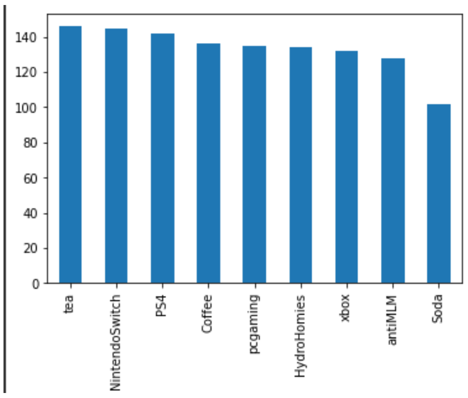
Text as Data Coursework

Name: Adam Fairlie
Student Number: 2461352F

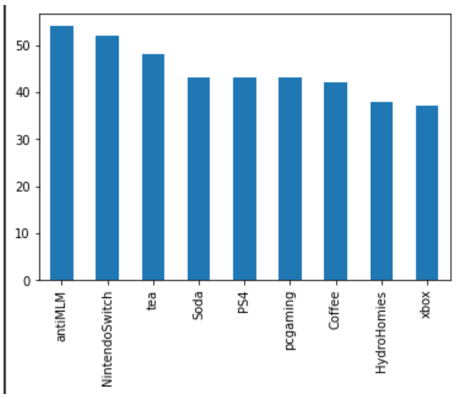
Question 1a (Distribution):

This is the distribution of labels for the test, validation and training sets:

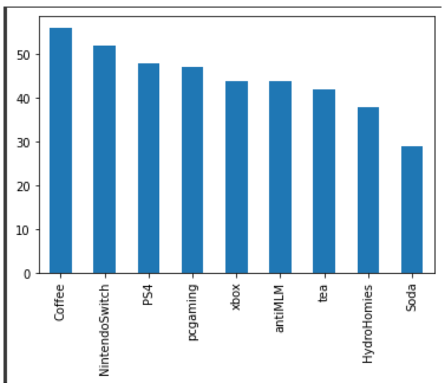
Training



Validation



Test



The maximum percentage difference in distribution from the most represented to least represented subreddit in the testing set is 3.67% in the testing set (between tea and Soda), 4.25% in the validation set (between antiMLM and xbox) and 6.75% in the test data (between Coffee and Soda). This means that the most popular subreddit has 3.67%, 4.25% and 6.75% more distribution respectively than the least popular subreddit.

The distribution in all cases is not ideal, having a slight slope in the bar chart rather than a roughly flat distribution. This may lead to some subreddits such as Soda and HydroHomies being underrepresented in the data and possibly harder to classify. In addition, the relative distributions are not the same between the three splits, with more representation for subreddits like tea and less for subreddits like AntiMLM. However, there are no massive outliers, as each subreddit has over 100 training cases and at least 30 validation and testing examples.

Question 1b (Training Models):

Classifier		**Precision**	**Recall**	**F1-Score**
Stratified	**Accuracy**			0.102
	Macro	0.103	0.103	0.102
	Weighted	0.105	0.102	0.103
Most Frequent	**Accuracy**			0.105
	Macro	0.012	0.111	0.021
	Weighted	0.011	0.105	0.020
One-Hot LR	**Accuracy**			0.770
	Macro	0.776	0.775	0.773
	Weighted	0.774	0.770	0.769
TF-IDF LR	**Accuracy**			0.792
	Macro	0.807	0.791	0.795
	Weighted	0.801	0.792	0.793
SVC	**Accuracy**			0.690
	Macro	0.755	0.693	0.701
	Weighted	0.751	0.690	0.696

The baselines provided by the dummy classifiers are 10.2% accuracy in the stratified strategy and a slightly higher 10.5% using the most frequent subreddit. This suggests that the uneven distribution of the dataset is not enough for the baseline models to be high accuracy, and so less consideration for how to balance the dataset is needed in this case. It is clear from these results that the SVC model is less appropriate to this task than logistic regression. This may be to the simplicity of the model in comparison, which may avoid overfitting more than the SVC model.
Of the logistic regression models, clearly TF-IDF vectorization provided better results. This is likely because more information about the frequency of terms in relation to the corpus, as opposed to a simple binary one-hot approach to encoding, will provide more powerful predictions in the classifier.

Question 1c (New Model):
I chose to implement an SVC classifier with TF-IDF vectorization. After seeing the improvement in accuracy when using TF-IDF encoding instead of one-hot encoding in the logistic regression model (from 0.77 to 0.792 accuracy) I suspected that the accuracy would largely improve in the SVC model. This was indeed the case, as is reflected in the results:

SVC (TF-IDF)	**Accuracy**			0.777
	Macro	0.807	0.773	0.784
	Weighted	0.797	0.777	0.782

The accuracy largely improves (from 0.69 to 0.777), suggesting that similarly the TF-IDF vectorization is more effective than the one-hot vectorization in this classification task. This change of vectorization strategy makes the new model the second most effective for this task, but it does not create enough improvement to justify the change from a logistic regression model, as the best performance still comes from the logistic regression model with TF-IDF vectorization.

Question 2a (Parameter Tuning):

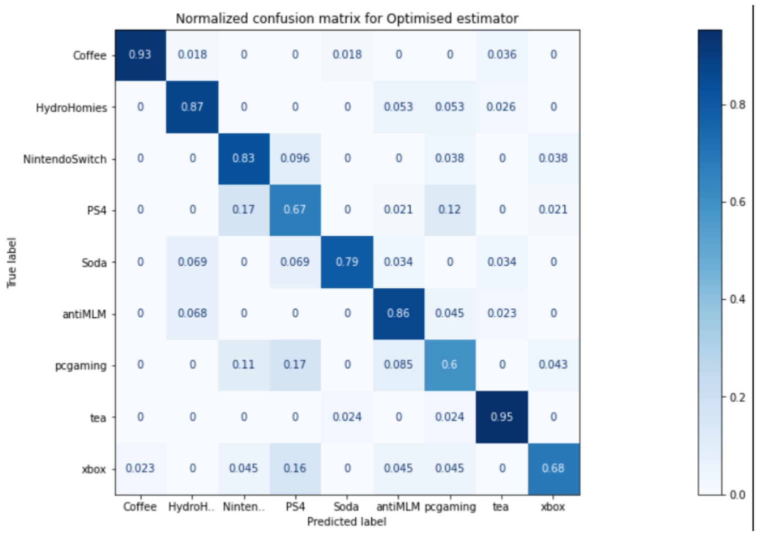
Parameter	**Value**		**Precision**	**Recall**	**F1-Score**
Base model		**Accuracy**			0.792
		Macro	0.807	0.791	0.795
		Weighted	0.801	0.792	0.793
C	**1.0**	**Accuracy**			0.797
		Macro	0.808	0.797	0.800
		Weighted	0.803	0.797	0.798
Sublinear_tf	**True**	**Accuracy**			0.797
		Macro	0.808	0.797	0.800
		Weighted	0.803	0.797	0.798
Max_features	**20000**	**Accuracy**			0.797
		Macro	0.808	0.797	0.800
		Weighted	0.803	0.797	0.798
tol	**0.01**	**Accuracy**			0.797
		Macro	0.808	0.797	0.800
		Weighted	0.803	0.797	**0.798**

I chose to search for the best tolerance of the logistic regression classifier (the precision level at which the model stops), as I guessed that it may influence the accuracy, by adjusting the amount the model overfits or underfits to the training set to have the best result on the validation set.

I used the validation set for cross-fold validation, by combining the data frames and defining a split of test to validation to be used for the grid search, and using the test set to check the overall performance. The grid search produced a very slight improvement to the model, although not statistically significant. In general, the parameters seemed to make little difference to the weighted F1 score of the model, the only parameter which produced a difference was the first regularisation C-value, which provided a small difference which is statistically insignificant. I found when testing the grid search functions that the outputs were very inconsistent. The optimal parameters which were returned by the grid search would vary wildly, and the results of the exact same models could fluctuate largely when redefined and refitted multiple times.

This may suggest that these parameters do not have enough of an effect on the overall inner logic of the model to make a meaningful enough difference, particularly with a sample size this small (only 400 test and validation cases), to offset the slight random variation of the model's fittings.

Question 2b (Manual error checking):
Below is a confusion matrix for the newly optimised estimator:



From analysing this data, it would seem that most of the mistakes are between the tech-related subreddits (nintendoswitch, ps4, pcgaming, xbox). This is likely because these subreddits will be far more similar to each other than the rest of the corpus, using more tech-related words which will be shared across posts on these subreddits (e.g. device specs, cross-platform games). Similarly, coffee and tea are commonly mistaken for each other, likely because they have similar words, possibly relating to taste, ingredients, colour or action (drink, sip etc.)/ A larger sample size would likely be useful in clarifying the content of these subreddits by building the vocabulary to make more nuanced distinctions between the common features of different consoles or hot beverages.

Question 3a (New features):
The two features I chose to implement were:

-Include the title of the post in the data to be used by the model
-Use a random forest classifier rather than a logistic regression

I chose to include the title of the post in the model data because it is likely to contain information relevant to the success of the model. Particularly, titles will typically be more concise than the text body and contain lots of keywords as the user will want to get their point across in the title as it will be seen by the most users, so it should be specific and to the point. I hypothesise that this means we will have more accurate TF-IDF values within the title which will be useful in making a classification. In addition, it is generally likely that having more data per case will lead to a more “informed” decision.

I chose to use a random forest classifier rather than a logistic regression classifier as I am under the impression that these are more commonly used in classification tasks. The logistic regression model is likely too simple for this task, as the data between subreddit text, particularly those of similar subreddits, may not easily be separable through a line. This means a more complicated model is required to provide the nuance to improve accuracy. It may produce better results to split the dataset into decisions about the post content (now title and body), to allow for finer clustering to identify the differences between subreddits with similar terms.

Question 3b (Testing of new model):

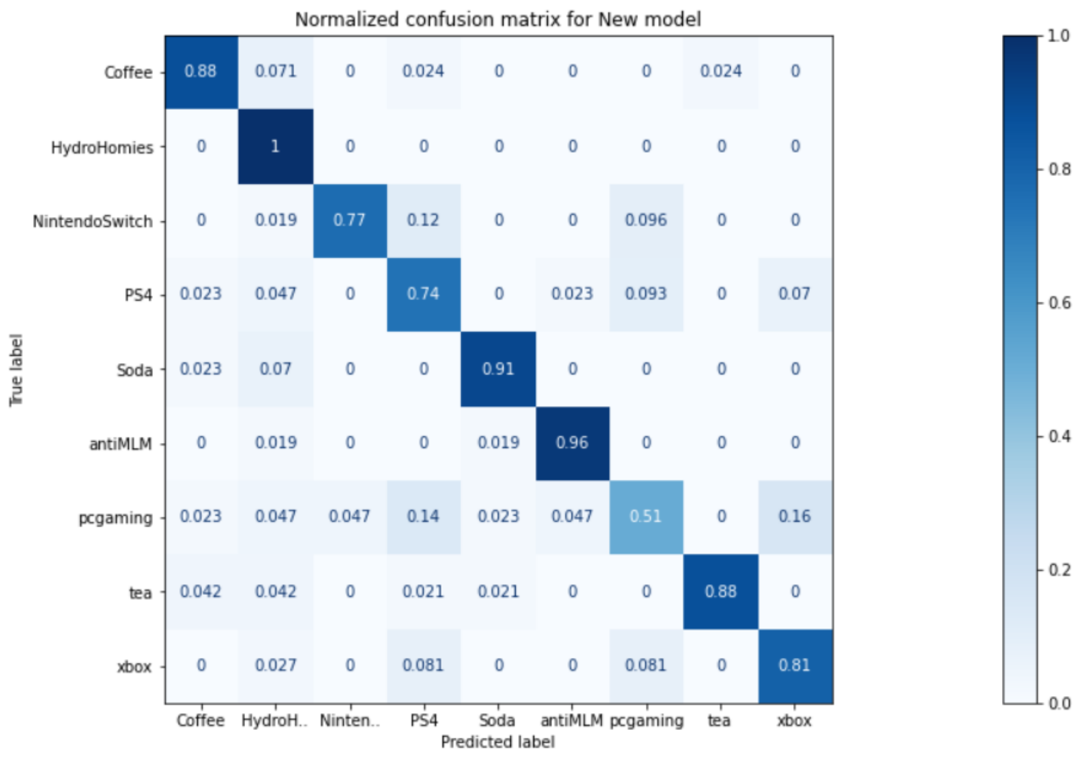
The new model was trained on the training data and tested on the testing data, here it is compared to the former best model (the optimised model from Q2):

Model			**Precision**	**Recall**	**F1-Score**
Optimised		**Accuracy**			0.797
		Macro	0.808	0.797	0.800
		Weighted	0.803	0.797	**0.798**
New		**Accuracy**			0.843
		Macro	0.841	0.841	0.839
		Weighted	0.848	0.843	**0.843**

As seen in the results, the models weighted f1 score has improved by 0.45, and accuracy has gone from 79.7% to 84.3%, a fairly large improvement.

Question 3c (Evaluation of new model):

From the results in question 3b, the new features I proposed for the model have clearly made a significant improvement. I believe my hypothesis from Q3a was correct: that the inclusion of the titles is likely the most important component to the improved performance. The post body may be missing some of the important context, relying on any user who has read the body to have read the title, and so it is natural that it is included. Particularly with a small sample size, the added context for each post can make a huge difference in the models ability to "understand" what each subreddits posts are likely about. Using a confusion matrix, we can analyse where the improvements have been made:



True label \ Predicted label	Coffee	HydroH..	Ninten..	PS4	Soda	antiMLM	pcgaming	tea	xbox
Coffee	0.88	0.071	0	0.024	0	0	0	0.024	0
HydroHomies	0	1	0	0	0	0	0	0	0
NintendoSwitch	0	0.019	0.77	0.12	0	0	0.096	0	0
PS4	0.023	0.047	0	0.74	0	0.023	0.093	0	0.07
Soda	0.023	0.07	0	0	0.91	0	0	0	0
antiMLM	0	0.019	0	0	0.019	0.96	0	0	0
pcgaming	0.023	0.047	0.047	0.14	0.023	0.047	0.51	0	0.16
tea	0.042	0.042	0	0.021	0.021	0	0	0.88	0
xbox	0	0.027	0	0.081	0	0	0.081	0	0.81

The new model seems to be far more able to distinguish the different tech models, for example eliminating all cases where NintendoSwitch posts are misclassified as xbox. This may be due to the decision tree model being a more appropriate fit to cluster the posts, or it may be due to the title providing the needed context to separate them, being more specific to the console than the post body.

The model seems to be refined in all areas, for example it can now almost perfectly distinguish HydroHomies posts in the corpus, only making a single mistake. I believe that this could be further improved through using a grid search to tune the parameters for the new random forest regression model, most importantly the number of decision trees, as finding the correct parameters for this case will likely have a large influence on the end results of the model (there may be a balance between a model which can distinguish the more closely related subreddits such as tea and coffee, but avoids overfitting to the exact vocabulary used in the posts and performing poorly with new words and synonyms / different phrasings).

Another improvement which likely can be made is a better form of vectorisation. The TF-IDF vectorizer is the most efficient of the two I have tried, but recent developments in the area would suggest that a better performance could be gained using more modern text representations, such as word2vec or even BERT context vectors, which have been proven very effective when used in similar classification tasks. These representations have been trained on a far larger corpus, and so each word vector representation should be less likely to overfit to the corpus (if the small corpus used here is biased to certain words, the TF-IDF scores may not reflect a true score of how common words are in posts).