Evaluation Criteria (100 Points):

1. Define Problem Statement and perform Exploratory Data Analysis (10 points
    - Definition of problem (as per given problem statement with additional views)
    - Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required) , missing value detection, statistical summary.
    - Univariate Analysis (distribution plots of all the continuous variable(s) barplots/countplots of all the categorical variables)
    - Bivariate Analysis (Relationships between important variables such as workday and count, season and count, weather and count.
    - Illustrate the insights based on EDA
        - Comments on range of attributes, outliers of various attribute
        - Comments on the distribution of the variables and relationship between them
        - Comments for each univariate and bivariate plots
2. Data Preprocessing (10 Points)
    - Duplicate value check
    - Missing value treatment
    - Outlier treatment
    - Feature engineering
    - Data preparation for modeling
3. Model building (10 Points)
    - Build the Linear Regression model and comment on the model statistics
    - Display model coefficients with column names
4. Testing the assumptions of the linear regression model (50 Points)
    - Multicollinearity check by VIF score (variables are dropped one-by-one till none has VIF>5) (10 Points)
    - The mean of residuals is nearly zero (10 Points)
    - Linearity of variables (no pattern in the residual plot) (10 Points)
    - Test for Homoscedasticity (10 Points)
    - Normality of residuals (almost bell-shaped curve in residuals distribution, points in QQ plot are almost all on the line) (10 Points)
5. Model performance evaluation (10 Points)
    - Metrics checked - MAE, RMSE, R2, Adj R2

- Train and test performances are checked
- Comments on the performance measures and if there is any need to improve the model or not

6. Actionable Insights & Recommendations (10 Points)
    - Comments on significance of predictor variables
    - Comments on additional data sources for model improvement, model implementation in real world, potential business benefits from improving the model (These are key to differentiating a good and an excellent solution)

Problem Statement:

Jamboree needs to undestand what factors are important in graduate admissions and how these factors are interrelated among themselves and also help predict one's chances of admission given the rest of the variables.

```
import pandas as pd
pd.options.plotting.backend = "plotly"

import numpy as np
import missingno as msno
import pandas_profiling as pf
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from scipy import stats
from sklearn import linear_model
%matplotlib inline
```

Basic Metrics

Size, shape and data types

```
df = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/839/orig
display(df.head())
print()
print(f"Rows/columns dimension - {df.shape}")
print()
```

	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

```
Rows/columns dimension : (500, 9)
```

```
df=df.drop(['Serial No.'], axis=1)
df.isnull().sum()/len(df)*100
```

```
GRE Score            0.0
TOEFL Score          0.0
University Rating    0.0
SOP                  0.0
LOR                  0.0
CGPA                 0.0
Research             0.0
Chance of Admit      0.0
dtype: float64
```

We do not have any null values in our data set which makes it easier for us to conduct our data analysis.

Data Wrangling

```
#Convert columns to categorical format
df["Research"]=df["Research"].astype("category")
```

Summary

```
df.describe(include="category").T
```

	count	unique	top	freq
Research	500	2	1	280

```
df.describe()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.00000
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.72174
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.14114
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.34000
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.63000
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	0.72000
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	0.82000
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	0.97000

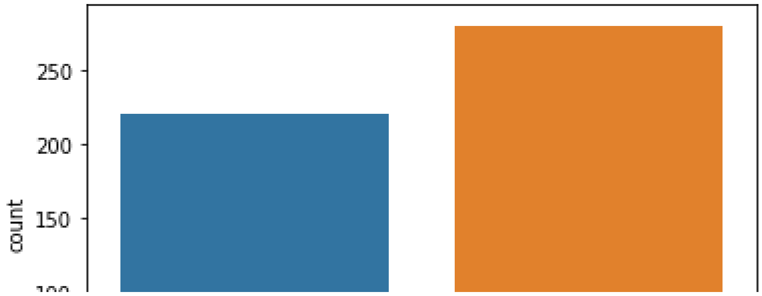
EDA

Univariate Analysis

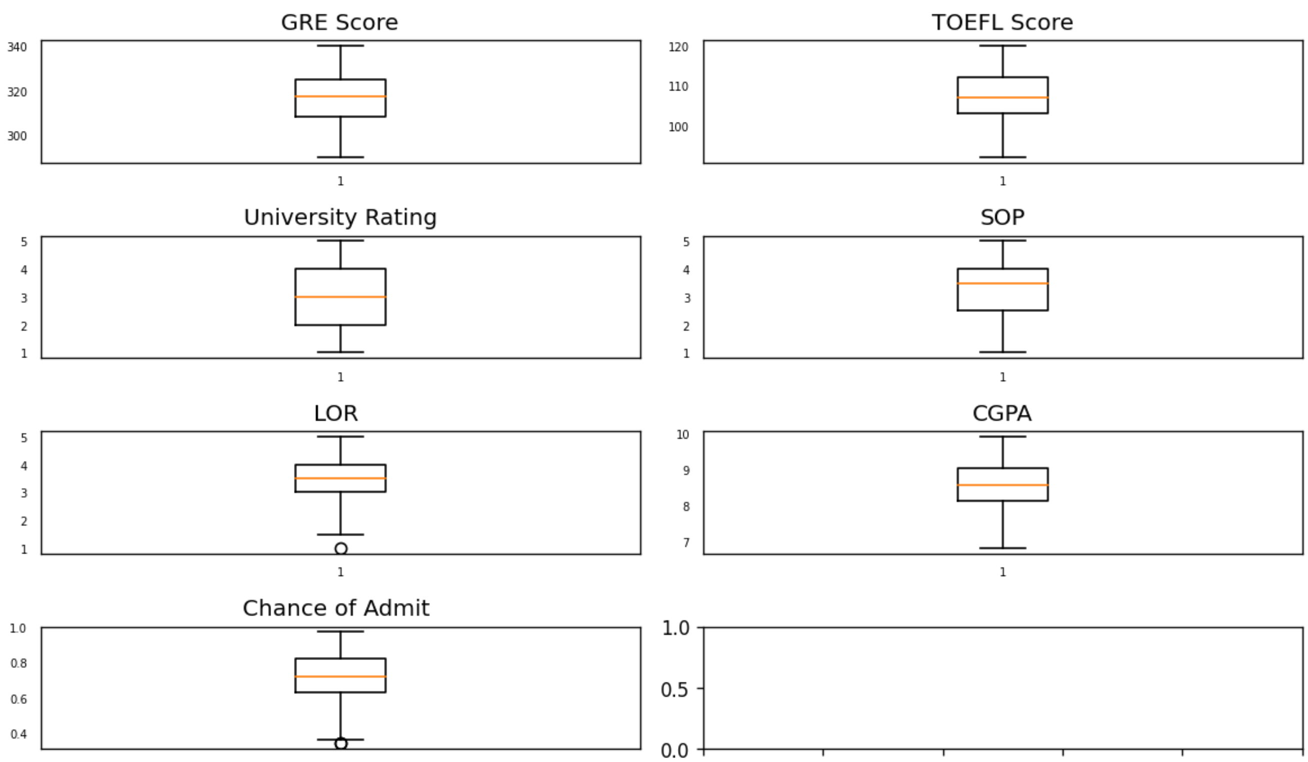
```
def plot_box_plot(df,nrows, ncols, maxColumns=0, plotkind='boxplot', figsize=(10,6)):
  fig, axes = plt.subplots(nrows, ncols, dpi=120, figsize=figsize)
  for i, ax in enumerate(axes.flatten()):
    if maxColumns != 0 and i+1 == maxColumns:
      break
    data = df[df.columns[i]]
    if plotkind is 'boxplot':
      ax.boxplot(data)
    elif plotkind is 'kdeplot':
      sns.kdeplot(data,ax=ax)
    # Decorations
    ax.set_title(df.columns[i])
    ax.xaxis.set_ticks_position('none')
    ax.yaxis.set_ticks_position('none')
    ax.tick_params(labelsize=6)
  plt.tight_layout();
```

```
sns.countplot(df['Research'])
```

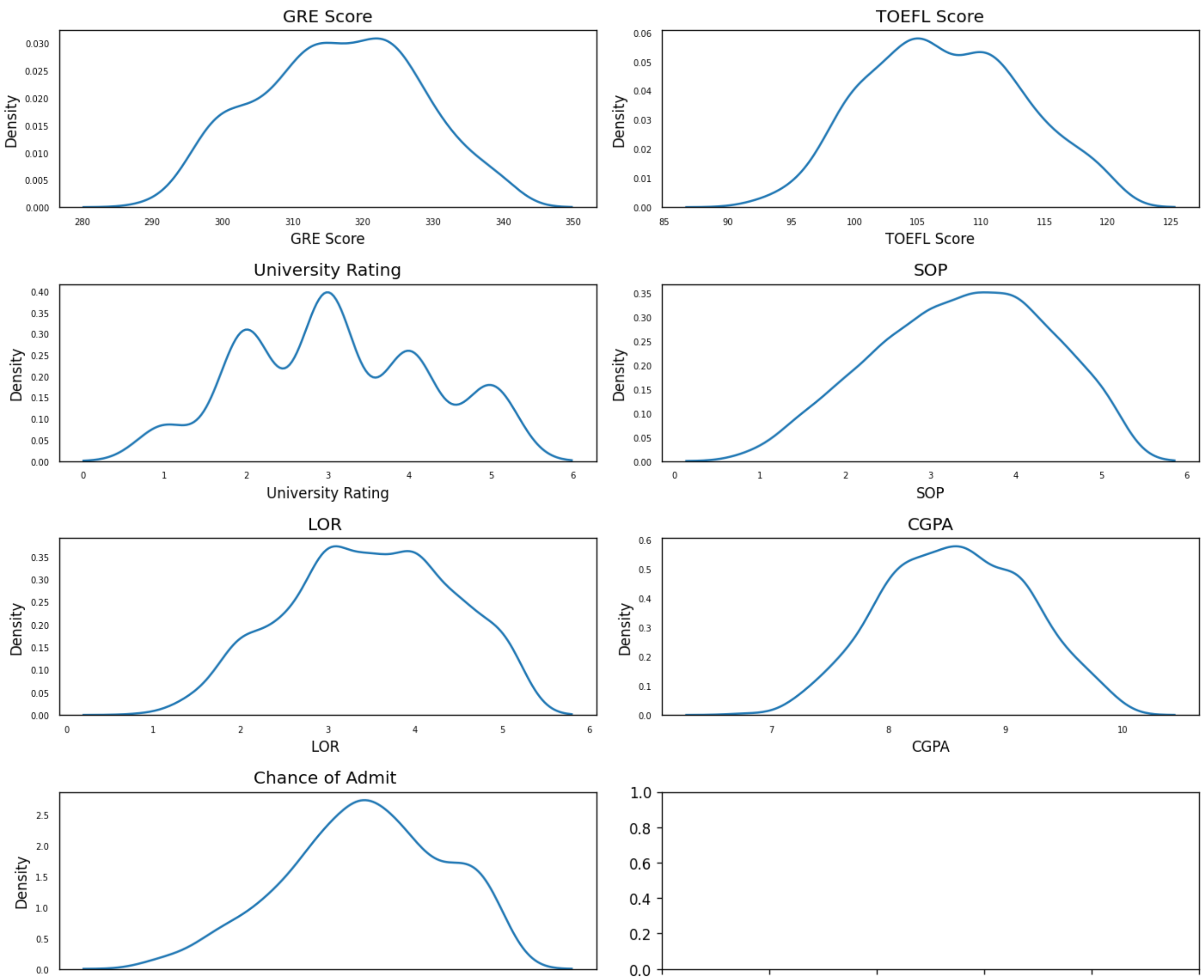
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f40ed423fd0>
```




```
df_temp = df.drop(['Research'], axis=1)
plot_box_plot(df_temp, 4, 2, 8)
```



```
plot_box_plot(df_temp,4, 2, 8, 'kdeplot', (12,10))
```

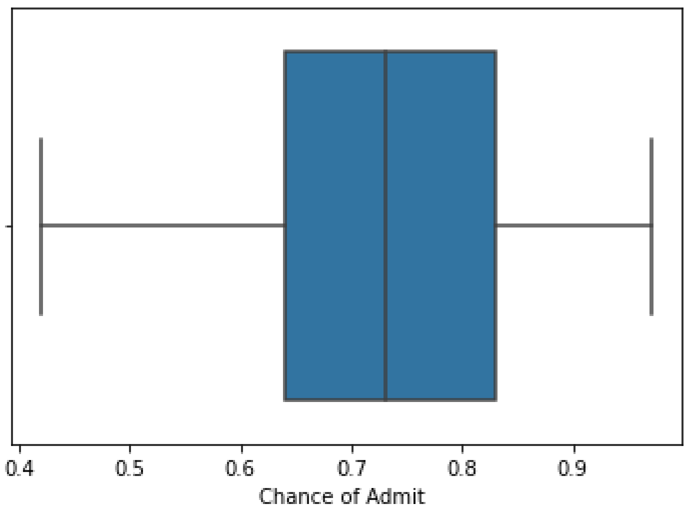


outlier Treatment

Target variable has outliers which will impact in linear regression

```
df = df[(df['Chance of Admit '] >= 0.4)]
sns.boxplot(x=df['Chance of Admit '])
```

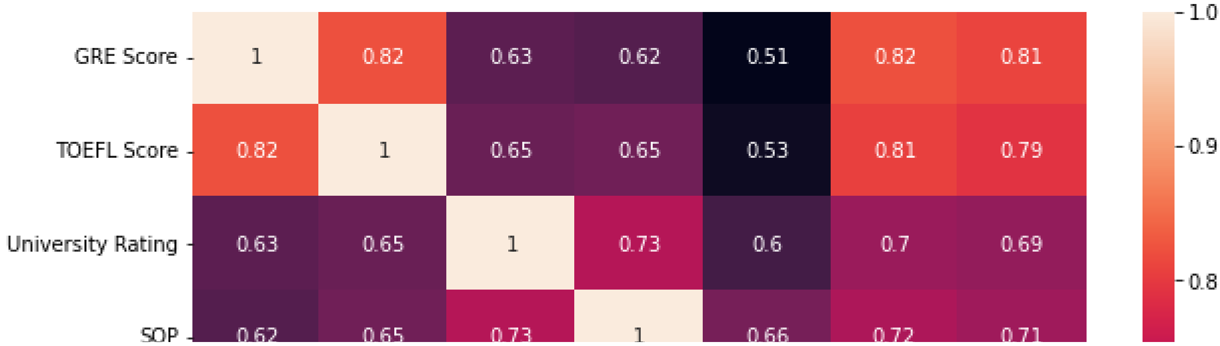
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f40e80f1c50>
```



Mutli variate Analysis

```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f40e6842b90>
```

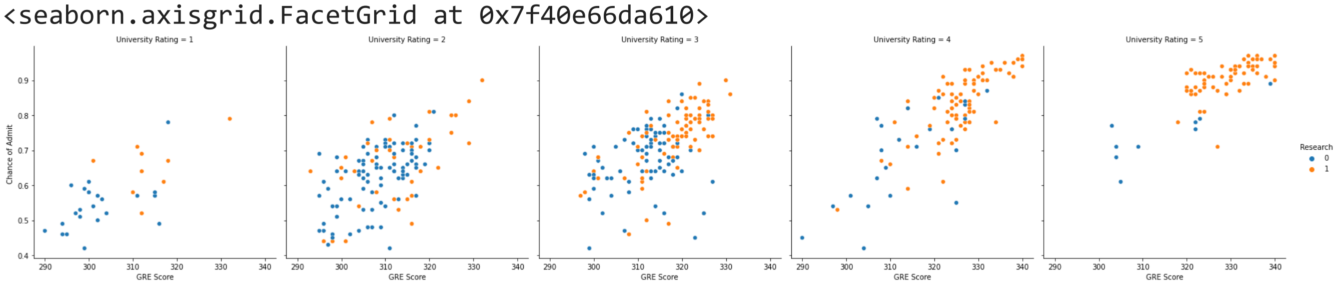



```
sns.pairplot(df, hue='Research')
```

```
<seaborn.axisgrid.PairGrid at 0x7f40e6842dd0>
```

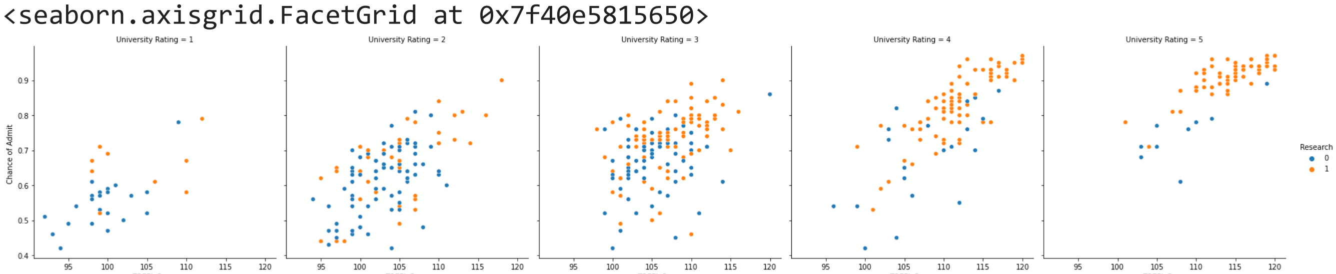
```
sns.relplot(
    data=df,
    x="GRE Score", y='Chance of Admit ', col="University Rating",hue="Research",
)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f40e66da610>
```



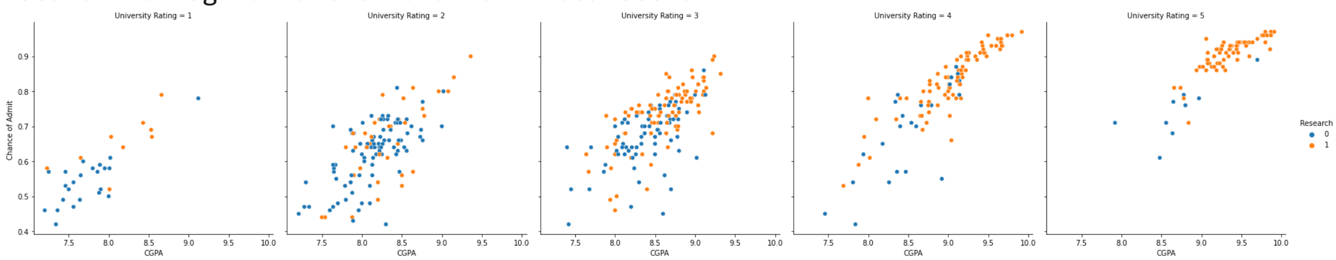
```
sns.relplot(
    data=df,
    x="TOEFL Score", y='Chance of Admit ', col="University Rating",hue="Research",
)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f40e5815650>
```



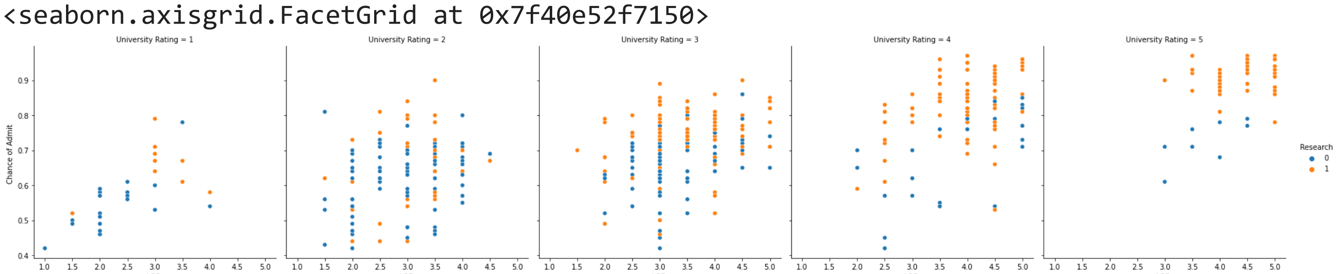
```
sns.relplot(
    data=df,
    x="CGPA", y='Chance of Admit ', col="University Rating",hue="Research",
)
```

<seaborn.axisgrid.FacetGrid at 0x7f40e548ee10>



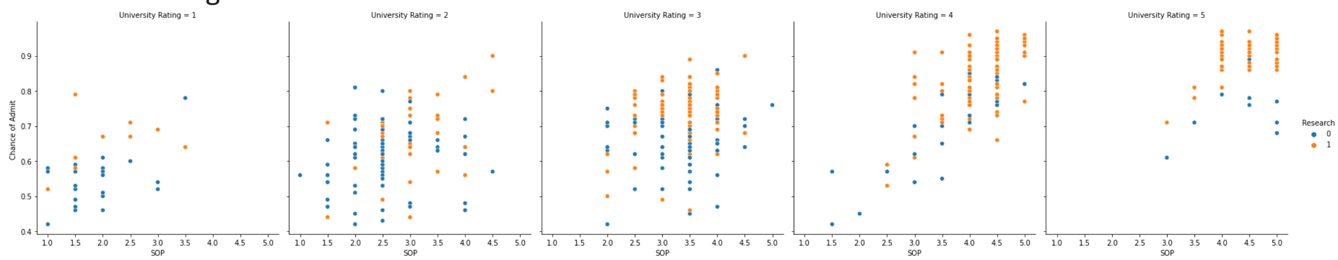

```
sns.relplot(
    data=df,
    x="LOR ", y='Chance of Admit ', col="University Rating",hue="Research",
)
```

<seaborn.axisgrid.FacetGrid at 0x7f40e52f7150>



```
sns.relplot(
    data=df,
    x="SOP", y='Chance of Admit ', col="University Rating",hue="Research",
)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f40e50b6310>
```



Insights

- Those who did research has high chance of admit.
- From visualization and correlation factor CGPA,TOEFL and GRE score looks like high correlation.
- SOP and LOR has high correlation which is visually visible and also correlation value from heatmap also support the statement.
- For university rating 5, those who has GRE score more than 300 and TOFEL score more than 100 has high chance to get admit.

Data Preprocessing

```
#Duplicate valuce check
print("Before removing duplicate dataframe size - "+str(df.shape[0]))
bool_series = df.duplicated(keep='first')
df=df[~bool_series]
print("After removing duplicate dataframe size - "+str(df.shape[0]))
```

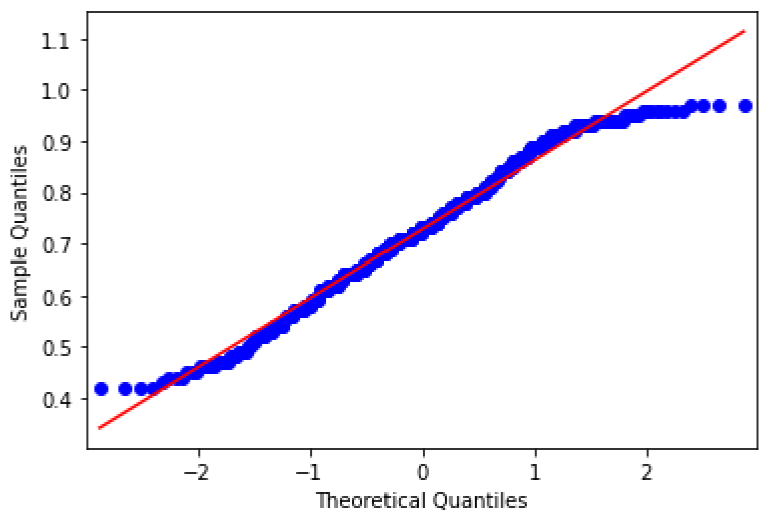
```
Before removing duplicate dataframe size - 492
After removing duplicate dataframe size - 492
```

```
from scipy.stats import shapiro
from statsmodels.api import qqplot
def check_normality_test(x):
  _,p = shapiro(x)
  if p < 0.05:
    print("Target variable is not gausian distribution")
  else:
    print("Target variable is gausian distribution")
  print()
  fig = qqplot(x,line='s')
  plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
  import pandas.util.testing as tm
```

```
check_normality_test(df['Chance of Admit '])
```

```
Target variable is not gausian distribution
```



Insights

- Looks like output variable doesnt have gaussion distribution

```
#Model preparation
X = df[df.columns.drop('Chance of Admit ')]
Y = df["Chance of Admit "]
```

```
#Normalization
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = pd.DataFrame(sc.fit_transform(X[X.columns]),columns=X.columns)
```

Model building & Testing Assumptions

```
import statsmodels.api as sm
from statsmodels.stats.diagnostic import het_breuschpagan
from statsmodels.stats.stattools import durbin_watson
def get_stat_model_summary(X,Y):
  X_sm = sm.add_constant(X)
  sm_model = sm.OLS(list(Y), X_sm).fit()
  _,p,_,_ = het_breuschpagan(sm_model.resid, sm_model.model.exog)
  print("----------------------------------")
  print("| residual mean of model:"+ str(round(sm_model.resid.mean(),2)) +"     |")
  print("| Independent error value:"+ str(round(durbin_watson(sm_model.resid),2)) +"     |" )
  if p < 0.05:
    print("| Homoscedasticity is present.    |")
  else:
    print("| Homoscedasticity is not present.|")
  print("----------------------------------")

  print()

  print(sm_model.summary())
  return sm_model
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
def get_stat_vif(X,Y):
  vif = pd.DataFrame()
  X_t = X
  vif['Features'] = X_t.columns
  vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1])]
  vif['VIF'] = round(vif['VIF'], 2)
  vif = vif.sort_values(by = "VIF", ascending = False)
  print(vif)
```

```
from sklearn.linear_model import LinearRegression
def get_model(X,Y):
  model = LinearRegression()
  model.fit(X,Y)
  print("model coefficient: "+ str(model.coef_))
  print("model intercept: "+ str(model.intercept_))
  print("model score: "+ str(round(model.score(X,Y),3)))
  print("Adjusted R-squared:", round(1 - (1-model.score(X, Y))*(len(Y)-1)/(len(Y)-X.shape[1]-
  return model
```

```
sm_model = get_stat_model_summary(X,Y)
```

```
 ---------------------------------
| residual mean of model:-0.0     |
| Independent error value:0.8     |
| Homoscedasticity is present.    |
 ---------------------------------

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.825
Model:                            OLS   Adj. R-squared:                  0.822
Method:                 Least Squares   F-statistic:                     325.6
Date:                Wed, 27 Apr 2022   Prob (F-statistic):         1.44e-178
Time:                        07:09:35   Log-Likelihood:                 717.37
No. Observations:                 492   AIC:                            -1419.
Df Residuals:                     484   BIC:                            -1385.
Df Model:                           7
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 0.7275      0.003    284.281      0.000       0.723       0.733
GRE Score             0.0212      0.005      3.970      0.000       0.011       0.032
TOEFL Score           0.0157      0.005      3.142      0.002       0.006       0.026
University Rating     0.0067      0.004      1.634      0.103      -0.001       0.015
SOP                   0.0058      0.004      1.320      0.187      -0.003       0.014
LOR                   0.0141      0.004      3.896      0.000       0.007       0.021
CGPA                  0.0651      0.006     11.699      0.000       0.054       0.076
Research              0.0111      0.003      3.568      0.000       0.005       0.017
==============================================================================
Omnibus:                      122.512   Durbin-Watson:                   0.804
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              323.659
Skew:                          -1.220   Prob(JB):                     5.23e-71
Kurtosis:                       6.136   Cond. No.                         5.60
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specif
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:117: FutureWarning:
  x = pd.concat(x[::order], 1)
```

```
get_stat_vif(X,Y)
```

```
            Features   VIF
5               CGPA  4.73
0          GRE Score  4.36
1        TOEFL Score  3.82
3                SOP  2.94
2  University Rating  2.60
4                LOR  1.99
6           Research  1.49
```

```
model_1 = get_model(X,Y)
```

```
model coefficient: [0.02121273 0.01571731 0.00674277 0.00578991 0.01407165 0.06514653
 0.01113415]
model intercept: 0.7275406504065041
model score: 0.825
Adjusted R-squared: 0.822
```

Insights based on stats models

- Residual mean of model is zero and also Homoscedasticity is also there.
- Independent error is lesser than 2 which proves statistically positive correlation.
- Pvalue of University Rating and sop is greater than 0.05 which proves statiscally which has high correlation.
- From visualization, correlation factor TOEFL Score almost linear relationship with GRE Score and also GRE Score close to vif value of 5. Based on Domain knowledge TOEFL Score is best choice to drop from the data.

```
X_1 = X.drop(['University Rating','SOP','TOEFL Score','Research'], axis=1)
sm_model = get_stat_model_summary(X_1,Y)
```

```
----------------------------------
| residual mean of model:-0.0     |
| Independent error value:0.89     |
| Homoscedasticity is present.    |
----------------------------------

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.813
Model:                            OLS   Adj. R-squared:                  0.812
Method:                 Least Squares   F-statistic:                     706.2
Date:                Wed, 27 Apr 2022   Prob (F-statistic):          4.50e-177
Time:                        07:09:35   Log-Likelihood:                 701.02
No. Observations:                 492   AIC:                            -1394.
Df Residuals:                     488   BIC:                            -1377.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.7275      0.003    276.124      0.000       0.722       0.733
GRE Score      0.0356      0.005      7.673      0.000       0.027       0.045
LOR            0.0196      0.003      5.775      0.000       0.013       0.026
CGPA           0.0770      0.005     15.039      0.000       0.067       0.087
==============================================================================
Omnibus:                      110.810   Durbin-Watson:                   0.894
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              296.733
Skew:                          -1.098   Prob(JB):                     3.67e-65
Kurtosis:                       6.106   Cond. No.                         3.78
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specif
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:117: FutureWarning:
  x = pd.concat(x[::order], 1)
```

```
get_stat_vif(X_1,Y)
```

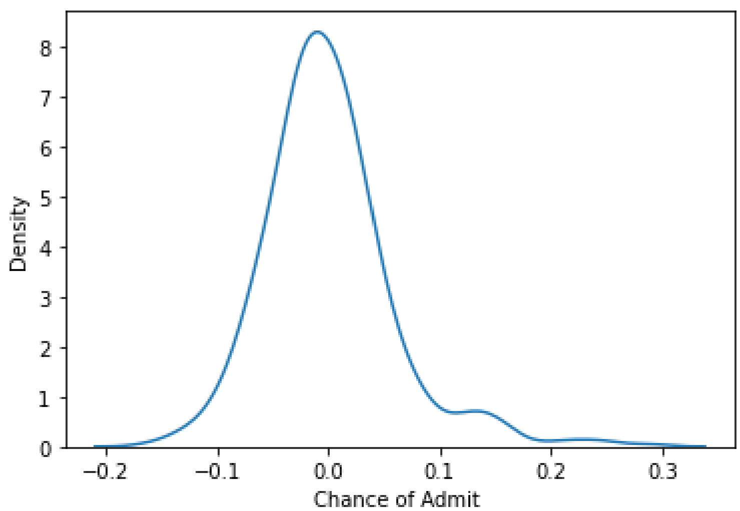
```
      Features   VIF
2         CGPA  3.77
0    GRE Score  3.11
1          LOR  1.65
```

```
model = get_model(X_1,Y)
```

```
model coefficient: [0.03564324 0.01955734 0.07698386]
model intercept: 0.7275406504065041
model score: 0.813
Adjusted R-squared: 0.812
```

```
preds = model.predict(X_1)
errors = preds - Y
sns.kdeplot(errors)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f40d5696410>
```



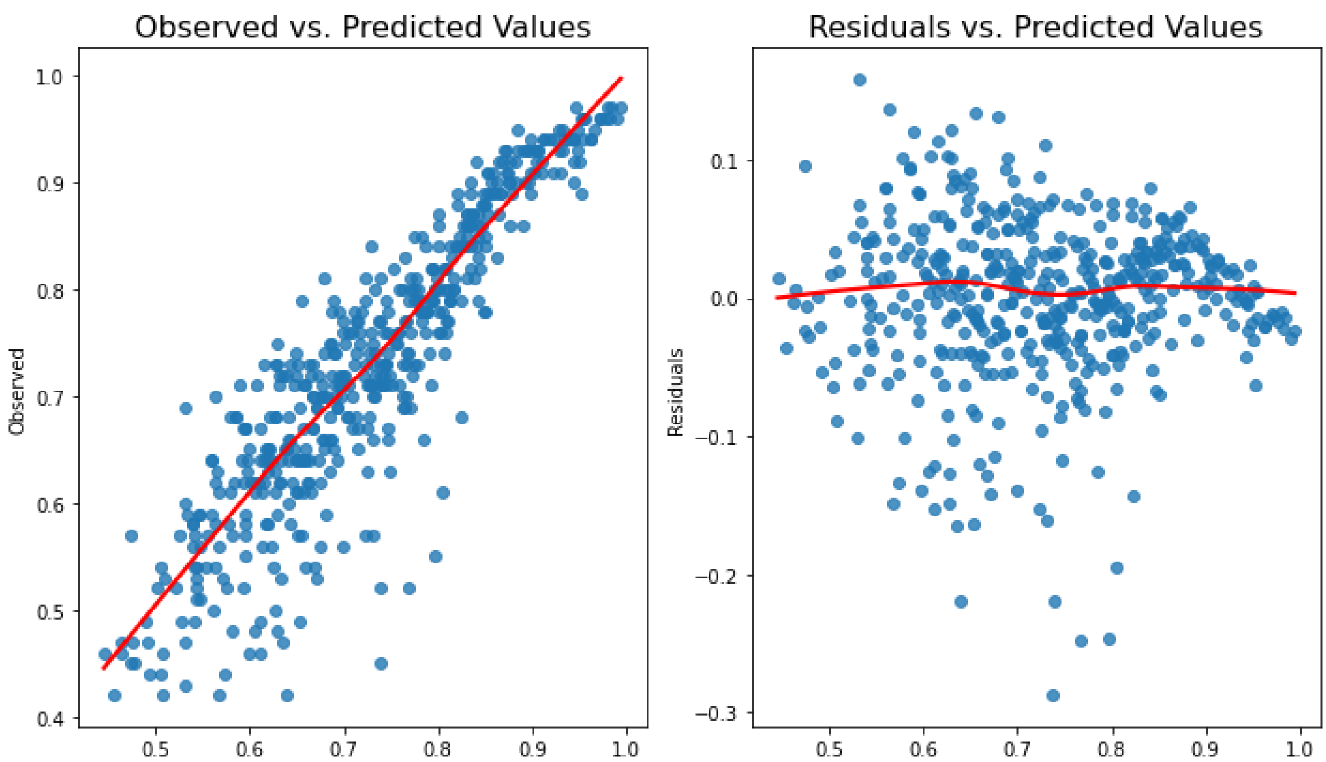

```
check_normality_test(errors)
```

```
Target variable is not gausian distribution
```


```
from IPython.core.pylabtools import figsize
#residual plot
fig, ax = plt.subplots(1,2,figsize=(10,6))

sns.regplot(x=preds, y=Y, lowess=True, ax=ax[0], line_kws={'color': 'red'})
ax[0].set_title('Observed vs. Predicted Values', fontsize=16)
ax[0].set(xlabel='Predicted', ylabel='Observed')

sns.regplot(x=preds, y=sm_model.resid, lowess=True, ax=ax[1], line_kws={'color': 'red'})
ax[1].set_title('Residuals vs. Predicted Values', fontsize=16)
ax[1].set(xlabel='Predicted', ylabel='Residuals')
plt.tight_layout()
```



Insights based on stats final models

- Residual mean of model is zero and also Homoscedasticity is also there.
- Independent error is lesser than 2 which proves statistically positive correlation.
- Residual error almost has gaussian distribution which is visibile by qq plot and kde plot.

- By droping columns R2score drop by 1% which is negligible since our model becames much simpler.

Model performance evaluation

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error,mean_absolute_percentage_

def model_performance(X,Y):
  x_train,x_test,y_train, y_test = train_test_split(X,Y, test_size=0.1,random_state = 1)
  model = get_model(x_train,y_train)
  y_pred = model.predict(x_test)
  print("Mean absolute error:", round(mean_absolute_error(y_test, y_pred),3) )
  print("Mean squared error:", round(mean_squared_error(y_test, y_pred),3))
  print("Root Mean squared error:",round(np.sqrt(mean_squared_error(y_test, y_pred)),3))
  print("Mean absolute percentage error:", round(mean_absolute_percentage_error(y_test,y_pred
```

```
#Lets check with all columns
model_performance(X,Y)
```

```
model coefficient: [0.01972149 0.01655993 0.00514903 0.0066046  0.01398846 0.06657419
 0.01298132]
model intercept: 0.7267804744164308
model score: 0.828
Adjusted R-squared: 0.825
Mean absolute error: 0.039
Mean squared error: 0.003
Root Mean squared error: 0.057
Mean absolute percentage error: 0.057
```

```
model_performance(X_1,Y)
```

```
model coefficient: [0.03572918 0.0190849  0.07820101]
model intercept: 0.7272037000083033
model score: 0.815
Adjusted R-squared: 0.814
Mean absolute error: 0.042
Mean squared error: 0.003
Root Mean squared error: 0.056
Mean absolute percentage error: 0.061
```

Insights and Recommendation

- From two models(one with all columns and one with selected columns) has almost equal metrics which helps to design our model is simple and also performs well as equal first model.

- More the GRE score, CGPA and LOR, more the chance of admit.
- We need more datasets based on chance of admit from 0.5 to 0.8 where error value is high compared to other

Recommendations

- Based on customer's LOR score, CGPA score, GRE score, we can suggest approriate rating of selected university
- With this model, we can auto chatbot so that customers wont need to wait get information and with our model and chatbot system they will get list of universities.
- With this approach and we can create a model not only for us based universities but also to European and canadian universities.