
















KoilaBetty /  
Twitter-sentiment-NLP-model



[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#)



 **0 stars**  **0 forks**  **1 watching**  [Branches](#)  [Activity](#)  
 [Tags](#)








 [Public repository](#)




 [1 Branch](#)  [0 Tags](#)  



[Go to file](#) [+](#) [Add file](#) 


[Code](#) 

|  |                         |   |
|--|-------------------------|---|
|  <b>KoilaBetty</b> presentation pdf | 8659d08 · 2 minutes ago |  |
|  Images                             | code done               | yesterday   |
|  README.md                          | Updated README          | 29 minutes ago  |
|  judge-1377884607_tweet_pr...       | First Commit            | 2 days ago  |
|  presentation.pdf                   | presentation pdf        | 2 minutes ago   |
|  project.ipynb                     | Updated README          | 29 minutes ago  |

 **README**  

# PHASE 4 PROJECT : TWEETS SENTIMENTS ANALYSIS

## Topic 3: Natural Language Processing (NLP)

 [text](#)

### Project Overview

Our aims is to build a Natural Language Processing (NLP) model that can analyze the sentiment expressed in Tweets about Apple and Google products. The dataset consists of over 9,000 Tweets, which have been rated by human annotators as having a positive, negative, or neutral sentiment.

The main objective is to build a model that can automatically classify a Tweet based on its sentiment, enabling businesses, marketers, or product managers to quickly analyze customer feedback and sentiment about products and services. Initially, you would create a binary sentiment classifier (positive vs. negative Tweets), and later expand it into a multiclass classifier to include neutral sentiment as well.

## Business Problem

In today's digital world, companies like Apple and Google receive a vast amount of user-generated content in the form of Tweets, reviews, and comments across social media platforms. Monitoring and understanding customer sentiment on social media is crucial for businesses to:

**Track brand health:** Know whether customers have positive or negative feelings about their products or services.

**Measure customer satisfaction:** Quickly identify areas where customers are satisfied or dissatisfied, so that businesses can take proactive measures to address issues.

**Support marketing strategies:** Understand customer sentiment to better align marketing campaigns, advertisements, and promotional strategies with consumer expectations.

**Product development insights:** Feedback from users can reveal product flaws, feature requests, or unmet needs, allowing companies to make informed decisions on product improvements.

## 1.Data Understanding.

We are using a dataset sourced from **CrowdFlower via Data.world**, containing approximately 9,000 tweets expressing sentiments about Apple and Google products. This dataset includes columns such as `tweet_text`, `emotion_in_tweet_is_directed_at`, and `is_there_an_emotion_directed_at_a_brand_or_product`. The main objective is to accurately classify each tweet into one of three sentiment categories: positive, negative, or neutral.

## 2.Exploratory Data Analysis (EDA)

The dataset contains **9,093** rows and **3** columns, with `tweet_text` missing **1** value and `emotion_in_tweet_is_directed_at` missing **5,802** values. There are **22** duplicated rows and `is_there_an_emotion_directed_at_a_brand_or_product` has no missing values

### 2.1 Data Visualization.

#### 2.1a Sentiment Distribution

#### 2.1b Most Frequent Tweets

Above outputs and plot indicate that **No emotion toward brand or product** has 5,389 Tweets which is 58.9% of the total tweets showing that a significant number of Tweets do not express any emotion toward a specific brand or product. These could be neutral or unrelated Tweets. While the least tweets had a **I can't tell** feedback with 156 Tweets (1.7%), these Tweets are ambiguous or unclear in terms of sentiment. It may be challenging to classify them as positive or negative, or they could contain mixed emotions.

#### 2.1c Sentiment by Product

The bar chart above indicates that **iPad** is the most frequently mentioned product in the tweets, followed by other **Apple products** (iPad, iPhone, and Apple) and **Google products**. **Android-related products** received fewer mentions, highlighting the dominance of Apple products in user-directed sentiments.

## 2.2 Column Renaming

Rename columns with long names for clarity and ease of analysis. These long column names are renamed as `emotion_in_tweet_is_directed_at` to `Product_brand` and the column `is_there_an_emotion_directed_at_a_brand_or_product` to `Sentiment`

## 2.3 Handle Missing Values

Using `SimpleImputer` to fill missing values with a constant strategy

## 2.4 Handle Duplicate Values

Drop duplicate values in the data set

## 2.5 Mapping the Columns

Here we are converting the `tweet_text` data type to strings while re-mapping the `Product_brand` column to fewer brands and the `Sentiment` column to either **Positive**, **Negative** or **Neutral**. **iPad** and **Apple** are the frequently mentioned products in the tweets as compared to **Google** and **Android** with few mentions.

Our focus is comparing **Apple** and **Google** products after mapping product brands to the `brand_item` variable. This will help us further analyze the sentiment and mentions between these two major brands. The plot shows that for **positive sentiment**, Apple has a significantly higher count compared to Google, indicating a strong positive reaction toward Apple products. **Negative sentiment** is more balanced but still higher for Apple than Google. Both brands have very low counts in the **neutral sentiment** category, with Apple showing slightly more mentions than Google. This comparison suggests that Apple products generate more engagement, particularly in positive sentiment, than Google products.

## 3. Data Processing.

Here we clean and prepare the `tweet_text` column by:

- Lowercasing the text
- Removing URLs, Mentions, and Hashtags
- Removing special characters, punctuation, and numbers
- Tokenizing the text (splitting it into words)
- Removing stop words (common words like "the", "is", etc.)
- Lemmatizing (reducing words to their root form like "running" -> "run")

### 3.1 Text preprocessing - Cleaning Text

The tweets contain unnecessary elements like URLs, mentions, special characters, etc. Let's clean the text.

### 3.2 Text preprocessing - Lemmatization and Stopword Removal

-We reduce words to their base or root form, preserving valid words by **lemmatization**. This helps in standardizing word forms and improves model accuracy. -We remove common words that don't add significant meaning to the analysis by doing the **stopword removal**, helping the model focus on important words

### 3.3 Word Cloud for the tweets.

This is a great way to visualize the most frequent words in the set of tweets, where larger words indicate higher frequency. This visualization will help you identify trends, such as common terms associated with Apple or Google products in the tweets.

#### *Negative Sentiment Word Cloud Analysis:*

The word cloud reveals that iPhone, Google, and app are frequent terms in tweets with negative sentiment. Common complaints highlighted by the users involve issues such as crashes, dead devices, and the need for upgrades. This suggests that product performance and reliability are key concerns for users, indicating that frustration with technical problems is prevalent in their experiences with these products.

#### *Positive Sentiment Word Cloud Analysis:*

The word cloud generated from positive sentiment tweets also features frequent terms like iPhone, Google, and app, indicating that these products receive significant positive attention. Words such as "awesome", "upgrade", and "festival" suggest excitement and satisfaction. These terms reflect positive user experiences, highlighting appreciation for product features, performance, or even related events that enhance customer enthusiasm.

#### *Neutral Sentiment Word Cloud Analysis:*

The word cloud for neutral sentiment tweets shows frequent terms such as iPhone, Google, and app, which also appear in both positive and negative sentiment tweets. However, the neutral tone suggests these tweets are more factual and less emotionally charged. Terms like "link", "year", and "plugin" are prominent, indicating that users are likely discussing general information or sharing details about these products without expressing strong opinions or emotions.

## 4. Modelling

### *Building and Evaluating Sentiment Classification Models*

In this section, we will build and evaluate several machine learning models to classify sentiments in the dataset. The objective is to identify the most effective model for both binary classification (positive/negative) and multi-class classification (positive/negative/neutral). We will compare various models and analyze their performance to select the best one for the task at hand.

### 4.1 Preparing Data for Binary or Multi-class Classification

We start by preparing the data for modeling, ensuring that it aligns with the classification task at hand. For binary classification (positive/negative sentiment), we filter the dataset to include only the positive and negative sentiment labels. For multi-class classification (positive/negative/neutral sentiment), we retain the neutral sentiment labels as well.

Next, we encode the target sentiment labels into numerical values, which will be used by machine learning models. Additionally, we prepare the features for the model, which include the processed tweet text and product brand. This step ensures that both textual data and categorical information (like product brand) are appropriately formatted for input into the model.

## 4.2 Vectorization Using TF-IDF

To prepare the text data for machine learning, we transform it into numerical form using the `TfidfVectorizer`. This technique converts the `preprocessed_text` into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features, which capture the importance of each word relative to the document and the entire corpus. This transformation helps represent the text data in a format that machine learning models can understand.

In addition to text, we also encode the categorical variable `product_brand` using `OneHotEncoder`. This method creates binary features for each brand, allowing the model to utilize brand information during classification. To streamline this process, a `ColumnTransformer` is employed to apply the `TfidfVectorizer` to the text feature and the `OneHotEncoder` to the categorical brand feature simultaneously.

## 4.3 Pipelines (Binary Classification)

To streamline the process of preprocessing and model training, we define several pipelines for different machine learning algorithms. Each pipeline ensures that the steps of data transformation, feature extraction, and model training are executed seamlessly in sequence. The pipelines for binary classification include the following algorithms:

*Logistic Regression:* A simple yet effective model for binary classification that works well for linearly separable data.

*Random Forest :* An ensemble learning method that builds multiple decision trees to improve accuracy and robustness.

These pipelines allow for easy experimentation with different models while ensuring consistent preprocessing across all models. Each model will be trained and evaluated using the same data transformations.

### 5.1 Training the Models

We apply the defined pipelines to train various machine learning algorithms. Each model is fitted on the training data, which has undergone the necessary preprocessing, including vectorization of text and encoding of categorical variables.

### 5.2 Model Evaluation

After training the models, we evaluate their performance on the test dataset. This involves comparing their predicted sentiments to the actual sentiments in the test data. Key performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix are used to assess the effectiveness of each model.

### 5.5 Model Testing

Here, we test the performance of the selected model using a randomly chosen sample from the test dataset. The model makes predictions based on the processed text and product brand features. The predicted sentiment is then compared to the actual sentiment from the test data.

The primary goal is to evaluate the model's accuracy on individual cases and assess how well it generalizes to unseen data. This step ensures that the model is not overfitting to the training data and can make accurate predictions on new, real-world examples. Additionally, this process can be repeated for different models to compare their performance and select the best one for the task.

## Handling Class Imbalances – Using Class Weighting and SMOTE

In this step, we address class imbalance by applying class weighting and SMOTE. We evaluate both binary and multi-class models using accuracy and F1 scores. The model with the highest F1 score is chosen, ensuring a balanced assessment of precision and recall for all classes.

### 1. Calculating Class Weights to Handle Imbalanced Data

Comparing the performance of logistic regression and random forest models for both binary and multi-class classification tasks. We've applied two strategies—class weighting and SMOTE (Synthetic Minority Over-sampling Technique)—to address imbalances in your dataset. Here's a breakdown of the results:

## Binary Classification (Positive vs Negative)

- **With Class Weighting:**
  - Logistic Regression:
    - Accuracy: 88.60%
    - F1 Score: 88.53%
  - Random Forest:
    - Accuracy: 88.65%
    - F1 Score: 87.23%
- **With SMOTE:**
  - Logistic Regression:
    - Accuracy: 88.71%
    - F1 Score: 88.65%
  - Random Forest:
    - Accuracy: 88.98%
    - F1 Score: 87.98%

## Multi-class Classification (Positive vs Negative vs Neutral)

- **With Class Weighting:**
  - Logistic Regression:
    - Accuracy: 88.60%
    - F1 Score: 88.53%
  - Random Forest:
    - Accuracy: 88.65%
    - F1 Score: 87.23%
- **With SMOTE:**
  - Logistic Regression:
    - Accuracy: 88.71%
    - F1 Score: 88.65%
  - Random Forest:

- Accuracy: 88.98%
- F1 Score: 87.98%

## Key Insights:

1. **SMOTE helps slightly improve performance** across both models for both binary and multi-class tasks, particularly boosting F1 scores, which indicate better balance between precision and recall.
2. **Random Forest performs better than Logistic Regression** in terms of accuracy and F1 score in both binary and multi-class settings when using SMOTE.
3. **Class weighting** improves performance as well but not to the same extent as SMOTE, especially for Random Forest, where the F1 score benefits from SMOTE more.

## Conclusion:

SMOTE appears to be the more effective approach for improving model performance, especially for Random Forest. However, the difference in performance between the two approaches is relatively small, so either could be a good choice depending on your specific needs (e.g., model interpretability or speed).

## Recommendation:

Based on the results, here are a few recommendations:

### 1. Use SMOTE for Imbalanced Data:

- Since SMOTE consistently improves performance across both models (Logistic Regression and Random Forest), it is recommended to use SMOTE if you are dealing with an imbalanced dataset. SMOTE works well to improve both accuracy and F1 scores by generating synthetic samples for the underrepresented class, which can lead to better overall model performance.

### 2. Prefer Random Forest for Higher Performance:

- Random Forest generally outperforms Logistic Regression in both binary and multi-class classification tasks in terms of accuracy and F1 score, especially when SMOTE is used. If model interpretability is not the top priority, Random Forest should be the preferred model for better predictive performance.

### 3. Class Weighting:

- Class weighting improves performance, but the gain is not as significant as SMOTE. It can still be useful in scenarios where SMOTE is not feasible or when computational efficiency is a key concern. It can be considered as an alternative if resources or time are limited.

### 4. Further Experimentation:

- While the results are promising with SMOTE, further experimentation with hyperparameter tuning (especially for Random Forest) or trying other sampling methods like Tomek links or ADASYN might lead to even better results.



## Conclusion:

In conclusion, **SMOTE** proves to be a more effective strategy than class weighting for improving performance on imbalanced datasets, particularly for **Random Forest**, which yields higher accuracy and F1 scores. Random Forest outperforms Logistic Regression in this case, making it a better choice when prioritizing performance. Therefore, for achieving the best results in both binary and multi-class classification tasks, it is recommended to use **SMOTE with Random Forest**.

For situations where **interpretability** is crucial or if Random Forest is not suitable, **Logistic Regression with SMOTE** remains a solid alternative.

## Releases

No releases published

[Create a new release](#)

## Packages

No packages published

[Publish your first package](#)

## Languages

● Jupyter Notebook 100.0%