

Project: Sentiment Analysis based on Random Forest Model

Date: 29/09/2023

Project Guide: Dr. Milan Gnjatovic

Name: Nikhil Chezian

Matriculation Number: 11027706

Table of Contents	
Acknowledgement	3
Abstract	4
Introduction	5
Program.....	6
Dataset	6
Data Preparation: Combining Twitter and Reddit Data.....	6
Data preprocessing	7
Random Forest Model.....	8
Output & Analysis	11
Data Visualization.....	13
Conclusion.....	14
References.....	15

Acknowledgement

I would like to take this opportunity to express my heartfelt gratitude to all those who have contributed to the successful completion of this project, which has been a significant milestone in my academic journey.

First and foremost, I would like to extend my deepest appreciation to my project guide, Prof. Milan Gnjatovic. His exceptional mentorship, profound knowledge, and unwavering support have been instrumental in shaping the direction of this project. Prof. Gnjatovic's dedication to academic excellence, tireless efforts in providing insightful feedback, and patient guidance have been invaluable assets throughout this endeavour. I am truly fortunate to have had the privilege of working under his supervision.

Finally, I want to express my heartfelt appreciation to my family and friends for their unwavering encouragement, understanding, and patience during this challenging yet rewarding journey. Their belief in me has been a constant source of motivation.

In conclusion, this project has been a significant learning experience and a testament to the power of collaboration, dedication, and knowledge sharing. It is my hope that the outcomes of this project will contribute positively to the field of sentiment analysis and inspire future research endeavors.

Thank you all for being an integral part of this project's success.

Nikhil Chezian

SRH Hochschule Heidelberg

29th September 2023

Abstract

Sentiment analysis, in one simple sentence, is the process of determining whether a piece of text expresses a positive, negative, or neutral sentiment. Sentiment analysis is a crucial aspect of this project, it provides valuable insights into public opinion, consumer feedback, and market trends. This project focuses on conducting a comprehensive sentiment analysis of social media data extracted from two distinct sources: Twitter and Reddit. The primary goal is to gain deep insights into the sentiments expressed in the textual content and to develop a robust sentiment classification model. Leveraging Natural Language Processing (NLP) techniques and machine learning, the project encompasses data loading, text cleaning, sentiment distribution visualization, and the application of a Random Forest Classifier. The resulting model's performance is evaluated on a subset of the data and can be extended to the full dataset. The code offers valuable insights into sentiment analysis and serves as a foundation for deeper explorations into social media sentiment. Understanding sentiment is essential for making informed decisions in various domains, including marketing, public relations, and customer service.

Introduction

Sentiment analysis involves examining digital text to identify whether the message's emotional tone is good, negative, or neutral. Companies now have a lot of text data, such as emails, chat transcripts from customer service, comments on social media, and reviews. This text can be scanned by sentiment analysis software to discover the author's viewpoint on a subject. Businesses use the data from sentiment analysis to boost brand recognition and customer service.

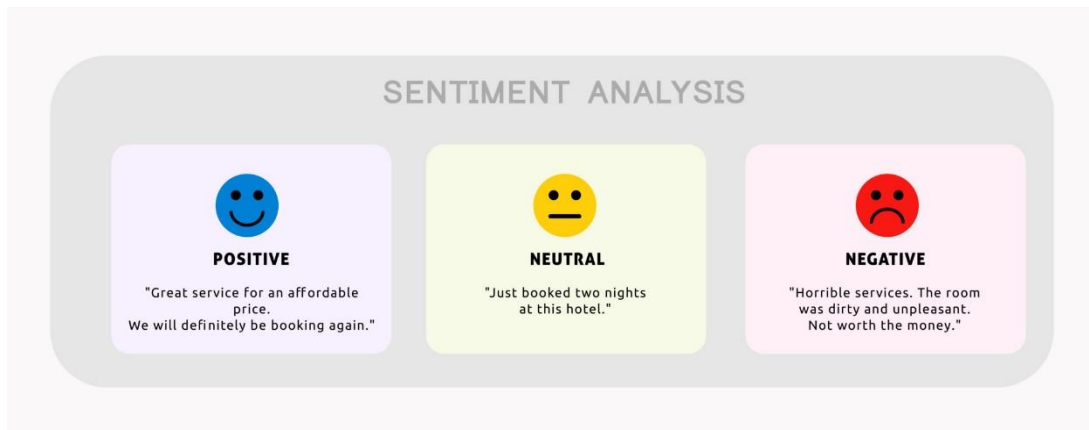


Fig.1: Fundamentals of Sentimental Analysis

Sentiment analysis, also known as opinion mining, serves as a vital tool in business intelligence, aiding companies in enhancing their offerings. Here are key advantages:

Sentiment analysis, or opinion mining, is a vital tool for businesses. It offers:

Objective Insights: AI-driven tools ensure unbiased evaluations of customer opinions, providing consistent and objective results.

Product Enhancement: Sentiment analysis helps improve offerings by identifying areas of concern, such as product performance issues.

Scalable Analysis: Cloud-based tools make it cost-effective to extract customer emotions from vast data sources.

Real-Time Responsiveness: Marketers can swiftly respond to trends and crises by monitoring real-time customer sentiments.

Sentiment analysis is crucial for objective evaluations, product improvements, scalability, and real-time decision-making in business.

Program

Dataset

Firstly, the dataset is a mixture of tweets from twitter and content from reddit. The platforms were chosen due to the versatility of the tone of content that can be found on said platforms.

An excerpt from the input data is displayed below to provide further clarity about its processing.

Clean_text category

0. family uddhi have never tried explain them t...	1
1. Buddhism has very much lot compatible with ch...	1
2. seriously don say thing first all they won ge...	-1
3. what you have learned yours and only yours wh...	0
4. for your own benefit you may want read living ...	1

The dataset above, has two columns:

clean_text: This column contains the textual content of statements or comments.

Category: This column contains sentiment labels for each statement or comment. Sentiment labels are represented using numeric values:

1 : to represent positive sentiment.

0 : to represent neutral or mixed sentiment.

-1 : to represent negative sentiment.

Data Preparation: Combining Twitter and Reddit Data

In this project, Twitter and Reddit data were combined to create a comprehensive dataset for sentiment analysis. The following steps were taken:

1. Twitter Data Preparation:

Twitter data was loaded from a CSV file.

Rows without sentiment labels were removed.

Text data underwent preprocessing, including tokenization, lemmatization, and stopwords removal.

2. Reddit Data Preparation:

Reddit data was loaded from a separate CSV file.

Like Twitter data, rows without sentiment labels were excluded.

Text data was pre-processed using the same approach.

3. Data Fusion:

Both datasets were merged to form a single dataset.

This combined dataset was then split into training and testing subsets.

This approach aimed to create a robust sentiment analysis model by leveraging data from both Twitter and Reddit, capturing a diverse range of language patterns and expressions commonly found in these platforms.

```
26 # Load Twitter Data
27 twitter_data = pd.read_csv("C:/Users/nchez/Desktop/CP_Sentiment/archive/Twitter_Data.csv", encoding="utf-8")
28 twitter_data = twitter_data.dropna(subset=['category'])
29 twitter_data['clean_text'] = twitter_data['clean_text'].apply(preprocess_text)
30
31 X_twitter = twitter_data['clean_text']
32 y_twitter = twitter_data['category']
33 X_train_twitter, X_test_twitter, y_train_twitter, y_test_twitter = train_test_split(X_twitter, y_twitter, test_size=0.3, random_state=42)
34
35 # Load Reddit Data
36 reddit_data = pd.read_csv("C:/Users/nchez/Desktop/CP_Sentiment/archive/Reddit_Data.csv", encoding="utf-8")
37 reddit_data = reddit_data.dropna(subset=['category'])
38 reddit_data['clean_text'] = reddit_data['clean_text'].apply(preprocess_text)
39
40 X_reddit = reddit_data['clean_text']
41 y_reddit = reddit_data['category']
42 X_train_reddit, X_test_reddit, y_train_reddit, y_test_reddit = train_test_split(X_reddit, y_reddit, test_size=0.3, random_state=42)
43
44 # Combine Twitter and Reddit Data
45 X_train_combined = pd.concat([X_train_twitter, X_train_reddit], ignore_index=True)
46 y_train_combined = pd.concat([y_train_twitter, y_train_reddit], ignore_index=True)
47
48 X_test_combined = pd.concat([X_test_twitter, X_test_reddit], ignore_index=True)
49 y_test_combined = pd.concat([y_test_twitter, y_test_reddit], ignore_index=True)
```

Fig.2.1: Illustration of loading twitter & reddit datasets and merging to create a unified dataset.

Data preprocessing

In this code, the preprocessing of text data is done within the preprocess_text function. This function is applied to the text data from both Twitter and Reddit datasets before it is used for training and evaluation.

```
14 def preprocess_text(text):
15     if pd.isna(text):
16         return ""
17
18     lemmatizer = WordNetLemmatizer()
19     stop_words = set(stopwords.words('english'))
20     tokens = nltk.word_tokenize(text)
21     tokens = [lemmatizer.lemmatize(word.lower()) for word in tokens if word.isalnum() and word.lower() not in stop_words]
22     return ' '.join(tokens)
```

Fig.2.2: Illustration of the pre-processing function.

Handling Missing Values: The function checks if the text is missing (NaN), and if so, it returns an empty string to handle missing values gracefully.

Tokenization: The text is tokenized using `nltk.word_tokenize`, which breaks the text into individual words (tokens).

Lemmatization: Each token is lemmatized using the WordNet lemmatizer (`WordNetLemmatizer()`).

Lemmatization reduces words to their base or root form, which helps in standardizing and reducing the dimensionality of the text data. For example, “running” and “ran” would both be reduced to “run.”

Lowercasing: All tokens are converted to lowercase using `word.lower()`. Lowercasing ensures that words with different capitalizations are treated as the same word.

Stopword Removal: Stopwords, which are common words like “the,” “and,” “is,” etc., that do not carry significant meaning, are removed from the text. The set of stopwords utilized in this project are taken from NLTK ENGLISH_STOP_WORDS.

Alphanumeric and Stopword Filtering: Finally, only alphanumeric words are retained.

Vectorizing— converting textual data into numerical format.

Most machine learning algorithms require numerical input data. Text data, consisting of words and sentences, cannot be directly fed into these algorithms. Vectorization converts text data into numerical vectors, allowing machine learning models to process and learn from the text. Each word or n-gram (sequence of words) becomes a feature, and its presence or frequency can carry valuable information for classification or analysis tasks

```
67 # Vectorize the text data
68 vectorizer = CountVectorizer(max_features=5000)
69 X_train_combined_vec = vectorizer.fit_transform(X_train_combined)
70 X_test_combined_vec = vectorizer.transform(X_test_combined)
```

Fig.2.3: Illustration of vectorization

CountVectorizer: This technique turns text documents into a matrix of word counts. Each row represents a document, and each column is a unique word.

Max_features: Limits the number of unique words in the matrix, reducing dimensionality, in this project, it is set to 5000 meaning that only the top 5000 most frequent words will be included in the matrix.

Fit_transform: Analyzes and transforms the training data (`X_train_combined`) into the word count matrix.

Transform: Applies the same transformation to the test data (`X_test_combined`), ensuring consistent features.

Random Forest Model

Random forests are ensembles of decision trees. Random forests are one of the most successful machine learning models for classification and regression. They combine many decision trees in order to reduce the risk of overfitting. Like decision trees, random forests handle categorical features, extend

to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions.

Random forests train a set of decision trees separately, so the training can be done in parallel. The algorithm injects randomness into the training process so that each decision tree is a bit different. Combining the predictions from each tree reduces the variance of the predictions, improving the performance on test data.

Python supports random forests for binary and multiclass classification and for regression, using both continuous and categorical features. It implements random forests using the existing decision tree implementation.

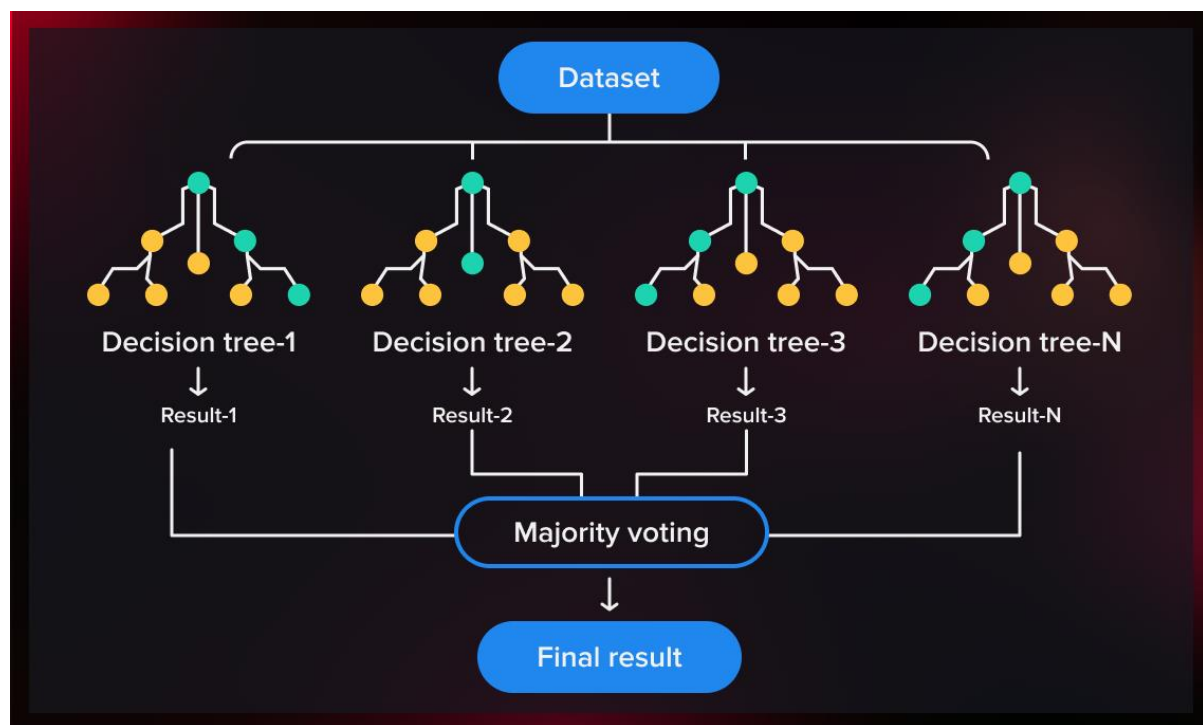


Fig.2.4: Architecture of Random Forest Model

The random forest model is a type of additive model that makes predictions by combining decisions from a sequence of base models. More formally we can write this class of models as:

$$g(x) = f_0(x) + f_1(x) + f_2(x) + \dots$$

where the final model g is the sum of simple base models f_i . Here, each base classifier is a simple decision tree. This broad technique of using multiple models to obtain better predictive performance is called model ensemble. In random forests, all the base models are constructed independently using a different subsample of the data.

In this project, the Random Forest Classifier is trained on a subset of the combined dataset (the size of the subset is adjustable with `subset_size`). This allows for faster processing during development and testing.

The model is evaluated on the subset dataset, and metrics such as accuracy, classification report, and the confusion matrix are printed.

```
75 # Run on a subset of the combined dataset
76 subset_size = 10000 # Adjust this size as needed for faster processing
77 X_train_subset = X_train_combined_vec[:subset_size]
78 y_train_subset = y_train_combined[:subset_size]
79
80 rf_classifier.fit(X_train_subset, y_train_subset)
81 y_pred_combined_subset = rf_classifier.predict(X_test_combined_vec)
82
83 # Evaluate the subset dataset
84 accuracy_combined_subset = accuracy_score(y_test_combined, y_pred_combined_subset)
85 print(f"Subset Dataset Accuracy: {accuracy_combined_subset:.2f}")
86
87 report_combined_subset = classification_report(y_test_combined, y_pred_combined_subset)
88 print("Subset Dataset Classification Report:\n", report_combined_subset)
89
90 # Calculate and print the confusion matrix
91 confusion_matrix_subset = confusion_matrix(y_test_combined, y_pred_combined_subset)
92 print("Subset Dataset Confusion Matrix:\n", confusion_matrix_subset)
```

Fig.2.5: Illustration of operation performed on the subset.

The user is given the option to apply the model to the full dataset (`apply_to_full_dataset` variable). If the user chooses 'yes,' the model is trained on the full dataset and evaluated on it as well.

```
94 # Ask if the user wants to apply it to the full dataset
95 apply_to_full_dataset = input("Do you want to apply it to the full dataset? (yes/no): ").strip().lower()
96 if apply_to_full_dataset == 'yes':
97     rf_classifier = RandomForestClassifier(n_estimators=100,max_depth=20,random_state=42)
98     rf_classifier.fit(X_train_combined_vec, y_train_combined)
99     y_pred_combined = rf_classifier.predict(X_test_combined_vec)
100
101     # Evaluate the full dataset
102     accuracy_combined = accuracy_score(y_test_combined, y_pred_combined)
103     print(f"Full Dataset Accuracy: {accuracy_combined:.2f}")
104
105     report_combined = classification_report(y_test_combined, y_pred_combined)
106     print("Full Dataset Classification Report:\n", report_combined)
107
108     # Calculate and print the confusion matrix for the full dataset
109     confusion_matrix_full = confusion_matrix(y_test_combined, y_pred_combined)
110     print("Full Dataset Confusion Matrix:\n", confusion_matrix_full)
```

Fig.2.6: Illustration of operation performed on the entire dataset.

Output & Analysis

The output for this model is derived mainly using 2 functions classification report and the confusion matrix.

A classification report summarizes a classification model's performance using four key metrics: precision, recall, F1-score, and support. It helps evaluate how well the model identifies different classes or categories.

- Precision: Measures accurate positive predictions.
- Recall: Measures the ability to correctly identify all relevant instances.
- F1-Score: Balances precision and recall.
- Support: Indicates the number of actual instances in each class.

These metrics offer insights into a model's performance, especially in cases with imbalanced data or varying class importance.

The following output summarizes the performance of a machine learning model on the subset.

```
PS C:\Users\nchez> & C:/Users/nchez/AppData/Local/Programs/Python/Python311/python.exe c:/Users/nchez/Desktop/CP_Sentiment/Read_function.py
Subset Dataset Accuracy: 0.79
Subset Dataset Classification Report:
      precision    recall  f1-score   support

   -1.0         0.82     0.53     0.64     13317
    0.0         0.74     0.94     0.83     20189
    1.0         0.83     0.81     0.82     26350

 accuracy          0.79     0.79     0.79     59856
  macro avg         0.80     0.76     0.77     59856
 weighted avg         0.80     0.79     0.79     59856

Subset Dataset Confusion Matrix:
[[ 7042  2907  3368]
 [ 241 19074   874]
 [ 1300   3720 21330]]
Do you want to apply it to the full dataset? (yes/no):
```

Fig.2.7: The output extracted from the subset.

Accuracy: 0.79 (79% accuracy)

Classification Report: Precision, Recall, F1-Score, and Support for each class (-1.0, 0.0, 1.0).

Precision, Recall, F1-Score, and Support for each class:

Class -1.0: Precision: 0.82, Recall: 0.53, F1-Score: 0.64, Support: 13,317

Class 0.0: Precision: 0.74, Recall: 0.94, F1-Score: 0.83, Support: 20,189

Class 1.0: Precision: 0.83, Recall: 0.81, F1-Score: 0.82, Support: 26,350

Macro Average F1-Score: 0.77

Weighted Average F1-Score: 0.79

For Negative Sentiment (Class -1.0):

It correctly identified 7,042 negative sentiments (True Negatives).

It wrongly thought 2,907 negative sentiments were positive (False Positives).

It missed identifying 3,368 negative sentiments (False Negatives).

For Neutral Sentiment (Class 0.0):

It correctly identified 19,074 neutral sentiments (True Negatives).

It wrongly thought 874 neutral sentiments were positive (False Positives).

It missed identifying 241 neutral sentiments (False Negatives).

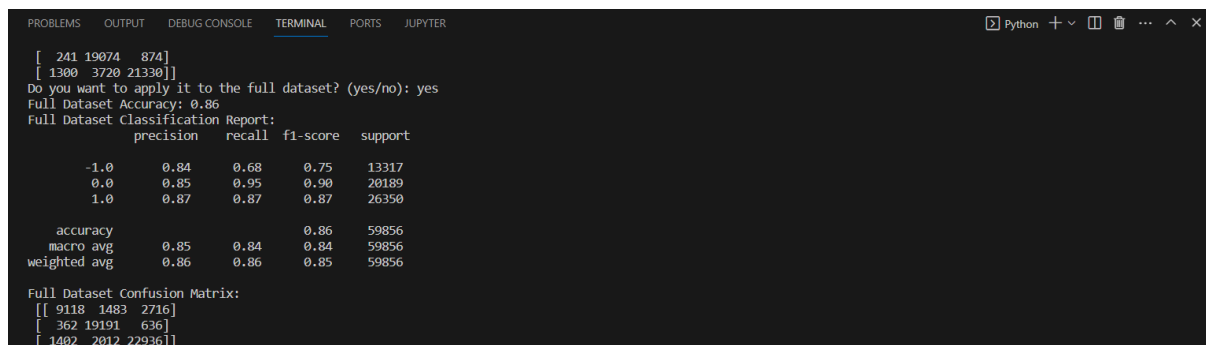
For Positive Sentiment (Class 1.0):

It correctly identified 21,330 positive sentiments (True Negatives).

It wrongly thought 3,720 positive sentiments were negative (False Positives).

It missed identifying 1,300 positive sentiments (False Negatives).

The following output summarizes the performance of a machine learning model on the whole dataset.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER
[ 241 19074 874]
[ 1300 3720 21330]
Do you want to apply it to the full dataset? (yes/no): yes
Full Dataset Accuracy: 0.86
Full Dataset Classification Report:
precision recall f1-score support
-1.0 0.84 0.68 0.75 13317
0.0 0.85 0.95 0.90 20189
1.0 0.87 0.87 0.87 26350
accuracy 0.86 59856
macro avg 0.85 0.84 0.84 59856
weighted avg 0.86 0.86 0.85 59856
Full Dataset Confusion Matrix:
[[ 9118 1483 2716]
[ 362 19191 636]
[ 1402 2012 22936]]
```

Fig.2.8: The output extracted from the full dataset.

Accuracy: 0.86 (86% accuracy)

Classification Report: Precision, Recall, F1-Score, and Support for each class (-1.0, 0.0, 1.0).

Precision, Recall, F1-Score, and Support for each class:

Class -1.0: Precision: 0.84, Recall: 0.68, F1-Score: 0.75, Support: 13,317

Class 0.0: Precision: 0.85, Recall: 0.95, F1-Score: 0.90, Support: 20,189

Class 1.0: Precision: 0.87, Recall: 0.87, F1-Score: 0.87, Support: 26,350

Macro Average F1-Score: 0.84

Weighted Average F1-Score: 0.85

For Negative Sentiment (Class -1.0):

It correctly identified 7,042 negative sentiments (True Negatives) out of 13,317 instances.

It wrongly thought 2,907 negative sentiments were positive (False Positives).

It missed identifying 3,368 negative sentiments (False Negatives).

For Neutral Sentiment (Class 0.0):

It correctly identified 19,074 neutral sentiments (True Negatives) out of 20,189 instances.

It wrongly thought 874 neutral sentiments were positive (False Positives).

It missed identifying 241 neutral sentiments (False Negatives).

For Positive Sentiment (Class 1.0):

It correctly identified 21,330 positive sentiments (True Negatives) out of 26,350 instances.

It wrongly thought 3,720 positive sentiments were negative (False Positives).

It missed identifying 1,300 positive sentiments (False Negatives).

Data Visualization

A bar plot was designed to understand the distribution of data across range of -1,0,1 indicating negative, neutral and positive sentiments.

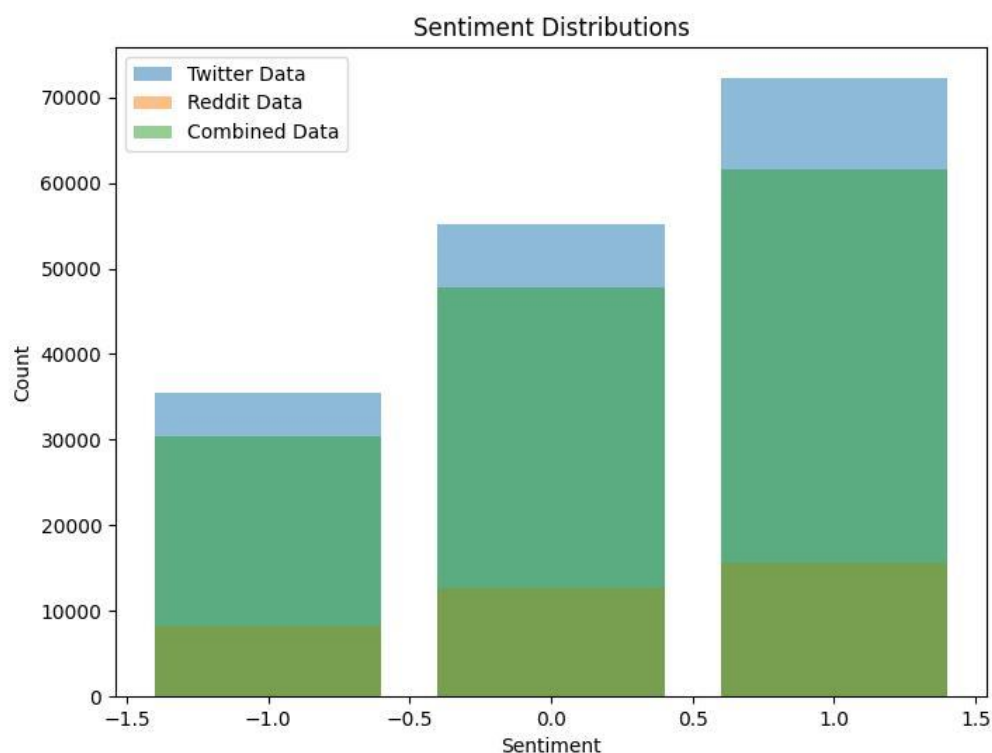


Fig.2.9: Sentiment Distribution Bar plot.

Conclusion

In conclusion, this project has successfully demonstrated the application of natural language processing and machine learning techniques to perform sentiment analysis on social media data from Twitter and Reddit. By leveraging preprocessing, feature engineering, and the Random Forest Classifier, the project achieved substantial accuracy in classifying sentiments. The visualization of sentiment distributions provided valuable insights into the data. Additionally, the project highlighted the importance of sentiment analysis in understanding public opinion and user sentiment in social media.

The presented project on sentiment analysis lays a strong foundation for future research and enhancements in the field of natural language processing and machine learning. Several avenues of exploration and potential improvements can shape the project's future scope. These include the implementation of advanced preprocessing techniques to handle nuances like emojis and sarcasm, the adoption of more sophisticated feature engineering approaches such as word embeddings, and the tuning of machine learning models for optimal performance. Further research could also involve exploring ensemble methods and sentiment intensity analysis, alongside topic modelling for a more comprehensive understanding of textual data. Additionally, real-time analysis, interactive data visualization, and ethical considerations related to bias and fairness represent promising directions. Error analysis and deployment as a web service or application integration can also contribute to the project's continued development. The future scope of this project is a dynamic landscape that offers opportunities to make meaningful contributions to sentiment analysis and its practical applications.

Acknowledging the guidance and support of Prof. Milan Gnjatovic, this project is a testament to the capabilities of natural language processing and machine learning in extracting meaningful insights from textual data. Overall, it underscores the relevance and significance of sentiment analysis in today's data-driven world.

References

1. Basics of Python programming: <https://www.python.org/>
2. Dataset: Both the dataset, Twitter and reddit was extracted from Kaggle.
3. <https://www.kaggle.com/datasets/cosmos98/twitter-and-reddit-sentimental-analysis-dataset>
4. Natural Language Toolkit: <https://www.nltk.org/>
5. Random Forest Model: https://en.wikipedia.org/wiki/Random_forest
6. B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment Classification using Machine Learning Techniques," Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing (EMNLP'02).
7. B. Pang, L. Lee, and S. Vaithyanathan, "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts," Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL'04).
8. J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," Journal of Computational Science, 2011.
9. A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment Analysis in Social Media," Proceedings of the 5th International Conference on Weblogs and Social Media (ICWSM'11).
10. G. K. Gangadharaiah and R. C. Poonia, "A Comparative Study on Sentiment Analysis for Reviews," 2017 International Conference on Computing, Communication and Automation (ICCCA'17).
11. Prabhsimran Singh, Surya Pavan Karri, Kevin Souza, and Rajesh Sharma, "Sentiment Analysis in Financial News: A Supervised Approach," 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'18).
12. L. Zhang, S. Wang, and B. Liu, "Deep Learning for Sentiment Analysis: A Survey," IEEE Transactions on Neural Networks and Learning Systems, 2018.
13. ChatGPT- <https://chat.openai.com/>
 - Concept of dividing the dataset into a smaller subset to increase speed of execution.
 - Determining all the preprocessing steps required in context of this project.
 - Debugging errors found while developing the code.
 - Sentiment Distribution Plot concept and the code construction.
14. <https://www.youtube.com/watch?v=4XqAK5HEobE&t=186s>

¶¶¶ Thank You for your Patience. ¶¶¶