Name: Fynnba Biney

Course Title: Natural language Processing

Name: Dennis A. Owusu

Report

Introduction

Naïve Bayes classification is a method based on Bayes' theorem that computes the probability of the given feature being associated with a class. It assumes conditional independence for every feature, which means that the algorithm assumes all the features are independent of each one.

Logistic regression classification is a linear method that learns the probability of a given sample belonging to a certain class. Logistic regression tries to determine the most efficient decision boundary that best separates the classes which is like determining the line of best fit.

Both Naive Bayes and Logistic regression are used to determine if a sample belongs to a certain class, for example, if an e-mail is spam or not or if a review is positive or negative. Naive Bayes is a generative model and Logistic regression is a discriminative.

Libraries Used

I used Sklearn and NLTK in cresting both my Naive Bayes and Logistic regression. To clean the data and process it, I cleaned and vectorised it with TFIDF and CountVectorise which were from Sklean. TFIDF or tf–idf is short for term frequency–inverse document frequency. CountVectorise basically converts a collection of text documents to a matrix of token counts.

CountVectorizer can lowercase letters, disregard punctuation and stopwords, but it can't lemmatize or stem words. It has the attributes Min_df and Max_df. Min_df allows you to set a minimum threshold to determine if a word would make it into the vocabulary. Min_df=0.8 requires that a term appear in 80% of the documents for it to be considered part of the vocabulary. Max_df, it ignores terms that have a document frequency strictly higher than the given threshold.

TFIDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. Using TFIDF instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus.

The formula that is used to compute the TFIDF of term t is;

TFIDF (d, t) = TF(t) * TFIDF (d, t)

The TFIDF is computed as TFIDF (d, t) = log [n / df(d, t)] + 1 (if smooth_idf=False), where n is the total number of documents and df(d, t) is the document frequency; the document frequency is the number of documents d that contain term t. Terms that occur in all documents in a training set, will not be entirely ignored. A variation of TFIDF is TFIDF Transformer which also cleans the data and removes stop words.

How Classifiers were Trained and Evaluated

Sklean and NLTK have in-built functions to calculate the precision, recall, F-1 Measure and Accuracy. I used these to evaluate my libraries. I first used 80 percent of the data provided to train the models and tested on 20% of the data (the remaining data after training). Then before submitting, I used all the data to train the model.

Results

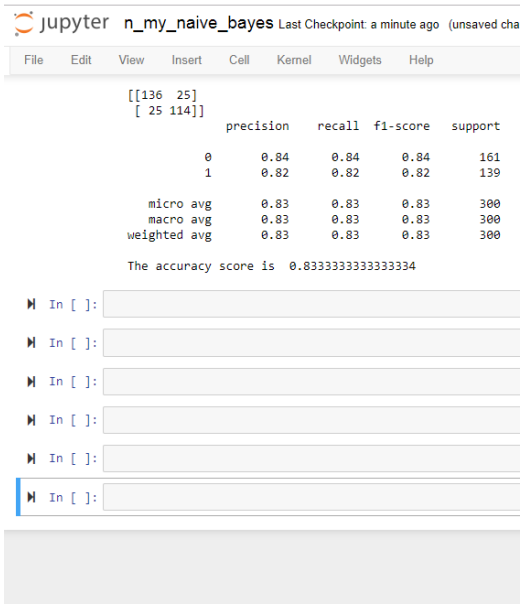

Fig 1.

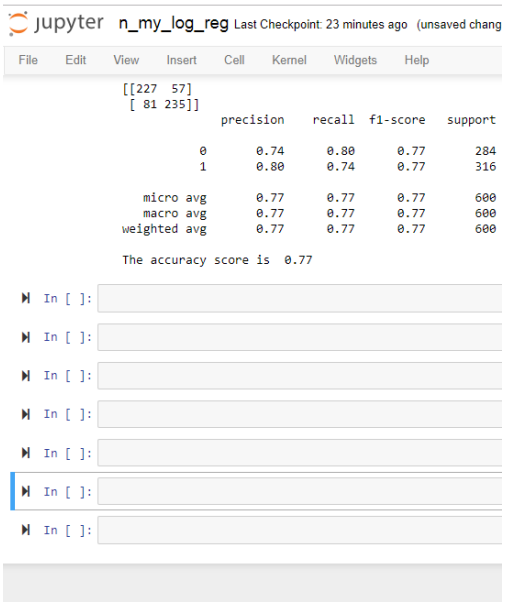

Fig 2.

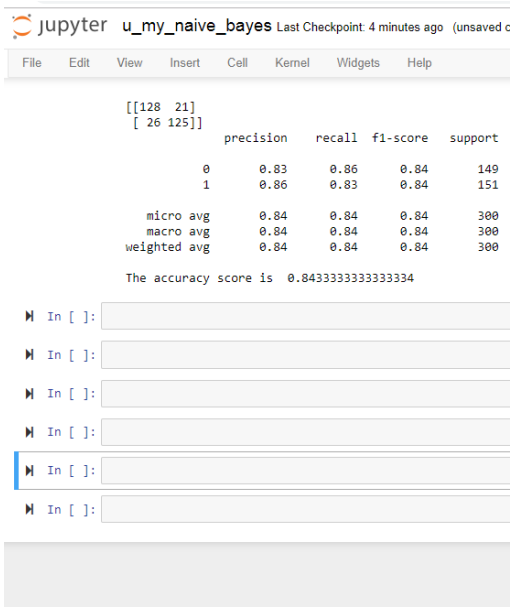

Fig 3.

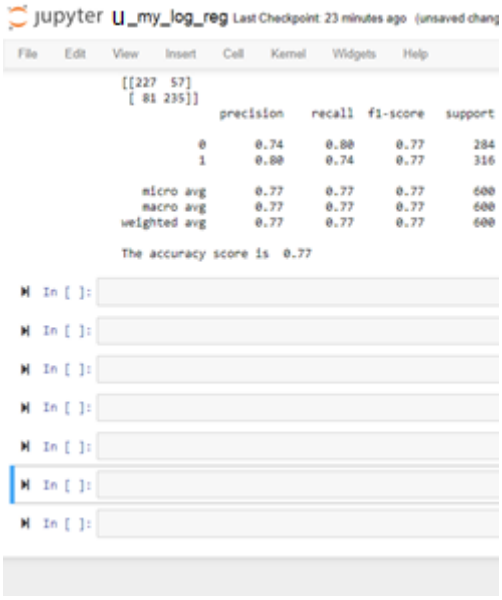

Fig 4.

The result for both the normalized and un-normalized classifiers were very close (there was only an average difference of 0.05 between the normalized and un- normalized). This is because of how the data was handled prior to training the model. For the un- normalized models, the data included stop words and if a word appeared in 20 % of the documents, it was included in the vocabulary.