My Text Analytics Report For Offensive Speech Classification

Abstract

.

There a quite several offensive words spoken in social media. It is important to detect offensive languages so as to reduce their effect on people. In these work, text from tweets are loaded, preprocessed and classified into either offensive (OFF) or not offensive (NOT) using two different techniques, the first method being the XGBClassifier and the second is the MLP model. The different subsets of data is utilized to train these models and the model is tested on test data to predict the class of the data. The performance of the models are also determined to see which model performed better in their prediction. The XGBClassifier() model outperformed the MLP model when tested on the test data.

1 Materials

The following documents are added to this report:

- **Code**: The link to the code for the project is found **here**
- **Google folder**:All the data, models and output files are found **here in Colab**
- **Presentation**: My Project report presentation is found **here**

2 Model Selection (Task 1)

The models selected are

- XGBClassifier
- Multilayer Perceptron (MLP) model

2.1 Summary of 2 selected Models

2.1.1 XGBClassifier

This classifier is a component of the eXtreme Gradient Boosting (XGBoost) package. It is a training technique that is very scalable and prevents overfitting. It can manage missing data and has a variety of hyperparameters that can be modified to optimize the model's performance. It is also renowned for its features of speed, accuracy, and scalability.

2.1.2 Multilayer Perceptron (MLP) model

High-dimensional data can be handled by MLP models, which can also learn word representation in a vector space. The model's adaptability enables the addition of further layers and activation functions, which enables them to recognize intricate patterns in text. When trained correctly, the model performs well on classification tasks.

2.2 Critical Discussion and Justification of Model Selection

Both models were trained on different subsets of the train dataset.The dataset utilized for the training, validating and testing of these models are from (Zampieri et al., 2019). Method 1 utilizes the XGBClassifier while method 2 utilized the Multilayer Perceptron (MLP) model. In method 1, the cross validation score (based on accuracy) with cv = 10 was carried out on the model to serve as hyperparameters tuning, ensuring the model is generalized.The mean of the validation score and the accuracy of the model of validation data were approximately the same and higher on test data. This means the model performs well on new data.The performance metrics of the models on validation data when trained with 100% of training data are shown in table 1 below:

Model	Accuracy	Precision	Recall	F1
XGB	0.77	0.77	0.77	0.76
MLP	0.80	0.80	0.80	0.79

Table 1: Performance Metrics on validation data with model training on 100% training data

Figure 1 and Figure 2 shows the classification report on validation data when the model is trained with 7100% of the trainset.

	precision	recall	f1-score	support
0	0.78	0.93	0.85	619
1	0.76	0.46	0.57	308
accuracy			0.77	927
macro avg	0.77	0.69	0.71	927
weighted avg	0.77	0.77	0.76	927

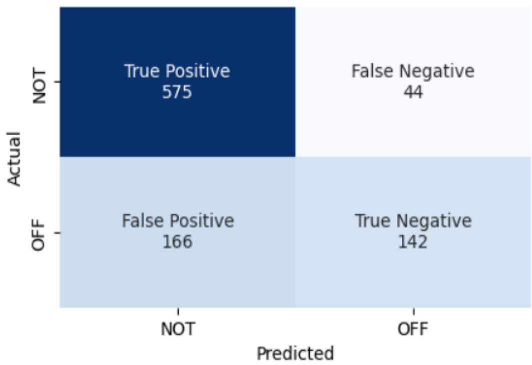

Figure 1: Method 1: Confusion Matrix on model validation using validation data

	precision	recall	f1-score	support
0	0.81	0.91	0.86	619
1	0.76	0.58	0.66	308
accuracy			0.80	927
macro avg	0.79	0.74	0.76	927
weighted avg	0.80	0.80	0.79	927

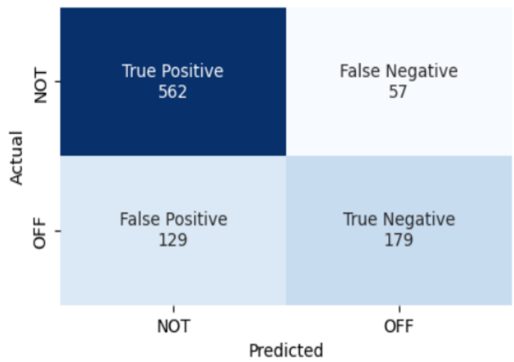

Figure 2: Method 2: Confusion Matrix on model validation using validation data

The model's performance on test data are discussed on further sessions.

3 Design and implementation of Classifiers (Task 2)

The data (training data, test data and validation data) were cleaned and preprocessed by going through the following processes:

- Tweet preprocessor: Tweets may contain URL's, Mentions, Hashtags, Emojis, Smileys and specific words. This process removes unwanted all these unwanted characters.
- The characters of the data are all converted to lowercase.
- Extra white spaces are removed
- Stop words removal: Some frequent and common words are removed and does not affect the meaning of the words
- Stemming: This process convert words to their root word without affecting the meaning of the words.

All these process improves the performance of the models. After the proprocessing, the data, which are in object type, are converted or vectorized for the model to understand. The features of the data are extracted with a feature of 5000 and utilized for model training. Different subsets of the training data are utilized to train the model and validated with the validation data. Prediction of the data label are also carried out on the test data with these models. Details of model training and the data are discussed in section 4. The model pipeline was designed and implemented as shown in Figure 3:

4 Data Size Effect (Task 3)

The dataset utilized for the training, validating and testing of this model is the dataset from (Zampieri et al., 2019). The training dataset named train.csv has about 12313 entries, the test dataset has about 860 entries and the validation dataset has about 970 entries.Table 2 shows the details of the data.Table 2 shows the details of the data subsets The training

Dataset	Total	% OFF	% NOT
Train	12313	66.7	33.2
Valid	927	66.7	33.2
Test	860	72.1	27.9

Table 2: Dataset Details

data with 12313 entries was divided into 4 subsets: 25%, 50%, 75% and 100%. Each subset is the exact percentage of the train data and is utilized to train the models. The models are validated on the validation dataset (valid.csv). The models was tested on the test data, predicting the label of the data. Table 3 shows the dataset statistics of different sizes.

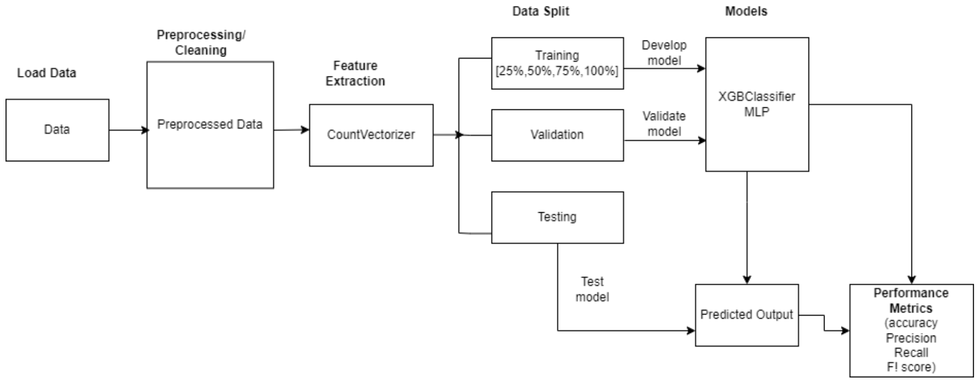

Figure 3: Design and Implementation of the Models

Data %	Total	% OFF	% NOT
25%	3078	33.24	66.76
50%	6125	33.24	66.76
75%	9219	33.24	66.76
100%	12313	33.23	66.77

Table 3: Train Dataset Statistics of Different Size

4.1 Method 1: XGBClassifier()

This model was trained with different training data sizes. The cross validation score of the model's accuracy with cv = 10 was implemented to fine tune the model. Tables 4,5,6 and 7 shows the performance metrics with the for each subset of data. When compared with the mean cross validation score, the model is well generalized. Figures 4,5,6 and 7 shows the classification report and the confusion matrix of the model performance metrics on test data for different data sizes. From these figures, it is evident that when the model is trained with 75% and 100% of the data, the model performs better than others data sizes for both testing and validation. This demonstrates how utilizing more training data can improve the model's performance.

25%Data	Accuracy	Precision	Recall	F1
Validation	0.75	0.74	0.75	0.72
Testing	0.81	0.81	0.81	0.79

Table 4: Performance metrics for 25% subset of train data

The cross validation scores on the test data for each model trained with different data sizes are approximately 91%. This means the model is performing well on new data and well generalized.

50%Data	Accuracy	Precision	Recall	F1
Validation	0.76	0.76	0.76	0.74
Testing	0.81	0.81	0.81	0.79

Table 5: Performance metrics for 50% subset of train data

75%Data	Accuracy	Precision	Recall	F1
Validation	0.78	0.78	0.78	0.76
Testing	0.82	0.82	0.82	0.80

Table 6: Performance metrics for 75% subset of train data

100%Data	Accuracy	Precision	Recall	F1
Validation	0.77	0.77	0.77	0.76
Testing	0.81	0.81	0.81	0.79

Table 7: Performance metrics for 100% subset of train data

4.2 Method 2: Multilayer Perceptron

The Multiplayer Perceptron is a neural network model that was trained with two linear transformation layers, one ReLU activation function layer, one dropout rate of 0.2 and a learning rate of $3e^{-5}$. The loss function of the training and validation data was computed and plotted as shown in Figure 8, 9, 10 and 11. These losses decreases as the epoch increases. The model is learning to better capture the underlying patterns in the data without overfitting to the training data.

Tables 8,9,10 and 11 shows the performance metrics of the models on validation data and predictions on test data.

The performance metrics of the models is not generalized on test data when trained with data sizes 25% and 50% as shown in Table 8 and 9.

	precision	recall	f1-score	support
0	0.81	0.97	0.88	620
1	0.82	0.40	0.54	240
accuracy			0.81	860
macro avg	0.81	0.69	0.71	860
weighted avg	0.81	0.81	0.79	860

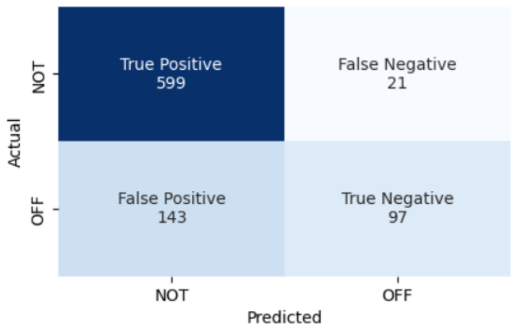

Figure 4: Metrics on test data with 25% data size

	precision	recall	f1-score	support
0	0.81	0.96	0.88	620
1	0.81	0.42	0.55	240
accuracy			0.81	860
macro avg	0.81	0.69	0.71	860
weighted avg	0.81	0.81	0.79	860

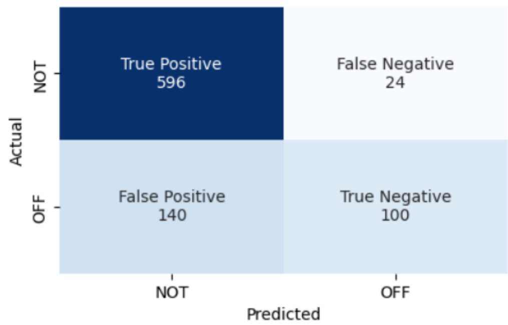

Figure 5: Metrics on test data with 50% data size

	precision	recall	f1-score	support
0	0.82	0.96	0.88	620
1	0.82	0.45	0.58	240
accuracy			0.82	860
macro avg	0.82	0.71	0.73	860
weighted avg	0.82	0.82	0.80	860

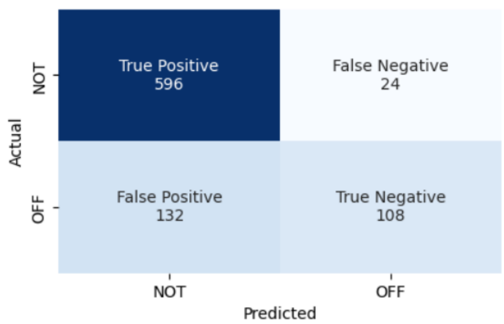

Figure 6: Metrics on test data with 75% data size

	precision	recall	f1-score	support
0	0.82	0.96	0.88	620
1	0.80	0.44	0.57	240
accuracy			0.81	860
macro avg	0.81	0.70	0.73	860
weighted avg	0.81	0.81	0.79	860

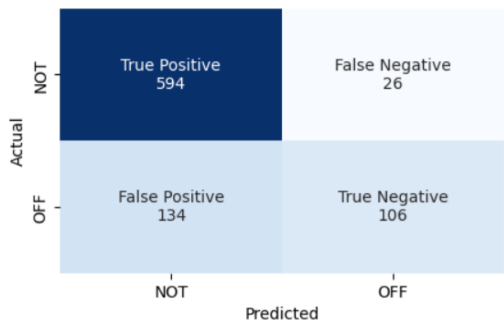

Figure 7: Metrics on test data with 100% data size

Figure 8: Loss vs epoch with 25% data size

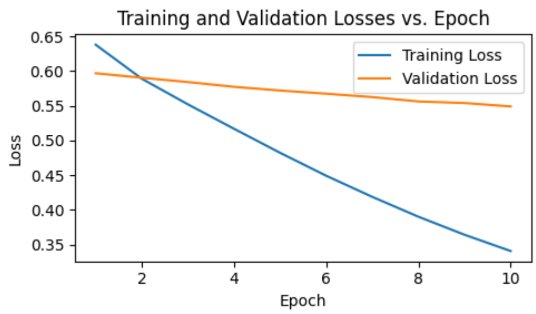

Figure 9: Loss vs epoch with 50% data size

This means these models are overfitting. Although the model still overfit,increasing the training data to 75% and 100% improved the performance of the models as shown in Tables 10 and 11. For model trained with 100% of the data, the performance of the model is 79% on test data which an improved performance when compared with models performance on test data when trained with other data sizes. The model's performance can be further improved if trained with more data and if the parameters of the model is further tuned.

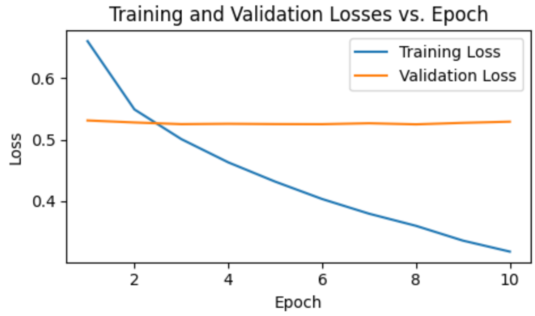

Figure 10: Loss vs epoch with 75% data size

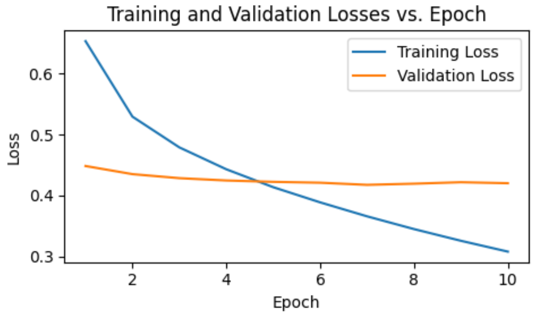

Figure 11: Loss vs epoch with 100% data size

25%Data	Accuracy	Precision	Recall	F1
Validation	0.68	0.71	0.68	0.56
Testing	0.72	0.52	0.72	0.60

Table 8: Performance metrics of model using 25% subset of data

50%Data	Accuracy	Precision	Recall	F1
Validation	0.74	0.73	0.74	0.72
Testing	0.70	0.61	0.70	0.62

Table 9: Performance metrics of model using 50% subset of data

75%Data	Accuracy	Precision	Recall	F1
Validation	0.74	0.73	0.74	0.73
Testing	0.70	0.61	0.70	0.62

Table 10: Performance metrics of model using 75% subset of data

100%Data	Accuracy	Precision	Recall	F1
Validation	0.80	0.80	0.80	0.80
Testing	0.79	0.78	0.79	0.78

Table 11: Performance metrics of model using 100% subset of data

4.3 Comparing the two methods with tweet examples

Some examples are selected from the test data to see how the models predicted the labels of the data. Table 13 shows the comparison of the model's performance on prediction when trained with 100% of the training data. Added to the table are possible labels from the researcher and remarks.

5 Summary (Task 4)

5.1 Discussion of work carried out

Two different methods were utilized to classify text as either offensive (OFF) or not offensive(NOT). Method 1 used a machine learning model named XGBClassifier to classify the texts. The model was well generalized with an accuracy of approximately 81% on test data. Method 2 utilized a neural network algorithm named Multilayer Perceptron. Two neural network layers, one dropout layer of 0.2, and a learning rate of $3e^{-5}$ was utilized to create the model. After training the model with different sizes of the train data, the performance of the model on test data was higher when trained with 100% of the data with accuracy of about 79%. Method 1 performed better on test data with an accuracy of 81% on same data size.

5.2 Lessons Learned

Some traditional machine learning models outperformed the neural network or deep learning models. This is evidence in both methods utilized. The performance of method 2 can be improved by changing the parameters of the model which may include the learning rate, the optimizer, additional drop out layer and additional hidden layers. The disadvantage of this is that the model might become more complex, starts to overfit and very expensive.

Example	GT	M1(100%)	M2(100%)	Possible Label	Remark
gets all the bitches	OFF	OFF	OFF	OFF	GT label correct. Offensive towards women'
@USER School shootings aren't controversial. Wanting gun control is.	NOT	NOT	NOT	NOT	GT label correct. Comment not offensive
@USER $500,000 for her wedding. Clinton's always know what's best.	NOT	NOT	NOT	NOT	GT label correct, comment not offensive
voice in my head while I transfer money to my credit card bullshit that I bought and have to pay for	OFF	OFF	OFF	OFF	GT label correct, offensive word used
Are you fucking serious? URL	NOT	OFF	OFF	OFF	GT label mistake. Vulgar word used in text

Table 12: Comparing two Models using 100% data

Example	GT	M1(25%)	M1(50%)	M1(75%)	M1(100%)	Possible label
And these entertainment agencies just gonna have to be an ass about it. URL	OFF	OFF	OFF	OFF	OFF	OFF
#MonbebeSelcaDay she is beautiful URL	NOT	NOT	NOT	NOT	NOT	NOT
#MAGA Expect more censorship URL	NOT	NOT	NOT	NOT	NOT	NOT
#Conservatives @USER - You're a clown! URL	OFF	NOT	NOT	NOT	NOT	NOT
@USER #metoo are all racist!	OFF	NOT	OFF	OFF	OFF	OFF

Table 13: Comparing Model Size: Sample Examples and model output using Model 1 with different Data Size

Example	GT	M2(25%)	M2(50%)	M2(75%)	M2(100%)	Possible Label
And females b on the same shit URL	OFF	NOT	NOT	NOT	OFF	OFF
. . . What the fuck did he do this time?	OFF	NOT	NOT	NOT	OFF	OFF
#LiberalLogic If #Liberals get their way..... URL	NOT	NOT	NOT	NOT	NOT	NOT
#Spaldo is my guy. He is my godson	NOT	NOT	NOT	NOT	NOT	NOT
#2A GUN CONTROL KILLS PEOPLE, THIS IS WHY WE NEED CONCEALED CARRY URL	NOT	NOT	OFF	OFF	OFF	NOT

Table 14: Comparing Model Size: Sample Examples and model output using Model 2 with different Data Size

References

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.