INTO TO DATA SCIENCE HW2 REPORT

Belal Daboor: 206404360. Fadi Kees: 324222512.

JULY 12, 2024

Data Preprocessing and EDA

The source data was given unclean, hence we needed to clean it, Tokenize it and then Stem it for most of the evaluations.

The cleaning was done by following the classic setup when working with Natural Languages dataset.

1. We removed words that have no real meaning in the real world, essentially stop words that would have just confused the models. Using NLTK English stop words as source for words to remove.
2. After the unnecessary words were removed, we stemmed the dataset using the WordNetLemitizer
3. Then we added the part of speech tags using the nltk POS tagger(*5. Categorizing and Tagging Words*, n.d.)

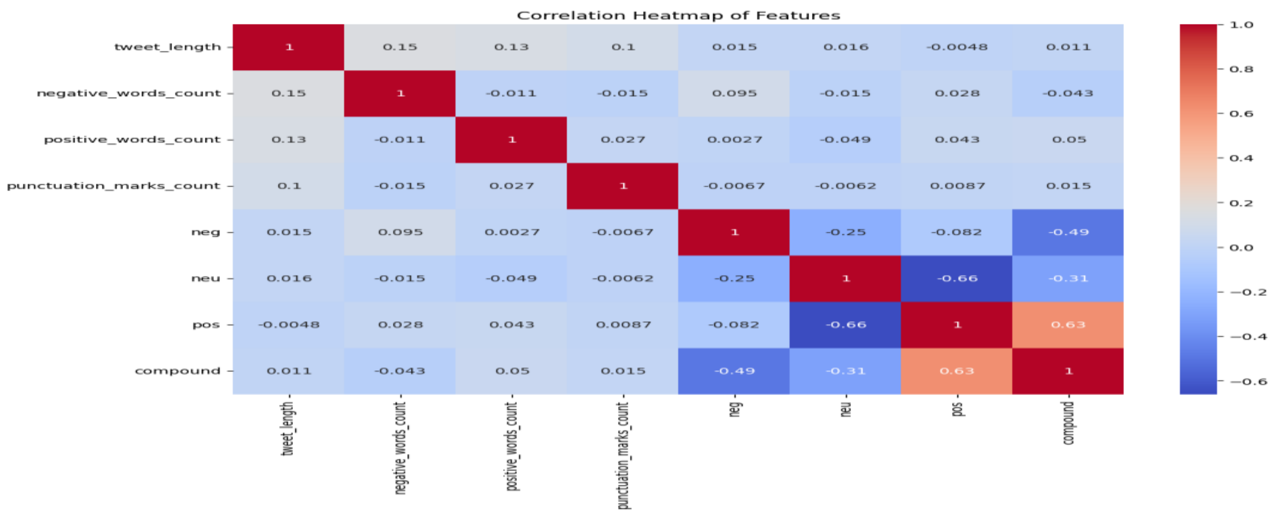
Data Cleaning Steps Summary

1. **Load the Dataset:**
   - **Explanation:** This step involves reading the data from CSV files into pandas DataFrames. It prepares the data for further processing by loading it into a format that is easy to manipulate and analyze.
2. **Remove Duplicates and Missing Values:**
   - **Explanation:** This step removes duplicate rows to ensure that each observation is unique, and handles missing values by removing rows with any missing data. This ensures data quality and integrity, preventing any potential issues during analysis.
3. **Preprocess Text:**
   - **Convert text to lowercase:** Standardizes the text data, making it case-insensitive. This helps in treating words like "Good" and "good" as the same word.
   - **Remove URLs:** Eliminates irrelevant information that does not contribute to sentiment analysis.
   - **Remove user mentions:** Strips out user-specific references that are not useful for sentiment analysis.
   - **Remove non-word characters:** Cleans the text by removing punctuation and special characters, except for exclamation and question marks which can convey sentiment.
   - **Handle negations:** Adjusts for negations by prepending "not_" to words following negation words, ensuring phrases like "not good" are interpreted correctly in terms of sentiment.
   - **Tokenize text:** Splits the text into individual words, which is essential for further text processing and analysis.
   - **Remove stopwords and lemmatize words:** Filters out common words that do not carry significant meaning (like "and", "the") and reduces words to their base forms (like "running" to "run"). This focuses on the meaningful content of the text.

EDA

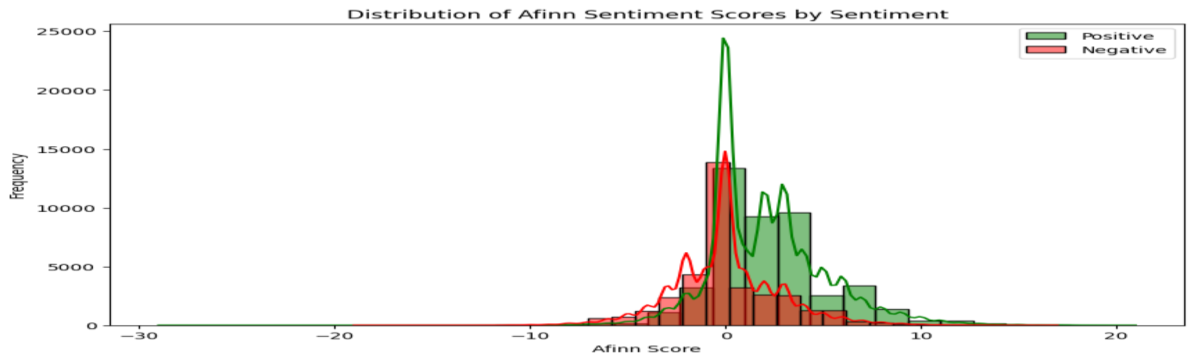
Correlation Heatmap of Features

# This heatmap shows the correlation between different features in the dataset, It helps identify which features are strongly correlated with each other. High correlation between features can be useful or problematic depending on the context.



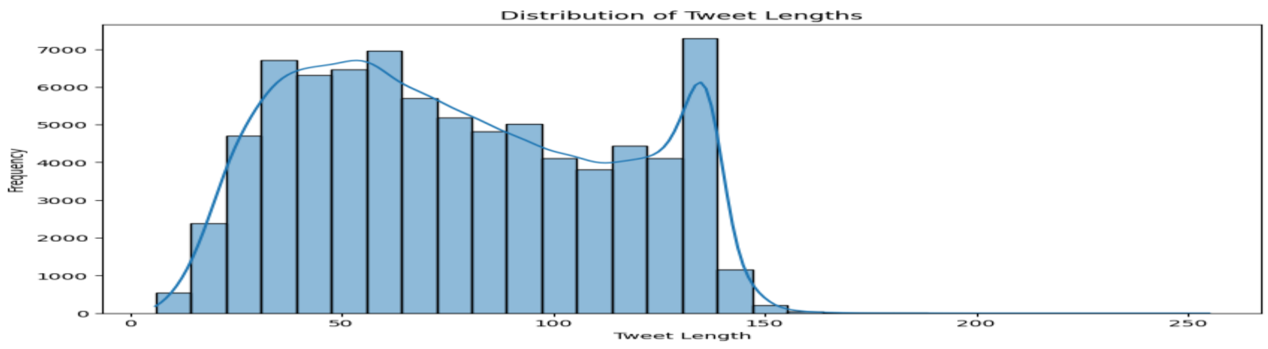
Distribution of Afinn Sentiment Scores by Sentiment

# This EDA shows the distribution of Afinn sentiment scores in the dataset. # It helps us understand how the Afinn scores are spread. # Afinn scores provide another perspective on sentiment, complementing VADER scores.



Distribution of Tweet Lengths

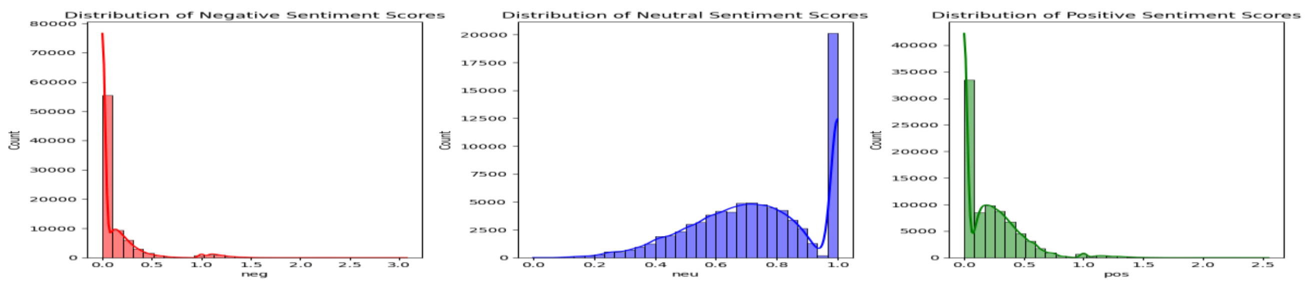
# This EDA shows the distribution of tweet lengths in the dataset. It helps us understand how the length of tweets varies.Tweet length can be an important feature, as longer or shorter tweets may correlate with sentiment.
So from this EDA Graph we can analyze and learn about the relation Between the sentiment score of the tweet text ,and the tweet length.



Distribution of Negative Sentiment Scores && Distribution of Neutral Sentiment Scores

&& Distribution of Positive Sentiment Scores

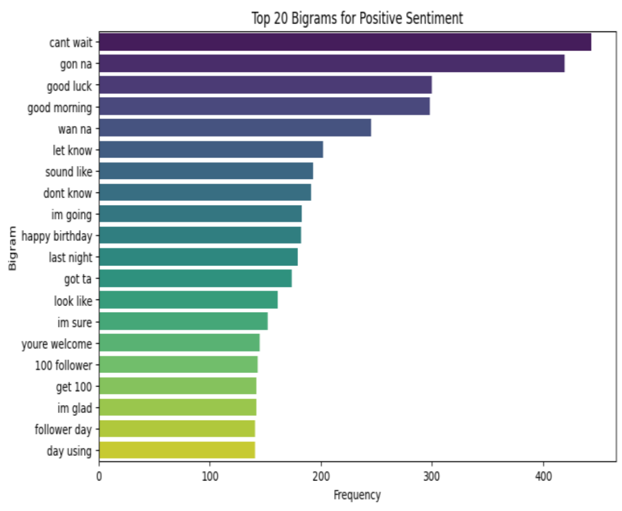
# This EDA shows the distribution of negative, neutral, and positive sentiment scores from VADER. It helps us understand how sentiment scores are spread across the dataset. This information can help in tuning the sentiment analysis model.

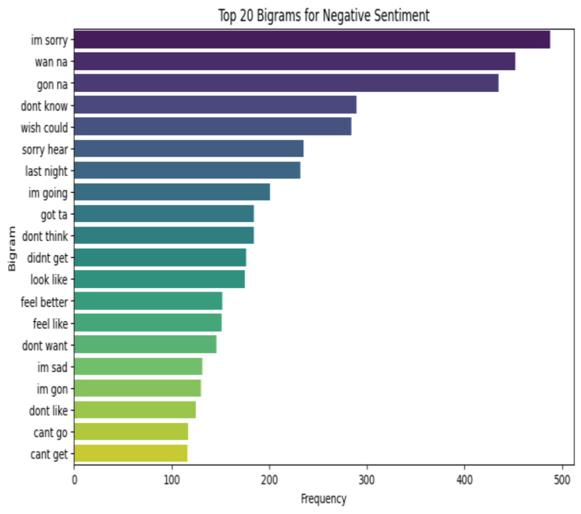


Top 20 Bigrams for Positive Sentiment && Top 20 Bigrams for Negative Sentiment

# This EDA shows the top 20 bigrams in positive sentiment tweets. It helps identify common phrases and patterns associated with positive sentiment. And it also shows the top 20 bigrams in negative sentiment tweets, which helps identify

common phrases and patterns associated with negative sentiment . These bigrams can be useful features for distinguishing positive sentiment AND negative sentiment.





Note: In the Code program File that we will submit, We exctracted another special EDA'S that we learned from the dataset .

Feature extraction

Feature Name	Feature Type	Description
afinn_score	Numeric	the Affinity Score we got on each row – using afinn sentiment calculation,Provides an additional sentiment score using the Afinn lexicon, offering another perspective on sentiment.
negative_words_count	Numeric	Counts occurrences of predefined negative words, indicating the intensity of negative sentiment.
positive_words_count	Numeric	Counts occurrences of predefined positive words, indicating the intensity of positive sentiment.
punctuation_marks_count	Numeric	Counts specific punctuation marks (e.g., exclamation and question marks), which can convey strong emotions.
likes_count	Numeric	Count of occurrences of the word "like".
retweets_count	Numeric	Count of occurrences of the word "retweet".
replies_count	Numeric	Count of occurrences of the word "reply"
pos_tags	Numeric	Frequency of each POS tag (e.g., NN, VB, JJ, etc.), Provides counts of different parts of speech (POS) tags, helping in understanding the grammatical structure related to sentiment..
mentions_count	Numeric	Count of occurrences of mentions (words starting with '@').
Neg	Normalized	Quantifies the negative sentiment in the text, which helps in understanding the extent of negative emotions expressed.
Pos	Normalized	Quantifies the positive sentiment in the text, which helps in understanding the extent of positive emotions expressed.
Neu	Normalized	Quantifies the neutral sentiment in the text, indicating parts of the text that are emotionally neutral.
Compound	Normalized	Provides an overall sentiment score combining positive, neutral, and negative scores, which gives a comprehensive view of the sentiment.
tweet_length	Numeric	Length of the cleaned text, Measures the length of the tweet, providing insight into how sentiment may vary with the length of the text. Longer or shorter tweets may have different sentiment patterns.
TF-IDF Features	Sparse Matrix	Extracted from the cleaned text. Captures the importance of words in the corpus, essential for text analysis.
Topic_0 to Topic_9 (Topic Extraction feature)	Numerical (Topic)	LDA topic distributions. Identifies and adds latent topics in the text using Latent Dirichlet Allocation (LDA), which can capture underlying themes and topics related to sentiment.

Additionally Explanation for the User Interaction Features:

- **Purpose:** Counts occurrences of interaction-related words (e.g., "like", "retweet", "reply", mentions), indicating user engagement and potentially sentiment.

Models

The model we decided to use was LightGBM, but here are models we checked and got the best results when doing split training.

- lightGBM
- Logistic Regression
- Naïve Bayes

LightGBM

LightGBM is an advanced implementation of gradient boosting designed for high performance and efficiency. It builds decision trees sequentially and focuses on reducing computation time. It's designed with large datasets in mind, hence it was made to be very resource efficient. Hence leading to it being very scalable, being able to handle a large amount of features and data. Because of its leaf-wise design the output of the model often times is very stable and reliable.

Gradiant Boosting: The basics of gradiant boosting is the model trains essembles of mini models. Usually Random forest over a dataset after a while making the Random Forests as a weak learning enhancing the accuracy of the model after each iteration.

Why Useful for Sentiment Analysis:

- **Capturing Complex Patterns:** Can capture complex interactions between features, which is beneficial for nuanced tasks like sentiment analysis.
- **Robustness:** Handles missing values and categorical features well, making it robust for real-world data.

Logistic Regression

Logistic Regression is a type of statistical model, that calculates the probability of a single independent variables as output. Used for gategorial regression as opposed to linear data of the Linear Regression algorithm which uses the same basic concepts but outputs a linear independent variable.

There are 3 types of logistic regressions models ,The **BINARY REGRESSION**,MULTINUMAL REGRESSION,**ORDINAL REGRESSION**, And the one we interested in Using is:

- Binary Regression (This Type was the one we used to train And Test our Data set): In this approach the output variable is a binary group meaning the output can either be 1 or 0. Some popular examples of this use case include, scam analyses and if a tumor is malignant or not.

Why Useful for Sentiment Analysis:

- **Capturing Complex Patterns:** Can capture complex interactions between features, which is beneficial for nuanced tasks like sentiment analysis.
- **Robustness:** Handles missing values and categorical features well, making it robust for real-world data.

XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful and scalable machine learning algorithm based on gradient boosting framework. It has gained popularity due to its robust performance and efficiency.

How It Works:

- **Boosting Framework**: XGBoost is built on the gradient boosting framework, which builds models in a sequential manner. Each new model aims to correct the errors of the previous models.
- **Ensemble Method**: It combines the predictions of several weak learners (usually decision trees) to form a strong predictor. The weak learners are trained sequentially, each trying to improve upon its predecessor.
- **Gradient Descent Optimization**: The model optimizes the loss function using gradient descent, ensuring that each new model minimizes the residual errors from the previous models.
- **Regularization**: XGBoost includes L1 (Lasso) and L2 (Ridge) regularization, which helps in reducing overfitting by penalizing large coefficients.
- **Parallel Processing**: It supports parallel processing, which makes it extremely fast and scalable for large datasets.
- **Handling Missing Values**: XGBoost can handle missing values internally, making it robust in real-world scenarios.

Purpose:

- **Accuracy**: XGBoost is known for achieving high accuracy on a wide range of machine learning problems.
- **Efficiency**: It is highly efficient in terms of both computation and memory usage, making it suitable for large datasets.
- **Flexibility**: The algorithm can be used for both regression and classification tasks and supports various objective functions.

Why It Is Beneficial for Text Analysis:

- **Handles High-Dimensional Data**: Text data often results in high-dimensional feature spaces (due to techniques like TF-IDF, word embeddings). XGBoost efficiently handles such high-dimensional data.
- **Feature Importance**: XGBoost provides feature importance scores, which help in understanding which words or features contribute the most to the predictions.
- **Regularization**: Its built-in regularization techniques help in managing overfitting, which is a common issue in text analysis due to the sparse nature of text data.
- **Speed and Scalability**: XGBoost is optimized for speed and performance, which is crucial when dealing with large text datasets.
- **Robustness**: It can handle noisy data and missing values, which are common in text data.

Model Evaluation

Most model evaluations happen with resampling evals. And running metrics on the excluded sample from the data. Such as Random Split, Time-Based Split, K-fold Cross Validation, Straitfield K-Fold, etc.

We chose to use the Random Split method, because our data is too random and isn't correlated to anything. Like time-series data and validating categories.

Because our model is a classification model, and according to most Medium posts about evaluating classification models, a matrix called the confusion matrix can be constructed which demonstrates the number of test cases correctly and incorrectly classified.

	Actual 0	Actual 1
Predicted 0	True Negative (TN)	False Negative (FN)
Predicted 1	False Positive (FP)	True Positive (TP)

- TN: Number of negative cases correctly classified.

- TP: Number of positive cases correctly classified.
- FN: Number of positive cases incorrectly classified as negative.
- FP: Number of negative cases correctly classified as positive.

Accuracy

Accuracy is the simplest metric and can be defined as the number of test cases correctly classified divided by the total number of test cases.

$$Acc = (TP + TN)/(TP + TN + FP + FN)$$

It can be applied to most generic problems but is not very useful when it comes to unbalanced datasets. For example if the Model is trying to predict a category that occurs at 1% we might get an accuracy measure of 99% when in fact we got FN for every place we needed to get TP.

Therefore, for such a case, a metric is required that can focus on the ten positive data points which were completely missed by the model

Precision

Precision is the metric used to identify the correctness of classification.

$$Pre = TP/(TP + FP)$$

This equation is the ratio of correct positive classifications to the total number of predicted positive classifications. The greater the fraction, the higher is the precision, which means better is the ability of the model to correctly classify the positive class.

Recall

Recall tells us the number of positive cases correctly identified out of the total number of positive cases.

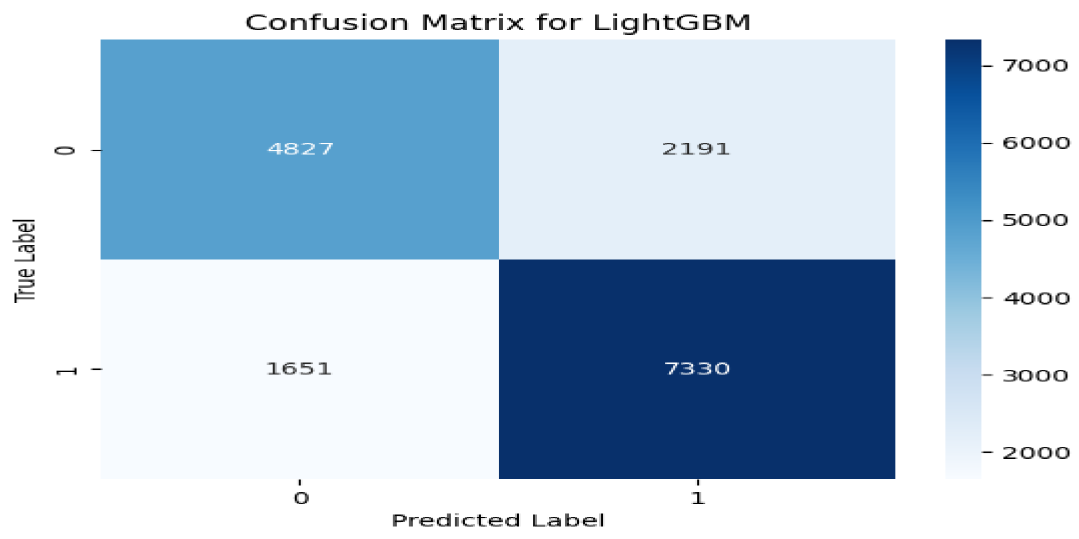
$$Recall = TP/(TP + FN)$$

F1 Score

F1 score is the harmonic mean of Recall and Precision and therefore, balances out the strengths of each. It is useful in cases where both recall and precision can be valuable. Such as Sentiment analyses.

$$F1\ Score = \frac{Pre * Recall}{Pre + Recall}$$

- Our Model's (LightGBM) Scores after training and evaluating on the Train File :



F1 Score: `0.792346773321803`.

Recall: `0.8161674646475894` .

Precision: `0.7698771137485558` .

Accuracy: `0.7598599912494531` .

Kaggle Competition

Our Group Name in the Kaggle competition is : FADI && BELAL.