Big Data Project Report

Project title chosen: Machine Learning using Spark Streaming

Design details

The above project was implemented on the “Sentiment Analysis” dataset. This project is scripted entirely in python and uses the Spark Streaming tool to send batches of data to aid in incremental learning of the data. Incremental learning is a methodology of Machine Learning wherein the hyperparameters of the model do not start from scratch but rather, use pre-calculated hyperparameters, where available. Each batch, post the first batch of data, helps in improving the metrics of the model to ensure better predictions. This implementation makes use of a bunch of python modules: Scikit-learn, ntlk, pyspark and so on. The use-case of each of these modules will be enumerated in the respective implementation unit. For every incoming batch of data, the data is preprocessed to aid in modelling textual data. The dataframe is passed through a hyperparameter tuning script to map the ideal set of parameters for predictions. Post this, the dataframe is passed to three different classifiers, these classifiers are trained and used for predictions. The predicted values are compared against the actual values based on a variety of metrics to assess the models. A Clustering model is used to cluster the tweets into the two categories of sentiments.

Surface level implementation details about each unit

The project is divided into 3 main units which have been listed below. The flow of this project is preprocessing followed by modelling for each category of models specified post-preprocessing.

1. Preprocessing: This step involved cleaning the data and bringing it into a form that could be used and modelled by the algorithms to generate useful insights. As a part of pre-processing, we first had to convert the incoming DStream into a Dataframe. We use regex to manipulate the data. First, we remove usernames located in the tweet followed by the removal of usernames located at the start of the tweet. Then we remove any websites that might have been linked along with any numbers present in the tweet. We then expand all abbreviated negations, such as don’t, aren’t and so on, to their expanded form. We remove special characters from the tweet and numbers, again. All of the above removals are done using regex checks. Following this, the nltk package is used to tokenize tweets and remove stopwords from the tweets.

2. Classification: As a part of the classification, we built three main models: Multinomial Naive Bayes, Stochastic Gradient Descent Classifier, and a PassiveAggressive Classifier. Before running these models, Grid Search with K-Fold cross-validation is performed to map out the best set of hyperparameters for each classifier. Post this, the training tweets and training labels are passed to a partial fit function to perform incremental learning. Once a batch finishes execution, the hyperparameters and insights learned by the model are pickled and saved. When the next batch comes in, this model is reloaded and trained on the new batch which leads to updation of its hyperparameters and insights. The reasons behind our choice of models will be covered in part 3 of this report. The metrics for evaluation used for classification are accuracy, F1 Score, Recall, and Precision.

3. Clustering: As a part of clustering, we pass the pre-processed dataset to the clustering model. The clustering model used here is MiniBatchK-Means. As a part of the clustering model, we defined the number of clusters to 2, since there are assumed to be one of two sentiments assigned to each tweet. We also define a maximum number of iterations. K-Means by nature is an unsupervised learning algorithm that either runs till it converges or till a maximum number of iterations are run. As before, we use partial fit to train the model to implement incremental learning. The metrics used for analysing the clustering model is the same as the classification models.

Reason behind design decisions

This section of the report is divided into 3 categories to explain major decisions made in the above units.

1. Preprocessing
    a. Removal of usernames, websites and numbers: These values are removed from the tweets since they add no value when it comes to analysing the sentiment of the tweet
    b. Expansion of negations: The abbreviated negations on their own add very little to no value when it comes to classifying based on sentiment. The expansion of the negations leads to the addition of the not word which improves understanding of the sentiment.
    c. Tokenization of the tweet: Tokenization is done to break down the tweets into smaller segments that are easier to comprehend and makes the context easier for modelling

2. Classifying
    a. Hyperparameter tuning: This is done using the grid-search model from Scikit-learn. It uses K-Fold cross-validation to ensure that the best estimator is assigned to the model. This helps in ensuring that the model is fitted with the best hyperparameters possible for the scenario.
    b. Multinomial Naive Bayes: Naive Bayes is usually a baseline for most text classification problems owing to its property of calculating probabilities which aid in easier classification. Multinomial NB in particular is more effective when text is attributed to distinct properties, here, the distinct value is the sentiment.
    c. Stochastic Gradient Descent Classifier: Stochastic gradient descent is known to handle large datasets as is the case in this project. The reason SGD is so efficient is that it converges quickly due to more frequent updates.
    d. Passive Aggressive Classifier: The passiveAggressive classifier works by responding passively to correct classification and aggressively to incorrect classifications. PassiveAggressive Classifier is a category of Online Learning which can handle bigger and sequential data. It also has a regularisation parameter.

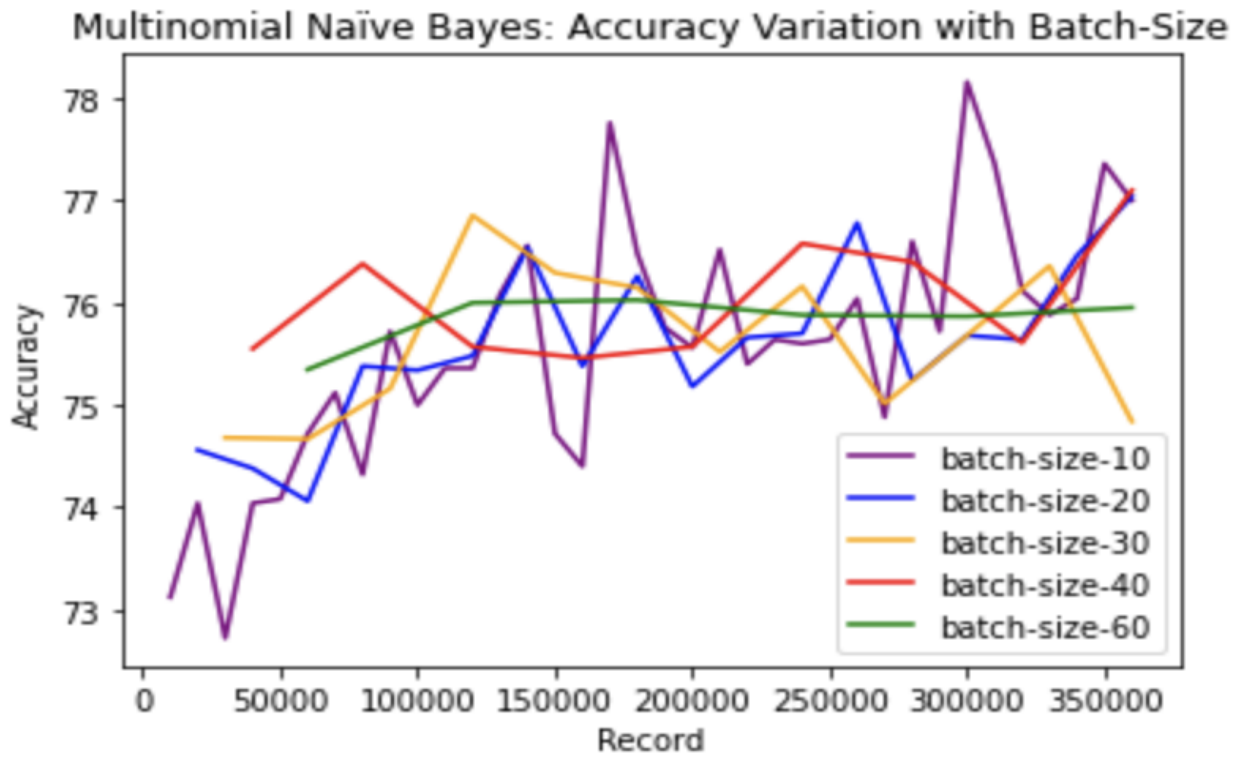
3. Clustering: Mini Batch K-Means works on the concept of selecting a random sample from the dataset and running the K-Means algorithm for n iterations on the sample or until convergence. Due to sampling, Mini Batch K-Means is computationally better in terms of resources. This is a significant benefit in our case since the dataset being passed is huge which increases computational time and resources. The downside is that being unsupervised, performance in this case takes a massive hit. One can assume that the reason for this is that the clustering occurs based on tweet similarity, and not sentiment.

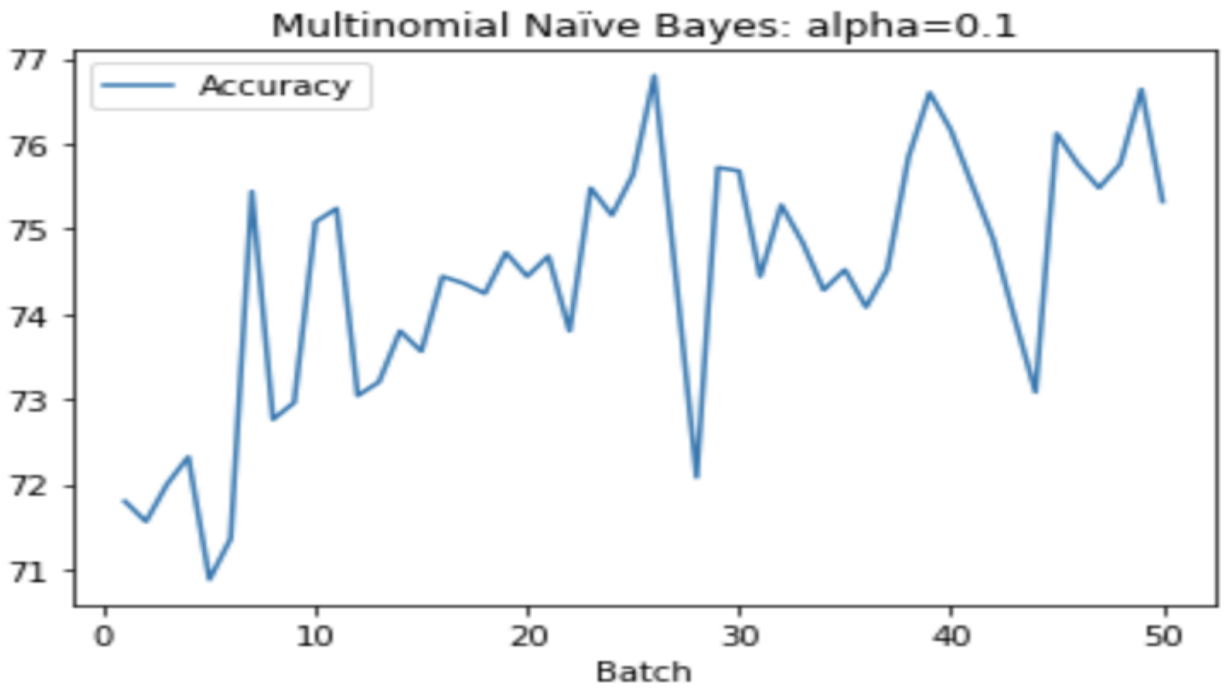
Takeaway from the project

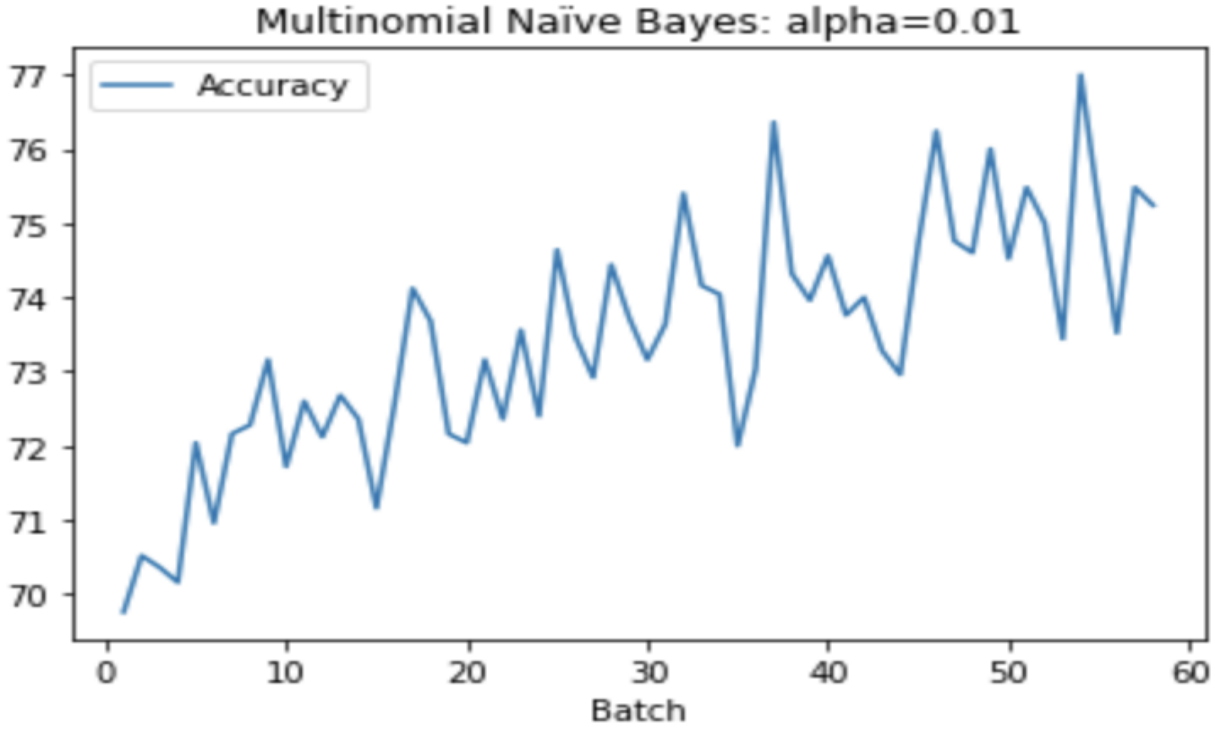
This project helped us experience how incremental learning works and can be implemented in real-case scenarios. It helped us map out the exact algorithms to use and helped us understand the benefits of hyperparameter tuning.

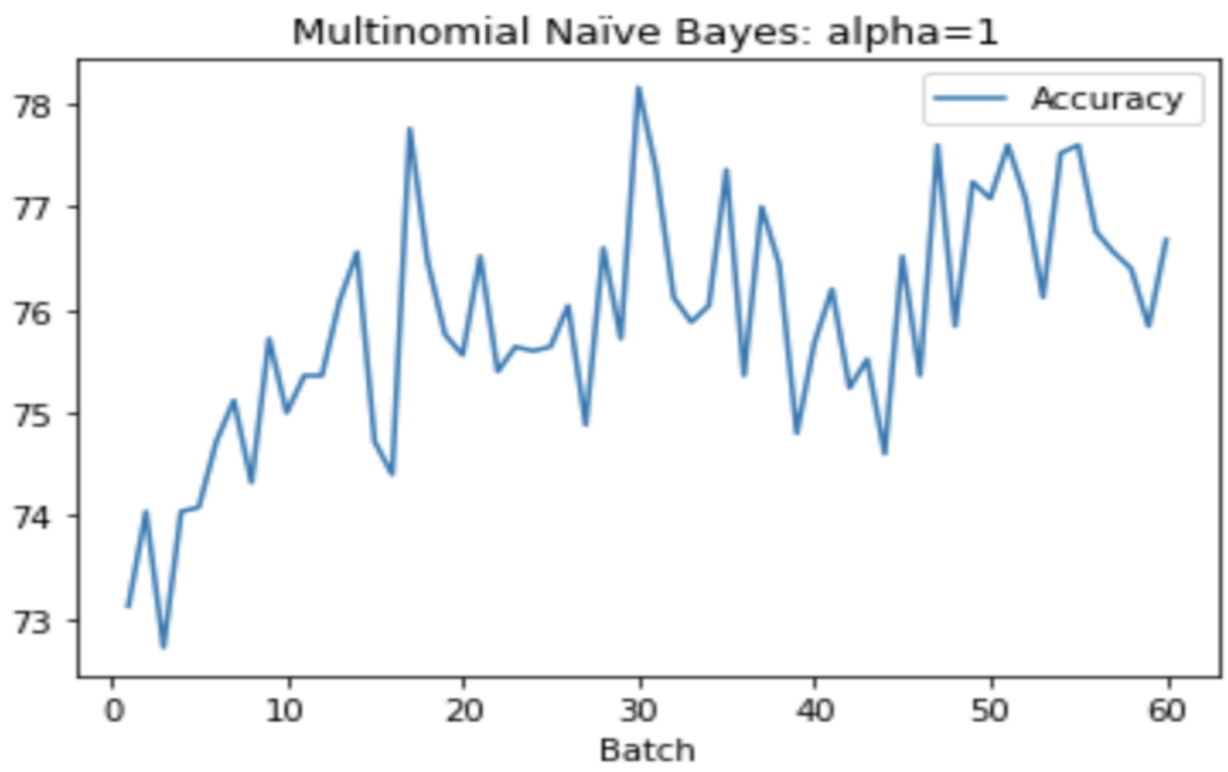
Analysis:

1. Multinomial Naive Bayes:

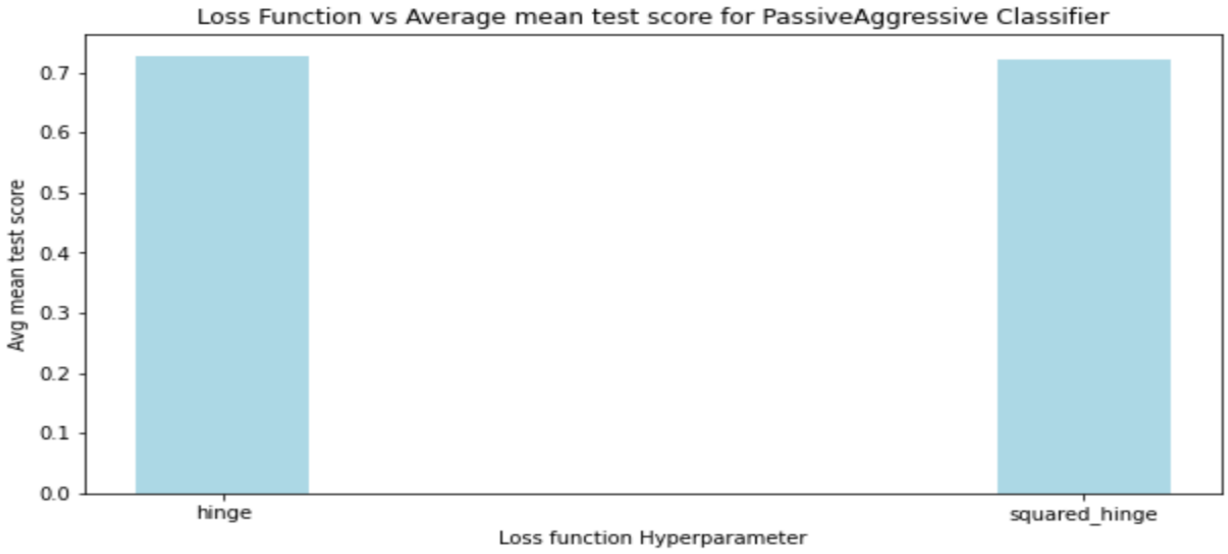


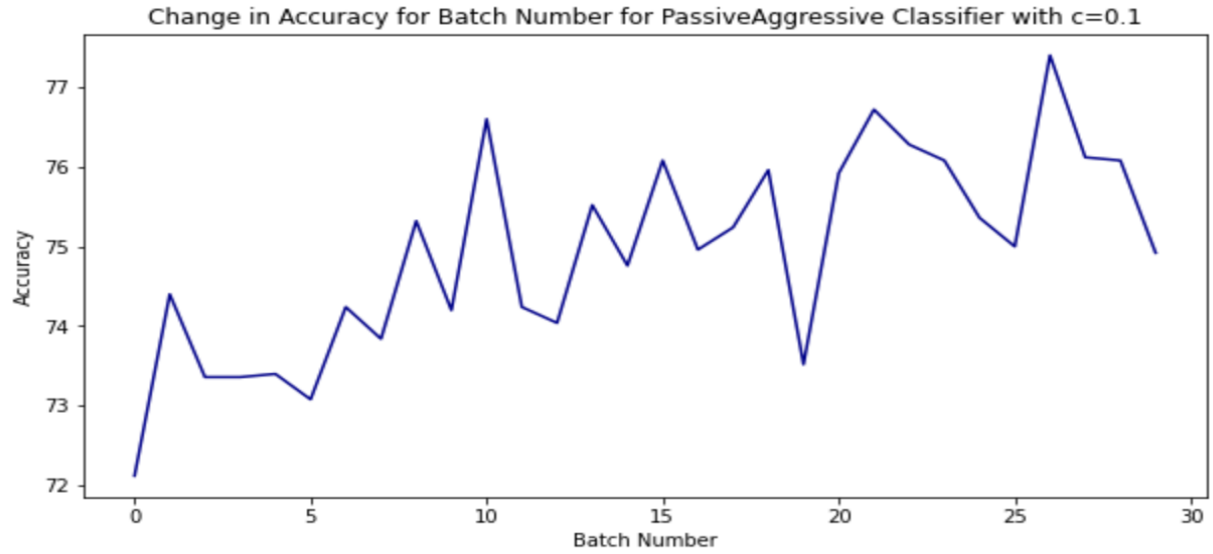

Multinomial Naïve Bayes: alpha=0.1
Accuracy
77
76
75
74
73
72
71
0
10
20
30
40
50
Batch

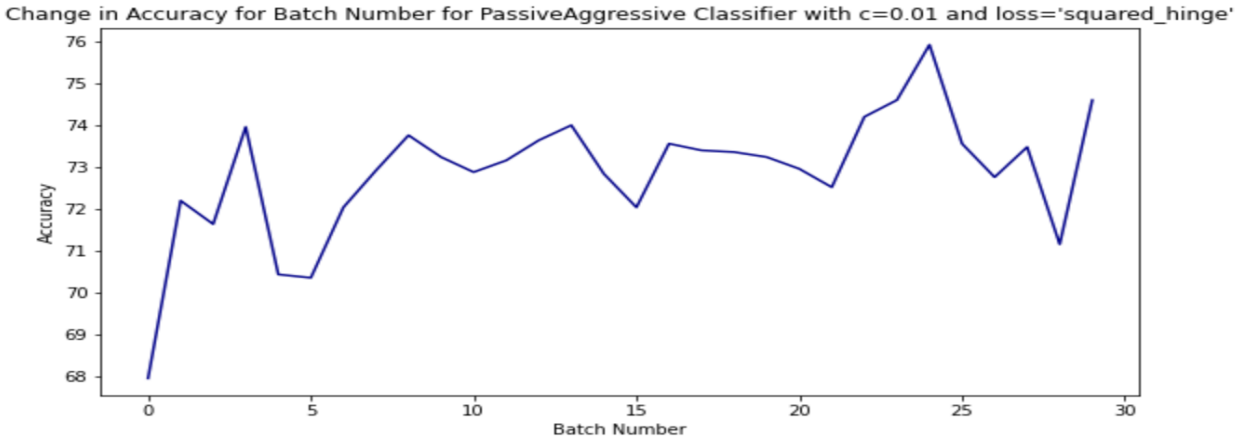

Multinomial Naïve Bayes: alpha=0.01
Accuracy
77
76
75
74
73
72
71
70
0
10
20
30
40
50
60
Batch

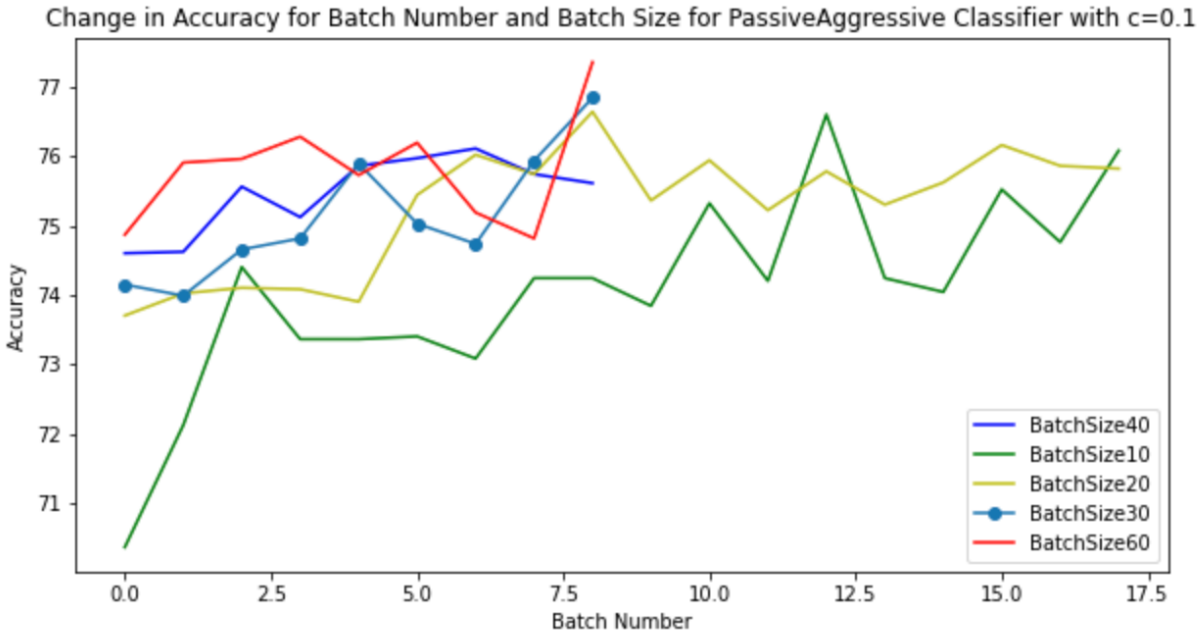

Multinomial Naïve Bayes: alpha=1
Accuracy
78
77
76
75
74
73
0
10
20
30
40
50
60
Batch

2. Passive Aggressive:

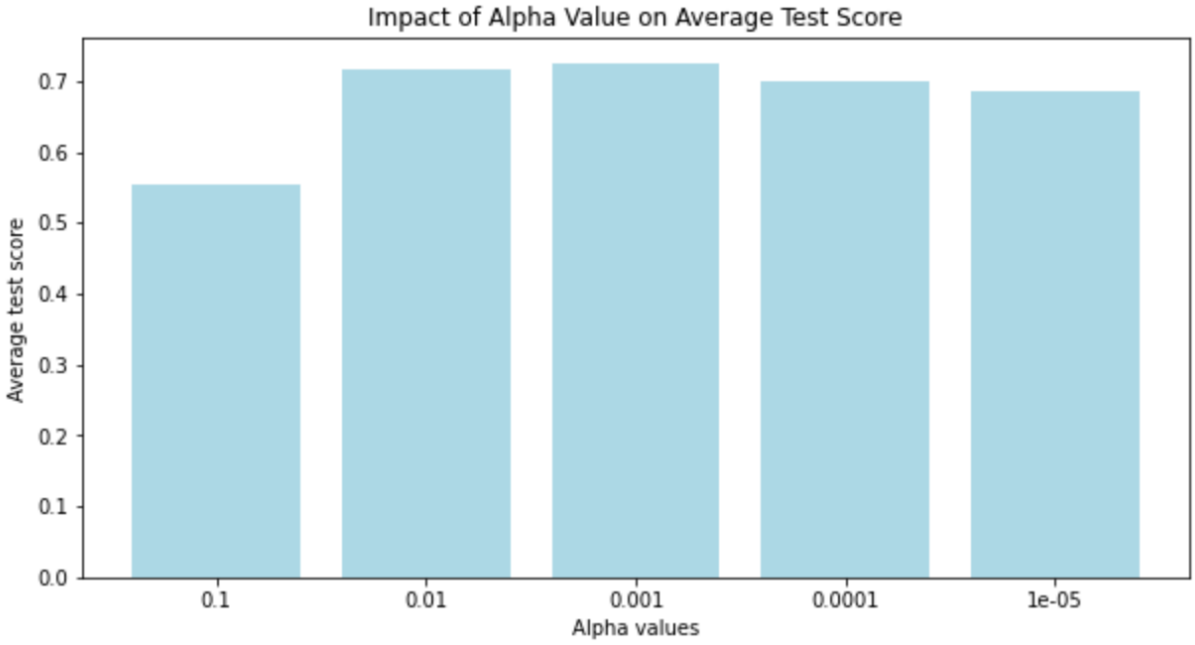


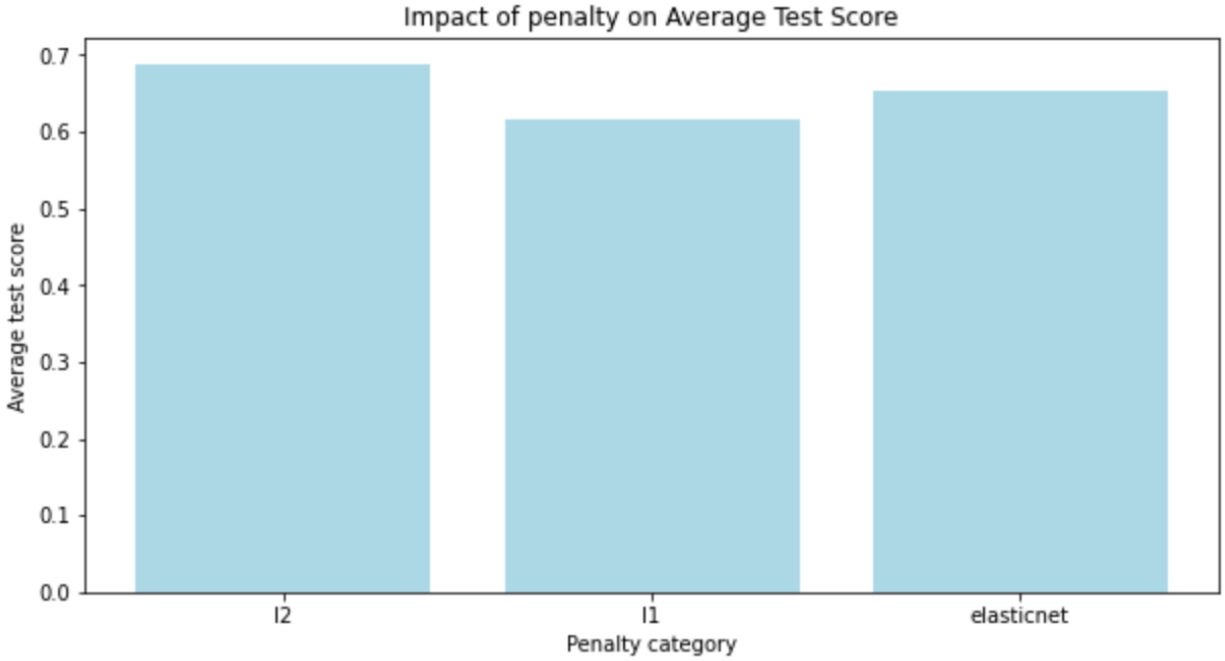



Change in Accuracy for Batch Number for PassiveAggressive Classifier with c=0.01 and loss='squared_hinge'
Accuracy
Batch Number
76
75
74
73
72
71
70
69
68
0
5
10
15
20
25
30


Change in Accuracy for Batch Number and Batch Size for PassiveAggressive Classifier with c=0.1
Accuracy
Batch Number
77
76
75
74
73
72
71
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
BatchSize40
BatchSize10
BatchSize20
BatchSize30
BatchSize60

3. SGD:


Impact of Alpha Value on Average Test Score
Average test score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.1
0.01
0.001
0.0001
1e-05
Alpha values


Impact of penalty on Average Test Score
Average test score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
l2
l1
elasticnet
Penalty category