Linear Regression Model

Step 1:

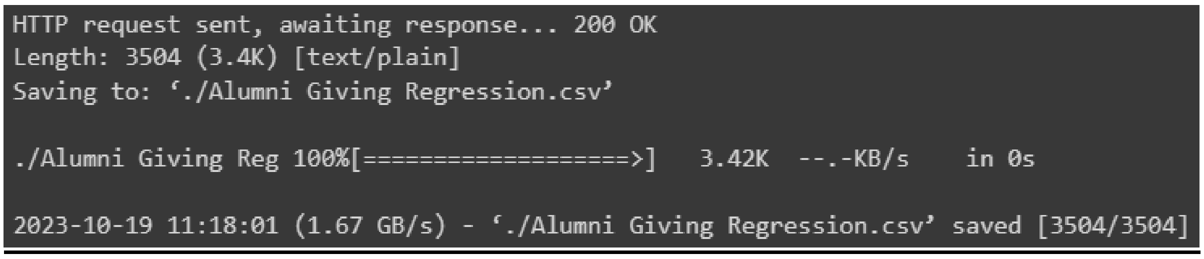
First, we imported the required libraries into our Jupyter environment. Then we created a linear regression model for our specific application. We used *Linear Regression* because we want to predict a discrete value rather than just a binary 1/0, True/False or Yes/No which is the case in *Logistic Regression*. We could have used other possible Regression models such as Ridge Regression (prevents overfitting and implements L2 regulation), Lasso Regression (for feature selection and implements L1 regulation) and Passive Aggressive Regression(for online learning and adapt to changing data) etc.

We downloaded the specified dataset using the ***!wget c***ommand that mitigates the need of manually downloading the CSV file from the internet.

The code for the steps mentioned above is given below:

```
###{Importing Libraries}###
#Using two libraries, Keras(used for model training, based on Tensorflow) and SKLearn
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn import linear_model
from sklearn import preprocessing
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
import pandas as pd
import csv
###{Downloading Dataset}###
np.random.seed(21)
#Manually downloaded but can be retrieved using the following line of code.
!wget
https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni%20Giving%20Regression%20%28
Edited%29.csv?dl=0 -O "./Alumni Giving Regression.csv"
```

Output:



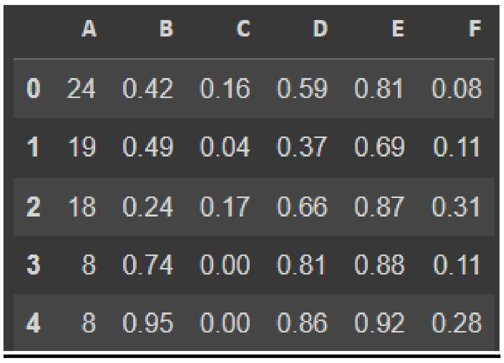
```
HTTP request sent, awaiting response... 200 OK
Length: 3504 (3.4K) [text/plain]
Saving to: ‘./Alumni Giving Regression.csv’

./Alumni Giving Reg 100%[==================>]   3.42K  --.-KB/s    in 0s

2023-10-19 11:18:01 (1.67 GB/s) - ‘./Alumni Giving Regression.csv’ saved [3504/3504]
```

Step 2:

Next, we explore the dataset and its columns using the head command which displays the top 5 entries of the dataset but we can enter a different number as a parameter to display that number of the top entries from the dataset.

```
df = pd.read_csv("/content/Alumni Giving Regression.csv", delimiter = ",")
###{Displaying the Head of the Dataset}###
df.head()
```

Output:



	A	B	C	D	E	F
0	24	0.42	0.16	0.59	0.81	0.08
1	19	0.49	0.04	0.37	0.69	0.11
2	18	0.24	0.17	0.66	0.87	0.31
3	8	0.74	0.00	0.81	0.88	0.11
4	8	0.95	0.00	0.86	0.92	0.28

Step 3:

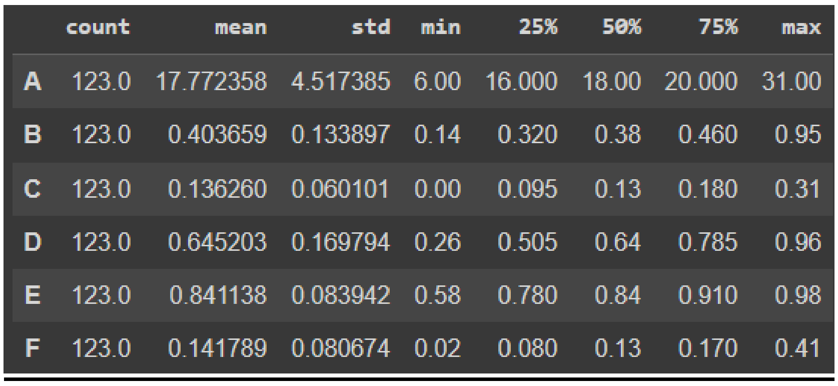
In this step, we describe the datasets information and overview which includes the following parts:

- .Count
- Mean
- Standard Deviation
- Minimum Value
- 25 Percentile
- 50 Percentile
- 75 Percentile
- Maximum Value

The code to do this is as follows:

```
###{Describing the Dataset}###
df.describe().T
```

Output:



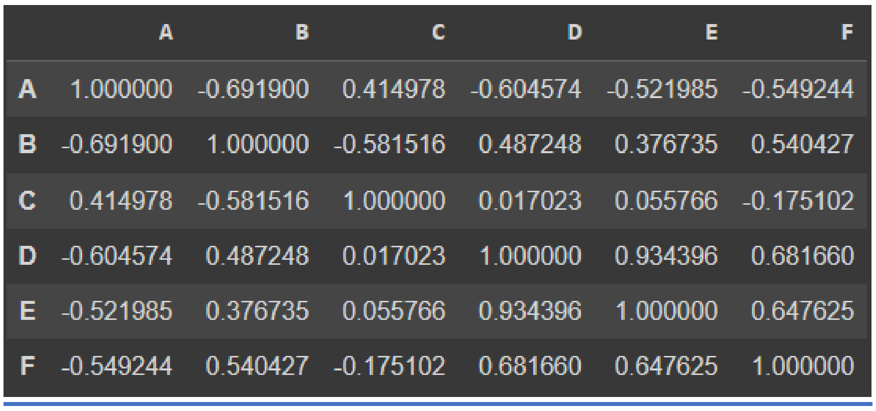
	count	mean	std	min	25%	50%	75%	max
A	123.0	17.772358	4.517385	6.00	16.000	18.00	20.000	31.00
B	123.0	0.403659	0.133897	0.14	0.320	0.38	0.460	0.95
C	123.0	0.136260	0.060101	0.00	0.095	0.13	0.180	0.31
D	123.0	0.645203	0.169794	0.26	0.505	0.64	0.785	0.96
E	123.0	0.841138	0.083942	0.58	0.780	0.84	0.910	0.98
F	123.0	0.141789	0.080674	0.02	0.080	0.13	0.170	0.41

Step 4:

In this step, we create the correlation matrix using the following code.

```
###{Correlation Calculation}###
corr=df.corr(method ='pearson')
corr
```

Output:



```
          A          B          C          D          E          F
A  1.000000  -0.691900   0.414978  -0.604574  -0.521985  -0.549244
B -0.691900   1.000000  -0.581516   0.487248   0.376735   0.540427
C  0.414978  -0.581516   1.000000   0.017023   0.055766  -0.175102
D -0.604574   0.487248   0.017023   1.000000   0.934396   0.681660
E -0.521985   0.376735   0.055766   0.934396   1.000000   0.647625
F -0.549244   0.540427  -0.175102   0.681660   0.647625   1.000000
```

Step 5:

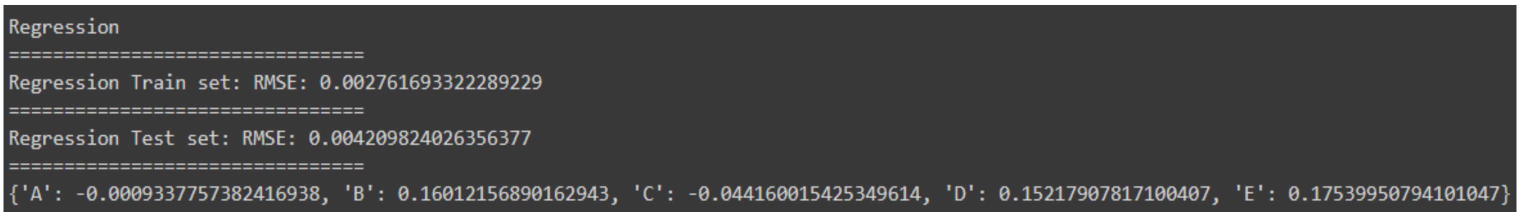
In this step, we built/train the Linear Regression Model and the also test it. We also display it's root mean square error.

The code is as below:

```
###{Linear Regression Model Training and Testing}###
Y_POSITION = 5
model_1_features = [i for i in range(0,Y_POSITION)]
X = df.iloc[:,model_1_features]
Y = df.iloc[:,Y_POSITION]
#creating the model
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.20,
random_state=2020)
model1 = linear_model.LinearRegression()
model1.fit(X_train, y_train)
y_pred_train1 = model1.predict(X_train)
print("Regression")
print("================================")
RMSE_train1 = mean_squared_error(y_train,y_pred_train1)
print("Regression Train set: RMSE: {}".format(RMSE_train1))
print("================================")
y_pred1 = model1.predict(X_test)
RMSE_test1 = mean_squared_error(y_test,y_pred1)
print("Regression Test set: RMSE: {}".format(RMSE_test1))
print("================================")
```

```
coef_dict = {}
for coef, feat in zip(model1.coef_,model_1_features):
    coef_dict[df.columns[feat]] = coef
print(coef_dict)
```

Output:



```
Regression
=================================
Regression Train set: RMSE: 0.002761693322289229
=================================
Regression Test set: RMSE: 0.004209824026356377
=================================
{'A': -0.0009337757382416938, 'B': 0.16012156890162943, 'C': -0.044160015425349614, 'D': 0.15217907817100407, 'E': 0.17539950794101047}
```