An Efficient Approach For Text Classification Using Ensemble Models in Machine Learning

AXEL GORIS RAFAT KHAN
goris | rafatk @kth.se

September 21, 2022

Abstract

In machine learning, the models process given inputs and produce an outcome. The outcome is a prediction based on what pattern the models finds during the training process. Machine learning algorithms have their limitations and producing a model with high accuracy is challenging. In many cases, one model is not enough for producing a reliable prediction because of the high variance of data which might result in low accuracy as model relies heavily on too few features while making a prediction. Instead, if we build and combine multiple models, we have the chance to boost the overall accuracy. The combination of models will reduce the model error and maintain the generalization of the model by aggregating the output from each model. In ensemble learning architecture, the weak learners receive inputs as well as predictions from each weak learner. A final ensemble model can be built using the combined predictions.

Contents

1 Data **2**
1.1 Analysis of our Dataset 2
1.2 Pre-Processing 3

2 Models **6**
2.1 KNN : K Nearest Neighboors 6
2.2 Naives Bayes 7
2.3 XGBoost 7
2.4 Result 7
2.4.1 Confusion Matrix 7

List of Acronyms and Abbreviations

1 Data

For this research, we want to compare the accuracy and performance of different models and an ensembling model on the same dataset. Said dataset must be a text dataset because we are working with sentiment analysis on a text but it does not have to have any particular property. The only requirements is that we use the same dataset for all of our models. We have decided to use a COVID dataset [1] because it has been used multiple times before and it is a manually tagged dataset of Covid related tweets. The creator tagged every tweet with one of the five following tag: Extremely Negative - Negative - Neutral - Positive - Extremely Positive.
In the current form, the dataset is too large for us to train our models on (memory error) so we will first need to pre-process them. Before doing that, we will analyse our dataset [2]. For the analysis and pre-processing of our datas, we have decided to use Python.

1.1 Analysis of our Dataset

Our dataset is already split in two parts: a training dataset containing 41 157 entries and a test dataset containing 3 798 entries. The dataset looks like that:

Figure 1: Example of our dataset

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3799	48751	London	16-03-2020	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...	Neutral
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	Positive
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	Positive
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp...	Positive
4	3803	48755	NaN	16-03-2020	Me, ready to go at supermarket during the #COV...	Extremely Negative

We can also find that they are 9424 missing locations. However, since we are going to drop the column, this does not matter.
For the rest of the analysis, we will regroup ”Extremely Positive” and ”Positive” as ”Positive” and ”Extremely Negative” and Negative” as ”Negative”.First, we find out that the classes are not evenly distributed, with positive being bigger than the other two classes (”Negative” and ”Neutral”).

Figure 2: Classes Distribution

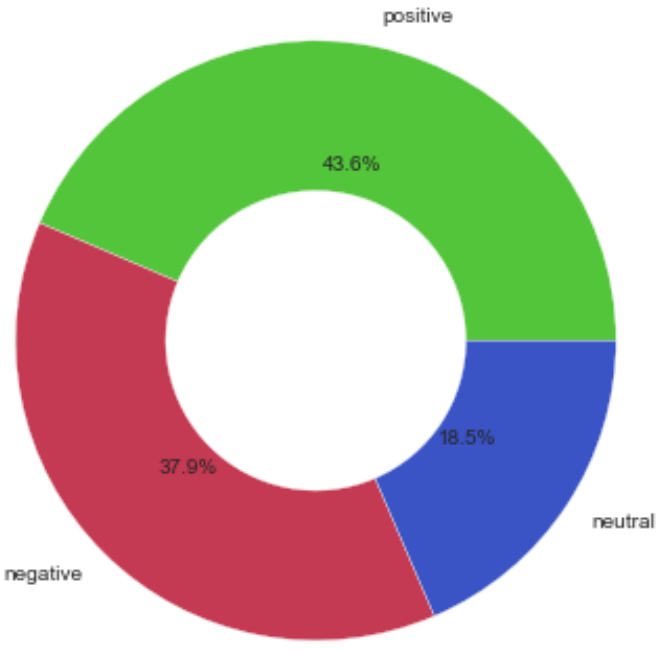
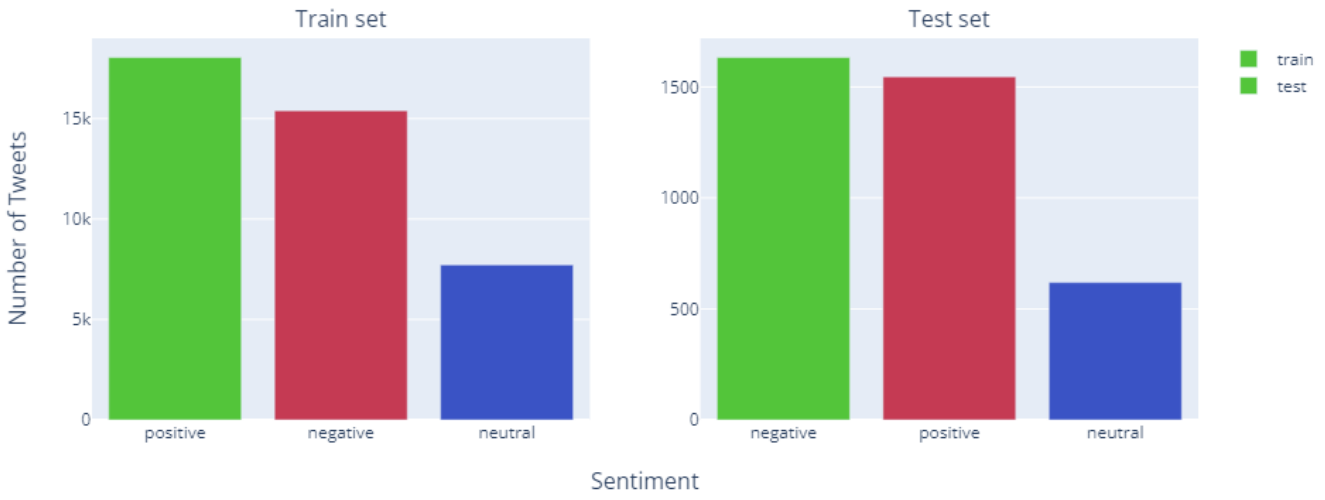
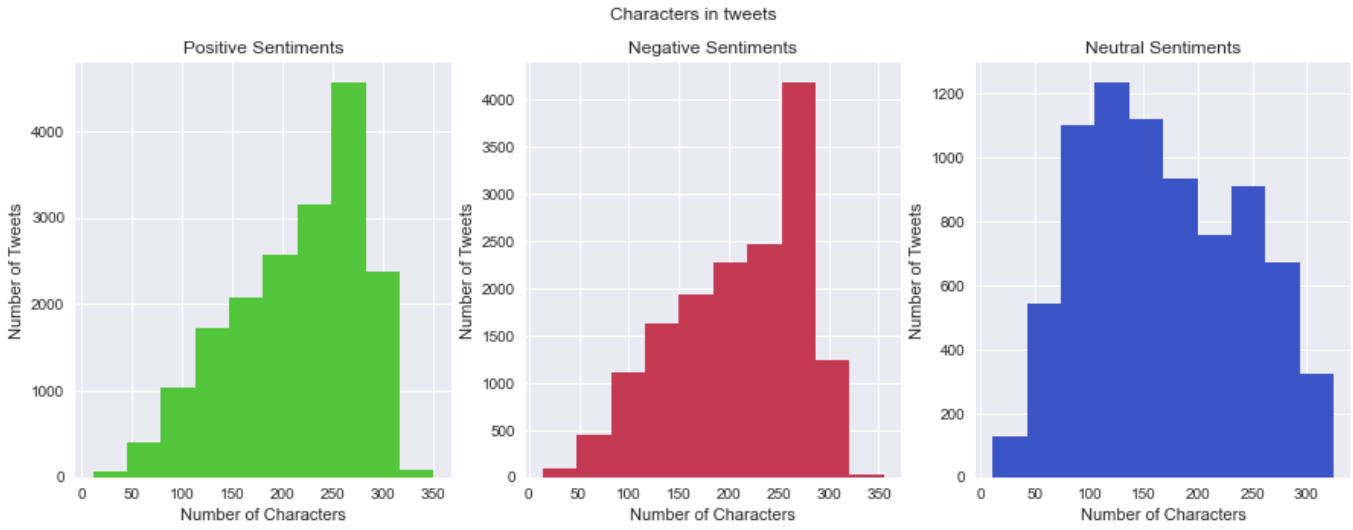


Figure 3: Classes Distribution Bar



We found out that the classes in a tweet are related to the number of characters and words in a tweet. Moreover, the classes also depends on the average word length in a tweet.

Figure 4: Numbers of characters in a tweet depending on the sentiment



Some words are called ”Stopwords”: they are very present in our daily language but they don’t convey useful meaning for sentiment classification. Here are the most used words in the dataset for the positive sentiment. The other two classes follow a similar ranking.
Same as the stopwords, urls, punctuations, mentions do not convey any valuable meaning and are very present in our dataset.

Thanks to the previous analysis, we know that there is a lot of data to clean up.

1.2 Pre-Processing

To try improving the sentiment analysis, we followed the steps of pre-processing of the flow 4 in [3] which provided the best accuracy for Naive Bayes sentiment analysis. The following steps have been executed to reduce the amount of data in our dataset and make it easier to use.

- Change everything to lower case so that ”Covid” and ”COVID” are treated the same for example.

Figure 5: Number of words in a tweet depending on the sentiment

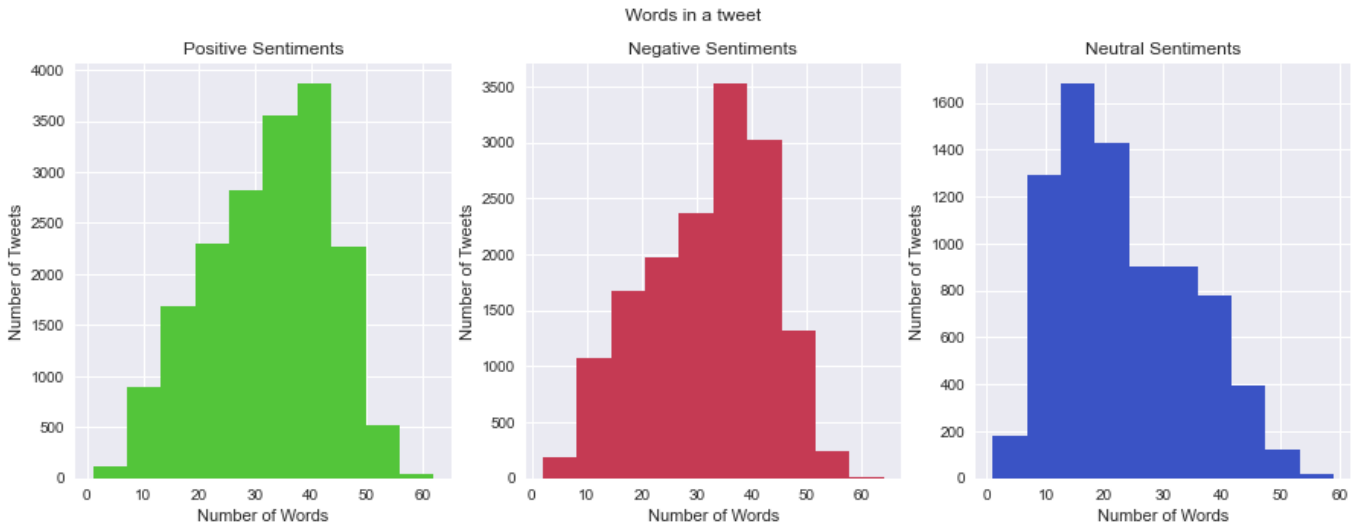
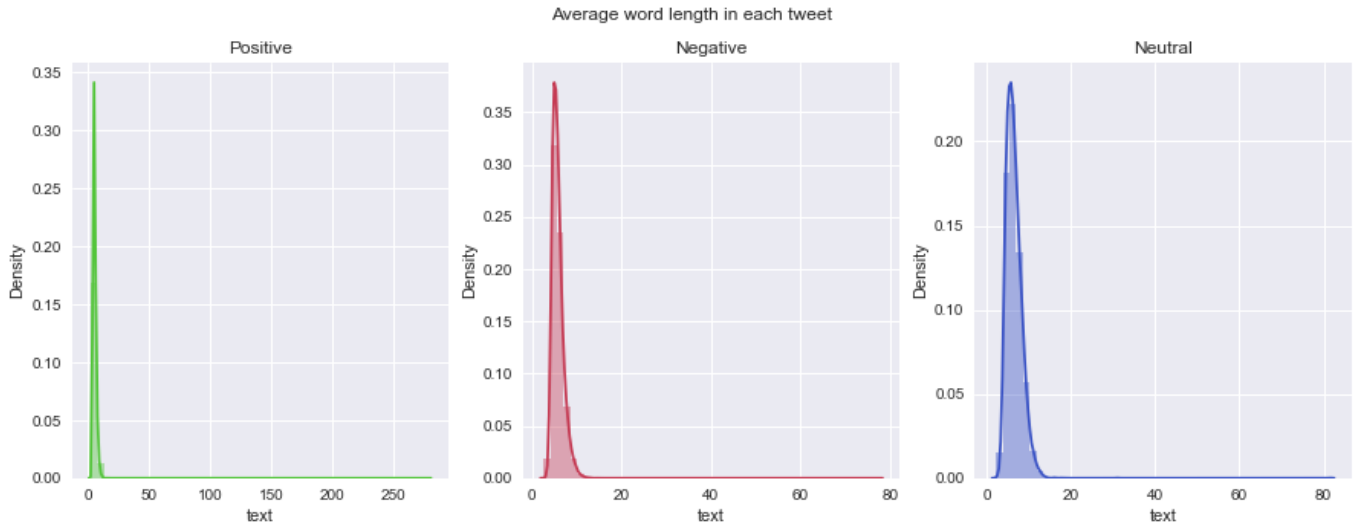


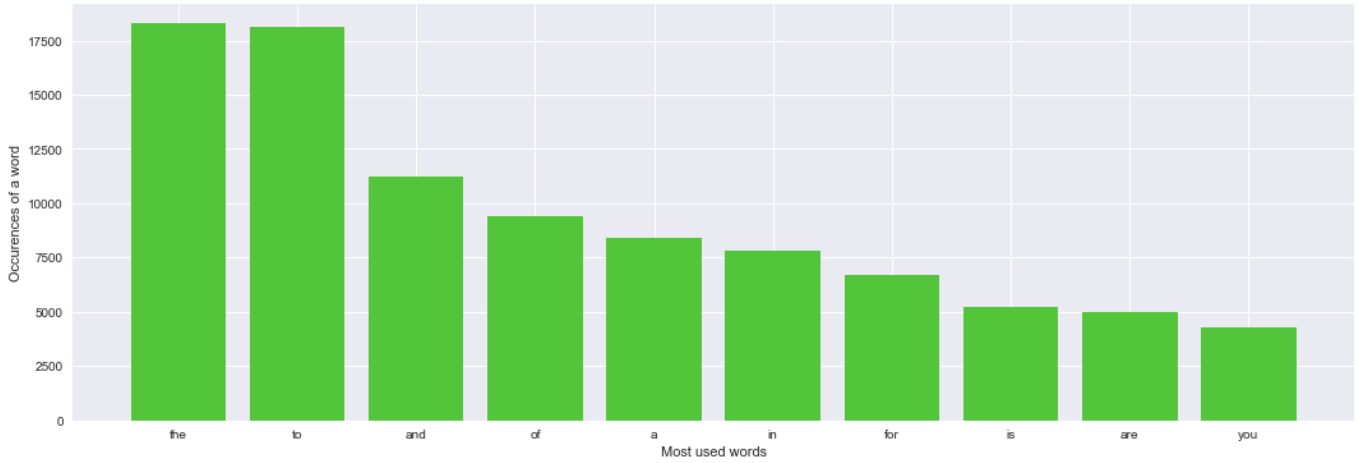
Figure 6: Length of words in a tweet depending the sentiment



- Remove all of the URL and HTML character from the tweets.
- Expand all contractions. ”Don’t” becomes ”Do not”. So on and so forth.
- Remove punctuation.
- Remove digits because they don’t convey meaning for a sentiment analysis.
- Remove stopwords. They happens quite often in our dataset and don’t convey any valuable information.

By processing our data, we were able to use them with our models without running out of memory, which was our first issue with this dataset. Moreover, we know can get some wordclouds to have an idea of what are the most used words after our pre-processing.

Figure 7: Length of words in a tweet depending the sentiment



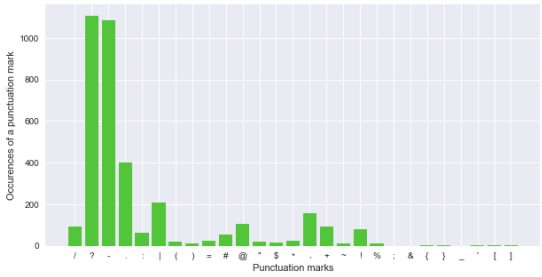


Figure 8: Punctuations occurrence in the positive class

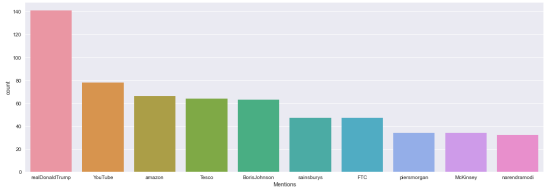
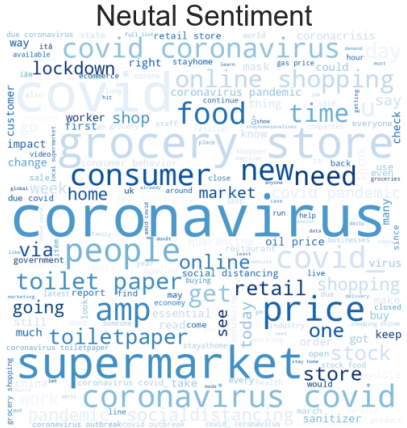


Figure 9: Mentions occurrences in the dataset

Figure 10: Words clouds depending on the tweets sentiment



2 Models

For our analysis, we will use 3 models and we will ensemble them and compare the performance in our final report. The 3 selected models are : KNN - Naives Bayes - XGBoost. In order to analyse our results, we will use a Confusion Matrix [4]. A Confusion Matrix evaluates the number of times a model is giving the correct answer or confuses it with another class. In our cases, we have 5 classes so the confusion Matrix will be a 5*5 matrix. The correct results are on the diagonal of the confusion Matrix. Considering we have a multi-class problem we will consider each class as a binary classification problem. It is whether right or wrong for each class.
In order to calculate some properties, we will use the concept of True Negative, True Positive, False Negative, False Positive.

Predicted	Reality	Type	Explanation
True	True	True Positive (TP)	Prediction was positive : Result was positive
False	False	True Negative (TN)	Prediction was negative : Result was negative
True	False	False Positive (FP)	Prediction was positive : Result was negative
False	True	False Negative (FN)	Prediction was negative : Result was positive

We will be interested in the Macro average of each class to have more of a "macro" point of view for the 4 following properties.

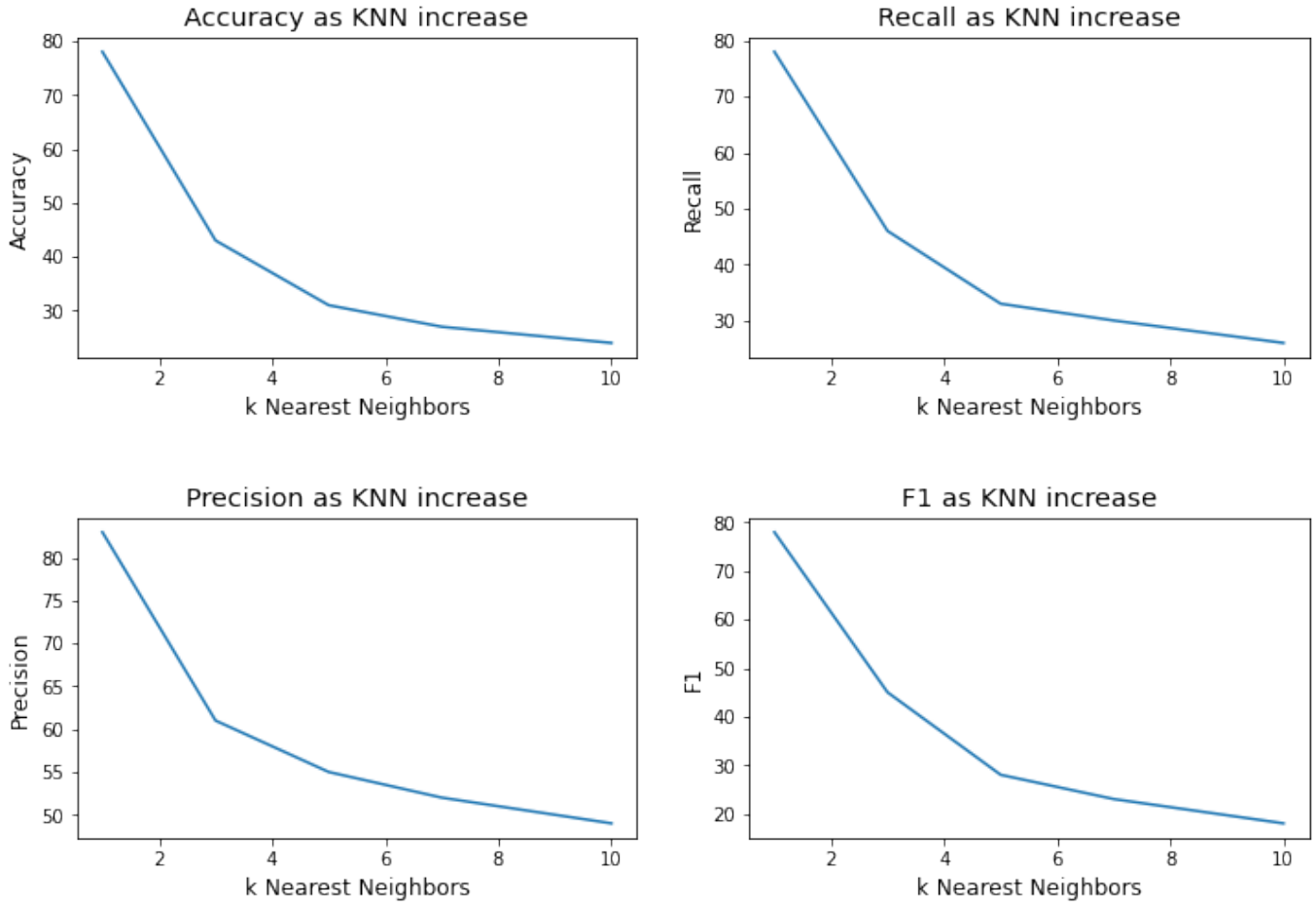
1. Accuracy : Measure of how many times our model made a correct prediction. Good metric to measure the overall performance of our model.
$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$
2. Precision : Measure of how many positive predictions were True Positive.
$$Precision = \frac{TP}{TP+FP} \tag{2}$$
3. Recall : Measure of how many positives were correctly predicted, over all positive cases.
$$Recall = \frac{TP}{TP+FN} \tag{3}$$
4. F1-Score [5] : Harmonic mean of Precision and Recall, helping balancing the two previous metrics.
$$F1-Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$
5. MacroAverage : Average on each class for each of the previous property. Example with $m = NumberOfClasses$:
$$MacroAverageF1 = \frac{\sum_{n=1}^{m} F1_n}{m} \tag{5}$$

For each of the model, we will use the pre-processed dataset for training and testing, with the available implementation of sklearn.

2.1 KNN : K Nearest Neighboors

To make a prediction, the K-NN algorithm will base itself on the entire dataset [6]. Indeed, for an observation, which is not part of the dataset, that we want to predict, the algorithm will look for the K instances of the dataset closest to our observation. Then for these K neighbors, the algorithm will use their output variables y to calculate the value of the variable y of the observation we want to predict.
A knn model depends a lot on the value of K: K is the number of neighboors to consider for classification. A low value of K means that our model will have a low Bias and a High Variance (risk of overfitting). A high value of K means that our model will have a high Bias and a low Variance. However, our data being text with few features in common, $k = 1$ gives us the best result.

Figure 11: Influence of K on our metrics



2.2 Naives Bayes

The naive Bayesian classification method is a supervised machine learning algorithm that classifies a set of observations according to rules determined by the algorithm itself [7].

2.3 XGBoost

XGBoost stands for Extreme Gradient Boosting. It is a scalable, distributed gradient-boosted decision tree (GBDT). It is using parallel tree boosting to achieve high accuracy and high performance [8].

2.4 Result

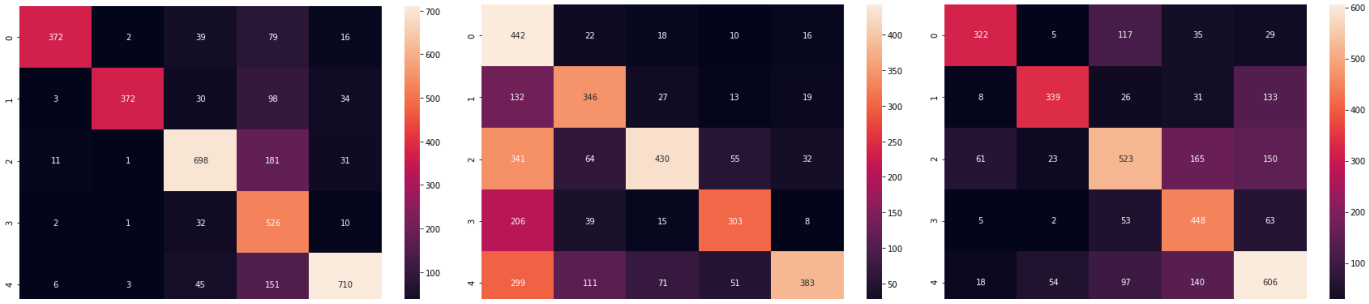
On our dataset, we have different result for each one of our three algorithms. We're using K=1 for the KNN Algorithm.

Macro Average				
	Accuracy	Recall	Precision	F1-score
KNN	78%	78%	83%	78%
Naives Bayes	55%	59%	64%	56%
XGBoost	65%	66%	69%	67%

2.4.1 Confusion Matrix

A confusion matrix is a table that is used to define the performance of a classification algorithm. We can visualizes and summarizes the performance of the algorithm by seeing this confusion matrix. We can compare the confusion matrix of these three algorithms.

Figure 12: Confusion Matrix: KNN (k=1), Naive Bayes, XGBoost.



References

[1] "Text Classification." [Online]. Available: https://kaggle.com/code/matleonard/text-classification

[2] "COVID 19 Tweets EDA & Viz ." [Online]. Available: https://kaggle.com/code/datatattle/covid-19-tweets-eda-viz

[3] M. Palomino and F. Aider, "Evaluating the Effectiveness of Text Pre-Processing in Sentiment Analysis," *Applied Sciences*, vol. 12, p. 8765, Aug. 2022. doi: 10.3390/app12178765

[4] Z. Karimi, *Confusion Matrix*, Oct. 2021.

[5] Y. Sasaki, "The truth of the F-measure," *Teach Tutor Mater*, Jan. 2007.

[6] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Using kNN model for automatic text categorization," *Soft Computing*, vol. 10, Mar. 2006. doi: 10.1007/s00500-005-0503-y

[7] D. Li-guo, D. Peng, and L. Ai-ping, "A New Naive Bayes Text Classification Algorithm," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 12, Sep. 2014. doi: 10.11591/telkomnika.v12i2.4180

[8] T. Chen and C. Guestrin, *XGBoost: A Scalable Tree Boosting System*, Aug. 2016, pages: 794.