Tree-Augmented Naïve Bayes Methods for Real-Time Training and Classification of Streaming Data

By
Kunal SinghaRoy

March 8th, 2018

A master project submitted to the
Department of Computer Science
In partial fulfillment of the requirements for the
Degree of Master of Science

Major Professor: Dr. Robert van Engelen
Dr. Robert A. van Engelen
Major Professor

Dr. Michael Mascagni
Committee Member

Dr. Shayok Chakraborty
Committee Member
Acknowledgements

First and foremost, I would like to thank my project advisor Dr. Robert van Engelen for his constant guidance and encouragement without which this project would not have been possible. His constant new ideas about how to improve the project worked as a motivating factor to achieve more from this project.
I would also like to thank my family and friends who always stood beside me and inspired me to work on things that I believe in.
Finally, I would like to thank my committee members Dr. Michael Mascagni and Dr. Shayok Chakraborty for their time and support.
# Table of Contents

Abstract ................................................................................................................................. 6

Chapter 1 ................................................................................................................................. 7
  1.1 Bayes’ Theorem: ............................................................................................................... 7
  1.2 Conditional Probability: ................................................................................................. 7
  1.3 Marginal Probability and Joint Probability: ............................................................... 8

Chapter 2 ................................................................................................................................. 11
  2.1 Bayesian Networks: ....................................................................................................... 11
  2.2 Complexity of Bayesian Networks: ........................................................................... 16

Chapter 3 ................................................................................................................................. 18
  3.1 Naïve Bayes Model ........................................................................................................ 18
  3.2 Naïve Bayes Model Example ....................................................................................... 19
  3.3 Laplace Correction: ..................................................................................................... 21
  3.4 Naïve Bayes Training and Classification .................................................................... 22
  3.5 Flowchart For Naïve Bayes Model ............................................................................ 23
  3.6 Complexity of Naïve Bayes Training and Classification ........................................... 27

Chapter 4 ................................................................................................................................. 29
  4.1 Machine Learning Methods for Streaming Data ......................................................... 29

Chapter 5 ................................................................................................................................. 31
  5.1 Applying Naïve Bayes Models to Streaming Data ....................................................... 31
  5.2 Flowchart for Naïve Bayes Model Applied to streaming data ................................... 31
  5.3 Complexity of the Naïve Bayes Model Applied to Streaming Data ......................... 36

Chapter 6 ................................................................................................................................ 38
  6.1 Tree-Augmented Naïve Bayes ..................................................................................... 38
  6.2 The TAN model using max abs error ......................................................................... 47

Chapter 7 ................................................................................................................................ 52
  7.1 Batch Algorithm Approach: ....................................................................................... 52
  7.2 Incremental TAN Learning ......................................................................................... 56

Chapter 8 ................................................................................................................................ 65
  8.1 Experimental Results ................................................................................................... 65
  8.2 Data Preparation ........................................................................................................... 65
  8.3 Brief Descriptions of Datasets used for testing ........................................................... 65

4
8.4  Performance evaluation of NB and TAN Classification .......................................................... 66

8.5  Performance result of different models .................................................................................. 66

Chapter 9 ......................................................................................................................................... 70

References: ........................................................................................................................................ 72
Abstract

Naïve Bayes (NB) and Tree Augmented Naïve Bayes (TAN) are simple probabilistic classifiers that are based on the application of Bayes’ Theorem. Popular applications of NB classifiers can be found in email spam Detection, face recognition, medical diagnosis, and weather prediction. In classification problem in general, we attempt to determine the category to which a new instance belongs by using knowledge acquired from analyzing previous instances (i.e. training set) for which the category was already known. This form of learning is generally referred to as supervised learning. The NB classifier assumes the value of an attribute is independent of the value of another attribute given the class variable. This is a very simplistic approach to build a probabilistic model which does not always hold true. If we can incorporate the information of correlation between the attributes within the Naïve Bayes model, the classification accuracy can be improved. Tree Augmented Naïve Bayes (TAN) is one such model. In Tree Augmented Naïve Bayes, the attributes are not independent as in NB, but the level of interaction between the attributes is still limited in TAN to keep the computational cost down. During TAN classification, each attribute conditionally depends on the class and one other attribute from the feature set. Therefore, a TAN is usually more realistic model which can increase the performance of classification without increasing the cost of training significantly. With the advent of internet there is continuous flow of data. We want our classification model to learn from this flow of non-stationary data. The main component in the TAN model is the tree structure. If we have to restructure the tree whenever a new training instance is fed into the model it will use significant processing power. Whereas if the tree structure does not reflect the correlation among the attributes correctly it will reduce the classification accuracy. In this project I implemented the TAN model on streaming data in such a way that whenever a new training instance is fed into model, metrics are used to access whether the current tree is outdated and may negatively affect the accuracy. I reviewed several different approaches for such metrics to decide when to restructure the tree.
Chapter 1

1.1 Bayes’ Theorem:

In probability theory, Bayes’ Theorem calculates the probability of an event based on the previous knowledge of conditions that might be related to the event. Bayes’ Theorem is mathematically stated using the following equation:

\[ P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \]  

[1]

where \( A \) and \( B \) are events and \( P(B) \neq 0 \).

\( P(A|B) \) is a conditional probability: the likelihood of event \( A \) given event \( B \) occurs.

\( P(B|A) \) is a conditional probability: the likelihood of event \( B \) given event \( A \) occurs.

\( P(A) \) and \( P(B) \) are marginal probabilities: the probabilities of observing \( A \) and \( B \) independent of each other.

Take, for instance, 10% of patients have obesity i.e. the prior probability of having obesity is \( P(\text{Obesity})=0.10 \). Among obese patients, 7% are sedentary i.e. \( P(\text{Sedentary}|\text{Obesity})=0.07 \). Also, given, 5% of the patients are sedentary i.e. \( P(\text{Sedentary})=0.05 \). Using Bayes’ Theorem, we can calculate the probability of obesity given sedentary:

\[ P(\text{Obesity}|\text{Sedentary}) = \frac{P(\text{Sedentary}|\text{Obesity}) * P(\text{Obesity})}{P(\text{Sedentary})} \]

\[ = \frac{0.07\times0.10}{0.05} \]

\[ = 0.14 \]

1.2 Conditional Probability:

The conditional probability of an event is expressed as the probability of an event to occur given another event occurs. For an example if we are interested in the probability of event \( A \), and event \( B \) has occurred then the “conditional probability of \( A \) given \( B \)” is mathematically expressed as \( P(A|B) \) [2]. The probability of a person having fever may be considerably low but if we already observed that the person is having malaria then the probability of the person having fever is considerably higher.

The conditional probability formula can be expressed in mathematical form as follows:

\[ P(A|B) = \frac{P(A \cap B)}{P(B)} \]

Where \( P(A \cap B) \) is the joint probability of events \( A \) and \( B \) and \( P(B) \) is the probability distribution of event \( B \).
P(A|B) may or may not be equal probability of P(A) i.e. unconditional probability of event A. When the conditional probability value is equal to unconditional probability value then the two events are called independent of each other. So, the occurrence of event A does not affect the probability of the event B and vice versa. From the concepts of conditional probability, the Bayes’ Theorem can be derived. By the equation of conditional probability. The probability of event A given event B is expressed as:

\[ P(A|B) = \frac{P(A \cap B)}{P(B)} \quad \text{given } P(B) \neq 0 \]  

The probability of event B given event A is expressed as:

\[ P(B|A) = \frac{P(B \cap A)}{P(A)} \quad \text{given } P(A) \neq 0 \]

Or, \[ P(B\cap A) = P(B|A) * P(A) \]

Now putting the value of equation (ii) in equation (i) we get:

\[ P(A|B) = P(B|A) * P(A)/P(B) \] which is the statement of Bayes’ Theorem.

In the field of probability theory, the chain rule allows to calculate the joint probability distribution of any number of random variables using only the conditional probabilities. [3]

Let \( n \) events be indicated by \( A_1, A_2, A_3, \ldots, A_n \). Then the joint probability distribution of the events \( A_1, A_2, A_3, \ldots, A_n \) can be expressed with the chain rule as follows:

\[ P(A_1, A_2, A_3, \ldots, A_n) = P(A_1) * P(A_2|A_1) * P(A_3|A_1, A_2) * \ldots * P(A_n|A_1, A_2, A_3, \ldots, A_n) \]

### 1.3 Marginal Probability and Joint Probability:

In terms of probability theory and statistics, the marginal probability is the probability of an event occurring without taking into consideration the other event(s). That is, marginalizing the other event(s). For example, when we are flipping a coin the probability of getting head is considered marginal probability because we are not considering the other events that may affect the outcome. [4]

In probability theory and statistics, joint probability is the probability of two or more different events observed together. For example, suppose a dice is rolled. Let A and B be two discrete random variables associated with the outcome of the event. A takes up value 1 when the dice rolled is an even number (2,4,6) and 0 otherwise. B takes up value 1 when the number rolled is prime (2,3,5) and 0 otherwise.

<table>
<thead>
<tr>
<th>A/B</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
In this case, joint distribution of A and B expressed in terms of probability mass function is as follows:

\[
P(A=0, B=0) = P\{1\} = 1/6 \\
P(A=1, B=0) = P\{4,6\} = 1/6 + 1/6 = 2/6 \\
P(A=1, B=1) = P\{2\} = 1/6 \\
P(A=0, B=1) = P\{3,5\} = 1/6 + 1/6 = 2/6
\]

Given a set of random variables \(X_1, X_2, X_3, \ldots, X_n\) the joint probability distribution can be expressed as \(P(X_1, X_2,\ldots,X_n)\). The distribution takes up probability values assigned by all the states that are represented by the random variables. One important aspect of joint probability distribution function of random variables is that it must be consistent with the marginal distribution of those variables. The marginalization rule is expressed as follows:

\[
P(x) = \sum_y P(x,y)
\]

Let’s take for example the probability of a person being hit by a car while crossing the road at pedestrian crossing without paying attention to the traffic light. There are two random variables, \(H\) which takes up values \{Hit, Not Hit\} and \(L\) which is color of traffic signal which takes up values \{Read, Yellow, Green\}. [4]

In practical scenario the probability of getting hit by a car is dependent on the color of the traffic light. That is, \(H\) is dependent on \(L\). A person is more likely to get hit by a car when the person tries to cross the road when the color of the light is green than when the color of the light is red. For any give pair of values of \(H\) and \(L\) the joint probability distribution of \(H\) and \(L\) must be considered to find out the probability of both the events happening together.

In calculating the marginal probability of being Hit we are ignoring the color of the light of the traffic signal. The \(P(H=Hit)\) can happen when the color of the light is red, green or yellow. So, the marginal probability can be calculated by summing up all the values for \(P(H=Hit|L)\) for all possible values of \(L\) where each value of \(L\) is weighted by its occurring.

The conditional probability table \(P(H|L)\) is as follows:

<table>
<thead>
<tr>
<th></th>
<th>Green</th>
<th>Yellow</th>
<th>Red</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit</td>
<td>0.90</td>
<td>0.10</td>
<td>0.02</td>
</tr>
<tr>
<td>Not Hit</td>
<td>0.10</td>
<td>0.90</td>
<td>0.98</td>
</tr>
</tbody>
</table>

The sum of probability across each column is 1 because sum of probability of being hit and not hit is 1 regardless of the state of the light.

Now, let the probability of having red light \(P(L=Red) = .2\) and yellow light \(P(L=Yellow) = .1\) and green light \(P(L=Green) = .7\) respectively.

Then the joint probability distribution can be calculated as:

\[P(H, L):\]
<table>
<thead>
<tr>
<th></th>
<th>Green</th>
<th>Yellow</th>
<th>Red</th>
<th>Marginal Probability P(H)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit</td>
<td>0.63</td>
<td>0.01</td>
<td>0.004</td>
<td>0.644</td>
</tr>
<tr>
<td>Not Hit</td>
<td>0.07</td>
<td>0.09</td>
<td>0.196</td>
<td>0.356</td>
</tr>
<tr>
<td>Total</td>
<td>0.70</td>
<td>0.10</td>
<td>0.200</td>
<td></td>
</tr>
</tbody>
</table>

As we can see, the value of the marginal probability is the sum of the value of the joint probability distribution for all possible values of the random variable in this case i.e. the color of the light being red, green or yellow.
Chapter 2

2.1 Bayesian Networks:

Bayesian networks (BN) are also known as Belief Networks. BN are probabilistic graphical models that are used to represent information in uncertain domain. It is based on the random variables of the domain and the interactions between them. Every node in the graph represents a random variable and the edges between the random variables represents the dependencies between those corresponding variables. In the graph the conditional dependencies are often calculated using graphical and statistical principles. Hence, BN combines the principles from graph theory, probability theory and statistical theory. [6]

Let us take an example that illustrates the details about an athlete in a national competition. The different random variables which are used in this example are skill, difficulty, rank and score.

i. *Fitness* denotes the skill of the players of the team achieved through practice. It can take two values → High – f¹ and Low – f⁰.

ii. *Difficulty* denotes the difficulty faced by the athlete in the competition. It can take two values → Hard – d¹ and Easy – d⁰.

iii. *Rank* refers to the position of the athlete in the competition. It can take three values → 1st – r¹, 2nd – r², 3rd – r³.

iv. *PFT (Physical Fitness Test)* refers to score achieved by the athlete in the standardized physical fitness test. It can take two values → High – p¹ and Low – p⁰.

v. *Certificate* refers to the Certificate that the athlete receives from Government based on his performance. It takes values → good & bad.

The marginal distribution of the team skill and difficulty from opponent team is illustrated in the Table 1.

Table 1: Marginal Probability distribution of difficulty and skill in the team.

<table>
<thead>
<tr>
<th>D</th>
<th>P(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>d⁰</td>
<td>0.6</td>
</tr>
<tr>
<td>d¹</td>
<td>0.4</td>
</tr>
</tbody>
</table>
The joint distribution over the rank and fitness of the athlete is illustrated in the Table 2.

**Table 2:** Joint Probability distribution over rank and fitness of the athlete.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Fitness</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>0.19</td>
<td>0.09</td>
</tr>
<tr>
<td>2&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>0.07</td>
<td>0.25</td>
</tr>
<tr>
<td>3&lt;sup&gt;rd&lt;/sup&gt;</td>
<td>0.04</td>
<td>0.36</td>
</tr>
<tr>
<td>Total</td>
<td>0.30</td>
<td>0.70</td>
</tr>
</tbody>
</table>

In Table 2, the Summation of the probabilities of the random variable Fitness across all values (i.e. High & low) is equal to the marginal distribution of Fitness. Similarly, when the probabilities of the random variable Rank across all values (i.e. 1<sup>st</sup>, 2<sup>nd</sup> & 3<sup>rd</sup>) is added then the sum equals to the marginal distribution of Rank. In a joint probability distribution, the sum of all the entries in the distribution must add up to 1. The same is happening in Table 2.
The joint probability distribution gives the probability of occurrence of all possible combinations of all possible values of all the random variables in the system. Given a joint probability distribution, we are often interested in the prior probability of a variable or the conditional probability of the variable given the states of one or more other variables in the system. From a computational point of view, this cannot be so easily achieved. Take for instance $N$ random variables associated with a domain of discernment. Each of these random variables can take binary values (0,1). The joint distribution in that case would contain $2^N$ entries. In practical world, the value of $N$ can be very large and each of these random variables will take many possible values. So, it will be very inefficient to build such a huge joint distribution table which will take a considerable computing power as well as the memory utilization will also be inefficient as only specific part of the table will be used frequently.

In order to solve this problem more efficiently, the dependencies between the random variables should be taken into account and the independent random variables should be identified. If two random variables do not depend on each
other no additional information can be derived from their joint probability distribution. So, we can ignore those joint probability entries of the independent random variables which will reduce the overall complexity.

One of the simple and efficient ways of representing dependencies between random variables is using graphs. In the graphs, each random variable is represented by a node and the edges between the nodes represent the dependencies among them. There are two types of graphs used with probabilistic graphical models: directed and undirected.

A Markov network is a graphical model with undirected edges. This network represents independence between two distinct nodes using the concept of Markov Blankets [6]. Undirected graphical models are used extensively in the field of computer vision and statistical physics.

BNs are probabilistic graphical models based on directed acyclic graphs (DAG). It is mathematically rigorous and intuitively understandable [6]. It represents an easy to understand representation and efficient way of calculating joint probability distribution over the set of random variables. It is used extensively in the field of machine learning, statistics and artificial intelligence. In following paragraphs, I will focus on BNs.

The directed acyclic graph of a BN consists of two sets: a set of nodes and set of directed edges. The random variables of the system form the set of nodes in the graph. The directed edges between nodes constitute the dependences between variables. A directed edge from node $X_i$ to node $X_j$ means that the variables are dependent, or in other words that $X_i$ influences $X_j$. The node with variable $X_i$ is called the parent of $X_j$ and similarly $X_j$ is called the child of $X_i$. The set of descendants, is the set of nodes that can be reached in a direct path from that node. The set of ancestors is the set of nodes from which there is a direct path to the node.

A BN is a DAG. No cycles are allowed in the graph. The main principle on which factorization of a joint probability of collection of nodes is based on the DAG structure that represents conditional dependences between variables modeled by the directed edges. Even though the arrows represent the direct causal relationship between two nodes, information can flow in any direction based on the reasoning process [7]. A BN identifies the conditional independence between random variables. Each variable is independent of its non-descendants in the graph given the state of its parent [6]. This property helps to significantly reduce the number of parameters that are required to characterize the joint probability distribution of the variables. This makes the calculation of the posterior probabilities more efficient.

The Bayes net assumption says:

“Each variable is conditionally independent of its non-descendants, given its parents” [22].
It is possible to explain independence using this statement, but d-separation is a more formal process to determine independence [22]. Let’s explain this with the help of three variables X, Y & Z. Using d-separation we would check in different scenarios whether X & Y are conditionally independent given Z.

Causal Trail:
X → Z → Y => X Y (moralizing and then deleting given Z)
Here X and Y are conditionally independent given Z as they are not connected.

Evidential Trail:
X ← Z ← Y ⇒ X Y (moralizing and then deleting given Z)
Here Y and X are conditionally independent given Z as they are not connected.

Common Cause:
X ← Z → Y ⇒ X Y (moralizing and then deleting given Z)
Here X and Y are conditionally independent given Z since they are not connected to each other anymore.

Common Effect:
X → Z ← Y ⇒ X – Y (moralizing and then deleting given Z)
Here Y and X are not conditionally independent given Z as they are connected via Z.

So, we can conclude that if two variables are d-separated then they are conditionally independent. But if they are not d-separated then they are not necessarily dependent.

Along with the directed acyclic graph there is need of specifying the quantitative parameters of the model. Following the principles satisfying Markovian property, the conditional probability distribution in each of the node only depends on its parents. In case of discrete random variables, the table consist of local probability of each possible value of the local parameter to the combination of each possible value of its parent. The local conditional probability tables can be used to determine the joint probability distribution of set of variables.

Using the above observations, a formal definition of BN can be formulated. A BN B is a directed acyclic graph that represents the reduced joint probability distribution over a set of random variables V [6]. The network B is defined by a pair B = <G, Θ>, where G represents the directed acyclic graph which consist of set of nodes X₁, X₂,….,Xₙ which are the random variables, and there is set of edges which shows the dependencies between those random variables. Using the graph G, the independence between the random variables can be easily inferred. Each variable Xᵢ is independent of its non-descendant given its parents in G. Θ denotes the set of parameters within the network. This contains the parameter θᵢ|Πᵢ = P_B(xᵢ|Πᵢ) for each value of xᵢ of the random variable Xᵢ conditioned on Πᵢ, the set
of parents of $X_i$ in graph $G$ [6]. So, $B$ defines a unique joint probability distribution over $V$ given as follows:

$$P_B(X_1, X_2, \ldots, X_n) = \prod_{i=1}^n P_B(X_i | \pi_i) = \prod_{i=1}^n \theta_i X_i | \pi_i$$

If node $X_i$ has parents, it is called a conditional node, otherwise it is called an unconditional node. If the variable is represented by an observable node it is called an evidence node, otherwise it is called a hidden node. Nodes can serve as inputs (evidence) and any node can also serve as output, based on the way BN inference algorithms exploit Bayes’ theorem to compute prior and posterior probabilities of variables given some evidence.

### 2.2 Complexity of Bayesian Networks:

The determination of interdependence between the random variables in a domain is exponential if the structure of the BN is unknown. In fact, inference with BN is NP-hard [https://www.sciencedirect.com/science/article/pii/00437029390036B] and even approximate inference to obtain a fixed degree of accuracy is NP-hard[https://www.sciencedirect.com/science/article/pii/00437029390036B]. As there is no knowledge about the correlation between the random variables the joint probability distribution cannot be reduced. Hence, in a BN classification model there is need to determine the joint probability distribution conditioned on class. The classification model is defined by following formula:

$$P(C|X_1 \ldots X_N) = \frac{P(X_1 \ldots X_N | C) \times P(C)}{P(X_1 \ldots X_N)}$$

During classification we need to find the maximum likelihood class that is the class value with maximum probability. As the denominator remains the same for the calculation of the probability for each class value, so we can ignore it and use the highest classification value to determine the most likely class (or we can weigh the classification results instead if probabilities are required).

The values of $P(X_1 \ldots X_N)$ must be calculated and stored for each possible combination of values of random variables and class labels. This values can be then used for calculation of $P(C|X_1 \ldots X_N)$.

Let, $N$ be number of attributes in the data set. Let, $C$ be number of class labels in the data set. Let each attribute can take only discrete values.
Let \( \{A_1, \ldots, A_N\} \) be number of possible attribute values for each of the attributes. Let \( A_{\text{max}} \) be the maximum of the \( \{A_1, \ldots, A_N\} \).

The joint probability distribution of \( X_1 \ldots X_N \) needs to be calculated conditioned on the class. Take for instance, if \( N=2 \) and one attribute can take 5 values and the other attribute can take 6 values then we have 30 combinations. So, total number of values that needs to be calculated is 30 * C.

Assuming, counts for each possible combination can be done by one scan through the entire training data set the total number of computations required:

\[
\text{Total Computations} = (A_1 \cdot A_2 \cdot \ldots \cdot A_N) \cdot C 
\leq A_{\text{max}}^N \cdot C 
= O(A_{\text{max}}^N \cdot C)
\]

Therefore, the complexity of calculating the joint probability distribution is exponential. In practical world we need to impose some restriction between the interaction of the variables in order to reduce the complexity of the calculations.

The following sections focus on reducing the complexity of the BN by imposing restrictions on the interactions between the random variables and study how it affects the performance of the classification model.
Chapter 3

3.1 Naive Bayes Model

The NB model is a popular classification technique with certain limitations. It is a very narrow and specialized form of Bayesian network. In the NB model the assumption is random variables are independent of each other. The diagram of structure of NB model is given below:

![Diagram of Naive Bayes Model](image)

Here, C represents the class label which can hold all the possible class values \{C_1,..,C_N\} and X_1,X_2,…..,X_N represents all the attributes or the random variables associated with the system. Given the class label in an NB model as evidence, all the attributes are independent of each other. This significantly reduces the complexity of inferring probabilities with NB models. However, it is simplistic to assume that attributes are unrelated to each other. Take for instance a person decides to play soccer based on the weather conditions on that day. Let the random variables involved in the system be outlook, temperature, humidity and wind. The class variable in this case takes up two values yes and no, which predicts whether the person will play soccer on that day. As, the random variables are independent of each other the joint distribution reduces to:

\[ P(X_1,…..,X_N|C) = P(C) * \prod_{i=1}^{n} P(X_i|C) \]

In reality though the random variables are related to each other. If the outlook is sunny then the temperature is most likely to be high, if the outlook is rainy the temperature is most likely to be lower. In same way if humidity is high then the
outlook is most likely to be sunny or overcast. So, the assumption that the random variables are independent of each other is not correct. Despite its strong assumption of independence, the NB model has been shown to perform well for many practical problems.

In the NB model during training, the conditional probabilities of each of the random variable given the class label is calculated. During the classification, the posterior probabilities of each of the class label is calculated by calculating the product of the prior probability of the class label and the conditional probability of the attribute given the class. According to Bayes’ rule:

\[ P(C|X_1\ldots X_N) = \frac{P(C) \times P(X_1\ldots X_N|C)}{P(X_1\ldots X_N)} \]

The class label with maximum probability value is assigned as the prediction for that instance. For any test record, the denominator \(P(X_1\ldots X_N)\) is used for calculation of probability for each of the class label.

### 3.2 Naive Bayes Model Example

Let us consider a text classification problem using NB. In this problem we are given a document that we need to classify to determine which category the document falls under. In formal words given problem can be stated as: Given a collection of documents \(D_1,\ldots, D_N\) which belongs to category \(C_1,\ldots, C_N\) and vocabulary \(V\) that contains all the words that can appear in those documents. Now, when a new document \(D\) is given, we want to find out which category the particular document belongs to.

The occurrence of a particular word in a document does not influence the presence of another word in the document as per independence assumption of the NB model. There is a random variable representing each word in the vocabulary. It takes two values 0 and 1, where 1 represents the word present in the document and 0 representing the word is absent in the document.

During training the classifier calculates the prior probability of each of the class label.

Let, \(N_C\) is the count of the number of training data with particular class label

Let, \(T\) be the total number of training instances.

Then, prior probability \(P(C) = \frac{N_C}{T}\)
During training it calculates the conditional probability of each of the word given the class label.
Let \( N_i \) be the count of number of records with class label \( C \) in which word \( W_i \) appears.
\( N_C \) be the total number of documents of class \( C \).

Then, \( P(W_i|C) = \frac{N_i}{N_C} \)

Let, the dataset for training be as follows:

<table>
<thead>
<tr>
<th>Document</th>
<th>Text</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>“A great game”</td>
<td>Sports</td>
</tr>
<tr>
<td>D2</td>
<td>“The election was over”</td>
<td>Not sports</td>
</tr>
<tr>
<td>D3</td>
<td>“Very clean match”</td>
<td>Sports</td>
</tr>
<tr>
<td>D4</td>
<td>“A clean but forgettable game”</td>
<td>Sports</td>
</tr>
<tr>
<td>D5</td>
<td>“It was a close election”</td>
<td>Not sports</td>
</tr>
</tbody>
</table>

[8]

Based on the training data we can create the conditional probability table as follows:

| Word       | \( P(\text{Word}|\text{Sports}) \) | \( P(\text{Word}|\text{Not Sports}) \) |
|------------|-----------------------------------|---------------------------------------|
| A          | 2/3                               | 0/2                                   |
| Great      | 1/3                               | 0/2                                   |
| Game       | 2/3                               | 0/2                                   |
| The        | 0/3                               | \( \frac{1}{2} \)                     |
| Election   | 0/3                               | 2/2                                   |
| Was        | 0/3                               | 2/2                                   |
| Over       | 0/3                               | \( \frac{1}{2} \)                     |
| Very       | 1/3                               | 0/2                                   |
| Clean      | 2/3                               | 0/2                                   |
| Match      | 1/3                               | 0/2                                   |
| But        | 1/3                               | 0/2                                   |
| Forgettable| 1/3                               | 0/2                                   |
| It         | 0/3                               | \( \frac{1}{2} \)                     |
| Close      | 0/3                               | \( \frac{1}{2} \)                     |

Now, probability of sports category: \( P(\text{Sports}) = \frac{3}{5} \)
Probability of Not sports category: \( P(\text{Not Sports}) = \frac{2}{5} \)

Some of the conditional probability values are zero. This means that when we will try to calculate the posterior probability using those conditional probability values, it will evaluate to zero. This may lead to incorrect classification (as in that case) even though the other words probability for that particular class may be higher. This type of situation is handled using Laplace smoothing which is described below.

### 3.3 Laplace Correction:

Sometimes a given class and feature do not appear together in the training data set, as a result the frequency-based probability calculation in those cases will turn out to be zero. This is a hindrance because it will nullify the effects other probabilities when it is multiplied.

As we can see \( P(\text{over}|\text{Sports}) = 0 \), but if the test document contains sentence “game is over” the \( P(\text{Sports}|\text{game is over}) \) becomes equal to zero which in turn will lead to a wrong classification.

In order to accommodate the unobserved instances, we can assign a small probability value by using small sample correction called pseudocount in all probability estimates [9]. It can be achieved by adding small constant value to the count of values and adjusting the denominator accordingly. This is called Laplace Smoothing.

Conditional probability after Laplace correction:

| Word     | \( P(\text{Word}|\text{Sports}) \) | \( P(\text{Word}|\text{Not Sports}) \) |
|----------|------------------------------------|----------------------------------------|
| A        | \( \frac{3}{4} \)                | \( \frac{1}{3} \)                     |
| Great    | \( \frac{2}{4} \)                | \( \frac{1}{3} \)                     |
| Game     | \( \frac{3}{4} \)                | \( \frac{1}{3} \)                     |
| The      | \( \frac{1}{4} \)                | \( \frac{2}{3} \)                     |
| Election | \( \frac{1}{4} \)                | \( \frac{3}{3} \)                     |
| Was      | \( \frac{1}{4} \)                | \( \frac{3}{3} \)                     |
| Over     | \( \frac{1}{4} \)                | \( \frac{2}{3} \)                     |
| Very     | \( \frac{2}{4} \)                | \( \frac{1}{3} \)                     |
| Clean    | \( \frac{3}{4} \)                | \( \frac{1}{3} \)                     |
| Match    | \( \frac{2}{4} \)                | \( \frac{1}{3} \)                     |
But  2/4  1/3  
Forgettable  2/4  1/3  
It  1/4  2/3  
Close  1/4  2/3  

**Classification:**

Now, given a document containing text “a very close game” we need to classify which category it belongs to.

\[
P(\text{Sports}| \text{a very close game}) = P(\text{Sports}) \times P(\text{a}|\text{Sports}) \times P(\text{very}|\text{Sports}) \times P(\text{close}|\text{Sports}) \times P(\text{game}|\text{Sports})
\]

(Ignoring the denominator as we are only concerned with finding maximum value)

\[
= \frac{3}{5} \times \frac{3}{4} \times \frac{2}{4} \times \frac{1}{4} \times \frac{3}{4}
\]

\[
= 0.04219
\]

\[
P(\text{Not Sports}| \text{a very close game}) = P(\text{Not Sports}) \times P(\text{a}|\text{Not Sports}) \times P(\text{very}|\text{Not Sports}) \times P(\text{close}|\text{Not Sports}) \times P(\text{game}|\text{Not Sports})
\]

\[
= \frac{2}{5} \times \frac{1}{3} \times \frac{1}{3} \times \frac{2}{3} \times \frac{1}{3}
\]

\[
= 0.00987
\]

As Sports class label has maximum posterior probability among all the other class labels, so the test document is classified as Sports.

**3.4 Naïve Bayes Training and Classification**

Following are the steps that needs to be followed during the implementation of Naïve Bayes classification [10]:

**Initialization:**

C \leftarrow \text{List of all possible class values}

len(C) \leftarrow \text{The number of class values possible}

listOfAttributes \leftarrow \text{List of all the attributes for the dataset i.e. all the random variables associated with this dataset.}

possibleAttributeValues \leftarrow \text{It is a list of lists where possibleAttributeValues[i] contains the list of all the possible values that attribute can take}

len(possibleAttributeValues[i]) \leftarrow \text{Number of attribute values that particular attribute can take}

T \leftarrow \text{Total number of records in the training dataset.}
L_c \leftarrow \text{Laplace Constant}

\text{train} \leftarrow \text{List of all the training records. Each record containing attribute values for random variables } X_1, \ldots , X_N.

\text{test} \leftarrow \text{List of all the test dataset. Each record containing attribute values for random variables } X_1, \ldots , X_N.

\textbf{Computation}

\textbf{Training:}

For each c in C:
- \text{N}_c = \text{Count of class label } c \text{ in the training dataset.}
- \text{priorProbability}[c] = (\text{N}_c + L_c)/(T+(L_c * \text{len}(C)))

For each i in listOfAttributes:
- For each a_i in possibleAttributeValues[i]:
  - \text{C}_{a,c} = \text{Count of number of records in training dataset which contains combined occurrences of value } a_i \text{ for attribute } i \text{ and class label } c.
  - \text{condProbability}[i=a_i|c] = \frac{v(C_{a,c}+L_c)}{(N_c+(L_c*\text{Len(possibleAttributeValues[i])})}

return priorProbability, condProbability

\textbf{Testing:}

For each r in test:
- For each c in C:
  - \text{posteriorProbability}(C|[x_1 x_2 \ldots x_N]) = \text{prior}[c] * \prod_{i=1}^{N} \text{condProbability}(i = x_i|C = c)
  - \text{predictedClass} = \text{max}\{\text{posterior}[c]\}

\textbf{3.5 Flowchart For Naïve Bayes Model}

In the NB model prior probability calculation, conditional probability calculation and testing are the most important steps. During the initialization phase train data and test data are separated and read. Following are the flowchart representation of important steps in classification using NB.
Flowchart for prior probability calculation:

Start

C[1 … N] = List of all possible Class Labels.
Lc = Laplace Constant
T = Total no. of Test records

For i = 1;
i<=len(C); i++

Nc = Count of C[i] in test data

prior[C] = (Nc+Lc)/(T+Lc*len(C))

Return priorProbability
Flowchart for Conditional probability calculation:

```
Start

C[1 ... N] = List of all possible Class Value.
A[1 ... L] = List of all attributes.
poss[i] = Attribute values possible for particular attribute i
Lc = Laplace Constant

For i = 1; i<=len(C);
  i++

For j = 1; j<=len(A);
  j++

For k = 1;
  k<=len(poss[j]);
  k++

N_{j=a,c} = Count of combined instances where class value i= c and attribute value j = poss[j][k]

prob = (N_{j=a,c} + Lc)/(Nc + (Lc * len(poss[j])))
and store it in condProbability

Return condProbability
```

25
Flowchart for Testing:

Test[1 … R] = List of all Test records.
C[1 … N] = List of all possible classes.
predictClass[1 … R] = List of predicted classes which is initialized.

For i = 1; i<=len(Test); i++
For j = 1; j<=len(C); j++

Posterior probability C[j] is calculated using formula
P(C|X_1 … X_M) = Prior(C) Π-condProbability(X_i=X|C[j])
for all X_i=X and for each X_i in [X_1 … X_M]

predictClass[i] = max(C[1 … N])

Return TestClass
3.6 Complexity of Naïve Bayes Training and Classification

Let $C$ be the total number of class labels possible for the system.
Let $N$ be the total number of attributes associated with the NB model.
Let $S_1, S_2, S_3, \ldots, S_N$ be the attributes associated with the system.
Let $A_1, A_2, A_3, \ldots, A_N$ be the number of attribute values possible for each of the attribute $S_1, S_2, S_3, \ldots, S_N$.
Let $A_{\text{max}}$ be the maximum of \{A_1, A_2, A_3,\ldots,A_N\}.
Let $T$ be the number of training instances
Let $R$ be the number of test instances.

Let us assume that the number of class labels and the count of each attribute state corresponding to given class label can be calculated in a single scan through the training instances. So, the time complexity for counting is $O(T)$.

During calculation of prior probability, probability value of each of the class label needs to be computed. So, the time complexity is $O(C)$.

During the calculation of conditional probability, we need to compute the probability of each attribute state for each of the attribute given the class. As, we have already assumed the count of each attribute state given a class is done in a single scan through the training dataset, the number of computations involved will be:

\[
\text{Total Number of Computations} = \sum_{i=1}^{N} (C \cdot A_i)
\]
\[
= C \cdot \sum_{i=1}^{N} A_i
\]
\[
\leq C \cdot (A_{\text{max}} + A_{\text{max}} + \ldots + A_{\text{max}})
\]
\[
\leq N \cdot C \cdot A_{\text{max}}
\]

So, the time complexity of conditional probability calculation is $O(NCA_{\text{max}})$.

During testing, we need to calculate the posterior probability for each of the class label using values from prior probability and conditional probability. Accessing values from prior probability and conditional probability can be done in constant time. For each testing record we need to find the product of conditional probability for each of the attribute state corresponding to the attribute value of the testing instance. So, the number of computations involved is $N$ as we need to multiply the product conditional probabilities with the prior probability. So, the time
complexity for classifying each test instance is $O(N^C)$. If we classify $R$ number of test instances the time complexity will be $O(N^CR)$.

As it can be observed the calculation of conditional probability is most computationally intensive step in the NB model. So, it dominates the overall time complexity in this model.
Chapter 4

4.1 Machine Learning Methods for Streaming Data

In traditional machine learning approach, the model generally deals with static dataset. At first the algorithm is fetched with training data. The algorithm analyzes that data and learns from it. Then the model is given testing data and it predicts the outcome based on the learning from training dataset.

In today’s highly connected world in which information evolves quickly this requirement is changing. Online data is arriving in a form of data stream. We should be able to train our model from the data stream as it arrives in leaps and bounds, so that we can use it intermittently for incremental training and actual classification, depending on whether or not the classifier value is known or not.

Suppose for example a website visited by a customer. New training data may actually arrive about the customer in real time as we learn more facts about the customer, which we want to absorb into our model. In addition to training events, we also want to use the classifier to make predictions about the customer (e.g. based on other customers data). In order for this to work, we need our model to be equipped to learn from the continuous flow of non-stationary data.

There is a lot of current research on making machine learning algorithms able to work on the streaming data. There are two different approaches to it:

**Incremental Algorithms:**
These are machine learning algorithms that learn incrementally over the data. That is, the model is updated each time it sees a new training instance [12].

**Periodic Retraining with a batch algorithm:**
It is the more straightforward solution. Here, we simply buffer the relevant data and retrain our model “every so often” [12].

The incremental algorithm can work in batch setting, but the opposite is not true, that is, batch algorithm cannot work in an incremental setting. The incremental algorithm will work in batch setting by passing the instances of the batch one by one to the model. Many batch algorithms can only be made to work incrementally with significant work or power sacrifices, and some things just can’t be done.
Incremental learning is advantageous in many cases. There is no buffering and no explicit retraining of the model, which can cause delays that are not acceptable for real-time prediction. With incremental learning the model remains up to date.
Chapter 5

5.1 Applying Naïve Bayes Models to Streaming Data

This is the simple incremental learning approach with the NB model applied to streaming data. We want our NB model to learn as the streaming data arrives. In this project we are mainly dealing with categorical data. So, implementing the NB model on streaming data is more or less similar to the implementation of the NB model. We use a batch training algorithm approach where we switch between training phase and testing phase according to the streaming data.

A contingency table is kept for each attribute state of each attribute given any class label. Also, there is another table that contains the frequency of each class label. At the beginning all the tallies are initialized to zero. As the streaming data arrives, the contingency table is updated.

When the testing data arrives, it switches from training mode to the testing mode. It first calculates the prior probabilities and the conditional probabilities. While calculating the probabilities we need to get the frequencies of attribute state given a class label which can be done in constant time as the count table is stored in a dictionary.

For each test record instance, the posterior probability is calculated. The class with maximum posterior probability is the predicted class.

5.2 Flowchart for Naïve Bayes Model Applied to streaming data

Following are the flowcharts representing important steps in applying the NB model to streaming data.
Flowchart of Naïve Bayes model during training on streaming data instance:

- **Start**
- **R[1 … N] = Training data instance**
- **Update Class count**
- **For i = 1; i<=len(A); i++**
  - **Update count of attribute state given class and store it in frequency map.**

The flowchart illustrates the process of updating the Naïve Bayes model during training on streaming data instances.
Flowcharts of Naïve Bayes for Switching between Training to Testing phase:

(a.)

```
Start

For i = 1; i<=len(C);
i++

prior[C] = C[i]/T
where T = no. of training instance

Return TestClass
```
Start

C[1 ... N] = List of all possible Class Value.
A[1 ... L] = List of all attributes.
poss[i] = Attribute values possible for particular attribute i
Lc = Laplace Constant
T = Total no. of Test records

For i = 1; i<=len(C); i++

For j = 1; j<=len(A); j++

For k = 1; k<=len(poss[j]); k++

Calculate conditional probability using count from frequency map and store it in cond Probability

Return condProbability
Flowchart for Testing Naïve Bayes Model on streaming data:

1. Start

2. Test[1 … R] = List of all Test records.
   C[1 … N] = List of all possible classes.
   predictClass[1 … R] = List of predicted classes which is initialized.

3. For i = 1; i<=len(Test); i++
   For j = 1; j<=len(C); j++
   
   - Posterior probability C[j] is calculated using formula
     \[ P(C|X_1 \ldots X_M) = \text{Prior}(C) \Pi \text{condProbability}(X_i=X|C[j]) \]
     for all \( X_i = X \) and for each \( X_i \) in \( \{X_1 \ldots X_M\} \)

4. \( \text{predictClass}[i] = \max(C[1 \ldots N]) \)

5. Return TestClass
5.3 Complexity of the Naïve Bayes Model Applied to Streaming Data

Let $C$ be the total number of class labels possible for the system.
Let $N$ be the total number of attributes associated with the model.
Let $S_1, S_2, S_3, \ldots, S_N$ be the attributes associated with the system.
Let $A_1, A_2, A_3, \ldots, A_N$ be the number of attribute values possible for each of the attribute $S_1, S_2, S_3, \ldots, S_N$.
Let $A_{\text{max}}$ be the maximum of $\{A_1, A_2, A_3, \ldots, A_N\}$.
Let $R$ be the number of test instances.

While training from streaming data, we need to update the class count table and the frequency table. The number of records that needs need to be updated in the frequency table is $N$. So, the total complexity while learning from streaming data is $O(N)$.

While transitioning from training to the classification phase, we need to calculate the prior probabilities and conditional probabilities. Calculation of prior probabilities will take $C$ computation as the counts of each class label can be accessed in constant time. Hence, the time complexity for calculation of prior probability is $O(C)$.

Now, to calculate the conditional probability we can access the frequency count for each attribute state given a class in constant time. So, the total number of computations needed for conditional probability calculation is as follows:

$$\text{Total Computations} = \sum_{i=1}^{N} (C \times A_i)$$
$$= C \times \sum_{i=1}^{N} A_i$$
$$\leq C \times (A_{\text{max}} + A_{\text{max}} + \ldots + A_{\text{max}})$$
$$\leq N \times C \times A_{\text{max}}$$

Hence, the complexity of conditional probability calculation is $O(NCA_{\text{max}})$.

During testing we need to calculate the posterior probability for each of the class label using values from prior probability and conditional probability. Accessing values from prior probability and conditional probability can be done in constant time. For each testing record we need to find the product of conditional probability for each of the attribute state corresponding to the attribute value of the testing instance. So, the number of computations involved is $N$ as we need to multiply the product conditional probabilities with the prior probability. Therefore, the time
complexity for classifying each test instance is $O(N^C)$. If we classify $R$ number of test instances the time complexity will be $O(NCR)$.

As it can be observed from switching between the training phase to the classification phase, the most computationally intensive step in the NB model applied to streaming data is the conditional probability calculation. It dominates the overall time complexity in this model.
Chapter 6

6.1 Tree-Augmented Naïve Bayes

In the NB model the assumption is that all the attributes are independent of each other given the class. In reality, the attributes are rarely (conditionally) independent of each other. In most cases attributes are directly or indirectly dependent on each other. Although this naïve assumption rarely holds up, the Bayesian Network (BN) model may still perform quite well. If we take into account some of the stronger (conditional) dependences between the attributes, the performance of NB classification can be improved [12] by moving closer to a BN DAG representation of the model. A joint probability distribution is accurately represented by a BN, but a full BN is costly to construct and costly for inference. The joint probability distribution may also be biased when it is not modeling complete data, i.e. it may only be suitable under closed-world assumptions. Therefore, approximations of BN often suffice [21].

In a BN we capture the interdependence between the random variables as well as the independence between them using a graph. Using a BN we can calculate the conditional probabilities of each random variable (node) given its parents. Each random variable is independent of its non-descendants given its parents. Given this, we can calculate the joint probability distribution using the local conditional probabilities taking into consideration independence assumption specified in the network. Although learning such a BN can be very costly and complex as the number of random variables associated can be very large and each random variable can take up large number of possible values. In [12] it is stated that "While the induction of the Naïve Bayesian classifier requires only simple book keeping (storing conditional probabilities given the label), the induction of BN requires searching the space of all possible networks, i.e., the space of all possible combination of edges." Also, single random variable can have many parents, so calculation of conditional probability will become more complex.

By contrast, the main parameter that determines the NB model classification is simply P(C|A_1A_2…..A_N) which takes into account the attributes associated with the system without their interdependencies. However, this may sometimes lead to lower classification performance [10]. For a better classification performance, we need an extension of the NB model that also includes some (conditional) dependencies between the random variables [12], but without constructing a full BN.
A better classification model can be achieved by augmenting the Naïve Bayes model. An augmented Naïve Bayes model retains the basic structure of the NB model but augments it by adding edges between the attributes to include the information of interdependence between the attributes. This process will increase the computational complexity of the system but may make classification more accurate. Each attribute can be correlated to multiple attributes so finding those interdependence and also calculating the conditional probability based on multiple attributes can be computationally intensive. There can be restriction imposed on the level of interaction between variables to reduce the computational complexity.

One such model is the Tree-Augmented Naïve Bayes Model (TAN). In the TAN model we restrict the level of interaction between the random variables to one. In this model the basic structure of NB model is retained. There are direct edges between the class node and all the attributes. Therefore, it will take into consideration all the attributes while calculating the \( P(C|A_1A_2\ldots A_N) \). In addition of that, each random variable is connected to one other random variable through a direct edge except for the specialized attribute called the root. Except for the root variable (the class), each variable in the network will have one or two parents: one is class node and the other one, when present, is another random variable. As interaction between the attributes have been limited to only one the computational complexity of this model is greatly reduced. "Thus, TAN maintains the robustness and computational complexity of the Naive Bayes model and at the same time displays better accuracy" [12].
### 6.1.1 The Tree-Augment Naïve Bayes Model

TAN imposes the restriction that the number of correlation between the attributes associated with the system is limited to one. Below figure shows an example TAN model for the car evaluation data set [17].

![Structure of a TAN model for car evaluation](image)

In this car evaluation data set we try to evaluate whether a car is acceptable given attributes related to price, technology and comfort. For this dataset if we consider the attribute like safety of the vehicle, it is closely related to the buying price of the vehicle. If the buying price of the car is high, then the car is generally safer and similarly if the buying price is low generally the safety features related to the car is lesser. In case of NB model, these would be considered as two separate independent events, so it will over penalize the class label [10]. Whereas, in case of TAN we model the important correlation between these events.

Note that in the above figure, the edges between the class labels and attributes are represented using solid lines and edges between the attributes are represented using the dotted lines. As it can be seen all the attributes except the buying cost attribute has two parents, so it is considered as the root. If we remove all the edges from the class label to the attributes a tree structure can be visualized. All the edges can be seen pointing outwards from the root. This is an example of tree structure in the TAN model.
6.1.2 Implementation:

The most important aspect in TAN model is the tree structure. To construct the tree structure for each attribute we need to find the attribute with which it is most correlated. This is how the parent of each attribute is found out. Chow and Liu [13] explain the procedure for building this tree structure as follows: "This procedure reduces the problem of constructing a maximum likelihood tree to a maximum weighted spanning tree in a graph" [12].

In order to find the most correlated attributes within this model mutual information is used [15]. The mutual information is calculated between each pair of attributes which form the weight of the edges. The edges are added between the attributes which are highly interdependent. If there are \( N \) attributes in the system, there will \( N \) nodes in the tree structure and there will be \( N-1 \) edges needed to connect all the nodes within the graph. The sum of the mutual information on the edges within the graph should form maximum weight spanning tree. Prim’s algorithm is used in this project to find the maximum weighted spanning tree [16].

Mutual information between two attributes form the weight of edges. The mutual information between two random variables \( X \) and \( Y \) is defined by the following [12]:

\[
I(X; Y) = \sum_{x,y} P(x,y) \ast \log (P(x,y)/P(x)P(y))
\]

If two variables are given the mutual information will calculate how much one variable provide information about the other. Conditional mutual information given the class label is used to construct the tree structure in TAN model. The conditional mutual information that is used to construct the tree structure is given below [12]:

\[
I(X; Y | Z) = \sum_{x,y,z} P(x,y,z) \ast \log (P(x,y | z)/P(x | z)P(y | z))
\]

The tree can be constructed for the TAN model by using the following steps:

1. Compute \( I_p(A_i, A_j|C) \) for each pair of attributes where \( i \neq j \).
2. An undirected graph is built with nodes from \( A_1, A_2, \ldots, A_N \). Assign the weight of edge between \( A_i, A_j \) using \( I_p(A_i, A_j|C) \).
3. Build the maximum weighted spanning tree.
4. The undirected graph is converted into a directed graph by choosing a random variable as root and directing all the edges outward from it.

After the construction of the tree structure the conditional probability of the root given the class is computed and stored. Also, the conditional probability of all of the attributes given the parent and the class is computed and stored. The prior probability of each class label is also computed and stored. In this project I considered the first attribute to always represent the root node. During classification the posterior probability of each of the class label $P(C|A_1,A_2,\ldots,A_N)$ is calculated. The class label with maximum posterior probability is used to classify the test record. As discussed previously, we use Laplace correction to mitigate the condition where given class and feature do not appear together in the training dataset. The formula for TAN classification is given by:

$$P(C|A_1,A_2,\ldots,A_N) = P(C) \cdot P(A_{\text{root}}|C) \prod_i P(A_i|C,A_{\text{parent}})$$

### 6.1.3 Flowchart for TAN Model Training and Classification:

To construct a TAN model, the pair wise mutual information is calculated between all nodes. The maximum weighted tree is then built using Prim’s spanning tree algorithm. Also, the prior probabilities and conditional probabilities needs to be calculated, which is very similar to NB model. Then the classification of the test records is done. Flowcharts describing the important steps of TAN model is given below. The flowcharts for the steps that are similar to Naïve Bayes are excluded here.
Flowchart for Mutual Information calculation in TAN model:

Start

C[1 … N] = List of all Class Labels.
A[1 … M] = List of all Attributes.

For i = 1; i<=len(C); i++

For j = 1; j<=len(A); j++

For k = j+1; k<=len(C); k++

Calculate the mutual information between each pair $I_P(A_j;A_k|C[i])$ and store it in mutInformation.

Return mutInformation
Flowchart for Constructing Tree in TAN model:

1. Start
2. Add $A_1$ in the VertexList. Add all attributes $[A_1 \ldots A_M]$ as nodes in the graph
3. While $\text{len(VertexList)} \neq \text{len(A)}$
   - Find the maximum weight of edge which has one vertex in VertexList and other not in VertexList.
   - Add the other vertex in VertexList. Add the edge between nodes.
4. Return graph
Flowchart for testing in TAN model:

Start

Test[1 … R] = List of all Test records.
C[1 … N] = List of all possible classes.
predictClass[1 … R] = List of predicted classes which is initialized.

For i = 1; i<=len(Test); i++

For j = 1; j<=len(C); :

Calculate the posterior probability of C[j] using formula
P(C[j]|A_1 … A_M) = prior(C[j])* conProb (root|C[j]) ∏
Prob(A_i|C[j], parent A_i) for all A in (A_1 … A_M) except root.

predictClass[i] = max(C[1 … N])

Return TestClass
6.1.4 Time Complexity of the TAN Training and Classification:

The most important steps in TAN model are mutual information calculation between each pair of attribute, prior probability calculation and conditional probability calculation conditioned on class value and the value of its parent attribute.

Let C be the total number of class labels possible for the system.
Let N be the total number of attributes associated with the naïve bayes model.
Let $S_1, S_2, S_3, \ldots, S_N$ be the attributes associated with the system.
Let $A_1, A_2, A_3, \ldots, A_N$ be the number of attribute values possible for each of the attribute $S_1, S_2, S_3, \ldots, S_N$.
Let $A_{\text{max}}$ be the maximum of $\{A_1, A_2, A_3, \ldots, A_N\}$.
Let T be the number of training instances
Let R be the number of test instances.

Let us assume that the number of class labels and the count of each attribute state corresponding conditioned on given class label can be calculated in a single scan through the training instances. The time complexity for counting is $O(T)$.

For prior probability calculation the time complexity is the same as the NB model which is $O(C)$.

For the mutual information calculation, we need to consider each pair of attributes. If there are N attributes the number of pairs possible is $^N \text{C}_2$ which is $N(N-1)/2$. For each pair of attributes, all the possible attribute states must be considered. If we take $A_{\text{max}}$ as the number of attribute state possible for each attribute, then the total number of computations can be upper bound by:

$$\text{Total computations} \leq (\frac{N(N-1)}{2}) \times C \times A_{\text{max}}^2$$

So, the total complexity is $O(N^2C \times A_{\text{max}}^2)$.

The conditional probability calculation of the TAN model is little bit different compared to the NB model. In this model we need to find conditional probability of each attribute conditioned on its parent attribute value and the class label value.

For $i = 1$ to $N$
    For $j = 1$ to $A_i$
        For $k = 1$ to $\text{len(parent}(A_i))$
For \( l = 1 \) to \( \text{len}(C) \)
Calculate conditional probability \( P(x=a|y=b,z=c) \)

Where \( a \) represents the attribute state at \( A_i \), \( b \) represents the attribute state of parent of \( A_i \) and \( z \) represents the class value at \( C[l] \).

If we take \( A_{\text{max}} \) as the total number of attribute states for each attribute the maximum number of computations can be upper bound by:

\[
\text{Total computations} = \sum_{i=1}^{N} C \times A_{\text{max}} \times A_{\text{max}} = N \times C \times A_{\text{max}}^2
\]

Hence, time complexity for conditional probability calculation is \( O(NCA_{\text{max}}^2) \).

As it can be seen the time complexity of TAN construction is polynomial whereas the time complexity to construct a BN is exponential. As we increase the level of interaction between the attributes within the system the performance of the classification increases but the time complexity of the model also increases. Similarly, if we decrease the level of interaction among the attributes the time complexity decreases but also it may affect the performance of the classification.

### 6.2 The TAN model using max abs error

TAN model is a more practical model than the NB model when the attributes are correlated. In the TAN model the level of interaction among the attributes is restricted to only one. To build the tree structure in the TAN model where most correlated attributes have a parent child relationship, a maximum weight spanning tree is built. Mutual Information between two variables is used to find correlation between those two attribute which forms the weight of the edge. Mutual information formula is given by:

\[
I(X; Y) = \sum_{x,y} P(x,y) \times \log \left( \frac{P(x,y)}{P(x)P(y)} \right)
\]

Where \( x \) and \( y \) are the random variables.

Logarithmic operations are computationally expensive. In TAN model the weight between each pair of edges is calculated. So, if there is \( N \) attributes and maximum attribute state \( A_{\text{max}} \) is assigned to number of attribute states for each attribute, then total number of computations can be upper bounded by:

\[
\text{Total computations} \leq \frac{(N^2-N)/2}{2} \times C \times A_{\text{max}}^2 \leq \frac{(N^2-N)/2}{2} \times C \times A_{\text{max}}^2
\]
So there are \(((N^2-N)/2) * C * A_{\text{max}}^2\) logarithmic operations which makes the weights of edges calculation more computationally intensive.

In this project, I attempted to make the TAN model less computationally expensive. I tested the use of the max abs error instead of the mutual information to measure the correlation between each pair of attributes, which will save more computational CPU cycles. The mutual information provides an upper bound on the max abs error.

\[
\max\{x,y\} \ |p(x)p(y) - p(xy)| \leq 0.5\sqrt{I(x,y)}
\]

I investigated empirically with TAN applied to several data sets to determine if the max abs error measure improves classification or makes it worse, compared to the mutual information measure.

If we use squared max abs error the value obtained will be related to mutual information between the variables. But as the logarithmic operations are not used, the calculation will take less computational cycles. Hence, instead of using mutual information to calculate the weight of edges between each pair of attributes, in this model we use squared max-absolute error to calculate the weight of edges. So, the learning model using this approach will be significantly faster in execution.

The max absolute error calculation should be conditioned on class:

\[
\max\{x,y|c\} \ |p(x|c)p(y|c) - p(xy|c)|^2 \leq 0.5 \sqrt{I(x,y|c)}
\]

After the above formula is used to calculate the weight of edges between each pair of attributes the maximum weight spanning tree is built using prims algorithm. The rest process remains similar to the TAN model.

If the performance of this modified version remains comparable to the classical TAN model, we would be able to achieve faster execution time using this model.

6.2.1 Implementation:

The implementation remains the same except mutual information calculation is replaced with squared max absolute error calculation.
6.2.2 Flowchart:

In this modified version of TAN model pair wise squared max absolute error needs to be calculated. The rest of the steps remains similar to TAN model. The maximum weighted tree is then built using prims algorithm. Also, the prior probabilities and conditional probabilities needs to be calculated. Then the classification of the test records is done. Below there are flowcharts describing the important step of weights of edges calculation in the modified TAN model. The rest of the flowcharts remains same as TAN model.
Flowchart for calculating weight of edges:

1. Start

2. \( C[1 \ldots N] = \text{List of all Class Labels.} \)
   \( A[1 \ldots M] = \text{List of all attributes.} \)

3. For \( i = 1; i \leq \text{len}(C); i++ \)
   - For \( j = 1; j \leq \text{len}(A); j++ \)
     - For \( k = j+1; k \leq \text{len}(C); k++ \)
       - Calculate the squared maximum absolute error between each pair \( I(A_j,A_k|C[i]) \) and store it in edges

4. Return edges
6.2.3 **Time Complexity:**
Time complexity is the same as the TAN model with the mutual information divergence.
Chapter 7

Different versions of TAN modeling for streaming data:

Nowadays there is huge emphasis of implementing machine learning algorithms to work on streaming data. There is a continuous evolution of machine learning algorithms so that they can learn in real time working on non-static data. This machine learning algorithms can train their model as continuous stream of data is fed into it.

In this project, TAN model has been implemented to handle streaming data. TAN model has been implemented in different ways so that it is able to work on streaming data. In next couple of sections, those different approaches are discussed in detail.

7.1 Batch Algorithm Approach:

7.1.1 TAN model on streaming data:

In this TAN model for streaming data we want our model to learn as streaming data arrives. We use a batch training algorithm approach where we switch between training phase to testing phase based on the streaming data.

A class label count table and pair wise count table for each attribute state of each attribute given any class label is maintained. At the beginning all the counts are initialized to 0. Now, as the streaming data arrives based on each record, the count tables are updated.

As, the testing data comes in it switches from training mode to testing mode. During the switching it first calculates the prior probabilities of each class label. It also calculates the pair wise mutual information between the attributes. Then the maximum weight spanning tree is built using the prim’s algorithm. Then the conditional probabilities of each attribute conditioned on class label and its parent are calculated and stored in dictionary.

The testing phase of this model remains similar to the TAN model on static data.
7.1.2 Flowchart:

Flowchart of the testing phase remains the same. Flowchart of the training phase and the switching phase are given below:

Flowchart of TAN model during training from streaming data:

Start

R[1 … N] = Training data

Update class count

For i = 1; i<=len(A); ... 

For j = 1; j<=len(A); ...

Update pair wise count of attribute state given a class and store it in frequency map

i

j
Flowchart of TAN model during switching from training to testing phase:
(a.)

Start

For $i = 1; i <= \text{len}(C);$ ;

$\text{prior}[C] = C[i]/T$
where $T = \text{no. of training instance}$

$i$

Return prior
(b.)

Start

C[1 … N] = List of all Class Labels.
A[1 … M] = List of all attributes.

For i = 1; i<=len(C); i++

For j = 1; j<=len(A); j++

For k = j+1; k<=len(C); k++

Calculate the mutual information between each pair I(A_j; A_k|C[i]) using count from frequency map and store it in mutInformation.

Return mutInformation
(c.) Building of tree using Prim’s algorithm remains same as TAN model.

(d.) Calculation of conditional probabilities also remains same as TAN model.

7.1.3 **Time Complexity:**

Time complexity in this model remains more or less the same as the TAN model. During training the class count needs to be updated and also pair wise attribute states needs to be updated. The time complexity for updating the class count table is constant time and updating the pair wise attribute count is \( O(N^2) \) if there is \( N \) attributes for the dataset.

Now, while switching from training phase to testing phase. We need to calculate prior probabilities which is \( O(C) \) if there is \( C \) class labels. The pair wise mutual information needs to be calculated which is \( O(N^2*C*A_{\text{max}}^2) \).

7.2 **Incremental TAN Learning**

When the TAN model is being implemented on streaming data using incremental approach we need to make sure to recalculate the model whenever new training instance can affect the model considerably and not recalculate for each new training instance. One of the most important parts of TAN model is its tree structure. Based on the tree structure the parent of each attribute is decided which is then used for conditional probability calculation. So, we need to recalculate the model whenever new training instance has the potential to affect the tree structure.

7.2.1 **TAN model on streaming data using sum of weights of edges:**

During the learning phase in TAN model, we calculate sum of edges between each pair of attributes which is then used to build the tree structure of the model using prims algorithm. Given a new training instance we need to recalculate the model whenever it has the potential to affect the tree structure. If the tree is updated for each training instance, then it will increase the computational cycle usage considerably. In fact, then it will be like using batch algorithm approach for each new training instance.

For each new training instance, we need to assess whether it has potential to change the tree structure, if so the tree structure should be reconstructed otherwise it should be left unchanged. In this modified version of TAN model, I have tried to assess whether the new training instance can update the tree structure using the
sum of weights of edges as the deciding factor. The edges between each pair of attributes is calculated for each training instance. The sum of weights of these edges are also calculated. Now, for a new training instance if the sum of weights of edges varies considerably compared to previous sum of weights of edges then that training instance has the potential to update the tree structure. So, the model is then recalculated.

In this model each time a new training instance arrives the class count table and the count table for pairwise count of each attribute state is updated. Then the weights of edges between each pair of attributes are calculated. The summation of weights of edges between each pair of attributes is performed to arrive at the grand sum. If the grand sum of edges differs from the previous grand sum of edges by more than a threshold value, a new tree structure is constructed using the Prim’s algorithm.

The calculation of prior probabilities is performed based on the class count table. The conditional probabilities are calculated based on the current tree structure of the model.

The testing phase remains similar to the TAN model where posterior probabilities are calculated for each class label. The class label with maximum posterior probability is assigned to the training instance.
7.2.1.1 **Flowchart:**
Most of the steps remain the same as the TAN model. The flowchart for the training phase which is different is given below:

Start

R[1 … N] = Training data instance.

Update Class count.

Update pairwise count of attribute state and store it in frequency map.

Calculate weight of edges between each pair of attributes and calculate the grand total of sum of all the edges.

\[
\left| \text{sum of edges}\right|_{\text{current}} - \left| \text{sum of edges}\right|_{\text{previous}} \geq \text{THRESHOLD}
\]

YES

Recalculate the tree structure.

NO

Stop
7.2.1.2 **Time Complexity:**

Time complexity remains the same as TAN model when trees are being updated. Time complexity when trees are not being updated during learning is equal to time required for calculating the weights of edges between each pair of attributes which as we know is equal to $O(N^2*C*A_{\text{max}}^2)$.

7.2.2 **TAN model for streaming data based on average of sum of weights:**

In this model, we are more concerned with how the average weight of sum of edges being affected for each training instance. During training we are calculating the sum of edges between each pair of attributes along with that we keep a track of cumulative moving average of the sum of weights.

As the training instance arrive, if the sum of weights varies considerably from the cumulative moving average it can be an indicator that the model needs to be recalculated. Generally, as the number of training instances increase the sum of weights of edges and the cumulative average of sum of weights should move towards each other. If for particular instance, the difference between sum of weights of edges and cumulative moving average is considerable then it is a good indicator that the model needs to be retrained as the tree structure may change because of such variance.

In this model each time a new training instance arrives the class count table and the count table for pairwise count of each attribute state is updated. Then weights of edges between each pair of attributes is calculated. A summation is performed on weights of all the edges to arrive at the grand total. Along with that cumulative moving average is maintained for the summation of weights of edges. If the difference between the grand total and moving average is beyond a threshold value, then the tree structure is reconstructed using the prim’s algorithm. The calculation of prior probabilities is performed based on the class count table. The conditional probabilities are calculated based on the current tree structure of the model.

The testing phase remains similar to the TAN model where posterior probabilities are calculated for each class label. The class label with maximum posterior probability is assigned to the training instance.

7.2.2.1 **Flowchart:**
Most of the steps remains same as the TAN model. The flowchart for the training phase which is different is given below:

Start

R[1 … N] = Training data instance

Update Class count.

Update pairwise count of attribute state and store it in frequency map.

Calculate weight of edges between each pair of attributes and calculate the grand total of sum of all the edges.

\[|\text{sum of edges}_{\text{current}} - \text{sum of edges}_{\text{previous}}| \geq \text{THRESHOLD}\]

NO

Stop

YES

Recalculate the tree structure.
7.2.2.2 **Time Complexity:**

Time complexity remains the same as TAN model when trees are being updated. Time complexity when trees are not being updated during learning is equal to time required for calculating the weights of edges between each pair of attributes which as we know is equal to \(O(N^2C*A_{\text{max}}^2)\).

7.2.3 **TAN model on streaming data based on weights of edges of non-root attributes:**

In the previous streaming models that has been discussed, the weights of edges between each pair of attributes are calculated whenever new training instance is fed into the model. The complexity of calculating the weights of edges between each pair of attributes is \(O(N^2C*A_{\text{max}}^2)\). This step is considerably computationally intensive.

Instead of observing weights of edges between each pair of attributes, we try to concentrate on the weights of edges for only non-root tree nodes and how it varies. Once a tree structure is calculated, the weights of edges of maximum weight spanning tree for non-root nodes are observed. Whenever a new training instance is fed into the model, it calculates the weights of edges for only non-root nodes. If the calculated weights differ from the previously calculated weights by a considerable amount, then there is big chance that the tree needs to be restructured.

So, in such cases the tree structure is again recalculated. In the following paragraph the important steps of this model are explained.

Whenever there is new training instance, the TAN model updates the class count table and the pair wise attribute state. If there is no tree structure present, it calculates the maximum weight spanning tree using the weights of edges between each pair of attributes. If there is already a tree structure present, then it calculates the weights of edges only for the pair of attributes between which the edges are already present. If the newly calculated weights differ from previously calculated weight by more than a threshold value, then the tree structure is again rebuilt after calculating weights of edges between each pair of attributes. The calculation of prior probabilities is performed based on the class count table. The conditional probabilities are calculated based on the current tree structure of the model.
The testing phase remains similar to the TAN model where posterior probabilities are calculated for each class label. The class label with maximum posterior probability is assigned to the training instance.

7.2.3.1 **Flowchart:**

Most of the steps remain same as the TAN model. The flowchart for the training phase which is different is given below:
Start

R[1 \ldots N] = \text{Training data instance.}

Update Class count.

Update pairwise count of attribute state and store it in frequency map.

Is tree structure present?

\begin{array}{l}
\text{Calculate weight of edges between attributes for which edges are already present.} \\
\text{Calculate weight of edges between each pair of attributes.}
\end{array}

\text{Calculate tree and store the weights of edge for each node in the tree.}

\begin{array}{l}
\text{Calculate weight of edges for which } |(\text{edges})_{\text{current}} - (\text{of edges})_{\text{previous}}| \geq \text{THRESHOLD for} \\
\text{Calculate weight of edges between each pair of attributes.}
\end{array}

\text{Calculate tree and store the weights of edge for each node in the tree.}

\text{Stop}

\text{Stop}
7.2.3.2 **Time Complexity:**

The time complexity remains the same as TAN model when trees are being reconstructed. Time complexity when trees are not being rebuilt during learning is equal to time required for calculating the weights of edges for N attributes which is $O(N)$. 
Chapter 8

8.1 Experimental Results

The datasets for this project were obtained from the UCI machine learning repository [14]. It comprises only discrete valued datasets. All data need to undergo some preprocessing steps, to make them suitable for using as input to the program. From the data set two third of data is selected in random for training and one third of it is used for testing.

8.2 Data Preparation

For each dataset there is two different files. One is attribute file which contains all the information related to classes and attributes in the following format:

- Class values: Following the text there is comma separated list of class values possible.
- Attributes: Following this text in each line there is a name of the attribute and comma separated list of values possible for that attribute.

The dataset is contained in a separate file where each line constitutes for a record instance which contains comma separated values.

8.3 Brief Descriptions of Datasets used for testing

Lymphography: Each class refers to the current state of lymphoma. The attribute values refer to different features related to lymphoma. The classification problem is to identify the current state of lymphoma based on the attribute values [20].

Car: Each class refers to different values of car acceptability. Attributes are the different features related to price and technology related to the car. The classification problem is to identify car acceptability based on the attribute values[17].

Votes: Each class refers to different political party. The different attributes refer to voting pattern to different congressional voting agendas. The classification problem is to identify the political party based on the attribute values [19].

Nursery: Each class value refers to rank of application to nursery school. The different attribute values refer to different features related to occupation of the
parents, financial, social and health standings. The classification problem is to identify the rank of the application based on the attribute values [18].

<table>
<thead>
<tr>
<th>Name of Dataset</th>
<th>Number of Classes</th>
<th>Number of Attributes</th>
<th>Number of Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>2</td>
<td>18</td>
<td>142</td>
</tr>
<tr>
<td>Car</td>
<td>4</td>
<td>6</td>
<td>1728</td>
</tr>
<tr>
<td>Votes</td>
<td>2</td>
<td>16</td>
<td>435</td>
</tr>
<tr>
<td>Nursery</td>
<td>5</td>
<td>8</td>
<td>12960</td>
</tr>
</tbody>
</table>

8.4 **Performance evaluation of NB and TAN Classification**

We consider a classification problem which consist of two class values. In case of such classification problem, if we randomly classify a test instance as one of the two class we might end up with fifty percent accuracy [10]. So, any sophisticated classification model needs to perform better than the randomized model. Classifying instances based on prior probability of classes may perform better than random classification. In this method the prior probability of each class is calculated from the training dataset. Then all the test instances are classified to the label which has the highest prior probability. In case of two class problems, it can give accuracy of at least fifty percent. So, the prior probability performs better than the random classification.

These two methods act as the baseline for any classifier model. Hence, any sophisticated method should perform significantly better than these methods. The performance increase should be worthy of the increase of complexity in the model [10].

8.5 **Performance result of different models**

**Naïve Bayes:** The classification performance achieved using Naïve Bayes is given below:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>80.85</td>
</tr>
<tr>
<td>Car</td>
<td>85.07</td>
</tr>
<tr>
<td>Votes</td>
<td>89.66</td>
</tr>
<tr>
<td>Nursery</td>
<td>89.91</td>
</tr>
</tbody>
</table>
**Naïve Bayes on streaming data:**
The classification performance of Naïve Bayes on streaming data model is given below, which is the same:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>80.85</td>
</tr>
<tr>
<td>Car</td>
<td>85.07</td>
</tr>
<tr>
<td>Votes</td>
<td>89.66</td>
</tr>
<tr>
<td>Nursery</td>
<td>89.91</td>
</tr>
</tbody>
</table>

A, we are using batch algorithm approach for streaming data the performance of this model remains the same as that of Naïve Bayes model.

**Tree Augmented Naïve Bayes:** The classification performance of TAN and its comparison to NB is given below:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TAN</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>85.11</td>
<td>80.85</td>
</tr>
<tr>
<td>Car</td>
<td>88.02</td>
<td>85.07</td>
</tr>
<tr>
<td>Votes</td>
<td>91.72</td>
<td>89.66</td>
</tr>
<tr>
<td>Nursery</td>
<td>91.97</td>
<td>89.91</td>
</tr>
</tbody>
</table>

**TAN using max absolute error:** The classification performance of TAN using max absolute error and its comparison to TAN and NB is given below:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TAN using max absolute error</th>
<th>TAN</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>85.11</td>
<td>85.11</td>
<td>80.85</td>
</tr>
<tr>
<td>Car</td>
<td>89.06</td>
<td>88.02</td>
<td>85.07</td>
</tr>
<tr>
<td>Votes</td>
<td>95.86</td>
<td>91.72</td>
<td>89.66</td>
</tr>
<tr>
<td>Nursery</td>
<td>92.38</td>
<td>91.97</td>
<td>89.91</td>
</tr>
</tbody>
</table>

The timing comparison of this model with TAN model is given below:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TAN using max absolute error(seconds)</th>
<th>TAN(seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>2.87</td>
<td>3.06</td>
</tr>
<tr>
<td>Car</td>
<td>4.81</td>
<td>5.43</td>
</tr>
<tr>
<td>Votes</td>
<td>2.36</td>
<td>2.43</td>
</tr>
<tr>
<td>Nursery</td>
<td>92.15</td>
<td>94.72</td>
</tr>
</tbody>
</table>

As we can see the timings of TAN using max absolute error are better when compared to timings of TAN. The time complexity for calculation weight of edges between each pair of attributes is $O(N^2C_{\text{max}}^2)$. As, the number of attributes in
these datasets and the number of possible attribute states are not that large so the time difference is not so huge.

**TAN on streaming data using batch algorithm approach:**
The classification performance of TAN model on streaming data model is given below:

<table>
<thead>
<tr>
<th>Datasets</th>
<th>TAN on streaming data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>85.11</td>
</tr>
<tr>
<td>Car</td>
<td>88.02</td>
</tr>
<tr>
<td>Votes</td>
<td>91.72</td>
</tr>
<tr>
<td>Nursery</td>
<td>91.97</td>
</tr>
</tbody>
</table>

As, we are using batch algorithm approach for streaming data the performance of this model remains the same as of TAN model.

**TAN model on streaming data using sum of weights of edges (TAN 1):**
The classification performance of TAN model on streaming data using sum of weights of edges and its comparison with TAN model and NB model is given below:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TAN 1</th>
<th>TAN</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>85.11</td>
<td>85.11</td>
<td>80.85</td>
</tr>
<tr>
<td>Car</td>
<td>90.10</td>
<td>88.02</td>
<td>85.07</td>
</tr>
<tr>
<td>Votes</td>
<td>91.72</td>
<td>91.72</td>
<td>89.66</td>
</tr>
<tr>
<td>Nursery</td>
<td>91.97</td>
<td>91.97</td>
<td>89.91</td>
</tr>
</tbody>
</table>

This current model is referred as TAN 1.
The number of times tree is reconstructed using this model is given below:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#reconstruction of tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>94</td>
</tr>
<tr>
<td>Car</td>
<td>427</td>
</tr>
<tr>
<td>Votes</td>
<td>289</td>
</tr>
<tr>
<td>Nursery</td>
<td>8550</td>
</tr>
</tbody>
</table>

**TAN model for streaming data based on average of sum of weights (TAN 2):**
The classification performance of this model and its comparison to TAN 1(discussed above), TAN model and NB is given below:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TAN 2</th>
<th>TAN 1</th>
<th>TAN</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>82.98</td>
<td>85.11</td>
<td>85.11</td>
<td>80.85</td>
</tr>
<tr>
<td>Car</td>
<td>89.58</td>
<td>90.10</td>
<td>88.02</td>
<td>85.07</td>
</tr>
</tbody>
</table>
The number of times tree is reconstructed in this model and its comparison with TAN 1 is given below:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TAN 3</th>
<th>TAN 2</th>
<th>TAN 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>3</td>
<td>11</td>
<td>94</td>
</tr>
<tr>
<td>Car</td>
<td>8</td>
<td>2</td>
<td>427</td>
</tr>
<tr>
<td>Votes</td>
<td>3</td>
<td>14</td>
<td>289</td>
</tr>
<tr>
<td>Nursery</td>
<td>14</td>
<td>17</td>
<td>8550</td>
</tr>
</tbody>
</table>

**TAN model on streaming data based on weights of edges of non-root attributes (TAN 3):**

The classification performance of this model and its comparison to TAN 2, TAN 1, TAN model and NB model is given below:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TAN 3</th>
<th>TAN 2</th>
<th>TAN 1</th>
<th>TAN</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>85.11</td>
<td>82.98</td>
<td>85.11</td>
<td>85.11</td>
<td>80.85</td>
</tr>
<tr>
<td>Car</td>
<td>90.28</td>
<td>89.58</td>
<td>90.10</td>
<td>88.02</td>
<td>85.07</td>
</tr>
<tr>
<td>Votes</td>
<td>94.48</td>
<td>96.55</td>
<td>91.72</td>
<td>91.72</td>
<td>89.66</td>
</tr>
<tr>
<td>Nursery</td>
<td>91.97</td>
<td>80.09</td>
<td>91.97</td>
<td>91.97</td>
<td>89.91</td>
</tr>
</tbody>
</table>

The number of times tree is reconstructed in this model and its comparison with TAN 1 and TAN 2 is given below:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TAN 3</th>
<th>TAN 2</th>
<th>TAN 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lymphography</td>
<td>3</td>
<td>11</td>
<td>94</td>
</tr>
<tr>
<td>Car</td>
<td>8</td>
<td>2</td>
<td>427</td>
</tr>
<tr>
<td>Votes</td>
<td>3</td>
<td>14</td>
<td>289</td>
</tr>
<tr>
<td>Nursery</td>
<td>14</td>
<td>17</td>
<td>8550</td>
</tr>
</tbody>
</table>
Chapter 9

Conclusion and Future work:

I found that the classification performance of the Tree-Augmented Naïve Bayes (TAN) model is better than the Naïve Bayes (NB) model if the attributes associated with the model are correlated. Furthermore, with the advent of internet, the need for machine learning algorithms on streaming data has increased. When applying the TAN model to streaming data, the batch algorithm approach is inefficient as it takes considerable amount of time to switch from training phase to testing phase. For real time applications the incremental approach is more valuable. As soon as new data training arrives, it can be fed into the model and the model will adjust according to the training data.

Reconstruction of the TAN spanning tree is computationally expensive. In the incremental models for training there is a need to minimize the number of times tree is reconstructed. It should be reconstructed only when it is required. For my first TAN algorithm (TAN1) the TAN tree is updated very frequently, which would make the system not very practical in the real time environment. The second and third algorithms (TAN2 and TAN3), the tree is updated only if required. This produces comparable or better performance in classification in certain cases. But TAN2 is computationally more expensive compared to TAN3 as it tries to calculate edges between each pair of attributes. TAN3 is preferable in real time environment.

In case of TAN model and all its modified version the first attribute is being selected as root for the tree. It may be beneficial to explore if there is potential for performance gain if other attributes are selected as root.

In case of TAN model on streaming data using incremental algorithms, threshold value plays a very important role in deciding whether the tree needs to be reconstructed when training data is fed into the model. In these implementations we are using a constant value as a threshold value. But, the threshold value should be dependent on the dataset. There is a need of further exploration how to dynamically adjust the threshold value based on the dataset.

The number of attributes for some datasets may be huge which increases the time complexity of the model. In those cases, there can be further exploration made how
does reduction of strongly related attributes with less number of attributes will affect the outcome.

In the TAN model, we can also keep different tree structures for each class label. So, when a new training instance is fed into the model according to the class label we will only adjust that tree structure. During testing, posterior probabilities for each class label can be calculated based on each of these tree structures. There is a need for further exploration how that will affect the performance of the classification.
References:

[1]: https://en.wikipedia.org/wiki/Bayes%27_theorem
[4]: https://en.wikipedia.org/wiki/Marginal_distribution
[8]: https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/
[9]: https://en.wikipedia.org/wiki/Naive_Bayes_classifier
[15]: https://en.wikipedia.org/wiki/Mutual_information
[16]: https://en.wikipedia.org/wiki/Prim%27s_algorithm
[18]: https://archive.ics.uci.edu/ml/datasets/Nursery
[20]: https://archive.ics.uci.edu/ml/datasets/Lymphography