Machine Learning Online

IMDB Movie Rating Prediction | Naive Bayes

In this problem, we will use the Na¨ıve Bayes algorithm for text classification. The dataset for this problem is a subset of the IMDB movie review dataset (look at the IMDB movie review dataset here - http://ai.stanford.edu/~amaas/data/sentiment/)). Given a movie review, task is to predict the rating given by the reviewer. Read the IMDB website for more details about the dataset. You have been provided with separate training and test files containing 25,000 reviews (samples) each. Data is available at the Github Repository - http://cb.lk/ml18. A review comes from one of the eight categories (class label). Here, class label represents rating given by the user along with the review. You are provided four files i) Train text ii) Train labels iii) Test text iv) Test labels. Text files contain one review in each line and label files contain the corresponding rating.

(a) Implement the **Naive Bayes algorithm** to classify each of the articles into one of the given categories. Report the accuracy over the training as well as the test set. In the remaining parts below, we will only worry about test accuracy.

Make sure to use the **Laplace smoothing** for Na¨ıve Bayes (as discussed) to avoid any zero probabilities. Use c = 1.
You should implement your algorithm using logarithms to avoid underflow issues.
You should implement Naive Bayes from the first principles and not use any existing Python Libraries. Use the Multinomial Event Model as discussed in the videos, write the code from scratch without using the steps given the in PDF.

(b) **Accuracy** What is the test set accuracy that you would obtain by randomly guessing one of the categories as the target class for each of the articles (random prediction). What accuracy would you obtain if you simply predicted the class which occurs most of the times in the training data (majority prediction)? How much improvement does your algorithm give over the random/majority baseline?

(c) **Confusion matrix**. Draw the confusion matrix for your results in the part (a) above (for the test data only). Which category has the highest value of the diagonal entry? What does that mean? What other observations can you draw from the confusion matrix? Include the confusion matrix in your submission and explain your observations.

(d) **Data Cleaning** The dataset provided to is in the raw format i.e., it has all the words appearing in the original set of articles. This includes words such as ‘of’, ‘the’, ‘and’ etc. (called stopwords). Presumably, these words may not be relevant for classification. In fact, their presence can sometimes hurt the performance of the classifier by introducing noise in the data.

Similarly, the raw data treats different forms of the same word separately, e.g., ‘eating’ and ‘eat’ would be treated as separate words. Merging such variations into a single word is called stemming. • Read about stopword removal and stemming (for text classification) from videos shared

- Write one script to perform stemming and remove the stopwords in the training as well as the test data. Learn a new model on the transformed data. Again, report the accuracy.
- How does your accuracy change over test set? Comment on your observations.

(e) **Feature engineering** is an essential component of Machine Learning. It refers to the process of manipulating existing features/constructing new features in order to help improve the overall accuracy on the prediction task. For example, instead of using each word as a feature, you may treat bi-grams (two consecutive words) as a feature. Come up with at least two alternative features and learn a new model based on those features. Add them on top of your model obtained in part (d) above. Compare with the test set accuracy that you obtained in parts (a) and parts (d). Which features help you improve the overall accuracy? Comment on your observations.