Northeastern University
College of Computer and Information Science

CS5100: Foundations of Artificial Intelligence

Project Proposal

Vineet Trivedi, Sharad Boni, Shail Shah

December 1, 2018

1 Project Description

The H1-B 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 H1-B applications. The H1-B application process can be divided into stages, and one of them involves the employer filing 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 clean the data by removing unwanted inconsistent data, and merge the data into a single datasource. From there, we will divide the data into 80% training and 20% test 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 The output will be classifications (either 'Certified', 'Denied', 'Withdrawn', or 'Certified Withdrawn') computed by the different models, which will be validated against the actual labels for accuracy and other statistical measures.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 H1-B applications filed over the years is steadily increasing. It would be beneficial to uncover useful insights which might help employers getting their LCAs approved.

2 Algorithms

To solve our problem, we need to experiment with classification algorithms. This problem has been solved previously using OneR, Naïve Bayes, and J48. We plan to use the following algorithms:

1. **k-Nearest Neighbors** (Shail Shah)

   This algorithm stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest k neighbors of the point.

   - **k**: We plan to experiment with what number of neighbours (k) gives the best result. We suspect that a larger k will reduce noise, but blur

boundaries.

- **Distance Metric**: We would also like to see which distance metric gives the best result. We will try Manhattan, Euclidean, and Chebyshev distances.
- **Distance weights**: We can apply uniform weights or weights inversely proportional to the distances.

2. **Random Forest** (Sharad Boni)

   This classifier fits multiple decision tree classifiers on subsets of the data. It averages the result of each classifier. This algorithm picks a sample of observations rather than all of them (bagging), and picks a sample of features rather than all of them (random subspace method), minimizing overfitting.

   - **Number of Trees**: We will set the number to 8, 16, 64, and 128.
   - **Quality of split metric**: We can use Gini impurity and information gain.
   - **Maximum number of features to consider**: We'll use the square root, log, or all of the features.
   - **Bootstrapping**: We are interested to see the result with and without bootstrapping.

3. **Multi-layer Perceptron** (Vineet Trivedi)

   A multi-layer perceptron is a supervised learning algorithm which is capable of learning linear as well as non-linear functions. It takes a set of inputs xi where i is in {1, 2, 3, .. n} and computes an output y. There can be one or more hidden layers between the input and the output. The neurons at each hidden layer and output layer perform a weighted summation of all its inputs, the result of which is then fed to a non-linear activation function like sigmoid or tanh. The output of the activation function is the output of the neuron.

   In this algorithm we would like to see the effects of changing the following parameters:

   - **Activation Function**: We plan to experiment with sigmoid ('logistic' as stated by scikit) and tanh functions.
   - **Hidden Layer Sizes**: We would like to see the effect of changing the hidden layer sizes on classification. We can initially set the size to 100, 50, 200. After inspecting these results, we can further tweak the hidden layer sizes.

- **Learning Rate**: Additionally, we would like to see what happens when we use a constant learning rate vs an adaptive learning rate.

3 Results

For each of the algorithm stated above, we would like to run variations of it and compare them amongst themselves. Then, we would pick the one that runs the best, and compare that with the best variations of other algorithms. We will try to reason why one algorithm works better than the other, based on the metrics and graphs.

For evaluating the quality of a model's classification, we will use the following metrics:

- **Accuracy**: Percentage of labels correctly classified.
- **Precision**: The ratio of true positives and all positives.
- **Recall**: The ratio of true positives and the sum of true positives & false negatives.
- **F1 score**: A weighted average of the precision and recall.
- **Zero-one classification loss**: The fraction of misclassifications.

4 References

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

https://www.kaggle.com/trivedicharmi/h1b-disclosure-dataset/home

https://en.wikipedia.org/wiki/Labor_Condition_Application

https://scikit-learn.org/stable/modules/multiclass.html

https://stats.stackexchange.com/questions/36165/
does-the-optimal-number-of-trees-in-a-random-forest-depend-on-the-number-of-pred