Assignment #3

Large-Scale Computing for the Social Sciences
MACS 30123, Autumn 2020
Due Wednesday, December 2 at 11:59pm (CST)

In this assignment, you will perform an exploratory data analysis of the Amazon Customer Reviews Dataset using Dask and also attempt to improve the PySpark machine learning model we trained on the same dataset in Lab 6. **You should continue to work with the subset of the customer reviews we explored in Lab 6: book reviews.** When you are finished with the assignment, you will submit a README.md file with your answers to the questions below, along with the code you used to produce your answers. You should commit these items to your Assignment 3 repository on GitHub and submit a link to your repository on Canvas for Assignment 3.

1. **Exploratory Data Analysis** (10 points). Use Dask on either an EMR cluster or the Midway RCC to explore the customer reviews data in more detail. For instance, how is the data in different columns distributed? How is the data in one column related to that in another column? Are there any interesting relationships and patterns that may be useful for predicting whether a review is a "good" or a "bad" review? Over the course of your exploratory data analysis, you should produce at least 5 visualizations (worth 1 point each), along with a 2 sentence description (minimum) describing the significance of each of your visualizations (1 point for each accompanying description).

2. **Balancing the Data** (2 Points). Recall from Lab 6 that the dataset is not balanced. There are considerably more "good" star ratings than there are "bad" star ratings and this makes it difficult to train a model that effectively predicts "bad" star ratings. Suppose we want to do a better job of predicting if a star rating will be "bad."

   One strategy for dealing with such data imbalances is to "downsample" the number of "good" star ratings (i.e. randomly sample a subset of reviews with good star ratings that is equal in size to the number of those with bad star ratings) in the training data. Use PySpark in an EMR Notebook to implement a downsampling strategy to balance out the number of reviews with good and bad star ratings in your training data (hint: PySpark's `sampleBy` is your friend). Produce a plot or table of the count of each label ("bad" and "good") in your downsampled dataset in order to demonstrate that your training data is now balanced.

3. **Implementing a Reproducible Machine Learning Pipeline** (6 Points). For this prompt, you will use PySpark in an EMR Notebook to:

   (a) (3 Points) Engineer at least 3 additional features to add to the existing logistic regression model in Lab 6, based on the columns in the Amazon Customer Reviews dataset. You should include at least one categorical feature and one feature derived from text data. Optionally, check out the pyspark.ml documentation if you would like to branch out beyond the feature engineering approaches covered in the DataCamp courses. For each feature, you should provide at least a one sentence description of why you are including the feature in your model.

   (b) (1 Point) Organize all of the transformers (i.e. those you used to engineer your features above) and estimators that operate on your training and test data into a reproducible Machine Learning pipeline that you can feed into a PySpark CrossValidator.

   (c) (2 Points) Describe in words what happens under the hood in Spark when you specify this series of transformations in a pipeline. Is your DataFrame actually processed when you chain together a sequence of transformations in a pipeline? Why or why not? If not, when and how will Spark process the DataFrame according to the specified transformations? How does this compare to Dask's execution model? Your answer should be a minimum of four sentences long.

4. **Finding Optimal Model Parameters** (2 Points). Once you have debugged your code for (3) and are confident that it will run, spin up an EMR cluster with up to 8 m5.xlarge EC2 instances in it (if you haven't already) and identify the optimal `regParam` (use `np.arange(0, .1, .01)` as the range of values in your grid) and `elasticNetParam` (use `[0, 1]` as the possible values in your grid) parameters for your model via a 5-fold cross-validated grid search. You should evaluate your models using AUC (Area under the ROC Curve).

   Report the optimal model's predictive performance on your test data. What does the model do well? What does it do poorly, and how might you improve it further?

   Additional Notes:

   - It is normal to see the "Exception in thread cell_monitor" exception when you perform cross validation in an EMR Notebook. This is a problem with the EMR Spark Job progress bar and not with your cross validation code. Your code will still successfully run if you see this exception.
   - If you have a spotty internet connection, it might be prohibitive to address this question in an EMR notebook. If you are in this position, we would recommend debugging your code locally and then only performing this cross-validation step on a small, randomly sampled subset of your training data. So long as your PySpark code is scalable and runs properly, you will receive full credit for this question if you pursue this option.