Predicting H-1B Visa status

CAPSTONE PROJECT
Machine Learning Engineering
Numan Yilmaz
May 10th, 2018

0- Overview

H-1B is a type of visa that allows US employers to employ foreign workers to work in the USA for a certain time. This type of visas are taking long time to process which could be 1-2 years. Does it really make sense for big tech companies to go through visa process for an employee that may or may not be a fit for a position? To solve this problem, given the occupation name, job title, position type and salary, a model could give us the chance of acceptance before even file a petition. That would save so much time and money for big companies.

1- Data Collection

The dataset for this project is coming from Kaggle.[1] It contains H-1B visa Petitions for the period 2011 to 2016. There are roughly 3 million samples. The dataset can be downloaded as csv file which is about 500MB.

The features in the dataset are case status, employer name, occupation name, job title, position type, prevailing wage, year, work site, latitude and longitude.

CASE_STATUS: This column indicates that if the petition was approved or not. This is our target feature. There were 7 possible values in the dataset.

EMPLOYER_NAME: Name of the employer that is submitting the application.

SOC_NAME: Name of the occupation defined by Standard Occupational Classification (SOC). System. There are 1584 unique jobs in the dataset.

JOB_TITLE: Title of the job.

FULL_TIME_POSITION: For a full time job this column value is “Y” and for part time positions, it is “N“.

PREVAILING_WAGE: Salary of the position.

YEAR: Petition application year.

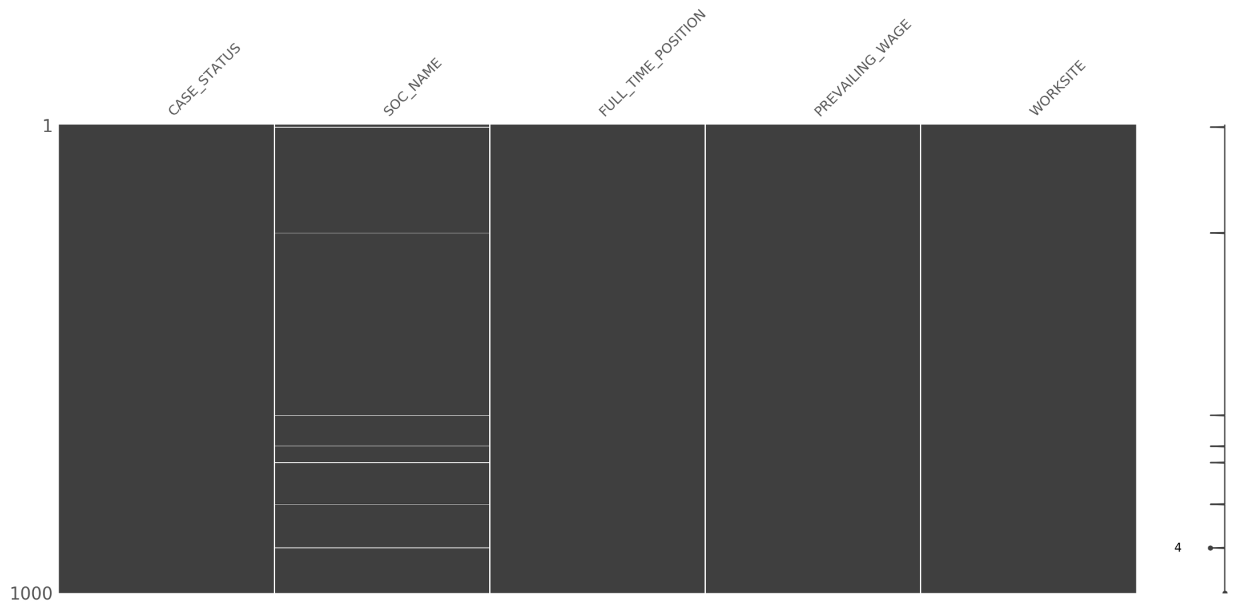
WORK_SITE: Location of the position.

Lon: Longitude of the WORK_SITE.

Lan: Latitude of the WORK_SITE.

2- Data Wrangling

For this project, I will be using five features of the dataset. These are case status, occupation name, position type, prevailing wage, and work site. Case status is the target which we are trying to predict given the other features.

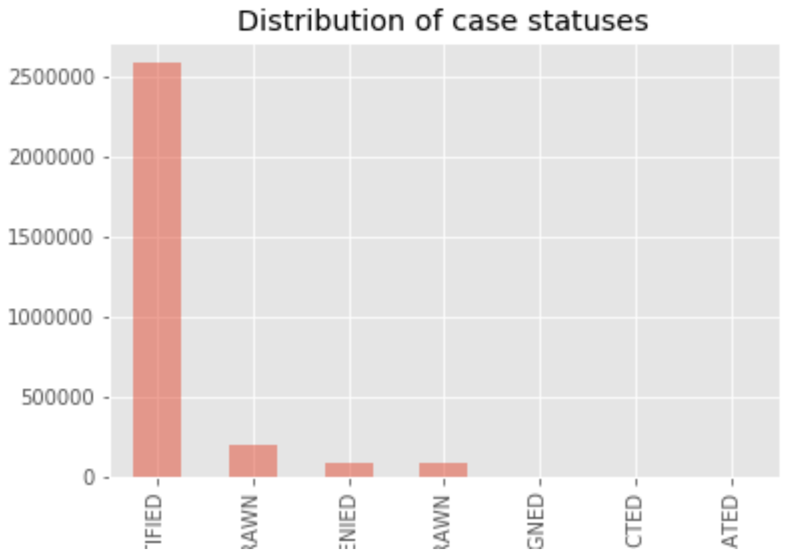


Before we start, it is important to check the missing values. We see that most of the missing values are in the occupation name column(SOC_NAME). Since we can't impute those values, it is better to remove them.

Current format of the worksite column is (City Name, State) for instance PHOENIX, ARIZONA. For this project we will focus on only state values. Therefore city names were removed.

There are extreme maximum and minimum wage values such as 6 billion dollars and $0. Nobody is making $6 billion and applying H1-B visa. Clearly some wage values are incorrect. Approximately, 12000 wages were below $25,000 or above $500,000 dollars, those records were removed. However, this dataset still has outliers that would affect the model.

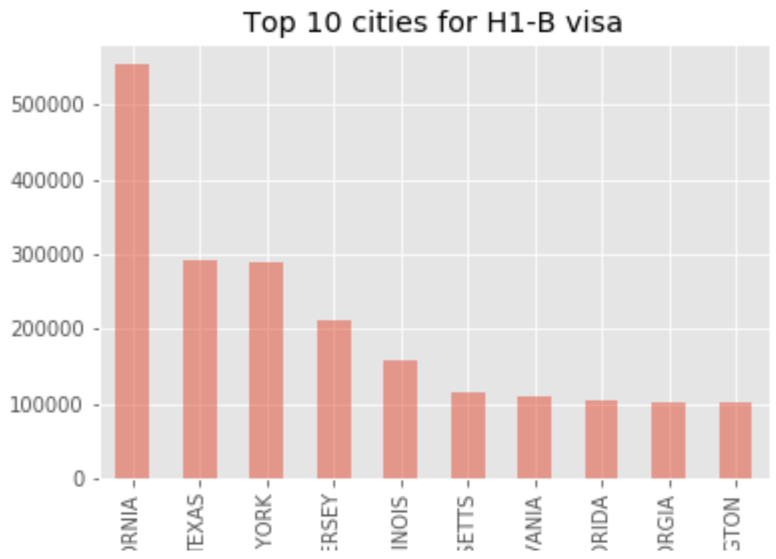
3- Data Exploration



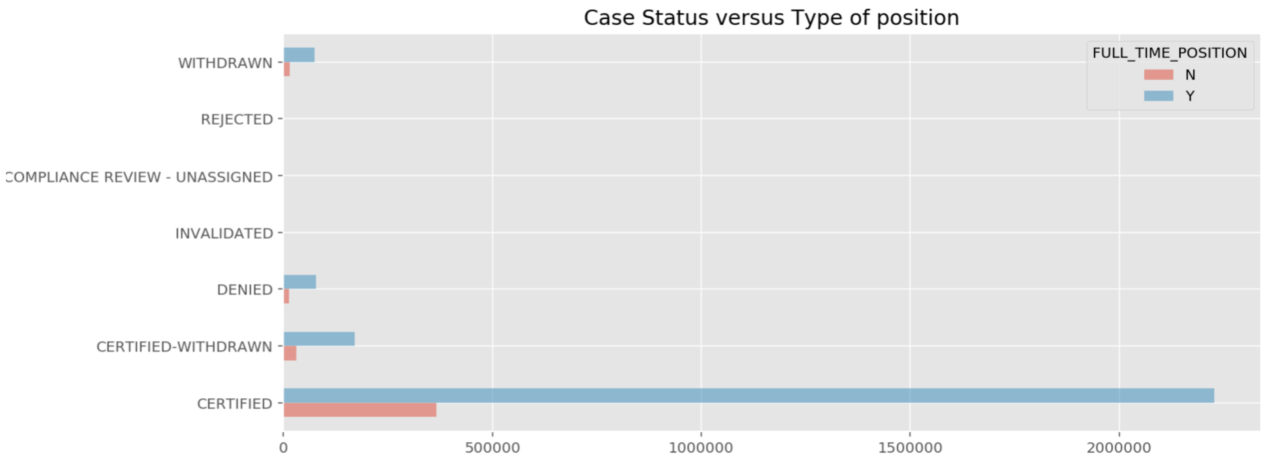
After cleaning the dataset, we have;

Number of positive cases: 2593332
Number of negative cases: 200845

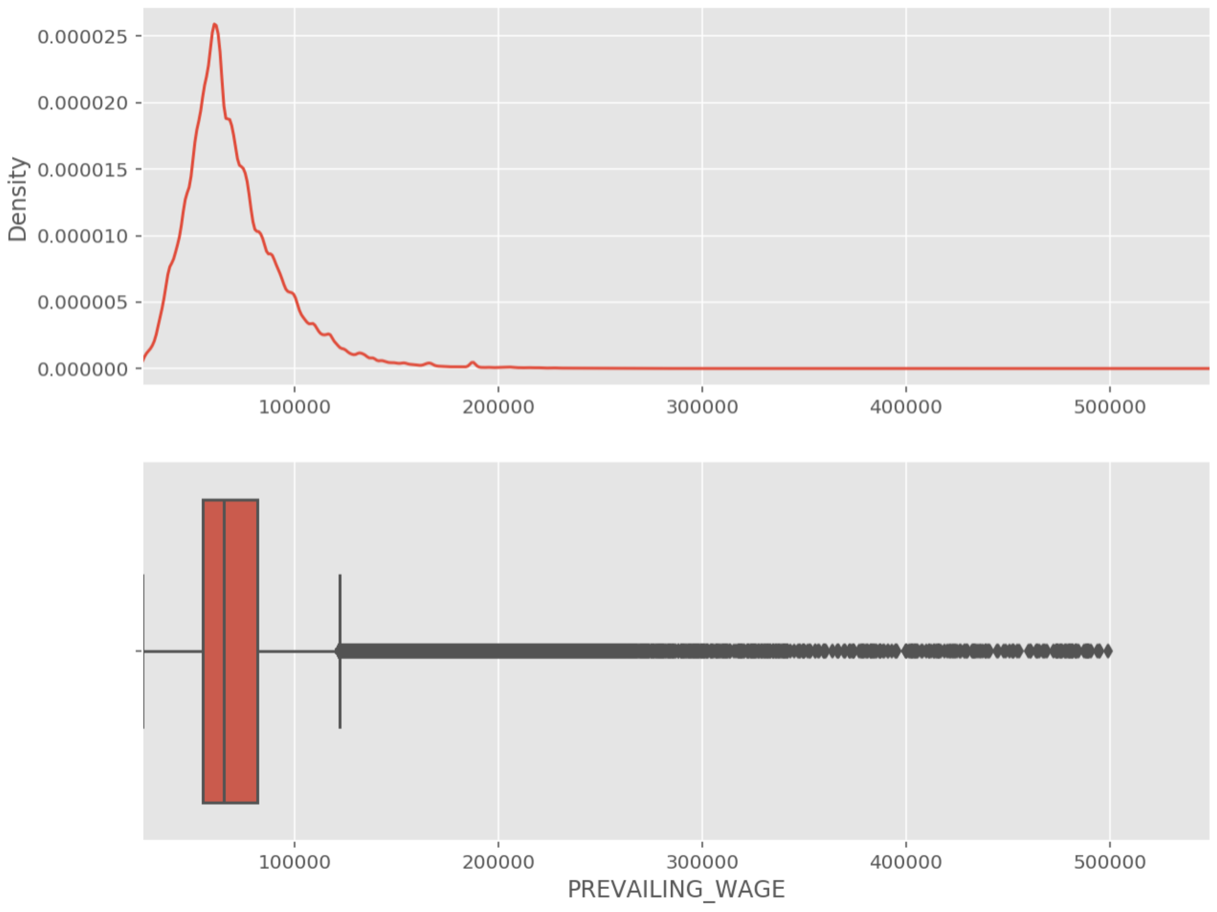
Note that, Certified is the only positive case type, all other types are uncertified and are considered as negative cases. That means 87% of the cases were positive.



California is the most popular place to apply H1-B visa. It is probably not case that Hollywood is in California but because Silicon Valley. Texas is also an attractive place for temporary workers.



In this plot, we see that some of the certified job are part time. First impression I had was part time positions wouldn't be approved. However, that is not correct.

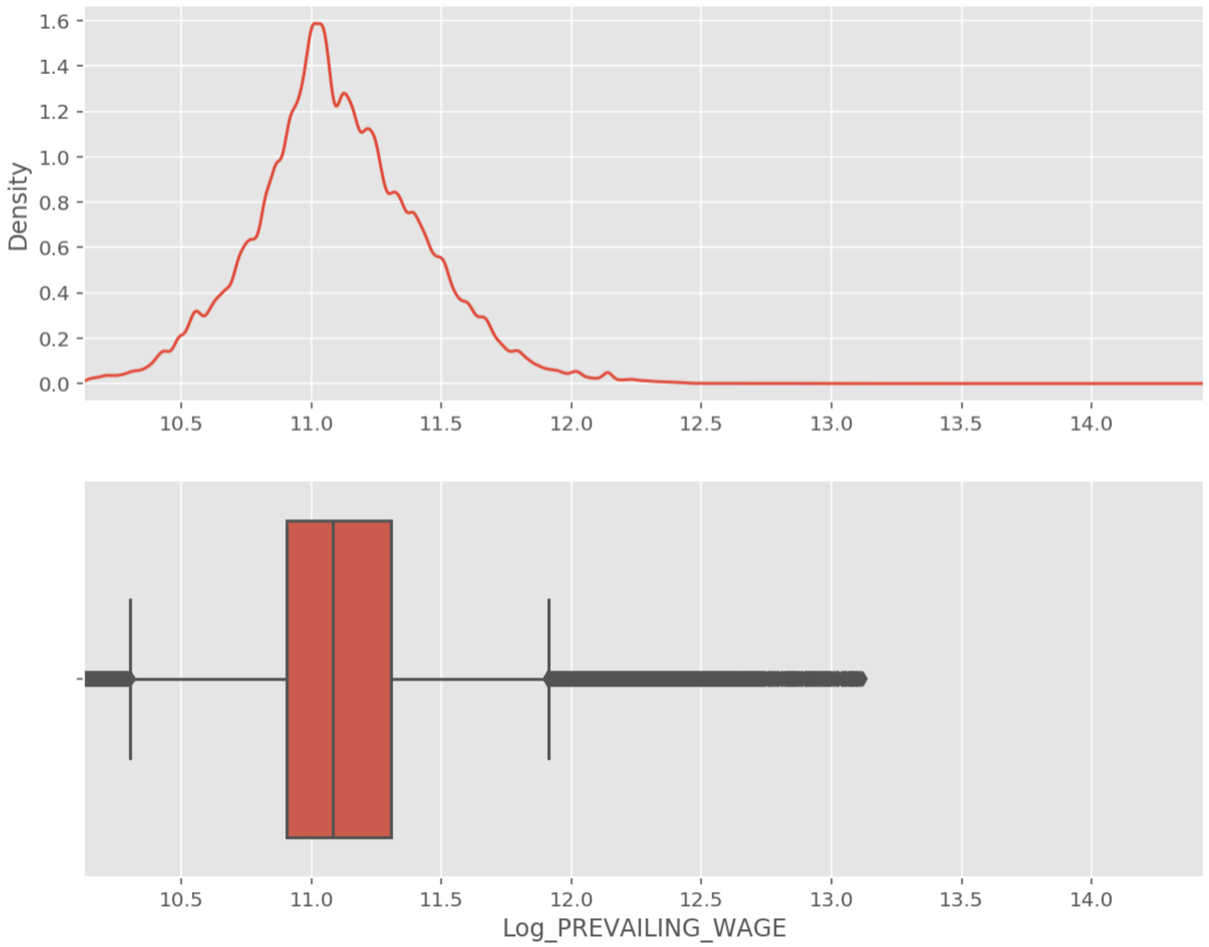


As we have pointed in the data wrangling section, wage column has extreme values. Even though I removed some of the data, it is still right skewed. Here we have two plots, the density plot and the box plot. This is a good way to view the data as we can see in the density plot (top) that there is some data points in the tails but it is difficult to see, however it is clear in the box plot.[2] At this point, I am not removing more data.

4- Data Transformation and Processing

For highly skewed features, it is always good to do transformation. Transformation will reduce the skewness and also helps us to see the patterns in the data much more clear way. Wage column has tail on the right and we will apply the logarithmic transformation on it. We will use Numpy's log transformation which is base *e*. [3]

$$X \, --> \, f(x) \, = \, log(x)$$



After transformation density plot looks smoother where wage is increasing and decreasing slowly. In box-plot, we can see the outliers, as well.[4]

In addition to performing transformations on the wage column that was highly skewed, it is often good practice to perform some type of scaling on numerical features. It is also as known as "Normalization". In this project Robust Scaler was applied to the log transformed wage column. It uses interquartile range so it is robust to the outliers.[5][6] In the previous plot, we have seen the interquartile range and outliers.

$$x_i \, - \, Q_1\,(x) \, / \, Q_3(x) - Q_1(x)$$

Applying normalization to the data does not change the shape of each feature's distribution; however, normalization ensures that each feature is treated equally when applying the model.

As we have seen before, There are 7 possible values for case status feature in the dataset and we reduced it to 2. Because only "Certified" has a positive meaning and rest of the statues have a negative meaning.

SOC_NAME feature has 1250 unique values and WORKSITE feature has 53. FULL_TIME_POSITION has 2 possible values. Once One-hot-encoding applied to these three columns, final features dimension is (2972646, 1307), meaning that 2,972,646 rows with 1307 columns.

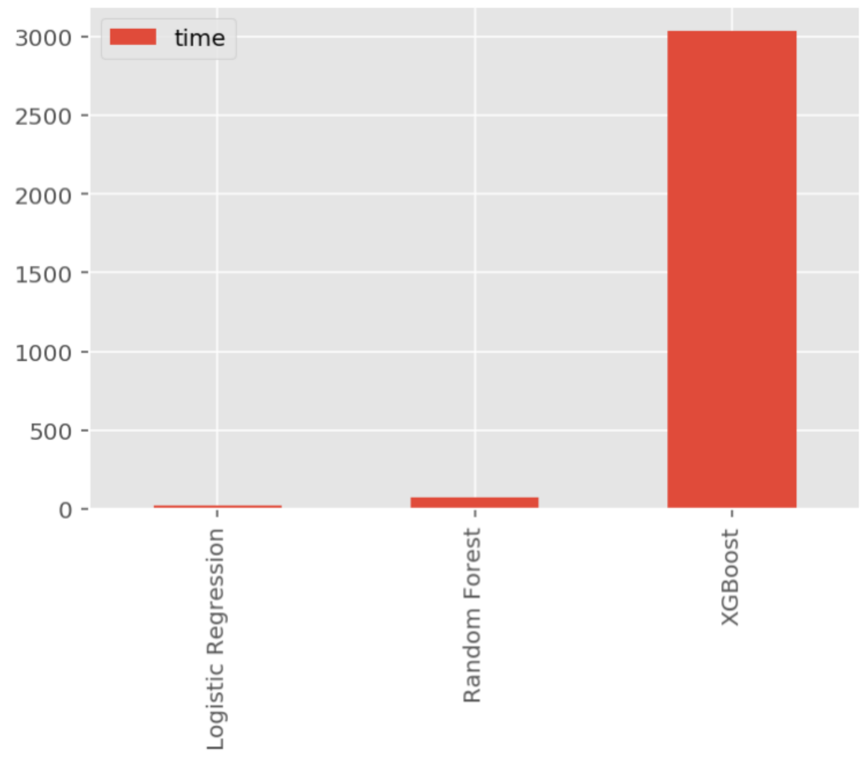
5- Data Modeling

For this project, I utilized three classifiers.

Logistic Regression: It performs a linear relationship between features and target with a sigmoid function. Performance is generally not competitive with best supervised learning algorithms. However, It is powerful when the outcome is two class. Training and prediction are fast. Tuning is easy. Interpretable. Overall, logistic regression is a good choice for this project.

Random Forest: It is an ensemble model. Meaning that it consists of multiple weak learners that come together and create a stronger learner. It is good choice for high dimensional datasets. Less likely to overfit. Because of its structure, training time is more than logistic regression. It is hard to interpret.[7]

XGBoost (Extreme Gradient Boosting): It is fairly a new highly complex model which can be learned [here.][8]



In term of training times, logistic regression was the fastest out of all classifiers. Random forest was okay but XGBoost was extremely slow due to its complexity.

6- Model Evaluation

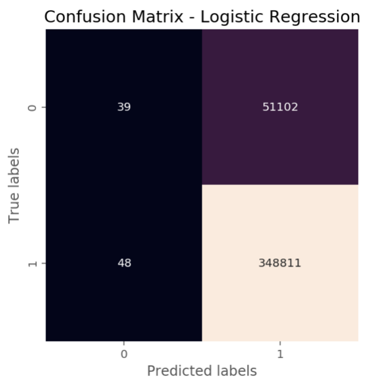
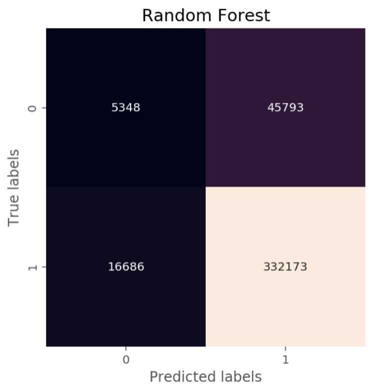
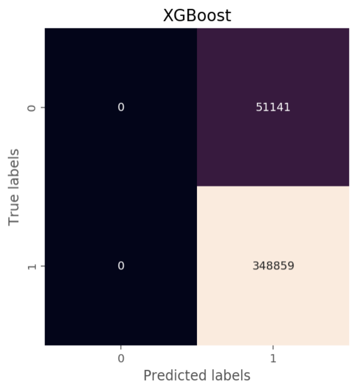
Naive Model
As we have seen in the data exploration part, we have highly imbalanced data. If we say, all the H1 visa applications approved, %87 of the time we would be correct. For classification tasks, accuracy is not the best way to compare the models. Instead, I will use f-0.5 score, confusion matrix and AUC score(Area Under Curve). I chose 0.5 for f-score to give equal weight to precision and recall. Below is the naive model evaluation scores.

Accuracy score	F-0.5 score
0.8721	0.8950

Naive predictor

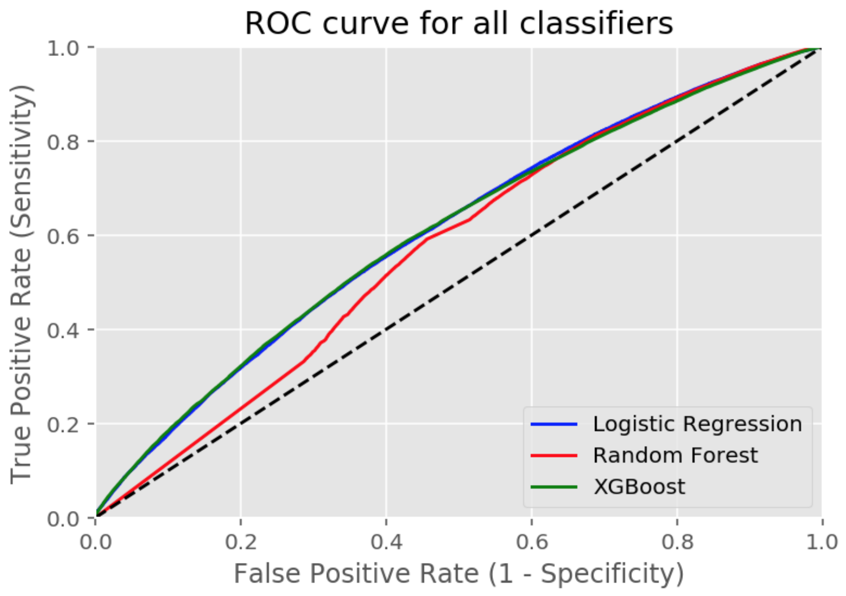
Finding the best model

H1-B visa prediction is a two class classification problem. Either a visa is certified or not certified. When the true label is Certified, how often is the prediction correct? (Recall). When the prediction is Certified, how often is the prediction correct? (Precision)

Logistic Regression	**Random Forest**	**XGBoost**
Accuracy: 0.8721	Accuracy: 0.8438	Accuracy: 0.8721
F-0.5 score: 0.8951	F-0.5 score: 0.8926	F-0.5 score: 0.8950
ROC AUC score: 0.6078	ROC AUC score: 0.5744	ROC AUC score: 0.6064
		

From the table above, we clearly see that in every single metric Logistic Regression and XGBoost do better job than Random Forest. At this point, we continue with these two and eliminate Random Forest.

Accuracy score is the same for both Logistic Regression and XGBoost. F-score and AUC (Area Under Curve) scores are also very close. The interesting part is the confusion matrices. We see that XGBoost model has no ability to detect uncertified cases. Basically, XGBoost classified the dataset just like the naive predictor. On the other hand, Logistic Regression model was able to catch 39 uncertified cases but it also misclassified 58 certified cases as uncertified.



Area Under Curve(AUC) score is a single number that allows us to determine which learner is better than the other. As we can see in the plot above, Logistic Regression and XGBoost are slightly better than Random Forest.

Previously, we saw that training time for XGBoost is significantly higher than Logistic Regression. Therefore, in this project Logistic Regression is my choice for hyperparameter tuning.

Hyperparameter Tuning

Logistic Regression is a kind of model that creates linear relationships. Therefore tuning is easy and doesn't effect a lot how model behaves. I have tried to optimize three parameters 1- C, 2- penalty, 3- tol. After doing a Grid Search on testing dataset, we see that F-score has not changed.

Unoptimized model
Accuracy score on testing data: 0.8719
F-score on testing data: 0.8948

Optimized Model
Final accuracy score on the testing data: 0.8717
Final F-score on the testing data: 0.8948

References:

[1] https://www.kaggle.com/nsharan/h-1b-visa
[2] http://stamfordresearch.com/outlier-removal-in-python-using-iqr-rule/
[3] https://docs.scipy.org/doc/numpy-1.14.0/reference/generated/numpy.log.html
[4] http://stamfordresearch.com/outlier-removal-in-python-using-iqr-rule/
[5] http://benalexkeen.com/feature-scaling-with-scikit-learn/
[6] http://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py
[7] https://www.youtube.com/watch?v=D_2LkhMJcfY
[8] http://xgboost.readthedocs.io/en/latest/model.html