Classifiers:

In this project, there are four classifiers generated in the order of

1. Bag of words model on Logistic Regression Classifier
2. Majority-voting Classifier
3. Spare Vector (Tf-idf) model on Logistic Regression Classifier
4. Dense Vector (Word2Vec) model on Logistic Regression Classifier

The Statistical Test results collected are:

1. Accuracy score of the predictions and the actual label.
2. Precision Score (macro) of the predictions and the actual label.
3. Recall Score (macro) of the predictions and the actual label.
4. F1 Score (macro) of the predictions and the actual label.

And the results are as showing the table below.

	Accuracy Score	Precision Score	Recall Score	F1 Score
Majority Voting	0.7948	0.1987	0.25	0.2214
Bag-of-word	0.7564	0.2883	0.2906	0.2824
Tf-idf	0.7939	0.2518	0.3164	0.2803
Word2Vec	0.7901	0.2271	0.2743	0.2450

Table 1.Statical Test Results

Description of Step 2

In step 2, two classifiers have been built, a logistic regression classifier using a bag of words, and a majority-voting classifier. To build the classifiers, the program will read in column F and Column I as “conment_text” and “comment_toxi” from the file “SFUcorpus.xlsx”. Then preprocess the input text by removing punctuation and lower case all the letters. Now, the documents are ready to be used to generate the bag of words model. Calling the “CountVectorizer” from the “Sklearn” package to create the vector for each document. The bag of words model has the same number of rows as the original number of documents, and for each row, it stores a vector that has all the unique vocabulary that has occurred in all documents. For instance, the bag of words model has 1043 rows and each row has 8116 elements, it corresponds to 1043 lines of comment text from the file, and 8116 unique words in all documents.

In order to do cross-validation on the Logistic Classifier, the program uses “K-Fold” to split the dataset into 10 fair shared portions. The Logistic Regression Classifier will be trained by 90 percent of the dataset, and be tested on the last 10 percent. This equivalent to using 9 Folds of data to train and 1 fold of data to test. The Cross-Validation will also run the process 10 times to make sure that every single portion of the data gets a chance to be the test set. Each iteration of the program will measure the statistical test results and save them into the corresponding list. The Statistic result will be calculated by getting the mean of all ten iterations.

The Majority-voting classifier will use the most common label, in this case, the label is ‘1’. Regarding the input, the Majority-voting classifier will always predict ‘1’. Get the statistical test result by comparing the predictions with the actual label. The statistical test is performing “macro averaging” which doesn't take label imbalance into account.

Bag of words VS. Majority-voting

According to the data in Table 1, the accuracy score of the Bag of words model is less than the Majority-voting model. The reason is the input dataset is unbalanced, which has too many ‘1’s, and so less ‘2’s, ‘3’s, and ‘4's. Even though all the predictions are predicted to ‘1’, it still has about 79 percent accuracy compared with the actual label. So using the accuracy score to compare the models on an unbalanced dataset does not give us enough useful information.

Although the accuracy is not informative, the precision score, recall score and f1 score could have some meaning to them. By the definition of each term, a higher precision score indicates less false positives a classifier gives. A higher recall score means fewer false negatives a classifier gives. And the F1 score is the average of the precision score and recall score. So based on the test results from Table 1, all three scores (precision, recall, F1) of the Bag of words model are higher than the Majority-voting model. This means the Bag of words model make less number of False-positive and False-negative predictions than the Majority-voting model. Thus, we conclude that the Bag of words model has an overall better performance than the Majority-voting model.

Description of Step 3

The two models for step 3 are the Tf-IDF model which is a Sparse Vector semantic representation, and the Word2Vec model which is a Dense Vector semantic representation. The Tf-IDF model is generated by the ‘TfidfVectorizer’ from the sklearn class. Transfer the documents from the input file to a Tf-IDF model according to the frequency of the word appear in the sentence. The Word2Vec model is generated based on the data from the pre-trained Google Word2Vec model. Since it is impossible to load all the data from the Google pre-trained model, the limit of most frequent words is set to be 100000. I experimented with load more words from the Google pre-trained model, the improvement is not significant. The Word2Vec model contains a vector for each document, in each vector, it will have 300 decimal numbers calculated by getting mean of all the words in the document. After finishing generate these two models, use the same cross-validation strategy as described in step 2 to evaluate the models on the statistical test results.

Tf-idf VS Word2Vec VS Bag of words

According to the data in Table 1, the Tf-IDF model and Word2Vec model have similar performance in terms of accuracy score, they both better than Bag of words model and Majority-voting model. The precession score, recall score and F1 score of both the Tf-IDF model and Bag of words model are higher than the Word2Vec model. That means the Word2Vec model gives more False positives and False negatives. The reason for this result could due to limited words loaded from the Google pre-trained model, and how the vector of each document to be calculated.

Though TF-IDF has the same accuracy score as the Word2Vec model, the precision and recall scores are not the same. The Tf-IDF model has a lower precision score than the Bag of words model, but higher recall score. This means compared to the Bag of words model, Tf-IDF gives more false negatives and less false positives. The reason could potentially be how other minority labels in the dataset. For example, the class ‘2’ of the Tf-IDF model has more false positives, then the precision score will be lower.

The scores are calculated using “macro averaging” instead of “micro averaging” is because of the imbalanced dataset. If the calculation us “micro averaging”, the class ‘1’ will impact the scores and make them identical. So since the macro averaging does not take label imbalance into account, it will give relatively more useful information than “micro averaging” scores.

Simple T-test:

Hypothesis:

The first model send into t-test, is better than the second model into t-test.

	T-value	P-value
TF-idf vs BOW	0.9098259661199047	0.3866315161974877
W2V vs BOW	0.7763615218826584	0.45745246544138285
TF-idf vs W2V	0.3583441536327849	0.7283471738391647

Analysis:

The P-value of all three comparison does greater than 0.10. So we can conclude that the differences will be meaningful. So the data in the Table 1 meaningful to make the comparison and analysis.

Step 5:

Question: How the Classification result will be affected, if we extracting more optimal feature from the input dataset?

Result:

	Accuracy Score	Precision Score	Recall Score	F1 Score
Majority Voting	0.7948	0.1987	0.25	0.2214
Bag-of-word	0.7776	0.3510	0.3301	0.3239
Tf-idf	0.7928	0.2323	0.2911	0.2582
Word2Vec	0.7833	0.2476	0.2908	0.2627

Table 2.Statical Test Results of optimal extractions

	Time
Original Text Normalization	25.71 seconds
Optimal Text Normalization	15.47 seconds

Table 3. Time to compute the result

Intuition:

The original pre-processing only do two operations to the features, lowering all the letters and remove all the punctuations. But after doing these operations, the features still contain many words like numbers, ‘that’, etc. How it will affect the classification, if we extract the most optimal features from the input data?

Experiment Setup:

Keep the original pre-processing function to compare with the outcome of the new optimal pre-processing function. The first two steps in the new function will be the same with the original function, lower case all the letters and remove all the punctuations. Then replace the numbers with num, and remove the stopwords. Finally, stem the words in the documents. After executing these steps, I expected to have a

smaller size of features and all the words are useful. Use the new features to generate the same models and train the classifier. And obtain the result as shown in the table above.

Analysis:

1. **Bag of words Model**

I observe that the accuracy score of the Bag of words models slightly improved by almost 2 percent. And the precision score, recall score and F1 score improve significantly. As I expected before making the experiment, the size of the feature reduces to 5712. So for each vector in the Bag of words model, the size reduces almost 2500. Furthermore, I observed that by applying the new pre=processing function the execution time of the program reduces about 10 seconds, which is about 40% improvement on time.

2. **TF-idf Model**

The accuracy score of the TF-IDF model does not have a dramatic improvement by applying the new function. On the contrary, the precision score and recall score decreased a little bit compared to the original results. But the pattern of recall is higher than precision is not changed in the new result.

3. **Word2Vec Model**

The accuracy score of Word2Vec model does not have a dramatic improvement, although, the number above is not as high as previous. Because the difference is too small to tell, it is still in the range of fluctuation. At the same time, the precision score and recall score is higher than the previous results. The reason has then calculated the mean of each vector, there are fewer features that affect the mean. So the mean of each document becomes more relevant to the optimal features.

Conclusion:

Since the Text-normalization strategy is the only thing that has been changed in the executing process, then I can conclude that the Classification result of Bag of words model and Word2Vec model perform observable better, if we extracting more optimal feature from the input dataset. The TF-IDF performs not better than previous by using the new function. And the time consumption of the new function improves significantly.