Natalia Katchoura

Hate Speech and Offensive language detection

FINAL REPORT

Building models that can detect Hate speech and Offensive language.

Problem Statement

Online social media has become one of the most important ways to communicate and exchange ideas. Unfortunately, the discourse is often crippled by abusive language that can have damaging effects on social media users.

Online social media networks normally deal with the offensive language problem by simply filtering out a post when it is flagged as offensive.

Offensive language identification in social media is a difficult task because the textual contents in such an environment is often unstructured, informal, and even misspelled.

In our project we will build a model that can detect the hate speech and offensive language

Data Wrangling

For our project we will be using the 25K+ labeled tweets dataset

Contributors viewed short text and identified them as
- 0 - contained hate speech,
- 1 - was offensive but without hate speech, or
- 2 - was not offensive at all.

At first, we checked the dataset for the missing data but there wasn't any missing data. We have a clean dataset.

The dataset has 7 columns. We will drop all the columns that contain numbers of CF users who judged the tweet and keep only 2 columns: the tweet and class

Pre-processing and Training Data Development

- Lowercase text
- Remove whitespace
- Remove numbers
- Remove special characters
- Remove emails
- Remove stop words
- Remove NAN
- Remove weblinks
- Expand contractions (if possible not necessary)
- Tokenize

For text preprocessing we were using regular expressions (regex) and NLTK.

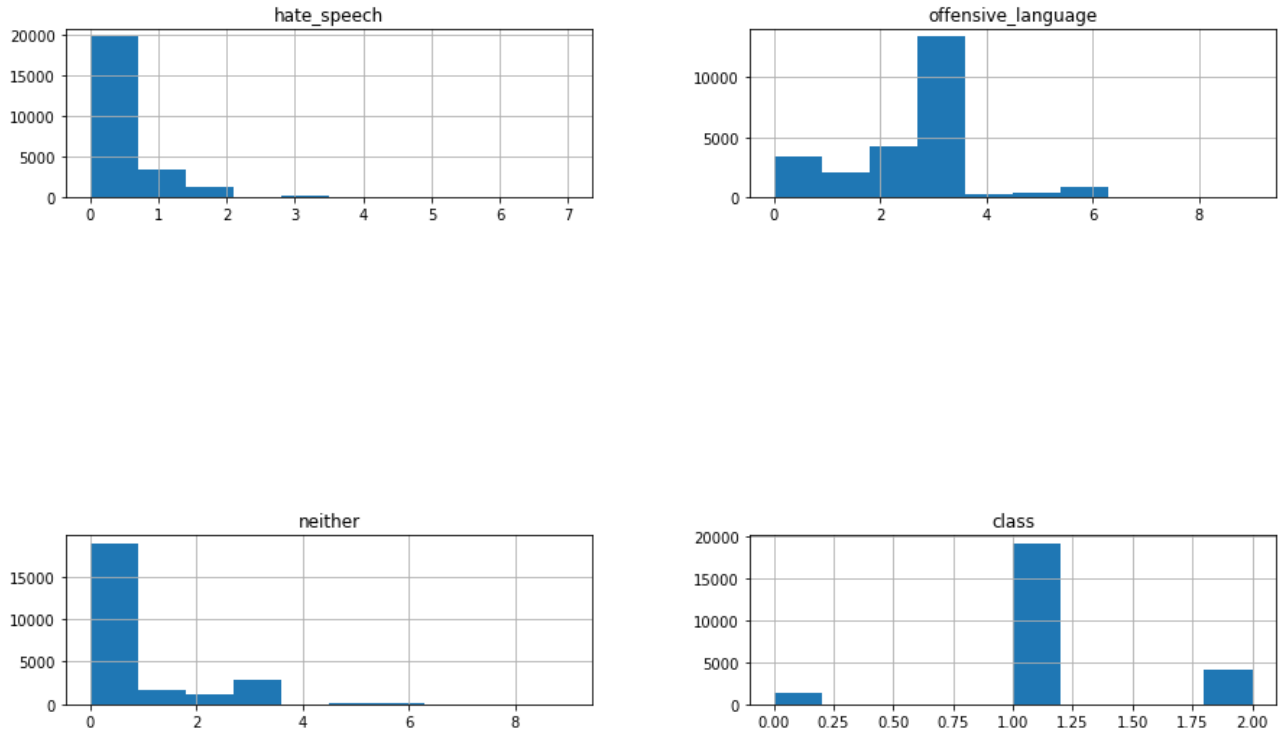
Not all tasks need the same level of preprocessing. For some tasks, you can get away with the minimum. However, for others , like our dataset that is very noisy that, if we don’t preprocess enough, it’s going to be garbage-in-garbage-out.

	tweet	0	1	2	cleanTweet	tokenized_tweets
0	!!! RT @mayasolovely: As a woman you shouldn't...	0	0	1	mayasolovely woman complain cleaning house amp...	[mayasolovely, woman, complain, cleaning, hous...
1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	0	1	0	mleew boy dats cold tyga dwn bad cuffin dat ho...	[mleew, boy, dats, cold, tyga, dwn, bad, cuffi...
2	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	0	1	0	urkindofbrand dawg sbabylife ever fuck bitch s...	[urkindofbrand, dawg, sbabylife, ever, fuck, b...
3	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...	0	1	0	c_g_anderson viva_based look like tranny	[c_g_anderson, viva_based, look, like, tranny]
4	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...	0	1	0	shenikaroberts shit hear might true might fake...	[shenikaroberts, shit, hear, might, true, migh...
...	...	...	...	...	...	...
24778	you's a muthaf***in lie “@LifeAsKing: @2...	0	1	0	muthaf lie lifeasking _pearls corey_emanuel ri...	[muthaf, lie, lifeasking, _pearls, corey_emanu...
24779	you've gone and broke the wrong heart baby, an...	0	0	1	gone broke wrong heart baby drove redneck crazy	[gone, broke, wrong, heart, baby, drove, redne...
24780	young buck wanna eat!!.. dat nigguh like I ain...	0	1	0	young buck wanna eat dat nigguh like aint fuck...	[young, buck, wanna, eat, dat, nigguh, like, a...
24781	youu got wild bitches tellin you lies	0	1	0	youu got wild bitches tellin lies	[youu, got, wild, bitches, tellin, lies]
24782	~~Ruffled | Ntac Eileen Dahlia - Beautiful col...	0	0	1	ruffled ntac eileen dahlia beautiful color com...	[ruffled, ntac, eileen, dahlia, beautiful, col...

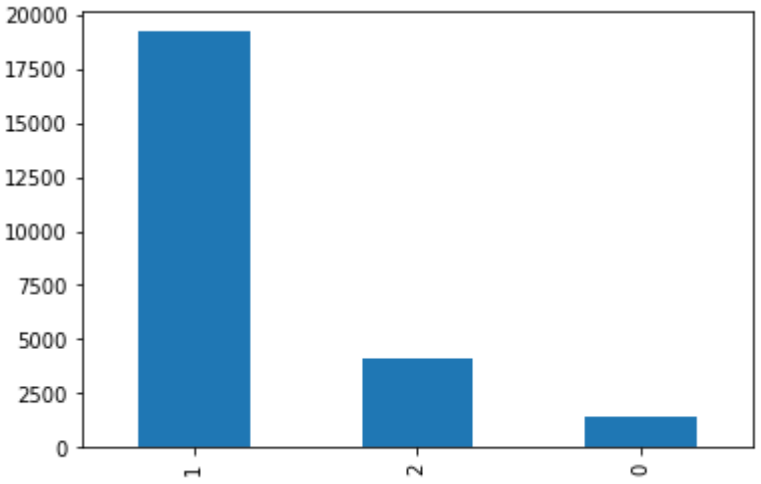
Exploratory Data Analysis

Let’s take a look on the columns distribution

Column distribution



Class label (0 - hate speech, 1 - offensive language, 2 - neither)

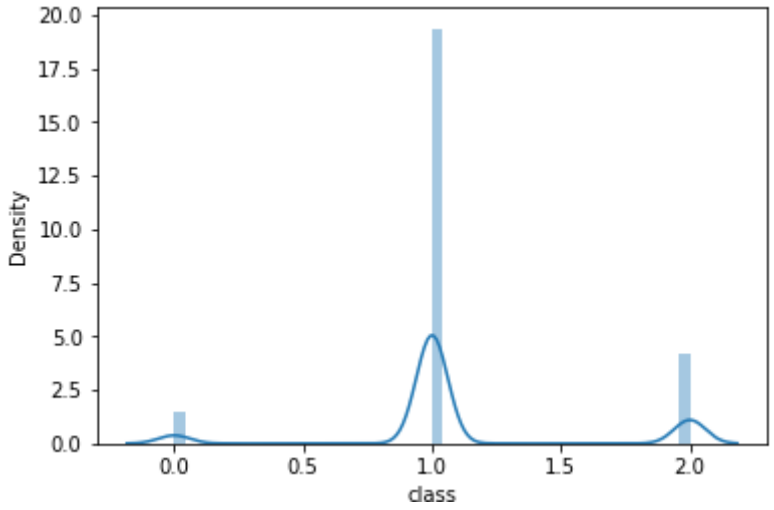


Our dataset contains records:

```
Total : 24783

Hate : 5.77% of total
Offensive : 77.43% of total
Neither : 16.8% of total
```

We have unbalanced dataset

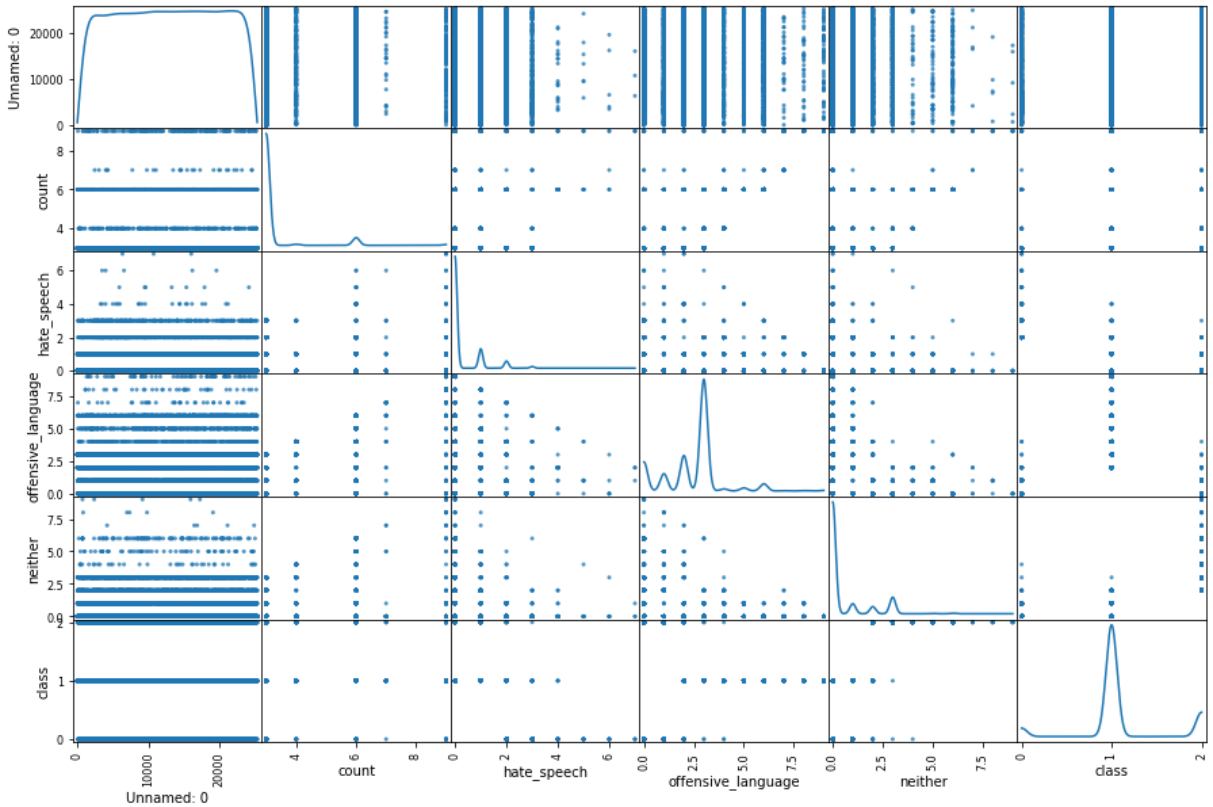


Overall statistics

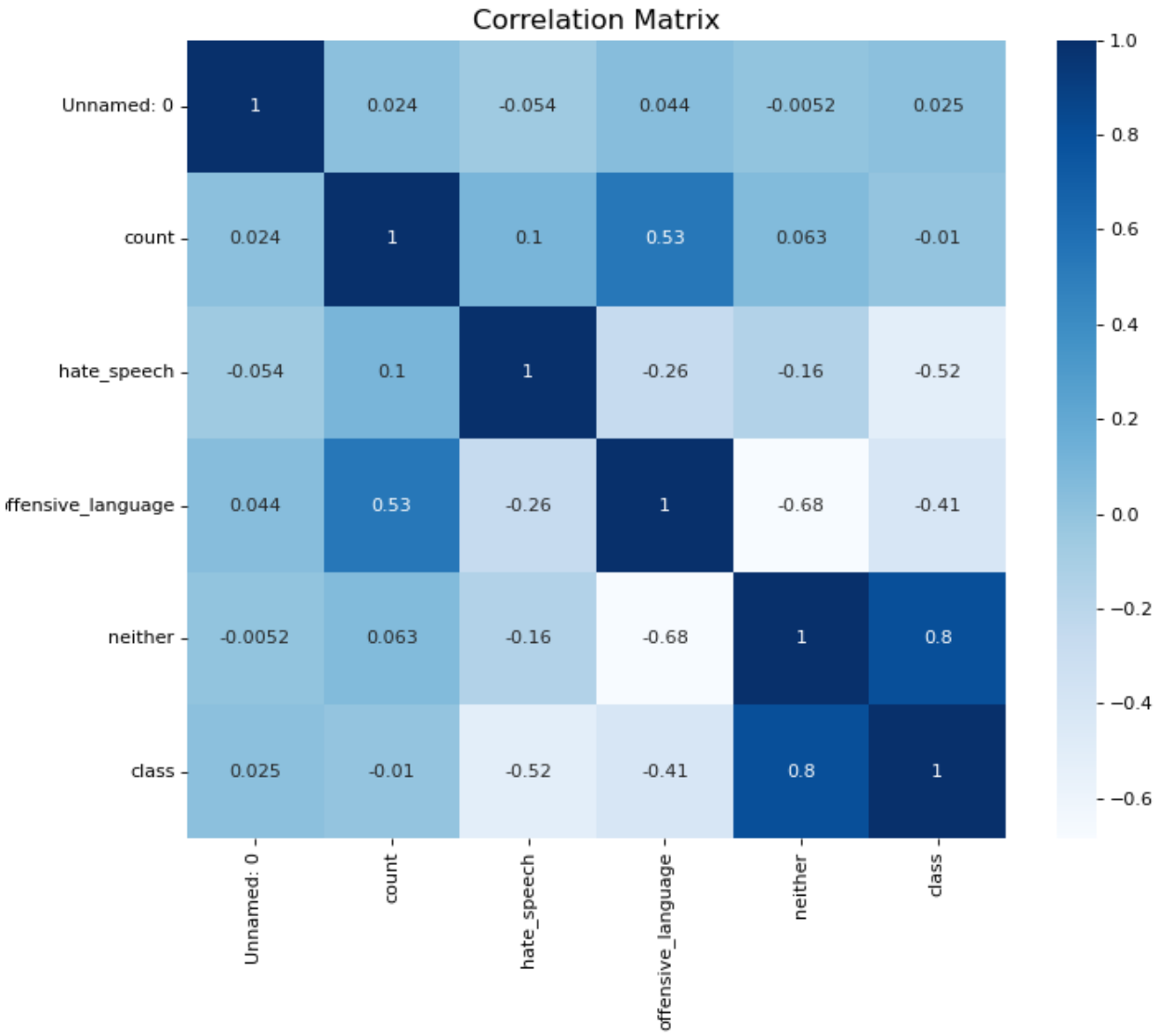
	0	1	2
count	24783.000000	24783.000000	24783.000000
mean	0.057701	0.774321	0.167978
std	0.233182	0.418037	0.373854
min	0.000000	0.000000	0.000000
25%	0.000000	1.000000	0.000000
50%	0.000000	1.000000	0.000000
75%	0.000000	1.000000	0.000000
max	1.000000	1.000000	1.000000

Scatter and Density Plot of the tweets subjectives

Another observation of the distribution of a variable in a dataset.



Pearson correlation matrix of the tweets subjectives



Neither and offensive column show high positive correlation with the column class

Visualization . Word Clouds

Word Clouds are a popular way of displaying how important words are in a collection of texts. The more frequent the word is, the greater space it occupies in the image.

Offensive language

The words from the tweets classified as offensive language in out dataset

Hate language

The words from the tweets classified as hate language in out dataset

Neither offensive or hate language

The words from the tweets classified as not offensive nor hate language in out dataset

Word embeddings

Word embeddings are in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning.

We used Bag-of-words vectors (built in scikit-learn) to predict the text label(category) based on the tweets wording.
We have chosen the BoW method because it is simple and works well, but it treats all words equally and cannot distinguish very common words or rare words. Tf-idf solves this problem of BoW Vectorization.

Modeling

Naive Bayes model

NB model Commonly used when training a new supervised model using text vector data

Naive Bayes algorithm is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature

We train and test a Naive Bayes model using the CountVectorizer and TF-idf data

Results for BoW data:

Accuracy:

```
score = metrics.accuracy_score(y_test, pred)
score
```

```
0.8700774693350549
```

F1 score:

```
F1 = metrics.f1_score(y_test,pred, average='weighted')
F1
```

```
0.8432642139247257
```

Recall score:

```
recall = metrics.recall_score(y_test,pred, average="macro")
recall
```

```
0.5516192716069943
```

Results for Tf-idf data:

Accuracy:

```
accuracy = metrics.accuracy_score(y_test, pred)
accuracy
```

```
0.7995480955455132
```

F1 score:

```
F1_1 = metrics.f1_score(y_test,pred, average='weighted')
F1_1
```

```
0.7302614657992207
```

Recall score:

```
recall_1 = metrics.recall_score(y_test,pred, average="macro")
recall_1
```

```
0.384095079288546
```

We can see that Multinomial Naive Bayes do not perform very well on imbalanced datasets. NB models show relatively low accuracy scores. That is nor what we expected. Using BoW showed higher accuracy score then Tf-idf

Classification with Random Forest

We used Random Forest model hate speech and offensive language detection. Because we have unbalanced data set we created 3 models: standard, model with data weighting and easy ensemble model for imbalances classification

Standard classification returned slightly better results for accuracy score but model with data weighting returned better result for F1 score

Accuracy

```
accuracy = metrics.accuracy_score(y_test, y_pred_rf)
accuracy
```

```
0.89380245319561
```

Recall score

```
recall_rf = metrics.recall_score(y_test,y_pred_rf, average="macro")
recall_rf
```

```
0.6871828655888127
```

F1 score

```
F1_rf = metrics.f1_score(y_test,y_pred_rf, average='weighted')
F1_rf
```

```
0.8866994014172133
```

Against

	precision	recall	f1-score	support
0	0.39	0.29	0.33	359
1	0.93	0.94	0.94	4800
2	0.84	0.87	0.85	1037
accuracy			0.89	6196
macro avg	0.72	0.70	0.71	6196
weighted avg	0.89	0.89	0.89	6196

And

	precision	recall	f1-score	support
0	0.23	0.63	0.34	359
1	0.98	0.81	0.88	4800
2	0.77	0.94	0.85	1037
accuracy			0.82	6196
macro avg	0.66	0.79	0.69	6196
weighted avg	0.90	0.82	0.85	6196

Deep neural network for hate speech detection

Tweets cannot be directly interpreted we need to transform them in a sequence of meaningful integer number in order to be analyzed by NN model
We used Glove embedding for tweet transformation

GloVe embeddings were trained on a dataset of one billion words with a vocabulary of 400 thousand words. There are few different embedding vector sizes such as 50, 100, 200 and 300 dimensions. We use 50 dimensions vector

```
[86] embedding_matrix

     array([[ 0.        ,  0.        ,  0.        , ...,  0.        ,
              0.        ,  0.        ],
            [-0.94415998, -0.30311999, -0.20235001, ..., -1.05350006,
              0.58445001,  0.71354002],
            [-0.13101   , -0.20428   , -0.62955999, ..., -0.84228998,
             -0.49031001,  0.11032   ],
            ...,
            [-0.15861   ,  0.47294   , -0.11118   , ..., -0.64332002,
              0.12794   ,  1.07009995],
            [ 0.16339999,  0.53283   , -0.66540003, ..., -1.20159996,
              0.31439999, -0.12135   ],
            [-0.32337999, -0.67427999, -0.41141999, ..., -0.55097002,
              0.65728003,  0.47944   ]])
```

```
[87] embedding_matrix.shape

     (31204, 50)
```

Because we have an unbalanced dataset using Accuracy as our metrics return not a great result.
Test accuracy with RNN shows us 0.1680116206407547

Choosing metrics for the models comparison

Metrics to evaluate ML models:

1.**Accuracy**
Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of the number of correct predictions to the total number of input samples. It works well only if there are an equal number of samples belonging to each. That is not our case. We established above that we have an unbalanced dataset.

In our case, even if you predict all samples as the most frequent class you would get

a high accuracy rate, which does not make sense at all (because your model is not learning anything, and is just predicting everything as the top class)

2. **Precision**
Precision (#samples correctly predicted/#samples predicted) is a valid choice of evaluation metric when we want to be very sure of our prediction Being very precise.

3. **Recall**
Recall is a valid choice of evaluation metric when we want to capture as many positives as possible.

4. **F1 Score**
F1 Score (2*Precision*Recall/(Precision+Recall) combines precision and recall - used to measure a test's accuracy

Depending on the application, you may want to give higher priority to recall or precision. But there are many applications in which both recall and precision are important.
**It seems that F1 is the best metric to use for our task."

RNN modeling (cont.)

We trained and evaluated the RNN model :

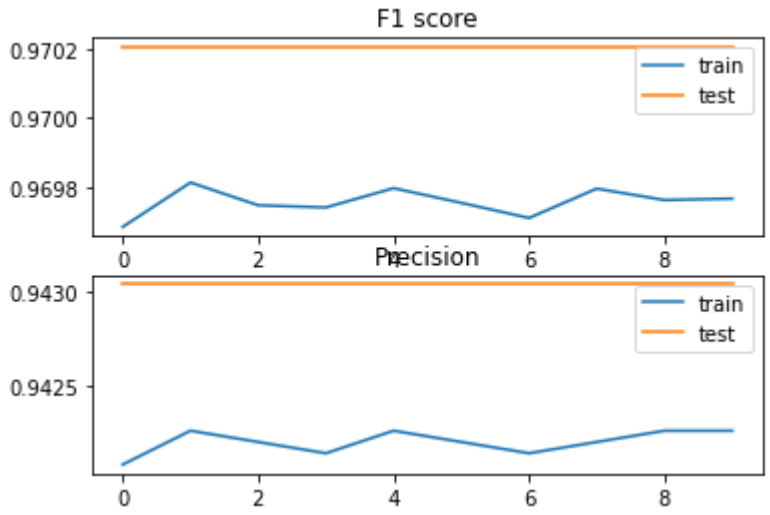
```
# compile the model
rnnmodel.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc',f1_m,precision_m, recall_m])

# fit the model
history = rnnmodel.fit(X_train, y_train, validation_split=0.3, epochs=10, verbose=0)
```

Results:

```
WARNING:tensorflow:Model was constructed with shape (None, 200) for input KerasTensor(typ
WARNING:tensorflow:Model was constructed with shape (None, 200) for input KerasTensor(typ
WARNING:tensorflow:Model was constructed with shape (None, 200) for input KerasTensor(typ
Test metrics with RNN:
Test accuracy: 0.7746933698654175
Test loss: -51091184.0
Test f1_score: 0.9696630239486694
Test precision: 0.942074716091156
Test recall: 1.0
```

Plot F1 score and Precision during the training



The plots look very similar.

Conclusion

I have evaluated each model in terms of model Accuracy score, F1 score and Recall score for both the training and test data, and plotted them.

- The Naitive Bayer model performs better using BoW than TF-idf best.
- Random Forest model performs better than both NB models

For the imbalances dataset the Random Forest model with Class Weighting performs slightly better than the Standard Random Forest or the Random Forest model with Easy Ensemble for imbalanced classification.

- NN models (using Glove embedding) returned following result
    - Test metrics with RNN:
    - Test accuracy: 0.7746933698654175
    - Test loss: -51091184.0
    - Test f1_score: 0.9696630239486694
    - Test precision: 0.942074716091156
    - Test recall: 1.0

Future Research

For future research I would like to add other NLP models for research such as BERT, ELMo
I also would like to train recurrent neural networks on a tweet emotion dataset to learn to recognize emotions in tweets.