BD PROJECT REPORT
Machine Learning Spark Streaming

Mahendra N (PES1UG19CS285)
Maneesh Sriram (PES1UG19CS257)
Chand Kumar (PES1UG19CS121)
Vandana (PES1UG19CS556)

Project title chosen:

Machine learning with pyspark
First we imported the all the spark context and from pyspark.sql.session we imported sparks ession and from pyspark.streaming, we imported streaming context.So there was a main function where all the required streaming preprocessing and modelling was done. Firstly the pipelining was done so the stages of pipeline were regex tokenizer, stop words removal,string indexing,n-gram,hashing tf. Then there were 4 separate functions named bernoulli naive bayes, linear SGD and perceptron and clustering Which were having the model part. Then we had called main functions for all these RDDs which will do all these for every batch.

Design Details:

So there were 2 files for train and test. The model which we trained using train.py was tested using test.py. Different functions were made for each model and the parameters taken were vectorized form of content and string indexed form of label. . In each function, model was loaded using joblib module and then accuracy was printed in each function.

Surface level implementation:

While streaming, subject email and label was zipped into a dataframe and then for this preprocessing has to be done and in preprocessing first we took the input column as email content which will produce the tokenized output column and then this is taken as input by stop word removal whose output is taken as input for n-gram then output of n gram is taken as input hashing tf and the input from the dataframe which is label is taken as input by string indexer and given as output label index.

Regex tokenizer will tokenize the sentence in list of words , then the stop word removal will take the tokenized input and removes the stop word from it. Then the hashing tf will take the output of n gram and produces the vectorized form of output and string indexer will change the output label as 0 or 1.

Reason behind design decision:

We use pipeline as its a sequence of stages. Each stage will preprocess the data and then next stage will take the input as output of the previous stage. So this would be useful while working with batches then we used try and except while training the model because when we train the model for the first time it will go to except block get trained and this is dumped into .pkl file using joblib. When it is trained for the next batch , it just uses the pickle file and gets trained. This is how the incremental learning was implemented.

Takeaway from the project:

Machine learning model using spark streaming, While dealing the data streams, each stream was preprocessed and incremental model was built.
And it was observed that more batch size gave better accuracy in all the models.

We have used three classification models namely -

i) **Bernoulli NB (sklearn.naive_bayes.BernoulliNB)**

This classifier is suitable for discrete data. BernoulliNB is designed for binary/boolean features. It is a Naive Bayes classifier for multivariate Bernoulli models. The parameters involved are alpha, binarize, fit_prior, class_prior.

ii) **SGD (sklearn.linear_model.SGDClassifier)**

Linear classifiers with SGD (stochastic gradient descent) training. This implementation works with data represented as dense or sparse arrays of floating point values for the features. The model it fits can be controlled with the loss parameter; by default, it fits a linear support vector machine (SVM). The parameters involved are loss, penalty, alpha, l1_ratio, fit_intercept and many others.

iii) **Perceptron (sklearn.linear_model.Perceptron)**

Perceptron() is equivalent to SGDClassifier(loss="perceptron", eta0=1, learning_rate="constant", penalty=None). Some of the parameters involved are eta0, class_weight, warm_start, etc.

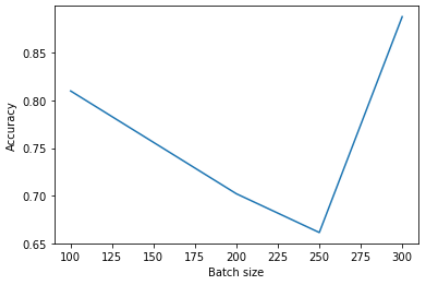
We have built one clustering model namely -

i) **Mini-Batch K-Means clustering (sklearn.cluster.MiniBatchKMeans)**

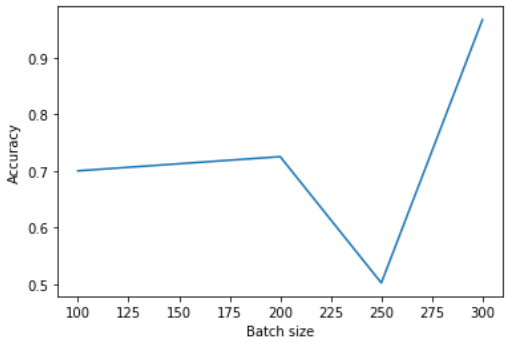
The Mini-batch K-means clustering algorithm is a version of the standard K-means algorithm in machine learning. It uses small, random, fixed-size batches of data to store in memory, and then with each iteration, a random sample of the data is collected and used to update the clusters. The parameters involved are n_clusters, max_iter, tol, batch_size, init_size, etc.

We have used joblib(lightweight pipelining in Python) to load our classification and clustering models.

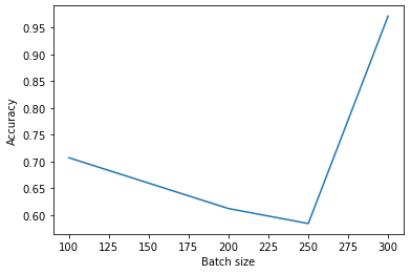
Graph for Bernoulli Naive Bayes:



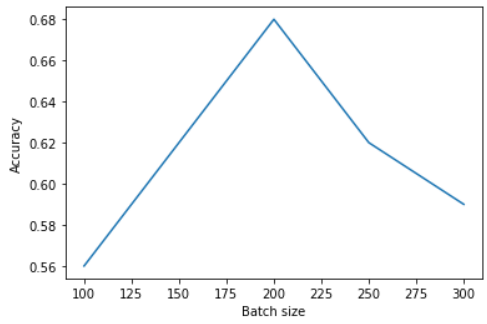
Graph for SGD:



Graph for Perceptron:



Graph for Clustering:



Summarizing all graphs: