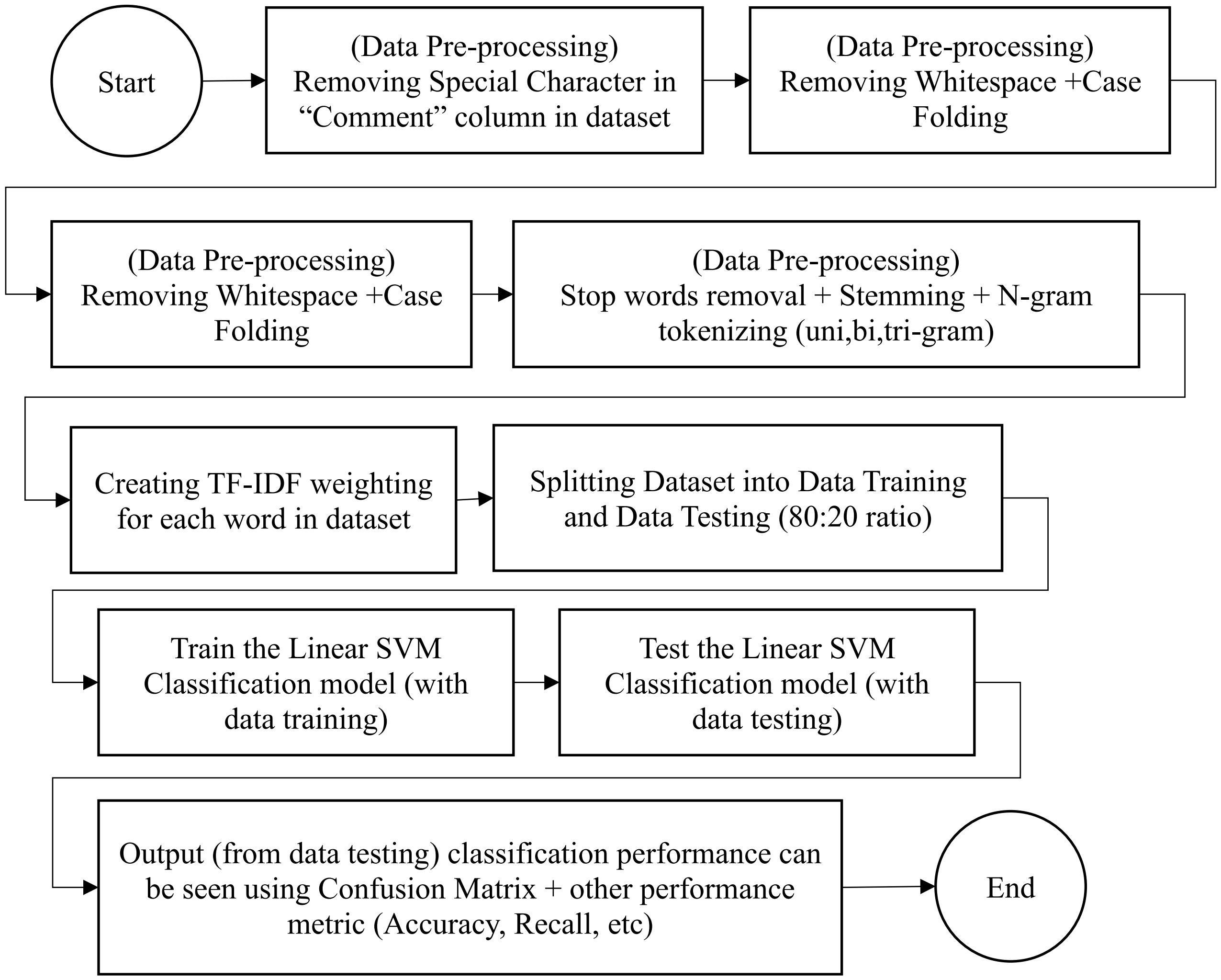
This documentation will explain the workflow and every essential detail of each task in high language level /non-technical language, so most people could comprehend and easily understand this documentation.

Spam Classification Task:

This Spam Classification task will be using the Linear SVM algorithm. This flowchart will perfectly show the overall workflow of all the Classification process.


Start
(Data Pre-processing) Removing Special Character in "Comment" column in dataset
(Data Pre-processing) Removing Whitespace +Case Folding
(Data Pre-processing) Removing Whitespace +Case Folding
(Data Pre-processing) Stop words removal + Stemming + N-gram tokenizing (uni,bi,tri-gram)
Creating TF-IDF weighting for each word in dataset
Splitting Dataset into Data Training and Data Testing (80:20 ratio)
Train the Linear SVM Classification model (with data training)
Test the Linear SVM Classification model (with data testing)
Output (from data testing) classification performance can be seen using Confusion Matrix + other performance metric (Accuracy, Recall, etc)
End

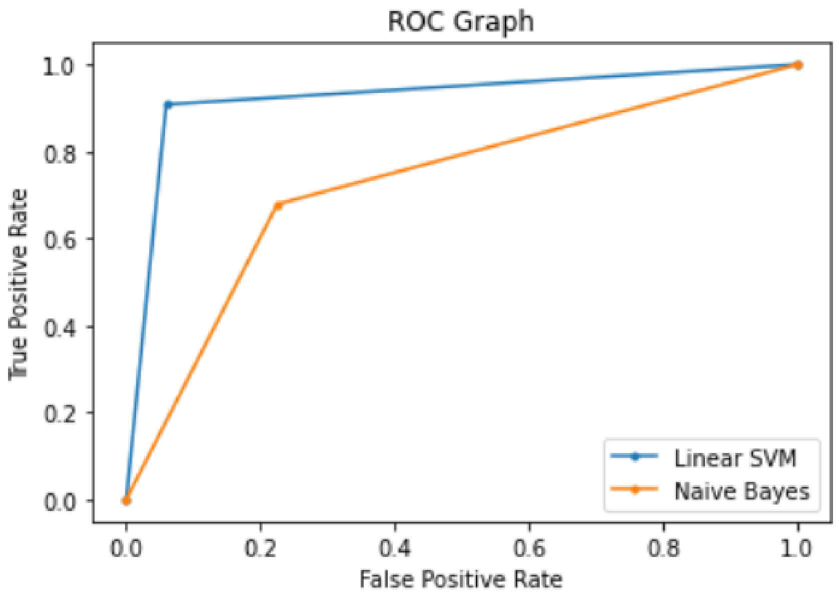
Corpus was used in this classification is Stop word Corpus from NLTK library. This corpus is in English language and used for stop words removal (removing most common English words).

Machine Learning input has to be numbers therefore word embedding method is required (to convert words into vectors/numbers that machine can learn). There are several methods such as TF-IDF, Word2vec, CountVectorizer, etc. This model uses TF-IDF word embedding method because TF-IDF is the most common word embedding method to be used in most text classifications. TF-IDF provides word weighting value that based on how important that word in the corpus/ dataset. This method proved to be so successful and only several word embedding methods could compete their performance with TF-IDF such as Word2Vec, GloVe, etc. TF-IDF is mostly used for the text classification that heavily based on semantic similarity. With TF-IDF simplicity algorithm, comes with less computation cost than other candidates (Word2Vec and GloVe). Therefore, TF-IDF was chosen as the word embedding method for this task.

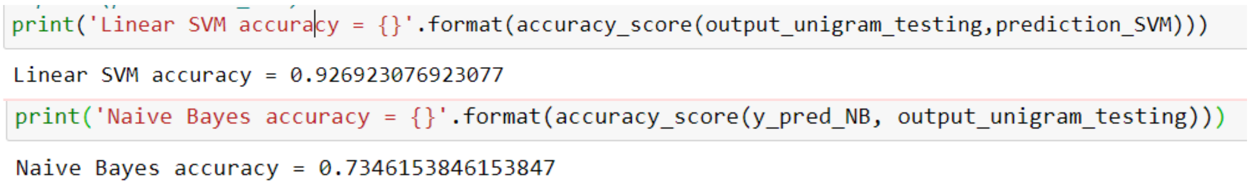
There were several reasons to choose Linear SVM (Scikit-Learn library, SVC) as a model for this task. If we look at the size of dataset, we can consider it as a small size dataset. Therefore, deep learning algorithm will not be sufficient to use (large dataset needed) and another Machine Learning algorithm can replace it. There are 2 common machine learning classifier algorithms for text classification namely, SVM (Support Vector Machine) and NB (Naïve Bayes). However, SVM performed much better than NB for this dataset based on Confusion Matrix, ROC graph (as shown below), accuracy, precision, AUC score, etc.

For this classification, uni-gram (1-gram) has best accuracy performance than uni-gram, where other N-gram, such as bi-gram and tri-gram, has worse accuracy performance. Therefore, uni-gram is used in this classification.

Here are the Classifier model performance results:



```
Linear SVM AUC score: 0.9243271158636613
Naive Bayes AUC score: 0.72686675982745
```



```
print('Linear SVM accuracy = {}'.format(accuracy_score(output_unigram_testing,prediction_SVM)))
```

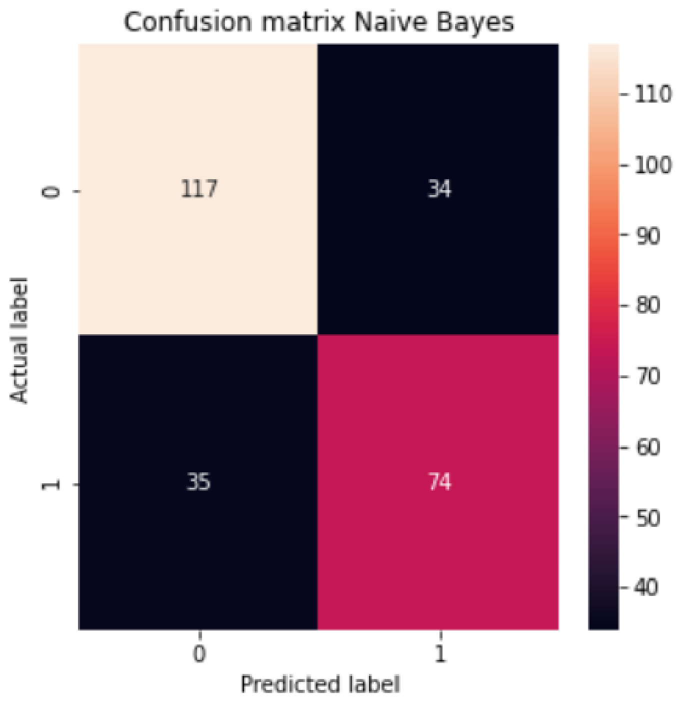
```
Linear SVM accuracy = 0.926923076923077
```

```
print('Naive Bayes accuracy = {}'.format(accuracy_score(y_pred_NB, output_unigram_testing)))
```

```
Naive Bayes accuracy = 0.7346153846153847
```

```
plot_cm(output_unigram_testing, y_pred_NB, "Naive Bayes") # Naive Bayes
```

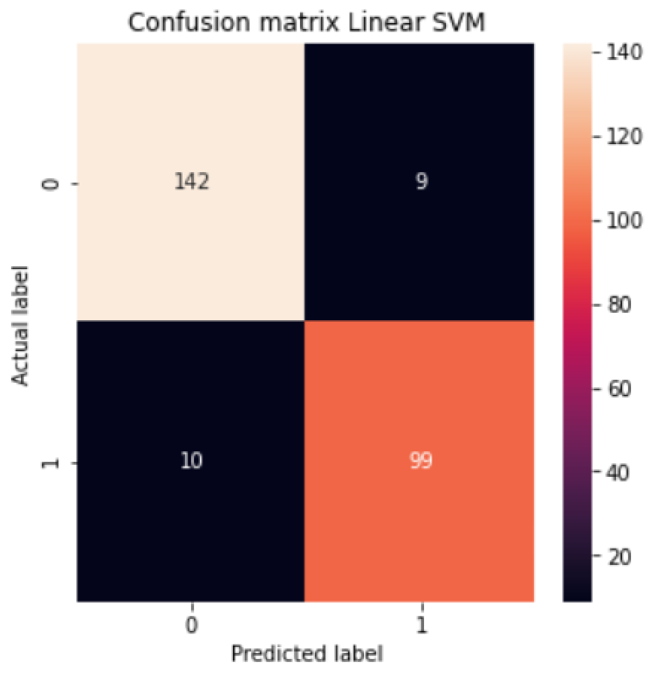
```
Legitimate Transactions Detected (True Negatives):  117
Legitimate Transactions Incorrectly Detected (False Positives):  34
Fraudulent Transactions Missed (False Negatives):  35
Fraudulent Transactions Detected (True Positives):  74
Total Fraudulent Transactions:  109
```



NB Accuracy: (117 + 74)/ (117 + 34 + 35 + 74) = 0.73
NB Precision: (74)/ (74+34) = 0.685
NB Recall: 74/ (74+35) = 0.678
NB F-1 Score: 2 * ((0.685*0.678)/(0.685+0.678)) = 0.681

```
plot_cm(output_unigram_testing, prediction_SVM, "Linear SVM") # Using SVM
```

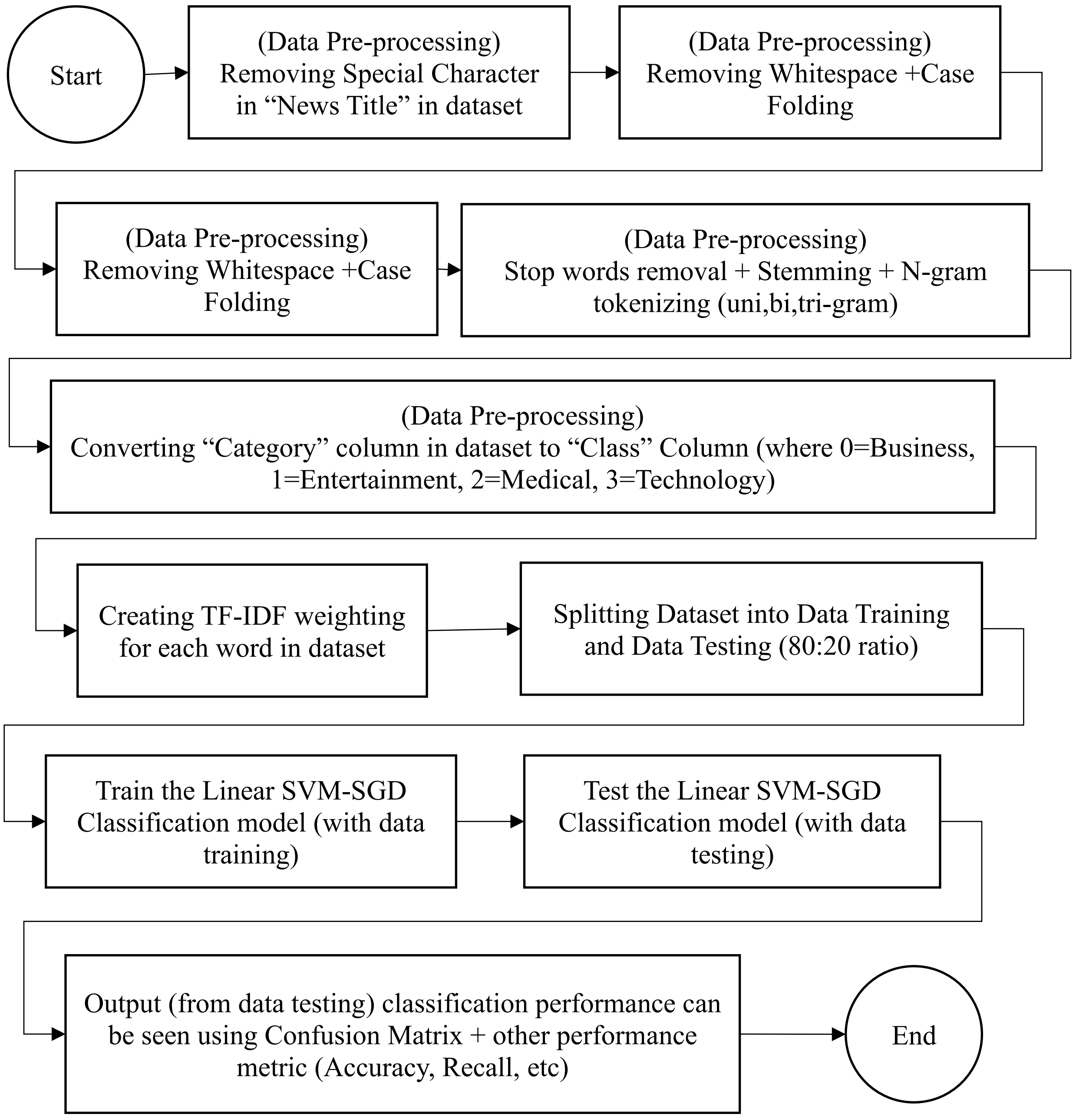
```
Legitimate Transactions Detected (True Negatives):  142
Legitimate Transactions Incorrectly Detected (False Positives):  9
Fraudulent Transactions Missed (False Negatives):  10
Fraudulent Transactions Detected (True Positives):  99
Total Fraudulent Transactions:  109
```



Linear SVM Accuracy: (99+142)/(142+9+10+99) = **0.926**
Linear SVM Precision: 99/ (99+9) = **0.916**
Linear SVM Recall: 99/(99+10) = **0.908**
Linear SVM F-1 Score: 2 * ((0.916*0.908)/ (0.916+0.908)) = **0.911**

News Title Classification Task:

This Spam Classification task will be using the Linear SVM with SGD (Stochastic Gradient Descent) algorithm. This flowchart will perfectly show the overall workflow of all the Classification process.


Start
(Data Pre-processing) Removing Special Character in "News Title" in dataset
(Data Pre-processing) Removing Whitespace +Case Folding
(Data Pre-processing) Removing Whitespace +Case Folding
(Data Pre-processing) Stop words removal + Stemming + N-gram tokenizing (uni,bi,tri-gram)
(Data Pre-processing) Converting "Category" column in dataset to "Class" Column (where 0=Business, 1=Entertainment, 2=Medical, 3=Technology)
Creating TF-IDF weighting for each word in dataset
Splitting Dataset into Data Training and Data Testing (80:20 ratio)
Train the Linear SVM-SGD Classification model (with data training)
Test the Linear SVM-SGD Classification model (with data testing)
Output (from data testing) classification performance can be seen using Confusion Matrix + other performance metric (Accuracy, Recall, etc)
End

Corpus was used in this classification is Stop word Corpus from NLTK library. This corpus is in English language and used for stop words removal (removing most common English words).

This model uses TF-IDF word embedding method because TF-IDF is the most common word embedding method to be used in most text classifications and has low computational cost.

There were several reasons to choose Linear SVM with SGD (Scikit-Learn library, SGDClassifier) as a model for this task. If we look at the size of dataset, we can consider it as a large size dataset and has multiclass (Class has more than 2). Therefore, deep learning algorithm will be sufficient to use (large dataset needed). There is a common machine learning classifier algorithm for text classification namely, SVM-SGD (Support Vector Machine). Unfortunately, Deep Learning such as ANN (Artificial Neural Network) has lots of hyperparameter that need to be tuned properly to achieve high performance.

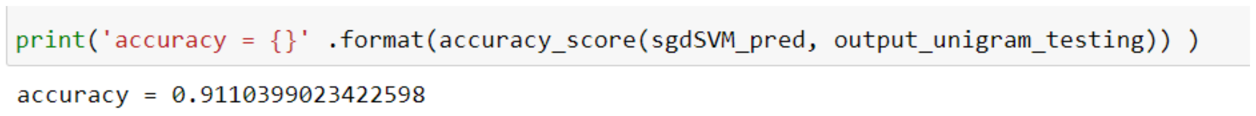
In theoretical, ANN such as CNN or RNN could outperform SVM, but in practical, ANN mostly underperformed compared to SVM due lots of hyperparameter configurations needed, high computational cost, and complex data preparation.

Moreover, in this model, the SVM with SGD algorithm is slightly different than the previous task (SVM). The model algorithm is using SVM with SGD due to the huge dataset. SGD could reduce computational cost by using mini batch of data and randomize it to reduce bias in data. Fortunately, Scikit-Learn library (SGDClassifier) support multi-class classification. Therefore, SVM with SGD is chosen as classification model for this task.

In the future, I would likely to try implementing CNN or RNN or another deep learning algorithm to do text classification model (that has huge dataset and lots of classes) ,and try to compare the performance result with SVM-SGD.

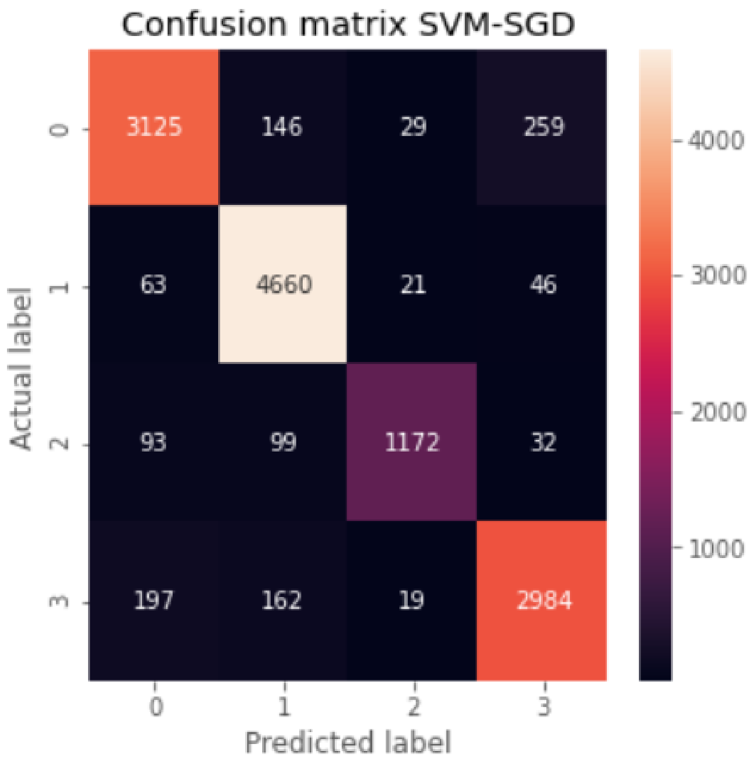
For this classification, uni-gram (1-gram) has best accuracy performance, where other N-gram, such as bi-gram and tri-gram, has worse accuracy performance than uni-gram. Therefore, uni-gram is used in this classification.

Here are the Classifier model performance results:



```
print('accuracy = {}' .format(accuracy_score(sgdSVM_pred, output_unigram_testing)) )
```

```
accuracy = 0.9110399023422598
```



Label 0 (Business)
TP = 3125
FN = (146 + 29 + 259) = 434
FP = (63 + 93 + 197) = 353
TN = (4660 + 21 + 46 + 99 + 1172 +32+162+19+ 2984) = 9195
Accuracy = 0.939
Precision: 3125/ (3125+353) **= 0.898**
Recall: 3125/ (3125+434) **= 0.878**
F-1 Score: 2 * ((0.898*0.878)/ (0.898+0.878)) **= 0.887**

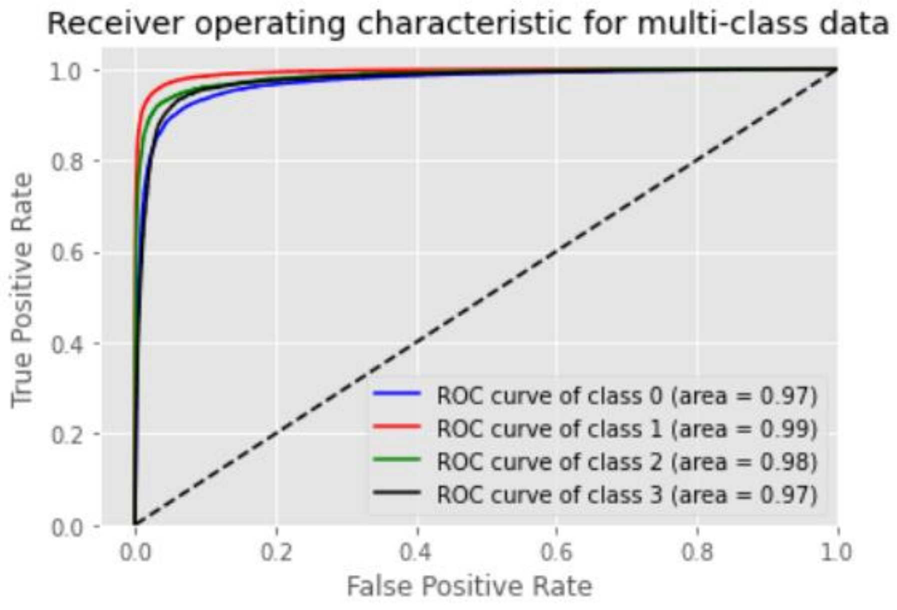
Label 1 (Entertainment)
TP = 4660
FN = (63 + 21 + 46) = 130
FP = (146 + 99 + 162) = 407
TN = (3125 + 29 + 259 + 93 + 1172 + 32 + 197 + 19 + 2984) = 7910
Accuracy = 0.95
Precision: 4660/ (4660+407) **= 0.919**
Recall: 4660/ (4660+130) **= 0.972**
F-1 Score: 2 * ((0.919*0.972)/ (0.919+0.972)) **= 0.944**

Label 2 (Medical)
TP = 1172
FN = (93 + 99 + 32) = 224
FP = (29 + 21 + 19) = 69

TN = (3125 + 146 + 259 + 63 + 4660 + 46 + 197 + 162 + 2984) = 11642
Accuracy = 0.977
Precision: 1172/ (1172+69) **= 0.944**
Recall: 1172/ (1172+224) **= 0.839**
F-1 Score: 2 * ((0.944*0.839)/ (0.944+0.839)) **= 0.888**

Label 3 (Technology)
TP = 2984
FN = (197 + 162 + 19) = 378
FP = (259 + 46 + 32) = 337
TN = (3125 + 146 + 29 + 63 + 4660 + 21 + 93 + 99 + 1172) = 9408
Accuracy = 0.94
Precision: 2984/ (2984+337) **= 0.898**
Recall: 2984/ (2984+378) **= 0.887**
F-1 Score: 2 * ((0.898*0.887)/ (0.898+0.887)) **= 0.892**

ROC Graph for SVM-SGD Classifier =


Receiver operating characteristic for multi-class data
True Positive Rate
False Positive Rate
ROC curve of class 0 (area = 0.97)
ROC curve of class 1 (area = 0.99)
ROC curve of class 2 (area = 0.98)
ROC curve of class 3 (area = 0.97)