## Modelling

To analyse the tweets, we built models using the classifiers shown below, we then did comparative analysis to decide the best performing classifier. The classifiers used to build the models were as follows :-

- Sequential Minimum Optimization
- J48
- Random Forrest
- Naïve Bayes
- Naïve Bayes Multinomial
- Simple logistics

**Sequential Minimum Optimization.**

This classifier implements John Platt's sequential minimal Optimization algorithm for training a support vector classifier. This implementation transforms nominal attributes to binary ones. Coefficients are based on the normalized data. [1].

**J48**

This is a decision tree generating classifier that based on the C4.5 algorithm. It generates pruned and unpruned C4. C4.5 is an extension of quinlan's earlier ID3 algorithm. J48 builds decision trees from labelled training data, using the information entropy. The logic used here is that each data attribute can be used to make a decision by the data splitting further into smaller subsets. [2]

**Naive Bayes classifiers**

Naïve bayes classifiers are a family of simple probabilistic classifiers based on applying bayes theorem with strong (naïve) independence assumptions between the features. [3]

Two naïve bayes classifiers were experimented with, the **Naïve Bayes Classifer** and the **Multinomial Naïve bayes classifer.** the naïve bayes classifier refers to the conditional independence of each of the features in the model while the multinomial Naive Bayes classifier is a specific instance of a Naive Bayes classifier which uses a multinomial distribution for each of the features. [4]

**Random Forests**

This is the generalization of random decision forests that are an ensemble learning technique for classification. The classifier constructs a forest of random trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual tree [2]

**Simple Logistic**

This is a classifier for building linear logistic regression models

These classifiers were selected because they have a good reputation in weka. Each classifier was run on data that was passed through a filter using Weka.classifiers.meta.FilteredClassifier.

We then used the features to optimize the best model for each subtask. These features were tokenizers and confidence factor.

**Modeling subtask A**

Models based on different classifiers were built using the supplied training and test set. The best performing model based on accuracy was then singled out and optimized further, investigating different parameters like tokenizers and confidence factor. Training data was uploaded via the pre-processing panel in weka.

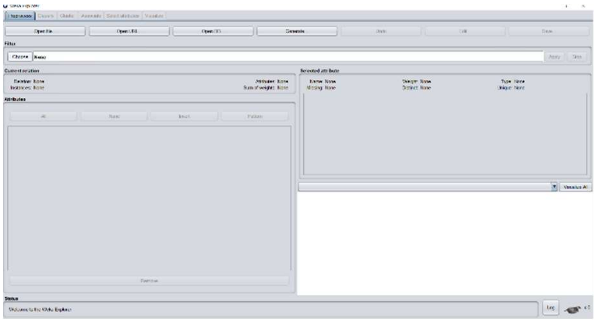


*Figure 1: preprocess*

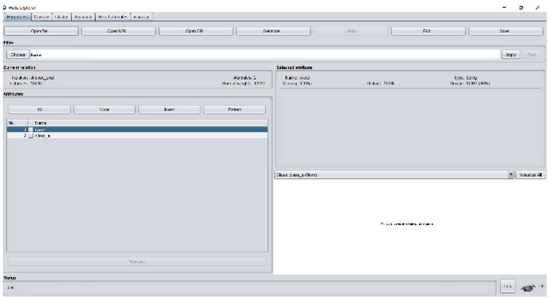


*Figure 2: preprocess*

The *meta.FliteredClassifier* was selected so that classification and filters were done at the same time. The filter used was string to word vector along with the 6 different classifiers

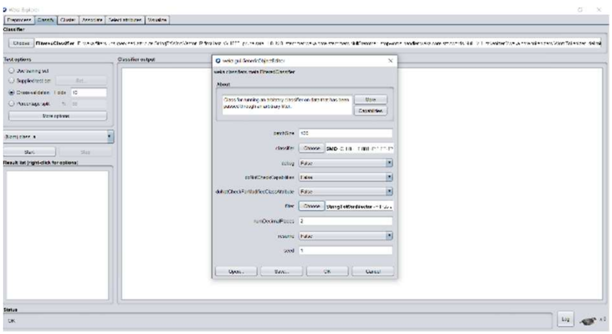
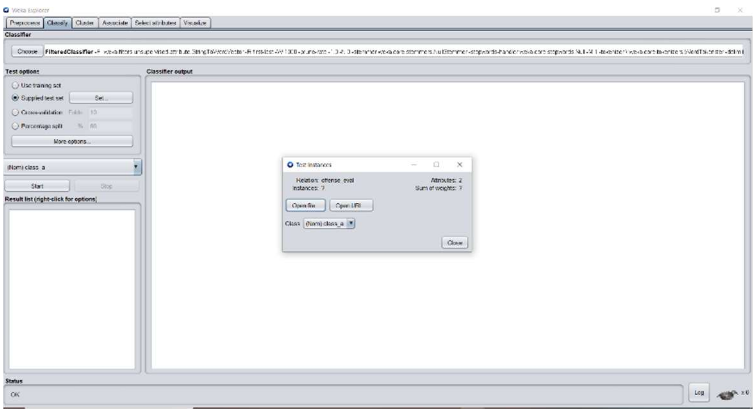


*Figure 3: Classify*

Before we ran the model we chose the supplied test set option and the provided test dataset was used.



### Modeling Subtask B and C

The same methodology that was used in the preprocessing and classification for subtask A was used in Subtask B and C.

The model performance was saved for every classifier and comparative analysis was done using their results. The figure below shows the results of our best performing classifier for subtask A

```
Correctly Classified Instances         700               81.3953 %
Incorrectly Classified Instances       160               18.6047 %
Kappa statistic                          0.4715
Mean absolute error                      0.186
Root mean squared error                  0.4313
Relative absolute error                 43.6808 %
Root relative squared error             95.4916 %
Total Number of Instances              860

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.463    0.050    0.782      0.463    0.581      0.498    0.706     0.512     OFF
                 0.950    0.538    0.820      0.950    0.880      0.498    0.706     0.815     NOT
Weighted Avg.    0.814    0.401    0.810      0.814    0.797      0.498    0.706     0.731

=== Confusion Matrix ===

   a   b   <-- classified as
 111 129 |   a = OFF
  31 589 |   b = NOT
```

*Figure 5: modelling output*

### Evaluation

The key metrics used to evaluate the performance of the models were:-

- Accuracy which is the measure of correctly classified tweets

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

- F1 score which is the measure that combines precision and recall. It is the harmonic mean of precision and recall.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

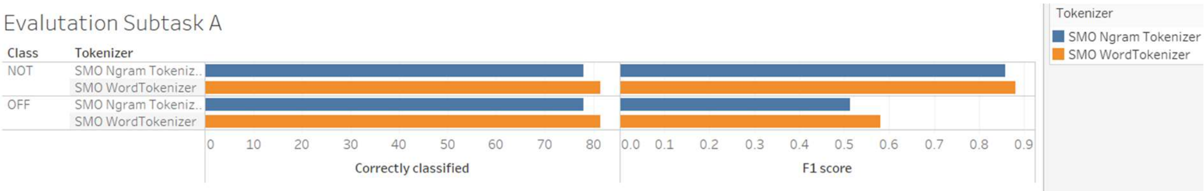
Where Tp is true positive which is the result where a model correctly predicts that a tweet is offensive. tn which is true negative which is the result where a model correctly predicts that a tweet is not offensive. Fp which is false positive which is the result where the model incorrectly predicts an offensive tweet and fn which is false negative which is the result where the model incorrectly predicts a non-offensive tweet.

Below is the evaluation for task A. from the table you can see that the best performing classifier was SMO, which had the highest F1 score and accuracy.

| Class | classifier | Correctly classified % | TP rate | FP | Precision | Recall | F1 score |
|---|---|---|---|---|---|---|---|
| OFF | SMO | 81.40 | 0.46 | 0.05 | 0.78 | 0.46 | 0.58 |
| | Simple Logistics | 80.12 | 0.44 | 0.06 | 0.75 | 0.44 | 0.55 |
| | Naïve bayes multino.. | 79.65 | 0.45 | 0.07 | 0.72 | 0.45 | 0.55 |
| | Random Forrest | 79.53 | 0.35 | 0.03 | 0.80 | 0.35 | 0.49 |
| | J48 | 74.65 | 0.48 | 0.15 | 0.55 | 0.48 | 0.51 |
| | Naïve Bayes | 57.79 | 0.44 | 0.37 | 0.32 | 0.44 | 0.37 |
| NOT | SMO | 81.40 | 0.95 | 0.54 | 0.82 | 0.95 | 0.88 |
| | Simple Logistics | 80.12 | 0.94 | 0.56 | 0.81 | 0.94 | 0.87 |
| | Naïve bayes multino.. | 79.65 | 0.93 | 0.55 | 0.81 | 0.93 | 0.87 |
| | Random Forrest | 79.53 | 0.97 | 0.65 | 0.79 | 0.97 | 0.87 |
| | J48 | 74.65 | 0.85 | 0.53 | 0.81 | 0.85 | 0.83 |
| | Naïve Bayes | 57.79 | 0.63 | 0.56 | 0.74 | 0.63 | 0.68 |

*Table1 : Results for subtask A*

Following the comparative analysis, we singled out the best classifier for optimization

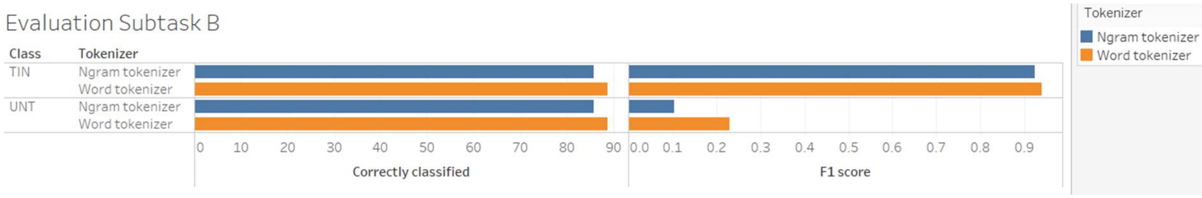



We used different tokenizers and the word tokenizer performed better than the Ngram tokenizer

Below is the comparative analysis for subtask B

| Class | classifier | Correctly classified & | Incorrectly classified % | F1 score | TP rate | FP Rate | Precision | Recall |
|---|---|---|---|---|---|---|---|---|
| TIN | SMO | 88.75 | 11.25 | 0.94 | 0.98 | 0.85 | 0.90 | 0.98 |
| | Simple L.. | 88.75 | 11.25 | 0.94 | 1.00 | 1.00 | 0.89 | 1.00 |
| | Random .. | 88.33 | 11.67 | 0.94 | 1.00 | 1.00 | 0.89 | 1.00 |
| | J48 | 88.33 | 11.67 | 0.94 | 0.99 | 0.96 | 0.89 | 0.99 |
| | Naïve Ba.. | 87.92 | 12.08 | 0.94 | 0.99 | 0.96 | 0.89 | 0.99 |
| | Naïve Ba.. | 83.33 | 16.67 | 0.90 | 0.88 | 0.56 | 0.93 | 0.88 |
| UNT | SMO | 88.75 | 11.25 | 0.23 | 0.15 | 0.02 | 0.50 | 0.15 |
| | Simple L.. | 88.75 | 11.25 | | 0.00 | 0.00 | | 0.00 |
| | Random .. | 88.33 | 11.67 | 0.00 | 0.00 | 0.01 | 0.00 | 0.00 |
| | J48 | 88.33 | 11.67 | 0.07 | 0.04 | 0.01 | 0.33 | 0.04 |
| | Naïve Ba.. | 87.92 | 12.08 | 0.07 | 0.04 | 0.01 | 0.25 | 0.00 |
| | Naïve Ba.. | 83.33 | 16.67 | 0.38 | 0.44 | 0.12 | 0.32 | 0.44 |

A lot of classifiers performed well in this task, but SMO again came out on top. Again we singled it out for optimization, using different tokenizer to optimize our model.




Again the best performing optimizer was Word tokenizer.

Below is the comparative analysis for subtask C

| class | Classifier | Correctly classified | Incorrectly classifi.. | F1 score | Tp rate | Fp rate | precision | recall |
|---|---|---|---|---|---|---|---|---|
| GRP | J48 | 65.26 | 34.74 | 0.64 | 0.58 | 0.13 | 0.71 | 0.58 |
| | Naïve Bayes | 63.85 | 36.15 | 0.64 | 0.67 | 0.24 | 0.61 | 0.67 |
| | Naïve bayes multino.. | 63.85 | 36.15 | 0.62 | 0.59 | 0.18 | 0.66 | 0.59 |
| | Simple Logistics | 63.38 | 36.62 | 0.63 | 0.59 | 0.16 | 0.69 | 0.59 |
| | SMO | 59.15 | 40.85 | 0.58 | 0.55 | 0.21 | 0.61 | 0.55 |
| IND | J48 | 65.26 | 34.74 | 0.75 | 0.92 | 0.49 | 0.63 | 0.92 |
| | Naïve Bayes | 63.85 | 36.15 | 0.75 | 0.84 | 0.35 | 0.68 | 0.84 |
| | Naïve bayes multino.. | 63.85 | 36.15 | 0.75 | 0.90 | 0.45 | 0.64 | 0.90 |
| | Simple Logistics | 63.38 | 36.62 | 0.74 | 0.89 | 0.47 | 0.63 | 0.89 |
| | SMO | 59.15 | 40.85 | 0.70 | 0.82 | 0.46 | 0.61 | 0.82 |
| OTH | J48 | 65.26 | 34.74 | 0.11 | 0.06 | 0.01 | 0.67 | 0.06 |
| | Naïve Bayes | 63.85 | 36.15 | 0.00 | 0.00 | 0.03 | 0.00 | 0.00 |
| | Naïve bayes multino.. | 63.85 | 36.15 | 0.00 | 0.00 | 0.01 | 0.00 | 0.00 |
| | Simple Logistics | 63.38 | 36.62 | 0.00 | 0.00 | 0.02 | 0.00 | 0.00 |
| | SMO | 59.15 | 40.85 | 0.05 | 0.03 | 0.04 | 0.13 | 0.03 |

The best performing classifier was J48. This was singled out for further optimization.


We used the confidence factor to optimize. The model performs better when the confidence factor is 0.1

## References


1. Eecs.yorku.ca. 2020. *Smoreg*. [online] Available at: <https://www.eecs.yorku.ca/tdb/_doc.php/userg/sw/weka/doc/weka/classifiers/functions/SMOreg.html> [Accessed 12 May 2020].

2. Samal, A., Pani, S. and Pramanik, J., 2016. *Comparative Study Of J48, AD Tree, REP Tree And BF Tree Data Mining Algorithms Through Colon Tumour Dataset*. [online] Ijsrd.com. Available at: <http://www.ijsrd.com/articles/IJSRDV4I31613.pdf> [Accessed 12 May 2020].

3. Chedella, S., 2020. *Intuition Behind Naive Bayes Algorithm & Laplace(Additive)Smoothing*. [online] Medium. Available at: <https://medium.com/analytics-vidhya/intuition-behind-naive-bayes-algorithm-laplace-additive-smoothing-e2cb43a82901> [Accessed 14 May 2020].

4. Stuart J. Russell and Peter Norvig. 2003. Artificial Intelligence: A Modern Approach (2 ed.). Pearson Education. *See p. 499 for reference to "idiot Bayes" as well as the general definition of the Naive Bayes model and its independence assumptions*