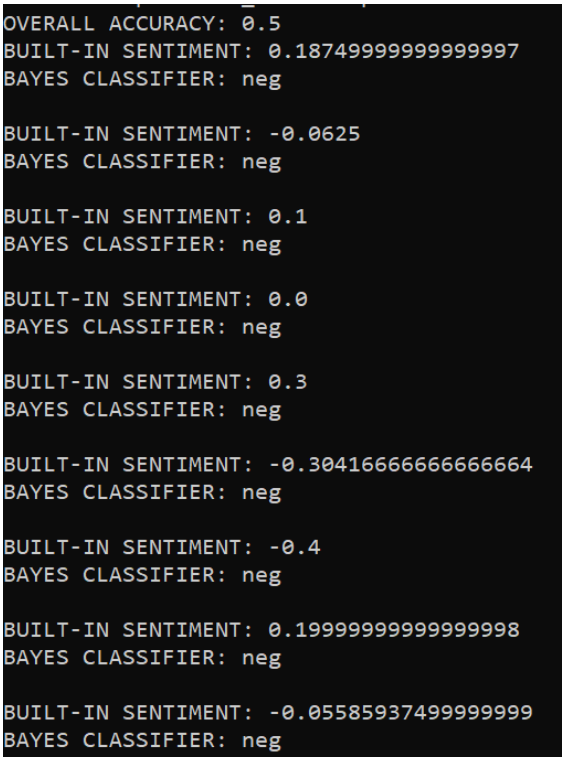
Model Evaluation

As part of our project we have tested a number of prediction models in terms of their accuracy for tweet sentiment classification. We used TextBlob's Python libraries, which contain functions to make different classifiers, such as Naive Bayes and Decision Trees to make classifiers, and used scikit-learn to perform clustering. All of our training and test data consisted of pieces of text under 280 characters long, which is the limit on length for a tweet. Our training and test sets contained real tweets and pieces of texts made by us.

Naive Bayes Classification

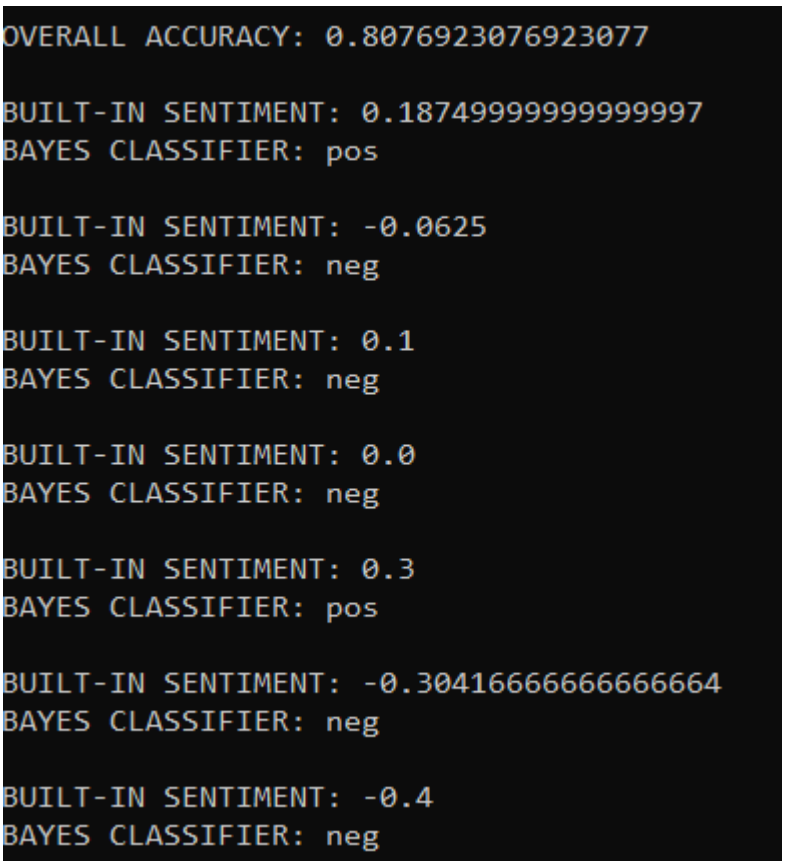
First Steps

When first developing the Naive Bayes classifier, we selected random tweets in our database from different months and manually gave them a sentiment score (positive or negative). We selected 30 tweets for our training data, 15 positives and 15 negatives. For the test data, we selected 16 tweets.
The accuracy of the classification when evaluated was of only 50%. When evaluating the results, we noticed that values were mostly negatives. When further analysing the results, we noticed that most false negatives were only slightly positive. After a meeting with our supervisor where we discussed the accuracy of the model, we were recommended to make a positive training set to improve the accuracy, and thus we updated our training set. Some of the results from the initial test is shown below:



Improving our training and test sets

Our supervisor also mentioned that our test set might not be ambiguous enough and recommended we add unambiguous text to our test data, we added 18 unambiguous pieces of text, 9 positive and 9 negative. We then tested the accuracy of the model again with the updated test data. The recorded accuracy rose to 57%. When analysing the results, we noticed that there were a large amount of false negatives, for this reason, we decided to add 5 unambiguous positive text data made by us to the test set. The results showed an increase to 65% accuracy, but now the results contained more false positives. To balance the ratio of positives to negatives, we decided to add 10 more texts, 5 positive and 5 negative. This resulted in an accuracy of 69%. By this stage, our training set was 24 texts. To increase the accuracy, we increased the size by 10, making 10 more unambiguous texts. The final test set consisted of 34 tweets15 coming from our database and 18 made by us, with half of the tweets being positive and the other half being negative. The results showed a sharp increase in accuracy to 80%, with a smaller number of false negatives being presented in the results.



Decision Tree Classifier

After we reached a high accuracy with the Naive Bayes Classifier, we decided to compare its accuracy with Decision Tree Classification. We used the same training and test data. Our initial results showed an accuracy of 69%. When analysing the classification of each test data one by one, we found that there were many false negatives. We also noticed that the accuracy was not consistent, with results ranging from 65-71% accuracy, changing each time we ran our test.

To improve the accuracy of our model, we decided to add 5 more positive texts to our training data. The results showed similar results, with accuracy of 66%, but this time the differences in accuracy were larger, with a maximum and minimum accuracy of 69% and 62% recorded. We then added 5 negative texts to our training data, meaning we had now added 10 more pieces of texts to our training data. The accuracy was slightly lower at 62%, with minimums of 60% recorded. We then added 5 negative texts to our training data, and removed the 5 positives ones to test for any change. The accuracy was the same as with the original training and test sets.

We concluded that to achieve a reliable accuracy, we would need a much more extensive set of data and testing and Naive Bayes worked better for sentiment prediction.

```
OVERALL Decision Tree Classifier ACCURACY: 0.685714285714285

BUILT-IN SENTIMENT: 0.18749999999999997
DECISION TREE CLASSIFIER: neg

BUILT-IN SENTIMENT: -0.0625
DECISION TREE CLASSIFIER: neg

BUILT-IN SENTIMENT: 0.1
DECISION TREE CLASSIFIER: neg

BUILT-IN SENTIMENT: 0.0
DECISION TREE CLASSIFIER: neg

BUILT-IN SENTIMENT: 0.3
DECISION TREE CLASSIFIER: neg

BUILT-IN SENTIMENT: -0.30416666666666664
DECISION TREE CLASSIFIER: neg

BUILT-IN SENTIMENT: -0.4
DECISION TREE CLASSIFIER: neg
```

K-Nearest Neighbour (KNN) Classification

After measuring the accuracy of two classifiers, we decided to test the accuracy of a model that uses Euclidean Distance in its algorithm for sentiment analysis with the same training and test sets. The first clustering method we implemented was K-Nearest Neighbour (KNN). KNN is a classification model, but is also defined as clustering due to its reliance on proximity and similarity measures. It estimates the classification of an unseen instance by finding the k closest training instances and selecting the most commonly occurring classification among them. To predict the sentiment of the text we first had to transform the dictionary into a 2D matrix for the clustering to be able to be performed. Depending on the number of neighbours we defined, the accuracy of the model changed.

At the default value of 2, the average accuracy was 54%, when analysing the results, we noticed that all of the wrong predictions were false negatives. With a k value at 3, the accuracy decreased to 49%. At k values of 4 the accuracy was 51% and at 5 the accuracy remained at 46%.

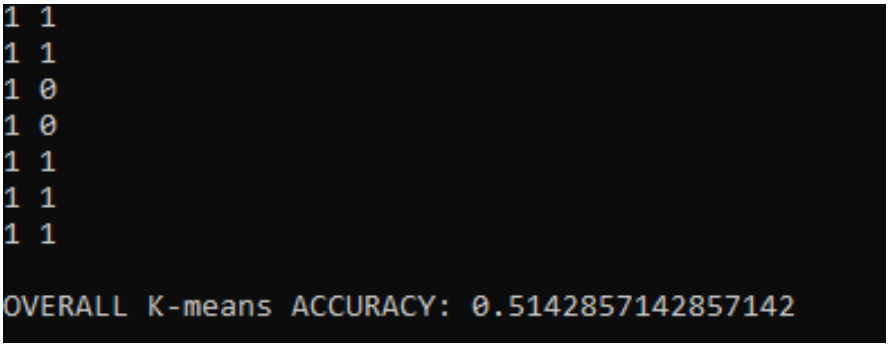
```
neg neg
neg neg
neg pos
neg pos
neg pos

OVERALL KNN ACCURACY:0.5428571428571428
```

K-means Clustering

The K-means clustering algorithm works by selecting a value k, which are the initial k centroids. Each object is assigned to the cluster for which it is nearest to the centroid. The centroids of the k clusters are then recalculated. These last two steps are repeated until the centroids no longer move.

The last clustering method we evaluated was K-means clustering. Again, we used the same training and test sets. Like with KNN, to predict the sentiment of the text we first had to transform the dictionary into a 2D matrix for the clustering to be able to be performed. We used the same training and test sets as our Naive Bayes Classifier model. Our initial tests with a k value of 2 showed an average accuracy of 51%, but had results as low as 46%. At k = 3, the accuracy was slightly lower with consistent results of 49% and a minimum of only 2% recorded once. At k=4, accuracy had a max value of 60% but we noted a minimum value of only 5%, meaning that 2 out of the 34 tests were accurately predicted. Analysing the results to examine if there were more false positives or negatives was difficult due to the fact that the categorical labels had been converted to numerical values for the matrix for K-means to be produced. Positive values were transformed into “1” and negative values into “0”. The results contained a large number of false positives.



Evaluation of Clustering model accuracy

After our results for KNN and K-means clustering we concluded that models such as K-means or KNN are not suitable for our type of dichotomous data. This is because the Euclidean distance is used in both models to select what clusters objects belong to. In the case of K-means clustering, because the data is either true or false, this means that on each step where the Euclidean distance has to be measured to choose which cluster an object belongs to will often result in many objects having the same value and each object will be assigned to a cluster by random.

Built-In Textblob Classifier

Our next step was to compare the accuracy of the models we trained up to now with TextBlob's "polarity()" function. We measured the accuracy of the function with the same test set used for the other models. The accuracy was 86%, higher than all of the models we made. For this reason, we decided to use this as our model to graph the sentiments.

```
OVERALL Built-in Polarity function ACCURACY: 0.8571428571428571
```

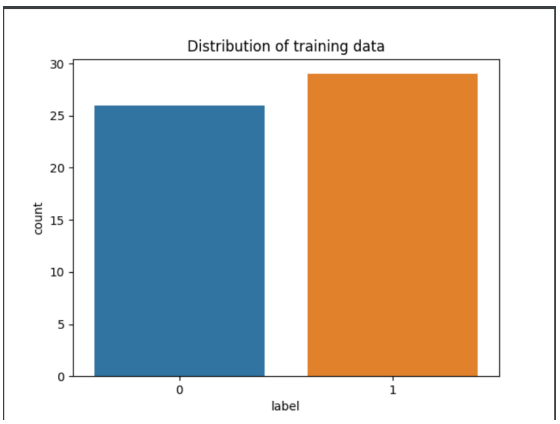
Logistic Regression Model

Logistic regression (LR model) is a statistical model used to analyse the relationship between a dependent variable (also known as the response variable) and one or more independent variables (also known as predictor variables). We believed that it would be suitable for sentiment analysis as it is also a binary classification algorithm like Naïve Bayes and Decision Trees. Our research suggests that it performs close to or better than Naïve Bayes for our use case. The LR model can output a probability score for each class, which can be interpreted as the likelihood of the input text belonging to that class. In sentiment analysis, the binary classes can be positive or negative.

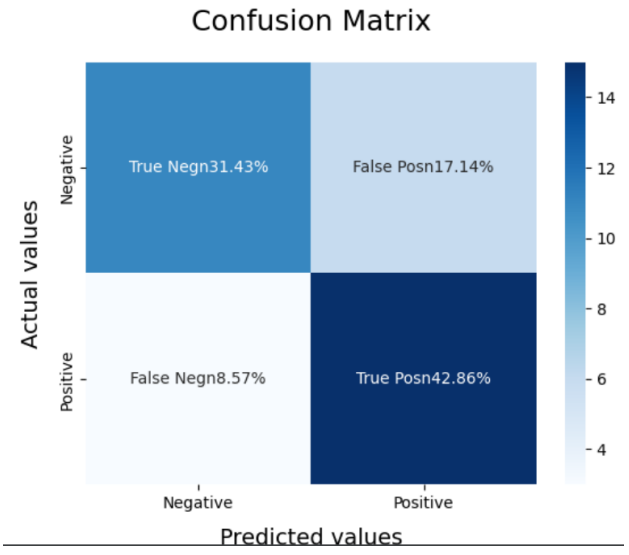
Another advantage of logistic regression is its ability to handle sparse data, which is common in text analysis, where the number of features (i.e., words or phrases) can be very large compared to the number of training examples. Moreover, logistic regression is computationally efficient and relatively easy to interpret, which makes it a popular choice for sentiment analysis applications.

Training and Evaluation

Initially, we used a small set of 55 tweets to train our model and 35 test tweets. The results were as follows:



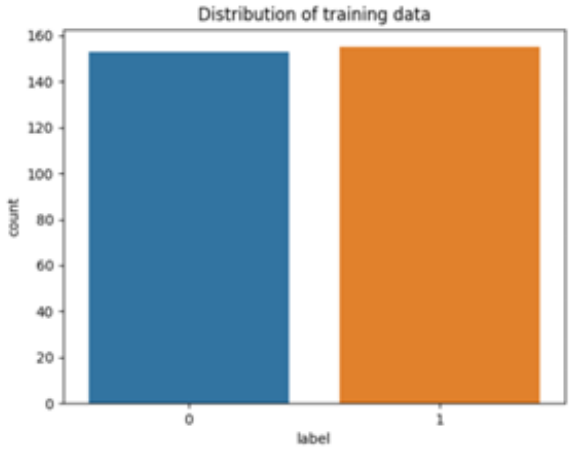
	precision	recall	f1-score	support
0	0.79	0.65	0.71	17
1	0.71	0.83	0.77	18
accuracy			0.74	35
macro avg	0.75	0.74	0.74	35
weighted avg	0.75	0.74	0.74	35



We concluded that this performance was inadequate and after some research, we suspected the cause was the low amount of training data. We also found that our training data itself was problematic in the way that it was inherently biassed, being centred around tweets concerning Covid-19. There was likely to be a

higher distribution of negative tweets than positive tweets and this was interfering with the model's ability to learn patterns in the data, leading to many false positives and negatives in the confusion matrix.

We expanded our training set to contain over 300 tweets, now including tweets from the Sentiment140 dataset that is widely used to train sentiment analysis classifiers. Our new training set was more balanced as illustrated below with almost equal amounts of positive and negative data.

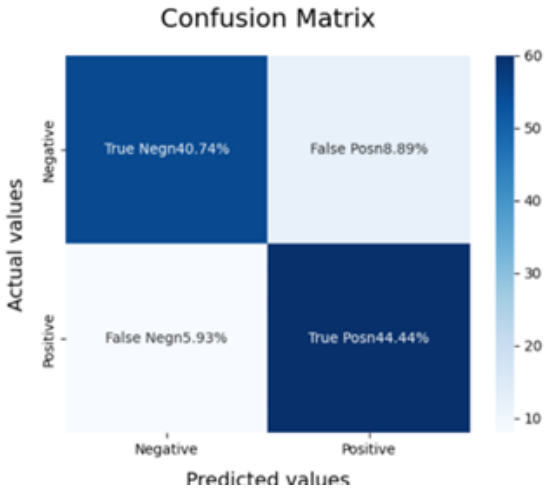


(0: neg, 1: pos)

Our test set was also expanded, now containing about 135 tweets. Using the new training data, we found that our accuracy metrics including precision, recall and F1 score were greatly improved.

	precision	recall	f1-score	support
0	0.87	0.82	0.85	67
1	0.83	0.88	0.86	68
accuracy			0.85	135
macro avg	0.85	0.85	0.85	135
weighted avg	0.85	0.85	0.85	135

Our confusion matrix also saw large reductions in false positives and negatives.



Further Analysis and Update of Training and Test Sets

After developing the linear regression model, we realised that both the test and training sets were too small to achieve high accuracy in the model. We found a publicly available dataset on Kaggle with tagged tweets containing sentiment scores(positive or negative) [1]. We collected and added 250 tweets to the training set, and 100 tweets to the test set. We determined that a larger test set would demonstrate the performance of the developed models more accurately. We added a total of 250 tweets to the training set for all of the other models too. We ran the tests again with the test set now containing 135 tweets for each model, and the resulting accuracies are as follows:

- Naive Bayes Classifier: 83% accuracy - increased by 2% from previous tests.4
- Logistic Regression: 85% accuracy - increased by 11% from initial tests.
- Decision Tree Classifier: 77% accuracy - increased by 8% from previous tests.
- K-Means Clustering: 50% accuracy - remained the same as in previous tests.
- KNN Clustering: 65% accuracy - same accuracy results as in previous tests
- TextBlob's Built-In Classifier: 90% accuracy - increased by 4% from previous tests.

The change in accuracy demonstrated not only the consistency of TextBlob, but it also showed that the other predictors improved from the larger training data produced. For K-means, the big increase might be because of the increased training set, but we determined that the results were still not reliable enough to make a consistent prediction.

References

Kaggle dataset used to update our test set: https://www.kaggle.com/datasets/kazanova/sentiment140