Old Bailey Decisions: Final Report

U1265169

1. Overview of the Project

Old Bailey Decisions holds digitized records of the Old Bailey Court Proceedings. Every proceeding deals with criminal court cases from England and Wales with a judgement being passed as “guilty” or “not guilty”.
Aim: This is the crux of our project. We are training classifier models on the train data to predict the decision of the court - “guilty” or “not guilty”.

We have been given different feature representation of the data.

- BOW – bag of words dataset represents most frequently used words in the trail and its counts. 10000 words have been considered for each trial.
- Tfidf – Is an improvement over bow representation, where the frequent words are weighted down using the idf. Since bow has a dimension of 10000, Tfidf also has 10000.
- Glove – This dataset consists of word embeddings weighted by Tfidf.
- Misc – This dataset has categorically attributes.

Each of these datasets has a train, test and eval files. We can pick any dataset and train our model on the training files of that dataset. Test our model on the test file and evaluate it on the eval.

We must be very sure we do not use any of the testing file data while training, this will corrupt our model. This might give us less empirical error but high generalization error on eval dataset.

2. Important Ideas Explored.

- Data Cleaning/pre-processing
- Cross – Validation
- Hyper-parameters
- Evaluation techniques
- Overfitting/Underfitting
- Generalization error
- Python Packages of Random Forest and Neural Nets
- Algorithms: Perceptron,ID3,SVM,Logistic Regression, Ensemble.

3. Ideas from the Class

I have ventured into the topics explained in the class and the algorithms implemented in the assignments.

- The project has label as 0 and 1 but most of our algorithms taught in the class need -1 and 1; So that the weights can be updated accordingly. For e.g. perceptron needs -1 and 1.
- Scaling/Normalizing the features help to lower the effect of the dominant features and standardize the dataset and give a good accuracy. We can also scale the size of the dataset.
- Cross Validation is an important concept from class, it has helped in finding the apt hyper-parameters to train the model for making it work at its optimum.
- Hyper-parameters, they help to tune the algorithm better or to avoid over-fitting.
- Binning/discretizing feature values, it is an important concept for working with categorically data, it is very useful for working with misc dataset.
- Ensemble and Random Forest is another idea I have implemented from the class, which helps in boosting the accuracy of weak classifiers.
- I have used algorithms like perceptron and its variants, SVM, ID3, Random Forest and ensemble.

4. Learnings

The course of the project has taught the following:

- Data given needs cleaning; we cannot just plug un-processed data into an algorithm. e.g. the miscellaneous dataset had some unknown values or age written as “forty” in words rather than in the numerical format. Data cleaning/pre-processing was an important part of this project.
- Classifiers like perceptron gave good accuracy on training but did not do as well on test and eval dataset. That is, it had a low empirical error but comparatively high generalization error, suggesting that if we gave it more example to train on it would have given a better generalization error or a higher accuracy on train and eval. [my understanding]
- Or it was over-fitting the training dataset.
- Misc dataset, seemed to work the best with most of the algorithms, my take on this is in other datasets it had noise and hence needed some tweaking in the sense of knowing which columns to consider while training a model, whereas misc dataset was data rich and compact.
- ID3 works well with categorical data but needs binning to take care of real value numbers and if not binned properly then it can lead to recursion errors while testing.

5. Results

i. Average Perceptron:

This is the first algorithm I implemented. It is a variation of a simple perceptron. I used the TFIDF dataset as it was already scaled. In my opinion it has helped in training the model better. Also, the dataset had numerical values so, it was easy to implement the algorithm. I had to do pre-processing on the "libsvm" format and changed the labels to -1 and 1 so that the weights get updates correctly.

Filename: avg_perceptron_tfidf_submission1.py
Dataset: Tfidf
Accuracy on Train: 80.14 %
Accuracy on Test: 70.488 %
Accuracy on Eval: 71.161 % [on Kaggle results(2).csv]

ii. Margin Perceptron:

This was part of my second checkpoint submission. It is also a variant of simple perceptron. I wanted to explore margin hyper-parameters along with the learning rate. I used BOW dataset. I used the max value of each column to scale the features. I was hoping for a better accuracy than average, but it did not give me a boost as I had expected. I feel my scaling was not appropriate. But I observed that the generalize error is not far from the empirical error, I feel by using two hyper-parameters I was able to control over-fitting.

Filename: avg_perceptron_tfidf_submission1.py
Dataset: BOW
Accuracy on Train: 72.09142857 %
Accuracy on Test: 67.3 %
Accuracy on Eval: 69.9% % [on Kaggle results.csv]

iii. ID3:

This was also a part of my second checkpoint submission. To implement this algorithm, I had to pre-process the dataset quite a bit. I used the misc dataset. I had to use nlp techniques to extract the age column and created a dataframe to have labels column.

For binning I used the following techniques:

Binned using a specified intervals: 8<=age <= 22 – group A, 22< age<=38 – group B, 38<age<=54-group C ,54 < age <=70-groupD and 70<age -group E. This gave me recursion error in test (I guess my implementation was faulty)

Binned using the mean of the column into 2 bins thinking many bins were causing the issue age<mean- group A and mean>=age -group B. This gave me a very low accuracy on test.

Hence, I considered age as multiclass feature and just applied ID3 on it without binning.

Filename pre_processing_misc.py; ID3_misc.py
Dataset: misc
Accuracy on Train: 81 %
Accuracy on Test: 77 %
Accuracy on Eval: 75% [on Kaggle results.csv(old)]

iv. Ensemble Perceptron-ID3:

This is a part of my checkpoint3 submission. As mentioned above my ID3 implementation was incorrect as I had not performed binning. So, after HW6 I got an idea to implement ID3 without requiring binning.
So, I used two weak classifiers.
Used average- perceptron to train on misc dataset and create a bag of classifiers - 200(bagging).
Used them to predict for each row of train, test and eval to get a new dataset consisting of N rows and 200 columns of (-1,1).
Trained the ID3 on this new dataset to find the root and used that to predict on test and eval.

Filename: perceptron_tree_ensemble_working.ipynb
Dataset: misc
Accuracy on Train: 81 %
Accuracy on Test: 79.27 %
Accuracy on Eval: 79.390% [on Kaggle perceptron_id3_ensemble.csv]

v. SVM (support vector machines)

This is a part of my checkpoint3 submission. I used SVM that we had implemented in class for HW6. I used BOW for SVM, to compare if this gives any better results than the margin-perceptron on the BOW. It does give an improvement but not much.
I feel BOW dataset has noise and I need to weed out the unimportant columns.

Filename: implementing_svm.py
Dataset: BOW
Accuracy on Train: 74.04571%
Accuracy on Test: 71.1%
Accuracy on Eval: 70.81% [on Kaggle svm.csv]

vi. Ensemble SVM-Perceptron

This is a part of my checkpoint3 submission. I used an Ensemble of SVM and Perceptron to see if both together can give better accuracy.
So I implemented it on two datasets
Tfidf so that I could see the difference between just one algorithm (SVM/Perceptron) and its ensemble. Could see slight improvement.

Filename: perceptron_svm_ensemble_tfidf.ipynb
Dataset: Tfidf
Accuracy on Train: 80.1 %
Accuracy on Test: 72.7%
Accuracy on Eval: 73.330% [on Kaggle perceptron_svm_ensemble_tfidf.csv]

Implemented it on misc too, just to see if this gives me a better score proving the point that Tfidf and bow has noise in the dataset and I need to take care of it .

Filename: perceptron_svm_ensmble.ipynb
Dataset: Tfidf
Accuracy on Train: 83.3 %
Accuracy on Test: 79.27%
Accuracy on Eval: 79.504% [on Kaggle perceptron_svm_ensemble.csv]

vii. Python Library: Random Forest

Throughout the project I have been trying to get my ID3 work in its normal form. Hence, I thought I will implement Random forest on misc. I used the following sklearn libraries StandardScaler, RandomForestClassifier, accuracy_score.
controlled the depth hyper-parameter to 7 using parameter max_depth (did few trials to find the best)
n_estimators parameter lets me set the number of trees it creates. Then uses the output of all the trees to predict the final verdict.

Filename: Random_forest_misc_final.ipynb
Dataset: misc

Accuracy on Train: 79.35 %
Accuracy on Test: 79.64%
Accuracy on Eval: 80.45% [on Kaggle random_forest_final.csv]

6. Future Improvement Ideas

- As I have mentioned in most of my implementations, I wanted to extract important columns from Tfidf and bow dataset which I was not able to achieve. I would like to explore SVD (singular vector decomposition), I have read it gives us the columns which hold the most information.
- And I would like to implement binning correctly on ID3 and try SVM over trees from HW6.
- Also, would try Neural Nets.