PREMIER UNIVERSITY CHITTAGONG

Department of Computer Science & Engineering

Report Submission.

Course Title: Artificial Intelligence Laboratory
Course Code: CSE 318
Assignment no.: 03
Name of Assignment: Pre-Processing and Feature Engineering with SVM

Submitted to:

Md. Ataur Rahman
Department of Computer Science and Engineering.
Premier University Chittagong.

Submitted by:

Name	: Sadia Chowdhury Dola
ID	: 1703310201465
Department	: C.S.E.
Batch	: 33rd
Semester	: 6th
Section	: C6B

Performance Date: 22-03-2021

Submission Date: 31-03-2021

Lab report: Pre-Processing and Feature Engineering with SVM

TABLE OF CONTENTS

1) <u>ABSTRACT:</u>

Machine learning (ML) is the study of computer algorithms that improve automatically through experience and by the use of data. Support Vector Machine is a classification of ML. Support Vector Machines is a very appealing and helpful instrument for classification and relapse. The paper begins with an outline of structural risk minimization (SRM) standard, and depicts the instrument of how to build SVM. For a two-class design acknowledgment issue, we talk about in detail the grouping component of SVM in three cases of linearly separable, linearly nonseparable and nonlinear. At long last, for nonlinear case, we give another capacity planning procedure: By picking a fitting portion work, the SVM can plan the low-dimensional information space into the high dimensional feature space, and develop an ideal isolating hyperplane with greatest edge in the component space. In this tutorial we present a brief introduction to SVM, and we discuss about SVM from published papers, workshop materials & material collected from books and material available online on the World Wide Web. In the beginning we try to define SVM and try to talk as why SVM, with a brief overview of statistical learning theory. The mathematical formulation of SVM is presented, and theory for the implementation of SVM is briefly discussed. Finally some conclusions on SVM and application areas are included. Support Vector Machines (SVMs) are competing with Neural Networks as tools for solving pattern recognition problems. This tutorial assumes you are familiar with concepts of Linear Algebra, real analysis and also understand the working of neural networks and have some background in AI.

2) <u>INTRODUCTION</u>:

Support Vector Machine (SVM) is one of the most popular Machine Learning Classifier. It falls under the category of supervised learning algorithms and uses the concept of Margin to classify between classes. It gives better accuracy than KNN, Decision Trees and Naive Bayes Classifier and hence is quite useful. Elements help provide concurrency, security, data integrity and uniform administration procedures.
I guess by now you would've accustomed yourself with linear regression and logistic regression algorithms. If not, I suggest you have a look at them before moving on to support vector machine. Support vector machine is another simple algorithm that every machine learning expert should have in his/her arsenal. Support vector machine is highly preferred by many as it produces significant accuracy with less computation power. Support Vector Machine, abbreviated as SVM can be used for both regression and classification tasks. But, it is widely used in classification objectives.

➢ <u>What is Support Vector Machines?</u>

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.

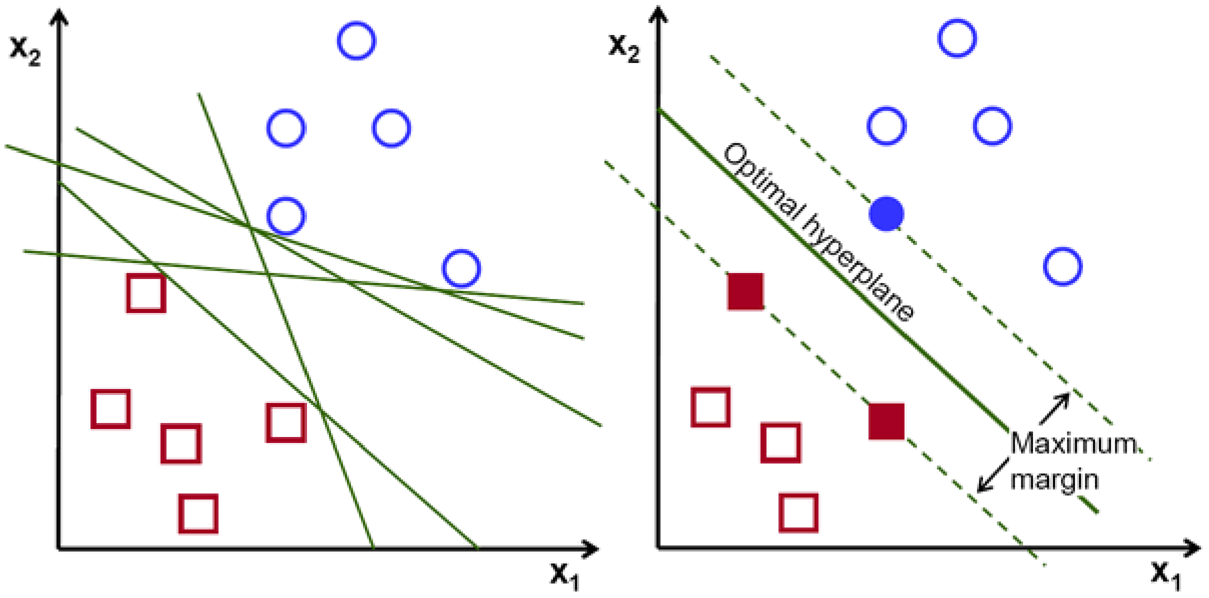


Figure: Possible hyperplanes.

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. The kernel performs its tasks, such as running processes, managing hardware devices such as the hard disk, and handling interrupts, in this protected kernel space.

3) WORKING PROCEDURE:

☞ All Function:

- **CountVectorizer ():** CountVectorizer counts the word frequencies. CountVectorizer is used to convert a collection of text documents to a vector of word counts.
  sklearn.feature_extraction.text.CountVectorizer:
  class *sklearn.feature_extraction.text.CountVectorizer(***, input='content', encoding='utf8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.int64'>***)*

- **TfidfVectorizer():**TF-IDF, which stands for term frequency –inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.
  sklearn.feature_extraction.text.TfidfVectorizer:
  class *sklearn.feature_extraction.text.TfidfVectorizer(***, input='content',encoding='utf8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, analyzer='word', stop_words=None, t**

oken_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), max_df=1.0, min_df=1, max_features=None,vocabulary=None,binary=False,dtype=<class'numpy.float64'>, norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=False*).*

- **pipeline():**pipeline function is used to make an object of classifier where I pass tfidf vectorizer or count vectorizer and multinomialNB at the same time .
  sklearn.pipeline.Pipeline:
  class **sklearn.pipeline.Pipeline(steps, *, memory=None, verbose=False)**

- **accuracy_score():**The accuracy_score() method determine the accuracy of the model based on the labels the model guessed and the labels of the given gold data.
  sklearn.metrics.accuracy_score:
  sklearn.metrics.accuracy_score(y_true, y_pred, *, normalize=True, sample_weight=None)

- **classification_report():** The classification_report() method determine the precision , recall , f1-score , support , accuracy , macro avg and weighted avg .
  sklearn.metrics.classification_report:
  sklearn.metrics.classification_report(y_true, y_pred, *, labels=None, target_names=None, sample_weight=None, digits=2, output_dict=False, zero_division='warn')

- **classifier.fit():** In classifier.fit() method I passed the 78% train data into the classifier .

- **classifier.predict():** In classfier.predict I passed 22% test document into the method and the model will predict the labels of the test document .

- **confusion_matrix():**This function is used to evaluate the performance of the model .

- **SVC():**
  sklearn.svm.SVC:
  class **sklearn.svm.SVC(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*).**

☞ Data:

For this assignment, we are using the dataset containing the Opinions about Corona_datasets. The reviews came from six classes (Neutral, Positive,
Extremely Negative, Negative and Extremely Positive). A statistic of the original training dataset is shown in Fig-a. The actual training data (‘train.tsv’) contains a total of 41157 reviews while the test/development data (‘test.tsv’) contains around 3798 labelled examples. Also encoding ‘utf-8’.

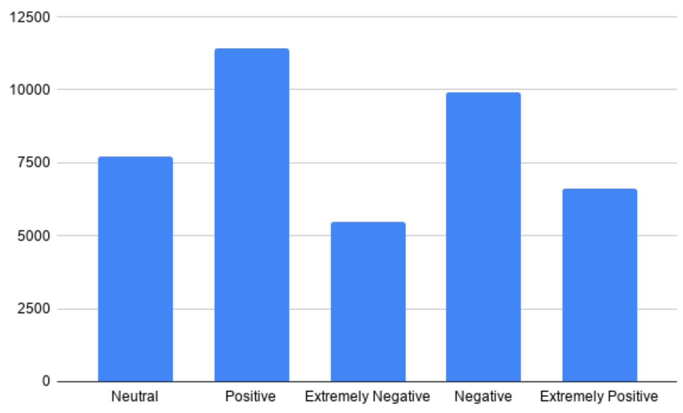


Figure: Statistics of the Training Data

☞ **Pre-Processing List:**

1. Remove URL,
2. Remove Hashtag,
3. Remove Whitespaces,
4. Tokenize Sentence,
5. Remove Punctuation,
6. Remove Stopwords,
7. Apply Stemmer,
8. Lemmatize Word.

4) SYSTEM OUTPUT:

◆ **Support Vector Machines:**

I am working on the multiclass classification with the dataset that we have mentioned in the assignment.

Default settings:

Runing a support vector machine with a linear kernel on the multiclass classification(cls = svm.SVC(kernel='linear', C=1.0). Using default settings and reporting results using the portion of the test/development set below.

Setting C:

Trying to change the value of the C parameter and On the other hand, lower C values generally lead to more support vectors, which may increase prediction time. Therefore, lowering the value of C involves a trade-off between fitting time and prediction time. That's why I set the c value is default 0.1 and also used different value but result is not satisfactory.

Using a non-linear kernel:

Using a non-linear kernel (**rbf/sigmoid/liner/poly** etc.) kernel. I also have to specify a gamma parameter, in addition to C:

cls = svm.SVC(kernel='rbf', gamma=0.7, C=1.0)

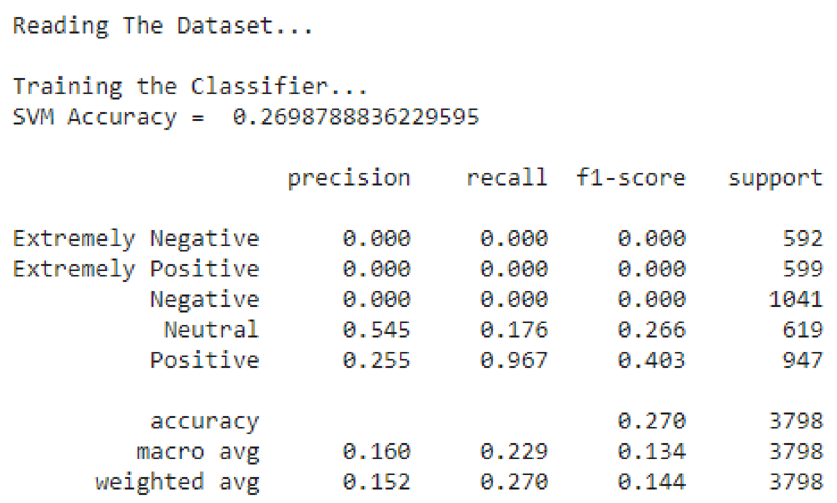
I can check information on the gamma value is = 0.1, 0.7 etc. (and C = 1.0) parameter here. Experiment with changing these values, and I reported with screenshot what I observe. It has been claimed that **rbf/sigmoid** kernels are great for text classification, and are normally outperformed by **linear** kernels.

- **Tfidf Vectorizer :**

When Tfidf = True,

No ngrams _range,
SVC(kernel='linear'),
pre_processing(remove_url, remove_whitespaces).

The result is:

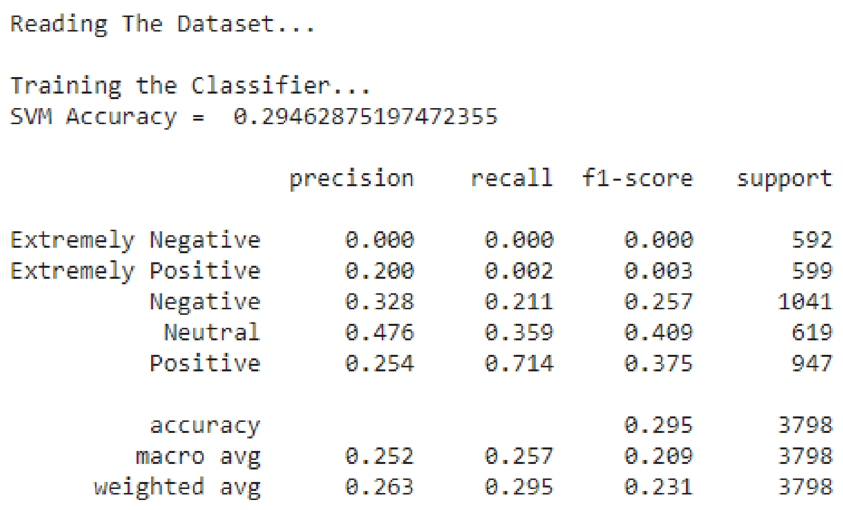


```
Reading The Dataset...

Training the Classifier...
SVM Accuracy =  0.2698788836229595

                    precision    recall  f1-score   support

Extremely Negative      0.000     0.000     0.000       592
Extremely Positive      0.000     0.000     0.000       599
          Negative      0.000     0.000     0.000      1041
           Neutral      0.545     0.176     0.266       619
          Positive      0.255     0.967     0.403       947

          accuracy                          0.270      3798
         macro avg      0.160     0.229     0.134      3798
      weighted avg      0.152     0.270     0.144      3798
```

Figure: 'Linear' kernel

Another example, pre_processing (remove_url, tokenize_sentence, remove_stopwords).

The result is:

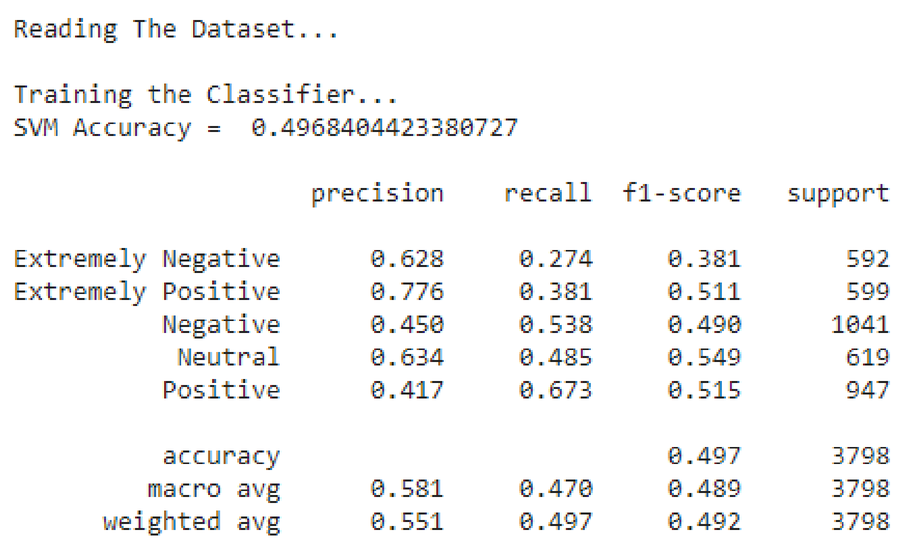


```
Reading The Dataset...

Training the Classifier...
SVM Accuracy =  0.29462875197472355

                    precision    recall  f1-score   support

Extremely Negative      0.000     0.000     0.000       592
Extremely Positive      0.200     0.002     0.003       599
          Negative      0.328     0.211     0.257      1041
           Neutral      0.476     0.359     0.409       619
          Positive      0.254     0.714     0.375       947

          accuracy                          0.295      3798
         macro avg      0.252     0.257     0.209      3798
      weighted avg      0.263     0.295     0.231      3798
```

Figure: 'Liner' kernel

- **Tfidf Vectorizer :**
  When Tfidf = True,
  TfidfVectorizer(preprocessor = identity, lowercase=True, analyzer='char', ngram_range = (3, 7)),
  SVC(C=1.0, kernel='rbf'),
  pre_processing(remove_url,remove_hashtag,remove_punctuation,apply_stemmer,tokenize_sentence, remove_stopwords).

The result is:

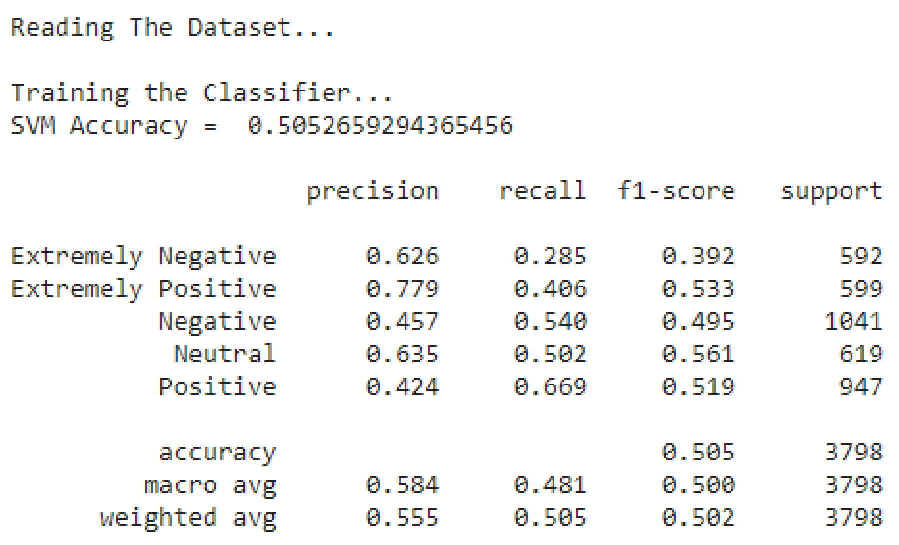


```
Reading The Dataset...

Training the Classifier...
SVM Accuracy =  0.4968404423380727

                    precision    recall  f1-score   support

Extremely Negative      0.628     0.274     0.381       592
Extremely Positive      0.776     0.381     0.511       599
          Negative      0.450     0.538     0.490      1041
           Neutral      0.634     0.485     0.549       619
          Positive      0.417     0.673     0.515       947

          accuracy                          0.497      3798
         macro avg      0.581     0.470     0.489      3798
      weighted avg      0.551     0.497     0.492      3798
```

Figure: ‘rbf’ kernel, C=1.0

- **Tfidf Vectorizer :**
  When Tfidf = True,
  TfidfVectorizer(preprocessor = identity, lowercase=True, analyzer='char', ngram_range = (3, 7)),
  SVC(C=1.0, gamma=0.7, kernel='rbf'),
  pre_processing(remove_url,remove_hashtag,remove_punctuation,apply_stemmer,tokenize_sentence, remove_stopwords).

The result is:

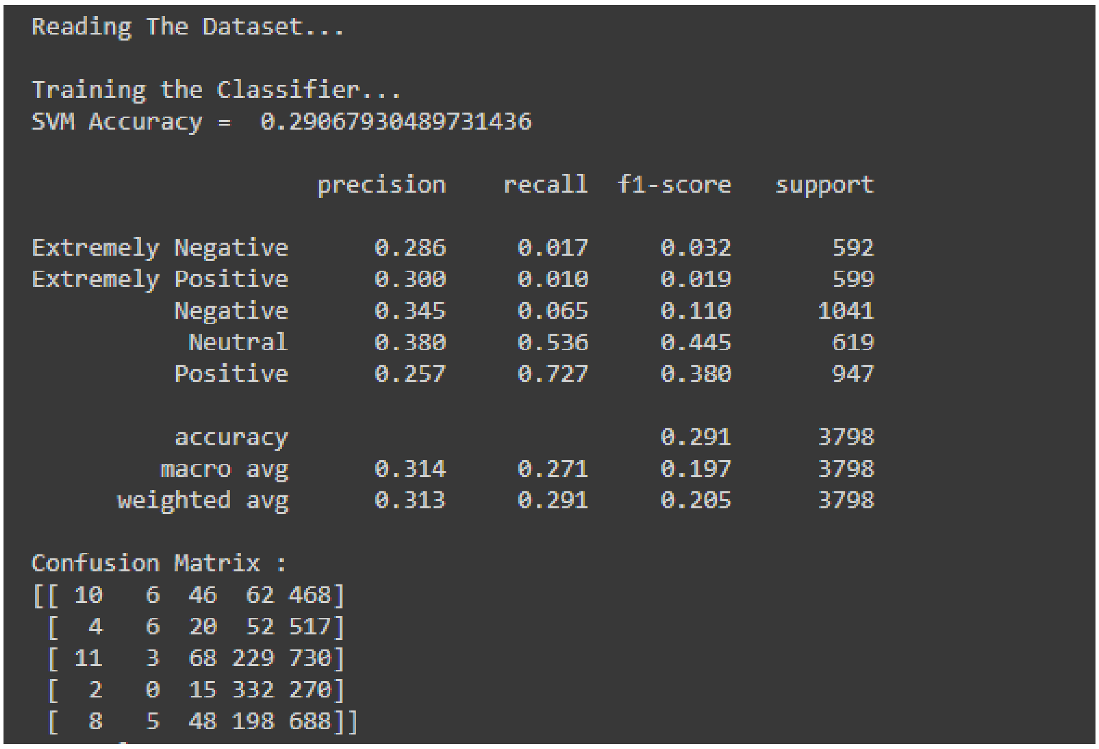


```
Reading The Dataset...

Training the Classifier...
SVM Accuracy =  0.5052659294365456

                    precision    recall  f1-score   support

Extremely Negative      0.626     0.285     0.392       592
Extremely Positive      0.779     0.406     0.533       599
          Negative      0.457     0.540     0.495      1041
           Neutral      0.635     0.502     0.561       619
          Positive      0.424     0.669     0.519       947

          accuracy                          0.505      3798
         macro avg      0.584     0.481     0.500      3798
      weighted avg      0.555     0.505     0.502      3798
```

Figure: C=1.0, gamma=0.7, kernel='rbf'

- **Counter Vectorizer :**
  When Tfidf = False,
  TfidfVectorizer(preprocessor = identity, lowercase=True, analyzer='char', ngram_range = (4, 8)),
  SVC(C=3.0, gamma=0.7, kernel='linear'),
  pre_processing(remove_url,remove_hashtag,remove_punctuation,apply_stemmer,tokenize_sentence, remove_stopwords, lemmatize_word),
  Confusion matrix

The result is: So bad

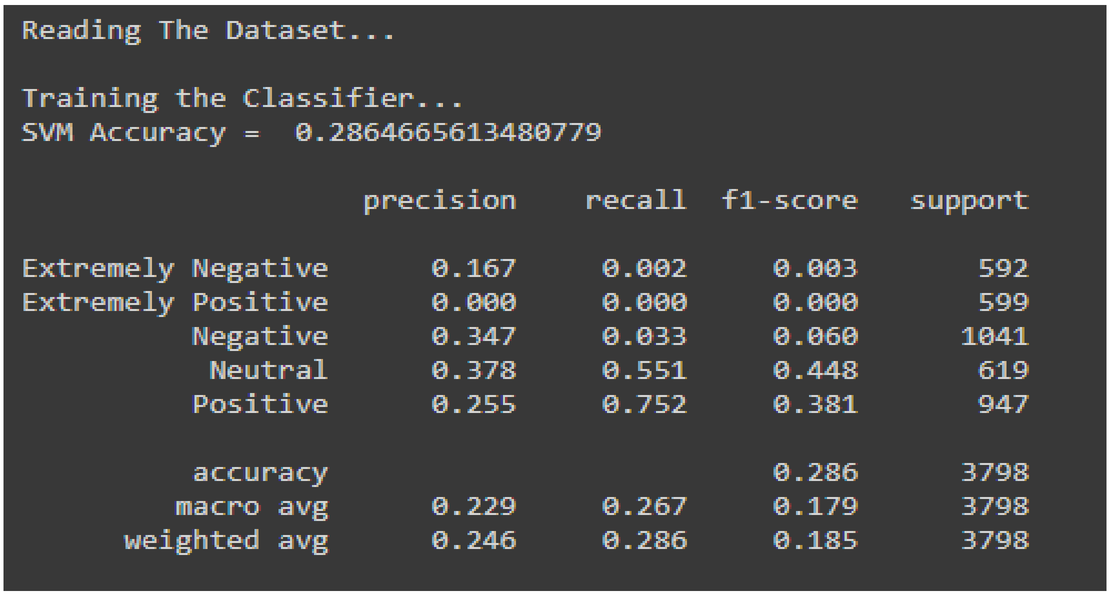


```
Reading The Dataset...

Training the Classifier...
SVM Accuracy =  0.29067930489731436

                    precision    recall  f1-score   support

Extremely Negative      0.286     0.017     0.032       592
Extremely Positive      0.300     0.010     0.019       599
          Negative      0.345     0.065     0.110      1041
           Neutral      0.380     0.536     0.445       619
          Positive      0.257     0.727     0.380       947

          accuracy                          0.291      3798
         macro avg      0.314     0.271     0.197      3798
      weighted avg      0.313     0.291     0.205      3798

Confusion Matrix :
[[ 10   6  46  62 468]
 [  4   6  20  52 517]
 [ 11   3  68 229 730]
 [  2   0  15 332 270]
 [  8   5  48 198 688]]
```

Figure: C=3.0, gamma=0.7, kernel='linear'

- **Counter Vectorizer :**
  When Tfidf = False,
  TfidfVectorizer(preprocessor = identity, lowercase=True, analyzer='char', ngram_range = (3, 3)),
  SVC(C=1.0, gamma=0.1, kernel='linear'),
  pre_processing(remove_url,emove_whitespaces,remove_punctuation,apply_stemmer, tokenize_sentence, remove_stopwords, lemmatize_word),
  Confusion matrix

The result is:



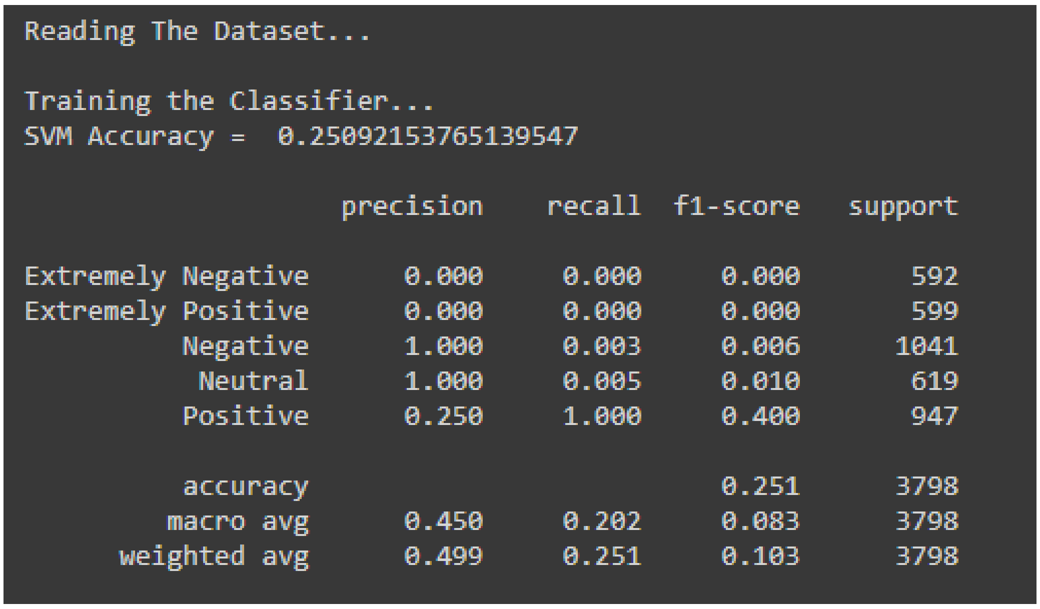
```
Reading The Dataset...

Training the Classifier...
SVM Accuracy =  0.2864665613480779

                    precision    recall  f1-score   support

Extremely Negative      0.167     0.002     0.003       592
Extremely Positive      0.000     0.000     0.000       599
          Negative      0.347     0.033     0.060      1041
           Neutral      0.378     0.551     0.448       619
          Positive      0.255     0.752     0.381       947

          accuracy                          0.286      3798
         macro avg      0.229     0.267     0.179      3798
      weighted avg      0.246     0.286     0.185      3798
```

Figure: C=1.0, gamma=0.1, kernel='linear'

◆ **Counter Vectorizer :**

When Tfidf = False,
TfidfVectorizer(preprocessor = identity, lowercase=True, analyzer='char', ngram_range = (6, 9)),
SVC(C=2.0, gamma=0.5, kernel='linear'),
pre_processing(remove_url, remove_punctuation,apply_stemmer,tokenize_sentence, remove_stopwords, lemmatize_word),
Confusion matrix

The result is:

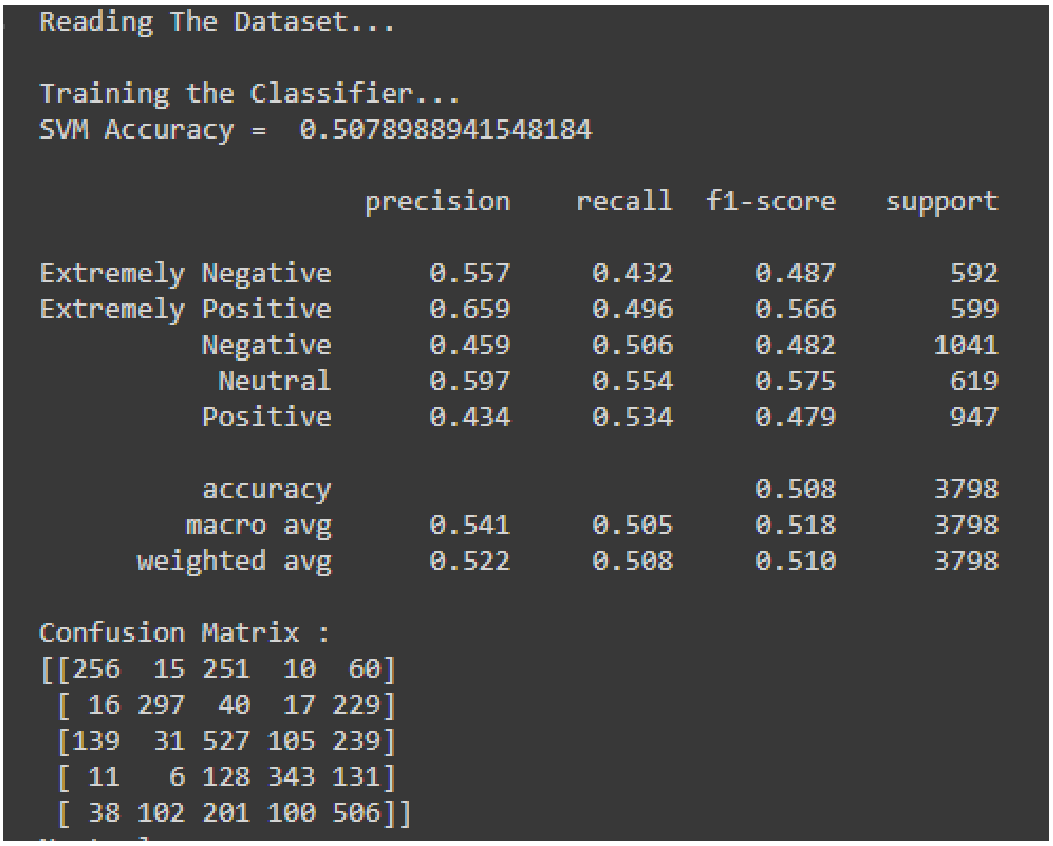


```
Reading The Dataset...

Training the Classifier...
SVM Accuracy =  0.25092153765139547

                    precision    recall  f1-score   support

Extremely Negative      0.000     0.000     0.000       592
Extremely Positive      0.000     0.000     0.000       599
          Negative      1.000     0.003     0.006      1041
           Neutral      1.000     0.005     0.010       619
          Positive      0.250     1.000     0.400       947

          accuracy                          0.251      3798
         macro avg      0.450     0.202     0.083      3798
      weighted avg      0.499     0.251     0.103      3798
```

Figure: C=2.0, gamma=0.5, kernel='rbf'

◆ **Tfidf Vectorizer :**
When Tfidf = True,
TfidfVectorizer(preprocessor = identity, lowercase=True, analyzer='char', ngram_range = (3, 7)),
SVC(C=1.0, gamma=0.1, kernel='rbf'),
pre_processing(remove_url,remove_hashtag,remove_punctuation,apply_stemmer,tokenize_sentence, lemmatize_word).

The result is:

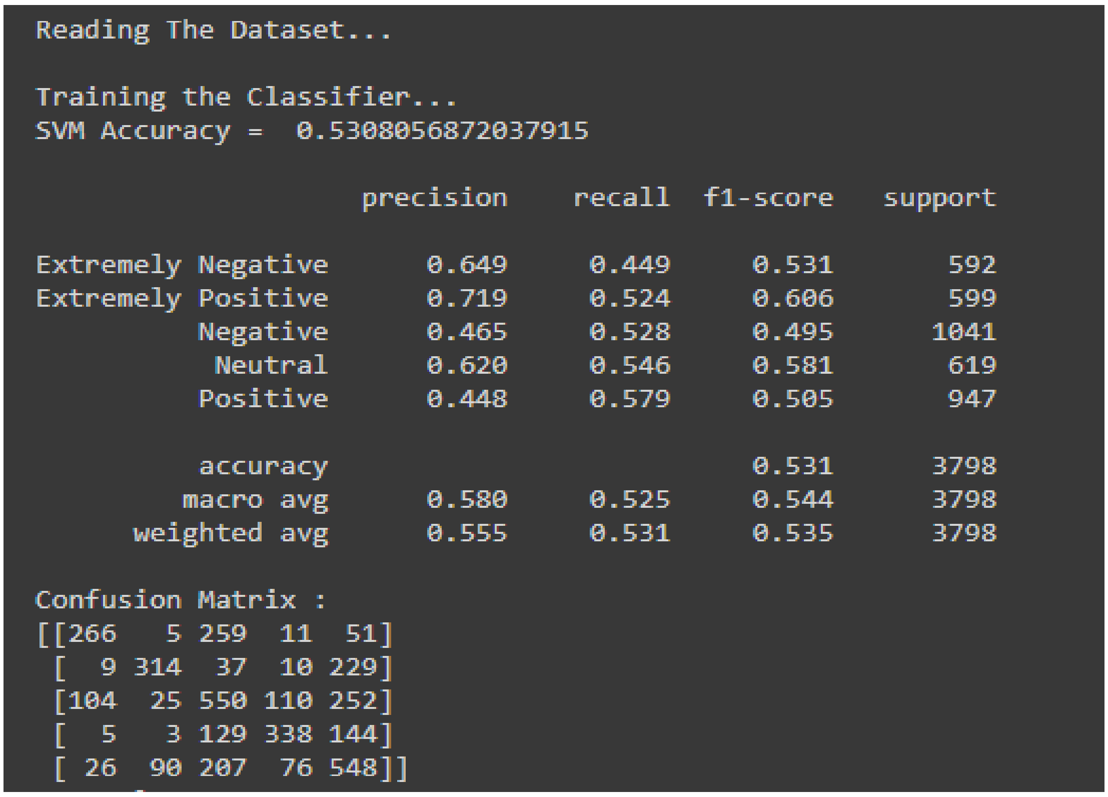


```
Reading The Dataset...

Training the Classifier...
SVM Accuracy =  0.5078988941548184

                    precision    recall  f1-score   support

Extremely Negative      0.557     0.432     0.487       592
Extremely Positive      0.659     0.496     0.566       599
          Negative      0.459     0.506     0.482      1041
           Neutral      0.597     0.554     0.575       619
          Positive      0.434     0.534     0.479       947

          accuracy                          0.508      3798
         macro avg      0.541     0.505     0.518      3798
      weighted avg      0.522     0.508     0.510      3798

Confusion Matrix :
[[256  15 251  10  60]
 [ 16 297  40  17 229]
 [139  31 527 105 239]
 [ 11   6 128 343 131]
 [ 38 102 201 100 506]]
```

Figure: C=1.0, gamma=0.1, kernel='rbf'

◆ **Tfidf Vectorizer :**
When Tfidf = True,
TfidfVectorizer(preprocessor = identity, lowercase=True, analyzer='char', ngram_range = (3, 3)),
SVC(C=1.0, gamma=0.6, kernel='linear),
pre_processing(remove_url,remove_hashtag,remove_punctuation,apply_stemmer,tokenize_sentence, lemmatize_word).

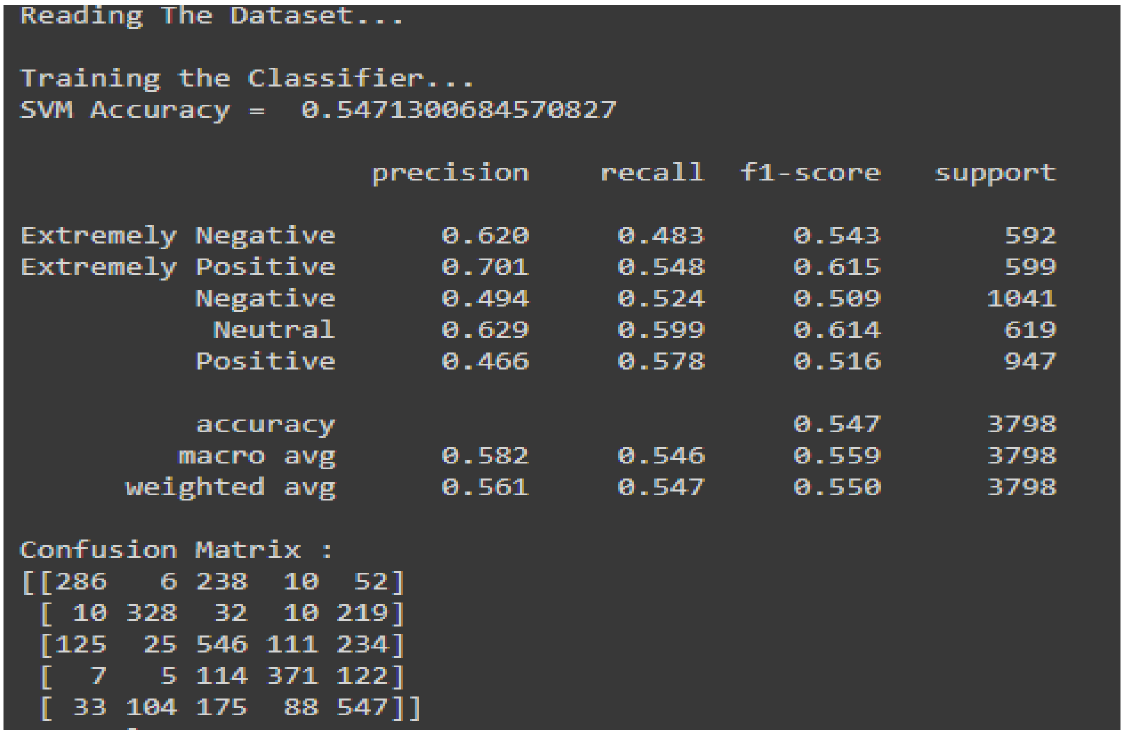
The result is: Best result



```
Reading The Dataset...

Training the Classifier...
SVM Accuracy =  0.5308056872037915

                    precision    recall  f1-score   support

Extremely Negative      0.649     0.449     0.531       592
Extremely Positive      0.719     0.524     0.606       599
          Negative      0.465     0.528     0.495      1041
           Neutral      0.620     0.546     0.581       619
          Positive      0.448     0.579     0.505       947

          accuracy                          0.531      3798
         macro avg      0.580     0.525     0.544      3798
      weighted avg      0.555     0.531     0.535      3798

Confusion Matrix :
[[266   5 259  11  51]
 [  9 314  37  10 229]
 [104  25 550 110 252]
 [  5   3 129 338 144]
 [ 26  90 207  76 548]]
```

Figure: C=1.0, gamma=0.6, kernel='linear

The result is: Best SVM Model



```
Reading The Dataset...

Training the Classifier...
SVM Accuracy =  0.5471300684570827

                    precision    recall  f1-score   support

Extremely Negative      0.620     0.483     0.543       592
Extremely Positive      0.701     0.548     0.615       599
          Negative      0.494     0.524     0.509      1041
           Neutral      0.629     0.599     0.614       619
          Positive      0.466     0.578     0.516       947

          accuracy                          0.547      3798
         macro avg      0.582     0.546     0.559      3798
      weighted avg      0.561     0.547     0.550      3798

Confusion Matrix :
[[286   6 238  10  52]
 [ 10 328  32  10 219]
 [125  25 546 111 234]
 [  7   5 114 371 122]
 [ 33 104 175  88 547]]
```

Figure: C=1.0, gamma=0.8, kernel='linear,ngram_range(2,4)

5) QUESTION ANSWER:

✓ **Explain why the C parameter is used for:**

The behavior of the model is very sensitive to the gamma parameter. If gamma is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent over fitting. When gamma is very small, the model is too constrained and cannot capture the complexity or “shape” of the data. It may make sense to use the smaller C values, since very high C values typically increase fitting time. So, I used C value is 0.1(which is default). And testing using different value of C.

✓ **Do you observe the same?**

cls = svm.SVC(kernel=’rbf’, gamma=0.7, C=1.0)

No, I observe the changing these values ‘linear’ kernels is improved than ‘rbf’ kernel.

✓ **Possibly n-grams (word/char) might help?**

ngram_range : tuple (min_n, max_n), default=(1, 1)

The lower and upper boundary of the range of n-values for different word n-grams or char n-grams to be extracted. All values of n such such that min_n <= n <= max_n will be used. For example an ngram_range of (1, 1) means only unigrams, (1, 2) means unigrams and bigrams, and (2, 2) means only bigrams. I used different ngram_range to getting better results like ngram_range of (3, 7), (5, 7) etc. But I must say when I used ngram_range of (3, 7) the value is 0.507, then I used ngram_range of (3, 3) the value is 0.547. I also used n-grams word and char. But I prefer char is best of the model.

✓ **Why you might thing the results have improved/degraded from the previous experiment(s)?**

It’s quite improved the previous experiment. The previous experiment value is 0.4 something but the experiment best value is 0.547. So I said that the result much better.

6) LIMITATION:

There are many limitations:

- Running the program, it take so much time.
- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

7) CONCLUSION:

In this assignment, corona_dataset is given .First the corona_dataset is tokenize and separate the labels and documents with a function call. Then, writing different pre-processing method to find out different function. Result find out for train.tsv and test.tsv. The text is big so it take more and more time to get output. Here we use two kind of vectorizer. One is tfidf Vectorizer and another is count vectorizer. I took tfidf value True or False and also took counter vectorizer value True or False so it take time to execute. When I took false value the accuracy sore is much better than True value.

Finally, this assignment has enough limitations, those can't create any problem in assignment. This assignment has been run successfully and performance of the system has been found satisfactory. Use of many function the accuracy, f1-score, recall, support, macro avg, weighted avg, confusion matrix etc, we can observed the score. Then the model gives a result by predicting the test labels and test docs or train labels and train docs for both binary and multiclass with using vectorizer and confusion matrix also.

8) REFERENCES:

☞ https://course.ccs.neu.edu/cs5100f11/resources/jakkula.pdf
☞ https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/
☞ https://towardsdatascience.com/a-brief-introduction-to-support-vector-machine-adf0f103a80f
☞ https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
☞ https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
☞ https://github.com/scikit-learn/scikit-learn/blob/95119c13a/sklearn/feature_extraction/text.py#L1524