RAINFALL FORECASTING IN AUSTRALIA

Machine Learning Approach

A report submitted in partial fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

By

BUDUMURU YUGANDHAR

Reg.no : 21B91A0426

Under Supervision of Mr. Gundala Nagaraju

Henotic Technology Pvt Ltd, Hyderabad

(Duration: 5th July 2023 to 5th September 2023)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

S.R.K.R. ENGINEERING COLLEGE

(Autonomous)

SRKR MARG, CHINNA AMIRAM, BHIMAVARAM-534204, A.P

(Recognized by A.I.C.T.E New Delhi) (Accredited by NBA & NAAC)

(Affiliated to JNTU, KAKINADA)

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE (Autonomous)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ESTD:1980

CERTIFICATE

This is to certify that the Summer Internship Report titled " **RAINFALL PREDICTION IN AUSTRALIA**" is the bonafide work done by **Mr.BUDUMURU YUGANDHAR** bearing Register Number **21B91A0426** at the end of second year second semester at **Henotic Technology Pvt Ltd**, Hyderabad from **5th July 2023** to **5th September 2023** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Electronics and Communication Engineering**.

Department Internship coordinator **Dean -T & P Cell** **Head of the Department**

ACKNOWLEDGEMENT

I would like to acknowledge the contributions of the following people without whose help and guidance this report would not have been completed.

I acknowledge the counsel and support of our training coordinator, **Mr. Gundala Nagaraju, Henotic Technology Pvt Ltd, Hyderabad,** with respect and gratitude, whose expertise, guidance, support, encouragement, and enthusiasm has made this report possible. Their feedback vastly improved the quality of this report and provided an enthralling experience. I am indeed proud and fortunate to be supported by him.

I am also thankful to **Prof. (Dr.) N. UDAYA KUMAR**, **H.O.D of Electronics and Communication Engineering Department, SRKR Engineering College, Bhimavaram** for his constant encouragement, valuable suggestions and moral support and blessings.
Although it is not possible to name individually, I shall ever remain indebted to the Dean of the Training and placement cell, **Dr. K. R. Satyanarayana sir** for his persistent support and cooperation extended during this work.

This acknowledgement will remain incomplete if I fail to express our deep sense of obligation to my parents and God for their consistent blessings and encouragement.

BUDUMURU YUGANDHAR

21B91A0426

Table of contents:-

i. Front Page
ii. Certificate
iii. Acknowledgement

1. Introduction to machine learning
a. History of machine learning
b. Types of machine learning
i. Supervised learning
ii. Unsupervised learning
iii. Reinforcement learning
iv. Semi-supervised learning
c. Challenges of machine learning
d. Understanding of human
e. Applications of machine learning
f. Optimization Training workshop
g. Objectives
h. Methodologies
2. Technology Implemented
a. Python- The new generation language
b. Features
c. Python is perfect for machine learning
d. Data preparation, Analysis and Visualization
e. Machine learning Algorithms 3. Result Discussion
a. Result
b. Overview
c. Dataset Description
d. Data source
e. Machine learning model result
f. Project code and output 4. Advantages and Disadvantages
a. Advantages
b. Disadvantages
5. Conclusion

Chapter 1 Introduction to Machine Learning

Machine Learning is the science of getting computers to learn without being explicitly programmed. It is closely related to computational statistics, which focuses on making prediction using computer. In its application across business problems, machine learning is also referred as predictive analysis. Machine Learning is closely related to computational statistics. Machine Learning focuses on the development of computer programs that can access data and use it to learn themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

History of Machine Learning

The name machine learning was coined in 1959 by Arthur Samuel. Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "**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***." This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?". In Turing's proposal the characteristics that could be possessed by a thinking machine and the various implications in constructing one are exposed.

Types of Machine Learning

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve. Broadly Machine Learning can be categorized into four categories.

I. Supervised Learning
II. Unsupervised Learning
III. Reinforcement Learning
IV. Semi-supervised Learning

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly.

I.Supervised Learning

Supervised Learning is a type of learning in which we are given a data set and we already know what are correct output should look like, having the idea that there is a relationship between the input and output. Basically, it is learning task of learning a function that maps an input to an output based on example input- output pairs. It infers a function from labeled training data consisting of a set of training examples.

Supervised learning problems are categorized

II.Unsupervised Learning

Unsupervised Learning is a type of learning that allows us to approach problems with little or no idea what our problem should look like. We can derive the structure by clustering the data based on a relationship among the variables in data. With unsupervised learning there is no feedback based on prediction result. Basically, it is a type of

selforganized learning that helps in finding previously unknown patterns in data set without pre-existing label.

III.Reinforcement Learning

Reinforcement learning is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best.

IV.Semi-Supervised Learning

Semi-supervised learning fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.

The Challenges Facing Machine Learning

While there has been much progress in machine learning, there are also challenges. For example, the mainstream machine learning technologies are black-box approaches, making us concerned about their potential risks. To tackle this challenge, we may want to make machine learning more explainable and controllable. As another example, the computational complexity of machine learning algorithms is usually very high and we may want to invent lightweight

algorithms or implementations. Furthermore, in many domains such as physics, chemistry, biology, and social sciences, people usually seek elegantly simple equations (e.g., the Schrödinger equation) to uncover the underlying laws behind various phenomena. Machine learning takes much more time. You have to gather and prepare data, then train the algorithm

Applications of Machine Learning

Applications of Machine Learning include:

- **Web Search Engine:** One of the reasons why search engines like google, bing etc work so well is because the system has learnt how to rank pages through a complex learning algorithm.

- **Photo tagging Applications:** Be it facebook or any other photo tagging application, the ability to tag friends makes it even more happening. It is all possible because of a face recognition algorithm that runs behind the application.

- **Spam Detector:** Our mail agent like Gmail or Hotmail does a lot of hard work for us in classifying the mails and moving the spam mails to spam folder. This is again achieved by a spam classifier running in the back end of mail application.

- **Database Mining for growth of automation:** Typical applications include Web-click data for better UX, Medical records for better automation in healthcare, biological data and many more.

- **Applications that cannot be programmed:** There are some tasks that cannot be programmed as the computers we use are not modelled that way. Examples include Autonomous Driving, Recognition tasks from unordered data (Face Recognition/ Handwriting Recognition), Natural language Processing, computer Vision etc.

Understanding Human Learning:

This is the closest we have understood and mimicked the human brain. It is the start of a new revolution, The real AI. Now, after a brief insight lets come to a more formal definition of Machine LearningFuture of Machine Learning is as vast as the limits of human mind. We can always keep learning, and teaching the computers how to learn. And at the same time, wondering how some of the most complex machine learning algorithms have been running in the back of our own mind so effortlessly all the time. There is a bright future for machine learning. Companies like Google, Quora, and Facebook hire people with machine learning. There is intense research in machine learning at the top universities in the world. The global machine learning as a service market is rising expeditiously mainly due to the Internet revolution. The process of connecting the world virtually has generated vast amount of data which is boosting the adoption of machine learning solutions. Considering all these applications and dramatic improvements that ML has brought us, it doesn't take a genius to realize that in coming future we will definitely see more advanced applications of ML, applications that will stretch the capabilities of machine learning to an unimaginable level.

Organization of Training Workshop

Company Profile:

Henotic Technology pvt ltd. was created with a mission to create skilled software engineers for our country and the world. It aims to bridge the gap between the quality of skills demanded by industry and the quality of skills imparted by conventional institutes. With assessments, learning paths and courses authored by industry experts, DreamUny helps businesses and individuals benchmark expertise across roles, speed up release cycles and build reliable, secure products.

Objectives

Main objectives of training is to learn:

- How to determine and measure program complexity,
- Python Programming
- ML Library Scikit, Numpy , Matplotlib, Pandas , Theano , TensorFlow Statistical Math for the Algorithms.
- Learning to solve statistics and mathematical concepts.

- Supervised and Unsupervised Learning
- Classification and Regression
- ML Algorithms
- Machine Learning Programming and Use Cases.

Methodologies

There were several facilitation techniques used by the trainer which included question and answer, brainstorming, group discussions, case study discussions and practical implementation of some of the topics by trainees on flip charts and paper sheets. The multitude of training methodologies was utilized in order to make sure all the participants get the whole concepts and they practice what they learn, because only listening to the trainers can be forgotten, but what the trainees do by themselves they will never forget. After the post-tests were administered and the final course evaluation forms were filled in by the participants, the trainer expressed his closing remarks and reiterated the importance of the training for the trainees in their daily activities and their readiness for applying the learnt concepts in their assigned tasks. Certificates of completion were distributed among the participants at the end.

Chapter 2 Technology Implemented

Python – The New Generation Language:

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for an emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Features

- **Interpreted**
  - In Python there is no separate compilation and execution steps like C/C++. It directly run the program from the source code. Internally, Python converts the source code into an intermediate form called bytecodes which is then translated into native language of specific computer to run it.
- **Platform Independent**
  - Python programs can be developed and executed on the multiple operating system platform. Python can be used on Linux, Windows, Macintosh, Solaris and many more.
- **Multi- Paradigm**
  - Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect- oriented programming .
- **Simple**

 Python is a very simple language. It is a very easy to learn as it is closer to English language. In python more emphasis is on the solution to the problem rather than the syntax.

 Rich Library Support

 Python standard library is very vast. It can help to do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, email, XML, HTML, WAV files, cryptography, GUI and many more.

 Free and Open Source

 Firstly, Python is freely available. Secondly, it is open-source. This means that its source code is available to the public. We can download it, change it, use it, and distribute it. This is called FLOSS (Free/Libre and Open Source Software).

As the Python community, we're all headed toward one goal- an ever-bettering Python.

Why Python Is a Perfect Language for Machine Learning?

For the above question the answer is,

 A great library ecosystem -

A great choice of libraries is one of the main reasons Python is the most popular programming language used for AI. A library is a module or a group of modules published by different sources which include a pre-written piece of code that allows users to reach some functionality or perform different actions. Python libraries provide base level items so developers don't have to code them from the very beginning every time. ML requires continuous data processing, and Python's libraries let us access, handle and transform data. These are some of the most widespread libraries you can use for ML and AI:

- Scikit-learn for handling basic ML algorithms like clustering, linear and logistic regressions, regression, classification, and others.

- Pandas for high-level data structures and analysis. It allows merging and filtering of data, as well as gathering it from other external sources like Excel, for instance.
- Keras for deep learning. It allows fast calculations and prototyping, as it uses the GPU in addition to the CPU of the computer.
- TensorFlow for working with deep learning by setting up, training, and utilizing artificial neural networks with massive datasets.
- Matplotlib for creating 2D plots, histograms, charts, and other forms of visualization.
- NLTK for working with computational linguistics, natural language recognition, and processing.
- Scikit-image for image processing.
- PyBrain for neural networks, unsupervised and reinforcement learning.
- Caffe for deep learning that allows switching between the CPU and the GPU and processing 60+ mln images a day using a single NVIDIA K40 GPU.
- StatsModels for statistical algorithms and data exploration.In the PyPI repository, we can discover and compare more python libraries.

✞ **A low entry barrier -**

Working in the ML and AI industry means dealing with a bunch of data that we need to process in the most convenient and effective way. The low entry barrier allows more data scientists to quickly pick up Python and start using it for AI development without wasting too much effort into learning the language. In addition to this, there's a lot of documentation available, and Python's community is always there to help out and give advice.

✞ **Flexibility-**

Python for machine learning is a great choice, as this language is very flexible:

- It offers an option to choose either to use OOPs or scripting.
- There's also no need to recompile the source code, developers can implement any changes and quickly see the results.

- Programmers can combine Python and other languages to reach their goals.

- **Good Visualization Options-**

  For AI developers, it's important to highlight that in artificial intelligence, deep learning, and machine learning, it's vital to be able to represent data in a humanreadable format. Libraries like Matplotlib allow data scientists to build charts, histograms, and plots for better data comprehension, effective presentation, and visualization. Different application programming interfaces also simplify the visualization process and make it easier to create clear reports.

- **Community Support-**

  It's always very helpful when there's strong community support built around the programming language. Python is an open-source language which means that there's a bunch of resources open for programmers starting from beginners and ending with pros. A lot of Python documentation is available online as well as in Python communities and forums, where programmers and machine learning developers discuss errors, solve problems, and help each other out. Python programming language is absolutely free as is the variety of useful libraries and tools.

- **Growing Popularity-**

  As a result of the advantages discussed above, Python is becoming more and more popular among data scientists. According to StackOverflow, the popularity of Python is predicted to grow until 2020, at least. This means it's easier to search for developers and replace team players if required. Also, the cost of their work maybe not as high as when using a less popular programming language.

Data Preprocessing, Analysis & Visualization

Machine Learning algorithms don't work so well with processing raw data. Before we can feed such data to an ML algorithm, we must preprocess it. We must apply some transformations on it. With data preprocessing, we convert raw data into a clean data set. To perform data this, there are 7 techniques -

- **Rescaling Data -**

  For data with attributes of varying scales, we can rescale attributes to possess the same scale. We rescale attributes into the range 0 to 1 and call it normalization. We use the MinMaxScaler class from scikit- learn. This gives us values between 0 and 1.

- **Standardizing Data -**

  With standardizing, we can take attributes with a Gaussian distribution and different means and standard deviations and transform them into a standard Gaussian distribution with a mean of 0 and a standard deviation of 1.

- **Normalizing Data -**

  In this task, we rescale each observation to a length of 1 (a unit norm). For this, we use the Normalizer class.

- **Binarizing Data -**

  Using a binary threshold, it is possible to transform our data by marking the values above it 1 and those equal to or below it, 0. For this purpose, we use the Binarizer class.

- **Mean Removal-**

  We can remove the mean from each feature to center it on zero.

- **One Hot Encoding -**

When dealing with few and scattered numerical values, we may not need to store these. Then, we can perform One Hot Encoding. For k distinct values, we can transform the feature into a k-dimensional vector with one value of 1 and 0 as the rest values.

✞ **Label Encoding -**

Some labels can be words or numbers. Usually, training data is labelled with words to make it readable. Label encoding converts word labels into numbers to let algorithms work on them.

Machine Learning Algorithms

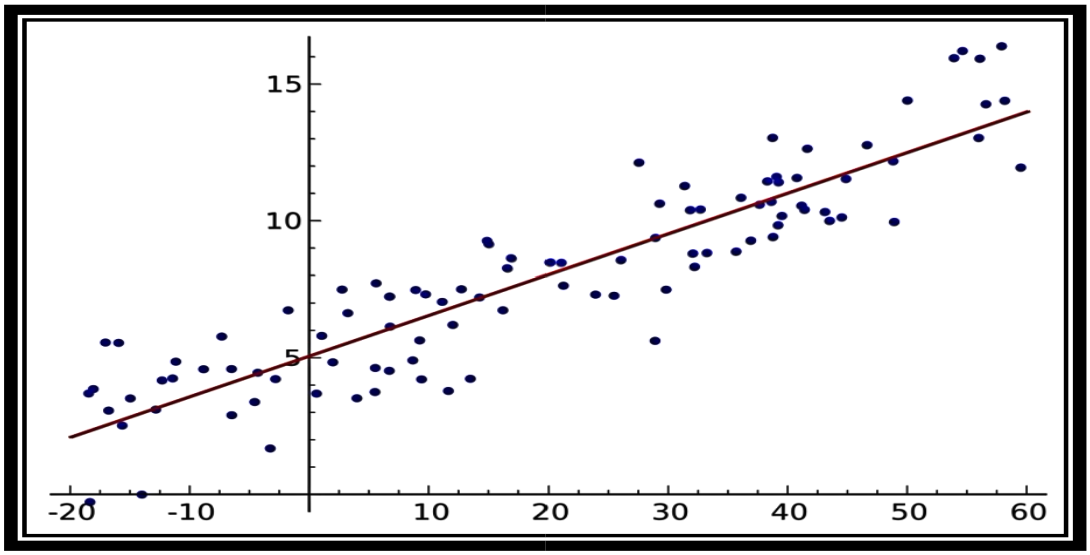
There are many types of Machine Learning Algorithms specific to different use cases. As we work with datasets, a machine learning algorithm works in two stages. We usually split the data around 20%-80% between testing and training stages. Under supervised learning, we split a dataset into a training data and test data in Python ML. Followings are the Algorithms of Python Machine Learning

1.Linear Regression-

Linear regression is one of the supervised Machine learning algorithms in Python that observes continuous features and predicts an outcome. Depending on whether it runs on a single variable or on many features, we can call it simple linear regression or multiple linear regression.

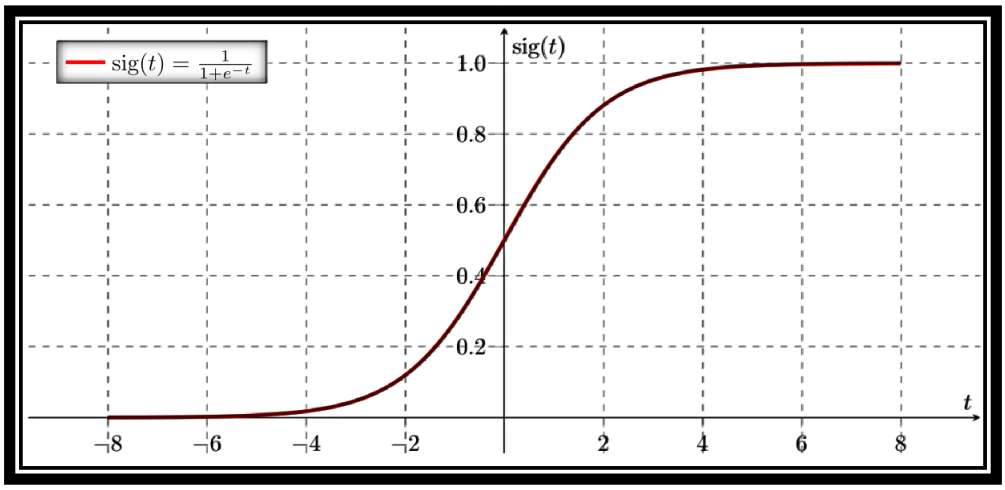
This is one of the most popular Python ML algorithms and often underappreciated. It assigns optimal weights to variables to create a line ax+b to predict the output. We often use linear regression to estimate real values like a

number of calls and costs of houses based on continuous variables. The regression line is the best line that fits Y=a*X+b to denote a relationship between independent and dependent variables.



2.Logistic Regression -

Logistic regression is a supervised classification is unique Machine Learning algorithms in Python that finds its use in estimating discrete values like 0/1, yes/no, and true/false. This is based on a given set of independent variables. We use a logistic function to predict the probability of an event and this gives us an output between 0 and 1. Although it says 'regression', this is actually a classification algorithm. Logistic regression fits data into a logit function and is also called logit regression.



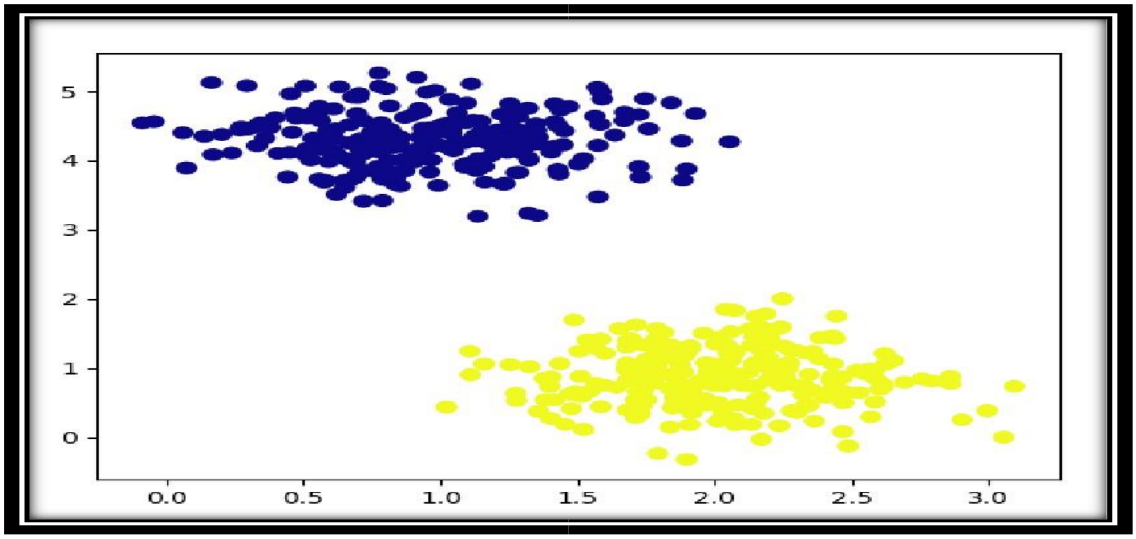
3.Decision Tree -

A decision tree falls under supervised Machine Learning Algorithms in Python and comes of use for both classification and regression- although mostly for classification. This model takes an instance, traverses the tree, and compares important features with a determined conditional statement. Whether it descends to the left child branch or the right depends on the result. Usually, more important features are closer to the root.

Decision Tree, a Machine Learning algorithm in Python can work on both categorical and continuous dependent variables. Here, we split a population into two or more homogeneous sets. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

4.Support Vector Machine (SVM)-

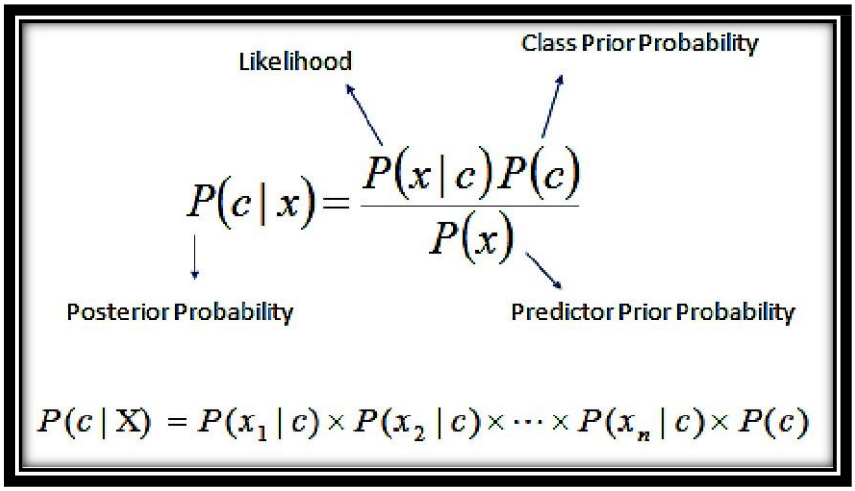
SVM is a supervised classification is one of the most important Machines Learning algorithms in Python, that plots a line that divides different categories of your data. In this ML algorithm, we calculate the vector to optimize the line. This is to ensure that the closest point in each group lies farthest from each other. While you will almost always find this to be a linear vector, it can be other than that. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. When data are unlabeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups.



5.Naïve Bayes Algorithm -

Naive Bayes is a classification method which is based on Bayes' theorem. This assumes independence between predictors. A Naive Bayes classifier will assume that a feature in a class is unrelated to any other. Consider a fruit. This is an apple

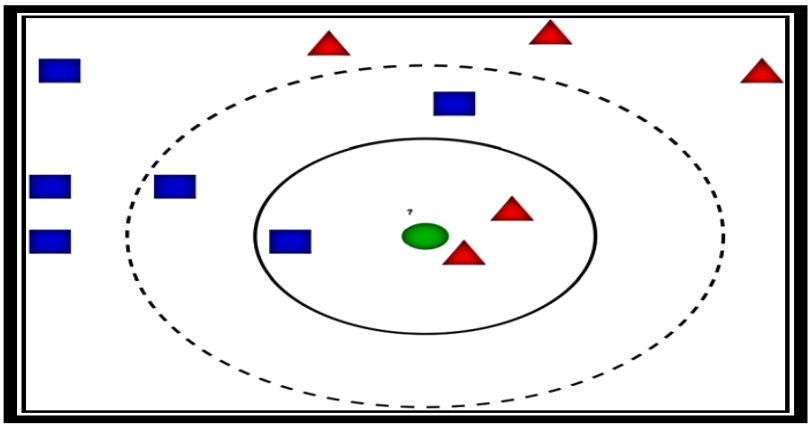
if it is round, red, and 2.5 inches in diameter. A Naive Bayes classifier will say these characteristics independently contribute to the probability of the fruit being an apple. This is even if features depend on each other. For very large data sets, it is easy to build a Naive Bayesian model. Not only is this model very simple, it performs better than many highly sophisticated classification methods. Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.



6.KNN Algorithm -

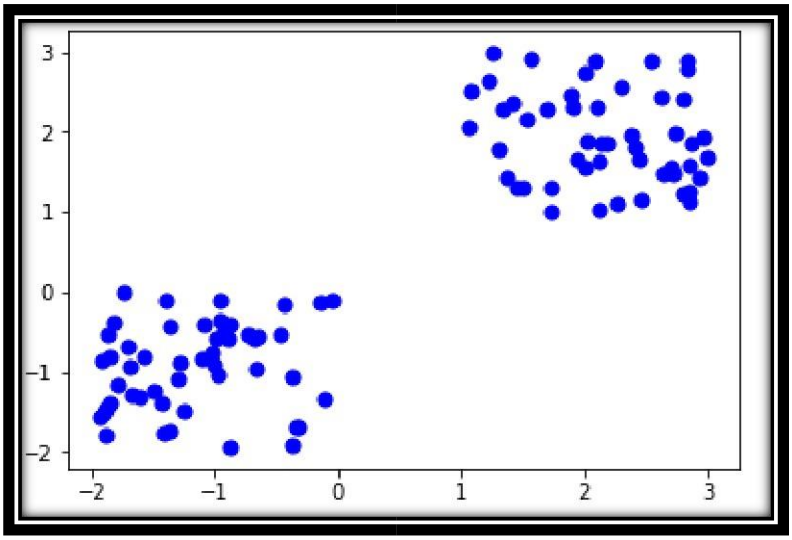
This is a Python Machine Learning algorithm for classification and regression-mostly for classification. This is a supervised learning algorithm that considers different centroids and uses a usually Euclidean function to compare distance. Then, it analyzes the results and classifies each point to the group to optimize it to place with all closest points to it. It classifies new cases using a majority vote of k of its neighbors. The case it assigns to a class is the one most common among its K nearest neighbors. For this, it uses a distance function. *k*-NN is a type of

instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. *k*-NN is a special case of a variable- bandwidth, kernel density "balloon" estimator with a uniform kernel



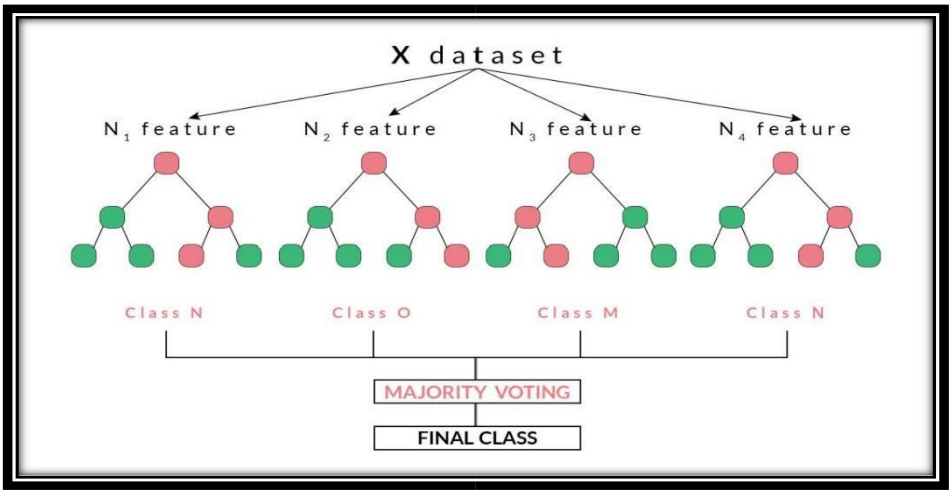
7.K-Means Algorithm -

k-Means is an unsupervised algorithm that solves the problem of clustering. It classifies data using a number of clusters. The data points inside a class are homogeneous and heterogeneous to peer groups. *k*-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. *k*-means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. *k*-means clustering is rather easy to apply to even large data sets, particularly when using heuristics such as Lloyd's algorithm. It often is used as a preprocessing step for other algorithms, for example to find a starting configuration. The problem is computationally difficult (NP-hard). *k*-means originates from signal processing, and still finds use in this domain. In cluster analysis, the *k*-means algorithm can be used to partition the input data set into *k* partitions (clusters). *k*-means clustering has been used as a feature learning (or dictionary learning) step, in either (semi-)supervised learning or unsupervised learning



8.Random Forest -

A random forest is an ensemble of decision trees. In order to classify every new object based on its attributes, trees vote for class- each tree provides a classification. The classification with the most votes wins in the forest. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.



Chapter -3 Result Discussion

Result

This training has introduced us to Machine Learning. Now, we know that Machine Learning is a technique of training machines to perform the activities a human brain can do, albeit bit faster and better than an average human-being. Today we have seen that the machines can beat human champions in games such as Chess, Mahjong, which are considered very complex. We have seen that machines can be trained to perform human activities in several areas and can aid humans in living better lives. Machine learning is quickly growing field in computer science. It has applications in nearly every other field of study and is already being implemented commercially because machine learning can solve problems too difficult or time consuming for humans to solve. To describe machine learning in general terms, a variety models are used to learn patterns in data and make accurate predictions based on the patterns it observes.

Machine Learning can be a Supervised or Unsupervised. If we have a lesser amount of data and clearly labelled data for training, we opt for Supervised Learning. Unsupervised Learning would generally give better performance and results for large data sets. If we have a huge data set easily available, we go for deep learning techniques. We also have learned Reinforcement Learning and Deep Reinforcement Learning. We now know what Neural Networks are, their applications and limitations. Specifically, we have developed a thought process for approaching problems that machine learning works so well at solving. We have learnt how machine learning is different than descriptive statistics.

Finally, when it comes to the development of machine learning models of our own, we looked at the choices of various development languages, IDEs and Platforms. Next thing that we need to do is start learning and practicing each machine learning technique. The subject is vast, it means that there is width, but if we consider the depth, each topic can be learned in a few hours. Each topic is independent of each other. We need to take into consideration one topic at a time, learn it, practice it and implement the algorithm/s in it using a language choice of yours. This is the best way to start studying Machine Learning. Practicing one topic at a time, very soon we can acquire the width that is eventually required of a Machine Learning expert.

Overview-

A dataset related to the weather data for that day in major cities in Australia. This project presents a set of experiments that involve the use of common machine learning techniques to create models that can predict whether it will rain tomorrow or not . **Dataset Description-**

The data set consists of anonymous information such as Date, Location, MinTemp, MaxTemp, Rainfall, Evaporation, Sunshine, WindGustDir, WindGustSpeed, WindDir9am, Humidity, Pressure at Different Time periods, Cloud at Different time Periods, Rainfall On That Day. Each row is labelled as either if Rain Tomorrow or not for training the model

Dataset Source: https://www.kaggle.com/datasets/gauravduttakiit/weather-inaus

aus

Result-

MY project successfully Predicts whether it will rain tomorrow or not based on the weather data for that day in major cities in Australia with 85 % Accuracy

Project Code And Output

Importing Libraries :

```
import pandas as pd import
numpy as np
import matplotlib.pyplot as plt %matplotlib
inline
# Ignore harmless warnings  import warnings
warnings.filterwarnings("ignore") # Set to
display all the columns in dataset
pd.set_option("display.max_columns", None)
Import psql to run queries
 import pandasql as psql
```

Loading Weather Dataset

```
weather=pd.read_csv(r"C:\Users\Dlc\Desktop\datasets\weatherAUS.csv",       header=0)
weather_bk = weather.copy()
```

weather.sample(10)

Output

Out[2]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	Winc
59950	2009-08-03	Sale	7.6	13.7	5.2	6.0	7.4	W	100.0	W	W	24.0	
127694	2011-05-18	Hobart	9.6	14.8	0.0	0.8	0.2	NNW	37.0	WNW	NNW	7.0	
108474	2016-01-18	Albany	18.8	21.0	15.0	6.2	0.0	NaN	NaN	SE	NaN	9.0	
42640	2012-07-30	Wollongong	8.7	14.6	0.0	NaN	NaN	S	59.0	SW	S	20.0	
8811	2017-03-02	Cobar	21.2	36.2	0.0	NaN	NaN	ESE	35.0	E	SSE	17.0	
82746	2011-06-28	Brisbane	12.0	22.4	0.0	2.2	8.8	E	30.0	SSW	SE	6.0	
25021	2013-04-04	Penrith	15.2	20.3	2.8	NaN	NaN	SSW	26.0	SSW	SSW	9.0	
80841	2015-02-22	Dartmoor	17.9	39.9	0.0	10.4	7.2	WNW	46.0	WSW	WNW	2.0	
19297	2013-10-12	NorahHead	14.8	25.4	0.0	NaN	NaN	NE	44.0	S	NE	2.0	
140115	2015-12-04	Katherine	25.5	37.0	0.0	5.8	NaN	E	48.0	WNW	NW	9.0	

Checking Dataset Information

weather.info()

Output

```
<class 'pandas.core.frame.DataFrame'> RangeIndex:
142193 entries, 0 to 142192
Data columns (total 24 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
0  Date           142193 non-null  object
1  Location       142193 non-null  object
2  MinTemp        141556 non-null  float64
3  MaxTemp        141871 non-null  float64
4  Rainfall       140787 non-null  float64
5  Evaporation    81350 non-null   float64
6  Sunshine       74377 non-null   float64
7  WindGustDir    132863 non-null  object
8  WindGustSpeed  132923 non-null  float64
9  WindDir9am     132180 non-null  object
10 WindDir3pm     138415 non-null  object
11 WindSpeed9am   140845 non-null  float64
12 WindSpeed3pm   139563 non-null  float64
13 Humidity9am    140419 non-null  float64
14 Humidity3pm    138583 non-null  float64
15 Pressure9am    128179 non-null  float64
16 Pressure3pm    128212 non-null  float64
17 Cloud9am       88536 non-null   float64
18 Cloud3pm       85099 non-null   float64
19 Temp9am        141289 non-null  float64
```

```
20 Temp3pm         139467 non-null  float64
21 RainToday       140787 non-null  object
22 RISK_MM         142193 non-null  float64  23
RainTomorrow   142193 non-null  object  dtypes:
float64(17), object(7)
memory usage: 26.0+ MB
```

DATA PREPROCESSING

checking Duplicate Values in the Dataset

weather.isnull().sum()

Output

False

Checking Any Missing Values In the Dataset

weather.isnull().sum()

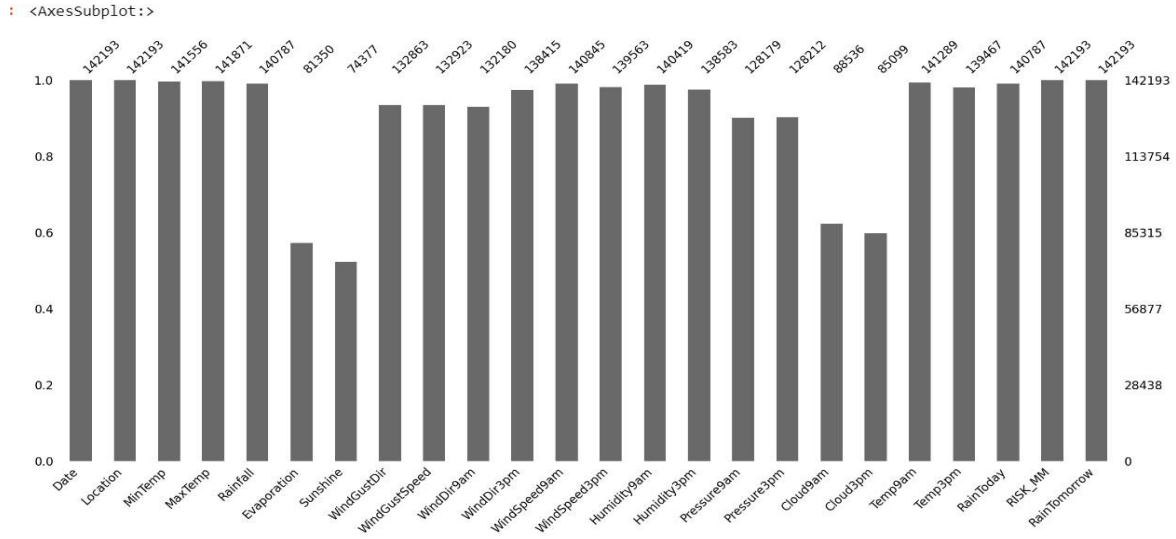
output

```
Date                0
Location            0
MinTemp           637
MaxTemp           322
Rainfall         1406
Evaporation     60843
Sunshine        67816
WindGustDir      9330
WindGustSpeed    9270
WindDir9am      10013
WindDir3pm       3778
WindSpeed9am     1348
WindSpeed3pm     2630
Humidity9am      1774
Humidity3pm      3610
Pressure9am     14014
Pressure3pm     13981
Cloud9am        53657
Cloud3pm        57094
Temp9am           904
Temp3pm          2726
RainToday        1406
RISK_MM             0 RainTomorrow
0
dtype: int64
```

Representing The Missing Values in Graphical Format

import missingno as msno msno.bar(weather)

Output



Filling The Missing Values Using KNNImputer(Numerical Values)

from sklearn.impute import KNNImputer

create an object for KNNImputer imputer_knn = KNNImputer(missing_values=np.nan)

#fill the missing values for 'MinTemp'
weather['MinTemp']=imputer_knn.fit_transform(weather[[‘MinTemp']])
weather['MinTemp'] = weather['MinTemp'].astype(int)

#fill the missing values for 'MaxTemp'
weather['MaxTemp']=imputer_knn.fit_transform(weather[[‘MaxTemp']])
weather['MaxTemp'] = weather['MaxTemp'].astype(int)

#fill the missing values for 'Rainfall'
weather['Rainfall']=imputer_knn.fit_transform(weather[['Rainfall']]) weather['Rainfall'] = weather['Rainfall']. astype (int)

#fill the missing values for 'Evaporation'
weather['Evaporation']=imputer_knn.fit_transform(weather[['Evaporation']])
weather['Evaporation'] = weather['Evaporation']. astype(int)

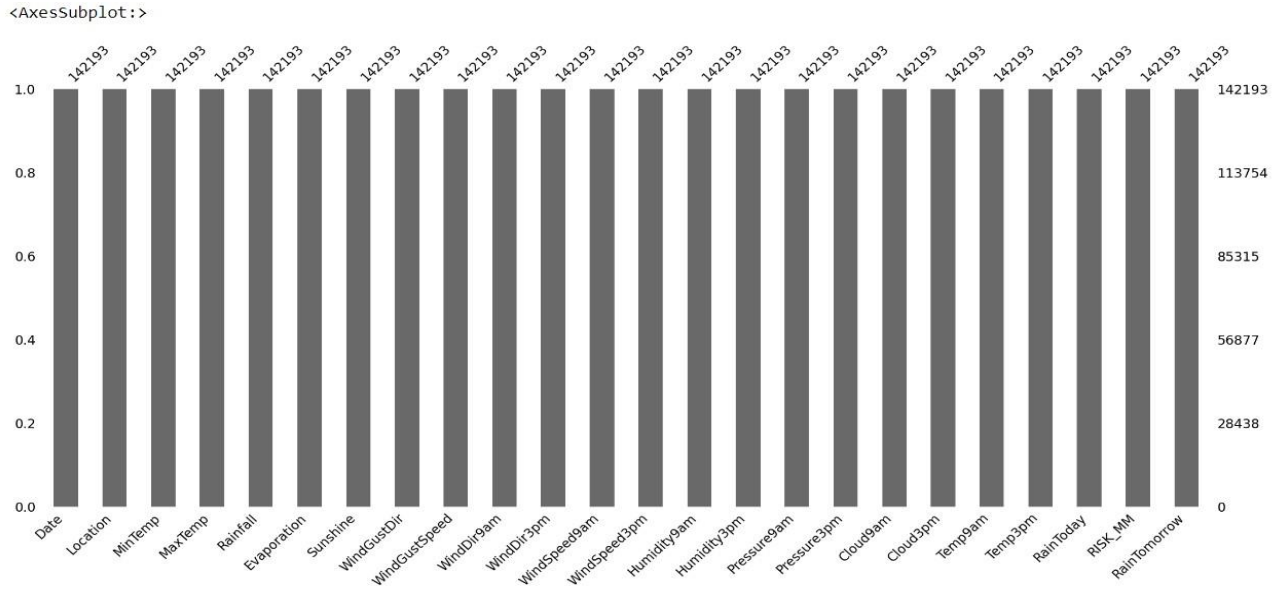
```
#fill the missing values for 'Sunshine'
weather['Sunshine']=imputer_knn.fit_transform(weather[['Sunshine']])
weather['Sunshine'] = weather['Sunshine'].astype(int)

#fill the missing values for 'WindGustSpeed'
weather['WindGustSpeed']=imputer_knn.fit_transform(weather[['WindGustSpeed']])
weather['WindGustSpeed'] = weather['WindGustSpeed'].astype(int)

#fill the missing values for 'WindSpeed9am'
weather['WindSpeed9am']=imputer_knn.fit_transform(weather[['WindSpeed9am']])
weather['WindSpeed9am'] = weather['WindSpeed9am'].astype(int)

#fill the missing values for 'WindSpeed3pm'
weather['WindSpeed3pm']=imputer_knn.fit_transform(weather[['WindSpeed3pm']])
weather['WindSpeed3pm'] = weather['WindSpeed3pm'].astype(int)

#fill the missing values for 'Humidity9am'
weather['Humidity9am']=imputer_knn.fit_transform(weather[['Humidity9am']])
weather['Humidity9am'] = weather['Humidity9am'].astype(int)
#fill the missing values for 'Humidity3pm'
weather['Humidity3pm']=imputer_knn.fit_transform(weather[['Humidity3pm']])
weather['Humidity3pm'] = weather['Humidity3pm'].astype(int)


#fill the missing values for 'Pressure9am'
weather['Pressure9am']=imputer_knn.fit_transform(weather[['Pressure9am']])
weather['Pressure9am'] = weather['Pressure9am'].astype(int)


#fill the missing values for 'Pressure3pm'
weather['Pressure3pm']=imputer_knn.fit_transform(weather[['Pressure3pm']])
weather['Pressure3pm'] = weather['Pressure3pm'].astype(int)


#fill the missing values for 'Cloud9am'
weather['Cloud9am']=imputer_knn.fit_transform(weather[['Cloud9am']])
weather['Cloud9am'] = weather['Cloud9am'].astype(int)
```

```
#fill the missing values for 'Cloud3pm'
weather['Cloud3pm']=imputer_knn.fit_transform(weather[['Cloud3pm']])
weather['Cloud3pm'] = weather['Cloud3pm'].astype(int)


#fill the missing values for 'Temp9am' weather['Temp9am'] =
imputer_knn.fit_transform(weather[['Temp9am']]) weather['Temp9am'] =
weather['Temp9am'].astype(int)
#fill the missing values for 'Temp3pm' weather['Temp3pm'] =
imputer_knn.fit_transform(weather[['Temp3pm']]) weather['Temp3pm'] =
weather['Temp3pm'].astype(int)


#fill the missing values for 'Temp3pm' weather['Temp3pm'] =
imputer_knn.fit_transform(weather[['Temp3pm']]) weather['Temp3pm'] =
weather['Temp3pm'].astype(int)
```

Filling The Missing Values Using simpleImputer(Catagorical Values)

```
# simpleImputer Techinue from
sklearn.impute import SimpleImputer
#simpleImputer(missing_values=nan,strategy='mean',fill_value=None,verbose
='deprecated',copy=True,add_indicator=False,keep_empty_features=False)
#create an object for simpleImputer imputer_si =
SimpleImputer(missing_values=np.nan,strategy='most_frequent')
weather['WindGustDir']=imputer_si.fit_transform(weather[['WindGustDir']])
weather['WindDir9am']=imputer_si.fit_transform(weather[['WindDir9am']])
weather['WindDir3pm']=imputer_si.fit_transform(weather[['WindDir3pm']])
weather['RainToday']=imputer_si.fit_transform(weather[['RainToday']])
```

Graphical Representation After Filling Missing Values



Identifying Unique Values For Each Variable in The Dataset

weather.nunique() **Output**

```
Date             3436
Location           49
MinTemp            41
MaxTemp            53
Rainfall          179
Evaporation        73
Sunshine           15
WindGustDir        16
WindGustSpeed      67
WindDir9am         16
WindDir3pm         16
WindSpeed9am       44
WindSpeed3pm       45
Humidity9am       101
Humidity3pm       101
Pressure9am        61
Pressure3pm        63
Cloud9am           10
Cloud3pm           10
Temp9am            48
Temp3pm            52
RainToday           2
RISK_MM           681
RainTomorrow        2 dtype:
int64
```

Counting Unique Values for Catagorical Variables inorder to label Encoding Process

```
weather['Location'].value_counts()
weather['WindGustDir'].value_counts()
weather['WindDir9am'].value_counts()
weather['WindDir3pm'].value_counts()
```

Label Encoding

```
# use Label encoder for Catagorical  variables
from sklearn.preprocessing import LabelEncoder
LE = LabelEncoder() weather['RainTomorrow']
=LE.fit_transform(weather['RainTomorrow'])weather['RainToday'] =
LE.fit_transform(weather['RainToday']) weather['Location'] =
LE.fit_transform(weather['Location']) weather['WindGustDir'] =
LE.fit_transform(weather['WindGustDir']) weather['WindDir9am'] =
LE.fit_transform(weather['WindDir9am']) weather['WindDir3pm'] =
LE.fit_transform(weather['WindDir3pm'])
```

Deleting unwanted Variables which does not impact on Target Variable

```
del weather['Date'] del
weather['RISK_MM']
```

Identifying The Proportion of Target Variable in Dataset

```
# Count the target or dependent variable by '0' & '1' and their proportion
# (>= 10 : 1, then the dataset is imbalance data)
RainTomorrow_count = weather.RainTomorrow.value_counts() print('Class 0:', RainTomorrow_count[0]) print('Class 1:', RainTomorrow_count[1]) print('Proportion:', round(RainTomorrow_count[0] / RainTomorrow_count[1], 2), ': 1') print('Total weatherAus  records:', len(weather))
```

Output

Class 0: 110316
Class 1: 31877
Proportion: 3.46 : 1
Total weatherAus records: 142193

Identify the independent and target (dependent) variables

# Identify the independent and targrt (dependent) variables
IndepVar = [] for col in weather.columns: if col != 'RainTomorrow': IndepVar.append(col) TargetVar ='RainTomorrow' x = weather[IndepVar] y = weather[TargetVar]

Split the data into train and test (random sampling)

Split the data into train and test (random sampling) from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
Display the shape for train & test data
x_train.shape, x_test.shape, y_train.shape, y_test.shape

Output

```
((99535, 21), (42658, 21), (99535,), (42658,))
```

Scaling the features by using MinMaxScaler

Scaling the features by using MinMaxScaler

from sklearn.preprocessing import MinMaxScaler

mmscaler = MinMaxScaler(feature_range=(0, 1))

x_train = mmscaler.fit_transform(x_train) x_train

= pd.DataFrame(x_train) x_test =
mmscaler.fit_transform(x_test) x_test =
pd.DataFrame(x_test)

Comparing Algorithms

KNN Algorithm

```
# Load the results dataset for KNN
KNN_Results = pd.read_csv(r"C:\Users\Dlc\Desktop\datasets\KNN_Results.csv", header=0)

# Bild KNN Model from sklearn.neighbors import KNeighborsClassifier from
sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import sklearn.metrics as metrics from sklearn.metrics import roc_curve,
roc_auc_score accuracy = [] for a in range(1, 21, 1):
    k = a
# Build the model
ModelKNN = KNeighborsClassifier(n_neighbors=k)
# Train the model
ModelKNN.fit(x_train, y_train)
# Predict the model y_pred =
ModelKNN.predict(x_test) y_pred_prob =
ModelKNN.predict_proba(x_test)
print('KNN_K_value = ', a) # Print the model
name print('Model Name: ', ModelKNN) #
confusion matrix in sklearn from sklearn.metrics
import confusion_matrix from sklearn.metrics
import classification_report
# actual values
actual = y_test  #
predicted values
predicted = y_pred
```

```
# confusion matrix
matrix =
confusion_matrix(
actual,predicted,
labels=[1,0],sampl
e_weight=None,
normalize=None)
 print('Confusion matrix : \n', matrix) # outcome values order in sklearn tp,
fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3); accuracy =
round((tp+tn)/(tp+fp+tn+fn), 3); balanced_accuracy =
round((sensitivity+specificity)/2, 3); precision =
round(tp/(tp+fp), 3); f1Score = round((2*tp/(2*tp + fp +
fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn) MCC = round(((tp * tn) - (fp *
fn)) / sqrt(mx), 3) print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%') print('Recall :',
round(sensitivity*100,2), '%') print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC) # Area under ROC curve  sklearn.metrics import
roc_curve, roc_auc_score print('roc_auc_score:',
round(roc_auc_score(actual, predicted), 3))
# ROC Curve from sklearn.metrics import
roc_auc_score from sklearn.metrics import
roc_curve model_roc_auc =
```

```
roc_auc_score(actual, predicted) fpr, tpr,
thresholds = roc_curve(actual,
ModelKNN.predict_proba(x_test)[:,1])
plt.figure()
   # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % model_roc_auc)     plt.plot([0, 1],
[0, 1],'r--')     plt.xlim([0.0, 1.0])     plt.ylim([0.0, 1.05])     plt.xlabel('False
Positive Rate')     plt.ylabel('True Positive Rate')     plt.title('Receiver operating
characteristic')     plt.legend(loc="lower right")     #plt.savefig('Log_ROC')
plt.show()
   #-------------------------------------------------------------------------
   new_row = {'Model Name' : ModelKNN,
            'KNN K Value' : a,
            'True_Positive' : tp,
            'False_Negative' : fn,
            'False_Positive' : fp,
            'True_Negative' : tn,
            'Accuracy' : accuracy,
            'Precision' : precision,
            'Recall' : sensitivity,
            'F1 Score' : f1Score,
            'Specificity' : specificity,
            'MCC':MCC,
            'ROC_AUC_Score':roc_auc_score(actual, predicted),
            'Balanced Accuracy':balanced_accuracy}
   KNN_Results = KNN_Results.append(new_row, ignore_index=True)
   #------KNN_Results-----------------------------------------------------------------
```

SVM Algorithm

```
#Comparing the all SVM Models

x_train_reduced = x_train.sample(10000)

y_train_reduced = y_train.sample(10000) from

sklearn.svm import SVC

ModelSVM1 = SVC(C=1.0, kernel='linear', degree=3, gamma='scale', coef0=0.0, shrinking=True,
probability=True, tol=0.001, cache_size=200, class_weight=None, verbose=False,
max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None)
ModelSVMPoly = SVC(kernel='poly', degree=2, probability=True)

ModelSVMSig = SVC(kernel='sigmoid', random_state = 42, class_weight='balanced',
probability=True)

ModelSVMGaussian = SVC(kernel='rbf', random_state = 42, class_weight='balanced',
probability=True)

models=[ModelSVM1,ModelSVMPoly,ModelSVMSig,ModelSVMGaussian]

for i in models:

   i.fit(x_train_reduced, y_train_reduced)

# Prediction   y_pred = i.predict(x_test)

y_pred_prob = i.predict_proba(x_test)

 # Print the model name   print('Model Name: ',

i)   # confusion matrix in sklearn from

sklearn.metrics import confusion_matrix from

sklearn.metrics import classification_report

# actual values actual = y_test # predicted values predicted = y_pred # confusion matrix matrix =

confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)

print('Confusion matrix : \n', matrix)     # outcome values order in sklearn     tp, fn, fp, tn =

confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)     print('Outcome values : \n', tp, fn,

fp, tn)

   # classification report for precision, recall f1-score and accuracy

C_Report = classification_report(actual,predicted,labels=[1,0])

print('Classification report : \n', C_Report)
```

```
   # calculating the metrics     sensitivity = round(tp/(tp+fn),
3);     specificity = round(tn/(tn+fp), 3);     accuracy =
round((tp+tn)/(tp+fp+tn+fn), 3);     balanced_accuracy =
round((sensitivity+specificity)/2, 3);     precision =
round(tp/(tp+fp), 3);     f1Score = round((2*tp/(2*tp + fp +
fn)), 3);
  # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
   # A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt    mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)     MCC =
round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)     print('Accuracy :',
round(accuracy*100, 2),'%')     print('Precision :', round(precision*100,
2),'%')     print('Recall :', round(sensitivity*100,2), '%')     print('F1 Score :',
f1Score)     print('Specificity or True Negative Rate :',
round(specificity*100,2), '%'  )     print('Balanced Accuracy :',
round(balanced_accuracy*100, 2),'%')     print('MCC :', MCC)
   print('----------------------------------------------------------------------------------------------------')
  new_row = {'Model Name' : i,
            'True_Positive' : tp,
             'False_Negative' : fn,
             'False_Positive' : fp,
             'True_Negative' : tn,
             'Accuracy' : accuracy,
             'Precision' : precision,
             'Recall' : sensitivity,
             'F1 Score' : f1Score,
             'Specificity' : specificity,
             'MCC':MCC,
             'ROC_AUC_Score':roc_auc_score(actual, predicted),
             'Balanced Accuracy':balanced_accuracy}
```

EMResults1 = EMResults1.append(new_row, ignore_index=True)

EMResults1.head()

Output

Out[44]:

	Model Name	True_Positive	False_Negative	False_Positive	True_Negative	Accuracy	Precision	Recall	F1 Score	Specificity	MCC	ROC_AUC_S
0	SVC(kernel='linear', probability=True)	0	9525	0	33133	0.777	NaN	0.000	0.000	1.000	NaN	0.50
1	SVC(degree=2, kernel='poly', probability=True)	0	9525	0	33133	0.777	NaN	0.000	0.000	1.000	NaN	0.50
2	SVC(class_weight='balanced', kernel='sigmoid',...	5800	3725	16399	16734	0.528	0.261	0.609	0.366	0.505	0.095	0.55
3	SVC(class_weight='balanced', probability=True,...	5564	3961	18154	14979	0.482	0.235	0.584	0.335	0.452	0.030	0.51

Naive_bayes Algorithm

Training the Naive Bayes model (GaussianNB) on the Training set

from sklearn.naive_bayes import GaussianNB modelGNB =

GaussianNB(priors=None, var_smoothing=1e-09)

Fit the model with train data

modelGNB.fit(x_train,y_train) # Predict the

model with test data set y_pred =

modelGNB.predict(x_test) y_pred_prob =

modelGNB.predict_proba(x_test)

Confusion matrix in sklearn from

sklearn.metrics import confusion_matrix from

sklearn.metrics import classification_report

```
# actual values actual = y_test # predicted values predicted = y_pred #
confusion matrix matrix = confusion_matrix(actual,predicted,
labels=[1,0],sample_weight=None, normalize=None) print('Confusion matrix :
\n', matrix) # outcome values order in sklearn tp, fn, fp, tn =
confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1) print('Outcome
values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy C_Report
= classification_report(actual,predicted,labels=[1,0]) print('Classification
report : \n', C_Report)
# calculating the metrics sensitivity
= round(tp/(tp+fn), 3); specificity =
round(tn/(tn+fp), 3); accuracy =
round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy =
round((sensitivity+specificity)/2,
3); precision = round(tp/(tp+fp),
3); f1Score = round((2*tp/(2*tp +
fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1
to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model from
math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn) MCC = round(((tp * tn) - (fp *
fn)) / sqrt(mx), 3) print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%') print('Recall :',
round(sensitivity*100,2), '%') print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%' )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC) # Area under ROC curve  from sklearn.metrics
```

```
import roc_curve, roc_auc_score print('roc_auc_score:',
round(roc_auc_score(actual, predicted), 3))
# ROC Curve from sklearn.metrics import roc_auc_score from
sklearn.metrics import roc_curve model_roc_auc = roc_auc_score(actual,
predicted) fpr, tpr, thresholds =
roc_curve(actual,modelGNB.predict_proba(x_test)[:,1]) plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc) plt.plot(fpr,
tpr, label= 'Classification Model' % model_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
```

comparing All Classification Algorithm

```
# Build the Calssification models and compare the results
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier from
sklearn.ensemble import RandomForestClassifier from
sklearn.ensemble import ExtraTreesClassifier from
sklearn.neighbors import KNeighborsClassifier
#from sklearn.svm import SVC
# Create objects of classification algorithm with default hyper-parameters
ModelLR = LogisticRegression()
ModelDC = DecisionTreeClassifier()
ModelRF = RandomForestClassifier()
ModelET = ExtraTreesClassifier()
ModelKNN = KNeighborsClassifier(n_neighbors=19)
#ModelSVM = SVC(kernel='linear', random_state = 42, class_weight='balanced',
probability=True)
# Evalution matrix for all the algorithms
```

```
MM = [ModelLR, ModelDC, ModelRF, ModelET, ModelKNN]
for models in MM:    # Fit the model    models.fit(x_train,
y_train)
  # Prediction    y_pred =
models.predict(x_test)    y_pred_prob =
models.predict_proba(x_test)
  # Print the model name    print('Model Name:
', models)   # confusion matrix in sklearn    from
sklearn.metrics import confusion_matrix
from sklearn.metrics import
classification_report
  # actual values    actual = y_test    # predicted values    predicted = y_pred    #
confusion matrix    matrix = confusion_matrix(actual,predicted,
labels=[1,0],sample_weight=None,                normalize=None)
 print('Confusion matrix : \n', matrix)  # outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
 # classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
 # calculating the metrics  sensitivity = round(tp/(tp+fn),
3);  specificity = round(tn/(tn+fp), 3);  accuracy =
round((tp+tn)/(tp+fp+tn+fn), 3);  balanced_accuracy =
round((sensitivity+specificity)/2, 3);  precision =
round(tp/(tp+fp), 3);  f1Score = round((2*tp/(2*tp + fp +
fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1
to +1.
```

```
 # A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn) MCC =
round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)     print('Specificity or
True Negative Rate :', round(specificity*100,2),
'%'  )     print('Balanced Accuracy :',
round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)     # Area under ROC curve
from sklearn.metrics import roc_curve,
roc_auc_score     print('roc_auc_score:',
round(roc_auc_score(actual, predicted), 3))
   # ROC Curve     from sklearn.metrics import roc_auc_score     from
sklearn.metrics import roc_curve     model_roc_auc = roc_auc_score(actual,
predicted)     fpr, tpr, thresholds = roc_curve(actual,
models.predict_proba(x_test)[:,1])     plt.figure()
   # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % model_roc_auc)     plt.plot([0,
1], [0, 1],'r--')     plt.xlim([0.0, 1.0])     plt.ylim([0.0, 1.05])     plt.xlabel('False
Positive Rate')     plt.ylabel('True Positive Rate')     plt.title('Receiver operating
characteristic')     plt.legend(loc="lower right")     plt.savefig('Log_ROC')
plt.show()     new_row = {'Model Name' : models,
            'True_Positive' : tp,
            'False_Negative' : fn,
            'False_Positive' : fp,
            'True_Negative' : tn,
```

```
            'Accuracy' : accuracy,
            'Precision' : precision,
            'Recall' : sensitivity,
            'F1 Score' : f1Score,
            'Specificity' : specificity,
            'MCC':MCC,
            'ROC_AUC_Score':roc_auc_score(actual, predicted),
            'Balanced Accuracy':balanced_accuracy}
    EMResults = EMResults.append(new_row, ignore_index=True)
EMResults
```

Output **Fig : Result Of All Algorithms**

	Model Name	True_Positive	False_Negative	False_Positive	True_Negative	Accuracy	Precision	Recall	F1 Score	Specificity
0	LogisticRegression()	4681	4844	1916	31217	0.842	0.710	0.491	0.581	0.942
1	DecisionTreeClassifier()	5116	4409	5945	27188	0.757	0.463	0.537	0.497	0.821
2	(DecisionTreeClassifier(max_features='auto', r...	4850	4675	1655	31478	0.852	0.746	0.509	0.605	0.950
3	(ExtraTreeClassifier(random_state=554780901), ...	4677	4848	1578	31555	0.849	0.748	0.491	0.593	0.952
4	KNeighborsClassifier(n_neighbors=18)	4042	5483	1450	31683	0.837	0.736	0.424	0.538	0.956

Predict the values with Random Forest Algorithm(High Accuracy)

```
y_pred = ModelRF.predict(x_test)
```

Display the Final result

```
#create new dataframe with actual vs prdict values
#Display the Final result
Results = pd.DataFrame({'RainTomorrow_A':y_test,'RainTomorrow_P':y_pred})
#Merge two Dataframes on index of both the dataframes
ResultsFinal = weather_bk.merge(Results,left_index=True,right_index=True)
# 'map' function to convert the numerical values to Catagorical format
ResultsFinal['RainTomorrow_A'] = ResultsFinal['RainTomorrow_A'].map({0:'No',1:'Yes'})
ResultsFinal['RainTomorrow_P'] = ResultsFinal['RainTomorrow_P'].map({0:'No',1:'Yes'})
# Same as str.replace command
# Display 10 records randomly
ResultsFinal.sample(10)
```

Output

Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RISK_MM	RainTomorrow	RainTomorrow_A	RainTomorrow_P
79.0	1009.9	1008.9	NaN	NaN	14.6	16.4	Yes	7.6	Yes	Yes	Yes
75.0	1007.3	1005.8	8.0	8.0	18.3	18.9	Yes	5.2	Yes	Yes	Yes
39.0	1024.4	1021.2	0.0	0.0	13.2	19.0	No	0.0	No	No	No
68.0	1012.4	1014.2	2.0	6.0	23.0	21.5	No	0.0	No	No	No
32.0	1007.7	1003.9	7.0	7.0	21.0	29.1	No	0.0	No	No	Yes
47.0	1018.4	1017.0	0.0	3.0	12.5	18.0	No	0.0	No	No	No
29.0	1017.3	1014.2	3.0	3.0	20.9	32.1	No	0.0	No	No	No
52.0	NaN	NaN	1.0	1.0	24.4	28.5	No	0.0	No	No	No
58.0	1008.3	1013.1	NaN	NaN	5.2	9.2	Yes	0.6	No	No	No
19.0	1017.5	1013.2	NaN	NaN	24.7	31.4	No	0.0	No	No	No

Chapter 5 Advantages & Disadvantages

Advantages of Machine Learning

Every coin has two faces, each face has its own property and features. It's time to uncover the faces of ML. A very powerful tool that holds the potential to revolutionize the way things work.

1. **Easily identifies trends and patterns -**

   Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an ecommerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. **No human intervention needed (automation) -** With ML, we don't need to babysit our project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software. they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. **Continuous Improvement -**

   As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say we need to make a weather forecast model. As the amount of data, we have keeps growing, our algorithms learn to make more accurate predictions faster.

4. **Handling multi-dimensional and multi-variety data -**

   Machine Learning algorithms are good at handling data that are multidimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. **Wide Applications -**

   We could be an e-seller or a healthcare provider and make ML work for us. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning

With all those advantages to its powerfulness and popularity, Machine Learning isn't perfect. The following factors serve to limit it:

1. **Data Acquisition -**

   Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources -

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for us.

3. Interpretation of Results -

Another major challenge is the ability to accurately interpret results generated by the algorithms. We must also carefully choose the algorithms for your purpose.

4. High error-susceptibility -

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Chapter 6 - Conclusion

So far, the research offered some important insights into the quality of customer service and other factors affecting passenger satisfaction and experience. Among the key factors affecting the customers' negative experiences were the poor quality of food, low seating comfort, and arrival or departure delays.

Among the top factors affecting positive customers' experiences were the Type of Travel , Class, Inflight wifi service, Online boarding, Seat comfort, Inflight entertainment .However, the top three factors linked to negative customers' experiences were different. Poor quality of food was most often mentioned in negative reviews, followed by seating comfort and cabin crew work. Thus, one major finding of this research is that an improvement in factors affecting

negative customer experience might not always contribute to positive feedback. For instance, improving in-flight entertainment might not help to achieve better satisfaction scores if customers' experiences are negatively affected by more important factors, such as food quality.

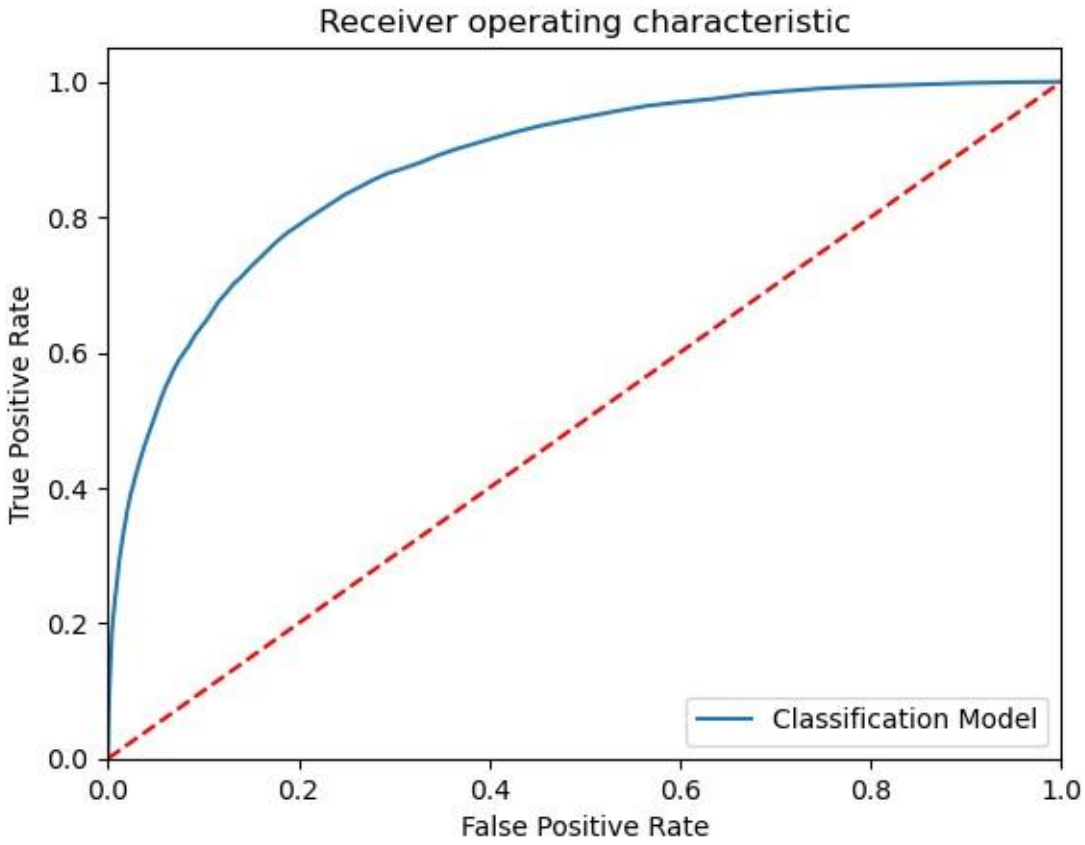
This research has significant implications for practice, as it allows airlines to focus on the specific factors affecting customer satisfaction. Moreover, it helps the airlines to prioritise the aspects of service correctly to achieve best results in customer experience. For instance, quality of entertainment might not be as significant to the majority of customers as food quality and in-flight crew performance, so airlines could focus on improving the aspects of service that have a more significant impact on customer satisfaction. This could allow carriers to reduce the budget spent on enhancing service while at the same time providing a better customer experience.

The model results in the following order by considering the model accuracy, F1 score and RoC AUC score.

1) KNN algorithm
2) SVM algorithm
3) Logistic Regression

We recommend model – **Random Forest ALgorithm** as a best fit for the given dataset. We considered Random Forest because it uses bootstrap aggregation which can reduce bias and variance in the data and can leads to good predictions with Weather datasets.

Fig:Roc curve of the Random Forets algorithm



Plotting The Result

```
Out[57]: <AxesSubplot:xlabel='RainTomorrow', ylabel='count'>
```