CS 6350: Project Final Report

Name: Tejas Ramesh Pawar — u1452462

Kaggle username: *tejasrameshpawar*

Overview

In this report, I present the findings and analysis of a machine learning project undertaken to explore the feasibility of predicting trial outcomes based on transcribed dialogue from trials held at the Old Bailey, the historic Central Criminal Court of England and Wales.

The project expected me to make 6 different submissions and work on minimum 4 different algorithms to classify.

I was provided three distinct data representations for this project. I was allowed to use these datasets just as they were or to preprocess them to my specifications. Additionally, I have used several datasets to get the best results by mixing and matching them. I used this in a few different entries. In every submission, I preprocessed the data as well. I read through the datasets and changed all of the 0s to -1s. Because we cannot merge two datasets with the same column headings, every time I had to merge two datasets, I had to rename the columns in each dataset.

Finding appropriate classifiers to use came next once the data was preprocessed. I made the decision to put the following classifiers into practice: Perceptron, Decaying Perceptron, Support Vector Machine (SVM), Logistic Regression, and Ensemble of SVM and Perceptron and AdaBoost(using libraries). These classifiers produced predictions with varying accuracies because they were trained on various combinations of training datasets. I tried experimenting with different values for the seed and epoch counts when training the models of the aforementioned classifiers. I also experimented with other hyperparameters, such as learning rate and tradeoff.

Important Ideas Explored

The following ideas were explored while working on this project:

- Data/text Preprocessing Techniques: Preprocessing played a crucial role in this project to accurately create classifiers to better classify the data. These techniques came into use from the second project milestone and huge improvements the classifier was seen after incorporating them. Merging the dataset, Tokenizing the examples, filtering *Nan* or 0 values, converting labels from 0s to -1s and grouping a subset of examples into one in order to better represent the data for the classifier to learn were some techniques used throughout.

- Feature Representation: Various feature representation techniques were investigated to capture meaningful information from the dataset. The miscellaneous dataset even though small was helpful in enriching the data when considered with other features.

- Supervised Learning Algorithms: Different supervised learning algorithms were explored to build predictive models based on the extracted features and the preprocessed data. This helped select the appropriate algorithms for this project. Algorithms like Logisitic Regression, Support Vector Machine, Ensemble were worked on.
- Hyperparameter Tuning and Cross-Validation: Optimization of model hyperparameters was explored to enhance classifier performance. Cross validation was one of the techniques used in choosing the correct hyperparamter for a classifier. Throughout this project the data was split into a 5-fold validation and training data. Also, randomly searching and runnning a model also was also an option chosen for this task.
- Ensemble Learning: The concept of ensemble learning was investigated by combining multiple base classifiers to improve prediction accuracy. Techniques like AdaBoost and majority voting were explored to leverage the same.

Ideas used from the class

The following ideas from the class were used while working on this project:

- Supervised Learning Algorithms: A variety of supervised learning algorithms taught to us in the class were employed throughout this project. At the first milestone, the decision tree(ID3)(Filename: *tree.py*) algorithm was used on the dataset, but since the dataset would need binifying the examples(as taught in class), a better algorithm(perceptron) was used. It was helpful to know the merits and demirts of a particular algorithm when used on a dataset and thus it made the work easier. The SVM and Logistic Regression objective functions taught in class made it easy to implement them from scratch and so was the ensemble technique.
- Cross-Validation and Hyperparameter tuning: Cross-validation was implemented by splitting the training dataset into 2 parts (of 80% and 20%) and used the first part for training and the second part as a validation dataset. This was done for every hyperparameter(or combination) to identify the best one, and then used this hyperparamter(s) to train the model on the entire dataset.
- Regularization: Regularization and the loss concepts taught in the class helped to choose the best value of epochs to be set on the classifier to train. This also helped in a sense that it avoided overfitting the data and thus providiing a good fit for the training set.

What I learned

Working on this project helped combining theoretical knowledge from class with practical application in a real-world scenario. The project provided with hands-on experience on the Machine Learning practices and algorithm implementation from scratch.

The project helped me get comfortable with concepts like cross validation and hyperparameters, along with helping me realize the importance of these techniques in improving the performance of the models.

Exploring a variety of supervised learning algorithms, such as logistic regression, decision trees, perceptron, and support vector machines (SVM), provided insights into their strengths, weaknesses, and suitability for different classification tasks.

I also learned the importance of data processing and the techniquest involved.

Summary

As mentioned in the first milestone, I started with using the *tfidf*, *bag-of-words* and *glove* dataset separately using the decision tree classifier. Later, I moved on to using the Perceptron algorithm on the same datasets separately. With this algorithm, I made the first Kaggle submission with the *tfidf* dataset giving the highest accuracy.

Later, I used the data preprocessing techniques and cleaned up the data. I made use of the *miscellaneous* data and cleaned it as follows - I replaced the defendant gender column with male, f emale and indeterminate columns, and gave the values as 0, 1 accordingly. Similarly, I did the same with victim genders. I did the same with offence category feature, where I numbered each of the offenses from 1 to n(number of unique offenses). I appended this dataset with all three datasets and ran the perceptron again which made my second kaggle submission. I also used the Support Vector Machine classifier on the same and made the third kaggle submission.

Finally, I implemented logistic regression and made a submission with it. Since I had the perceptron and SVM coded already, I used ensemble learning using the majority voting along with these two classifiers and made the fifth submission. At the end, I used the *sklearn* library to implement the AdaBoost algorithm with the base classifier as decision stumps and made the sixth and final submission.

Results and Discussion

Here is the breakdown of all submissions and algorithms used:

- Simple Perceptron(without preprocessing and merging datasets)

  First algorithm I tested on all the three data sets. Of which, *tfidf* gave the highest accuracy.

  I ran the cross validation on 5 fold to find the best hyperparameter. The best hyperparameter I found was 0.1.

  After running the algorithm on all the three datasets, I decided to submit the file with the the highest test accuracy on kaggle.

	Accuracy on training set	**Accuracy on testing set**
tfidf	90.82%	70.00%
bag-of-words	88.64%	69.24%
glove	63.15%	64.35%

  Number of epochs used: 100

  Kaggle Score: **0.68**

  Filename: *perceptron.py*

NOTE: From here onwards, I will be using the preprocessed data along with the *miscellaneous* data merged.

- Simple Perceptron(with preprocessing)

  After preprocessing the *miscellaneous* dataset and merging it with each of the three datasets, the same perceptron algorithm was run.

  Similar to the above,

	Accuracy on training set	**Accuracy on testing set**
tfidf + misc	84.07%	83.02%
bag-of-words + misc	91.70%	80.35%
glove + misc	74.80%	72.17%

  Number of epochs used: 100

  Kaggle score: **0.81**

  Filename: *simpleperceptron.py*

  I also ran all these three datasets with slight modification in my perceptron code. I coded up the 'Decay' perceptron(Filename: *decayperceptron.py*) and ran the same. I did not find any significant improvement over the simple perceptron, thus, chose to submit the predicition of simple pereceptron on kaggle.

  I also managed to merge the *tfidf*, *misc* and the *bag-of-words* datasets and tried running the above code. Not only did it take a lot of time to run, but also it was not fruitful. I did not see any improvement by employing this step.

- Support Vector Machines

  I used the Stochastic Gradient Descent version of the algorithm. I used 5 fold cross validation to choose the best combination of hyperparameters. I ended up choosing γ as 0.01 and the tradeoff value(C) as 10.

  I ran the experiment on the below mentioned dataset, and chose the best of them to submit to kaggle.

	Accuracy on training set	**Accuracy on testing set**
tfidf + misc	78.39%	78.93%
bag-of-words + misc	78.20%	78.71%
glove + misc	74.29%	74.04%

  Number of epochs used: 15

  Kaggle score: **0.80**

  Filename: *svm.py*

- Logisitic Regression

  Similarly, I used Stochastic Gradient Descent version for this algorithm. I used the same method of cross validation to choose the best combination of hyperparameters. I ended up choosing the learning rate γ as 0.1 and the tradeoff(σ^2) as

10000.

	Accuracy on training set	**Accuracy on testing set**
tfidf + misc	80.67%	80.88%
bag-of-words + misc	86.82%	77.42%
glove + misc	76.76%	74.93%

Number of epochs used: 15

Kaggle score: **0.78**

Filename: *logisticregression.py*

- Ensemble with majority voting(SVM + perceptron)

  Here I combined the SVM and perceptron algorithm and used Ensemble learning to implment this new algorithm. I used perceptron as the base over SVM. I noticed that with this method, I got the highest accuracy with the *bag-of-words* dataset, thus I ended up submitting the same on kaggle as my fifth submission.

	Accuracy on training set	**Accuracy on testing set**
tfidf + misc	76.93%	77.20%
bag-of-words + misc	79.34%	79.42%
glove + misc	74.24%	74.17%

  Kaggle score: **0.76**

  Filename: *ensemble_svm_perceptron.py*

- AdaBoost (using *sklearn*)

  For my sixth and final submission, I used the python library *sklearn*. I imported the *AdaBoostClassifier* for AdaBoost and as the base estimator I used decision stumps which was implemented by importing *DecisionTreeClassifier*.
  The following parameters were fed to the AdaBoost classifier - *n_estimators* = 50, *random_state* = 42.
  I had to clean up the data here as well and ignore the Nan values.

	Accuracy on training set	**Accuracy on testing set**
tfidf + misc	83.10%	84.48%
bag-of-words + misc	82.58%	83.33%
glove + misc	79.16%	78.71%

  I ended up submitting the results with the highest test accuracy.

  Kaggle score: **0.83**

  Filename: *adaboost.py*

Continuing the project in future

- Since I did not succeed implementing ID3, I will try to implement binning correctly on it and try the SVM over trees algorithm that I implemented in HW6.
- I would try Neural Networks as well.
- I would love to try visualization techniques that I learnt from the other class I am taking this semester(visualization for scientific data) on this dataset and gain deeper insights over time and epochs of each of the classifiers over the experiment.
- Use the High Performance Computing facility to train the dataset, after merging and matching multiple combinations of each dataset.
- I would deepen the exploration of ensemble learning methods by implementing sophisticated and better ensemble strategies.