DAY 6 LAB EXPERIMENTS

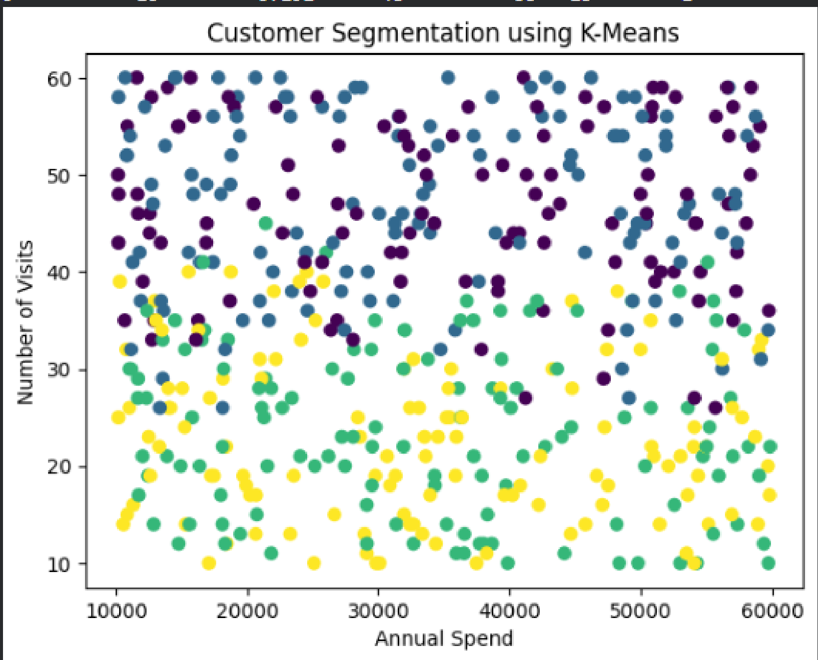
Name: L Aravind

Reg No : 192424080

EXP_ 31

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

df = pd.read_csv("customer_segmentation_data_500.csv")

features = df[['Annual_Spend', 'Visits', 'Time_Spent', 'Age']]

scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

kmeans = KMeans(n_clusters=4, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_features)

print(df.head(10))

plt.figure()
plt.scatter(
    df['Annual_Spend'],
    df['Visits'],
    c=df['Cluster']
)
plt.xlabel("Annual Spend")
plt.ylabel("Number of Visits")
plt.title("Customer Segmentation using K-Means")
plt.show()

# Step 8: Analyze cluster characteristics
cluster_summary = df.groupby('Cluster').mean()
print("\nCluster Summary:")
print(cluster_summary)
```

	Customer_ID	Annual_Spend	Visits	Time_Spent	Age	Cluster
0	1	25795	39	129	46	3
1	2	10860	55	115	49	0
2	3	48158	14	63	41	2
3	4	54732	21	37	31	2
4	5	21284	25	69	31	2
5	6	16265	35	112	55	0
6	7	26850	35	71	54	0
7	8	47194	57	70	44	0
8	9	31962	30	130	42	3
9	10	57191	48	35	25	1



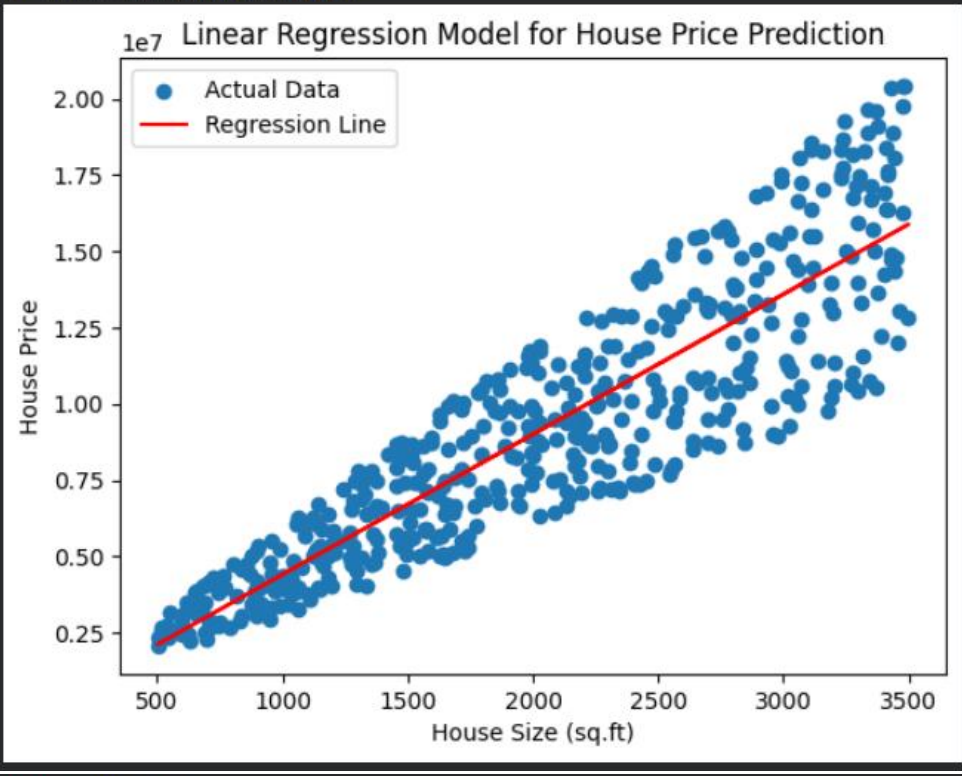
Cluster Summary:

Cluster	Customer_ID	Annual_Spend	Visits	Time_Spent	Age
0	226.050420	36054.109244	46.537815	85.159664	48.630252
1	267.174242	33028.348485	47.196970	73.106061	28.954545
2	253.671875	33857.312500	23.718750	51.609375	40.062500
3	253.000000	33107.793388	22.702479	108.173554	37.280992

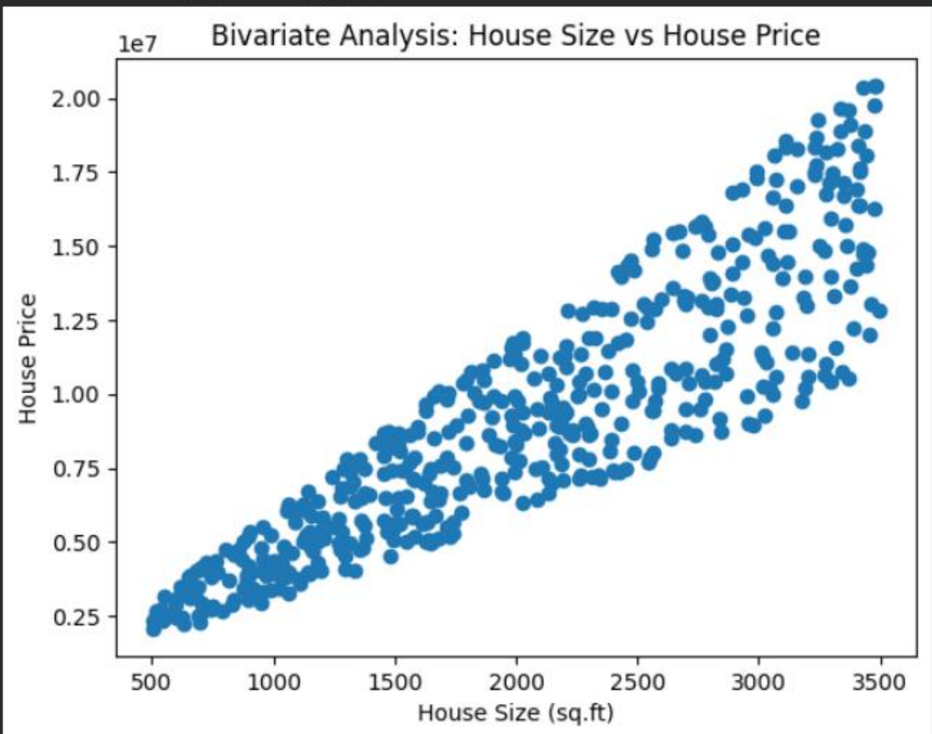
EXP_32

```python
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, r2_score

df = pd.read_csv("house_price_dataset_500.csv")

print("Dataset Preview:")
print(df.head())

plt.figure()
plt.scatter(df['House_Size_sqft'], df['House_Price'])
plt.xlabel("House Size (sq.ft)")
plt.ylabel("House Price")
plt.title("Bivariate Analysis: House Size vs House Price")
plt.show()

X = df[['House_Size_sqft']]   # Independent variable
y = df['House_Price']         # Dependent variable

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\nModel Evaluation:")
print("Mean Squared Error (MSE):", mse)
print("R² Score:", r2)
plt.figure()
plt.scatter(X, y, label="Actual Data")
plt.plot(X, model.predict(X), color='red', label="Regression Line")
plt.xlabel("House Size (sq.ft)")
plt.ylabel("House Price")
plt.title("Linear Regression Model for House Price Prediction")
plt.legend()
plt.show()
```

```
Model Evaluation:
Mean Squared Error (MSE): 3815179089697.245
R² Score: 0.7897057805416927
```




```
Dataset Preview:
   House_Size_sqft  House_Price
0             1360      6502160
1             1794     10493106
2             1630      9468670
3             1595      8754955
4             2138      9901078
```



EXP_33

Files

- ..
- sample_data
- car_price_dataset_500.csv
- customer_segmentation_data_5...
- house_price_dataset_500.csv

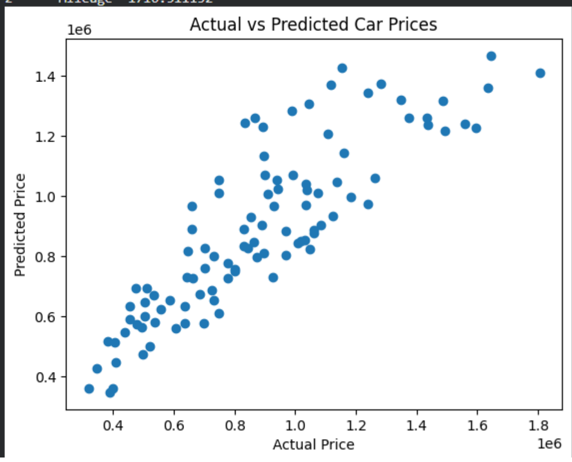
[3]

```
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, r2_score
df = pd.read_csv("car_price_dataset_500.csv")

print(df.head())
X = df[['Engine_Size', 'Horsepower', 'Mileage']]
y = df['Price']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))
importance = pd.DataFrame({
    "Feature": X.columns,
    "Coefficient": model.coef_
})

print("\nFeature Importance:")
print(importance)
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted Car Prices")
plt.show()
```

Files

- ..
- sample_data
- car_price_dataset_500.csv
- customer_segmentation_data_5...
- house_price_dataset_500.csv

```
   Engine_Size  Horsepower  Mileage    Price
0         3974         185       10  1635358
1         1660         177       24   646392
2         2094         107       11   662757
3         1930         148       23   629294
4         1895         296       17   641504
Mean Squared Error (MSE): 27869509320.7804
R2 Score: 0.7444708415577418

Feature Importance:
       Feature  Coefficient
0  Engine_Size   301.540411
1   Horsepower   803.342570
2      Mileage -1716.511132
```



Disk 86.55 GB available

EXP_34

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
df = pd.read_csv("medical_knn_dataset_500.csv")

print(df.head())
X = df[['Age', 'Gender', 'Blood_Pressure', 'Cholesterol']]
y = df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, pos_label="Good"))
print("Recall:", recall_score(y_test, y_pred, pos_label="Good"))
print("F1 Score:", f1_score(y_test, y_pred, pos_label="Good"))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
results = pd.DataFrame({
    "Actual": y_test.values,
    "Predicted": y_pred
})

print("\nPrediction Results:")
print(results.head(10))
```

..
sample_data
car_price_dataset_500.csv
customer_segmentation_data_5...
house_price_dataset_500.csv
medical_knn_dataset_500.csv

```
   Age  Gender  Blood_Pressure  Cholesterol Outcome
0   58       1             166          259     Bad
1   71       0             122          273     Bad
2   48       0              90          236     Bad
3   34       1             110          204    Good
4   62       0             144          181     Bad

Accuracy: 0.97
Precision: 0.90625
Recall: 1.0
F1 Score: 0.9508196721311475

Classification Report:
              precision    recall  f1-score   support

         Bad       1.00      0.96      0.98        71
        Good       0.91      1.00      0.95        29

    accuracy                           0.97       100
   macro avg       0.95      0.98      0.96       100
weighted avg       0.97      0.97      0.97       100


Prediction Results:
  Actual Predicted
0    Bad       Bad
1    Bad       Bad
2    Bad       Bad
3   Good      Good
4   Good      Good
5   Good      Good
6    Bad       Bad
7    Bad       Bad
8   Good      Good
9   Good      Good
```

EXP_35

..
sample_data
car_price_dataset_500.csv
customer_segmentation_data_5...
house_price_dataset_500.csv
medical_knn_dataset_500.csv
retail_kmeans_dataset_500.csv

```
7    Bad      Bad
8    Good     Good
9    Good     Good
```

[5]

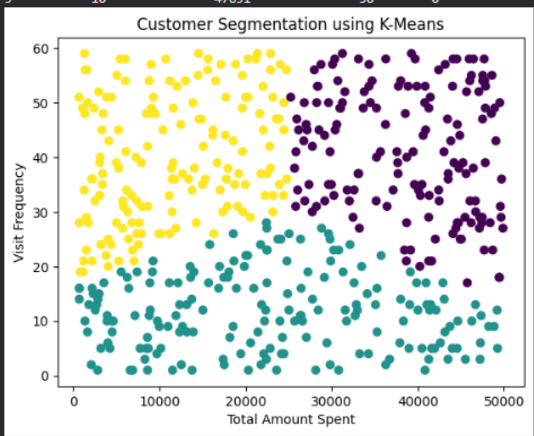
```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

df = pd.read_csv("retail_kmeans_dataset_500.csv")

print(df.head())
X = df[['Total_Amount_Spent', 'Visit_Frequency']]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)
print("\nClustered Customers:")
print(df.head(10))
plt.scatter(
    df['Total_Amount_Spent'],
    df['Visit_Frequency'],
    c=df['Cluster']
)
plt.xlabel("Total Amount Spent")
plt.ylabel("Visit Frequency")
plt.title("Customer Segmentation using K-Means")
plt.show()
cluster_summary = df.groupby('Cluster').mean()
print("\nCluster Summary:")
print(cluster_summary)
```

Files

..
sample_data
car_price_dataset_500.csv
customer_segmentation_data_5...
house_price_dataset_500.csv
medical_knn_dataset_500.csv
retail_kmeans_dataset_500.csv

Disk 86.55 GB available

```
   Customer_ID  Total_Amount_Spent  Visit_Frequency
0            1               16295               26
1            2                1360               48
2            3               38658               21
3            4               45232               39
4            5               11784               36

Clustered Customers:
   Customer_ID  Total_Amount_Spent  Visit_Frequency  Cluster
0            1               16295               26        2
1            2                1360               48        2
2            3               38658               21        0
3            4               45232               39        0
4            5               11784               36        2
5            6                6765               33        2
6            7               17350               30        2
7            8               37694               37        0
8            9               22462               23        1
9           10               47691               58        0
```




```
Cluster Summary:
         Customer_ID  Total_Amount_Spent  Visit_Frequency
Cluster
0         254.288462        38809.493590        41.128205
1         263.924324        22911.340541        11.864865
2         231.163522        11562.220126        40.157233
```

EXP_36

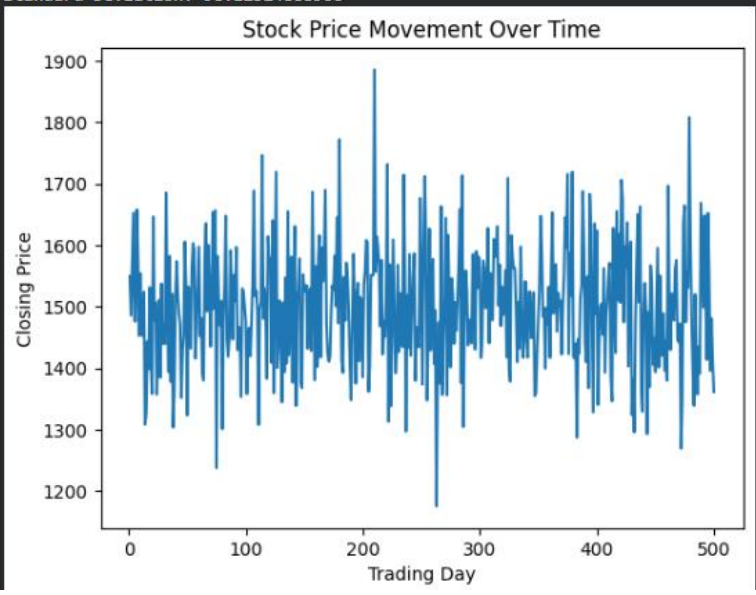
sample_data
car_price_dataset_500.csv
customer_segmentation_data_5...
house_price_dataset_500.csv
medical_knn_dataset_500.csv
retail_kmeans_dataset_500.csv
stock_price_variability_500.csv

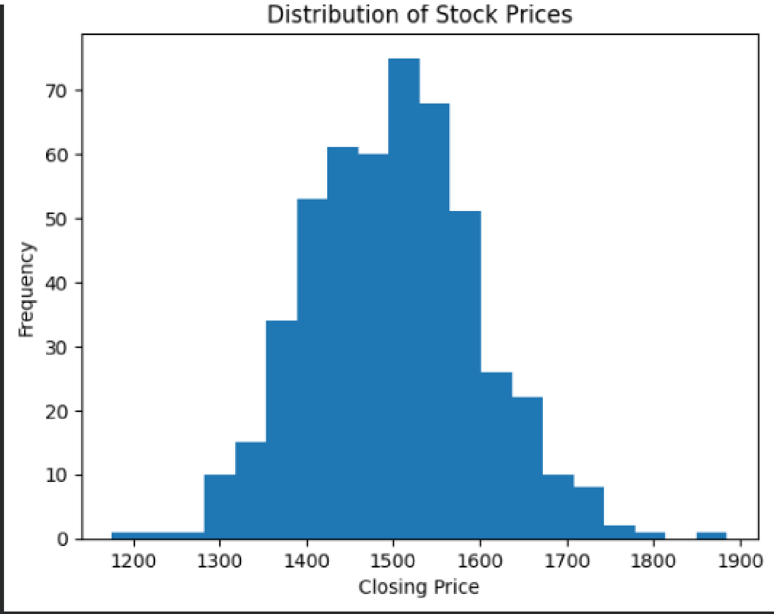
```
1        263.924324        22911.340541        11.864865
2        231.163522        11562.220126        40.157233
```

[7]

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv("stock_price_variability_500.csv")

print(df.head())

mean_price = df['Closing_Price'].mean()
variance_price = df['Closing_Price'].var()
std_dev_price = df['Closing_Price'].std()

print("\nStock Price Statistics:")
print("Mean Price:", mean_price)
print("Variance:", variance_price)
print("Standard Deviation:", std_dev_price)
plt.figure()
plt.plot(df['Day'], df['Closing_Price'])
plt.xlabel("Trading Day")
plt.ylabel("Closing Price")
plt.title("Stock Price Movement Over Time")
plt.show()
plt.figure()
plt.hist(df['Closing_Price'], bins=20)
plt.xlabel("Closing Price")
plt.ylabel("Frequency")
plt.title("Distribution of Stock Prices")
plt.show()
# ---------------------------------------------
if std_dev_price > 100:
    print("\nInsight: The stock shows high price volatility.")
else:
    print("\nInsight: The stock shows relatively stable price movement.")
```

```
   Day  Closing_Price
0    1        1549.67
1    2        1486.17
2    3        1564.77
3    4        1652.30
4    5        1476.58

Stock Price Statistics:
Mean Price: 1500.6839600000003
Variance: 9628.564469657722
Standard Deviation: 98.12524888966
```






```
Insight: The stock shows relatively stable price movement.
```

EXP_37

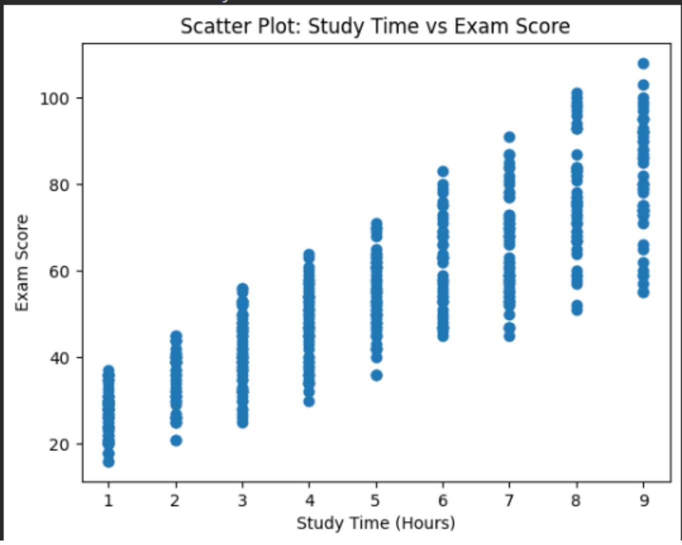
- ..
- sample_data
- car_price_dataset_500.csv
- customer_segmentation_data_5...
- house_price_dataset_500.csv
- medical_knn_dataset_500.csv
- retail_kmeans_dataset_500.csv
- stock_price_variability_500.csv
- study_time_exam_score_500.csv

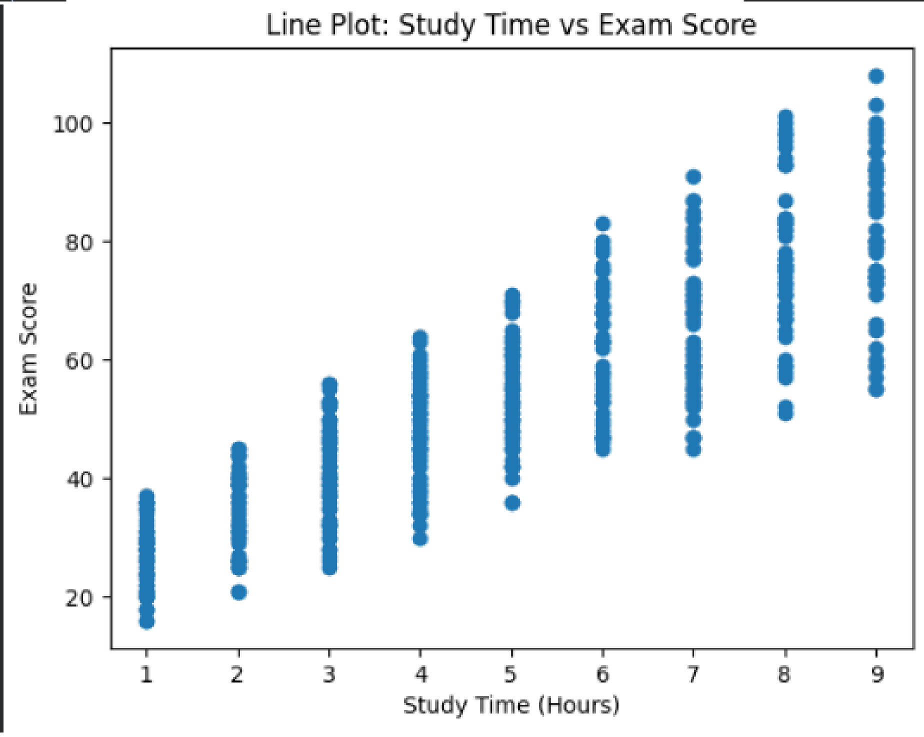
```
Insight: The stock shows relatively stable price movement.
```

[8]

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
df = pd.read_csv("study_time_exam_score_500.csv")

print(df.head())
correlation = df['Study_Time_Hours'].corr(df['Exam_Score'])
print("\nCorrelation between Study Time and Exam Score:", correlation)

plt.figure()
plt.scatter(df['Study_Time_Hours'], df['Exam_Score'])
plt.xlabel("Study Time (Hours)")
plt.ylabel("Exam Score")
plt.title("Scatter Plot: Study Time vs Exam Score")
plt.show()
plt.figure()
plt.plot(df['Study_Time_Hours'], df['Exam_Score'], 'o')
plt.xlabel("Study Time (Hours)")
plt.ylabel("Exam Score")
plt.title("Line Plot: Study Time vs Exam Score")
plt.show()
plt.figure()
df.boxplot(column='Exam_Score', by='Study_Time_Hours')
plt.xlabel("Study Time (Hours)")
plt.ylabel("Exam Score")
plt.title("Box Plot: Exam Score Distribution by Study Time")
plt.suptitle("")
plt.show()
if correlation > 0.5:
    print("\nInsight: Strong positive correlation – more study time generally leads to higher scores.")
elif correlation > 0:
    print("\nInsight: Moderate positive correlation between study time and exam score.")
else:
    print("\nInsight: Weak or no correlation between study time and exam score.")
```

```
   Study_Time_Hours  Exam_Score
0                 7          85
1                 4          45
2                 8          98
3                 5          56
4                 7          59

Correlation between Study Time and Exam Score: 0.8724061772649271
```

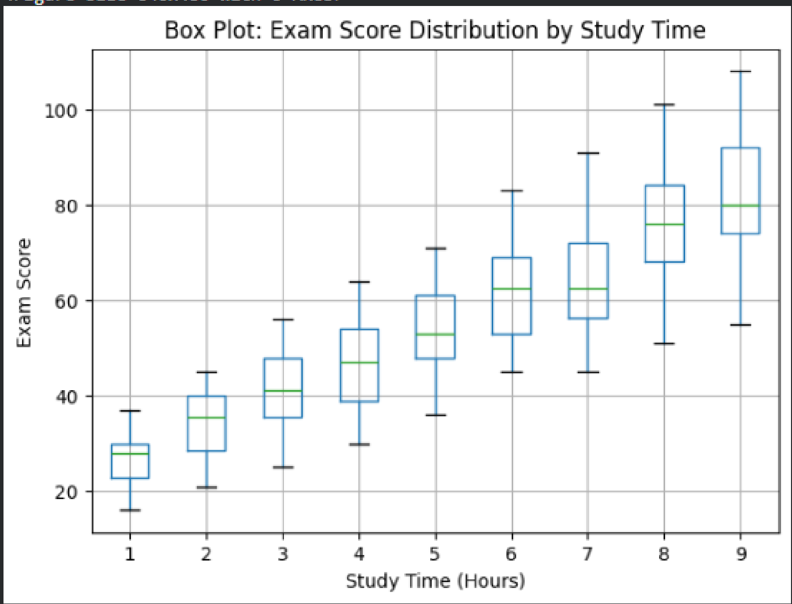




EXP_38

```python
import pandas as pd
df = pd.read_csv("city_temperature_data.csv")

print(df.head())


mean_temp = df.groupby('City')['Temperature'].mean()
print("\nMean Temperature for Each City:")
print(mean_temp)

std_temp = df.groupby('City')['Temperature'].std()
print("\nStandard Deviation of Temperature for Each City:")
print(std_temp)

temp_range = df.groupby('City')['Temperature'].max() - df.groupby('City')['Temperature'].min()
print("\nTemperature Range for Each City:")
print(temp_range)
highest_range_city = temp_range.idxmax()
print("\nCity with Highest Temperature Range:", highest_range_city)

most_consistent_city = std_temp.idxmin()
print("City with Most Consistent Temperature:", most_consistent_city)
```

```
<Figure size 640x480 with 0 Axes>
```




```
Insight: Strong positive correlation — more study time generally leads to higher scores.
```

EXP_39

```python
import pandas as pd
df = pd.read_csv("city_temperature_data.csv")

print(df.head())


mean_temp = df.groupby('City')['Temperature'].mean()
print("\nMean Temperature for Each City:")
print(mean_temp)

std_temp = df.groupby('City')['Temperature'].std()
print("\nStandard Deviation of Temperature for Each City:")
print(std_temp)

temp_range = df.groupby('City')['Temperature'].max() - df.groupby('City')['Temperature'].min()
print("\nTemperature Range for Each City:")
print(temp_range)
highest_range_city = temp_range.idxmax()
print("\nCity with Highest Temperature Range:", highest_range_city)

most_consistent_city = std_temp.idxmin()
print("City with Most Consistent Temperature:", most_consistent_city)
```

```
      City  Day  Temperature
0  Chennai    1        32.48
1  Chennai    2        29.31
2  Chennai    3        33.24
3  Chennai    4        37.62
4  Chennai    5        28.83

Mean Temperature for Each City:
City
Bengaluru    30.178000
Chennai      30.049890
Delhi        29.810219
Kolkata      30.408548
Mumbai       30.634603
Name: Temperature, dtype: float64

Standard Deviation of Temperature for Each City:
City
Bengaluru    4.992915
Chennai      4.740282
Delhi        5.097348
Kolkata      5.106776
Mumbai       4.784817
Name: Temperature, dtype: float64

Temperature Range for Each City:
City
Bengaluru    30.58
Chennai      35.47
Delhi        28.87
Kolkata      27.49
Mumbai       27.40
Name: Temperature, dtype: float64

City with Highest Temperature Range: Chennai
City with Most Consistent Temperature: Chennai
```

..
sample_data
car_price_dataset_500.csv
city_temperature_data.csv
customer_segmentation_data_5...
ecommerce_kmeans_dataset_50...
house_price_dataset_500.csv
medical_knn_dataset_500.csv
retail_kmeans_dataset_500.csv
stock_price_variability_500.csv
study_time_exam_score_500.csv

```
City with Most Consistent Temperature: Chennai
```

[10]
✓ 1s

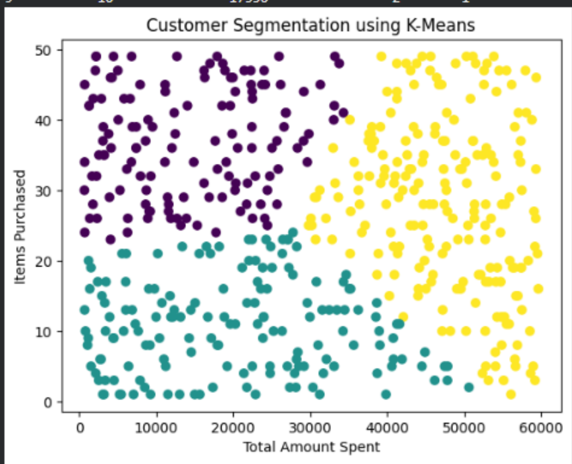
```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

df = pd.read_csv("ecommerce_kmeans_dataset_500.csv")

print(df.head())
X = df[['Total_Amount_Spent', 'Items_Purchased']]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)
print("\nClustered Data:")
print(df.head(10))
plt.scatter(
    df['Total_Amount_Spent'],
    df['Items_Purchased'],
    c=df['Cluster']
)
plt.xlabel("Total Amount Spent")
plt.ylabel("Items Purchased")
plt.title("Customer Segmentation using K-Means")
plt.show()
cluster_summary = df.groupby('Cluster').mean()
print("\nCluster Summary:")
print(cluster_summary)
```

Files

..
sample_data
car_price_dataset_500.csv
city_temperature_data.csv
customer_segmentation_data_5...
ecommerce_kmeans_dataset_50...
house_price_dataset_500.csv
medical_knn_dataset_500.csv
retail_kmeans_dataset_500.csv
stock_price_variability_500.csv
study_time_exam_score_500.csv

Disk 86.55 GB available

```
   Customer_ID  Total_Amount_Spent  Items_Purchased
0            1               56922                8
1            2               16295               46
2            3                1360               16
3            4               38658               14
4            5               54843               12

Clustered Data:
   Customer_ID  Total_Amount_Spent  Items_Purchased  Cluster
0            1               56922                8        2
1            2               16295               46        0
2            3                1360               16        1
3            4               38658               14        1
4            5               54843               12        2
5            6               45232               23        2
6            7               11784               15        1
7            8               55386               28        2
8            9                6765               34        0
9           10               17350                2        1
```




```
Cluster Summary:
         Customer_ID  Total_Amount_Spent  Items_Purchased
Cluster
0         236.059701        14542.514925        36.873134
1         260.639241        20393.518987        10.715190
2         252.100962        47657.413462        27.889423
```

EXP_40

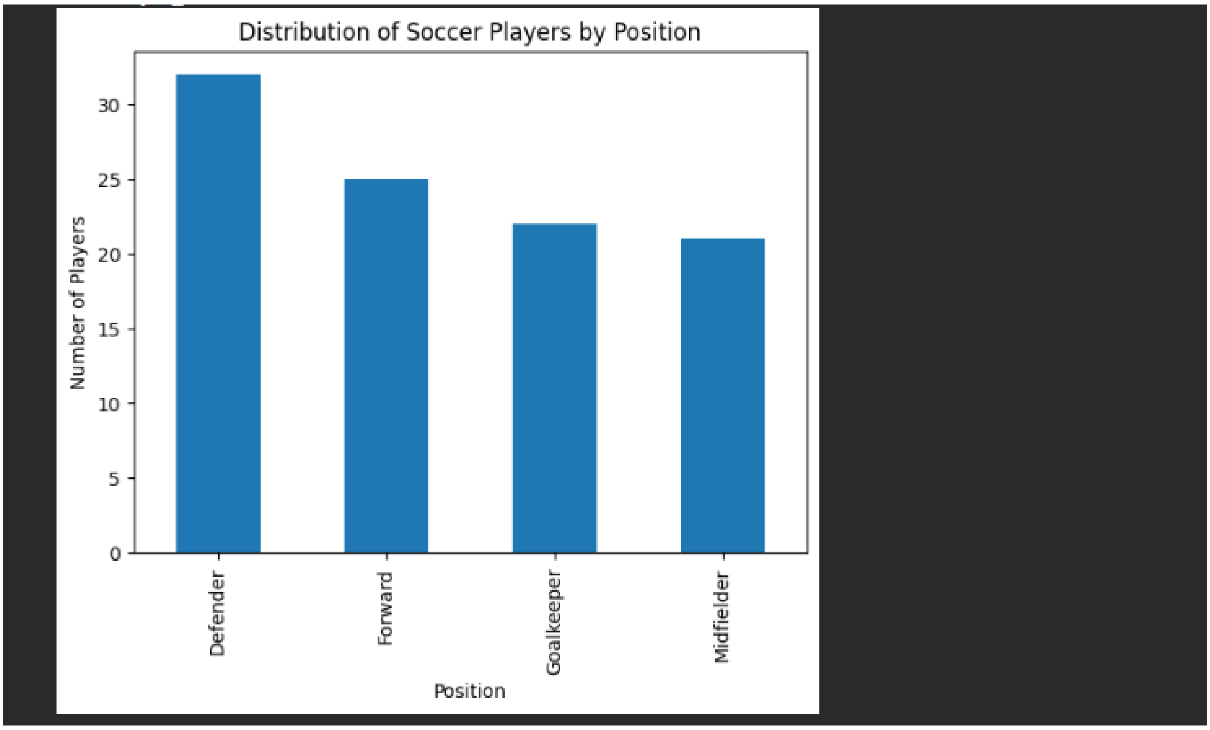
house_price_dataset_500.csv
medical_knn_dataset_500.csv
retail_kmeans_dataset_500.csv
soccer_players_data.csv
stock_price_variability_500.csv
study_time_exam_score_500.csv

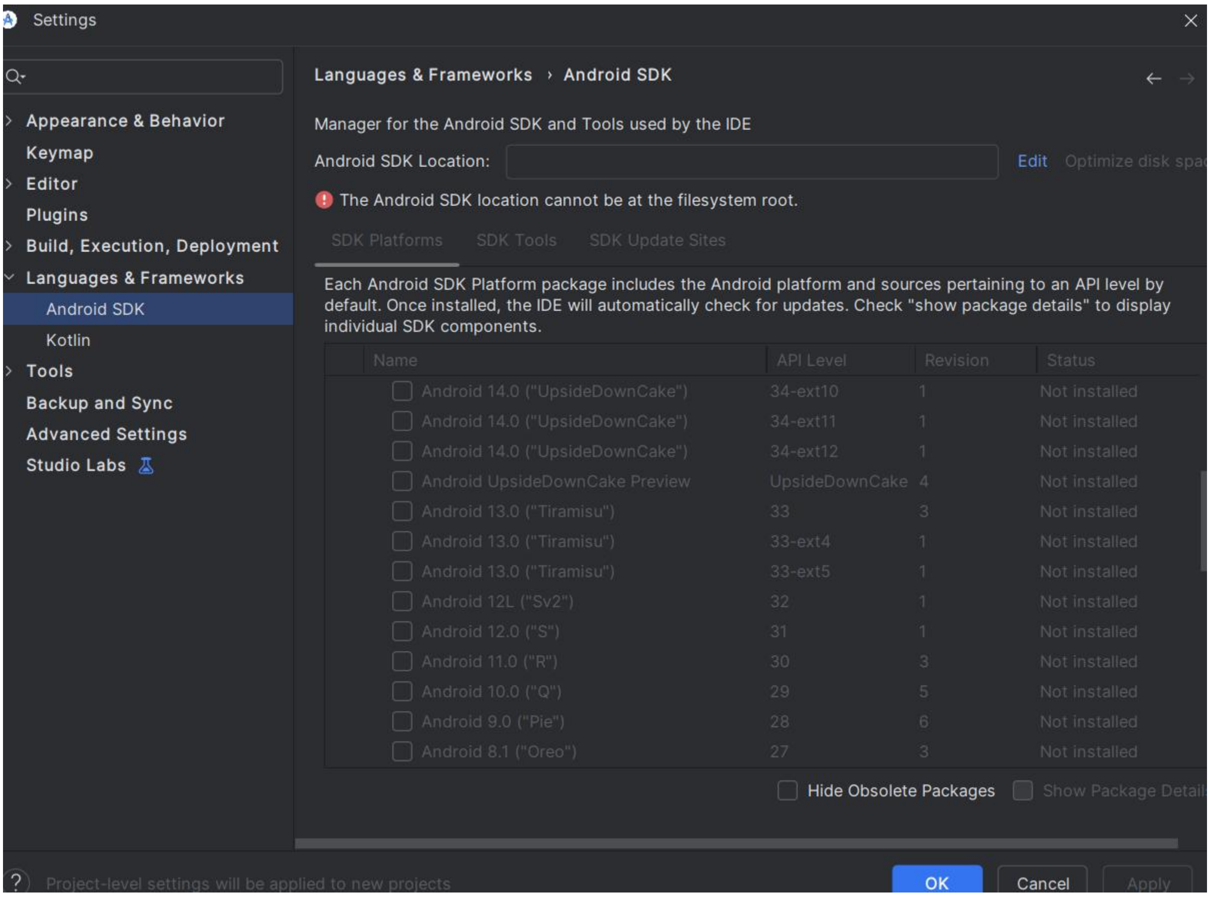
[11]

```python
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("soccer_players_data.csv")

print(df.head())
top_goals = df.sort_values(by="Goals_Scored", ascending=False).head(5)
print("\nTop 5 Players with Highest Goals:")
print(top_goals[['Player_Name', 'Goals_Scored']])
top_salary = df.sort_values(by="Weekly_Salary", ascending=False).head(5)
print("\nTop 5 Players with Highest Salaries:")
print(top_salary[['Player_Name', 'Weekly_Salary']])
average_age = df['Age'].mean()
print("\nAverage Age of Players:", average_age)
above_avg_age = df[df['Age'] > average_age]
print("\nPlayers Above Average Age:")
print(above_avg_age[['Player_Name', 'Age']])
position_count = df['Position'].value_counts()

plt.figure()
position_count.plot(kind='bar')
plt.xlabel("Position")
plt.ylabel("Number of Players")
plt.title("Distribution of Soccer Players by Position")
plt.show()
```

```
  Player_Name  Age    Position  Goals_Scored  Weekly_Salary
0    Player_1   24    Defender            31         342890
1    Player_2   37  Goalkeeper            23          99377
2    Player_3   32  Goalkeeper            11         491620
3    Player_4   28  Midfielder             1         366189
4    Player_5   25    Defender             2         410032

Top 5 Players with Highest Goals:
   Player_Name  Goals_Scored
75   Player_76            34
73   Player_74            34
52   Player_53            34
62   Player_63            34
81   Player_82            34

Top 5 Players with Highest Salaries:
   Player_Name  Weekly_Salary
88   Player_89         498982
33   Player_34         497456
48   Player_49         495101
72   Player_73         494209
70   Player_71         492905

Average Age of Players: 27.01
```

```
...  Players Above Average Age:
       Player_Name  Age
     1    Player_2   37
     2    Player_3   32
     3    Player_4   28
     6    Player_7   36
     7    Player_8   28
     8    Player_9   28
     13  Player_14   29
     17  Player_18   29
     18  Player_19   29
     19  Player_20   34
     21  Player_22   33
     22  Player_23   32
     23  Player_24   32
     24  Player_25   36
     25  Player_26   29
     26  Player_27   37
     29  Player_30   36
     33  Player_34   35
     35  Player_36   31
     36  Player_37   35
     39  Player_40   37
     40  Player_41   32
     42  Player_43   29
     44  Player_45   32
     46  Player_47   31
     47  Player_48   34
     49  Player_50   35
     56  Player_57   35
     57  Player_58   29
     61  Player_62   31
     62  Player_63   33
     63  Player_64   32
     65  Player_66   31
     67  Player_68   33
     68  Player_69   30
     69  Player_70   35
     70  Player_71   32
     71  Player_72   30
     73  Player_74   32
     74  Player_75   30
     79  Player_80   29
     81  Player_82   28
     82  Player_83   36
     83  Player_84   34
     95  Player_96   29
     98  Player_99   33
```


Distribution of Soccer Players by Position
Number of Players
0
5
10
15
20
25
30
Defender
Forward
Goalkeeper
Midfielder
Position


Settings
Appearance & Behavior
Keymap
Editor
Plugins
Build, Execution, Deployment
Languages & Frameworks
Android SDK
Kotlin
Tools
Backup and Sync
Advanced Settings
Studio Labs
Languages & Frameworks › Android SDK
Manager for the Android SDK and Tools used by the IDE
Android SDK Location:
Edit
Optimize disk spa
The Android SDK location cannot be at the filesystem root.
SDK Platforms
SDK Tools
SDK Update Sites
Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, the IDE will automatically check for updates. Check "show package details" to display individual SDK components.
Name
API Level
Revision
Status
Android 14.0 ("UpsideDownCake") 34-ext10 1 Not installed
Android 14.0 ("UpsideDownCake") 34-ext11 1 Not installed
Android 14.0 ("UpsideDownCake") 34-ext12 1 Not installed
Android UpsideDownCake Preview UpsideDownCake 4 Not installed
Android 13.0 ("Tiramisu") 33 3 Not installed
Android 13.0 ("Tiramisu") 33-ext4 1 Not installed
Android 13.0 ("Tiramisu") 33-ext5 1 Not installed
Android 12L ("Sv2") 32 1 Not installed
Android 12.0 ("S") 31 1 Not installed
Android 11.0 ("R") 30 3 Not installed
Android 10.0 ("Q") 29 5 Not installed
Android 9.0 ("Pie") 28 6 Not installed
Android 8.1 ("Oreo") 27 3 Not installed
Hide Obsolete Packages
Show Package Detail
Project-level settings will be applied to new projects
OK
Cancel
Apply