Neural Networks & Deep Learning - ICP- 4

V V S Murthykolla
700729142

Video link:
https://drive.google.com/file/d/1P_7KVPiDhI1NNdng7jHIQGo7iYXdV0je/view?usp=sharing

Github link:

https://github.com/murthykolla/ICP_4.git

Question -1

In these steps i am importing the required libraries and reading the dataset and evaluating the dataset

```
#Importing all the required libraries
import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Reading the Data
data=pd.read_csv('data.csv')
```

```
#Viewing the shape of the data
data.shape
```

```
(169, 4)
```

Showing the basic statistical description about the data

```
#Showing the basic statistical description about the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  169 non-null    int64  
 1   Pulse     169 non-null    int64  
 2   Maxpulse  169 non-null    int64  
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
```

```
#describe() function will describe the data count,mean,std,min,25%,50%,75%,max
data.describe()
```

	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

Checking the null values in the dataset

```
#Checking for the null values in the dataset
data.isnull().any().sum()
```

```
1
```

Replacing the null values with the mean and checking for null values again

```
#Replace the null values with the mean
df = data.fillna(data.mean())
```

```
#After replacing the null value with mean the reault as zero null values
df.isnull().any().sum()
```

```
0
```

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

```
#for agregatting the data used the Duration and Pulse column
#fetching the mean of Duration and Pulse column from dataset
df[['Duration', 'Pulse']].mean()
```

```
Duration     63.846154
Pulse       107.461538
dtype: float64
```

```
#fetching the min of Duration and Pulse column from dataset

df[['Duration', 'Pulse']].min()
```

```
Duration    15
Pulse       80
dtype: int64
```

```
#fetching the max of Duration and Pulse column from dataset

df[['Duration', 'Pulse']].max()
```

```
Duration    300
Pulse       159
dtype: int64
```

```
#fetching the count of Duration and Pulse column from dataset

df[['Duration', 'Pulse']].count()
```

```
Duration    169
Pulse       169
dtype: int64
```

Filter the dataframe to select the rows with calories values between 500 and 1000.

```
#Filtering the dataframe to select the rows with calories values between 500 and 1000.
print(df[(df['Calories'] < 1000) & (df['Calories'] > 500)])
```

```
     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
90        180    101       127     600.1
99         90     93       124     604.1
103        90     90       100     500.4
106       180     90       120     800.3
108        90     90       120     500.3
```

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

```
#Filtering the dataframe to select the rows with calories values > 500 and pulse < 100.
print(df[(df['Pulse'] < 100) & (df['Calories'] > 500)])
```

```
     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
```

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

```
#Create a new "df_modified" dataframe that contains all the columns from df except for
# Maxpulse
df_modified= df.drop(['Maxpulse'], axis=1)
```



```
df_modified.shape
```

```
(169, 3)
```

```
#this step shows the modified dataframe without the maxpulse column
df_modified.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  169 non-null    int64  
 1   Pulse     169 non-null    int64  
 2   Calories  169 non-null    float64
dtypes: float64(1), int64(2)
memory usage: 4.1 KB
```

Delete the “Maxpulse” column from the main df dataframe

```
#Deleting the "Maxpulse" column from the modified df dataframe
del df["Maxpulse"]
```

```
#after deleting the maxpulse from the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  169 non-null    int64  
 1   Pulse     169 non-null    int64  
 2   Calories  169 non-null    float64
dtypes: float64(1), int64(2)
memory usage: 4.1 KB
```

Convert the datatype of Calories column to int datatype

```
#Convert the datatype of Calories column to int datatype
df = df.astype({"Calories":'int'})
```

```
#this step indicates the converting of float datatype into int datatype
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  169 non-null    int64
 1   Pulse     169 non-null    int64
 2   Calories  169 non-null    int64
dtypes: int64(3)
memory usage: 4.1 KB
```

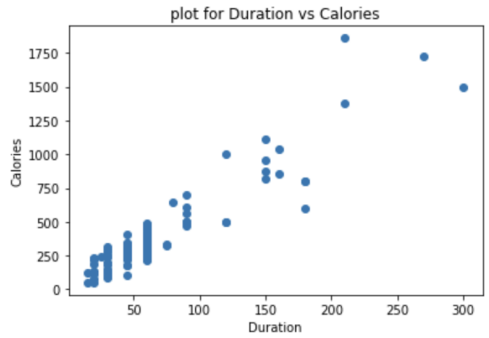
Using pandas create a scatter plot for the two columns (Duration and Calories)

```
#Creating the Scatter plot for Duration and Calories
#Scatter plot for Duration against Calories
plt.scatter(df['Duration'], data['Calories'])

#Displaying the title for the plot
plt.title("plot for Duration vs Calories")

# Setting the X and Y labels
plt.xlabel('Duration')
plt.ylabel('Calories')

plt.show()
```



Question - 2

Importing the all the required libraries

```
#Importing all the required libraries
import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import mean_squared_error

from sklearn.linear_model import LinearRegression
```

Reading the salary.csv file

```
#Reading the Data
data=pd.read_csv('Salary_Data.csv')
```

Printing the basic statistical description about the data

```
#Showing the basic statistical description about the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

Split the dataset

```
# Splitting the dataset into the Training set and Test set
X_Trainset, X_Testset, Y_Trainset, Y_Testset = train_test_split(X, Y, test_size=0.3, random_state=0)
```

Training the model and predicting the data ,calculating the mean square error

```
# Fitting Simple Linear Regression to the training set
lineaRregressor = LinearRegression()
lineaRregressor.fit(X_Trainset, Y_Trainset)

# predict the Test set result
Y_Predset = lineaRregressor.predict(X_Testset)

# calculating mean square error
meansquareerror = mean_squared_error(Y_Testset,Y_Predset)
print(f"\nMean Square Error = {meansquareerror}")
```

```
Mean Square Error = 23370078.800832972
```

Visualizing the training set results

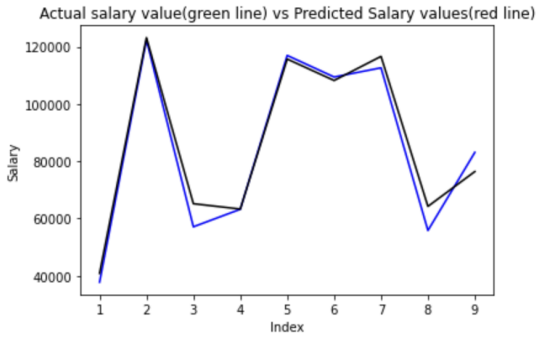
```
# Visualising the Training set results
plt.scatter(X_Trainset, Y_Trainset,  color='blue')
plt.title('Training Dataset (Salary vs Years Of Experience)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Visualizing the Testing set results

```
# Visualising the Testing set results
plt.scatter(X_Testset, Y_Testset,  color='blue')
plt.title('Test Dataset (Salary vs Years Of Experience)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

plotting the actual and predicted values

```
# plotting the actual and predicted values
compare = [i for i in range(1, len(Y_Testset)+1, 1)]
plt.plot(compare, Y_Testset, color='blue', linestyle='-')
plt.plot(compare, Y_Predset, color='black', linestyle='-')
plt.xlabel('Index')
plt.ylabel('Salary')
plt.title('Actual salary value(blue line) vs Predicted Salary values(black line)')
plt.show()
```



Plotting the test data and predicted data

```
# Plotting  the test data and predicted data
plt.scatter(X_Testset, Y_Testset,  color='blue')
plt.plot(X_Testset, Y_Predset, color='black', linewidth=3)
plt.title('Salary vs Years Of Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```