

Project Report

On

**STUDY ON CLASSIFICATION
ALGORITHMS OF MACHINE LEARNING**

Submitted

in partial fulfilment of the requirements for the degree of

BACHELOR OF SCIENCE

in

MATHEMATICS

by

RAMEEZA V M

(Register No. AB21BMAT001)

Under the Supervision of

DR. ELIZABETH RESHMA M T



DEPARTMENT OF MATHEMATICS

ST. TERESA'S COLLEGE (AUTONOMOUS)

ERNAKULAM, KOCHI - 682011

APRIL 2024

ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM



CERTIFICATE

This is to certify that the dissertation entitled, **STUDY ON CLASSIFICATION ALGORITHMS OF MACHINE LEARNING** is a bonafide record of the work done by Ms. **RAMEEZA V M** under my guidance as partial fulfillment of the award of the degree of **Bachelor of Science in Mathematics** at St. Teresa's College (Autonomous), Ernakulam affiliated to Mahatma Gandhi University, Kottayam. No part of this work has been submitted for any other degree elsewhere.

Date:16-02-2024

Place: Ernakulam

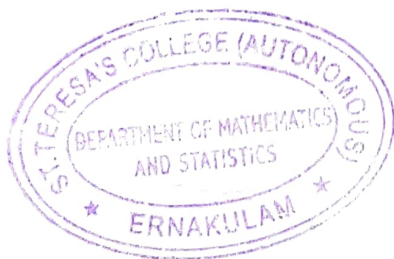
Dr. Elizabeth Reshma M T

Assistant Professor,

Department of Mathematics,

St.Teresa's College (Autonomous),

Ernakulam.



Dr. Ursala Paul

Assistant Professor and Head,

Department of Mathematics,

St. Teresa's College (Autonomous),

Ernakulam.

External Examiners

1:.....

2:

DECLARATION

I hereby declare that the work presented in this project is based on the original work done by me under the guidance of Dr. Elizabeth Reshma M T, Assistant Professor, Department of Mathematics, St. Teresa's College (Autonomous), Ernakulam and has not been included in any other project submitted previously for the award of any degree.

Place: Ernakulam.

Date: 16-02-2024



RAMEEZA V M

AB21BMAT001

ACKNOWLEDGEMENT

I must mention several individuals who encouraged me to carry this work. Their continuous invaluable knowledgeably guidance throughout the course of this study helped me to complete the work up to this stage.

I am very grateful to my project guide Dr. Elizabeth Reshma M T for her guidance and support throughout the work. I also want to express my gratitude to my classmates, who assisted me in formulating my research topics by sharing their knowledge. Last but not least, I'd like to express my heartfelt gratitude to God Almighty for always being my emotional rock and teaching me how to confront every challenge with intelligence and confidence.

Place: Ernakulam.

Date: 16-02-2024



RAMEEZA V M

AB21BMAT001

Contents

<i>CERTIFICATE</i>	<i>ii</i>
<i>DECLARATION</i>	<i>iii</i>
<i>ACKNOWLEDGEMENT</i>	<i>iv</i>
<i>CONTENTS</i>	<i>v</i>

1. MACHINE LEARNING

1.1. Evolution of machine learning.....	1
1.2. Types of machine learning.....	2
1.3. Applications of machine learning.....	4
1.4. Limitations of machine learning.....	4
1.5. Classification algorithms.....	5

2. LOGISTIC REGRESSION

2.1. Mathematical framework.....	6
2.2. Implementation of algorithm.....	11
2.3. Advantages and disadvantages.....	15

3. DECISION TREE

3.1. Mathematical framework.....	16
3.2. Implementation of algorithm.....	20
3.3. Advantages and disadvantages.....	24

4. SUPPORT VECTOR MACHINE (SVM)

4.1. Mathematical framework.....	26
4.2. Implementation of algorithm.....	36
4.3. Advantages and disadvantages.....	38

5. NAIVE BAYES	
5.1. Mathematical framework.....	40
5.2. Types of Naive Bayes classifiers.....	43
5.3. Implementation of algorithm.....	44
5.4. Advantages and disadvantages.....	48
BIBLIOGRAPHY.....	50

CHAPTER 1

MACHINE LEARNING

Machine Learning is a branch of artificial intelligence that develops algorithms by learning the hidden patterns of the datasets used it to make predictions on new similar type data, without being explicitly programmed for each task. Traditional Machine Learning combines data with statistical tools to predict an output that can be used to make actionable insights. It is a subfield of artificial intelligence that involves the development of algorithms and statistical models that enable computers to improve their performance in tasks through experience.

Arthur Samuel, an early American leader in the field of computer gaming and artificial intelligence, coined the term “Machine Learning” in 1959 while at IBM. He defined machine learning as “the field of study that gives computers the ability to learn without being explicitly programmed”

1.1 EVOLUTION OF MACHINE LEARNING

1950 - Alan Turing proposes "learning machine"

1952 - Arthur Samuel developed first machine learning program that could play Checkers

1957 - Frank Rosenblatt designed the first neural network program simulating human brain

1967 - Nearest neighbour algorithm created - start of basic pattern recognition

1979 - Stanford University students develop first self - driving cart that can navigate and avoid obstacles in a room

1982 - Recurrent Neural Network developed

1989 - Reinforcement Learning conceptualized

Beginning of commercialization of Machine Learning

1995 - Random Forest and Support Vector machine algorithms developed

1997 - IBM's Deep Blue beats the world chess champion Gary

Kasparov

2006 - First machine learning competition launched by Netflix

Geoffrey Hinton conceptualizes Deep Learning

2010 - Kaggle, a website for machine learning competitions, launched

2011 - IBM's Watson beats two human champions in Jeopardy

2016 - Google's AlphaGo program beats unhandicapped professional human player

2017 - Transfer learning becomes a prominent technique, allowing pre-trained models to be adapted to new tasks with limited data.

2018 - Transformers, a novel architecture based on self-attention mechanisms, revolutionize natural language processing tasks.

2019 - Microsoft launched the Turing Natural Language Generation generative language model with 17 billion parameters

2020 - GPT-3, developed by OpenAI, showcases the power of large-scale language models in various natural language understanding and generation tasks.

2022 - ChatGPT was released for public use.

2023 - Advancements in quantum machine learning begin to emerge, exploring the potential of quantum computing to accelerate certain machine learning algorithms.

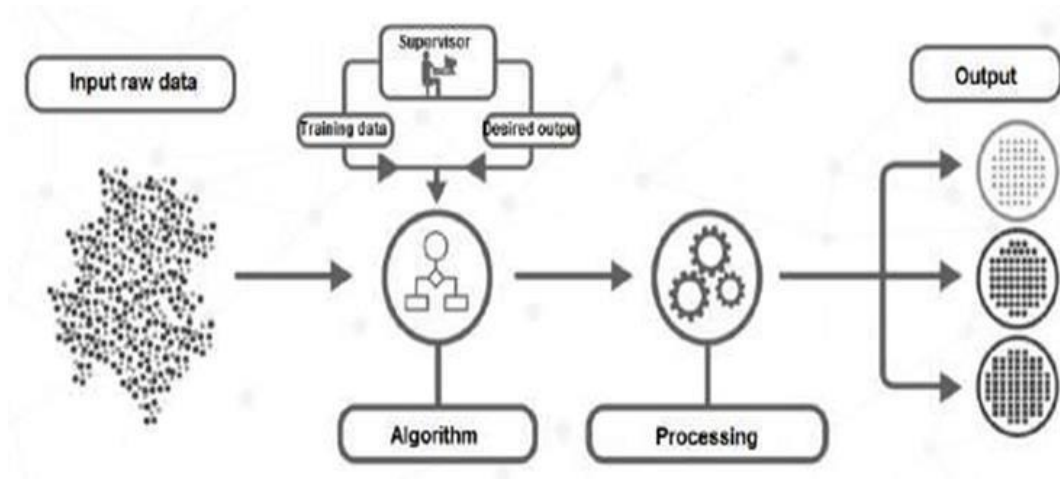
2024 - Continued progress in federated learning and privacy-preserving machine learning techniques enhances collaboration on shared datasets while preserving data privacy

1.2 TYPES OF MACHINE LEARNING

Machine learning implementations are classified into three major categories, depending on the nature of the learning “signal” or “response” available to a learning system which are as follows:

Supervised learning: Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.

In supervised learning, each example in the training set is a pair consisting of an input object (typically a vector) and an output value. A supervised learning algorithm analyses the training data and produces a function, which can be used for mapping new examples. In the optimal case, the function will correctly determine the class labels for unseen instances. Both classification and regression problems are supervised learning problems. A wide range of supervised learning algorithms are available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised learning problems.



Unsupervised learning:

Unsupervised learning is a type of machine learning algorithm used to draw from datasets consisting of input data without labelled responses. In unsupervised learning algorithms, a classification or categorization is not included in the observations. There are no output values and so there is no estimation of functions. Since the examples given to the learner are unlabelled, the accuracy of the structure that is output by the algorithm cannot be evaluated. The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data.

Reinforcement learning:

This is somewhere between supervised and unsupervised learning. The algorithm gets told when the answer is wrong, but does not get told how to correct it. It has to explore and try out different possibilities until it works out how to get the answer right. Reinforcement learning is sometime called learning with a critic because of this monitor that scores the answer, but does not suggest improvements.

Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards. A learner (the program) is not told what actions to take as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situations and, through that, all subsequent rewards.

1.3 APPLICATIONS OF MACHINE LEARNING

Companies are using Machine Learning to improve business decisions, increase productivity, detect disease, forecast weather, and do many more things. With the exponential growth of technology, we not only need better tools to understand the data we currently have, but we also need to prepare ourselves for the data we will have. To achieve this goal, we need to build intelligent machines. We can write a program to do simple things. But most of the time, Hardwiring Intelligence in it is difficult. The best way to do it is to have some way for machines to learn things themselves. A mechanism for learning – if a machine can learn from input, then it does the hard work for us. This is where Machine Learning comes into action.

Some of the most common examples are: Image Recognition, Speech Recognition, Recommender Systems, Fraud detection, Self-driving cars, Medical Diagnosis, Stock Market Trading, Virtual Try on.

1.4 LIMITATIONS OF MACHINE LEARNING

Machine learning has been transformative in some fields, machine-learning programs often fail to deliver expected results. Reasons for this are numerous:

lack of (suitable) data, lack of access to the data, data bias, privacy problems, badly chosen tasks and algorithms, wrong tools and people, lack of resources, and evaluation problems. In 2018, a self-driving car from Uber failed to detect a pedestrian, who was killed after a collision.

Attempts to use machine learning in healthcare with the IBM Watson system failed to deliver even after years of time and billions of dollars invested. Machine learning has been used as a strategy to update the evidence related to systematic review and increased reviewer burden related to the growth of biomedical literature. While it has improved with training sets, it has not yet developed sufficiently to reduce the workload burden without limiting the necessary sensitivity for the findings research themselves.

1.5 CLASSIFICATION ALGORITHMS

Classification is the process of recognizing, understanding, and grouping ideas and objects into preset categories or “sub-populations.” Using pre-categorized training datasets, machine learning programs use a variety of algorithms to classify future datasets into categories.

Classification algorithms in machine learning use input training data to predict the likelihood that subsequent data will fall into one of the predetermined categories. One of the most common uses of classification is filtering emails into “spam” or “non-spam.” Using classification algorithms, text analysis software can perform tasks like aspect-based sentiment analysis to categorize unstructured text by topic and polarity of opinion.

The study of classification in statistics is vast, and there are several types of classification algorithms depending on the dataset. Following are the most common algorithms in machine learning:

- Logistic Regression
- Decision Tree
- Support Vector Machines
- Naive Bayes

CHAPTER 2

LOGISTIC REGRESSION

Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not. It is a kind of statistical algorithm, which analyse the relationship between a set of independent variables and the dependent binary variables. It is a powerful tool for decision-making. For example, email spam or not.

In this chapter we introduce an algorithm that is admirably suited for discovering the link between features or cues and some particular outcome: logistic regression. Indeed, logistic regression is one of the most important analytic tools in the social and natural sciences. In natural language processing, logistic regression is the base- line supervised machine learning algorithm for classification, and also has a very close relationship with neural networks.

2.1 MATHEMATICAL FRAMEWORK

BASIC DEFINITIONS

Independent variables: The input characteristics or predictor factors applied to the dependent variable's predictions.

Dependent variable: The target variable in a logistic regression model, which we are trying to predict.

Logistic function: The formula used to represent how the independent and dependent variables relate to one another. The logistic function transforms the input variables into a probability value between 0 and 1, which represents the likelihood of the dependent variable being 1 or 0.

Odds: It is the ratio of something occurring to something not occurring. It is different from probability as the probability is the ratio of something occurring to everything that could possibly occur.

Log-odds: The log-odds, also known as the logit function, is the natural logarithm of the odds. In logistic regression, the log odds of the dependent variable are modelled as a linear combination of the independent variables and the intercept.

Coefficient: The logistic regression model's estimated parameters, show how the independent and dependent variables relate to one another.

Intercept: A constant term in the logistic regression model, which represents the log odds when all independent variables are equal to zero.

LOGISTIC FUNCTION (SIGMOID FUNCTION):

The sigmoid function is a mathematical function used to map the predicted values to probabilities. It maps any real value into another value within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function. In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

ALGORITHM OF LOGISTIC REGRESSION

Logistic regression is a supervised machine learning algorithm mainly used for binary classification where we use a logistic function, also known as a sigmoid function that takes input as independent variables and produces a probability value between 0 and 1. For example, we have two classes Class 0 and Class 1 if the value of the logistic function for an input is greater than 0.5 (threshold value) then it belongs to Class 1 it belongs to Class 0. It's referred to as regression because it is the extension of linear regression but is mainly used for classification problems. The difference between linear regression and logistic regression is that linear regression output is the

continuous value that can be anything while logistic regression predicts the probability that an instance belongs to a given class or not.

Let the independent input features be,

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

The dependent variable is Y having only binary value i.e. 0 or 1.

$$Y = \begin{cases} 0 & \text{if class 1} \\ 1 & \text{if class 2} \end{cases}$$

then apply the multi-linear function to the input variables X

$$Y = (\sum_{i=1}^n \omega_i x_i) + b$$

Here x_i is the observation of X,

$\omega_i = [\omega_1, \omega_2, \dots, \omega_m]$ is the weights or Coefficient, and b is the bias term also known as intercept. Simply this can be represented as the dot product of weight and bias.

$$Z = \omega \cdot X + b$$

Now we use the sigmoid function where the input will be z and we find the probability between 0 and 1.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

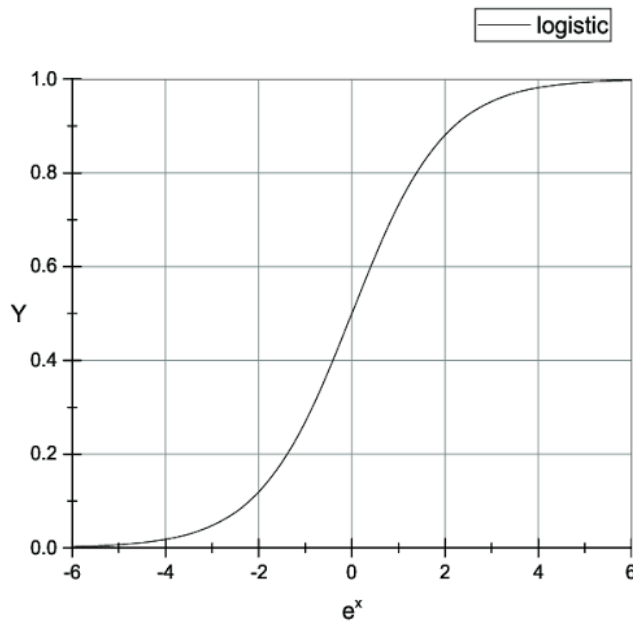


Figure 2.1

As shown above, the figure sigmoid function converts the continuous variable data into the probability i.e. between 0 and 1.

- $\sigma(\mathbf{z})$ Tends towards 1 as

$$Z \rightarrow \infty$$

- $\sigma(\mathbf{z})$ Tends towards 0 as

$$Z \rightarrow -\infty$$

- $\sigma(\mathbf{z})$ is always bounded between 0 and 1

Where the probability of being a class can be measured as :

$$P(y = 1) = \sigma(\mathbf{z})$$

$$P(y = 0) = 1 - \sigma(\mathbf{z})$$

LOGISTIC REGRESSION EQUATION

The odd is the ratio of something occurring to something not occurring. it is different from probability as the probability is the ratio of something occurring to everything that could possibly occur. So odd will be;

$$\frac{P(x)}{1-P(x)} = e^z$$

Applying natural log on odd. then log odd will be;

$$\log \left[\frac{P(x)}{1-P(x)} \right] = z$$

$$\log \left[\frac{P(x)}{1-P(x)} \right] = \omega \cdot X + b$$

then the final Logistic regression equation will be:

$$p(X; \mathbf{b}, \boldsymbol{\omega}) = \frac{e^{\omega \cdot X + b}}{1 + e^{\omega \cdot X + b}} = \frac{1}{1 + e^{-\omega \cdot X + b}}$$

LIKELIHOOD FUNCTION FOR LOGISTIC REGRESSION

The likelihood function is the probability that the observed value of the dependent variable may be predicted from the observed values of the independent variables.

The predicted probabilities will $p(X; \mathbf{b}, \boldsymbol{\omega}) = p(x)$ for $y=1$ and for $y = 0$ predicted probabilities will $1 - p(X; \mathbf{b}, \boldsymbol{\omega}) = 1 - p(x)$.

$$L(b, \omega) = \prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

Taking natural logs on both sides

$$l(b, \omega) = \log(L(b, \omega))$$

$$= \sum_{i=1}^n y_i \log p(x_i) + (1-y_i) \log (1 - p(x_i))$$

$$= \sum_{i=1}^n y_i \log p(x_i) + \log(1 - p(x_i)) \sum_{i=1}^n (1-y_i)$$

$$= \sum_{i=1}^n \log(1 - p(x_i)) + \sum_{i=1}^n y_i \log \frac{p(x_i)}{1-p(x_i)}$$

$$= \sum_{i=1}^n -\log 1 - e^{-(\omega_i x_i + b)} + \sum_{i=1}^n y_i (\omega_i x_i + b)$$

$$= -\log 1 + e^{\omega_i x_i + b} + \sum_{i=1}^n y_i (\omega_i x_i + b)$$

GRADIENT OF THE LOG – LIKELIHOOD FUNCTION

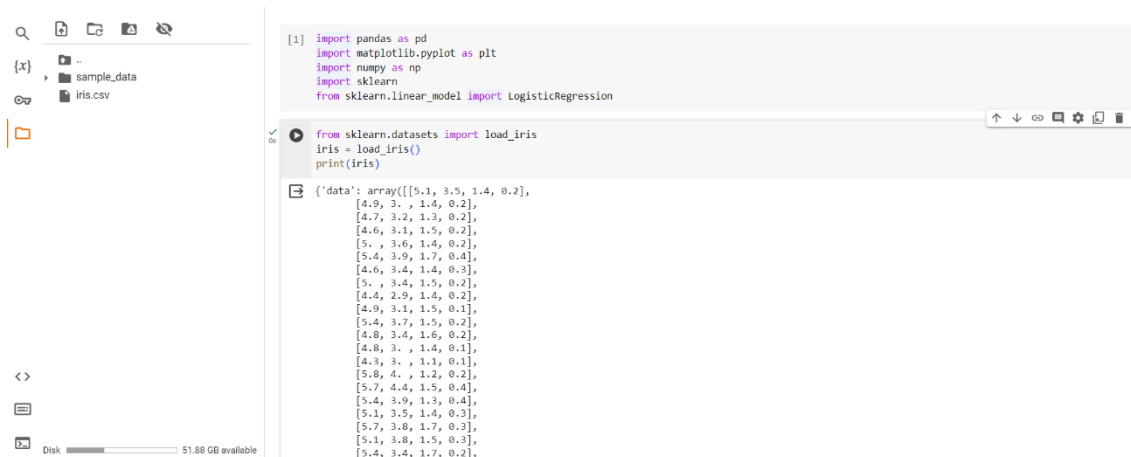
To find the maximum likelihood estimates, we differentiate with respect to w ,

$$\begin{aligned} \frac{\partial J(l(b, \omega))}{\partial \omega_j} &= - \sum_{i=1}^n \frac{1}{1 + e^{\omega_i x_i + b}} e^{(\omega_i x_i + b)} x_{ij} + \sum_{i=1}^n y_i x_{ij} \\ &= - \sum_{i=1}^n p(x_i; b, \omega) x_{ij} + \sum_{i=1}^n y_i x_{ij} \\ &= \sum_{i=1}^n (y_i - p(x_i; b, \omega)) x_{ij} \end{aligned}$$

2.2 IMPLEMENTATION OF LOGISTIC REGRESSION

IRIS DATASET

The Iris flower data set or Fisher's Iris data set is a multivariate data set used and made famous by the British statistician and biologist Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres. The dataset contains a set of 150 records under five attributes: sepal length, sepal width, petal length, petal width and species.



```
[1] import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import sklearn
from sklearn.linear_model import LogisticRegression

from sklearn.datasets import load_iris
iris = load_iris()
print(iris)
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
               [4.9, 3. , 1.4, 0.2],
               [4.7, 3.2, 1.3, 0.2],
               [4.6, 3.1, 1.5, 0.2],
               [5. , 3.6, 1.4, 0.2],
               [5.4, 3.9, 1.7, 0.4],
               [4.6, 3.4, 1.4, 0.3],
               [5. , 3.4, 1.5, 0.2],
               [4.4, 2.9, 1.4, 0.2],
               [4.9, 3.1, 1.5, 0.1],
               [5.4, 3.7, 1.5, 0.2],
               [4.8, 3.4, 1.6, 0.2],
               [4.8, 3. , 1.4, 0.1],
               [4.3, 3. , 1.1, 0.1],
               [5.8, 4. , 1.2, 0.2],
               [5.7, 4.4, 1.5, 0.4],
               [5.4, 3.9, 1.3, 0.4],
               [5.1, 3.5, 1.4, 0.3],
               [5.7, 3.8, 1.7, 0.3],
               [5.1, 3.8, 1.5, 0.3],
               [5.4, 3.4, 1.7, 0.2],
```

Here the first code imports various libraries including Pandas, Matplotlib, Numpy, Sklearn, and Logistic Regression. These libraries are used for the data analysis, visualization, Machine Learning and statistical modeling. The second code imports sklearn.datasets module and then loads the iris dataset.



```
[3] iris.keys()
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])

[4] print(iris.target_names)
['setosa' 'versicolor' 'virginica']

[5] print(iris.feature_names)
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

[6] df=pd.DataFrame(iris.data)
print(df.head())
```

```
   0  1  2  3
0  5.1 3.5 1.4 0.2
1  4.9 3.0 1.4 0.2
2  4.7 3.2 1.3 0.2
3  4.6 3.1 1.5 0.2
4  5.0 3.6 1.4 0.2
```

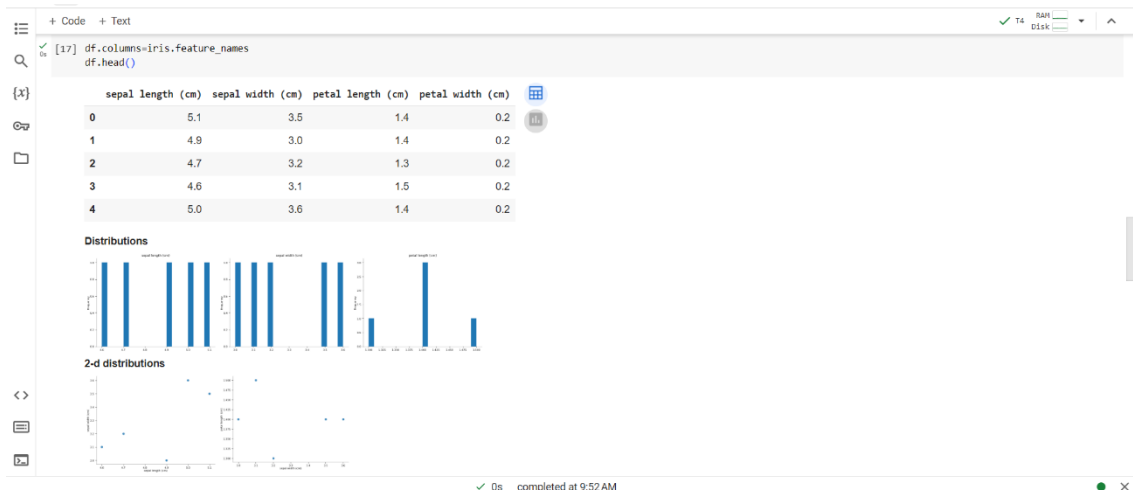
```
[17] df.columns=iris.feature_names
df.head()
```

```
sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
```

- iris.keys() imports the keys of iris data set.
- print(iris.target_names) prints the target names of the iris dataset which are 'setosa','versicolor','virginica'.
- print(iris.feature_names). This code is used to print the feature names of the Iris dataset. The names of the features are "sepal length (cm)", "sepal width (cm)", "petal length (cm)", and "petal width (cm)".
- df=pd.DataFrame(iris.data) print(df.head())

This code creates a Pandas DataFrame from the iris.data variable. The iris.data variable is a NumPy array containing the sepal length, sepal width, petal length, and petal width measurements for 150 iris flowers.

The df.head() method prints the first five rows of the DataFrame.



The code `df.columns=iris.feature_names` `df.head()` is used to print the column names and the first five rows of a DataFrame named `df`.



- The code `df.info()` is a function in the pandas library that prints information about a DataFrame. The function prints the number of rows and columns in the DataFrame, the data types of each column, memory usage of the DataFrame.

- The code

`X=iris.data`

`Y=iris.target` `print(X.shape)`

`print(Y.shape)`

prints the shape of the data and target arrays. The data array has a shape of (150, 4), which means that there are 150 samples, each with 4 features. The target array has a shape of (150,), which means that there are 150 target values, one for each sample.

- The code

`X_train,X_test,Y_train,Y_test=sklearn.model_selection.train_test_split(X,`

`Y_test_size=0.25,random_state=2)` is a Python code for splitting data into train and test sets. The code splits the data into two sets, `X_train` and `X_test`. The `X_train` set will be used to train the model, and the `X_test` set will be used to test the model. The `train_test_split()` function is a useful tool for splitting data into train and test sets.

- The code `log=LogisticRegression(random_state=0)`
`log.fit(x_train,Y_train)` is for a Python program that implements a logistic regression classifier. Logistic regression is a statistical model that is used to predict the probability of an event occurring. In this case, the program is being used to predict whether or not a flower is a setosa, versicolor, or virginica.



```
[40] Y_pred=log.predict(X_test)
print(Y_pred)

[0 0 2 0 0 2 0 2 2 0 0 0 0 0 1 1 0 1 2 1 2 1 2 1 1 0 0 2 0 2 2 0 1 2 1 0 2
 1]

[41] from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,Y_pred)
cm

array([[16,  0,  0],
       [ 0, 10,  1],
       [ 0,  0, 11]])

[43] from sklearn.metrics import accuracy_score
print(accuracy_score(Y_test,Y_pred))

0.9736842105263158
```

- The code `Y_pred=log.predict(X_test)`
`print (Y_pred)` is a code snippet from a Python program. It uses the `LogisticRegression` class from `scikit-learn` to predict the species of a flower based on its sepal length, sepal width, petal length, and petal width. The code first splits the data into training and test sets. Then, it trains a `LogisticRegression` model on the training set. Finally, it evaluates the model on the test set and prints the accuracy.

- The code `from sklearn.metrics import confusion_matrix`
`cm=confusion_matrix(Y_test ,Y_pred)` then creates a confusion matrix variable called `cm` and assigns it the value of the `confusion_matrix` function call. The `confusion_matrix` function call takes the `y_test` and `y_pred` variables as arguments. The `y_test` variable is the actual labels of the test data and the `y_pred` variable is the predicted labels of the test data.

- The code `from sklearn.metrics import accuracy_score`
`print(accuracy_score(Y_test,Y_pred))` is a Python code snippet that uses the `accuracy_score` function from the `sklearn.metrics` module to calculate the accuracy of a classification model.

Here we got an 97% accuracy which shows that the model built is very accurate.

2.3 ADVANTAGES AND DISADVANTAGES

Advantages

Logistic Regression performs well when the dataset is linearly separable. It is less prone to over-fitting but it can overfit in high dimensional datasets. It not only gives a measure of how relevant a predictor (coefficient size) is, but also its direction of association (positive or negative). It is easier to implement, interpret and very efficient to train.

Disadvantages

Main limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables. In the real world, the data is rarely linearly separable. Most of the time data would be a jumbled mess. If the number of observations are less than the number of features, Logistic Regression should not be used, otherwise it may lead to overfit. It can only be used to predict discrete functions. Therefore, the dependent variable of Logistic Regression is restricted to the discrete number set. This restriction itself is problematic, as it is prohibitive to the prediction of continuous data.

CHAPTER 3

DECISION TREE

Decision tree learning is one of the most widely used and practical methods for inductive inference. It is a method for approximating discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions. This chapter describes a family of decision tree learning algorithms that includes widely used algorithms such as ID3, CHI – SQUARE and REDUCTION IN VARIANCE. These decision tree learning methods search a completely expressive hypothesis space and thus avoid the difficulties of restricted hypothesis spaces. Their inductive bias is a preference for small trees over large trees.

Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. Learned trees can also be re-represented as sets of if-then rules to improve human readability. These learning methods are among the most popular of inductive inference algorithms and have been successfully applied to a broad range of tasks from learning to diagnose medical cases to learning to assess credit risk of loan applicants.

3.1 MATHEMATICAL FRAMEWORK

Decision tree is a hierarchical data structure that represents data through a divide and conquer strategy. In this class we discuss decision trees with categorical labels, but non-parametric classification and regression can be performed with decision trees as well.

BASIC DEFINITIONS

Root Node: A decision tree's root node, which represents the original choice or feature from which the tree branches, is the highest node.

Internal Nodes (Decision Nodes): Nodes in the tree whose choices are determined by the values of particular attributes. There are branches on these nodes that go to other nodes.

Leaf Nodes (Terminal Nodes): The branches' termini, when choices or forecasts are decided upon. There are no more branches on leaf nodes.

Branches (Edges): Links between nodes that show how decisions are made in response to particular circumstances.

Splitting: The process of dividing a node into two or more sub-nodes based on a decision criterion. It involves selecting a feature and a threshold to create subsets of data.

Parent Node: A node that is split into child nodes. The original node from which a split originates.

Child Node: Nodes created as a result of a split from a parent node.

Decision Criterion: The rule or condition used to determine how the data should be split at a decision node. It involves comparing feature values against a threshold.

Pruning: The process of removing branches or nodes from a decision tree to improve its generalization and prevent overfitting.

Entropy: A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogeneous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. if Entropy $E(s) = 0$ means it is Completely homogenic.

Consider a dataset with 'c' classes. The entropy may be calculated using the formula below:

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

p_i is the probability of randomly selecting an example in class i . The logarithm of fractions gives a negative value, and hence a '-' sign is used in the entropy formula to negate these negative values.

Some useful relation between probability and entropy (consider binary classification):

1. Probability (Both Class) = 0.5 & Entropy = 1
2. Probability (Either or Both Class) = 0 & Entropy = 0 it is called Leaf Node & stop Split.

Information gain : The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

$$\begin{aligned} \text{Gain}(S, A) &\equiv \text{Entropy}(S) - \sum_{v \in D_A} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \\ &\equiv \text{Entropy}(S) - \text{Weighted entropy gain} \end{aligned}$$

Weighted entropy gain:

Weighted entropy gain is a measure used in decision tree algorithms to determine the best split for a given node in the tree. It calculates the reduction in entropy (or increase in information purity) that would result from splitting the data based on a particular attribute. The "weighted" part refers to taking into account the distribution of data samples across different classes when calculating the entropy gain. It's essentially the weighted average of the entropy gains for each possible split, where the weight is the proportion of data samples in each class.

$$\text{Weighted entropy gain} = \sum_{v \in D_A} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

Above is weighted average for all node

1. $|S_v|$ = sum of every node values

2. $|S|$ = sum of set S node values
3. Entropy (S_v) = entropy of current node

TYPES OF DECISION TREE ALGORITHMS

There are many types of decision tree algorithms used today for all sorts of tasks, but we've chosen three of the more popular versions and illustrated their general abilities. Algorithms are ID3, Chi-Square and Reduction in Variance. All types can be used to solve problems or answer questions from the simple to the most intricate and detailed

1. ID3

Iterative Dichotomise 3, or ID3, developed by Ross Quinlan in 1983, is a nominal decision tree used to split two or more features into groups at every step. It uses a top-down greedy approach, which means you start from the top and go down, while

"greedy" means you pick the best option at that moment and move to the next step.

STEPS OF ID3 ALGORITHMS

1. Select Root node(S) based on low Entropy and Highest Information Gain
2. On each iteration of an algorithms, it calculates the Entropy and Information gain, considering that every node is unused
3. Select node base on Lowest Entropy or Highest I.G
4. Then splits set S to produce the subsets of data
5. An algorithm continuous to recur on each subset and make sure that attributes are fresh and creates the decision, Tree.

2. CHI - SQUARE

Chi -Square is also known as CHAID (Chi-Squared automatic interaction detection) is a highly visual and easy-to-understand decision tree that uses input variables to help decide the best possible result. These can often be used in direct marketing situations where having an idea of how participants are likely to best respond would be helpful. Chi-square also allows for much more precision and accuracy because nodes can be split multiple times and allow for more data to be processed.

3. REDUCTION IN VARIANCE

This algorithm splits the chosen nodes in a continuously changing target variable. It's named after the fact that it uses variance to measure and decide how nodes are split into child nodes or sub-nodes. Let's learn about most widely used ID3 algorithm more.

Among these ID3 algorithm is most commonly used. Let's learn more about ID3 algorithm and steps involved.

3.2 IMPLEMENTATION OF DECISION TREE

BREAST CANCER DATASET

Breast cancer dataset is taken to implement decision tree algorithm. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

		diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	fractal_dimension_se	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
842302 M		17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	1.095	0.9053	8.589	153.4	0.006399	0.04904	0.05373	0.01587	0.03003	0.006193	25.38	17.33	184.6	2019	0.1622	0.6656	0.7119				
842517 M		20.57	17.77	132.9	1236	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.0186	0.0134	0.01389	0.003532	24.99	23.41	158.8	1956	0.1238	0.1866	0.2416				
8430903 M		19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	0.7456	0.7869	4.585	94.03	0.00615	0.04006	0.03832	0.02058	0.0225	0.004571	23.57	25.53	152.5	1709	0.1444	0.4245	0.4504				
84348301 M		11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	0.4956	1.156	3.445	27.23	0.00911	0.07458	0.05661	0.01867	0.05963	0.009208	14.91	26.5	98.87	567.7	0.2098	0.8663	0.6869				
84358402 M		20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883	0.7572	0.7813	5.438	94.44	0.01149	0.02461	0.05688	0.01885	0.01756	0.005115	22.54	16.67	152.2	1575	0.1374	0.205	0.4				
843786 M		12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613	0.3345	0.8902	2.217	27.19	0.00751	0.03345	0.03672	0.01137	0.02165	0.005082	15.47	23.75	103.4	741.6	0.1791	0.5249	0.5355				
844399 M		18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742	0.4467	0.7732	3.18	53.91	0.004314	0.01382	0.02254	0.01039	0.01369	0.002179	22.88	27.66	153.2	1606	0.1442	0.2576	0.3784				
84458202 M		13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451	0.5835	1.377	3.856	50.96	0.008805	0.03029	0.02488	0.01448	0.01486	0.005412	17.06	28.14	110.6	897	0.1654	0.3682	0.2678				
844981 M		13	21.82	87.5	513.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389	0.3063	1.002	2.406	24.32	0.005731	0.03502	0.03553	0.01226	0.02143	0.003749	15.49	30.73	106.2	793.3	0.1703	0.5401	0.539				
84501001 M		12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243	0.2976	1.599	2.039	23.94	0.007149	0.07217	0.07743	0.01432	0.01789	0.01008	15.09	40.68	97.65	711.4	0.1853	1.058	1.105				
845636 M		16.02	23.24	102.7	757.8	0.08206	0.06669	0.03299	0.03233	0.1528	0.05697	0.3795	1.187	2.466	40.51	0.004029	0.009269	0.01101	0.007591	0.0146	0.002085	19.19	33.88	123.8	1150	0.1181	0.1551	0.1459				
84610002 M		15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082	0.5058	0.9849	3.564	54.16	0.005771	0.04061	0.02791	0.01282	0.02008	0.004144	20.42	27.28	136.5	1299	0.1396	0.5609	0.3965				
846226 M		19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	0.2397	0.078	0.9555	3.568	11.07	116.2	0.003139	0.08297	0.0889	0.0409	0.04484	0.01284	20.96	29.94	151.7	1332	0.1037	0.3903	0.3639				
846381 M		15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	0.1847	0.05338	0.4033	1.078	2.903	36.58	0.009769	0.03126	0.05051	0.01992	0.02981	0.003002	16.84	27.66	112	876.5	0.1131	0.1924	0.2322				
84667401 M		13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	0.2069	0.07682	0.2121	1.169	2.061	19.21	0.006429	0.05936	0.05501	0.01628	0.01961	0.008093	15.03	32.01	108.8	697.7	0.1651	0.7725	0.6943				
84799002 M		14.54	27.54	96.73	638.8	0.1139	0.1595	0.1639	0.07364	0.2303	0.07077	0.37	1.033	2.879	32.55	0.005607	0.0424	0.04741	0.0109	0.01857	0.005466	17.46	37.13	124.1	943.2	0.1678	0.6577	0.7026				
848406 M		14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05239	0.1586	0.05922	0.4727	1.24	3.195	45.4	0.005718	0.01162	0.01998	0.01099	0.0141	0.002085	19.07	30.88	123.4	1138	0.1464	0.1871	0.2914				
84862001 M		16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	0.2164	0.07356	0.5692	1.073	3.854	54.18	0.007026	0.02501	0.03188	0.01297	0.01689	0.004142	20.96	31.48	136.8	1315	0.1789	0.4233	0.4784				
849014 M		19.81	22.15	130	1260	0.09831	0.1027	0.1479	0.09498	0.1582	0.05395	0.7582	1.017	5.865	112.4	0.006494	0.01893	0.03391	0.01521	0.01356	0.001997	27.32	30.88	186.8	2398	0.1512	0.315	0.5372				
8510426 B		13.54	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781	0.1885	0.05766	0.2699	0.7886	2.058	23.56	0.008462	0.0146	0.02387	0.01315	0.0198	0.0023	15.11	19.26	99.7	711.2	0.144	0.1779	0.239				
8510653 B		13.08	15.71	85.63	520	0.1075	0.127	0.04568	0.0311	0.1967	0.06811	0.1852	0.7477	1.383	14.67	0.004097	0.01898	0.01698	0.00649	0.01678	0.002425	14.5	20.49	96.09	630.5	0.1312	0.2776	0.189				
8510824 B		9.504	12.44	60.34	273.9	0.1024	0.06492	0.02956	0.02076	0.1815	0.06905	0.2773	0.9788	1.909	15.7	0.009606	0.01432	0.01985	0.01421	0.02027	0.002968	10.23	15.66	65.13	314.9	0.1324	0.1148	0.08867				
8511133 M		15.34	14.26	102.5	704.4	0.1073	0.2135	0.2077	0.09756	0.2521	0.07332	0.4388	0.7696	3.384	44.91	0.006789	0.05328	0.06446	0.02252	0.03672	0.004394	18.07	19.08	125.1	980.9	0.139	0.3954	0.6305				
851509 M		21.16	23.04	137.2	1404	0.09438	0.1022	0.1097	0.08632	0.1769	0.05278	0.6917	1.127	4.303	93.99	0.004728	0.01259	0.01715	0.01038	0.01083	0.001987	29.17	35.59	188	7815	0.1401	0.26	0.3155				
852552 M		16.65	21.38	110	904.6	0.1121	0.1457	0.1525	0.0917	0.1995	0.0633	0.8068	0.9017	5.455	102.6	0.006468	0.01882	0.02741	0.0113	0.01488	0.002801	26.46	31.56	177	2215	0.1805	0.3578	0.4695				
852631 M		17.14	16.4	116	912.7	0.1186	0.2276	0.2229	0.1401	0.304	0.07413	1.046	0.976	7.276	111.4	0.008029	0.03799	0.03732	0.02397	0.02308	0.007444	22.25	21.4	152.4	1461	0.1545	0.3949	0.3853				
852763 M		14.58	21.53	97.41	644.8	0.1054	0.1868	0.1425	0.08783	0.2252	0.06924	0.2545	0.9832	2.11	21.05	0.004452	0.03055	0.02681	0.01352	0.01454	0.003711	17.62	33.21	122.4	896.9	0.1525	0.6643	0.5539				
852781 M		18.61	20.25	122.1	1094	0.0944	0.1066	0.149	0.07731	0.1697	0.05699	0.8529	1.849	5.632	93.54	0.01075	0.02722	0.05081	0.01911	0.02293	0.004217	21.31	27.26	139.9	1403	0.1338	0.2117	0.3446				
852973 M		15.3	25.27	102.4	732.4	0.1082	0.1697	0.1683	0.08751	0.1926	0.0654	0.439	1.012	3.498	43.5	0.005233	0.03057	0.01083	0.01768	0.002967	20.27	36.71	149.3	1269	0.1641	0.611	0.611	0.611	0.611	0.611	0.611	
853201 M		17.57	15.05	115	955.1	0.09847	0.1157	0.09875	0.07953	0.1739	0.06149	0.6003	0.8225	4.655	61.1	0.005627	0.03033	0.04037	0.01354	0.01925	0.003742	20.01	19.52	134.9	1227	0.1255	0.2812	0.2489				
853401 M		18.63	25.11	124.8	1088	0.1064	0.1887	0.2319	0.1244	0.2183	0.06197	0.8307	1.466	5.574	105	0.006248	0.03374	0.05196	0.01158	0.02007	0.00456	23.15	34.01	160.5	1670	0.1491	0.4257	0.6133				
853612 M		11.84	18.7	77.93	440.6	0.1109	0.1516	0.1218	0.05182	0.2301	0.07799	0.4825	1.03	3.475	41	0.005531	0.03414	0.04205	0.01044	0.02273	0.005667	16.82	28.12	119.4	888.7	0.1637	0.5775	0.6956				

```

[2] Import numpy
     Import matplotlib.pyplot as plt
     Import pandas as pd
     Import seaborn as sns

[3] df=pd.read_csv("data.csv")

[4] df.head()

      id  diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  compactness_mean  concavity_mean  concave points_mean  ...
0    842302      M         17.99         10.38          122.80        1001.0         0.11840         0.27760         0.3001         0.14710  ...
1    842517      M         20.57         17.77          132.90        1326.0         0.08474         0.07864         0.0869         0.07017  ...
2    8430903      M         19.69         21.25          130.00        1203.0         0.10960         0.15990         0.1974         0.12790  ...
3    84348301      M         11.42         20.38           77.58         386.1         0.14250         0.28390         0.2414         0.10520  ...
4    84358402      M         20.29         14.34          135.10        1297.0         0.10030         0.13280         0.1980         0.10430  ...

5 rows x 33 columns

[5] df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 569 entries, 0 to 568
dtypes: float64(1), int64(1), object(1)
memory usage: 140.4+ KB

completed at 6:50 PM

```

```

[5] df.describe()

      id  diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  compactness_mean  concavity_mean  concave points_mean  ...
count  569         0         0.000000         0.000000         0.000000         0.000000         0.000000         0.000000         0.000000  ...
mean    0         0         15.330000         12.910000         118.420000        984.400000         0.127000         0.147000         0.175000         0.147000  ...
std     0         0         4.849000         4.349000         38.860000        584.900000         0.047000         0.047000         0.047000         0.047000  ...
min     0         0          9.504000          9.504000         60.340000        273.900000         0.029560         0.020760         0.020760         0.020760  ...
max     0         0         21.160000         23.040000        137.200000       1404.000000         0.109700         0.109700         0.109700         0.109700  ...

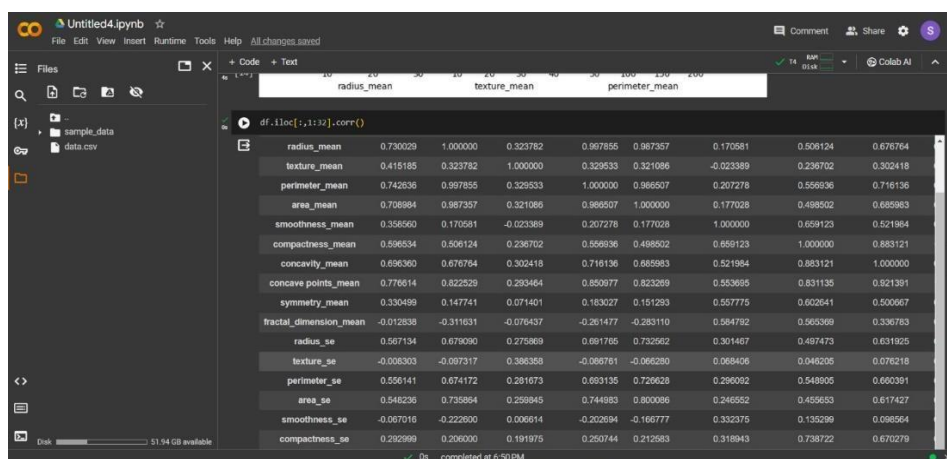
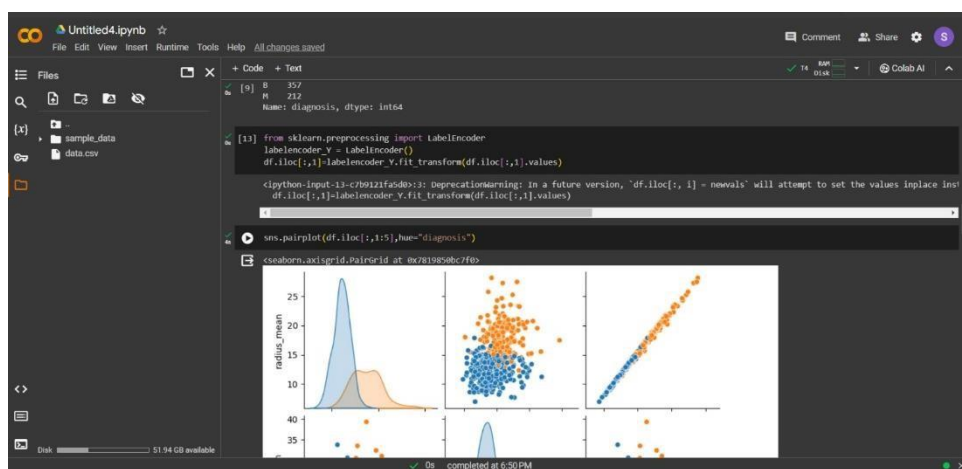
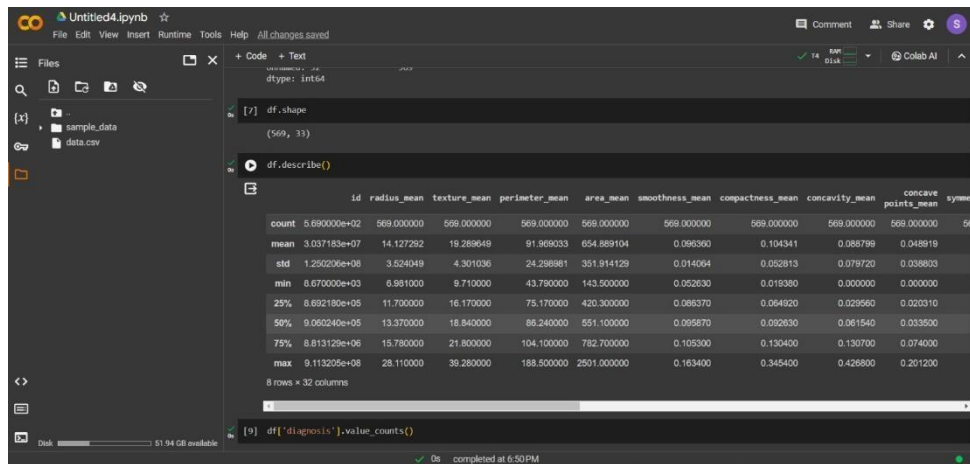
dtypes: float64(1), int64(1), object(1)
memory usage: 140.4+ KB

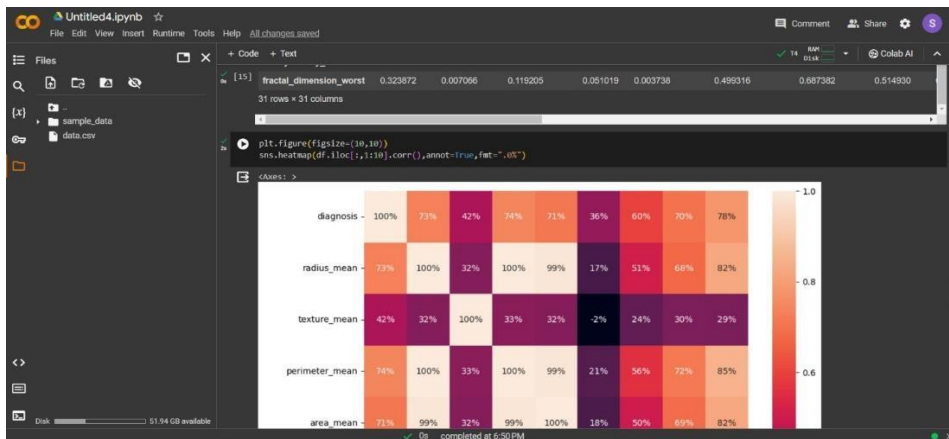
[6] df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 569 entries, 0 to 568
dtypes: float64(1), int64(1), object(1)
memory usage: 140.4+ KB

completed at 6:50 PM

```





```
[15] fractal_dimension_worst 0.323872 0.007066 0.119205 0.051019 0.003738 0.499316 0.687382 0.514930
31 rows x 31 columns

[16] plt.figure(figsize=(10,10))
sns.heatmap(df.iloc[:,1:10].corr(),annot=True,fat='0.05')

[17] X=df.iloc[:,2:31].values
Y=df.iloc[:,1].values

[18] from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20,random_state=0)

[19] from sklearn.preprocessing import StandardScaler
X_train=StandardScaler().fit_transform(X_train)
X_test=StandardScaler().fit_transform(X_test)

[22] from sklearn.tree import DecisionTreeClassifier

[23] model = DecisionTreeClassifier()

[25] model.fit(X_train, Y_train)

[26] Y_pred=model.predict(X_test)

[27] from sklearn import metrics
```

```
564 926424 1 21.56 22.39 142.00 1479.0 0.11100 0.11590 0.10340 0.14400 0.09791
565 926682 1 20.13 28.26 131.20 1261.0 0.09780 0.10340 0.14400 0.09791
566 926954 1 16.60 28.08 108.30 858.1 0.08455 0.10230 0.09251 0.05302
567 927241 1 20.60 29.33 140.10 1265.0 0.11780 0.27700 0.35140 0.15200
568 92751 0 7.76 24.54 47.92 181.0 0.05283 0.04362 0.00000 0.00000
569 rows x 33 columns

[28] print('Predicted values:')
print(Y_pred)
print('Actual values:')
print(Y_test)

Predicted values:
[1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0
 1 0 1 0 0 1 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 0
 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0
 1 1 0]
Actual values:
[1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0
 1 0 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1
 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0
 1 1 0]
```

3.3 ADVANTAGES AND DISADVANTAGES

Advantages

Decision trees are self-explanatory and when compacted they are also easy to follow. In other words, if the decision tree has a reasonable number of leaves, it can be grasped by non-professional users. Furthermore, decision trees can be converted to a set of rules. Thus, this representation is considered as comprehensible. They can handle both nominal and numeric input attributes. Decision tree representation is rich enough to represent any discrete value classifier. They are capable of handling datasets that may have errors and missing values. Decision trees are considered to be a nonparametric method. This means that decision trees have no assumptions about the space distribution and the classifier structure.

Disadvantages

Most of the algorithms (like ID3 and C4.5) require that the target attribute will have only discrete values. As decision trees use the "divide and conquer" method, they tend to perform well if a few highly relevant attributes exist, but less so if many complex interactions are present. One of the reasons for this is that other classifiers can compactly describe a classifier that would be very challenging to represent using a decision tree. A simple illustration of this phenomenon is the replication problem of decision trees (Pagallo and Huassler, 1990). Since most decision trees divide the instance space into mutually exclusive regions to represent a concept, in some cases the tree should contain several duplications of the same sub-tree in order to represent the classifier. For instance, if the concept follows the following binary function: $y = (A_1 \wedge A_2) \vee (A_3 \wedge A_4)$ then the minimal univariate decision tree that represents this function is illustrated in Figure 9.3. Note that the tree contains two copies of the same subtree. The greedy characteristic of decision trees leads to another disadvantage that should be pointed out. This is its over-sensitivity to the training set, to irrelevant attributes and to noise (Quinlan, 1993).

CHAPTER 4

SUPPORT VECTOR MACHINE (SVM)

Support Vector Machines (SVM's) are a relatively new learning method used for binary classification. It was introduced by Vladimir Vapnik and colleagues. The earliest mention was in (Vapnik, 1979), but the first main paper seems to be (Vapnik, 1995). The basic idea of SVM is to find a hyperplane which separates the d-dimensional data perfectly into its two classes. However, since data is often not linearly separable, SVM's introduce the notion of a 'kernel an induced feature space' which casts the data into a higher dimensional space where the data is separable. Typically, casting into such a space would cause problems computationally, and with overfitting. The key insight used in SVM's is that the higher-dimensional space doesn't need to be dealt with directly (as it turns out, only the formula for the dot-product in that space is needed), which eliminates the above concerns. Furthermore, the VC-dimension (Vapnik-Chervonenkis dimension, a measure of a system's likelihood to perform well on unseen data) of SVM's can be explicitly calculated, unlike other learning methods like neural networks, for which there is no measure. Overall, SVM's are intuitive, theoretically well- founded, and have shown to be practically successful. SVM's have also been extended to solve regression tasks (where the system is trained to output a numerical value, rather than "yes/no" classification). In this chapter we are going see the mathematical formulation of support vector machine, the implementation of dataset in SVM and some importance and limitations.

4.1 MATHEMATICAL FRAMEWORK

BASIC DEFINITIONS

Hyperplane: Hyper plane is the decision boundary that is used to separate the data points of different classes in a feature space. In the case of a linear equation, it will be a linear equation i.e., $w\mathbf{x} + b = 0$

Support Vectors: Support vectors are the closest data points to the hyperplane, which plays an important role in deciding the hyperplane and the margin.

Margin: Margin is the distance between the support vector and the hyperplane. The main objective of the support vector machine algorithm is to maximize the margin. The wider the margin indicates better classification performance.

Kernel: The SVM kernel is a function that takes low-dimensional input space and transforms it into higher-dimensional space, that is it converts non separable problems to separable problems. It is mostly useful in non-linear separation problems. Simply put the kernel, does some extremely complex data transformation and then finds out the process to separate the data based on the labels or output defined.

Linear: $K(\mathbf{w}, \mathbf{b}) = \mathbf{w}^T \mathbf{x} + \mathbf{b}$

Polynomial: $K(\mathbf{w}, \mathbf{x}) = (\gamma \mathbf{w}^T \mathbf{x} + \mathbf{b})^N$

Gaussian RBF: $K(\mathbf{w}, \mathbf{x}) = \exp(-\gamma ||x_i - x_j||^n)$

Sigmoid: $K(x_i, x_j) = \tanh(\alpha x_i^T x_j + \mathbf{b})$

There are other such kernels too.

Hard Margin: The maximum-margin hyperplane or the hard margin hyperplane is a hyperplane that properly separates the data points of different categories without any misclassifications.

Soft Margin: When the data is not perfectly separable or contains outliers, SVM permits a soft margin technique. Each data point has a slack variable introduced by the soft margin SVM formulation, which softens the strict

margin requirement and permits certain misclassification or violations. It discovers a compromise between increasing the margin and reducing violations.

C: Margin maximization and misclassification fines are balanced by the regularisation parameter C in SVM. The penalty for going over the margin or misclassifying data items are decided by it. A stricter penalty is imposed with a greater value of C , which results in a smaller margin and perhaps fewer misclassifications.

Hinge Loss: A typical loss function in SVM is hinge loss. It punishes incorrect classifications or margin violations. The objective function in SVM is frequently formed by combining it with the regularisation term.

Dual Problem: A dual problem of the optimisation problem that requires the Lagrange multipliers related to the support vectors can be used to solve SVM. The dual problem enables the use of kernel tricks and more effective computing.

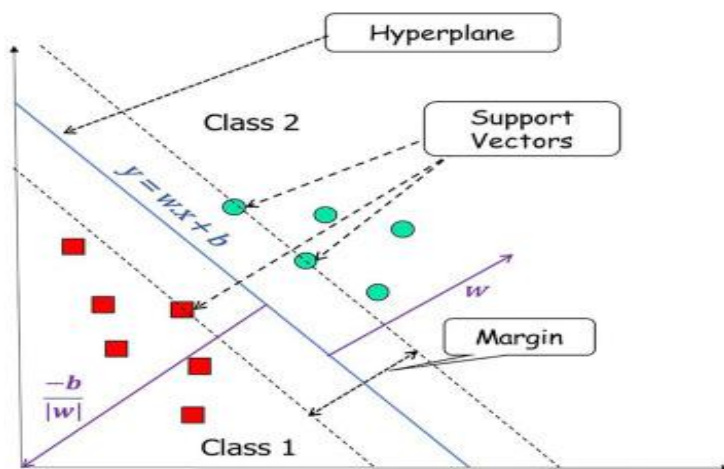


Figure 4.1

The figure 4.1 shows the graph of linear SVM in accordance with some of the basic terminologies that we went through.

Types of Support Vector Machine

Based on the nature of the decision boundary, SVM can be classified into two main parts:

1. Linear Support Vector Machine

Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated, linear SVMs are very suitable. This means that a single straight line (2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.

Linear Separable Binary Classification (Hard Margin)

We have L training points, where each input \mathbf{x}_i has D attributes (i.e. is of dimensionality D) and is in one of two classes $\mathbf{y}_i = -1$ or $+1$, i.e. our training data is of the form:

$$\{\mathbf{x}_i, \mathbf{y}_i\} \text{ where } i = 1 \dots L, \mathbf{y}_i \in \{-1, 1\}, \mathbf{x} \in \mathbb{R}^D$$

Here we assume the data is linearly separable, meaning that we can draw a line on a graph of \mathbf{x}_1 vs \mathbf{x}_2 separating the two classes when $D = 2$ and a hyperplane on graphs of $\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_D$ for when $D > 2$.

This hyperplane can be described by $\mathbf{w} \cdot \mathbf{x} + \mathbf{b} = 0$ where:

- \mathbf{w} is normal to the hyperplane.
- $\frac{b}{\|\mathbf{w}\|}$ is the perpendicular distance from the hyperplane to the origin.

Support Vectors are the examples closest to the separating hyperplane and the aim of Support Vector Machines is to orientate this hyperplane in such a way as to be as far possible from the closest members of both classes.

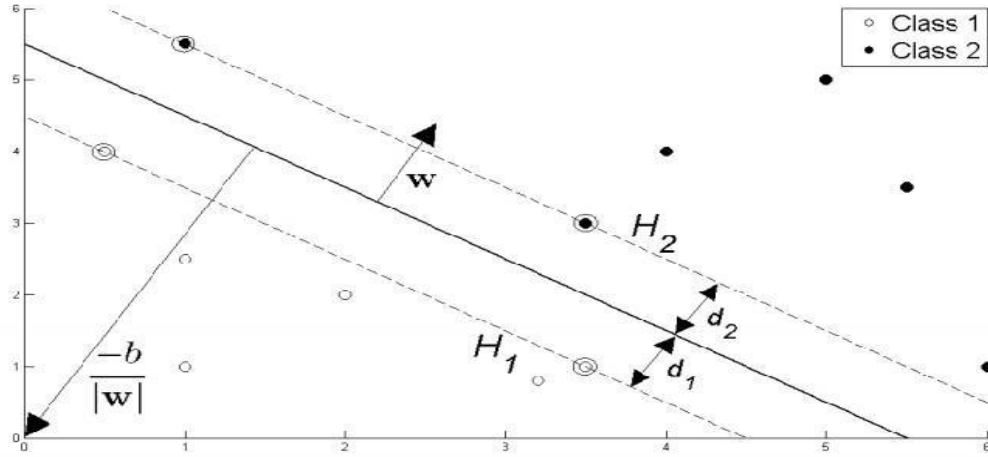


Figure 4.2: Hyperplane through two linearly separable classes.

From Figure 4.2, implementing a SVM boils down to selecting the variables \mathbf{w} and \mathbf{b} so that our training data can be described by:

$$\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b} \geq +1 \quad \text{for } y_i = +1 \quad (4.1)$$

$$\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b} \leq -1 \quad \text{for } y = -1 \quad (4.2)$$

These equations can be combined into:

$$y_i (\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b}) - 1 \geq 0 \quad \forall i \quad (4.3)$$

If we now just, consider the points that lie closest to the separating hyper plane, i.e. the Support Vectors (shown in circles in the diagram), then the two planes H_1 and H_2 that these points lie on can be described by:

$$\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b} = +1 \quad \text{for } H_1 \quad (4.4)$$

$$\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b} = -1 \quad \text{for } H_2 \quad (4.5)$$

Referring to Figure 4.2, we define \mathbf{d}_1 as being the distance from H_1 to the hyperplane and \mathbf{d}_2 from H_2 to it. The hyperplane is equidistance from H_1 and H_2 means that $\mathbf{d}_1 = \mathbf{d}_2$ 'a quantity known as the SVM's margin'. To orientate the hyperplane to be as far from the Support Vectors as possible, we need to maximize this margin.

Simple vector geometry shows that the margin is equal to $\frac{1}{||w||}$ and maximizing subject to the constraint in (4.3) is equivalent to finding:

$$\min ||w|| \quad \text{such that} \quad y_i (x_i \cdot w + b) - 1 \geq 0$$

Minimizing $||w||$ is equivalent to minimizing $\frac{1}{2} ||w||^2$ and the use of this term makes it possible to perform Quadratic Programming (QP) optimization later. We therefore need to find:

$$\min \frac{1}{2} ||w||^2 \quad \text{show that} \quad y_i (x_i \cdot w + b) - 1 \geq 0 \quad \forall i \quad (4.6)$$

To cater for the constraints in this minimization, we need to allocate them with Lagrange multipliers α , where $\alpha_i \geq 0 \quad \forall i$:

$$L_P \equiv \frac{1}{2} ||w||^2 - \alpha [y_i (x_i \cdot w + b) - 1 \quad \forall i] \quad (4.7)$$

$$\equiv \frac{1}{2} ||w||^2 - \sum_{i=1}^L \alpha_i [y_i (x_i \cdot w + b) - 1] \quad (4.8)$$

$$\equiv \frac{1}{2} ||w||^2 - \sum_{i=1}^L \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^L \alpha_i \quad (4.9)$$

We wish to find the w and b which minimizes, and the α which maximizes (4.9) (while keeping $\alpha_i \geq 0 \quad \forall i$). We can do this by differentiating L_P with respect to

w and b setting the derivatives to zero:

$$\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^L \alpha_i y_i x_i \quad (4.10)$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^L \alpha_i y_i = 0 \quad (4.11)$$

Substituting (4.10) and (4.11) into (4.9) gives a new formulation which, being dependent on α , we need to maximize:

$$L_D \equiv \sum_{i=1}^L \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \quad \text{show that} \quad \alpha_i \geq 0 \quad \forall i \quad \sum_{i=1}^L \alpha_i y_i = 0 \quad (4.12)$$

$$\equiv \sum_{i=1}^L \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i H_{ij} \alpha_j \quad \text{where} \quad H_{ij} \equiv y_i y_j x_i \cdot x_j \quad (4.13)$$

$$\equiv \sum_{i=1}^L \alpha_i - \frac{1}{2} \alpha^T H \alpha \quad \text{show that} \quad \alpha_i \geq 0 \quad \forall i, \quad \sum_{i=1}^L \alpha_i y_i = 0 \quad (4.14)$$

This new formulation L_D is referred to as the Dual form of the Primary P . It is worth noting that the Dual form requires only the dot product of each input vector x_i to be calculated, this is important for the Kernel Trick.

Having moved from minimizing L_P to maximizing L_D , we need to find:

$$\max_{\alpha} \left[\sum_{i=1}^L \alpha_i - \frac{1}{2} \alpha^T H \alpha \right], \text{ such that } \alpha_i \geq 0 \forall i \text{ and } \sum_{i=1}^L \alpha_i y_i = 0 \quad (4.15)$$

This is a convex quadratic optimization problem, and we run a QP solver which will return α and from (4.10) will give us w . What remains is to calculate b .

Any data point satisfying (4.11) which is a Support Vector x_s will have the form:

$$y_s (x_s \cdot w + b) = 1$$

Substituting this in (4.10):

$$y_s \left(\sum_{m \in S} \alpha_m y_m x_m \cdot x_s + b \right) = 1$$

Where S denotes the set of indices of the Support Vectors. S is determined by finding the indices i where $\alpha_i > 0$. Multiplying through by y_s and then using $y_s^2 = 1$ from (4.1) and (4.2):

$$\begin{aligned} y_s^2 (\sum_{m \in S} \alpha_m y_m x_m \cdot x_s + b) &= y_s \\ b &= y_s - \sum_{m \in S} \alpha_m y_m x_m \cdot x_s \end{aligned}$$

Instead of using an arbitrary Support Vector x_s , it is better to take an average over all the Support Vectors in S :

$$b = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m x_m \cdot x_s)$$

We now have the variables w and b that define our separating hyperplane's optimal orientation and hence our Support Vector Machine.

Binary Classification for Data that is not Fully Linearly Separable (Soft Margin)

In order to extend the SVM methodology to handle data that is not fully linearly separable, we relax the constraints for (4.1) and (4.2) slightly to allow for misclassified points. This is done by introducing a positive slack variable $\xi_i, i = 1, \dots, L$:

$$x_i \cdot w + b \geq +1 - \xi_i \quad \text{for } y_i = +1$$

$$x_i \cdot w + b \leq -1 + \xi_i \quad \text{for } y_i = -1$$

$$\xi_i \geq 0 \quad \forall i$$

Which can be combined into:

$$y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0 \quad \text{where } \xi_i \geq 0 \quad \forall i$$

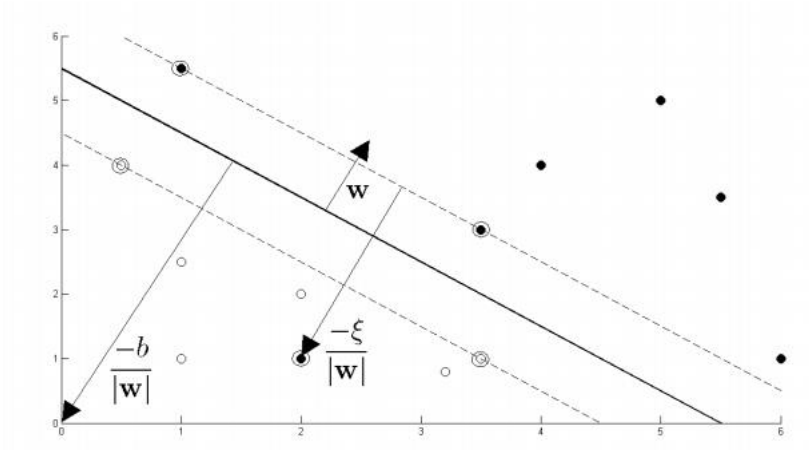


Figure 4.3: Hyperplane through two non-linear separable classes

In this soft margin SVM, data points on the incorrect side of the margin boundary have a penalty that increases with the distance from it. As we are trying to reduce the number of misclassifications, a sensible way to adapt our objective function (4.6) from previously, is to find:

$$\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^L \xi_i \quad \text{such that} \quad y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0 \quad \forall i \quad (4.16)$$

Where the parameter C controls the trade-off between the slack variable penalty and the size of the margin. Reformulating as a Lagrangian, which as

before we need to minimize with respect to \mathbf{w} , \mathbf{b} and ξ_i and maximize with respect to α (where $\alpha_i \geq 0$, $\mu_i \geq 0 \forall i$).

$$L_P \equiv \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^L \xi_i - \sum_{i=1}^L \alpha_i [y_i(x_i \cdot \mathbf{w} + b) - 1 + \xi_i] - \sum_{i=1}^L \mu_i \xi_i \quad (4.17)$$

Differentiating with respect to \mathbf{w} , \mathbf{b} and ξ_i and setting the derivatives to zero:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^L \alpha_i y_i x_i \quad (4.18)$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^L \alpha_i y_i = 0 \quad (4.19)$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow C = \alpha_i + \mu_i \quad (4.20)$$

Substituting these in, L_D has the same form as (4.14) before. However (4.20) together with $\mu_i \geq 0 \forall i$, implies that $\alpha \leq C$. We therefore need to find:

$$\max \alpha, [\sum_{i=1}^L \alpha_i - \frac{1}{2} \alpha^T H \alpha], \text{ such that } 0 \leq \alpha_i \leq C \forall i \text{ and } \sum_{i=1}^L \alpha_i y_i = 0$$

\mathbf{b} is then calculated in the same way as in (4.6) before, though in this instance the set of Support Vectors used to calculate \mathbf{b} is determined by finding the indices i where $0 < \alpha_i < C$.

Non-Linear Support Vector Machine

Non-Linear SVM can be used to classify when it cannot be separated into two classes by a straight line ((in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel function into a higher-dimensional feature space, where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.

When applying our SVM to linearly separable data we have started by creating a matrix \mathbf{H} from the dot product of our input variables:

$$H_{ij} = y_i y_j k(x_i, x_j) = x_i \cdot x_j = x_i^T x_j \quad (4.21)$$

$k(\mathbf{x}_i, \mathbf{x}_j)$ is an example of a family of functions called Kernel Functions ($k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ being known as a Linear Kernel). The set of kernel functions is composed of variants of (4.2) in that they are all based on calculating inner products of two vectors.

This means that if the functions can be recast into a higher dimensionality space by some potentially non-linear feature mapping function $\mathbf{x} \mapsto \phi(\mathbf{x})$, only inner products of the mapped inputs in the feature space need be determined without us needing to explicitly calculate ϕ .

The reason that this Kernel Trick is useful is that there are many classification/regression problems that are not linearly separable in the space of the inputs \mathbf{x} , which might be in a higher dimensionality feature space given a suitable mapping $\mathbf{x} \mapsto \phi(\mathbf{x})$.

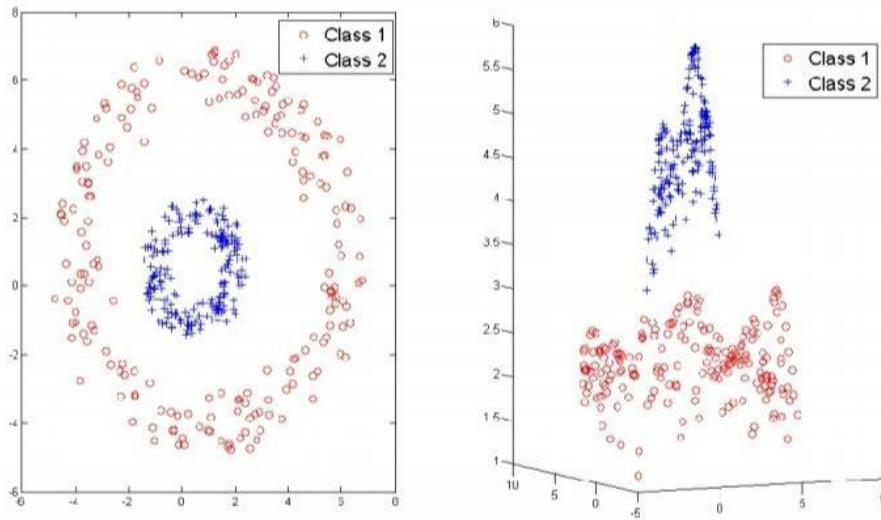


Figure 4.4: Dichotomous data re-mapped using Radial Basis Kernel (RBF)

Referring to Figure 4.4, if we define our kernel to be:

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)} \quad (4.22)$$

then a data set that is not linearly separable in the two-dimensional data space \mathbf{x} (as in the left-hand side of Figure 4.4) is separable in the

nonlinear feature space (right hand side of Figure 4.4) defined implicitly by this non-linear kernel function - known as a Radial Basis Kernel.

Other popular kernels for classification and regression are the Polynomial Kernel

$$k(x_i, x_j) = (x_i \cdot x_j + a)^b$$

and the Sigmoidal Kernel

$$k(x_i, x_j) = \tanh(ax_i \cdot x_j - b)$$

where a and b are parameters defining the kernel's behaviour.

There are many kernel functions, including ones that act upon sets, strings and even music. There are requirements for a function to be applicable as a kernel function that lies beyond the scope of this very brief introduction to the area.

Illustration

To use an SVM to solve a classification problem on data that is not linearly separable, we need to first choose a kernel and relevant parameters which you expect might map the non-linearly separable data into a feature space where it is linearly separable. This is more of an art than an exact science and can be achieved empirically - e.g. by trial and error. Sensible kernels to start with are the Radial Basis, Polynomial and Sigmoidal kernels.

The first step, therefore, consists of choosing our kernel and hence the mapping $x \mapsto \phi(x)$

For classification, we would need to:

- Create H , where $H_{ij} = y_i y_j \phi(x_i) \cdot \phi(x_j)$
- Choose how significantly misclassifications should be treated, by selecting a suitable value for the parameter C .
- Find α so that $\sum_{i=1}^L \alpha_i - \frac{1}{2} \alpha^T H \alpha$ is maximized, subject to the constraints

$$0 \leq \alpha_i \leq C \quad \forall i \quad \text{and}$$

$$\sum_{i=1}^L \alpha_i y_i = 0$$

This is done using a QP solver.

- Calculate

$$w = \sum_{i=1}^L \alpha_i y_i \phi(x_i)$$

- Determine the set of Support Vectors \mathcal{S} by finding the indices such that

$$0 < \alpha_i < C$$

- Calculate $b = \frac{1}{N_s} \sum_{s \in \mathcal{S}} (y_s - \sum_{m \in \mathcal{S}} \alpha_m y_m \phi(x_m) \cdot \phi(x_s))$

Each new point x' is classified by evaluating $y' = \text{sgn}(w \cdot \phi(x') + b)$.

4.3 IMPLEMENTATION OF DATASET

HEART DISEASE PREDICTION

In this section we are implementing the dataset using python code and finding the accuracy of dataset using SVM. Here we are using the heart disease dataset containing 303 samples; 14 features such as age, sex, cp (chest pain type), trtbps (the person's resting blood pressure), chol(cholesterol), fbs (fasting blood sugar), restecg (resting electrocardiographic result), thalachh (the person's maximum heart rate), exang (exercise induced angina), oldpeak (ST depression induced by exercise relative to rest, were ST relates to positions on the ECG plot), slp (the slope of the peak exercise ST segment), caa (number of major vessels), thal (a blood disorder called thalassemia) and the target values 0 (if the person is having the heart disease) and 1 (if the person is not having the heart disease). Now the code for implementing the dataset is given:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, classification_report

# Load the CSV file using Pandas
data = pd.read_csv('/content/heart.csv')

# Assuming the last column is the target column
X = data.iloc[:, :-1] # Features
y = data.iloc[:, -1] # Target variable

# If your data contains non-numeric values, encode them
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling for better performance (optional but often recommended)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize the SVM classifier
svm_classifier = SVC(kernel='linear') # You can change the kernel as per your requirement

# Train the SVM classifier
svm_classifier.fit(X_train, y_train)

# Make predictions on the test set
predictions = svm_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, predictions)
print(f'Accuracy: {accuracy:.2f}')

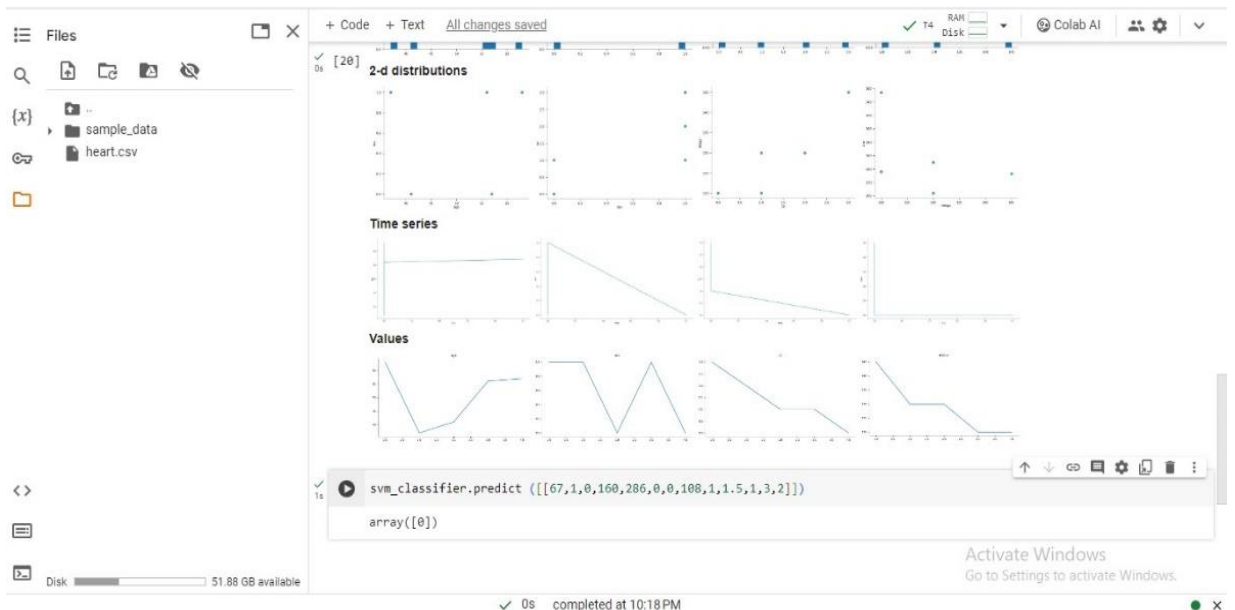
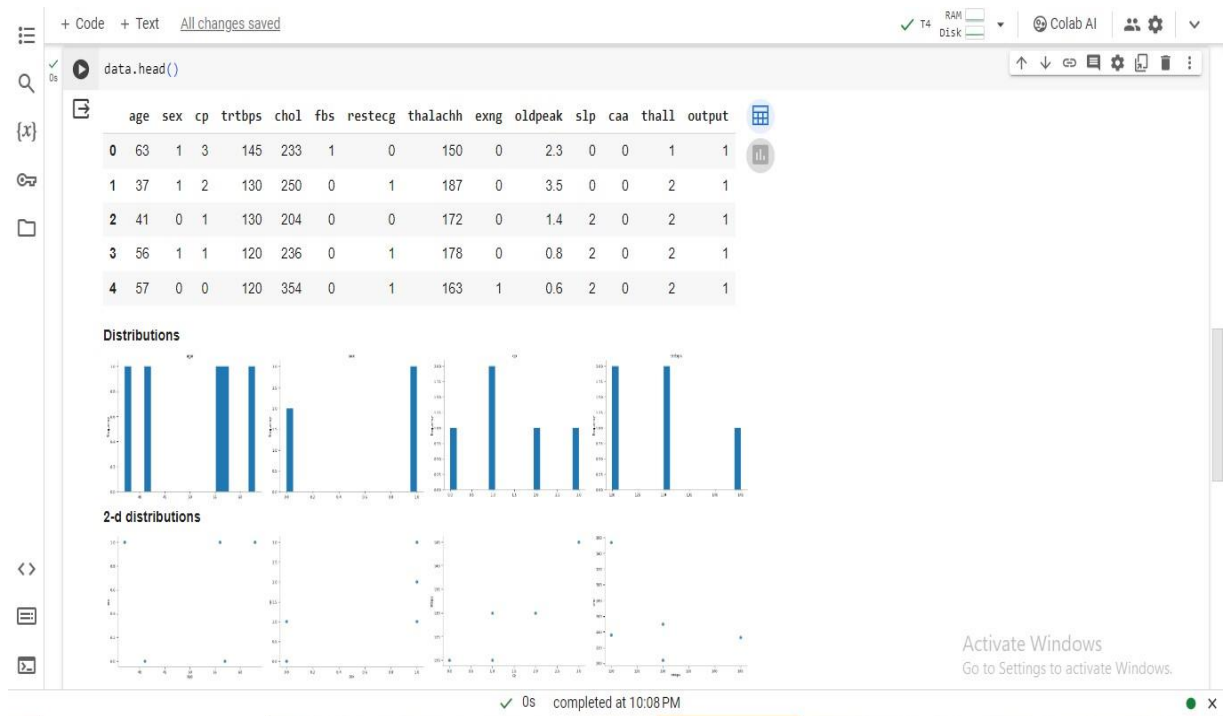
# Print classification report
print(classification_report(y_test, predictions))

# Optionally, you can save the trained model for future use
# from joblib import dump
# dump(svm_classifier, 'svm_model.joblib')
```

Accuracy: 0.87

	precision	recall	f1-score	support
0	0.86	0.86	0.86	29
1	0.88	0.88	0.88	32
accuracy			0.87	61
macro avg	0.87	0.87	0.87	61
weighted avg	0.87	0.87	0.87	61

After implementing the code, we get the accuracy rate as 0.87 for the heart disease dataset. Here we have taken all the features that may cause or may not cause the heart disease.



By using SVM classifier the array we got is 0 which means the person is having heart disease.

4.3 ADVANTAGES AND DISADVANTAGES

Advantages

SVM's are very good when we have no idea on the data. They work well with even unstructured and semi structured data like text, images and trees. SVMs use a subset of training points (support vectors) in the decision function,

which makes them memory efficient, especially when dealing with large datasets. They can be used for both classification and regression tasks. Additionally, they can handle both linear and non-linear relationships between features and targets using appropriate kernels. The kernel trick is real strength of SVM. With an appropriate kernel function, we can solve any complex problem. Unlike in neural networks, SVM is not solved for local optima.

It scales relatively well to high dimensional data. They maximize the margin between classes, which helps in generalization and makes them less prone to overfitting, especially in high-dimensional spaces. It always gets compared with ANN. And when compared to ANN models, SVMs give better results.

Disadvantages

The performance of SVMs can be sensitive to the choice of kernel and its parameters. Selecting the right kernel and tuning its parameters requires domain knowledge and can be time-consuming. Even though SVMs are memory efficient in terms of the number of support vectors used, they can still require a significant amount of memory, particularly when dealing with very large datasets. They may become impractical for very large datasets, both in terms of computational time and memory requirements. They provide very little insight into the relationship between the input variables and the predicted output. They are considered as black box models, which might be a disadvantage in applications where interpretability is crucial.

CHAPTER 5

NAIVE BAYES

Naive Bayes is a simple and efficient algorithm based on the principles of conditional probability, mostly suitable for text classification. It is based on the assumption that all features work independently regardless of the assumption being impossible in its application in real world.

In this chapter, we provide an insight into the Naive Bayes classifier starting from explaining the working behind the algorithm to implementing the algorithm on a dataset using python whilst acknowledging the merits and demerits of the said algorithm.

5.1 Mathematical Framework

5.1.1 Definitions

- **Probability:** Probability is the branch of mathematics and statistics which deals with the chance that an event will occur. Mathematically it is calculated by taking the ratio of the number of times the event occurs to the total number of outcomes. The outcomes are always in between 1 or 0, with 1 being the event will occur and zero being the event won't occur.
- **Conditional Probability:** Conditional Probability is the probability that an event occurs given that another event had previously occurred. Let A and B be two events. The probability of event A occurring given event B had occurred is given by the equation:

$$P(A|B) = \frac{P(AB)}{P(B)} \quad (5.1)$$

In other words, conditional probability is the ratio of the probability that both events A and B occur to the probability that the event B occurs.

5.1.2 Bayes Theorem

Bayes theorem gives an alternate way to find the conditional probability of an event. Suppose A and B are two events and P(B) is not zero. Then Bayes Theorem is stated mathematically as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Here P(A) is called the prior probability, that is, the probability of event A without taking event B into consideration. Event B is called the evidence. The probability of B occurring is called the marginal probability. The conditional probability here is called the posteriori probability of evidence. P(B/A) is the likelihood probability which is the likelihood that a hypothesis will come true given the evidence.

5.2 Naive Bayes Classifier

Naive Bayes classifiers are a set of classifier algorithms based on the Bayes theorem. Naive stands for independence, that is, it is assumed that all the features are independent of each other. Let X be the set of vectors or dependant variables of a dataset and let C be the multi class labels. Assume that a dataset has n set of dependant variables and k class labels.

Assume that there are a total of X vectors and C target variables in a dataset such that $X = \{x_1, x_2, x_3, \dots, x_n\}$, $C_k = \{C_1, C_2, C_3, \dots, C_k\}$

For every target variable C_i

The value of i ranging from 1 to k, we have to find

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)}$$

P(X) is a common term in every $P(C_i|X)$, therefore, it is removed as it doesn't contribute much.

$$P(C_k|X) = P(X|C_k)P(C_k) \quad (5.2)$$

Comparing equations (5.1) and (5.2), we get

$$P(C_k|X) = P(X, C_k) = P(X_1, X_2, X_3, \dots, X_n, C_k)$$

Here we take an assumption that $X_1 = A$

And the rest of the terms as B. By equation (5.1), the equation gets transformed into $P(C_k|X) = P(A, B) = P(A|B) P(B) =$

$$P(X_1|X_2, X_3, \dots, X_n, C_k) P(X_2, X_3, \dots, X_n, C_k) \quad (5.3)$$

Take $a = P(X_1|X_2, X_3, \dots, X_n, C_k)$

Substituting a in (5.3), we get

$$P(C_k|X) = a * P(X_2, X_3, \dots, X_n, C_k)$$

Again, assume $X_2 = A$

And equate the rest of the terms as B like before and applying conditional probability again

$$P(C_k|X) = aP(X_2|X_3, X_4, \dots, X_n, C_k)P(X_3, X_4, \dots, X_n, C_k)$$

$$= abP(X_3, X_4, \dots, X_n, C_k)$$

(Chain Rule for conditional probability)

Repeating the procedure again till the last term is reached,

$$P(C_k|X) = abc \dots P(X_n|C_k)P(C_k)$$

$$P(C_k|X)$$

$$= P(X_1|X_2, X_3, \dots, X_n, C_k)P(X_2|X_3, X_4, X_5, \dots, X_n, C_k)P(X_3|X_4, X_5, \dots,$$

$$X_n, C_k) \dots P(X_{n-1}|X_n, C_k)P(X_n|C_k)P(C_k)$$

For implementation in Naive Bayes, assume that all the features are Naive. We assume that all the X_i are all independent of each other and only dependent on C_k for all terms. This assumption is called conditional independence. $P(A|B) = P(A)$

For independent events.

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

Is the conditional independence when B is independent,

$$P(C_k|X) = P(X_1|C_k)P(X_2|C_k) \dots P(X_n|C_k)P(C_k)$$

n

$$P(C_k|X) = P(C_k) \prod_{i=1}^n P(X_i|C_k)$$

As we haven't considered the denominator, the equation changes into

n

$$P(C_k|X) \propto P(C_k) \prod_{i=1}^n P(X_i|C_k)$$

$$P(C_k|x) = \frac{1}{Z} P(C_k) \prod_{i=1}^n P(X_i|C_k)$$

Where $Z=P(X)$

Test the above equation with each class and the class producing the maximum value is chosen as y .

$$y = \arg \max P(C_k) \prod_{i=1}^n P(X_i|C_k)$$

Remark: Here, the value of y is chosen by the Maximum A Posteriori rule (MAP). MAP estimate of the random variable X , is the value of x that maximises.

5.3 Types of Naive Bayes algorithms:

Gaussian Naive Bayes:

It implements the algorithm based on the Gaussian or Normal distribution.

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

Multinomial Naive Bayes:

It is used for multinomially distributed data.

$$\theta_{yi} = \frac{(N_{yi} + \alpha)}{N_y + \alpha n}$$

Here, θ is the probability $P(x_i|y)$, N_{yi} is the number of times i appears in a sample of class y , N_y is the total count of all features of class y .

Complement Naive Bayes:

$$\theta_{yi} = \frac{\alpha_i + \sum_{j:y_j \neq c} d_{ij}}{\alpha + \sum_{j:y_j \neq c} \sum_k d_{kj}}$$

$$c = \arg \min \sum_i t_i w_{ci}$$

It is more suitable for imbalanced datasets. It performs better at text classification than Multinomial Naive Bayes and as the name denotes, uses complements of each class to compute the model's weights.

Bernoulli Naive Bayes:

It works on the basis of multivariate Bernoulli distributions. So its usage of preference is on binary valued features.

$$\mathbf{P}(\mathbf{X}_i|\mathbf{y}) = \mathbf{P}(\mathbf{x}_i = \mathbf{1}|\mathbf{y})\mathbf{x}_i + (\mathbf{1} - \mathbf{P}(\mathbf{x}_i = \mathbf{1}|\mathbf{y}))$$

Categorical Naive Bayes:

It is used for categorically distributed data. It assumes that each feature has its own categorical distribution.

$$P(x_i = t|y = c; \alpha) = \frac{N_{tic} + \alpha}{N_c + \alpha n_i}$$

Where N_{tic} is the number of times category t appears in the sample x_i which belongs to class c . N_c is the number of samples with class c , α is the smoothing parameter and n_i is the number of available categories of feature i .

5.4 Implementation of dataset

Let us now implement the algorithm on a dataset using Python code. Here, in this case, we take a dataset of 14 entries to predict the target variable 'play' using the attributes 'Outlook', 'Temperature', 'Humidity' and 'Windy'. The dataset is given below:

	A	B	C	D	E
1	Outlook	Temperature	Humidity	Windy	Play
2	Sunny	Hot	High	Weak	No
3	Sunny	Hot	High	Strong	No
4	Overcast	Hot	High	Weak	Yes
5	Rain	Mild	High	Weak	Yes
6	Rain	Cool	Normal	Weak	Yes
7	Rain	Cool	Normal	Strong	No
8	Overcast	Cool	Normal	Strong	Yes
9	Sunny	Mild	High	Weak	No
10	Sunny	Cool	Normal	Weak	Yes
11	Rain	Mild	Normal	Weak	Yes
12	Sunny	Mild	Normal	Strong	Yes
13	Overcast	Mild	High	Strong	Yes
14	Overcast	Hot	Normal	Weak	Yes
15	Rain	Mild	High	Strong	No

From the dataset, it is visible that the first four columns act as the dependent variables on which the output column (play) is predicted.

The code for implementing the algorithm:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from google.colab import drive

# This will prompt you to click on a link and get an authorization code
drive.mount('/content/gdrive')
```

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.moun

Numpy, matplotlib etc are various libraries in python which helps the programming process easier.

```
import os

# List the contents of your Google Drive
os.listdir('/content/gdrive/My Drive/')

['Front Page (1).gdoc',
 'Front Page.gdoc',
 'Colab Notebooks',
 'Project ',
 'Naive Bayes .gslides',
 'MathLens .gsheet']
```

```

import pandas as pd

# Load the dataset (replace 'your_dataset.csv' with the actual name of your dataset)
csv_path = '/content/gdrive/MyDrive/Project /weather_forecast.csv'
df = pd.read_csv(csv_path)

```

```

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   Outlook         14 non-null    object
1   Temperature     14 non-null    object
2   Humidity        14 non-null    object
3   Windy           14 non-null    object
4   Play            14 non-null    object
dtypes: object(5)

```

```

df['Play'].value_counts()

Yes      9
No       5
Name: Play, dtype: int64

```

```

import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB

```

Here, the Gaussian Naive Bayes is chosen because of its popularity of being able to be implemented easily and its usage of mean and standard deviation for training the dataset which is suitable for the type of data used here.

```

Outlook=LabelEncoder()
Temperature=LabelEncoder()
Humidity=LabelEncoder()
Windy=LabelEncoder()

inputs=df.drop('Play',axis='columns')
target=df['Play']

```

The label encoder is implemented on the variable columns to transform the categorical data into numerical data.

0	No
1	No
2	Yes
3	Yes
4	Yes
5	No
6	Yes
7	No
8	Yes

9	Yes
10	Yes
11	Yes
12	Yes
13	No

Name: Play, dtype: object

```

inputs['Outlook_n']= Outlook.fit_transform(inputs['Outlook'])
inputs['Temp_n']= Temperature.fit_transform(inputs['Temperature'])
inputs['Hum_n']= Humidity.fit_transform(inputs['Humidity'])
inputs['win_n']= Windy.fit_transform(inputs['Windy'])

inputs_n=inputs.drop(['Outlook','Temperature','Humidity','Windy'],axis='columns')
inputs_n

```

	Outlook_n	Temp_n	Hum_n	win_n
0	2	1	0	1
1	2	1	0	0
2	0	1	0	1
3	1	2	0	1
4	1	0	1	1
5	1	0	1	0
6	0	0	1	0
7	2	2	0	1
8	2	0	1	1
9	1	2	1	1
10	2	2	1	0
11	0	2	0	0
12	0	1	1	1
13	1	2	0	0

```

classifier = GaussianNB()
classifier.fit(inputs_n,target)

```

GaussianNB

GaussianNB()

```
classifier.predict([[0,0,0,1]])  
  
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have  
warnings.warn(  
array(['Yes'], dtype='<U3')  
  
classifier.score(inputs_n,target)  
  
0.9285714285714286
```

After implementation, the classifier is put to test by testing out its predictive ability.

```
classifier.predict([[0,0,0,1]])  
  
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have  
warnings.warn(  
array(['Yes'], dtype='<U3')  
  
classifier.score(inputs_n,target)  
  
0.9285714285714286
```

Here [0,0,0,1] stands for ‘Overcast’ in ‘Outlook’, ‘Cool’ in ‘Temperature’, ‘High’ in ‘Humidity’ and ‘Weak’ in ‘Windy’ attributes. Calculating the probability accordingly, it produces the output ‘Yes’ which implies the weather is apt for playing.

5.3 Advantages and Disadvantages

Advantages

One of the merits that adds to the popularity of this classifier algorithm is that it’s simple to implement. Naive Bayes doesn’t need much computation and can be used on both binary and multi class features. It handles missing data by only acknowledging the present data given in the dataset. The different algorithm gives for implementation on various data regardless of its limited boundaries. Compared to other algorithms, Naive Bayes require less training data as all features are assumed to be independent. As the algorithm requires only a small training data, it works well for small datasets.

Disadvantages

Even with all the perks, this classifier algorithm, famous for its easy implementation shares its equal share of downfalls, the major and obvious reason being its assumption of independence of features. Due to this, it doesn't capture much relations between features and may result in inaccurate results with high bias for specific classes as the method do not include noise data for the process. Naive Bayes has difficulty handling continuous data variables and often show bias towards features with high frequency. As a result, imbalanced sets work poorly for this algorithm. One of the major disadvantages is the case of zero frequency. It occurs when the test data contains a category which wasn't present in the training data. As a result, it will assign zero probability to that category and thus, renders it from appearing in the outcome as prediction.

BIBLIOGRAPHY

1. Boswell, Dustin. "Introduction to support vector machines." *Departement of Computer Science and Engineering University of California San Diego* 11, 2002.
2. Chowdhury, Sultana Mubarika Rahman. "A Review of Logistic Regression and its Application.", 2021.
3. Das, Chandramouli, Dutt. Education, Pearson. *Machine Learning, 1e*. Pearson Education India, 2019.
4. Eldridge, Stephen. "conditional Probability". Encyclopaedia Britannica, <https://www.britannica.com/science/conditional-probability>. Accessed 22 January 2024
5. Fletcher, Tristan. "Support vector machines explained." *Tutorial paper*, 2009.
6. Jain, Akshay. "Advantages and Disadvantages of Logistic Regression in Machine Learning." *Medium*, medium.com/@akshayjain_757396/advantages-and-disadvantages-of-logistic-regression-in-machine-learning-a6a247e42b20, July 2020.
7. Patel, Fenin. *LinkedIn*, <https://www.linkedin.com/pulse/decision-tree-id3-algorithm-maths-behind-fenil-patel>, Oct 2022.
8. Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *the Journal of machine Learning research* 12: 2825-2830, 2011.
9. "Probability." Merriam-Webster.com Dictionary, Merriam-Webster, <https://www.merriam-webster.com/dictionary/probability>. Accessed 24 Jan. 2024
10. Rai, Khuswant. "The Math behind Logistic Regression." *Medium*, Analytics Vidya, medium.com/analytics-vidhya/the-math-behind-logistic-regression-c2f04ca27bca, June 2020.
11. Sasidharan, Aswati. "Support Vector Machine (SVM) Algorithm." *Geeksforgeeks.Org*, www.google.com/amp/s/www.geeksforgeeks.org/support-vector-machine-algorithm/amp/. Accessed June 2023.
12. Sharma, Abhishek K. "Decision Tree in Machine Learning." *Geeksforgeeks.Org*, www.geeksforgeeks.org/decision-tree-introduction-example/. Accessed Dec. 2023.
13. "Logistic Regression in Machine Learning." *Geeksforgeeks.Org*, www.geeksforgeeks.org/understanding-logistic-regression/. Accessed Jan. 2024.