Submitted by – Yash Vijaynarayan Gupta

Big Data Management & Analysis (CS6350)

Tweet Processing & Classification using Pipelines

In this part, we will work with a set of Tweets about US airlines and examine their sentiment polarity. More details about the dataset is available at:

https://www.kaggle.com/crowdflower/twitter-airline-sentiment

It is part of the Kaggle competition on Twitter US Airline Sentiment. Our aim is to learn to classify Tweets as either "positive", "neutral", or "negative" by using logistic regression classifier and pipelines for pre-processing and model building.

Create a pipeline with the following steps. We need a pipeline so that we do not have to run these steps individually. Below are the steps of the project:

1. Loading: First step is to load the text file from the path specified. After that, you will need to remove rows where the text field is null.

2. Pre-Processing: Start by creating a pre-processing step with the following stages:

- Stop Word Remover: Remove stop-words from the text column.
  (Using the org.apache.spark.ml.feature.StopWordsRemover class)
- Tokenizer: Transform the text column into words by breaking down the sentence into words.
  (Using the import org.apache.spark.ml.feature.Tokenizer class)
- Term Hashing: Convert words to term-frequency vectors
  (Using the import org.apache.spark.ml.feature.HashingTF class)
- Label Conversion: The label is a string e.g. "Positive", which you need to convert to numeric format
  (Using the import org.apache.spark.ml.feature.StringIndexer class)

Remember to create a pipeline of the above steps and then transform the raw input dataset to a pre-processed dataset.

3. Model Creation - Create a logistic regression classification model. You will have to create a ParameterGridBuilder for parameter tuning and then use the CrossValidator object for finding the best model parameters. More details can be seen here:

https://spark.apache.org/docs/2.2.0/api/scala/index.html

4. Model Testing & Cross Validation: Next, you will need to train and test your model on the given dataset and output classification evaluation metrics, such as accuracy, etc. You can see details of multi-class evaluation metrics at:

https://spark.apache.org/docs/2.2.0/mllib-evaluation-metrics.html

5. Output: Finally, you have to write the output the classification metrics to a file.