Gemini

Jamboore Admission case study

Jamboree has helped thousands of students like you make it to top colleges abroad. Be it GMAT, GRE or SAT, their unique problem-solving methods ensure maximum scores with minimum effort.

They recently launched a feature where students/learners can come to their website and check their probability of getting into the IVY league college.

This feature estimates the chances of graduate admission from an Indian perspective.

How can you help here?

Your analysis will help Jamboree in understanding what factors are important in graduate admissions and how these factors are interrelated among themselves. It will also help predict one's chances of admission given the rest of the variables.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('/content/Jamboree_Admission.csv')
```

Double-click (or enter) to edit

```
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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.describe()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	L
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.484
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.925
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.500
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.000

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

```
df.shape
```

```
(500, 8)
```

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

```
0
```

```
df.isna().sum()
```

	0
GRE Score	0
TOEFL Score	0
University Rating	0
SOP	0
LOR	0
CGPA	0
Research	0
Chance of Admit	0

dtype: int64

Found out to be no null and no duplicates

```
df.columns.values
```

```
array(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ',
       'CGPA', 'Research', 'Chance of Admit '], dtype=object)
```

```
df.dtypes
```

	0
GRE Score	int64
TOEFL Score	int64
University Rating	int64
SOP	float64
LOR	float64
CGPA	float64
Research	int64
Chance of Admit	float64

dtype: object

```
a=df['GRE Score'].sort_values(ascending=True).head()
b=df['GRE Score'].sort_values(ascending=False).head()
a
```

	GRE Score
377	290
117	290
168	293
79	294
272	294

dtype: int64

```
b
```

	GRE Score
429	340
84	340
81	340
143	340
202	340

dtype: int64

```
x=df['GRE Score'].sum()
y=df['GRE Score'].count()
```

```
avg=x/y
avg
```

```
316.472
```

AVG GRE SCORE-316.472

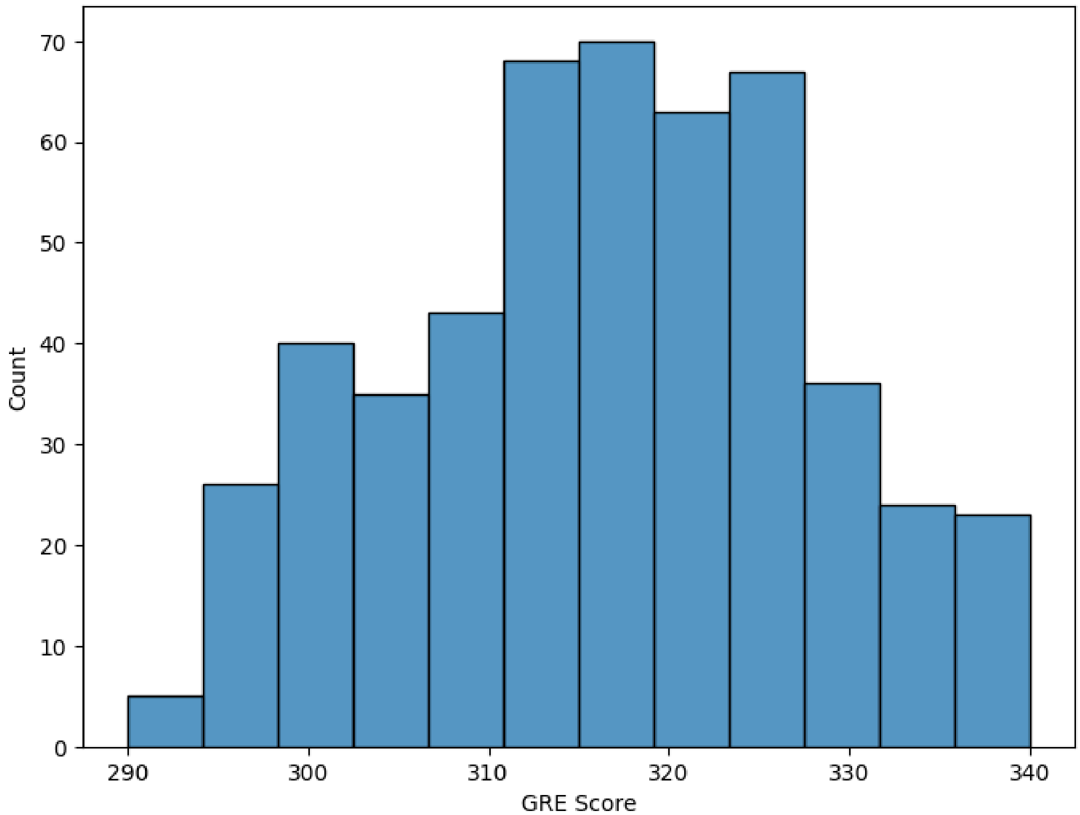
Least GRE Score- 290

Top GRE Score-340

```
plt.figure(figsize=(8,6))
sns.histplot(x='GRE Score',data=df)
plt.xticks
```

```
matplotlib.pyplot.xticks
def xticks(ticks: ArrayLike | None=None, labels: Sequence[str] |
None=None, *, minor: bool=False, **kwargs) -> tuple[list[Tick] |
np.ndarray, list[Text]]
```

```
/usr/local/lib/python3.11/dist-packages/matplotlib/pyplot.py
Get or set the current tick locations and labels of the x-axis.

Pass no arguments to return the current values without modifying th
```



```
df['TOEFL Score'].value_counts().head()
```

TOEFL Score	count
110	44
105	37
104	29
107	28
106	28

dtype: int64

```
c=df['TOEFL Score'].sum()
d=df['TOEFL Score'].count()
c/d
```

```
107.192
```

```
e=df['TOEFL Score'].sort_values(ascending=True).head()
f=df['TOEFL Score'].sort_values(ascending=True).tail()
e
```

	TOEFL Score
368	92
28	93
79	93
411	94
347	94

dtype: int64

f

	TOEFL Score
81	120
97	120
297	120
143	120
497	120

dtype: int64

Least Tofel Score-92

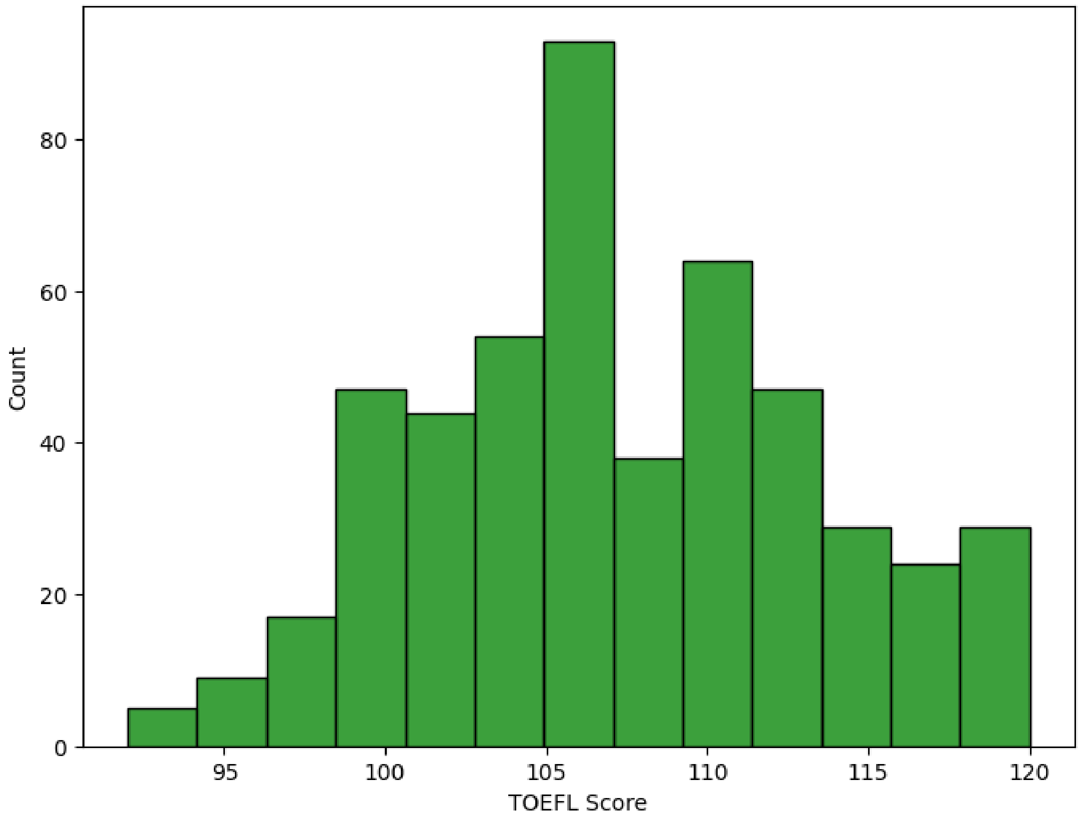
Top Tofel Score-120

AVg tofel score- 107.192

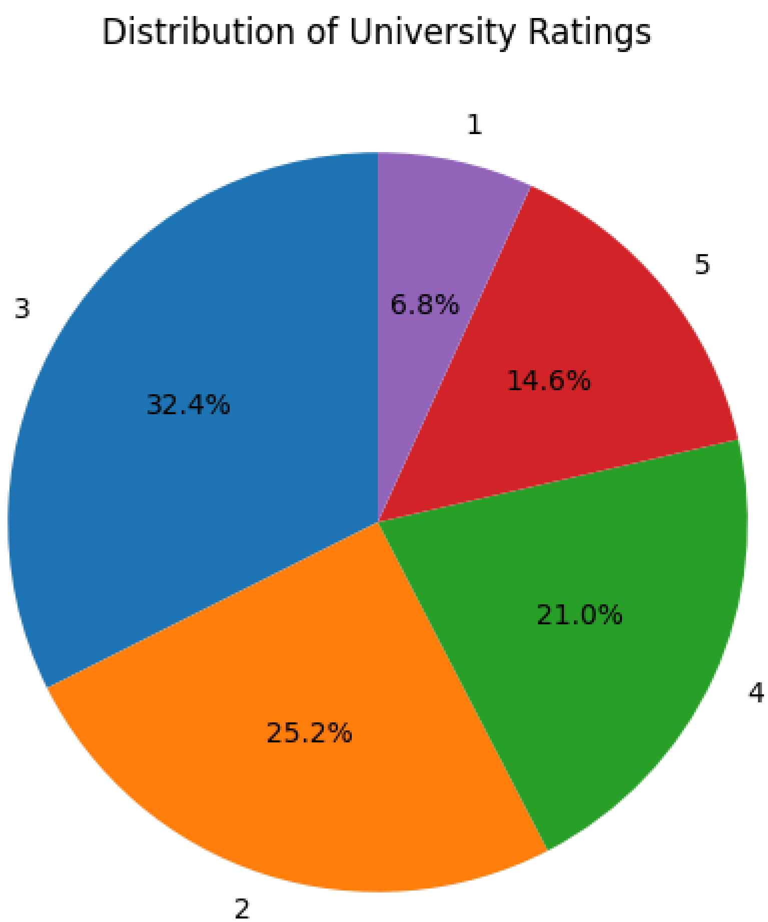
```
plt.figure(figsize=(8,6))
sns.histplot(x='TOEFL Score',data=df,color='Green')
plt.xticks
```

```
matplotlib.pyplot.xticks
def xticks(ticks: ArrayLike | None=None, labels: Sequence[str] |
None=None, *, minor: bool=False, **kwargs) -> tuple[list[Tick] |
np.ndarray, list[Text]]
```

```
/usr/local/lib/python3.11/dist-packages/matplotlib/pyplot.py
Get or set the current tick locations and labels of the x-axis.

Pass no arguments to return the current values without modifying th
```



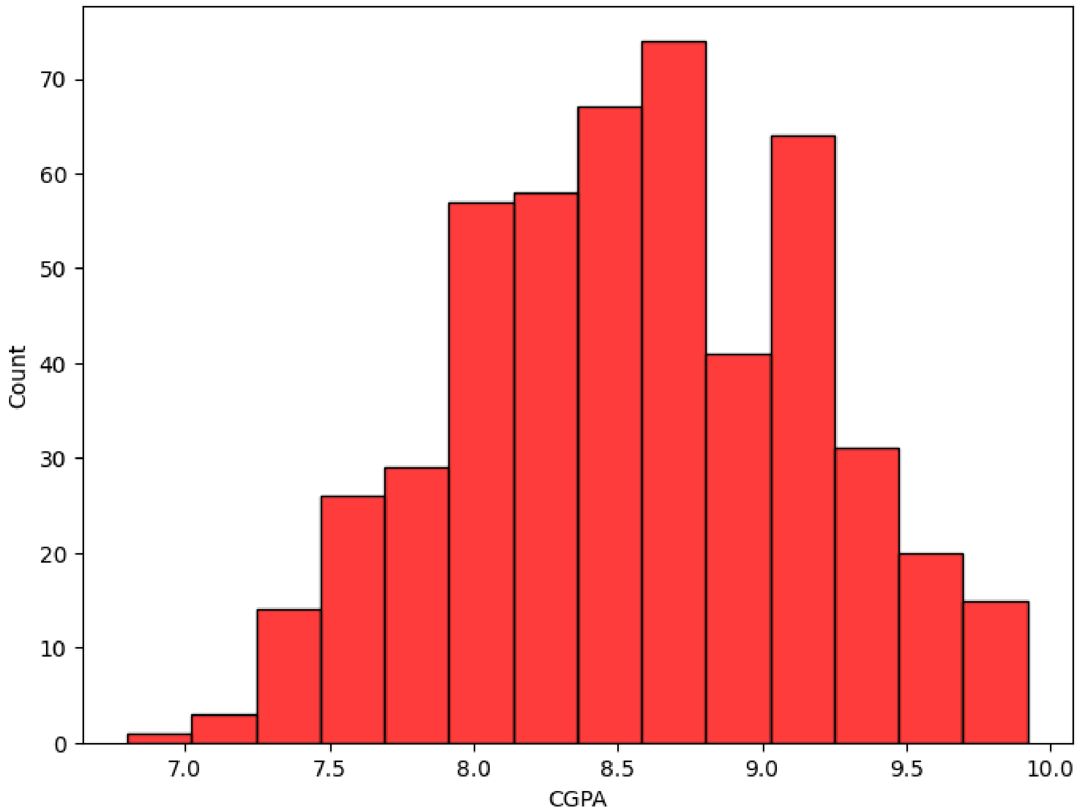
```
plt.figure(figsize=(8, 6))
rating_counts = df['University Rating'].value_counts()
plt.pie(rating_counts, labels=rating_counts.index, autopct='%1.1f%%', starta
plt.title('Distribution of University Ratings')
plt.show()
```



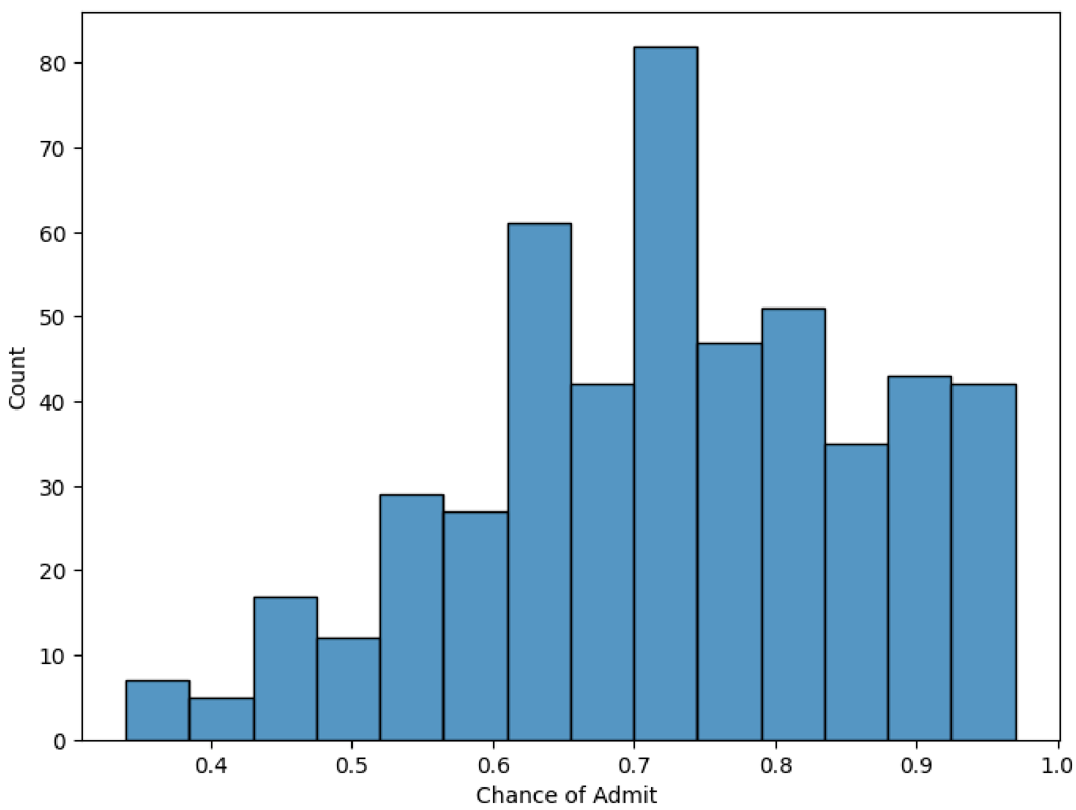
```
plt.figure(figsize=(8,6))
sns.histplot(x='CGPA',data=df,color='RED')
plt.xticks
```

```
matplotlib.pyplot.xticks
def xticks(ticks: ArrayLike | None=None, labels: Sequence[str] |
None=None, *, minor: bool=False, **kwargs) -> tuple[list[Tick] |
np.ndarray, list[Text]]
```

```
/usr/local/lib/python3.11/dist-packages/matplotlib/pyplot.py
Get or set the current tick locations and labels of the x-axis.

Pass no arguments to return the current values without modifying th
```



```
plt.figure(figsize=(8,6))
sns.histplot(x='Chance of Admit ', data=df)
plt.show()
```



INFERENCES FROM PLOTS

. A huge number of graduates score between 310 - 330 in GRE.

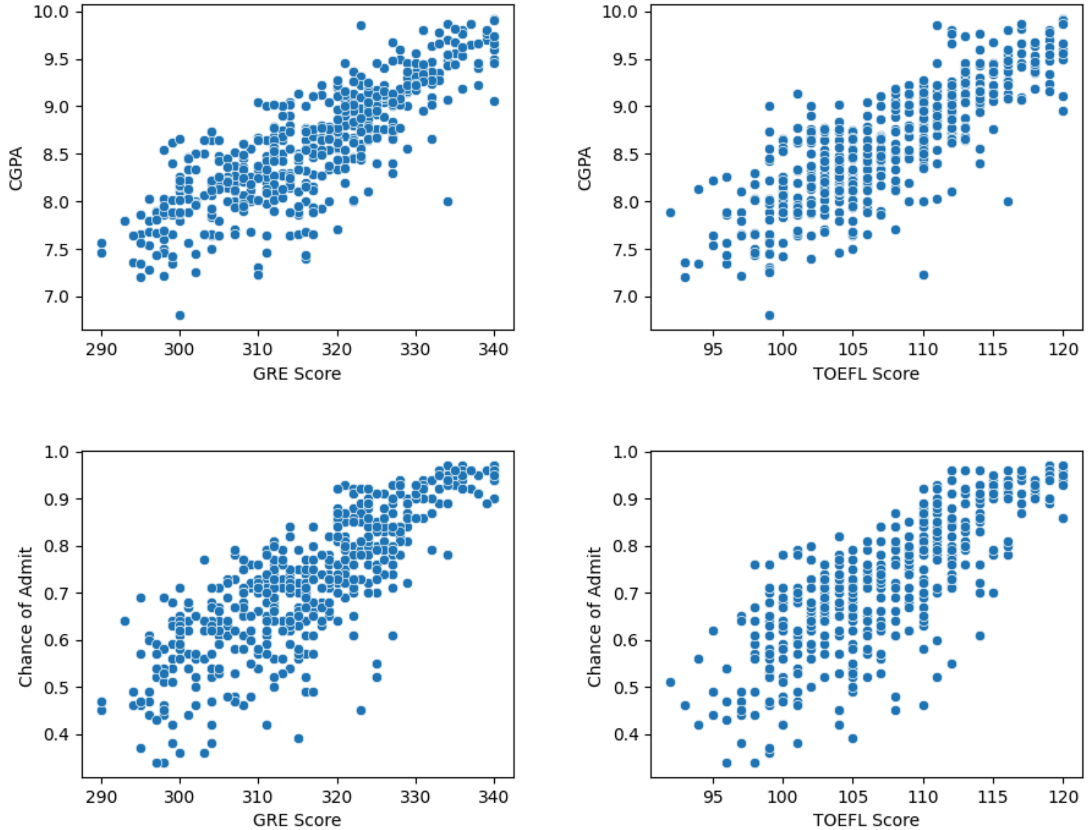
.The median of graduate score in TOEFL is around 107.

.Many universities rank b/w 2.0 - 3.5.

.Majority of the students have their cgpa between 8.0 - 9.3.

```
fig, axes = plt.subplots(2,2,figsize=(10,8))
fig.tight_layout(pad=5.0)

sns.scatterplot(data=df,x='GRE Score',y='CGPA',ax=axes[0,0])

sns.scatterplot(data=df,x='TOEFL Score',y='CGPA',ax=axes[0,1])

sns.scatterplot(data=df,x='GRE Score',y='Chance of Admit ',ax=axes[1,0])

sns.scatterplot(data=df,x='TOEFL Score',y='Chance of Admit ',ax=axes[1,1])
```

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

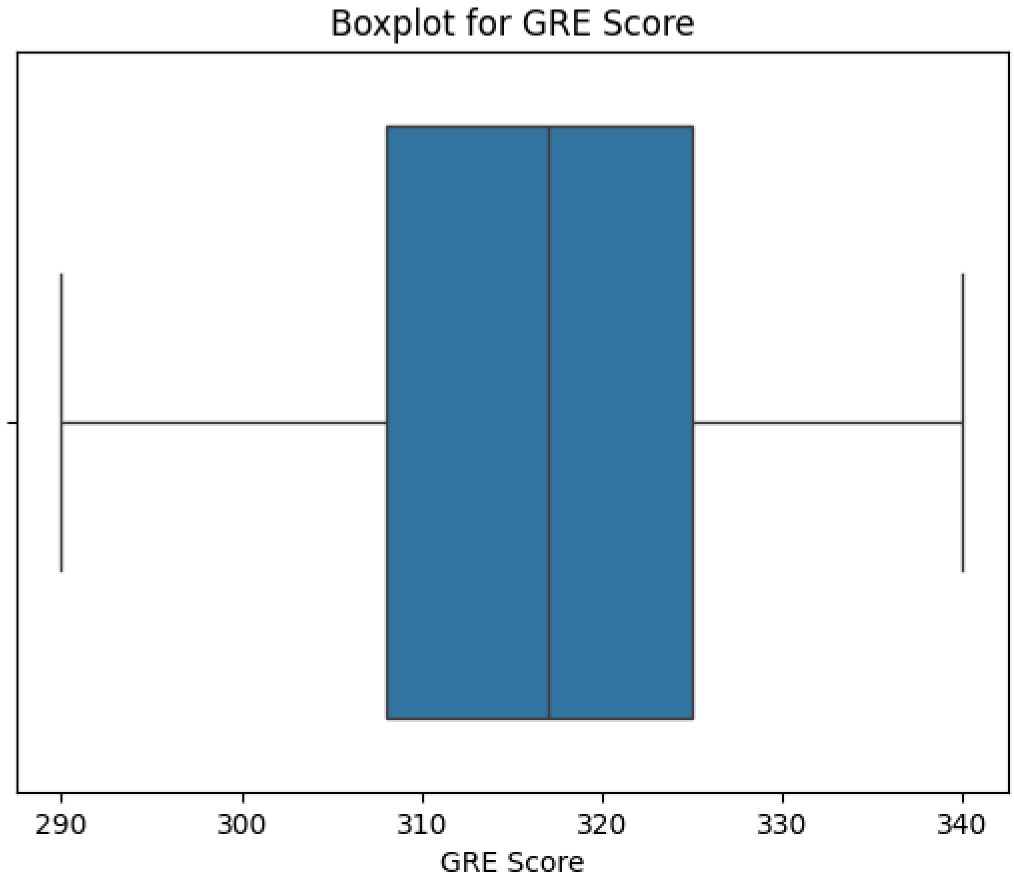


How can I create a plot in Colab?

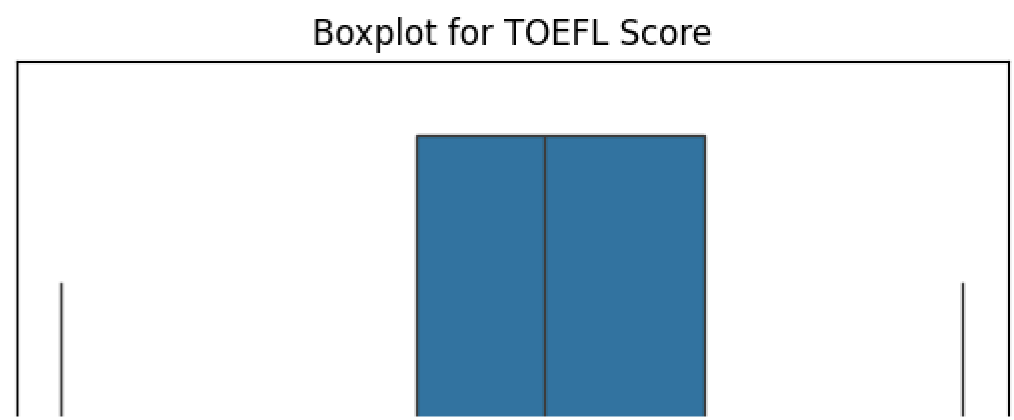
Show me a list of publicly available datasets

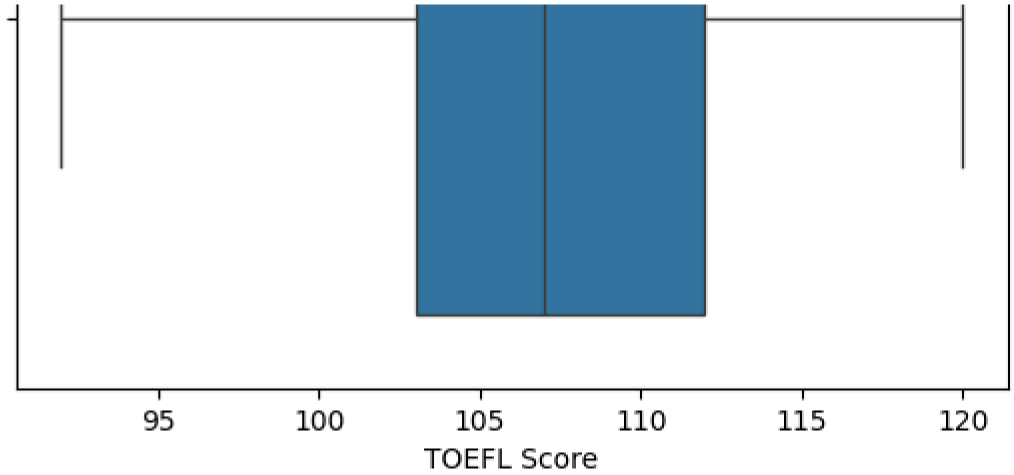
```
cols=['GRE Score','TOEFL Score','University Rating','CGPA','Chance of Admit



def visualize_boxplot(dataframe):
  for i in cols:

      plt.figure(figsize=(6,4))
      fig = plt.figure()
      fig.tight_layout(pad=5.0)
      plt.figure()
      sns.boxplot(data=dataframe, x = i)
      plt.title(f"Boxplot for {i}")
      plt.show()
visualize_boxplot(df)
```

```
<Figure size 600x400 with 0 Axes>
<Figure size 640x480 with 0 Axes>
```

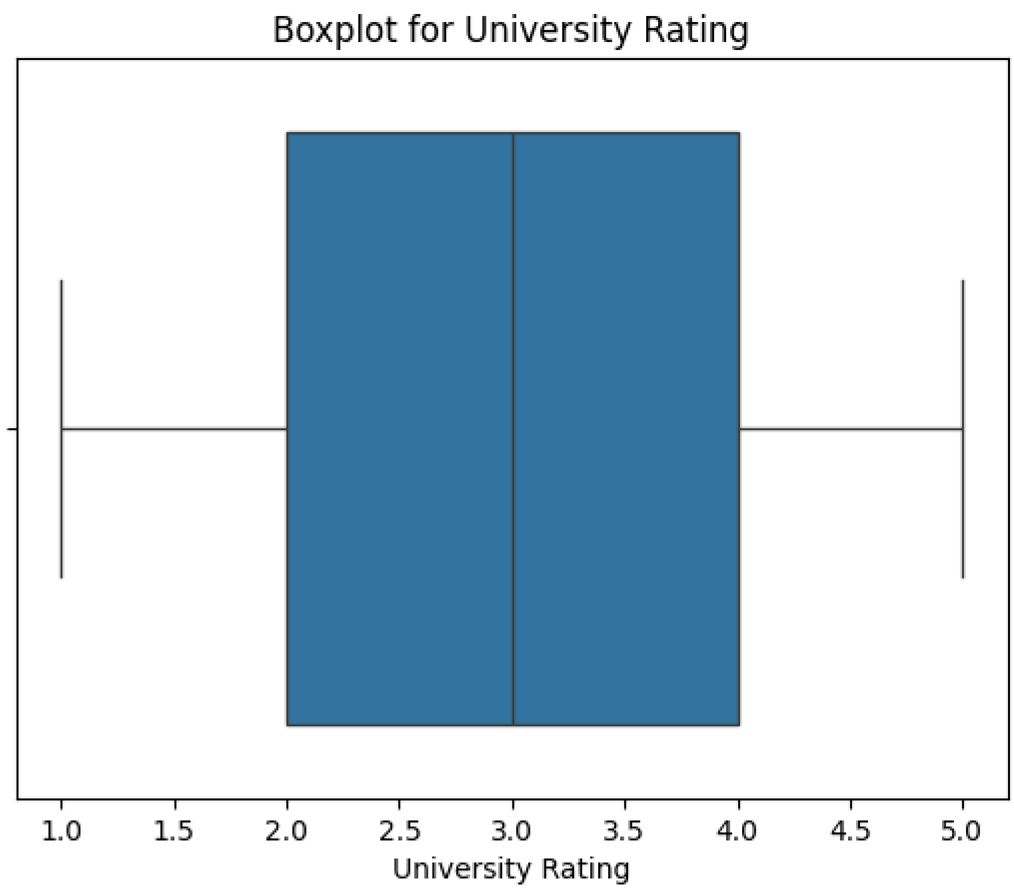



```
<Figure size 600x400 with 0 Axes>
<Figure size 640x480 with 0 Axes>
```

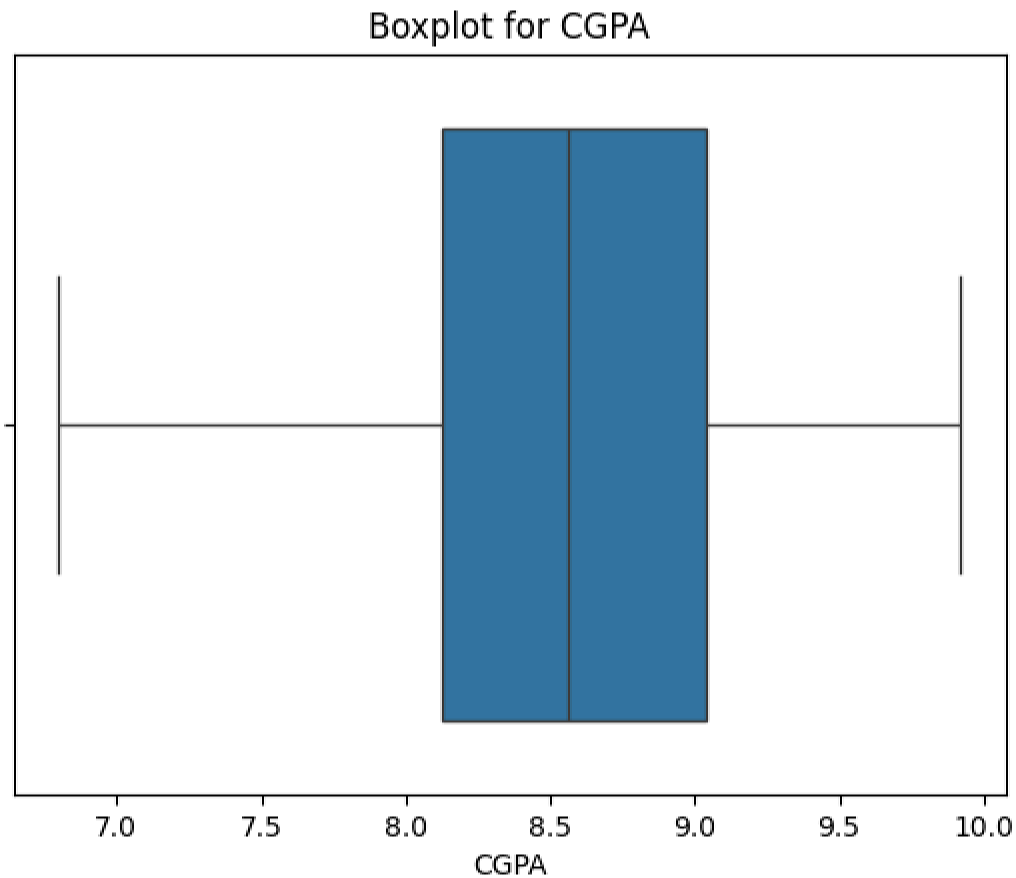




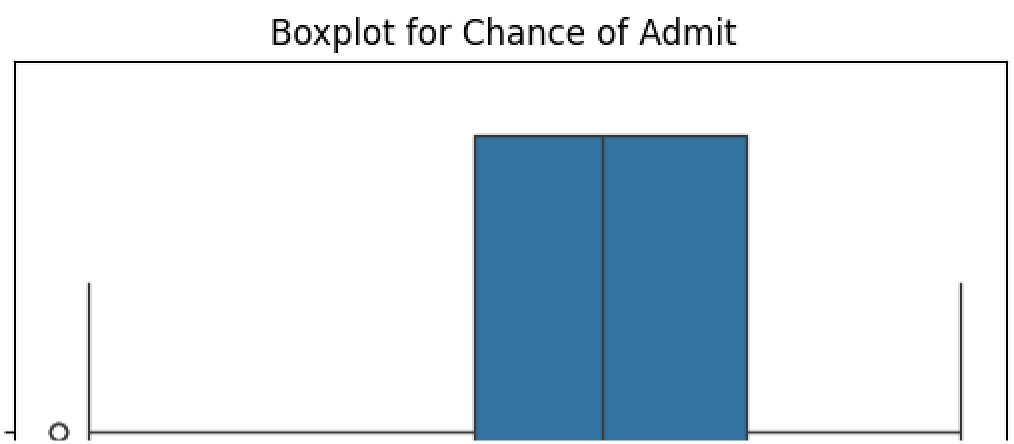
<Figure size 600x400 with 0 Axes>
<Figure size 640x480 with 0 Axes>

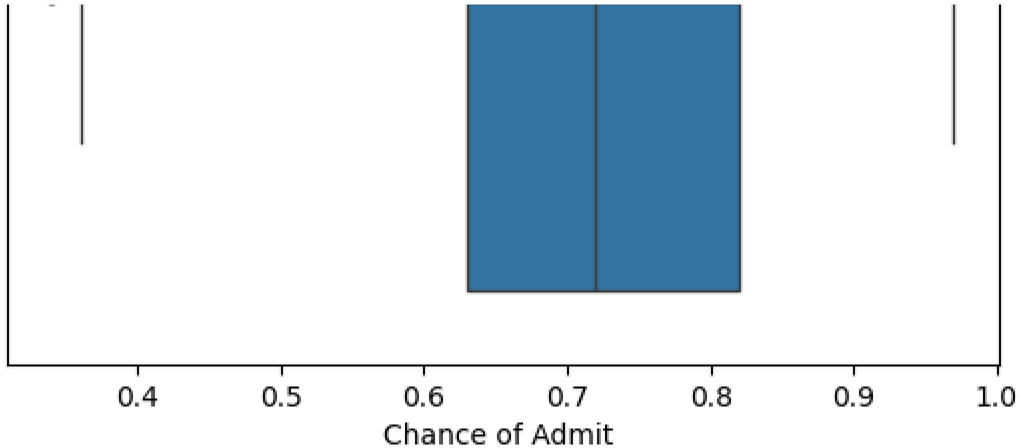


```
<Figure size 600x400 with 0 Axes>
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 600x400 with 0 Axes>
<Figure size 640x480 with 0 Axes>
```




0.4
0.5
0.6
0.7
0.8
0.9
1.0
Chance of Admit

Feature Engineering

```
x=df.drop(columns=['Chance of Admit '])
```

```
x # prints all other columns except chance of admit
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	337	118	4	4.5	4.5	9.65	1
1	324	107	4	4.0	4.5	8.87	1
2	316	104	3	3.0	3.5	8.00	1
3	322	110	3	3.5	2.5	8.67	1
4	314	103	2	2.0	3.0	8.21	0
...	...	...	...	...	...	...	...
495	332	108	5	4.5	4.0	9.02	1
496	337	117	5	5.0	5.0	9.87	1
497	330	120	5	4.5	5.0	9.56	1
498	312	103	4	4.0	5.0	8.43	0
499	327	113	4	4.5	4.5	9.04	0

500 rows × 7 columns

Next steps: Generate code with x | View recommended plots | New interactive sheet

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

Data preparation for modelling

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state
```

```
from sklearn.preprocessing import StandardScaler # scaling the dataset
scaler=StandardScaler() #The StandardScaler is used to standardize features b
x_train_scaled=scaler.fit_transform(x_train) # Calculates the mean and standa
#Applies the standardization using the calculated mean and standard deviation
x_test_scaled=scaler.transform(x_test)#We avoid using fit_transform on x_test
#ensuring the model doesn't learn anything from the test set during training.
```

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train_scaled,y_train)
```

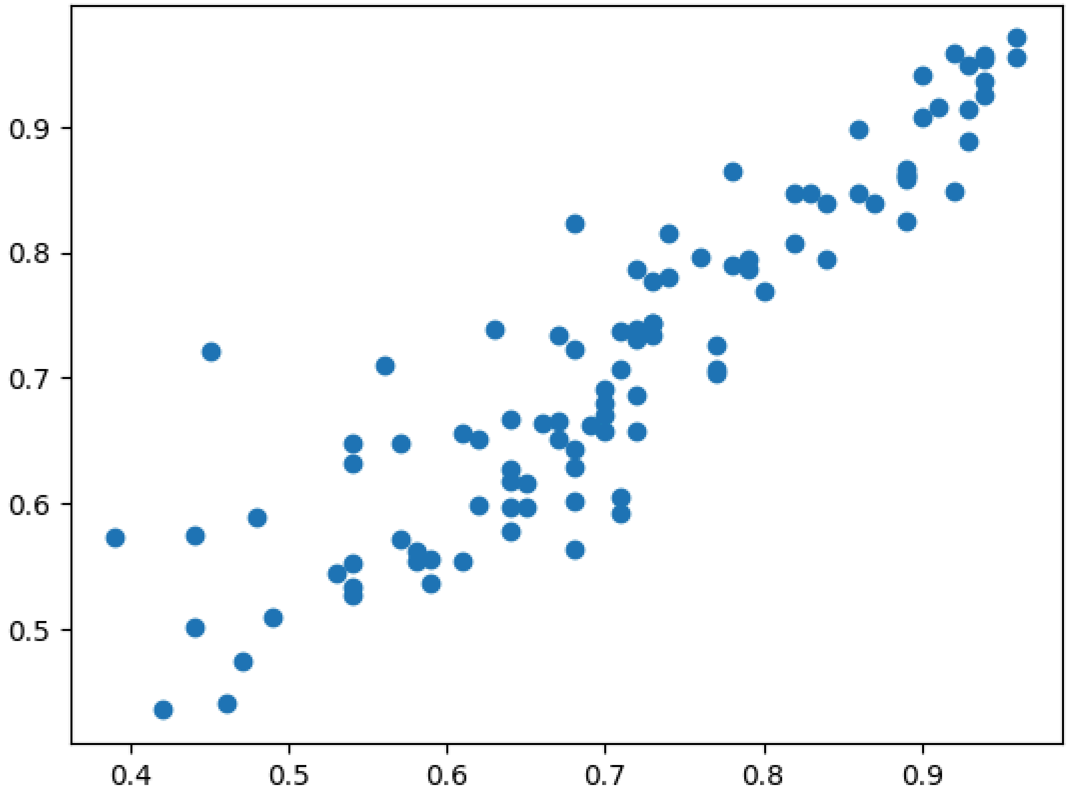
```
▾ LinearRegression
LinearRegression()
```

```
y_train_pred=model.predict(x_train_scaled)
y_pred = model.predict(x_test_scaled)
print(x_test.shape,y_test.shape)
```

```
(100, 7) (100,)
```

```
plt.scatter(y_test,y_pred)
```

```
<matplotlib.collections.PathCollection at 0x790e5ebe3b50>
```



Model building

1. Build the Linear Regression model and comment on the model statistics
2. Display model coefficients with column names
3. Try out Ridge and Lasso regression

```
coefficients = model.coef_
coef_df = pd.DataFrame({'Features':x_train.columns, 'Coefficients':coefficien
coef_df['Intercept'] = model.intercept_
coef_df
```

	Features	Coefficients	Intercept
0	GRE Score	0.026671	0.724175
1	TOEFL Score	0.018226	0.724175
2	University Rating	0.002940	0.724175
3	SOP	0.001788	0.724175
4	LOR	0.015866	0.724175
5	CGPA	0.067581	0.724175
6	Research	0.011940	0.724175

Next steps: Generate code with coef_df View recommended plots New interactive

```
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
ridge_model = Ridge(alpha=0.10)
ridge_model.fit(x_train_scaled, y_train)
```

```
▾ Ridge
Ridge(alpha=0.1)
```

```
ridge_train_pred = ridge_model.predict(x_train_scaled)
ridge_y_pred = ridge_model.predict(x_test_scaled)
```

```
# Lasso Regression
lasso_model = Lasso(alpha=0.1)
```

```
lasso_model.fit(x_train_scaled,y_train)
```

```
▾ Lasso
Lasso(alpha=0.1)
```

```
lasso_train_pred = lasso_model.predict(x_train_scaled)
lasso_y_pred = lasso_model.predict(x_test_scaled)
```

OLS

```
# Checking which features have a high impact on the target variable.

# constant term represents the expected value of the dependent variable wher
# all the independent variables are zero.
new_x_train = sm.add_constant(x_train)
ols_model = sm.OLS(y_train, new_x_train)
result = ols_model.fit()

print(result.summary())
```

```
                                OLS Regression Results
=======================================================================
Dep. Variable:       Chance of Admit    R-squared:
Model:                           OLS    Adj. R-squared:
Method:                Least Squares    F-statistic:
Date:               Fri, 14 Feb 2025    Prob (F-statistic):          3.4
Time:                       12:56:41    Log-Likelihood:
No. Observations:                400    AIC:
Df Residuals:                    392    BIC:
Df Model:                          7
Covariance Type:           nonrobust
=======================================================================
                      coef    std err          t      P>|t|      [0.02
-----------------------------------------------------------------------
const              -1.4214      0.123    -11.549      0.000      -1.66
GRE Score           0.0024      0.001      4.196      0.000       0.00
TOEFL Score         0.0030      0.001      3.174      0.002       0.00
```

```
University Rating     0.0026      0.004      0.611      0.541      -0.0(
SOP                   0.0018      0.005      0.357      0.721      -0.0(
LOR                   0.0172      0.005      3.761      0.000       0.0(
CGPA                  0.1125      0.011     10.444      0.000       0.0(
Research              0.0240      0.007      3.231      0.001       0.0(
==============================================================================
Omnibus:                       86.232   Durbin-Watson:
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              1
Skew:                          -1.107   Prob(JB):                      5
Kurtosis:                       5.551   Cond. No.                      1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is c
[2] The condition number is large, 1.37e+04. This might indicate that th
strong multicollinearity or other numerical problems.
```

From the above results it is clear that GRE Score, TOEFL Score, LOR, CGPA, and Research significantly impact the target variable whereas University Ranking and SOP don't.

finding MSE, MAE, R2 SCORE, ADJ R2 SCORE

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_scor
import numpy as np

# TRAINING MODEL
n = len(y_train)
p = len(model.coef_) if len(model.coef_.shape) == 1 else len(model.coef_[0])
mse = np.round(mean_squared_error(y_true=y_train, y_pred=y_train_pred), 2)
mae = np.round(mean_absolute_error(y_true=y_train, y_pred=y_train_pred), 2)
r2score = np.round(r2_score(y_true=y_train, y_pred=y_train_pred), 2)
adj_r = np.round(((1 - r2score) * (n - 1)) / (n - p - 1), 2)

print(f"TRAINING MODEL\nMSE: {mse}\nMAE: {mae}\nr2 score: {r2score}\nadjuste
print('-' * 30)
```

```
# TESTING MODEL
n = len(y_test)
p = len(model.coef_) if len(model.coef_.shape) == 1 else len(model.coef_[0])
mse = np.round(mean_squared_error(y_true=y_test, y_pred=y_pred), 2)
mae = np.round(mean_absolute_error(y_true=y_test, y_pred=y_pred), 2)
r2score = np.round(r2_score(y_true=y_test, y_pred=y_pred), 2)
adj_r = np.round(((1 - r2score) * (n - 1)) / (n - p - 1), 2)

print(f"TESTING MODEL\nMSE: {mse}\nMAE: {mae}\nr2 score: {r2score}\nadjustec
```

```
TRAINING MODEL
MSE: 0.0
MAE: 0.04
r2 score: 0.82
adjusted r: 0.18
------------------------------
TESTING MODEL
MSE: 0.0
MAE: 0.04
r2 score: 0.82
adjusted r: 0.19
```

As there is no difference between the results of the training and testing model, conlusion can be drawn that there is no overfitting.

checking multicollinearity with VIF

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif['Feature'] = x_train.columns
vif['VIF'] = [variance_inflation_factor(x_train_scaled,i) for i in range(x_t
vif = vif.sort_values(by="VIF",ascending = False)
```

```
vif
```

	Feature	VIF
5	CGPA	4.654540
0	GRE Score	4.489983
1	TOEFL Score	3.664298
3	SOP	2.785764
2	University Rating	2.572110
4	LOR	1.977698
6	Research	1.518065

Next steps: Generate code with vif View recommended plots New interactive shee

As there is no VIF>5 we can say that there is no multicollinearity b/w any two features.

calculate mean of residuals

```
residuals = y_test - y_pred
```

```
residuals
```

	Chance of Admit
361	0.015425
73	0.044819
374	-0.182660
155	0.062630
104	-0.075883
...	...
347	-0.015631
86	0.033357

```
print(residuals)
print('-'*55)
print(residuals.mean())
```

```
361    0.015425
73     0.044819
374   -0.182660
155    0.062630
104   -0.075883
         ...
347   -0.015631
86     0.033357
75    -0.065988
438   -0.064694
15    -0.108657
Name: Chance of Admit , Length: 100, dtype: float64
-------------------------------------------------------
-0.005453623717661251
```

```
plt.figure(figsize=(12,5))
sns.histplot(residuals, kde=True)
```

```
<Axes: xlabel='Chance of Admit ', ylabel='Count'>
```