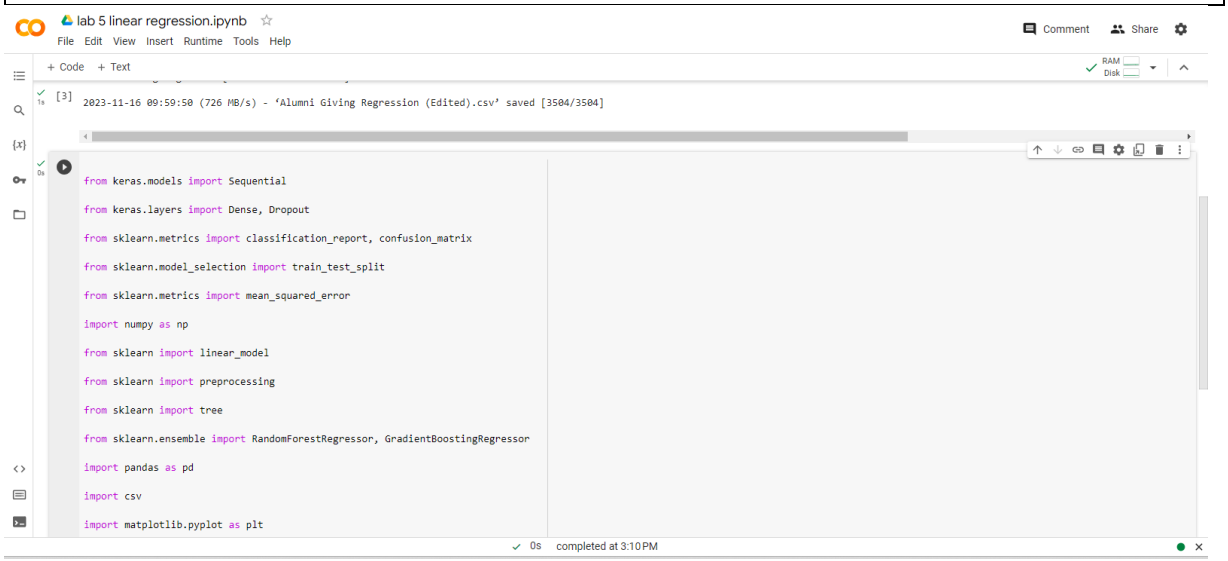
IN LAB

LAB TASK 1 :

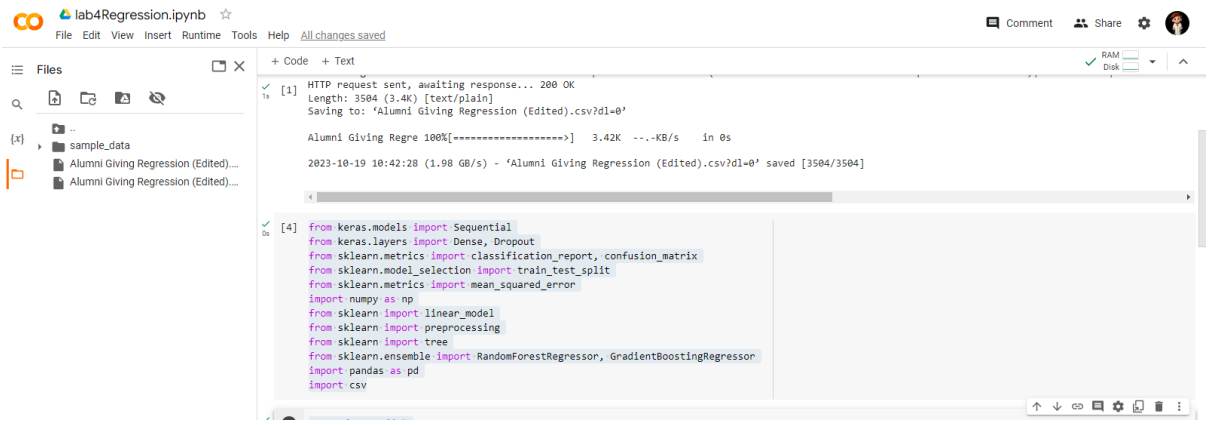
```
from keras.models import Sequential

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

import matplotlib.pyplot as plt
```



LAB TASK 2:

Importing important file into the google collab using python commands.

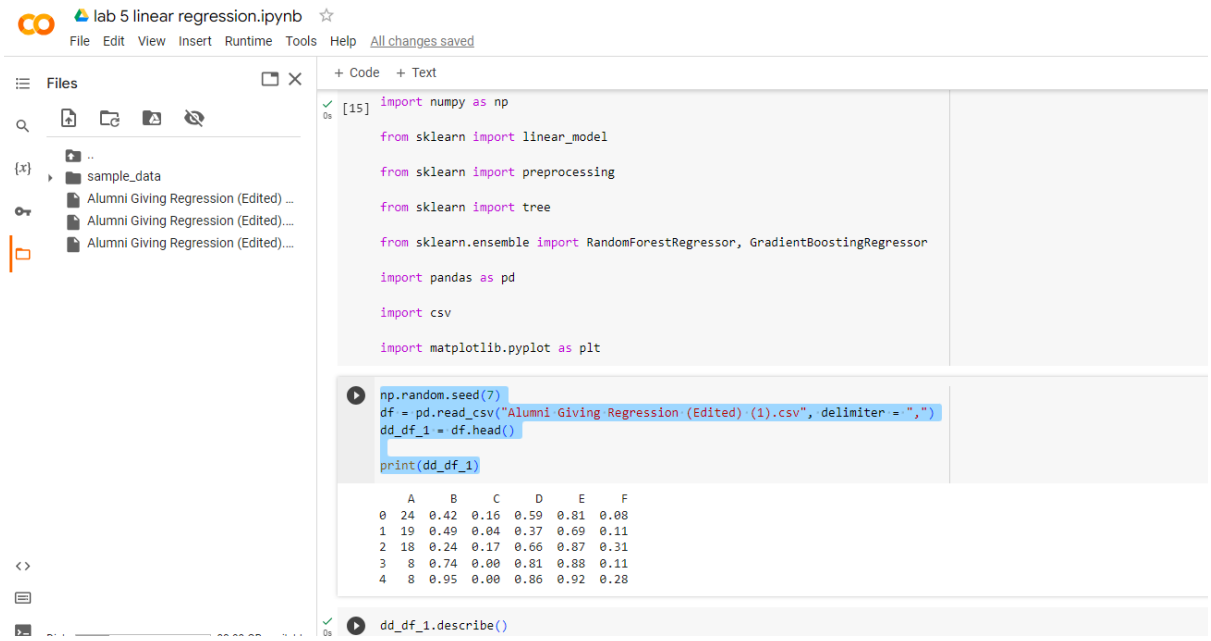
```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn import linear_model
from sklearn import preprocessing
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
import pandas as pd
import csv
```



LAB TASK 3:

Displaying some rows and columns of data which are at the starting of the file.

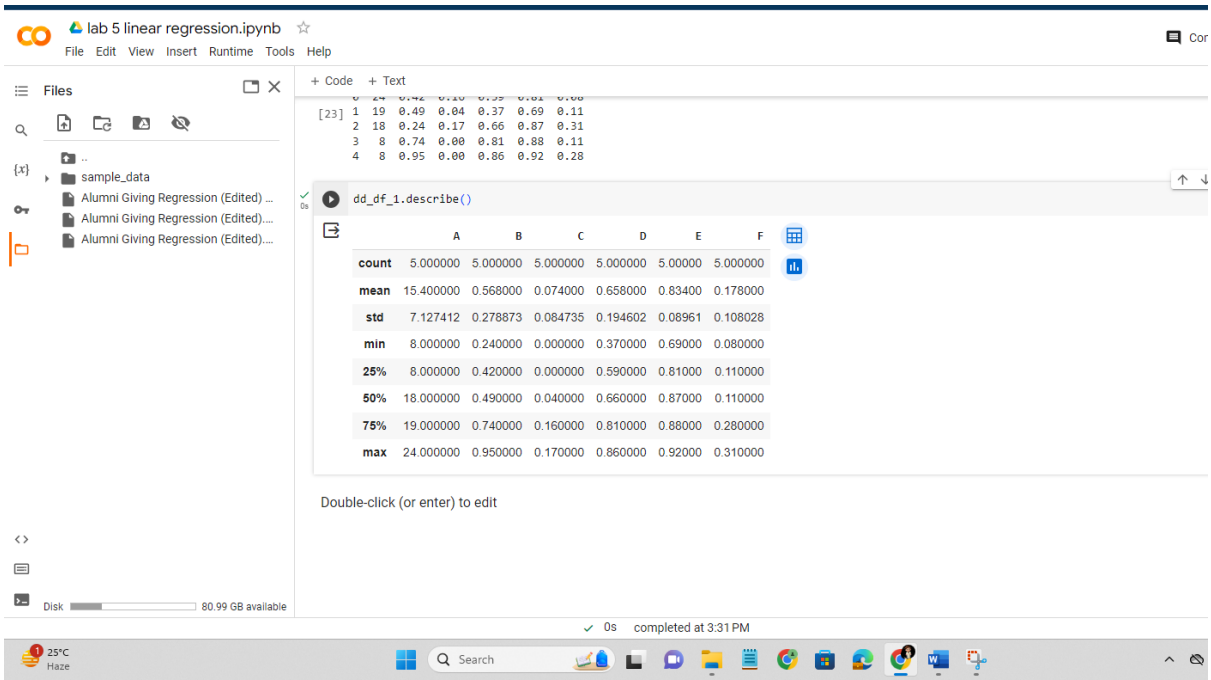
```
np.random.seed(7)
df = pd.read_csv("Alumni Giving Regression (Edited) (1).csv",
delimiter = ",")
dd_df_1 = df.head()

print(dd_df_1)
```



LAB TASK 4:

This command will give us the whole analysis of the file with some parameters like count, mean, minimum, maximum etc

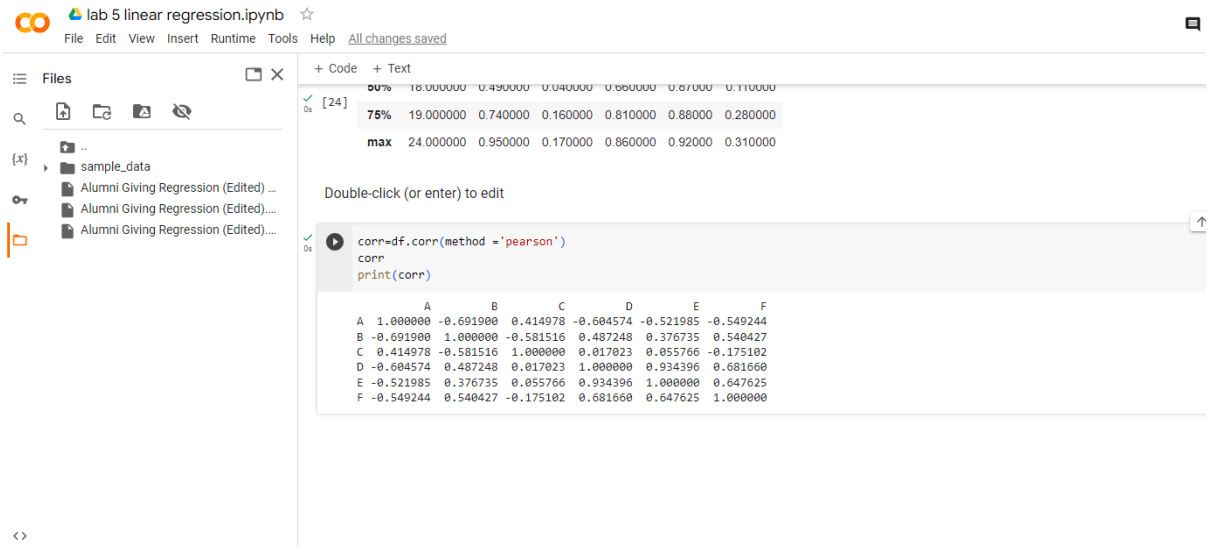
```
dd_df_1.describe()
```



LAB TASK 5:

Here we calculate the Pearson correlation coefficients between the columns (variables) in a DataFrame and store the resulting correlation matrix in the variable corr.

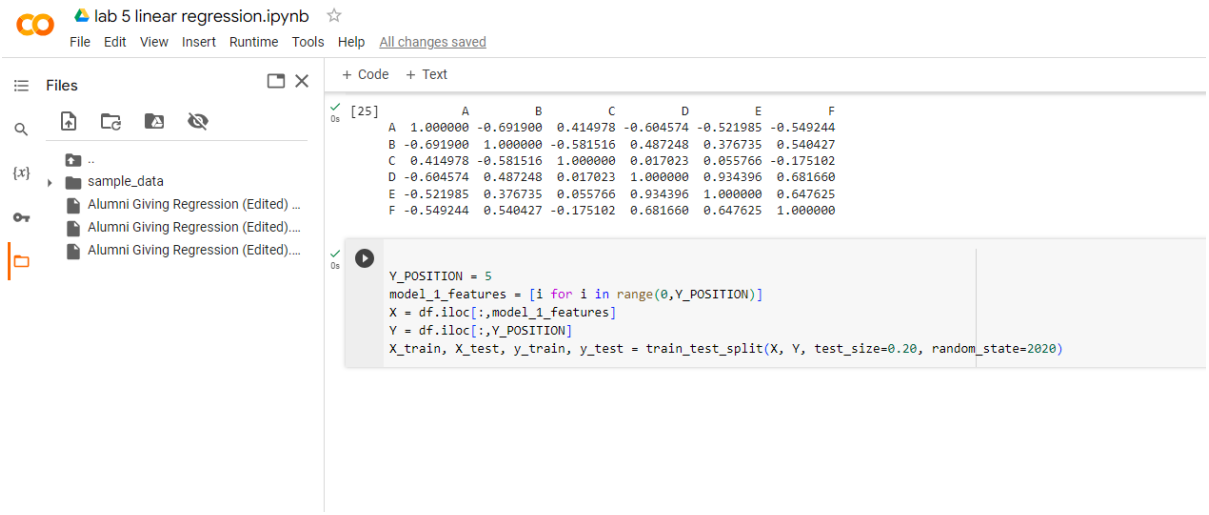
```
corr=df.corr(method ='pearson')
corr
print(corr)
```



LAB TASK 6:

These lines of code prepare data for regression modeling by selecting the independent features (predictors) and the target variable

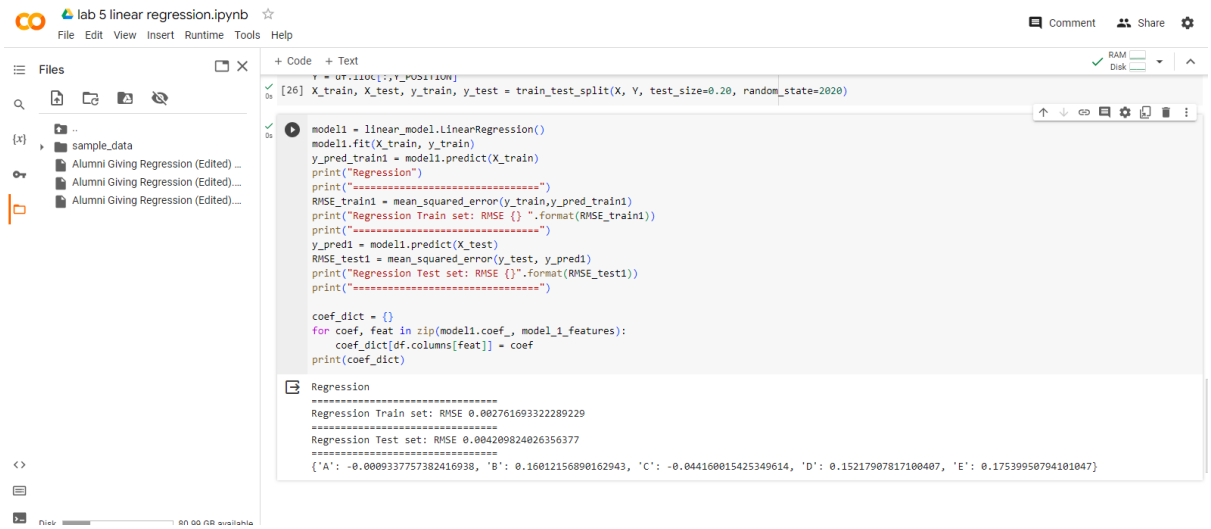
```
Y_POSITION = 5
model_1_features = [i for i in range(0,Y_POSITION)]
X = df.iloc[:,model_1_features]
Y = df.iloc[:,Y_POSITION]
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.20, random_state=2020)
```



LAB TASK 6:

This is the main model. In this code, a linear regression model is trained using the training data and evaluated on both the training and testing datasets. It calculates and reports the Root Mean Squared Error (RMSE), a measure of the model's predictive performance. Additionally, it extracts and displays the coefficients of the features, indicating their influence on the target variable in the linear regression model.

```
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)
```



Lab task 7

```
x_values = np.arange(len(y_test))

plt.scatter(x_values, y_test, color='red', label='Actual')

#Scatter plot of predicted values (y_pred) in green

plt.scatter(x_values, y_pred1, color='green', label='Predicted')

plt.xlabel('Index or Sequence of Values')

plt.ylabel('Values')

plt.title("Actual vs Predicted Values")

plt.legend()

plt.show()
```


lab 5 linear regression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
sample_data
Alumni Giving Regression (Edited) ...
Alumni Giving Regression (Edited)....
Alumni Giving Regression (Edited)....
Disk 80.99 GB available
+ Code + Text
#Scatter plot of predicted values (y_pred) in green
plt.scatter(x_values, y_pred1, color='green', label='Predicted')
plt.xlabel('Index or Sequence of Values')
plt.ylabel('Values')
plt.title("Actual vs Predicted Values")
plt.legend()
plt.show()
Actual vs Predicted Values
Actual
Predicted
Values
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
5
10
15
20
25
1s completed at 3:50 PM