LfD Assignment1 - Excercise 1.5

Chen, Yu-Wen
S3619362
y.w.chen.1@student.rug.nl

Abstract

The purpose of this experiment is to estimate the given model performances of the binary classification and multi-classes classification. We use different modules in scikit-learn to train and test data, and eventually to evaluate the results. We get 0.782 of the overall accuracy score in the binary classification. The result could be improved with cross-validation.

1 Introduction

We use modules in scikit-learn to split data into training and testing set. And we train these datas and estimate how well it is likely to perfom on out-of-sample data through evaluation functions such as precision, recall, f-score, and also with confusion matrix. And the distinct values of prior and posterior probabilities also shows how important is the additional information taken into account.

2 Data

The corpus file consists of reviews with two meaningful tags which indicate the topics and sentiments of each review. The raw data is preprocessed with a function called `read_corpus`. Thus, it is transformed into two lists - documents and labels, which are the given values of variable X and Y. Each of them has 6000 elements in it. Then it is split into training and testing set, which is 75 % and 25 % of the instances repectively. To avoid overfitting, it is essential to split the data into training and testing set to calculate the testing accuracy which is a better estimate of out-of-sample performance.

3 Method/Approach

I do not use cross-validation in this assignment since we can only get resulting scores making it difficult to inspect the results using the confusion matrix. For Exercise 1.2, with changing the boolean value of variable `use_sentiment`, I can use different types of classifications, either binary or multi-classes, and evaluate them. When the boolean value of `use_sentiment` is `True`, we get evaluating result of binary classification. And if it is `False`, the result is of multi-classes classification. For Exercise 1.3, I import the function `confusion_report` from the module `sklearn.metrics`. It computed the precision, recall, f-score for each class and their average values. For the probabilities, we assume the prior probabilities of each class is in uniform distribution, that is it is one's belif about this quantity before any evidence is taken into account. On the other hand, posterior probabilities is what we get here with the properties of Naive Bayes. I use the method called `predict_proba(X)` to estimate the posterior probability of the instances for each class. In the binary classification case, the prior probabilities of the sample for each class is 50%. As for the posterior probabilities, we can see from the final results are different from the prior. It is because now we have taken the relevant evidence into account.

4 Results

From the classification report of the binary classification, the classifier obtained 0.71 precision and 0.93 recall in the `negative` class. This means 93% of the negative reviews are classified correctly while only 71% of the instances that are classified as `negative` are really negative reviews.

And according to the confusion matrix, there are 48 negative instances are incorrectly labeled as `positive` whilst 683 negative instances are correctly labeled as `negative`. We can tell it tends

to predict things as negative no matter what kind of instances it faces. In my opinion, if this classifier will be fed with more negative reviews, it would be better at labelling negative reviews.

For the `positive` label, it has a percision of 0.91, which means 91% of the instances that are classified as `positive` are really positive reviews. But with only 0.64 of recall, we can tell that it suffers when facing a positive reviews. As we can see from confusion matrix, there are 279 positive reviews are incorrectly classifiec as nagative, while there are 490 instances are correctly-labeled. I think if we feed this classifier more positive reviews(instnaces), it will be more precise on recognizing positive reviews.

5 Discussion/Conclusion

Splitting the dataset into training and testing set is essential to supervised learning. And we should also avoid overfitting which is testing on its own train data because it might still perform poorly at unseen data.

Furthermore, I would like to use cross-validation to reduce the variance associated with only one train/test split, that is `StratifiedKFold` function from `sklearn.model_selection`. The folds will preserve the percentage of samples for each class in both training and testing set. In this way, we can have more accurate estimate of out-of-sample accuracy.

And the features of my task are `negative` and `positive`. This two classes classification could be used by setting the boolean value of `use_sentiment` as `True`.

Also, the confusion matrix provides us useful information about what type of mistakes the classifier makes, and thus we could fix on it.