PROJECT REPORT

Team ID: BD2_031_275_375

Project Title: **Machine Learning with Spark MLlib**: Spam email detection using the Spam dataset.

Design Details: Our project implementation begins with the streaming of data in batches using a Spark Streaming Context. This is followed by parsing the streamed json, and then converting it into a dataframe with the appropriate schema. We then preprocess the message column of the batch of data using RegexTokenizer - to tokenize each sentence, StopWordsRemover and Word2Vec, that converts our preprocessed text into a vector so that it can be trained into a model. StringIndexer is used to encode our target column into 0's for Ham and 1's for Spam. This is followed by fitting the pipeline output into the different ML models (Logistic Regression, Gaussian Naive Bayes, SGD Classifier, MLP Classifier) for training incrementally using the partial_fit function. Parameter tuning was performed at this stage to obtain the most optimum parameters for the model. Once all the batches are streamed, the models are saved and then tested using the test dataset. We also implemented clustering using the MiniBatchKMeans function from sklearn.cluster. We trained the model incrementally and then predicted the clusters for the test data. We also plotted a graph for the same, to visualize how accurately the data was being clustered into spam and ham. This is followed by evaluation of the test results. The libraries we used are: pyspark, numpy, sklearn, matplotlib, pickle and json

Surface Level implementation details: Our aim is to train machine learning models to be able to accurately classify spam emails from ham ones. Our project utilises Spark Streaming to read the train dataset in batches. Since the spam dataset consists of only categorical columns, the first step was passing each batch into a pipeline for preprocessing. This included tokenizing, removing stop-words and encoding our target column into binary values - where 0 denotes ham and 1 denotes spam. This was followed by incrementally training our models. We achieved this using scikit learn's partial_fit function for the MiniBatchKMeans Clustering, SGD, Gaussian Naive Bayes and MLP classifiers, and warm_start parameter for Logistic Regression. Finally, after all batches of data were successfully streamed, preprocessed and fit into the model, we saved our models into separate pickle files.

For testing, we once again streamed the data in batches of an arbitrary batch size, following which we passed the data through the same pipeline we used for training. On completion of preprocessing and encoding, we predict the values for the inputs and compare it with the actual values to obtain the score for each model. Following this, we compute the F1 score, recall, precision, accuracy and display the confusion matrix to evaluate our results. We also plotted a few graphs to help visualize the same.

Reason behind design decisions: We have implemented streaming of the dataset using Spark Streaming which is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. The data is ingested from a TCP socket and is first converted to a JSON object and then into a dataframe. Once the batch is converted into a dataframe, we proceed with the preprocessing of the data.

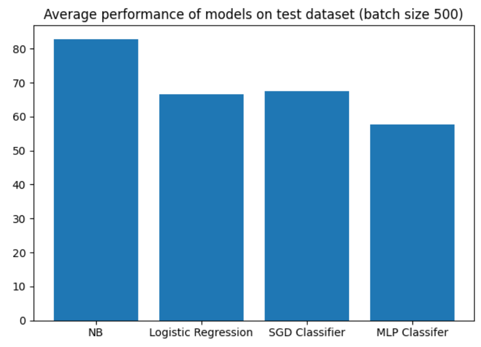
The preprocessing of the data used in our program involves separating the body of the email into regex tokens or individual words, after which the stop words present in the list were removed. This is done so that words that appear frequently in both spam and ham emails are removed and the performance of the algorithms is increased. The filtered words are then converted into their respective vector forms so that we can proceed to train the models. The target columns are also converted into numeric values by indexing them by converting "Ham" values as 0 and "Spam" values as 1. This completes the preprocessing of the data.

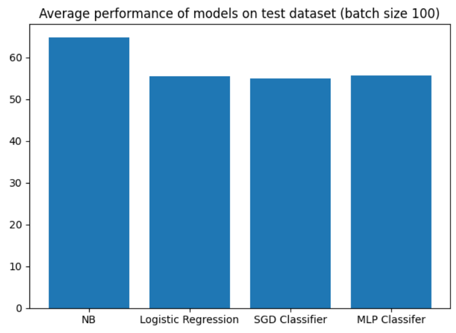
After the preprocessing we proceed with the implementation of the Machine Learning algorithms. Spark MLlib is an extremely useful library, which consists of many algorithms that can be implemented for machine learning. In our code, we have implemented four machine learning algorithms namely, gaussian naive bayes algorithm, the multilayer perceptron algorithm, the stochastic gradient descent algorithm and the logistic regression algorithm. The reason we implemented these four algorithms is because they are all popularly used classification algorithms and accept the vectorized form of the filtered words as input while training the model. They also had functions that supported incremental learning, i.e the partial_fit function and gave us impressive accuracies for a spam/ham email classifier. Moreover, the Naive Bayes algorithm has widely been used in the past for spam detection, and hence we chose the same for our analysis.

We have also calculated various metrics for the different models such as precision, recall, F1 score and the confusion matrix for each batch. Various plots on the scores of the different models have also been plotted.

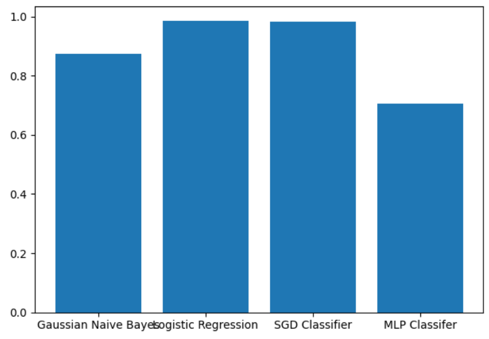
Take Away from the project: From this project we have learned how data can be streamed in batches from data of very large size, how we can work on that data and how we can analyse and study this data. Spark streaming has proven to be an interesting topic and it has been a lot of fun. Implementing machine learning algorithms in spark turned out to be a simple task as many functions were predefined, easy to use and well documented. We also learnt that of the four models we implemented, the Naive Bayes algorithm proved to be the most accurate, followed by the SGD model, Logistic Regression model and then the MLP model. While testing, larger batch sizes showed to have higher accuracies per batch than smaller batch sizes. We achiever accuracies of over 80% using Naive Bayes Classifier for a batch size of 500, and around 60% for a batch size of 100, as shown in the graphs below.

Average performance of the different models for a batch size of 500 and 100.

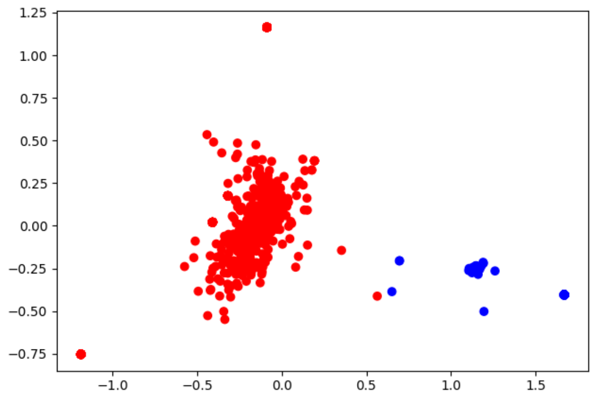




Training accuracies for the various models:



Plot for clustering of a subset of the data, where red denotes ham and blue denotes spam:



1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.0
−0.5
0.0
0.5
1.0
1.5