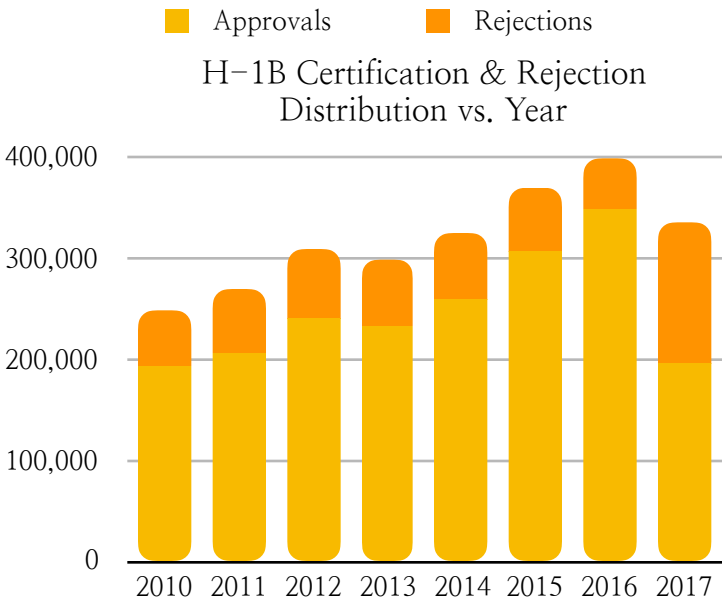
Reject or Certified: Predicting H-1B VISA Result

Extreme Value Theorem (Citina Liang, Yan Kang, Zheng Wang)

Introduction



H-1B VISA, being described by Dr. Michio Kaku as the "secret weapon that suck up all the brains from the world", is certainly one of the most crucial thing for every international student who decide to work or even stay in the United States. However, while most of the applicant of H-1B VISA gets Approved, a lot of applicants still get rejected. So what could potentially be some factors that undermine the probability of approval? What factors could boost the chances of being certified? Bearing these problems in mind, our group decided to investigated the trend of H-1B approval rate and develop a model that predicts the outcome of H-1B application. Our investigation of the data can be summarized in the following 4 steps:

1. Determining the characteristic of the data
2. Cleaning the data to simplify the model building process
3. Choose a model[1] that is most compatible with the data
4. Adjust the model to optimize it's performance

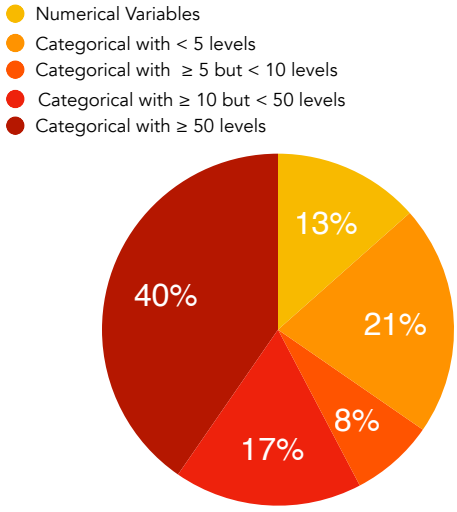
At the end of our journey, we were able to obtain a model that predicts the H-1B outcome with more than 91.5% of accuracy.

---

[1] This can include:
(a) Logistic Regression (b) K-Nearest Neighbor (c) Generalized Boosting Model
(d) Random Forest (e) Linear Discriminant Analysis
and other models with capacity of making classification

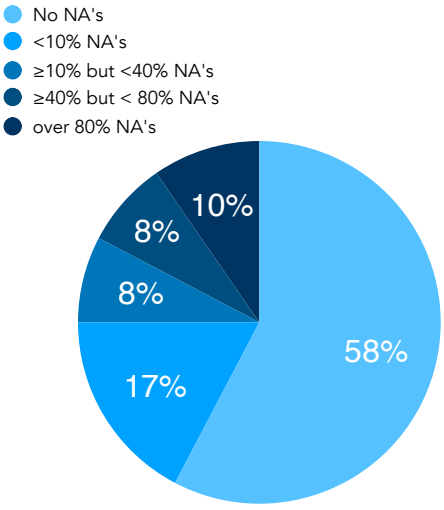
Character of the Data

Our construction of the model begin with looking at our dataset and determine its characteristic. We first noticed that most of the variables provided to us are categorical variables (only 7 out of 52 are numerical). Moreover, a lot of the categorical variables given to us has more than one thousand levels. (*Refer to the Red figure*) This observation force us to take actions that deal with the original data as the original dataset would be too complicated to make deep analysis.



* variable categories summary

Secondly, almost all of the variables have some missing values, and the proportion of the missing values can range up to over 90% of the 4918 observations. (*Refer to the blue figure*) This fact about our dataset makes the traditional approach of simply omitting roles with "NA" impossible since doing so would cause a severe loss of observations available.



* NA rate of all variables

Finally, we fix some of the incorrect information filling[2] of in some of columns such as *SOC_CODE*. Since this minor issues only affect very few rows, we simply omit them as this will make little change to the dataset.

Cleaning the Data

Our next step is clean the data so that we fix the issues mentioned above. For convenience, we start from dealing with the NA's. For this issue, our solution is simple:

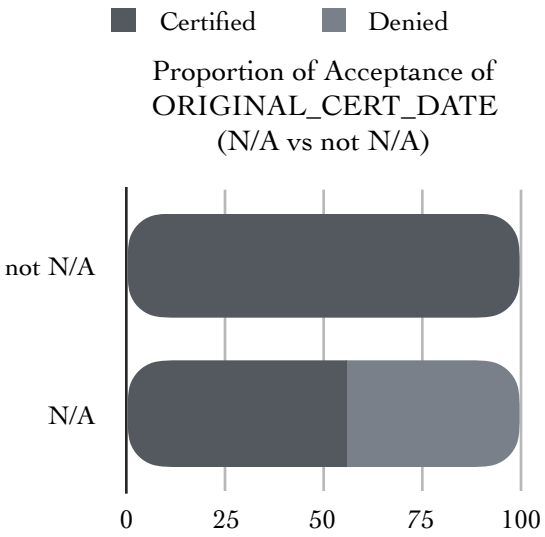
[2] This includes filling information in the wrong column and filling information that make no sense

substitute NA's for the string "N/A" and make it as an extra level. This solution have two main advantages. On the one hand, it will not harm the original structure of the categorical variables since we have kept all of the original data; on the other hand, this extra level provides additional information that the information is missing. We envision that this knowledge might help making predictions. For numerical variables, we decided to make the NA's to be 0. This is because for all numerical variables, there are only few rows (in fact less than 5) that have NA's. Thus changing them to zero will not have strong effect to the overall training dataset.

After dealing with the missing data, we then try to reduce the complexity of the categorical variables, the methods are mainly:

1. Combine the levels[3] by common character
2. "Numericalize[4]" the categorical variable with external data

After that, we also make some transformation of the numerical variables so that they are more mutually comparable. For the three WAGE-related variables, we unify their unit into years by making the assumption that 40 hours of work is done each week and there are 52 weeks in a year. For the other four date-related variables, we take the differences between the start and end dates to create a derived variable. Also, we transform each of the date variables to weekdays, which we consider



[3] This method is mainly used for factors with more than a thousand levels. Most of the times, we simply combine all of the "N/A" as "Information not Provided" and all the rest as "Information Provided". But for variables like *SOC_CODE*, which are naturally subcategories of several major classes, we merge all the subclasses into their major ones.

[4] This is use some of the numerical character to explain the categorical variables. For example, for the variable *WORKSIT_STATE*, which is qualitative, we can use population, GDP, and unemployment rate (which are numerical values) to describe a particular state.

could potentially affect the acceptance probability of the H-1B VISA application.

Finally, we plot all of the variables against the approval rate of the H-1B VISA and pick the variables that have significant influence (an example is the Figure about ORIGINAL_CERT_DATE on the page above) on the acceptance rate. The result this filtration is 29 potentially useful predictors.

Model Selection

So far, we have made the dataset friendly enough that we can structure a prediction model based on it. Based on a pool of candidacy models[5], we first rule out the Logistic Regression and Generalized Boosting Model. This is because the complicated categorical variables forces the regression models to estimate a lot of dummy variables and make them easily over-predicting the outcome. Then we can also rule out the KNN model since it does not work really well under high dimensions, given the fact that we are trying to make a model with approximately 29 variables, KNN should not be a good option. Finally, due to the difficult of checking the normality assumption for categorical variables, we also choose not to use LDA for our model.

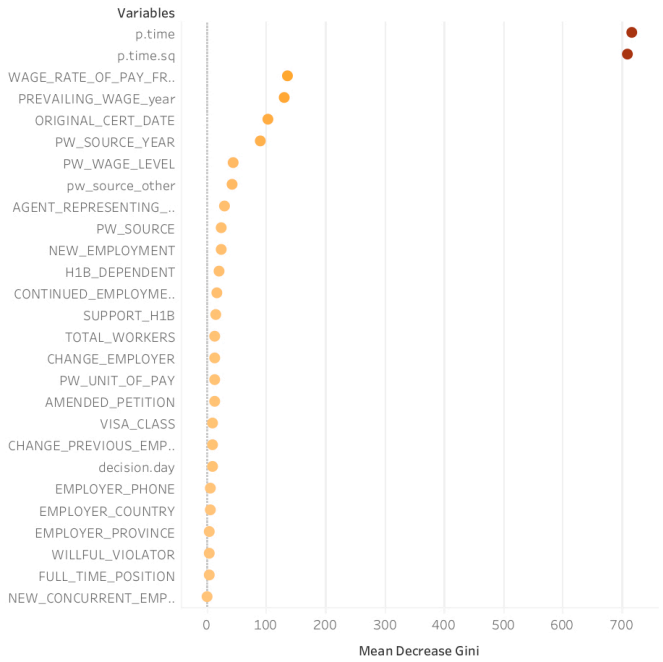
Hence based on the characteristic of the data, we decided to construct our model based on RandomForest. At this stage, the accuracy rate is 90.5%.

Model Optimization

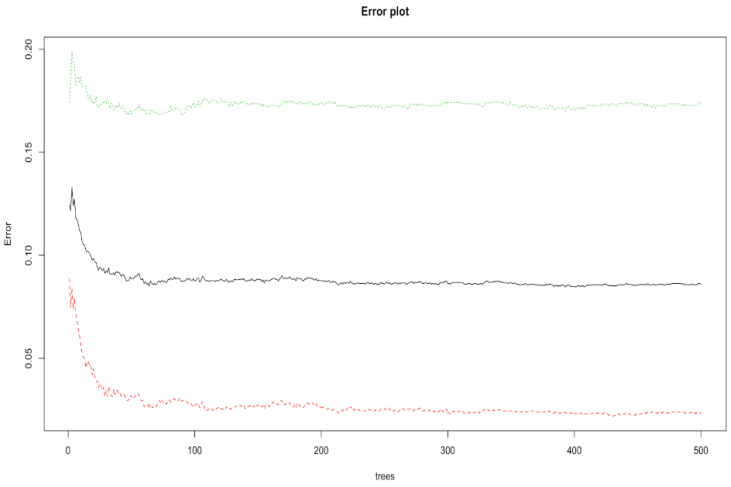
Reaching this stage, we have already had a model that is working reasonably well. Yet, there could be some potential to the model we created; therefore, we then try to optimize the performance by doing intensive cross validations. We begin by choosing the subset of the full model and do 10-fold cross validations for it. In this step, we conclude that a model with 28 variable could actually predict with ~1% more

[5] Refer to footnote 1 on Page 1

accuracy rate than the full model with 29 predictors. The significance of each of the 28 variables is concluded in the Variable Importance Plot on the right. Finally, we run a cross validation to find the best number of trees and mtry for the model. The result of the cross of our cross validation is summarized in the Error Plot below.

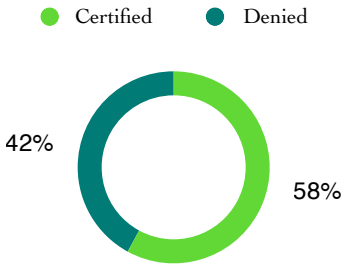


Variable Importance Plot



Based on this result, we see no significant changes in the accuracy rate after we reach over 200 tree (the general pattern is similar for mtry). Thus for both of the parameter, we decried to use the default values.

Assumption Checking & Discussion



Finally, to conclude our project, we check the assumption: the relatively even split between the outcome. Based on the pie chart showing the split, we conclude that the assumption for Random Forest model is valid.

In this project, we start from Random Forest right away. However, it is not always the best way to make models. Based on result of other classmates' model that works better. It is likely a good idea to begin with a simple model, and then see if a more sophisticated model can make improvements.