Project 3: Classifying Subreddit Posts

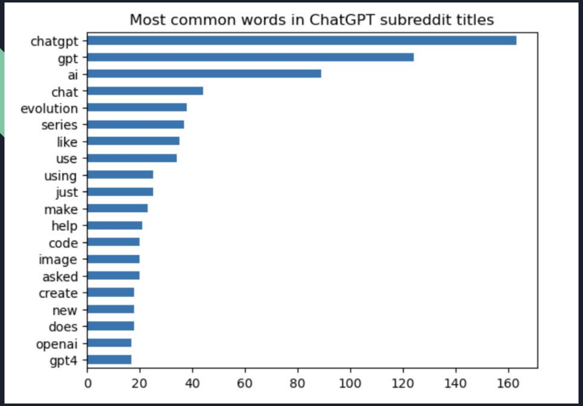
Problem Statement

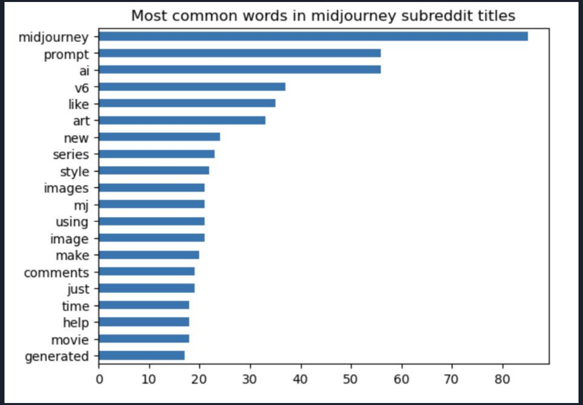
You are a financial analyst trying to classify media articles by subtopic across many media sources for subsequent sentiment analysis. To do this, build and optimize a model to classify posts from two different but similar subreddits, in this case ChatGPT and Midjourney.

Data Collection

- I used the PRAW Reddit API wrapper to pull posts from the ChatGPT and Midjourney subreddits
- I felt these were similar enough that there would be a significant number of overlapping words which might be challenging for the model to deal with
- I pulled the data over 5 days given API limits which were limiting me to around 100 posts per day after the initial pull
- My final dataset had ~3300 posts of which 60% were from Midjourney
- Our baseline was therefore 60.4%

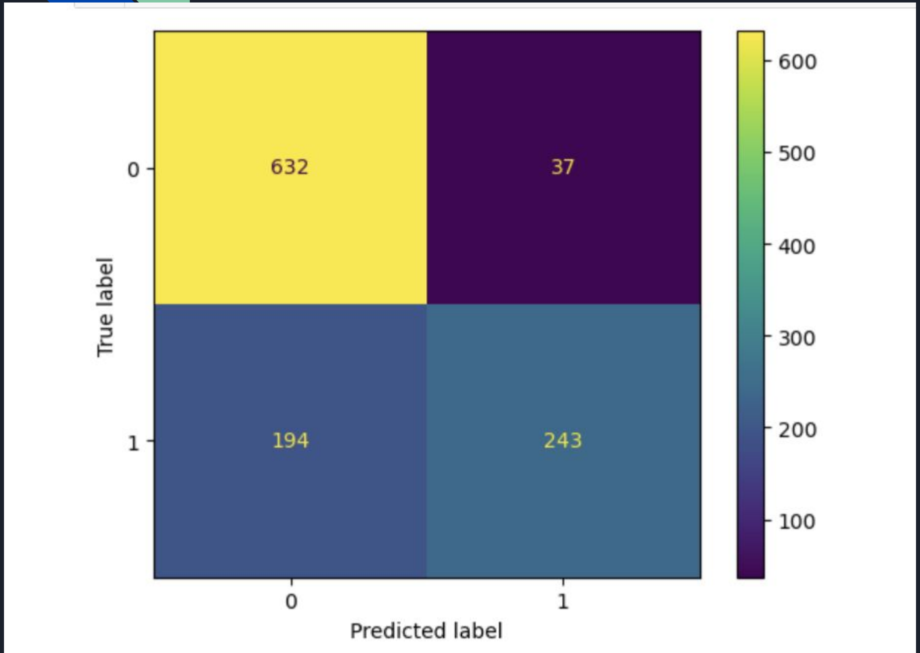
Most common words





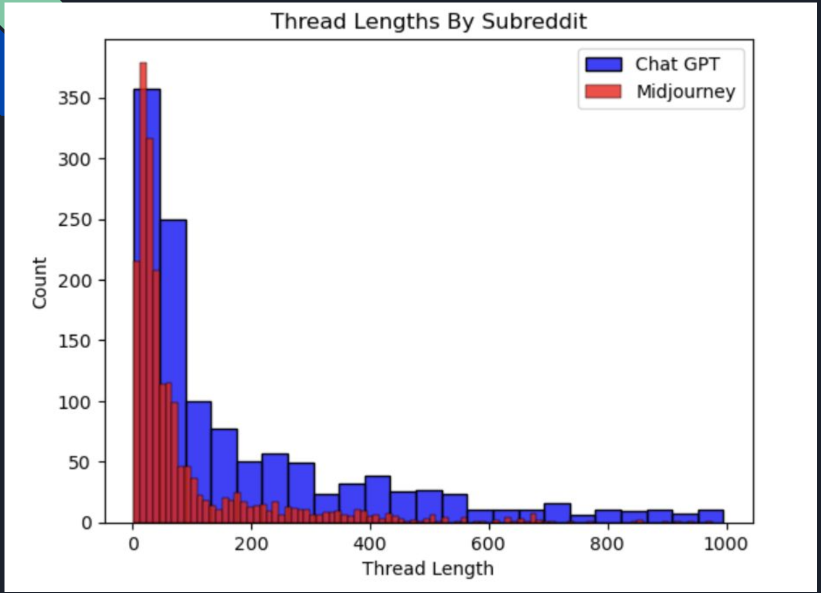
- As expected there were a number of very common words in each dataset which clearly helped the model classify subreddits
- Using a LR model on the titles only I got a test score of 0.81 which increased to 0.84 when I included post content
- I then removed the 10 most obvious words by including them in a modified stop word list and this led to a 0.05 drop in test score, but the model was still heavily overfit (0.95 train, 0.79 test using non optimized LR and count vectorizer)

Low sensitivity, high specificity



- A consistent problem for all models was the inability to correctly classify ChatGPT subreddits
- Here we show the confusion matrix from the previous model (LR, CVEC, stop words 10 most obvious words removed)
- Sensitivity (ie low false negatives, classifying ChatGPT which was 1) was only 55.6% here whereas specificity was 0.944

Low sensitivity, high specificity



- I looked into thread length and I believe this gives us some clue about the poor performance on ChatGPT posts
- Midjourney threads are significantly shorter than ChatGPT threads (title & post) which are more like Stack Overflow discussions
- Midjourney threads are often limited to titles for the artwork that is being presented and so the language is much more specific. Similarly I used TF-IDF which would tend to reward highly occurring language in rarely occurring documents across the corpus, so this might have had some relevance
- Also Midjourney was the majority class
- So the combination of the model correctly identifying posts given more distinct language across the majority class likely caused all models to be heavily biased to Midjourney

Hyperparameter tuning

	Training Score	Test Score	CV	Sensitivity	Specificity	Precision	Balanced Acc	f1
LR_cvec	0.721925	0.704340	{0.7072293588927776}	0.361556	0.928251	{0.7669902912621359}	{0.6449035925747298}	{0.8399506908418511}
LR_tvec	0.729055	0.695298	{0.7063315303849824}	0.478261	0.837070	{0.6572327044025157}	{0.6576655618379151}	{0.7363298636997051}
NB_tvec	0.691622	0.663653	{0.6684397868278715}	0.226545	0.949178	{0.7443609022556391}	{0.5878612499273139}	{0.8343840834583174}
RF_tvec	0.763815	0.694394	{0.7085666560610882}	0.370709	0.905830	{0.72}	{0.6382694892817928}	{0.8022947925860547}
SVM_tvec	0.840018	0.739602	{0.7335079939548204}	0.505721	0.892377	{0.7542662116040956}	{0.6990487527064884}	{0.8175295101775258}
KNN_tvec	0.523173	0.452080	{0.6871639357301941}	0.697941	0.291480	{0.39152759948652116}	{0.4947101620301485}	{0.33417614833544035}

- I tried a number of models, using TF-IDF as this gave me a better score for LR and so I took this forward
- I found that SVM gave me the best performance across the board
- I also tried adjusting the prediction threshold and this led to a worse performance across all models

Evaluation and conclusion

	Training Score	Test Score	CV	Sensitivity	Specificity	Precision	Balanced Acc	f1
LR_cvec	0.721925	0.704340	{0.7072293588927776}	0.361556	0.928251	{0.7669902912621359}	{0.6449035925747298}	{0.8399506908418511}
LR_tvec	0.729055	0.695298	{0.7063315303849824}	0.478261	0.837070	{0.6572327044025157}	{0.6576655618379151}	{0.7363298636997051}
NB_tvec	0.691622	0.663653	{0.6684397868278715}	0.226545	0.949178	{0.7443609022556391}	{0.5878612499273139}	{0.8343840834583174}
RF_tvec	0.763815	0.694394	{0.7085666560610882}	0.370709	0.905830	{0.72}	{0.6382694892817928}	{0.8022947925860547}
SVM_tvec	0.840018	0.739602	{0.7335079939548204}	0.505721	0.892377	{0.7542662116040956}	{0.6990487527064884}	{0.8175295101775258}
KNN_tvec	0.523173	0.452080	{0.6871639357301941}	0.697941	0.291480	{0.39152759948652116}	{0.4947101620301485}	{0.33417614833544035}

- Our final performance then was 0.74 test score (0.73 CV), though the model was still slightly overfit (0.84 training score)
- My main focus on the evaluation metrics was balanced accuracy - as the model was very good at dealing with the majority class and we had unbalanced data, standard accuracy was slightly misleading. Balanced accuracy as (specificity+sensitivity)/2 was a more useful benchmark and SVM was again the best performing model, with the highest sensitivity (other than KNN which performed poorly otherwise)
- SVM is generally effective in dealing with unbalanced datasets and complex decision boundaries which is likely the case for our data given the closeness of the subject matters
- Overall then our model significantly outperformed the baseline (60.4%)
- In further work I would like to look into oversampling as a first approach to dealing with weak sensitivity across the models