CptS 475/575: Data Science, Fall 2024

Assignment 5 – Part 2: Classification

Release Date: Friday, Oct. 25, 2024 **Due Date:** Friday, Nov. 1, 2024 (11:59 pm)

General instruction: This is the second part of Assignment 5. It focuses on classification and requires you to take a set of articles from an online newspaper and classify them based on which category they belong to.

Your solution will be submitted as a PDF/HTML file, which must **include your full, functional code and relevant results as stated in each part**. You are encouraged to use R Markdown or Quarto to prepare your file if you work with R. You are free to use Python to solve the problems; you can use Jupyter notebook to prepare your file in that case.

This problem carries a total of 40 points. You will use a Naïve Bayes model and a Logistic Regression model to classify newspaper articles by their category. You are provided a set of news articles collected from BBC News (http://mlg.ucd.ie/datasets/bbc.html). The articles have been pre-processed and cleared of any major confounding factors such as HTML tags, but it is up to you to check for other problems and to prepare them for classification. The dataset can be found on the Modules page (*bbc.csv*) under Datasets for Assignments. The dataset consists of the text of news articles, and the category each article belongs to. There are a total of 5 categories – business, entertainment, politics, sport, and tech. Prepare the dataset for classification as suggested in question 1 below and then perform the classification tasks in question 2.

1. (20 points) Tokenization
   In order to classify the text effectively, you will need to split your text into tokens. It is common practice when doing this to reduce your words to their stems so that conjugations produce less noise in your data. For example, the words "speak", "spoke", and "speaking" are all likely to denote a similar context, and so a stemmed tokenization will merge all of them into a single stem. R has several libraries for tokenization, stemming, and text mining. Examples of such libraries that you may want to use as a starting point are tokenizers, SnowballC, and tm, respectively. Alternatively, some of you may want to consider using quanteda, which will handle these functionalities along with others needed in building your model in the next step. Similarly, Python has libraries such as sklearn and nltk for processing text.

   You will need to produce a document-term matrix from your stemmed tokenized data. This will create a large feature set (to be reduced in the following step) where each word represents a feature, and each article is represented by the number of occurrences of each word.

Before representing the feature set in a non-compact storage format (such as a plain matrix), you will want to remove any word which appears in too few documents. For this assignment, you will remove 15% of the words corresponding to the least frequent words in the document i.e., only 85% of the terms should be kept.

To demonstrate your completion of this part, print the feature vector of the words that are appear 4 or more times in the 2205th article in the dataset. Your output should show the words and the number of occurrences in the article.

2. (20 points) Classification

   For this part of the assignment, you will build and test a Multinomial Naïve Bayes classifier and a Multinomial Logistic Regression classifier to handle the multiple classes in the dataset.

   First, reduce the feature set using a feature selection method, such as removing highly correlated features. The caret package in R or similar libraries in Python like sklearn can be used to help reduce the number of features and improve model performance. You may wish to try several different feature selection methods to see which produces the best results.

   Next, split your data into a training set and a test set. Your training set should comprise approximately 80% of your articles and your test set the remaining 20%. In splitting your data into training and test sets, ensure that the five categories are nearly equally represented in both sets. Experiment with other split percentages (than 80-20) to ensure a balanced representation of the five categories, and use the split that gives you the best result for the required classifiers below.

   Next, build a Multinomial Naïve Bayes classifier using your training data. In R, you can use the multinomial_naive_bayes() function from the naivebayes package, and in Python, the MultinomialNB class from sklearn can be used. After building the model, use it to predict the categories of your test data.

   Once you have produced a model that generates the best predictions you can get, print a confusion matrix of the results to demonstrate your completion of this task. For each class, give scores for precision (TruePositives / TruePositives+FalsePositives) and recall (TruePositives / TruePositives+FalseNegatives).

   Finally, build a Multinomial Logistic Regression classifier using the same training and test sets and compare the results using a confusion matrix, as well as precision and recall scores for each class, with those from the Multinomial Naïve Bayes classifier.