

RonnyMuthomi / Group4_NLP_Project

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

☆ 0 stars

🍴 1 fork

👁 0 watching

🔗 Branches

📈 Activity

🏷 Tags

🌐 Public repository

7 Branches

0 Tags

🔍 Go to file

t

Go to file

+

Add file

Code

...

RonnyMuthomi

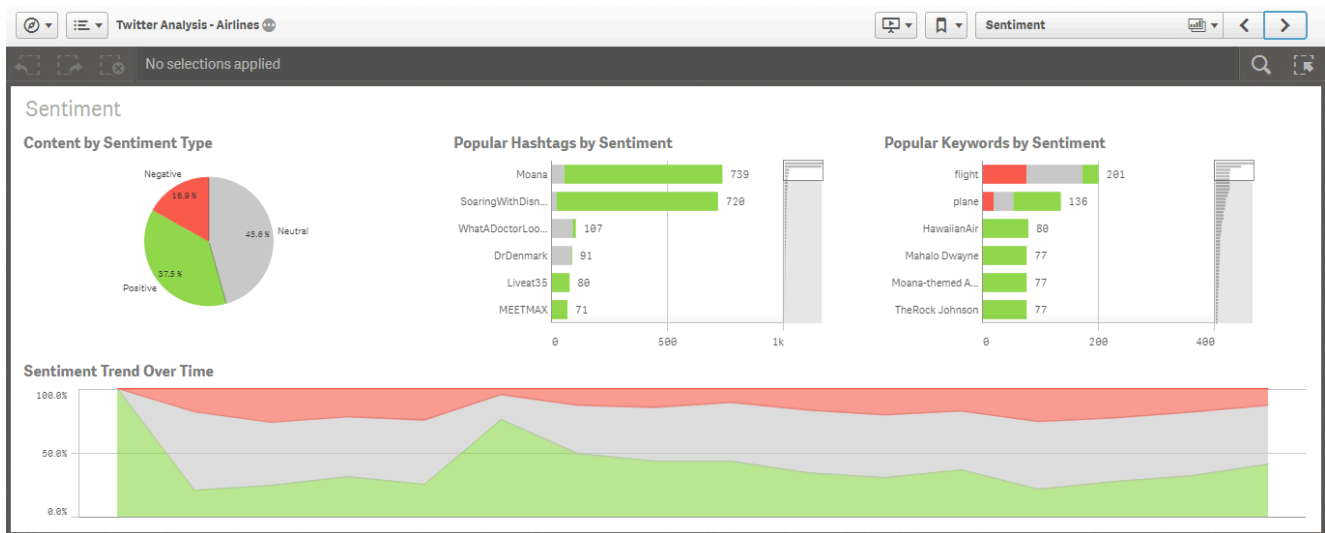
notebook file

effd1e5 · now

📁 app	Add files via upload	yesterday
📁 data	cleaned dataset for modeling saved a...	4 days ago
📁 models	saved model	2 days ago
📄 .gitignore	.gitingore file	5 days ago
📄 CRISP-DM.pdf	Data report	22 minutes ago
📄 README.md	Update README.md	2 days ago
📄 index.ipynb	conclusion and recommendation	50 minutes ago
📄 notebook.pdf	notebook file	now
📄 presentation.pdf	presention file	31 minutes ago
📄 requirements.txt	module file	yesterday

📖 README

Tweet Sentiment Classification



This project offers a practical solution by using Natural Language Processing (NLP) and machine learning to process raw tweets and classify their sentiment. The resulting model can help stakeholders

- Gauge public reaction to a product, event, or policy
- Detect and address negative feedback early

The key stakeholder here is any organization or analyst interested in understanding human emotion and behavior through digital text.

Data Understanding

The dataset used in this project consists of over 9,000 real-world tweets, each annotated with three key columns: tweet_text, product, and sentiment. The tweet_text column contains the raw content of the tweet, which serves as the primary input for natural language processing tasks. The product column identifies the brand or item referenced in the tweet ("iPhone"), while the sentiment column is the target variable representing the emotion expressed. Sentiment labels include Positive emotion, Negative emotion, No emotion toward brand or product, and I can't tell. The dataset, sourced from [CrowdFlower via data.world](#)

A preliminary review of the data revealed that the tweet text is often noisy and contains elements such as user mentions, hashtags, and URLs, all of which can interfere with model training if not properly cleaned. Furthermore, the dataset exhibits class imbalance, with a greater proportion of tweets expressing neutral or positive sentiment compared to negative ones. Since the core objective is to classify sentiment based on tweet content, most preprocessing and feature engineering steps are focused on the tweet_text column.

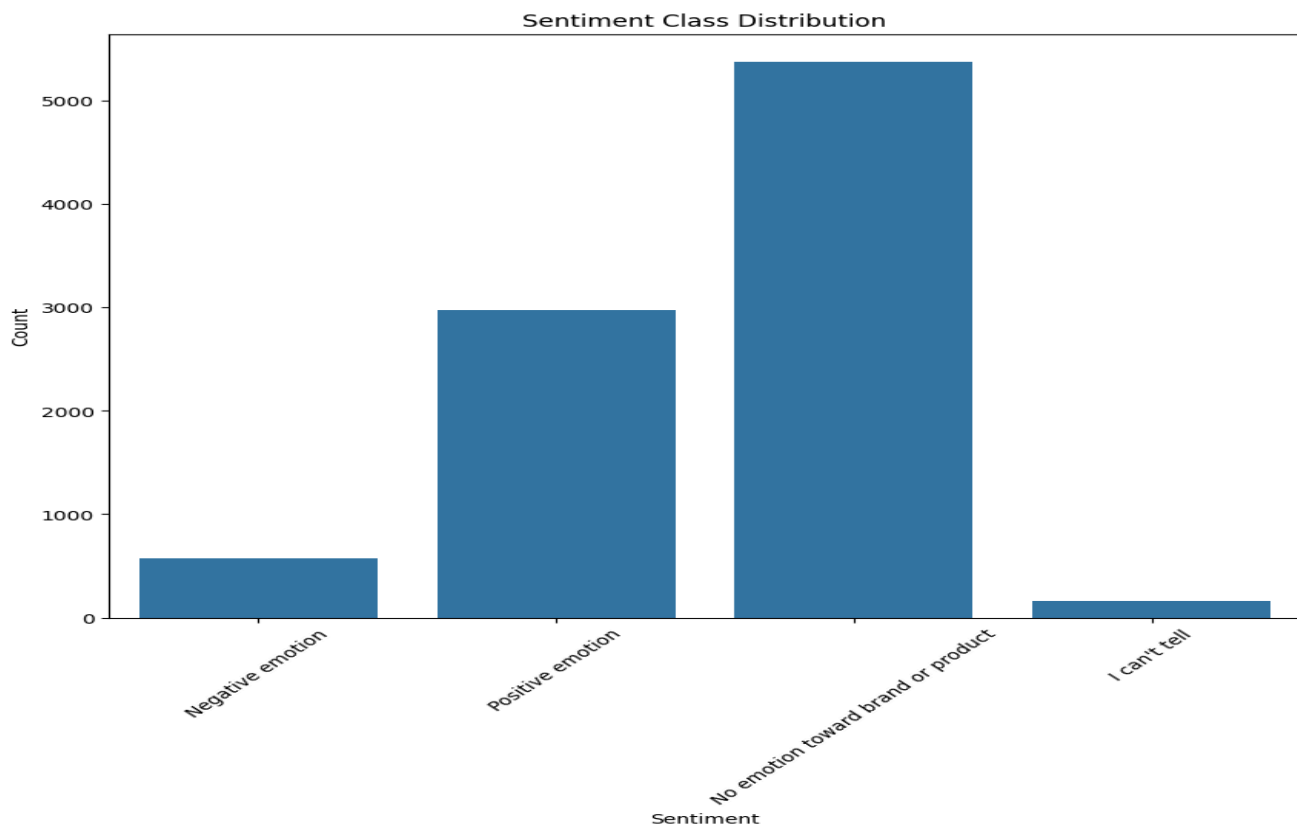
Data Preparation

Preprocessing played a critical role in transforming raw tweet text into a form suitable for machine learning. The process began with text cleaning, where regular expressions (regex) were used to remove URLs, user mentions, hashtags, punctuation, and to convert all text to lowercase for consistency. Next, the cleaned text was tokenized using NLTK, breaking each tweet into individual words or tokens. To enhance relevance, stop words common words like "the" and "is" that do not carry meaningful sentiment were removed. The tokens were then passed through a stemming process, reducing words to their root form ("running" to "run"), which helped reduce dimensionality. Finally, TF-IDF vectorization was applied to convert the processed text into numerical feature vectors, making it possible to train machine learning models. These steps ensured the model could focus on the most informative features, and the cleaned, tokenized, and vectorized versions of each tweet were stored in a new column named `processed_tweet`.

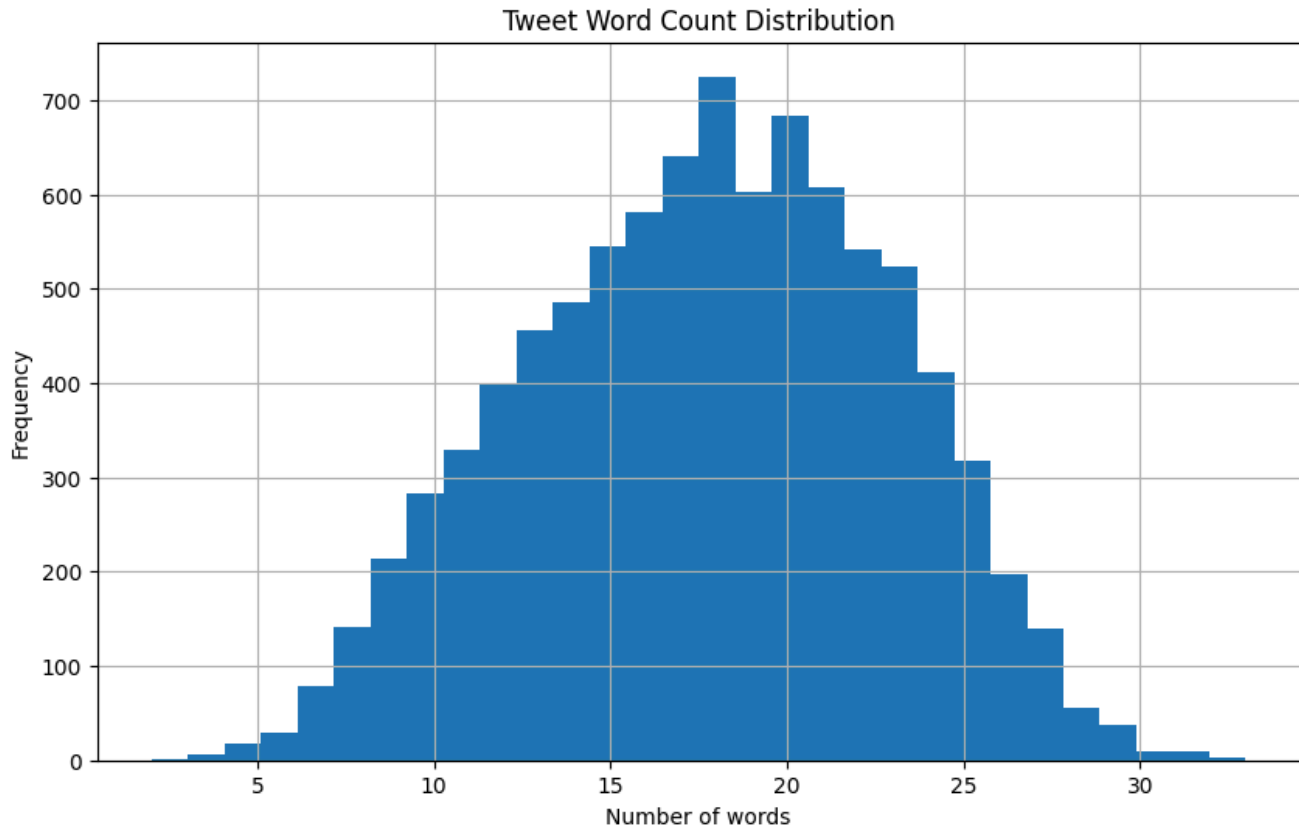
```
Raw: "@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it was dead!"  
Processed: ['g', 'iphon', 'hr', 'tweet', 'dead', 'need', 'upgrad', 'plugin', 'station']
```

Exploratory Data Analysis (EDA)

The goal of Exploratory Data Analysis EDA in this project was to gain insights into the structure and patterns within the dataset to inform better modeling decisions. One of the first observations was the clear class imbalance in the sentiment labels most tweets reflected either no emotion or positive sentiment, while negative sentiments were significantly fewer. This imbalance highlighted the need for careful evaluation metrics and possibly rebalancing techniques during model training.



Further inspection of the tweets by product category showed how sentiment varied across different brands, suggesting that brand perception might influence sentiment expression. Visualizations like bar plots and pie charts were used to show the frequency distribution of sentiments and products. Additionally, text-specific analyses such as word clouds and most frequent terms revealed common themes and vocabulary associated with each sentiment class. For example, negative tweets often contained words like "broken," "hate," or "problem," while positive tweets featured terms like "love," "great," or "happy." These findings confirmed that the textual content carried distinguishable sentiment signals that a model could learn from.



Modeling

For the modeling phase, we used the `processed_tweet` column containing cleaned and preprocessed tweet text as input features (X), and the `sentiment` column as the target variable (y).

As a starting point, we implemented a baseline **Logistic Regression** model because of its simplicity, interpretability, and proven effectiveness in text classification.

The text data was vectorized using **TF-IDF** (Term Frequency–Inverse Document Frequency), transforming each tweet into a numerical feature vector that captures the importance of words across the corpus.

Benchmarking Multiple Models

To benchmark performance, we trained and compared several machine learning models:

- Logistic Regression
- Random Forest
- Naive Bayes

- K-Nearest Neighbors (KNN)
- XGBoost

Each model was built as a pipeline:

1. TF-IDF vectorizer to transform text
2. Classifier as the final estimator

Evaluation & Metrics

All models were evaluated on the test set using:

- **Accuracy** – overall correctness
- **Precision, Recall, F1-score** – to balance false positives and false negatives
- **Weighted F1-score** – chosen as the primary metric due to class imbalance (to give fair weight to minority classes)

Best results:

Model	Accuracy	Weighted F1-score
Random Forest	~0.68	~0.66
Logistic Regression	~0.67	~0.64
XGBoost	~0.66	~0.63
Naive Bayes	~0.65	~0.62
KNN	~0.64	~0.59

The **Random Forest** pipeline achieved the best overall accuracy (~68%) and balanced performance across sentiment classes.

Hyperparameter Tuning

To boost performance, we performed **GridSearchCV** tuning:

- For Logistic Regression: regularization strength `C`, penalty type, solver
- For Random Forest: number of estimators, max depth, and class weighting
- For KNN: number of neighbors, distance metric
- For Naive Bayes: smoothing parameter `alpha`
- For XGBoost: learning rate, number of estimators, max depth

Tuning led to meaningful improvements, particularly in recall for minority sentiment classes.

Key Visuals from Modeling Phase

Confusion Matrix



[Confusion Matrix](#)

Classification Report Heatmap

 [Classification Report](#)

Top Influential Features (Logistic Regression)

 [Feature Importance](#)

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Contributors 3


-  RonnyMuthomi
-  Mohammed-Abdi-Farhan Mohammed Abdi Farhan
-  mer-cy-12-96

Languages

 Jupyter Notebook 99.7%  Python 0.3%


Suggested workflows

Based on your tech stack

 **Python Package using Anaconda**


Configure

Create and test a Python package on multiple Python versions using Anaconda for package management.

 **Publish Python Package**

Configure

Publish a Python Package to PyPI on release.

 **Pylint**

Configure

Lint a Python application with pylint.