Analysis of H-1B Data Classification using Supervised Learning Techniques

Shail Shah, Sharad Boni, and **Vineet Trivedi**
College of Computer and Information Sciences
Northeastern University
Boston, MA

Abstract

The H-1B is a visa that allows U.S. employers to temporarily employ foreign workers in specialty occupations. Every year, the United States Citizenship and Immigration Services (USCIS) receives thousands of H-1B applications. The H-1B application process can be divided into stages, and one of them involves the employer filling a Labor Condition Application (LCA) for the employee. This application contains attestations about wages, working conditions, and more. Based on the application, United States Department of Labor (DOL) approves or denies it. This project aims to build and compare classification models to determine the fate of a given LCA.From a computational perspective, we have a dataset consisting of multiple .csv files (inputs). Each entry inside the dataset contains 51 attributes, and a classification (that can have one of four values). We would like to perform feature analysis and retain only those fields which are relevant. From there, we will divide the data into training and testing data. We will use k-fold Cross Validation to train and validate our models, and test data to finally compare the developed models. The output will be classifications (either Certified, Denied, Withdrawn, or Certified Withdrawn) computed by the different models, which will be validated against the actual labels for accuracy and other statistical measures. This problem is especially interesting because the number of H-1B applications filed over the years is steadily increasing. It would be beneficial to uncover useful insights which might help employers getting their LCAs approved.

Introduction

Technology has a trickle-down effect on the lives of the people, the state of the community and the economy and well-being of a nation. It is often a reflection of the state itself. The advent in technology lures investment, creates jobs, raises the standard of living and, primary to the sustenance of this advent itself, attracts the top talent in world. The top talent in the world ensures and acts as a catalyst to this advent in technology. Thus, the self-sustaining cycle begins. The United States has been particularly intelligent and ahead of its time in understanding and harnessing the power of this cycle. Case in point, Silicon Valley has employed at least 1.2 million people each year since 2001.

The United States of America introduced the H-1B visa program under the Immigration and Nationality Act (INA), section 101(a)(15)(H). The singular aim being to attract highly skilled foreign workers to supplement the tech boom. It allows US employers to temporarily hire foreign workers in specialty occupations. Section 214(i)(1) of the INA, 8 USC 1184(i)(1), defines a specialty occupation as an occupation that requires

- theoretical and practical application of a body of highly specialized knowledge, and
- attainment of a bachelors or higher degree in the specific specialty (or its equivalent) as a minimum for entry into the occupation in the United States.

The H-1B application process is segregated into several stages. A critical aspect of the application process involves filing the Labor Condition Application (LCA) for the non-immigrant employee. A Labor Condition Application must constitute attestations regarding wages, working conditions, strike, lockout, or work stoppage and notice. Employers are also responsible for recording and furnishing, if required, supporting documentation for the attestations.

The wages attestation addresses the following two primary points of concern:

- The non-immigrant employee shall be paid at least the same wage that is paid to other similarly employed workers at the company.
- The non-immigrant employee shall be paid at least the same wage that is paid for the given occupation in the given geographical area.

The working conditions attestation address the following two primary points of concern:

- The hiring of the non-immigrant employee will not adversely alter the working conditions of other similarly employed employees at the company.
- The non-immigrant employee shall be exposed to the same working conditions as native US employees.

The strike, lockout and work stoppage attestation address the following two primary points of concern:

- A strike, lockout or work stoppage has not occurred at the company with respect to the given occupation.

- In the event of a strike, lockout or work stoppage at the company with respect to the given occupation, post the submission of the application, it is the responsibility of the employer to notify the ETA of the strike, lockout or work stoppage. Additionally, the company will not use the application to file for work authorization until a time where the ETA has satisfactorily determined that the strike, lockout or work stoppage has terminated.

The notice attestation addresses the following two primary concerns:

- Notice and a copy of the application must be provided to the non-immigrant employee.
- Notice of the application must be provided to the employees in the company.

Over 600,000 Labor Condition Applications are filed annually. Processing these applications under time restrictions is a gargantuan task. The applications are finally evaluated as:

- CERTIFIED
- DENIED
- WITHDRAWN
- CERTIFIED WITHDRAWN

Over 100,000 applications are either DENIED, WITHDRAWN or CERTIFIED WITHDRAWN. This poses a dual loss, time spent in processing applications which will eventually be withdrawn or denied due to errors in application filing and a resource loss in terms of time and manpower on the part of the non-immigrant employee and employer in filing the application again in case it is denied due to technical errors. Hence this paper aims to predict the outcome of a Labor Condition Application. An employer can filter out applications which have a strong chance of being DENIED. The employer can then work on the application and file an Labor Condition Application which has a strong prospect of being CERTIFIED. The labor department can use this to prioritize applications which have a low probability of being WITHDRAWN or CERTIFIED WITHDRAWN hence making the system more efficient.

The rest of this paper is structured as follows, ALGORITHMS, which discusses the algorithms used, why they were used as opposed to the other options available, how they are traditionally used and how we have used them; MODEL CREATION AND FEATURE ANALYSIS, which discusses the model creation and feature importance; RESULTS which contains detailed graphical representation of the results; ANALYSIS, which puts forth an interpretation of the results; CONCLUSION AND FUTURE SCOPE, which discusses the conclusions that we have drawn from the results and analysis and the scope of this paper going ahead; and finally REFERENCES.

Algorithms

Classification is the process of predicting the class of given data points. More specifically, it is the process of finding a set of models that describe and distinguish data classes and concepts, with the goal being to use the model to predict the classes of objects whose class labels are unknown. Classification algorithms are examples of supervised learning. These algorithms are given a set of training examples with the correct class label, and a set of test data. The task is to predict the correct class for each test datum. Mathematically, a classifier/model is given three inputs:

- A training set, $D = (x_1, y_1), (x_2, y_2), .., (x_n, y_n)$: This set consists of n tuples. A tuple x_i, y_i consists of x_i, an ordered set of attribute values; and y_i, the correct class/label for the given training example.
- A query tuple x_t: The correct class for this set of attributes is not provided; the model needs to predict it.
- Additional parameters, P: These parameters can be user-defined, and algorithm-specific.

Given these three inputs, the output of the classifier is to calculate $y_t = f(D, x_t, P)$, which is the correct class for x_t.

Classification is a two-step process. Step one, training, consists of building the classifier/model. This is done by learning from the training examples. In step two, prediction, the model is used on the test data, and its accuracy (among other metrics) is measured.

Classification has many use-cases, including credit scoring, handwriting recognition, document categorization, and more. For classifying the LCA applications, we have leveraged three algorithms: k-Nearest Neighbors, Random Forest, and Multi-layer Perceptron.

K-Nearest Neighbors

KNN is a supervised learning algorithm that is often used as a benchmark for more complex classifiers. This algorithm finds k training samples closest in distance to the query tuple x_t, and predicts the label from these. k here is a constant defined by the user, but there exists a variation in which it varies based on local point density (radius-based neighbor learning). KNN is non-parametric, instance-based, and used in a supervised learning setting. Non-parametric means it does not make any explicit assumptions about the classifier/model. It hence avoids the dangers of mismodeling the underlying distribution of the data. For example, if our data were non-Gaussian but the learning model assumed a Guassian distribution, wed get poor classifications.

Instance-based learning means that kNN does not explicitly learn a model. Instead, it chooses to memorize the training instances. The training instances are used as knowledge for the prediction step. Only when a query is made (x_t), will kNN use the training instances to output y_t.

Although the training step is minimal, kNN comes with both a memory cost and a computational cost. The memory cost is due to the fact that the training step involves storing the entire training set D in memory. The computational cost is incurred during the prediction step, where classifying x_t involves going through each entry in the training set D. KNN searches the memorized training observations for the K instances that most closely resemble the new instance and assigns to it the their most common class. KNN classifies x_t by the following two steps:

- The classifier runs through the entire training set D, computing the distance between each training observation and x_t. A set consisting of the K nearest points in the training data, A, is found.
- The classifier then estimates the conditional probability for each class.

$$P(y = j|X = x_t) = 1/K \sum_a I(y_a = j)$$

$$I(x) = 1 if x = True, else False$$

The class with the largest probability is assigned as y_t.

The distance function can also play a role in the classifiers accuracy. Hu et al. concluded that Euclidean distance function performs reasonably well over the categorical and numerical datasets, but not for the mixed type of datasets. Chi-square distance also performed well in their experiments.

K controls the shape of the decision boundary. A smaller value provides a more flexible fit. It will have high variance but low bias. The boundaries will be more rough. A higher value of K will have a smoother boundary, since there are more voters in each prediction. It will have low variance with more bias.

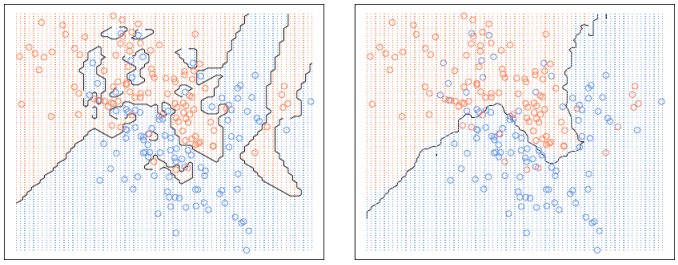


Figure 1: Applying KNN Algorithm with K = 1 vs. K = 20

While tuning the hyperparameter K, it is possible to overfit the model, which leads to the incapability of generalizing new observations. This happens when we use the test set as a training set. So, the test set must only be used at the very end. To solve this problem, we can use cross validation. In this method, we split the training set into k smaller sets. A model is trained using k-1 of the folds as training data. The resulting model is evaluated on the remaining part of the data.

Random Forest

Random Forest is another supervised learning algorithm, that makes use of multiple decision trees to classify the test data. Each forest builds multiple trees, that are able to classify a feature vector/test data. To classify x_t, it is given as input to each tree in the forest. Each tree gives a classification, and votes for that class. The forest chooses the classification with the most votes.

Formally, with the given the training set $D = (x_i, y_i)$, we train a family of classifiers $h_k(x)$. a random forest is a classifier based on the family of classifiers $h(x_{t1}), , h(x_t|_k)$, based on a classification tree with parameters $_k$ randomly chosen from a model random vector . For the final classification, $f(x_t)$, each tree casts a vote for the most popular class at input x_t, and the class with the most votes wins.

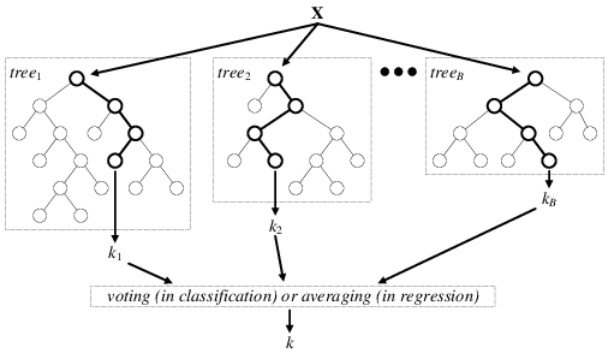


Figure 2: Random Forest algorithm

The number of trees in the forest is an important hyperparameter. Increasing the number of trees would lead to more computation time, however, as noted by Oshiro et al., it does not always translate to a better model. They found that 64 to 128 give a good balance between AUC, processing time, and memory usage.

The depth of the tree is a hyperparameter that can affect the quality of the random forest. It is usually recommended to prune the trees if there is noisy data. Fully developed trees can be used for shorter trees.

Another hyperparameter that can be adjusted is whether or not to apply bootstrapping. It is a metric that relies on random sampling with replacement. Bootstrapping allows assigning measures of accuracy (defined in terms of bias, variance, confidence intervals, prediction error or some other such measure) to sample estimates. This technique allows estimation of the sampling distribution of almost any statistic using random sampling methods. If bootstrapping is enabled, then bootstrap samples are used when building trees for the random forest.

While building individual trees, we can measure the quality of each split using either gini impurity or entropy.

$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

$$Entropy(E) = -\sum_{j=1}^{c} p_j log(p_j)$$

In Raileanu et Al.s work shows that both split almost identically, with entropy being a little slower because of the log calculations.

Multi-layer Perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of, at least, three

layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. The algorithm learns a function $f(\bullet) : R^m \rightarrow R^o$ by training on a dataset, where is the number of dimensions for input and is the number of dimensions for output. Given a set of features $X = x_1, x_2, ..., x_m$ and a target y, it can learn a non-linear function approximator for classification (or regression as well).

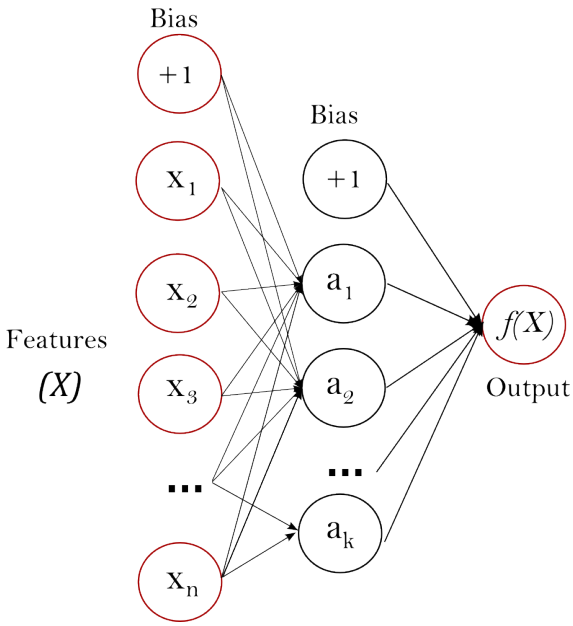


Figure 3: A visual representation of a multi-layer perceptron

The input layer consists of a set of neurons $\{x_1, x_2, ..., x_n\}$. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation $w_1x_1 + w_2x_2 + ... + w_mx_m$ followed by a non-linear activation function $g(\bullet) : R \rightarrow R$. The output layer receives the values from the last hidden layer and transforms them into output values.

Implementation

We retrieved the dataset from Kaggle. Our experiments consisted of trying to find the best set of parameters, that would yeild the best classification metrics for the LCA dataset. We used the Scikit Learn library to write our experiments and generate data. We ran them on a Windows 10 machine with 8GB RAM and a Intel i5 quad-core processor. We also used AWS t3.small instances for additional compute power.

Initially, tuning the hyperparameters was inadvertently treating the test data as training data, and so we observed overfitting. We tried to mitigate the problem by oversampling the examples with a minority class, undersampling the ones with a majority class, and k-fold cross validation. We found cross validation to be the most effective, so we started using them throughout the rest of the experiments.

After we obtained data from the experiments, we leveraged R studio for visualizing the data. We used ggplot2 to make the correlation pair plots and bar charts, comparing the three algorithms and their variants.

Results and Analysis

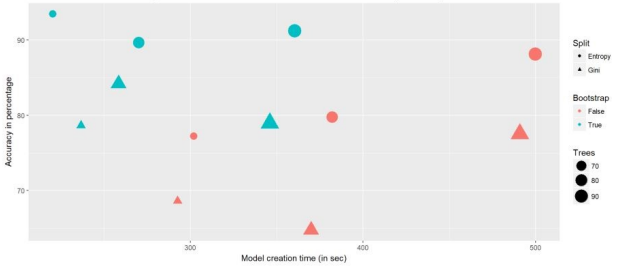
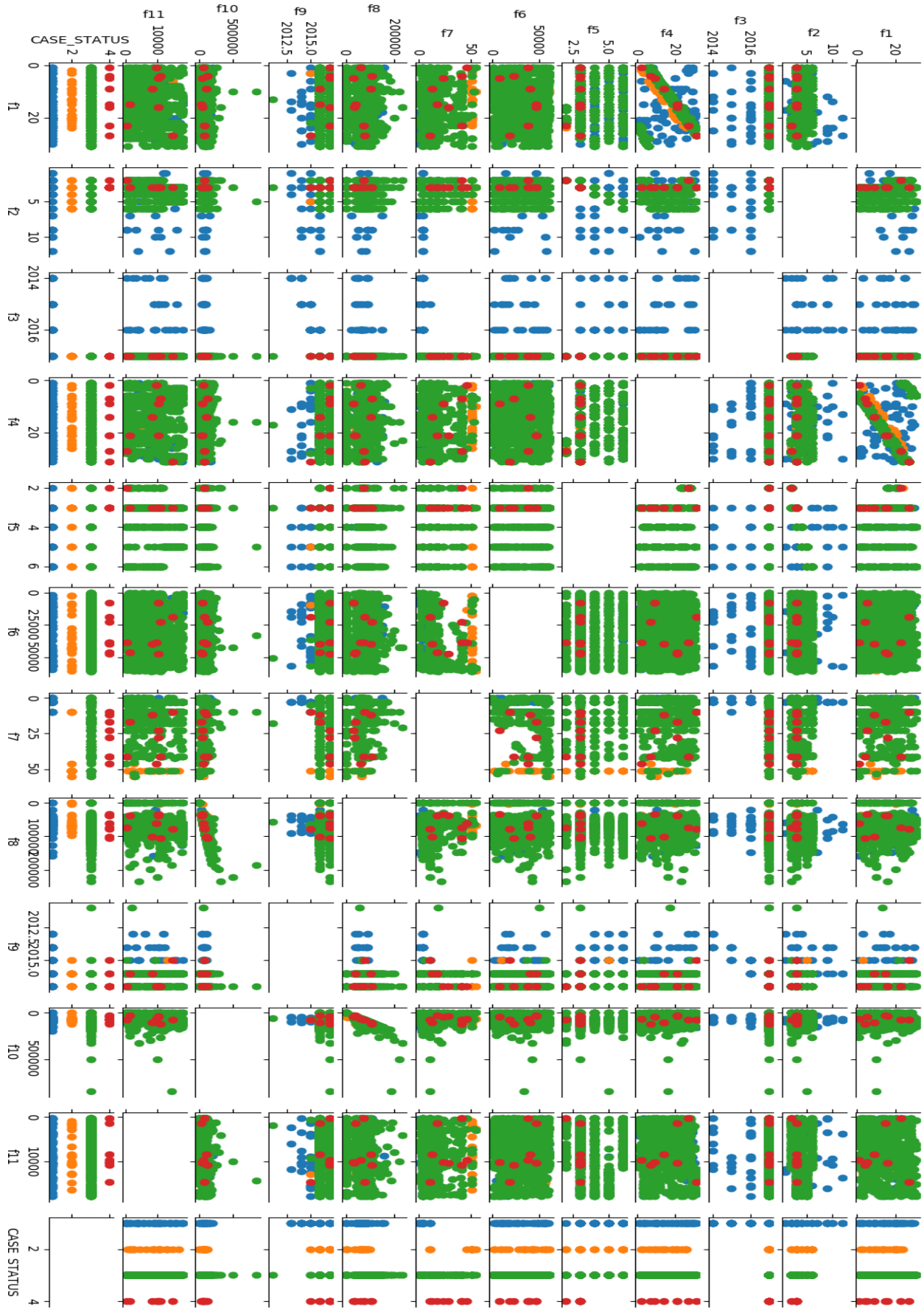


Figure 4: Random Forest Model Performance Evaluation (k-fold=3)

In figure 4 we can view the performance of various Random Forest Models created using k-fold cross validation. For the above model the value of k in k-fold is 3. On the x-axis we have the model creation time in seconds. On the y-axis we have the accuracy of the model expressed as a percentage. The shape corresponds to the Split. Entropy is represented by a triangle and Gini is represented by a circle. The size corresponds to the number of trees used in the random forest model. The larger the size of the plotted point, more are the number of trees. We have used 64, 80 and 96 trees respectively. The color represents the value of bootstrap. Blue corresponds to True and red corresponds to False. The model with the highest accuracy has bootstrap set to true, split set to entropy and number of trees are 64. It has an accuracy of 93 percent (rounded to the nearest integer) and the lowest model creation time of 220 seconds (rounded to the nearest integer). The model with the lowest accuracy has bootstrap set to false, split set to gini and number of trees are 80. It has an accuracy of 65 percent (rounded to the nearest integer) and model creation time of 370 seconds (rounded to the nearest integer). From figure 4 we can see that the blue (bootstrap=true) points occupy the top left region of the graph and the red (bootstrap=false) points occupy the bottom right region of the graph. This means that when bootstrap is set to false we get higher model creation times and the accuracy is relatively lower. The preferred option is to keep bootstrap true, in order to get lower model creation time and higher accuracy. We can also observe that keeping all other factors constant if we change the Split of the model from gini to entropy we can get a 5-10 percent improvement in accuracy. In general, keeping all other factors constant, the models that have split as entropy give a better accuracy than models which have split as gini. The shapes are uniformly distributed across model creation time and that suggests that the split does not heavily influence the model creation time. The sizes of the points are uniformly distributed across percentage accuracy and there does not seem to be a discernible pattern to identify which sizes fare



ss

Figure 5: Correlation Pair Plot

better. This means that changing the number of trees does not have a significant impact on the overall accuracy. The number of trees does influence the model creation time, as is clearly discernible from the graph the smaller points i.e. the points with lesser number of trees are focused on the left side of the graph indicating smaller model creation time and points that are larger are focused on the right side of the graph indicating greater model creation time.

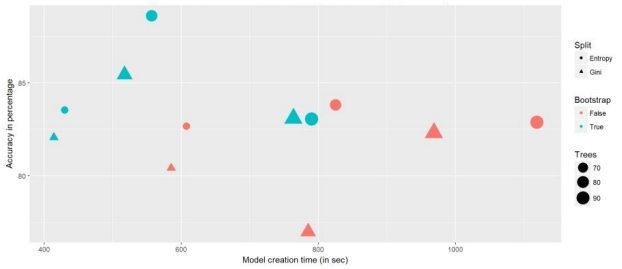


Figure 6: Random Forest Model Performance Evaluation (k-fold=5)

In figure 6 we can view the performance of various Random Forest Models created using k-fold cross validation. For the above model the value of k in k-fold is 5. On the x-axis we have the model creation time in seconds. On the y-axis we have the accuracy of the model expressed as a percentage. The shape corresponds to the Split. Entropy is represented by a triangle and Gini is represented by a circle. The size corresponds to the number of trees used in the random forest model. The larger the size of the plotted point, more are the number of trees. We have used 64, 80 and 96 trees respectively. The color represents the value of bootstrap. Blue corresponds to True and red corresponds to False. The model with the highest accuracy has bootstrap set to true, split set to entropy and number of trees are 80. It has an accuracy of 89 percent (rounded to the nearest integer) and model creation time of 557 seconds (rounded to the nearest integer). The model with the lowest accuracy has bootstrap set to false, split set to gini and number of trees are 80. It has an accuracy of 77 percent (rounded to the nearest integer) and model creation time of 785 seconds (rounded to the nearest integer). This graph is very similar to Figure 4. From figure 6 we can see that the blue (bootstrap=true) points occupy the top left region of the graph and the red (bootstrap=false) points occupy the bottom right region of the graph, however in this graph the distinction is not as stark as it was in Figure 4 and we do have a central mingling of red and blue points. This means that when bootstrap is set to false we get higher model creation times and the accuracy is relatively lower. The preferred option is to keep bootstrap true, in order to get lower model creation time and higher accuracy. We can also observe that keeping all other factors constant if we change the Split of the model from gini to entropy we can get a 5-10 percent improvement in accuracy. In general, keeping all other factors constant, the models that have split as entropy give a better accuracy than models which have split as gini. Again, for every circular plot point we can clearly see a trailing triangular plot point in the south west. The shapes are uniformly distributed across model creation time and that suggests that the split does not heavily influence the model creation time. The sizes of the points are uniformly distributed across percentage accuracy and there does not seem to be a discernible pattern to identify which sizes fare better. This means that changing the number of trees does not have a significant impact on the overall accuracy. The number of trees does influence the model creation time, as is clearly discernible from the graph the smaller points i.e. the points with lesser number of trees are focused on the left side of the graph indicating smaller model creation time and points that are larger are focused on the right side of the graph indicating greater model creation time. Additionally we notice that by changing the number of folds from 3 to 5 we have moved to a higher range of model creation times while simultaneously transitioning into a lower bracket of accuracy. Thus, keeping all other factors constant, if we change the number of folds from 3 to 5, then it will take more time to create a model which will have lesser accuracy.

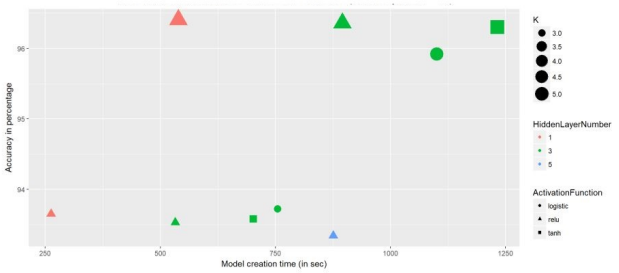


Figure 7: Neural Network Model Performance Evaluation (hidden layer size=13)

In figure 7 we can view the performance of various Neural Network Models created using k-fold cross validation. For the above model the value of hidden layer sizes is 13. On the x-axis we have the model creation time in seconds. On the y-axis we have the accuracy of the model expressed as a percentage. The shape corresponds to the Activation function. Relu is represented by a triangle, Logistic is represented by a circle and Tanh is represented by a square. The size corresponds to the k in k-fold. The larger the size of the plotted point, higher is the value of k. We have used 3 and 5 as the values of k respectively. The color represents the number of hidden layers. Red corresponds to 1 hidden layer, green corresponds to 2 hidden layers and Blue corresponds to 3 hidden layers. The model with the highest accuracy has relu as the activation function, 5 folds and 1 hidden layer. It has an accuracy of 96 percent (rounded to the nearest integer) and model creation time of 539 seconds (rounded to the nearest integer). The model with the lowest accuracy has relu as the activation function, 3 folds and 5 hidden layers. It has an accuracy of 93 percent (rounded to the nearest integer) and model creation time of 876 seconds (rounded to the nearest integer). From figure 7 we can see that the color of the plotted points is uniformly distributed across percentage accuracy. The number of hidden layers does not influence the

accuracy. The colors are segregated across model creation time though. Red (1 hidden layer) occupies the left region indicating, green (3 hidden layers) occupies the middle transitioning region and blue (5 hidden layers) occupies the right region. Meaning higher the number of hidden layers more is the time to create the model. The shapes are uniformly distributed across percentage accuracy and model creation time. Thus indicating that the activation function does not play an important role in altering the accuracy of the model or altering the time that it takes to create the model. The size of the plotted points is the most distinguishing aspect of the graph. The size is uniformly distributed across the model creation time although it can be noticed that models with more folds take slightly more to time to be created. Across percentage accuracy the size of the plotted points forms a visible divide. The bigger points are aligned horizontally at the very top of the graph and the smaller points are aligned horizontally at the vary bottom. Meaning 5 folds has a higher accuracy as compared to a 3 fold cross validation model. It is of pristine importance to point out the scale of the y axis. The y axis spans percentage accuracy from 93 to 97. The clear divide that we see does exist but it exists over a 1 or 2 percentage. Thus while acknowledging that increase in the number of folds of cross validation improved the accuracy of the model it is important to note that it improves the accuracy of the model by 1 or 2 percent.

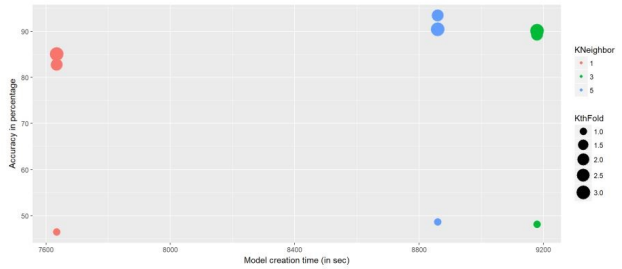


Figure 8: KNN Model Performance Evaluation (k-fold=3)

In figure 8 we can view the performance of various K Nearest Neighbors Models created using k-fold cross validation. For the above model the value of k in k-fold is 3. On the x-axis we have the model creation time in seconds. On the y-axis we have the accuracy of the model expressed as a percentage. The color corresponds to the number of neighbors used in k nearest neighbors model. Red corresponds to 1 neighbor, green corresponds to 3 neighbors and blue corresponds to 5 neighbors. The size represents the kth fold. The larger the size of the plotted point, higher is the value of k. The model with the highest accuracy has 5 neighbors. It has an accuracy of 93 percent (rounded to the nearest integer) and model creation time of 8860 seconds (rounded to the nearest integer). The model with the lowest accuracy has 1 neighbor. It has an accuracy of 46 percent (rounded to the nearest integer) and model creation time of 7633 seconds (rounded to the nearest integer). From figure 8 we can observe that the color has an impact on accuracy. Red (1 neighbor) has the least among the three, followed by green (3 neighbors) and blue (5 neighbors) having the maximum percentage accuracy. Thus, it can be said that as we increase the number of neighbors from 1 to 5, in this case, the accuracy increases. There should ideally be an increase in the model creation time with the increase in the number of neighbors. However, it is observed that 1N takes the least amount of model creation time and 3N takes the max as opposed to 5N. This anomaly can probably attributed to the fact that while other algorithms took several minutes for model creation, knn took a minimum of 2 hours. In this duration the machine would be running several other processes which could have easily distorted the model creation time data. Hence we shall not focus on the individual model creation times of knn models. Rather it is to be noted that each knn model took over 2 hours to be created whereas each random forest and neural network model was created in under ten minutes.

In figure 9 we can view the importance of all the features for classification. We extracted feature weights from our best random forest model (Accuracy=93 percent (rounded to the nearest integer), Model creation time=220 seconds (rounded to the nearest integer), k-fold=3, split=entropy, bootstrap=true, number of trees=64). On the x-axis we have the feature weight alias. The mapping between the alias can be found in the legend present on the top left of the figure. On the y-axis we have the weight of the features on a scale of 0-1. 0 being not important at all and 1 being highly important. The features are arranged in ascending order of importance. There are 26 features. The feature with the highest importance is soc name and it has a feature importance of 0.187 (rounded to the 3rd decimal place). The feature with the lowest importance is employer country and it has a feature importance of 0. We do notice some interesting trends. Having a dependent or being a willful violator does not have a strong impact on LCA. Features related to case submission date like case submission year, month and day have a strong correlation with the case status. Employer name has a strong correlation with the case status as well. The soc name, the feature with the highest feature, has the strongest correlation with the case status outcome. The Bureau of Labor Statistics' official website defines SOC as, "The 2018 Standard Occupational Classification (SOC) system is a federal statistical standard used by federal agencies to classify workers into occupational categories for the purpose of collecting, calculating, or disseminating data". Meaning the occupation and specific wage level within the occupation have the highest correlation with the case status.

In figure 10 we can compare the performance of random forest algorithm, neural network algorithm and k nearest neighbor algorithm using the following metrics: Accuracy, Precision, Recall and F1-Score. On the x axis we have the metrics and for each metric we have plotted the value of random forest, neural network and k nearest neighbor for the metric. On the y axis we have the metric score on a normalized scale of 0-1. 1 meaning the algorithm functions perfectly and 0 meaning the algorithm functions completely wrong. The color represents the algorithm. Blue is for Random forest, red is for neural network and green is for k nearest neighbor. For each algorithm we have taken the best model we created. Random Forest = Percentage Accuracy:

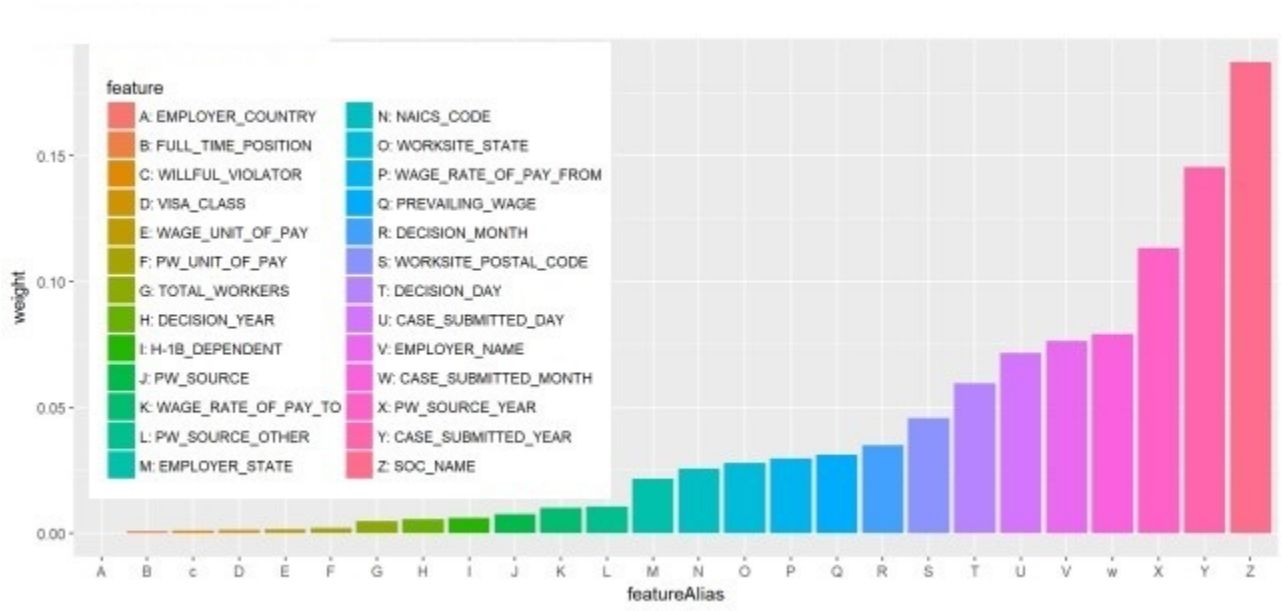


Figure 9: Feature Weights

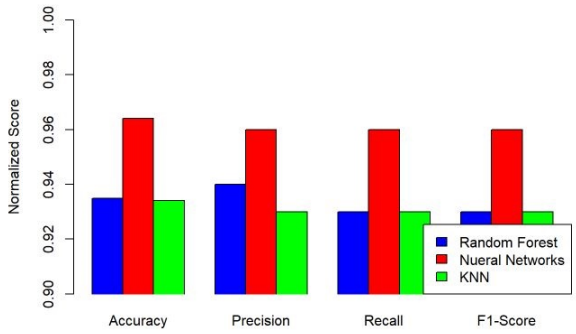


Figure 10: Performance Comparison of Machine Learning Algorithms

93 (rounded to the nearest integer), Model Creation Time: 220 seconds (rounded to the nearest integer), bootstrap: true, split: entropy, number of trees: 64, k-folds: 3 folds. Neural Network = Percentage Accuracy: 96 (rounded to the nearest integer), Model Creation Time: 539 seconds (rounded to the nearest integer), activation function: relu, k-fold: 5 folds, hidden layer number: 1, hidden layer size: 13. K Nearest Neighbors = Percentage Accuracy: 93 (rounded to the nearest integer), Model Creation Time: 8860 seconds (rounded to the nearest integer), number of neighbors: 5, k-fold: 3 folds. We can see that Neural Networks perform best across all the performance metrics. Its normalized score is roughly 0.02 higher across accuracy, precision, recall and f1-score. Random forest and k nearest neighbors have performed similarly except precision where random forest performs better by roughly 0.01. It is worth noting that the random forest model was created in 220 seconds (rounded to the nearest integer) as opposed to 8860 seconds (rounded to the nearest integer) taken to create the k nearest neighbors model.

Conclusion and Future Scope

In this paper we have analyzed the LCA dataset using Random Forest, Neural Netowrks and K Nearest Neighbor Models. We did a feature analysis to determine the relative importance of feature. We came to the conclusion that metrics related to case submission date are especially important. The most important feature is the occupation and the specific wage level within the occupation. Further we determined that employer name is also of importance in determining the outcome of the case status. Features like having a dependent, being a willful violator have a low correlation, the last one seeming interesting. For each of the 3 above stated algorithms we created several models, varying the number of folds in k-fold cross validation. Across the models the increase in the number of folds led to increased model creation times. We also varied other parameters specific to the algorithm. In random forest we varied the number of trees, the split and bootstrap values. Overall models with 3 folds fared better than models with 5 folds. Entropy models fared better than gini models. Models with bootstrap set to true fared better than models with bootstrap set to false. In neural networks we varied the number of hidden layers and the activation function. The accuracy of the models was visibly influenced by the number of folds in k-fold cross validation. The remaining factors did not influence accuracy to a noticeable extent. In k nearest neighbors we varied the number of neighbors. We concluded that for our dataset increasing the number of nearest neighbors yielded better accuracy. However, owing to the exceptionally high model creation

time of around two and a half hours (as opposed to a couple of minutes for random forest and neural network models) we limited the models to 1, 3 and 5 neighbors respectively. Overall neural network models (best model had accuracy of roughly 96 percent) fared the best, followed by random forest models (best model had accuracy of roughly 93 percent) followed closely by k nearest neighbor models (best model had accuracy of roughly 93 percent). Employers can use these models to predict the case status of their Labor Condition Applications with high accuracy. Similarly the labor department can use these models to deprioritize applications which are predicted to be eventually withdrawn. In the future we can build on this paper. The models can be improved to also give recommendations on how to improve the Labor Condition Application. To conclude, the h-1b program is an important program to bring the top talent from around the world into the country. This paper gives a much needed insight into what factors contribute towards the case status of an Labor Condition Application and provide tools for the accurate prediction of the case status of the Labor Condition Application.

References

H1-B Disclosure Dataset. Retrieved from https://www.kaggle.com/trivedicharmi/h1b-disclosure-dataset/home

OFLC Performance Data. Retrieved from https://www.foreignlaborcert.doleta.gov/performancedata.cfm

Wikipedia contributors. (2018, September 17). *Labor Condition Application.* Retrieved 21:11, December 12, 2018, from https://en.wikipedia.org/wiki/Labor_Condition_Application

Wikipedia contributors. (2018, December 10). *H-1B visa.* Retrieved 21:16, December 12, 2018, from https://en.wikipedia.org/wiki/H-1B_visa

Wikipedia contributors. (2018, December 8). *Neural network.* Retrieved 21:18, December 12, 2018, from https://en.wikipedia.org/wiki/Neural_network

Wikipedia contributors. (2018, December 4). *Random forest.* Retrieved 21:18, December 12, 2018, from https://en.wikipedia.org/wiki/Random_forest

Wikipedia contributors. (2018, November 3). *K-nearest neighbors algorithm.* Retrieved 21:19, December 12, 2018, from https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

Pedregosa et al. Scikit-learn: Machine Learning in Python. *JMLR 12*, pp. 2825-2830, 2011.

Buitinck et al. API design for machine learning software: experiences from the scikit-learn project, 2013.

Number of jobs with percent change over prior year. Retrieved from https://siliconvalleyindicators.org/data/economy/employment/job-growth/number-of-jobs-with-percent-change-over-prior-year/

INA Act 214(h). Retrieved from https://www.uscis.gov/ilink/docView/SLB/HTML/SLB/0-0-0-1/0-0-0-29/0-0-0-3422/0-0-0-3751.html

Standard Occupational Classification. Retrieved from https://www.bls.gov/soc/

Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach.* Upper Saddle River: Prentice hall.

Machine Learning Classifiers. Retrieved from https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

A Complete Guide to K-Nearest-Neighbors with Applications in Python and R. Retrieved from https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

The distance function effect on k-nearest neighbor classification for medical datasets. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4978658/

Mathematics of Random Forests. Retrieved from http://math.bu.edu/people/mkon/MA751/L19RandomForestMath.pdf

Bootstrapping(Statistics). Retrieved from https://en.wikipedia.org/wiki/Bootstrapping

How Many Trees in a Random Forest?. Retrieved from https://www.researchgate.net/publication/230766603_How_Many_Trees_in_a_Random_Forest

Neural network models (supervised) - Scikit Learn. Retrieved from https://scikit-learn.org/stable/modules/neural_networks_supervised.html

Multilayer Perceptron. Retrieved from https://en.wikipedia.org/wiki/Multilayer_perceptron