CONTENT

Date: 25-07-2024

Experiment No. 1

Aim: To import csv file, perform data manipulation and analysis using pandas

CO1: Practice the concepts of Data collection and wrangling.

CO2: Apply EDA techniques for data pre-processing.

Procedure:

```
import pandas as pd
data = pd.read_csv("house-prices.csv")
data = data.dropna()
x = data[['SqFt', 'Bedrooms', 'Bathrooms']]
y = data['Price']
print(x)
print(y)
```

Output:

```
     SqFt  Bedrooms  Bathrooms
0    1790         2          2
1    2030         4          2
2    1740         3          2
3    1980         3          2
4    2130         3          3
..    ...       ...        ...
123  1900         3          3
124  2160         4          3
125  2070         2          2
126  2020         3          3
127  2250         3          3

[128 rows x 3 columns]
0      114300
1      114200
2      114800
```

```
3       94700
4      119800
        ...
123    119700
124    147900
125    113500
126    149900
127    124600
Name: Price, Length: 128, dtype: int64
```

Result: The program was executed and the result was successfully obtained. Thus CO1 and CO2 is obtained.

Date: 25-07-2024

Experiment No. 2

Aim: Program to merge two dataframes

CO1: Practice the concepts of Data collection and wrangling.

Procedure:

```
import pandas as pd
df1=pd.read_csv("house-prices.csv")
df2=pd.read_csv("duplicatemis.csv")
df1=df1.dropna()
df2=df2.dropna()
merged_data=pd.merge(df1,df2,on='Home',how='outer')
print("Merged Dataframes:",merged_data)
```

Output:

```
Merged Dataframes:      Home  Price_x  SqFt_x  ...  Offers_y  Brick_y  Neighborhood_y
0       1   114300    1790  ...       NaN      NaN            NaN
1       2   114200    2030  ...       3.0       No           East
2       3   114800    1740  ...       NaN      NaN            NaN
3       4    94700    1980  ...       3.0       No           East
4       5   119800    2130  ...       NaN      NaN            NaN
..    ...      ...     ...  ...       ...      ...            ...
123   124   119700    1900  ...       3.0      Yes           East
124   125   147900    2160  ...       3.0      Yes           East
125   126   113500    2070  ...       2.0       No          North
126   127   149900    2020  ...       1.0       No           West
127   128   124600    2250  ...       4.0       No          North

[128 rows x 15 columns]
```

Result: The program was executed and the result was successfully obtained. Thus CO1 is obtained.

Date: 29-07-2024

Experiment No. 3

Aim: Write a program to upload dataset using pandas and perform the following

1. find the number of rows in the given dataset
2. find the number of columns in the given dataset
3. find all the rows having price greater than 1Lakh
4. Sort the square-feet value in ascending order

CO1: Practice the concepts of Data collection and wrangling.

Procedure:

```
import pandas as pd
data = pd.read_csv("house-prices.csv")
data1 = data.dropna()
print(f"Number of rows is : {len(data)}")
print(f"Number of columns is : {len(data.columns)}")
print(f"Price Greater than 100000: \n{data[data['Price']>100000]}")
print(f"Sorted SqFt: \n{data.sort_values(by='SqFt', ascending=True)}")
```

Output:

```
Number of rows is : 128
Number of columns is : 8
Price Greater than 100000:
     Home   Price  SqFt  Bedrooms  Bathrooms  Offers Brick Neighborhood
0       1  114300  1790         2          2       2    No         East
1       2  114200  2030         4          2       3    No         East
2       3  114800  1740         3          2       1    No         East
4       5  119800  2130         3          3       3    No         East
5       6  114600  1780         3          2       2    No        North
..    ...     ...   ...       ...        ...     ...   ...          ...
123   124  119700  1900         3          3       3   Yes         East
124   125  147900  2160         4          3       3   Yes         East
125   126  113500  2070         2          2       2    No        North
126   127  149900  2020         3          3       1    No         West
127   128  124600  2250         3          3       4    No        North
```

```
[116 rows x 8 columns]
Sorted SqFt:
     Home   Price  SqFt  Bedrooms  Bathrooms  Offers Brick Neighborhood
65     66  111100  1450         2          2       1   Yes        North
84     85   90500  1520         2          2       3    No        North
40     41  106600  1560         2          2       1    No         East
28     29   69100  1600         2          2       3    No        North
61     62  100900  1610         2          2       2    No        North
..    ...     ...   ...       ...        ...     ...   ...          ...
37     38  147000  2420         4          3       4    No         West
103   104  211200  2440         4          3       3   Yes         West
96     97  133300  2440         3          3       3    No         East
105   106  146900  2530         4          3       4    No         West
14     15  176800  2590         4          3       4    No         West
```

Result: The program was executed and the result was successfully obtained. Thus CO1 is obtained.

Date: 29-07-2024

Experiment No. 4

Aim: To familiars numpy library

CO1: Practice the concepts of Data collection and wrangling.

Procedure:

```
import numpy as np
data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print("Mean of the dataset:", np.mean(data))
print("Standard Deviation of the dataset:", np.std(data))
print("Median of the dataset:", np.median(data))
```

Output:

```
Mean of the dataset: 5.5
Standard Deviation of the dataset: 2.8722813232690143
Median of the dataset: 5.5
```

Result: The program was executed and the result was successfully obtained. Thus CO1 is obtained.

Date: 29-07-2024

Experiment No. 5

Aim: Declare 2 matrix using numpy library and perform various matrix operations

CO1: Practice the concepts of Data collection and wrangling.

Procedure:

```
import numpy as np
mat1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
mat2 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Sum of two matrix: \n", mat1+mat2)
print("Difference of two matrix: \n", mat1-mat2)
print("Multiplication of two matrix: \n", np.matmul(mat1,mat2))
print("Transpose of a matrix: \n", np.transpose(mat1))
print("Negative of a matrix: \n", np.invert(mat1))
```

Output:

```
Sum of two matrix:
 [[ 2  4  6]
 [ 8 10 12]
 [14 16 18]]
Difference of two matrix:
 [[0 0 0]
 [0 0 0]
 [0 0 0]]
Multiplication of two matrix:
 [[ 30  36  42]
 [ 66  81  96]
 [102 126 150]]
Transpose of a matrix:
 [[1 4 7]
 [2 5 8]
 [3 6 9]]
Negative of a matrix:
```

Result: The program was executed and the result was successfully obtained. Thus CO1 is obtained.

Date: 29-07-2024

Experiment No. 6

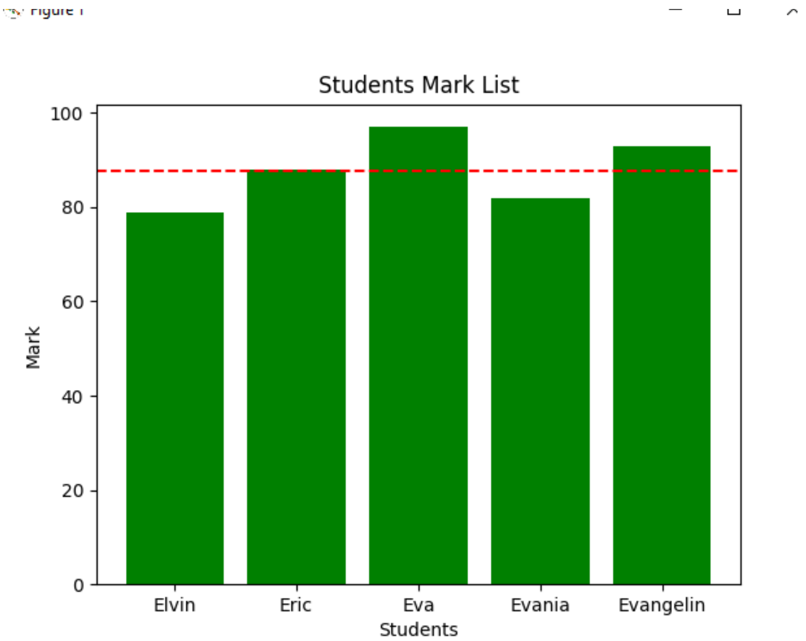
Aim: Write a program to read marks of students using numpy library. Calculate the average mark. Plot a bar graph, using matplotlib and depict the average mark obtained by the students.

CO3: Integrate data visualization in big-data analytics.

Procedure:

```
import numpy as np
import matplotlib.pyplot as plt
name = np.array(['Elvin','Eric','Eva','Evania','Evangelin'])
mark = np.array([79,88,97,82,93])
avg_mark=np.mean(mark)
plt.bar(name,mark,color='Green')
plt.ylabel('Mark')
plt.xlabel('Students')
plt.title("Students Mark List")
plt.axhline(avg_mark,color='red',linestyle="--")
plt.show()
```

Output:



Result: The program was executed and the result was successfully obtained. Thus CO3 is obtained.

Date: 29-07-2024

Experiment No. 7

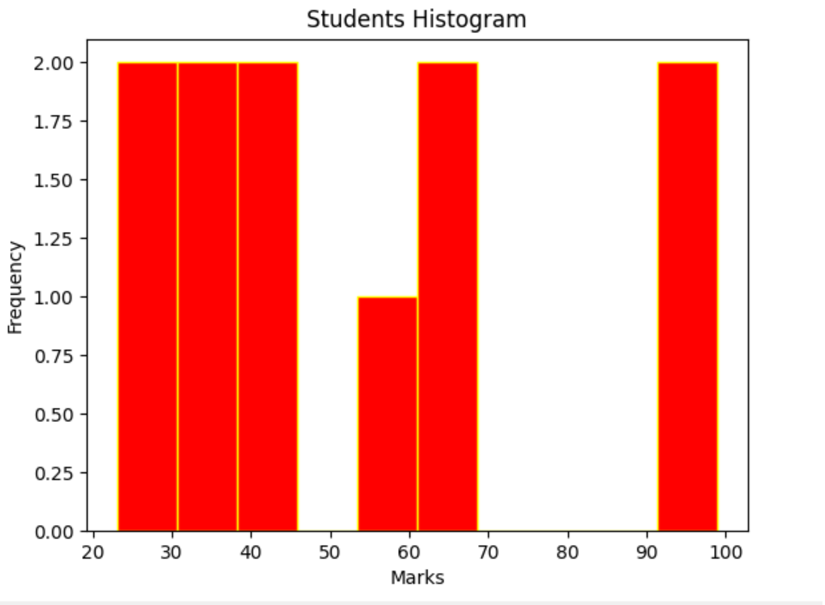
Aim: Analyse the given data using histogram

CO3: Integrate data visualization in big-data analytics.

Procedure:

```
import numpy as np
import matplotlib.pyplot as plt
mark = np.array([23,67,43,68,33,23,99,45,99,55,34])
plt.hist(mark,bins=10,color='red',edgecolor='yellow')
plt.title("Students Histogram")
plt.ylabel("Frequency")
plt.xlabel("Marks")
plt.show()
```

Output:



Result: The program was executed and the result was successfully obtained. Thus CO3 is obtained.

Date: 01-08-2024

Experiment No. 8

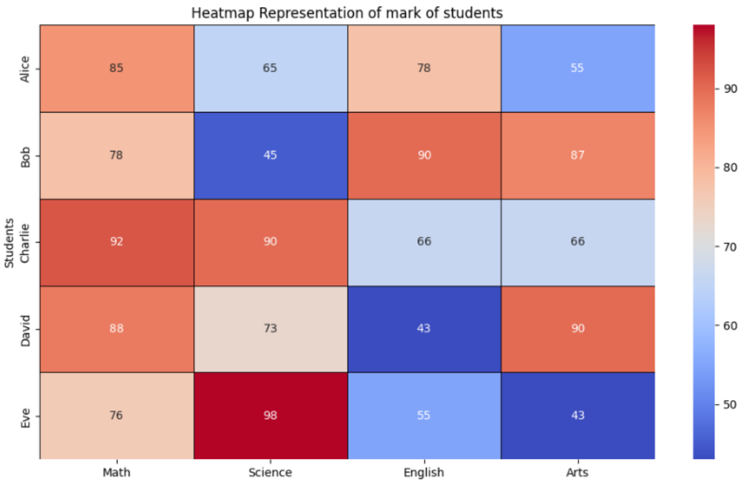
Aim: Using seaborn and matplotlib library, create a heatmap for the given dataset

CO3: Integrate data visualization in big-data analytics.

Procedure:

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Students': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Math': [85, 78, 92, 88, 76],
    'Science': [65, 45, 90, 73, 98],
    'English': [78, 90, 66, 43, 55],
    'Arts': [55, 87, 66, 90, 43]
}

df = pd.DataFrame(data)
df.set_index('Students', inplace=True)
plt.figure(figsize=(10, 6))
sns.heatmap(df, annot=True, cmap='coolwarm', linewidths=0.5, linecolor='black')
plt.title("Heatmap Representation of marks of students")
plt.show()
```

Output:



Result: The program was executed and the result was successfully obtained. Thus CO3 is obtained.

Date: 01-08-2024

Experiment No. 9

Aim: Using seaborn and matplotlib library, create a scatterplot for the given dataset

CO3: Integrate data visualization in big-data analytics.

Procedure:

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [25, 30, 35, 40, 45],
    'Salary': [50000, 60000, 70000, 80000, 90000]
}

df = pd.DataFrame(data)
plt.figure(figsize=(8, 5))
sns.scatterplot(x='Age', y='Salary', data=df, color='green',label='Salary')
sns.lineplot(x='Age', y='Salary', data=df, color=yellow,label='Trend line')
plt.title("Scatter plot of Age vs Salary")
plt.xlabel('Age')
plt.ylabel('Salary')
plt.show()
```

Output:

Result: The program was executed and the result was successfully obtained. Thus CO3 is obtained.

Date: 05-08-2024

Experiment No. 10

Aim: Select a dataset with at least three numerical variables (e.g., population, income, and education level by city). Create a bubble chart that represents the data by using bubble sizes and colours to encode information. Additionally, create a density chart (e.g., a 2D density plot) to show the concentration of data points. Answer the following questions:

i. How does the bubble chart help in visualizing multivariate data?

ii. What insights can you gain from the density chart in terms of data concentration?

iii. Are there any interesting patterns or outliers in the data?

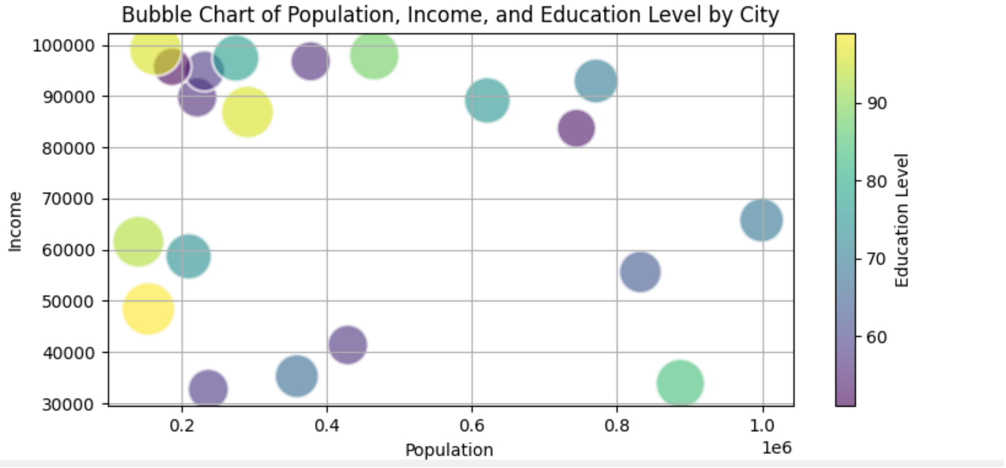
CO3: Integrate data visualization in big-data analytics.

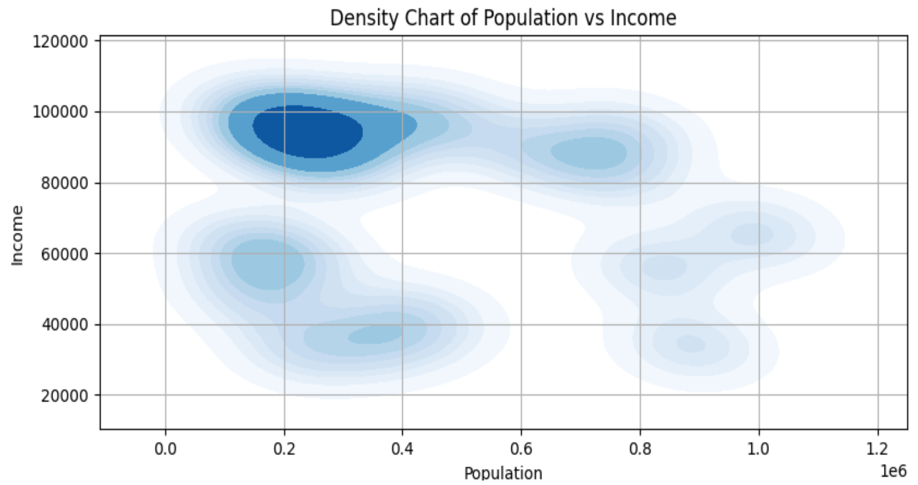
Procedure:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
np.random.seed(42)
cities = ['City ' + str(i) for i in range(1, 21)]
population = np.random.randint(100000, 1000000, size=20)
income = np.random.randint(30000, 100000, size=20)
education = np.random.randint(50, 100, size=20)
data = {
   'City': cities,
   'Population': population,
   'Income': income,
   'Education': education
}


df = pd.DataFrame(data)
df.head()
plt.figure(figsize=(14, 8))
scatter = plt.scatter(df['Population'], df['Income'], s=df['Education']*10,
c=df['Education'], cmap='viridis', alpha=0.6, edgecolors='w', linewidth=2)
plt.colorbar(scatter, label='Education Level')
plt.title('Bubble Chart of Population, Income, and Education Level by City')
plt.xlabel('Population')
plt.ylabel('Income')
```

```
plt.grid(True)
plt.show()
plt.figure(figsize=(14, 8))
sns.kdeplot(x=df['Population'], y=df['Income'], cmap='Blues', shade=True,
bw_adjust=0.5)
plt.title('Density Chart of Population vs Income')
plt.xlabel('Population')
plt.ylabel('Income')
plt.grid(True)
plt.show()
```

Output:





Result: The program was executed and the result was successfully obtained. Thus CO3 is obtained.

Date: 05-08-2024

Experiment No. 11

Aim: Illustrate the working of k- nearest neighbour algorithm using iris dataset

CO5: Apply Basic Machine Learning Algorithms

Procedure:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
iris = load_iris()
x = iris.data
y = iris.target
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.2,random_state=42)
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train, y_train)
V = knn.predict(x_test)
print(V)
result = accuracy_score(y_test, V)
print(result)
new_data_point = [[5.1, 3.5, 1.4, 0.2]]
prediction = knn.predict(new_data_point)
print(iris.target_names)
print(prediction)
predicted_species = iris.target_names[prediction]
print("New data point prediction :", prediction)
print("Predicted species for the new data point :", predicted_species)
```

Output:

```
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
0.9666666666666667
['setosa' 'versicolor' 'virginica']
[0]
New data point prediction : [0]
Predicted species for the new data point : ['setosa']
```

Result: The program was executed and the result was successfully obtained. Thus CO5 is obtained.

Date: 05-08-2024

Experiment No. 12

Aim: Illustrate the working of k- nearest neighbour algorithm using wine dataset

CO5: Apply Basic Machine Learning Algorithms

Procedure:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_wine
from sklearn.metrics import accuracy_score
wine = load_wine()
x = wine.data
y = wine.target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train, y_train)
V = knn.predict(x_test)
print(V)
result = accuracy_score(y_test, V)
print(result)
new_data_point = [[13.9, 1.69, 2.33, 15.2, 125, 2.6, 3.0, 1.28, 2.39, 6.64,    1.14, 4.92,
1063]]
prediction = knn.predict(new_data_point)
print(wine.target_names)
print(prediction)
predicted_species = wine.target_names[prediction]
print("New data point prediction :", prediction)
print("Predicted species for the new data point :", predicted_species)
```

Output:

```
[0 0 2 0 1 0 2 0 2 0 2 2 0 1 0 1 1 2 0 1 0 1 2 1 1 1 1 2 1 0 0 1 2 0 0 0]
0.6944444444444444
['class_0' 'class_1' 'class_2']
[0]
New data point prediction : [0]
Predicted species for the new data point : ['class_0']
```

Result: The program was executed and the result was successfully obtained. Thus CO5 is obtained.

Date: 08-08-2024

Experiment No. 13

Aim: Upload the given csv file and perform the following operations.

1. Read and write data files.
2. Sort the data by age.
3. Create a subset data where age>30(perform filtration)
4. Find average salary per department (perform aggregation operations on data)
5. Quantify missing values per column
6. Fill missing salary with the mean salary(filling)
7. Drop duplicate values if any
8. Perform one hot encoding to gender and department.

CO1: Practice the concepts of Data collection and wrangling.

CO2: Apply EDA techniques for data pre-processing.

Procedure:

```
import pandas as pd
df = pd.read_csv('New_Data.csv')
print(df)
df_sorted = df.sort_values(by='age')
print("\nSorted Data by Age:")
print(df_sorted)
subset_df = df[df['age'] > 30]
print("\nSubset Data where Age > 30:")
print(subset_df)
aggregated_df=df.groupby('department')['salary'].mean()
print("\nMean Salary of each department:")
print(aggregated_df)
missing_val=df.isnull().sum()
print("\nMissing values per column:")
print(missing_val)
df['salary'].fillna(df['salary'].mean(),inplace=True)
print("\nData after filling the missing values:")
print(df)
df.drop_duplicates(inplace=True)
print(df)
encoded_df=pd.get_dummies(df,columns=['gender','department'])
print("\nEncoded Data:")
print(encoded_df)
```

Output:

```
   id     name  age  gender   salary department
0   1     John   28    Male  50000.0      Sales
1   2     Jane   32  Female  60000.0  Marketing
2   3      Bob   25    Male  45000.0      Sales
3   4    Alice   30  Female  70000.0         HR
4   5  Charlie   35    Male      NaN         IT
5   6    David   40    Male  80000.0    Finance
6   7      Eve   29  Female  52000.0      Sales
7   8    Frank   33    Male  62000.0         IT
8   9    Grace   31  Female  73000.0  Marketing
9  10     Hank   36    Male  68000.0         HR
```

```
Sorted Data by Age:
   id     name  age  gender   salary department
2   3      Bob   25    Male  45000.0      Sales
0   1     John   28    Male  50000.0      Sales
6   7      Eve   29  Female  52000.0      Sales
3   4    Alice   30  Female  70000.0         HR
8   9    Grace   31  Female  73000.0  Marketing
1   2     Jane   32  Female  60000.0  Marketing
7   8    Frank   33    Male  62000.0         IT
4   5  Charlie   35    Male      NaN         IT
9  10     Hank   36    Male  68000.0         HR
5   6    David   40    Male  80000.0    Finance
```

```
Subset Data where Age > 30:
   id     name  age  gender   salary department
1   2     Jane   32  Female  60000.0  Marketing
4   5  Charlie   35    Male      NaN         IT
5   6    David   40    Male  80000.0    Finance
7   8    Frank   33    Male  62000.0         IT
8   9    Grace   31  Female  73000.0  Marketing
9  10     Hank   36    Male  68000.0         HR
```

```
Mean Salary of each department:
department
Finance      80000.0
HR           69000.0
IT           62000.0
Marketing    66500.0
Sales        49000.0
Name: salary, dtype: float64
```

```
Missing values per column:
id            0
name          0
age           0
gender        0
salary        1
department    0
dtype: int64
```

```
Data after filling the missing values:
   id     name  age  gender        salary department
0   1     John   28    Male  50000.000000      Sales
1   2     Jane   32  Female  60000.000000  Marketing
2   3      Bob   25    Male  45000.000000      Sales
3   4    Alice   30  Female  70000.000000         HR
4   5  Charlie   35    Male  62222.222222         IT
5   6    David   40    Male  80000.000000    Finance
6   7      Eve   29  Female  52000.000000      Sales
7   8    Frank   33    Male  62000.000000         IT
8   9    Grace   31  Female  73000.000000  Marketing
9  10     Hank   36    Male  68000.000000         HR
```

```
   id     name  age  gender        salary department
0   1     John   28    Male  50000.000000      Sales
1   2     Jane   32  Female  60000.000000  Marketing
2   3      Bob   25    Male  45000.000000      Sales
3   4    Alice   30  Female  70000.000000         HR
4   5  Charlie   35    Male  62222.222222         IT
5   6    David   40    Male  80000.000000    Finance
6   7      Eve   29  Female  52000.000000      Sales
7   8    Frank   33    Male  62000.000000         IT
8   9    Grace   31  Female  73000.000000  Marketing
9  10     Hank   36    Male  68000.000000         HR
```

```
   department_Finance  department_HR  department_IT  department_Marketing  \
0               False          False          False                 False
1               False          False          False                  True
2               False          False          False                 False
3               False           True          False                 False
4               False          False           True                 False
5                True          False          False                 False
6               False          False          False                 False
7               False          False           True                 False
8               False          False          False                  True
9               False           True          False                 False

   department_Sales
0              True
1             False
2              True
3             False
4             False
5             False
6              True
7             False
```

Result: The program was executed and the result was successfully obtained. Thus CO1 and CO2 is obtained.

Date: 08-08-2024

Experiment No. 14

Aim: Illustrate the working of naive bayes algorithm using iris dataset.

CO5: Apply Basic Machine Learning Algorithms

Procedure:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
iris = load_iris()
x = iris.data
y = iris.target
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.2, random_state=42)
gn = GaussianNB()
gn.fit(x_train, y_train)
V = gn.predict(x_test)
print(V)
result = accuracy_score(y_test, V)
print("Accuracy :",result)
new_data_point = [[5.1, 3.5, 1.4, 0.2]]
prediction = gn.predict(new_data_point)
print(iris.target_names)
print(prediction)
predicted_species = iris.target_names[prediction]
print("New data point prediction :", prediction)
print("Predicted species for the new data point :", predicted_species)
```

Output:

```
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
Accuracy : 1.0
['setosa' 'versicolor' 'virginica']
[0]
New data point prediction : [0]
Predicted species for the new data point : ['setosa']

Process finished with exit code 0
```

Result: The program was executed and the result was successfully obtained. Thus CO5 is obtained.

Date: 12-08-2024

Experiment No. 15

Aim: Create a CSV file based on the given data and perform the following operations.

1. Drop Missing Values
2. Filter the rows where the temperature is above 25 degrees.
3. Sort the data by Humidity in descending order
4. Group the data by the Date column and calculates the mean of the Temperature, Humidity, WindSpeed, and Precipitation columns for each date.
5. Display the heatmap , showing the correlations between Temperature, Humidity, WindSpeed, and Precipitation.
6. Generate a scatter plot to visualize the relationship between Temperature and Humidity in the dataset.
7. Generate a scatter plot to visualize the relationship between WindSpeed and Precipitation in the dataset.
8. Generate a histogram to visualize the distribution of Temperature in the dataset.
9. Generate a histogram to visualize the distribution of Humidity in the dataset.
10. Create a bubble chart to visualize the relationship between Temperature, Humidity, and WindSpeed in the dataset. Use Temperature and Humidity as the x and y axes, respectively, and use WindSpeed to determine the size of the bubbles.
11. Create a line chart to visualize the trend of Temperature over time.
12. Create a bar chart to compare the average Humidity for each date.

CO1: Practice the concepts of Data collection and wrangling.

CO2: Apply EDA techniques for data pre-processing.

Procedure:

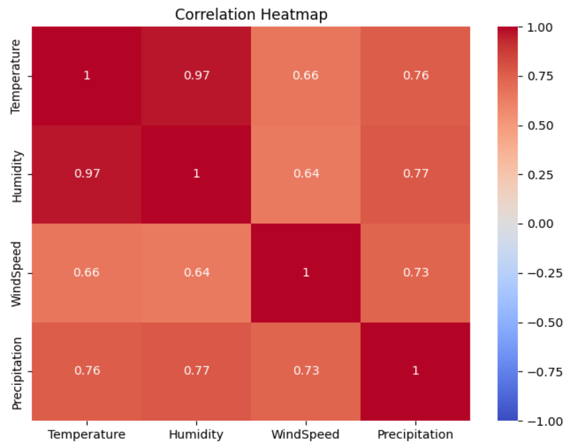
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('weather_data.csv')
df = df.dropna()
filtered_df = df[df['Temperature'] > 25]
print(filtered_df)
sorted_df = df.sort_values(by='Humidity', ascending=False)
print(sorted_df)
grouped_df = df.groupby('Date').mean()
correlation_matrix = df[['Temperature', 'Humidity', 'WindSpeed', 'Precipitation']].corr()
print(correlation_matrix)
plt.figure(figsize=(8, 6))
```

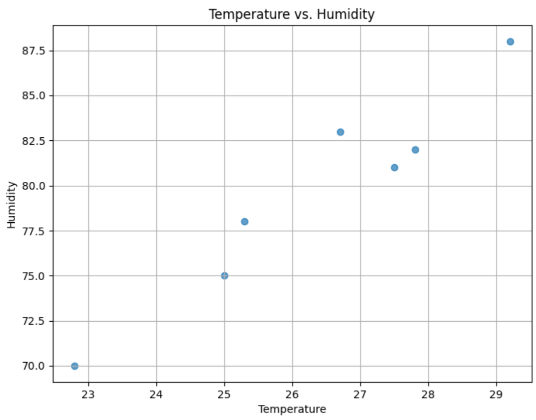
```
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Heatmap')
plt.show()
plt.figure(figsize=(8, 6))
plt.scatter(df['Temperature'], df['Humidity'], alpha=0.7)
plt.title('Temperature vs. Humidity')
plt.xlabel('Temperature')
plt.ylabel('Humidity')
plt.grid(True)
plt.show()
plt.figure(figsize=(8, 6))
plt.scatter(df['WindSpeed'], df['Precipitation'], alpha=0.7)
plt.title('WindSpeed vs. Precipitation')
plt.xlabel('WindSpeed')
plt.ylabel('Precipitation')
plt.grid(True)
plt.show()
plt.figure(figsize=(8, 6))
plt.hist(df['Temperature'], bins=10, edgecolor='k', alpha=0.7)
plt.title('Distribution of Temperature')
plt.xlabel('Temperature')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
plt.figure(figsize=(8, 6))
plt.hist(df['Humidity'], bins=10, edgecolor='k', alpha=0.7)
plt.title('Distribution of Humidity')
plt.xlabel('Humidity')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
plt.figure(figsize=(8, 6))
plt.scatter(df['Temperature'], df['Humidity'], s=df['WindSpeed']*10, alpha=0.5,
edgecolors='w', linewidth=0.5)
plt.title('Bubble Chart of Temperature, Humidity, and WindSpeed')
plt.xlabel('Temperature')
plt.ylabel('Humidity')
plt.show()
df['Date'] = pd.to_datetime(df['Date'])
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['Temperature'], marker='o')
plt.title('Temperature Trend Over Time')
plt.xlabel('Date')
```

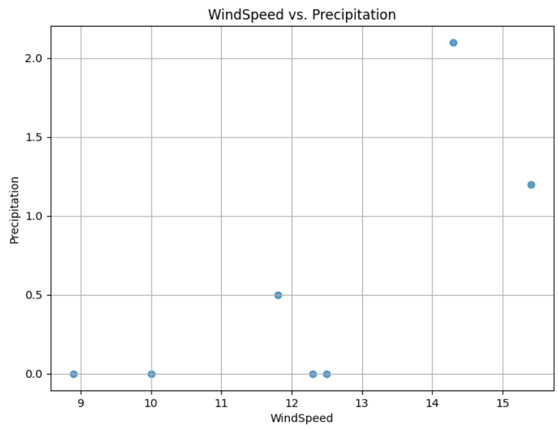
```
plt.ylabel('Temperature')
plt.grid(True)
plt.xticks(rotation=45)
plt.show()
average_humidity = df.groupby('Date')['Humidity'].mean()
plt.figure(figsize=(10, 6))
average_humidity.plot(kind='bar')
plt.title('Average Humidity for Each Date')
plt.xlabel('Date')
plt.ylabel('Average Humidity')
plt.grid(True)
plt.xticks(rotation=45)
plt.show()
```

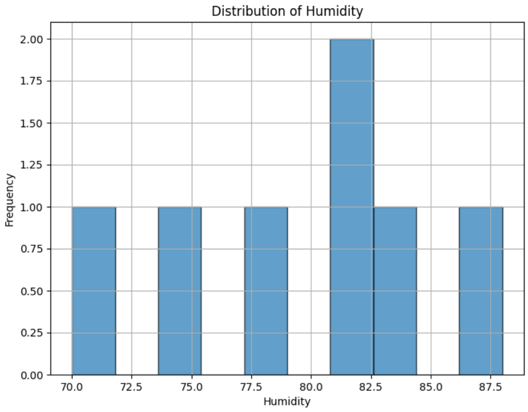
Output:

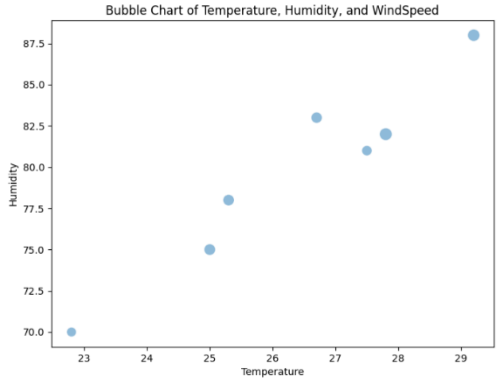
```
         Date  Temperature  Humidity  WindSpeed  Precipitation
0  2024-08-01         25.3      78.0       12.3            0.0
1  2024-08-02         27.8      82.0       15.4            1.2
6  2024-08-07         29.2      88.0       14.3            2.1
7  2024-08-08         26.7      83.0       11.8            0.5
8  2024-08-09         27.5      81.0       10.0            0.0
         Date  Temperature  Humidity  WindSpeed  Precipitation
6  2024-08-07         29.2      88.0       14.3            2.1
7  2024-08-08         26.7      83.0       11.8            0.5
1  2024-08-02         27.8      82.0       15.4            1.2
8  2024-08-09         27.5      81.0       10.0            0.0
0  2024-08-01         25.3      78.0       12.3            0.0
9  2024-08-10         25.0      75.0       12.5            0.0
4  2024-08-05         22.8      70.0        8.9            0.0
               Temperature  Humidity  WindSpeed  Precipitation
Temperature       1.000000  0.965118   0.663675       0.761736
Humidity          0.965118  1.000000   0.642622       0.774804
WindSpeed         0.663675  0.642622   1.000000       0.733623
Precipitation     0.761736  0.774804   0.733623       1.000000
```

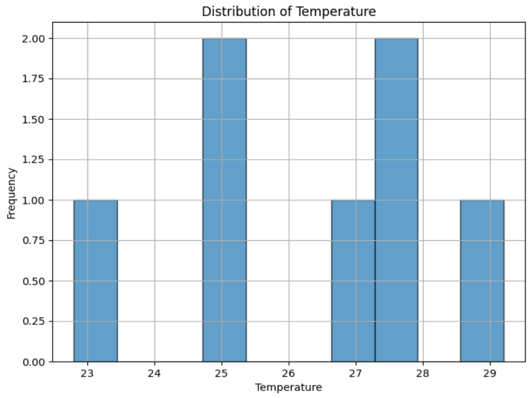


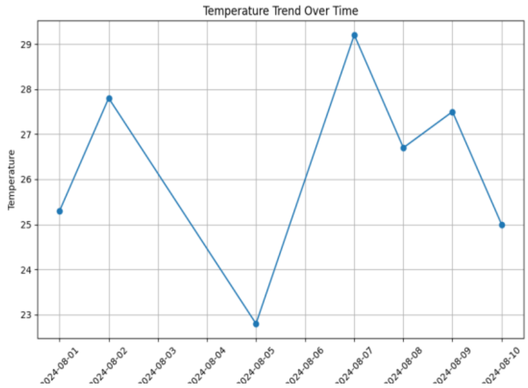


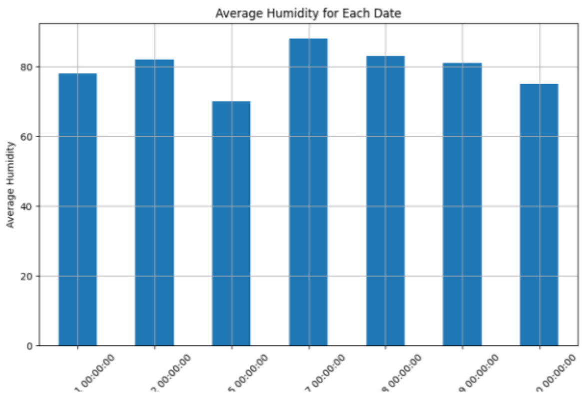












<u>Result:</u> The program was executed and the result was successfully obtained. Thus CO1 and CO2 is obtained.

Date: 12-08-2024

Experiment No. 16

Aim: Create a CSV file based on the given data and perform the following operations.

1. Drop Missing Values
2. Filter Rows by Sales
3. Sort Data by Price
4. Group Data by Date and Calculate Mean
5. Display a heatmap showing the correlations between Sales, Price, and Units Sold
6. Generate a scatter plot to visualize the relationship between Sales and Price in the dataset
7. Generate a histogram to visualize the distribution of Sales in the dataset.

CO1: Practice the concepts of Data collection and wrangling.

CO2: Apply EDA techniques for data pre-processing.

Procedure:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('sales_data.csv')
print(df.dtypes)
df = df.dropna()
filtered_df = df[df['Sales'] > 5000]
print(filtered_df)
sorted_df = filtered_df.sort_values(by='Price')
print(sorted_df)
grouped_df = df.groupby('Date')
print(grouped_df[['Sales', 'Price', 'Units_Sold']].mean())
correlation_matrix = df[['Sales', 'Price', 'Units_Sold']].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Heatmap')
plt.show()
plt.figure(figsize=(8, 6))
plt.scatter(df['Sales'], df['Price'], alpha=0.7)

plt.title('Sales vs. Price')
plt.xlabel('Sales')
plt.ylabel('Price')
plt.grid(True)
```

```
plt.show()
plt.figure(figsize=(8, 6))
plt.hist(df['Sales'], bins=10, edgecolor='k', alpha=0.7)
plt.title('Distribution of Sales')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

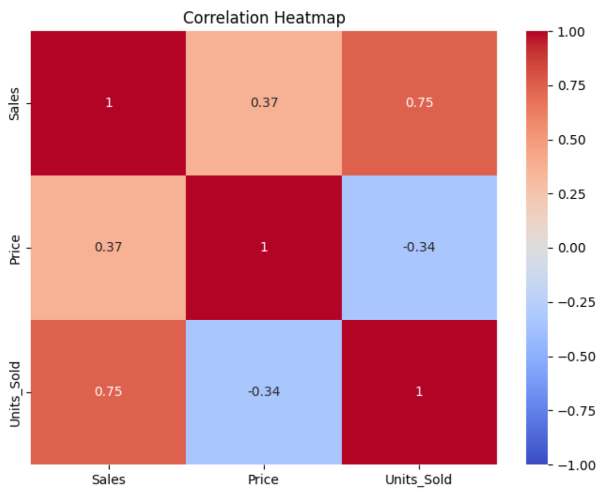
Output:

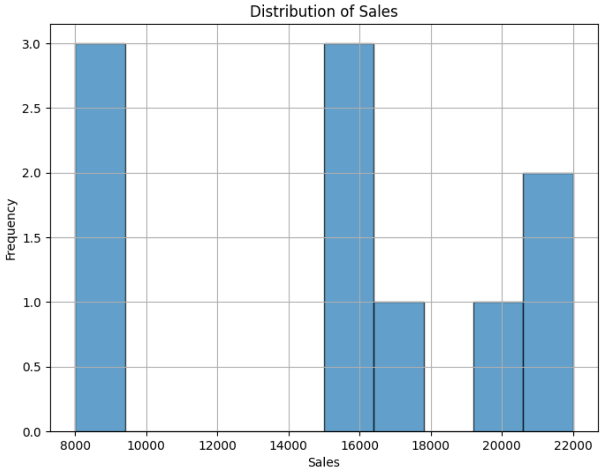
```
Date           object
Product        object
Sales           int64
Price           int64
Discount       object
Region         object
Units_Sold      int64
dtype: object
```

Date	Sales	Price	Units_Sold
2024-08-01	17500.0	625.0	30.0
2024-08-02	12500.0	600.0	20.5
2024-08-03	18500.0	640.0	28.5
2024-08-04	15250.0	475.0	31.5
2024-08-05	12250.0	600.0	21.0

	Date	Product	Sales	Price	Discount	Region	Units_Sold
0	2024-08-01	Laptop	15000	750	5%	North	20
1	2024-08-01	Smartphone	20000	500	10%	South	40
2	2024-08-02	Tablet	8000	400	7%	East	20
3	2024-08-02	Laptop	17000	800	5%	West	21
4	2024-08-03	Smartphone	21000	520	12%	North	38
5	2024-08-03	Laptop	16000	760	6%	South	19
6	2024-08-04	Tablet	8500	420	7%	East	21
7	2024-08-04	Smartphone	22000	530	10%	West	42
8	2024-08-05	Laptop	15500	770	5%	North	20
9	2024-08-05	Tablet	9000	430	8%	South	22

	Date	Product	Sales	Price	Discount	Region	Units_Sold
2	2024-08-02	Tablet	8000	400	7%	East	20
6	2024-08-04	Tablet	8500	420	7%	East	21
9	2024-08-05	Tablet	9000	430	8%	South	22
1	2024-08-01	Smartphone	20000	500	10%	South	40
4	2024-08-03	Smartphone	21000	520	12%	North	38
7	2024-08-04	Smartphone	22000	530	10%	West	42
0	2024-08-01	Laptop	15000	750	5%	North	20
5	2024-08-03	Laptop	16000	760	6%	South	19
8	2024-08-05	Laptop	15500	770	5%	North	20
3	2024-08-02	Laptop	17000	800	5%	West	21





Result: The program was executed and the result was successfully obtained. Thus CO1 and CO2 is obtained.

Date: 29-08-2024

Experiment No. 17

Aim: Consider a dataset representing sales data for a retail store, create a CSV file and perform the following aggregation operations.
1. What is the total revenue generated from all sales?
2. What is the total revenue for each day?
3. What is the total quantity sold for each product?
4. What is the average daily revenue?
5. What is the maximum revenue from a single sale?
6. What are the statistics (mean, median, min, max) of revenue for each product.

CO4: Apply mathematical and statistical functions for practicing data science.

Procedure:

```
import pandas as pd
data = pd.read_csv('data.csv')
df = pd.DataFrame(data)
total_revenue = df['Revenue'].sum()
print("Total Revenue:", total_revenue)
total_revenue_per_day = df.groupby('Date')['Revenue'].sum()
print("\nTotal Revenue per Day:")
print(total_revenue_per_day)
total_quantity_per_product = df.groupby('Product')['Quantity'].sum()
print("\nTotal Quantity per Product:")
print(total_quantity_per_product)
average_daily_revenue = df.groupby('Date')['Revenue'].sum().mean()
print("\nAverage Daily Revenue:", average_daily_revenue)
max_revenue_single_sale = df['Revenue'].max()
print("\nMaximum Revenue from a Single Sale:", max_revenue_single_sale)
statistics_per_product = df.groupby('Product')['Revenue'].agg(['mean', 'median', 'min',
'max'])
print("\nStatistics per Product:")
print(statistics_per_product)
```

Output:

```
Total Revenue: 540

Total Revenue per Day:
Date
Widget    540
Name: Revenue, dtype: int64

Total Quantity per Product:
Product
A    30
B    16
Name: Quantity, dtype: int64

Average Daily Revenue: 540.0

Maximum Revenue from a Single Sale: 120

Statistics per Product:
          mean  median  min  max
Product
A        100.0   100.0   80  120
B         80.0    75.0   60  105
```

Result: The program was executed and the result was successfully obtained. Thus CO4 is obtained.

Date: 29-08-2024

Experiment No. 18

Aim: Examine the various evaluation metrics for finding the performance of an algorithm

Classification Metrics, Accuracy, Precision, Recall (Sensitivity), F1 Score, ROC Curve and AUC, Confusion Matrix, Regression Metrics, Mean Absolute Error (MAE), Mean Squared Error (MSE)

CO5: Apply Basic Machine Learning Algorithms

Procedure:

1. Classification Matrix

The classification matrix, often called the confusion matrix, is a table used to summarize theperformance of a classification algorithm. It displays the counts of true positive (TP), true negative(TN), false positive (FP), and false negative (FN) predictions. These four components are used to calculate various performance metrics.

- True Positives (TP): Correctly predicted positive instances.
- True Negatives (TN): Correctly predicted negative instances.
- False Positives (FP): Negative instances incorrectly classified as positive.
- False Negatives (FN): Positive instances incorrectly classified as negative.

The classification matrix is foundational to understanding the accuracy and errors of a classifier.

2. Accuracy

Accuracy is the most straightforward metric, representing the proportion of correct predictions (both positive and negative) out of the total number of predictions. It is defined as:

Accuracy=TP+TN / TP+TN+FP+FN

While it provides a quick overall assessment, it can be misleading if the dataset is imbalanced, i.e., when the classes are unevenly distributed.

3. Precision

Precision is a metric that focuses on the proportion of true positive predictions among all instances predicted as positive. It answers the question: "Of all instances predicted as positive, how many are actually positive?" Precision is especially important when the cost of a false positive is high.

Precision=TP / TP+FP

High precision means that when the model predicts a positive, it is likely to be correct.

4. Recall (Sensitivity, True Positive Rate)

Recall (or Sensitivity, or the True Positive Rate) measures the proportion of actual positives that were correctly identified. It answers the question: "Of all the actual positives, how many were correctly identified by the model. Recall is crucial when the cost of a false negative is high.

Recall=TP / TP+FN

High recall means that most of the true positives are correctly identified, though it may result in more false positives.

5. F1 Score

The F1 Score is the harmonic mean of precision and recall. It provides a single metric that balances both concerns and is useful when the data is imbalanced. The F1 score is particularly helpful when you need a balance between precision and recall and there is a class imbalance.

F1 Score =2 × (Precision×Recall) / (Precision+Recall)

An F1 score closer to 1 indicates better performance, whereas a score closer to 0 indicates poor performance.

6. ROC Curve (Receiver Operating Characteristic Curve)

The ROC Curve is a graphical representation of a classifier's performance across all possible thresholds. It plots the True Positive Rate (Recall) against the False Positive Rate (FPR) for every possible decision threshold. The curve helps assess the trade-off between sensitivity and specificity for different threshold values.

- True Positive Rate (TPR) is Recall.
- False Positive Rate (FPR) is FP / FP+TN

ROC curve closer to the top-left corner indicates better performance, where both the True Positive Rate is high, and the False Positive Rate is low.

7. AUC (Area Under the ROC Curve)

The AUC is the area under the ROC curve, which provides a single scalar value to evaluate the performance. It quantifies the overall ability of the model to discriminate between positive and negative classes.

- AUC = 0.5 indicates a model with no discriminative ability (random guessing).
- AUC = 1 indicates perfect classification.

Higher AUC values imply better model performance.

8. Confusion Matrix

A Confusion Matrix is a tabular summary that provides a detailed breakdown of how the model classifies the instances into the four categories: TP, TN, FP, and FN. These values are used to calculate accuracy, precision, recall, and other metrics. It helps to understand the types of errors the model is making (e.g., how many false positives vs. false negatives).

9. Regression Matrix

The Regression Matrix is a summary of how well the regression model predicts continuous outcomes. While not as commonly referenced as classification metrics, it includes different performance measures such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2

(coefficient of determination).

10. Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is the average of the absolute differences between the predicted values and the actual values. It gives a sense of how far off the predictions are from the true values, but does not take into account the direction of errors (overprediction or underprediction).

MAE = 1/n ∑ ∣yi−y^i∣

11. Mean Squared Error (MSE)

Mean Squared Error (MSE) is similar to MAE but it squares the errors before averaging them. This penalizes larger errors more heavily than smaller ones, which can be useful when you want to prioritize avoiding large mistakes.

MAE = 1/n ∑ (yi−y^i) 2

Like MAE, MSE also measures the average deviation between predicted and actual values, but the squaring emphasizes outliers or large errors. Lower MSE values indicate better performance.

Result: The program was executed and the result was successfully obtained. Thus CO5 is obtained.

Date: 02-09-2024

Experiment No. 19

Aim:

1. Calculate the mean, median, and mode for each numerical column.
2. Calculate the variance and standard deviation for each numerical column.
3. Create a correlation matrix and visualize it using a heatmap.
4. Calculate skewness and kurtosis for each numerical column.
5. Determine the 25th, 50th (median), and 75th percentiles (quantiles) for each numerical column.

CO4: Apply mathematical and statistical functions for practicing data science.

Procedure:

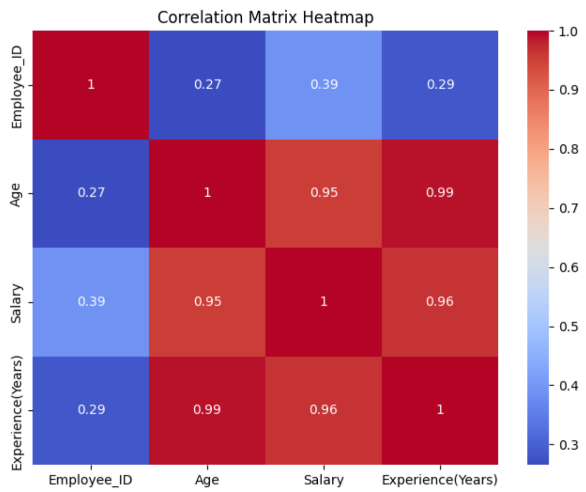
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import skew, kurtosis
df = pd.read_csv('employee_data.csv')
mean_values = df.mean()
median_values = df.median()
mode_values = df.mode().iloc[0]
print("Mean Values:")
print(mean_values)
print("\nMedian Values:")
print(median_values)
print("\nMode Values:")
print(mode_values)
variance_values = df.var()
std_dev_values = df.std()
print("\nVariance Values:")
print(variance_values)
print("\nStandard Deviation Values:")
print(std_dev_values)
correlation_matrix = df.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Heatmap')
plt.show()
skewness_values = df.skew()
kurtosis_values = df.kurt()
print("\nSkewness Values:")
print(skewness_values)
print("\nKurtosis Values:")
```

```
print(kurtosis_values)
quantiles = df.quantile([0.25, 0.5, 0.75])
print("\nQuantiles:")
print(quantiles)
```

Output:

```
Mean Values:
Employee_ID             5.5
Age                    33.9
Salary              67600.0
Experience(Years)      10.0
dtype: float64

Median Values:
Employee_ID             5.5
Age                    32.0
Salary              66000.0
Experience(Years)       7.5
dtype: float64

Mode Values:
Employee_ID             1
Age                    23
Salary              48000
Experience(Years)       1
Name: 0, dtype: int64

Variance Values:
Employee_ID          9.166667e+00
Age                  7.698889e+01
Salary               2.189333e+08
Experience(Years)    7.200000e+01
dtype: float64

Standard Deviation Values:
Employee_ID              3.027650
Age                      8.774331
Salary               14796.395958
Experience(Years)        8.485281
dtype: float64
```

```
Skewness Values:
Employee_ID          0.000000
Age                  0.696346
Salary               0.156797
Experience(Years)    0.675189
dtype: float64

Kurtosis Values:
Employee_ID         -1.200000
Age                 -0.430047
Salary              -1.446022
Experience(Years)   -0.983126
dtype: float64

Quantiles:
      Employee_ID    Age   Salary  Experience(Years)
0.25         3.25  28.25  57500.0               3.25
0.50         5.50  32.00  66000.0               7.50
0.75         7.75  38.75  78750.0              16.50
```



Result: The program was executed and the result was successfully obtained. Thus CO4 is obtained.

Date: 02-09-2024

Experiment No. 20

Aim: Covariance and Pairplot Visualization(use the same previous data)

1. Calculate the covariance matrix for the numerical columns in the dataset.
2. Interpret the covariance values in terms of the relationship between variables.
3. Visualize pairwise relationships between the variables using a pairplot.
4. Covariance measures the degree to which two variables change together. Positive covariance indicates a direct relationship, while negative indicates an inverse relationship.
5. Pairplot is a powerful visualization tool to understand the relationships between multiple pairs of variables simultaneously.

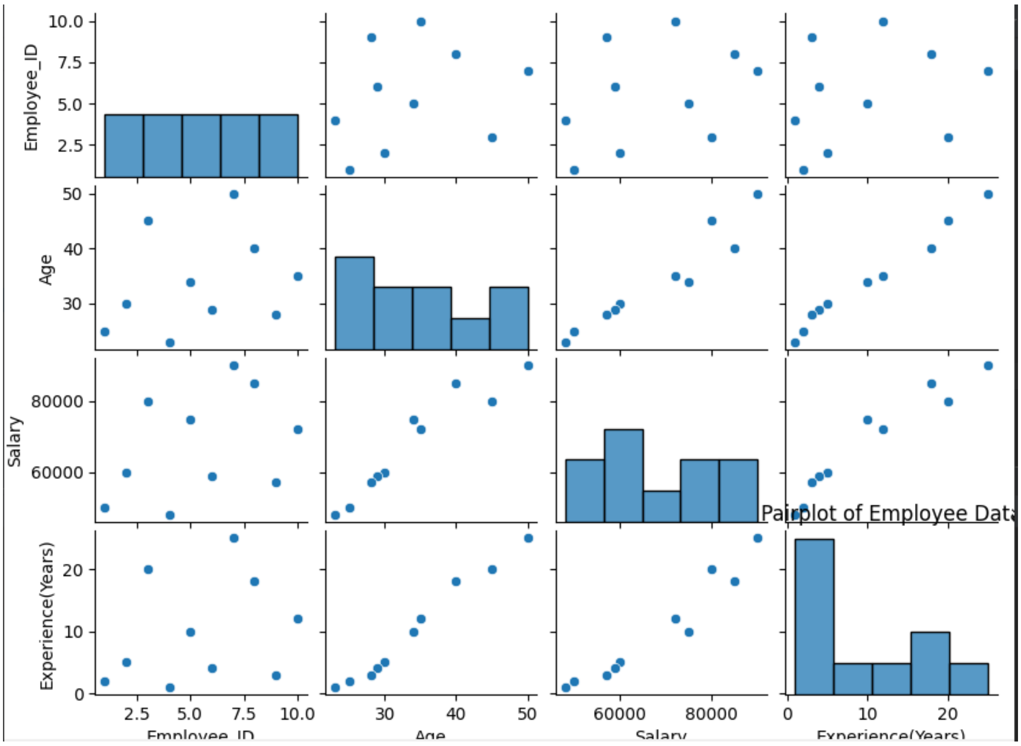
CO4: Apply mathematical and statistical functions for practicing data science.

Procedure:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('employee_data.csv')
covariance_matrix = df.cov()
print("Covariance Matrix:")
print(covariance_matrix)
sns.pairplot(df)
plt.title('Pairplot of Employee Data')
print("\nCovariance Interpretation:")
print("1. Covariance between 'Age' and 'Salary':", covariance_matrix.loc['Age', 'Salary'])
print("   Interpretation: Positive covariance indicates that as employees age, their salaries tend to increase.")
print("2. Covariance between 'Age' and 'Experience':", covariance_matrix.loc['Age', 'Experience(Years)'])
print("   Interpretation: Positive covariance suggests that as employees get older, their experience also tends to increase.")
print("3. Covariance between 'Salary' and 'Experience':", covariance_matrix.loc['Salary', 'Experience(Years)'])
print("   Interpretation: Positive covariance indicates that employees with more experience tend to have higher salaries.")
plt.show()
```

Output:

```
Covariance Matrix:
                   Employee_ID            Age        Salary  Experience(Years)
Employee_ID           9.166667       7.055556  1.733333e+04           7.333333
Age                   7.055556      76.988889  1.238444e+05          73.666667
Salary            17333.333333  123844.444444  2.189333e+08      121000.000000
Experience(Years)     7.333333      73.666667  1.210000e+05          72.000000

Covariance Interpretation:
1. Covariance between 'Age' and 'Salary': 123844.44444444444
   Interpretation: Positive covariance indicates that as employees age, their salaries tend to increase.
2. Covariance between 'Age' and 'Experience': 73.66666666666666
   Interpretation: Positive covariance suggests that as employees get older, their experience also tends to increase.
3. Covariance between 'Salary' and 'Experience': 121000.0
   Interpretation: Positive covariance indicates that employees with more experience tend to have higher salaries.
```



Result: The program was executed and the result was successfully obtained. Thus CO4 is obtained.

Date: 02-09-2024

Experiment No. 21

Aim: 1. Calculate the mean, median, and mode for each numerical column (Sales_Amount,Quantity_Sold, Discount (%)).
2. Calculate the variance and standard deviation for each numerical column.
3. Create a correlation matrix for the numerical columns and visualize it using a heatmap.
4. Calculate skewness and kurtosis for each numerical column.
5. Determine the 25th, 50th (median), and 75th percentiles (quantiles) for each numerical column.
6. Calculate the covariance matrix for the numerical columns in the dataset.
7. Interpret the covariance values in terms of the relationship between variables.
8. Visualize pairwise relationships between the variables using a pairplot.

CO4: Apply mathematical and statistical functions for practicing data science.

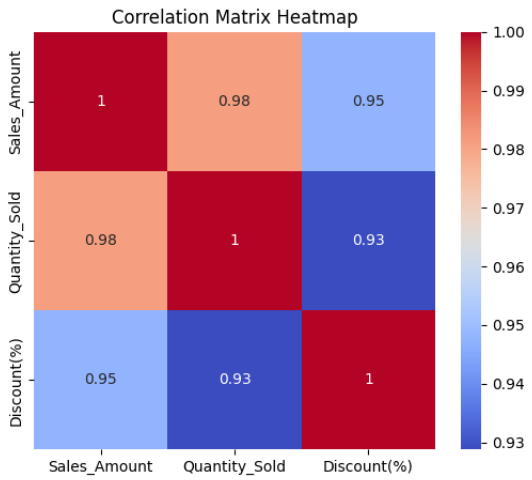
Procedure:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import skew, kurtosis
df = pd.read_csv('product_data.csv')
df['Sales_Amount'] = pd.to_numeric(df['Sales_Amount'], errors='coerce')
df['Quantity_Sold'] = pd.to_numeric(df['Quantity_Sold'], errors='coerce')
df['Discount(%)'] = pd.to_numeric(df['Discount(%)'], errors='coerce')
mean_values = df[['Sales_Amount', 'Quantity_Sold', 'Discount(%)']].mean()
median_values = df[['Sales_Amount', 'Quantity_Sold', 'Discount(%)']].median()
mode_values = df[['Sales_Amount', 'Quantity_Sold', 'Discount(%)']].mode().iloc[0]
print("Mean Values:")
print(mean_values)
print("\nMedian Values:")
print(median_values)
print("\nMode Values:")
print(mode_values)
variance_values = df[['Sales_Amount', 'Quantity_Sold', 'Discount(%)']].var()
std_dev_values = df[['Sales_Amount', 'Quantity_Sold', 'Discount(%)']].std()
print("\nVariance Values:")
print(variance_values)
print("\nStandard Deviation Values:")
print(std_dev_values)
correlation_matrix = df[['Sales_Amount', 'Quantity_Sold', 'Discount(%)']].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
```

```
plt.title('Correlation Matrix Heatmap')
plt.show()
skewness_values = df[['Sales_Amount', 'Quantity_Sold', 'Discount(%)']].skew()
kurtosis_values = df[['Sales_Amount', 'Quantity_Sold', 'Discount(%)']].kurt()
print("\nSkewness Values:")
print(skewness_values)
print("\nKurtosis Values:")
print(kurtosis_values)
quantiles = df[['Sales_Amount', 'Quantity_Sold', 'Discount(%)']].quantile([0.25, 0.5,
0.75])
print("\nQuantiles:")
print(quantiles)
covariance_matrix = df[['Sales_Amount', 'Quantity_Sold', 'Discount(%)']].cov()
print("\nCovariance Matrix:")
print(covariance_matrix)
print("\nCovariance Interpretation:")
print("1. Covariance between 'Sales_Amount' and 'Quantity_Sold':",
covariance_matrix.loc['Sales_Amount', 'Quantity_Sold'])
print("   Interpretation: Positive covariance indicates that as sales amount increases,
quantity sold tends to increase.")
print("2. Covariance between 'Sales_Amount' and 'Discount(%)':",
covariance_matrix.loc['Sales_Amount', 'Discount(%)'])
print("   Interpretation: Positive covariance suggests that as sales amount increases, the
discount percentage also tends to increase.")
print("3. Covariance between 'Quantity_Sold' and 'Discount(%)':",
covariance_matrix.loc['Quantity_Sold', 'Discount(%)'])
print("   Interpretation: Positive covariance indicates that as quantity sold increases, the
discount percentage tends to increase.")
sns.pairplot(df[['Sales_Amount', 'Quantity_Sold', 'Discount(%)']])
plt.suptitle('Pairplot of Product Data', y=1.02)
plt.show()
```

Output:

```
Mean Values:
Sales_Amount     18900.0
Quantity_Sold      237.0
Discount(%)          7.6
dtype: float64

Median Values:
Sales_Amount     18750.0
Quantity_Sold      230.0
Discount(%)          7.5
dtype: float64

Mode Values:
Sales_Amount     12000
Quantity_Sold      150
Discount(%)          0
Name: 0, dtype: int64

Variance Values:
Sales_Amount     1.582222e+07
Quantity_Sold    2.978889e+03
Discount(%)      1.804444e+01
dtype: float64

Standard Deviation Values:
Sales_Amount     3977.715704
Quantity_Sold      54.579198
Discount(%)         4.247875
dtype: float64
```



Result: The program was executed and the result was successfully obtained. Thus CO4 is obtained.

Date: 28-10-2024

Experiment No. 22

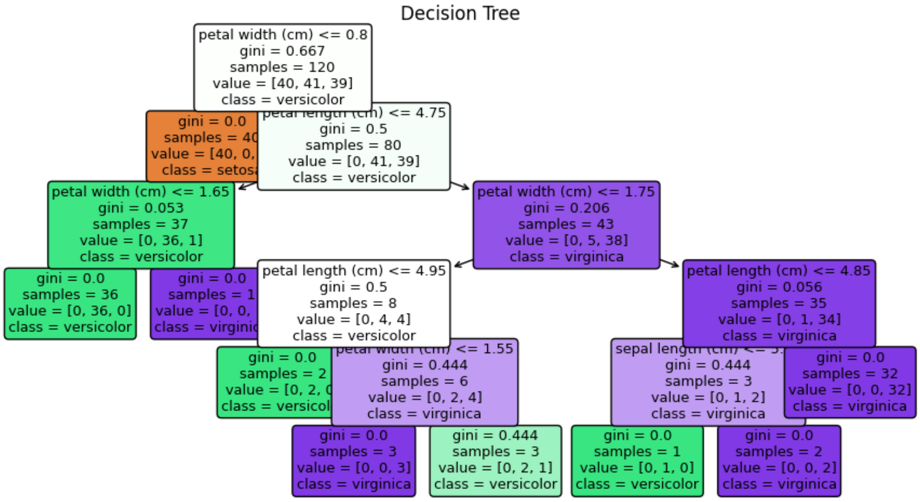
Aim: Program to implement decision tree using standard dataset iris available in scikit learn and find the accuracy of the algorithm also construct a decision tree with maximum depth=5

CO5: Apply Basic Machine Learning Algorithms

Procedure:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
iris = load_iris()
x = iris.data
y = iris.target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(max_depth=5)
clf.fit(x_train, y_train)
plt.figure(figsize=(15,20))
plot_tree(clf,filled=True,feature_names=iris.feature_names,class_names=iris.target_na
mes,rounded=True)
plt.title("Decision Tree")
plt.show()
V = clf.predict(x_test)
print(V)
result = accuracy_score(y_test, V)
print(result)
```

Output:




```
C:\Users\ajcemca\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonProject1\Exp22.py
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
1.0
```

Result: The program was executed and the result was successfully obtained. Thus CO5 is obtained.

Date: 28-10-2024

Experiment No. 23

Aim: Load the iris dataset and implement decision tree algorithm. Import the metrics accurracy_score, classification_report, confusion_matrix from sklearn.metrics and evaluate the performance of the model

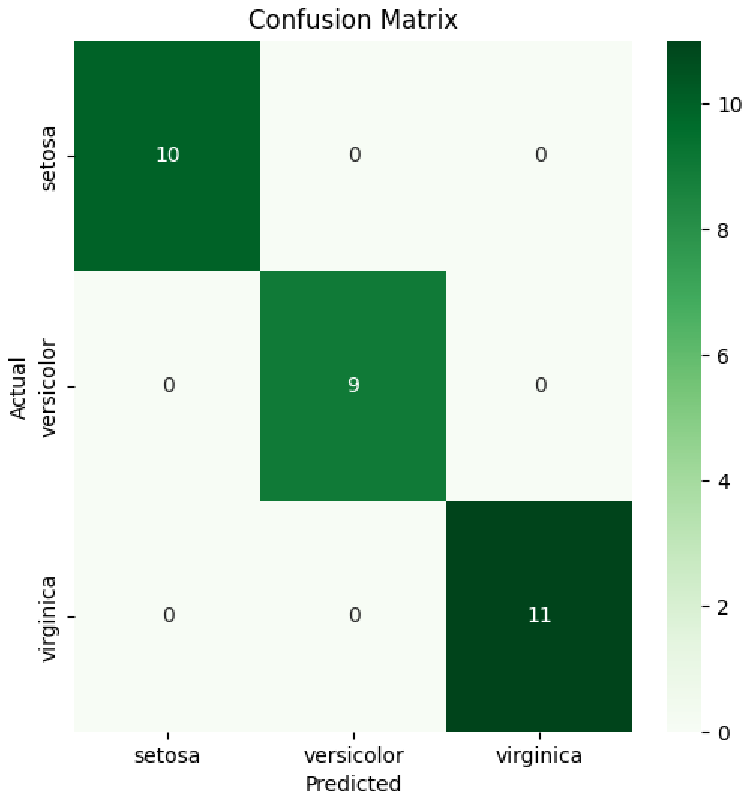
CO5: Apply Basic Machine Learning Algorithms

Procedure:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
import seaborn as sns
iris = load_iris()
x = iris.data
y = iris.target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(max_depth=10)
clf.fit(x_train, y_train)
V = clf.predict(x_test)
result = accuracy_score(y_test, V)
report=classification_report(y_test,V,target_names=iris.target_names)
print("Accurracy:",result)
print("Classification Report :",report," \n")
conf_matrix=confusion_matrix(y_test, V)
print("Confusion Matrix:\n ",conf_matrix)
plt.figure(figsize=(6,6))
# sns.heatmap(conf_matrix,annot=True,fmt="d",cmap="Blues",xticklabels=["Predicted
0","Predicted 1","Predcited 2"],yticklabels=["Actual 0","Actual 1","Actual 2"])
sns.heatmap(conf_matrix,annot=True,fmt="d",cmap="Greens",xticklabels=iris.target_n
ames,yticklabels=iris.target_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Output:

```
Accurracy: 1.0
Classification Report :               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


Confusion Matrix:
  [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```



<u>Result:</u> The program was executed and the result was successfully obtained. Thus CO5 is obtained.

Date: 28-10-2024

Experiment No. 24

Aim: Given a one dimensional dataset represented with numpy array. Write a program to calculate the slope and intercept.

CO5: Apply Basic Machine Learning Algorithms

Procedure:

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
x_values=np.array([20,30,40,50]).reshape(-1,1)
y_values=np.array([58,54,50,46])
model=LinearRegression()
model.fit(x_values,y_values)
slope=model.coef_[0]
intercept=model.intercept_
print("Linear Regression ")
print("Slope: ",slope)
print("Intercept",intercept)
print("y = ", intercept, " + ", slope,"x")
y_pred=model.predict(x_values)
mse=mean_squared_error(y_values,y_pred)
print("Mean Squared Error :",mse)
```

Output:

```
Linear Regression
Slope:  -0.4
Intercept 66.0
y =  66.0  +  -0.4 x
Mean Squared Error : 0.0
```

Result: The program was executed and the result was successfully obtained. Thus CO5 is obtained.

Date: 28-10-2024

Experiment No. 25

Aim: Program to implement multiple linear regression with dataset in a public domain(house-price.csv)

CO5: Apply Basic Machine Learning Algorithms

Procedure:

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
df=pd.read_csv('house-prices.csv')
print(df.head())
x=df[['SqFt','Bedrooms','Bathrooms']]
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)
mse=mean_squared_error(y_test,y_pred)
print("Mean Squared Error :",mse)
```

Output:

```
   Home   Price  SqFt  Bedrooms  Bathrooms  Offers Brick Neighborhood
0     1  114300  1790         2          2       2    No         East
1     2  114200  2030         4          2       3    No         East
2     3  114800  1740         3          2       1    No         East
3     4   94700  1980         3          2       3    No         East
4     5  119800  2130         3          3       3    No         East
Mean Squared Error : 320149938.2302678

Process finished with exit code 0
```

Result: The program was executed and the result was successfully obtained. Thus CO5 is obtained.