

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belagavi-590 018, Karnataka



21INT68 -Innovation/Entrepreneurship /Societal Internship

“CROP RECOMMENDATION SYSTEM”

**Submitted in partial fulfillment of the requirements for the award of the degree of
Bachelor of Engineering**

In

Computer Science & Engineering

Submitted by

1BI21CS186

ASHISH A ANKAM

Under the guidance of

Prof. Manjushree N S

Assistant Professor

Department of CSE

BIT



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
BANGALORE INSTITUTE OF TECHNOLOGY**

K. R. Road, V. V. Puram, Bengaluru - 560 004

2023-2024



An ISO 9001:2015 Certified Robotics and Automation Training Institute

iObrain Innovative LabZ



Ref: iObrain/intern/100-01/24

Dated: 24-01-2024

CERTIFICATE OF INTERNSHIP

This is to certify that "ASHISH A ANKAM" USN: 1BI21CS186,

5th Semester from dept. of "Computer Science and Engineering from

"BANGALORE INSTITUTE OF TECHNOLOGY"

has successfully completed an **Internship Programme** on

"DATA Analysis using Python" along with the live projects

for 4 weeks from 25-10-23 to 23-11-23.

Resource Person: Dr. Nagarajan Srinivasan



We wish the student every success in career and future endeavors.

Ramakrishna K S



CEO, iObrain



START WITH US TODAY...STAND FOR TOMORROW

Call: +91 93536 94456

Mail: iobrainlabz@gmail.com

web: www.iobrain.in

#786/A, 1st Floor, First Cross 2nd Phase, 7th Main, Banahsankari 3rd Stage, Bangalore - 560 085.

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belagavi-590 018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY

K R Road, V V Puram, Bengaluru-560 004



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate

Certified that the **21INT68-Innovation/Entrepreneurship/Societal Internship** (21INT68) work entitled “Crop Recommendation System” carried out by Mr. Ashish A Ankam USN 1BI21CS186 a bonafide student of Bangalore Institute of Technology in partial fulfillment for the award of Bachelor of Engineering in Computer Science & Engineering of the **Visvesvaraya Technological University, Belagavi** during the academic year 2023-2024. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library.

The Internship report has been approved as it satisfies the academic requirements in respect of Internship work prescribed for the said degree.

Guide Name

Prof. Manjushree N S

Assistant Professor

Department of CSE, BIT

Dr J Girija

Professor and Head

Department of CSE, BIT

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without complementing those who made it possible and whose guidance and encouragement made my efforts successful. So, my sincere thanks to all those who have supported me in completing this Internship successfully.

My sincere thanks to **Dr. M. U. Aswath**, Principal, BIT and **Dr. J. Girija**, HOD, Department of CSE, BIT for their encouragement, support and guidance to the student community in all fields of education. I am grateful to our institution for providing us a congenial atmosphere to carry out the Internship successfully.

I would not forget to remember **Dr. Bhanushree K J**, Associate Professor **21INT68-Innovation/Entrepreneurship/Societal** Internship Coordinator, Department of CSE, BIT, for her encouragement and more over for her timely support and guidance till the completion of the Internship.

I avail this opportunity to express my profound sense of deep gratitude to my esteemed guide **Prof. Manjushree N S** Assistant Professor, Department of CSE,BIT, for her moral support, encouragement and valuable suggestions throughout the Internship.

I extend my sincere thanks to all the department faculty members and non- teaching staff for supporting me directly or indirectly in the completion of this Internship.

NAME: Ashish A Ankam

USN: 1BI21CS186

TABLE OF CONTENTS

Sl.no	Description		Page No
1	Chapter 1 – Introduction		1-4
	1.1	Overview	1
	1.2	Objective	1
	1.3	Purpose, Scope and applicability	2-3
	1.3.1	Purpose	2
	1.3.2	Scope	2
	1.3.3	Applicability	3
	1.4	Organization of Report	4
2	Chapter 2 - Problem Statement		5-11
3	Chapter 3 - Algorithm		12-17
4	Chapter 4 - Tools/Technologies		18-22
5	Chapter 5 - Implementation		23-33
	5.1	Source code	24-32
6	Chapter 6 - Results		34-35
7	Chapter 7 - Reflection Notes		36-37
8	Chapter 8 – References		38
	Internship Certificate		

LIST OF FIGURES

Figure No.	Description	Page No.
2.1	Input values in .csv file	5
2.2	Input values to be given in streamlit UI	5
2.3	Input values to be given in Flask UI	6
2.4	Output provided in Streamlit UI	6
2.5	Output provided in Flask UI	6
2.6	Boxplot of an input variable	7
2.7	An heatmap representing probability distribution of input data	7
2.8	Classification report of the accuracy of the model	8
2.9	Confusion Matrix	8
2.10	Barplot of model's accuracy	9
2.11	Output labels versus input variable (humidity)	9
2.12	Pairplot	10
2.13	Plot of Algorithms with Accuracy (in %)	11
2.14	Plot of Algorithms with Accuracy (in %)	11
3.1	A Pictorial Representation of Random Forest Classifier	12
3.2	A Pictorial Representation of Decision Tree Classifier	14
3.3	A Graphical Representation of Decision Tree Classifier	15
3.4	A Pictorial Representation of SVM Classifier	16
3.5	A Graphical Representation of SVM Classifier	17
4.1	A Symbolic Representation of the Machine Learning Process.	21
6.1	Output seen in streamlit UI	34
6.2	Model provides the required answer as per the input values.	34
6.3	Output seen in Flask UI	34
6.4	Classification Report about Model's Accuracy.	35
6.5	Confusion Matrix in Visualised manner done by Seaborn library.	35
6.6	Output as seen in Flask UI.	35

Chapter 1

Introduction

Chapter 1

INTRODUCTION

1.1 Overview

- The agricultural sector plays a pivotal role in sustaining global food security and economic stability.
- With the increasing challenges posed by climate change, soil degradation, and a growing population, there is a pressing need for innovative technologies to enhance agricultural productivity.
- In modern agriculture, the integration of technology has significantly transformed traditional farming methods.
- One such advancement is the development of crop recommendation systems powered by machine learning algorithms.
- These systems analyze various factors such as soil properties, climate conditions, and historical crop data to suggest the most suitable crops for a particular region or farm.
- This system leverages advanced algorithms to analyze various environmental and soil parameters, providing farmers with personalized crop recommendations to optimize yield and resource utilization.

1.2 Objective

- The primary goal of this project is to design and implement a Crop Recommendation System that assists farmers in selecting the most suitable crops based on their specific geographical and environmental conditions.
- The system aims to improve crop yield, resource efficiency, and overall farm sustainability.
- This project aims to provide an overview of the crop recommendation system implemented using Python and machine learning techniques.
- It will cover the rationale behind such systems, the methodology employed, and potential benefits for farmers and agricultural stakeholders.

1.3 Purpose, Scope and Applicability

1.3.1 Purpose

Crop recommendation system projects using machine learning serve several important purposes in the agricultural sector:

Here are some key uses and purposes of crop recommendation system:

- **Increased Crop Yield and Productivity:** By recommending suitable crop varieties and planting seasons, the system can help farmers optimize their land use and maximize their output.
- **Reduced Input Costs and Resource Optimization:** By recommending crops with similar water or nutrient requirements, the system can help farmers efficiently manage their irrigation and fertilization strategies.
- **Improved Decision-Making and Risk Management:** By predicting potential challenges based on historical data and weather forecasts, the system can help farmers prepare and adopt preventative measures.
- **Sustainability and Environmental Benefits:** By recommending crops suited to local soil conditions and minimizing resource use, the system can promote sustainable agricultural practices.
- Crop recommendation systems can connect farmers with markets and potential buyers, facilitating better price realization for their produce.

1.3.2 Scope

The scope should be clearly defined, considering resources, technical expertise, and project goals. Thus, here are some of the key features related to the project:

1. **Individual farmers:** Systems designed for individual farmers might focus on recommending crops for specific plots based on local data and user inputs.
2. **Agricultural organizations or communities:** Broader scope could involve regional recommendations considering market demands, resource availability, and environmental factors.
3. **Limited data:** Projects might start with simpler models using easily accessible data like soil type and historical yields.
4. **Large datasets:** Complex models incorporating sensor data, weather forecasts, and economic parameters may be employed.
5. **Basic recommendation:** Suggesting suitable crops based on input parameters.
6. **Simple web application:** User-friendly interface for input and receiving recommendations.

1.3.3 Applicability

- Crop recommendation systems can help farmers **optimize the use of resources** such as water, fertilizers, and pesticides.
- By recommending crops that require specific inputs based on local conditions, **farmers can reduce waste** and improve resource efficiency.
- These systems can help farmers mitigate **risks** associated with weather **variability, pests, and diseases**.
- By recommending resilient crop varieties or suggesting appropriate preventive measures, **farmers can better protect their crops from adverse conditions**.
- Crop recommendation systems can assist in **planning crop rotations** to improve soil health and fertility.
- By suggesting rotations that **break pest and disease cycles** and enhance nutrient levels, farmers can maintain a more sustainable and productive agricultural system.
- Integrating crop recommendation systems with other technologies such as **precision agriculture tools, drones, and sensors allows for real-time monitoring** and adjustment of cultivation practices.
- This enhances **the overall efficiency and effectiveness** of farming operations.
- Agricultural extension services can use the recommendation system to **disseminate information and best practices to farmers**, fostering knowledge-sharing and community support.
- The system can promote sustainable farming practices by recommending crops that are well-suited to the local environment, **reducing the need for excessive inputs and minimizing environmental impact**.
- Designing the **system to be user-friendly** and accessible to farmers with varying levels of technological expertise can ensure widespread adoption and impact.
- The system can take into account the specific preferences and constraints of individual farmers, **providing personalized recommendations** based on their unique circumstances and goals.
- By recommending crops with higher market demand or better price stability, **the system can assist farmers in making financial decisions** and planning their crop cycles for improved profitability.

1.3.4 Organization of Report

Chapter 2: It defines the problem statement, the input parameters as well as expected output, results generated with valid input as well as the desired outcome from the problem statement.

Chapter 3: This includes the components of the crop recommendation system. The project showcases the realistic aspect of the problem statement. The program is also modified into an application having a user interface for better communication purpose.

Chapter 4: This section of the report comprises of the Tools/Technologies used in the internship work are explained in detail using screenshots. The explanation includes the purpose and reasons why particular tools/technologies are used, important features and other details like versions, open source or not, compatible or not and so on.

Chapter 5: In this section, the implementation is explained. Important functionalities are explained along with the source code used to solve the given problem. Further details such as code requirements, and concepts used along with code explanation and working of the code is present.

Chapter 6: The results and desired outputs snippets with proper naming and present in a proper sequence are provided in this chapter.

Chapter 7: It depicts the reflection notes on how the internship project duration went on.

Chapter 2

Problem Statement

Chapter 2

PROBLEM STATEMENT

The aim of the given project is to develop a simple crop recommendation system using the concepts of Machine Learning.

Input: (1)

	A	B	C	D	E	F	G	H
1	N	P	K	temperature	humidity	ph	rainfall	label
2	90	42	43	20.87974371	82.00274423	6.502985292	202.9355362	rice
3	85	58	41	21.77046169	80.31964408	7.038096361	226.6555374	rice
4	60	55	44	23.00445915	82.3207629	7.840207144	263.9642476	rice
5	74	35	40	26.49109635	80.15836264	6.980400905	242.8640342	rice
6	78	42	42	20.13017482	81.60487287	7.628472891	262.7173405	rice
7	69	37	42	23.05804872	83.37011772	7.073453503	251.0549998	rice
8	69	55	38	22.70883798	82.63941394	5.70080568	271.3248604	rice
9	94	53	40	20.27774362	82.89408619	5.718627178	241.9741949	rice
10	89	54	38	24.51588066	83.5352163	6.685346424	230.4462359	rice
11	68	58	38	23.22397386	83.03322691	6.336253525	221.2091958	rice
12	91	53	40	26.52723513	81.41753846	5.386167788	264.6148697	rice
13	90	46	42	23.97898217	81.45061596	7.50283396	250.0832336	rice
14	78	58	44	26.80079604	80.88684822	5.108681786	284.4364567	rice
15	93	56	36	24.01497622	82.05687182	6.98435366	185.2773389	rice


Figure 2.1: Input values in .csv file

A sample input values consisting of 7 columns and 2200 rows among which 6 rows labelled are input values and last row is the outcome variable.

Input: (2)

Figure 2.2: Input values to be given in streamlit UI

Input: (3)



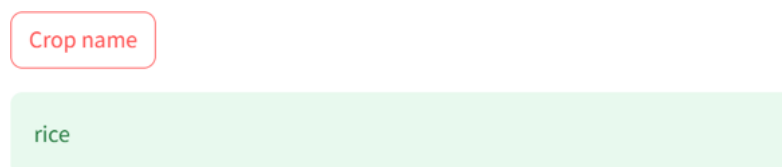
The image shows a web form titled "Crop Recommendation System" with a small plant icon. The form contains several input fields for agricultural data: Nitrogen, Phosphorus, Potassium, Temperature, Humidity, pH, and Rainfall. Each field has a placeholder text indicating the unit or type of input. A blue button labeled "Get Recommendation" is positioned at the bottom center of the form.

Input Field	Placeholder Text
Nitrogen	Enter Nitrogen
Phosphorus	Enter Phosphorus
Potassium	Enter Potassium
Temperature	Enter Temperature in °C
Humidity	Enter Humidity in %
pH	Enter pH value
Rainfall	Enter Rainfall in mm

Get Recommendation

Figure 2.3: Input values to be given in Flask UI

Output: (1)



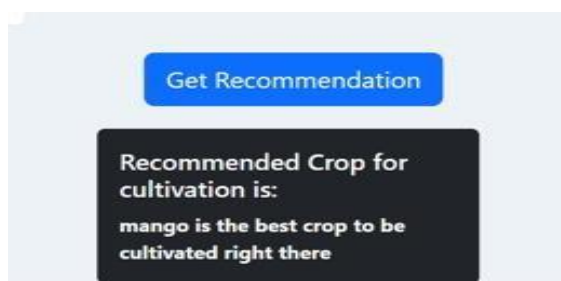
The image shows a Streamlit UI output. It features a red-bordered box labeled "Crop name" in red text. Below this, a green box displays the word "rice" in green text.

Crop name

rice

Figure 2.4: Output provided in Streamlit UI

Output: (2)



The image shows a Flask UI output. It features a blue button labeled "Get Recommendation" at the top. Below the button, a dark gray box contains the text "Recommended Crop for cultivation is:" followed by "mango is the best crop to be cultivated right there" in white text.

Get Recommendation

Recommended Crop for cultivation is:
mango is the best crop to be cultivated right there

Figure 2.5: Output provided in Flask UI

Boxplots of Input data to detect outliers’:

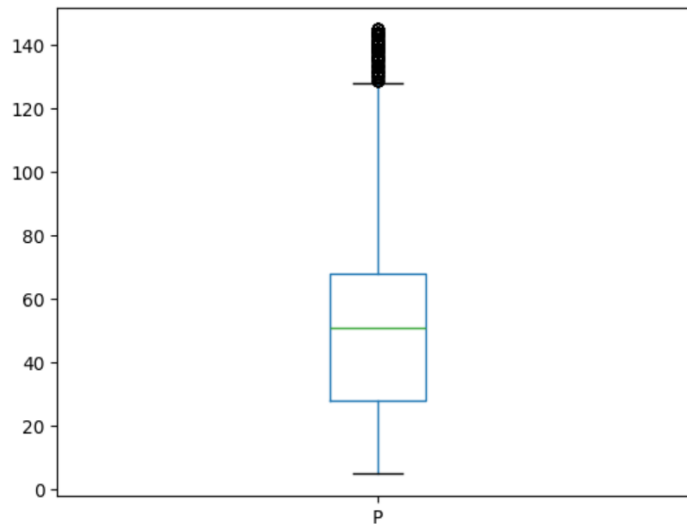


Figure 2.6: Boxplot of an input variable

Correlation heatmap of input variables’:

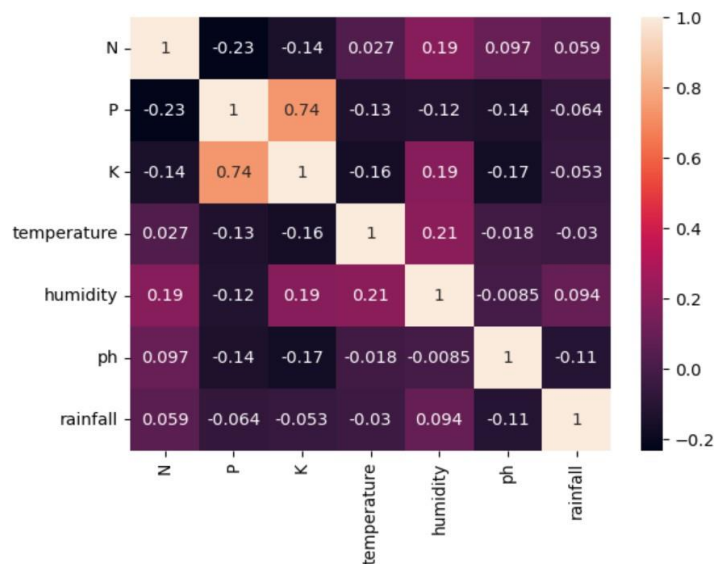


Figure 2.7: An heatmap representing probability distribution of input data

Classification report of score of the ML model about the prediction:

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	25
banana	1.00	1.00	1.00	25
blackgram	1.00	1.00	1.00	25
chickpea	1.00	1.00	1.00	25
coconut	1.00	1.00	1.00	25
coffee	1.00	1.00	1.00	25
cotton	1.00	1.00	1.00	25
grapes	1.00	1.00	1.00	25
jute	0.89	1.00	0.94	25
kidneybeans	1.00	1.00	1.00	25
lentil	1.00	1.00	1.00	25
maize	1.00	1.00	1.00	25
mango	1.00	1.00	1.00	25
mothbeans	1.00	1.00	1.00	25
mungbean	1.00	1.00	1.00	25
muskmelon	1.00	1.00	1.00	25
orange	1.00	1.00	1.00	25
papaya	1.00	1.00	1.00	25
pigeonpeas	1.00	1.00	1.00	25
pomegranate	1.00	1.00	1.00	25
rice	1.00	0.88	0.94	25
watermelon	1.00	1.00	1.00	25

Figure 2.8: Classification report of the accuracy of the model

Confusion Matrix about the truthness of the input variables:

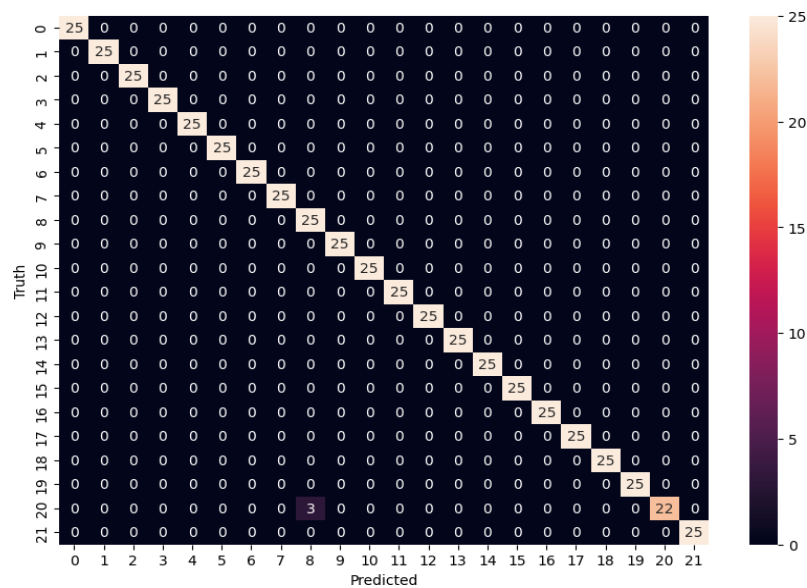


Figure 2.9: Confusion Matrix

Accuracy Comparison using Barplot:

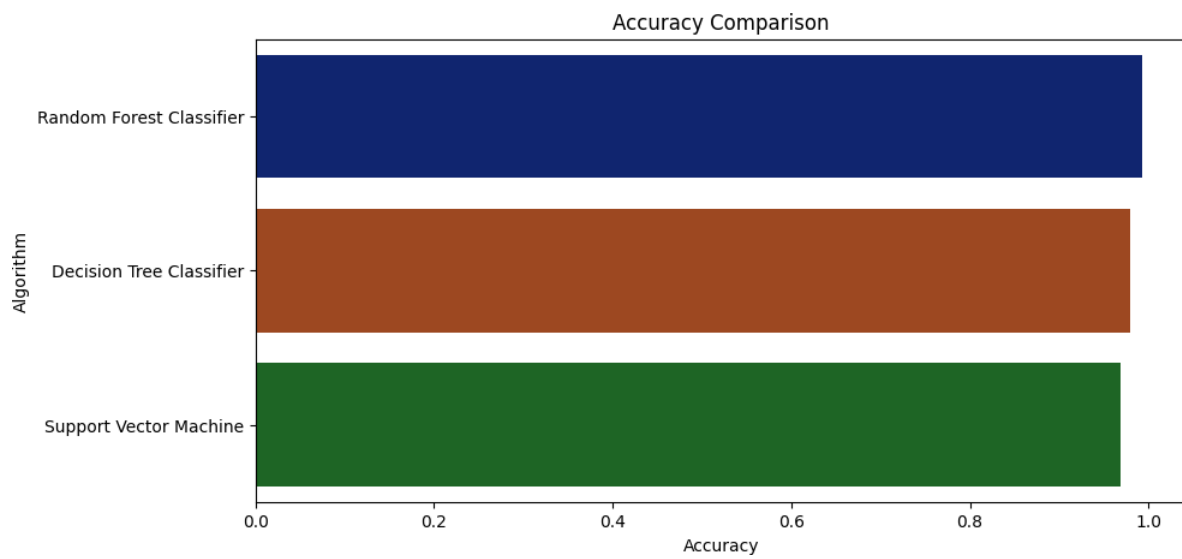


Figure 2.10: Barplot of model's accuracy

Barplot of output labels versus input variable (humidity):

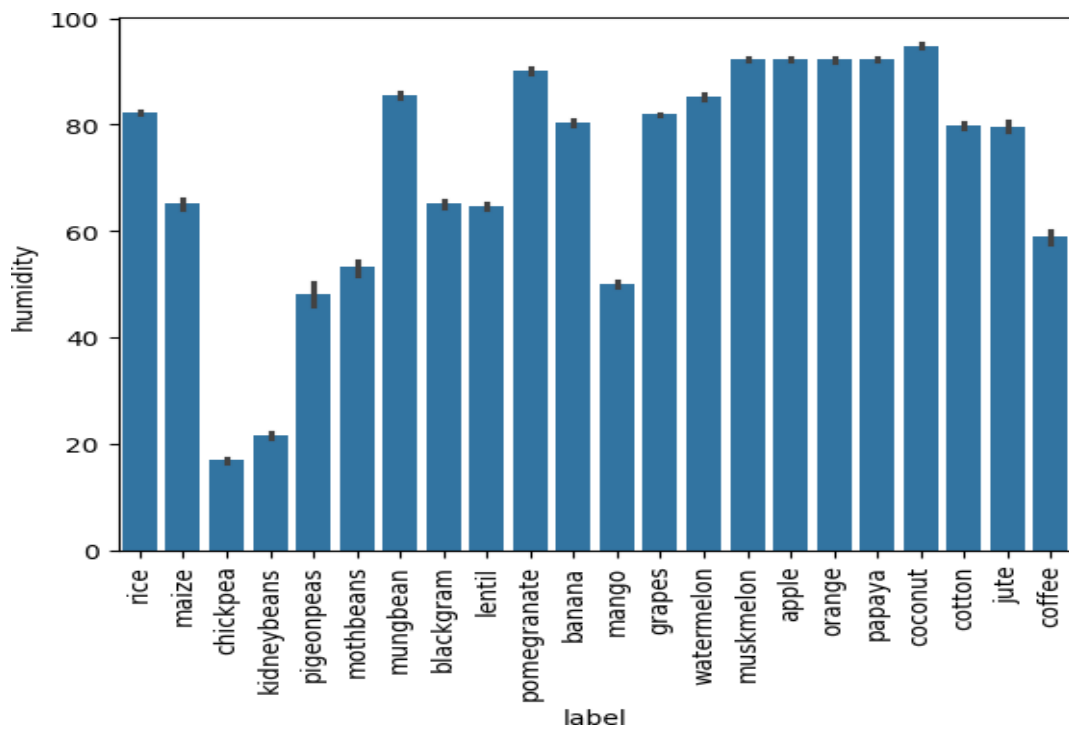


Figure 2.11: Output labels versus input variable (humidity)

Pairplot of all the input values:

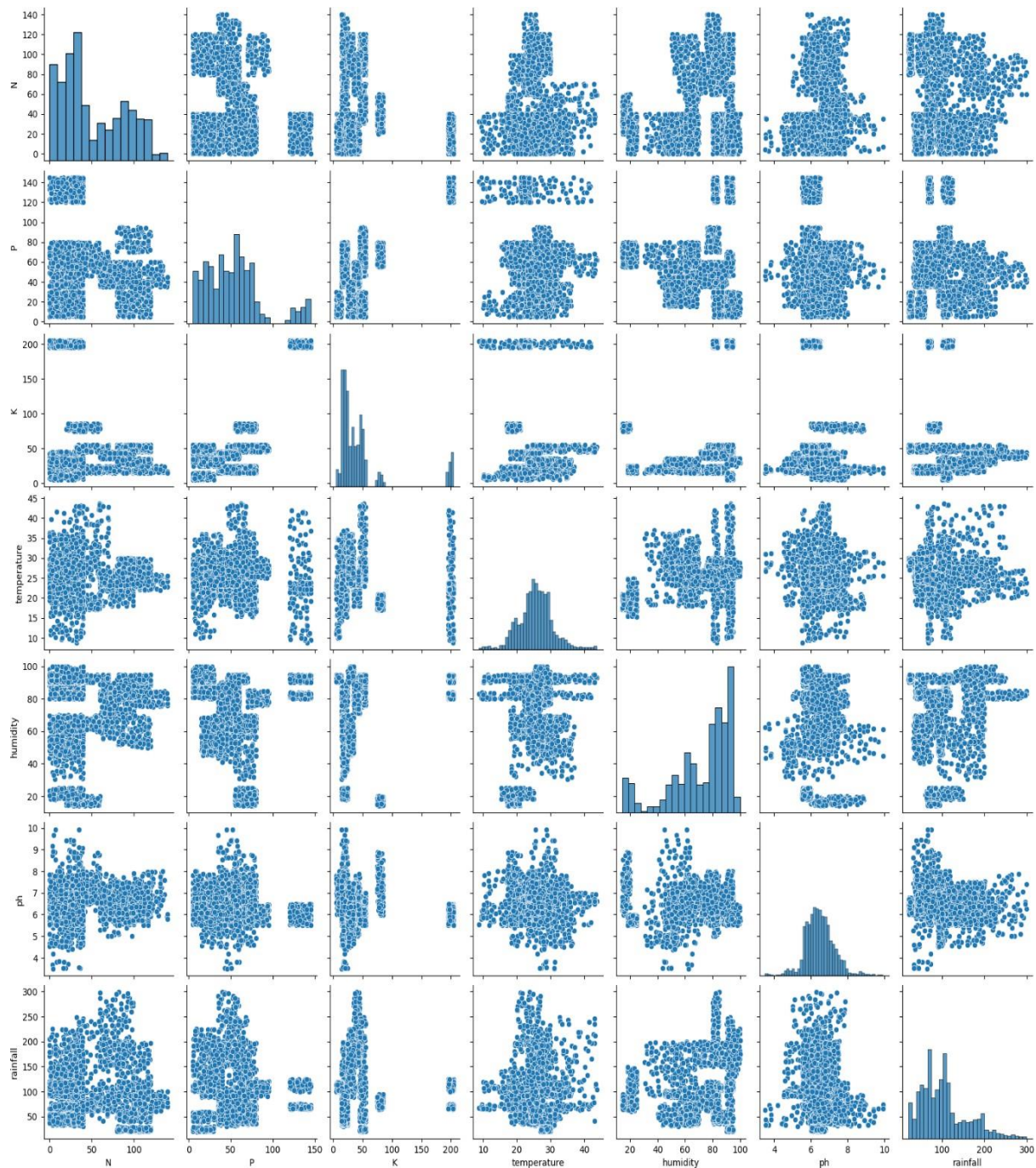


Figure 2.12: Pairplot (A multivalued plot showing comparison between all the input variables)

Plots of Machine Learning Algorithms used with Accuracy(in %):

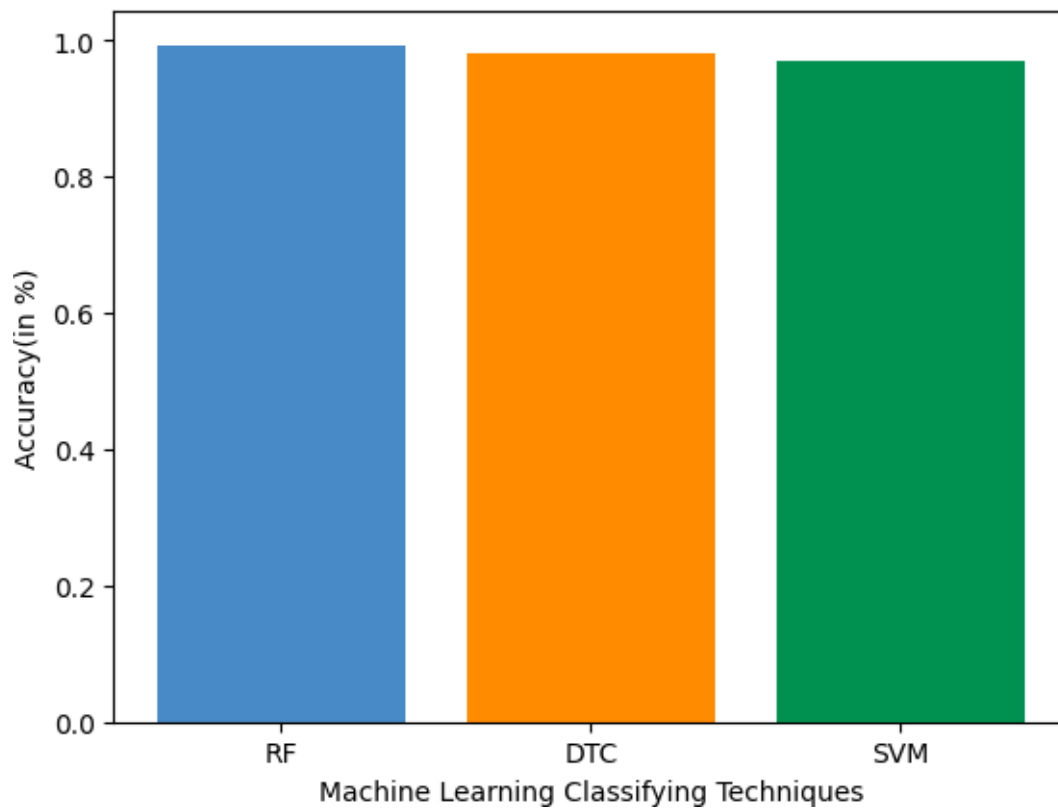


Figure 2.13: Plot of Algorithms with Accuracy(in %)

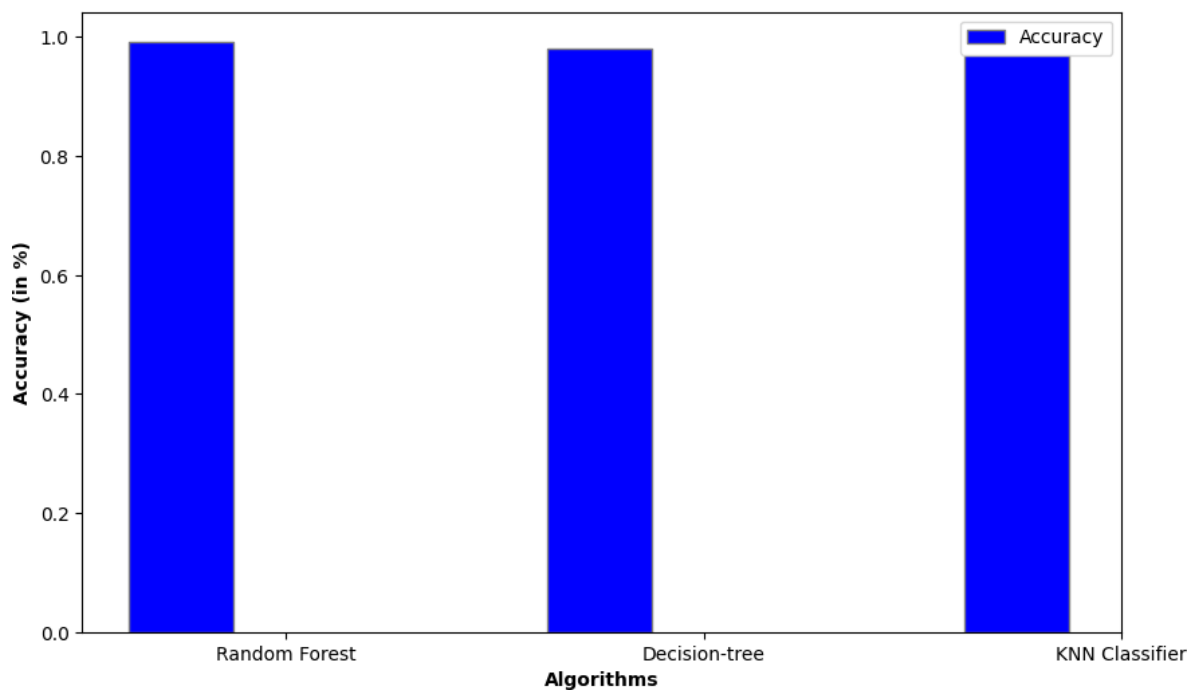


Figure 2.14: Plot of Algorithms with Accuracy(in %)

Chapter 3

Algorithm

Chapter 3

ALGORITHM

There are many algorithms that we can use for training a Machine Learning model. According to our requirements such as Classification or Clustering or Regression, algorithms are being classified. As this is a Recommendation System project, I have used three such algorithms, namely, Random Forest Classifier, Decision Tree Classifier and Support Vector Machine. I shall explain the same in detail in future concepts.

3.1 Random Forest Classifier

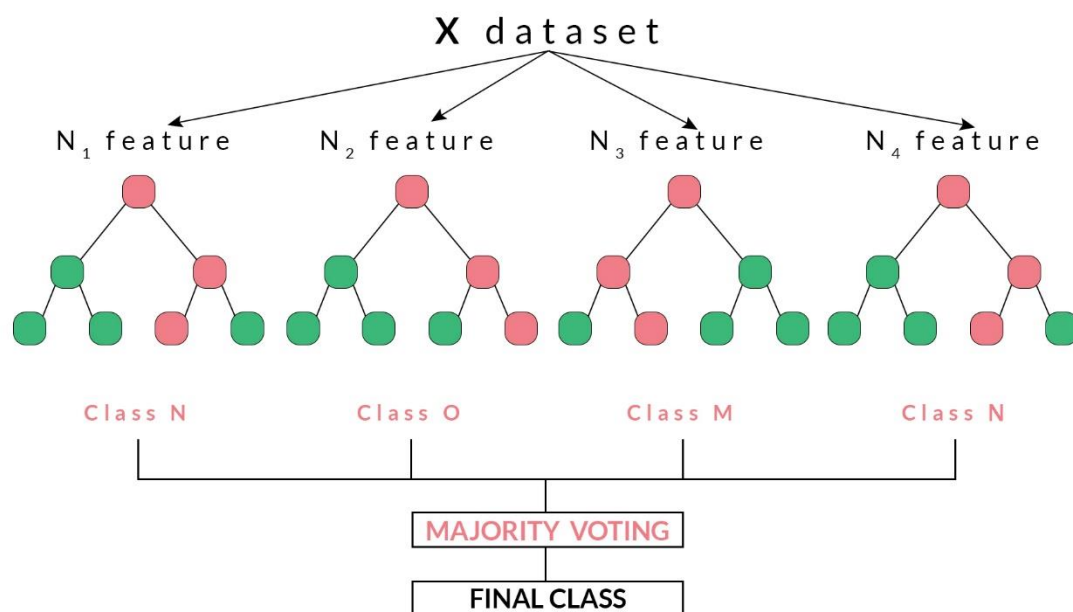


Figure 3.1: A Pictorial Representation of Random Forest Classifier

- Random Forest is built on the foundation of decision trees. A decision tree is a flowchart-like structure where each internal node represents a decision based on a feature, each branch represents the outcome of the decision, and each leaf node represents the final prediction.
- Random Forest creates multiple decision trees by training each tree on a different subset of the training data.

- During training, a random subset of the data (with replacement) is sampled for each tree. This process is known as bootstrapping.
- In addition to sampling data, Random Forest introduces randomness by considering only a random subset of features at each split in the decision tree.
- This helps in decorrelating the trees and ensures that each tree in the forest is diverse.
- In addition to sampling data, Random Forest introduces randomness by considering only a random subset of features at each split in the decision tree.
- After training the individual trees, the Random Forest aggregates their predictions.
- For classification tasks, it uses a majority voting mechanism to determine the final predicted class.
- For regression tasks, it calculates the average of the predictions made by each tree.
- **Ensemble Learning:** Random Forest combines the predictions of multiple models, leading to improved generalization and robustness.
- **Reduction of Overfitting:** The randomness introduced during both data and feature selection helps mitigate overfitting.
- **Feature Importance:** Random Forest provides a measure of feature importance based on how much each feature contributes to the model's predictive performance.
- Random Forest has several hyperparameters that can be tuned, such as the number of trees in the forest, the maximum depth of each tree, and the minimum number of samples required to split a node.
- Grid search or randomized search can be employed to find the optimal set of hyperparameters.
- Random Forest is widely used in various domains, including finance, healthcare, and image analysis.
- It is effective when dealing with large datasets and high-dimensional feature spaces.
- Random Forest is known for its versatility, ease of use, and ability to handle a wide range of data types.
- Its robustness makes it a popular choice for many machine learning tasks.

3.2 Decision Tree Classifier

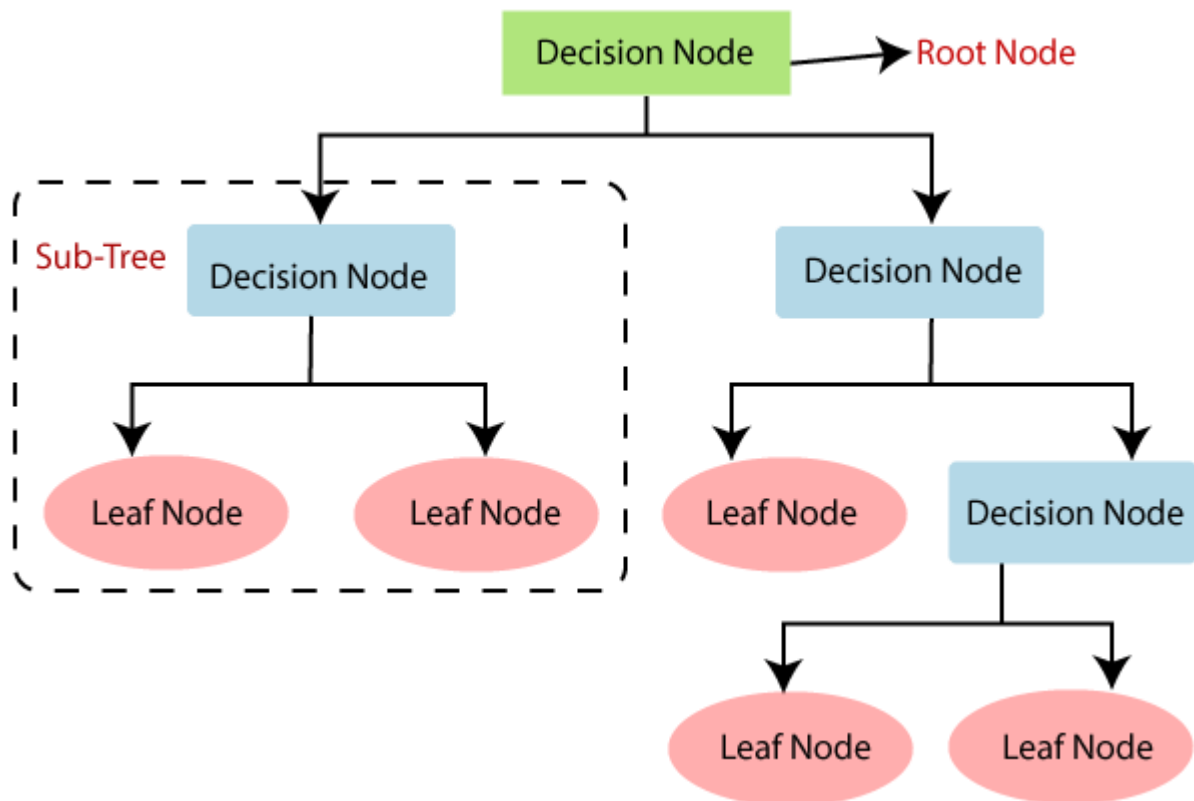


Figure 3.2: A Pictorial Representation of Decision Tree Classifier

- Decision trees are a popular machine learning algorithm used for both classification and regression tasks.
- A decision tree is a tree-like model where each node represents a decision or test on an attribute, each branch represents an outcome of that test, and each leaf node represents a class label or numerical value.
- Decision trees make decisions based on splitting data at each node. The criteria for splitting are chosen to maximize information gain (for classification) or variance reduction (for regression).
- The process of splitting continues recursively until a stopping condition is met, resulting in a tree structure.
- Root Node: The topmost node in the tree, representing the best feature to split the data.
- Internal Nodes: Nodes that represent a decision or test on a feature.

- Leaf Nodes: Terminal nodes that represent the final predicted outcome or class.
- Entropy is a measure of impurity in a set of data. Decision trees aim to reduce entropy with each split.
- Information Gain quantifies the reduction in entropy after a split. Features with higher information gain are preferred for splitting.
- Decision trees are prone to overfitting, capturing noise in the training data. Pruning is a technique to prevent overfitting by removing branches that do not contribute significantly to predictive accuracy.
- Decision trees can handle both categorical and numerical features. For categorical features, the algorithm uses methods like one-hot encoding and for, or numerical features, the algorithm selects optimal thresholds for splitting.
- One of the strengths of decision trees is their interpretability. The decision-making process is easy to understand and visualize.
- Decision trees provide a natural way to measure the importance of different features in making predictions.
- Decision trees are widely used in various fields such as finance, healthcare, and marketing for tasks like fraud detection, medical diagnosis, and customer segmentation.

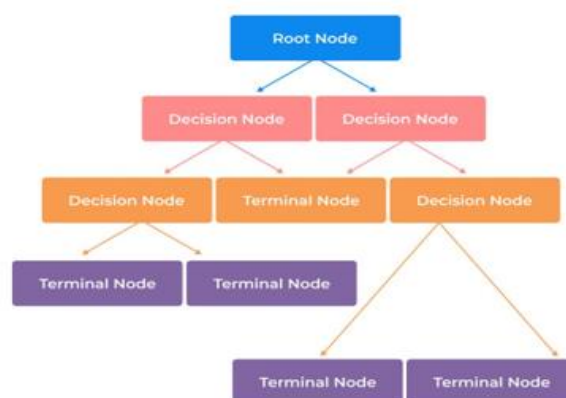


Figure 3.3: A Graphical Representation of Decision Tree Classifier

3.3 Support Vector Machine

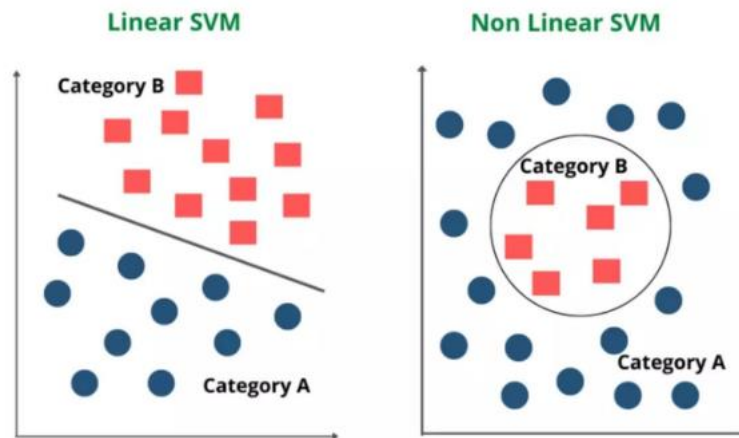


Figure 3.4: A Pictorial Representation of SVM Classifier

- SVM is a supervised machine learning algorithm used for classification and regression.
- It aims to find the optimal hyperplane that maximally separates data into different classes.
- It is effective for linearly separable data. The hyperplane is a high-dimensional decision boundary that separates classes.
- Margin is the distance between the hyperplane and the nearest data point of any class. Support Vectors are the critical data points defining the margin.
- SVM handles non-linear data through kernel functions (e.g., polynomial, RBF) that map data to higher dimensions.
- C parameter balances margin width and classification errors. Small C values prioritize a wider margin, larger C values result in a narrower margin.
- It inherently supports binary classification, extended to multi-class using techniques like One-vs-One or One-vs-All.
- Used in image classification, text categorization, bioinformatics, finance, and other fields.

- Parameters like kernel choice, C parameter, and kernel-specific parameters require tuning.
- Cross-validation aids in parameter optimization.
- SVM may face challenges with large datasets and high dimensionality.
- Selection of the appropriate kernel and tuning parameters is critical.
- Hard Margin aims for a strict separation without misclassifications.
- Soft Margin allows some misclassifications to accommodate noisy or overlapping data.
- In SVM, the regularization parameter C controls the trade-off between achieving a smooth decision boundary and classifying training points correctly.
- Smaller C values lead to a wider margin but may allow more misclassifications, while larger C values result in a narrower margin but fewer misclassifications.

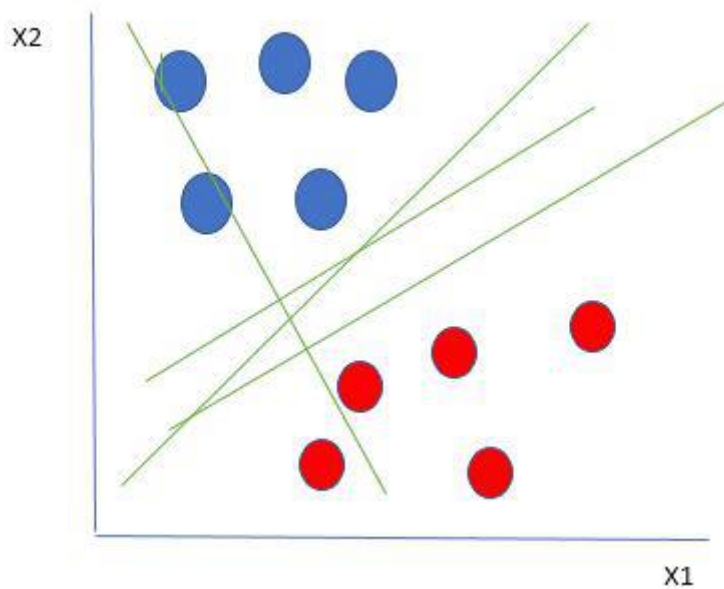


Figure 3.5: A Graphical Representation of SVM Classifier

Chapter 4

Tools/Technologies

Chapter 4

TOOLS/TECHNOLOGIES

During the internship to enhance the crop recommendation system, I have made use of various tools and technologies to facilitate development, testing, and use cases of this project. Below are some of the essential tools and technologies that I have employed during the internship to complete this project:

4.1 Google Colab (Colaboratory):

Google Colab (Colaboratory) is a cloud-based platform provided by Google that allows us to write and execute Python code in a web-based environment.

It's also known as Cloud-based Jupyter Notebook. It's particularly popular in the machine learning community due to several advantages:

1. Google Colab provides free access to Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). This is crucial for training deep learning models, as these hardware accelerators significantly speed up the training process.
2. This allows users to accelerate training of machine learning models by leveraging these specialized hardware accelerators.
3. Colab is seamlessly integrated with Google Drive. We can save our Colab notebooks directly to our Google Drive, making it easy to share and access our work from any device with an internet connection.
4. Users can install additional libraries using the `!pip install` command directly within the notebook.
5. Colab comes pre-installed with many machine learning libraries and frameworks such as TensorFlow, PyTorch, Keras, and scikit-learn. This saves time and effort in setting up the environment.
6. Colab is hosted on the cloud, eliminating the need for users to set up their own local environments. This makes it easy for teams to collaborate on machine learning projects as they can work on the same notebook simultaneously.
7. Colab supports interactive data visualization with libraries like Matplotlib and Seaborn. You can generate plots and charts directly within the notebook, making it easy to analyze and visualize your machine learning results.

4.2 Programming Language – Python:

1. Python has a rich ecosystem of libraries and frameworks specifically designed for machine learning and data science. Prominent examples include TensorFlow, pytorch, scikit-learn, keras, and Numpy.
2. Python has a large and active community of developers, data scientists, and machine learning practitioners. This community support is invaluable for sharing knowledge, discussing best practices, and solving problems collaboratively.
3. Python's syntax is clear, readable, and straight forward. This makes it an excellent language for beginners and facilitates collaboration among team members.
4. Many important machine learning tools and libraries in python are open source. This openness fosters collaboration, innovation, and the sharing of knowledge within the community.
5. For example, Python can be used for web development, and machine learning models implemented in Python can be integrated into web applications.

4.3 Scikit-Learn for Building Model:

Scikit-learn is a powerful machine learning library for Python that provides simple and efficient tools for data analysis and modeling.

1. It offers a comprehensive set of tools for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, and more.
2. It is designed with a simple and consistent interface, making it easy for users to implement machine learning models. This consistency helps users focus on the task at hand rather than dealing with complex syntax.
3. The library includes efficient implementations of a wide range of machine learning algorithms, such as Support Vector Machines (SVM), Decision Trees, Random Forests, k-Means, and more. These implementations are optimized for performance and scalability.
4. It is built on top of two other popular Python libraries: NumPy (for numerical computing) and SciPy (for scientific computing). This ensures seamless integration with other scientific computing tools.
5. It provides tools for feature extraction and preprocessing, including methods for handling missing data, scaling features, encoding categorical variables, and more.
6. The library supports various cross-validation techniques, enabling users to assess the performance of their models and reduce the risk of overfitting.

4.4 Numpy, Pandas, Matplotlib, Seaborn for Data Analysis:

1. NumPy provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. They are implemented in C and Fortran.
2. Pandas is a powerful library for data manipulation and analysis. It provides data structures like dataframes, which are efficient for working with structured data. This is crucial in the data preprocessing phase of Machine Learning.
3. Matplotlib is a versatile plotting library that enables the creation of a wide variety of static, animated, and interactive visualizations. Visualization is critical in ML for exploring data, understanding patterns, and communicating results.
4. Seaborn is built on top of Matplotlib and provides a high-level interface for creating informative and attractive statistical graphics. It simplifies the creation of complex visualizations, such as heatmaps, pair plots, and violin plots.
5. Pandas allows for the integration of various data formats, making it easy to read and write data from/to different sources like CSV, Excel, SQL databases, etc.

4.5 Streamlit and Flask for Model Deployment:

1. Streamlit is designed for creating web applications with minimal effort. It allows data scientists and ML practitioners to rapidly prototype and deploy interactive data applications without requiring extensive web development experience.
2. It provides widgets (sliders, buttons, text inputs, etc.) that make it easy to add user interactivity to ML applications.
3. Its simplicity makes it an excellent choice for deploying machine learning models as web applications. It helps bridge the gap between the data science and software development aspects of a project.
4. Flask is a micro web framework that is more general-purpose than Streamlit. It gives developers more control over the structure of their web applications and is suitable for building more complex and customized applications.
5. It can be integrated with frontend frameworks like React, enabling the development of feature-rich and highly customized web applications.
6. It is well-suited for building scalable web applications. It provides the flexibility to structure the application as needed, making it suitable for larger projects with more complex requirements.

4.6 Some more concepts related to Tools which can be used:

R: Another programming language, particularly popular in statistics and data analysis.

Integrated Development Environments (IDEs): Such as PyCharm, VSCode, or Spyder, provide a more traditional coding environment for larger projects.

TensorFlow: An open-source ML framework developed by Google.

PyTorch: Another popular open-source ML framework, often preferred for its dynamic computation graph.

Keras: High-level neural networks API that can run on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK).

XGBoost, LightGBM, CatBoost: Libraries for gradient boosting algorithms.

Scrapy, BeautifulSoup: For web scraping.

OpenCV: For computer vision tasks.

NLTK, spaCy: Libraries for natural language processing.

FastAPI, Django: Web frameworks for building APIs to serve ML models.

TensorFlow Serving, ONNX, TorchServe: Tools for serving ML models in production.

Google Cloud Platform (GCP), AWS, Azure: Provide cloud services for scalable computing resources, storage, and ML services.

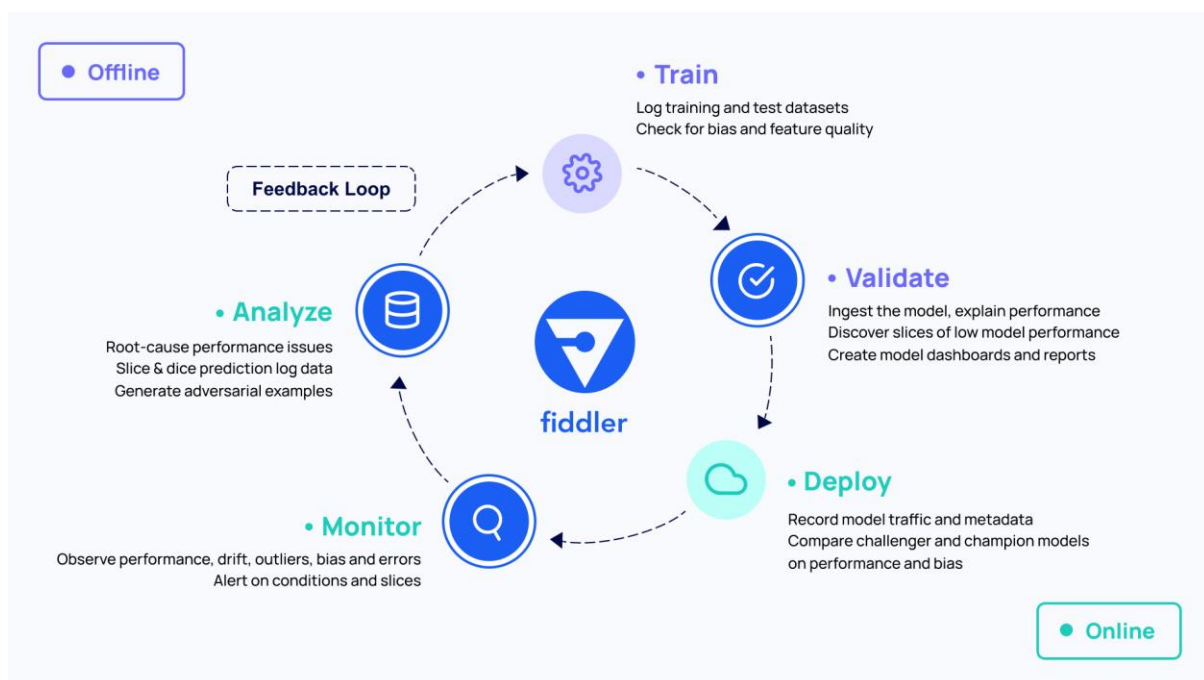


Figure 4.1: A Symbolic Representation of the Machine Learning Process.

4.7 Concepts used in this project:

1. Numpy for usage of arrays which are efficient than python lists in manipulating Machine Learning data processing techniques.
2. Pandas work well with dataframes. A dataframe is a one consisting of columns labelled as classes, and rows have the values for those classes. There may be 30,000 to more than that rows in a particular dataset.
3. Matplotlib works well with statistical and mathematical computations required while processing data series. It also helps to plot various plots/graphs required for visualising the dataset to clean the “dirty” data.
4. Seaborn is the extension of Matplotlib library. It extrapolates the graphs/plots plotted using Matplotlib for the purpose of visualising them in beautiful manner than Matplotlib. Examples are) Pairplot, Barplot, etc.
5. Scikit-learn is a library mainly used for Machine Learning process starting from scaling the data until cross-validating the model after model building process.
6. StandardScaler, MinMaxScaler used for scaling the data range from 0 to 1 rather than leaving the data in the original way it exists to improve the efficiency of the model.
7. train_test_split is a function from model_selection module purposefully used to split the data into training and testing set with test_size or train_size mentioned explicitly.
8. Various Machine Learning algorithms Support Vector Classifier (SVC) from SVM module, Decision Tree Classifier from tree module, Random Forest Classifier from Ensemble module and K Neighbors Classifier from neighbors have been imported.
9. accuracy_score, classification report (recall, f1 score, precision) and confusion matrix plotted with heatmap of seaborn library are all used to cross-validate the model's accuracy in predicting the values.
10. Label Encoder to transform and label the predicted values into range of integers.
11. Pickle or Joblib modules' can be used to dump the ML model and use the same in Flask or Streamlit API's for further better User Interface creation of the same.
12. Flask and Streamlit API's are used to create UI's of the system using various built-in methods provided in these libraries for better visualisation and providing the output of the same.
13. To host the same in some of the cloud platforms could be done in later stages.

Chapter 5

Implementation

Chapter 5

IMPLEMENTATION

5.1 Code Requirements:

1. A web browser with stable internet connection in which google colab is to be used or else the use of Jupyter notebook embedded with Visual Studio Code is mandatory.
2. Gathering the dataset relevant to the problem statement from various open source libraries, Kaggle dataset, from well-known libraries or toy dataset with less data values is the primary step in the process.
3. Installing the dependencies such as Numpy, Pandas, Matplotlib, Seaborn, Scikit-learn and Pickle into the working directory for kickstarting the model building process.
4. To be well-known with Flask API, Streamlit User Interface and installing the same for UI purpose is the final stage of coding process.

5.2 Concepts Used:

1. Data Collecting: It is a systematic process of gathering observations or measurements.
2. Data Cleaning: Data cleaning is a crucial step in the data mining process. It ensures that the data used for analysis is accurate, reliable, and free from errors.
3. Data Preprocessing: It involves cleaning, transforming, and integrating raw data to make it suitable for analysis.
4. Data Visualisation: It is a powerful way to convey information and insights from data. It allows you to present complex data in a clear and engaging manner.
5. Data Splitting: It involves dividing the available dataset into separate subsets for training, validation, and testing the model.
6. Model Building: It involves utilizing built-in functions from libraries that recognize patterns in data or make predictions.
7. Cross-Validation of the model: It is a technique used in machine learning to evaluate the performance of a model on unseen data.
8. Model Deployment: It involves making our trained model available for use in a production environment.

5.3 Source Code

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
"""data collection"""

df=pd.read_csv('Crop_recommendation.csv')

df.head(3)

"""data preprocessing, data cleaning, data visualisation"""

import seaborn as sb
sb.pairplot(df)

sb.barplot(x='label', y='humidity', data=df)
plt.xticks(rotation=90)

df.shape

df.describe()

df.isna().sum()

df.boxplot(column='N',grid=True)
plt.show()

df.boxplot(column='P',grid=False)
plt.show()

df.boxplot(column='K',grid=False)
plt.show()

df.boxplot(column='temperature',grid=False)
plt.show()

df.boxplot(column='humidity',grid=False)
plt.show()

df.boxplot(column='ph',grid=False)
plt.show()

df.boxplot(column='rainfall',grid=False)
plt.show()
```

```
df.dtypes

df['label'].value_counts()

df.size

df['label'].unique()

X=df.drop(['label'],axis='columns')
y=df.label

import seaborn as sb
sb.heatmap(X.corr(),annot=True)
plt.show()

X.tail(5)

"""data scaling"""

from sklearn.preprocessing import StandardScaler
Scaler=StandardScaler()

X_Scaled=Scaler.fit_transform(X)
X_Scaled

print(len(X),len(y))

X.shape

X.describe()

"""data splitting"""

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_Scaled,y,test_size=0.25,stratify=y,random_state=20,shuffle=True)

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,stratify=y,random_state=20,shuffle=True)

print(len(X_train),len(y_test))

"""model building"""

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```

from sklearn.ensemble import RandomForestClassifier

#model=KNeighborsClassifier()
model=RandomForestClassifier()
model.fit(X_train,y_train)

model3=SVC()
model3.fit(X_train,y_train)
model3.score(X_test,y_test)

model2=DecisionTreeClassifier()
model2.fit(X_train,y_train)

"""cross-validation of the model"""

print("Random Forest:",model.score(X_train,y_train),"\n","Decision
Tree:",model2.score(X_train,y_train))
print("Random Forest:",model.score(X_test,y_test),"\n","Decision
Tree:",model2.score(X_test,y_test))
print("SVC:",model3.score(X_test,y_test),"\n","DSVC:",model3.score(X_train,y_t
rain))

acclist=[]
modellist=[]

from sklearn.metrics import accuracy_score
y_predicted=model.predict(X_test)
y_predicted[:6]
accuracy_score=accuracy_score(y_test,y_predicted)
acclist.append(accuracy_score)
modellist.append('Random Forest Classifier')
accuracy_score

from sklearn.metrics import accuracy_score
y_predicted2=model2.predict(X_test)
y_predicted2[:6]
accuracy_score2=accuracy_score(y_test,y_predicted2)
acclist.append(accuracy_score2)
modellist.append('Decision Tree Classifier')
accuracy_score2

y_predicted3=model3.predict(X_test)
y_predicted3[:6]
accuracy_score3=accuracy_score(y_test,y_predicted3)
acclist.append(accuracy_score3)
modellist.append('Support Vector Machine')

plt.figure(figsize=[10,5],dpi = 100)

```

```

plt.title('Accuracy Comparison')
plt.xlabel('Accuracy')
plt.ylabel('Algorithm')
sb.barplot(x = acclist,y = modellist,palette='dark')

"""classification report"""

modellist=['RF','DTC','SVM']

X_position=np.arange(len(acclist))
plt.bar(X_position,acclist,color=['#488AC7','#ff8c00','#009150'])
plt.xticks(X_position,modellist)
plt.ylabel('Accuracy(in %)')
plt.xlabel('Machine Learning Classifying Techniques')
plt.show()

from sklearn.metrics import classification_report
cl_report=classification_report(y_test,y_predicted)
print(cl_report)

cl_report_2=classification_report(y_test,y_predicted2)
print(cl_report_2)

cl_report_3=classification_report(y_test,y_predicted3)
print(cl_report_3)

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_predicted)
cm

cm2=confusion_matrix(y_test,y_predicted2)
cm2

cm3=confusion_matrix(y_test,y_predicted3)
cm3

"""confusion matrix"""

import seaborn as sb
plt.figure(figsize=(10,7))
sb.heatmap(cm,annot=True,fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.show()

import seaborn as sb
plt.figure(figsize=(10,7))
sb.heatmap(cm2,annot=True,fmt='d')

```

```

plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.show()

plt.figure(figsize=(10,7))
sb.heatmap(cm3,annot=True,fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.show()

# set width of bar
barWidth = 0.25
fig = plt.subplots(figsize =(10, 6))

# set height of bar
Algorithms = ['Random Forest', 'Decision-tree', 'KNN Classifier']
#Accuracy = [b1, b2, b3]

# Set position of bar on X axis
br1 = np.arange(len(acclist))
br2 = [x + barWidth for x in br1]
br3 = [x + barWidth for x in br2]

# Make the plot
plt.bar(br1, acclist, color = 'blue', width = barWidth,
        edgecolor = 'grey', label = 'Accuracy')

# Adding Xticks
plt.xlabel('Algorithms', fontweight = 'bold', fontsize = 10)
plt.ylabel('Accuracy (in %)', fontweight = 'bold', fontsize = 10)
plt.xticks([r + barWidth for r in range(len(acclist))],
           Algorithms)

plt.legend()
plt.show()

"""cross val score"""

from sklearn.model_selection import cross_val_score
scores=cross_val_score(model,X_test,y_test,cv=5)
scores.mean()

from sklearn.model_selection import cross_val_score
scores=cross_val_score(model2,X_test,y_test,cv=5)
scores.mean()

scores=cross_val_score(model3,X_test,y_test,cv=5)
scores.mean()

```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dfle=df
dfle=le.fit_transform(y_test)
y_t=dfle
y_t
y_p=le.fit_transform(y_predicted)
y_p
```

```
from sklearn.metrics import r2_score
res=r2_score(y_true=y_t,y_pred=y_p)
res
```

```
dfle2=df
dfle2=le.fit_transform(y_test)
y_t=dfle2
y_t
y_p=le.fit_transform(y_predicted2)
y_p
res=r2_score(y_true=y_t,y_pred=y_p)
res
```

```
dfle3=df
dfle3=le.fit_transform(y_test)
y_t=dfle3
y_t
y_p=le.fit_transform(y_predicted3)
y_p
res=r2_score(y_true=y_t,y_pred=y_p)
res
```

```
from sklearn.metrics import mean_squared_error
dfle=df
dfle=le.fit_transform(y_test)
y_t=dfle
y_t
y_p=le.fit_transform(y_predicted)
y_p
n=len(y_t)
sum1=sum(((y_t-y_p)**2)/n)
mse=sum1
#mse=mean_squared_error(y_t,y_p)
mse
```

```
dfle=df
dfle=le.fit_transform(y_test)
y_t=dfle
```



```
y_t
y_p=le.fit_transform(y_predicted2)
y_p
mse=mean_squared_error(y_t,y_p)
mse

dfle=df
dfle=le.fit_transform(y_test)
y_t=dfle
y_t
y_p=le.fit_transform(y_predicted3)
y_p
mse=mean_squared_error(y_t,y_p)
mse

print(len(X_train),len(X_test),len(y_train),len(y_test))

print(y_test[525:531])

"""final prediction"""

res=model.predict([[34,45,23,29.21780035, 87.93724219,6.54450214,43.1386631]])
sol=res[0]

res2=model2.predict([[34,45,23,29.21780035,
87.93724219,6.54450214,43.1386631]])
sol2=res2[0]

res3=model3.predict([[34,45,23,29.21780035,
87.93724219,6.54450214,43.1386631]])
sol3=res3[0]

print("Random Forest:",sol)
print("Decision Tree:",sol2)
print("SVM:",sol3)

y_test.value_counts()

"""file dumping"""

import pickle
with open('Crop recommendation.pkl','wb') as file:
    pickle.dump(model,file)

import pickle
with open('Crop recommendation2.pkl','wb') as file:
    pickle.dump(model2,file)
```

```
with open('Crop_recommendation3.pkl','wb') as file:
    pickle.dump(model3,file)
```

```
"""file loading"""
```

```
with open('Crop_recommendation.pkl','rb') as file:
    loaded_model1=pickle.load(file)
```

```
with open('Crop_recommendation2.pkl','rb') as file:
    loaded_model2=pickle.load(file)
```

```
with open('Crop_recommendation3.pkl','rb') as file:
    loaded_model3=pickle.load(file)
```

```
ans=loaded_model3.predict([[2,4,3,32,54,6,76]])
ans[0]
```

```
ans=loaded_model1.predict([[2,4,3,32,54,6,76]])
ans[0]
```

```
ans=loaded_model2.predict([[2,4,3,32,54,6,76]])
ans[0]
```

Streamlit file:-

```
def crop_recommendation(input_data):
    i_data=np.asarray(input_data)
    final_data=i_data.reshape(1,-1)
    prediction=loaded_model.predict(final_data)
    return prediction[0]
def main():
    st.subheader(':red[Take your desired crop!] :rose: ')
    styles={
        "padding":"20px",
        "color":"red",
    }
    N=st.text_input('Nitrogen content in the field',placeholder="Nitrogen")
    P=st.text_input('Phosphorous content in the
field',placeholder="Phosphorous")
    K=st.text_input('Potassium content in the field',placeholder="Potassium")
    Temperature=st.text_input('Temperature of the crop
field',placeholder="Temperature")
    Humidity=st.text_input('Humidity of the field
region',placeholder="Humidity")
    pH=st.text_input('ph of the respective crop',placeholder="pH")
    Rainfall=st.text_input('Rainfall near the crop
field',placeholder="Rainfall")
    input_file=[N,P,K,Temperature,Humidity,pH,Rainfall]
    recommended=''
    if (st.button('Crop name')):
```

```

recommended=crop_recommendation(input_file)

if st.success(recommended):
    st.title('ಅನ್ನದಾತೋ ಸುಖೀಭವ:')
if __name__ == '__main__':
    main()

```

Flask code used as HTML, CSS:

```

from flask import Flask,request,render_template
import numpy as np
import pandas
import sklearn
import pickle

# importing model
model = pickle.load(open('Crop recommendation.pkl','rb'))
#sc = pickle.load(open('standscaler.pkl','rb'))
#ms = pickle.load(open('minmaxscaler.pkl','rb'))

# creating flask app
app = Flask(__name__)

@app.route('/')
def index():
    return render_template("index.html")

@app.route("/predict",methods=['POST'])
def predict():
    N = request.form['Nitrogen']
    P = request.form['Phosphorus']
    K = request.form['Potassium']
    temp = request.form['Temperature']
    humidity = request.form['Humidity']
    ph = request.form['Ph']

    rainfall = request.form['Rainfall']

    feature_list = [N, P, K, temp, humidity, ph, rainfall]
    single_pred = np.array(feature_list).reshape(1, -1)

    #scaled_features = ms.transform(single_pred)
    #final_features = sc.transform(scaled_features)
    prediction = model.predict(single_pred)

    # crop_dict = {1: "Rice", 2: "Maize", 3: "Jute", 4: "Cotton", 5: "Coconut", 6: "Papaya", 7: "Orange",
    #              8: "Apple", 9: "Muskmelon", 10: "Watermelon", 11: "Grapes", 12: "Mango", 13: "Banana",
    #              14: "Pomegranate", 15: "Lentil", 16: "Blackgram", 17: "Mungbean", 18: "Mothbeans",
    #              19: "Pigeonpeas", 20: "Kidneybeans", 21: "Chickpea", 22: "Coffee"}
    crop = prediction[0]
    result = "{} is the best crop to be cultivated right there".format(crop)

    #else:
    # result = "Sorry, we could not determine the best crop to be cultivated with the provided data."
    return render_template('index.html',result = result)

```

5.4 Source Code Explanation:

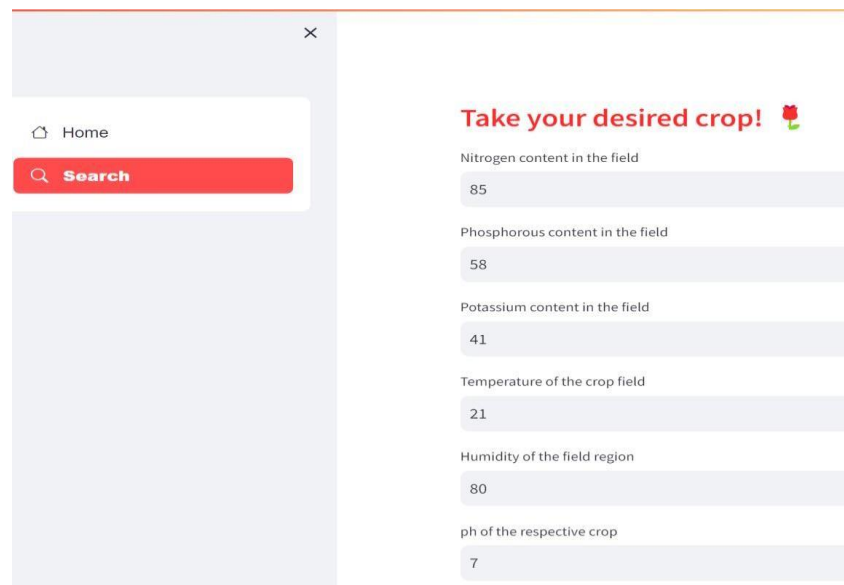
1. Initially, data from .csv file is being loaded into colab directory in which the .ipynb file is present.
2. The Pairplot, barplot of input data are drawn using matplotlib library for data visualisation.
3. The data is checked for null values. If present, it is filled up using suitable techniques.
4. To check the datatypes and size of various input variables and splitting the input and output variable (target variable) into X and Y respectively.
5. Plotting of heatmap for X variables for better probability of input variables.
6. To scale the input and output variables to improve the efficiency of the model using various libraries and split X and Y using train_test_split function with suitable test size.
7. Create a model instance using built-in model from sklearn library and fit the training data into the model and calculate the score of the model's performance using testing data.
8. Calculate Mean Squared Error, R-Squared Error, plot heatmap of the model's prediction, cross val score, classification report for looking about model's performance.
9. Make the model to predict by providing the values manually and load the model into pickle file or joblib file.
10. Open streamlit file and load the pickle or joblib file. Then, using suitable UI, display the model's output using streamlit functions.
11. Similarly, open a python file and import flask api into it.
12. With suitable HTML and CSS, make the UI of the page and after loading the pickle file, and other UI components, make the model to give its predictions into the Web Page.
13. Thus, the explanation of the code.
14. Here, streamlit, flask and jupyter files have been used to create this project.

Chapter 6

Results

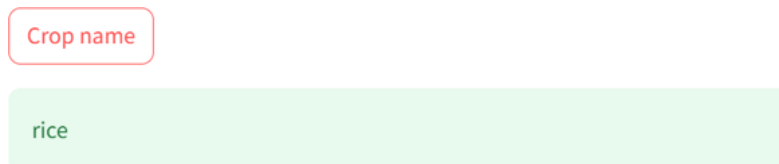
Chapter 6

RESULTS



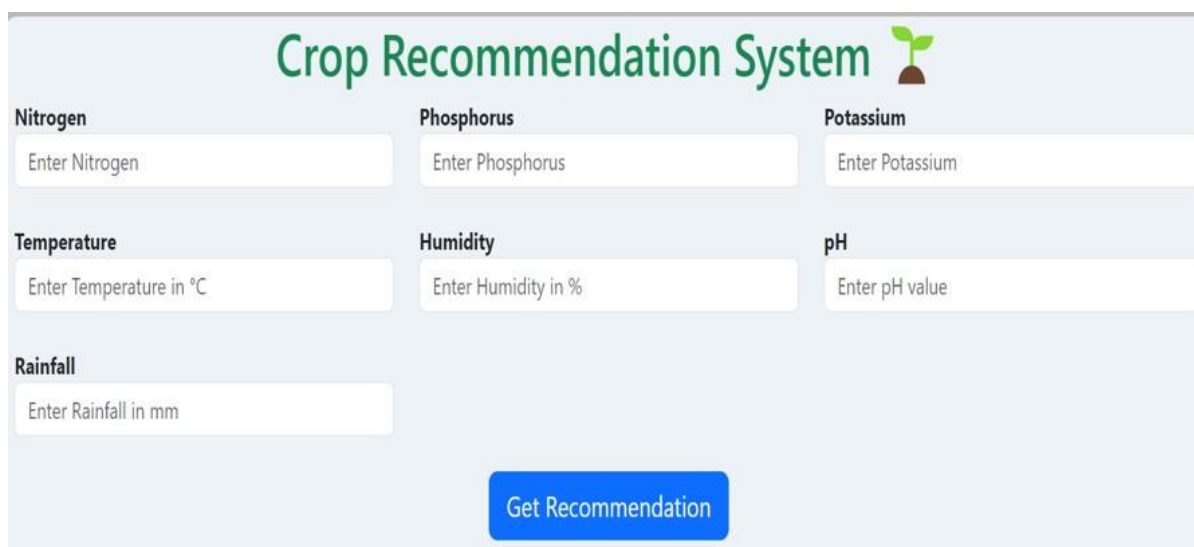
The Streamlit UI displays a sidebar with a 'Home' link and a 'Search' button. The main area shows the title 'Take your desired crop!' with a flower icon. Below the title, seven input fields are listed with their corresponding values: Nitrogen content in the field (85), Phosphorous content in the field (58), Potassium content in the field (41), Temperature of the crop field (21), Humidity of the field region (80), and ph of the respective crop (7).

Figure 6.1: Output seen in streamlit UI



The Flask UI shows a 'Crop name' input field with the value 'rice' entered. The input field is highlighted with a green background.

Figure 6.2: Model provides the required answer as per the input values.



The Flask UI displays the 'Crop Recommendation System' interface. It features a title bar with a plant icon. Below the title bar, there are six input fields arranged in a 2x3 grid: Nitrogen (Enter Nitrogen), Phosphorus (Enter Phosphorus), Potassium (Enter Potassium), Temperature (Enter Temperature in °C), Humidity (Enter Humidity in %), and pH (Enter pH value). A 'Rainfall' input field (Enter Rainfall in mm) is located below the Temperature and Humidity fields. A blue 'Get Recommendation' button is positioned at the bottom center of the form.

Figure 6.3: Output seen in Flask UI

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	25
banana	1.00	1.00	1.00	25
blackgram	1.00	1.00	1.00	25
chickpea	1.00	1.00	1.00	25
coconut	1.00	1.00	1.00	25
coffee	1.00	1.00	1.00	25
cotton	1.00	1.00	1.00	25
grapes	1.00	1.00	1.00	25
jute	0.89	1.00	0.94	25
kidneybeans	1.00	1.00	1.00	25
lentil	1.00	1.00	1.00	25
maize	1.00	1.00	1.00	25
mango	1.00	1.00	1.00	25
mothbeans	1.00	1.00	1.00	25
mungbean	1.00	1.00	1.00	25
muskmelon	1.00	1.00	1.00	25
orange	1.00	1.00	1.00	25
papaya	1.00	1.00	1.00	25
pigeonpeas	1.00	1.00	1.00	25
pomegranate	1.00	1.00	1.00	25
rice	1.00	0.88	0.94	25
watermelon	1.00	1.00	1.00	25

Figure 6.4: Classification Report Depicting Recall, F1 score about Model's Accuracy.

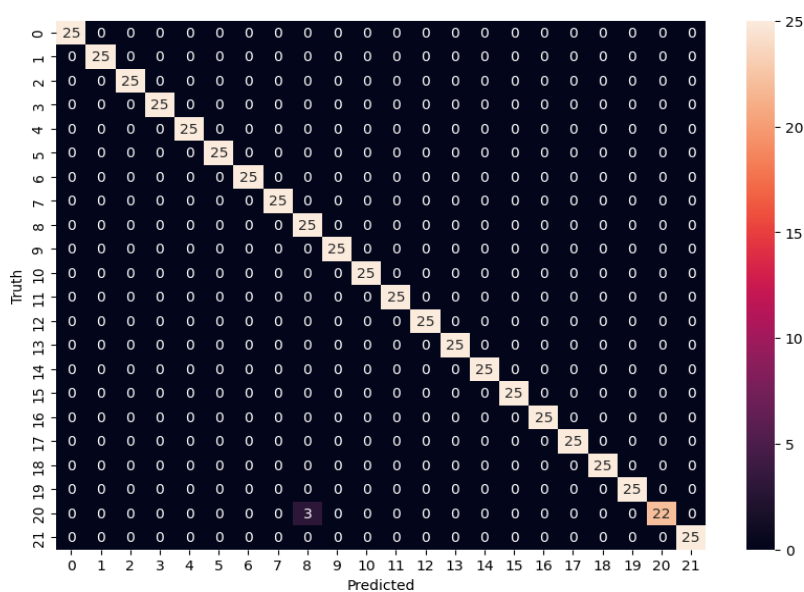


Figure 6.5: Confusion Matrix in Visualised manner done by Seaborn library.

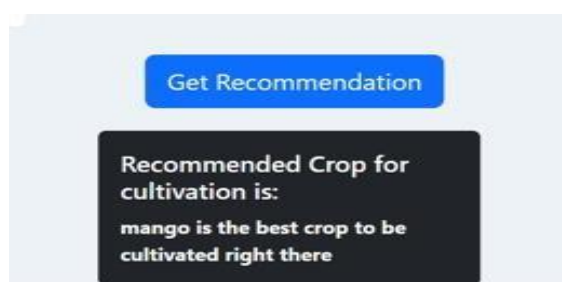


Figure 6.6: Output as seen in Flask UI.

Chapter 7

Reflection Notes

Chapter 7

REFLECTION NOTES

During my internship, I had the opportunity to work on crop recommendation system using machine learning, which provided a valuable learning experience and contributed to my technical skills. Below are some reflections on my internship experience and the outcomes I have achieved:

7.1 Technical Outcomes:

Python Programming Skills: Working on the crop recommendation system allowed me to further develop my Python programming skills. I became more proficient in machine learning concepts such as Numpy, Pandas, Scikit-Learn, Matplotlib, Seaborn to manage “dirty” data and process it.

Machine Learning Algorithms: Implementing the crop recommendation system involved using various machine learning algorithms such as Random Forest, Decision tree and SVM, KNN and so on. This experience enhanced my understanding of algorithms and how they can be effectively utilized to handle real-world scenarios.

7.2 Non-Technical Outcomes:

Improved Verbal and Written Communication: Throughout the internship, I had regular meetings with my mentor to discuss progress, ask questions, and seek guidance. These interactions improved my verbal communication skills as I learnt to articulate our thoughts and ideas more effectively. Additionally, working on code documentation and explanations helped enhance my written communication skills.

Personality Development: The internship provided a platform for personal growth and development. I learned to adapt to different tasks, take ownership of my work, and work on it. I also gained confidence in tackling complex problems and overcoming challenges.

Time Management and Resource Utilization: Managing the tasks and deadlines during the internship helped me to develop better time management skills. I learned to prioritize tasks, set achievable goals, and allocate resources efficiently to meet project requirements.

7.3 Contributions to the Organization:

During the internship, my contributions to the organization included:

Code Development: I actively participated in developing the crop recommendation system, creating the main functionality in the project. My contributions helped create a functional and efficient system.

Bug Fixing and Improvements: I identified and resolved bugs and issues in the code. This involved debugging and testing the system to ensure its reliability and robustness.

Documentation: I prepared detailed documentation for the codebase, including code comments and explanations. This documentation aimed to facilitate future maintenance and enhance the code's readability for other's who study it.

7.4 Overall Assessment

My internship experience was enriching and fulfilling. I had the opportunity to apply theoretical knowledge gained from my academic studies to real-world scenarios. The hands-on experience in Python programming, Machine Learning Algorithms' implementation has significantly boosted my confidence as a programmer.

In conclusion, the internship experience has been instrumental in shaping my career aspirations and has ignited a sense of passion and dedication towards a career in software development. I am grateful for the support and guidance I received during the internship, which has paved the way for my growth as a professional.

As we conclude our internship, we feel well-equipped to tackle future challenges in software development and eager to continue my journey in the field of technology. The internship has reinforced our passion for programming and has solidified my commitment to pursuing a career in software engineering. We are grateful for the opportunities and experiences we gained during this internship, and we look forward to applying our skills and knowledge to make a positive impact in the tech industry

Chapter 8

REFERENCES

<https://www.geeksforgeeks.org>

<https://www.w3schools.com>

[scikit-learn: machine learning](#)

[in Python — scikit-learn 1.4.1](#)

[documentation](#)