News Article Classification

Antonin Musil[1], Oskar Koloszko[2], and Ben Jordan[3]

[1]Faculty of Information Technology, Czech Technical University Prague
[2]Faculty of Electronics Telecommunications and Information, Gdansk University of Technology
[3]Department of Statistics, University of Warwick

July 2, 2024

Introduction

This report delves into the Machine Learning models learnt this semester applied to news article classification. The primary objective was to investigate the efficacy of diverse Machine Learning algorithms in classifying news articles into different categories. Special emphasis was placed on Ensemble methods. As we navigate through this report, we will detail the preprocessing steps, outline the details of text classification methods and explore the selection of various machine learning models.

Data Preprocessing

Exploratory Data Analysis

We performed Exploratory Data Analysis ("EDA") on the full dataset of 148122 observations across the variables: *category* the true label of the article, *headline* the headline of the article, *author* the articles author (name, institution, name and description), *short_description* a short description of the article, *link* the URL of the website, *date* the date the article was published.

The dataset is missing data as can be seen in Figure 1: headlines are missing a negligible amount of values; 8% of short descriptions are missing; and 16% of authors are missing; all other features are complete. Furthermore, the amount of missing data is not entirely uniform across categories with some categories not having any missing data. Hence, it is unlikely the data is Missing Completely at Random. However, since there is only a small amount of missing data and for simplicity, we omitted all observations with missing values as this still left us with a sufficient amount of data.

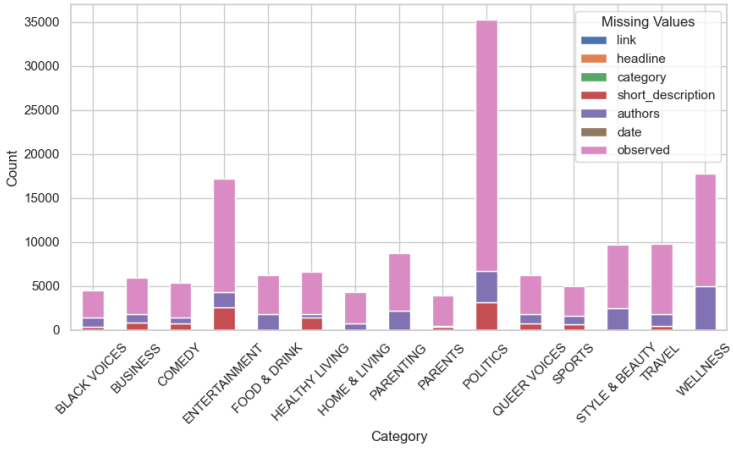

Figure 1: **Missing Data** by category

The response variable of interest, *category* assumes 16 different values from COMEDY to PARENTING. Politics is the largest class by far but still only assumes 26% of the observations. This means the majority classifier is very poor and there is a lot of scope for improvement on this.
We did notice that some classes have similar labels. Therefore, we decided to combine several classes: PARENTING into PARENTS; HEALTHY LIVING into WELLNESS; COMEDY into ENTERTAINMENT; BLACK VOICES and QUEER VOICES into MINORITY VOICES. This will reduce the complexity of our models and improve accuracy without compromising the utility of the model. In Figure 2, we can see the merged categories are slightly imbalanced but appear to be well-populated so the imbalance may not be an issue.

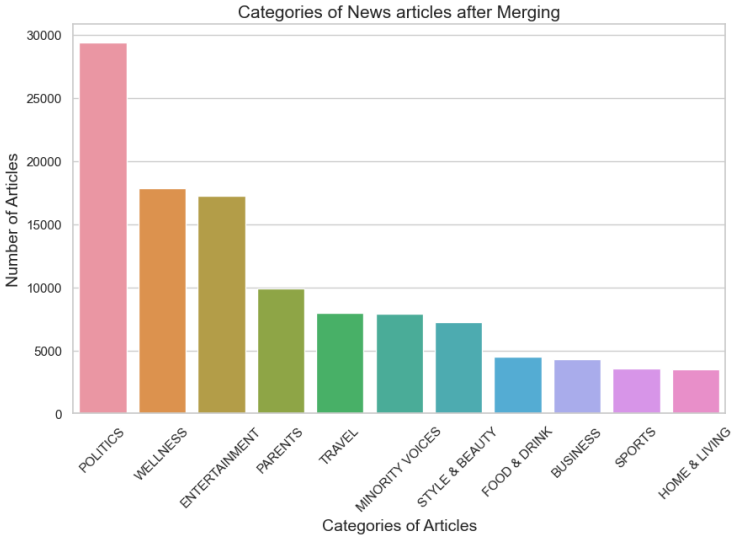

Figure 2: **Category frequency** after merging and omitting missing data.

We then examined our potential explanatory variables. *date* spans from 2012 to 2022, however, the majority of the articles come from 2012-2017. Furthermore, we did identify a slight relationship between *category* and *date* with an increase in the frequency of POLITICS articles in later years and a fall in WELLNESS articles for example. However, because this study focuses on classifying the documents using text features and not examining temporal trends, *date* was dropped.

We also dropped *link* as all articles are from the same website and any information in the URL about the article is covered in *headline*. The final three features *headline*, *short_description*, *authors* are the features we will use for the classification. No-

tably, we decided to keep *author* because as well as some authors being known for writing a specific genre more often, this feature included descriptions or names of companies/institutions which could offer more general information for classifying the document.

Finally, for computational reasons, we selected 10% of the remaining observations from each category. This still left us with 11334 observations.

Text Preprocessing

We then cleaned our remaining text features: tokenised the words, removed all punctuation and removed stop words. Then we leaned toward lemmatization over stemming. Because of the reasonably small size of our final dataset, we could decide on the former one, which even if less computationally efficient does not remove common suffixes, but ensures the output word is an existing normalized form of the word that can be found in the dictionary.

A quick check of a couple of examples showed we achieved the desired results.

Type of sentence	Sentence
Headline	I Quit My Job Every Year, And You Should Too
Preprocessed headline	quit job every year
Description	Every December 31st, I quit my job. The next day I decide if I want to take the job again for the New Year.
Preprocessed description	every december quit job next day decide want take job new year
Author	Aaron Hurst, ContributorCEO - Imperative & Author - The Purpose Economy
Preprocessed author	aaron hurst contributorceo imperative author purpose economy

Table 1: Examples of preprocessing

Training and Testing Splits

We are only using 10% of the original data however this is still a large amount of data so we opted for an 80%/20% stratified training-test split.

Details of Text Classification Models

Text Vectorisation

Before we could begin training and testing our models we needed to convert our text features into numerical vectors for our array of classification tools. We first merged all our text into one feature and then vectorised it using TF-IDF. We choose TF-IDF over word embeddings because, in a quite small dataset of long sentences, word embeddings could produce unnecessary noise and overfitting, as they represent words in a much more complex way. The size of our dataset made us want to use the entirety of our data, which is possible only by using TF-IDF.

In Figure 3 and Figure 4, we can see the word clouds produced from the training data, where the size of each word is proportional to the frequency of occurrence of a word in a given category.

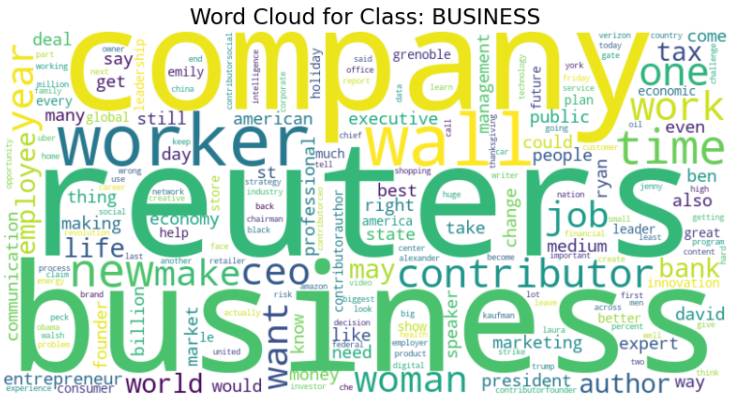

Figure 3: **A word cloud for business** produced from the vectorised training data.

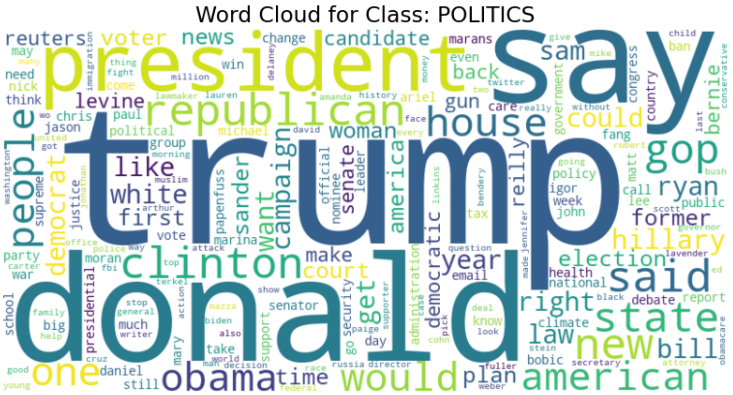

Figure 4: **A word cloud for politics** produced from the vectorised training data.

We can use the word clouds to check our preprocessing, there are not many stop words in the document and no punctuation or different forms of the same word. Furthermore, we can check the TF-IDF is performing as expected. Based on our common sense and prior knowledge the most prominent words agree with what we may expect. For example, based on the period when most of the POLITICS articles came from it is natural that 'donald' and 'trump' are prominent words. Similarly, for BUSINESS the words agree with our intuition. Business and politics are similar genres and we can see that there is overlap between some of the less prominent words.

Exploring Model Performances

The breadth of Machine Learning models is both a blessing and a curse. To gauge which models were most suited to the data and the task we experimented. We trained a mix of models including Ensemble models, Neural Networks and Logistic Regression all using the default parameters to get a feel for which models we should take an in-depth look at. In Figure 5, we can see the average Cross Validation ("CV") accuracy and the average training set accuracy from a 5-fold CV. We can see several models are prone to overfitting such as Decision Trees with very high training CV accuracy and much lower test CV accuracy. Other models are not well-suited to the data at all such as Naive Bayes. As well as accuracy we examined various other metrics such as F1 score, Recall and Precision for each model. These metrics

along with computational reasons and our desire to examine Ensemble methods further led us to choose SVM and XGBoosting for a more in-depth fitting and fine-tuning.

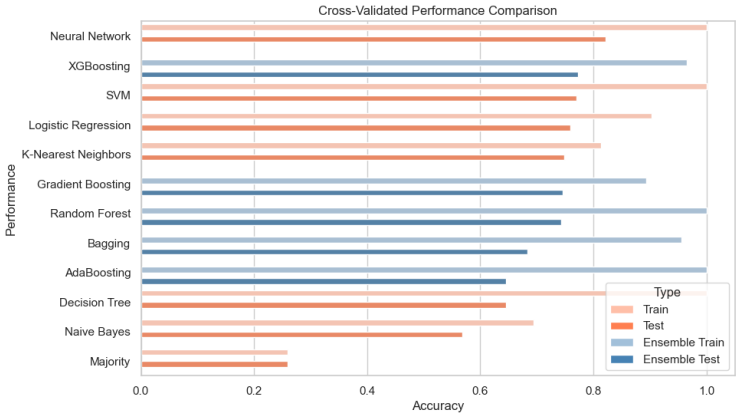

Figure 5: **CV Training and Test Accuracy** of a selection of classification models.

Hyper-parameter Tuning

We used CV to provide a more robust measure of our model performance while choosing our hyper-parameters. We explored the use of Grid-Search and Genetic Algorithms ("GA") to find the best-performing parameters. For Grid-Search we specified a sparse matrix of hyper-parameter values and simply tried each of them. We set up the GA as follows: the gene space was an array corresponding to potential values of the hyper-parameters, and the fitness function (to be maximised) evaluated the CV accuracy of the model with given hyper-parameters which were the solutions of the generations.

SVM

SVM has a few key parameters. The most important one is the kernel. The literature states that the linear kernel is usually most appropriate for text classification tasks. However, Radial Basis Function ("rbf") and Polynomial kernels ("poly") may still perform well. The C parameter is another important measure that can greatly affect the results of the model. In the case of choosing the Radial Basis Function or Polynomial kernels, we should take into account also the Gamma parameter which controls the shape of the decision boundary. However, after performing a grid search CV for the three kernels, we observed that the linear kernel is the best for this task as seen in Figure 6.

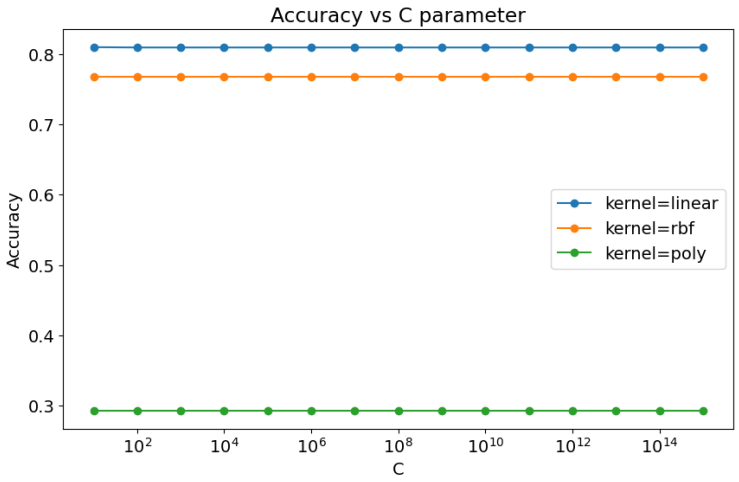

Figure 6: **CV Accuracy over different kernels** for SVM

Because we chose the Linear kernel the only parameter we needed to tune was C. We used the Genetic Algorithm. However, since the C parameter provides the highest possible accuracy on a wide range of values, the Genetic Algorithm got the highest one on the first generation and then was only gliding on the plateau of the right answers. Therefore we chose the minimal C value for the maximum accuracy. The value of C was 10 and we hit the accuracy of 0.815.

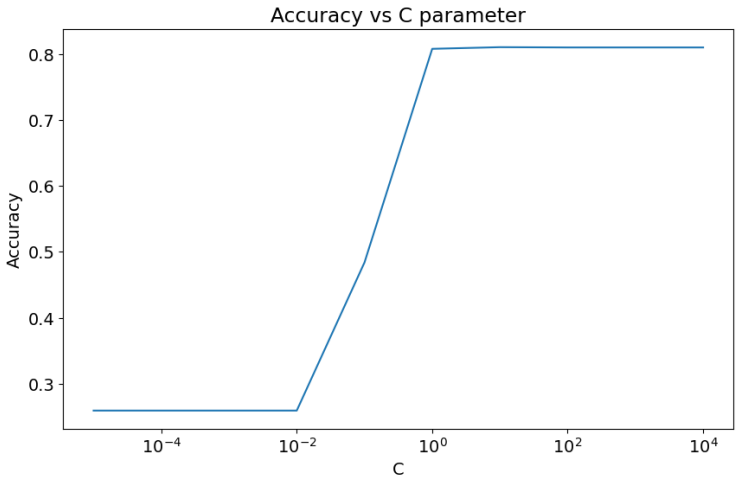

Figure 7: **CV Accuracy over different C values** for SVM

XGBoost

XGBoost has several hyperparameters to assess. We examined the Tree booster and the simpler Linear booster. The default booster is a Tree and we also examined the Linear booster which is simpler and computationally less expensive. The hyper-parameters relevant to these two boosters differ so we did the hyper-parameter tuning separately.

For the tree booster variant, we experimented with the following parameters: *learning_rate*, *max_depth*, *min_child_weight*, *subsample* and *column_sample_by_tree*. For the linear booster variant, we experimented with the following parameters: *learning_rate*, *n_estimators*, *alpha*, *lambda*. Note that we only explored *n_estimators* for the linear booster due to it being far less computationally expensive. The grid for grid-search was filled with a few different levels of each of these parameters and CV on accuracy was used to assess the various combinations. We then tried the GA to see if we could find better hyper-parameters.

In Figure 8, we can see the change in the accuracy of the best model at each generation. After 12 generations the GA finds a local solution with 0.778.

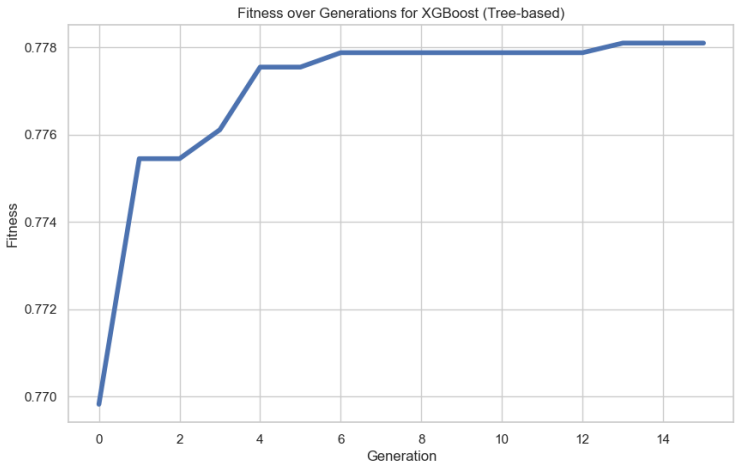

Figure 8: **CV Accuracy over generations** for Tree XGBoost

Similarly, in Figure 9, we can see the change in the accuracy of the best linear XGBoost model at each generation. This time the accuracy hits 0.82 after the 3rd generation.

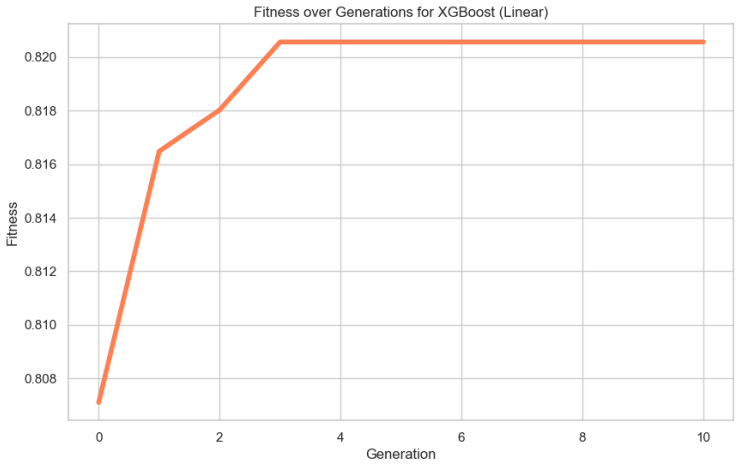

Figure 9: **CV Accuracy over generations** for linear XGBoost

Examination of Final models

SVM

The final SVM model is a linear kernel SVM model, with the hyper-parameters shown in the code snippet below.

Listing 1: Final SVM model

```
SVM_best = SVC(C=10.0, kernel='linear')
```

XGBoost

The final XGBoost model is a linear booster XGBoost model, with the hyper-parameters shown in the code snippet below. Instead of the more well-known Tree-booster variant, we have the Linear booster variant so we are utilising an ensemble of generalized linear models with elastic net to get our prediction probability. The linear booster may be outperforming the Tree booster as linear tends to be able to handle high dimensional sparse data which is what we have as a consequence of using TF-IDF. Furthermore, the Elastic Net regularisation may also help our model avoid overfitting while a simpler model assuming linear relationships can adequately capture the pattern.

Listing 2: Final XGBoost model

```
xgb_ga_best = XGBClassifier(learning_rate=5.75439937e
    -03, n_estimators=320, alpha=1.31825674e-05,
    reg_lambda=1.00000000e-05, booster='gblinear')
```

Results

Performance

Since the XGBoost model outperformed the SVM model after hyper-parameter tuning we will present the results regarding this model on the hold-out test set. Firstly, the model achieved the following aggregated (weighted-mean) scores: accuracy 0.83; F1-Score 0.82; Precision 0.84, Recall 0.83. The model performs well across multiple performance metrics and demonstrates a balanced trade-off between precision and recall. The most important metric for us is arguably accuracy as we want to maximise how often the model correctly classifies a news article. With an accuracy of 0.83, the XGBoost model has a strong ability to make correct predictions, meaning it is quite reliable for predicting news article categories.

In Table 2, we have the performance broken down by category. It is clear to see that although the aggregated performance was balanced well this is not the case when broken down by category. We can see that BUSINESS articles are one of the worst classified articles. This category has one of the smaller supports in the test set and correspondingly in the train set. Moreover, we can see that the categories with smaller supports tend to be classified worse. Nonetheless BUSINESS is noticeably the worst category for being classified correctly. It has a low Recall meaning that it is often misclassified as something else but a reasonable precision meaning it is not often the model is incorrect when it predicts it is a BUSINESS article.

Category	Precision	Recall	F1	Sup
BUSINESS	0.82	0.43	0.56	87
ENTERTAINMENT	0.82	0.84	0.83	345
FOOD & DRINK	0.93	0.73	0.82	90
HOME & LIVING	0.96	0.70	0.81	70
MINORITY VOICES	0.88	0.64	0.74	159
PARENTS	0.79	0.78	0.79	198
POLITICS	0.84	0.95	0.89	587
SPORTS	0.98	0.58	0.73	71
STYLE & BEAUTY	0.93	0.83	0.88	144
TRAVEL	0.85	0.83	0.84	159
WELLNESS	0.74	0.93	0.83	357

Table 2: Classification Metrics

In Figure 10, we can see the number of correct and incorrect classifications for each category. Standing out is again BUSINESS which has over half miss-classifications and compared to HOME & LIVING with a similar support this is low. We can investigate further what exactly is happening with these misclassifications. Another obvious trend is that the larger classes have a higher accuracy.

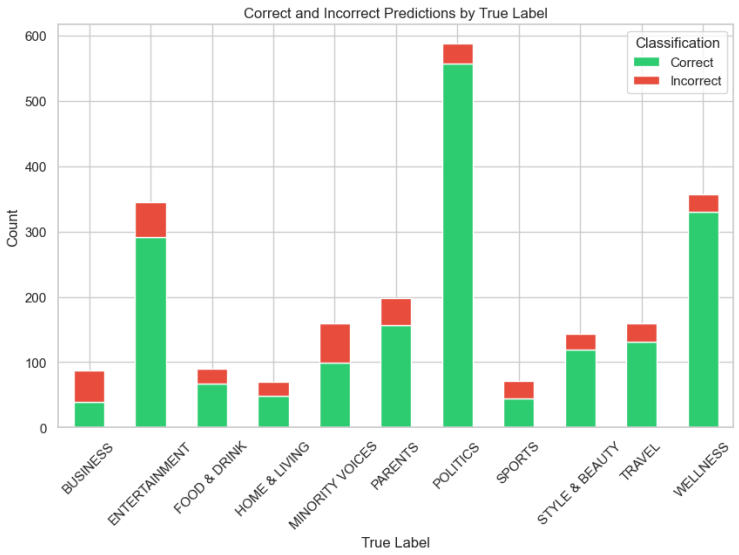

Figure 10: **Prediction performance** by true category.

Investigation of performance

Figure 11, paints a similar picture to the above showing us that the performance is not consistent across categories and that BUSINESS is the worst classified category with only 43% accuracy on the test set and it is often misclassified as POLITICS. Intuitively, this makes sense however in the text vectorisation we

showed that the word clouds for these two classes were reasonably distinct but it seems the model is still very poor at distinguishing BUSINESS from POLITICS. It seems likely that the imbalanced classes are a factor here as POLITICS is a much larger class. Lastly, the confusion matrix does validate our model in the sense that the common misclassifications are from intuitively similar categories with the odd exception.

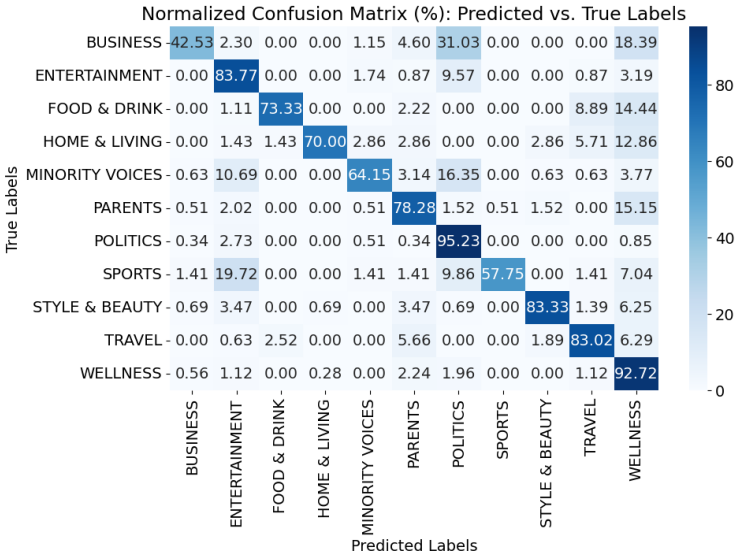

Figure 11: **Confusion Matrix** for linear XGBoost

Best performing model

As mentioned in Examination of Final models, the linear Booster XGBoost was the best-performing model, effectively an enhanced logistic regression model. The model takes the hyper-parameters shown in 2. Firstly, the *learning_rate* is relatively small at approx. 0.005 but the *n_estimators*, the number of boosting rounds, is larger likely to compensate for the smaller learning rate. This combination allows for better convergence and fitting a more complex model. The larger number of boosting rounds does make us wary of overfitting but as we saw in the results the test accuracy is higher than the training accuracy. The *alpha* and *reg_lamba* are L1 and L2 penalisation coefficients. They are both very small indicating that regularisation was not needed very much for our model. From the initial exploration, it seems that using a linear booster was enough to eradicate any overfitting seen with this model.

However, although the XGBoost had good performance all around it could not beat the default Neural Network in terms of just accuracy. Even though this suffered from overfitting the test performance was still marginally better however it was far more computationally expensive and more complex.

Discussion

Conclusion and evaluation of project aims

In this report, we successfully explored many different methods for the classification of news articles and implemented critical thinking and several techniques to process our text data beforehand. We ended up with a selection of well-performing models tuned through the use of Genetic Algorithms.

Surprises and challenges

The first challenge came from the sheer amount of data provided. This meant many of the methods took far too long to train especially if we wanted to implement CV and Genetic Algorithms to tune them. We, therefore used a small subset. To be able to explore more models and tune them more effectively. We still achieved good results but this would have reduced our accuracy. A quick check shows that training our final model on all the data and testing it gives 0.87 accuracy. So there is definite scope for improvement here.

A surprise for us was that the Tree booster was outperformed by the Linear booster for the XGBoost model. Moreover, we did expect some categories to be easier to identify than others but the gaps in performance across categories was another surprise and warrants further investigation.

The model confused articles of similar categories for the most part except for potentially BUSINESS and WELLNESS. These seem like quite different categories of articles and we expected them not to be confused as much as they are. One explanation could be that these articles are all from the *HuffPost* an American outlet, so there may be a stronger overlap between these articles due to the American working culture.

Limitations and scope for improvement

Due to computational reasons, we only used a 10^{th} of the available date. Training our model on more data and tuning the hyperparameters on the full data would likely improve the results further. As we could see the scores with lower support did not perform very well and using more data may negate any class imbalance issues e.g. BUSINESS vs POLITICS.

As well as our use of only 10% of the data another limitation is the data set itself. All articles come from one source HuffPost an American Progressive news outlet. This may mean our model is biased towards this kind of article. Further, the data comes from a very short time interval and for example, with POLITICS it seems to have put a great influence on 'donald' and 'trump'. This is fine for our data and gives good performance for POLITICS but as times change this may mean our model won't adapt well.

Bayesian hyper-parameter tuning has now become the state-of-the-art method for finding the optimal hyper-parameters. Using this may be able to squeeze out even more performance out of our models.

Despite, all the hyper-parameter tuning we did we still could not match the accuracy of the default Neural Network. Therefore, arguably the next step would be to examine the exact differences in performances here and if necessary move to using a Neural Network instead.

Furthermore, we decided to omit all observations with missing data, however, we may have biased our results. Firstly, as noted the missingness is not uniform across categories so we will have to change the class balances slightly. Moreover, a more thorough investigation of the missing data mechanism may show that the missing data is problematic and should be addressed. Either, since no observations contained missing values across all three features we could model with the data and see how sensitive the results are or we could investigate possibly imputing the missing text data and seeing how this affects the results.