CENTER FOR DEVELOPMENT OF
ADVANCED COMPUTING

Report on FarMitra

PG-DBDA Sept 2022

Submitted by:

Project Team 12

Mohd Afraaz Firoz Khan - 220940325042

Shubham Chaudhari - 220940325069

Rahul Joshi - 220940325053

Aishwarya Bhalbhar - 220940325005

Mandar Ghaisas - 220940325041

1. INTRODUCTION

Agriculture is one of the important occupation practiced in India. It is the broadest economic sector and plays a most important role in the overall development of the country. More than 60% of the land in the country is used for agriculture in order to suffice the needs of 1.3 billion people Thus adopting new agriculture technologies is very important. This will be leads the farmers of our country towards profit [1]. Prior crop prediction and yield prediction was performed on the basis of farmers experience on a particular location. They will prefer the prior or neighborhood or more trend crop in the surrounding region only for their land and they don't have enough of knowledge about soil nutrients content such as nitrogen, phosphorus, potassium in the land. Being this as the current situation without the rotation of the crop and apply an inadequate amount of nutrients to soil it leads to reduce in the yield and soil pollution (soil acidification) and damages the top layer. Considering all these problems takes into the account we designed the system using a machine learning for betterment of the farmer. Machine learning(ML) is a game changer for agriculture sector. Machine learning is the part of artificial intelligence, has emerged together with bigdata technologies and high-performance computing to create new opportunities for data intensive science in the multi-disciplinary agri-technology domain. In the Agriculture field machine learning for instance is not a mysterious trick or magic, it is a set of well define model that collect specific data and apply specific algorithms to achieve expected results

The designed system will recommend the most suitable crop for particular land. Based on weather parameter and soil content such as Rainfall, Temperature, Humidity and pH.

A crop recommendation system is a powerful tool that uses data analytics and machine learning algorithms to provide farmers with tailored advice on the best crops to plant based on factors such as soil type, climate, and market demand. With the global population projected to reach 9.7 billion by 2050, the demand for food will continue to increase, and farmers will need to produce more crops with fewer resources. A crop recommendation system can help farmers optimize their yields, increase their profits, and reduce their environmental impact by recommending the most suitable crop varieties and providing guidance on best practices for planting, fertilizing, and harvesting. This technology has the potential to revolutionize the agricultural industry and contribute to the sustainable development of agriculture around the world.

Crop recommendation and crop disease identification by image is an innovative technology that uses machine learning and image recognition algorithms to help farmers make informed decisions about their crops. With the increasing demand for food and the unpredictable nature of weather patterns, it is essential for farmers to optimize their crop yields and reduce the risk of crop failure. This technology enables farmers to identify crop diseases quickly and accurately by analyzing images of the crops, allowing them to take appropriate action to prevent further damage. In addition, crop recommendation systems can provide farmers with personalized advice on the best crop varieties to plant based on factors such as soil type, climate, and market demand. By using this technology, farmers can increase their productivity, reduce waste, and maximize profits while also contributing to the sustainability of agriculture. Overall, crop recommendation and crop disease identification by image has the potential to revolutionize the way farmers manage their crops and ensure food security for the growing global population.

Models used in this project :

1. SVC
2. RANDOM FOREST
3. DECISION TREE
4. LOGISTIC REGRESSION
5. GAUSSION NAÏVE BAYES

2. PROBLEM STATEMENT

The problem statement of the project is to address the issue of declining soil fertility caused by changes in soil properties, temperature, and humidity, which results in reduced crop yields and decreased cultivation. To overcome this problem, the project aims to recommend the best crop to plant in a particular soil and temperature conditions, which will ultimately lead to high yield and increased cultivation. Additionally, the project seeks to tackle the problem of plant disease by utilizing image analysis techniques to identify and diagnose plant diseases.

Farmers around the world face a significant challenge in maintaining the health and productivity of their crops. Crop diseases can quickly spread and cause extensive damage, leading to reduced yields, higher expenses, and decreased profits. Identifying crop diseases early is crucial for effective control measures and preventing further spread, but it can be challenging, particularly for small-scale farmers who may lack access to specialized knowledge and resources. Additionally, farmers may struggle with deciding which crop varieties to plant, given the variable conditions they face. They need access to accurate and personalized recommendations on the best crop varieties and cultivation practices to optimize yields while minimizing environmental impact. As a result, there is a pressing need for a crop recommendation and crop disease identification system that leverages machine learning and image recognition technologies to provide farmers with accurate, real-time information about their crops' health and the best practices for cultivation. Such a system would help farmers make more informed decisions about crop selection, identify and treat crop diseases quickly and accurately, and maximize yields while reducing environmental impact.

3. LITERATURE SURVEY

3.1 Introduction

"What other people think" has always been an important piece of information for most of us during the decision-making process. The Internet and the Web have now (among other things) made it possible to find out about the opinions and experiences of those in the vast pool of people that are neither our personal acquaintances nor well-known professional critics — that is, people we have never heard of. And conversely, more and more people are making their opinions available to strangers via the Internet.

3.2 EXISTING METHODS

3.2.1 Crop yield prediction and recommendation

1. "Crop yield prediction and recommendation: a review" by Sarfraz et al. (2021). This paper presents a comprehensive review of recent research on crop yield prediction and recommendation systems. The authors discuss the various data-driven approaches and machine learning algorithms used in these systems and evaluate their effectiveness in optimizing crop yields.
2. "Crop Disease Diagnosis and Recommendation System: A Review" by M. S. Hassan et al. (2021) - This paper provides a comprehensive review of recent advances in crop disease diagnosis and recommendation systems, including those based on image processing and machine learning techniques.
3. "A survey on deep learning techniques for image classification in agriculture" by V. K. Gupta et al. (2021) - This paper provides a detailed overview of deep learning techniques for image classification in agriculture, including applications for crop disease detection.
4. "Image-Based Plant Disease Detection: A Review of Recent Developments and Future Directions" by X. Zhang et al. (2021) - This paper provides an overview of recent developments in image-based plant disease detection, including the use of machine learning algorithms, and discusses future research directions.
5. "Crop Diseases Detection and Classification Based on Deep Learning: A Review" by Y. Wang et al. (2020) - This paper provides a comprehensive review of deep learning techniques for crop disease detection and classification, including those based on image processing and convolutional neural networks.
6. "Recent advances and challenges in machine learning-based plant disease detection: A review" by K. Ntanos et al. (2020) - This paper provides a review of recent advances in machine learning-based plant disease detection, including applications of deep learning and computer vision techniques.

Overall, these literature surveys and studies provide a broad overview of recent advances and challenges in crop disease detection and image prediction, highlighting the potential of these approaches for improving crop management and food security

3.2.2 Machine learning algorithm for prediction

In our system we used supervised machine learning algorithm having subcategories as classification and regression. Classification algorithm will be most suitable for our system Machine learning algorithm for crop disease and image recommendations

There are several machine learning algorithms that can be used for crop disease and image recommendations, depending on the specific problem and available data. Here are a few examples:

Convolutional Neural Networks (CNNs): CNNs are a type of deep learning algorithm that are commonly used for image recognition tasks. They work by applying a series of filters to an input image, extracting features at different levels of abstraction. CNNs have been shown to be effective for identifying and classifying different types of crop diseases based on visual symptoms.

Decision Trees: Decision trees are a type of algorithm that work by recursively partitioning the input data into smaller and smaller subsets, based on the most important features. They are often used for classification tasks, such as predicting which crops are most likely to be affected by a particular disease. Decision trees are easy to interpret and can be useful for generating actionable insights for farmers.

Random Forests: Random forests are an extension of decision trees that work by creating an ensemble of multiple decision trees, each trained on a different subset of the data. They are often used for classification tasks, such as predicting the likelihood of a crop disease outbreak based on historical data. Random forests can be more robust than individual decision trees, as they are less prone to overfitting.

Support Vector Machines (SVMs): SVMs are a type of algorithm that work by finding the hyperplane that best separates the input data into different classes. They are often used for classification tasks, such as predicting which crops are most likely to be affected by a particular disease. SVMs can be effective when the input data is high-dimensional and nonlinear.

Overall, the choice of machine learning algorithm will depend on the specific problem and available data. It is often useful to compare the performance of multiple algorithms and select the one that achieves the best results on the given task.

4. LIBRARIES USED

1. Pandas Datareader

The Pandas datareader is a sub package that allows one to create a dataframe from various internet datasources, currently including:

- Disease detection dataset(Kaggle)
- Fertilizer prediction dataset (Kaggle)
- Crop recommendation dataset (Kaggle)
- Government of India (Agricultural Department)
- Geo Data

Datareader basic example

```
from pandas_datareader import data

df = pd.read_csv(‘dataset/Test_farmmitra_Dataset.csv’)
```

2. Pandas

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data.

Pandas gives you answers about the data. Like:

- Is there a correlation between two or more columns?
- What is average value?
- Max value?
- Min value?

Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called *cleaning* the data.

3. Numpy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

4. Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

5. Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Important features of scikit-learn:

- Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
- Accessible to everybody and reusable in various contexts.
- Built on the top of NumPy, SciPy, and matplotlib.
- Open source, commercially usable – BSD license.

6.Seaborn

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

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

Statistical analyses require knowledge about the distribution of variables in your dataset. The seaborn function displot() supports several approaches to visualizing distributions. These include classic techniques like histograms and computationally-intensive approaches like kernel density estimation. Seaborn also tries to promote techniques that are powerful but less familiar, such as calculating and plotting the empirical cumulative distribution function of the data

5. ALTERNATE MODELS and THEIR DISADVANTAGES

5.1 Support Vector Classifier (SVC):

Support Vector Machines (SVM) are popularly and widely used for classification problems in machine learning.
It tries to find a line/hyperplane (in multidimensional space) that separates these two classes. Then it classifies the new point depending on whether it lies on the positive or negative side of the hyperplane depending on the classes to predict.

Disadvantages:

- Long training time for large datasets.
- Difficult to understand and interpret the final model, variable weights and individual impact.
- Since the final model is not so easy to see, we can not do small calibrations to the model hence its tough to incorporate our business logic.

5.2 Random Forest (RF):

It is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification.

Disadvantages:

- **Complexity:** Random Forest creates a lot of trees (unlike only one tree in case of decision tree) and combines their outputs. By default, it creates 100 trees in Python sklearn library. To do so, this algorithm requires much more computational power and resources. On the other hand decision tree is simple and does not require so much computational resources.
- **Longer Training Period:** Random Forest require much more time to train as compared to decision trees as it generates a lot of trees (instead of one tree in case of decision tree) and makes decision on the majority of votes.

5.3 Decision Tree :

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

Decision Tree Terminology:

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

<u>Disadvantages:</u>

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.

- It may have an overfitting issue, which can be resolved using the Random Forest algorithm.

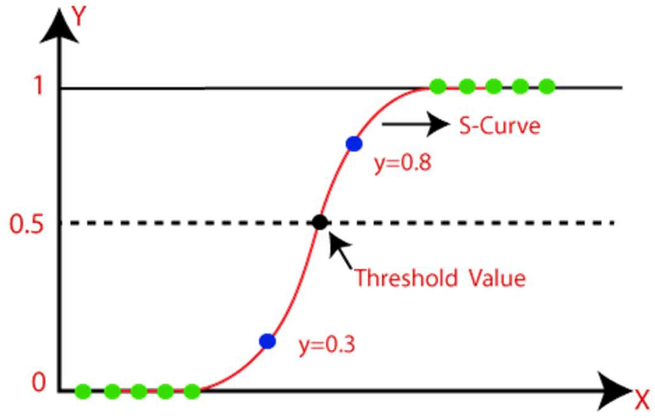
5.4 Logistic Regression:

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification.



5.5 Gaussian Naïve Bayes :

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

It is mainly used in text classification that includes a high-dimensional training dataset.

Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles**.**

Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Advantages Naïve Bayes Classifier :

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

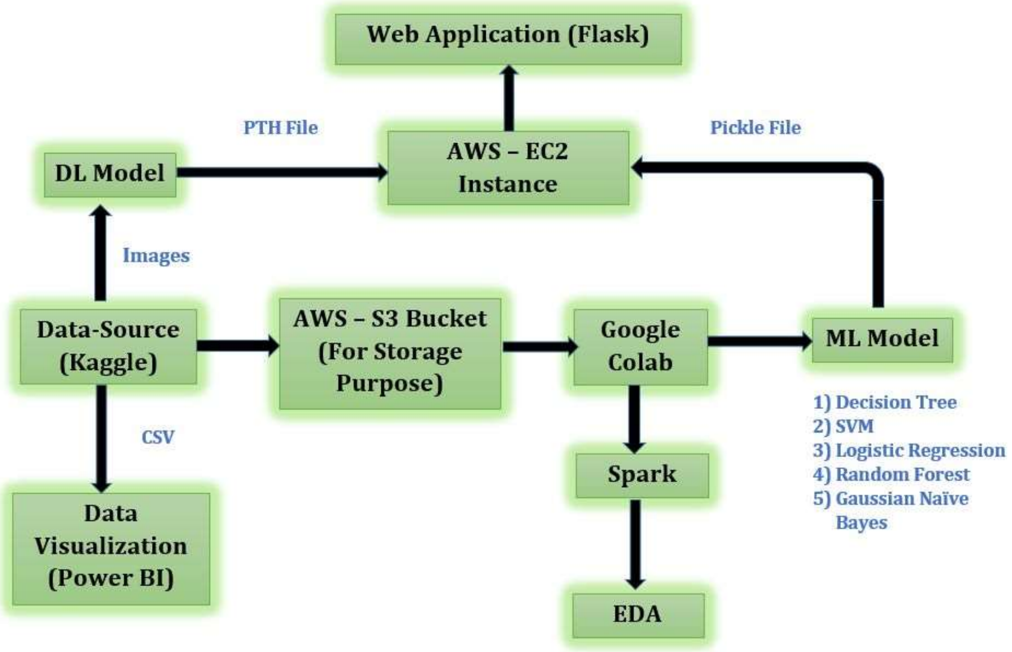
Disadvantage Naïve Bayes Classifier

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

Gaussian: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.

FLOW CHART

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.



PROJECT ARCHITECTURE

1) Preprocessing of data

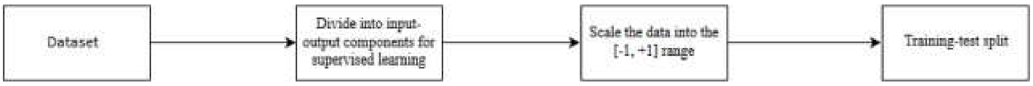

Dataset
Divide into input-output components for supervised learning
Scale the data into the [-1, +1] range
Training-test split

Fig :Pre-processing of data

2) Overall Architecture

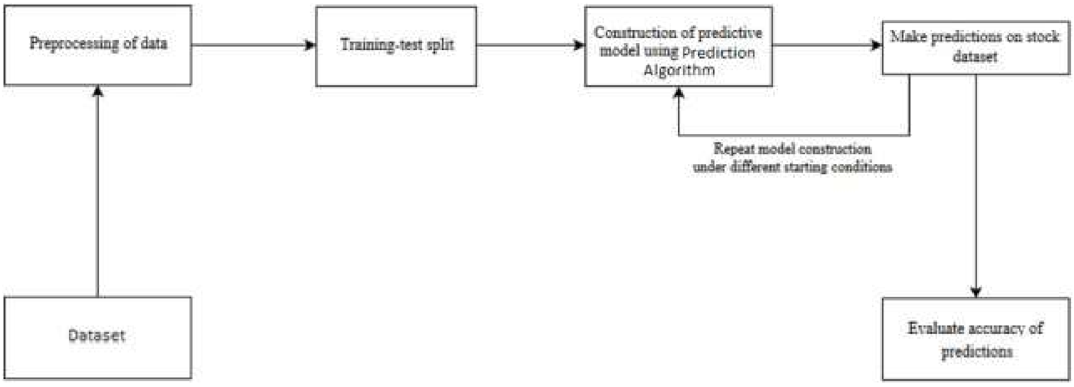
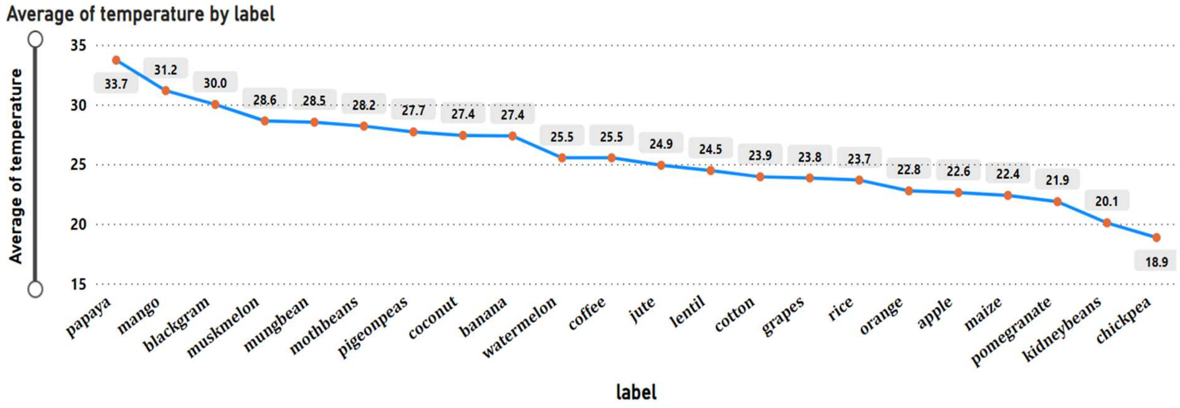

Preprocessing of data
Training-test split
Construction of predictive model using Prediction Algorithm
Make predictions on stock dataset
Repeat model construction under different starting conditions
Dataset
Evaluate accuracy of predictions

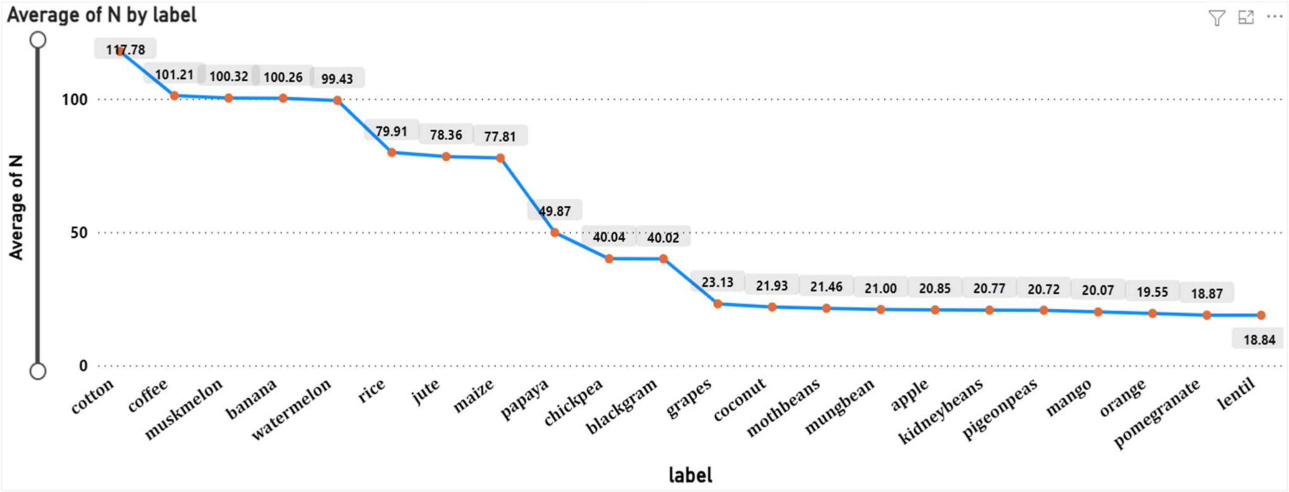
Fig : Overall Architecture

6. VISUALIZATIONS

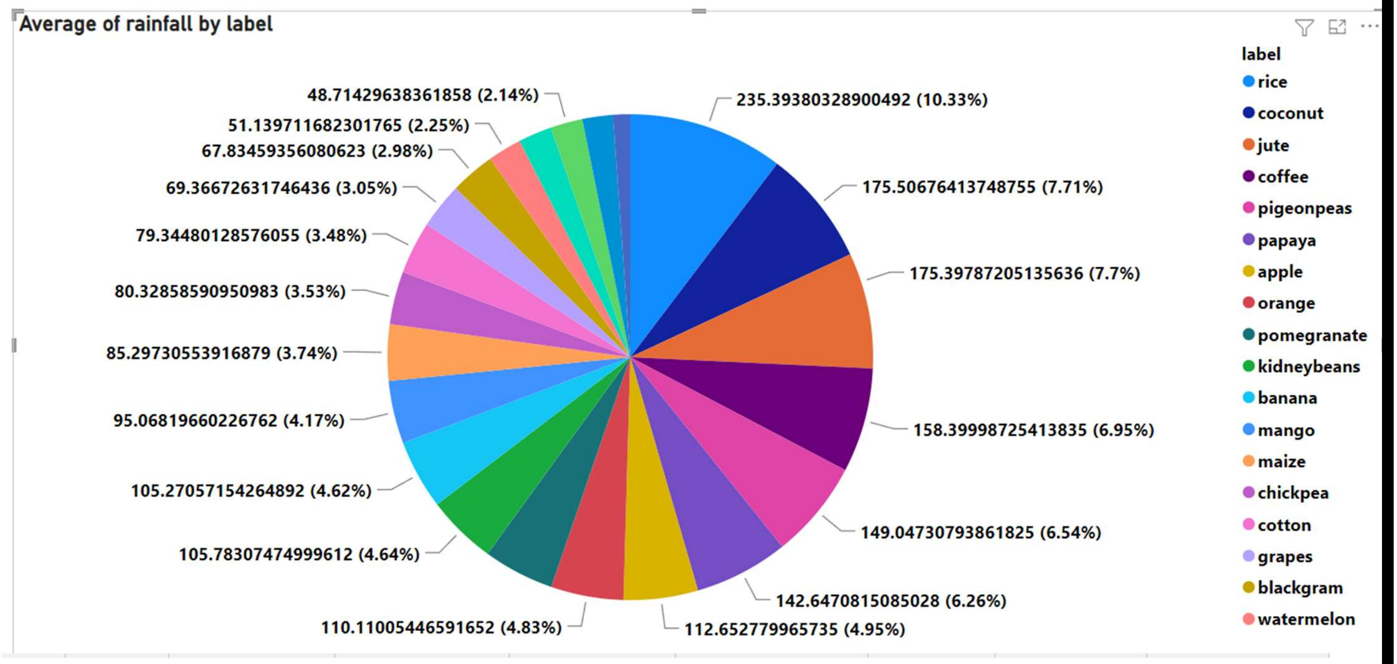
This is graph of average temperature against plants



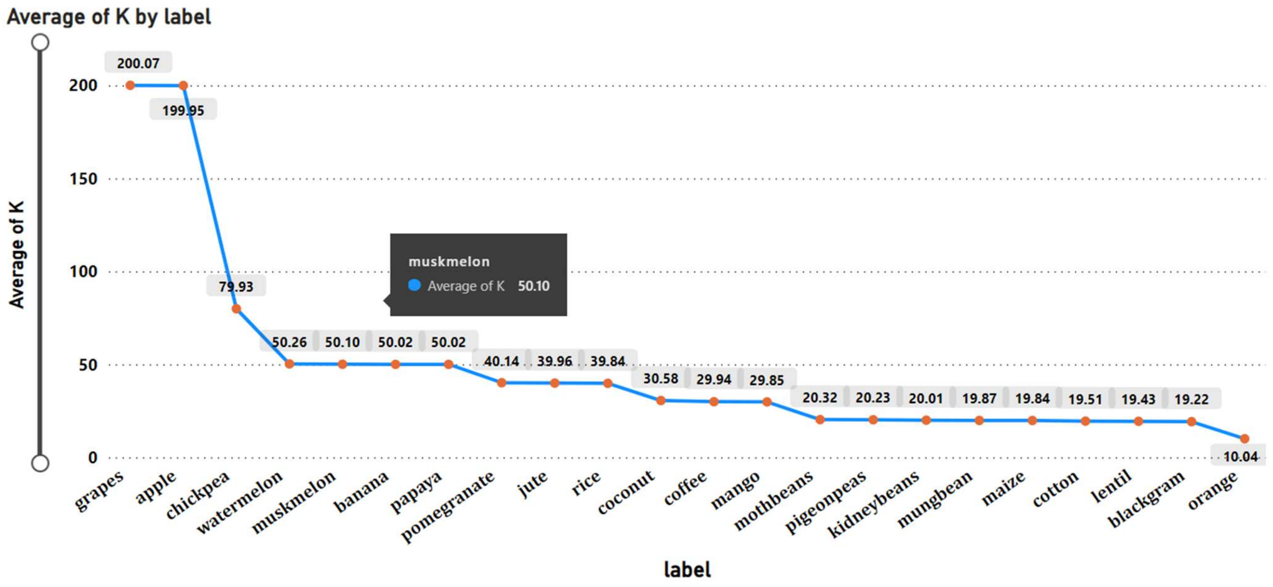
This graph shows various of average value of nitrogen to plants



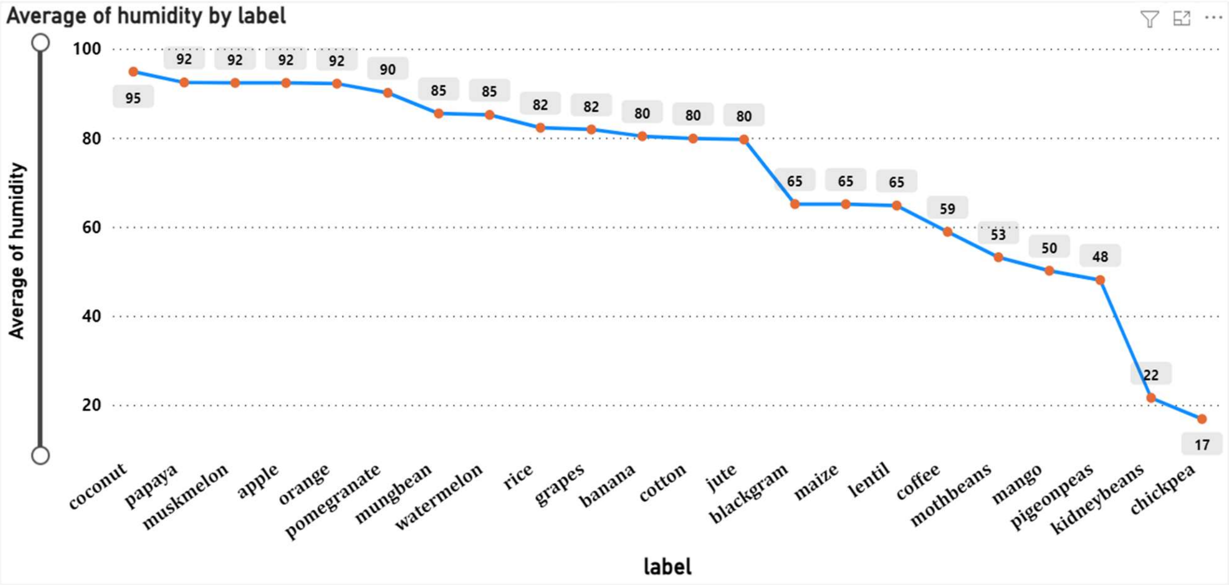
The average rainfall required for each plant



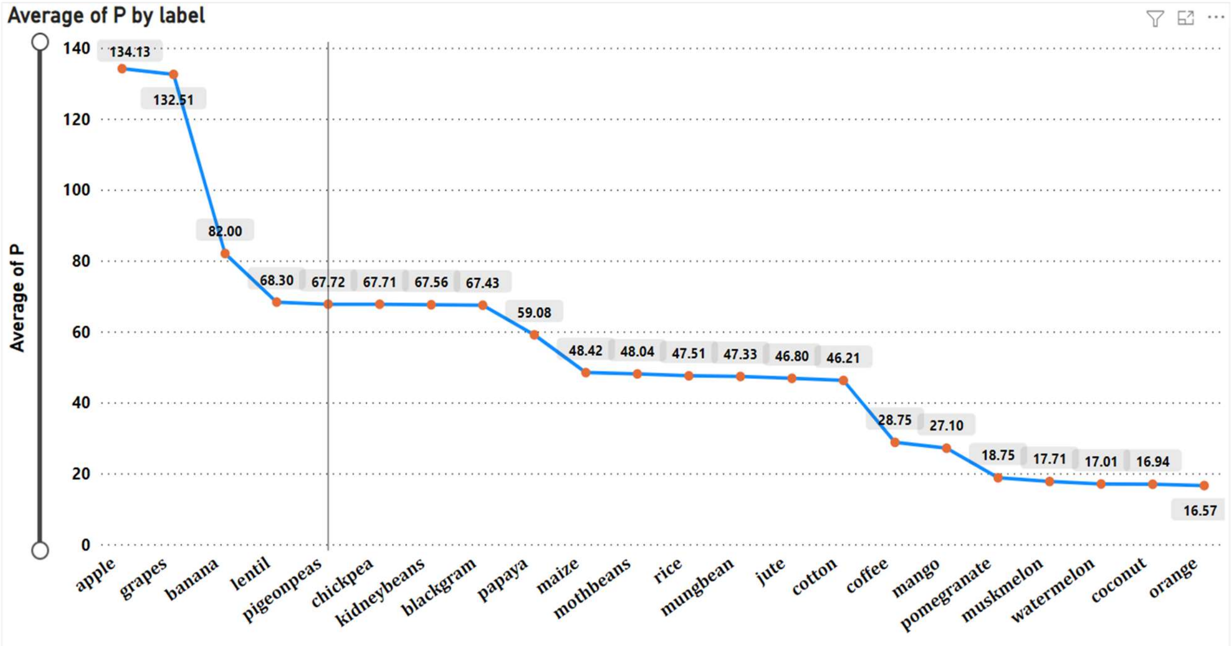
This is distribution of average potassium required for each type of plant



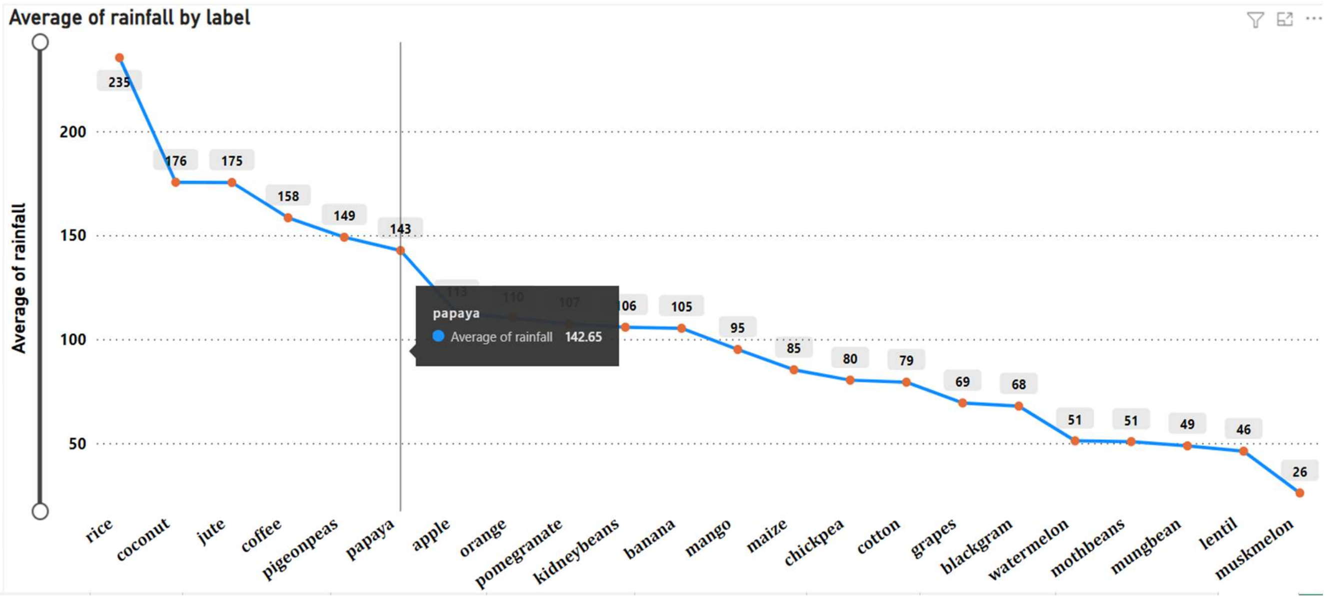
This is average humidity distribution against types of plants



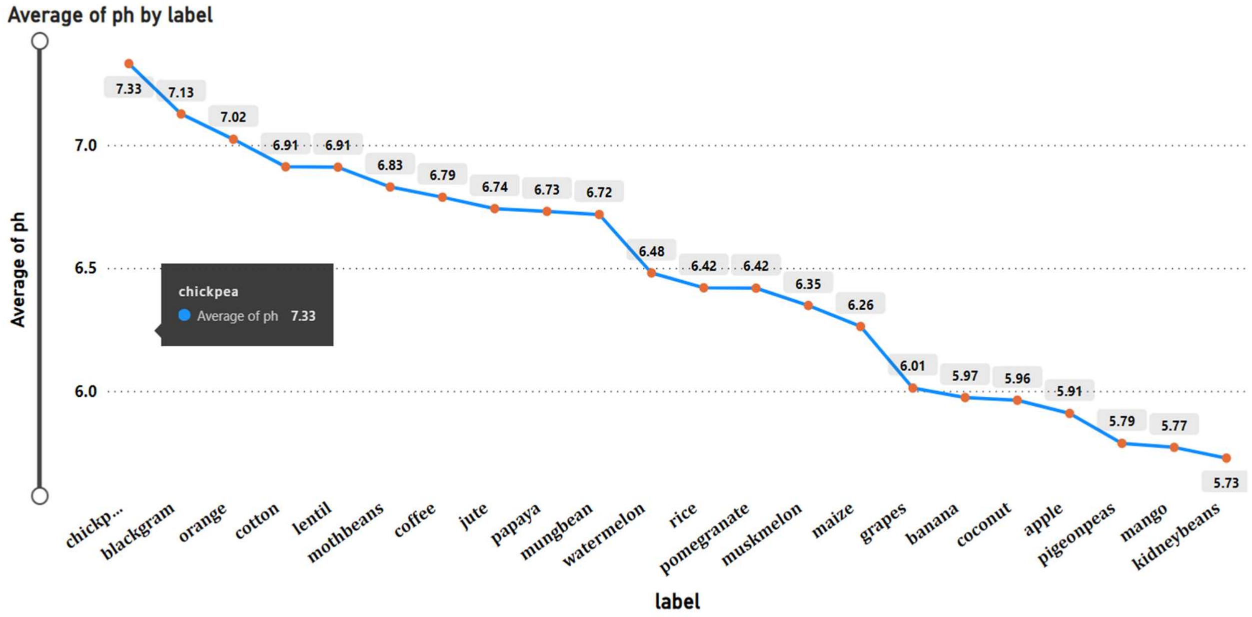
This is graph for average value of phosphorus against plants



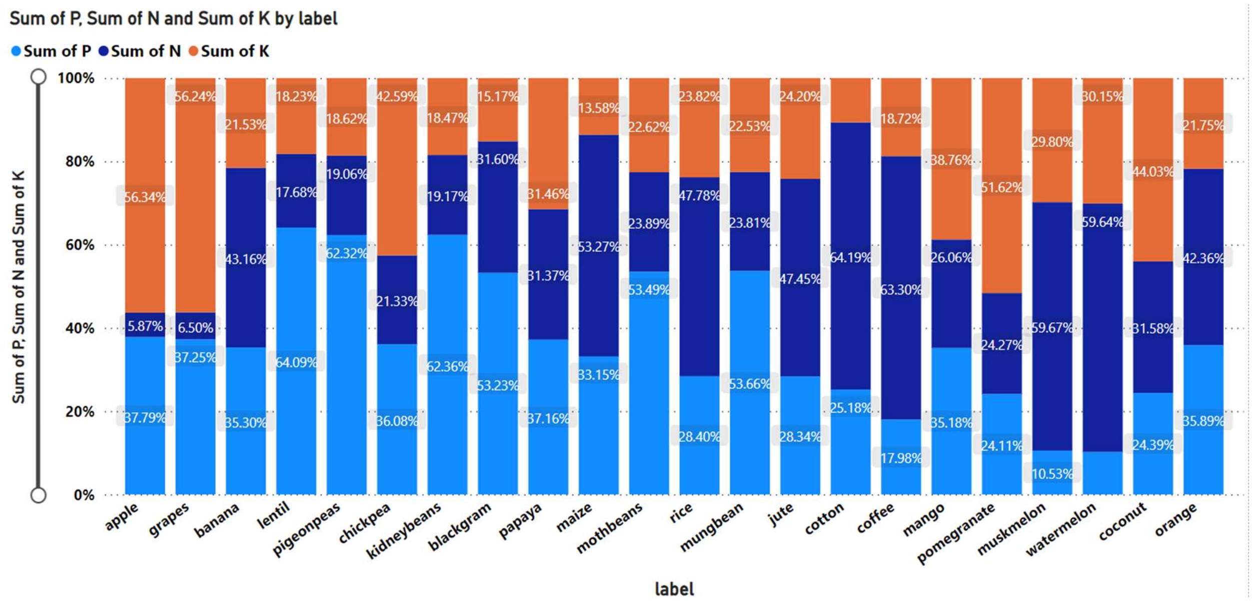
Distribution shows average rainfall required for each plant



Distribution for average of ph value required for each plant



This is stacked graph showing the contribution of important ingredient for growth of every plant



7. MODELING

Importing Libraries

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

Loading in DataFrame

```
df = pd.read_csv(‘Dataset/Test_farmmitra_Dataset.csv’)
df.head()
```

Ploting Heatmap of Data

```
sns.heatmap(df.corr(), annot=True)
```

Train Test Split

```
X = df.drop(['label'], axis=1).values #Leaving target
y = df['label'] #Laebl --> target
X
y
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
X_train.shape
X_test.shape
y_train.shape
y_test.shape
```

Data Preprocessing

```
df.rename(columns={'N': 'Nitrogen', 'P': 'Phosphorous', 'K': 'Potassium'}, inplace=True)
df['label'].unique()
len(df['label'].unique())
# The Number of unique values are 22
df[df.duplicated() == True].count()
```

Model Creation

```
#Creating list to save model and its accuracy
model = []
accuracy = []
dict_accuracy_of_model = {}

features = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]
target = df['label']
label = df['label']
```

1) Decision Tree:

```
from sklearn.tree import DecisionTreeClassifier
Decisiontree = DecisionTreeClassifier(criterion="entropy",random_state=2,max_depth=5)
model_decision_tree = Decisiontree.fit(X_train, y_train)

#Predicting values of test data i.e X_test
y_pred_decisiontree = model_decision_tree.predict(X_test)

decision_test_df = pd.DataFrame(y_test)

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
dtree_accuracy = accuracy_score(y_test, y_pred_decisiontree)
dtree_accuracy
```

2) SVM:

```
from sklearn.svm import SVC
from sklearn.preprocessing import MinMaxScaler
norm = MinMaxScaler().fit(X_train)
X_train_norm = norm.transform(X_train)
X_test_norm = norm.transform(X_test)
svm = SVC(kernel='poly', degree=3, C=1)
svm.fit(X_train_norm, y_train)
predicted_values = svm.predict(X_test_norm)
x = accuracy_score(y_test, predicted_values)
model.append('SVM')
print("SVM's Accuracy is: ", x)
```

```
# Cross validation score (SVM)
score = cross_val_score(svm,features,target,cv=5)
score
```

3) Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
reg = LogisticRegression(random_state=2)
reg_model = reg.fit(X_train,y_train)
y_pred_reg = reg_model.predict(X_test)

x = accuracy_score(y_test, predicted_values)
print("Logistic Regression's Accuracy is: ", x)
print(classification_report(y_test,y_pred_reg))

score = cross_val_score(reg_model,features,target,cv=5)
score
dict_accuracy_of_model['Logistic Regression'] = x
dict_accuracy_of_model
```

4) Random Forrest

```
from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n_estimators=20, random_state=0)
RF.fit(X_train,y_train)
y_pred_rf = RF.predict(X_test)
x = accuracy_score(y_test, y_pred_rf)
print(x)
print(classification_report(y_test,y_pred_rf))
score = cross_val_score(RF,features,target,cv=5)
score
dict_accuracy_of_model['RF'] = x

import pickle
# Dump the trained Naive Bayes
classifier with Pickle
RF_pkl_filename = 'Models/RF.pkl'
# Open the file to save as pkl file
RF_Model_pkl =
open(RF_pkl_filename, 'wb')
pickle.dump(RF, RF_Model_pkl)
# Close the pickle instances
NB_Model_pkl.close()
```

```
from sklearn.naive_bayes import
GaussianNB
NaiveBayes = GaussianNB()
NaiveBayes.fit(X_train,y_train)
```

5) Gaussian Naïve Bayes

```
x_input=test_data[len(test_data)-n_steps:].reshape(1,-1)
x_input.shape
```

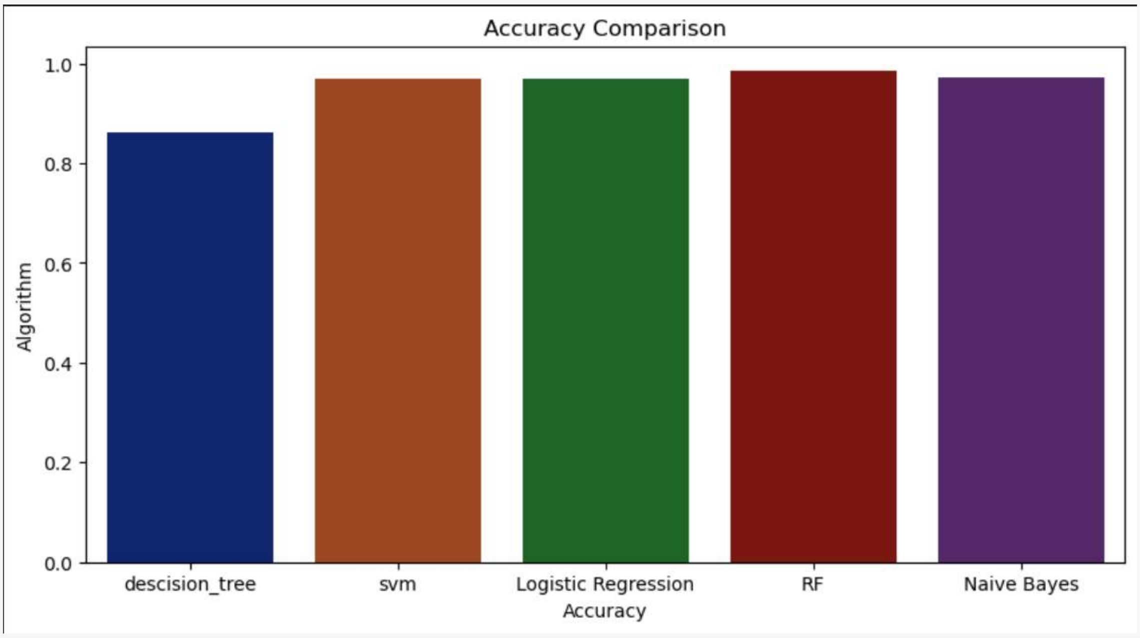
```
temp_input=list(x_input)
temp_input=temp_input[0].tolist()

NaiveBayes = GaussianNB()
NaiveBayes.fit(X_train,y_train)

y_pred_nb = NaiveBayes.predict(X_test)
x = accuracy_score(y_test, y_pred_nb)
print(x)

print(classification_report(y_test,y_pred_nb))

score = cross_val_score(NaiveBayes,features,target,cv=5)
score

dict_accuracy_of_model['Naive Bayes'] = x

import pickle
# Dump the trained Naive Bayes classifier with Pickle
NB_pkl_filename = 'Models/NB.pkl'
# Open the file to save as pkl file
NB_Model_pkl = open(NB_pkl_filename, 'wb')
pickle.dump(NaiveBayes, NB_Model_pkl)
# Close the pickle instances
NB_Model_pkl.close()
```

```
plt.figure(figsize=[10,5],dpi = 100)
plt.title('Accuracy Comparison')
```

```
plt.xlabel('Accuracy')
plt.ylabel('Algorithm')
sns.barplot(x = list(dict_accuracy_of_model.keys()),y =
list(dict_accuracy_of_model.values()),palette='dark')
```

```
descision_tree  -->  0.8618181818181818
svm  -->  0.9697272727272728
Logistic Regression  -->  0.9697272727272728
RF  -->  0.9854545454545455
Naive Bayes  -->  0.9711818181818181
```

Plotting of accuracy of models against various models

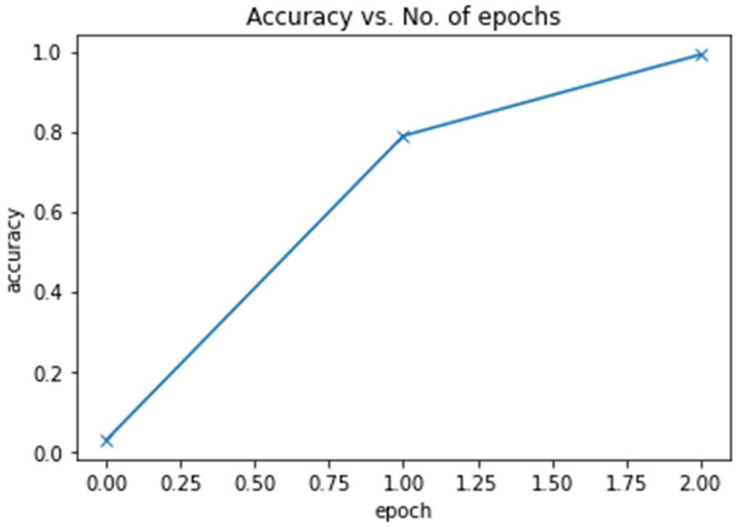


Disease Detection

Importing Libraries

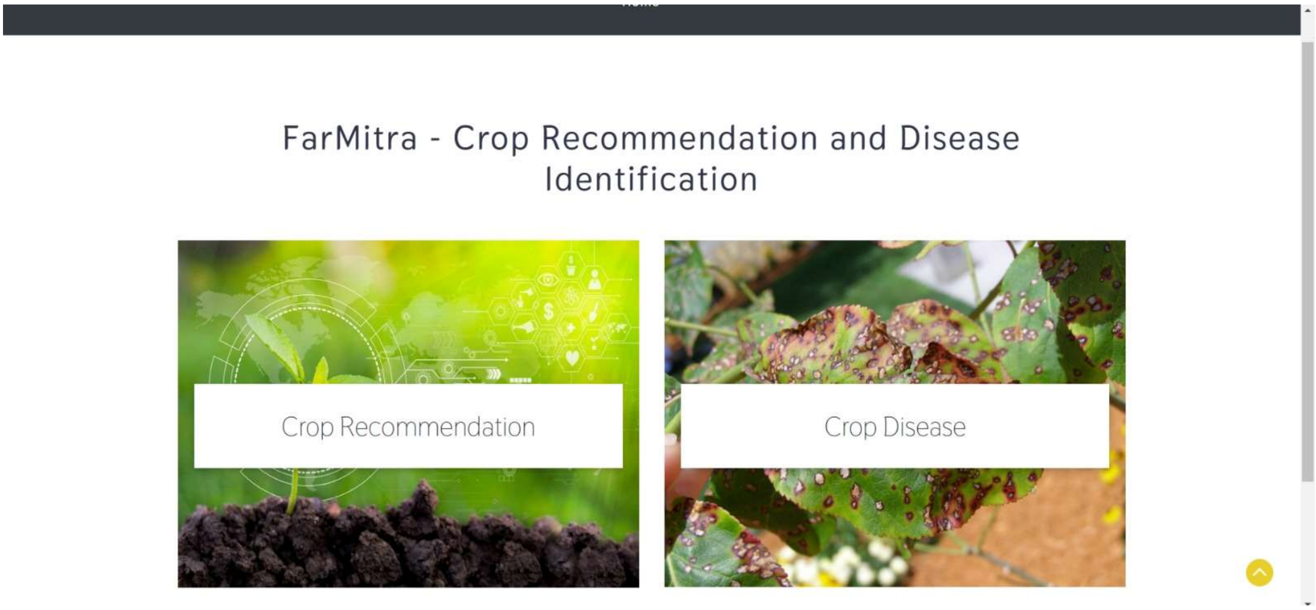
```
import os                      # for working with files
import numpy as np             # for numerical computationss
import pandas as pd            # for working with dataframes
import torch                   # Pytorch module
import matplotlib.pyplot as plt # for plotting informations on graph and images using tensors
import torch.nn as nn          # for creating  neural networks
from torch.utils.data import DataLoader # for dataloaders
from PIL import Image          # for checking images
import torch.nn.functional as F # for functions for calculating loss
import torchvision.transforms as transforms   # for transforming images into tensors
from torchvision.utils import make_grid       # for data checking
from torchvision.datasets import ImageFolder  # for working with classes and images
from torchsummary import summary              # for getting the summary of our model

# Architecture for training

# convolution block with BatchNormalization
def ConvBlock(in_channels, out_channels, pool=False):
    layers = [nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
             nn.BatchNorm2d(out_channels),
             nn.ReLU(inplace=True)]
    if pool:
        layers.append(nn.MaxPool2d(4))
    return nn.Sequential(*layers)

# resnet architecture
class ResNet9(ImageClassificationBase):
    def __init__(self, in_channels, num_diseases):
        super().__init__()

        self.conv1 = ConvBlock(in_channels, 64)
        self.conv2 = ConvBlock(64, 128, pool=True) # out_dim : 128 x 64 x 64
        self.res1 = nn.Sequential(ConvBlock(128, 128), ConvBlock(128, 128))

        self.conv3 = ConvBlock(128, 256, pool=True) # out_dim : 256 x 16 x 16
        self.conv4 = ConvBlock(256, 512, pool=True) # out_dim : 512 x 4 x 44
        self.res2 = nn.Sequential(ConvBlock(512, 512), ConvBlock(512, 512))

        self.classifier = nn.Sequential(nn.MaxPool2d(4),
                                       nn.Flatten(),
                                       nn.Linear(512, num_diseases))
    def forward(self, xb): # xb is the loaded batch
        out = self.conv1(xb)
        out = self.conv2(out)
        out = self.res1(out) + out
        out = self.conv3(out)
        out = self.conv4(out)
        out = self.res2(out) + out
        out = self.classifier(out)
        return out
# defining the model and moving it to the GPU
model = to_device(ResNet9(3, len(train.classes)), device)
model
# getting summary of the model
INPUT_SHAPE = (3, 256, 256)
print(summary(model.cuda(), (INPUT_SHAPE)))
# for training
@torch.no_grad()
def evaluate(model, val_loader):
    model.eval()
    outputs = [model.validation_step(batch) for batch in val_loader]
    return model.validation_epoch_end(outputs)


def get_lr(optimizer):
    for param_group in optimizer.param_groups:
        return param_group['lr']
def fit_OneCycle(epochs, max_lr, model, train_loader, val_loader, weight_decay=0,
                grad_clip=None, opt_func=torch.optim.SGD):
    torch.cuda.empty_cache()
    history = []
    optimizer = opt_func(model.parameters(), max_lr, weight_decay=weight_decay)
    # scheduler for one cycle learniing rate
    sched = torch.optim.lr_scheduler.OneCycleLR(optimizer, max_lr, epochs=epochs,
steps_per_epoch=len(train_loader))
```

```
for epoch in range(epochs):
        # Training
        model.train()
        train_losses = []
        lrs = []
        for batch in train_loader:
            loss = model.training_step(batch)
            train_losses.append(loss)
            loss.backward()

            # gradient clipping
            if grad_clip:
                nn.utils.clip_grad_value_(model.parameters(), grad_clip)

            optimizer.step()
            optimizer.zero_grad()

            # recording and updating learning rates
            lrs.append(get_lr(optimizer))
            sched.step()

        # validation
        result = evaluate(model, val_loader)
        result['train_loss'] = torch.stack(train_losses).mean().item()
        result['lrs'] = lrs
        model.epoch_end(epoch, result)
        history.append(result)

    return history

plot_accuracies(history)
```

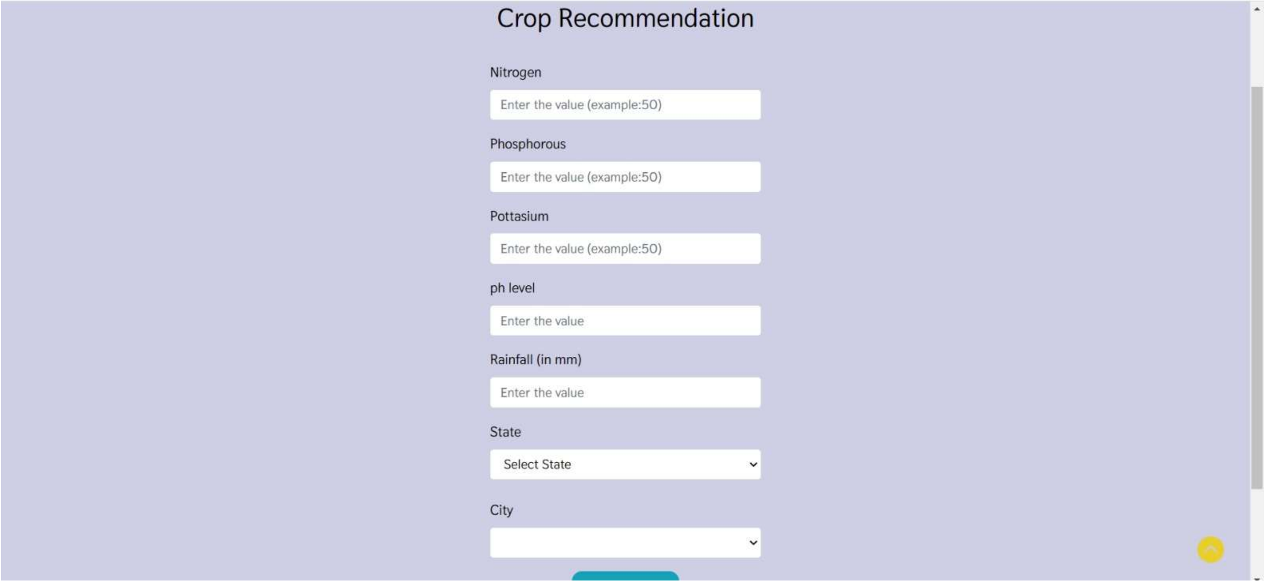


8.WEBSITE CREATION

The snapshots of our websites are attached below



FarMitra - Crop Recommendation and Disease Identification



Crop Recommendation

Nitrogen

Enter the value (example:50)

Phosphorous

Enter the value (example:50)

Pottasium

Enter the value (example:50)

ph level

Enter the value

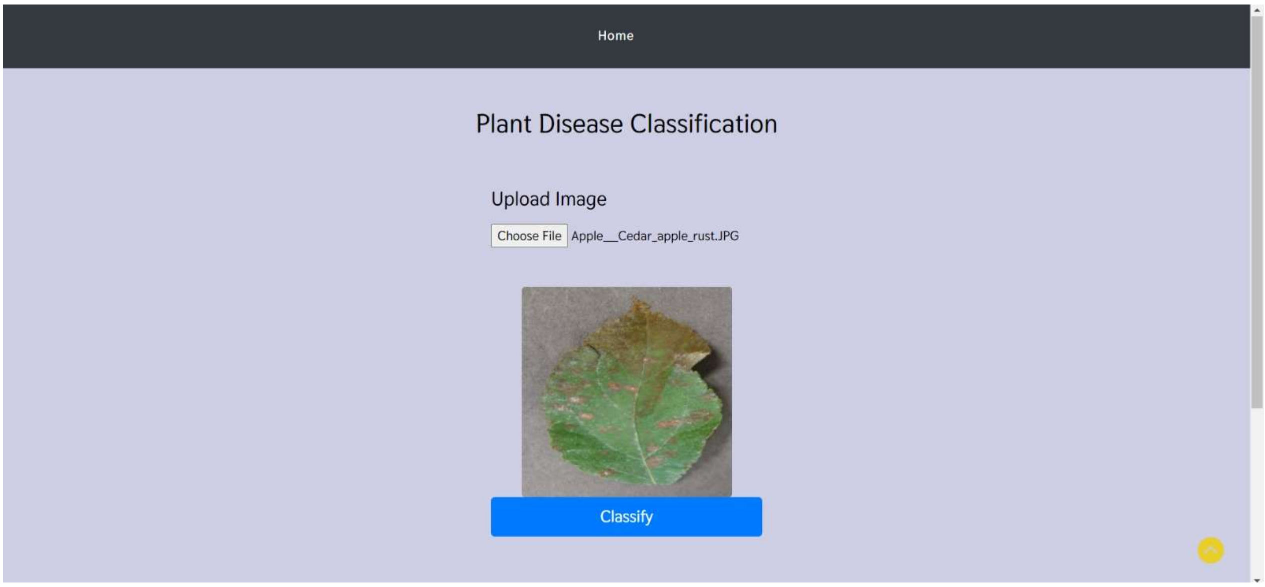
Rainfall (in mm)

Enter the value

State

Select State

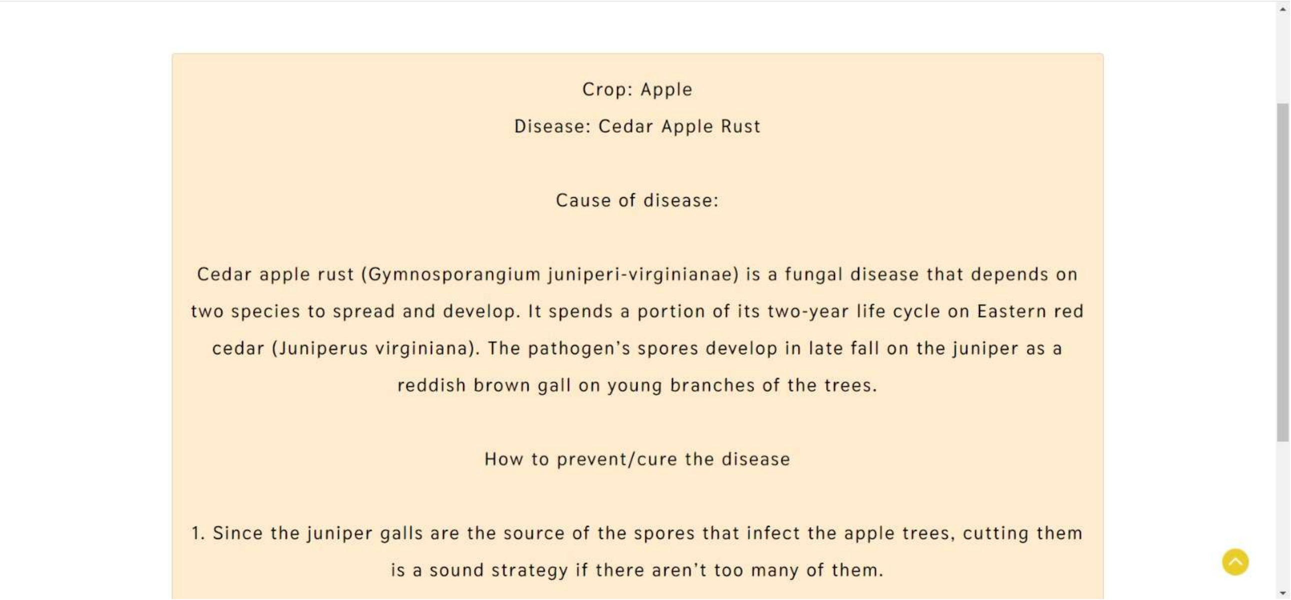
City



Plant Disease Classification

Upload Image

Choose File Apple__Cedar_apple_rust.JPG



Crop: Apple

Disease: Cedar Apple Rust

Cause of disease:

Cedar apple rust (Gymnosporangium juniperi-virginianae) is a fungal disease that depends on two species to spread and develop. It spends a portion of its two-year life cycle on Eastern red cedar (Juniperus virginiana). The pathogen's spores develop in late fall on the juniper as a reddish brown gall on young branches of the trees.

How to prevent/cure the disease

1. Since the juniper galls are the source of the spores that infect the apple trees, cutting them is a sound strategy if there aren't too many of them.

CONCLUSION & FUTURE SCOPE

Conclusion:

In conclusion, the Crop Recommendation and Disease Classification technologies are significant advancements in the field of agriculture. The Crop Recommendation technology utilizes various factors such as rainfall, temperature, location, and soil conditions to recommend the most suitable crop to plant, which can help farmers increase their yields and profitability. On the other hand, Disease Classification technology uses image analysis to classify the disease of a given plant, which is crucial in preventing and controlling crop diseases. Both technologies can contribute to sustainable agriculture by improving crop productivity and reducing crop losses. This project presented two subparts, including a crop recommendation system based on soil characteristics and a crop disease classification system based on image analysis. These subparts can be further developed and integrated into a comprehensive agricultural technology system to support farmers in making informed decisions and achieving optimal crop production.

Future Scope:

The Crop Recommendation and Disease Classification technologies have significant potential for future development and application in the field of agriculture.

In terms of Crop Recommendation, future research can focus on expanding the predictive capabilities of the technology by incorporating more data sources, such as satellite imagery and historical weather patterns. Additionally, the technology can be tailored to specific regions and crops to provide more accurate recommendations.

For Disease Classification, future research can focus on improving the accuracy of image analysis algorithms to detect a wider range of diseases and pests. The technology can also be integrated with sensors and other monitoring tools to provide real-time disease detection and prevention.

Furthermore, there is potential for the integration of these technologies into a comprehensive precision agriculture system that can optimize crop production and reduce environmental impact by minimizing the use of fertilizers, pesticides, and water.

Overall, the future scope of Crop Recommendation and Disease Classification technologies is vast, and ongoing research and development can contribute to the sustainable growth of the agriculture industry.

8. REFERENCES

- **Dataset**:

  https://www.kaggle.com/datasets/atharvingle/crop-recomendaation-dataset

- **Models:**

  1. SVC:
  https://www.analyticsvidhya.com/blog/2020/03/support-vector-classification-tutorial-for-machine-learning/

  2. Random Forrest:

  https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
  https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm

  3. Decision Tree:

  https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

  4. Logistic Regression:

  https://www.javatpoint.com/logistic-regression-in-machine-learning

  5. Gaussian Naïve Bayes:

  https://www.javatpoint.com/machine-learning-naive-bayes-classifier