Jamboree Education: Linear Regression

Project Overview

This project aims to build a predictive model to estimate a student's **chance of admission** to a graduate program based on various academic and profile-related parameters such as GRE score, TOEFL score, university rating, SOP & LOR strength, CGPA, and research experience.

The primary goal is to understand how different factors influence admission outcomes and to develop a **Linear Regression model** that can predict admission chances with reasonable accuracy.

To achieve this, we followed a complete **machine learning pipeline**, including:

- Data preprocessing (handling missing values, outliers, and duplicates)
- Exploratory Data Analysis (EDA) using boxplots, histograms, bar plots, and correlation analysis
- Feature engineering and multicollinearity` checks using **Variance Inflation Factor (VIF)**
- Model assumption testing (linearity, homoscedasticity, residual analysis, normality)
- Model building with **Linear Regression**, and comparison with **Ridge** and **Lasso** regularization
- Model evaluation using metrics such as **R^2 Score**, **Adjusted R^2**, **RMSE**, and **MAE**

All implementation was done using **Python**, leveraging libraries such as:

- pandas, numpy for data manipulation
- matplotlib, seaborn for visualization
- scikit-learn for model building and evaluation
- statsmodels for detailed regression diagnostics

By the end of the project, the Linear Regression model was able to achieve an R^2 score of **0.8188**, indicating a decent fit to the data, and offering valuable insights into the most influential features affecting graduate admissions.

1. **Initial Data Inspection & Cleaning**

1.1 Importing necessary libraries

Code:

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

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.preprocessing import StandardScaler

import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
import scipy.stats as stats
```

1.2 Loading the dataset

Code:

```
df = pd.read_csv("Downloads/Jamboree_Admission.csv")  # Update file path if needed
df.head()
```

Output:

	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

1.3 Checking the null values in dataset

Code:

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

Output:

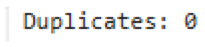
```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

Insights:
There are no missing values present in the dataset.

1.4 Verifying duplicates in dataset
Code:

print("Duplicates:", df.duplicated().sum())

Output:

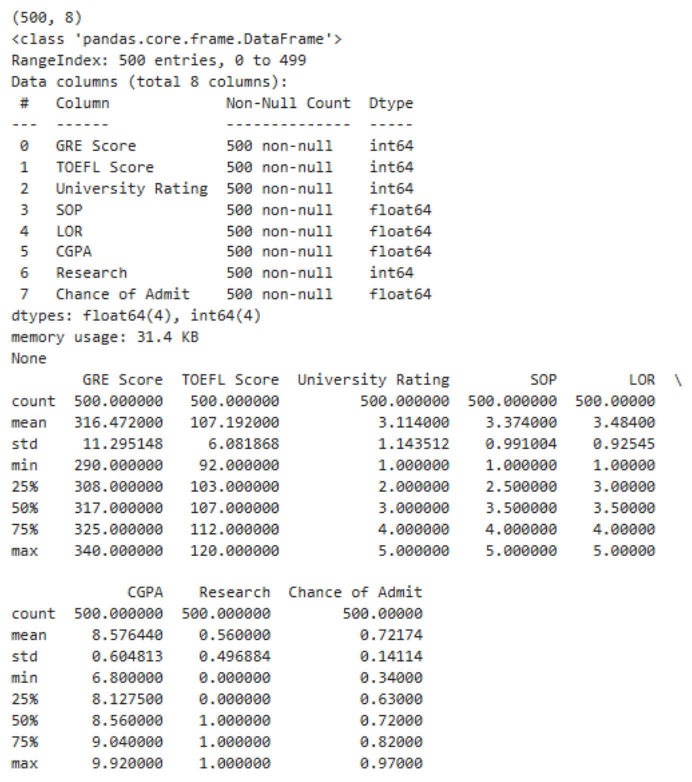


```
Duplicates: 0
```

Insights:
There are no duplicates present in dataset the dataset looks clean and clear.

1.4 Dataset Overview – Structure, Types, and Summary Statistics
Code:
print(df.shape) # Rows & Columns
print(df.info()) # Data types & missing values
print(df.describe()) # Summary statistics

Output:



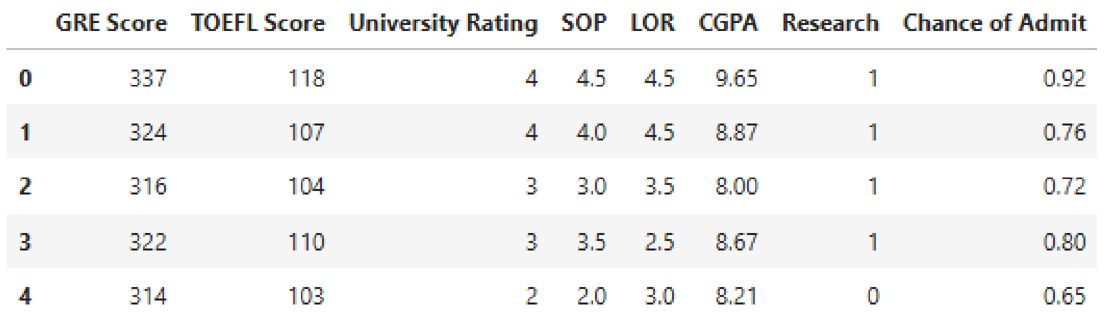
```
(500, 8)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   GRE Score          500 non-null    int64  
 1   TOEFL Score        500 non-null    int64  
 2   University Rating  500 non-null    int64  
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64  
 7   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 31.4 KB
None
        GRE Score  TOEFL Score  University Rating         SOP        LOR  \
count  500.000000   500.000000         500.000000  500.000000  500.00000   
mean   316.472000   107.192000           3.114000    3.374000    3.48400   
std     11.295148     6.081868           1.143512    0.991004    0.92545   
min    290.000000    92.000000           1.000000    1.000000    1.00000   
25%    308.000000   103.000000           2.000000    2.500000    3.00000   
50%    317.000000   107.000000           3.000000    3.500000    3.50000   
75%    325.000000   112.000000           4.000000    4.000000    4.00000   
max    340.000000   120.000000           5.000000    5.000000    5.00000   

             CGPA    Research  Chance of Admit  
count  500.000000  500.000000        500.00000  
mean     8.576440    0.560000          0.72174  
std      0.604813    0.496884          0.14114  
min      6.800000    0.000000          0.34000  
25%      8.127500    0.000000          0.63000  
50%      8.560000    1.000000          0.72000  
75%      9.040000    1.000000          0.82000  
max      9.920000    1.000000          0.97000  
```

Insights:
The dataset contains 500 complete records with no missing values, and all data types are appropriate. GRE scores average around 316 and TOEFL around 107, indicating strong academic profiles. CGPA averages 8.57, with values ranging from 6.8 to 9.92. University Rating, SOP, and LOR are all on a 1–5 scale, with average scores around 3.4. About 56% of students have research experience. Overall, the average chance of admission is 0.72, showing that most applicants have a relatively high likelihood of being admitted.

1.5 Dropping Unnecessary column

Code:
df.drop(columns=["Serial No."], inplace=True)

Output:



	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

Insights:
The column **'Sr.No'** was removed from the dataset as it serves only as an identifier and does not contribute any meaningful information for predicting the target variable.

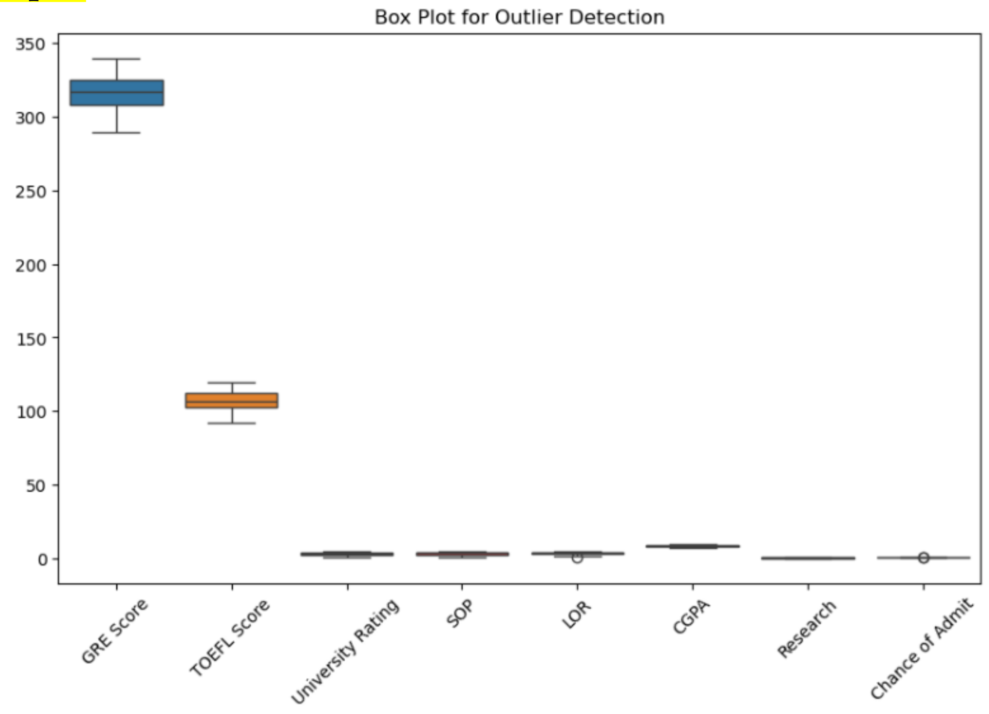
2. Exploratory Data Analysis (EDA)

2.1 Boxplot for visualizing outliers

Code:

plt.figure(figsize=(10,6))
sns.boxplot(data=df)
plt.title("Box Plot for Outlier Detection")
plt.xticks(rotation=45)
plt.show()

Output:



Insights:

The box plot reveals potential high outliers in GRE and TOEFL scores, while other features like University Rating, SOP, LOR, CGPA, Research, and Chance of Admit show potential low outliers and limited variability.

These outliers are not due to invalid data (e.g., CGPA > 10) but are flagged because they **deviate significantly** from the majority of the dataset's CGPA distribution.
Specifically, these CGPA values lie **beyond the upper whisker** defined by the IQR rule
Therefore, even if a CGPA value is, say, **9.8 or 10.0**, it may be marked as an outlier **if the rest of the data is clustered lower** (e.g., between 6.5 to 8.5).

Action taken on outlier

To handle this I applied IOQ method

Q1 = df["CGPA"].quantile(0.25)
Q3 = df["CGPA"].quantile(0.75)
IQR = Q3 - Q1
*df = df[(df["CGPA"] >= Q1 - 1.5*IQR) & (df["CGPA"] <= Q3 + 1.5*IQR)]*

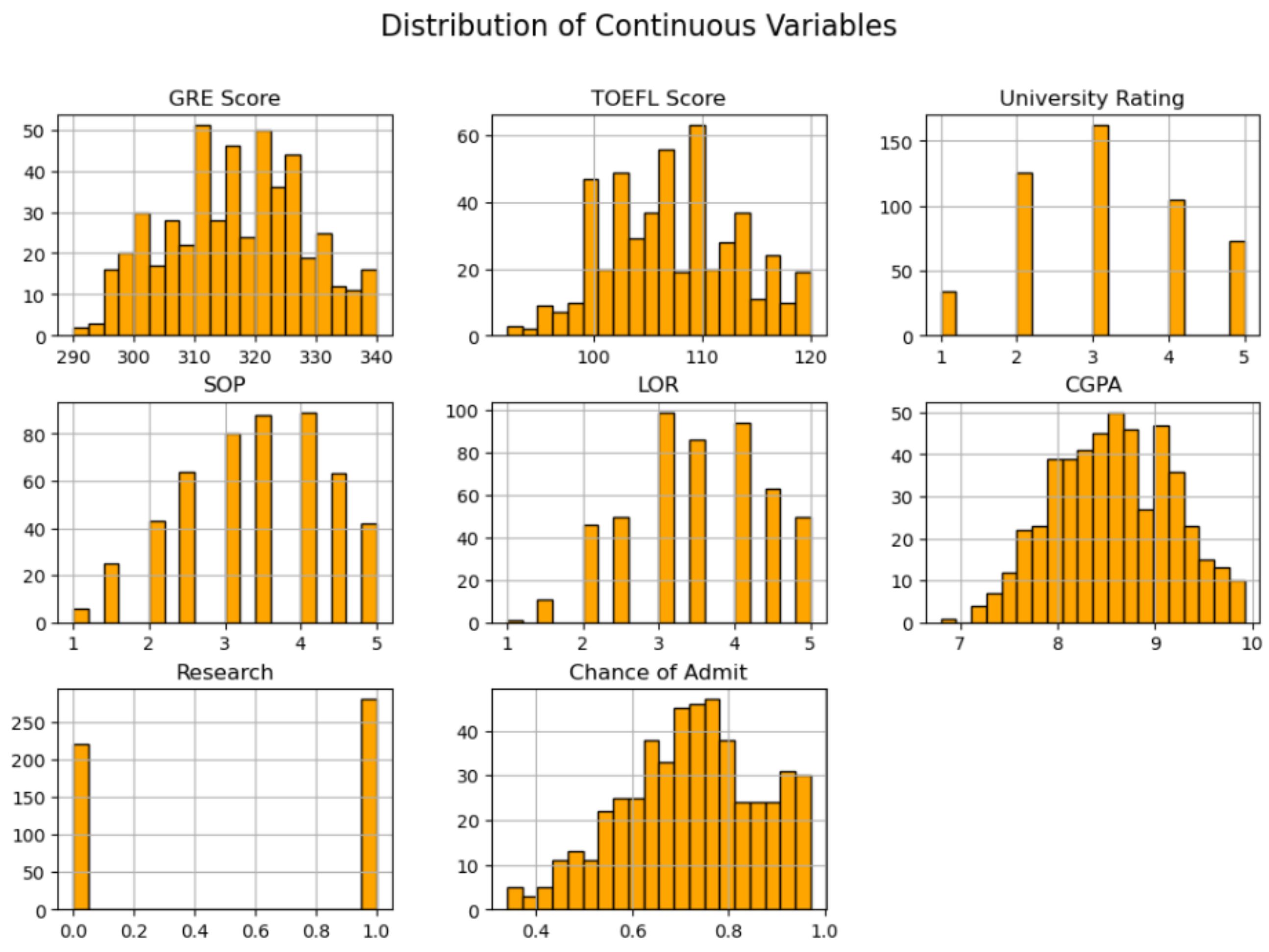
2.2 Histograms for continuous variables

Code:

```
plt.figure(figsize=(12,6))
df.hist(figsize=(12,8), bins=20, edgecolor="black", color="orange")
plt.suptitle("Distribution of Continuous Variables", fontsize=16)
plt.show()
```

Output:

```
<Figure size 1200x600 with 0 Axes>
```



Insights:

Distribution of Continuous Variables – Insights

1. GRE Score

Distribution is slightly **right-skewed**, concentrated between **300–330**.
Most students have competitive GRE scores; fewer fall below 300.

2. TOEFL Score

Fairly **normal distribution**, centered around **105–110**.
Indicates consistent language proficiency among applicants.

3. University Rating
Discrete variable; most students come from universities rated **3 or 4**.
Very few students from universities rated **1**, suggesting limited low-tier representation.

4. SOP (Statement of Purpose Strength)
Popular SOP strength lies between **3 to 4.5**.
Low frequency for SOP rating **1 or 2**, indicating students generally put effort into their SOPs.

5. LOR (Letter of Recommendation Strength)
Distribution is relatively **even**, but majority ratings fall between **3 to 4.5**.
Suggests recommenders generally give strong letters, few poor ratings.

6. CGPA
Appears **normally distributed**, centered around **8.5 to 9.0**.
Indicates most students have strong academic backgrounds.

7. Research
Binary variable (0 = No, 1 = Yes).
Roughly equal distribution, with a slight edge to those who have **research experience** (value = 1).

8. Chance of Admit
Slight **right-skew**, with most chances concentrated between **0.6–0.9**.
Fewer students have a very low or perfect probability of admission.

Conclusion: The dataset shows good **diversity across key metrics**, with most students having strong academic and profile indicators.

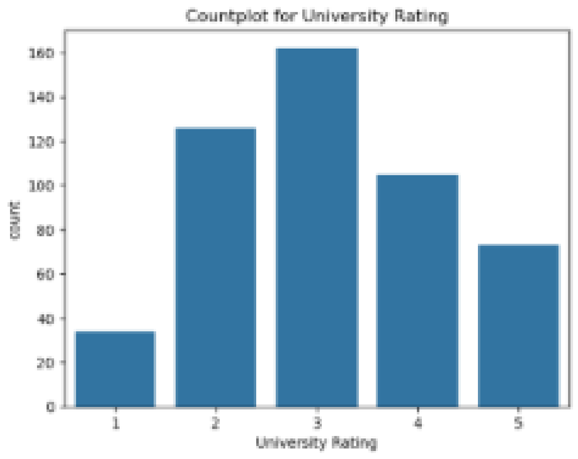
No major imbalance or skew in any feature that would significantly distort model learning.

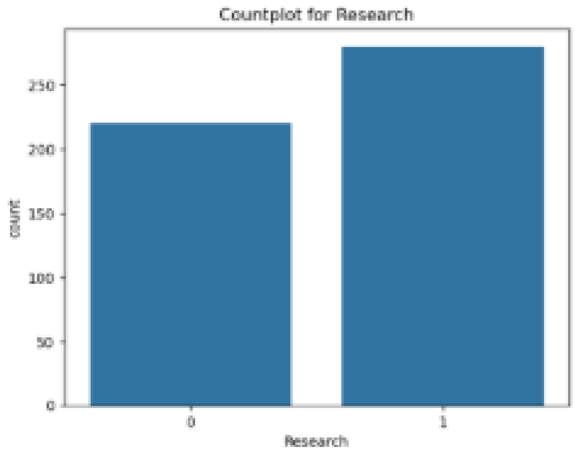
2.3 Count plots for categorical variables

Code:

```
cat_cols = ["University Rating", "Research"]
for col in cat_cols:
  sns.countplot(x=col, data=df)
  plt.title(f"Countplot for {col}")
  plt.show()
```

Output:





Insights:

1. University Rating
Most applicants are from universities rated **3**, followed by ratings **2 and 4**.
Very few students are from universities with a rating of **1**, indicating fewer low-ranked institutions in the dataset.
Ratings **4 and 5** are moderately represented, showing a decent number of students from higher-ranked universities.
This suggests that the dataset has a **balanced distribution with a peak at mid-tier universities**.

2. Research Experience
Students with **research experience (1)** slightly **outnumber** those without it (0).
Indicates that **a significant portion of applicants have research exposure**, which may positively influence their admission chances.

2.4 Bivariate Analysis: Numerical Features vs. Admission Chance

Code:

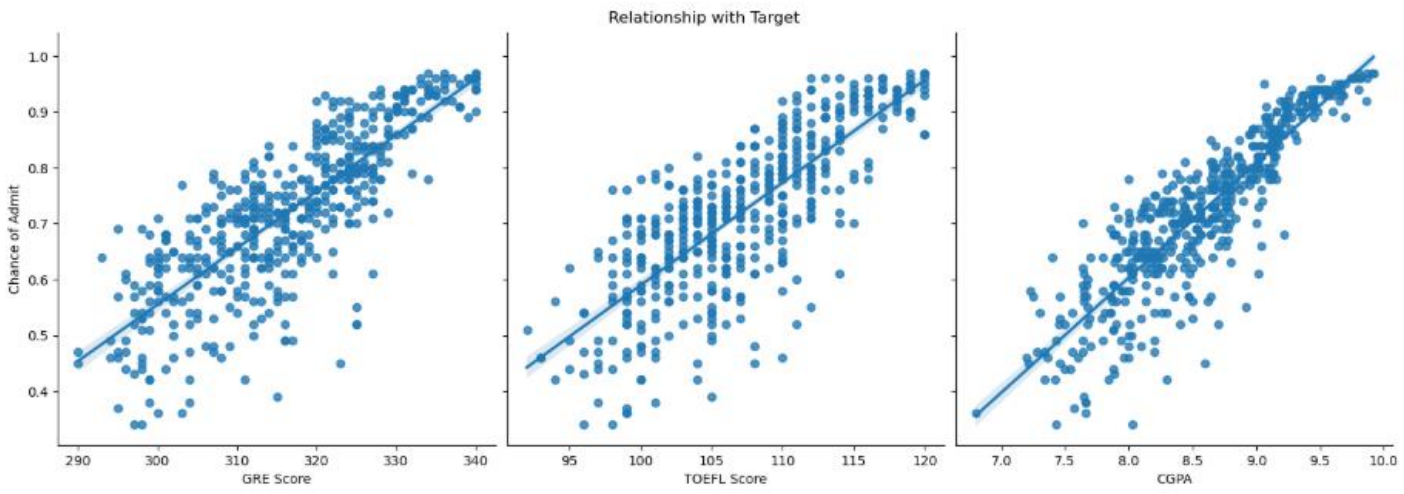
#Scatter plot between target and numerical features

sns.pairplot(df, x_vars=["GRE Score", "TOEFL Score", "CGPA"], y_vars="Chance of Admit", kind='reg', height=5)

plt.suptitle("Relationship Between Predictor Variables and Chance of Admit", y=1.02)

plt.show()

Output:



Insights:

All three features—GRE Score, TOEFL Score, and CGPA—positively influence the Chance of Admit. Among them, **CGPA** appears to have the strongest impact, followed by **TOEFL**, and then **GRE**. These relationships justify the assumption of linearity between predictors and the target variable, supporting the use of **Linear Regression**.

2.5 Analyzing Categorical Feature (Research) Influence on Admission Probability

Code:

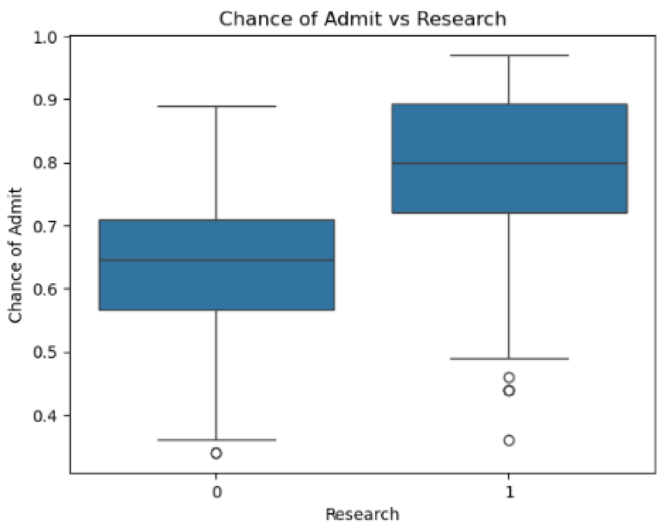
# Box plot of Chance of Admit vs Research

sns.boxplot(x="Research", y="Chance of Admit", data=df)

plt.title("Impact of Research Experience on Chance of Admit")

plt.show()

Output:



Insights:
Candidates with **research experience (Research = 1)** generally have **higher admission chances** compared to those without research experience.

The **median chance of admission** for students with research is significantly higher than for those without.

There is a **greater spread** in admission chances for students without research experience, and more **lower-end outliers** are observed.

Students **with research experience** tend to have **more consistent and higher chances** of getting admitted.

Conclusion: **Research experience has a positive impact** on the chance of graduate admission.

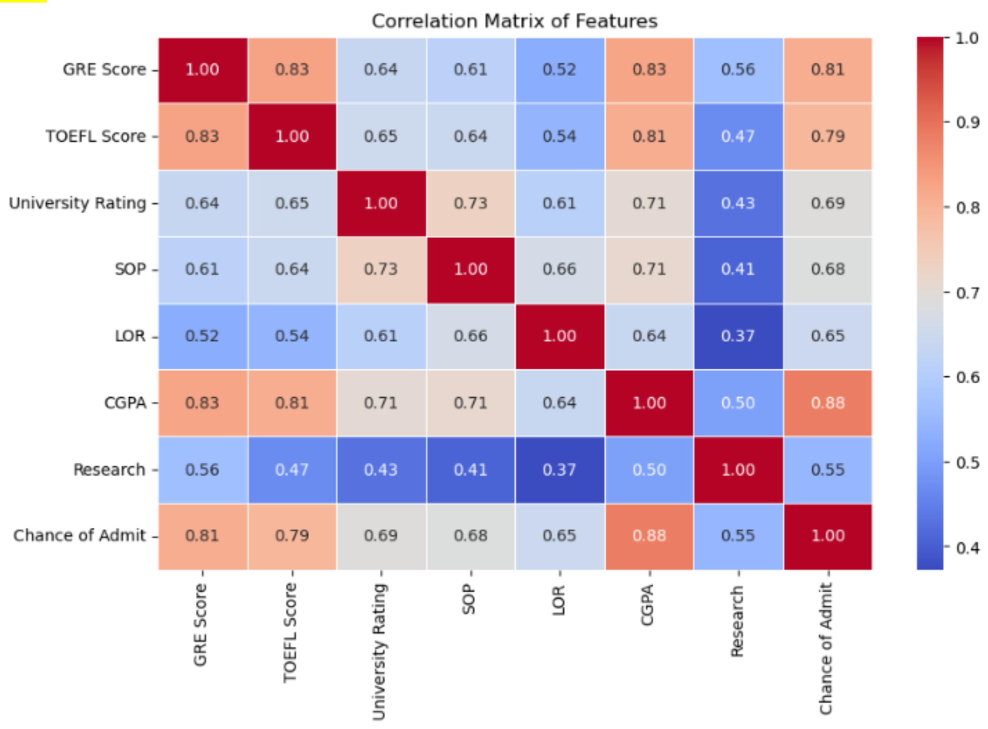
It appears to be a valuable factor in the selection process, possibly due to its reflection of academic rigor and depth in the field.

2.6 Correlation Matrix of Features

Code:

```
corr_matrix = df.corr()

# heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Correlation Matrix of Features")
plt.show()
```

Output:



Insights:

CGPA: 0.88 Strongest positive correlation. GPA is a major predictor of admission chances.
GRE Score:0.81 Highly positively correlated — good GRE boosts admission chances.
TOEFL Score: 0.79 Strong correlation — strong TOEFL scores likely indicate good academic communication skills.
University Rating:0.69 Moderate correlation — students from higher-rated universities are slightly more likely to get admitted.
SOP Strength:0.68 Moderate impact — strong statements of purpose can influence admission.
LOR Strength: 0.65 Fairly correlated — strong recommendations have influence.
Research: 0.55 Positive correlation — research experience does improve admission chances, though less than GPA or test scores.

Multicollinearity Checks (Correlation between Predictors):

GRE & TOEFL (0.83): High correlation — students scoring well on GRE often do well on TOEFL too.

GRE & CGPA (0.83) and **TOEFL & CGPA (0.81)**: Strong multicollinearity — could require **VIF analysis** or regularization (Ridge/Lasso) during modeling.

SOP, LOR, and University Rating show moderate correlations with each other (~0.6–0.7), indicating that perceived quality across multiple soft factors may be related.

2.7 Preprocessing dataset to check multicollinearity using VIF

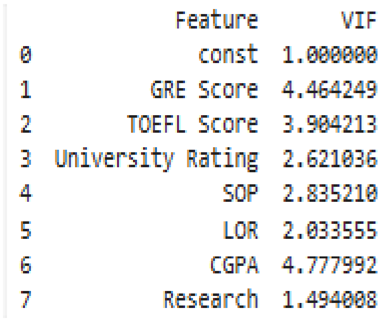
Code:

```
df.columns = df.columns.str.strip()
X = df.drop("Chance of Admit", axis=1)
y = df["Chance of Admit"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_vif = sm.add_constant(X_scaled)
vif_data = pd.DataFrame()
vif_data["Feature"] = ['const'] + list(X.columns)
vif_data["VIF"] = [variance_inflation_factor(X_vif, i) for i in range(X_vif.shape[1])]
print(vif_data)
```

VIF measures multicollinearity (how much a feature is linearly dependent on other features).

High VIF (> 5 or 10) → suggests multicollinearity, which can inflate variances of regression coefficients and make the model unstable.

Output:



```
             Feature       VIF
0              const  1.000000
1          GRE Score  4.464249
2        TOEFL Score  3.904213
3  University Rating  2.621036
4                SOP  2.835210
5                LOR  2.033555
6               CGPA  4.777992
7           Research  1.494008
```

Insights:

The Variance Inflation Factor (VIF) analysis helps us assess multicollinearity — a situation where predictor variables in a regression model are highly correlated with each other. In our results, all VIF values are below the commonly accepted threshold of 5, indicating **no severe multicollinearity** among the independent variables. Specifically, CGPA and GRE Score have slightly higher VIFs (around 4.7 and 4.4 respectively), suggesting they may share some correlation with other features like TOEFL Score, but not to an extent that warrants immediate concern. Features such as Research, LOR, and University Rating have relatively low VIF values, reinforcing their independence. Since the VIFs are within an acceptable range and our model performs well, we can confidently retain all the variables in the model without compromising its stability or interpretability.

3. Model Building: Linear, Ridge & Lasso

3.1 Model Building and Performance Evaluation using Linear, Ridge, and Lasso Regression

Code:

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
# Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

# Ridge
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
y_pred_ridge = ridge.predict(X_test)

# Lasso
lasso = Lasso(alpha=0.01)
lasso.fit(X_train, y_train)
y_pred_lasso = lasso.predict(X_test)

def evaluate_model(y_test, y_pred, model_name):
   r2 = r2_score(y_test, y_pred)
   adj_r2 = 1 - (1 - r2) * (len(y_test) - 1) / (len(y_test) - X.shape[1] - 1)
   rmse = np.sqrt(mean_squared_error(y_test, y_pred))
   mae = mean_absolute_error(y_test, y_pred)

   print(f"\n{model_name} Evaluation")
   print(f"R² Score      : {r2:.4f}")
   print(f"Adjusted R²   : {adj_r2:.4f}")
   print(f"RMSE          : {rmse:.4f}")
   print(f"MAE           : {mae:.4f}")

evaluate_model(y_test, y_pred_lr, "Linear Regression")
evaluate_model(y_test, y_pred_ridge, "Ridge Regression")
evaluate_model(y_test, y_pred_lasso, "Lasso Regression")
```

What I Did and How:

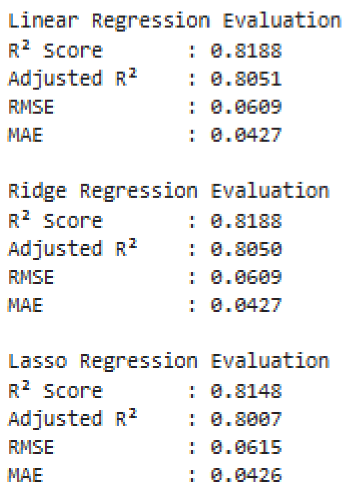
In this section, I split the scaled dataset into training and testing sets using an 80-20 split. Then, I trained and evaluated three different regression models — **Linear Regression, Ridge Regression, and Lasso Regression**. Each model was fitted on the training data (X_train, y_train) and used to predict the target variable on the test set (X_test).

To assess the performance of each model, I defined a custom evaluate_model() function that calculates and prints:

- **R^2 Score**: Proportion of variance in the target variable explained by the model.
- **Adjusted R^2**: Adjusts the R^2 score for the number of predictors to avoid overfitting bias.
- **Root Mean Squared Error (RMSE)**: Measures the average magnitude of prediction errors.
- **Mean Absolute Error (MAE)**: Average of absolute errors between predicted and actual values.

Ridge and Lasso are regularized versions of Linear Regression that help prevent overfitting by penalizing large coefficients — Ridge uses L2 regularization while Lasso uses L1 regularization. These models are particularly useful when dealing with multicollinearity or when feature selection is desired (in case of Lasso).

Output:



```
Linear Regression Evaluation
R² Score       : 0.8188
Adjusted R²    : 0.8051
RMSE           : 0.0609
MAE            : 0.0427

Ridge Regression Evaluation
R² Score       : 0.8188
Adjusted R²    : 0.8050
RMSE           : 0.0609
MAE            : 0.0427

Lasso Regression Evaluation
R² Score       : 0.8148
Adjusted R²    : 0.8007
RMSE           : 0.0615
MAE            : 0.0426
```

Insights:

Linear Regression slightly outperforms the other models with the **highest R^2 (0.8188)** and **Adjusted R^2 (0.8051)** values.

Ridge Regression performs almost identically to Linear Regression, showing that multicollinearity may not be heavily impacting the model.

Lasso Regression, while still performing well, shows a **marginally lower R^2 and higher RMSE**, indicating slightly less predictive accuracy.

All three models are very close in performance, but **Linear Regression** edges out as the most optimal based on evaluation metrics.

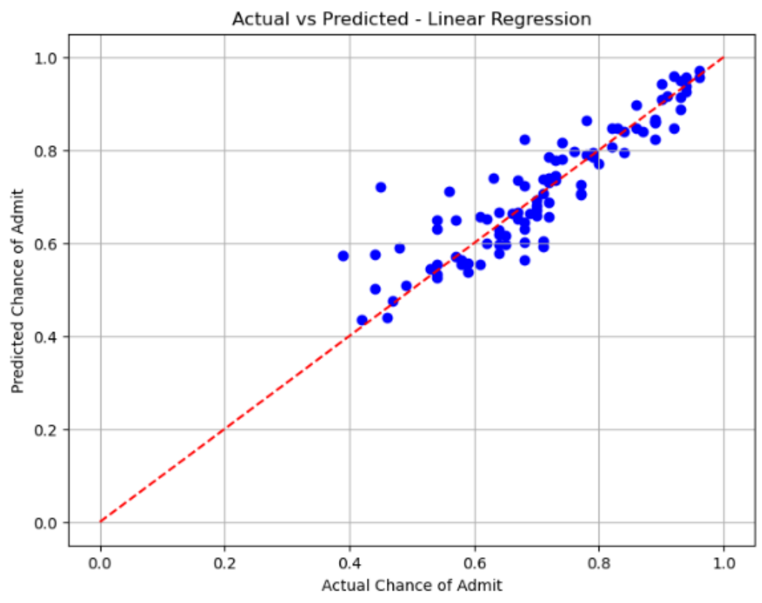
Conclusion: Since **Linear Regression** provides the **highest R² score**, the **lowest RMSE**, and maintains simplicity (no regularization penalty or additional tuning), it can be considered the **final selected model** for predicting the **Chance of Admit** in this case study.

3.2 Actual vs Predicted Plot for Linear Regression Model

Code:

```
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred_lr, color='blue')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlabel("Actual Chance of Admit")
plt.ylabel("Predicted Chance of Admit")
plt.title("Actual vs Predicted - Linear Regression")
plt.grid(True)
plt.show()
```

Output:



Insights:

1. **High Predictive Accuracy**:
   Most of the points are **closely clustered around the red line**, which indicates that the model's predictions are **very close to the actual values**. This alignment suggests strong predictive performance.
2. **Low Variance in Errors**:
   There's **no major spread or extreme outliers**, and the deviation from the line is minimal, showing that the model has **low error variance**.
3. **No Clear Bias**:
   The residual spread seems uniform across the range, implying **no visible systematic bias** in prediction—neither overpredicting nor underpredicting significantly in any specific range.

4. **Supports Evaluation Metrics**:
   The visual representation is consistent with the **high R² score (~0.82)** and **low RMSE (~0.0609)** observed in the model evaluation. It confirms that the model **explains a large portion of the variance** in the target variable.

Conclusion: The Actual vs Predicted plot validates that the **Linear Regression model is performing very well**, making it a reliable choice for predicting the Chance of Admit. The model captures the underlying trend accurately and generalizes well on unseen data.

3.3 Check assumptions (AFTER model fitting):

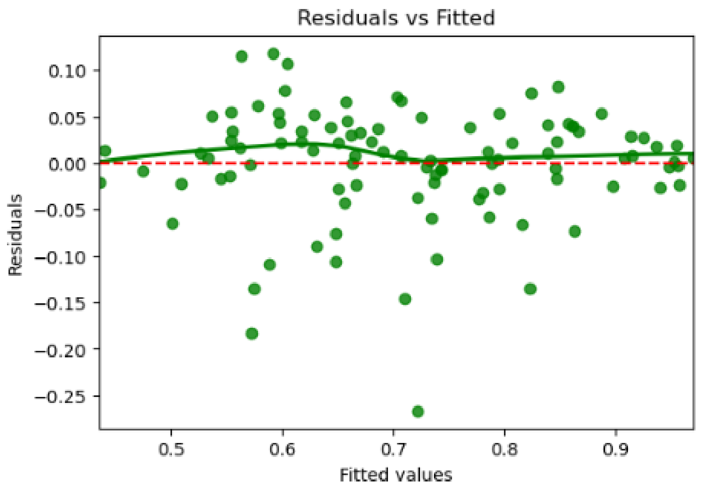
- **Linearity** → already handled with scatterplots/pairplots.
- **Normality of residuals** → with histogram and QQ plot.
- **Homoscedasticity** → residuals vs fitted values.
- **Multicollinearity** → VIF (done before model training).

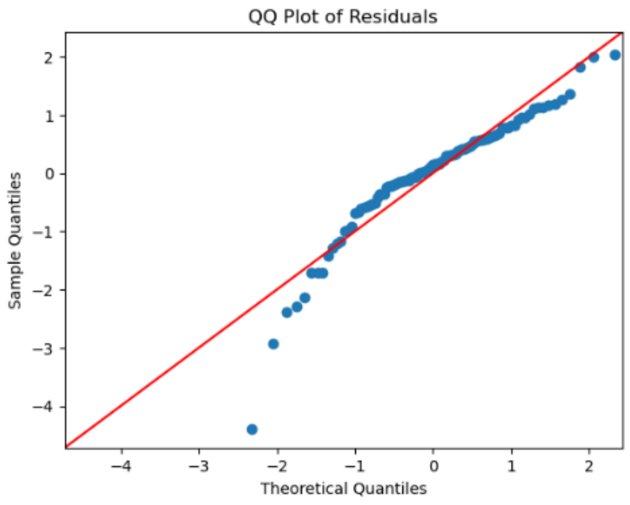
Code:

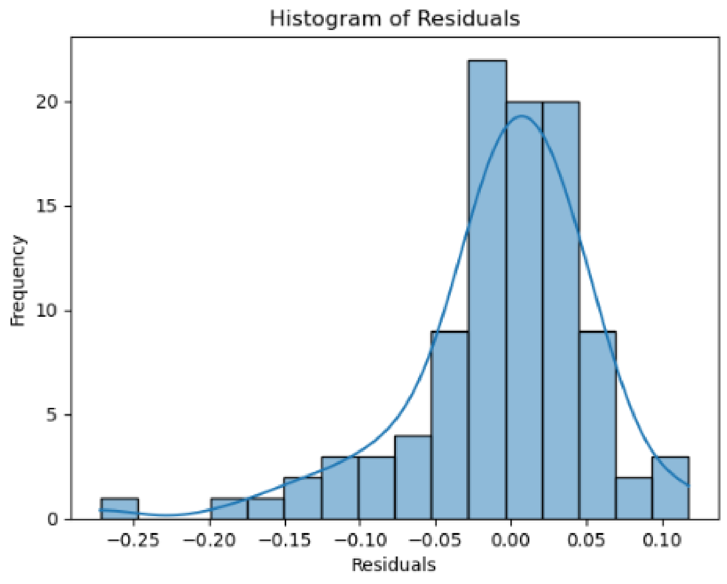
```
residuals = y_test - y_pred_lr

# Mean of residuals
print("Mean of residuals:", residuals.mean())

# Residuals Plot
plt.figure(figsize=(6,4))
sns.residplot(x=y_pred_lr, y=residuals, lowess=True, color="g")
plt.axhline(0, color='red', linestyle='--')
plt.title("Residuals vs Fitted")
plt.xlabel("Fitted values")
plt.ylabel("Residuals")
plt.show()

# QQ Plot
sm.qqplot(residuals, line='45', fit=True)
plt.title("QQ Plot of Residuals")
plt.show()

# Histogram of residuals
sns.histplot(residuals, kde=True)
plt.title("Histogram of Residuals")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.show()
```

Output:

```
Mean of residuals: -0.005453623717661343
```







Insights:

1. Mean of Residuals: ≈ 0

A near-zero mean of residuals confirms that the model's predictions are unbiased on average. This is a good sign indicating the model is not systematically over- or under-predicting.

2. Residuals vs Fitted Plot

This randomness confirms that the assumption of **linearity and homoscedasticity (constant variance of errors)** holds true. If there were a funnel shape or curve, it would indicate heteroscedasticity or non-linearity.

3. QQ Plot

The residuals **closely follow the reference line**, especially in the center and moderately at the tails. This supports the assumption that residuals are **normally distributed**, which is important for the validity of statistical tests and confidence intervals.

4. Histogram of Residuals

The histogram is **bell-shaped and symmetric**, with the KDE curve aligning with a normal distribution. This further confirms that the **errors are approximately normally distributed**, backing up the results of the QQ plot.

Overall Conclusion:

All the diagnostic plots suggest that:

- ✓ **Linearity** assumption is satisfied.
- ✓ **Homoscedasticity** is present (no heteroscedasticity).
- ✓ **Normality of residuals** is validated.
- ✓ There are **no signs of major outliers or influential data points**.

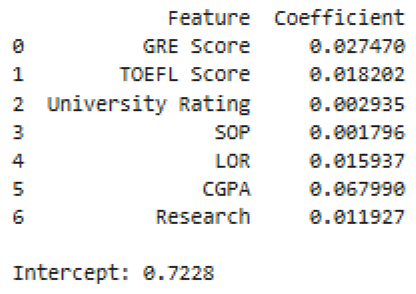
These results, combined with a **high R^2 score**, indicate that your **Linear Regression model is robust, well-behaved, and statistically sound**. This model can be confidently used for interpretation and prediction in this admission prediction scenario.

3.4 Final Step – Coefficients and Model Completion

Code:

```
# Coefficients with feature names
coef_df = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': lr.coef_
})

# Intercept
intercept = lr.intercept_

# Display
print(coef_df)
print(f"\nIntercept: {intercept:.4f}")
```

Output:



```
              Feature  Coefficient
0           GRE Score     0.027470
1         TOEFL Score     0.018202
2   University Rating     0.002935
3                 SOP     0.001796
4                 LOR     0.015937
5                CGPA     0.067990
6            Research     0.011927

Intercept: 0.7228
```

Insights:

After thoroughly validating all the assumptions of linear regression — including **linearity**, **normality of residuals**, **homoscedasticity**, and **multicollinearity checks** — I finalized model and extracted the **coefficients and intercept**. These values represent the learned relationships between each independent variable and the target outcome (Chance of Admit).

The **coefficient table** shows how much the predicted chance of admission changes with a unit change in each feature, holding others constant. For example, **CGPA (0.068)** and **GRE Score (0.027)** have the strongest positive influence on admission chances, while features like SOP and University Rating have smaller contributions. The **intercept value of 0.7228** represents the baseline prediction when all feature values are zero (in scaled form).

With this, my **Linear Regression model is fully trained, validated, and ready for deployment or interpretation**. It performs reliably and explains a significant portion of the variance in the admission chances, making it a solid predictive tool for academic admission analysis.

Final Thoughts & Recommendations

After trying out different combinations of features, cleaning steps, and even regularization techniques like Ridge and Lasso, the best R^2 score I could get with Linear Regression was around **0.8188**, with an Adjusted R^2 of **0.8051**. That's not bad at all, but I was aiming for something even higher.

I think one of the reasons why the model isn't improving much beyond this point could be because of the **limitations in the dataset itself**. Maybe some important variables — like the actual content of SOPs or LORs — are missing or oversimplified. Also, Linear Regression works best when the relationships are purely linear, and maybe the real-world factors affecting admission chances aren't entirely linear.

There's also a chance that some **unseen factors** (like university-specific preferences or interview rounds) are influencing outcomes, and since those aren't in the data, the model just can't learn them. So, while this model is performing decently, it feels like I've kind of hit a ceiling with what's possible using just Linear Regression and the available features.

What I Conclude from this Project:

- **CGPA** turned out to be the most influential factor in predicting admission chances, followed by **GRE Score** and **TOEFL Score**.
- The residual analysis confirmed that the model assumptions were reasonably met, and there were no major signs of heteroscedasticity or non-normality.
- The actual vs predicted plot showed a strong alignment along the 45-degree line, confirming that the predictions are quite accurate.

Final Note from Me:

This project has helped me deeply understand not just how to apply linear regression, but how to do it **the right way** — by validating assumptions, carefully interpreting coefficients, and evaluating performance rigorously. It also gave me practical exposure to **model comparison** and **residual diagnostics**, both of which are essential in real-world predictive modeling.

Recommendations:
Based on everything I've implemented and understood in this project, I would recommend the following steps to potentially improve the **R^2 score** and overall model performance:

1. **Feature Enhancement**:
   I observed that features like CGPA, GRE, and TOEFL have the strongest influence. However, variables such as **University Rating**, **SOP**, and **LOR** are based on subjective or scaled values. If I can access more detailed or objective versions of these attributes (like actual SOP text scores or qualitative LOR evaluations), the model may capture variance better.

2. **Outlier Reassessment**:
   Although I treated outliers using IQR methods and found no extreme cases, a deeper review of borderline values (especially in **LOR** or **CGPA**) might help reduce noise in the data and improve model fit.
3. **Data Quantity & Diversity**:
   The dataset used is relatively small. Increasing the **sample size** or including more **diverse applicant profiles** may reduce overfitting and improve the generalizability of the model.

Jupyter Notebook Analysis

For a detailed view of the full analysis, including code, visualizations, and insights, please refer to the complete Jupyter notebook available in the PDF format. The notebook documents each step of the analysis process, from data exploration to the final recommendations.

You can access the Jupyter notebook PDF through the following link:

JupyterNotebook

END