CROP RECOMMENDATION SYSTEM

Data Science With Python Lab Project Report

Bachelor

in

Computer Science

By

Lenka Yasaswini and Yerra Bhargavi

S200281

S200286

Rajiv Gandhi University Of Knowledge And Technologies

S.M. Puram , Srikakulam -532410

Andhra Pradesh, India

Abstract

Crop recommendation is important for farmers to grow the right crops at right time .This project focuses on how to suggest the best crops for farmers depending on the several parameters . We consider parameters such as soil quality, climate, and market demand. By taking advice from our project , farmers can increase their yield and income. This approach aims to improve agricultural productivity and support sustainable farming practices.The main motto of our project is "Precision Agriculture." This crop recommendation system helps farmers make informed decisions about farming strategy. This system works based on different parameters that would affect the growth of any particular crop and suggests what crop to grow in a particular area. As it is clear, this system showcases exciting possibilities for increased crop productivity and expansion of the farming sector. Additionally, incorporating techniques such as crop rotation and intercropping can further enhance sustainability and resilience in agriculture. This research aims to empower farmers with the knowledge and tools needed to optimize their crop selection, leading to improved yields, and economic prosperity in farming communities.

Contents

Chapter 1

Introduction

1.1 Introduction Towards Our Project

In our modern world, where agriculture plays a crucial role in sustaining livelihoods, the need for efficient crop recommendations is necessary.However, farmers often face challenges in deciding which crops to cultivate due to factors like varying climates, soil conditions, and market demands. To address this, we're employing data science techniques to develop a crop recommendation system. By analyzing vast datasets encompassing weather patterns, soil health, historical yields, and market trends, our project aims to provide personalized and accurate crop recommendations to farmers. This project not only enhances agricultural productivity but also empowers farmers to make informed decisions, ultimately contributing to food security and economic prosperity in farming communities. Crop Recommendation over Parameters take:
N-Ratio of Nitrogen Content in soil,
P-Ratio of Phosporous Content in soil
K-Ratio of Potassium Content in soil
Temperature-Temperature in Degree Celsius
humidity - relative humidity in percentage
ph - ph value of the soil
rainfall - rainfall in mm

1.2 Applications

We all know India is an agricultural country. Agriculture is the main profession, but these days the situation of farmers is worsening day by day. We believe technology should always aid in the better advancement of humanity and mankind; our project aims for the same! It helps in:
*Producing good and healthy crops, which leads to better crop prices.
Expanding the scope of farming.
Eliminating food shortages, which can also make India a rich country in cultivation.

1.3 Motivation Towards our Project

When we hear the term ’farmer’ on any news channel, the main context is often about the loss of crops due to extreme rain and other weather conditions, as well as the plight of the farmer. Many farmers are attempting suicide due to severe crop losses caused by rain washing out entire crops, resulting in lower productivity and bankruptcy. We often face shortages in our daily needs due to the decreased production of such crops, leading to a vast increase in prices. Therefore, we are motivated by this project to reduce the suicide rate and increase crop productivity, ultimately eliminating shortages of any kind of product.

1.4 Problem Statement

Our project is used to recommend the most suitable crops for a particular farm based on various parameters. The project aims to develop a machine learning model that can predict the accurate crop over parameters. The dataset for this project is taken from the Kaggle website. This project will utilize data that consists of various parameters such as potassium content, nitrogen content, phosphorus content in soil, etc., and various crops. By analyzing the data and given input, the model should forecast an accurate crop using a better machine learning model.

Chapter 2

Approach To Your Project

2.1 Explain About Your Project

This project is about recommending or suggesting crops based on several parameters of the land.It will benefit farmers as well as it reduces the food scarcity.It helps farmers to cultivate not suitable crops.This helps India to become rich in agricultural sector.

2.2 Data Set

The Dataset for this Crop recommendation project is taken from Kaggle website.This Dataset contains some columns like N-Ratio of Nitrogen Content in soil,
P-Ratio of Phosporous Content in soil
K-Ratio of Potassium Content in soil
Temperature-Temperature in Degree Celsius
humidity - relative humidity in percentage
ph - ph value of the soil
rainfall - rainfall in mm

2.3 Prediction Technique

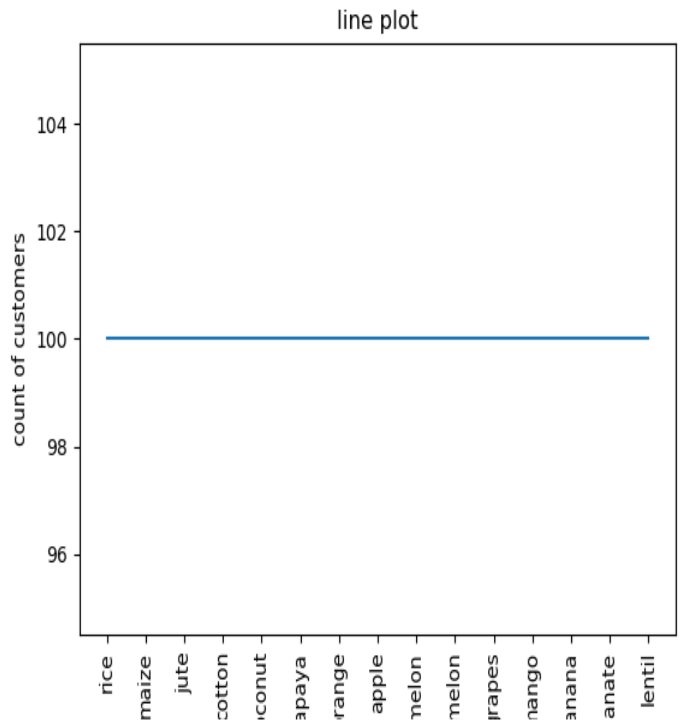
The prediction techniques we used are Linear Regression,Polynomial Regression Lasso Regression and RandomForestRegressor.Linear Regression is a model that shows the relation between dependent and independent variables.we select a suitable regression model that is RandomForest Regressor.we use Random forest as it predicts output with high accuracy,even for the large dataset as it runs efficiently.

2.4 Graphs

```
import matplotlib.pyplot as plt
import seaborn as sn
```

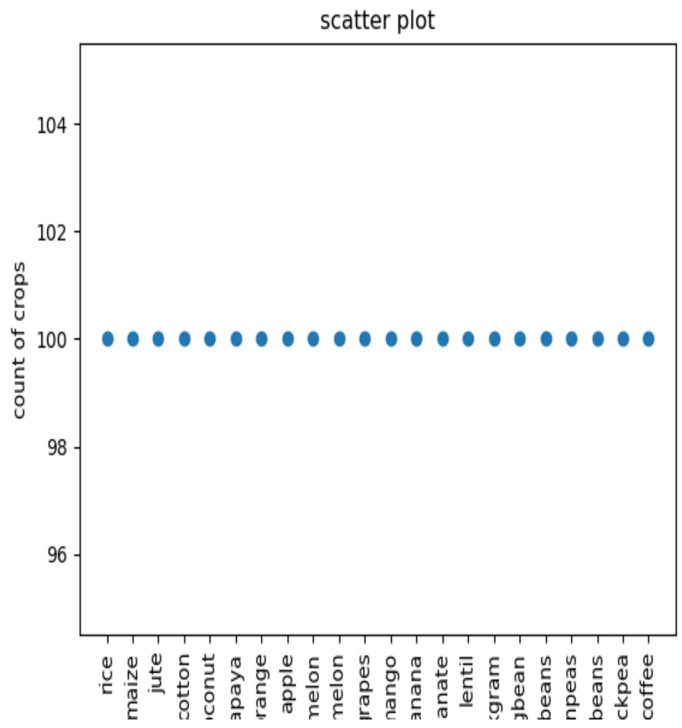
1.Line plot

```
label_value_counts1=label_value_counts.head(15)
plt.plot(label_value_counts1.index,label_value_counts1)
plt.title(’line plot’)
plt.xlabel(’crops’)
plt.ylabel(’count of crops’)
plt.xticks(rotation=90)
plt.show()
```



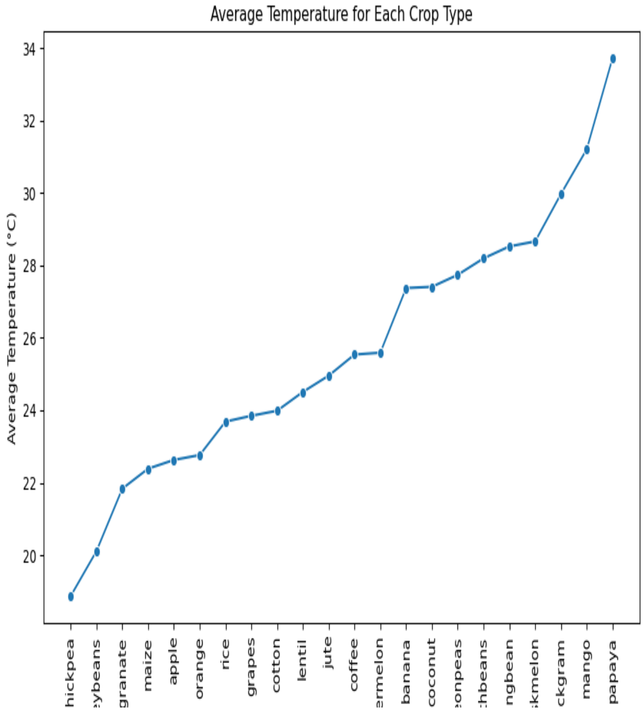
2.Scatter graph

```
plt.scatter(label_value_counts.index,label_value_counts)

plt.title(’scatter plot’)

plt.xlabel(’crops’)

plt.ylabel(’count of crops’)

plt.xticks(rotation=90)

plt.show()
```



3.Line plot

```
avg_temperature = df.groupby(’label’)[’temperature’].mean().sort_values()

sn.lineplot(x=avg_temperature.index, y=avg_temperature.values, marker=’o’)

plt.title(’Average Temperature for Each Crop Type’)

plt.xlabel(’Crop Type’)

plt.ylabel(’Average Temperature (°C)’)

plt.xticks(rotation=90)

plt.grid(True)

plt.show()
```



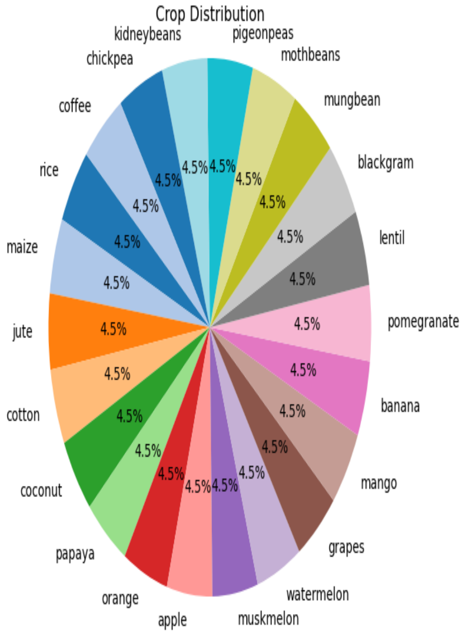
4.Pie Chart

crop_distribution = df['label'].value_counts() plt.pie(crop_distribution, labels=crop_distribution.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.tab2

plt.title('Crop Distribution')

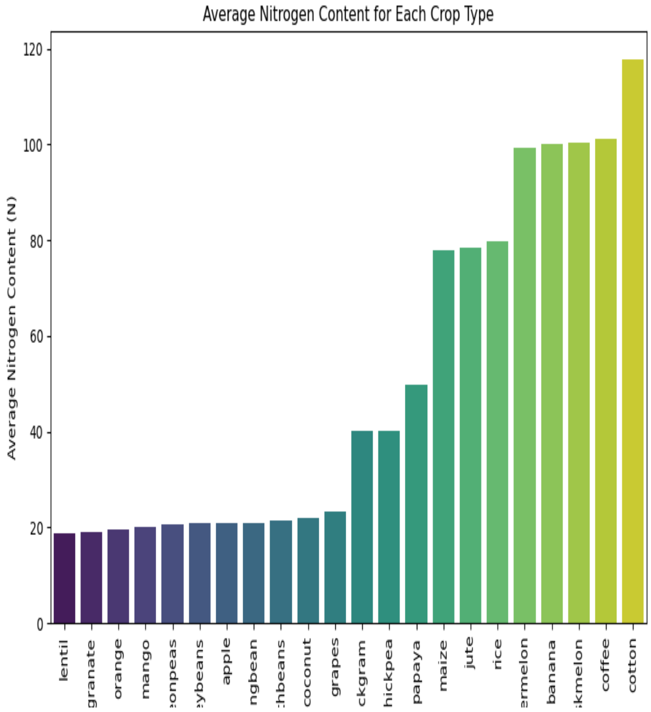
plt.axis('equal')

plt.show()



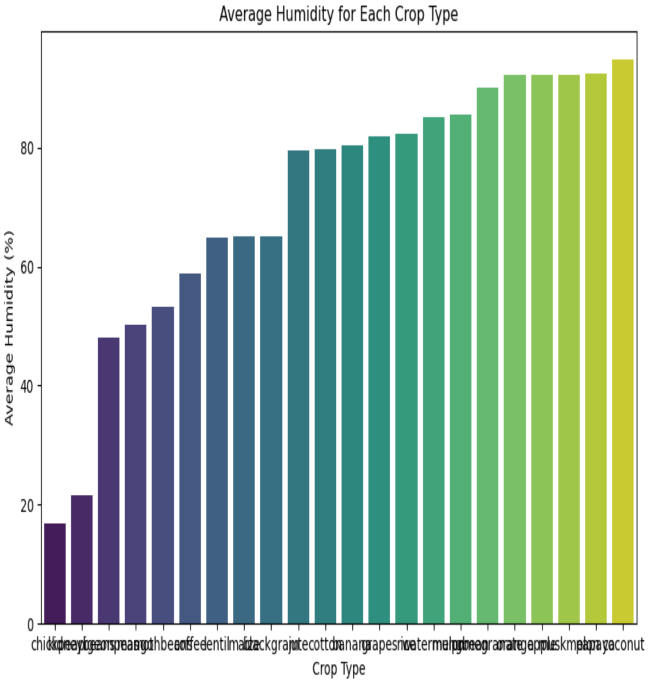
5.Bar Plot

```
avg_nitrogen = dfġroupby('label')['N'].mean().sort_values()

sn.barplot(x=avg_nitrogen.index, y=avg_nitrogen.values, palette='viridis')

plt.title('Average Nitrogen Content for Each Crop Type')

plt.xlabel('Crop Type')

plt.ylabel('Average Nitrogen Content (N)')

plt.xticks(rotation=90)

plt.grid(True)

plt.show()
```



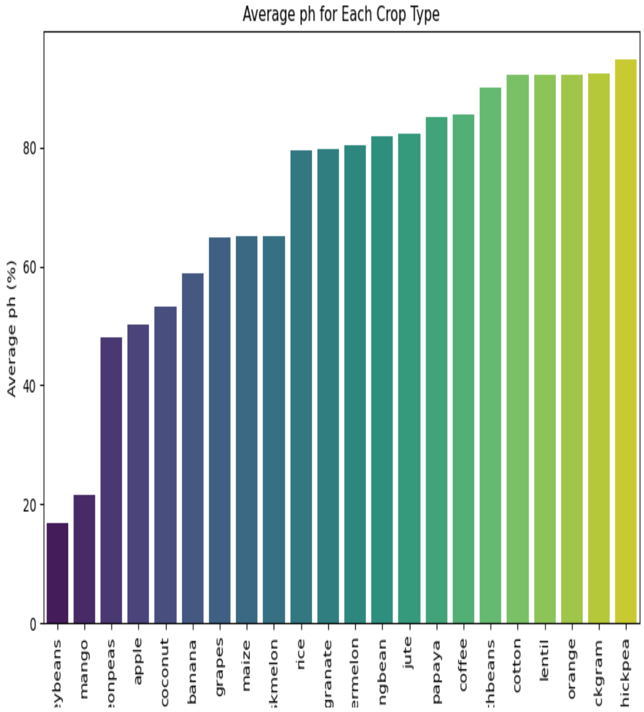
6.Bar Plot

```
avg_humidity = df.groupby(’label’)[’humidity’].mean().sort_values()

sn.barplot(x=avg_humidity.index, y=avg_humidity.values, palette=’viridis’)

plt.title(’Average Humidity for Each Crop Type’)

plt.xlabel(’Crop Type’)

plt.ylabel(’Average Humidity (plt.xticks(rotation=90)

plt.show()
```



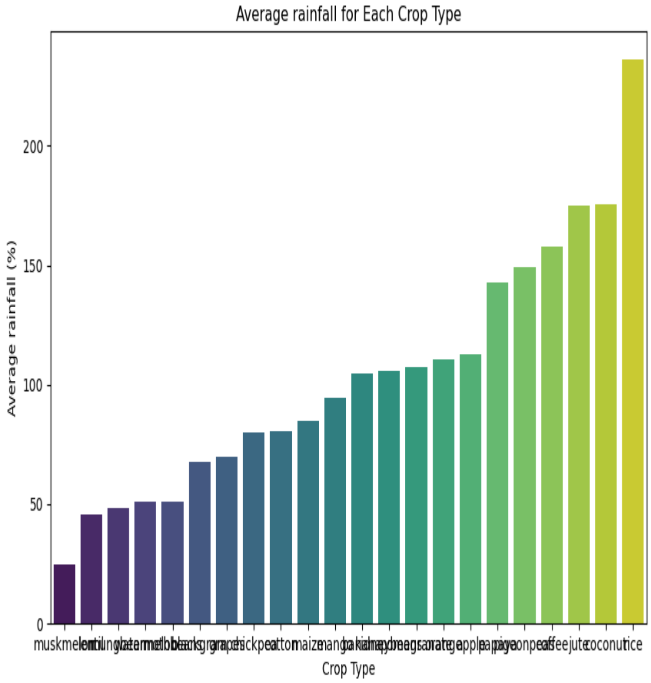
7.Bar Plot

```
avg_ph = df.groupby(’label’)[’ph’].mean().sort_values()
sn.barplot(x=avg_ph.index, y=avg_ph.values, palette=’viridis’)
plt.title(’Average ph for Each Crop Type’)
plt.xlabel(’Crop Type’)
plt.ylabel(’Average ph (%)’)
plt.xticks(rotation=90)
plt.show()
```



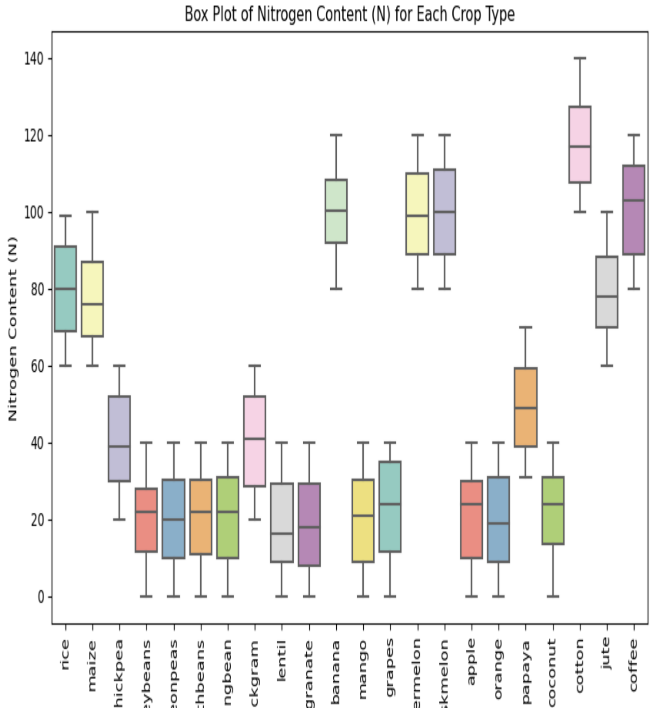
8.Bar Plot

```
avg_rainfall = df.groupby(’label’)[’rainfall’].mean().sort_values()
sn.barplot(x=avg_rainfall.index, y=avg_rainfall.values, palette=’viridis’)
plt.title(’Average rainfall for Each Crop Type’)
plt.xlabel(’Crop Type’)
plt.ylabel(’Average rainfall (%)’)
plt.xticks(rotation=90)
plt.show()
```



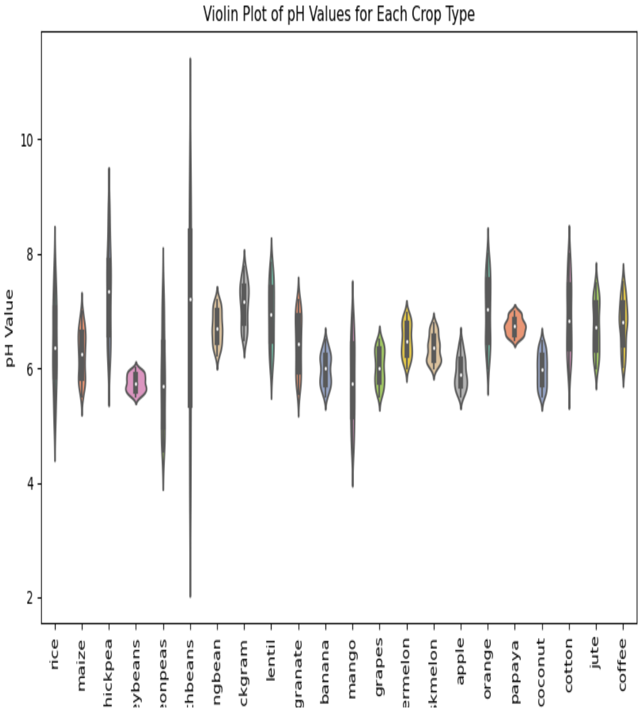
9.Box Plot

```
plt.figure(figsize=(10, 6))

sn.boxplot(x=’label’, y=’N’, data=df, palette=’Set3’)

plt.title(’Box Plot of Nitrogen Content (N) for Each Crop Type’)

plt.xlabel(’Crop Type’)

plt.ylabel(’Nitrogen Content (N)’)

plt.xticks(rotation=90)

plt.show()
```



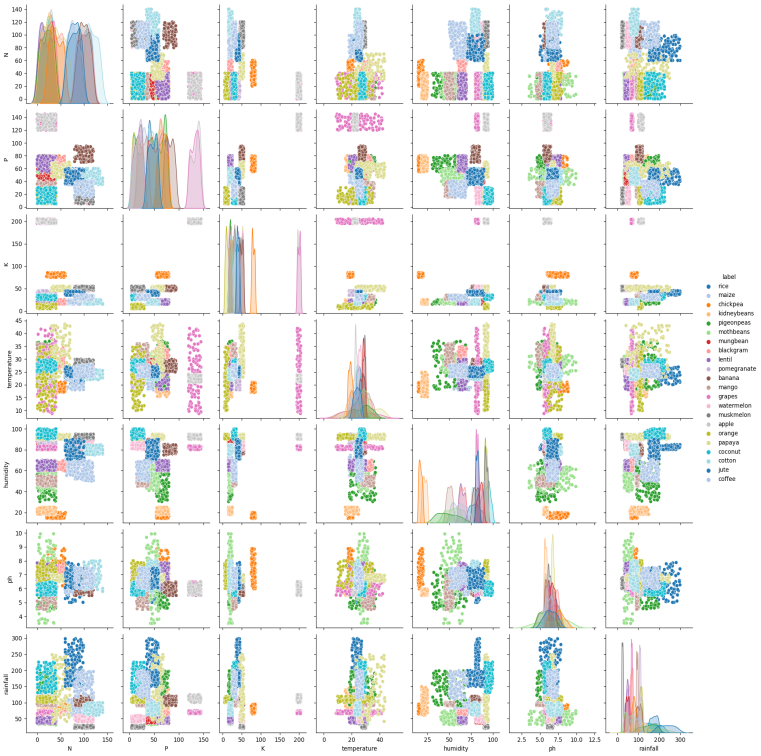
10.Violin Plot

```
plt.figure(figsize=(10, 6))

sn.violinplot(x=’label’, y=’ph’, data=df, palette=’Set2’)

plt.title(’Violin Plot of pH Values for Each Crop Type’)

plt.xlabel(’Crop Type’)

plt.ylabel(’pH Value’)

plt.xticks(rotation=90)

plt.show()
```



11.Pair Plot

```
sn.pairplot(df, hue='label', markers='o', palette='tab20')
plt.suptitle('Scatter Plot Matrix of Crop Parameters', y=1.02)
plt.show()
```


N
P
K
temperature
humidity
ph
rainfall
label
● rice
● maize
● chickpea
● kidneybeans
● pigeonpeas
● mothbeans
● mungbean
● blackgram
● lentil
● pomegranate
● banana
● mango
● grapes
● watermelon
● muskmelon
● apple
● orange
● papaya
● coconut
● cotton
● jute
● coffee

Chapter 3

Code

3.1 Pandas

• Pandas is a popular open-source library in Python.

• It is used for data manipulation and analysis.

• It has functions for analyzing, cleaning, exploring, and manipulating data.

• Pandas allows us to analyze big data and make conclusions based on statistical theories.

• Pandas can clean messy data sets, and make them readable and rele- vant.

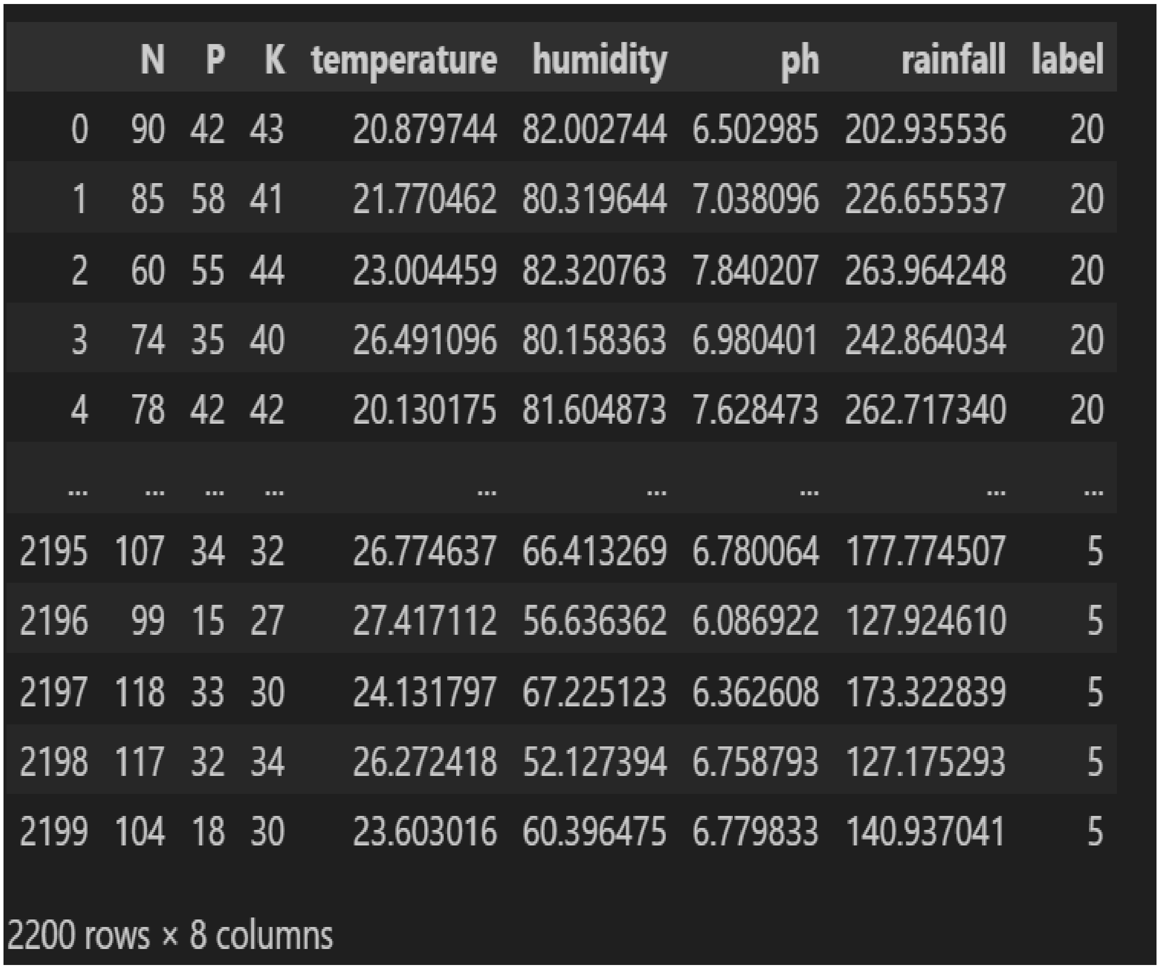
Importing Essential Libraries

```
import numpy as np
import pandas as pd
```

Importing csv file to jupyter notebook using Pandas

df=pd.read_csv(”data.csv”)

df



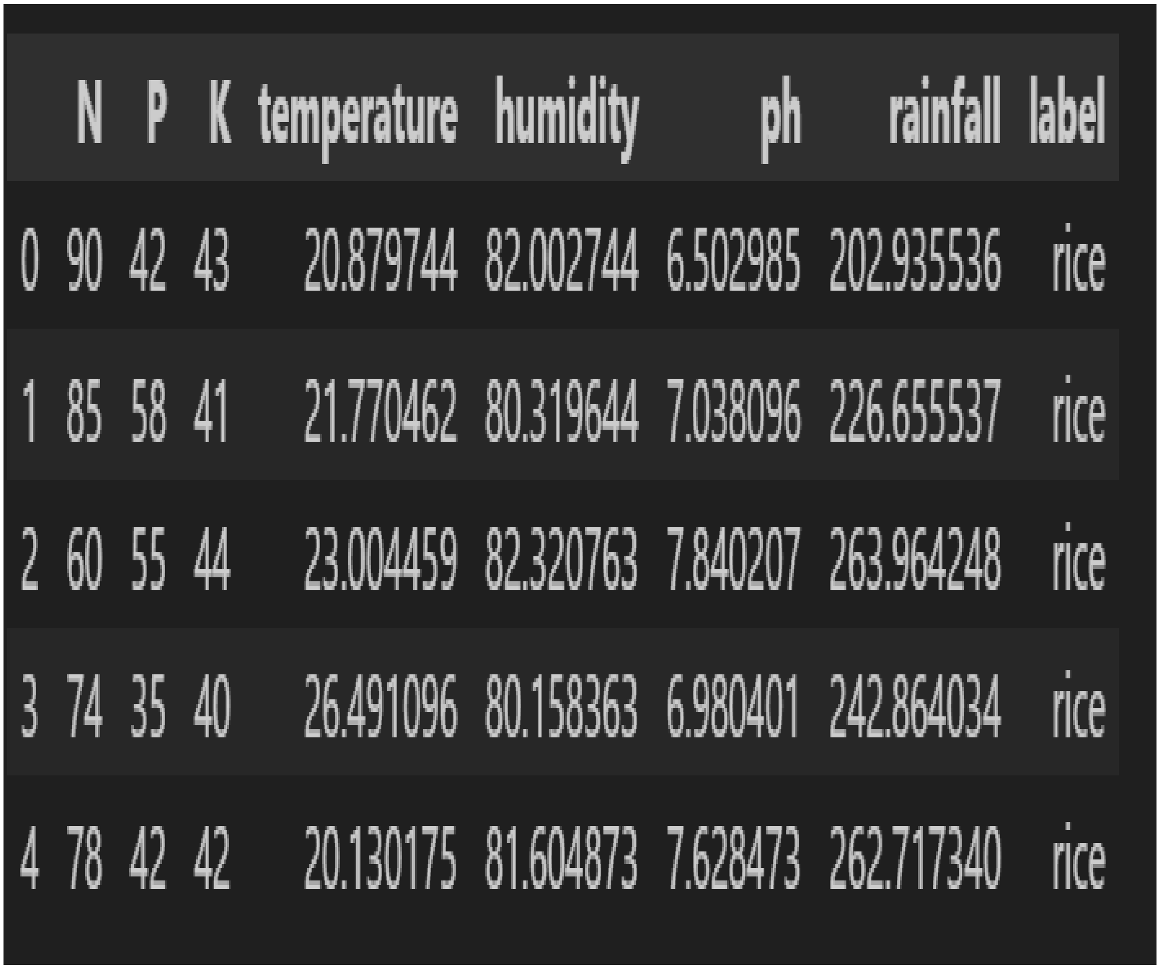
	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	20
1	85	58	41	21.770462	80.319644	7.038096	226.655537	20
2	60	55	44	23.004459	82.320763	7.840207	263.964248	20
3	74	35	40	26.491096	80.158363	6.980401	242.864034	20
4	78	42	42	20.130175	81.604873	7.628473	262.717340	20
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	5
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	5
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	5
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	5
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	5

2200 rows × 8 columns

HEAD AND TAIL

1.first few rows of the dataset

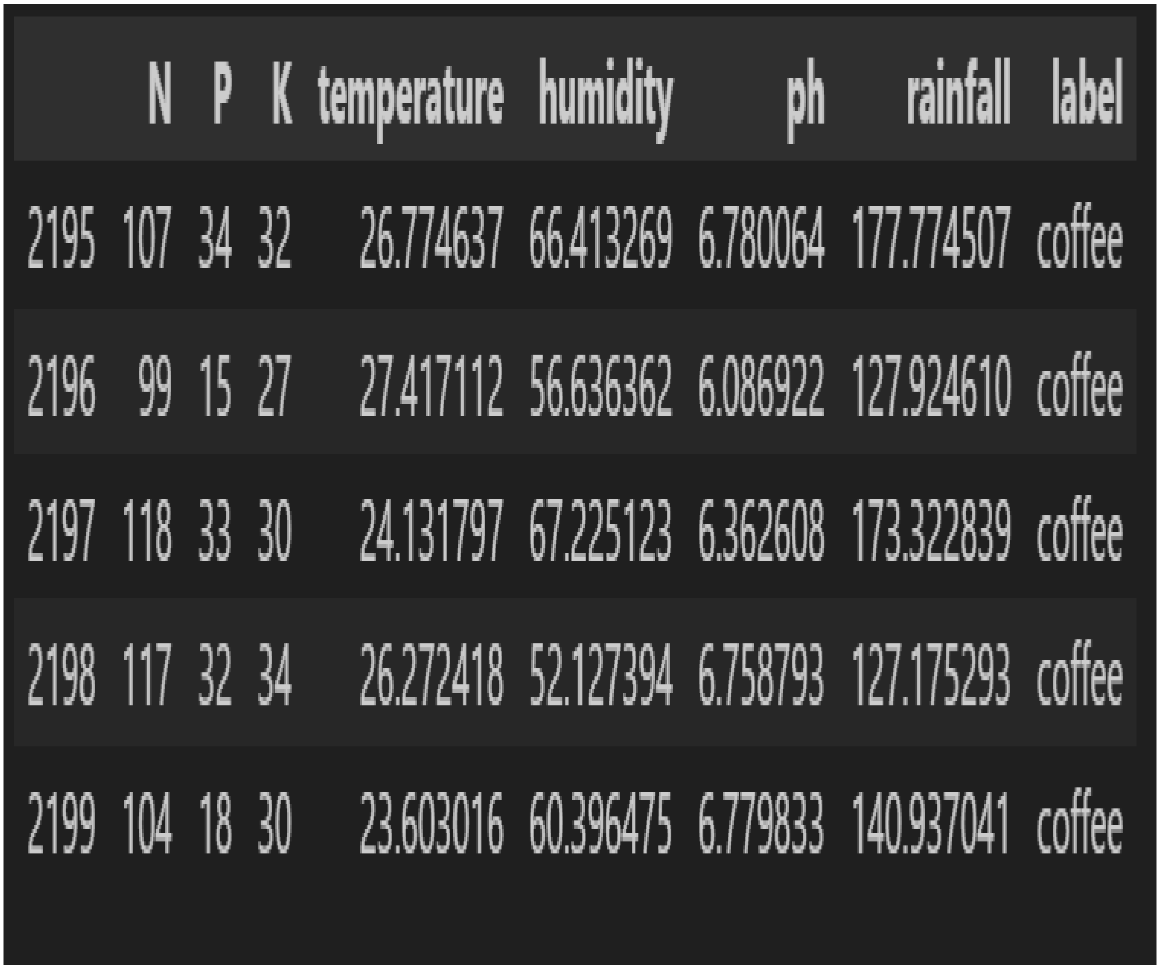
df.head()



	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

2.Last few rows of the dataset

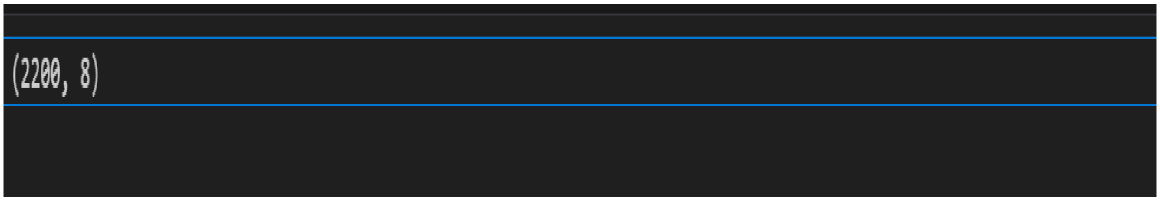
```
df.tail()
```



	N	P	K	temperature	humidity	ph	rainfall	label
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

3.df.shape will give the shape of the dataset

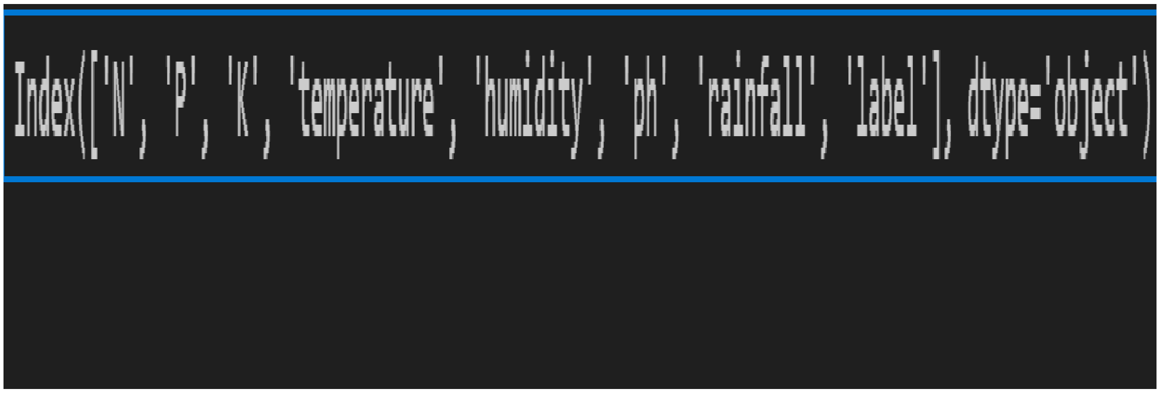
df.shape()



```
(2200, 8)
```

4.df.columns will give information about columns

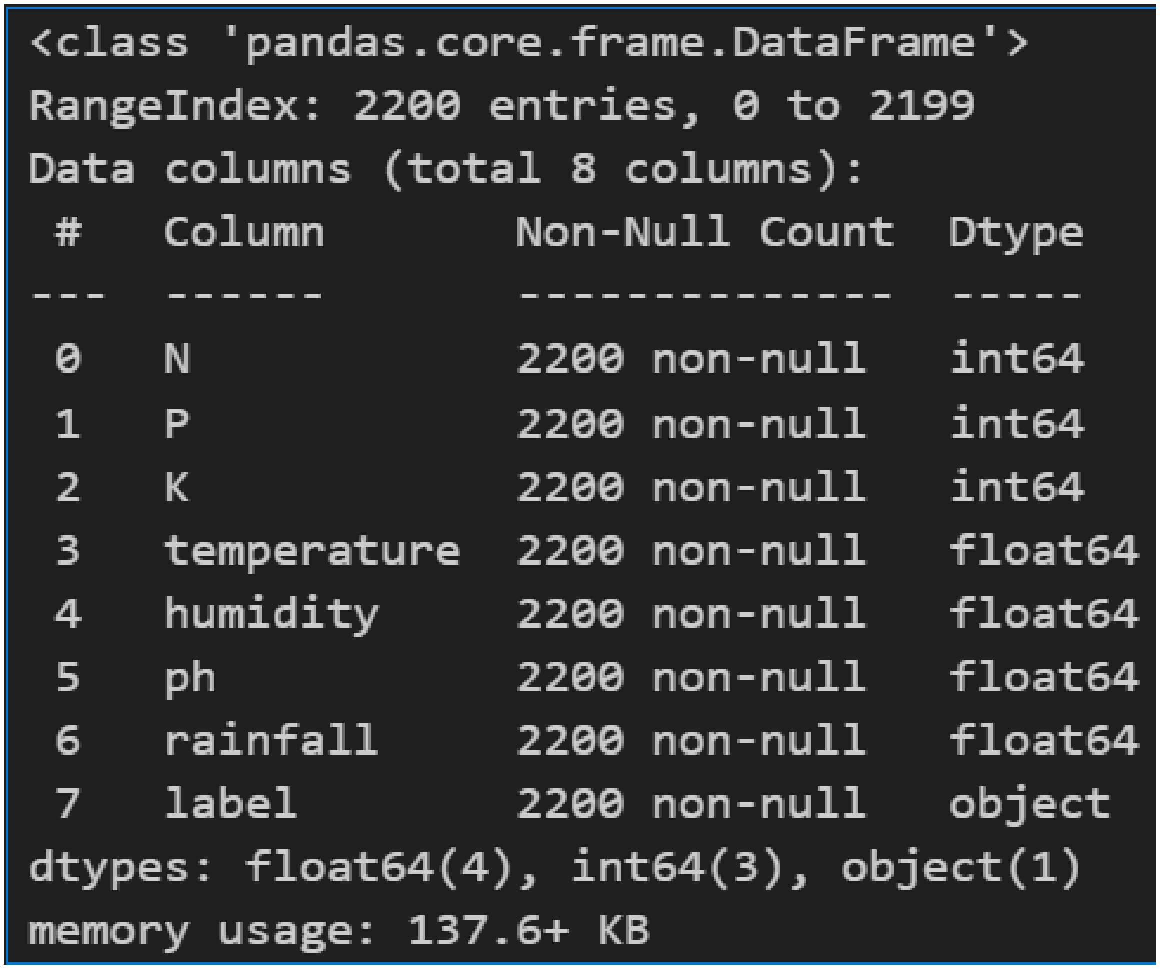
df.columns



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

5.df.info will give information of the dataset

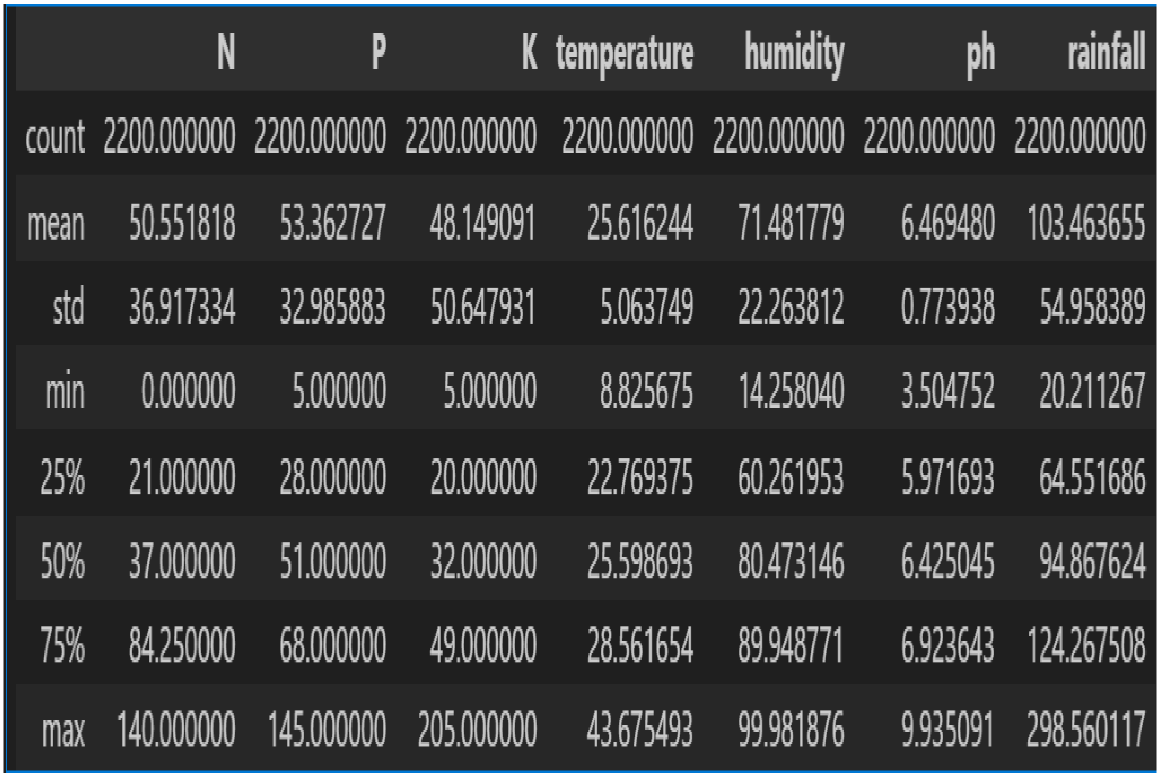
df.info()



```
<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
```

6.df.describe will describe about the numerical columns in dataset

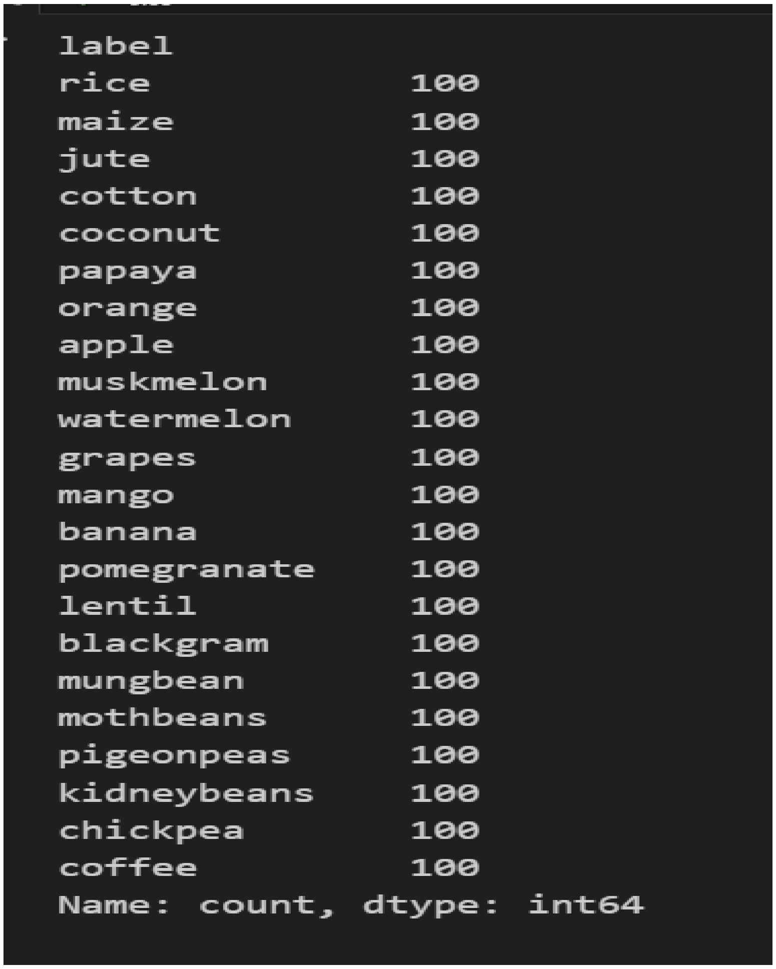
df.describe()



```
                 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
max     140.000000   145.000000   205.000000    43.675493    99.981876     9.935091   298.560117
```

7.df.columnname.value_counts() will give the count of column values with in the dataset

df.label.value_counts()

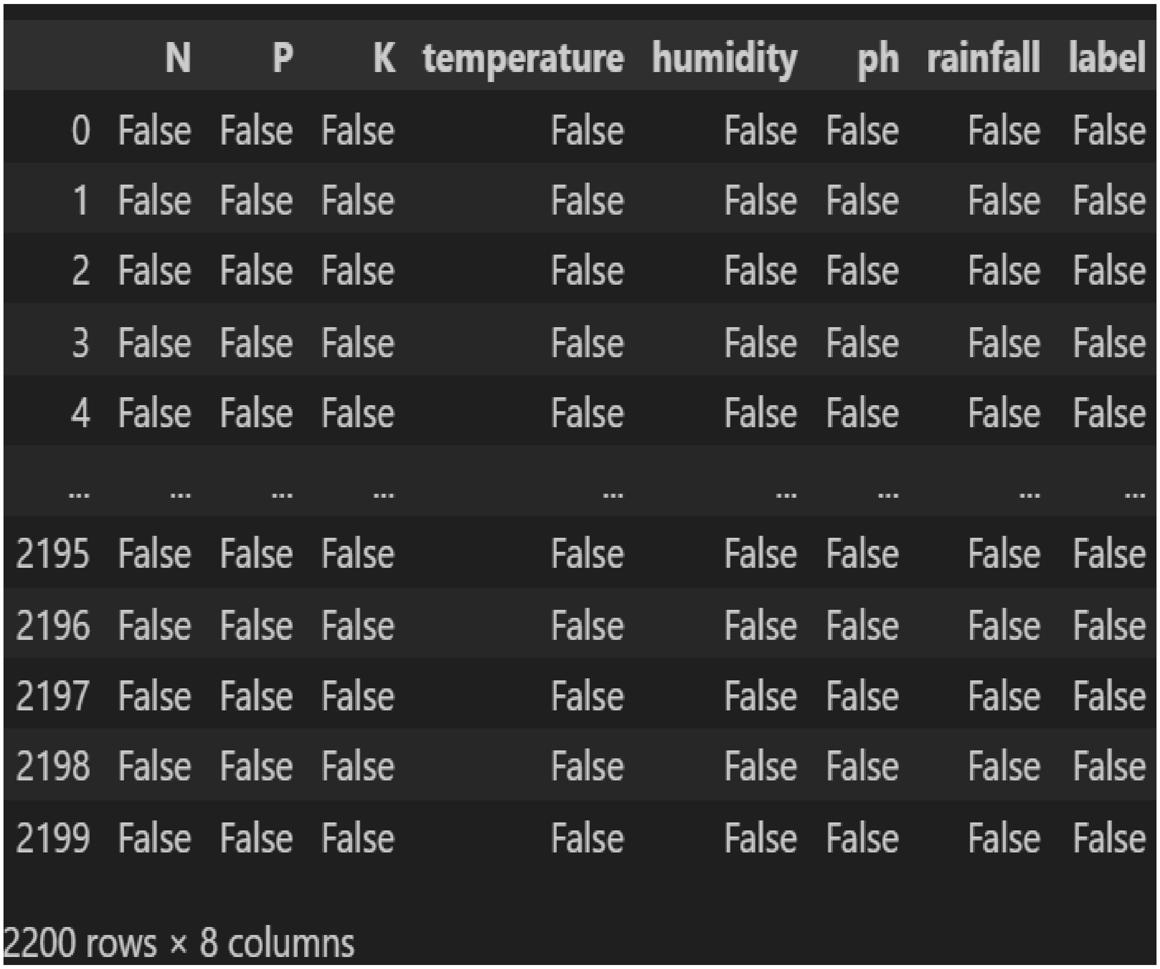


```
label
rice              100
maize             100
jute              100
cotton            100
coconut           100
papaya            100
orange            100
apple             100
muskmelon         100
watermelon        100
grapes            100
mango             100
banana            100
pomegranate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

Checking Null Values

1.df.isnull will give information about null values in boolean

df.isnull()

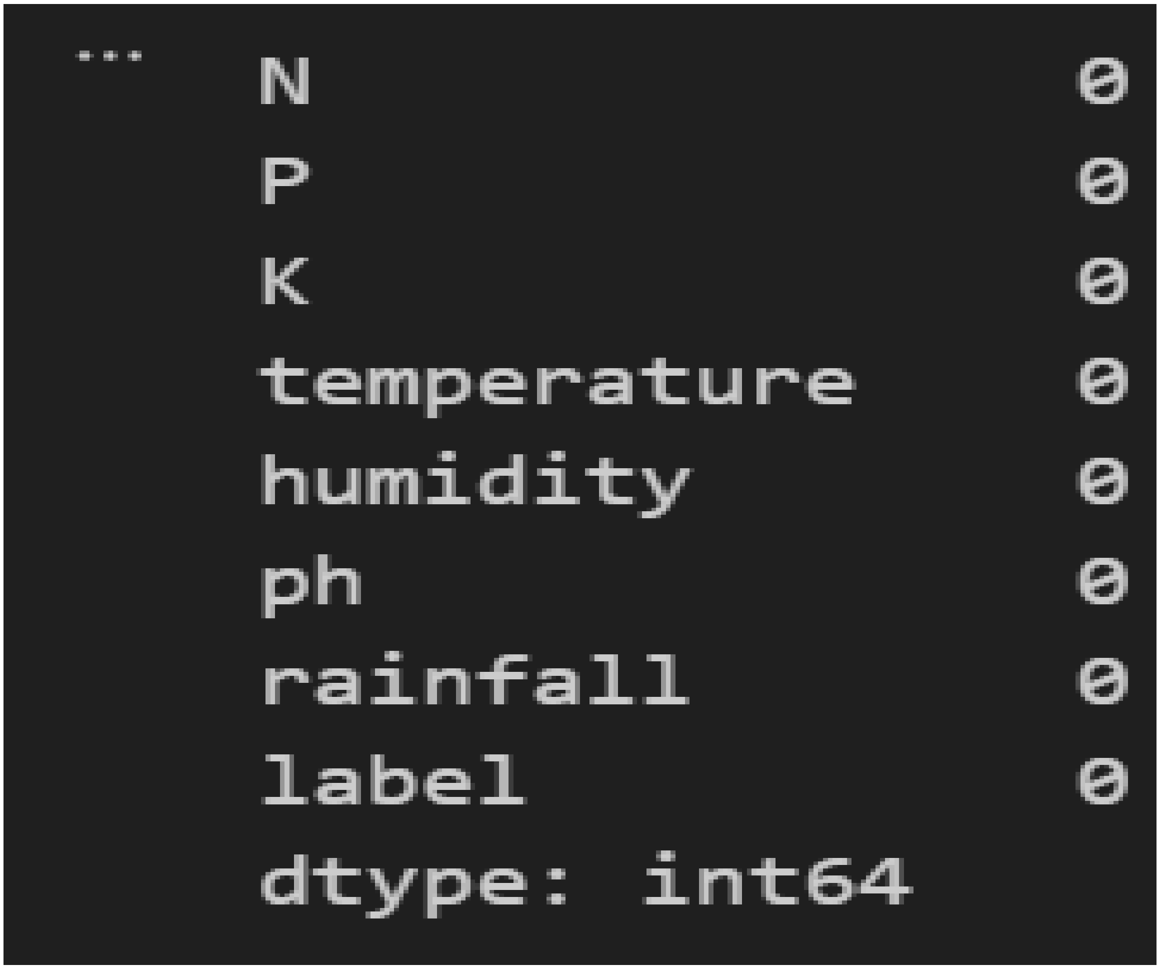


	N	P	K	temperature	humidity	ph	rainfall	label
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
2195	False	False	False	False	False	False	False	False
2196	False	False	False	False	False	False	False	False
2197	False	False	False	False	False	False	False	False
2198	False	False	False	False	False	False	False	False
2199	False	False	False	False	False	False	False	False

2200 rows × 8 columns

2.df.isnull().sum() will give information about null values

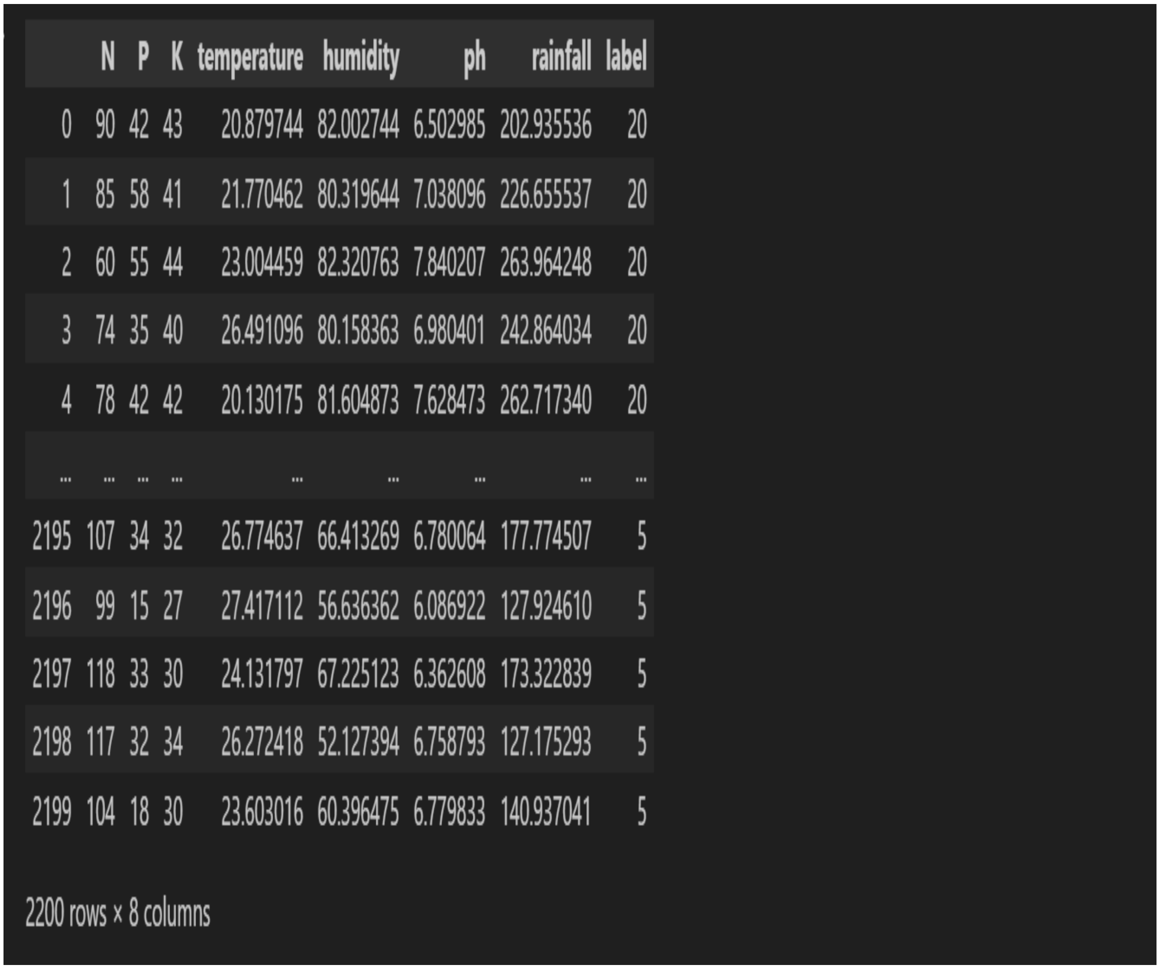
```
df.isnull().sum()
```



```
...   N              0
      P              0
      K              0
      temperature    0
      humidity       0
      ph             0
      rainfall       0
      label          0
      dtype: int64
```

Converting categorical column to numerical column

```
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

df[’label’]=le.fit_transform(df[”label”])

df
```



	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	20
1	85	58	41	21.770462	80.319644	7.038096	226.655537	20
2	60	55	44	23.004459	82.320763	7.840207	263.964248	20
3	74	35	40	26.491096	80.158363	6.980401	242.864034	20
4	78	42	42	20.130175	81.604873	7.628473	262.717340	20
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	5
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	5
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	5
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	5
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	5

2200 rows × 8 columns

Building the Model

```
importing required libraries

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Lasso

from sklearn.ensemblel importRandomForestRegressor

from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

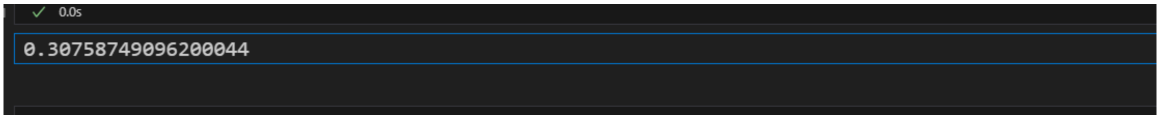
Splitting the Dataset

Initializing the dependent and independent variables and splitting the dataset into train set and test set.Let us assume label is our target variable.

```
X = df.drop([’label’], axis=1)

y = df[’label’]

Xtrain,Xtest,ytrain,ytest=train_test_split(X,y,test_size=0.3)
```

Linear Regression

```
model=LinearRegression()

model.fit(Xtrain,ytrain)

model.score(Xtrain,ytrain)
```

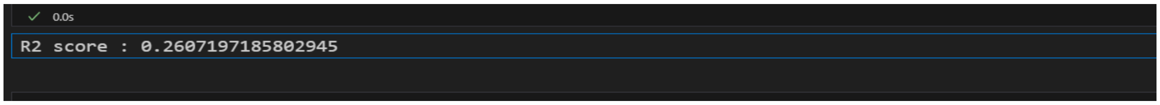


```
✓ 0.0s

0.30758749096200044
```

Predicting using Linear Regression

Predicting using Linear regression model and store it in y_pred variable

```
y_pred = model.predict(Xtest)

print(”R2 score :”,r2_score(ytest,y_pred))
```



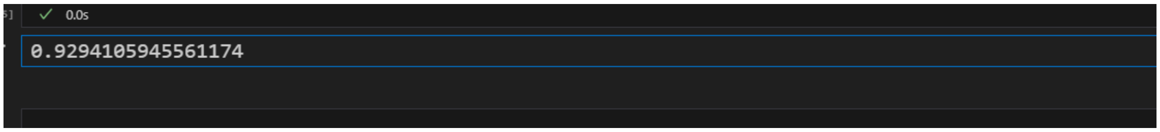
```
R2 score : 0.2607197185802945
```

Polynomial Regression

```
degree = 4

polyn_features = PolynomialFeatures(degree=degree)

Xtrain_polyn = polyn_features.fit_transform(Xtrain)

polyn_regression_model = LinearRegression()

polyn_regression_model.fit(Xtrain_polyn, ytrain)
```

Predicting using Polynomial Regression

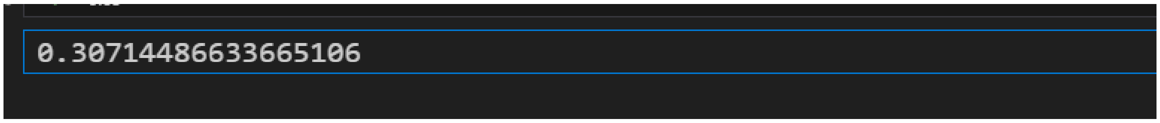
```
polyn_regression_model.score(Xtrain_polyn,ytrain)
```



```
0.9294105945561174
```

Lasso Regression

```
alpha=0.1

model2=Lasso(alpha=alpha)

model2.fit(Xtrain, ytrain)

model2.score(Xtrain,ytrain)
```

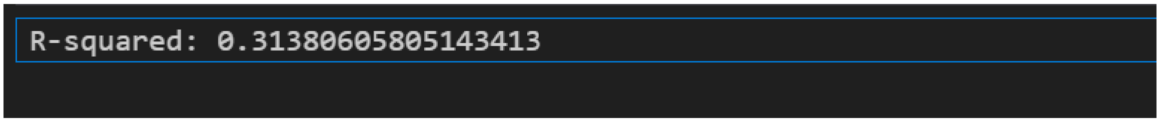


```
0.30714486633665106
```

Predicting using Lasso Regression

Predicting using Lasso Regression model and store it in y_predicted variable

```
y_predicted=model2.predict(Xtest)

print(”R-squared :”,r2_score(ytest,y_predicted))
```

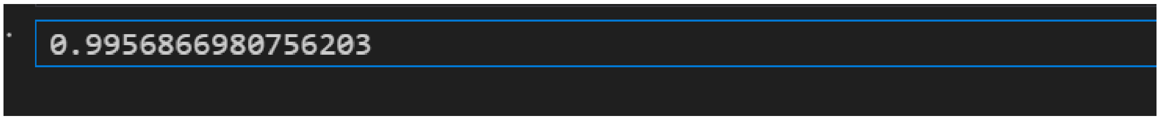


```
R-squared: 0.31380605805143413
```

RandomForestRegressor

```
rf=RandomForestRegressor()
```

```
rf.fit(Xtrain,ytrain)

rf.score(Xtrain,ytrain)
```

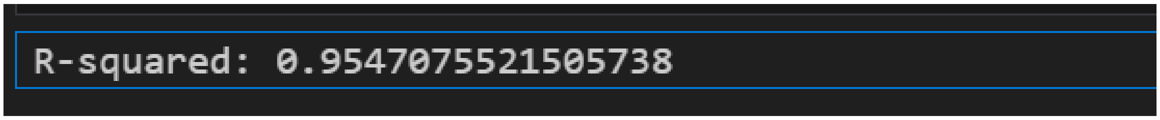


```
0.9956866980756203
```

Predicting using Random Forest Regressor

Predicting using Random Forest Regreesor model and store it in y_predict variable

```
y_predict=rf.predict(Xtest)

print(”R-squared :”,r2_score(ytest,y_predict))
```



```
R-squared: 0.9547075521505738
```

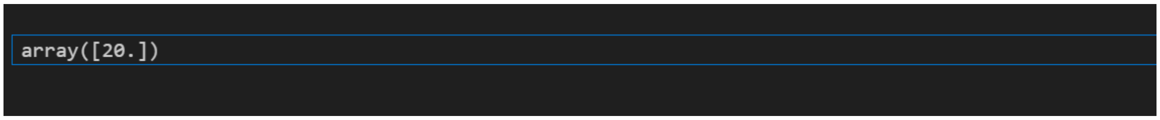
Here we observe that r2 score of our model is 0.95

Model Evaluation

Checking whether our model is recommending well or not

Testing the Data with the model

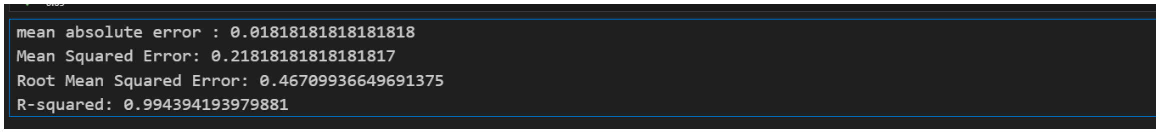
```
rf.predict([[’60’ ,’55’ ,’44’, ’23.00445915’, ’82.3207629’, ’7.840207144’, ’263.9642476’]])
```



```
array([20.])
```

Metrics

```
y_predi=rf.predict(Xtest)

mae=mean_absolute_error(ytest,y_predi)

print(”mean absolute error :”, mae)

mse = mean_squared_error(ytest, y_predi)

rmse = mean_squared_error(ytest, y_predi, squared=False)

print(”Mean Squared Error:”, mse)

print(”Root Mean Squared Error:”, rmse)

r_squared = r2_score(ytest, y_predi)

print(”R-squared:”, r_squared)
```



```
mean absolute error : 0.018181818181818
Mean Squared Error: 0.21818181818181817
Root Mean Squared Error: 0.46709936649691375
R-squared: 0.994394193979881
```

Chapter 4

Conclusion and Future Work

4.1 conclusion

In conclusion, our project aimed to recommend the farmers to grow suitable crops using some learning models like Linear Regression , Polynomial Regression, Lasso Regression and Random Forest Regressor. By using Random Forest Regressor model we can recommend the suitable crop of a particular area over some parameters. By this model we can achieve 99% accuracy.