CS580L
Introduction to Machine Learning

- PROJECT REPORT -

U.S Permanent Visa Applications

Pragya Vishalakshi
Burak Sivrikaya

1. Problem Definition

A permanent labor certification issued by the Department of Labor (DOL) allows an employer to hire a foreign worker to work permanently in the United States. And USCIS receives and processes about 6 million immigration applications from individuals and employers every year. Counting applicants already living in the U.S., experts estimated up to 5.5 million green card applications were pending at the close of 2012. Hence, it becomes easy for the applicants and their employers to know based on the data of the applicants whether they would be able to get the visa certified or not.

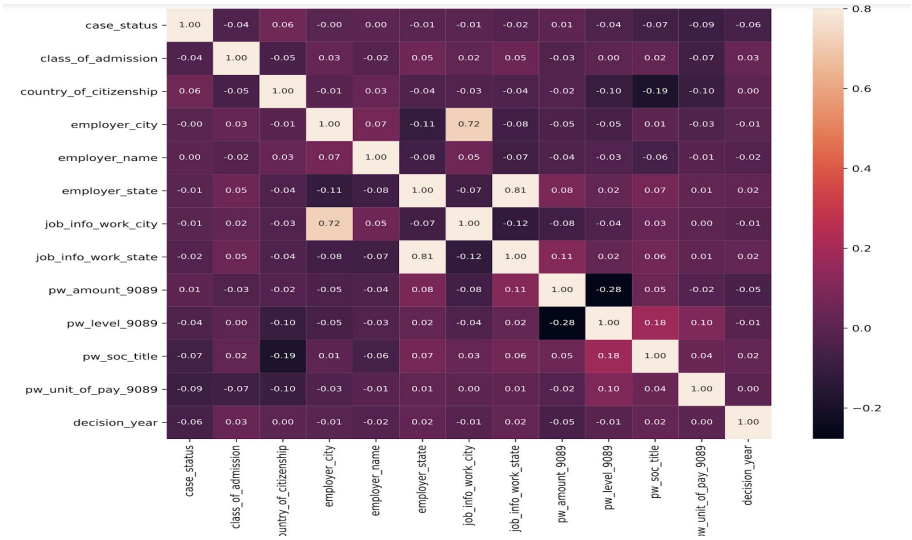
Our problem is a classification problem, predicting whether the applicant will be certified with the visa they applied for or not based on their data that they provide to Homeland Security.

2. Feature Analysis

Dataset was taken from Kaggle. They had the data from Department of Labour which contained **374362** applicants and **153** features (including target variable) features for initially.

Observation in dataset after feature analysis:

1. There were features for which had maximum of the applicants who didn't had any value.
2. There were features which had no relevance, as we already had this information through other columns. For example: 'employer_address_1', 'employer_postal_code' etc
3. The date of application was in exact date format and similarly the wages were in all category of hourly, bi-weekly, yearly etc, to make dataset consistent.
4. Understand **Multicollinearity** and handle it. For example: as we see from below correlation matrix that job_info_work_city gives almost same information as employer_work_city.



3. Pre-Processing

After analysing the features, we selected the important features and the applicants who had all the values for the selected features.

Following are the features for which pre-processing was done to make it usable:

1. Decision status (Target variable):

   There were four different values for this ("Certified", "Denied", "Certified-Expired", "Withdrawn"). As we were only dealing with approval of visa or not, we changed "Certified-Expired" to "Certified" and removed "Withdrawal" applicants. This made the problem a **binary-classification** problem, giving us more algorithms to apply and test.
2. Job Title:

   Job titles were updated to be consistent. For example: "Developer", "Software Engineer", "Software developer" etc were all categorized under "Software developer" and so on.
3. Yearly salary:

   Wages were converted in yearly wage and the **outliers** were removed which occured because of converting to yearly wage.
4. Decision year:

   Decision date was converted to decision year covering data from 2012-2017 year.

Following are the features whose value were same as initial dataset after processing:

5. Class of admission
6. Country of citizenship
7. Employer name

8. Employer state
9. Permanent work level

Other pre-processing:

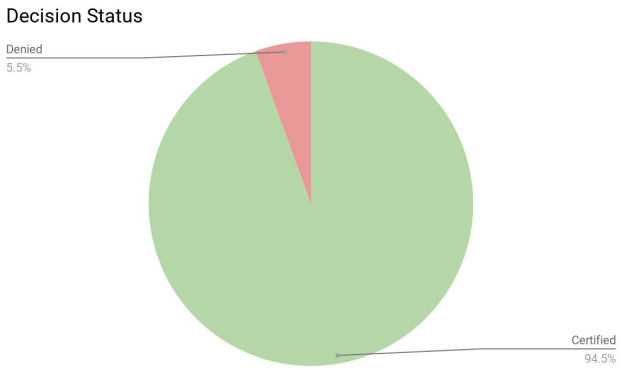
1. **Labelling**: Data was labelled **numerically** so that different algorithms could be used as the dataset was mix of numerical and textual data.
2. **Scaling**: After labelling, the dataset had huge scaling issue as all the data were of different range mostly the wage data. So we used **MinMax** Scaling to bring all the data for all the features in same range for the applicants.

After pre-processing we were finally dealing with **291052** applicants and **9** features (including target variable).

4. Model Training

Problem of Imbalanced Datasets

Before diving into model training, we realized a problem in our dataset: **the classes were imbalanced.** This problem appears in many domains like fraud detection, disease screening in which there is a disproportionate ratio of observations in each class. Likewise, among 291,052 applications, 274,941 of them were certified, and only 16,111 of them were denied.



The problem with imbalanced datasets is that most of the machine learning algorithms are designed to maximize overall accuracy. But, accuracy is misleading in our case, why? Let's assume a model that always gives "Certified" as predicted decision no matter what you give as the input. With that model, we could get ~95% accuracy since most of the test instances were also coming from "Certified" class, but we wouldn't be able to detect any "Denied" cases. Then that would be a helpful model and it shows that accuracy is not good indicator of performance in this situation.

There were different approaches to combat the problem of imbalanced. We considered three approaches and decided to use two of them.

1. Up/Down Sampling. We decided **not** to use that approach. Downsampling basically discards potentially useful data. The main disadvantage of upsampling is that by making exact or very similar copies of existing examples, it makes overfitting more likely. Also, it increases and computation cost and learning time**.** This was our argumentation of skipping this approach.
2. Using tree-based or cost-sensitive algorithms that increase the cost of classification mistakes on the minority class. We applied some specific algorithms (like penalized-SVM) and the outcomes are discussed in Results section of this report.
3. Changing performance metric. We've discussed that accuracy is not a good metric for imbalanced datasets. So, instead, we decided to use **Area Under ROC Curve (AUROC)** as our main performance indicator. AUROC represents the likelihood of a model distinguishing observations from two classes. It's only applicable for binary classification and it requires prediction probabilities. Both requirements were satisfied by our problem and selected algorithms.

Training & Validation Process

Steps of our model training and validation were as below:

1. Data was split by giving 75% (~225K) to training and 25%(~75K) to testing.
2. Stratified 5-folds were created for cross-validation. Stratification was crucial at this point because it keeps the disproportion of classes in each fold and that makes our training & validation more realistic.
3. Seven different classification algorithms were selected and, using GridSearchCV method of **scikit-learn** library, grid search was applied with given set of parameters of each algorithm.
4. Best parameters maximizing AUROC for each algorithms were selected.
5. Models with their best parameter sets were tested against our test dataset.

Hyper-parameter Selection

Selected algorithms and parameter types for which we gave different set of values for each were as below:

- Logistic Regression(LR)
  Regularization type, regularization constant
- Multilayer Perceptron(MLP)
  Learning rate, activation function, alpha, hidden layer sizes
- Penalized Support Vector Classifier(SVC)(with bagging)
  Kernel, C, Gamma, number of estimators, number of maximum samples

- Decision Trees(DT)
  Max depth, max features
- Random Forest(RF)
  Max depth, max features, number of estimators
- AdaBoost
  Learning rate, number of estimators
- K Nearest Neighbor(k-NN)
  Number of neighbors

5. Results

In the table below, results coming from each algorithm are provided. In the first line, our performance evaluation metric, AUROC values, are given for the test set. Apart from that, AUROC value of best-parameter set in the training dataset, test accuracy, false positive, false negative, true positive and true negative values are shown.

	LR	DT	RF	Adaboost	k-NN	MLP	SVC
test_auc_roc	0.666	0.723	0.752	0.73	0.721	0.687	0.633
test_acc	0.9446	0.9449	0.9462	0.9460	0.9448	0.9446	0.9446
train_auc_roc	0.666	0.718	0.746	0.73	0.711	0.681	0.625
false_positives	4028	3886	3880	3128	3967	4028	4028
false_negatives	0	120	33	798	51	1	0
true_positives	68735	68615	68702	67937	68684	68734	68735
true_negatives	0	142	148	900	61	0	0

Interpretations

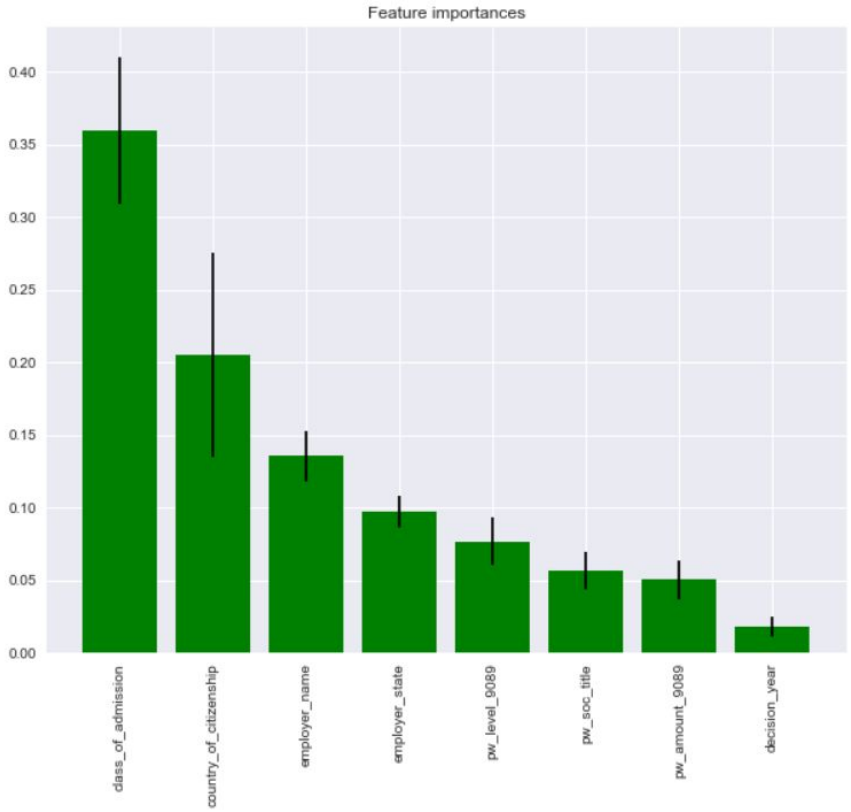
- There was almost no difference in test accuracy amongst all algorithms.
- Linear models (LR, Linear-SVC) and simple models(MLP with few layers) worked poor.
- Penalized-SVC was potentially promising but it performed very bad. However, we should give more details about that. SVC is a costly model and in official documentation of scikit-learn library's SVC implementation, it is said that after 10K samples, the algorithm gets quadratically slower. We had around 200K instances for training and when we wanted to try different hyper-parameter sets, cross-validation took forever. As an alternative approach, we applied

applied bagging SVC and created classifiers by limiting number of samples to 5K for each them. This solved computation problem; however, we could feed our bagging SVC models with very few data and we think that it caused poor results.

- Algorithms mentioned above couldn't distinguish minority class cases from majority clases cases and predicted everything as "Certified" (with one exception for MLP)
- Tree based algorithms worked well. Their hierarchical structure allowed them to learn signals from both classes.
- Tree ensembles outperformed decision trees.
- k-NN was not influenced by the size of the classes.
- Boosting helped imbalanced classes problem.

Random Forest performed the best. So, **which features are more important?**

Since random forest is a white-box model, we can get an idea about the features that contributed more to the model. In the graph below, mean and standard deviation of importance of features coming from the trees that build the random forest model, are given.



We see that all features have a degree of importance in our final model. Class of admission(like H1B, J1, F1 etc.) is by far the most important feature. Country of citizenship and employer information follow it as the second and third important features respectively. This insight can be useful in future research.

6. Future Work

There is a big room for improvement in our research. We could get decent results but they can be improved with some future work.

As in most of research, we also had some limitations. We had to run all algorithms on our laptops and this did not let us to try costly models very efficiently. With high-performance computers, more extensive hyper-parameter selection and costly but potentially powerful algorithms could be applied. Deep learning algorithms can be applied. Also, using different datasets like H1B visa applications which is available on Kaggle, our model can be improved in the future.

7. References

- https://www.kaggle.com/jboysen/us-perm-visas
- https://elitedatascience.com/imbalanced-classes
- http://www.dataschool.io/roc-curves-and-auc-explained/