Machine Learning with Spark

Team ID - BD_098_099_130_155

Darshan Madesh - PES1UG19CS130
Ayush Kapasi - PES1UG19CS099
Ayush Godbole - PES1UG19CS098
Sai Manoj - PES1UG19CS155

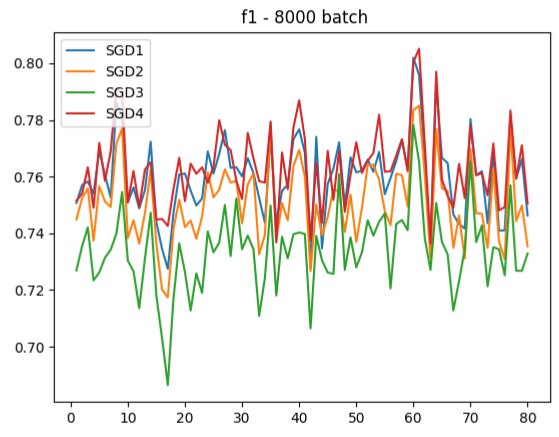
Dataset Chosen -
Sentiment Analysis

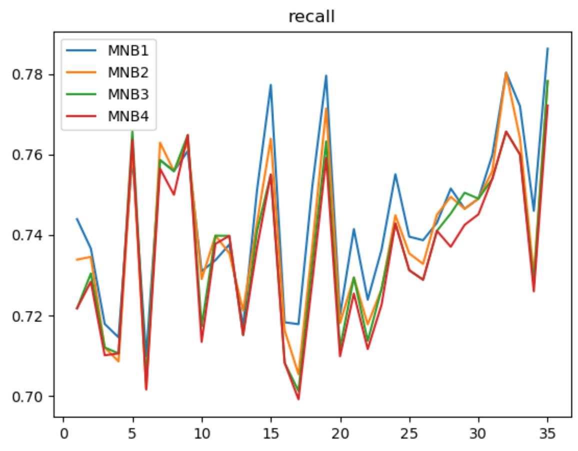
Design Details and Implementation -

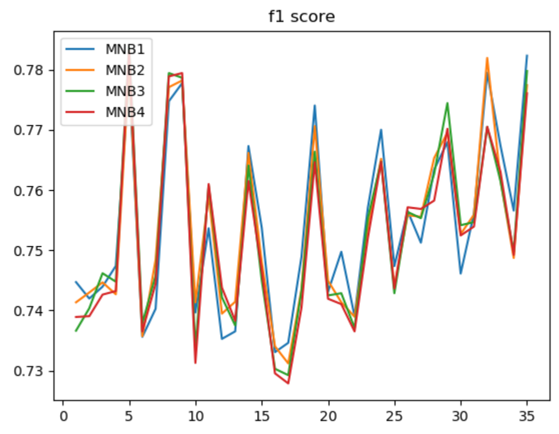
- Receiving data : Using Spark's streaming context, we were able to read the data being sent through the tcp socket connection. Since it was a stream of data, it was streamed as a sequences of RDDs. We now had to perform our processing on each of the RDD that was being streamed one after the other and therefore had to perform incremental learning. The data being streamed in json format, needed to be parsed to and mapped to a usable RDD.
- Data cleaning and preprocessing : Our dataset had sentiments as [4,0] which was encoded to [1,0] for uniformity. Once the data was received properly, it was cleaned and preprocessed using custom functions written to deal with emoji, urls, non-ascii characters and punctuations. The text was then lemmatized using the word-net lemmatises of nltk package. To apply the method on a spark data frame, we converted the custom functions to a udf (user defined functions).
- Feature Extraction : Once we had the cleaned and preprocessed text, we had to now convert text to numerical features (Feature extraction). To perform this, we first collected the text and sentiment into numpy arrays, then we used hashing vectoriser to perform the feature extraction. We hashed the text have 2**20 features to ensure no collision during hashing.
- Model training : Now our data was ready to be fitted with a model, since we were performing incremental learning we used partial fit to learn and used the following classifiers -
- SGD Classifier
- Perceptron

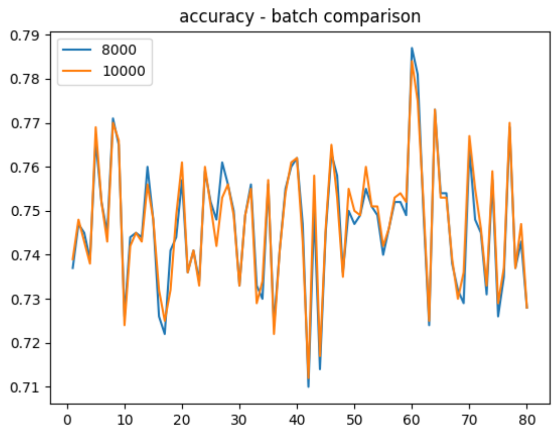
- Multinomial Naive Bayes
- Passive Aggressive Classifier

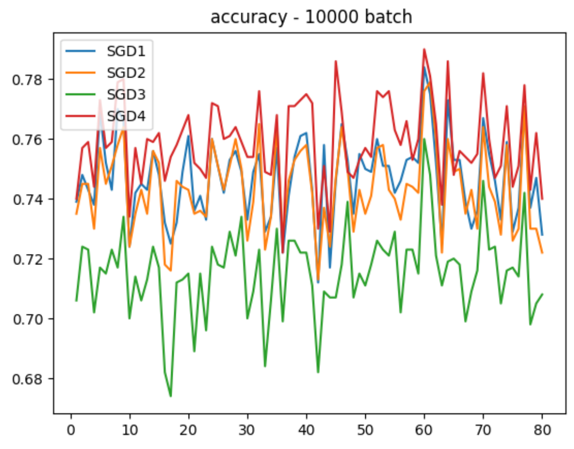
To simulate real-time analytics, we trained multiple models of the same classifier with different parameters to get the a best model from each classifier. Each trained model was pickled, and stored on disk to be used to test on the test data stream. Each model was tested and we obtained various metrics such as recall, precision, f1score, and accuracy. These metrics were stored in a json file and then were plotted to evaluate the models performance. We compared the best model of each classifier to obtain the best overall classifier.

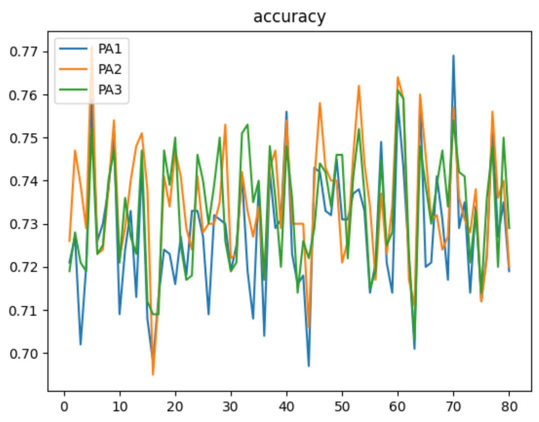


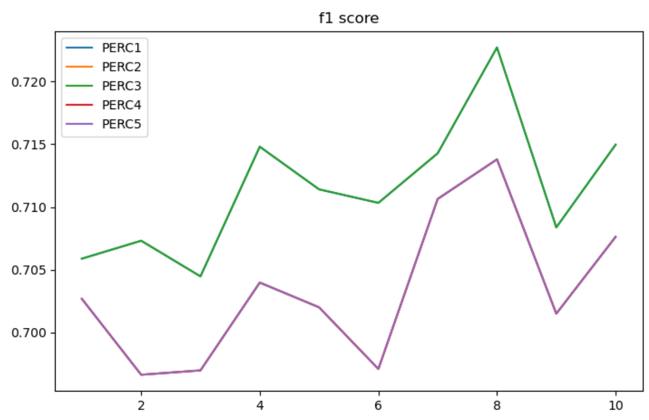












Reason for design choices -
We used batch sizes of 8000&10000, to train the model to check the change in performance. In total we trained around 10 models to ensure that we were able to get the best one in the end. Since incremental learning was deprecated in spark MLlib, we converted cleaned and preprocessed spark data frame into numpy arrays. Which could then be used with scikit-learn's limited supply of methods and models available to perform incremental learning.

Takeaway from the project -
We were able to simulate and perform online/incremental learning, this helped us understand the procedure and methodologies applied in real time analytics. We were also able to familiarise ourselves with spark and spark streaming. We also understood the method to be followed to perform a real time analytics project with constant flow of data.