INTRODUCTION TO MACHINE LEARNING

(CSE-3008)

CROP RECOMMENDATION (WEB APPLICATION)

Varad Vijay Temghare
19BCI7045

Vellore Institute of Technology, Amaravati

2021-22

GUIDED BY :
DR. ARUNDHATI DAS

19BCI7045

VARAD VIJAY TEMGHARE

Colab link : https://colab.research.google.com/drive/1Sti1YhGgSuOUs8PFFwwJ7DRFNWffM7Qd?usp=sharing

```
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
```

```
Mounted at /content/gdrive
```

```
import pandas as pd
import io
import numpy as np
import random
from sklearn.model_selection import cross_val_score

import matplotlib.pyplot as plt
import seaborn as sns

import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots
import seaborn as sns
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn import tree
import warnings
from sklearn.metrics import confusion_matrix
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('/content/gdrive/MyDrive/ML Lab/Crop.csv')
df.head()
```

	n	p	k	temperature	humidity	ph	rainfall	crop	season	area	pr
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice	winter	99	
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice	whole year	277	
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice	autumn	149	
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice	kharif	163	
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice	autumn	295	

```
df.shape
```

```
(2200, 11)
```

```
df.columns
```

```
Index(['n', 'p', 'k', 'temperature', 'humidity', 'ph', 'rainfall', 'crop',
       'season', 'area', 'production'],
      dtype='object')
```

```
df.isnull().any()
```

```
n              False
p              False
k              False
temperature    False
humidity       False
ph             False
rainfall       False
crop           False
season         False
area           False
production     False
dtype: bool
```

```
print("Number of various crops: ", len(df['crop'].unique()))
print("List of crops: ",df['crop'].unique())
```

```
Number of various crops:  22
List of crops:  ['rice' 'maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans'
 'mungbean' 'blackgram' 'lentil' 'pomegranate' 'banana' 'mango' 'grapes'
 'watermelon' 'muskmelon' 'apple' 'orange' 'papaya' 'coconut' 'cotton'
 'jute' 'coffee']
```

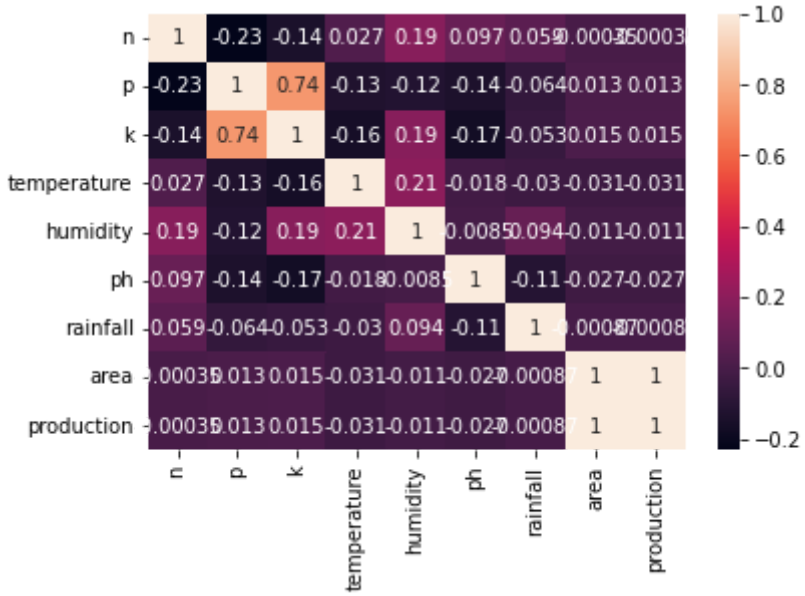
```
df['crop'].value_counts()
```

```
jute           100
coconut        100
coffee         100
cotton         100
muskmelon      100
papaya         100
orange         100
pigeonpeas     100
lentil         100
grapes         100
apple          100
pomegranate    100
chickpea       100
mothbeans      100
blackgram      100
mungbean       100
banana         100
kidneybeans    100
rice           100
watermelon     100
maize          100
```

```
mango           100
Name: crop, dtype: int64
```

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

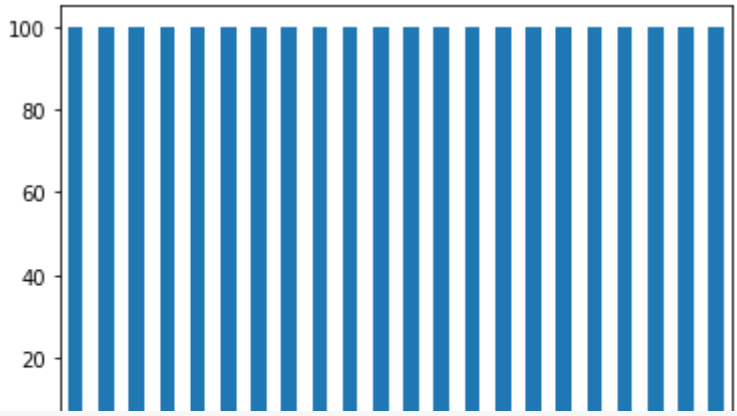
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3c1c25b2d0>
```




```
crop_summary = pd.pivot_table(df,index=['crop'],aggfunc='mean')
crop_summary.head()
```

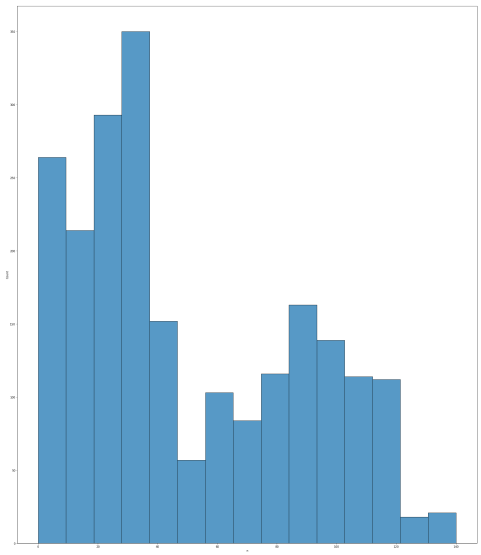
crop	area	humidity	k	n	p	ph	production	rainfall
apple	180.40	92.333383	199.89	20.80	134.22	5.929663	3968.80	112.654779
banana	163.85	80.358123	50.05	100.23	82.01	5.983893	3604.70	104.626980
blackgram	170.40	65.118426	19.24	40.02	67.47	7.133952	3748.80	67.884151
chickpea	177.40	16.860439	79.92	40.09	67.79	7.336957	3902.80	80.058977
coconut	173.04	94.844272	30.59	21.98	16.93	5.976562	3806.88	175.686646

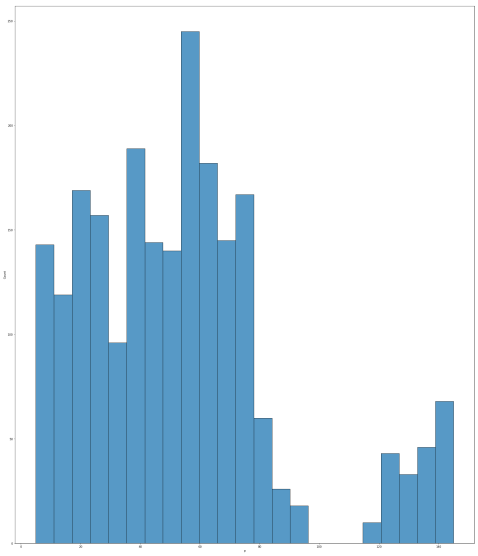
```
all_columns = df.columns[:-1]

labels = df["crop"].unique()
df["crop"].value_counts().plot(kind="bar")

plt.show()
```

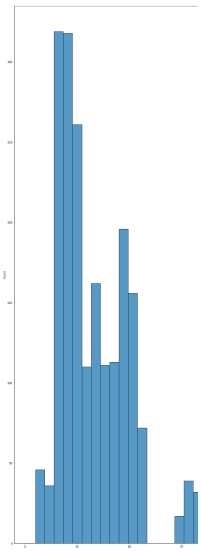


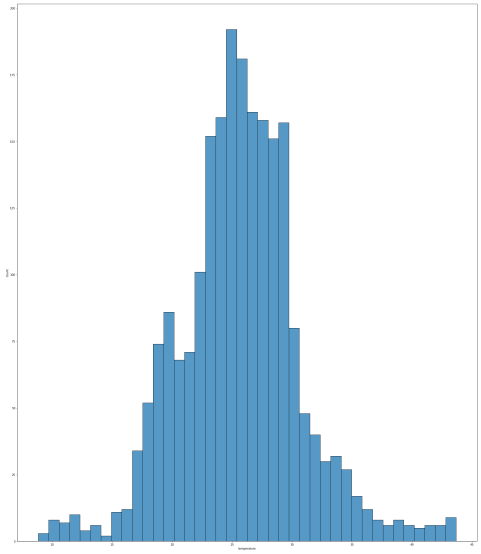

```
colorarr = ['#0592D0','#Cd7f32', '#E97451', '#Bdb76b', '#954535', '#C2b280', '#808000','#C
            '#32cd32','#39ff14','#00ff7f', '#008080', '#36454f', '#F88379', '#Ff4500', '#Ff
            '#Faf0e6', '#8c92ac', '#Dbd7d2','#A7a6ba', '#B38b6d']
```

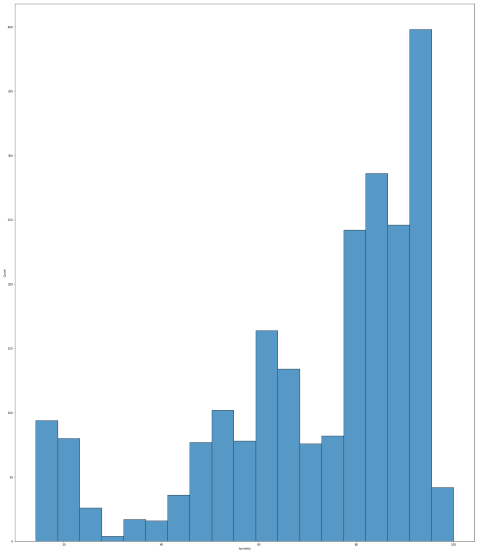
```
all_columns = df.columns[:-1]

plt.figure(figsize=(100,120))
i = 1
for column in all_columns[:-1]:
    plt.subplot(3,3,i)
    sns.histplot(df[column])
    i+=1
plt.show()

sns.histplot(df[all_columns[-1]])
plt.show()
```

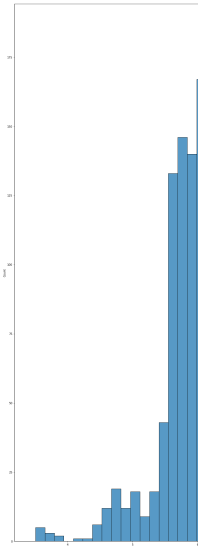


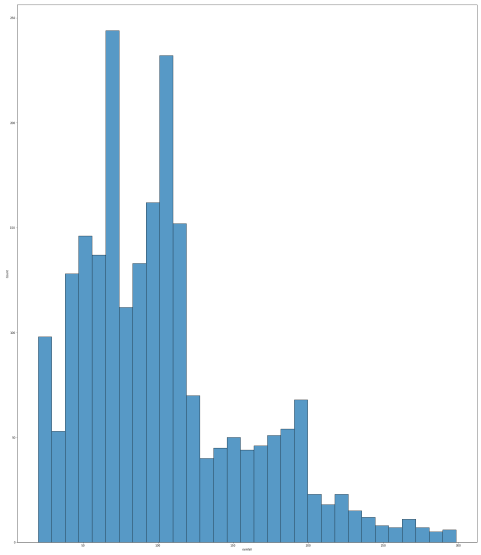


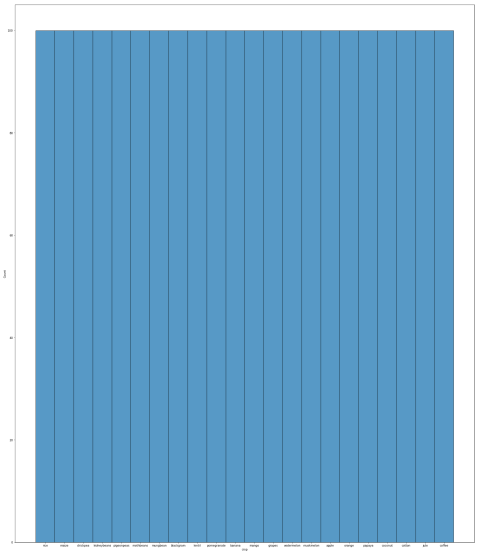


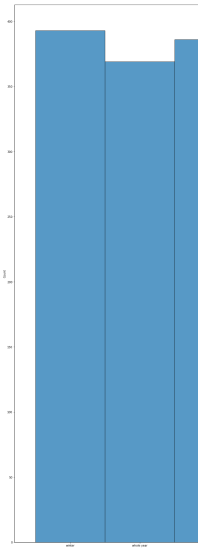


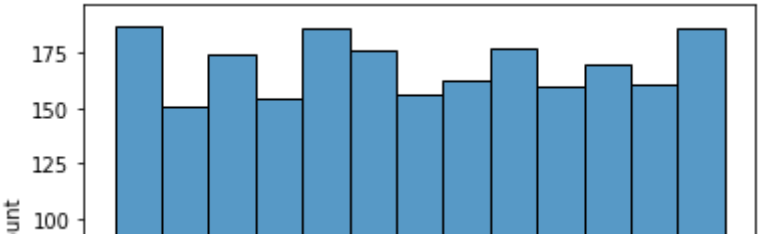


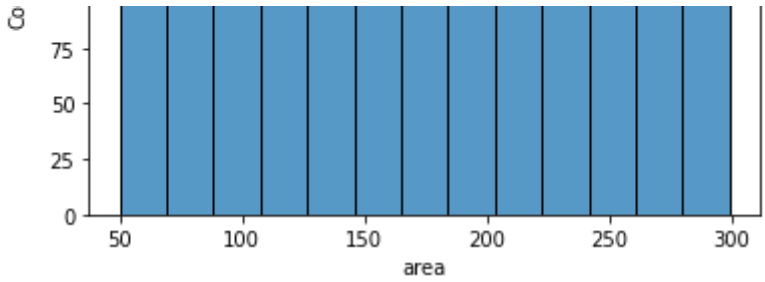






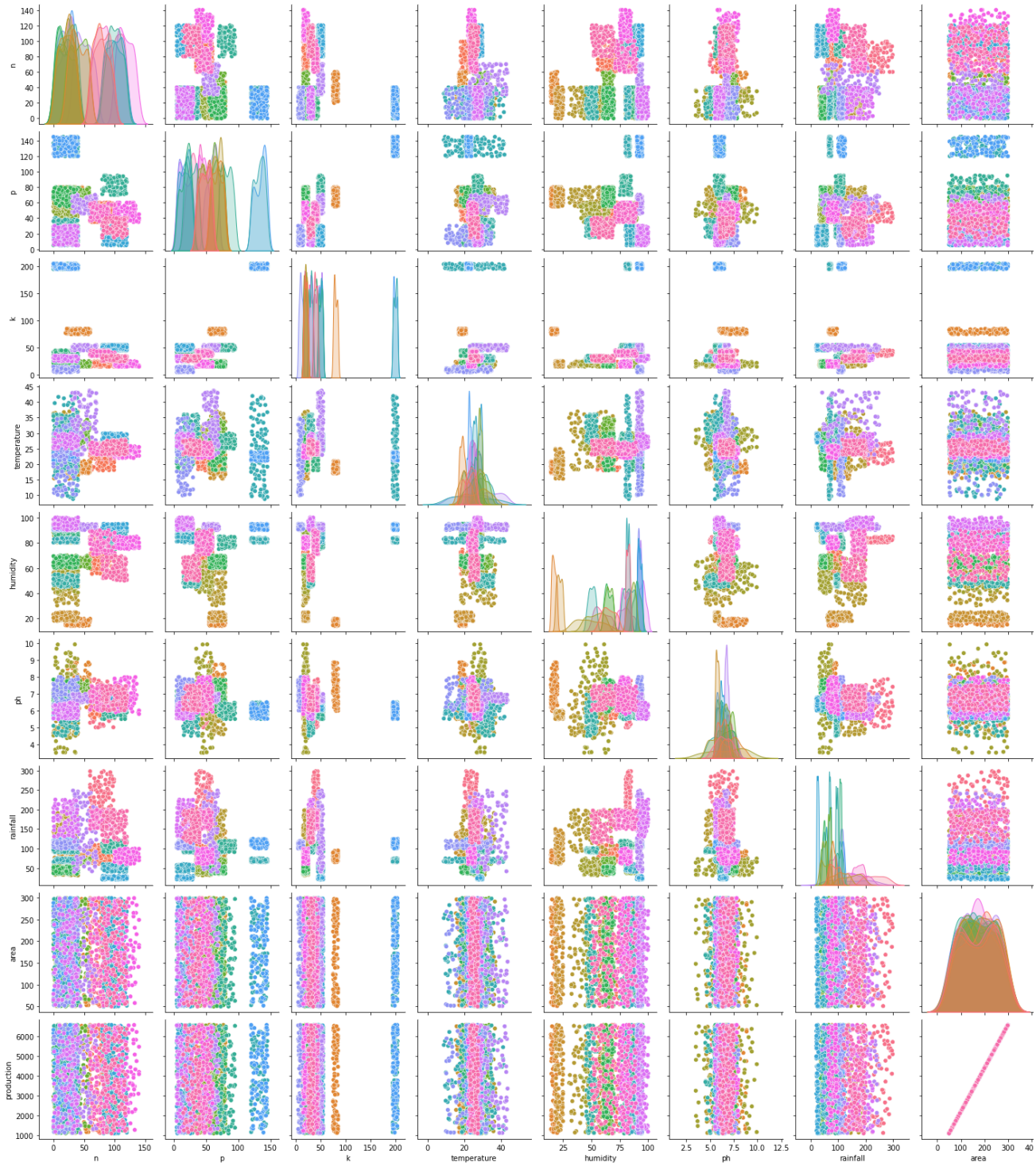



175
150
125
100
unt




```
plt.figure(figsize=(19,17))
sns.pairplot(df, hue = "crop")
plt.show()
```

```
<Figure size 1368x1224 with 0 Axes>
```

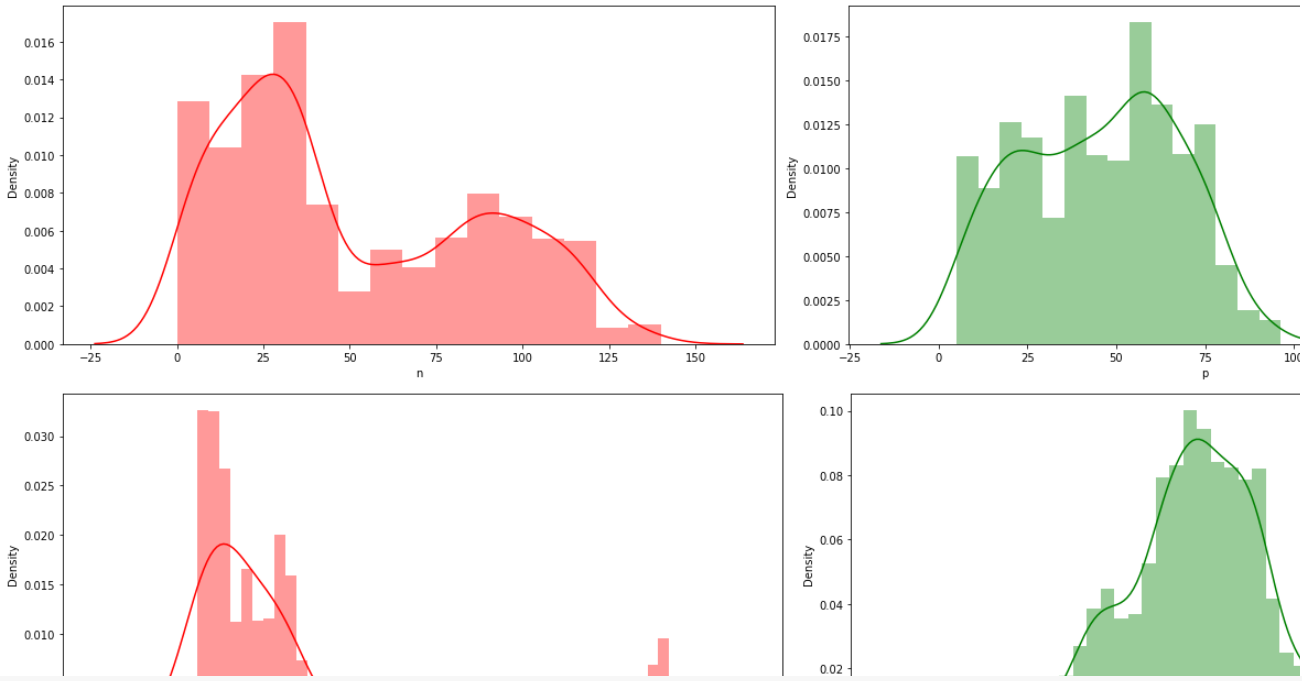



```
from sklearn import preprocessing
le=preprocessing.LabelEncoder()
le.fit(['df.season'])
df['season']=le.fit_transform(df['season'])
```

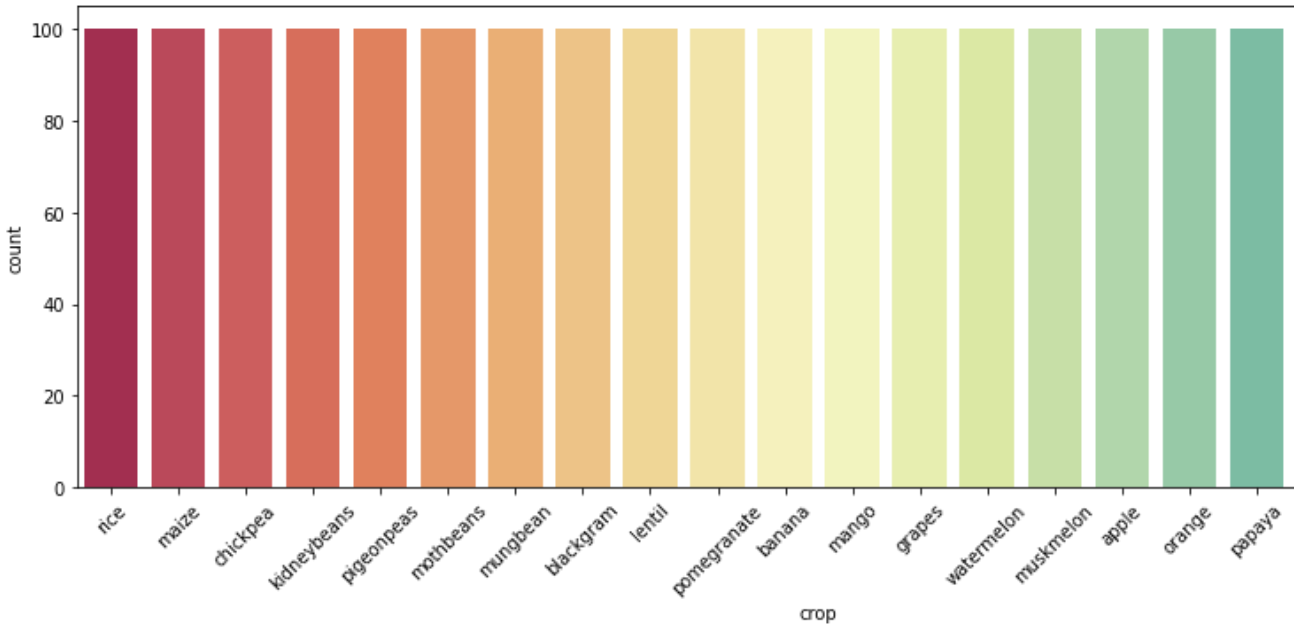
```
f= plt.figure(figsize=(20,5))
ax=f.add_subplot(121)
sns.distplot(df['n'] , color ='red',ax=ax)

ax=f.add_subplot(122)
sns.distplot(df['p'] , color ='green' , ax = ax)
```

```
plt.tight_layout()
f= plt.figure(figsize=(20,5))
ax=f.add_subplot(121)
sns.distplot(df['k'] , color ='red',ax=ax)

ax=f.add_subplot(122)
sns.distplot(df['temperature'] , color ='green' , ax = ax)
plt.tight_layout()
f= plt.figure(figsize=(20,5))
ax=f.add_subplot(121)
sns.distplot(df['humidity'] , color ='red',ax=ax)

ax=f.add_subplot(122)
sns.distplot(df['ph'] , color ='green' , ax = ax)
plt.tight_layout()
sns.distplot(df['rainfall'],color ='red')
sns.distplot(df['season'],color ='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3c1bc254d0>
```




```
f= plt.figure(figsize=(15,5))
sns.countplot(df['crop'] , palette = 'Spectral')
plt.xticks(rotation=45)
plt.show()
```

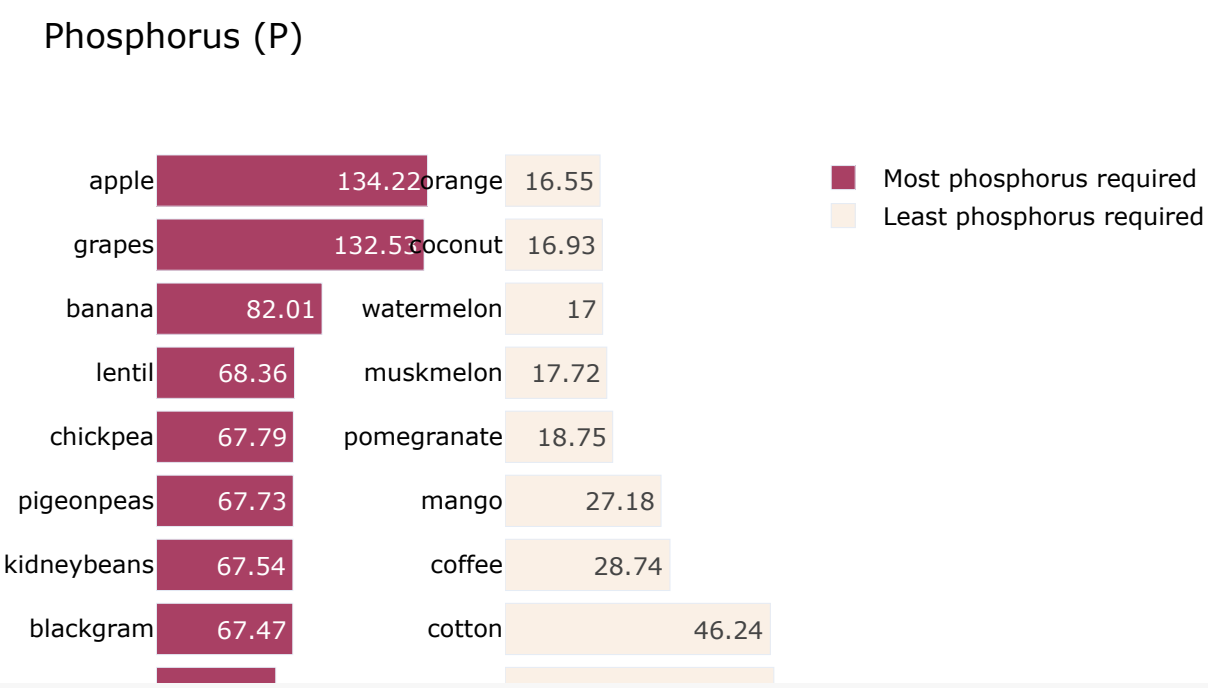



```
crop_summary_N = crop_summary.sort_values(by='n', ascending=False)

fig = make_subplots(rows=1, cols=2)

top = {
    'y' : crop_summary_N['n'][0:10].sort_values().index,
    'x' : crop_summary_N['n'][0:10].sort_values()
}
```

```
last = {
    'y' : crop_summary_N['n'][-10:].index,
    'x' : crop_summary_N['n'][-10:]
}

fig.add_trace(
    go.Bar(top,
           name="Most nitrogen required",
           marker_color=random.choice(colorarr),
           orientation='h',
           text=top['x']),

    row=1, col=1
)

fig.add_trace(
    go.Bar(last,
           name="Least nitrogen required",
           marker_color=random.choice(colorarr),
           orientation='h',
          text=last['x']),
    row=1, col=2
)
fig.update_traces(texttemplate='%{text}', textposition='inside')
fig.update_layout(title_text="Nitrogen (N)",
                  plot_bgcolor='white',
                  font_size=12,
                  font_color='black',
                 height=500)
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```

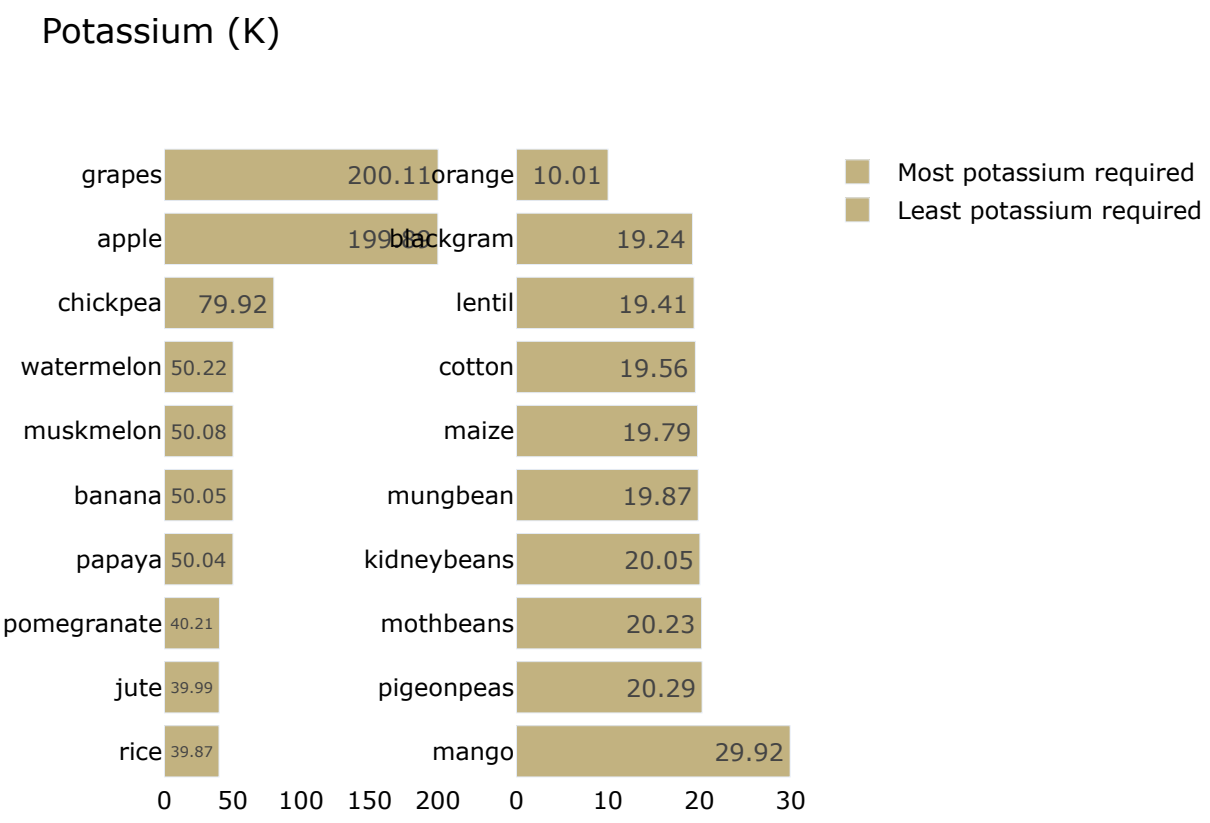
Nitrogen (N)

```
crop_summary_P = crop_summary.sort_values(by='p', ascending=False)

fig = make_subplots(rows=1, cols=2)

top = {
    'y' : crop_summary_P['p'][0:10].sort_values().index,
    'x' : crop_summary_P['p'][0:10].sort_values()
}

last = {
    'y' : crop_summary_P['p'][-10:].index,
    'x' : crop_summary_P['p'][-10:]
}

fig.add_trace(
    go.Bar(top,
           name="Most phosphorus required",
           marker_color=random.choice(colorarr),
           orientation='h',
           text=top['x']),
           row=1, col=1
)

fig.add_trace(
    go.Bar(last,
           name="Least phosphorus required",
           marker_color=random.choice(colorarr),
           orientation='h',
           text=last['x']),
           row=1, col=2
)
fig.update_traces(texttemplate='%{text}', textposition='inside')
fig.update_layout(title_text="Phosphorus (P)",
                  plot_bgcolor='white',
                  font_size=12,
                  font_color='black',
                 height=500)

fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```



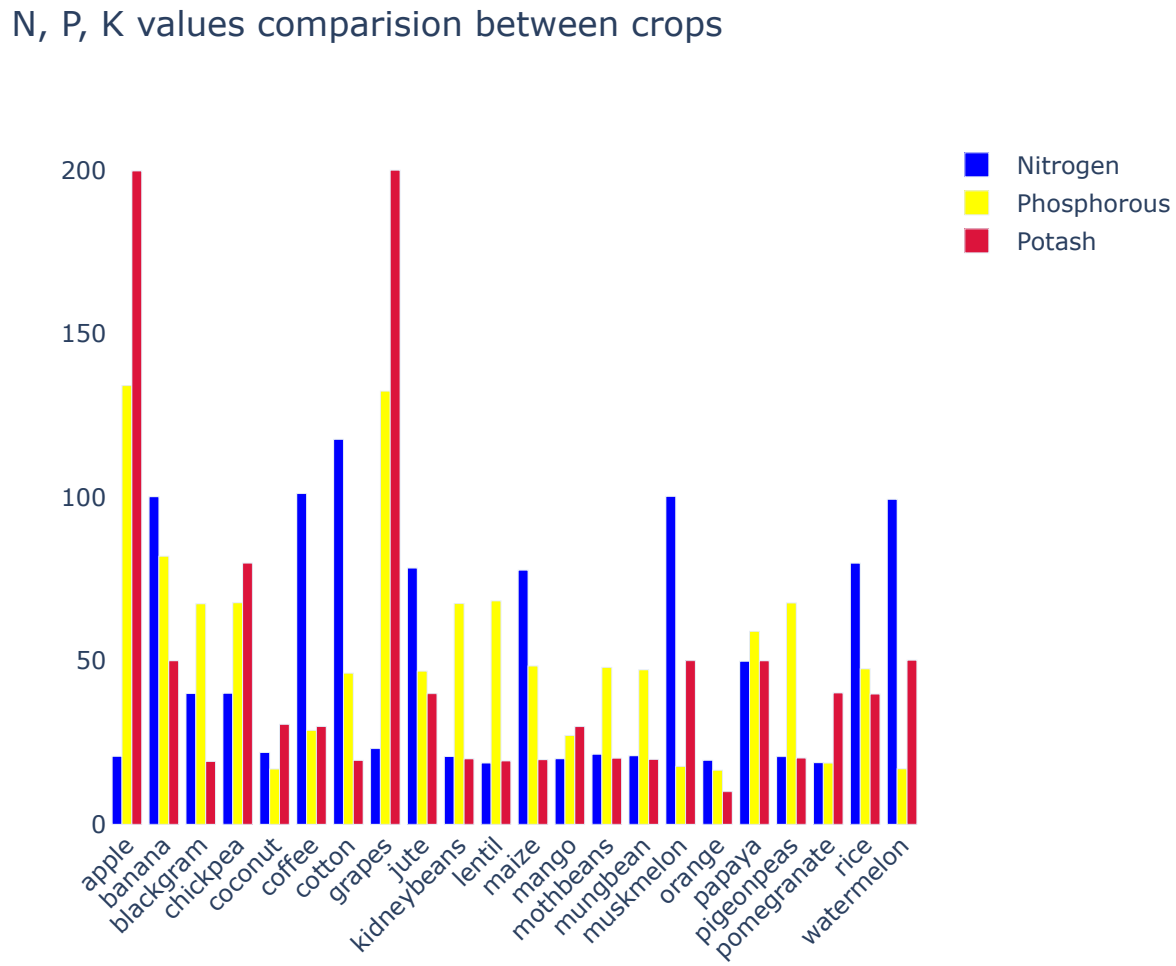

```
crop_summary_K = crop_summary.sort_values(by='k', ascending=False)

fig = make_subplots(rows=1, cols=2)

top = {
    'y' : crop_summary_K['k'][0:10].sort_values().index,
    'x' : crop_summary_K['k'][0:10].sort_values()
}

last = {
    'y' : crop_summary_K['k'][-10:].index,
    'x' : crop_summary_K['k'][-10:]
}

fig.add_trace(
    go.Bar(top,
           name="Most potassium required",
           marker_color=random.choice(colorarr),
           orientation='h',
          text=top['x']),
          row=1, col=1
)

fig.add_trace(
    go.Bar(last,
           name="Least potassium required",
           marker_color=random.choice(colorarr),
           orientation='h',
          text=last['x']),
    row=1, col=2
)
fig.update_traces(texttemplate='%{text}', textposition='inside')
fig.update_layout(title_text="Potassium (K)",
                  plot_bgcolor='white',
```

```
                  font_size=12,
                  font_color='black',
                 height=500)

fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```



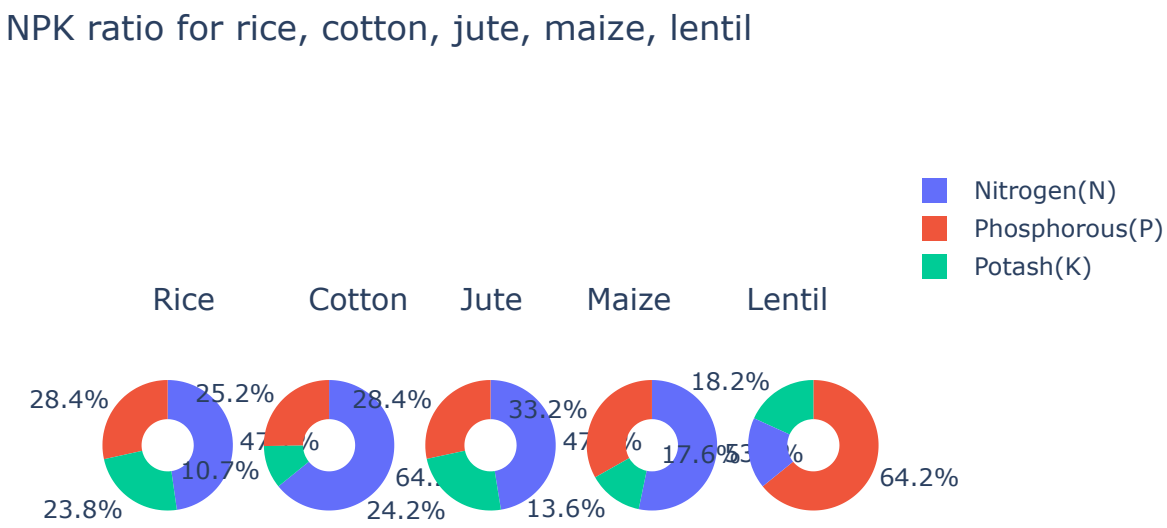

```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=crop_summary.index,
    y=crop_summary['n'],
    name='Nitrogen',
    marker_color='blue'
))
fig.add_trace(go.Bar(
    x=crop_summary.index,
    y=crop_summary['p'],
    name='Phosphorous',
    marker_color='yellow'
))
fig.add_trace(go.Bar(
    x=crop_summary.index,
    y=crop_summary['k'],
    name='Potash',
    marker_color='crimson'
))
```

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

fig.show()
```



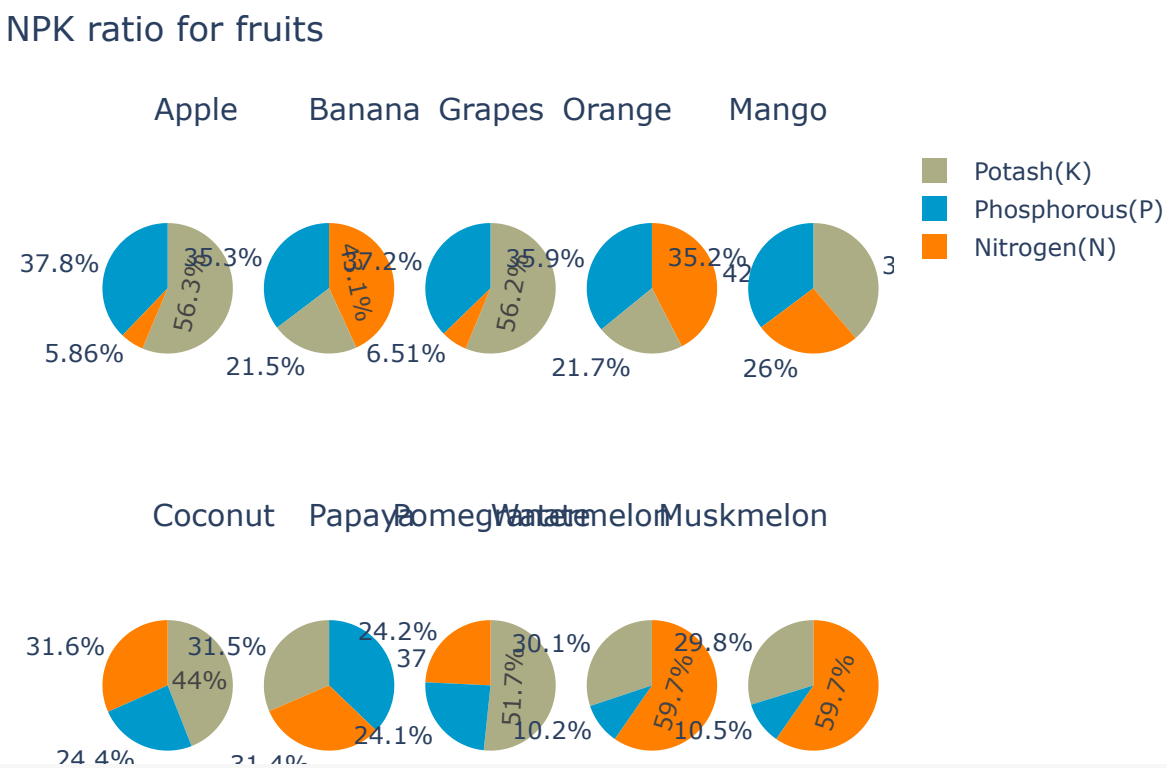

```
labels = ['Nitrogen(N)','Phosphorous(P)','Potash(K)']
fig = make_subplots(rows=1, cols=5, specs=[[{'type':'domain'}, {'type':'domain'},
                                            {'type':'domain'}, {'type':'domain'},
                                            {'type':'domain'}]])

rice_npk = crop_summary[crop_summary.index=='rice']
values = [rice_npk['n'][0], rice_npk['p'][0], rice_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Rice"),1, 1)

cotton_npk = crop_summary[crop_summary.index=='cotton']
values = [cotton_npk['n'][0], cotton_npk['p'][0], cotton_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Cotton"),1, 2)
jute_npk = crop_summary[crop_summary.index=='jute']
values = [jute_npk['n'][0], jute_npk['p'][0], jute_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Jute"),1, 3)

maize_npk = crop_summary[crop_summary.index=='maize']
values = [maize_npk['n'][0], maize_npk['p'][0], maize_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Maize"),1, 4)
```

```
lentil_npk = crop_summary[crop_summary.index=='lentil']
values = [lentil_npk['n'][0], lentil_npk['p'][0], lentil_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Lentil"),1, 5)
fig.update_traces(hole=.4, hoverinfo="label+percent+name")
fig.update_layout(
    title_text="NPK ratio for rice, cotton, jute, maize, lentil",
    annotations=[dict(text='Rice',x=0.06,y=0.8, font_size=15, showarrow=False),
                 dict(text='Cotton',x=0.26,y=0.8, font_size=15, showarrow=False),
                 dict(text='Jute',x=0.50,y=0.8, font_size=15, showarrow=False),
                 dict(text='Maize',x=0.74,y=0.8, font_size=15, showarrow=False),
                dict(text='Lentil',x=0.94,y=0.8, font_size=15, showarrow=False)])
fig.show()
```

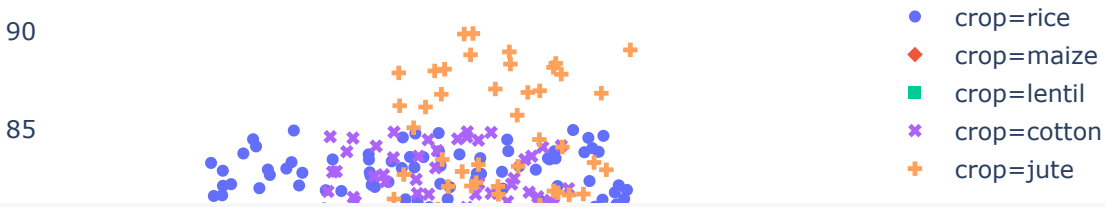



```
labels = ['Nitrogen(N)','Phosphorous(P)','Potash(K)']
specs = [[{'type':'domain'}, {'type':'domain'}, {'type':'domain'}, {'type':'domain'}, {'ty
         {'type':'domain'}, {'type':'domain'}, {'type':'domain'}, {'type':'domain'}, {'typ
fig = make_subplots(rows=2, cols=5, specs=specs)
cafe_colors =  ['rgb(255, 128, 0)', 'rgb(0, 153, 204)', 'rgb(173, 173, 133)']

apple_npk = crop_summary[crop_summary.index=='apple']
values = [apple_npk['n'][0], apple_npk['p'][0], apple_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Apple", marker_colors=cafe_colors)

banana_npk = crop_summary[crop_summary.index=='banana']
values = [banana_npk['n'][0], banana_npk['p'][0], banana_npk['k'][0]]
```

```
fig.add_trace(go.Pie(labels=labels, values=values,name="Banana", marker_colors=cafe_colors

grapes_npk = crop_summary[crop_summary.index=='grapes']
values = [grapes_npk['n'][0], grapes_npk['p'][0], grapes_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Grapes", marker_colors=cafe_colors

orange_npk = crop_summary[crop_summary.index=='orange']
values = [orange_npk['n'][0], orange_npk['p'][0], orange_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Orange", marker_colors=cafe_colors

mango_npk = crop_summary[crop_summary.index=='mango']
values = [mango_npk['n'][0], mango_npk['p'][0], mango_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Mango", marker_colors=cafe_colors)

coconut_npk = crop_summary[crop_summary.index=='coconut']
values = [coconut_npk['n'][0], coconut_npk['p'][0], coconut_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Coconut", marker_colors=cafe_color

papaya_npk = crop_summary[crop_summary.index=='papaya']
values = [papaya_npk['n'][0], papaya_npk['p'][0], papaya_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Papaya", marker_colors=cafe_colors

pomegranate_npk = crop_summary[crop_summary.index=='pomegranate']
values = [pomegranate_npk['n'][0], pomegranate_npk['p'][0], pomegranate_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Pomegranate", marker_colors=cafe_c

watermelon_npk = crop_summary[crop_summary.index=='watermelon']
values = [watermelon_npk['n'][0], watermelon_npk['p'][0], watermelon_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Watermelon", marker_colors=cafe_co

muskmelon_npk = crop_summary[crop_summary.index=='muskmelon']
values = [muskmelon_npk['n'][0], muskmelon_npk['p'][0], muskmelon_npk['k'][0]]
fig.add_trace(go.Pie(labels=labels, values=values,name="Muskmelon", marker_colors=cafe_col
fig.update_layout(
    title_text="NPK ratio for fruits",
    annotations=[dict(text='Apple',x=0.06,y=1.08, font_size=15, showarrow=False),
                 dict(text='Banana',x=0.26,y=1.08, font_size=15, showarrow=False),
                 dict(text='Grapes',x=0.50,y=1.08, font_size=15, showarrow=False),
                 dict(text='Orange',x=0.74,y=1.08, font_size=15, showarrow=False),
                dict(text='Mango',x=0.94,y=1.08, font_size=15, showarrow=False),
                dict(text='Coconut',x=0.06,y=0.46, font_size=15, showarrow=False),
                 dict(text='Papaya',x=0.26,y=0.46, font_size=15, showarrow=False),
                 dict(text='Pomegranate',x=0.50,y=0.46, font_size=15, showarrow=False),
                 dict(text='Watermelon',x=0.74,y=0.46, font_size=15, showarrow=False),
                dict(text='Muskmelon',x=0.94,y=0.46, font_size=15, showarrow=False)])
fig.show()
```




```
crop_scatter =df[(df['crop']=='rice') |
                      (df['crop']=='jute') |
                      (df['crop']=='cotton') |
                     (df['crop']=='maize') |
                     (df['crop']=='lentil')]

fig = px.scatter(crop_scatter, x="temperature", y="humidity", color="crop", symbol="crop")
fig.update_layout(plot_bgcolor='white')
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```




```
df['season']
```

```
0       5
1       4
2       0
3       1
4       0
       ..
2195    4
2196    0
2197    2
2198    4
2199    5
Name: season, Length: 2200, dtype: int64
```

```
fig, ax = plt.subplots(1, 1, figsize=(15, 9))
sns.heatmap(df.corr(), annot=True,cmap='Wistia' )
ax.set(xlabel='features')
ax.set(ylabel='features')

plt.title('Correlation between different features', fontsize = 15, c='black')
plt.show()
```


```
target=['crop']
features=['n','p','k','temperature','humidity','ph','rainfall','season']
```

```
X=df[features]
y=df[target]
```

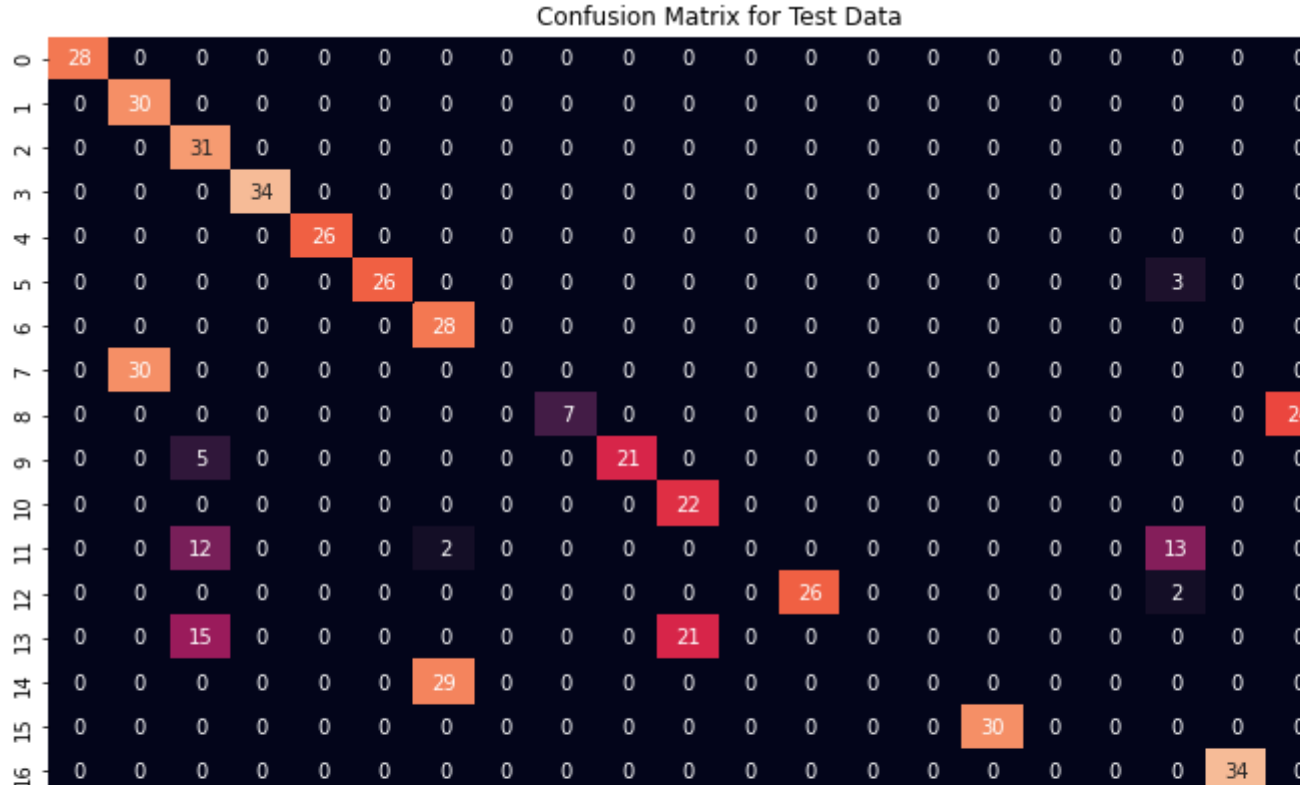
```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain,Ytest = train_test_split(X, y, test_size = 0.3,
                                                shuffle = True, random_state = 0)
```

```
# Initializing empty lists to append all model's name and corresponding name
acc = []
model = []
```

ID3 DECISION TREE

```
from sklearn.tree import DecisionTreeClassifier

DecisionTree = DecisionTreeClassifier(criterion="entropy",random_state=2,max_depth=4)

DecisionTree.fit(Xtrain,Ytrain)

predicted_values = DecisionTree.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Decision Tree')
print("DecisionTrees's Accuracy is: ", x*100)

print(classification_report(Ytest,predicted_values))
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(Ytest, predicted_values), annot = True)
plt.title("Confusion Matrix for Test Data")
plt.show()
```

```
DecisionTrees's Accuracy is:  65.30303030303031
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        28
      banana       0.50      1.00      0.67        30
   blackgram       0.49      1.00      0.66        31
    chickpea       1.00      1.00      1.00        34
     coconut       1.00      1.00      1.00        26
      coffee       1.00      0.90      0.95        29
      cotton       0.47      1.00      0.64        28
      grapes       0.00      0.00      0.00        30
        jute       1.00      0.23      0.37        31
 kidneybeans       1.00      0.81      0.89        26
      lentil       0.51      1.00      0.68        22
       maize       0.00      0.00      0.00        27
       mango       1.00      0.93      0.96        28
   mothbeans       0.00      0.00      0.00        36
    mungbean       0.00      0.00      0.00        29
   muskmelon       1.00      1.00      1.00        30
      orange       0.00      0.00      0.00        34
      papaya       0.00      0.00      0.00        39
  pigeonpeas       0.61      1.00      0.76        28
 pomegranate       0.48      1.00      0.65        32
        rice       0.37      1.00      0.54        37
  watermelon       1.00      1.00      1.00        25

    accuracy                           0.65       660
   macro avg       0.57      0.68      0.58       660
weighted avg       0.54      0.65      0.56       660
```



NAIVE BAYES ALGORITHM

```
from sklearn.naive_bayes import GaussianNB
```

```
NaiveBayes = GaussianNB()

NaiveBayes.fit(Xtrain,Ytrain)

predicted_values = NaiveBayes.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Naive Bayes')
print("Naive Bayes's Accuracy is: ", x*100)

print(classification_report(Ytest,predicted_values))
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(Ytest,predicted_values), annot = True)
plt.title("Confusion Matrix for Test Data")
plt.show()
```

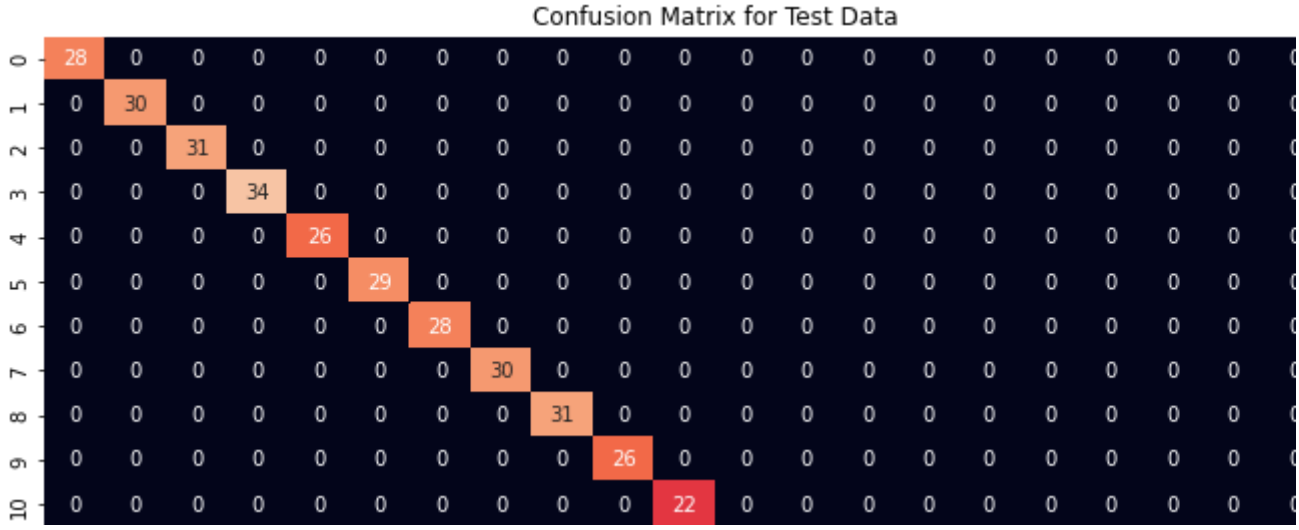
```
Naive Bayes's Accuracy is:  99.39393939393939
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        28
      banana       1.00      1.00      1.00        30
   blackgram       1.00      1.00      1.00        31
    chickpea       1.00      1.00      1.00        34
     coconut       1.00      1.00      1.00        26
      coffee       1.00      1.00      1.00        29
      cotton       1.00      1.00      1.00        28
      grapes       1.00      1.00      1.00        30
        jute       0.91      0.97      0.94        31
  kidneybeans       1.00      1.00      1.00        26
      lentil       1.00      1.00      1.00        22
       maize       1.00      1.00      1.00        27
       mango       1.00      1.00      1.00        28
   mothbeans       1.00      1.00      1.00        36
    mungbean       1.00      1.00      1.00        29
   muskmelon       1.00      1.00      1.00        30
```

KNN Classifier

```
   pigeonpeas       1.00      1.00      1.00        28
```

```
from sklearn.neighbors import KNeighborsClassifier
clf_knn = KNeighborsClassifier(n_neighbors=4)
clf_knn.fit(Xtrain,Ytrain)
y_pred1 = clf_knn.predict(Xtest)
accu=metrics.accuracy_score(Ytest,y_pred1)

acc.append(accu)
model.append('KNN Classifier')


print("Accuracy Score of KNN:",accu*100)



print(classification_report(Ytest,y_pred1))
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(Ytest,y_pred1), annot = True)
plt.title("Confusion Matrix for Test Data")
plt.show()
```

```
Accuracy Score of KNN: 97.72727272727273
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        28
      banana       1.00      1.00      1.00        30
   blackgram       0.97      1.00      0.98        31
    chickpea       1.00      1.00      1.00        34
     coconut       1.00      1.00      1.00        26
      coffee       1.00      1.00      1.00        29
      cotton       0.97      1.00      0.98        28
      grapes       1.00      1.00      1.00        30
        jute       0.79      1.00      0.89        31
 kidneybeans       0.87      1.00      0.93        26
      lentil       0.96      1.00      0.98        22
       maize       1.00      0.96      0.98        27
       mango       1.00      1.00      1.00        28
   mothbeans       1.00      0.94      0.97        36
    mungbean       1.00      1.00      1.00        29
   muskmelon       1.00      1.00      1.00        30
      orange       1.00      1.00      1.00        34
      papaya       1.00      0.97      0.99        39
  pigeonpeas       1.00      0.86      0.92        28
 pomegranate       1.00      1.00      1.00        32
        rice       1.00      0.81      0.90        37
  watermelon       1.00      1.00      1.00        25

    accuracy                           0.98       660
   macro avg       0.98      0.98      0.98       660
weighted avg       0.98      0.98      0.98       660
```



LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression

LogReg = LogisticRegression(random_state=2)

LogReg.fit(Xtrain,Ytrain)

predicted_values = LogReg.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
```

```
acc.append(x)
model.append('Logistic Regression')
print("Logistic Regression's Accuracy is: ", x*100,'\n')

print(classification_report(Ytest,predicted_values))
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(Ytest,predicted_values), annot = True)
plt.title("Confusion Matrix for Test Data")
plt.show()
```

```
Logistic Regression's Accuracy is:  95.9090909090909

              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        28
      banana       1.00      1.00      1.00        30
   blackgram       0.84      0.87      0.86        31
    chickpea       1.00      1.00      1.00        34
     coconut       1.00      1.00      1.00        26
      coffee       1.00      1.00      1.00        29
      cotton       0.82      0.96      0.89        28
      grapes       1.00      1.00      1.00        30
        jute       0.88      0.97      0.92        31
```

RANDOM FOREST

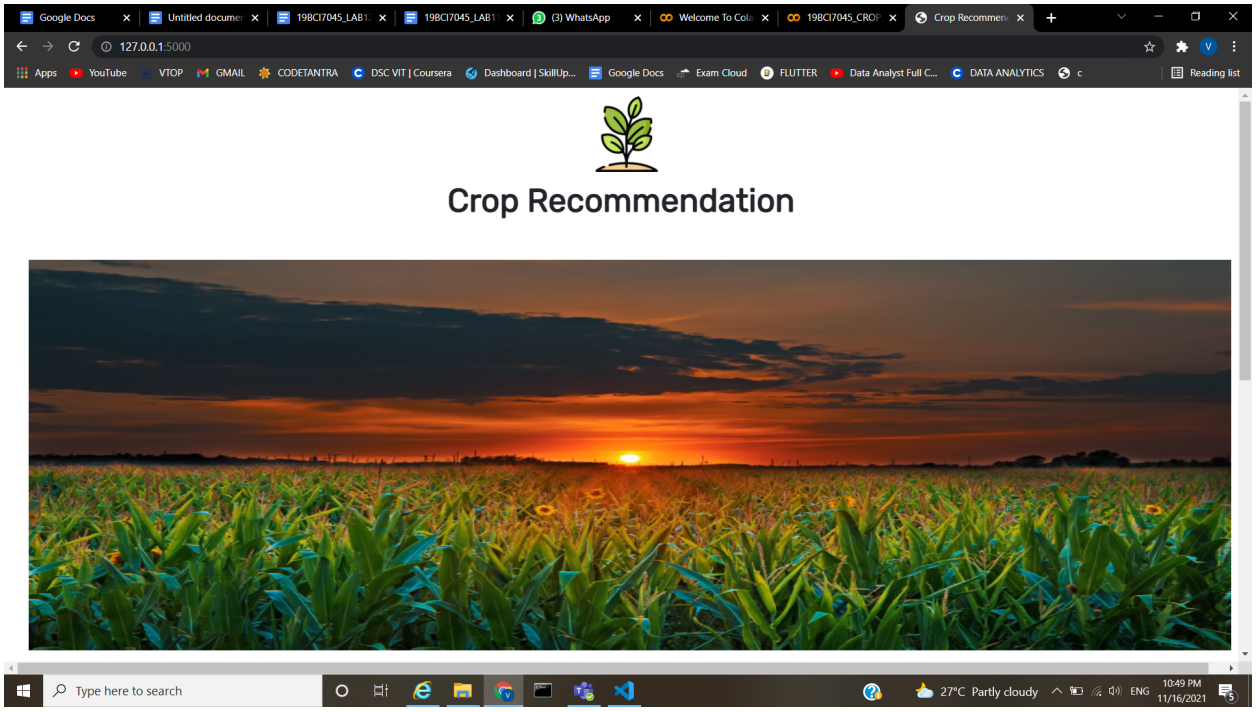
```
       mango       1.00      1.00      1.00        28
```

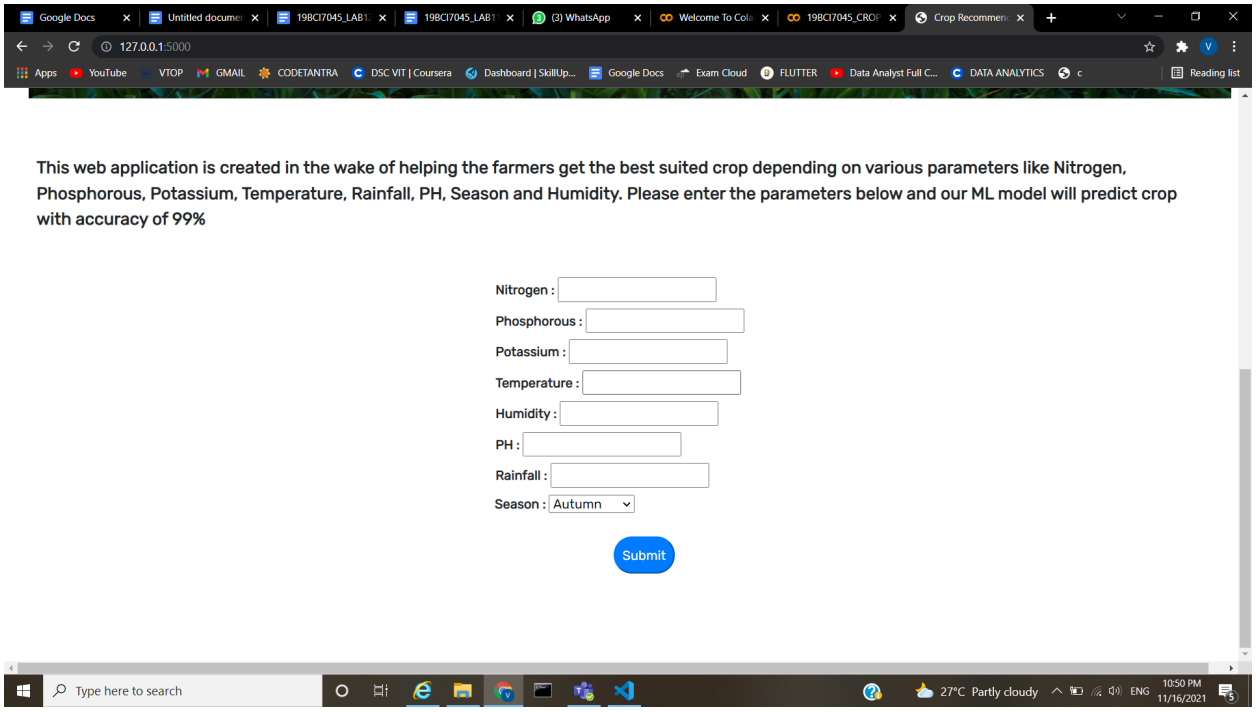
```
from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n_estimators=20, random_state=0)
RF.fit(Xtrain,Ytrain)

predicted_values = RF.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Random Forest')
print("RF's Accuracy is: ", x*100)

print(classification_report(Ytest,predicted_values))
plt.figure(figsize = (15,9))
sns.heatmap(confusion_matrix(Ytest,predicted_values), annot = True)
plt.title("Confusion Matrix for Test Data")
plt.show()
```

```
RF's Accuracy is:  99.69696969696969
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        28
      banana       1.00      1.00      1.00        30
   blackgram       1.00      0.97      0.98        31
    chickpea       1.00      1.00      1.00        34
     coconut       1.00      1.00      1.00        26
      coffee       1.00      1.00      1.00        29
      cotton       1.00      1.00      1.00        28
      grapes       1.00      1.00      1.00        30
        jute       0.97      1.00      0.98        31
 kidneybeans       1.00      1.00      1.00        26
      lentil       1.00      1.00      1.00        22
       maize       0.96      1.00      0.98        27
       mango       1.00      1.00      1.00        28
   mothbeans       1.00      1.00      1.00        36
    mungbean       1.00      1.00      1.00        29
   muskmelon       1.00      1.00      1.00        30
      orange       1.00      1.00      1.00        34
      papaya       1.00      1.00      1.00        39
  pigeonpeas       1.00      1.00      1.00        28
 pomegranate       1.00      1.00      1.00        32
        rice       1.00      0.97      0.99        37
  watermelon       1.00      1.00      1.00        25

    accuracy                           1.00       660
   macro avg       1.00      1.00      1.00       660
weighted avg       1.00      1.00      1.00       660
```


```
plt.figure(figsize=[10,5],dpi = 100)
plt.title('Accuracy Comparison')
plt.xlabel('Accuracy')
plt.ylabel('Algorithm')
sns.barplot(x = acc,y = model,palette='dark')
```

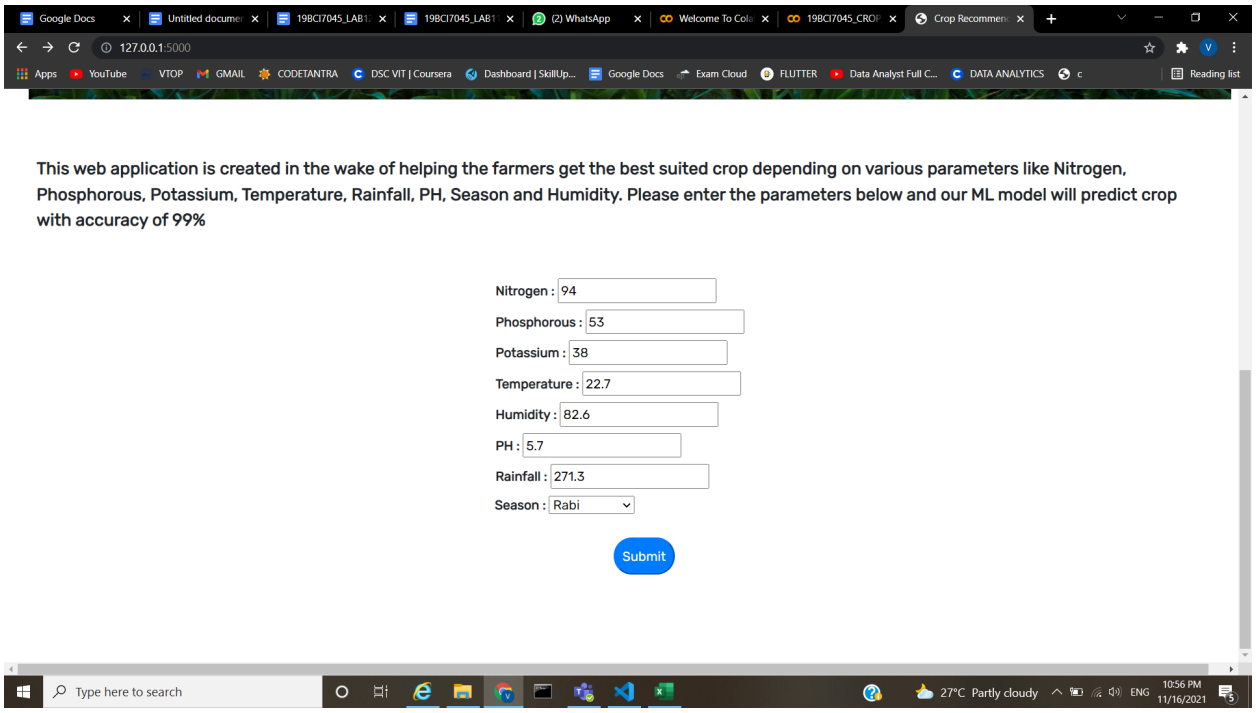
DISPLAYING WEBPAGE :

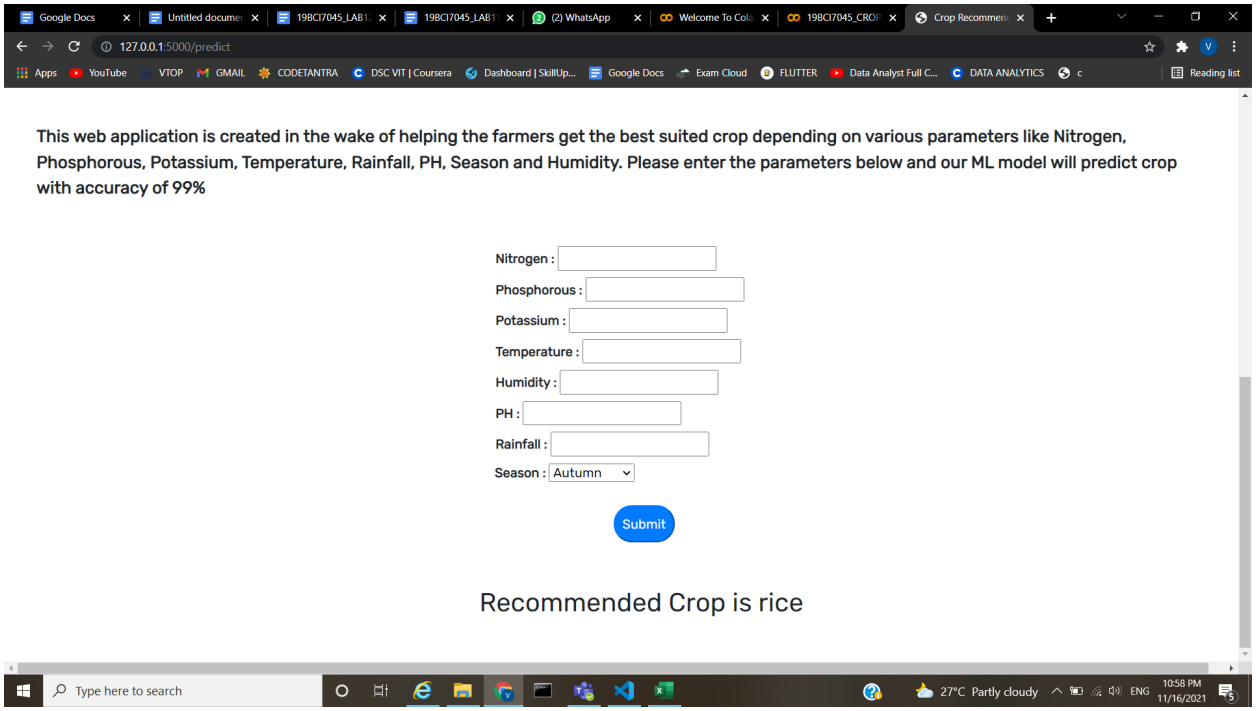




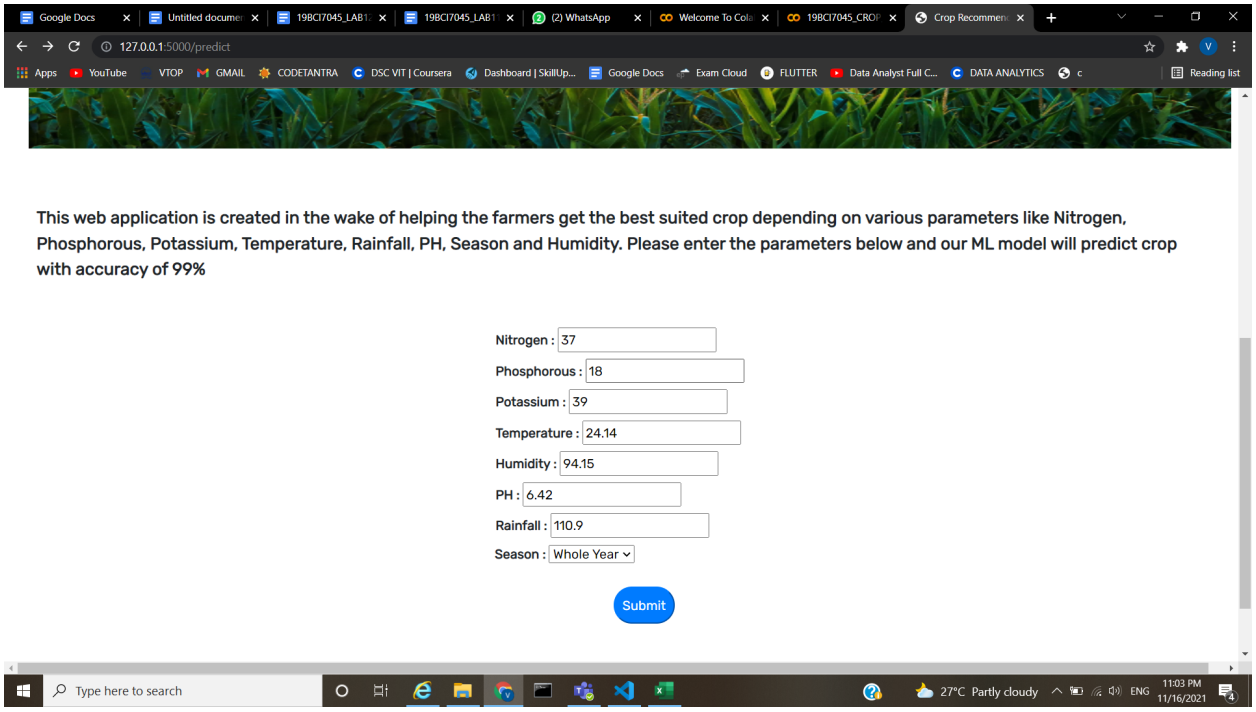
EXAMPLES :

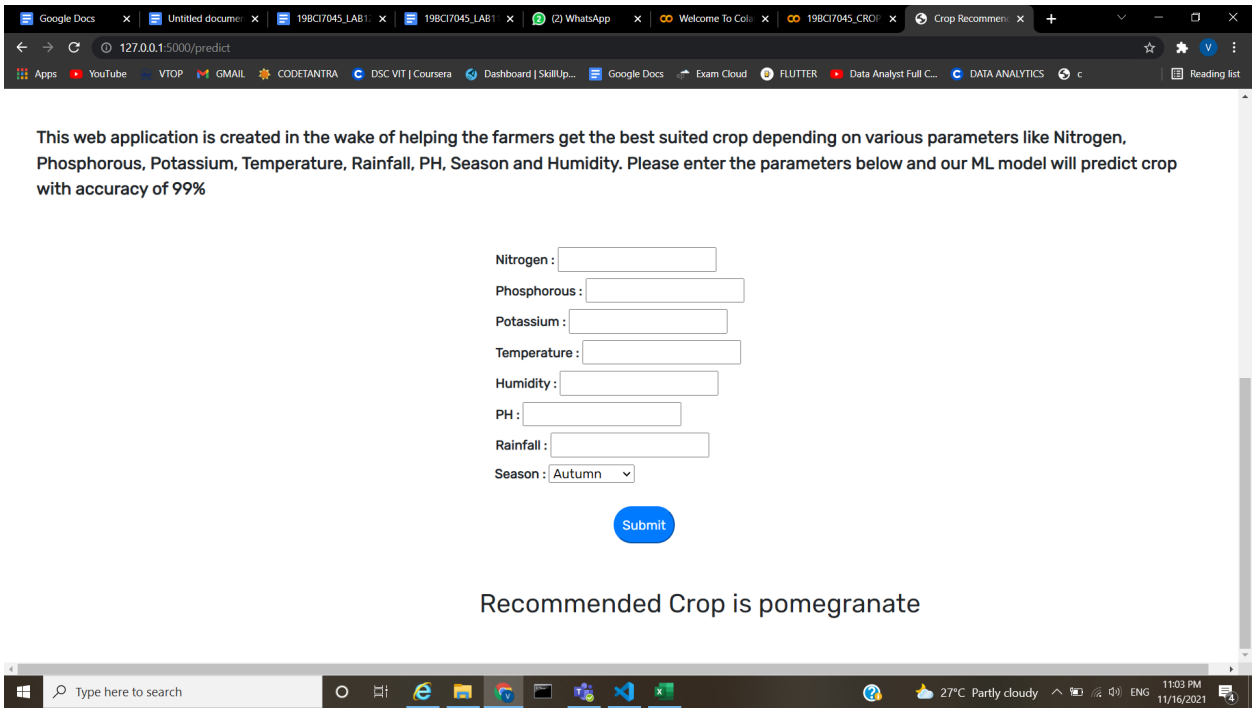
1.


127.0.0.1:5000
This web application is created in the wake of helping the farmers get the best suited crop depending on various parameters like Nitrogen, Phosphorous, Potassium, Temperature, Rainfall, PH, Season and Humidity. Please enter the parameters below and our ML model will predict crop with accuracy of 99%
Nitrogen : 94
Phosphorous : 53
Potassium : 38
Temperature : 22.7
Humidity : 82.6
PH : 5.7
Rainfall : 271.3
Season : Rabi
Submit

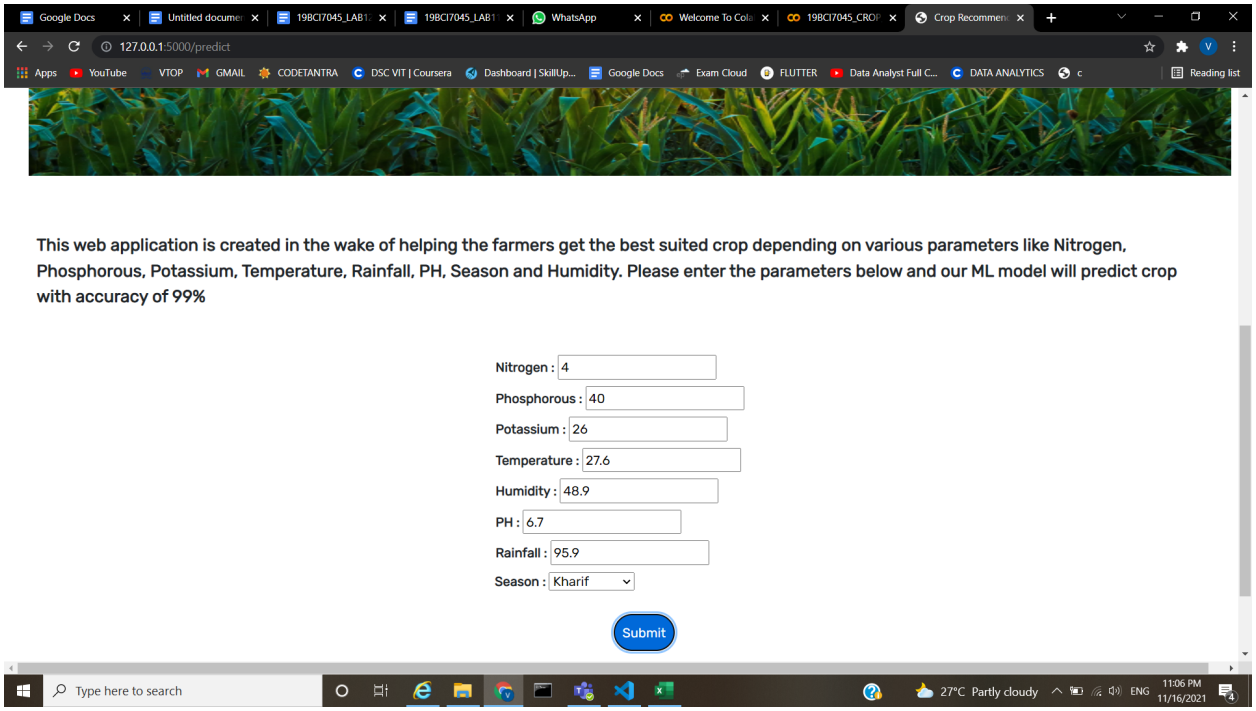

127.0.0.1:5000/predict
This web application is created in the wake of helping the farmers get the best suited crop depending on various parameters like Nitrogen, Phosphorous, Potassium, Temperature, Rainfall, PH, Season and Humidity. Please enter the parameters below and our ML model will predict crop with accuracy of 99%
Nitrogen :
Phosphorous :
Potassium :
Temperature :
Humidity :
PH :
Rainfall :
Season : Autumn
Submit
Recommended Crop is rice

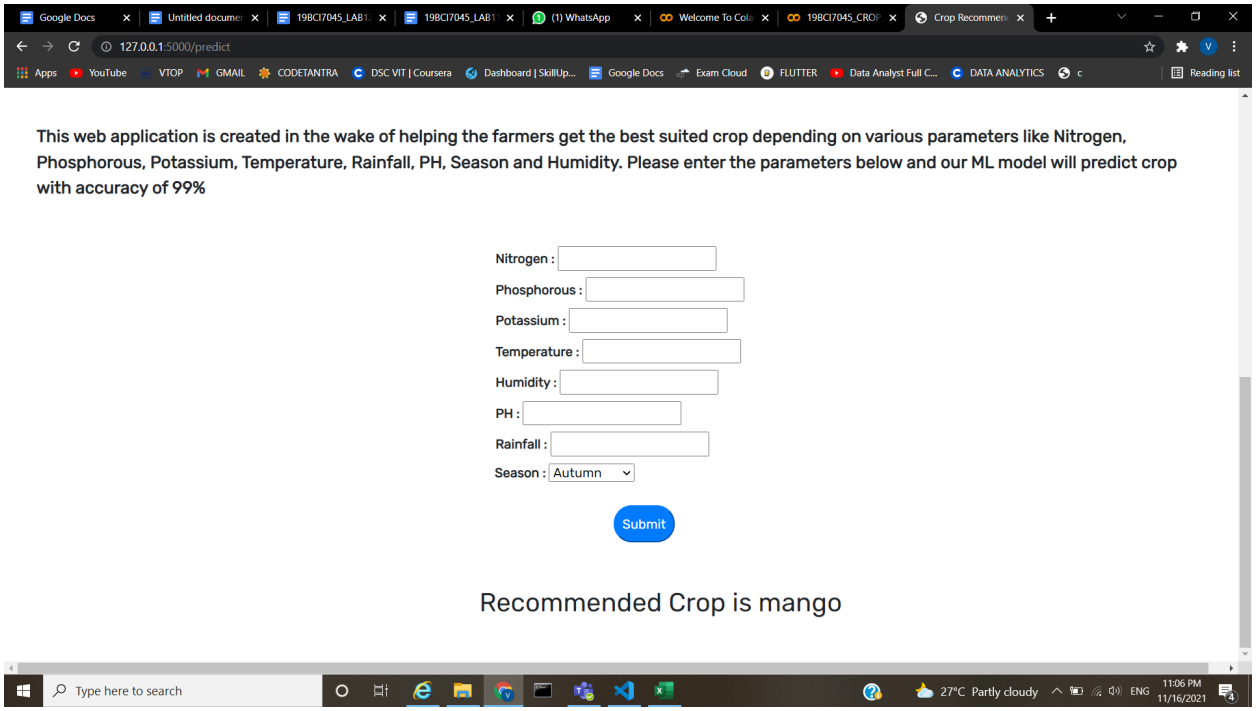
2.


127.0.0.1:5000/predict
This web application is created in the wake of helping the farmers get the best suited crop depending on various parameters like Nitrogen, Phosphorous, Potassium, Temperature, Rainfall, PH, Season and Humidity. Please enter the parameters below and our ML model will predict crop with accuracy of 99%
Nitrogen : 37
Phosphorous : 18
Potassium : 39
Temperature : 24.14
Humidity : 94.15
PH : 6.42
Rainfall : 110.9
Season : Whole Year
Submit

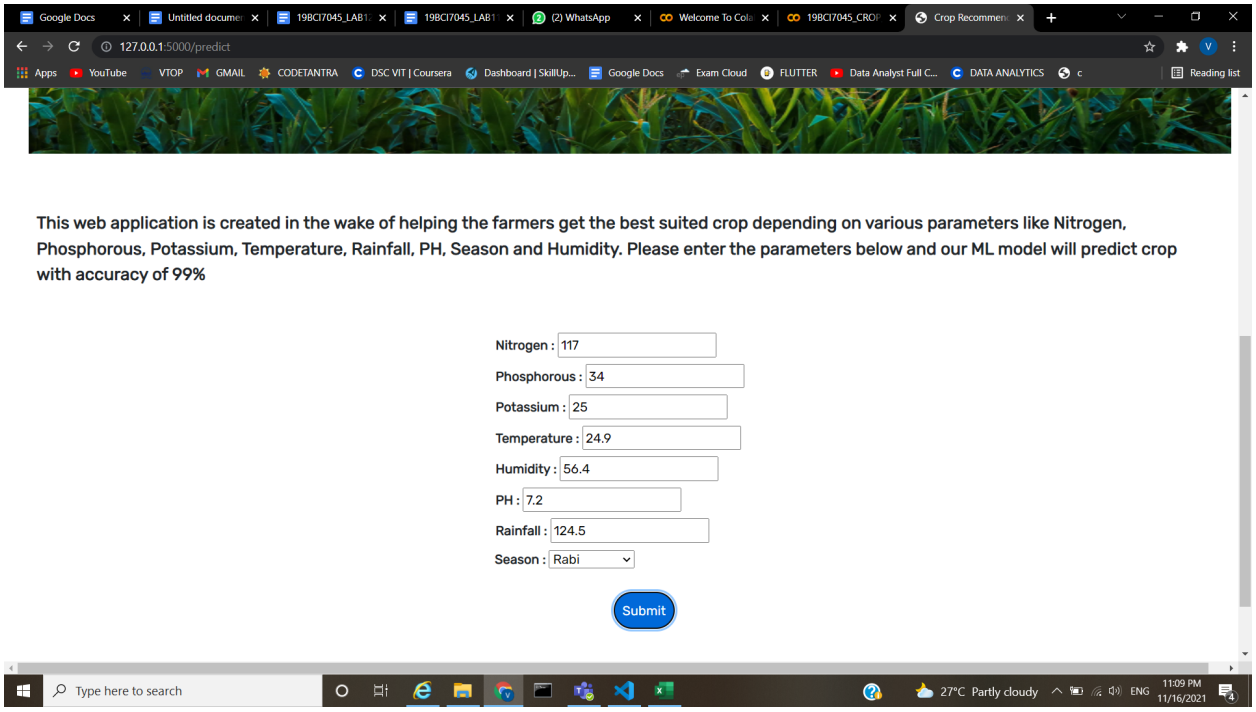

127.0.0.1:5000/predict
This web application is created in the wake of helping the farmers get the best suited crop depending on various parameters like Nitrogen, Phosphorous, Potassium, Temperature, Rainfall, PH, Season and Humidity. Please enter the parameters below and our ML model will predict crop with accuracy of 99%
Nitrogen :
Phosphorous :
Potassium :
Temperature :
Humidity :
PH :
Rainfall :
Season : Autumn
Submit
Recommended Crop is pomegranate

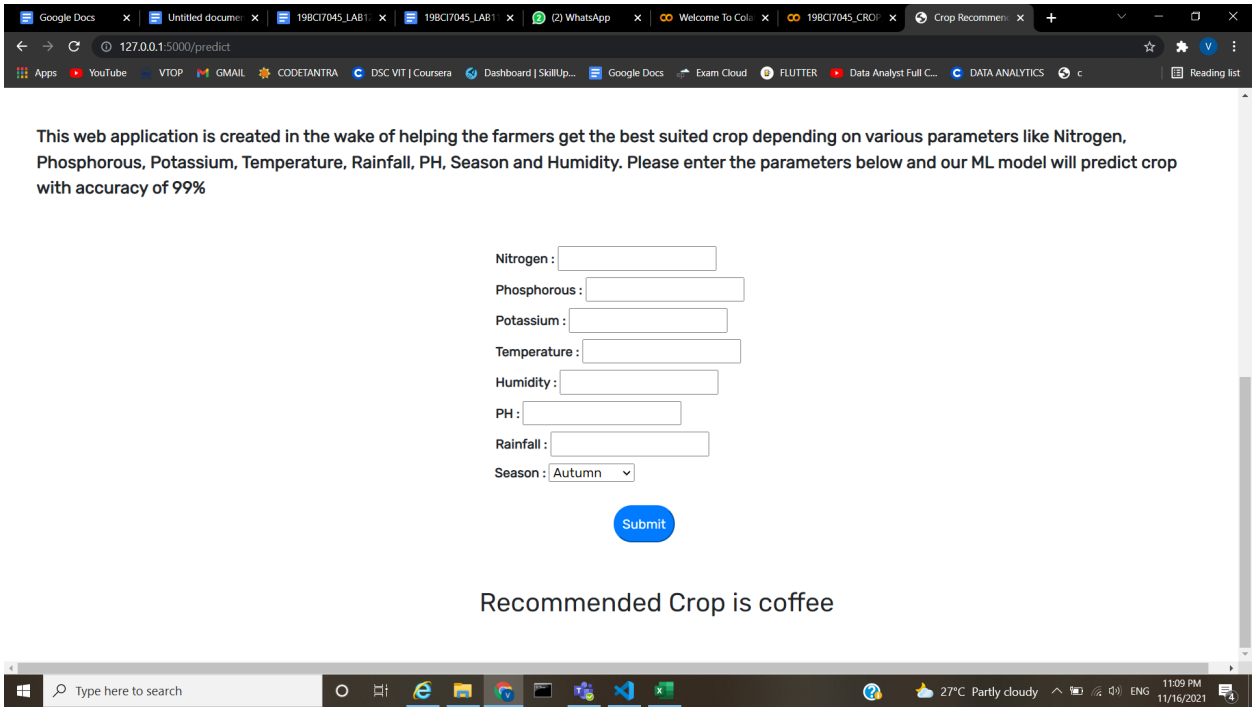
3.


127.0.0.1:5000/predict
This web application is created in the wake of helping the farmers get the best suited crop depending on various parameters like Nitrogen, Phosphorous, Potassium, Temperature, Rainfall, PH, Season and Humidity. Please enter the parameters below and our ML model will predict crop with accuracy of 99%
Nitrogen : 4
Phosphorous : 40
Potassium : 26
Temperature : 27.6
Humidity : 48.9
PH : 6.7
Rainfall : 95.9
Season : Kharif
Submit
27°C Partly cloudy
11:06 PM
11/16/2021

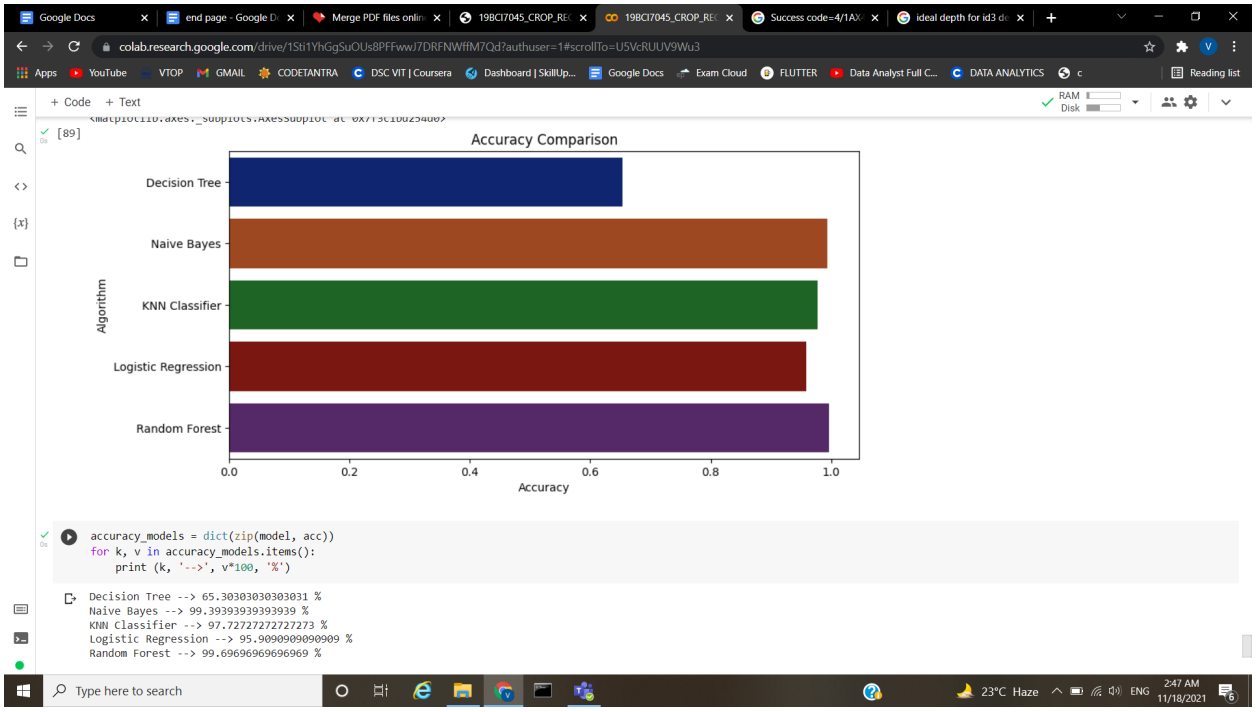

127.0.0.1:5000/predict
This web application is created in the wake of helping the farmers get the best suited crop depending on various parameters like Nitrogen, Phosphorous, Potassium, Temperature, Rainfall, PH, Season and Humidity. Please enter the parameters below and our ML model will predict crop with accuracy of 99%
Nitrogen :
Phosphorous :
Potassium :
Temperature :
Humidity :
PH :
Rainfall :
Season : Autumn
Submit
Recommended Crop is mango
27°C Partly cloudy
11:06 PM
11/16/2021

4.


127.0.0.1:5000/predict
This web application is created in the wake of helping the farmers get the best suited crop depending on various parameters like Nitrogen, Phosphorous, Potassium, Temperature, Rainfall, PH, Season and Humidity. Please enter the parameters below and our ML model will predict crop with accuracy of 99%
Nitrogen : 117
Phosphorous : 34
Potassium : 25
Temperature : 24.9
Humidity : 56.4
PH : 7.2
Rainfall : 124.5
Season : Rabi
Submit


127.0.0.1:5000/predict
This web application is created in the wake of helping the farmers get the best suited crop depending on various parameters like Nitrogen, Phosphorous, Potassium, Temperature, Rainfall, PH, Season and Humidity. Please enter the parameters below and our ML model will predict crop with accuracy of 99%
Nitrogen :
Phosphorous :
Potassium :
Temperature :
Humidity :
PH :
Rainfall :
Season : Autumn
Submit
Recommended Crop is coffee

ACCURACY :



Highest Accuracy : 99.69% (Random Forest)

Least Accuracy : 65.3 % (Decision Tree)

Colab link :

https://colab.research.google.com/drive/1Sti1YhGgSuOUs8PFFwwJ7DRFNWffM7Qd?usp=sharing