Analysis on graduate school admission

Ryeongkyung Yoon | u1136736

1 REPOSITORY

https://github.com/rkyoon12/CS6350-spring-2019

2 INTRODUCTION

Annually over thousands of students in the world have applied to the graduate program of university in US. Since these processes are highly competitive and the university needs to evaluate lots of applicant based on the application, the university requests students many things to prove themselves. So student applies to the program by not only competing application form, but also preparing many things what the university request them such as Transcript, Statement of Purpose, Letter of Reference, GRE score and TOEFL score (only for international students). Furthermore, students are willing to appeal their talent by providing their research description or highlighting the ranking of their undergraduate. After applying to several school, the only thing they can do is waiting. Almost students were forced to rely on the standards set given by the school. But those criteria are also not easily accessible. They usually have to ask through individual e-mail, but even that is more likely to be unanswered. Sometimes schools present statistical data on the successful candidates in their website, but it is quite abstract . Some of students started creating a site that shared their specifications with results. Based on the information there, students used to predict their passing or failure. However the site was not only a small number of information-roll operations, but it also added to the confusion among students due to the wide variation in acceptance.

There is a statistical approach to dealing with a lot of data. However, statistical methods, i.e., means, deviations, etc., can analyze and predict the trend of data for each features. But in our case which is affected by multiple feature, statistical methods can't analyze the tendency of entire dataset, and they can't return the reliable estimation on it.

Machine learning, on the other hand, can answer many questions. What is the most important factor among attributes in determining their admission and so In what areas should the student focus more on? Can we expect the result of admission based on the given dataset? Furthermore, what is the reference score for each item? and so on. Actually, we can do supervised learning with data that has labels. Through this, we can firstly find the attribute in the upper level. Therefore, it is possible to analyze the importance of each item, such as which items students should focus more on. Again, the machine learning allows us to use large amount of data and so the results obtained can be highly reliable because a large amount of data has been reflected. In addition, it is possible to predict acceptance by using a trained

model using methods such as decision tree and preceptron. At this point, theories such as PAC learning theory suggest how much data I need to use for the much reliability and accuracy I want, so these machine learning methods can predict fairly accurate results.

3 Machine Learning Model.

1. Collect and Clean the dataset

The dataset has 7 number of attributes (GRE Score, TOEFL Score, University Ranking, Statement of Purpose, Letter of Reference, GPA, Number of Research Experience) and the last column is the chance of admission. I split 400 numbers of dataset into 300 number of training data and 100 number of test data. Here, every features are the real valued and these are normalized by

$$\tilde{x} = \frac{x - x_{min}}{x_{max} - x_{min}},$$

which makes values in between 0 and 1. Lastly, our labels are the probabilities to be accepted. To transform it into the binary variable, I setup the threshold as 0.8 and return label 1 if probability is greater than 0.8 and return label -1 otherwise.

2. Decision Tree with ID3 Algorithm

Decision Tree is the nonlinear classification method. From the decision tree we learned from the training dataset, we can expect the which attribution is most important in admission. To check the most effective attribute, I compared the Information Gain based on the cross entropy at each depth. Since we have 7 numbers of the attributes, I allow the the decision tree to stretch upto depth 7. Here are the train and test accuracy at each depths. As we

Depth	train accuracy	test accuracy
1	84.0	85.0
2	89.66	87.0
3	89.66	87.0
4	91.66	88.0
5	93.33	88.0
6	94.0	87.0
7	94.0	87.0

Table 3.1: Decision Tree by varying the depth.

can see the result, the train accuracy increase as the decision tree is deeper. But the test accuracy increases upto depth =4, but after that it decreases again and this is caused by the overfitting. From the learned decision tree, I find out the first attribute is the CGPA. In fact, this is coincide with the real admission grading system because there's screening system which sort all applicants based on the GPA score at the beginning. That means if the applicants has the GPA score below some level, their applications are never examined by the committee members.

As shown in the Table 3.1, I got the best test accuracy 88.0 with the train accuracy 93.33. This results is good, but I will try to apply the other machine learning method to get higher accuracy on the test data.

```
{'CGPA': {0: {'TOEFL_Score': {0: 0,
                              1: {'Research': {0.0: 0,
                                               1.0: {'GRE_Score': {0: 0,
                                                                   1: {'University_Rating': {0: {'LOR': {0: 1,
                                                                                                         1: 0}},
                                                                                             1: 0}}}}}}}},
          1: {'Research': {0.0: {'GRE_Score': {0: {'University_Rating': {0: 0,
                                                                         1: 1}},
                                               1: {'LOR': {0: {'SOP': {0: {'TOEFL_Score': {0: 1,
                                                                                           1: {'University_Rating': {0: 0,
                                                                                                                     1: 0}}}},
                                                                       1: 0}},
                                                           1: {'TOEFL_Score': {0: 0,
                                                                               1: {'University_Rating': {0: 1,
                                                                                                         1: {'GRE_Score': {1: 1}}}}}}}}}},
                           1.0: {'SOP': {0: {'University_Rating': {0: {'LOR': {0: {'GRE_Score': {0: {'GRE_Score': {0: 0}},
                                                                                                 1: {'GRE_Score': {1: 0}}}},
                                                                               1: {'GRE_Score': {1: {'GRE_Score': {1: 0}}}}}},
                                                                   1: {'TOEFL_Score': {0: 0,
                                                                                       1: 1}}}},
                                         1: {'TOEFL_Score': {0: {'LOR': {0: 1,
                                                                         1: {'GRE_Score': {0: {'GRE_Score': {0: 0}},
                                                                                           1: 0}}}},
                                                             1: {'GRE_Score': {0: {'University_Rating': {0: {'GRE_Score': {0: 0}},
                                                                                                         1: 0}},
                                                                               1: {'LOR': {0: 1,
                                                                                           1: {'University_Rating': {0: 1,
                                                                                                                     1: 1}}}}}}}}}}}}}}
```

Figure 3.1: Learned Decision Tree with threshold $th = 0.8$

3. Linear Classifier : Perceptron and SVM

For the linear classifier, I used the perceptron method and the SVM so that learned weight vectors for each. About the perceptron method, it go through every training dataset and update the learned weight vector when the corresponding dataset is misclassified. In the lecture, we learned that the averaging perceptron works better than standard perceptron and voting perceptron. Here, the average error of testdata can be different depending on the learning rate. If the learning rate is large, then the learned weight vector will changes dramatically based on the dataset. In my code, I used learning rate r= 0.01. On the other hand, the SVM algorithm penalizes not only on misclassifed data but also on data inside the margin. Here, I used the soft SVM and stochastic gradient descent method with learning rate = 0.0001. To check the convergence of the algorithm, I used the curve of loss function which is shown in the code.

Method	train accuracy	test accuracy
Averaging Perceptron	90.0	94.0
Soft SVM	90.33	94.0

Table 3.2: Linear Classifier : Averaging Perceptron with learning rate 0.01 and Soft SVM with learning rate 0.0001

4. Nonlinear Classifier : : Nonlinear SVM and Neural Network For the nonlinear classifiers, I used the nonlinear SVM using the kernel trick and Neural Network. About the nonlinear SVM, the idea is very close with the SVM above. Using the Gaussian kernel, it cast the

nonlinear mapping substituting the inner product in the linear SVM. In the code, I used the gamma = 0.5. On the other hand, the neural network is very powerful method. It construct the network so that I can vary not only the number of nodes at each layer but also the number of hidden layers. At each layers, we compute the linear combination of the weight with the output from the previous layers and take the activation function which becomes the output for the hidden layer. In my code, I fixed it as three layer network having 25 nodes and each nodes in the layers are fully connected. I also used the sigmoid function as an activation function.

Method	train accuracy	test accuracy
Nonlinear SVM	92.0	94.0
Neural Network	91.00	94.0

Table 3.3: Noninear Classifier : Nonlinear SVM and Neural Network.

Let's consider the one specific cases. The student is not good at English and his GRE and TOEFL score are below the median. But his university is top school(rank =5) and his SOP, LOR scores are really high (score =5). Also he was good as undergraduate and has research experience. Then $x = [0,0,5,5,5,1,1,1]$. Then the prediction from every methods is 1. Hence we can predict that it is highly probable that he will be accepted by more than 80%.

5. Other Results with Different Thresholds.

As shown in the tables, I applied the each machine learning methods and get the better performance than the decision tree. With the linear classifiers, I got around 90% accuracy on traindata and got around 94% accuracy on the testdata. With the nonlinear classifiers, I got around 92% accuracy on the traindata and got 94% accuracy on the testdata. These accuracy is higher than decision tree but still is not close to 100% because I modified the labels by changing the real valued labels into the binary variables. So the accuracy can be different depending on how I set up the threshold. Here is the results of each methods with dataset modified with threshold 0.7 and 0.9. I slightly tune the hyperparameters providing the best accuracy. As shown tables below, with the threshold 0.9, I got higher accuracy which means that my algorithm is reliable.

Method	train accuracy	test accuracy
Decision Tree	82.33	85.0
Averaging Perceptron	84.0	89.0
Soft SVM	80.33	90.0
Nonlinear SVM	87.33	88.0
Neural Network	85.0	90.0

Table 3.4: Accuracy of methods with threshold = 0.7

Method	train accuracy	test accuracy
Decision Tree	90.0	93.0
Averaging Perceptron	97.0	99.0
Soft SVM	95.66	100
Nonlinear SVM	95.66	99.0
Neural Network	97.0	100.0

Table 3.5: Accuracy of methods with threshold = 0.9

4 FEATURE PLAN

Until now, I only applied the Non-probabilistic models. But there are probabilistic learning models like Logistic regression or Naive Bayes. I will apply them and compared with my results and check which method is better for my dataset. About the Neural Network, we didn't consider the regularization so it can be overfitting. As I researched, there are several ways to regularize algorithms like dropout or including L2 regularization terms. I want to apply both ways and compare with the my result. I also can consider more wider or deeper networks too.

5 REFERENCE

1) `https://www.kaggle.com/mohansacharya/graduate-admissions`