Data Exploration GRE Scores Case Study

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
#reading the data
df= pd.read_csv("/content/Admission_Predict.csv")
#how the data looks
df.head()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

```
df.shape
```

```
(400, 8)
```

```
print("DATA INFORMATION AND DATA TYPES")
df.info()
```

```
DATA INFORMATION AND DATA TYPES
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   GRE Score          400 non-null    int64
 1   TOEFL Score        400 non-null    int64
 2   University Rating  400 non-null    int64
 3   SOP                400 non-null    float64
 4   LOR                400 non-null    float64
 5   CGPA               400 non-null    float64
 6   Research           400 non-null    int64
 7   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 25.1 KB
```

```
df.drop("SOP",axis=1,inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   GRE Score          400 non-null    int64
 1   TOEFL Score        400 non-null    int64
 2   University Rating  400 non-null    int64
 3   SOP                400 non-null    float64
 4   LOR                400 non-null    float64
 5   CGPA               400 non-null    float64
 6   Research           400 non-null    int64
```

```
 7   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 25.1 KB
```

```
print('MISSING DATA (IF ANY)')
df.isnull().sum()
```

```
MISSING DATA (IF ANY)
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

```
df.describe()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Cl
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	40
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	

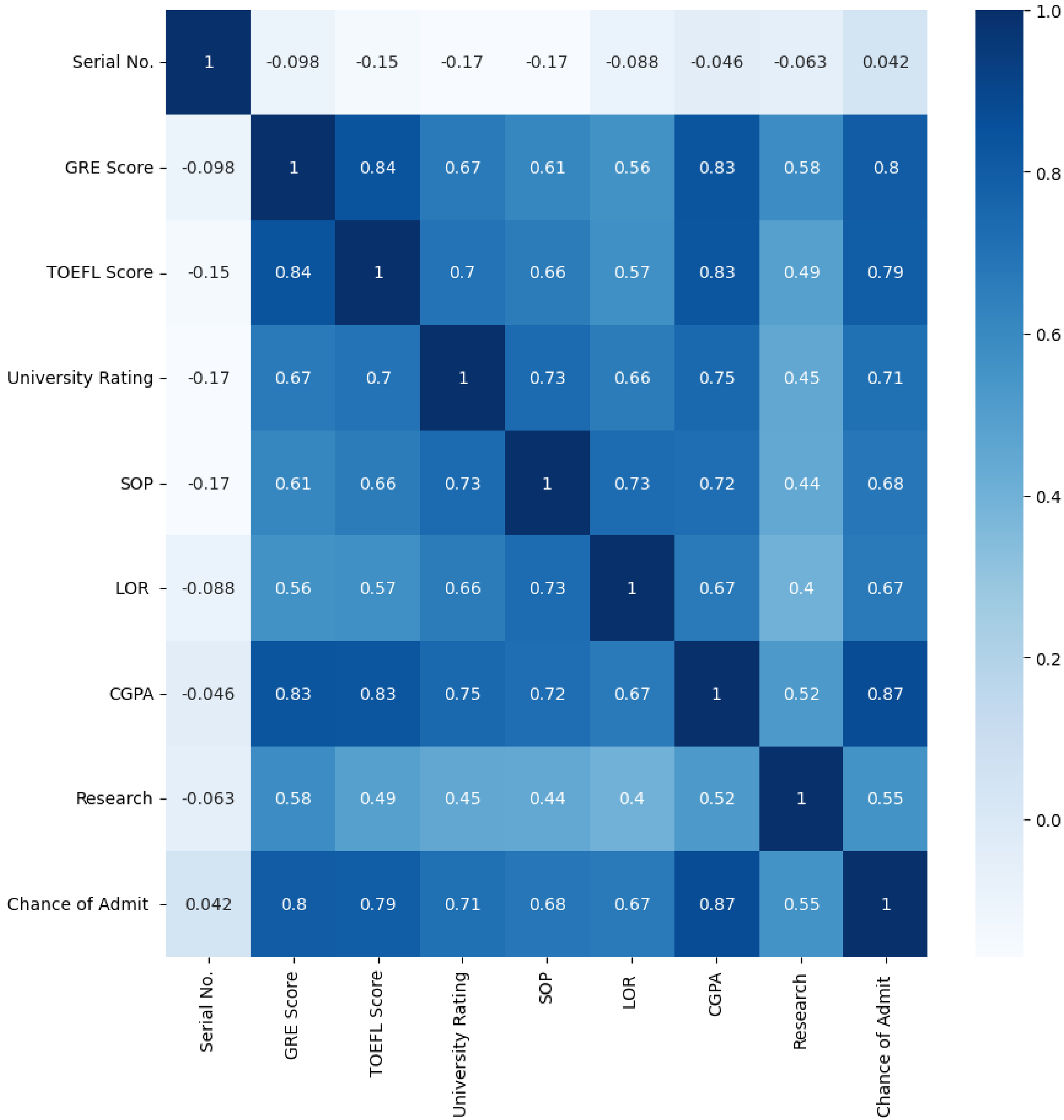
```
df.corr()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Cl
Serial No.	1.000000	-0.097526	-0.147932	-0.169948	-0.166932	-0.088221	-0.045608	-0.063138	
GRE Score	-0.097526	1.000000	0.835977	0.668976	0.612831	0.557555	0.833060	0.580391	
TOEFL Score	-0.147932	0.835977	1.000000	0.695590	0.657981	0.567721	0.828417	0.489858	
University Rating	-0.169948	0.668976	0.695590	1.000000	0.734523	0.660123	0.746479	0.447783	
SOP	-0.166932	0.612831	0.657981	0.734523	1.000000	0.729593	0.718144	0.444029	
LOR	-0.088221	0.557555	0.567721	0.660123	0.729593	1.000000	0.670211	0.396859	
CGPA	-0.045608	0.833060	0.828417	0.746479	0.718144	0.670211	1.000000	0.521654	
Research	-0.063138	0.580391	0.489858	0.447783	0.444029	0.396859	0.521654	1.000000	
Chance of Admit	0.042336	0.802610	0.791594	0.711250	0.675732	0.669889	0.873289	0.553202	

There is a 0.802 correlation between the GRE score and the chance of admission. So there might be a big chance that these variables (data) are highly related. In fact, the correlation is the second-highest, after the CGPA. So, we can determine that CGPA and GRE scores are most important in determining the chances of admission.

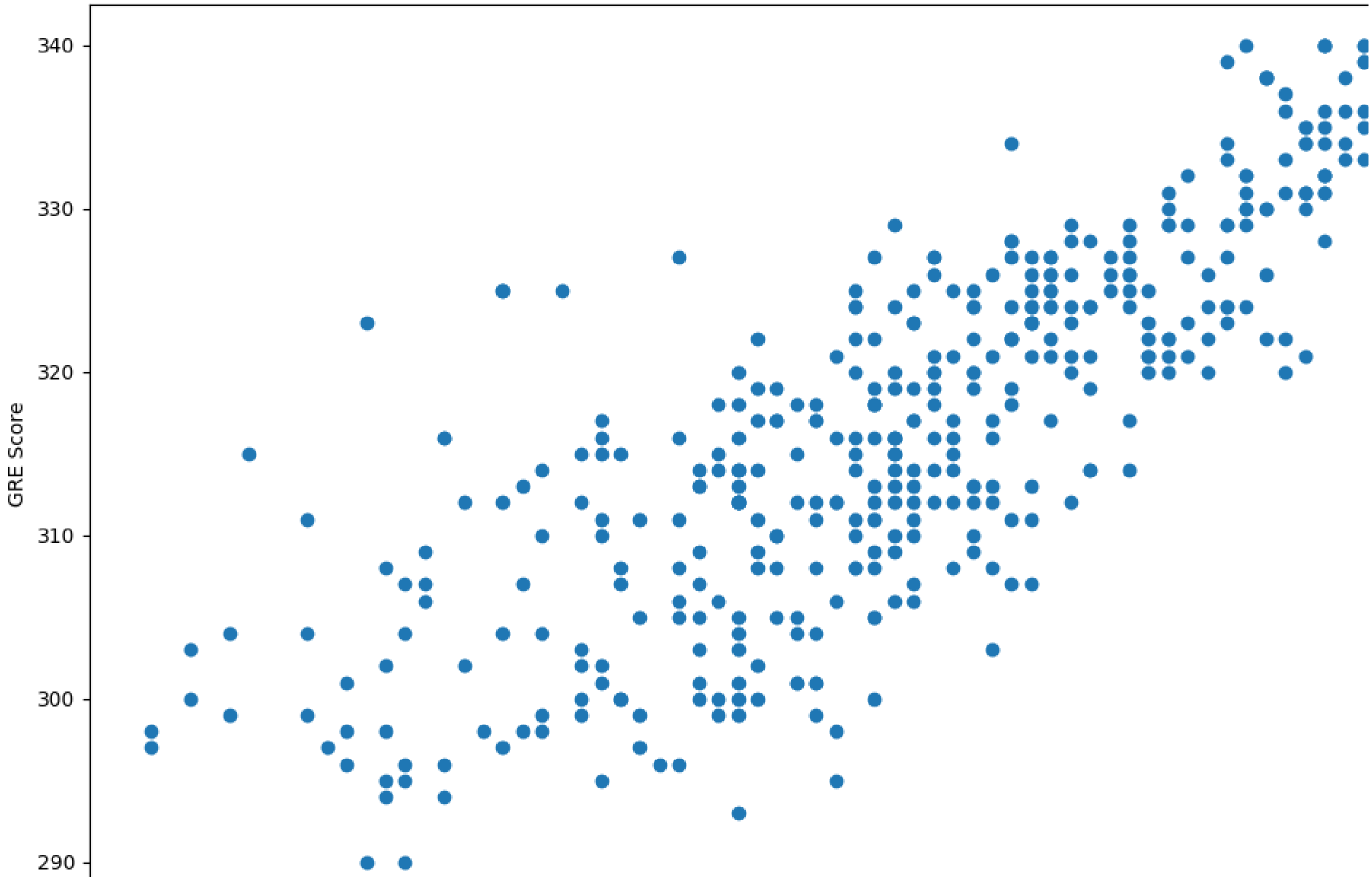
```
plt.figure(figsize = (10,10))
sns.heatmap(df.corr(),annot=True, cmap='Blues')
```

```
<Axes: >
```




```
plt.subplots(figsize=(12,8))
plt.scatter(df["Chance of Admit "],df["GRE Score"])
plt.xlabel("Chance of Admit")
plt.ylabel("GRE Score")
```

```
Text(0, 0.5, 'GRE Score')
```



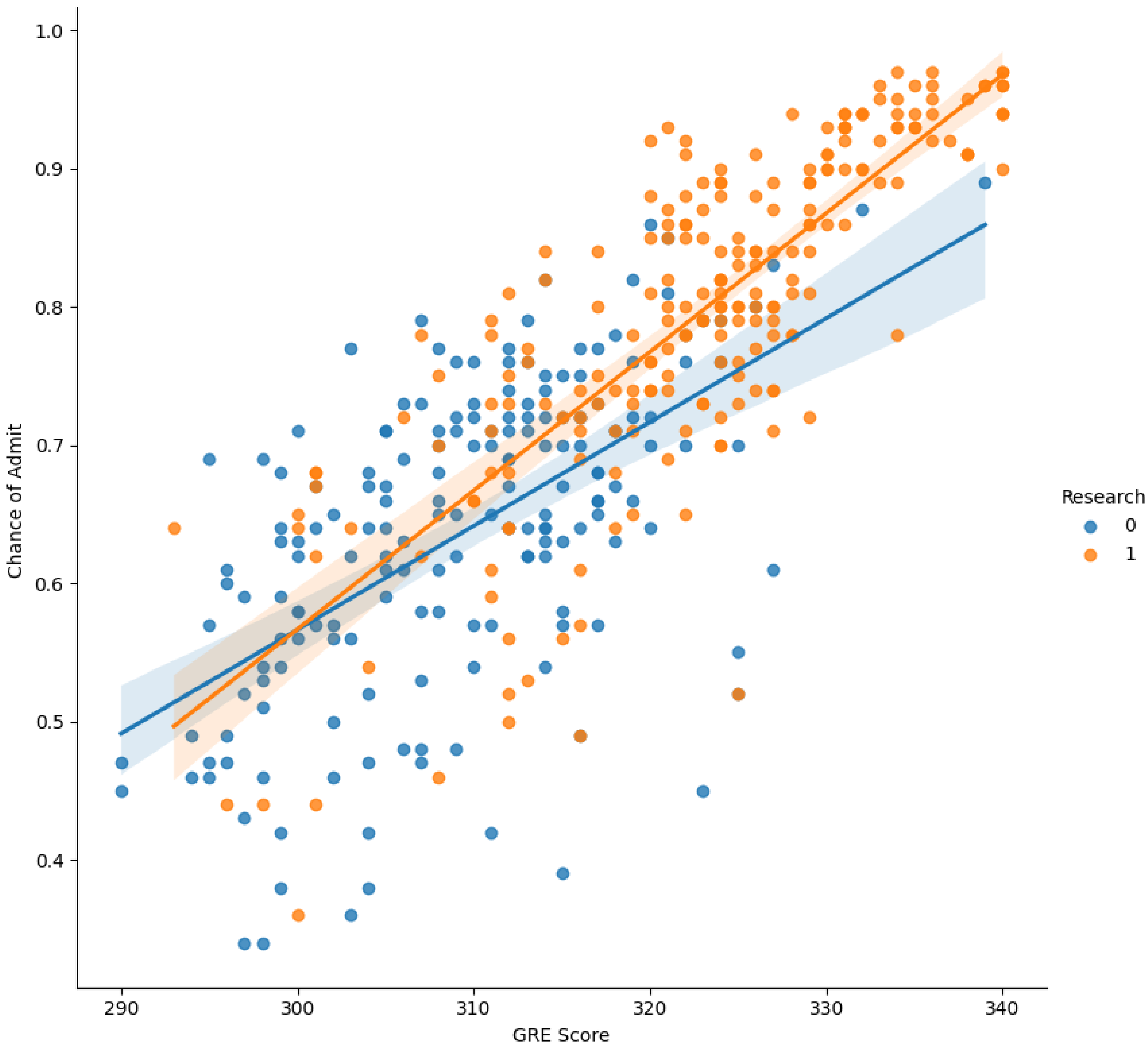

```
#There does appear to be a connection between the two variables. Some exploration needs to be done.

plt.subplots(figsize=(12,8))
sns.regplot(x="GRE Score", y="Chance of Admit ", data=df)
```

<Axes: xlabel='GRE Score', ylabel='Chance of Admit '>


```
# Research experience of a candidate helps in getting admits
sns.lmplot(x="GRE Score", y="Chance of Admit ", data=df, hue="Research",height= 8)
#The data does show that candidates having research experience (orange in the figure), usually have more chance of admit
#Having research experience is very important.
```

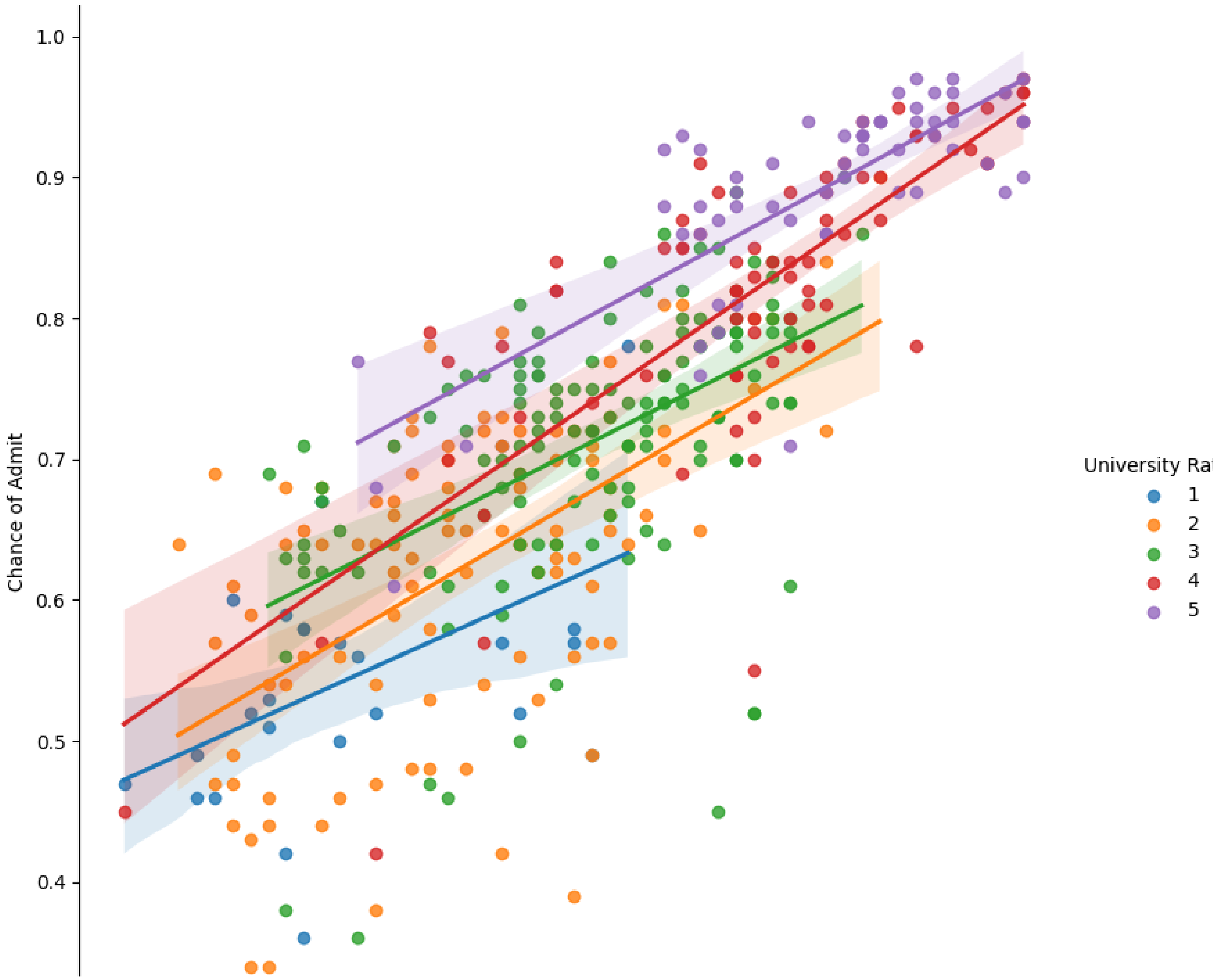
<seaborn.axisgrid.FacetGrid at 0x7e0b50fc12a0>




```
#university ratings

sns.lmplot(x="GRE Score", y="Chance of Admit ", data=df, hue="University Rating",height=8)
```

```
<seaborn.axisgrid.FacetGrid at 0x7e0b50bc96c0>
```



Observations :

Students having higher GRE scores (>320) usually have a high chance of admission into the university with higher ratings (4/5). A lower GRE score has a lower chance of admission, that too for universities of low ratings. Students having a higher chance of admission, all have good GRE scores and University ratings of 4 or 5. Now we take some data where we take chances of admit to being 0.8 or higher and check how important are GRE scores.

```
admit_high_chance= df[df["Chance of Admit "]>=0.8]
admit_high_chance.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 128 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         128 non-null    int64
 1   GRE Score          128 non-null    int64
 2   TOEFL Score        128 non-null    int64
 3   University Rating  128 non-null    int64
 4   SOP                128 non-null    float64
 5   LOR                128 non-null    float64
 6   CGPA               128 non-null    float64
 7   Research           128 non-null    int64
 8   Chance of Admit    128 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 10.0 KB
```

```
admit_high_chance.corr()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Cl
Serial No.	1.000000	-0.140435	-0.223184	-0.211793	-0.088391	-0.141164	-0.220561	-0.031246	-
GRE Score	-0.140435	1.000000	0.722463	0.358013	0.320138	0.246629	0.754434	0.167532	
TOEFL Score	-0.223184	0.722463	1.000000	0.274811	0.337175	0.302047	0.648308	0.083921	
University Rating	-0.211793	0.358013	0.274811	1.000000	0.584860	0.531448	0.479284	0.190083	
SOP	-0.088391	0.320138	0.337175	0.584860	1.000000	0.601405	0.519791	0.148911	
LOR	-0.141164	0.246629	0.302047	0.531448	0.601405	1.000000	0.441634	0.050772	
CGPA	-0.220561	0.754434	0.648308	0.479284	0.519791	0.441634	1.000000	0.158186	
Research	-0.031246	0.167532	0.083921	0.190083	0.148911	0.050772	0.158186	1.000000	
Chance of Admit	-0.227214	0.716187	0.673774	0.584556	0.565463	0.488480	0.871533	0.226028	

Now let us look at the distribution of Chance of Admit and GRE score.

```
plt.subplots(figsize=(12,8))
sns.set_theme(style="darkgrid")
sns.distplot( admit_high_chance["GRE Score"])
```

```
<ipython-input-12-02911b31d2ab>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot( admit_high_chance["GRE Score"])
<Axes: xlabel='GRE Score', ylabel='Density'>
```

```
plt.subplots(figsize=(12,8))
sns.set_theme(style="darkgrid")
sns.distplot( admit_high_chance["Chance of Admit "])
```

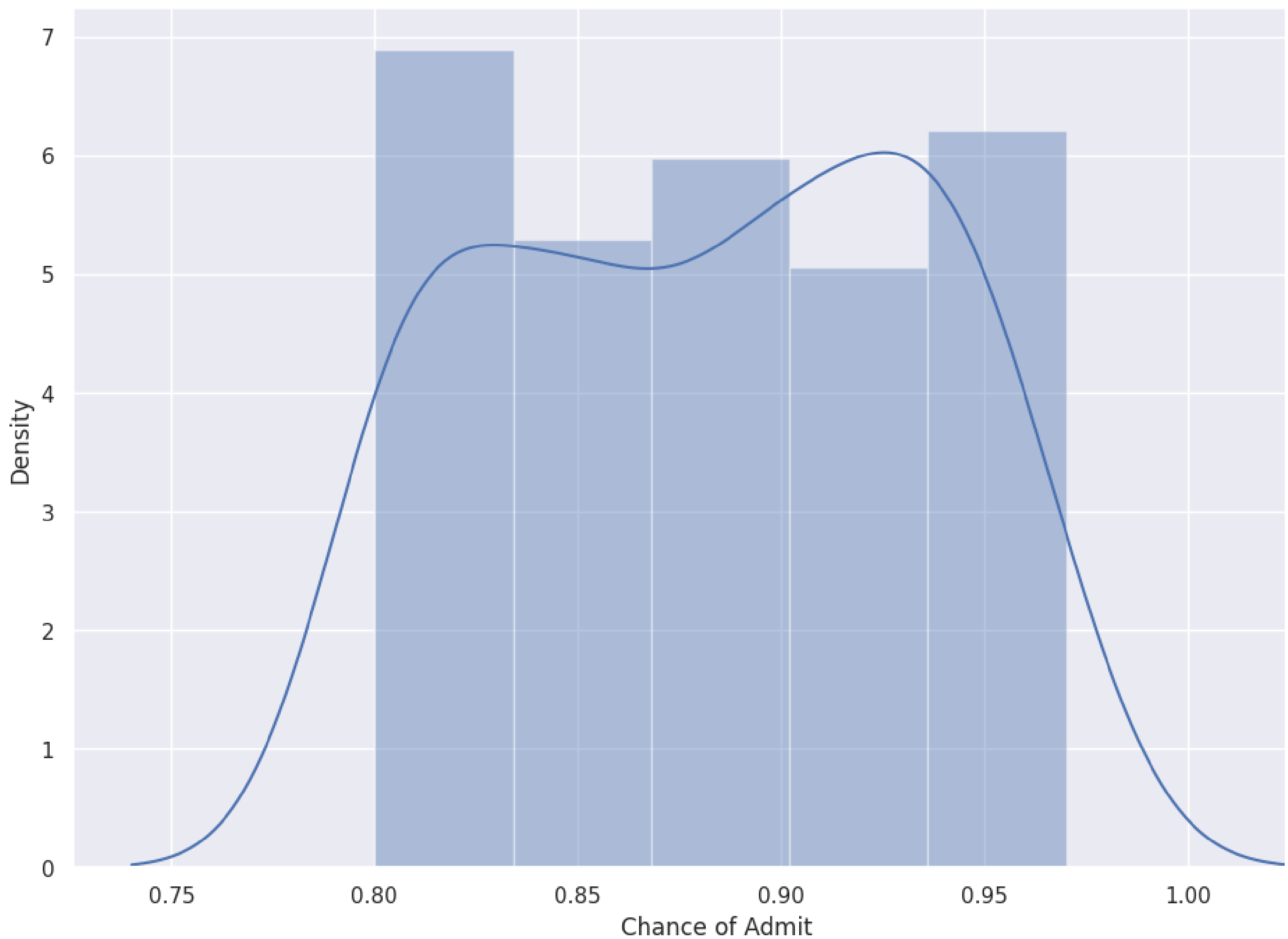
```
<ipython-input-13-ab381f61a609>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot( admit_high_chance["Chance of Admit "])
<Axes: xlabel='Chance of Admit ', ylabel='Density'>
```



Observations :

For a higher chance of admission, the GRE score is also high. Maximum GRE scores are in the range of 320-340.

```
#Linear Regression between GRE Scores and the chance of admit:
X= df["GRE Score"].values
#bringing GRE score in a range of 0-1

X=X/340
y= df["Chance of Admit "].values
#sk learn train test split data

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
#sk learn linear regression

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
#training the model on training data
lr.fit(X_train.reshape(-1,1), y_train)
y_pred = lr.predict(X_test.reshape(-1,1))
#model score

lr.score(X_test.reshape(-1,1),y_test.reshape(-1,1))
```

```
0.6334295343566941
```

```
plt.subplots(figsize=(12,8))
plt.scatter(X_train, y_train, color = "red")
plt.plot(X_train, lr.predict(X_train.reshape(-1,1)), color = "green")
plt.title("GRE Score vs Chance of Admit")
plt.xlabel("GRE Score")
plt.ylabel("Chance Of Admit")
plt.show()
```

The model is not performing that well, but we do understand that there is a correlation between GRE scores and the chance of admit.

```
#test input

test= 320
val= test/340

val_out=lr.predict(np.array([[val]]))

print("Chance of admission :", val_out[0])
```

```
Chance of admission : 0.754513490079177
```

```
#Creating a Model on the entire data:
x = df.drop(['Chance of Admit ','Serial No.'],axis=1)
y = df['Chance of Admit ']
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.25, random_state = 7)
#random forest regression

from sklearn.ensemble import RandomForestRegressor

regr = RandomForestRegressor(max_depth=2, random_state=0, n_estimators=5)
regr.fit(X_train,y_train)
regr.score(X_test, y_test)
```

```
0.6901443456671795
```

```
#Let us work with a sample input.

val=regr.predict([[325, 100, 3, 4.1, 3.7, 7.67, 1]])

print("Your chances are (in %):")
print(val[0]*100)
```

```
Your chances are (in %):
54.47694678499888
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but
  warnings.warn(
```

Conclusion: GRE Score is important for admission. Students having good GRE score, seem to have good overall profiles. There are obviously exceptions, which comprise the outliers.

A machine learning model classifier using Decision tree to predict

```
df.head()
```

```
df['Chance of Admit '] = [1 if each > 0.75 else 0 for each in df['Chance of Admit ']]
df.head()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	1
1	2	324	107	4	4.0	4.5	8.87	1	1
2	3	316	104	3	3.0	3.5	8.00	1	0
3	4	322	110	3	3.5	2.5	8.67	1	1
4	5	314	103	2	2.0	3.0	8.21	0	0

```
x = df[['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research']]

y = df['Chance of Admit ']
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25,random_state=1)
```

```
print(f"Size of splitted data")
print(f"x_train {x_train.shape}")
print(f"y_train {y_train.shape}")
print(f"y_train {x_test.shape}")
print(f"y_test {y_test.shape}")
```

```
Size of splitted data
x_train (300, 7)
y_train (300,)
y_train (100, 7)
y_test (100,)
```

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LogisticRegression
model_dt = DecisionTreeRegressor(random_state=1)
model_rf = RandomForestRegressor(random_state=1)
model_lr = LogisticRegression(random_state=1,solver='lbfgs',max_iter=1000)
```

```
model_dt.fit(x_train,y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=1)
```

```
model_rf.fit(x_train,y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=1)
```

```
model_lr.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression(max_iter=1000, random_state=1)
```

```
y_pred_dt = model_dt.predict(x_test) #int
```

```
y_pred_rf = model_rf.predict(x_test) #float
y_pred_lr = model_lr.predict(x_test) #
```

```
result = pd.DataFrame({
       "Actual": y_test,
       "predicted" : y_pred_dt })
result
```

	Actual	predicted
398	0	1.0
125	0	0.0
328	1	1.0
339	1	0.0
172	1	1.0
...	...	...
300	0	0.0
277	0	0.0
289	1	0.0
260	1	1.0
173	1	1.0

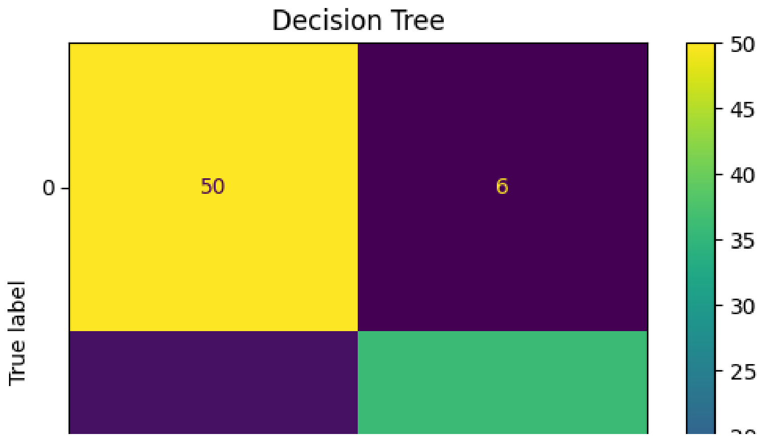
100 rows × 2 columns

```
 y_pred_rf = [1 if each > 0.75 else 0 for each in y_pred_rf]
```

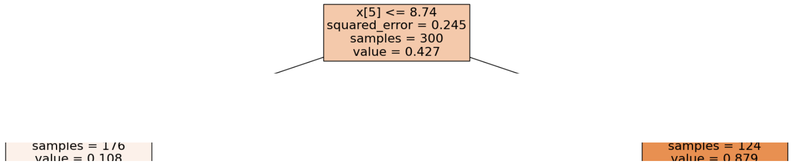
```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score
from sklearn.metrics import classification_report
```

Decision Tree

```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_dt)
plt.title('Decision Tree')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_dt)}")
print(classification_report(y_test,y_pred_dt))
```

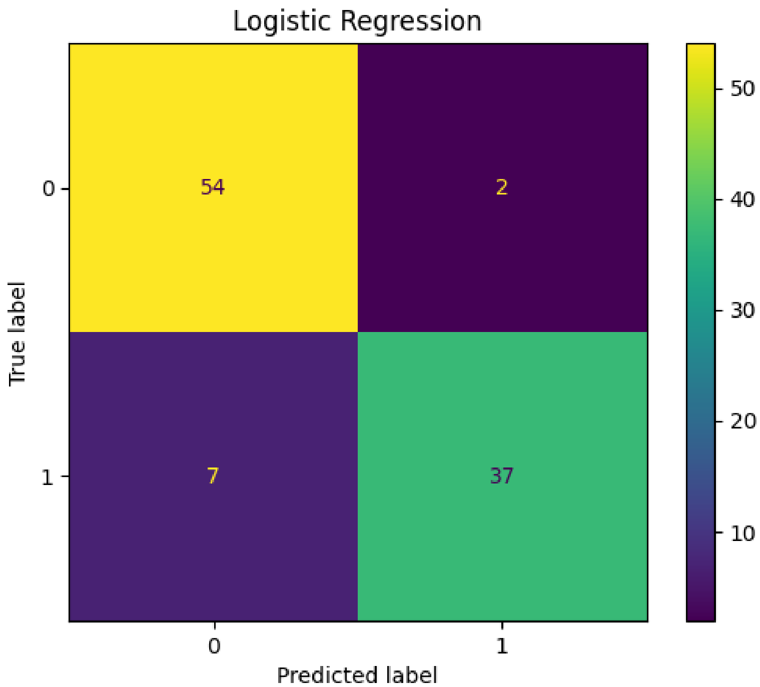



```
from sklearn import tree
import matplotlib.pyplot as plt
plt.figure(figsize=(30,30))
tree.plot_tree(model_dt, filled=True, fontsize=16)
plt.show()
```



Logistic Regression

```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_lr)
plt.title('Logistic Regression')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_lr)}")
print(classification_report(y_test,y_pred_lr))
```



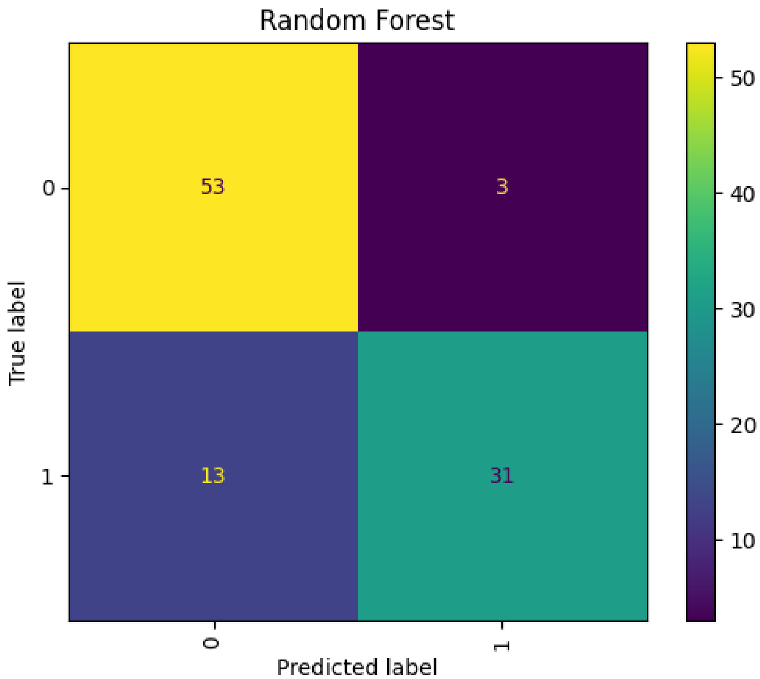

```
Accuracy is 0.91
              precision    recall  f1-score   support

           0       0.89      0.96      0.92        56
           1       0.95      0.84      0.89        44

    accuracy                           0.91       100
   macro avg       0.92      0.90      0.91       100
weighted avg       0.91      0.91      0.91       100
```

Random Forest

```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_rf,xticks_rotation='vertical')
plt.title('Random Forest')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_rf)}")
print(classification_report(y_test,y_pred_rf))
```




```
Accuracy is 0.84

           0       0.80      0.95      0.87        56
           1       0.91      0.70      0.79        44

    accuracy                           0.84       100
   macro avg       0.86      0.83      0.83       100
weighted avg       0.85      0.84      0.84       100
```