```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sbr
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
```

Q1.

Define a function that will return the Mean Square, Mean absolute error and root mean Squared error.

```
Actual_Val = [2,6,45,7,3,89,3,7,8,23,56,87]
Predicted_Val = [5,8,34,9,4,67,3,8,52,34,40,78]

Actual_Val = np.array(Actual_Val)
Predicted_Val = np.array(Actual_Val)

X = Actual_Val[:,np.newaxis]
y = Predicted_Val[:,np.newaxis]



lr.fit(X,y)

y_pred = lr.predict(X)
```

```
print('Mean Squared Error:', metrics.mean_squared_error(y, y_pred))

print('Mean Absolute Error:', metrics.mean_absolute_error(y, y_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y, y_pred)))
```

```
Mean Squared Error: 2.4865553116653975e-29
Mean Absolute Error: 4.255854927729767e-15
Root Mean Squared Error: 4.986537186931827e-15
```

Find out the value of c and m after 500 epochs which one is better to user m=1 or m = 0 before iteration

Step size for first iteration

LR=0.002, m=0, c=0 , epoch=0, sample size=10

```
Explanatory_variable=np.array  ([1,2,4,3,34,78,5,4,3,5])
Explained_variable=np.array     ([1,3,3,2,45,6,7,0,345,2])
```

3.1)Plot Correlations heatmap.

```
df=pd.read_excel('/content/drive/MyDrive/Data/ML_EVAL_1 (1).xlsx')
```

```
df.head()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	work experience	Family income	Age	Chance of Admit
0	1.0	337.0	118.0	4.0	4.5	4.5	9.65	NaN	360000.0	18.0	0.92
1	2.0	324.0	107.0	4.0	4.0	4.5	8.87	1.0	320000.0	19.0	0.76
2	3.0	316.0	104.0	3.0	3.0	3.5	8.00	1.0	240000.0	20.0	0.72
3	4.0	322.0	110.0	3.0	3.5	2.5	8.67	1.0	280000.0	21.0	0.80
4	5.0	314.0	103.0	2.0	2.0	3.0	8.21	0.0	160000.0	22.0	0.65

```
df.columns
```

```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR', 'CGPA', 'work experience', 'Family income', 'Age',
       'Chance of Admit'],
      dtype='object')
```

```
df1=df.drop(['Serial No.'],axis=1)
```

```
df1.head()
```

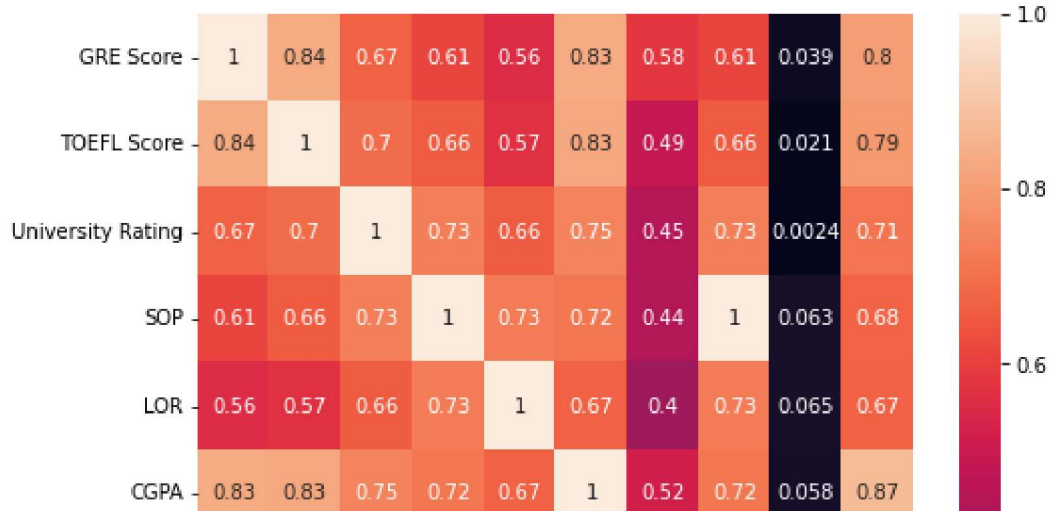
	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	work experience	Family income	Age	Chance of Admit
0	337.0	118.0	4.0	4.5	4.5	9.65	NaN	360000.0	18.0	0.92
1	324.0	107.0	4.0	4.0	4.5	8.87	1.0	320000.0	19.0	0.76
2	316.0	104.0	3.0	3.0	3.5	8.00	1.0	240000.0	20.0	0.72
3	322.0	110.0	3.0	3.5	2.5	8.67	1.0	280000.0	21.0	0.80
4	314.0	103.0	2.0	2.0	3.0	8.21	0.0	160000.0	22.0	0.65

```
df1.corr()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	work experience	Family income	Age	Chance of Admit
GRE Score	1.000000	0.835977	0.668976	0.612831	0.557555	0.833060	0.579232	0.612831	0.039481	0.802610
TOEFL Score	0.835977	1.000000	0.695590	0.657981	0.567721	0.828417	0.488252	0.657981	0.020596	0.791594
University Rating	0.668976	0.695590	1.000000	0.734523	0.660123	0.746479	0.446782	0.734523	0.002431	0.711250
SOP	0.612831	0.657981	0.734523	1.000000	0.729593	0.718144	0.442659	1.000000	0.062635	0.675732
LOR	0.557555	0.567721	0.660123	0.729593	1.000000	0.670211	0.395284	0.729593	0.065412	0.669889
CGPA	0.833060	0.828417	0.746479	0.718144	0.670211	1.000000	0.520204	0.718144	0.058031	0.873289
work experience	0.579232	0.488252	0.446782	0.442659	0.395284	0.520204	1.000000	0.442659	0.065632	0.551950
Family income	0.612831	0.657981	0.734523	1.000000	0.729593	0.718144	0.442659	1.000000	0.062635	0.675732

```
plt.figure(figsize=(8,8))
sbr.heatmap(df1.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f22cf6976d0>
```



2) To predict the chance of admission, which type of models can be used? Give 3 examples.

1.Linear regression

2.

Double-click (or enter) to edit

3) Find null values.

```
df.isnull().sum()
```

```
Serial No.           0
GRE Score            0
```

```
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
work experience      1
Family income        0
Age                  0
Chance of Admit      0
dtype: int64
```

- Work experience has one null Value.

```
0
```

- 4) Find duplicate values (At Least 7 columns should have duplicate value) don'tinclude predicted value.

```
df.duplicated().sum()
```

```
0
```

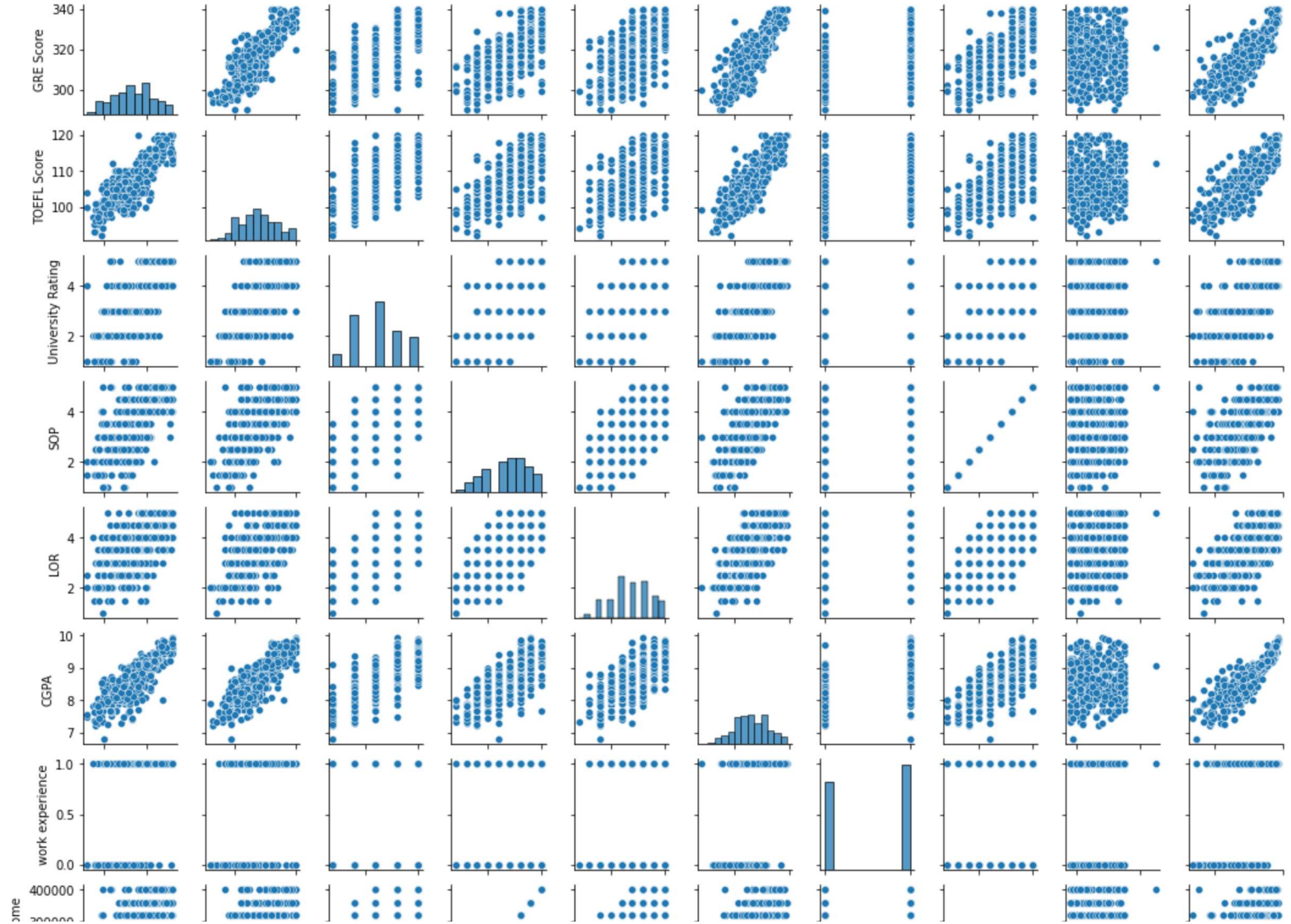
5) Import the dataset and save into csv json and excel file in your local system(Please write the code only, don't need to upload the file).

- 6) Which columns are highly co-relate?

1. GRE Score
2. Tofel Score
3. CGPA

```
sbr.pairplot(df1,height=1.5)
```

```
<seaborn.axisgrid.PairGrid at 0x7f22cf327850>
```



7) Find out the value of the intercept and coefficient model that you train.

```
df1.head()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	work experience	Family income	Age	Chance of Admit
0	337.0	118.0	4.0	4.5	4.5	9.65	NaN	360000.0	18.0	0.92
1	324.0	107.0	4.0	4.0	4.5	8.87	1.0	320000.0	19.0	0.76
2	316.0	104.0	3.0	3.0	3.5	8.00	1.0	240000.0	20.0	0.72
3	322.0	110.0	3.0	3.5	2.5	8.67	1.0	280000.0	21.0	0.80
4	314.0	103.0	2.0	2.0	3.0	8.21	0.0	160000.0	22.0	0.65

```
x=df1[['GRE Score','TOEFL Score','CGPA']]
y=df1[['Chance of Admit']]
```

```
df2.head()
```

	GRE Score	TOEFL Score	CGPA	Chance of Admit
0	337.0	118.0	9.65	0.92
1	324.0	107.0	8.87	0.76
2	316.0	104.0	8.00	0.72
3	322.0	110.0	8.67	0.80
4	314.0	103.0	8.21	0.65

```
le=LabelEncoder()
lr=LinearRegression()
```

```
x.shape
```

```
(400, 3)
```

```
y.shape
```

```
(400, 1)
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)


lr.fit(x_train,y_train)

     LinearRegression()


print('coefficient model',lr.coef_)

     coefficient model [[0.00207483 0.00272864 0.151904  ]]


  print('coefficient model',lr.coef_)
  print('Intercept of model',lr.intercept_)

     coefficient model [[0.00207483 0.00272864 0.151904  ]]
     Intercept of model [-1.53201899]
```

✓ 0s completed at 12:00 AM