Hafsa-mohamed-Ad3n / **Sentimental-Analysis-of-Tweets**

Code Issues Pull requests Actions Projects Wiki Security Insights Settin

0 stars 2 forks 0 watching Branches Activity
Tags

Public repository

6 Branches 0 Tags Go to file t Go to file Add file Code

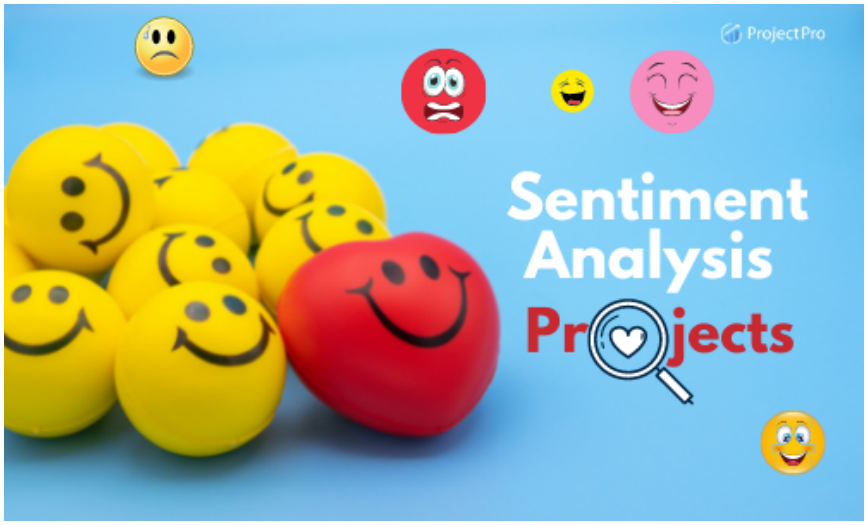
RWKarimi Merge pull request #3 from Hafsa-mohamed-Ad3n/feature/model-deployment

423f361 · now

Data	Initial commit from local Phase 4 Pr...	3 days ago
Image	ds-2 commit	3 minutes ago
model_registry	Add FastAPI app and deployment s...	yesterday
static	ft-Create a user interface	15 hours ago
templates	ft-Create a user interface	15 hours ago
.gitignore	ft-model tuning	2 days ago
README.md	ds-merge deployment branch on to...	41 minutes ago
Sentiment_Analysis.pdf	ds-2 commit	3 minutes ago
app.py	f- Update on the app.py file	9 minutes ago
requirements.txt	Add FastAPI app and deployment s...	yesterday
sentiment_model.pkl	f- Update on the app.py file	9 minutes ago
tweets_analysis.ipynb	ds-2 commit	3 minutes ago
tweets_analysis.pdf	ds-2 commit	3 minutes ago
tweets_analysis.py	f-Update on the user interface	14 hours ago

README

Sentiment Analysis of Tweets about Apple and Google Products



Authors

1. Hafsa M. Aden
2. Ryan Karimi
3. Harrison Kuria
4. Rose Muthini
5. [Lewis Mbugua]
6. Elizabeth Ogutu

Project Overview

- The aim of this project is to build a Natural Language Processing (NLP) model that can classify tweets about Apple and Google products into Positive, Negative, or Neutral sentiments.
- By leveraging machine learning techniques, the project seeks to provide a scalable way to analyze public opinion from social media, offering valuable insights into customer perceptions and brand sentiment.
- The ultimate goal is to demonstrate how our analysis can support decision-making and market understanding for technology companies.

Project Colloborations

We used Notion as our project management tool to organize the project timeline, assign and track tasks, and coordinate team contributions, ensuring smooth and effective collaboration throughout the project.

Objectives

- Exploratory Data Analysis (EDA)

    - Create visualizations (word clouds) to understand the nature of the text and most common words.
    - Plot histograms to analyze the distribution of sentiment classes.

- Data Preprocessing - Preparing the tweet text for sentiment analysis by

    - Removing URLs, user mentions, hashtags, numbers, and special characters
    - Perform tokenization, stop-word removal, lemmatization, and stemming on the text data.
    - Encoded target variable and vectorized feature matrix for machine learning.
    - Building pipeline to combine preprocessing, feature extraction, and modeling into a single reproducible process.

- Model Selection & Deplyoment.

    - Build an Natural Language Processing (NLP) model to analyze Twitter sentiment about Apple and Google products.
    - Evaluate the model (accuracy, precision, recall, F1-score).
    - Provide insights on how this model can be useful for businesses.

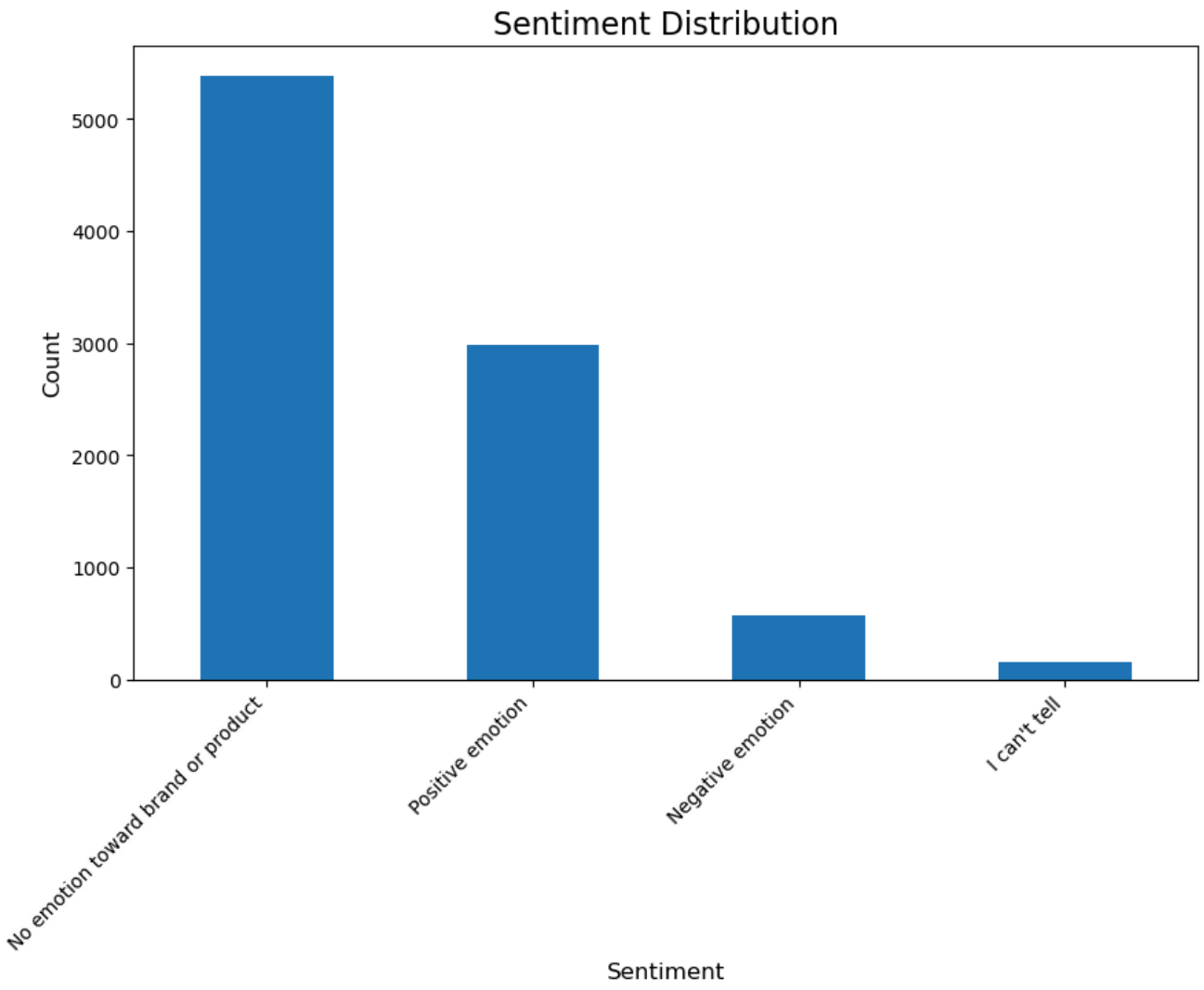
Project Workflow

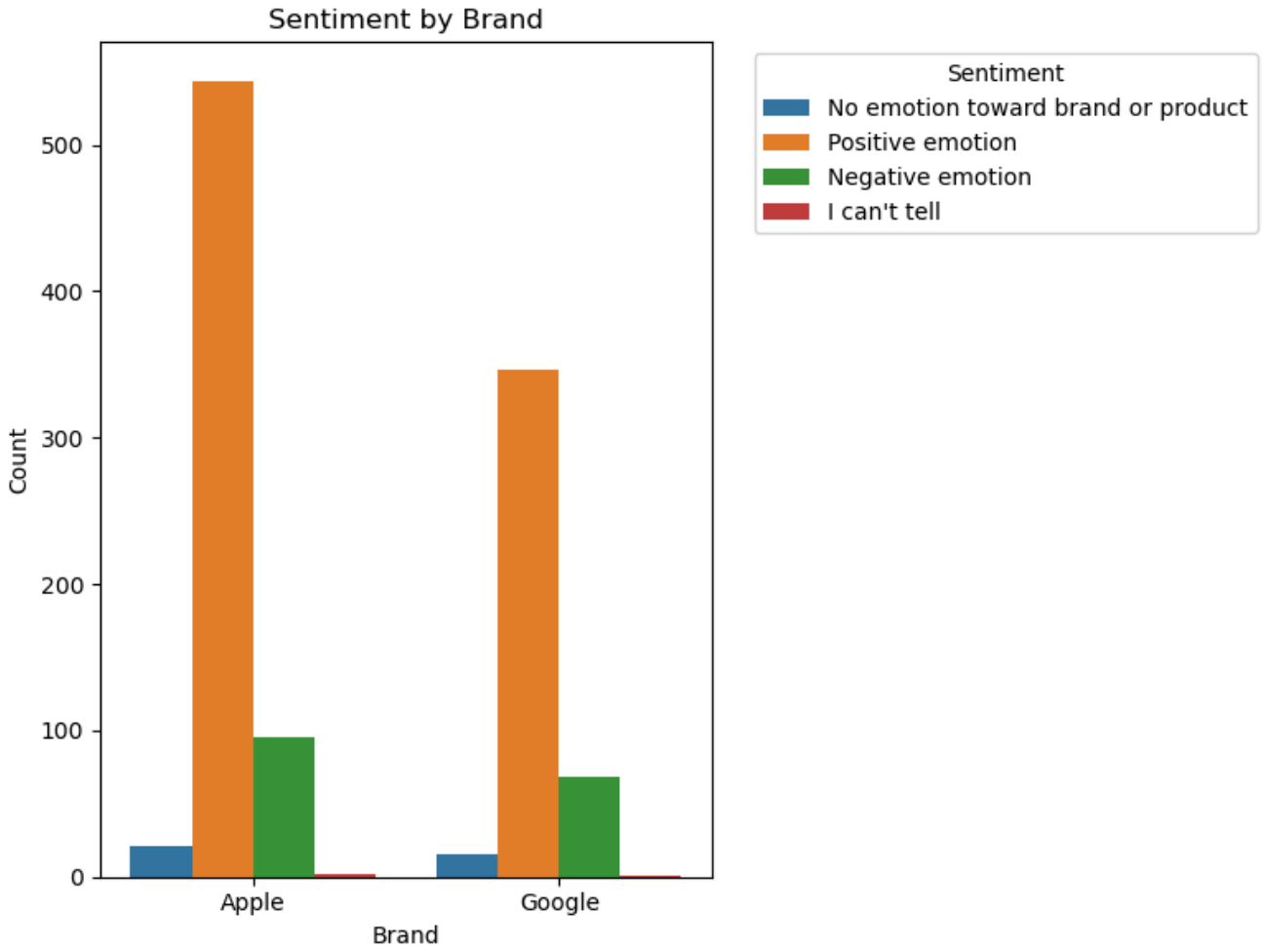
Data Source.

The dataset used in this project comes from CrowdFlower and contains over 9,000 tweets, each labeled by human raters as expressing a positive, negative, or neutral sentiment.

Exploratory Data Analysis (EDA)

- In order to better understand the dataset and prepare it for sentiment analysis, we focused on the following checks: - Preview the data: Inspect the first few rows to quickly grasp the dataset's structure. - Handled any missing data that could introduce bias or cause issues during preprocessing and modeling. - Removed duplicated tweets to prevent overrepresentation of certain entries. - Reviewed the balance of sentiment categories, since skewed classes may bias the model toward majority classes. - Explored brand distribution and compared tweets related to Apple vs. Google to see if one brand dominates the dataset.


Sentiment Distribution
Count
5000
4000
3000
2000
1000
0
No emotion toward brand or product
Positive emotion
Negative emotion
I can't tell
Sentiment



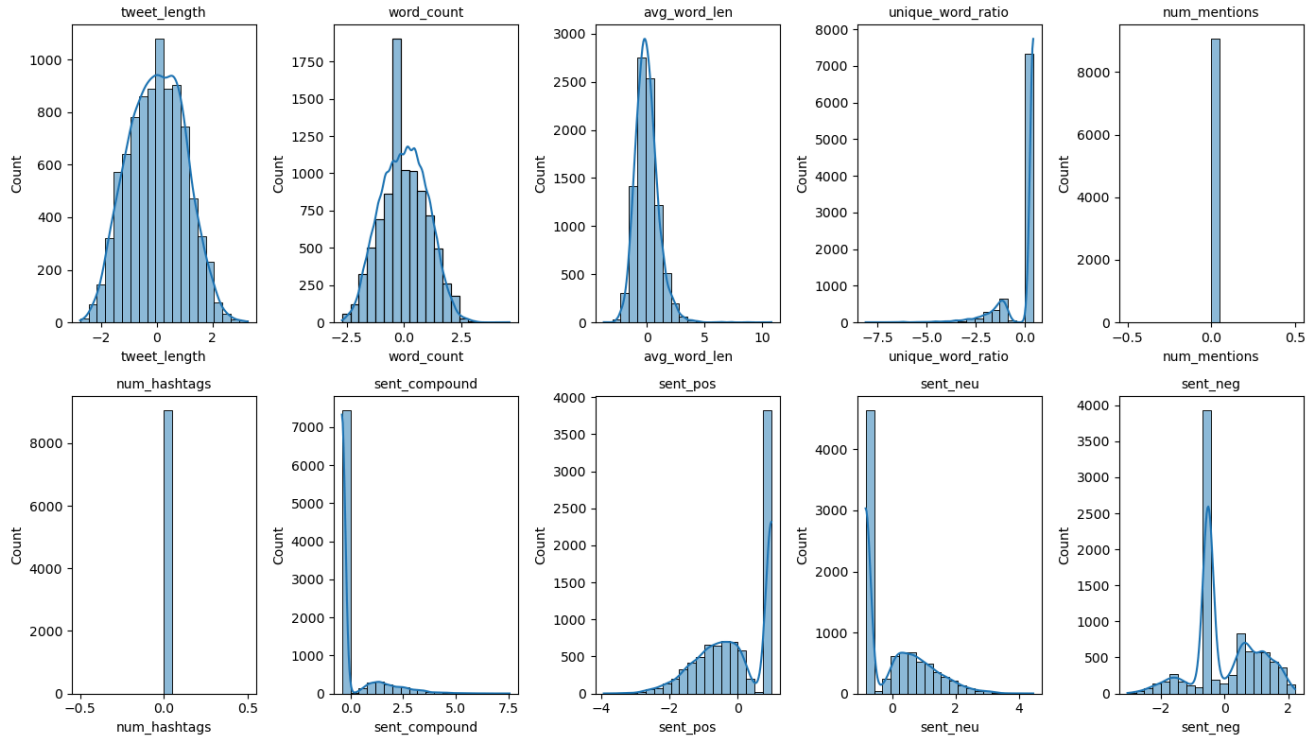
Data Preprocessing

- To prepare the tweets for modeling, we applied several text-cleaning and transformation steps:
  - Text Cleaning: Remove URLs, user mentions, hashtags, numbers, and special characters.
  - Tokenization: Break sentences into words for analysis.
  - Stop-word Removal: Exclude common words (the, and, is) that add little meaning.
  - Lemmatization: Reduce words to their base forms to treat similar terms consistently.
  - Vectorization: Convert text into numerical features using techniques like TF-IDF or CountVectorizer.
  - Label Encoding: Encode sentiment classes into numeric values for model training.

Feature Engineering

- To convert textual data into machine-readable form, we applied: - Bag of Words (BoW): Captures word frequency within tweets. - Term Frequency–Inverse Document Frequency (TF-IDF): Assigning higher weight to distinctive words while reducing weight for common ones. - N-grams: Included bi-grams and tri-grams to capture short word sequences that add context beyond single words.

- These allowed our models to better capture the semantic and syntactic patterns of the tweets.



Pipelines

- We built pipelines to combine preprocessing, feature extraction, and modeling into a single reproducible process. This ensured:
  - Consistency across training and testing
  - Simplified experimentation with different models

Modeling

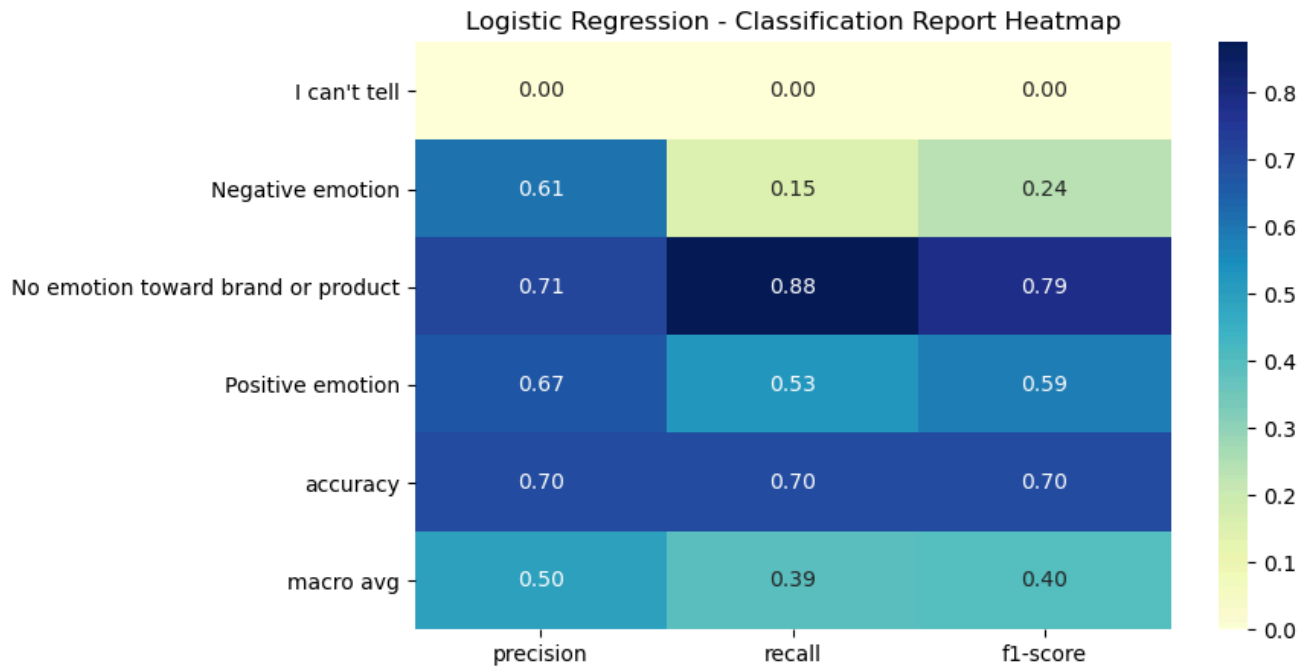
- The model building process for this project was carried out in four main steps:
  - Data preparation:
    - Features (X): The text of each tweet (tweet_text).
    - Target (y): Whether an emotion was directed at a brand (is_there_an_emotion_directed_at_a_brand_or_product).
    - The dataset was split into training (80%) and testing (20%) sets.
  - Baseline Model Comparison - We tested three ML classifiers to establish baseline performance:
    - Logistic Regression
    - Support Vector Machine (SVM)
    - Random Forest
  - Each model was trained on the preprocessed training data and evaluated on the test set. Key metrics included:
    - Accuracy
    - Precision, Recall, F1-score (per class)

        - Classification reports
    - To better understand the performance of the model:
        - We plotted Bar charts showing accuracy scores for all models for comparisons.
        - Heatmaps for each model to visualize prediction errors across classes.
        - Detailed breakdown of Logistic Regression performance (precision, recall, F1).
    - We performed hyperparameter optimization using GridSearchCV for the two strongest text classification models
    - The grid search was performed with 5-fold cross-validation, selecting the best parameters based on accuracy.

Observation

- Model Performance:
    - Among the baseline models, Logistic Regression achieved the highest accuracy (~70%) and overall balanced performance across classes.
    - Linear Support Vector Machine performed slightly lower (~68%) but showed comparable results.
    - Random Forest underperformed (~63%), struggling particularly with minority classes.
- After Hyperparameter Tuning:
    - Both Logistic Regression and Linear SVM converged to similar performance (~65% accuracy), but showed improved recall for the Negative emotion class.
    - Performance on the "I can't tell" class remained poor across all models, likely due to class imbalance and limited training samples.

Model	Accuracy
Logistic Regression	0.70
Linear SVM	0.68
Random Forest	0.63
Tuned Logistic Regression	0.65
Tuned Linear SVM	0.65



Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 6

Languages

Jupyter Notebook 99.1% Other 0.9%

Suggested workflows

Based on your tech stack

Publish Python Package Configure

Publish a Python Package to PyPI on release.

Node.js

Build and test a Node.js project with npm.

Configure

Jekyll using Docker image

Package a Jekyll site using the jekyll/builder Docker image.

Configure

More workflows

Dismiss suggestions