Name: Shobhit Pathak

Sap Id: 590012807

Subject: EOAIML

Batch: 12

```
import numpy as np

# Create a ID NumPy array with 20 elements (e.g.
, 1 to 20) array = np.arange(l, 21)

print( "Original Array: " )
print(array)

# Extract every second element using
slicing every_second_element array[ : : 2]
print("Every second element:
") print
(every_second_element)
```

```
Original Array:
[ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
Every second element:
[ 1 3 5 7 9 11 13 15 17 19]
```

```
import pandas as pd

# Read the Excel file df = pd. /
content/ Bookl.xlsx")

# Get the number of columns
and rows num columns =
df.shape[l] num rows =
df.shape[0]

print( "Number of columns: "
num_columns) print( "Number of
rows: " , num_rows )
```

```
# Convert each row into a
list of lists data as list of lists
= df.values.tolist()
```

```
! pip install scikit-learn
Requirement already satisfied: numpy>=1.19.5 scipy>=1.6.0 joblib>=1.2.0 scikit-learn in /usr/10ca1/1ib/python3.11/dist-packages (1 . 6. 1) in /usr/10ca1/1ib/python3.11/dist-packages (from scikit-learn) (2.0.2) in
Requirement already satisfied: /usr/10ca1/1ib/python3.11/dist-packages (from scikit-learn)
Requirement already satisfied: (1.16.1) in /usr/10ca1/1ib/python3.11/dist-packages (from scikit-
```

```
Requirementalready learn) (1.5.1) threadp001ct1>=3.1.e in
Requirementalready /usr/10ca1/1ib/python3.11/dist-packages (from scikit-learn) (3.6.0)
```

```
print("\nData as a list of lists: print (data_as_list_of_lists)
```

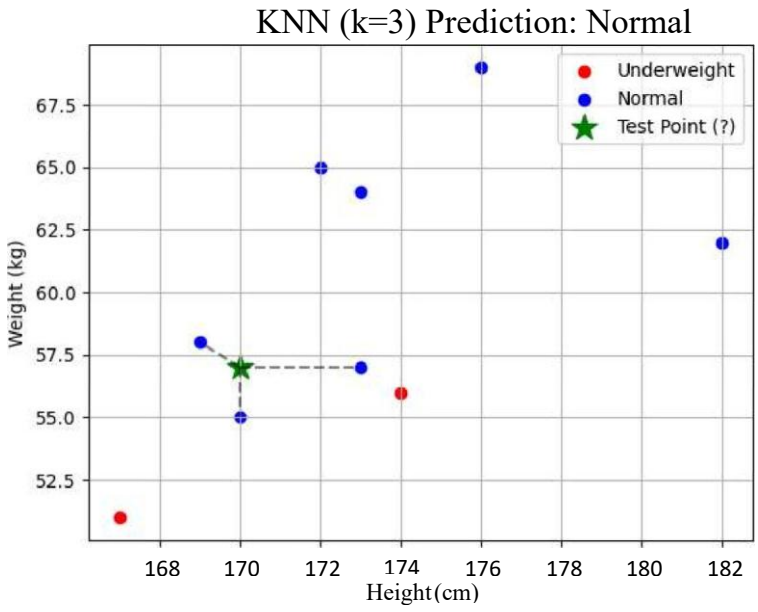
```
Number of columns: 3
Number of rows: 10

Data as a list of lists:
[ [167, 51, ' Underweight '[182, 62, ' Normal[176, 69, 'Normal '[173, 64, ' Normal [172, 65, 'Normal ' ] ' [174, 56, '
Underweight ' ] ,
```

```
import math

def cosine_similarity(vl, v2) :
    dot = sum(a*b for a, b in zip(vl, v2))
    magl = math.sqrt(sum(a*a for a in VI))
    mag2 = math.sqrt(sum(b*b for b in v2))
    return dot / (magl * mag2) if magl and
    mag2 else 0
# Example

print( "Cosine Similarity: " , cosine_similarity (VI, v2))
```

```
Cosine Similarity: 0.9746318461970762

import math

def euclidian_distance(pl, p2) :
   return + (p2[1]

pointl =
point2

print( "Distance: 'euclidian_distance(pointl, point2) )
```

```
Distance: 5.0
import math from
collections import Counter

# Dataset (Height, Weight,
Class) data =
    [167,51,"Underweight
            " ] ,
    [182,62,"Normal " ] '
    [176,69,"Normal " ],
    [173,64,"Normal " ] '
    [172,65,"Normal " ],
    [174,56,"Underweight
            " J ,
    [169,58,"Normal " ],
    [173,57,"Normal " ] '
```

```
    [170,55,"Normal " ] '
    [170,57, "?"]  # Test point
train_data  --  data[:-l]
test_point = data[-1][:2]

# Euclidean distance def
euclidean_distance(pl,
p2) :
    return math.sqrt((p1[0] - p2[0])**2 + (pl[l] - p2[1])**2)

# KNN asumption def
knn_predict(train,  test,
k):
    distances = [(euclidean_distance(test, row[ : row) for row
    in  train] distances.sort(key=lambda  x:  x[Ø]) nearest  =
    distances[:k] prediction = Counter([row[2] for , row in
    nearest]).most_common(l)  [0]  [0]  return  prediction,
    nearest

# Predict with k3

predicted class, nearest_neighbors = knn_predict(train_data, test_point, k)

# Plot points colors -- {"Normal": "blue" ,
"Underweight . "red", " ? "    ' green " }
for height, weight, cls in train_data:
    plt.scatter(height, weight, color=colors[cls], label—cls if cls not in
            plt.gca() .get_legend_handles_labels()[l] else"")
# Plot the test point
PI .scatter(test_point[0], test_point [1], color="green", marker="*", s=200, label="Test Point
(?) ") # Connect nearest neighbors for      neighbor in nearest_neighbors:
    plt.plot( test_point[0], neighbor[0]], [test_point [1], neighbor[l]], ' k , alpha=e.5)
# Labels and title plt.x1abe1("Height (cm) ")
plt.ylabel( "Weight  (kg)  ") plt.tit1e(f"KNN
(k={k})  Prediction:  {predicted_class}"  )
PIt. legend() pit. grid (True) PIt. show( )
```




```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.16.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.5.1)
```

Requirement already satisfied: threadp001ct1>=3.1.e in /usr/10ca1/1ib/python3.11/dist-packages (from scikit-learn) (3.6.0)

```
open( " / content/ Bookl .
xlsx" ) import pandas as pd

df=pd . read_excel( / content/ Bookl . xlsx ' )
import pandas as pd

# Load the data from the Excel file, skipping the
first row df = pd. / content/ Bookl.xlsx" ,
skiprows=l)
# Manually assign the correct column
names df.columns = [ 'Height(cm) ' ,
'Weight(kg) ' Cass ' ]

# Separate the features (x) and the target (y)
        'Weight(kg) ' , ' Height(cm) ' ] ]
Y = df[ 'Cass' ]

# Map 'Normal' to 1 and ' Underweight ' to e in the target
                        variable (y) ' Underweight : 0})

print("Features (X) : 'I )
display(x.head())

print( "\nTarget (Y) : "
) display(y . head())
```

FileNotFoundErrorTraceback (most recent call last) ÉmpAuython-input-1817842996.py. in <cell line:

```
2
3 # Load the data from the Excel file, skipping the first row
    = pd.                          , skiprows=l)
5
6 # Manually assign the correct column names
```

3 frames in get_handle(path_or_buf, mode, encoding, compression, memory_map, is text, errors, storage_options) 880 else:

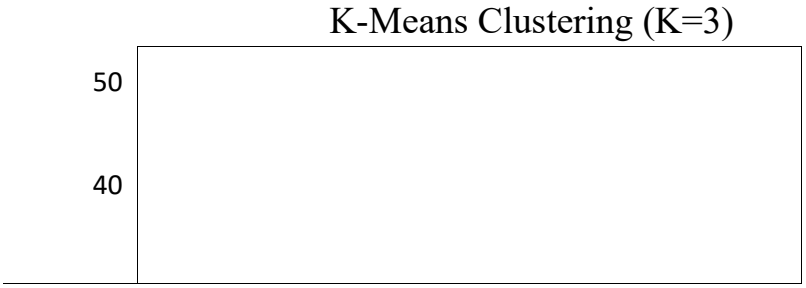
```
881             # Binary mode
882             handle = open(handle, ioargs . mode)
883             handles . append (handle)
884
```

FileNotFoundError: [Errno 2] No such file or directory: ' /content/Book1. xlsx '

Next steps:Explain error

```
import numpy as np
import matplotlib.pyplot
as plt from sklearn.cluster
import KMeans data =
np.array([

  [0, 1],    [1, 3] ,          [6, 22],         [9, 11] , [10, 10] ,
   [0, 40] , [€,41] , [1,42] , [3,45] , [4,46] , [5,47] , [6,48] , [7 , 50] , [8, 44] , [9,49] ,
   [35, 35] , [36, 36], [37 , 40] , [39, 42] , [40, 41] , [45, 50], [50, 50] , [50,45]
```

```
3 kmeans = KMeans(n_c1usters=K,     .fit(data) labels =
kmeans.labels centroids = kmeans.cluster centers plt.scatter(data[ : ,0],
data[ : ,l], c=labels, cmap= ' rainbow' , s=50) centroids[ :,l]
      c='black' , marker=     s-200, label= ' Centroids
PIt .xlabel ( "diml " ) plt . y
label ( "dim2 " ) plt.tit1e(f"K-
Means Clustering (K={K})")
plt . legend ( ) plt . show()
```




```
import pandas as pd import
matplotlib.pyplot as plt from
sklearn.cluster import KMeans df
= pd.          .xlsx")
                'dim2' ]]
kmeans = KMeans(n_c1usters=K, random_state=e, n _ 1 • nit• = 'auto' )
kmeans . fit (X) cluster_assignments kmeans . labels cluster centers =
kmeans. cluster centers plt.scatter(X[ 'diml' ], X[ 'dim2' ],
c=cluster_assignments, s=50, cmap= 'viridis ' ) plt . : , 0] , cluster_centers [
: , 1], c='red' , s=2ee, marker= 'x' , label= 'Centroids plt.tit1e(f"K-Means
Clustering (k={K})") plt.x1abe1("Dimension 1") plt.y1abe1("Dimension
2") plt . legend() plt.grid(True) plt . show()

FileNotFoundError                Traceback (most recent call last)
Limp.Lipython-input-2207411179.p_y. in <cell line:
    2 import matplotlib.pyplot as plt
    3 from sklearn.cluster import KMeans
---- > 4 df = pd . read_excel ( "cluster_data . xlsx" )
                'dim2 ' ] ]
    6 K = 3

                                3 frames
/usr/10ca1/1ib/python3.12/dist-packagesLpandas/io/common .py_ in get_handle(path_or_buf, mode,
encoding, compression, memory_map, is text, errors, storage_options)
   880
   881          # Binary mode
   882          handle      open (handle, ioargs . mode)
   883          handles . append (handle)
   884

FileNotFoundError: [Errno 2] No such file or directory: 'cluster data.xlsx'

Next steps:Explain error


import numpy as np
import matplotlib.pyplot
as PIt
X = np. 1200, 1500, 1800, 2000, 2500, 3000] ) Y =
np. 25, 28, 39, 35, 40, 45])
```

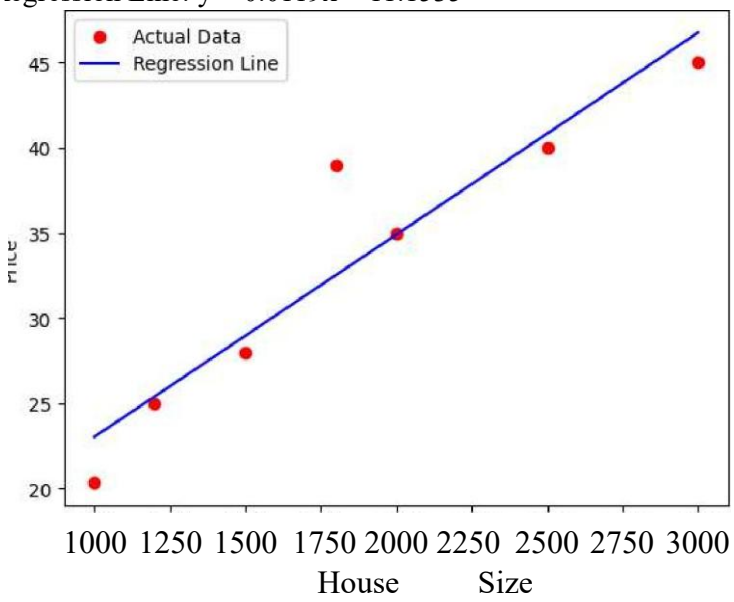
x mean = np.mean(X) Y_mean = np.mean (Y) num = x_mean) * (Y y_mean)) den = np.sum((X x_mean) m = num / den c = y_mean m x mean

print(f"Regression Line: Y

Y_pred

plt.scatter(X, Y, color=' red label='Actua1 Data ') plt.p10t(X, Y_pred, color= 'blue' , label='Regression Line') plt.x1abe1("House Size (sqft)") plt . y label("Price") plt . legend() plt . show()

Regression Line: y = 0.0119x + 11.1535



(sqft) import matplotlib.pyplot as PIt

```
x = [1, 2, 3, 4, 5]
```

y =[10, 60, 30, 40, 50] data = [10, 20, 20, 30, 40, 40, 40, 50] sizes = [25, 35, 20, 20] labels = ['Category A' , 'Category B' , 'Category C' , 'Category D']

fig,ax = plt.subp10ts(2, 3, figsize=(12, 6))

ax[0,0] . plot (x, y, marker='o' label= ' color=' blue' ,Line')

ax[0,0] set_title("Line Graph")

ax[0,e] . legend()

ax[0,1] . bar(x, y, color=' skyblue')

ax[0,1] set_title("Bar Chart")

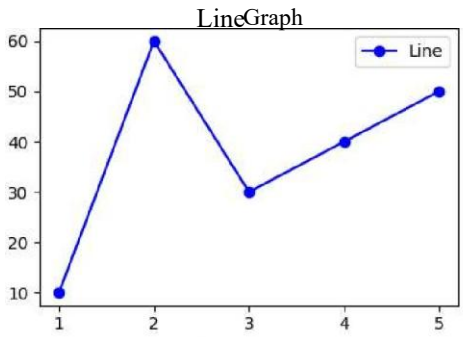
ax[0,2] . hi st (data, bins=5, color= 'orange')

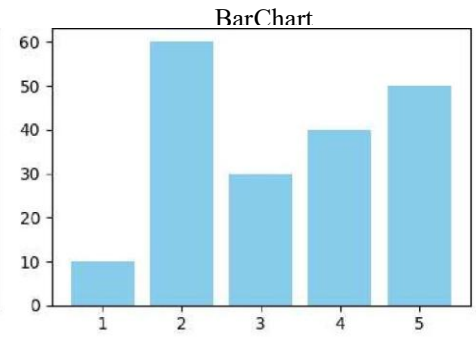
ax[0,2] . set_title("Histogram")

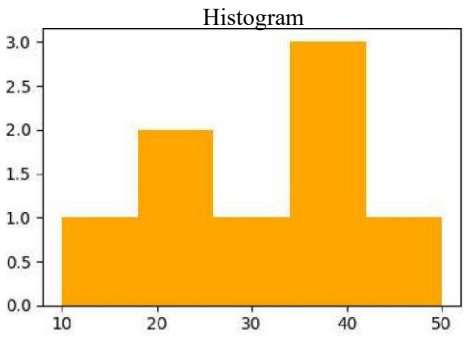
ax[1,0] . scatter (x, Y, color='red' , marker=. x,)

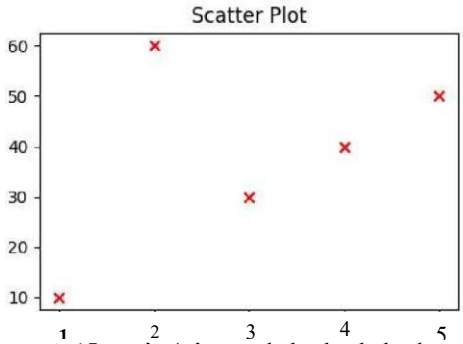
ax e] .set_tit1e("Scatter Plot")

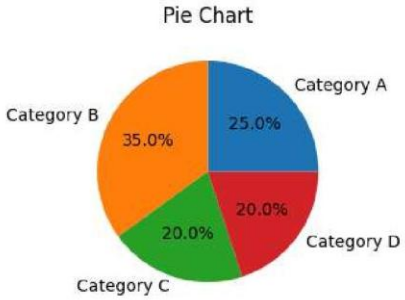
[1,











ax[l,1] . pie(sizes, labels-labels, autopct= '
0/01 . 1 f%%.)
ax[l,1] set_title("Pie Chart")
ax[l,2] . axis(off')
plt .
tight_layout (
) plt . show()

```
import pandas as pd

# Create the
DataFrame data
    'SAP ID' :
    'AGE' : [25, 26, 23, 21, 24.5, 21.5,
    20, 19] , 'MARKS' : [70, 71, 82, 45,
    39, 78, 91, 90], ' FUTURE
    DREAM' :
    'Cyber Security' , 'Data Science"Al/ML Expert'

                                        ,
        ' Full Stack' , 'Dev Ops' , 'Cyber
        Security' ,
        'Data Science' , 'Cloud Expert '

' LABEL' : [ 'Hard' , ' Easy     'Hard' , 'Easy' , '
Easy' , 'Hard ' ,
df = pd.DataFrame(data) encoded df = pd .
get_dummies (df , columns=[ ' LABEL ' ] )
print (encoded_df)
```

```
   SAP ID AGE MARKS FUTURE DREAM
        LABEL_Easy LABEL Hard 1 25.0 70 Cyber
        Security False True
1       2 26.0   71      Data Science   True
         False
2       3 23.0   82      Al/ML Expert   False   True
3       4 21.0   45      Full Stack     True
         False
```

4 5 24.5 39 Dev Ops True
False
5 6 21.5 78 Cyber Security False True
6 7 20.0 91 Data Science True
False
7 8 19.0 90 Cloud Expert
False False

import as 'Easy' , ' Moderate]
numpy
[40, 50, 48, 55, 80]
45, 70,
2, 5, 8, 11]

LABEL Moderate
False
False
False
False
False
False
False
True

```
# Covariance
matrix cov matrix =
np.cov(X, Y)
cov_xy =
cov_matrix[Ø, 1]
# Correlation coefficient
corr_xy = np.corrcoef()(,
Y) [0, 1] print(
"Covariance , cov_xy)
print("Corre1ation -
corr_xy)
```

Covariance = 52.88095238095238
Correlation = e. 9936740547587747

```
import numpy as np X
= np.array([2, 3, 4, 5])
Y = np.array([3, 4, 5,
6]) data =
np.c01umn_stack((X,
Y))

mean = np.mean(data,
axis=e) centered data = data
mean

cov matrix = np. cov (centered_data, rowvar=Fa1se)
```

```
eigvals, eigvecs = np.linalg.eig(cov_matrix)

order = np.argsort(eigvals)[::-1]
principal_axes = eigvecs[:, order]

pc1 = principal_axes[:, 0]
projections = centered_data.dot(pc1)

print("Principal axes (each column):")
print(principal_axes)
print("Projections along first principal axis:")
print(projections)
```

```
Principal axes (each column):
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
Projections along first principal axis:
[-2.12132034 -0.70710678  0.70710678  2.12132034]
```

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA

X = [2, 3, 4, 5]
```