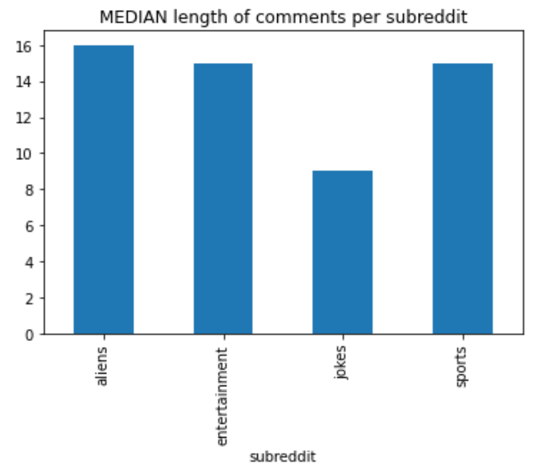
# Assignment 2

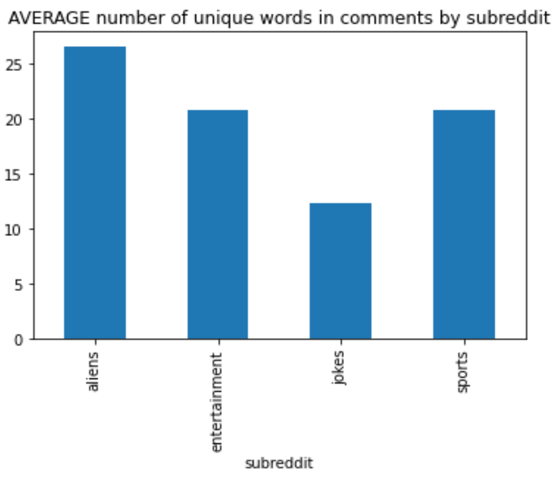
## Predicting a subreddit given an individual comment

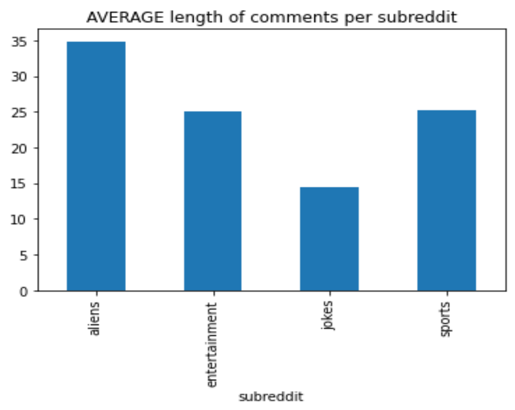
## PART 1

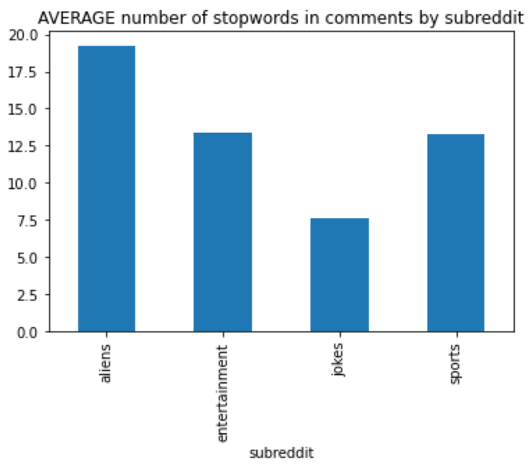
Reddit is an enormous social platform where millions of users communicate. We decided to create a project that would predict the subreddit given a comment; a classic classification task. Rather than using a particular dataset, we created our own dataset of comments from 4 randomly selected subreddits. The comments from each of these subreddits were attained using the Reddit API. The randomly selected subreddits were 'Jokes', 'entertainment', 'sports', and 'aliens'. After creating our dataset using the API, we cleaned our comments (data) by lowering the case of every comment, removing emojis, and punctuation.

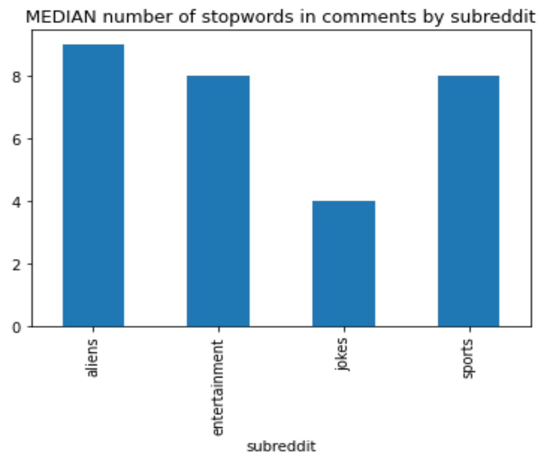
Since we are looking at words we did some exploratory data analysis on specific text statistics related to the comments from the subreddit. We maxed out the number of comments for each subreddit to be 15,000. There were no null/missing values within our data; which makes sense as Reddit's API grabs information directly from the subreddits.

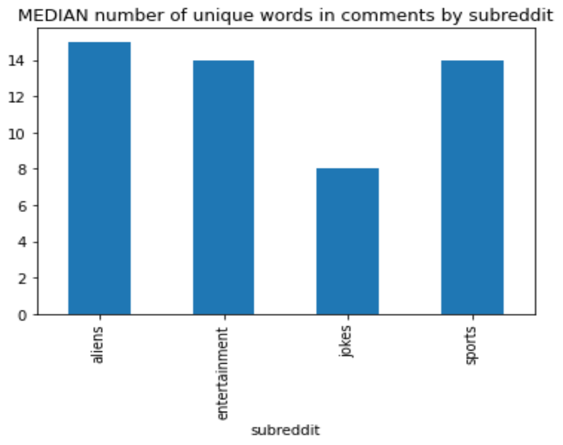






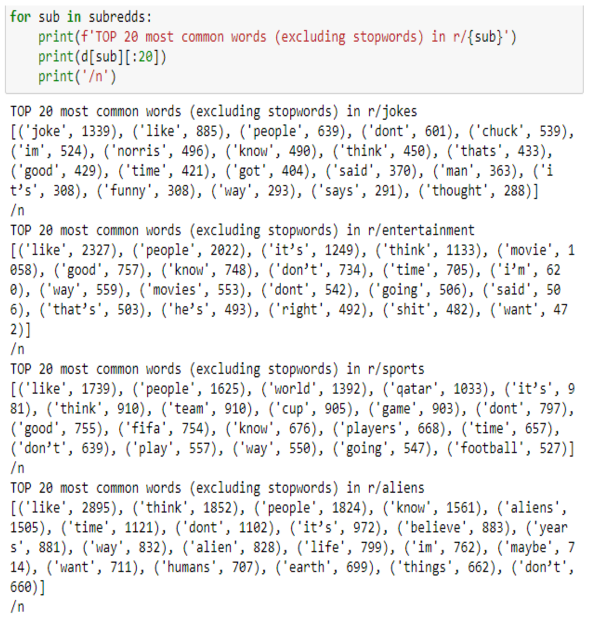






From these bar graphs we can see that entertainment and sports seem extremely similar to each other. This is not surprising though, as entertainment and sports go hand in hand as both these genres are connected to each other. The sporting industry is an entertainment industry. Sports are kind of a sub category of entertainment so one can expect overlaps of comments, topics, and qualities between the two subreddits. Aside from those two categories, aliens and jokes have different statistics. The aliens subreddit has the highest value in every bar graph. This could potentially be because of how complex a topic aliens are. Whether it be new conspiracy theories or sightings that a user might share, the information within these subreddits tend to have more information and longer comments, allowing users to clearly share their information with others. Coming to the jokes subreddit we can see that it has the least value in every graph by a significant amount. This also makes sense because jokes are usually short and straightforward. Many of them are depicted by pictures with only a few words as well, explaining such low values from the jokes subreddit.

Let's look at some of the most common words, excluding stopwords, to see if we can gain some more insight about each subreddit.

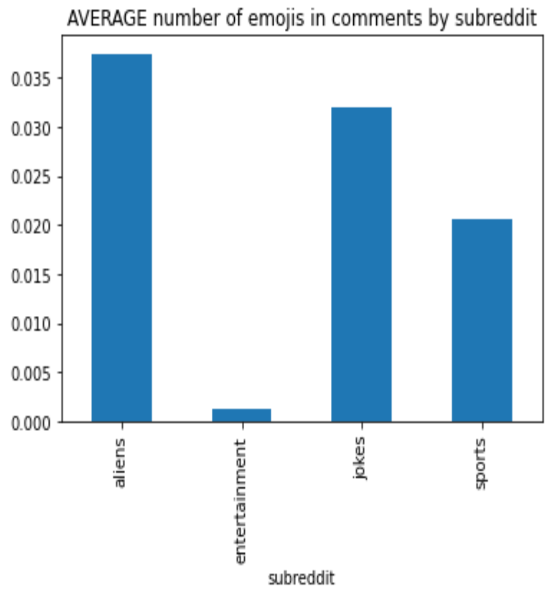


```
for sub in subredds:
    print(f'TOP 20 most common words (excluding stopwords) in r/{sub}')
    print(d[sub][:20])
    print('/n')
```

```
TOP 20 most common words (excluding stopwords) in r/jokes
[('joke', 1339), ('like', 885), ('people', 639), ('dont', 601), ('chuck', 539),
('im', 524), ('norris', 496), ('know', 490), ('think', 450), ('thats', 433),
('good', 429), ('time', 421), ('got', 404), ('said', 370), ('man', 363), ('i
t's', 308), ('funny', 308), ('way', 293), ('says', 291), ('thought', 288)]
/n
TOP 20 most common words (excluding stopwords) in r/entertainment
[('like', 2327), ('people', 2022), ('it's', 1249), ('think', 1133), ('movie', 1
058), ('good', 757), ('know', 748), ('don't', 734), ('time', 705), ('i'm', 62
0), ('way', 559), ('movies', 553), ('dont', 542), ('going', 506), ('said', 50
6), ('that's', 503), ('he's', 493), ('right', 492), ('shit', 482), ('want', 47
2)]
/n
TOP 20 most common words (excluding stopwords) in r/sports
[('like', 1739), ('people', 1625), ('world', 1392), ('qatar', 1033), ('it's', 9
81), ('think', 910), ('team', 910), ('cup', 905), ('game', 903), ('dont', 797),
('good', 755), ('fifa', 754), ('know', 676), ('players', 668), ('time', 657),
('don't', 639), ('play', 557), ('way', 550), ('going', 547), ('football', 527)]
/n
TOP 20 most common words (excluding stopwords) in r/aliens
[('like', 2895), ('think', 1852), ('people', 1824), ('know', 1561), ('aliens',
1505), ('time', 1121), ('dont', 1102), ('it's', 972), ('believe', 883), ('year
s', 881), ('way', 832), ('alien', 828), ('life', 799), ('im', 762), ('maybe', 7
14), ('want', 711), ('humans', 707), ('earth', 699), ('things', 662), ('don't',
660)]
/n
```

Even though the graphs show the entertainment and sports subreddits as very similar, they seem to be very different. For the sports subreddit we can see that FIFA is primarily being talked about because the World Cup is currently going on. As for the rest of the subreddits, many of the most common words in each are very similar. There are some unique words that we might associate each subreddit with; but overall many of the common words can easily be intertwined between each category.

Let’s look at the average number of emojis per subreddit to see if we can differentiate each of the subreddits a little more.



The subreddits with the most amount of emojis are the aliens and jokes subreddit. This is not surprising to us as depictions of aliens and information related to them can easily be displayed in pictures. There are numerous emojis related to that genre. Same goes for jokes. Jokes are often responded to and created using emojis so it’s not a surprise it has the second highest average emoji count in its comments. However, we are surprised by the fact that jokes don’t contain the highest value. Apart from these two subreddits the sports subreddit is the third highest while the entertainment subreddit is very very low and distinct compared to the others. This is kind of surprising because of how low this value is. However, these emoji counts have given us more insight on our dataset.

From our EDA we can start to see some of the problems we might encounter and have to overcome. In addition to that, it has also given us insightful distinctions between each of the subreddits that we can incorporate into our model.

## PART 2

The predictive task we want to study with this dataset is predicting the subreddit given a comment. We will evaluate our model using accuracy as our statistic. We are doing categorical classification so we decided to go with accuracy as it can tell us how our model is performing. There are a multitude of features that we want to explore when it comes to our prediction through text mining. Here are our features that we created:

All of these features were computed after we cleaned our comments by removing emojis, punctuation, and making it lowercase. We didn’t use emojis as a feature because of how scarce they were. Almost more than 95% of the comments contained no emojis so it wouldn’t really be an effective feature.

1. Bag of Words with stopwords
    a. To create this feature we created a vector, for each comment, whose indices refer to a unique word in our dictionary. The values for each of these indices in our vector represent how often that specific word occurs in a comment.
2. Bag of Words excluding stopwords
    a. To create this feature we created a vector, for each comment, whose indices refer to a unique word in our ictionary that excludes stop words. The values

for each of these indices in our vector represent how often that specific word occurs in a comment.
3. Number of words
    a. To create this feature, we used the number of words within the comment
4. Number of stopwords
    a. To create this feature, we used the number of stopwords within the comment
5. Ratio of stopwords to total_num_words
    a. To create this feature, we divided the number of stopwords with number of words within the comment
6. Number of unique words
    a. To create this feature, we counted the number of unique words within the comment
7. TFIDF
    a. To create this feature, we used sklearn's TfidfVectorizer to create our feature vector from the comments

After creating these features, we used Logistic Regression on them to predict our results.

## PART 3

We tried multiple models from the features we created above. We tried combinations of models from the first 5 features that we created ourselves as the Tfidf vectorizer was a method of its own. The model that worked best for us was the TFIDF vectorizer. We optimized it by tuning our regularization parameter. As we increased it, it started to overfit the data with barely any increase in validation accuracy. The c value that we believe was best for the model without overfitting the data was 1. Other models we considered were logistic regression models with a combination of features from the first 6 that we created ourselves. The best combination that worked from those were the 2:Bag of Words (excluding stopwords) and 5:Ratio of stopwords to total_num_words.

Unsuccessful attempts along the way were mostly due to the runtime of our models. We tried different models with our features such as RBF SVM, logistic regression with a higher dictionary count, and Gaussian Naive Bayes. RBF SVM was taking an extremely long time to run so we just stopped the cells running this model. For the Logistic Regression model with our features, when we tried higher dictionary counts it took a very long time as well so we stuck to a relatively lower dictionary count. When trying to run Naive Bayes, even in Google Colab, our notebook would just crash because there was too much memory being used. Therefore, in the end we stuck with logistic regression and tuned hyper parameters to see how much we could improve our various models with our feature combinations and Tfidf vectorizer.

The strengths and weaknesses of our different logistic regression models are mainly overfitting. Our first logistic regression model with features 2 and 5, performs a little worse than our TFIDF vectorizer but it doesn't overfit. The TFIDF does perform relatively higher but it also greatly overfits the data where the train and validation accuracy differences are between 10-12%. To adjust for this, we tried to find a c that could keep the accuracy higher than our first model but lower the difference between the train and validation for its model. One is less accurate and overfits less, but the other is more accurate and overfits a lot more. We do believe that RBF SVM would perform better in terms of accuracy and not overfitting as much but it just takes too long to run.

## PART 4

The dataset that was used for the prediction task came from Reddit. We scraped 20,000 comments on each of the top four most popular subreddits including ‘Jokes’, ‘entertainment’, ‘sports’, and ‘aliens'.’ We scraped by creating a scraping agent and looping through each of the subreddits and fetching the comments for each subreddit and saved the data that was collected to a dictionary which would be efficient for data cleaning, exploratory data analysis, and feature engineering, and model training, performance, and evaluation. Since scraping 20,000 comments in each Reddit takes over ten minutes, we decided to include four subreddits. Ideally, we would want to include many more than four subreddits but due to running time constraints on our laptops, this was not possible. The dictionary that includes all the comments of the subreddits was converted to a data frame after cleaning which included removing emojis, emoticons, symbols and pictographs, transport and map symbols, flags, and foreign characters, and removing punctuation and converting to lowercase for easier analysis. The data frame includes a text column that includes comments and a subreddit column that includes the type of subreddit that comment is under.

There are many past studies that involved prediction tasks associated with Reddit. One includes “Predicting sentiment of comments to news on Reddit" by Bruno Jakic. Another one includes “Predicting the Popularity of Reddit posts with AI” by Juno Kim which predicts the popularity of a Reddit post based on the number of upvotes it receives using a neural network model. Although a specific literature study hasn’t been done on this exact prediction task at hand, a similar one that we drew inspiration from was a paper called “Using Machine Learning to Predict the Popularity of Reddit Comments", which was done by Sean Deaton, Scott Hutchinson, and Suzanne Matthews from the Department of Electrical Engineering and Computer Science in the United States Military Academy. Many studies on prediction tasks related to Reddit are performed exclusively on posts, but the Popularity of Reddit Comments literature focused on the comments which we found to be interesting and unique. Since there were already many studies like the ones mentioned above which focused on popularity, we decided to take a different approach and predict the actual subreddit based on comments. The features that Deaton, Hutchinson, and Matthews used to train their model included bag of words, the sentiment of the comment, hour of creation, and controversiality. Though theirs focused on popularity, some of the same features were used in our model for predicting the correct subreddit such as bag of words.

## PART 5

The model that performed the best for us without taking an extremely long time to run was our TFIDF feature with logistic regression and a regularization constant of 0.7. We decided with this constant value because it minimized overfitting while also giving us a higher accuracy compared to other models. We tried using ngram but that

just continued to overfit the training data even more.
Train set: 48,000 comments

| Model | Training Accuracy | Validation Accuracy |
|---|---|---|
| TFIDF w/ Logistic Reg | 0.785 | 0.690 |
| Features 2 & 5 w/ Logistic Reg | 0.671 | 0.632 |
| Features 2 & 5 w/ RBF SVM, Trained on 10,000 comments only, C=1 | 0.646 | 0.588 |
| TFIDF w/ RBF SVM, Trained on 10,000 comments only, C=1 | 0.689 | 0.625 |
| TFIDF w/ RBF SVM, Trained on 20,000 comments only, C=1 | 0.776 | 0.657 |
| TFIDF w/ RBF SVM, Trained on 30,000 comments only, C=1 | 0.843 | 0.678 |

As seen from our results the model that performs the best is our TFIDF vectorizer with Logistic Regression. It does have an overfitting problem but with a little C constant tweaking we were able to lower it. The accuracy from this model is our highest compared to others. However, we believe that RBF SVM would be a better model than TFIDF if we could train it on our full train set. The only problem is that it takes too long to run. That's why we tried it with smaller subsets of the training set. Even with a smaller subset it does better than the LR model we created using our own features so we would expect it to perform the best when it can run entirely on the train set.
The significance of our results is that we can spend more time to create a better model with the necessary resources that allow us to run some more intensive models. Asides from that we can also try other text models such as word2vec and other approaches online.
The feature representations that worked well were TFIDF and Bag of Words (excluding stop words). The other features we created ourselves were still more generalized statistics that didn't give a large insight on a comment's qualities. Much of the accuracy is from either of these features. The other additional features added to the BoW (no stop words) model was to give additional information and boost to our models.
Our proposed model succeeded because of the TFIDF feature. TFIDF is a known feature that is good at classification tasks given text. Using the sklearn vectorizer we were able to create this feature without having such a long runtime. Therefore, we decided to use this final feature and model. As we expected, it performed the best in accuracy compared to our other models.
Other models failed because of run time and the notebook crashing. It was too intensive to compute with the given resources so we tried other models with a smaller training set. Also, all of our models overfit to our training data when we feed them larger data sets. Especially our

stronger models like RBF SVM and TFIDF with Logistic Regression.

## REFERENCES

Jakic, Bruno. “Predicting Sentiment of Comments to News on Reddit.” *Academia.edu*, 25 May 2014, https://www.academia.edu/3525454/Predicting_sentiment_of_comments_to_news_on_Reddit.

Kim, Juno. “Predicting the Popularity of Reddit Posts with Ai.” *ArXiv.org*, 17 June 2021, https://arxiv.org/abs/2106.07380.

*Using Machine Learning to Predict the Popularity of Reddit Comments*. https://www.seandeaton.com/publications/reddit-paper.pdf.