crop-recommendation

September 10, 2023

0.1 Installing all the required libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import missingno as msno
from sklearn.metrics import classification_report ,␣
↪accuracy_score,confusion_matrix
import warnings
warnings.filterwarnings('ignore')
```

0.2 Installing intel ONE API scikit learn library

[2]:
```
!pip install scikit-learn-intelex
```

```
Collecting scikit-learn-intelex
  Downloading scikit_learn_intelex-2023.2.1-py310-none-manylinux1_x86_64.whl
(128 kB)
                           128.7/128.7

kB 2.1 MB/s eta 0:00:00
Collecting daal4py==2023.2.1 (from scikit-learn-intelex)
  Downloading daal4py-2023.2.1-py310-none-manylinux1_x86_64.whl (14.0 MB)
                           14.0/14.0 MB
34.6 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn>=0.22 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn-intelex) (1.2.2)
Collecting daal==2023.2.1 (from daal4py==2023.2.1->scikit-learn-intelex)
  Downloading daal-2023.2.1-py2.py3-none-manylinux1_x86_64.whl (75.3 MB)
                           75.3/75.3 MB
12.2 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.19 in
/usr/local/lib/python3.10/dist-packages (from daal4py==2023.2.1->scikit-learn-
intelex) (1.23.5)
Requirement already satisfied: tbb==2021.* in /usr/local/lib/python3.10/dist-
packages (from daal==2023.2.1->daal4py==2023.2.1->scikit-learn-intelex)
```

```
(2021.10.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn>=0.22->scikit-learn-intelex) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn>=0.22->scikit-learn-intelex) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->scikit-learn-
intelex) (3.2.0)
Installing collected packages: daal, daal4py, scikit-learn-intelex
Successfully installed daal-2023.2.1 daal4py-2023.2.1 scikit-learn-
intelex-2023.2.1
```

0.3 Using Intel(R) Extension for Scikit-learn

[3]:
```
from sklearnex import patch_sklearn
patch_sklearn()
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

```
Intel(R) Extension for Scikit-learn* enabled (https://github.com/intel/scikit-
learn-intelex)
```

[ ]:
```
#!pip install modin
```

```
Collecting modin
  Downloading modin-0.23.0-py3-none-any.whl (1.1 MB)
                           1.1/1.1 MB
6.5 MB/s eta 0:00:00
Collecting pandas<2.1,>=2 (from modin)
  Downloading
pandas-2.0.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.3
MB)
                           12.3/12.3 MB
42.6 MB/s eta 0:00:00
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from modin) (23.1)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-
packages (from modin) (1.22.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from modin) (2023.6.0)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages
(from modin) (5.9.5)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas<2.1,>=2->modin) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas<2.1,>=2->modin) (2022.7.1)
Collecting tzdata>=2022.1 (from pandas<2.1,>=2->modin)
  Downloading tzdata-2023.3-py2.py3-none-any.whl (341 kB)
```

```
                                  341.8/341.8 kB
34.7 MB/s eta 0:00:00
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas<2.1,>=2->modin) (1.16.0)
Installing collected packages: tzdata, pandas, modin
  Attempting uninstall: pandas
    Found existing installation: pandas 1.5.3
    Uninstalling pandas-1.5.3:
      Successfully uninstalled pandas-1.5.3
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
google-colab 1.0.0 requires pandas==1.5.3, but you have pandas 2.0.3 which is
incompatible.
Successfully installed modin-0.23.0 pandas-2.0.3 tzdata-2023.3
```

[ ]:
```
#import modin.pandas as md
```

0.4 Loading Dataset

[4]:
```
df =pd.read_csv('/content/Crop_recommendation.csv')
```

[5]:
```
df.head()
```

[5]:

	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

0.5 Data Preprocessing

[6]:
```
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

[7]:
```
df.columns =
 ↪['Nitrogen','Phosphorus','Potassium','Temperature','Humidity','pH','Rainfall','Label']
```

[8]:
```
df.isna().sum()
```

[8]:
```
Nitrogen       0
Phosphorus     0
Potassium      0
Temperature    0
Humidity       0
pH             0
Rainfall       0
Label          0
dtype: int64
```

[9]:
```
type(df)
```

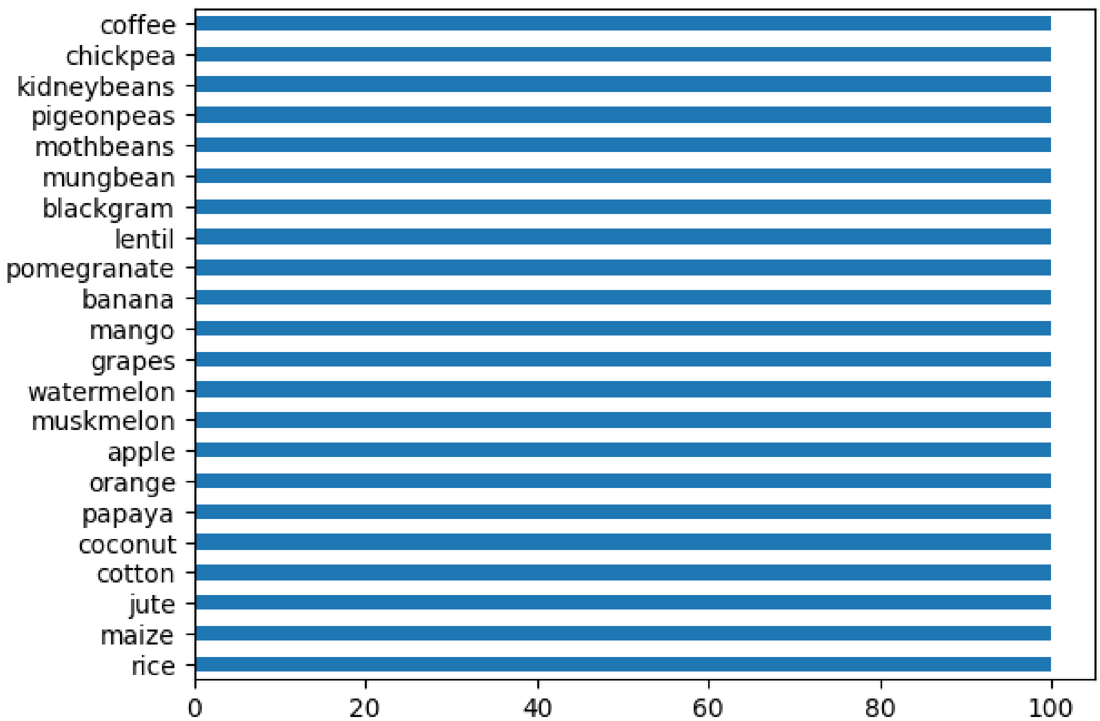
[9]: pandas.core.frame.DataFrame

[10]:
```
df.head()
```

[10]:
```
   Nitrogen  Phosphorus  Potassium  Temperature   Humidity        pH  \
0        90          42         43    20.879744  82.002744  6.502985
1        85          58         41    21.770462  80.319644  7.038096
2        60          55         44    23.004459  82.320763  7.840207
3        74          35         40    26.491096  80.158363  6.980401
4        78          42         42    20.130175  81.604873  7.628473

     Rainfall Label
0  202.935536  rice
1  226.655537  rice
2  263.964248  rice
3  242.864034  rice
4  262.717340  rice
```

0.6 Data Visualization

[11]:
```
df["Label"].value_counts().plot.barh()
plt.show()
```



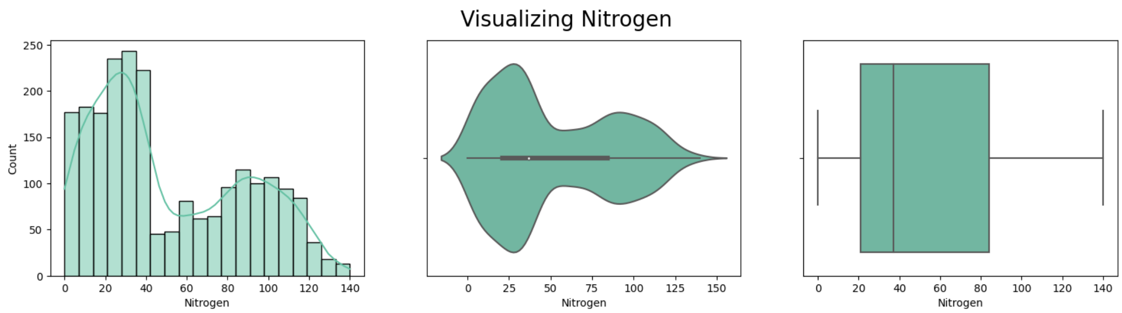
[12]:
```
df.describe()
```

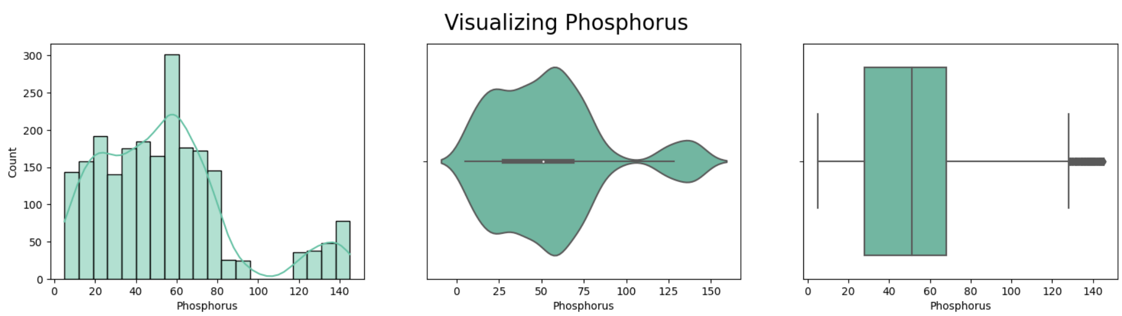
[12]:

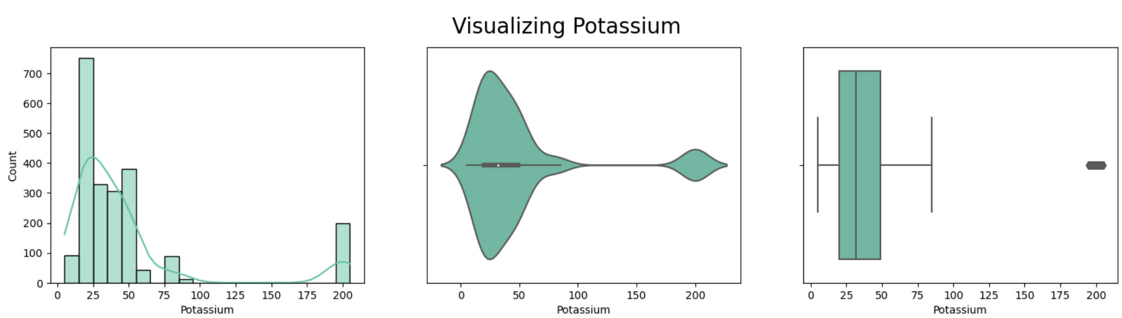
	Nitrogen	Phosphorus	Potassium	Temperature	Humidity
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779
std	36.917334	32.985883	50.647931	5.063749	22.263812
min	0.000000	5.000000	5.000000	8.825675	14.258040
25%	21.000000	28.000000	20.000000	22.769375	60.261953
50%	37.000000	51.000000	32.000000	25.598693	80.473146
75%	84.250000	68.000000	49.000000	28.561654	89.948771
max	140.000000	145.000000	205.000000	43.675493	99.981876

	pH	Rainfall
count	2200.000000	2200.000000
mean	6.469480	103.463655
std	0.773938	54.958389
min	3.504752	20.211267
25%	5.971693	64.551686
50%	6.425045	94.867624
75%	6.923643	124.267508
max	9.935091	298.560117

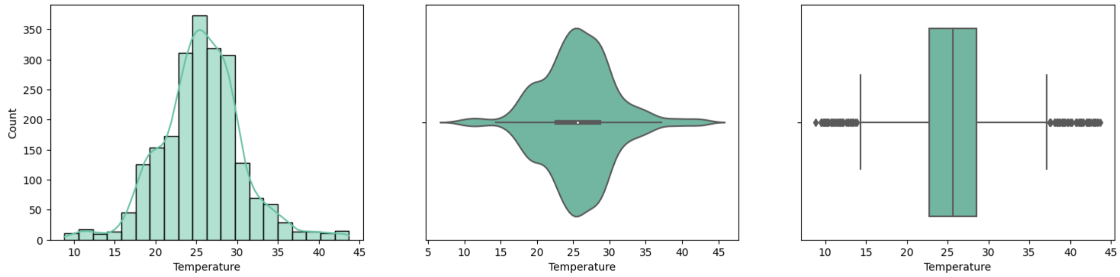
[13]:
```python
plt.style.use('fast')
sns.set_palette("Set2")
for i in df.columns[:-1]:
    fig,ax = plt.subplots(1,3,figsize=(18,4))
    sns.histplot(data = df,x=i,kde = True,bins = 20,ax = ax[0])
    sns.violinplot(data = df,x = i,ax =ax[1])
    sns.boxplot(data = df,x = i,ax =ax[2])
    plt.suptitle(f'Visualizing {i}',size =20)
```



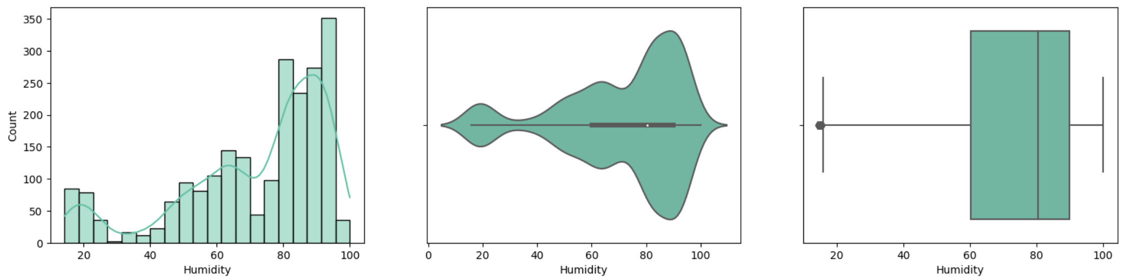




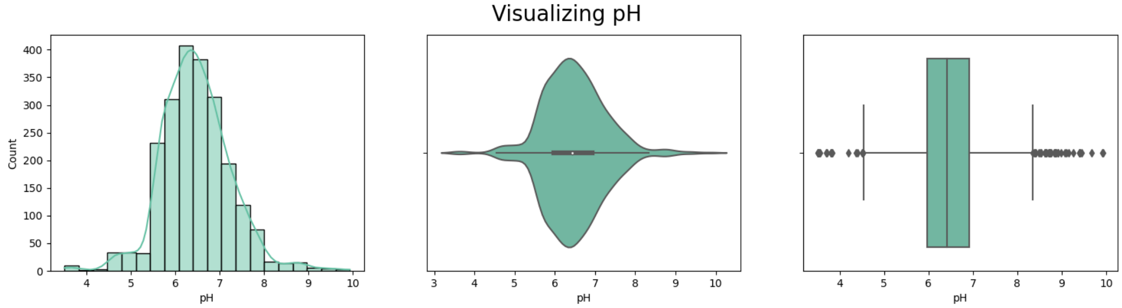
Visualizing Temperature



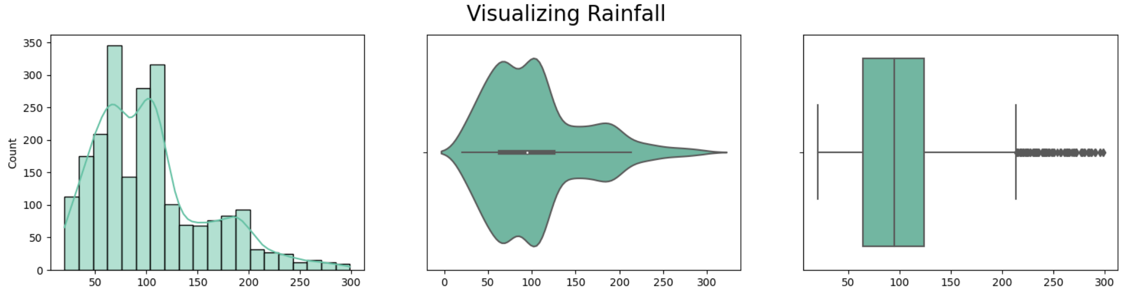
Visualizing Humidity



Visualizing pH



Visualizing Rainfall



0.7 Data Analysis

[14]:
```
grouped = df.groupby(by = 'Label').mean().reset_index()
grouped
```

[14]:

	Label	Nitrogen	Phosphorus	Potassium	Temperature	Humidity
0	apple	20.80	134.22	199.89	22.630942	92.333383
1	banana	100.23	82.01	50.05	27.376798	80.358123
2	blackgram	40.02	67.47	19.24	29.973340	65.118426
3	chickpea	40.09	67.79	79.92	18.872847	16.860439
4	coconut	21.98	16.93	30.59	27.409892	94.844272
5	coffee	101.20	28.74	29.94	25.540477	58.869846
6	cotton	117.77	46.24	19.56	23.988958	79.843474
7	grapes	23.18	132.53	200.11	23.849575	81.875228
8	jute	78.40	46.86	39.99	24.958376	79.639864
9	kidneybeans	20.75	67.54	20.05	20.115085	21.605357
10	lentil	18.77	68.36	19.41	24.509052	64.804785
11	maize	77.76	48.44	19.79	22.389204	65.092249
12	mango	20.07	27.18	29.92	31.208770	50.156573
13	mothbeans	21.44	48.01	20.23	28.194920	53.160418
14	mungbean	20.99	47.28	19.87	28.525775	85.499975
15	muskmelon	100.32	17.72	50.08	28.663066	92.342802
16	orange	19.58	16.55	10.01	22.765725	92.170209
17	papaya	49.88	59.05	50.04	33.723859	92.403388
18	pigeonpeas	20.73	67.73	20.29	27.741762	48.061633
19	pomegranate	18.87	18.75	40.21	21.837842	90.125504
20	rice	79.89	47.58	39.87	23.689332	82.272822
21	watermelon	99.42	17.00	50.22	25.591767	85.160375

	pH	Rainfall
0	5.929663	112.654779
1	5.983893	104.626980
2	7.133952	67.884151
3	7.336957	80.058977
4	5.976562	175.686646
5	6.790308	158.066295
6	6.912675	80.398043
7	6.025937	69.611829
8	6.732778	174.792798
9	5.749411	105.919778
10	6.927932	45.680454
11	6.245190	84.766988
12	5.766373	94.704515
13	6.831174	51.198487
14	6.723957	48.403601
15	6.358805	24.689952
16	7.016957	110.474969

```
17  6.741442  142.627839
18  5.794175  149.457564
19  6.429172  107.528442
20  6.425471  236.181114
21  6.495778   50.786219
```

[15]:
```python
for i in grouped.columns[1:]:
    print(f'Top 5 Most {i} requiring crops :')
    for j,k in grouped.sort_values(by = i,ascending =False)[:5][['Label',i]].
 ↪values:
        print(f'{j}-->{k}')
    print(f'********************************')
```

```
Top 5 Most Nitrogen requiring crops :
cotton-->117.77
coffee-->101.2
muskmelon-->100.32
banana-->100.23
watermelon-->99.42
********************************
Top 5 Most Phosphorus requiring crops :
apple-->134.22
grapes-->132.53
banana-->82.01
lentil-->68.36
chickpea-->67.79
********************************
Top 5 Most Potassium requiring crops :
grapes-->200.11
apple-->199.89
chickpea-->79.92
watermelon-->50.22
muskmelon-->50.08
********************************
Top 5 Most Temperature requiring crops :
papaya-->33.7238587388
mango-->31.2087701513
blackgram-->29.9733396789
muskmelon-->28.663065756
mungbean-->28.5257747353
********************************
Top 5 Most Humidity requiring crops :
coconut-->94.84427180610001
papaya-->92.4033876826
muskmelon-->92.34280196089999
apple-->92.3333828756
orange-->92.17020876340001
```

```
********************************
Top 5 Most pH requiring crops :
chickpea-->7.33695662374
blackgram-->7.13395162948
orange-->7.01695745276
lentil-->6.927931571609999
cotton-->6.91267549578
********************************
Top 5 Most Rainfall requiring crops :
rice-->236.181113594
coconut-->175.686645804
jute-->174.792797536
coffee-->158.066294882
pigeonpeas-->149.4575638135
********************************
```

[16]:

```
for i in grouped.columns[1:]:
    print(f'Top 5 Least {i} requiring crops:')
    print(f'********************************')
    for j ,k in grouped.sort_values(by=i)[:5][['Label',i]].values:
        print(f'{j} --> {k}')
    print(f'********************************')
```

```
Top 5 Least Nitrogen requiring crops:
********************************
lentil --> 18.77
pomegranate --> 18.87
orange --> 19.58
mango --> 20.07
pigeonpeas --> 20.73
********************************
Top 5 Least Phosphorus requiring crops:
********************************
orange --> 16.55
coconut --> 16.93
watermelon --> 17.0
muskmelon --> 17.72
pomegranate --> 18.75
********************************
Top 5 Least Potassium requiring crops:
********************************
orange --> 10.01
blackgram --> 19.24
lentil --> 19.41
cotton --> 19.56
maize --> 19.79
********************************
Top 5 Least Temperature requiring crops:
```

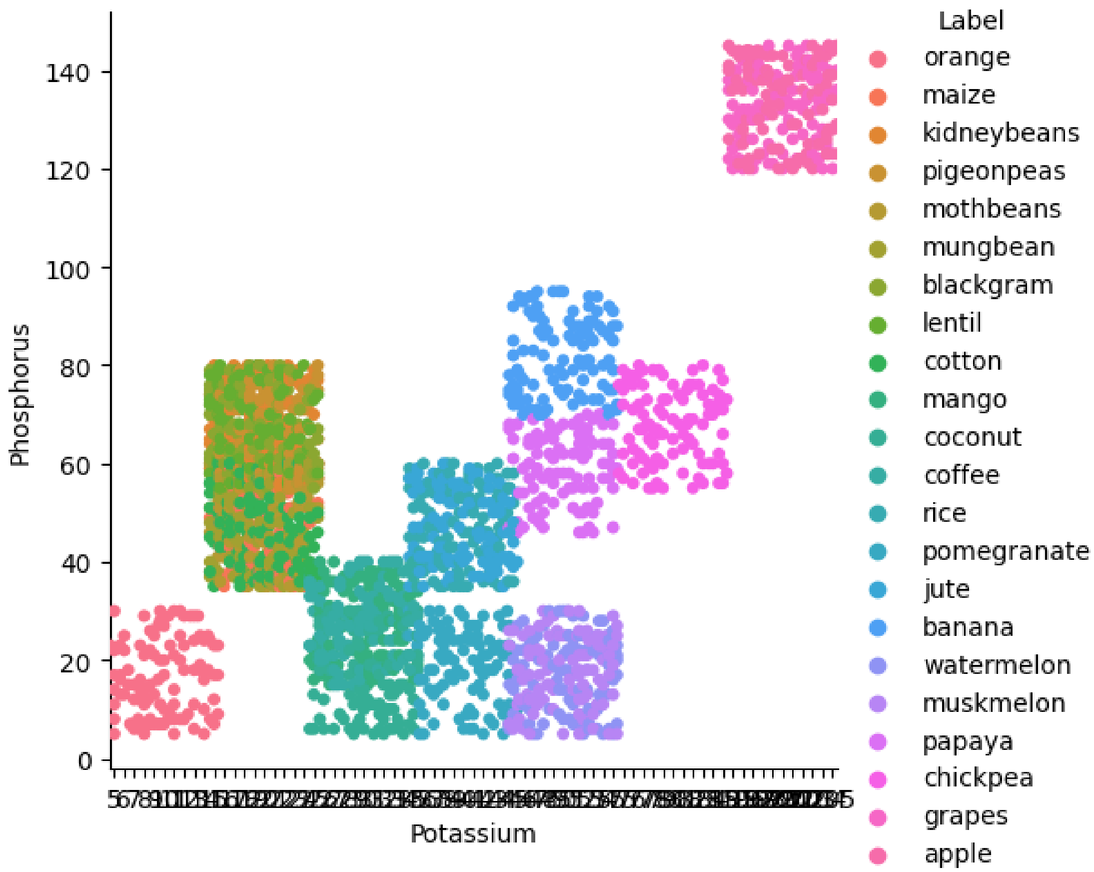
```
********************************
chickpea --> 18.8728467519
kidneybeans --> 20.1150846851
pomegranate --> 21.837841721999997
maize --> 22.3892039102
apple --> 22.6309424132
********************************
Top 5 Least Humidity requiring crops:
********************************
chickpea --> 16.8604394237
kidneybeans --> 21.6053567295
pigeonpeas --> 48.0616330847
mango --> 50.1565726953
mothbeans --> 53.16041802790001
********************************
Top 5 Least pH requiring crops:
********************************
kidneybeans --> 5.749410585870001
mango --> 5.766372799660001
pigeonpeas --> 5.794174879790001
apple --> 5.929662931809999
coconut --> 5.97656212619
********************************
Top 5 Least Rainfall requiring crops:
********************************
muskmelon --> 24.689952066
lentil --> 45.680454204
mungbean --> 48.403600902899996
watermelon --> 50.7862189449
mothbeans --> 51.198487045700006
********************************
```

[17]:
```
df.head()
```

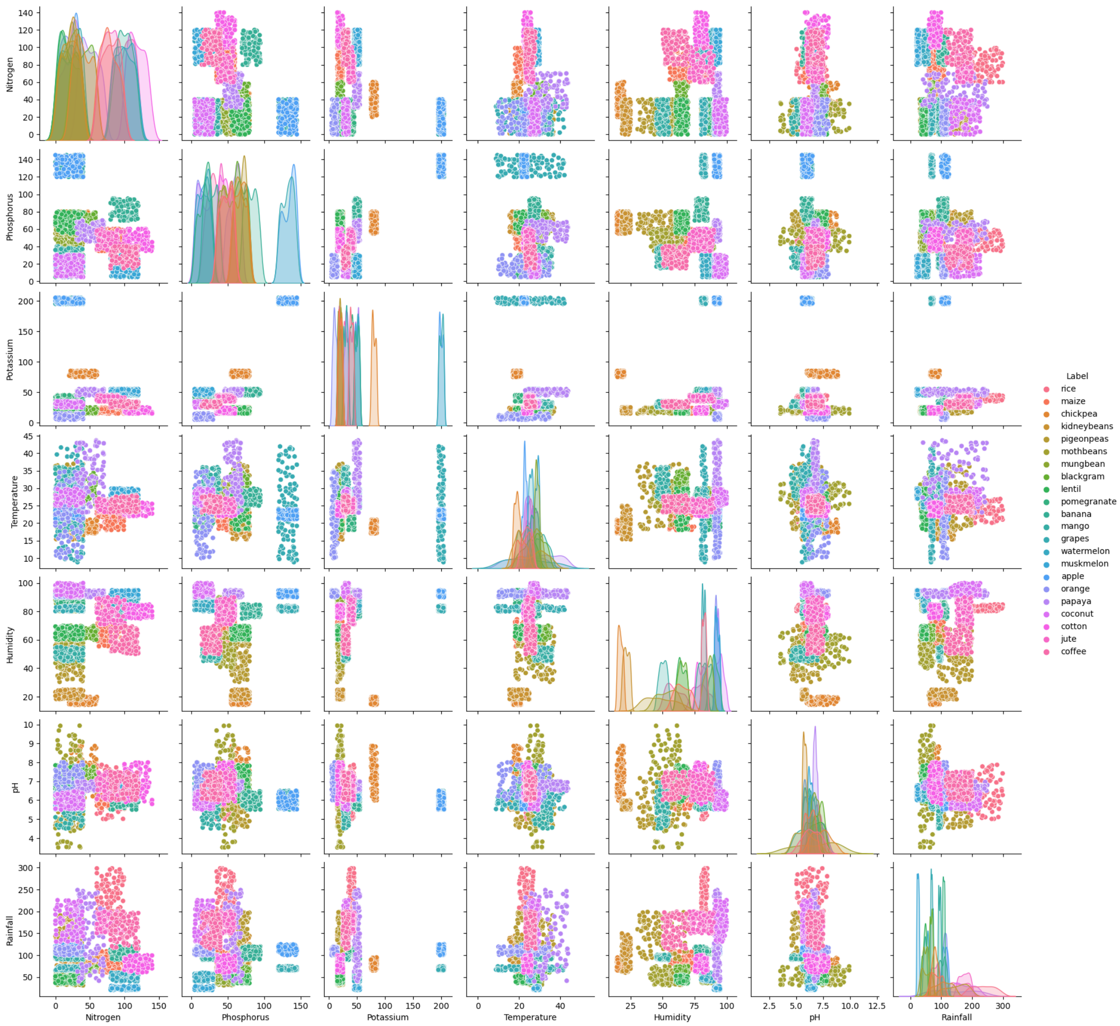
[17]:
```
   Nitrogen  Phosphorus  Potassium  Temperature   Humidity        pH  \
0        90          42         43    20.879744  82.002744  6.502985
1        85          58         41    21.770462  80.319644  7.038096
2        60          55         44    23.004459  82.320763  7.840207
3        74          35         40    26.491096  80.158363  6.980401
4        78          42         42    20.130175  81.604873  7.628473

     Rainfall Label
0  202.935536  rice
1  226.655537  rice
2  263.964248  rice
3  242.864034  rice
4  262.717340  rice
```

[18]:
```
sns.catplot(data=df, x="Potassium", y="Phosphorus", hue="Label", kind="swarm")
```

[18]: <seaborn.axisgrid.FacetGrid at 0x7c310ff3ffa0>



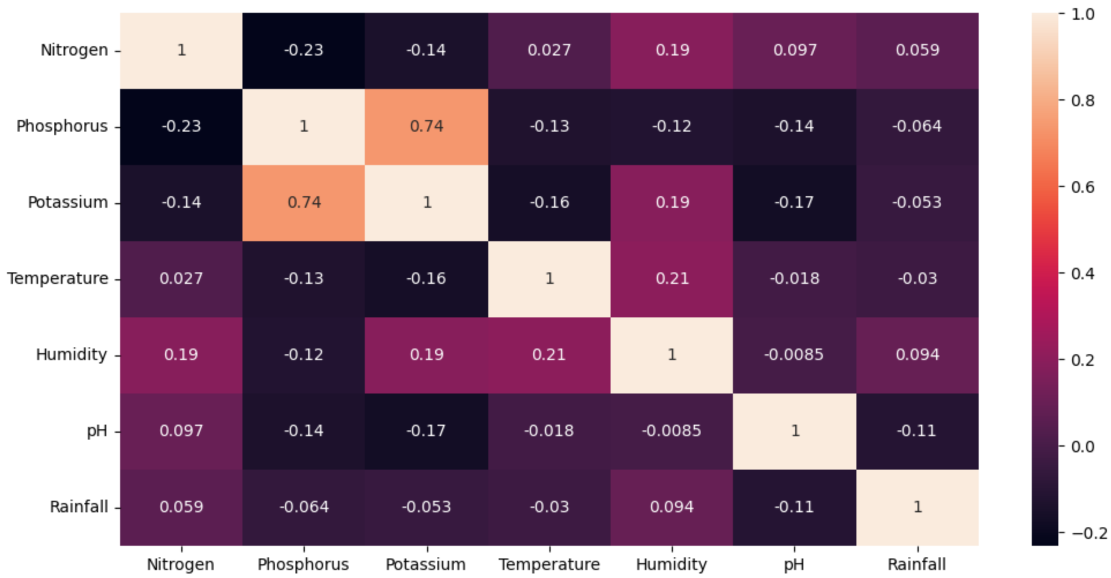
[19]:
```
sns.pairplot(data = df ,hue = 'Label')
plt.show()
```



1 HeatMap

[20]:
```
figure = plt.figure(figsize=(12, 6))
sns.heatmap(df.corr(),annot=True)
```

[20]: <Axes: >



As observed from our heat map Potassium and Phosphorus has high corelation value of 0.74

[21]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
df_pca = pca.fit_transform(df.drop(['Label'],axis =1))
df_pca = pd.DataFrame(df_pca)
fig = px.scatter(x = df_pca[0],y = df_pca[1],color = df['Label'],title =
  "Decomposed Using PCA")
fig.show()
```

[22]:
```
pca3=PCA(n_components=3)
df_pca3=pca3.fit_transform(df.drop(['Label'],axis=1))
df_pca3=pd.DataFrame(df_pca3)
fig = px.
  scatter_3d(x=df_pca3[0],y=df_pca3[1],z=df_pca3[2],color=df['Label'],title=f"Variance
  Explained : {pca3.explained_variance_ratio_.sum() * 100}%")
fig.show()
```

[23]:
```
fig = px.
  scatter(x=df['Nitrogen'],y=df['Phosphorus'],color=df['Label'],title="Nitrogen
  VS Phosphorus")
fig.show()
```

[24]:
```
fig = px.
  scatter(x=df['Phosphorus'],y=df['Potassium'],color=df['Label'],title="Phosphorus
  VS Potassium")
fig.show()
```

[25]:
```
names = df['Label'].unique()
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
df['Label']=encoder.fit_transform(df['Label'])
df.head()
```

[25]:
```
   Nitrogen  Phosphorus  Potassium  Temperature   Humidity        pH  \
0        90          42         43    20.879744  82.002744  6.502985
1        85          58         41    21.770462  80.319644  7.038096
2        60          55         44    23.004459  82.320763  7.840207
3        74          35         40    26.491096  80.158363  6.980401
4        78          42         42    20.130175  81.604873  7.628473

     Rainfall  Label
0  202.935536     20
1  226.655537     20
2  263.964248     20
3  242.864034     20
4  262.717340     20
```

[26]:
```
X=df.drop(['Label'],axis=1)
y=df['Label']
#Splitting into training and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
 ↪3,shuffle = True, random_state = 42,stratify=y)
```

[27]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_train=pd.DataFrame(X_train,columns=X.columns)
X_train.head()
```

[27]:
```
   Nitrogen  Phosphorus  Potassium  Temperature  Humidity        pH  Rainfall
0 -1.335936    0.417499  -0.535091     0.378274 -0.489416  0.105457 -1.006138
1  1.797538    0.874355  -0.061709    -0.056432  0.352421 -1.102431  0.037615
2 -1.308923    0.234757  -0.554816    -0.672000 -2.173304 -0.662710 -0.486121
3 -0.282441    0.752527  -0.554816    -1.248506 -2.271540 -1.031842 -0.422218
4 -1.173860   -1.013983  -0.712610    -1.765899  1.047107  0.007107  0.121738
```

[28]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
 ↪1,random_state=42)
```

[29]:
```
!pip install catboost
```

```
Collecting catboost
```

```
  Downloading catboost-1.2.1-cp310-cp310-manylinux2014_x86_64.whl (98.7 MB)
                         98.7/98.7 MB
9.1 MB/s eta 0:00:00
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-
packages (from catboost) (0.20.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-
packages (from catboost) (1.23.5)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-
packages (from catboost) (1.5.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from catboost) (1.10.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages
(from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=0.24->catboost) (2023.3.post1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.1)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.2.3)
Installing collected packages: catboost
Successfully installed catboost-1.2.1
```

[30]:
```
!pip install lightgbm
```

```
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-
packages (4.0.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from lightgbm) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from lightgbm) (1.10.1)
```

[ ]:
```
pip install --upgrade pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2022.7.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-
packages (from pandas) (1.22.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

1.1 Applying different models

[31]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.svm import NuSVC
from xgboost import XGBClassifier
#import lightgbm as lgb
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from catboost import CatBoostClassifier
```

[32]:
```
models = {'Logistic Regression': LogisticRegression(),
         'Random Forest': RandomForestClassifier(),
         'Tree': DecisionTreeClassifier(),
         "SVC": SVC(),
          "Linear SVC":LinearSVC(C=2),
          "NU SVC":NuSVC(),
         "XGBoost": XGBClassifier(),
         "KNN":KNeighborsClassifier(n_neighbors = 5, p=2),
          #"Light GBM": lgb.LGBMClassifier(),
          "LDA":LinearDiscriminantAnalysis(),
          "Gaussian NB":GaussianNB(),
          "AdaBoost":AdaBoostClassifier(),
          "Gradient Boosting":GradientBoostingClassifier(),
```

```
          "Bagging":BaggingClassifier(),
          "Extra Trees":ExtraTreesClassifier(),
          "Cat Boost":CatBoostClassifier(verbose=False)}

def fit_and_score(models,X_train,X_test,y_train,y_test):
    np.random.seed(42)
    model_scores = {}
    for name,model in models.items():
        model.fit(X_train,y_train)
        model_scores[name] = model.score(X_test,y_test)

    return model_scores
```

1.2 Compare accuracy of different models

[33]:
```
model_scores = fit_and_score(models,X_train,X_test,y_train,y_test)
model_scores
```

[33]:
```
{'Logistic Regression': 0.9636363636363636,
 'Random Forest': 1.0,
 'Tree': 0.9954545454545455,
 'SVC': 0.9727272727272728,
 'Linear SVC': 0.8272727272727273,
 'NU SVC': 0.9454545454545454,
 'XGBoost': 0.9954545454545455,
 'KNN': 0.9772727272727273,
 'LDA': 0.9363636363636364,
 'Gaussian NB': 1.0,
 'AdaBoost': 0.14545454545454545,
 'Gradient Boosting': 0.990909090909091,
 'Bagging': 0.9954545454545455,
 'Extra Trees': 0.9954545454545455,
 'Cat Boost': 0.9954545454545455}
```

[34]:
```
type(model_scores)
```

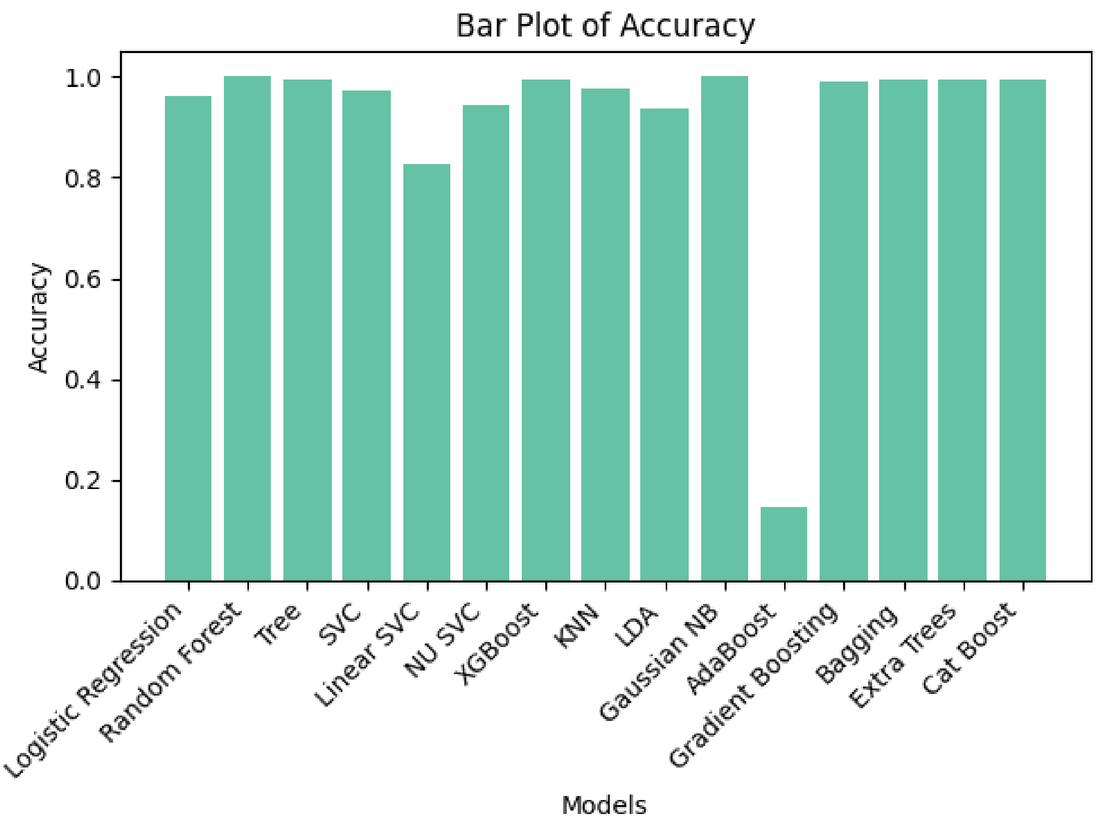
[34]:
```
dict
```

1.3 Analysing results of different models

[35]:
```
def plot_dict_as_bar(dict_data, title=None):
    keys = list(dict_data.keys())
    values = list(dict_data.values())

    plt.bar(keys, values)
    plt.xlabel('Models')
```

```
    plt.ylabel('Accuracy')
    if title:
        plt.title(title)
    # Rotate x-axis labels by 45 degrees for better alignment
    plt.xticks(rotation=45, ha='right')

    plt.tight_layout()  # Adjusts the layout to prevent overlapping
    plt.show()
```

[36]:
```
plot_dict_as_bar(model_scores,title = 'Bar Plot of Accuracy')
```

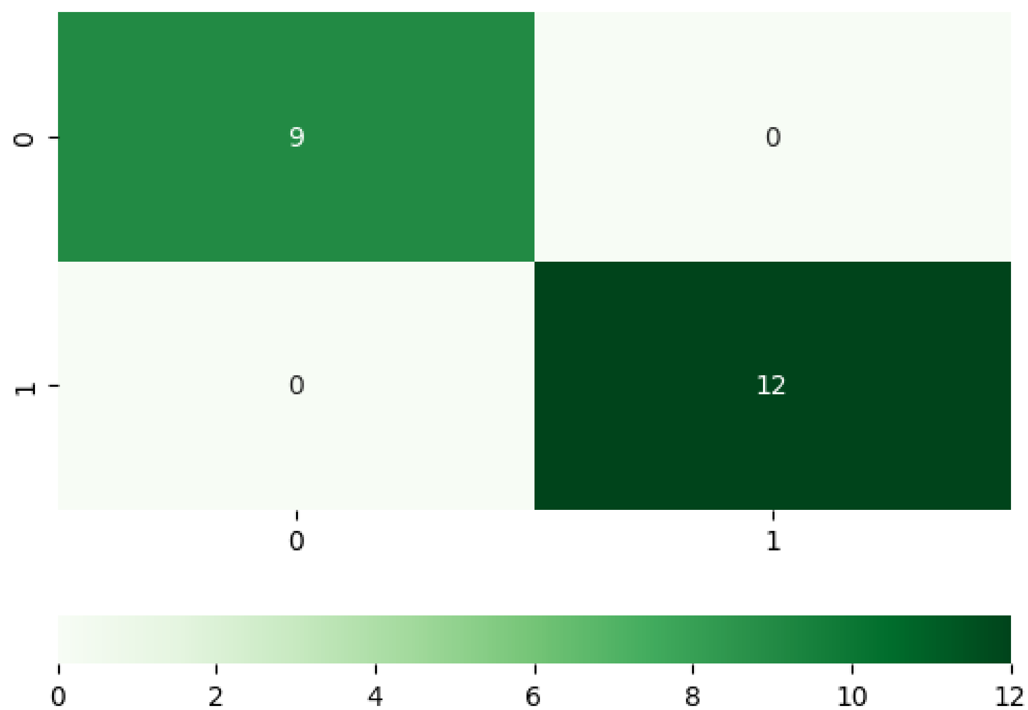


[37]:
```
def cm_and_score(models,X_train,X_test,y_train,y_test):
    np.random.seed(42)
    for name,model in models.items():
        print('**************   '+ name + '   ***********')
        y_pred = model.predict(X_test)

        Acc = accuracy_score(y_pred,y_test)
        cm = confusion_matrix(y_test,y_pred,labels = [0,1])
        print('Confusion Matrix')
```

```
        sns.heatmap(cm,cmap = 'Greens',annot = True,cbar_kws = {'orientation':
↪'horizontal'})
        plt.show()

        print(classification_report(y_test,y_pred))
        print('.:.'+ name +' Accuracy'+'\033[1m {:.3f}%'.format(Acc*100)+' .:.')
        print('      ')
        print('      ')
        print('      ')
```

[38]:
```
cm_and_score(models,X_train,X_test,y_train,y_test)
```

```
**************   Logistic Regression   ***********
Confusion Matrix
```



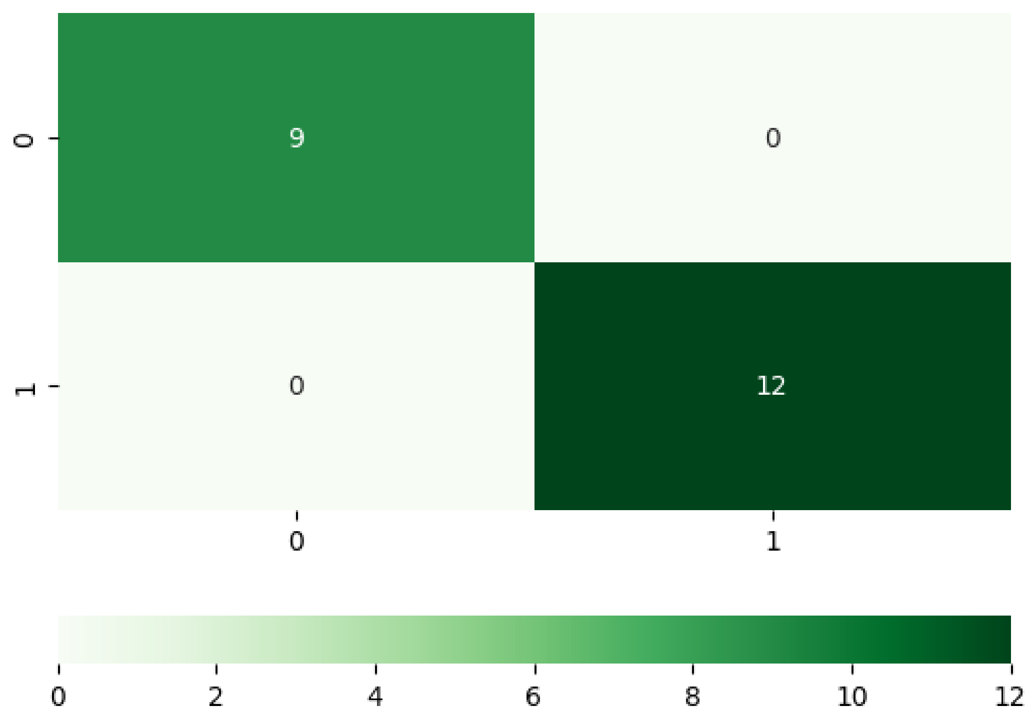

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       1.00      1.00      1.00        12
           2       0.85      0.85      0.85        13
           3       1.00      1.00      1.00        13
           4       1.00      1.00      1.00        15
           5       1.00      1.00      1.00         9
           6       0.86      1.00      0.92         6
```

7	1.00	1.00	1.00	8
8	0.85	1.00	0.92	11
9	1.00	1.00	1.00	13
10	1.00	1.00	1.00	7
11	1.00	0.83	0.91	12
12	1.00	1.00	1.00	4
13	0.83	0.91	0.87	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	0.92	1.00	0.96	12
18	1.00	0.83	0.91	12
19	1.00	1.00	1.00	10
20	1.00	0.88	0.93	8
21	1.00	1.00	1.00	9
accuracy			0.96	220
macro avg	0.97	0.97	0.97	220
weighted avg	0.97	0.96	0.96	220

.:.Logistic Regression Accuracy **96.364%** .:.

*****************   Random Forest   *************

Confusion Matrix

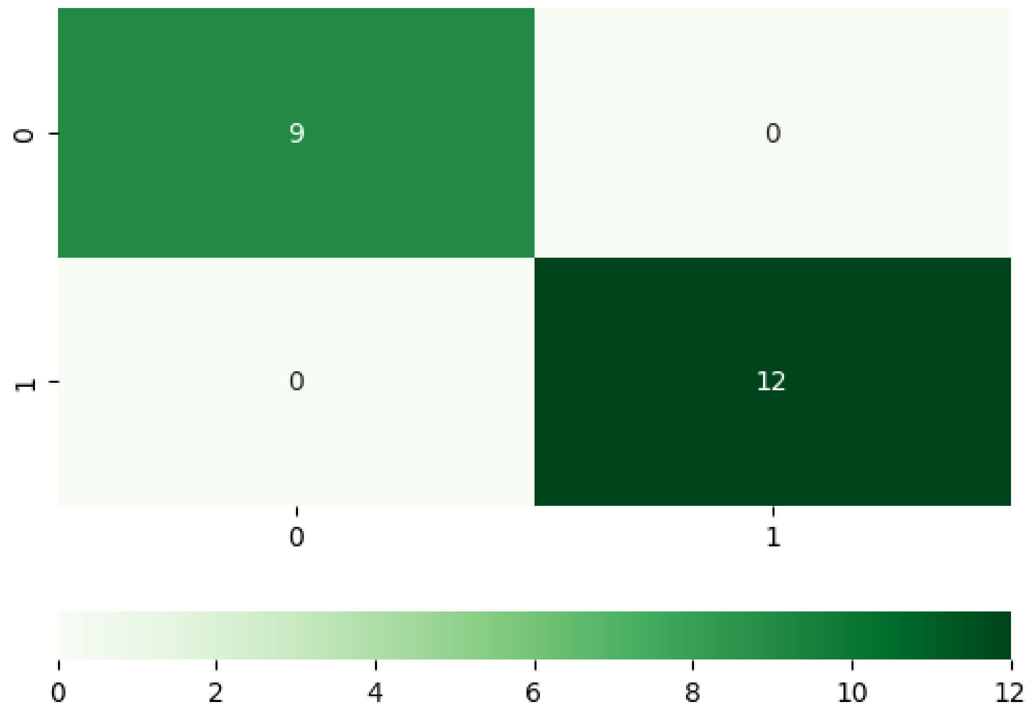


	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12
2	1.00	1.00	1.00	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	1.00	1.00	1.00	6
7	1.00	1.00	1.00	8
8	1.00	1.00	1.00	11
9	1.00	1.00	1.00	13
10	1.00	1.00	1.00	7
11	1.00	1.00	1.00	12
12	1.00	1.00	1.00	4
13	1.00	1.00	1.00	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	12
18	1.00	1.00	1.00	12
19	1.00	1.00	1.00	10
20	1.00	1.00	1.00	8
21	1.00	1.00	1.00	9
accuracy			1.00	220
macro avg	1.00	1.00	1.00	220
weighted avg	1.00	1.00	1.00	220

.:.Random Forest Accuracy **100.000%** .:.

**************   Tree   ***********

Confusion Matrix



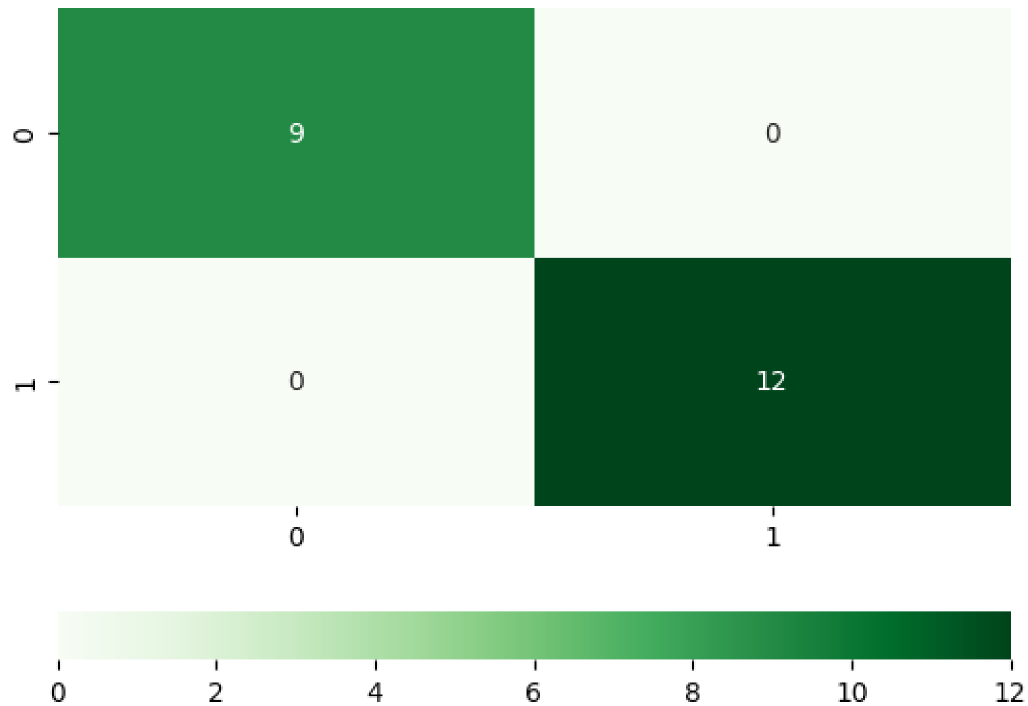
	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12
2	0.93	1.00	0.96	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	1.00	1.00	1.00	6
7	1.00	1.00	1.00	8
8	1.00	1.00	1.00	11
9	1.00	1.00	1.00	13
10	1.00	1.00	1.00	7
11	1.00	1.00	1.00	12
12	1.00	1.00	1.00	4
13	1.00	0.91	0.95	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	12
18	1.00	1.00	1.00	12
19	1.00	1.00	1.00	10
20	1.00	1.00	1.00	8
21	1.00	1.00	1.00	9

accuracy			1.00	220
macro avg	1.00	1.00	1.00	220
weighted avg	1.00	1.00	1.00	220

.:.Tree Accuracy 99.545% .:.

************** SVC ***********

Confusion Matrix



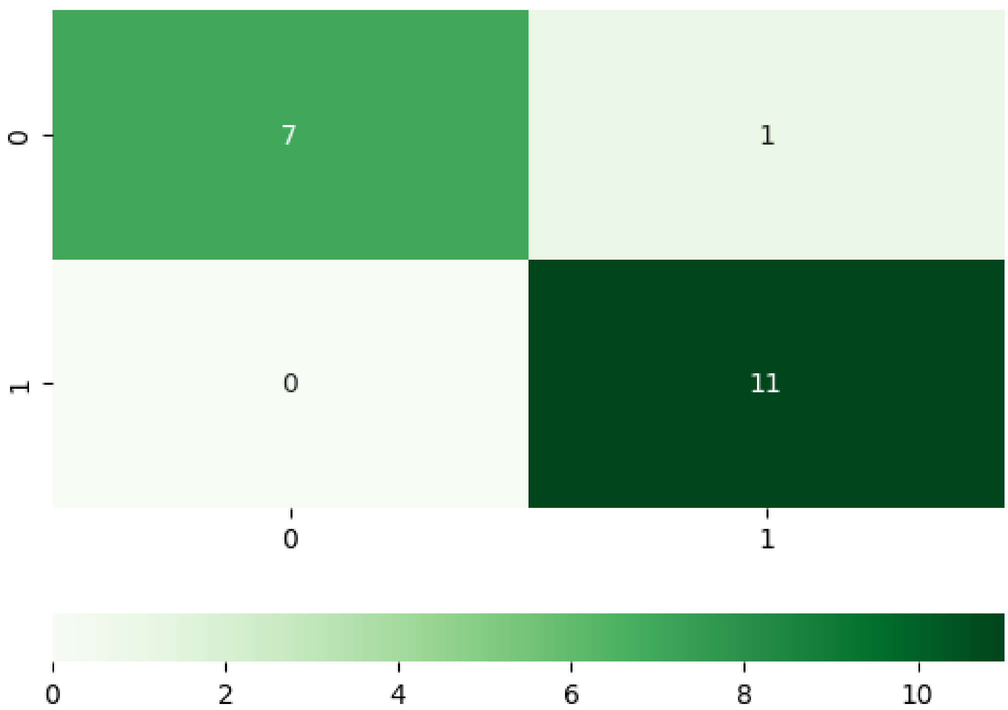
	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12
2	0.87	1.00	0.93	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	0.86	1.00	0.92	6
7	1.00	1.00	1.00	8
8	0.85	1.00	0.92	11
9	1.00	1.00	1.00	13
10	0.88	1.00	0.93	7
11	1.00	0.92	0.96	12

12	1.00	1.00	1.00	4
13	1.00	0.91	0.95	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	12
18	1.00	0.83	0.91	12
19	1.00	1.00	1.00	10
20	1.00	0.75	0.86	8
21	1.00	1.00	1.00	9
accuracy			0.97	220
macro avg	0.97	0.97	0.97	220
weighted avg	0.98	0.97	0.97	220

.:.SVC Accuracy 97.273% .:.

************** Linear SVC ***********

Confusion Matrix



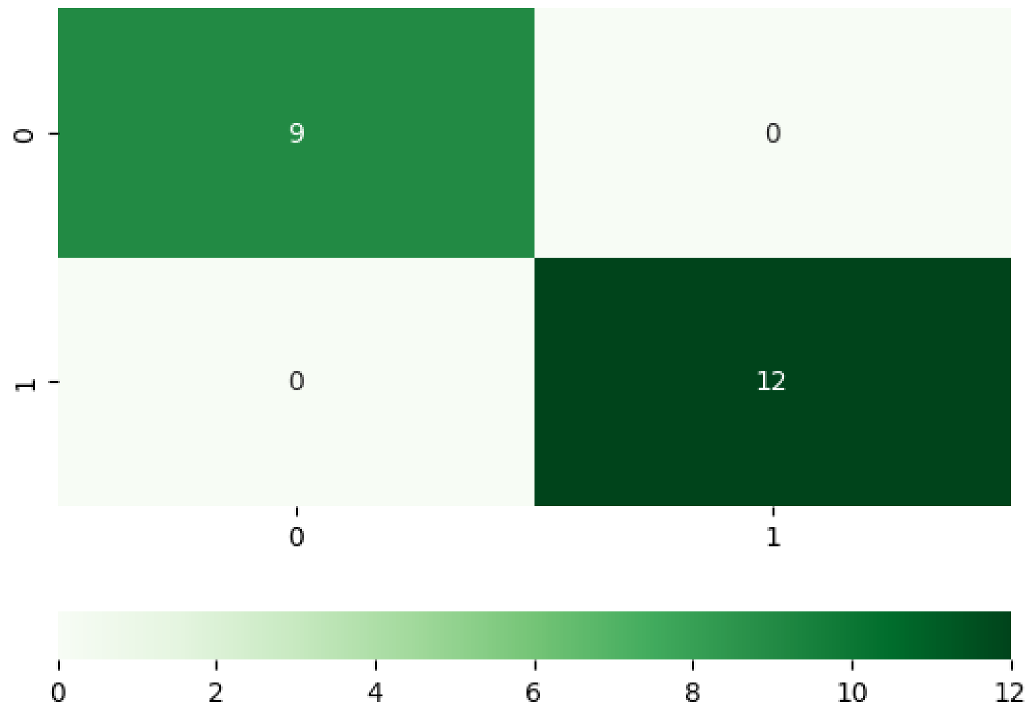
	precision	recall	f1-score	support
0	1.00	0.78	0.88	9
1	0.92	0.92	0.92	12

2	0.55	0.85	0.67	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	1.00	0.33	0.50	6
7	1.00	1.00	1.00	8
8	0.50	0.09	0.15	11
9	1.00	1.00	1.00	13
10	1.00	0.71	0.83	7
11	0.46	0.50	0.48	12
12	1.00	1.00	1.00	4
13	0.75	0.82	0.78	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	0.69	0.75	0.72	12
18	0.90	0.75	0.82	12
19	0.88	0.70	0.78	10
20	0.44	1.00	0.62	8
21	0.90	1.00	0.95	9
accuracy			0.83	220
macro avg	0.86	0.83	0.82	220
weighted avg	0.85	0.83	0.82	220

.:.Linear SVC Accuracy **82.727%** .:.

**************   **NU SVC**   ***********

Confusion Matrix



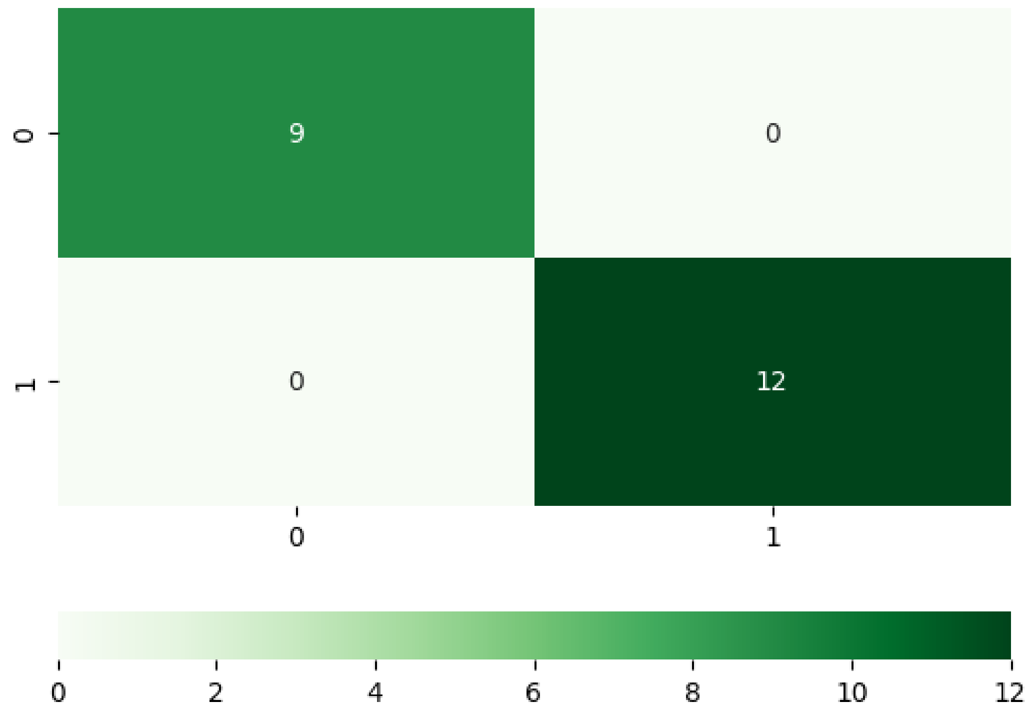
	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12
2	0.81	1.00	0.90	13
3	1.00	1.00	1.00	13
4	1.00	0.87	0.93	15
5	1.00	1.00	1.00	9
6	0.86	1.00	0.92	6
7	1.00	1.00	1.00	8
8	0.85	1.00	0.92	11
9	1.00	1.00	1.00	13
10	0.88	1.00	0.93	7
11	1.00	0.92	0.96	12
12	0.80	1.00	0.89	4
13	1.00	0.82	0.90	11
14	0.83	1.00	0.91	10
15	1.00	1.00	1.00	7
16	0.90	1.00	0.95	9
17	1.00	0.83	0.91	12
18	1.00	0.75	0.86	12
19	0.91	1.00	0.95	10
20	1.00	0.75	0.86	8
21	1.00	1.00	1.00	9

	precision	recall	f1-score	support
accuracy			0.95	220
macro avg	0.95	0.95	0.94	220
weighted avg	0.95	0.95	0.94	220

.:.NU SVC Accuracy **94.545%** .:.

****************   XGBoost   *************

Confusion Matrix



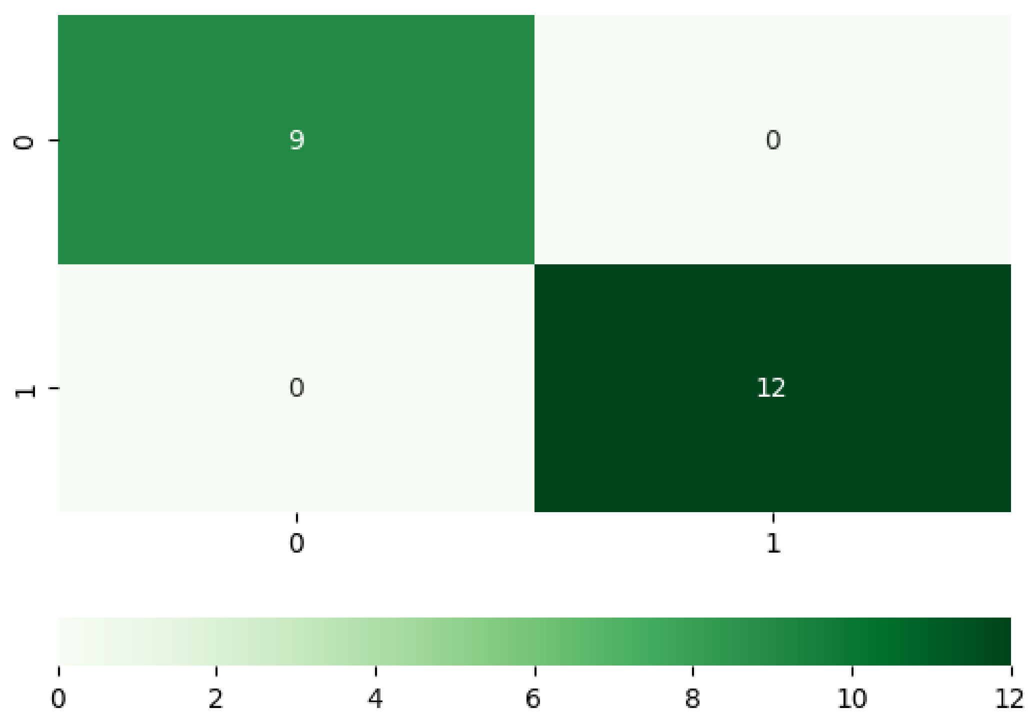
	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12
2	0.93	1.00	0.96	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	1.00	1.00	1.00	6
7	1.00	1.00	1.00	8
8	1.00	1.00	1.00	11
9	1.00	1.00	1.00	13
10	1.00	1.00	1.00	7
11	1.00	1.00	1.00	12

12	1.00	1.00	1.00	4
13	1.00	1.00	1.00	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	12
18	1.00	0.92	0.96	12
19	1.00	1.00	1.00	10
20	1.00	1.00	1.00	8
21	1.00	1.00	1.00	9
accuracy			1.00	220
macro avg	1.00	1.00	1.00	220
weighted avg	1.00	1.00	1.00	220

.:.XGBoost Accuracy **99.545%** .:.

************** KNN ***********

Confusion Matrix



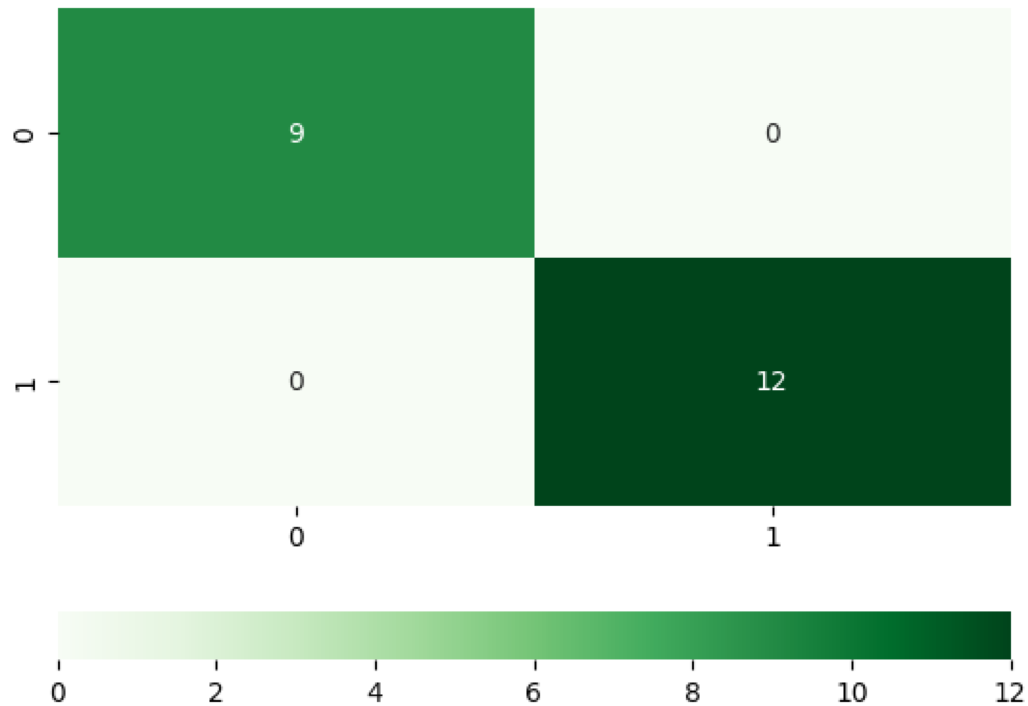
	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12

2	0.93	1.00	0.96	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	0.86	1.00	0.92	6
7	1.00	1.00	1.00	8
8	0.79	1.00	0.88	11
9	1.00	1.00	1.00	13
10	1.00	1.00	1.00	7
11	1.00	0.92	0.96	12
12	1.00	1.00	1.00	4
13	1.00	1.00	1.00	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	12
18	1.00	0.92	0.96	12
19	1.00	1.00	1.00	10
20	1.00	0.62	0.77	8
21	1.00	1.00	1.00	9
accuracy			0.98	220
macro avg	0.98	0.98	0.97	220
weighted avg	0.98	0.98	0.98	220

.:.KNN Accuracy **97.727%** .:.

**************   **LDA**   ***********

Confusion Matrix



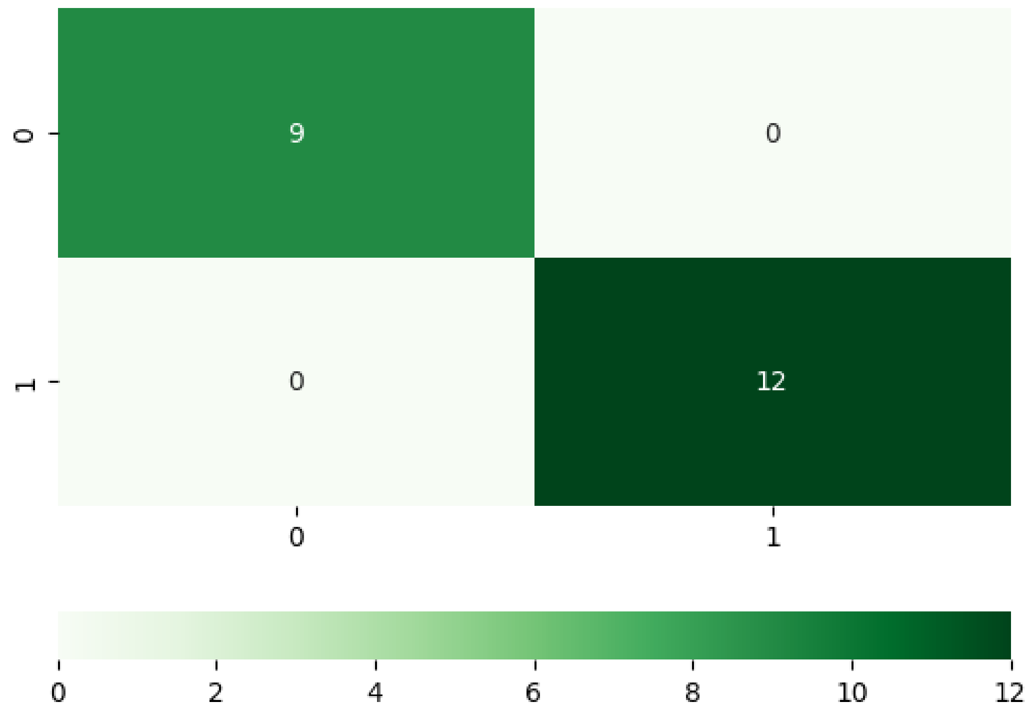
	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12
2	0.69	0.85	0.76	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	0.86	1.00	0.92	6
7	1.00	1.00	1.00	8
8	0.79	1.00	0.88	11
9	1.00	1.00	1.00	13
10	0.60	0.86	0.71	7
11	1.00	0.92	0.96	12
12	0.80	1.00	0.89	4
13	1.00	0.82	0.90	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	12
18	1.00	0.58	0.74	12
19	1.00	1.00	1.00	10
20	1.00	0.62	0.77	8
21	1.00	1.00	1.00	9

	precision	recall	f1-score	support
accuracy			0.94	220
macro avg	0.94	0.94	0.93	220
weighted avg	0.95	0.94	0.94	220

.:.LDA Accuracy 93.636% .:.

************** Gaussian NB ***********

Confusion Matrix



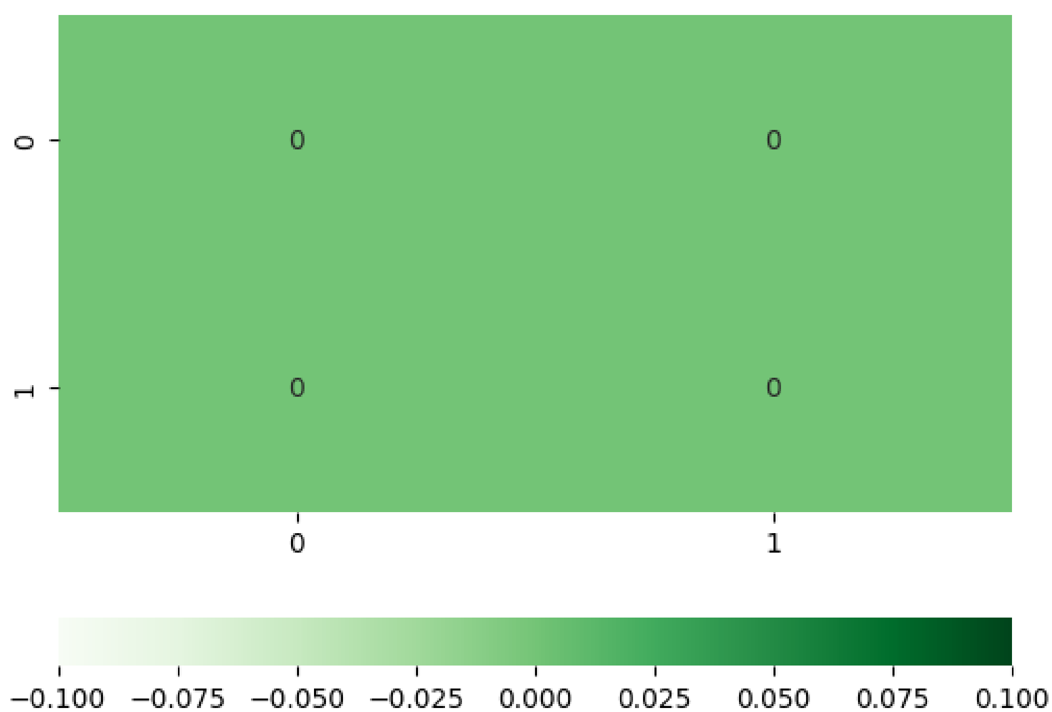
	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12
2	1.00	1.00	1.00	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	1.00	1.00	1.00	6
7	1.00	1.00	1.00	8
8	1.00	1.00	1.00	11
9	1.00	1.00	1.00	13
10	1.00	1.00	1.00	7
11	1.00	1.00	1.00	12

12	1.00	1.00	1.00	4
13	1.00	1.00	1.00	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	12
18	1.00	1.00	1.00	12
19	1.00	1.00	1.00	10
20	1.00	1.00	1.00	8
21	1.00	1.00	1.00	9
accuracy			1.00	220
macro avg	1.00	1.00	1.00	220
weighted avg	1.00	1.00	1.00	220

.:.Gaussian NB Accuracy **100.000%** .:.

************** **AdaBoost** ***********

Confusion Matrix



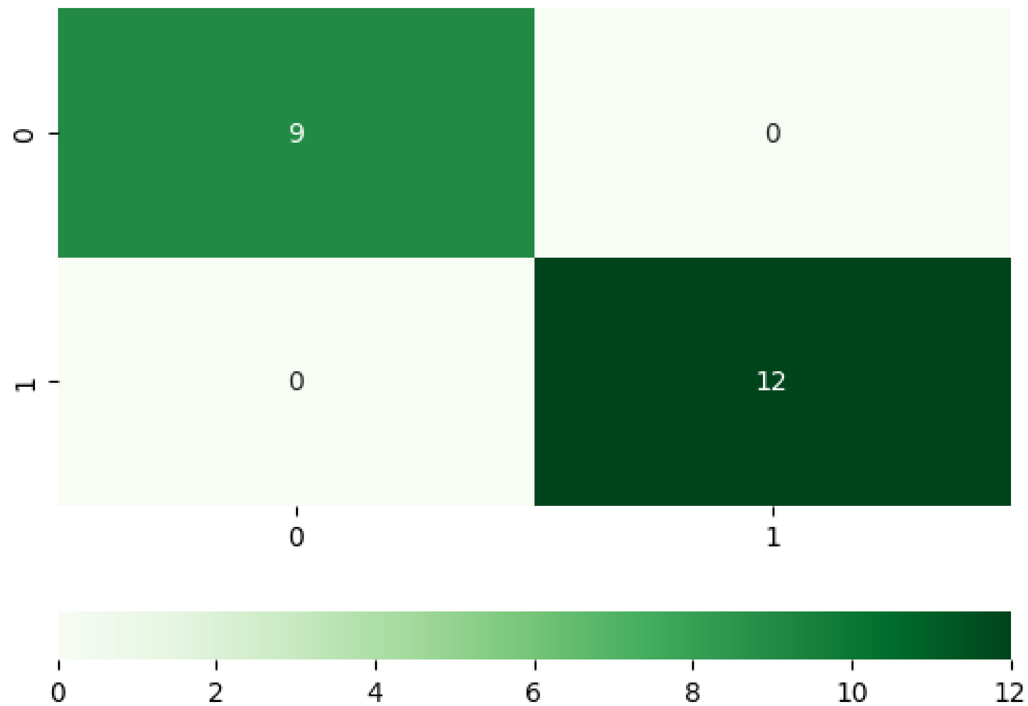
	precision	recall	f1-score	support
0	0.00	0.00	0.00	9
1	0.00	0.00	0.00	12

2	0.00	0.00	0.00	13
3	0.00	0.00	0.00	13
4	0.00	0.00	0.00	15
5	0.00	0.00	0.00	9
6	0.00	0.00	0.00	6
7	0.47	1.00	0.64	8
8	0.00	0.00	0.00	11
9	0.50	1.00	0.67	13
10	0.00	0.00	0.00	7
11	0.00	0.00	0.00	12
12	0.02	1.00	0.05	4
13	0.00	0.00	0.00	11
14	0.00	0.00	0.00	10
15	1.00	1.00	1.00	7
16	0.00	0.00	0.00	9
17	0.00	0.00	0.00	12
18	0.00	0.00	0.00	12
19	0.00	0.00	0.00	10
20	0.00	0.00	0.00	8
21	0.00	0.00	0.00	9
accuracy			0.15	220
macro avg	0.09	0.18	0.11	220
weighted avg	0.08	0.15	0.10	220

.:.AdaBoost Accuracy **14.545%** .:.

************** **Gradient Boosting** ***********

Confusion Matrix



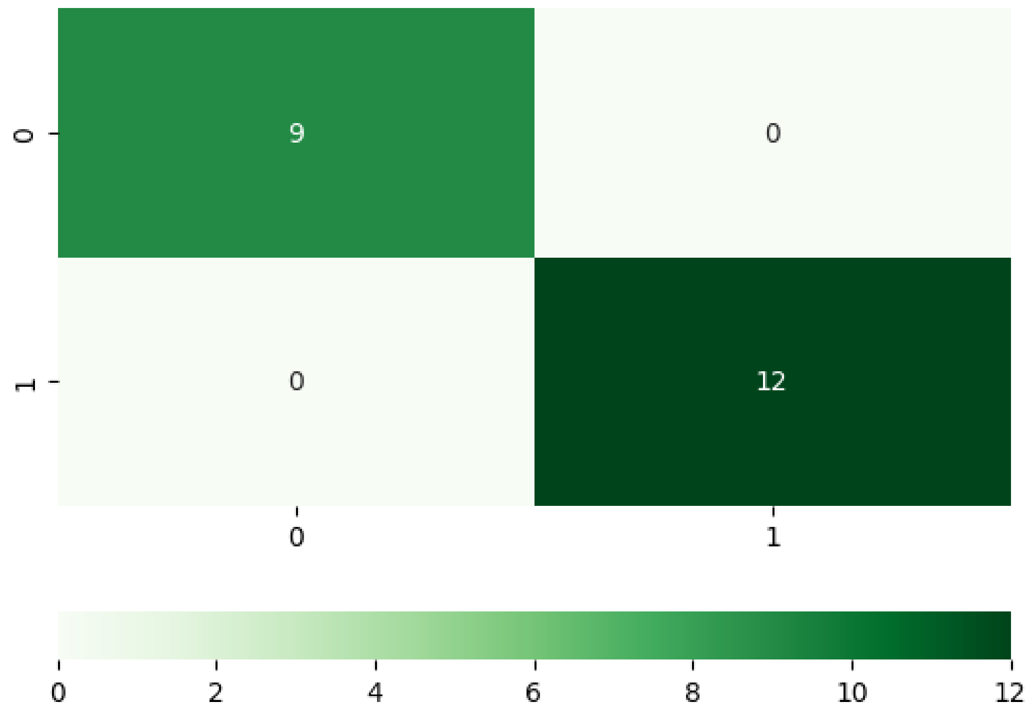
	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12
2	0.93	1.00	0.96	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	1.00	1.00	1.00	6
7	1.00	1.00	1.00	8
8	0.92	1.00	0.96	11
9	1.00	1.00	1.00	13
10	1.00	1.00	1.00	7
11	1.00	1.00	1.00	12
12	1.00	1.00	1.00	4
13	1.00	1.00	1.00	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	12
18	1.00	0.92	0.96	12
19	1.00	1.00	1.00	10
20	1.00	0.88	0.93	8
21	1.00	1.00	1.00	9

accuracy			0.99	220
macro avg	0.99	0.99	0.99	220
weighted avg	0.99	0.99	0.99	220

.:.Gradient Boosting Accuracy 99.091% .:.

************** Bagging ***********

Confusion Matrix



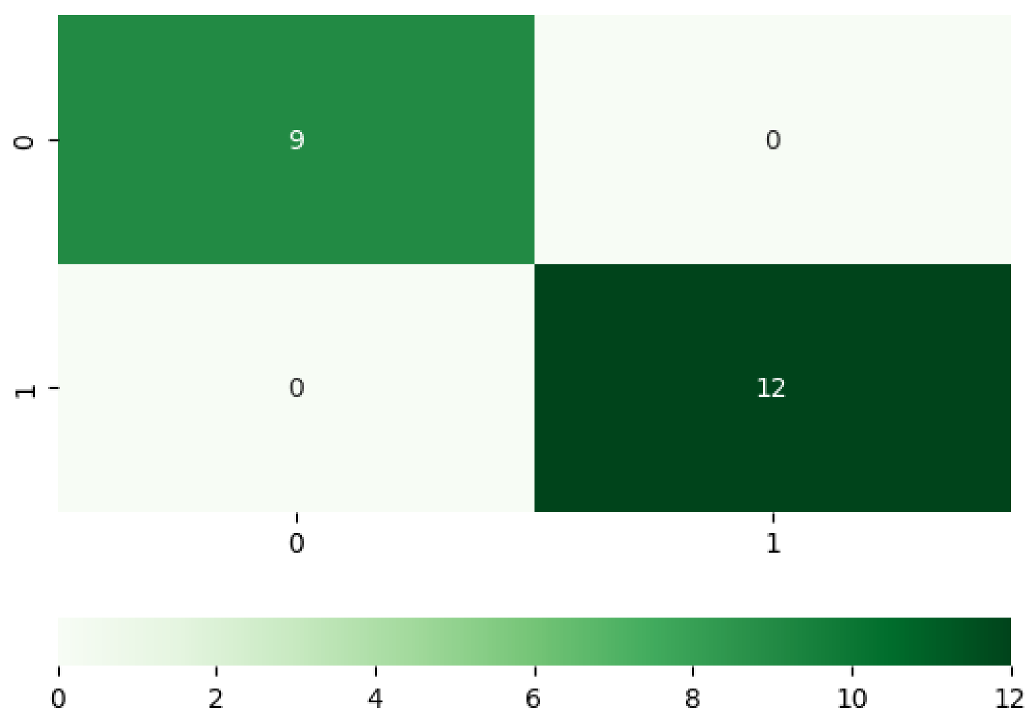
	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12
2	0.93	1.00	0.96	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	1.00	1.00	1.00	6
7	1.00	1.00	1.00	8
8	1.00	1.00	1.00	11
9	1.00	1.00	1.00	13
10	1.00	1.00	1.00	7
11	1.00	1.00	1.00	12

	precision	recall	f1-score	support
12	1.00	1.00	1.00	4
13	1.00	0.91	0.95	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	12
18	1.00	1.00	1.00	12
19	1.00	1.00	1.00	10
20	1.00	1.00	1.00	8
21	1.00	1.00	1.00	9
accuracy			1.00	220
macro avg	1.00	1.00	1.00	220
weighted avg	1.00	1.00	1.00	220

.:.Bagging Accuracy 99.545% .:.

**************   **Extra Trees**   ***********

Confusion Matrix



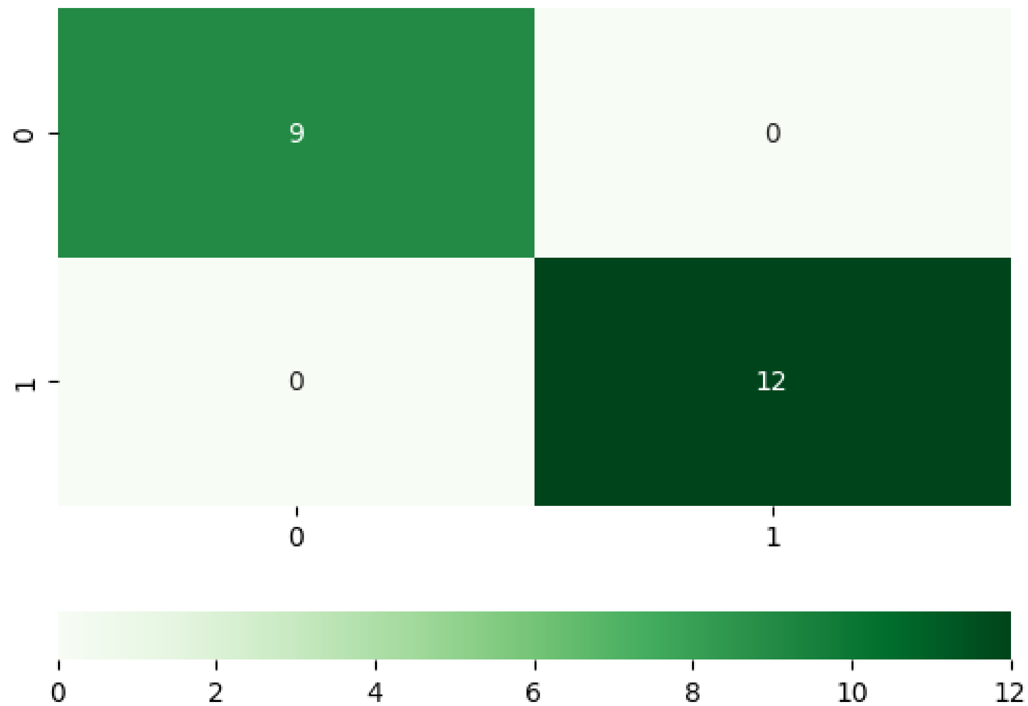
	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12

2	1.00	1.00	1.00	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	1.00	1.00	1.00	6
7	1.00	1.00	1.00	8
8	1.00	1.00	1.00	11
9	1.00	1.00	1.00	13
10	0.88	1.00	0.93	7
11	1.00	1.00	1.00	12
12	1.00	1.00	1.00	4
13	1.00	0.91	0.95	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	12
18	1.00	1.00	1.00	12
19	1.00	1.00	1.00	10
20	1.00	1.00	1.00	8
21	1.00	1.00	1.00	9
accuracy			1.00	220
macro avg	0.99	1.00	0.99	220
weighted avg	1.00	1.00	1.00	220

.:.Extra Trees Accuracy **99.545%** .:.

************** **Cat Boost** ***********

Confusion Matrix



	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	1.00	1.00	12
2	0.93	1.00	0.96	13
3	1.00	1.00	1.00	13
4	1.00	1.00	1.00	15
5	1.00	1.00	1.00	9
6	1.00	1.00	1.00	6
7	1.00	1.00	1.00	8
8	1.00	1.00	1.00	11
9	1.00	1.00	1.00	13
10	1.00	1.00	1.00	7
11	1.00	1.00	1.00	12
12	1.00	1.00	1.00	4
13	1.00	1.00	1.00	11
14	1.00	1.00	1.00	10
15	1.00	1.00	1.00	7
16	1.00	1.00	1.00	9
17	1.00	1.00	1.00	12
18	1.00	0.92	0.96	12
19	1.00	1.00	1.00	10
20	1.00	1.00	1.00	8
21	1.00	1.00	1.00	9

```
    accuracy                           1.00       220
   macro avg       1.00      1.00      1.00       220
weighted avg       1.00      1.00      1.00       220

.:.Cat Boost Accuracy 99.545% .:.
```

[ ]: