NLP Project

Detecting Misogynistic Aggression

Faqeha Nadeem

18I-0674

Section B

Table of Contents

Introduction

As the usage of social media increases with every passing day, Digital spaces are becoming a more and more important part of our daily lives.

In particular women who have previously had less mobility than men to do these things in physical spaces can now turn to digital spaces.

Therefore it is imperative that these spaces are free from harmful,hateful and aggressive language aimed towards women in particular.

Therefore this project presents a method in which a model is trained to detect Misogynistic Aggression, This has been achieved by using a Support Vector Machine (SVM) to train a model on an already annotated multilingual corpus of misogyny and aggression(Bhattacharya and Singh, 2020).

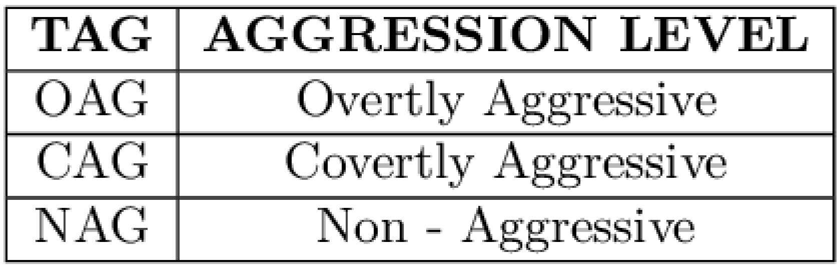
The corpus was discovered reading another paper that used it, the classes and their names used in these papers are the same used in this project. This paper also detected misogyny but by building a complex architecture through BERT (Samghabadi et al., 2020).

The success of the model is then judged by using the metrics precision,accuracy and recall.

Background and Related work

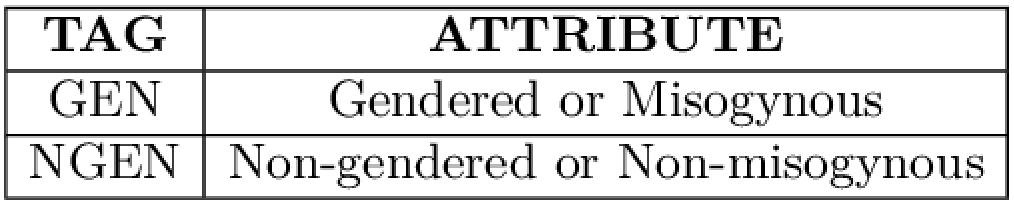
The research paper “Developing a Multilingual Annotated Corpus of Misogyny and Aggression”(Bhattacharya and Singh, 2020) gives us significant detail about how the data set used is extracted and structured. The data has been extracted carefully and methodically, and a strict and carefully composed criteria has been created to label the data set. According to the paper:

> *“The data set is collected from comments on YouTube videos and currently contains a total of over 20,000 comments. The comments are annotated at two levels -aggression (overtly aggressive, covertly aggressive, and non-aggressive) and misogyny (gendered and non-gendered”*



TAG	AGGRESSION LEVEL
OAG	Overtly Aggressive
CAG	Covertly Aggressive
NAG	Non - Aggressive

Table 1:Aggression Annotation Tagset – Sub Task A(Bhattacharya and Singh, 2020)



TAG	ATTRIBUTE
GEN	Gendered or Misogynous
NGEN	Non-gendered or Non-misogynous

Table 2:Misogyny Annotation Tagset– Sub Task B(Bhattacharya and Singh, 2020)

The paper gives guidelines on how to annotate the text with these tags.

Although the corpus is multilingual with Hindi and Bangla samples along with English samples, I only used the English data set which has English and Roman Hindi/Urdu samples.

The idea of dividing the analysis in sub tasks came from the paper titled

“Aggression and Misogyny Detection using BERT: A Multi-Task Approach” (Samghabadi et al., 2020),

The tasks of identifying a text as gendered or misogynous was isolated from identifying it as aggressive, This was helpful in bringing some objectivity into identifying the data, misogynistic aggression seems subjective and difficult to define, however defining something as aggressive or not, or gendered or not is relatively simpler.

The overlap between what is aggressive and what is gendered is then defined as misogynistic aggression according to this sub-task approach.

Project Specification

Initially BERT(**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) was proposed as the model to be trained for this task after some fine-tuning. However this idea was discarded for a variety of reasons.

What is BERT and why it was discarded ?

BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks. It was pre-trained on two tasks that were Masked Language modeling and next sentence prediction.

After some research I realized that BERT is a significantly large and complex system with the standard version of BERT having 110 million parameters and the Large version having 345 million parameters. It is also a computationally expensive system with some usage of a GPU recommended for even small tasks(Rizvi, 2019).

BERT also does a lot of pre-processing for you and therefore gives you less control over the working of the model. This is quiet significant as simple changes such as removing or not removing stopwords can have an impact on your system, and decision can vary according to the task or problem at hand (Schumacher, 2019).

Further research revealed that state of the art models are sometimes not the best choice and simpler traditional models are more successful at completing the task with less computational power.

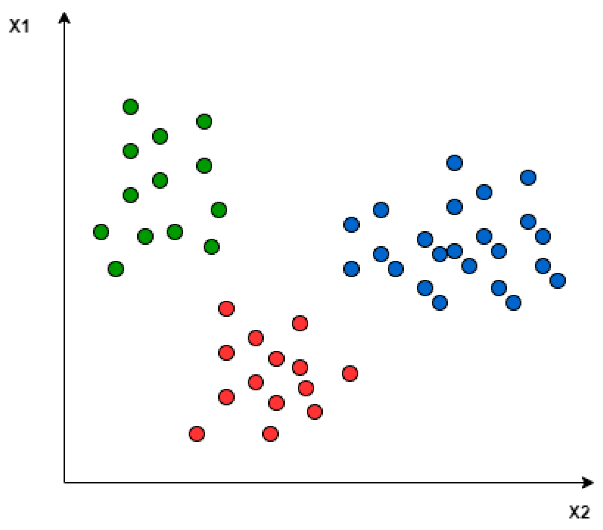
If the task is not the same as the one you are trying to solve, the model may not be the right choice. You may be able to get it working with a lot of fine-tuning and tweaking, but it may not be worth the effort if your task is very different. As BERT was trained on next sentence prediction and masked language modeling it would require significant fine tuning for sentiment analysis or text classification (Horan, 2019).

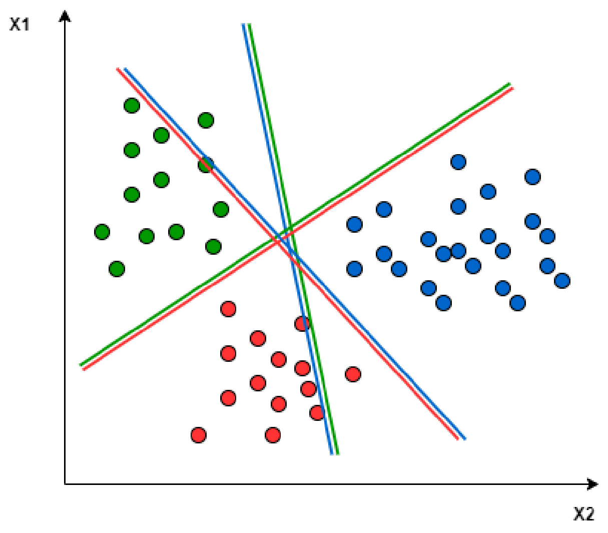
Support Vector Machine for Multi Class Classification

Support Vector Machine (SVM) was what I adopted and used instead as my model. It is regarded as one of the best methods for text classification.

SVM is a supervised machine learning algorithm that helps in classification or regression problems. It aims to find an optimal boundary between the possible outputs.

In the base form, linear separation, SVM tries to find a line that maximizes the separation between a two-class data set of 2-dimensional space points. **To generalize, the objective is to find a hyperplane that maximizes the separation of the data points to their potential classes in an n-dimensional space.** The data points with the minimum distance to the hyperplane (closest points) are called *Support Vectors(baeldung, 2020)*.





In SVM for Multi class classification (*One-to-One* approach), we need a hyperplane to separate between every two classes, neglecting the points of the third class. This means the separation takes into account only the points of the two classes in the current split. For example, the red-blue line tries to maximize the separation only between blue and red points. It has nothing to do with green points.

Why should SVM work well for text classification?

SVM is suitable for the problem as text has

- high dimensional input space
- few irrelevant features
- Most text categorization problems are linearly separable

  (Kaestner, 2013)

Problem Analysis and Solution Design

To complete this task we would have to develop a model or method that can interpret a tweet as misogynist aggression. A simple way to solve the problem is to look closely at its sub problems, the report has already discussed why we chose our model and discarded another one, now we will have a closer look at the sub problems and their corresponding solutions

Problem:

Defining misogynistic aggression is not as straightforward as defining relatively objective criteria, this has led to the following decisions.

Solutions:

We divide this into **two sub-tasks** of identifying gendered language and aggressive language separately in order to reduce subjectivity.

As the classification is not straightforward an unsupervised model or method would not be appropriate, there is no exact rule or criteria simplistic enough for a computer to work with on which we could either make a rule-based system or perform unsupervised machine learning.

The appropriate approach is to use a **supervised model with an annotated corpus** (Schmidt, 2019).

Problem:

There was some noise and irrelevant information in the text and it needed to be cleaned, but how do we clean text without removing important information?

Solutions:

When cleaning the text and pre-processing the method does not use excessive operations and sticks to a conservative approach, so that less relevant features are removed in the cleaning process. To achieve this the program only converts the data to **lowercase** , **removes links,** r**emoves special characters** and r**emoves punctuation.**

The program avoids using other commonly used pre-processing techniques for a variety of reasons.

- Stopwards are not removed as they can sometimes change or define the meaning of a sentence for example:

  Stacy gave her doll to the puppy.

  Stacy gave her doll the puppy.

- Lemmatization and Stemming is also not used because it is best to use a conservative approach here , if they are not significantly improving performance they must not be used.

  As we are using sentence vectors instead of word vectors removing stopwords, lemmatization ans stemming will all decrease the system's understanding of the meaning of the sentences. The latter two will alter the form of words and possibly alter the meaning and removing stopwords may removing the meaning behind how to elements are related

  (Schumacher, 2019)

Problem: Are input is text that's nature must be detected by the system, we need to convert each sample (with average size of 8 words) into a vector that represents the meaning of the input.

Solution: Using TF-IDF vectors for our sentences.

By evaluating TF-IDF or a number of *"the words used in a sentence vs words used in overall document"*, we understand -

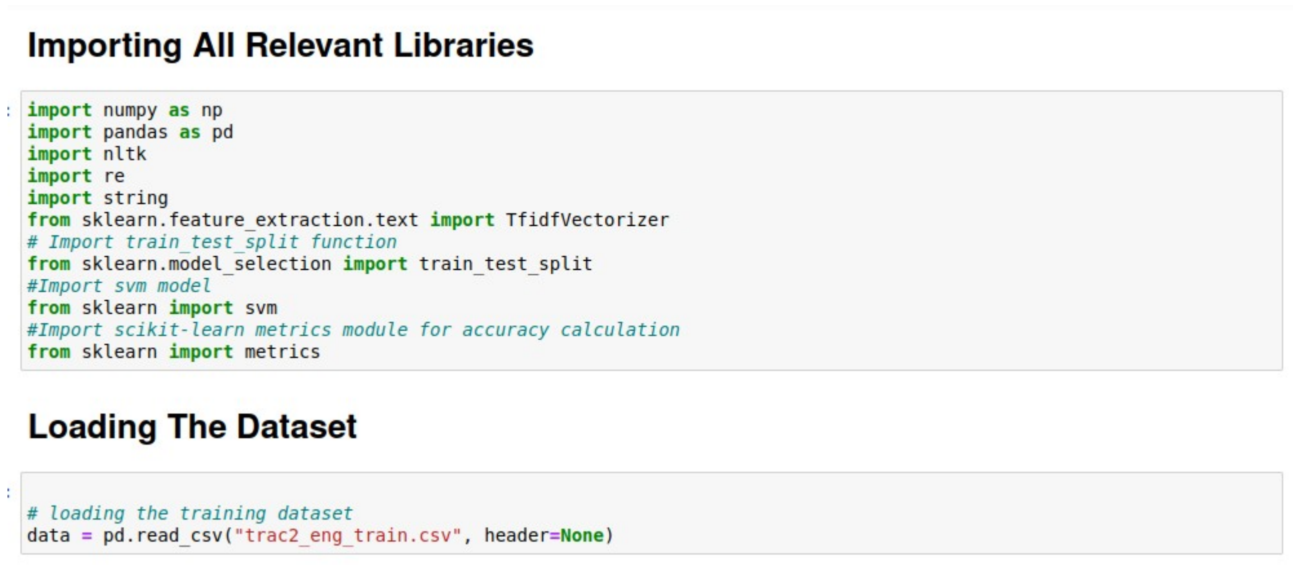
1. how useful a word is to a sentence (which helps us understand the importance of a word in a sentence).
2. how useful a word is to a document (which helps us understand the important words with more frequencies in a document).
3. helps us ignore words that are misspelled

(Madan, 2018)

Implementation

I used pandas and numpy for handling data and re, string and nltk for text pre-processing.

For my SVM function and for metrics I used scikit-learn



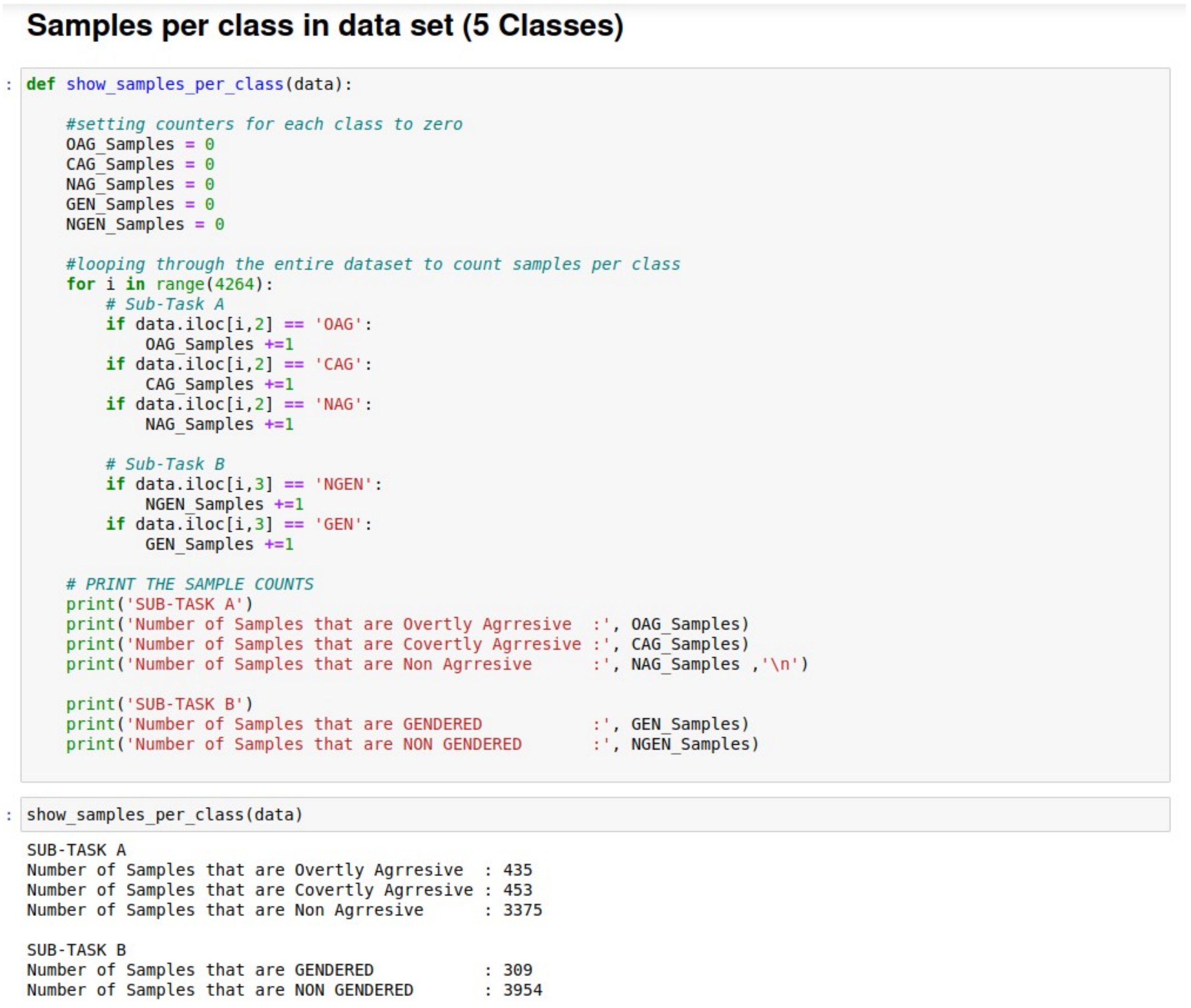
Importing All Relevant Libraries

```
import numpy as np
import pandas as pd
import nltk
import re
import string
from sklearn.feature_extraction.text import TfidfVectorizer
# Import train_test_split function
from sklearn.model_selection import train_test_split
#Import svm model
from sklearn import svm
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
```

Loading The Dataset

```
# loading the training dataset
data = pd.read_csv("trac2_eng_train.csv", header=None)
```

I checked the samples per each class to get an idea of the data and wrote a custom function for it, our data set is unbalanced as only a small amount can be identified as gendered or aggressive.



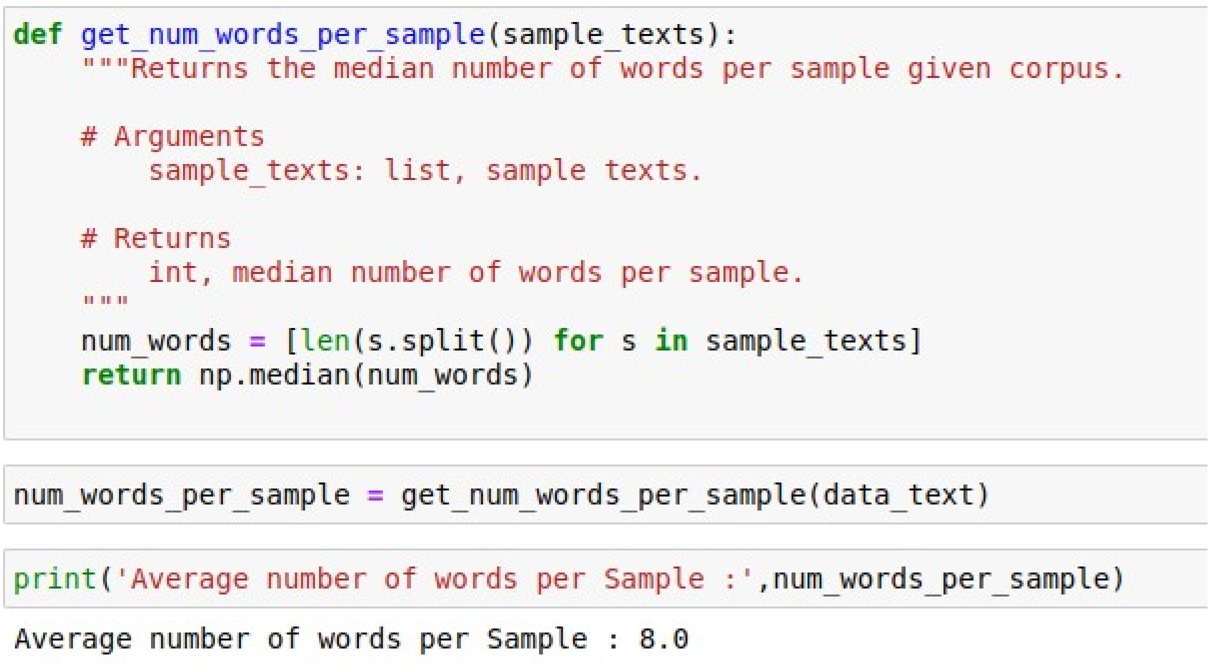
Samples per class in data set (5 Classes)

```python
def show_samples_per_class(data):

    #setting counters for each class to zero
    OAG_Samples = 0
    CAG_Samples = 0
    NAG_Samples = 0
    GEN_Samples = 0
    NGEN_Samples = 0

    #looping through the entire dataset to count samples per class
    for i in range(4264):
        # Sub-Task A
        if data.iloc[i,2] == 'OAG':
            OAG_Samples +=1
        if data.iloc[i,2] == 'CAG':
            CAG_Samples +=1
        if data.iloc[i,2] == 'NAG':
            NAG_Samples +=1

        # Sub-Task B
        if data.iloc[i,3] == 'NGEN':
            NGEN_Samples +=1
        if data.iloc[i,3] == 'GEN':
            GEN_Samples +=1

    # PRINT THE SAMPLE COUNTS
    print('SUB-TASK A')
    print('Number of Samples that are Overtly Agrresive  :', OAG_Samples)
    print('Number of Samples that are Covertly Agrresive :', CAG_Samples)
    print('Number of Samples that are Non Agrresive      :', NAG_Samples ,'\n')

    print('SUB-TASK B')
    print('Number of Samples that are GENDERED           :', GEN_Samples)
    print('Number of Samples that are NON GENDERED       :', NGEN_Samples)
```

```python
show_samples_per_class(data)
```

```
SUB-TASK A
Number of Samples that are Overtly Agrresive  : 435
Number of Samples that are Covertly Agrresive : 453
Number of Samples that are Non Agrresive      : 3375

SUB-TASK B
Number of Samples that are GENDERED           : 309
Number of Samples that are NON GENDERED       : 3954
```

The average number of words per text were determined by a custom function to be 8, this length can inform our choice of numerical representation of text



```
def get_num_words_per_sample(sample_texts):
    """Returns the median number of words per sample given corpus.

    # Arguments
        sample_texts: list, sample texts.

    # Returns
        int, median number of words per sample.
    """
    num_words = [len(s.split()) for s in sample_texts]
    return np.median(num_words)
```

```
num_words_per_sample = get_num_words_per_sample(data_text)
```

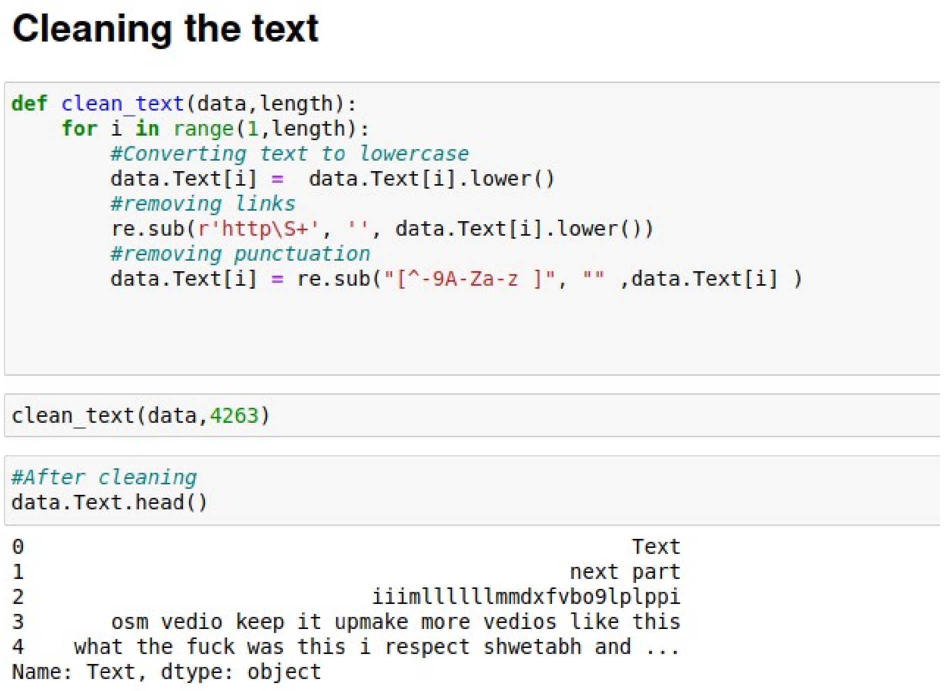
```
print('Average number of words per Sample :',num_words_per_sample)
```

```
Average number of words per Sample : 8.0
```

I skipped lemmatization and removing stopwords due to previously mentioned reasons

and cleaned the text by converting to lowercase, removing links, special characters and punctuation

I then extracted feature vectors for each sample text through **tfidfvectorizer** in sci-kit learn



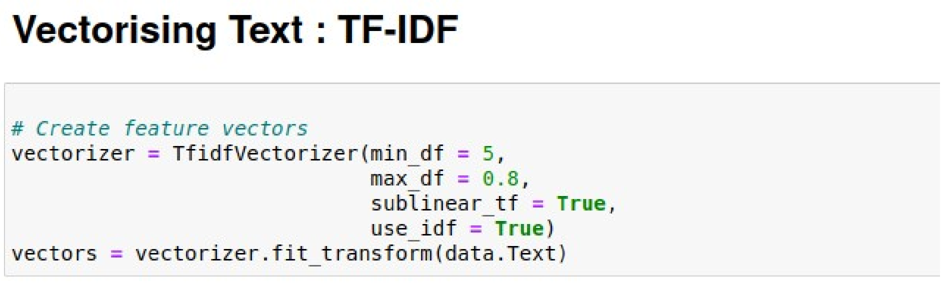
Cleaning the text

```
def clean_text(data,length):
    for i in range(1,length):
        #Converting text to lowercase
        data.Text[i] =  data.Text[i].lower()
        #removing links
        re.sub(r'http\S+', '', data.Text[i].lower())
        #removing punctuation
        data.Text[i] = re.sub("[^-9A-Za-z ]", "" ,data.Text[i] )
```

```
clean_text(data,4263)
```

```
#After cleaning
data.Text.head()
```

```
0                                                 Text
1                                            next part
2                              iiimlllllmmdxfvbo9lplppi
3        osm vedio keep it upmake more vedios like this
4     what the fuck was this i respect shwetabh and ...
Name: Text, dtype: object
```



Vectorising Text : TF-IDF

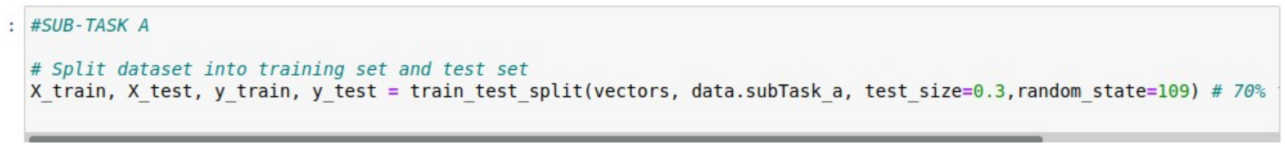
```
# Create feature vectors
vectorizer = TfidfVectorizer(min_df = 5,
                             max_df = 0.8,
                             sublinear_tf = True,
                             use_idf = True)
vectors = vectorizer.fit_transform(data.Text)
```

I split the data into training and test set and implemented SVM through sci-kit learn library function

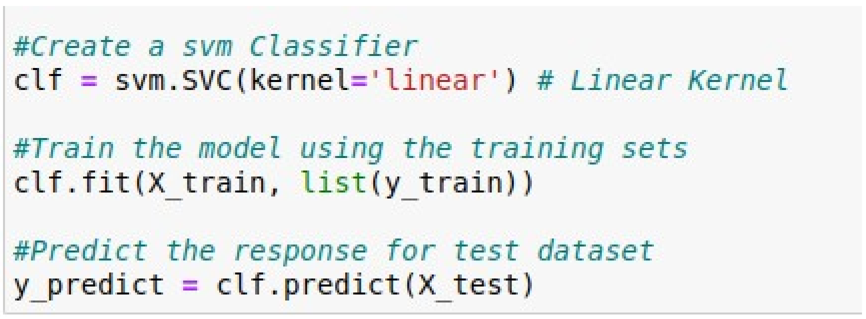
I carried this out for sub task A and then repeated the same process for Sub Task B and misogynistic aggression overall (overlap of sub tasks A and B)

Creating a Linear SVM (Support Vector Machine) Model

Sub Task A : Aggression

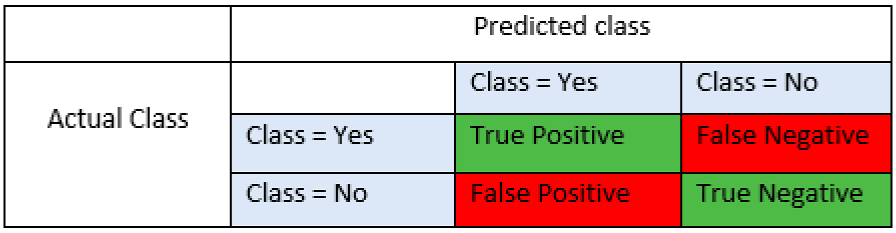


```
#SUB-TASK A

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(vectors, data.subTask_a, test_size=0.3,random_state=109) # 70%
```



```
#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, list(y_train))

#Predict the response for test dataset
y_predict = clf.predict(X_test)
```

Evaluation



	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

I evaluated the success of the model through precision, recall and f1 score/f measure.

On finding the samples per class we can observe that the data is unbalanced and therefore accuracy would not be an appropriate measure in this scenario.

These metrics were calculated through the ‘metrics’ functionality in sci-kit learn

The model seems to be working well for Sub Task B. The metrics for sub task A are only slightly less and still adequate.

Once we are confident about the model’s ability to detect Aggression and Gendered language separately we can move on to evaluating it for Misogynistic Aggression overall.

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

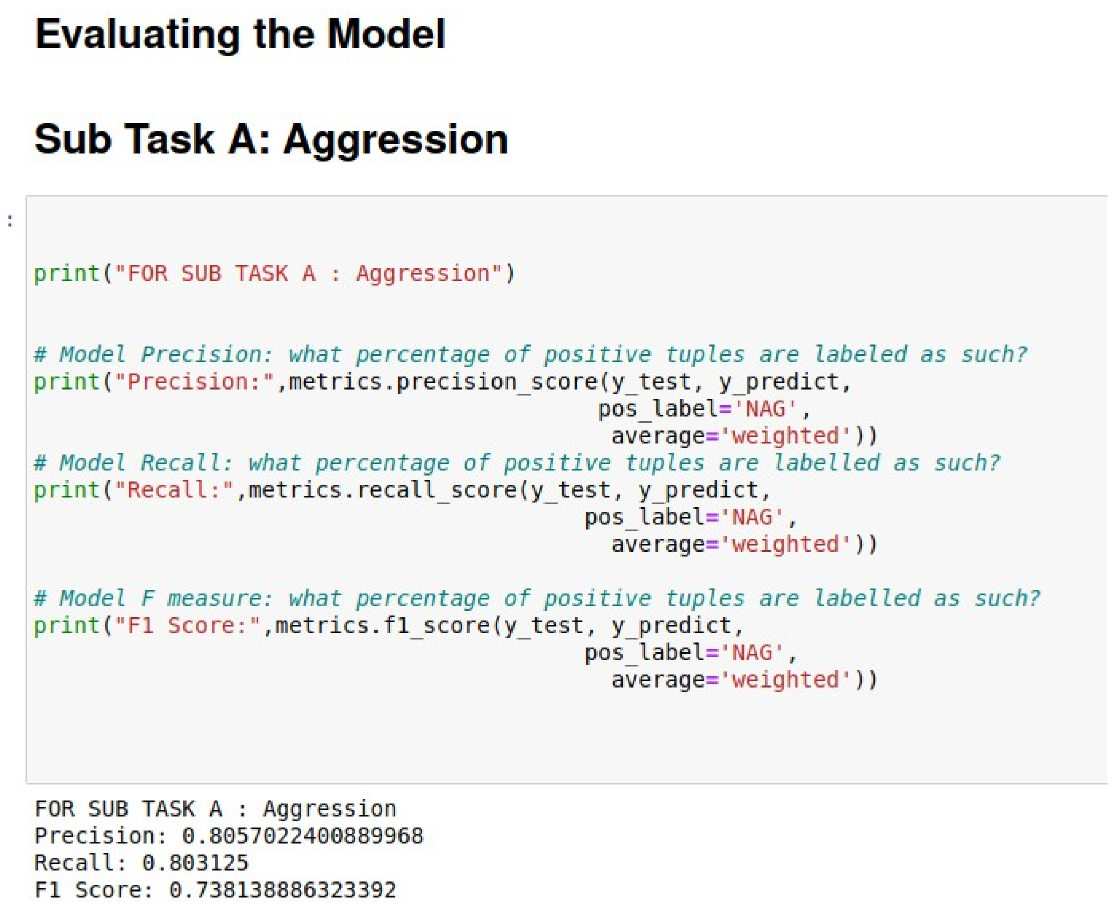
Precision = TP/TP+FP

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. The question recall answers is: Of all the passengers that truly survived, how many did we label? We have got recall of 0.631 which is good for this model as it’s above 0.5.

Recall = TP/TP+FN

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall. In our case, F1 score is 0.701.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

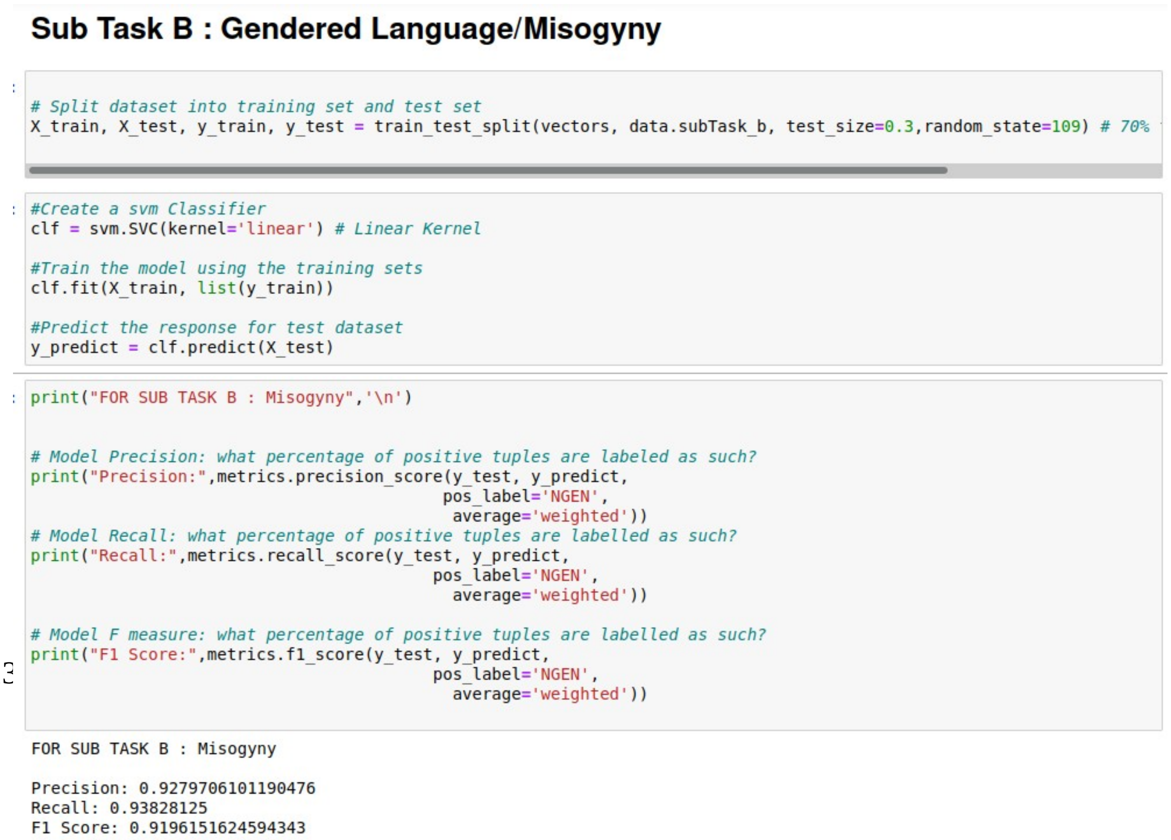


Evaluating the Model

Sub Task A: Aggression

```
print("FOR SUB TASK A : Aggression")


# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_predict,
                                           pos_label='NAG',
                                            average='weighted'))
# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_predict,
                                          pos_label='NAG',
                                            average='weighted'))

# Model F measure: what percentage of positive tuples are labelled as such?
print("F1 Score:",metrics.f1_score(y_test, y_predict,
                                          pos_label='NAG',
                                            average='weighted'))
```

```
FOR SUB TASK A : Aggression
Precision: 0.8057022400889968
Recall: 0.803125
F1 Score: 0.738138886323392
```



Sub Task B : Gendered Language/Misogyny

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(vectors, data.subTask_b, test_size=0.3,random_state=109) # 70%
```

```
#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, list(y_train))

#Predict the response for test dataset
y_predict = clf.predict(X_test)
```

```
print("FOR SUB TASK B : Misogyny",'\n')


# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_predict,
                                           pos_label='NGEN',
                                            average='weighted'))
# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_predict,
                                          pos_label='NGEN',
                                            average='weighted'))

# Model F measure: what percentage of positive tuples are labelled as such?
print("F1 Score:",metrics.f1_score(y_test, y_predict,
                                          pos_label='NGEN',
                                            average='weighted'))
```

```
FOR SUB TASK B : Misogyny

Precision: 0.9279706101190476
Recall: 0.93828125
F1 Score: 0.9196151624594343
```

Conclusion

After cleaning the text without processing it too much and losing important information we applied SVM with TF-IDF as vectors for text we were able to train our model to provide precise results with an F1 score of 0.7381 for sub task A and F1 score of 0.9196 for sub task B.

References

1. Bhattacharya, S. and Singh, S., 2020. *Developing a Multilingual Annotated Corpus of Misogyny and Aggression*. [online] Aclweb.org. Available at: <https://www.aclweb.org/anthology/2020.trac-1.25.pdf> [Accessed 20 June 2021].
2. Samghabadi, N., Patwa, P., PYKL, S., Mukherjee, P., Das, A. and Solorio, T., 2020. *Aggression and Misogyny Detection using BERT: A Multi-Task Approach*. [online] ACL Anthology. Available at: <https://www.aclweb.org/anthology/2020.trac-1.20/> [Accessed 25 June 2021].
3. Rizvi, M., 2019. *What is BERT | BERT For Text Classification*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/> [Accessed 20 June 2021].
4. Schumacher, A., 2019. *When (not) to Lemmatize or Remove Stop Words in Text Preprocessing*. [online] Open Data Group. Available at: <https://opendatagroup.github.io/data%20science/2019/03/21/preprocessing-text.html> [Accessed 20 June 2021].
5. Horan, C., 2019. *When Not to Choose the Best NLP Model*. [online] FloydHub Blog. Available at: <https://blog.floydhub.com/when-the-best-nlp-model-is-not-the-best-choice/> [Accessed 20 June 2021].
6. baeldung, b., 2020. *Multiclass Classification Using Support Vector Machines*. [online] Available at: <https://www.baeldung.com/cs/svm-multiclass-classification> [Accessed 20 June 2021].
7. Kaestner, C., 2013. Support Vector Machines and Kernel Functions for Text Processing. *Revista de Informática Teórica e Aplicada*, 20(3), p.130.
8. Schmidt, T., 2019. *Unsupervised Text Classification*. [online] Medium. Available at: <https://medium.com/@ai_medecindirect/unsupervised-text-classification-695392c6fac7> [Accessed 20 June 2021].
9. Madan, R., 2018. *TF-IDF/Term Frequency Technique: Easiest explanation for Text classification in NLP with Python*. [online] Medium. Available at: <https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3> [Accessed 20 June 2021].