Machine Learning lab Manual

1.Install and set up Python and essential libraries like NumPy and Pandas

Setting up Python and essential libraries on a Windows system for machine learning involves a series of straightforward steps that prepare the environment for data analysis and algorithm development. By installing Python along with NumPy and Pandas, users can handle a wide array of data manipulation tasks efficiently.

Follow the below steps to set up Python and essential libraries such as NumPy and Pandas for machine learning on Windows.

Step 1: Install Python

Download Python: Go to the official Python website at python.org, navigate to the "Downloads section, and download the latest version for Windows. Choose the executable installer.

Install Python: Execute the downloaded file. It is crucial to check the box labeled "Add Python 3.x in PATH" at the start of the installation wizard. Select "Customize installation" and ensure all options including "pip", are selected. In the "Advanced Options," choose "Install for all users" and set the installation path to C:\Python. Proceed by clicking "Install".

Step 2: Install PIP

PIP generally comes installed with Python 3.4 and later. To confirm its installation, open Command Prompt and execute:

```
C:\Users\IPG3 APIN>pip --version
```

If pip is not installed or if we need to update it, we can use the following command to install or upgrade pip:

```
C:\Users\IPG3 APIN>python -m ensurepip --upgrade
```

After installation, we can verify that pip is installed correctly by running:

```
C:\Users\IPG3 APIN> python -m pip –version
```

Step 3: Installing Necessary Tools:

Essential Libraries: Libraries such as Jupyter, NumPy, pandas, Matplotlib, and Scikit-Learn should be installed if they are not already present. These can be installed using pip, which is Python's package manager. Open Command Prompt and enter the following command:

```
C:\Users\IPG3 APIN>python -m ensurepip --upgrade
```

```
C:\Users\IPG3 APIN>pip install jupyter
```

```
C:\Users\IPG3 APIN> pip install jupyter notebook
```

```
C:\Users\IPG3 APIN>pip install numpy
```

```
C:\Users\IPG3 APIN>pip install scikit-learn
C:\Users\IPG3 APIN>pip install pandas
C:\Users\IPG3 APIN>pip install openpyxl
```

2) Introduce scikit-learn as a machine learning library.

Scikit-learn is a popular open-source machine learning library in Python that offers a comprehensive set of tools and algorithms for data analysis, modeling, and machine learning tasks. It is built on foundational libraries like NumPy, SciPy, and Matplotlib. Scikit-learn provides a user-friendly and efficient framework for bath beginners and experts in the field of data science.

Some key points to introduce scikit-learn as a machine learning library:

1. Comprehensive Machine Learning Library: Scikit-learn offers a wide range of machine learning algorithms and tools for various tasks such as classification, regression, clustering, dimensionality reduction, and more.

2. User-Friendly and Easy to Use: It is designed with a user-friendly interface and simple syntax, making it accessible for both beginners and experienced machine learning practitioners.

3. Integration with Scientific Computing Libraries: Scikit-learn integrates well with other scientific computing libraries in Python such as NumPy, SciPy, and Matplotlib, providing a powerful environment for machine learning tasks.

4. Extensive Documentation and Community Support: The library comes with comprehensive documentation, tutorials, and examples to help users understand and implement machine learning algorithms effectively. Additionally, there is a vibrant community around scikit-learn that provides support and contributions.

5. Efficient Implementation of Algorithms: Scikit-learn is built on top of NumPy, SciPy, and Cython, which allows for efficient implementation of machine learning algorithms and scalability to large datasets.

6. Support for Model Evaluation and Validation: The library provides tools for model evaluation hyperparameter tuning, cross- validation, and performance metrics, enabling users to assess and Improve the quality of their machine learning models.

7. Flexibility and Customization: Scikit-learn offers flexibility for customization and parameter tuning, allowing users to adapt algorithms to their specific requirements and datasets.

8. Wide Adoption and Industry Usage: Due to its ease of use, performance, and versatility, scikit-learn is widely adopted Overall industry for various machine learning applications.

Overall, scikit-learn is a powerful and versatile machine learning library in Python that empowers users to build and deploy machine learning models efficiently for a wide range of tasks and applications.

3) Install and set up scikit-learn and other necessary tools.

Pip upgrade:

Using pip:

- Pip is Python's package manager.
- It usually comes installed with Python.
- Open a terminal/command prompt.

```
Py –m install pip - -upgrade pip
```

a) Install NumPy :-

```
pip install numpy
```

Press Enter. This command will download and install NumPy.

b) Install pandas :-

```
pip install pandas
```

Press Enter. This command will download and install pandas.

c)Install matplotlib:-

```
pip install matplotlib
```

Press Enter. This command will download and install matplotlib.

d)Install scipy:-

```
pip install scipy
```

Press Enter. This command will download and install scipy

e)Install scikit-learn(sklearn):-

```
pip install scikit-learn
```

Press Enter. This command will download and install scikit-learn

```
import sklearn
import numpy
import pandas
import matplotlib
print(sklearn._ _version_ _)
print(numpy._ _version_ _)
print(pandas._ _version_ _)
print(matplotlib._ _version_ _)
```

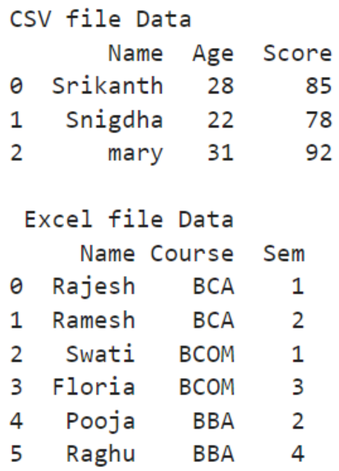
output:

```
1.2.2
1.26.4
2.1.4
3.8.0
```

4. Write a program to Load and explore the dataset of .CSV and excel files using pandas.

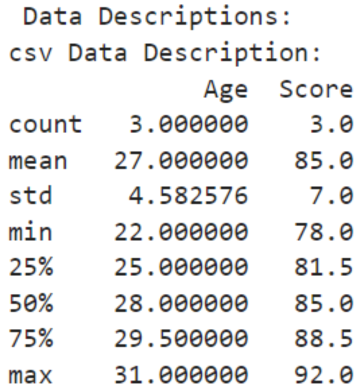
```
import pandas as pd
csvfilepath='C:\\ml_projects\\sample_data.csv'
excelfilepath='C:\\ml_projects\\sample_data.xlsx'
datacsv=pd.read_csv(csvfilepath)
print("CSV file Data")
print(datacsv)
dataexcel=pd.read_excel(excelfilepath)
print("\n Excel file Data")
print(dataexcel)
print("\n Data Descriptions:")
print("Csv Data Description:")
print(datacsv.describe())
print("\n Data Descriptions:")
print("\n Excel data Descriptions:")
print(dataexcel.describe())
print("\n Data types in CSV file:")
print(datacsv.dtypes)
print("\n Data types in Excel file:")
print(dataexcel.dtypes)
```

Output:

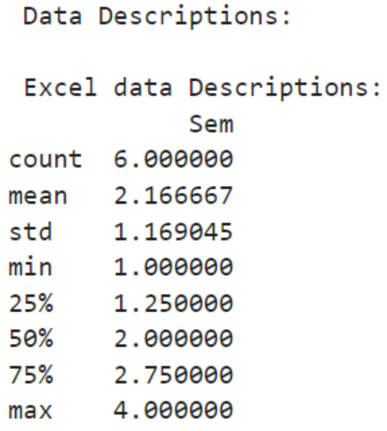


```
CSV file Data
       Name  Age  Score
0  Srikanth   28     85
1   Snigdha   22     78
2      mary   31     92

 Excel file Data
     Name Course  Sem
0  Rajesh    BCA    1
1  Ramesh    BCA    2
2   Swati   BCOM    1
3  Floria   BCOM    3
4   Pooja    BBA    2
5   Raghu    BBA    4
```

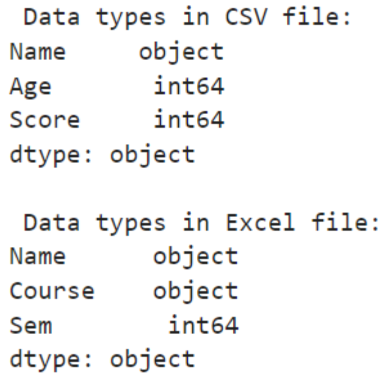


```
 Data Descriptions:
csv Data Description:
             Age  Score
count   3.000000    3.0
mean   27.000000   85.0
std     4.582576    7.0
min    22.000000   78.0
25%    25.000000   81.5
50%    28.000000   85.0
75%    29.500000   88.5
max    31.000000   92.0
```



```
 Data Descriptions:

 Excel data Descriptions:
            Sem
count  6.000000
mean   2.166667
std    1.169045
min    1.000000
25%    1.250000
50%    2.000000
75%    2.750000
max    4.000000
```



```
 Data types in CSV file:
Name      object
Age        int64
Score      int64
dtype: object

 Data types in Excel file:
Name       object
Course     object
Sem         int64
dtype: object
```

Note:

1. Create the CSV file sample_data.csv with the following fields.

Name,Age,Score

Srikanth,28,85

Snigdha,22,78

mary,31,92

2.Create the Excel file sample_data.xlsx with the following fields.

Name	Course	Sem
Rajesh	BCA	1
Ramesh	BCA	2
Swati	BCOM	1
Floria	BCOM	3
Pooja	BBA	2
Raghu	BBA	4

5. Write a program to Visualize the dataset to gain insights using Matplotlib or Seaborn by plotting scatter plots, and bar charts.

```python
import pandas as pd
import matplotlib.pyplot as plt
data=pd.read_csv("C:\\ml_projects\\study_data.csv")
plt.figure(figsize=(14,7))
plt.subplot(1,2,1)
plt.scatter(data['Studyhours'], data['Examscores'], color='dodgerblue',edgecolor='k',alpha=0.7)
plt.title('Studyhours vs. Examscores')
plt.xlabel('Studyhours')
plt.ylabel('Examscores')
plt.grid(True)
bins=[0,2,4,6,8,10,12]
labels=['0-2','2-4','4-6','6-8','8-10','10-12']
data['StudyhourRange']=pd.cut(data['Studyhours'], bins=bins,labels=labels,right=False)
grouped_data=data.groupby('StudyhourRange')['Examscores'].mean()
plt.subplot(1,2,2)
grouped_data.plot(kind='bar',color='salmon')
plt.title('Average Examscore by Studyhour Range')
plt.xlabel('Studyhour Range')
plt.ylabel('Average Examscore')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

Note:

Create the CSV file study_data.csv with the following fields.

StudentID,Studyhours,Examscores

1,5,82

2,2,48

3,8,90

4,1,35

5,3,50

6,4,66

7,9,95

8,6,75

9,7,88

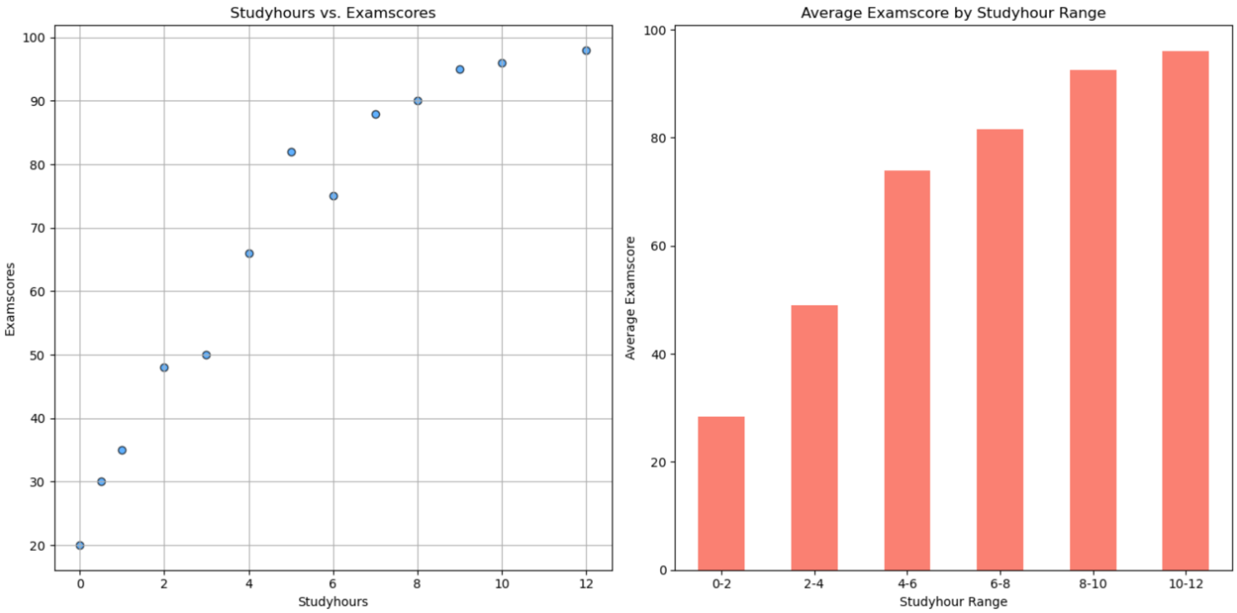
10,0.5,30

11,10,96

12,0,20

13,12,98

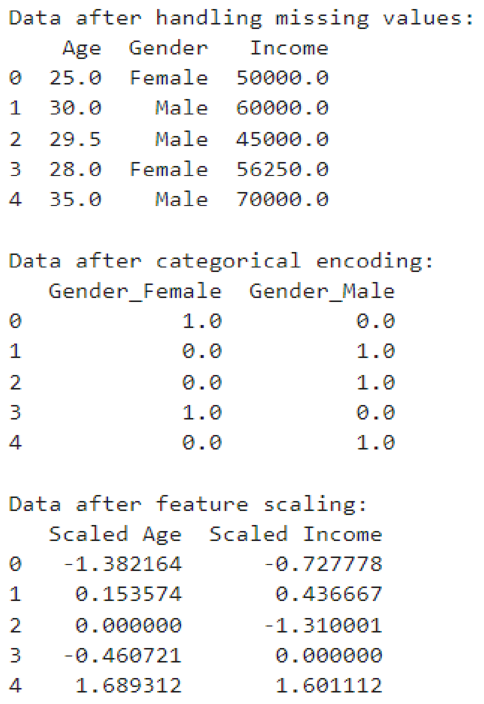
Output:



6.Write a program to Handle missing data, encode categorical variables, and perform feature scaling.

```
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
data = {
'Age': [25, 30, None, 28, 35],
'Gender': ['Female', 'Male', 'Male', 'Female', 'Male'],
'Income': [50000, 60000, 45000, None, 70000]
}
df =pd.DataFrame(data)
imputer =SimpleImputer(strategy='mean')
df[['Age', 'Income']] = imputer.fit_transform(df[['Age', 'Income']])
print("Data after handling missing values:")
print(df)
encoder=OneHotEncoder()
encoded_data= encoder.fit_transform(df [['Gender']]).toarray()
encoded_df =pd.DataFrame(encoded_data, columns=encoder.get_feature_names_out(['Gender']))
print("\nData after categorical encoding:")
print(encoded_df)
scaler =StandardScaler()
scaled_data =scaler.fit_transform(df [['Age', 'Income']])
scaled_df = pd.DataFrame (scaled_data, columns=['Scaled Age', 'Scaled Income'])
print("\nData after feature scaling:")
print(scaled_df)
```

<u>Output</u>:



```
Data after handling missing values:
    Age  Gender   Income
0  25.0  Female  50000.0
1  30.0    Male  60000.0
2  29.5    Male  45000.0
3  28.0  Female  56250.0
4  35.0    Male  70000.0

Data after categorical encoding:
   Gender_Female  Gender_Male
0            1.0          0.0
1            0.0          1.0
2            0.0          1.0
3            1.0          0.0
4            0.0          1.0

Data after feature scaling:
   Scaled Age  Scaled Income
0   -1.382164      -0.727778
1    0.153574       0.436667
2    0.000000      -1.310001
3   -0.460721       0.000000
4    1.689312       1.601112
```

7. Write a program to implement a k-Nearest Neighbours (k-NN) classifier using scikit-learn and Train the classifier on the dataset and evaluate its performance.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
X = np.array([[80, 75], [95, 90], [60, 50], [45, 30], [30, 40], [85, 95], [70, 60], [50, 55], [40, 45], [60, 70]])
y= np.array([1, 1, 0, 0, 0, 1, 1, 0, 0, 1])
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2, random_state=42)
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred=knn.predict(X_test)
accuracy=accuracy_score(y_test, y_pred)
print("Accuracy on the test set: {:.2f}".format(accuracy))
exam_score1 = float(input("Enter Exam Score 1: "))
exam_score2 = float(input("Enter Exam Score 2: "))
user_input = np.array([[exam_score1 , exam_score2]])
predicted_outcome =knn.predict(user_input)
if predicted_outcome[0] == 1:
   print("The student is predicted to pass.")
else:
   print("The student is predicted to fail.")
```

<u>Output 1</u>:

```
Accuracy on the test set: 1.00
Enter Exam Score 1:  75
Enter Exam Score 2:  89
The student is predicted to pass.
```

<u>Output 2</u>:

```
Accuracy on the test set: 1.00
Enter Exam Score 1:  45
Enter Exam Score 2:  50
The student is predicted to fail.
```

8. Write a program to implement a linear regression model for regression tasks and Train t he model on a dataset with continuous target variables.

```
import numpy as np
from sklearn.linear_model import LinearRegression
X=([[1000,2],[1500,3],[1200,2],[1800,4],[900,2],[2000,3]])
y=np.array([300000,400000,350000,500000,280000,450000])
model=LinearRegression()
```

```
model.fit(X,y)
size=int(input("Enter the size of the house in sqft:"))
rooms=int(input("Enter the number of rooms:"))
newdata=np.array([[size,rooms]])
predictedprice=model.predict(newdata)
print("Predicted price for a house with size {} sqft and {} rooms is Rs.
{:.2f}".format(size,rooms,predictedprice[0]))
```

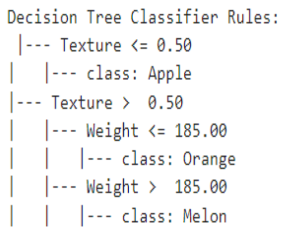
<u>Output</u>:

```
Enter the size of the house in sqft: 1200
Enter the number of rooms: 2
Predicted price for a house with size 1200 sqft and 2 rooms is Rs. 324327.87
```

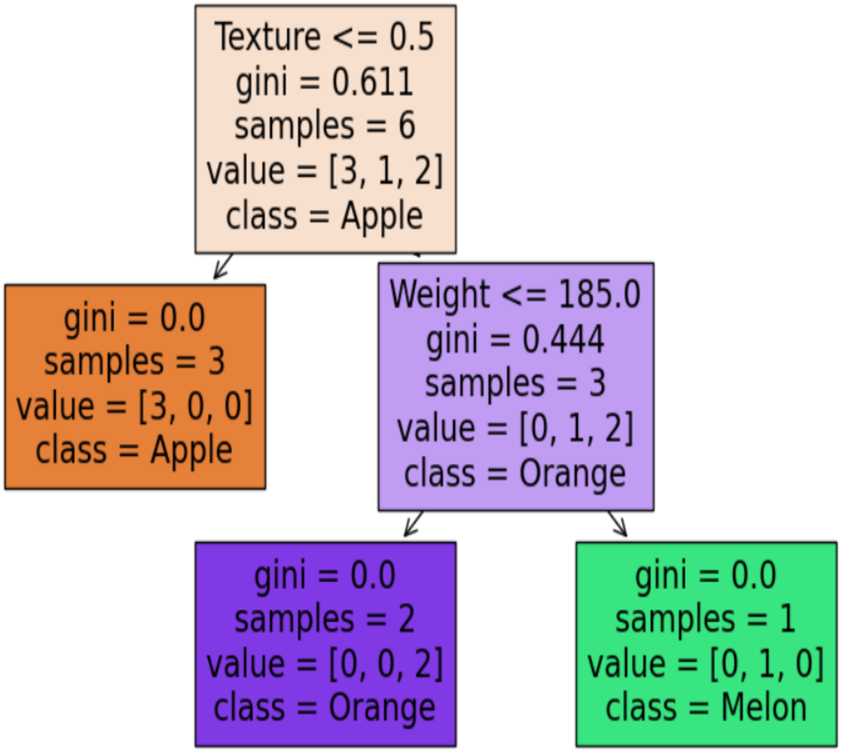
9. Write a program to implement a decision tree classifier using scikit-learn and visualize the decision tree and understand its splits.

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.tree import export_text
import matplotlib.pyplot as plt
X=np.array([[150, 0], [170, 1], [120, 0], [140, 1], [200, 1], [130, 0]])
y=np.array(['Apple', 'Orange', 'Apple', 'Orange', 'Melon', 'Apple'])
clf= DecisionTreeClassifier(random_state=42)
clf.fit(X, y)
tree_rules=export_text(clf, feature_names=['Weight', 'Texture'])
print("Decision Tree Classifier Rules: \n", tree_rules)
plt.figure(figsize=(10, 6))
plot_tree(clf, filled=True, feature_names=['Weight', 'Texture'], class_names=np.unique(y))
plt.show()
```

Output:



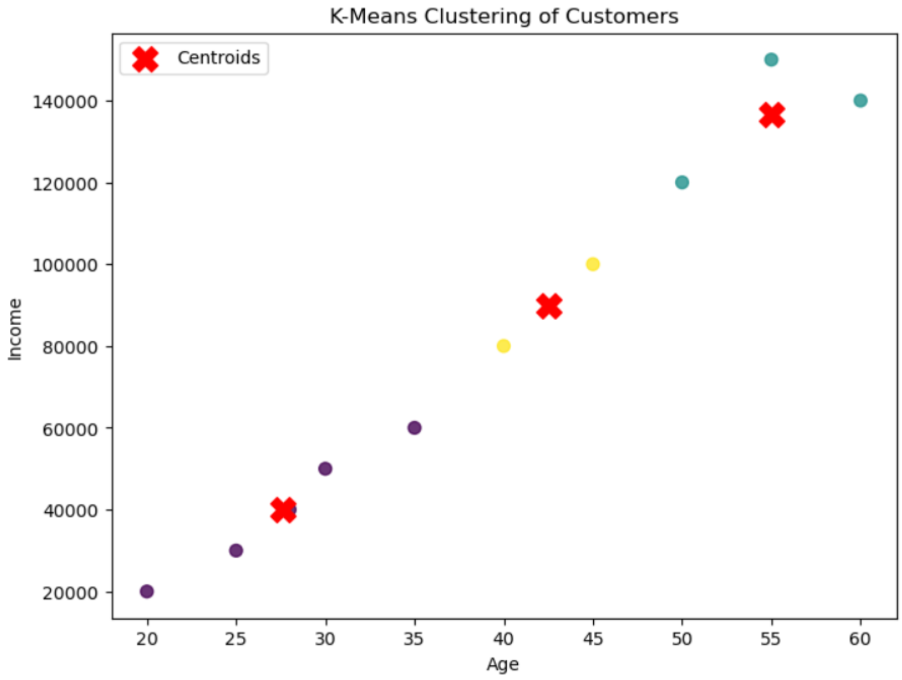
```
Decision Tree Classifier Rules:
 |--- Texture <= 0.50
|   |--- class: Apple
|--- Texture >  0.50
|   |--- Weight <= 185.00
|   |   |--- class: Orange
|   |--- Weight >  185.00
|   |   |--- class: Melon
```



10. Write a program to Implement K-Means clustering and Visualize clusters.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
X= np.array([[30, 50000], [35, 60000], [40, 80000], [25, 30000], [45, 100000], [20, 20000], [50,
120000], [55, 150000], [60, 140000], [28, 40000]])
kmeans=KMeans(n_clusters=3, random_state=0)
kmeans.fit(X)
labels =kmeans.labels_
centers =kmeans.cluster_centers_
plt.figure(figsize=(8,6))
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=50, alpha=0.8)
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, marker='X', label='Centroids')
plt.xlabel('Age')
plt.ylabel('Income')
plt.title('K-Means Clustering of Customers')
plt.legend()
plt.show()
```

Output:


K-Means Clustering of Customers
✖ Centroids
Income
Age