Big Data - Fall 2019

Homework Assignment #3
Transparency and Explanations
Due: 11:55 AM (5 minute to **noon**), November 27, 2019
NO LATE SUBMISSIONS WILL BE ACCEPTED

Goal

In this assignment, you will work with the prediction output of a particular machine learning model – a classifier – and utilize the open-source LIME library to explain its predictions. You will then perform feature selection based on the set of generated textual explanations in order to improve the classifier's performance. To have a concrete understanding of this library, take a look at the paper describing LIME.

Description

Unlike in previous assignments where you worked with tabular data, now you will explore text data to validate sentiment analysis. More specifically, we will use the 20 newsgroups text corpus, and focus only on two classes, Atheism and Christianity. The provided ML model takes in this text corpus and performs binary classification to predict for a given document if it has Atheistic or Christian sentiment.

Your job is to work with this prediction output of this model. Note that this model is built in Spark using its machine learning libraries ML and MLlib, and so you will be working with Spark data structures.

After completing this assignment, you will:

1. Understand how to use LIME to generate locally interpretable explanations of classification decisions on a text corpus.
2. Learn how to use local explanations for feature selection, ultimately improving classification accuracy on a text corpus.

Note: LIME does not provide causal explanations, so be careful to not think of these associations as causal links!

Specifications

You are required to use Python 3.6.5, Spark 2.4.0, and the libraries imported in the provided code. After loading the modules, use the following command to run your code:

```
spark-submit --conf spark.pyspark.python=$PYSPARK_PYTHON netID.py
```

Make sure to change the python file name to your netID.

Code, Report, Results and Submission Instructions

You should submit:

1. netID.py (code)
2. netID_report.txt (report)
3. netID_misclassified_ordered.csv
4. netID_words_weight.csv

Zip these files together into one zip folder and name it with your netID (for example, tbl245.zip)

Submitted results must match the outputs of submitted code – YES, we will run your code.

Grading

This assignment is worth 100 points or 10% of the overall course grade. If this assignment is submitted late, you will receive no credit.

This assignment is to be completed individually. Please consult the course syllabus for a description of our academic honesty policy.

Tasks

Task 1 (50 points): Generating explanations

1. Use the provided code and load the required data. This will fetch the training and test datasets from the 20 newsgroups corpus for two categories, Atheism and Christianity.
    a. Compute and report the number of documents in training and test datasets.
    b. Index the documents in each dataset (training and test) by creating a 0-based index column for each dataset.
        - Name this column "id"
        - Paste the first 5 lines of indexed *test set* into your report (truncate = True)

2. In order to analyze text sentiment, we tokenize the text, weigh each word using the TF-IDF measure ("term frequency-inverse document frequency"), and then use logistic regression for binary classification. These steps are wrapped inside a pipeline, which, when invoked with a training dataset, produces a binary classifier.

    When the resulting classifier is invoked on a document from the test set, it associates with that document a probability distribution, quantifying the likelihood that the document belongs to either class (Atheism or Christianity), and finally assigns the document to the more likely class.

This pipeline is provided for you, **train the model**.

a. Compute and report the F1-score on the test dataset.
b. Report the schema of the model's prediction output. Giving names and data types of columns is sufficient.

In the provided code, we illustrate the use of LIME's explainer to generate an explanation for a document randomly chosen from the test dataset. Refer to this example for questions below.

3. For *test* set documents with ids **0, 275,** and **664**, report in your txt file their:
   (i) categories (ground truth)
   (ii) probabilities over categories computed by the classifier
   (iii) predicted category for each document, and
   (iv) LIME's generated textual explanation, in terms of 6 features.

4. The difference between the imputed probabilities of the two classes for a document, denoted conf, can be a measure of the model's confidence in its prediction for that document.

   conf = |prob(Christian)-prob(Atheism)|

   For the misclassified documents, conf can be used to quantify the magnitude of the error.

   Generate explanations for **all** misclassified documents in the *test* set, sorted by conf in **descending** order, and save this output to **netID_misclassified_ordered.csv** for submission.

   For each document per line, list its *ID, conf, and LIME's textual explanation in 6 features.* Thus, we expect 3 columns in your comma-separated file. An example for outputs of a misclassified document looks like this:

   6, 0.5959862, [('Freemasonry', -0.22918663477083728), ('Page', - 0.10337955399482289), ('equality', 0.0885735890266602 1), ('10th', -0.074338694087527), ('Ministry', 0.06657848659291651), ('Posting', 0.0633698144289709)]

5. Identify **all** words that contributed to the misclassification of some document. Naturally, some words will be implicated for multiple documents. For each word (call it **word_j**), compute,
   a. the number of documents it helped misclassify (call is **count_j**) and
   b. the total weight of that word in all documents it helped misclassify (**weight_j**) (**sum of absolute values** of **weight_j** for each misclassified document). Use absolute values because LIME assigns a positive or a negative sign to **weight_j** depending on the class to which **word_j** is contributing.

   Output all such words to **netID_words_weight.csv**, sorted by count_j in **descending** order. Present one word per line, along with its count and weight. For example, the output may contain the following line:

   Organization, 200, 124.70125

Task 2 (50 points): Feature selection

Focusing on all misclassified documents for which **conf_i >= 0.1**, identify **one strategy for feature selection** that improves classifier accuracy. You are to:

- identify features (words) to remove from the training set
- remove these words from the training set only
- retrain the provided model on this adjusted training set
- report model accuracy (F1-score)

In no way are you allowed to alter the specifications and design of the provided model, for example, changing the model's learning rate, loss function, or number of epochs. In other words, when you retrain the classifier, simply call *pipeline.fit(your_train_set)*.

In your report,

- show and explain, step-by-step, your strategy,
- explain the nature of the improvement. You may, for example, argue that the number of misclassified documents is now lower and that some difficult examples that were misclassified previously are now correctly classified. You may also want to argue that the explanations of the classification decisions are now more reasonable.
- List the model accuracy (F1-score) before and after feature selection.
- Additionally, you should output information about **all documents that are classified correctly after feature selection and that were misclassified earlier** (under Task 1). Use the same format to output document information as in Task 1. Unlike in Task 1, there is no need to output the list of words (features) that contribute to misclassification under this task as a separate file.

Have a Wonderful Thanksgiving!