Author: Ruchin Patel,
Email: ruchinpa@usc.edu
Phone number: 4322662949
Date: 3rd Dec 2018

1. Abstract

Description: A typical recommender system applies collaborative filtering technique on the basis of the ratings given by the users. However, these systems do not take into consideration the sentiment of the reviews given to these products. So, the main goal of this project is to create a recommender system that works on the basis of the underlying sentiment of reviews given to the products. The final result should be such that the recommendations made on the basis of review sentiments should be similar to the recommendations based on the true ratings. Not only that but the recommendations should match the choices of the users.

Approach: The approach of this project would be as follows:

- Consider a few ML algorithms to choose from (Naïve Bayes, Random Forest, AdaBoost, SVM, Logistic regression).
- Choose a final algorithm from the 5 algorithms mentioned above.
- Do hyper-parameter tuning to select the best hyper-parameters.
- Do final training on entire data-set.
- Use the final model to find sentiment of all the reviews in the entire dataset.
- Apply Item-Item type collaborative filtering technique to this dataset and check how close the results are from the recommendations made on true ratings.

2. Introduction

2.1. 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 it. 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 am I 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.

2.2. Literature Review (Optional)

Collaborative Filtering: Collaborative filtering is a technique used by recommender systems. It is a technique of making predictions of the taste of a particular user by considering the collective preferences or taste of many other users. There are two types of collaborative filtering techniques.

- User-User: This technique finds the similarity scores between users. On the basis of these scores it picks out similar users and recommend products which these similar users buy.
- Item-Item: In this technique, we find the similarities between items and then recommend products to a particular user that are similar to the products bought by him or her. We will be using this collaborative filtering technique in our project.

  - The question is how do we find the similarity. Cosine Similarity between users is given by $S_{u,v}$: $\frac{u.v}{||u||*||v||}$. Here u and v are vectors which indicate the profile of a user in case of user-user technique and of product in case of item-item technique. This profile vector has all the information about the products that the user has bought and the ratings related to it in case of user-user collaborative filtering technique where as the profile vectors becomes a vector containing the information of rating given by every user to a particular product in case of item-item filtering technique.
  - Predictions are made by the following formula.

    - $P_{u,i} = \sum_N(s_{i,N} * R_{u,N}) / \sum_N(|s_{i,N}|)$
    - $P_{u,i}$ = Prediction of an item.
    - $R_{u,N}$ = Rating given by a user u to product n.

- $s_{i,N}$ = Cosine similarity between product i and another product n.

➢ The reference link would be given in section 7.

Current works for recommender Systems: Most of the recommender systems currently work on the ratings given by the users. They do not take into consideration the sentiment of the reviews. I am trying to achieve similar results of a typical recommender system by capturing the underlying sentiment of the reviews. And for that classifying the reviews is an important task that needs to be taken care of.

2.3. 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 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 is 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.

$$F\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + recall}$$

In particular, when β=0.5, more emphasis is placed on precision. This is called the **$F_{0.5}$ score.**

2. *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 $F_{0.5}$ score and AUC we get the insights of the True Positives and False Positives through $F_{0.5}$ 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.

Step-1: Make a Naïve predictor as the base model. This Naïve predictor would classify every review as positive

Step-2: Randomly select 7000 data points from the data.
Calculate the time, training and testing accuracy/$F_{0.5}$ score for the subset of those 7000 randomly selected data points. Select the model or classifier that has its $F_{0.5}$ score greater than the Naïve predictor's $F_{0.5}$ score. Moreover, the model should have a good balance between training time and the performance metric.

Step-3: Use the selected classifier and do cross validation on full dataset for hyper-parameter selection and generate a final best classifier model.

Step-4: Use this best classifier to train on the entire training dataset, and test it on the entire training data. Note the performance metrics.

Step-5: 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.

3. Implementation

3.1. Data Set

Source: Ups and downs: Modeling the visual evolution of fashion trends with 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:

Review:

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

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	----	----

3.2. 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 R_{MxN} where M = 346,355 and N = 140,198. In this new vectorized reviews data the occurrence of the words are the features.

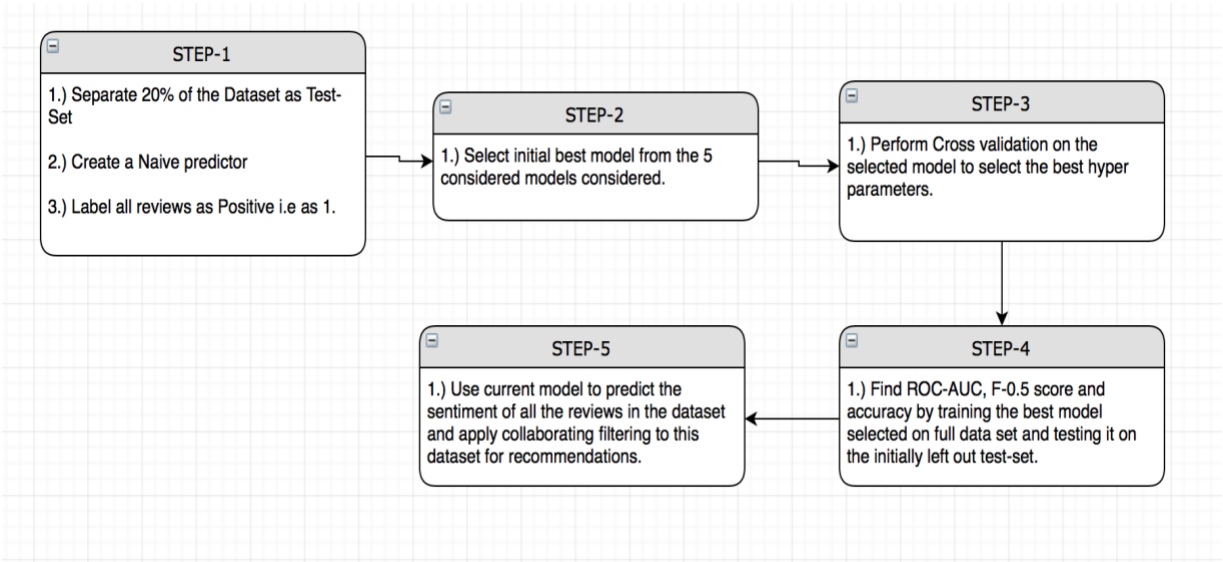
Thus, R is a very high dimensional data with features = 140,198. Also, most of the word from this Vocabulary of 140,198 words occur with low frequencies and hence R is also a very sparse matrix.

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 review's 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 be used to capture the sentiment of every review. Thus, in R_{MxN} M = 346,355 and N = 625. Thus, the matrix R is no longer sparse.

Libraries Used: Sk-Learn CountVectorizer, Pandas, Numpy

3.3. 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 69,271 data points

- 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 $F_{0.5}$ 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 $F_{0.5}$ score.

Accuracy	Precision	Recall	$F_{0.5}$
0.8078	0.8078	1.0	0.8401

- Thus, we should build a model that is better than this Naïve model and has $F_{0.5}$ score better than this Naïve predictor.

STEP – 2(*Model Selection*): Note that this is a part of the training process and hence most of the details will be discussed in section 3.5. 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 $F_{0.5}$ 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 $F_{0.5}$ score. From this 95% confidence interval check the upper and lower range of accuracies and $F_{0.5}$ score.
- *Libraries Used:* MatplotLib, Seaborn, Numpy, pandas, sklearn

STEP – 3(*Model Tuning*) : 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 $F_{0.5}$ score on the full training(80% of full data) and testing data(20% of initially held out set).

- Calculate ROC and AUC. The ROC curve is calculated using the k-fold cross validation technique on the entire dataset. Here k = 4.
- 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.

3.4. 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 $F_{0.5}$ would be as follows.

  - *Precision*: $\frac{\text{TP}}{\text{TP+FP}} = \frac{279{,}801}{279{,}801 + 66{,}545} = 0.8078$ = Accuracy
  - *Recall:* $\frac{\text{TP}}{\text{TP+FN}} = \frac{279{,}801}{279{,}801 + 0} = 1.0$
  - $F_{0.5}$: $F\beta = (1+\beta^2)\cdot\frac{\text{precision}\cdot\text{recall}}{(\beta^2\cdot\text{precision})+recall} = (1+0.25)\cdot\frac{80.78*100}{(0.25*\ 80.78)+100} = 0.8401$

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 $F_{0.5}$ score but it should also have a good AUC in its ROC characteristics.

STEP-2(*Model Selection*): This step follows the procedure in section 3.3. More detailed discussion would be done in section 3.5.

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 80% training or development set.

- ♦ *n_estimators:* A total of 90 values starting 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 has been found, the results of which are mentioned in section 4.

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. Find the ROC-AUC by the k-fold cross validation technique mentioned in *STEP-4* of section 3.3.

- The size of the Training set is: 277,084
- The size of the Testing set is: 69,271

STEP-5: After looking at the performance metric on the full Test-set, predict the sentiment of all the reviews in the dataset as positive or Negative using the model from *STEP-4*. Apply item-item collaborative filtering technique on the original data with the true ratings and then apply the same filtering technique on the sentiment predicted by our model. Compare if the products recommended for both, the true ratings and the predicted sentiments are similar. Also for a new user who signs up, the most popular items should be recommended.

Item-Item Collaborative Filtering:

- ♦ This algorithm uses the similarity between each pair of products to recommend them. So in our case we basically find the similarity between each product pair and based on that we recommend products to the users which are similar to the products liked by them in the past.
- ♦ Here however I am using the GraphLab and Apple's TuriCreate Libraries to do this collaborative filtering. These libraries come with the inbuilt capability of calculating similarities between different products.
- ♦ The reason for using these libraries is that, the main goal of this project is to find the sentiment or polarities of the products.

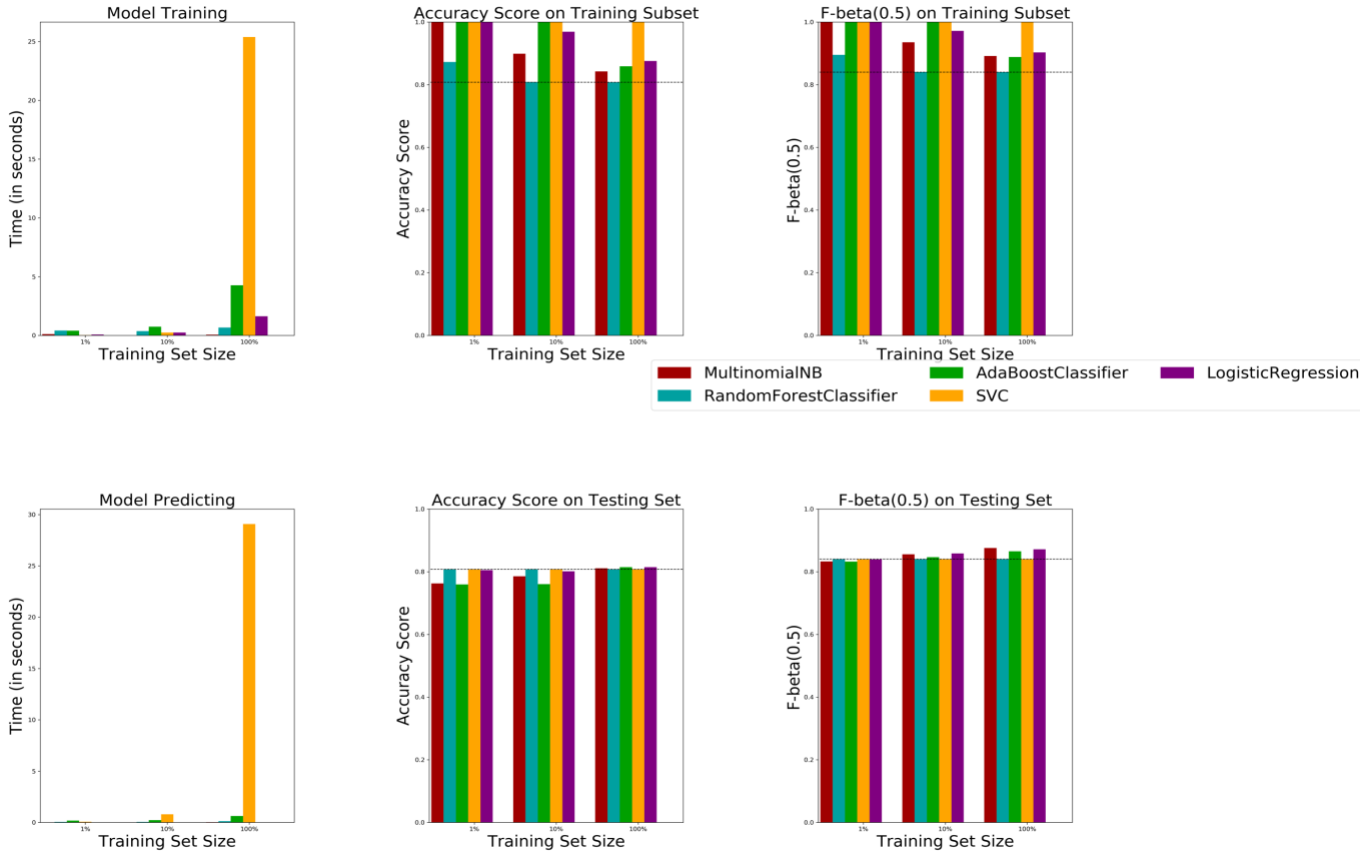
3.5. Model Selection and Comparison of Results

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



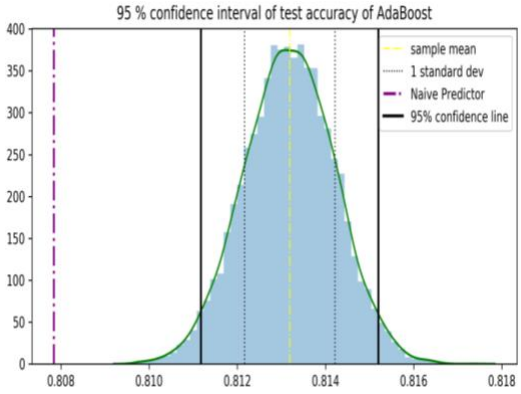
Models		*Train acc*	*Test acc*	*Train $F_{0.5}$*	*Test $F_{0.5}$*
Naïve Bayes *Laplace smoothing param : 1.0*	*1%*	*1.0*	*0.76*	*1.0*	*0.83*
	10%	*0.89*	*0.79*	*0.93*	*0.85*
	100%	*0.84*	*0.81*	*0.90*	*0.87*
Random Forest *n_estimators = 200 max_depth = 2*	*1%*	*0.87*	*0.80*	*0.89*	*0.84*
	10%	*0.80*	*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*	*0.87*
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*

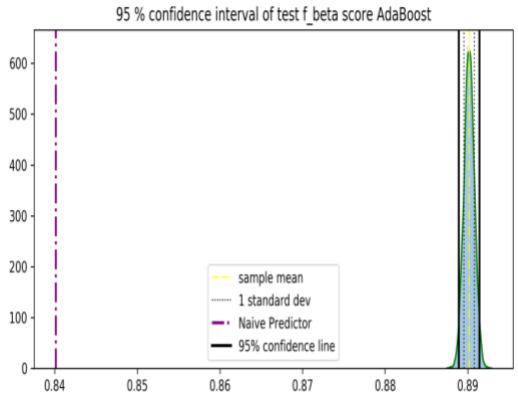
As seen from the above graph and table, Naïve Bayes, AdaBoost and Logistic regression perform the best. It should be noted that we could have gotten a better model with SVM, but since this is a huge dataset, and the time take for training SVM is pretty high as compared to other models we ignore it and consider the models that gave us the highest $F_{0.5}$ score.

- Naïve Bayes gives us a good performance here and its training time is also very less. However it's as good as it can get. There are other models like AdaBoost and Logistic regression that can give us better results if we tune its parameters and hence we will go with them.

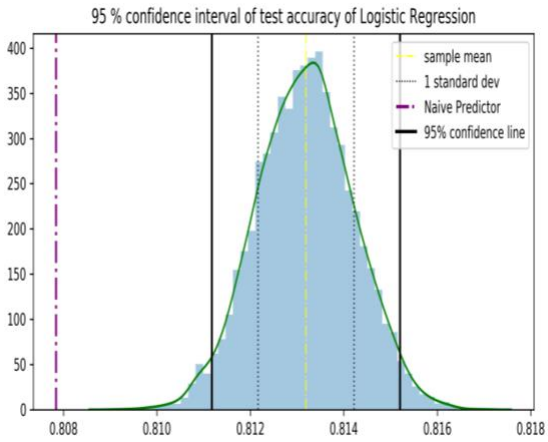
    - ♦ Statistics related to Logistic Regression and AdaBoost Classifier: Here we will randomly sample 5000 data points from the original dataset, do the training and testing on those 5000 data points and calculate accuracy and $F_{0.5}$ score. We repeat this process 1000 times to create a distribution of accuracies and $F_{0.5}$ and create a 95% confidence interval. After that we check if the value of accuracies and $F_{0.5}$ 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 $F_{0.5}$ 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.

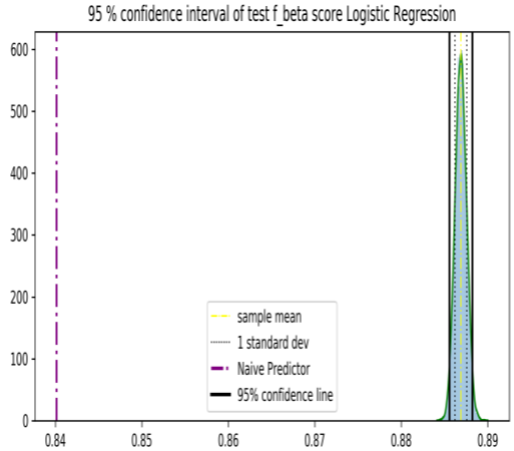
        - ➢ Adaboost:





        - ➢ Logistic Regression:





- The above shown plots are the distribution of accuracies and $F_{0.5}$ 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.
- *$F_{0.5}$:* 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 $F_{0.5}$ scores. Thus our final model would perform better than the Naïve predictor.

♦ Now Logistic regression has an $F_{0.5}$ score of 0.87 whereas the AdaBoost classifier has and $F_{0.5}$ 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 the method of which is given in *STEP-3*.

4. Final Results and Interpretation

➢ AdaBoost Classifier(Final Model): Final performance metrics are:

♦ Final model is:

Base estimator	n_estimators	learning rate
Decision Stump	500	1.5

Some examples of reviews written by me and classified as per its sentiment.

Review	True Sentiment	predicted Sentiment
1.) I am disappointed in this product. It should have worked better.	Negative	Negative
2.) There isn't a product which can be worse than this. I have no idea why I even bought this.	Negative	Negative
3.) I find this product comfortable. This product can never be worse.	Positive	Positive

- The reviews shown in the above table are written by me. And as shown the predicted values match the true sentiment.
- It is particularly important to note that in review number 3, there is a negation, and even though there is a negation in it , our model predicts the true sentiment which is 'Positive'. Thus we can say that the sentiment classifier model that we created works just fine.

- AdaBoost Classifier(Final metrics): Final performance metrics are:

  - Final accuracies and $F_{0.5}$ score from *STEP-4* are as follows:

	Accuracy	$F_{0.5}$
Training Set	0.8334	0.8712
Testing Set	0.8325	0.8704

  - Interpretation:

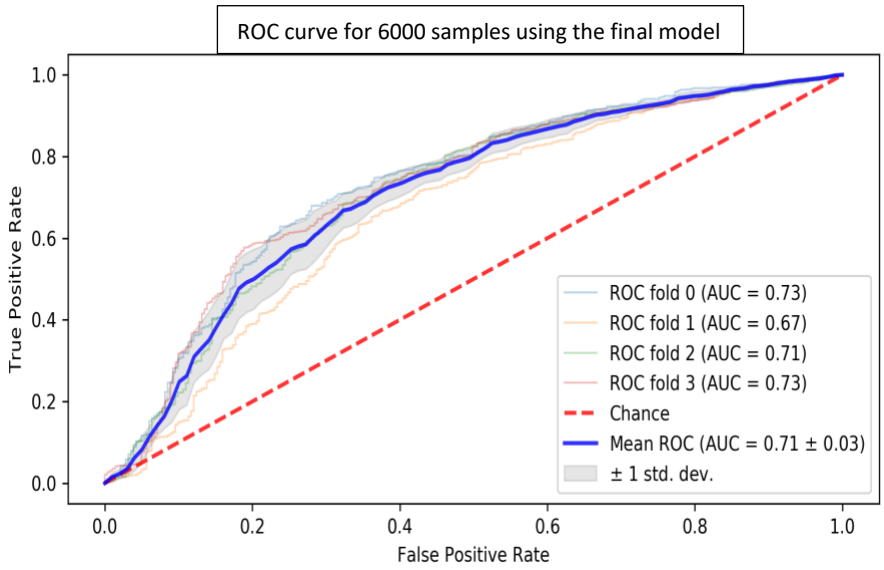
    - As we can see from the above table the training and testing accuracies and $F_{0.5}$ scores are very near to each other. In other words the difference between training and testing performance score is not very high.
    - Thus we can say that our model performs as good on the unseen test set as it performs on the seen training dataset.
    - Now if our model would have over fitted the data then the difference between the performance on training-set and testing-set would have been much higher. And since this is not the case, we can say that our model is not overfitting the data and it would work as good on any other unseen dataset.

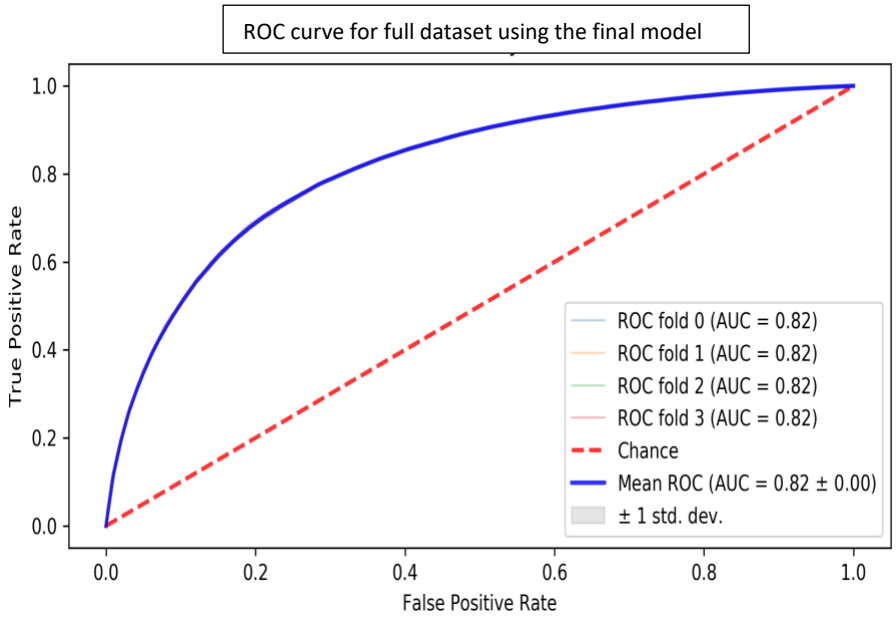
      Bias: The term bias tells us how much the average function that we would learn using different data sets would deviate from the target function that generated these datasets. This term is appropriately called bias as it measure how much our learning model is biased away from the target function.

      Variance: The term variance measures the variance in the final hypothesis. That is it measures what is the range of variation in different models when they are trained on different datasets.

    - Thus model's *variance is very less*. This is because variance is defined by how much farther different models are from the average model. And since in our case the model seems to be less affected by unseen data we can conclude that models on different dataset won't differ much from each other and also the average model, and hence give us a low variance.
    - The performance metrics generated by our final model are pretty good. This means that our model is not under fitting the data. This in turn means that the average function that we generate by training on different dataset would also not underfit any of those datasets. Thus the average function would not be too far away from the true target function. And in that case the *bias would also be low*.
    - Thus due to above two arguments we can say that our model has a good tradeoff between bias and variance with both of its values being low. And hence the E_{out} of this model would also be very low.

♦ The ROC characteristics of the final model is as follows:





➢ Interpretation: The best part about the ROC-AUC is that it is not affected by class imbalance. This thing is very important in our case since the positive classes are 4 times that of the negative classes.

- The reason of showing two ROC curves one on 6000 samples and one on the full dataset is to show that when we do K-fold cross validation AUC will have different values for those K-folds when we take a smaller dataset. However as the size of the dataset increases this variation decreases if our model has a good trade-off between bias an variance and their value is also low.
- Now as shown in the ROC curve above, the AUC of all the K-folds is same. This means that when a model is trained on different datasets their performance is the same. Moreover due to this fact the average model would also have values close to these different models and hence the *variance is very low.*

- Also as per the argument made for the performance metric section above, bias should also be very less.
- Thus this gives us another proof that bias and variance of this model has a low value and hence E_{out} = *bias* + *variance* would also be low.
- The value of AUC ranges from 0 to 1.
  - When AUC = 1 it means that model has the maximum separability i.e it can distinguish between all the class labels perfectly.
  - When AUC = 0 it means that it has negative separability which is nothing but when a model predicts positive class as negative class and vice versa.
  - When AUC = 0.5, it means that it has no separability, i.e it is a random classifier.
  - In our case the AUC value is pretty high(0.82) and this means that the separability of our model is also pretty high.
  - Thus AUC gives us an in-depth interpretation of how our model works well.

➢ Comparison with baseline model:

	Testing Accuracy	Testing $F_{0.5}$
Base Model(Naïve pred)	0.8078	0.8401
Final Model	0.8325	0.8704

- Interpretation: The accuracies and $F_{0.5}$ score for the final model are better as compared to the baseline model. The reasons for that are as follows:
  - Base line model naively predicts every review as a positive review whereas the final model that we trained captures the underlying distribution of the true target function.

➢ Collaborative Filtering results: Here we will consider two users, look at the products that they liked and then compare how the collaborative technique recommends based on the true ratings and based on the predicted sentiments. We will show the three highest rated products by each of the two customers and show the three highest ranked products recommended by the collaborative filter.

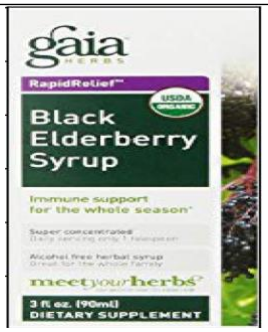
USER ID	NAME
AKHFY3VCNZL2W	Dan Huse
A3D0HMC6RQT0N0	J. Flood

- Recommendation analysis for user **'Dan Huse'**:
  - Products bought and rated the best by Dan Huse:

Laundry detergent| rating: 4

Deodorant| rating: 5

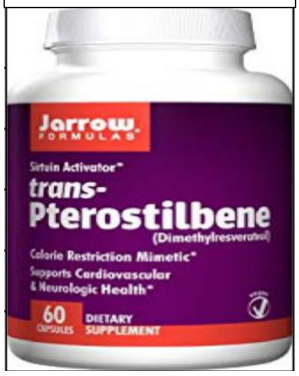
Dietary supplement| rating: 5



  - Products recommended to Dan Huse by collaborative filtering using true ratings.

Post shave balm|
mean rating: 4.37| rank = 1

Vitamin capsules
mean rating: 4.3| rank = 2



Tide Detergent and laundry materials
mean rating: 4.51 | rank = 3

  - Products recommended to Dan Huse by collaborative filtering using predicted sentiment of reviews.

Axe Shave Gel|
mean rating: 4.25| rank = 1

Vitamin pills(brain)| *mean*
rating: 4.01| rank = 2

Tide scented liquid detergent|
mean rating: 4.48| rank = 3

- Insights: The products bought by Dan Huse and rated the highest by him are as follows:
  - *Deodorant:* With rating 5 'Dan Huse' has rated it the highest
  - *Gaia dietary supplement:* This is also rated with a rating of 5 by 'Dan Huse'.
  - *Purex Ultra packs(Detergent pods):* These is the second highest rated product by 'Dan Huse'.

  Now our recommendations should be such that they are similar to these highest rated products and should also be similar to the recommendation made by collaborative filter on true labels.

  - Ture rating recommendations: As shown in the figure above the recommended products are described below. Here the rank represents priority of recommendations to 'Dan Huse' when the collaborative filter is applied on the basis of true ratings.

    - *Post Shave balm(rank - 1):* The mean rating for this is 4.37. Also since 'Dan' purchased a deodorant and gave it the highest rating, recommending this with highest rank makes sense.
    - *Jarrow calorie reduction capsules(rank - 2)*: The mean rating of this is 4.3 and again since 'Dan' actually purchased a dietary supplement and gave it the highest rating recommending this product with second priority makes sense.
    - *Laundry set by Tide(rank – 3)*: With a mean rating of 4.51, this product is recommended with the third priority because 'Dan' bought a detergent pod and gave it the second highest rating.

  - Predicted Sentiment recommendation: These are the recommendations made on the basis of the sentiments of the reviews predicted by our model. This means that collaborative filter is applied on the basis of the predicted sentiments.

    - *Axe shave Gel(rank - 1):* The mean rating for this is 4.25. Since 'Dan' purchased body grooming products and gave it the highest rating, recommending Axe shave Gel makes sense. Moreover this product is similar to the post shave balm recommended by the filtering on true recommendations.
    - *Jarrow brain cognition capsules(rank - 2)*: The mean rating of this is 4.01 and again since 'Dan' actually purchased a dietary supplement and gave it the highest rating recommending this product with second priority makes sense. This is also similar to the calorie reduction capsules from Jarrow with rank 2 recommended by the filtering on true recommendations.
    - *Tide Liquid Detergent (rank – 3)*: With a mean rating of 4.48, this product is recommended with the third priority because 'Dan' bought detergent pods and gave it the second highest rating. This liquid detergent from Tide is similar to the laundry set from Tide recommended by filtering on true recommendations.

  - Thus we can say that since the collaborative filtering on True ratings and predicted sentiments gives us results that are very similar to each other and also the user's choices, our recommendation engine works and in that sense our review prediction model also works.

- Recommendation analysis for user **'J. Flood'**:
  - Products bought and rated the best by J. Flood:

Panasonic shaver| rating: 4

Old Spice deo stick| rating: 5

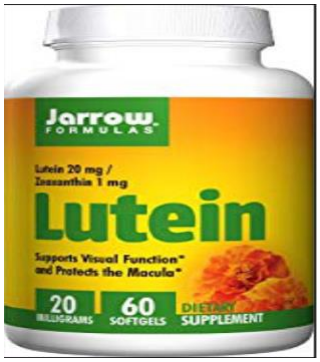
Blender bottled(Gym)| rating: 5

  - Products recommended to J. Flood by collaborative filtering using true ratings.

Remington cordless shaver
mean rating: 4.7| rank = 1

Secret Anti-Perspirant Deo
mean rating: 4.19| rank = 2

Jarrow weight reduction Pills
mean rating: 4.37 | rank = 3



  - Products recommended to J. Flood by collaborative filtering using predicted sentiment of reviews.

Remington cordless shaver
mean rating: 4.7| rank = 1

Secret Solid Anti-Perspirint
mean rating: 4.25| rank = 2

Jarrow weight reduction Pills
mean rating: 4.37 | rank = 3

- The insights in the case of user ‘J. Flood’ would be similar to those of ‘Dan Huse’. However, it should be noted that in the case of ‘J. Flood’ the recommendations from both the cases are much similar. This gives us another proof that our sentiment analysis model really works and because of that the recommendation engine works too.

- The most popular products are recommended when a new user signs up. Note that these are the products that have maximum mean rating. And since when a user signs up and has no history associated to him or her, it makes sense in recommending the most popular product to such a user.

  - Three most popular products recommended based on true ratings:

Energizer 9V Alkaline bat
mean rating: 4.9| rank = 1

Energizer Watch battery
mean rating: 4.9| rank = 2

Pocket Magnifier
mean rating: 4.8| rank = 3

  - Three most popular products recommended based on predicted sentiments of reviews:

Energizer 9V Alkaline bat
mean rating: 4.9| rank = 1

Pocket Magnifier
mean rating: 4.8| rank = 2

Energizer Watch battery
mean rating: 4.9| rank = 3

  - <u>Insight</u>: The popularity model for the true ratings and predicted sentiments give us very similar results. Now here we are calculating the mean rating on the basis of the weighted average keeping into context the final ratings or the predicted sentiments. Now since our model has a very high $F_{0.5}$ score, this means that the precision is also good and hence the total number of True positives is also very high. Due to this reason the weighted average on the basis of true ratings would be very close

to the weighted average on the basis of predicted sentiments, this in turn means that the popularity model would have similar predictions.

5. Contributions of each team member

Individual.

6. Summary and conclusions

Summary/Conclusions: The key findings from this project are as follows:

- We can create a good recommendation system on the basis of the sentiment of the users' reviews. This is evident from the Collaborative filtering results of section 4. As seen in that section, the recommendations made on the basis of true ratings are very similar to the recommendations made on the basis of review sentiments.
- If dataset is bigger than it already is, we could use faster algorithms like Naïve Bayes which can perform reasonably good in sentiment analysis tasks.
- The review classification examples from section 4 shows us that this model that we trained is good enough to classify any new reviews with a very high accuracy.

Future works: We can use the model and use the specific steps to then build a complete recommender system application with GUI, based on the sentiment of reviews.

7. References

Collaborative filtering: https://en.wikipedia.org/wiki/Collaborative_filtering