U.S. Permanent Certification of Labor Applications

Keara June Buck & Michelle Hsiung Russell

April 11, 2018

Abstract

For foreign workers to permanently work in the United States, they must apply for a Permanent Certification of Labor (PERM) from the United States of Labor. These foreign workers hope to obtain their PERM so that they can create a better life for themselves and their families. We use statistical and machine learning algorithms to distinguish the patterns in certified applications in the past six years and try to determine the likelihood of an applicant getting certified or denied. The best performing machine learning algorithm is Random Forest which can accurately predict whether an applicant is certified or denied almost 80% of the time. We also noticed that blank blank and blank are important features in determining an applicant's certification status.

1 Problem Statement and Motivation

Every year, thousands of immigrant visas are granted to foreign workers and their immediate family based on their job skills. Some forms of visa application require an individual to already have a job lined up in the United States. In such cases, the foreign worker must first obtain a Permanent Certification of Labor (PERM) from the U.S. Department of Labor before they can submit an immigration petition to the Department of Homeland Security's U.S. Citizenship and Immigration Services (USCIS) to apply for a visa. The PERM verifies that there are not enough qualified and willing U.S. workers to fill the position being offered at the offered wage. The certification also shows that the foreign worker will not negatively affect the wages and working conditions of similarly employed U.S. workers. We want to answer the following questions about the Permanent Certification of Labor (PERM):

I. What is the likelihood of an individual getting certified by the Department of Labor given their background?

II. Which variables have significant negative effects on receiving a PERM?

III. Which variables have significant positive effects on receiving a PERM?

These questions are very compelling because knowing what factors accredit or discredit getting a labor certification will help both employers and employees understand the foreign worker application process. We can also see if there are bias or flaws in their decision-making and how the immigration system can be improved in the United States.

Extensive research on Permanent Certification of Labor applicants has not been done. Those who have done some research on this have not use machine learning methods to answer our questions. However, we will experiment with various machine learning methods to classify these applicants. If we can get an accurate predictor, we can help those foreign workers know, with a certain amount of surety, if their application will be denied or not before they go through the time and money to submit an application.

2 Data

The dataset we will use is U.S. Permanent Visa Applicants, which can be found on kaggle.com. This dataset includes PERM application data from the years 2012-2016 and can be downloaded in a csv table here[1]. The Department of Labor discloses the PERM information every year. Since this dataset comes from the U.S. Department of Labor, we believe that it is reliable and that the information given is accurate. We also think that there should not be a bias other than people filling out their information wrong due to misunderstanding of the question.

The dataset has 374,362 data points with 154 features. We cleaned our data by removing features that are irrelevant, merging features that convey the same information, standardizing feature information, and removing duplicate data points. After this cleaning, we have 352849 applicants with 23 features. Some important features include employer state, salary, country of citizenship, class of admission, unit of pay, and education level. To see the details of the data cleaning please refer to this GitHub page[2].

3 Methods

In order to predict whether a person will be PERM certified, we will 1) address our unbalanced dataset problem and 2) use various machine learning methods to our answer questions.

3.1 Data Imbalance

Out of 352,849 applicants in the dataset, 330,325 of the applicants are PERM certified,and only 24,524 of the applicants were denied. Therefore, 93% of our dataset is PERM certified.

Having 93% of the applicants certified may cause inaccuracies in making predictions. This is because majority of machine learning algorithms are designed to improve accuracy and reduce errors. With this design, algorithms will be biased towards classifying any individuals as certified. There will be a biased towards a "certified" classification because even if the algorithm naively classifies all of its applicants as "certified", the algorithm will have about 93% accuracy.

In order to remove this bias, we will 1) undersample our dataset and 2) use ensemble to make our predictions.

3.1.1 Undersampling

The idea of undersampling is to create a new dataset where half of the data points are classified one way and the other half of the data points are classified the other way. This new dataset will then be used to train the machine algorithms to make predictions.

In order to undersample, we first need to create a train-test-split so that our test set also has an equally distributed classification. The following is how the train and test dataset was created:

Test Set:

1. Randomly sample 25% of the data points that is classified as "denied".
2. Randomly sample the same amount of data points as above that is classified as "certified".
3. Combine the two samples, use it as the test set.
4. Shuffle the new test set.

Train Set:

[1]https://www.kaggle.com/jboysen/us-perm-visas/data
[2]https://github.com/kearajune/PERM-classification/blob/master/DataCleaning.ipynb

1. Use all of the remaining "denied" data points in the training set.
2. Randomly sample from the remaining "certified" data points using the same amount of data points as the remaining "denied" data points.
3. Combine the samples.
4. Shuffle the data points and use it as the training set.

In this dataset, there will be 36,786 data points in the training set, and 12,262 data points in the test set (75 to 25 train-test split). To see the python code to undersample, refer to Appendix Section 7.1.

3.1.2 Ensemble

One of the disadvantages of undersampling is that we do not use the entirety of the data points to train the model. Hence, information is lost.

Using the undersampling method mentioned above, the training and testing sets consist of all of the "denied" data points but only about 10% of the "certified" data points.

In order to have a better representation of the dataset, we will train our algorithms multiple times with a different random sample of "certified" data points and all of the "denied" data points in our training set, keeping a 50-50 certified-denied ratio. In other words, we train multiple models using different training samples before determining our final prediction.

Each model will make a prediction with the sampled training set at each iteration. The final prediction of whether an applicant is certified or denied is decided by the majority of the predictions.

3.2 Machine Learning Methods

Each machine learning algorithms has distinct advantages and disadvantages. In order to best answer the question "What is the likelihood of an individual getting certified by the Department of Labor given their background?", we will try different machine learning algorithms to find the optimal model for our problem.

We can also optimize each machine learning algorithms by trying different parameters for each of the algorithms. For example, a machine learning method can add different weights on certain attributes, or an algorithm can place certain penalties in the regularization. For each of our machine learning methods, we will do a grid search to determine the best parameters for each model. That is, we run the model once with a variety of parameters and then choose the parameters with the best accuracy for our dataset.

After deciding on the best parameter for our dataset we will use the undersampling and ensemble method explained above to determine our predictions and accuracy of our model.

3.2.1 Scoring

The scores that will be used to measure how well our model our data is prediction is predictive accuracy and F_2 score.

Predictive Accuracy The standard scoring method is accuracy score. It is simply

$$\frac{\#\text{ of correct predictions}}{\#\text{ of all predictions}}$$

.

This will be the primary scoring we will use to measure how well each machine learning model is predicting the certification status of applicants.

F_2 Score The F_2 score is defined as

$$F_2 = 5 * \frac{\text{true positive} * \text{true negative}}{4 * \text{false negative} + \text{false positive}} \quad (1)$$

.

Where *true positive* is the number of correctly predicted denied applicants, *false negative* is number of incorrectly predicted certified applicants, and *false positive* is the incorrectly predicted denied applicants.

The F_2 is used because it puts a greater weight on recall , hence higher score is rewarded for higher true positives, which is the number of correctly classified denied applicants.

Because majority of the data is certified, it is important to correctly identify the applicants that are likely to be denied. Hence, F_2 score will help give higher values to models that have higher recall.

Like the predictive accuracy, the F_2 score ranges between 0 and 1.

3.2.2 Preliminary Exploration

To explore our dataset, we first took five machine learning methods (Naive Bayes, Random Forest, Gradient Boosted Trees, Random Forest, Logistic Regression) and naively fit them on our dataset. We simply used the default parameters for all the methods in order to get a clear idea of which algorithms work well with our dataset. We plotted both the predictive accuracy and the F_2 score of 15 test sets in Figure 1.

Notice in Figure 1, Naive Bayes clearly has the lowest scores in predictive accuracy; it also has lots of variance in its F_2 scores. On the other hand, Random Forest has the best predictive accuracy with `scikit-learn`'s default hyperparameters while Gradient Boosted Trees (XGboost) and K-Nearest Neighbor have comparable results. Logistic Regression does not classify as well as the prior three algorithms, but it can still provide meaningful results.

Hence, we will further explore Random Forest, XGBoost, K-Nearest Neighbor and Logistic Regression to determine the best model to answer our question.

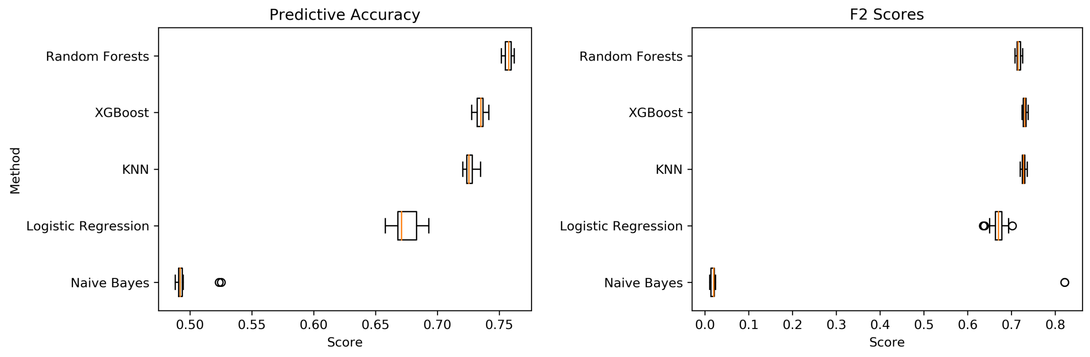

Figure 1: Predictive Accuracy (left) and F2 Scores (right) of various Machine Learning Algorithms.

3.2.3 Random Forest

A Random Forest is useful for our dataset not only because they are robust with regards to missing data and outliers, but also because they work well with a datasets that are mixtures of both categorical and numerical data. In a random forest, once there is a forest of multiple trees, our new data point runs through each tree and the random forest assigns the data point to the class with the most "votes" from each tree.

For the PERM dataset, we used `scikit-learn`'s `RandomTreeClassifier` to build a random forest on our dataset. The two parameters that we want to determine that will build the best classifier is the criterion used and the number of trees we want to build. The two types of criterion are *gini* and *entropy*. If criterion is set to *gini*, it means that each tree is split based on reducing the Gini impurity measure, while *entropy* requires each split to be maximizing information gain. Random Forests also have an attribute that allows you to look at feature importance, which we will use to determine the most necessary criteria in these applications.

Extra Trees: Scikit-learn has another method similar to Random Forests called Extra Trees that implements random decision trees, but at each split, the samples are drawn from the whole dataset, not a bootstrap sample and the splits are more random than Random Forest. This results in a lot more leaves in each tree. This implies than Random Forests are more compact, but Extra Trees are cheaper to train. There's give and take for both algorithms, but after we ran Extra Trees, we still got slightly better results with our Random Forest method. Extra Trees also has feature importance attributes that we will use.

3.2.4 XGBoost

XGBoost stands for extreme gradient boosting. XGBoost is very competitive in machine learning due to its speed and model performance. We choose to use XGBoost on our dataset because it performs well with categorical datasets and has high accuracy and speed. Gradient boosting uses a gradient descent algorithm to help determine which new tree, when added, will minimize our loss function. Then, each added tree will be apart of the model to determine the classification of an applicant based on the features they have.

The two parameters that we will experiment with is η and γ. The parameter η is the learning rate of the training process. Each gradient update, η is used to shrink each consecutive feature weights to prevent over-fitting. The γ represents the minimum value of loss required to make a particular split of a leaf node. The larger the γ, the more conservative the model will be. See XGBoost Parameters[3] for details.

We hope that using Gradient Boosting will give us a high predictive accuracy. Even though this is the case, XGBoost can easily over-fit, and we are not sure how well it will do at preventing false negatives. Just like the other Tree classifiers used, we will also look at the splits of the tree to get feature importance.

3.2.5 K-Nearest Neighbors

K-Nearest Neighbors is a classifier that determines which class a given data point belongs in by looking at the known data points nearest to it. The "K" in K-Nearest Neighbors is the parameter for how many data points to look at. We also look at the parameters *weights* and *algorithm*. *Weight* determines the weight function used in prediction. The two weight functions we used were *distance*, which weights the neighboring points be the inverse of their distance (ie, closer points have more pull than farther away points), and *uniform*, which weighs all neighbor points equally. The two algorithms used are *auto* and *brute*. *Auto* attempts to find the best algorithm based on the values from the fit method, while *brute* simply provides a brute-force search.

3.2.6 Logistic Regression

Although logistic regression models did not perform as well as the other machine learning models explored in the preliminary exploration (See Section 3.2.2), logistic regression is a classic statistical classification method that can not only make predictions but also make inferences on data.

[3]http://xgboost.readthedocs.io/en/latest/parameter.html

The logistic regression model is expressed in the following form:

$$\left(log\frac{p_i}{1-p_i}\right) = \beta^T x$$

Note that y_i is the case status of the ith applicant (and is Bernoulli distributed), x_i is all the features of the ith individual, and β is an array of coefficients we are trying to determine in order to optimize the log probability of categorizing applicants correctly.

Therefore, logistic regression can be used to see what factors positively and negatively influence the log probability of an individual getting certified or denied by looking at the coefficient of any particular feature.

We will use `scikit-learn`'s logistic regression method to find the optimal coefficents. We will use a grid search to determine the best solver, regularization, and predictive threshold to increase our predictive accuracy and F_2 score. Not only so, we will use the model to analyze the coefficients to determine the positive and negative effects of certain features.

4 Results

4.1 Predictions

The following are the parameters that produces the best models for each of the machine learning algorithms:

Random Forest: For random forests, the best model uses the Gini impurity to measure the quality of a split. However, the better predictive accuracy score was found using only 40 trees, but the best F_2 score was produced using 100 trees, which was the max amount we tested with.

Extra Trees: The best model based on F_2 scores for Extra Trees uses the Gini impurity to measure the quality of a split and has 100 trees, which again was are maximum in our gridsearch. However, the better predictive accuracy score was found using the entropy criterion with only 60 trees.

XGBoost: The model that had a maximum tree depth of 10, $\eta = .3$, and $\gamma = .7$ gave the best model that balances having high accuracy and high F_2 score.

K-Nearest Neighbor: The model that performed the best used 10 neighbor data points, with weight given by distance, with scikit-learn's 'auto' algorithm. This model had the highest predictive accuracy, as well as highest F_2 score in all our gridsearch.

Logistic Regression: We have determined that the best regularization to use was an L1-regularization with the regularization strength of .1. The liblinear solver was the only solver that was able to converge and find an optimal solution.

Table 1 depicts of the average accuracy and average F_2 of the models of trained.

Model	Avg. Accuracy	Avg. F_2
Random Forest	.758	.740
Extra Trees	.750	.737
XGBoost	.735	.731
K-Nearest Neighbor	.723	.713
Logistic Regression	0.716	0.698

Table 1: The average scores and accuracy of each model with the selected parameters.

4.2 Features

As mentioned before, we ran feature importance on the Random Forest and XGBoost. The basic feature importance counts the the number of splits used on a feature.

The top fifteen features that are split the most for Random Forest, Extra Trees,and XGboost are shown in Figure 2, Figure 3, and Figure 4, respectively.

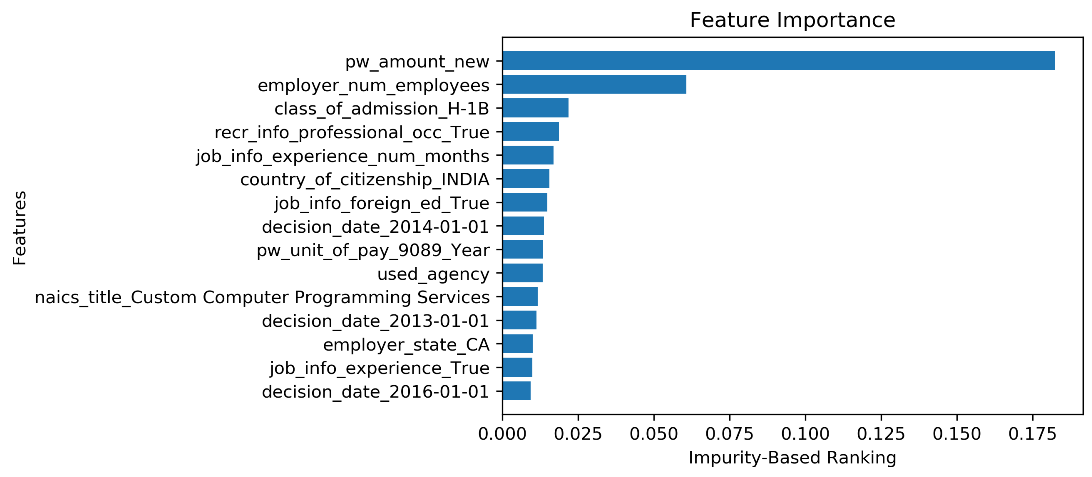

Figure 2: Random Forest Relative Feature Importance

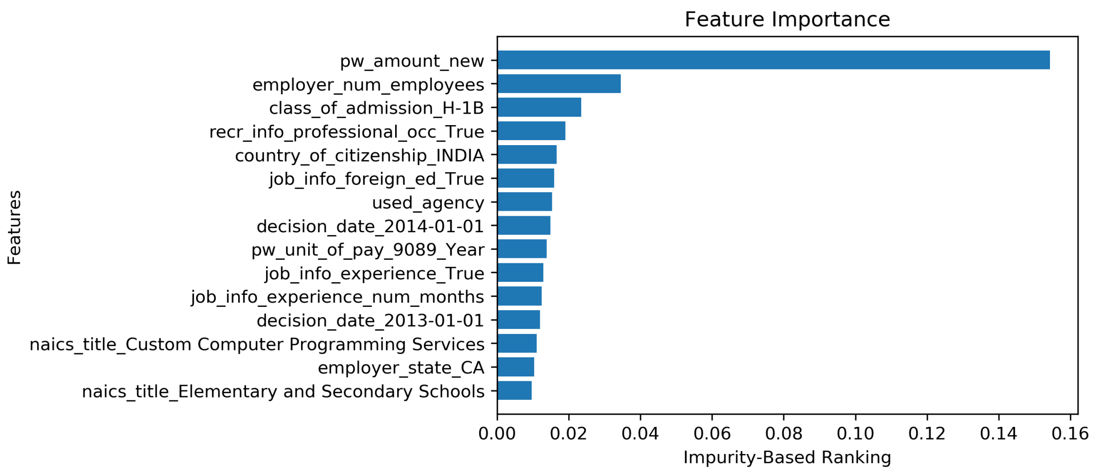

Figure 3: Extra Trees Relative Feature Importance

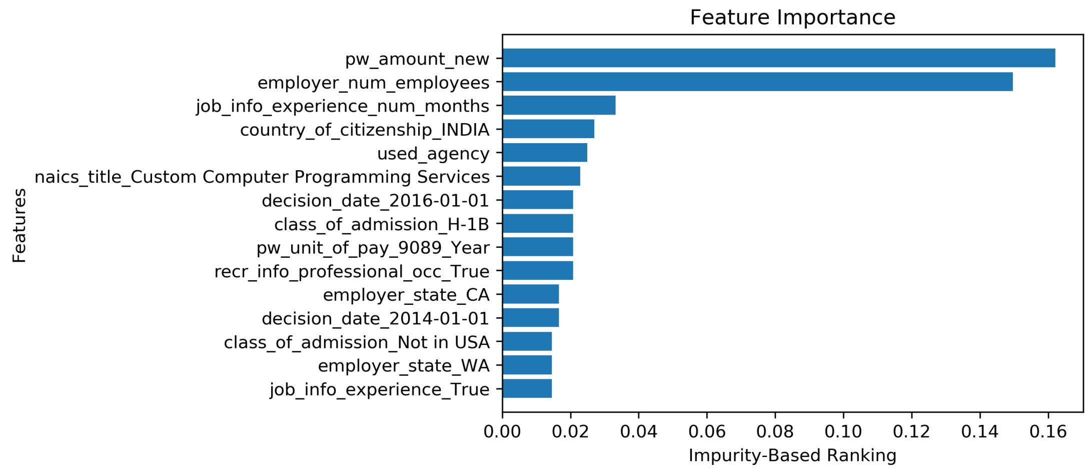

Figure 4: XGBoost Relative Feature Importance

Logistic Regression coefficient analysis is currently not working, so there are no analysis on the coefficients yet.

5 Analysis

5.1 Predictions

The main question was to determine the likelihood of an individual getting classified by the Department of Labor given his or her background.

After understanding the data, it has become apparent that 93% of the applicants are certified. So the question is not necessary what the likelihood of certification, but rather identifying the individuals that is likely to be denied.

Based on our results, Random Forest has the greatest accuracy and F_2 score. However, the accuracy is not as ideal as we hoped it to be.

With tests sets that are evenly split, only around 75% of the data are predicted correctly. In fact, the confusion matrix in Table 2 reveal that there are 7159 individuals that are denied and were classified as denied. Therefore 72.98% of all the individuals were actually denied their application were predicted correctly (recall). Not only so, 77.56% of the individuals that were predicted denied were predicted correctly (precision).

		Actual Case Status	
		Denied	Certified
Random Forest Predictions	Denied	7159	2071
	Certified	2651	7739

Table 2: Confusion Matrix on the Random Forest model.

As mentioned before, it is more important to have a high recall than a high precision in our case, because we want to make sure that those who were actually denied were predicted as denied.

However, trying to find the model that would increase the recall percentage has been difficult. Changing the threshold did not increase recall and F_2 score very much while dramatically reduced the predicted accuracy. Creating a completely new objective function that would account for increasing recall maybe a solution to this problem in future research.

5.2 Features

We also hope identify features that have significant positive and negative effects on getting PERM certified. At this time, the only machine learning algorithm we used that can effectively answer this question is logistic regression. Unfortunately, we were not able to find the significant features and its coefficients due to the singular matrix of our data of 2000+ features (after creating dummy variables for categorical variables).

However, using feature importance on our trees algorithms, we were able to identify that salary, size of employer, whether they used agency as some of the features that are crucial to identifying the status of an applicant.

6 Conclusion

In conclusion, our predictions produce about a 75% accuracy rate, which is not ideal in our situation. In order to help as many people as we can, we need to have a higher F_2 score so that we can correctly identify applicants who will be denied. However, the interesting part of our model is feature importance. We have found about five features that strongly predict whether or not an applicant will be certified or denied. Once we find the patterns for each feature, we may be able to better predict certification manually in the future.

7 Appendix

7.1 Train Test Split code

```
import numpy as np
import pandas as pd
  def train_test_split_50(data1):
      """This train test split, creates a test data that is 50% with target ↩
          of 1 and 50% with target of 0.Currently, 25% of the data is in the↩
           testing set."""

      certified = data1.case_status_Denied==0
      denied = data1.case_status_Denied==1
      d = len(data1[denied])
      # Test split index (at 40%).
      tsi = int(np.ceil(d*.25))

      # Data that is declined and shuffle.
      d_data = data1[denied].sample(frac=1)

      # Data that is certified and shuffle.
      c_data = data1[certified].sample(frac=1)

      pd = list(np.random.permutation(len(d_data)))
      pc = list(np.random.permutation(len(c_data)))

      # Sample 40% of the denied data.
      test_d = d_data.iloc[pd[:tsi]]
      test_c = c_data.iloc[pc[:tsi]]

      test_set = (test_d.append(test_c)).sample(frac=1) # Shuffle the ↩
          dataset.

      X_test,Y_test = test_set.drop(["case_status_Denied"],axis = 1),↩
          test_set["case_status_Denied"]

      # Train data

      train_d = d_data.iloc[pd[tsi:]]
      train_c = c_data.iloc[pc[tsi:]]

      train_set = (train_d.append(train_c)).sample(frac=1) # Shuffle.

      return train_set,X_test,Y_test
```

7.2 Feature Names and Descriptions