Name: Karlis Siders
Student Number: 2467273
Word Count: 1836

Q1: Comparing Classifiers (10 marks)

Q1a:

The labels are split 60:20:20 for train:validation:test data, which is a split used very commonly in practice.

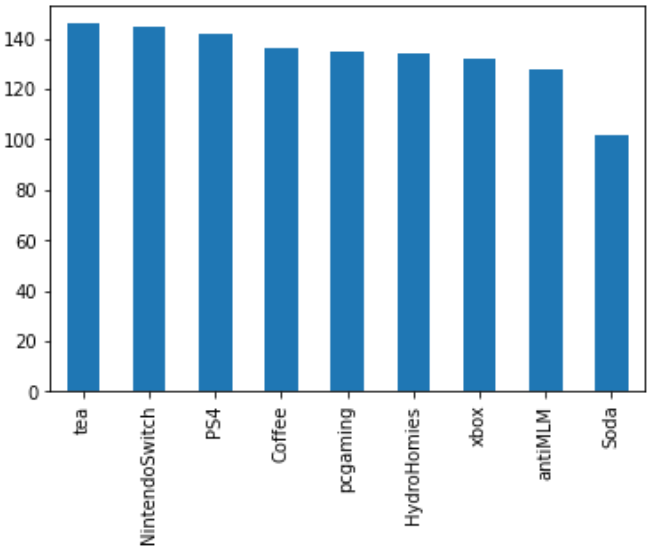

Figure 1. Posts grouped by subreddit in training data

However, the distribution in each set is different and non-uniform (as seen in Figure 1 for the training set), which means that the data might be biased towards the more frequent labels in the training data, e.g., tea, and less towards the much less frequent, e.g., Soda.

Q1b:

Classifier	**Accuracy**		**Precision (weighted avg)**		**Recall (weighted avg)**		**F1 (weighted avg)**	
	Validation	Test	Val.	Test	Val.	Test	Val.	Test
Dummy (most frequent)	0.120	0.105	0.014	0.011	0.120	0.105	0.026	0.020
Dummy (stratified)	0.105	0.085	0.097	0.088	0.105	0.085	0.100	0.086

Logistic Regression (one-hot)	0.762	0.765	0.766	0.768	0.762	0.765	0.763	0.764
Logistic Regression (TF-IDF)	**0.733**	**0.775**	**0.732**	**0.783**	**0.733**	**0.783**	**0.731**	**0.776**
SVC (one-hot)	0.708	0.690	0.771	0.751	0.708	0.690	0.719	0.696

Figure 2: Classifiers and their metrics.

Naturally, both Dummy classifiers performed poorly (less than 10% in weighted-average F1 score) in modelling the data as they both do not consider the input feature. The one with the 'most frequent' strategy performed marginally worse most probably because the training data is biased towards values that are not biased to in the test data, while the 'stratified' strategist guesses randomly, which, even though it performs very poorly in multi-class classification as the number of classes is relatively high, guesses correctly enough times to outperform the other dummy. If these classifiers act as baselines for the performance of the other ones, one can see that every one of them performs exceptionally well.

The C-Support Vector Classifier (SVC) with one-hot encoding works several times better than the dummy classifiers because, most importantly, it actually takes into account the input data. Because the data is not very large, it does better than it would with a larger dataset. However, it performs worse than other classifiers partly because it uses one-hot encoding, which loses data about the frequency of tokens used in the posts, which could be an important factor to determine the theme/focus of a post. Secondly, SVC should be used with finetuned parameters, especially C and gamma, because those change the performance drastically, which is why the default ones are subpar.

Finally, Logistic Regression is the best-performing classifier out of all of the above. One of the reasons for the better performance might be that the dataset is linearly separable[1], i.e., there can be found thresholds which describe when a given post is more likely to be from one subreddit rather than another. Additionally, it deals very well with dependencies between features. While the classifier with one-hot encoded vectorisation performs worse because of the loss of data when one-hot encoding, the one with TF-IDF vectorisation performs better (and best of all given classifiers) because of the added context to how frequently tokens are being used in each post.

[1] Raj, Ashwin. Perfect Recipe for Classification Using Logistic Regression. https://towardsdatascience.com/the-perfect-recipe-for-classification-using-logistic-regression-f8648e267592. Last accessed: 12 March, 2022.

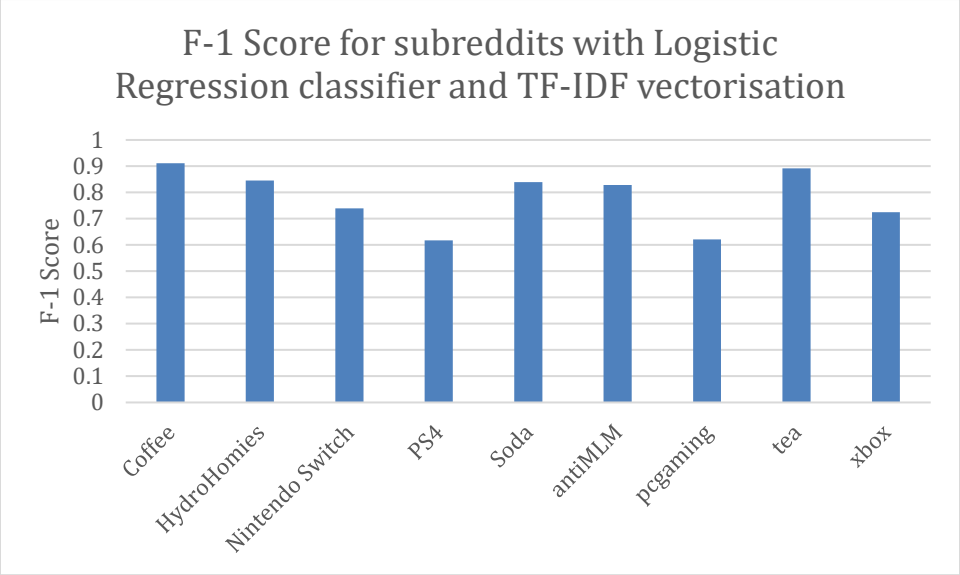

Figure 3: Logistic Regression (TF-IDF) results in more detail by subreddit

Q1c:

Classifier	Accuracy		Precision (weighted avg)		Recall (weighted avg)		F1 (weighted avg)	
	Val.	Test	Val.	Test	Val.	Test	Val.	Test
Linear SGD	0.795	0.782	0.795	0.783	0.795	0.782	0.794	**0.781**

Figure 4: SGDClassifier(loss='log', alpha=1e-3, max_iter=5)

Because the SVC performs worse when using large sets of data and the Reddit data seems big enough, I decided to use a classifier more appropriate to larger datasets, the Linear SGD Classifier, which is supported by the sklearn documentation[2]. This classifier performed 0.5% better than the best classifier in Q1b based on its weighted-average F1 score. The 'log' loss function was used because Logistic Regression was the best-performing classifier of the previous ones, which I wanted to improve upon with this classifier and added parameters. Additionally, the alpha value was chosen to be 10 times higher than the default value to make stronger regularisation as I imaged the variation in the text of the body to be very high. Finally, the max_iter value was lowered from the default 1000 to just 100 to try to prevent overfitting. Having all of these parameters with the chosen classifier improved performance, but only by half a percent, which is why more finetuning is necessary.

Q2: Tuning and Error Analysis (10 marks)

Q2a:

[2] sklearn.svm.SVC. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC. Last accessed: 11 March, 2022.

Classifier	Parameters				Metrics			
	Sublinear TF	Max features	C value	Solver	Accuracy	Precision (weighted avg)	Recall (weighted avg)	F1 (weighted avg)
Logistic Regression (TF-IDF)	True	None	1 000 000	saga	0.805	0.806	0.805	**0.804**

Figure 5: Logistic Regression (TF-IDF) with fine-tuned parameters

As is evident from Figure 5, the fine-tuned model performs best with sublinear TF set to True, max features not being implemented (None), C value being 1 million, and the additional change of the solver to ‘saga’ improves the performance even more, which is approximately 3% better than the baseline TF-IDF model.

The model performs better with sublinear scaling on because Reddit posts containing a word more times than another word does not necessarily mean that it is that many times more important. Specifically, important key words, like console-exclusive games or specific beverages, might appear less frequently because mentioning them only once acts as an implicit focus of the entire post and needs not be mentioned again.

By not setting a maximum on features, the model is able to use as many as it can, which prevents underfitting and ensures that the aforementioned important tokens are kept in the vocabulary.

By setting the C value the highest in the given range, a balance is reached with the other parameters in regularisation of the data. The danger of overfitting by not setting a maximum on features and regularisation already being enforced by the LR classifier in SKLearn is balanced by the nature of a very high C value being prone to underfitting.

Finally, choosing the solver to be ‘saga’ achieves a much faster convergence rate because the Reddit dataset is quite large, which is why the default ‘lbfgs’ solver might not be the most appropriate and had to be changed. Additionally, not wanting to delve deep into the mathematics of the solvers and penalty functions, a more modern solver seemed more appropriate because of the advancements of NLP nowadays.

Q2b:

The confusion matrix for the predicted test labels is this:

		Actual Subreddits								
		Coffee	HydroHomies	Nintendo Switch	PS4	Soda	antiMLM	pcgaming	tea	xbox
Predicted subreddits	Coffee	54	1	0	0	1	0	0	0	0
	HydroHomies	0	35	0	1	0	1	1	0	0
	Nintendo Switch	1	0	41	4	0	1	3	0	2
	PS4	0	0	7	32	0	0	6	0	3
	Soda	1	1	0	1	25	0	0	1	0
	antiMLM	1	3	0	1	0	36	1	1	1
	pcgaming	0	1	3	9	0	3	28	1	2
	tea	1	0	0	0	1	0	1	39	0
	xbox	0	0	0	6	1	3	2	0	32

As is immediately clear from the matrix, the most confused subreddits seem to be ones with a similar theme or hobby, such as 9 confused posts from the ‘PS4’ subreddit as ‘pcgaming’ posts and 6 posts vice versa, both of which are about video games, just on different devices. Similarly, 7 ‘NintendoSwitch’ posts are confused as ‘PS4’ posts and 4 vice versa, both being not just about gaming, but also gaming on a console. Finally, ‘xbox’ and ‘PS4’ posts seem to be confused (6 and 4) for the same reason, both being gaming consoles. Surprisingly enough, there does not seem to be a bias towards the ‘tea’ subreddit even though the training data is biased towards it.

Naturally, these errors happen when a post’s body is about a general topic that the two confused subreddits share, like gaming or beverages, instead of delving deeper into the specific topic or mentioning key words of the subreddit. For example, the following post about my favourite video game:

Daniel Bloodworth(he [sic] *reviewed the game for GameTrailers.com) did an awesome Q&A after about 100hrs of gameplay, there's a lot of interesting stuff if you are interested into The Witcher 3 or just can't wait to finally play it yourself.*

http://www.twitch.tv/gametrailers/v/4972732

Enjoy!

is posted to the ‘PS4’ subreddit, even though it either talks about the game in general or is implicitly about the release of the game for the PlayStation 4 console. Either way, even an average gamer would not be able to categorise this as a ‘PS4’ or a ‘pcgaming’ post just because of the generality of it (while even people having watched the Q&A would know it was done on an Xbox console even if it was reviewed on a PS4 by the same reviewers). Thus, if even a person cannot distinguish a subreddit for the post by the text while having an abundance of context for gaming and subreddits in general, any learning model having the body of the post as its only feature would not be able to perform well with these kinds of posts.

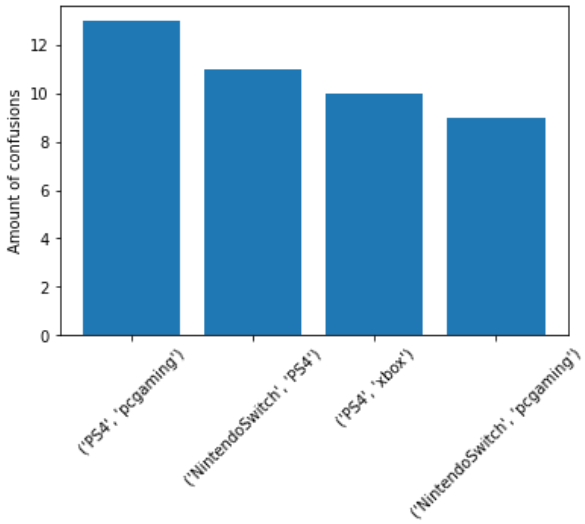

Figure 6: Top 4 most confused pairs of subreddits

As can be seen in Figure 6, the most confused subreddits are these very similar ones.

In summary, subreddits with similar themes/topics and an added human error of posting less relevant content or not mentioning the reason for/relevance of their posts are the main causes of confusion in this specific case of text classification by the body of the post.

Q3: Feature Engineering

Q3a:

The first of two features will be chosen as the title of the post, present in the data, as it is an additional source of information and possible key to classifying a post. A hypothesis is that sometimes authors of posts choose to specify key features and topic of the post in the title and only elaborate on the title in the body of the post, in which case using only the body of the post might cause loss of important data to help the algorithm distinguish between the subreddits.

The second feature will be chosen as the author of the post as it might be a helping feature to differentiate between similar posts. The hypothesis is that authors of posts will not post in separate but similar subreddits because they might prefer one over the other. They might post in very different subreddits, in which case the title and the body of the post will be the most important features to distinguish them, but similar posts in two very similar subreddits might have different authors who wrote them just because they play on different consoles (PS4/Xbox/PC) or prefer a different beverage (soda/water/tea). The assumption is that the average person would be passionate enough only about one specific topic in a general category (gaming/beverages) to post in its respective subreddit.

Q3b:

Classifier	Features (weight)			Metrics			
				Accuracy	Precision (weighted avg)	Recall (weighted avg)	F1 (weighted avg)
Logistic Regression (TF-IDF)	Body (30%)	Title (30%)	Author (40%)	0.835	0.839	0.835	**0.834**

Figure 7: Fine-tuned Logistic Regression (TF-IDF) Classifier with 2 additional weighted features

As demonstrated in Figure 7, the classifier from Q2 with two added (weighted) features performs exactly 3% better in its weighted-average F1 score than its predecessor. Additionally, the three other metrics are also improved by 3% or more, its precision being more of an improvement than other metrics.

Q3c:

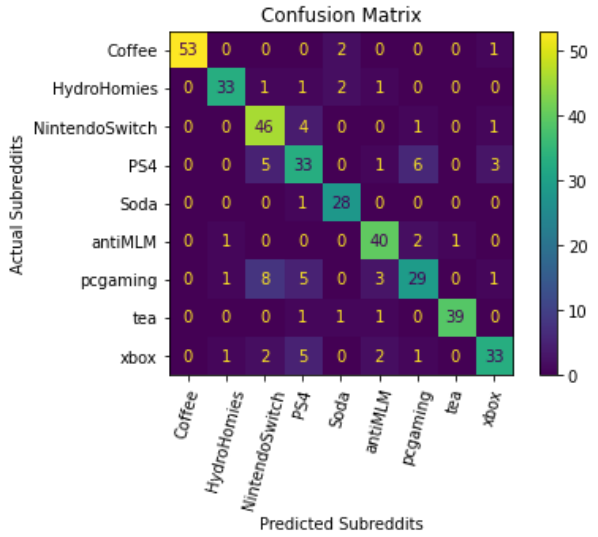

Figure 8: Confusion Matrix of Classifier from Figure 7[3]

As is evident from Figure 8, there seem to be fewer confusions between the PS4 and Xbox consoles, as well as between PS4 and PC, which possibly worked because of the author feature being implemented and splitting posts made by different posters into their respective subreddits. However, the confusions between PC and Nintendo Switch increased, which highlights a possible disadvantage of this feature – if a Redditor posts in two very similar subreddits, they are extra vulnerable to being confused between themselves. As such, possibly the feature should have less weight than initially suggested.

However, adding both features with their added weights seems to have improved the classifier overall because of their advantages mentioned in part a.

[3] Rao, Girish. How To Plot SKLearn Confusion Matrix With Labels? https://blog.finxter.com/how-to-plot-sklearn-confusion-matrix-with-labels/. Last accessed: 12 March, 2022.

Finally, even though this classifier has been improved much upon the initial classifiers in Q1, 83.4% is not a reliable performance services could depend on. As such, much improvement could be done in many key areas. Firstly, the models could be trained using cross-validation, which would not only ensure the dataset is not very biased, but also prevent overfitting on the data. Secondly, having very few manually chosen features plateaus the models' performance, which is why adding many features and then implementing feature selection by filtering or model search would allow the models to be trained on the features that distinguish between subreddits the most. Thirdly, Reddit post data having titles as very important features that posters think more about leads to possible improvements by upweighting (both the title and possibly its connection to the body of the post) and a more thoroughly analysed weight distribution of the features. Fourthly, a more human-readable error analysis could be done to help developers understand where exactly the confusions are rising from, e.g., by using LIME. With these improvements in mind, posters manually choosing which subreddit to post to could become a thing of the past.