# Implementing Different Machine Learning Algorithms on Old Bailey Decisions Dataset

Mehdi Ganjkhani, U1268148

***Abstract*—In this project we are going to implement different classifiers to predict if the convicted person is guilty or not based on the transcript features and victim characteristic in the Old Bailey project. I implemented perceptron (two submission), super vector machine (SVM), logistic regression, SVM over tree and neural network (using a library) for this task. Based on the results, the performance of the algorithms is quite similar with the same dataset. In addition, the feature selection performs a key role in the classification task.**

***Index Terms*—Classification, feature selection.**

## I. Introduction

### A. Project Background and Problem Description

THE Old Bailey is the colloquial name for the Central Criminal Court of England and Wales, which deals with major criminal cases in Greater London, and also sometimes from the rest of England and Wales. This court has existed in some form or another since the 16th century. The Proceedings of the Old Bailey are made up of 120 million words, recording 197,000 trials held at the Old Bailey, or Central Criminal Court in London, between 1674 and 1913. The proceedings of this court have been digitized and are available online in [1]. In order to create a fully searchable resource, it was necessary to digitise the text (and not just scan page images) of the proceedings. The text of the 1674 to October 1834 proceedings was manually typed by the process known as "double rekeying". The text of the November 1834 to 1913 Proceedings was manually keyed once and a second transcription was created using optical character recognition software. The two files were then compared and differences between them manually resolved.

Since all the digitised text of the trials from 1674 to 1913 are available, we can ask the following text classification question:

- Is it possible to predict the decision of the court using the transcribed dialogue during a trial?
- If yes, how likely the prediction is correct?
- which classification algorithm performs the others?

The goal of this project is to answer these questions.

### B. Project dataset

Each example for classification is a single trial. In a single trial, there may be more than one defendant, and more than one charge. To simplify things, this project has restricted the dataset to trials where there is exactly one defendant and one charge. As a result, each trial has exactly one outcome: eitherguilty (label 0) or not guilty (label 1). In this project in total 25,000 examples are randomly selected as a subset of the entire data, which originally consists of 197,000 examples. These examples have been split into three subsets as below:

- **train:** This is the training split of the data, on which the models are trained after hyper-parameter selection. The training split consists of 17,500 examples.
- **Test:** This is the test set that are used to evaluate the models locally. The test set consists of 2,250 examples.
- **Eval:** This is the "evaluation" set with 5,250 examples. The labels for these examples are hidden. The trained classifiers should predict labels for this dataset and then the labels should be uploaded to Kaggle.

### C. Feature represenation of the Trials

Instead of working with the raw text directly, the data has been pre-processed and four different feature sets are extracted as follows:

*1) bag-of-words::* The bag of words representation represents the text of the trial as a set of its words, with their counts. That is, each dimension(i.e feature) corresponds to a word, and the value of the feature is the number of times the word occurs in the trial. To avoid the dimensionality from becoming too large, we have restricted the features to use the 10,000 most frequent words in the data.

*2) Term frequency–inverse document frequency:* : The term frequency–inverse document frequency (tfidf) is a popular document representation that seeks to improve on the bag of words representation by weighting each feature in the bag-of-words representation such that frequent words are weighted lower. As with the bag-of-words, we use the 10,000 most frequent words in the data.

*3) Glove:* : The previous two feature representations are extremely sparse vectors, with only a small fraction of the 10,000 features being non-zero for any vector. The glove representation represents a document by the average of its "word embeddings", which are vectors that are trained to capture a word's meaning. The word embeddings are dense, 300 dimensional vectors, and for the purpose of this project, each word is weighted by its tfidf score.

*4) Miscellaneous:* : In addition to the above-mentioned features, also miscellaneous categorical attributes are extracted from the trial data. This consists of the following features for each trial:

1) defendant age
2) defendant gender
3) number of victims
4) genders of the victims
5) offence category
6) offense subcategory

TABLE I
MISCELLANEOUS DATASET BEFORE PRE-PROCESSING

| defendant_ age | defendant_ gender | num _victims | victim_ genders | offence_ category | offence_ subcategory |
|---|---|---|---|---|---|
| 62 | female | 1 | male | theft | theftFromPlace |
| 17 | male | 1 | male | theft | pocketpicking |
| not known | male | 1 | male | theft | pocketpicking |
| not known | male | 1 | male | theft | simpleLarceny |
| 52 | male | 1 | female | theft | pocketpicking |
| 40 | male | 0 | NaN | sexual | bigamy |
| not known | male | 1 | female | theft | pocketpicking |
| not known | female | 1 | male | theft | housebreaking |

### D. Project milestones and report flow

In order to overcome the project milestones I used Perceptron, Super vector machine (SVM), Logistic regression (LR), SVM over trees and neural network (NN).

## II. PRE-PROCESSING AND IMPLEMENTATION

All of the classifiers and the pre-process are implemented in Python in this project. In order to present clear results and illustrate the flow, Jupyter is utilized as the coding interface. Among the four feaure representations of the dataset, three of them (bow, tfidf, glove) are already pre-processed and do not need to need to be processed further. However, the miscellaneous features are required to be pre-processed before being used for classification task. The miscellaneous dataset contains categorical variables. These variables are mainly text values which represent various traits. In order to be able to use them in classification they need to be converted to numerical dataset. Table I illustrates the miscellaneous data before pre-processing. As can be seen, except column 1 and 3 the other columns contain text dataset. In addition, column 1 has some missing data as column 4. I substituted the no number arrays in column 1 with 0 and NaN in column 4 with no_gender. A common approach to covert text to number is called one hot encoding. Despite the different names, the basic strategy is to convert each category value into a new column and assigns a number to the column. This method is used for the miscellaneous data to convert the texts to numbers. The final dataset contains 139 numerical features. More procedure and details can be seen in the simulation files. The baseline has been run to achieve more information about the data. The train dataset contains $50.34\%$ of no-guilty samples, however, this number for test dataset is $48.84\%$. After processing the miscellaneous dataset and adding the labels to it, different dataset are trained using the Perceptron algorithm to quantify the performance of the algorithm for classifying the dataset. As it will be explained in the next subsection, using miscellaneous and tfidf provided the best result among all of the dataset. Therefore, these dataset are selected and used for the other classifiers. The result for each classifier is presented in the next subsections.

### A. Perceptron Algorithm

As an online algorithm and the first one after decision tree, because of having flexibility with using various data (float and int) this classifier is opted as the first option for the project.

TABLE II
PERCEPTRON ALGORITHM PERFORMANCE FOR DIFFERENT DATASET

| **Dataset** | **r** | $\eta$ | **Cross val. acc. (%)** | **Train accuracy (%)** | **Test accuracy (%)** |
|---|---|---|---|---|---|
| glove | 0.01 | 0.1 | 64.62 | 66.87 | 65.11 |
| bow | 0.01 | 1 | 70.63 | 86.43 | 71.24 |
| tfidf | 0.01 | 0.1 | 71.93 | 80.88 | 72.84 |
| misc. | 0.1 | 1 | 78.46 | 78.94 | 79.47 |
| tfidf+misc. | 0.1 | 0.5 | **80.98** | **86.81** | **82.36** |

Among all version of the perceptron, margin perceptron was the one with the highest accuracy.

The tuned hyper-parameters and the cross validation and test accuracy is reported in Table II in detailed for all individual dataset and combination of misc. and tfidf sets together, where $r$ and $\eta$ denote learning rate and margin respectively. As can be seen, miscellaneous + tfidf data outperforms the other datasets. However, bow has high accuracy in training set, but the test set accuracy was much lower which can be because of overfitting.

### B. Super Vector Machine Algorithm

As we saw in the previous subsection, miscellaneous + tfidf dataset had the highest accuracy. However, I decided to focus on misc. dataset for the rest of algorithms because the misc. dataset had good accuracy in perceptron algorithm. In addition, the size of the dataset is very small comparing tfidf. The tuned hyper-parameters and the accuracy for cross validation, train and test sets are illustrated in Table III, where $\gamma$ and $C$ denote learning rate and trade-off cost respectively.

TABLE III
SVM ALGORITHM PERFORMANCE FOR MISCELLANEOUS DATASET

| $\gamma$ | $C$ | **Cross val. acc. (%)** | **Train accuracy (%)** | **Test accuracy (%)** |
|---|---|---|---|---|
| 0.0001 | 1000.0 | 78.19 | 78.57 | 79.73 |

### C. Logistic Regression Algorithm

Similar to SVM algorithm I used the same dataset for Logistic regression. The tuned hyper-parameters and the accuracy for cross validation, train and test sets are illustrated in Table IV, where $\gamma$ and $\sigma^2$ denote learning rate and trade-off cost respectively.

TABLE IV
LOGISTIC REGRESSION ALGORITHM PERFORMANCE FOR MISCELLANEOUS DATASET

| $\gamma$ | $\sigma^2$ | **Cross val. acc. (%)** | **Train accuracy (%)** | **Test accuracy (%)** |
|---|---|---|---|---|
| 0.001 | 100 | 78.09 | 78.09 | 78.89 |

### D. SVM over tree Algorithm

In order to implement SVM over tree I used miscellaneous data again. However, in order that the dataset can be used for decision tree implementation, I converted the defendant age and number of victims to one-hot encoding features so all

the features are binary and can be used in decision tree. The dataset contains 226 one-hot features and used for the SVM over tree. The tuned hyper-parameters and the accuracy for cross validation, train and test sets are illustrated in Table V.

TABLE V
SVM OVER TREE ALGORITHM PERFORMANCE FOR MISCELLANEOUS DATASET

| $\gamma$ | $C$ | depth | Cross val. acc. (%) | Train accuracy (%) | Test accuracy (%) |
|---|---|---|---|---|---|
| 0.0001 | 1000.0 | 4 | 78.49 | 79.08 | 79.91 |

### E. Neural Network Algorithm

Similarly, the miscellaneous dataset is used for the Neural Network implementation and evaluation. In order to implement NN, I used "sklearn" library in python. The "adam" optimizer is used in the training process, and different neurons and hidden layers are tried in addition to different learning rate. I tried a network with 1, 2 and 3 layers with different number of neurons. The tuned hyper-parameters and the accuracy for cross validation, train and test sets are illustrated in Table VI. As can be seen in the results, the network with only one hidden layer outperforms the other hyper parameters with more hidden layers. This illustrates that the simpler network works better than more complicated network for this dataset.

TABLE VI
NEURAL NETWORK ALGORITHM PERFORMANCE FOR MISCELLANEOUS DATASET

| $\gamma$ | neurons | Cross val. acc. (%) | Train accuracy (%) | Test accuracy (%) |
|---|---|---|---|---|
| 0.0001 | 50 | 78.9 | 79.09 | 79.82 |

## III. Comparison Between Different Methods

Now that the all five algorithm are implemented using the most relevant data feature, here we can compare the performance of the algorithm in the classification task. In total six different scenarios should be implemented and the results should be submitted in Kaggle. These six scenarios are chosen based on the highest accuracy as follows:

- **Case I**: The dataset for this case is miscellaneous features and the classifier is perceptron algorithm.
- **Case II**: The dataset for this case is miscellaneous and tfidf features at the same time, and the classifier is perceptron algorithm.
- **Case III**: The dataset for this case is miscellaneous features and the classifier is SVM algorithm.
- **Case IV**: The dataset for this case is miscellaneous features and the classifier is Logistic regression algorithm.
- **Case V**: The dataset for this case is miscellaneous features and the classifier is SVM over trees algorithm.
- **Case VI**: The dataset for this case is miscellaneous features and the classifier is neural network algorithm.

The train, test and evaluation set accuracy for the above-mentioned cases is reported in Fig. 1. As can be seen, the perceptron algorithm with misc. and tfidf (Case II) outperforms the other cases in training, test and evaluation accuracy. The result is also shown in Table VII showing that apart from Case II, the evaluation accuracy is almost similar together and they have slight difference. We can observe that with the same dataset if we implement the algorithm and select the hyper parameters correctly, the result is going to be close together.

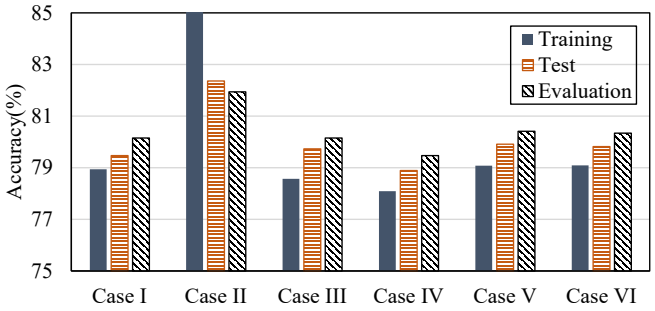


Fig. 1. Training, test and evaluation accuracy for different scenarios

TABLE VII
TRAINING, TEST AND EVALUATION ACCURACY FOR DIFFERENT SCENARIOS

| | Case I | Case II | Case III | Case IV | Case V | Case VI |
|---|---|---|---|---|---|---|
| **Training** | 78.94 | 86.81 | 78.57 | 78.09 | 79.08 | 79.09 |
| **Test** | 79.47 | 82.36 | 79.73 | 78.89 | 79.91 | 79.82 |
| **Evaluation** | 80.15 | 81.94 | 80.15 | 79.47 | 80.41 | 80.34 |

## IV. Discussion and Conclusion

As illustrated, the performance of the algorithm is highly dependant on the selected features. That is why feature selection is one of the most important tasks in machine leaning even most of the time more than the algorithm itself. What I have learned in this project was that first of all we need to represent the feature and analyse them to acqire the intution and the impact of each feature on the classification task. For instance, the age of the defendant can be a key feature to improve the accuracy even if we have several missing data. I have learned how easy it is to pre-process a text oriented dataset and convert it to numbers to use in algorithms. I have learned about handling different dataset in python and use them for the required task. I learned how to select the hyper-parameters for algorithm and how to use the optimum one.

In addition to the algorithms that I implemented and reported I tried several different approaches but some of the did not work. I noticed that in order to obtaion higher accuracy, the algprothm is not going to be helpful too much. Instead, we need to use the feature transformation or feature reduction to include all the useful features together. I tried some feature reduction methods like PCA, and normalization but it was not helpful. In addition, I implemented a feature reduction method using entropy and information gain. In order to do that, first, the whole dataset is converted to binary using normalization and threshold and then the information gain for each feature is calculated one by one. Finally the specific number of features which have the better information gain are selected and the rest are discarded. I tried this method and implemented and used

10% of tfidf + misc. features which gave me a good evaluation score (80.6%). However, the process for information gain calculation took a while and that was the reason that I did not include the codes and the results for the other methods. If I had much more time I would definitely be trying to find or implement an efficient method to reduce the feature of the bow, glove and tfidf feature representations and combine them to improve the performance of the classification method.

In conclusion we saw that with the same feature provided that the hyperparameters are selected correctly, the performance of the different classifiers are going to be similar. In order to achieve higher performance, we should work on the feature selection and transformation.

## V. Implementation Details

All the Code implementation is done in Jupyer interface in Python. There are in total 4 Jupyter notebooks that I submitted. In the beggining of each of the notebook, first the misc. and tfidf dataset are loaded. Then the misc. dataset is converted from text to numeric data. The process can be followed based on the order of each cell. Then, the labels are added to the misc dataset using tfidf. The training dataset is splitted to 5 folds for cross validation task. After cross validation and the hyper-parameter selection, the train set is used to train the network using the implemented algorithm. The test set is evaluated and the result is printed. Finally, labels are generated for the evaluation samples and stored as CSV files in the same directory. The correlated dataset should be copied in the same directory to load the dataset. The first notebook is related to perceotron implementation for 5 different datasets. The second notebook is for SVM and logistic regression for misc. dataset. The third notebook represents the implementation and result for SVM over tree algorithm. Finally the last notebook provides the code and result for NN.

## References

[1] T. H. Clive Emsley and R. Shoemaker, "Old bailey project description," https://www.oldbaileyonline.org/, 2020.