Categorising Political Bias using Machine Learning Methods

Paul Long
School of Computing
National College of Ireland
Dublin, Ireland
Email: x22166475@student.ncirl.ie

Abstract

Politics is an innate part of human society. Data and Media have become massively important debates in the late teens and early 20s of the 2000s – particularly arounds the use of data to market certain political ideologies to people.

This project attempts to look at how these actions were undertaken by using Machine Learning to classify respondents in a survey as to their political leaning and classifies a newspaper into its political leaning based on the political leaning of the newspaper it came from.

Keywords – Machine Learning, Political Bias, Classification, Natural Language Processing, Count Vectorizer, Term Frequency Inverse Document Frequency, K Nearest Neighbours, Decision Tree, Random Forest, Gradient Boosting, Support Vector Machine, Neural Network

I. Introduction

Background

This report is based on a project which classified survey responses [1] and newspaper headlines [2] based on their political leaning. The political leanings for the surveys were 'Left' and 'Right', and the political learnings for the newspaper headlines were 'Centre-Left', 'Centre', 'Centre-Right' and 'Right'.

The project looked at two different types of data: numerical (converted from categorical) for the survey data, and textual data from the newspaper headlines.

Note: some explanations for the parameters are given within the code. The code took 11 minutes and 30 seconds for the text part of the project from a fresh Jupyter Notebook, and the survey part of the project took 2 minutes.

Project Aims

The project utilized a multitude of different Machine Learning algorithms for the classification. The project also included some optimization of certain machine learning methods based on different parameters included. Since the newspaper headlines used text data, each model was run twice using two different text vectorizers:

- 'Cross Vectorization', also known as bag of words, which counts the words used as is.
- 'Term Frequency - Inverse Document Frequency', which adds some information on the importance of each word.

This projected aimed to transform and analyse the data using machine learning methods, and then comparing the methods on how accurately they classified the test data. The methods used in the machine learning analysis included:

- Logistic Regression
- K Nearest Neighbours
- Linear Discriminant Analysis
- Decision Tree (using C5.0 and CART)
- Random Forest
- AdaBoost
- XGBoost
- Naïve Bayes
- Support Vector Classification
- Neural Network

Project Issues

It should be noted here that some of the survey data that could have been relevant may have been excluded. This included auxiliary data such as 'age', which is often conflated with political view , rows answer 'who they would vote for in the next election' (which had a very large number of missing data), and 'their worldview' (which was too complex to convert to meaningfully analyse.

Some of the analysis of survey data included integers that were converted from categorical responses. For example, there were multiple times where "Strongly Agree" through to "Strongly Disagree" were converted to 2 to -2. This could have caused some rounding errors and there is also inherent error in the question, as people answer based on their own interpretation.

Data Description

The survey data was a survey of about 11000 people from Europe and was described as "A post-election survey about populism in the US and the EU-28". This project only focused on that data regarding the EU.

The newspaper data was a list of 33000 headlines from 15 UK newspapers, including: BBC, Sun, Mirror, Daily Mail, Independent, Guardian, Manchester Evening News, Sky News, Metro, Telegraph, Daily Express, Times, Liverpool Echo, Birmingham Live, and Evening Standard. After a lack of data from Centre papers (400 entries), the centre paper (Sky News) was excluded. The dataset was described as "Headlines generated by the top 15 UK news websites over 20 days".

II. Literature Review

The literature on this topic was quite rich. An article from Medium provided a particularly good comparison, which analysed the textual data from UK Newspaper Articles [3]. The article used similar methods in this one and achieved 95% accuracy on some of the methods. However, it should be noted that the article from Medium focused on full sections of text, which had a lot more to draw from than this dataset did, so an expectation of 95% accuracy is unrealistic.

A similar project was undertaken using American Newspapers, which seems to focus on sentences designed for political testing [4]. This project used Recursive Neural Networks and Recurrent Neural Networks, and a Hidden Markov Model, in addition to the MLP model from SkiKitLearn, which was used as the Neural Network in this project.

A similar project was undertaken using Newspaper headlines in Telugu [5]. The Machine Learning methods included: Naive Bayes, SVMs, CNNs, Branched CNNs, LSTMs and GRUs.

Another notable example of using machine learning for political bias, which isn't strictly relevant to this project, is the 2016 Cambridge Analytica scandal [6], which was a real-world application of identification of political bias or potential for political influence. Cambridge Analytica famously used Machine Learning to evaluate and identify Facebook users for the purposes of marketing of right-wing political data to influence the 2016 election.

III. Methodology

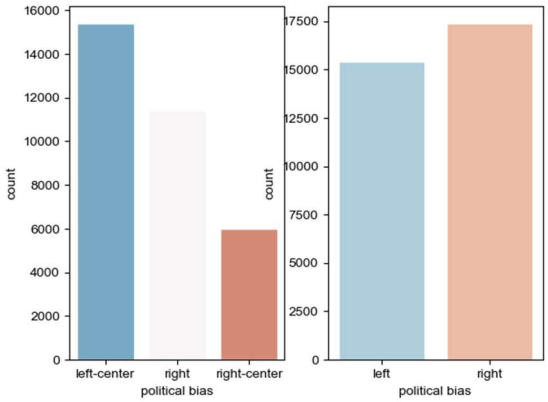
Text Analysis

Focusing first on the text data, the two dataframes were read into Python – the primary dataset and the auxiliary dataset which contained information about each Newspaper – political bias, ownership, journalism style and each paper's party support for the 2019 General Election. The datasets were then merged on 'Newspaper' to ensure each headline had an entry for its paper's political bias. Irrelevant columns were dropped.

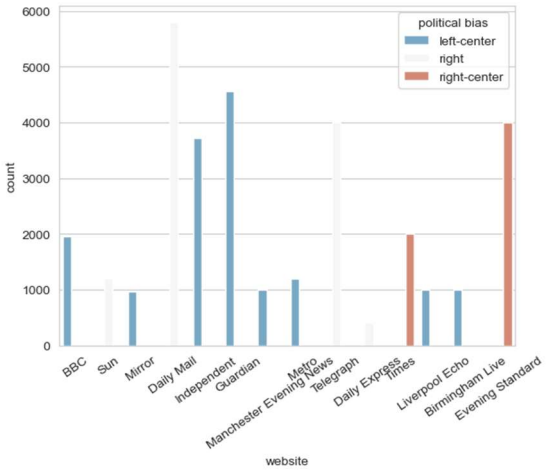
The centre papers were then excluded, as they made up such an insignificant amount of the data – only 400 headlines. The data was tested for NaN values, and it was found that there

were entries for all the data. The unique values were checked to ensure every row had a valid entry.

A copy of the dataframe was made, which converted the three political leanings – "Left-Centre", "Right-Centre" and "Right" into "Left" and "Right". This was only created for visualisation. It was found that there was a roughly even split between 'Right' and 'Left' newspaper headlines, with 'Centre-Left' having the highest number when accounting for all the biases. Of the three biases, there was roughly double the 'Centre-Left' headlines as there was 'Right' headlines.



Hence, there was a low level of imbalance. The number of headlines by political leanings for both sets of data were calculated and visualised. Another visualisation was created which showed each paper's number of headlines and which political bias the paper had.



The total number of words in the corpus was calculated at 36173.

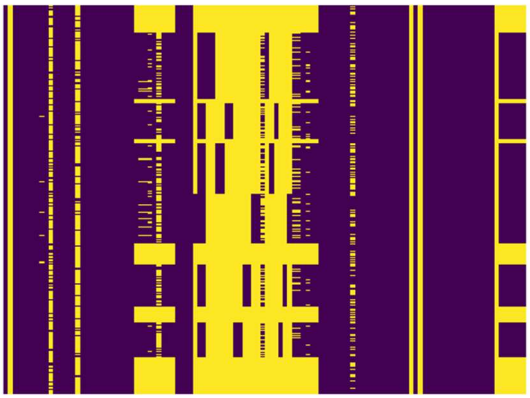
The data was tokenized and lemmatized. The tokens were then converted into vectors using two forms: Count Vectorizer, also known as Bag of Words, and Term Frequency – Inverse Document Frequency. The features lengths for the Count Vectorizer were chosen to be between 1 and 2 – that is 1- and 2-word features, as the data only looked at headlines. The number of features for the TF IDF was chosen to be 15000, as that represented about 40% of the total words in the Corpus.

Subsequently, the machine learning methods were fitted to the data with a train test split ratio of 70:30. The methods were: Logistic Regression, K Nearest Neighbours, Random Forest, Naïve Bayes, and Support Vector Classification, on both the CV and TF-IDF vectorizers. These methods were then evaluated using 'Accuracy Scores' and 'Classification Reports'.

Survey Analysis

Focusing then on the survey data, the dataframes was read into Python. It should be noted here that the data had already underwent a small amount of pre-processing. The data was in Excel format, and so there were 7 pages worth of data that needed to be clicked through. Each page referred to a particular section. The pages that were not necessary were deleted and the data that was useful was then moved to the first page. There were Four pages that had data moved to Python. Both the original copy and the edited copy have been uploaded alongside this report.

The rows were then re-evaluated. There was a very large number of NaN data.

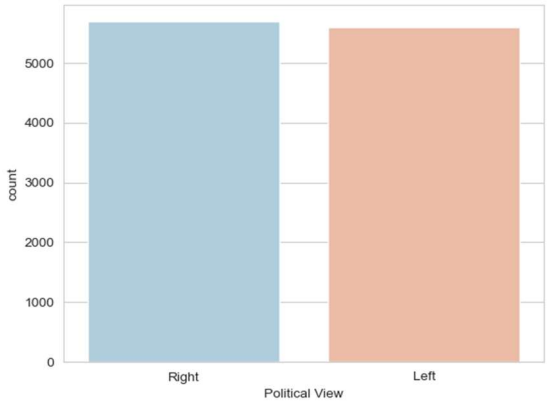


Irrelevant data was excluded. This included some data that may have been indirectly relevant, however they were excluded in ease of simplifying the data processing. These data included demographic information, engagement with types of media, did the interviewee vote in the last election.

Also excluded were questions such as which party did the interviewee vote for in the last election and which party will they vote for in the next election. While this was extremely relevant information, there were different categories for each country, with only six countries listed (Spain, Great Britain, France, Germany, Poland, and Italy. As such, there was a large amount of NaN data, so these columns were excluded also. Similarly, the ranking of political issues of importance also had mostly NaN data, so these columns were excluded, despite also having relevancy.

The data was then tested for NaN values, and it was found that there were no NaN values remaining. This left 40 columns remaining. These responses were all in the form of True or False values, categorical values such as 'Strongly Agree', 'Somewhat Agree', 'Neither Agree nor Disagree', 'Somewhat Disagree', 'Strongly Disagree', or some numerical ranges such as monthly income being 'Under 200', '200 to 400', etc. All of these columns were converted to integer values.

The dependent variable – Political View, had six values – 'Center Left' [sic], 'Extreme Left', 'Center Right' [sic], 'Extreme Right', 'Left' and 'Right'. These were all converted to 'Left' or 'Right, to create a binary classifier. Finally, the data was run through a 100% sampler to shuffle the values around. It was found that the first number of values were all politically left. This was noticed after many of the classifiers had skewed results (94% recall for left and 3% recall for right). These were then graphed to ensure that the dataset was not imbalanced.



Subsequently, the machine learning methods were fitted to the data with a train test split ratio of 70:30. The methods were: Logistic Regression, K Nearest Neighbours (including optimization for the ideal K), Decision Tree (including CART and C5.0), AdaBoost & XGBoost, Random Forest, Linear Discriminant Analysis and Neural Network. These methods were then evaluated using 'Accuracy Scores' and 'Classification Reports'.

IV. Results

Text Analysis

Since there was only a low level of imbalance between the three biases, accuracy was a suitable metric to evaluate each model. A classification report was also produced for some of the models.

It was found that the Logistic Regression for the Count Vectorization had the best accuracy with 74.0%. The next best models were the Support Vector Classifier for CV with 73.4% accuracy, the Random Forest for CV with 72.5% and for TF with 72.2%, and Naïve Bayes accuracy with 72.1% accuracy.

```
LR CV Accuracy: 0.7395060473625369
LR CV Accuracy:
               precision    recall  f1-score   support

 left-center       0.80      0.73      0.76      5117
       right       0.76      0.76      0.76      3412
right-center       0.53      0.72      0.61      1310

    accuracy                           0.74      9839
   macro avg       0.70      0.74      0.71      9839
weighted avg       0.75      0.74      0.74      9839


  Support Vector Machine CV Accuracy : 0.7335095029982722

  Support Vector Machine CV Classification Report :
               precision    recall  f1-score   support

   left-center       0.80      0.72      0.76      5118
         right       0.75      0.76      0.76      3381
  right-center       0.53      0.71      0.61      1340

      accuracy                           0.73      9839
     macro avg       0.69      0.73      0.71      9839
  weighted avg       0.75      0.73      0.74      9839
```

The worst performing model was the K Nearest Neighbours, with 58.5% accuracy for the CV model and 65.3% accuracy for the TF IDF model. This was the only method that didn't have a model with accuracy above 70%. The other models were above 65% and less than 70%.

```
K Nearest Neighbours CV Accuracy: 0.5856286207947963
K Nearest Neighbours CV Classification Report:
               precision    recall  f1-score   support

 left-center       0.95      0.54      0.69      8214
       right       0.20      0.78      0.32       880
right-center       0.37      0.87      0.52       745

    accuracy                           0.59      9839
   macro avg       0.51      0.73      0.51      9839
weighted avg       0.84      0.59      0.64      9839
```

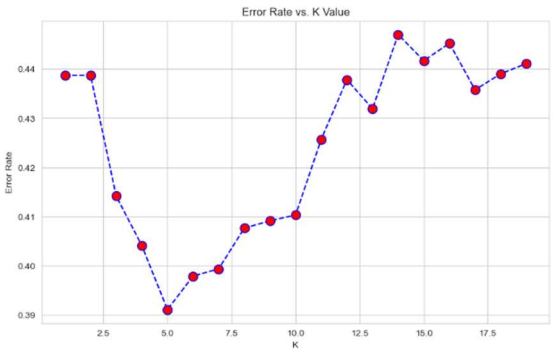
These results suggest that there is room for improvement in the models but given the limits of the project – the data only uses headlines instead of full text and only categorises 'Right-Centre' differently to 'Right', these were reasonably good models. Looking at the classification report for the three models, it was found that the least accurately predicted outcome was the 'Right' political bias, which would suggest the models had difficulties in identifying the nuanced differences between 'Centre-Right' and 'Right.

Survey Analysis

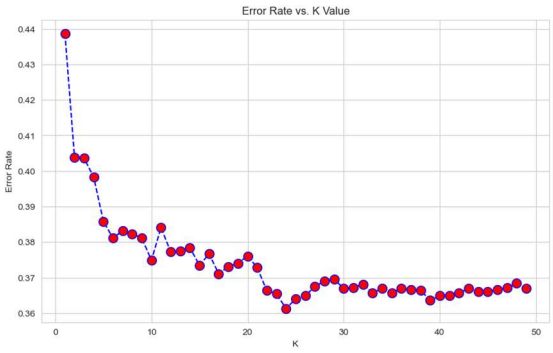
The survey data focused on numerical data into a binary classification. There were roughly equal people considering themselves left and right. Each model was again evaluated using classification reports and accuracy.

The outcomes seemed to have rather poor results. It was found that the majority of the algorithms had methods between 55% and 65%. Logistic Regression had an accuracy of 62% and K Nearest Neighbours had an accuracy of 56%. K Nearest Neighbours optimized for number of neighbours, but it was found that there wasn't a significant difference, with there being an error less decrease of only 2% between 10 and 80 neighbours.

There were two models for Decision Trees – CART, which used 'Gini' as it's splitting criteria, and C5.0, which used 'entropy' as it's splitting criteria. It was found that C5.0 had a better accuracy at 58% compared to 56%. Following this, the decision tree was optimized for depth, based on error loss. It was found that a max depth of 5 was the most optimal parameter.



A similar optimization was done for the AdaBoost, which found that the optimal number of estimators seemed to even out at 37% error after 20 estimators. The AdaBoost had an accuracy of 62%.



Gradient Boosting had an accuracy of 64%. Random Forest and Linear Discriminant Analysis both had accuracies at 62%.

```
Random Forest Accuracy: 0.6407680945347119
RandomForest Classification:
              precision    recall  f1-score   support

         0.0       0.60      0.65      0.63      1568
         1.0       0.68      0.64      0.65      1817

    accuracy                           0.64      3385
   macro avg       0.64      0.64      0.64      3385
weighted avg       0.64      0.64      0.64      3385
```

```
XGB Accuracy: 0.6375184638109306

XGB Classification:
              precision    recall  f1-score   support

         0.0       0.58      0.65      0.61      1489
         1.0       0.70      0.63      0.66      1896

    accuracy                           0.64      3385
   macro avg       0.64      0.64      0.64      3385
weighted avg       0.64      0.64      0.64      3385
```

The neural network had some erratic results. The model would alter between classifying everything as one outcome and giving an expected result. After changing the max iterations to 200,000, this seemed to even out. However, occasionally the model did give the skewed results.

Expected Classification Report:

```
Neural Network Accuracy : 0.6165435745937962
Neural Network Classification Report :
              precision    recall  f1-score   support

         0.0       0.62      0.61      0.62      1712
         1.0       0.61      0.62      0.62      1673

    accuracy                           0.62      3385
   macro avg       0.62      0.62      0.62      3385
weighted avg       0.62      0.62      0.62      3385
```

Skewed Classification Report:

```
Neural Network Accuracy : 0.5025110782865584
Neural Network Classification Report :
              precision    recall  f1-score   support

         0.0       0.96      0.50      0.66      3240
         1.0       0.05      0.58      0.09       145

    accuracy                           0.50      3385
   macro avg       0.51      0.54      0.37      3385
weighted avg       0.92      0.50      0.63      3385
```

Conclusions

Regarding the text data, it was found that across the board, the Count Vectorization was much more successful than TF IDF, with the exception of the worst performing model. It was also found that, despite not much parameter optimization, Support Vector Classification was one the more successful classification methods, followed closely by random forest, which may have been more accurate, but took up far more processing time. The most successful model was the Logistic Regression.

Further research of interest in this topic would be to look at full articles of text to ensure there is more data to classify each political bias. Furthermore, it would be preferrable to find more data that included a 'Left' political bias and had more data on the 'Centre'. Some sampling could also be done to ensure that there is an even amount of data for each political bias.

Regarding the survey data, there were multiple solutions attempted to solve the Neural Network Problem including parameter optimization, testing the train and test data, changing the test size, converting all the data to float. Despite having multiple classifiers with the problem of lopsided results, all the other classifiers were fixed with these alterations. However, the Neural Network could not give consistent results, switching between normal results and classifying everything into the one category.

Further to this, it was found that the data was difficult to model, given the generally low predictive ability. The random forest was found to be the most successful model, but most models having similarly good performances around 62%-64%.

Further research would include ensuring that the neural network become fully operational by testing it on different datasets to get a greater understanding of the issue that effected it in this project. Another potential area of research would include creating new models from scratch opposed to using pre-defined models from SkLearn, which felt like a step too far for this project.

V. References

[1] https://www.kaggle.com/datasets/dexmasa/uk-news-headlines/data

[2] https://www.kaggle.com/datasets/daliaresearch/trump-effect

[3] Danilo Najkov, "Detecting political bias in online articles using NLP and classification modles." medium.com. https://medium.com/@danilo.najkov/detecting-political-bias-in-online-articles-using-nlp-and-classification-models-c1a40ec3989b

[4] Minh Wu, "Political News Bias Detection[using Machine Learning," Earlham College, https://portfolios.cs.earlham.edu/wp-content/uploads/2018/12/senior-thesis-political.pdf

[5] RamaRohit Reddy, Suma Reddy Duggenpudi, Radhika Mamidi, "Detecting Political Bias in News Articles Using Headling Attention", International Institute of Information Technology, Hyderabad, https://aclanthology.org/W19-4809.pdf

[6] Janosch Delcker "Decoded: How Cambridge Analytica used AI — No, Google didn't call for a ban on face recognition — Restricting AI exports", PoliticoAI, https://www.politico.eu/newsletter/ai-decoded/politico-ai-decoded-how-cambridge-analytica-used-ai-no-google-didnt-call-for-a-ban-on-face-recognition-restricting-ai-exports/#:~:text=The%20company%20also%20used%20machine,company%20had%20gained%20access%20to.

Additional References for Code

https://melaniewalsh.github.io/Intro-Cultural-Analytics/05-Text-Analysis/03-TF-IDF-Scikit-Learn.html