# Analysis of College Dataset to Determine Competitiveness of a College

## Assignment 3

### Course Name: Intermediate Analytics

### Name of the Instructor: Roy Wada, PhD

### Name of the Student: Kaushal Nagrecha

### **1. Introduction**

This report details an analysis of a dataset, beginning with an Exploratory Data Analysis (EDA) to identify key patterns and relationships. Following this, a Generalized Linear Model (GLM) was employed to examine the predictive capabilities of the data. The report also includes an evaluation of model performance through statistical metrics, confusion matrix results, and a Receiver Operating Characteristic (ROC) curve analysis.

### **2. Exploratory Data Analysis (EDA)**

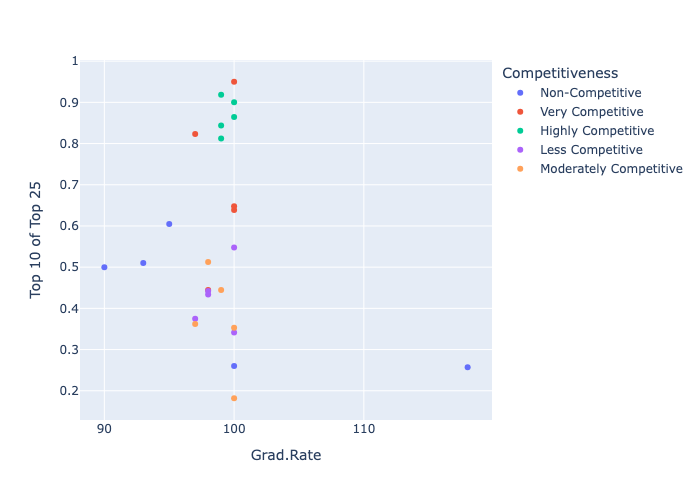
The EDA phase is crucial for understanding the data's structure, distribution, and potential anomalies. Key observations from the analysis include:

#### **2.1 Data Summary**

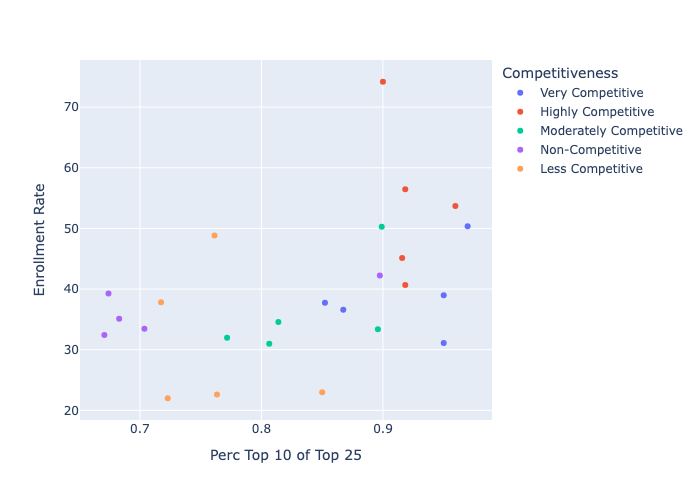
|  |  |
| --- | --- |
| **Column Name** | **Descriptive Stats** |
| Private | Yes 565 (0.73) |
|  | No 212 (0.27) |
| Apps | 3001.64 (3870.20) |
| Accept | 2018.80 (2451.11) |
| Enroll | 779.97 (929.18) |
| Top10perc | 27.56 (17.64) |
| Top25perc | 55.80 (19.80) |
| F.Undergrad | 3699.91 (4850.42) |
| P.Undergrad | 855.30 (1522.43) |
| Outstate | 10440.67 (4023.02) |
| Room.Board | 4357.53 (1096.70) |
| Books | 549.38 (165.11) |
| Personal | 1340.64 (677.07) |
| PhD | 72.66 (16.33) |
| Terminal | 79.70 (14.72) |
| S.F.Ratio | 14.09 (3.96) |
| perc.alumni | 22.74 (12.39) |
| Expend | 9660.17 (5221.77) |
| Grad.Rate | 65.46 (17.18) |

#### **2.2 Distribution of Features**

Graph 1: Graduation Rate vs Percentage of Top 10 of Percentage of students from Top 25 % from high school

The above graph shows that the Probability of the student graduating tends to increase with their standing in high school. Also we note that higher the competitiveness of getting into the college has more number of students belonging to top 10% of their high school batch and are thus more motivated to graduate from the college.

The below graph shows that as the competitiveness of getting into a college increases, the higher percentage of the top 25% of high school graduates are in the top 10% of their class.





The above graph shows that as the competitiveness of getting into a college increases, their enrollment rate also increases. Enrollment rate here is defined as number of students enrolled/ number of students accepted.

### **3. Regression Analysis**

Building on the EDA findings, a Generalized Linear Model (GLM) with a Gaussian family and identity link function was applied. The results are as follows:

* **Dependent Variable**: y
* **Number of Observations**: 543
* **Log-Likelihood**: 583.81
* **Deviance**: 3.7022
* **Pseudo R-squared**: 0.06945

Key coefficients and their significance levels:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Coefficient | Estimate | Std. Error | z-value | P>|z| | 95% CI [Lower, Upper] |
| Intercept | -0.1087 | 0.022 | -4.995 | 0.0 | [-0.151, -0.066] |
| x1 | 0.1099 | 0.026 | 4.195 | 0.0 | [0.059, 0.161] |
| x2 | 0. 0006 | 0.00 | 2.537 | 0.011 | [0.000, 0.001] |
| x3 | 0.0007 | 0.00 | 2.417 | 0.016 | [0.000, 0.001] |

These results indicate that all predictors significantly influence the dependent variable y, with x1 having the strongest effect.

### **4. Model Evaluation Metrics**

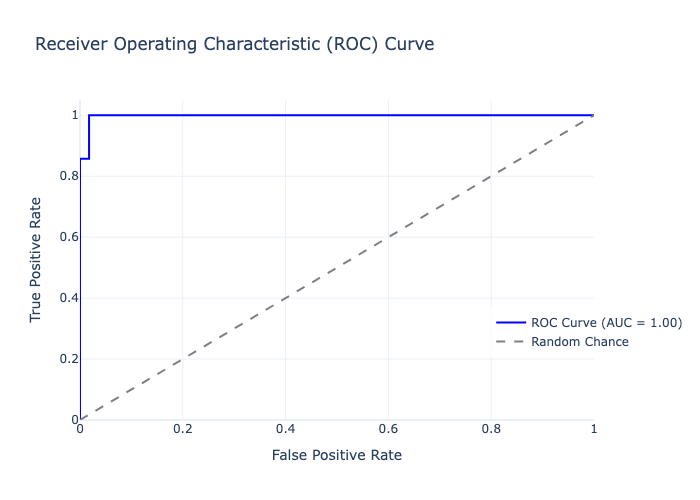
The predictive performance of the model was assessed using key metrics derived from the confusion matrix:

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Accuracy | 97% |
| Precision | 0 % |
| Recall | 0 % |
| Specificity | 100 % |

Accuracy indicates the overall correctness of predictions, while specificity measures the model's ability to identify true negatives. However, precision and recall are notably low, suggesting difficulties in identifying positive outcomes.

### **5. ROC Curve Analysis**

To further evaluate the model, a Receiver Operating Characteristic (ROC) curve was generated. The Area Under the Curve (AUC) value is 1.00, indicating excellent discriminatory performance.



### **6. Conclusion and Recommendations**

The analysis highlights the model's strengths in overall accuracy and specificity but underscores challenges in predicting positive outcomes, as evidenced by low precision and recall scores. To address these issues and improve the model's predictive power, the following steps are recommended:

1. **Addressing Outliers**: Apply robust statistical techniques or transformations to mitigate the influence of outliers, particularly in x3.
2. **Feature Engineering**: Introduce additional variables or transform existing ones to capture more variance in the dependent variable.
3. **Model Optimization**: Experiment with other models, such as decision trees or ensemble techniques, which may handle imbalances better.
4. **Resampling Strategies**: Use techniques like oversampling (SMOTE) or undersampling to balance the dataset and improve recall.
5. **Threshold Tuning**: Adjust the decision threshold to find a better balance between precision and recall.

### **References**

* James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer.
* Fawcett, T. (2006). *An introduction to ROC analysis*. Pattern Recognition Letters, 27(8), 861-874. <https://doi.org/10.1016/j.patrec.2005.10.010>

### Appendix

import pandas as pd

import numpy as np

import plotly.express as px

import statsmodels.api as sm

from sklearn.model\_selection import train\_test\_split

import plotly.graph\_objects as go

from sklearn.metrics import roc\_curve, auc

from sklearn.preprocessing import label\_binarize

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, confusion\_matrix

df = pd.read\_csv('College.csv')

summary\_stats = []

for col in df.columns:

if pd.api.types.is\_numeric\_dtype(df[col]):

mean = df[col].mean()

std = df[col].std()

summary\_stats.append((col, f"{mean:.2f} ({std:.2f})"))

elif col != 'Unnamed: 0':

value\_counts = df[col].value\_counts()

for value, count in value\_counts.items():

proportion = count / len(df)

summary\_stats.append((col, f"{value} {count} ({proportion:.2f})"))

summary\_df = pd.DataFrame(summary\_stats, columns=['Column Name', 'Descriptive Stats'])

summary\_df.to\_excel('summary\_stats.xlsx', index = False)

df.rename(columns={'Unnamed: 0': 'College Name'}, inplace=True)

df.columns

df['Acceptance Rate'] = (df['Accept'] / df['Apps'] ) \* 100

df['Enrollment Rate'] = (df['Enroll'] / df['Accept'] ) \* 100

enrollment\_rate\_df = df[['College Name', 'Acceptance Rate', 'Enrollment Rate']]

enrollment\_rate\_df['Competitiveness'] = pd.cut(enrollment\_rate\_df['Acceptance Rate'], bins = 5, right = False, labels = ['Highly Competitive', 'Very Competitive', 'Moderately Competitive', 'Less Competitive', 'Non-Competitive'])

# top\_enrollment\_rate\_df = enrollment\_rate\_df.sort\_values(by='Enrollment Rate', ascending=False).head(10)

fig = px.scatter(

enrollment\_rate\_df,

x = 'Enrollment Rate',

y = 'Acceptance Rate',

color = 'Competitiveness'

)

fig.write\_image('Enrollment Rate vs Competitiveness.png')

fig.show()

top\_10\_top\_25\_df = df[['College Name', 'Acceptance Rate', 'Top10perc', 'Top25perc', 'Enrollment Rate']]

top\_10\_top\_25\_df['Perc Top 10 of Top 25'] = top\_10\_top\_25\_df['Top10perc'] / top\_10\_top\_25\_df['Top25perc']

top\_10\_top\_25\_df['Competitiveness'] = pd.cut(top\_10\_top\_25\_df['Acceptance Rate'], bins = 5, right = False, labels = ['Highly Competitive', 'Very Competitive', 'Moderately Competitive', 'Less Competitive', 'Non-Competitive'])

top\_top\_10\_top\_25\_df = top\_10\_top\_25\_df.sort\_values(by='Perc Top 10 of Top 25', ascending=False).groupby(by='Competitiveness').head(5)

fig = px.scatter(

top\_top\_10\_top\_25\_df,

x = 'Perc Top 10 of Top 25',

y = 'Enrollment Rate',

color = 'Competitiveness'

)

fig.write\_image('Enrollment Rate vs Percentage of Top 10 of Percentage of Top 25.png')

fig.show()

grad\_rate\_df = df[['College Name', 'Top10perc', 'Top25perc', 'Acceptance Rate', 'Grad.Rate']]

grad\_rate\_df['Top 10 of Top 25'] = grad\_rate\_df['Top10perc'] / grad\_rate\_df['Top25perc']

grad\_rate\_df['Competitiveness'] = pd.cut(grad\_rate\_df['Acceptance Rate'], bins = 5, right = False, labels = ['Highly Competitive', 'Very Competitive', 'Moderately Competitive', 'Less Competitive', 'Non-Competitive'])

top\_grad\_rate\_df = grad\_rate\_df.sort\_values(by='Grad.Rate', ascending=False).groupby(by='Competitiveness').head(5)

fig = px.scatter(

top\_grad\_rate\_df,

x = 'Grad.Rate',

y = 'Top 10 of Top 25',

color = 'Competitiveness'

)

fig.write\_image('Grad Rate vs Percentage of Top 10 of Percentage of Top 25.png')

fig.show()

model\_df = df[['College Name', 'Grad.Rate', 'Acceptance Rate', 'Enrollment Rate','Top10perc', 'Top25perc']]

model\_df['Top10percOfTop25perc'] = model\_df['Top10perc'] / model\_df['Top25perc']

model\_df.drop(columns=['Top10perc', 'Top25perc'], inplace=True)

model\_df['Competitiveness'] = pd.cut(model\_df['Acceptance Rate'], bins = 5, right = False, labels = [1, 2, 3, 4, 5])

X = np.asarray(model\_df[['Top10percOfTop25perc', 'Grad.Rate', 'Enrollment Rate']])

y = np.asarray(model\_df['Competitiveness'].apply(lambda x: 1 if x == 1 else 0))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

X\_train = sm.add\_constant(X\_train)

model = sm.GLM(y\_train, X\_train)

results = model.fit()

results.summary()

X\_test = sm.add\_constant(X\_test)

y\_pred = results.predict(X\_test)

# Convert probabilities to binary predictions

threshold = 0.5

y\_pred\_class = (y\_pred >= threshold).astype(int)

# Calculate confusion matrix

tn, fp, fn, tp = confusion\_matrix(y\_test, y\_pred\_class).ravel()

# Calculate metrics

accuracy = accuracy\_score(y\_test, y\_pred\_class)

precision = precision\_score(y\_test, y\_pred\_class)

recall = recall\_score(y\_test, y\_pred\_class)

specificity = tn / (tn + fp) # True Negative Rate

# Print results

print(f"Accuracy: {accuracy:.2f}")

print(f"Precision: {precision:.2f}")

print(f"Recall: {recall:.2f}")

print(f"Specificity: {specificity:.2f}")

fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred)

roc\_auc = auc(fpr, tpr)

# Create the ROC curve plot

fig = go.Figure()

# Add ROC curve

fig.add\_trace(go.Scatter(

x=fpr, y=tpr,

mode='lines',

name=f'ROC Curve (AUC = {roc\_auc:.2f})',

line=dict(color='blue')

))

# Add diagonal line for random chance

fig.add\_trace(go.Scatter(

x=[0, 1], y=[0, 1],

mode='lines',

name='Random Chance',

line=dict(dash='dash', color='gray')

))

# Customize layout

fig.update\_layout(

title='Receiver Operating Characteristic (ROC) Curve',

xaxis\_title='False Positive Rate',

yaxis\_title='True Positive Rate',

xaxis=dict(range=[0.0, 1.0]),

yaxis=dict(range=[0.0, 1.05]),

legend=dict(x=0.8, y=0.2),

template="plotly\_white"

)

fig.write\_image('ROC Curve.png')

# Show plot

fig.show()