ABSTRACT

Agriculture is one of the most important components of our society. Soil is a critical factor for a successful agriculture. The composition of soil differs from soil to soil. The Growth of Crops is affected by these chemical features of soil. Choosing the right type of crops for that particular type of soil is also important. The proper classification of soil and the informed selection of suitable crops are essential factors in optimizing agricultural productivity. This project aims to develop a system that combines soil classification techniques with crop suggestion algorithms to assist farmers in making informed decisions about crop selection based on soil characteristics. Machine Learning techniques can be used to classify the soil series data. The results of such classification can further be combined with crop dataset to predict the crops that are suitable for the soil series of a particular region and its climatic conditions. Soil dataset and crop dataset are used. The datasets comprise of chemical attributes and geographical attributes of soil and crops. Machine learning is one of the budding technologies in the field of agriculture. Machine Learning can be used to improve the productivity and quality of the crops in the agricultural sector.

CONTENTS

CHAPTER 1 : INTRODUCTION

1.1 Problem Definition

The aim of this project is to develop a machine learning solution for soil classification and crop suggestion. The dataset includes various soil attributes such as Nitrogen (N), Phosphorous (P), Potassium (K), temperature, humidity, pH, rainfall, and corresponding crop labels. The goal is to build a robust model that accurately classifies soil types and provides crop recommendations based on these soil attributes.

1.2 Objective Of Project

The project aims to develop a robust machine learning model capable of accurately classifying various soil types based on critical soil characteristics like texture, pH level, moisture content, organic matter, and nutrient composition. Concurrently, the system will integrate a crop recommendation mechanism that considers not only soil conditions but also incorporates additional factors such as climate, temperature, and rainfall patterns. This holistic approach seeks to empower farmers with tailored insights for optimal crop selection and resource management. The implementation will include a user-friendly interface, fostering accessibility for farmers. Moreover, the project emphasizes precision agriculture by potentially incorporating real-time data from IoT devices and sensors for dynamic adjustments in crop recommendations. By focusing on data integrity, model interpretability, and scalability, the project aims to not only improve agricultural decision-making but also contribute to increased crop yields, resource efficiency, and overall economic impact within the farming community. Continuous refinement and adaptability will be pivotal in ensuring the system's effectiveness across diverse regions and evolving agricultural landscapes.

1.3 Limitations Of Project

Data Quality: The quality of input data is crucial. Inaccurate or incomplete soil data can lead to Incorrect classifications and crop recommendations. Soil data should be collected with care and verified for accuracy.

Data Availability: Access to comprehensive and up-to-date soil data can be a challenge, especially in remote or developing areas. Obtaining reliable soil data for a given region may be limited.

Regional Variability: Soil types and conditions can vary significantly from one region to another. Models trained on data from one area may not generalize well to other regions.

Weather Variability: Crop recommendations should ideally consider weather patterns and climate data. Weather conditions can fluctuate from year to year, making it challenging to provide long-term recommendations.

Long-Term Maintenance: Keeping the machine learning system up-to-date with evolving data and technologies can be an ongoing challenge, requiring continuous maintenance.

CHAPTER 2 : ANALYSIS

2.1 Introduction

Agriculture is one of the most important components of our society. Soil is a critical factor for a successful agriculture. The composition of soil differs from soil to soil. The Growth of Crops is affected by these chemical features of soil. Choosing the right type of crops for that particular type of soil is also important. The proper classification of soil and the informed selection of suitable crops are essential factors in optimizing agricultural productivity. This project aims to develop a system that combines soil classification techniques with crop suggestion algorithms to assist farmers in making informed decisions about crop selection based on soil characteristics. Machine Learning techniques can be used to classify the soil series data. The results of such classification can further be combined with crop dataset to predict the crops that are suitable for the soil series of a particular region and its climatic conditions. Soil dataset and crop dataset are used. The datasets comprise of chemical attributes and geographical attributes of soil and crops. Machine learning is one of the building technologies in the field of field of agriculture. Machine Learning can be used to improve the productivity and quality of the crops in the agricultural sector.

2.2 Software requirement specification

2.2.1 Software requirement

- Visual Studio Code
- SQL
- Python

2.2.2 Hardware requirement

- OS: Windows 10 or Higher
- Processor: Intel i5 processor or Higher
- Ram: Minimum 8 GB or Higher
- Hard Drive: Minimum 256 GB or Higher

2.3 Modules

Pandas(import pandas as pd**):** Pandas is a data manipulation and analysis library for Python. It provides data structures and functions needed to manipulate structured data.

Numpy (import numpy as np) : Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.

Matplotlib.pyplot(import matplotlib.pyplot as plt): Matplotlib is a 2D plotting library for Python. Pyplot is a module in Matplotlib that provides a convenient interface for creating various types of plots and charts.

Seaborn(import seaborn as sns): Seaborn is a data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

plotly.graph_objects (import plotly.graph_objects as go): Plotly is a graphing library that makes interactive, publication-quality graphs online. The graph_objects module provides a variety of chart types and customization options.

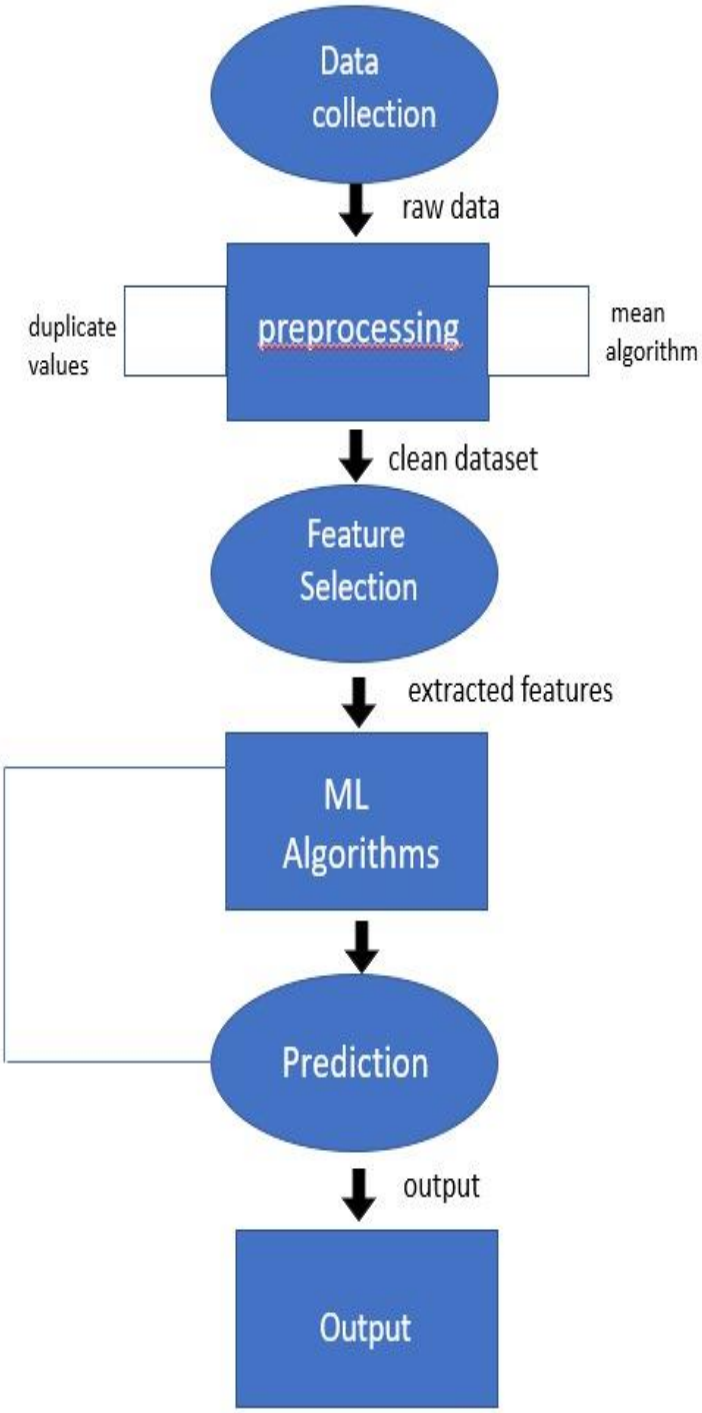
plotly.express (import plotly.express as px):Plotly Express is a high-level interface for creating a variety of interactive visualizations. It simplifies the process of creating complex charts.

plotly.subplots (from plotly.subplots import make_subplots):The make_subplots function allows the creation of subplots within a single plot, enabling the display of multiple charts in a grid.

scikit-learn (sklearn) :It is a popular machine learning library in Python, providing simple and efficient tools for data analysis and modeling. It includes various algorithms for classification, regression, clustering, and dimensionality reduction, along with utilities for model selection and data preprocessing.

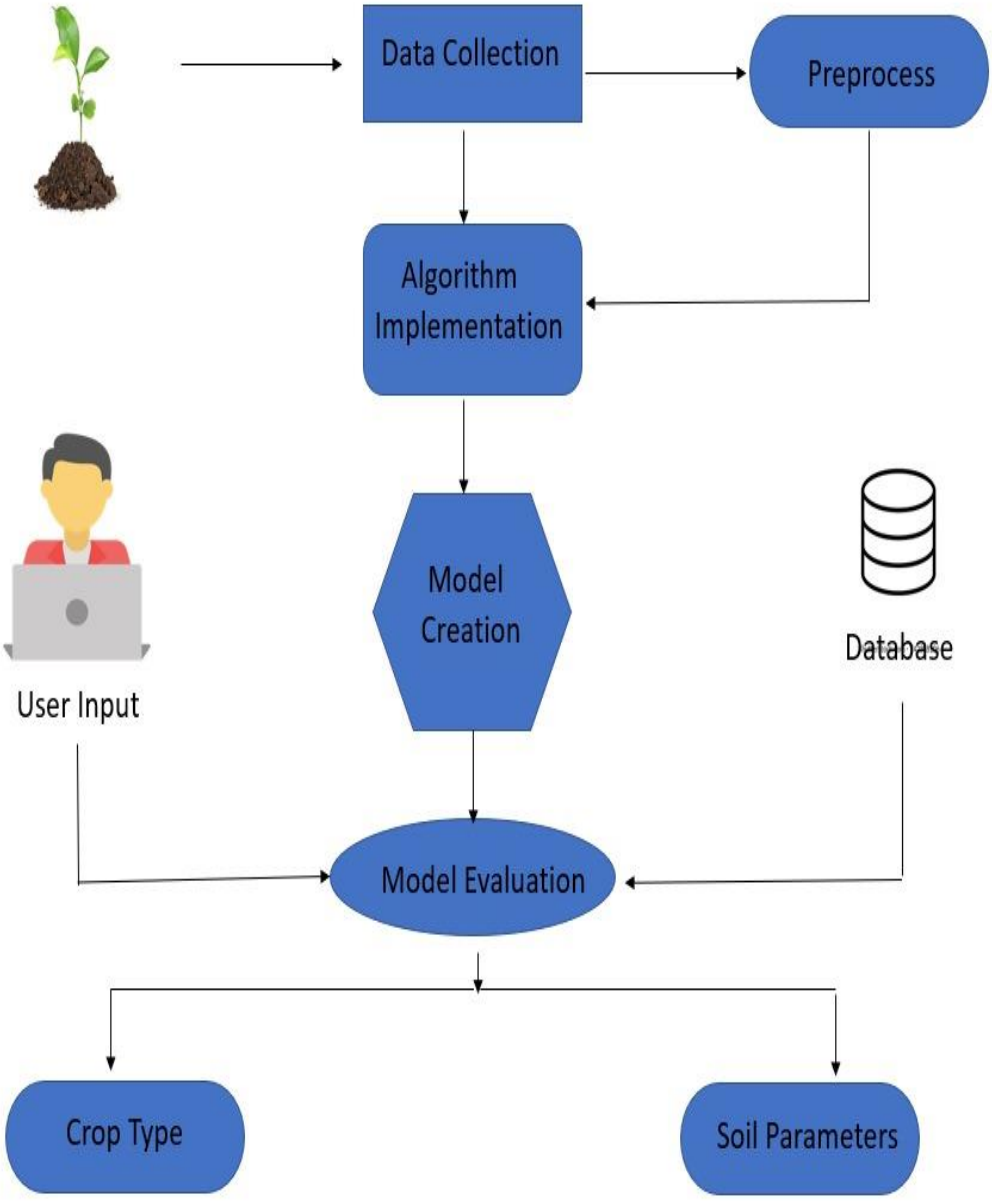
XGBoost (Extreme Gradient Boosting):It is a machine learning module that employs a gradient boosting framework for efficient and scalable tree-based ensemble learning. It is widely used for classification and regression tasks, known for its high performance and optimization techniques.

2.4 Architecture

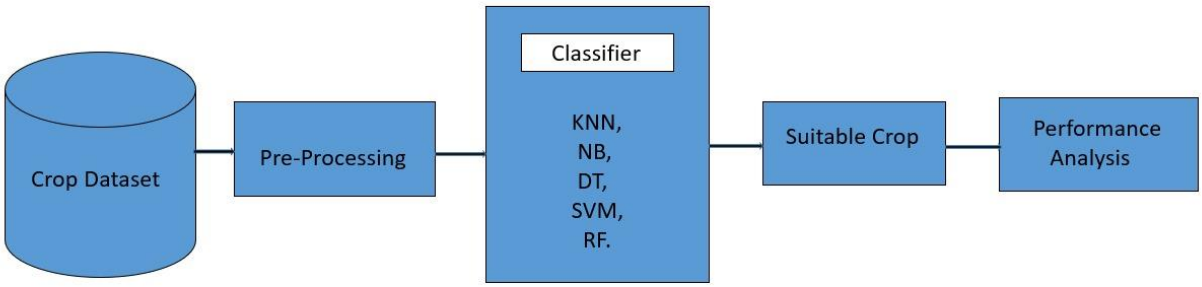


CHAPTER 3 : DESIGN

3.1 UML DIAGRAM



DataFlow Diagram:



3.2 Data Set Descriptions

Nitrogen (N):

Represents the quantity of nitrogen in the soil, measured in units not specified in the provided data.

Example: 90, 85, 60, 74, 78, 69.

Phosphorous (P):

Indicates the phosphorous content in the soil, measured in units not specified in the provided data.

Example: 42, 58, 55, 35, 42, 37.

Potassium (K):

Reflects the potassium levels in the soil, measured in units not specified in the provided data.

Example: 43, 41, 44, 40, 42, 42.

Temperature:

Denotes the temperature of the region, possibly in degrees Celsius.

Example: 20.87974371, 21.77046169, 23.00445915, 26.49109635, 20.13017482, 23.05804872.

Humidity:

Represents the humidity percentage in the atmosphere.

Example: 82.00274423, 80.31964408, 82.3207629, 80.15836264, 81.60487287, 83.37011772.

pH (Potential of Hydrogen):

Indicates the pH level of the soil, representing its acidity or alkalinity.

Example: 6.502985292, 7.038096361, 7.840207144,6.980400905, 7.628472891, 7.073453503.

Rainfall:

Represents the amount of rainfall in the region, measured in units not specified in the provided

data.

Example: 202.9355362, 226.6555374, 263.9642476, 242.8640342, 262.7173405, 251.0549998.

Label (Crop Name):

Denotes the name of the crop corresponding to the given set of soil attributes.

Example: rice, kidney beans, moth bean, coffee.

```
crop = pd.read_csv('../input/crop-recommendation-dataset/Crop_recommendation.csv')
crop.head(5)
```

[2] Python

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

```
[7] crop['label'].unique()
```

```
array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
       'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
       'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
       'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
      dtype=object)
```

```
[8] crop['label'].nunique()
```

```
22
```

```
[9] crop['label'].value_counts()
```

```
rice           100
maize          100
jute           100
cotton         100
coconut        100
papaya         100
orange         100
apple          100
muskmelon      100
watermelon     100
grapes         100
mango          100
banana         100
pomegranate    100
lentil         100
blackgram      100
mungbean       100
mothbeans      100
pigeonpeas     100
kidneybeans    100
chickpea       100
coffee         100
Name: label, dtype: int64
```

3.3 Data Preprocessing Techniques

Data Cleaning:

Handling Missing Values: Check for missing values in the dataset and decide on an appropriate strategy to handle them. You can either remove the rows with missing values or impute them using techniques like mean, median, or machine learning-based imputation.

Duplicate Check: Identify and remove duplicate rows.

Data Transformation:

Encoding Categorical Labels: Convert categorical labels like "Crop Name" into numerical representations using techniques like one-hot encoding or label encoding. This is necessary for many machine learning algorithms.

Normalizing/Scaling Numerical Features: Normalize or standardize numerical features like Nitrogen, Phosphorous, Potassium, Temperature, pH, and Rainfall. This is important for algorithms that are sensitive to the scale of input features. Common methods include Min-Max scaling or Z-score normalization.

Feature Engineering: Create additional relevant features based on domain knowledge. For example, you might calculate the ratio of Nitrogen to Phosphorous or create a new feature representing the combination of certain soil attributes.

Data Splitting:

Splitting Features and Labels: Explicitly separate features (X) and labels (y) to ensure clarity in your code.

Data Visualization:

Pairplot Creation: Generate a pairplot using Seaborn.

```
( import seaborn as sns
  sns.pairplot(crop,hue = 'label').
```

Heatmap Creation: Generate a heatmap to visualize feature correlations.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 8))
sns.heatmap(crop.corr(), annot=True, cmap='viridis')
plt.title('Correlation Heatmap')
plt.show()
```

Data Reshaping:

By reshaping we can add or remove dimensions or change number of elements in each dimension.

crop.shape

```
crop = pd.read_csv('../input/crop-recommendation-dataset/Crop_recommendation.csv')
crop.head(5)
```

[2]

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

```
crop.info()
```

[3]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   N            2200 non-null   int64
 1   P            2200 non-null   int64
 2   K            2200 non-null   int64
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

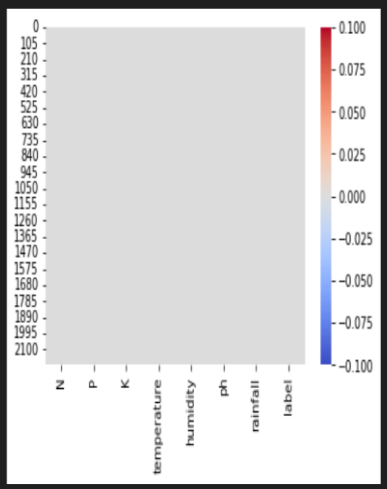
```
crop.describe()
```

[4]

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508

```
sns.heatmap(crop.isnull(),cmap="coolwarm")
plt.show()
```

[10]



```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features,target,test_size = 0.2,random_state =2)
```
[20]

```
crop.columns
```
[5]

```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')
```

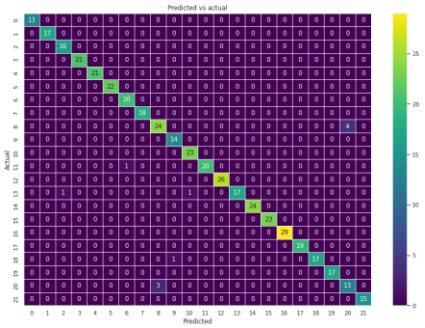
```
crop.shape
```
[6]

```
(2200, 8)
```

+ Code + Markdown

```
crop['label'].unique()
```
[7]

```
array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
       'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
       'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
       'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
      dtype=object)
```



3.4 Methods& Algorithms

ALGORITHMS USED:

K Nearest Neighbors (KNN):

- K Nearest Neighbors (KNN) is a simple yet effective classification algorithm**.**
- It classifies a data point based on the majority class of its k-nearest neighbors.

Decision Tree:

- Decision Trees are tree-like models used for decision-making in a hierarchical manner.
- They split the data at each node based on the most significant attribute.

Random Forest:

- Random Forest is an ensemble learning method that combines multiple decision trees.
- It constructs diverse trees using random subsets of data and features.

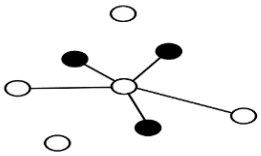
Naive Bayes:

- Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem.
- It assumes that features are conditionally independent given the class.

XG Boost:

- XG Boost (Extreme Gradient Boosting) is an optimized implementation of gradient boosting.
- It sequentially builds decision trees, each correcting errors of the previous ones.

XGBoost



METHODS USED:

Data Loading:

- The **pd.read_csv()** method is used to load the dataset from a CSV file.

Data Exploration and Analysis:

- Various methods like **info()** , **describe()** , and **shape** are used to explore and analyze the dataset's structure, statistics, and dimensions.
- Heatmaps **(sns.heatmap)** are used for visualizing missing values.

Data Visualization:

- Matplotlib, Seaborn, and Plotly are used for creating various plots, including histograms, pair plots, heatmaps, and bar charts.

Model Training and Evaluation:

- Models are trained using the **fit()** method of each respective model class.
- Model predictions are made using the **predict()** method.
- Model evaluation metrics such as accuracy, classification report, and confusion matrix are calculated using scikit-learn metrics.

Hyperparameter Tuning:

- Grid search for hyper parameter tuning is performed using **GridSearchCV** from scikit-learn.

3.5 Model Development &Training

MODEL TRAINING:

Problem Definition: Clearly define the problem you want the model to solve. In this case, it seems to be a classification problem where you want to predict the crop label based on soil attributes.

Data Collection: Gather a dataset that includes relevant features ('N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall') and the corresponding labels ('label') indicating the crop type.

Data Preparation: Clean the data by handling missing values and outliers. Ensure that the data is in a format suitable for training (numeric values, no categorical variables).

Feature and Label Separation: Separate the features (input variables) from the labels (output variable) in your dataset. 'X' typically represents the features, and 'y' represents the labels.

Label Encoding: Encode categorical labels (crop names) into numerical representations. This step is essential for training many machine learning algorithms.

Model Selection: Choose a suitable machine learning algorithm for your problem. K-Nearest Neighbors (KNN) is used.

MODEL TESTING:

Prepare the New Data: Prepare a new dataset with features similar to your training data. The features should include 'N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall'.

Preprocess the New Data: Apply the same preprocessing steps that you used for the training data. This includes handling missing values, encoding categorical variables, and scaling or normalizing numerical features. Use the same preprocessing steps and parameters applied during training to maintain consistency.

Make Predictions: Use the trained model to make predictions on the new dataset.This involves using the predict method on the new, preprocessed data:

Evaluate the Predictions: If you have labels for the new data (ground truth), you can compare the model predictions to the actual labels to evaluate the model's performance on the new dataset.

3.6 Model Evaluation Metrics

Accuracy: The ratio of correctly predicted instances to the total instances.

Formula: (TP + TN) / (TP + TN + FP + FN)

Accuracy measures the overall correctness of the model's predictions.

Precision: The ratio of correctly predicted positive observations to the total predicted positives.

Formula: TP / (TP + FP)

Precision measures the accuracy of the positive predictions.

Recall (Sensitivity, True Positive Rate): The ratio of correctly predicted positive observations to the all observations in the actual class.

Formula: TP / (TP + FN)

Recall measures the ability of the model to capture all the positive instances.

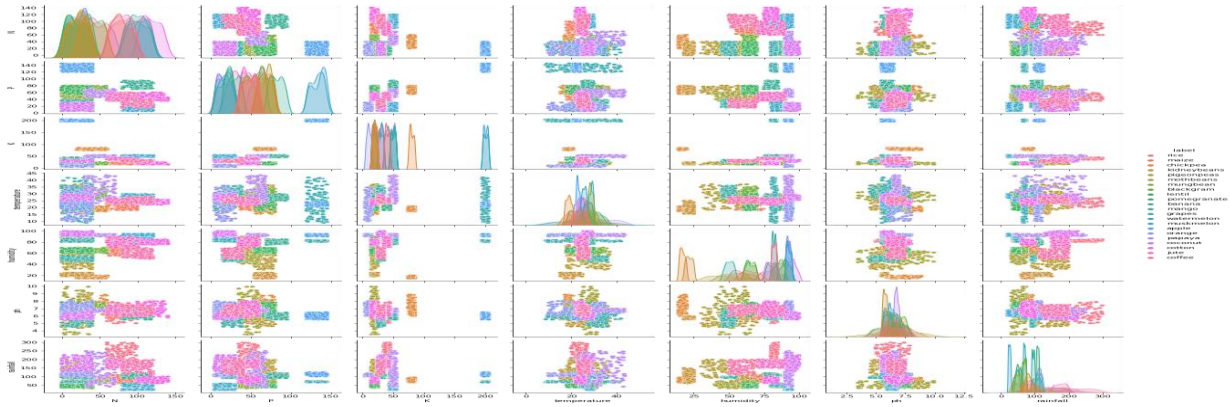
F1-Score: The weighted average of precision and recall. It considers both false positives and false negatives.

Formula: 2 * (Precision * Recall) / (Precision + Recall)

F1-Score is a balanced metric that considers both precision and recall.

Confusion Matrix: A table used to evaluate the performance of a classification algorithm. It shows the number of true positives, true negatives, false positives, and false negatives.

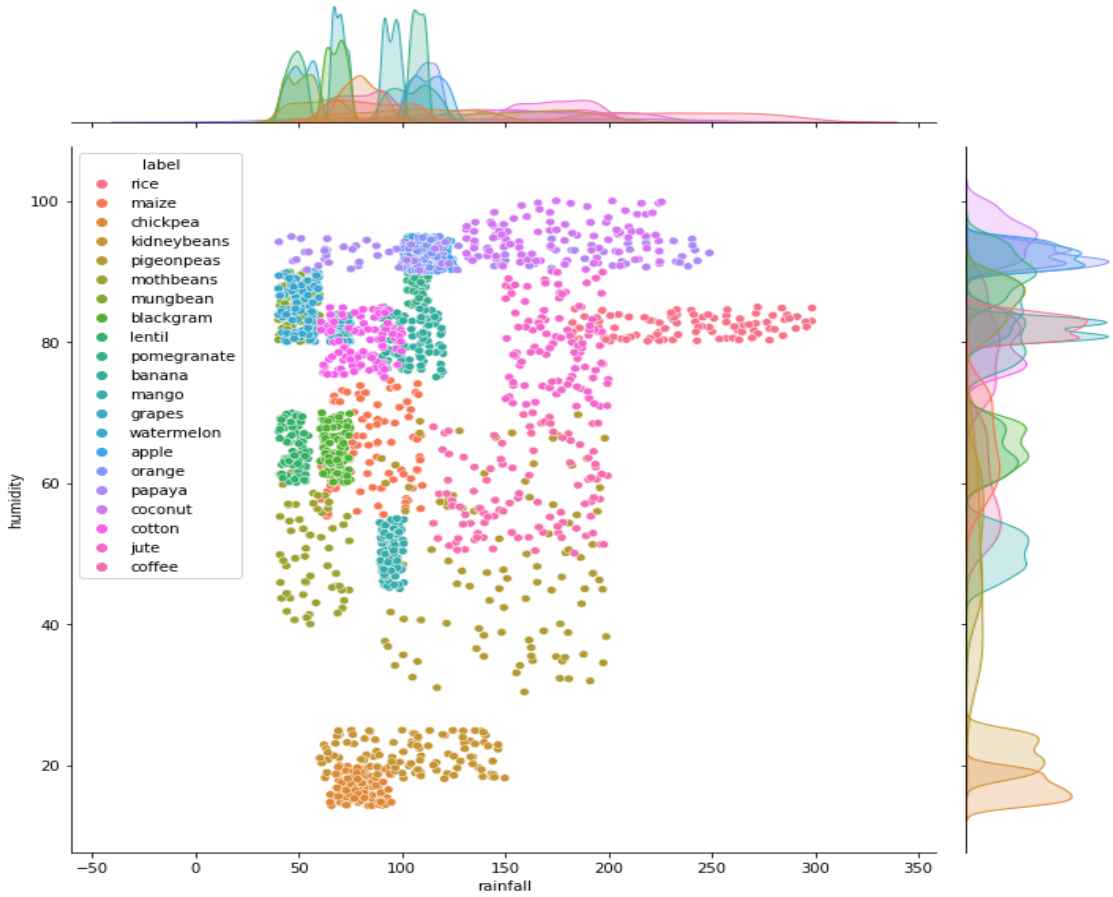
Model integration involves incorporating machine learning models into an application or system to enable intelligent decision-making or prediction based on the model's output. In the context of the provided dataset with crop labels, model integration would mean deploying a machine learning model trained on this data into a real-world application where it can make predictions or provide insights.

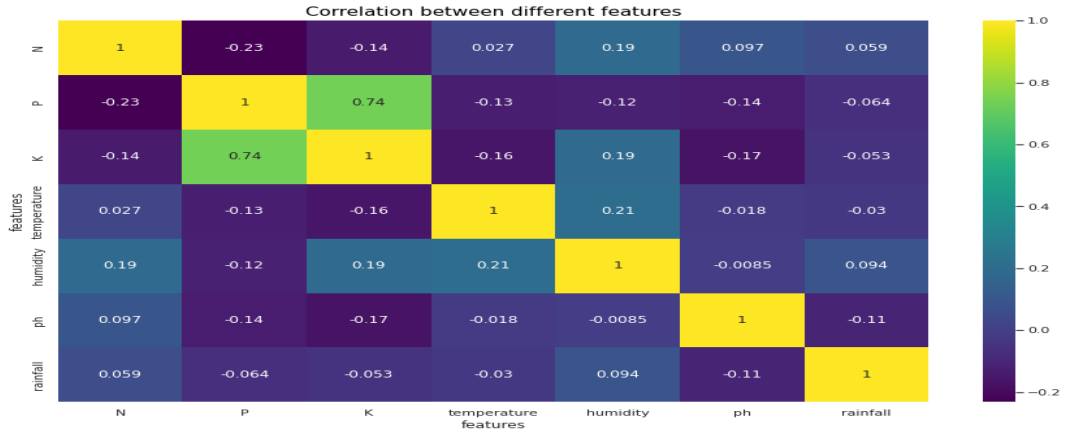


PAIR PLOT: A pair plot for the provided dataset, which includes different crops as labels, would allow you to visually explore the relationships between pairs of variables. Each point in the pair plot represents a data point.

JOINT PLOT: A joint plot is a type of statistical visualization that combines univariate plots for each variable along with bi variate plots between two variables. In this dataset with crop labels, a joint plot can help to visualize the distribution of individual variables and explore the relationships between pairs of variables.

The points in the plot will be colored based on the categorical variable 'label' (crop names). This allows you to distinguish data points belonging to different crops.





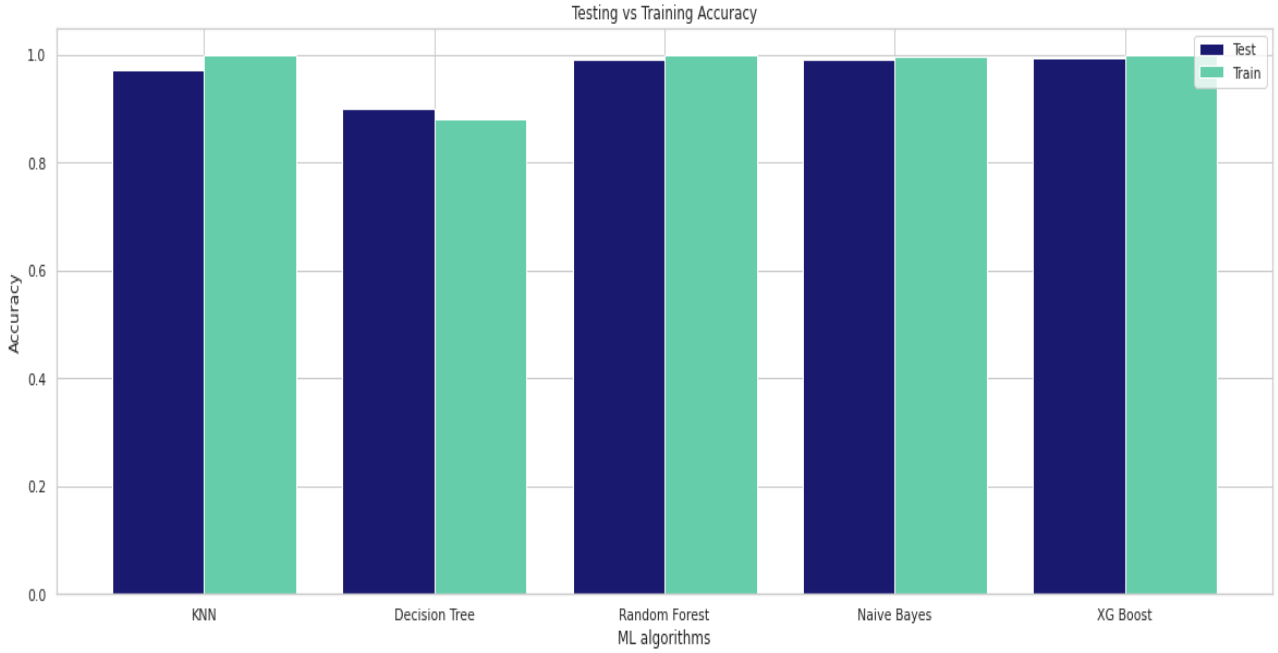
The dataset includes features such as Nitrogen (N), Phosphorous (P), Potassium (K), temperature, humidity, pH, rainfall, and crop labels. Select relevant features for the heatmap. In this dataset, features like N, P, K, temperature, humidity, pH, and rainfall are important for correlation analysis. A heatmap visualizes the correlation matrix of numerical features. It helps identify patterns and relationships between different variables. Features with higher correlations will be visually represented with distinct colors. Use the **sns.heatmap()** function to create the heatmap. Specify the correlation matrix as input data. Adjust parameters such as the color map, annotation, and axis labels for better visualization.

Training Accuracy:

Training accuracy measures how well a machine learning model performs on the same data it was trained on. It is calculated by comparing the model's predictions to the actual labels in the training set. A high training accuracy may indicate that the model has learned the patterns present in the training data.

Testing Accuracy:

Testing accuracy, also known as validation accuracy, evaluates how well a machine learning model generalizes to new, unseen data. It is calculated by comparing the model's predictions to the actual labels in the testing set. The goal is to have a high testing accuracy, indicating that the model can make accurate predictions on data it has not seen during training.


Testing vs Training Accuracy
Test
Train
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
KNN
Decision Tree
Random Forest
Naive Bayes
XG Boost
ML algorithms

CHAPTER 4 : DEPLOYMENT AND RESULTS

4.1 SOURCE CODE

```
__future__ import print_function
import pandas as pd # data analysis
import numpy as np # linear algebra

#import libraries for data visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots


from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn import tree
from sklearn.model_selection import cross_val_score
import warnings
warnings.filterwarnings('ignore')
crop = pd.read_csv('../input/crop-recommendation-dataset/Crop_recommendation.csv')
crop.head(5)
crop.info()
crop.describe()
crop.columns
crop.shape
crop['label'].unique()
crop['label'].nunique()
crop['label'].value_counts()
sns.heatmap(crop.isnull(),cmap="coolwarm")
plt.show()
plt.figure(figsize=(12,5))
plt.subplot(1, 2, 1)
# sns.distplot(df_setosa['sepal_length'],kde=True,color='green',bins=20,hist_kws={'alpha':0.3})
sns.distplot(crop['temperature'],color="red",bins=15,hist_kws={'alpha':0.5})
plt.subplot(1, 2, 2)
sns.distplot(crop['ph'],color="green",bins=15,hist_kws={'alpha':0.5})
sns.pairplot(crop,hue = 'label')
sns.jointplot(x="rainfall",y="humidity",data=crop[(crop['temperature']<40) &
                                    (crop['rainfall']>40)],height=10,hue="label")
sns.set_theme(style="whitegrid")
fig, ax = plt.subplots(figsize=(30,15))
sns.boxplot(x='label',y='ph',data=crop)
fig, ax = plt.subplots(1, 1, figsize=(15, 9))
sns.heatmap(crop.corr(), annot=True,cmap='viridis')
ax.set(xlabel='features')
ax.set(ylabel='features')
```

```
plt.title('Correlation between different features', fontsize = 15, c='black')
plt.show()
crop_summary = pd.pivot_table(crop,index=['label'],aggfunc='mean')
crop_summary.head()
fig = go.Figure()
fig.add_trace(go.Bar(
    x=crop_summary.index,
    y=crop_summary['N'],
    name='Nitrogen',
    marker_color='mediumvioletred'
))
fig.add_trace(go.Bar(
    x=crop_summary.index,
    y=crop_summary['P'],
    name='Phosphorous',
    marker_color='springgreen'
))
fig.add_trace(go.Bar(
    x=crop_summary.index,
    y=crop_summary['K'],
    name='Potash',
    marker_color='dodgerblue'
))

fig.update_layout(title="N-P-K values comparision between crops",
                  plot_bgcolor='white',
                  barmode='group',
                  xaxis_tickangle=-45)

fig.show()
features = crop[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]
target = crop['label']
acc = []
model = []
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features,target,test_size = 0.2,random_state =2)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()

knn.fit(x_train,y_train)

predicted_values = knn.predict(x_test)

x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)
model.append('K Nearest Neighbours')
print("KNN Accuracy is: ", x)

print(classification_report(y_test,predicted_values))
score = cross_val_score(knn,features,target,cv=5)
print('Cross validation score: ',score)
#Print Train Accuracy
```

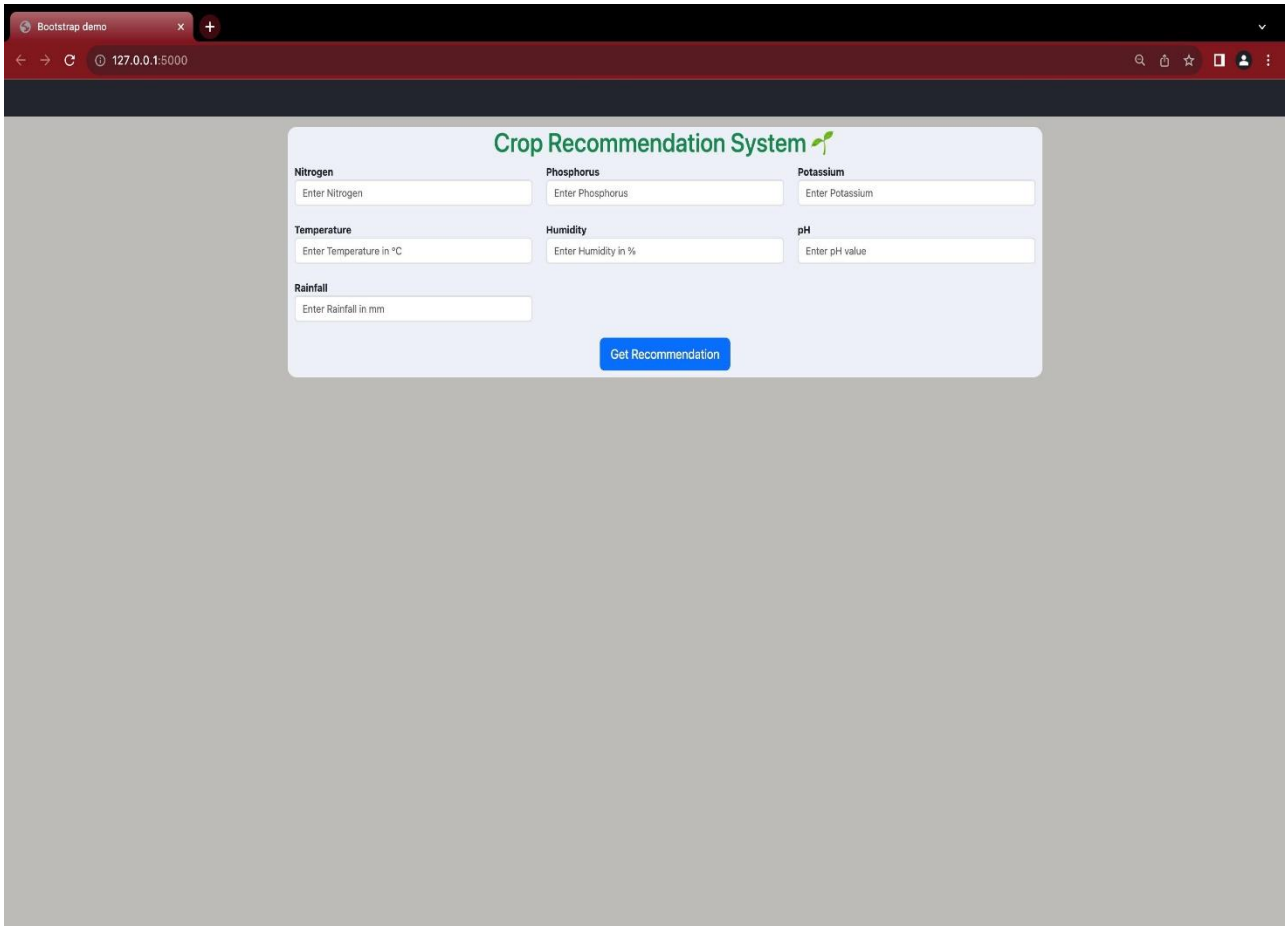
```
knn_train_accuracy = knn.score(x_train,y_train)
print("knn_train_accuracy = ",knn.score(x_train,y_train))
#Print Test Accuracy
knn_test_accuracy = knn.score(x_test,y_test)
print("knn_test_accuracy = ",knn.score(x_test,y_test))
y_pred = knn.predict(x_test)
y_true = y_test

from sklearn.metrics import confusion_matrix

cm_knn = confusion_matrix(y_true,y_pred)

f, ax = plt.subplots(figsize=(15,10))
sns.heatmap(cm_knn, annot=True, linewidth=0.5, fmt=".0f",cmap='viridis', ax = ax)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Predicted vs actual')
plt.show()
mean_acc = np.zeros(20)
for i in range(1,21):
    #Train Model and Predict
    knn = KNeighborsClassifier(n_neighbors = i).fit(x_train,y_train)
    yhat= knn.predict(x_test)
    mean_acc[i-1] = metrics.accuracy_score(y_test, yhat)

mean_acc
loc = np.arange(1,21,step=1.0)
plt.figure(figsize = (10, 6))
plt.plot(range(1,21), mean_acc)
plt.xticks(loc)
plt.xlabel('Number of Neighbors ')
plt.ylabel('Accuracy')
plt.show()
from sklearn.model_selection import GridSearchCV
grid_params = { 'n_neighbors' : [12,13,14,15,16,17,18],
            'weights' : ['uniform','distance'],
            'metric' : ['minkowski','euclidean','manhattan']}
gs = GridSearchCV(KNeighborsClassifier(), grid_params, verbose = 1, cv=3, n_jobs = -1)
g_res = gs.fit(x_train, y_train)
g_res.best_score_
g_res.best_params_
# Using the best hyperparameters
knn_1 = KNeighborsClassifier(n_neighbors = 12, weights = 'distance',algorithm = 'brute',metric
= 'manhattan')
knn_1.fit(x_train, y_train)
# Training & Testing accuracy after applying hyper parameter
knn_train_accuracy = knn_1.score(x_train,y_train)
print("knn_train_accuracy = ",knn_1.score(x_train,y_train))
#Print Test Accuracy
knn_test_accuracy = knn_1.score(x_test,y_test)
print("knn_test_accuracy = ",knn_1.score(x_test,y_test))
from sklearn.tree import DecisionTreeClassifier
```

```
DT = DecisionTreeClassifier(criterion="entropy",random_state=2,max_depth=5)

DT.fit(x_train,y_train)

predicted_values = DT.predict(x_test)
x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)
model.append('Decision Tree')
print("Decision Tree's Accuracy is: ", x*100)

print(classification_report(y_test,predicted_values))
score = cross_val_score(DT, features, target,cv=5)
print('Cross validation score: ',score)
#Print Train Accuracy
dt_train_accuracy = DT.score(x_train,y_train)
print("Training accuracy = ",DT.score(x_train,y_train))
#Print Test Accuracy
dt_test_accuracy = DT.score(x_test,y_test)
print("Testing accuracy = ",DT.score(x_test,y_test))
y_pred = DT.predict(x_test)
y_true = y_test

from sklearn.metrics import confusion_matrix

cm_dt = confusion_matrix(y_true,y_pred)

f, ax = plt.subplots(figsize=(15,10))
sns.heatmap(cm_dt, annot=True, linewidth=0.5, fmt=".0f",  cmap='viridis', ax = ax)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title('Predicted vs actual')
plt.show()
from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n_estimators=20, random_state=0)
RF.fit(x_train,y_train)

predicted_values = RF.predict(x_test)

x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)
model.append('RF')
print("Random Forest Accuracy is: ", x)

print(classification_report(y_test,predicted_values))
score = cross_val_score(RF,features,target,cv=5)
print('Cross validation score: ',score)
#Print Train Accuracy
rf_train_accuracy = RF.score(x_train,y_train)
print("Training accuracy = ",RF.score(x_train,y_train))
#Print Test Accuracy
rf_test_accuracy = RF.score(x_test,y_test)
```

```
print("Testing accuracy = ",RF.score(x_test,y_test))
y_pred = RF.predict(x_test)
y_true = y_test

from sklearn.metrics import confusion_matrix

cm_rf = confusion_matrix(y_true,y_pred)

f, ax = plt.subplots(figsize=(15,10))
sns.heatmap(cm_rf, annot=True, linewidth=0.5, fmt=".0f",  cmap='viridis', ax = ax)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title('Predicted vs actual')
plt.show()
from sklearn.naive_bayes import GaussianNB
NaiveBayes = GaussianNB()

NaiveBayes.fit(x_train,y_train)

predicted_values = NaiveBayes.predict(x_test)
x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)
model.append('Naive Bayes')
print("Naive Bayes Accuracy is: ", x)

print(classification_report(y_test,predicted_values))
score = cross_val_score(NaiveBayes,features,target,cv=5)
print('Cross validation score: ',score)
#Print Train Accuracy
nb_train_accuracy = NaiveBayes.score(x_train,y_train)
print("Training accuracy = ",NaiveBayes.score(x_train,y_train))
#Print Test Accuracy
nb_test_accuracy = NaiveBayes.score(x_test,y_test)
print("Testing accuracy = ",NaiveBayes.score(x_test,y_test))
y_pred = NaiveBayes.predict(x_test)
y_true = y_test

from sklearn.metrics import confusion_matrix

cm_nb = confusion_matrix(y_true,y_pred)

f, ax = plt.subplots(figsize=(15,10))
sns.heatmap(cm_nb, annot=True, linewidth=0.5, fmt=".0f",  cmap='viridis', ax = ax)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title('Predicted vs actual')
plt.show()
import xgboost as xgb
XB = xgb.XGBClassifier()
XB.fit(x_train,y_train)

predicted_values = XB.predict(x_test)
```

```
x = metrics.accuracy_score(y_test, predicted_values);
acc.append(x)
model.append('XGBoost')
print("XGBoost Accuracy is: ", x)

print(classification_report(y_test,predicted_values))
score = cross_val_score(XB,features,target,cv=5)
print('Cross validation score: ',score)
#Print Train Accuracy
XB_train_accuracy = XB.score(x_train,y_train)
print("Training accuracy = ",XB.score(x_train,y_train))
#Print Test Accuracy
XB_test_accuracy = XB.score(x_test,y_test)
print("Testing accuracy = ",XB.score(x_test,y_test))
y_pred = XB.predict(x_test)
y_true = y_test

from sklearn.metrics import confusion_matrix

cm_nb = confusion_matrix(y_true,y_pred)

f, ax = plt.subplots(figsize=(15,10))
sns.heatmap(cm_nb, annot=True, linewidth=0.5, fmt=".0f",  cmap='viridis', ax = ax)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title('Predicted vs actual')
plt.show()
plt.figure(figsize=[14,7],dpi = 100, facecolor='white')
plt.title('Accuracy Comparison')
plt.xlabel('Accuracy')
plt.ylabel('ML Algorithms')
sns.barplot(x = acc,y = model,palette='viridis')
plt.savefig('plot.png', dpi=300, bbox_inches='tight')
label = ['KNN', 'Decision Tree','Random Forest','Naive Bayes','XG Boost']
Test = [knn_test_accuracy, dt_test_accuracy,rf_test_accuracy,
       nb_test_accuracy, XB_test_accuracy]
Train = [knn_train_accuracy,  dt_train_accuracy, rf_train_accuracy,
        nb_train_accuracy, XB_train_accuracy]

f, ax = plt.subplots(figsize=(20,7)) # set the size that you'd like (width, height)
X_axis = np.arange(len(label))
plt.bar(X_axis - 0.2,Test, 0.4, label = 'Test', color=('midnightblue'))
plt.bar(X_axis + 0.2,Train, 0.4, label = 'Train', color=('mediumaquamarine'))

plt.xticks(X_axis, label)
plt.xlabel("ML algorithms")
plt.ylabel("Accuracy")
plt.title("Testing vs Training Accuracy")
plt.legend()
#plt.savefig('train vs test.png')
plt.show()
```

4.2 Final Results



CHAPTER 5: CONCLUSION

5.1 PROJECT COCLUSION

This project has successfully implemented machine learning models for soil classification and crop recommendation, demonstrating the feasibility of leveraging data-driven approaches in agriculture. The developed models showcase varying performances, and the crop recommendation system offers practical insights for farmers, aiding in informed decision-making. While the project contributes to improving crop selection processes, acknowledging limitations and continuous model refinement are crucial for ensuring the system's reliability and relevance in real-world agricultural scenarios.

5.2 FUTURE SCOPE

There is a scope for further development in our project to a great extent. In future suitable fertilizers are suggested for thewell growth of the crop cultivated. The present models deals with available old data whereas the future model contain the real time a data that is directly received from agricultural land that is placed with sensors .The sensors senses the soil fertilityand other minerals contained in the soil.

REFERENCES

[1] Gholap, J., Ingole, A., Gohil, J., Gargade, S. and Attar, V., 2012. Soil data analysis using classification techniques and soil attribute prediction. arXiv preprint arXiv:1206.1557.

[2] H. Eswaran, R. Ahrens, T. J. Rice and B. A. Stewart, Soil classification: a global desk reference, CRC Press, 2002

[3]M. P. K. Devi, U. Anthiyur and M. S. Shenbagavadivu, "Enhanced Crop Yield Prediction and Soil Data Analysis Using Data Mining", International Journal of Modern Computer Science, vol. 4, no. 6, 2016.

SOIL CLASSIFICATION AND CROP SUGGESTION USING MACHINE LEARNING

A project report submitted to
MALLA REDDY UNIVERSITY
in partial fulfillment of the requirements for the award of degree of

BACHELR OF TECHNOLOGY

IN

COMPUTER SCIENCE &ENGINEERING(AI & ML)

Submitted by

MADHU. S : (2111CS020253)

SANDHYA KUMARI. P : (2111CS020254)

MADHURI. M : (2111CS020255)

MADHURI SAGIRAJU : (2111CS020256)

MAHABOOB VALI SK : (2111CS020257)

MAHATHI M : (2111CS020258)

Under the guidance of

Dr.K. Rajeshwar Rao

Associate professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

2023

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

COLLEGE CERTIFICATE

This is to certify that this is the bonafide record of the application development entitled, **"Soil Classification and Crop Suggestion Using Machine Learning"** Submitted by

MADHU. S : (2111CS020253),

SANDHYA KUMARI. P : (2111CS020254),

MADHURI. M : (2111CS020255),

MADHURI SAGIRAJU : (2111CS020256),

MAHABOOB VALI SK : (2111CS020257),

MAHATHI M : (2111CS020258).

B. Tech III year I semester, Department of CSE (AI&ML) during the year 2023-24. The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE **HEAD OF THE DEPARTMENT**

Dr.K.Rajeshwar Rao Dr .ThayyabaKhatoonCSE(AI &ML)

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to my guide, Prof.Rajeshwar Rao, and head of department, Dr.Thayyaba Khatoon, for their invaluable guidance and unwavering support throughout the development of this project. Their insightful feedback helped me to refine my ideas and develop a comprehensive understanding of the subject matter.

Their mentorship was instrumental in shaping my approach towards the project, and I am grateful for the knowledge and experience they shared with me. Without their encouragement and support, this project would not have been possible. Once again, I extend my sincere thanks to my guides and head of department for their unwavering support and guidance.

Our sincere thanks to all the teaching and non-teaching staff of Department of Computer Science and Engineering (AI &ML) for their support throughout our project work.

SOIL CLASSIFICATION AND CROP SUGGESTION USING MACHINE LEARNING

A project report submitted to
MALLA REDDY UNIVERSITY
in partial fulfillment of the requirements for the award of degree of

BACHELR OF TECHNOLOGY

IN

COMPUTER SCIENCE &ENGINEERING(AI & ML)

Submitted by

MADHU. S : (2111CS020253)

SANDHYA KUMARI. P : (2111CS020254)

MADHURI. M : (2111CS020255)

MADHURI SAGIRAJU : (2111CS020256)

MAHABOOB VALI SK : (2111CS020257)

MAHATHI M : (2111CS020258)

Under the guidance of

Dr.K. Rajeshwar Rao

Associate professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

2023

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

COLLEGE CERTIFICATE

This is to certify that this is the bonafide record of the application development entitled, **"Soil Classification and Crop Suggestion Using Machine Learning"** Submitted by

MADHU. S : (2111CS020253),

SANDHYA KUMARI. P : (2111CS020254),

MADHURI. M : (2111CS020255),

MADHURI SAGIRAJU : (2111CS020256),

MAHABOOB VALI SK : (2111CS020257),

MAHATHI M : (2111CS020258).

B. Tech III year I semester, Department of CSE (AI&ML) during the year 2023-24. The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE **HEAD OF THE DEPARTMENT**

Dr.K.Rajeshwar Rao Dr .ThayyabaKhatoonCSE(AI &ML)

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to my guide, Prof.Rajeshwar Rao, and head of department, Dr.Thayyaba Khatoon, for their invaluable guidance and unwavering support throughout the development of this project. Their insightful feedback helped me to refine my ideas and develop a comprehensive understanding of the subject matter.

Their mentorship was instrumental in shaping my approach towards the project, and I am grateful for the knowledge and experience they shared with me. Without their encouragement and support, this project would not have been possible. Once again, I extend my sincere thanks to my guides and head of department for their unwavering support and guidance.

Our sincere thanks to all the teaching and non-teaching staff of Department of Computer Science and Engineering (AI &ML) for their support throughout our project work.