Spring 2024: CS5720 Neural Networks & Deep Learning -

ICP-4 Assignment-4

Name: Baddam Sangeetha
STUDENT ID:700757191

GitHub link: https://github.com/Sangeetha-Baddam/Assignment_4

Drive link:
https://drive.google.com/file/d/1lS20HD5_FU2J6EuaIBG7iye7N7x-DN8q/view?usp=drive_link

1. Data Manipulation

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

b. https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing

c. Show the basic statistical description about the data.

Settings Help

Untitled.ipynb

Code Notebook Python (Pyodide)

[24]:
```
import pandas as pandas
import pandas as pd
import numpy as np
data_manup=pd.read_csv("data.csv")#to read csv file
data_manup.describe()#to describe basic ststical description from data
```

[24]:

	Duration	Pulse	Maxpulse	Calories
count	169.000000	169.000000	169.000000	164.000000
mean	63.846154	107.461538	134.047337	375.790244
std	42.299949	14.510259	16.450434	266.379919
min	15.000000	80.000000	100.000000	50.300000
25%	45.000000	100.000000	124.000000	250.925000
50%	60.000000	105.000000	131.000000	318.600000
75%	60.000000	111.000000	141.000000	387.600000
max	300.000000	159.000000	184.000000	1860.400000

d. Check if the data has null values.

i. Replace the null values with the mean

[2]:
```
show_null=data_manup.isnull().sum()#display null values
print(show_null)#print the null values
```

```
Duration    0
Pulse       0
Maxpulse    0
Calories    5
dtype: int64
```

[3]:
```
data_manup.fillna(data_manup.mean(),inplace=True)#replace null values with mean value
print(data_manup)#print after replacing with mean
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
```

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

```
[4]: data_manup=data_manup[["Duration","Pulse","Maxpulse","Calories"]]
aggregate={"Duration":["max","min","count","mean"],
           "Pulse":["max","min","count","mean"],
           "Maxpulse":["max","min","count","mean"],
           "Calories":["max","min","count","mean"]}# to find max,min,count,mean of all the columns
aggregate_data_manup=data_manup.agg(aggregate)#function to aggregate
print(aggregate_data_manup)#print the aggregate
```

```
          Duration       Pulse    Maxpulse     Calories
max    300.000000  159.000000  184.000000  1860.400000
min     15.000000   80.000000  100.000000    50.300000
count  169.000000  169.000000  169.000000   169.000000
mean    63.846154  107.461538  134.047337   375.790244
```

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

```
[5]: calories_in_range=(data_manup["Calories"]>=500) & (data_manup["Calories"]<=1000)#defining range of values to be displayed
filters_result=data_manup[calories_in_range]#adding the defined range to new variable
print(filters_result)#printing the new result
```

```
     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 dataframe to select the rows with calories values > 500 and pulse < 100.

```
[6]: calories_pulse_filter=(data_manup["Calories"]>500)&(data_manup["Pulse"]<100)#defining range
filters_result=data_manup[calories_pulse_filter]#adding the result to new variable
print(filters_result)#printing the result
```

```
     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”.

```
[7]: data_manup_modified=data_manup.drop(columns=["Maxpulse"])#displaying every column except Maxpulse
print(data_manup_modified)#printing the rersult
```

```
     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 dataframe

```
[8]: del data_manup["Maxpulse"]#command to delete entire row
     print(data_manup)
```

```
     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.

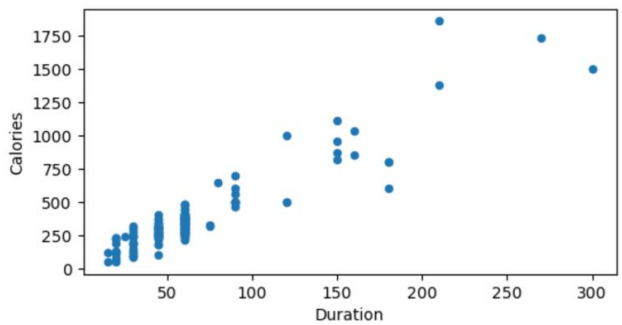
```
[9]: data_manup['Calories'] = data_manup['Calories'].fillna(0).astype(int)#converting to int data type
     print(data_manup)
```

```
     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).

```
[10]: import matplotlib.pyplot as plt
      data_manup.plot(kind='scatter', x='Duration', y='Calories', figsize=(6,3))
      plt.show()
```



2. Linear Regression

a) Import the given "Salary_Data.csv"

```
[29]: #excluding last column i.e., years of experience column
      A = Lin_Re.iloc[:, :-1].values
      #only salary column
      B = Lin_Re.iloc[:, 1].values
```

```
[30]: # (b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.
      from sklearn.model_selection import train_test_split
      A_train, A_test, B_train, B_test = train_test_split(A, B, test_size=1/3, random_state=0)
```

```
[31]: # (c) Train and predict the model.
      from sklearn.linear_model import LinearRegression
      lRegression = LinearRegression()
      lRegression.fit(A_train, B_train)
      B_Pred = lRegression.predict(A_test)
      B_Pred
```

```
[31]: array([ 40835.10590871, 123079.39940819,  65134.55626083,  63265.36777221,
             115602.64545369, 108125.8914992 , 116537.23969801,  64199.96201652,
              76349.68719258, 100649.1375447 ])
```

```
[32]: # (d) Calculate the mean_squared error
      import numpy as np
      Sum_Serror = np.sum((B_Pred - B_test) ** 2)
      mean_squared_error = Sum_Serror / B_test.size
      mean_squared_error
```

```
[32]: 21026037.329511303
```

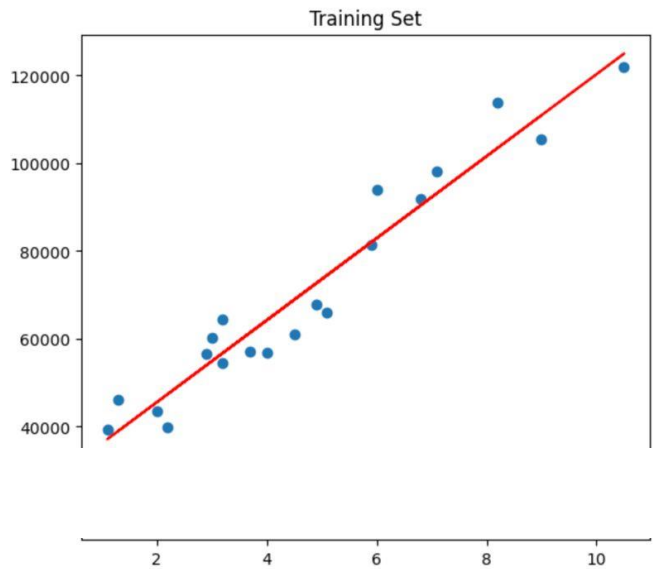
b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.

c) Train and predict the model.

d) Calculate the mean_squared error

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

[33]:
```
# (e) Visualize both train and test data using scatter plot.
import matplotlib.pyplot as plt
# Training Data set
plt.scatter(A_train, B_train)
plt.plot(A_train, lRegression.predict(A_train), color='red')
plt.title('Training Set')
plt.show()
```



[34]:
```
# Testing Data set
plt.scatter(A_test, B_test)
plt.plot(A_test, lRegression.predict(A_test), color='red')
plt.title('Testing Set')
plt.show()
```