Gerald Joshua Virality_predictor

What features did you consider?

- Features I considered when building this modal were the virality metric, views, bookmarks, likes, comment created and follow.
- I then used the virality column I created to form a new column called viral chances with two attributes: high or low. If the virality of an article is above 500 (500 inclusive) then the chances are High and low if otherwise. (See the csv files created in the csv folders.)

What model did you use and why?

- RandomForest modal applying the random-forest-classifier algorithm. This because Random forests is a set of multiple decision trees making it highly accurate and a robust model. It does not suffer from overfitting of the data, unlike decision trees, making it better at predicting results and It can also handle missing variables as there is a few missing variables in the data obtain.

What was your evaluation metric for this?

- Since I made it a classification problem, the metrics used for evaluation are accuracy, confusion matrix, precision recall, and F1 values. (See results on csv in files directory)
  Precision: The ratio of correct positive predictions to the total predicted positives.
  Recall: probability of detection.
  F1 Score is the weighted average of Precision and Recall. takes both false positives and false negatives into account.

```
    accuracy                           0.65      1031
   macro avg       0.18      0.18      0.17      1031
weighted avg       0.65      0.65      0.64      1031

0.6459747817652765
Accuracy: 0.6459747817652765
```

The accuracy achieved with 1000 trees is 65%. Increasing the number of estimators might have increased the accuracy.

```
[[29  0  0 ...  0  0  0]
 [ 0 68  0 ...  0  0  0]
 [ 0  0 41 ...  0  0  0]
 ...
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  1  0]]
```

Confusion matrix extract describing the performance of a classification model on a set of test data for which the true values are known.

What features would you like to add to the model in the future if you had more time?

- Article title is another nice feature that I would have liked to add. Sometimes it is the title that makes a person read the article in the first place.
- Language is another feature that I would have considered.
- I might have removed comment created from features created since it does not actually symbolize that this article will go viral

What other things would you want to try before deploying this model in production?

- There is a lot more to be done before deploying it into production. I need to study the beedly system and find out how they will be feeding the model new articles to check if they are going to go viral or not. With that said, I need to see how I can incorporate my model with other systems from other platforms
- A way to visualize at what percentage is my prediction of the article going viral will be.
- I need to design a database to store my results for easy access. The storing of the result in csv won't work in the production world. The database of my choice would be a MongoDB database because the data obtained from platforms such as Beedly might be unstructured and might require joins. MongoDB eliminates the need of joins and thus becomes less expensive compared to using SQL databases/ structured schema databases. There would be a database request handler script, responsible in handling and modeling the data stored in the database.
- Try other models and compare my results to see which one is more accurate.
- Creating an endpoint that will allow my model to receive the new Articles that will be accessed to check on if they will go viral or not.
- Try to remove the least important features so as to increase the accuracy and decrease misleading data and noise.
- Look into writing an extensive test for my system that creates the model and any other features that I might add, like a function that waits for reading platforms to input their new article data for prediction and more.