H-2B Analysis

Abhinav K, Matthew Z, Samarth D, Trent P

Problem Statement

What is the data science problem you are trying to solve? Why does the problem matter? What could the results of your predictive model be used for? Why would we want to be able to predict the thing you're trying to predict?

The data science problem we attempted to solve was to predict whether or not an employer's application for H-2B visas would be certified given the features of their application. This problem matters because it would give potential applicants insight into how likely their application is to be certified or denied, and would also allow us to explore both the preferences of and possible biases associated with the application process itself.

Ultimately, the results of our predictive model could be used for uncovering trends regarding which applications and attributes tend to lead to an application being certified. In doing so, we would be able to use the model to provide potential applicants with a more accurate perspective on their chances to be approved for an H-2B visa while also providing suggestions to improve their chances for approval.

Data

Describe your dataset. This may also include insights from data exploration.

The dataset we used was the outcomes of H-2B applications during fiscal 2019, taken from the Office of Foreign Labor Certification (a branch under the US Department of Labor). H-2B applications are submitted by employers requesting temporary labor conditions for nonagricultural jobs, and employers are required to establish that there are both not enough US workers qualified and available to perform the job, and that employing H-2B workers will not negatively impact the wages of similarly employed US workers. Although we had originally sought to perform this analysis on H-1B visas, the vast majority of the applicants in the H-1B dataset were granted a visa, resulting in a 100:1 accepted:denied class imbalance. In contrast, the H-2B visa dataset had a much more acceptable 7:1 accepted:denied class imbalance.

The original uncleaned dataset included many irrelevant columns that we dropped for concerns related to inapplicability (VISA_CLASS, which always held a value of "H-2B", and NBR_CERTIFIED_WORKERS and CERTIFICATION_BEGIN_DATE, which only held values if the application was approved), granularity (FEIN, which was a unique identifier for each individual business, EMPLOYER_NAME, which also essentially acted as a unique identifier per application, and CASE_NUMBER, which was literally a unique identifier for each application), data cleanliness (MAJOR, NAME_REQUIRED_TRAINING, and WORKSITE_COUNTY, all of which contained spelling errors, multiple dictations of the same entity, and other associated errors), data sparsity (OTHER_EDUCATION and SECOND_DIPLOMA, all of which were "None"), and repetitive data (JOB_TITLE and SOC_TITLE, both of which were superseded by SOC_CODE, and TRAINING_REQUIRED, which was superseded by NUM_OF_MONTHS_TRAINING).

Specific steps to perform data cleaning are addressed in the following section. After cleaning, we were left only with columns that were either directly relevant or could be used in feature engineering for generating additional relevant features. These features included information about the job being applied for (SOC and NAICS code, representing the occupational identification code and industry code, respectively), whether the employer is represented by an agent or attorney, the number of workers requested, whether the job is full-time, how many hours the job involves, and both the basic rate of pay and overtime rate (if applicable) of the job, the nature of the temporary need, the required education level, the daily work schedule, the work location, if the job is associated with a state workforce agency, and additional information regarding the agent representing the employer. Finally, the class label we used was CASE_STATUS, which we cleaned to create a binary classification problem: the CASE_STATUS was either certified, representing an approved application, or denied.

After cleaning the dataset, we performed an initial examination of the shape and potential trends in the data. In particular, we looked into summary statistics for each feature and created histograms representing the distributions of the categorical features with regard to both approved and denied applications. Many of the trends we had initially hypothesized (such as the effect of attorney representation on visa certification) were reinforced through the examination of the data, and we identified a couple of attributes specific to accepted and denied cases (such as particular job types). The ability to identify various relationships in the data reaffirmed our decision to move ahead and build classifier models for the H2-B dataset.

Method

Describe your data science approach, any assumptions made, nuances, research done, feature engineering done, innovations in your procedure used, etc. Walk us through the process you used.

We began by researching the specific meanings behind the column headers present in the dataset in advance of data cleaning. The uncleaned dataset contained 62 features, and upon initial inspection, we believed that many of them were irrelevant towards the classification problem we were trying to solve. We utilized a provided record layout that explained the meanings behind the columns (https://www.foreignlaborcert.doleta.gov/pdf/PerformanceData/2019/H-2B_FY19_Record_Layout.pdf) as part of our research in determining which features to keep, and which to drop.

Using this information, we chose to drop columns following 5 criteria. First, we dropped the columns that explicitly could not be applied to classification, such as those columns that held either identical values for all records or only contained values if the application as certified. Then, we dropped categorical columns that were extremely granular, such as those columns that were unique identifiers for each application or listed such information as the name of an applicant's attorney, given that such columns would have excessively bloated the dimensionality of our training data post-one-hot-encoding without providing much benefit. From there, we dropped categorical columns that had data cleanliness problems, where the data was rife with misspellings and errors, making it so that its values could not be appropriately used, and we also dropped columns that were excessively sparse, where only a few of the rows had values that weren't "None". Finally, we dropped features that were essentially duplicates of other features, such as the zipcode of an employer when we already had the more suitable city name where that employer resided.

After dropping irrelevant features, we performed additional cleaning on the data. We standardized the class labels, dropping ambiguous application results and converting it into a binary classification problem (certified or denied). From there, we used the values present in repetitive categorical features to “0-fill” the values in their corresponding numeric features; for example, if a row had a value of “No” for the column TRAINING_REQUIRED, then that row’s value for NUM_OF_MONTHS_TRAINING was set to 0. We also converted some of the sparser categorical features into binary categorical features (whether a value was present for that value or not), filled NaN values where appropriate and used the values of one column as direct input into another column (specifically, with OVERTIME_RATE_FROM and OVERTIME_RATE_TO), and used imputation to fill columns that were only filled given the value of another column with a unique representative value. Then, we dropped any rows that still contained NaN values or were duplicates, standardized string-type categorical value columns to be lowercase, and standardized both the SOC_CODE and NAICS_CODE columns to be only two digits to reduce the granularity of those features while retaining broad topic information. This concluded our process of data cleaning.

Then, we performed feature engineering. We started by creating two new columns, STATE_MATCH and CITY_MATCH to represent if the employer and representing agent (if any) were located in the same city / state. We chose to engineer this feature in order to gain insight into whether companies that could afford to hire attorneys located in different cities had an advantage in gaining certification over those companies that utilized local attorneys, and also to avoid the dimensionality associated with the four columns used to engineer these features.

Next, we created the columns WORK_DAY_LENGTH and DAYTIME_WORK, which represented (respectively) the number of hours that the job would entail each day, and whether or not the job was performed over “daytime” hours (starting between 5 - 10 AM, and lasting 12 hours or less). What we sought to understand from this feature is whether jobs with abnormally long work days (or abnormally short work days) and more "normal" daytime hours affected the possibility that that job was successfully certified.

The final feature we engineered was HAS_OVERTIME, which represented whether or not the job offered overtime pay. We engineered this feature in order to better understand if there exists a significant correlation between a job paying overtime pay and the job being certified, and we also wanted to standardize the presence of overtime pay as a binary categorical feature as a reduction from the complexity of the numeric columns OVERTIME_RATE_FROM and OVERTIME_RATE_TO.

We dropped all of the feature used to engineer these features for the reasons that they now contained repetitive information encoded in the newly engineered features, their values were too granular to be effectively used in forming a predictive trend, and they contributed to a greater level of dimensionality in our dataset that was unfavorable to the creation of our predictive model.

Finally, we built our predictive model. Before applying our model, we built a pipeline for hosting all three steps to take place on the training data within the cross-validation loop: preprocessing, PCA, and class balancing. We began by preprocessing our data, scaling the numeric features using scikit-learn’s standard scaler, and then one-hot encoding our categorical features. From there, we also applied principal component analysis to reduce the dimensionality associated with our dataset, in particular seeking to combat the expansion in dimensionality caused by the necessary one-hot encoding.

Then, in order to deal with the class imbalance present in our dataset, we tested two separate class balancing methods in two separate notebooks (ModelSelection-SMOTENC.ipynb and ModelSelection-UnderOver.ipynb). Both of these methods were performed in the cross-validation loop, as we were instructed (via Piazza) to only perform class balancing on the training set. The first method was to only use the imbalanced-learn package's SMOTENC, which allowed for synthetic minority oversampling for a dataset containing both continuous and categorical features. This method was applied before preprocessing and PCA in the pipeline. The second, alternate method was to first utilize imbalanced-learn's ClusterCentroids undersampling technique, which undersampled the majority class by replacing clusters of majority samples by the cluster centroids determined by a KMeans algorithm, and then perform random oversampling of the minority class to enforce a 1:1 class balance in the training set. This method was applied after preprocessing and PCA in the pipeline.

From there, for both methods, we tested the performance of various supervised classification models and their associated hyperparameters on our dataset. We performed model selection on the Decision Tree, K-Nearest Neighbors, Linear Support Vector Classification, Neural Network, and Random Forest models, utilizing a nested cross validation loop to test various hyperparameters for each of the models, and keeping track of the hyperparameters associated with the best accuracy.

Ultimately, after analyzing the performance of each of the individual classifiers, we decided to create a soft voting ensemble classifier, utilizing a Decision Tree, K-Nearest Neighbors, Gaussian Naive Bayes, Linear Support Vector Classification, Neural Network, and Random Forest models, and setting their hyperparameters to the optimal hyperparameters that we derived from the nested cross validation loops. After running the model through one final cross-validation loop, we compared the performances of the final classifiers that utilized the two different class balancing methods, and although there were only minor differences between the two's results (as analyzed by classification report), we ended up choosing the model that used SMOTENC due to its superior performance.

Challenges

Did you run in to any challenges? What worked well vs. what was more difficult than anticipated? Did you try anything without success?

The biggest challenge we ran into was regarding the massive class imbalance that existed within the dataset. The majority of applicants that existed within the dataset were granted visas which meant the initial classifiers we built were only sensitive to the majority class. Both the precision and recall for class 0 (denied) were very low despite a high overall accuracy on the training set. In order to combat this, we first tried to upsample class 0 data and downsample class 1 (certified) data so that the distribution of each class was closer to 50/50. Unfortunately, we later found out that we weren't supposed to actually modify the original data.

We then attempted two methods of performing class balancing. First, we tried modifying our pipeline to include both upsampling and downsampling, and built our final classifier off of that, but the precision and recall for class 0 continued to be mediocre. We also tried modifying our pipeline to utilize SMOTENC as an upsampling technique, but saw only marginal improvements. This goes to highlight how bad the class imbalance really was in our original data given the persistence of subpar classification statistics associated with the minority class despite the addition of class balancing techniques, but regardless, we ended up sticking with SMOTENC for our upsampling technique since it still gave us the best overall performance for our classifiers.

Another challenge we faced in terms of feature engineering was determining thresholds for the features, such as regarding how we should classify each applicant in terms of whether their work involved a day or night shift. Our data initially recorded information regarding the start and end time of each worker's shift. However, since we cared more about the nature of the work rather than the individual start and end times themselves, we decided it would be more beneficial to replace those features with a DAYTIME_WORK column indicating if the work took place over daytime hours. As such, when engineering this feature, one of our problems involved deciding what constituted a daytime shift. We ultimately settled on classifying a job as daytime if it started between 5am and 10am and lasted 12 hours or less. We utilized these bounds as they were consistent with the majority of daytime shifts that existed within the data.

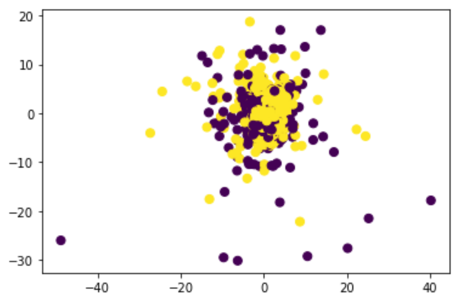
We also faced some challenges during data cleaning. We kept the AGENT_ATTORNEY_CITY and AGENT_ATTORNEY_STATE columns as we thought the presence of an attorney might indicate a higher probability of getting H-2B certification. However, all the rows that didn't have an attorney had N/A values for those two columns, so they got dropped during the data cleaning process. To fix this, we imputed the value "no attorney representation" for all instances of N/A in those two columns. Another challenge we had was with SOC_CODE and NAICS_CODE. Both of these codes were at most 6 digits long, which resulted in an excessively high dimensionality after one hot encoding. After doing some research, we found that the codes are split into sections based on the first two digits, so we truncated the values in the columns to two digits to reduce the granularity and dimensionality while still maintaining topic information.

One more challenge we faced was with trying to incorporate a support vector machine classification model. Originally, we used scikit-learn.svm.svc as our support vector classifier, but when we introduced class balancing into our cross-validation loops, our model selection loop for determining its ideal hyperparameters took an excessively long period of time. After looking into it, we discovered that it had a quadratic time complexity for fitting samples, and the introduction of class balancing was vastly increasing the number of training samples to be fitted on. As such, in order to ensure that we could perform cross-validation and hyperparameter selection in a more realistic timeframe, we switched to scikit-learn.svm.LinearSVC to perform support vector classification in linear time complexity, and were able to complete hyperparameter selection at a much more suitable rate.

Results

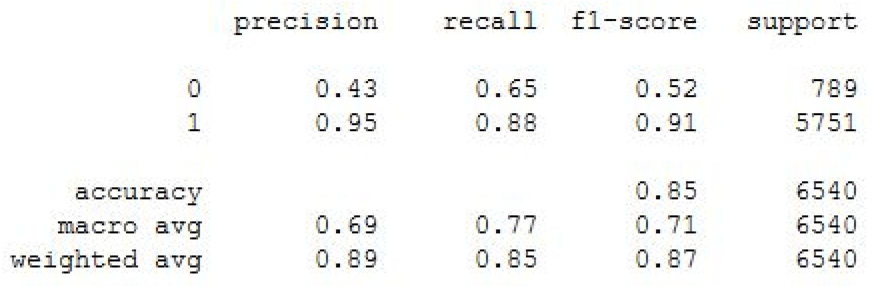
What were your results?

The general trend, regardless of the model, is that the models were much better at classifying applications from the majority class (records that were certified) than records from the minority class (applications that were denied), recording higher precision, recall, and f-score values. This is most likely due to the class imbalance problem mentioned earlier, where even despite our efforts to perform class balancing in the training set in the cross validation loop, the lack of additional actual minority class examples and the inability to easily distinguish minority records from majority records (as seen when plotting the records in 2 dimensions) hampered our classifier's ability to properly classify minority records.



Scatterplot of the majority and minority records, post pre-processing and PCA, projected into 2 dimensions.

However, we came to select the ensemble classifier because although specific models might have had better precision, recall, or f1-score values, these models generally traded off their higher values for one class for much lower values for the other class, whereas the ensemble classifier maintained consistently high statistics as reported by the classification report. Between the two final ensemble classifiers we created, which used different class balancing techniques, there was almost no difference between their statistics in the classification report, except that the SMOTENC-using classifier recorded a precision of 0.43 on the minority class compared to the oversampling-and-undersampling-using classifier's recorded precision of 0.42. However, in addition to this marginal difference, we noted that only utilizing SMOTENC was much faster than using both undersampling and oversampling, so we chose the SMOTENC-using ensemble classifier as our final predictive model.



```
              precision    recall  f1-score   support

           0       0.43      0.65      0.52       789
           1       0.95      0.88      0.91      5751

    accuracy                           0.85      6540
   macro avg       0.69      0.77      0.71      6540
weighted avg       0.89      0.85      0.87      6540
```

Classification report of the soft-voting ensemble classifier using SMOTENC

Finally, with regard to our final predictive model's overall statistics, the model performs very well at classifying certified applications, recording a precision of 95%, a recall of 88%, and an f1-score of 0.91. Moreover, although its classification of denied applications recorded worse results, including a precision of 43%, a recall of 65%, and an f1-score of 0.52, with regard to our original context of a class imbalance weakening our ability to classify records of the minority class, our final model's application of SMOTENC to perform class balancing resulted in much improved performance in classifying denied applications.

We are very pleased with our final predictive model's performance. As discussed prior, we initially perceived binary classification of the applications to be a difficult problem, but the high accuracy of the model, with its phenomenal performance on certified applications and its favorable performance on denied applications, surpassed expectations and invokes additional exploration into expanding the usage of this classifier to additional settings.

Next Steps

What could be done with this next? How could the results be used in the real world? What additional data analysis could be done on the data? Could additional data be added/supplemented to predict more things? Etc.

Given the high accuracy of our model, we believe that a potentially viable business model exists in that we could establish a service that assists employers applying for H-2B visas in checking whether their application is likely to be certified or denied. By having a sort of “offline judge” for H-2B visa applications, employers could better tailor their applications to improve their odds of having their application be certified by the model, and by extension, the actual H-2B process as well. Moreover, as seen through data exploration, we could also offer potential applicant employers access to our aggregated statistics and visualizations associated with certified and denied applications so that they could get a better idea of the ideal features their application should have to optimize their chances of being successfully certified, and even provide employers with data-informed decisions on whether they should even apply for a visa given the low probability of success by nearest neighbor applications.

Given that we only had a limited time to work with this dataset, the depth of data exploration and model selection that we performed was not exceptionally thorough. As done in the data exploration notebook, we could perform additional analysis to determine whether certain features are more influential in deciding an application’s outcome, which could be used to allow potential users to determine which attributes they should focus on, and we could also come up with additional feature engineering ideas from dropped / present columns for a more nuanced representation of the dataset. Moreover, additional data could be added to our current dataset, given that H-2B application data is easily accessible all the way back to fiscal year 2008, and this additional data would create a richer dataset for us to train on and potentially boost our accuracy as well.