**Progress Report**

**Analyzing User Behavior, Trends and Communities using Reddit Posts**

**Sentiment Analysis of Subreddits**

**1. Goals:**

1. **Data Preprocessing:** The raw data collected will be cleaned and transformed into a suitable format for further analysis.
2. **Handling Sarcastic Comments:** A unique aspect of this project is dealing with sarcastic comments. The sentiment scores of these comments will be adjusted to reflect their true sentiment.
3. **Training a Baseline Model for Sentiment Analysis**: A baseline model will be trained for sentiment analysis. This model will provide a starting point for measuring the performance of more advanced models.
4. **Training an advanced Model:** To improve the accuracy of sentiment analysis, a neural network based model will be trained. This advanced model is expected to provide a more accurate prediction of sentiment scores.

**2. Introduction:**

The project began with the identification of trending topics on Reddit, a popular online platform where users can discuss and vote on content. By conducting a trend analysis, we were able to pinpoint the most discussed topics across a wide range of categories including Education, Sports, Entertainment, Food, Companies, Politics, Nature, Memes, Technology, and Gaming.To delve deeper into these topics, we focused on the most influential subreddits within each category. Subreddits are smaller communities within Reddit where users can share and discuss content related to a specific topic. By targeting these subreddits, we were able to gather a rich and diverse dataset for our analysis.

Data collection was carried out using PRAW (Python Reddit API Wrapper), a Python package that allows for simple access to Reddit’s API. We extracted various features from the data, including the category, comment creation time (UTC), subreddit, title, comment body, comment score, and comment author.

**3. Data Preprocessing:**

The data preprocessing stage involved several steps to clean and prepare the data for further analysis.

## **Text Cleaning:** The first step was text cleaning, which involved removing non-word characters from the text. This was done to ensure that the text data was free of any special characters or symbols that could interfere with the analysis.

## **Case Normalization:**The next step was case normalization, which involved converting all the text to lower case. This was done to ensure consistency in the data and to prevent the same words in different cases from being treated as different words.

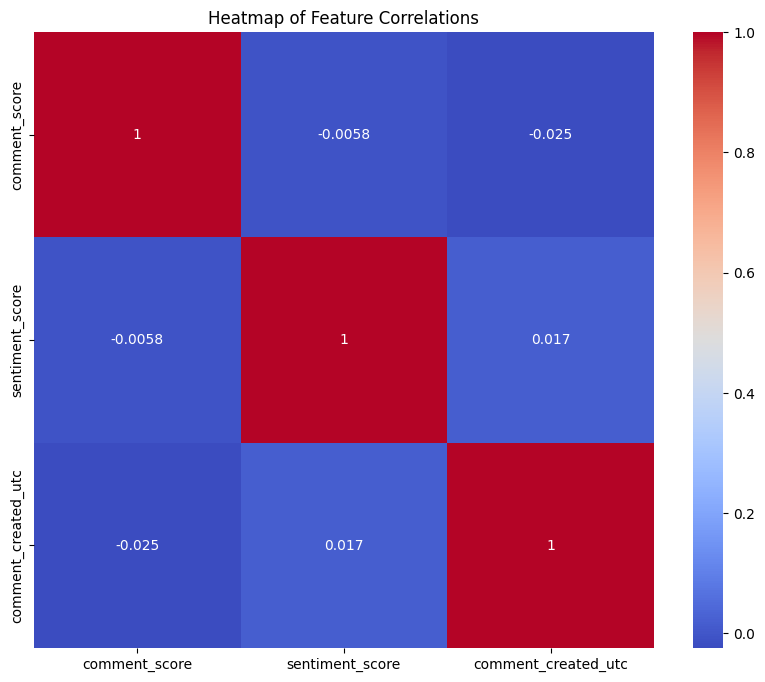
## **Tokenization:** After cleaning the text and normalizing the case, the text was tokenized. This involved splitting the text into individual words or tokens. This is a crucial step in text analysis as it allows us to analyze the text at the word level.

## **Stopword Removal and Lemmatization:** The next step involved removing stopwords and lemmatizing the tokens. Stopwords are common words like ‘is’, ‘the’, ‘and’, etc., that do not carry much meaning and are often removed in text analysis. Lemmatization is the process of reducing a word to its base or root form. For example, the words ‘running’, ‘runs’, and ‘ran’ would all be reduced to the base form ‘run’. This helps in reducing the dimensionality of the data and makes the analysis more efficient.

## **Handling Missing Values:** The dataset was then checked for missing values. Any rows with missing values in the ‘comment\_body’ column were dropped. Alternatively, missing values could be filled with an empty string or a string of your choice. This ensures that the dataset is complete and ready for analysis.

## **Handling Emojis and Emoticons:** The text data was then processed to handle emojis and emoticons. Emojis were converted to words, and the colons around the word representation of the emoji were removed. Emoticons, which are text representations of facial expressions, were replaced with their meanings. This helps in capturing the sentiment expressed through emojis and emoticons.

## **Handling GIFs:** Finally, the comments were processed to handle GIFs. Any URLs ending in .gif were identified, and the GIFs were downloaded and converted into a sequence of images. This was done to allow for the analysis of the visual content in the comments.



*Fig 1: Correlation Matrix for features after preprocessing*

Based on the correlation values, it appears that there is very little to no linear correlation between the pairs of features. This means that changes in one feature do not linearly correspond to changes in another feature.

**Independence of Features:** The features are largely independent of each other. Logistic regression assumes that the input features are independent of each other, so it can be used as a baseline model.

# **4. Handling Sarcasm in Sentiment Analysis:**

In the next phase of the project, we tackled the challenge of identifying and handling sarcastic comments in our dataset. Sarcasm can often invert the sentiment of a statement, making it difficult for sentiment analysis models to accurately predict the sentiment score. To address this, we developed a method to detect sarcastic comments and adjust their sentiment scores accordingly.

## **Training a Model on Sarcastic Reddit Comments**

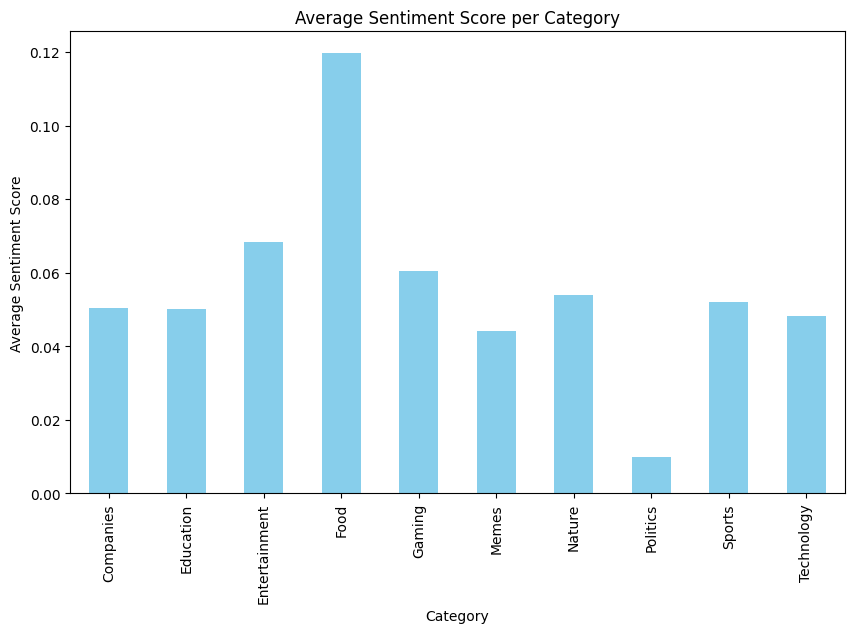
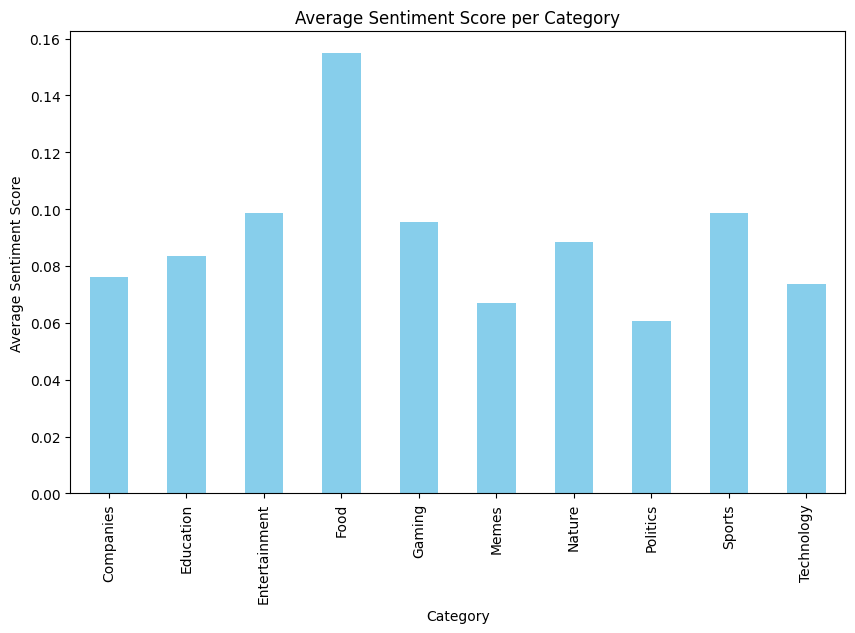
We began by training a model on a separate dataset of sarcastic Reddit comments. This dataset contains a wealth of Reddit comments that have been labeled as either sarcastic or non-sarcastic. We used a Naive Bayes model for this task, which is a popular choice for text classification problems. To prepare the text data for the model, we used a TF-IDF vectorizer, which converts the text into a matrix of TF-IDF features. TF-IDF, short for term frequency-inverse document frequency, is a numerical statistic that reflects how important a word is to a document in a collection or corpus.After splitting the data into training and test sets, we trained the Naive Bayes model on the training data and evaluated its accuracy on the test data.

**Predicting Sarcasm in the Original Dataset**

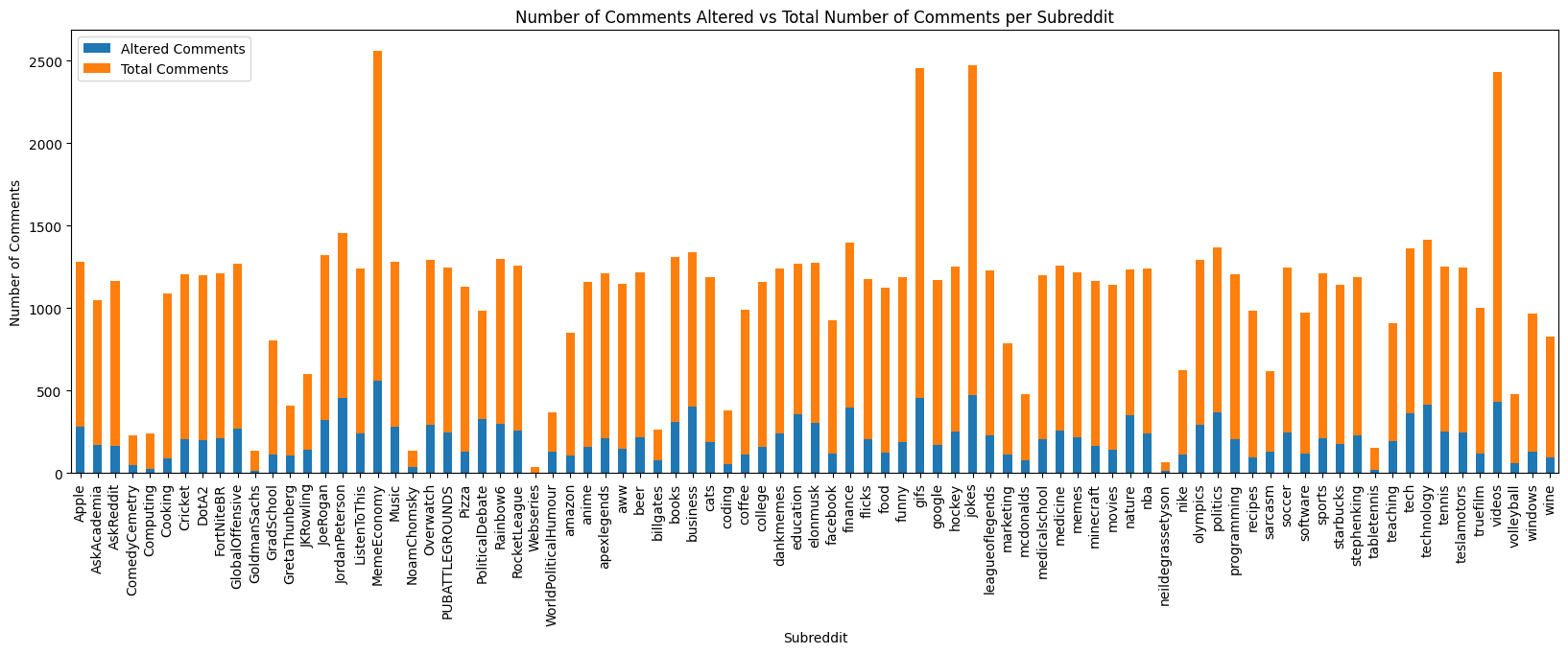
With the model trained, we then used it to predict whether the comments in our original dataset were sarcastic or not. We transformed the comments in the original dataset using the same TF-IDF vectorizer and made predictions using the trained model. The predictions were added as a new column in the original DataFrame, effectively labeling each comment as either sarcastic or non-sarcastic. In addition to the predictions, we also calculated the predicted probabilities of each comment being sarcastic and added these as another new column in the DataFrame.

## **Adjusting Sentiment Scores**

The final step was to adjust the sentiment scores of the comments based on whether they were predicted to be sarcastic or not. For each comment that was predicted to be sarcastic, we inverted its sentiment score.. For example, the statement “I just love getting stuck in traffic” is likely to be sarcastic, and it's true sentiment is negative, not positive. The Fig 2 below suggests the average sentiment reduces for all topics after we consider sarcastic comments.



*Fig 2: Compare avg sentiment scores before and after checking for sarcasm*



*Fig 3: Analyzing the the number of comments altered for various subreddits*

The plot in Fig 3 offers an insightful analysis into the prevalence of sarcasm across various subreddits. It distinguishes between 'altered' (sarcastic) and regular' comments within each subreddit. Notably, the subreddits 'politics', 'worldnews', 'news', 'funny', 'gifs', 'pics', 'aww', 'gaming', 'videos', and 'todayilearned' exhibit a higher incidence of sarcastic comments. This suggests that these communities frequently employ sarcasm in their discussions, possibly as a form of humor or critique. However, it's important to consider the specific context and culture of each subreddit, as these factors can significantly influence communication styles. This study underscores the pervasive use of sarcasm in online discourse and highlights its potential as a significant aspect of community interaction and engagement in subreddits that are based on Politics, Memes, Gaming in particular, which was quite expected.

# **5. Baseline Model for Sentiment Analysis:**

After preprocessing the data and handling sarcastic comments, we proceeded to build a baseline model for sentiment analysis on our dataset. The purpose of this model is to provide a benchmark for evaluating the performance of more advanced models. For our baseline model, we chose a simple Logistic Regression model.

**Data Preparation:** We started with a dataset containing Reddit comments, each with an associated sentiment score. The sentiment score is a continuous value, which we converted into a binary class (positive or negative) based on a threshold.

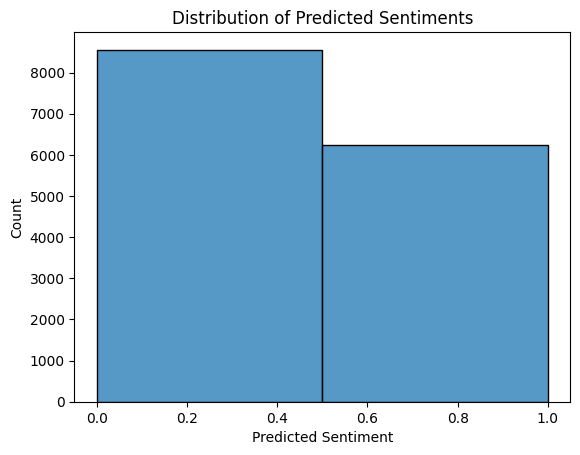
**Train-Test Split:** We split the dataset into a training set and a testing set using a 80-20 split. This means that 80% of the data was used for training the model, and the remaining 20% was used for testing its performance.

**Text Vectorization:** We transformed the text comments into a numerical format that could be used as input to a machine learning model. We did this using the CountVectorizer class from the sklearn.feature\_extraction.text module, which converts a collection of text documents to a matrix of token counts.

**Model Training:** We trained a Logistic Regression model on the training data. Logistic Regression is a simple yet powerful linear model for binary classification tasks, which makes it a good choice for a baseline model.

**Model Evaluation:** We evaluated the model’s performance by making predictions on the test data and comparing these predictions to the true sentiment classes. We used accuracy as our evaluation metric, which measures the proportion of correct predictions out of all predictions made.

**Analysis:** We plotted the training loss and accuracy over iterations to visualize how the model is learning over time. This can help identify issues like overfitting or underfitting and guide improvements to the model.

*Fig 4: Distribution indicating low bias for data*

The logistic regression model achieved an accuracy of 81.34% on the test set. The histogram of our predicted sentiment classes does not have a very high difference in heights for both bins, this suggests that our model is predicting both positive and negative sentiment classes about equally. This is generally a good sign, as it indicates that the model is not biased towards predicting one class over the other. The model is still predicting both classes equally, this could indicate that your model is performing quite well, as it’s not just predicting the majority class.

However it’s possible that the simplicity of the logistic regression model is contributing to the overfitting. Logistic regression is a linear model, which means it may not be able to capture complex patterns in the data. More complex models, such as neural networks, might be able to better capture these patterns and thus reduce overfitting.

# **6. Advanced Model: RNN for Sentiment Analysis:**

After establishing a baseline model, we decided to experiment with a more complex model to see if we could improve our results. We chose to use a Long Short-Term Memory (LSTM) model, which is a type of Recurrent Neural Network (RNN). LSTMs are particularly good at processing sequential data, making them a popular choice for text analysis tasks.

**Data Preparation:**

We started by encoding our target variable, using a LabelEncoder. This converted our categorical target variable into a format that could be used with our LSTM model.

Next, we tokenized our text data using Keras’s Tokenizer class. We set a vocabulary size of 10,000 words and used the ‘<OOV>’ token to represent words that were not in our vocabulary. We then converted our text data into sequences of tokens and padded these sequences to a consistent length of 128 tokens.

**Model Building:**

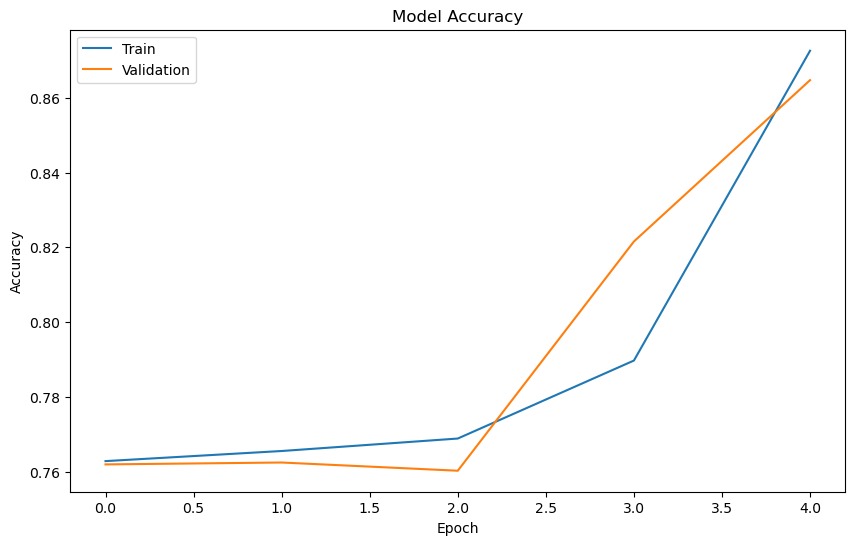
Our LSTM model was built using Keras’s Sequential API. We started with an Embedding layer, which converts our tokenized text data into dense vectors of fixed size. We set the output dimension of our Embedding layer to 128.

Next, we added an LSTM layer with 64 units. The LSTM layer analyzes the sequence of word vectors from the Embedding layer and encodes this sequence into a fixed-length vector.

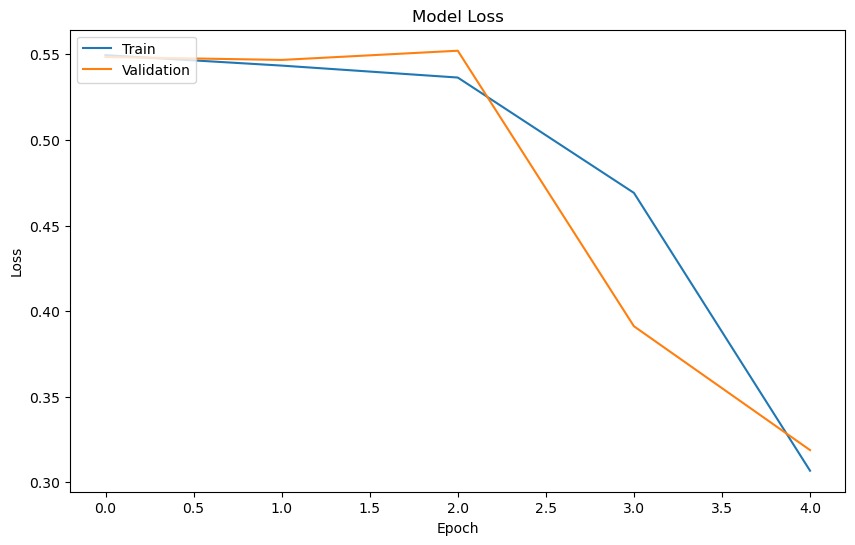
Finally, we added a Dense output layer with a sigmoid activation function. This layer takes the vector from the LSTM layer and outputs a probability that the input text is sarcastic.

**Model Training:**

We compiled our model with the Adam optimizer and binary cross-entropy loss, which is suitable for our binary classification task. We also decided to monitor accuracy during training.

We trained our model for 5 epochs with a batch size of32. We used 20% of our training data as a validation set to monitor our model’s performance on unseen data during training.*Fig 5: Model Accuracy vs Epochs*

The LSTM model achieved an accuracy of 86.87% and the baseline model had an accuracy around 81.34%, then the LSTM model has indeed outperformed the baseline model. This suggests that the added complexity of the LSTM model, which can capture longer-term dependencies in the text data, has led to improved performance on this task.



*Fig 6: Loss vs Epoch*

Fine-tuning the model can potentially lead to improvements in its performance. The sudden increase in accuracy and decrease in loss from epoch 2 to 5 suggests that the model is learning and improving its predictions over time.

**6. Next Steps:**

**Fine-Tuning Pretrained Models**

After establishing a baseline model and training an initial BERT model, the next step is to fine-tune these models to better fit our specific task. Fine-tuning involves continuing the training process on our specific task (sentiment analysis on subreddit comments) while keeping the pretrained weights. This allows the model to adjust to the specifics of our task while retaining the general language understanding capabilities learned from pretraining.

**Experimenting with Various Pretrained Models**

In addition to BERT, there are many other pretrained models available that we can experiment with. These include but are not limited to:

**DistilBERT:** A smaller, faster, cheaper version of BERT.

**RoBERTa**: A version of BERT that uses a different training approach and has been shown to outperform BERT on several tasks.

**GPT-2/GPT-3**: Models that are particularly good at generating coherent, contextually relevant sentences.

By training and fine-tuning these models on our task, we can compare their performance and gain insights into which models are most effective for sentiment analysis on subreddit comments.

**Comparative Analysis**

Once we have trained and fine-tuned various models, we can perform a comparative analysis. This involves comparing the performance of the models using various metrics (accuracy, F1 score, etc.), and analyzing the types of errors they make. This will help us understand the strengths and weaknesses of each model, and guide us in choosing the best model(s) for our task.

**Reaching a Conclusion**

The final step is to use our best-performing model(s) to test our hypothesis: “Subreddits focused on specific topics have distinct sentiment patterns in their comments.” By applying our model(s) to a large number of comments from different subreddits, we can analyze the sentiment patterns in these subreddits. If certain topics consistently attract more positive or negative sentiment, this would provide evidence in support of our hypothesis.