Thakur Educational Trust's (Regd.)
THAKUR COLLEGE OF SCIENCE & COMMERCE
AUTONOMOUS COLLEGE, PERMANENTLY AFFILIATED TO UNIVERSITY OF MUMBAI
NAAC Accredited Grade 'A' (3rd Cycle) & ISO 9001: 2015 (Certified)
Best College Award by University of Mumbai for the Year 2018-2019
CELEBRATING 25 YEARS OF GLORY

PRACTICAL JOURNAL OF DATA SCIENCE

SUBJECT GUIDE
Mrs. Rimsy Dua
(ASST. PROFESSOR)

SUBMITTED BY:
MOHAMMED USMAN
ROLL NO: 4830

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR QUALIFYING MSCIT PART 1
(SEMESTER I EXAMINATION)

Thakur Educational Trust's (Regd.)
THAKUR COLLEGE OF SCIENCE & COMMERCE
AUTONOMOUS COLLEGE, PERMANENTLY AFFILIATED TO UNIVERSITY OF MUMBAI
NAAC Accredited Grade 'A' (3rd Cycle) & ISO 9001: 2015 (Certified)
Best College Award by University of Mumbai for the Year 2018-2019
CELEBRATING 25 YEARS OF GLORY

CERTICICATE OF APPROVAL

This is to certify that **Mr. Mohammed Usman** student of "Master of Science(Information Technology)" of "Thakur College Of Science and Commerce". Roll No.4830 has successfully completed and submitted the practical & assignment entitled "Data Science" in partial fulfillment as per the syllabus defined by the University of Mumbai in the academic year 2023-24.

It is further certified that the student has completed all the required phases of the practical & assignment.

_____

HEAD OF DEPARTMENT

_____ _____

PROFESSOR INCHARGE **EXTERNAL EXAMINER**

INDEX

SR. NO	PRACTICAL NAME	DATE	PAGE	SIGN
1	TO PERFOM HORUS CONVERSION ON TEXT DELIMITED FILES AND PICTURE FILES	19-08-2023		
2	TO PERFORM AUDITING WITH: A. FIXER UILITIES B. DATA BINNING C. AVERAGING OF DATA D. OUTLIER DETECTION	02-09-2023		
3	TO UNDERSTAND THE USE OF THE RETRIEVE SUPERSTEP	09-09-2023		
4	TO UNDERSTAND THE USE OF ASSESS SUPERSTEP	09-09-2023		
5	TO PREDICT THE PRICE (IN FUTURE) USING A SUPERVISED LEARNING ALGORITHM (LINEAR REGRESSION)	30-10-2023		
6	TO PREDICT THE LABEL USING AN UNSUPERVISED LEARNING ALGORITHM (CLUSTERING)	07-10-2023		
7	TO PERFORM VISUALIZATION OF DATA	14-10-2023		
8	TO PERFORM CLASSIFICATION USING SVM	21-10-2023		
9	TO PERFORM DATA CLEANING	14-10-2023		
10	TO PERFORM LINE DETECTION USING HOUGH TRANSFORM	21-10-2023		
11	TO UNDERSTAND THE USE OF BAR CHART,PIE CHART AND HISTOGRAM IN DATASCIENCE	28-10-2023		
12	TO IMPLEMENT COMPLEX DATA VISUALIZATIONS	28-10-2023		
13	TO IMPLEMENT BASIC TEXT PROCESSING USING NATURAL LANGUAGE PROCESSING	28-10-2023		

PRACTICAL-1
A-Utility Start CSV to HORUS

```
import pandas as pd
sInputFileName = '/content/Country.csv'
InputData = pd.read_csv(sInputFileName, encoding = "latin-1")
print('Input Data Values ==================')
print(InputData, "\n==================")
# Processing Rules
ProcessData = InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis = 1, inplace = True)
ProcessData.drop('ISO-3-Code', axis = 1, inplace = True)
# Rename Country and ISO-M49
ProcessData.rename(columns = {'Country': 'CountryName'}, inplace = True)
ProcessData.rename(columns = {'ISO-M49': 'CountryNumber'}, inplace = True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace = True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis = 0, ascending =False, inplace=True)
print('\n=====Process Data Values=====')
print(ProcessData)
# Output Agreement
OutputData=ProcessData
sOutputFileName='/content/drive/MyDrive/Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('CSV to HORUS - Done')
```

Output:

```
Input Data Values ==================================
                     Country ISO-2-CODE ISO-3-Code  ISO-M49
0                Afghanistan         AF         AFG        4
1              Aland Islands         AX         ALA      248
2                    Albania         AL         ALB        8
3                    Algeria         DZ         DZA       12
4             American Samoa         AS         ASM       16
..                       ...        ...         ...      ...
242  Wallis and Futuna Islands         WF         WLF      876
243            Western Sahara         EH         ESH      732
244                     Yemen         YE         YEM      887
245                    Zambia         ZM         ZMB      894
246                  Zimbabwe         ZW         ZWE      716
[247 rows x 4 columns]
==========================
=================Process Data Values=============
                            CountryName
CountryNumber
716                            Zimbabwe
894                              Zambia
887                               Yemen
732                      Western Sahara
876           Wallis and Futuna Islands
...                                 ...
16                      American Samoa
12                             Algeria
8                              Albania
248                      Aland Islands
4                          Afghanistan
[247 rows x 1 columns]
CSV to HORUS - Done
```

PRACTICAL-1

B-Image to Horus

```
from skimage import io
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
sInputFileName='/content/audi.jpg'
InputData = io.imread(sInputFileName, pilmode='RGBA')
plt.imshow(InputData)
InputData.shape
print('Input Data Values')
print('X: ',InputData.shape[0])
print('Y: ', InputData. shape[1])
print('RGBA: ', InputData.shape[2])
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
ProcessRawData
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
sColumns= [ 'XAxis', 'YAxis', 'Red', 'Green', 'Blue','Aplha']
ProcessData.columns=sColumns
ProcessData
print('Rows: ',ProcessData.shape[0])
print('Columns :',ProcessData.shape[1])
OutputData = ProcessData
OutputData.to_csv('Image to HORUS.csv', index = False)
```

Output:

```
Input Data Values
X:  168
Y:  300
RGBA:  4
Rows:  33600
Columns : 6
```

PRACTICAL-2

A-Fixer Uilities

```
#Program to demonstrate fixer utilities
#Removing leading or lagging spaces from a data entry
print("#Removing leading or lagging spaces from a data entry")
baddata="  Datascience with too many spaces is bad   "
print('>',baddata,'<')
cleandata=baddata.strip()
print(">",cleandata,"<")
#Removing non-printable characters from the dataentry
print("\n#Removing non-printable characters from the data entry")
import string
printable=set(string.printable)
baddata="Data\x00science with too many funny\x01 characters is \x10bad!!!"
cleandata=''.join(filter(lambda x:x in string.printable,baddata))
print("Baddata:",baddata)
print("Cleandata:",cleandata)
print('\n#Reformatting date entry to match specific formatting criteria')
print('#Convert YYYY/MM/DD TO DD Month YYYY')
import datetime
baddate = datetime.date(2019,10,31)
baddata = format(baddate,'%Y-%m-%d')
gooddate = datetime.datetime.strptime(baddata,'%Y-%m-%d')
gooddata = format(gooddate,'%d %B %Y')
print('BadData: ',baddata)
print('GoodData: ',gooddata)
```

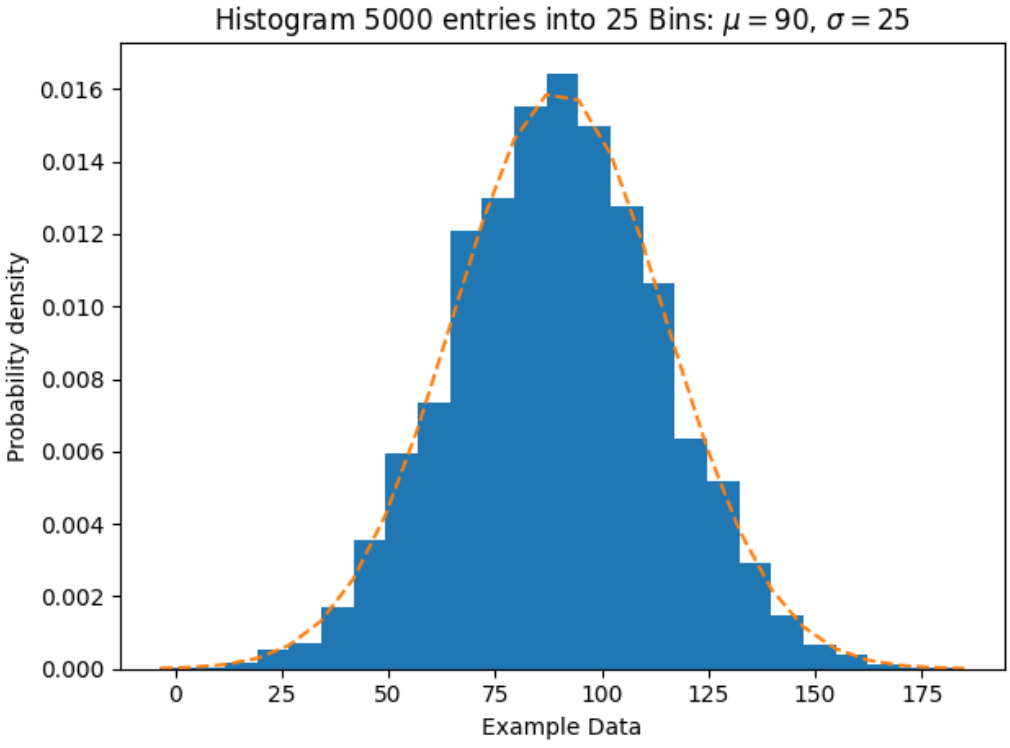
OutPut:

```
#Removing leading or lagging spaces from a data entry
>    Datascience with too many spaces is bad    <
> Datascience with too many spaces is bad <
#Removing non-printable characters from the data entry
Baddata: Data?science with too many funny characters is bad!!!
Cleandata: Datascience with too many funny characters is bad!!!
#Reformatting date entry to match specific formatting criteria
#Convert YYYY/MM/DD TO DD Month YYYY
BadData:  2019-10-31
GoodData:  31 October 2019
```

PRACTICAL-2

B-Data Binning

```
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import scipy.stats as stats
np.random.seed(0)
# example data
mu = 90 # mean of distribution
sigma = 25 # standard deviation of distribution
x = mu + sigma * np.random.randn(5000)
num_bins = 25
fig, ax = plt.subplots()
# the histogram of the data
n, bins, patches = ax.hist(x, num_bins, density=1)
# add a 'best fit' line
y = stats.norm.pdf(bins, mu, sigma)
# mlab.normpdf(bins, mu, sigma)
ax.plot(bins, y, '--')
ax.set_xlabel('Example Data')
ax.set_ylabel('Probability density')
sTitle=r'Histogram ' + str(len(x)) + ' entries into ' + str(num_bins)
+ ' Bins: $\mu=' + str(mu) + '$, $\sigma=' +str(sigma) + '$'
ax.set_title(sTitle)
fig.tight_layout()
sPathFig = ' /content/Histogram.png'
fig.savefig(sPathFig)
plt.show()
```



PRACTICAL-2

C-Averaging Of Data

```
import pandas as pd
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
print('Working Base :',Base, ' using ')
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,usecols
=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
AllData=IP_DATA_ALL[['Country', 'Place_Name','Latitude']]
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
print(MeanData)
```

OutPut:

```
    Country Place_Name  Latitude
0         BW   Gaborone  -24.6464
1         BW   Gaborone  -24.6464
2         BW   Gaborone  -24.6464
3         BW   Gaborone  -24.6464
4         BW   Gaborone  -24.6464
..       ...         ...       ...
194       DZ    Algiers   36.7631
195       DZ    Algiers   36.7631
196       DZ    Algiers   36.7631
197       DZ    Algiers   36.7631
198       DZ    Algiers   36.7631

[199 rows x 3 columns]
------------(MEAN)-------------
Country  Place_Name
BW       Gaborone       -24.6464
DZ       Algiers         36.7631
GH       Accra           5.5500
         Kumasi          6.6833
         Takoradi        4.8833
         Tema            5.6167
MZ       Maputo          -25.9653
NE       Niamey          13.5167
Name: Latitude, dtype: float64
```

PRACTICAL-2

D-Outliers Detection

```
import pandas as pd
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
print('Working Base :',Base)
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,usecols
=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
LondonData=IP_DATA_ALL.loc[IP_DATA_ALL['Place_Name']=='London']
AllData=LondonData[['Country', 'Place_Name','Latitude']]
print('All Data\n', AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
StdData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].std()
print('Outliers')
UpperBound=float(MeanData+StdData)
print('Higher than ', UpperBound)
OutliersHigher=AllData[AllData.Latitude>UpperBound]
print(OutliersHigher)
LowerBound=float(MeanData-StdData)
print('Lower than ', LowerBound)
OutliersLower=AllData[AllData.Latitude<LowerBound]
print(OutliersLower)
print('Not Outliers')
OutliersNot=AllData[(AllData.Latitude>=LowerBound) &
(AllData.Latitude<=UpperBound)]
print(OutliersNot)
```

Output:

```
All Data
Empty DataFrame
Columns: [Country, Place_Name, Latitude]
Index: []
Outliers
Higher than  Series([], Name: Latitude, dtype: float64)
```

PRACTICAL-3

Retrieve Superstep

```
import os
import pandas as pd
InputFileName = '/content/IP_DATA_ALL.csv'
IP_DATA_ALL=pd.read_csv(InputFileName,header=0,low_memory=False,
usecols=['Country','Place.Name','Latitude','Longitude'])

IP_DATA_ALL.rename(columns={'Place.Name': 'Place_Name'}, inplace=True)
ROUTERLOC = IP_DATA_ALL.drop_duplicates(subset=None, keep='first',
inplace=False)

print('Rows :',ROUTERLOC.shape[0])
print('Columns :',ROUTERLOC.shape[1])
newFile = '/content/Retrieve_Router_Location.csv'
ROUTERLOC.to_csv(newFile, index = False)
print('Done...!')
```

Output:

```
Rows : 101376
Columns : 4
Done...!
```

PRACTICAL-4

ASSESS SUPERSTEP

1. REPLACING NAN VALUES WITH MEAN

```
import pandas as pd
import numpy as np
df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 21, 28], [55, np.nan, 8, 12],
 [15, 14, np.nan, 8], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
 columns=['Apple', 'Orange', 'Banana', 'Pear'],
 index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
 'Basket5', 'Basket6'])
print("THE ORIGINAL VALUES")
print(df)
print("REPLACING THE  VALUES WITH MEAN")
df.fillna(df.mean(),inplace=True)
df
```

Output:

```
THE ORIGINAL VALUES
         Apple  Orange  Banana  Pear
Basket1   10.0     NaN    30.0  40.0
Basket2    7.0    14.0    21.0  28.0
Basket3   55.0     NaN     8.0  12.0
Basket4   15.0    14.0     NaN   8.0
Basket5    7.0     1.0     1.0   NaN
Basket6    NaN     4.0     9.0   2.0
REPLACING THE  VALUES WITH MEAN
```

2.REPLACING NAN VALUES WITH MEDIAN

```
import pandas as pd
import numpy as np
df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 21, 28], [55, np.nan, 8, 12],
 [15, 14, np.nan, 8], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
 columns=['Apple', 'Orange', 'Banana', 'Pear'],
 index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
 'Basket5', 'Basket6'])
print("THE ORIGINAL VALUES")
print(df)
print("REPLACING THE  VALUES WITH MEAN")
df.fillna(df.median(),inplace=True)
df
```

Output:

```
THE ORIGINAL VALUES
         Apple  Orange  Banana  Pear
Basket1   10.0     NaN    30.0  40.0
Basket2    7.0    14.0    21.0  28.0
Basket3   55.0     NaN     8.0  12.0
Basket4   15.0    14.0     NaN   8.0
Basket5    7.0     1.0     1.0   NaN
Basket6    NaN     4.0     9.0   2.0
REPLACING THE  VALUES WITH MEAN
```

3.REPLACING NAN VALUES WITH MODE

```
import pandas as pd
import numpy as np
df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 21, 28], [55, np.nan, 8, 12],
 [15, 14, np.nan, 8], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
 columns=['Apple', 'Orange', 'Banana', 'Pear'],
 index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
 'Basket5', 'Basket6'])
print("THE ORIGINAL VALUES")
print(df)
print("REPLACING THE  VALUES WITH MEAN")
for column in df.columns:
  df[column].fillna(df[column].mode()[0], inplace=True)
df
```

Output:

```
THE ORIGINAL VALUES
         Apple  Orange  Banana  Pear
Basket1   10.0     NaN    30.0  40.0
Basket2    7.0    14.0    21.0  28.0
Basket3   55.0     NaN     8.0  12.0
Basket4   15.0    14.0     NaN   8.0
Basket5    7.0     1.0     1.0   NaN
Basket6    NaN     4.0     9.0   2.0
REPLACING THE  VALUES WITH MEAN
```

4.REPLACING NAN VALUES WITH MINIMUM

```
import pandas as pd
import numpy as np
df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 21, 28], [55, np.nan, 8, 12],
 [15, 14, np.nan, 8], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
 columns=['Apple', 'Orange', 'Banana', 'Pear'],
 index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
 'Basket5', 'Basket6'])
print("THE ORIGINAL VALUES")
print(df)
print("REPLACING THE  VALUES WITH MEAN")
df.fillna(df.min(),inplace=True)
df
```

Output:

```
THE ORIGINAL VALUES
         Apple  Orange  Banana  Pear
Basket1   10.0     NaN    30.0  40.0
Basket2    7.0    14.0    21.0  28.0
Basket3   55.0     NaN     8.0  12.0
Basket4   15.0    14.0     NaN   8.0
Basket5    7.0     1.0     1.0   NaN
Basket6    NaN     4.0     9.0   2.0
REPLACING THE  VALUES WITH MEAN
```

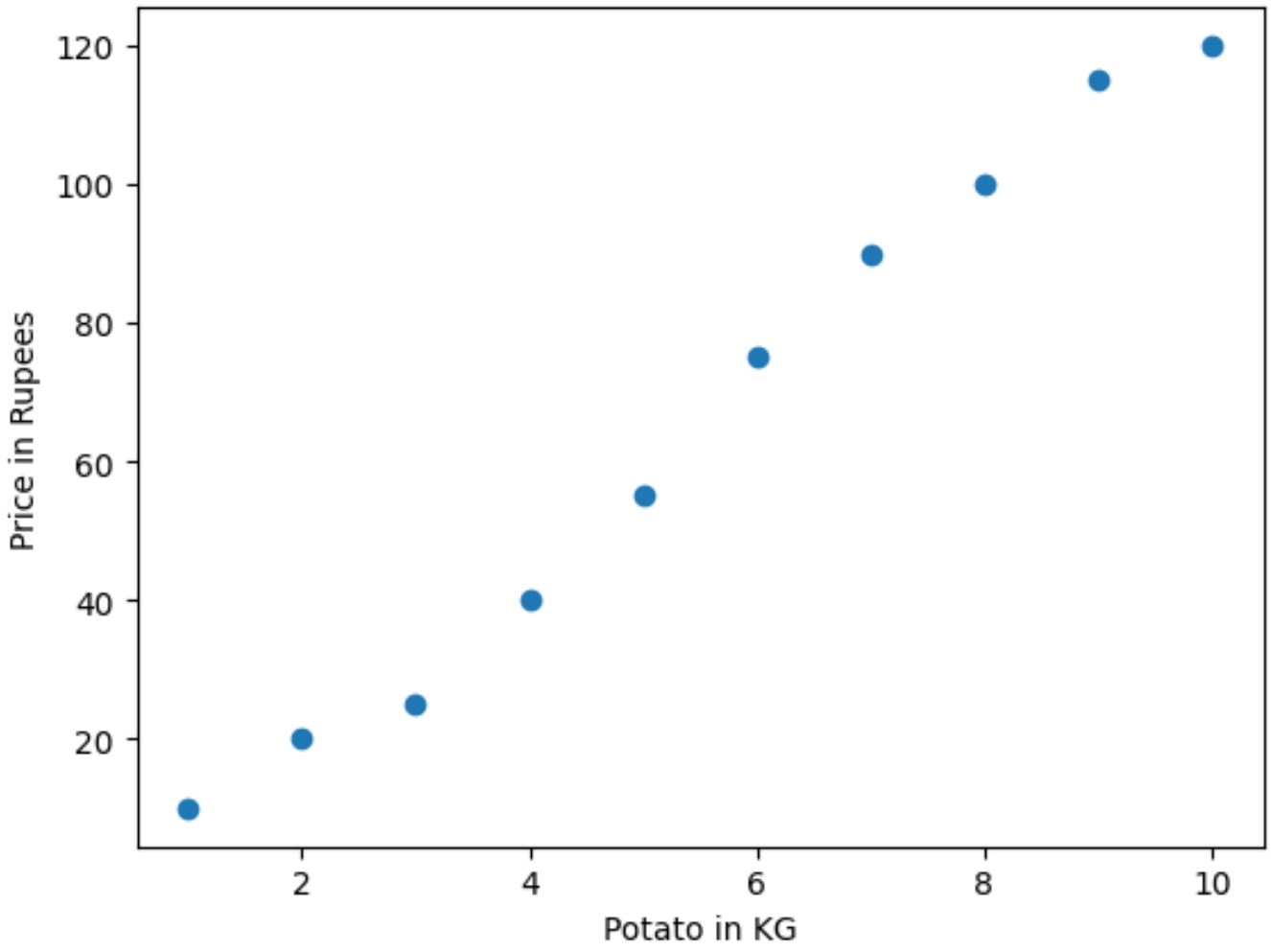
PRACTICLE-5

```
# To predict the price of any item using supervised learning algorithm.
# (Linear Regression)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

#Load the Dataset
df=pd.read_csv("/content/PotatoPrice.csv")
print(df)

#DATA VISUALIZATION
%matplotlib inline
plt.xlabel("Potato in Kg")
plt.ylabel("Price in Rupees")
plt.scatter(df.potato_kg,df.price)
X=df[["potato_kg"]]
Y=df["price"]
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2)
print("X_train", X_train)
print("X_test", X_test)
print("Y_train", Y_train)
print("Y_test", Y_test)

#Train Dataset using model
reg=LinearRegression()
reg.fit(X_train,Y_train)
reg.predict(X_test)
#ACCURACY OF THE MODEL
print('ACCURACY:', reg.score(x_test,y_test))

#Take the user input
x=input("Enter the potato quantity in kg: \n")
array=np.array(x)
fvalu=array.astype(np.float)
fvalu_2D=([[fvalu]])
my_prediction=reg.predict(fvalu_2D)
price=np.array(my_prediction)
price=price.item()
print('So',x,'Kilogram potato price is ',price,' Rupees')
```

PRACTICLE-5

Output:

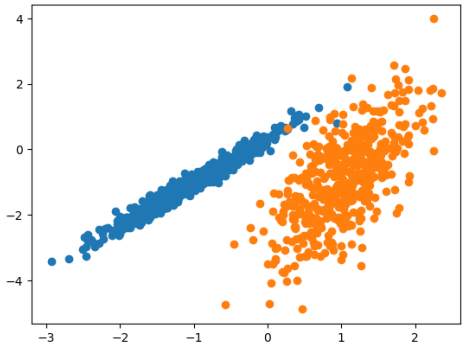
```
Trained Dataset:  [0.17857143 129.82142857]

ACCURACY: 0.9681122448979591

Enter the potato quantity in kg: 10
So 10 Kilogram potato price is  124.99999999999999  Rupees
```



PRACTICAL-6

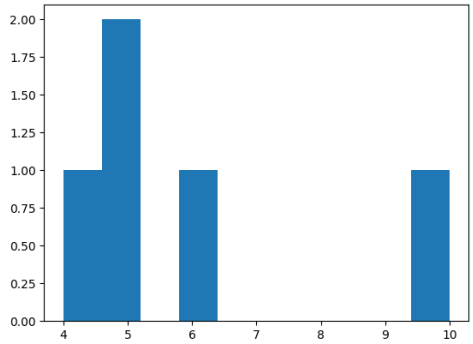
SCATTER PLOT

```
from numpy import where
from sklearn.datasets import
make_classification
from matplotlib import pyplot
x,y=make_classification(n_samples=1000,
    n_feat ures=2,n_informative=2,n_redundant=0,
    n_clusters_per_class=1,random_state=4)
for class_value in range(2):
row_ix=where(y==class_value)
pyplot.scatter(x[row_ix,0],x[row_ix,1])
pyplot.show()
```



HISTOGRAM USING MATPLOTLIB

```
# importing matplotlib module
from matplotlib import pyplot as plt
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot histogram
plt.hist(y)
# Function to show the plot
plt.show()
```

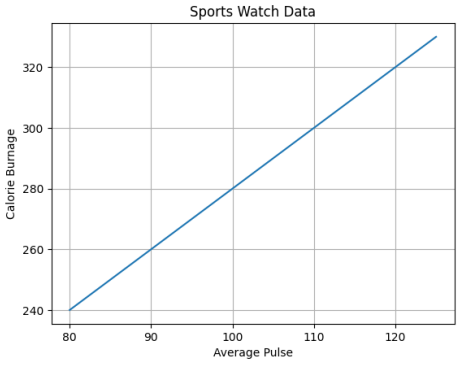


PIE CHARTS

```
import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
plt.pie(y)
plt.show()
```

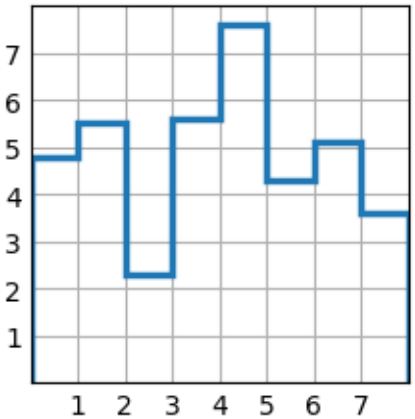
ADDING GRID LINES TO PLOT

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.plot(x, y)
plt.grid()
plt.show()
```



STAIRS(VALUES)

```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use("_mpl-gallery")
# make data
y = [4.8,5.5,2.3,5.6,7.6,4.3,5.1,3.6]
# plot
fig, ax = plt.subplots()
ax.stairs(y, linewidth=2.5)
ax.set(xlim=(0,8), xticks=np.arange(1,8),
       ylim=(0,8), yticks=np.arange(1,8))
plt.show()
```



PRACTICAL-7
Visualization Of Data

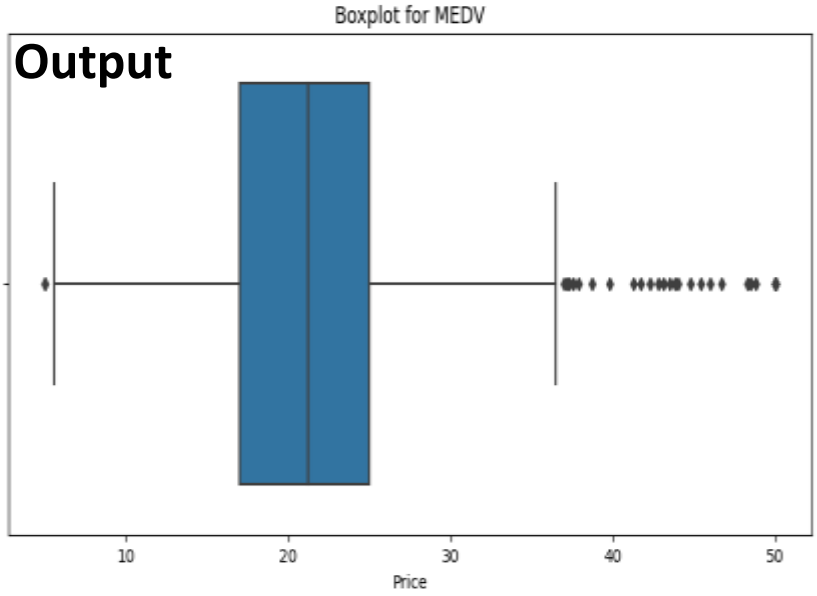
 # Step 1

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
import warnings
warnings.filterwarnings('ignore')
boston_df=pd.read_csv('/content/boston.csv')
print(boston_df)
```

 # Step 2

```
plt.figure(figsize=(10,5))
sns.boxplot(x = boston_df.Price)
plt.title('Boxplot for MEDV')
plt.show()
```

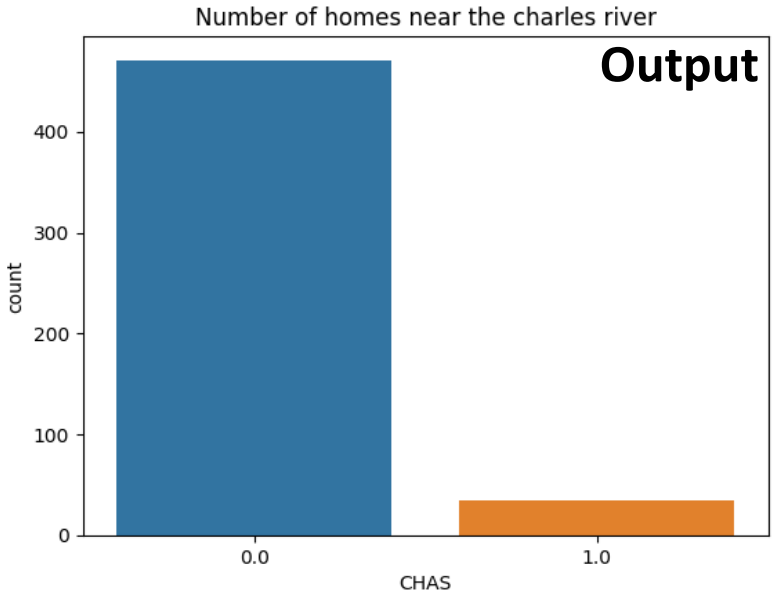
Output



 # Step 3

```
ax2 = sns.countplot
(x='CHAS', data=boston_df)
ax2.set_title
('No. of homes near charles river')
```

Output

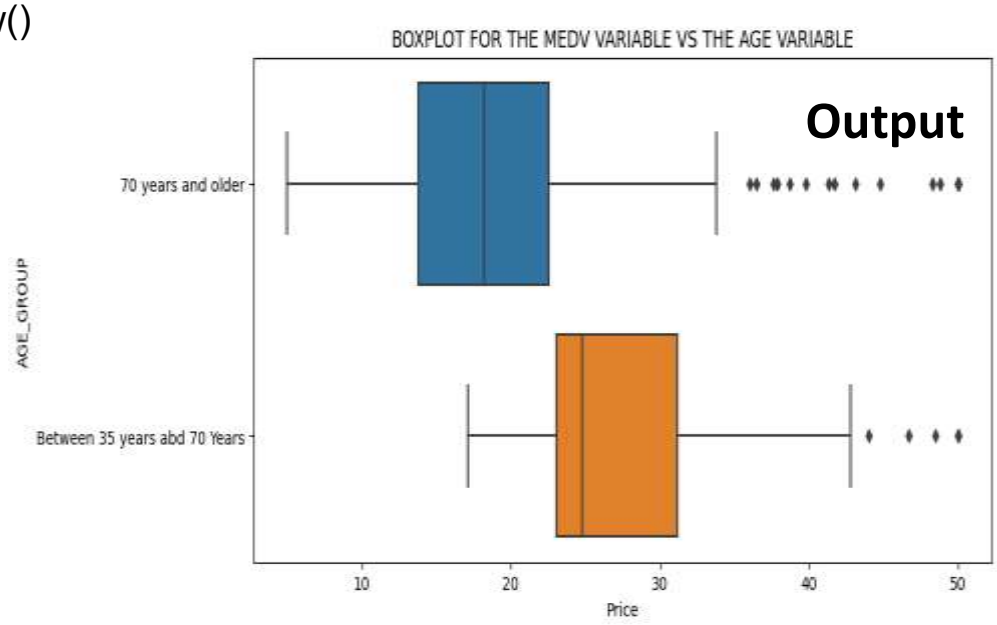


 # Step 4

```
boston_df.loc[(boston_df["AGE"]<35),"AGE_GROUP"] = "35 years and younger"
boston_df.loc[(boston_df["AGE"]<35) & (boston_df["AGE"]<70) ,"AGE_GROUP"]
 = "Between 35 years abd 70 Years"
boston_df.loc[(boston_df["AGE"]>=70),"AGE_GROUP"] = "70 years and older"
plt.figure(figsize=(10,5))
sns.boxplot(x=boston_df.Price, y=boston_df.AGE_GROUP, data=boston_df)
plt.title("BOXPLOT FOR THE MEDV VARIABLE VS THE AGE VARIABLE")
```

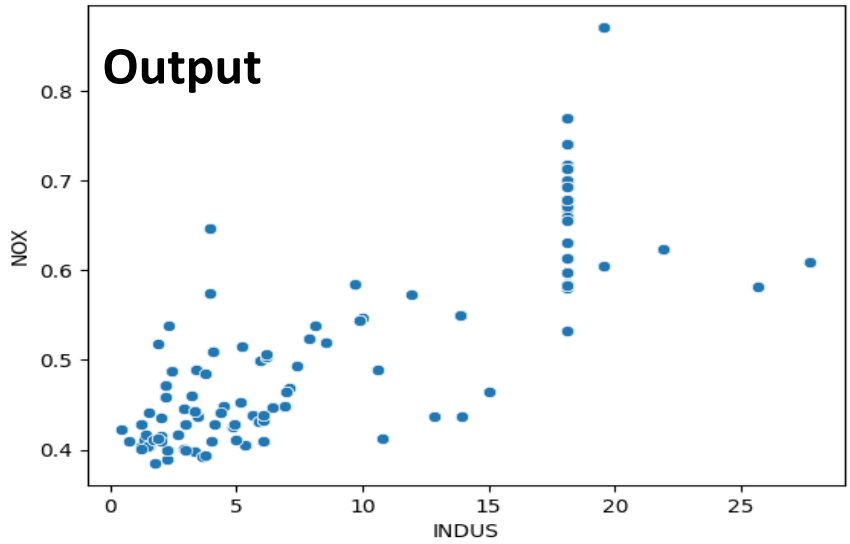
PRACTICAL-7

plt.show()



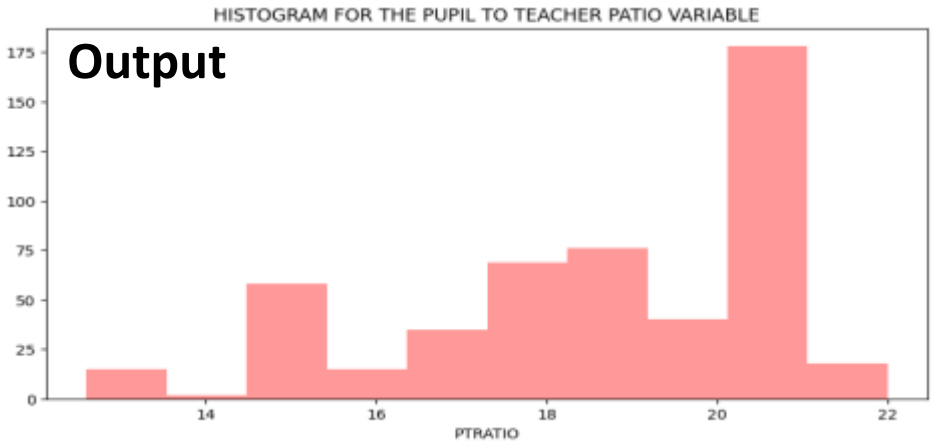
---

ax4 = sns.scatterplot(y='NOX', x="INDUS", data=boston_df)
ax4.set_title("Nitric oxide concentration per proportion of no-retail business acres per town")



---

plt.figure(figsize=(10,5))
sns.distplot(a=boston_df.PTRATIO, bins=10, kde=False, color="red")
plt.title("HISTOGRAM FOR THE PUPIL TO TEACHER PATIO VARIABLE")
plt.show()



PRACTICAL-8

UNDERSTAND THE USE OF DATA CLEANING

```
import pandas as pd
df=pd.read_csv("/content/credit.csv", encoding='latin-1')
df

# Remove all rows with null values
df.dropna(inplace =True)
print(df.to_string())

# convert to date
df['trdate'] = pd.to_datetime(df['trdate'])
print(df.to_string())
```

Output:

```
    ipaddress userid   accnum  age            shipping address      trdate    trtime  trvalue   prcategory  unitspur
0  3.56.123.0   john  2564511   32  152,orchid lane,WA 987, US  15-05-2020  15:00:05  $121.58     clothing         1
1  3.56.123.0   john  2564511   32  152,orchid lane,WA 987, US  10-06-2020  10:23:10   $79.23  electronics         2
    ipaddress userid   accnum  age            shipping address     trdate    trtime  trvalue   prcategory  unitspur
0  3.56.123.0   john  2564511   32  152,orchid lane,WA 987, US 2020-05-15  15:00:05  $121.58     clothing         1
1  3.56.123.0   john  2564511   32  152,orchid lane,WA 987, US 2020-10-06  10:23:10   $79.23  electronics         2
```

PRACTICAL-09

```
# SVC Classification
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as maping
import pandas as pd
dataset = pd.read_csv('/content/iris.csv')
dataset
dataset.head()
%matplotlib inline
img = maping.imread('/content/iris_types.jpg')
plt.figure(figsize=(5,15))
plt.axis('off')
plt.imshow(img)
X = dataset.iloc[:,:4].values
y = dataset['species'].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=82)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print('X_train', X_train)
print('X_test', X_test)
from sklearn.svm import SVC
svcclassifier = SVC(kernel = 'linear', random_state=0)
svcclassifier.fit(X_train, y_train)
y_pred = svcclassifier.predict(X_test)
print(y_pred)
y_compare = np.vstack((y_test,y_pred)).T
print(y_compare[:5,:])
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
a = cm.shape
corrPred = 0
falsePred = 0
for row in range(a[0]):
    for c in range(a[1]):
        if row == c:
            corrPred +=cm[row,c]
        else:
            falsePred += cm[row,c]
print('Correct predictions: ', corrPred)
print('False predictions', falsePred)
kernelLinearAccuracy = corrPred/(cm.sum())
print ('Accuracy of the SVC Clasification is: ', corrPred/(cm.sum()))
```

PRACTICAL-09

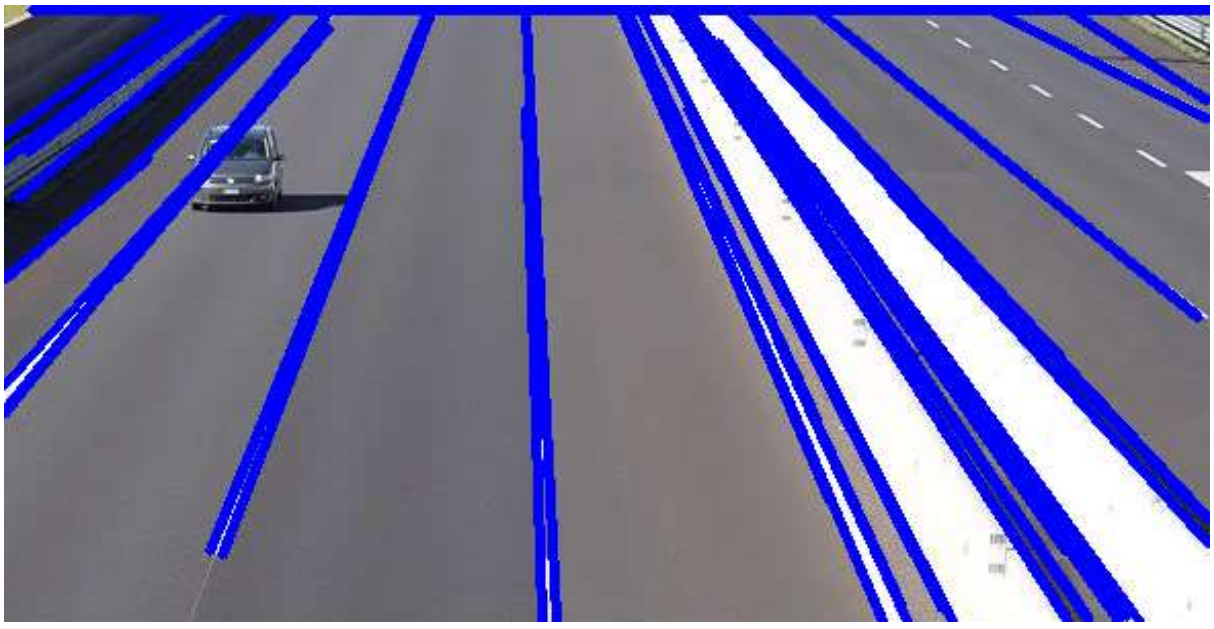
Output:

```
['virginica' 'virginica' 'setosa' 'setosa' 'setosa' 'virginica'
 'versicolor' 'versicolor' 'virginica' 'versicolor' 'versicolor'
 'virginica' 'setosa' 'setosa' 'setosa' 'setosa' 'virginica'
'versicolor'
 'setosa' 'versicolor' 'setosa' 'virginica' 'setosa' 'virginica'
 'virginica' 'versicolor' 'virginica' 'setosa' 'virginica'
'versicolor']
[['virginica' 'virginica']
 ['virginica' 'virginica']
 ['setosa' 'setosa']
 ['setosa' 'setosa']
 ['setosa' 'setosa']]
[[11  0  0]
 [ 0  8  1]
 [ 0  0 10]]
Correct predictions:  29
False predictions 1
Accuracy of the SVC Clasification is:  0.9666666666666667
```

PRACTICAL-10

```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
img = cv2.imread('/content/hough.jpg', cv2.IMREAD_COLOR)
cv2_imshow(img)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50, 200)
lines = cv2.HoughLinesP(edges, 1, np.pi/180, 68, minLineLength=15,
maxLineGap=250)
for line in lines:
   x1, y1, x2, y2 = line[0]
   cv2.line(img, (x1, y1), (x2, y2), (255, 0, 0), 3)
print("Line Detection using Hough Transform")
from google.colab.patches import cv2_imshow
cv2_imshow(img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

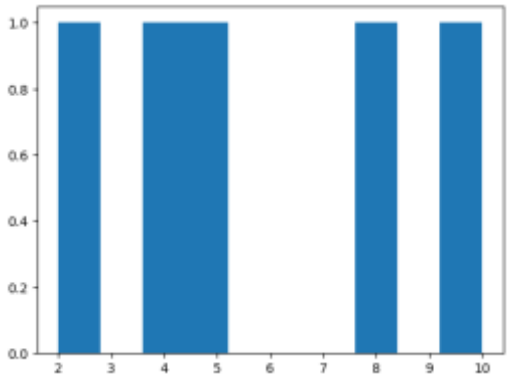
```
Line Detection using Hough Transform
```



PRACTICAL 11

 # HISTOGRAM USING MATPLOTLIB

```
# importing matplotlib module
from matplotlib import pyplot as plt
# Y-axis values
y = [10, 5, 8, 4, 2]
# Function to plot histogram
plt.hist(y)
# Function to show the plot
plt.show()
```

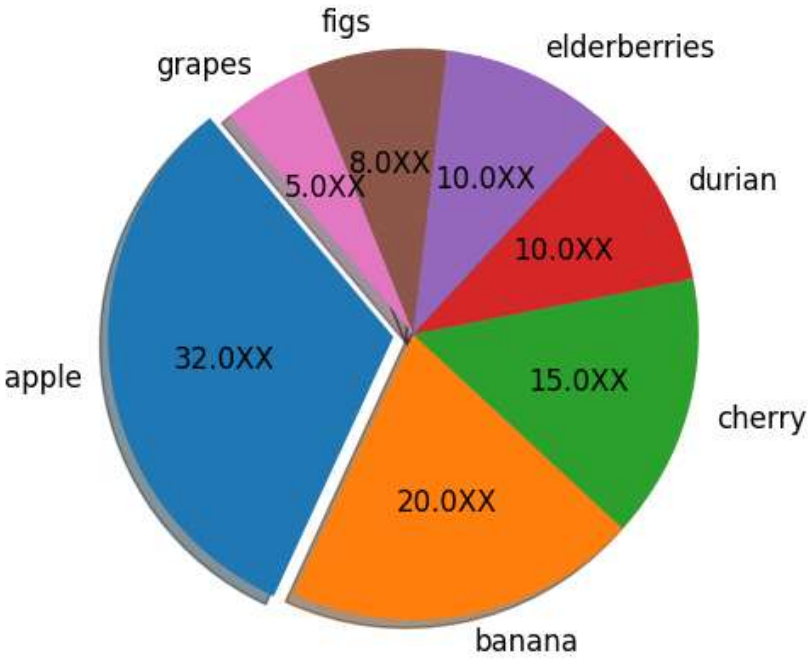


#PIE CHARTS

```
import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
plt.pie(y)
plt.show()
```

#PIE CHART WITH LABEL

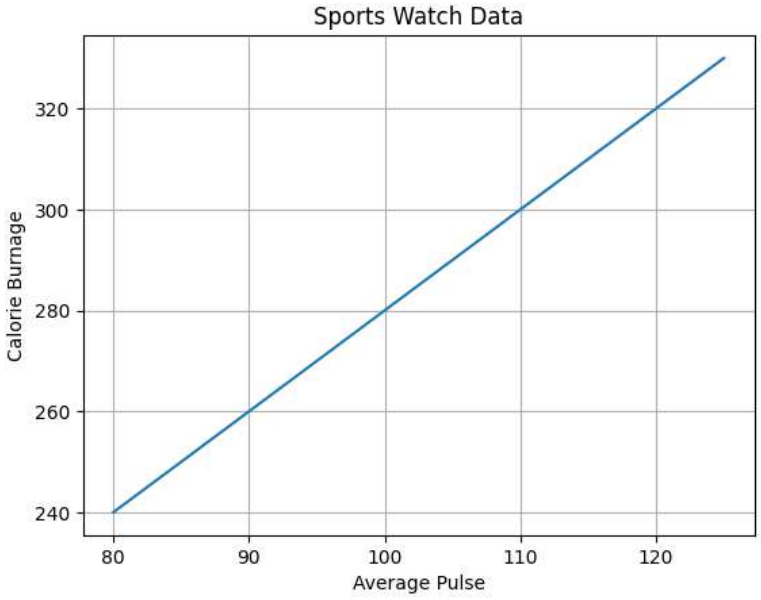
```
import matplotlib.pyplot as plt
labels = 'apple', 'banana', 'cherry', 'durian', 'elderberries', 'figs', 'grapes'
sizes= [32, 20, 15, 10, 10, 8, 5]
p = plt.pie (sizes, labels=labels, explode=(0.07, 0, 0, 0, 0, 0, 0),
            autopct='%1.1fXX', startangle=130, shadow=True)
plt.axis('equal')
for i, (apple, banana, cherry, durian, elderberries, figs, grapes) in enumerate(p):
  if i > 0:
    apple.set_fontsize(12)
    banana.set_fontsize(12)
    cherry.set_fontsize(12)
    durian.set_fontsize(12)
    elderberries.set_fontsize(12)
    figs.set_fontsize(12)
    grapes.set_fontsize(12)
plt.show()
```



PRACTICAL 11

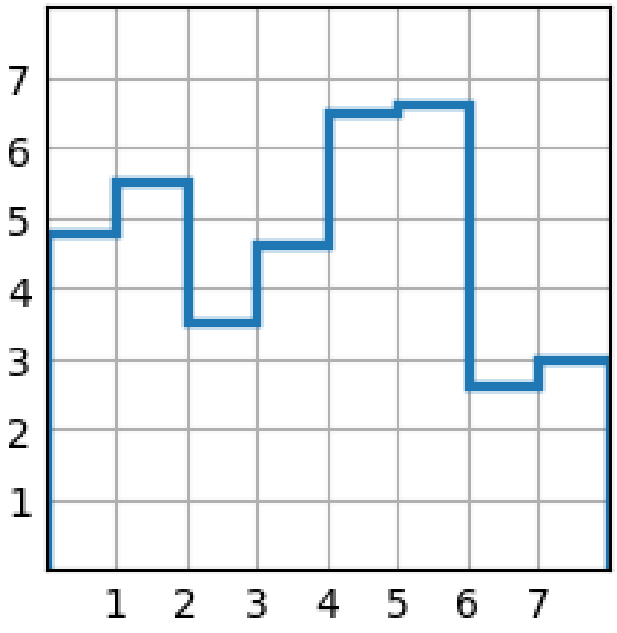
#ADDING GRID LINES TO PLOT

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.plot(x, y)
plt.grid()
plt.show()
```



#STAIRS VALUE

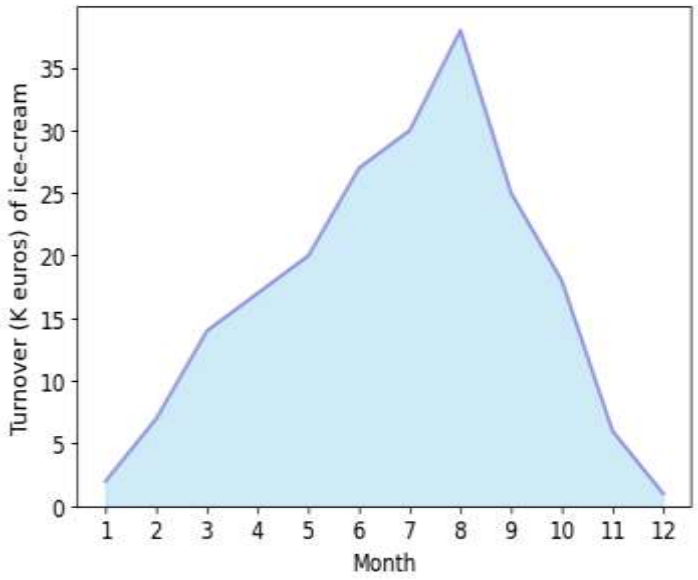
```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('_mpl-gallery')
# make data
y = [4.8, 5.5, 3.5, 4.6, 6.5, 6.6, 2.6, 3.0]
# plot
fig, ax = plt.subplots()
ax. stairs(y, linewidth=2.5)
ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))
plt.show()
```



PRACTICAL 12

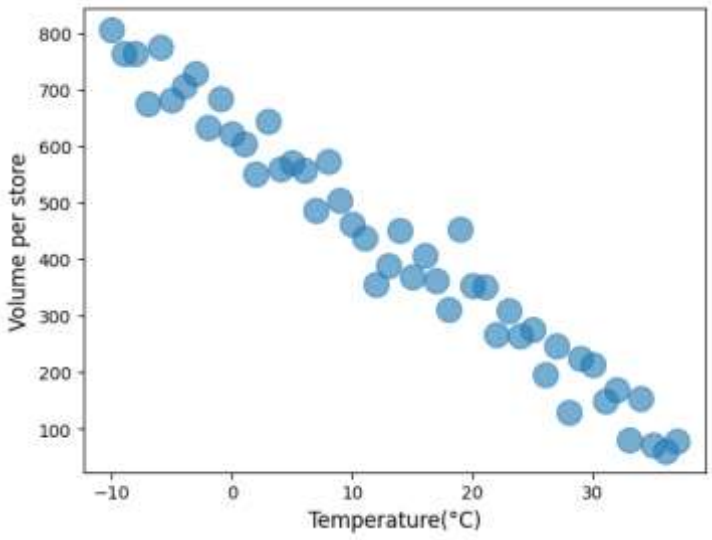
#Area Graph

```
import datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
turnover = [2, 7, 14, 17, 20, 27, 30, 38, 25, 18, 6, 1]
plt.fill_between(np.arange(12), turnover,
                 color="skyblue", alpha=0.4)
plt.plot(np.arange(12), turnover, color="Slateblue",
         alpha=0.6, linewidth=2)
plt.tick_params(labelsize=12)
plt.xticks(np.arange(12), np.arange(1,13))
plt.xlabel('Month', size=12)
plt.ylabel('Turnover (K euros) of ice-cream', size=12)
plt.ylim(bottom=0)
plt.show()
```



#Scatter plots

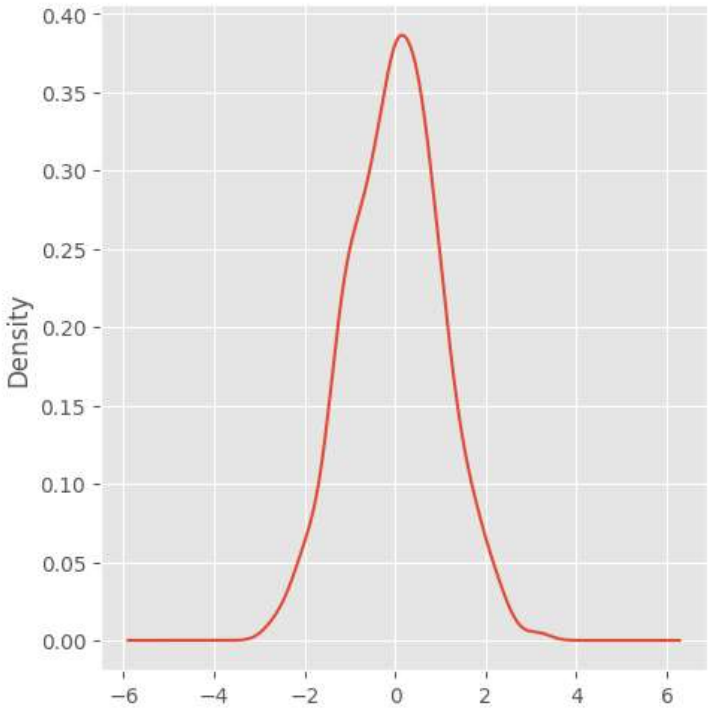
```
import numpy as np
import matplotlib.pyplot as plt
plt.scatter(x=range(-10, 38, 1),
y=range(770, 60, -15)-np.random.randn(48)*40,
            s=200,
            alpha=0.6)
plt.xlabel('Temperature(°C)', size=12)
plt.ylabel('Volume per store', size=12)
plt.show()
```



PRACTICAL 12

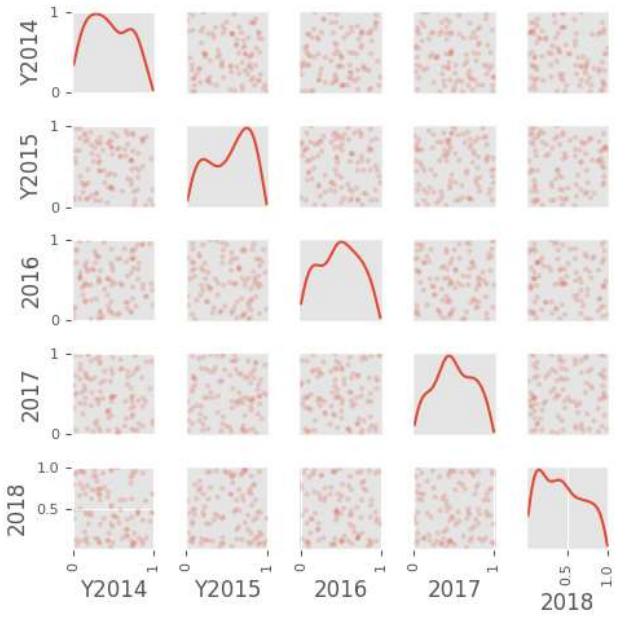
#Kernel Density Estimate

```
import sys
import os
import pandas as pd
import matplotlib as ml
import numpy as np
from matplotlib import pyplot as plt
ml.style.use('ggplot')
figl=plt.figure(figsize=(5, 5))
ser = pd.Series (np.random.randn(1000))
ser.plot(figsize=(5, 5), kind='kde')
sPicNameOut1='/content/kde.png'
plt.savefig(sPicNameOut1,dpi=600)
plt.tight_layout()
plt.show()
```



#Scatter Plot Matrix

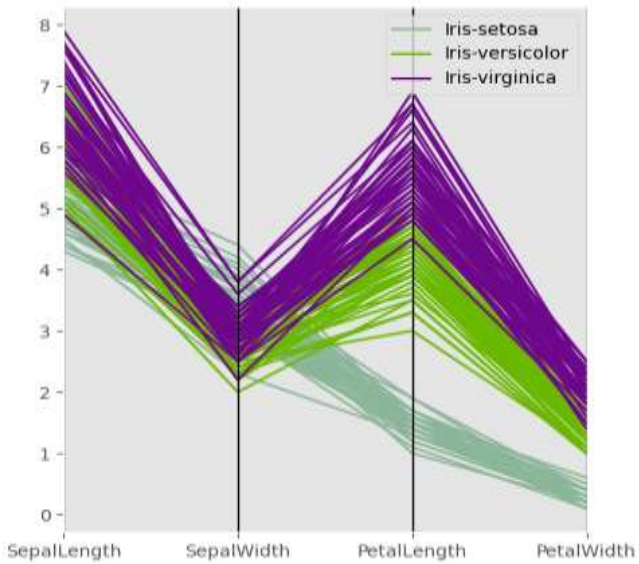
```
import sys
import os
import pandas as pd
import matplotlib as ml
import numpy as np
from matplotlib import pyplot as plt
fig2=plt.figure(figsize=(5, 5))
from pandas.plotting import scatter_matrix
df = pd.DataFrame(np.random.rand(100, 5),
columns=['Y2014', 'Y2015','2016','2017','2018'])
scatter_matrix(df, alpha=0.2,
               figsize=(5,5), diagonal='kde')
sPicNameOut2='/content/scatter_matrix.png'
plt.savefig(sPicNameOut2, dpi=600)
plt.tight_layout()
plt.show()
```



PRACTICAL 12

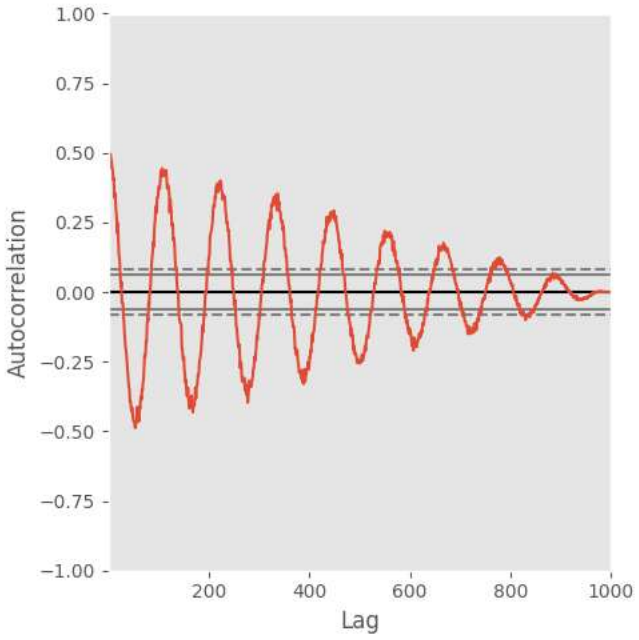
#Parallel Coordinates

```
import sys
import os
import pandas as pd
import matplotlib as ml
import numpy as np
from matplotlib import pyplot as plt
from pandas.plotting import parallel_coordinates
plt.figure(figsize=(5,5))
sDataFile='/content/iris1.csv'
data = pd.read_csv(sDataFile)
parallel_coordinates (data, 'Name')
sPicNameOut2='/content/parallel_coordinates.png'
plt.savefig(spicNameOut2, dpi=600)
plt.tight_layout()
plt.show()
```



#Autocorrelation Plot

```
import sys
import os
import pandas as pd
import matplotlib as ml
import numpy as np
from matplotlib import pyplot as plt
from pandas.plotting import autocorrelation_plot
plt.figure(figsize=(5,5))
data = pd.Series (0.7* np.random.rand(1000) + \
0.3* np.sin(np.linspace(-9* np.pi, 9 * np.pi, num=1000)))
autocorrelation_plot(data)
spicNameOut2='/content/autocorrelation_plot.png'
plt.savefig(sPicNameOut2,dpi=600)
plt.tight_layout()
plt.show()
```



PRACTICAL 13

#BASIC_TEXT_PROCESSING_USING_NLP

```
import nltk
nltk.download('punkt')
from nltk.tokenize import sent_tokenize, word_tokenize
Txt = "Good Day Mr. Vermeulen, \
how are you doing today?\
The weather is great, and Data Science is awesome.\
You are doing well!"
print (Txt, '\n')
print('Identify sentences')
print(sent_tokenize (Txt), '\n')
print('Identify Word')
print (word_tokenize (Txt))
```

Output:

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
Good Day Mr. Vermeulen, how are you doing today?The weather is great, and Data Science is awesome.You are doing well!

Identify sentences
['Good Day Mr. Vermeulen, how are you doing today?The weather is great, and Data Science is awesome.You are doing well!']

Identify Word
['Good', 'Day', 'Mr.', 'Vermeulen', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'weather', 'is', 'great', ',', 'and', 'Data', 'Science', 'is', 'awesome.You', 'are', 'doing', 'well', '!']
```