [142] [98] [134].

The paper by [133] focuses on applications in epistasis detection and compares Bayesian networks against random forests; whilst there was a successful implementation of Bayesian networks and random forests, no one method stood out with an advantage.

Research conducted in [142] looks into if Bayesian networks can be applied to microarray cancer datasets to identify genes that might anticipate the behaviour of this disease. Applying a Markov blanket model to reduce the number of attributes increased the model's accuracy.

Two papers by [85] and [68] have a focus on the independence test chi-squared. [85] compares a range of approaches to constructing a Bayesian network using local search, iterative local search and simulated annealing. In addition, [68] presents an improvement on the GSIMN algorithm called the Dynamic Grow-Shrink inference-based Markov network learning algorithm (DGSIMN). The new DGSIMN algorithm prevents unnecessary independence tests and yields up to 88% saving whilst achieving the same or better accuracy.

3.6.2 Scoring-based Construction Methods

Scoring algorithms work by assigning a score to the Bayesian network and searching for new solutions. The search strategy will work to alter a graph solution or create new solutions, and the objective is to minimise or maximise the score.

Scoring-based search strategies have been applied to healthcare in the following publications [69] [134] [148] [98] [108].

Audio classification for risk has been applied by [69] to children's activities using Bayesian networks, achieving an accuracy rate of up to 97%.

High-dimensionality datasets can cause issues for Bayesian networks. Research by [134] created a fuzzy ensemble of BNs to reduce the dimensionality when working with gene regulatory network microarray data. In addition, a new algorithm called weight-based structure algorithm to learn Bayesian networks in a fuzzy manner presented by [134] that helps with high dimensionality data. Another example of a Bayesian network application to microarray data is [148] for lung cancer detection and classification. Results from this study showed an 87.6% accuracy rate.

India's demographic and Health Survey (DHS) data was used to train Bayesian networks to find causal relationships in preventable diseases such as pneumonia and diarrhoea in childhood mortalities. This study was conducted by [98] who employed multiple training techniques from scoring, and constraint to hybrid-based methods.

Methods for training Bayesian networks for epidemiological studies have been developed by [108]. The study integrates medical domain knowledge and patient occupational history to assess the risk factors for developing a disease.

K2 Scoring Based Methods

The initial proposal of the K2 algorithm is presented by [44]. This paper covers the working of the algorithm in depth. Furthering the research into K2 by [85] and [38] compares and contrasts the K2 algorithm against local search, iterative local search and simulated annealing using the BDe and K2 metrics for scoring. K2 Scoring based search strategies publications are; [44] [85] [38] [184].

Further improvements are made to the K2 algorithm using ant colony optimization by [184]. In this research, two new algorithms are presented K2ACO and ChainACO. The study has found that Ant Colony Optimization (ACO) could work better for larger datasets with many attributes.

Scatter Search Scoring Based Methods

Scatter search was initially proposed by [76] in 2003 and takes an evolutionary approach. The search works from an initial set of solutions sorted into diverse and best solutions. These solutions are combined until better and more diverse new solutions are found. Scatter Search score-based strategies that have been published are; [50]

Discussed here is the scatter search implementation by [50] using the K2 scoring metric. The initial phase of this algorithm calls the diversification generation method to create solutions to be improved upon by the improvement method. Once the improvement method gets called, the solutions get improved by a local search. In order to improve the objective value, arc deletion is applied, which attempts to improve solutions. Both cyclic and acyclic graphs get placed into the population. Cyclic graphs are used as the diverse solution to be improved.

Tabu Scoring Based Methods

Tabu Search uses an adaptive memory to keep track of visited solutions so that the algorithm does not fall into a local optimum. Tabu score based strategies that have been published are; [134] [98] [159] [169].

A comparative-based study of Bayesian network learners conducted by [134] uses Tabu search amongst other scoring-based methods. In this study, a Tabu learning implementation in the R statistics package has been utilized. The paper focused on creating ensemble learners for greater accuracy, and the scoring function was the BDeu score.

Various implementations of Bayesian network search algorithms in R are covered by [159]. One such method is the Tabu search and gets compared against other methods. Another

such paper covering Tabu search in conjunction with other algorithms is [98] and discusses learning Bayesian networks from demographic and health survey data.

Most search algorithms for Bayesian networks search over the space of structures [169] take a different approach with Tabu search and searches over the space of orderings. This study shows that ordering-based search outperforms the standard baseline and is competitive against other search algorithms.

TAN Scoring Based Methods

Tree-augmented naive Bayesian (TAN) is a group of learning algorithms that fall into semi-naive Bayesian learning methods. It is to be noted that these methods only build partial Bayesian networks. The structure of these networks is tree graphs, with one parent node and many children. TAN works as a weighted maximum spanning tree to maximise the probabilities [69].

TAN score based strategies that have been published are; [69] [64] [35] [22]

Applications of TAN in risk classifications for child activities have been developed by [69]. This study looks at multiple Bayesian network methods to classify risks based on audio samples. The TAN method in the study achieved an accuracy of 99.92% and was the best-performing model for a few features compared with naive Bayesian and semi-naive Bayesian classifiers.

Two papers that benchmark TAN networks against other types of Bayesian networks are; [35] and [64]. The first [35] focuses on four types of BNs, these are Naive-Bayes, tree augmented Naive-Bayes (TANs), BN Augmented Naive-Bayes (BANs), and general (BNs). Both scoring and constraint methods are discussed and investigated for the construction of these networks.

The second paper [64] looks at the performance of TAN with naive Bayes and selective naive Bayes. The MDL score was used in the scoring of the networks. Using this score, though, the author believes that it may not have delivered the best results as it works on the global error and not the local error. In conclusion, the best-performing network was the TAN network.

A comprehensive review has been conducted in [22] that includes most major Bayesian network structures, including TAN, as well as different scoring methods.

Greedy Search Score-based Methods

The greedy search algorithm seeks to find the optimal solution at each graph step. Greedy search score based strategies that have been published are;[58] [36] [169]

A new and novel algorithm is presented by [58] that performs an ordering-based search using a greedy search approach. The algorithm has shown better performance than most other BNs in a similar class on limited datasets.

Learning equivalence-based structures using a greedy search is presented by [36] and uses the BDeu scoring for the BNs. The research shows that searching for Bayesian equivalent classes instead of individual structures yields better performance and accuracy.

[169] is a greedy hill-climbing TABU search and is discussed in the TABU scoring-based methods section.

Hill Climbing Score-based Methods

Hill Climbing score based strategies that have been published are; [159] [129] [169]

In a study on weighted learning of Bayesian networks for gene regulatory networks, a fuzzy ensemble of clustered Bayesian networks has been implemented and tested on a variety of BNs. One such learning strategy was the Max-Min Climbing algorithm.

[169] is a greedy hill-climbing TABU search and is discussed in the TABU scoring-based methods section.

R implementations of Bayesian network learners have been investigated by [159], one such learner is the Hill-Climbing algorithm. This research also looks at the varying structures different algorithms create.

Hill climbing is benchmarked against OR search in [129], whilst the best parameters were investigated for both searches, OR search outperformed hill climbing in this study.

Simulated Annealing Score-based Methods

Simulated Annealing is a stochastic algorithm. In a general manner, the SA algorithm adopts an iterative movement according to the variable temperature parameter, which imitates the annealing transaction of the metals [55].

Simulated Annealing score based strategies that have been published are; [134] [159] [92]

Multiple algorithms have been trained for a weighted ensemble of networks in [134]. One algorithm type was the simulated annealing structure learning. This learner was trained with the Weka software, and the Bayes score was used. Whilst not the worst-performing algorithm against the others used, it was not the best.

A publication that demonstrates the simulated annealing structure learning algorithm is [159]. Simulated annealing is then benchmarked against other score-based learners.

The working of simulated annealing is outlined and explained by [92], an in-depth look at the simulated annealing is conducted in this paper.

Other Scoring Algorithms

This section will now discuss other learning algorithms found during the search.

A study of multiple Bayesian learners for clustering has been presented in [134] looking at Bayesian network methods for classification in gene regulatory networks. However, two algorithms in this study remain to be discussed: Chow Liu Tree (CLT) and Simple Structure Learning (SSL).

Chow Liu Tree (CLT) [41] is described [134] as a score-based algorithm that finds the best tree representation of the graph. Weights get fixed to vertices based on mutual information. The study showed CLT was one of the lower-scoring algorithms for the datasets used in the study.

Simple Structure Learning (SSL) [130] is described [134] as a scored-based algorithm that finds the globally optimal structure for the BN. It computes all the scores for all variable pairs to find the best parent nodes and then the best networks. SSL was one of the best-performing algorithms in the study.

Greedy Equivalent Search and Fast Greedy Equivalent Search have been used to construct Bayesian networks in a study looking at demographic and health survey data [98]. These algorithms were compared to others to find the best construction heuristic. These algorithms have been compared against IAMB, MMHC, PC, RSMAX2, Saiyan and TABU.

Elephant Swarm Water Search Algorithm capabilities to construct a Bayesian network are demonstrated by [95], the algorithm works on a score and search technique.

Edges are deleted, reversed, moved and re-inserted to find the optimal solution. Finally, the algorithm is compared against simulated annealing and greedy search using the BDe score.

Work on the Ant colony algorithm is covered by [184]. Two novel algorithms based on ant colony are presented in [184], the first ChainACO and the second K2ACO, both search through the space of orderings. ChainACO used chain structures to reduce computational complexity, but K2ACO explores the richer structures.

The Immune algorithm is presented in [30]. This algorithm imitates the process of vaccination in immunology and aims to reduce search time. The study found that the time to search for network solutions was indeed reduced, but there is scope for further research to improve this algorithm.

A novel approach to using particle swarm optimization whilst allowing cyclic arcs is undertaken by [154]. Cyclic arcs are allowed to exist as a candidate solution but are given a

poor fitness value. The result reduced the number of objective function evaluations required to coverage on a target solution.

One approach to speeding up the search for structure(s) for a Bayesian network is to restrict the search space. The Sparse Candidate is an algorithm that does just that. The research in developing this algorithm by [65] finds that by restricting the parents of each variable, a significantly improved search speed can be achieved whilst still achieving quality solutions.

K2rev and K2opt are presented as part of a review of [38] Bayesian network search methods. K2opt was found to have obtained the lowest structural difference compared with other algorithms in the review. K2rev is the same as K2opt, but with revered ordering, the results were on par with local search, iterative local search and annealing.

Optimal Reinsertion (OR Search) has been applied to searching for the optimal Bayesian network structure by [129], by removing and reinserting arcs, the algorithm finds the optimal solution. Scoring was handled by the BDEU score for each of the networks. The results show that two orders of magnitude speed up on tested datasets.

3.7 Hybrid Construction Methods

Hybrid methods use a combination of construction methods to find the best Bayesian network. For example, many networks such as Tabu use a construction heuristic to build the initial graph before using scoring based combined with changing the arcs to search for a better solution. Construction heuristics used in the past have commonly used statistical tests such as the chi-squared test to find conditional independence between attributes.

Methods used in research by [1] use the chi-squared test to improve the performance of the K2 algorithm by selecting only variables with a good score and ordered by score value. Although it was stated in the research that inadequate ordering may lead to poor results and that ordering with the chi-squared test can ease this problem, the test also offers better initial variables on which to train with the K2 algorithm.

Two Tabu methods that utilization a conditional independence approach to finding the initial structure of the graph are [119] and [14], the two approaches differ in the fact that [119] used general conditional independence tests and [14] uses the chi-squared test. Once the initial graph is found, a scoring-based Tabu approach searches for the best structure.

In the research conducted in [134] and [98] using the hill-climbing algorithm, both papers use a constraint-based method for constructing the initial graph. Once the initial graph is found, they then apply a score and search-based hill-climbing method to search for the best graph.

3.8 Metrics for Evaluation of Bayesian Network performance

Many metrics exist for evaluating the performance of classifiers in general and those targeted at Bayesian networks and probabilistic models. Figure 3.6 shows the distribution of such metrics across chosen publications. These publications have been chosen in Bayesian networks for classification, and Figure 3.6 reflects the chosen metrics in this area, Table 3.5 details which of these metrics are used in each publication.

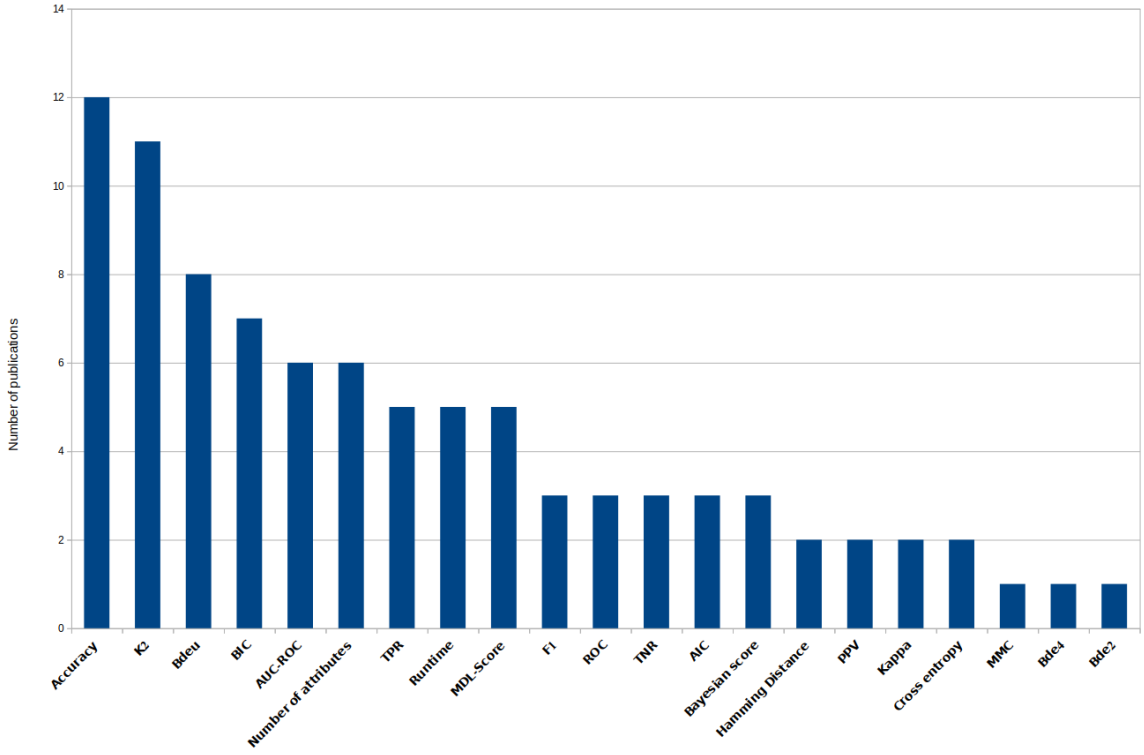


Fig. 3.6 Metrics used

Accuracy and the K2 score are the most prelevant metrics used in studies and reviews, followed by the BIC score and AUC-ROC. Although accuracy is a favoured metric, it can be sensitive to an imbalanced dataset. Not all studies featured in this review have stated if the datasets were balanced; most do not state this or show balance in classes.

The True Positive Rate (TPR), True Negative Rate and Precision or Positive Predictive Value (PPV) can be helpful metrics alongside accuracy to judge the effectiveness and performance of the classification model.

Metric	Full name	Description
Accuracy	Accuracy	The fraction of predictions the model got right
K2	K2	Evaluation method of how well a Bayesian learned from a dataset
Bdeu	Bayesian Dirichlet equivalence score	Uniform prior over the parameters of each local distribution in the network
BIC	Bayesian Information Criterion	An estimate of the performance of the Bayesian network on new data
AUC-ROC	Area under the ROC Curve	Aggregate measure of performance across all possible classification categories
Number of attributes	Number of attributes	The number of attributes in the dataset
TPR	True positive rate	The number of true positives over the true positives and false negatives
Runtime	Runtime	The total runtime of training a model
MDL-Score	Minimum Description Length	A scoring method from information theory that is equivalent to BIC
F1	F1	Combination of precision and recall scores to form one score
ROC	Receiver operating characteristic curve	Measures how well a model can classify
TNR	True negative rate	Measures the true negatives over true negatives plus the false positives
AIC	Akaike information criterion	Estimates prediction error and the quality of models, a lower score is better
Bayesian score	Bayesian score	Score of a Bayesian network that maximizes the posterior probabilities
Hamming Distance	Hamming Distance	Structural Distance Measures for Causal Bayesian Network
PPV	Positive predictive value	Getting a positive result will result in a true positive result
Kappa	Cohen's Kappa coefficient	Measure of inter-rater reliability for qualitative categories
Cross entropy	Cross entropy	Measure of the difference between two probability distributions
Bde4	Bayesian Dirichlet Score	Score equivalent Dirichlet posterior density
Bde2	Bayesian Dirichlet Score	Score equivalent Dirichlet posterior density

Table 3.4 Metric glossary

Other factors such as the number of attributes in the final model are essential in Bayesian networks as these represent nodes (parameters) when used alongside the BIC/AIC score to prevent overfitting [167]. The number of attributes alongside the number of edges gives a good understanding of the number of casual relationships in the model and how complex the model is.

Area Under the Curve (AUC) paired with the Receiver Operating Characteristic (ROC) is in the top five metrics used. An interesting point to note is that the AUC for the Precision-Recall Curve is never reported in all the literature.

Runtime is the 8th most used metric and can be important when looking at the number of attributes and scoring metrics to understand the model's performance. However, a more significant number of attributes may lead to more prolonged search and score times.

Scoring metrics featured are K2, BDeu, BIC, MDL, Bayesian score, BDe4, BDe2. The top 4 network scoring metrics were K2, BDeu, BIC and MDL.

Metric	Article	Total
Accuracy	[69] [33] [1] [142] [119] [134] [148] [110] [14] [139] [168] [22]	12
K2	[1] [148] [110] [50] [118] [98] [44] [85] [38] [184] [154]	11
BDeu	[58] [85] [65] [36] [169] [38] [129] [184]	8
BIC	[1] [134] [98] [63] [184] [95] [30]	7
AUC-ROC	[33] [1] [133] [110] [14] [139]	6
Number of attributes	[142] [119] [110] [58] [85] [169]	6
TPR	[33] [1] [133] [119] [98]	5
Runtime	[142] [58] [118] [85] [129]	5
MDL-Score	[64] [36] [35] [63] [154]	5
F1	[119] [134] [98]	3
ROC	[33] [119] [14]	3
TNR	[33] [1] [119]	3
AIC	[69] [36] [95]	3
Bayesian score	[36] [39] [38]	3
Hamming Distance	[134] [98]	2
PPV	[119] [98]	2
Kappa	[119] [14]	2
Cross entropy	[85] [38]	2
MMC	[148]	1
Bde4	[85]	1
Bde2	[38]	1

Table 3.5 Articles metrics used

3.9 Statistical Tests for Independence

Constraint-based methods rely on conditional independence tests to construct the Bayesian network graph. The graph represents conditional independences. These tests are carried out with statistical tests such as the chi-squared test with an alpha of 0.005. All tests used in the literature found during the search are detailed in Table 3.7.

Statistic type	Description
Chi-Squared	Used with contingency tables when testing for the independence of two categorical variables.
G-Test	Testing for statistical significance when the sample size is too large for chi-squared.
B-Statistic	Testing for statistical significance
Kernel density estimation	Non-parametric method to estimate the probability density function
Z-test	Test whether two population means are different when the variances are known

Table 3.6 Stats test glossary

Statistic type	Article	Total
Chi-Squared	[1] [133] [14] [98] [68]	5
G-Test	[133] [98]	2
B-Statistic	[133]	1
Kernel density estimation	[119]	1
Z-test	[145]	1

Table 3.7 Statistical tests by article

3.10 Bayesian Network, Machine Learning and Statistics Books

We have identified five main subject areas that are useful for understanding and developing Bayesian networks. These books were identified through Cardiff University library services, Google searches, conferences, Scopus, Packt publishing and Amazon. These are; Probability, statistics, Bayesian networks, Graph search heuristics and Data Science and Machine Learning. Books that have been listed in Table 3.8 fall into one of these five areas and provide valuable information in the pursuit of understanding and development of Bayesian networks.

Book Name	Author	Citation
Python for Probability, Statistics, and Machine Learning	José Unpingco	[172]
An Introduction to Statistics with Python: With Applications in the Life Sciences	Thomas Haslwanter	[84]
Algorithms in a Nutshell	George T. Heineman, Gary Pollice, Stanley Selkow	[86]
Probabilistic Deep Learning: With Python, Keras and Tensorflow Probability	Oliver Durr, Beate Sick, Elvis Murina	[52]
Statistics in a Nutshell: A Desktop Quick Reference	Sarah Boslaugh	[26]
Bayesian Statistics The Fun Way	Will Kurt	[100]
Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference	Cameron Davidson-Pilon	[49]
Practical Statistics for Data Scientists	Peter Bruce, Andrew Bruce, Peter Gedeck	[29]
Data Science from Scratch: First Principles with Python	Joel Grus	[79]
Data Mining: Practical Machine Learning Tools and Techniques	Witten, I.H. and Frank, E. and Hall, M.A. and Pal, C.	[183]
All of Statistics: A Concise Course in Statistical Inference	Larry Wasserman	[178]
Think Bayes: Bayesian Statistics in Python	Allen Downey	[51]
Numerical Python	Robert Johansson	[94]
Mastering Probabilistic Graphical Models Using Python	Ankur Ankan, Abinash Panda	[7]
Artificial Intelligence: A Modern Approach, Global Edition	Stuart Russell, Peter Norvig	[153]
Scatter Search Methodology and Implementations in C	Manuel Laguna, Rafael Martí, Rafael Cunquero Martí	[103]
Machine Learning Algorithms From Scratch with Python	Jason Brownlee	[27]
Imbalanced Classification With Python	Jason Brownlee	[28]

Table 3.8 Reviewed books

3.10.1 Probability and Statistics

Statistics in a Nutshell [26] and All of Statistics [178] cover all main areas of statistics in a reference-size book that is easily accessible to look up most of the critical theory needed for working with probability and Bayesian networks.

Think Bayes [51] and Bayesian Statistics The Fun Way focus [100] on Bayesian statistics with a practical application in R and Python. Think Bayes [51] uses the Python programming language and common data science libraries whilst Bayesian Statistics the Fun Way [100] shows examples in R. The languages are not essential as the theory is covered without the need to understand programming.

Numerical Python [94] covers a lot of mathematical subjects and is not limited to probability and statistics but also many other core topics such as calculus and algebra, which are essential for machine learning applications. In addition, visualizing data is an important topic when trying to understand mathematics and data, which has a chapter dedicated to this subject.

An Introduction to Statistics with Python [84], Python for Probability Statistics and Machine Learning, Practical Statistics for Data Scientists explain common topics such as hypothesis testing, distributions, multivariate analysis and more. All topics are explained well and show how to use all libraries in Python to carry out typical statistical work whilst also touching on machine learning.

3.10.2 Bayesian Networks

Artificial Intelligence - A Modern Approach [153]: covers a wide range of topics. Chapters 13 - 15 offer a good grounding and introduction to Bayesian networks and the theory that underlines Bayesian networks. These chapters are one of the most complete introductions to the theory in the books presented here.

Data Mining - Practical Machine Learning Tools and Techniques [183]: explains the theories behind Bayesian networks but offers practical labs within the WEKA workbench. All information and examples relating to Bayesian networks are presented in Chapter 9.

Probabilistic Deep Learning [52]; is an entire book dedicated to Bayesian methods. Although the information in this book is aimed at Bayesian Neural Networks, many of the techniques can be applied to Bayesian networks. For example, chapter 5 introduces the TensorFlow probability framework, which is built on top of the TensorFlow framework. TensorFlow probability can be used to build Bayesian networks, and the introduction to this framework given in the book will allow the reader the grounding to build these networks.

Chapter 7 is another helpful chapter discussing Bayesian-based learning with TensorFlow probability.

Bayesian Methods for Hackers [49]: Probabilistic Programming and Bayesian Inference introduce two frameworks for building Bayesian networks, TensorFlow Probability and PyMC2/3. The book is backed up by a set of rich examples given in both frameworks, which are clear and in-depth. In giving examples of both frameworks, the reader can compare and contrast the frameworks on offer. The book is also well integrated with the TensorFlow probability documentation, which extends the examples given to offer an even more in-depth look at the framework. Many examples in the TensorFlow Probability documentation are taken from this book.

Mastering Probabilistic Graphical Models Using Python [7] is dedicated to building Bayesian networks in Python using the pgmpy framework. The pgmpy framework offers many distributions and learning algorithms to construct Bayesian networks from data and learn the parameters of distributions. The book also covers inference from models with an in-depth look at how different inference methods work and the math behind these methods.

3.10.3 Graph Search Heuristics

Algorithms In a Nutshell [86]; explains the working of all fundamental algorithms needed for building and searching a graph and can be used as a helpful reference when working with search algorithms. Chapters 5 to 7 are the most useful when working with graphs; functional code and diagrams are included for all standard search methods. The book lacks more advanced algorithms like Tabu search and Scatter search, but all the essential principles for searching are included.

Artificial Intelligence - A Modern Approach [153]; Covers a vast amount of searching theory with detail and depth in each area and is one of the most comprehensive books on the subject in this review. Chapters 3 to 12 will take most people from beginner to intermediate knowledge of this subject area.

Scatter Search Methodology and Implementations in C [103]; This book looks at how Scatter search works and how to implement it. The book gives examples of how to implement key components such as the Diversification method, Improvement method, RefSet, Subset, and combination method and tie these components together to form a scatter search. Even though the implementations are in C, they are clear enough to implement in any programming language. Different designs with the use of memory to prevent the trap of local optima with examples of Tabu search, explicit memory and attribute memory are given. Important context is given in Chapter 7 with other population-based approaches such as genetic Algorithms

and Path Relinking. The book concludes the final chapters with demonstrable applications of Scatter search.

3.10.4 Data Science and Machine Learning

Numerical Python [94]; offers all the core concepts needed to understand data science and machine learning. All examples are given with the math and implemented in Python. Numerical Python is one of the most comprehensive books for mathematics with Python and covers everything from algebra to calculus. In addition, the book teaches about statistics, probability and machine learning, all with Python. It does not focus on just data science and machine learning but the tools in mathematics and machine learning for science in general.

Dr Jason Brownlee has written a series of books on mathematics, machine learning and data science. Two of the books most relevant to this review are Machine Learning Algorithms From Scratch [27] with Python and Imbalanced Classification With Python. Machine Learning Algorithms From Scratch with Python explains developing, testing, and evaluating machine learning algorithms. Python and Imbalanced Classification With Python goes through visualizing data, understanding the imbalances and methods for correcting or compensating for imbalanced data.

Data Science from Scratch [79] and Practical Statistics for Data Scientists [29] aim to build a skill set for hands-on data science in a methodical process. Data science from scratch focuses on mainly machine learning and AI whilst exploring subject areas such as NLP and network analysis. On the other hand, practical Statistics for Data Scientists takes a more traditional approach in looking at the theory of statistical methods and applications.

3.11 Discussion and Limitations

Risk assessment and prediction have been the most effective application of Bayesian networks in healthcare. Bayesian networks encapsulate uncertainties with graphical networks using probabilistic distributions to give the modeller and clinical staff intuition and understanding of those risks. Other uses of Bayesian networks in healthcare are the classification of diseases and the interaction of risks and diseases.

Attribute selection and dimensionality reduction can help improve the performance of a model both in terms of training time and accuracy. For example, Bayesian networks exploiting conditional dependencies using Markov blankets can significantly reduce the number of attributes required to predict the target classification variable. Other methods

include filtering and ranking variables before passing them to the Bayesian network learning algorithm to improve learning times and, in some cases, accuracy.

There exist many publications on learning methods for the structure of Bayesian networks, but most are score-based methods. Constraint-based methods have had limited research conducted and are normally paired with a score-based approach. These pairings form the hybrid methods with constraint-based methods supporting scoring methods to improve performance or generate the initial solution to be improved by the scoring-based method.

Most metrics are tailored towards the score-based approach, such as K2 BDeu, and MDL-Score. Accuracy most one of the most commonly used metrics but can be problematic with imbalance datasets, and most publications did not discuss how well-balanced the datasets were.

Constraint-based methods use statistical tests between pairs or grouping of variables to find the partial structures of the graphs. These partial structures are tested and combined with a search strategy. The most common statistical test was the chi-squared test, followed by the G-test. Other tests such as the B-statistic, Kernel density estimation and Z-test also came up in the literature.

Of the books that have been reviewed for the ability to support, understand and implement Bayesian networks, the following facets were decided on; Probability and Statistics, Bayesian networks, Graph search heuristics and Data Science and Machine Learning. These books covered the theory behind Bayesian networks with C, Java, R, and Python implementations.

All domain knowledge has been presented in the other reviews listed at the beginning of this review. In addition, the papers and books also contain the domain knowledge required to understand this topic.

3.12 Conclusions

Tremendous advancements have been made in Bayesian network machine learning, but this area will always be an NP-Hard problem. As a result, the most attention has been given to scoring-based methods, with algorithms like K2, TABU search, TAN, Greedy Search, and Hill Climbing being given the most focus.

Constraint-based methods still lag in development in contrast to scoring-based methods. One of the most underdeveloped areas is Scatter Search which only has one paper dedicated to a scoring-based approach. This Scatter search scoring-based method still needs further improvement but lays essential groundwork for future research. Further research into constraint-based scatter search methods is greatly needed and can potentially move the body of knowledge forward.

Chapter 4

A Scatter Search Metaheuristic for Bayesian Networks

4.1 Introduction

Bayesian networks offer a graphical representation of the relationships between variables that is interpretable by non-machine learning specialists. The graphical representation allows all stakeholders to engage with model development and understanding of findings. To speed up the development of these Bayesian network models machine learning can be used to build initial models for further discussions and understanding of the problem domain. Trained models by machine learning may already be good enough to be used once validated by domain experts.

A new approach using scatter search as a constraint-based method using statistical tests to construct the structure of the graphs is presented here. The design of the scatter search algorithm allows the searching of both local and global spaces. Local searches on the initial dependencies between attributes and then global searches combine the results of the local searches to build up a complete Bayesian network. Getting stuck in local optima can be a problem but by performing many smaller searches of local space we avoid getting stuck in one space.

Once these smaller solutions have been combined into many large networks the network size can be reduced by using Markov blankets. These small compact networks then have the ability to perform computationally efficient inference for the problem domain and in the case of this research classification is the end goal.

4.2 Related Work

Scatter search is an evolutionary meta-heuristic that works on a population of solutions. These are combined in order to find better solutions. To choose which solutions are combined, the concept of diversity has been introduced [76] [75]. Since the introduction of diversification, researchers have found many applications for the scatter search. Applications of the algorithm in machine learning have been done by [53] in detecting fraud and feature selection [116]. A scatter search algorithm for Bayesian network learning has been devised by [50] who, in contrast to the work presented in this chapter, use the scoring-based search paradigm (see chapter 3). The algorithm developed here aims to build a constraint-based scatter search learning method for the Bayesian network from statistical tests.

Explainable AI (XAI) is important so the decisions, classification or regression outputs of AI models are understood. Explaining how AI came to the outputs it did is important in building trust with end users and meeting compliance requirements. Furthermore, policy-makers and government bodies such as Select Committee on Artificial Intelligence [141] and EU High-Level Group on AI [43] want a clearer understanding and trust of AI models to turn black box learning into white box learning. The Royal Society has published a policy briefing document [162] explaining the importance of explainable AI which outlines the importance of Giving users confidence in the system, Safeguarding against bias, Meeting regulatory standards, policy requirements and Improving system design.

Methods of explainable AI differ depending on the AI model in question. There are many methods from heat maps for neural networks to producing rule-based diagrams for other classifiers. A large systematic review of most of the techniques has been provided in a literature review called Explainable Artificial Intelligence: a Systematic Review [173]. A more concise review of machine learning models that are easy to understand without the need for special tools is conducted in Comprehensible classification models: a position paper [62], one such method covered in this is Bayesian networks. It was found that Bayesian networks could offer an explanation of the relationships between variables and also the probabilities leading to the outcomes which can be represented by contingency tables. The author also found that the Bayesian networks can become complicated when there are many dependencies but this can be overcome by the use of Markov blankets. It was found that Markov blankets can greatly reduce the size of the network.

4.3 Methods

We provide a formal description of classification problems before we introduce the different approaches. Let $\mathscr{I}$ denote a set of individuals (e.g. patients) and let $\mathscr{D}$ denote the set of outcome measures for classifying these individuals. For each individual $i \in \mathscr{I}$, we observe a set of attributes $\mathscr{A}$ and the true outcome $d_i \in \mathscr{D}$. Let $\mathscr{V}_a$ denote the set of possible values for attribute $a \in \mathscr{A}$ and let $v_{i,a} \in \mathscr{V}_a$ denote the value of attribute a for individual i. We wish to predict d_i of individual i given the individual's values $v_{i,a}$ for each attribute $a \in \mathscr{A}$. Some attribute values $v_{i,a}$ may be missing. In the computational study, we briefly discuss how we handle these missing values (i.e., treating 'unknown' as a separate value). In this *supervised learning* problem, we assume the availability of labelled training data from many other individuals $j \in \mathscr{I} \setminus i$ whose attribute values $v_{j,a}$ and outcome d_j are known. This training data is used to learn a classification model which is then used for the prediction.

Informative attributes are defined as Xi... i.e. (X1, X2, X3, X4...) and the attribute to be predicted is labelled Y1 in our models.

Bayesian networks shown in figure 4.1 show the conditional relationships between attributes represented by graph edges and the probability distributions represented by nodes. The relationships between nodes will be discovered by using the chi-squared test over the attributes guided by the scatter search.

The Bayesian network shown in Figure X which gives the probability of a heart attack has 5 Variables Exercise (E), Smokes (S), Blood Pressure (BP), Cholesterol (Chol) and Attack (A). Given the (E), (S), (BP) and (Chol) we can see the probabilities of a heart attack (A).

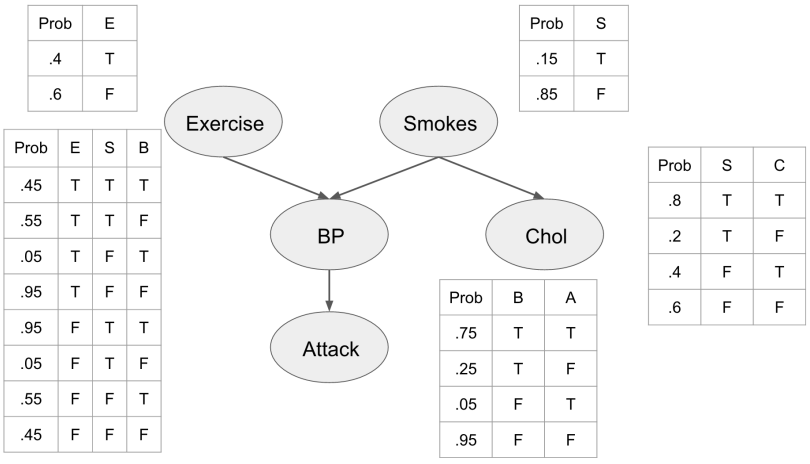


Fig. 4.1 Bayesian network [18] [19]

Once the attributes and relationships have been found the size of the network will be reduced using a Markov blanket. The Markov Blanket is a compact representation of the casual graphical network as shown in figure 4.2.

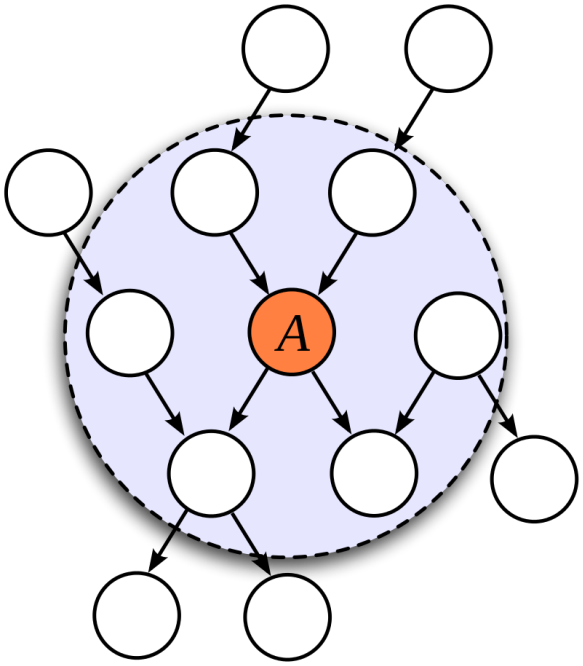


Fig. 4.2 Markov blanket [180]

The Markov Blanket contains the predictor variable that will be represented by Y1 in our research but is labelled as A in figure 4.2. This predictor variable (A) has its parents and children and the parents and children of those nodes. In this compact representation of our network, only the nodes needed to perform inference are present.

4.4 Scatter Search Heuristic

The scatter search algorithm originally proposed by [73] gives us a guided search strategy in which to perform statistical tests over attributes in the dataset. Implementing this algorithm we can start with a local search to find the initial values and then combine these solutions to build up the Bayesian network.

4.4.1 Scatter Search Heuristic Components

Scatter search works by starting with an initial set of very simple solutions, these solutions are combined in an iterative loop and added into a set of combined solutions. These new

combined solutions are added based on a chosen score. Diverse solutions which are defined as not the best solutions are also allowed. The set of combined solutions is split into good solutions and diverse solutions. The iterative loop continues until no new solutions can be found with the best solutions extracted from the set of combined solutions.

Algorithm 1 Scatter search main algorithm

```
1:  population_set = ∅
2:  ref_set = ∅
3:  diversification(α=0.05)
4:  populate_refset(number_of_target_solutions, number_of_diverse_solutions)
5:  last_refset_update := 0
6:  number_of_iterations := 0
7:  while last_refset_update < 𝒳 ∧ number_of_iterations < 𝒴 do
8:      last_refset_update := last_refset_update+1
9:      number_of_iterations := number_of_iterations+1
10:     subset_G := subset_generation_method()
11:     𝒢 := combination_method(subset_G)
12:     if add_solution_to_subset(𝒢, ref_set) then
13:         last_refset_update := 0
14:     end if
15: end while
```

Diversification Method

The implemented diversification method defines good solutions as $X_i \rightarrow Y_i$ with informative attributes pointing to the attribute to be predicted. Diverse solutions are among the informative attributes $X_i \rightarrow X_i$. No attribute can point to itself to cause a cyclic link.

All attribute scores are based on p values with a good "score" being less than 0.05, if this criterion is met the solution is added to the population. During this stage, only one-to-one relationships are created.

The objective of the diversification method is to create a population set of diverse solutions. Those solutions are created by using the chi-square test on pairs of attributes, these attributes become the nodes. Different strategies of searching for suitable node pairs can be employed, for this implementation, we test all attributes against the classification attribute and then all attributes paired together. These solutions will be combined together further in the process loop to maximize the prediction accuracy.

Algorithm 2 Diversification from target

```
1: function DIVERSIFICATION_FROM_TARGET(α)
2:     target_node := Y1
3:     nodes := {X1, X2, X3, ... }
4:     for all 𝒱 ∈ nodes do
5:         p_value = chi_test(α, {𝒱, target_node})
6:         if p_value ≤ α then
7:             𝒢 := ({𝒱,target_node},{(𝒱,target_node)})
8:             add_solution_to_subset(𝒢, population_set)
9:         end if
10:    end for
11: end function
```

Algorithm 3 Diversification comparing nodes

```
1: function DIVERSIFICATION_COMPARING_NODES(α)
2:     target_node := Y1
3:     nodes := {X1, X2, X3, ... }
4:     for all node1 ∈ nodes do
5:         for all node2 ∈ nodes do
6:             if node1 ≠ node2 then
7:                 𝒢 := ({𝒱,target_node},{(𝒱,target_node)})
8:                 p_value = chi_test(α, {𝒱, target_node})
9:                 if p_value ≤ α then
10:                    add_solution_to_subset(𝒢, population_set)
11:                end if
12:            end if
13:        end for
14:    end for
15: end function
```

Algorithm 4 Diversification

```
1: function DIVERSIFICATION(α = 0.05)
2:     diversification_from_target(α)
3:     diversification_comparing_nodes(α)
4: end function
```

Population set

The population set will accept any solution with a p-value of less than 0.05 and any solution must be a one-to-one relationship with two nodes. The direction doesn't matter as long as the solution is directed. There is no limit on the population size in our implementation and the list is ordered according to the best p-values.

RefSet

The RefSet holds the solutions to be improved upon. A limit is placed on the size of the RefSet and this limit is a parameter of the algorithm that can be changed when optimising the parameters to find the best-performing value. Literature suggests this be smaller than the population set.

RefSet Update Method

The RefSet update method will attempt to pick an equal number of both good and diverse solutions from the population set. Good solutions take priority over diverse solutions.

To populate the RefSet, two types of solutions are needed: high-quality and diverse solutions. We define high-quality as a low p-value solution that is directly connected to the attribute to be classified i.e. the target attribute or variable. The second type of solution is the diverse solution which we define as a solution that has at least 1 degree of separation from the target variable. The diverse solution p-value is not taken into account at this stage as all solutions in the set should have a p-value less than alpha.

Algorithm 5 Populate refset

```
1: function POPULATE_REFSET(number_of_target_solutions, number_of_diverse_solutions)
2:     i=0
3:     order_solution_set_by_p_value(population_set)
4:     for all 𝒢 ∈ population_set do
5:         i=i+1
6:         if i ≧ number_of_target_solutions then
7:             break
8:         end if
9:         if target_node ∈ 𝒢.nodes then
10:            add_solution_to_set(𝒢, ref_set)
11:        end if
12:    end for
13:    i=0
14:    for all 𝒢 ∈ population_set do
15:        i=i+1
16:        if i ≧ number_of_target_solutions then
17:            break
18:        end if
19:        if target_node ∉ 𝒢.nodes then
20:            add_solution_to_set(𝒢, ref_set)
21:        end if
22:    end for
23: end function
```

Subset Generation Method

A pair of solutions are picked from the RefSet to be combined, many parameters can be set here to improve the performance of the algorithm. Rules can be set to pick solutions that can be combined with the required nodes and edges to make the connections needed for a solution.

The subset generation method picks two random graphs from the RefSet, and these graphs cannot be the same graph. If the two graphs are the same, the loop continues until the graphs do not match.

Algorithm 6 Subset generation method

```
1: function SUBSET_GENERATION_METHOD
2:     G1 := get_random_solution_from_refset()
3:     G2 := G1
4:     while G1 = G2 do
5:         G2 := get_random_solution_from_refset()
6:     end while
7:     return {G1, G2}
8: end function
```

Combination Method

The combination method combines the pairs of solutions from the RefSet, a valid solution is one where the two graphs can be merged with connections between the two. The best-performing graph takes priority over the worst-performing graph with the secondary graph only giving features not already in the best-performing graph.

In order to find new solutions, the combination method combines the edges in the graphs passed to it to create a new solution. The solution is scored based on accuracy and passed to the RefSet Update Method. If the new solution meets the requirements to be added to RefSet, the add_solution_to_set method will add the graph to the RefSet.

Algorithm 7 Combination method

```
1: function COMBINATION_METHOD(G_i)
2:     G := compose(G_1, G_2)
3:     return G
4: end function
```

Add Solution to Solution Set Method

In order to ensure there are no duplicate solutions in the population or reference set, we define a function to only add a solution if it is unique. This function also checks the graph to ensure it conforms to a directed acyclic graph. If both of these requirements are met the graph is added to the correct solution set.

Algorithm 8 Add solution to solution set function

```
1: function ADD_SOLUTION_TO_SET(𝒢, solution_set)
2:     if 𝒢 ∉ solution_set then
3:         if is_directed_acyclic_graph(𝒢) is True then
4:             solution_set := 𝒢 ∪ solution_set
5:             return True
6:         end if
7:         return False
8:     end if
9: end function
```

Scatter search starts by generating a diverse set of solutions that will stop when no more solutions can be found, these solutions are placed in the population set. Once the population set is complete the RefSet is updated with a split of good solutions and diverse solutions. The split can be based on a ratio split or a set number of good solutions and diverse solutions.

A decision step is introduced to stop the algorithm if no new solutions are found after X number of iterations.

A subset of solutions are selected at random from the RefSet and passed to the combination method. The combination method attempts to combine these solutions to make a new solution, if the new solution has met the set criteria for a valid solution it is added back into the RefSet.

The RefSet updating method can restrict the number of solutions in the RefSet list. The RefSet updating method can clear out the worst solutions if the RefSet list becomes larger than X size keeping only good solutions in the list.

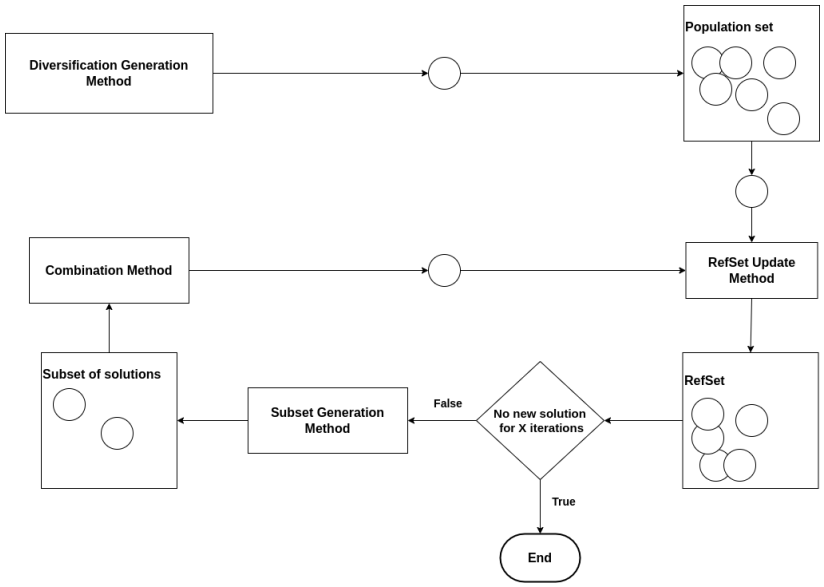


Fig. 4.3 Scatter search algorithm 1

Algorithm 9 Scatter Search main algorithm

```
1:  population_set = ∅
2:  ref_set = ∅
3:  diversification(α=0.05)
4:  populate_refset(number_of_target_solutions, number_of_diverse_solutions)
5:  last_refset_update := 0
6:  number_of_iterations := 0
7:  while last_refset_update < 𝒳 ∧ number_of_iterations < 𝒴 do
8:      last_refset_update := last_refset_update+1
9:      number_of_iterations := number_of_iterations+1
10:     subset_G := subset_generation_method()
11:     𝒢 := combination_method(subset_G)
12:     if add_solution_to_subset(𝒢, ref_set) then
13:         last_refset_update := 0
14:     end if
15: end while
```

4.4.2 Scatter Search Heuristic Algorithm Process

This section will run through the scatter search process step by step. The algorithm runs in an iterative loop until no more solutions can be found. All statistical tests are performed using

the chi-squared statistic. Once no more solutions can be found the algorithm terminates and a complete Bayesian network should exist.

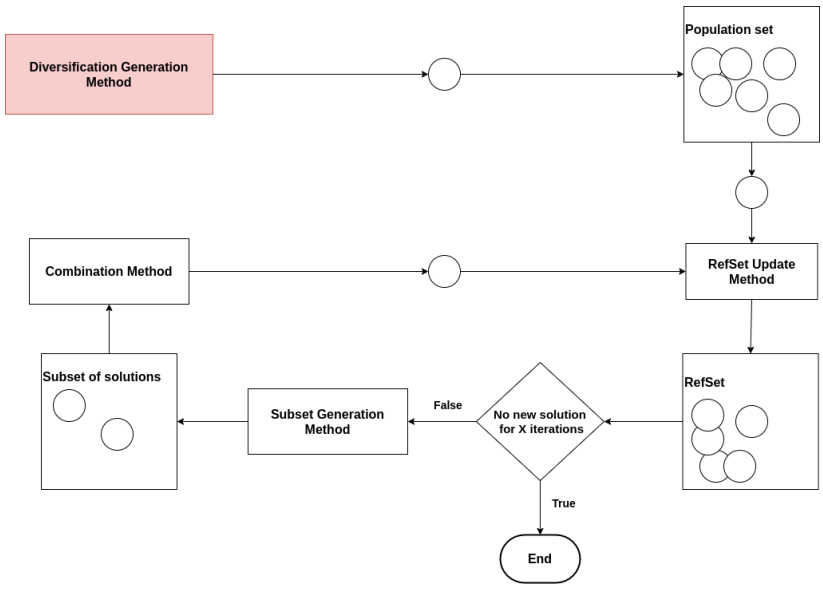


Fig. 4.4 Scatter search algorithm 2

The first step shown in figure 4.4 is to run the diversification method to generate the initial solutions. All solutions should have a p-value of 0.05 or less.

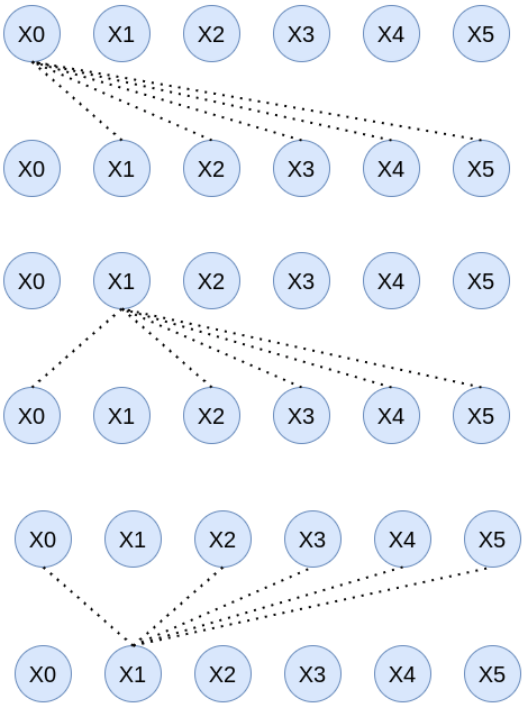


Fig. 4.5 Diverse attribute selection testing

Diversification will test for diverse solutions defined as not pointing back to Y1 the predictor attribute. All attributes that are not Y1 are compared against each other and tested with the chi-squared test for a value of less than 0.05.

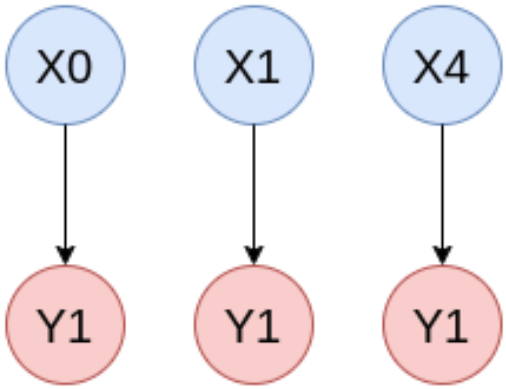


Fig. 4.6 Good graph solutions

All attributes are compared against Y1 with the chi-squared test for a value of less than 0.05. Given this step, only one to one relationship is being evaluated against Y1 the

computational time should be low using a simple statistical test on small to medium dataset sizes.

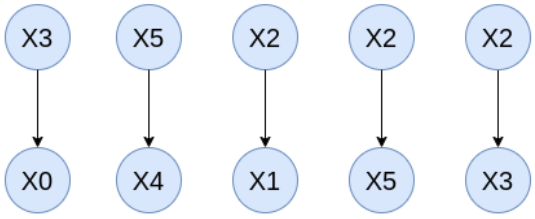


Fig. 4.7 Chosen diverse solutions

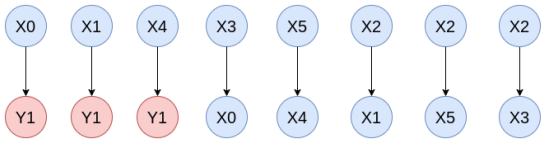


Fig. 4.8 Good and diverse chosen solutions

Once completed the population set should contain all one-to-one solutions with p-values of less than 0.05, these solutions are now ready to be moved to the RefSet.

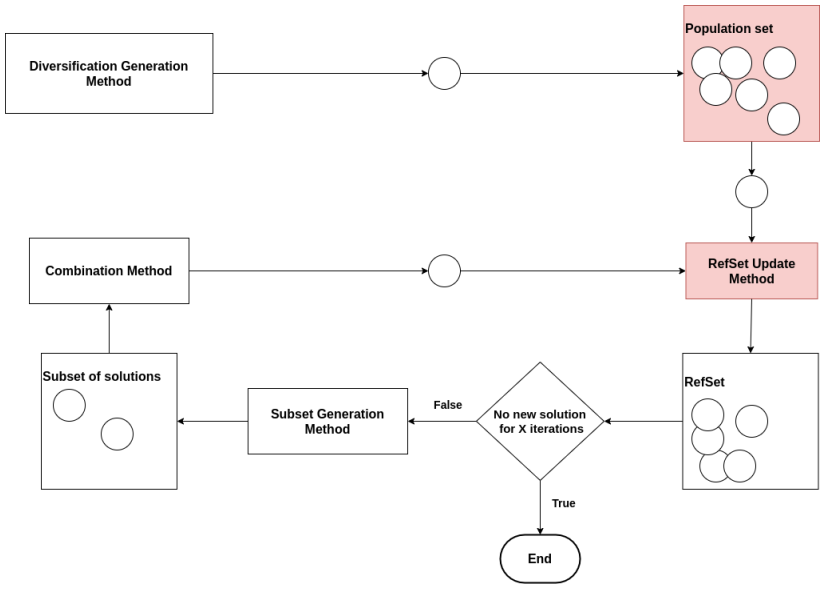


Fig. 4.9 Scatter search algorithm 3

The next stage is to update the RefSet with solutions from the population set. Solutions are split into good and diverse solutions. Diverse is not connected to Y1 and good solutions

are connected to Y1. These solutions are normally split based on a ratio of 50/50. Once split the solutions are ordered and chosen based on good p-values of less than 0.05.

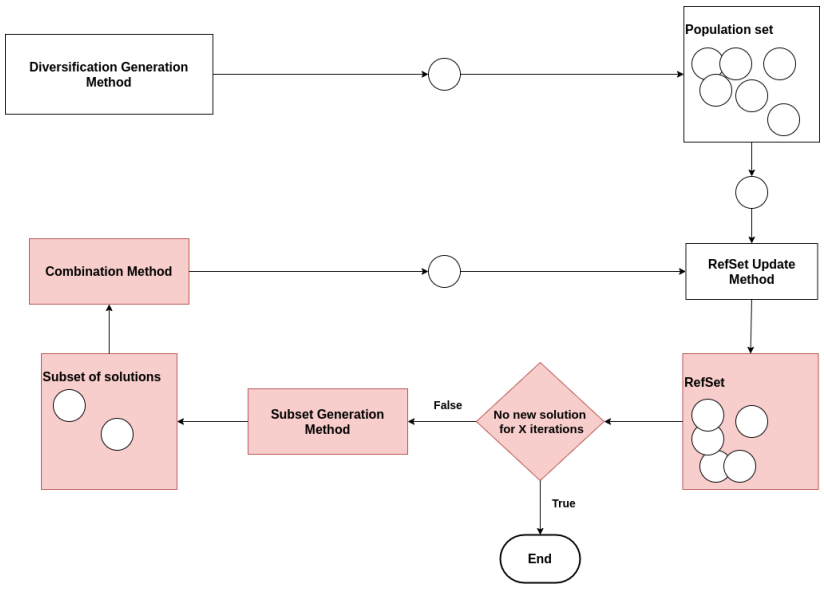


Fig. 4.10 Scatter search algorithm 4

With the RefSet populated the main iterative stage of the algorithm can begin. Solutions are chosen at random and must be compatible with each other, that is they have edges and nodes that can be joined.

Once chosen the combination can take place, the best scoring graph according to the p-value is the primary graph that takes priority. The graphs are merged and if the new graph is acyclic it is added back into the RefSet with a new p-value. The p-value must be less than 0.05.

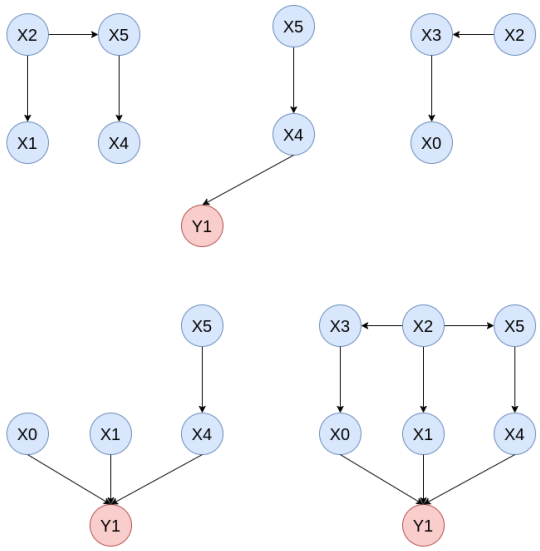


Fig. 4.11 Combining RefSet solutions

Figure 4.11 shows the process of combining graphs. Once the process is finished there should be multiple graphs with Y1. Each graph containing Y1 is taken from the RefSet and reduced to the Markov Blanket of the graph.

With a selection of graphs reduced in size to the Markov Blanket, the parameters are then fitted. Reducing the size of the graph improves parameter fitting time due to fewer nodes and edges. Once fitted the graphs can be tested and gauged based on accuracy and confusion matrix metrics.

4.5 Summary

Scatter search offers a way to build up graphs in small increments testing all possible attributes in the early stages without getting trapped in local optima. Many solutions can be found and discussed with stakeholders and any domain knowledge applied.

Reducing the size of the graph with Markov blankets has many advantages including training time and a simpler understanding of the graphs. Only knowledge that is needed is included in the graph.

The scatter search algorithm is very modular and easy to implement in different languages, with a modular design comes easier debugging and understanding of the components. Further improvements can be easily made to each of the components.

Chapter 5

Benchmarking results for the scatter search metaheuristic in healthcare

5.1 Introduction

Many classification models exist and there are different variations in Bayesian network training techniques. These methods are constraint-based, scoring-based and hybrid training methods. Scatter search learning of a Bayesian network has been performed using a scoring-based approach. The method used in this chapter is a newly developed constraint-based approach using statistical testing to learn the structure of multiple Bayesian networks with the best one used for classification. Here we outline the work already done in training Bayesian networks to perform classification tasks and the category under which they fall.

In healthcare, attribute selection and classification problems are present in many contexts. While attribute selection focuses on the discovery of relevant and non-redundant attributes (Li and Liu [107]), the classification of medical diagnoses belongs to the domain of supervised machine learning: Given some labelled training data for which attribute values and the class outcomes (diagnosis) are known, this training data is used to learn a classification model which is then used for the prediction of a diagnosis such as breast cancer [Zwitter et al.]. Classification methods that have been used in the past include decision rules, decision trees, support vector machines and deep learning. However, these methods have several shortcomings: Firstly, standard machine learning approaches often ignore causality between variables. Secondly, the interpretation of a Deep Learning model, for example, is difficult for a practitioner. Thirdly, the variable selection (e.g. for decision tree learning) has to be done separately from the actual classification problem.

In this chapter, we follow a different approach which unifies the attribute selection and the classification problem. We learn a Markov Blanket-based probabilistic graphical model which contains all relevant and non-redundant attributes for classification. This can be seen as a hybrid approach of both attribute selection and classification. Markov Blankets that we learn from the data are specific Bayesian networks that only contain the parent, children and parents of the children of a class attribute. In order to learn these networks, we develop a new Scatter Search heuristic method. We implement the algorithm in Python and evaluate it using publicly available benchmark data sets in health care that have been used in prior research. The computational results indicate that our procedure leads to competitive classification results yielding graphical models that allow decision-makers to interpret them. In addition, our results reveal that graphical models that are learned from the data have substantially fewer predictor variables than in the full data set. Our implementation is analyzed in terms of computation time, size of the graph and accuracy. We also benchmark our algorithm against other widely used classifiers. This novel algorithm can perform variable selection, as part of the procedure to learn the Markov Blanket, which may be more efficient than using other (classical) machine learning approaches.

The remainder of this chapter is structured as follows. The next section provides an overview of related work which highlights the different learning paradigms used for learning probabilistic graphical models. We conclude that section by highlighting similarities and differences of our work with the body of existing research. Section 5.3 describes our Scatter Search improvement heuristic. A comprehensive computational study is provided in Section 5.4 in which we show the benchmarking results of our algorithm using different datasets, performance metrics, and levels of detail. We discuss the implications of the methods and results in Section 5.5 followed by conclusions and opportunities for future work in Section 5.6.

5.2 Related Work

Algorithms for learning probabilistic graphical models have been reviewed by Larrañaga and Moral [104], while Scutari [160] has defined the following four types of search strategies in this context: Constraint-based methods, scoring-based methods, hybrid and parameter learning approaches. Table 5.1 provides an overview of related work broken down by these four approaches, which are detailed further.

Paradigm	Reference
Constraint-based methods	Spirtes et al. [165], Margaritis [121], Tsamardinos et al. [170], Yaramakala and Dimitris [185]
Scoring-based methods	Bai et al. [15], Campos [31], Zgurovskii et al. [187], Larrañaga et al. [105], Chickering [37, 40]
Hybrid	Friedman et al. [66], Tsamardinos et al. [171]
Parameter learning	Ji et al. [93]

Table 5.1 Overview of related work

5.2.1 Constraint-based Methods

The PC algorithm devised by Spirtes and Glymour [164] is a well-known example of a constraint-based learning method. Typically, hypothesis tests are used to find conditional (in)dependencies between attributes. Once the structure has been learned, the conditional probability distributions, required to fully specify the Bayesian network model, are estimated from data. The usual method for estimating the parameters is maximum likelihood estimation, although Laplace estimation and other Bayesian estimation approaches based on Dirichlet priors are also common.

5.2.2 Scoring-based Methods

Scoring methods are used in conjunction with search heuristics to construct graphs or improve upon initial graphs. Scoring guided search strategies are discussed in detail in Russell and Norvig [152]. The objective is to maximise the network score of the directed acyclic graph (DAG).

A method developed by Bai et al. [15] to improve an initial DAG uses a Tabu Search algorithm to manipulate edges (e.g. add, remove and switch) of the DAG to maximise the network score. Publications that look at different scoring methods to guide a greedy search are Chickering [37] and Chickering [40]. Larrañaga et al. [105] use a genetic algorithm approach for scoring and improving the network. Similarly, Zgurovskii et al. [187] evaluate the construction and scoring of the Bayesian networks whilst Campos [31] takes a different approach using mutual information tests.

5.2.3 Hybrid Approaches

Hybrid methods use constraint-based methods to find the conditional independence relationships within the Bayesian networks while a score-based approach is pursued to improve on those DAG structures to find the optimal graphical model. Tsamardinos et al. [171] presents an algorithm that combines a method to construct a Bayesian network using a Bayesian-scoring, greedy, and hill-climbing search to orient the edges. This is then compared against other methods. Friedman et al. [66] develop a new faster hybrid algorithm to be used on large datasets. This speed is achieved by restricting the search space. The scoring of networks is achieved via a mutual information approach. In contrast to their work, there are two main advantages of the algorithm presented in this chapter: improving search efficiency and improving the choice of candidates for better scoring networks.

5.2.4 Parameter Learning

Parameter learning is an important step in learning a functional Bayesian network. Fitting distributions for the nodes in the network can be achieved by many algorithms, and these can be suited to complete or incomplete data. The types of distributions are also important depending on the dimensionality of the data. Ji et al. [93] review different types of parameter learning algorithms for complete and incomplete data. The methods reviewed in this thesis are Maximum Likelihood Estimate, Bayesian Method, Expectation-Maximization, Robust Bayesian Estimate, Monte-Carlo Method and The Gaussian Approximation.

5.2.5 Conclusion

The algorithm proposed in this thesis can be categorized into and differentiated from the literature on probabilistic graphical models as follows: Our algorithm bridges the constraint-based and scoring-based learning paradigm because we use conditional independence tests when learning the graphical model but we also have scoring functions in the improvement of the network which rely on measures such as accuracy, precision and recall because we will only keep the best scoring networks in our Scatter Search. As a result, our work has similarities with the Tabu Search algorithm provided by Bai et al. [15] because of the use of diversification. The advantage of our work is, however, that we do not rely on a construction heuristic. The Scatter Search creates diverse graphs in the initialization step.

5.3 Methods

5.3.1 Problem Description, Graphical Models and Classification

Using the formal definition laid out in section 4.3 and taking a dataset of healthcare observations with each predictor variable labelled as X_i and the outcome variable labelled Y_i an attempt is made to find the relationships between X_i and Y_i for the best results. Multiple machine learning methods are applied to assess the performance of the scatter search Bayesian network model. The chosen metrics used in this chapter are accuracy, recall and precision. Relationships between variables are shown as graphs with vertices and edges.

All training has been conducted using a grid search to find the optimal settings for the machine learning algorithms. KFolds has been used to split the data with folds grid search of 2, 5 and 10. The final models and parameters are chosen based on accuracy. The SciKit learn library in Python has been used to achieve this using GridSearchCV in the model_selection module.

Probabilistic Graphical Models

In order to introduce Markov Blanket attribute selection, the notation of Bayesian networks, a type of probabilistic graphical model will be introduced. A Bayesian network is a directed acyclic graph (DAG) $\mathscr{G} := (\mathscr{V}, \mathscr{E})$ with vertices $\mathscr{V}$ and edges $\mathscr{E}$. Vertices represent variables and the edges encode the conditional independence relationships between these variables (each variable is conditionally independent of its non-descendants in the graph given its parents). Pearl [143] and Wasserman [177] provide further theoretical properties of Bayesian networks and other probabilistic graphical models. For an overview of statistical graphical models with applications in systems biology, see Nagarajan et al. [131] as well as Scutari and Strimmer [161].

The Markov Blanket (MB) is a subgraph of a Bayesian network. The MB of a vertex $v \in \mathscr{V}$, denoted by $MB(v)$, is a minimal subset of vertices containing vertex v, its direct parents and direct children as well as all direct parents of the children of v. The Markov Blanket of vertex v contains all the variables needed to predict the value of that variable since v is conditionally independent of all other variables given its Markov Blanket.

Classification Using Markov Blankets

Once we have trained the Markov Blanket using the training dataset, we assign a new instance i to the class d_i^* by employing Equation (5.1), which is called Maximum a-posteriori or MAP decision rule. The index set of the product $\prod_{a=1}^{|\mathscr{A}|}$ runs from attribute $a = 1$ to all attributes

$|\mathscr{A}|$ and multiplies the conditional probability of observing attribute value $v_{i,a}$ depending on d and all parents of a, encoded by Π_a. The classification accuracy is then evaluated using a confusion matrix which represents the predicted outcomes vs. the true i.e. actual outcome of the patient having the disease or not.

$$d_i^* = \arg\max_{d \in \mathscr{D}} \left\{ p(d) \prod_{a=1}^{|\mathscr{A}|} p(v_{i,a}|d, \Pi_a) \right\}. \tag{5.1}$$

5.3.2 Scatter Search Heuristic

Scatter Search can be used as both a construction heuristic and a meta-heuristic to improve probabilistic graphical models. The Scatter Search algorithm implemented here is used to create the initial graphs and improve on them by combining the nodes and edges from initial solutions. It is a meta-heuristic optimization algorithm that was first introduced by Glover [74]. Figure 4.3 provides an overview of our Scatter Search heuristic.

The figure shows that our algorithm first generates a diverse population set which includes high-quality solutions on the one hand but also diverse solutions on the other. Those are then combined if they lead to an improvement of, for example, prediction accuracy, following principles devised by Martí et al. [123].

Population Set and Reference Set (RefSet)

The population set contains graphs created by the diversification strategy. The RefSet contains X number of solutions with the best p-value and Y number of diverse solutions that are not connected directly back to the classification attribute. This ensures that there is a balanced trade-off between diversification and intensification.

In the main Scatter Search loop algorithm, we start by defining our two solution sets, which are the population set and the reference set (RefSet). The population set holds our initial solutions generated by the diversification method. This diversification method takes one parameter which is the alpha value to be set for the threshold of the p-value for the chi-squared test. If the solution meets the requirements of the alpha threshold it is added to the population set.

Once diversification has taken place and the population set has been populated, the RefSet will be updated with select solutions from the population set. Solutions are selected based on two different types of criteria - target solutions and diverse solutions. Best quality solutions (target solutions) are solutions that connect directly back to the attribute that is to be classified, and the quality of these solutions is based on the p-value. The second type is the diverse solution, and these solutions have at least one degree of separation from the

target/classification attribute/node. These diverse solutions can also be evaluated based on the p-value.

After the RefSet has been populated, the main loop that combines solutions can start. This loop has two conditions for completion defined as the last time the RefSet was updated and the total number of iterations completed. These are trade-offs between time, quality of solutions, and ensuring that the loop terminates.

The subset method is called to select two graphs. These graphs are taken from the subset method and passed to the combination method. Once this new subset has been passed to the combination method, the graphs are combined and a new solution graph is returned. This solution is then added to the RefSet if it is a unique solution and also a directed acyclic graph (DAG).

In conclusion, our Scatter Search works to create multiple solutions from an initial pool of solutions created by the diversification method. The Scatter Search metaheuristic then works to combine these solutions to find a subset of the best solutions.

5.4 Experimental Results

5.4.1 Datasets

We evaluated the algorithms using datasets from the UCI machine learning repository [91]. These datasets are from the life sciences section and aim to classify patients based on whether they have a specific diagnosis or not. Datasets that cover more levels in the life sciences domain are discussed in Herland et al. [87]'s review. Table 5.2 provides an overview of the selected data sets.

Dataset	Level	Attributes	Samples	Type	Reference
Acute Inflammations Diagnosis	Patient	8	120	Binary	[47] [46]
Audiology Standardized	Patient	71	200	Multi-Class	[16] [46]
Breast Cancer	Patient	10	286	Binary	[124]
Breast Cancer Wisconsin (Original)	Patient	10	699	Binary	[181][120]
Cryotherapy	Patient	7	90	Binary	[96] [56]
Fertility	Patient	10	100	Binary	[72] [48]
Parkinsons	Patient	23	195	Binary	[113] [112] [111]

Table 5.2 UC Irvine Machine Learning Repository Datasets [91]

The table reveals that the datasets vary in the number of attributes, sample sizes, and outcome categories - binary/multi-class classification problems. Also, the datasets have been evaluated in other studies which allow us to benchmark our results against those reported in other publications.

dataset	output type	class 0	class 1	count
acute-inflammations-diagnosis	binary	58.33 %	41.67 %	120
breast-cancer	binary	76.22 %	23.78 %	286
breast-cancer-wisconsin	binary	65.52 %	34.48 %	699
cryotherapy	binary	53.33 %	46.67 %	90
fertility	binary	88.0 %	12.0 %	100
parkinsons	binary	75.38 %	24.62 %	195

Table 5.3 Binary datasets class distribution

Table 5.3 shows the balance of the datasets which have a binary outcome variable. All datasets have an imbalance in the outcome variable. Two datasets which have a low imbalance are acute-inflammations-diagnosis and cryotherapy which isn't a problem for most machine learning applications. Datasets in which the imbalance is large are breast-cancer, breast-cancer-wisconsin, fertility and parkinsons. These can pose a problem to some machine learning algorithms which can be made worse by small sample sizes. Techniques can be applied such as smote oversampling. These datasets also offer a good opportunity to test how different machine learning algorithms perform with imbalanced datasets.

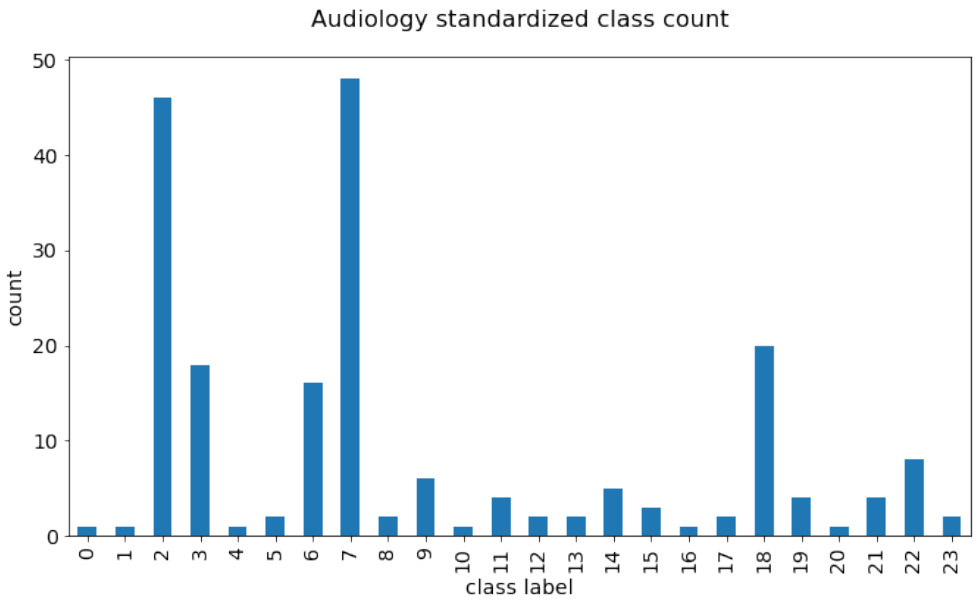


Fig. 5.1 Audiology standardized multi-class distribution

Figure 5.1 shows the distribution of data of the outcome variable in the Audiology standardized dataset. A balanced dataset should have a uniform distribution but in this case, data is distributed in classes 2,3,4,6,7 and 18. This can pose a problem in low-sample datasets but in this case, there are 699 samples which are larger than most in this study. Oversampling could once again be used to bring the distribution to a uniform distribution but for this study, the dataset is unchanged, apart from a basic pre-processing of the attributes to suit machine learning applications.

We implemented the Scatter Search algorithm using Python 3.8. We used the SciKit Learn Library [144] to calculate performance metrics such as accuracy and running cross-validation experiments. We also used the pgmpy library [8] to learn the probabilistic graphical models which we alter in our Scatter Search. To run conditional independence tests, we used the SciPy library [174]. Our implementation is available under https://github.com/jthrelfall777/ssmb.

5.4.2 Preprocessing

All datasets that have been used in this study have undergone preprocessing. All strings are transformed using the sklearn preprocessing library using the LabelEncoder [157] to convert strings to numerical values. The entire dataset is checked for missing values and those values are replaced by a zero value, i.e. "unknown". Once these steps have been taken, any continuous values are binned using the numpy histogram_bin_edges [136].

Numpy histogram_bin_edges [136] method is set to use the default options which take the maximum of the 'sturges' and 'fd' estimators for the bins. The Sturges method takes the number of samples and returns the number of bins as defined in the equation in figure 5.2. The Freedman Diaconis Estimator takes the interquartile range of the data and the number of samples and is defined in the equation in figure 5.3. Setting a small number of bins can lead to a loss of information but too many bins lead to spare coverage or overfitting over small data sets. Histogram binning is only one way of representing the data, kernel density estimation can also be used. Kernel density estimation fits smaller distributions over intervals of data and sums the distributions so they form one distribution and have an integral of one. Both sample size and distribution of the data affect the bin size. On returning the binning values (k) these are then applied to the dataset using the numpy digitize [135] function. This transformed dataset is written back out to a new file with a new header labelling predictor attributes X_i and prediction attributes Y_i. These header names are then used as the node names in the graphs.

$$k = \left\lceil \log_2 n \right\rceil + 1$$

Fig. 5.2 Sturges' rule for binning in histograms

$$k = \frac{IQR(x)}{\sqrt[3]{n}}$$

Fig. 5.3 Freedman–Diaconis rule for binning in histograms

5.4.3 Accuracy, Recall and Precision Evaluation

Accuracy

Table 5.4 shows the graphs in descending order of accuracy. Once the scatter search algorithm establishes the graphs there is very little improvement in the best solutions found. Some

datasets, such as Acute Inflammations Diagnosis, show no improvement even though the top 3 graphs are unique in structure. The top 3 graphs maybe differ in the number of nodes and edges but very little is added to the overall accuracy.

Data Set	Graphs $\mathscr{G}_i$		
	1	2	3
Acute Inflammations Diagnosis	1	1	1
Audiology (Standardized)	0.655	0.625	0.445
Breast Cancer	0.81	0.81	0.81
Breast Cancer Wisconsin (Original)	0.965	0.961	0.957
Cryotherapy	0.833	0.833	0.8
Fertility	0.95	0.9	0.9
Parkinsons	0.91	0.91	0.89

Table 5.4 Scatter Search Accuracy

Recall

Table 5.5 shows the graphs in descending order of accuracy. Acute Inflammations Diagnosis and Breast Cancer show no change in recall for the top 3 graphs. Audiology standardized, Breast Cancer Wisconsin (Original), Fertility and Parkinsons show an increase in recall as accuracy improves. Cryotherapy was the only dataset for recall to decrease as accuracy improved.

Data Set	Graphs $\mathscr{G}_i$		
	1	2	3
Acute Inflammations Diagnosis	1	1	1
Audiology (Standardized)	0.655	0.625	0.445
Breast Cancer	0.27	0.27	0.27
Breast Cancer Wisconsin (Original)	0.965	0.961	0.957
Cryotherapy	0.729	0.729	0.958
Fertility	0.66	0.16	0.16
Parkinsons	0.99	0.99	0.97

Table 5.5 Scatter Search Recall

Precision

Table 5.6 shows the graphs in descending order of accuracy. Acute Inflammations Diagnosis and Breast Cancer show no change in precision for the top 3 graphs. Audiology standardized, Breast cancer, Breast Cancer Wisconsin (Original), Cryotherapy and Parkinson's all show an increase in precision but Fertility shows a decrease in the metric as accuracy increases.

Data Set	Graphs $\mathscr{G}_i$		
	1	2	3
Acute Inflammations Diagnosis	1	1	1
Audiology (Standardized)	0.571	0.532	0.361
Breast Cancer	0.79	0.79	0.79
Breast Cancer Wisconsin (Original)	0.967	0.962	0.659
Cryotherapy	0.945	0.945	0.741
Fertility	0.88	1	1
Parkinsons	0.9	0.9	0.89

Table 5.6 Scatter Search Precision

Acute inflammations diagnosis is a well-balanced data set with 58.33 % for the false class and 41.67% for the true class. High accuracy is achieved at 100% with the scatter search with recall and precision getting a perfect score.

Audiology is a multiclass dataset that is unbalanced with data concentrated in 5 of the 23 classes. Recall shows the false negatives have been badly impacted affecting the true positives. Precision is even worse with the false positives impacting the true positives, precision achieved a best score of 0.57.

Breast cancer achieved a high accuracy but the dataset was highly unbalanced with 76.22% in the false class and 23.78% in the true class. This can be seen with the poor recall metric results where the false negatives were far higher than the true positives. Precision showed better results with the false positives but this is based on a highly unbalanced dataset with most of the data being in the false class.

Breast cancer wisconsin is also an unbalanced dataset although not as unbalanced as breast cancer. The outcome variable has 65.52% of the data in the false class with 34.48% in the true class. Accuracy was good with 81% and an excellent recall of 0.96 with the false negatives low and the same high result for precision at 0.96 with a low false positive.

Cryotherapy was a well-balanced dataset with the false class having 53.33% and the true class having 46.67% of the data. Recall scored 0.72 and precision scored 0.94 showing the false negatives and false positives at a low level in relation to the true positives.

Fertility was a very highly unbalanced dataset with 75.38% of the data in the false class and only 12% in the true class, even with an unbalanced dataset a high accuracy was achieved at 95%. Recall was towards the middle of the range with 0.66 let down by the false negatives. Precision was a lot better with lower false positives compared with the true positives.

Parkinson's had an accuracy of 91% but was a highly unbalanced dataset with 75.38% in the false class and 24.62% in the true class. Recall was 0.99 with very few false negatives and a precision of 0.9 with few false positives.

5.4.4 Graph Sizes and Computational Time

Table 5.7 shows the graphs in descending order of accuracy. The table shows the number of attributes and edges. The attributes are the nodes or vertices in the graph. The number of attributes can be greatly reduced by scatter search as shown with the Audiology dataset. As seen with the cryotherapy dataset more attributes do not always lead to better classification results, reducing the number of attributes and relationships between them can improve classification results.

Data set	original #attributes	Graph $\mathscr{G}_i$					
		1		2		3	
		$\lvert\mathscr{A}\rvert$	$\lvert\mathscr{E}\rvert$	$\lvert\mathscr{A}\rvert$	$\lvert\mathscr{E}\rvert$	$\lvert\mathscr{A}\rvert$	$\lvert\mathscr{E}\rvert$
Acute Inflammations	8	5	4	6	6	6	5
Audiology (Standardized)	71	6	5	11	10	8	7
Breast Cancer	10	5	5	6	6	6	5
Breast Cancer Wisconsin (Original)	10	5	4	6	6	4	3
Cryotherapy	7	2	1	3	2	3	2
Fertility	10	7	8	4	3	5	4
Parkinsons	23	4	3	5	4	3	2

Table 5.7 Graph sizes

Finding the structure of the graphs takes less than a second for the top 3 graphs as shown in Table 5.8, although the runtime of the entire algorithm may be considerably longer because

all the potential solutions have to be evaluated. Solutions may be combined to find new solutions creating even more graphs. The selection criteria for the RefSet may reduce the total number of solutions from the population set. These solutions in the RefSet are likely to grow again as the solutions are combined to form new solutions.

Data Set	Graphs $\mathscr{G}_i$		
	1	2	3
Acute Inflammations	630.224	587.205	354.466
Audiology (Standardized)*	701.944	186.809	645.615
Breast cancer	132.136	538.571	584.141
Breast Cancer Wisconsin (Original)	167.545	901.333	939.296
Cryotherapy	699.192	342.989	551.378
Fertility*	779.169	410.4	824.886
Parkinsons*	467.015	647.994	777.861

Table 5.8 Run times [ms] for the Scatter Search accuracy metric

5.4.5 Benchmarking Against other Classifiers

Choice of Classifiers

The classifiers used to benchmark the Scatter Search algorithm came from the scikit-learn[144] package. We chose diverse but common classifiers to give a broad range of benchmark results. These classifiers cover, for example, neural networks, support vector classifiers, tree classifiers, and Naive Bayes.

Results

	Metrics		
Model	Accuracy	Recall	Precision
Scatter Search	**1**	**1**	**1**
MLP Classifier	0.908	**1**	0.847
K Neighbors Classifier	0.817	0.82	0.829
SVC-Linear	0.842	0.82	0.857
SVC	0.808	0.82	0.821
Gaussian Process Classifier	0.825	0.84	0.829
Decision Tree Classifier	0.8	0.82	0.812
Random Forest Classifier	0.925	0.82	**1**
Ada Boost Classifier	0.808	0.84	0.812
Gaussian NB	0.917	0.82	0.981

Table 5.9 Benchmarking results - Acute Inflammations dataset

	Metrics		
Model	Accuracy	Recall	Precision
Scatter Search	0.655	0.655	0.571
MLP Classifier	0.5	0.5	0.536
K Neighbors Classifier	0.1	0.1	0.069
SVC-Linear	0.295	0.295	0.175
SVC	0.24	0.24	0.058
Gaussian Process Classifier	0.125	0.125	0.054
Decision Tree Classifier	0.56	0.56	0.537
Random Forest Classifier	0.48	0.48	0.446
Ada Boost Classifier	0.395	0.395	0.291
Gaussian NB	**0.685**	**0.685**	**0.678**

Table 5.10 Benchmarking results - Audiology (Standardized) dataset

	Metrics		
Model	Accuracy	Recall	Precision
Scatter Search	**0.81**	0.27	0.79
MLP Classifier	0.717	0.324	0.193
K Neighbors Classifier	0.699	**0.412**	**0.398**
SVC-Linear	0.71	0.279	0.179
SVC	0.759	0	0
Gaussian Process Classifier	0.762	0	0
Decision Tree Classifier	0.696	0.368	0.31
Random Forest Classifier	0.72	0.176	0.167
Ada Boost Classifier	0.706	0.368	0.189
Gaussian NB	0.703	0.353	0.185

Table 5.11 Benchmarking results - Breast Cancer dataset

	Metrics		
Model	Accuracy	Recall	Precision
Scatter Search	0.965	0.965	0.967
MLP Classifier	0.953	0.934	0.93
K Neighbors Classifier	0.96	0.954	0.932
SVC-Linear	0.964	0.963	0.936
SVC	0.83	1	**0.672**
Gaussian Process Classifier	0.96	0.946	0.939
Decision Tree Classifier	0.92	0.85	0.909
Random Forest Classifier	**0.963**	0.95	0.944
Ada Boost Classifier	0.947	0.913	0.935
Gaussian NB	0.951	**0.975**	0.899

Table 5.12 Benchmarking results - Breast Cancer Wisconsin (Original) dataset

	Metrics		
Model	Accuracy	Recall	Precision
Scatter Search	0.833	0.729	0.945
MLP Classifier	0.878	0.833	0.93
K Neighbors Classifier	0.756	0.812	0.75
SVC-Linear	0.9	0.917	0.911
SVC	0.622	**1**	0.585
Gaussian Process Classifier	0.867	0.854	0.896
Decision Tree Classifier	0.844	0.875	0.852
Random Forest Classifier	0.878	0.917	0.874
Ada Boost Classifier	**0.944**	0.938	**0.958**
Gaussian NB	0.856	0.938	0.823

Table 5.13 Benchmarking results - Cryotherapy dataset

	Metrics		
Model	Accuracy	Recall	Precision
Scatter Search	**0.95**	**0.66**	**0.88**
MLP Classifier	0.84	0.25	0.333
K Neighbors Classifier	0.81	0.25	0.236
SVC-Linear	0.88	0	0
SVC	0.88	0	0
Gaussian Process Classifier	0.88	0	0
Decision Tree Classifier	0.77	0.417	0.267
Random Forest Classifier	0.88	0	0
Ada Boost Classifier	0.82	0.167	0.312
Gaussian NB	0.48	0.333	0.086

Table 5.14 Benchmarking results - Fertility dataset

	Metrics		
Model	Accuracy	Recall	Precision
Scatter Search	**0.91**	0.99	0.9
MLP Classifier	0.78	**1**	0.774
K Neighbors Classifier	0.0.81	0.925	0.84
SVC-Linear	0.856	**1**	0.842
SVC	0.754	**1**	0.754
Gaussian Process Classifier	0.805	0.959	0.816
Decision Tree Classifier	0.733	0.83	0.819
Random Forest Classifier	0.785	0.966	0.793
Ada Boost Classifier	0.785	0.932	0.811
Gaussian NB	0.702	0.68	**0.911**

Table 5.15 Benchmarking results - Parkinsons dataset

Scatter search performed best with the acute inflammations dataset with perfect accuracy, recall and precision. Scatter search also outperformed all other classifiers on this data set with the Random forest classifier performing in a close second but a number of false negatives impacting overall performance. The division tree classifier was the worst performing classifier with 80% accuracy struggling with false negatives and false positive classifications shown in the recall and precision results.

The other three datasets where scatter search performed well were breast cancer, fertility and Parkinson's. Breast cancer accuracy was 81% but suffered with a very low recall of 0.27 because of false negative type two errors. Precision fared better with a score of 0.79 with fewer false positive type one errors. The SVC classifier comes in a close second but with a recall and precision of 0 showing that it was unable to achieve any true positive classifications. Taking into account the next best accuracy with a recall and precision greater than zero, it was found the Random Forest classifier performed second best. The worst performing classifier in terms of accuracy was the Decision Tree Classifier with an accuracy of 69%. With the fertility dataset scatter search achieved an accuracy of 95% with a recall of 0.66 and precision of 0.88. Other classifiers such as SVC Linear, SVC, Gaussian Process Classifier and Random Forest Classifier had high accuracies but had recall and precision scores of zero. The MLP classifier scored 84% but with low recall and precision scores. The Parkinson's dataset scored highest with an accuracy of 91%, recall of 0.99 and precision of 0.9. SVC-Linear was the second-best classifier with a perfect recall of 1. Even when the scatter search was not the best classifier it still performed well compared with other classifiers and was never the worst-performing classifier.

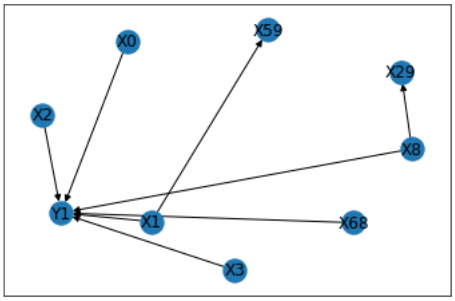
5.5 Discussion

5.5.1 Discussion of Construction Heuristic

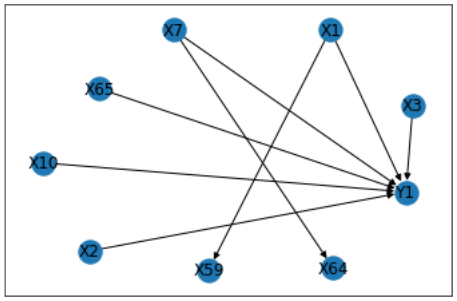
Our results demonstrate that the Scatter Search heuristic can substantially reduce the number of variables needed for classification. This highly depends on the construction heuristics and the nodes and edges handed over to the improvement heuristic stage. While the Scatter Search randomizes the graphs produced in the construction heuristic stage, other construction heuristics paired with our improvement strategy may come up with different performance figures.

5.5.2 Interpretability of the Graphical Models

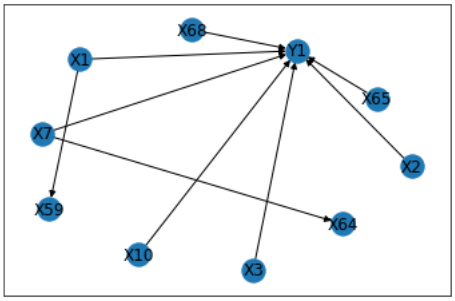
The audiology dataset has been chosen here to demonstrate the structures that are created by the scatter search, the dataset is complex with 69 predictor variables and 1 multi-class outcome variable. The dataset has 200 samples which is low considering the number of attributes the dataset includes and having a multiclass output. Audiology accuracy peaked at 65% which given the number of variables and the low sample rate was better than expected. The dataset produces graphs with many vertices and edges representing dependencies between dataset variables (attributes). Figures 5.4(a)–(c) show the probabilistic graphical model outputs of our scatter search applied to the audiology data set.



(a) Best performing graphical model



(b) Second best performing graphical model



(c) Third best performing graphical model

Fig. 5.4 Comparison of the three best performing probabilistic graphical models for the audiology data

The graphs help clinicians to visualize the relevant and non-redundant variables as well as their causal relationships. For example, in each of the graphs, the direct relationships between X1, X2 and X3 and Y1 are encoded. However, there are differences between the graphs which means that although our best-performing graph may lead to the best classification results, a clinician may be interested in a graph with a slightly lower accuracy but with alignment with current clinical guidelines.

Graph variable	Dataset variable	Type
X0	age_gt_60	binary
X1	air	multi-class
X2	airBoneGap	binary
X3	ar_c	multi-class
X7	bser	binary
X8	history_buzzing	binary
X10	history_fluctuating	binary
X29	m_s_gt_500	binary
X59	o_ar_u	multi-class
X64	static_normal	binary
X65	tymp	multi-class
X68	waveform_ItoV_prolonged	binary
Y1	class	multi-class

Table 5.16 Final models selected features

5.5.3 Discussion of the Discretization Method and Handling of Missing Values

As mentioned in Section 5.4, we used the numpy histogram_bin_edges [136] discretization method to discretize continuous attributes such as patients' age. There is always a trade-off between binning continuous attribute values, the information gain and the dimensionality that the algorithm can handle. Different binning methods may lead to different accuracy results and also large numbers of attributes will lead to a curse of dimensionality that our algorithm may not be able to handle anymore. Exploring this in more detail (e.g. the maximum number of discretized attributes) is an area of future research.

As mentioned in the computational results section, we replaced missing values with a "0" value, as suggested by Han et al. [82]. Evaluating other methods as reviewed in Young et al. [186] is an area of future research.

5.5.4 Discussion of Evaluation Metrics

Although we used accuracy as the objective function to maximize in the improvement stage, other metrics such as precision, recall and AUC can be used either in the i) optimization

objective and ii) as a performance metric. Our algorithm can be adapted straightforwardly to address these other metrics.

5.6 Summary and Conclusions

In this chapter, we have introduced a Scatter Search algorithm for Markov Blanket-based attribute selection and classification problem with an application to healthcare datasets. We benchmarked our heuristic using different performance metrics and levels of detail. Computational results reveal that our algorithm leads to competitive classification results. Another observation is that the graphical models can be inferred from the data relatively quickly. Furthermore, some models have substantially fewer predictor variables as compared to the variables in the full data set. In conclusion, our algorithm can be used in practice to classify patients effectively and efficiently. Combining our Scatter Search with other metaheuristics such as Tabu Search, and testing different improvement heuristics and larger datasets are promising future directions for this research.

Chapter 6

Ovarian Cancer Risk Factor Classification

6.1 Introduction

Ovarian cancer is the fifth most common cancer in women with it being often mistaken for irritable bowel syndrome, symptoms include bloating and abdominal pain [147]. Overall 5-year survival rates for this cancer are low at around 40%, even with advances in treatment options [42].

Early detection and screening could be vital to improving survival rates [126]; here we propose building a model from survey data that could be used as part of an app or other methods such as questioning in GP surgeries to raise awareness and alert women and practitioners due to an increased risk.

The models were trained from data provided by the Aneurin Bevan University Health Board with patients from two groups, a control group and patients with or that have had ovarian cancer. A total sample size of 438 patients was collected for the study by clinicians surveying patients. Training and test data were split using a KFolds approach with K=5 due to the small sample size of the data.

Five classification models were chosen: K Neighbors, Decision Tree, Random Forest, and Gaussian NB, with one newly developed model, Scatter Search. All selected models were chosen to work with clinicians to understand the models and make changes where needed for easy interoperability.

Models were assessed based on accuracy, ROC, recall precision, TN, FN, TP, NPV, PPV, sensitivity and specificity metric. Particular focus was given to accuracy and false negatives

because missing a patient with cancer as a result of a false negative result could lead to a delay in treatment.

6.2 Related Work

We break down our literature into two streams. First, we review Risk modelling and then focus on related work with respect to machine learning

6.2.1 Risk-based Modelling

Four risk-based modelling approaches to identifying women with a high risk of ovarian cancer which is related to this research are [109], [147], [67] and [149].

An epidemiological risk prediction model developed by [109] for ovarian cancer used non-invasive factors such as longer duration of hormone replacement therapy, body mass index, unilateral ovariectomy, longer duration of oral contraceptive use, and higher number of full-term pregnancies to name a few. It was found that the duration of hormone replacement therapy and higher body mass index increased the risk, whilst unilateral ovariectomy, longer duration of oral contraceptive use, and higher number of full-term pregnancies reduced the risk. The highest age group for risk was found to be 60 to 70.

Combining genomic information, clinical and epidemiological data in a study conducted by [147] was found to improve fitting patients into risk groups and to work with patients to monitor their health.

Another such study looks at combining genomic information, clinical and epidemiological data by [67], information in this study was collected by an online portal accessible by patients. Information gathered was used to develop an algorithm for personalised risk assessment. One hundred twenty-three volunteers took part in the study which were based on the uptake of ovarian cancer services and their impact on psychological health/quality of life. The study concluded the service has a positive effect on the patients involved.

A purely genetic screening approach was taken by [149]. This model differs in that model is used to predict survival rates rather than the risk of developing ovarian cancer.

6.2.2 Machine Learning

Using machine learning to analyse data and build predictive models could help build models with predictive capabilities. These algorithms could be used to inform healthcare partitioners and patients that an increased risk of cancer exists and that further consultation and testing are needed.

Many approaches have already been developed using datasets with different scopes from genetics, phenotypes and epidemiological methods. Six key publications have been identified through a literature search [54], [9], [2], [163], [10], [20].

Understanding patient survival based on the course of action and the patient's history is vital to the clinical staff's decision-making process. Traditional statistics offer survival models for this purpose in the survival analysis domain, but the question posed by [54] investigates using machine learning to predict survival time. The study found machine learning methods could achieve accuracy rates of up to 93% in predicting survival time. Data used in this study included patient records, gynaecology, obstetrics stage, grade and outcome of surgery.

Bayesian networks can incorporate prior knowledge into a model. For example, an approach taken by [9] evaluates taking prior knowledge such as patient data, genetic risks and pathology preoperative to predict if a patient has benign or malignant ovarian tumours. Even though ROC scores of 95.2 with a standard error of 3.4 were achieved, it was found discretisation of data could have a negative impact on data. In order to handle the negative impact of discretisation, it was found using expert knowledge for cut-off points could negate the impact on the final model inference.

Another approach was taken by [10] in the classification of benign or malignant ovarian tumours using Bagging and Random Forest AI on microarray data. The research found Bagging to be the best performing algorithm with 100% accuracy on 90% of the data, with Random Forest achieving 98.2% for 90% of the data. Using Mass spectrometry with machine learning algorithms for ovarian cancer has shown promising results in a research paper by [20]. Many classification algorithms and subsets of those algorithms were tested on data and found to have good results, with 85% or above for all algorithms. The best algorithm was found to be the multilayer perceptron neural network.

Artificial intelligence (AI) applied to the pathological diagnosis of ovarian cancer has been undertaken in [2], multiple AI models were applied using a support vector machine, random forest, naive Bayes, logistic regression, and XGBoost. 16 features were used from blood tests, patient background, and imaging tests. The highest accuracy achieved was 80% using the XGBoost algorithm. Similar research was undertaken by [163] by using Random Forest, Logistic, Multilayer Perceptron, Bagging, Classification Via Regression, LogitBoost, MultiClassifer, Simple Logistic, and Logistic Regression models. The results further back up the findings by [2] with comparative results.

Most literature focuses on lab testing and genetic testing with patient and family history. However, there are gaps in the literature for using survey data of current symptoms that could indicate ovarian cancer. The experiments conducted here using machine learning on survey

data will attempt to build a model that can classify a patient into binary groups that have ovarian cancer and don't have ovarian cancer.

6.3 Methods

The formal description of the ovarian cancer problem is as follows; let $\mathscr{I}$ denote a set of individuals (ovarian cancer patients). Let $\mathscr{N}$ denote the set of outcomes $\mathscr{N} := \{0,1\}$ which the patient either does not have cancer or the patient does have cancer. For each patient $i \in \mathscr{I}$, we observe a set of attributes $\mathscr{A}$, these attributes represent columns in the dataset such as age, height, has had children, etc. Let $\mathscr{V}_a$ denote the set of possible values for attribute $a \in \mathscr{A}$ and let $v_{i,a} \in \mathscr{V}_a$ denote the value of attribute a for patient i. We want to predict n_i when a patient i has cancer given the patient's values $v_{i,a}$ for each attribute $a \in \mathscr{A}$. Some attribute values $v_{i,a}$ may be missing. In this supervised learning problem, we assume the availability of labelled training data from surveyed patients $j \in \mathscr{I} \setminus i$ whose attribute values $v_{j,a}$ and outcomes n_j are known.

6.3.1 Attribute Ranking and Selection Techniques

Information Gain (IG) and Relief-F provide a quick estimate and ranking of relevant attributes and are techniques that are commonly used in attribute ranking (Hall and Holmes [81]).

Information Gain (IG) and Combined IG (IG*) Attribute Ranking. Given the prior probability $p(n)$ for the (has cancer) outcome $n \in \mathscr{N}$, we can compute the information entropy $H(\mathscr{N})$ and the conditional information entropy $H(\mathscr{N}|a)$ of $\mathscr{N}$ given an attribute $a \in \mathscr{A}$ ([81]). The information entropy $H(\mathscr{N})$ is calculated by Equation (6.1).

$$H(\mathscr{N}) = -\sum_{n \in \mathscr{N}} p(n) \ln p(n) \tag{6.1}$$

The negative sign ensures that $H(\mathscr{N})$ is positive or zero. The more uniformly an attribute value is distributed over all instances, the higher its entropy (see [23]). Using Equation (6.2), we can compute the conditional information entropy $H(\mathscr{D}|a)$ of $\mathscr{D}$ given an attribute $a \in \mathscr{A}$. Here, $p(v)$ is the prior probability of attribute value $v \in \mathscr{V}_a$ for attribute $a \in \mathscr{A}$ and $p(n|v)$ is the conditional probability of the (has cancer) outcome n given an attribute value $v \in \mathscr{V}_a$ of attribute $a \in \mathscr{A}$.

$$H(\mathscr{N}|a) = -\sum_{v \in \mathscr{V}_a} p(v) \sum_{n \in \mathscr{N}} p(n|v) \ln p(n|v) \tag{6.2}$$

The information gain $IG(a)$ of each attribute $a \in \mathscr{A}$ is then computed by Equation (6.3).

$$IG(a) = H(\mathscr{N}) - H(\mathscr{N}|a) \tag{6.3}$$

Relief-F Attribute Ranking. [97] have developed a class of algorithms which is very efficient for binary classification problems. [150] refined their algorithm and called it Relief-F. The result is a quality measure of each attribute Q_a, which can provide an attribute ranking, similar to $IG(a)$. Like IG*.

To rank the attributes further concerning the (has cancer) attribute, we employed the Chi-Square p-value. The Chi-Square p-value is used by the scatter search algorithm when testing for variable independence using multi-way tables. In presenting this information, we can better understand the p-values the scatter search will be using.

6.3.2 Machine Learning Algorithms

k-nearest Neighbors The k-nearest neighbours algorithm is a Non-parametric instance-based classification [24][89] method first developed by Evelyn Fix and Joseph Hodges [60]. k-nearest neighbours are based on measuring the distance between test data and training data using distance measures such as Euclidean, Chi-square, cosine and Minkowski distance to classify data. [89] It is based on the principle that features that are evaluated on a distance metric are similar.

Gaussian Naive Bayes The Gaussian Naive Bayes is a variant of Naive Bayes that follows the Gaussian normal distribution. Naive Bayes Classifiers are based on Bayes Theorem and assume independence of the random variable. Gaussian Naive Bayes supports continuous variables, but all presented data has been discretised. Therefore, the likelihood of all variables is considered to be Gaussian in nature.

Bayesian Networks The naive Bayes approach assumes that each attribute is only dependent on the class but not dependent on other features, which is rarely true. Thus, we extend the naive Bayes classifier to a Bayesian network classifier, where the set of conditional independence assumptions are encoded within the graph. As in the naive Bayes approach, we learn the conditional probabilities from the training data, but now we must condition not only on the class but also on any other parents Π_a of the given attribute a in the graph.

Decision Trees Decision tree learners automatically learn a decision tree from labelled training data. There are various methods to learn the structure of a decision tree from data: We use an algorithm which has been investigated by Hall and Holmes [81] in combination with attribute selection. We employ this algorithm because we can control the over-fitting of the decision tree and the tree size during the learning process.

6.4 Experimental Investigation

6.4.1 Study Design

Figure 6.1 shows the process flow from data cleaning through to training the classification models. The process starts by fixing any errors in the formatting of attributes in the dataset, followed by standardising attributes. Yes and No answers are converted to binary in a set of 0,1, and other attributes such as age are treated as discrete categorical variables in a set of . Missing attribute values were not an issue with this dataset as all relevant columns were complete. High and weight attributes were discretised and binned according to the Freedman–Diaconis [61] rule.

Data exploration covered in section 6.4.3 scores and ranks attributes using ANOVA F-value, the chi-squared p-value and information gain. The distribution of these ranked attributes is evaluated in accordance with the Y1/has_cancer attribute.

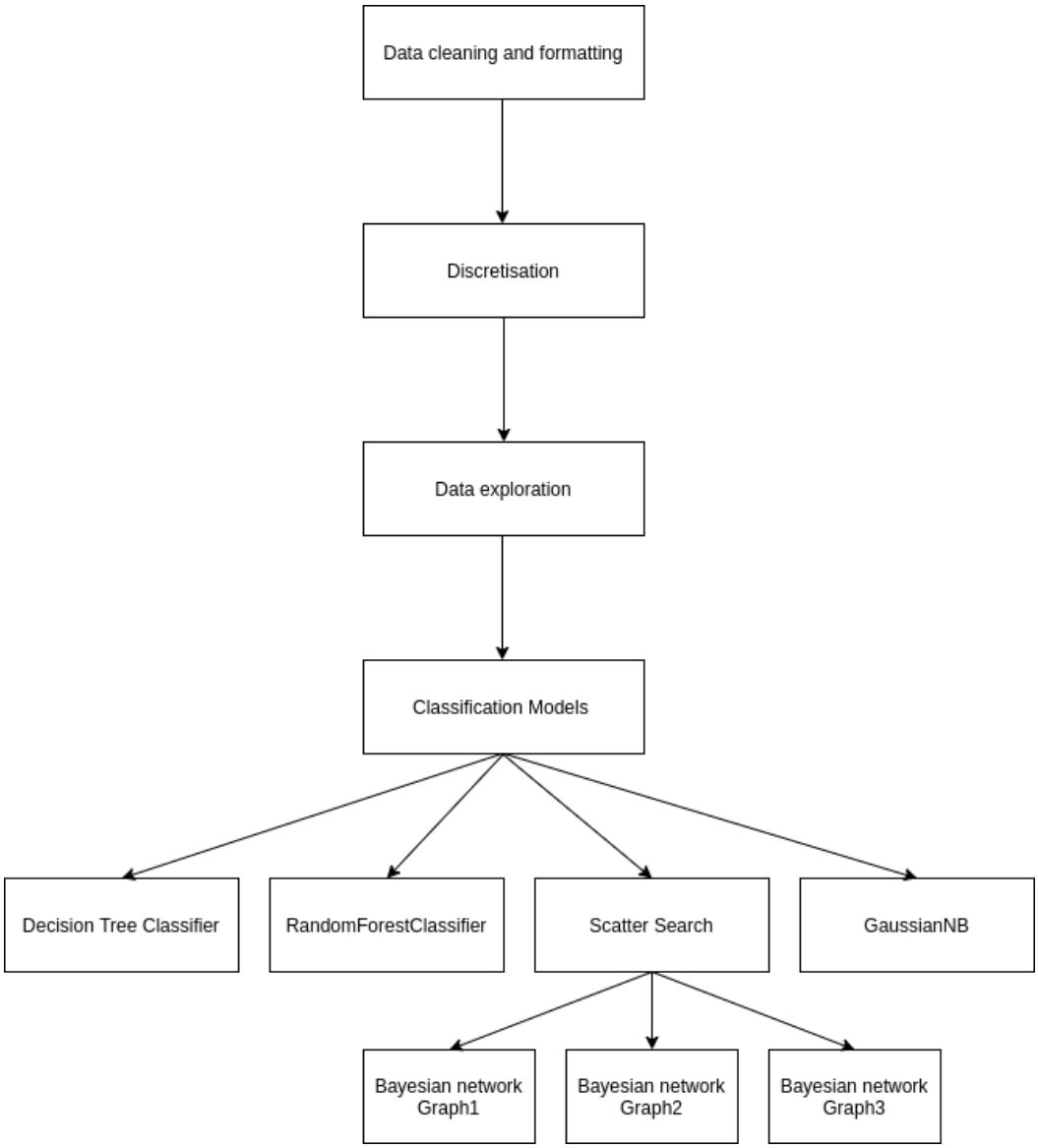


Fig. 6.1 Study design flowchart

Multiple classification models are used to find the best predictive model for the data. The chosen classifiers are Decision Tree Classifier, Random Forest Classifier, Scatter Search (Bayesian network) and Gaussian Naive Bayesian classifier.

In order to gauge the performance of the models' multiple metrics are used. These metrics are accuracy, roc, recall, precision, TN, FN, TP, FP, population, NPV, PPV, sensitivity and specificity. Two extra metrics are used for the Bayesian networks created by the scatter search. These are the number of vertices (nodes) and the number of edges in the graph.

Training and test data are split using k-fold cross-validation. To find the best performance for training, 2, 5 and 10 folds were used during training were used with both unbalanced

and balanced data. The number of samples in an unbalanced dataset was 438, and by using SMOTE, oversampling generated 756 samples.

6.4.2 Data and Information Documented for Ovarian Cancer dataset Prediction

All data was collected at the Aneurin Bevan University Health Board by clinical staff interviewing patients with ovarian cancer and those without ovarian cancer. The aim was 500 patients but a total of 438 patients were selected due to the small number of patients still alive with the disease. In the control group, there are 378 patients and in the group with ovarian cancer, there are 60. The data was collected between 17 May 2021 and 16 September 2021. All data is limited to the geographical area of South East Wales.

Table 6.1 gives a summary of the dataset with the mean, std, min, max and percentiles being reported. The Ovarian Cancer dataset has 438 samples with 756 when oversampled with the SMOTE method and 31 attributes including the attribute (Y1) to be classified. Y1 - has_cancer is a binary yes or no represented in set 0,1 and is the attribute to be predicted by the classifiers. The dataset is split with 596 samples for training and 160 for testing.

xy	field	mean	std	min	max	25%	50%	75%	data_type
X0	age	45.07	19.05	15.00	96.00	25.00	47.00	59.00	integer
X1	age period started	12.63	2.12	8.00	42.00	11.00	13.00	14.00	integer
X2	age started menopause	48.04	5.25	31.00	59.00	45.7500	49.00	52.00	integer
X3	irregular periods	0.39	0.48	0.00	1.00	0.00	0.00	1.00	binary
X4	use the contraceptive pill	0.83	0.37	0.00	1.00	1.00	1.00	1.00	binary
X5	years on the contraceptive pill	8.98	8.19	0.08	40.00	2.00	7.00	12.00	integer
X6	have children	0.62	0.48	0.00	1.00	0.00	1.00	1.00	binary
X7	age at 1st child birth	25.43	5.47	17.00	46.00	21.00	24.00	29.00	integer
X8	breastfed children	0.40	0.49	0.00	1.00	0.00	0.00	1.00	binary
X9	IVF treatment	0.03	0.19	0.00	1.00	0.00	0.00	0.00	binary
X10	HRT number of years	4.87	5.56	0.16	25.00	1.00	3.00	6.00	integer
X11	has a close family member had cancer	0.43	0.49	0.00	1.00	0.00	0.00	1.00	binary
X12	endometriosis	0.06	0.25	0.00	1.00	0.00	0.00	0.00	binary
X13	diagnosed with a genetic condition	0.04	0.21	0.00	1.00	0.00	0.00	0.00	binary
X14	polycystic ovary syndrome PCOS	0.07	0.25	0.00	1.00	0.00	0.00	0.00	binary
X15	height	163.39	6.85	124.46	182.88	157.48	162.56	167.64	integer
X16	weight	74.94	24.54	38.10	374.65	60.66	69.85	82.55	integer
X17	fatigue	0.43	0.49	0.00	1.00	0.00	0.00	1.00	binary
X18	bleeding from your bottom	0.06	0.25	0.00	1.00	0.00	0.00	0.00	binary
X19	tummy swelling	0.34	0.47	0.00	1.00	0.00	0.00	1.00	binary
X20	unexplained back pain	0.19	0.39	0.00	1.00	0.00	0.00	0.00	binary
X21	poor appetite	0.16	0.37	0.00	1.00	0.00	0.00	0.00	binary
X22	change in bowel habit	0.16	0.37	0.00	1.00	0.00	0.00	0.00	binary
X23	weeing all the time without water infection	0.17	0.38	0.00	1.00	0.00	0.00	0.00	binary
X24	bleeding from vagina	0.09	0.28	0.00	1.00	0.00	0.00	0.00	binary
X25	feel full very quickly after eating	0.20	0.40	0.00	1.00	0.00	0.00	0.00	binary
X26	unexpected weight gain	0.14	0.35	0.00	1.00	0.00	0.00	0.00	binary
X27	feel sick	0.18	0.38	0.00	1.00	0.00	0.00	0.00	binary
X28	tummy/pelvic pain	0.17	0.37	0.00	1.00	0.00	0.00	0.00	binary
X29	change in bowl habit	0.03	0.18	0.00	1.00	0.00	0.00	0.00	binary
X30	unexpected weight loss	0.07	0.25	0.00	1.00	0.00	0.00	0.00	binary
Y1	has cancer (Yes/No)	0.13	0.34	0.00	1.00	0.00	0.00	0.00	binary

Table 6.1 Dataset summary

All binary 0,1 attributes represent no (0) or yes (1) answers that patients were asked, most attributes are binary in nature. Attributes have both an X or Y name as well as a full field name, the purpose of the X and Y naming is for easy representation when labelling nodes in the graphs.

Fields in the set of X0, X1, X2, X5, X7, X10, X15, X16 were binned using the Freedman–Diaconis [61] method. Fields in the set of X15, X16 were binned and scaled for better representation of the normal distribution.

6.4.3 Attribute Ranking Results

Chi-squared results are presented in Table 6.2 with the top 10 attributes. Attributes are tested against Y1 (has_cancer) with a threshold set of $P < 0.005$. The results have been limited to the top 10.

xy	field	p_value
X0	age	4.417876e-59
X7	age_at_birth	1.517256e-43
X4	contraceptive_pill	2.241773e-32
X15	height	1.049543e-24
X5	yes_contraceptive_pill	6.719621e-24
X6	children	1.855719e-21
X3	irregular_periods	6.121122e-16
X2	age_started_menopause	8.142330e-12
X16	weight	2.405506e-09
X28	Tummy/pelvic pain	2.874448e-09

Table 6.2 Chi-squared test p values - top 10

The top 5 attributes scored significantly better than the bottom five attributes, with the bottom five scoring closer together. A difference between the top rank and the bottom rank result was observed at a value of 3.4420569999859865e-05. Of all attributes tested, age was the only integer, and the rest were binary values. Lower is better for this table.

Table 6.3 shows the ANOVA F value test for attributes tested against the Y1 (has_cancer) attribute. The lower the p-value, the better. Only the top 10 attributes are listed in the table and ranked by p-value.

xy	field	score
X0	age	9.651105e-75
X4	contraceptive_pill	5.188018e-36
X5	yes_contraceptive_pill	2.948741e-28
X6	children	4.493484e-23
X3	irregular_periods	7.171765e-17
X2	age_started_menopause	2.350261e-12
X7	age_at_birth	3.207272e-12
X28	Tummy/pelvic pain	1.118476e-09
X16	weight	4.856577e-09
X11	close_family_member_cancer	7.717391e-09

Table 6.3 ANOVA F-value p values - top 10

Table 6.3 (ANOVA) shares common attributes with Table 6.2 (Chi-squared test) although not all attributes are ranked the same. Attribute age_at_birth was ranked differently with ANOVA F-value ranking much lower than Chi-squared, yet age is ranked top in both methods. X15 height was ranked lowest in ANOVA but not at all with the Chi-squared test. The attribute weight has the same ranking in both feature selection methods.

Table 6.4 list the results from ranking with Mutual Information. Higher scores indicate stronger dependence between variables. In this case, attributes are scored against the Y1 (has_cancer) attribute.

xy	field	score
X0	age	0.259391
X7	age_at_birth	0.176218
X5	yes_contraceptive_pill	0.098449
X4	contraceptive_pill	0.097721
X15	height	0.093357
X6	children	0.063724
X3	irregular_periods	0.045402
X2	age_started_menopause	0.045014
X16	weight	0.034355
X14	polycystic_ovary_syndrome_PCOS	0.025136

Table 6.4 Mutual information (MI) - top 10

Once again table 6.4 shows age is at the top of the list, as shown in Table 6.3 and Table 6.2. X14 polycystic_ovary_syndrome_PCOS is a new attribute that does not appear in tables 6.3 and 6.2 although this attribute is at the bottom of the table.

Table 6.5 presents the intersection of Table 6.2, Table 6.3 and Table 6.4. These are the common attributes shared between all tables. Age was the biggest factor in all tables.

xy	field	f_classif_score	ig_selection_score	chi_test_p_value	p_value_met
X0	age	9.651105e-75	0.259391	4.417876e-59	True
X7	age_at_birth	3.207272e-12	0.176218	1.517256e-43	True
X5	yes_contraceptive_pill	2.948741e-28	0.098449	6.719621e-24	True
X4	contraceptive_pill	5.188018e-36	0.097721	2.241773e-32	True
X6	children	4.493484e-23	0.063724	1.855719e-21	True
X3	irregular_periods	7.171765e-17	0.045402	6.121122e-16	True
X2	age_started_menopause	2.350261e-12	0.045014	8.142330e-12	True
X16	weight	4.856577e-09	0.034355	2.405506e-09	True

Table 6.5 Intersection of attributes in F score, information gain and chi test

Common attributes have been listed in Table 6.5, the scores for each ranking method have been included with a threshold for p_values being set at 0.005. X0, X7, X5 were the highest-ranking attributes across all tables.

6.4.4 Exploratory Data Analysis

Data visualisations of attributes with the top-ranking scores are presented here with a heatmap representing correlations between attributes in the dataset. Integer attributes are represented with bar chart distributions and binary variables are represented with mosaic charts. All integer distributions represent two groups, that have cancer (1) and does not have cancer (2) in relation to the integer values on the X-axis.

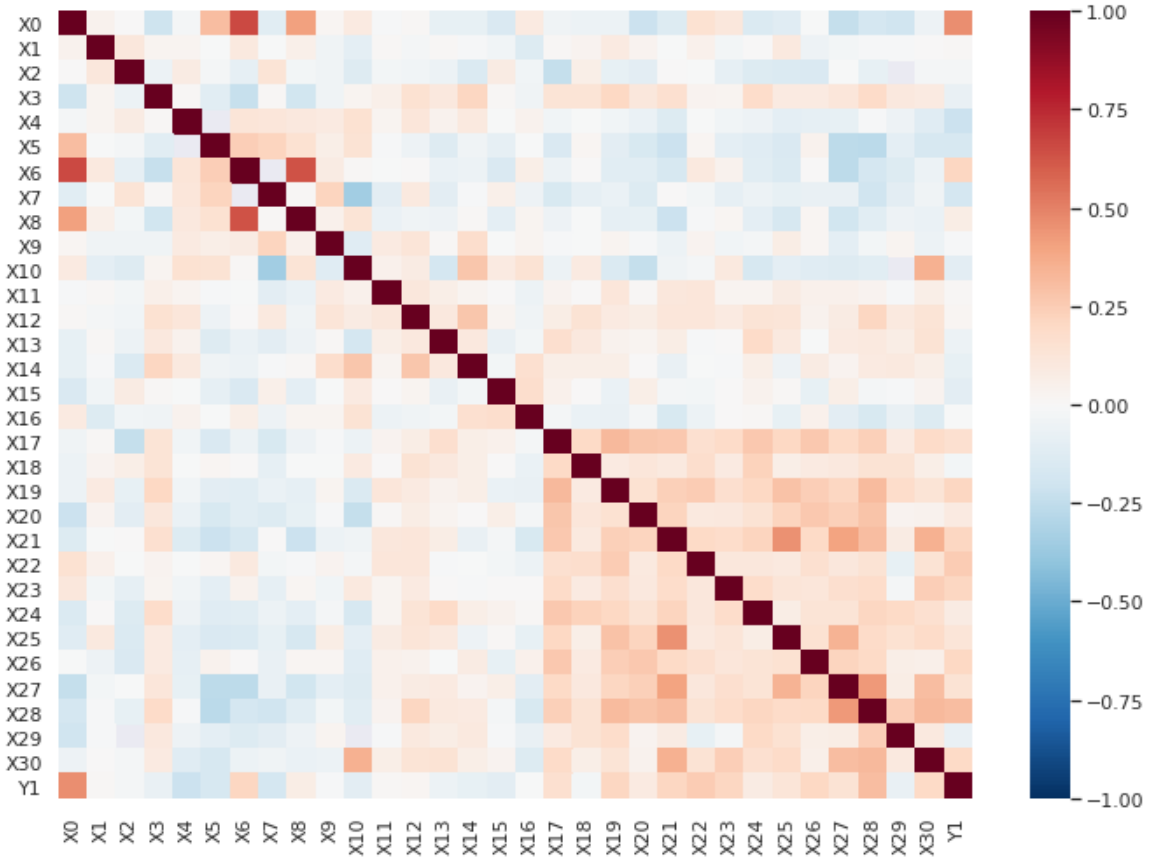
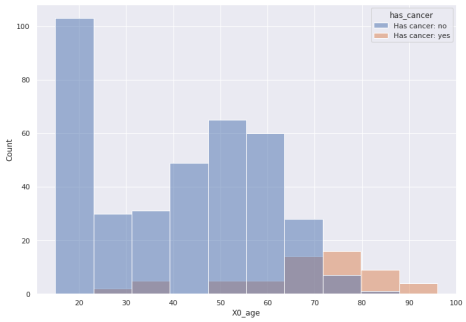


Fig. 6.2 Correlation Heatmap

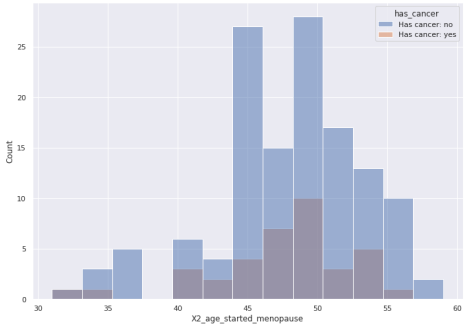
A correlation heatmap is presented in Figure 6.2 and shows a correlation between attributes in the dataset, red is a positive correlation and blue is negative. Attributes X0 (age), X2 (age started menopause), X6 (have children), X22 (change in bowel habit), and X28 (tummy/pelvic pain) have a strong positive correlation. Attributes with a strong negative correlation are X4 (use the contraceptive pill), and X5 (years on the contraceptive pill).

Interestingly X21 (poor appetite) is shown to have strong correlations with X25 (feel full very quickly after eating), X27 (feel sick), X28 (tummy/pelvic pain) and X30 (unexpected weight loss). X21 (poor appetite) was shown by the Chi-squared test and ANOVA F statistic to have a relation to Y1 (has_cancer). Colours in the heat map also indicate an X21 (poor appetite) correlation to Y1 (has_cancer).

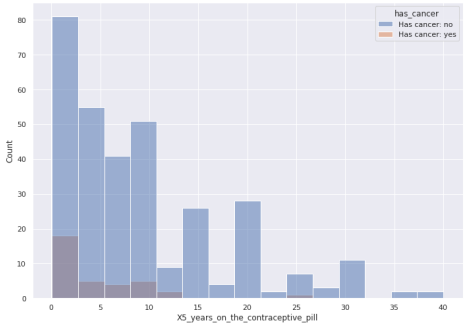
Another correlation shown in the heatmap is between X19 (tummy swelling) with X22 (change in bowel habit), X25 (feel full very quickly after eating) and X28 (tummy/pelvic pain). Attribute X0 (age) was selected by all three ranking techniques with relation to Y1 (has_cancer) with the heat map showing a positive correlation.



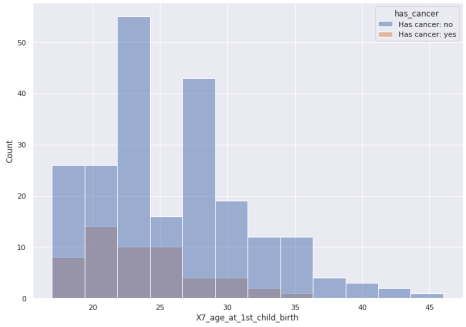
(a) x0 age range



(b) X2 age started menopause range



(c) X5 years on the contraceptive pill range



(d) X7 age at 1st child birth range

Fig. 6.3 Distributions attributes X0, X2, X5 and X7

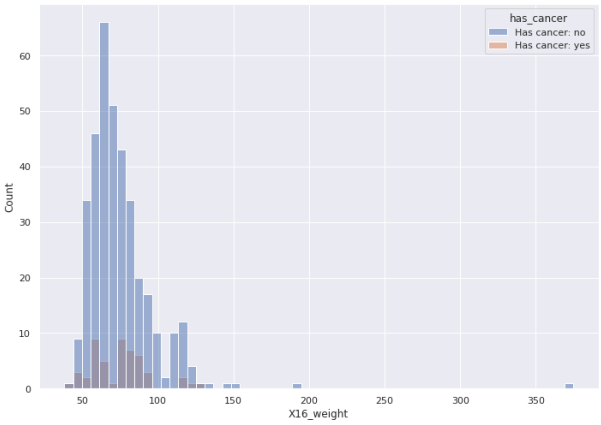
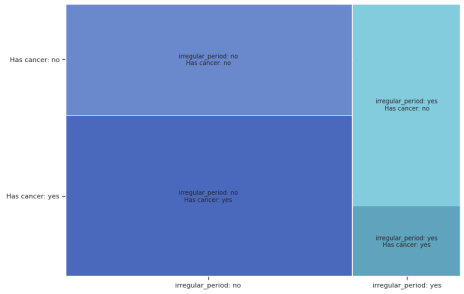
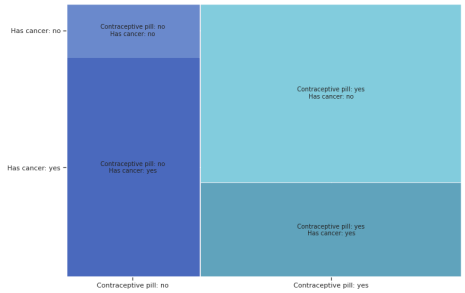


Fig. 6.4 X16 weight

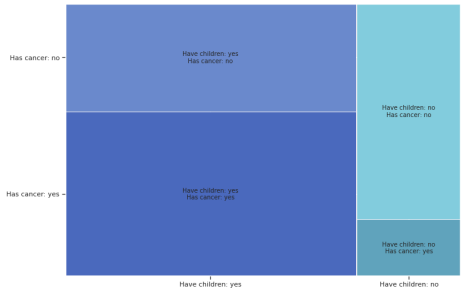
Figure 6.3a shows the age range for those with and without cancer. There is a clear distinction between the two groups, with those having cancer being in an older group range. The distribution for those with cancer has a mean of 68, a minimum of 25 and a maximum of 96 in contrast to those without with a mean of 41, a minimum of 15 and a maximum of 83.



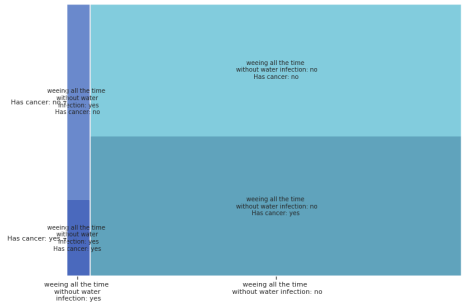
(a) X3 irregular periods



(b) X4 use the contraceptive pill



(c) X6 have children



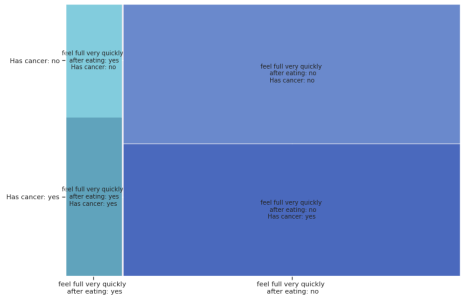
(d) X23 weeing all the time without water infection

Fig. 6.5 Distributions attributes X3, X4, X6 and X23

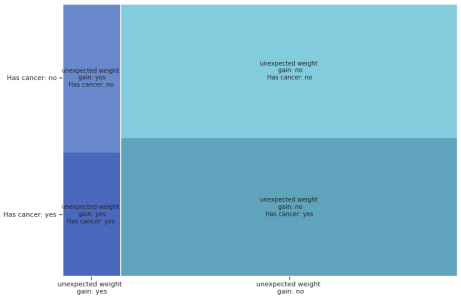
The distribution of age at which the menopause start is shown in figure 6.3b for those with and without cancer. A higher number of cases is shown for those who have started the menopause. Figure 6.3c shows the number of years on the contraceptive pill with Figure 6.3d showing the age at which the first child was born, zero indicates no children or failure to give this information.

Use of the contraceptive pill is shown in figure Figure 6.5b. A larger ratio of those not using the pill to those who are using the pill can be observed for has cancer. Those that are using the pill have a smaller ratio. Figure 6.5c shows 'have children: yes' have a higher ratio than for no with cancer whilst 'Have children: no' the ratio is much higher. Figure 6.5d: X23 weeing all the time without water infection shows an almost even split for given cancer and not having cancer.

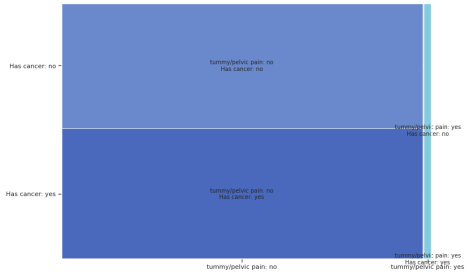
Figure 6.6a X25 feeling full very quickly after eating shows a higher ratio in favour of having cancer with an equal ratio between not feeling full very quickly given not having cancer. Unexpected weight gain X26 is shown in figure 6.6b, with the ratio lower for weight gain given cancer, not having unexpected weight gain given cancer has an equal ratio. Figure 6.6c tummy pelvic pain X28 shows a strong indication of the ratio of having cancer. Unexpected weight loss X30 given the individual has cancer shows a small ratio towards both having unexpected weight loss and having cancer in figure 6.6d.



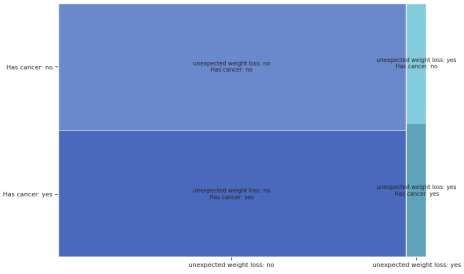
(a) X25 feel full very quickly after eating



(b) X26 unexpected weight gain



(c) X28 tummy pelvic pain



(d) X30 unexpected weight loss

Fig. 6.6 Distributions attributes X25, X26, X28 and X30

Summary of the Attribute Selection.

In total, there are 31 attributes with 1 binary outcome attribute (Y1 - has cancer).

Using ranking techniques, we were able to find seven attributes that ranked highly across all three ranking methods. Two of the highest-scoring attributes that came up across all methods were X0 (age) and X7 (age_at_birth). These attributes constantly scored better than other attributes and scored much higher than others in the dataset. Other attributes that ranked highly were X5 (yes_contraceptive_pill), X4 (contraceptive_pill), X6 (has_children), X3 (irregular_periods) and X2 (age_started_menopause).

The correlation heatmap shown in Figure 6.2 highlighted some interesting correlations with poor appetite, feeling sick, tummy pain and unexpected weight loss. X19 tummy swelling was found to correlate with X22 (change in bowel habit), X25 (feel full very quickly after eating) and X28 (tummy/pelvic pain). Distributions are shown in Figure 6.3 for the top-ranking integer attributes with Y1 (has_cancer), of which age was shown to be higher for those with cancer. Mosaic plots for binary attributes are shown in Figures 6.5 and 6.6.

6.4.5 Classification Results

Five classifiers have been used on the ovarian cancer dataset, these are KNeighborsClassifier, DecisionTreeClassifier, RandomForestClassifier, GaussianNB and Scatter Search. Table 6.6 shows the results from the first four classifiers and the Scatter Search results.

The training was conducted using a grid search of parameters for the models. The KFolds were set at 2, 5 and 10. Scatter search also had a grid search over the parameters Kfols, p_value and refset_size. These parameters were set as Kfold; {2,5,10}, p_values; {0.01, 0.05, 0.09} and refset_size; {10, 40}. A training loop is iterated over these parameters to find the best parameters for the model. Once trained all models were tested on 160 samples left out from the main training set and these were used to select the model best on accuracy. The training dataset has 596 samples. All results are reported in table 6.6 were tested on 160 samples left out from the training dataset.

Multiple metrics have been used to evaluate the models. TN (True negative), FN (False negative), TP (True positive) and FP (False positive) are the metrics used to derive the other metrics in the table.

Accuracy is the number of correct classifications over the total number of classifications and gets derived from (TP+TN/TP+TN+FP+FN).

Sensitivity, Recall or True Positive Rate (TPR) is the True Positives over True Positives and False Negatives (TP/TP+FN).

Specificity or True Negative Rate (TNR) is how many True Negatives there are over True Negatives and False Positives (TN/TN+FP).

The ROC score or ROC AUC reported on in the tables is the curve between sensitivity and specificity. A score of 0.5 is no better than random chance, and a score of 1 is a perfect classification. The closer the ROC AUC score is to 1, the better.

Precision is defined as the True Negatives over True Negatives and False Positives (TP/TP+FN).

Negative Predictive Value (NPV) is defined as the True Negatives over True Negatives and False Negatives (TN/TN+FN).

Positive Predictive Value (PPV) is defined as the True Positive over True Negative and False Positive (TP/TP+FP).

model name	accuracy	ROC	recall	precision	TN	FN	TP	FP	NPV	PPV	sensitivity	specificity
KNeighborsClassifier	78.125	0.781	0.675	0.857	71	26	54	9	0.732	0.857	0.675	0.888
DecisionTreeClassifier	78.125	0.781	0.775	0.785	63	18	62	17	0.778	0.785	0.775	0.788
RandomForestClassifier	81.250	0.812	0.762	0.847	69	19	61	11	0.784	0.847	0.762	0.862
GaussianNB	53.125	0.531	0.762	0.521	24	19	61	56	0.558	0.521	0.762	0.300
Scatter Search Graph 1	87.5	0.875	0.888	0.866	69	9	71	11	0.885	0.866	0.888	0.862
Scatter Search Graph 2	87.5	0.875	0.888	0.866	69	9	71	11	0.885	0.866	0.888	0.862
Scatter Search Graph 3	87.5	0.875	0.925	0.841	66	6	74	14	0.917	0.841	0.925	0.825

Table 6.6 Classification models

Table 6.6 shows the performance of the first four classifiers to be tested on the dataset.

KNeighborsClassifier came out as one of the second-best classifiers tied with the DecisionTreeClassifier for accuracy. Accuracy was reported at 78.12%, with a ROC score of 0.78. A false Negative (FN) was reported at 26, which is important because a false negative could lead to a potential patient with cancer being missed. The metric in which the algorithm did not perform best was false negative which could lead to an individual with cancer being missed.

DecisionTreeClassifier came out second best tied with the KNeighborsClassifier with an accuracy score of 78.12% and a ROC of 0.78. The algorithm did report a False Negative of 18 out of 160 and a False Positive of 17 out of 160. This algorithm would miss 18 out of every 160 cases to classify someone with cancer.

RandomForestClassifier was the best in terms of accuracy, with an accuracy score of 81% and a ROC of 0.81. False negatives came out at 19 out of every 160 patients and would rank the lowest in terms of False negatives.

GaussianNB is one of the lowest-performing algorithms tested with an accuracy score of 59% and a ROC of 0.53, which is only 3 points away from the random chance of getting a classification right. False positives came out at 56 out of every 160 patients.

Table 6.6 shows the performance of the scatter search algorithm. The top graphical probabilistic models have been selected from a population of possible solutions. All three graphs listed in table 6.6 have accompanying tables shown in Figure 6.7, Figure 6.8, and Figure 6.7.

The top-performing graph has an accuracy score of 87.5% and a ROC score of 0.87 False Negative is low at 9 out of 160 samples, and False Positive is 11 out of 160 samples. The graphical model has six vertices (attributes) and five edges.

Graph number 2 has an accuracy of 87.5% and a ROC of 0.87. True Negatives are 69 out of 160 samples which are one higher than the other two graphs. False Positives are more elevated with 11 out of 160 samples and are the highest of the three graphs.

Graph number 3 had an accuracy of 87.5% and a ROC of 0.87. True Negatives are 66 out of 160 samples which is lower than the other two graphs. False Positives are more elevated with 14 out of 160 samples and are the highest of the three graphs.

xy	field
X0	age
X4	use the contraceptive pill
X16	weight
Y1	has cancer (Yes/No)

Table 6.7 Graph 1 nodes

xy	field
X4	use the contraceptive pill
X7	age at 1st child birth
X15	height
Y1	has cancer (Yes/No)

Table 6.8 Graph 2 nodes

xy	field
X0	age
X7	age at 1st child birth
X16	weight
Y1	has cancer (Yes/No)

Table 6.9 Graph 3 nodes

Fig. 6.7 Top graph nodes

6.5 Summary and Conclusions

Much research has gone into both risk-based models and classification methods for ovarian cancer. A common theme in most publications is the ultimate goal of early diagnosis of ovarian cancer to further improve the survival rate of patients. Therefore, nearly all publications focused on lab tests and genetics to predict risk and classify ovarian cancer.

The purpose of the research conducted here was to classify if a patient has ovarian cancer from simple surveying techniques rather than costly lab and genetic testing. Information for the model could be captured by a mobile software application or GP data reducing pressure on health services and allowing quick access to a simple risk assessment.

A total of 438 samples were collected, and with the use of SMOTE oversampling, this could be further increased to 756 samples. The K folds algorithm was utilised to split the training data into training and test data.

Multiple machine learning algorithms were trained on the data, including a newly developed algorithm, called scatter search. RandomForestClassifier classifier was found to be the best algorithm with accuracy at 78%, ROC 0.781, recall 0.812 and precision of 0.0.857. The worst performing algorithm was Gaussian naive Bayes with an accuracy of 53%, ROC 0.531, recall 0.762 and precision 0.0.521. The newly developed scatter search optimised Bayesian network performed well with an accuracy of 87.5%, ROC of 0.875, recall of 0.888 and precision of 0.866.

It has been shown that machine learning from surveyed patient data can indeed make classification predictions and has the potential to be developed further into a software application. Larger sample sizes may further increase the accuracy of machine learning algorithms, with further research being needed.

In contrast to the results in this study, other studies have achieved higher values but with different datasets that include data from clinical testing. A study by Sharmistha Bhattacharjee [21] achieved an accuracy of 99.1% with a sensitivity of 100% and a specificity of 97.9%. A similar study to this one conducted by Munetoshi Akazawa [3] achieved accuracies of 78% with a Random Forest model and 80% with XGBoost. This study also found age, age at menopause and BMI a contributing factors, unlike this study lab tests were also included and found to be significant attributes.

Improvements are needed in this study in terms of sample size, socio-economic and geographical data. Further study would need to be conducted with data from outside of Wales preferably from other parts of the world such as Europe, Asia and the USA.

Chapter 7

Predicting No-Shows for Breast Cancer Follow-Up Visits: An India Perspective

7.1 Introduction

Breast cancer is one of the world's most common female cancers with about 1.4 million cases annually, accounting for nearly a quarter of all cancers in women [59]. More than half of the 500,000 or more breast cancer deaths worldwide occur in low and middle-income countries (LMICs) [59]. Faced with increasing breast cancer incidence rates [83], escalating demand is challenging health care systems in LMICs because, among others, screening policies, particularly in India, have proven to be not cost-effective [137].

Monitoring patients for follow-up using timely and effective approaches is becoming increasingly important. India, in particular, faces a diverse and daunting array of barriers and challenges due to critical factors such as clinical, financial, social, emotional and cultural beliefs and practices that call for new and innovative approaches to patient education and management as well as clinician support. This study aims to address the development and evaluation of better decision models for predicting compliance with breast cancer follow-up requirements for improved care delivery in developing countries. The following research questions will be addressed:

1. Which socio-demographic and socio-economic attributes of patients are responsible for predicting whether or not a patient attends follow-up visits for breast cancer in India?

2. Which classification accuracies, among other metrics, can be achieved when classifying these patients?

We answer these questions by investigating the application of machine learning methods to predict no-shows of breast cancer patients for follow-up treatments. We analyze patient data consisting of clinical, socio-economic and socio-demographic information of more than 300 patient records and nearly 150 variables from an oncology care provider in India. All patients underwent surgical treatment for their condition, hence follow-up care is particularly important for patients to maintain good health, manage side effects, and detect early signs of recurrence. For the care provider, consistent follow-up by patients allows monitoring of clinical and process outcomes and building contextual evidence and knowledge regarding protocols and approaches that are successful for their diverse groups of patients. Hence, the uniqueness of this dataset is two-fold: First, the dataset includes socio-economic and socio-demographic information such as education, occupation and economic status as well as extensive treatment information, and second, the dataset includes patient-level data on breast cancer treatments in an LMIC which has traditionally been a challenge to compile accurately. Our results show that, in general, machine learning approaches can predict whether or not a patient is a no-show with approximately 90% accuracy while the precision of classifying a patient with a no-show as the actual outcome reaches 100%. In a more detailed analysis in which we apply a cost-sensitive classification approach, we observe that an acceptable trade-off can be found using classification accuracy, precision and true positive outcome of no-shows.

For patients who are predicted to not show up for their follow-up appointment, the health system may create incentives to increase the likelihood of attending the appointment depending on the variables which are responsible for the no-show behaviour. Other interventions such as reminder systems [140] could be developed, targeting patients who are classified as no-shows using our machine learning approaches.

The remainder of the chapter is structured as follows. Section 7.2 provides a brief survey of relevant literature. Section 7.3 introduces the specific methods that are evaluated in this study. A performance analysis of these methods is given in Section 7.4, followed by concluding remarks in Section 7.5.

7.2 Related Work

We break down our literature into two streams. First, we review classification techniques and then focus on related work with respect to no-show prediction.

7.2.1 Classification Techniques

Since we deal with no-shows as a discrete attribute, we limit our search for relevant work to classification problems. Textbooks that cover attribute selection and classification are, for example, [182], [23] and [117]. More recently, [70] includes a review of attribute selection and classification techniques with an emphasis on healthcare where applications of both attribute selection and classification include [14], [57] and [128]. The application of machine learning methods in the study of [151] has some similarities with our study. The authors classify patients with an unbalanced class distribution using a variety of standard machine-learning techniques on a variety of measures. Another relevant study is the one of [127]. Similar to our work, they evaluate the effectiveness of a decision tree learner in a care setting. The major difference is that we evaluate a variety of attribute selection and classification techniques on different performance metrics which we trade off using cost-sensitive classification.

7.2.2 No-show Prediction Approaches

The literature on missed appointments shows a poorer outcome for a patient [132] and also leads to unnecessary costs and organizational inefficiencies [45]. Variations exist between countries, patient mix and practice types where missed appointment rates are typically in the range of 2–30% [6]. The prediction of no-shows for individual patients is not a new task but it is increasingly important because opportunity costs of scarce and expensive resources are high. While earlier work of Goldman et al. [78] applied a multivariate logistic regression approach to predict no-shows, more recent work connected no-show prediction with appointment scheduling. Samorani and LaGanga [155] as well as Glowacka et al. [77] use association rules to predict whether or not patients show up for an appointment and evaluate it in a resource allocation setting. Huang and Hanauer [90] predict no-shows using a logistic regression model and simulate an overbooking policy based on the predictions. Multiple methods to predict no-shows are evaluated in Alaeddini et al. [4] while the combination between no-show and cancellation prediction is evaluated by Alaeddini et al. [5]. Similar to our work the authors evaluate decision tree learners, among others, while the difference to our study is that we evaluate socio-economic attributes such as the patient's occupation and carry out a cost-sensitive classification because of our unbalanced dataset.

In conclusion, our study can be considered to be the first to employ attribute selection and classification methods for no-show prediction using data from LMICs in which we trade-off performance metrics by incorporating cost-sensitive classification.

7.3 Methods

We provide a formal description of the no-show classification problem. Let $\mathscr{I}$ denote a set of individuals (breast cancer patients) and let $\mathscr{N}$ denote the set of outcomes. For our classification problem, we have a binary set of outcomes $\mathscr{N} := \{0,1\}$ which means that a patient is either a no-show or not. For each patient $i \in \mathscr{I}$, we observe a set of attributes $\mathscr{A}$ at the time the patient leaves the care provider, while the patient's true outcome, $n_i \in \mathscr{N}$, is computed based on information whether or not the patient re-enters the care-providers practice. Let $\mathscr{V}_a$ denote the set of possible values for attribute $a \in \mathscr{A}$ and let $v_{i,a} \in \mathscr{V}_a$ denote the value of attribute a for patient i. We wish to predict n_i when patient i is a no-show given the patient's values $v_{i,a}$ for each attribute $a \in \mathscr{A}$. Some attribute values $v_{i,a}$ may be missing. In the computational study, we briefly discuss how we handle these missing values (i.e., treating 'unknown' as a separate value). In this supervised learning problem, we assume the availability of labelled training data from many other patients $j \in \mathscr{I} \setminus i$ whose attribute values $v_{j,a}$ and outcomes n_j are known. This training data is used to learn a classification model which is then used for the prediction.

7.3.1 Attribute Ranking and Selection Techniques

Information Gain (IG) and Relief-F provide a quick estimate and ranking of relevant attributes and are techniques that are commonly used in attribute ranking (Hall and Holmes [81]).

Information Gain (IG) and Combined IG (IG*) Attribute Ranking. Given the prior probability $p(n)$ for the no-show outcome $n \in \mathscr{N}$, we can compute the information entropy $H(\mathscr{N})$ and the conditional information entropy $H(\mathscr{N}|a)$ of $\mathscr{N}$ given an attribute $a \in \mathscr{A}$ ([81]). The information entropy $H(\mathscr{N})$ is calculated by Equation (7.1).

$$H(\mathscr{N}) = -\sum_{n \in \mathscr{N}} p(n) \ln p(n) \tag{7.1}$$

The negative sign ensures that $H(\mathscr{N})$ is positive or zero. The more uniformly an attribute value is distributed over all instances, the higher is its entropy (see [23]). Using Equation (7.2), we can compute the conditional information entropy $H(\mathscr{D}|a)$ of $\mathscr{D}$ given an attribute $a \in \mathscr{A}$. Here, $p(v)$ is the prior probability of attribute value $v \in \mathscr{V}_a$ for attribute $a \in \mathscr{A}$ and $p(n|v)$ is the conditional probability of the no-show outcome n given an attribute value $v \in \mathscr{V}_a$ of attribute $a \in \mathscr{A}$.

$$H(\mathscr{N}|a) = -\sum_{v \in \mathscr{V}_a} p(v) \sum_{n \in \mathscr{N}} p(n|v) \ln p(n|v) \tag{7.2}$$

The information gain $IG(a)$ of each attribute $a \in \mathscr{A}$ is then computed by Equation (7.3).

$$IG(a) = H(\mathscr{N}) - H(\mathscr{N}|a) \tag{7.3}$$

An alternative is to generate sub-rankings of attributes by taking into account attribute type information and then combine the two sub-rankings which contain k attributes in total. We denote this approach as 'Combined IG' (IG*).

Relief-F and Combined Relief-F (Relief-F*) Attribute Ranking. [97] have developed a class of algorithms which has been shown to be very efficient for binary classification problems. [150] refined their algorithm and called it Relief-F. The result is a quality measure of each attribute Q_a which can, similar to $IG(a)$, provide an attribute ranking. Like IG*, one can split the attribute ranking into a ranking of i) clinical attributes and ii) a ranking of socio-demographic/-economic attributes which are then combined. Similarly to IG*, we can generate a combined ranking. In the case of Relief-F, we denote this approach 'Combined Relief-F' (Relief-F*).

In order to describe the algorithm we first define the "k-nearest hits" and "k-nearest misses" for a sampled instance $i \in \mathscr{I}$. Let the set of k-nearest hits $\mathscr{H}_i(k) \subset \mathscr{I} \setminus i$ of an instance $i \in \mathscr{I}$ contain at most k instances $j \in \mathscr{I}, j \neq i$ which have the same no show outcome as instance i. More precisely, we choose those instances with $d_j = d_i$ which have the lowest $diff_{i,j}$-values as defined by Eqs. (7.4) and (7.5).

$$diff_{i,j} = \sum_{a \in \mathscr{A}} diff_{i,j,a} \tag{7.4}$$

$$diff_{i,j,a} = \begin{cases} 0, & \text{if} \quad v_{i,a} = v_{j,a} \\ 1, & \text{otherwise} \end{cases} \tag{7.5}$$

Furthermore, for each no show outcome $n \neq n_i$, let the set of k-nearest misses $\mathscr{M}_{n,i}(k) \subset \mathscr{I} \setminus i$ of instance i contain at most k instances $j \in \mathscr{I}, j \neq i$. More precisely, we choose those instances with $n_j = n$ which have the lowest $diff_{i,j}$-values as defined by Eqs. (7.4) and (7.5). Both the k-nearest hits and the k-nearest misses for each no-show outcome $n \in \mathscr{N}$ are used by Equation (7.6) which computes the quality measure Q_a for attribute $a \in \mathscr{A}$.

$$Q_a = \frac{1}{k \cdot |\mathscr{I}|} \sum_{i \in \mathscr{I}} \Big(- \sum_{h \in \mathscr{H}_i(k)} diff_{i,h,a} + \sum_{n \in \mathscr{N} \setminus n_i} \frac{p(n)}{1 - p(n_i)} \sum_{m \in \mathscr{M}_{n,i}(k)} diff_{i,m,a} \Big) \tag{7.6}$$

For each instance $i \in \mathscr{I}$ the k-nearest hits and k-nearest misses for each sampled instance $i \in \mathscr{I}$ are selected and used in Equation (7.6). Then, the attributes with the highest values of the quality measure are considered most relevant for classification.

7.3.2 Classification Techniques

In the following, we summarize four basic classification methods that are well-known in the literature: Naive Bayes (NB), Bayesian networks (BN), decision trees (also called classification trees) (DT) and decision rules. These algorithms were chosen because they represent different approaches to learning and they are relatively fast, state-of-the-art algorithms that are often used in data mining applications [81]. Also, we chose not to evaluate Deep Learning systems because of the lack of explainability and interpretability to clinicians [115]. For each of the four methods, the classifier is learned from a dataset of labelled training examples. This means that the true class of each breast cancer patient is known by the classification method. Afterwards, the classifier is applied to a separate dataset of unlabeled test examples. Here, the true outcome of each patient is unknown to the classification method and must be predicted. The NB and BN methods learn a probabilistic model from the training data, compute the posterior probability that the patient belongs to each outcome d given the patient's attributes $\mathscr{A}$ and assign the patient to the class with the highest posterior probability. The decision tree method, instead, learns a tree-structured set of decision rules from the training data and uses these rules to predict the patient's outcome. Each method is described in more detail below.

Naive Bayes The naive Bayes classifier assumes that all of a patient's attributes $a \in \mathscr{A}$ are conditionally independent given the patient's class value n. Under this assumption, the prior probability $p(n)$ of each class value $n \in \mathscr{N}$ is learned from the training data by maximum likelihood estimation, i.e. $p(n)$ is set equal to the proportion of training examples which belong to class $n \in \mathscr{N}$. Similarly, the conditional likelihood of each attribute value $v_{i,a}$ given each class value $n \in \mathscr{N}$ is learned from the training data by maximum likelihood estimation, i.e. $p(v_{i,a}|n)$ is set equal to the proportion of training examples of class $n \in \mathscr{N}$ which have value $v_{i,a}$ for attribute $a \in \mathscr{A}$.

We assign a new instance i to the no-show outcome n_i^* by employing Equation (7.7).

$$n_i^* = \arg\max_{n \in \mathcal{N}} \left\{ p(n) \prod_{a=1}^{|\mathcal{A}|} p(v_{i,a}|n) \right\} \tag{7.7}$$

The prior probability $p(n)$ of each no-show outcome n is learned from the training data by maximum likelihood estimation, i.e. $p(n)$ is set equal to the proportion of training examples which belong to class n. Similarly, the conditional likelihood of each attribute value $v_{i,a}$ given each no show outcome n is learned from the training data by maximum likelihood estimation, i.e. $p(v_{i,a}|n)$ is set equal to the proportion of training examples of class n which have value $v_{i,a}$ for attribute a.

Bayesian Networks The naive Bayes approach assumes that each attribute is only dependent on the class but not dependent on other attributes, which is rarely true. Thus, we extend the naive Bayes classifier to a Bayesian network classifier, where the set of conditional independence assumptions is encoded in a graph. As in the naive Bayes approach, we learn the conditional probabilities from the training data, but now we must condition not only on the class but also on any other parents Π_a of the given attribute a in the graph.

Similar to the Naive Bayes classification, we assign a new instance breast cancer patient i to the class n_i^* by employing Equation (7.8).

$$n_i^* = \arg\max_{n \in \mathcal{N}} \left\{ p(n) \prod_{a=1}^{|\mathcal{A}|} p(v_{i,a}|n, \Pi_a) \right\}. \tag{7.8}$$

Decision Trees Decision tree learners automatically learn a decision tree from labelled training data. There are various methods to learn the structure of a decision tree from data: We use an algorithm which has been investigated by Hall and Holmes [81] in combination with attribute selection. We employ this algorithm because we can control the over-fitting of the decision tree as well as the tree size during the learning process.

Decision Rules A simple decision rule can be stated as follows [81]: In the training set, we count how often a value of a certain attribute occurred with respect to each class attribute value. For each value, we create a mapping to the most frequent class. Now, for each instance in the testing set, we assign the instance to the class which is described by the decision rule and the observed attribute value.

7.4 Experimental Investigation

In the following, we provide an experimental investigation of the presented methods. We first give an overview of the data employed in our study, followed by a presentation of the attribute selection results and an evaluation of the classification techniques.

7.4.1 Study Design

We first carried out a semantic preprocessing of the free-text attributes. The number of binary attributes which were obtained by this procedure are shown in Table 7.1(bottom). We evaluated the attribute ranking techniques including IG, IG with the top 5 clinical and the top 5 demographic attributes (IG*), Relief-F and Relief-F*. For both IG variants, we removed missing values (NAs) from the class variable $\mathscr{N}$. Afterwards, we carried out a parameter optimization for the decision tree learner. The classification after attribute selection is compared to the one without attribute selection (w/o AS).

7.4.2 Data and Information Documented for No-Show Prediction

We evaluated the attribute selection and classification techniques using data from 336 breast cancer patients in Kerala, India. Summary statistics for our data set are provided in Table 7.1(top). For some patients, there exists free-text information. In order to employ this information, we manually processed the free-text and converted it into binary attributes. We observed a relative no-show rate of breast cancer patients of approximately 15% in this study setting which is lower than no-show rates of cancer patients and other patient groups [122, 6, 146].

The data was taken from a secondary care provider database in 2018. The outcome variable is binary (1 for no-show and 0 for attendance). Models are trained on this outcome variable. There are 144 predictor variables that vary from binary classes to multi-classes. A maximum sample size of 358 was pulled from the database from all available information and all information was anonymised after cleaning the data it was found 22 records were too incomplete to be used, and the final sample size was 336.

# patients	336
# socio-economic and -demographic attributes	12
# $\|\{n_i \in \mathscr{N} : n_i = 1\}\|$	52
# $\|\{n_i \in \mathscr{N} : n_i = 0\}\|$	284

Table 7.1 Summary statistics (top), #original and selected attributes (middle) and the result of our semantic processing of free-text (bottom)

7.4.3 Exploratory Data Analysis

Figure 7.1(top) shows a mosaic plot of the class attribute (no-shows) broken down by one key socio-economic attribute: "Occupation". The figure reveals that the attribute value "House wife" contains a higher proportion of patients who belong to the outcome 'no-shows'. The occupation 'teacher' does not reveal substantial differences in the proportion of no-shows or no no-shows. However, 'other' as occupation has differences in the proportion of no-shows which, however, is not significant because of the low number of patients belonging to that group. Figure 7.1(bottom) shows a similar plot for the attribute "Recommended Treatment". One can observe that the proportion of patients who received no treatment and are 'no shows' is not substantially different from the one belonging to no 'no show'. However, patients having only surgery as a treatment are much more likely for no-shows as compared to the group of patients with surgery and chemotherapy (CT) or CT, surgery and radiotherapy (RT). Patients who had both surgery and RT, or both non-adjuvant chemotherapy (NACT) and surgery have a higher likelihood to be in the group of no-shows.

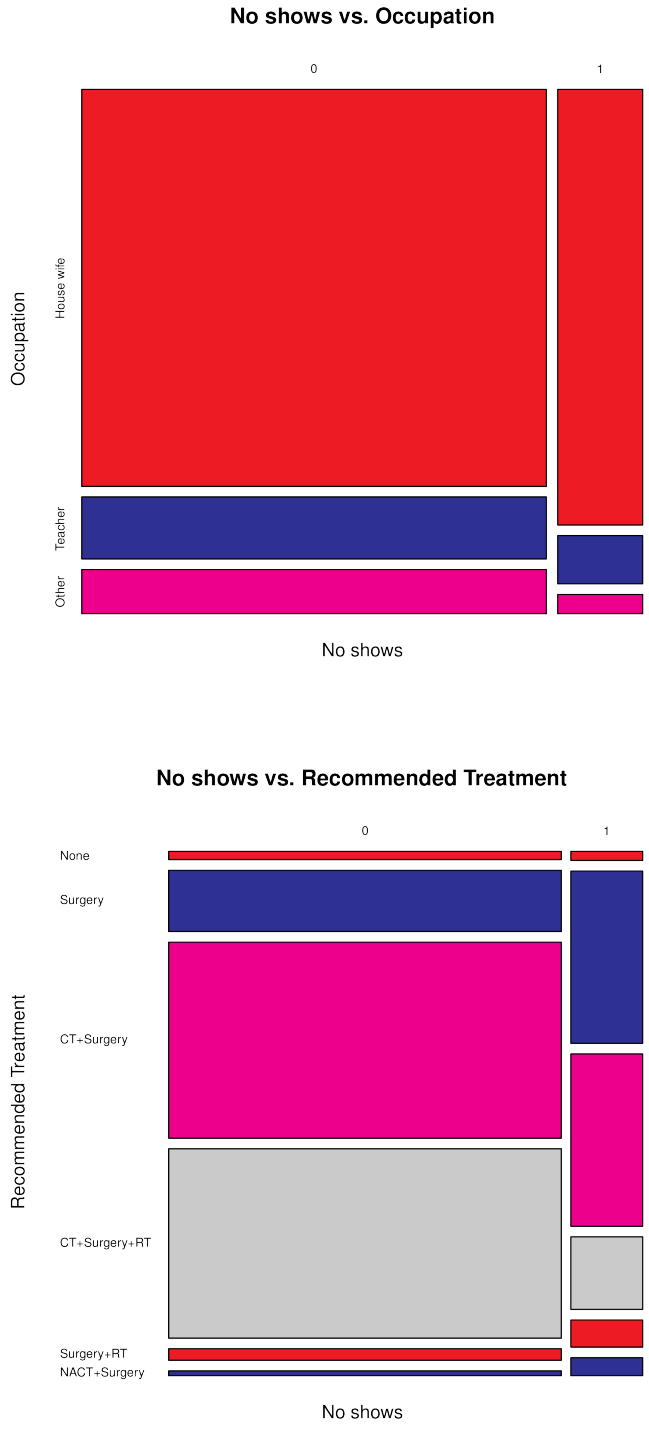


Fig. 7.1 No-show vs. occupation (top) and No-show vs. recommended treatment (bottom)

7.4.4 Attribute Ranking Results

Table 7.2 provides the results of the attribute rankings where the nearest-neighbour parameter of the Relief-F algorithm is set to $k = 10$, as suggested by Robnik-Šikonja and Kononenko [150].

Table 7.2 Results of the top ten attributes determined by IG (top left), Relief-F (top right), IG*(bottom left), Relief-F*(bottom right)

Rank	Attribute name	IG
1	Core Biopsy	0.107
2	Multifocal	0.075
3	Number of Treatments Recommended	0.075
4	Hormol Therapy (tamoxifen)	0.074
5	Recurrence? Y/N	0.059
6	Nodes documented in distance recurrence	0.053
7	Targeted Therapy (herceptin)	0.051
8	Treatment of Recurrence: Hormontherapy	0.050
9	Axillary Level	0.048
10	Site of distance recurrence: Pleura	0.045

Rank	Attribute name	Q_a
1	Number of Treatments Received	0.146
2	Number of Treatments Recommended	0.140
3	HERII/NEU Status	0.134
4	Recommended Treatment	0.122
5	Radiation Type	0.116
6	PR Status	0.109
7	Chronic illness (yes/no)	0.101
8	Hormol Therapy (tamoxifen)	0.088
9	ER Status	0.087
10	Tumor Histology	0.087

Rank	Attribute name	IG
1	Tumor size (cm)	0.342
2	Tumor volume (cm3)	0.216
3	pTNM	0.101
4	Size of palpable mass (cm)	0.070
5	Hormol therapy (tamoxifen)	0.050
1	Occupation	0.048
2	Economic Status	0.043
3	Education group	0.033
4	Parity	0.018
5	Type of Health Insurance	0.015

Rank	Attribute name	Q_a
1	Recommended treatment	0.192
2	Number of treatments received	0.170
3	Number of treatments recommended	0.162
4	Radiation type	0.160
5	Radiation dose	0.141
1	Number of Children	0.069
2	Multiple Children	0.061
3	Multiple Parity	0.057
4	Economic Status	0.028
5	Parity	0.023

Both attribute rankings, IG and Relief-F, indicate in a quantitative manner (based on IG and Q_a) that attributes such as "Hormol therapy (tamoxifen)", "Recommended treatment" and "Radiation type" belong to the top ten significant attributes. One explanation for this phenomenon is that chemotherapy is a challenging experience for the patient and as a consequence, they may not want to return to the same care provider. Moreover, if patients have chronic illnesses or other co-morbidities, they may simply be unable to show up for follow-up treatments because of their severe health conditions.

7.4.5 Attribute Selection Results

Summary of the Attribute Selection.

The attribute ranking and selection methods identify not only clinical information but also socio-economic information as relevant for classifying whether or not a patient is a no-show. Interestingly, the socio-demographic attribute 'age' was not among the top-ranked attributes although it was discovered to be significant in Parikh et al. [140]. One explanation for this phenomenon is that in our study setting, the age of patients has a lower standard deviation as compared to general settings and therefore the attribute is less informative. Other significant factors such as gender and specialty which were identified by Parikh et al. [140] and Lee et al. [106], respectively, are for our study setting irrelevant because we focus on female and oncology patients, only.

We close the attribute ranking and selection discussion with a summary statistic of the selected attributes in Table 7.1(middle) in which the binary class attribute is included in the numbers. It shows the original and selected number of attributes that we will choose for the classification analysis.

7.4.6 Classification Results

Evaluation Metrics.

All classifiers are assessed using the same performance indicators. The overall performance is measured in terms of classification accuracy which is the proportion of correctly classified patients. Accuracy, however, is a metric which is of lower interest for the care provider in India because the goal is to classify those patients who belong to the group of true no-shows. As a consequence, we report no-show precision and in our case, the metric is defined as the proportion of cases classified as belonging to the true "no-show" outcome that is correctly classified. Finally, we report the true positive (TP) rate of the "no-show" outcome. All performance indicators are measured using 10-fold cross-validation.

ML Model	Attribute selection method				
	w/o	IG	Relief-F	IG*	Relief-F*
KNeighborsClassifier	84.3	84.2	85.1	85.1	87.2
DecisionTreeClassifier	89.2	87.7	86.6	90.4	84.8
RandomForestClassifier	85.2	85.1	86.6	85.4	86.6
GaussianNB	66.9	36.0	83.0	81.8	80.0
Scatter Search	86.3	87.2	86.9	86.3	87.2

ML Model	Attribute selection method				
	w/o	IG	Relief-F	IG*	Relief-F*
KNeighborsClassifier	51.5	52.3	55.0	69.5	87.2
DecisionTreeClassifier	68.1	61.7	62.9	52.0	52.2
RandomForestClassifier	100.	66.6	81.8	81.8	81.8
GaussianNB	26.7	19.0	42.8	34.0	34.0
Scatter Search	87.5	90.9	83.3	84.6	84.6

ML Model	Attribute selection method				
	w/o	IG	Relief-F	IG*	Relief-F*
KNeighborsClassifier	32.6	21.1	21.1	36.5	30.7
DecisionTreeClassifier	57.6	55.7	32.6	59.6	25.0
RandomForestClassifier	03.8	07.6	17.3	05.7	17.3
GaussianNB	65.3	96.1	28.8	34.6	30.7
Scatter Search	13.4	19.2	19.2	13.4	21.1

Table 7.3 Accuracy (top), no-show precision (middle) and sensitivity (bottom).

Overall Accuracy, No-Show Precision and No-Show Sensitivity.

Table 7.3(top) shows the results using this metric and reveals that attribute selection can improve classification accuracy from 86.3 to 87.2% for scatter search. A detailed analysis of the decision rule learner revealed that after Relief-F attribute selection, the attribute "Hormol therapy" is used to create the decision rules.

7.4.7 Discussion, Limitations and Generalizability of the Results

Our results can generalize approaches that have been applied in the literature from a methodological and applications point of view. First, from a methodological point of view, our Relief-F* and IG* mixed attribute ranking methods can be used in contexts where clinical and socio-economic/-demographic factors can be separated. Second, from an applications perspective, our results can extend reminder systems [175]. Patients with a higher likelihood of no-shows could be informed by a reminder application about the medical necessity of the

follow-up treatment [80]. At the same time, patients can be educated e.g. about the risk of poor outcomes if the follow-up treatment is missed.

A consistent finding of our exploratory data analysis, the attribute ranking and is that the attribute "Occupation" is a promising attribute to explain no-shows. Further socio-economic or -demographic attributes including "economic status", "education status" or "religion" were discovered to be important as revealed by the attribute ranking techniques. However, the attribute "Occupation" may be considered redundant with "education status" or "economic status" because the occupation of a patient informs about the education and economic status. This may lead to the generalization that low incomes and a low education status are a predictor for the patients' occupation which, in turn, can be considered as a predictor of whether or not patients attend follow-up visits. In predicting true no show the results show 90% yet we only find 19% of them.

Psychological factors which were not collected in our data set may be important dependent variables in connection with no-show behaviour. For example, patients who have had a breast resection may not want to come back to the hospital because of the severe treatment effects and the possible psychological consequences. Patients may refuse additional treatments and could count as no-shows, too.

7.5 Summary and Conclusions

In this chapter, we have answered the following research questions:

1. Which socio-demographic and socio-economic attributes of patients are responsible for predicting whether or not a patient attends follow-up visits for breast cancer in India?

2. Which classification accuracies, among other metrics, can be achieved when classifying these patients?

The results from our attribute ranking methods indicated that attributes such as 'Hormol therapy (tamoxifen)´, 'Recommended treatment´ and 'Radiation type´ belong to the top ten significant attributes. Our analysis of the socio-economic attributes revealed that "Occupation" is a promising attribute to explain no-shows. In addition, the patients' "economic status", "education status" or "religion" were discovered important as revealed by the attribute ranking techniques. These insights answered the first research question.

The results from our classification experiment showed that a classification accuracy of 90.4% can be achieved while no-show precision and no-show sensitivity were 100.0% and 96.1%.

Our analysis of the socio-economic attributes lead to policy decisions such as granting better access to follow-up visits for patients with a certain income and education. More specifically, for patients with low income, monetary incentives or free treatments may be promising but are evaluated in future research. Also, for patients with a low education level, early education may help to reduce no-shows.

Especially since Bayesian approaches turned out to outperform the rule- and tree-based approaches, future work will examine hybrid methods such as Markov blanket attribute selection and classification. In addition, self-organizing maps will be evaluated to explore, besides socio-demographic and socio-economic attributes, other variables that may explain no-show behavior. Moreover, the improved prediction models will be deployed in a mobile health application to allow care providers to reach out to their no-show patients more effectively [80].

Chapter 8

Conclusions and discussion

8.1 Conclusions

8.1.1 Literature Review

The literature review found a lot of literature on Bayesian networks and machine learning in healthcare. All literature found was broken down into the following sections; previous literature reviews, applications in healthcare, attribute selection and classification, Bayesian network learning types, performance metrics, test statistics and books focused on Bayesian networks, data science and machine learning.

Previous literature reviews have focused on the implementation and adoption of Bayesian networks in healthcare. It was found that wide-scale adoption of Bayesian networks (BN) in a healthcare setting has still not happened, and the literature on the subject of BN in healthcare was limited to purely initial research that focuses on the technical aspects of Bayesian networks for healthcare.

Summary statistics show that a total of 59 articles of relevance for Bayesian networks in health for classification were found. Starting in 1989 and ending the search in 2021, the number of articles has increased by a small amount over 5-year periods. The number of articles from America far exceeded articles from individual continents such as Asia and Europe, although excluding those and combining the rest of the world, 20 articles were found.

A total of 10 papers with a core focus on the implementation in healthcare were found, with the most common theme being risk prediction in healthcare settings. Construction methods for Bayesian networks in healthcare were another common theme followed by feature selection. Feature selection in healthcare had a further focus on dimensionality

reduction for large sparse datasets. The common method for feature reduction was the use of Markov blankets.

Attribute selection and classification have been one of the key research topics in this thesis. The literature review found 7 key publications, with four of those that have a primary focus on dimensionality reduction. Two of these publications conducted research into filtering methods for the reduction of attributes.

There are many ways to train or learn the structure and parameters of a Bayesian network. Publications in this area were divided into three categories, constraint-based, scoring based and hybrid methods. Scoring-based methods were by far the most popular, with K2, TABU Search, TAN and hill climbing being the most popular, with Scatter Search, Ant colony and the Grow Shrink Algorithm being the most under-researched. Constright-based methods had fewer publications than scoring based within healthcare research; most had no publications that came up in the Scopus search. A constraint-based scatter search had zero publications in healthcare and none outside of healthcare. The only paper found for scatter search, and Bayesian networks was a scoring-based method.

Metrics are used to evaluate the performance of a Bayesian network but can also be used to guide the learning algorithm. A total of 21 metrics were found with accuracy, K2, BDeu, BIC, AUC-ROC, Number of attributes and TPR being the most common in the literature. Statistical tests used in training were Chi-Squared, G-Test, B-Statistic, Kernel density estimation and the Z-test.

8.1.2 Bayesian Networks

The Bayesian network allows conditional probability to be represented as a graph with vertices and edges. In the network, vertices represent probability distributions and edges conditional independencies. The building or learning of a Bayesian network consists of two parts learning the structure and learning the parameters of the probability distribution. The probability distribution of a node can be parametric or non-parametric. There are two main ways to search for the structure constraint-based or scoring based. Constright-based uses statistical tests to find the structure and scoring-based searches for the structure and scores the network as a whole using a scoring function. Hybrid methods exist that use both constraint-based and scoring-based methods. Two main approaches exist to estimating the probability distributions these are Bayesian estimation and maximum likelihood estimation.

Inference with Bayesian networks can be made in two ways, exact and approximate. Exact inference algorithms work through the network to calculate an exact singular probability value but approximate inference algorithms return a range of probability values between 0 and 1. An example of an exact algorithm is the junction tree algorithm which turns the

Bayesian network into a tree and works through it given some evidence to calculate the exact value. An approximate algorithm is the Markov chain Monte Carlo (MCMC) algorithm. This algorithm repeatedly samples the Bayesian network to determine probable values given some evidence.

One of the key research questions is how to reduce the number of attributes required for inference in a model so that only the key attributes or variables remain. The solution to this comes in the form of a Markov Blanket which takes only a subset of variables connected to the predictor variable. This is called the Markov boundary and can greatly reduce the number of variables in the model, therefore, overcoming the curse of dimensionality.

Performance metrics are important to judge the Bayesian network's ability to make probabilistic predictions. One of the most common metrics shown to be used in the literature review is accuracy, although this metric is sensitive to unbalanced data. To better understand this metric, other metrics can be used but are not limited to the true positives, false positives, true negatives, and false negatives, which, when used together, form a confusion matrix. This confusion matrix can be used to calculate metrics such as Recall, Precision, Receiver operating characteristics (ROC) and Area under the curse (AUC-ROC). Search heuristics are used to find the structure of a Bayesian network, with the three most common of these being Hill climbing, Simulated Annealing and Tabu Search. These search heuristics use a search strategy to work through the attributes systematically to find the conditional indecencies in the Bayesian network. Networks of nodes are built up and sometimes disregarded as better solutions are found.

8.1.3 The Scatter Search Algorithm

During the literature review, it was found the scatter search algorithm is completely underdeveloped for constraint-based structure searching for Bayesian networks. The algorithm was originally developed by Glover and offered great potential for selecting not only good solutions to a network-based problem but also diverse solutions, which stops the algorithm from getting trapped in local optima. Having shown promising potential in alleviating the local optima problem, this algorithm was chosen for the development of a Bayesian network search algorithm.

The scatter search algorithm is made up of seven main components, which are the Diversification method, Population set, RefSet Update method, RefSet, Subset Generation method, Subset of solutions and combination method. An initial set of solutions is generated by comparing possible combinations of attributes (nodes) and then added to the population set. A subset of both diverse and good solutions is selected and moved into the RefSet. From this, RefSet solutions are combined and statistically tested for significance before being

moved back into the RefSet. Only solutions with a statistical significance value above a threshold are allowed back into the RefSet, this continues until no more solutions are found. Following this procedure, a Bayesian Network is built up from initial pairs of nodes.

8.1.4 Benchmarking the Scatter Search on Healthcare Datasets

Once the scatter search algorithm was constructed, a collection of low sample size classification healthcare datasets were chosen. The lowest number of attributes in these datasets was 7, and the highest had 71 attributes with a low sample size of 91 and the highest being 699. Once chosen, the algorithm had two goals reduce the number of attributes to only the ones needed and achieve the highest accuracy possible without overfitting. Continuous variables were discretized before the tests began.

After running the algorithm, the most significant reduction in variables was the audiology dataset, with 71 variables reduced to 6, with the smallest reduction being acute inflammations from 8 to 5. In terms of attribute selection and reduction, the algorithm performed well in reducing the number of variables across the datasets. Accuracy, Recall and Precision were found to be competitive with other machine learning algorithms with the advantage the relationships between variables can be graphed and viewed.

8.1.5 Ovarian Cancer Risk Classification

With Ovarian cancer being the fifth most common cancer in women being able to detect it early could save lives with early treatment. Working with the Aneurin Bevan University Health Board, a case study was put together to gather questionnaire data from patients. The data collected had both patients with and without ovarian cancer so that a control group could be established. This data was then processed from spreadsheets into a dataset compatible with machine learning algorithms. Included in these algorithms was the Bayesian network scatter search. The goal of the machine learning algorithms is to classify if a patient could have ovarian cancer. The models could then be used in applications such as mobile applications or integrated into a GP system.

The dataset had 30 variables with one binary variable to be predicted. Attributes were ranked by p values with the Chi-squared and ANOVA tests as well as the Mutual information score. A heat map was used to visualise the correlations. Five variables always came top of the ranking these were age, years on the contraceptive pill, do you take the contraceptive pill, irregular periods and having children.

The machine learning algorithms chosen were k-nearest neighbours, gaussian naive Bayes, bayesian networks, and decision trees. K-nearest neighbours were the best-performing

algorithm with an accuracy of 88%, recall of 0.88 and precision of 0.94. The worst performing algorithm was Gaussian Naive Bayes with an accuracy of 57%, recall of 0.82 and precision of 0.55. The Scatter search had an accuracy of 83%, recall of 0.81 and precision of 0.84. Overall, the scatter search algorithm was competitive with other machine learning techniques.

The no-shows Indian data sensitivity metrics showed that without any filtering the GaussianNB classifier performed best followed best the Decision tree classifier. Unlike the Decision Tree classifier the GaussianNB classifier benefited from Information Gain parameter filtering which significantly improved the sensitivity, however, other attribute selection methods reduced the sensitivity for GaussianNB. The Decision Tree classifier did benefit from Information Gain mixed ranking which increased 4 points. Using mixed ranking did not show improvements in sensitivity across most of the classifiers.

It has been shown in this case study a machine-learning algorithm could be used in further research into the early prediction of ovarian cancer.

8.1.6 Predicting No-shows for Breast Cancer Follow-up Visits

Patients in India who have breast cancer surgery can be at risk of not showing up to follow-up visits which are important in continued treatment and detection of re-occurring breast cancer. Many factors such as clinical, financial, social, emotional and cultural factors were taken into account for this study. Using the data collected, this study looks into which factors are most important and if a machine learning algorithm can detect those most at risk to offer more support to improve the life of the patient.

The data collected was processed and reshaped for better use of the intended machine learning algorithms. The variables were broken down into clinical, socio-demographic and socio-economic factors. These attributes were then ranked using Information Gain (IG) and Relief-F. In using the rankings, the dataset attributes can be reduced in size using only the top-ranking attributes. Ten machine-learning algorithms were used. These were K-Neighbors Classifier, SVC-Linear, SVC, Gaussian Process Classifier, Decision Tree Classifier, Random Forest Classifier, Gaussian NB, Quadratic Discriminant Analysis and Scatter Search. All algorithms were trained on all attributes as well as attributes that only made the top ten rankings. It was found that ranking and restricting the attributes could help some machine learning algorithms. The top performing algorithm was SVC, with an accuracy of %92.2 and the worst GaussianNB with %66.9.

8.2 Discussion

Dimensionality reduction and structure learning via statistical tests with applications in health were key questions that were answered by this thesis with a focus on healthcare. During the search, it was found that scatter search with Bayesian networks was partially underdeveloped even though it had the potential to alleviate the problem of getting trapped in local optima. Only one full implementation of scatter search with Bayesian networks had been researched, but this was a scoring-based approach. During a wider, more broad search, still, no research was found for a constraint-based method using statistical tests. A constraint-based structure learning approach was taken during this thesis using statistical tests, the advantage of this being each pairing of attributes will have an associated p-value, therefore a better understanding of the relationship between attributes in the model. Many other algorithms have been widely developed, such as hill climbing and TABU search with both scoring and constraint-based methods.

The application of this machine learning technique was for small sample sizes that could potentially have many attributes. This could lead to a potential problem with the curse of dimensionality, which states that the data needed grows exponentially with the increasing number of attributes. During the literature search, it was found that Bayesian networks had been used for dimensionality reduction by taking the Markov blanket of the Bayesian network. When finding the Markov boundary and taking the Markov blanket of nodes, the smaller network contained only the necessary probability distributions to make predictions given some evidence. This technique allowed the Bayesian network to be learned from small sample sizes and avoid overfitting the model.

Once a scatter search constraint-based method has been developed, a better understanding of the performance of the model was needed in relation to other models on well-known healthcare datasets. Classification healthcare datasets from the UCI machine learning repository were chosen as the intended applications for this algorithm are in the healthcare domain. Benchmarking against other classification algorithms showed competitive results whilst retaining the ability to understand the model and relationships that had been discovered in the data.

Two further case studies were undertaken, one in ovarian cancer classification and the second in breast cancer follow-up no-show classification. Both of these studies had a binary outcome variable to be predicted. The first case study to be discussed is the ovarian cancer study. All data were collected via patients and put into many spreadsheets by clinical staff using a questionnaire. This data then had to be combined and transformed into a format best for the machine learning algorithms. Continuous data such as height was discretized, and yes/no answers were transformed into binary variables. To best understand the data, a

correlation heatmap was produced. Once the correlations were understood, the attributes were ranked according to p values. Classification machine learning algorithms were run against the data along with the Bayesian network scatter search. Of these, k-nearest neighbours were the best-performing algorithm and the scatter search algorithm also was competitive and scored well. An accuracy of 88% was achieved, with other metrics such as recall and precision showing good performance. The study showed it is possible to predict with reasonable accuracy if a patient may have ovarian cancer and need further investigation.

The final study conducted looked into patients at risk of no-shows to a follow-up breast cancer appointment in India. The data collected was broken down into three areas. These areas are clinical, socio-demographic and socio-economic. The variable to show or not to show is a binary variable. All data collected in the study needed to be transformed and balanced before machine learning algorithms could be run. Once balanced, statistical testing and mutual information techniques were used to rank the variables. From this, new datasets were formed with only the highest-ranking variables. All datasets with only high-ranking variables and with all attributes included were run with machine learning algorithms. In selecting only high-ranking variables, it was hoped that accuracy would improve as the noise had been removed from the data. Whilst this turned out to be true for some algorithms, no significate improvement was reported for a lot of the algorithms. The best-performing algorithm was SVC, with scatter search having competitive results and outperforming others such as naive Bayes.

Although the scatter search algorithm did not outperform all algorithms, the results were competitive and by far not the worst. The strength of the scatter search is the ability to output the structure of the learned attributes to understand better how attributes relate to each other and which ones are important when making decisions. Statistical tests have been shown to have the ability to find relationships, i.e. the structure of Bayesian networks. These tests can guide a search algorithm such as Scatter Search with a p-value of between 0.01 and 0.05. Although it was found for more diverse solutions, a temporally higher p-value can be used.

It has been concluded that statistical testing with a guided search such as scatter search can indeed learn the structure of a Bayesian network and, with the use of a Markov blanket, reduce the number of attributes in the model. Reduced-size Bayesian networks can indeed be applied to small healthcare datasets and produce acceptable results in terms of accuracy, precision and recall. A small Bayesian network with only the essential variables needed for the prediction of a binary variable whilst remaining competitive with other machine learning methods is possible.

8.3 Future Work

The ovarian cancer study could use a larger sample size from another group of patients to validate the results. One problem with this is ovarian cancer has a high mortality rate due to the late detection of the disease and finding people still alive who've had ovarian cancer to take the survey is difficult. The data is also limited to South East Wales and a larger population from across the world would also be beneficial to understand if there is any bias in the data. Models generated from the ovarian cancer study need further external validation and exposure to larger datasets gathered from around the world. The data was of small sample size and limited to South East Wales. CPRD is the clinical practice research datalink database that contains primary care data, this could be a good source of data to further train the model on. CPRD is limited to data from England. An international effort to further collect the required data for validation by professionals in the field would still be needed. Once complete survival analysis on patients that have had early detection and intervention because of the model would be needed to judge the overall effectiveness and usefulness of the model.

A similar case could be made for the no-show breast cancer models that, if fully validated, could be used in a clinical setting to help those missing appointments and hopefully improve healthcare outcomes.

All Bayesian networks that have been generated have had to use discrete data due to the conditional probability tables used. Further research is needed to develop the algorithm to work with continuous distributions and data. Discretizing data leads to a loss of information and performance issues when many groups of data are created. A large number of groupings in data is also impractical because of system memory constraints. In order to use continuous data we need to use parametric probability distributions instead of contingency tables which can become large over many discrete values. The output of one continuous probability distribution could be transformed into a mathematical function that would form the input of another distribution's parameters.

8.4 Key Contributions

The aim of this thesis was to develop a novel way to learn a Bayesian network structure using constraint-based methods in an evolutionary approach. By taking this approach the Bayesian network could be built up from many smaller statistically significant smaller networks. It was also found that using only statistically significant networks, learners could get stuck in local minima. The approach was to try to apply the scatter search algorithm to find many

small solutions to the problem and then combine them. These solutions could be statistically significant (good solutions) or solutions that had p-values less than 0.05 (diverse solutions). By allowing solutions that weren't the best it was found that traps of local minima could be minimized.

Once a successful algorithm was developed it was applied to healthcare datasets that had a discrete output variable. The algorithm was able to produce good results in comparison with other machine learning algorithms. The results of the algorithms were competitive and offered a significant contribution to the domain of classification algorithms in healthcare.

References

[1] Abidi, S., Roy, P., Shah, M., Yu, J., and Yan, S. (2018). A data mining framework for glaucoma decision support based on optic nerve image analysis using machine learning methods. *Journal of Healthcare Informatics Research*, 2(4):370–401. cited By 5.

[2] Akazawa, M. and Hashimoto, K. (2020a). Artificial intelligence in ovarian cancer diagnosis. *Anticancer research*, 40(8):4795–4800.

[3] Akazawa, M. and Hashimoto, K. (2020b). Artificial intelligence in ovarian cancer diagnosis. *Anticancer research*, 40(8):4795–4800.

[4] Alaeddini, A., Yang, K., Reddy, C., and Yu, S. (2011). A probabilistic model for predicting the probability of no-show in hospital appointments. *Health Care Management Science*, 14(2):146–157.

[5] Alaeddini, A., Yang, K., Reeves, P., and Reddy, C. K. (2015). A hybrid prediction model for no-shows and cancellations of outpatient appointments. *IIE Transactions on Healthcare Systems Engineering*, 5(1):14–32.

[6] Alhamad, Z. (2013). Reasons for missing appointments in general clinics of primary health care center in Riyadh military hospital, Saudi Arabia. *International Journal of Medical Science and Public Health*, 2(2):258–267.

[7] Ankan, A. and Panda, A. (2015a). *Mastering Probabilistic Graphical Models Using Python*. Packt Publishing.

[8] Ankan, A. and Panda, A. (2015b). pgmpy: Probabilistic graphical models using python. In *Proceedings of the 14th Python in Science Conference (SCIPY 2015)*. Citeseer.

[9] Antal, P., Verrelst, H., Timmerman, D., Moreau, Y., Van Huffel, S., De Moor, B., and Vergote, I. (2000). Bayesian networks in ovarian cancer diagnosis: potentials and limitations. In *Proceedings 13th IEEE Symposium on Computer-Based Medical Systems. CBMS 2000*, pages 103–108. IEEE.

[10] Arfiani, A. and Rustam, Z. (2019). Ovarian cancer data classification using bagging and random forest. In *AIP Conference Proceedings*, volume 2168, page 020046. AIP Publishing LLC.

[11] Arora, P., Boyne, D., Slater, J., Gupta, A., Brenner, D., and Druzdzel, M. (2019). Bayesian networks for risk prediction using real-world data: A tool for precision medicine. *Value in Health*, 22(4):439–445. cited By 27.

[12] Bai, X. (2005). Tabu search enhanced graphical models for classification of high dimensional data. In *Technical Report CMU-CALD-05-101*. School of Computer Science, Carnegie Mellon University.

[13] Bai, X. and Padman, R. (2005). Tabu search enhanced markov blanket classifier for high dimensional data sets. *Operations Research/ Computer Science Interfaces Series*, 29:337–354. cited By 9.

[14] Bai, X., Padman, R., Ramsey, J., and Spirtes, P. (2008a). Tabu search-enhanced graphical models for classification in high dimensions. *INFORMS Journal on Computing*, 20(3):423–437. cited By 16.

[15] Bai, X., Padman, R., Ramsey, J., and Spirtes, P. (2008b). Tabu search-enhanced graphical models for classification in high dimensions. *INFORMS Journal on Computing*, 20(3):423–437.

[16] Bareiss, E., Porter, B., and Wier, C. (1987). Protos: an exemplar-based learning apprentice. In *Proceedings of the Fourth International Machine Learning Workshop*, pages 12–23.

[17] Bellman, R. and Kalaba, R. (1959). A mathematical theory of adaptive control processes. *Proceedings of the National Academy of Sciences*, 45(8):1288–1290.

[18] Bertagnolli, N. (2022a). introduction to bayesian networks.

[19] Bertagnolli, N. (2022b). introduction to bayesian networks - bayesian network.

[20] Bhattacharjee, S., Singh, Y. J., and Ray, D. (2017a). Comparative performance analysis of machine learning classifiers on ovarian cancer dataset. In *2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 213–218. IEEE.

[21] Bhattacharjee, S., Singh, Y. J., and Ray, D. (2017b). Comparative performance analysis of machine learning classifiers on ovarian cancer dataset. In *2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 213–218. IEEE.

[22] Bielza, C. and Larrañaga, P. (2014). Discrete bayesian network classifiers: A survey. *ACM Computing Surveys*, 47(1). cited By 128.

[23] Bishop, C. (2006). *Pattern recognition and machine learning*. Springer, New York.

[24] Bonaccorso, G. (2017). *Machine learning algorithms: reference guide for popular algorithms for data science and machine learning*. Packt.

[25] Borgelt, C. and Kruse, R. (2001). An empirical investigation of the k2 metric. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 240–251. Springer.

[26] Boslaugh, S. (2012). *Statistics in a Nutshell*. In a nutshell. O'Reilly Media, Incorporated.

[27] Brownlee, J. (2016). *Machine Learning Algorithms From Scratch with Python*. Machine Learning Mastery.

[28] Brownlee, J. (2020). *Imbalanced Classification with Python: Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning*. Machine Learning Mastery.

[29] Bruce, P. and Bruce, A. (2017). *Practical Statistics for Data Scientists: 50 Essential Concepts*. O'Reilly Media.

[30] Cai, Z., Si, S., Sun, S., and Dui, H. (2015). Learning bayesian network structure with immune algorithm. *Journal of Systems Engineering and Electronics*, 26(2):282–291. cited By 3.

[31] Campos, L. d. (2006). A scoring function for learning bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7(Oct):2149–2187.

[32] Cano, R., Sordo, C., and Gutiérrez, J. M. (2004). Applications of bayesian networks in meteorology. In *Advances in Bayesian networks*, pages 309–328. Springer.

[33] Cao, Y., Raoof, M., Szabo, E., Ottosson, J., and Näslund, I. (2020). Using bayesian networks to predict long-term health-related quality of life and comorbidity after bariatric surgery: A study based on the scandinavian obesity surgery registry. *Journal of Clinical Medicine*, 9(6):1–12. cited By 2.

[34] Chen, L. (2009). *Curse of Dimensionality*, pages 545–546. Springer US, Boston, MA.

[35] Cheng, J. and Greiner, R. (1999). Comparing bayesian network classifiers. *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, UAI'99*, pages 101–108. cited By 304.

[36] Chickering, D. (2002a). Learning equivalence classes of bayesian-network structures. *Journal of Machine Learning Research*, 2(3):445–498. cited By 360.

[37] Chickering, D. (2002b). Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554.

[38] Chickering, D., Geiger, D., and Heckerman, D. (1995). Learning bayesian networks: Search methods and experimental results. *Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 112–128. cited By 120.

[39] Chickering, D., Heckerman, D., and Meek, C. (2004). Large-sample learning of bayesian networks is np-hard. *Journal of Machine Learning Research*, 5:1287–1330. cited By 351.

[40] Chickering, M. (2020). Statistically efficient greedy equivalence search. In *Conference on Uncertainty in Artificial Intelligence*, pages 241–249. PMLR.

[41] Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467. cited By 1631.

[42] Coleman, M., Forman, D., Bryant, H., Butler, J., Rachet, B., Maringe, C., Nur, U., Tracey, E., Coory, M., Hatcher, J., et al. (2011). Cancer survival in australia, canada, denmark, norway, sweden, and the uk, 1995–2007 (the international cancer benchmarking partnership): an analysis of population-based cancer registry data. *The Lancet*, 377(9760):127–138.

[43] Commission, E. (2023). Ethics guidelines for trustworthy ai. *Shaping Europes digital future*.

[44] Cooper, G. and Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347. cited By 2859.

[45] Crutchfield, T. M. and Kistler, C. E. (2017). Getting patients in the door: medical appointment reminder preferences. *Patient preference and adherence*, pages 141–150.

[46] Czerniak, J. (2009). Acute inflammations data set. *UC Irvine Machine Learning Repository*.

[47] Czerniak, J. and Zarzycki, H. (2003). Application of rough sets in the presumptive diagnosis of urinary system diseases. In *Artificial intelligence and security in computing systems*, pages 41–51. Springer.

[48] David Gil, J. G. (2013). Fertility dataset. *UC Irvine Machine Learning Repository*.

[49] Davidson-Pilon, C. (2015). *Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference*. Addison-Wesley Data & Analytics Series. Pearson Education.

[50] Djan-Sampson, P. and Sahin, F. (2004). Structural learning of bayesian networks from complete data using the scatter search documents. In *Structural learning of bayesian networks from complete data using the scatter search documents*, volume 4, pages 3619–3624, The Hague. cited By 5; Conference of 2004 IEEE International Conference on Systems, Man and Cybernetics, SMC 2004 ; Conference Date: 10 October 2004 Through 13 October 2004; Conference Code:64440.

[51] Downey, A. (2021). *Think Bayes*. O'Reilly Media.

[52] Duerr, O., Sick, B., and Murina, E. (2020). *Probabilistic Deep Learning: With Python, Keras and TensorFlow Probability*. Manning Publications.

[53] Duman, E. and Ozcelik, M. H. (2011). Detecting credit card fraud by genetic algorithm and scatter search. *Expert Systems with Applications*, 38(10):13057–13063.

[54] Enshaei, A., Robson, C., and Edmondson, R. (2015). Artificial intelligence systems as prognostic and predictive tools in ovarian cancer. *Annals of surgical oncology*, 22(12):3970–3975.

[55] Eren, Y., İbrahim B. Küçükdemiral, and İlker Üstoğlu (2017). Chapter 2 - introduction to optimization. In Erdinç, O., editor, *Optimization in Renewable Energy Systems*, pages 27–74. Butterworth-Heinemann, Boston.

[56] Fahime Khozeimeh, Roohallah Alizadehsani, M. R. P. L. (2018). Cryotherapy dataset. *UC Irvine Machine Learning Repository*.

[57] Fan, Y.-J. and Chaovalitwongse, W. (2010). Optimizing feature selection to improve medical diagnosis. *Annals of Operations Research*, 174(1):169–183.

[58] Feng, L., Fengzhan, T., and Qiliang, Z. (2007). A novel ordering-based greedy bayesian network learning algorithm on limited data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4830 LNAI:80–89. cited By 3; Conference of 20th Australian Joint Conference on Artificial Intelligence, AI 2007 ; Conference Date: 2 December 2007 Through 6 December 2007; Conference Code:71206.

[59] Ferlay, J., Shin, H., Bray, F., Forman, D., Mathers, C., and Parkin, D. (2010). Globocan 2008, cancer incidence and mortality worldwide: IARC cancer base no. 10. *International Journal of Cancer*, 127(12):2893–2917.

[60] Fix, E. and Hodges, J. L. (1951). Nonparametric discrimination: Consistency properties. *Randolph Field, Texas, Project*, pages 21–49.

[61] Freedman, D. and Diaconis, P. (1981). On the histogram as a density estimator: L 2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476.

[62] Freitas, A. A. (2014). Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, 15(1):1–10.

[63] Friedman, N. (1997). Learning belief networks in the presence of missing values and hidden variables. *Proc. 14th Int. Conf. Machine Learning*, pages 125–133. cited By 291.

[64] Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163. cited By 3378.

[65] Friedman, N., Nachman, I., and Pe'er, D. (1999). Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 206–215. cited By 511.

[66] Friedman, N., Nachman, I., and Pe'er, D. (2013). Learning bayesian network structure from massive datasets: The" sparse candidate" algorithm. *arXiv preprint arXiv:1301.6696*.

[67] Gaba, F., Blyuss, O., Liu, X., Goyal, S., Lahoti, N., Chandrasekaran, D., Kurzer, M., Kalsi, J., Sanderson, S., Lanceley, A., et al. (2020). Population study of ovarian cancer risk prediction for targeted screening and prevention. *Cancers*, 12(5):1241.

[68] Gandhi, P., Bromberg, F., and Margaritis, D. (2008). Learning markov network structure using few independence tests. In *Learning Markov network structure using few independence tests*, volume 2, pages 680–691, Atlanta, GA. Society for Industrial and Applied Mathematics Publications. cited By 11; Conference of 8th SIAM International Conference on Data Mining 2008, Applied Mathematics 130 ; Conference Date: 24 April 2008 Through 26 April 2008; Conference Code:73777.

[69] García-Domínguez, A., Galván-Tejada, C., Brena, R., Aguileta, A., Galván-Tejada, J., Gamboa-Rosales, H., Celaya-Padilla, J., and Luna-García, H. (2021). Children's activity classification for domestic risk scenarios using environmental sound and a bayesian network. *Healthcare (Switzerland)*, 9(7). cited By 0.

[70] Gartner, D. (2015). *Optimizing Hospital-wide Patient Scheduling*, volume 674 of *Lecture Notes in Economics and Mathematical Systems*. Springer.

[71] George, A. and Vidyapeetham, A. (2012). Anomaly detection based on machine learning: dimensionality reduction using pca and classification using svm. *International Journal of Computer Applications*, 47(21):5–8.

[72] Gil, D., Girela, J., De Juan, J., Gomez-Torres, M., and Johnsson, M. (2012). Predicting seminal quality with artificial intelligence methods. *Expert Systems with Applications*, 39(16):12564–12573.

[73] Glover, F. (1977a). Heuristics for integer programming using surrogate constraints. *Decision sciences*, 8(1):156–166.

[74] Glover, F. (1977b). Heuristics for integer programming using surrogate constraints. *Decision sciences*, 8(1):156–166.

[75] Glover, F. (1998). A template for scatter search and path relinking. *Lecture notes in computer science*, 1363:13–54.

[76] Glover, F., Laguna, M., and Martí, R. (2003). Scatter search. In *Advances in evolutionary computing*, pages 519–537. Springer.

[77] Glowacka, K. J., Henry, R. M., and May, J. H. (2009). A hybrid data mining/simulation approach for modelling outpatient no-shows in clinic scheduling. *Journal of the Operational Research Society*, 60(8):1056–1068.

[78] Goldman, L., Freidin, R., Cook, E., Eigner, J., and Grich, P. (1982). A multivariate approach to the prediction of no-show behavior in a primary care center. *Archives of Internal Medicine*, 142(3):563–567.

[79] Grus, J. (2015). *Data Science from Scratch: First Principles with Python*. O'Reilly Media.

[80] Gurol-Urganci, I., de Jongh, T., Vodopivec-Jamsek, V., Atun, R., and Car, J. (2013). Mobile phone messaging reminders for attendance at healthcare appointments. *The Cochrane Library*.

[81] Hall, M. and Holmes, G. (2003). Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1437–1447.

[82] Han, S. H., Yun, M. H., Kim, K.-J., and Kwahk, J. (2000). Evaluation of product usability: development and validation of usability dimensions and design elements based on empirical models. *International Journal of Industrial Ergonomics*, 26(4):477–488.

[83] Harford, J. B. (2011). Breast-cancer early detection in low-income and middle-income countries: Do what you can versus one size fits all. *The Lancet Oncology*, 12(3):306–312.

[84] Haslwanter, T. (2016). *An Introduction to Statistics with Python: With Applications in the Life Sciences*. Statistics and Computing. Springer International Publishing.

[85] Heckerman, D., Geiger, D., and Chickering, D. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243. cited By 2226.

[86] Heineman, G., Pollice, G., and Selkow, S. (2008). *Algorithms in a Nutshell*. In a Nutshell (O'Reilly). O'Reilly Media.

[87] Herland, M., Khoshgoftaar, T. M., and Wald, R. (2014). A review of data mining using big data in health informatics. *Journal of Big data*, 1(1):1–35.

[88] Holmes, D. E. and Jain, L. C. (2008). Introduction to bayesian networks. In *Innovations in Bayesian networks*, pages 1–5. Springer.

[89] Hu, L.-Y., Huang, M.-W., Ke, S.-W., and Tsai, C.-F. (2016). The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus*, 5(1).

[90] Huang, Y. and Hanauer, D. (2014). Patient no-show predictive model development using multiple data sources for an effective overbooking approach. *Applied Clinical Informatics*, 5(3):836–860.

[91] Irvine, U. (2023). Uc irvine machine learning repository. *UC Irvine Machine Learning Repository*.

[92] Janžra, M. and Nielsen, J. (2006). A simulated annealing-based method for learning bayesian networks from statistical data. *International Journal of Intelligent Systems*, 21(3):335–348. cited By 15.

[93] Ji, Z., Xia, Q., and Meng, G. (2015). A review of parameter learning methods in bayesian network. In *International Conference on Intelligent Computing*, pages 3–12. Springer.

[94] Johansson, R. (2018). *Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib*. Apress.

[95] Kareem, S. and Okur, M. (2020). Structure learning of bayesian networks using elephant swarm water search algorithm. *International Journal of Swarm Intelligence Research*, 11(2):19–30. cited By 2.

[96] Khozeimeh, F., Alizadehsani, R., Roshanzamir, M., Khosravi, A., Layegh, P., and Nahavandi, S. (2017). An expert system for selecting wart treatment method. *Computers in Biology and Medicine*, 81:167–175.

[97] Kira, K. and Rendell, L. A. (1992). A practical approach to feature selection. In *Proceedings of the 9th International Conference on Machine Learning*, pages 249–256.

[98] Kitson, N. and Constantinou, A. (2021). Learning bayesian networks from demographic and health survey data. *Journal of Biomedical Informatics*, 113. cited By 1.

[99] Kochenderfer, M. J. and Wheeler, T. A. (2019). *Algorithms for optimization*. Mit Press.

[100] Kurt, W. (2019). *Bayesian Statistics the Fun Way: Understanding Statistics and Probability with Star Wars, LEGO, and Rubber Ducks*. No Starch Press.

[101] Kyrimi, E., Dube, K., Fenton, N., Fahmi, A., Neves, M. R., Marsh, W., and McLachlan, S. (2021a). Bayesian networks in healthcare: What is preventing their adoption? *Artificial Intelligence in Medicine*, page 102079.

[102] Kyrimi, E., McLachlan, S., Dube, K., Neves, M. R., Fahmi, A., and Fenton, N. (2021b). A comprehensive scoping review of bayesian networks in healthcare: Past, present and future. *Artificial Intelligence in Medicine*, page 102108.

[103] Laguna, M. and Martí, R. (2012). *Scatter Search: Methodology and Implementations in C*. Operations Research/Computer Science Interfaces Series. Springer US.

[104] Larrañaga, P. and Moral, S. (2011). Probabilistic graphical models in artificial intelligence. *Applied soft computing*, 11(2):1511–1528.

[105] Larrañaga, P., Sierra, B., Gallego, M., Michelena, M., and Picaza, J. (1997). Learning bayesian networks by genetic algorithms: a case study in the prediction of survival in malignant skin melanoma. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 261–272. Springer.

[106] Lee, V., Earnest, A., Chen, M., and Krishnan, B. (2005). Predictors of failed attendances in a multi-specialty outpatient centre using electronic databases. *BMC Health Services Research*, 5.

[107] Li, D.-C. and Liu, C.-W. (2010). A class possibility based kernel to increase classification accuracy for small data sets using support vector machines. *Expert Systems with Applications*, 37(4):3104–3110.

[108] Li, J., Shi, J., and Satz, D. (2008). Modeling and analysis of disease and risk factors through learning bayesian networks from observational data. *Quality and Reliability Engineering International*, 24(3):291–302. cited By 15.

[109] Li, K., Hüsing, A., Fortner, R., Tjønneland, A., Hansen, L., Dossus, L., Chang-Claude, J., Bergmann, M., Steffen, A., Bamia, C., et al. (2015). An epidemiologic risk prediction model for ovarian cancer in europe: the epic study. *British journal of cancer*, 112(7):1257–1265.

[110] Lin, X., Li, X., Xiao, N., Ma, P., Jiang, J., and Yang, F. (2012). A learning method of bayesian network structure. In *A learning method of Bayesian network structure*, pages 666–670, Chongqing. cited By 0; Conference of 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2012 ; Conference Date: 29 May 2012 Through 31 May 2012; Conference Code:95177.

[111] Little, M. (2008). Parkinsons dataset. *UC Irvine Machine Learning Repository*.

[112] Little, M., McSharry, P., Hunter, E., Spielman, J., and Ramig, L. (2008). Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *Nature Precedings*, pages 1–1.

[113] Little, M., McSharry, P., Roberts, S., Costello, D., and Moroz, I. (2007). Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *Biomedical Engineering Online*, 6(1):23.

[114] Liu, Z., Malone, B., and Yuan, C. (2012). Empirical evaluation of scoring functions for bayesian network model selection. In *BMC bioinformatics*, volume 13, pages 1–16. Springer.

[115] London, A. J. (2019). Artificial intelligence and black-box medical decisions: accuracy versus explainability. *Hastings Center Report*, 49(1):15–21.

[116] López, F. G., Torres, M. G., Batista, B. M., Pérez, J. A. M., and Moreno-Vega, J. M. (2006). Solving feature subset selection problem by a parallel scatter search. *European Journal of Operational Research*, 169(2):477–489.

[117] Mackay, D. (2003). *Information theory, inference and learning algorithms*. Cambridge University Press, Cambridge.

[118] Maiyar, L., Cho, S., Tiwari, M., Thoben, K.-D., and Kiritsis, D. (2019). Optimising online review inspired product attribute classification using the self-learning particle swarm-based bayesian learning approach. *International Journal of Production Research*, 57(10):3099–3120. cited By 9.

[119] Mandal, I. (2015). Developing new machine learning ensembles for quality spine diagnosis. *Knowledge-Based Systems*, 73(1):298–310. cited By 16.

[120] Mangasarian, O. L., Street, W. N., and Wolberg, W. H. (1995). Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43:570–577.

[121] Margaritis, D. (2003). *Learning Bayesian network model structure from data*. PhD thesis, School of Computer Science, Carnegie Mellon University.

[122] Margolis, K. L., Lurie, N., McGovern, P. G., and Slater, J. S. (1993). Predictors of failure to attend scheduled mammography appointments at a public teaching hospital. *Journal of General Internal Medicine*, 8(11):602–605.

[123] Martí, R., Laguna, M., and Glover, F. (2006). Principles of scatter search. *european Journal of operational Research*, 169(2):359–372.

[124] Matjaz Zwitter, M. S. (1988). Breast cancer data (restricted access). *UC Irvine Machine Learning Repository*.

[125] McLachlan, S., Dube, K., Hitman, G. A., Fenton, N. E., and Kyrimi, E. (2020). Bayesian networks in healthcare: Distribution by medical condition. *Artificial Intelligence in Medicine*, 107:101912.

[126] Menon, U., Kalsi, J., and Jacobs, I. (2012). The ukctocs experience—reasons for hope? *International Journal of Gynecologic Cancer*, 22(S1).

[127] Meyer, G., Adomavicius, G., Johnson, P., Elidrisi, M., Rush, W., Sperl-Hillen, J., and O'Connor, P. (2014). A machine learning approach to improving dynamic decision making. *Information Systems Research*, 24(2):239–263.

[128] Miettinen, K. and Juhola, M. (2010). Classification of otoneurological cases according to Bayesian probabilistic models. *Journal of Medical Systems*, 34(2):119–130.

[129] Moore, A. and Wong, W.-K. (2003). Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In Fawcett T., M. N., editor, *Optimal Reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning*, volume 2, pages 552–559, Washington, DC. cited By 71; Conference of Proceedings, Twentieth International Conference on Machine Learning ; Conference Date: 21 August 2003 Through 24 August 2003; Conference Code:62778.

[130] Myllymäki, P., Silander, T., Tirri, H., and Uronen, P. (2002). B-course: A web-based tool for bayesian and causal data analysis. *International Journal on Artificial Intelligence Tools*, 11(3):369–387. cited By 118.

[131] Nagarajan, R., Scutari, M., and Lèbre, S. (2013). *Bayesian Networks in R: with Applications in Systems Biology*. Springer New York. cited By 174.

[132] Nguyen, D. L., DeJesus, R. S., and Wieland, M. L. (2011). Missed appointments in resident continuity clinic: patient characteristics and health care outcomes. *Journal of graduate medical education*, 3(3):350–355.

[133] Niel, C., Sinoquet, C., Dina, C., and Rocheleau, G. (2015). A survey about methods dedicated to epistasis detection. *Frontiers in Genetics*, 6(SEP). cited By 72.

[134] Njah, H. and Jamoussi, S. (2015). Weighted ensemble learning of bayesian network for gene regulatory networks. *Neurocomputing*, 150(PB):404–416. cited By 22.

[135] numpy (2020a). numpy.digitize - numpy v1.19 manual.

[136] numpy (2020b). numpy.histogram_bin_edges - numpy v1.19 manual.

[137] Okonkwo, Q. L., Draisma, G., der Kinderen, A., Brown, M. L., and de Koning, H. J. (2008). Breast cancer screening policies in developing countries: A cost-effectiveness analysis for India. *Journal of the National Cancer Institute*, 100(18):1290–1300.

[138] Osheroff, J. A., Teich, J. M., Levick, D., Saldana, L., Velasco, F. T., Sittig, D. F., Rogers, K. M., and Jenders, R. A. (2012). *Improving outcomes with clinical decision support: an implementer's guide*. Himss Publishing.

[139] Padman, R., Bai, X., and Airoldi, E. (2007). A new machine learning classifier for high dimensional healthcare data. *Studies in Health Technology and Informatics*, 129:664–668. cited By 1; Conference of 12th World Congress on Medical Informatics, MEDINFO 2007 ; Conference Date: 20 August 2007 Through 24 August 2007.

[140] Parikh, A., Gupta, K., Wilson, A. C., Fields, K., Cosgrove, N. M., and Kostis, J. B. (2010). The effectiveness of outpatient appointment reminder systems in reducing no-show rates. *The American Journal of Medicine*, 123(6):542 – 548.

[141] Parliament, U. (2023). Select committee on artificial intelligence. *publications parliament*.

[142] Passi, K., Nour, A., and Jain, C. (2017). Markov blanket: Efficient strategy for feature subset selection method for high dimensional microarray cancer datasets. In I, Y., editor, *Markov blanket: Efficient strategy for feature subset selection method for high dimensional microarray cancer datasets*, volume 2017-January, pages 1864–1871.

Institute of Electrical and Electronics Engineers Inc. cited By 3; Conference of 2017 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2017 ; Conference Date: 13 November 2017 Through 16 November 2017; Conference Code:133962.

[143] Pearl, J. (2000). Causal inference without counterfactuals: Comment. *Causality: Models, Reasoning, and Inference*. cited By 6884.

[144] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[145] Pellet, J.-P. and Elisseeff, A. (2008). Using markov blankets for causal structure learning. *Journal of Machine Learning Research*, 9:1295–1342. cited By 127.

[146] Percac-Lima, S., Cronin, P. R., Ryan, D. P., Chabner, B. A., Daly, E. A., and Kimball, A. B. (2015). Patient navigation based on predictive modeling decreases no-show rates in cancer care. *Cancer*, 121(10):1662–1670.

[147] Rahman, B., Meisel, S. F., Fraser, L., Side, L., Gessler, S., Wardle, J., and Lanceley, A. (2015). Population-based genetic risk prediction and stratification for ovarian cancer: views from women at high risk. *Familial cancer*, 14(1):135–144.

[148] Ramani, R. and Jacob, S. (2013). Improved classification of lung cancer tumors based on structural and physicochemical properties of proteins using data mining models. *PLoS ONE*, 8(3). cited By 33.

[149] Riester, M., Wei, W., Waldron, L., Culhane, A. C., Trippa, L., Oliva, E., Kim, S.-h., Michor, F., Huttenhower, C., Parmigiani, G., et al. (2014). Risk prediction for late-stage ovarian cancer by meta-analysis of 1525 patient samples. *Journal of the National Cancer Institute*, 106(5):dju048.

[150] Robnik-Šikonja, M. and Kononenko, I. (2003). Theoretical and empirical analysis of ReliefF and RReliefF. 53(1):23–69.

[151] Roumani, Y., May, J., Strum, D., and Vargas, L. (2013). Classifying highly imbalanced ICU data. *Health Care Management Science*, 16(2):119–128.

[152] Russell, S. and Norvig, P. (2002). Artificial intelligence: a modern approach.

[153] Russell, S. and Norvig, P. (2016). *Artificial Intelligence: A Modern Approach*. Always learning. Pearson.

[154] Sahin, F., Tillett, J., Rao, R., and Rao, T. (2004). An evolutionary algorithmic approach to learning a bayesian network from complete data. volume 5433, pages 88–99, Orlando, FL. cited By 3; Conference of Data Mining and Knowledge Discovery: Theory, Tools, and Technology VI ; Conference Date: 12 April 2004 Through 13 April 2004; Conference Code:63396.

[155] Samorani, M. and LaGanga, L. R. (2015). Outpatient appointment scheduling given individual day-dependent no-show predictions. *European Journal of Operational Research*, 240(1):245–257.

[156] Scanagatta, M., Salmerón, A., and Stella, F. (2019). A survey on bayesian network structure learning from data. *Progress in Artificial Intelligence*, 8(4):425–439.

[157] scikit learn (2020). sklearn.preprocessing.labelencoder - sklearn manual.

[158] Scutari, M. (2009). Learning bayesian networks with the bnlearn r package. *arXiv preprint arXiv:0908.3817*.

[159] Scutari, M. (2010). Learning bayesian networks with the bnlearn r package. *Journal of Statistical Software*, 35(3):1–22. cited By 623.

[160] Scutari, M. (2011). Measures of variability for graphical models.

[161] Scutari, M. and Strimmer, K. (2010). Introduction to graphical modelling. In Stumpf, M., Balding, D., and Girolami, M., editors, *Handbook of statistical systems biology*, chapter 11, pages 235–254. Wiley, Chichester.

[162] Society, T. R. (2019). Explainable ai the basics policy briefing. *The Royal Society*. 978-1-78252-433-5.

[163] Song, H.-J., Ko, S.-K., Kim, J.-D., Park, C.-Y., and Kim, Y.-S. (2013). Looking for the optimal machine learning algorithm for the ovarian cancer screening. *International Journal of Bio-Science and Bio-Technology*, 5(2):41–48.

[164] Spirtes, P. and Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1):62–72.

[165] Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction and search*. Springer, New York, 2nd edition.

[166] Stephenson, T. A. (2000). An introduction to bayesian network theory and usage. Technical report.

[167] Stoica, P. and Selen, Y. (2004). Model-order selection: a review of information criterion rules. *IEEE Signal Processing Magazine*, 21(4):36–47.

[168] Tan, Y. and Liu, Z. (2013). Feature selection and prediction with a markov blanket structure learning algorithm. *BMC Bioinformatics*, 14:A3. cited By 2.

[169] Teyssier, M. and Koller, D. (2005). Ordering-based search: A simple and effective algorithm for learning bayesian networks. pages 584–590, Edinburgh. cited By 141; Conference of 21st Conference on Uncertainty in Artificial Intelligence, UAI 2005 ; Conference Date: 26 July 2005 Through 29 July 2005; Conference Code:86614.

[170] Tsamardinos, I., Aliferis, C., and Statnikov, A. (2003). Algorithms for large scale Markov blanket discovery. In *Proceedings of the 16th International Florida Artificial Intelligence Research Society Conference*, pages 376–381.

[171] Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78.

[172] Unpingco, J. (2019). *Python for Probability, Statistics, and Machine Learning*. Springer International Publishing.

[173] Vilone, G. and Longo, L. (2020). Explainable artificial intelligence: a systematic review. *arXiv preprint arXiv:2006.00093*.

[174] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

[175] Wang, D., Crilly, J., Jaeger, L. A., and Palmer, G. (2007). Assessing patient preferences for delivery of reminders on scheduled visits in a psychiatry ambulatory service. In *AMIA Annual Symposium Proceedings*, volume 2007, page 776. American Medical Informatics Association.

[176] Wang, Y. (2018). Analysis of the max-min hill-climbing algorithm. In *2018 International Conference on Transportation & Logistics, Information & Communication, Smart City (TLICSC 2018)*, pages 509–511. Atlantis Press.

[177] Wasserman, L. (2004). *All of statistics: A concise course in statistical inference*. Springer, New York.

[178] Wasserman, L. (2013). *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics. Springer New York.

[179] Webster, J. and Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, pages xiii–xxiii.

[180] Wikimedia (2021). Markov blanket.

[181] William Wolberg, Olvi Mangasarian, N. S. W. S. (1995). Breast cancer wisconsin (diagnostic). *UC Irvine Machine Learning Repository*.

[182] Witten, I. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition.

[183] Witten, I., Frank, E., Hall, M., and Pal, C. (2016). *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science.

[184] Wu, Y., McCall, J., and Corne, D. (2010). Two novel ant colony optimization approaches for bayesian network structure learning. Barcelona. cited By 35; Conference of 2010 6th IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010 ; Conference Date: 18 July 2010 Through 23 July 2010; Conference Code:85187.

[185] Yaramakala, S. and Dimitris, M. (2005). Speculative Markov blanket discovery for optimal feature selection. In *ICDM '05: Proceedings of the fifth IEEE international conference on data mining*, pages 809–812, Washington, DC, USA. IEEE Computer Society.

[186] Young, W., Weckman, G., and Holland, W. (2011). A survey of methodologies for the treatment of missing values within datasets: Limitations and benefits. *Theoretical Issues in Ergonomics Science*, 12(1):15–43.

[187] Zgurovskii, M., Bidyuk, P., and Terent'ev, A. (2008). Methods of constructing bayesian networks based on scoring functions. *Cybernetics and Systems Analysis*, 44(2):219–224.

[Zwitter et al.] Zwitter, M., Soklic, M., Tan, M., and Schlimme, J. Breast cancer data set. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer.

Appendix A

Jupyter Notebooks

Scatter Search

A.1 Python Libraries Import

```
%matplotlib inline
import sys
sys.path.append("../src/")
from results import Experiment_Results

import math
import random
import pickle
import operator
import matplotlib
import numpy as np
import pandas as pd
import networkx as nx
from datetime import datetime
import networkx.algorithms.isomorphism as iso
from scipy import stats
from functools import reduce
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score, roc_auc_score, recall_score, precision_score, confusion_matrix
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination
from pgmpy.estimators import MaximumLikelihoodEstimator, BayesianEstimator
from pgmpy.factors.discrete.CPD import TabularCPD
from sklearn.feature_selection import mutual_info_classif
from IPython.display import display, Math
import matplotlib.pyplot as plt
```

A.1.1 Supporting Class to Store Results

This file should be located in ../src/ in relation to the running file.

```
import pickle
import pandas as pd


class Experiment_Results:
    def __init__(self):
        self.field = ['dt', 'experiment_name', 'graph', 'runtime', 'samples', 'folds', 'accuracy', 'ROC', 'recall', 'precision', \
                      'TN', 'FN', 'TP', 'FP', 'NPV', 'PPV', 'sensitivity', 'specificity', 'PRC']
```

```
        self.field_mapping = {'dt': pd.Series([], dtype='datetime64[ns]'),
                              'experiment_name': pd.Series([], dtype='string'),
                              'model_name': pd.Series([], dtype='string'),
                              'graph': pd.Series([], dtype='int'),
                              'p_value': pd.Series([], dtype='float'),
                              'refset_size': pd.Series([], dtype='int'),
                              'before_limit_refset_size': pd.Series([], dtype='int'),
                              'vertices': pd.Series([], dtype='int'),
                              'edge': pd.Series([], dtype='int'),
                              'runtime': pd.Series([], dtype='float'),
                              'folds': pd.Series([], dtype='int'),
                              'samples': pd.Series([], dtype='int'),
                              'population': pd.Series([], dtype='int'),
                              'accuracy': pd.Series([], dtype='float'),
                              'ROC': pd.Series([], dtype='float'),
                              'recall': pd.Series([], dtype='float'),
                              'precision': pd.Series([], dtype='float'),
                              'TN': pd.Series([], dtype='float'),
                              'FN': pd.Series([], dtype='float'),
                              'TP': pd.Series([], dtype='float'),
                              'FP': pd.Series([], dtype='float'),
                              'NPV': pd.Series([], dtype='float'),
                              'PPV': pd.Series([], dtype='float'),
                              'sensitivity': pd.Series([], dtype='float'),
                              'specificity': pd.Series([], dtype='float'),
                              'PRC': pd.Series([], dtype='float')}
        self.df = None

    def create_df(self):
        self.df = pd.DataFrame(self.field_mapping)

        return self.df

    def save_df(self, filename):
        fd = open(filename,'wb')
        pickle.dump(self.df, fd)
        fd.close()

    def load_df(self, filename):
        fd = open(filename,'rb')
        self.df = pickle.load(fd)
        fd.close

    def add_row_as_dict(self, a_row_dict):
        self.df = self.df.append(a_row_dict, ignore_index=True)

    def get_df(self):
        return self.df

    def save_df_as_excel(self, filename):
        self.df.to_excel(filename)

    def load_df_from_excel(self, filename):
        self.df = pd.read_excel(filename, index_col=0)
```

A.1.2 Load Discretized Data

```
dataset1 = {'name': "acute-inflammations-diagnosis-discretized", \
            'filename': "./acute-inflammations-diagnosis-discretized.csv", \
            'folds': [2,5,10], \
            'p_value': [0.05, 0.09, 0.15, 0.20, 0.25, 0.30, 0.35], \
            'refset_size': [10, 40, 100]}

dataset2 = {'name': "acute-inflammations-diagnosis-discretized-oversampled", \
            'filename': "./acute-inflammations-diagnosis-discretized-oversampled.csv", \
            'folds': [2,5,10], \
```

```
            'p_value': [0.05, 0.09, 0.15, 0.20, 0.25, 0.30, 0.35], \
            'refset_size': [10, 40, 100]}
datasets = [dataset1, dataset2]
```

A.1.3 Data Testing Class

```
class Data_Testing:
    def __init__(self, df=None):
        self.df = df

    def get_dataframe(self):
        return self.df

    def get_kfold_train_test(self, splits=3):
        cv = KFold(n_splits=splits, shuffle=False)
        for train_index, test_index in cv.split(df):
            df_train = df.iloc[train_index,:]
            df_test = df.iloc[test_index,:]
            yield {'df_train': df_train, 'df_test': df_test}

    def get_cats(self, col):
        values = np.sort(self.df.loc[:,col].value_counts().index.to_numpy())
        return values

    def get_data_from_source(self, data, target, cols=[]):
        ds_target = data.loc[:,target]
        ds_cols = []

        for col in cols:
            col1 = data.loc[:,col]
            ds_cols.append(col1)
        ds_ret = {"ds_target": ds_target, "ds_cols": ds_cols} #Return target series and array of

        return ds_ret

    def get_data(self, target, cols=[]):
        ds_ret = self.get_data_from_source(self.df, target, cols)

        return ds_ret

    def g_test(self, samples, target, cols=[]):
        results = None
        data = self.get_data(target, cols)
```

```
        con = pd.crosstab(data['ds_target'],data['ds_cols'])
        results = stats.chi2_contingency(con, lambda_="log-likelihood")
        return results[1]

    def chi_test(self, samples, target, cols=[]):
        results = None
        data = self.get_data(target, cols)
        con = pd.crosstab(data['ds_target'],data['ds_cols'])
        results = stats.chi2_contingency(con)
        return results[1]
```

A.1.4 Scatter Search Class

```
class Scatter_Search:

    class Metrics:
        def __init__(self, solutions):
            self.solutions = solutions

        def sort_solutions(self, metric='accuracy'):
            self.solutions = sorted(self.solutions, key=lambda solution: solution.get_metrics(metric=metric), reverse=True)

        def get_top_3_graphs(self):
            self.sort_solutions()
            return self.solutions[0:3]

        def get_top_graph_all_metrics(self, metric='accuracy'):
            self.sort_solutions(metric)
            return self.solutions.get_metrics()

        def get_number_of_nodes_edges(self, total_graphs=3, metric='accuracy'):
            graph_VE_metrics = []
            self.sort_solutions(metric)

            for i in range(0,total_graphs):
                v = self.solutions[i].get_number_of_nodes()
                e = self.solutions[i].get_number_of_edges()
                ve_metric = {'V': v, 'E': e}
                graph_VE_metrics.append(ve_metric)

            return graph_VE_metrics

        def get_top_graphs_time(self, total_graphs=3, metric='accuracy'):
            graph_times = {}
            self.sort_solutions(metric)
            for i in range(0,total_graphs):
                graph_times['graph-'+str(i+1)] = self.solutions[i].get_bayesian_model().get_total_time().microseconds/1000

            return graph_times

        def get_confusion_matrix(self, solution):
            cm_metrics = {}
            cm = solution.bn.get_metrics()['cm']
            cm_metrics['TN'] = cm[0][0]
            cm_metrics['FN'] = cm[1][0]
            cm_metrics['TP'] = cm[1][1]
            cm_metrics['FP'] = cm[0][1]
            cm_metrics['population'] = cm_metrics['TN']+cm_metrics['FN']+cm_metrics['TP']+cm_metrics['FP']
            cm_metrics['NPV'] = cm_metrics['TN']/(cm_metrics['TN']+cm_metrics['FN'])
            cm_metrics['PPV'] = cm_metrics['TP']/(cm_metrics['TP']+cm_metrics['FP'])
            cm_metrics['sensitivity'] = cm_metrics['TP']/(cm_metrics['TP']+cm_metrics['FN'])
            cm_metrics['specificity'] = cm_metrics['TN']/(cm_metrics['TN']+cm_metrics['FP'])

            return cm_metrics

        def get_top_3_metrix_metrics(self):
```

```
        top_3_metrics = []
        top_solutions = self.get_top_3_graphs()
        for solution in top_solutions:
            top_3_metrics.append(self.get_confusion_matrix(solution))

        return top_3_metrics

class Reporting:
    def __init__(self, metrics_handler):
        self.mh = metrics_handler

    def print_report_1(self):
        print("-----------Top three graphs-----------")
        for solution in self.mh.get_top_3_graphs():
            print(solution.get_metrics())
        print("-----------Top three graphs-----------")
        for metric in self.mh.get_number_of_nodes_edges(total_graphs=3,metric='accuracy'):
            print(metric)
        print("-----------Top three graphs-----------")
        print("---------------Accuracy(ms)-----------")
        print(self.mh.get_top_graphs_time(total_graphs=3, metric='accuracy'))
        print("-----------Top three graphs-----------")
        print("-----------------ROC(ms)--------------")
        print(self.mh.get_top_graphs_time(total_graphs=3, metric='roc'))
        print("-----------Top three graphs-----------")
        print("---------------Recall(ms)-------------")
        print(self.mh.get_top_graphs_time(total_graphs=3, metric='recall'))
        print("-----------Top three graphs-----------")
        print("---------------Precision(ms)----------")
        print(self.mh.get_top_graphs_time(total_graphs=3, metric='precision'))
        print("-----------Top three graphs-----------")
        print("------------CM metrics(ms)------------")
        for cm_metrics in self.mh.get_top_3_metrix_metrics():
            print(cm_metrics)


class Solution:
    def __init__(self, graph, p_value):
        self.graph = graph
        self.p_value = p_value
        self.bn = Scatter_Search.Bayesian_Model(graph)

    def get_number_of_nodes(self):
        return self.bn.get_bayesian_model().number_of_nodes()

    def get_number_of_edges(self):
        return self.bn.get_bayesian_model().number_of_edges()

    def get_bayesian_model(self):
        return self.bn

    def get_graph(self):
        return self.graph

    def get_score(self):
        return self.p_value

    def get_training_time(self):
        return self.bn.total_training_time

    def get_metrics(self, metric=None):
        if metric is None:
            return self.bn.get_metrics()
        else:
            return self.bn.get_metrics()[metric]

    def update_graph_from_bayesian_model(self):
        self.graph = self.bn.create_networkx_graph()

class Solution_Store:
    def __init__(self):
        self.solutions = []

    def get_solution_list_size(self):
        return len(self.solutions)
```

```
    def sort_by_score(self):
        self.solutions.sort(key=lambda sol: sol.get_score())

    def get_best_scoring(self):
        self.sort_by_score()
        return self.solutions[0].get_score()

    def get_worst_scoring(self):
        self.sort_by_score()
        return self.solutions[len(self.solutions)-1]

    def check_solution_is_in_population(self, new_solution):
        is_in_solutions = False
        if len(self.solutions) > 0:
            for solution in self.solutions:
                nm = iso.categorical_node_match('name', 'name')
                if nx.is_isomorphic(solution.get_graph(), new_solution.get_graph(), node_match=nm) == True:
                    is_in_solutions = True
            return is_in_solutions
        else:
            return is_in_solutions

    def check_solution_is_directed_acyclic_graph(self, new_solution):
        if (nx.is_directed_acyclic_graph(new_solution.get_graph()) == True):
            return True
        else:
            return False

    def add_solution_with_checks(self, new_solution):
        if self.check_solution_is_in_population(new_solution) == False:
            if self.check_solution_is_directed_acyclic_graph(new_solution) == True:
                self.solutions.append(new_solution)
                return True
            else:
                pass
        else:
            pass
        return False

    def add_solution(self, new_solution):
        self.solutions.append(new_solution)

    def get_solution_set(self):
        return self.solutions

    def get_metrics_handler(self):
        return Scatter_Search.Metrics(self.solutions)

    def save_solutions(self, filename='solutions_store.pkl'):
        fd = open(filename, "wb")
        pickle.dump(self.solutions,fd)
        fd.close()

    def score_all_solutions_over_all_data(self):
        for solution in self.solutions:
            solution.get_bayesian_model().score_network_over_all_data()

    def limit_solutions(self, solution_limit=10):
        if len(self.solutions) > solution_limit:
            random.shuffle(self.solutions)
            self.solutions = self.solutions[0:solution_limit]
            return self.solutions

class Population(Solution_Store):
    def __init__(self):
        super().__init__()

    def get_population_set(self):
        return self.get_solution_set()

class RefSet(Solution_Store):
    def __init__(self):
        super().__init__()
```

```
    def add_new_solutions(self, solutions, number_of_solutions):
        i=0
        for solution in solutions:
            i=i+1
            if i <= number_of_solutions:
                self.add_solution_with_checks(solution)
            else:
                break

    def clean_refset(self, target):
        i=0
        new_solutions = []
        for solution in self.solutions:
            has_target = solution.get_bayesian_model().get_bayesian_model().has_node(target)
            number_of_nodes = len(solution.get_bayesian_model().get_bayesian_model().in_edges(target))
            if ((number_of_nodes > 0) or (has_target is True)):
                new_solutions.append(solution)
            else:
                pass
            i=i+1
        self.solutions = new_solutions

    def run_fit_parameters_for_all_models(self, folds=2):
        for solution in self.solutions:
            solution.get_bayesian_model().fit_parameters(k_fold_splits=folds)

    def get_random_solution(self, contains_node=None):
        i = 0
        index_max = (self.get_solution_list_size()-1)
        index = random.randint(0,index_max)
        if contains_node != None:
            while (self.solutions[index].get_graph().has_node(contains_node) == False) and (i < 10):
                index = random.randint(0,index_max)
                i=i+1
        if i > 9:
            return None
        else:
            return self.solutions[index]

    def find_best_solution(self, metric="accuracy"):
        best_fit_solution = self.solutions[0]

        for solution in self.solutions:
            metric1 = best_fit_solution.get_bayesian_model().get_metrics()[metric]
            metric2 = solution.get_bayesian_model().get_metrics()[metric]
            if metric2 > metric1:
                best_fit_solution = solution

        return best_fit_solution

class Bayesian_Model:
    def __init__(self, graph, target='Y1'):
        self.bn = BayesianModel()
        self.graph = graph
        self.create_bayesian_network()
        self.metrics = {"accuracy": 0, "roc": 0, "recall": 0, "precision": 0}
        self.target = target
        self.start_training_time = 0
        self.total_training_time = 0

    def find_best_graph(self, models, metric):
        current_model = {"bn_model": None, "score": {"accuracy": 0, "roc": 0, "recall": 0, "precision": 0}}

        for model in models:
            if model['score'][metric] > current_model['score'][metric]:
                current_model = model

        return current_model

    def get_state_names(self, cols=[]):
        state_names_dict = {}
        data = data_testing.get_dataframe()

        for col in cols:
            values = list(data[[col]].drop_duplicates().to_numpy().flatten())
```

```
            state_names_dict[col] = values

        return state_names_dict

    def fit_parameters(self, k_fold_splits=2, estimator='mle'):
        bn_models = []
        bn_scores = []
        state_names_dict = {}
        self.set_start_training_time()
        self.trim_to_markov_blanket()
        for data in data_testing.get_kfold_train_test(splits=k_fold_splits):#This has been initalized as gobal in the scatter search init
            bn_model = self.bn.copy()
            if estimator == 'mle':
                bn_model.fit(data['df_test'][list(bn_model.nodes)], state_names=self.get_state_names(bn_model.nodes))
            elif estimator == 'bayesian':
                #estimator = BayesianEstimator(bn_model, data)
                bn_model.fit(data['df_test'][list(bn_model.nodes)], estimator=BayesianEstimator,
                \ state_names=self.get_state_names(bn_model.nodes), prior_type='K2')
            score = self.score_network(bn_model, data['df_test'][list(bn_model.nodes)])
            bn_scores.append(score)
            bn_models.append({"bn_model": bn_model, "score": score})
        final_result = self.find_best_graph(bn_models, "accuracy")
        self.bn = final_result['bn_model']
        self.metrics = final_result['score']
        self.set_stop_training_time()

    def score_network(self, bn, test_data):
        if self.target in list(bn.nodes):
            td = test_data.drop(self.target, axis=1)
            predictions = bn.predict(td)
            accuracy = accuracy_score(test_data[self.target], predictions)
            try:
                roc = roc_auc_score(test_data[self.target], predictions)
            except ValueError:
                roc = 0
            recall = recall_score(test_data[self.target], predictions)
            precision = precision_score(test_data[self.target], predictions)
            cmatrix = confusion_matrix(test_data[self.target], predictions)
            scores = {"accuracy": accuracy, "roc": roc, "recall": recall, \
                      "precision": precision, "cm": cmatrix}
        else:
            scores = {"accuracy": 0, "roc": 0, "recall": 0, "precision": 0, "cm": None}

        return scores

    def score_network_over_all_data(self):
        nodes = list(self.bn.nodes)
        df = data_testing.get_dataframe()[nodes]
        prediction_df = df.drop(self.target, axis=1)
        predictions = self.bn.predict(prediction_df)
        target_data = df[self.target]

        accuracy = accuracy_score(target_data, predictions)
        try:
            roc = roc_auc_score(target_data, predictions)
        except ValueError:
            roc = 0
        recall = recall_score(target_data, predictions)
        precision = precision_score(target_data, predictions)
        cmatrix = confusion_matrix(target_data, predictions)
        self.metrics = {"accuracy": accuracy, "roc": roc, "recall": recall, \
                        "precision": precision, "cm": cmatrix}

    def trim_to_markov_blanket(self):
        G = BayesianModel()
        nodes = self.bn.get_markov_blanket(self.target)
        nodes.append(self.target)
        for edge in self.bn.in_edges:
            if (edge[0] in nodes) or (edge[1] in nodes):
                G.add_edge(edge[0], edge[1])

        for edge in self.bn.out_edges:
            if (edge[0] in nodes) or (edge[1] in nodes):
                G.add_edge(edge[0], edge[1])
```

```
            self.bn = G

        def create_bayesian_network(self):
            for edge in self.graph.edges:
                self.bn.add_edge(edge[0], edge[1])
            #self.fit_parameters()

        def create_networkx_graph(self):
            G = nx.DiGraph()
            for edge in self.bn.edges():
                G.add_edge(edge[0], edge[1])

            return G

        def get_target_edge_nodes(self, target):
            out_nodes = set()
            edges = self.bn.edges

            for edge in edges:
                if edge[0] == target:
                    out_nodes.add(edge[1])

            return out_nodes

        def has_target(self, target):
            edges = self.bn.edges(target)

            if len(edges) > 0:
                return True
            else:
                return False

        def get_bayesian_model(self):
            return self.bn

        def get_metrics(self):
            return self.metrics

        def set_start_training_time(self):
            self.start_training_time = datetime.now()

        def set_stop_training_time(self):
            self.total_training_time = datetime.now() - self.start_training_time

        def get_total_time(self):
            return self.total_training_time

    ######################################################

    def __init__(self, df, target='Y1', x_cols=[], y_cols=[]):
        global data_testing
        data_testing = Data_Testing(df)
        self.data_testing = data_testing
        self.df = df
        self.target = target
        self.population = Scatter_Search.Population()
        self.refset = Scatter_Search.RefSet()

    def build_graph_from_cols(self, target, cols):
        G = nx.DiGraph()
        G.add_node(target, name=target)
        for col in cols:
            G.add_edge(col, target)

        return G

    def diversification_from_target(self, alpha):
        new_pop_solutions = []
        cols = (self.df.iloc[0:1].drop(self.target, axis=1)).columns
        for col in cols:
            p_value = self.data_testing.chi_test(len(df.index), self.target, [col])
            if p_value < alpha:
                G = nx.DiGraph()
                G.add_node(col, name=col)
                G.add_node(self.target, name=self.target)
```

```
            G.add_edge(col, self.target)
            new_pop_solutions.append(Scatter_Search.Solution(G, p_value))
    for new_solution in new_pop_solutions:
        self.population.add_solution_with_checks(new_solution)

def diversification_from_comparing(self, alpha):
    new_pop_solutions = []
    cols = (self.df.iloc[0:1].drop(self.target, axis=1)).columns
    for col1 in cols:
        for col2 in cols:
            if col1 != col2:
                p_value1 = self.data_testing.chi_test(len(df.index), col1, [col2])
                if p_value1 < alpha:
                    G1 = self.build_graph_from_cols(col1, [col2])
                    new_pop_solutions.append(Scatter_Search.Solution(G1, p_value1))
    for new_solution in new_pop_solutions:
        self.population.add_solution_with_checks(new_solution)


def diversification(self, alpha=0.05):
    self.diversification_from_target(alpha)
    self.diversification_from_comparing(alpha)

def best_scoring_solutions(self):
    target_solutions = Scatter_Search.Solution_Store()

    for solution in ss.population.get_population_set():
        G = solution.get_graph()
        add_graph = True
        for out_data in G.edges.data():
            if out_data[1] in [self.target]:
                target_solutions.add_solution_with_checks(solution)

    target_solutions.sort_by_score()

    return target_solutions

def diverse_solutions(self): #Find solutions not directly connected to Y
    population_solutions = ss.population.get_population_set()
    nodes = set()
    diverse_solutions = Scatter_Search.Solution_Store()

    for sol in population_solutions:
        for edge_tuple in sol.get_graph().edges:
            if self.target == edge_tuple[1]:
                nodes.add(edge_tuple[0])
    for sol in population_solutions:
        if sol.get_graph().has_node(self.target) == False:
            for out_data in sol.get_graph().edges:
                if out_data[0] in list(nodes):
                    diverse_solutions.add_solution_with_checks(sol)

    diverse_solutions.sort_by_score()

    return diverse_solutions

def populate_refset(self, limit_number_of_solutions=10):
    number_of_solutions_from_each = round(limit_number_of_solutions/2)
    self.refset.add_new_solutions(self.best_scoring_solutions().get_solution_set(), number_of_solutions_from_each)
    self.refset.add_new_solutions(self.diverse_solutions().get_solution_set(), number_of_solutions_from_each)

def combine_solutions(self, solution1, solution2):
    new_solution = None
    G = nx.compose(solution1.get_graph(),solution2.get_graph())
    if nx.is_directed_acyclic_graph(G) == True:
        new_solution = Scatter_Search.Solution(G, 0)
    else:
        print("Error graph is not acyclic....")
        pass

    return new_solution

def subset_generation_method(self):
    solution1 = self.refset.get_random_solution()
    solution2 = solution1
```

```
        while(solution1 == solution2 or solution2 is None):
            solution2 = self.refset.get_random_solution(contains_node=self.target)
        if (solution1 != None and solution2 != None):
            return self.combine_solutions(solution1, solution2)
        else:
            print("New solution not found, none type returned")
            return None

    def run_scatter_search(self, fold, p_value, refset_size, experiment_name):
        last_refset_update=0
        number_of_iterations=0
        startTime = datetime.now()
        ss.diversification(alpha=p_value)
        self.populate_refset()
        print("Refset size: " + str(self.refset.get_solution_list_size()))

        while(last_refset_update < 20 and number_of_iterations < 500):
            soultion = self.subset_generation_method()
            if soultion == None:
                print("No solutions found...")
                break
            if self.refset.add_solution_with_checks(soultion) == True:
                last_refset_update = 0
                #nx.draw_networkx(soultion.get_graph())
                plt.show()
            else:
                last_refset_update = last_refset_update+1
            number_of_iterations = number_of_iterations+1

        print("Number of iterations: " + str(number_of_iterations))
        print("Last refset update count: " + str(last_refset_update))
        print("Refset size: " + str(self.refset.get_solution_list_size()))
        self.refset.clean_refset('Y1')
        print("Refset size: " + str(self.refset.get_solution_list_size()))
        before_refset_limit = self.refset.get_solution_list_size()
        self.refset.limit_solutions(solution_limit=refset_size)
        print("Refset size: " + str(self.refset.get_solution_list_size()))
        self.refset.run_fit_parameters_for_all_models(fold)
        self.refset.score_all_solutions_over_all_data()
        solution = self.refset.find_best_solution()
        print(solution.get_bayesian_model().get_metrics())
        print("Number of nodes: " + str(solution.get_number_of_nodes()))
        self.refset.save_solutions()
        mh = self.refset.get_metrics_handler()
        solutions = mh.get_top_3_graphs()
        for solution in solutions:
            print("Accuracy: " + str(solution.get_metrics(metric="accuracy")))
        print("--------\n\n\n\n--------")
        print("Total runtime: " + str(datetime.now() - startTime))
        solutions = mh.get_top_3_graphs()
        for solution in solutions:
            bnm = solution.get_bayesian_model()
            print(bnm.get_metrics())
            print("Nodes: " + str(len(bnm.get_bayesian_model().nodes)) + " Edges: " + str(len(bnm.get_bayesian_model().edges)))
            nx.draw_networkx(bnm.get_bayesian_model())
            plt.show()
        report = self.Reporting(mh)
        report.print_report_1()

        erc = Experiment_Results()
        erc.load_df('../results/pd_results_dataframe.pkl')
        i=0
        for graph in mh.get_top_3_graphs():
            i=i+1
            results_row = {'dt': datetime.now(),
                           'experiment_name': experiment_name,
                           'model_name': 'scatter search',
                           'graph': i,
                           'p_value': p_value,
                           'refset_size': refset_size,
                           'before_limit_refset_size': before_refset_limit,
                           'vertices': graph.get_number_of_nodes(),
                           'edge': graph.get_number_of_edges(),
                           'runtime': graph.get_bayesian_model().get_total_time().microseconds/1000,
                           'folds': fold,
```

```
                            'samples': df.iloc[:,0].count(),
                            'population': mh.get_confusion_matrix(graph)['population'],
                            'accuracy': graph.get_bayesian_model().get_metrics()['accuracy'],
                            'ROC': graph.get_bayesian_model().get_metrics()['roc'],
                            'recall': graph.get_bayesian_model().get_metrics()['recall'],
                            'precision': graph.get_bayesian_model().get_metrics()['precision'],
                            'TN': mh.get_confusion_matrix(graph)['TN'],
                            'FN': mh.get_confusion_matrix(graph)['FN'],
                            'TP': mh.get_confusion_matrix(graph)['TP'],
                            'FP': mh.get_confusion_matrix(graph)['FP'],
                            'NPV': mh.get_confusion_matrix(graph)['NPV'],
                            'PPV': mh.get_confusion_matrix(graph)['PPV'],
                            'sensitivity': mh.get_confusion_matrix(graph)['sensitivity'],
                            'specificity': mh.get_confusion_matrix(graph)['specificity'],
                            'PRC': 0}
            erc.add_row_as_dict(results_row)

        erc.save_df('../results/pd_results_dataframe.pkl')
        return erc

for dataset in datasets:
    df = pd.read_csv(dataset['filename'])
    df = df.sample(frac=1)
    for fold in dataset['folds']:
        for p_value in dataset['p_value']:
            for refset_size in dataset['refset_size']:
                ss = Scatter_Search(df)
                erc = ss.run_scatter_search(fold, p_value, refset_size, dataset['name'])
```

A.2 Data Discretization

A.2.1 Python Libraries

```
%matplotlib inline

import matplotlib
import numpy as np
import pandas as pd
from imblearn.over_sampling import SMOTE
```

A.2.2 Load Data

```
df = pd.read_csv("acute-inflammations-diagnosis.csv",header=None)
for label, content in df.iteritems():
    if content.dtype == "object":
        df[label] = content.map(dict(yes=1, no=0))
df1 = df.apply(pd.to_numeric, errors='coerce')
continuous_index =  df1.dtypes[df1.dtypes == "float64"].index.values.tolist()
```

A.2.3 Bin Data

```
df2 = pd.DataFrame()
for i in df1.dtypes.index.values.tolist():
    if(i in continuous_index):
        npa = df1.iloc[:,i].to_numpy()
        npa1 = npa[~np.isnan(npa)]
        bins = np.histogram_bin_edges(npa1)
        npa2 = np.digitize(npa, bins)
        df2.insert(loc=i,column=i,value=npa2)
    else:
        series = df1.iloc[:,i]
        df2.insert(loc=i,column=i,value=series)
```

A.2.4 Output Dataset

```
header_names=['X1','X2','X3','X4','X5','X6','X7','Y1']

oversample = SMOTE()
cols = df2.columns
X = df2.iloc[:,0:len(cols)-1]
Y = df2.iloc[:,len(cols)-1]
print("Number of rows before SMOTE: " + str(len(Y)))
X, Y = oversample.fit_resample(X, Y)
print("Number of rows after SMOTE: " + str(len(Y)))
df3 = pd.DataFrame(X)
df3['Y1'] = pd.Series(Y)

df2.to_csv("acute-inflammations-diagnosis-discretized.csv", index=False, header=header_names)
df3.to_csv("acute-inflammations-diagnosis-discretized-oversampled.csv", \
index=False, header=header_names)
```

Appendix B

Python libraries and Frameworks

B.1 Libraries

B.1.1 Math

Math provides mathematical functions in python. Key methods used are absolute value, average, product, logs, factorial, etc.

URL: https://docs.python.org/3/library/math.html

B.1.2 Random

Random provides functionality to generate pseudo-random numbers that can be seeded for reproducibility. Random offers sampling, shuffling and various probability sampling distributions.

URL: https://docs.python.org/3/library/random.html

B.1.3 Pickle

Pickle allows objects within python code to be serialized for saving to flat files for reloading later. This library is used to save python objects containing results and models for use at a later date.

URL: https://docs.python.org/3/library/pickle.html

B.1.4 Matplotlib

Matplotlib is used for the plotting of charts and graphs. It is useful for the visualisation of data or graphs. The main use of this library here is to display the graph structure of networks.

URL: https://matplotlib.org/

B.1.5 Numpy

Numpy offers mathematical functions and data structures such as matrices, lists and vectors.
URL: https://numpy.org/

B.1.6 Pandas

Pandas is used to load structured data in python and handle data operations such as reshaping, contingency tables, iteration over rows and saving of structured table data to formats such as CSV.
URL: https://pandas.pydata.org/

B.1.7 NetworkX

NetworkX is a python library for working with structured data of graphs. It has many of the functions needed to manipulate, store, alter and merge graph structures.
URL: https://pandas.pydata.org/

B.1.8 Datetime

This library is used for handling start and end times so the runtime of code can be calculated. The library offers much more and can handle date-time operations and formatting of date and time stamps.
URL: https://docs.python.org/3/library/datetime.html

B.1.9 Scipy

Scipy offers many scientific computing functions and algorithms. The use in context here is for the stats functionality for statistical tests.
URL: https://scipy.org/

B.1.10 Scikit-learn (Sklearn)

Scikit learn is a machine learning library for python that offers many out of the box ready to go models. The library also has many machine learning support classes for tasks such as parameter searching, grid searching, metrics and reporting.

URL: https://scikit-learn.org/

B.1.11 IPython

Ipython offers an interactive shell and functions to control the visual output of the jupyter notebooks.

URL: https://ipython.org/

B.1.12 PyMC3

PyMC3 is a probabilistic bayesian framework for building machine learning models using bayesian methods.

URL: https://docs.pymc.io/en/v3/

B.1.13 PgmPy

PgmPy is a framework for the modelling of bayesian networks in python. The framework offers both exact and approximate inference methods. The main type of distribution used in this framework is conditional probability distributions (tables)

URL: https://pgmpy.org/