Prediction of US Permanent VISA application status

Prathwish S Shetty, Nikhar Gaurav

Abstract

Permanent visa application is the process of obtaining a labor certification which in turn enables foreign nationals seeking permanent residence in the USA to apply for green card through their employment. The US Department of Labor (DOL) is responsible for issuing the labor certification after making sure that there are no sufficient qualified workers in the U.S, willing and available to perform the specific job around intended employment and the employment of the foreign workers will not adversely impact the wages and working condition of similarly employed US workers. The whole process puts a substantial weight on job duties, training, experience, employer, wages offered, educational qualification and other capabilities. The objective of this project is to understand the various factors involved in the selection process and develop a framework that will enable us to predict the status of the visa application even before starting the process. We would like to leverage classification techniques like Logistic Regression, Naive Bayes and Neural Network to help predict the status.

1. Introduction

Getting a permanent residence status is arguably the most controversial and time taking process in recent years. With the number of people seeking status increasing exponentially day by day, the total waitlist is currently at an average of 12 years. And the process of applying for a labor certification from DOL (Department of Labor) is no less tedious where the decision processing can take time from 6 months to a year. In case of labor certification being denied from DOL, an appeal must be made to request for reconsideration where it can take more than a year for the final decision. All this process involves a continuous investment in term of time and money which can be exhaustive at times. So, there is a need for a solution where the person applying and concerned employee can predict their chances of success and improve. So, through a proper implementation of classifier model, we can save a lot of hassle for concerned people.

2. Technical Approach

2.1. Neural Network

Neural Networks have their origins in the study of the complex structure of the and behavior of the human brain. McCulloch and Pitts (1943) introduced simple models with binary neurons. Then, Rosenblatt (1958) proposed the multi-layer structure with a learning mechanism based on the work of Hebb (1949), the so-called perceptron, and the first neural networks applications began with Widrow (1959).[5.6]

.

Neural Networks are arranged into 3 or more layers of a neuron, a multi-layer structure:

- **Input Layer:** This is the very first layer of a neural network and it takes in the features of the data being modeled as the input to the network. The number of neurons in the first layer is determined by the number of features in the input data
- **Hidden Layers(s):** This layer takes the input and weights from the previous layer (either hidden or input) and then applies an activation function and gives out the output. The process of determining the number of neurons in the hidden layers and the number of hidden layers is by the process of trial and error to determine if the structure addresses the problem at hand
- **Output Layer:** The output layers take the input from the last layer of the network and then provides the estimation of the network

The training of a neural network is done by a two-step process, forward propagation, and backward propagation. In forward propagation, the network learns the activations at each of the neurons and in the back propagation, it learns the weights and biases.

The formula to learn activations is as follows and it happens during forward propagation:

$$Z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)},$$

$Z^{(l+1)}$ *is the input for next layer,* $W^{(l)}, a^{(l)}, b^{(l)}$ *are weights, activations and biases from previous layer*

$$a^{(l+1)} = f(z^{(l+1)})$$

$a^{(l+1)}$ *is the activavtion for the next layers and* $f(.)$ *is the activation function*

There are a variety of activation functions available to be used as $f(.)$ to activate the inputs to each neuron. The most popular ones being sigmoid, rectified linear units (ReLU) and Tangential Hyperbolic(tanH). For this implementation sigmoid (SoftMax implementation) is being used as the activation at the last layer and tangential hyperbolic at the initial layers

A cost function is used that will help measure the error and minimize the same to achieve the best fit model. Cross-entropy loss function is the best-fit cost function when using sigmoid in the last layer.

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}[y_n log a^n + (1-y)\log(1-a^n),\ \textit{where } a^n \textit{ is the activaion from the last layer}$$

And using this cost function the weights and biases from back propagation as:

$$\delta^{(n)} = (a^{(n)} - y)$$

This would be for the last layer with cross entropy cost function and sigmoid as activationo function

$$\delta^{(l)} = \left(\left(w^{(l)}\right)^T \delta^{(l+1)}\right) * f'\left(z^{(l)}\right), l = 1,2,\dots n \text{ , Every layer except the last one}$$

$$\frac{\partial J}{\partial w^{(l)}} = \delta^{(l+1)}(a^{(l)})^T, \qquad \frac{\partial J}{\partial b^{(l)}} = \delta^{(l+1)}$$

2.2. Naïve Bayes

Naïve Bayes classifier is based on principle where every pair of the feature being used for classification is considered independent of each other and assumed to

be contributing equally to the outcome. For classification, the dataset is divided into two parts, namely feature matrix and the response vector. Feature matrix contains all values of the given features from the dataset as rows and response vector contains the class variable for each row of feature matrix.

Let us consider X as a feature vector of size n where:

$$\mathrm{X} = (x_1, x_2, \ldots, x_n)$$

And y as the probability for class.

By Naïve Bayes theorem,

$$\mathrm{P}(\mathrm{y}|x_1, \ldots\ldots, x_n) = \frac{P(x_1|y)P(x_2|y)\ldots\ldots\ldots P(x_n|y)P(y)}{P(x_1)P(x_2)\ldots\ldots\ldots P(x_n)}$$

Which can be expressed as:

$$\mathrm{P}(\mathrm{y}|x_1, \ldots\ldots, x_n) => P(y)\prod_{i=1}^{n} P(x_i|y)$$

As the denominator remains constant it is removed.

For the classifier model, the probability of given set of inputs for all possible values of the class variable *y* is calculated and the output with maximum probability is selected. This can be expressed mathematically as:

$$\mathrm{y} = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

Where p(y) is the class probability and $P(x_i|y)$ is the conditional probability.

As various features in the dataset have continuous values, the conditional probability is calculated using Gaussian Probability Distribution Function:

$$\mathrm{P}(x_i|\mathrm{y}) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2})$$

Where μ is the mean of each feature corresponding to the given class

x_i is the feature

σ^2 is the variance

2.3. Logistic Regression

Classification through Logistic Regression is generally done for dataset where the response vector (class variable) is dichotomous (i.e. it takes values 0 or 1 only). Here, the relationship between the explanatory variables (X- variables) and class variable (Y- variable) is defined. To map predicted values to the probability the sigmoid function is used. The function maps any real values between 0 and 1.

$$\mathrm{S}(\mathrm{z}) = \frac{1}{1+e^{-z}}$$

Where S(z)= output between 0 and 1

Z= input to the function

e= base of natural log

The value obtained after sigmoid is between 0 and 1 which is mapped to a class by selecting a threshold of 0.5 and setting values with p>=0.5 as class 1 and p<0.5 as class 2. Next task is to find the optimal local minima which are done by calculating the gradient descent over cross entropy cost function i.e.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1 - y^{(i)}\log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Where y is the class value (0 or 1)

$h_\theta(x^{(i)})$ is the hypothesis calculated using the sigmoid function

λ is the regularization hyperparameter used to control overfitting.

3. Experimental Results

3.1. Dataset

We have obtained a dataset from the US Department of Labor website. For the scope of this project, we have taken the Permanent Visa Applications Datasets for the years 2015-2017 and the first quarter of 2018. We have filtered out individuals who have received a decision on their application. As the dependent variable, we have taken the case status for each application. The case status can either be 'certified', 'denied' or 'withdrawn'. We have only considered applications that have started the audit process hence ignoring the cases where the application has been withdrawn.
The dataset has 125 variables and over 300 thousand rows. The information that is available in the dataset can be broadly classified into 'Case Specifications', 'Employer Information', 'Job Specific Information', 'Hiring Process for the Job', 'Advertisement for the Job' and 'Foreign Worker Educational Information'. And from the initial analysis of the data, we observed that 34 of the columns have more than 50% of the columns missing. And most of the missing values are present in columns containing information related to the job and the recruiting process for the job.

3.2. Data Preprocessing

The dataset was obtained from the US Department of Labor website and had 125 variables and over 300 thousand rows. On the onset, the dataset appeared to have most of it values missing. But on further analysis of the visa application form, it showed that the columns cannot be used individually as a single entity rather have to combine the various features and make them into just one single feature. For example, the position the employee is being hired needs to be advertised in at least three different media for the application to be accepted, instead of having a binary flag the dataset had the dates the advertisements were run in each of the medium being tracked. Hence, to get the binary flag nearly twenty separate variables had to be combined. The next step of data engineering involved removing the features that seemed redundant and did not provide any information, and this was done after a thorough research on the application process. The labels for categorical variables

were encoded to numbers. In the end, there were just 46 variables for further processing.

3.3. PCA and One hot encoding

After the initial data processing was done, one hot encoding was performed on this dataset to get multi-class categorical variables into dummy variables with binary flags to prevent the categorical variables from being considered as continuous variables. This increased the number of features in the dataset to about 655.

As the number of features was very large Principal Component Analysis was leveraged to reduce the number of features. On doing PCA, it was inferred that 500 variables could explain 98% of the variability in the data. Hence these 500 variables were selected for the modeling effort.

3.4. Downsampling

The available data is imbalanced where the number of rows that belong to case status of 'Certified' is 294,678 and case status of 'Denied' is 23,130. Imbalanced data can lead to the result being biased towards the majority class and perform inaccurate classification with high accuracy leading to overfitting of data. To handle this problem in Naïve Bayes and logistic regression, downsampling has been performed over the dataset by decreasing the number of rows with majority class to 23,130 which is exactly same as the number of rows with the minority class. This solves the issue of imbalanced data by making the classification less biased and resolves the problem of overfitting that could have been caused by imbalanced data.

3.5. K- Fold Cross-Validation

To find the best model for classification, K-Fold Cross Validation has been applied for values of k on Naïve Bayes and Logistic Regression. The Cross Validation has been performed on training set where it separates the set into k parts and trains on the k-1 part and performs testing on the left-out part. This is performed until every part has been through the testing phase. After this, the best model is selected for performing classification. This helps in selecting the best model and achieving the highest accuracy for the final test set.

3.6. Implementation of Neural Network

The neural network algorithm was implemented on the dataset after splitting the dataset into 3 parts namely training, test and validation. To determine the best structure and the hyperparameters, the network was run on various iterations of the hyperparameters. The hyperparameters in this implementation were regularization (to prevent overfitting), learning rate (during gradient descent), batch size for gradient descent, number of hidden layers and number of neurons in the hidden layers.

Below is the table containing the matrices calculated for the validation dataset for each structure of the neural network. The model was run against a various number of neurons in the hidden layer and few of the better results obtained from the network is given below.

Hidden Layer Neurons	Batch Size	Regularization	Learning Rate	Accuracy	Precision	Recall	Specificity
4	500	0.01	0.001	54.65%	96.94%	52.80%	78.53%
4	100	0.1	0.01	51.74	97.2%	49.43%	81.63%
4	500	0.001	0.01	73.93%	95.58%	75.30%	56.19%
4	100	0.1	0.1	78.10%	94.4%	81.13%	39.07%
4	100	0.1	0.001	37.03%	95.78%	33.63%	80.90%
3	1000	0.01	0.1	75.9%	95.7%	77.5%	55.08%
3	500	0.0001	0.01	43.24%	97.40%	39.91%	86.11%
3	1000	0.01	0.01	48.91%	96.98%	46.39%	81.35%
3	1	0.1	0.0001	51.41%	91.82%	52.30%	39.89%
3	10	0.1	0.001	54.15%	92.67%	54.95%	43.91%

From the results of the various models, it can see that there is no one great model, as each model overcomes a different problem. Since the process of developing this classifier was to help people understand the cases of rejection there is need to try to aim for a higher specificity value rather instead of a model with a very high accuracy and precision. Also, it does not make sense to use a model that classifies most people as rejects and gets a very high specificity value. The ideal model correctly predicts most cases of visa rejects while at the same time making as few misclassifications in approvals.

Hence the model with 4 neurons in hidden layer, a batch size of 500, a regularization of 0.01 and learning rate of 0.001 was selected as it had a good accuracy score without sacrificing much of specificity. It has a test accuracy of 54.56% and a specificity value of 79.1%.

3.7. Implementation of Naïve Bayes

The process of classification through Naïve Bayes starts after downsampling where the dataset now contains equal distribution for both the classes. By using the K-fold cross-validation the best model is selected for classification on test set which is 20% of total dataset after downsampling i.e. 9252. The model here consists of mean and standard deviation for each class corresponding to each feature in the dataset. After model selection, the prediction is done for the test set by using the Gaussian Probability Distribution function.

Here it calculates the probability that a given data instance belongs to each class and selects the class corresponding to largest probability. Below results were achieved from implementation of Naïve Bayes as a classifier.

K	Train Accuracy	Test Accuracy
5	0.565385	0.562308
10	0.578898	0.572741
15	0.581037	0.576956
20	0.586486	0.566255

From the above result, it is evident that the training and test accuracy is linearly increasing with increasing number of folds but with an exception for k=20 where test accuracy shows a slight dip. So, from above we take that at 15-fold the Naïve Bayes implemented here will give a better result in term of accuracy for training and test set.

3.8. Implementation of Logistic Regression

Binary classification through logistic regression has been implemented on the downsampled data by finding the best model through K-Fold Cross-Validation where 9252 rows have been separated for testing. Hyperparameters have been tuned by constantly trying a various combination of learning rate and regularization through 5-fold and 10-fold Cross-Validation. Values considered for learning rate are 0.1, 0.01, 0.001 and 0.0001. Whereas for regularization they are 1000, 100, 1, 0.1, 0.01, 0.001 and 0.001. Below are the top 5 and bottom 5 results obtained from Logistic Regression:

K-Fold	alpha	lambda	Train Accuracy	Test Accuracy
10	0.1	0.1	74.953	73.173
5	0.1	0.1	74.304	73.151
5	0.1	1	73.981	73.162
10	0.1	0.01	73.953	73.411
10	0.1	0.001	73.952	73.411
5	0.1	1000	71.409	70.460
5	0.1	100	70.818	70.536
10	0.1	100	70.791	70.504
5	0.01	100	70.818	70.536
10	0.01	100	70.791	70.504

It is evident from above result that the best combination of hyperparameters is learning rate of 0.1 and regularization of 0.1 for which the training accuracy is 74.95% and test accuracy is 73.17% at 10-fold. The precision obtained is 78.8% and recall is 71.4%.

4. Participant Contributions

4.1. Prathwish S Shetty:

Data gathering, data preprocessing, PCA and one hot encoding, implementation of Neural Network, the experimental result for Neural Network, final project report.

4.2. Nikhar Gaurav:

Downsampling, K-Fold Cross-Validation, implementation of Naïve Bayes and Logistic Regression, the experimental result for Naïve Bayes and Logistic Regression, final project report.

5. Reference:

5.1. https://www.foreignlaborcert.doleta.gov/performancedata.cfm

5.2. https://www.geeksforgeeks.org/naive-bayes-classifiers/
5.3. http://www.holehouse.org/mlclass/06_Logistic_Regression.html
5.4. https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/
5.5. https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/
5.6. 'The measurement of technical efficiency: a neural network approach', Daniel Santín, Francisco J. Delgado & Aurelia Valiño