Big Data - Pyspark ML Report

Team: BD_174_219_240_281

PES1UG19CS281-Mrudhulraj Natarajan

PES1UG19CS240-KUSHAL T M

PES1UG19CS174-Guruprasad N B

PES1UG19CS219-Kirankumar K R

Process:

The first step is to create RDDs' from the streaming data. Based on the batch size RDD are created and sent to the respective model. After, receiving the RDD we convert RDD to pyspark dataframe and implement the respective pre-processing steps which include Regex_tokenization, Lemmatization and followed by vectorization using Word2Vec or HashingTF which will normalize and vectorize the tokenized words based on the size of vocabulary. Finally, we apply partial_fit to the model and get the results.

About the files:

get_data.py: It contains the creation of RDD and for calling the ML model interested by using the "id_inp" flag variable that calls the ML model of interest.

SGDlog_classifier.py: This python file classifier implements a logistic regression ML model which will fit on the data after the pre-processing and lemmatization on each RDD hence an incremental learner. It saves the last weights of the training in "weights/SGDlog.pkl" which will load for further training as well as the best weights 'max_SGDhinge.pkl'. The measured metrics are saved into csv file "SGD_log.csv".

SGDhinge_classifier.py: This python file classifier implements a SVM ML model which will fit on the data after the pre-processing and lemmmatization on each RDD hence an incremental learner. It saves the last weights of the training in "weights/SGDhinge.pkl" which will load for further training as well as the best weights 'max_SGDhinge.pkl'. The measured metrics are saved into csv file "SGD_hinge.csv".

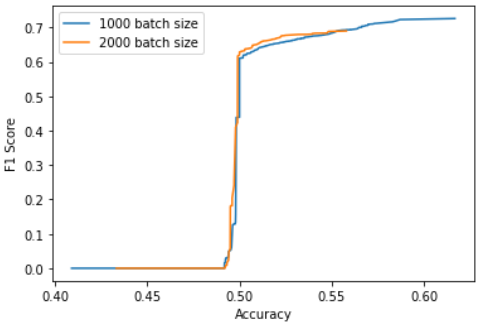
Naïve_multinomial.py: This python file classifier implements a Naïve Bayes ML model which will fit on the data after the pre-processing and lemmmatization on each RDD hence an incremental learner. It saves the last weights of the training in "weights/MNB.pkl" which will load for further training as well as the best weights 'max_MNB.pkl'. The measured metrics are saved into csv file "MNB.csv".

PAC_classifier.py: This python file classifier implements a Passive Aggressive ML model which will fit on the data after the pre-processing and lemmmatization on each RDD hence an incremental learner. It saves the last weights of the training in "weights/PAC.pkl" which will load for further training as well as the best weights 'max_PAC.pkl'. The measured metrics are saved into csv file "PAC.csv".

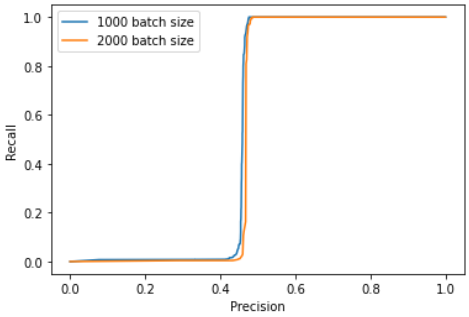
KMeans_classifier.py: This python file classifier implements a K-Means clustering ML model which will fit on the data after the pre-processing and lemmmatization on each RDD hence an incremental learner. It saves the last weights of the training in “weights/KMN.pkl” which will load for further training as well as the best weights ‘max_KMN.pkl’. The measured metrics are saved into csv file “KMN.csv”. It is observed that it is

Example Results:

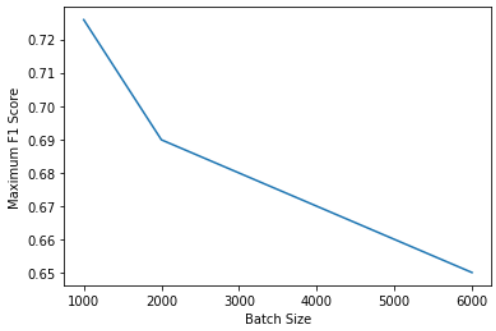
PAC Model:



Accuracy vs F1 Score



Precision vs Recall



Batch Size vs Max F1 Score

Observation:

Accuracy vs F1 Score: We can see a strong correlation between Accuracy and F1 Score. As accuracy increases F1 score also increases

Precision vs F1 Score: We can see a strong correlation between Precision and F1 Score. As precision increases F1 score also increases with ,this is because F1 score and Precision are proportional to TP rate

Batch Size vs Max F1 Score: We can observe that Maximum F1 score decreases with increase in batch size this is because even though training is faster the models doesn’t fit on large dataset

Clustering: On the implementation of clustering we observed highest F1 score to be 0.71 which creates **two** clusters and finds the F1 score based on comparing data belonging to either of the clusters and their true value.