Spring 2014

Comparative Analysis of Naive Bayes and Tree Augmented Naive Bayes Models

Harini Padmanaban
San Jose State University

Recommended Citation

Padmanaban, Harini, "Comparative Analysis of Naive Bayes and Tree Augmented Naive Bayes Models" (2014). *Master's Projects*. 356.
DOI: https://doi.org/10.31979/etd.n7jg-e3uh
https://scholarworks.sjsu.edu/etd_projects/356

Comparative Analysis of Naive Bayes and Tree Augmented Naive Bayes Models

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements of the Degree

Master of Science

by

Harini Padmanaban

May 2014

The Undersigned Writing Project Committee Approves the Writing Project Titled

Comparative Analysis of Naive Bayes and Tree Augmented Naive Bayes Models

by

Harini Padmanaban

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Sami Khuri, Department of Computer Science

Dr. Chris Pollett, Department of Computer Science

Dr. Chris Tseng, Department of Computer Science

ABSTRACT

Comparative Analysis of Naive Bayes and Tree Augmented Naive Bayes Models

by Harini Padmanaban

Naive Bayes and Tree Augmented Naive Bayes (TAN) are probabilistic graphical models used for modeling huge datasets involving lots of uncertainties among its various interdependent feature sets. Some of the most common applications of these models are image segmentation, medical diagnosis and various other data clustering and data classification applications. A classification problem deals with identifying to which category a particular instance belongs to, based on previous knowledge acquired by analysis of various such instances. The instances are described using a set of variables called attributes or features. A Naive Bayes model assumes that all the attributes of an instance are independent of each other given the class of that instance. This is a very simple representation of the system, but the independence assumptions made in this model are incorrect and unrealistic. The TAN model improves on the Naive Bayes model by adding one more level of interaction among attributes of the system. In the TAN model, every attribute is dependent on its class and one other attribute from the feature set. Since this model incorporates the dependencies among the attributes, it is more realistic than a Naive Bayes model. This project analyzes the performance of these two models on various datasets. The TAN model gives better performance results if there are correlations between the attributes but the performance is almost the same as that of Naive Bayes model, if there are not enough correlations between the attributes of the system.

Acknowledgements

First and foremost, I would like to express my deepest gratitude and appreciation to my project advisor, Dr. Sami Khuri, for his excellent guidance and encouragement throughout the project work. I would also like to thank my committee members Dr. Chris Pollett and Dr. Chris Tseng for their time and support.

A very special thanks to Natalia Khuri for her valuable inputs and support throughout the project work.

Table Of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Probabilistic Graphical Model (PGM) is a framework that helps in modeling systems that involve features with lots of uncertainties. This model helps predict the future results based on information obtained from past observations through probabilistic methods. Some of the main applications of PGMs are medical diagnosis, image segmentation and many other classification and clustering applications [2]. For example, in medical diagnosis, a doctor needs to analyze various aspects related to a person, such as the person's body temperature, blood pressure, sugar levels, height, weight, allergies, pain and various other factors in order to decide whether the person has a particular disease [2]. Also, the doctor cannot be very certain about all the observations as these are dependent on symptoms that the patient describes. So the data that the doctor has, involve lots of uncertainty and the doctor needs to decide if a person has a disease or not based on these symptoms and analyzing the probabilities that they would lead to a disease. In case of image segmentation, we need to determine whether a portion of an image belongs to a person, house, road etc, based on the given huge set of pixels. All these applications have uncertainties in the form of variables involved in them such as fever level, pain, head ache, etc or pixels of various parts of image [2]. These uncertainties can be modeled in the form of a graph, which depicts the interdependencies among these variables. Such a model, based on the probabilities of the observations, can then be used to find the state of a query variable given the state of other variables in the dataset.

The overview of the chapters is as follows. Chapter 2 discusses some of the probability theory concepts that are required for understanding PGM. Chapter 3 explains the concepts of Bayesian networks. Chapters 4 and 5 discuss the structure and implementations of two special types of Bayesian networks viz., Naive Bayes model and Tree Augmented Naive Bayes model respectively. Chapter 6 presents the results of both the Naive Bayes and TAN models on various datasets and analyzes the results. Finally, based on the comparative analysis of the results of TAN and NB models, we conclude that, the TAN model performs better if there are correlations between the variables in a system.

CHAPTER 2

Probabilistic Graphical Models

2.1 Introduction

Probabilistic graphical model is basically a graph that models the interdependencies among the variables in a dataset and helps in determining probabilistically the state of a query variable given the state of one or more variables in the data set [1] [2]. This can be done by representing the information at hand in the form of a graphical model and execute various reasoning algorithms on that model, in order to arrive at sensible predictions about the state of a variable in the system. A graphical model is used because it is a more common form of data structure that can be used to represent data and information relating to various fields and facilitates sharing of algorithms and techniques among different fields [2]. Probabilistic models are used because probability theory is the efficient way for handling uncertainty. A probabilistic model is more liberating as it helps in accommodating exceptional cases that are not currently observed in the system but also cannot be completely neglected. This is achieved by assigning small probabilities to such states. It also helps in learning from the observations. Probabilistic graphical model has three critical components viz., Representation, Inference, and Learning [2]**.** Representation refers to the modeling of the complex system that involves uncertainties close to the reality. Inference refers to the ability to query the model and get the relevant information. Learning is the process of obtaining this distribution based on the data at hand and some expert knowledge [2].

A probabilistic model is built based on the interaction between the variables that are involved in the system. These variables are called as random variables. For example, in a medical diagnostic system, height, weight, pain, fever, diseases, test results etc., form the random variables[2]. The system is built capturing the interaction between these variables and how they influence each other. This model can then predict the state of a variable in the system, given the states of some of the other variables in the system. It also models the independencies among those variables. In what follows, we introduce concepts, that are essential for understanding probabilistic graphical models.

A random variable is a variable whose value is subject to variations. It can either be discrete or continuous. A discrete random variable can take a value form a set of possible different values where each of the values is associated with a probability [13]. Continuous random variables can take from an infinite number of possible values associated with probability density function [13]. In a medical diagnosis system, a symptom such as pain can take values such as high, low, no pain and another variable like temperature can take values from a range of continuous values [1] [2]. The selection of these variables and their values is a very important factor that has a great impact on the queries that a model could answer.

2.2 Marginal and Joint Distribution

A random variable X can take a set of possible states or values. A probability distribution over the states that can be taken by a random variable X is called the marginal distribution over the random variable X. Let us consider an example that models the details about the student population. This example is taken from [2] and the tables relating to these examples are reproduced from [2]. The random variables involved in the student model example are difficulty, intelligence, grade, SAT and letter.

- *Difficulty* refers to the difficulty of the course that the student takes and it can take two values {hard- d^1, easy-d^0}
- *Intelligence* refers to the intelligence level of the student and it takes the values {low i^0, High i^1}
- *Grade* refers to the grade scored by the student and it takes the values {A-g^1, B- g^2, C-g^3}
- *SAT* refers to the SAT score of the student and it takes values {Low s^0, High s^1}
- *Letter* refers to the quality letter of recommendation that the student received {good, bad}

A marginal distribution of the student intelligence and course difficulty is described in Table 1

Table 1: Marginal Probability distribution of difficulty and intelligence in student model [2].

D	**P(D)**
d^0	0.6
d^1	0.4

(A)

I	**P(I)**
i^0	0.7
i^1	0.3

(B)

One very important property in probability theory is that the sum of the probabilities of the states that a random variable can take must equal 1. For example, in Table 1 the sum of the probabilities of difficulty = high and difficulty = low adds up to one.

$$\sum_{i=1}^{n} P(x_i) = 1$$

Sometimes queries on the states of more than one random variable need to be answered. An event of occurrence of a combination of states of different random variables may be of interest[2]. For example, events where the difficulty = high and grade = A, may need to be captured in order to determine the performance of students. In such cases, a joint probability distribution over both the random variables needs to be considered [2].

A joint distribution over a set of random variables (X_1,.., X_n) is given by P (X_1.... X_n). This distribution assigns probability for all the states that are described by these random variables [2]. A joint distribution over a set of random variables must also be consistent with the marginal distribution of those variables[2]. This is called the rule of marginalization and is defined as follows:

$$P(x) = \sum_{y} P\ (x, y)$$

For example, a joint distribution over the random variables student grade and intelligence, as given in [2] is shown in Table 2.

Table 2: Joint Probability distribution of difficulty and intelligence in student model [2].

		Intelligence		Sum
		Low	**High**	
Grade	**A**	0.07	0.18	0.25
	B	0.28	0.09	0.37
	C	0.35	0.03	0.38
	Sum	0.7	0.3	1

In Table 2, if the probabilities of the random variable intelligence is summed across all the events that it describes viz., high and low then it results in the marginal distribution of the random variable grade. Similarly the summation of the probabilities of all the events described by grade viz., A, B and C results in the marginal distribution of the random variable intelligence. In a joint probability distribution, the sum of all the entries in the distribution must add up to 1. In Table 2, the sum of all the six entries adds up to 1.

2.3 Conditional Probability

Conditional probability is a notion in probability theory that explains how the occurrence of an event can affect the probability of occurrence of another event. That is, given that an event A has occurred, what impact does it have on the occurrence of another event B. For example, in a medical diagnosis system, given that a person has muscle pain what is the probability that he has the flu[2]. The conditional probability of B given A is defined as:

$$P(B|A) = P(A,B) / P(A)$$

where P(A,B) is the joint probability of events A and B and P(A) refers to the marginal probability of event A. Another important concept in probability theory based on the conditional probability is the chain rule. The chain rule helps in determining the probability of combined occurrences of various states of the random variables involved in the system. It is calculated as a product of probability of occurrence of first event, the probability of occurrence of the next event, given the previous events and so on [2]. Given a set of random variables X_1 to X_n, the chain rule for finding the joint probability distribution is given by [2],

$$P(X_1, X_2, X_3,..., X_n) = P(X_1) \cdot P(X_2| X_1) \cdot P(X_3| X_1, X_2) \cdot ... \cdot P(X_n| X_1... X_{n-1}).$$

Conditional probability also leads to another important concept in the probability theory called Bayes' rule [2], which is defined as:

$$P(A|B) = P(B|A) \cdot P(A) /P(B)$$

Conditional distribution of random variable can be obtained by reducing their joint distribution. This is done by conditioning on one a particular value that the random variables can take and renormalizing the resulting distribution so that the probability values add up to 1. For example, let us consider the joint distribution of the random variables grade (G), intelligence (I) and difficulty (D) of the student model, as given in Table 3.

Table 3: Joint probability distribution of random variables intelligence, difficulty and grade [2].

I	**D**	**G**	**P(I,D,G)**
i^0	d^0	g^1	0.126
i^0	d^0	g^2	0.168
i^0	d^0	g^3	0.126
i^0	d^1	g^1	0.014
i^0	d^1	g^2	0.07
i^0	d^1	g^3	0.196
i^1	d^0	g^1	0.162
i^1	d^0	g^2	0.0144
i^1	d^0	g^3	0.0036
i^1	d^1	g^1	0.06
i^1	d^1	g^2	0.036
i^1	d^1	g^3	0.024

From Table 3, the conditional probability distribution of I and D conditioned on G=g^1 is obtained by removing all the entries in Table 3 for which the value of the random variable is not g^1. We then renormalize the resulting entries from the table so that they sum up to one. Table 4 explains this procedure.

Table 4: Conditioning and Renormalization of a joint distribution[2]

I	**D**	**G**	**P(I,D,g^1)**
i^0	d^0	g^1	0.126
i^0	d^1	g^1	0.014
i^1	d^0	g^1	0.162
i^1	d^1	g^1	0.06
Sum			**0.362**

(A)

I	**D**	**P(I,D | g^1)**
i^0	d^0	0.3481
i^0	d^1	0.0387
i^1	d^0	0.4475
i^1	d^1	0.1657

(B)

In Table 4 (A), the sum of all the entries in the last column do not add up not one. So it is renormalized by summing up all the entries in the last column and dividing each of the entry with the sum. The resulting Table 4 (B), is a conditional probability distribution of the intelligence and difficulty given grade = A.

The next chapter discusses Bayesian Networks, which is a type of probabilistic graphical model, that encodes the interactions between the variables involved in a system using directed acyclic graphs [2].

CHAPTER 3

Bayesian Networks

3.1 Motivation

A probabilistic graphical model for an application is mainly based on the random variables involved in the system and the interaction between them. The joint distribution of random variables in a system gives the probability of occurrence of all possible combinations of all possible values, that the random variables in the system can take. From this it can be concluded that, given the joint distribution of the random variables involved in a system, any queries regarding the state of any random variable can be answered. But this is not easy in reality. For example, let us consider a system which has n random variables and all those variables can take only binary values. Thus, the joint distribution of such a system would have 2^n entries. In reality, the value of 'n' will be very large, and most of the random variables would take more than two values. Hence, constructing such a huge joint distribution is inefficient for probabilistic inference, as only a smaller part of the huge distribution would only be used frequently and it also inefficient in terms of space (memory).

To tackle this problem, the interdependencies between the random variables should be studied and independent random variables need to be identified. If two random variables are independent of each other, the joint distribution of those two random variables would not provide any additional information. Thus, it helps in the reducing the complexity of the model as these interactions can be eliminated from the joint distribution. A Bayesian network can be viewed as a "data structure that provides the skeleton for representing a joint distribution compactly in a factorized way and a compact representation for a set of conditional independence assumptions about a distribution" [2]

One of the best ways for modeling the interdependencies between the random variables in a system is by using graphs. The nodes of the graphs represent the random variables and the edges represent the relationship between the connected nodes. There are 2 types of graphs. A graph can either be directed or undirected. Bayesian Networks is a commonly used graphical model that represents the complete probability distribution and the independencies involved in the system in the form of directed graphs. Markov Networks is another commonly used graphical model that uses undirected graphs. The next chapters will completely focus on Bayesian Networks, its types and properties and its representation as a probabilistic graphical model.

3.2 Conditional Independence

One of the key aspects in representing a system as a probabilistic graphical model is the identification of independent random variables in the system. Two random variables X and Y are independent if for any state x of X and any state y of Y, the joint probability distribution of X=x and Y=y is the product of marginal probability of X=x and marginal probability of Y=y. This is given by:

$$P(x, y) = P(x) \cdot P(y)$$

Also, the following statements hold true:

The conditional probability of X conditioned on a state y of Y does not depend on y.

$$P(x \mid y) = P(x)$$

Similarly, the conditional probability of Y conditioned on a state x of X is independent of x.

$$P(y \mid x) = P(y)$$

Consider the example for the random variables intelligence and difficulty in the student model.

Table 5: Joint distribution of intelligence and difficulty in student model [2].

I	**D**	**P(I,D)**
i^0	d^0	0.42
i^0	d^1	0.28
i^1	d^0	0.18
i^1	d^1	0.12

Table 6: Marginal distributions of intelligence and difficulty in student model [2]

I	P(I)
i^0	0.7
i^1	0.3

(A)

D	P(D)
d^0	0.6
d^1	0.4

(B)

Table 5 gives the joint distribution of random variables I and D and Table 6 (A) and (B) give the marginal distribution of random variables intelligence and difficulty. It can be observed that each of the entries in Table 5 is the product of the probability of corresponding states of random variables given in Table 6. Since the joint distribution is a product of the marginal distribution, it means that the random variables are independent of each other.

The previous example explains the independence between two random variables. Conditioning on a particular values of a random variable, can make the rest of the random variables in the system both gain and lose independence. Let us consider each of these cases in detail, using the student model example.

Table 7: Joint distribution of intelligence, SAT and grade [2].

I	S	G	P(I,D,G)
i^0	s^0	g^1	0.133
i^0	s^0	g^2	0.2261
i^0	s^0	g^3	0.3059
i^0	s^1	g^1	0.007
i^0	s^1	g^2	0.0119
i^0	s^1	g^3	0.0161
i^1	s^0	g^1	0.0444
i^1	s^0	g^2	0.01008
i^1	s^0	g^3	0.00552
i^1	s^1	g^1	0.1776
i^1	s^1	g^2	0.04032
i^1	s^1	g^3	0.02208

From the joint distribution given in Table 7 we can see that, all the three random variables are interrelated. Now, conditioning the entries in Table 7 on i^0, we get Table 8 (A).

Table 8: Conditional distribution of intelligence, SAT and grade [2].

S	**G**	**$P(S,G\| i^0)$**
s^0	g^1	0.19
s^0	g^2	0.323
s^0	g^3	0.437
s^1	g^1	0.01
s^1	g^2	0.017
s^1	g^3	0.023

(A)

S	**$P(S\|i^0)$**
s^0	0.95
s^1	0.05

(B)

G	**$P(G\|i^0)$**
g^1	0.2
g^2	0.34
g^3	0.46

(C)

It can be observed that the entries in Table 8 (A) can be obtained by the multiplication of the conditional probabilities of SAT given $i^0$and grade given i^0 given in Table 8 (B) & (C). This implies that, given information about the intelligence, the random variables SAT and grade are no more dependent on each other.

$$P(S,G|i^0) = P(S|i^0) \cdot P(G|i^0)$$

Intuitively, if no information is given about the intelligence of the student, then a SAT score may help predict that the student would score a good grade in his course. But given the student is intelligent, that in itself helps predict that the student would get good grades in his course, and the SAT score does not provide additional information. Thus, grade and SAT are independent given class [2]. Thus, conditioning on a random variable has resulted in *gaining independence* between two other random variables. A graphical depiction of this is given in Figure 1.

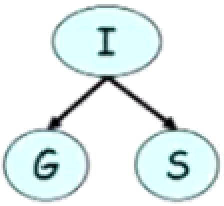


Figure 1: Conditioning can lose independence [2].

In Figure 1, I is the cause and G and S are the effects. Evidence (information) about the parent I affects the dependency between the children G and S.

Conditioning can also lose independence. Consider the joint distribution between the random variable intelligence(I) and difficulty(D) of the student model, given in Table 5. Table 6 gives the marginal distribution of intelligence(I) and difficulty(D). From these two tables it can be seen that random variables I and D are independent of each other.

$$P(I,D) = P(I) \cdot P(D)$$

Now, conditioning the distributions on G = g^1, results in the distribution in the Table 9.

Table 9: Conditional distribution of intelligence and difficulty [2].

I	**D**	**P(I,D | g^1)**
i^0	d^0	0.348066
i^0	d^1	0.038674
i^1	d^0	0.447513
i^1	d^1	0.165745

(A)

D	**P(D | g^1)**
d^0	0.79558
d^1	0.204419

(B)

I	**P(I | g^1)**
i^0	0.38674
i^1	0.61325

(C)

From Table 9 (A), (B) and (C), it can be seen that P(I,D |G) is not (P(I|G) · P(D|G)). Hence, P(I|G) and P(D|G) are not independent anymore. Thus, conditioning on the random variable G has lead to the loss of independence between random variables I and D.

Intuitively, without having any information about the grade, the difficulty of the grade has nothing to do with the intelligence. Hence, they are independent. Now, given that a student has scored A in a difficult course, it makes sense that the student can be intelligent. Hence, given some evidence about grade the random variables difficulty (D) and intelligence (I) depend on each other. A graphical depiction of this is given in Figure 2 [2].

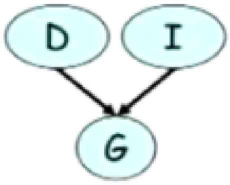


Figure 2: losing independence between variables by conditioning [2].

In Figure 2, D and I form the cause and G is the effect. Given some evidence (information) about the child G affects the dependency between the parents D and I. Thus, the relationship between the random variables can be depicted in the form of a parent child relationship in a graph. Also, the concept of loosing and gaining independence between the random variables due to conditioning plays a key role in determining the flow of influence between the random variables in a graphical model.

An important concept in Bayesian networks is the depiction of the probability distribution in the form of a graph. All the independence relations that are true in the distribution must also be true in the graph. Formalizing this concept we have the following theorem.

Def 1: Let G be a graph over the random variables $X_1,....,X_n$. The distribution P Factorizes over G if

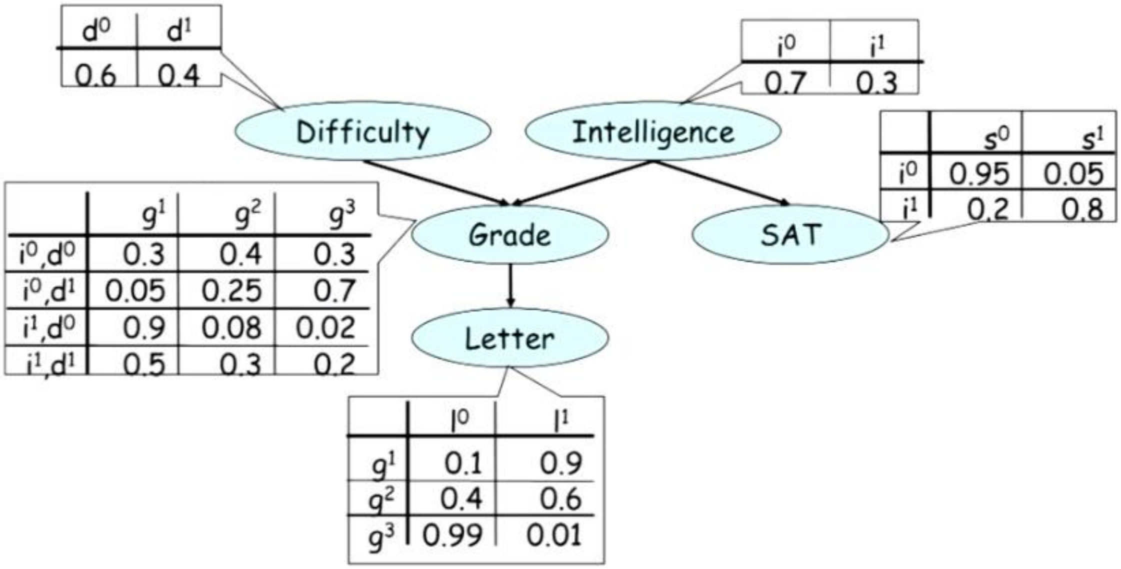
$$P(X_1, \dots, X_n) = \prod_i P(X_i | Par_G\ (X_i))$$

where Par_G refers to parent of G [2].

That is, a distribution P factorizes over a graph G if the joint distribution of the random variables in P equals the product of conditional probability distribution of the random variables given its parent.

3.4 Bayesian Networks Defined

Def 2: A Bayesian network is defined as a directed acyclic graph(DAG) whose nodes represent the random variables $X_i,...,X_n$ and for each X_i there is a conditional probability distribution(CPD) $P(X_i|Par_G(X_i))$ [2].
It is a directed acyclic graph (DAG) in which nodes are represented by random variables while the edges indicate the influence of one node on another[3]. Figure 3 is an example of Bayesian network for the student model [2].



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^1,d^0	0.9	0.08	0.02
i^1,d^1	0.5	0.3	0.2

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Figure 3: Bayesian network for the student model [2]

The chain rule for the joint distribution of random variables P(D,I,G,S,L) is given as [1]:

$$P(D,I,G,S,L) = P(I)\ P(D|I)\ P(G|I,D)\ P(L|G,I,D)\ P(S|I,D,G,L)$$

This can be reduced by the chain rule for the Bayesian networks to [1]:

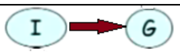
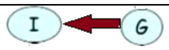
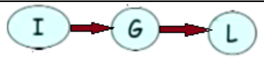
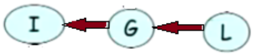
$$P(D,I,G,S,L) = P(D)\ P(I)\ P(G|I,D)\ P(S|I)\ P(L|G)$$

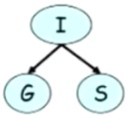
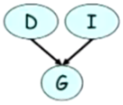
3.4 Flow of Influence in Bayesian Networks

The flow of influence in Bayesian networks is a notion that explains when can one random variable can influence another in the network[2]. Two random variables that are directly connected in a graph influence each. Another notion that helps in understanding flow of influence is the active trail. A trail is a path in the network between two random variable. The path between two random variable in a Bayesian network is said to be active, only if the two random variables in consideration can influence each other. An active path is an active trail. In case of direct connections between nodes (representing other random variables), the trail is always active. If the random variables in consideration are separated by intermediate nodes (representing other random variables) in the graph, then the existence of an active trail between the two random variables depends on the direction of the edges in the path and also on the availability of evidence about intermediate nodes. Table 10 explains all possibilities of an active trail in a Bayesian network. The contents of Table 10 was obtained from [2].

Note : In Table 10, X → Y implies a trail between X and Y. In Table 10, X, Y and Z represent three distinct sets of random variables rather than a single random variable.

Table 10: Active Trail in Bayesian Networks **[2].**

Trail Type	**Example**	**Explanation**	**Z Not Known**	**Z Known**
Direct Trail $X \to Y$		**(Y) :** $P(G^1) = 0.362$ [probability of student getting good grade G^1.] **(X => Y) :** $P(G^1\|I^0) = 0.2$ [Probability of student getting a good grade(G^1) conditioned on less/non intelligence I^0]	Active	Active
Direct Trail $Y \to X$		**(X) :** $P(I^1) = 0.3$ [Probability of student being intelligent] **(Y=> X):** $P(I^1\|G^1) = 0.61325$ [Probability of student being intelligent given that he scored a good grade G^1]	Active	Active
Causal Trail $X \to Z \to Y$		**(Y):** $P(L^1) = 0.50236$ [Probability of getting a good recommendation letter] **(X =>Y):** $P(L^1\|I^0) = 0.3886$ [Probability of student getting a good letter L^1 given that he is less intelligent I^0]	Active	Not active
Evidential Trail $X \leftarrow Z \leftarrow Y$		**(X) :** $P(I^0) = 0.7$ [Probability of a student being less intelligent] **(Y => X):** $P(I^0\|L^1) = 0.5418$ [Probability of the student being less intelligent I^0 given the evidence that student has got a good letter L^1	Active	Not active

Common Cause $X \leftarrow Z \rightarrow Y$:		**(Y) :** P(S^1) = 0.275 [Probability of student getting good score in SAT] **(X => Y)**: P(S^1|G^1) = 0.5099 [Probability of student scoring well S^1 given that he has got good grade G^1 in his course]	Active	Not active
Common Effect $X \rightarrow Z \leftarrow Y$:		**(Y) :** P(I^1)= 0.3 [Probability of student being intelligent] **(Z=>Y) :** P(I^1|G^3) = 0.07894 [Probability of the student being intelligent given a low grade G^3] **(X,Z = > Y) :** P(I^1|G^3,D^1) = 0.10909 [Probability of the student being intelligent given low grade G^3 in a difficult course D^1] Note: Without information about Z (grade) X (difficulty) cannot influence (intelligence)	Not active	Active

The first column of Table 10 gives the type of active trail. The second column gives an example for the type of trail from the student model followed by its explanation in the third column. The last two columns specify whether the intermediate nodes in the trail are observed or not.

By studying the active trails and the independence properties, it can be concluded that "if P factorizes over a graph G, then we can read those independencies from the structure of the graph"[1]. More formally, it is stated as, "If P factorizes over graph G, then in P, any variable is independent of its non descendants given its parents " [1].

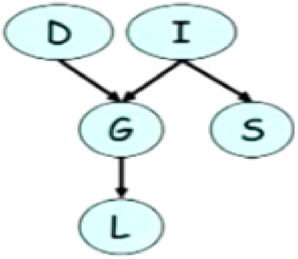


Figure 4: Bayesian network depicting parent child relationship [2].

From Figure 4, it can be observed that,

- D and I are non descendants of each other . Hence, they are independent of each other.
- D and G are non descendants of S and I is the parent of S. Hence, {D, G} and S are independent given I
- D, I and S are non descendants of L and G is the parent of L. Hence, {D, I, S} and L are independent given evidence about G

From the above observations, it can be concluded that, the structure of a graph of Bayesian network can be used to infer the independencies in the distribution also to reduce and represent the joint distribution in a compact form.

3.5 Complexity of Bayesian Network

If we do not have any knowledge about the structure of the Bayesian network, and we are to determine the correlations among various random variables in the network from a given dataset, the complexity of such a process is exponential. Since there is no knowledge about the dependencies among variables, the joint distribution cannot be reduced. So in order to build a classification model, we need to determine the joint distribution of all the variables conditioned on class. The formula is given by,

$$P(C \mid x_1 \dots x_n) = \frac{P(C) \cdot P(x_1 \dots x_n \mid C)}{P(x_1 \dots x_n)}$$

The denominator can be ignored as it is the same for all labels of the Class. Hence, we can classify by finding the maximum values of the numerator among all the class labels.

Thus, the value $P(x_1 \dots x_n \mid C)$ is calculated for all possible values that the random variables can take for each of the class label. These values need to be computed and stored before calculating $P(C \mid x_1 \dots x_n)$.

The complexity for calculating $P(x_1 \dots x_n \mid C)$, is discussed in what follows.
Let C be the total number of class labels in the dataset.
Let N be the total number of attributes in the dataset.
Let $\{S_1,...,S_N\}$ be the total number of values that each attribute can take.
Let S_{max} be the max $\{ S_1,...,S_N \}$.
Let R be the total number of training instances.
Now, we need to calculate probabilities for all possible combinations $(x_1 \dots x_n)$ for all the attributes conditioned on class. For example, if attribute A_1 can take 3 values and attribute A_2 can take 4 values, then we have 12 combinations. Thus, the total number of values that needs to be calculated is (12 x number of class labels).
Assuming that, the count of all the combinations can be done by scanning once through all the rows in the dataset, the total number of probability values needed to be calculated is given by

$$\begin{aligned} \textit{Total computations } &= (\mathrm{S_1} \cdot \mathrm{S_2} \cdot \dots \cdot \mathrm{S_n}) \cdot \mathrm{C} \\ &\leq \ S_{max}^{N} \cdot C. \end{aligned}$$

Thus, the complexity of calculating the joint probabilities of all the attributes is exponential. To reduce the complexity, we need to impose restrictions on the level of interaction between variables while modeling the Bayesian network for a system.

In the next chapters, we discuss reducing the computational complexity of Bayesian networks by imposing restrictions on the level of interaction between attributes. We also discuss the impact of such restrictions on the performance of the models.

CHAPTER 4

Naive Bayes Model

4.1 The Model

The Naive Bayes model is a special form of Bayesian network. This model is mainly used for classification problems. The important feature of Naive Bayes model is that, it has very strong independence assumptions [2]. The structure of a Naive Bayes model is shown in Figure 5.

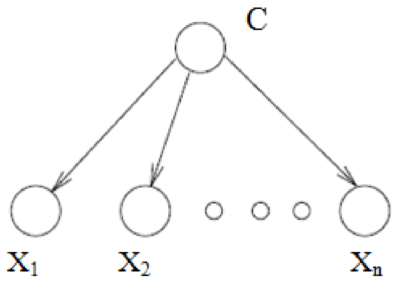


Figure 5: Naive Bayes Model [5]

In Figure 5, C represents a class label to which each instance needs to be classified. It can hold a finite set of class labels $\{C_1,...C_n\}$ and $X_{1,}...X_n$ are the random variables involved in the system. The independence assumption in this model is that given the class variable, all the random variables are independent of each other. But this assumption is rarely true in reality. In any real time system, the variables involved in the system are related to each other. For example, consider a model which decides whether or not to play tennis based on certain conditions of the day[6]. Let the variables involved in the system be, outlook, humidity, temperature, wind and location. The class variable can take 2 values {yes, no}, which predicts whether the weather is suitable to play tennis or not. In this example, if the outlook is sunny, then it is more probable that the temperature is high, and if the humidity is high, it is more probable that the temperature is high. As all these variables are related to each other, assuming independence between these variables is incorrect. But even with such assumptions, this model has performed better for various practical problems. From Figure 5, it can be seen that the joint distribution of the random variables is reduced to,

$$P(X_1, \dots, X_n) = P(C)\prod_i P(X_i|C)$$

In this model, the conditional probabilities of all the feature set or variables given the class label is calculated during the training. During the testing stage, the posterior probability for each of the class labels is calculated as a product of the prior probability and the conditional probabilities of all the variables. The class label with the maximum posterior probability value is assigned for that instance[5]. From the Bayesian rule we have,

$$\mathrm{P}(\mathrm{C} \mid \mathrm{X}_1, \dots, \mathrm{X}_\mathrm{n}) = \mathrm{P}(\mathrm{X}_1, \dots, \mathrm{X}_\mathrm{n} \mid \mathrm{C}) \cdot \mathrm{P}(\mathrm{C}) \,/\, \mathrm{P}(\mathrm{X}_1, \dots, \mathrm{X}_\mathrm{n}) \,.$$

The denominator $\mathrm{P}(\mathrm{X}_1, \dots, \mathrm{X}_\mathrm{n})$ in the above equation, is the same for all the class labels for a given test instance. As the denominator does not play a role in maximizing $\mathrm{P}(\mathrm{C} \mid \mathrm{X}_1, \dots, \mathrm{X}_\mathrm{n})$ over C, it can be ignored and only the numerator can be considered in order to classify the instances.

4.2 Example for NB Model

Let us consider a text classification problem for the Naive Bayes model. The problem is to classify each of the document to the category label. Formalizing the problem statement, we have,

"Given collection of documents $D_{1}, \dots D_{n}$ Belonging to categories $C_{1}, \dots C_{k}$, a vocabulary V is defined containing the possible terms that can occur in the documents. Now given a new document D, find to which category the document belongs to "[1][8].

The Naive Bayes model for the system has one variable for every word in the dictionary, and it can take two values viz., { 0 ,1}. A value of 0 represents that the word is absent in the document and a value of 1 represents that the word is present in the document [1]. The Naive assumption in this model is that, "the event of one word occurring in the document is independent of the occurrence of another word in the document given the class label" [1].

This model calculates the conditional probability of the terms in the dictionary, given the class label. The formula for calculating the conditional probability of the terms, as defined in [1] [7], is given by:

Nr = Number of documents in class C in which the word W_i occurs.

Dr = Total number of docs in class C.

Pr(W_i|C) = Nr/Dr

The calculation of the prior probability as given in [7] is given by:

$P(C_i)$ = Number of documents with class label C_i / Total number of documents in the training set.

The conditional probability for each document given the class label is calculated using the following formula as given in [7][8].

$$P(C|\ D) \ = \left(P(C) \cdot \prod_{i} P(Wi\ |C) \right) / \ P(D)$$

Since the denominator P(D) is the same for all the class labels, it can be ignored and the document is classified only using the numerator. The class label with the maximum value is assigned to that document.

Table 11: Simple Text classification corpus example [7]

	DOC	**Contents**	**Class**
Training	D_1	Dog Cat Dog Sell	Pets
	D_2	Dog Cat Duck	Pets
	D_3	Dog Parrot Duck	Pets
	D_4	Buy Sell Sell Dog	Finance
	D_5	Buy Sell Buy Sell	Finance
Test	D_6	Dog Dog Dog Sell	?
	D_7	Dog Cat Buy Dog Dog	?

Table 12: Conditional probability distribution of terms given the class label [7]

P(Wi| C)	**Prob**
Pr(Dog|P)	3/3
Pr(Cat|P)	2/3
Pr(duck|P)	2/3
Pr(Parrot|P)	1/3
Pr(Sell|P)	1/3

(A)

P(Wi| C)	**Prob**
Pr(Dog|F)	1/2
Pr(Buy|F)	2/2
Pr(Sell|F)	2/2

(B)

Table 11, reproduced and modified from [7] is a simple example for text classification model. Table 12 (A) & (B) give the conditional probabilities of the terms in the documents for each of the class label. For example, from Table 11, the prior probabilities of the two classes can be computed as follows:

P(Pets) = 3/5 and P(Finance) = 2/5

Also, according to Bayes rule: P(C | D) = P(D | C) · Pr(C)/ P(D), where C refers to the class label and D refers to the document instance. Ignoring the denominator as it is the same across all class labels, we have,

P(Pets | d6) = 1 · 1 · 1 · (1/3) · (3/5) = 0.2

P(Finance | d6) = (1/2) · (1/2) ·(1/2) · 1 · (2/5) = 0.05

Class(d6) is max { P(Pets|d6) , P(Finance|d6) }. Hence, the document d6 is classified as a pets document by a Naive Bayes model.

Laplace Correction

In the text classification example, consider a situation where the test document is D_7 given in Table 11. In that case, P(Pets|d7)=0 and P(Finance|d7)=0.

This is because P(Buy|P) = 0 and P(Cat|F) = 0. The word buy never occurred in the document pets and the word cat never occurred in the document finance during training. Thus, it results in the total probability of the class, given the document, to be zero.

Thus, it can be observed that if a random variable X_i that has not occurred during the training happens to show up during testing, then P(X_i| C) will be estimated to be zero. This will in turn lead to the final class probability, given the test sample, to become zero, no matter how other random variables in the model favor classification of the instance as C_i [8].

In order to accommodate for unobserved instance that may occur in the future, a small probability can be assigned to those values of the variable. This can be achieved by upping all the counts of values of variables by a constant. Table 13 shows the conditional probability distribution of the terms in Table 11, after applying Laplace correction. The Laplace constant used for computing the values in Table 13 is one.

Table 13: Conditional probability distribution of terms after Laplace correction[7]

P(Wi| C)	**Prob**	**P(Wi| C)**	**Prob**
Pr(Dog|P)	4/4	Pr(Dog|F)	2/3
Pr(Cat|P)	3/4	Pr(Cat|F)	1/3
Pr(duck|P)	3/4	Pr(duck|F)	1/3
Pr(Parrot|P)	2/4	Pr(Parrot|F)	1/3
Pr(Sell|P)	2/4	Pr(Sell|F)	3/3
Pr(Buy|P)	1/4	Pr(Buy|F)	3/3

Handling Underflow

Since calculating the posterior probabilities involves multiplication of the conditional probabilities of all the random variables, the values of which are very small, it might lead to the problem of underflow. That is, the values might fall below the smallest value that can be stored in memory. Hence, those values get replaced by negative values or zeros, depending on the data type. In order to avoid such a situation, we can take log of all the probabilities and then add them up. Since we are finding the maximum likelihood, the absolute value of the probabilities does not matter [8] .

4.3 Implementation

The steps to be followed to implemented the Naive Bayes model are described below.

Initialization :

```
attrList ← List of attributes in the dataset
attrStates[i] ← list of possible values that attribute 'i' can take.
N ← total number of instances in training set
C ←list of possible class labels
lc ← Laplace constant
tst [$x_1$,...$x_n$] = test sample with attribute values [$x_1$,...$x_n$]
Note : len(C) =  Count of values in C and len(attrStates[i]) = Count of values in attrStates[i]
```

Computation:

Training:

```
for each c in C
```

```
    $N_c$ = total number of occurrences of c
    Nr = $N_c$ + lc
    Dr = N + (lc · len(C))
    prior[c] = Nr / Dr

    for each i in attrList
        Count[i,c] = total number of combined occurrences of 'i' and 'c'
        Nr = Count[i,c] + lc
        Dr = $N_c$ + (lc · len(attrStates[i]))
        CondProb [x | c] = Nr / Dr

return prior, CondProb
```

Testing:

```
for each c in C
    posterior(c | [$x_1$,...$x_n$])  = $\log(\text{prior}[c]) + \sum_1^n \log(\text{CondProb}\ [xi\ |c])$
testClass = max { posterior[c] }
```

4.4 Flowchart Representing Naive Bayes Implementation

The important modules in building a NB model, namely, prior probability computation, conditional probability computation and classifying test instances are depicted in the flowcharts of Figure 6, Figure 7 and Figure 8, respectively.

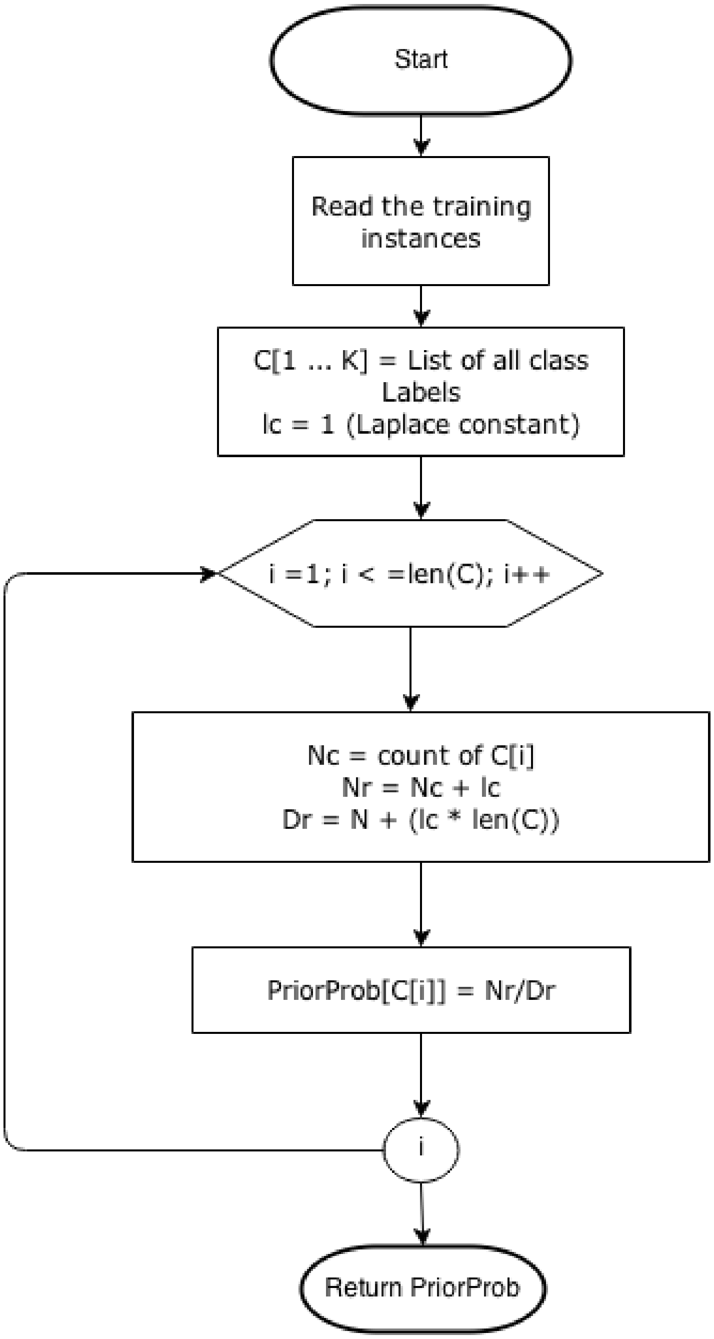


Figure 6: Flowchart Representing Computation of Prior Probability

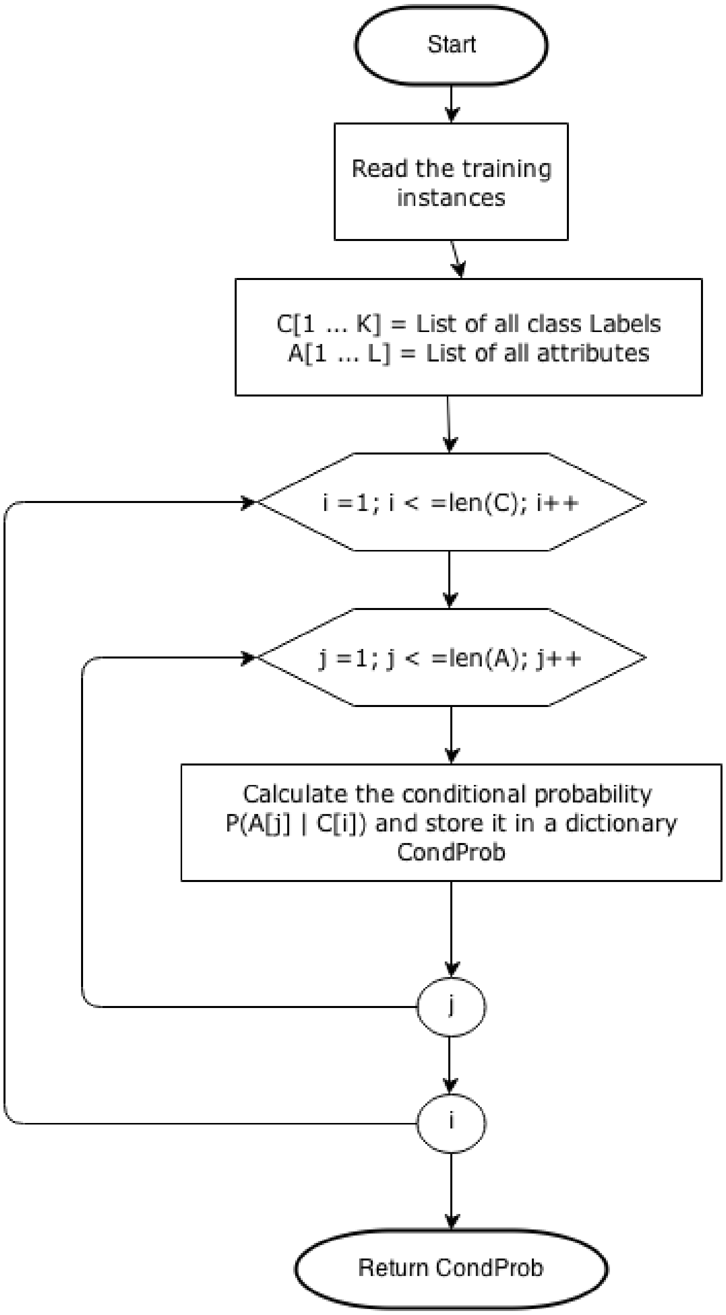

Start
Read the training instances
C[1 ... K] = List of all class Labels
A[1 ... L] = List of all attributes
i =1; i < =len(C); i++
j =1; j < =len(A); j++
Calculate the conditional probability P(A[j] | C[i]) and store it in a dictionary CondProb
j
i
Return CondProb

Figure 7: Flowchart Representing Conditional Probability Computation

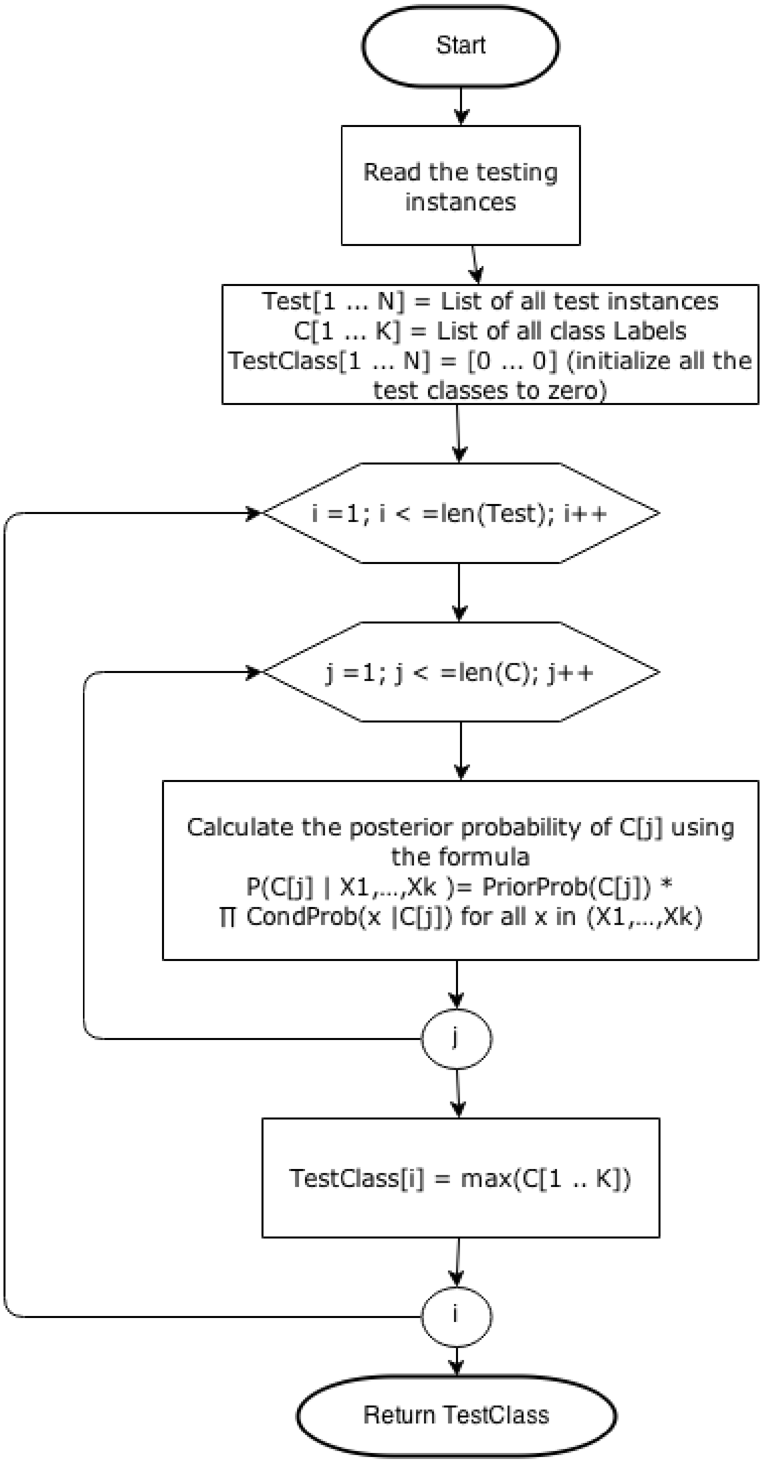

Start
Read the testing instances
Test[1 … N] = List of all test instances
C[1 … K] = List of all class Labels
TestClass[1 … N] = [0 … 0] (initialize all the test classes to zero)
i =1; i < =len(Test); i++
j =1; j < =len(C); j++
Calculate the posterior probability of C[j] using the formula
P(C[j] | X1,…,Xk)= PriorProb(C[j]) *
∏ CondProb(x |C[j]) for all x in (X1,…,Xk)
j
TestClass[i] = max(C[1 .. K])
i
Return TestClass

Figure 8: Flowchart representing testing of NB model

4.5 Complexity of Naive Bayes

The computation intensive steps that are involved in the NB model are the prior probability calculation and conditional probability calculation for all the attributes given a class label.

Let C be the total number of class labels in the dataset.
Let N be the total number of attributes in the dataset.
Let $\{S_1,...,S_N\}$ be the total number of values that each attribute can take corresponding to the attributes $\{A_1,...,A_N\}$.
Let S_{max} be the max $\{ S_1,...,S_N \}$.
Let R be the total number of training instances.
Let us assume that, counting the number of occurrences of each of the class labels and attribute states can be done in a single scan through the entire set of training instances. Hence, the complexity of the process is *O(R)*.
To calculate the prior probability, we need to find the probability of each of class labels in the training dataset. The complexity for this process is *O(C)*.
To calculate the conditional probability of each of the attributes, we need to find the probability of occurrences of every state of all the attributes conditioned on each of the class labels. Hence, the total number of values computed in this process is given by:

$$\begin{aligned} Total\ values\ computed &= \sum_{i=1}^{N} C \cdot S_i \\ &= C \cdot S_1 + \ C \cdot S_2 + \cdots + \ C \cdot S_N \\ &\leq C \cdot \ (S_{max} + S_{max} + \cdots + S_{max}) \\ &\leq N \cdot C \cdot \ S_{max} \ . \end{aligned}$$

Thus, the complexity of training the classifier is determined by the computation of conditional probability, which is given by $N \cdot C \cdot \ S_{max}$.
After training, the conditional probabilities and prior probabilities are stored and can be retrieved during the process of classification in constant time. When classifying the test data, the model needs to find the probabilities of each instance to belong to each of the class labels. To calculate each of those values, the operation loops over each class variable, over each of the attribute for every test instance. Hence, the process takes C·N·R time complexity. Thus, the time complexity

of the Naive Bayes model is mainly determined by the complexity of finding the conditional probabilities.

The next chapter describes the TAN model, which is an improvement over the Naive Bayes model by augmenting a tree structure among the attributes in the system.

CHAPTER 5

Tree Augmented Naive Bayes Model

5.1 Introduction

The Naive Bayes model discussed in the previous chapter, encodes incorrect independence assumptions that, given the class label, the attributes are independent of each other. But in the real world, the attributes of any system are mostly correlated and the case as in Naive Bayes rarely happens. In spite of such incorrect independent assumptions, the Naive Bayes model seems to perform fairly well. So, if the model also takes into account the correlations between the attributes, then the classification accuracy can be improved [5].

Bayesian networks capture all the correlations in the random variables in the form of a graph. Using such a Bayesian network, we can first calculate the conditional probabilities of all the random variables given its parents. Then using the independence statements encoded in the network, we can calculate the joint distribution using the local conditional probabilities[5]. But learning such a Bayesian network is very complex because there may be many random variables in a network, and each random variable may take many values. Also a single random variable can have many parents and finding the conditional distribution conditioned on all those parents increases the complexity. Above all, the main parameter that determines the classification in a Naive Bayes model, $P(C \mid X_1,...,X_n)$ takes into account all the variables. But a general Bayesian network may not have edges from the class node to all the variables. This might sometimes lead to lower accuracy in classification. Hence, we can conclude that, for better classification performance, we need a Bayesian network that encodes the structure of the Naive Bayes model and in addition to that, also captures the correlations between the variables in the system[5].

The solution to this issue can be an augmented Naive Bayesian network. An augmented Naive Bayesian network, maintains the structure of the Naive Bayesian network and augments it by adding edges between the variables in order to capture the correlations between the attributes. But this processes increases the computational complexity. "While the induction of the Naive Bayesian classifier requires only simple book keeping (storing conditional probabilities given the label), the induction of Bayesian networks requires searching the space of all possible networks, i.e., the space of all possible combination of edges."[5]

In order to reduce the computational complexity and also take into account the correlations between the variables, restrictions need to be imposed on the level of interaction between the variables. One such model, is the Tree Augmented Naive Bayesian (TAN) model. This model imposes a restriction on the level of interaction between the variables to one. In a TAN model, all the variables are connected to the class variables by means of direct edges. Hence, it takes into account all the variable while determining P(C | X_1,...,X_n). In addition to that, each variable can be connected to another variable in the network [5]. That is, each variable in the graph can have two parents viz., the class node and another variable node, except for one variable which is called root. The computational complexity of this model, is greatly reduced, as each variable has a maximum of two parents. "Thus TAN maintains the robustness and computational complexity of the Naive Bayes model and at the same time displays better accuracy" [5].

5.2 The Model

A Tree Augmented Naive Bayes model (TAN) imposes a tree structure on the Naive Bayes model, by restricting the interaction among the variables to a single level. A TAN model for a simple medical diagnostic system is shown in Figure 9.

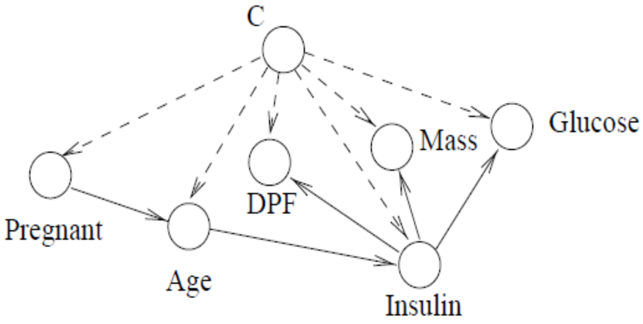


Figure 9: TAN model learned for a data set called Pima [5].

Figure 9 shows the TAN model for the Pima dataset, to classify if a patient tests positive or negative for diabetes. The variables in this data set are all derived from patients who were women more than 21 years of age. In this dataset, the variables glucose and insulin are closely related in determining the class variable. For example, very low glucose level, seen independently is surprising. But given the fact that the insulin level is also high, then it becomes unsurprising[5]. This correlation can be captured in the TAN model. But in case of Naive Bayes model, they are considered as two separate abnormal events. Thus, the Naive Bayes classifier will over penalize the class label[5].

In Figure 9, it can be observed that, all the attributes, except the attribute pregnant, have two parents viz., the class and another attribute. The interactions between the attributes is shown using solid edges and the interaction between the class and an attribute is shown using dotted edges in Figure 9. The attribute pregnant is the root node of the tree and it has only one parent i.e. the class. By removing the edges from class to all the attributes (all the dotted edges), a tree structure can be visualized. Also, it can be observed that all the edges in the tree point outward from the root node pregnant. This is an example of a tree structure, constructed between the attributes of the system, in a TAN model.

5.3 Implementation (Algorithm)

The key feature in the TAN model is the tree structure. In order to construct a tree, the parent of each of the attributes needs to be identified. Also, only the most correlated attributes must be connected to each other. Chow and Liu [9] describe a procedure for building this tree structure. "This procedure reduces the problem of constructing a maximum likelihood tree to a maximum weighted spanning tree in a graph"[5].

One other important notion that plays a key role in constructing the tree is the mutual information. To construct the tree, we need to measure the correlation between each pair of variables in the system and add edges only between those variables that are highly correlated. If there are N variables in a system, then the corresponding tree structure will have N nodes. Thus, N-1 edges should be added, to get a tree structure that connects all the nodes in the graph. Also, the sum of the weights of all these edges needs to be the maximum weight among all such tree structures. The measure of the correlation between two variables that forms the weight of an edge in the graph is called the mutual information.

The mutual information between two random X and Y variables as given in [5] is defined as:

$$I_p(X;Y) = \sum_{x,y} P(x,y) \log\left(\frac{P(x,y)}{P(x)P(y)}\right)$$

Given a pair of variables, this function measures how much information one variable provides about the other. To construct a tree for the TAN model, we use conditional mutual information between two attribute to determine the edges that will form the tree. The conditional mutual information used for constructing TAN as given in [5], is defined as:

$$I_p(X; Y \mid Z) = \sum_{x,y,z} P(x, y, z) \log\left(\frac{P(x, y \mid z)}{P(x \mid z)P(y \mid z\,)}\right)$$

The algorithm to construct a tree for the TAN model is as follows [5]:

"The tree construction procedure consists of 5 main steps

1. Compute $I_p(X_i ; X_j \mid C)$ between each pair of attributes $i \neq j$.
2. Build a complete undirected graph in which the vertices are the attributes $X_1,...,X_n$. And annotate the weight of an edge connecting X_i to X_j by $I_p(X_i ; X_j \mid C)$.
3. Build a maximum weighted spanning tree.
4. Transform the resulting undirected tree to a directed one by randomly choosing a root variable and setting the direction of all the edges outward from the root" [5].

After the construction of the tree, the conditional probability of each of the attributes conditioned on its parent and class label is calculated and stored. Also, the conditional probability of the root variable conditioned on class is computed and stored. Then, the posterior probability for each of the class label: $P(C \mid X_1,...,X_n)$, is calculated as a product of the prior probability, the conditional probability of root variable and conditional probability of all the attributes. The class label with the maximum posterior probability value, is assigned to the test sample. The TAN implementation also uses the same Laplace correction and under flow handling techniques discussed for the Naive Bayes model. In this project, the root variable is always the very first variable found in the dataset. The formula for the TAN classification is given by [5]:

$$\mathrm{P}(\mathrm{C} \mid \mathrm{X}_1, \dots, \mathrm{X}_n) = \mathrm{P}(\mathrm{C}) \cdot\ \mathrm{P}(\mathrm{X}_{\mathrm{root}} \mid \mathrm{C}\,) \prod_{\mathrm{i}} \mathrm{P}(\mathrm{X}_{\mathrm{i}} \mid \mathrm{C}\,, \mathrm{X}_{parent})$$

5.4 Flowchart Representing TAN Implementation

The important modules in building the TAN model: calculating pair wise probability, construction of tree and classifying test instances are depicted in the form of flowcharts in Figure 10, Figure 11 and Figure 12, respectively. The TAN model also includes modules to calculate prior and conditional probabilities. These are similar to the computations done in the NB model and their flowcharts are represented in Figure 6 and Figure 7, respectively.

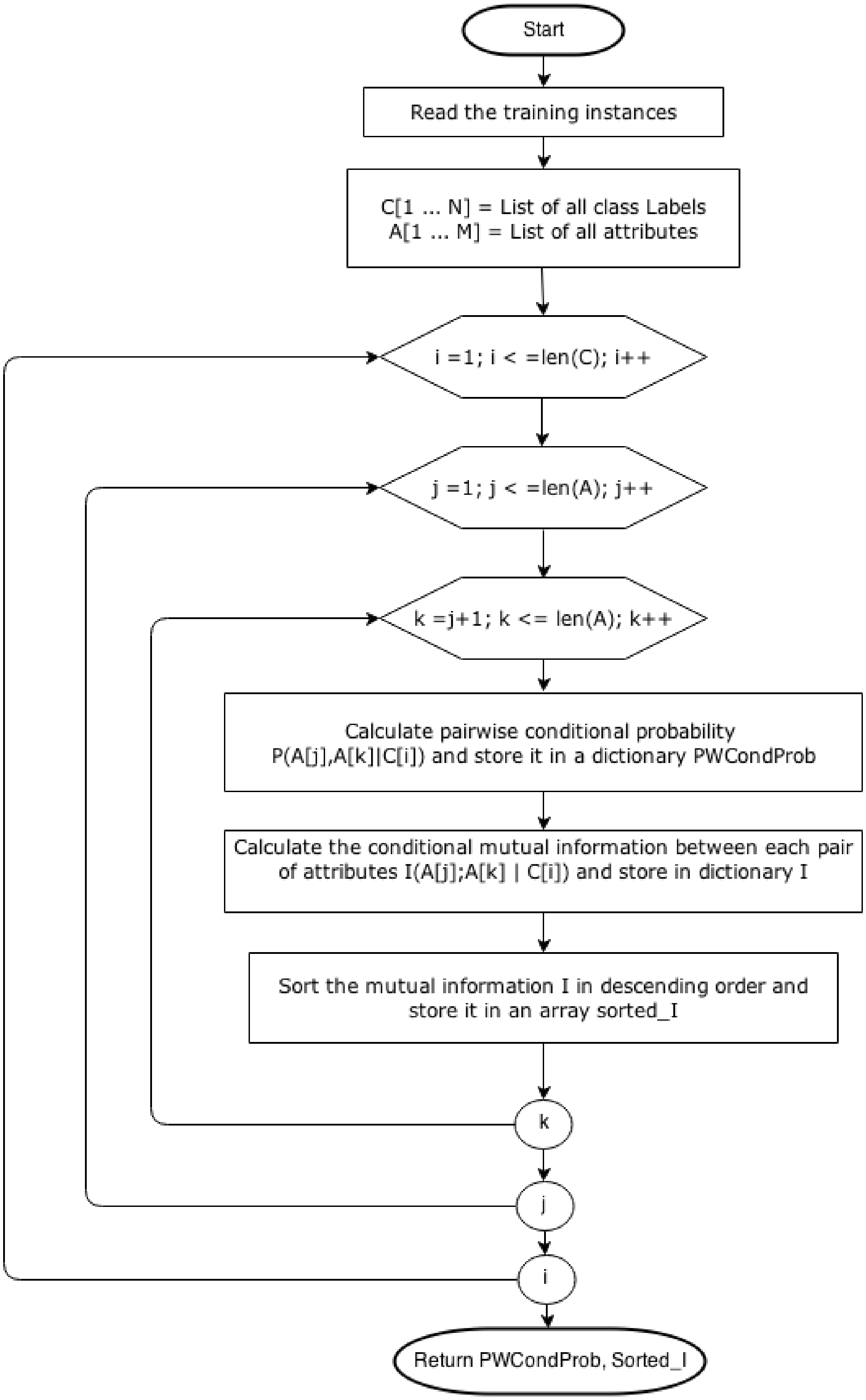

Start
Read the training instances
C[1 ... N] = List of all class Labels
A[1 ... M] = List of all attributes
i =1; i < =len(C); i++
j =1; j < =len(A); j++
k =j+1; k <= len(A); k++
Calculate pairwise conditional probability
P(A[j],A[k]|C[i]) and store it in a dictionary PWCondProb
Calculate the conditional mutual information between each pair
of attributes I(A[j];A[k] | C[i]) and store in dictionary I
Sort the mutual information I in descending order and
store it in an array sorted_I
k
j
i
Return PWCondProb, Sorted_I

Figure 10: Flowchart Depicting Pair-wise Conditional Probability Computation

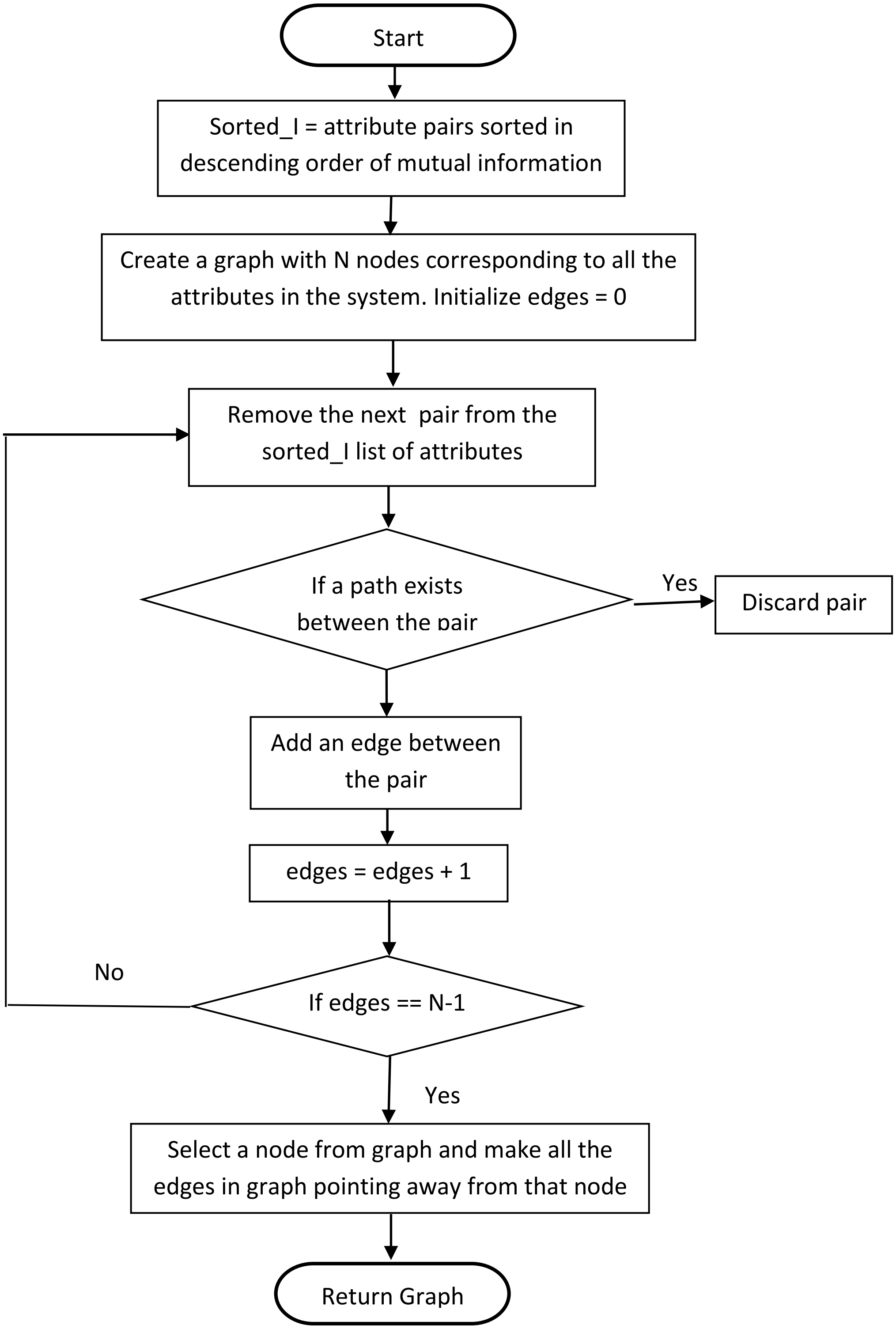

Start
Sorted_I = attribute pairs sorted in descending order of mutual information
Create a graph with N nodes corresponding to all the attributes in the system. Initialize edges = 0
Remove the next pair from the sorted_I list of attributes
If a path exists between the pair
Yes
Discard pair
Add an edge between the pair
edges = edges + 1
No
If edges == N-1
Yes
Select a node from graph and make all the edges in graph pointing away from that node
Return Graph

Figure 11: Flowchart Representing Procedure to Construct Tree in TAN Model

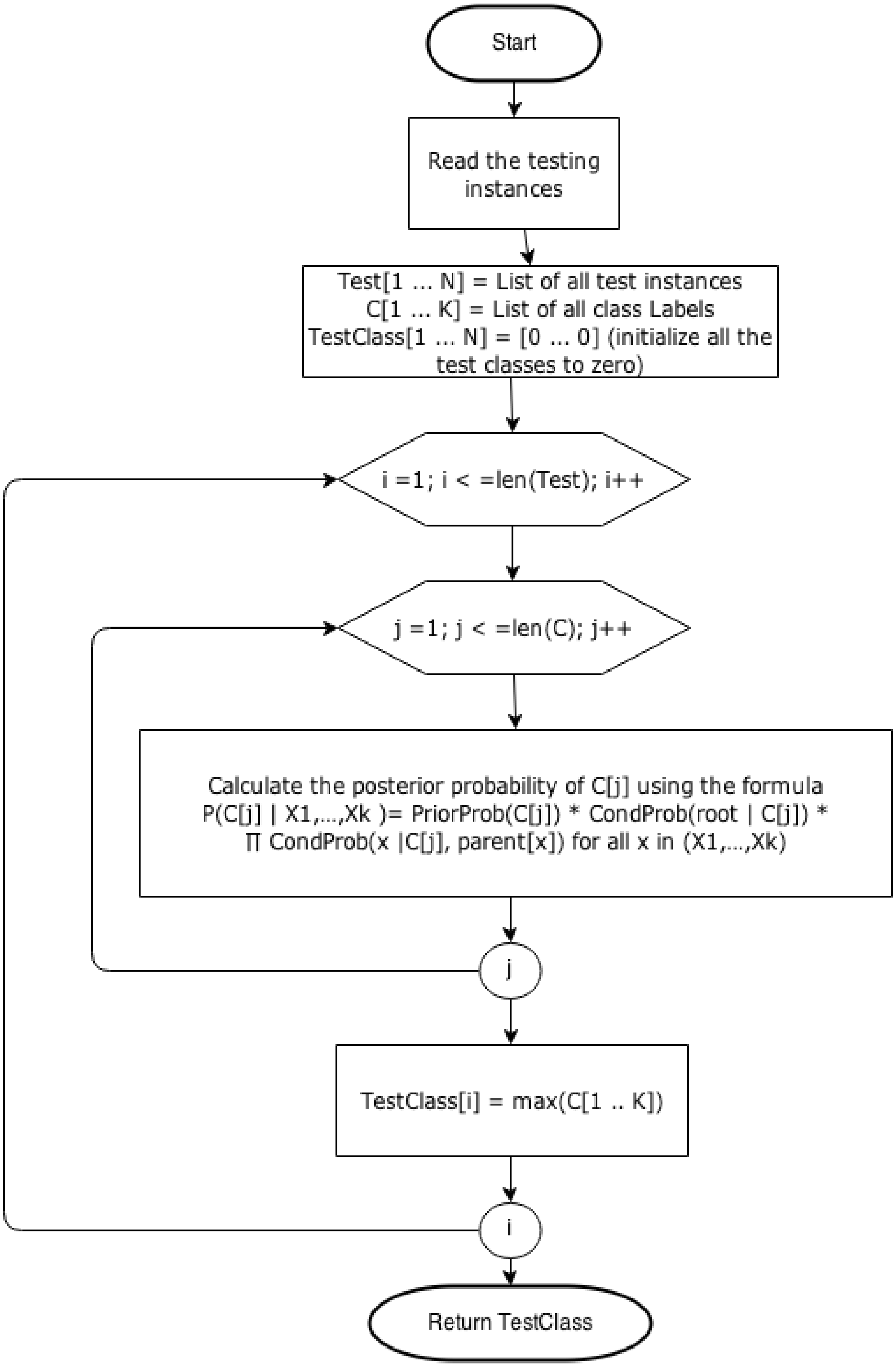

Start
Read the testing instances
Test[1 ... N] = List of all test instances
C[1 ... K] = List of all class Labels
TestClass[1 ... N] = [0 ... 0] (initialize all the test classes to zero)
i =1; i < =len(Test); i++
j =1; j < =len(C); j++
Calculate the posterior probability of C[j] using the formula
P(C[j] | X1,...,Xk)= PriorProb(C[j]) * CondProb(root | C[j]) *
∏ CondProb(x |C[j], parent[x]) for all x in (X1,...,Xk)
j
TestClass[i] = max(C[1 .. K])
i
Return TestClass

Figure 12: Flowchart Representing the Flow of Testing Procedure in TAN Model

5.5 Complexity of TAN

The computation intensive steps that are involved in the TAN model are: the prior probability calculation, the conditional probability calculation for every attribute conditioned on its parent and class and calculation of mutual information for every pair of attributes.

Let C be the total number of class labels in the dataset.
Let N be the total number of attributes in the dataset.
Let $\{S_1,...,S_N\}$ be the total number of values that each attribute can take.
Let S_{max} be the max $\{ S_1,...,S_N \}$.
Let R be the total number of training instances.
Let us assume that, counting the number of occurrences of each of the class labels and pair wise occurrences of attribute states, conditioned on class, can be done by a single scan through the entire set of training instances. Hence, the complexity of the process is *O(R)*.

Prior Probability

The complexity of prior probability calculation for TAN is the same as for the NB. I.e., *O(C).*

Mutual Information

The total number of pairs for N attributes is $\left(\frac{N(N-1)}{2}\right)$.

For each of the pairs, we need to calculate the probabilities of all combinations of all the states, that each attribute can take. By upper bounding the number of attribute states for both the attributes to S_{max}, the number of operations that should be performed to find the mutual information is given by:

$$\text{Total number of operations} \leq \left(\frac{N(N-1)}{2}\right) \cdot S_{max}^2$$

$$\leq \left(\frac{(N^2-N)}{2}\right) \cdot S_{max}^2 \ .$$

Conditional Probability

The number of operations for each conditional probability, differs slightly from NB. In TAN we need to find the conditional probability of each of the attributes, conditioned on its parent and class. Hence, the number of states of parent also needs to be multiplied. This is given by:

```
for i ←1 to N
        calculate $P(A_i = x \mid A_j = y, C = z)$
```

where A_j is the parent of A_i
$x \leftarrow (1 \text{ to } S_i)$ i.e., each of the states in S_i.
$y \leftarrow (1 \text{ to } S_j)$ i.e., each of the states in S_j.

Hence, the time complexity for calculating conditional probability is given by:

$$= \sum_{i=1}^{N} C \cdot S_i \cdot S_j$$

$$\leq \quad N\, S_{max}^2 \cdot C.$$

Thus, by observing the complexities of the tree operations, it can be concluded that the complexity of the TAN model, is highly influenced by the mutual information calculation, which increases with the increase in the number of attributes.

It can also be observed that, the complexity of the TAN model is polynomial, while the complexity of the Bayesian network is exponential. Thus, by restricting the level of dependency between the attributes , we can have a less complex model. Generally, it is a trade of between the complexity and performance. Models that have higher levels of dependency tend to perform better, but are very complex. On the other hand, models that have lower dependencies may not always perform in par with complex models.

The next chapter presents the results of Naive Bayes and TAN models on various datasets and analyzes the results.

CHAPTER 6

Analysis of Datasets and Results

The datasets for this project were obtained from the KEEL dataset repository [10]. It comprises both discrete and continuous valued datasets. All data need to undergo some preprocessing steps, to make them suitable for using as input to the program. The training and testing of the Naive Bayes and TAN classifiers on these datasets is done by using ten-fold cross validation methodology.

6.1 Data Preparation

The data files considered for this project were in KEEL format. "They have the following structure.

- **@relation:** Name of the data set
- **@attribute:** Description of an attribute (one for each attribute)
- **@inputs:** List with the names of the input attributes
- **@output:** Name of the output attribute
- **@data:** Starting tag of the data

The rest of the file contains all the instances belonging to the dataset in comma separated values format"[10].

The description of each of the attributes specifies whether it is discrete or continuous valued and in case of continuous, specifies if the attribute is of the type integer or real.

Data Discretization

If the dataset has attributes that are continuous valued, then those attributes are first discretized and then used by the classifier. The method used to discretize the data in this project is called equal width binning or equal width partitioning[4]. The steps used in this method are:

- Find the minimum and maximum values that the attribute can take.
- Calculate the range by finding the difference between minimum and maximum values.
- Divide the range into N equal parts. Each part corresponds to a bin with a sub range.

- Assign each of the attribute values to the corresponding bin (i.e. to the bin within whose sub range the attribute value fits).

The value of N depends on the range of the attribute and the number of classes in the dataset[4].

6.2 Ten-Fold Cross Validation

In a ten-fold cross validation, the original data set is divided into ten equal parts. Out of those ten parts, one part of the dataset is used for validation or testing and the remaining nine parts are used for training the classifier. This process is then repeated ten times and each of the ten parts is used as testing data, exactly once[11]. The accuracy for each fold, is measured as a percentage of correctly classified instances and an average of the accuracy across all the folds gives an overall estimation about the performance of the model. The advantage of this method is that it ensures each instance in the dataset is used both, as a training and testing sample and every instance is used exactly once as a testing sample[11].

6.3 Implementation

The Naive Bayes and TAN models were implemented using the programming language Python. A separate module named **`attr_extract`**, was coded to extract the attribute and class information from the training file. Another module named **`drawGraph`**, was coded to display the tree structure of attributes in the TAN model. We next explain the details of the methods used in building the classifier models.

Naive Bayes Model:

There are five methods used in the NB model viz., **`calcPriorProb`**, **`calcCondProb`**, **`NBTest`**, **`calcFoldAccuracy`** and **`OverallAcc`**. The method **`calcPriorProb`** is used to calculate the prior probabilities of all the class labels in the training data and store it in a dictionary. The method **`calcCondProb`** is used to calculate and store the conditional probabilities of each of the attribute values, conditioned on class label. The **`NBTest`** method, calculates the posterior probability of each of the class labels for all the test instances and classifies the test instances appropriately. The methods **`calcFoldaccuracy`** and **`OverallAcc`** are used to calculate the accuracies of the model for each fold and the final accuracy across all folds, respectively.

TAN Model:

The main methods in the TAN model are **`calcPWCondProb`**, **`mutualInfo`**, **`constructTree`**, **`calcTANCondProb`**, **`TANTestProb`**. Apart from these methods, TAN also uses the methods described in the NB model: **`calcPriorProb`**, **`calcCondProb`**, **`calcFoldAccuracy`** and **`OverallAcc`** . The method **`calcPWCondProb`** calculates the pair wise conditional probability of attribute values, conditioned on class and stores it in a dictionary. The method **`mutualInfo`**, calculates and stores the mutual information of all pairs of attributes sorted in descending order in an array. The **`constructTree`** method is used to construct the tree structure using the mutual information between attribute pairs. The **`TANCondProb`** method, computes the conditional probability of each attribute given its parent and class label and stores it in a dictionary. Finally the **`TANTestProb`** method, calculates the posterior probability of each of the class labels, for all the test instances and classifies the test instances appropriately.

The data structures mainly used for the implementation of both classifiers are data frames, dictionaries and lists. Data frame is a data structure that is similar to the table, with rows and columns. The rows have indexes and the columns have column headers. It is used to store the training and testing instances. Dictionaries are similar to hash tables and are used to store the probability values, with attributes and class labels as keys and probabilities as values. List are used to store data, such as list of attributes, list of values taken by each attribute etc.

6.4 Need for Better Classification Models

Let us consider a simple classification problem which has only two classes. For such a problem, it is possible that, by classifying the instances randomly as one of two classes, we might end up classifying fifty percent of the instances correctly. Hence, using any sophisticated model, that does not perform significantly better than such a randomized model, does not make any sense.

Classifying instances based on the prior probability of class labels, is another method, that might perform slightly better than random classification. In this method, firstly we need to find the prior probability of each of the class labels from the training set. Then we can classify all the test instances to the label that has the highest prior probability. Considering a simple two class model, this method will result in correct classification of at least fifty percent of the instances.

This is better than a random classification model because, in case of a two class model, a random classification model gives an accuracy around fifty percent. But using a prior probability classification gives at least fifty percent accuracy.

These two methods act as a baseline for any classifier model. Thus, any other sophisticated model, needs to perform significantly better than these methods. The performance increase must be worth the increase in the complexity of the model.

6.5 Analysis of Results

Now let us consider the performance of these baseline methods and more complex models such as Naive Bayes and TAN with some datasets. Another important aspect to be considered, is the way the performance scales with the increase in the number of classes.

Tables 14, 15, and 16 give the ten-fold cross validation results for three different datasets across four models. The description of the terms used in tables and in all further discussions is as follows:

RC: Random Classification (randomly choosing a class label).

PPC: Prior Probability Classification (classifying all the samples to the class having maximum prior probability).

NB: Naive Bayes model.

TAN: Tree Augmented Naive Bayes model.

Weka: A free machine learning software, that has a collection of data analysis algorithms [11]. *Weka NB* refers to the Naive Bayes model in the Weka package.

Note: The overall accuracy for RC, given in the tables, is the average of the ten-fold accuracies calculated over 100 iterations.

Table 14: Ten-fold cross validation results of various data models on Pima dataset

Dataset Name: Pima **Classes : 2** **Attributes: 8**

Description: Pima Indians diabetes dataset. The attributes are, test results of women patients, of at least 21 years of age. The classification problem is to identify based on the patient features, whether she tests positive or negative to diabetes [10].

Fold	**RC**	**PPC**	**NB**	**TAN**
1	45.45	64.93	70.13	80.52
2	57.14	64.93	70.13	76.62
3	42.31	64.10	73.08	75.64
4	55.26	65.79	82.89	76.32
5	59.74	64.93	72.73	79.22
6	54.54	64.93	75.32	79.22
7	48.68	65.79	69.74	80.26
8	54.54	64.93	79.22	83.12
9	50.00	65.79	78.95	78.95
10	45.45	64.93	72.73	66.23
Overall Accuracy	50.11	65.10	74.48	77.60
Weka NB accuracy :			78.12	

From Table 14, it can be inferred that, for the Pima dataset, the accuracy of NB and TAN models are better when compared to RC and PPC methods. The performance for RC method is the worst among the four.

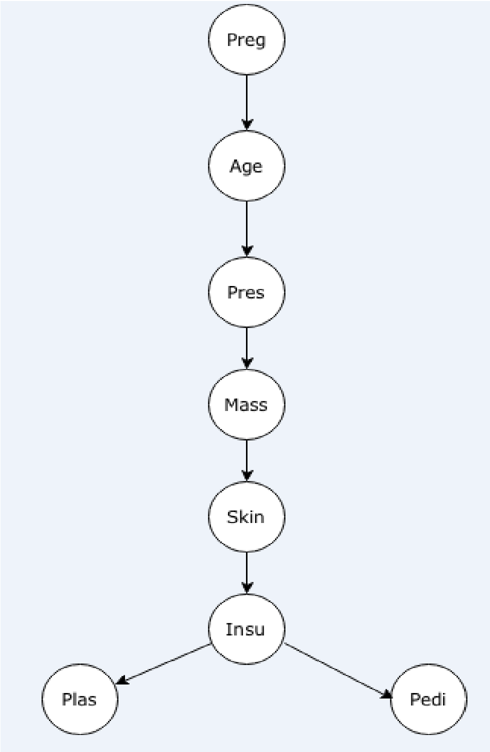


Figure 13: Augmented Tree structure of Attributes for the TAN model on the Pima Dataset

Figure 13 presents the augmented tree structure obtained by the TAN implementation of this project on the Pima dataset. In Figure 13, the thick end of the edges denote an arrow pointing towards that node. The graph depicts the relationship among various attributes in the dataset. The description of each of the attribute names is as follows [12]:

preg:	Number of times pregnant
Plas:	Plasma glucose concentration
Pres:	Diastolic blood pressure
Skin:	Triceps skin fold thickness
Insu:	2-Hour serum insulin
Mass:	Body mass index
Pedi:	Diabetes pedigree function
Age:	Age

From Figure 13, it can be observed that, the attribute *Preg* has no parents. Hence, it is the root node of the tree. It can also be seen that all the nodes are pointing away from the root node. This is a typical structure of a tree of TAN model. The tree has N- 1 edges. It can be seen that attribute *Insu* (Insulin) is the parent of *Plas* (Plasma glucose). This implies that *Insu* and *Plas* are among the highly correlated attributes in the dataset. This is not surprising since insulin and glucose levels are dependent on each other in real life. If the insulin is high, then the glucose level is low and vice versa. Hence, the TAN model captures the inherent dependency among the attributes in dataset. On the other hand, the NB model considers the occurrence of very low glucose level and very high insulin level as two separate abnormal events and hence over penalizes the class label [5]. But the TAN model captures the dependency among these feature and considers it as a single event. From Table 14 it can be seen that the performance of TAN is slightly better when compared to the NB model.

Table 15: Ten-fold cross validation results of various data models on Iris dataset

Dataset Name: Iris **Classes : 3** **Attributes: 4**

Description: Each class refers to a type of iris plant. Attributes are the petal and sepal features of the plant. The classification problem is to identify the type of the iris plant, based on the attribute values [10].

Fold	RC	PPC	NB	TAN
1	20.00	33.33	100.00	100.00
2	40.00	33.33	93.33	100.00
3	60.00	33.33	66.67	86.67
4	46.67	33.33	93.33	93.33
5	26.67	33.33	100.00	100.00
6	40.00	33.33	93.33	100.00
7	6.67	33.33	93.33	93.33
8	46.67	33.33	100.00	93.33
9	40.00	33.33	93.33	100.00
10	20.00	33.33	93.33	100.00
Overall Accuracy	33.35	33.33	92.67	96.67
Weka NB accuracy :			94.00	

The Iris dataset has three class labels. The ten folds of this dataset are divided such that, each of the training files contains 135 instances, with 45 instances belonging to each class label. The test files have 15 instances in total, with 5 instances belonging to each class label. Since this dataset has 3 class labels, a random classification should give an accuracy of 33.33 %. Also, the way in which the training and test files divided (containing equal number of instances of each class label), should lead to an accuracy value of 33.33% for PPC method as well. This can be observed in the results shown in Table 15. On the other hand the performance results of NB and the TAN models are far better, as they give around 60% more accuracy, when compared to the baseline models (RC and PPC). Another important fact to be noted is that, the overall accuracy for 3 class dataset, is worse for PPC and RC, when compared to the 2 class dataset.

Table 16: Ten-fold cross validation results of various data models on Zoo dataset

Name: Zoo **Class : 7** **Attributes: 16**

Description: The classification problem here is to classify all the instances into a type of animal, based on the features of the animals specified in the attributes[10].

Fold	RC	PPC	NB	TAN
1	8.33	33.33	100.00	91.67
2	25.00	33.33	91.67	100.00
3	0.00	44.44	100.00	100.00
4	0.00	50.00	100.00	100.00
5	12.50	50.00	100.00	100.00
6	30.00	40.00	100.00	100.00
7	33.33	44.44	100.00	100.00
8	10.00	40.00	90.00	90.00
9	20.00	33.33	86.67	80.00
10	0.00	50.00	100.00	100.00
Overall Accuracy	14.19	40.59	96.04	95.05
Weka NB accuracy :			95.05	

Table 16 shows the ten-fold cross validation results for Zoo data set. This dataset has 7 classes. From the results in Table 16, it can be observed that, the RC method performs very poorly when compared to the 2 class and 3 class datasets. The performance of PPC method is poor when compared to the NB and TAN model accuracies.

From Table 14, Table 15 and Table 16, it can be inferred that NB and TAN models perform far better when compared to the baseline methods. Also, as the number of class labels increases, the performance of RC and PPC tends to decrease. But the performance of NB and TAN models does not depend only on the number of class labels. This explains the need for sophisticated models such as NB and TAN.

Now, let us compare the performance of NB and TAN models. By looking at the accuracy results in Table 14 and Table 15, it can be observed that, TAN performs slightly better than the NB model. This implies that, there is more correlation among the attributes of Iris and Pima datasets. Hence, these correlation patterns are captured by the TAN model, leading to better performance. But in case of Zoo dataset, the performance of NB and TAN models is almost the same. Hence, it can be said that there was not enough correlations between the attributes of those datasets. In such cases, a simple NB model is more suitable for classification, as it is computationally less complex when compared to TAN, but gives results similar to TAN. Table 17 gives the overall accuracy results of NB, TAN, and Weka NB models for various datasets.

Table 17: Accuracy results on various datasets of different classifier models

Dataset	RC	NB Accuracy	TAN Accuracy	Weka NB Accuracy	Attributes	Classes
Iris	33.42 (±7.08)	94.00 (±4.67)	96.67 (±4.47)	94. 00	4	3
Letter	3.81 (±0.32)	73.64 (±0.61)	86.84 (±0.58)	74.04	16	26
Pima	50.03 (±4.29)	74.87 (±3.57)	76.69 (±2.86)	78.12	8	2
Vehicle	24.99 (±3.48)	61.11 (±2.97)	72.22 (±4.31)	62.65	18	4
Led7digit	10.27 (±2.95)	72.20 (±4.75)	71.60 (±4.63)	73.20	7	10
Car	25.00 (±2.46)	86.11 (±2.42)	94.73 (±1.97)	85.53	6	4
Chess	50.03 (±1.89)	87.73 (±1.39)	92.15 (±1.37)	87.89	36	2
Zoo	14.19 (±9.39)	94.06 (±5.68)	95.05 (±6.60)	95.05	16	7
Heart	50.05 (±6.34)	82.96 (±6.87)	80.74 (±8.25)	83.33	13	2
Breast	49.70 (±6.95)	74.01 (±8.07)	67.87 (±9.33)	71.68	9	2
TicTac	50.06 (±3.54)	69.31 (±4.01)	75.89 (±2.47)	69.62	9	2

Let us consider the performance of NB and TAN models on the Led7digit dataset. From Table 17, it can be seen that, there is not much difference between the performance of NB and TAN for this dataset. Now let us analyze the data. "The data contains seven Boolean attributes, one for each light-emitting diode of a 7-segment display" [10]. The classification problem is to identify the digit in the display based on the attribute values. This dataset perfectly fits the NB model. This is because, given the class label, there is no dependency among the attributes. For example, given the digit is 7, the attributes do not depend on each other. The class label itself helps in deciding which of the seven led segments would glow. The glowing of one segment is not dependent on the other, given the number in the display. Hence, there is not much of an improvement in the performance of the TAN model, when compared to the NB model. The tree structure for the Led7digit dataset is given in Figure 14.

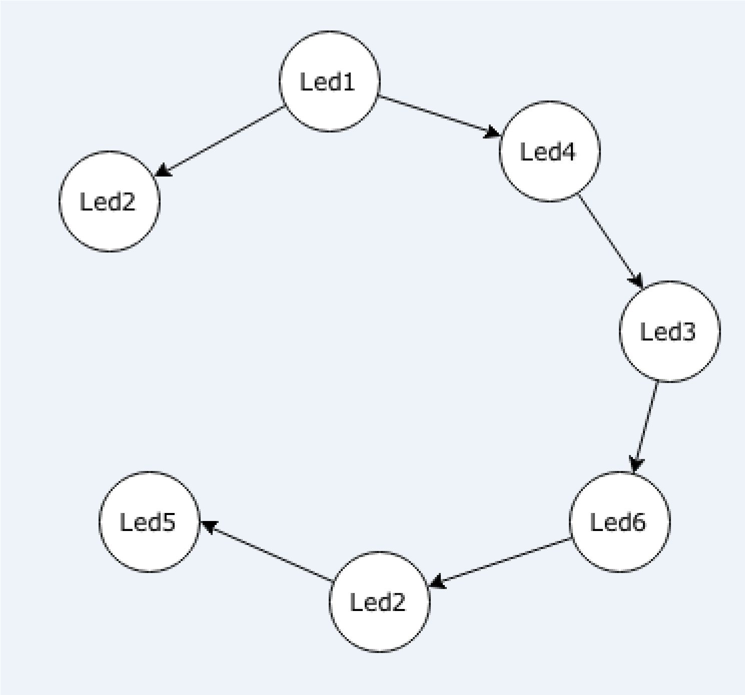


Figure 14: Tree structure of TAN model on Led7digit Dataset

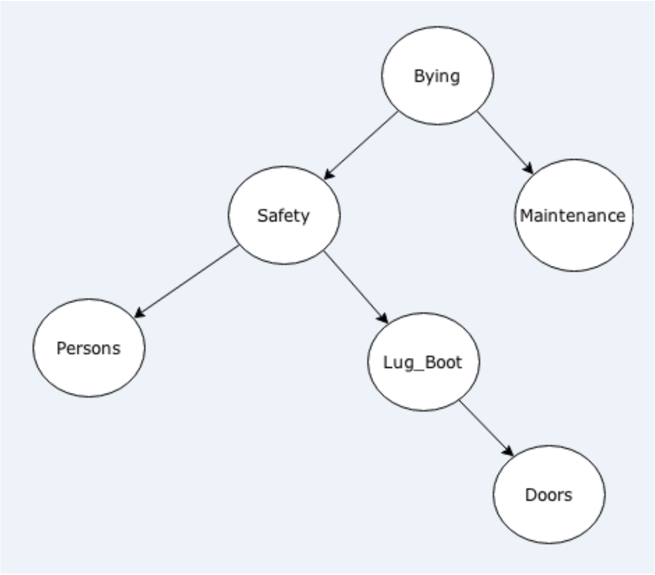


Figure 15: Augmented Tree structure of TAN model on Car Dataset

Now, let us consider the performance of NB and TAN models on the Car dataset. It can be observed that TAN performs far better when compared to the NB model. Figure 15 gives the augmented tree structure of TAN model. Now let us look into the dataset. Based on the values of the attributes: buying, maintenance, safety, persons, door and luggage boot, the classifier needs to evaluate the car as acceptable, unacceptable, very good and good[12]. From Figure 15, we can get some intuitive understanding about the attribute buying price. This attribute is directly connected to the attributes safety and maintenance price. It makes sense that, a car with high buying price would have less maintenance costs and good safety in order to be categorized as a good car. Similar correlations between various other attributes can be captured by the TAN model, hence leading to a better performance.

Table 18: Accuracy results of classifier models across various text data sets

Dataset	RC	NB	TAN	Weka NB	Classes	Attributes
Trade	49.98 (±0.71)	95.08 (±0.10)	96.02 (±0.21)	95.06	2	100
Corn	49.99 (±0.71)	97.61 (±0.20)	99.33 (±0.01)	97.61	2	100
Grain	49.99 (±0.76)	97.52 (±0.03)	98.10 (±0.23)	97.54	2	100
Wheat	50.01 (±0.65)	97.84 (±0.16)	98.81 (±0.02)	97.84	2	100
Acq	50.01 (±0.76)	90.86 (±0.58)	92.95 (±0.46)	90.84	2	100
Crude	50.03 (±0.78)	98.08 (±0.21)	98.57 (±0.34)	98.05	2	100
Earn	50.09 (±0.62)	96.13 (±0.01)	96.30 (±0.25)	96.12	2	100
Money	50.03 (±0.66)	94.30 (±0.11)	95.32 (±0.47)	94.28	2	100
Interest	50.00 (±0.62)	94.71 (±0.49)	96.22 (±0.02)	94.69	2	100
Ship	49.98 (±0.62)	98.93 (±0.21)	99.18 (±0.19)	98.91	2	100
CornGrain-Wheat	33.45 (±1.96)	76.86 (±0.33)	81.13 (±1.20)	-	3	173

Table 18 provides the results for NB and TAN models on text datasets. In Table 18, the first ten rows correspond to ten binary class datasets. The classifier problem is to identify if an instance belongs to the type of the dataset. For example, for Trade dataset, the classifier needs to identify if the instance belongs to the label Trade or not. Similarly for corn, wheat, grain and so on. Both the NB and the TAN models give almost the same performance for all the datasets. Though the TAN model seems to perform slightly better in some of the datasets, the improvement in the performance is not significant enough. Hence, it can be said that, there was not enough correlations between the attributes in these datasets. For such datasets NB model is preferable, as it is less complex and gives good performance results.

By analyzing the corn, wheat and grain dataset, it was found that, they have many overlapping attributes. Hence, these datasets were combined. All the negative classes in each of these datasets were removed. The combined dataset had three class labels: corn, grain and wheat. The classifier needs to classify each of the test instance to one of these class labels. The accuracy results of the classifier on the combined dataset is shown in Table 18. It can be observed that, the TAN model performs much better for this three class dataset. This can be attributed to the fact that, combining all the datasets resulted in bigger training set. So the model might be trained well. Also, combining the datasets has resulted in the increase in the number of attributes. So, correlations among the attributes, in the initial uncombined dataset of each class label, can be captured by the TAN model. For example, the correlation between the set of all attributes belonging to the original uncombined corn dataset, would be captured by the TAN model in the combined dataset. Hence, it would lead to a better performance of the TAN model.

CHAPTER 7

Conclusion and Future Work

The comparative analysis on the performance of TAN and NB models shows that, the TAN model performs better in many cases. Even if there are no inherent correlations between the variables in a system, the accuracy results of the TAN model are close to the NB model. This implies that, taking into account the correlations between the variables in a system, would lead to better performance. Thus, in this project, by adding one level of dependency among the attributes has given better accuracy results for many datasets. Hence, by increasing the level of interaction among the attributes we can achieve performance gains. But we also need to take into consideration the complexity of the model. The NB model is very simple and less complex. Thus, for datasets that do not have correlations between variables, NB model is the best. Using the TAN model for such datasets is a waste of both time and space, as it will not give better results than the NB model and is also more complex when compared to the NB model.

From the comparative analysis of the algorithms on multiple datasets, we observe that the gain from TAN over NB is considerable only in some data sets. Determining which data sets will benefit from TAN without running the TAN training algorithm can be a useful feature. Also, root selection is done by randomly picking a node. It needs to be explored if there is potential for performance gain with improved root selection mechanism.

There is always a trade of between the performance gain and complexity of classifiers. If the interaction among the attributes increases, the complexity of the model also increases. But higher level of interaction might not always lead to better performance results. For example, considering that all the attributes of a system are correlated may not be useful because, in reality, not all the attributes of the system are correlated. So connecting irrelevant attributes may not give good performance results, worth the increase in complexity. It might sometimes lead to worse performance results. Thus, finding the optimal level of interaction among attributes for various systems becomes necessary to model better classifiers.

REFERENCES

[1] Koller, D. Probabilistic Graphical Models. Coursera Stanford – Lectures. Retrieved December 6, 2013, from https://class.coursera.org/pgm/lecture

[2] Koller, D., & Friedman, N. (2009). Probabilistic graphical models: Principals and techniques MIT Press. ISBN 978-0262013192

[3] Koller, D., Friedman, N., Getoor, L., & Taskar, B. (2007). Graphical models in a nutshell. In L. Getoor, & B. Taskar (Eds.), An introduction to statistical relational learning () MIT Press.

[4] Data Mining Intro. Retrieved April 13, 2014, from http://www.cs.bu.edu/fac/gkollios/ada05/LectNotes/lect19-05.ppt

[5] Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers.

[6] Machine Learning : Naive Bayes Classifier. Retrieved April 10, 2014, from http://computersciencesource.wordpress.com/2010/01/28/year-2-machine-learning-Naive-Bayes-classifier/

[7] Jurafsky, Dan and Manning, Christopher. Natural Language Processing. Coursera Stanford - Lectures. Retrieved April 5, 2014, from https://class.coursera.org/nlp/lecture/28

[8] Naive Bayes for classifying Text. Retrieved April 5 , 2014, from http://www.cs.nyu.edu/faculty/davise/ai/BayesText.html

[9] Chow, C.K. & C.N. Liu (1968). Approximating discrete probability distributions with dependence trees. IEEE Trans. on Info. Theory, 14, 462- 467.

[10] KEEL-dataset citation paper: J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17:2-3 (2011) 255-287.

[11] Cross-Validation (Statistics). (n.d). In Wikipedia. Retrieved April 5, 2014, from http://en.wikipedia.org/wiki/Cross-validation_(statistics).

[12] UCI Machine Learning Repository. Pima Indians Diabetes Data Set. Retrieved April 10, 2014, form http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

[13] Random variable. (n.d). In Wikipedia. Retrieved April 5, 2014, from http://en.wikipedia.org/wiki/Random_variable