Supervised Machine Learning

Sai kumar Murarishetti

Week 7 : Final Project Part 2

(https:/https://www.kaggle.com/datasets/asishpandey/crop-production-in-india/data//)

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from google.colab import files
```

```
# This is used to upload the dataset
df=files.upload()
data = pd.read_csv('Crop_production.csv')
```

Choose Files Crop_production.csv

- **Crop_production.csv**(text/csv) - 9887752 bytes, last modified: 10/8/2023 - 100% done

Saving Crop_production.csv to Crop_production.csv

```
# This is used to display first 10 lines from the dataset
print(data.head(10))
```

```
   Unnamed: 0      State_Name Crop_Type       Crop    N   P   K    pH  \
0           0  andhra pradesh    kharif     cotton  120  40  20  5.46
1           1  andhra pradesh    kharif  horsegram   20  60  20  6.18
2           2  andhra pradesh    kharif      jowar   80  40  40  5.42
3           3  andhra pradesh    kharif      maize   80  40  20  5.62
4           4  andhra pradesh    kharif      moong   20  40  20  5.68
5           5  andhra pradesh    kharif       ragi   50  40  20  5.64
6           6  andhra pradesh    kharif       rice   80  40  40  5.54
7           7  andhra pradesh    kharif  sunflower   50  60  30  5.36
8           8  andhra pradesh      rabi  horsegram   20  60  20  6.00
9           9  andhra pradesh      rabi      jowar   80  40  40  5.50

   rainfall  temperature  Area_in_hectares  Production_in_tons  \
0    654.34    29.266667            7300.0              9400.0
1    654.34    29.266667            3300.0              1000.0
2    654.34    29.266667           10100.0             10200.0
3    654.34    29.266667            2800.0              4900.0
4    654.34    29.266667            1300.0               500.0
5    654.34    29.266667            6700.0             11800.0
6    654.34    29.266667           35600.0             75400.0
7    654.34    29.266667           35900.0             11100.0
8    288.30    25.460000             600.0               200.0
9    288.30    25.460000           18800.0              9400.0

   Yield_ton_per_hec
0           1.287671
1           0.303030
2           1.009901
3           1.750000
4           0.384615
5           1.761194
6           2.117978
7           0.309192
8           0.333333
9           0.500000
```

```
# This is used to Display a summary of DataFrame 'df' for data inspection and understanding.
print(data.info)
```

```
<bound method DataFrame.info of        Unnamed: 0      State_Name Crop_Type       Crop    N   P   K    pH  \
0               0  andhra pradesh    kharif     cotton  120  40  20  5.46
1               1  andhra pradesh    kharif  horsegram   20  60  20  6.18
2               2  andhra pradesh    kharif      jowar   80  40  40  5.42
```

```
3              3  andhra pradesh    kharif      maize   80  40  20  5.62
4              4  andhra pradesh    kharif      moong   20  40  20  5.68
...          ...             ...       ...        ...  ...  ..  ..   ...
99844      99844     west bengal      rabi      wheat   60  30  30  6.70
99845      99845     west bengal    summer      maize   80  40  20  5.68
99846      99846     west bengal    summer       rice   80  40  40  5.64
99847      99847     west bengal      rabi       rice   80  40  40  5.42
99848      99848     west bengal      rabi    sesamum   30  15  30  6.54

       rainfall  temperature  Area_in_hectares  Production_in_tons  \
0        654.34    29.266667            7300.0              9400.0
1        654.34    29.266667            3300.0              1000.0
2        654.34    29.266667           10100.0             10200.0
3        654.34    29.266667            2800.0              4900.0
4        654.34    29.266667            1300.0               500.0
...         ...          ...               ...                 ...
99844    152.54    22.280000            2013.0              5152.0
99845    182.50    29.200000             258.0               391.0
99846    182.50    29.200000             105.0               281.0
99847    152.54    22.280000          152676.0            261435.0
99848    152.54    22.280000             244.0                95.0

       Yield_ton_per_hec
0               1.287671
1               0.303030
2               1.009901
3               1.750000
4               0.384615
...                  ...
99844           2.559364
99845           1.515504
99846           2.676190
99847           1.712352
99848           0.389344

[99849 rows x 13 columns]>
```

```
#this is used to display rows and colums
data.shape
```

```
(99849, 13)
```

```
#This is Generate descriptive statistics for numeric columns in DataFrame.
data.describe()
```

	Unnamed: 0	N	P	K	pH	rainfall
count	99849.000000	99849.000000	99849.000000	99849.000000	99849.000000	99849.000000
mean	49924.000000	69.816823	41.593656	42.037827	5.643624	701.151085
std	28824.067851	39.571469	15.056508	28.430263	0.505283	604.701552
min	0.000000	10.000000	10.000000	10.000000	3.820000	3.274569
25%	24962.000000	50.000000	40.000000	20.000000	5.360000	157.310000
50%	49924.000000	75.000000	40.000000	30.000000	5.540000	579.750000
75%	74886.000000	80.000000	60.000000	50.000000	5.960000	1110.780000
max	99848.000000	180.000000	125.000000	200.000000	7.000000	3322.060000

```
#This is a square DataFrame where each cell contains the correlation coefficient between two columns
data.corr()
```

```
<ipython-input-14-34400226bd36>:2: FutureWarning: The default value of numeric_only in D
  data.corr()
```

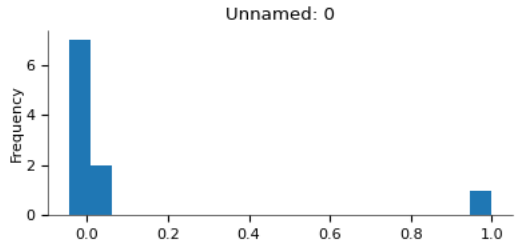
1 to 10 of 10 entries Filter

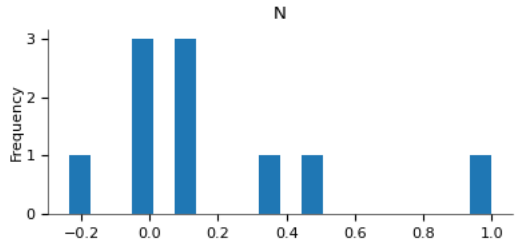
index	Unnamed: 0	N	P	
Unnamed: 0	1.0	0.011617835252197839	0.004737001136601322	0.0059
N	0.011617835252197839	1.0	0.34252121767707605	0.486
P	0.004737001136601322	0.34252121767707605	1.0	0.21
K	0.005913171875390982	0.48665034117593303	0.2103491823801325	
pH	-0.0034184834244922937	-0.23543679940310416	-0.2547760999622341	-0.246
rainfall	-0.044061321895436587	0.11190034413008412	0.11068634876320008	0.36
temperature	-0.02492186752427526	-0.044754647171951316	-0.05698758253585576	-0.078
Area_in_hectares	-0.0027550568849649968	0.009286557435104452	-0.05751426914280822	-0.120
Production_in_tons	0.02250865662629071	0.09788833146675463	-0.010697861601572543	-0.0263
Yield_ton_per_hec	0.00675608435157629	0.09022286552635281	0.07680552857027549	0.076

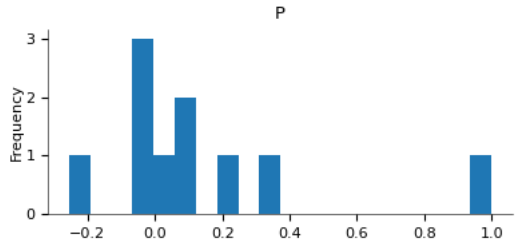
Show 25 per page

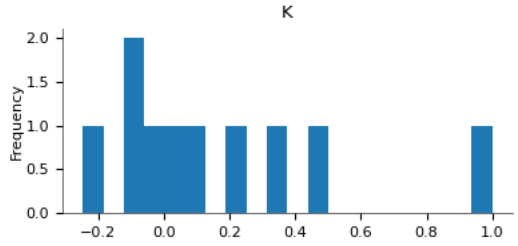
Like what you see? Visit the data table notebook to learn more about interactive tables.

Distributions

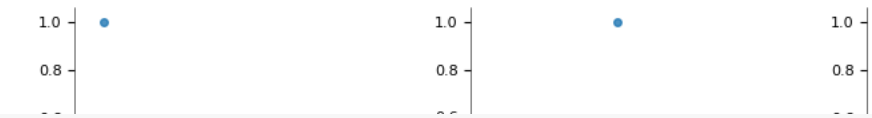






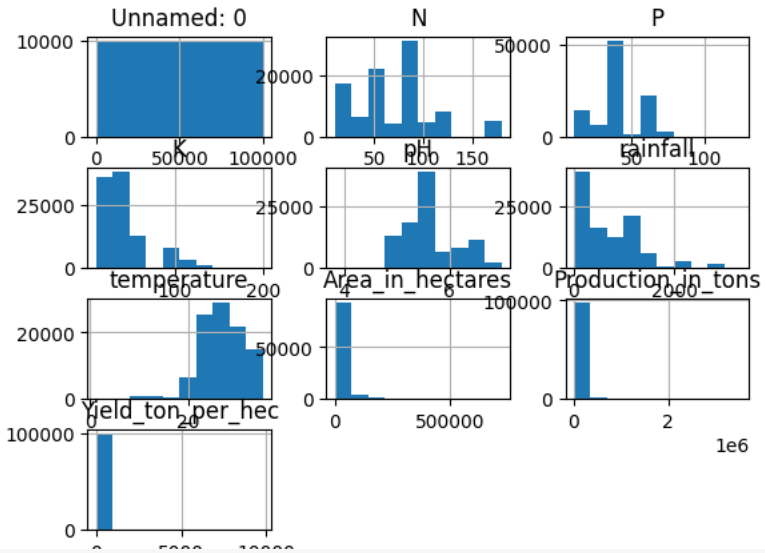


2-d distributions




```
# This is used to create histograms for all the numeric columns
data.hist()
```

```
array([[<Axes: title={'center': 'Unnamed: 0'}>,
        <Axes: title={'center': 'N'}>, <Axes: title={'center': 'P'}>],
       [<Axes: title={'center': 'K'}>, <Axes: title={'center': 'pH'}>,
        <Axes: title={'center': 'rainfall'}>],
       [<Axes: title={'center': 'temperature'}>,
        <Axes: title={'center': 'Area_in_hectares'}>,
        <Axes: title={'center': 'Production_in_tons'}>],
       [<Axes: title={'center': 'Yield_ton_per_hec'}>, <Axes: >,
        <Axes: >]], dtype=object)
```




```
# Data preprocessing
# This for assuming you a target column classification

X = data.drop(columns=['State_Name', 'Crop_Type', 'Crop'])
y = data['Crop_Type']
```

```
#printing of columns
print(data.columns)
```

```
Index(['Unnamed: 0', 'State_Name', 'Crop_Type', 'Crop', 'N', 'P', 'K', 'pH',
       'rainfall', 'temperature', 'Area_in_hectares', 'Production_in_tons',
       'Yield_ton_per_hec'],
      dtype='object')
```

```
# this is to Encode the target variable
le = LabelEncoder()
y = le.fit_transform(y)
```

```
#This is for Dropping varaibles columns from the dataset
data = data.drop(columns=['State_Name', 'Crop_Type',  'Crop'])
```

```
# this will Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Technique 1: Logistic Regression

logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
logistic_predictions = logistic_model.predict(X_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
# Evaluate the Logistic Regression model

print("Logistic Regression Model:")
print("Classification Report:\n", classification_report(y_test, logistic_predictions))
print("Accuracy Score:", accuracy_score(y_test, logistic_predictions))
```

```
Logistic Regression Model:
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.87      0.86      7640
           1       0.92      0.96      0.94      5577
           2       0.85      0.72      0.78      1400
           3       0.87      0.82      0.84      5353

    accuracy                           0.87     19970
   macro avg       0.87      0.84      0.85     19970
weighted avg       0.87      0.87      0.87     19970

Accuracy Score: 0.8726089133700551
```

```
# Technique 2: Decision Tree Classifier Model

from sklearn import tree
model=tree.DecisionTreeClassifier(criterion="entropy")
model.fit(X_train, y_train)
y_pred=model.predict(X_test)
```

```
# Evaluate Decision Tree Classifier Model

print(y_pred)
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred*100))
```

```
[3 1 0 ... 3 0 3]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7640
           1       1.00      1.00      1.00      5577
           2       1.00      1.00      1.00      1400
           3       1.00      1.00      1.00      5353

    accuracy                           1.00     19970
   macro avg       1.00      1.00      1.00     19970
weighted avg       1.00      1.00      1.00     19970

Accuracy Score: 0.38257386079118677
```