Honours Project Report

Weather Forecasting with Bayesian Networks

Causal Modelling

Michael Kent

Supervised By:

Dr. Hanh Le

Dr. Mark Tadross

Dr. Anet Potgieter

Department of Computer Science

University of Cape Town

2008

Abstract

The weather model was a system for forecasting weather. It was a combination of research fields from the Department of Computer Science and the Department of Environmental and Geographical Science at the University of Cape Town. The objective of the weather model was to investigate the methods and effects of forecasting weather for stations in Southern Africa. Furthermore it consisted of three sections namely, Causal Modelling, Dynamic Bayesian Network Learning, and a Visualization System. This report focuses on causal modelling. The Naive Bayes, K2, LK2, and Greedy Thick Thinning algorithms were implemented and evaluated. The results show that the Naive Bayes algorithm constructs networks in the shortest time but with the lowest predictive accuracy. The networks produced by the LK2 algorithm forecasted with the highest predictive accuracy when using precipitation data, while the Greedy Thick Thinning algorithm produced the highest predictive accuracy networks for minimum and maximum temperature data.

Keywords: Graph algorithms, Computations on discrete structures.

Acknowledgements

I would like to thank my supervisor, Hanh Le, for her management and support throughout the duration of the project. I would also like to thank my co-supervisor, Mark Tadross, for offering freely of his time and expertise. Furthermore, I would like to thank Anet Potgieter for providing the background knowledge on Bayesian Networks, and Chris Jack and Chris Lennard of the Environmental and Geographical Department for helping us acquire the station data. Lastly I would like to thank my team members, Pascal Haingura and Matthew de Kock, for taking the risk involved in working on an honours project that was project proposed by a student.

Table of Contents

List of Figures

1 Introduction

Weather forecasting is a complex and difficult task. Firstly there are the atmospheric processes to consider, including large scale weather systems and the effect of terrain and sea surface temperatures on the weather. Secondly there are the computational requirements of the algorithms or models responsible for producing the forecasts. So the weather has to be accurately forecasted in a computationally feasible way.

Fortunately, Bayesian Networks provide us with such a way for which to forecast weather with. Traditionally the structure of the Bayesian Network is based on either the weather variables explicitly, or on the weather stations which record them. In this report we investigate the different techniques involved with finding the latter, namely the structure based on weather stations distributed throughout Southern Africa.

Although Bayesian Networks provide a means for forecasting weather, finding the structure of the networks remains an NP-hard problem. The reason for this is that there is an enormously large number of ways in which the stations can be linked to each other. In order to mitigate this problem, a number of algorithms have been proposed. These include the Naive Bayes Classifier, K2, LK2 and Greedy Thick Thinning algorithms.

In this report we wish to determine the algorithm which produces the network with highest predictive accuracy, and is constructed in the least amount of time. These are two evaluation metrics that we will use to compare algorithms with when forecasting weather. Specifically in this report we wish to investigate the network construction times of the K2 and LK2 algorithms. In addition to this, we also want to evaluate the effect that increasing the number of number of neighbours has on the predictive accuracy of the LK2 algorithm. Lastly we wish to determine which algorithms produce the networks with the highest predictive accuracy for each weather variable, and determine the effect that varying the bin count has on the predictive accuracy and construction times of the networks.

This report covers the entire development process of the network construction algorithms. Section 2 provides a background into Bayesian Network theory while Section 3 covers the design method used in the development period. Section 4 and 5 detail the pre-processing of the data, and the implementation of the algorithms respectively. Lastly section 6 reports on the findings, while Section 7 concludes the report.

2 Bayesian Networks and Causal Modelling Background

Bayesian Networks provide a mechanism for reasoning about uncertainty. Each Bayesian Network (BN) is comprised of a set of nodes, states, and arcs that together form a directed acyclic graph (DAG). Each node represents a random variable within the problem domain. Within the context of this report, the nodes represent weather stations from around Southern Africa.

The states of a node are the set of values that it can record. For example a station might forecast that tomorrow, its recorded temperature will range between three states, *cold*, *warm* and *hot*. When tomorrow comes and we know for certain that state of the station is *hot*, then we say that the station has been *observed* [17].

The arcs of the network represent the presence of a relationship between any two nodes. This is analogous to how observing the weather at one station allows us to better forecast the weather at another station.

Once a relationship has been discovered, the degree of the relationship is maintained in a conditional probability table (CPT). Each node has a CPT associated with it, and the CPT represents the degree to which the sates of a child node are affected by all the states of its parents. This effect is calculated through a process known as *inference.* Unfortunately, this makes the size of the CPT grow exponentially with an increase in the number of possible parents [17]. For instance a node with five states and two parents requires a CPT of 125 values, and that same node with 3 parents will require a CPT of 625 values.

This exponential growth in the size of the CPT therefore limits the number of parents that a node can have[a]. In response to this other storage mechanisms such as decision trees and decision graphs have been proposed to reduce the space requirement of the CPT [7].

The DAG or causal model (CM) structure is the foundation of the BN. There are three different approaches that can be taken in order to find the structure. The first is through the extraction of information from a domain expert. The second is through finding or mining the variable relationships directly from the observed data. The third method is a hybrid form, which attempts to combine the benefits of the first two approaches.

In the case of the domain expert, the causal model is constructed based on the information supplied by an expert in the problem field. This technique was used in the construction of the Hailfinder severe storm model [1]. The expert is examined and asked a series of questions which are used to create dependencies between the different variables in the problem domain. The benefit to this approach is that the CM is based on the experience of an expert, without it having to be defined from the data. In the case of the Hailfinder model, the relevant data was not available in sufficient quantity to be mined, and so there was no alternative approach to using a domain expert. However the domain expert had to fill in a total of 3975 probabilities for the CM. The domain expert found the task of calculating the probabilities to be time consuming.

The second technique used to construct CMs is by extracting them directly from the observed data. Within this approach there are two sub-approaches, the constraint and metric based approaches. Although the constraint based techniques are sub-optimal in representing the variable relationships, they simple to implement and understand [17].

2.1 Naive Bayes Classifier

The constraint based approach attempts to isolate the dependency between any two variables, while not considering the effect of other variables on their relationship. An example of the constraint based sub-approach is the Naive Bayes algorithm. A Naïve Bayes Classifier is a CM with only one parent which forms the classifier node (see Figure 1). All the other nodes within the network form its children. This unique structure is based upon the principle that all the child nodes are independent of each other, given that we can identify the state of the parent, or classifier node [18].

[a] This is ultimately hardware dependant.

This is not always a wise option as the network structure may not accurately represent the real word domain, and therefore it will be unable to fully capture the variable relationships [17].

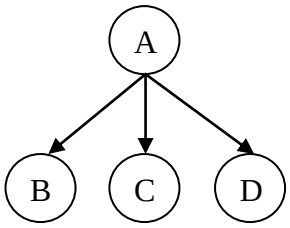


Figure 1: A Naive Bayes Network with node A as the Classifier (parent).

However, there are advantages to using this structure. These include a shorter construction time and most notably a shorter inference time, when compared to networks that allow multiple parents. These advantages can be attributed to the lower number of arcs between the nodes that have to be constructed, compared to nodes with multiple parents.

2.2 K2

The second sub-approach is the metric approach, which incorporates a *search* and *score* technique to finding the CM. This technique attempts to *search* through the entire space of possible node relationships and find the one with the maximum *score*. The most developed algorithm in this sub-approach is the K2 algorithm, which was proposed by Cooper and Herskovits [8] and selects it possible parents using a greedy scoring technique.

There are an enormously large amount of node arrangements that are possible, as with an increase in the number of nodes there is an exponential increase in the number of arrangements. So the K2 algorithm can only isolate the best structure from the set it iterates through. In order to compensate for this, the algorithm has to iterate through many structures to ensure that the best scoring one is found. To score the network, the K2 algorithm requires that the order of the variables be known before scoring can start. This constraint prevents cycles from being introduced into the DAG. The K2 scoring function for an individual node is given in figure 2. The final network score is simply the product of individual node scores.

$$f(i,\pi_i) = \prod_{j=1}^{|\Phi_i|} \frac{(r_i-1)!}{(N_{ij}+r_i-1)!} \prod_{k=1}^{r_i} \alpha_{ijk}!$$

Figure 2: K2 node scoring function [23] (see Appendix A).

2.3 LK2

An improvement on K2 was proposed by R. Cano *et al* [5]. They proposed that within the context of meteorology, the number of potential parents for the K2 algorithm could be reduced based on the geographical location of the nodes or stations. This algorithm is known as the Local K2 or LK2 algorithm. In this algorithm, only the k-nearest stations are chosen to be potential parents for a node. By reducing the number of parents based on their location, they were able to show a decrease in the construction time of the network. Figure 3 details the network construction process of the LK2 algorithm.

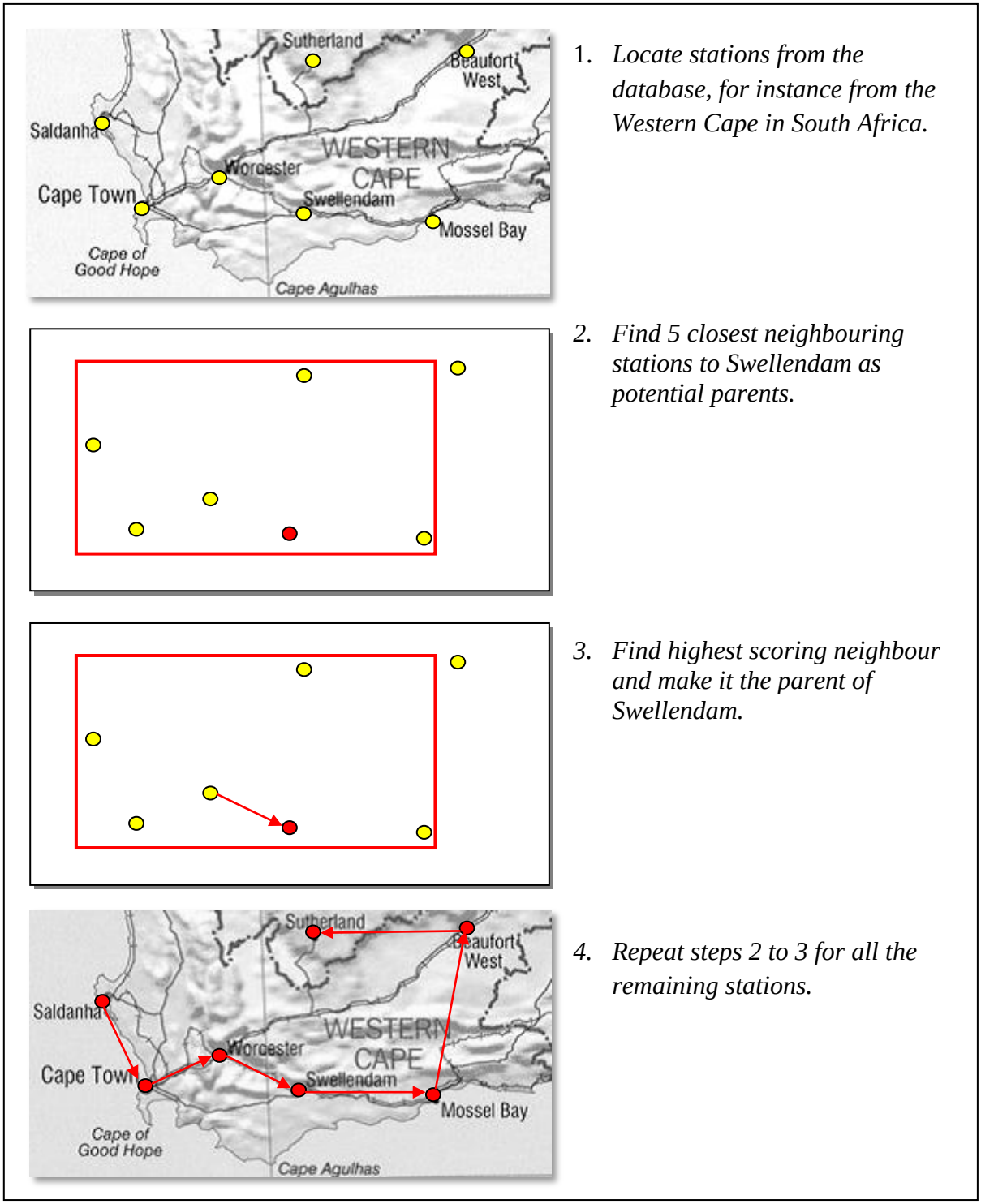


Figure 3: An example showing the steps of the LK2 network construction algorithm (Image courtesy of [19]).

2.4 Greedy Thick Thinning

The Greedy Thick Thinning algorithm (GTT) [11] is a further variant of the metric sub-approach. GTT optimizes an existing structure by modifying the structure and scoring the result. By repeating this process a number of times, GTT is able to isolate the best scoring network. GTT assesses the score of a network originally generated by the PC algorithm. The PC algorithm starts with a fully connected DAG and subsequently removes arcs between nodes based on conditional independence tests. These tests examine three related nodes at a time.

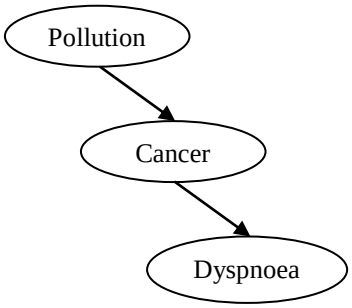


Figure 4: Cancer example of conditional independence [17].

For instance, using the cancer example from figure 4. Say that we know *pollution* causes *Cancer* which results in *Dyspnoea*. If we already know that a patient has *Cancer*, then knowing that they have been exposed to pollution does not have change our knowledge of them having *Dyspnoea*. Therefore the Dyspnoea node is conditionally independent of the Pollution node given that we know the patient has Cancer.

From these tests the PC algorithm is able to determined which nodes are best linked together to form a network. However the PC algorithm does not scale well as if an arc is removed erroneously, the resulting error produced in the required correlations is likely to increase as well [17].

The GTT algorithm offers a solution to this problem by greedily scoring the network, unlike the PC algorithm which does not use a scoring function [6]. GTT considers modifications to the existing network. These modifications involve adding an arc if one does not already exist, and removing or reversing an arc if it does already exist between two nodes. The former process is known as *thickening* and the latter as *thinning* [11]. The network is then scored and if no improvement in the score is found then the algorithm finishes executing. However to prevent only local maxima scoring networks from being found, perturbations in the network are produced after each scoring. A predetermined number of perturbations are used to ensure that the best scoring network is produced [11, 17].

An alternative to the K2 scoring method was developed by L. de Campos [3]. In this algorithm the score is calculated according to a mutual information test (MIT). The MIT score between two variables is a function of its Kullback-Leibler Divergence value [17], and their corresponding marginal dependency values. The Kullback-Leibler algorithm is a measure for testing the accuracy of approximate inference algorithms, and measures the true distribution against the calculated distribution of a node. The corresponding marginal dependency is the degree to which the values of two variables correlate. The MIT value acts as a measure of how highly correlated any two nodes are.

Therefore the more highly correlated the nodes are the greater the probability is that they are directly related. The benefit of the MIT scoring function is that it offers quick convergence to a high scoring network when compared to K2.

Furthermore, two optimizations to the metric sub-approach are the use of decision graphs and candidates. Decision graphs were proposed by Chickering and Heckerman [7], to replace CPTs. They offer an alternative to having all the nodes states stored in the CPT given all the states of its parents. They propose that equal constraints amongst the different states should rather be isolated and represented as a *single* arc in a decision graph. This means that equal relationships over a number of states between the child and parent nodes are now condensed into a single arc structure. This process reduces the space requirement to store the CPT values.

The candidate approach [9], reduces the size of the network space when constructing a CM. It accomplishes this by limiting the number of potential parents to only include those which are part of a candidate set. The candidate set represents a subset of all the possible nodes in the network. The network construction starts with a non-optimal network, scores it, finds the candidate set, and then repeats this process. The candidate set is found by grouping strongly correlated nodes together. Only from this candidate set can parents be selected for a node. The more iterations it performs, the better the candidate set becomes. The candidate set reduces the size of the potential set of parents for the nodes, and therefore increases the performance of the algorithm. It also yields a high scoring algorithm relatively quickly.

The last approach to learning causal models is the hybrid approach. Here two methods are reviewed, constraining parameter estimates, and use of structural restrictions. Constraining parameter estimates [22] attempts to improve on an *already* established network structure, by using knowledge from a domain expert. Information on the structure from a domain expert is used to manually remove various arcs from the network through the use of a software tool. This hybrid form allows the model to benefit from insight that might not be learned directly from the data. This removal of arcs reduces the computational overhead of traversing the structure when forecasting with the CM.

The structural restriction algorithm [4], attempts to translate the knowledge of a domain expert into a set of restrictions. These restrictions are used *during* the construction of the CM to remove and include arcs. This process was found to yield improved network structures, often in less time than K2.

The network construction algorithms all provide ways of building networks in different environments. Domain experts are able to improve effectively on existing networks, though they are not able to manually construct networks in a timely fashion. When there are no domain experts available or a large problem domain exists, learning CMs from the data directly becomes an alternative solution.

3 Weather Model Design

3.1 Weather Model Overview

The objective of the entire weather model honours project was to provide forecasts for weather stations based in Southern Africa. The model was composed of three sections namely the, causal modelling (CM), dynamic Bayesian Network learning (DBN), and visualization system (VS) sections.

3.1.1 Causal Model

The CM section was completed by Michael Kent. The goal of this particular section was to investigate the different techniques used to construct CMs. The CMs would be evaluated on data one day in advance. From this section the best CMs were to be made available for use in the DBN section

3.1.2 Dynamic Bayesian Network Learning

Matthew de Kock was responsible for the DBN section. The DBN section examined the static structures from the CM section in an effort to improve on their predictive accuracy. This was realized through a process of adding temporal arcs between nodes to create Dynamic Bayesian Networks. Once the arcs had been added, an investigation into the forecast range was conducted. Finally a set of forecasts for a period of days was produced and these were provided to the VS section.

3.1.3 Visualization System

The visualization section was completed by Pascal Haingura. This section involved taking the DBN forecasts and visualizing them in a way which was appropriate for two types of users namely, experts and novices. Furthermore this section investigated different approaches in displaying forecasts for a number of days in advance.

3.1.4 Review Meetings

The weather model project combined the research fields of artificial intelligence and atmospheric science from the CS and EGS departments respectively. Furthermore as we had proposed the project ourselves, we saw the need to not only have an expert in the CS department, but to also have one in the EGS department. Mark Tadross offered his insight and experience to aid in the development of our weather model. His research field included seasonal forecasting, which was similar to the goal of our weather model. He aided in providing an EGS context for our work and offered his evaluation of the VS section.

3.2 Section Interfaces

The different sections of the weather model needed a common format in order to pass results from one section to the next, so a series of formats were defined to facilitate this process.

3.2.1 Between the CM and DBN section

The networks produced in the CM section were to be used and improved upon in the DBN section. This required that there be a common format between the two sections, so that the static networks could be transferred. As we were not using a database to store the station data, text files were chosen instead. For each network produced, six files would be sent to the DBN section. Within these files, there was enough information to create and evaluate a dynamic network in the DBN section.

1. The first was a header file, *header.txt*, which contained the all information regarding the construction of the network and the period of the station data or dataset. This consisted of the start and end time for the entire data set. The file also included the total number of nodes and the start and end dates for the training and test sets.

2. The second file was the training set file, *train.txt*. This file contained all the station names and their respective values for the period defined in the header file. This file was to be used during the training of the BN.

3. The third file was the test set file, *test.txt.* This file contained station names and their respective data. However this data was to be used in the evaluation of the CM.

4. The fourth file was the *discretizedTrain.txt* file which contained the discretized station data for use in the training of the CM.

5. The fifth file, the *discretizedTest.txt*, contained the discretized station data for use in evaluating the CM.

6. The last file was the *network.xdsl*, which was an XML file containing the network structure and a list of all the nodes with their corresponding CPT values. This file was used in the DBN section to add temporal arcs to the individual nodes.

3.2.2 Between DBN and VS section

The DBN produced a set of forecasts for a number of days in advance. The VS section had to display this information to the users. Therefore two text files were provided to the VS section:

1. The same header file from the CM section was to provide information on both the forecast period and the bin ranges used in the discretization of the data.

2. A 2D dataset of stations with their corresponding forecasted days, in the form of a text file.

Together, these formats allowed results to be passed between the different sections of the weather model.

3.3 Rapid Prototyping and Causal Model Design

We chose to use the rapid prototyping software design method for developing the weather model. This method was preferred over others such as the Waterfall or KEBN model [17], as it uses a cycle based approach to software development. The use of cycles ensures that there is always a working version of the software throughout its development period.

The advantage of this technique was that it mitigated the effect of risks, such as a team member falling ill or a hardware fault occurring. Should either of these problems have occurred, we would have been able to continue development with the work produced from the last complete cycle. Rapid prototyping therefore mitigated the risks involved in developing the weather model.

The design for the specific CM section of the model was primarily split into two sections, namely the pre and post-prototype sections. The pre-prototype section was focused on gaining knowledge on how Bayesian Networks operate, and investigating potential tools to construct BNs with. From this knowledge the original prototype was then developed and presented.

3.3.1 Original Prototype

The original prototype for this specific section contained a simple Naive Bayes algorithm which successfully learnt a three node network. The data that was used was provided by a tutorial within the Smile software package[b]. After completing the prototype we began working on the design for the whole CM section.

3.3.2 Prototype Cycles 1 to 3

The subsequent post-prototype part of the CM design concentrated on defining the contents for the individual cycles within the section.

The first cycle of the CM section was set apart for cleaning the data. The station data was in plain text files and needed to have errors removed, and queried data discretized.

The second cycle was designed to provide a theoretical background to the CM section. This included research into basic statistical methods and Bayesian Network theory. Furthermore the second cycle included investigating evaluation techniques, which would be used to assess the produced networks.

The objective of the third cycle was to implement and evaluate the various algorithms. During the duration of this cycle the optimal network structures would be developed and provided for use in the DBN section. Figure 5 represents the planned and actual development dates for the CM section.

Rapid Prototype Cycles	Start Date	End Date	Duration
Cycle 1: Data Cleaning			
Planned	1 Aug	22 Aug	21 Days
Occurred	19 Aug	2 Sep	14 Days
Cycle 2: Background and Evaluation Methods			
Planned	22 Aug	12 Sep	21 Days
Occurred	2 Sep	24 Sep	22 Days
Cycle 3: Implementation and Evaluation			
Planned	12 Sep	03-Oct	21 Days
Occurred	24 Sep	22-Oct	28 Days

Figure 5: Duration of Rapid Prototype Cycles

[b] The Smile software package is discussed further in section 5.1.2.c

In retrospect the greatest concern was the delay in development. At the time of starting the first cycle we had to devote our time to other work. This led to an almost three week delay in the development of the project. Though we manage to complete the data cleaning cycle in less time than anticipated, we could not recover the full time lost on other work. The result of this was that less time was available for evaluating the results and completing the report, than was originally planned for.

3.3.3 Abstract Class Diagram

The last part of the design of the CM section concentrated on the classes required, and their relations to each other (Figure 6). The CM algorithms were all implemented through the "Algorithm" class except for the Best Naive Bayes Classifier (BNBC) as this class used the predictive accuracy function of the "Evaluation" class. Therefore the BNBC was implemented directly in to the "Evaluation" class to prevent the redundancy of implementing two predictive accuracy functions.

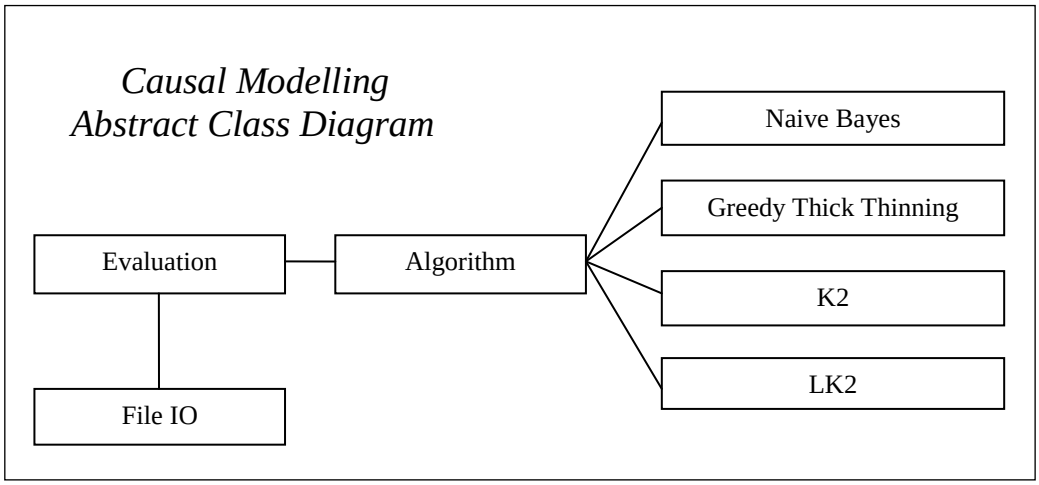


Figure 6: Abstract Class Diagram for the Causal Modelling Section.

3.3.4 Deviations from Proposal

During the development phase of the CMs, a number of changes had to be made with respect to the original proposal. The first of these was the use of station data. In the proposal we sought to use data from the *MyWeather* database that was provided by the EGS department. This database contained eight weather stations in total that were geographically located around the Cape Peninsula. However constructing networks with so few nodes was not inline with similar works [2, 5] that used 167 and 100 nodes respectively. So in view of this information we chose to use the station data from Southern Africa which contained significantly more stations.

The second deviation from the proposal was the inclusion of the effects of other weather variables on our forecasts. The data we received only contained data for minimum and maximum temperatures as well as precipitation levels. No other data was available for comparing their effects to our forecasts. Therefore we were unable to determine the effects of other weather variables on our forecasts.

3.4 Proposed Outcomes

The objective of this section of the weather model project was to investigate, implement, and evaluate different CM techniques. The artefacts produced would include:

- For each network, 6 data files would be created to enable their further development in the DBN section. We knew that this could be done for the Naive Bayes algorithm at the time of the first post-prototype cycle. This was due to prior successful use of the algorithm in the prototype.

- A list of algorithms which constructed the networks with the highest predictive accuracies and lowest construction times.

The DBN section would further enhance networks based on the static structures developed in this section. Therefore the more accurate the static networks were, the more accurate the dynamic networks would be. These results would then be reflected within the VS section.

4 Data Pre-processing

4.1 The Data and Area of Study

In Southern Africa there are a number of weather stations distributed over the land (see Figure 7). These stations record the state of the atmosphere on a daily basis. They are able to record minimum and maximum temperatures, as well as rainfall or precipitation levels for their location. By combining the weather data from all these stations we are able to get a picture of the state of the atmosphere for Southern Africa.

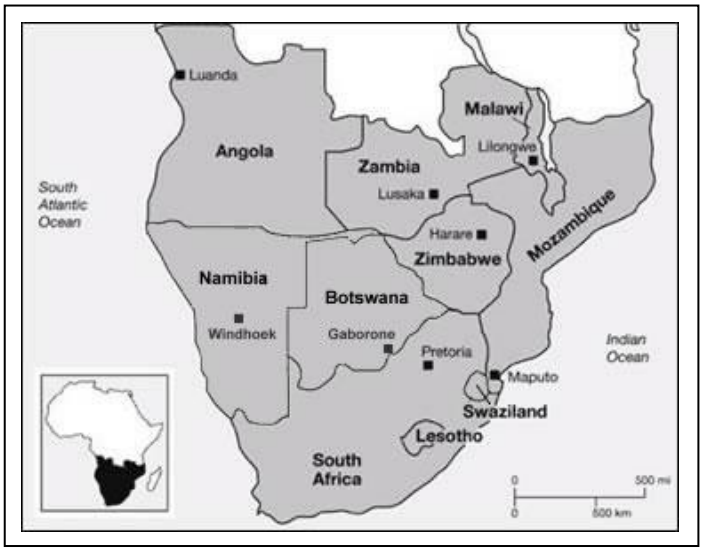


Figure 7: A map of Southern Africa representing the research area [20].

The Department of Environmental and Geographical Science (EGS) at the University of Cape Town (UCT) has a database of these stations and their recordings. The database is the combination of the station data from numerous external sources which has been operational between 1850 and 2006. We were able to copy and use this data in our project to model and forecast the weather in Southern Africa.

The data we received were in the form of text files. For the each of the three weather variables (Eg: minimum temperature) there was a header file accompanied by a set of station files. This set of station files represented the recordings for a weather variable by the various stations. In total there were 3203 stations for minimum temperature, 3204 stations for maximum temperature, and 6820 for precipitation levels.

As one of the sections of this project involved visualizing the station data, we saw the need to have a set of stations that were common to all three weather variables. This set of stations would allow comparisons to take place between the different weather variables within the visualization section. The final set of stations that was common to all three weather variables consisted of 2108 stations.

However not all the stations had been in service since 1850 and some of the stations had periods of time were no observations were recorded. Furthermore, the same station from multiple sources had different latitudinal and longitudinal values. The significance of this was 2km difference in location when using the Haversine distance formulae [10]. Therefore a process of data cleaning was started.

To calculate the missing values in the station data, the expectation maximization (EM) [17] algorithm was used. For the stations of each weather variable, the EM algorithm was applied to the missing data, which was denoted by "-999" in the station files. We assumed that sufficient convergence of the values had occurred after 200 iterations of the algorithm. The convergence value was determined by the amount of time we wanted the EM algorithm to take per station. To ensure that the most accurate station coordinates were used, we isolated the best coordinates from the maximum temperature data and included them in our processed data.

In light of the range of operational periods, a query tool was designed to extract station data from the files within a specified range. Furthermore, by specifying the range, we would also be limiting the number of stations that had recordings in that period. This was found to reduce the number of nodes to be included in the BNs, which was also inline with other similar projects [2, 5] that used 167 and 100 nodes respectively.

Due to us only using queried subsets of the station data, we did not see the need for a database. In addition to this, the overhead of creating a database, the time spent maintaining it, and the uncertainty how to pass results between the different project sections all made the use of a database a risk. So as to mitigate our risks, we chose to not implement a database.

At the end of these processes we had a set of files containing the cleaned station data for the three weather variables, and a query tool to extract the station data with.

4.2 Data Discretization

All the station data extracted by the query tool had an associated range. For instance the range for the maximum temperature recorded by a station could start from 1.6°C and peak at 45.6°C depending on the parameters of the query. In order to represent all the states that the station could be in, the node would require a CPT of approximately 460[c] values. This was not a feasible option as the number of states of the CPT grows exponentially with an increase in the number of parent nodes [17].

[c] A node ranging from 0°C to 45°C with an accuracy of $1/10^{th}$ of a degree requires a CPT of 460 states.

In view of this problem we adopted the process of discretization. Discretization is a technique which reduces the range of data. The values of the data are reduced into a number of bins, the amount of which is specified by the user. The number of bins represents the degree to which the data range will be reduced by. The specific discretization technique employed was the Uniform Bin Count technique, which was provided by the software. This technique determined the minimum (X_{min}) and maximum (X_{max}) values of the data, and calculated the range (σ) or size of each of the *k* bins according to the formula in figure 8.

$$\sigma = \frac{X_{max} - X_{min}}{k}$$

Figure 8: The Uniform Bin Count formula [15].

If the same above station was discretized using a bin count of 5, the final values for each day recorded by the station would only range from 0 to 4. This would reduce the range of the station from 460 values down to 5.

This discretization technique was incorporated into the query tool. So any subsequent queries would result in a set of discretized data being produced in addition to the non-discretized data. The sets of data were divided into training and test sets for the creation and evaluation of the CMs. The CM learning uses the training set to find the relationships between the different variables. The test set is used for the evaluation of the network constructed from the training set. If however these two sets were not used and rather one large dataset was used, the model would suffer from *overfitting*. Overfitting results in the evaluation of the model being bias towards the training set, as the model is evaluated on the same data that it learned from [17]. Therefore to prevent this problem we implemented the use of separate training and test sets.

Once the desired station data had been cleaned and discretized by the query tool, it was then suitable for BN construction.

5 Implementation

This section discusses the details of how we implemented the design of the CMs.

5.1 Development Environment

5.1.1 System Specifications

The networks were all constructed and evaluated on one desktop computer with the following specifications.

- Operating System: Gentoo Linux
- Processor Type: AMD Athlon^{tm} 64 Processor 3500+
- Processor Speed: 2.21 GHz
- Memory: 1GB

Gentoo Linux was chosen as the operating system to use in this section of the project as it is optimized for each unique architecture, unlike Ubuntu or Windows XP. Therefore Gentoo was chosen as it would be able to provide the best performing platform to develop with.

5.1.2 Software Tools

There were various software tools that were considered for the development of the causal models.

5.1.2.a. The Probability Network Library

The Probability Network Library (PNL) is based in C++ and was developed by Intel. PNL offers DBN support, but only for first order processes [14]. As part of the objective of the DBN section was to examine forecasts spanning multiple days, first order support would not have been sufficient. Furthermore, we were advised to not use this software tool by Anet Potgieter as it required six months of prior experience before successful implementation could commence.

5.1.2.b. JavaBayes

JavaBayes is a BN tool based in Java [16]. It consists of a user interface, a core inference engine, and a set of parsers. As JavaBayes in implemented in Java, it is also dependant upon the Java Virtual Machine (JVM). This was seen as a constraint to the performance of the software, as we needed fast inference support for the DBN section.

5.1.2.c. Smile and Genie

Smile is a set of C++ header files and libraries compiled for developing BNs in either Linux or Windows [14, 24]. Genie is the front end visualization tool used primarily for displaying BNs that are generated in Smile.

Smile provided a framework in which to construct BNs. This framework included network and node structures with support for learning them. The source files are not provided and therefore only extensions of the existing framework are possible. The default CM algorithms that it provided were the Naive Bayes and Greedy Thick Thinning algorithms.

Furthermore the Smile network data structure provided a set of inference algorithms to learn evidence with. This included the default junction tree clustering algorithm which uses exact inference to calculate the resultant states of a node. Therefore we chose to use the Smile and Genie software package to develop with.

5.1.2.d. Netbeans with C++ plug-in

Neatbeans is an interactive development environment that was used to create and compile the C++ classes. It was chosen because of the code completion feature that it possesses. This feature allowed partially constructed variable, and method names to be automatically completed by the software. We found this feature greatly enhanced our ability to learn the Smile software language.

5.2 Causal Model Algorithms

5.2.1 Naive Bayes Classifier

These advantages of Naïve Bayes Networks make them best suited to problem domains which contain a large number of random variables, and where the conditional independencies between the child nodes can be assumed.

Furthermore their short construction times make them ideal candidates for real time system modelling. Given this information, we have implemented the Naive Bayes algorithm to investigate if, and how the reduced inference time and conditional independence assumption affects its accuracy.

The Naïve Bayes algorithm was provided as part of the Smile software and was implemented in the *NaiveBayes* class. The default classifier node is specified to be the first node within the training set (dataset), though this can be modified to be any node within the network. The selection of the classifier node can be performed through many processes including, through eliciting knowledge from a domain expert, or from a clustering program (Eg: Weka [25]) which combines multiple nodes into the single parent node. As part of the goal in this project was to be able to produce a forecast for individual nodes, clustering them was not seen as an option. Rather individual stations were selected to become the potential parent.

5.2.2 K2

The K2 implementation in this project was *not* provided by the software. Rather it was constructed based on the formula in figure 2 and trained using a sample set (see Appendix A). By producing the same output as the sample set, we could assume that we had produced the same algorithm. However the α_{ijk} term had to be modified in our implementation. In the original formula it represents the number of training set correlations of potential parents, to the current node. Following this, the factorial of the resultant value is then calculated. The pseudo code for the K2 algorithm can be seen in figure 9, with π_i representing the parents of node x_i and Pred(x_i) representing all the predecessors of x_i.

```
1.  Procedure K2:
2.  {Input: A set of n nodes, an ordering on the nodes, an upper bound u on the
3.          number of parents a node may have, and a database D containing m cases.}
4.  {Output: For each node, a printout of the parents of the node.}
5.  for i := 1 to n do
6.     π_i := ∅;
7.     P_old := f(i, π_i );
8.     OKToProceed := true;
9.     While OKToProceed and π_i < u do
10.       Let z be the node in Pred(x_i) - π_i that maximizes f(i, π_i ∪{z});
11.       P_new := f(i, π_i ∪{z});
12.       if P_new > P_old then
13.          P_old := P_new;
14.          π_i := π_i ∪ {z};
15.       else OKToProceed := false;
16.    end {while};
17.    write("Node: ",x_i "Parents of x_i: ",π_i);
18. end {for};
19. end {K2};
```

Figure 9: The pseudo code for the K2 algorithm [23].

In the sample set there were only 10 elements, but in our implementation the training set was much larger with 602 elements. This posed a computational problem which was resolved by representing the count as a percentage of the total training set size. This however reduced the accuracy of the count as the percentage calculation rounds off values to the nearest integer. This in turn reduced the score of the individual nodes and subsequently the network score (see Appendix A).

Nevertheless when our implementation of the algorithm was run using the sample set, the final structure was the exact same as that produced without the modified term. The only difference was the modified score of the nodes and the network, which was notably smaller than the original scores.

Another modification was also made to our implementation of the K2 algorithm. During the scoring of an individual node, correlations are made between the current node and its potential parent set. This process compares all the possible states of the child node to all the possible sates of all the current parents. This process of comparing the current node to its parents involves finding correlations between its station data, and the data for the parents of the node.

Subsequently due to the size of the CPT[d], the algorithm is not always able to find that same number of combinations of states within the training set. This can lead to those states in the node that are not observed in the training set of having zero or no probability of occurring in the CPT.

This particular problem was exemplified in the case of precipitation values, which tended to only be observed in extreme high or low values. The result of this is that the node was never able to assume or forecast the value in the future. We could see that this weakened the forecast ability of our model, and so included a simple counting solution to solve this problem.

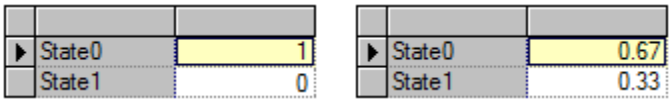


Figure 10: Incorrect count (left) and corrected count (right).

The counting solution [17] makes the assumption that all the possible values in the CPT are observed once before any correlations are actually counted in the training set. Following from this is that the count for an individual state for a node, given the states of its parents, is never zero. Thus the probability of the node assuming that state, is no longer zero. In figure 10, the left CPT has no (zero) probability of forecasting state 1, while the corrected count does have ability to forecast sate 1. The advantage to this is that the node was then able to observe or forecast that value in the future, which made the network more robust.

[d] The size the CPT = number of states$^{number\ of\ parents+1}$

5.2.3 Greedy Thick Thinning

We chose to implement GTT as a comparable search and score technique to BNBC, K2, and LK2. Therefore it would serve as a benchmark with which to compare and contrast the above mentioned algorithms with. It was also provided by the Smile software package, and was implemented in the *greedythickthinning* class.

5.2.4 Best Naive Bayes Classifier

The Best Naive Bayes Classifier (BNBC) extends the idea of the Naive Bayes Network. It achieves this by not only constructing a Naive Bayesian network, but also by finding the one with the highest accuracy using a brute force technique that we developed.

The BNBC algorithm constructs each possible network structure from the given training data set. This involves taking each node and making it the parent node, while the rest of the nodes become the children of the current node for that specific iteration of the algorithm. At the end of each iteration, the accuracy of the produced network is calculated. If this node has produced a network which is more accurate, then this node becomes the BNBC.

The desired result should be a network with the same advantages of the Naive Bayesian structure, but with a higher accuracy than if no parent selection was applied.

5.2.5 LK2 Algorithm using Haversine Distance

We chose to investigate the LK2 algorithm in order to determine what the effect of choosing the k-nearest neighbours has on the accuracy of a network, and if it is able to improve on the construction time of K2 as according to [5].

As the LK2 algorithm is modification of K2, we constructed it ourselves. The set of potential parents was determined before the scoring of the network or individual nodes took place. During this prior process the k-nearest neighbours for each node were found and stored in a list. However this search process is still performed within the context of the traditional ordering of the nodes that was used in the K2 algorithm. This maintains that no cycles are formed when arcs are created between nodes.

The set of potential parents is determined by calculating the k-nearest nodes. This calculation uses the Haversine distances [10] between the stations. In the original paper it is unclear as to which distance formulae was used. The Haversine distance is a formula for calculating the distance between two points. However unlike the Euclidian distance formula, the Haversine formula takes the curvature of the Earth into account. This therefore produces a set of distances between the nodes which better represents the actual geographical distribution of the stations. The Vincenty distance formula [10] was also considered as an alternative, but in view of the degree of inaccuracy that the location data displayed, the precision of the resulting distance would have been lost.

Once the set of potential parents has been determined, all subsequent parents are selected from that set. With the K2 algorithm the selection of parents involved determining which parents preceding the current node in the node order, increased the score of a node.

By only selecting the geographically near stations, the LK2 algorithm reduces the size of the parent set. The result of this is that for each iteration[e] of the algorithm there is a constant number of parents that the node can be linked to.

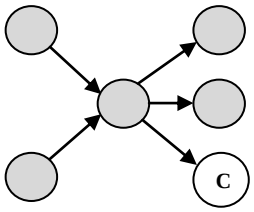
However the K2 algorithm does not limit the number of potential parents, and so subsequent iterations of the algorithm involve more potential parents to select from. This is reason behind the proposed improvement in construction time.

Following from our construction of the K2 algorithm, the LK2 algorithm also has the modified α_{ijk} term and the counting solution which prevents probability values of zero from occurring in CPTs.

5.3 Causal Model Evaluations

5.3.1 Predictive Accuracy

Predictive Accuracy (PA) [17] is an evaluation technique which entails calculating the number of correct forecasts against the total number of possible forecasts.



Node C	
Rain	0.7
Sun	0.3

Figure 11: Calculating the PA for node C while observing all other nodes (shaded).

The process of calculating the PA for a network is the average PA for all the nodes in the network. When working with a single node, the algorithm assumes that all the other nodes in the network have been observed. The observations for the other nodes are taken from the test set produced by the query tool. After setting the evidence for the other nodes, the effect of their state change on the current node is calculated through inference. The outcome for the node is a set of resultant states for it, along with their probabilities of occurring. From these states, the sate with the highest probability is taken to be the sate that the station will forecast. For instance in figure 11, node C would forecast *rain* based on its CPT, when all the other nodes are observed.

This forecasted state is then compared to the equivalent day in the test set. If the forecasted state is the same as the test set sate, then that station has forecasted correctly for that day. The node has the PA calculated for all the days within its corresponding test set. The PA algorithm is then applied the rest of the nodes of the network, and the average PA for all of them is taken as the PA for the network.

[e] This excludes the first few iterations, where the number of prior nodes is less than the number of neighbours.

5.3.2 Causal Model Experiments

A set of experiments were designed to evaluate the performances of the networks produced by the different algorithms. The results are represented graphically with their original corresponding tables provided in Appendix B. All construction times were taken in microseconds but are displayed in other formats for visual clarity. During the evaluation of the networks, two sets of data were used.

Dataset 1:

The first dataset was generated by the query tool for the period from 1 January 2004 to the 30 October 2005. This period was chosen as it yielded exactly 100 nodes (stations) on which to base the comparisons. The training set size was set to be 90% (602 days) of the total period, which made the size of the test set to be 10% (67 days). The training and set percentages were recommended by [17]. The weather variable selected was maximum temperature (TMAX). Furthermore there were 5 bins used in the discretization of the station data.

Dataset 2:

The second dataset was generated by the query tool for the period from 1 January 2001 to 31 December 2001. This period was chosen as it yielded more than 100 nodes (stations) for each weather variable. From this same period, 100 stations common to all three weather variables were found in order to facilitate inter weather variable comparisons. The training set size was set to be 90% (328 days) of the total period, which made the size of the test set to be 10% (37 days). Furthermore there were 5 bins used in the discretization of the station data. The final results were 3 datasets for the same time period all containing the same number of training and test days.

5.3.2.a. Experiment 1: Varying the Number of Parents

Objective:

The objective of the first experiment was to determine what the difference on predictive accuracy and construction time are, when increasing the potential number of parents that a node can choose from.

Method:

The experiment was run using Dataset1, and compared the K2, LK2, and GTT algorithms. The LK2 algorithm was set to use 10 nearest neighbours. The BNBC was not compared as it can only ever have one parent in the context of this report.

Results:

The results produced for the effect of varying the number of parents can be seen in figure 12 and figure 13.

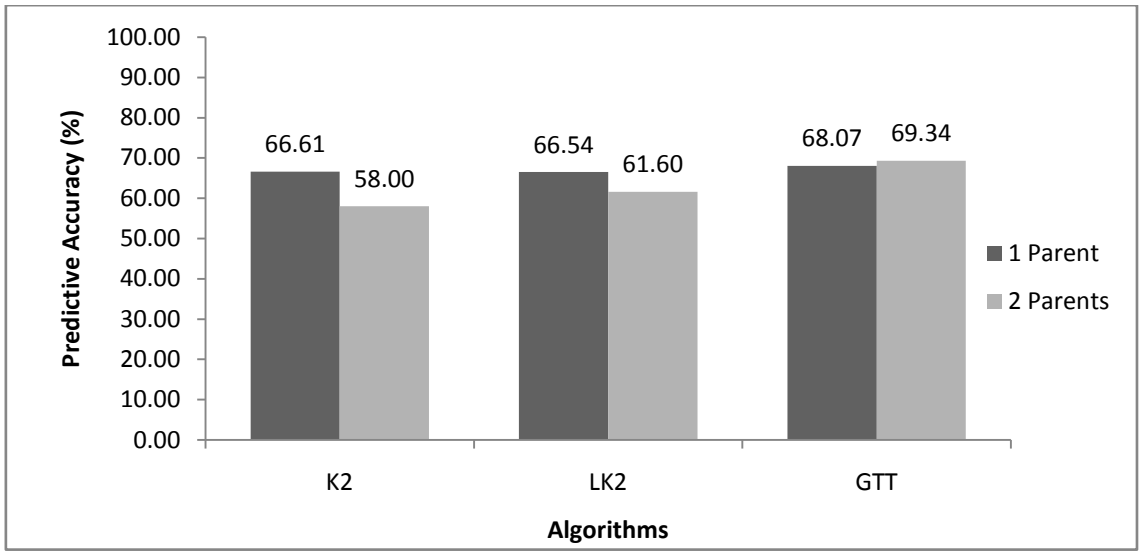


Figure 12: The effect of additional parents on predictive accuracy.

From figure 12 we can see that the K2 and LK2 algorithms produce networks which have the highest PA when using 1 parent. However the GTT algorithm produces its highest PA when using two parents.

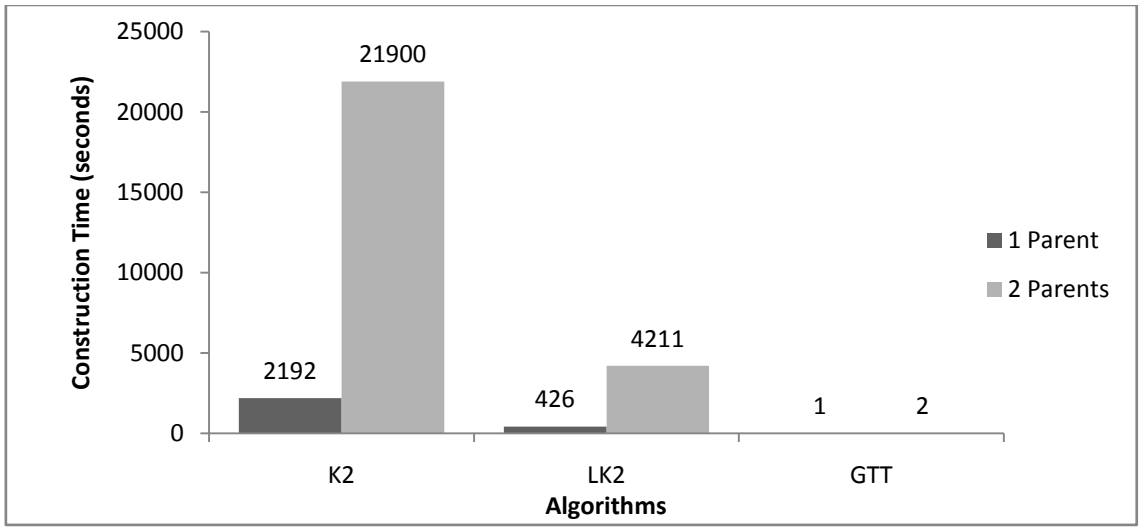


Figure 13: The construction times of the networks with additional parents.

The construction time results (Figure 13) show that for every algorithm there is an increase in the construction time when additional parents are included. This is what we expected as an increase in the number of potential parents for a node increases the amount of values in each CPT. Therefore as the CPT grows exponentially with an increase in the number of parents, so too does the amount of time required to correlate the child data with the data of the parents.

However, in the original paper [5] the construction time of their K2 network was less than 120 seconds. Our implementation however, took approximately 6 hours to complete. Although their evaluation consisted of networks containing 50 or less nodes, if their algorithm had used 100 nodes it would most likely have finished within 6 hours.

In view of this evidence further research was conducted into the construction of the K2 algorithm. In the original sample data (see Appendix A) that was used to construct the K2 algorithm, we did produce the same structure and the same scores as the sample nodes did. However we did find the algorithm was not able to scale from the 10 elements in the sample to the 602 elements that we trained with. Therefore we implemented the modified α_{ijk} term to overcome this problem. However we were unaware of the optimal data structure that was used in the original algorithm proposed by Cooper and Herskovits [8]. In the original algorithm, *index trees* were used to calculate α_{ijk}, while in our implementation a series of iterations were used. This resulted in a sub-optimal implementation of the algorithm with respect to construction time.

Despite this and the fact that multiple node orders therefore could not be generated within the period of this project, the K2 and LK2 algorithms still used the same node order and modified α_{ijk} terms. Therefore the two algorithms remained comparable in both construction time and PA.

5.3.2.b. Experiment 2: LK2 with Varying k-Nearest Neighbours

Objective:

For this experiment we wished to investigate the effect of varying the number of neighbours that the LK2 algorithm can use. This experiment was based on the results and conclusions that were presented by R. Cano *et al* [5]. With an increase in the number of neighbours we should see an increase the construction time, as the algorithm becomes more like the K2 algorithm in performance.

In addition to this we will evaluate what effect a change in the number of neighbours has on the predictive accuracy of the networks.

Method:

The experiment was run using Dataset1 while the number of k-nearest neighbours and the number of parents were allowed to vary. The LK2 algorithm was applied to the dataset.

Results:

The outcomes from this experioment show (Figure 14) that there is always an increase in construction time with an increase in the number of neighbours for dataset 1. This is in agreement with our prior objective as with an increase in the number of nieghbours there are more parents that the scoring function has to search through in order to find the best sccoring parents

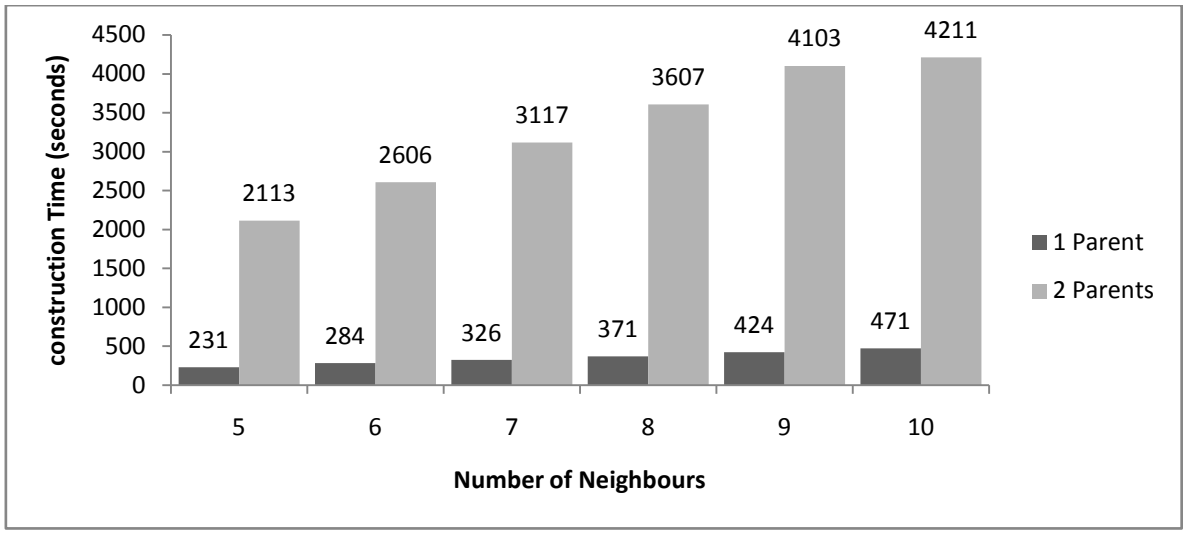


Figure 14: The construciton time of the LK2 networks with a varying number of neighbours and parents..

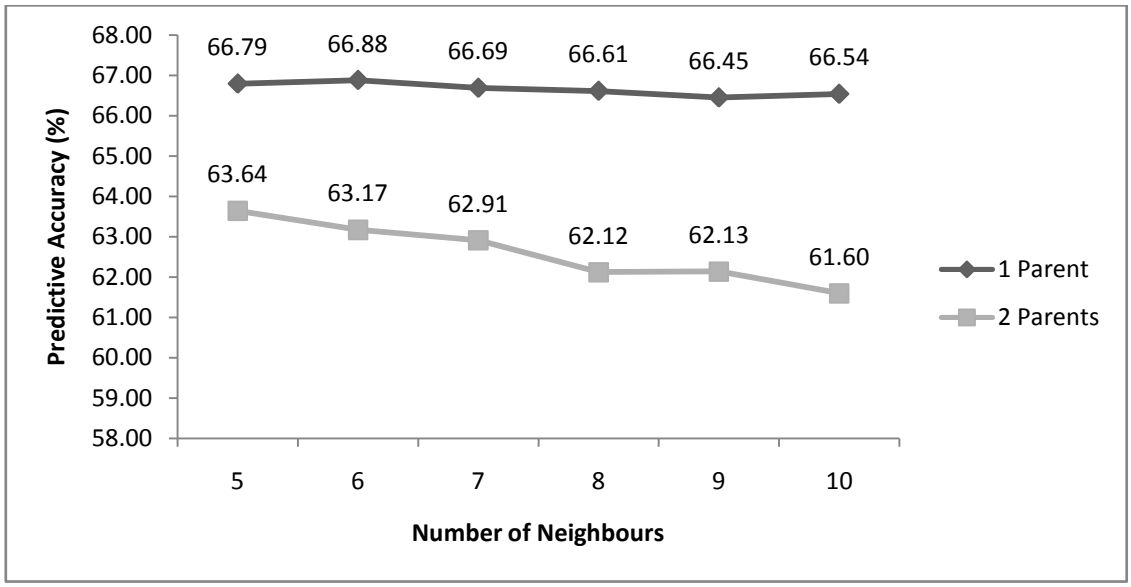


Figure 15: Construction time with varying number of neighbours.

From figure 15 we can see that varying the number of neighbours when only one parent from them is selected, shows no constant decreasing tread in PA. This can be attributed to the fact that the single closest neighbour most likely also has a similar node score, which increases its chance of being selected as the parent.

However when considering selecting 2 parents from the set of neighbours, there is an over all decrease (except for 9 neighbours) in the PA for the network. Therefore we can see that as the algorithm is able to select more neighbours for a node, their geographical significance to the current node decreases. Therefore by limiting the parents to the set of stations closest to the node, we will increase the PA of the current node and subsequently the network.

When compared to K2, the LK2 algorithm was sometimes able to produce a higher PA when using 1 parent (Figure 14). However when LK2 constructed networks with 2 parents, it was able to produce networks with higher PA than the 58% that K2 produced. Therefore with multiple parents and 5 nearest neighbours the LK2 is able to out perform the K2 algorithm in both network construction time and PA.

5.3.2.c. Experiment 3: Effect of Weather Variable Data on Predictive Accuracy

Objective

For the third experiment we wanted to investigate the effect that the different weather variables had on the predictive accuracy and construction time of the networks. The results from these should indicate whether we can select the best algorithm for all weather variables, or whether we will have to select a different algorithm for each one.

Method:

The experiment was run using Dataset 2 including the best parameters which maximized PA from experiments 1 and 2. These were the best number of parents for each algorithm and the best number of neighbours. The K2 and LK2 used 1 parent, while GTT used 2 parents. The LK2 selected parents from 5 nearest neighbours.

The average Naive Bayes (Ave NB) is the average for all the possible parents that the Naive Bayes classifier can use as the parent. It represents the difference between selectively choosing the parent node as opposed to randomly choosing it from all the possible nodes.

Results:

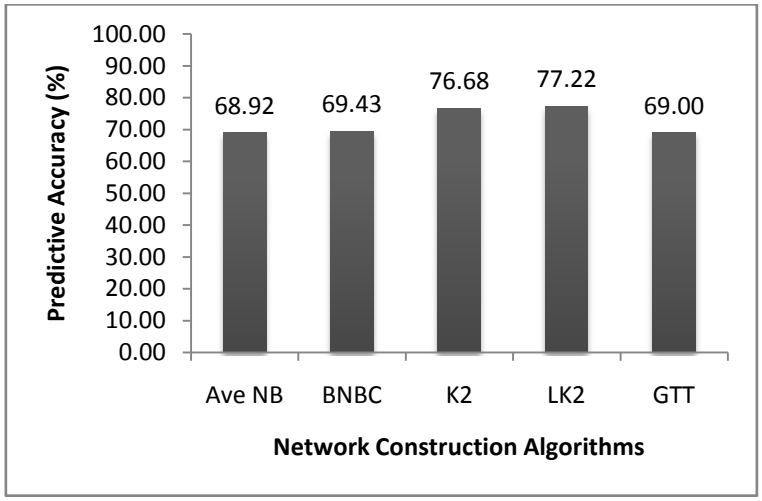


Figure 16: Predictive accuracy with precipitation data.

The results for the precipitation data effect on PA (Figure 16) show that LK2 has the highest accuracy. The reason for this is the locality of the parents being geographically close to their child node. The significance of this has been shown in experiment 2. We can see that the BNBC does not improve the PA much (less than 1%) when compared to randomly selecting the parent.

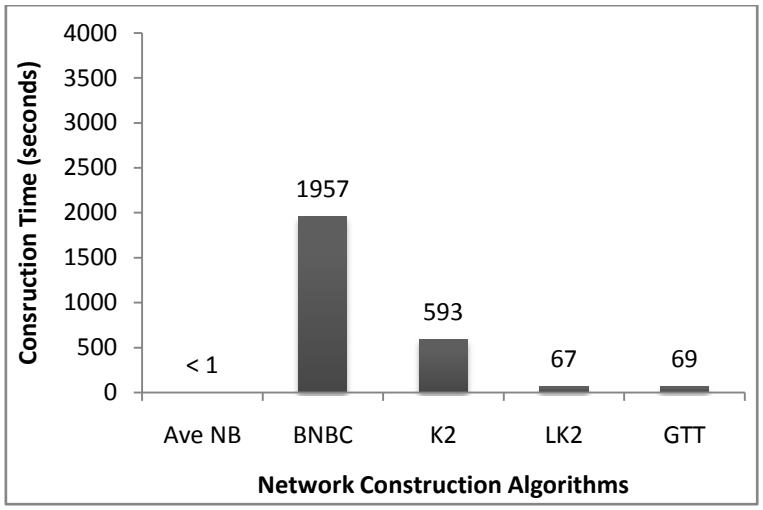


Figure 17: Construction time with precipitation data.

We can see from figure 17 that the average NB algorithm constructs in less than a second. While the BNBC takes the most amount of time to find and construct a network. Therefore in the case of using the BNBC for improved PA, the BNBC only improves by less than 1% PA, but requires much more time than the average NB does.

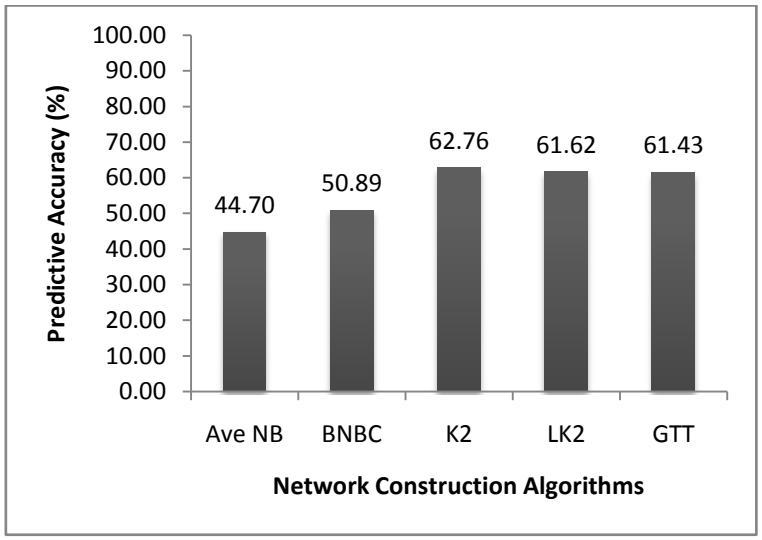


Figure 18: Predictive accuracy with minimum temperature data.

For the results produced by the minimum temperature we can see that the K2 algorithm predicted (Figure 17) the forecasts with the highest PA. Therefore in this case the correlations of the data on which the K2 scores, are more significant than the locality or network perturbations that the LK2 and GTT algorithms use respectively in constructing networks.

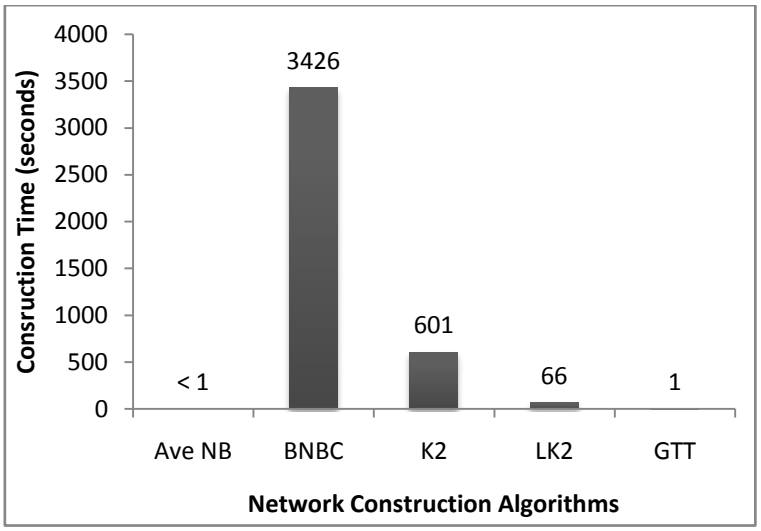


Figure 19: Construction time with minimum temperature data.

The minimum temperature construction times in figure 19, shows similar results to those produced using the precipitation data (Figure 17). However the increase in construction times for BNBC, K2 and LK2 are most likely due to anomalies within the data. An interesting aspect that we can see is that the GTT algorithm takes *less* when constructing a network based on minimum temperature data, than when constructing a network based on precipitation data. The perturbations of the GTT algorithm are most likely the source of the fast construction time.

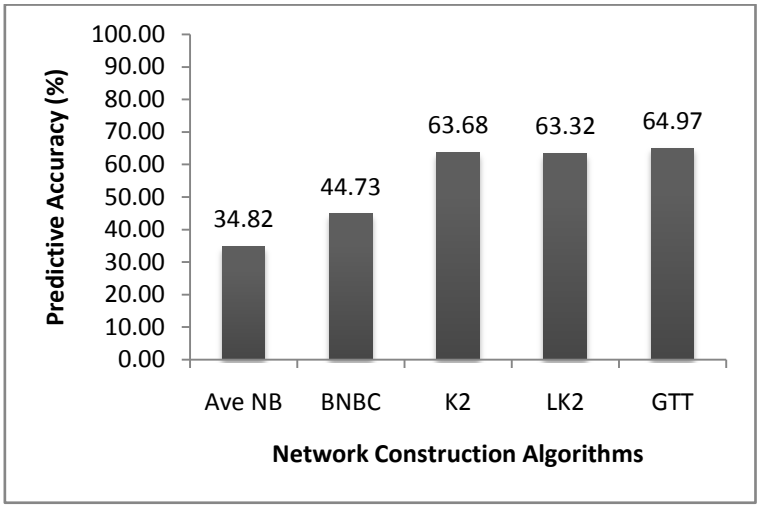


Figure 20: Predictive accuracy with maximum temperature data.

The maximum temperature (Figure 20) shows the best PA for the GTT algorithms. We can also see that the average NB scores the lowest PA. The construction times in figure 21 are similar to those for precipitation and minimum temperature (Figures 17 and 19). These results too show the NB with the lowest construction time, with the BNBC taking the most amount of time to construct a network.

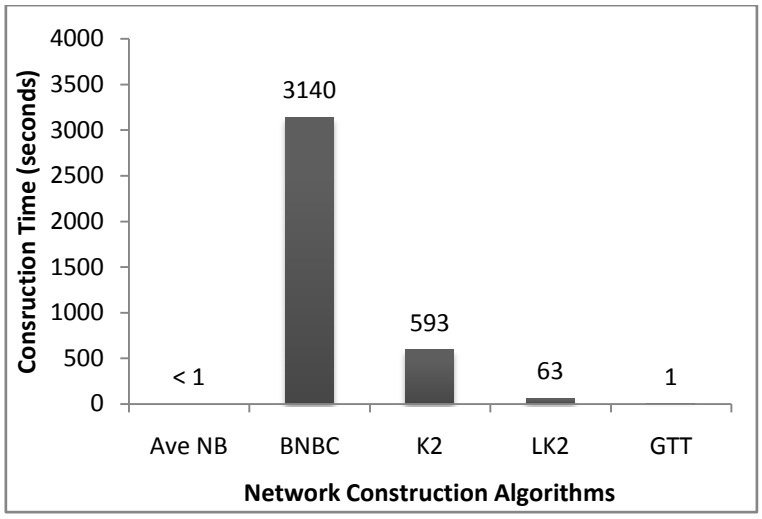


Figure 21: Construction time with maximum temperature data.

These results from this experiment show that different algorithms work best for different weather variables. The LK2 forecasts best for precipitation while the GTT forecasts the best for minimum and maximum temperature. Although BNBC improves on the PA when using different weather variable data, the added construction time out weighs the PA gained. Therefore BNBC is not a viable option as other algorithms (Eg: GTT, LK2) produce networks with higher PA in less time.

5.3.2.d. Experiment 4: Effect of Varying Bin Count on Predictive Accuracy

Objective:

As the uniform bin count discretization technique was used, we wanted to determine what effect varying the bin size would have on the construction times and predictive accuracies of the different networks. So we conducted a varying bin size discretization experiment to investigate these effects.

Method:

The experiment was run using Dataset 2, but the query tool did not discretize the data for this experiment. Rather the non-discretized data was discretized using a varying number of bins. We chose to range the number of bins from 3 to 7 as these values were 2 less than and 2 greater than the current number of bins that we were using. The BNBC, K2, LK2, and GTT algorithms were used. The BNBCs had already been determined for these datasets in experiment 3, and so the subsequent stations were used to construct the BNBC. These were nodes 99 (station 596) for precipitation, 82 (station 572) for maximum temperature, and node 31 (station 474) for minimum temperature. Therefore the construction times for the BNBC will not reflect the time to find it, but rather the time to construct it given that we know the PA of the best station already. The K2 and LK2 used 1 parent each and GTT used 2 parents. The LK2 algorithm selected parents from the 5 nearest neighbours.

Within this report TMAX refers to the maximum temperature, TMIN refers to the minimum temperature, and PPT refers to precipitation levels.

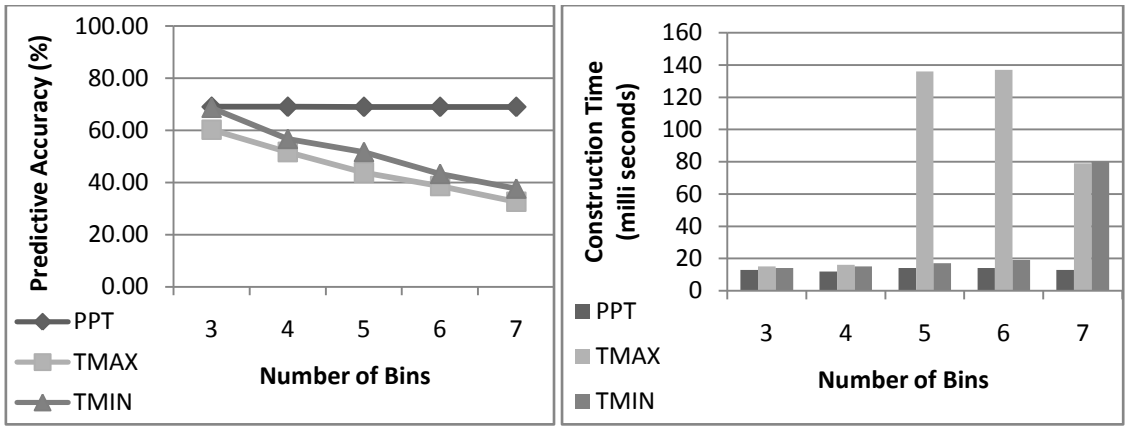


Figure 22: The predictive accuracies and construction times for BNBC.

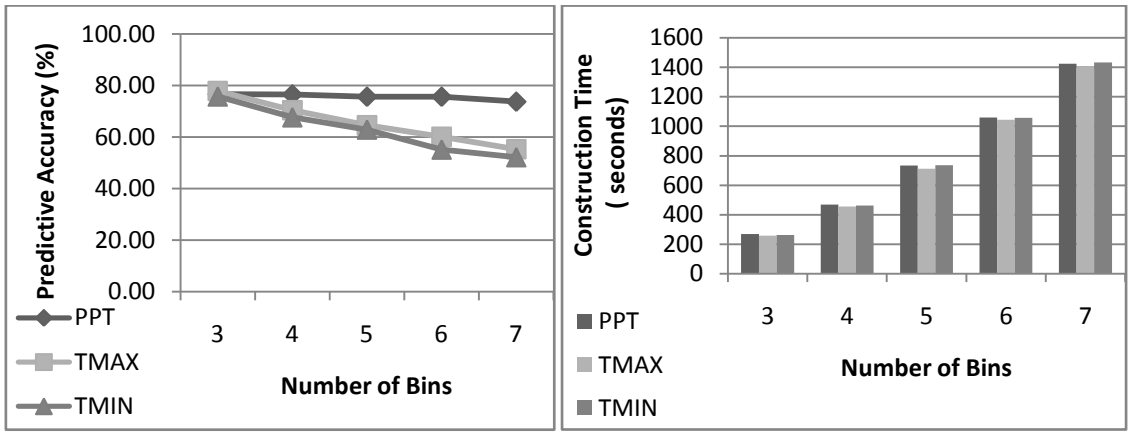


Figure 23: The predictive accuracies and construction times for K2.

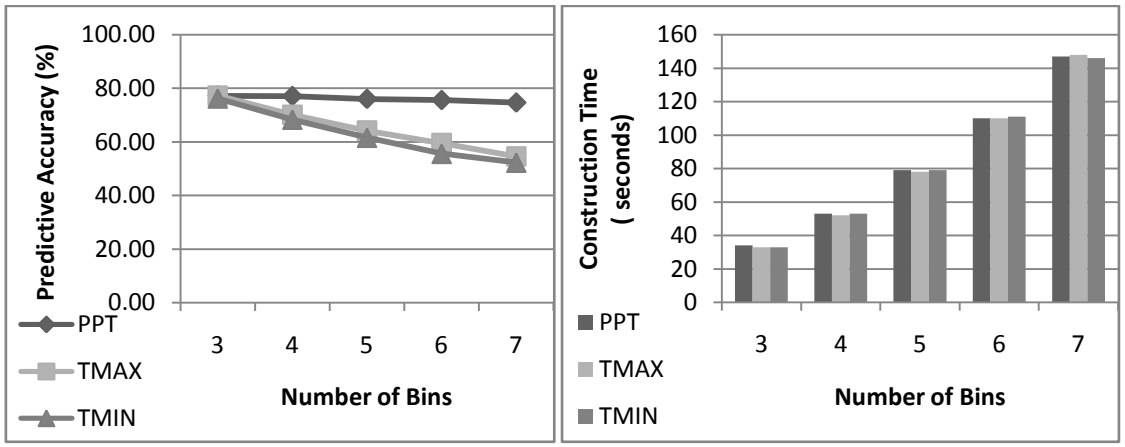


Figure 24: The predictive accuracies and construction times for LK2.

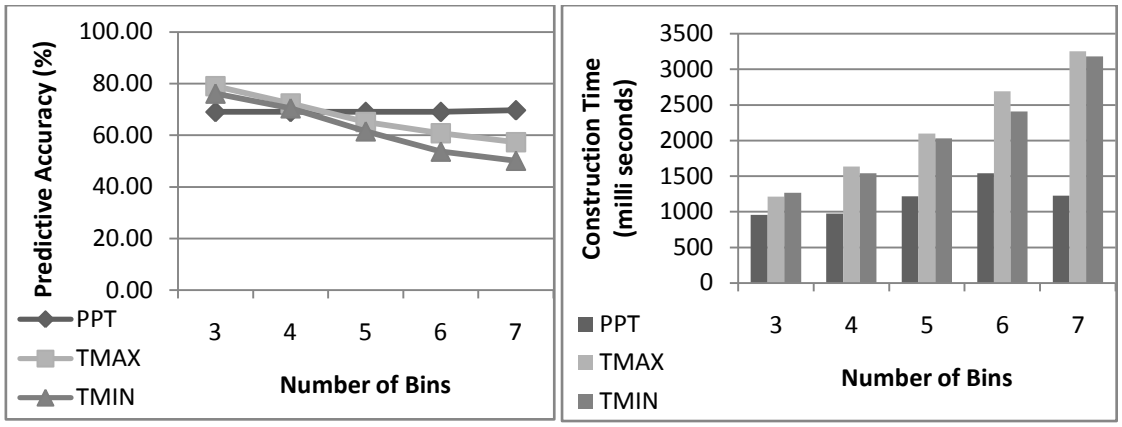


Figure 25: The predictive accuracies and construction times for GTT.

We can see from the figures 22 to 25, that there is a general decrease in the PA of the networks with an increase in the number of bins. In response to this we can also see that there is an increase in the construction times for the networks as the number of bins increases. The BNBC construction times (Figure 22) show two outlying maximum temperature values when using 5 and 6 bins. This was most likely caused by anomalies within the data that were exposed through the particular number of bins used.

The most interesting aspect to note is that all the networks have an almost constant PA for precipitation, despite there being an increase in the number of bins being used. The precipitation accuracies should be the same as those of minimum and maximum temperature, namely showing a decrease in PA. However when we examine the bin ranges for the precipitation data, we find that most of the precipitation data falls within the range of the last bin. The rest of the bins have no observed values which fall into them. The reason for this is that precipitation normally occurs in long periods of little or no rainfall followed by short periods of heavy rainfall. The result of this is that the discretized data only falls into the range covered by the last bin. Therefore increasing the number of bins has little impact on the PA of the networks when dealing with precipitation data.

However the precipitation data does result in an artificially high PA for the networks, as they produce a higher PA when forecasting the absence of rainfall. This can bee seen when comparing the PAs between the different weather variables in experiment 3.

Another interesting observation is between the construction times of K2, LK2 and GTT (Figures 23, 24, and 25). The LK2 and K2 indicate no difference or dependency towards the type of data that they are constructing networks from, while the GTT algorithm takes less time for the networks based on precipitation data. The reason for the K2 and LK2 algorithms remaining independent of the data is due to their method of scoring. Their scoring technique relies on multiple comparisons between the discretized data from different nodes. The number of comparisons is independent on the data being compared, and therefore the construction times using the different weather variables are the same for K2 and LK2.

GTT constructs precipitation networks in less time than its maximum and minimum temperature based networks. The cause behind this might be a dependency of the PC algorithm on the data which reduces the time to perform the conditional independence tests between the nodes. This could also explain the decrease in construction time of the precipitation based networks, when using 7 bins as apposed to 6 bins (Figure 25).

6 Findings and Critical Analysis

6.1 BNBC Review

The objective of the BNBC was to offer a better performing NB network while still offering the same advantages as an average NB network would. Although in all the cases within experiment 3, the BNBC improved on the NB, it took more time than other higher scoring networks did. Therefore the BNBC algorithm is not recommended for future implementation.

6.2 K2 Compared to LK2

For all the experiments LK2 was able to construct networks in less time than the K2 algorithm was able to. Furthermore the resultant PAs of the networks differed by less than 2% when compared to K2. Although the implementation of the K2, and therefore the LK2, were done through the use of a sub-optimal data structure, they both used the same assumptions. Therefore the LK2 algorithm is recommended for implementation in future works as apposed to the K2 algorithm.

6.3 Arc Removal Problem

When we constructed networks with a maximum of 1 parent per node, we found that a number of nodes were in fact never linked to any other nodes. This was not a fault in any of the learning algorithms, but it now does pose a forecasting problem. With an isolated node, no change of evidence for the other nodes will affect it. Therefore the station is essentially only reliant on its past data, and not on any other node. As part of the objective of this section was to forecast the weather for every station, isolated stations were a problem.

One solution could be to link them to their nearest neighbour, but this could undo the benefits gained in PA by them not being linked. An alternate solution might be to cluster them together and link the cluster to another station, but this would remove the ability to forecast for individual stations. Therefore as of yet, no solution by us has been found to solve this problem. The only solution that we have used, is to base the forecasts solely on the past station data while not using any links to other stations.

7 Conclusion

In this section of the report we have shown that there are a number of network construction algorithms available to build BNs with. The Naive Bayes algorithm constructed networks in the least amount of time, but produces the lowest PA. The LK2 algorithm was able to construct networks faster than the K2 in all the experiments performed, and with fewer neighbours it gains PA.

Therefore the best algorithm in terms of PA and construction time is the Naive Bayes algorithm. Although it produces the lowest PA networks, its construction time is less than a second. This makes it the fastest construction algorithm. However if more time is available to construct networks, then there are different algorithms to consider.

The LK2 produces the highest PA for precipitation, while the GTT algorithm scores the highest PA for the minimum and maximum temperatures. Therefore the LK2 algorithm should be used to forecast precipitation, while the GTT should be used to forecast minimum and maximum temperature.

We also found that the number of bins used, effects both the PA and construction times of the networks. With an increase in the number of bins we observed an increase in the network construction times, but we also observed a decrease in the PAs of the networks. Although the option of using fewer bins appears to be the best solution, careful consideration must be taken as with a decrease in the number of bins, there is also a loss of precision in the values that the bins are able to capture. So in light of this, each project should carefully determine what tradeoffs between construction time, PA, and precision are acceptable by their model.

7.1 Compared to Similar Works

The work by R.Cano *et al* [5] examined the ability of their LK2 algorithm to construct networks in less time than the K2 algorithm. In this report we have also been able to show this fact and improve on this work by showing that the LK2 algorithm provides networks with higher PA when using closer neighbours.

R Niculescu *et al* [21] was involved with forecasting ozone levels over Lake Michigan in the United States of America. They also took the approach of taking weather stations as the nodes of a Bayesian Network, but they forecasted data for the gridded region of the lake. However their model was complex and this limited their forecast range to be one day in advance, while we were able to use DBNs to forecast multiple days in advance.

Furthermore, in the paper by Byoungkoo and Joseph [2] they also implemented the LK2 algorithm to the region of the Pacific Northwest of the United States. They used precipitation data, and arrived at the same conclusion that we did, namely that the LK2 algorithm constructs networks in less time than the K2 algorithm does. Furthermore they arranged the node order of their K2 algorithm from Southwest to Northeast in order to capture the wind pattern dependencies of the region. However they were only able to examine the results produced by single time-step processs, while we were again able to use DBNs to forecast multiple days in advance.

7.2 Success of Project

The project was a success as we were able to produce the best static structures for use in the DBN section, namely the LK2 and GTT networks. We were also able to successfully investigate the differences between the K2 and LK2 algorithms, and the effects of varying the number of bins used to discretize the data.

Lastly although the BNBC algorithms are not worth the extra construction time, they were able to determine which stations made the highest PA for each weather variable. Therefore in future weather models use of the calculated stations will reduce the construction time and lead to a higher PA if used.

7.3 Lessons Learnt

As this project was proposed by the team members we gained much experience though out the duration of the project. The author had to understand the needs of the project and acquire the relevant data. There was also the need to divide the work into even parts for each member of the group, while making sure that the work was of the level expected of honours students.

As we had never done research into the various fields prior to the project, we also had to ensure that there was enough flexibility in the work that we had divided. So if a section became difficult to complete in the future, then we would be able to include or expand on an already existing idea to compensate for the difficulty.

The group meetings were another lesson that the author learnt. The meetings were held on average every 3 to 7 days, and were used to generate ideas and work on the interfaces between the different sections. This enabled us to work through any problems or issues early in the project. Furthermore many ideas were developed during the meetings and were later included into the effected sections. The meetings were therefore crucial to the success of the project, and they made the integration of the different sections successful.

If we had to do a similar project in the future we would do a number of tasks differently. Firstly we would plan more time for evaluating the networks and for writing the reports. We found that in some cases the algorithms took days to finish executing, which left us little time to execute complex tests and handle debugging problems. The report writing was often prioritized below implementing algorithms, which resulted in less time to have the report drafts approved. Lastly we would ensure that we acquired the data early in the development of the model, as to relieve the pressure of the consequences if the data was not found.

8 Future Work

There were a number of ideas that were conceptualized during the duration of the project. However we were unable to implement some of them due to software and time constraints. We would recommend to others wanting to replicate our work, to also consider these concepts.

8.1 Decision Graphs and Trees

With the growth of the CPT the number of parents that a node can support is ultimately limited by the hardware specifications. However Chickering and Heckerman [7] proposed the use of decision trees and graphs as a solution, and an alternative to the CPT. The idea of the decision tree is too exploit the same repeating values within the CPT. These repeating values are essentially redundant and therefore occupy space unnecessarily. Decision trees solve this problem by representing the CPT in the more compact form of a tree structure. The use of the graph structures further improves on this by allowing two decision nodes to become the parents of a single common child node.

8.2 Seasonal Variations in Causal Models

During the evaluation phase of the CM, the data was only taken from specified periods that were complete and had the desired number of nodes. However, no dataset was constructed to represent seasonal changes. Seasonal structures could contain information and trends that are local to that season, but are otherwise lost in the larger annual structures.

The structures would have to be based on the first and last six months of the year in order to capture the summer and winter variations. Following their creation, comparisons and contrasts could be found to determine if seasonal forecasting provided higher accuracies, than annual forecasting did. Naturally this could be extended into the DBN and Visualization sections, where the effect of past seasons on future seasons could also be examined.

Though if a seasonal structure is implemented, a new discretization technique would also need to be developed. This is due to the fact that Smile offers no per dataset discretization method, rather it can only discretized data that is within a single list (vector). This means that a new approach will need to be developed to calculate the training and test set periods for the seasonal data.

8.3 Discretization Technique for Precipitation Data

The current discretization technique in this report uses 5 bins to quantise the station data. However this makes the range of each bin the *same*. The problem that arises from this is that when dealing with precipitation data, the observed data often occurs in extremes. The extreme values are the result of long periods of little or no rainfall followed by short periods of heavy rainfall.

The extreme values mean the discretization technique can place all the rainfall values in one bin, while leaving the rest of the bins filled with zeros. This therefore artificially inflates the predictive accuracy of the networks constructed on the precipitation data, as they are more likely to predict correctly when they forecast the absence of rainfall.

In response to this problem [2] proposes that that the ranges of the individual bins should be specified manually in advance. They propose that the bins rages should set according to figure 26.

1-5, 5-15, 15-40, 40-100, and > 100 mm/day

Figure 26: Recommended bin ranges.

By using the bin ranges the finer details of the precipitation patterns will be captured, and the observations will fall into the ranges of *all* the bins. This will prevent only one bin from having the rage which encompasses all the precipitation observations, and will therefore prevent artificially high predictive accuracies for networks based on precipitation data.

9 Glossary of Terms

BN	Bayesian Network
BNBC	Best Naive Bayes Classifier algorithm
CM	Causal Model (section)
CPT	Conditional Probability Table
CS	Department of Computer Science at UCT
DAG	Directed Acyclic Graph
DBN	Dynamic Bayesian Network (Learning section)
EGS	Department of Environmental and Geographical Science at UCT
GTT	Greedy Thick Thinning algorithm
PA	Predictive Accuracy
PPT	Precipitation
TMAX	Maximum Temperature
TMIN	Minimum Temperature
UCT	University of Cape Town
VS	Visualization System (section)

References

[1] Abramson, B., Brown, J., Edwards, W., Murphy, A. and Winkler, R. L. Hailfinder: A Bayesian system for forecasting severe weather. *Int. J. Forecast., 12*, 1 (1996), 57-71.

[2] Byoungkoo, L. and Joseph, J. Learning a Probabilistic model of rainfall using graphical models. School of Computer Science. Carnegie-Mellon University, 2006.

[3] de Campos, L. A scoring function for Learning Bayesian Networks based on Mutual Information and Conditional Independence Tests. Journal of Machine Learning Research. 7:2149-2187, 2006.

[4] de Campos, L. and Castellano, J. Bayesian Network Learning Algorithms Using Structural Restrictions. International Journal of Approximate Reasoning. 45:233-254, 2007.

[5] Cano, R., Sordo, C. and Gutierrez, J. M. Applications of Bayesian Networks in Meteorology. *Studies in Fuzziness And Soft Computing, 146* (2004), 309-328.

[6] Cheng, J., Bell, D. A. and Liu, W. An algorithm for Bayesian belief network construction from data. In Anonymous *Proceedings of AI & STAT'97.* (). , 1997, 83-90.

[7] Chickering, D. Heckerman, D. and Meek, C. A Bayesian Approach to Learning Bayesian Networks with Local Structure. Microsoft Research. Redmond WA, 98052-6399.

[8] Cooper, G. F. and Herskovits, E. A Bayesian Method for the Induction of Probabilistic Networks from data. *Machine Learning, 9*, 4 (1992), 309-347.

[9] Friedman, N., Nachman, I. and Peer, D. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In Anonymous *Proc. UAI.* (). , 1999.

[10] Haversine and Vincenty Distance Formulae [Online] Available At : http://www.movable-type.co.uk/scripts/latlong.html. Accessed: 31 October 2008.

[11] Heckerman, D. A tutorial on learning with Bayesian networks. *Learning in Graphical Models*, (1998), 301–354.

[12] Heckerman, D. Bayesian Networks for Data Mining. *Data Mining and Knowledge Discovery, 1*, 1 (1997), 79-119.

[13] Heckerman, D., Geiger, D. and Chickering, D. M. Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learning, 20*, 3 (1995), 197-243.

[14] Hulst, J. Modelling Physiological Processes with Dynamic Bayesian Networks. *Faculty of Electrical Engineering, Mathematics, and Computer Science, University of Pittsburgh*, 2006.

[15] Ismail. M. An Empirical Investigation of The Impact of Discretization on Common Data Distributions. Dissertation for the Department of Computer Science, RMIT University, Melbourne Australia, 2003.

[16] JavaBayes [Online] Available At: http://www.cs.cmu.edu/~javabayes/Home/ Accessed: 31 October 2008.

[17] Korb, K. and Nicholson, A. Bayesian Artificial Intelligence. Chapman and Hall/CRC. ISBN 1-58488-387-1, 2004.

[18] Lowd, D. and Domingos, P. Naive Bayes models for probability estimation. *Machine Learning-International Workshop Then Conference*, 2005.

[19] Map of South Africa [Online] Available At : http://www.lib.utexas.edu/maps/africa/south_africa_rel_2005.pdf Accessed: 31 October 2008.

[20] Map of Southern Africa [Online] Available At: http://www.canoncollins.org.uk/about/index.php Accessed: 1 November 2008.

[21] McMillan, N., Bortnick, S. M., Irwin, M. E. and Mark Berliner, L. A hierarchical Bayesian model to estimate and forecast ozone through space and time. *Atmos. Environ., 39*, 8 (2005), 1373-1382.

[22] Niculescu, R. S., Mitchell, T. M. and Rao, R. B. Bayesian Network Learning with Parameter Constraints. *The Journal of Machine Learning Research, 7* (2006), 1357-1383.

[23] Ruiz, C. Illustration of the K2 Algorithm for Learning Bayes Net Structures. *Department of Computer Science*, WPI, 2005.

[24] Smile and Genie [Online] Available at: http://genie.sis.pitt.edu/about.html Accessed: 31 October 2008.

[25] Weka [Online] Available at: http://www.cs.waikato.ac.nz/ml/weka/ Accessed: 31 October 2008.

Appendix A

The sample set provided by Ruiz [23], which was used to test our initial implementation of the K2 algorithm (Figure 28), and the node scores (Figure 29) from the original and modified K2 implementations. Figure 27 describes the K2 node scoring function in detail.

$$f(i,\pi_i) = \prod_{j=1}^{|\Phi_i|} \frac{(r_i-1)!}{(N_{ij}+r_i-1)!} \prod_{k=1}^{r_i} \alpha_{ijk}!$$

Figure 27: The K2 node scoring function [23].

The K2 node scoring function in both figure 2 and figure 28 is detailed as follows:

- i The current node that the K2 algorithm is operating on.
- r_i The number of states that i can have.
- π_i The parents of i.
- $|\Phi_k|$ The number of values within the CPT of i.
- α_{ijk} The number of cases in the dataset in which i has its k^{th} value and π_i have their j^{th} value in the CPT.
- N_{ij} $= \sum_{k=1}^{r_i} \alpha_{ijk}$: The sum of the α_{ijk} for each state of i.

Case	x_1	x_2	x_3
1	1	0	0
2	1	1	1
3	0	0	1
4	1	1	1
5	0	0	0
6	0	1	1
7	1	1	1
8	0	0	0
9	1	1	1
10	0	0	0

Figure 28: The sample training set for the K2 algorithm.

Node	Original Score	Modified Score
1	0.00036075	9.81E-32
2	0.00111111	3.64E-24
3	0.00555556	3.74E-14

Figure 29: Node scores from the sample training set.

Appendix B

This appendix lists the original and unedited results that were found during the experiments (Figures 30 to 38). The construction times were recorded in micro seconds and were converted into their plotted graphs by using the "QUOTIENT" function found within Microsoft Excel.

Experiment 1

	1 Parent	2 Parents
K2	66.6148	58
LK2	66.5373	61.597
GTT	68.0746	69.3433

	1 Parent	2 Parents
K2	2192249694	21900338113
LK2	426142844	4211771236
GTT	1605750	2911639

Figure 30: The PA (left) and construction times (right).

Experiment 2

	1 Parent	2 Parents
5	66.791	63.6418
6	66.8806	63.1719
7	66.6866	62.9104
8	66.6119	62.1194
9	66.4478	62.1343
10	66.5373	61.597

	1 Parent	2 Parents
5	231204459	2113481961
6	284798910	2606378077
7	326552431	3117344961
8	371249086	3607388416
9	424833688	4103427657
10	471059151	4211771236

Figure 31: The PA (left) and construction times (right).

Experiment 3

Ave NB	68.9236
BNBC	69.4324
K2	76.6757
LK2	77.2162
GTT	69

Ave NB	14241
BNBC	1957189981
K2	593367580
LK2	67194106
GTT	69619025

Figure 32: The PAs (left) and construction times (right) for the precipitation data.

Ave NB	44.6962
BNBC	50.8919
K2	62.7568
LK2	61.6216
GTT	61.4324

Ave NB	80721
BNBC	3426366725
K2	601397901
LK2	66863780
GTT	1976742

Figure 33: The PAs (left) and construction times (right) for the minimum temperature data.

Ave NB	34.817
BNBC	44.7297
K2	63.6757
LK2	63.3243
GTT	64.973

Ave NB	76949
BNBC	3140221539
K2	593241872
LK2	63320738
GTT	1996237

Figure 34: The PAs (left) and construction times (right) for the maximum temperature data.

Experiment 4

	PPT	TMAX	TMIN
3	69.0541	60.2432	68.6487
4	69.0541	51.7027	56.7297
5	69.027	43.8378	51.7297
6	69.027	38.5946	43.3514
7	69.027	32.6216	37.7297

	PPT	TMAX	TMIN
3	13195	15218	14023
4	12152	16449	15463
5	14870	136599	17780
6	14121	137218	19147
7	13821	79047	80466

Figure 35: The PAs (left) and construction times (right) for the BNBC.

	PPT	TMAX	TMIN
3	76.6486	77.9189	75.7838
4	76.5405	70.4595	67.7027
5	75.6757	64.5946	62.9459
6	75.6757	60.1622	55.1892
7	73.7568	55.3514	52.2432

	PPT	TMAX	TMIN
3	269436966	258081043	263122247
4	468184533	456971775	463733183
5	734985594	712235998	735388431
6	1059075687	1045242037	1057869257
7	1424653278	1406291620	1433426008

Figure 36: The PAs (left) and construction times (right) for K2.

	PPT	TMAX	TMIN
3	77.2162	77.2432	76.1892
4	77.0541	70.1351	68.2973
5	76	64.1892	61.5676
6	75.5676	59.5405	55.5676
7	74.6487	54.5946	52.2703

	PPT	TMAX	TMIN
3	34303839	33540091	33807415
4	53509028	52986920	53577981
5	79165700	78176965	79782097
6	110775086	110859473	111262558
7	147375946	148740813	146215500

Figure 37: The PAs (left) and construction times (right) for LK2.

	PPT	TMAX	TMIN
3	69	79	76.1081
4	69	72.2973	70.4054
5	69	65.1081	61.4595
6	69	60.7838	53.5946
7	69.6487	57.3243	50.0541

	PPT	TMAX	TMIN
3	956347	1212305.00	1266625
4	977854	1635437.00	1543225
5	1216042	2099798.00	2032776
6	1542350	2691197.00	2409099
7	1227425	3255403.00	3181700

Figure 38: The PAs (left) and construction times (right) for GTT.