Mathematical Analysis and Comparison of Machine Learning Algorithms

Project report to be submitted in partial fulfilment of requirements of degree of
BACHELOR OF TECHNOLOGY IN
ELECTRONICS AND INSTRUMENTATION ENGINEERING

Supervision by
Dr. Debjani Mitra
Professor
(Department of Electronics Engineering)

Submitted by
Mrinalini Singh (16JE001958)
Apoorva S. Chouhan (16JE002048)

Department of Electronics Engineering,
INDIAN INSTITUTE OF TECHNOLOGY
(INDIAN SCHOOL OF MINES), DHANBAD

MAY, 2020

CERTIFICATE

This is to certify that the report entitled, "***Mathematical Analysis and Comparison of Machine Learning Algorithms***" submitted by **Mrinalini Singh (16JE001958)** and **Apoorva Singh Chouhan (16JE002048)** in partial fulfilment of the requirement for the award of Bachelor of Technology degree in Electronics and Instrumentation Engineering at Indian Institute of Technology (ISM), Dhanbad is an authentic work carried out by them under my supervision. To the best of my knowledge, the matter embodied in the report has not been submitted to any other University/Institute for the award of any Degree or Diploma.

_____

(Signature of supervisor)

Dr. Debjani Mitra
Department of Electronics Engineering,
Indian Institute of Technology,
Indian School of Mines, Dhanbad

Date:

Acknowledgement

We, the students of B.tech Electronics and Instrumentation Engineering, Indian Institute of Technology (Indian School of Mines) Dhanbad, are glad to present a project report for the same entitled "***Mathematical Analysis and Comparison of Machine Learning Algorithms***".

First and Foremost, we would like to express our deepest appreciation to our supervisor and professor for the course of Machine Learning, Dr. Debjani Mitra, for her invaluable encouragement, support and guidance throughout the project, as well as for providing us with the knowledge of the various algorithms and the involved mathematics, working through the very basics to the current scenario, both during the course study as well as during the project duration.

We would also like to express our gratitude towards the Department of Electronics Engineering, Indian Institute of Technology (Indian School of Mines) Dhanbad, for providing us with the basic requirements and facilitate our working on this project.

At the end, we'd like to thank almighty God for his wisdom and guidance who made us and everyone who they are today.

Mrinalini Singh (16JE001958)

Apoorva Singh Chouhan (16JE002048)

B.Tech 2020 (Electronics and Instrumentation Engineering)

Abstract

Machine learning is a popular topic of research and application at this time. It focuses on developing algorithms to “teach” a computer how to learn patterns and correlations in data. This field aims to make problem tasks such as classification, clustering, and other completely automated. More advanced algorithms can be used for facial recognition, sentiment analysis, and reinforcement learning and so on. It all started with wanting to teach a computer to win games of checkers, and here we are. Often, people working with these algorithms treat them as black-boxes, but in order to completely understand an algorithm, it is important to have a mathematical understanding of them, it is important to understand the nuances of various algorithms having similar principles.

The aim of this project is to explain the basic mathematical principles of two of the most used algorithms – decision trees and naïve bayes classifiers. An attempt has been made to explain the working using simple examples, and to understand their advantages and disadvantages, to compare them, and suggest the best field of application for each.

Table of Contents

CHAPTER 1

1. Introduction

Computers as we know today, when they were first conceived, did not serve the same purpose as they do now. They started as simple machines to solve simple equations, then used for quicker calculations using simple algorithms, advancing to complex code-breaking machines and today, they are now used to automate simple and even complex tasks which were earlier performed by humans. Because of their ability to ease the lives of humans, they were often thought of as machines that could accomplish all the tasks, hence eventually rendering humans useless. Several works of fiction have been written where computers have advanced so much that they have taken over our planet and completely destroyed the humans. The truth is, even if such a situation arises, it is in the very far future.

Ever since their invention, people have wondered if computers could be taught to learn on their own. But, they are essentially programmed by humans, so they can only perform the tasks for which clear algorithms are written by humans. A successful understanding of how to make computers learn would open up many new uses of computers and new levels of competence and customization. And a detailed understanding of information and processing algorithms for machine learning will lead to a better understanding of human learning abilities (and disabilities) as well. As of now, even after major technological developments, computers are far from learning as well as humans.

The special field of work which consists of study and development of algorithms that enable computers to learn certain behaviours is called machine learning. Several formal definitions of machine learning algorithms have been given by academicians and industry experts. The definition in its simplest and most basic from is:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. Such a computer program forms a machine learning algorithm for the particular task T.*

Machine learning has been an interesting field of study for several decades, but could not flourish well and have major advancements earlier due to lack of data and expensive computational needs. Today, several gigabytes of data is produced and systematically stored every day. With less expensive and more powerful computers, there is a boom in this field of study.

Now, due to its popularity and development in the open source community, anyone can download tools for application of several algorithms in their field of interest. Such accessibility is a boon, but it has led to the treatment of algorithms as "magic black-boxes".

As engineers, it is important for us to understand the basics and working of anything before its applications. The aim of this project is to understand the basic mathematical principles of various machine learning algorithms, to explain those using simple examples, to understand their advantages and disadvantages, to compare them and suggest the best field of application for each. The algorithms discussed are **Decision Trees** and **Naïve Bayes**.

CHAPTER 2

2. Datasets

For the purpose of this project, data from two tables will be used.

2.1. Play Tennis Dataset

First, “Play Tennis” dataset is used to explain the mathematical approach of algorithms Decision Tree and Naive Bayes Classifier. This is a popular dataset used to explain the working of machine learning algorithms. It is as in Table 2.1.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Rainy	Hot	High	Weak	No
2	Rainy	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Sunny	Mild	High	Weak	Yes
5	Sunny	Cool	Normal	Weak	Yes
6	Sunny	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Rainy	Mild	High	Weak	No
9	Rainy	Cool	Normal	Weak	Yes
10	Sunny	Mild	Normal	Weak	Yes
11	Rainy	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Sunny	Mild	Hot	Strong	No

Table 2.1: Play Tennis dataset

2.2. Iris Dataset

The Iris dataset is one of the most well-known databases found in pattern recognition and machine learning literature. It consists of 3 classes with 50 instances each. The classes refer to a type of iris plant- iris Setosa, iris versicolour, and iris Virginica.

This dataset has four attributes - sepal width, sepal length, petal width and petal length. All of the lengths are given in centimetres. Description of the dataset is available in Table 2.2.

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Table 2.2: Description of Iris dataset.

The correlation matrix of different attributes is in Figure 2.1. From this matrix, it is inferred that the attributes sepal width and sepal length are least correlated (r = -0.109).

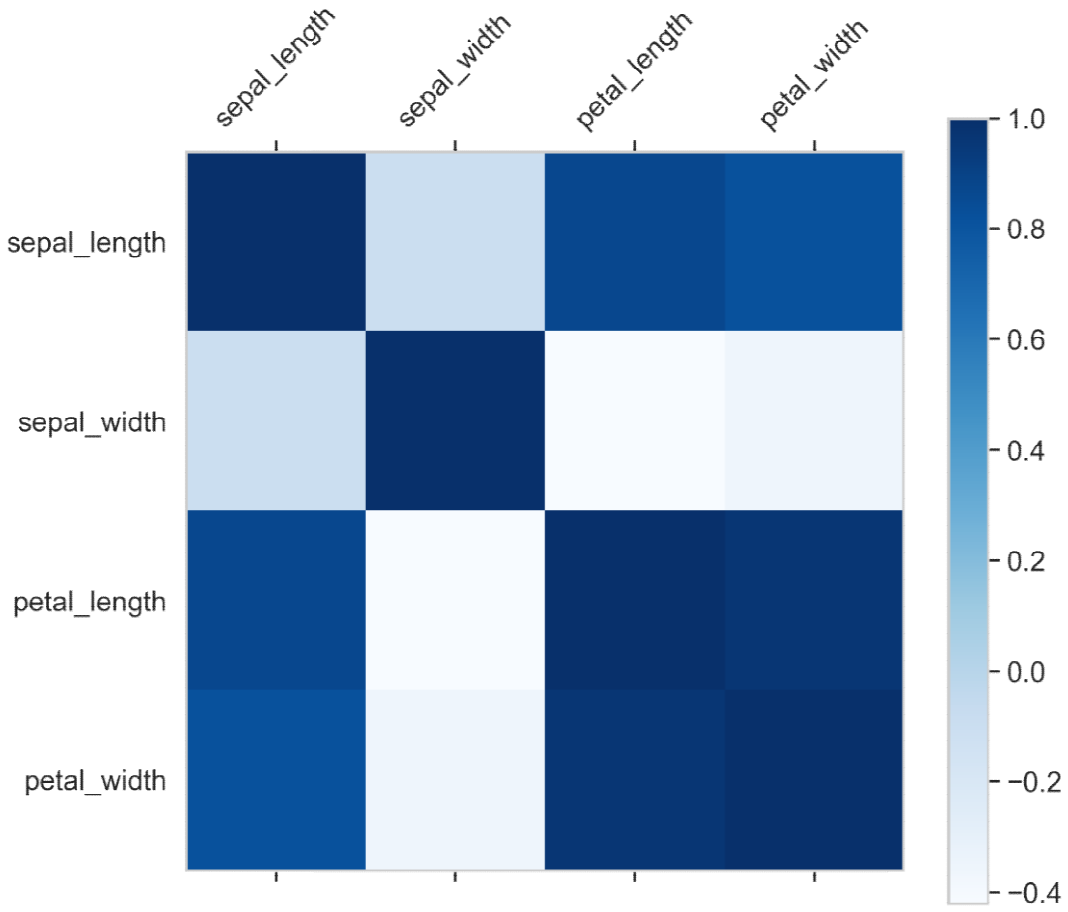


Figure 2.1: Correlation matrix of attributes for iris dataset.

In order to check the separability of the classes, the dataset is visualised using pair-plots. These plots are available in Figure 2.2. It is clear that iris Setosa is linearly separable from the other two classes, while the latter ones are not linearly separable.

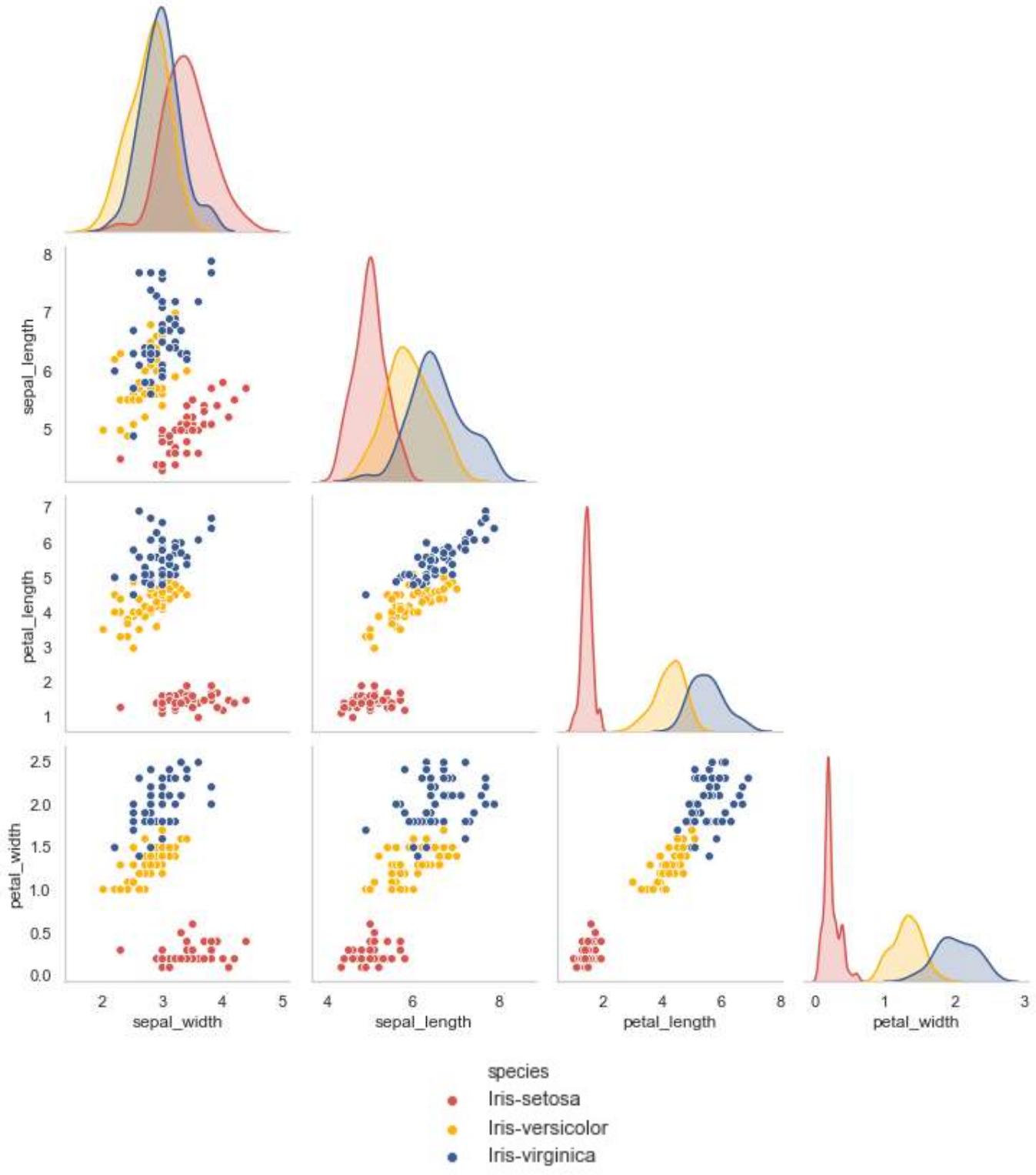


Figure 2.2: Pair-plots of attributes for iris dataset.

For the purpose of this project, the models are trained on the entire dataset and then the models are judged on how well they predict on already seen data.

2.3. Bank Note Dataset

The bank note dataset is another well-known database found in pattern recognition and machine learning literature. Data was extracted from images of genuine and forged banknote specimens. Industrial camera was used for print inspection, and wavelet transformation was performed on the images.

The dataset consists of four attributes – variance (var), skewness (skew) and curtosis of wavelet transformed images, and entropy of the images. It consists of a total of 1372 instances, with 762 belonging to class 'fake' and the rest 610 belonging to class 'authentic'. The dataset has continuous values. Description of the dataset is available in Table 2.3.

	var	skew	curtosis	entropy
count	1372.000000	1372.000000	1372.000000	1372.000000
mean	0.433735	1.922353	1.397627	-1.191657
std	2.842763	5.869047	4.310030	2.101013
min	-7.042100	-13.773100	-5.286100	-8.548200
25%	-1.773000	-1.708200	-1.574975	-2.413450
50%	0.496180	2.319650	0.616630	-0.586650
75%	2.821475	6.814625	3.179250	0.394810
max	6.824800	12.951600	17.927400	2.449500

Table 2.3: Description of bank note dataset.

The correlation matrix of different attributes is in Figure 2.3. From this matrix, it is inferred that the attributes var and skew are least correlated
(r = 0.264).

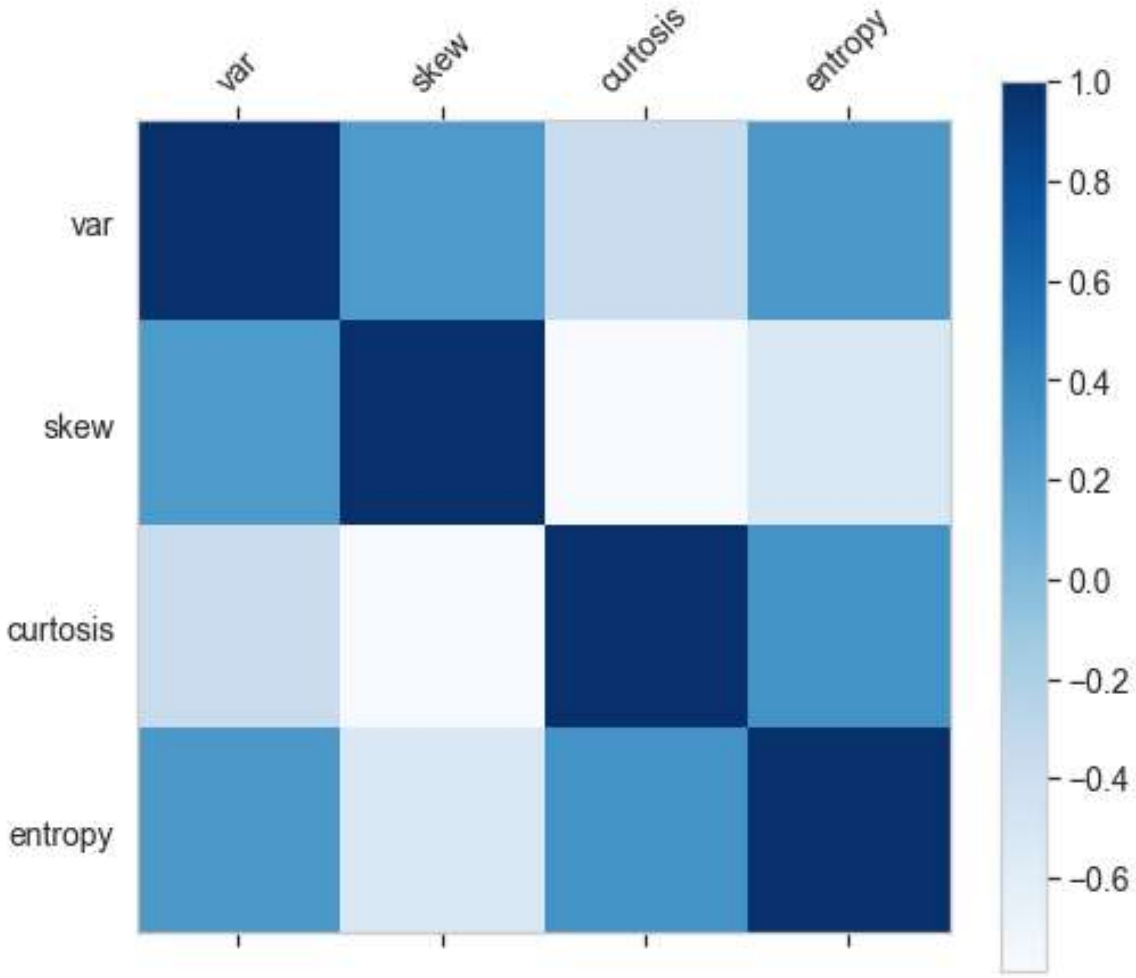


Figure 2.3: Correlation matrix of attributes for bank note dataset.

In order to check the separability of the classes, the dataset is visualised using pair-plots. These plots are available in Figure 2.4. It is clear that the data-points are not linearly separable.

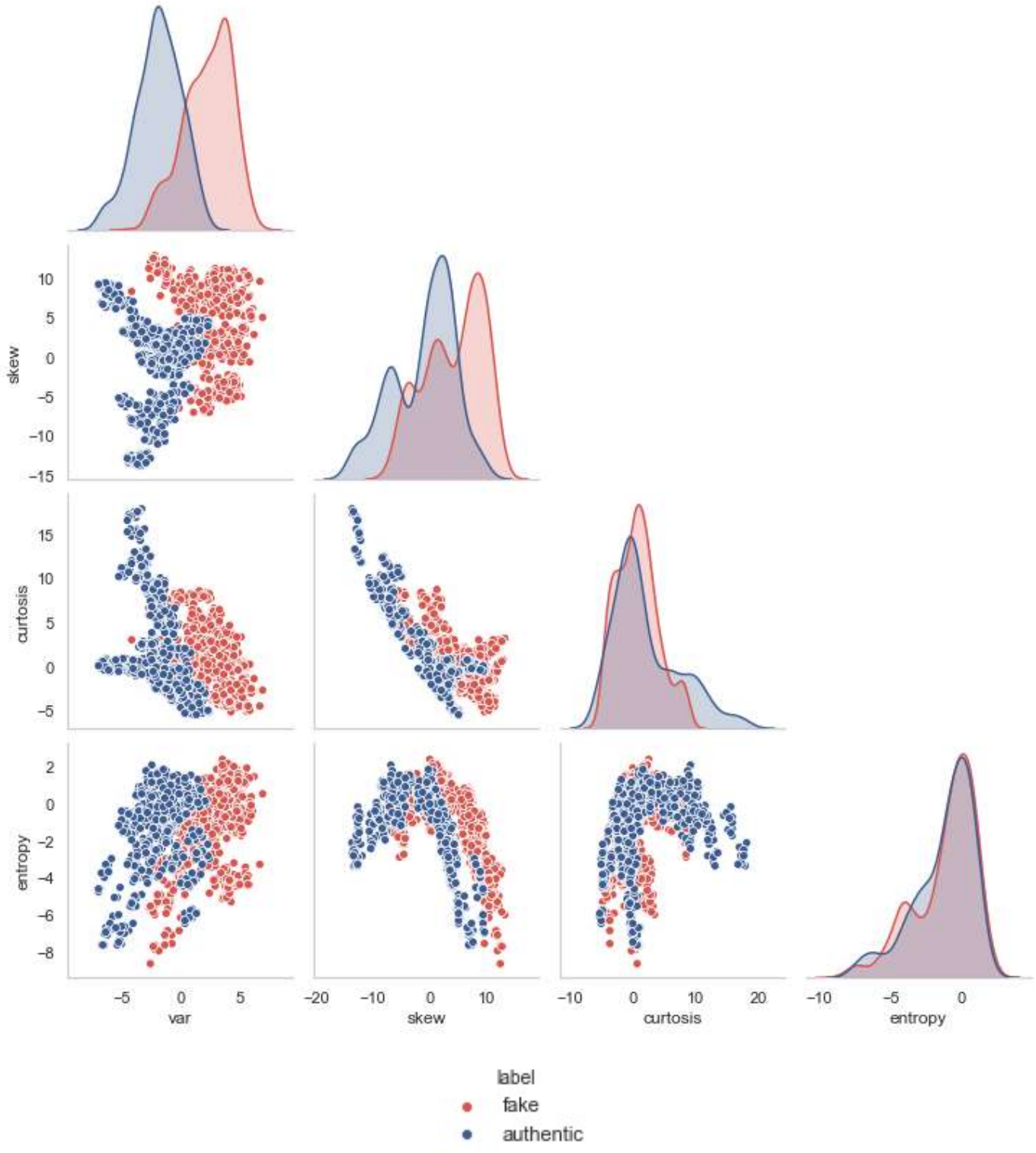


Figure 2.4: Pair-plots of attributes for bank note dataset.

For the purpose of this project, the models are trained on the entire dataset and then the models are judged on how well they predict on already seen data.

CHAPTER 3

3. Algorithms

3.1. Decision Tree

3.1.1. Introduction

Decision tree learning is one of the most common and powerful algorithms for classification. It is easy to understand the intuition behind the algorithm even without good knowledge of mathematics. A method for approximating discrete-valued functions, it is robust to noisy data and capable of learning disjunctive expressions. It falls under the category of supervised learning algorithms. With advancements in the first algorithm, decision trees can also work with continuous data, and also be used for regression. For the scope of this project, we have focused specifically on CART (Classification & Regression Trees) type decision trees.

A tree can be *"learned"* by splitting the data set into subsets based on attribute values. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. The recursion is terminated when all the subsets at a node have the same value as the target variable, or when splitting no longer improves the quality of predictions.

The final hypothesis can simply be understood as a set of if-then-else rules. This can simply be represented in the form of a flowchart.

3.1.2. Mathematics and metrics involved

Decision trees are constructed using a top-down approach, by choosing a variable at each step that best splits the set of instances. Different algorithms use different metrics for measuring *"best split"*. These generally measure the homogeneity of the target variable within the subsets (also called nodes). These metrics are applied to each candidate subset, and the resulting values are combined to provide a measure of the quality of the split.

The most commonly used metrics are:

a. *Information Gain*

Information gain is the expected reduction in entropy caused by partitioning the examples according to a particular attribute. In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called entropy that characterizes the (im)purity of an arbitrary collection of examples. Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this Boolean classification is

$$Entropy(S) = -\sum_{i=1}^{c} p_i log_2(p_i)$$

Equation 3.1

where p_i, is the proportion of instances in S belonging to class *i*, and *c* is the total number of classes. In all calculations involving entropy, $0log_20$ is defined to be 0.

Next, we define *information gain* or simply called *gain*, as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Equation 3.2

where, *Values(A)* is the set of all possible values for attribute *A*, and S_v, is the subset of *S* for which attribute *A* has value *v* (that is,
$S_v = \{s \epsilon S | A(s) = v\}$). The first term in Equation (3.2) is simply the entropy of the original collection *S*, and the second term is the expected value of the entropy after *S* is partitioned using attribute *A*. The expected entropy described by this second term is the sum of the entropies of each subset S_v, weighted by

the fraction of examples $\frac{|S_v|}{|S|}$ that belong to S_v. *Gain(S, A)* is therefore the expected reduction in entropy caused by knowing the value of attribute *A*.

Information gain is the measure used by the algorithm ID3 to select the best attribute at each step in growing the tree. The higher the value of gain for a particular attribute, the better it is for splitting at that step.

b. Gini Impurity

Gini impurity or *gini impurity* is a measure of inequality in instances. Having values between 0 and 1, it measures how often a randomly chosen instance from the set would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the subset. Gini index of value 0 means instances are perfectly homogeneous whereas, Gini index of value 1 means maximal inequality among elements. If a subset is homogeneous this means that all the instances are from the same class. Gini index is mathematically defined as the sum of the square of the probabilities of each class,

$$Gini(A = v) = 1 - \sum_{i=1}^{c} p_{i,v}^2$$

Equation 3.3

$$Gini\ Index(A) = \sum_{j=1}^{a} p_j \cdot Gini(A = v_j)$$

Equation 3.4

where *c* represents different classes, *a* represents number possible values of *v*, *v* are the different possible values of *A*, $p_{i,v}$is the proportion of instances belonging to each class for the same value *v* of attribute *A*, andp_j is the proportion of instances of subsets having the same *v*. This metric is used in CART, to evaluate the split of subset or node. The attribute with the lowest

value of gini index is taken as the root node for that particular tree, or if one split has already taken place, it will be the root node of the subsequent sub-tree. This will be clearer in the next section, as the evaluation is explained with an example.

3.1.3. Illustration

The algorithm begins by calculating the values of gini index of various attributes of the "Play Tennis" dataset from Table 2.1. Outlook, temperature, humidity and wind are the various attributes or features of the data.

a. Outlook

It can take three values - sunny, overcast and rainy. Summarising the decision for outlook features,

Outlook\Class	Yes	No	Total
Overcast	4	0	4
Sunny	2	3	5
Rainy	3	2	5

Table 3.1: Outlook summary of Play Tennis dataset

$$Gini\ (Outlook = Overcast) = 1 - (4/4)^2 - (0/4)^2$$
$$= 1 - 1 - 0$$
$$= 0$$

$$Gini\ (Outlook = Sunny) = 1 - (2/5)^2 - (3/5)^2$$
$$= 1 - 0.16 - 0.36$$

$= 0.48$

Gini (Outlook = Rainy) $= 1 - (3/5)^2 - (2/5)^2$

$= 1 - 0.36 - 0.16$

$= 0.48$

Gini Index (Outlook) $= (4/14)\times 0 + (5/14)\times 0.48 + (5/14\times 0.48)$

$= 0.342$

b. Temperature

It can take three values - hot, mild and cool. Summarising the decision for temperature features,

Temperature\Class	**Yes**	**No**	**Total**
Hot	2	2	4
Mild	4	2	6
Cool	3	1	4

Table 3.2: Temperature summary of Play Tennis dataset

Gini (Temperature = Hot) $= 1 - (2/4)^2 - (2/4)^2$

$= 1 - 0.25 - 0.25$

$= 0.5$

Gini (Temperature = Mild) $= 1 - (4/6)^2 - (2/6)^2$

$= 1 - 0.444 - 0.111$

$= 0.445$

Gini (Temperature = Cool) = $1 - (3/4)^2 - (1/4)^2$

$= 1 - 0.5625 - 0.0625$

$= 0.375$

Gini Index (Temperature) = $(4/14)\times0.5 + (6/14)\times0.445 + (4/14\times0.375)$

$= 0.439$

c. Humidity

It can take two values - high and normal. Summarising the decision for humidity,

Humidity\Class	**Yes**	**No**	**Total**
High	3	4	7
Normal	6	1	7

Table 3.3: Humidity summary of Play Tennis dataset

Gini (Humidity = High) = $1 - (3/7)^2 - (4/7)^2$

$= 1 - 0.184 - 0.327$

$= 0.489$

Gini (Humidity = Normal) = $1 - (6/7)^2 - (1/7)^2$

$= 1 - 0.735 - 0.020$

$= 0.245$

Gini Index (Humidity) = $(7/14)\times0.489 + (7/14)\times0.245$

$= 0.367$

d. Wind

It can take two values - strong and weak. Summarising the decision for wind,

Wind\Class	**Yes**	**No**	**Total**
Strong	3	3	6
Weak	6	2	8

Table 3.4: Wind summary of Play Tennis dataset

Gini (Wind = Strong) $= 1 - (3/6)^2 - (3/6)^2$

$= 1 - 0.25 - 0.25$

$= 0.5$

Gini (Wind = Weak) $= 1 - (6/8)^2 - (2/8)^2$

$= 1 - 0.5625 - 0.0625$

$= 0.375$

Gini Index (Wind) $= (6/14)\times 0.5 + (8/14)\times 0.375$

$= 0.428$

Attribute	Gini Index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

Table 3.5: Gini index values for root node #0.

Since the value of gini index for the attribute 'Outlook' is minimum, it forms the root node of the tree, and the dataset is divided into three subsets - one with outlook = sunny, one with outlook = overcast and the last one with outlook = rainy, as shown in Figure 3.1.

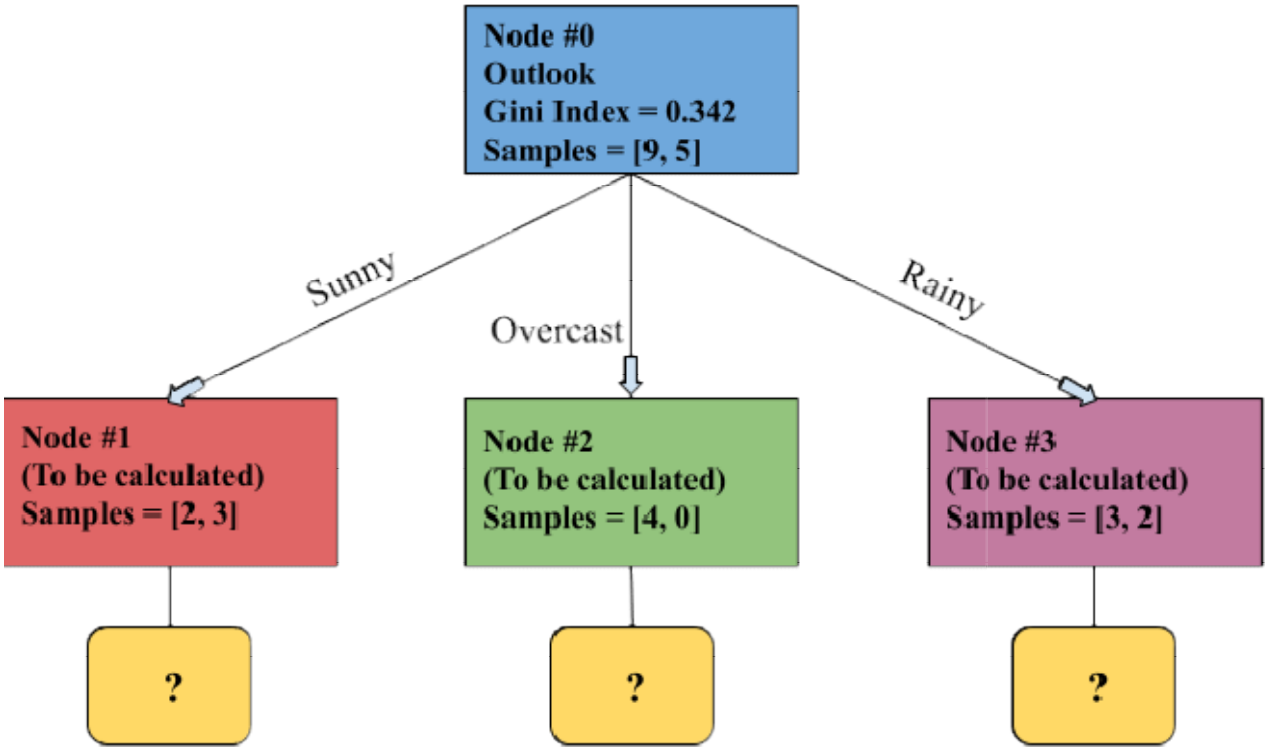


Figure 3.1: First/root node of decision tree.

The new subsets for each node are as shown in Tables 3.6, 3.7 and 3.8.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
4	Sunny	Mild	High	Weak	Yes
5	Sunny	Cool	Normal	Weak	Yes
6	Sunny	Cool	Normal	Strong	No
10	Sunny	Mild	Normal	Weak	Yes
14	Sunny	Mild	Hot	Strong	No

Table 3.6: New subset for node #1.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

Table 3.7: New subset for node #2.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Rainy	Hot	High	Weak	No
2	Rainy	Hot	High	Strong	No
8	Rainy	Mild	High	Weak	No
9	Rainy	Cool	Normal	Weak	Yes
11	Rainy	Mild	Normal	Strong	Yes

Table 3.8: New subset for node #3.

Next, the gini index values for the rest of the attributes in each subset are calculated, and split again. Splitting of the tree is continued till all the instances have been correctly classified or the desired accuracy has been attained. The final decision tree is as in Figure 3.2.

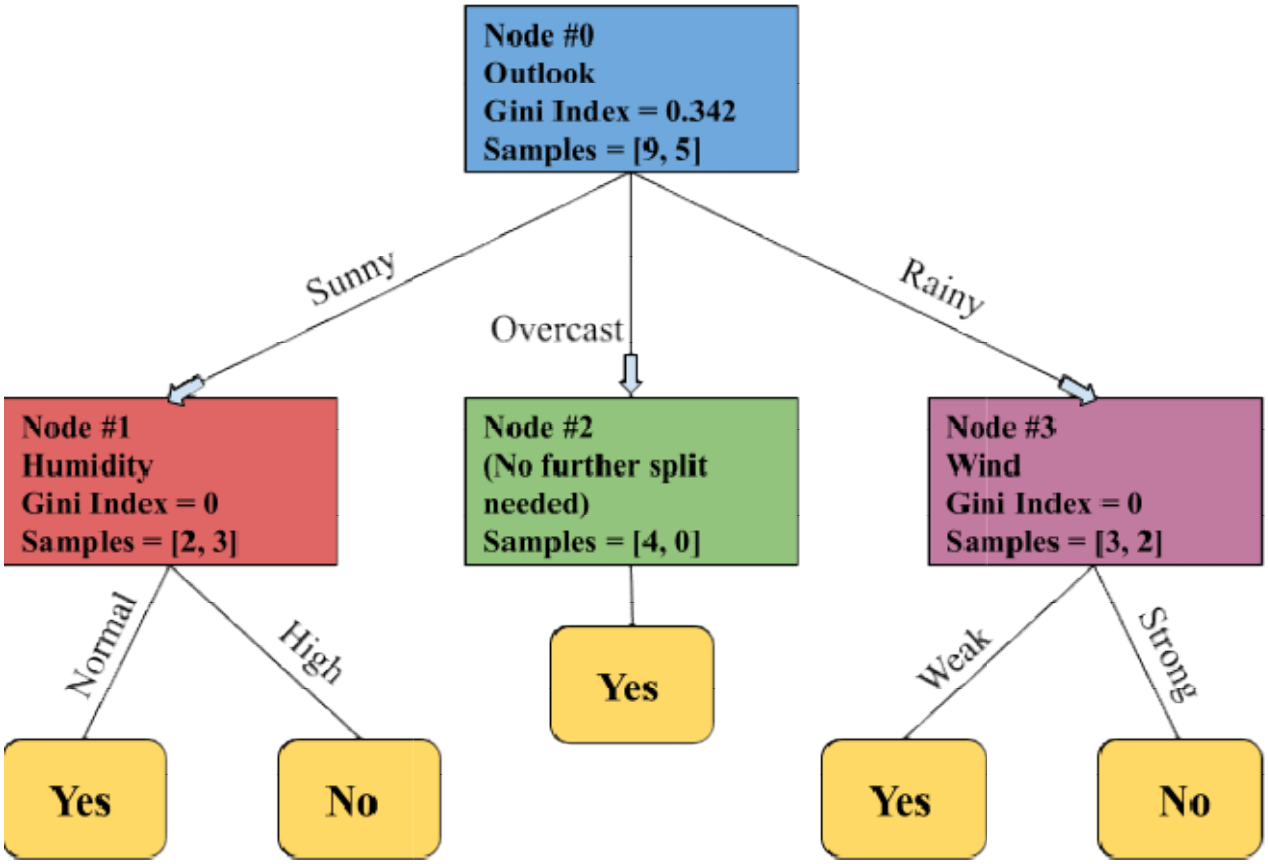


Figure 3.2: The resultant decision tree.

In case the values of different attributes are not discrete like in the example explained above, bins are created. For continuous values, the average value of attribute in a subset is considered to be the splitting point.

3.1.4. Pseudo code

GenerateDecisionTree (Set of instances S, List of attributes A):

1. Create a node *N*;
2. If all the samples of the set belong to the same class *C*, then, label *N* with *C; terminate;*
3. If *A* is empty, label *N* with the most common class *C* of *S*; terminate;
4. G = 1; *a;*
5. For attribute a_i in *A*:

   g = *GiniIndex(S,* a_i)

   If g < G:

   $a = a_i$

   G = g
6. For each value *v* of *a*:

   Create a branch from *N*, with condition $a = v$

   Let S_v be the subset of *S*, for which $a = v$

   If S_v is empty, attach a leaf node with *C* as the most commonclass in *S*,

   Else, attach node generated by *GenerateDecisionTree (*S_v, *A-a)*
7. Return node N;

3.2. Naïve Bayes

3.2.1. Introduction

Naïve Bayes is a type of supervised leaning algorithm in machine learning, that is, the model is initially "taught" using some sample example inputs and their corresponding outputs, trained until a desired accuracy/efficiency is acquired. As is clear from the name itself, it is based primarily on "Bayes theorem".

But, why naïve? The reason this method is called naïve is because a naïve assumption is made when dealing with classification problems by this method, the assumption being that every pair of feature being classified is conditionally independent of each other given the value of class variable. Thus it is assumed that all the features are independent of each other.

3.2.2 The Mathematics and Terminology involved:

In the 1700s, Reverend T. Bayes initiated a research, concluded by none other than Pierre-Simon Laplace to give the popular expression for Inverse Conditional Probability, as follows:

$$P\left(\frac{A}{B}\right) = \frac{P\left(\frac{B}{A}\right).P(A)}{P(B)}$$

Equation 3.5

The above equation determines the probability of occurrence of event A, given event B's, also called evidence, occurrence has been recorded. This evidence is an attribute value of an unknown instance.

Terminology involved:

- *P(A)*: It is the *prior probability of preposition*, i.e. the probability of occurrence of given preposition prior to the evidence is recorded.
- *P(B)*: It is the *prior probability of evidence*, i.e. the probability of occurrence of evidence.
- *P(B/A)*: It is called the *likelihood*.
- *P(A/B)*: It is called the *posterior probability*, i.e. probability of occurrence of the given preposition given the evidence is already recorded. This posterior probability for occurrence of event A given event B has already occurred is calculated.

Naïve Bayes is a simple to implement, yet an efficient method for predictive modeling of data, which can be implemented for binary as well as multiclass classification problems.

3.2.3 Gaussian Naïve Bayes classifier:

The classifier employed in this application is the Gaussian Naïve Bayes classifier. This is an upgrade to the traditional Naïve Bayes version of classifier, wherein the continuous values associated with every feature is assumed to follow a specific distribution curve, known as the Gaussian distribution curve, or the Normal distribution Curve. The shape of this curve is bell-like, symmetrically spread around the axis passing through the mean of the feature values. For this distribution, the conditional probability equation takes the form:

$$P\left(\frac{x_i}{y}\right) = \frac{1}{\sqrt{2\pi{\sigma_i}^2}} \exp\left(\frac{-(x_i - \mu_y)^2}{2{\sigma_y}^2}\right)$$

Equation 3.6

This expression has a shorthand notation as follows:

$$P\left(\frac{x_i}{y}\right) = \emptyset\left(\frac{x}{\mu_y}, \sigma_i^2\right)$$

Equation 3.7

3.2.4 Illustration

The main methodology behind working of Naïve Bayes classifier is explained using the tennis game dataset.

When it comes to working with a dataset, the Bayes theorem can be used as:

$$P\left(\frac{Y}{X}\right) = \frac{P\left(\frac{X}{y}\right)P(y)}{P(X)}$$

Equation 3.8

Here X is the feature vector, sized n, and y is the class variable. For example in the 4th data value given in the dataset, the corresponding X and Y will be:

X = (Sunny, Mild, High, Weak)

Y= Yes

Hence the above values imply the probability of playing tennis when the weather conditions can be expressed as Sunny, with mild temperature, high humidity and no strong wind blowing.

The naïve part of the Naïve Bayes classifier specifies that the class variable is dependent of each feature explicitly, which modifies equation to:

$$P\left(\frac{y}{x_1, x_2, x_3, x_4}\right) = \frac{P\left(\frac{x_1}{y}\right).P\left(\frac{x_2}{y}\right).P\left(\frac{x_3}{y}\right).P\left(\frac{x_4}{y}\right).P(y)}{P(x_1).P(x_2).P(x_3).P(x_4).}$$

Equation 3.9

Here the products of individual class probabilities can be clubbed together in a product as:

$$P\left(\frac{y}{\vec{X}}\right) = \frac{P(y).\prod_{i=1}^{n} P\left(\frac{x_i}{y}\right)}{\prod_{i=1}^{n} P(x_i)}$$

Equation 3.10

Now as for all possible values of class variables, the value of denominator will remain a constant, hence for the sake of developing a predictive classification model, as we are not interested in the exact values of probability of occurrence of individual class variable but which one will have the maximum value of probability, the denominator is neglected and values of numerator for the given expression are found out, and one with the highest value comes out as the outcome of our model, as the relation held can be expressed as follows:

$$P\left(\frac{y}{\vec{X}}\right) \propto \; P(y).\prod_{i=1}^{n} P\left(\frac{x_i}{y}\right) = P_h\,(y, \overrightarrow{X})$$

Equation 3.11

Thus pre-computed class probabilities *P(y)* and conditional probabilities $P\left(\frac{x_i}{y}\right)$ are needed. Carrying on these pre-computations, the following sets of data for conditional probabilities can be formed.

a. *Outlook*

Outlook\Class	Yes	No	P(Yes)	P(No)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	1	1

Table 3.9: Probability table for outlook.

b. *Temperature*

Temperature\Class	Yes	No	P(Yes)	P(No)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	4/5
Cool	3	1	3/9	1/5
total	9	5	1	1

Table 3.10: Probability table for temperature.

c. *Humidity*

Humidity\Class	Yes	No	P(Yes)	P(No)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	1	1

Table 3.11: Probability table for humidity.

d. Wind

Wind\Class	Yes	No	P(Yes)	P(No)
Weak	6	2	6/9	2/5
Strong	3	3	3/9	3/5
Total	9	5	1	1

Table 3.12: Probability table for wind.

e. Class probability:

Class variable	Class probability
Yes	9/14
No	5/14
Total	1

Table 3.13: Probability table for various classes.

Thus pre-calculated the values of conditional probabilities ($P(x_i/y)$) are obtained for every x_i in X being Outlook, Temperature, Humidity, and Wind condition. For an instance, probability that a game of tennis will happen when Outlook is sunny is 2/9 or 22.23%. Also class probabilities ($P(y_i)$) have been calculated.

Now the classifier can be put to use for random data of feature vector. For test purposes, let the feature vector used be,

$$\vec{X} = (Sunny, Hot, Normal, Weak)$$

that is, will a game of tennis take place in sunny, hot temperature, normal humidity and no wind weather conditions?

So, the highest of $P\left(\frac{yes}{\vec{X}}\right) and\ P\left(\frac{no}{\vec{X}}\right)$ need to be determined, and whichever is higher will be declared the outcome of the classifier.

Now, as for calculating the higher numerator for both values of class variables:

$$P\left(\frac{y = yes}{\vec{X}}\right) \propto P_h\left(y = yes, \vec{X}\right) = \quad P(y = yes).\prod_{i=1}^{n} P\left(\frac{x_i}{y = yes}\right)$$

$$P\left(\frac{y = no}{\vec{X}}\right) \propto P_h\left(y = no, \vec{X}\right) = \quad P(y = no).\prod_{i=1}^{n} P\left(\frac{x_i}{y = no}\right)$$

So,

$$P_{\boldsymbol{h}}\left(\frac{yes}{\vec{X}}\right) = \frac{2}{9}.\frac{2}{9}.\frac{6}{9}.\frac{6}{9}.\frac{9}{14} \approx \ 0.014$$

And,

$$P_{\boldsymbol{h}}\left(\frac{no}{\vec{X}}\right) = \frac{3}{5}.\frac{2}{5}.\frac{1}{5}.\frac{2}{5}.\frac{5}{14} \approx 0.007$$

Now, since it can be noted that because the $P_{\boldsymbol{h}}(yes)$ value greater than that of $P_{\boldsymbol{h}}(no)$, the prediction for class variable for the given feature vector is “Yes”.

Also, to calculate probability values for the same, we know that $P\left(\frac{yes}{\vec{X}}\right)$ and $P\left(\frac{no}{\vec{X}}\right)$add up to give 1. Hence the values can be determined as:

$$P\left(\frac{yes}{X}\right) = \frac{P_{\boldsymbol{h}}\left(\frac{yes}{\vec{X}}\right)}{P_{\boldsymbol{h}}\left(\frac{yes}{\vec{X}}\right) + P_{\boldsymbol{h}}\left(\frac{no}{\vec{X}}\right)} \approx 0.667$$

$$P\left(\frac{no}{X}\right) = \frac{P_{\boldsymbol{h}}\left(\frac{no}{\vec{X}}\right)}{P_{\boldsymbol{h}}\left(\frac{yes}{\vec{X}}\right) + P_{\boldsymbol{h}}\left(\frac{no}{\vec{X}}\right)} \approx 0.333$$

3.2.5 Pseudo code

Classifier (Dataset T):

1. Read the training dataset T;
2. Calculate the mean and standard deviation of the predictor variables in each class;
3. Repeat

   Calculate the probability of f_i using the gauss density equation in each class;

   Until the probability of all predictor variables ($f_1, f_2, f_3 ..... f_n$) has been calculated.
4. Calculate the likelihood for each class;
5. Return the greatest likelihood;

3.2.6 Performance on Iris Dataset:

The dataset for tennis game was discrete, hence the fore-mentioned method worked fine. But for continuous dataset, some assumptions are needed to be made regarding the distribution of the values of each feature within the dataset.

Here the continuous distribution of feature space is considered to follow Gaussian distribution, i.e. the working formula takes the form:

$$P\left(\frac{x_i}{y}\right) = \frac{1}{\sqrt{2\pi{\sigma_y}^2}} \exp\left(\frac{-(x_i - \mu_y)^2}{2{\sigma_y}^2}\right)$$

Equation 3.12

Here the terminology involved is as follows:

- $P\left(\frac{x_i}{y}\right)$: Probability density function owing to continuous distribution for X vector.

- $\sigma_{y,i}$: denotes the *standard deviation* for i[th] feature in feature space for each class variable in training dataset.

$$\sigma_{y,i} = \sqrt{\frac{1}{\sum_{i=1}^{n} X_i(y)} \left(\sum_{i=1}^{n} X_i(y) \,.\, (x_{k,i} - \mu_y)^2 \right)}$$

Equation 3.13

- μ_y: denotes the *mean value* for i[th] feature in feature space for each class variable in the training dataset.

$$\mu_{y,i} = \frac{1}{n} \sum_{i=1}^{n} X_i(y) . x_{k,i}$$

Equation 3.14

Both σ and μ values are calculated during training of the model, and are fixed in the probability density function to determine the probability for a given feature vector during testing phase to determine the class variable that yields maximum probability value.

This shorthand for the given notation is represented as follows:

$$N\left\{P\left(\frac{x_i}{y}\right)\right\} = \emptyset\left(x_i, \mu_{i,y}, \sigma_{i,y}^2\right)$$

Equation 3.15

Here N denotes the numerator part of the given expression for probability density function, as for a given feature, the denominator stays a constant for all class variables, hence can be neglected as was depicted in the Discrete Naïve Bayes application in the tennis game dataset.

The iris flower classifier involves predicting the species for a given iris flower given its feature vector. The dataset consists of 150 different data values and there are 4 different feature values per feature vector that classify the flower into one of three main classes. The features include Sepal Length, Sepal width, petal length and petal width, and classes being Iris Setosa, Iris Versicolor, and Iris Virginica.

2D classification for iris dataset:

In the above equations, y denotes the class variable, i.e. the species of the flower, having three distinct values. The two elementary variables considered are:

X_1 : Sepal length and X_2: sepal width.

For any given elementary variable set $(x_1,\ x_2)$, we need to evaluate the expression for Gaussian Naïve Bayes objective function for all 3 values of class variables, i.e. Setosa, Versicolor, and Virginica.

$$P_h(\vec{X}, y) = \prod_{i=1}^{2} P\left(\frac{x_i}{y}\right).P(y)$$

$$= \{\emptyset(x_1, \mu_{1,y}, \sigma_{1,y}).\emptyset(x_2, \mu_{2,y}, \sigma_{2,y}).P(y)\}$$

Equation 3.16

The above expression is analogous to that used in discrete Naïve Bayes classifier, the only change being that the calculation of class probabilities involves Gaussian distribution adjustment.

Hence the three values are calculated for the three class variables:

$$P_h(\vec{X}, y = Setosa) = \emptyset(x_1, \mu_{1,y}, \sigma_{1,y}).\emptyset(x_2, \mu_{2,y}, \sigma_{2,y}).P(y)$$

$$P_h(\vec{X}, y = Versicolor) = \emptyset(x_1, \mu_{1,y}, \sigma_{1,y}).\emptyset(x_2, \mu_{2,y}, \sigma_{2,y}).P(y)$$

$$P_h(\vec{X}, y = Verginica) = \emptyset(x_1, \mu_{1,y}, \sigma_{1,y}).\emptyset(x_2, \mu_{2,y}, \sigma_{2,y}).P(y)$$

These values are compared, and the given feature vector is classified into the class which yields the maximum value of $P_h(\vec{X}, y)$.

CHAPTER 4

4. Comparison

Effectiveness measurement:

There are 4 measures of effectiveness based on comparison using confusion matrix, which include:

a. True Positive (TP): Positive observations, that are predicted to be positive as well.
b. True Negative (TN): Negative observations, that are predicted to be negative as well.
c. False Positives (FP): Negative observations, that are predicted to be positive.
d. False Negative (FN): Positive observations that are predicted to be negative.

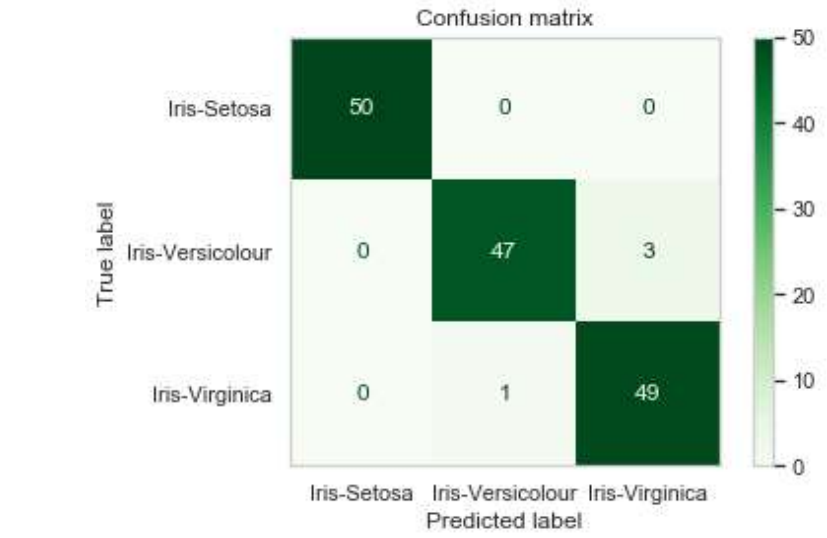
4.1 Confusion matrix:

This matrix summarizes the classification results of a problem, depicting clearly the TP, TN, FP and FN for the class variables involved. More the number of True Positives and True Negatives in the overall observations more is the correctness of the classification method. The structure of confusion matrix is as in Table 4.1.

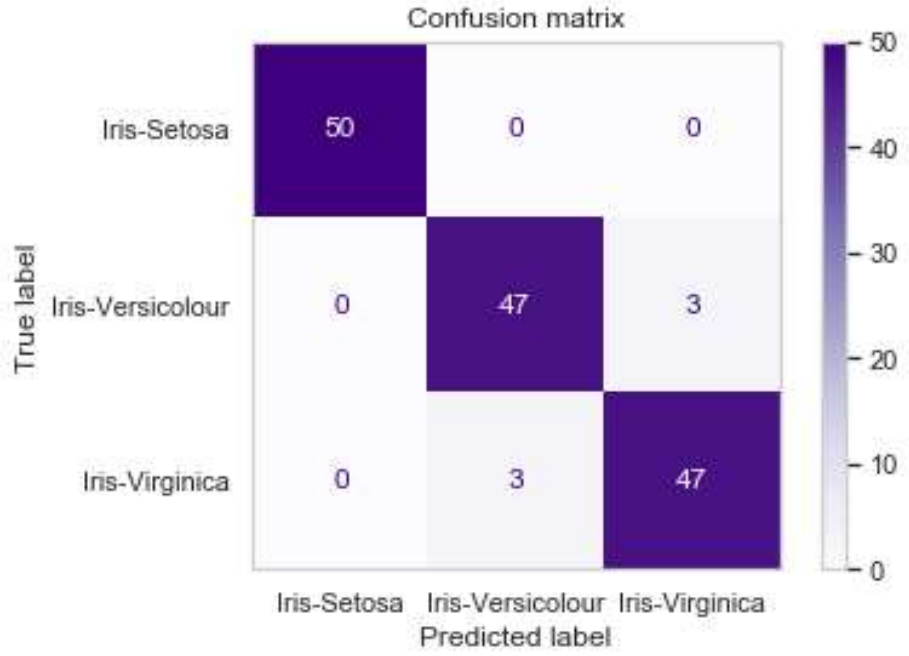
		Predicted Label	
		Positive prediction	Negative prediction
True Label	Actual Positive	True Positive	False Positive
	Actual Negative	False negative	True Negative

Table 4.1: Format of confusion matrix.

The confusion matrices generated in the study are as in Fig. 4.1 for iris dataset and in Fig. 4.2 for bank note dataset.

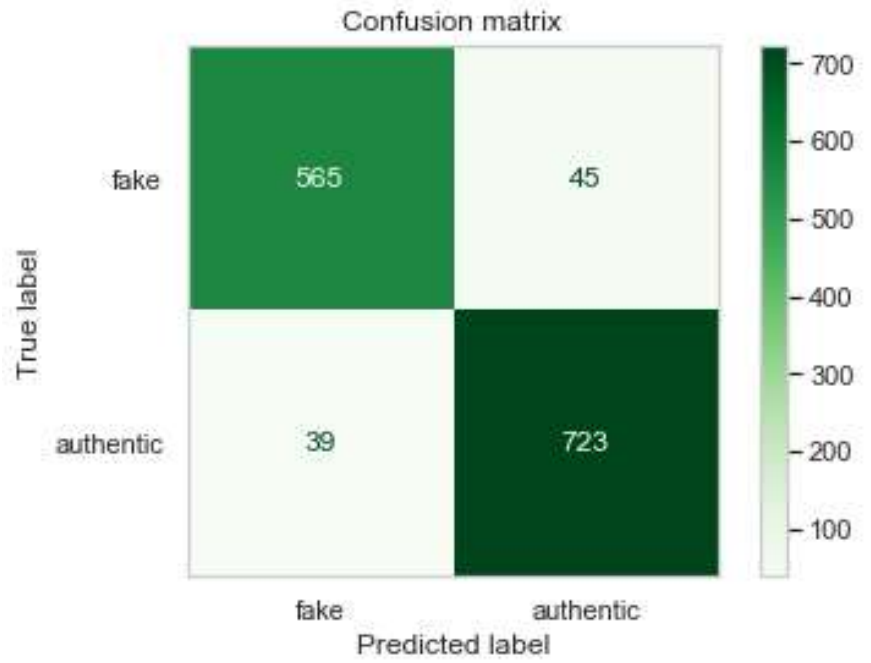


(i)

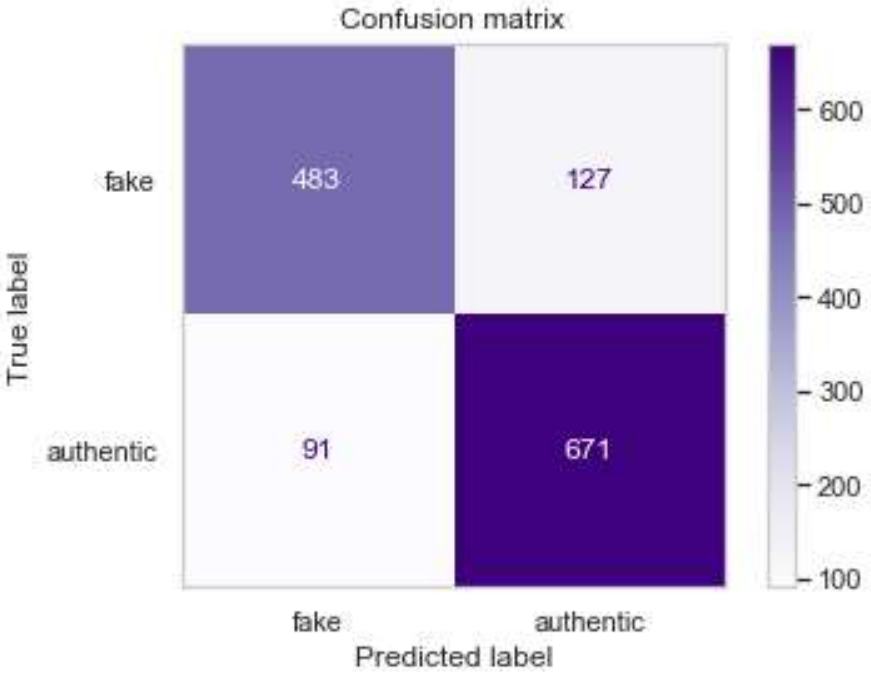


(ii)

Figure 4.1: Confusion matrix for Iris dataset using (i) Decision Tree and (ii) Naive Bayes.



(i)



(ii)

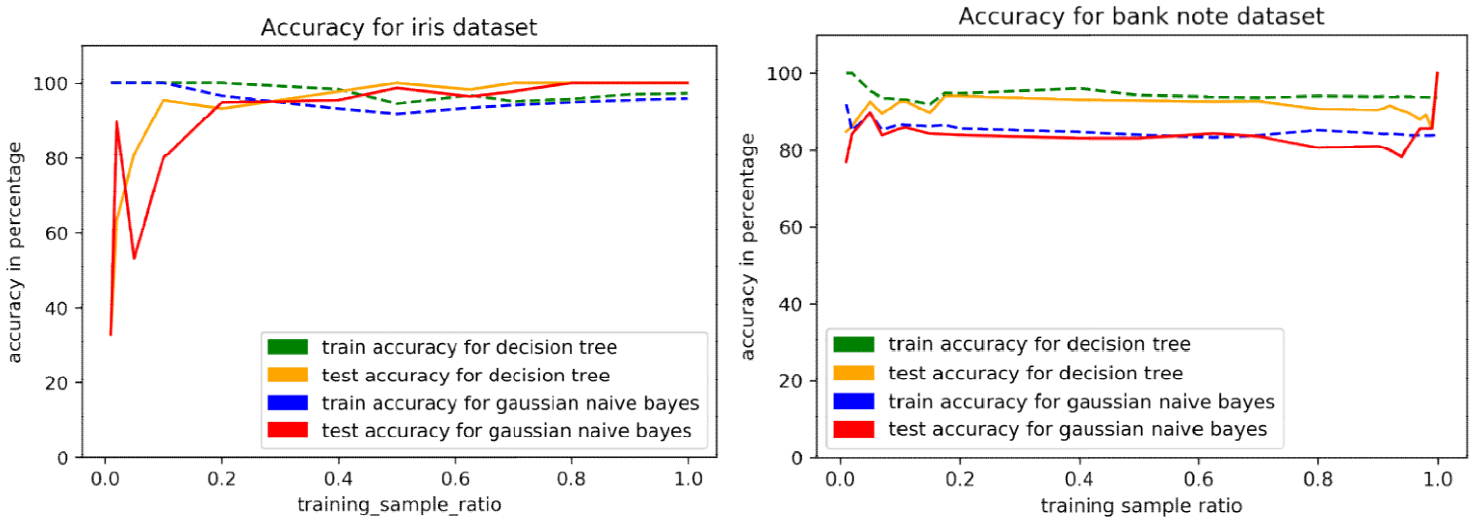
Figure 4.2: Confusion matrix for bank note dataset with (i) Decision Tree and (ii) Naive Bayes.

4.2 Accuracy:

The ratio of all **true** predicted instances against all the predictions made by the classifier is termed as the accuracy of a classifier. Accuracy can be calculated as:

$$Accuracy\ (A) = \frac{TP + TN}{TP + TN + FP + FN}$$

The accuracy performance of the different models on different dataset is as in Fig. 4.3.



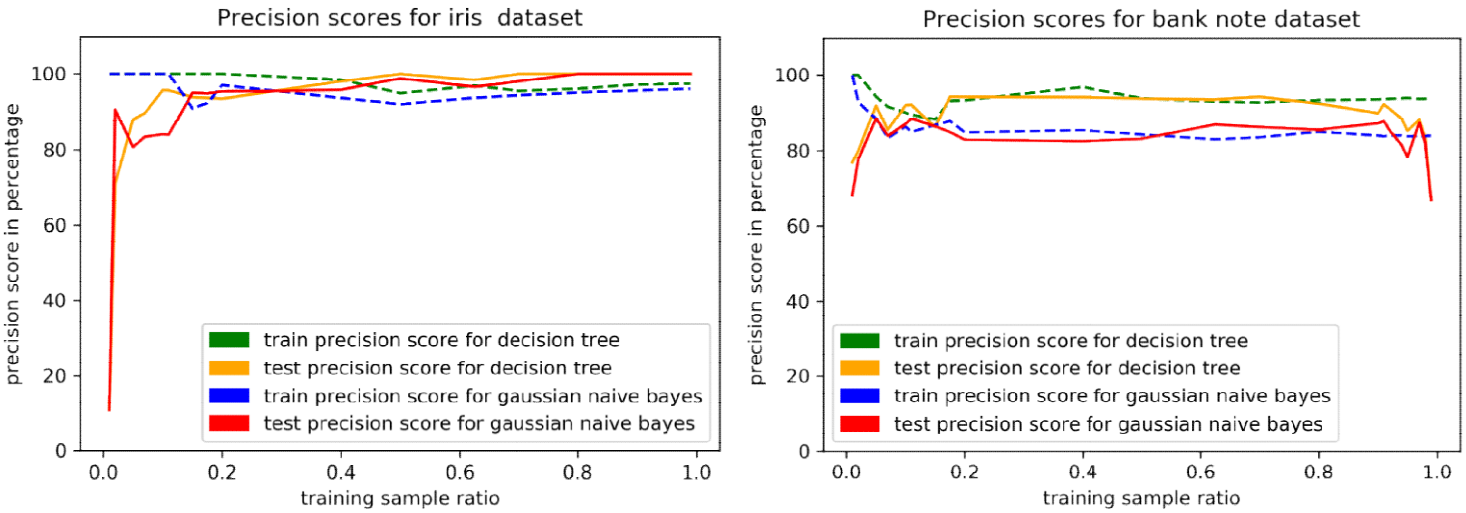
(i) (ii)

Figure. 4.3.: Accuracy score for (i) iris and (ii) bank note dataset for various train ratios.

4.3 Precision and Recall

Precision, in simple terms, is the ability of a classifier to not predict an instance positive, if it's not a true positive, i.e. if the classifier predicts zero False Positives, we say that the classifier is 100% precise.
Mathematically,

$$Precision(P) = \frac{TP}{TP + FP}$$

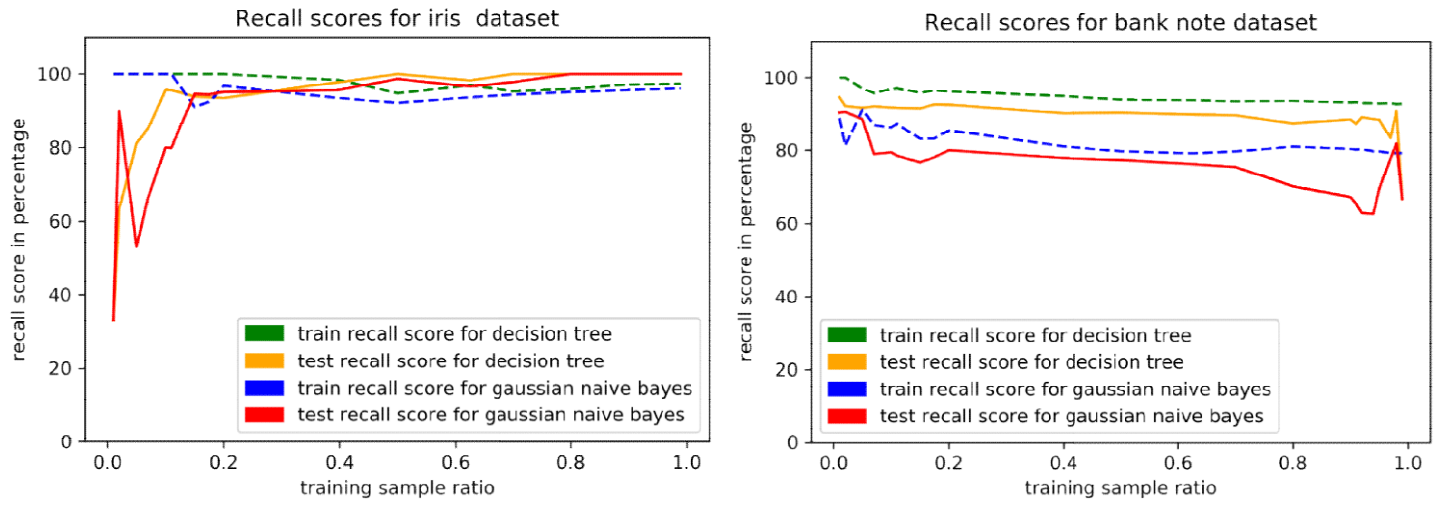


(i) (ii)

Figure. 4.4.: Precision score for (i) iris and (ii) bank note dataset for various train ratios.

Recall, is the ability of a classifier to predict all True Positives correctly, i.e. if there are no False Negatives, that means the classifier has not given a wrong prediction for any True Positive, hence its value for recall is 1. Mathematically,

$$Recall\ (R) = \frac{TP}{TP + FN}$$



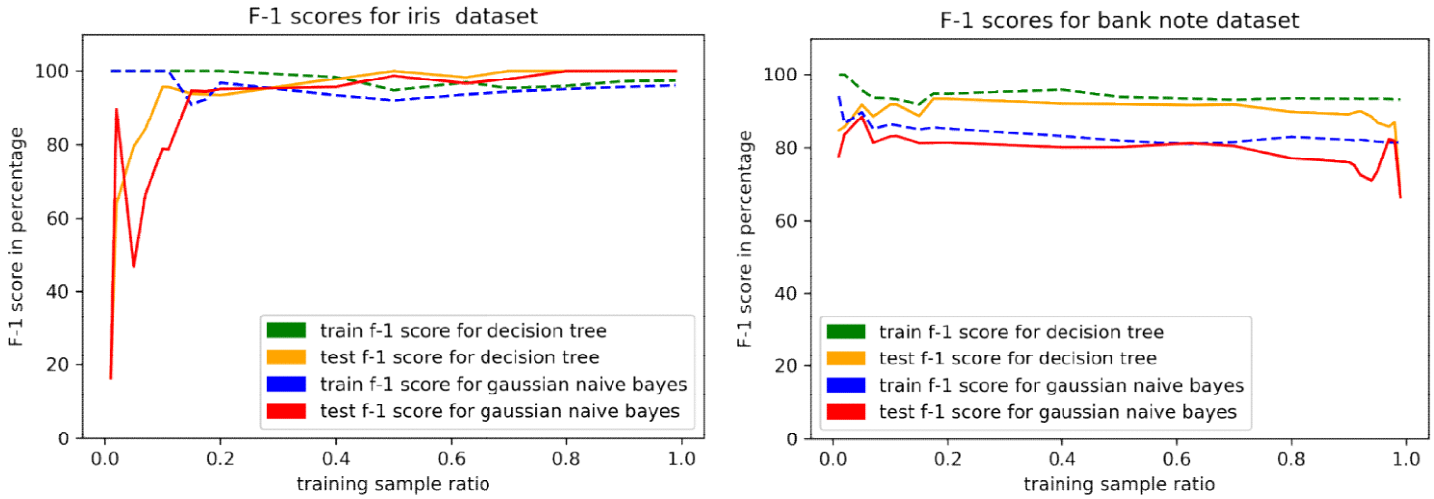
(i) (ii)

Figure. 4.5.: Recall score for (i) iris and (ii) bank note dataset for various train ratios.

4.3 F-1 Score

F1 score is a harmonic average of Precision and Recall. Mathematically,

$$F-1\ score = \frac{2.P.R}{P+R}$$



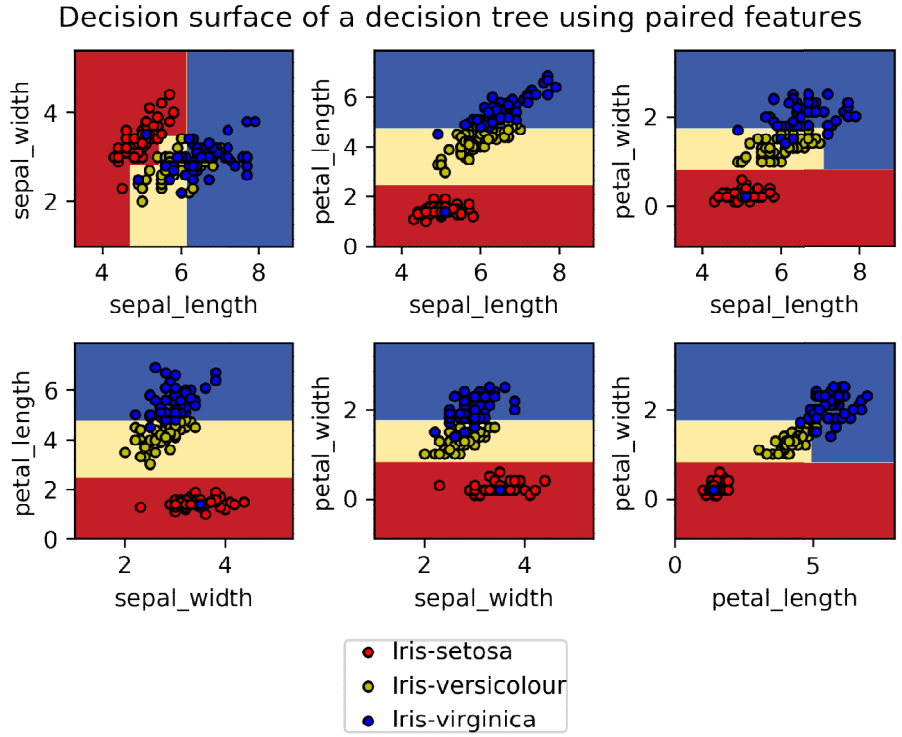
(i) (ii)

Figure. 4.6.: F-1 score for (i) iris and (ii) bank note dataset for various train ratios.

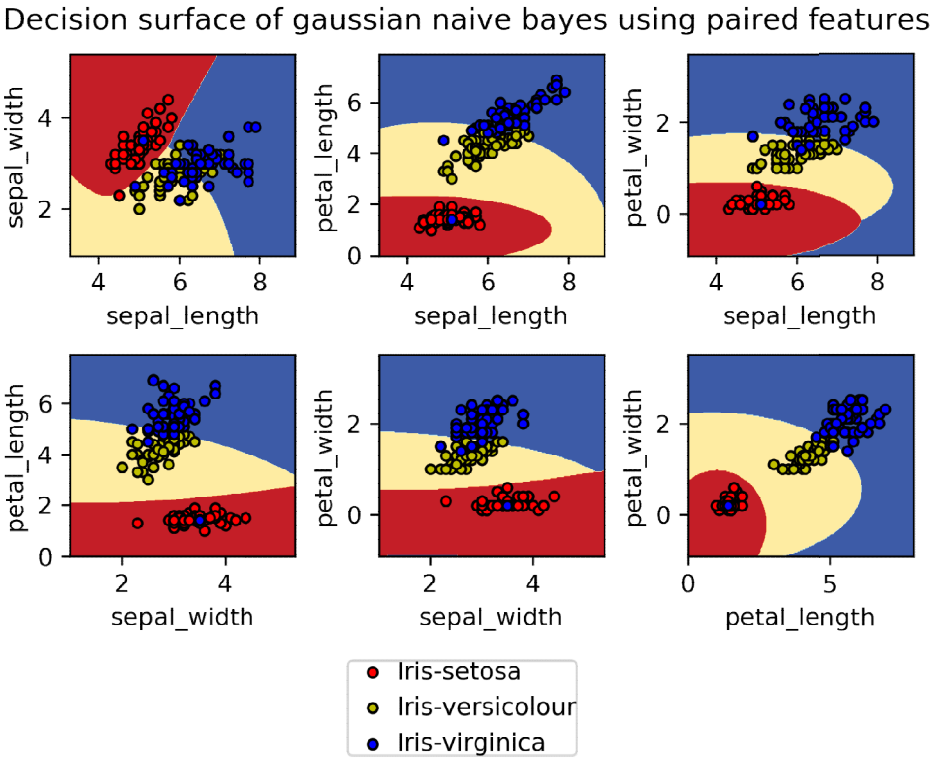
This parameter is most commonly used after accuracy score. When this score it used for performance measurement, it helps in more accurate judgement in class imbalanced data. For datasets where there are significantly more instances of one class over the other, accuracy score will be good even if the algorithm predicts all the instances belonging to the majority class, F-1 score measure helps in better understanding the result.

4.4 Decision Boundary

The decision boundaries of the datasets using different attributes are as in Fig. 4.4 and Fig. 4.5.

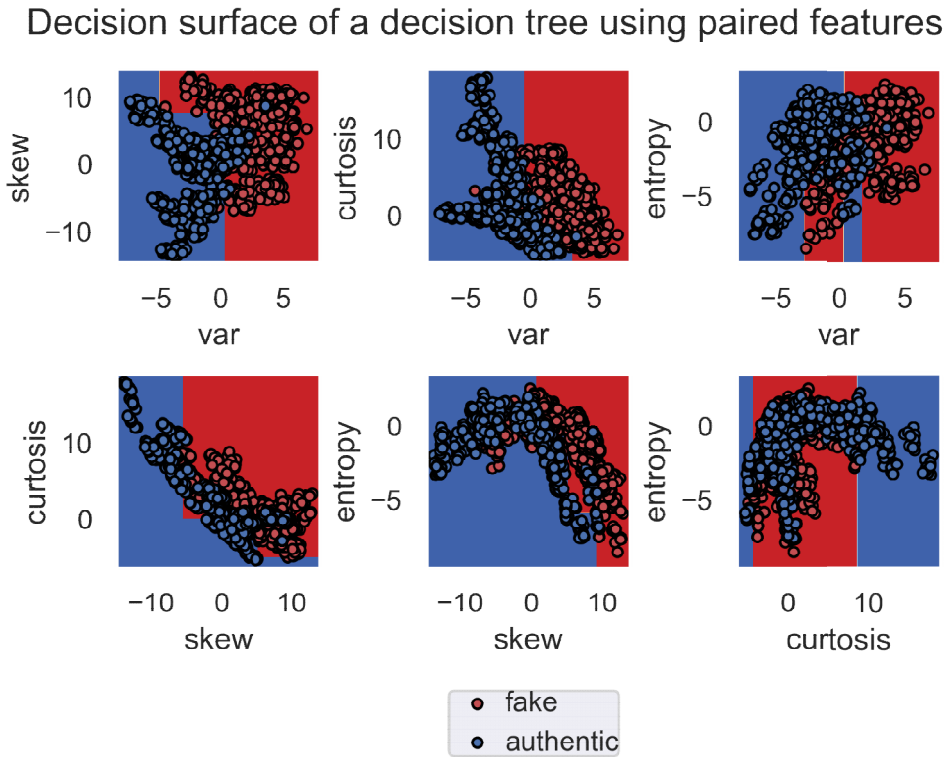


(i)

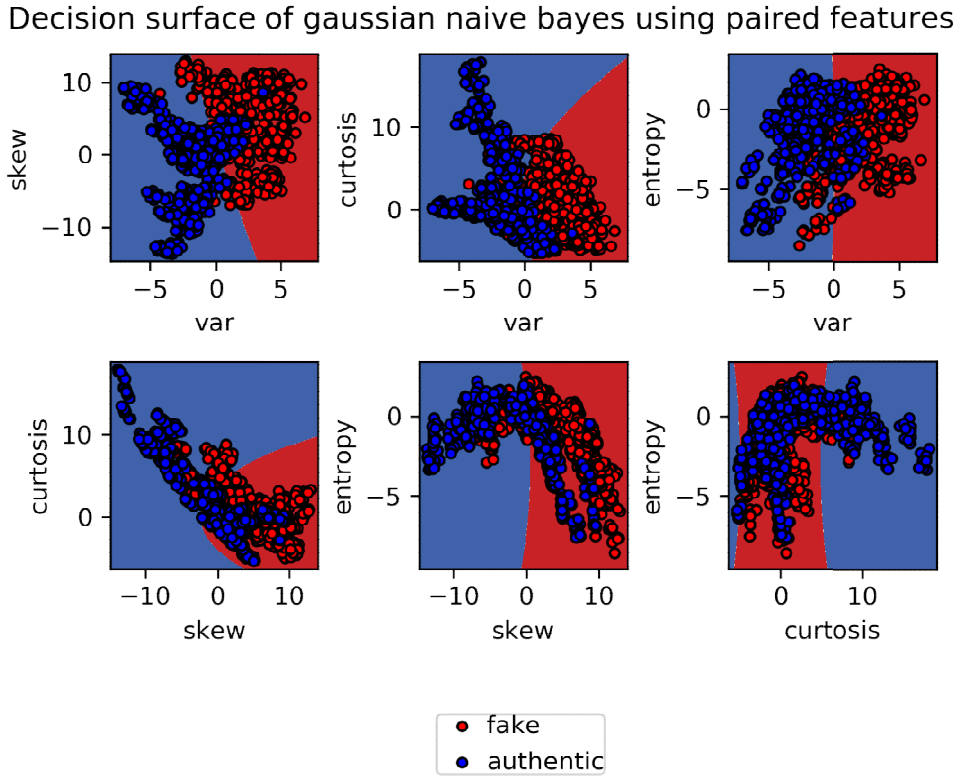


(ii)

Figure 3.5: Pair-wise plot of decision boundaries for Iris dataset using (i) Decision Tree and (ii) Naive Bayes.



(i)



(ii)

Figure 3.5: Pair-wise plot of decision boundaries for bank note dataset using (i) Decision Tree and (ii) Naive Bayes.

CHAPTER 5

5. Conclusion and Discussion

The whole work performed is compiled, complete with involved mathematics and theorem derivations, calculations, dataset analysis and classification. Major focus is kept on analysing the functioning of the two major algorithms used in the field of Machine learning, i.e. the Decision tree and Gaussian Naïve Bayes algorithm. A simple analysis using the Play Tennis database has been performed using both the classifiers in their core form using the very basic foundations of the algorithms themselves without including the machine (computer) and the computations, pre-calculations and predictions are performed manually. The pseudo codes for the algorithms were also mentioned.

Libraries like sklearn, numpy and matplotlib were used to visualise iris dataset and bank note dataset, and to perform calculation on Python 3.7. Scatter plots have been plotted for the Iris dataset and bank note dataset, and the models have been trained using the available data. Sufficient accuracy levels were registered, and Decision Tree approach was established as relatively better fit algorithm compared to the Naïve Bayes approach.

Decision trees

Decision Tree algorithm is excellent for distinct type data – for categorical and ordinal type variable. In order to create models containing continuous data, the data values are bucketed into intervals. The algorithm mentioned in this project cannot handle missing values, so it is important to use thoroughly clean data for this. Hyperparameters include the maximum depth of the tree, minimum number of instances in leaf nodes, minimum weighted fraction of the sum total of weights required to be at a leaf node (samples have equal weight when sample weight is not provided, this is to deal with class imbalance), minimum impurity decrease (a node will be split if this split induces a decrease of the impurity greater than or equal to this value), class weights (again, to handle class imbalance), and random state. Decision

trees tend to overfit the data so for practical applications, it is necessary to split the data into training and test sets in order to evaluate the performance of the algorithm. Overfitting can be prevented by fine-tuning the various hyperparameters and by a method called tree-pruning. Tree-pruning involves "clipping off" the bottom of the tree to result in greater accuracy on training as well as test data.

Using this basic algorithm, several newer and advanced algorithms have been developed. These include Random Forest – this algorithm involves creating a "forest" of trees using different features, random states and other values of hyperparameters and then predicting by majority vote; and Gradient Boosting Tree – this algorithm is similar to random forest, but involves a varied variety of trees, creating subsets of data by subsampling along the rows and columns of the data, and using gradient descent to find the best learners. The latest algorithms are even capable of handling missing values.

Overall, decision tree is a simple, yet powerful algorithm and is used by academicians and industry researchers in initial stages for exploring data. In some cases, its advanced version is used for final application too, giving good results.

Naive Bayes

The naive Bayes is used usually with different kinds of distribution and assumptions to solve classification problems in different scenarios. Some of the most prevalent of them include multinomial and Bernoulli methods.

Multinomial Naive Bayes, the naive Bayes classifier which is mostly employed for document classification problem, i.e. to classify which category a particular document or data block belongs, be it sports, technology, industry specifics etc. The feature vectors used for prediction as the basis of classification include usually the words and frequency maps in the document.

Bernoulli Naive Bayes, is in many ways similar to multinomial naive Bayes classifier, but the predictors in this case can take only true/false values, i.e. are two

valued, or Boolean to be more specific. The parameters used to predict the values of class variables in this classifier also take up Boolean values.

Their time complexity is lesser than their other time or resource intensive supervised learning counterparts, but the major disadvantage of this approach is the clause of “naivety” in the method, which is the requirement of predictors to be independent so that class probabilities can be considered individually. In most of the real life cases, the predictors and feature space involved in the prediction and classification are dependent of one another, and this hinders the overall performance of the method.

Lastly,

Understanding of machine learning algorithms is very important, but what is necessary is the understanding of data – if it is structured or unstructured, the origin of data, underlying patterns and correlations, and the type of analysis needed for solving a particular problem. An algorithm works only as good if the researcher is knowledgeable about his/her domain.

CHAPTER 6

6. Further Development

Future scope includes testing and performing comparisons of other popular supervised learning machine learning algorithms on the same dataset to bookmark their performance in similar conditions as the two used herein, or to use other datasets and compare the feasibility of the results brought about, so that more areas of this genre can be touched and explored. Advanced versions of the same algorithms may also be studied.

Along with understanding the mathematics behind various algorithms, it is also important to analyse how computationally intensive each of them is, if their models be created using parallel computing, and how long it takes to train them.

CHAPTER 7

7. References:

- Datasets:
  - Iris dataset: UCI Machine Learning Repository.
  - Play tennis dataset: https://www.kaggle.com/m1thr4nd1r/tennis
  - Bank note dataset: UCI Machine Learning Repository.

- Fisher, R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to Mathematical Statistics" (John Wiley, NY, 1950).

- Machine Learning by Tom Mitchell

- An Introduction to Statistical Learning: With Applications in R by Gareth M. James, Trevor Hastie, Daniela Witten and Robert Tibshirani

- Kang Hanhoon, Yoo Seong Joon, Han Dongil., "Senti-lexicon and improved Naive Bayes algorithms for sentiment analysis of restaurant reviews",

- Zhang, Harry. (2004). the Optimality of Naive Bayes. Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004. 2

- https://www.oreilly.com/library/view/thoughtful-machine-learning/9781449374075/ch04.html