**David Geoffrey Agumya**

**Data Mining Final Project Paper**

**December 10th 2017**

1. **INTRODUCTION**

For this data mining project, I am working with H1-B Visa Petition Data from H1-B Visa Petition filed with the Office of Foreign Labor Certification between the years 2011 and 2016. The H-1B is an employment-based, non-immigrant visa category for temporary foreign workers in the United States. For a foreign national to apply for H1-B visa, an US employer must offer a job and subsequently petition for H-1B visa on behalf of the employee with the US immigration department. This is the most common visa status applied for and held by international students once they complete college/ higher education (Masters, PhD) and work in a full-time position.

The problem that I am trying to solve is “Can we predict the outcome of H1-B visa petitions based on information about the employee and employer who submitted the application? “.

This would be useful to employees that are thinking about applying for H1-B, they could gauge if they will get a successful outcome or not. Since the cost of applying for a visa is so high, such information will help employees and employers in the application process. Secondly, international students who are thinking about higher education in the U.S can use the model and analysis to decide what professions to go into and what geographical location would be best to be in to increase chances of getting hired and having their petitions certified. Moreover, policy makers and politicians can use the model and analysis to find out what skills foreign labor brings to the national market and how to leverage foreign labor to advance the country’s economic goals. The information for prevailing wage can even help companies and employees negotiate compensation and monitor the trend of compensation for a position or job.

The models and analysis from this project would benefit, international students, foreign employees seeking to work it United States of America. Policy makers and U.S companies.

1. **DATA**

I collected all the data I needed from Kaggle online dataset repo. I downloaded it as a csv file with 3,002,458 instances. Each instance had 10 features.

* 1. Original Features were:

1. CASE\_STATUS -> Outcome of the Visa Petition Process
2. EMPLOYER\_NAME -> Name of employer sponsoring employee on Visa Petition Application
3. SOC\_NAME -> A Standard Occupation Code per SOC Classification system.
4. JOB\_TITLE -> Job Title of Employee
5. FULL\_TIME\_POSITION -> Y = Full Time, N = Part Time
6. PREVAILING\_WAGE -> Prevailing Wage for the job being requested for temporary labor condition. The wage is listed at annual scale in USD. The prevailing wage for a job position is defined as the average wage paid to similarly employed workers in the requested occupation in intended employment. The prevailing wage is based on the employer’s minimum requirements for the position.
7. YEAR -> Year in which the H1-B visa petition was filed
8. WORKSITE -> City and State Information of foreign workers intended area of employment
9. Lon -> Longitude coordinates of worksite
10. Lat -> Latitude coordinates of worksite

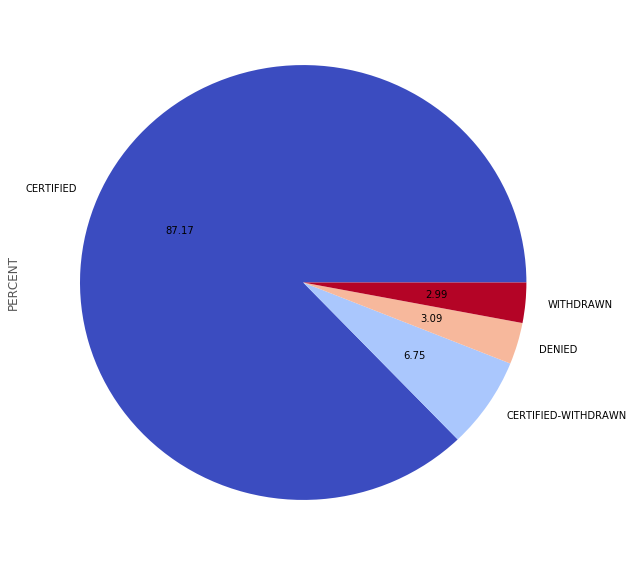
After reading in the data and analyzing the features available I decided that I would have to remove some features because I would not need them. The features that I deemed as unnecessary were SOC\_NAME. SOC\_NAME captures the same information as JOB\_TITLE, moreover this feature had the highest number of missing feature values. I also removed *Lon* and *Lat* since these coordinates were not used in my analysis and modeling and they also captured the same information as WORKSITE.

This Data was originally collected from the Office of Foreign Labor Certification. [The Office of Foreign Labor Certification](https://www.foreignlaborcert.doleta.gov/performancedata.cfm) compiles all application information. This reflects all the information of Visa Petition Applicants for the years between 2011 and 2016. This data is gathered annually. This data was not in the right format for quick analysis. A set of transformation was applied by another unknown data scientist to derive the data that I use for my analysis. The transformation allows for more efficient analysis and modeling. After reading in the data I shortly realized that the number of instance was too large to process the data for classification. So, for classification I sampled 20, 000 instances out of the roughly 3 million instances using sampling without replacement. I maintained the original distribution of the target outcomes. I did this because my local machine could not process more that 20,000 instances without significantly slowing down. For Data exploration, however is use all the instances available without missing feature values.

1. **DATA EXPLORATION / UNDERSTANDING**

With Data Exploration, I tried to tell a story. To look for meaningful analysis that would benefit the stakeholders I mentioned in the introduction. In Data exploration, I used all the data available without sampling. Here are several questions, whose answers I was able to derive from the analysis phases of this project.

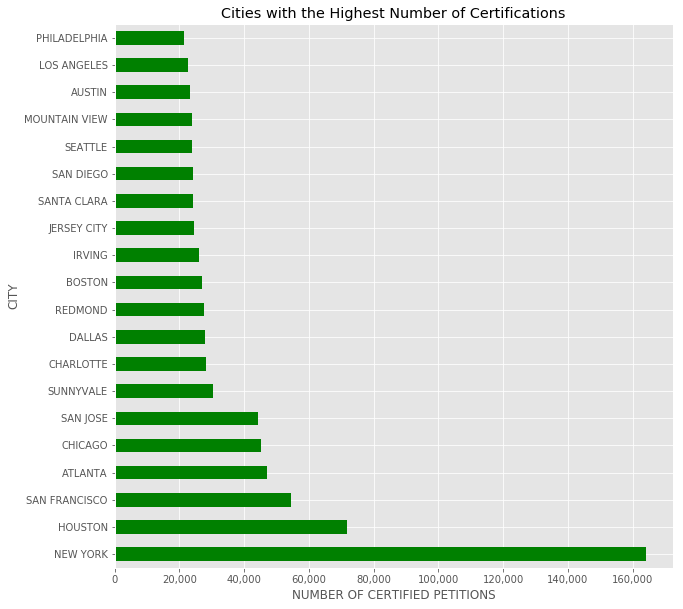
* 1. On Average, how many petition applications are certified between 2011 and 2016



Per the data, 87.17 percent of all application are certified, 2.99 are usually withdrawn. 3.09 percent are denied and 6.75 percent of petition application between 2011 and 2016 are certified withdrawn.

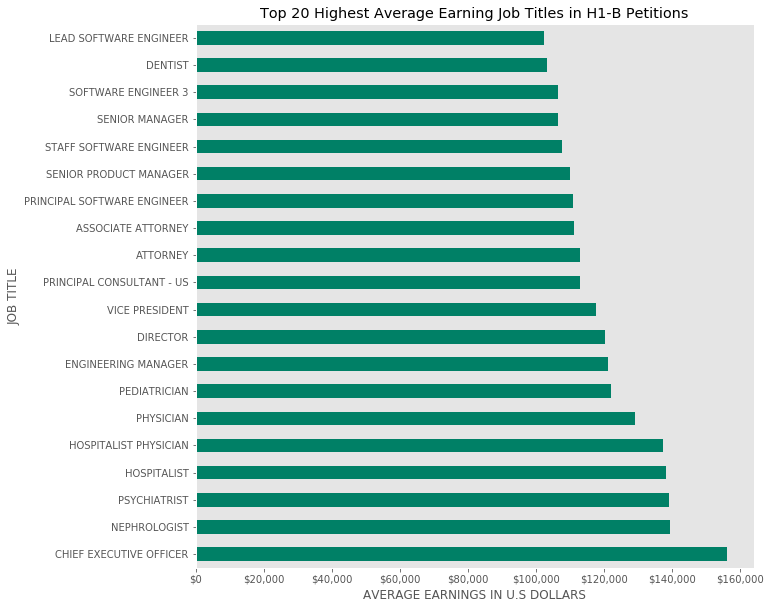
A brief note about the outcomes of the applications. CERTIFIED outcomes are application where the petition is granted and the applicant can go ahead and apply for H1-B Visa from the Immigration Authorities. CERTIFIED-WITHDRAWN outcomes are situations where the petition is granted (certified) and the employee can go ahead and apply for an H1-B Visa, however if the employee leaves the employment of the employer who sponsored the application then their application case status is changed from CERTIFIED to CERTIFIED-WITHDRAWN. WITHDRAWN outcomes are situations where the application is removed from consideration before a final decision is made as to whether to CERTIFY the application of DENY it. DENIED outcome is a situation where the application is DENIED.

* 1. Of the Applications that were certified, which cities had the highest number of applicants that were certified?



From this we can see that most of the cities represent states with powerful and diverse economies. Some of these are Texas (Austin, Houston), New York (New York City), California (Los Angeles, San Francisco, San Jose, Sunnyvale, Irving) Washington (Seattle, Redmond). Part of the reason the top 5 are there is also because these are cities where most of the applications come from. So, there is a correlation there. For an foreign employee, this analysis would useful to help him or her decide which cities or states to focus their job search in if they want to work in the U.S with a goal of settling in a city where they stand a better chance of getting certified to get their H1-B Visa.

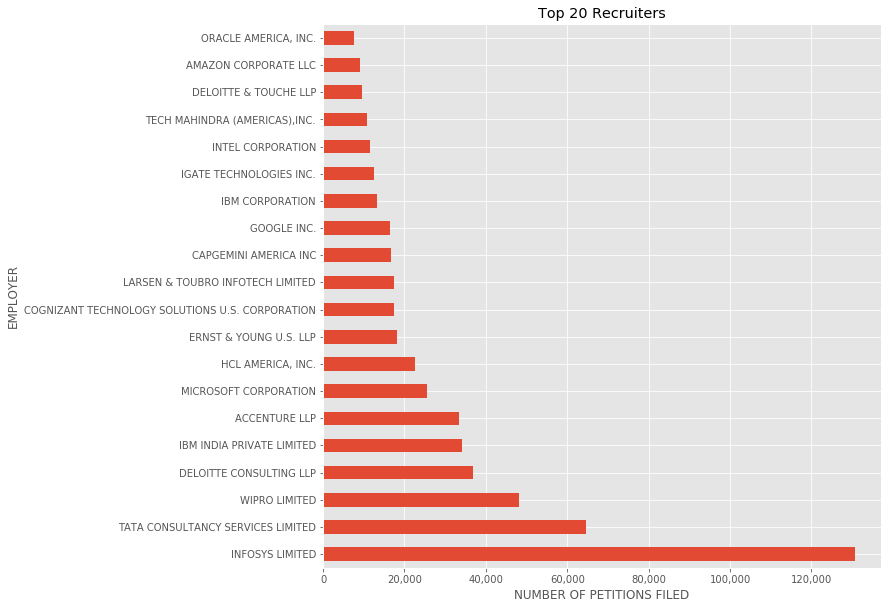
* 1. Of all the applications where a job title occurred at least 1000 times, what were the top 20 highest paying jobs?



I choose to use a 1000 count minimum threshold for support of this analysis to get a more realistic mean of the prevailing wage in the profession.

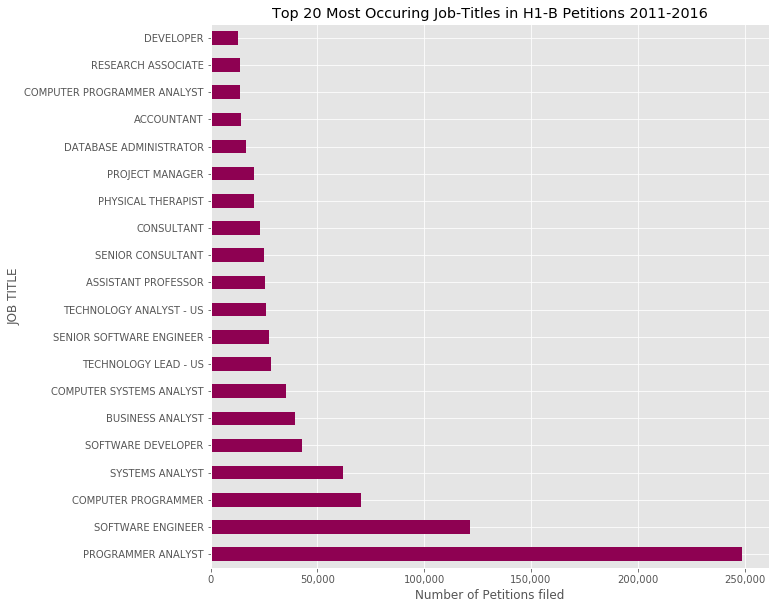
From this analysis, we see that for foreign workers the highest paying professions are in the HealthCare, Tech, Consulting and Management Industry. These are the average earnings in the profession. Naturally management is involving in this category because the position required experience and thus well compensated for that rare experience. HealthCare is there due to mismatch between demand and supply of workers in this industry. The one interesting discovery here is attorney, and this is interesting because that a profession that is highly regulated along national lines (each country has its own set of laws and law system) which is interesting that their skills can be transferred to the U.S and highly compensated.

* 1. For all applicants, who are the top recruiters of H1-B employees?



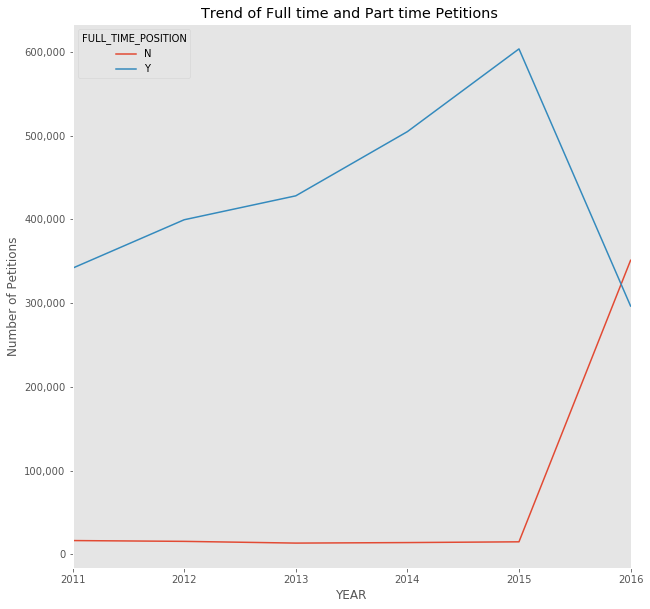
An interesting discovery here is that the top 3 recruiters of Foreign Employees are Foreign Companies, particularly Indian Tech Companies. Further research into this, by talking to a fellow student who used to work as a programmer analyst for Tata Consultancy Services Limited revealed that this is one of the top recruiter because U.S companies hire Tata to build custom software products. When these are being developed, and maintained for the U.S client so if the firm’s programming analysts must come to the U.S to work and because of the high demand for these firm’s services. They become top recruiters of foreign labor.

* 1. What job title occur the most in the applications?



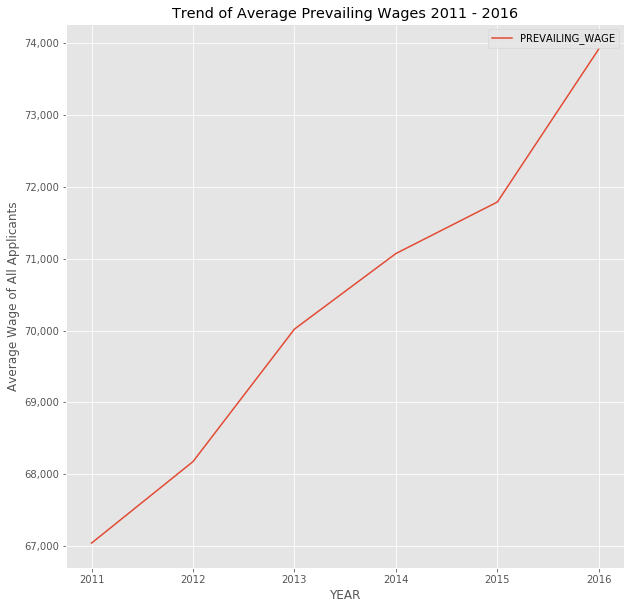
The most occurring job title in the application by a wide margin is programmer analyst. This supports the previous bar chart above as the top 3 recruiters usually apply on behalf of programmer analyst brought to the U.S to complete software development work for U.S clients. Programmer analyst and Software engineer at the same role with different titles. Therefore, the most occurring job title or the profession where foreign labor is recruited the most is Software engineering and programming. This information would be useful to policy makers as they try to understand the value of foreign labor to national economic output and address shortages in local talents that requires companies recruit foreign labor for these professions.

* 1. Analyzing the type of employment, what is the trend of Full time applicants vs Part time applicants from 2011 to 2016?



From 2011 to 2013 the number of full time applicants increased overtime while part-time was steady, however all this changes in 2016. Where we had more part time employee applications

* 1. Now, looking at the wages over the years. Are they increasing or decreasing?



Wages have been increasing over the years at a relatively steep rate

1. **DATA PREPROCESSING**

Data processing is a series of operation, performed on Data to get it ready for classification training and testing or data exploration.

* 1. **Data Cleaning**

The first step I undertook in the data preprocessing phase is to clean the data. This involved identifying the missing feature values. After I inspected all the missing values in each feature for some sort of pattern. I concluded that there was no pattern and the data was missing completely at random. Since most of the missing values were in columns that were categorical, I could not use any methods of imputation to fill in the missing values. Then assessing that all the instances that had missing values made up 0.04 percent of my data. I decide to drop all instances with missing values. There was no duplicate data to handle.

The next step I took was to inspect features with numerical values for outliers. Outliers, would distort my analysis and increase the chances of my classifiers underperforming. I examined year and prevailing wage. For year, I wanted to make sure all values were within the range of 2011 to 2016. For prevailing wage, I used a box plot to understand the distribution of the data in this feature. Upon closer inspect I realized that there were a lot of outliers. The highest prevailing wage close to 7 billion dollars. I decided to remove outliers, I removed all instances where prevailing annual wage was greater than two standard deviations from mean. I reasoned that this was a good threshold for outliers. This drastically reduced the number of outliers, however some still existed.



*Left: Before Removing outliers* *Right: After removing outliers*

Furthermore, I also converted the prevailing wage values and year values from floats to int64 data type.

* 1. **Data Aggregation**

During the data exploration phase I did some data aggregation via group-by. To get the top earnings by job title. I had to perform a group by on job title, and from that derived the count of application in each job title as well as the average wage for the job title. I also executed a group by on all the case status outcomes to derive the percentage of outcomes of all applications. I executed a group by on years to get the average wage for each year and plot the trend of wages over the years.

Apart from data aggregation, I also performed another preprocessing task under data transformation and this was encoding. I had several categorical features in my data that I used to train and tests my model. The classifiers I used, do not accept categorical feature values. I had to use patsy to encode the data such that it would maintain its meaning and convert to a format that the classifiers can learn and process. I used a module called patsy that converts categorical feature values to columns and uses 1 to reflect that an instance has that feature in its attributes or 0 to reflect that an instance does not have that feature. Initially this is where I realized I could not use all the data I had to train and test my model. My machine could not process encoded categorical features over 3 million instances.

I had to sample my data, after some experimenting with sample size and computational expenses I settled on a sample size of 20 000 instances. My next challenge was sampling the roughly 3 million instances in such a way that I maintain the original distribution of outcomes. I already had the problem of the DENIED case status being under represented in the data. (it was 3.09 percent of all outcomes). I used sampling without replacement to extract the 20, 000 instances I needed to train and test.

Upon sampling my data, I realized that my goal was to predict the like hood that an applicant’s application is denied or certified. I had a binary outcome analysis. However, my data, reflected four outcomes which were certified, certified-withdrawn, denied, or withdrawn. I converted the outcomes that were certified-withdrawn to certified, since this was the initial status. This reflected the true outcome of the application process not what happened after that process. I kept denied as my negative outcome, and dropped all instances with a withdrawn outcome. Since a decision was not made on these application, it was not relevant to include them in the data.

* 1. **Data Reduction**

I also performed some data reduction in the form of eliminating some features that I did not need, and using PCA (principle component analysis) to reduce feature space. I eliminated the SOC\_NAME, Lon and Lat features. After, I used patsy to encode the categorical features, the shape of my data frame was (20,000, 8000) roughly, I used the scikit-learn PCA implementation to reduce the number of components to 20 so that I had a resulting dataframe of (20,000, 20). This significantly reduced the time it took to train models and alleviated the computational workload on my local machine.

Another task undertaken under reduction is sampling. I performed sampling three times before train my Decision Tree Classifier, Random Forest Classifier and ExtraTrees Classifier.

Sampling without replacement

Stratified Sampling

SMOTE

Classifier

A

B

D

C

* A - > Sample without replacement to get 20, 000 instances
* B -> From those 20, 000 instances apply stratified sampling, to compensate for the imbalance of the minority class. Stratified sampling splits the data into test and training data where test data is 0.1 and training data is 0.9 (started with 0.3 0.7 split but needed more train data). This creates 18, 000 instances for training and 2,000 for testing.
* C -> However, I had problems at first whereby the models had a high number of false positive and true positive but very low number of false negatives or true negatives. When I checked the class probability from a naïve Bayes classifier I realized the probability of class certified was 0.9 awhile the probability of denied was 0.003.. I suspected the data imbalance problem that I tried to solve with stratified sampling was not yet solve. The reason is stratified sample, samples such that the original distributing is maintained. The problem was the distribution in the original data was skewed with a low representation of instances with denied outcomes relative to certified. To solve the data imbalance problem, I applied SMOTE. SMOTE is short for Synthetic Minority Over-Sampling Technique. This increased the representation of the minority class in the training data. I applied border1 SMOTE to the independent and dependent training data. To new training data with a 50 percent representation of each outcome in the training data.
* The training values from SMOTE sampling are the ones that used to train the classifier. The test data from the stratified split does not undergo the SMOTE sampling technique.

1. **MODELING**

For this project, I choose to perform SUPERVISED LEARNING. Classification, is the task of assigning objects to one of several predefined categories. I performed a classification task because the target variable to classify is categorical. I trained five classifier models.

The first model I trained was a K-nearest neighbors classifier. This is a rote classifier that memorized the training data. The idea behind this classifier is that the classifier will find the training data from the data it has memorized that is closest to the test instance and classify the test instance as that class from the training instance. This model took the longest time to train, mainly because it is so computationally expensive. I also suspect that it did not perform as well on my data because of outliers in the prevailing wage feature. K nearest neighbors, extends this idea in that instead of fining the closest one training instance to use to classify. It finds K of the nearest instances and classifies the test instance as a majority of those K nearest instances.

The second model I trained was a naïve Bayes classifier, Gaussian Naïve Bayes classifier, implemented in scikit-learn. The naïve Bayes classifier, idea lies in conditional probability backed by Bayes Theorem. Given a new instance that the classifier has not seen before, the classifier assesses the probability that the instance belongs to each of the available categories or outcome. The instance is classified as the class with the highest probability assessed. This classifier is a little more suited to my data than K-nearest neighbors since this classifier is more robust to outliers or noise which this data has a little of. It is also robust enough to handle irrelevant attributes. The one thing that can degrade its performance are when the same feature appears multiple times in a dataset. Which my data does not have.

The third classification model I trained was a Decision Tree classifier. The Decision Tree classifier used inductive learning from the training data to create a decision tree structure whereby each node is a condition on a feature and the edges are feature values. This tree is constructed such that the leaves have the outcome or decision. I used Entropy for splitting criterion which leverages information gain to make splitting decision as the tree is constructed.

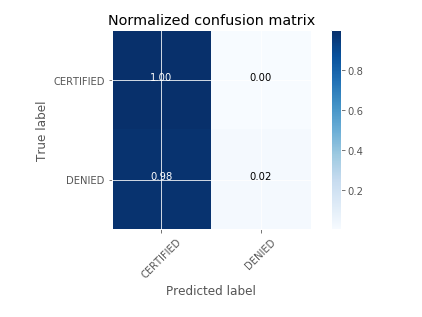
The fourth classification model I trained was a random Forest classification model. This is a type of ensemble method classifier. The goal of these kinds of classifiers is to improve performance and robustness of the of classification by averaging the classifications of several models. Instead of relying on one model. You have several models; this reduces the chance of over reliance on one model. Models are built separately and independently; we just average their results to get the final prediction. In random forest, each tree in the ensemble is constructed using a sample with replacement from the training set.

The fifth classifier I trained was like random trees, except there is another level of randomness introduced and this is called an Extra Trees Classifier.

1. **EVALUATION**
   1. **K Nearest Neighbors**
      1. Classification Report.

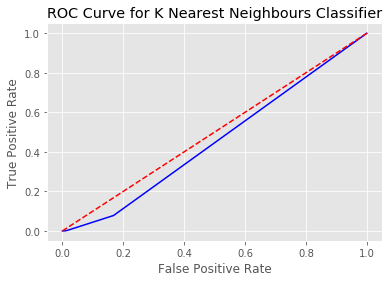
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Support |
| CERTIFIED | 0.97 | 1.00 | 0.98 | 3875 |
| DENIED | 0.17 | 0.02 | 0.03 | 125 |
| Avg / total | 0.94 | 0.97 | 0.95 | 4000 |

* + 1. Accuracy Score: 0.967
    2. Precision Score: 0.97
    3. Recall Score: 0.97
    4. Confusion Matrix



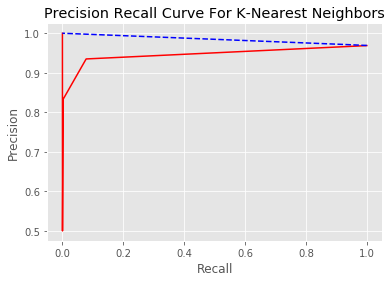
No false positive, all the positive instances are correctly predicted. A high number of False positives though. This model has a high tendency to label most instances positive. Probably because of the under sampling the in the training data

* + 1. ROC Curve



This classifier performs slightly worse than a random classifier represented by the red line

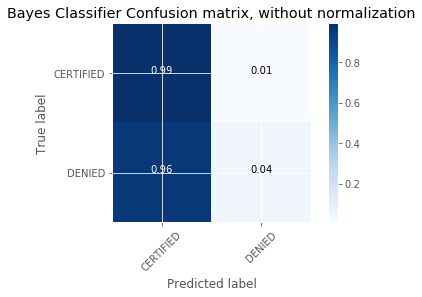
* + 1. Precision Recall Curve



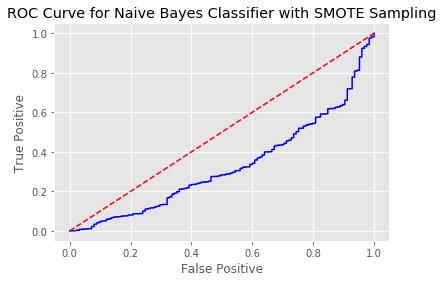
* 1. **Naïve Bayes (Gaussian Naïve Bayes Classifier)**
     1. Classification Report.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Support |
| CERTIFIED | 0.97 | 1.00 | 0.98 | 3875 |
| DENIED | 0.13 | 0.04 | 0.06 | 125 |
| Avg / total | 0.94 | 0.96 | 0.95 | 4000 |

* + 1. Accuracy Score 0.962
    2. Confusion Matrix

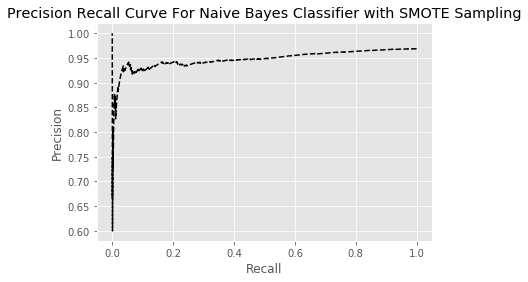


* + 1. ROC Curve

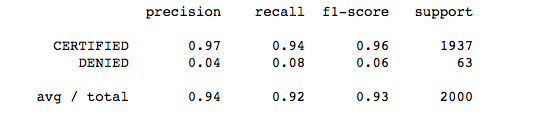


This means this classifier performs much worse than the K-Nearest Neighbors, classifier and even worse than a random classifier which is represented by the red diagonal line

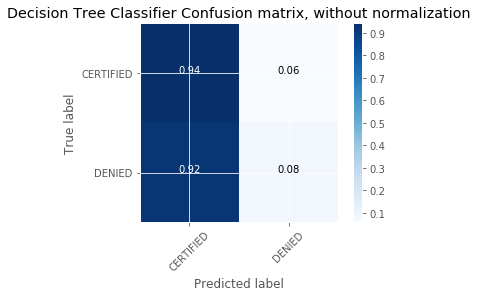
* + 1. Precision Recall Curve



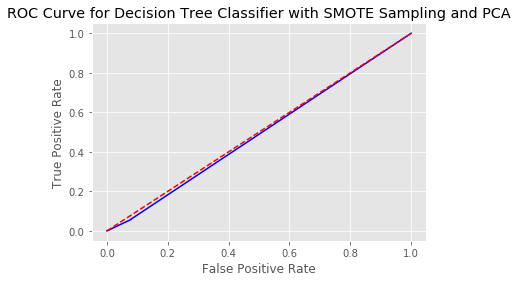
* 1. **Decision Tree Classifier**
     1. Classification Report.



* + 1. Precision Score 0.97
    2. Recall Score 0.94
    3. Confusion Matrix

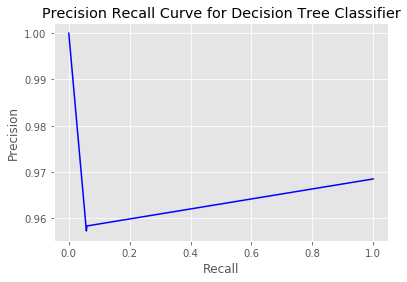


* + 1. ROC Curve

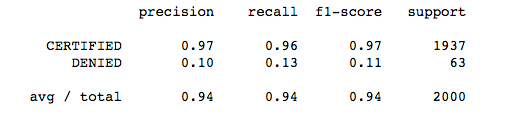


The model is performing slightly better, than the previous classifier. This is the best performing so far.

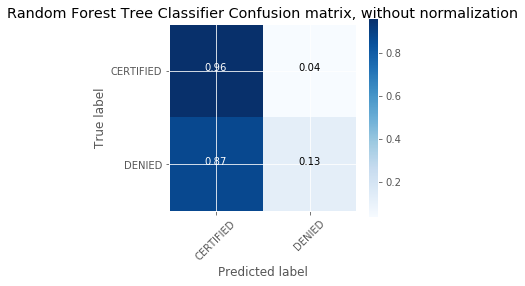
* + 1. Precision Recall Curve



* 1. **Random Forest Classifier**
     1. Classification Report.

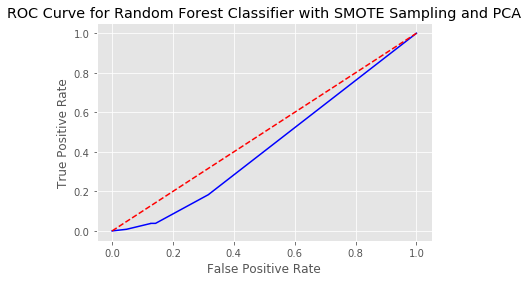


* + 1. Precision Score 0.97
    2. Recall Score 0.96
    3. Confusion Matrix



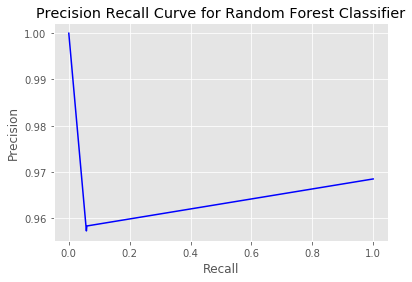
This classifier is so far the best at predicting TN and the first to significantly reduce the number of FP. This is better at predicting minority classes.

* + 1. ROC Curve

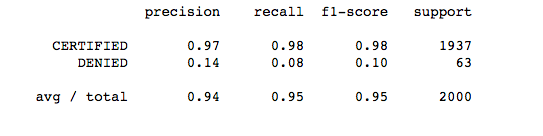


The ROC curve shows that this model does not perform as well as Decision tree classifier

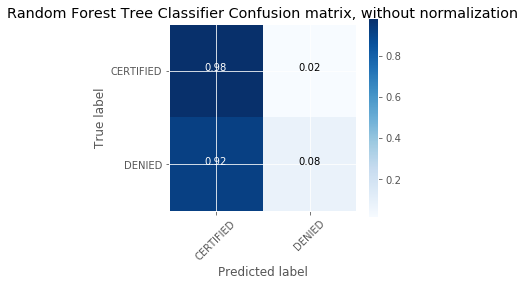
* + 1. Precision Recall Curve



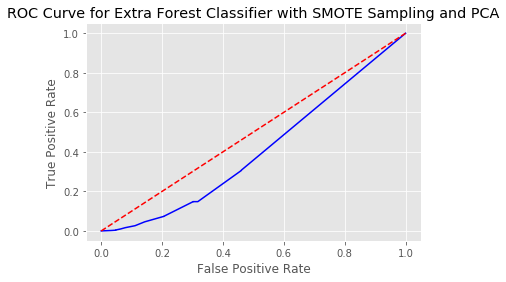
* 1. **ExtraTrees Classifier**
     1. Classification Report.



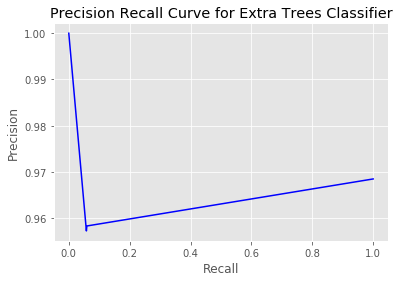
* + 1. Precision Score: 0.97
    2. Recall Score: 0.98
    3. Confusion Matrix



* + 1. ROC Curve



* + 1. Precision Recall Curve



For the most part the recall and precision values are in the same range or close to each other for all the classifiers. This is reflected in the precision recall plots that all look very similar. Furthermore, we can see that the performance of all the classifiers is great at predicting the positive outcome of certified but when it comes to predicting the negative outcome then these models fail to do a good job in that area. I think this is because of the low number of negative outcomes in the dataset, such that even SMOTE could not significantly make up for this difference. From the above classifiers, I think the random forest classifier it the best because not only does it have a similar performance in predicting positive outcomes, it also has the best performance in predicting positive outcomes.

1. **CONCLUSION**
   1. **Uses of model**

This model and analysis could be used by companies that recruit a lot of foreign employees to help them submit better applications. The company can tweak certain parameters and predict if an employee stands a better change in the visa petition. For example, a company can try several wage ranges until it meets a range that it can predict with significant confidence that if the employee applies with then they stand a better chance of being accepted or certified. Or analyze if an employ for that title should apply as Full time or part time. Furthermore, U.S universities can use this model to help their international student pick lucrative careers or career for which they have a high chance of working in the U.S after they complete their studies.

* 1. **Future scope of improvement**

If I had a couple more month to dedicate to this project I would, probably use cloud computing services with more processing power to process the whole data set instead of just sampling 20 000 instance from the original data. I believe that most of my models did not perform as well because of limited training data. If I could leverage the whole data set. I could get better performance. I also think if I had more time, I would like to try boosting methods on top of ensemble methods like random forest.