|  |  |  |  |
| --- | --- | --- | --- |
| 3 | Feature Scaling and Dummification |  |  |
| 4 | Hypothesis Testing |  |  |
| 5 | ANOVA (Analysis of Variance) |  |  |
| 6 | Regression and Its Types |  |  |
| 7 | Logistic Regression and Decision Tree |  |  |
| 8 | K-Means Clustering |  |  |
| 9 | Principal Component Analysis (PCA) |  |  |
| 10 | Data Visualization and Storytelling |  |  |

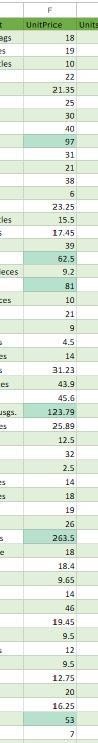
**Aim:**

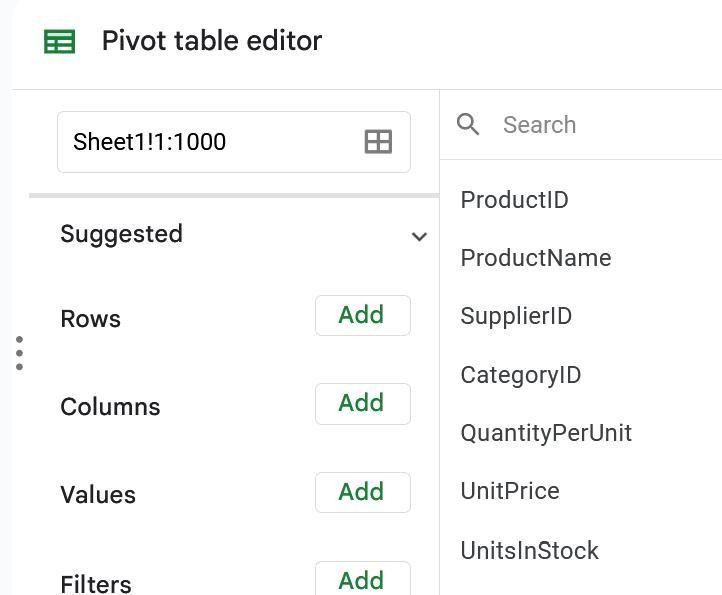
# Practical No 1

Program 1: Introduction to Excel Perform conditional formatting on a dataset using various criteria. Create a pivot table to analyze and summarize data.

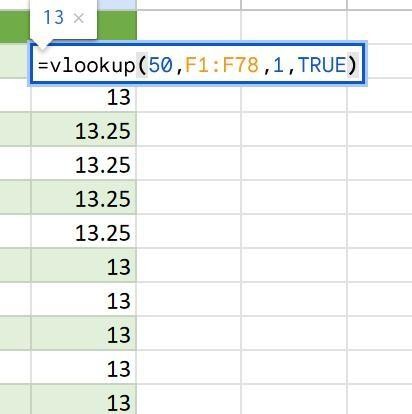
Use VLOOKUP function to retrieve information from a different worksheet or table. Perform what-if analysis using Goal Seek to determine input values for desired output.

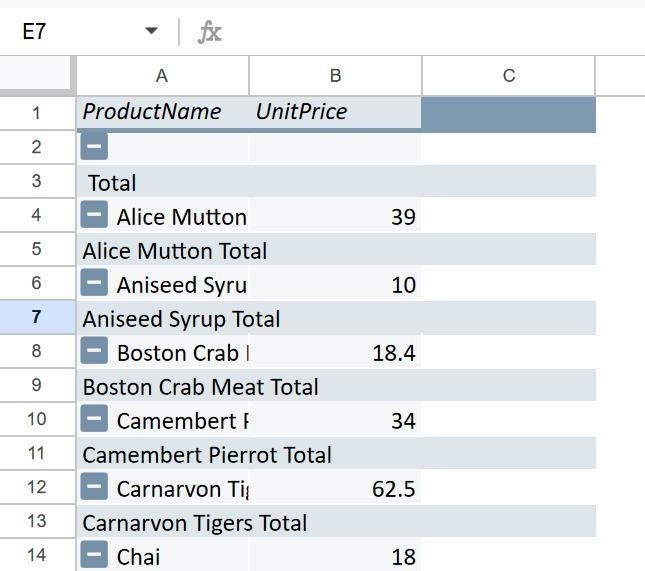
Output:



Creating a pivot table

Using vlookup-





# Practical No 2

**Aim:**Data Frames and Basic Data Pre-processing Read data from CSV and JSON files into a data frame.

Perform basic data pre-processing tasks such as handling missing values and outliers.

Manipulate and transform data using functions like filtering, sorting, and grouping.

Code:

import pandas df =

pandas.read\_csv('C:\\Users\\user1\\Downloads\\Employee\\acbb2271e66c10a5b73aacf82ca8278 4-e38afe62e088394d61ed30884dd50a6826eee0a8\\employees.csv')

df\_json = pandas.read\_json("C:\\Users\\user1\\Downloads\\emplyejson2\\8235702- a50f7c449c41b6dc8eb87d8d393eeff62121b392\\employees.json")

missing\_values = df.isnull().sum() print("Missing values are: ") print(missing\_values)

df = df.dropna()

#df\_json = df\_json.fillna(df\_json.mean())

print(df) #print(df\_json)

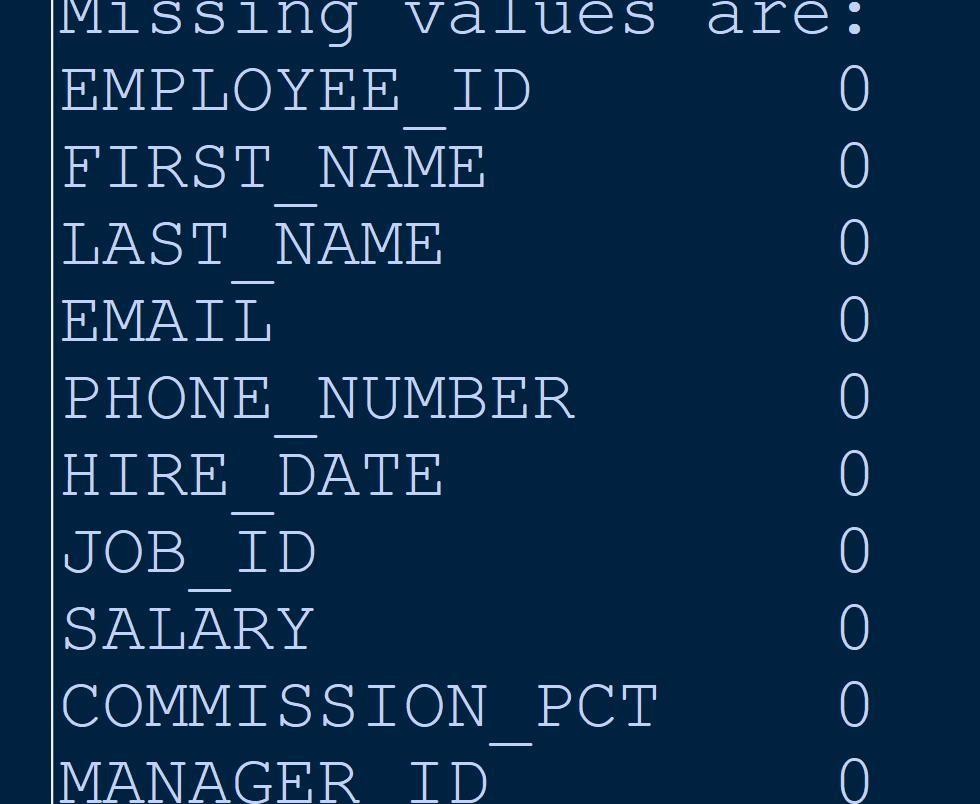
filtered\_df = df[df['Column'] > 50]

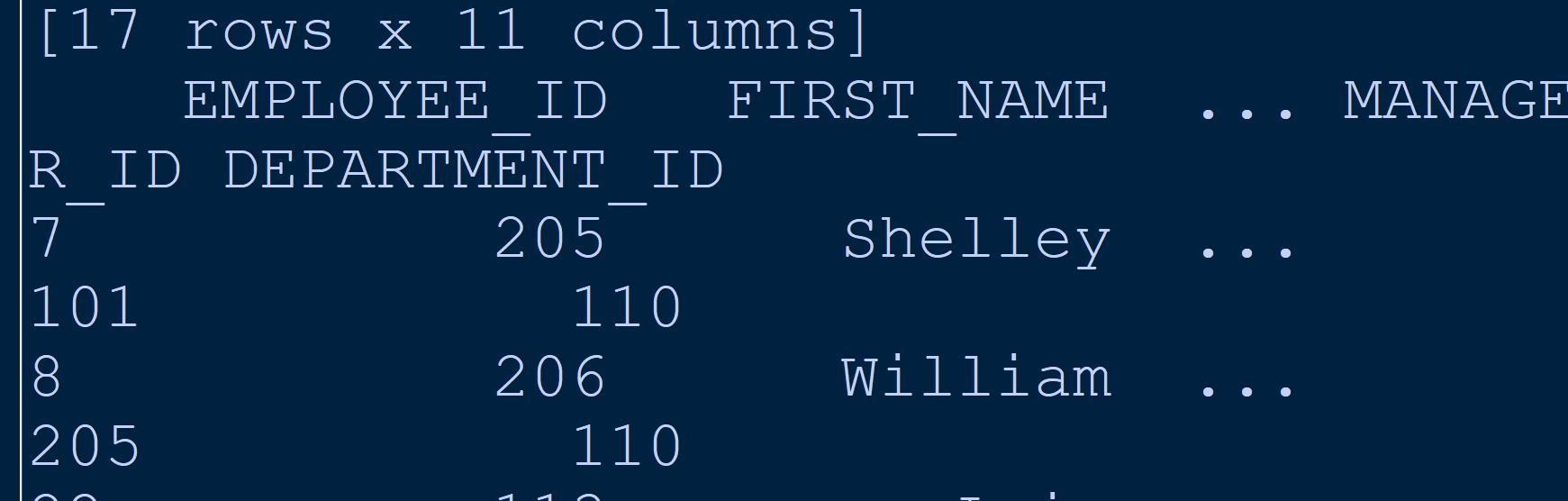
print(filtered\_df)

sorted\_df = df.sort\_values(by='Column', ascending=False) print(sorted\_df)

grouped\_df = df.groupby('Category')['Numeric\_Column'].mean() print(grouped\_df)

Output:





Practical 3

Code:

import pandas

from sklearn.preprocessing import StandardScaler,MinMaxScaler

df = pandas.read\_csv("C:\\Users\\RDNC\\Downloads\\8836201- 6f9306ad21398ea43cba4f7d537619d0e07d5ae3\\8836201- 6f9306ad21398ea43cba4f7d537619d0e07d5ae3\\iris.csv")

# replace path with any dataset csv file print(df)

numerical\_features = df.select\_dtypes(include=["int64", "float64"])

print(numerical\_features) scaler = StandardScaler()

standardized\_features = scaler.fit\_transform(numerical\_features)

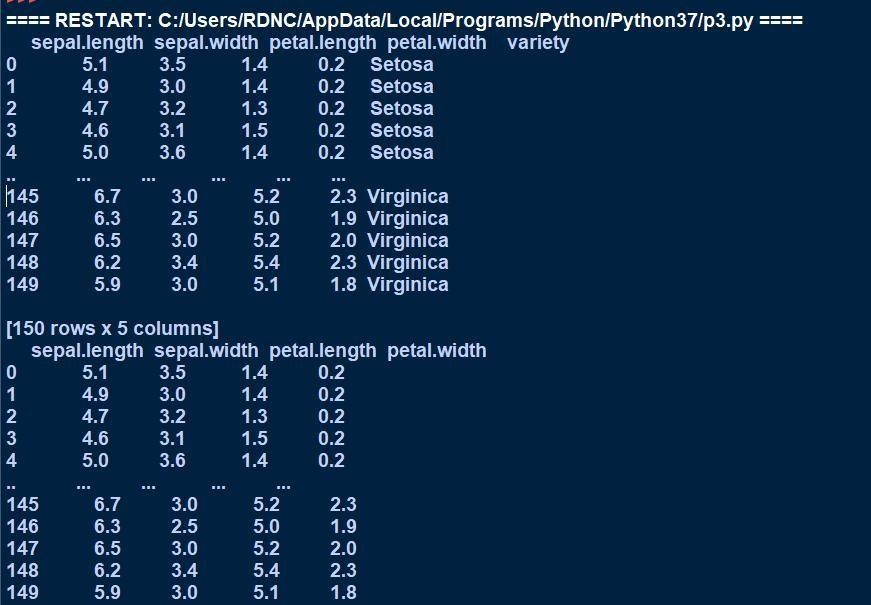
print(standardized\_features)

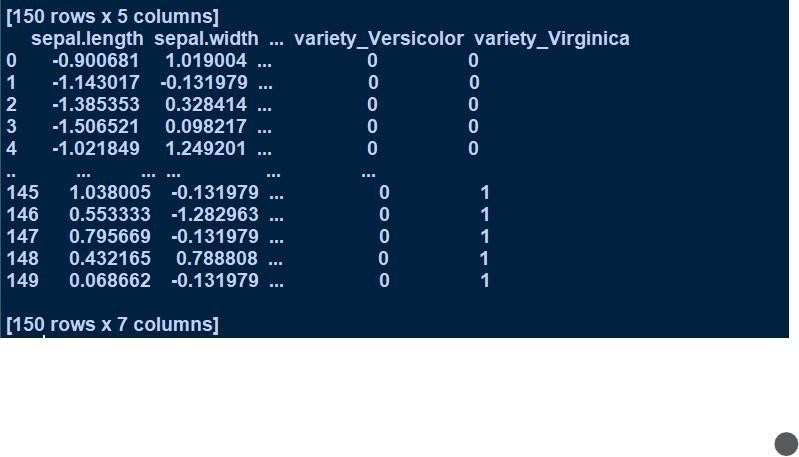
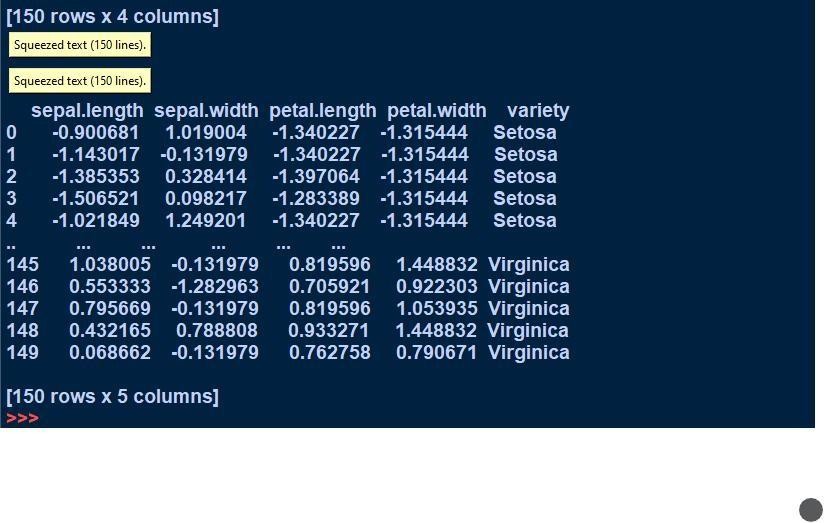
scaler = MinMaxScaler()

normalized\_features = scaler.fit\_transform(numerical\_features)

print(normalized\_features) df[numerical\_features.columns] = standardized\_features print(df)

print(pandas.get\_dummies(df))

Output:



Practical No 4

Aim:

Hypothesis Testing Formulate null and alternative hypotheses for a given problem. Conduct a hypothesis test using appropriate statistical tests (e.g., t-test, chisquare test). Interpret the results and draw conclusions based on the test outcomes.

Code:

import numpy as np import scipy.stats as stats

# Create a dummy dataset of 10 year old children's weight data = np.random.randint(20, 40, 10)

# Define the null hypothesis

H0 = "The average weight of 10 year old children is 32kg."

# Define the alternative hypothesis

H1 = "The average weight of 10 year old children is more than 32kg."

# Calculate the test statistic

t\_stat, p\_value = stats.ttest\_1samp(data, 32)

# Print the results print("Test statistic:", t\_stat) print("p-value:", p\_value)

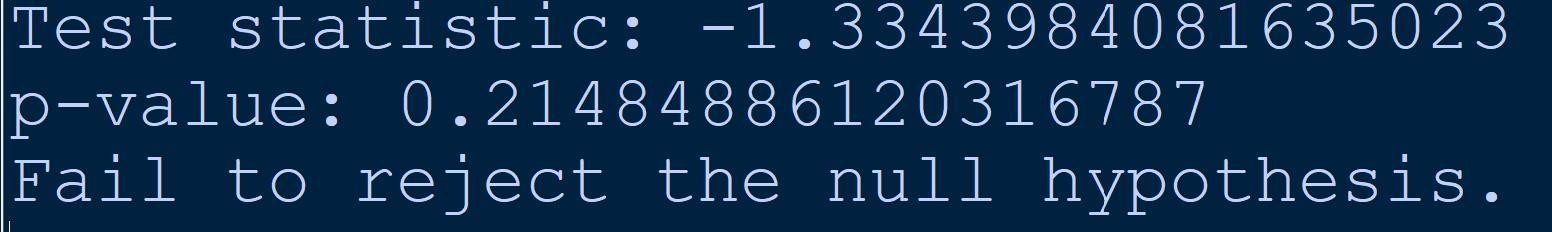
# Conclusion

if p\_value < 0.05:

print("Reject the null hypothesis.")

else:

print("Fail to reject the null hypothesis.")

Output:

Practical No 5

Aim:ANOVA (Analysis of Variance) Perform one-way ANOVA to compare means across multiple groups. Conduct post-hoc tests to identify significant differences between group means.

Code:

import scipy.stats as stats

from statsmodels.stats.multicomp import pairwise\_tukeyhsd

group1 = [23,25,29,34,30]

group2 = [19,20,22,25,24]

group3 = [15,18,20,21,17]

group4 = [28,24,26,30,29]

all\_data = group1+group2+group3+group4

group\_labels = ['Group1'] \* len(group1) +['Group2'] \* len(group2) +['Group3'] \* len(group3)

+['Group4'] \* len(group4)

f\_statistic,p\_value = stats.f\_oneway(group1,group2,group3,group4) print("One way Anova: ")

print("F statistic: ",f\_statistic) print("P value: ",p\_value)

tukey\_results = pairwise\_tukeyhsd(all\_data,group\_labels)

print("\n Tukey-Kramer post host test: ") print(tukey\_results)

Output:

One way Anova:

F statistic: 12.139872842870115

P value: 0.00021465200901629603

Tukey-Kramer post host test:

Multiple Comparison of Means - Tukey HSD, FWER=0.05

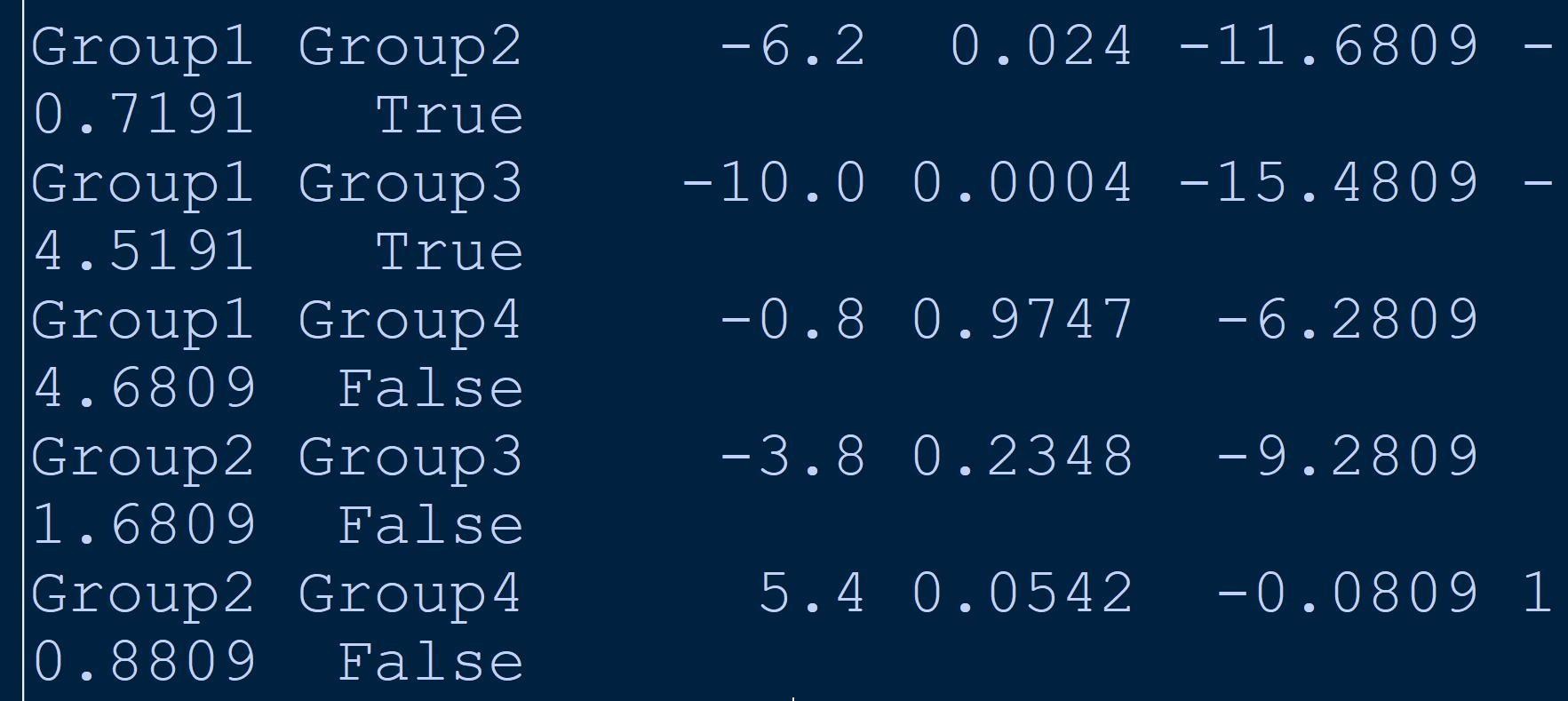
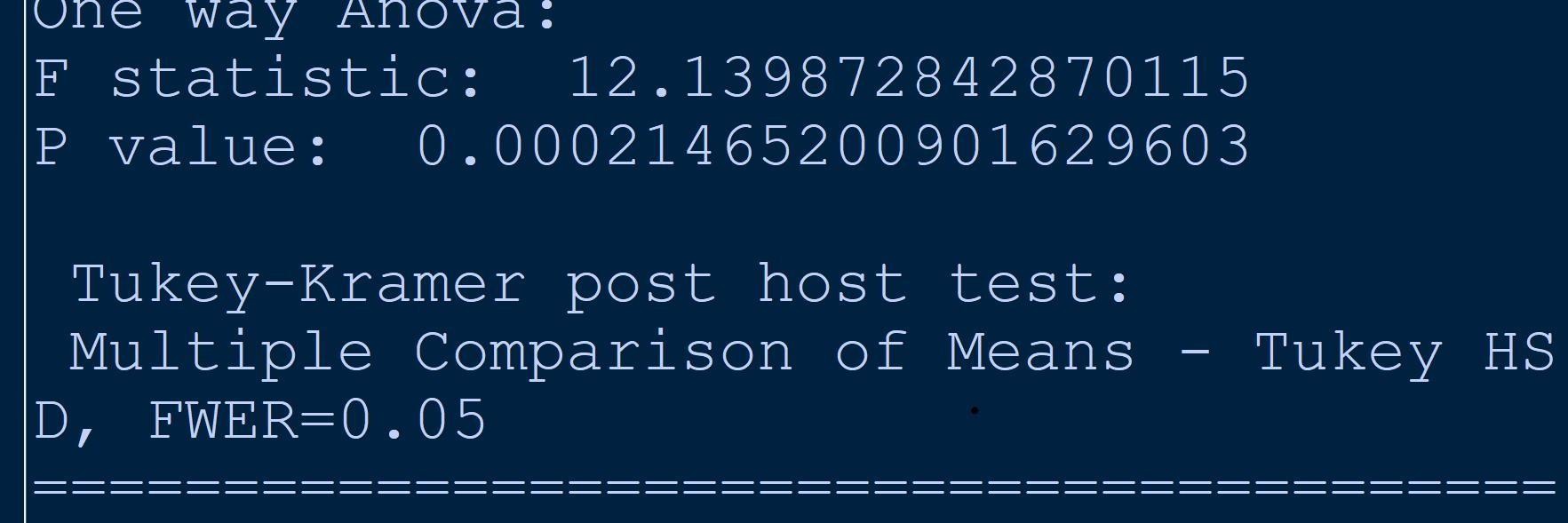
=====================================================

group1 group2 meandiff p-adj lowerupper reject

Group1 Group2 -6.2 0.024 -11.6809 -0.7191 True

Group1 Group3 -10.0 0.0004 -15.4809 -4.5191 True

|  |  |  |
| --- | --- | --- |
| Group1 Group4 | -0.8 0.9747 -6.2809 4.6809 | False |
| Group2 Group3 | -3.8 0.2348 -9.2809 1.6809 | False |
| Group2 Group4 | 5.4 0.0542 -0.0809 10.8809 | False |
| Group3 Group4 | 9.2 0.001 3.7191 14.6809 | True |



Practical No 6

Aim:

Regression and Its Types Implement simple linear regression using a dataset. Explore and interpret the regression model coefficients and goodness-of-fit measures. Extend the analysis to multiple linear regression and assess the impact of additional predictors.

Code:

import numpy as np import pandas as pd

from sklearn.datasets import fetch\_california\_housing from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error,r2\_score

housing = fetch\_california\_housing()

housing\_df = pd.DataFrame(housing.data,columns=housing.feature\_names) print(housing\_df)

housing\_df['PRICE'] = housing.target x = housing\_df[['AveRooms']]

y = housing\_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)

mse = mean\_squared\_error(y\_test,y\_pred) r2 = r2\_score(y\_test,y\_pred)

print("Mean Squared Error: ",mse) print("R-Squared: ",r2) print("Intercept: ",model.intercept\_) print("Coefficient: ",model.coef\_)

### Multiple Linear Regression:

import numpy as np import pandas as pd

from sklearn.datasets import fetch\_california\_housing

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error,r2\_score

housing = fetch\_california\_housing()

housing\_df = pd.DataFrame(housing.data,columns=housing.feature\_names) print(housing\_df)

housing\_df['PRICE'] = housing.target

x = housing\_df.drop('PRICE',axis=1) y = housing\_df['PRICE']

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=42) model = LinearRegression()

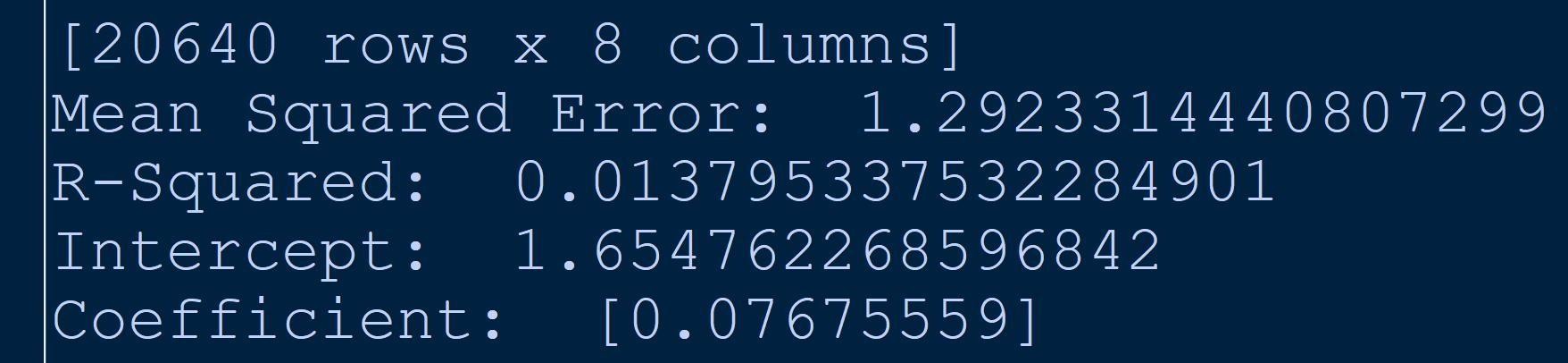
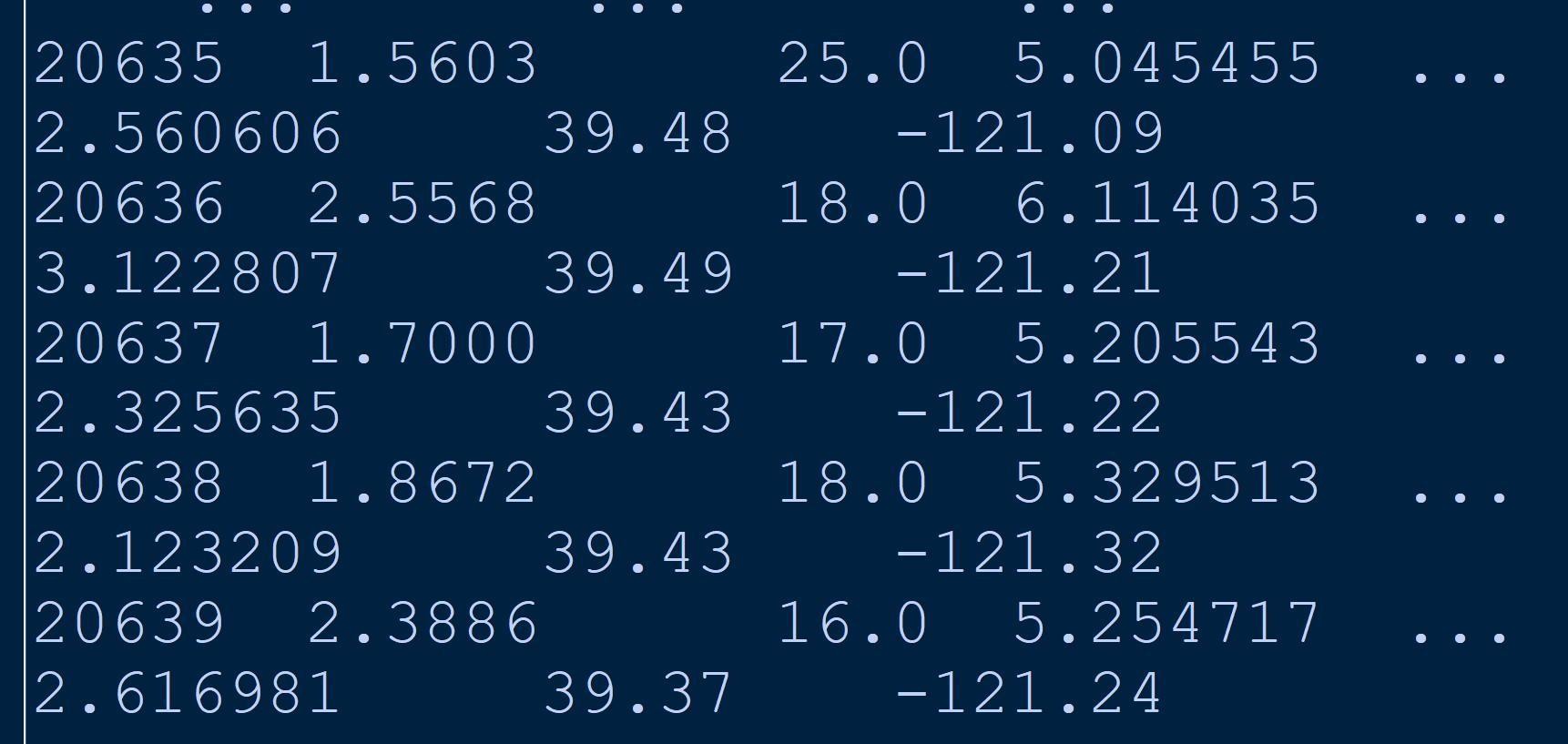
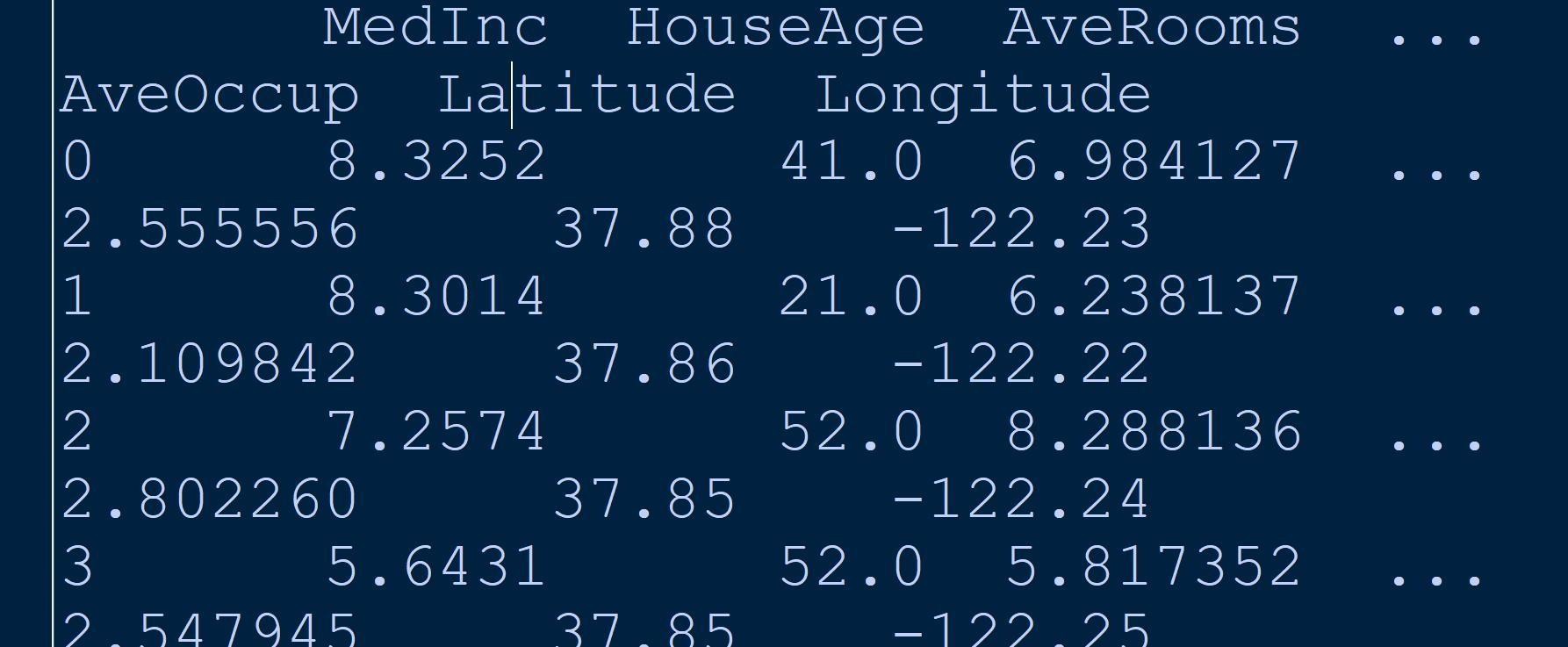
model.fit(x\_train,y\_train)

y\_pred = model.predict(x\_test)

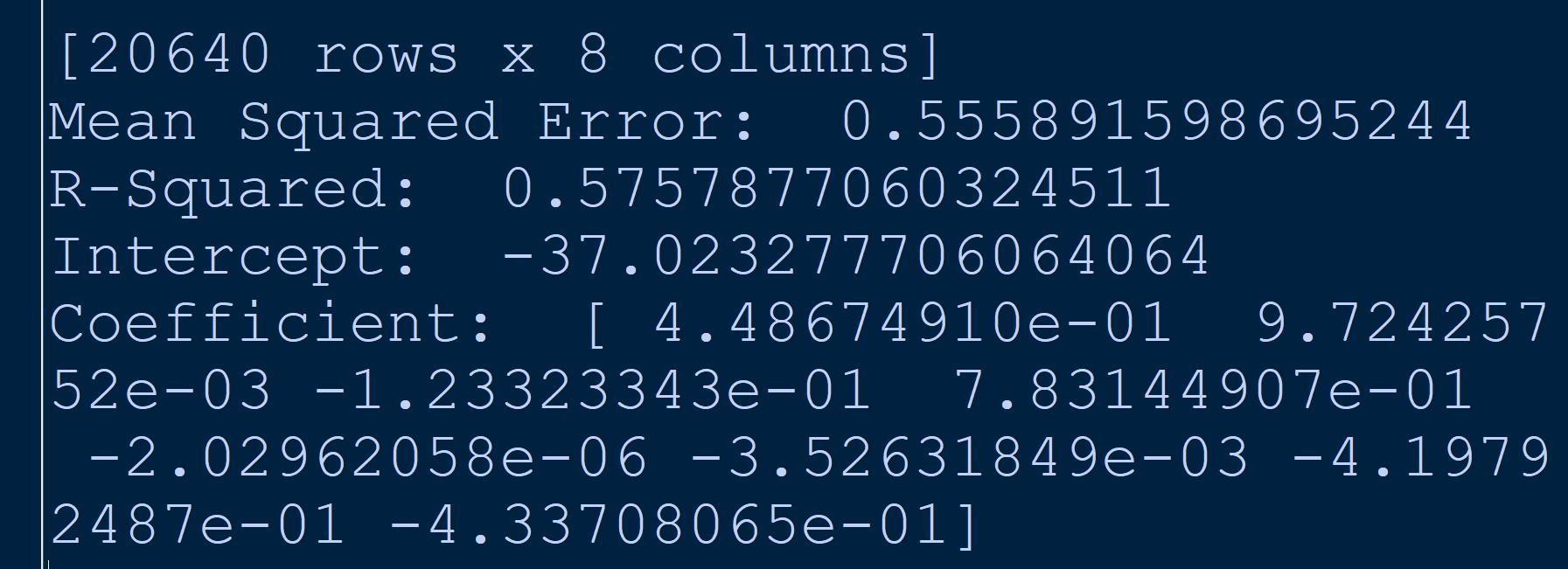
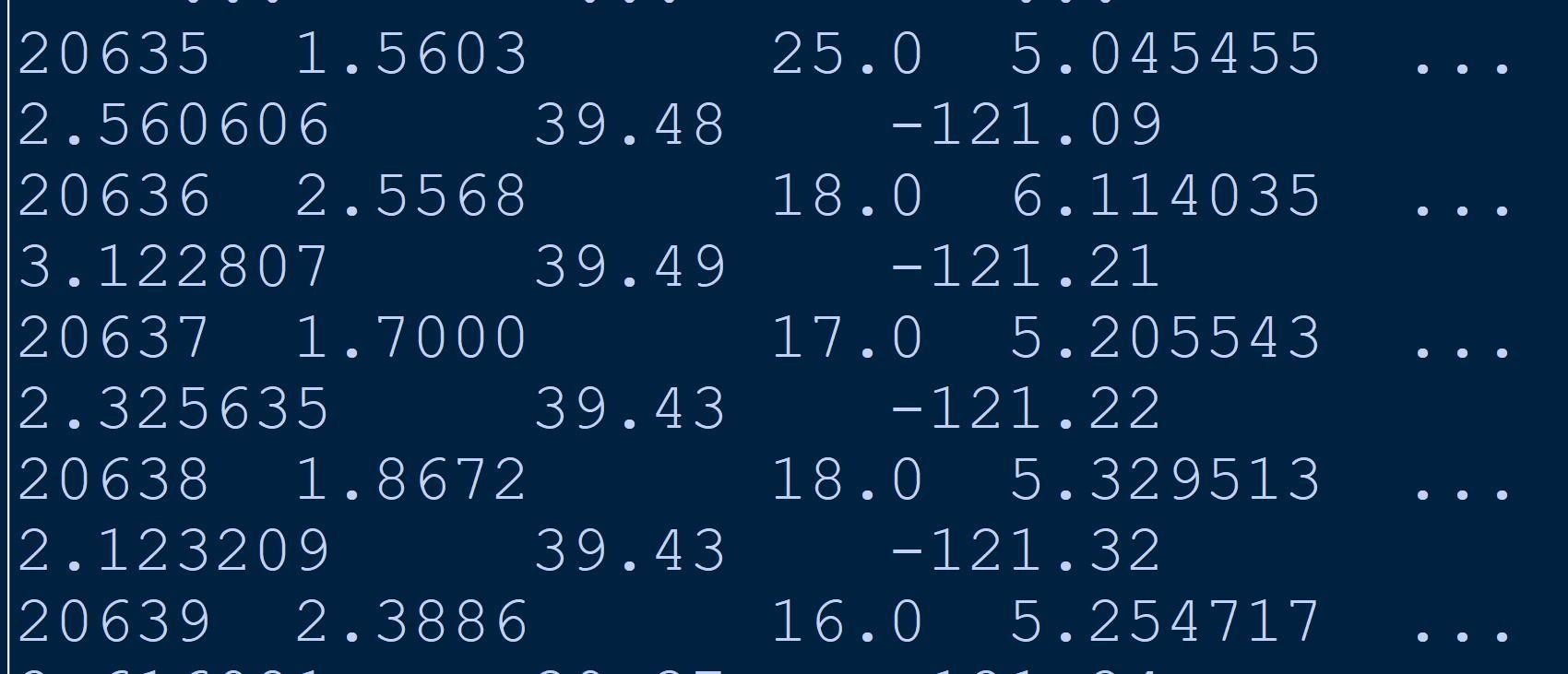
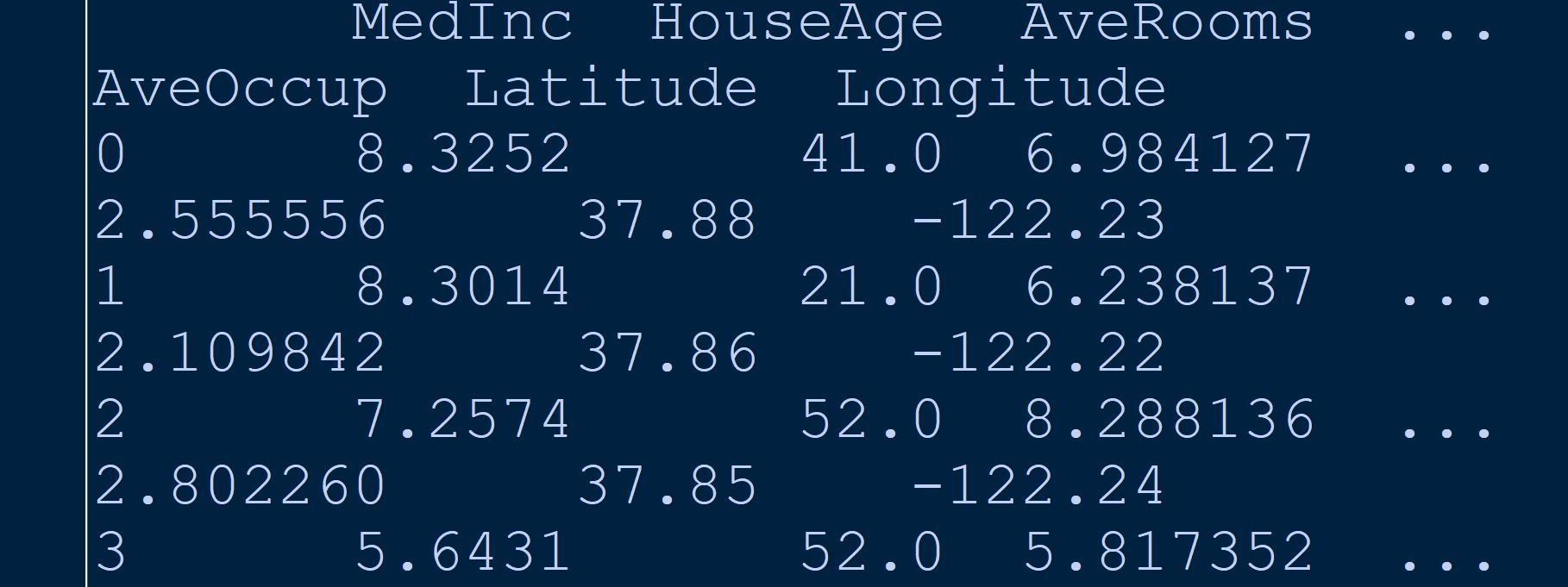
mse = mean\_squared\_error(y\_test,y\_pred) r2 = r2\_score(y\_test,y\_pred)

print("Mean Squared Error: ",mse) print("R-Squared: ",r2) print("Intercept: ",model.intercept\_) print("Coefficient: ",model.coef\_)

Output:



**Multiple Linear Regression:**



Practical No 7

Aim:Logistic Regression and Decision Tree

Build a logistic regression model to predict a binary outcome.

Evaluate the model's performance using classification metrics (e.g., accuracy, precision, recall).

Construct a decision tree model and interpret the decision rules for classification.

Code:

import numpy as np import pandas as pd

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LogisticRegression from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score,precision\_score,recall\_score,classification\_report

iris = load\_iris() iris\_df =

pd.DataFrame(data=np.c\_[iris['data'],iris['target']],columns=iris['feature\_names']+['target']) binary\_df = iris\_df[iris\_df['target']!=2]

x = binary\_df.drop('target',axis=1) y = binary\_df['target']

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=42) logistic\_model = LogisticRegression()

logistic\_model.fit(x\_train,y\_train)

y\_pred\_logistic = logistic\_model.predict(x\_test)

print("Logistic Regression Metrics: ")

print("Accuracy: ",accuracy\_score(y\_test,y\_pred\_logistic)) print("Precision: ",precision\_score(y\_test,y\_pred\_logistic)) print("Recall: ",recall\_score(y\_test,y\_pred\_logistic))

print("\n Classification Report: ") print(classification\_report(y\_test,y\_pred\_logistic))

decision\_tree\_model = DecisionTreeClassifier()

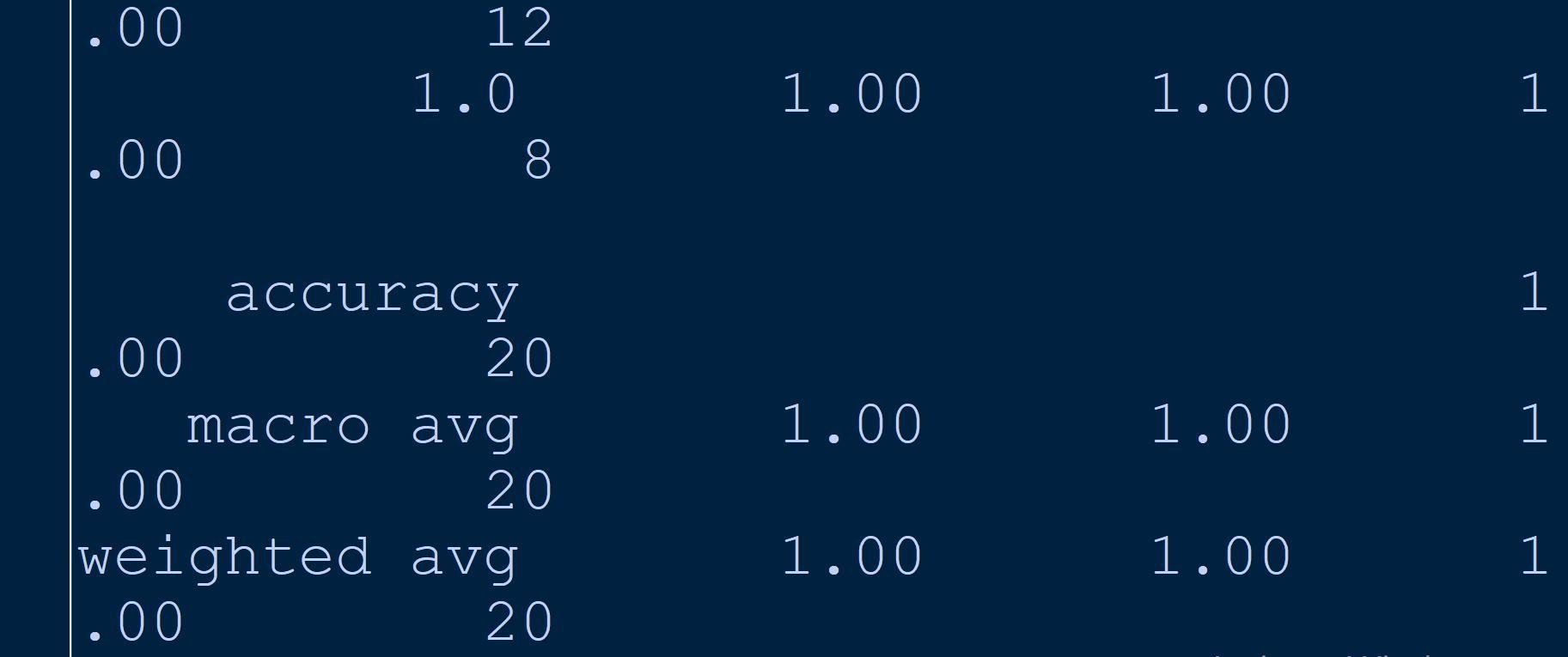
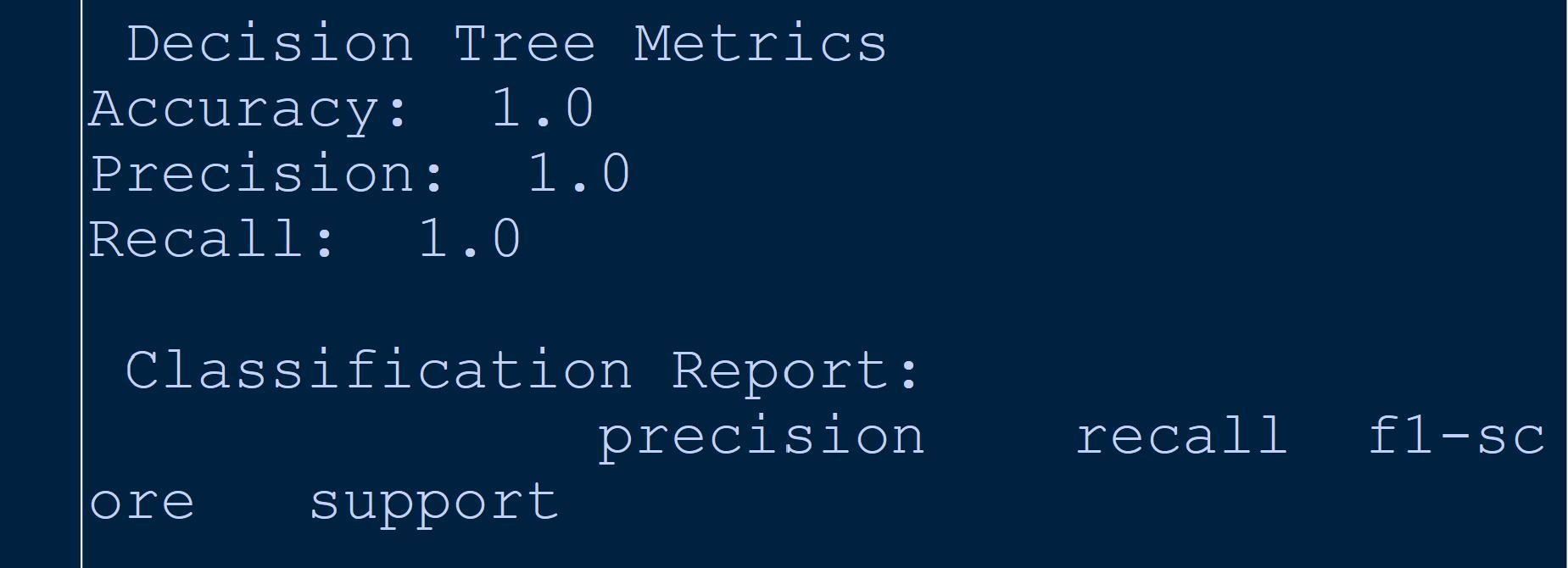
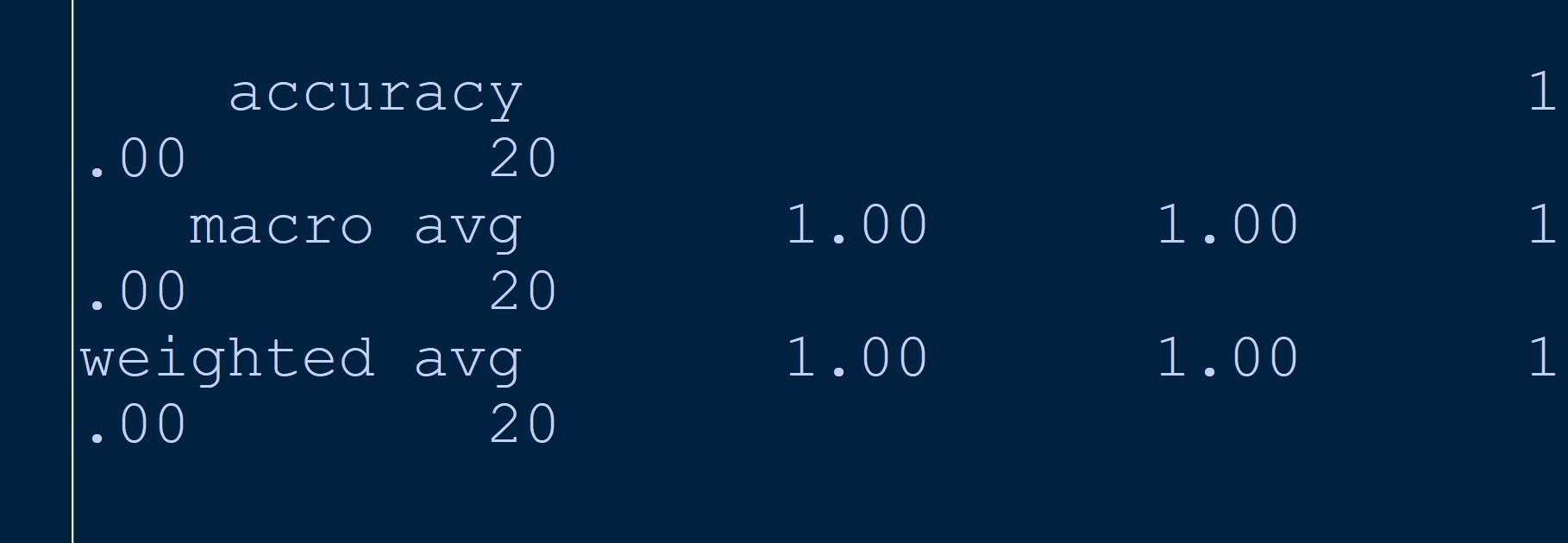
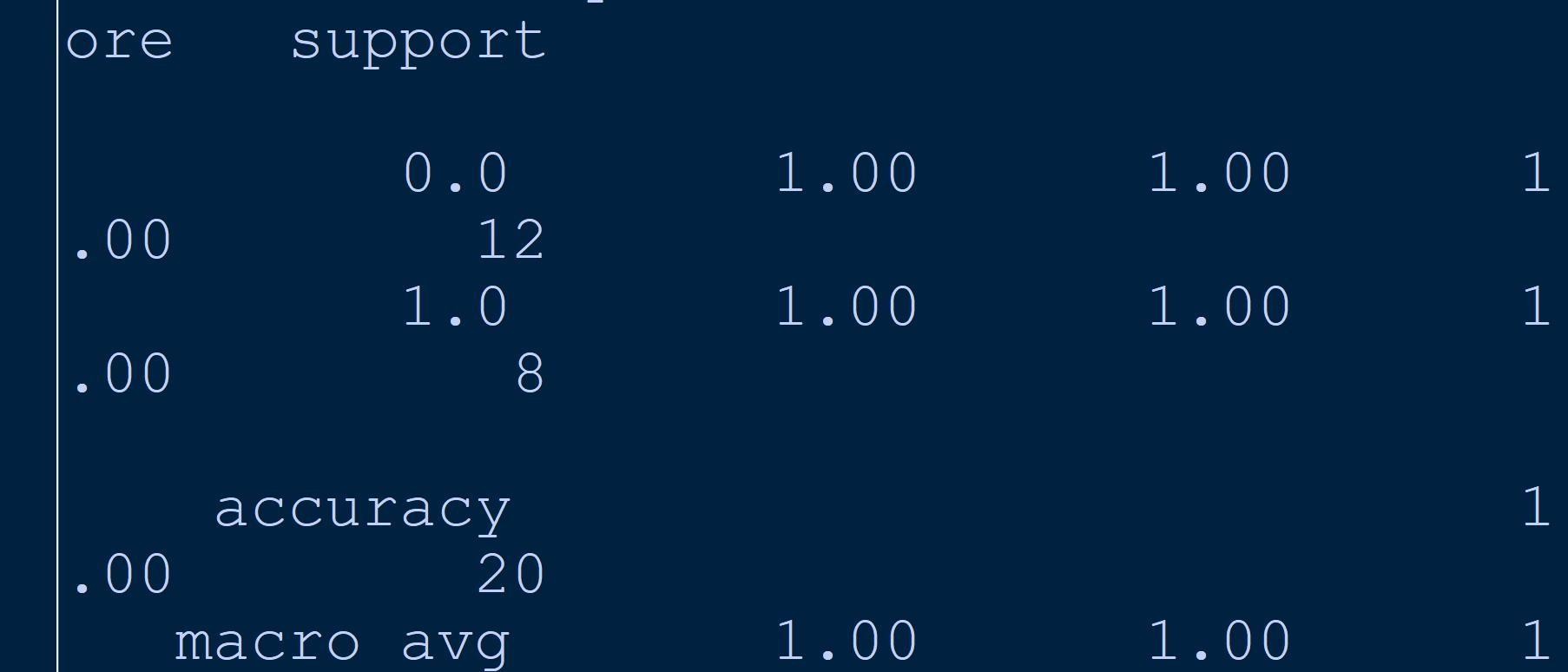
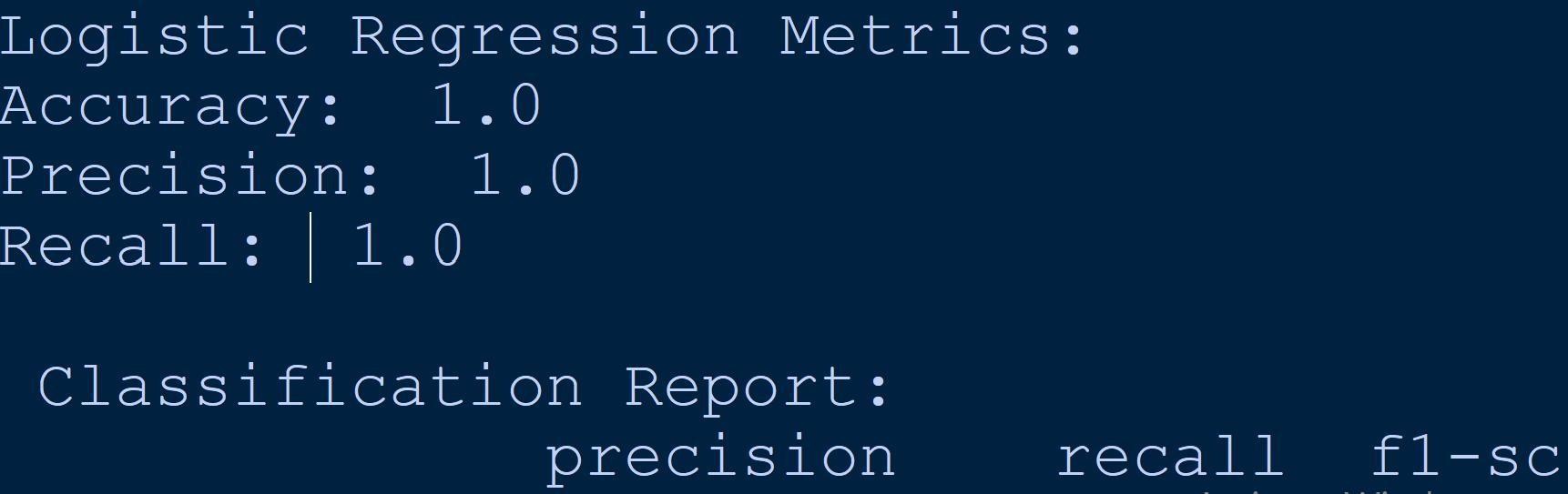
decision\_tree\_model.fit(x\_train,y\_train) y\_pred\_tree = decision\_tree\_model.predict(x\_test)

print("\n Decision Tree Metrics")

print("Accuracy: ",accuracy\_score(y\_test,y\_pred\_logistic)) print("Precision: ",precision\_score(y\_test,y\_pred\_logistic)) print("Recall: ",recall\_score(y\_test,y\_pred\_logistic))

print("\n Classification Report: ") print(classification\_report(y\_test,y\_pred\_tree))

Output:



Practical No 8

Aim:K-Means Clustering

Apply the K-Means algorithm to group similar data points into clusters. Determine the optimal number of clusters using elbow method or silhouette analysis.

Visualize the clustering results and analyze the cluster characteristics.

Code:

import numpy as np import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler from sklearn.decomposition import PCA

iris = load\_iris()

x = pd.DataFrame(data=iris['data'], columns=iris['feature\_names']) x\_scaled = StandardScaler().fit\_transform(x)

wcss = [KMeans(n\_clusters=i, init='k-means++', random\_state=42).fit(x\_scaled).inertia\_ for i in range(1, 11)]

plt.figure(figsize=(8,6))

plt.plot(range(1,11), wcss, marker='o', linestyle='--') plt.title('Elbow Method')

plt.xlabel('Number of Clusters') plt.ylabel('WCSS')

plt.show()

kmeans = KMeans(n\_clusters=3, init='k-means++', random\_state=42).fit(x\_scaled) x\_pca = PCA(n\_components=2).fit\_transform(x\_scaled)

plt.figure(figsize=(8,6))

plt.scatter(x\_pca[:,0], x\_pca[:,1], c=kmeans.labels\_, cmap='viridis', s=50, alpha=0.5) plt.scatter(kmeans.cluster\_centers\_[:,0], kmeans.cluster\_centers\_[:,1], marker='o', c='red', s=200, edgecolor='k')

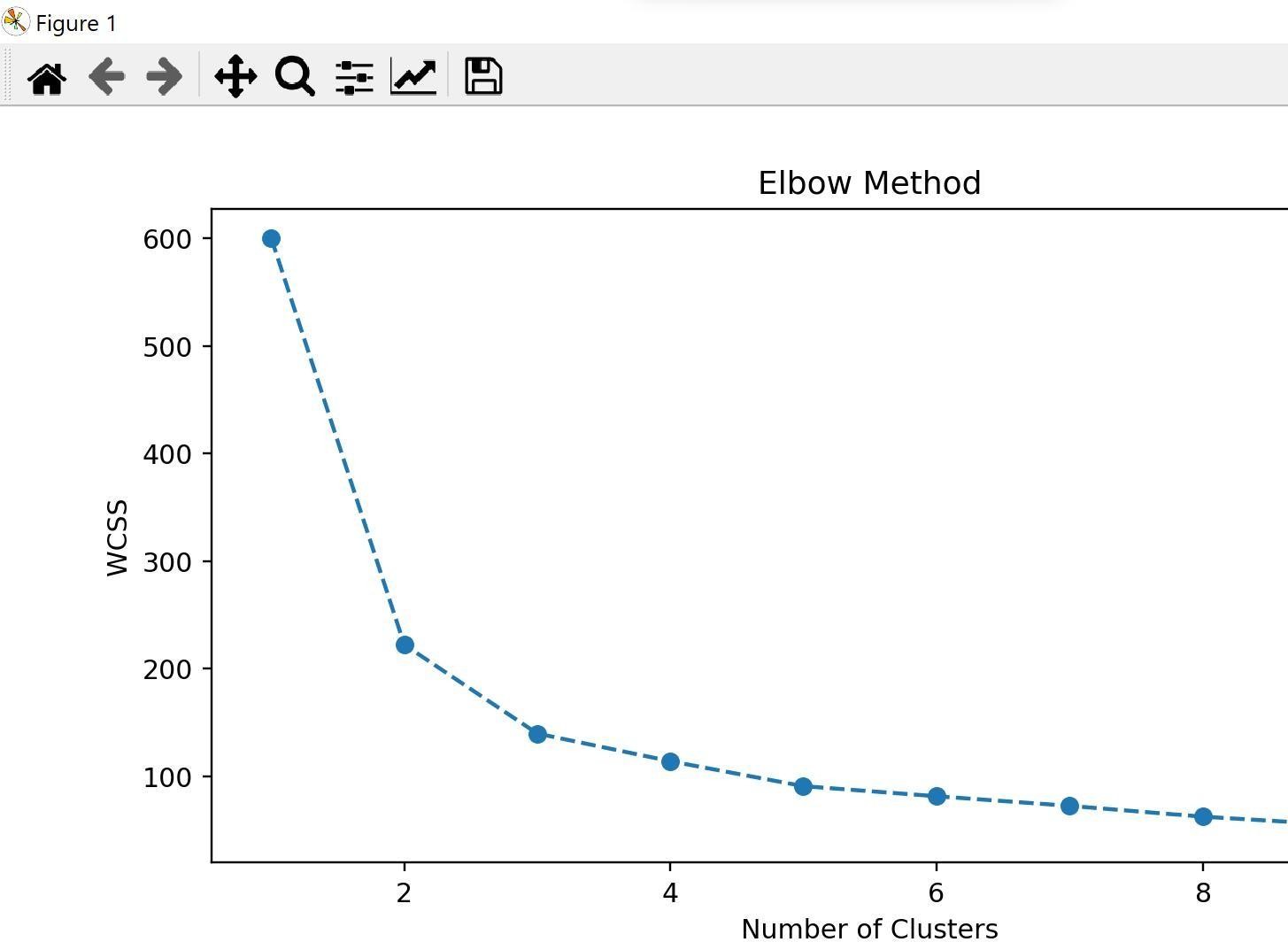
plt.title('K-Means Clustering') plt.xlabel('Principal Component 1')

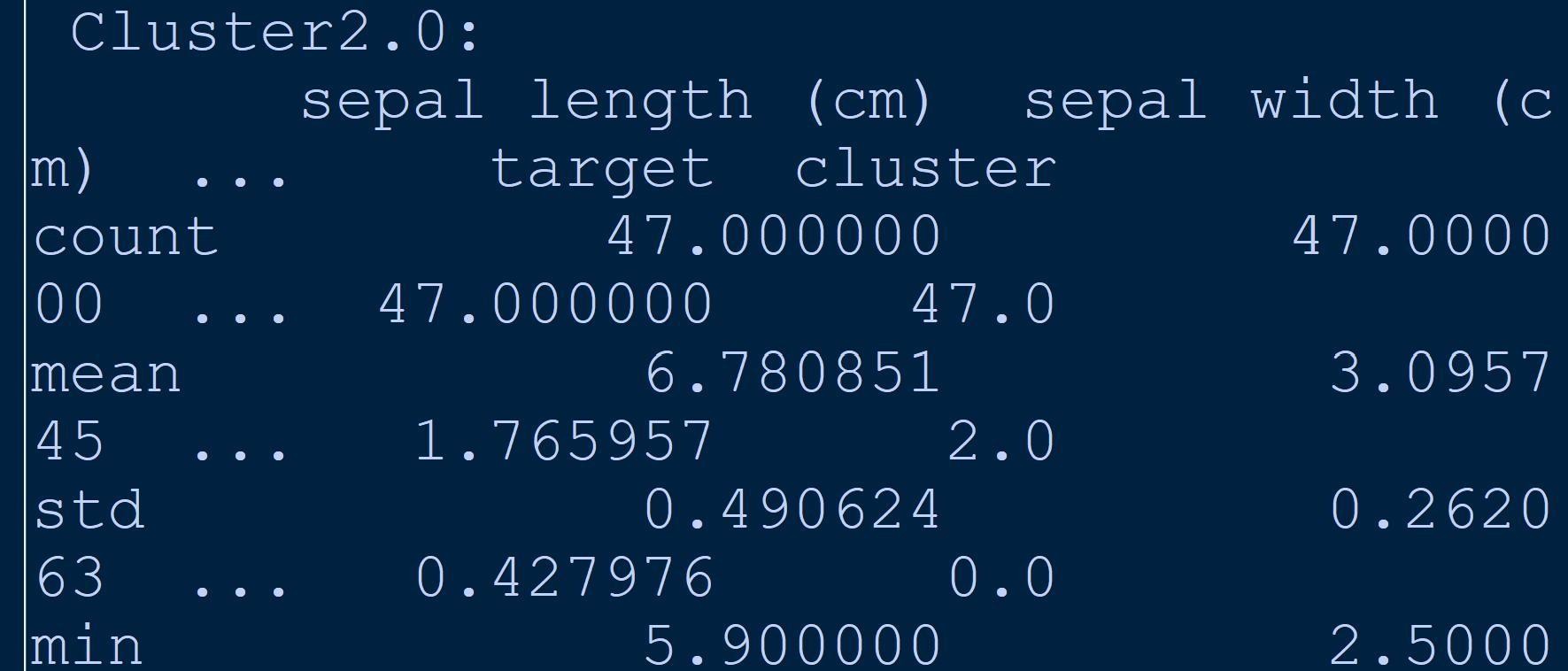
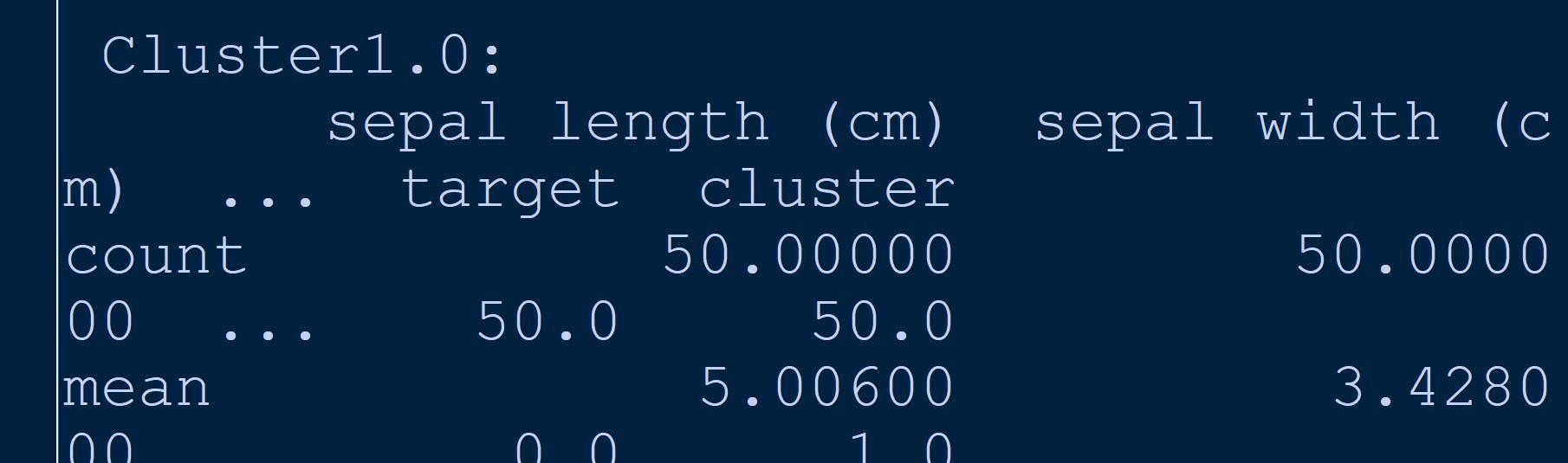
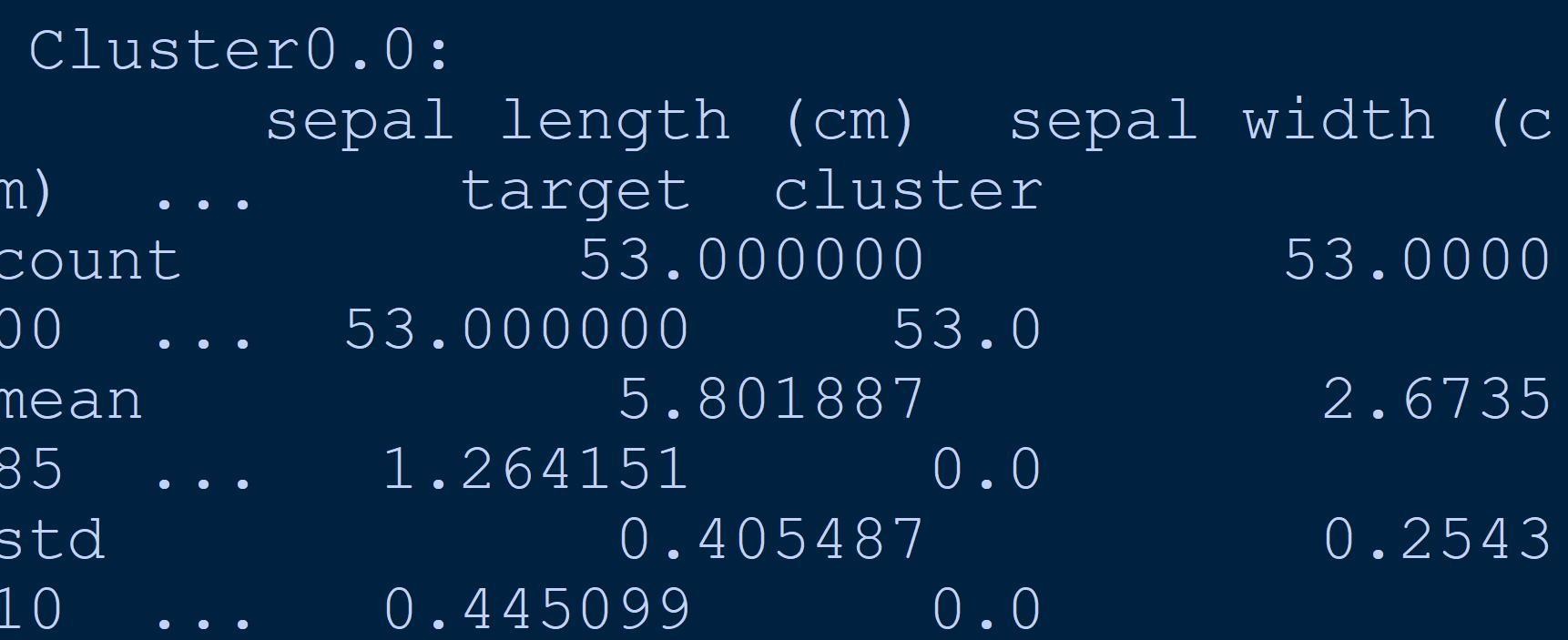
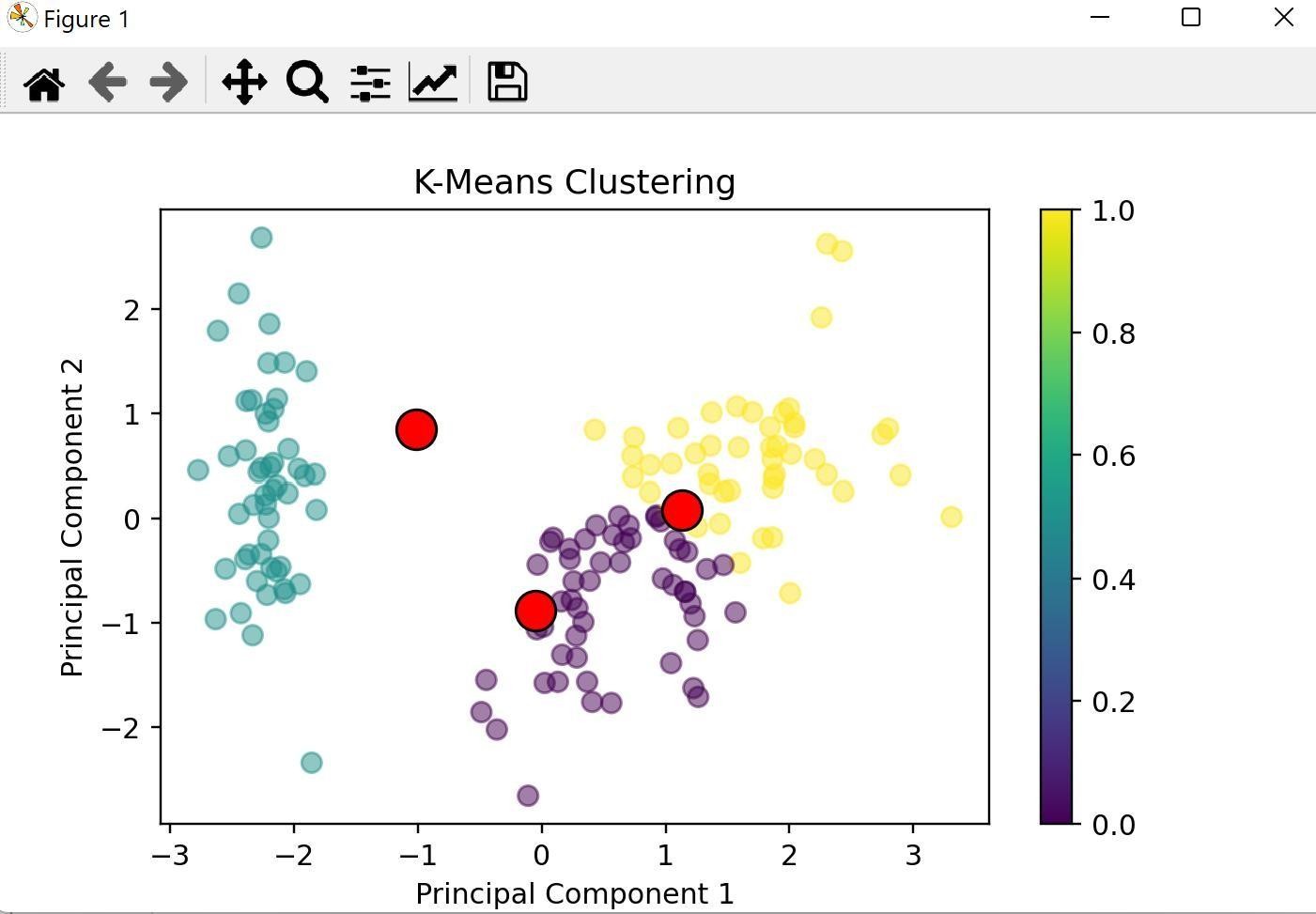
plt.ylabel('Principal Component 2') plt.colorbar()

plt.show()

cluster\_df = pd.DataFrame(data=np.c\_[iris['data'], iris['target'], kmeans.labels\_], columns=iris['feature\_names']+['target', 'cluster'])

for cluster in sorted(cluster\_df['cluster'].unique()): print(f"\n Cluster{cluster}:")

print(cluster\_df[cluster\_df['cluster']==cluster].describe()) Output:



Practical 9

Aim:Principal Component Analysis (PCA)

Perform PCA on a dataset to reduce dimensionality.

Evaluate the explained variance and select the appropriate number of principal components.

Visualize the data in the reduced-dimensional space.

from sklearn import datasets

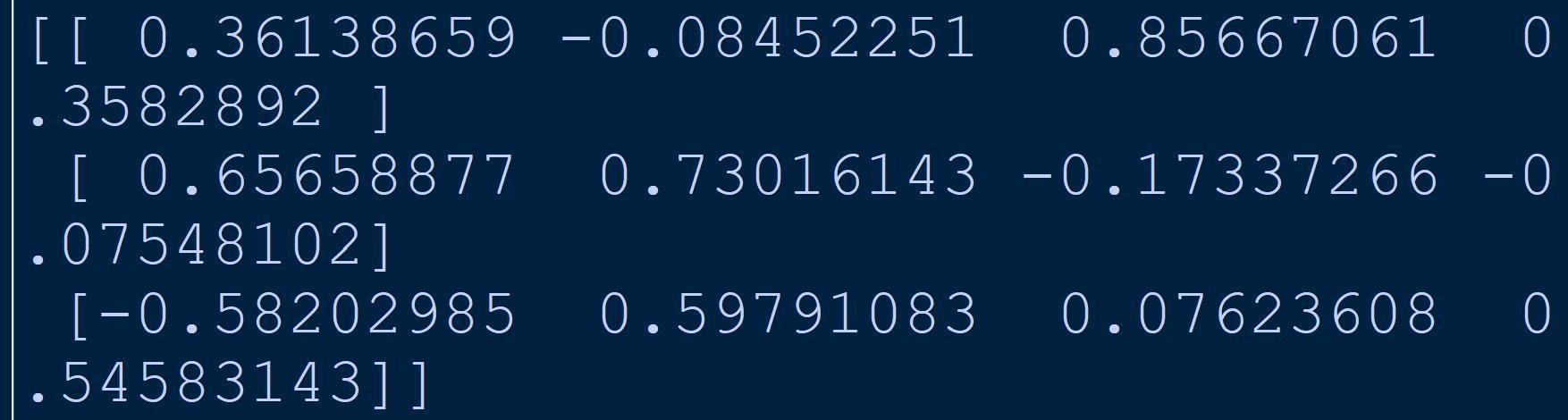
iris\_data = datasets.load\_iris(as\_frame=True) df = iris\_data.data

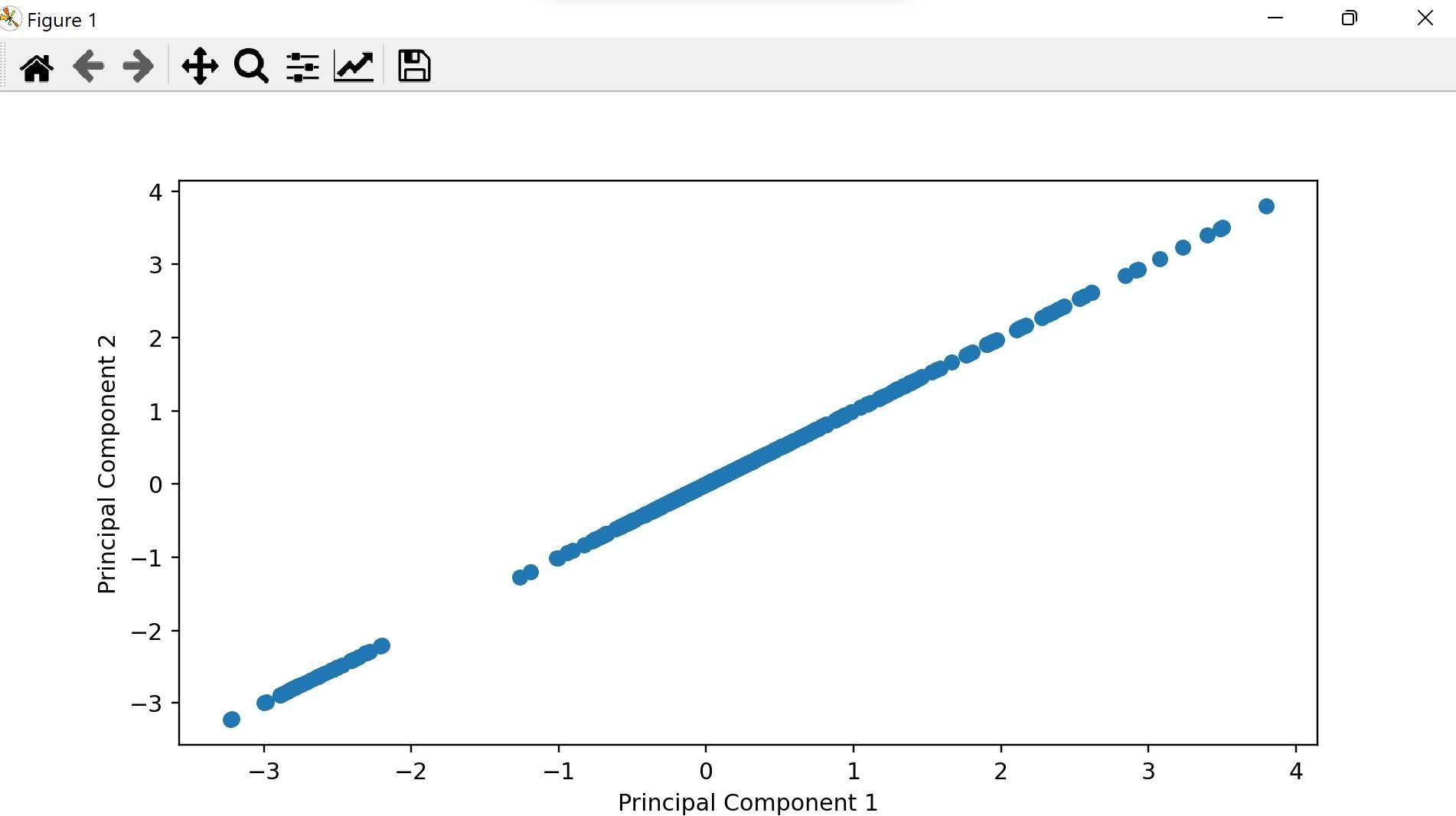
from sklearn.decomposition import PCA pca = PCA(n\_components=3)

to\_plot = pca.fit\_transform(df) print(pca.components\_) import matplotlib.pyplot as plt plt.scatter(to\_plot,to\_plot)

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2') plt.show()

Output:



## Practical 10

Aim:Data Visualization and Storytelling

import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

import numpy as np

# Generate random data

np.random.seed(42) # Set a seed for reproducibility # Create a DataFrame with random data

data = pd.DataFrame({

'variable1': np.random.normal(0, 1, 1000),

'variable2': np.random.normal(2, 2, 1000) + 0.5 \* np.random.normal(0, 1, 1000),

'variable3': np.random.normal(-1, 1.5, 1000),

'category': pd.Series(np.random.choice(['A', 'B', 'C', 'D'], size=1000, p=[0.4, 0.3, 0.2, 0.1]), dtype='category')

})

# Create a scatter plot to visualize the relationship between two variables plt.figure(figsize=(10, 6))

plt.scatter(data['variable1'], data['variable2'], alpha=0.5) plt.title('Relationship between Variable 1 and Variable 2', fontsize=16) plt.xlabel('Variable 1', fontsize=14)

plt.ylabel('Variable 2', fontsize=14) plt.show()Roll No : CS21016/1108251 Name : Jerusha Joshua

Monteiro

# Create a bar chart to visualize the distribution of a categorical variable plt.figure(figsize=(10, 6))

sns.countplot(x='category', data=data) plt.title('Distribution of Categories', fontsize=16) plt.xlabel('Category', fontsize=14) plt.ylabel('Count', fontsize=14) plt.xticks(rotation=45)

plt.show()

# Create a heatmap to visualize the correlation between numerical variables plt.figure(figsize=(10, 8))

numerical\_cols = ['variable1', 'variable2', 'variable3'] sns.heatmap(data[numerical\_cols].corr(), annot=True, cmap='coolwarm') plt.title('Correlation Heatmap', fontsize=16)

plt.show()

# Data Storytelling

print("Title: Exploring the Relationship between Variable 1 and Variable 2")

print("\nThe scatter plot (Figure 1) shows the relationship between Variable 1 and Variable 2. We can observe a positive correlation, indicating that as Variable 1 increases, Variable 2 tends to increase as well. However, there is a considerable amount of scatter, suggesting that other factors may influence this relationship.")

print("\nScatter Plot")

print("Figure 1: Scatter Plot of Variable 1 and Variable 2")

print("\nTo better understand the distribution of the categorical variable 'category', we created a bar chart (Figure 2). The chart reveals that Category A has the highest frequency, followed by Category B, Category C, and Category D. This information could be useful for further analysis or decision-making processes.")Roll No : CS21016/1108251

Name : Jerusha Joshua Monteiro

print("\nBar Chart")

print("Figure 2: Distribution of Categories")

print("\nAdditionally, we explored the correlation between numerical variables using a heatmap (Figure 3). The heatmap shows that Variable 1 and Variable 2 have a strong positive correlation, confirming the observation from the scatter plot. However, we can also see that Variable 3 has a moderate negative correlation with both Variable 1 and Variable 2, suggesting that it may have an opposing effect on the relationship between the first two variables.")

print("\nHeatmap")

print("Figure 3: Correlation Heatmap")

print("\nIn summary, the visualizations and analysis provide insights into the relationships between variables, the distribution of categories, and the correlations between numerical variables. These findings can be used to inform further analysis, decision-making, or to generate new hypotheses for investigation.")

Output:

