

PREDICTION OF FERTILIZERS FOR THE EFFICIENT YIELD THROUGH MACHINE LEARNING

*A Project report submitted in partial fulfillment of the requirements for the award of
the degree of*

**BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING**

Submitted by

P. PRUDHVIRAJU (317126512047)

N. SUMITRANJALI(317126512041)

S. S MANOJ (317126512050)

G. DAVID ELIEZER(317126512020)

**Under the guidance
Of
Mr. N. RAM KUMAR
Assistant Professor**



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)**

*(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade)
Sangivalasa, bheemili mandal, visakhapatnam dist.(A.P)2020-2021*

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)**

*(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC
with 'A' Grade)*


Sangivalasa, Bheemili mandal, Visakhapatnam dist.(A.P)




ANITS

CERTIFICATE

This is to certify that the project report entitled "PREDICTION OF FERTILIZERS FOR THE EFFICIENT YIELD THROUGH MACHINE LEARNING" submitted by Prudhviraaju Polamarasetty(317126512047), Sumitranjali Nagothi (317126512041), Satyanarayana Manoj Samudrala (317126512050) ,David Eliezer Gorremuchu (317126512020) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering of Andhra University, Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.


**Project Guide
N.Ram Kumar
Assistant Professor
Department of E.C.E
ANITS**

**Assistant Professor
Department of E.C.E.
Anil Neerukonda**


**Head of the Department
Dr. V.Rajyalakshmi
Professor&HOD
Department of E.C.E
ANITS**

**Head of the Department
Department of E C E
Anil Neerukonda Institute of Technology & Sciences
Sangivalasa - 531 162**

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **N.Ram Kumar**, Assistant professor ,Department of Electronics and Communication Engineering, ANITS, for his/her guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Dr. V. Rajyalakshmi**, Head of the Department, Electronics and Communication Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS, Sangivalasa**, for their encouragement and cooperation to carry out this work.

We express our thanks to all **teaching faculty** of Department of ECE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank **all non-teaching staff** of the Department of ECE, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT STUDENTS

Prudhviraju Polamarasetty (317126512047),

Sumitranjali Nagothi (317126512041),

Satyanarayana Manoj Samudrala (317126512050),

David Eliezer Gorremuchu(317126512020)

ABSTRACT

Richness of soil assumes a vital part in gaining great yield from the harvest. Present day headways in innovation are end up being a help in catalyzing the harvest yield. A large portion of the ranchers have faith in legends in the public eye and develop without earlier information or appropriate examination of manures that are most appropriate for the given soil and harvest type. Soil testing, determining the best reasonable compost will expand the agrarian creation by improving the supplement content accessible in the dirt. Utilization of wrong manures definitely impacts the yield and soundness of the harvest. ML is an up-and-coming field of informatics that can be applied effectively to the horticultural area, so we proposed an ML model which break down the given informational collection with the distinctive ML models like Decision Tree, Random Forest, Gradient Boost, Ada Boost, Gaussian NB and anticipate the most appropriate fertilizer by choosing the best suitable model. ML methods helps in compost prediction consequently, assists the ranchers with improving the harvest yield.

CONTENTS

ABSTRACT	iv
LIST OF SYMBOLS	viii
LIST OF FIGURES	ix
LIST OF TABLES	xi
LIST OF ABBREVIATIONS	xii
CHAPTER 1 INTRODUCTION	01
1.1 Project Outline	01
1.2 Project Objective	02
CHAPTER 2 METHODOLOGY	03
2.1 Introduction to Machine Learning	04
2.1.1 Supervised Learning	05
2.1.2 Unsupervised Learning	05
2.1.3 Reinforcement Learning	05
2.2 Technologies Used	06
2.2.1 Python	06
2.2.2 Libraries	07
2.2.3 Pandas	07
2.2.4 Matplotlib	09
2.2.5 NumPy	10
2.2.6 Scikit Learn	11

2.2.7	Seaborn	12
2.2.8	Jupyter Notebook IDE	14
2.2.9	Jupyter Kernals	15
2.3	Algorithms Used	16
2.3.1	Decision Tree (DT)	16
2.3.2	Random Forest (RF)	18
2.3.3	Gaussian Naïve Bayes (NB)	19
2.3.3.1	Bayes' Therom	20
2.3.3.2	Naïve Bayes classifier	20
2.3.3.3	Gaussian Naïve Bayes	21
2.3.4	Boosting	22
2.3.4.1	Ada Boost (AB)	23
2.3.4.2	Gradient Boost (GB)	24
CHAPTER 3 DATA VISUALIZATION		25
3.1	Cleaning of Data set	25
3.2	What is Data Visualization?	28
3.2.1	Why visualization?	28
CHAPTER 4 RESULTS		38
4.1	Evaluation Parameters	38
4.1.1	Confusion Matrix	38
4.1.2	Accuracy	39
4.1.3	Precision	39
4.1.4	Recall	39
4.1.5	F1-Score	39
4.2	Gradient Boost	43
4.3	Decision Tree	44
4.4	Random Forest	45
4.5	Gaussian NB	46

4.6 Ada Boost	47
4.7 Comparison Table	48
CHAPTER 5 CONCLUSION AND FUTURE WORK	49
REFERENCES	50
APPENDIX	52

LIST OF SYMBOLS

Symbol	Description	Page no
H(s)	Entropy	16
P(x)	Probability of the attribute	17
IG (S, A)	Information Gain	17
TN	True Negative	38
TP	True Positive	38
FP	False Positive	38
FN	False Negative	38

LIST OF FIGURES

Figure no	Title	Page no
Fig. 2.1	Steps involved in prediction of fertilizers.	03
Fig. 2.2	Machine Learning Pipeline.	04
Fig. 2.3	A Trained Model of Supervised Learning.	06
Fig. 2.4	Seaborn Plots.	13
Fig. 2.5	Jupyter IDE	14
Fig. 2.6	Decision tree	17
Fig. 2.7	Random Forest Classifier	19
Fig. 2.8	Formulas for Bayes' Theorem	20
Fig. 2.9	Gaussian NB Classifier	22
Fig. 2.10	Steps in Ada Boost Classifier	23
Fig. 2.11	Steps in Gradient Boost Classifier	24
Fig. 3.1	Steps involved in Data Cleaning	25
Fig. 3.2	Observations of Null Values and Unique Values of Parameters	26
Fig. 3.3	Types of Crops in that are considered	27
Fig. 3.4	Different Fertilizers Used	27
Fig. 3.5	Types of Data Visualization Techniques	28
Fig. 3.6	Histogram of Humidity	30
Fig. 3.7	Histogram of Moisture	31
Fig. 3.8	Histogram of Nitrogen	32
Fig. 3.9	Histogram of Phosphorous	

Fig. 3.10	Histogram of Potassium	33
Fig. 3.11	Correlation Matrix	35
Fig. 3.12	Temperature vs Fertilizer	36
Fig. 3.13	Moisture vs Fertilizer	37
Fig. 3.14	Nitrogen vs Fertilizer	37
Fig. 3.15	Potassium vs fertilizer	37
Fig. 4.1	Confusion Matrix of Gaussian NB	40
Fig. 4.2	Confusion Matrix of Decision Tree	40
Fig. 4.3	Confusion Matrix of Random Forest	41
Fig. 4.4	Confusion Matrix of Gradient Boost	41
Fig. 4.5	Confusion Matrix of Ada Boost	42
Fig. 4.6	Gradient Boost Classifier	43
Fig. 4.7	Decision Tree Classifier	44
Fig. 4.8	Random Forest Classifier	45
Fig. 4.9	Gaussian NB Classifier	46
Fig. 4.10	Ada Boost Classifier	47

LIST OF TABLES

Table no	Title	Page no
Table 3.1	Observations made from Correlation Matrix	35
Table 4.1	Confusion Matrix Representation	38
Table 4.2	Results of Gradient Boost Classifier	43
Table 4.3	Results of Decision Tree Classifier	44
Table 4.4	Results of Random Forest Classifier	45
Table 4.5	Results of Gaussian NB Classifier	46
Table 4.6	Results of Ada Boost Classifier	47
Table 4.7	Comparison Table for all proposed algorithms	48

LIST OF ABBREVIATIONS

AB	Ada Boost
GB	Gradient Boost
DT	Decision Tree
RF	Random Forest
ML	Machine Learning
IoT	Internet of Things
N	Nitrogen
P	Phosphorus
K	Potassium

CHAPTER -1

INTRODUCTION

1.1 PROJECT OUTLINE

Agribusiness expects a critical part in monetary headway similarly as occupation for country like India. India is the second greatest maker of horticulture products. One of the major issues looked by the ranchers is picking the correct manure to the particular yield. There are numerous expensive composts available in the market, yet they fail to give great outcomes since ranchers do not think about their individual requirements which results in awful yield. The choice of compost relies upon numerous components consequently it cannot be universalized.

These days there is a great deal of advancement in modern innovation which assists with picking a superior compost, one of the arising fields is Machine learning. Machine Learning is a part of man-made reasoning (AI) zeroed in on building applications that gain from information and improve their precision after some time without being modified to do so. It has various calculations with the assistance of which we can make right predictions a portion of the algorithms are Random Forest, Decision Tree, Ada Boost, Gradient Boost and Gaussian NB. Machine Learning does complex calculations and an assortment of exact information to work keenly. It utilizes past information to peruse the examples and afterward play out the proposed task as indicated by the characterized rules and calculations dependent on the examination it delivered.

The central point influencing crop yield are the crop type, soil type the supplements like N, P, K present in the soil, temperature dampness and mugginess. Without considering these in the event that we purchase a manure it wouldn't give accurate results. This project focuses on comparing five different ML models and concludes the accurate model to predict the suitable fertilizer when provided with the macro nutrient (NPK) values obtained from soil test along with the temperature, humidity, crop type, soil type by the user.

1.2 PROJECT OBJECTIVE

Fertilizers are chemical substances supplied to the crops to increase their productivity. These are used by the farmers daily to increase the crop yield. The fertilizers contain the essential nutrients required by the plants, including nitrogen, potassium, and phosphorus. They enhance the water retention capacity of the soil and increase its fertility. Fertility of soil plays a crucial role in getting good yield from the crop. A fertile soil will contain all the major nutrients for basic plant nutrition like Nitrogen(N), Potassium(k), Phosphorous(P). This project helps the farmer to predict the suitable fertilizer for the given nutrient levels of the soil obtained from the soil test. Every combination of soil and plant are unique, and they require different form of nutrients. So, the type of fertilizer required for them also vary. Farmers may not know the exact requirement by the soil or plant until they get the result. Hence, farmer in one region may end up with good yield due to the right selection of fertilizer while farmer in a different region with same type of soil and plant yield improper result. This ensures that the fertilizer being used in the farm is based on any unclear predictions. The key is to get this balance right and to maintain a level of nutrients in soils that will support our crops. So, there is a need to establish a platform to suggest the right fertilizer for a given crop and soil type.

CHAPTER -2

METHODOLOGY

An informational collection is gathered which contains the data of temperature, mugginess, N, P, K values crop type, soil type and the most appropriate manure for the given conditions. further this informational index is checked for any invalid qualities or any copies whenever discovered they are taken out. Informational index is then changed over into reference diagrams, relationship networks. Charts are plotted against every single boundary so the framework can have a superior comprehension of the information. 75% of the information is feed to the machine for training it and remaining information is utilized to test it to see whether it's making the correct prediction, distinctive ML algorithms like Random Forest, Decision Tree, Ada Boost Gradient boost and Gaussian NB are utilized to do this. when each model does the forecast we ascertain the precision with assistance of confusion matrix, F-Score, precision and recall by contrasting these qualities we choose the best appropriate model to do the prediction. Once the best suited Model is chosen, the system is provided with the macro nutrient (N, P, K) values, temperature, humidity, crop type, soil type by the user and gives the best preferred fertilizer for the given conditions. Fig 2.1 shows the complete methodology of fertilizer prediction.

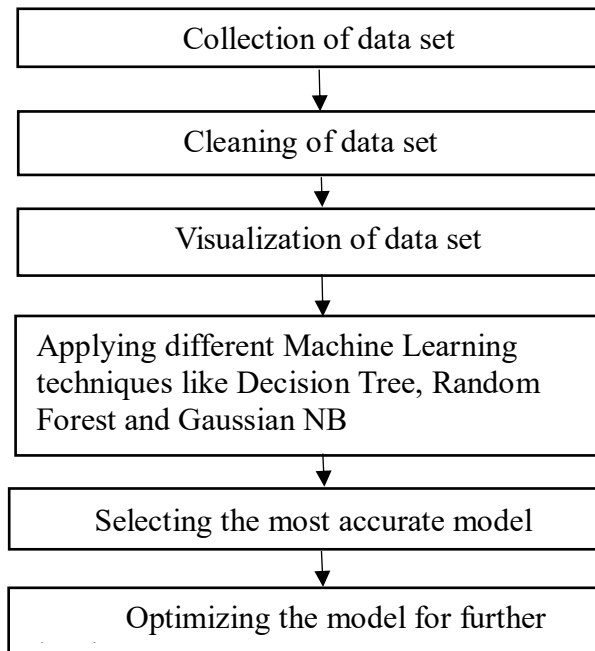


Fig 2.1: Steps involved in prediction of fertilizers.

2.1 INTRODUCTION TO MACHINE LEARNING

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

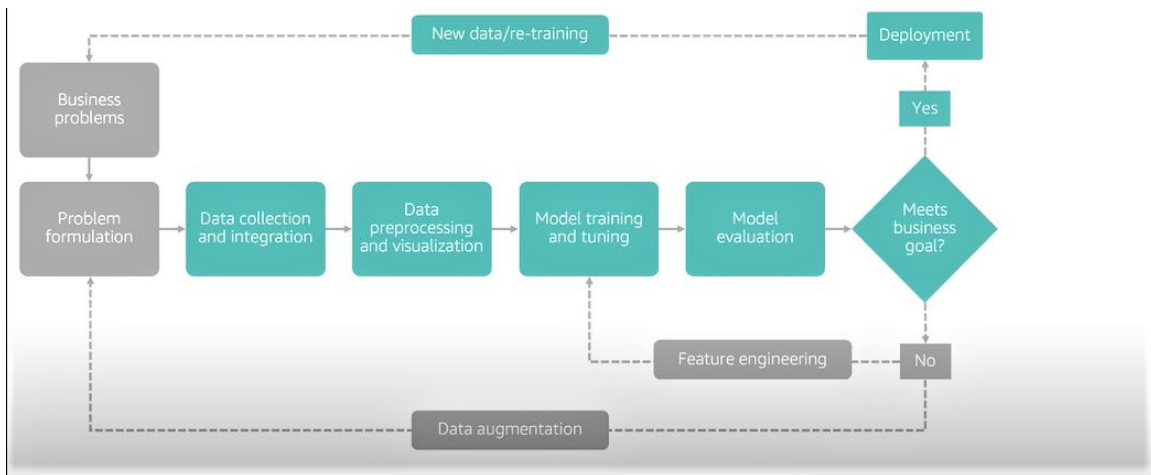


Fig 2.2: Machine Learning Pipeline

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed. Machine learning implementations are classified into three major categories as follows: -

2.1.1 Supervised Learning

In supervised learning, the computer is provided with example inputs that are labelled with their desired outputs. The purpose of this method is for the algorithm to be able to “learn” by comparing its actual output with the “taught” outputs to find errors and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabelled data.

2.1.2 Unsupervised Learning

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of un-correlated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

2.1.3 Reinforcement Learning

When you present the algorithm with examples that lack labels, as in unsupervised learning. However, you can accompany an example with positive or negative feedback according to the solution the algorithm proposes comes under the category of Reinforcement learning, which is connected to applications for which the algorithm must make decisions (so the product is prescriptive, not just descriptive, as in unsupervised learning), and the decisions bear consequences. In the human world, it is just like learning by trial and error. Errors help you learn because they have a penalty added (cost, loss of time, regret, pain, and so on), teaching you that a certain course of action is less likely to succeed than others. An interesting example of reinforcement learning occurs when computers learn to play video games by themselves. In this project we have used different supervised algorithms. Fig 2.3 shows the trained supervised model

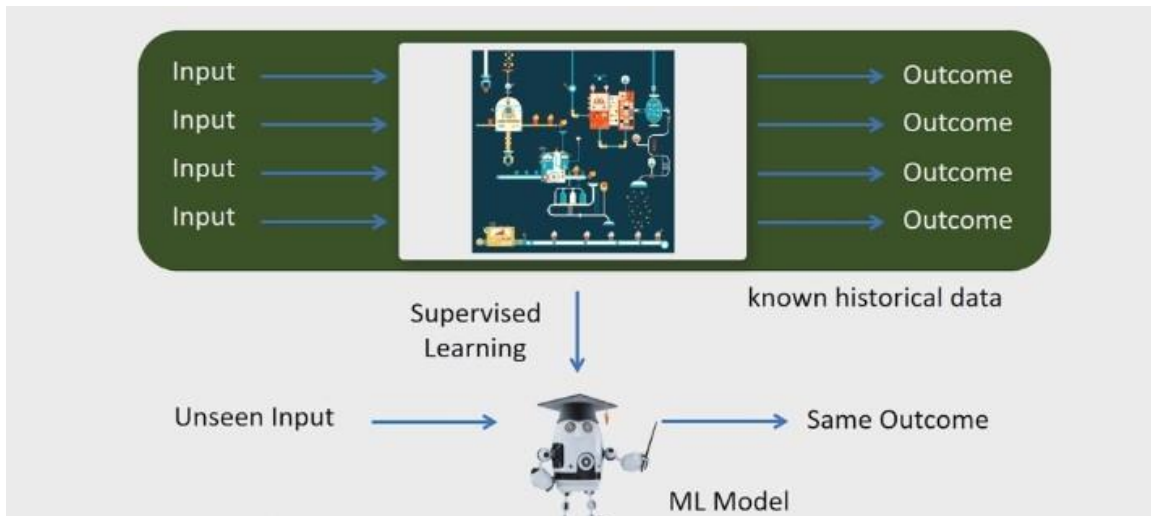


Fig 2.3: A trained model of Supervised Learning

2.2 TECHNOLOGIES USED

2.2.1 Python:

Python is an interpreted high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical codes for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (mainly procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage

collection system using reference counting. Python 3.0 was released in 2008 and was a significant revision of the language that is not entirely backwards compatible, and much Python 2 code does not run unmodified on Python 3. Python 2 was discontinued version 2.7.18 in 2020.

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 (including metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

2.2.2 Libraries:

Python's sizeable standard library, commonly cited as one of its greatest strengths provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary-precision decimals, manipulating regular expressions, and unit testing.

Specifications cover some parts of the standard library (for example, the Web Server Gateway Interface (WSGI) implementation follows PEP, but most modules are not. They are specified by their code, internal documentation, and test suites. However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.

2.2.3 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for

manipulating numerical tables and time series. It is free software released under the three clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. Wes McKinney started building what would become pandas at AQR Capital while he was a researcher from 2007 to 2010.

Features:

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.

2.2.4 Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. A procedural "pylab" interface is also based on a state machine (like OpenGL), designed to resemble MATLAB, though its use is discouraged closely. SciPy makes use of Matplotlib.

John D. Hunter originally wrote Matplotlib. Since then, it has an active development community and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and was further joined by Thomas Caswell. Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python 3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged not to support Python 2 past 2020 by signing the Python 3 Statement.

Tools required:

- Base map: map plotting with various map projections, coastlines, and political boundaries
- Cartopy: a mapping library featuring object-oriented map projection definitions and arbitrary point, line, polygon and image transformation capabilities. (Matplotlib v1.2 and above)
- Excel tools: utilities for exchanging data with Microsoft Excel • GTK tools: interface to the GTK library
- Qt interface
- Mplot3d: 3-D plots

- Nat grid: interface to the nat grid library for gridding irregularly spaced data.
- matplotlib2tikz: export to Pgfplots for smooth integration into LaTeX documents
- Seaborn: provides an API on top of Matplotlib that offers sane choices for plot style and colour defaults, defines simple high-level functions for common statistical plot types, and integrates with the functionality provided by Pandas

2.2.5 NumPy

NumPy is a library for the Python programming language, adding support for large, multidimensional arrays and matrices, along with an extensive collection of high-level mathematical functions to operate on these arrays. NumPy, Numeric, was created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Num array into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

Features:

- NumPy targets the C Python reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms are written for this version of Python often run much slower than compiled equivalents.
- NumPy addresses the slowness problem partly by providing multi-dimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, primarily inner loops, using NumPy.
- Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts many

additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language.

- Moreover, complementary Python packages are available; SciPy is a library that adds MATLAB-like functionality, and Matplotlib is a plotting package that provides MATLAB like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.
- Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image.
- The NumPy array as a universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

2.2.6 Scikit-learn

Scikit-learn (formerly scikits.learn and sklearn) is a free software machine learning library for the Python programming language.

It features various classification, regression and clustering algorithms, including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-learn is written mainly in Python and uses NumPy extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in Cython to improve performance.

A Cython wrapper around LIBSVM implements support vector machines, logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR. In such cases, extending these methods with Python may not be possible.

Scikit-learn integrates well with many other Python libraries, such as Matplotlib and Plotly for plotting, NumPy for array vectorization, Pandas data frames, SciPy, and many more.

2.2.7 Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Seaborn helps to explore and understand the data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

Its dataset-oriented, declarative API lets you focus on what the different elements of the plots mean rather than on the details of how to draw them.

There is no universally best way to visualize data. Different plots best answer different questions. Seaborn makes it easy to switch between different visual representations by using a consistent dataset-oriented API.

When statistical values are estimated, seaborn uses bootstrapping to compute confidence intervals and draw error bars representing the estimate's uncertainty.

Statistical analyses require knowledge about the distribution of variables in your dataset. The seaborn function `displot()` supports several approaches to visualizing distributions. These include classic techniques like histograms and computationally-intensive approaches like kernel density estimation.

Some seaborn functions combine multiple kinds of plots to give informative summaries of a dataset quickly. One, `jointplot()`, focuses on a single relationship. It plots the joint distribution between two variables along with each variable's marginal distribution

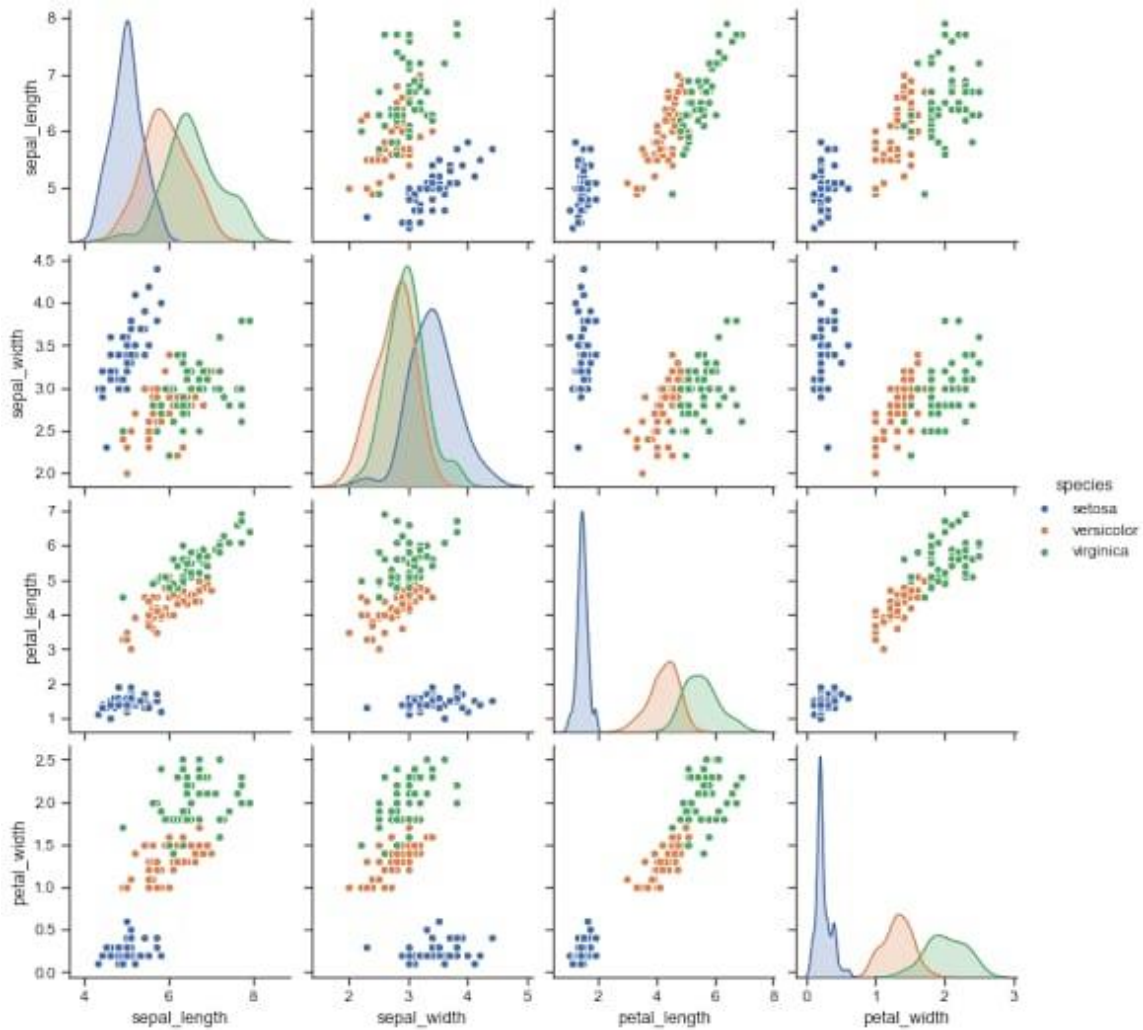


Fig 2.4: Seaborn plots

2.2.8 Jupyter Notebook IDE

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially refer to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format, depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells containing code, text (using Markdown), mathematics, plots and rich media, usually ending with the ". ipynb" extension. A Jupyter Notebook can be converted to several open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface via the nbconvert library or "jupyter nbconvert" command-line interface in a shell. To simplify the visualization of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document and convert it to HTML on the fly display it to the user.

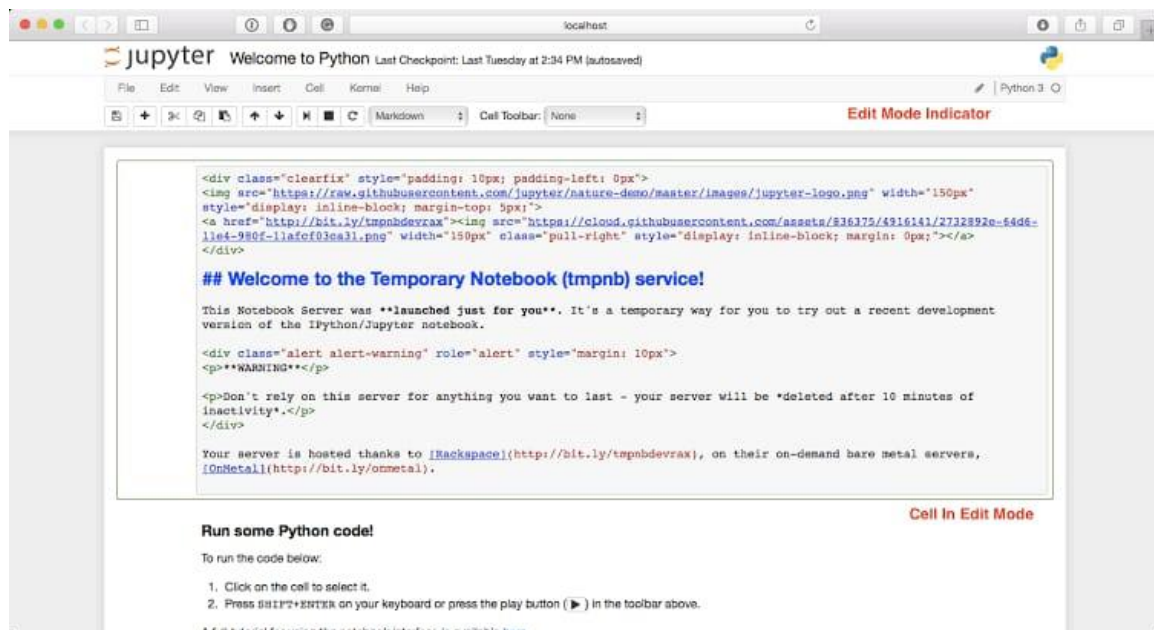


Fig 2.5: Jupyter IDE

Jupyter Notebook provides a browser-based REPL built upon many popular open-source libraries:

- IPython
- ØMQ (ZeroMQ)
- Tornado (web server)
- jQuery
- Bootstrap (front-end framework)
- MathJax

Jupyter Notebook can connect to many kernels to allow programming in different languages. By default, Jupyter Notebook ships with the IPython kernel. The 2.3 release (October 2014) was 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia, and Haskell.

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. According to The Atlantic, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

2.2.9 Jupyter kernels

A Jupyter kernel is responsible for handling various requests (code execution, code completions, inspection) and reply. Kernels talk to the other components of Jupyter using ZeroMQ and thus can be on the same or remote machines. Unlike many other Notebook-

like interfaces, in Jupyter, kernels are not aware that they are attached to a specific document and can be connected to many clients at once. Usually, kernels allow only a single language, but there are a couple of exceptions.

The Jupyter Notebook has become a popular user interface for cloud computing, and major cloud providers have adopted the Jupyter Notebook or derivative tools as a front-end interface for cloud users. Examples include Amazon's SageMaker Notebooks, Google's Colaboratory and Microsoft's Azure Notebook.

2.3 ALGORITHMS USED

We have used four different Machine Learning algorithms in this study to compare and contrast their performances, the algorithms are as follows:

2.3.1 Decision Tree (DT):

Decision Tree makes use of a model, were in a structure that resembles a tree used to make decisions and their likely outcomes, as well as chance event outcomes, resource costs, and utility. Each node in the tree is a representation of a conditional statement('if') and on the whole, the decision tree can be viewed as a representation of a nested conditional.

A mandatory node that is present for all tree is root node. Root node definitely have leaf nodes which return the decision based on the attribute or parameter on each node. For each level on the tree two parameters are determined: Information Gain and Entropy.

Entropy: is nothing but quantitative measure randomness or uncertainty. If entropy is 0, the data has no randomness. Zero value of entropy depicts the decision to be certain and on the contrary higher values of entropy describes higher levels of uncertainty.

$$H(s) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

Information Gain: It is the change that occurs with entropy after deciding a particular attribute with respect to the independent variables.

$$IG(S, A) = H(s) - \sum_{i=0}^n p(x) * H(x)$$

Where, p(x)=is the probability of the attribute

These parameters are again calculated after a decision is returned by a leaf node and the attribute with the highest IG is removed from the list. This process keeps on repeating resulting in the depletion of attributes and finally classification of the datasets.

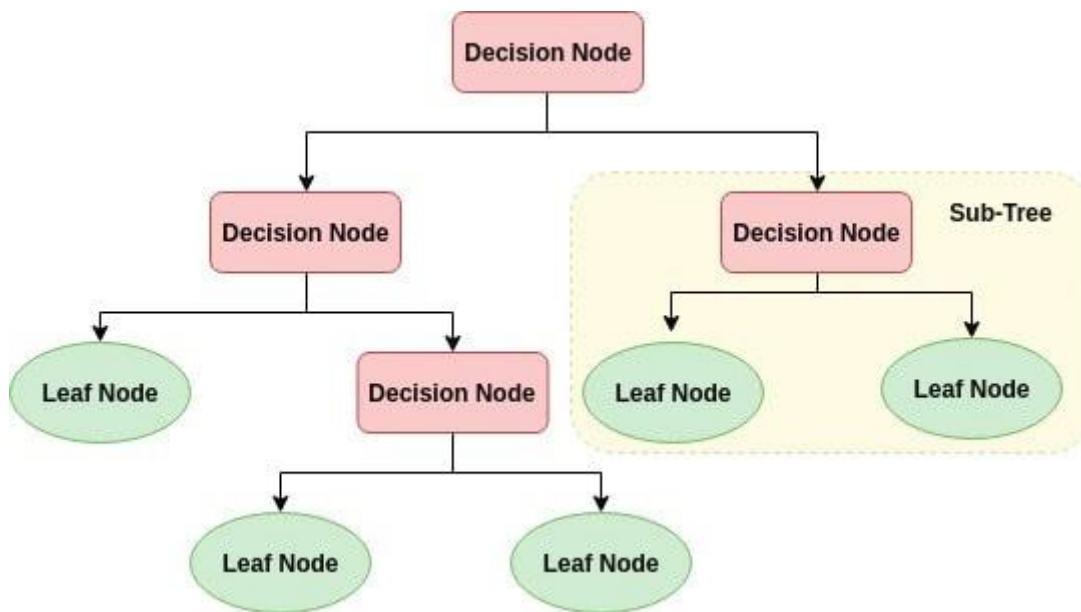


Fig 2.6: Decision Tree

Steps in DT classification are:

- 1.Importing libraries (for DT) and input dataset in python.
- 2.The class variable split and the validation split is done to obtain the test and train dataset.

3. The decision tree optimization is done by specifying the criterion for the attribute selection
4. The evaluation parameters are obtained to compare with the other classifiers
5. Then the model is compiled and fit to give the prediction of faults in the dataset

2.3.2 Random Forest (RF)

Random forest is an ensemble learning (process by which multiple several classifiers are created and combined to solve a problem) method that is applicable for classification as well as regression by combining an aggregate of decision trees at training time and the output of this algorithm is based on the output (can be either mode or mean/average) of the individual trees that constitute the forest.

The design of RF is such that group of diverse (to an extent) models operate together to solve a problem. This mathematically can be represented with correlation (low correlation in this case). This method is proven to be a better option for several problem throughout time because each tree try to make up for the mistakes made by any other. DT are very specific on the data that they are trained on, even small changes made to the set can disturb the structure of the tree

Each tree in RF get data such that each tree is not very similar to any other tree such that making the behaviour of trees diverse. This is achieved by two methods:

Bagging: random sampling from a dataset and this is done without any replacement (this is repeated for each individual trees). It is important to note that this method does not take any unique subset of the dataset and training each DTs on them. Replacement plays an important role in making sure that DTs make up for each other's mistakes.

Feature Randomness: Splitting in DTs occur such that there is most separation between the left and right nodes after considering every feature present in the dataset. But in the case of RF, as there are only limited features in each tree the splitting occurs

with very different separation in individual trees and it is very unlikely that two trees will be exactly the same.

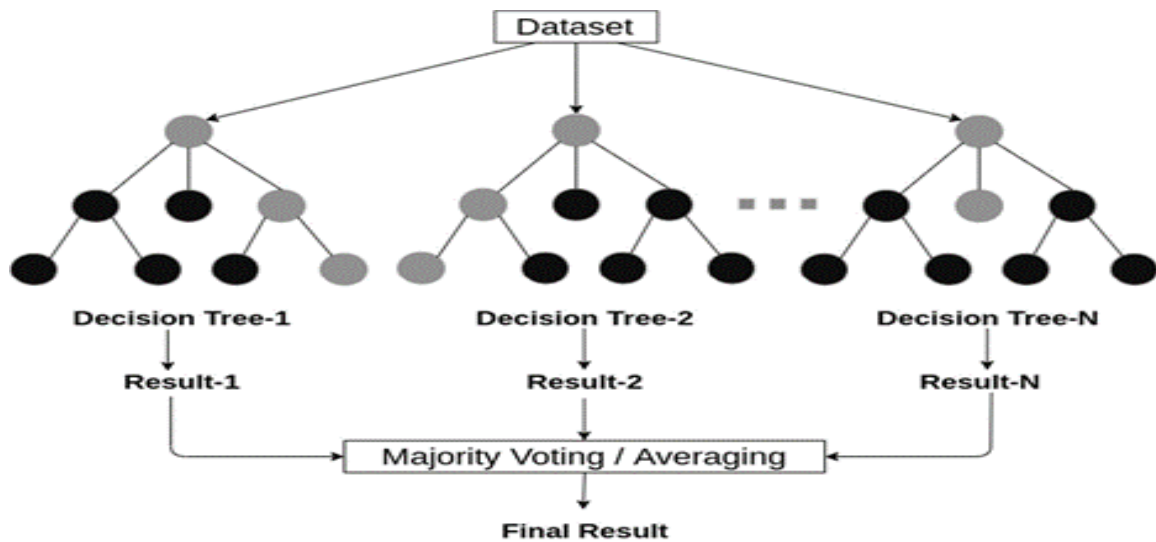


Fig 2.7: Random forest classifier

2.3.3 Gaussian Naive Bayes

Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data. We have explored the idea behind Gaussian Naive Bayes along with an example. Before going into it, we shall go through a brief overview of Naive Bayes.

Naive Bayes are a group of supervised machine learning classification algorithms based on the Bayes theorem. It is a simple classification technique, but has high functionality. They find use when the dimensionality of the inputs is high. Complex classification problems can also be implemented by using Naive Bayes Classifier.

2.3.3.1 Bayes Theorem

Bayes Theorem can be used to calculate conditional probability. Being a powerful tool in the study of probability, it is also applied in Machine Learning.

The Formula For Bayes' Theorem Is

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

where:

$P(A)$ = The probability of A occurring

$P(B)$ = The probability of B occurring

$P(A|B)$ = The probability of A given B

$P(B|A)$ = The probability of B given A

$P(A \cap B)$ = The probability of both A and B occurring

Fig 2.8: Formula for bayes theorem

Bayes Theorem has widespread usage in variety of domains.

2.3.3.2 Naive Bayes Classifier

Naive Bayes Classifiers are based on the Bayes Theorem. One assumption taken is the strong independence assumptions between the features. These classifiers assume that the value of a particular feature is independent of the value of any other feature. In a supervised learning situation, Naive Bayes Classifiers are trained very efficiently. Naive Bayed classifiers need a small training data to estimate the parameters needed for classification. Naive Bayes Classifiers have simple design and implementation and they can applied to many real life situations.

2.3.3.3 Gaussian Naive Bayes

When working with continuous data, an assumption often taken is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. The likelihood of the features is assumed to be-

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

Sometimes assume variance is independent of Y (i.e., σ_i), or independent of X_i (i.e., σ_k) or both (i.e., σ)

Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian (normal) distribution.

An approach to create a simple model is to assume that the data is described by a Gaussian distribution with no co-variance (independent dimensions) between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all what is needed to define such a distribution.

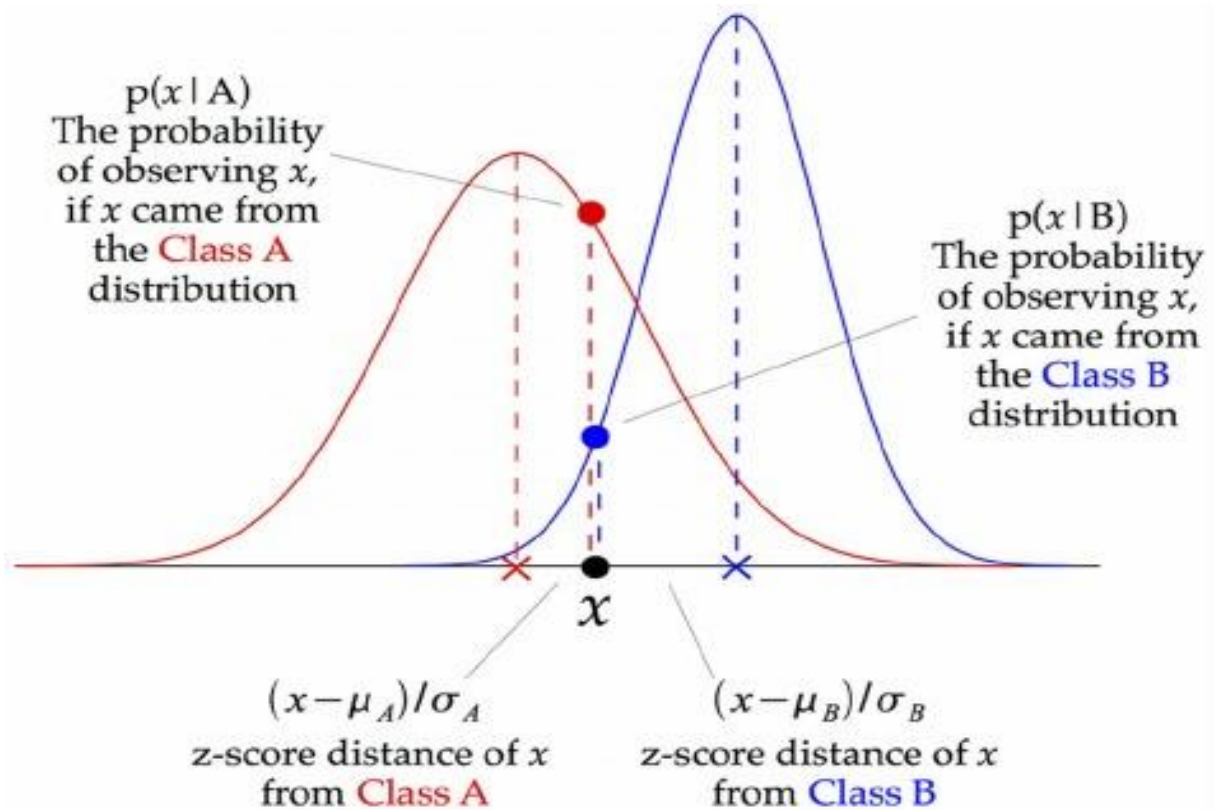


Fig 2.9: Gaussian NB classifier

The above illustration indicates how a Gaussian Naive Bayes (GNB) classifier works. At every data point, the z-score distance between that point and each class-mean is calculated, namely the distance from the class mean divided by the standard deviation of that class.

Thus, we see that the Gaussian Naive Bayes has a slightly different approach and can be used efficiently.

2.3.4 Boosting

Ensemble meta-algorithm that is mainly used to reduce bias (group of presumptions made so the target function is relatively easier to learn) and variance (relative change in the estimated target function when the training data is changed).

2.3.4.1 Ada Boost (AB)

It is ML boosting algorithm, which aggregates several “weak classifiers” (nothing but classifiers with performance just slightly better than the random chance) to form a “strong classifier. In the case of AdaBoost the weak learners are also known as decision stumps. The decision stumps are characterized as weak learners because they are trees with only a single split. This algorithm mainly focuses on the difficult classification instances rather than the ones that are clearly classified. The process starts with each observation having equal weight by training a DT. After the evaluation of the first DT, the modified such that the observations that are difficult to classify are assigned with higher weights and vice versa. A new tree is now created and works with this highly weighted data. This process continues based on the input parameters. These weak learners are put together to form a new strong learner as mentioned above.

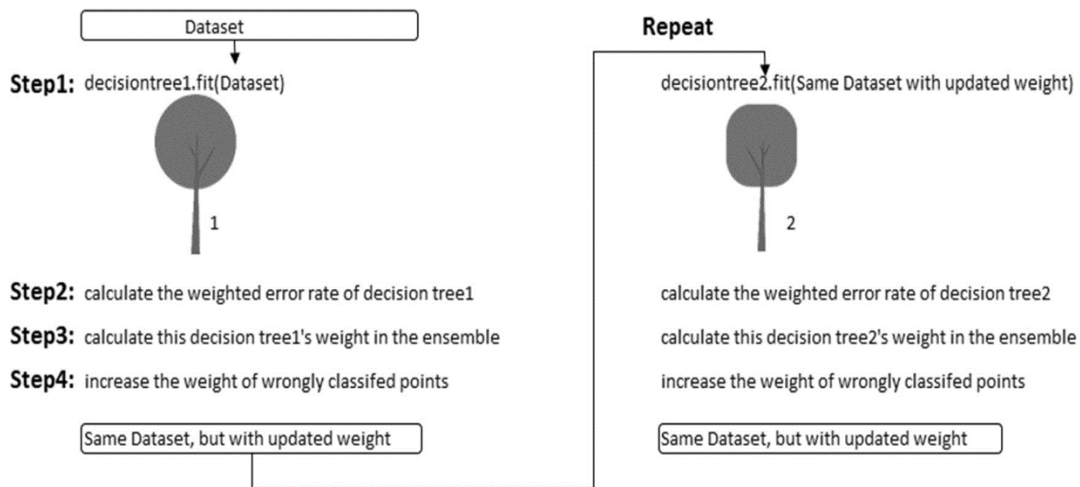


Fig 2.10: Steps in AdaBoost classifier

2.3.4.2 Gradient Boost (GB)

Training of models is done in a subsequent and additive method. So, the intuition behind gradient boosting algorithm is to repetitively leverage the patterns in residuals and strengthen a model with weak predictions and make it better. The main difference between AB and GB is the way both these algorithms identify the faults of the weak learners. The three causes of discrepancy between original and predicted values are noise, variance, and bias. As noise cannot be reduced only the other two are reduced by the use of ensemble methods While the AB uses weighted data points, GB uses gradients in the loss function (LF). LF is used as a measure of how well our model matches the training data that we are given. A main advantage of GB is that it allows the user to choose the cost function and optimizes that cost function.

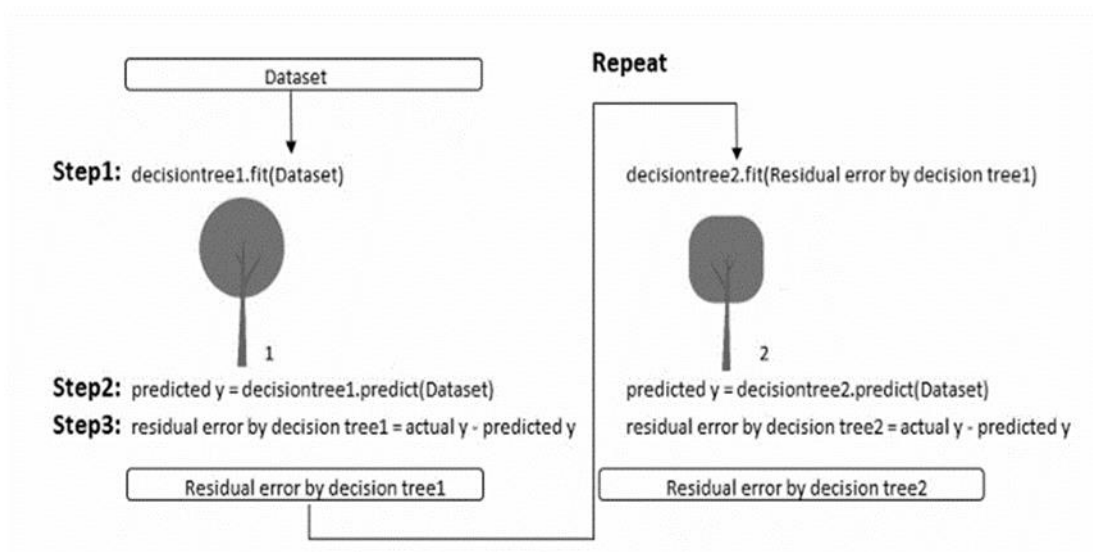


Fig 2.11: Steps in Gradient Boost classifier

CHAPTER-3

DATA VISUALIZATION

3.1 CLEANING OF DATA SET

This is an essential step to perform before creating a visualization. Clean, consistent data will be much easier to visualize. Clean data is data that is free of errors or anomalies which may make it hard to use or analyze the data. Starting from a clean dataset allows you to focus on creating an effective visualization rather than trying to diagnose and and fix issues while creating visualizations. Data cleaning tasks will be very dependent on the dataset that you're working with. In most cases, data cleaning involves:

- 1.Removing unnecessary variables
- 2.Deleting duplicate rows/observations
- 3.Addressing outliers or invalid data
- 4.Dealing with missing values
- 5.Standardizing or categorizing values
- 6.Correcting typographical errors

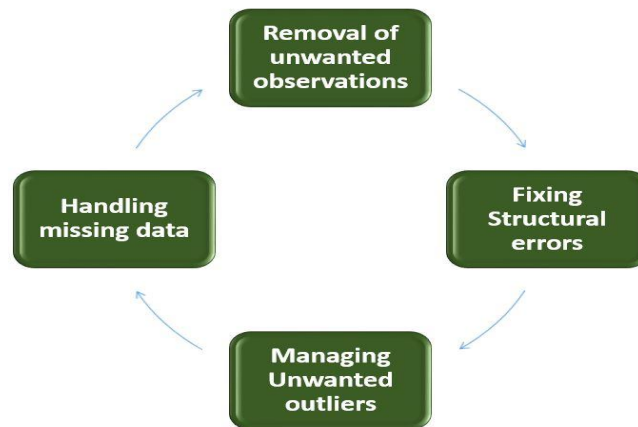


Fig 3.1: Steps Involved in Data Cleaning

The following observations have been made during the process of data cleaning before visualizing the data

```
In [8]: 1 df.nunique()
Out[8]: Temperature      16
        Humidity         20
        Moisture         41
        Soil Type         6
        Crop Type        11
        Nitrogen          26
        Potassium         13
        Phosphorous       33
        Fertilizer Name    7
        dtype: int64

In [9]: 1 df.shape
Out[9]: (149, 9)

In [10]: 1 df.isnull().sum()
Out[10]: Temperature      0
         Humidity         0
         Moisture         0
         Soil Type         0
         Crop Type         0
         Nitrogen          0
         Potassium         0
         Phosphorous       0
         Fertilizer Name    0
         dtype: int64
```

Fig 3.2: Observations of Null Values and Unique Values of parameters

Figure 3.2 shows the number of null values and also the number of unique values of parameters that are available in the existing data set.

```
In [19]: 1 data['Crop Type'].value_counts().to_dict()
```

```
Out[19]: {'Sugarcane': 49,  
          'Cotton': 45,  
          'Millets': 41,  
          'Paddy': 40,  
          'Pulses': 40,  
          'Wheat': 36,  
          'Ground Nuts': 28,  
          'Barley': 28,  
          'Tobacco': 28,  
          'Oil seeds': 25,  
          'Maize': 24}
```

Fig 3.3 : Types of Crops in that are considered.

Figure 3.3 shows the different types of crops that have been considered for the project and the count of each crop of in the dataset

```
In [10]: 1 df['Fertilizer Name'].value_counts().to_dict()
```

```
Out[10]: {'Urea': 88,  
          'DAP': 69,  
          '28-28': 65,  
          '20-20': 53,  
          '14-35-14': 53,  
          '17-17-17': 28,  
          '10-26-10': 17,  
          '10-26-26': 11}
```

Fig 3.4: Different fertilizers used.

Figure 3.4 shows the different types of fertilizers that have been considered for the project and the count of each fertilizer of in the data set.

3.2 WHAT IS DATA VISUALIZATION?

With the help of data visualization, we can see how the data looks like and what kind of correlation is held by the attributes of data. It is the fastest way to see if the features correspond to the output. With the help of following Python recipes, we can understand ML data with statistics.

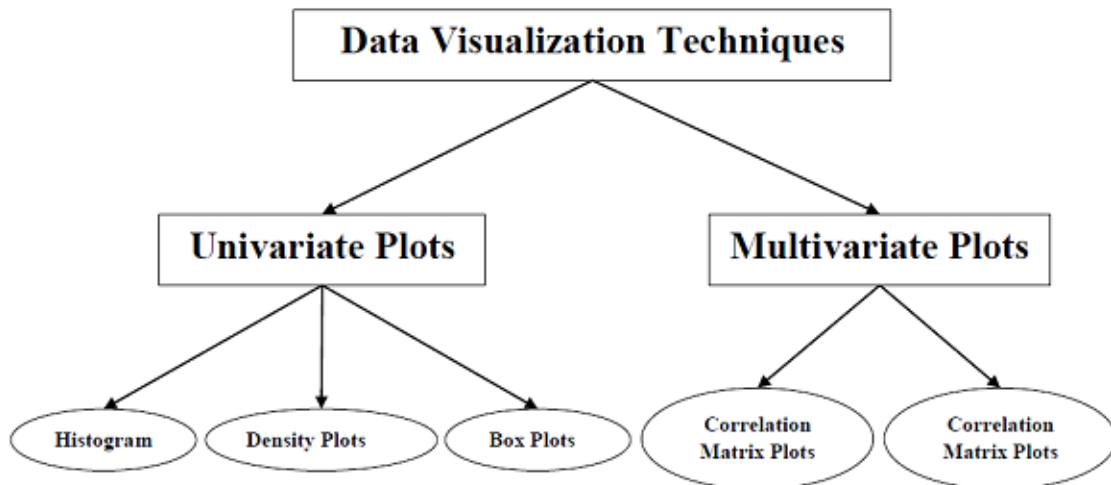


Fig 3.5: Types of Data Visualization Techniques

3.2.1 Why Visualization?

Do you think giving you the data of let's say 1 million points in a table/Database file and asking you to provide your inferences by just seeing the data on that table is feasible? Unless you're a super human its not possible. This is when we make use of Data visualization, wherein all the data will be transformed into some form of plots and analyzed further from that. As being a human, we are more used to grasp a lot of info from diagrammatic representation than the counterparts. Okay ! So since it is said that we need to convert the data from a boring table into interesting pictorial form like a scatter plot or Bar chart, You may wonder, How can I do it? Do I need to write my own code for that ? **NO!** Actually we can make use of very good packages from some popular programming languages which are readily available and can make the work pretty much simple with just a single line of code. That's the power of modern programming . As a human, we can just visualize anything in

either in 2-d or 3-d. But trust me almost of the data that you obtain in real world won't be this way. As a Machine learning engineer, working with more than 1000-dimensional data is very common. So, what can we do in such cases where data is more than 3D? There are some **Dimensionality Reduction (DR)** techniques like PCA, TSNE, LDA etc which helps you to convert data from a higher dimension to a 2D or 3D data in order to visualize them. There may be some loss of information with each DR techniques, but only they can help us visualize very high dimensional data on a 2d plot. TSNE is one of the state-of-the-art DR techniques employed for visualization of high dimensional data.

From perspective of building models, by visualizing the data we can find the hidden patterns, explore if there are any clusters within data and we can find if they are linearly separable/too much overlapped etc. From this initial analysis we can easily rule out the models that won't be suitable for such a data and we will implement only the models that are suitable, without wasting our valuable time and the computational resources. This part of data visualization is a predominant one in initial **Exploratory Data Analysis (EDA)** on the field of Data science/ML.

The following techniques have been used to visualize the data from the collected data set that consists of N, P, K values, temperature, humidity, moisture, crop type, soil type and fertilizer name corresponding to the respective data.

- a. Histograms
- b. Correlation Matrix
- c. Bar Plots

a. Histogram

Histograms group the data in bins and is the fastest way to get idea about the distribution of each attribute in dataset. The following are some of the characteristics of histograms –

- It provides us a count of the number of observations in each bin created for visualization.
- From the shape of the bin, we can easily observe the distribution i.e. whether it is Gaussian, skewed or exponential.
- Histograms also help us to see possible outliers.

The following observations have been made while plotting the histograms of different input parameters like temperature, moisture, humidity and the macro nutrient values present in the soil like nitrogen(N), phosphorous(P), potassium(k) (these nutrient values can be obtained by testing of the soil) excluding the crop type, soil type, and fertilizer names as they have not been labelled with values. So, labelling has been done for those parameters to proceed further. Different histograms have been plotted so as to get a clear understanding for the machine to read the data like the number of values of an input parameter that are present at a particular instant.

These are some of the histograms that have been plotted i.e., humidity, moisture, temperature, nitrogen, potassium and phosphorus.

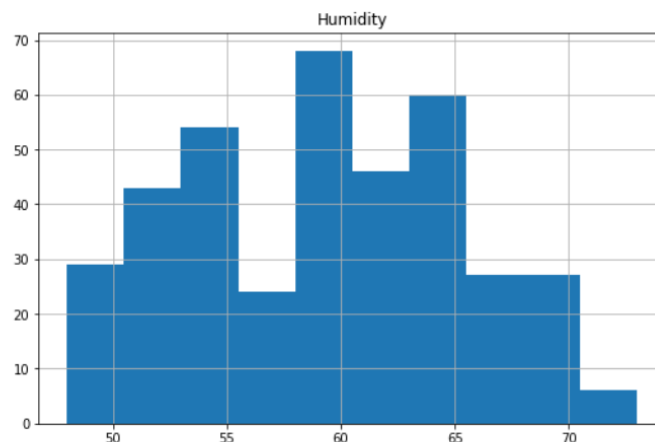


Fig3.6: Histogram of humidity

Figure 3.6 shows the values of humidity that are in between 15-55 with a highest frequency of 55, 55-60 occurred with a highest frequency of 68, 60-65 occurred with a highest frequency of 60, 65-70 occurred with highest frequency of 29, 70-74 occurred with highest frequency of 5.

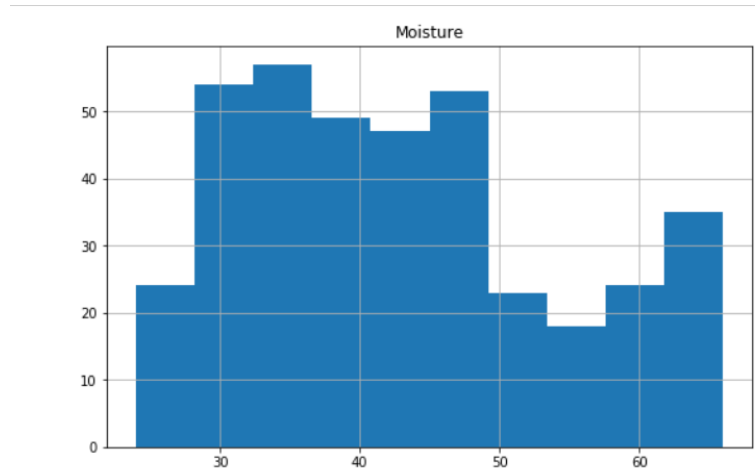


Fig 3.7: Histogram of Moisture

Figure 3.7 shows the values of moisture that are in between 0-30 with a highest frequency of 24, 30-40 occurred with a highest frequency of 58, 40-50 occurred with a highest frequency of 52, 50-60 occurred with highest frequency of 25, 60-65 occurred with highest frequency of 35.

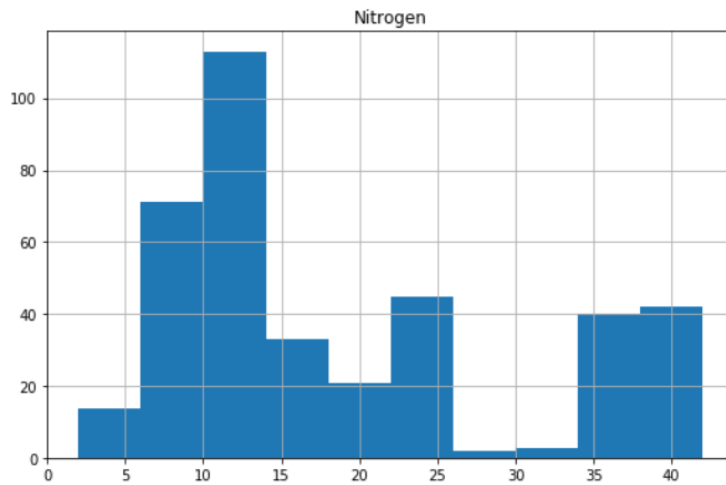


Fig 3.8: Histogram of nitrogen

Figure 3.2.1(c) shows the values of nitrogen that are in between 0-5 with a highest frequency of 15, 5-10 occurred with a highest frequency of 65, 10-15 occurred with a highest frequency of 115, 15-20 occurred with highest frequency of 20, 20-25 occurred with highest frequency of 43, 25-30 occurred with a highest frequency of 5, 30-35 occurred with a highest frequency of 40, 35-43 occurred with a highest frequency of 42

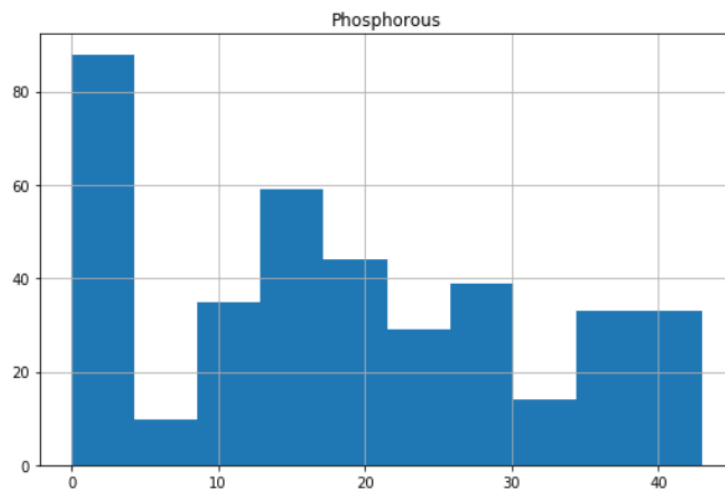


Fig 3.9: Histogram of Phosphorous

Figure 3.9 shows the values of phosphorous that are in between 0-10 with a highest frequency of 85, 10-20 occurred with a highest frequency of 43, 20-30 occurred with a highest frequency of 38, 30-40 occurred with highest frequency of 35, 40-45 occurred with highest frequency of 35.

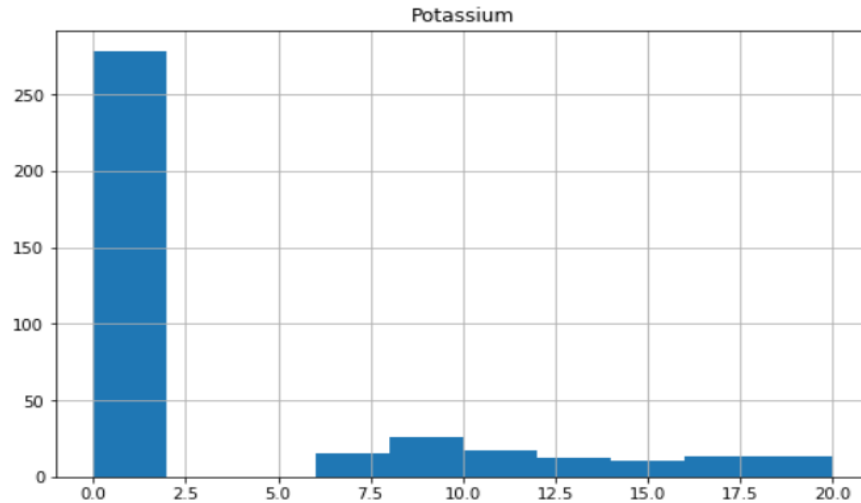


Fig 3.10: Histogram of potassium

Figure 3.10 shows the values of potassium that are in between 0.0-2.5 with a highest frequency of 255, 2.5-5.0 occurred with a highest frequency of 0, 5.0-7.5 occurred with a highest frequency of 10, 7.5-10.0 occurred with highest frequency of 17, 10-12.5 occurred with highest frequency of 15.12.5-15.0 occurred with highest frequency of 7, 15.0-17.5 occurred with highest frequency of 10, 17.5-20.0 occurred with highest frequency of 11.

b. Correlation Matrix

Correlation coefficients are indicators of the strength of the linear relationship between two different variables, x and y. A linear correlation coefficient that is greater than zero indicates a positive relationship. A value that is less than zero signifies a negative relationship. The correlation coefficient (ρ) is a measure that determines the degree to which the movement of two different variables is associated. The most common

correlation coefficient, generated by the Pearson product-moment correlation, is used to measure the linear relationship between two variables. However, in a non-linear relationship, this correlation coefficient may not always be a suitable measure of dependence.

The possible range of values for the correlation coefficient is -1.0 to 1.0. In other words, the values cannot exceed 1.0 or be less than -1.0. A correlation of -1.0 indicates a perfect negative correlation, and a correlation of 1.0 indicates a perfect positive correlation. If the correlation coefficient is greater than zero, it is a positive relationship. Conversely, if the value is less than zero, it is a negative relationship. A value of zero indicates that there is no relationship between the two variables.

Note: When interpreting correlation, it's important to remember that just because two variables are correlated, it does not mean that one causes the other

Following Correlation Matrix has been obtained for the given input parameters showing how they are related to each other.

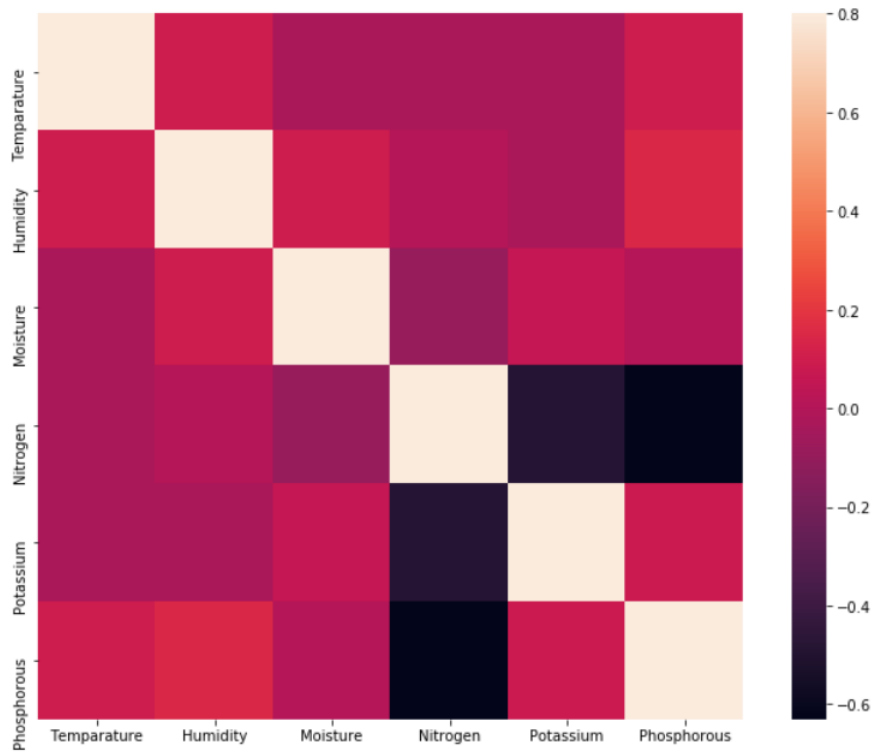


Fig 3.11: Correlation Matrix

The results that are inferred from the above Correlation Matrix are tabulated below

Table 3.1: Observations made from Correlation Matrix

	Temperature	Humidity	Moisture	Nitrogen	Potassium	Phosphorous
Temperature	1.000000	0.094126	-0.029274	-0.025380	-0.029641	0.094987
Humidity	0.094126	1.000000	0.092427	0.009129	-0.031292	0.147147
Moisture	-0.029274	0.092427	1.000000	-0.90268	0.057861	0.001713
Nitrogen	-0.025380	0.009129	-0.90268	1.000000	-0.48831	-0.629624
Potassium	-0.029641	-0.031292	0.057861	-0.48831	1.000000	0.081873
Phosphorous	0.094987	0.147147	0.001713	-0.629624	0.081874	1.000000

- Correlation coefficients are used to measure the strength of the linear relationship between two variables.
- A correlation coefficient greater than zero indicates a positive relationship while a value less than zero signifies a negative relationship
- A value of zero indicates no relationship between the two variables being compared.
- A negative correlation, or inverse correlation, is a key concept in the creation of diversified portfolios that can better withstand portfolio volatility.
- Calculating the correlation coefficient is time-consuming, so data are often plugged into a calculator, computer, or statistics program to find the coefficient.

c. Bar Plots

A bar plot shows categorical data as rectangular bars with the height of bars proportional to the value they represent. It is often used to compare between values of different categories in the data.

The following Bar Plots have been obtained.

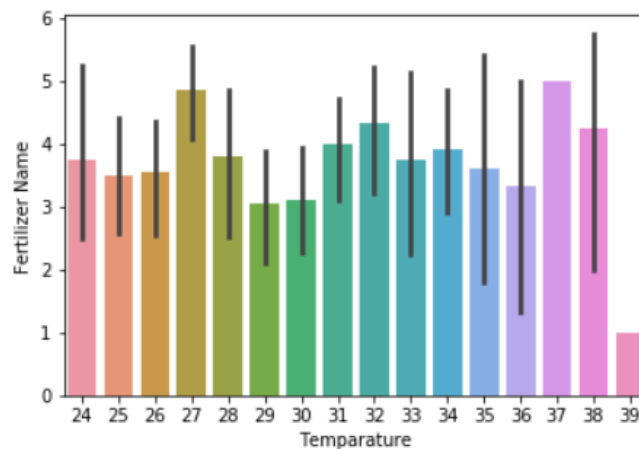


Fig 3.12: Temperature vs Fertilizer

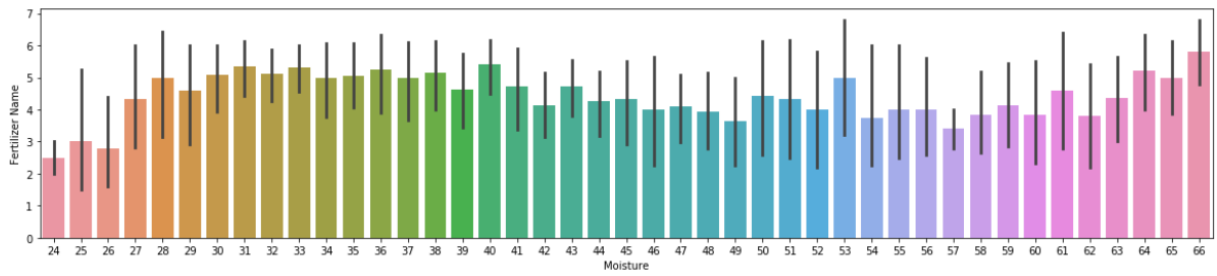


Fig 3.13: Moisture vs Fertilizer

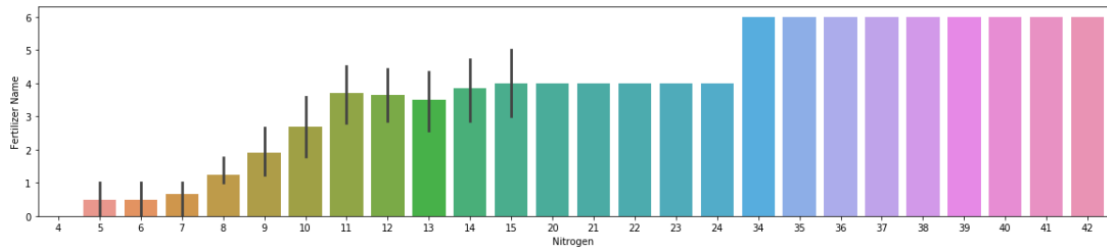


Fig 3.14: Nitrogen vs Fertilizer

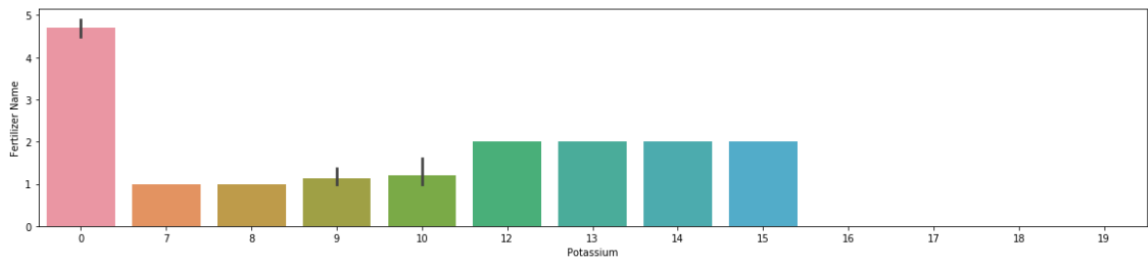


Fig 3.15: Potassium vs Fertilizer

CHAPTER-4

RESULTS AND DISCUSSION

4.1 EVALUATION PARAMETERS

4.1.1 Confusion Matrix: In every algorithm, a set of test data will be kept aside for comparison by the Cross Validation method and the comparison of the performance of the classifier on this test dataset with the true labelled values is visualized as confusion matrix.

A Confusion matrix looks like:

Table 4.1: Confusion Matrix Representation

Confusion Matrix		Predicted	
		Yes	No
Actual	Yes	TP	FN
	No	FP	TN

Where,

TP = Nets that actually trojan nets and are predicted as trojan nets. FN = Nets that are actually trojan nets but predicted as normal nets.

FP = Nets that are actually normal nets but are predicted as trojan nets. TN = Nets that are actually normal nets and are predicted as normal nets.

4.1.2 Accuracy: Accuracy is defined as the fraction of the predictions that the classifier got right

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

4.1.3 Precision: Precision is defined as how often the classifier predicts the output correctly, irrespective of it being yes or no

$$Precision = \frac{TP}{TP + FP} \text{ or } \frac{TN}{TN + FN}$$

4.1.4 Recall: Recall is defined as the number of actual positives our model labels as positive

$$Recall = \frac{TP}{TP + FN}$$

4.1.5 F1Score: F1 score is a function of Precision and Recall. It provides a balance between precision and recall as it is the weighted average of the two

$$F1 \text{ Score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Confusion Matrices of all the proposed Machine Learning algorithms have been obtained i.e., for Decision Tree, Random Forest, Gaussian NB, Ada Boost, Gradient Boost through which we can infer that evaluation parameters like accuracy, recall, precision and f-score.

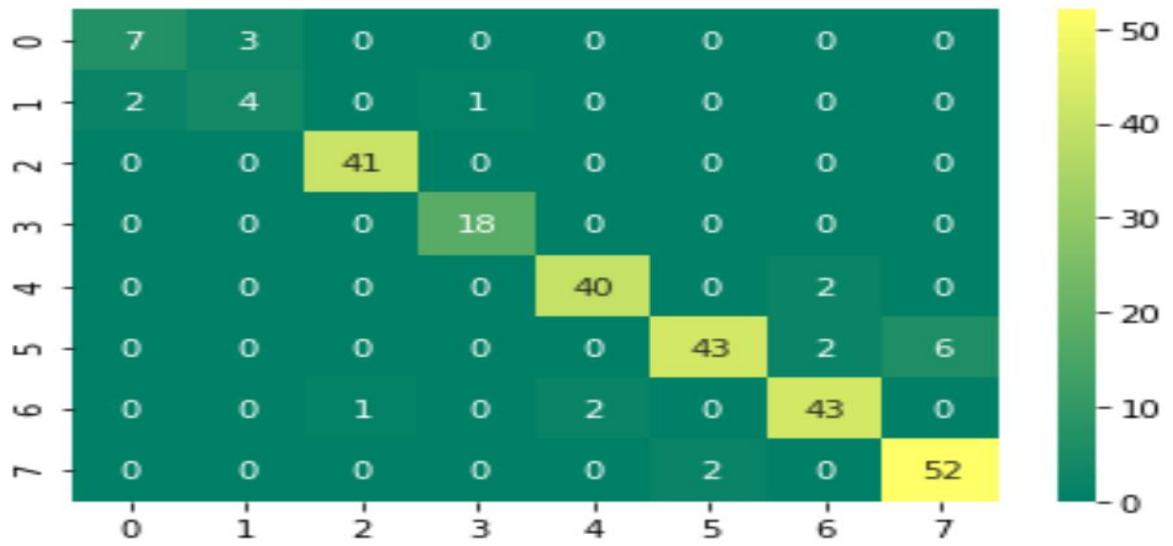


Fig 4.1: Confusion Matrix for Gaussian NB

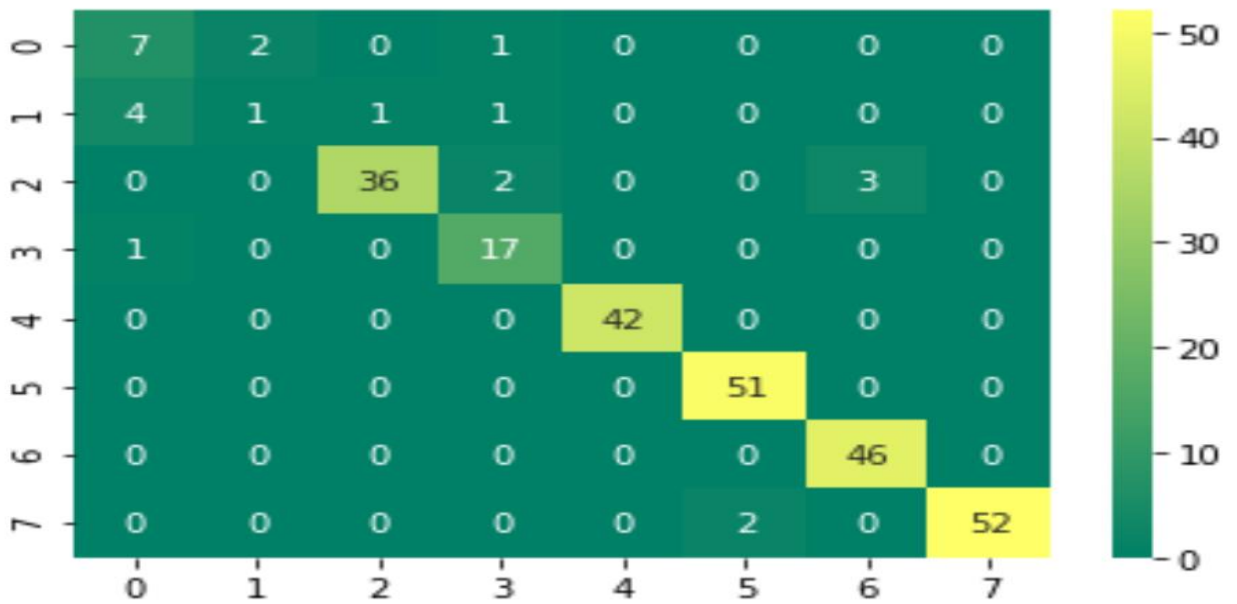


Fig 4.2: Confusion Matrix for Decision Tree

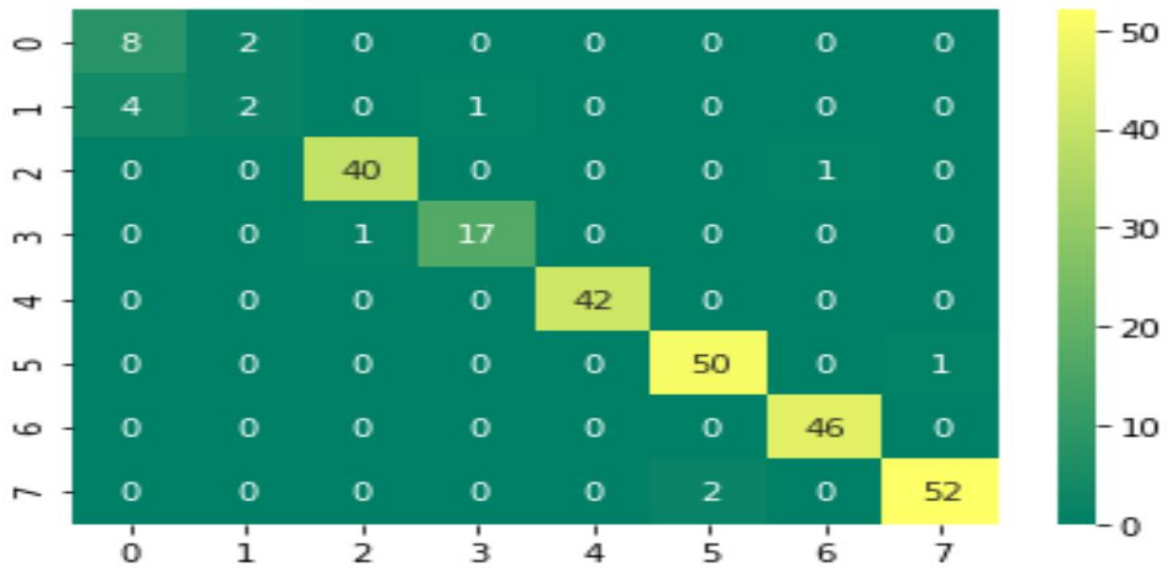


Fig 4.3: Confusion Matrix for Random Forest

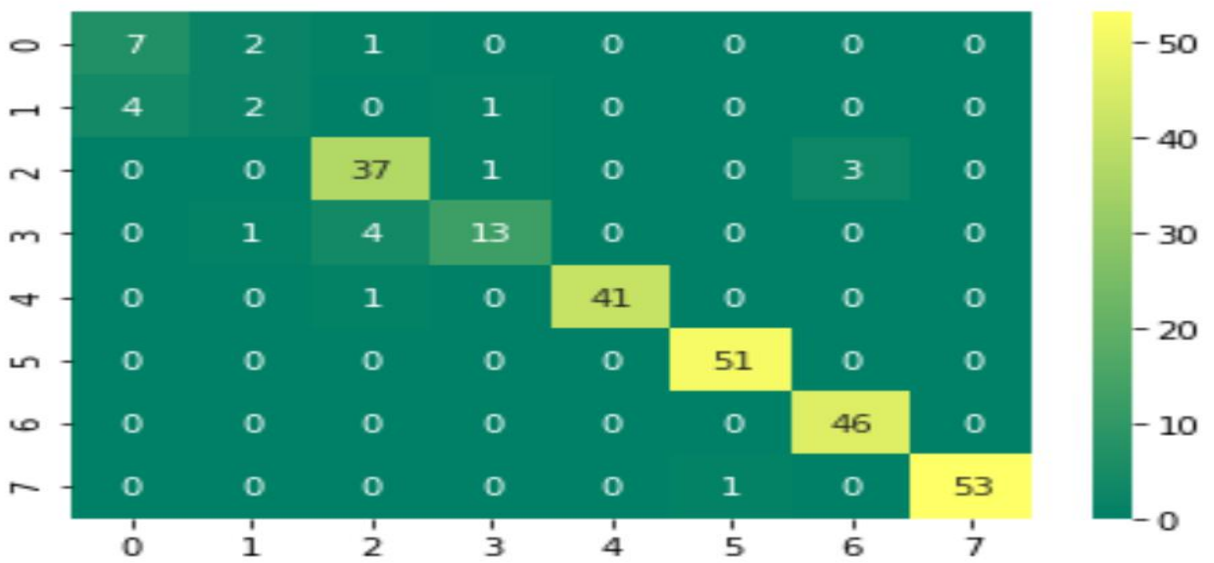


Fig 4.4: Confusion Matrix for Gradient Boost

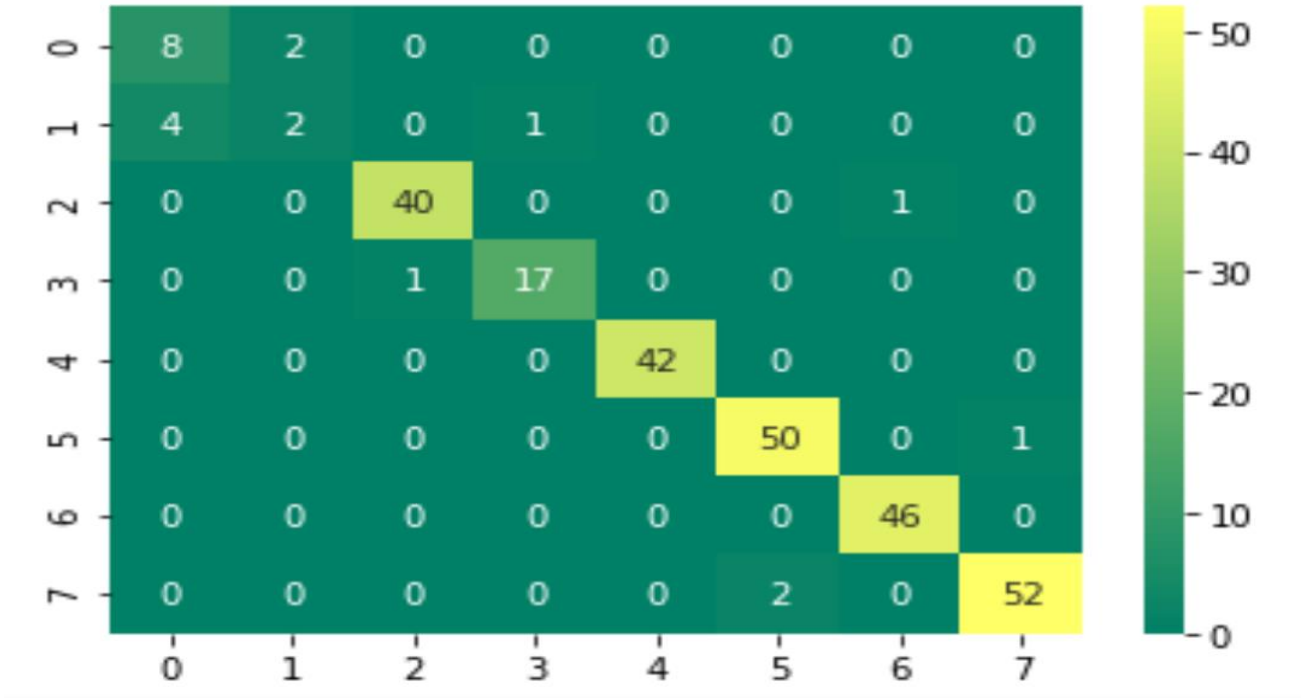


Fig 4.5: Confusion Matrix for Ada Boost

4.2 GRADIENT BOOST CLASSIFIER

```
1 from sklearn.ensemble import GradientBoostingClassifier
2
3 #Create Gradient Boosting Classifier
4 gb = GradientBoostingClassifier()
5
6 #Train the model using the training sets
7 gb.fit(X_train, y_train)
8
9 #Predict the response for test dataset
10 g_pred = gb.predict(X_test)
11 x_pred = gb.predict(X_train)
```

```
1 from sklearn import metrics
2 # Model Accuracy, how often is the classifier correct?
3 print("Accuracy of gradient boost:",metrics.accuracy_score(y_test, g_pred))
4 print("Classification Report")
5 print(classification_report(y_test,g_pred))
6 print('\n')
7 data=confusion_matrix(y_test, g_pred)
8 sns.heatmap(data,annot=True,cmap="summer")
9 plt.show()
```

Fig 4.6: Gradient Boost Classifier

Table 4.2: Results of Gradient Boost Classifier

	Precision	Recall	F1-score
0	0.75	0.69	0.72
1	0.40	0.22	0.29
2	0.90	0.92	0.91
3	0.95	0.73	0.79
4	1.00	0.97	0.98
5	0.98	0.93	0.95
6	1.00	0.82	0.90
7	0.75	0.98	0.85
Accuracy			0.88
Macro average	0.83	0.78	0.80
Weighted average	0.88	0.88	0.87

4.3 DECISION TREE

```

1
2 from sklearn.tree import DecisionTreeClassifier
3 dtc = DecisionTreeClassifier().fit(X_train, y_train)
4 dtc.fit(X_train, y_train)
5 d_pred = dtc.predict(X_test)
6 print('Accuracy of Decision Tree classifier on training set: {:.2f}'
7       .format(dtc.score(X_train, y_train)))
8 print('Accuracy of Decision Tree classifier on test set: {:.2f}'
9       .format(dtc.score(X_test, y_test)))

```

Accuracy of Decision Tree classifier on training set: 1.00
 Accuracy of Decision Tree classifier on test set: 0.93

```

1 from sklearn.metrics import classification_report
2 k_range = range(1, 10)
3 scores=[]
4 for k in k_range:
5     clf2 = DecisionTreeClassifier(max_depth = k).fit(X_train, y_train)
6     scores.append(clf2.score(X_train, y_train))
7 print(max(scores))
8 print('Accuracy of Decision Tree classifier on training set: {:.2f}'
9       .format(clf2.score(X_train, y_train)))
10 print('Accuracy of Decision Tree classifier on test set: {:.2f}'
11       .format(clf2.score(X_test, y_test)))
12 print("Classification Report")
13 print(classification_report(y_test, d_pred))
14 print('\n')
15 data=confusion_matrix(y_test, d_pred)
16 sns.heatmap(data,annot=True,cmap="summer")
17 plt.show()

```

Fig4.7: Decision Tree Classifier

Table 4.3: Results of Decision Tree Classifier

	Precision	Recall	F1-score
0	0.80	0.62	0.70
1	0.57	0.44	0.50
2	0.90	0.92	0.91
3	0.88	0.93	0.90
4	1.00	1.00	1.00
5	0.98	0.93	0.95
6	0.95	1.00	0.97
7	0.95	1.00	0.97
Accuracy			0.93
Macro average	0.88	0.85	0.86
Weighted average	0.93	0.93	0.93

4.4 RANDOM FOREST:

```
1 from sklearn.ensemble import AdaBoostClassifier
2 from sklearn import metrics
3 clff = AdaBoostClassifier(n_estimators=100,learning_rate=1,random_state=1).fit(X_train, y_train)
4 print('Accuracy of adaboost classifier on training set: {:.2f}'
5       .format(clff.score(X_train, y_train)))
6 print('Accuracy of adaboost classifier on test set: {:.2f}'
7       .format(clff.score(X_test, y_test)))
8 a_pred=classifier.predict(X_test)
9 print("Classification Report ")
10 print(classification_report(y_test,a_pred))
11 print('\n')
12 data=confusion_matrix(y_test, a_pred)
13 sns.heatmap(data,annot=True,cmap="summer")
14 plt.show()
```

Fig 4.8: Random Forest Classifier

Table 4.4: Results of Random Forest Classifier

	Precision	Recall	F1-score
0	0.82	0.69	0.75
1	0.00	0.00	0.00
2	0.88	0.92	0.90
3	0.79	1.00	0.88
4	0.92	1.00	0.96
5	1.00	0.91	0.95
6	1.00	1.00	1.00
7	0.94	1.00	0.97
Accuracy			0.93
Macro average	0.79	0.82	0.80
Weighted average	0.90	0.93	0.91

4.5 GAUSSIAN NB

```
1 from sklearn.naive_bayes import GaussianNB
2 from sklearn.metrics import classification_report, confusion_matrix
3 nb = GaussianNB()
4 nb.fit(X_train, y_train)
5 n_pred=nb.predict(X_test)
6 print('Accuracy of GaussianNB classifier on training set: {:.3f}'.format(nb.score(X_train, y_train)))
7 print('Accuracy of GaussianNB classifier on test set: {:.3f}'.format(nb.score(X_test, y_test)))
8 print('Classification Report:')
9 print(classification_report(y_test, n_pred))
10 print('\n')
11 print('Confusion Matrix:')
12 data=confusion_matrix(y_test, n_pred)
13 sns.heatmap(data,annot=True,cmap="summer")
14 plt.show()
```

Fig 4.9: Gaussian NB Classifier

Table 4.5: Results of Gaussian NB Classifier

	Precision	Recall	F1-score
0	0.62	0.77	0.69
1	0.40	0.22	0.29
2	1.00	0.92	0.96
3	0.94	1.00	0.97
4	1.00	1.00	1.00
5	0.98	0.93	0.95
6	0.95	1.00	0.97
7	0.95	0.98	0.97
Accuracy			0.94
Macro average	0.85	0.85	0.85
Weighted average	0.93	0.94	0.93

4.6 ADA BOOST

```
1 from sklearn.ensemble import AdaBoostClassifier
2 from sklearn import metrics
3 clff = AdaBoostClassifier(n_estimators=100,learning_rate=1,random_state=1).fit(X_train, y_train)
4 print('Accuracy of adaboost classifier on training set: {:.2f}'
5       .format(clff.score(X_train, y_train)))
6 print('Accuracy of adaboost classifier on test set: {:.2f}'
7       .format(clff.score(X_test, y_test)))
8 a_pred=classifier.predict(X_test)
9 print("Classification Report ")
10 print(classification_report(y_test,a_pred))
11 print('\n')
12 data=confusion_matrix(y_test, a_pred)
13 sns.heatmap(data,annot=True,cmap="summer")
14 plt.show()
```

Fig 4.10: Ada Boost Classifier

Table 4.6: Results of Ada Boost Classifier

	Precision	Recall	F1-score
0	0.82	0.69	0.75
1	0.00	0.00	0.00
2	0.88	0.92	0.90
3	0.79	1.00	0.88
4	0.92	1.00	0.96
5	1.00	0.91	0.95
6	1.00	1.00	1.00
7	0.94	1.00	0.97
Accuracy			0.93
Macro average	0.79	0.82	0.80
Weighted average	0.90	0.93	0.91

4.7 COMPARISION TABLE

Table 4.7: Comparison Table for all proposed algorithms

	Decision Tree	Gaussian NB	Random Forest	Ada Boost	Gradient Boost
Accuracy	93.33	93.68	92.5	55	87.7
Precision	93	93	90	90	88
Recall	93	94	93	93	88
F-Score	93	93	91	91	87

CHAPTER 5

CONCLUSION AND FUTURE WORK

Utilizing ML, we could anticipate the most appropriate compost for the given harvest soil n p k qualities assisting the ranchers with expanding the yield. This framework does the forecast utilizing five distinct ML models. Decision Tree and Gaussian NB have shown similarities in F-Score that is 93% but have shown 93.33% and 93.68% respectively in terms of Accuracy. Based on Accuracy and F-score results conclusion has been made that Gaussian NB makes the accurate prediction. Also further predicted the suitable fertilizer through Gaussian NB model when provided with the parameters by the end user. In addition to this work the dataset can be extended by getting more values from the analysis of the soil and extend the work for more crops. A user interface can be developed so that the farmers can directly get their results from the given inputs. Using advanced algorithms in Machine Learning also increase the level of accuracy.

REFERENCES

- [1] Bondre, Devdatta A., and Santosh Mahagaonkar. "Prediction of crop yield and fertilizer recommendation using machine learning algorithms." *International Journal of Engineering Applied Sciences and Technology* 4, no. 5 (2019): 371-376.
- [2] Archana, K., and K. G. Saranya. "Crop Yield Prediction, Forecasting and Fertilizer Recommendation using Voting Based Ensemble Classifier." *SSRG Int. J. Comput. Sci. Eng* 7 (2020).
- [3] Kim, Yun Hwan, Seong Joon Yoo, Yeong Hyeon Gu, Jin Hee Lim, Dongil Han, and Sung Wook Baik. "Crop pests prediction method using regression and machine learning technology: Survey." *IERI Procedia* 6 (2014): 52-56.
- [4] Kalimuthu, M., P. Vaishnavi, and M. Kishore. "Crop Prediction using Machine Learning." In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 926-932. IEEE, 2020.
- [5] Kumar, Y. Jeevan Nagendra, V. Spandana, V. S. Vaishnavi, K. Neha, and V. G. R. R. Devi. "Supervised Machine learning Approach for Crop Yield Prediction in Agriculture Sector." In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pp. 736-741. IEEE, 2020.
- [6] Medar, Ramesh, Vijay S. Rajpurohit, and Shweta Shweta. "Crop yield prediction using machine learning techniques." In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, pp. 1-5. IEEE, 2019.
- [7] Nigam, Aruvansh, Saksham Garg, Archit Agrawal, and Parul Agrawal. "Crop yield prediction using machine learning algorithms." In *2019 Fifth International Conference on Image Information Processing (ICIIP)*, pp. 125-130. IEEE, 2019.
- [8] Jahan, Nusrat, and Rezvi Shahariar. "Predicting fertilizer treatment of maize using decision tree algorithm." *Indonesian Journal of Electrical Engineering and Computer Science* 20, no. 3 (2020): 1427-1434.

- [9] Kanuru, L., Tyagi, A.K., Aswathy, S.U., Fernandez, T.F., Sreenath, N. and Mishra, S., 2021, January. Prediction of Pesticides and Fertilizers using Machine Learning and Internet of Things. In 2021 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-6). IEEE. Materials Science and Engineering, vol. 1022, no. 1, p. 012104. IOP Publishing, 2021.
- [10] Motia, Sanjay, and S. R. N. Reddy. "Method for dataset preparation for soil data analysis in decision support applications." In IOP Conference Series:
- [11] Pandiarajaa, P. "A survey on machine learning and text processing for pesticides and fertilizer prediction." Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12, no. 2 (2021): 2295-2302.
- [12] Maeda, Yuichiro, Taichi Goyodani, Shunsaku Nishiuchi, and Eisuke Kita. "Yield prediction of paddy rice with machine learning." In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), pp. 361-365. The Steering and Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2018

APPENDIX

```
#importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score

df=pd.read_csv("C:/Users/S S Manoj/Downloads/Fertilizer Prediction1.csv")
df.head()
type(df)
df.nunique()
df.shape
df.isnull().sum()
df.dtypes
df.columns,df.dtypes
df['Fertilizer Name'].value_counts().to_dict()
data=df.drop(labels="Fertilizer Name",axis=1)
y=df['Fertilizer Name']
df.hist(figsize = (20,20))
plt.show()
corrmat=df.corr()
fig = plt.figure(figsize = (12,9))
sns.heatmap(corrmat, vmax =.8, square=True)
plt.show()
corrmat
```



```

from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['Fertilizer Name']= label_encoder.fit_transform(df['Fertilizer Name'])
df['Fertilizer Name'].unique()
df['Fertilizer Name'].value_counts().to_dict()
fig_dims = (20, 4)
fig, ax = plt.subplots(figsize=fig_dims)
sns.barplot(x='Moisture',data=df,y='Fertilizer Name')
sns.barplot(x='Temperature',data=df,y='Fertilizer Name')
fig_dims = (20, 4)
fig, ax = plt.subplots(figsize=fig_dims)
sns.barplot(x='Phosphorous',data=df,y='Fertilizer Name')

fig_dims = (20, 4)
fig, ax = plt.subplots(figsize=fig_dims)
sns.barplot(x='Nitrogen',data=df,y='Fertilizer Name')
fig_dims = (20, 4)
fig, ax = plt.subplots(figsize=fig_dims)
sns.barplot(x='Phosphorous',data=df,y='Fertilizer Name')
fig_dims = (20, 4)
fig, ax = plt.subplots(figsize=fig_dims)
sns.barplot(x='Potassium',data=df,y='Fertilizer Name')
data['Crop Type'].value_counts().to_dict()
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
data['Crop Type']= label_encoder.fit_transform(data['Crop Type'])
data['Crop Type'].unique()
data['Crop Type'].value_counts().to_dict()
data['Soil Type'].value_counts().to_dict()

```

```

from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
data['Soil Type']= label_encoder.fit_transform(data['Soil Type'])
data['Soil Type'].unique()
data['Soil Type'].value_counts().to_dict()
from sklearn.model_selection import train_test_split
ip=data
op=df['Fertilizer Name']
X_train,X_test,y_train,y_test=train_test_split(ip,op,test_size=0.7,random_state=1)
X_train
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix
nb = GaussianNB()
nb.fit(X_train, y_train)
n_pred=nb.predict(X_test)
print('Accuracy of GaussianNB classifier on training set: {:.3f}'.format(nb.score(X_train,
y_train)))
print('Accuracy of GaussianNB classifier on test set: {:.3f}'.format(nb.score(X_test,
y_test)))
print('Classification Report:')
print(classification_report(y_test,n_pred))
print('\n')
print('Confusion Matrix:')
data=confusion_matrix(y_test, n_pred)
sns.heatmap(data,annot=True,cmap="summer")
plt.show()
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier().fit(X_train, y_train)
dtc.fit(X_train, y_train)

```

```

d_pred = dtc.predict(X_test)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
      .format(dtc.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
      .format(dtc.score(X_test, y_test)))
from sklearn.metrics import classification_report
k_range = range(1, 10)
scores=[]
for k in k_range:
    clf2 = DecisionTreeClassifier(max_depth = k).fit(X_train, y_train)
    scores.append(clf2.score(X_train, y_train))
print(max(scores))
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
      .format(clf2.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
      .format(clf2.score(X_test, y_test)))
print("Classification Report")
print(classification_report(y_test,d_pred))
print('\n')
data=confusion_matrix(y_test, d_pred)
sns.heatmap(data,annot=True,cmap="summer")
plt.show()
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=50,max_depth=10, random_state=1)
classifier.fit(X_train, y_train)
from sklearn.metrics import classification_report, accuracy_score
print('Accuracy of Random Forest classifier on training set: {:.2f}'
      .format(classifier.score(X_train, y_train)))
print('Accuracy of Random Forest classifier on test set: {:.2f}'

```

```

        .format(classifier.score(X_test, y_test)))
r_pred=classifier.predict(X_test)
print("Classification Report")
print(classification_report(y_test,r_pred))
print('\n')
data=confusion_matrix(y_test, r_pred)
sns.heatmap(data,annot=True,cmap="summer")
plt.show()
from sklearn.ensemble import GradientBoostingClassifier

#Create Gradient Boosting Classifier
gb = GradientBoostingClassifier()

#Train the model using the training sets
gb.fit(X_train, y_train)

#Predict the response for test dataset
g_pred = gb.predict(X_test)
x_pred = gb.predict(X_train)
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy of gradient boost:",metrics.accuracy_score(y_test, g_pred))
print("Classification Report")
print(classification_report(y_test,g_pred))
print('\n')
data=confusion_matrix(y_test, g_pred)
sns.heatmap(data,annot=True,cmap="summer")
plt.show()
from sklearn.ensemble import AdaBoostClassifier

```

```

from sklearn import metrics
clff =
AdaBoostClassifier(n_estimators=100,learning_rate=1,random_state=1).fit(X_train,
y_train)
print('Accuracy of adaboost classifier on training set: {:.2f}'
      .format(clff.score(X_train, y_train)))
print('Accuracy of adaboost classifier on test set: {:.2f}'
      .format(clff.score(X_test, y_test)))
a_pred=classifier.predict(X_test)
print("Classification Report ")
print(classification_report(y_test,a_pred))
print('\n')
data=confusion_matrix(y_test, a_pred)
sns.heatmap(data,annot=True,cmap="summer")
plt.show()
print("Accuracy scores of each model")
print("Decision tree   :",accuracy_score(y_test,d_pred))
print("Random forest   :",accuracy_score(y_test,r_pred))
print("GaussianNB       :",nb.score(X_test, y_test))
print("Adaboost         :",clff.score(X_test, y_test))
print("Gradient Boost   :",accuracy_score(y_test,g_pred))
def one():
    temperature=int(input('Enter temperature'))
    humidity=int(input('Enter humidity'))
    moisture=int(input('Enter moisture'))
    soiltype=int(input('Enter soil Type 0-black 1-clayey 2-loamy 3-red 4-sandy'))
    croptype=int(input('Enter Crop Type 0 :Tobacco 1 :Cotton 2 :Ground Nuts 3 :Maize 4
:millets 5 :oil seeds 6 :pulses 7 :paddy 8 :sugarcane 9 :Barley 10:wheat'))
    nitrogen=int(input('Enter nitrogen value'))

```

```

potassium=int(input('Enter potassium value'))
phosphorous=int(input('Enter phosphorous value'))

temp=nb.predict([[temperature,humidity,moisture,soiltype,croptype,nitrogen,potassium,p
hosphorous]])
return temp

```

```

def two():
    temperature=int(input('Enter temperature'))
    humidity=int(input('Enter humidity'))
    moisture=int(input('Enter moisture'))
    soiltype=int(input('Enter soil Type 0-black 1-clayey 2-loamy 3-red 4-sandy'))
    croptype=int(input('Enter Crop Type 0 :Tobacco 1 :Cotton 2 :Ground Nuts 3 :Maize 4
:millets 5 :oil seeds 6 :pulses 7 :paddy 8 :sugarcane 9 :Barley 10:wheat'))
    nitrogen=int(input('Enter mitrogen value'))
    potassium=int(input('Enter potassium value'))
    phosphorous=int(input('Enter phosphorous value'))

```

```

temp=dtc.predict([[temperature,humidity,moisture,soiltype,croptype,nitrogen,potassium,p
hosphorous]])
return temp

```

```

def three():
    temperature=int(input('Enter temperature'))
    humidity=int(input('Enter humidity'))
    moisture=int(input('Enter moisture'))
    soiltype=int(input('Enter soil Type 0-black 1-clayey 2-loamy 3-red 4-sandy'))

```

```
croptype=int(input('Enter Crop Type 0 :Tobacco 1 :Cotton 2 :Ground Nuts 3 :Maize 4  
:millets 5 :oil seeds 6 :pulses 7 :paddy 8 :sugarcane 9 :Barley 10:wheat'))
```

```
nitrogen=int(input('Enter mitrogen value'))
```

```
potassium=int(input('Enter potassium value'))
```

```
phosphorous=int(input('Enter phosphorous value'))
```

```
temp=classifier.predict([[temperature,humidity,moisture,soiltype,croptype,nitrogen,potas  
sium,phosphorous]])
```

```
return temp
```

```
def four():
```

```
temperature=int(input('Enter temperature'))
```

```
humidity=int(input('Enter humidity'))
```

```
moisture=int(input('Enter moisture'))
```

```
soiltype=int(input('Enter soil Type 0-black 1-clayey 2-loamy 3-red 4-sandy'))
```

```
croptype=int(input('Enter Crop Type 0 :Tobacco 1 :Cotton 2 :Ground Nuts 3 :Maize 4  
:millets 5 :oil seeds 6 :pulses 7 :paddy 8 :sugarcane 9 :Barley 10:wheat'))
```

```
nitrogen=int(input('Enter mitrogen value'))
```

```
potassium=int(input('Enter potassium value'))
```

```
phosphorous=int(input('Enter phosphorous value'))
```

```
temp=clff.predict([[temperature,humidity,moisture,soiltype,croptype,nitrogen,potassium,  
phosphorous]])
```

```
return temp
```

```
def five():
```

```
temperature=int(input('Enter temperature'))
```

```
humidity=int(input('Enter humidity'))
```

```

moisture=int(input('Enter moisture'))
soiltype=int(input('Enter soil Type 0-black 1-clayey 2-loamy 3-red 4-sandy'))
croptype=int(input('Enter Crop Type 0 :Tobacco 1 :Cotton 2 :Ground Nuts 3 :Maize 4
:millets 5 :oil seeds 6 :pulses 7 :paddy 8 :sugarcane 9 :Barley 10:wheat'))
nitrogen=int(input('Enter nitrogen value'))
potassium=int(input('Enter potassium value'))
phosphorous=int(input('Enter phosphorous value'))

temp=gb.predict([[temperature,humidity,moisture,soiltype,croptype,nitrogen,potassium,p
hosphorous]])
return temp

switcher = {
    1: one,
    2: two,
    3: three,
    4: four,
    5: five
}

print("1:Guassian Naive bayes 2:Decision Tree 3:randomForest")
print("4:Adaboost 5:GradientBoost")
n=int(input('Enter model'))
func =switcher.get(n)
print(func())

```