JSPM UNIVERSITY PUNE

Faculty of Science and Technology

School of Computational Sciences

Lab Practical File

FY Master of Computer Application

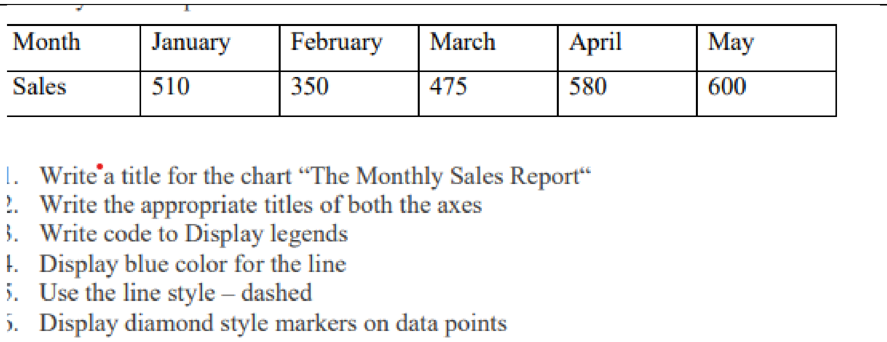
Academic year 2024-25
Semester-I

Course Name: Introduction to Artificial Intelligence and Machine Learning Lab

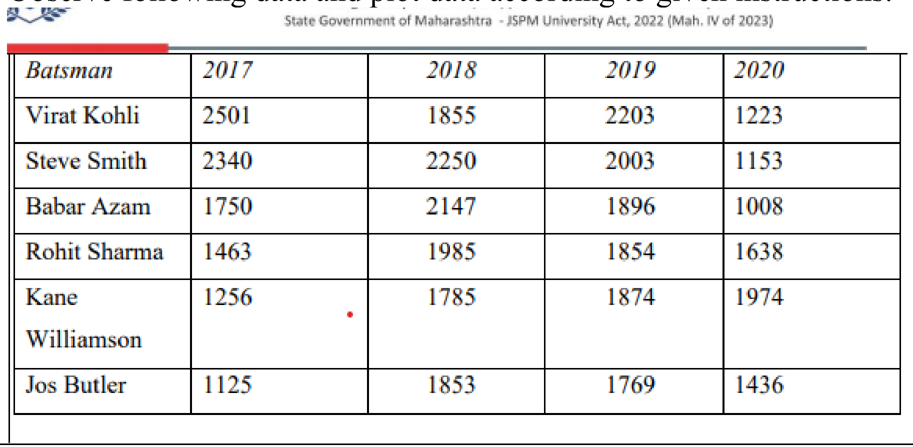
Course Code: 240GCAM41_01

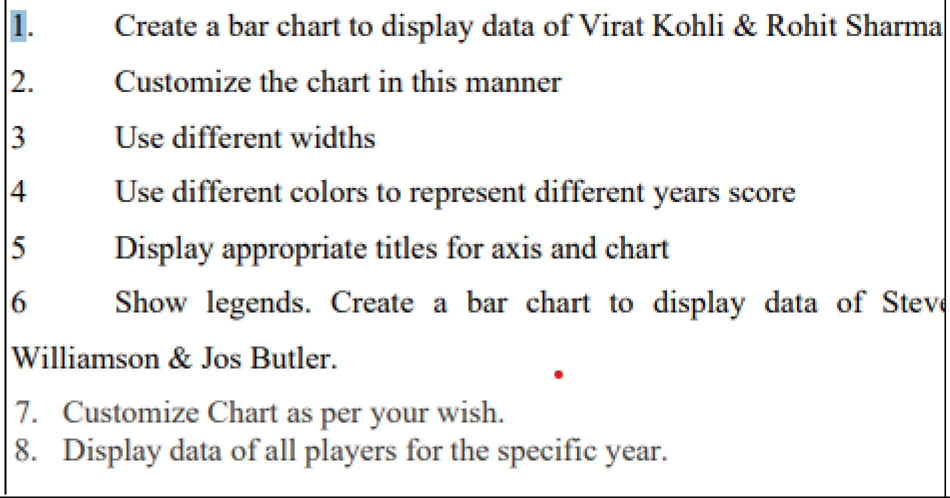
Submit By: **Submit To:**

Index

<table>
<tr><td></td><td>

Month	January	February	March	April	May
Sales	510	350	475	580	600

1. Write a title for the chart "The Monthly Sales Report"
2. Write the appropriate titles of both the axes
3. Write code to Display legends
4. Display blue color for the line
5. Use the line style – dashed
6. Display diamond style markers on data points</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td>Observe following data and plot data according to given instructions:

State Government of Maharashtra - JSPM University Act, 2022 (Mah. IV of 2023)

Batsman	*2017*	*2018*	*2019*	*2020*
Virat Kohli	2501	1855	2203	1223
Steve Smith	2340	2250	2003	1153
Babar Azam	1750	2147	1896	1008
Rohit Sharma	1463	1985	1854	1638
Kane Williamson	1256	1785	1874	1974
Jos Butler	1125	1853	1769	1436
</td><td></td><td>8</td><td></td></tr>
<tr><td>6</td><td>Write a python program to create a 3*3 numpy array with all the elements as per the user choice and print sum of all elements of the array

1. Create a bar chart to display data of Virat Kohli & Rohit Sharma
2. Customize the chart in this manner
3 Use different widths
4 Use different colors to represent different years score
5 Display appropriate titles for axis and chart
6 Show legends. Create a bar chart to display data of Stev
Williamson & Jos Butler.
7. Customize Chart as per your wish.
8. Display data of all players for the specific year.</td><td></td><td>10</td><td></td></tr>
<tr><td>7</td><td>Using LINEAR REGRESSION technique, write a python code to predict Housing price if area is given. Consider a Houseprices.csv file as below:</td><td></td><td></td><td></td></tr>
<tr><td>8</td><td>Using Multi LINEAR REGRESSION technique, write a python code to predict Housing price if area, no_rooms, age are given. Consider a Houseprices_MLR.csv file.</td><td></td><td></td><td></td></tr>
</table>

9	Implement Polynomial Regression algorithm by employee_position.csv (position,level,salary) dataset.			
10	Implement KMeans algorithm by Elbow method using cust_data (income,spending) dataset.			
11	Implement KMeans algorithm by Elbow method using student clustering.csv dataset.			
12	Implement KMeans algorithm by Silhoutte method using. Mall_Customers.csv dataset.			
13	Implement Principal Component Analysis Algorithm by using dataframe			

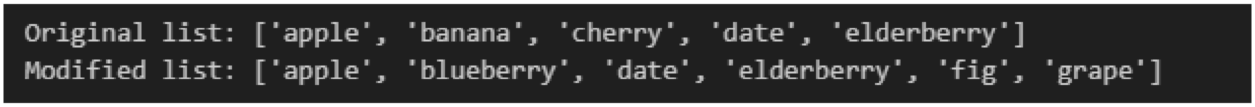
Q1 : Basic Programs –Python Variables and Data Types. Write a python code to create a list of the items. After creating the list write code for the below operations:

- **Print the list of items.**
- **Change any single element in the list.**
- **Append the item at the end of the list.**
- **Insert an element at a specified index.**
- **Remove an element from list.**
- **Sort the elements.**

Ans →

```
 # Create a list of items

my_list = ["apple", "banana", "cherry", "date", "elderberry"]

# Print the list of items

print("Original list:", my_list)

# Change the second element (index 1)

my_list[1] = "blueberry"

# Append an item to the end of the list

my_list.append("fig")

# Insert an item at index 2

my_list.insert(2, "grape")

# Remove the item at index 3

my_list.pop(3)

# Sort the list alphabetically

my_list.sort()

# Print the modified list

print("Modified list:", my_list)
```

Output →



```
Original list: ['apple', 'banana', 'cherry', 'date', 'elderberry']
Modified list: ['apple', 'blueberry', 'date', 'elderberry', 'fig', 'grape']
```

Q2 : Write a python program to create a dictionary. After creating the dictionary write a code to:

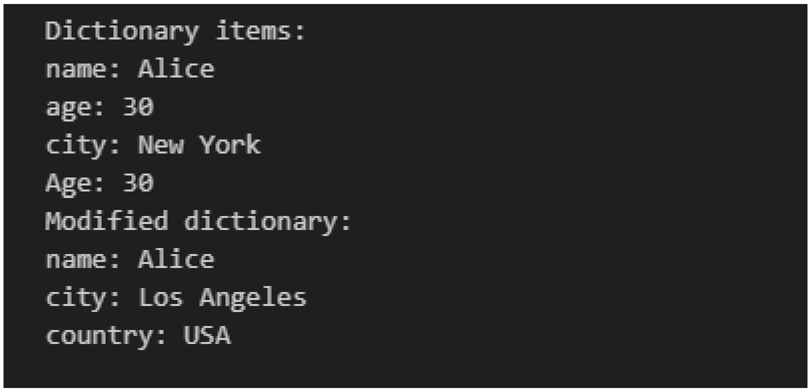
- **Print dictionary items.**
- **Access an element of the dictionary.**
- **Change any element of the dictionary.**
- **Add new element to the dictionary.**
- **Remove any element from dictionary.**

Ans →

```
# Create a dictionary
my_dict = {
   "name": "Alice",
   "age": 30,
   "city": "New York"
}
# Print all items in the dictionary
 print("Dictionary items:")
for key, value in my_dict.items():
   print(f"{key}: {value}")
# Access an element (e.g., age)
age = my_dict["age"]
print("Age:", age)
# Change an element (e.g., city)
my_dict["city"] = "Los Angeles"
# Add a new element (e.g., country)
my_dict["country"] = "USA"
# Remove an element (e.g., age)
del my_dict["age"]


# Print the modified dictionary
```

```
print("Modified dictionary:")

for key, value in my_dict.items():

    print(f"{key}: {value}")
```

Output →



```
Dictionary items:
name: Alice
age: 30
city: New York
Age: 30
Modified dictionary:
name: Alice
city: Los Angeles
country: USA
```

Q3 : Create the following DataFrame Sales containing year wise sales figures for five salespersons in INR. Use the years as column labels, and salesperson names as row labels

Name	2019	2020	2021
Kapil	205	177	189
Kamini	165	175	190
Shikhar	206	157	179
mohini	198	183	169

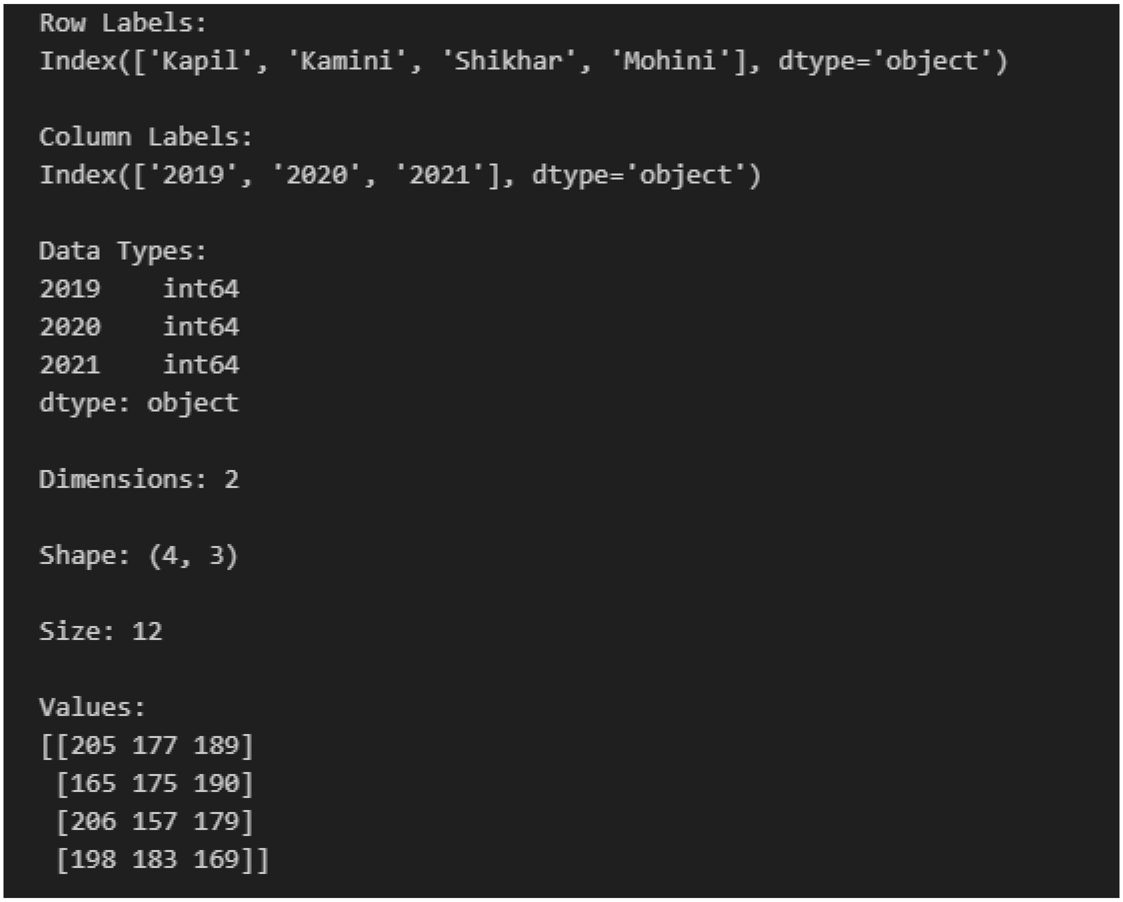
- **create the Data Frame.**
- **Display the row labels of Sales.**
- **Display the column labels of Sales.**
- **Display the data types of each column of Sales.**
- **Display the dimensions, Shape, Size and values of Sales.**

Ans →

```
import pandas as pd
# Create the DataFrame
sales_data = {
   '2019': [205, 165, 206, 198],
   '2020': [177, 175, 157, 183],
   '2021': [189, 190, 179, 169]
}

sales = pd.DataFrame(sales_data, index=['Kapil', 'Kamini', 'Shikhar', 'Mohini'])

# Display row labels
print("Row Labels:")
print(sales.index)

# Display column labels
print("\nColumn Labels:")
print(sales.columns)

# Display data types of each column
print("\nData Types:")
print(sales.dtypes)
# Display dimensions, shape, size, and values
print("\nDimensions:", sales.ndim)
print("\nShape:", sales.shape)
print("\nSize:", sales.size)
print("\nValues:")
```

print(sales.values)

Output →



```
Row Labels:
Index(['Kapil', 'Kamini', 'Shikhar', 'Mohini'], dtype='object')

Column Labels:
Index(['2019', '2020', '2021'], dtype='object')

Data Types:
2019    int64
2020    int64
2021    int64
dtype: object

Dimensions: 2

Shape: (4, 3)

Size: 12

Values:
[[205 177 189]
 [165 175 190]
 [206 157 179]
 [198 183 169]]
```

Q4 : Plot the following data on a line chart and customize the chart according to the below-given instructions:

Monthly Sales Report

Month	January	February	March	April	May
sales	510	350	475	580	600

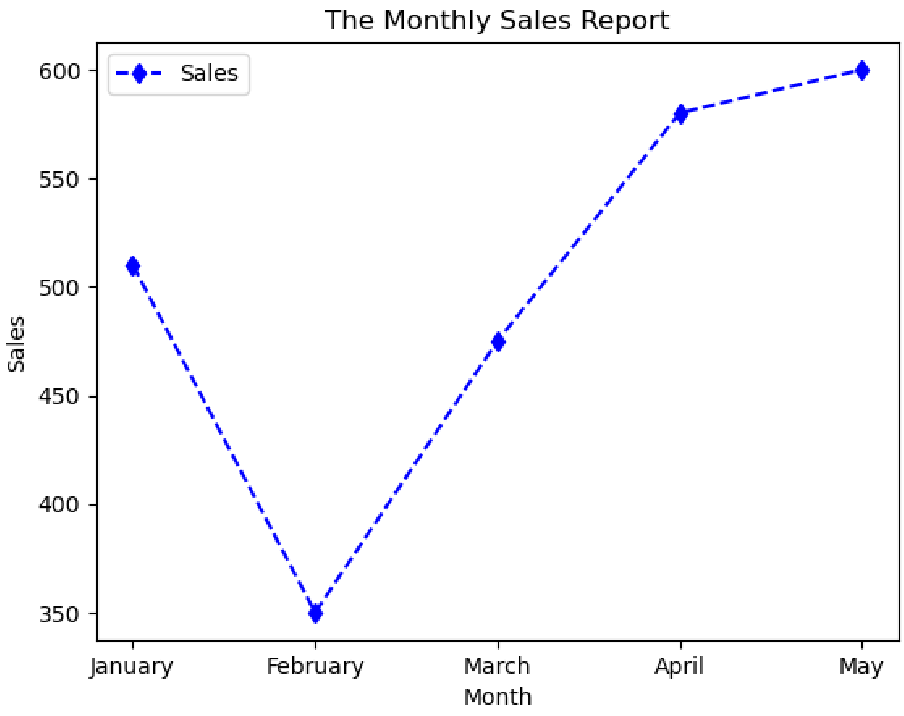
- **write a title for the chart “The Monthly Sales Report “**
- **Write the appropriate titles of both the axes**
- **Write code to Display legends**
- **Display blue color for the line Use the line style – dashed**

- **Display diamond style markers on data points**

Ans →

```
import matplotlib.pyplot as plt

# Data for the chart
months = ['January', 'February', 'March', 'April', 'May']
sales = [510, 350, 475, 580, 600]

# Create the line chart
plt.plot(months, sales, color='blue', linestyle='dashed', marker='d')

# Customize the chart
plt.title('The Monthly Sales Report')
plt.xlabel('Month')
plt.ylabel('Sales')

# Display legend
plt.legend(['Sales'])

# Show the chart
plt.show()
```

Output →



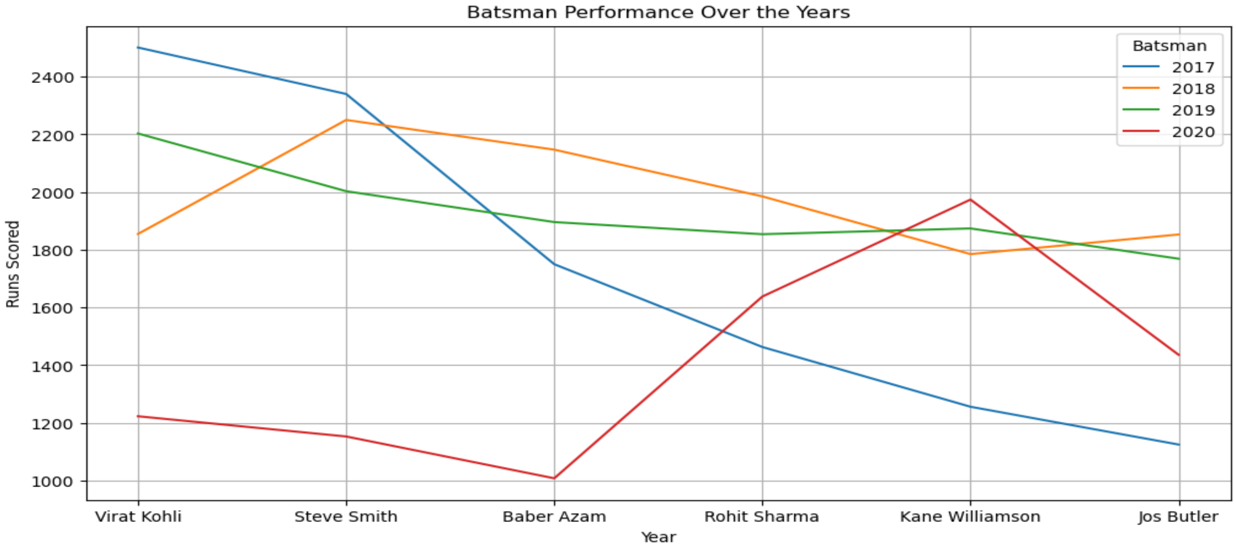
Q5 : Observe following data and plot data according to given instructions:

Batsman	2017	2018	2019	2020
Virat Kohli	2501	1855	2203	1223
Steve Smith	2340	2250	2003	1153
Baber Azam	1750	2147	1896	1008
Rohit Sharma	1463	1989	1854	1638
Kane Williamson	1256	1785	1874	1974
Jos Butler	1125	1853	1769	1436

Ans →

```
import matplotlib.pyplot as plt

import pandas as pd


# Create a DataFrame from the data

data = {'Batsman': ['Virat Kohli', 'Steve Smith', 'Baber Azam', 'Rohit Sharma', 'Kane
Williamson', 'Jos Butler'],
    '2017': [2501, 2340, 1750, 1463, 1256, 1125],
    '2018': [1855, 2250, 2147, 1985, 1785, 1853],
    '2019': [2203, 2003, 1896, 1854, 1874, 1769],
    '2020': [1223, 1153, 1008, 1638, 1974, 1436]}
df = pd.DataFrame(data)

# Set the index to 'Batsman'
df.set_index('Batsman', inplace=True)

# Plot the data
df.plot(figsize=(12, 6))

# Customize the plot
plt.title('Batsman Performance Over the Years')
plt.xlabel('Year')
plt.ylabel('Runs Scored')
plt.legend(title='Batsman')
plt.grid(True)

# Show the plot
plt.show()
```

output →



Q6 : WAP to create a 3*3 numpy array with all the elements as per the user choice and print sum of all elements of the array

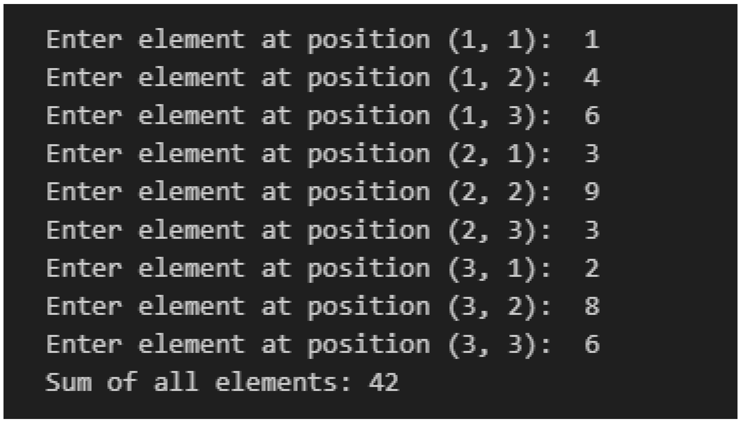
- **create a bar chart to display data of Virat Kohli & Rohit Sharma.**
- **Customize the chart in this manner**
- **Use different widths**
- **Use different colors to represent different years score**
- **Display appropriate titles for axis and chart**
- **Show legends. Create a bar chart to display data of Steve Smith, Kane Williamson & Jos Butler**
- **Customize Chart as per your wish.**
- **Display data of all players for the specific year**

Ans →

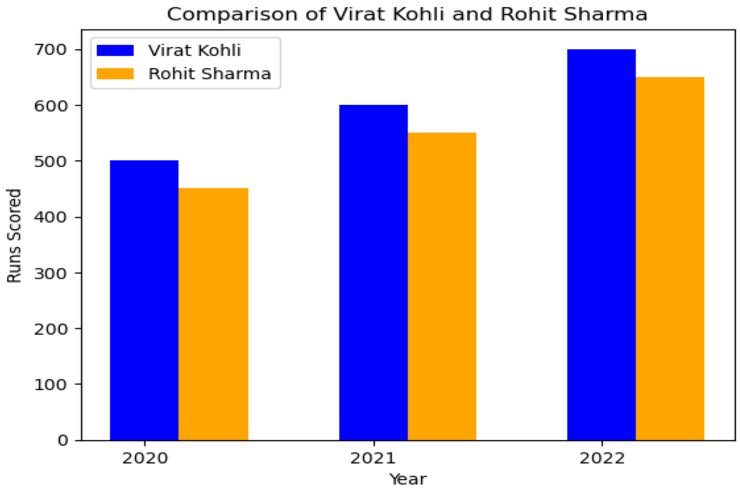
```
import numpy as np
import matplotlib.pyplot as plt

# Create a 3x3 numpy array with user input
array = np.zeros((3, 3), dtype=int)
for i in range(3):
    for j in range(3):
        array[i][j] = int(input(f"Enter element at position ({i+1}, {j+1}): "))

# Calculate and print the sum of all elements
array_sum = np.sum(array)
print("Sum of all elements:", array_sum)

# Data for Virat Kohli and Rohit Sharma
kohli_scores = [500, 600, 700]
rohit_scores = [450, 550, 650]
years = [2020, 2021, 2022]

# Create a bar chart
plt.bar(years, kohli_scores, width=0.3, label='Virat Kohli', color='blue')
plt.bar(np.array(years) + 0.3, rohit_scores, width=0.3, label='Rohit Sharma', color='orange')

# Customize the chart
plt.xlabel('Year')
plt.ylabel('Runs Scored')
plt.title('Comparison of Virat Kohli and Rohit Sharma')
plt.xticks(years)
plt.legend()
plt.show()

# Data for Steve Smith, Kane Williamson, and Jos Buttler
smith_scores = [400, 500, 600]
```

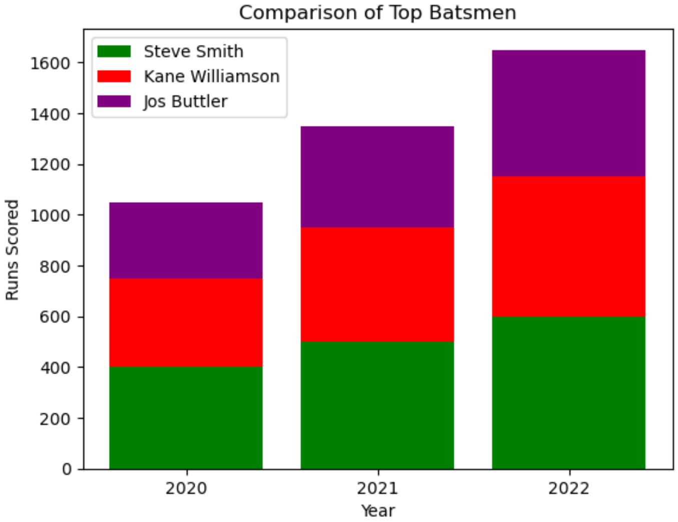
```
williamson_scores = [350, 450, 550]
buttler_scores = [300, 400, 500]

# Create a bar chart
plt.bar(years, smith_scores, label='Steve Smith', color='green')
plt.bar(years, williamson_scores, bottom=smith_scores, label='Kane Williamson', color='red')
plt.bar(years, buttler_scores, bottom=[x+y for x, y in zip(smith_scores, williamson_scores)], label='Jos Buttler', color='purple')

# Customize the chart
plt.xlabel('Year')
plt.ylabel('Runs Scored')
plt.title('Comparison of Top Batsmen')
plt.xticks(years)
plt.legend()
plt.show()

# Display data for a specific year
year_to_display = int(input("Enter the year to display: "))
print(f"Scores for {year_to_display}:")
print("Virat Kohli:", kohli_scores[year_to_display - 2020])
print("Rohit Sharma:", rohit_scores[year_to_display - 2020])
print("Steve Smith:", smith_scores[year_to_display - 2020])
print("Kane Williamson:", williamson_scores[year_to_display - 2020])
print("Jos Buttler:", buttler_scores[year_to_display - 2020])
```

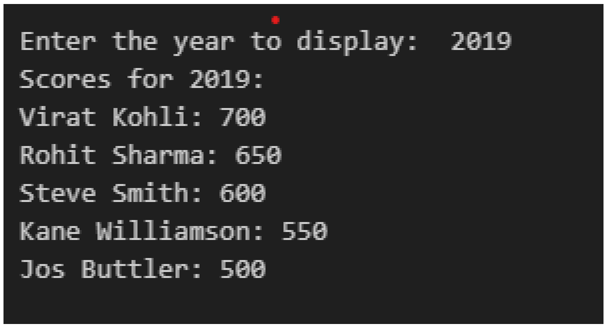
Output →



```
Enter element at position (1, 1):  1
Enter element at position (1, 2):  4
Enter element at position (1, 3):  6
Enter element at position (2, 1):  3
Enter element at position (2, 2):  9
Enter element at position (2, 3):  3
Enter element at position (3, 1):  2
Enter element at position (3, 2):  8
Enter element at position (3, 3):  6
Sum of all elements: 42
```







```
Enter the year to display:  2019
Scores for 2019:
Virat Kohli: 700
Rohit Sharma: 650
Steve Smith: 600
Kane Williamson: 550
Jos Buttler: 500
```

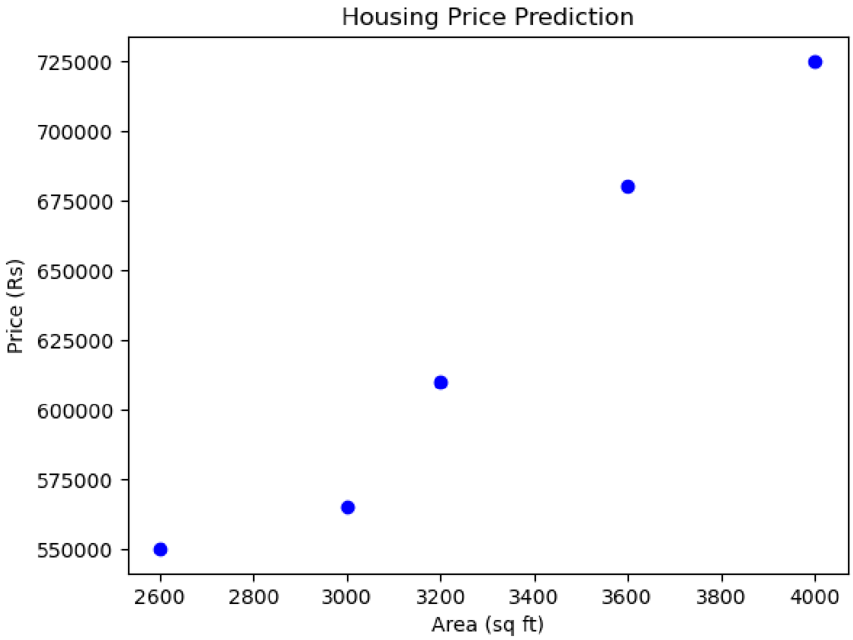
Q7 : Using LINEAR REGRESSION technique, write a python code to predict Housing prices if area is given. Consider a Houseprices.csv file as below:

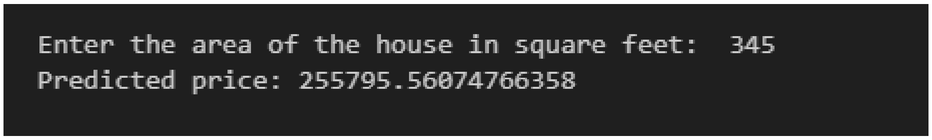
Ans →

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Read the data from the CSV file
data = pd.read_csv('House_Price.csv')

# Extract features (area) and target variable (price)
X = data['Area'].values.reshape(-1, 1)  # Reshape to 2D array
y = data['Price'].values

# Split the data 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)
```

```
# Create a linear regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error:", mse)

# Visualize the  data and the regression line
plt.scatter(X, y, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.xlabel('Area (sq ft)')
plt.ylabel('Price (Rs)')
plt.title('Housing Price Prediction')
plt.show()

# Predict the price of a house with a given area
area = int(input("Enter the area of the house in square feet: "))
predicted_price = model.predict([[area]])
print("Predicted price:", predicted_price[0])
```

Output →





```
Enter the area of the house in square feet:  345
Predicted price: 255795.56074766358
```

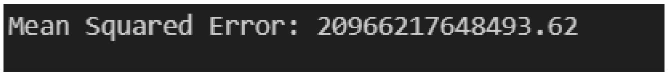
Q8 : Using Multi LINEAR REGRESSION technique, write a python code to predict Housing price if area, no_rooms, age are given. Consider a Houseprices_MLR.csv file.

Ans →

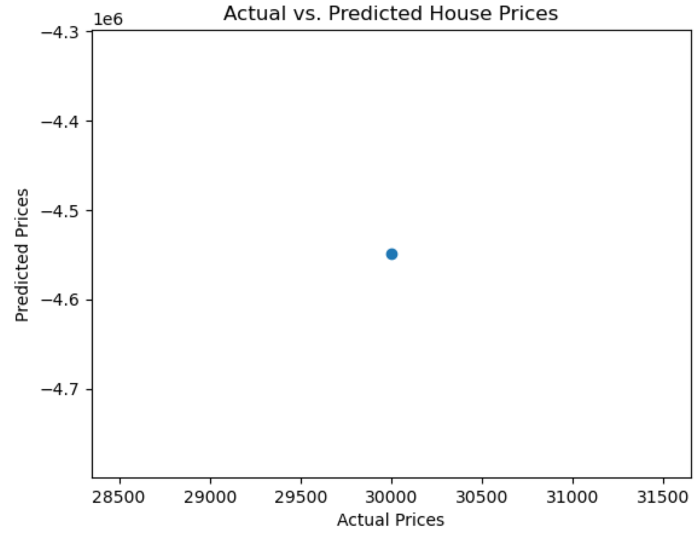
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load the data from the CSV file
data = pd.read_csv("Houseprices_MLR.csv")

# Define the features (independent variables) and the target variable (dependent variable)
```

```
X = data[['area', 'no_rooms', 'age']]
y = data['price']

# Split the data 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)

# Create a linear regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)

# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Visualize the results (optional)
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs. Predicted House Prices")
plt.show()
```

Output →



```
Mean Squared Error: 20966217648493.62
```



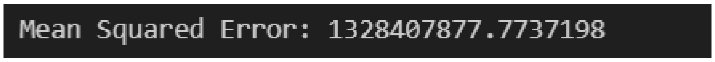
Q9 : Implement Polynomial Regression algorithm by employee_position.csv (position,level,salary) dataset.

Ans →

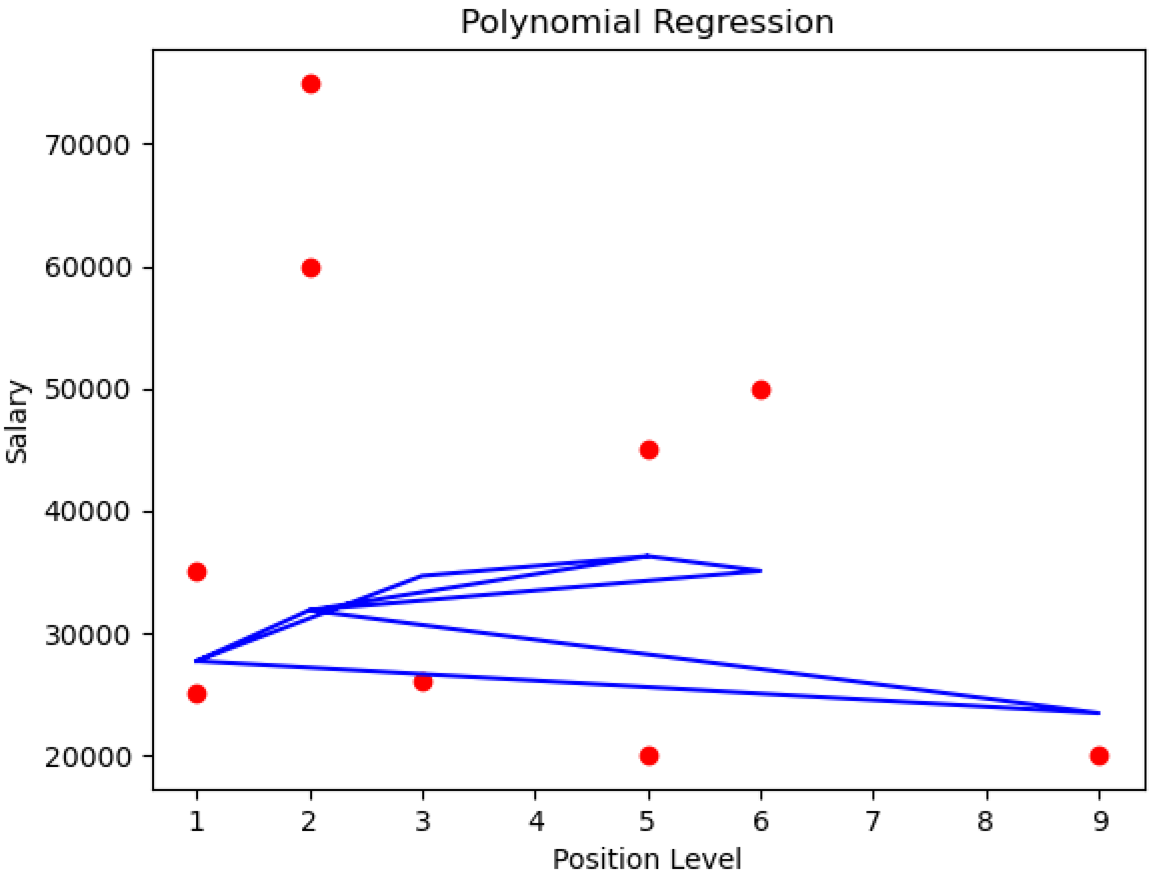
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load the data
data = pd.read_csv('employee_position.csv')

# Prepare the data
X = data[['level']]
y = data['salary']

# Create polynomial features
```

```
polynomial_features = PolynomialFeatures(degree=2)  # Adjust the degree as needed
X_poly = polynomial_features.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2, random_state=0)

# Create and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Visualize the results
plt.scatter(X, y, color='red')
plt.plot(X, model.predict(X_poly), color='blue')
plt.title('Polynomial Regression')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.show()
```

Output →



```
Mean Squared Error: 1328407877.7737198
```



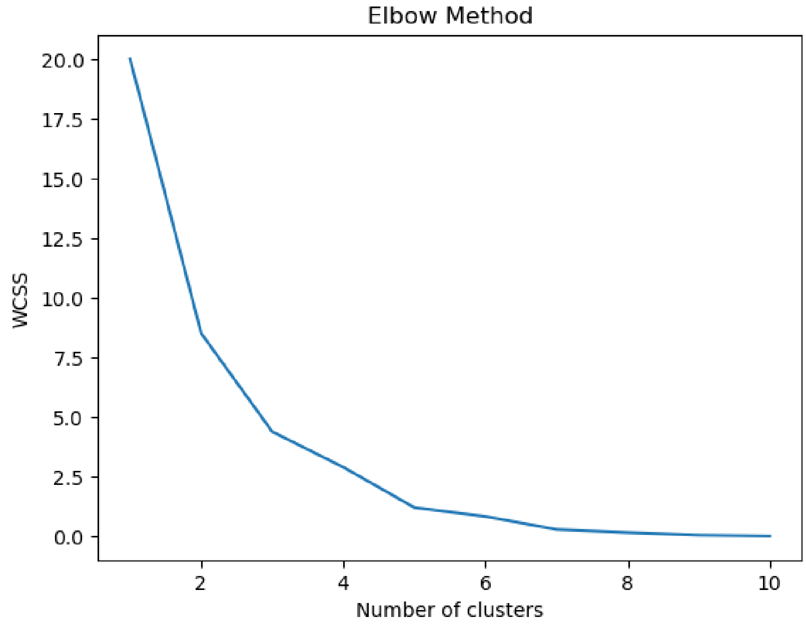
Q10 : Implement KMeans algorithm by Elbow method using cust_data (income,spending) dataset.

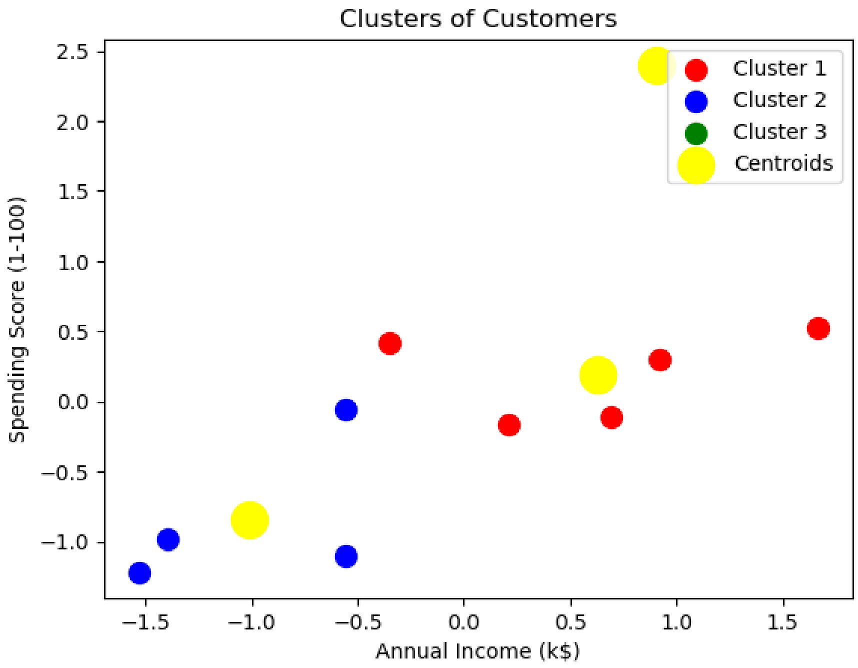
Ans →

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler


# Load the data

data = pd.read_csv('cust_data.csv')
```

```
# Preprocess the data
X = data[['income', 'spending']]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Determine the optimal number of clusters using the Elbow Method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Based on the Elbow Method, choose the optimal number of clusters (e.g., 3)
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
y_kmeans = kmeans.fit_predict(X_scaled)

# Visualize the clusters
plt.scatter(X_scaled[y_kmeans == 0, 0], X_scaled[y_kmeans == 0, 1], s=100, c='red', label='Cluster 1')
plt.scatter(X_scaled[y_kmeans == 1, 0], X_scaled[y_kmeans == 1, 1], s=100, c='blue', label='Cluster 2')

plt.scatter(X_scaled[y_kmeans == 2, 0], X_scaled[y_kmeans == 2, 1], s=100, c='green', label='Cluster 3')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='yellow', label='Centroids')

plt.title('Clusters of Customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()
```

Output →





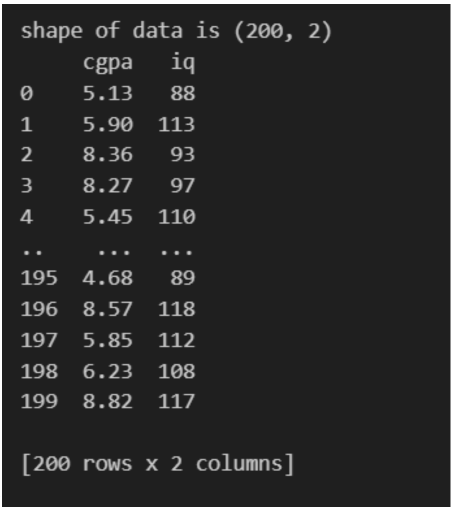
Q11 : Implement KMeans algorithm by Elbow method using student clustering.csv dataset.

Ans→

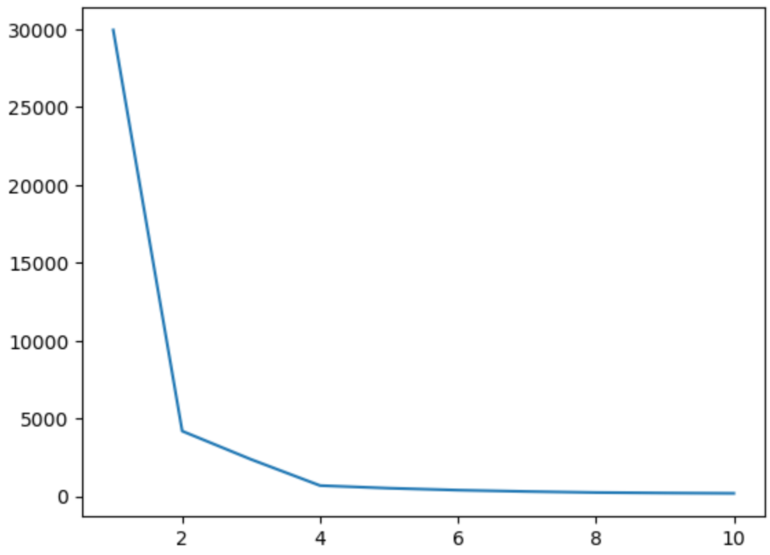
```
 import pandas as pd
df= pd.read_csv('student_clustering.csv')
print("shape of data is", df.shape)
print(df)
from sklearn.cluster import KMeans
wcss= []
for i in range (1,11):
   km=KMeans(n_clusters=i)
   km.fit_predict(df)
   wcss.append(km.inertia_)
import matplotlib.pyplot as plt
plt.plot(range(1,11),wcss)
```

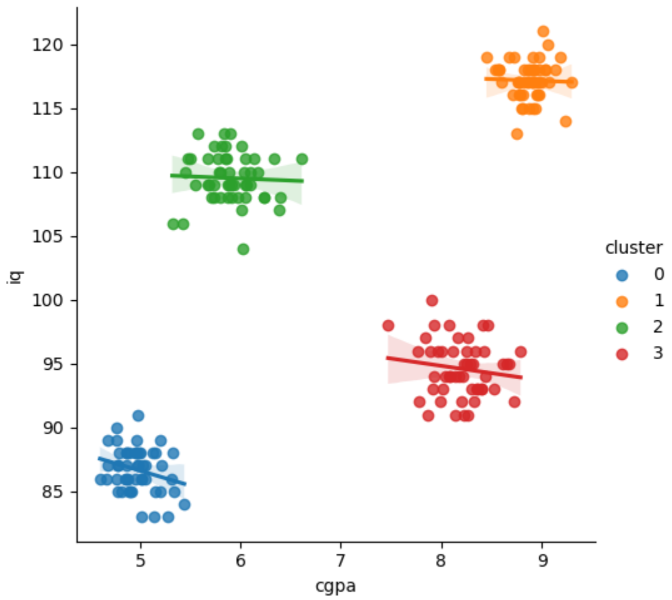
```
from sklearn.cluster import KMeans

km=KMeans(n_clusters=4)

km.fit(df)

pred=km.predict(df)

pred

df['cluster']=pd.DataFrame(pred,columns=['cluster'])

df

import seaborn as seb

seb.lmplot(x='cgpa',y='iq',data=df,hue='cluster')

plt.show()
```

Output →



```
shape of data is (200, 2)
     cgpa   iq
0    5.13   88
1    5.90  113
2    8.36   93
3    8.27   97
4    5.45  110
..    ...  ...
195  4.68   89
196  8.57  118
197  5.85  112
198  6.23  108
199  8.82  117

[200 rows x 2 columns]
```





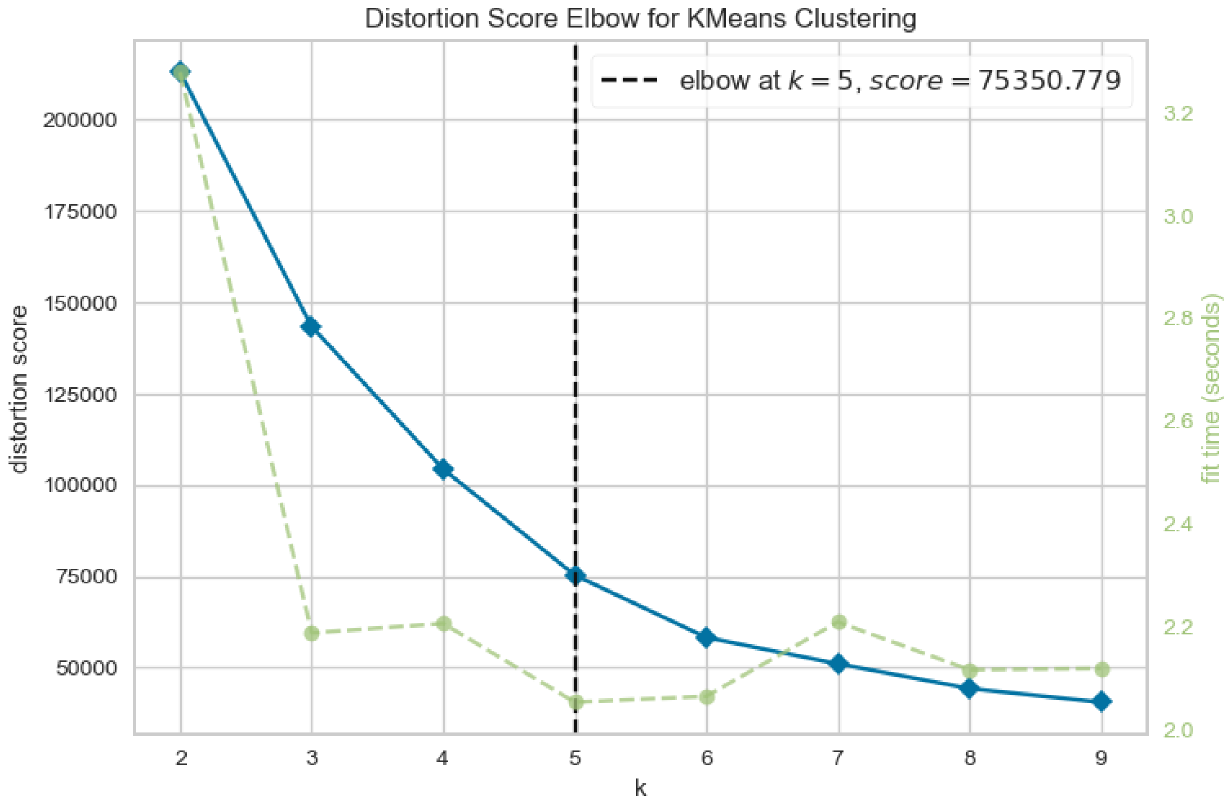
Q12 : Implement KMeans algorithm by Silhoutte method using. Mall_Customers.csv dataset.

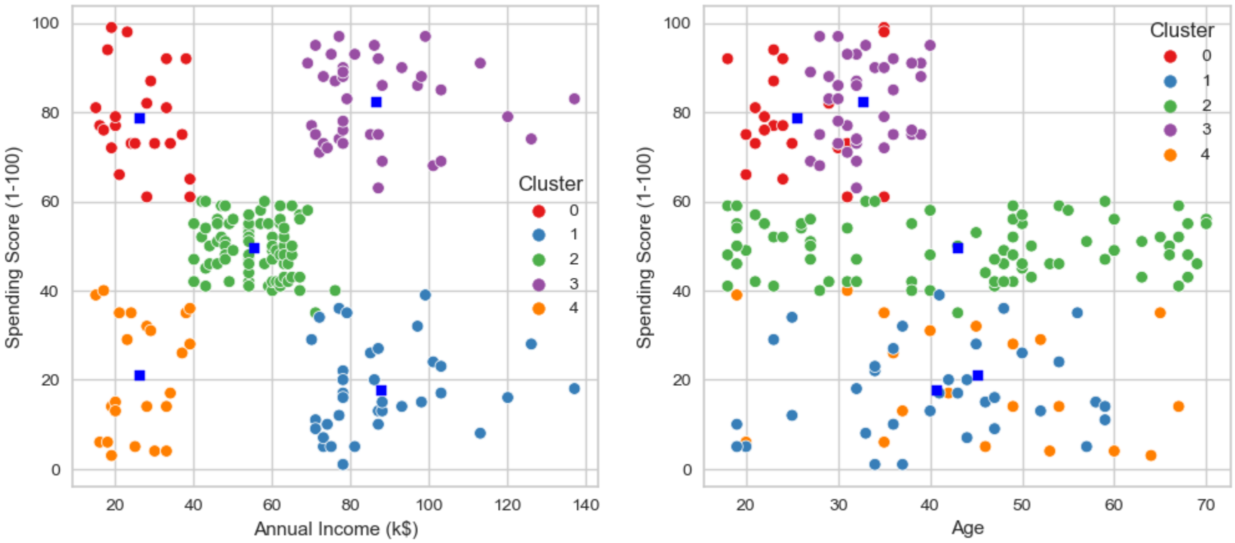
Ans →

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

from sklearn.cluster import KMeans
from scipy import stats
mall_data = pd.read_csv('Mall_Customers.csv')
mall_data.head()
X_numerics = mall_data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']] # subset with numeric variables only

from yellowbrick.cluster import KElbowVisualizer

model = KMeans(random_state=1)
visualizer = KElbowVisualizer(model, k=(2,10))

visualizer.fit(X_numerics)
visualizer.show()
plt.show()

KM_5_clusters = KMeans(n_clusters=5, init='k-means++').fit(X_numerics) # initialise and fit K-Means model

KM5_clustered = X_numerics.copy()
KM5_clustered.loc[:,'Cluster'] = KM_5_clusters.labels_ # append labels to points
fig1, (axes) = plt.subplots(1,2,figsize=(12,5))

scat_1 = sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)', data=KM5_clustered,
            hue='Cluster', ax=axes[0], palette='Set1', legend='full')

sns.scatterplot(x='Age', y='Spending Score (1-100)', data=KM5_clustered,
            hue='Cluster', palette='Set1', ax=axes[1], legend='full')
```

```
axes[0].scatter(KM_5_clusters.cluster_centers_[:,1],KM_5_clusters.cluster_centers_[:,2],
marker='s', s=40, c="blue")

axes[1].scatter(KM_5_clusters.cluster_centers_[:,0],KM_5_clusters.cluster_centers_[:,2],
marker='s', s=40, c="blue")

plt.show()
```

Output →





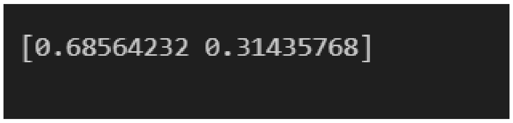
Q13 : Implement Principal Component Analysis Algorithm by using dataframe

Ans →

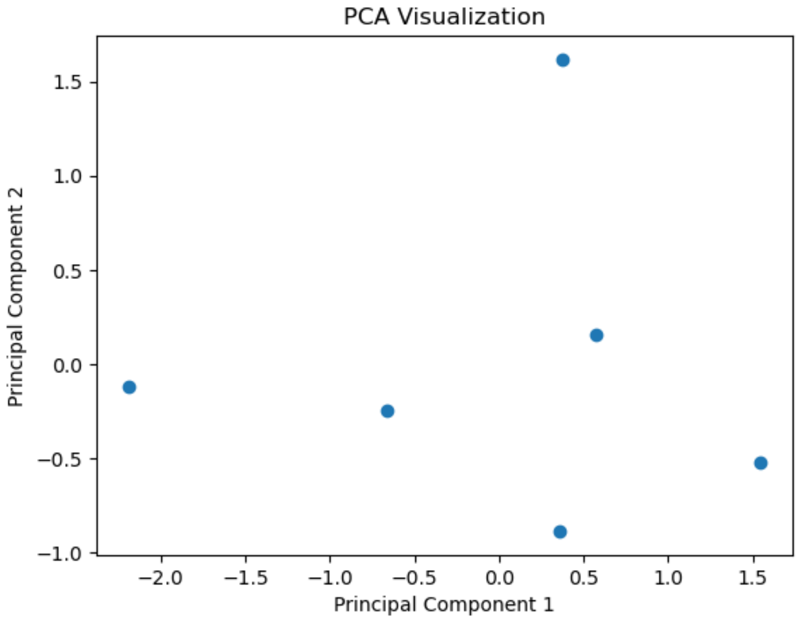
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Load the data
data = pd.read_csv('fileee.csv')

# Preprocess the data
X = data.drop('IQ', axis=1)  # Replace 'target_column' with your target variable
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Apply PCA
pca = PCA(n_components=2)  # Adjust the number of components as needed
X_pca = pca.fit_transform(X_scaled)

# Create a new DataFrame with the PCA components
pca_df = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])

# Visualize the data in the reduced dimension
plt.scatter(pca_df['PC1'], pca_df['PC2'])
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Visualization')
plt.show()

# Print the explained variance ratio
print(pca.explained_variance_ratio_)
```

Output →



```
[0.68564232 0.31435768]
```


PCA Visualization
Principal Component 2
1.5
1.0
0.5
0.0
−0.5
−1.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Principal Component 1