Predicting Graduate Admissions using Machine Learning

A project report submitted for the EECS 738 Machine Learning course.

Submitted by

Sudha Chandrika Yadlapalli (3097401)
Jagadeesh Sai Dokku (3097403)
Sai Soujanya Ambati (3096548)
Mahitha Reddy Bayyapu (3096546)
Apurva Idupulapati (3108999)

Under the esteemed guidance of

Dr. Zijun Yao

Department of Electrical Engineering and Computer Science
University of Kansas
Lawrence, KS 66045

Spring 2023

Chapter 1

Introduction

1.1 Motivation

The process of applying to graduate schools can be overwhelming and stressful for students. The admission process is highly competitive, and students often invest a significant amount of time, money, and effort into their applications. Unfortunately, the admission decision is not always transparent, and students may not receive clear feedback on why they were not accepted. Developing a machine learning model that can predict the likelihood of admission can be incredibly beneficial for students. The model can provide insights into the admissions process and help students make informed decisions about their future academic endeavors. By having a better understanding of their chances of admission, students can allocate their resources more effectively and make more informed decisions about where to apply, what programs to pursue, and how to improve their chances of acceptance.

1.2 Background of the problem

The problem is that the process of applying to graduate schools can be stressful, competitive, and lacking in transparency. This can make it difficult for students to make informed decisions about where to apply and how to improve their chances of acceptance. Developing a machine learning model that can predict the likelihood of admission can provide valuable insights into the admissions process and help students allocate their resources more effectively and make more informed decisions about their academic endeavors.

Chapter 2

Dataset and Formation

2.1 Dataset description

The dataset used for our project implementation was collected from Kaggle. Below is the link. https://www.kaggle.com/datasets/nitishabharathi/university-recommendation

The dataset consists of data related to graduate university admission of 45 different universities which have 14,798 unique values and 53000 data points with 25 input features and one target variable ("admit"). The input features include features such as the applicant's undergraduate college, major, research experience, industry experience, specialization, TOEFL score, GRE scores, and CGPA, among others. The target variable "admit" indicates whether the applicant was admitted to the university or not.

The dataset statistics show that the distribution of the target variable "admit" is imbalanced, with only 22% of the applicants being admitted to the university. The dataset has a mix of categorical and continuous features. Some of the categorical features include the applicant's major, specialization, and program, among others. The continuous features include the applicant's GRE scores, TOEFL score, CGPA, research experience, and industry experience, among others.

The dataset provides a good representation of the factors that could influence the admission of a student to a graduate university and can be used to build models for predicting whether an applicant would be admitted or not.

2.2 Visualization

We performed data analysis on the data set to learn more about the properties and relation among the features. This would help us in choosing the best fitting technique to build a model. Following are the brief findings from data analysis and visualization.

1. In order to identify missing values in our dataset, we utilized the 'missingno' library. As shown in Figure 2.1, we found that three features - 'gmatA', 'gmatV', and 'gmatQ' - contained only a

few entries. Given the small number of entries for these features, we determined that omitting them from the analysis would not significantly impact the admission decision-making process.

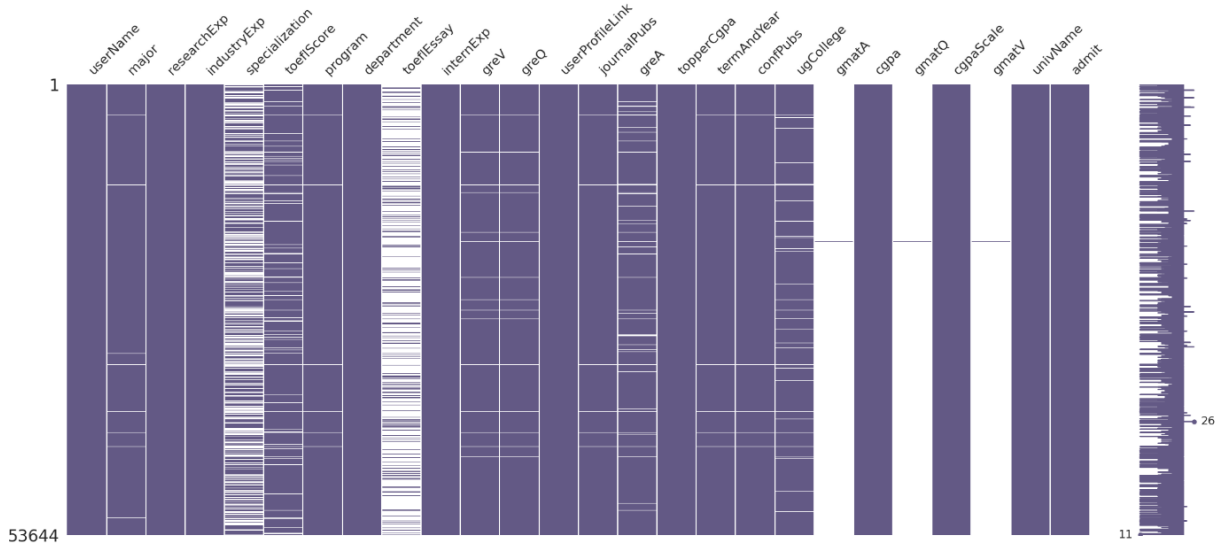

Figure 2.1- Missing values representation

2. As part of our data analysis, we split the 'termAndYear' feature into two separate entities: 'term' and 'year'. However, some values were missing in the 'year' feature. To address this, we employed the K-nearest neighbor (KNN) imputation technique. This method uses the values of the 'K' closest observations in the dataset to estimate the missing value in the 'year' feature. By doing this, the 'year' feature could be completed, and the dataset could be used for further analysis
3. To find the correlation between the variables we have used a heat map as shown in fig 2.2. From this we inferred that GRE verbal and GRE quantitative had strong positive correlation with total score. This means that higher scores on the GRE verbal and quantitative sections are associated with higher total scores. Cgpa and topperCgpa also show a good positive correlation.

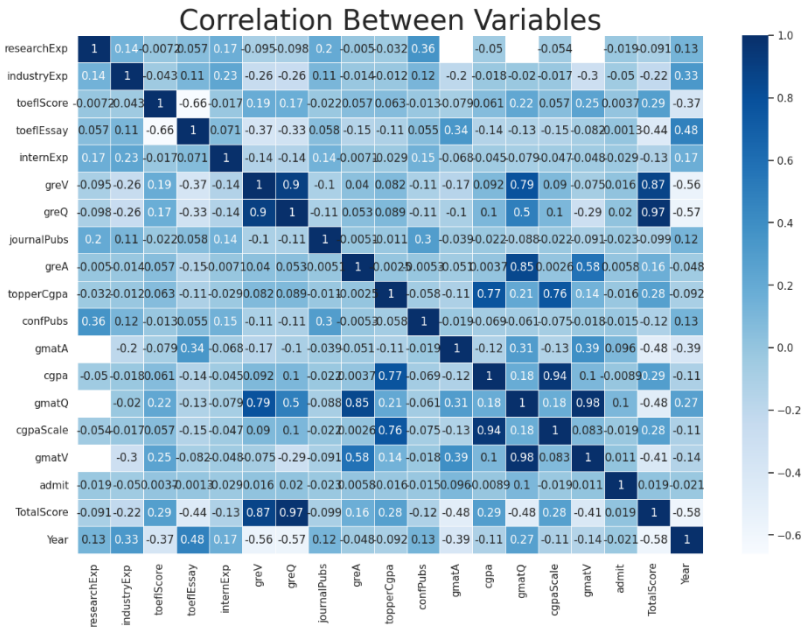

Figure 2.2- Correlation among features heat map

4. We also plotted histograms of a few features to understand the distribution of values. Fig 6.3 and Fig 6.4 are the histograms of topperscore and cgpa respectively. The histograms suggest that most students fall in the lower spectrum of scores. Number of students scoring higher Cgpa and Topper score are comparatively low. This implies that high academic performance is not very common in the dataset and is achieved by only a small number of students.

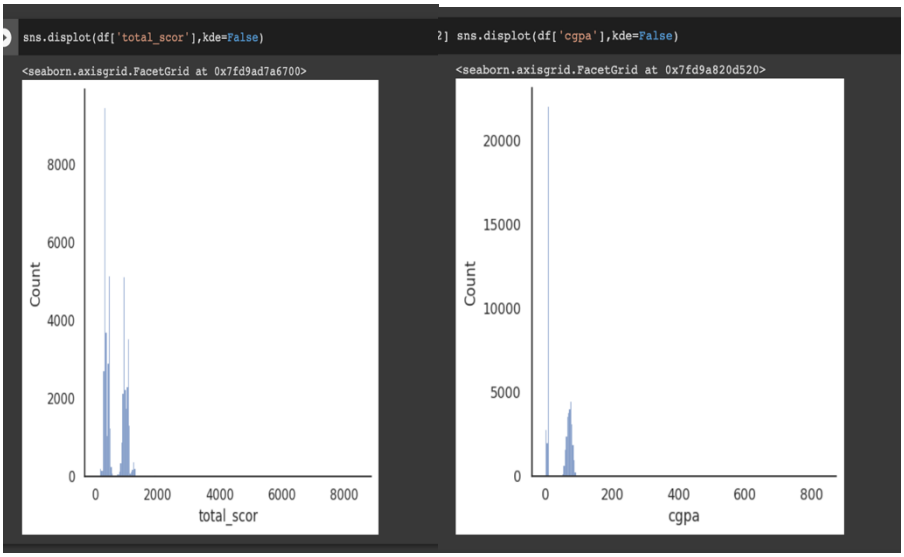

Fig.2.3 Histogram of total_score and Fig.2.4 Histogram of cgpa

5. KDE plot are particularly useful for visualizing the shape of a distribution, as they can reveal features such as skewness, multimodality, and the presence of outliers. Unlike histograms, which bin data into discrete intervals, KDE plots estimate the underlying distribution of the data using a continuous, smooth curve. Figure 6.5 shows the KDE plot between admit and year.

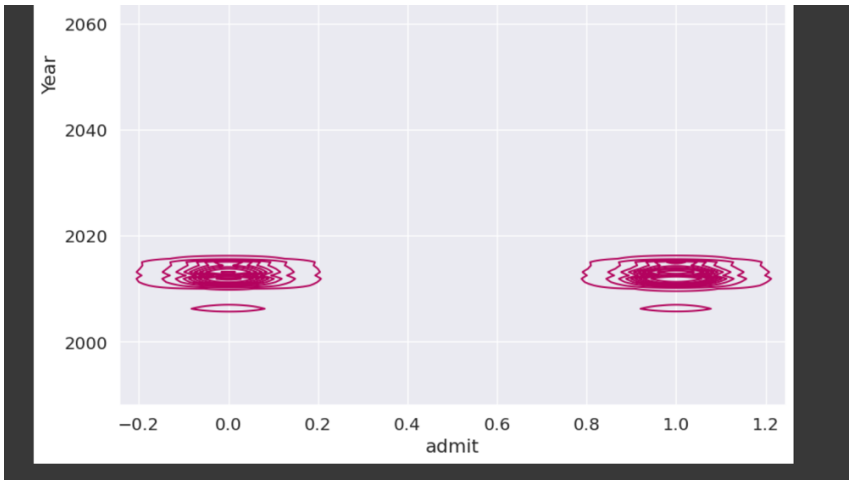

Figure 2.5 KDE plot between admit and year

6. We used Box plots for summarizing the central tendency, variability, and skewness of a dataset, and for identifying potential outliers.

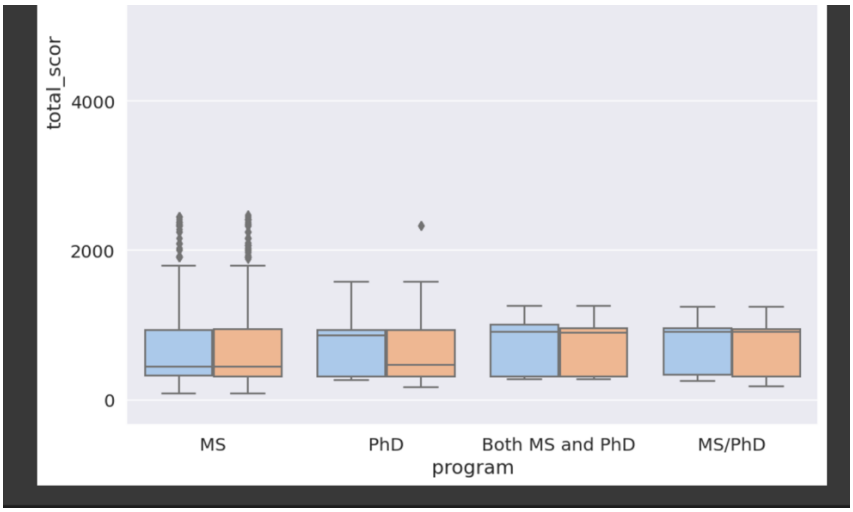

Figure 2.6 Box plot

Chapter 3

Framework

3.1 Input and Output of the framework

The Input of our framework consists of username, major, researchExp, industryExp, specialization, toeflScore, program, department, toeflEssay, internExp, greV, greQ, userProfileLink, journalPubs, greA, topperCgpa, termAndYear, confPubs, ugCollege, gmatA, cgpa, gmatQ, cgpaScale, gmatV and univName

The output of my framework is the "admit" column, which indicates whether the applicant was admitted to the graduate program. The prediction generated by the model will be a binary value, either 0 or 1, representing the probability of admission based on the input features.

The data is split into training and testing sets using a 80/20 split. The evaluation metrics used are accuracy, confusion matrix, recall, precision, and F1 score.

3.2 Data Preprocessing

Preprocessing in machine learning is an important step because the quality of the data used in machine learning algorithms directly affects the accuracy of the output. We have used the following preprocessing techniques which help us build accurate models.

1. Data cleaning: The following columns were dropped from the dataset: 'gmatA', 'gmatQ', 'gmatV', 'userProfileLink', 'termAndYear', 'cgpaScale', 'journalPubs', 'toeflEssay', 'specialization'. These columns were not relevant to the analysis.
2. Missing value imputation: The KNNImputer was used to impute missing values in the numerical columns (toeflScore, internExp, greV, greQ, greA, topperCgpa, TotalScore). The SimpleImputer was used to impute missing values in the categorical columns (major, department, ugCollege, season, program, confPubs).
3. Handling categorical variables: We then convert categorical variables into numerical variables using one-hot encoding.
4. Data summarization: Finally, we summarize the data by grouping and aggregating the values.

Pivot tables are also used to show the relationship between variables.

3.3 Models Utilized

In this project, we used seven different machine learning models to predict the admission of a student to a graduate school. The models used are as follows:

1. Naive Bayes: It is a simple probabilistic model that works on the principle of Bayes' theorem. This model is fast and can handle large datasets easily.
2. Logistic Regression: It is a popular linear model used for binary classification problems. It is easy to interpret and can provide the probability of admission for a student.
   "We have developed our own logistic regression algorithm instead of using a built-in function for better optimization"
3. K-Nearest Neighbors (KNN): It is a non-parametric algorithm that classifies data based on the similarity between the input features. This model is suitable for large datasets and can handle noisy data.
4. Support Vector Machine (SVM): It is a powerful algorithm that works well for high-dimensional datasets. It is especially useful when the decision boundary between classes is non-linear.
5. Decision Tree: It is a tree-like model that classifies data by recursively splitting the data based on the most significant features. It can handle both numerical and categorical data.
6. Random Forest: It is an ensemble model that uses multiple decision trees to make predictions. It is useful for reducing the risk of overfitting and can handle large datasets.
7. XGBoost: It is an advanced implementation of gradient boosting that is designed for speed and performance. It can handle missing data and provides better accuracy than traditional gradient boosting.

We chose these models because they are widely used and have shown good performance in similar classification tasks. By using a variety of models, we can compare their performances and choose the one that gives the best results for our dataset.

Chapter 4

Experiments

In this project to obtain best results we tried training with multiple models. And decided to go with ensembled approach. Ensemble methods can be powerful tools for improving the performance of machine learning models, particularly when individual models have complementary strengths and weaknesses.

4.1 Training Process

To build an effective machine learning model, it is important to go through a structured process. This process includes preprocessing and visualization of the data, data splitting into training and testing sets, model selection based on the characteristics of the data and the problem being solved, model training and evaluation using metrics such as accuracy, confusion matrix, recall, precision, and F1 score, and finally model selection and tuning based on the best performing model. This technique ensures that the model is not overfitting the data, can generalize well to new inputs, and is optimized for performance.

Chapter 5

Analysis and Evaluation

5.1 Analysis

Based on our evaluation results, it appears that some of the models may be suffering from overfitting, particularly the Decision Tree (DT) and Random Forest (RF) models. This is because they both have very high training accuracies of 99.27% and 99.86%, respectively, but their testing accuracies are not as high, with the DT model having a testing accuracy of 70.95% and the RF model having a testing accuracy of 73.72%.

In the case of the DT and RF models, some possible ways to address overfitting could be to reduce the maximum depth of the tree, increase the minimum number of samples required to split a node, or

use a smaller number of trees in the forest. Additionally, cross-validation could be used to better assess model performance and prevent overfitting.

We have used XGB to get better results for our data set where we got better F1score of about 80.99 and the recall and precision values are also high. We have also implemented ensemble model which is voting classifier added weights to obtain better overall evaluation with our models.

5.2 Evaluation

In our project, we utilized several evaluation metrics to measure the performance of our classification models. These included accuracy, precision, recall, F1-score, and confusion matrix. Accuracy measures the proportion of correctly classified instances, precision measures the proportion of true positives among all predicted positives, recall measures the proportion of true positives among all actual positives, and F1-score is the harmonic mean of precision and recall. The confusion matrix is a table that shows the number of true positives, true negatives, false positives, and false negatives. Scikit-learn library functions were used to calculate all of these metrics in the code. The results are discussed in detail in chapter 6.

Chapter 6

Results and Conclusion

6.1 Results

Based on the evaluation metrics of the different models, it can be concluded that the XGBoost model performs the best with a testing accuracy of 75.57%, followed by the Random Forest model with a testing accuracy of 73.72%. The Naive Bayes model has the lowest testing accuracy of 39.05%.It is also observed that some models perform better in terms of recall, while others perform better in terms of precision. For example, the Support Vector Machine (SVM) model has a high recall of 86.71%, but a lower precision of 70.49%. On the other hand, the Random Forest model has a lower recall of 88.35%, but a higher precision of 74.09%.

In the end, when selecting a model, it is critical to consider the trade-off between recall and precision, depending on the specific requirements of the problem. It is also suggested that the chosen model be

further optimized to increase its performance. Detailed values are listed in the following tables 6.1 and 6.2

Models	Training Accuracy	Testing Accuracy	Confusion Matrix	Precision	Recall	F1 Score
Naïve Bayes	51.51%	39.05%	[1054 77] [1727 102]	56.98%	5.58%	10.16%
Logistic Regression	80.41%	68.07%	[584 547] [398 1431]	72.35%	78.24%	75.18%
KNN	74.17%	60.91%	[398 733] [424 1405]	65.72%	76.82%	70.83%
SVM	78.74%	69.36%	[467 664] [243 1586]	70.49%	86.71%	77.76%
Decision Tree	99.27%	70.95%	[695 436] [424 1405]	76.32%	76.82%	76.57%
Random Forest	99.86%	73.72%	[566 565] [213 1616]	74.09%	88.35%	80.60%
XGB Classifier	87.02%	75.57%	[697 434] [289 1540]	78.01%	84.20%	80.99%

Table 6.1- Model evaluation

Ensemble Model	Training Accuracy	Testing Accuracy	F1 Score
Voting Classifier	95.22%	75.07%	74.32%

Table 6.2- Ensemble Model evaluation

6.2 Related Work

Previous work related to our project is in the references section [1] of this report. In their work, they have implemented a similar study on a small dataset of about 500 data points.

For our study, we initially researched different datasets. And after conducting an extensive search, we came across a larger dataset with over 53,000 data points. This larger data set provided us with a greater amount of data to train our models on, and ultimately led to more accurate results.

In addition to using existing machine learning models, we also implemented our own logistic regression model to further refine our predictions. By developing our own model, we were able to tailor it to the specific needs of our project and achieve even better results.

Overall, our experience in selecting and utilizing datasets for this project reinforced the importance of conducting thorough research and being open to alternative options. This allowed us to develop a more robust and accurate model that could better serve our intended application.

6.3 Challenges

Working on this project with 50,000 data points proved to be a challenging task. Due to the size of the dataset, understanding it and preprocessing it required a significant amount of time and effort. We had to experiment with multiple techniques to handle missing values, address categorical variables, and summarize the data effectively. Despite these challenges, our efforts ultimately paid off, as we were able to significantly improve the performance of our model

6.4 Conclusion

In this project, we developed a graduate admission prediction system using machine learning techniques on a dataset of 53,000 data points. By leveraging data visualization, we were able to gain insights into the data distribution and apply appropriate preprocessing methods. We experimented with seven different classification models, including Naive Bayes, Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Decision Tree, Random Forest, and XGBoost. We have achieved a training accuracy of 98.86% and a testing accuracy of 73.72% using Random Forest approach.

While the use of multiple models improved our accuracy, it also increased our runtime, which could hinder the model's performance as the dataset size grows. To address this issue in future work, we aim to implement more efficient processing methods.

Overall, this project demonstrated the potential of machine learning techniques for predicting graduate admissions and highlighted the importance of appropriate data visualization and preprocessing in achieving accurate results. With further development, this model could be a valuable tool for universities and admissions committees seeking to streamline their application processes.

Google Colab Project Link:

https://colab.research.google.com/drive/1FGCZXYYt6aAxWsuiSLm_OqbLyusekCmQ?usp=sharing

References

[1] Mohan Sacharya. (2018, Dec 28). Graduate Admission 2. Kaggle. https://www.kaggle.com/datasets/mohansacharya/graduate-admissions

[2] Mohammed Ayad. (2022, Oct 13). Graduate Admission 2. Kaggle. https://www.kaggle.com/code/mohammedayad/project-classification

[3] Nithya. (2022, May 1). University Classification. Kaggle. https://www.kaggle.com/code/nitishabharathi/university-classification

Contributions

Team Member	Student ID	Contribution
Sai Soujanya Ambati	3096548	Preprocessing, Milestone report
Apurva Idupulapati	3108999	Preprocessing, Milestone report
Mahitha Reddy Bayyapu	3096546	Visualization, Final Report
Sudha Chandrika Yadlapalli	3097401	Models and Evaluation, Final Report
Jagadeesh Sai Dokku	3097403	Models and Evaluation, Final Report