Name: Maddikunta Dakshyani

Student Id: 700666204

Code:

[Assignment4/Assignment4.ipynb at main · dxm62040ucm/Assignment4 (github.com)]

Data Manipulation

a. Read the provided CSV file 'data.csv'.

```
#Data Manipulation
#a. Read the provided CSV file 'data.csv'
import pandas as pd

# Read the CSV file
df = pd.read_csv('data.csv')
```

Reading the given CSV File data.csv. Here I used read_csv file method to read csv file content.

c. Show the basic statistical description about the data.

```
# Show basic statistical description
description = df.describe()
print("Basic Statistical Description:")
print(description)
```

Describe method on data frame will give complete information about the dataframe.

d. Check if the data has null values.

```
# Check for null values
null_values = df.isnull().sum()
print("\nNull Values:")
print(null_values)
```

Checking the Null values in the given data frame and count the number of null values.

i. Replace the null values with the mean

```
# Replace null values with the mean
df.fillna(df.mean(), inplace=True)

# Verify that null values have been replaced
null_values_after_replace = df.isnull().sum()
print("\nNull Values after Replacement:")
print(null_values_after_replace)
```

Given code gives null values with the mean of the values.

e. Select at least two columns and aggregate the data using: min, max, count, mean.

```
#e. Select at least two columns and aggregate the data using: min, max, count, m
selected_columns = ['Duration', 'Pulse']

aggregated_data = df[selected_columns].agg({
    'Duration': ['min', 'max', 'count', 'mean'],
    'Pulse': ['min', 'max', 'count', 'mean']
})

print("\nAggregated Data:")
print(aggregated_data)
```

Agg function gives the aggregate values of min max and count and mean.

f. Filter the data frame to select the rows with calories values between 500 and 1000.

On top of the data frame I performed condition of calories with grater than 500 and less than 1000.

```
#Filter the dataframe to select the rows with calories values between 500 and 10
filtered_df = df[(df['Calories'] >= 500) & (df['Calories'] <= 1000)]
print("\nFiltered DataFrame with calories values between 500 and 1000:")
print(filtered_df)
```

```
Filtered DataFrame with calories values between 500 and 1000:
     Duration  Pulse  Maxpulse  Calories
51         80    123       146     643.1
62        160    109       135     853.0
65        180     90       130     800.4
66        150    105       135     873.4
67        150    107       130     816.0
72         90    100       127     700.0
73        150     97       127     953.2
75         90     98       125     563.2
78        120    100       130     500.4
83        120    100       130     500.0
90        180    101       127     600.1
99         90     93       124     604.1
101        90     90       110     500.0
102        90     90       100     500.0
103        90     90       100     500.4
106       180     90       120     800.3
108        90     90       120     500.3
```

g. Filter the data frame to select the rows with calories values > 500 and pulse < 100.

Performing condition of values greater than 500 and pulse is less than 1000 on the given data frame.

```
[ ] #g. Filter the dataframe to select the rows with calories values > 500 and pulse
    filtered_rows = df[(df['Calories'] > 500) & (df['Pulse'] < 100)]
    print("\nRows with Calories > 500 and Pulse < 100:")
    print(filtered_rows)
```

```
Rows with Calories > 500 and Pulse < 100:
     Duration  Pulse  Maxpulse  Calories
65        180     90       130     800.4
70        150     97       129    1115.0
73        150     97       127     953.2
75         90     98       125     563.2
99         90     93       124     604.1
103        90     90       100     500.4
106       180     90       120     800.3
108        90     90       120     500.3
```

h. Create a new “df_modified” dataframe that contains all the columns from df except for “Maxpulse”.

```
#h. Create a new “df_modified” dataframe that contains all the columns from df e
#“Maxpulse”.
df_modified = df.drop(columns=['Maxpulse'])
print("\nModified DataFrame (excluding 'Maxpulse'):")
print(df_modified)
```

```
Modified DataFrame (excluding 'Maxpulse'):
     Duration  Pulse  Calories
0          60    110     409.1
1          60    117     479.0
2          60    103     340.0
3          45    109     282.4
4          45    117     406.0
..        ...    ...       ...
164        60    105     290.8
165        60    110     300.0
166        60    115     310.2
167        75    120     320.4
168        75    125     330.4

[169 rows x 3 columns]
```

i. Delete the “Maxpulse” column from the main df data frame

Given code drop Maxpulse column completely from the dataframe.

```
#i. Delete the “Maxpulse” column from the main df dataframe
df.drop(columns=['Maxpulse'], inplace=True)
print("\ndf (after removing Maxpulse column):")
print(df)
```

```
df (after removing Maxpulse column):
     Duration  Pulse  Calories
0          60    110     409.1
1          60    117     479.0
2          60    103     340.0
3          45    109     282.4
4          45    117     406.0
..        ...    ...       ...
164        60    105     290.8
165        60    110     300.0
166        60    115     310.2
167        75    120     320.4
168        75    125     330.4

[169 rows x 3 columns]
```

j. Convert the datatype of Calories column to int datatype.

```
[ ] #j. Convert the datatype of Calories column to int datatype.
    df['Calories'] = df['Calories'].astype(int)
    print("\nDataFrame after deleting 'Maxpulse' column:")
    print(df)
```

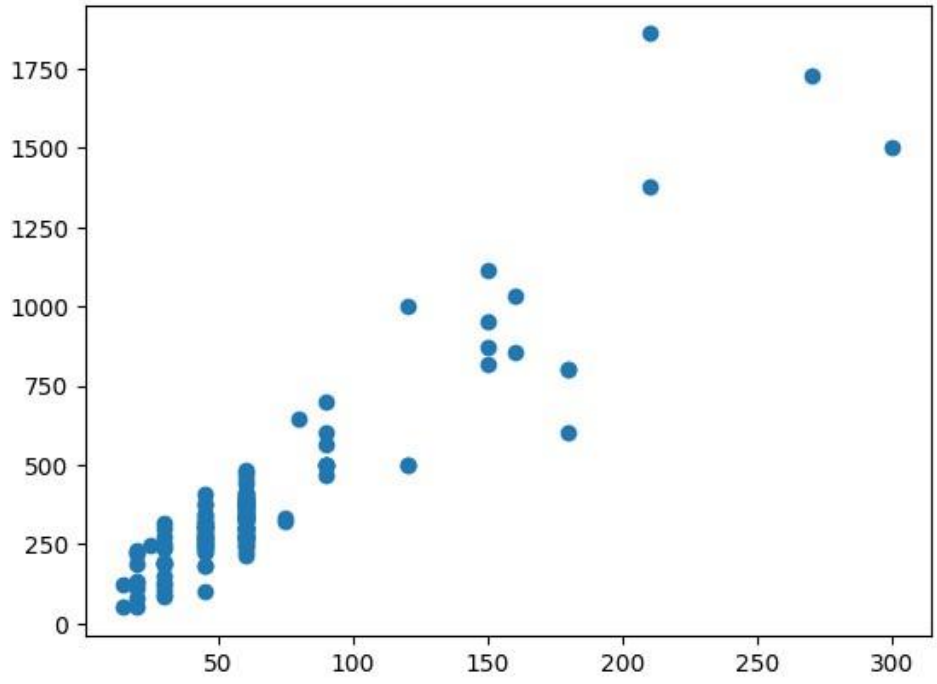
```
DataFrame after deleting 'Maxpulse' column:
     Duration  Pulse  Calories
0          60    110       409
1          60    117       479
2          60    103       340
3          45    109       282
4          45    117       406
..        ...    ...       ...
164        60    105       290
165        60    110       300
166        60    115       310
167        75    120       320
168        75    125       330

[169 rows x 3 columns]
```

k. Using pandas create a scatter plot for the two columns (Duration and Calories).

Below code explains scatter plot of columns Duration and Calories.

```
#k. Using pandas create a scatter plot for the two columns (Duration and Calorie
#Example
import matplotlib.pyplot as plt
plt.scatter(df['Duration'], df['Calories'])
```

```
<matplotlib.collections.PathCollection at 0x7a61c1296e00>
```



2) Linear Regression

a) Import the given “Salary_Data.csv”

It’s Clear that read_csv method reads the complete information of Salary_Data.csv.

After that printed the corresponding Data Frame information.

```python
#2a) Import the given "Salary_Data.csv"
import pandas as pd

# Read the CSV file
df = pd.read_csv('Salary_Data (2).csv')
print(df)
```

```
    YearsExperience    Salary
0               1.1   39343.0
1               1.3   46205.0
2               1.5   37731.0
3               2.0   43525.0
4               2.2   39891.0
5               2.9   56642.0
6               3.0   60150.0
7               3.2   54445.0
8               3.2   64445.0
9               3.7   57189.0
10              3.9   63218.0
11              4.0   55794.0
12              4.0   56957.0
13              4.1   57081.0
14              4.5   61111.0
15              4.9   67938.0
16              5.1   66029.0
17              5.3   83088.0
18              5.9   81363.0
19              6.0   93940.0
20              6.8   91738.0
21              7.1   98273.0
22              7.9  101302.0
23              8.2  113812.0
24              8.7  109431.0
25              9.0  105582.0
26              9.5  116969.0
27              9.6  112635.0
```

✓ 0s completed at 8:46 PM

b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.

Segregating the complete data frame into 2 sets Train Set and Test Set.

Size of Test Set is 33.33% and size of Train Set is 66.66%

```python
#b) Split the data in train_test partitions, such that 1/3 of the data is reserv
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(df, test_size=1/3, random_state=42)

# Display the training and testing sets
print("Training Set:")
print(train_set)
print("\nTesting Set:")
print(test_set)
```

```
Training Set:
    YearsExperience    Salary
4               2.2   39891.0
16              5.1   66029.0
5               2.9   56642.0
13              4.1   57081.0
11              4.0   55794.0
22              7.9  101302.0
1               1.3   46205.0
2               1.5   37731.0
25              9.0  105582.0
3               2.0   43525.0
21              7.1   98273.0
26              9.5  116969.0
18              5.9   81363.0
29             10.5  121872.0
20              6.8   91738.0
7               3.2   54445.0
10              3.9   63218.0
14              4.5   61111.0
19              6.0   93940.0
6               3.0   60150.0

Testing Set:
    YearsExperience    Salary
27              9.6  112635.0
15              4.9   67938.0
23              8.2  113812.0
```

c) Train and predict the model.

Applied Linear Regression on Bothe sets of data and Finally collected the Actual data and Predicted Data

```
#c) Train and predict the model
from sklearn.linear_model import LinearRegression
X_train, X_test, y_train, y_test = train_test_split(df[['YearsExperience']], df

# Train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Display the predicted values and the actual values in the test set
result_df = pd.DataFrame({'Actual Salary': y_test, 'Predicted Salary': y_pred})
print(result_df)
```

```
    Actual Salary  Predicted Salary
27       112635.0     115814.307562
15        67938.0      71511.925348
23       113812.0     102617.853286
17        83088.0      75282.340855
8         64445.0      55487.659440
9         57189.0      60200.678825
28       122391.0     122412.534701
24       109431.0     107330.872670
12        56957.0      63028.490456
0         39343.0      35692.978025
```

d) Calculate the mean_squared error

Importing mean_squared_error and passed test and predicted values to it.

```
#d) Calculate the mean_squared error
from sklearn.metrics import mean_squared_error
# Calculate the mean squared error
mse = mean_squared_error(y_test, y_pred)

# Display the results
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 35301898.887134895
```

e) Visualize both train and test data using scatter plot Drawn

the Scatter plot of Train and Test Data.

```
#e) Visualize both train and test data using scatter plot
import matplotlib.pyplot as plt

plt.scatter(X_train, y_train, label='Train Data')
plt.scatter(X_test, y_test, label='Test Data', marker='x')
```

```
<matplotlib.collections.PathCollection at 0x7a61b6e79de0>
```