Logistic Regression Model in U.S. College Admission Result Forecast

Yunyi Ding
Barstow School of Ningbo campus
Email:
irisding20020213@outlook.com

Abstract: Logistic Regression is a type of classification algorithm involving a linear discriminant. Logistic Regression model measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logit/sigmoid function. Unlike linear regression, logistic regression does not try to predict the value of a numeric variable given a set of inputs. Instead, the output is a probability that the given input point belongs to a certain class. With the help of the logistic regression model, in this paper, we will create a program to predict the admission decision for U.S. graduate school of Chinese applicants.

I. Introduction:

A. Background Information

According to NAFSA(Association of International Educators), there are more and more students study abroad, especially for Chinese students, who constitute more than 50% of all the international students (NAFSA, p. 4) . Graduate school admission has always been a mysterious puzzle, where the tools of assessment, the competitive nature of the school itself, the applicant pool and the taste of the admission committee can all play a significant role in the decision process. This complexity is exacerbated by the exocity: due to the difference in the education system, particularly the scoring mechanism and admission criteria, foreign applicants have had a hard time in application material preparation and accurate self-assessment. Therefore, the admission decision sometimes seems random and chaotic and students have always been concerned about it. In order to address this problem and find some patterns in predicting the final results, we try to leverage the power of data science and use the logistic regression model to seek for predictive indicators.

Logistic regression model is a standard method of data analysis that was first proposed in 1970. When we handle binary results, logistic regression model can do a great job as it can overcome the limitation of ordinary least squares (Ahmed, 2017). The model can be used in many aspects of life including medical studies and researches. Their common purpose is to formulate models that sorting whether or not an outcome happens.

Logistic regression, like many other predictive models, uses the relationship between the dependent variable and one or more independent variables to estimate the probability of a certain class (event) existing. The reason why we choose logistic regression, a classification

model, to solve this problem is that the admission result is dichotomous (binary), i.e. admitted or denied ("1" or "0"). Also, though not a direct linear relationship between the dependent and independent variable, it makes intuitive sense that the admission decision is positively or negatively correlated with some factors such as GPA and standardized test scores. In logistic regression, the logarithm of the odds (a.k.a. log-odds) for the dependent variable labeled as "1" is a linear combination of one or more independent predictors. And the independent variables in logistic regression will be either continuous or binary (has value only 1 or 0). Using the sigmoid function (logistic function), the log-odds can be converted to the actual probability, as shown below:

$$\ln\frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_n x_n = F$$

$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_n x_n} = e^F$$

$$p = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_n x_n}}{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_n x_n} + 1}$$

$$p = \frac{1}{e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_n x_n)} + 1} = \frac{1}{e^{-F} + 1}$$

Here, p is the probability for the dependent variable being "1", $x_1, x_2 \ldots x_n$ are the independent variables that will be used to predict the value of p. $\beta_0, \beta_1, \beta_2 \ldots \beta_n$ are the coefficients that the logistic regression model aims to measure to produce the prediction.

According to Kassambara (2018), for the logistic model to work properly, we need to satisfy several assumptions.

1. The outcome is a binary or dichotomous variable like yes vs no, positive vs negative, 1 vs 0. Here our raw data has more than 2 categories of results, and later we will categorize them into 2 categories, i.e. whether get admitted (1) or not (0).
2. The input variables are continuous or binary (represented by a 0 or 1).
3. There is a linear relationship between the logit of the outcome and each predictor variable.
4. There is no influential values (extreme values or outliers) in the continuous predictors
5. The independent variables, as implied by the name, will be independent with each other (no multicollinearity)

The first assumption is straightforward, and for the rest, we will need to process the data and validate them once moving to the data step.

A. Input And Output Data Information

- *Independent Variable*

Original Variable Name	Original Format	Data Transformation	Final Variable
GRE	Applicant's GRE score. A rich text content that records each score with its component sub scores	The final format will be four columns of numerical data that contains the total GRE score, the GRE Verbal score, the GRE Quantitative score, and the Analytical Writing (AW) score. Null values will be replaced with the average score of that particular subject*.	GRE_all_filled GRE_V_filled GRE_Q_filled AW
TOEFL	Applicant's TOEFL score. Similar format of GRE score, rich text that shows all the components of TOEFL grades altogether	The final version of the numerical score contains five columns of the reading, speaking, listening, writing and the total scores. Null values will be replaced with the average score of the respective section.	TOEFL_all TOEFL_L TOEFL_R TOEFL_S TOEFL_W
app_school	String variable. The raw dataset lists all the applying school names in alphabetic order (352 schools in total after data clean, before have thousands of variations)	We rearrange the universities by introducing the external dataset about universities ranking in year 2013-2016 with U.S. News rank, and group schools into six rank bins: Top 10/ 10-20/ 21-30/ 31-40/ 41-50/ other	rank_bin→ top10 10_20… 41-50 other
degree	String variable. The degree the applicant applies. There are 86 unique Degrees names in the original data	There are too many unique degrees with small sample size. Therefore, we rearrange the data into 7 top categories ranked by frequency and leave the rest as category "others"	degree_bin→ MA MS PhD … MLA

app_maj or	String variable. There are 1442 unique values of applying majors with both Chinese and English names and related variations.	We did data cleaning, and then group majors into bins based on similarity (total 18 bins), e.g. we turned Economics and International Economics Relationship into a broader category “economics”. Besides, we turned major with Chinese names into English names.	App_major Business Economics Biology Communication Law … Public Health
year	Numeric variable. Application year	We get rid of the outliers in the year 2011 and 2012.	year
ug_GPA	Combination of string and float values. Candidate GPA in different formats and scales, such as AA, S…100-scale, 5-scale and 4-scale.	Convert different GPA formats into one comparable numerical format, which is between 0 and 1.	ug_gpa_mapping
ug_level	Applicant’s undergraduate school rank in terms of “project”	We introduced data from Ministry of Education of PRC to map different Chinese schools into corresponding projects (4 categories in total)	ug_level_mapping 985&211 211 other oversea Ug_level mapping

- Additional Data Source

Data Source	Related variable	Functionality
U.S.NewsTop 50 universities	app_school	Group application schools (unique: 352) into 5 different ranking groups.
985 &211 school list	ug_level	Map all applicants’ undergraduate schools into

		border category: 985&211, 211, oversea or others.
GRE conversion	GRE_old_nt GRE_new_nt	Convert old GRE grades into new GRE grades.
Major mapping	ug_major	Turn unique and specific majors into a border major category. E.g.Urban Design and Landscape Architecture grouped into “Architecture.”
Ug_GPA mapping	ug_gpa_mapping	Convert different fashions of GPA all into 0-1 range (%). e.g.: 4-scale GPA and AA,S Law GPA into % GPA.

Variable name	Range	Meaning
y_predict	1 or 0	Same format as the dependent variable: given unseen applicants and their info, the model will output 1 or 0 as whether they will get admitted to the school they apply or not.

II. Data Processing

In this part, we processed the raw data collected through Internet by dealing with the outliers, imputing missing values, using one-hot encoding to transform variables and resolving issues specific to this model. The reason we need this step are 1) wrong data would influence the overall accuracy of the model and 2) not all variables are readily readable to the model.

A. Outlier Detection & Processing

When we collect data, outliers may appear. Unlike support vector machine (SVM), Logistic Regression result will be heavily impacted by the outliers (as all data points contribute). If any independent variable in the regression model involves the wrong data, the probability can change hugely.

For example, we dropped all GRE total score lower than 250 and Verbal score lower than 120 since they are unlikely to happen in reality and probably out of data entry error.

```
data_full.drop(data_full[data_full['GRE_all']<250].index,axis=0,inplace=True)
data_full.drop(data_full[data_full['GRE_V']<120].index,axis=0,inplace=True)
```

Graph2.1

B. Null/Missing Value Imputing

Since logistic regression model involves multiplication and addition operations of independent variables, null values cannot be applied into the model directly. Hence it needs to be handled properly. We processed null values in two steps.

First, we dropped the rows with all possible empty entries as null with the idea that this problematic and incomplete data provides limited amount of information. We will not drop entries with fewer null values though (i.e. not all null, just some). If we do so, we would lose half of the data points that can be useful for modeling. Instead, we impute null values with their column averages, respectively, as the average number is an easy-calculated representative proxy for non-skewed data with no outliers. For example, N/A values exist in GRE scores. We processed those values by filling them up with the mean values of their respective section score columns. Secondly, for the categorical null values, we created an "other" category to account for it. Some of the examples can be seen here:

```
# drop the row if all possible null entries are null
data_full=data_model.dropna(thresh=7) # same as data_full=data_model.dropna(how='all')
```

Graph2.2

```
# fill N/A score with average score,respectively. However, if the original score is NA, do NOT fill it (JD & LLM student)
gre_not_all_null=(~data_full['GRE'].isnull())
oor_all=(data_full['GRE_all']>340)|(data_full['GRE_all']<170)
oor_q=(data_full['GRE_Q']>170)|(data_full['GRE_Q']<100)
oor_v=(data_full['GRE_V']>170)|(data_full['GRE_V']<100)

data_full['GRE_all']=np.where(gre_not_all_null&((data_full['GRE_all'].isnull())|oor_all),data_full.GRE_all.mean(),data_full.GRE_all)
data_full['GRE_Q']=np.where(gre_not_all_null&((data_full['GRE_Q'].isnull())|oor_q),data_full.GRE_Q.mean(),data_full.GRE_Q)
data_full['GRE_V']=np.where(gre_not_all_null&((data_full['GRE_V'].isnull())|oor_v),data_full.GRE_V.mean(),data_full.GRE_V)
```

Graph2.3

With the above being said, the "random missing" assumption does not completely hold true. More details will be offered in the "**Discussion & Future Work**" section.

C. Turning Categorical Variables into Numerical Value/One-hot Encoding

For logistic regression model, all categorical variables must be turned into numbers, or otherwise the model cannot run successfully (see formula above). One-hot encoding is a common way of transformation. In our dataset, however, there is an extra step before the conversion: binding and grouping. After this, we use one-hot encoding to handle the transformed variable bins. It is basically representing categorical variables as binary vectors of values 0 and 1.

For instance, there are 352 unique applying university names and it would be too cumbersome and inefficient to convert it directly, as 352 separate columns will be created.

Moreover, it is possible that data would be too sparse for any pattern (especially true for less opular schools). Hence we grouped these universities into 6 categories as mentioned above (by school ranking), and then applied one-hot encoding to the rankings.

```
# Import external dataset: US News Top 50 schools
school_ranking=pd.read_excel(data_dir+'US-News-Rankings-Universities-Through-2020.xlsx'
                             ,skiprows=1
                             ,parse_cols=[0,7,8,9,10]
                             )
school_ranking.columns=['University Name','rank_2016','rank_2015','rank_2014','rank_2013']

# Get the average rank for the past four years
school_ranking['year_avg']=school_ranking.mean(axis=1)

# Group ranking into different bins
school_ranking.loc[school_ranking['year_avg']<=10,'rank_bin']=['top10']
school_ranking.loc[(school_ranking.year_avg>10)&(school_ranking.year_avg<=20),'rank_bin']=['11-20']
school_ranking.loc[(school_ranking.year_avg>20)&(school_ranking.year_avg<=30),'rank_bin']=['21-30']
school_ranking.loc[(school_ranking.year_avg>30)&(school_ranking.year_avg<=40),'rank_bin']=['31-40']
school_ranking.loc[(school_ranking.year_avg>40)&(school_ranking.year_avg<=50),'rank_bin']=['41-50']
```

Graph 2.4

```
feature_L=pd.get_dummies(data_full[feature_list_L])
```

Graph 2.5

D. Dealing with Specific Issues in the Dataset

For this dataset, we need to deal with some variables in specific ways. Take TOEFL grades as an example. The original data collected was too rich and all information is staying in one column, which made it hard for modeling. Hence, we separated the scores to make them more readable, as shown below.

The original TOEFL grades in one cell:

```
In  [90]: data_full.TOEFL[13]
Out[90]: 'Overall: 116, \n              R: 29 /\n              L: 29 /\n              S: 29 /\n              W: 29'
```

Graph 2.6

After being processed:

```
# #Split TOEFL score into separate entries
TOEFL_score=data_full.TOEFL.str.findall('\d*\.?\d+')
TOEFL_score.dropna(how='all',inplace=True)
```

	TOEFL_all	TOEFL_R	TOEFL_L	TOEFL_S	TOEFL_W
3	105.0	29.0	29.0	20.0	27.0
4	103.0	26.0	29.0	20.0	28.0
5	100.0	29.0	24.0	22.0	25.0
6	NaN	NaN	NaN	NaN	NaN
7	107.0	26.0	27.0	26.0	28.0

Graph 2.7

GRE score is handled in a similar manner, but it had an extra step of mapping old and new scores, which can be found in graph 2.8 below (ETS, 2017). GRE changed its grading scale in 2011 and the data we collected contained both old and new grades (as GRE score is valid for 5 years). We standardized the variable by converting the old grades into new ones using ETS official conversion table.

The original GRE grades (old format):

	GRE_all	GRE_V	GRE_Q	AW
89	1350.0	550.0	800.0	3.0
93	1170.0	380.0	790.0	3.5
105	1350.0	560.0	790.0	3.5
380	1490.0	NaN	NaN	NaN
512	1460.0	660.0	800.0	4.0

Graph 2.8

After being processed:

```
GRE_old_total=pd.merge(GRE_old_total.reset_index(),GRE_cvt[['Old_V','New_V']],left_on='GRE_V',right_on='Old_V',how='left').set_index('index')
GRE_old_total=pd.merge(GRE_old_total.reset_index(),GRE_cvt[['Old_Q','New_Q']],left_on='GRE_Q',right_on='Old_Q',how='left').set_index('index')
GRE_old_total.drop(['GRE_V','GRE_Q','Old_V','Old_Q','GRE_all'],axis=1,inplace=True)
GRE_old_total['GRE_all']=GRE_old_total['New_V']+GRE_old_total['New_Q']
GRE_old_total.columns=['AW','GRE_V','GRE_Q','GRE_all']
GRE_old_total=GRE_old_total[['GRE_all','GRE_V','GRE_Q','AW']]

#replace old GRE score with new score
GRE_list.loc[GRE_old_total.index,['GRE_all','GRE_V','GRE_Q','AW']]=GRE_old_total
```

Graph 2.9

	GRE_all	GRE_V	GRE_Q	AW
21	325.0	160.0	165.0	4.0
39	323.0	163.0	160.0	165.0
40	323.0	163.0	160.0	165.0
41	323.0	163.0	160.0	165.0
42	323.0	163.0	160.0	165.0

Graph 2.10

After all the data processing steps above, now our data is free from outliers, missing values and incompatible formats and it is ready for modeling.

III. Materials and Methodology

For this program, we utilize iPython Jupyter Notebook (python 3.7) to build a predictive model. The model consists of a number of predictors that will be used to predict future admission results.

The following libraries were used：

```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
from copy import deepcopy

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression as LR
from sklearn.ensemble import RandomForestClassifier as RF
import statsmodels.api as sm
from sklearn.metrics import roc_auc_score,accuracy_score,confusion_matrix
from sklearn.metrics import classification_report,recall_score,roc_curve,precision_score,f1_score
```

Graph3.1

Libraries like pandas, numpy and pyplot are quite standard python data processing/plotting libraries that we use for basic data manipulation. Additionally, for this model, we imported the sklearn library, which is a powerful package that contains our main algorithm of logistic regression. Also, packages like preprocessing and metrics in sklearn can help us preprocess data and evaluate the performance of the model. For instance, the confusion matrix and its derived metrics would be used to visualize and evaluate the model performance.

A. Calibration & Training

In this section, we introduced SMOTE to handle imbalance data (to minimize the effect of selection bias), recursive feature elimination (RFE) to select feature and regression summary report interpret impact of each coefficient.

- *SMOTE to Handle Data Imbalance*

Due to the selection bias, our collected data is highly imbalanced. As people can imagine, students who successfully get admitted into universities are more likely to share their experience and application information, whereas it is harder to tell a failure publically. The ratio of 1 to 0 (admitted to denial) is close to 5:1 in our data as a result, which is highly unlikely in reality. Therefore, w**e** utilize SMOTE to generate synthetic data of minor class (rejected, or "0") in order to mitigate problems and train for a better model. The code can be seen below. After processing, the ratio of admitted and declined application for training

```
# this is an extra step to handle imbalanced dataset.
oversample=SMOTE(random_state=531)
os_X,os_y=oversample.fit_sample(train_x,train_y['result'])
os_X=pd.DataFrame(data=os_X,columns=train_x.columns)
os_y=pd.DataFrame(data=os_y,columns=['result'])
```

dataset is close to a well-balanced 1:1.

Graph3.2

- *Recursive Feature Elimination (RFE)*

The overall purpose of recursive feature elimination is to get rid of weak features and only keep strong performers—feature selection. Redundant variables can cause many issues. The unnecessary data collection, processing and storage are definitely one. Besides, if there are too many independent variables with too few data points, the overfitting issue can arise. Over-fitting is caused by independent variables attempting to capture patterns that are actually noises. Before using RFE, we need to standardize our variables to ensure that they range in the same scale, as seen in graph 3.3.

After running the code, 45 variables are reduced to 25 variables: year variables, GPA, some school rank variables, some major variables and some standardized test result variables (graph 3.4).

Standardize variables:

```
# Feature Standardization, scaling
from sklearn import preprocessing as pp
scale_var=['GRE_all_filled',
           'GRE_Q_filled',
           'TOEFL_all_filled',
           'TOEFL_L_filled',
           'TOEFL_S_filled',
           'TOEFL_W_filled']

mm_scaler=pp.MinMaxScaler()
X_mm_scale=mm_scaler.fit_transform(feature[scale_var])
# make it a dataframe and rename the columns

scale_feature=pd.DataFrame(X_mm_scale)
scale_feature.columns=['GRE_all_filled_s',
                       'GRE_Q_filled_s',
                       'TOEFL_all_filled_s',
                       'TOEFL_L_filled_s',
                       'TOEFL_S_filled_s',
                       'TOEFL_W_filled_s' ]
#reset index for feature, so that we can merge two dataset easily
feature.reset_index(inplace=True)
feature.drop('index',axis=1,inplace=True)
```

Graph 3.3

Five variables being selected after feature sections:

```
# feature ranking with recursive feature elimination
#(by recursively considering smaller and smaller sets of features)
# please read more here:# https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html
from sklearn.feature_selection import RFE
logreg=LR(solver='liblinear')
#choose 25 variables out of 45
rfe=RFE(logreg,25)
# rfe=rfe.fit(train_x,train_y)
rfe=rfe.fit(os_X,os_y.values.ravel())
```

```
['year_2013',
 'year_2014',
 'year_2015',
 'ug_gpa_mapping',
 'rank_bin_top10',
 'degree_bin_JD',
 'degree_bin_MA',
 'degree_bin_MEd',
 'degree_bin_MLA',
 'degree_bin_MS',
 'degree_bin_PhD',
 'major_mapping_Art',
 'major_mapping_Business',
 'major_mapping_Communication',
 'major_mapping_Economics',
 'major_mapping_Education',
 'major_mapping_Engineering',
 'major_mapping_Environment',
 'major_mapping_Mathematics',
 'major_mapping_Other',
 'major_mapping_Stats',
 'ug_level_mapping_other',
 'ug_level_mapping_oversea',
 'GRE_Q_filled_s',
 'TOEFL_all_filled_s']
```

Graph 3.4

- *Model Summary Table & Explanation*

Since we carefully choose our variables and implement RFE ahead, all variables are statistically significant (p-value <0.025). Looking at the coefficients, here are some takeaways:

1. The continuous variable coefficients interpretation is straightforward. Larger the coefficient, the higher the contribution the variable is to the success admission result (absolute value, if negative, then higher negative contribution). For example, looking at the standardized test scores like TOEFL and GRE, we find that both have positive impact to the admission result (no wonder). Also, GRE has a bigger coefficient than TOEFL, meaning that GRE is a more significant factor than TOEFL in regarding to the graduate school application.

2. For the dummy (categorical) variables, the sign (+ or -) of each independent variable depends on the removed variable(s) in the same category. The removed variable(s) serve as the benchmark and the positive and negative numbers are the estimation in

regarding to the benchmark. For example, the year 2015 is removed and it is the hardest year of getting admitted. Hence, comparing to year 2015, the coefficients of 2013 and 2014 are both positive, which are 4.91 and 5.06 respectively. These positive coefficients are in regarding to year 2015, indicating that students are more likely to get admitted in those years than 2015, keeping everything else constant. Another example is related to variables of university ranking. The coefficients increase as the universities ranking decreases (lower ranking universities). As it can be seen in the table, the coefficient for top10, rank_bin 31-40 and 41-50 is -0.5456, 0.6479 and 0.9051, respectively. For the rank_bin "others", the coefficient is even larger, which reaches 1.1856. This monotonic increasing coefficients confirm with the intuition: it is easier for students to get admitted into lower ranking universities than higher ranking universities as those schools are usually more selective and competitive.

In [302]:
```
# fit a model and see how it runs
# drop the insignificant variables
#here we use statsmodel.api instead of sklearn as it can generate a nice report,  LogisticReg
logit_model=sm.Logit(os_y,sm.add_constant(train_os_x_rfe))
result_logit1=logit_model.fit()
result_logit1.summary2(title='Logistic Regression Result for Grad School Application Result Fo
```

```
Optimization terminated successfully.
         Current function value: 0.575945
         Iterations 9
```

Out[302]:

Model:	Logit	Pseudo R-squared:	0.169
Dependent Variable:	result	AIC:	12937.0370
Date:	2019-11-22 01:20	BIC:	13127.4199
No. Observations:	11186	Log-Likelihood:	-6442.5
Df Model:	25	LL-Null:	-7753.5
Df Residuals:	11160	LLR p-value:	0.0000
Converged:	1.0000	Scale:	1.0000
No. Iterations:	9.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-2.1707	0.5794	-3.7467	0.0002	-3.3063	-1.0352
year_2013	5.1520	0.5828	8.8398	0.0000	4.0097	6.2943
year_2014	5.6013	0.5746	9.7475	0.0000	4.4751	6.7276
ug_gpa_mapping	1.4878	0.4415	3.3699	0.0008	0.6225	2.3531
semester_Fall	-2.7153	0.3860	-7.0335	0.0000	-3.4719	-1.9586
semester_Spring	-1.7525	0.4827	-3.6309	0.0003	-2.6985	-0.8065
rank_bin_11-20	0.6479	0.0901	7.1925	0.0000	0.4713	0.8245
rank_bin_31-40	0.8895	0.1002	8.8748	0.0000	0.6931	1.0859
rank_bin_41-50	0.9051	0.0864	10.4699	0.0000	0.7357	1.0746
rank_bin_other	1.1856	0.0708	16.7359	0.0000	1.0468	1.3245
rank_bin_top10	-0.5465	0.0849	-6.4365	0.0000	-0.7130	-0.3801
degree_bin_MA	-0.8704	0.0788	-11.0419	0.0000	-1.0249	-0.7159
degree_bin_MEd	-0.9176	0.1434	-6.3987	0.0000	-1.1986	-0.6365
degree_bin_MLA	-0.8608	0.1203	-7.1532	0.0000	-1.0967	-0.6249
degree_bin_MS	-0.9856	0.0792	-12.4437	0.0000	-1.1408	-0.8303
degree_bin_PhD	-1.8170	0.0802	-22.6594	0.0000	-1.9742	-1.6599
degree_bin_others	-0.6060	0.1132	-5.3553	0.0000	-0.8278	-0.3842
major_mapping_Art	1.3043	0.3140	4.1543	0.0000	0.6890	1.9197
major_mapping_Biology	-0.4772	0.1013	-4.7116	0.0000	-0.6757	-0.2787
major_mapping_Design	-1.0753	0.2613	-4.1145	0.0000	-1.5875	-0.5631
major_mapping_Medical	1.3134	0.2855	4.5996	0.0000	0.7537	1.8730
major_mapping_Stats	-1.0809	0.1569	-6.8893	0.0000	-1.3884	-0.7734
GRE_all_filled_s	3.0683	0.4069	7.5404	0.0000	2.2707	3.8658
TOEFL_L_filled_s	0.8040	0.1673	4.8066	0.0000	0.4762	1.1318
TOEFL_S_filled_s	0.8441	0.2036	4.1464	0.0000	0.4451	1.2431
TOEFL_W_filled_s	0.5248	0.1604	3.2728	0.0011	0.2105	0.8391

Graph 3.5

B. Basic Application

After the modeling being trained, we can use it to predict new candidate application result. Given candidates profile including their applying school, applying major, degree, undergraduate school level, their TOEFL score (listening, speaking, reading, writing), GRE grades and GPA, our model can predict whether they will be admitted or not. The result can directly show all aspects' contributions to the admission decision and whether an improvement can impact the result significantly. We extracted one sample from our testing result to demonstrate the prediction here.

For example, for the below candidate who applies for a master of science degree in NYU Polytechnic School of Engineering in Fall 2015 with a GPA of 75%, GRE of 308 and TOEFL listening, speaking and writing of 17, 22, 22, respectively, his/her application will likely be rejected. Moreover, based on our model, if the candidate can improve his/her GPA to 85%, TOEFL listening score to 27 and GRE score to 325 (I scaled them back to actual scores rather than the post-standardization result), he will be admitted (according to our model, not actual result).

```
In [144]: #Look at 1 individual applicant in detail in testing pop
          test_x[selected_col].iloc[18]

Out[144]: year_2013              0.000000
          year_2014              0.000000
          ug_gpa_mapping         0.750000
          semester_Fall          1.000000
          semester_Spring        0.000000
          rank_bin_11-20         0.000000
          rank_bin_31-40         0.000000
          rank_bin_41-50         0.000000
          rank_bin_other         1.000000
          rank_bin_top10         0.000000
          degree_bin_MA          0.000000
          degree_bin_MEd         0.000000
          degree_bin_MLA         0.000000
          degree_bin_MS          1.000000
          degree_bin_PhD         0.000000
          degree_bin_others      0.000000
          major_mapping_Art      0.000000
          major_mapping_Biology  0.000000
          major_mapping_Design   0.000000
          major_mapping_Medical  0.000000
          major_mapping_Stats    0.000000
          GRE_all_filled_s       0.655172
          TOEFL_L_filled_s       0.235294
          TOEFL_S_filled_s       0.500000
          TOEFL_W_filled_s       0.384615
          Name: 3176, dtype: float64
```

```
In [166]: print('The candidate is predicted to be {} and the actual result is {}'.format(y_pred_int[18],test_y.iloc[18][0]))
          The candidate is predicted to be 0 and the actual result is 0
```

```
wi_change=sm.add_constant(test_x[selected_col]).iloc[18]
wi_change['ug_gpa_mapping']=0.85
wi_change['TOEFL_L_filled_s']=0.8
wi_change['GRE_all_filled_s']=0.75
```

```
1 if result_logit1.predict([wi_change])>cutoff else 0
```

```
1
```

Graph 3.6

III. Results and Performance

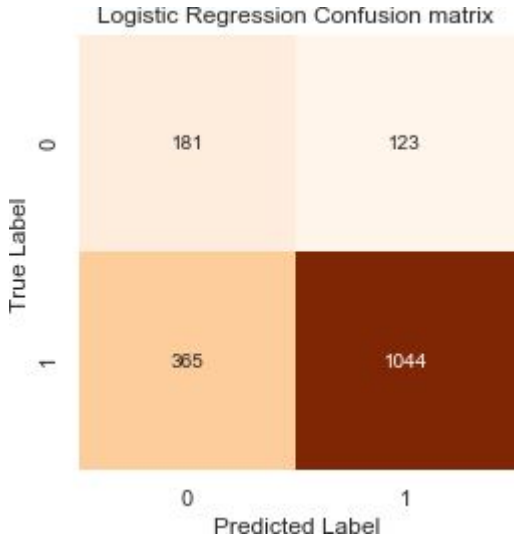
A. Confusion Matrix And Baseline Model

Confusion matrix is a table that visualizes the relationship between the prediction and actual result on a 2 x 2 matrix.

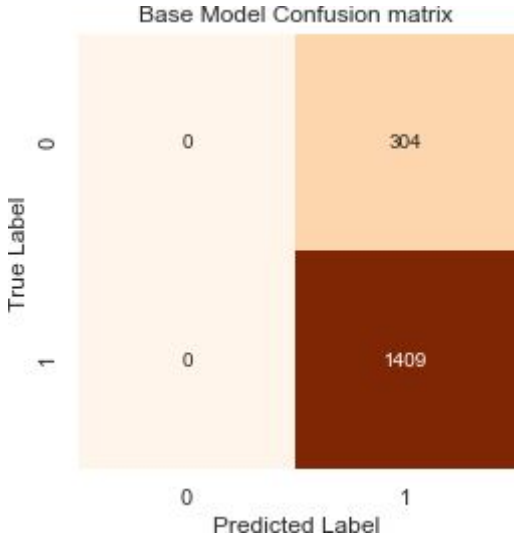
Baseline model is the model we use to compare performance with the logistic regression model we trained. We use "all ones" for the baseline model, i.e. everyone will be admitted (common, 2015)). We choose it instead of 50-50 random guess because of "zero-rule". For the situation of such imbalanced dataset with almost 80% admission rate, setting all as admitted has a better performance than random guess (i.e. picking a better benchmark to compare). The performance of the regression model and baseline model can be seen below.

TP(true positive)	Predicting as admit and actual admit
TN(true negative)	Predicting as reject and getting reject
FP(false positive)	Predicting as admit and getting reject
FN(false negative)	Predicting as reject and getting admit

Logistic Model confusion matrix:



Base Model confusion matrix:



Graph 3.7 Graph 3.8

B. Performance Metrics

For this model, we use precision instead of accuracy to measure the performance due to the imbalanced data issue we mentioned earlier. Taking cancer detection as an example: 99% of people do not have cancer and only 1% do. If we use accuracy as a measure, a simple model claiming everyone healthy will achieve 99% accuracy score, though in reality it provides no insight. Instead, another model that identifies the cancer patient along with several other false positives (healthy people identified as having cancer) may seem perform worse in terms of accuracy, but it does provide value as it correctly identifies the true positive (cancer patient). Similarly, given our data, if we use accuracy, we could get an 82% accuracy score model simply by setting all students as admitted. Since it is more important for us to detect true rejections, we use precision to measure the performance of the model. It considers false positive (FP) as more costly and aims to minimize it. As people can see below, our model achieves a precision score of 0.89 whereas the baseline model achieves 0.82. We do get a better model than baseline model.

```
# model classification report
print(classification_report(test_y,y_pred_int))
```

	precision	recall	f1-score	support
0	0.33	0.60	0.43	304
1	0.89	0.74	0.81	1409
accuracy			0.72	1713
macro avg	0.61	0.67	0.62	1713
weighted avg	0.79	0.72	0.74	1713

```
# base model classification report
print(classification_report(test_y,y_base))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	304
1	0.82	1.00	0.90	1409
accuracy			0.82	1713
macro avg	0.41	0.50	0.45	1713
weighted avg	0.68	0.82	0.74	1713

Graph 3.9

C. AUC-ROC curve

AUC ROC curve is a visual way to measure the performance of different classification

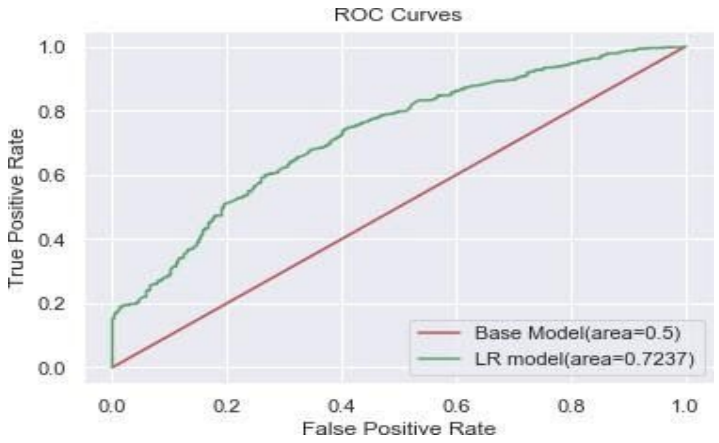
```
rg_fpr,rg_tpr,_ = roc_curve(test_y, y_base)
LR_fpr, LR_tpr, _ = roc_curve(test_y,y_pred)

# Plot both curves
roc_auc_LR=round(roc_auc_score(y_true=test_y,y_score=y_pred),4)
plt.plot(rg_fpr, rg_tpr, 'r', label = 'Base Model(area=0.5)')
plt.plot(LR_fpr, LR_tpr, 'g', label = 'LR model(area={})'.format(roc_auc_LR))

plt.legend();
plt.legend(loc="lower right")
plt.xlabel('False Positive Rate');
plt.ylabel('True Positive Rate');plt.title('ROC Curves');
plt.show();
```

models. It is a trade-off between the FP and TP with different cutting points. Without further detail into the technical explanations, the bottom-line for the curve is: the larger the area under the curve, the better the model is. Based on the curve below, our logistic regression model shows a better performance than the baseline model.

Graph 3.10

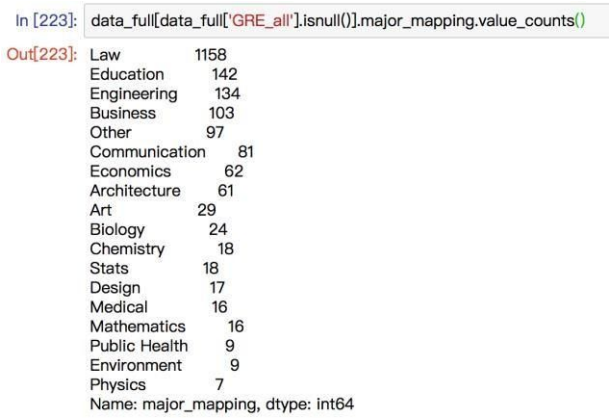


Graph 3.11

IV. Discussion & Future Work

There are improvements that could be made to this program.

1. *Null value imputing*: When processing the null data, we found that there are a lot of missing GRE and TOEFL grades in students' profiles. For simplicity, we just filled them with average grades. However, as seen below, there is actually a pattern behind these missing data points, as majors like Law do not require students to submit their GRE grades. Instead, they ask for LSAT scores (which is not recorded here). To better handle this problem, we can collect additional score information such as the LSAT scores. Also, we can apply the data with tree model and make the GRE null values as a separate new category along with applying major.



Graph 4.1

2. Model Selection: Logistic regression model may not be the best choice when dealing with too many categorical variables and missing values. In our program, there are lots of missing values. Random forest model is able to handle mixed types of missing data better by nature. Also, we assumed linearity when applying the logistic regression, which may not be true. Tree models can handle the non-linear relationship better (Breiman & Cutler, p. 1). We can do more exploratory work on that.

3. Data Mapping: When processing the rank of the universities, we only used the U.S. News Top 50 U.S. universities, which may not be a comprehensive representation of school rankings. There are plenty other renowned universities out of the state, such as the University of Oxford, University of Toronto, and Cambridge University, just to name a few. In our model, these schools are listed in the "other" category, which does not accurately reflect their rankings. In the future, we can introduce more comprehensive ranking systems that includes world ranking to assess the difficulty of entrance represented by the school ranking.

V. Acknowledgement

I would like to thank Stella Zhang who gave corrections and suggestions to this paper.

VI. Reference

December 10, 2019, from http://www.moe.gov.cn/srcsite/A22/s7065/200512/t20051223_82762.html.
Reiter, A. G. (n.d.). U.S. News & World Report Historical Liberal Arts College and University

Rankings.

Retrieved December 10, 2019, from http://andyreiter.com/datasets/.

Ahmed, L. A. (2017, August 16). Using Logistic Regression in Determining the Effective

Variables in Traffic Accidents. Retrieved December 10, 2019, from

https://pdfs.semanticscholar.org/e629/580aa89bf9d5f8f5e548413de0a0d9e9f628.pdf.

Breiman, L., & Cutler, A. (n.d.). Random Forests. Retrieved December 10, 2019, from

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.

common 5311 silver badge33 bronze badges, & Claesen Marc Claesen 15.9k11 gold badge3636

silver badges6262 bronze badges, M. C. (2015, April 24). What is the chance level accuracy in

unbalanced classification problems? Retrieved December 10, 2019, from

https://stats.stackexchange.com/questions/148149/what-is-the-chance-level-accuracy-in-u

imbalanced-classification-problems.

ETS. (2017). GRE(R) Verbal and Quantitative Reasoning Concordance Tables. Retrieved

December 10, 2019, from https://www.ets.org/s/gre/pdf/concordance_information.pdf.

NSFSA. (n.d.). Trends in U.S. Study Abroad. Retrieved December 10, 2019, from

https://www.nafsa.org/policy-and-advocacy/policy-resources/trends-us-study-abroad.

K assambara. (2018, March 11). Logistic Regression Assumptions and Diagnostics in R.

Retrieved December 10, 2019, from

http://www.sthda.com/english/articles/36-classification-methods-essentials/148-logistic-re

regression-assumptions-and-diagnostics-in-r/.