**ASSIGNMENT 6**

**Compare MNB and SVMs for Kaggle Sentiment Classification**

**Introduction:**

Textual data is available in abundance in social media, e-commerce websites, customer feedbacks which needs to be analyzed so that businesses can benefit from this or the ground truth is revealed. To be able to find patterns in text data, most common text mining algorithms used are Naïve Bayes and Support Vector Machines. Naïve Bayes makes a Naïve assumption of all the features to be independent and calculates probability of test data to be in each category, assigning the label which has the highest probability. For Naïve Bayes the input has restriction of being Term Frequency for Multinomial and Boolean for Bernoulli which sometimes can cause difficulties. SVM can have input which can be Boolean, Term Frequency or TFIDF which is an algorithm based on creating a hyperplane (straight line) to separate 2 categories and assign new data to 1 of the 2 categories separated by his hyperplane. In most of the text mining applications, the data is linearly separable and so using linear SVM is useful which is not the case with other applications.

The goal of this assignment is to compare Multinomial Naïve Bayes and SVM Algorithm performance on the Kaggle Movie Review Dataset using Scikit-learn in Python. To change different parameters such as min\_df, different vectorization techniques, ngrams variation etc. to find out which algorithm works the best along with the hyper parameters that contribute to the success. It also aims to build best possible SVM algorithm to be tested by Kaggle Sentiment data by tuning the parameters.

**Data Description:**

The data used here is taken from Kaggle website which is divided into test and train data. The data is movie review dataset from Rotten Tomatoes. Train and test are in the .tsv format which was analyzed using Scikit-learn package in Python. The data in train dataset was 156060 rows and in test dataset 66292 rows. For test data the ground truth is not available in the dataset whereas for train data we have ground truth along Phrase Id, Sentence Id and the Phrase to be analyzed. In train data, sentences are divided into different phrases and each of those phrases being part of 1 sentence are analyzed to make algorithm learn better and see if it performs well on phrases rather than on sentences. For Task 1 and task 2, we have split the training data into 40:60 ratio where training will be done on 60% of the train data and testing/validation will be performed using the remaining 40% of the training data. For Task 3, I used whole of the training data for training and for testing used the test data provided by Kaggle whose ground truth is available with Kaggle. These 2 ways of using part or whole of training data would have different results.

**Method:**

The train and test data which is in .tsv format which are read into pandas in Python. From training data, Sentiment and Phrases values are extracted and is put into y and X variables. X represents the whole sentiment the total sentiments and x would represent a particular sentiment which is a sample from the whole sentiment X. Similarly, all the Labels are put in Y variable and a particular label is assigned to y variable which is the difference in annotation. For task 1 and 2, train data is split into 60:40 ratio using test\_size parameter used to determine the % of the data to be used for testing purpose out of the whole data. Here we use test\_size to be 0.40 so we have testing data 40% and training data 60% of the total training data. Random\_state parameter is set to guarantee the same set of values in train and test data every time we reload the data. After splitting the train data, we have 93636 rows in training data and 62424 rows in test data. In training and test data, labels along with the count of rows where this label occurs is calculated to check if the dataset is balanced or skewed. In this dataset, there are 5 labels from 0 to 4 for very negative to very positive. For each category, number of phrases if almost same, the dataset is called balanced. But as in this case, the greatest number of phrases are neutral assigned label of 2 (47718 in total), so the data is skewed.

For Task 1, I used Unigrams as mentioned in the requirements, along with various options for vectorizations such as using Boolean, Term Frequency, TFIDF, min\_df, stop\_words, lowercase combinations of which is to be tuned for best performance for both MNB and SVM. For Task 2, I used Bigrams as mentioned in the requirements, keeping the other vectorization parameters same as in Task 1 and then comparing MNB and SVM. For Task 3, no splitting of training data into 60,40 ratios is done instead whole 100% of the train data is used to train SVM model with best parameters tuned. Testing will be done on Kaggle test data and cross-validation is used to test the performance along with Kaggle accuracy prediction. The input to MNB is restricted to only Term Frequency but for SVM any input can be used such as Boolean, TF, TFIDF without any compulsion on the input type.

Random guess for the baseline to test performance is not suitable in this case as the data is very heavily skewed. Majority vote would be used to decide the baseline as the dataset is skewed. The baseline for the accuracy testing would be around 51% based on Majority Vote.

The vectorization would be done by fitting and transforming the training data. Fitting creates bag of unique words called vocabulary as tokens whereas transform converts each document into vector based on the vocabulary. For testing dataset, I only used Transform so that the vocab doesn’t include even the unseen data from testing data causing it to memorize all the tokens and thereby causing overfitting.

**Results:**

**Task 1:**

For Task 1, Unigram MNB and Unigram SVM model is built by varying other parameters and then compared.

For Vectorization of MNB model, other parameters which could influence the results are min\_df, stop\_words removal or not removal, lowercase tokens or not, as we have fixed input which is Term Frequency for MNB. The choice of the values of these parameters didn’t only account for the highest accuracy but also other parameters such as Precision, Recall, F1 score. Also, the most positive and negative words would change somewhat for different values of these parameters.

For SVM Unigram Model, there was no compulsion of using Term Frequency which makes it flexible to even choose TFIDF, Boolean vectorizers if needed. All the other parameters are tuned same as in case of MNB model.

MNB Unigram Model:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Min\_df=3  No Stop Words, lowercase tokens | Min\_df=4  No Stop Words, lowercase tokens | Min\_df=5  No Stop Words, lowercase tokens | Min\_df=5  No lowercase tokens, No stop words | Min\_df=5  Not Remove Stop Words,  Lowercase tokens |
| Accuracy | 0.6048 | 0.6056 | 0.6064 | 0.6050 | 0.6069 |
| Precision | 0.52,0.59 | 0.52,0.59 | 0.53,0.59 | 0.52,0.59 | 0.52,0.59 |
| Recall | 0.44,0.60 | 0.44,0.61 | 0.44,0.61 | 0.44,0.61 | 0.43,0.61 |
| F1 Score | 0.47,0.59 | 0.47,0.59 | 0.47,0.59 | 0.47,0.59 | 0.46,0.59 |

MNB Vectorization Techniques Comparison for Task 1 using Unigram and TF

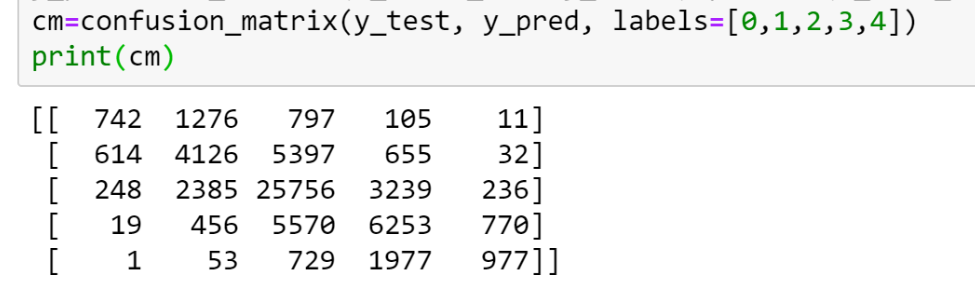
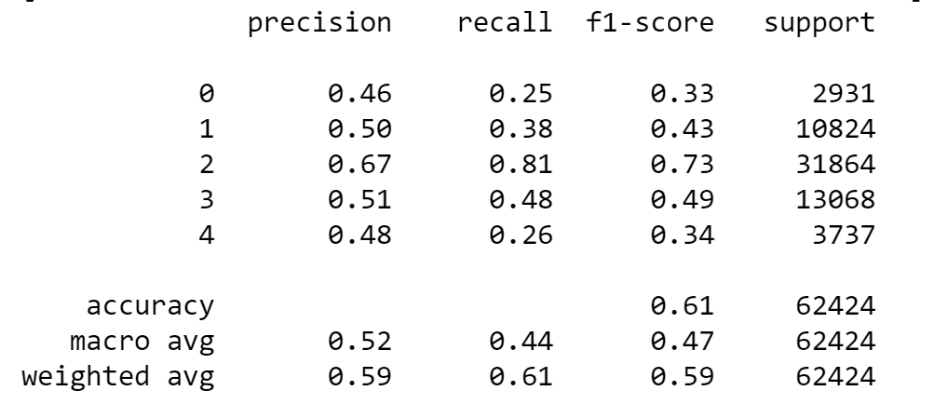
The above table compares various techniques used for vectorization including changing the document frequency, keeping or removing stop words, keeping the tokens lowercase or not for MNB with Unigram and Term Frequency as the input. The values in Precision, Recall and F1 score are macro avg, weighted average for the metrics. It can be seen from the table that min\_df of 5 along with lowercase tokens and removing stop words would give the best results. For not removing stop words, the accuracy increased slightly as seen from the table, but this causes words such as as, in, it, that, is, of etc. to appear in the very negative and very positive category which is of no use. Keeping the tokens lowercase is a way to avoid duplicate/repeated words in the text which is clear by looking at the numbers. For lowercase, the bag of words had 11967 words in training data whereas for no lowercase the bag size increased to 12763. So, it would be best to use lowercase tokens. Also removing stop words in English makes sense here as the words such as of, the, and, that appear in the very positive and negative criteria which would lead to model not learning words it should learn. Accuracy is not best suitable to look for this data as the data is not balanced so the values of other parameters needs to be looked at especially F1 score and Recall.

The top 10 positive and negative words for MNB Unigram with TF, min\_df=5, stop\_words=’English’, lowercase are as follows.

Most Positive: Moving, Beautiful, Beautifully, Powerful, Solid, Touching, Gorgeous, Excellent, Wonderful, Rich

Most Negative: Worst, Bad, Stupid, Worse, Contrived, Unfunny, Awful, Poorly, Waste, Pathetic

For Neural category, the model has a very high accuracy, precision, recall and F1 score as the data is skewed and more than 50% of the phrases are in the neutral category making it easily predict for the model. Precision Recall and F1 score for most negative (0) and most positive (4) were the lowest among the 5 categories as the number of phrases in these categories were the least. The model does well for neutral category but for most negative and most positive words it faces difficulties more than neutral category. The confusion matrix along with other metrics for MNB Unigram model are shown below.

F1 score and Recall are more important here as the dataset is skewed not balanced for accuracy to be considered as 1st priority. Neutral category is easy for model but most positive and most negative are difficult ones.

SVM Unigram Model:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Min\_df=5  No Stop Words, lowercase tokens, C=1, TF | Min\_df=5  No Stop Words, lowercase tokens, C=1, TFIDF | Min\_df=4  No Stop Words, lowercase tokens, C=1, TFIDF | Min\_df=3  lowercase tokens, No stop words, C=1, TFIDF | Min\_df=3  Not Remove Stop Words,  Lowercase tokens, C=1, TFIDF | Min\_df=3,  No Stop Words, lowercase tokens, C=1, Boolean |
| Accuracy | 0.6238 | 0.6254 | 0.6267 | 0.6272 | 0.6358 | 0.6241 |
| Precision | 0.55,0.60 | 0.55,0.61 | 0.55,0.61 | 0.55,0.61 | 0.57,0.62 | 0.55,0.60 |
| Recall | 0.46,0.62 | 0.46,0.63 | 0.46,0.63 | 0.46,0.63 | 0.47,0.64 | 0.46,0.62 |
| F1 Score | 0.49,0.60 | 0.49,0.61 | 0.49,0.61 | 0.49,0.61 | 0.51,0.62 | 0.49,0.60 |

SVM Vectorization Techniques Comparison for Task 1 using Unigram

The above table compares various techniques used for vectorization including changing the document frequency, keeping or removing stop words, keeping the tokens lowercase or not, using Boolean, TF or TFIDF for SVM with Unigram. The values in Precision, Recall and F1 score are macro avg, weighted average for the metrics. It is evident that Min\_df of 3 with lowercase tokens would produce a good combination along with using TFIDF as the input. For not removing stop the accuracy and other metrics seem to be better but results in words which don’t make sense as most negative and positive words. Also, there is difference of only around 1% for keeping stop words so it would be better to remove stop words. Duplicate/Repeated words removal is achieved by using lowercase tokens and TFIDF gives better accuracy and other parameters so using TFIDF. Furthermore, TFIDF gives less weightage to most occurring words and more weightage for least occurring words it is more suitable. The bag of words has increased to 14113 instead of 11967 due to changing Minimum document frequency to 3. This combination performances well on all metrics along with best accuracy.

The top 10 positive and negative words for SVM Unigram with TFIDF, min\_df=3, stop\_words=’English’, lowercase, C=1 would be having somewhat difference. As in the case with SVM, n classifiers produce n combinations for One vs All approach for classification for SVM. For each category, we have most negative and not most negative words which is due to the SVM binary classification. So, most positive vs not most positive and most negative vs not very negative words are reported.

Very Negative Words: Disgusting, Disappointment, Stinker, Repugnant, Meaningless, Unbearable, Unwatchable, Distasteful, Worst, Worthless

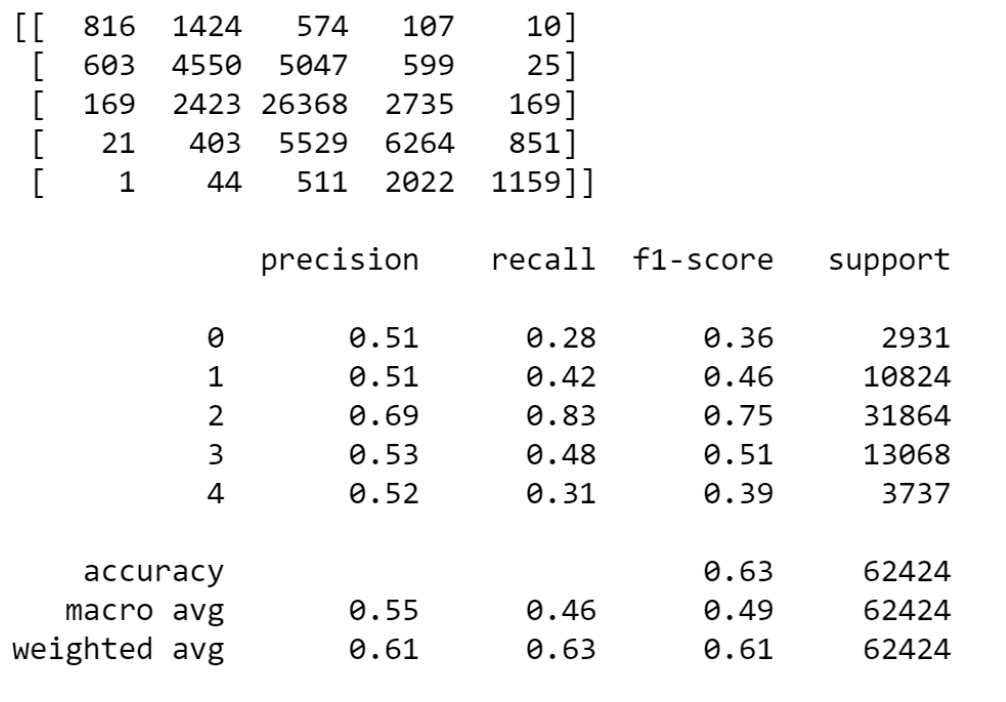
Not Very Negative Words: Readily, Engrossing, Remarkable, Loving, Rehashes, Modern, Innocence, Activity, Collar, Giddy

Very Positive Words: Perfection, Masterful, Glorious, Miraculous, Zings, Stunning, Magnificent, Masterfully, Masterpiece, Phenomenal

Not Very Positive Words: Nonchallenging, Sacrifices, Sketch, Bore, Falls, 12, Lame, Treated, Argue, Maintained

Thus, for each category of label we have 2 categories.

For Neutral category, the model has a very high accuracy, precision, recall and F1 score as the data is skewed and more than 50% of the phrases are in the neutral category making it easily predict for the model. Precision Recall and F1 score for most negative (0) and most positive (4) were the lowest among the 5 categories as the number of phrases in these categories were the least. Very Negative and Very positive category is difficult for the model whereas Neutral category is easy. Precision Recall and F1 score for neutral category was higher for SVM model as compared to MNB model. The confusion matrix along with other metrics for SVM Unigram model are shown below.



**Task 2:**

For Task 2, Bigram MNB and Bigram SVM model is built by only changing ngrams from Unigram to Bigrams keeping all the configurations of parameters and their values chosen to be same as required to see the effect of bigrams on the models built in Task 1 and if bigrams improve performance of the models.

MNB Bigram Model:

By adding Bigrams in MNB model built earlier in Task 1, the number of words in the bag (vocabulary) increased to 34579 from 11967. The position of words such as class was changed. Word class was earlier on position 1858 in Unigram model but in bigram model it got shifted by a large number of places to 5020.

The most positive and most negative words almost remained same without any noticeable difference.

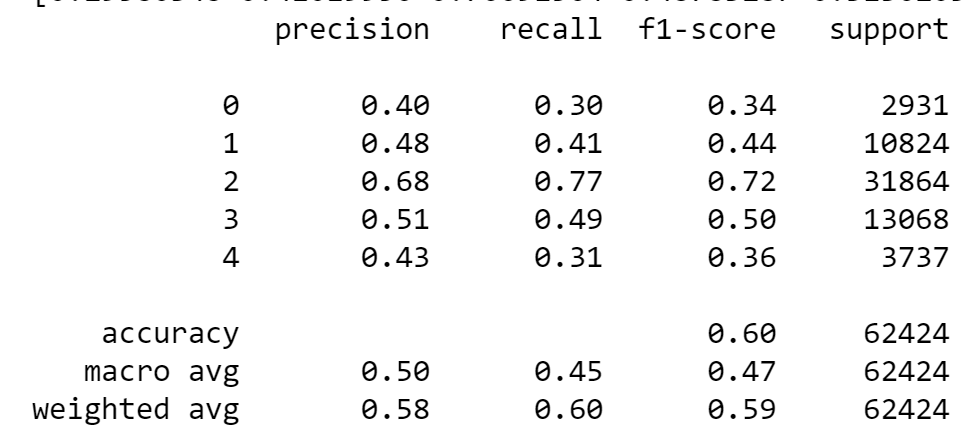
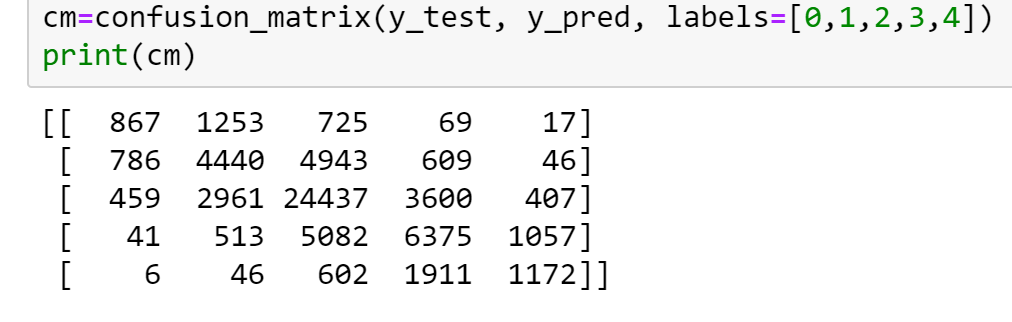
Most Positive Words: Moving, Beautiful, Beautifully, Powerful, Solid, Touching, Gorgeous, Excellent, Wonderful, Rich

Most Negative Words: Bad Movie, Waste, Poorly, Awful, Unfunny, Contrived, Worse, Stupid, Bad, Worst

One noticeable change in the most negative words was the inclusion of the word “bad movie” which is not present earlier. Bad movie replaced Pathetic word in the list in this model. This is not a significant thing as the dataset is of movie reviews with more than 90000 rows in which movie and bad combination is very common to occur. For this dataset, movie word can be used as a stop word to be removed.

Accuracy for bigrams model was reduced by 1% as compared to Unigram Model. Precision, Recall and F1 score reduced to a very negligible value not much to be noticeable. But still for the neutral category, the model performance in terms of precision, recall and F1 score increased making it easier to predict. For most positive and most negative category also the metrics improved even though accuracy might have decreased by 1%, making it a bit easier to predict these 2 categories using Bigrams.

For an unbalanced dataset like this one, F1 score and Recall are more important to consider than accuracy. So, for this reason, this model performs better than Unigram model. This model was more helpful for MNB case. The Confusion Matrix and other metrics are shown below.



SVM Bigram Model:

By adding Bigrams in SVM model built earlier in Task 1, the number of words in the bag (vocabulary) increased to 54449 from 14113. The position of words such as class was changed. Word class was earlier on position 2207 in Unigram model but in bigram model it got shifted by a large number of places to 7738.

Also, words such as “class wilde” and “chilling affecting” appeared which is a combination of words class, wilde, chilling, affecting. This is not as much effective information as bigrams would bring in.

For most positive and negative words, we would have most positive vs not most positive and most negative vs not most negative due to One vs All approach as discussed earlier in SVM.

Most Negative Words: Disappointment, Waste, Disgusting, Garbage, Distasteful, Unwatchable, Repugnant, Unbearable, Like Trapped, Entirely Witless

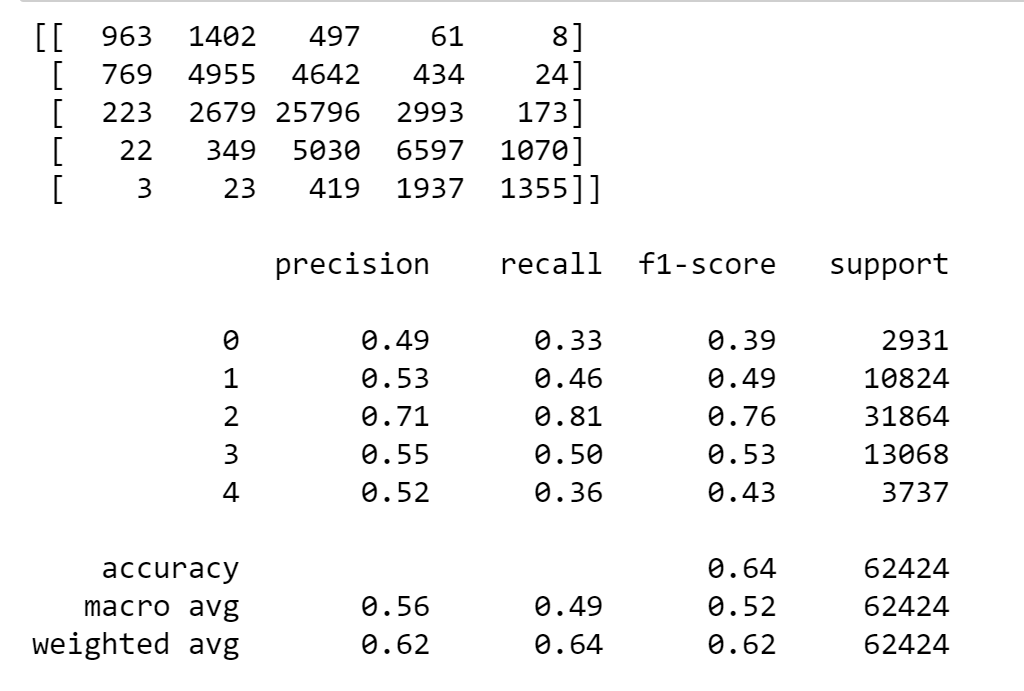
Not Most Negative Words: Completely Wreaked, Got Huge, Going Really, Ca Actor, Just Like, Movie Way, Way Wo, Good Good, Mind Ugly, Man Garbage

Most Positive Words: Excellent, Masterpiece, Brilliant, Amazing, Perfection, Phenomenal, Thrill Ride, Magnificent, Terrific, Best War

Not Most Positive Words: Satisfying Movie, Say Unburdened, Strong Script, Funny Uplifting, Argue, Brilliant Piece, Thanks Actors, Bore, Able Appreciate, Enigma.

For Bigrams in SVM, the words do make sense such as Thrill ride, Entirely Witless, Best War, Satisfying Movie. Adding bigrams does make a difference in SVM model which is seen from the most positive and most negative words.

Overall the accuracy of this model increased by around 1.5%. Precision, Recall and F1 score for the most positive and most negative categories is better using SVM with Bigrams rather than Unigrams. This model is making the task of predicting the difficult categories most negative and most positive easier by somewhat margin along with making it easier for neutral category also. Recall and F1 score being more important make this model better one as compared to the Unigram one. The confusion matrix and other metrics are shown below for this model.



**Task 3:**

For this task, whole training data of Kaggle Movie Review was used for training purpose rather than splitting the data into test and train. The model after being trained was tested on test data from Kaggle and predictions were uploaded to get the accuracy score from Kaggle.

For this model, various parameters in SVM were tuned before utilizing the model for testing the performance on test data. This is the first and foremost aim of this task. 2 Metrics were used for evaluation of model being built. Cross-validation score and the Kaggle Test Data Accuracy Score.

To start with, I used Unigrams with TFIDF by using lowercase tokens as the base choice which was derived from Task 1 where these parameters were varied and tested for best combination. But, here in this case, we are using whole dataset rather than just 60% of it for training, so I again tested some of the combinations and found that TFIDF gave best performance with lowercase tokens. Min\_df was changed along with Unigram changed to Bigrams and removing and keeping stop words to see the changes in cross-validation score. C was initially set to 1 but then it was also changed to see the best combination which produces best cross-validation score and Kaggle Test data accuracy score. The table below takes C=1 for combinations for now.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Unigram, min\_df=5, TFIDF, lowercase tokens, Remove Stop Words | Unigram, min\_df=4, TFIDF, lowercase tokens, Remove Stop Words | Unigram, min\_df=3, TFIDF, lowercase tokens, Remove Stop Words | Bigram, min\_df=5, TFIDF, lowercase tokens, Remove Stop Words | Bigram, min\_df=4, TFIDF, lowercase tokens, Remove Stop Words | Bigram, min\_df=3,  TFIDF, lowercase tokens, Not Remove Stop Words |
| Cross-validation Score | 0.5744 | 0.5739 | 0.5727 | 0.5667 | 0.5662 | 0.5781 |

SVM with different combinations for Vectorization keeping C=1 constant

By choosing the best combination where C=1 is Bigrams, min\_df=3, without removing stop words using lowercase tokens. Min\_df=3 was selected because the Kaggle sentiment score came out to be more for this combination which I used earlier in Task 2 and Task 1 as well.

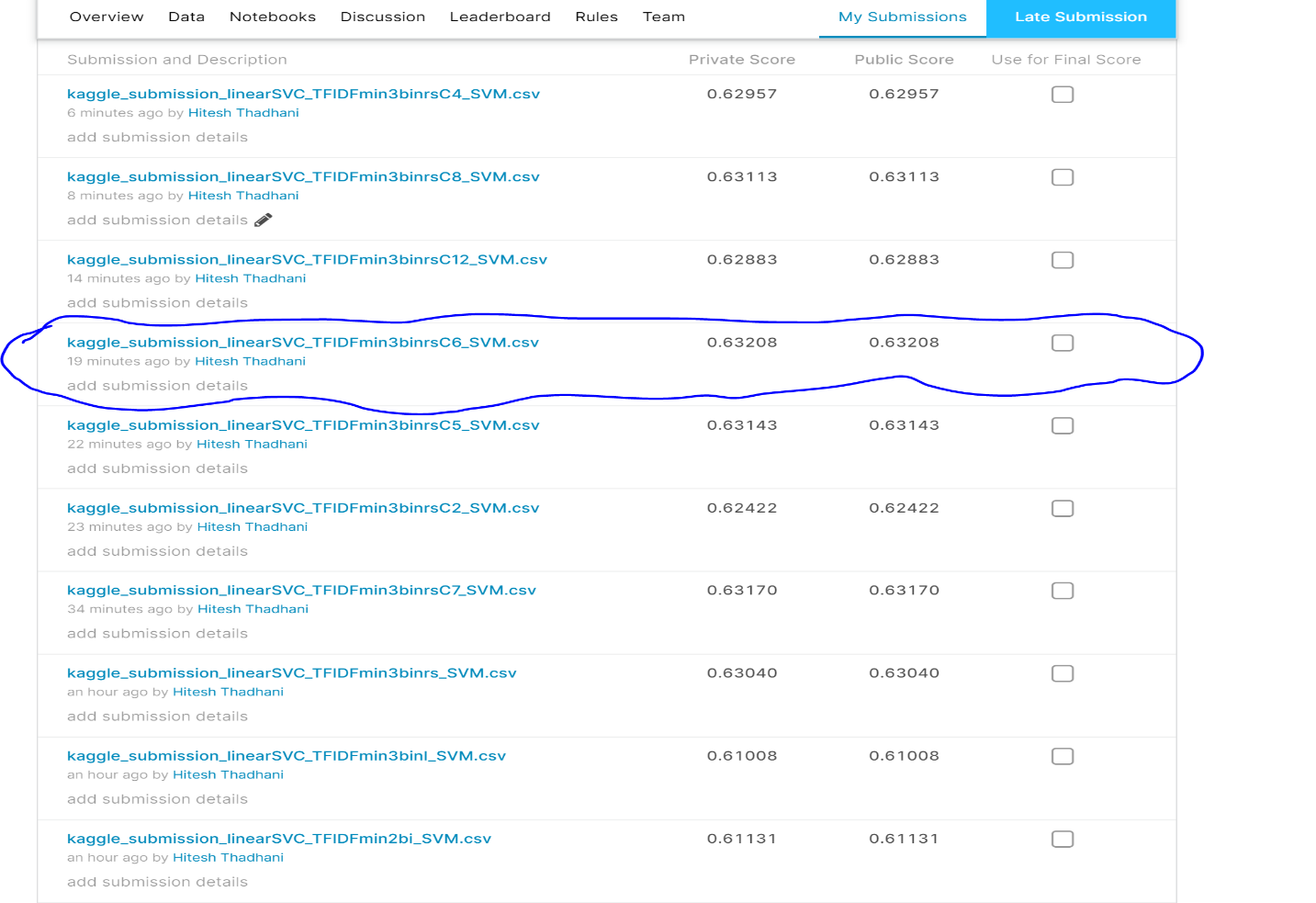
Now, for C value, which is penalty applied for misclassification on SVM model, I varied the value from 1.2 to 0.2 taking cross-validation and Kaggle Sentiment Accuracy Score in consideration. The table below shows the tuning of value of C to get the most Kaggle Sentiment Accuracy Score. By changing the value of C, the Kaggle score on test data changed. I tried to use TF as an experiment to see the effect rather than using TFIDF as vectorization method. The Kaggle sentiment score dropped to the lowest value of 0.62001 keeping all the other parameters values same. The table below shows how changing the value of C affects Cross-validation and Kaggle score and which is the tuned value of C to use to get the maximum Kaggle score.

|  |  |  |
| --- | --- | --- |
|  | Kaggle Sentiment Accuracy Score | Cross-Validation Score |
| C=1 | 0.63040 | 0.5781 |
| C=0.2 | 0.62422 | 0.5889 |
| C=0.4 | 0.62957 | 0.5894 |
| C=0.5 | 0.63143 | 0.5878 |
| **C=0.6** | **0.63208** | 0.5862 |
| C=0.7 | 0.63170 | 0.5844 |
| C=0.8 | 0.63113 | 0.5820 |
| C=1.2 | 0.62883 | 0.5741 |

SVM Model C value Tuning

From the table, it is evident that value of C=0.6 is the optimized one for Kaggle Accuracy Score giving the maximum accuracy on test data of 0.63208. Going lower than this value or higher reduces the accuracy of prediction. Even C=0.7 is a good value as the difference is very less from 0.63208.

C=0.6, min\_df=3, Not Removing Stop Words, Bigrams, TFIDF, lowercase tokens usage gave the best accuracy in terms of Kaggle Movie Review Test Data Prediction. The screenshot of Kaggle Submission showing the one with the best accuracy along with others is attached below.



**Conclusion:**

MNB and SVM are 2 different models working on textual data for sentimental analysis. MNB has restriction of requiring only TF as the input whereas SVM can take input as Boolean, TF or TFIDF. Through this assignment it is seen that Bigrams do contribute to the better MNB and SVM models being built. SVM with TFIDF works better than SVM with TF. SVM with lower Minimum DF was accurate than having Minimum DF to be a bit higher. Even stop words removal did not have any effect on Bigrams SVM model built in Task 3 as TFIDF will give very less weightage to the stop words. In Task 3 without removal of stop words better performance was achieved than by removing stop words in SVM. The value of C for getting best accuracy score came to be 0.6. TF with SVM usage depicted lowest Kaggle score of 0.62001, so TFIDF can be used with SVM almost majority of the time. F1 and Recall scores are more important to be considered when the data is skewed as in this case and the majority vote baseline to be used instead of random guess. All the models performed well above the baseline which was set by the neutral category with maximum number of reviews.