GRADUATE ADMISSION ANALYSIS AND PREDICTION

Kanshuai Wang
NetID: kw2606

Junjia Wang
NetID: jw5703

Member responsible for Uploading Submissions
Kanshuai Wang

March 12, 2020

ABSTRACT

With more and more students pay attention to the emerging programs, it is practical significant to analyze the relationship between Chance of Admit and other applying requirements with a limit dataset. We would set models with both regression methodology and classification methodology to predict our target feature. In addition, we would compare our models and tuning parameters in several ways to choose a most robust model. Finally, this model could be used for students to get their Chance of Admit for a particular program given other features.

1 Introduction and Motivation

1.1 Motivation

As graduate students, we still remember the difficulty to choose the graduate programs from a bunch of them and the uncertainty of getting admissions. So, our group want to find, for a particular program and university, which features have more significance and predict the rate of admission with the features provided by a particular student. In this way, students who are planning to pursue graduate degrees can fit the codes and conclusion from this project to their program choices with the public datasets. Also, the workers of programs can figure out their unconscious preference for the candidates to improve their admission system.

1.2 Problem Description

We got the admission data of IEEE International Conference on Computational Intelligence in Data Science 2019 from Kaggle. An instance of the dataset includes all information for applying the program: GRE Scores, TOEFL Scores, University Rating, Statement of Purpose, Letter of Recommendation Strength, Undergraduate CGPA, Research Experience, and Chance of Admit.

The main problem we want to deal with is finding the best prediction models to estimate the target feature (Chance of Admit) in our case. Also, we'd like to find the relationship

between features and their correlations with the Chance of Admit, figuring out which features are more important.

1.3 Difficulty

After loading the data, we found that the size of our data is pretty small. However, it is hard to get more relative data because the feature of this kind of problems. With the development of all science fields, more and more new programs is coming out to replace some outdated ones, and we believe that this trend would last for a long time. Consequently, this project would emphasize on the methodology and comparing the algorithms with such data size. For methodology, we plans to separate it to two parts, regression and classification. In regression models, we will use the Linear Regression, K-Nearest Regressor, Decision Tree Regressor, and SVM Regressor; in the classification models, we would try the Decision Tree, Random Forest, Logistic Regression, K-Nearest Neighbors, SVM, and XGBoost.

Another two challenges for us is how to find the best parameters for every algorithms and how to compare them to each other.

2 Methodology

2.1 Data Cleaning and Exploration

Firstly, we load our data in the notebook and check what features we have. Then, we drop the Serial No. column because it is useless when we have index in the Data Frame. Next, we checked that there is no missing values or wrong value in our dataset and the size of our data is 500. This data size is pretty small, comparing to other datasets used in other analysis. However, just like we discussed in the last section, it is a common circumstance to set a model with a such small scale dataset when we have a new program.

Our features have different magnitudes which can be found in figure1. For example, the largest GRE Score is 320 but the range of the University Rating is only 1-5. These features in different scales would have an influence on the results of our models, especially for the distance based methodology, such as KNN. So, we standardized our features in the same scale, and the figure2 shows boxplots of our features after standardizing. According to these boxplots, there is only one outlier value in the LOR feature which won't affect our result in an obvious scale. Then, we look at the correlations between every feature, particularly, the correlation between our target feature and others. According to the figure3, we can learn that Research actually have no visible relationship with Chance of Admit. But, considering there are only seven features left, we decide maintain this feature no matter if it is helpful for our prediction.

We are going to fix the target feature, Chance of Admit, a little bit to fit in the classification models. In order to make the classification models are comparable to the regressions, we'd like to separate the Chance of Admit to 4 classes (Excellent when Chance of Admit >0.9 as label 3, Good when in the range (0.7, 0.9] as label 2, Ordinary when (0.5, 0.7] as 1, and Poor as 0 when [0, 0.5]). After relabel, the histogram in figure4 shows that the relabeled target seems relatively balanced enough to deal with.

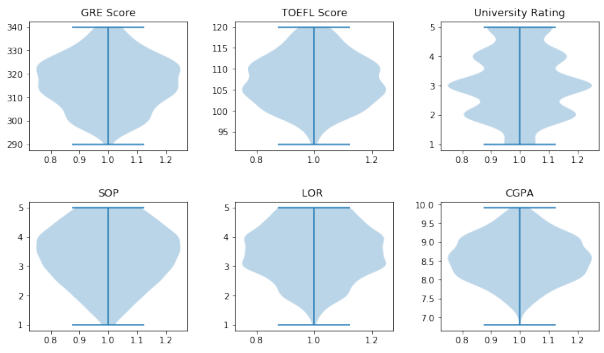

Figure 1: Violin Plot for Features

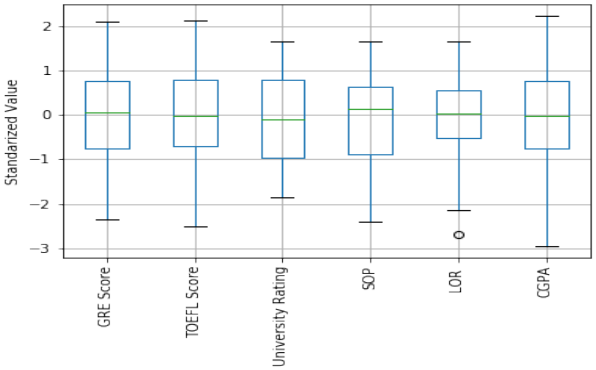

Figure 2: Boxplot after Standardize

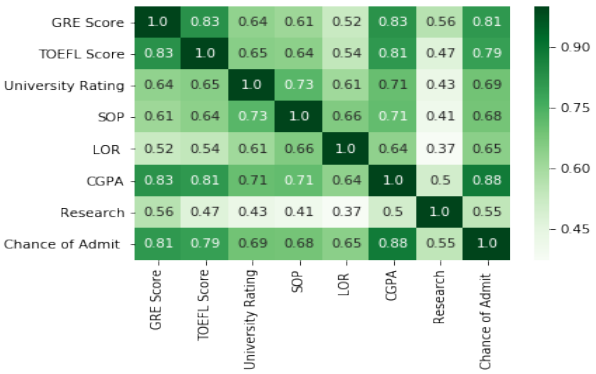

Figure 3: Correlation Heatmap

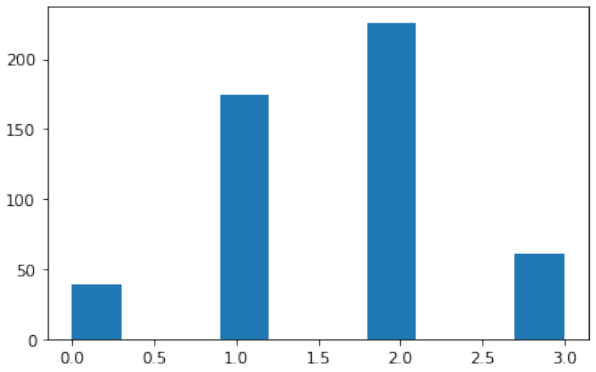

Figure 4: Histogram of Relabeled Target

2.2 Regression Methodology

In the regression part, we use the original target values rather than the relabeled ones. Even though classification problems are more common in data science than regression, we believe that the regression methodology is very suitable for our project because the numerical features and continuous target variable. Therefore, we are going to use four regressors in this part: Decision Tree, Linear Regression, K-nearest Neighbors, and SVM.

There are two particular designs we deem important when handling the regression. One is the cross validation, and the other is model selection function. Cross validation can keep our model from overfitting or underfitting by randomly splitting data into k folds, fitting and testing several times to get a reliable accuracy of results[4]. Model selection is a function built in the sklearn package which can fit different models in one code line and easier for reader to understand. After running the codes, we get the accuracy rates for different models (0.7961 for Linear Regression, 0.5643 for Decision Tree, 0.6643 for SVM, and 0.7414 for KNN).

2.3 Classification Methodology

Firstly, we use six classification algorithms in this part, Decision Tree, Random Forest(RF), Logistic Regression(LR), K-Nearest Neighbors(KNN), and XGBoost. For getting a stable accuracy rate for every model, we make a 1000 times 'for loop' to randomly split our data into training and validation with ratio 3:1, and fit our models in different training data and get the scores with validation. Then, we get the mean scores of 1000 times, the model ranking with the accuracy score shows in figure5.

In classification part, we also have two particular designs which can help a lot for this project. The first one is the utilize of AUC and the way to calculate it in multi-class problems. The selection of evaluation metric is closely related to the deployment of models in practice. Not only accuracy of models must be considered, but also real-life meaning should be evaluated. Hence, most commonly used evaluation metrics for classification problem is Area

Under the ROC Curve(AUC). AUC clearly explains how much model is capable of distinguishing among classes. A classifier with higher AUC indicates that it can have more true positives with the same number of false positives, and therefore more accurate the model is. Since we have four unique class labels in our target variable. In order to calculate AUC score, we need to apply the One-vs-All label binarizer to our target values[3]. In figure6, we have the final model ranking with average AUC scores of 1000 times.

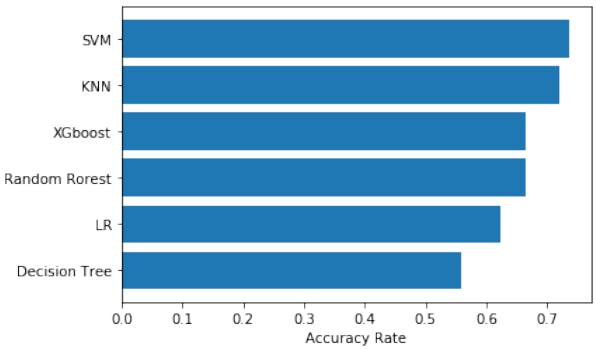

Figure 5: Model Ranking with Accuracy Rates

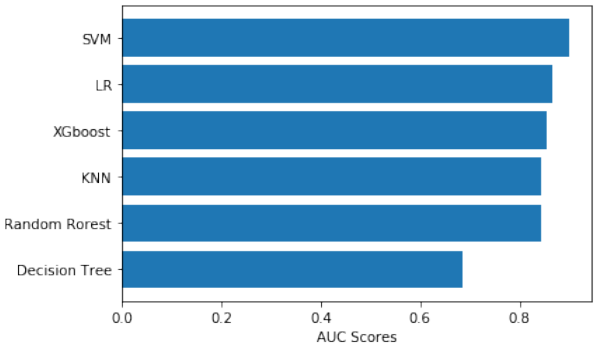

Figure 6: Model Ranking with AUC

The second design is tuning parameters for each classification models. As we got from figure5 and figure6, the Decision tree have a poor performance in both ranking. So, we only tuning parameters for other models to get a better result. Grid-search is used to find the optimal hyper-parameters of a model which results in the most 'accurate' predictions[2]. In addition, the grid search uses cross validation method as one of its basement, which means, we can correct the overfitting or underfitting results we got before in last section.

Tuning Parameters for Logistic Regression: With default settings of logistic regression in sklearn package, we use the combination of penalty ['L1','L2'] and regularization [0.001,.009,0.01,.09,1,5,10,25] to train the model.

Tuning Parameters for Random Forest: The combination of parameters are 'criterion' ['gini','entropy'], min samples split [2,3,4,5,6], and min samples leaf' [1,2,3,4,5].

Tuning Parameters for KNN: K nearest neighbors classifies a sample based on the class of its k nearest neighbors, so the only parameters we would change is the number of k so that we can find the best AUC in this algorithm[5]. The k number we tried are [1,2,3,4,5,6,7,8,9,10].

Tuning Parameters for SVM: For SVM model, we tried the combination of parameters as following, 'C' [0.001,.009,0.01,.09,1,5,10,25], 'kernel' ['linear', 'poly', 'rbf'], and 'degree':[3,6,9,12,24]

Tuning Parameters for XGBoost: Minimum sum of instance weight (hessian) needed in a child. The larger "min child weight" is, the more conservative the algorithm will be. Since we want to make sure that the algorithm should be robust, we have tried value range from 0 to 10. Gama is the minimum loss reduction required to make a further partition on a leaf node of the tree, and we used the gama set [0.5,1,1.5,2,2.5]. The last tuning parameter is the maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit[1]. We try out small values like [3, 4, 5, 6, 7].

After tuning parameters, we get our final model ranking with the accuracy score in figure7.

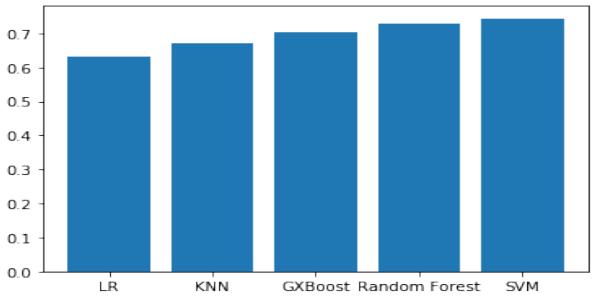

Figure 7: Model Ranking after Tuning Parameters

3 Results

After all works, we got the best regression model which is Linear Regression(accuracy rate is 0.7961), and the best classification model is SVM with the linear kernel(accuracy rate is 0.744). Also, we compute the AUC of the SVM with linear kernel is 0.9037, which is in a pretty high level. Therefore, we believe that we have solved our project with a great result. The figure8 shows the comparison between the randomly chosen prediction versus the real Chance of Admit. Except some extreme values, the linear regression model performance very good. And we also think that the linear regression model is a better model comparing with all the classifications. For getting this, We classify the predictions in the linear regression model in the same way as the classifications, and calculate an average accuracy rate like 0.78, which is higher than the SVM classification. In addition, the top four important features to decide the Chance of Admit are CGPA, GRE, LOR, and TOEFL.

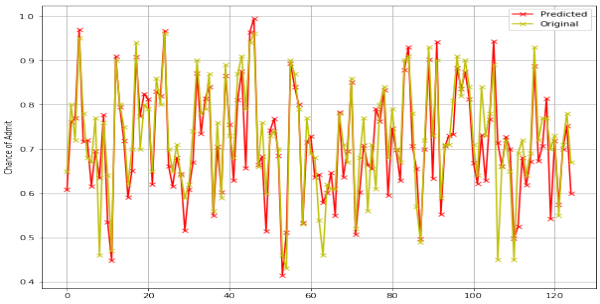

Figure 8: Prediction vs. Real Data

4 Discussion

The result we got is good enough that no more improvement we can think out about methodology. Nevertheless, some limitations indeed existence in this case. The first limitation is data size because the number of students who will apply for a specific program every year is often small. Also, it is not reasonable to combine the data from different programs. So, when the data can be accumulated in a long-term, the model would be more comprehensive. The second one is the Letter of Recommendation(LOR) and Statement of Purpose(SOP) is difficult to get scores by students themselves. To solve this, we have an idea to make a automatic score system based on NLP techniques in the future.

5 Conclusion

In conclusion, we success to touch the primary goal we set in the beginning to find a suitable model and predict the Chance of Admit given other features. In this process, we set both regression models and classification models with several algorithms. Furthermore, we utilized cross validation, model selection function, One-vs-All binarizer, and Grid Search to compare models and find the best parameters. Finally, we choose the Linear Regression as our best model; the best accuracy we get is about 0.8.

In the future, we want to add the NLP techniques to score the LOR and SOP automatically in our model. Also, making our models select the best algorithm automatically to fit in different dataset is absolutely the future direction.

References

[1] Aarshay Jain. *Complete Guide to Parameter Tuning in XGBoost with codes in Python*. URL: https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/. (Published: 3.1.2016).

[2] Rohan Joseph. *Grid Search for model tuning*. URL: https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e. (Published: 12.29.2018).

[3] Eric Plog. *AUC ROC Curve Scoring Function for Multi-class Classification*. URL: https://medium.com/@plog397/auc-roc-curve-scoring-function-for-multi-class-classification-9822871a6659. (Published: 7.9.2018).

[4] Sanjay.M. *Why and how to Cross Validate a Model?* URL: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f. (Published: 11.13.2018).

[5] Tavish Srivastava. *Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm (with implementation in Python R)*. URL: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/. (Published: 3.26.2018).