**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%28Edited%29.csv?dl=0 -O "./Alumni Giving Regression.csv" |

**Output:**

**A screen shot of a computer

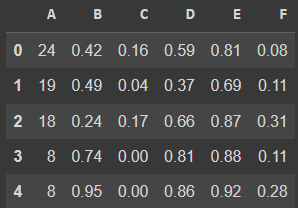
Description automatically generated**

**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:**

****

**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:**

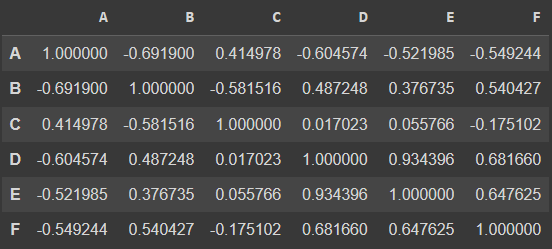
****

**Step 4:**

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

|  |
| --- |
| **###{Correlation Calculation}###**  corr=df.corr(method ='pearson')  corr |

**Output:**

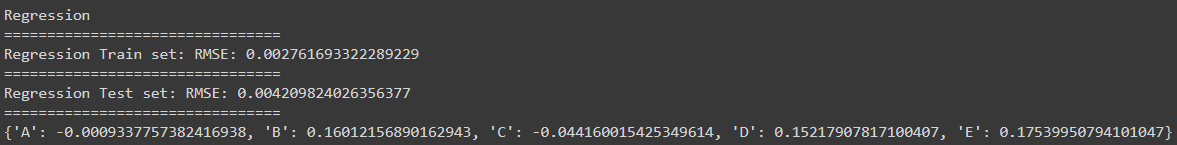
****

**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:**