## Name : Kshitij V Darwhekar Roll No: TETB19

Sub: Soft Computitng Batch: B2

## Experiment 6: Implementation of IOT Solution using Machine Learning

Importing the libraries

import sklearn import numpy as np import pandas as pd

## Importing the dataset

from google.colab import drive drive.mount('/content/drive/')

Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive

dataset = pd.read\_csv("/content/drive/MyDrive/ML/Crop\_recommendation.csv")

X = dataset.iloc[:, :-1].values y = dataset.iloc[:, -1].values

dataset.head()

**N P K temperature humidity ph rainfall label**

Data Preprocessing

**2** 60 55 44

23.004459 82.320763 7.840207 263.964248

rice

**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

Taking care of missing data

from sklearn.impute import SimpleImputer

**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

imputer = SimpleImputer(missing\_values=np.nan, strategy='mean') imputer.fit(X[:,:])

X[:, :] = imputer.transform(X[:, :])

## Encoding categorical data

from sklearn.preprocessing import LabelEncoder le = LabelEncoder()

y = le.fit\_transform(y)

## Splitting the dataset into the Training set 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 = 0.25, random\_state =

print(X\_train)

[[134. 56. 18. ... 83.91902605 6.6912681

70.97358303]

[ 29. 122. 196. ... 81.15595212 5.63832848

73.06862952]

[ 25. 68. 19. ... 64.25510719 7.10845012

67.47677295]

...

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [ | 35. |  | 64. | 15. ... 63.53604453 | 6.50014496 |
|  | 69.5274407 | ] |  |  |  |
| [ | 39. |  | 65. | 23. ... 69.12613376 | 7.6859593 |

41.02682925]

[ 14. 22. 9. ... 91.13772765 6.54319181

112.5090516 ]]

print(y\_train)

[ 6 7 2 ... 2 10 16]

print(X\_test)

[[105. 14. 50. ... 87.6883982 6.41905219

59.65590798]

[ 91. 12. 46. ... 85.49938185 6.34394252

48.31219031]

[ 14. 121. 203. ... 83.74765639 6.15868941

74.46411148]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ...  [ 84. |  | 27. | 29. ... 53.00366334 | 7.16709259 |
| 168.2644287 | ] |  |  |  |
| [ 31. |  | 13. | 33. ... 95.21224392 | 6.34246371 |
| 148.3003692 | ] |  |  |  |
| [ 5. |  | 24. | 40. ... 93.87030088 | 6.29790758 |
| 104.6735454 | ]] |  |  |  |

print(y\_test)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [21 | 21 | 7 | 3 | 2 | 20 | 13 | 9 | 15 | 1 | 13 | 5 | 10 | 14 | 12 | 0 | 5 | 10 | 5 | 12 | 4 | 2 | 9 | 8 |
| 6 | 5 | 10 | 16 | 13 | 9 | 19 | 20 | 11 | 15 | 4 | 6 | 12 | 12 | 21 | 13 | 11 | 2 | 18 | 21 | 18 | 14 | 9 | 9 |
| 6 | 14 | 13 | 2 | 0 | 15 | 18 | 1 | 17 | 12 | 10 | 6 | 16 | 14 | 21 | 20 | 15 | 0 | 7 | 5 | 0 | 16 | 4 | 19 |
| 9 | 11 | 7 | 13 | 3 | 11 | 8 | 12 | 20 | 2 | 21 | 21 | 15 | 6 | 11 | 10 | 13 | 17 | 2 | 8 | 14 | 7 | 14 | 11 |
| 5 | 8 | 10 | 3 | 16 | 8 | 14 | 1 | 1 | 20 | 21 | 5 | 18 | 15 | 15 | 12 | 5 | 7 | 16 | 19 | 14 | 10 | 11 | 8 |
| 19 | 10 | 16 | 3 | 3 | 2 | 19 | 16 | 3 | 17 | 13 | 13 | 15 | 14 | 11 | 14 | 4 | 19 | 16 | 2 | 2 | 7 | 0 | 5 |
| 3 | 0 | 8 | 12 | 21 | 17 | 16 | 4 | 13 | 1 | 19 | 3 | 21 | 2 | 0 | 8 | 10 | 18 | 8 | 9 | 9 | 15 | 20 | 15 |
| 1 | 16 | 18 | 0 | 13 | 4 | 6 | 14 | 9 | 19 | 17 | 16 | 20 | 17 | 17 | 18 | 9 | 1 | 4 | 18 | 20 | 17 | 11 | 8 |
| 13 | 20 | 11 | 5 | 18 | 4 | 3 | 12 | 4 | 19 | 11 | 13 | 13 | 16 | 15 | 11 | 18 | 1 | 3 | 2 | 18 | 16 | 13 | 14 |
| 12 | 17 | 15 | 19 | 20 | 20 | 2 | 17 | 2 | 5 | 11 | 5 | 16 | 20 | 13 | 14 | 16 | 9 | 19 | 4 | 12 | 14 | 6 | 20 |
| 3 | 14 | 0 | 18 | 2 | 20 | 21 | 2 | 19 | 16 | 11 | 7 | 3 | 18 | 8 | 17 | 19 | 5 | 12 | 13 | 8 | 21 | 19 | 20 |
| 7 | 4 | 8 | 10 | 3 | 5 | 5 | 17 | 19 | 11 | 20 | 3 | 18 | 16 | 19 | 18 | 4 | 9 | 19 | 15 | 13 | 12 | 10 | 1 |
| 2 | 12 | 9 | 12 | 6 | 14 | 17 | 7 | 7 | 18 | 17 | 8 | 20 | 3 | 15 | 5 | 21 | 20 | 8 | 17 | 7 | 15 | 2 | 13 |
| 13 | 3 | 2 | 12 | 1 | 12 | 19 | 8 | 16 | 15 | 3 | 10 | 6 | 17 | 7 | 9 | 10 | 0 | 20 | 15 | 0 | 17 | 2 | 8 |
| 3 | 13 | 10 | 7 | 8 | 9 | 15 | 17 | 7 | 17 | 20 | 5 | 15 | 13 | 1 | 17 | 16 | 9 | 21 | 18 | 0 | 21 | 21 | 18 |
| 9 | 13 | 9 | 8 | 4 | 6 | 9 | 16 | 6 | 18 | 19 | 6 | 6 | 0 | 6 | 0 | 16 | 11 | 7 | 1 | 0 | 13 | 20 | 9 |
| 1 | 20 | 10 | 3 | 19 | 1 | 3 | 15 | 19 | 0 | 10 | 15 | 16 | 2 | 15 | 13 | 12 | 3 | 19 | 12 | 3 | 4 | 15 | 1 |
| 18 | 17 | 8 | 10 | 6 | 20 | 1 | 4 | 20 | 2 | 11 | 16 | 21 | 20 | 0 | 7 | 18 | 7 | 3 | 12 | 8 | 19 | 11 | 12 |
| 7 | 1 | 14 | 18 | 1 | 6 | 2 | 0 | 0 | 8 | 8 | 21 | 3 | 1 | 19 | 1 | 9 | 7 | 11 | 5 | 11 | 8 | 7 | 5 |
| 14 | 2 | 8 | 16 | 18 | 18 | 15 | 13 | 21 | 14 | 21 | 17 | 14 | 14 | 14 | 19 | 16 | 13 | 0 | 5 | 4 | 11 | 4 | 7 |
| 7 | 3 | 3 | 12 | 9 | 17 | 16 | 14 | 17 | 18 | 2 | 17 | 15 | 2 | 1 | 20 | 5 | 6 | 7 | 8 | 3 | 15 | 1 | 7 |
| 21 | 15 | 18 | 8 | 18 | 6 | 21 | 19 | 5 | 4 | 11 | 20 | 14 | 9 | 21 | 14 | 0 | 0 | 21 | 1 | 13 | 14 | 0 | 14 |
| 6 | 20 | 17 | 6 | 17 | 3 | 0 | 19 | 13 | 20 | 2 | 12 | 16 | 8 | 1 | 17 | 5 | 6 | 12 | 5 | 4 | 19] |  |  |

## Feature Scaling

from sklearn.preprocessing import StandardScaler sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train) X\_test = sc.transform(X\_test)

print(X\_train)

[[ 2.25367108 0.07555744 -0.59141091 ... 0.56115786 0.28639844

-0.58838147]

[-0.58434455 2.06834149 2.90385791 ... 0.43651791 -1.09903674

-0.55053196]

[-0.69245943 0.43788181 -0.57177457 ... -0.3258651 0.83531751

-0.65155552]

...

[-0.42217223 0.31710702 -0.65031993 ... -0.35830141 0.03492274

-0.61450776]

[-0.31405735 0.34730072 -0.4932292 ... -0.10613716 1.5951916

-1.12940532]

[-0.98977536 -0.95102828 -0.76813798 ... 0.88678747 0.09156286

0.16200634]]

print(X\_test)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [[ 1.46983819 -1.19257786 | 0.03695202 | ... | 0.73119109 | -0.07177737 |
| -0.79284878] |  |  |  |  |
| [ 1.09143611 -1.25296526 | -0.04159334 | ... | 0.63244639 | -0.17060505 |
| -0.99778658] |  |  |  |  |
| [-0.98977536 2.03814779 | 3.0413123 | ... | 0.55342752 | -0.41435709 |
| -0.52532091]  ...  [ 0.90223507 -0.80005979 | -0.37541115 | ... | -0.83340833 | 0.91247799 |
| 1.16929376] |  |  |  |  |
| [-0.53028711 -1.22277156 | -0.29686578 | ... | 1.07058549 | -0.17255083 |
| 0.8086192 ] |  |  |  |  |
| [-1.23303384 -0.89064089 | -0.15941139 | ... | 1.01005156 | -0.23117683 |
| 0.02044856]] |  |  |  |  |

# Random Forest

## Training the Random Forest Classification model on the Training set

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy', random\_state classifier.fit(X\_train, y\_train)

RandomForestClassifier(criterion='entropy', n\_estimators=10, random\_state=0)

## Predicting the Test set results

y\_pred\_RF = classifier.predict(X\_test) print(np.concatenate((y\_pred\_RF.reshape(len(y\_pred\_RF),1), y\_test.reshape(len(y\_test),1)),

|  |  |
| --- | --- |
| [[21 | 21] |
| [21 | 21] |
| [ 7 | 7] |
| ...  [ 5 | 5] |

[ 4 4]

[19 19]]

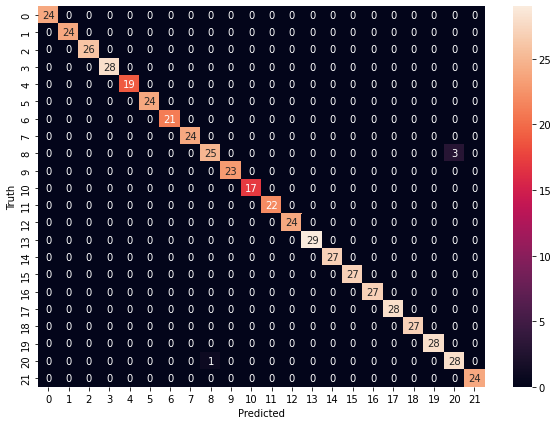
Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix, accuracy\_score cm = confusion\_matrix(y\_test, y\_pred\_RF)

%matplotlib inline

import matplotlib.pyplot as plt import seaborn as sn plt.figure(figsize=(10,7)) sn.heatmap(cm, annot=True) plt.xlabel('Predicted') plt.ylabel('Truth')

Text(69.0, 0.5, 'Truth')



accuracy\_score(y\_test, y\_pred\_RF) 0.9927272727272727

# Naive Bayes

## Training the Naive Bayes model on the Training set

from sklearn.naive\_bayes import GaussianNB classifier = GaussianNB() classifier.fit(X\_train, y\_train)

GaussianNB()

y\_pred\_NV = classifier.predict(X\_test) print(np.concatenate((y\_pred\_NV.reshape(len(y\_pred\_NV),1), y\_test.reshape(len(y\_test),1)),

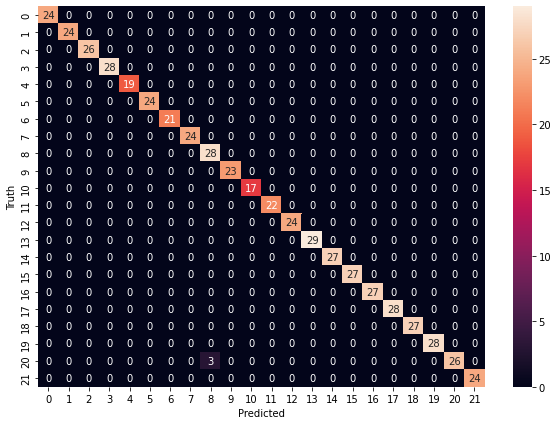
|  |  |
| --- | --- |
| [[21 | 21] |
| [21 | 21] |
| [ 7  ... | 7] |
| [ 5 | 5] |
| [ 4 | 4] |
| [19 | 19]] |

from sklearn.metrics import confusion\_matrix, accuracy\_score cm = confusion\_matrix(y\_test, y\_pred\_NV)

%matplotlib inline

import matplotlib.pyplot as plt import seaborn as sn plt.figure(figsize=(10,7)) sn.heatmap(cm, annot=True) plt.xlabel('Predicted') plt.ylabel('Truth')

Text(69.0, 0.5, 'Truth')



accuracy\_score(y\_test, y\_pred\_NV) 0.9945454545454545

 0s completed at 2:22 PM

Poster:

