Spark Streaming for Machine Learning

Team members:

1. Dheeraj - PES1UG19CS142
2. Kanish - PES1UG19CS207
3. Karthik - PES1UG19CS217
4. Shubham - PES1UG19CS475

Project title Chosen:

Machine Learning with Spark Streaming using Sentiment Dataset

Design details:

In this spark streaming model the basic steps were to convert the incoming text data into a data frame for spark streaming using RDD. Then pre-process the data to convert the text data into numerical data, vectors specifically which can then be fed to a classification algorithm for further processing and learning.

The classification algorithms used were Naive Bayes, SGD classifier, K-means Clustering and Passive Aggressive classifier. A pipeline technique was used to preprocess each incoming batch of data, which is first converted to a dataframe. Then the dataframe goes through a series of steps for the model to learn and then tested on the training data

Implementation details:

Multiple batch sizes were tested starting from 100 to 10000 and the most optimal results were observed at 10000 for all the four models built.

A pipeline approach involving the splitting of data into training and testing data in the ratio of 8:2. Then the series of steps involves extracting the dataframe data from a particular feature and converting it to a numpy array.

Then the numpy array goes through a series of steps from conversion of text data into vectors with intermediate steps involving removal of stop words. The vector is then fed for tf-idf count to count the frequencies of the word and the weight associated with them.
The output of the tf-idf is processed using certain classification algorithms followed by testing on test data.

Reason behind Design:

The incoming data is a Dstream of data that is internally being in the form of JSON. So, first the JSON is loaded and converted to a list of tuples, then the tuples are extracted and converted to a dataframe using the built-in function.

Once the data frame is ready header column names are added and used for preprocessing. Since this is a word classification problem the text data needs to be converted to numerical data for classification algorithms. The text data is converted to numerical data using vectorization techniques where the output is a list of vectors.

Then the term-frequency and inverse document-frequency is calculated to understand the text data. Then the output is fed to a classifying algorithm using a built-in function of partial-fit in sklearn library.

The partial-fit supports incremental learning of data, since the data arrives in batch incremental learning is necessary where the new RDD's data is built upon all the data arriving before and the classifying algorithm’s learning.

Takeaway:

Enormous amount of data is produced each and every hour of the day. This data is noisy and interpretations cannot be drawn from them. It is required to be preprocessed to reduce the noise and extract meaningful and business related information from the data for further learning and building.

Since the data used in certain industries demand real-time processing, spark comes in to convert the budget data into chunks and send in the form of batches. These individual batches are then preprocessed.

Spark is a powerful tool for implementation of major classification and regression models for learning and testing in the real-world. Both incremental and batch wise learning techniques can be used to build models that provide results in real-time