Predicting H-1B Visa status

CAPSTONE PROJECT
Machine Learning Engineering

Numan Yilmaz
May 19th, 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. For an employer, H-1B visa process starts with finding an employee outside of the USA. After that, employer offers the job to the employee and file a petition to the US Immigration Office. Response for a petition depends on the demand, job market, time of the submission, employee qualification etc and is about 2 - 6 months. Entire process of bringing an employee from abroad could take up to 24 months. Duration of stay in the US is between 3 to 6 years after extensions.

Problem Statement

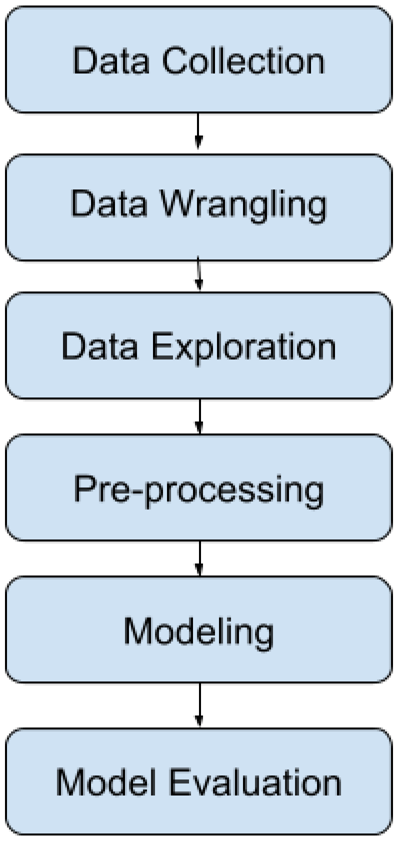
H-1B visas are taking long time to process which could be 1-2 years. According to paysa.com research, employees are staying at the big tech companies less than 2 years.[12] After all, does it really make sense for big tech companies to go through visa process for an employee that may or may not be a 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.

A group of student from UCSD has done a similar project in past.[9] Their approach to the problem was different than this project. To overcome imbalance dataset, they have retained 100,000 positive and 100,000 negative cases, total of 200,000 samples. Therefore, their baseline model f-score and accuracy were a lot smaller. They utilized four algorithms (Gaussian Naive Bayes, Logistic Regression, . Random Forest Classifier, AdaBoost Classifier). They got interesting results. Besides predicting case status, there are a few other projects has done related to this dataset. For example, predicting wages based on the text analysis on the dataset such as job title, company name, employer name, visa status and work place city. [10] There are also many more exploratory data analysis projects can be found in Kaggle website.[11]

Solution Statement

In order to predict if a foreign labor might get H-1B visa, I will create a model that takes occupation code, job title, prevailing wage, position type, work site as inputs and based on these inputs, the model is going to predict the case status as output. This is a two-class classification model. The output from the model could be Certified or not Certified.

After data cleaning and munging, I will apply log transformation to the dataset to reduce the skewness. For data normalization, I will use Robust Scaler, which is a better choice for highly imbalanced dataset like this.



I will make use of three different machine learning algorithms. The first will be the logistic regression. It is easy to implement and interpret. It is very fast to train and generally perform great on the test dataset. Second algorithm will be Random Forest. It is an ensemble model, meaning that it consist of multiple decision trees. It takes times to train the model. The nice thing about Random Forest is that it gives us the importance of each feature in the dataset. The third algorithm will be XGBoost which is a more sophisticated algorithm. It is hard to train and tune the model but gives us to find non-linear patterns in the dataset.

Finally, model evaluation using metrics for classification problems. And then, hyperparameter tuning to improve score to get better results.

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.

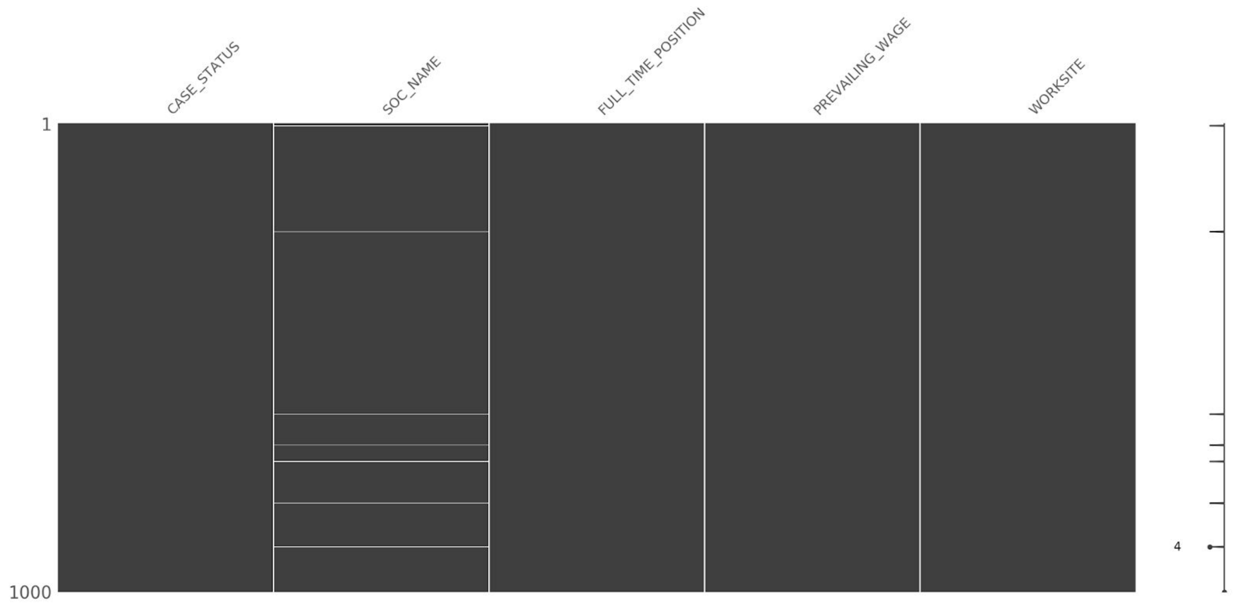
WORK_SITE: Location of the position.

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.

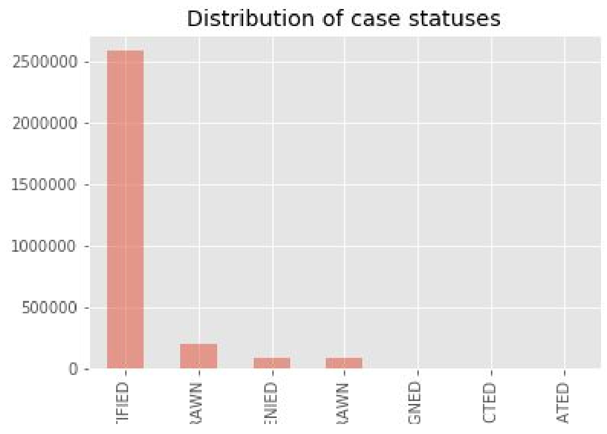


Before we start, it is important to check the missing values. We see that most of the missing values are in the occupation name column(SOC_NAME). Since we can't impute those values, it is better to remove them.

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.

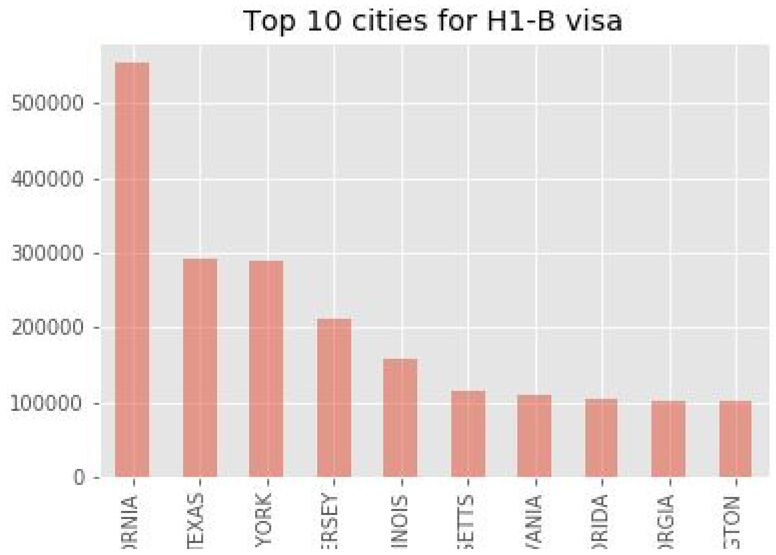
3- Data Exploration



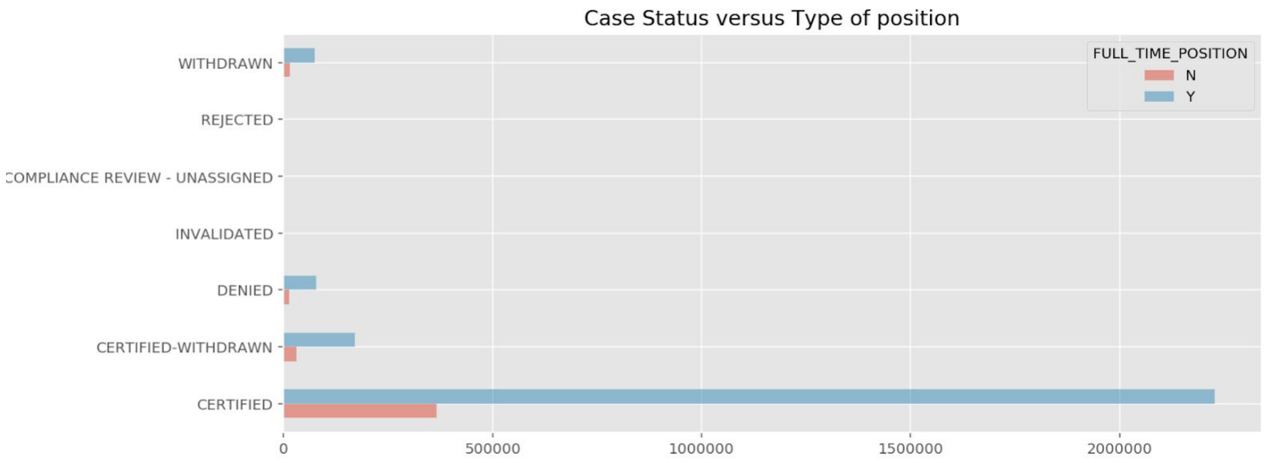
After cleaning the dataset, we have;

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

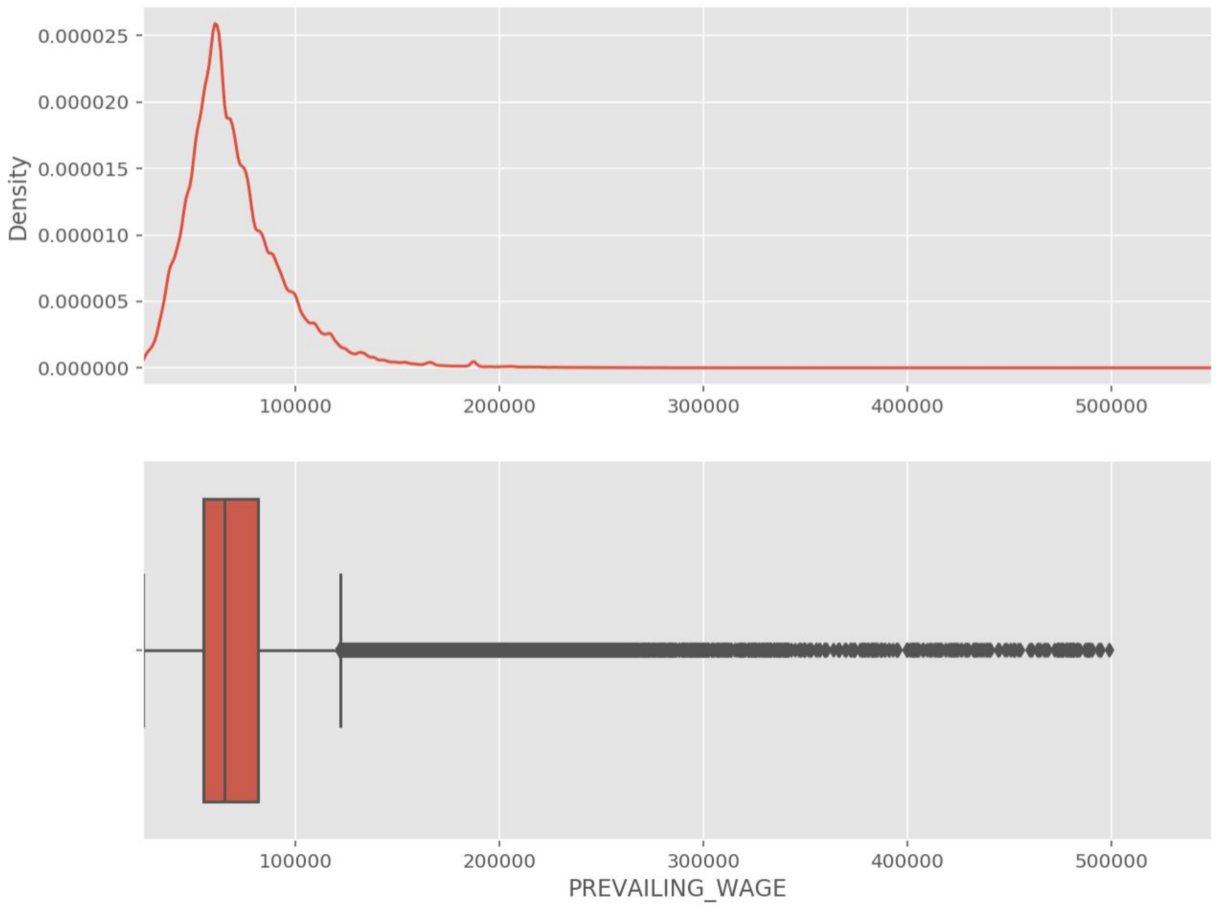
Note that, Certified is the only positive case type, all other types are uncertified and are considered as negative cases. That means 87% of the cases were positive.



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.

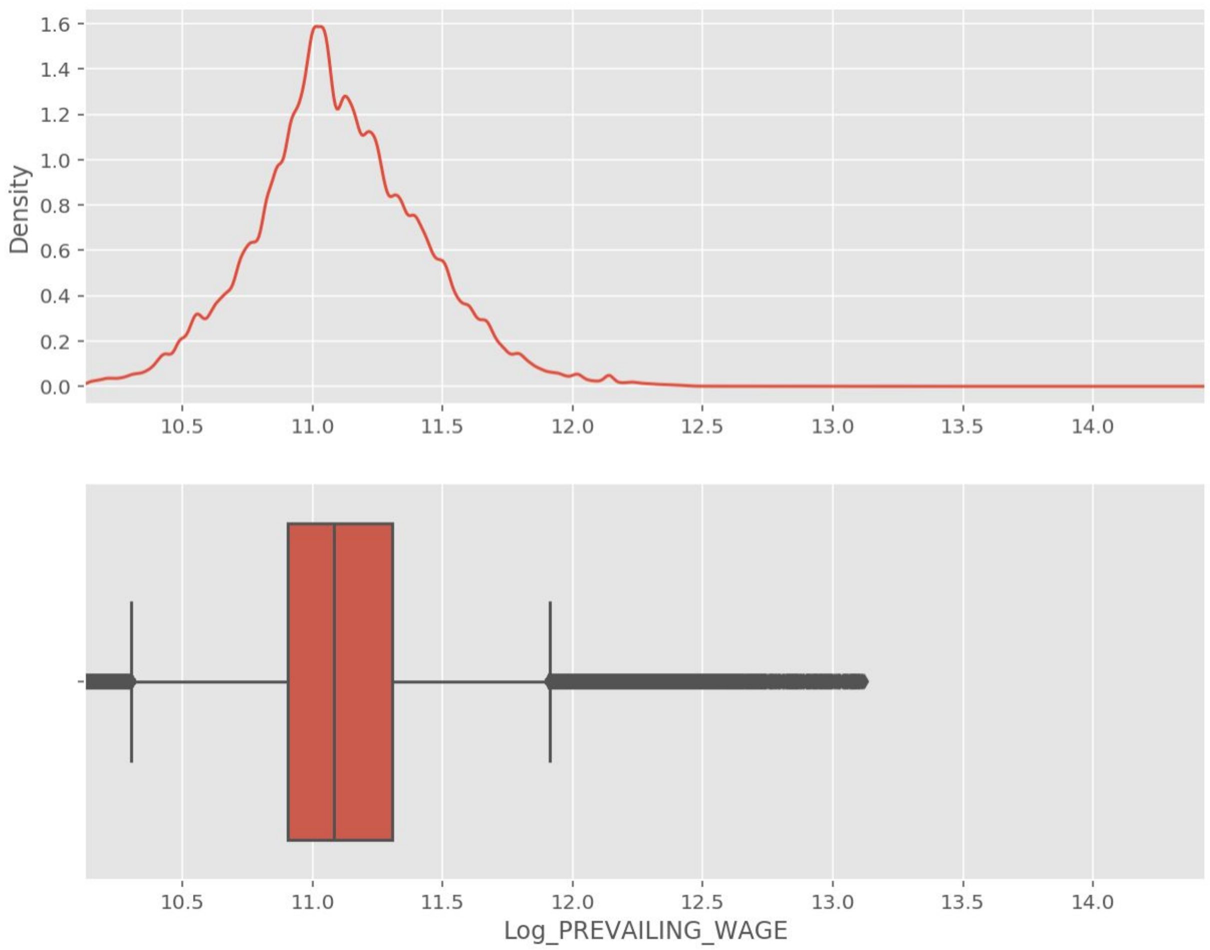


As we have pointed in the data wrangling section, wage column has extreme values. Even though I removed some of the data, it is still right skewed. Here we have two plots, the density plot and the box plot. This is a good way to view the data as we can see in the density plot (top) that there is some 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]

$$X \text{ --> } f(x) = log(x)$$



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

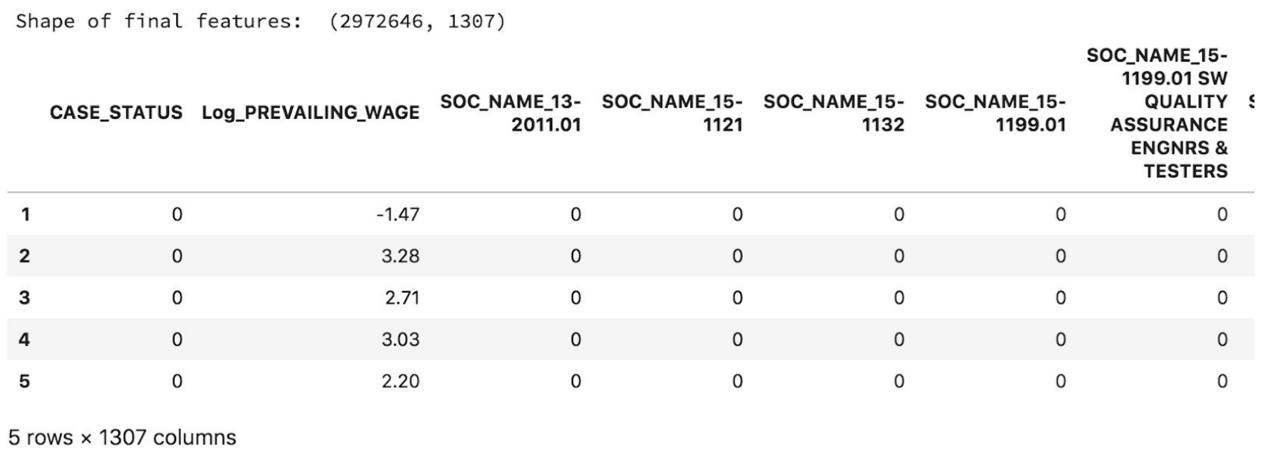
In addition to performing transformations on the wage column that was highly skewed, it is often good practice to perform some type of scaling on numerical features. It is also as known as “Normalization”. In this project Robust Scaler was applied to the log transformed wage column. It uses interquartile 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.



```
Shape of final features:  (2972646, 1307)
```

	CASE_STATUS	Log_PREVAILING_WAGE	SOC_NAME_13-2011.01	SOC_NAME_15-1121	SOC_NAME_15-1132	SOC_NAME_15-1199.01	SOC_NAME_15-1199.01 SW QUALITY ASSURANCE ENGNRS & TESTERS
1	0	-1.47	0	0	0	0	0
2	0	3.28	0	0	0	0	0
3	0	2.71	0	0	0	0	0
4	0	3.03	0	0	0	0	0
5	0	2.20	0	0	0	0	0

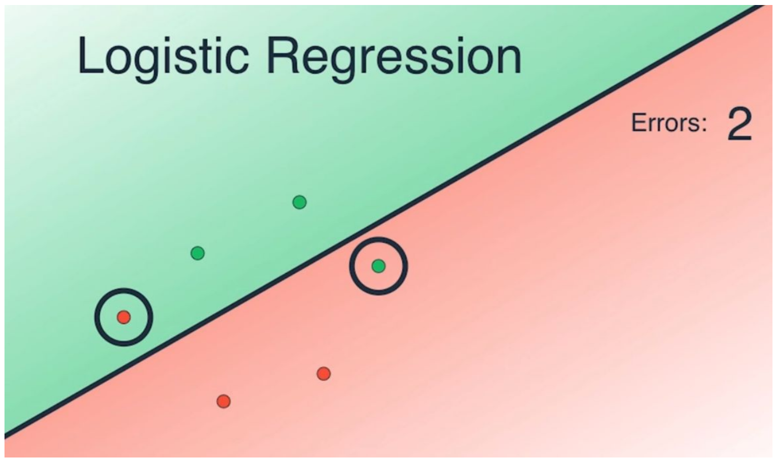
5 rows × 1307 columns

5- Data Modeling

Algorithms

For this project, I utilized three classifiers.

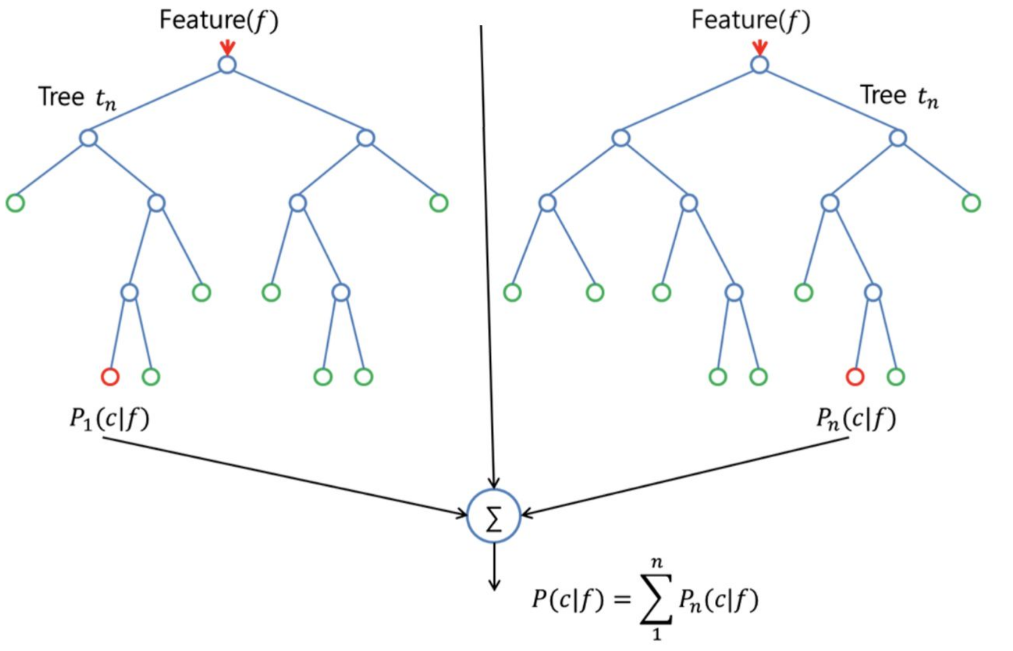
Logistic Regression: It performs a linear relationship between features and target with a sigmoid function.



Let's consider this image. The model puts every single data into this coordinate based on features. Green points represent certified petitions, red is the uncertified petitions. The model doesn't know how to draw a line that separates these two classes. So, it draws a random line and calculates the error. Error is the misclassified petitions. It moves the line and calculates the

error again. This process continues until it finds the best line that separates these two classes with the least error. When we want to predict a new data point(a new petition), it puts that data point into the coordinate and checks which area (red or green) it belongs to. Based on where the data is, it predicts and tells us if that petition is certified or not.
Performance of logistic regression is generally not competitive with the 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.

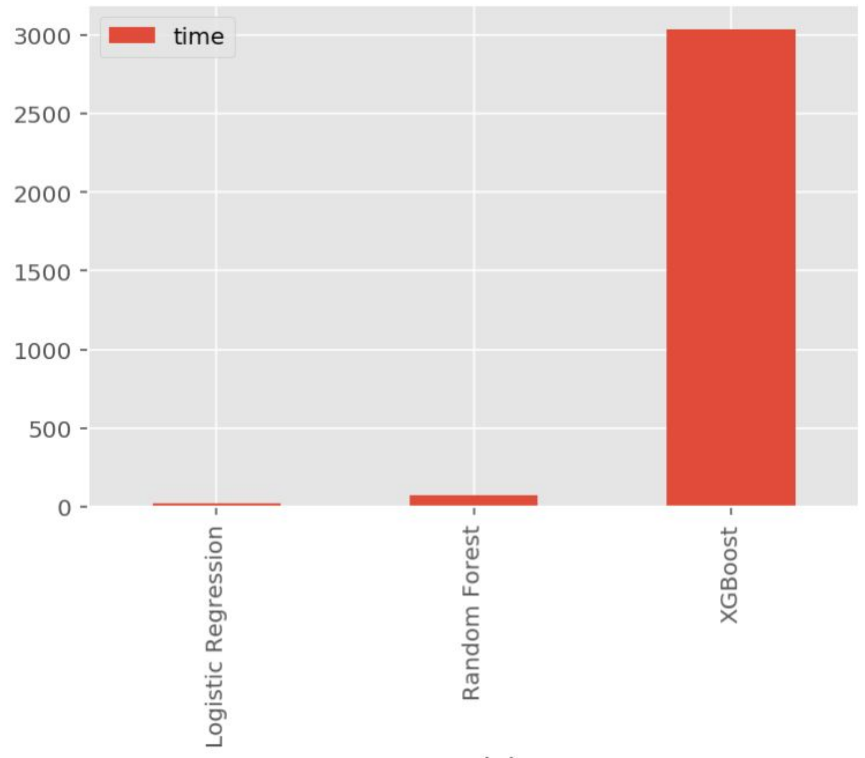
Random Forest: It is an ensemble model. Meaning that it consists of multiple weak learners that come together and create a stronger learner. Below is a demonstration of how a random forest would look like with two trees.



Credit: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

Random Forest creates many decision trees and it is good choice for high dimensional datasets. Less likely to overfit. Because of its structure, training time is more than logistic regression and it is hard to interpret.[7]

XGBoost (Extreme Gradient Boosting): It is fairly a new, highly complex model[8]. Before we learn how it works, we should know what **Boosting** is. Boosting is an ensemble method that creates strong learners based on the weak learners. It creates a weak model and then corrects the errors in the next model until the data is predicted accurately. XGBoost is using the same technique with gradient descent to minimize the error in the model. It is supported for both classification and regression.



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.

Implementation

Model implementation was not so difficult after data cleaning at the beginning. There was a SOC_NAME **<FONT><FONT>CARPINTEROS</FONT></FONT>** had to be removed. I went back to data wrangling part and remove that data and rerun the whole notebook. Another difficulty was the sample size. The dataset has 3 million records, computation power I have was not sufficient to use all the data even for logistic regression. Therefore, I used 500,000 samples in this project. In addition, XGBoost was extremely slow in every single part of the project.

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,

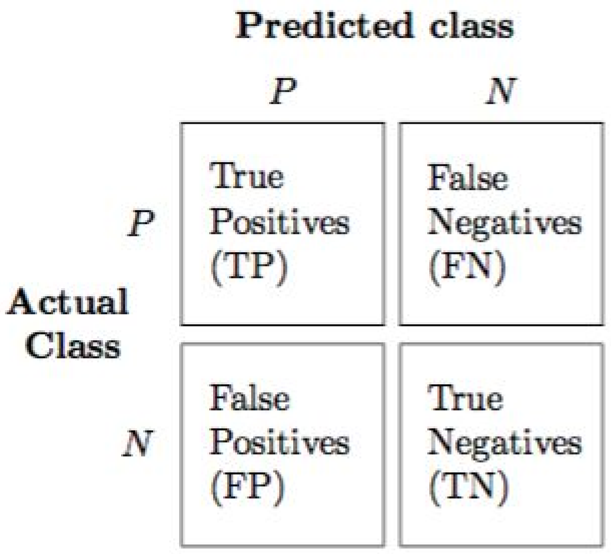
accuracy is not the best way to compare the models. Instead, I will use f-0.5 score, confusion matrix and AUC score(Area Under Curve). I chose 0.5 for f-score to give twice weight to precision. Below is the naive model evaluation scores.

Accuracy score	F-0.5 score
0.8714	0.8944

Naive predictor

Metrics

Confusion Matrix: A confusion matrix is an excellent way to visualizate how the model is doing in term of predictions and actual values. For binary classification problems, confusion matrix would like this:



Reference: [13]

True Positives (TP): We correctly predicted that the visa is certified
True Negatives (TN): We correctly predicted that the visa is not certified
False Positives (FP): We incorrectly predicted that the visa is certified (a "Type I error")
False Negatives (FN): We incorrectly predicted that the visa is not certified (a "Type II error")

Precision: When a positive value is predicted, how often is the prediction correct?

$$TP / TP + FP$$

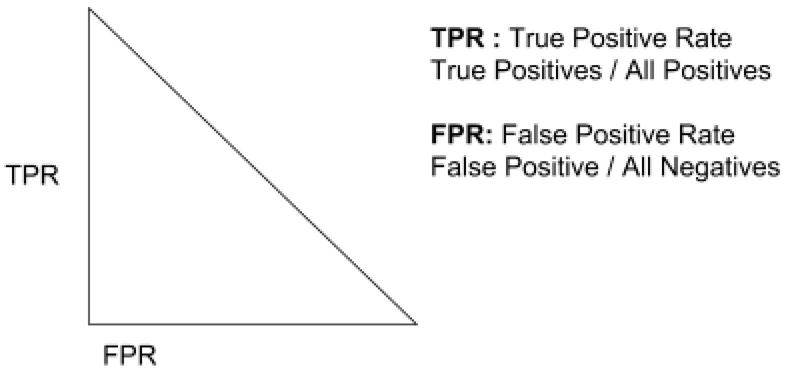
Recall: When the actual value is positive, how often is the prediction correct?

TP / TP + FN

F-score: Precision and recall are very powerful accuracy method. However, having two numbers to compare multiple models is not practical. Instead, we can use a single number.

$$F_{beta} = (1 + beta^2)\ \frac{presicion * recall}{(beta^2 * precision) + recall}$$

ROC AUC Score: It is a great way to visualize performance of classifiers for binary class problems. To give equal weight to TPR and FPR, I chose 0.5 for f-score.

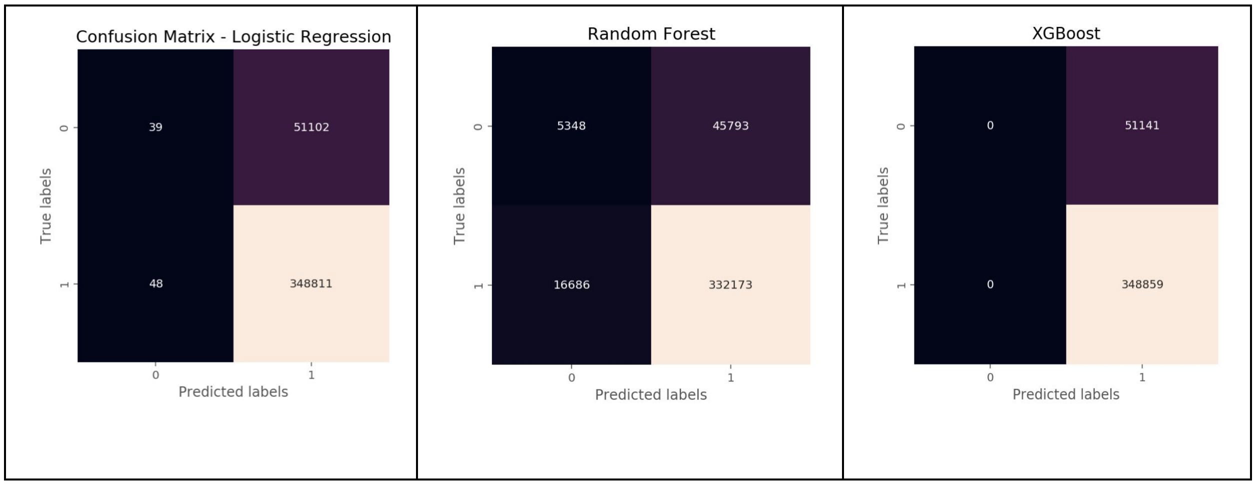

TPR
FPR
TPR : True Positive Rate
True Positives / All Positives
FPR: False Positive Rate
False Positive / All Negatives

AUC (Area Under Curve) score is another single number to compare multiple classifiers.

Finding the best model

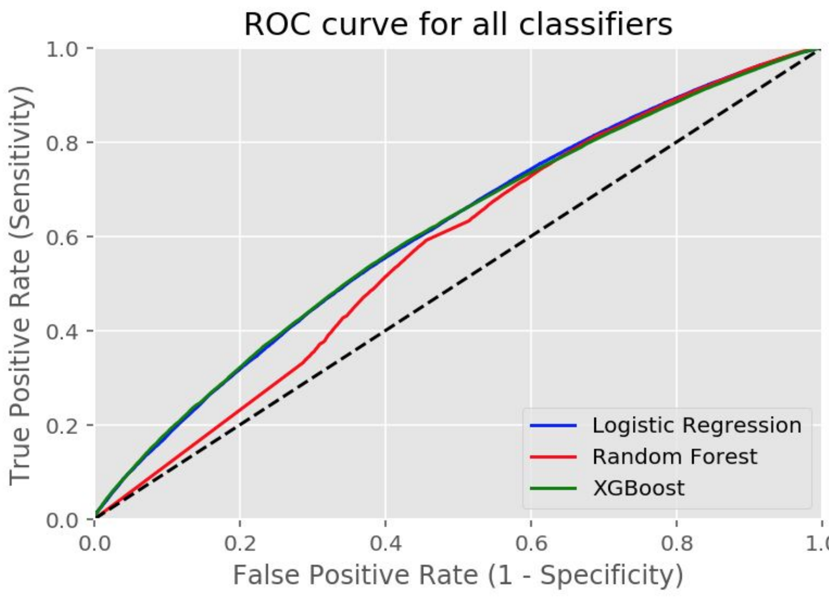
H1-B visa prediction is a two class classification problem. Either a visa is certified or not certified. When the true label is Certified, how often is the prediction correct? (Recall). When the prediction is Certified, how often is the prediction correct? (Precision)

Logistic Regression	**Random Forest**	**XGBoost**
Accuracy: 0.8717	Accuracy: 0.8436	Accuracy: 0.8717
F-0.5 score: 0.8947	F-0.5 score: 0.8923	F-0.5 score: 0.8946
ROC AUC score: 0.6041	ROC AUC score: 0.5747	ROC AUC score: 0.6029



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: 0.8726
F-score on testing data: 0.8954

Optimized Model
Final accuracy score on the testing data: 0.8726
Final F-score on the testing data: 0.8954

Robustness of Optimized Model

Finally, I have trained the best model with 100, 1000, 10000, and 100000 training points and tested on the test set. I got some interesting results.

Sample Size	Accuracy Score	F-0.5 Score
100	0.8723	0.8953
1000	0.8723	0.8954
10000	0.8726	0.8954
100000	0.8726	0.8954
500000	0.8726	0.8954

I trained the model with 500000 data points. However, only 10000 data points were enough to reach the same accuracy and f score.

Justification

The final optimized Logistic Regression model's accuracy and f-score on testing data can be seen below. Based on these results, the model is not enough the solve the problem.

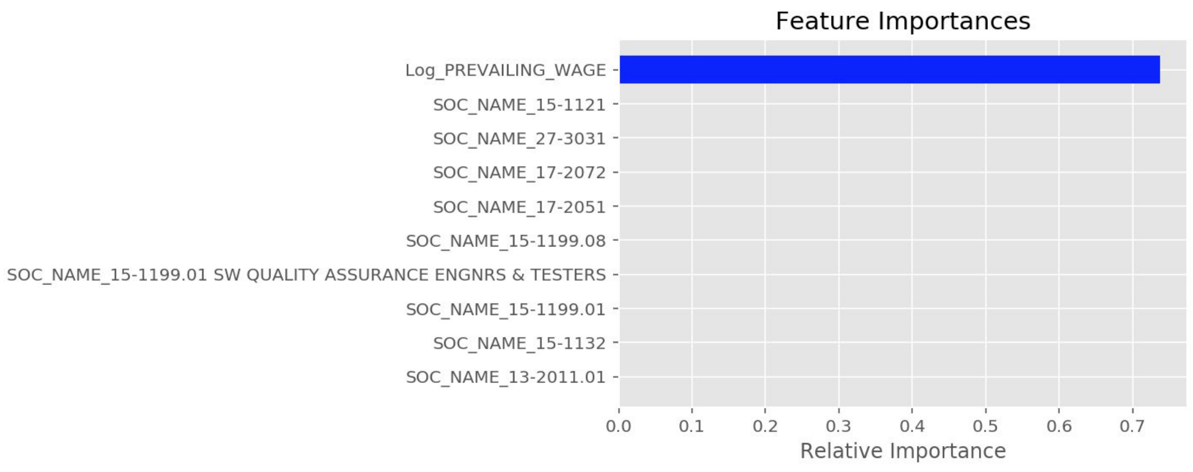
	Accuracy score	F-0.5 score
Baseline model	0.8728	0.8956
Optimized Logistic Regression model	0.8726	0.8954

Both accuracy and f-score was slightly lower than baseline model on the test dataset.

7- Conclusion

Free Form Visualization

Below I have displayed the top ten most important feature that affect how the model makes decision. As expected prevailing wage is by far the most important feature. On the other hand, would expect to see some state names on the list. Apparently, job titles are more effective than states.



Reflection

One aspect of the project was to work with high dimensional dataset. There were 2,972,646 samples with 1307 features. At model creation stage, I fed all dataset into the Logistic Regression model which is generally faster to train compare to other algorithms. However, with the hardware of my computer, it was not possible to train the model. I reduced the dataset to 1 million samples. But it was impossible to train XGBoost. Then, I chose 100,000 data points. It was definitely faster to train but f-score was lower than before. Finally, I decided to go with 500,000 data points.

In data exploration part, We observed that most of the cases are certified. That was a surprising result to me. This means that companies are carefully applying H1-B visa.

Improvement

Employer name was ignored in this project. However, it might be a feature that is the affecting case status. But it would significantly increase the number of features. Another improvement could be for the wage values. We have seen the distribution of wages and there were outliers. I have remove some of the outliers. More outliers could be removed. Also, for wage values, we could split into buckets.

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] https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html
[9] https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a054.pdf
[10] https://blog.bigml.com/2013/10/01/using-text-analysis-to-predict-h1-b-wages/
[11] https://www.kaggle.com/nsharan/h-1b-visa/kernels
[12] https://www.paysa.com/blog/wp-content/uploads/2017/07/DisruptorsA8.png
[13] https://rasbt.github.io/mlxtend/user_guide/evaluate/confusion_matrix/
https://github.com/justmarkham/scikit-learn-videos/blob/master/09_classification_metrics.ipynb
https://stackoverflow.com/questions/44101458/random-forest-feature-importance-chart-using-python