Recommender System(Review based)

EE 660 Project Type: Individual

Author: Ruchin Patel,

Email: [ruchinpa@usc.edu](mailto:ruchinpa@usc.edu)

USC ID: 9520665364

Phone number: 4322662949

Date: 3rd Dec 2018

**Please organize your report along the lines of this template**; you may use any word processing software you like, as long as you submit your report as the required pdf file described below.

**Your report must be typewritten and submitted as a pdf document**, in machine readable form (no scans or screen shots).

**Please submit your code as a second pdf file**, all code in the one file, also required to be machine readable (no scans or screenshots).

# Abstract

A brief, informative description of your project. Include the problem, approach (naming the machine learning methods you used in your project), and key results.

The abstract should be considered a “stand alone” section – it should be understandable on its own, and includes only information that is described (and supported) elsewhere in the report.

Tip: many people find it works better to write the abstract last, even though it will be read first.

# Introduction

# Problem Type, Statement and Goals

**Problem Type:** Classification + Sentiment Analysis + Recommendation

**Statement:** Build a recommendation engine on the basis of the reviews written by users for the products purchased by them. Classify reviews and then use collaborative filtering on the polarity of reviews to find similarities.

**Goal:**

*Typical Systems:*Typical recommender systems perform collaborative filtering technique which uses “User Behavior” for recommending items. The “User Behavior” here is nothing but the purchasing pattern of every user. It is the most commonly used technique in the industry as it not dependent on any other additional information. However every or most of the recommender systems tend to use the final ratings(1-5) to find the correlation or similarity between different users and thereby finally recommending items to different users.

*My Approach:*Instead of using the final ratings(1-5) to find the likings and dis-likings of every user, I am using the reviews given by every user to identify those underlying sentiment of likings and dis-likings related to a product. For this the first thing that is needed, is to classify the reviews and identify the underlying polarity of every review i.e ‘is the review Positive or is it Negative ?’. If the review corresponding to a product is classified positive, then the user likes that product and if classified negative the user dislikes . After finding the polarity I plan on using the same collaborative filtering model to recommend products to users, the only difference would be that instead of using the final ratings for finding similarities, the collaborative filter would now use the sentiment or polarities of reviews to find those similarities.

*Importance of My Approach:*  The recommender system that I am trying to build is quite different from the typical recommender systems. I think words can better express the sentiment of a persons’ thoughts rather than rating from 1 to 5 for a given product. And hence a recommender system based on the sentiment analysis of reviews can better capture a persons’ likings and dis-likings and thereby recommend better products.

**Difficulties:** The main difficulties faced in this project are as follows:

1. *Size:*The total number of data points used are 346,355. Total users in this data set are 38,609 and total items or products included are 263,032. Thus not only I am supposed to classify 346,355 reviews, but after that I am even supposed to find the similarities between 38,609 users.
2. *Sparsity:* The reviews would be inherently vectorized through the bag-of-words technique and since there would we many words in those 346,355 reviews, the final vectorized version of the reviews would be highly sparse.

# Literature Review (Optional)

Briefly describe existing approaches to your problem. The literature review doesn’t need to be exhaustive, but it should cover well known publications, so you are aware of their approach, tools and results.

# Prior and Related Work (Mandatory)

Prior and Related Work - None

# Overview of Approach

**Models Used:**

The algorithms used were specifically used to classify the reviews as positive or negative. As a result only classification algorithms were used and the best one from 5 of them was selected. The algorithm considered initially were:

1. *Naïve Bayes:* The reason for choosing Naïve Bayes is that empirical results have proven that Naïve Bayes performs a very good task at sentiment analysis. It is used in many email-Spam filters.
2. *AdaBoost Classifier:* Adaboost algorithm is adaptive and the subsequent weak learners are tweaked so that the previously misclassified points are classified correctly. It is also less susceptible to over fitting and hence a better choice for our problem.
3. *Logistic Regression:* Logistic Regression in our problem gives us the probability of a review being classified as positive or negative. Thus every review can either be positive or negative depending with some probability.
4. *Random Forest Classifier:* The classifier creates many random decision trees at the training time and finally prediction is based on the mode or the mean of the classes predicted by those random trees.
5. *SVM:* The reason for trying out the support vector machine is that if the true distribution of the polarity of reviews is not linearly separable then an SVM with ‘rbf’ kernel would help in giving us nonlinear boundaries.

**Performance Metrics:**

1. *F-Beta Score:* We are interested in recommending products similar to the products that the user has bought and liked. Additionally identifying a review as positive when it actually negative(i.e False Positive) would be determinantal to our recommender system since we are looking to recommend products similar to the products liked by the user and in that sense determining the True positive and False Positive of review classification is more important. Therefore, a model's ability to precisely predict those reviews that have an actual rating of 4 or 5 is *more important* than the model's ability to **recall** those reviews. We can use **F-beta score** as a metric that considers both precision and recall.

Fβ = (1+β2)⋅

In particular, when β=0.5, more emphasis is placed on precision. This is called the **F0.5 score.**

1. *ROC\_AUC:* AUC(area under the curve of the Receiver Operating characteristics) is used because since the data is highly biased towards one review class positive, our classifier might decide to classify all the reviews of the test set to be in the positive class, and we will still get a very high accuracy, but such a classifier would be of no use for a new data. However when we use F0.5 score and AUC we get the insights of the True Positives and False Positives through F0.5 score and The True Positive rate and False Positive rate through the ROC and AUC.

**Model Comparison/Selection Methodology:**

Since the dataset is huge along with the performance metric, training time Is also of the utmost importance because some algorithms like SVM take very long time to get trained whereas algorithms like Naïve Bayes take very little time comparatively. Thus the following methodology was carried out for selecting the initial model from the 5 models considered above and the steps are.

1. Make a Naïve predictor as the base model. This Naïve predictor would classify every review as positive
2. Randomly select 7000 data points from the data.

Calculate the time, training and testing accuracy/F0.5 score for the subset of those 7000 randomly selected data points. Select the model or classifier that has its F0.5 score greater than the Naïve predictor’s F0.5  score. Moreover the model should have a good balance between training time and the performance metric.

1. Use the selected classifier and do cross validation for hyper-parameter selection and generate a final best classifier model.
2. Use this best classifier to train on the entire training dataset, and test it on the entire training data. Note the performance metrics.
3. Use this final model to predict the sentiment of all the reviews in the data and make a new column of binary polarities(Positive and Negative). Finally use item-item collaborative Filtering technique on user-items with target as the reviews sentiment to generate recommendations.

.

# Implementation

## Data Set

**Source:** Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering  
R. He, J. McAuley  
*WWW*, 2016

## **Description:** This dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014.This dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs).

**DATA:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | reviewerID | asin | reviewText | overall |
| **Feature Info** | ID of the reviewer | ID of the product | text of the review | rating of the product |
| **Feature Type** | Unordered Categorical | Unordered Categorical | Text | Ordered Categorical |
| **Range** | 38,609 | 263,032 | Reviews total: 346,355  Vocabulary: 140,198 | [1,2,3,4,5]  5 being the best and 1 being the worst |

### *Review:*

*Metadata:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | asin | title | imUrl |
| **Feature Info** | ID of the product | name of the product | url of the product image |
| **Feature Type** | Unordered Categorical | Text | Image URL |
| **Range** | 263,032 | ---- | ---- |

## Preprocessing, Feature Extraction, Dimensionality Adjustment

The goal of this project is to classify reviews. Also there are no missing values in the for ‘reviewerID’, ‘asin’, ‘reviews’ and ‘overall’ features and hence preprocessing and Dimensionality reduction are to be done only for the ‘reviews’ feature.

**Preprocessing:** In order for the text data to be applied to an ML classification algorithm, it has to be converted to number form. To do this we convert it to bag of words format. This technique works in the following way.

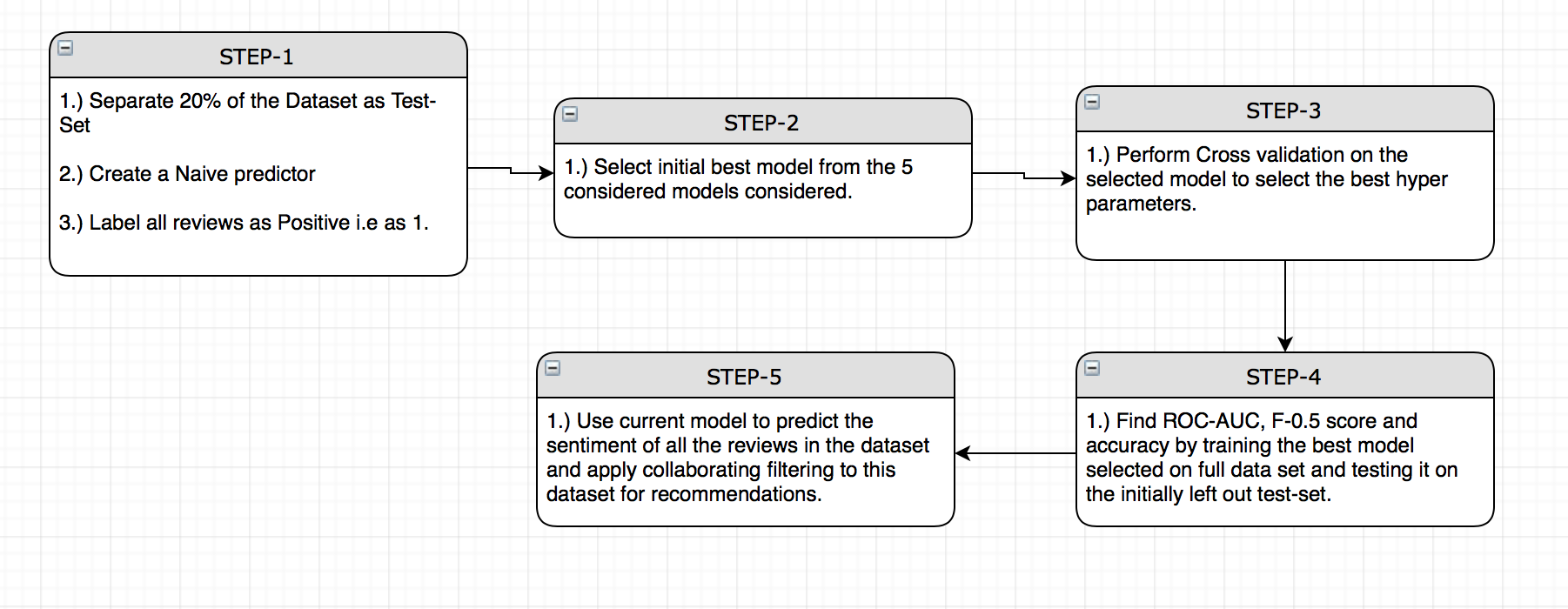
* Find all the unique words occurring in 346,355 reviews. This set of unique words is called the Vocabulary. Here the size of the Vocabulary is 140,198.
* For every review create a vector V the size of the Vocabulary and for every word in the review append the occurrence of that word to the corresponding position in vector V.
* Thus the final matrix of vectorized reviews will be of the RMxN where M = 346,355 and N = 140,198. In this new vectorized reviews data the occurrence of the words are the features. Thus R is a very high dimensional data with features = 140,198. Also most of the word from this Vocabulary of 140,198 words occur very less and hence R is also a very sparse matrix.

**Dimensionality Reduction:** A text data contains stop words, punctuations etc. Moreover when it comes to sentiment analysis there would be certain words which do not contribute to the sentiment classification at all. For example words like ‘comfortable’, ‘good’, ‘best’ etc. capture positive sentiments, whereas words like ‘bad’, ‘unhappy’ etc. correspond to negative sentiments. These words also tend to occur more in the reviews and the words that have no contribution to sentiment tend to occur less in the reviews. So the dimensionality reduction is done in the following way.

* *Remove all the stop words* from the reviews as they do not provide any important information for a reviews sentiment. Stop words include words like ‘is’, ’at’, ’which’, ’on’ etc. These are the most commonly occurring words in the English Language.
* *Remove punctuations.*
* *Remove words occurring less that 1% of the time*. This is needed as there might occur cases when a person mentions a brand of the product or uses some word not common to sentiment analysis. If we include such words we might throw off our sentiment classifier.
* After doing the above steps the final Vocabulary obtained has 625 words which would essentially be used to capture the sentiment of every review. Thus in RMxN M = 346,355 and N = 625. Thus the matrix R is no longer sparse.

**Libraries Used:** Sk-Learn CountVectorizer, Pandas, Numpy

## Dataset Methodology



**STEP-1:**  The first step for any Machine Learning System is to create a base model against which every other model would be compared. It is also wise to separate the test-set before all the other processes like model selection, hyper-parameter tuning etc. The following are the steps carried out during step one.

* Separate The test set from the original data. This test set is 20% of the entire data and hence it will contain
* Since we are interested in the sentiment/polarity of the reviews we convert the entire rating column to a binary column. Thus any review having a rating of 4 or 5 is considered as positive and given the value 1 whereas any rating having values 1,2 or 3 is considered negative and assigned the value 0. Thus the final distribution of the class labels is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Rating | Review value | Total reviews | Total review percent |
| Positive | 1 | 279,801 | 80.78% |
| Negative | 0 | 66,545 | 19.22% |

It is to be noted that the total number of positive reviews is almost 4 times that of the negative reviews and as a result of this we use the metrics F0.5  and ROC-AUC to know the True and False positive rates.

* Since 80.78% of all reviews are positive, even if we create a classifier which classifies everything as 1, we would get the following results for accuracy, precision and F0.5  score.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy | Precision | Recall | F0.5 |
| 80.78% | 80.78% | 100% | 84.01% |

* Thus we should build a model that is better than this Naïve model and has F0.5 score better than this Naïve predictor.

**STEP – 2(*Model Tuning*):**  Note that this is a part of the training process and hence most of the details will be discussed in section 3.4. This step includes selecting initial model on the basis of training time and performance. The following are the steps taken:

* Randomly select 7000 points from the dataset, all the while maintaining the ratio of classes in these 7000 points.
* Create subset of 1%, 10 % and 100% from these 7000 data points.
* Train and test on all of these subsets and record the Time, accuracy and F0.5 score.
* Select the final model that has a good balance between performance and training time.
* To check whether the selected model works fine, sample 5000 data points randomly from the entire dataset 1000 times all the while keeping the ratio of classes and generate a 95% confidence interval for the distribution of accuracies and F0.5 score. From this 95% confidence interval check the upper and lower range of accuracies and F0.5 score.
* *Libraries Used:*  MatplotLib, Seaborn, Numpy, pandas, sklearn

**STEP – 3(*Model Selection*) :** Cross-Validation for hyper-Parameter Tuning. The following are the steps carried out.

* The initial model being chosen in *Step-2*, we now have to tune it properly so that it gives the best result.
* For this we carry out a grid-search on the entire training dataset i.e 80% of the total data and find the best model that gives the best result on the testing set derived from that same 80% training data.
* *Libraries Used:*  Numpy, pandas, sklearn-gridSearchCV

**STEP – 4 :** Train the final model using the best classifier from *step-3.*

* Train the final best estimator on the training data. Use that model to find the accuracy and F0.5  score on the full training and testing data.
* Calculate ROC and AUC.
* If the performance is better than the Naïve predictor continue with *Step-5*.
* *Libraries Used:*  Numpy, pandas, sklearn-AdaBoostClassifier

**STEP – 5 :**  Conduct the following steps.

* Use the final model to predict the sentiment of all the reviews.
* After this apply collaborative filtering to original ratings and the predicted sentiments and checks if the products recommended by both the method are the similar in their nature and similarities. Use item-item similarity collaborative filtering technique here.
* *Libraries Used:* GraphLab and TuriCreate for collaborative filtering.

## Training Process

**STEP-1:** From the table in *STEP-1*  of section 3.3 we see that the Naïve predictor is a simple predictor that gives us a simple learner that classifies all the reviews as positive. It gives us a good performance. The reason for this good performance are as follows:

* Accuracy of 80.78 is due to the fact that the percentage of all positive reviews is 80.78% and since we are classifying every review as positive the accuracy is same as the proportion of the positive classes.
* True Positive = 279,801, False Positive = 66,545 and False Negatives = 0. Thus values of precision Recall and F0.5 would be as follows.

*Precision*: = = 80.78% = Accuracy

*Recall:* = = 100%

*F0.5:* Fβ = (1+β2)⋅ = (1+0.25)⋅ = 84.01%

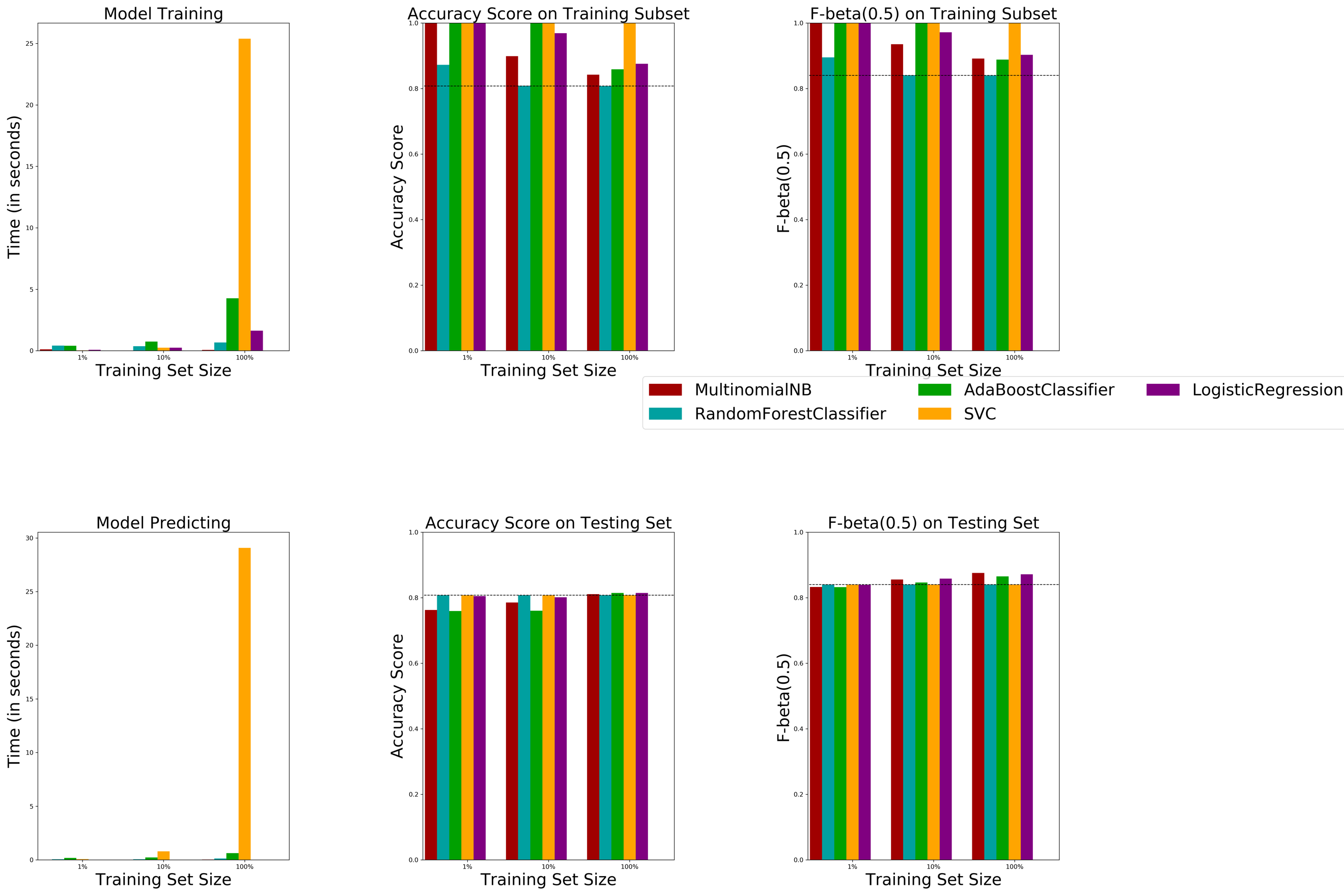
The reason for this high score is due to the fact that precision and recall are very high. Thus when we train our final model it should not only have better estimates of accuracy and F0.5 score but it should also have a good AUC in its ROC characteristics.

**STEP-2(*Model Selection*):**  Selecting the initial model from the following 5 models. Note that we would be considering the basic case with a belief that if a basic model performs better than the Naïve predictor, then the same model with tuned hyper parameters would definitely work better.

* *Naïve Bayes*:
* *Random Forest Classifier*
* *AdaBoost Classifier*
* *SVM with rbf kernel*
* *Logistic Regression*

The following results were obtained after training the above 5 models with the technique described in *STEP-2* in section 3.3.

The plot for selecting the initial best model

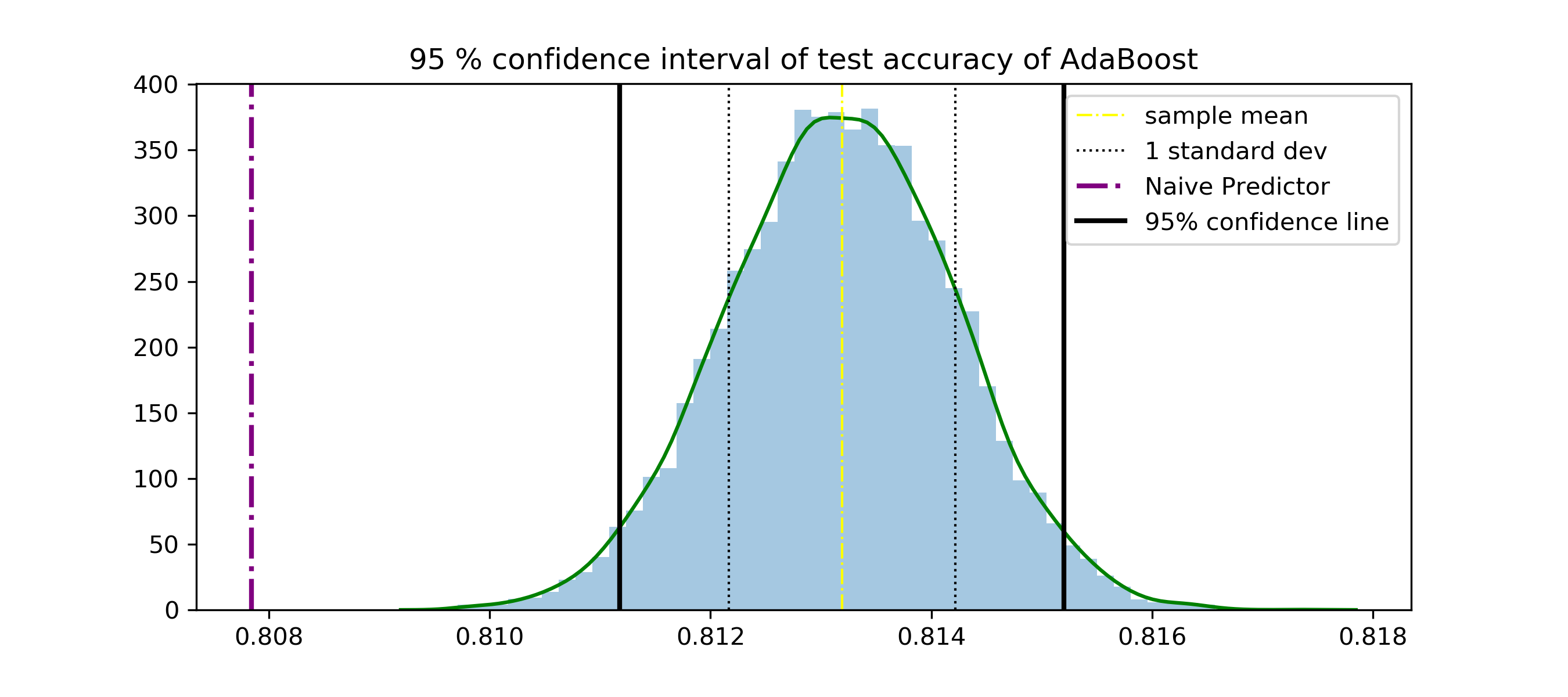
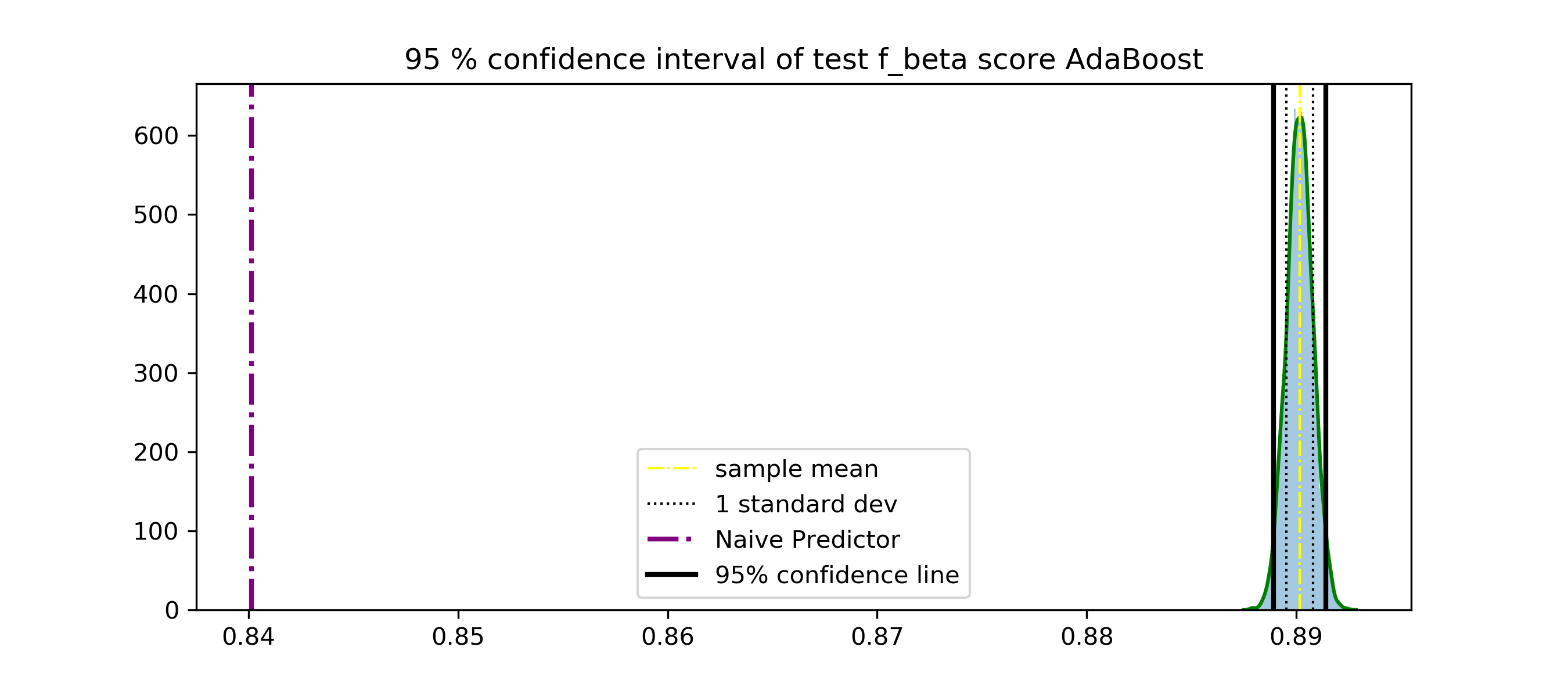


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Models** |  | *Train acc* | *Test acc* | *Train F0.5* | *Test F0.5* |
| ***Naïve Bayes***  *Laplace smoothing param : 1.0* | *1%* | *1.0* | *0.76* | *1.0* | *0.83* |
| *10%* | *0.89* | *0.79* | *0.93* | *0.85* |
| *100%* | *0.84* | *0.81* | *0.90* | *0.87* |
|  |  |  |  |  |
| ***Random Forest***  *n\_estimators = 200*  *max\_depth = 2* | *1%* | *0.87* | *0.80* | *0.89* | *0.84* |
| *10%* | *0.80* | *0.80* | *0.84* | *0.84* |
| *100%* | *0.81* | *0.80* | *0.85* | *0.84* |
|  |  |  |  |  |
| ***AdaBoost***  *n\_estimators = 200*  *base estimator:*  *DecisionTree(max\_depth = 1)* | *1%* | *1.0* | *0.75* | *1.0* | *0.83* |
| *10%* | *1.0* | *0.76* | *1.0* | *0.84* |
| *100%* | *0.85* | *0.81* | *0.89* | *0.88* |
|  |  |  |  |  |
| ***SVM***  *kernel : ‘rbf’*  *C:2*  *gamma: 20* | *1%* | *1.0* | *0.8* | *1.0* | *0.84* |
| *10%* | *1.0* | *0.81* | *1.0* | *0.84* |
| *100%* | *0.99* | *0.80* | *0.99* | *0.84* |
|  |  |  |  |  |
| ***Logistic Regression***  *penalty: l2 regularization* | *1%* | *1.0* | *0.80* | *1.0* | *0.84* |
| *10%* | *0.86* | *0.80* | *0.97* | *0.85* |
| *100%* | *0.87* | *0.81* | *0.90* | *0.87* |
|  |  |  |  |  |

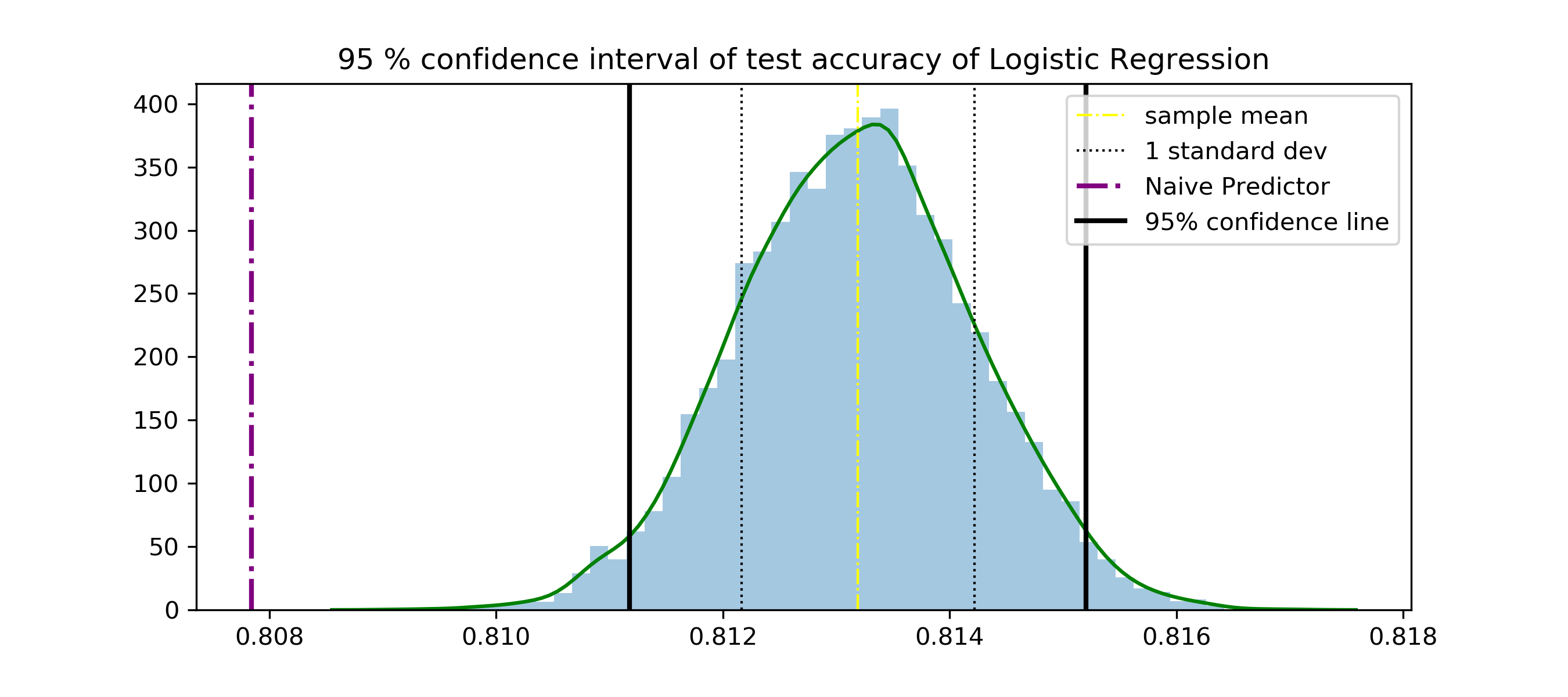
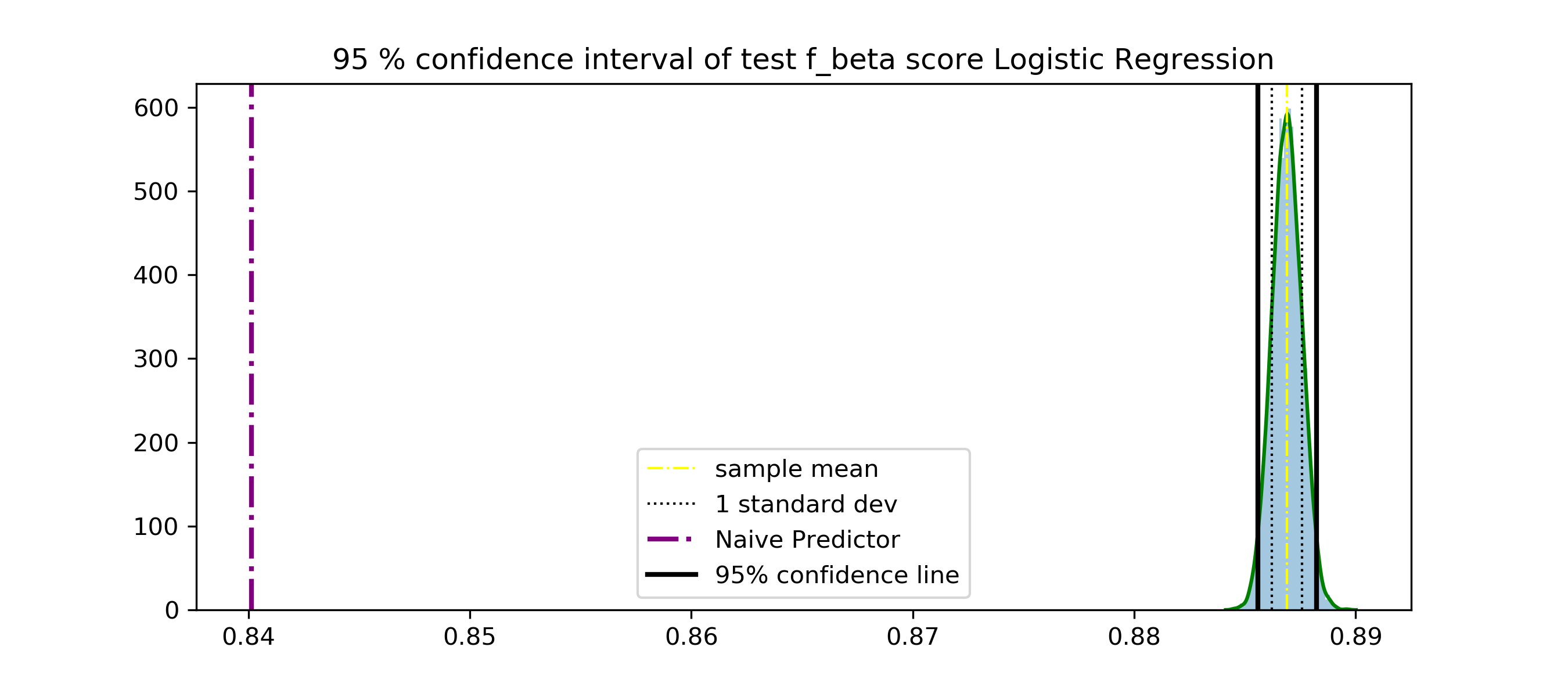
As seen from the above graph and table, Naïve Bayes, AdaBoost and Logistic regression perform the best. It should be noted that we could have gotten a better model with SVM, but since this is a huge dataset, and the time take for training SVM is pretty high as compared to other models we ignore it and consider the models that gave us the highest F0.5 score.

* Naïve Bayes gives us a good performance here and its training time is also very less. However its as good as it can get. There are other models like AdaBoost and Logistic regression that gives us better results and hence we will go with them.
  + - * + Statistics related to Logistic Regression and AdaBoost Classifier: Here we will randomly sample 5000 data points from the original dataset, do the training and testing on those 5000 data points and calculate accuracy and F0.5 score. We repeat this process 1000 times to create a distribution of accuracies and F0.5 and create a 95% confidence interval. After that we check if the value of accuracies and F0.5 of Naïve predictor lie within or outside this 95% confidence interval. If it is outside this interval we can be 95 percent confident that the accuracy and F0.5 of our final model would lie within this range. And since the performance metric of the Naïve predictor is way outside this interval, we can be fairly certain that our final model would perform better as compared to the Naïve predictor.

Adaboost:

Logistic Regression:

The above shown plots are the distribution of accuracies and F0.5 score of the AdaBoost classifier. Also since both the plots are fairly normal we can create a 95% confidence interval on the basis of the normal probability table.

*Accuracy:*  The plot on the left indicates that we are 95% confident that the value of the accuracy of our final model would be within the area covered by the two black lines. Also since the accuracy of the Naïve predictor is very far away from the this 95% confidence range we can be sure that the final model would perform better than the Naïve predictor.

*F0.5:* The same argument goes for the F\_beta score in which the Naïve predictor score is many standard deviations away from the normal plot of the F0.5  scores. Thus our final model would perform better than the Naïve predictor.

* + - * + Now Logistic regression has an F0.5 score of 0.87 whereas the AdaBoost classifier has and F0.5 score of 0.88. Also the standard deviation of the distribution of AdaBoost classifier is less as compared to that of logistic regression. Because of these reason I have selected AdaBoost classifier as my final model and will tune it for the best hyper parameters.

**STEP-3(*Model Tuning*):**  Having selected the AdaBoost classifier in *STEP-2* we will now have to tune the Hyper parameters in order to get the best model. But first let’s take a look at what this AdaBoost classifier can do.

AdaBoost:  AdaBoost is a technique of fitting weak learners repeatedly on the data , finally to get an overall better classifier.

* + - * + *Strengths:* The main strength of AdaBoost is that it has very less hyper parameters to tune(namely 'base\_estimator', 'n\_estimator' and 'learning\_rate'. Due to this the AdaBoost algorithm is pretty fast as compared to other algorithms like neural networks and SVM.
        + *Weakness:* AdaBoost algorithm is highly affected by the quality of data that we are working with. If our data has quite some number of outliers, then such outliers would affect the performance of the Adaboost classifier.
        + *Why a good Candidate:* Our dataset is large but we have cleaned it to quite some extent. This means that when we removed the low frequency words we were actually removing the outliers. As a result I think that using the Adaboost algorithm with decision stump would be quick in training and would also generate good results on the test data.

AdaBoost Working:

* + - * + The AdaBoost algorithm works by fitting a sequence of weak learners on data that is being modified at every iteration of those fittings of weak learners. At the end of the algorithm the predictions are made by combining all those weak models through a weighted sum. The process takes place as follows:

*Training:* At the start of the training a weak decision tree classifier with depths as low as 1 is used to fit the data in such a way that minimum error is produced. After that all the correctly and incorrectly classified points are counted and the weights of the incorrectly classified points are modified in such a way that the total weight of correctly and incorrectly classified points become equal(This increase in weight is done to make sure that the next weak classifier classifies those incorrectly classified points perfectly when the fitting is done again.) The above mentioned process continues till there are no misclassified points or till maximum iteration is reached. Thus at the end of the training session we have a certain number of weak classifiers which we combine in the prediction step.

*Combining the Models*: As mentioned in the training step above we created a certain number of weak classifiers. There is a weight associated with every such model. This weight can be any real number. For every such model the area classified as positive is given a positive value of the corresponding weight and the areas classified as negative are given negative value of the corresponding weight. After doing this all the models are combined and a weighted sum of all the regions is taken. In the end certain areas of the final model will have values of weights greater than zero and certain areas will have values of weights less than zero.

*Prediction*: Finally any point that belongs to a positive area of the final model is classified as positive and any model belonging to the negative areas is classified as negative.

AdaBoost Hyper-Parameter Tuning: As stated above the hyper parameters that are to be tuned for the AdaBoost classifier are ‘n\_estimators’ and the ‘learning rate’. The base estimator is set to be equal to a Decision stump. Note that the hyper-parameter tuning is done on the the 80% training set.

* + - * + *n\_estimators:* A total of 90 values staring from 100 and going till 1000 with a step of 10.
        + *learning\_rate:* A total of 75 values starting from 0.5 to 2 with a step size of 0.02.
        + After doing the GridSearch, the final best estimator had the following values of hyper\_parameters.

|  |  |  |
| --- | --- | --- |
| **Base estimator** | **n\_estimators** | **learning rate** |
| Decision Stump | 500 | 1.5 |

**STEP-4:** Train the final model using the best classifier from *STEP-3* and train it on the entire Training set and test the trained model on the initially held out test set.

* + - * + Final results are as follows:

|  |  |  |
| --- | --- | --- |
|  | **Accuracy** | **F0.5** |
| **Training Set** | 83.34% | 87.12% |
| **Testing Set** | 83.25% | 87.04% |

Describe how you trained your model, the classifiers or regression processes you used, and the parameters you chose.

For each machine learning method or model you use:

* Describe your model and your algorithm in detail (with formulas and flowcharts). It generally isn’t necessary to repeat equations given in EE 660 just to describe the model and algorithm; however, you must give enough information to clearly define which model and algorithm (and which version of the model and algorithm) you are using. Also, if you want to refer to any equations (e.g., for your interpretation or analysis), you must include those equations in your report. If needed, also explain in detail what you did to adapt the method to your case, and explain the assumptions or “tricks” you used.
* Give some justification for why you chose this method. This can be a simple statement such as: simple method for baseline; non-linear method because problem seems complex; we believe this method would yield the best performance; etc.
* State the parameters of the model and how they were chosen. If a parameter is chosen by heuristics, state so. If a parameter is chosen by some model selection or validation process, state so and describe the details in Sec. 3.3 or 3.5.
* Analyze the complexity of your hypothesis set to the extent possible. Compare with the number of data points you have and the dimension of the pre-processed feature space. Explain what you did to avoid overfitting and underfitting.
* If you have sets of results to show for this machine learning method, include them here. (For a comparison of results from *different* machine learning methods, use the next subsection.)

## Model Selection and Comparison of Results

If you had multiple potential models in the beginning, explain how you performed model selection. No need to repeat what was covered in the Dataset Methodology subsection above.

Present performance comparison of your different models and methods here. Be sure to clearly show what dataset each result is from (training, validation, test if any, averages over multiple cross-validation runs, etc.). Use table(s) and/or plots. **Do not paste print screen images.**

Is the difference between these results as expected? In the case of classification, you can choose two salient features and plot your decision boundaries w.r.t. them. In the case of regression, you can plot the resulting regression function in 3D (as a function of two salient features) or in a few 2D plots (each as a function of one salient feature).

# Final Results and Interpretation

Document your final results. Describe your final system and its parameter values. Give the final performance of your system(s) and an estimate of it’s out of sample performance. Compare with your baseline results and with any results you found in the literature. Include figures, plots, and/or tables if appropriate.

**If you are working on an online competition, report the performance of your best submission and compare it to others on the leader board.** If you want to compare your results with other work, do so here.

Interpretation: Why do you think the results came out the way they did? What has been learned from them? Anything particularly noteworthy or unexpected? Showing your understanding of your work and results is an important part of this project. Note: you can interpret results throughout the report, but this section should contain a final interpretation (e.g., why this system worked the best, and how much better it is or is not, and what else might improve it further)

# Contributions of each team member

If a team project, state here what the contribution of each team member was (i.e., who did what).

# Summary and conclusions

Briefly summarize key findings, and optionally state what would be interesting or useful to do next.

# References

Cite the sources of your information that came from elsewhere. This includes other related works you compare with (if any), and sources of descriptions of systems or methods that you included in your report.

# Your code

**Please submit your code in a separate file.**  All your code should be in one pdf file, machine readable (no screen shots or scans).