UE19CS322 Big Data Project 2
Machine Learning with Spark MLlib

Team:

1. Darshil Shah - PES1UG19CS131
2. Mahim Dashora - PES1UG19CS251
3. Nityam Churamani - PES1UG19CS307
4. S Ritesh Kumar- PES1UG19CS403

Design Details (Different Models Implemented):

1. Pre Processing
2. Naive Bayes Classifier
3. SGD Classifier
4. Passive Aggressive Classifier
5. MiniBatchKMeans

Implementation:

1. **Training:**
    a. The spam train dataset is first streamed through spark using stream.py. While executing the stream file the batch size is mentioned as an argument. We used three different batch sizes: 100, 500, 1000
    b. The data streamed from stream.py is collected in train.py. train.py collects each batch as a rdd and processes it and creates a dataframe of the rdd, i.e converts the json to a dataframe for training the machine learning model.
    c. The different machine learning training algorithms are implemented in the pipeline.py file. train.py passes the dataframe generated from a batch to the respective functions to train the model.
    d. Incremental learning is performed by partially fitting the data frame to the model. We used sklearn's partial_fit function to partially fit the data to the model that is being trained.
    e. When all the data has been streamed in batches the algorithm stores the trained model using pickle.

2. **Testing:**
    a. The spam test dataset is first streamed through spark using stream.py. While executing the stream file the batch size is mentioned as an argument. We used three different batch sizes: 100, 500, 1000.
    b. The data streamed is collected in test.py. test.py converts the stream data batch from json to a dataframe. It then tests the data batch wise and prints the accuracies.

Design Decisions:

1. **Pre-Processing**: We pre-process the data using stemming and using stop words. We use one hot encoding for spam/ham data.

2. **Naive Bayes Classifier(Gaussian Naive Bayes)**: belongs to probabilistic classifiers based on **Bayes Theorem.** An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification. Gaussian Naive Bayes allows the user to enter negative values as well and scales them between 0 and 1, thereby allowing seamless training and testing.

3. **SGD Classifier**: This is a linear classifier optimised using Stochastic Gradient Descent (SGD). The main advantages of SGD are efficiency and ease of implementation.

4. **Passive Aggressive Classifier**: Passive Aggressive classifier is an online learning algorithm where you train a system incrementally by feeding it instances sequentially.

5. **Mini Batch KMeans**: was used instead of K-means because as the number of clusters increases, the computation time increases because of its constraint of needing the whole dataset in main memory. Mini k-means uses smaller batches which can be stored in the memory, so the computation time is not as high as K-means.

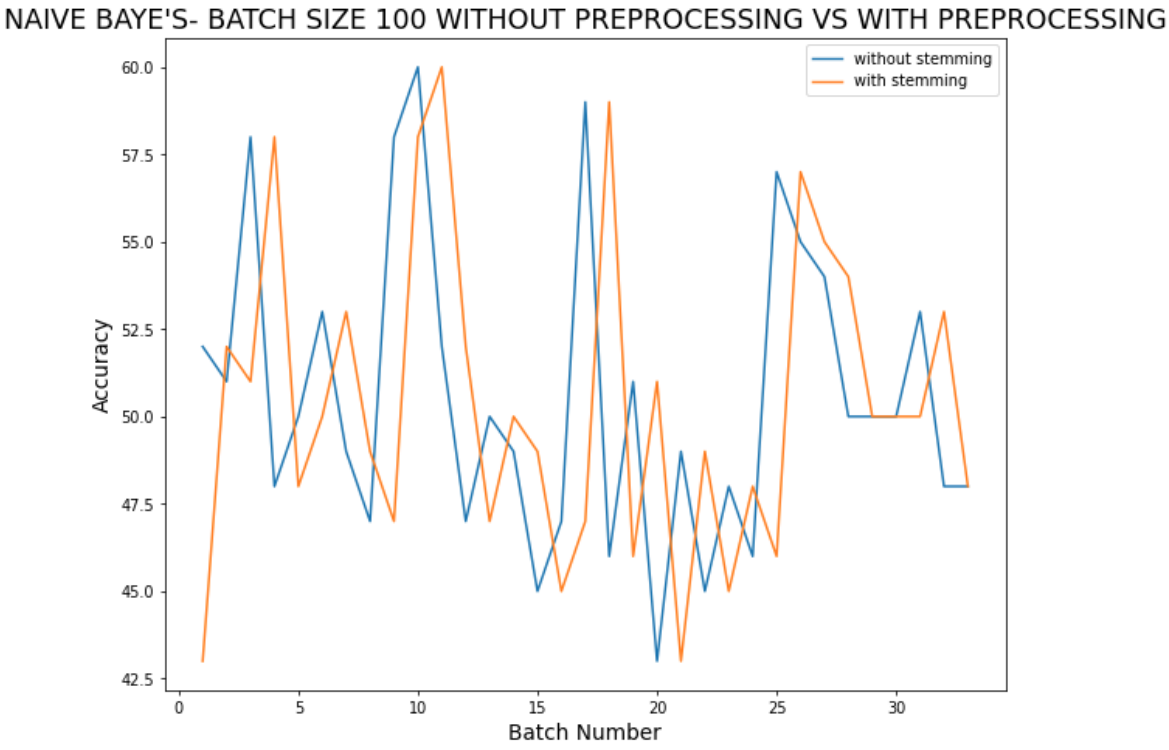
    All the models have been trained under the partial_fit feature provided under scikit-learn. This allows incremental learning on the batches as and when they come in.

Take Away from the Project:

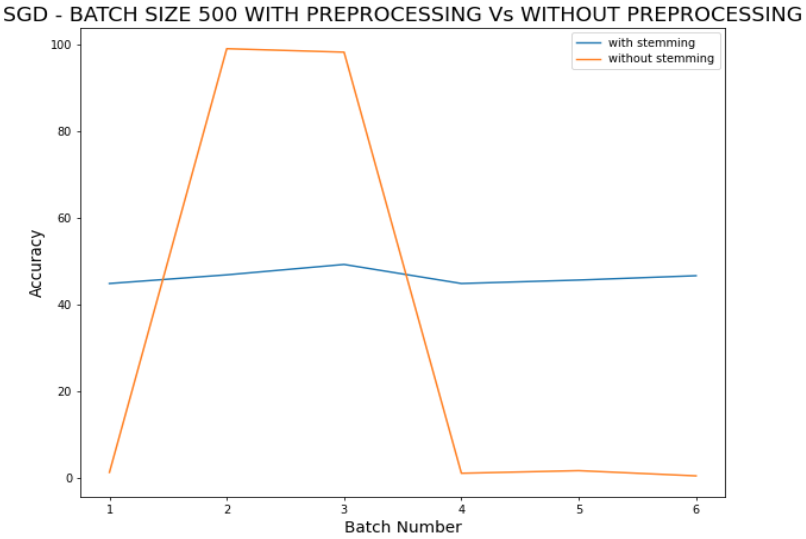
We learnt how real world applications use data collected in real time to train machine learning models. We learnt how spark can be used to stream data from different nodes and train the model simultaneously instead of just collecting all the data and training the model later. This project was a great experience and made us appreciate the usefulness of streaming in spark.

Inferences

Naive Bayes - Batch Size 100 (Without Preprocessing vs With Preprocessing)

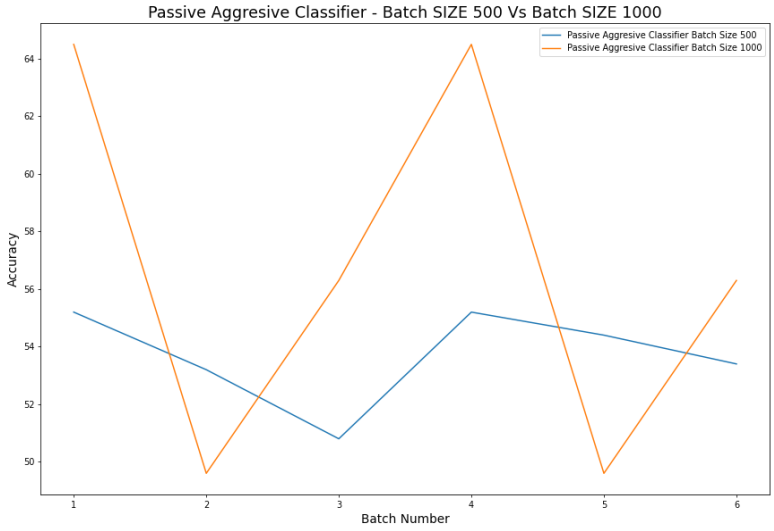


SGD



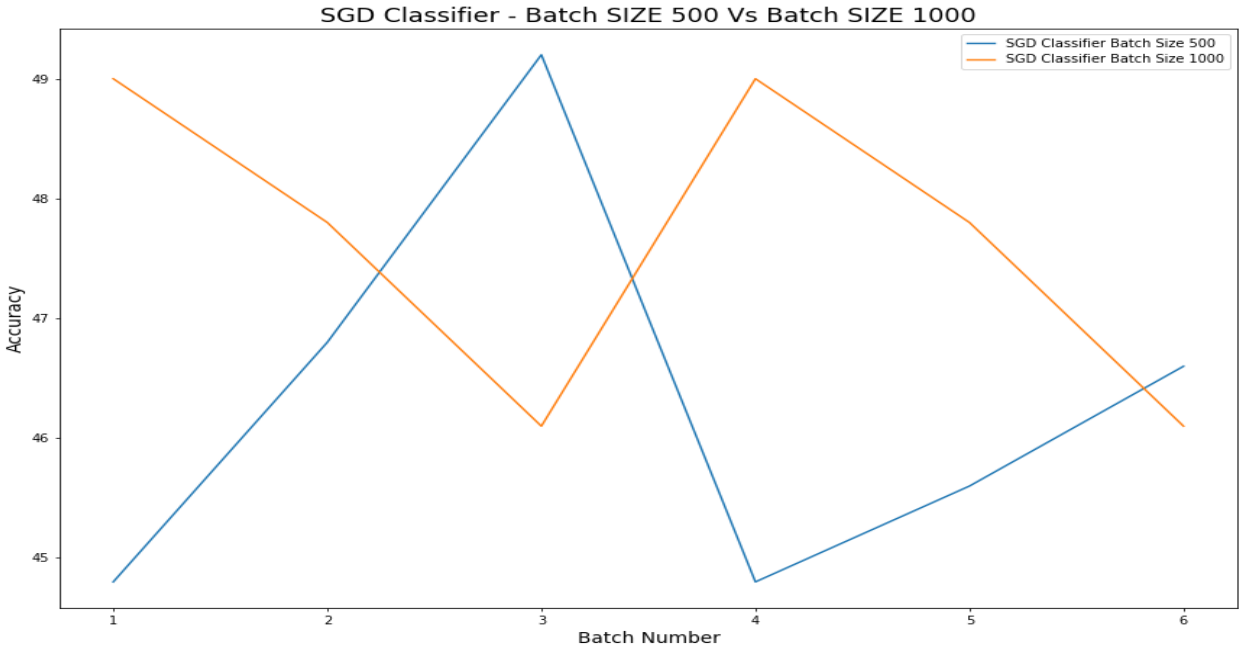
From the above graph we can see that when the data is not pre-processed the accuracy of the model is random but if the data is pre-processed with the help of stemming the accuracy is more consistent.

Passive Aggressive Classifier(500 Vs 1000 Batch Size)



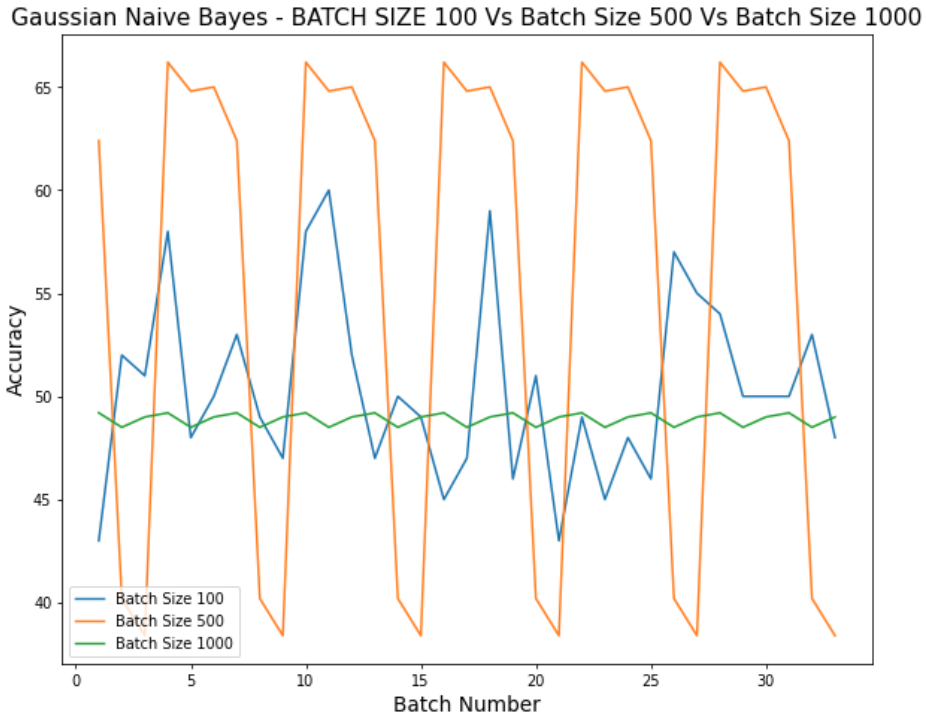
The graph clearly shows that as the batch size increases then the accuracy of the model improves. The reason lies in the working of the model. Passive Aggressive Classifier increases the weights of the wrongly classified samples for the next batch.

SGD Classifier (500 Vs 1000 batch Size)

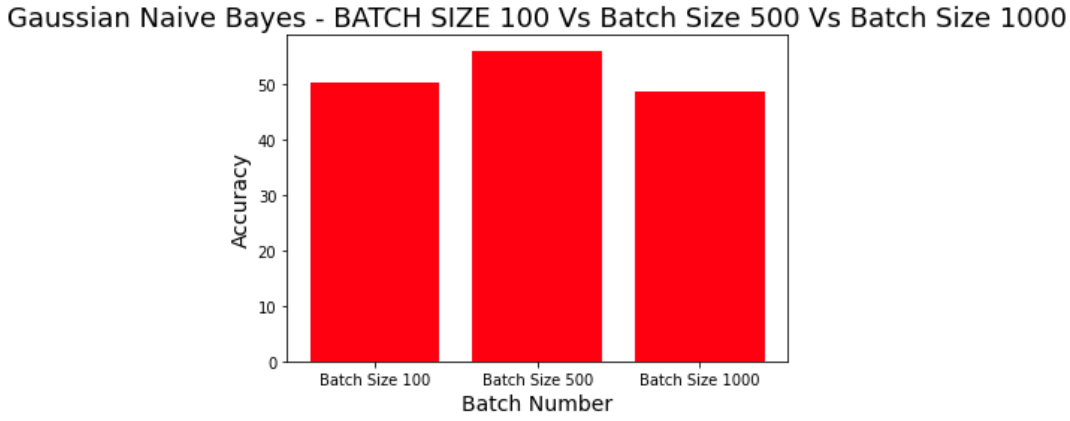


This shows the variations in accuracy with respect to 500 and 1000 batch sizes.

Gaussian Naive Bayes (Comparison of different Batch sizes)

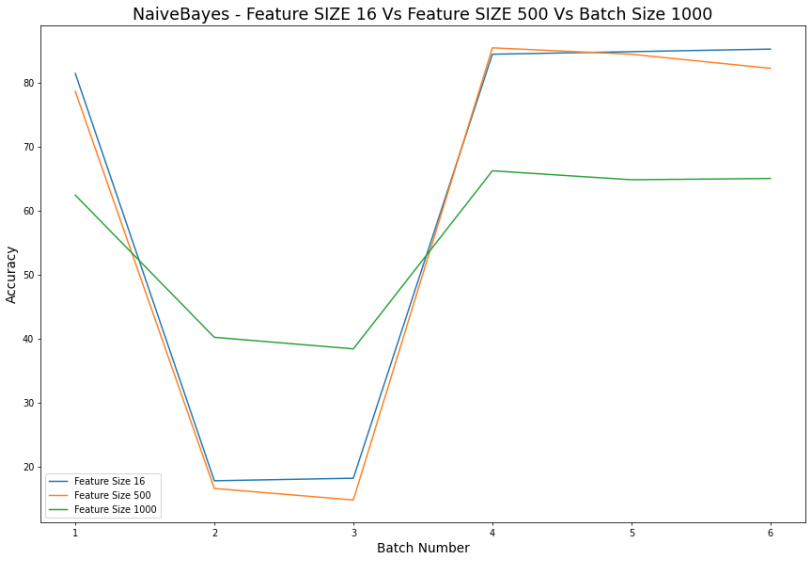


From the above plot we can observe that the accuracy of the model when trained on the data with a batch size of 1000, has more consistent accuracy compared to the models trained on batch size 100 and 500.

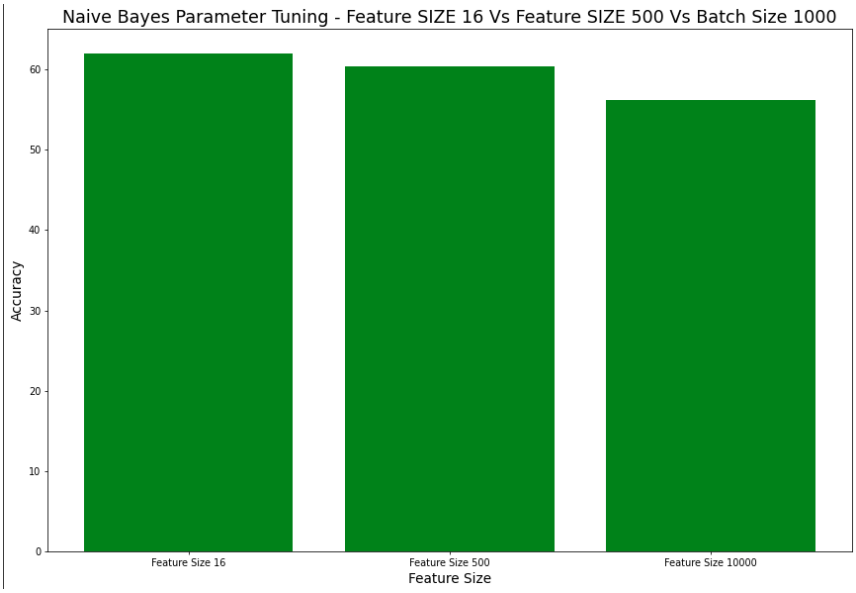


On calculating the average accuracy of the models trained on different batch sizes, it is observed that the model trained on batch size 500 has the higher average accuracy compared to batch size 100 and 1000

Change in Accuracies with change in Feature Size (Changing the Hyperparameters)



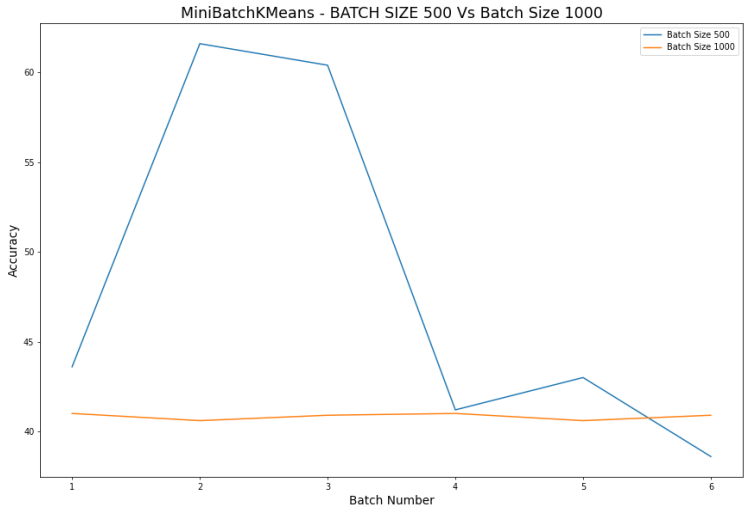
As the features values are increased the collisions in the hashtable decrease and thus the stability in the graph is noticed. Thus, the graph's bottom starts to move up.



From the above plots we can observe that Naive bayes model when tuned with 16 parameters produces the model with highest accuracy

Mini Batch KMeans:

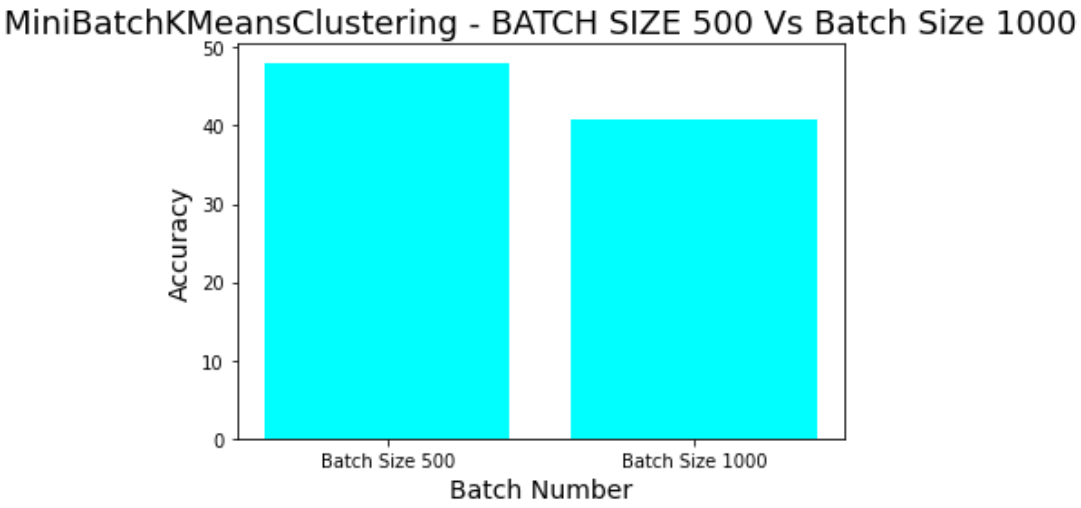
Below are a few plots: -



This plot shows the variation in accuracy for MiniBatchKMeans against varying batch sizes.

As is evident, with a batch size of 500 accuracy turns out to be better.

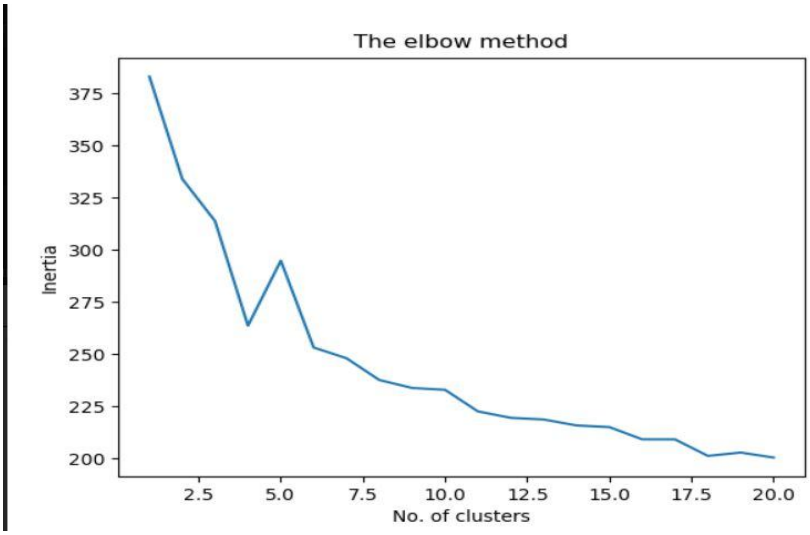
A bar plot backing our claim: -



Elbow Curve:

An elbow curve is the best method to determine the best K value that should be considered for clustering.

Thus, the process of clustering can be optimized.



Here inertia depicts the sum of squared errors of points in a cluster from a cluster centre, summed up over the number of clusters.

This clearly shows that as the number of clusters increases the error decreases which is stated in theory as well.

Thus, our implementation is correct and supported by theory as well.