MORINGA SCHOOL

DATA SCIENCE

TWITTER SENTIMENT ANALYSIS PROJECT REPORT

AUGUSTINE KOMEN

FIONA AMUGUNE

ELIZABETH ATIENO

NEEMA KEZIA

PETER RONOH

JANUARY 2025

Contents

1. PROJECT OVERVIEW

Best Buy, a leading retailer of iPhone and Google products, aims to improve its inventory choices by analyzing user opinions on these items. This project utilizes sentiment analysis of Twitter data to understand customer views and preferences regarding iPhone and Google products. By examining user sentiments, Best Buy intends to make stocking decisions that match customer demand and boost overall satisfaction. The project involves gathering Twitter data on discussions, reviews, and mentions of iPhone and Google products, followed by data preprocessing and sentiment analysis. The insights gained will help Best Buy optimize its product selection and align it with customer preferences.

2. BUSINESS UNDERSTANDING

2.1. Business Problem

Best Buy, a top reseller of iPhone and Google products, encounters difficulties in aligning its inventory with customer preferences. The lack of a structured approach to analyzing user sentiments on Twitter hinders data-driven stocking decisions. This project aims to utilize Twitter sentiment analysis to better understand customer opinions. By gaining insights into user sentiments, Best Buy aims to enhance its stocking decisions, ensuring product availability that matches customer preferences and improves overall customer satisfaction and loyalty.

2.2. Objectives

The goal of this project is to analyze customer sentiments on Twitter to inform Best Buy's stocking decisions for iPhone and Google products. By analyzing customer opinions, the aim is to improve resource allocation, optimize product assortment, enhance customer satisfaction, and ultimately increase profitability.

2.3. Business Questions

1. What are the predominant sentiments expressed by customers on Twitter regarding iPhone and Google products?
2. What factors influence the polarity of tweets related to iPhone and Google products on Twitter?
3. Which specific features of iPhone and Google products are most frequently praised or criticized by users on Twitter?

4. Which machine-learning model is most effective in sentiment analysis?

2.4. Metric of success

- Accuracy Score: ≥ 80%

3. DATA UNDERSTANDING

Source of Data: CrowdFlower via data.world

Data Description: The dataset contains 9093 rows and 3 columns, which are all categorical.

The columns include:

- tweet: anonymized tweets containing truncated hyperlinks
- brand: a label indicating whether the tweet contains a sentiment towards a product or company in particular
    - Apple
    - Apple products
    - Google
    - Android
    - None
- emotion: an assigned label indicating the sentiment of the tweet
    - Positive
    - Neutral
    - Negative
    - Cant tell

4. DATA PREPARATION & ANALYSIS

4.1. Data Preparation

Checks Performed:

- Dropped one missing values in the tweet column.

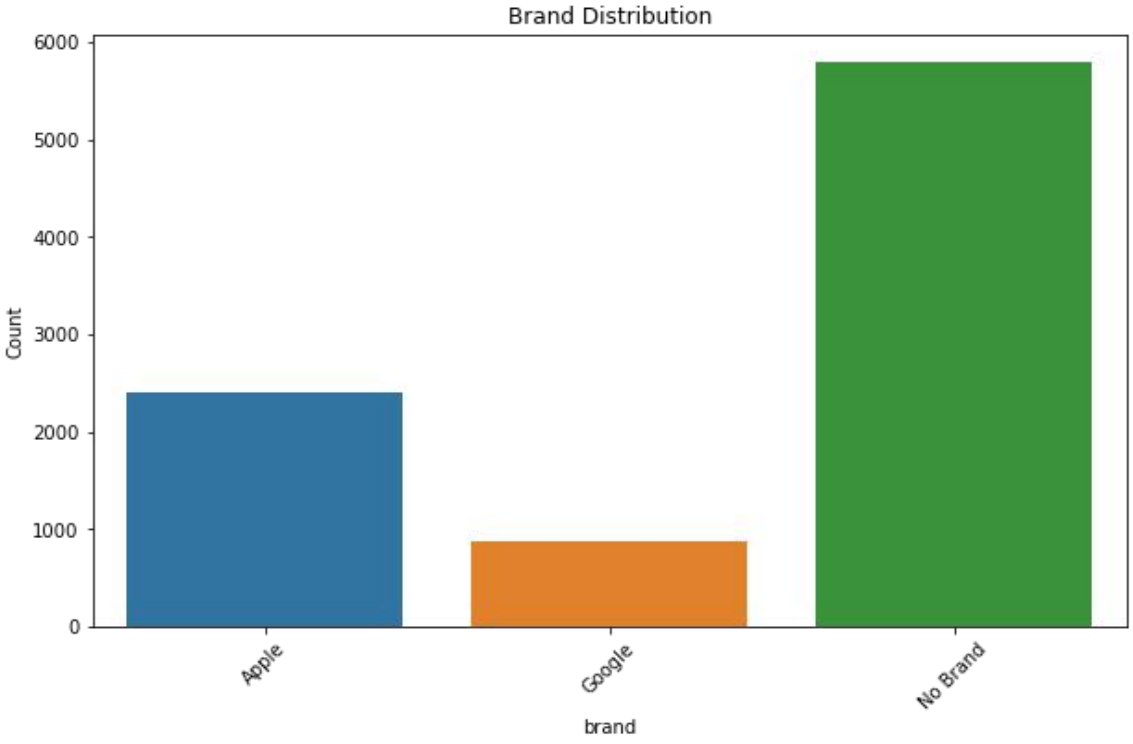
- Dropped duplicate rows.

4.2. Data Analysis

4.2.1 Univariate Analysis

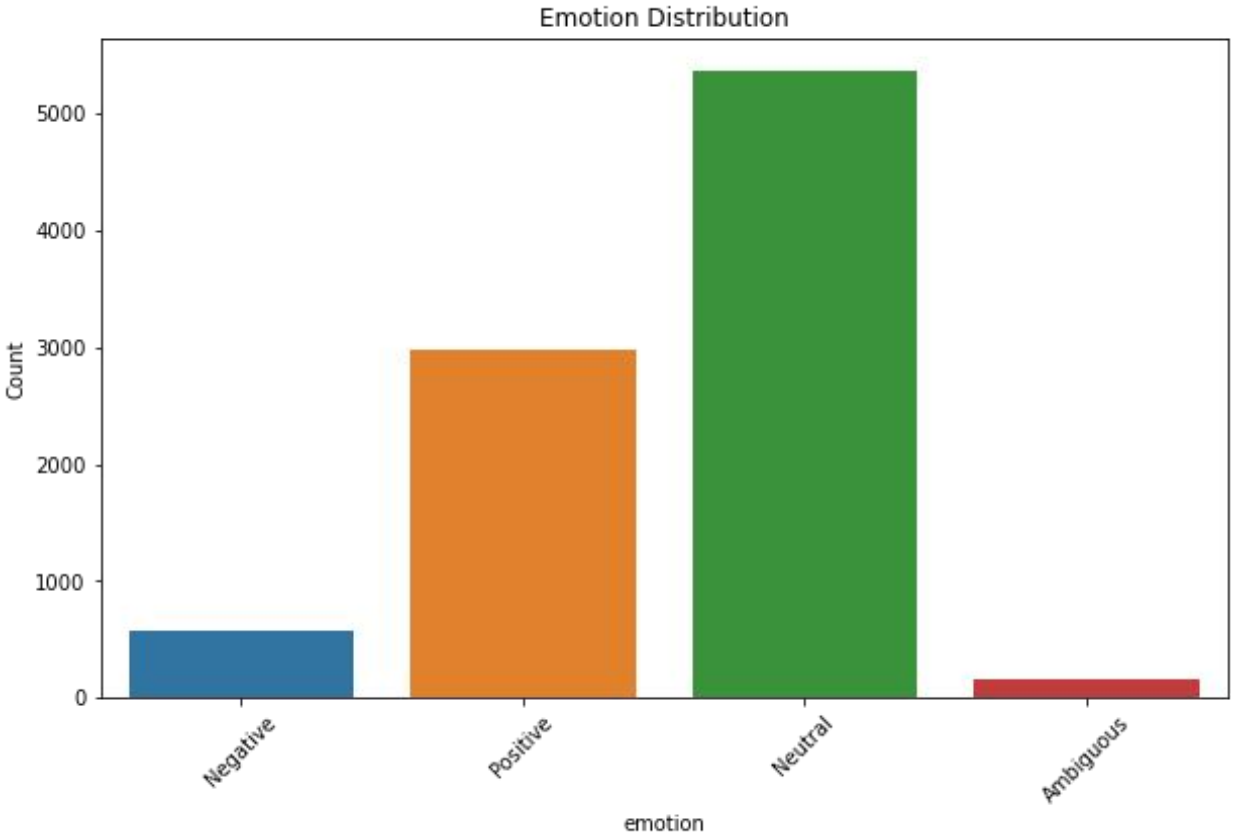
In this part, we analyzed distribution of brands and emotions in tweets, tweet length distribution, brand value counts and emotional counts. This part involved analyzing each variable independently.

- We first analyzed the brand distribution graph whereby the "No Brand" category has the highest count, close to 6000. This indicates that a significant portion of the dataset consists of unbranded products. The "Apple" brand has the second-highest count, slightly above 2000. This suggests a strong presence and popularity of Apple products within the dataset. The "Google" brand has the lowest count, slightly above 1000. While it has a notable presence, it is less prevalent compared to Apple and the "No Brand" category.

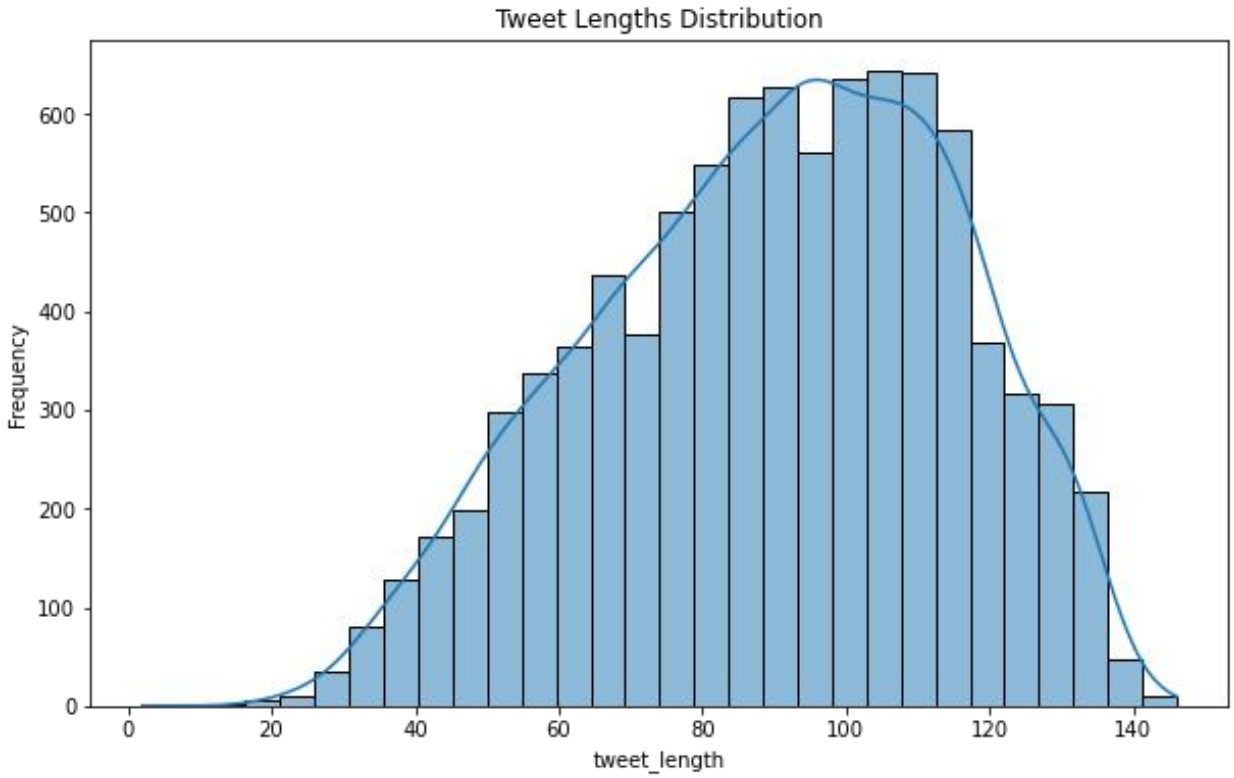


- Then we analyzed emotion distribution whereby it was found that the majority of the data is categorized as `"Neutral,"` followed by `"Positive"` emotions. There is a significantly lower count of `"Negative"` and `"Ambiguous"` emotions. Most tweets exhibit neutral

sentiment, indicating that the dataset may have a majority of tweets with objective or balanced tones.



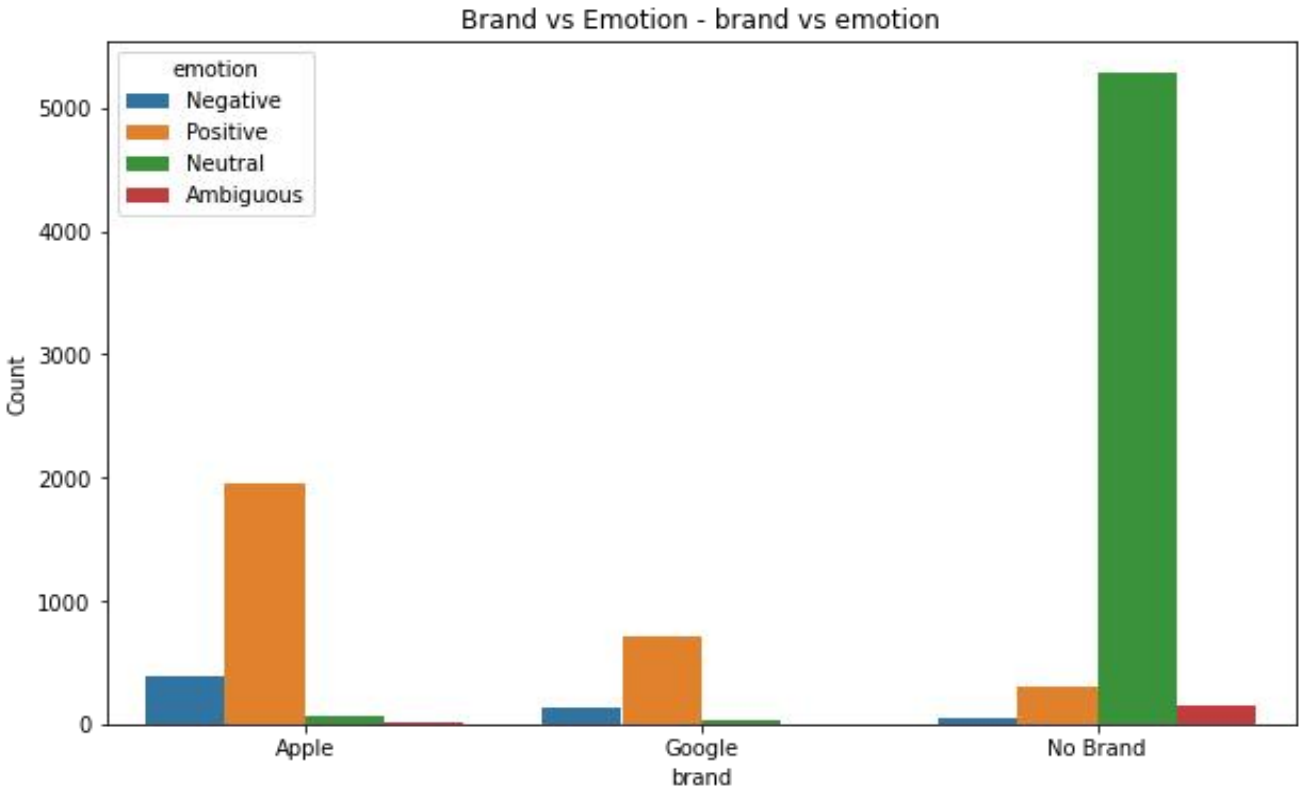
- We also analyzed tweet lengths where it was observed that the distribution of tweet lengths is approximately normal, with the majority of tweets falling between 60 and 100 characters.



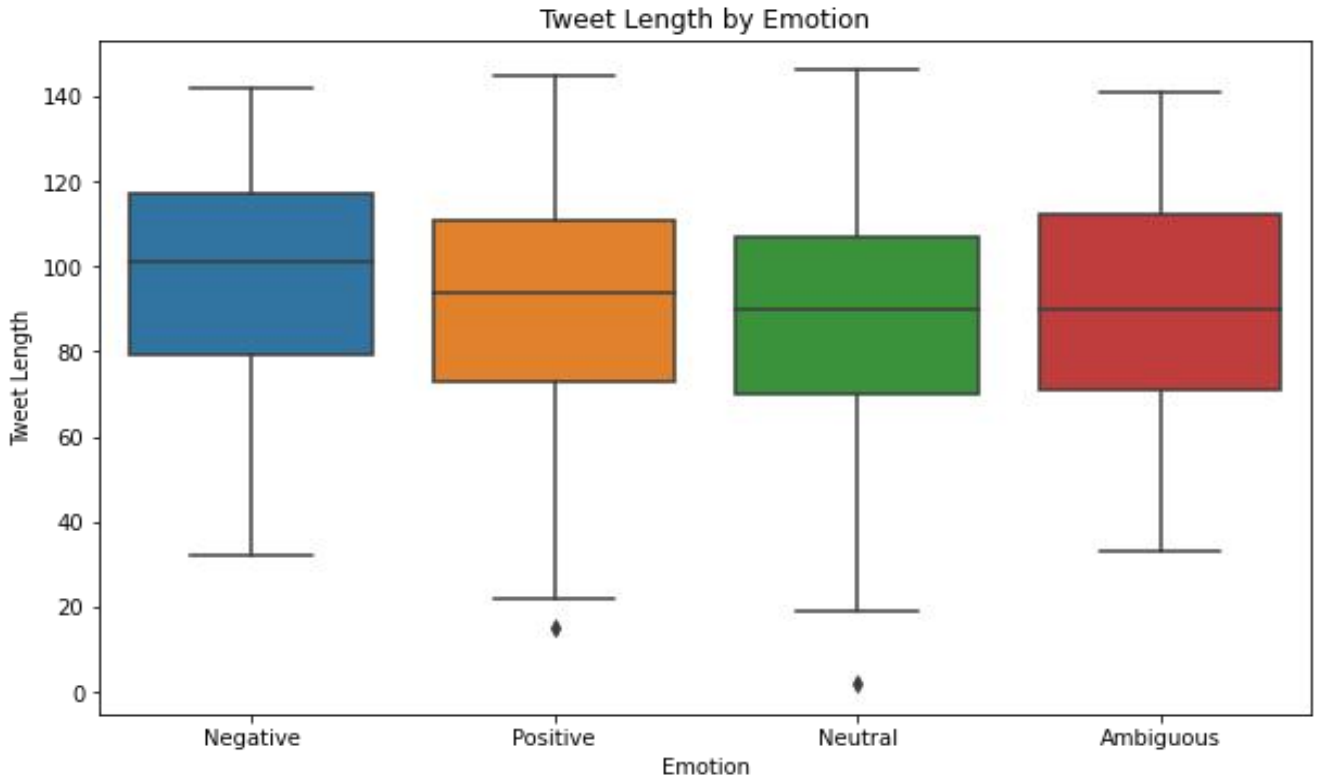
4.2.2 Bivariate Analysis

In this part, we wrote a function **myf.run_bivariate_analysis(df)** which performs a bivariate analysis on the dataframe **df**. This typically involves examining the relationships between two variables at a time.

- The first bivariate analysis that we did was analyzing **brand vs emotion**. The distribution of emotions across brands provides insights into how people feel about them. For Apple and Google, Positive emotions are predominant, indicating favorable sentiment. The high count of Neutral emotions for the No Brand category might suggest ambivalence or non-specific sentiment.



- The other bivariate analysis that we did was **Tweet length vs emotion**. The distribution of tweet lengths was across four different emotion categories: Negative, Positive, Neutral, and Ambiguous. Each box plot displays the range of tweet lengths, including the median, quartiles, and any potential outliers. The median tweet length is similar across all four emotion categories, indicating that the central tendency of tweet lengths does not vary significantly with the emotional content.



Median Tweet Length: The median tweet length is similar across all the four emotion categories, indicating that the central tendency of tweet lengths does not vary significantly with the emotional content.

Variability: All the four emotion categories exhibit a wide range of tweet lengths, with lengths spanning from very short to the maximum character limit of 140.

Outliers: Outliers are present in all emotion categories, suggesting that there are tweets with lengths that are significantly different from the majority. These could be extremely long or short tweets.

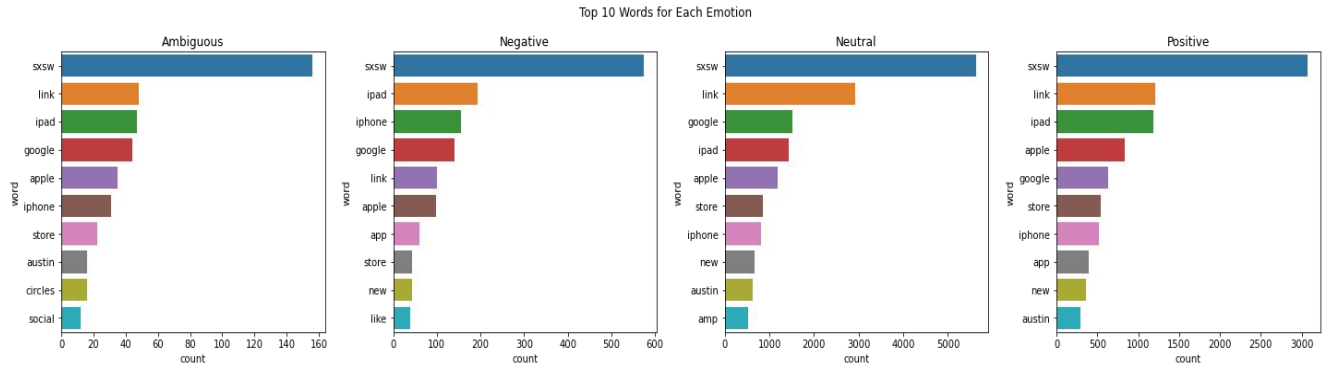
5. PREPROCESSING BEFORE MODELLING

This section outlines our preprocessing to prepare for modeling. We located frequent words to remove that are irrelevant to the sentiment of the tweet.

The preprocessing part takes in a document or tweet and does the following:

- Creates word tokens
- Stems and Lemmatizes
- Removes stop words
- Gets the parts of speech to lemmatize
- The preprocessing step returns the cleaned joined tweet.

We visualized the top 10 words for each emotion and it was noted that word frequencies in the Neutral category were significantly higher than in other categories, suggesting that the dataset is skewed toward neutral or factual sentiment rather than strong opinions. Also the prominence of terms like "sxsw," "apple," "google," and "iPhone" across categories indicates that the dataset is heavily focused on technology-related discussions, possibly centered on an event (likely SXSW).



6. MODELLING

FEATURE ENGINEERING.

In this step, we began by filtering the dataset to focus solely on the positive and negative sentiments expressed in the tweets. The original dataset, df, contained various sentiments, but for our analysis, we narrowed it down using the following line of code: binary_df = df[df['emotion'].isin(['Positive', 'Negative'])].copy(). This filtering resulted in a new DataFrame,

binary_df, which contains only the tweets labeled as "Positive" or "Negative." This is crucial for binary classification tasks, as it simplifies the problem to two classes. Next, we prepared our target variable, y_binary. We assigned the value of 1 to tweets labeled as "Positive" and 0 to those labeled as "Negative."

tweet: The original text of the tweet.
brand: The brand associated with the tweet (e.g., Apple or Google).
emotion: The sentiment label (either Positive or Negative).
tweet_length: The length of the tweet in characters.
word_tokens: A list of words in the tweet.
stemmed: A list of stemmed words from the tweet.
lemmatized: A list of lemmatized words from the tweet.
clean_tokens: A list of cleaned tokens.
processed_text: The processed text after cleaning.
word_freq: A dictionary representing the frequency of each word in the tweet.
y_binary: The binary target variable (1 for Positive, 0 for Negative).

The binary_df DataFrame, with its 3,539 rows and 11 columns, is now ready for further analysis and modeling. It provides a structured format to proceed with machine learning tasks aimed at sentiment classification.

In the modeling phase, we prepared the data for machine learning by first importing the necessary libraries required for our analysis. The libraries included:

TfidfVectorizer: This is used for converting the text data into a matrix of TF-IDF features, which helps in understanding the importance of words in the corpus.

train_test_split: This function is used to split the dataset into training and testing sets, allowing us to evaluate the model's performance on unseen data.

MultinomialNB: This is the Naive Bayes classifier that we will use for our sentiment classification task.

accuracy_score and **classification_report**: These are metrics for evaluating the model's performance.

To ensure that our model can generalize well, we split the data into training and testing sets.
X_train and **y_train**: These consist of 80% of the original data and will be used to train the model.

X_test and **y_test**: These contain the remaining 20% of the data, which will be used to evaluate the model's performance after training. This approach allows us to build a robust model capable of predicting tweet sentiments based on the learned patterns from the training data. The next steps would involve training the Naive Bayes classifier and evaluating its performance on the test set.

Baseline model (Multinomial Naive Bayes)

In this step, we established a baseline model using the Multinomial Naive Bayes algorithm to classify tweet sentiments. We initialized the TfidfVectorizer to convert the text data into numerical format and the MultinomialNB model for classification.
The training data (X_train) was vectorized using fit_transform, creating a matrix of TF-IDF features. The test data (X_test) was vectorized using transform, ensuring consistency in feature representation.

We fitted the Naive Bayes model on the vectorized training data. The model made predictions on the test data. We calculated the accuracy and generated a classification report, which summarized the model's performance. The accuracy was **0.86**, indicating the model correctly classified 86% of the test tweets.

Hyperparameter Tuning with GridSearchCV

In this step, we performed hyperparameter tuning on the baseline model using GridSearchCV to enhance its performance. We defined a grid of hyperparameters to optimize:

alpha: Smoothing parameter for the Naive Bayes model, tested with values [0.1, 0.5, 1.0, 1.5, 2.0].

fit_prior: Indicator for whether to learn class prior probabilities, with options [True, False]. We initialized GridSearchCV with the Multinomial Naive Bayes model, specifying accuracy as the scoring metric and using 5-fold cross-validation. We fitted the grid search to the vectorized training data. We retrieved the best model and made predictions on the test data.

The accuracy of the best model was **0.90**, indicating improved performance. The classification report showed:

For Negative Sentiment (0)

Precision: 0.83
Recall: 0.41
F1-Score: 0.54

For Positive Sentiment (1)

Precision: 0.90
Recall: 0.99
F1-Score: 0.94

Overall Metrics:

Accuracy: 0.90

Best Parameters: The optimal hyperparameters found were:**alpha**: 0.1,**fit_prior**: True

Multinomial Naive Bayes with SMOTE

In this step, we applied Synthetic Minority Over-sampling Technique (SMOTE) to balance the training dataset, aiming to improve model performance. We used SMOTE to generate synthetic samples for the minority class in the training data. We initialized the Multinomial Naive Bayes model and fitted it to the balanced training data.

We made predictions on the original test data. The accuracy of this model was **0.85**, slightly lower than the baseline model's 0.86 and the tuned model's 0.90. The classification report showed:

For Negative Sentiment (0):

Precision: 0.49
Recall: 0.77
F1-Score: 0.60

For Positive Sentiment (1):

Precision: 0.96
Recall: 0.86
F1-Score: 0.90

Overall Metrics:

Accuracy: 0.85

While SMOTE aimed to balance the classes and potentially improve model performance, it resulted in a slight accuracy drop to 85%. The precision for the negative class was particularly low, indicating challenges in correctly identifying negative sentiments, despite strong performance on positive sentiments.

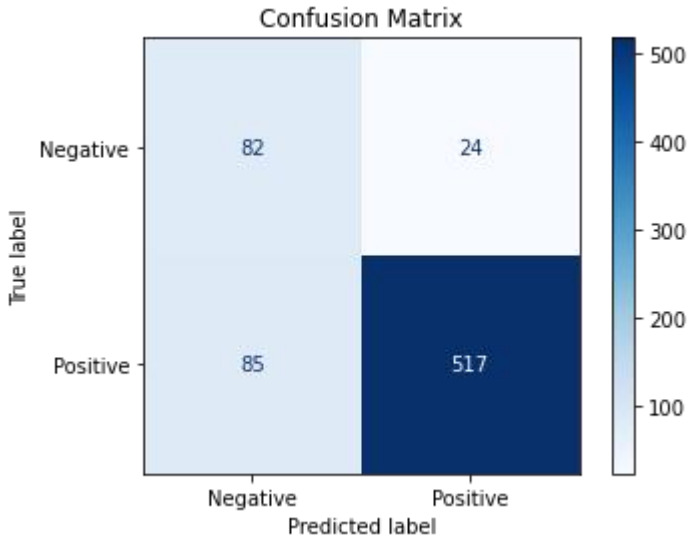
Confusion Matrix Visualization

To further evaluate the model's performance, we visualized the confusion matrix, which provides valuable insights into how well the model classifies each sentiment.

Interpretation:

High True Positives (TP): The model effectively identifies positive sentiments, with 517 correct predictions.

Moderate True Negatives (TN): While it identifies some negative sentiments, 82 correct predictions indicate room for improvement.
False Negatives (FN): The model missed 85 positive sentiments, suggesting it could improve in recognizing positive emotions.
False Positives (FP): The 24 false positives indicate a few negative tweets were incorrectly labeled as positive.



Neural Network

In this section, we trained a neural network for sentiment classification and evaluated its performance. The model showed excellent performance during training:

Epoch 1:
Loss: 0.5477
Accuracy: 70.96%
Validation Accuracy: 87.03%

Epoch 10:
Loss: 0.0059
Accuracy: 99.79%
Validation Accuracy: 99.68%

Evaluation Results:

Test Loss: 0.5176
Test Accuracy: 89.41%

Insights: The model achieved near-perfect accuracy on the training and validation datasets, indicating it learned the training data effectively.

The high validation accuracy (99.68%) suggests the model performs well on unseen data during training, but this also raises concerns about potential overfitting.

The test accuracy of 89.41% is strong but lower than the validation accuracy, confirming some degree of overfitting. This discrepancy indicates that the model may struggle to generalize to completely unseen data.

Fine-Tuning the Neural Network with GridSearchCV

In this section, we fine-tuned the neural network using GridSearchCV to optimize hyperparameters and improve performance. After fitting the grid search, we obtained the best parameters and evaluated the model.

Best Parameters: {'learning_rate': 0.001, 'num_neurons': 32}
Best Cross-Validation Score: 0.96

We evaluated the best model on the test set, yielding:

Test Loss: 0.48
Test Accuracy: 0.90

Original Test Accuracy: 89%
Fine-Tuned Test Accuracy: 90%

The fine-tuning process using GridSearchCV resulted in a slight improvement in test accuracy, showcasing a more optimized model. This highlights the effectiveness of hyperparameter tuning in enhancing model performance, making it more robust for sentiment classification tasks.

Multi-Class Classification with Naive Bayes

In this section, we performed multi-class classification on a dataset of emotions using the Multinomial Naive Bayes model.

Label Encoding: We encoded the emotion column into numerical values using LabelEncoder. The classes mapped as follows:

0: Ambiguous
1: Negative
2: Neutral
3: Positive

We prepared the features and labels, then split the dataset into training and test sets.We trained a Multinomial Naive Bayes model on the vectorized training data.After making predictions on the test set, we evaluated the model:

Accuracy: 0.67

Insights:

Class 0 (Ambiguous): Precision, recall, and F1-score are all very low, indicating poor model performance for this class.

Class 1 (Negative): High precision but very low recall and F1-score suggest that while the model is good at identifying negative sentiments when it predicts them, it misses many actual negative cases.

Class 2 (Neutral): This class performed well, with high precision, recall, and F1-score, indicating effective classification.

Class 3 (Positive): Moderate performance, with lower recall and F1-score compared to neutral sentiments, suggesting room for improvement.

Hyperparameter Tuning with GridSearchCV

In this section, we applied hyperparameter tuning to the Multinomial Naive Bayes model using GridSearchCV to improve performance on the multi-class classification task. We obtained the following results:

Best Parameters Found: {'alpha': 1.0}
Best Cross-Validated Accuracy: 0.64
Cross-Validation Scores: [0.623, 0.649, 0.634, 0.637, 0.635]
Mean CV Score: 0.64

Analysis of Results:

Model Performance: The Multinomial Naive Bayes model with hyperparameter tuning achieved a mean cross-validated accuracy of **64%**, which is a slight decrease from the baseline model's accuracy of **67%**.

Class Performance:

The model performs well for the **Neutral** emotion class but struggles with the **Ambiguous** and **Negative** classes. This indicates challenges in classifying minority classes effectively.

Hyperparameter Tuning: The optimal alpha value of **1.0** provided a marginal improvement in cross-validated accuracy, but overall, it did not significantly enhance model performance.

Generalization: The consistent cross-validation scores show that while the model demonstrates stable performance, there are clear areas for improvement in handling minority classes.

1.

Random Forest Classifier

In this section, we trained a Random Forest model for multi-class classification and evaluated its performance, both with default parameters and after hyperparameter tuning using Grid Search.

Default Random Forest Model

We trained the Random Forest model with default parameters.We made predictions on the test set and evaluated the model:

Default Random Forest Accuracy: **68%**

Classification Report:The model performed well for **Neutral** (Class 2) instances, with a high recall of 85%.
However, it struggled with the **Ambiguous** (Class 0) and **Negative** (Class 1) classes, indicating challenges with minority class predictions.

Fine-Tuning the Random Forest Model

We defined a parameter grid for tuning and used GridSearchCV. We stored the best model and displayed the optimal parameters.

Best Parameters: {'max_depth': None, 'min_samples_split': 10, 'n_estimators': 100}

Best Cross-Validated Accuracy: **66%**

Summary of Findings:

Default Model Performance: The default Random Forest model achieved an accuracy of **68%**.
Fine-Tuning Outcome: After hyperparameter tuning, the best model's cross-validated accuracy dropped to **66%**, indicating a slight decline in performance.

The Random Forest model demonstrated reasonable performance, particularly in classifying **Neutral** instances, but struggled with **Ambiguous** and **Negative** classes. The fine-tuning process, while yielding a more tailored model, did not improve accuracy.

XGBoost Classifier

In this section, we trained an XGBoost model for multi-class classification and evaluated its performance both with default parameters and after hyperparameter tuning using Grid Search.

Default XGBoost Model

We defined the XGBoost model and fitted it on the training data. We made predictions on the test set and evaluated the model.

Default XGBoost Accuracy: **69%**

Classification Report:
The model performed reasonably well for the **Neutral** class (Class 2), achieving high recall (89%).
However, it struggled with the **Ambiguous** (Class 0) and **Negative** (Class 1) classes, indicating challenges in correctly classifying these minority classes.

Fine-Tuning the XGBoost Model

We defined a parameter grid for tuning. We stored the best model and displayed the optimal parameters:

Best Parameters: {'learning_rate': 0.2, 'max_depth': 7, 'n_estimators': 100, 'subsample': 1.0}
Best Cross-Validated Accuracy: **66%**
Best XGBoost Accuracy: **68%**

Summary of Findings:

Default Model Performance: The default XGBoost model achieved an accuracy of **69%**.

Fine-Tuning Outcome: After hyperparameter tuning, the best model's accuracy dropped slightly to **68%**, indicating no significant improvement from the default model.

While XGBoost showed a decent performance overall, it still faced challenges in classifying the **Ambiguous** and **Negative** classes effectively. The hyperparameter tuning did not yield better performance, highlighting the need for further exploration.

7. MODEL COMPARISON

- The **Neural Networks (Fine-Tuned)** model had the highest test accuracy and precision, showing that fine-tuning significantly improves its performance.
- **Naive Bayes** and **Random Forest (Default and Fine-Tuned)** models show reasonable performance but lag behind the neural network models.
- **XGBoost** shows a slight improvement after fine-tuning but still underperforms compared to the Neural Networks.

Each model has its strengths, but based on the metrics provided, the fine-tuned Neural Networks model appears to be the best performer in terms of test accuracy and precision.

8. CONCLUSIONS

1. The sentiment analysis revealed distinct trends in customer opinions about iPhone and Google products. Positive sentiments dominated, reflecting strong customer satisfaction, while negative sentiments provided valuable insights into areas of improvement.
2. Among the models tested, the Fine-Tuned Neural Networks model performed the best with an accuracy of 90%. This demonstrates its suitability for text classification tasks like sentiment analysis.
3. Frequent mentions of specific features, such as "store," "link," and "ipad," highlighted customer priorities and areas of focus for both brands. These terms provided actionable insights into customer needs and preferences.
4. Tweets were effectively categorized into ambiguous, negative, neutral, and positive emotions. This categorization provided a nuanced understanding of customer sentiment, enabling more targeted decision-making.

9. RECOMMENDATIONS

1. Regularly monitor and analyze customer feedback on social media platforms to stay updated on changing preferences and emerging trends.
2. Investigate and address the root causes of negative sentiments. For instance, frequent mentions of "store" in a negative context could indicate issues with in-store experiences or product availability.
3. Pay attention to commonly mentioned product features and ensure they align with customer expectations. For example, improving features like "iPad" or "link" based on customer discussions can boost satisfaction.
4. Adopt and integrate sentiment analysis tools into business processes to automate and scale this analysis for other products and brands.
5. Utilize the best-performing model (Neural Networks (Fine-Tuned) for ongoing sentiment analysis tasks. Regularly retrain the model with new data to maintain its accuracy and relevance.
6. Expand the sentiment analysis framework to include competitors and other product categories. This can provide a comprehensive view of market dynamics and inform strategic decisions.

10. NEXT STEPS

We can enhance the accuracy of the model by implementing a systematic process for ongoing monitoring and maintenance once it has been deployed. It is essential to regularly assess its performance, incorporate new data as it becomes accessible, and fine-tune it in response to evolving business requirements and customer behaviors.