SENTIMENT ANALYSIS FOR TWITTER DATASET

TEAM 3:

Pranshu Acharya (1)

Shruti Balaji (5)

Pawan Karthik (29)

Contents

Table of Figures

Problem Statement

Airline companies face the challenge of managing many customer comments and feedback. To address this, an effective sentiment analysis model is crucial. This model evaluates public sentiment and provides actionable insights to enhance operational efficiency and improve customer satisfaction. Ultimately, it helps companies stay ahead by understanding and responding to customer needs and preferences.

Introduction to Sentiment Analysis

Sentiment analysis, also called opinion mining, is a technique that uses natural language processing (NLP) to determine the emotional tone or sentiment expressed in a piece of text. In our case, we are utilizing tweets to comprehend emotions. The primary objective of sentiment analysis is to classify and understand the subjective opinions present in the text as positive, negative, or neutral. With the advent of social media, online reviews, and other user-generated content, sentiment analysis has become an indispensable tool for comprehending people's opinions effectively.

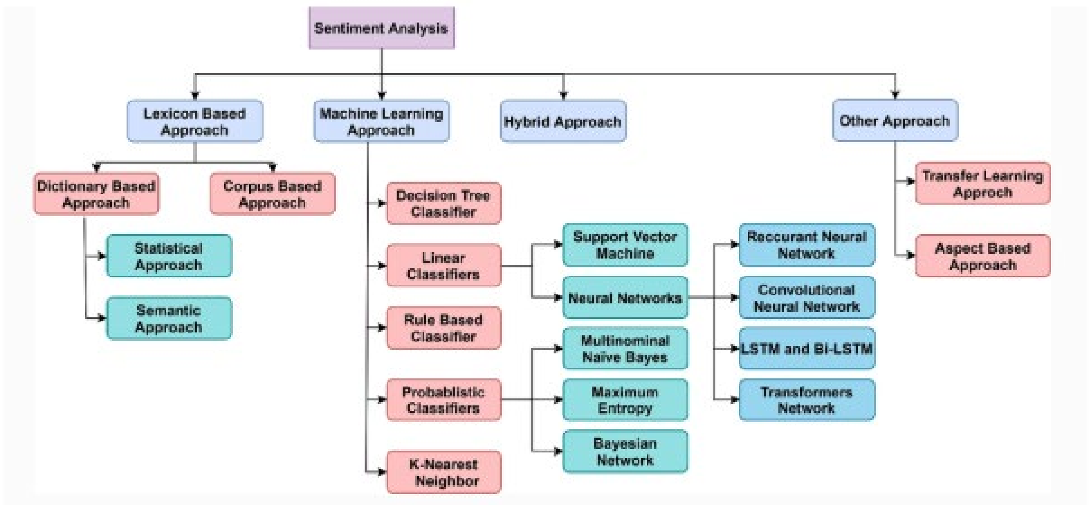


Figure 1: Sentiment Analysis Approaches

ML Pipeline

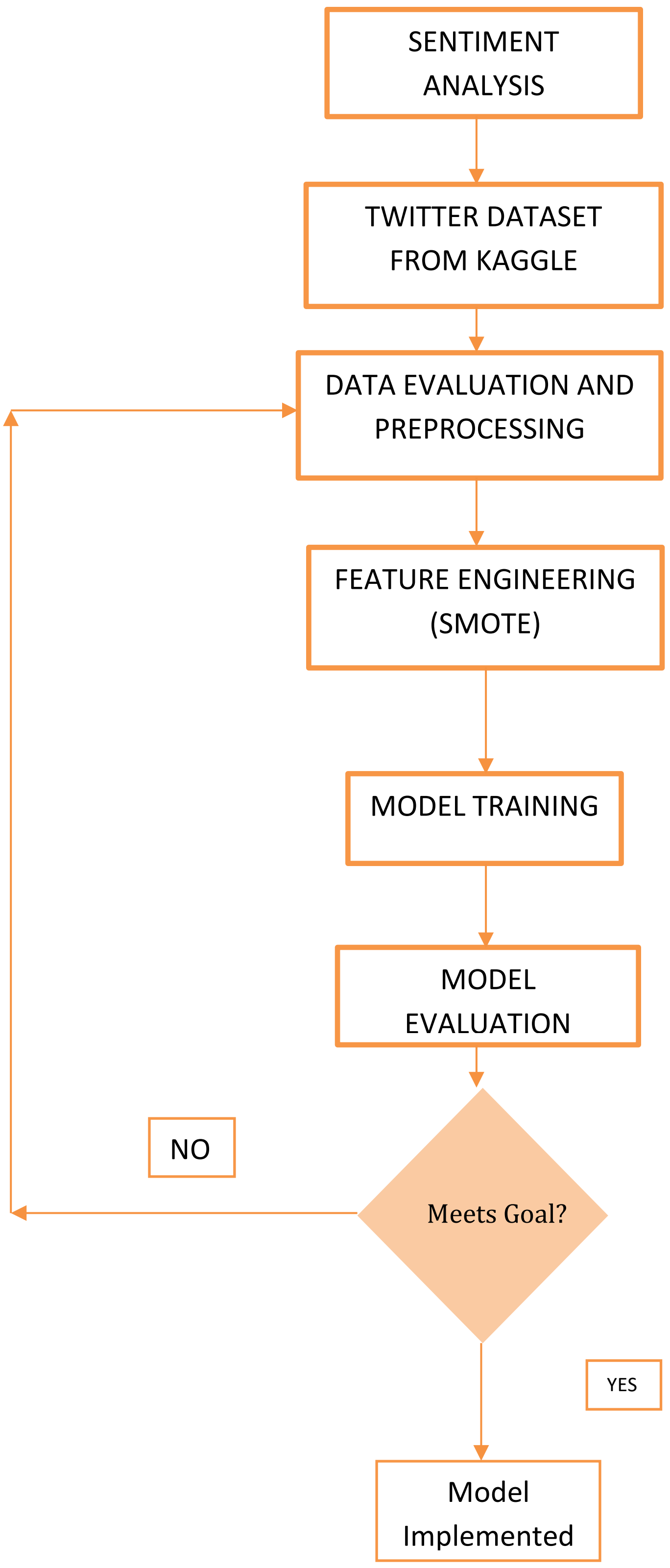

SENTIMENT ANALYSIS
TWITTER DATASET FROM KAGGLE
DATA EVALUATION AND PREPROCESSING
FEATURE ENGINEERING (SMOTE)
MODEL TRAINING
MODEL EVALUATION
Meets Goal?
NO
YES
Model Implemented

Figure 2: ML Pipeline

Data Collection

Source of Data:

- Twitter US Airline Sentiment dataset on Kaggle
- Tweets directed to major US airline companies (United, US Airways, Delta, etc.)

Collection Timeframe:

- Data scraped from Twitter in February 2015

Labeling:

- Contributors manually labeled tweets into three categories: Positive, Neutral, or Negative sentiment

Multi-Class Classification:

- Three sentiment categories: Positive, Neutral, and Negative
- Appropriate for multi-class classification algorithms

Data Columns:

It includes columns such as:

- tweet_id: Unique identifier for each tweet
- airline_sentiment: Sentiment label (Positive, Neutral, Negative)
- airline: Mentioned airline in the tweet (e.g., United, US Airways, Delta)
- text: Actual text content of the tweet

Usage:

- Suited for sentiment analysis tasks
- It helps in understanding public sentiment toward different airline companies

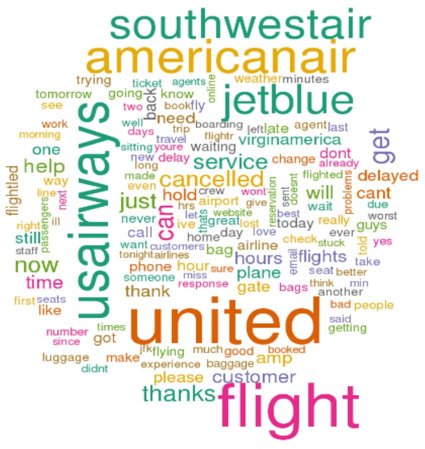
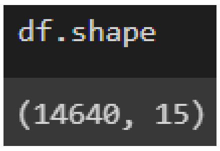


Figure 3: Word Cloud for the Dataset

Data Evaluation

The data contains 14640 rows, with 15 features from the code below.

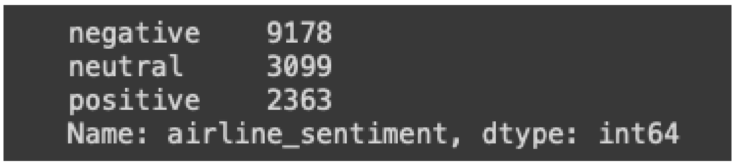


```
df.shape
```

```
(14640, 15)
```

Figure 4: Shape of the dataset

Among the target column, we find the distribution to be the following:



```
negative    9178
neutral     3099
positive    2363
Name: airline_sentiment, dtype: int64
```

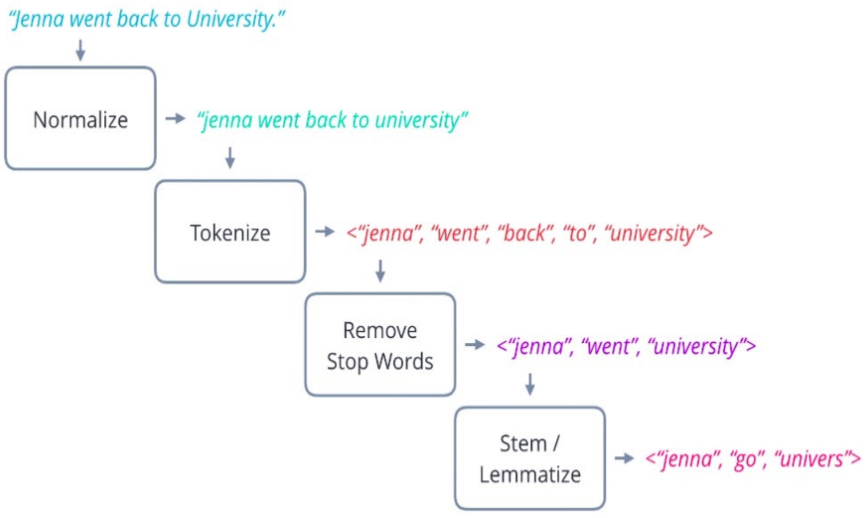
Figure 5: Target class distribution

We can see that the data is skewed negatively, i.e., there are more negative target classes than positive and neutral classes.

Data Preprocessing

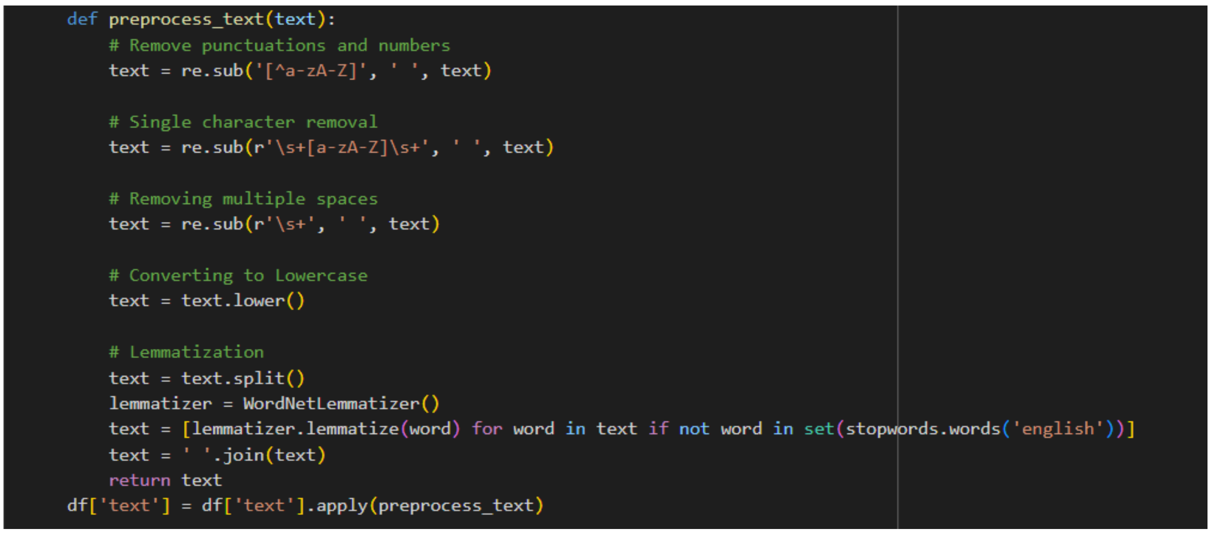
Several essential preprocessing and normalization techniques were applied in the initial phase of preparing our raw Twitter data for sentiment analysis with the NLTK library.

Firstly, the entire text corpus was transformed into lowercase to ensure uniformity and ease of analysis, mitigating variations arising from case differences. Subsequently, duplicate rows were systematically removed from the dataset to enhance its quality and eliminate redundancies. Handling missing values is a critical step in any data preprocessing pipeline, and in this context, any instances with missing information were dropped from consideration. To facilitate a more granular analysis, the tweets were tokenized into individual words, breaking down the text into its constituent elements. Stop words and common language terms with little analytical value were then systematically removed, streamlining the dataset further. Lastly, stemming was employed to reduce each token to its root form, aiding in standardization and capturing the core meaning of words. These meticulous steps collectively contributed to a refined and normalized dataset, laying a robust foundation for subsequent sentiment analysis tasks.



"Jenna went back to University."
Normalize
"jenna went back to university"
Tokenize
<"jenna", "went", "back", "to", "university">
Remove Stop Words
<"jenna", "went", "university">
Stem / Lemmatize
<"jenna", "go", "univers">

Figure 6: NLP Preprocessing steps



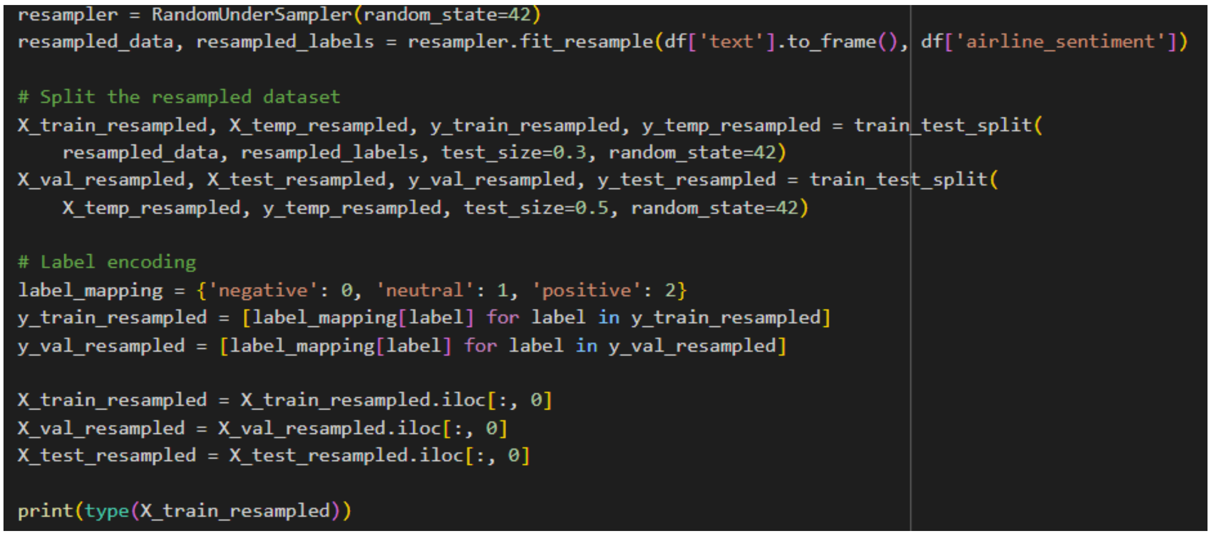
```
def preprocess_text(text):
    # Remove punctuations and numbers
    text = re.sub('[^a-zA-Z]', ' ', text)

    # Single character removal
    text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text)

    # Removing multiple spaces
    text = re.sub(r'\s+', ' ', text)

    # Converting to Lowercase
    text = text.lower()

    # Lemmatization
    text = text.split()
    lemmatizer = WordNetLemmatizer()
    text = [lemmatizer.lemmatize(word) for word in text if not word in set(stopwords.words('english'))]
    text = ' '.join(text)
    return text
df['text'] = df['text'].apply(preprocess_text)
```

Figure 7: NLP Preprocessing code

Feature Engineering

Sampling Using SMOTE

In addressing the class imbalance within our dataset, we implemented the Synthetic Minority Over-sampling Technique (SMOTE), an effective oversampling technique. The primary objective was to balance the representation of classes, which is particularly crucial given the bias towards the 'negative' sentiment category in our dataset. By generating synthetic instances of the minority class, SMOTE enhances the overall dataset's equilibrium.

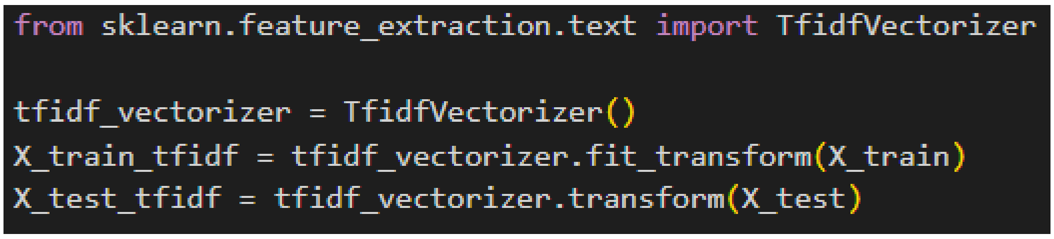


```
resampler = RandomUnderSampler(random_state=42)
resampled_data, resampled_labels = resampler.fit_resample(df['text'].to_frame(), df['airline_sentiment'])

# Split the resampled dataset
X_train_resampled, X_temp_resampled, y_train_resampled, y_temp_resampled = train_test_split(
    resampled_data, resampled_labels, test_size=0.3, random_state=42)
X_val_resampled, X_test_resampled, y_val_resampled, y_test_resampled = train_test_split(
    X_temp_resampled, y_temp_resampled, test_size=0.5, random_state=42)

# Label encoding
label_mapping = {'negative': 0, 'neutral': 1, 'positive': 2}
y_train_resampled = [label_mapping[label] for label in y_train_resampled]
y_val_resampled = [label_mapping[label] for label in y_val_resampled]

X_train_resampled = X_train_resampled.iloc[:, 0]
X_val_resampled = X_val_resampled.iloc[:, 0]
X_test_resampled = X_test_resampled.iloc[:, 0]

print(type(X_train_resampled))
```

Figure 8: SMOTE Code

TF-IDF Vectorization

We employed the TF-IDF (Term Frequency - Inverse Document Frequency) vectorization technique to extract meaningful features from the textual data. This involved two key components: TF (Term Frequency), representing the word count in each document, and IDF (Inverse Document Frequency), quantifying how commonly a word occurs across the entire dataset. By combining these components, we derived the importance or weight of each word, providing a nuanced understanding of its significance within the context of the dataset.



```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Figure 9: TF-IDF Code

One-Hot Encoding

For efficient encoding of the target column, we utilized one-hot encoding. This process involved adding three additional features, namely 'positive,' 'negative,' and 'neutral,' and encoding the target categories into binary values ('1' or '0'). This transformation not only facilitated machine learning model compatibility but also effectively represented the sentiment classes in a binary format, enhancing the overall model's interpretability and performance.

Models Used

Gradient Boosting:

Gradient Boosting is an ensemble learning technique that combines the predictions of multiple weak learners, usually decision trees, to create a robust predictive model. It works sequentially, with each new tree correcting the errors of the previous ones. In our sentiment analysis, Gradient Boosting was instrumental in capturing complex relationships within the tweet data, enhancing the overall predictive performance.

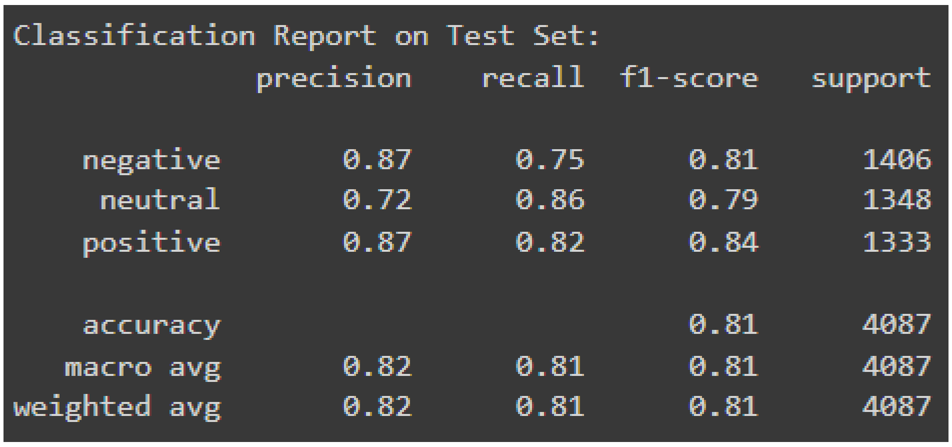
Accuracy and Hyper-parameters:

- Achieved an accuracy of 81.1353%.
- Hyper-parameters used include 200 estimators and a tree depth of 6.

Ensemble Strengths:

- The Gradient-boosting ensemble approach effectively captured complex relationships in the data.
- With 200 trees at a depth of 6, the boosting process demonstrated robust performance in sentiment classification.

Classification Report:



```
Classification Report on Test Set:
              precision    recall  f1-score   support

    negative       0.87      0.75      0.81      1406
     neutral       0.72      0.86      0.79      1348
    positive       0.87      0.82      0.84      1333

    accuracy                           0.81      4087
   macro avg       0.82      0.81      0.81      4087
weighted avg       0.82      0.81      0.81      4087
```

Figure 10: Classification Report of Gradient Boosting

XGBoost:

XGBoost, an extension of Gradient Boosting, is renowned for its efficiency and scalability. It incorporates regularization techniques and parallel processing, making it faster and more accurate. In our sentiment analysis, XGBoost was crucial in refining the model's accuracy and speed, providing a robust framework for classifying tweets into Positive, Neutral, or Negative sentiments.

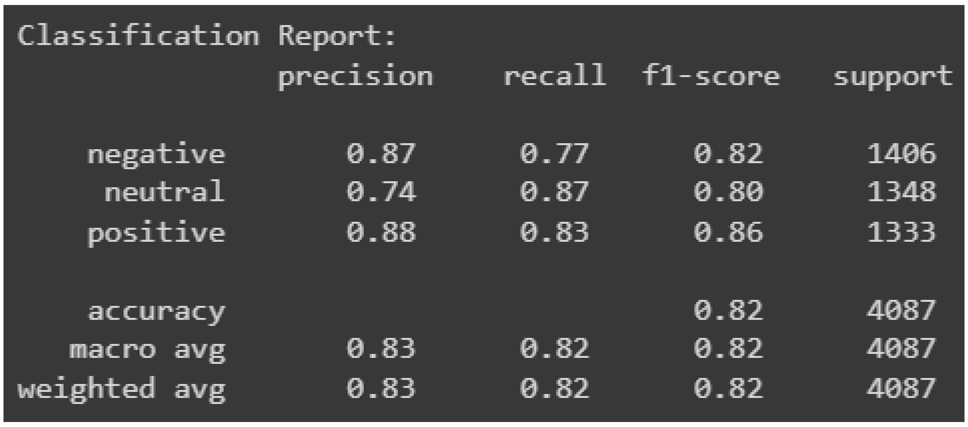
Advanced Features and Regularization:

- XGBoost incorporates advanced features, including L1 (Lasso) and L2 (Ridge) regularization techniques, enhancing model robustness.
- The built-in hyperparameter tuning capability and the ability to fill in missing values contribute to XGBoost's effectiveness in sentiment analysis.

Performance Metrics:

- We achieved an accuracy of 82.3343%, indicating the percentage of correctly classified sentiments.
- Using regularization techniques and advanced features in XGBoost contributes to its strong performance in handling nuanced language and complex sentiment expressions.

Classification Report:



```
Classification Report:
              precision    recall  f1-score   support

    negative       0.87      0.77      0.82      1406
     neutral       0.74      0.87      0.80      1348
    positive       0.88      0.83      0.86      1333

    accuracy                           0.82      4087
   macro avg       0.83      0.82      0.82      4087
weighted avg       0.83      0.82      0.82      4087
```

Figure 11: Classification Report of XGBoost

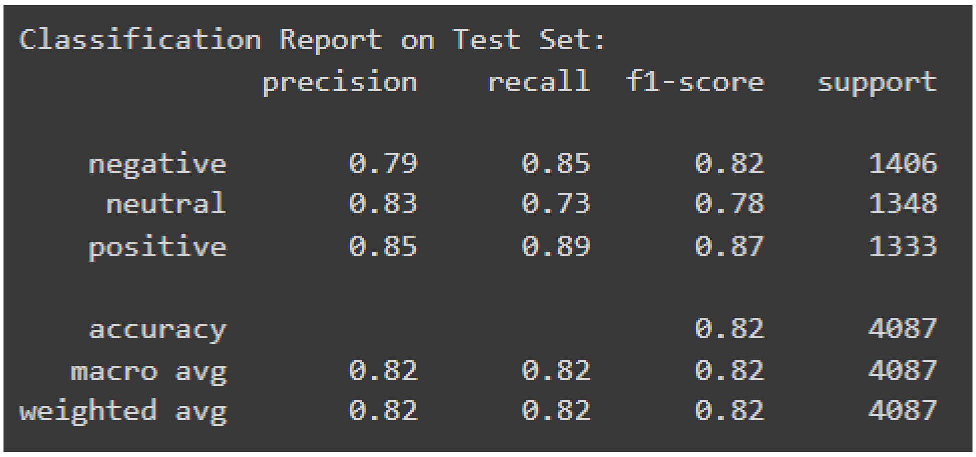
Multinomial Naive Bayes:

Multinomial Naive Bayes is a probabilistic classification algorithm particularly suited for text data. It's based on Bayes' theorem and assumes independence between features. In our sentiment analysis, MultiNomial Naive Bayes demonstrated effectiveness in handling the textual content of tweets, offering a simple yet powerful approach to categorizing sentiments based on the probabilistic nature of word occurrences.

Accuracy and Training Process:

- We achieved an accuracy of 82.1874%, indicating the model's ability to classify sentiments in the dataset accurately.
- Multinomial Naive Bayes is based on Bayes' theorem and the assumption that the features (words in this case) are independent. The training process involves providing labeled training data, enabling the classifier to learn associations between words and sentiments.

Classification Report



```
Classification Report on Test Set:
              precision    recall  f1-score   support

    negative       0.79      0.85      0.82      1406
     neutral       0.83      0.73      0.78      1348
    positive       0.85      0.89      0.87      1333

    accuracy                           0.82      4087
   macro avg       0.82      0.82      0.82      4087
weighted avg       0.82      0.82      0.82      4087
```

Figure 12: Classification Report of Naive Bayes

Support Vector Machine (SVM):

Support Vector Machine is a robust classification algorithm that aims to find the optimal hyperplane to separate different classes in a high-dimensional space. In our sentiment analysis, SVM proved valuable in identifying intricate decision boundaries within the tweet data, allowing for effective classification of sentiments, especially in cases where sentiments are complexly intertwined.

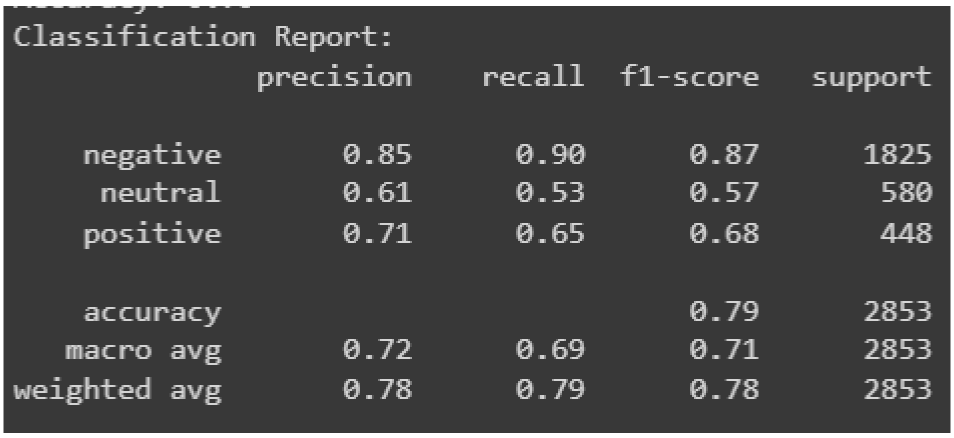
Accuracy and Precision-Recall:

- Tuned SVM achieved 79% accuracy, with solid precision (0.85) and recall (0.90) for negative sentiment.
- Positive and neutral classes had lower scores, indicating challenges in classification.

Challenges and Visualization:

- Misclassification of neutral tweets was a prominent issue.
- SVM's decision boundaries were well-plotted, showing a clear separation of sentiment classes.

Classification Report:



```
[illegible]
Classification Report:
              precision    recall  f1-score   support

    negative       0.85      0.90      0.87      1825
     neutral       0.61      0.53      0.57       580
    positive       0.71      0.65      0.68       448

    accuracy                           0.79      2853
   macro avg       0.72      0.69      0.71      2853
weighted avg       0.78      0.79      0.78      2853
```

Figure 13: Classification Report of SVM

Neural Network:

Neural Networks, particularly deep learning models, are capable of automatically learning intricate patterns and representations from data. In our sentiment analysis, a Neural Network provided a deep learning approach to understanding the nuanced language in tweets. It allowed the model to capture complex relationships and dependencies, enhancing the overall accuracy of sentiment classification.

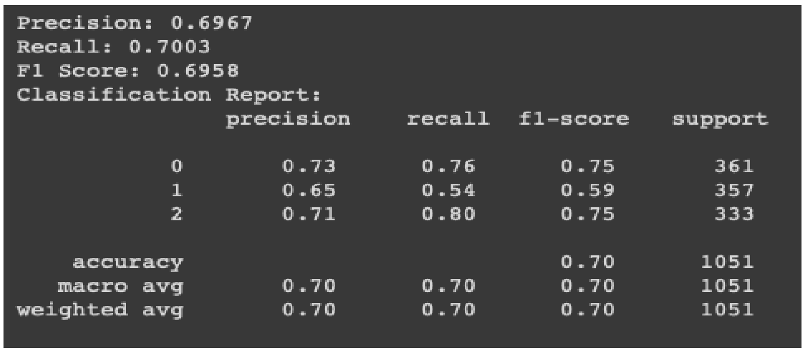
Network Architecture and Hyperparameters:

- We implemented a feedforward neural network using Keras for sentiment analysis.
- We utilized a simple one-way architecture with three hidden layers (16, 32, 16 neurons) and softmax activation for multi-class classification.
- Employed ReLU activation for hidden layers, 'categorical_crossentropy' loss, 'adam' optimizer, and 'accuracy' metric.
- Trained for 20 epochs with a vocabulary limit of 6000 words, implementing early stopping to prevent overfitting.

Performance Highlights:

- The neural network effectively handled sentiment analysis, showing a clear separation of classes.
- Validation data usage for each epoch, along with early stopping, contributed to generalization and prevented overfitting.
- Hyperparameters like 'adam' optimizer and 'categorical_crossentropy' loss were vital in achieving accuracy.

Classification Report



```
Precision: 0.6967
Recall: 0.7003
F1 Score: 0.6958
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.76      0.75       361
           1       0.65      0.54      0.59       357
           2       0.71      0.80      0.75       333

    accuracy                           0.70      1051
   macro avg       0.70      0.70      0.70      1051
weighted avg       0.70      0.70      0.70      1051
```

Figure 14: Classification Report of Neural Network

Results

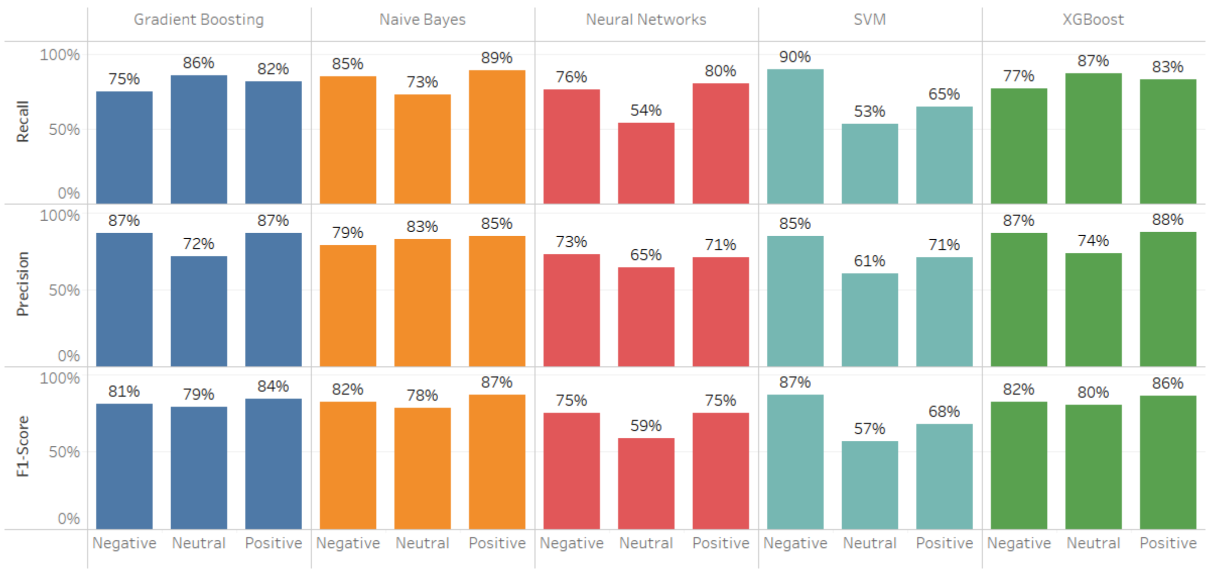


Figure 15: Comparison of each Model

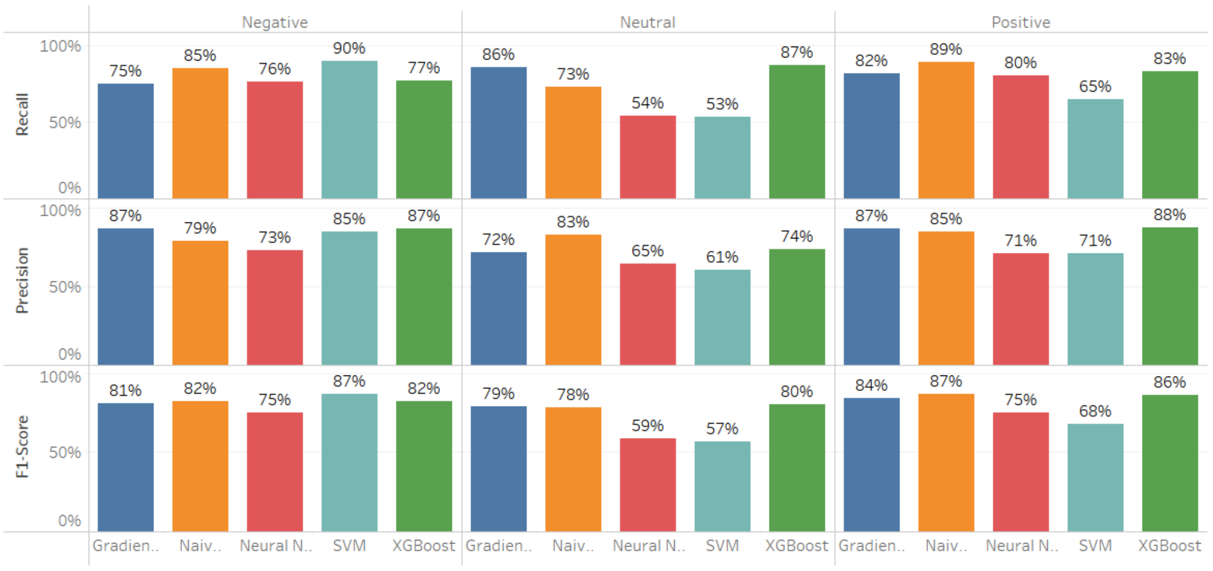


Figure 16: Comparison of each class

Conclusion

In wrapping up our Tweet Sentiment Analysis project, we used different models like SVM, XGBoost, a FeedForward neural network, Gradient Boosting, and Naive Bayes. We clearly see that XGBoost outperforms Gradient Boosting in all metrics since it improves GBM. Among all the models and algorithms, we found that XGBoost and Naive Bayes performed the best overall.

If we go into specifics, for the negative classification, naive Bayes dominates XGBoost regarding recall, but the opposite happens in the precision metric. So, the use of the model comes down to what the business scenario demands and what the user is looking for, whether it be precision or recall.

To sum it up, our project taught us a lot about sentiment analysis and showed us how important it is to pick the right tool based on the data. We learned that diversity in models is vital, and this will help us make better choices in future projects when dealing with the ever-changing world of social media data.

References

SMOTE **|** Saturn Cloud. https://saturncloud.io/glossary/smote/

Kaggle **|** https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment

“XGBoost Algorithm - Amazon SageMaker.” Docs.aws.amazon.com, docs.aws.amazon.com/sagemaker/latest/dg/xgboost.html.

Bebis, George, and Michael Georgiopoulos. "Feed-forward neural networks." Ieee Potentials 13.4 (1994): 27-31.

Ahmad, Munir, et al. "Sentiment analysis using SVM: a systematic literature review." International Journal of Advanced Computer Science and Applications 9.2 (2018).