DMA Final Project: H1B Case Prediction

Yin Qiu, Chuhan Wang, Yuexi Wang

Introduction

H-1B visas are a category of employment-based, non-immigrant visas for temporary foreign workers in the United States. It is also the most common visa status applied for and held by international students once they complete college or higher education and begin working in a full-time position.

For a foreign national to apply for an H1-B visa, a US employer must offer them a job and submit a petition for a H-1B visa to the US immigration department. Laws limit the number of H-1B visas issued each year: petitions filed to the LCA enter a “lottery” system, in which around 30-40% of the cases are selected randomly for visa application (in fiscal year 2015, 172,500 cases were received and 65,000 cases were selected for regular cap).

As international students, we are interested in this space because:

1. Being a documented worker is a process we must go through once we graduate and want to stay in the U.S. to work.
2. The H1B petition is known to be difficult to obtain, and we want to ease our mind a little by fitting a model based on historical data to predict the outcome and discover factors that may contribute to a case getting certified.

Dataset

The Office of Foreign Labor Certification (OFLC) generates data about H1-B visas. The disclosure data is updated annually and is available online. This is the raw data for the Kaggle dataset.

We used the H1B dataset on kaggle.com, This dataset contains five year's worth of H-1B petition data, with approximately 3 million records overall. The columns in the

dataset include case status, employer name, worksite coordinates, job title, prevailing wage, occupation code, and year filed.

We used Tableau for some basic EDA and data visualization to see the top work locations, employers and job names. The top work locations are New York, Texas and California, and the top employers are within the technology and consulting sector. The top job names are computer programmers, software engineers and developers. We also plotted the distribution of prevailing wage and checked the distribution of the wage of those who got rejected and certified. It seems like most annual wages are around 50k to 10k and certified petitions have a median that's slightly higher than the denied ones. We also discovered in our EDA that full time positions are around 3.7 times higher than non full time positions.

Research Question

When we first obtained the dataset, we thought it is providing us information about whether or not an H-1B visa petition is approved. However, in the CASE_STATUS column, "CERTIFIED" does not mean the applicant got his/her H-1B visa approved, it just means that he/she is eligible to file an H-1B.

Since the dataset only provides us records of petitions that get the eligibility to file for an H1B petition, we set out the below research questions:

1. What factors influence the H1B petition cases being "Certified"?
2. How can we best predict, using known data, the chance of a H1B case getting certified by the LCA?

Introduction to Approaches

Below are a list of methods we've used in the data-processing, feature engineering and model fitting process:

1. Oversampling and Undersampling for Imbalanced Dataset

We noticed that in this dataset, the majority class outcome is many times more than the minority class, this might cause imbalanced data problems. To address this, we apply undersampling/oversampling methods. Since the training data has over 1.2 million rows, we think it would be more efficient to first use undersampled data to train our models, and then if necessary, use oversampled data to train the models again. The ultimate goal is to find the model with best performance.

2. Standardization scaling
3. Cross-validation
4. Grid search for hyperparameters tuning
5. Hand-picked testing set

   We've observed that all the models we applied have achieved higher accuracy on test data than our train data. We do not have an overfitting problem for all the models, we proposed the hypothesis that our model might be blindly predicting the majority class. To address this problem, we manually picked rows from our original dataset, and set the ratio of 1s : 0s to be 7:3 (consistent with our oversampling strategy) to be our hand-picked test set.

   We held out this part as our testing dataset, and went through the same process to train the models.

Below is a list of algorithms we've used for the binary classification problem.

1. Logistic Regression
2. Random Forest Classifier
3. XGBoost Classifier
4. Multi-Layer Perceptron

Tasks	**Yuexi**	**Yin**	**Chuhan**
Data cleaning, pre-processing, EDA	50%	30%	20%

Feature Engineering, Imbalanced Data, Standardization	33%	33%	33%
Model Selection and Hyperparameters Tuning	0%	100%	0%
Hand-picked test set, Retraining model & Troubleshooting	0%	0%	100%
Presentation & Report	80%	10%	10%

Individual Work

(between one and two pages per section per group member): Describe the work that you as an individual contributed to the project. Make sure to discuss your contributions in detail, noting (where applicable) how you applied concepts and techniques learned in class. Please include your name on each of your individual section pages.

Yuexi

My responsibilities include data preprocessing, EDA, feature engineering, and the final presentation and report write up. I've also participated in discussions on how to tackle the imbalanced data problem and test set accuracy score problem.

When we first encountered the dataset, it contained over 3 million entries, with 6 categorical variables and only 3 quantitative variables. Because the dataset is too large, I've decided to truncate the dataset to only keep entries from the most recent years (2014-2016). We've also decided to extract state name from "work site", and also combined our target variable "case status" so that it is binary (we kept the denied cases, combined certified-withdrawn cases with certified cases and dropped the rest of the case status).

I then plug the dataset into Tableau for some data visualization and EDA. Tableau allowed us to quickly discover top work locations, employers and job names. I also plotted the prevailing wage distribution across different case status and discovered that denied cases have a lower median than certified cases, and that the wage distribution of denied cases are skewed to the left (people with lower wages take up a higher percentage in the denied bucket). When we plot prevailing wages across different years and compare them across case status, we can also see that for each case status, although the median prevailing wage has been increasing, median wage on denied cases is consistently lower than certified cases. This discovery is especially useful later when we try to find features that attributes the most to prediction of case status, since our model results were not very interpretable, we had to speculate based on the discovery we've made in EDA that prevailing wage is an important factor.

We've also discovered the imbalanced data problem during EDA, we can see that after combining our target variables, over 90% is in majority class (certified). Drawing from knowledge in this class and a previous class (applied machine learning), we know that most algorithms will fail out of box on imbalanced data by blindly predicting the majority class. We decided to oversample/undersample the dataset and I've tried different approaches such as SMOTE (creates artificial data based on the feature space similarities between existing minority examples) but because we have too many categorical variables, we finally decided to use the random under sampler and random over sampler from imblearn.

There are many other methods to deal with imbalanced data, such as using other metrics instead of accuracy score to measure performance or doing stratified randomization to separately randomize majority class and minority class observations in test/train and cross

validation. Or doing some algorithm level adjustment so that it is more cost sensitive and vary penalty for different class examples.

Yin

My responsibilities include EDA, feature engineering, model selection and hyperparameters tuning.
In EDA, I helped revise some plots and cleaned up some of the features for better visualization. In feature engineering, I kept the state names in WORKSITE and encoded SOC_NAME, YEAR, and FULL_TIME_POSITION with the WOE encoder. I also inspected the outliers in wages, and dropped the longitude latitude. After vectorizing the features, I did the undersampling and oversampling for the dataset, and train test split. Then I went on to select the four models and tune the hyperparameters with grid search and 5-fold cross validation. Below are more details:
First, I used the undersampled data to train 4 different models as prototypes. I used Grid Search to tune the hyperparameters for the best performance in each model. I found XGBoost Classifier achieved the highest accuracy 74.3%, while the baseline Logistic Regression model has accuracy 61.4%.
However, after undersampling, our train data has only 41,316 rows, compared to 319,026 rows in test data. Randomly ruling out the Class 1 data in training may substantially compromise learning the cases of Class 1. Thus, our four models do not perform very well on test data. The benefits of implementing undersampled data is it saves us time in prototyping and tuning hyperparameters. After this step, we think it will be worthwhile to feed in more training data and see if the performance of models can improve.
Our next step is to run the four models on oversampled data, where train data has 1,632,078 rows and test data has 319,026 rows. As expected, the accuracy on oversampled data all increased to at least 93.8%. The performance of Random Forest (96.5%) and XGBoost (95.9%) are ranked first and second, while Logistic Regression (93.8%) and MLP (93.8%) are third. One thing to note is Logistic Regression takes least time while MLP takes most time to run.

One interesting takeaway is the tradeoffs in model selection, where Logistic Regression can achieve reasonably good performance with more data inputs in the shortest time and MLP also performs better with more inputs but takes the longest time and most memory. Random Forest and XGBoost are in the middle of the spectrum in terms of time and memory, but these two

tree-based algorithms both achieved very good performance. Also, Logistic Regression, Random Forest and XGBoost are all interpretable. We can plot the feature importance to understand which features have more importance in predicting the label.

Chuhan

I am mainly responsible for troubleshooting(namely hand picking testset), and I also coded the WOE(weight of evidence) of the feature engineering part.

We think the extremely high test accuracy looks suspicious, so we first tried cross validation to divide the data into ten folds. It turns out that the cross-validated accuracy is not that much different from the accuracy we gained without cross validation -- they are all around 63%. And then, we proposed the hypothesis that our model might be blindly predicting the majority class. Because the accuracy on our testset was roughly 94%, and that's exactly what you'll get by blindly predicting 1s on the test test.

So, we checked the false positive and true positive rate. The true positive rate is: 95%, which means our model does a great job at predicting the majority class. The true negative rate is 22%, which means our model fails miserably at predicting the minority class.

By comparing the results generated by our hand-picked data and our original approach, we can conclude that using our hand-picked test set balanced the precision, F-score and recall rates between the majority class and the minority class. In other words, we were able to increase the true negative rate by manually generating the test set at a given ratio. To be more specific, we set the ratio of 1s:0s to be 7:3, exactly the same as the ratio in our oversampling strategy.

More thoughts on this: inspired by GSI Priyam, we realized that this could also be an anomaly detection problem depending on what our research question is. For example, if we are particularly interested in predicting the H1B cases that will get denied, we can use the approach of anomaly detection. Moreover, the tactic of anomaly detection is also widely used in the field of banking, for instance, detecting fraud. In our case, we didn’t implement this strategy in this final project because we feel like we care about both the minority and the majority classes. However, one scenario could be that since our outcome variable is binary, we can use anomaly detection to detect anomalies and meanwhile label the non-anomalies as “the other class”. It will be interesting to see how this technique works in the future.

Conclusions

In conclusion, we've had a successful run of this project using H1-b petition data. We were able to discover issues with the dataset quickly by performing EDA. We've found solutions to the imbalanced data problem by doing both oversampling and undersampling. By comparing the results and run time efficiency, we did see a trade-off between error rate and computational efficiency when choosing re-sampling strategies. We also resolved the multiple categorical variable problem by applying WOE encoder instead of the traditional one-hot encoding, due to the nature of our categorical variables (many unique values which would cause a huge dimension of our dataset after one-hot encoding).

When doing model selection and hyperparameter tuning, we compared across 4 different models (logistic regression, random forest classifier, XGboost/gradient boosting classifier and neural network MLP classifier) on both under-sampled dataset and over-sampled dataset. We also used grid search to tune our hyper parameters. All models have achieved higher performance score on oversampled data. With a logistic regression model, the highest accuracy score is 93.8%; with a random forest model the highest accuracy is 96.5%; with XGboost the highest accuracy we've achieved is 95.9%, and lastly with the MLP classifier the highest accuracy we've achieved is 93.8%.

As previously mentioned, we have a big problem of imbalanced data, although we try to resolve the problem with resampling, the results still shows that our model is blindly predicting the majority class since the precision score on minority class is much lower (<1%) and the precision score on majority class is very high (>90%). We implemented a fix with hand-picked test set, and re-run our best performing model, the random forest classifier. With the fix implemented, the highest random forest score we can get is 73%, with a much more balanced result across precision score, F-score and recall rates between the majority class and the minority class.

To make our model more interpretable, we've also looked at the feature importance in their contribution to making the prediction. It is worth noting that before using hand-pick test set, in our random forest model, the highest contributing factor is prevailing wage, followed by full time position. After implementing the hand-picked test set, the highest contributing factors changed to SOC name (job title) and prevailing wage.

Real-World Implications

As mentioned before, as international students, filing a H1-B petition is something we have to go through if we want to get a legal work status in the U.S.. This project has been very informative for us in the following ways: 1. We get to see what kind of variables the Office of Foreign Labor Certification consider when looking at an H1-B petition 2. According to our model results and feature importance, we get to see factors that might contribute more to a case getting certified/denied. Prevailing wage being an important factor is consistent with our EDA discovery that cases denied have more data points distributed in the lower end of prevailing wage. SOC name is another important feature our model suggests, and although we found no strong feature correlation between SOC name and prevailing wage, we generally understand that jobs in the technology industry and finance industry have a higher prevailing wage than other industries. If we can get more data or time to explore in the future on this relationship, we would like to understand if the current system is biased and favor the higher paying occupations against lower paying occupations.

Moreover, as Chuhan suggested in her conclusion, if we are particularly interested in predicting the H1B cases that will get denied, we can use the approach of anomaly detection to drill down into things that might lead to a case getting denied.

In terms of learnings and reflections on fitting models, we discovered that although generally more data inputs can lead to better model performance, we shouldn’t dismiss the value for less complex models. They actually are handy for prototyping (when tuning hyperparameters) as more complex models need more time and computational resources to run and tune.