Problem Statement:1

In [1]:
```python
import pandas as pd
```

In [2]:
```python
Admission=pd.read_excel("E:\Data science training\R AND PYTHON KPMG\stat and ml\ASSIGNMENT\ADMISSION_DATA.xlsx")
Admission.head()
```

Out[2]:

	admit	gre	gpa	rank
0	0	380	3.61	3
1	1	660	3.67	3
2	1	800	4.00	1
3	1	640	3.19	4
4	0	520	2.93	4

In [3]:
```python
Admission1=Admission[['admit','gpa']]
Admission1
```

Out[3]:

	admit	gpa
0	0	3.61
1	1	3.67
2	1	4.00
3	1	3.19
4	0	2.93
...	...	...
395	0	4.00
396	0	3.04
397	0	2.63
398	0	3.65
399	0	3.89

400 rows × 2 columns

In [6]:
```python
## Defining the X and Y
Y = Admission1[['admit']]

X = Admission1[['gpa']]
```

In [8]:
```python
#For building the model i keep train_data  to 80%
#### and for testing i should keep 20% of datapoints

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, train_size = 0.8, random_state = 1234)

len(X_train), len(X_test), len(Y_train), len(Y_test)
```

Out[8]:
```
(320, 80, 320, 80)
```

In [9]:
```python
## Define our model object

from sklearn.linear_model import LinearRegression

slr = LinearRegression()

### Fit this model object on our training dataset

model = slr.fit(X_train, Y_train)

model
```

Out[9]:
```
LinearRegression()
```

In [10]:
```python
# Y = mX + C

#  Find the values of slope, intercept

## slope is m, Intercept is C, X is gpa, Y is admit

print(model.coef_) # this will give the m value
print(model.intercept_) # this will give me the Constant/Intercept value
```

```
[[0.17503163]]
[-0.24152309]
```

In [48]:
```python
# Find the R-sq value of my model .

r_sq = model.score(X_train,Y_train)
r_sq
```

```
C:\Users\Pratik\anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning: The feature names should match those that were passed during fit. Starting ver
sion 1.2, an error will be raised.
Feature names unseen at fit time:
- gpa
Feature names seen at fit time, yet now missing:
- gre

  warnings.warn(message, FutureWarning)
```

Out[48]:
```
-0.801554781018516
```

admit=(0.17503163*gpa) + (-0.24152309)

so here one unit increase in gpa means chance of getting admit ={(0.175031632)-0.24152309}-{(0.175031631)-0.24152309}= 0.17503163

In [22]:
```python
(0.17503163*2)-0.24152309
```

Out[22]:
```
0.10854016999999999
```

In [24]:
```python
(0.17503163*1)-0.24152309
```

Out[24]:
```
-0.06649146
```

In [25]:
```python
0.10854016999999999-(-0.06649146)
```

Out[25]:
```
0.17503163
```

In [26]:
```python
## b)For one unit increase in GRE , the chance of being admitted to graduate school increase by What amount?

Admission.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   admit   400 non-null    int64  
 1   gre     400 non-null    int64  
 2   gpa     400 non-null    float64
 3   rank    400 non-null    int64  
dtypes: float64(1), int64(3)
memory usage: 12.6 KB
```

In [27]:
```python
Admission2=Admission[['admit','gre']]
Admission2.head()
```

Out[27]:

	admit	gre
0	0	380
1	1	660
2	1	800
3	1	640
4	0	520

In [29]:
```python
## Defining the X and Y
Y1 = Admission2[['admit']]

X1 = Admission2[['gre']]
```

In [58]:
```python
#For building the model i keep train_data  to 80%
#### and for testing i should keep 20% of datapoints

from sklearn.model_selection import train_test_split

X1_train, X1_test, Y1_train, Y1_test = train_test_split(X1,Y1, train_size = 0.8, random_state = 1234)

len(X1_train), len(X1_test), len(Y1_train), len(Y1_test)
```

Out[58]:
```
(320, 80, 320, 80)
```

In [31]:
```python
## Define our model object

from sklearn.linear_model import LinearRegression

slr1 = LinearRegression()

### Fit this model object on our training dataset

model1 = slr1.fit(X1_train, Y1_train)

model1
```

Out[31]:
```
LinearRegression()
```

In [57]:
```python
# Y = mX + C

#  Find the values of slope, intercept

## slope is m, Intercept is C, X is gpa, Y is admit

print(model1.coef_) # this will give the m value
print(model1.intercept_) # this will give me the Constant/Intercept value
# Find the R-sq value of my model .

r_sq = model1.score(X1_train,Y1_train)
r_sq
```

```
[[0.00072914]]
[-0.0772943]
```

Out[57]:
```
0.03126836704385416
```

admit=(0.00072914*gre) + (-0.0772943)

so here one unit increase in gpa means chance of getting admit ={(0.000729142) + (-0.0772943)}-{(0.000729141) + (-0.0772943)}= 0.0007291400000000031 increased.

In [50]:
```python
(0.00072914*1) + (-0.0772943)
```

Out[50]:
```
-0.07656516
```

In [49]:
```python
(0.00072914*2) + (-0.0772943)
```

Out[49]:
```
-0.07583601999999999
```

In [36]:
```python
-0.07583601999999999-(-0.07656516)
```

Out[36]:
```
0.0007291400000000031
```

In [40]:
```python
#C)Having attended an under graduate institute with  of RANK 2 versus an institution of rank 1 increase the chance af admission
## or decerase the chance of an admission or by what ammount?
Admission3=Admission[['admit','rank']]
Admission3.head()
```

Out[40]:

	admit	rank
0	0	3
1	1	3
2	1	1
3	1	4
4	0	4

In [52]:
```python
## Defining the X and Y
Y2 = Admission3[['admit']]

X2 = Admission3[['rank']]
```

In [53]:
```python
#For building the model i keep train_data  to 80%
#### and for testing i should keep 20% of datapoints

from sklearn.model_selection import train_test_split

X2_train, X2_test, Y2_train, Y2_test = train_test_split(X2,Y2, train_size = 0.8, random_state = 1234)

len(X2_train), len(X2_test), len(Y2_train), len(Y2_test)
```

Out[53]:
```
(320, 80, 320, 80)
```

In [44]:
```python
## Define our model object

from sklearn.linear_model import LinearRegression

slr2 = LinearRegression()

### Fit this model object on our training dataset

model2 = slr2.fit(X2_train, Y2_train)

model2
```

Out[44]:
```
LinearRegression()
```

In [59]:
```python
# Y2 = mX2 + C

#  Find the values of slope, intercept

## slope is m, Intercept is C, X2 is rank, Y2 is admit

print(model2.coef_) # this will give the m value
print(model2.intercept_) # this will give me the Constant/Intercept value
# Find the R-sq value of my model .

r_sq2 = model2.score(X2_train,Y2_train)
r_sq2
```

```
[[-0.12442793]]
[0.65291854]
```

Out[59]:
```
0.060019678563658974
```

admit =(-0.12442793rank) + 0.65291854 Chance of Getting admit Rank 2 vs Rank 1: chance of getting admit ={(-0.124427932) + 0.65291854}-{(-0.12442793*rank) + 0.65291854)=-0.12442793000000008 So here We can see that chance of getting admit to an under graduate institution are Decreased by 0.12442793

In [63]:
```python
Rank2=( -0.12442793*2) + 0.65291854
```

In [64]:
```python
Rank1=( -0.12442793*1) + 0.65291854
```

In [66]:
```python
Chance_value=Rank2-Rank1
Chance_value
```

Out[66]:
```
-0.12442793000000008
```

In [ ]:
```python
## question D:using your model if a candidate with Rank=3, GRE=180 and Gpa =2.1 wii get admission or not?
```

In [67]:
```python
Admission.head()
```

Out[67]:

	admit	gre	gpa	rank
0	0	380	3.61	3
1	1	660	3.67	3
2	1	800	4.00	1
3	1	640	3.19	4
4	0	520	2.93	4

In [88]:
```python
## Defining the X and Y
Y4 = Admission[['admit']]

X4 = Admission[['gre','gpa','rank']]
```

In [89]:
```python
#For building the model i keep train_data  to 80%
#### and for testing i should keep 20% of datapoints

from sklearn.model_selection import train_test_split

X4_train, X4_test, Y4_train, Y4_test = train_test_split(X4,Y4, train_size = 0.8, random_state = 1234)

len(X4_train), len(X4_test), len(Y4_train), len(Y4_test)
```

Out[89]:
```
(320, 80, 320, 80)
```

In [90]:
```python
# Build our model on training data... this is a two step process

# create a model object

from sklearn.linear_model import LogisticRegression

LR = LogisticRegression()

# fit the model object on training data for building the model

model_lr = LR.fit(X4_train, Y4_train)

model_lr
```

Out[90]:
```
LogisticRegression()
```

In [92]:
```python
Y4_test['Pred_admit'] = model_lr.predict(X4_test)
```

In [93]:
```python
Y4_test
```

Out[93]:

	admit	Pred_admit
44	0	0
110	0	0
78	0	0
181	0	0
285	0	0
...	...	...
299	0	0
55	1	0
129	0	0
153	0	0
90	0	1

80 rows × 2 columns

In [98]:
```python
## Create a validation data
val_data = pd.DataFrame({"gre" : [180],"gpa" : [2.1],"rank" : [3]})

#### predict the Admit offered
Predict_admit = model_lr.predict(val_data)

Predict_admit
```

Out[98]:
```
array([0], dtype=int64)
```

In [99]:
```python
## using My Model,I can conclude that candidate with rank 3, GRE =180, Gpa=2.1 will not get admission
```

In [ ]:

In [ ]: