Customer Reviews using Logistic Regression

Data Observations

Our data is unbalanced where there are approx 45% negative reviews and 55% positive reviews. Ideally in data mining an unbalanced training data might result in an inaccurate training of the model, hence usually depending on the use case or data we should clean the data first by balancing it. In our given data we were not required to clean the data as in our case it did not affect the accuracy of the score.

Approach

Our code is a machine learning implementation for sentiment analysis where we determine the sentiment of a given text, i.e. whether it is positive, negative or neutral. The code uses Python's libraries, Pandas, Numpy, and Scikit-learn along with the Natural Language Toolkit (nltk) to build and evaluate a logistic regression model for sentiment analysis.

The code first loads the training data into a pandas dataframe and performs some data cleaning operations to fill any null values present. The sentences and their corresponding labels are then separated into two separate lists. A preprocessor function is defined which is used to lowercase the text, remove any punctuation and digits from the text. The text is also tokenized using the word_tokenize function from the nltk library.

The code then applies a count vectorizer to form a bag of words from the preprocessed text and also converts them into lowercase. The data is then split into training and test sets using the train_test_split function from the scikit-learn library.

The code then initializes the logistic regression model and fits the model to the training data. The model's accuracy is then evaluated using the accuracy_score function from the scikit-learn library.

The code then applies the same model on the test data to make predictions on the sentiment of the sentences present in the test data. The predictions are then written to a text file.

In summary, this code uses a logistic regression model for sentiment analysis on the review data. The code uses the bag of words representation and the count vectorizer approach to preprocess the text data, followed by the logistic regression model to make predictions on the sentiment of the text data.

Iterations

- An iteration of code on this data set was applying the TDF-IDF Vectorizer from the scikit-learn library, but that did not make much of a difference in terms of accuracy.
- My initial approach involved a balanced dataset which is an ideal input for any training data. Hence making the data set unbalanced but it did not affect the accuracy hence it was not considered in the final implementation.
- An iteration involving stemming was also used where the words were broken down into their basic terms. But that resulted in loss of data as stated in the documentation. This reduced the accuracy from 0.85 to 0.84 hence was discarded. Implementing stemming of words also increased the run time of the code exponentially which is not recommended for a real time implementation of Prediction algorithms.
- Stop words are also a type of data cleaning techniques which involves removal of stop words, which are not crucial in identifying the outcome of the review. On implementing stop words in the initial iteration of the code did not affect the accuracy, hence it was discarded.
- On removal of stop words the accuracy went from 0.85 to 0.86 hence the final implementation does not have any stop words removal implementation in place.

Accuracy

We calculate the accuracy of the logistic regression model by comparing the actual labels (y_test) with the predicted labels (y_pred) using the accuracy_score function from scikit-learn's metrics module. The accuracy_score function computes the mean accuracy rate across all classes, i.e., the number of correct predictions divided by the total number of predictions. The accuracy score is stored in the variable acc.

Miner Score

Username :- deeppp15
Score:- 0.86
Rank:- 207