Vishw Jigneshkumar Patel
Module 3 Assignment: AI Solution Assignment
EAI6020: AI Systems Technology
Date: March 21, 2025

Introduction

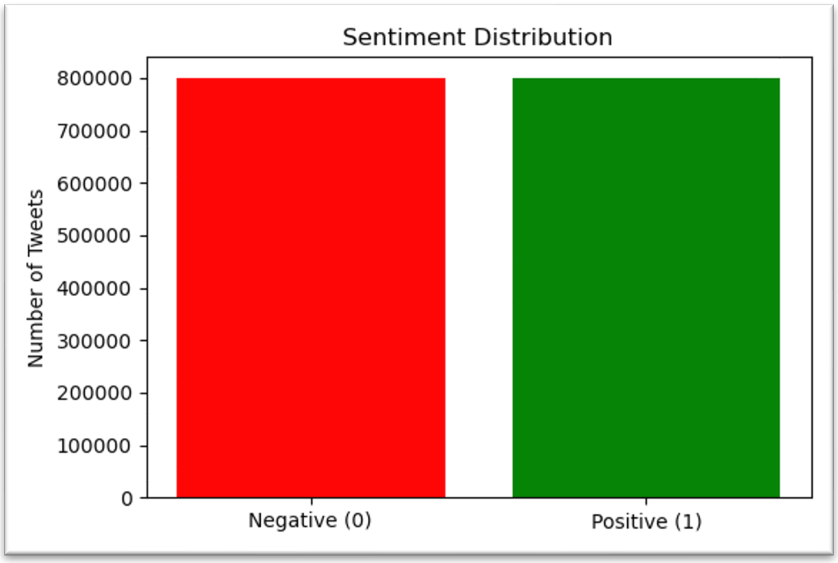
In the current digital era, social media sites like Twitter act as immediate reflections of public opinion. Each day, millions of users express their thoughts, feelings, and responses, making platforms like Twitter a significant source of unstructured text data. Conducting sentiment analysis on this data allows organizations to uncover insights related to customer satisfaction, public sentiments, and market trends. Nevertheless, engaging with social media data presents various challenges due to its noisy, casual, and ever-changing characteristics.

This project aims to develop a comprehensive machine learning pipeline that categorizes tweets as positive or negative. Utilizing the Sentiment140 dataset, which comprises 1.6 million labeled tweets, the pipeline encompasses the entire process—from data preprocessing, feature extraction using TF-IDF, model training with Logistic Regression, evaluation, to ultimately deploying the model as a Flask API. The goal is to create a lightweight and scalable solution capable of processing new tweets and providing real-time sentiment predictions.

Dataset and Problem Statement

The project utilizes the Sentiment140 dataset, which consists of 1.6 million tweets that have been categorized by sentiment. The initial labels are 0 for negative tweets and 4 for positive tweets. To maintain consistency, the label '4' was modified to '1', transforming the problem into a binary classification task. Each tweet contains metadata including tweet ID, user details, date, and the primary content (text).

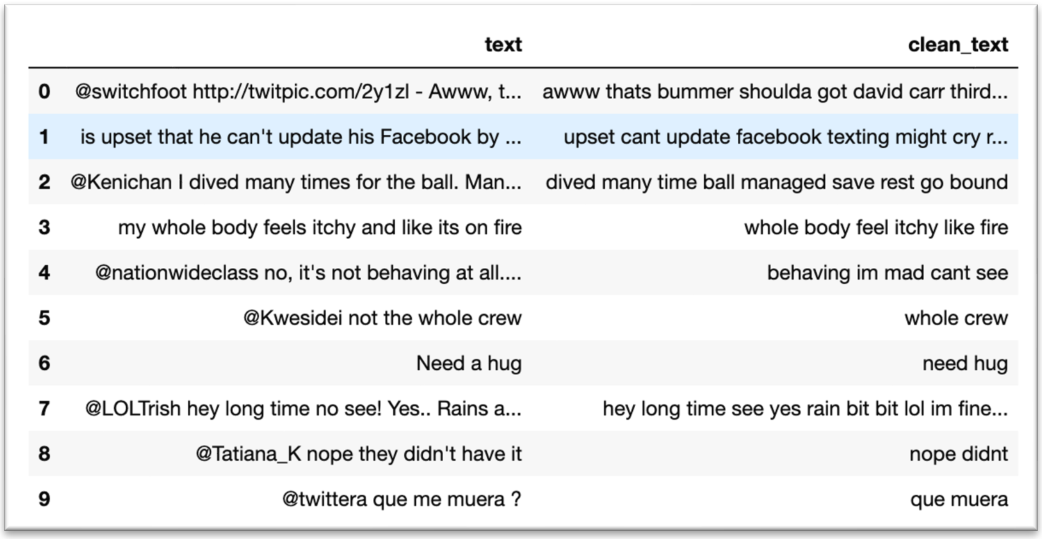
The objective is to automatically classify unseen tweets as either positive or negative by employing a supervised machine learning model that has been trained on this dataset. An initial analysis indicated that the dataset is well-balanced, with approximately 50% of the tweets classified as positive and 50% as negative, thereby eliminating any major class imbalance.



Data Preprocessing

Reprocessing textual data is an essential phase before it is input into any machine learning model. In this project, the tweets went through a variety of transformations. Initially, all text was converted to lowercase, and URLs, mentions (@username), hashtags, numbers, and punctuation were eliminated using regular expressions. Tokenization was carried out utilizing NLTK's word_tokenize, and stopwords (frequent words like "the", "and", "is") were removed to preserve significant content. Ultimately, lemmatization was applied to convert words to their root form (e.g., "running" → "run").

After the tweets were sanitized, they were converted into numerical forms through TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This method highlights words that are distinctive and minimizes the influence of common ones. The top 5,000 features (words) were chosen to ensure the model remains efficient while still covering a wide range of vocabulary.



	text	clean_text
0	@switchfoot http://twitpic.com/2y1zl - Awww, t...	awww thats bummer shoulda got david carr third...
1	is upset that he can't update his Facebook by ...	upset cant update facebook texting might cry r...
2	@Kenichan I dived many times for the ball. Man...	dived many time ball managed save rest go bound
3	my whole body feels itchy and like its on fire	whole body feel itchy like fire
4	@nationwideclass no, it's not behaving at all....	behaving im mad cant see
5	@Kwesidei not the whole crew	whole crew
6	Need a hug	need hug
7	@LOLTrish hey long time no see! Yes.. Rains a...	hey long time see yes rain bit bit lol im fine...
8	@Tatiana_K nope they didn't have it	nope didnt
9	@twittera que me muera ?	que muera

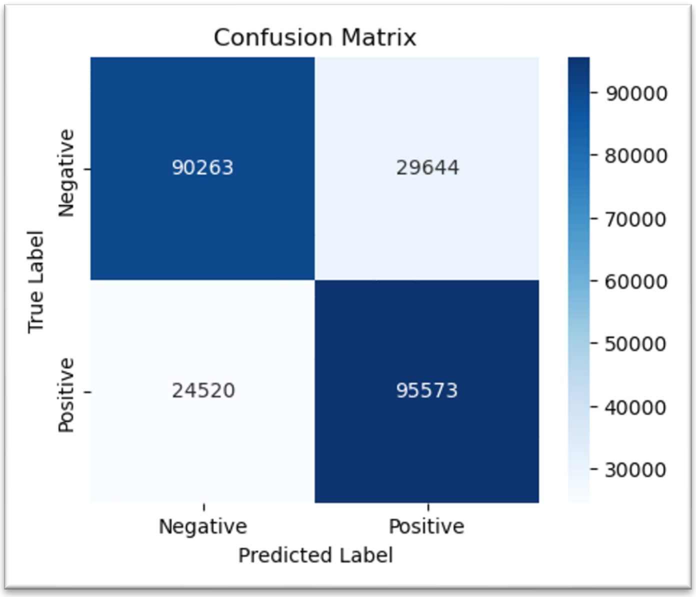
Model Selection and Training

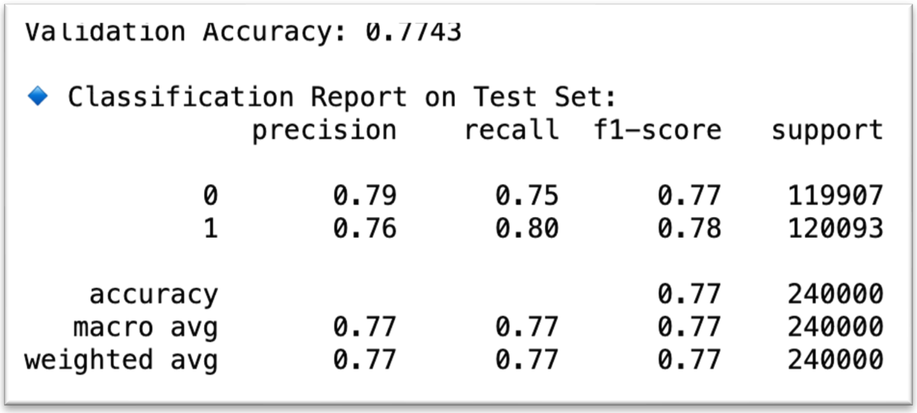
I tested several models and determined that Logistic Regression was the most efficient concerning both performance and training speed. Although Random Forest is a powerful option, it proved to be resource-intensive and slower with this extensive dataset.

The dataset was divided into three subsets: 70% for training, 15% for validation, and 15% for testing. A maximum of 1000 iterations was set for the logistic regression model to guarantee convergence. Following training on the TF-IDF transformed data, the model reached a validation accuracy of 77.43%.

To assess the final model, predictions were generated on the test dataset. The model delivered commendable results, obtaining an overall accuracy of 77%, with a precision of 76%, a recall of

80%, and an F1-score of 77% for the positive class. This demonstrates a well-balanced performance in accurately predicting positive tweets while minimizing false positives.





```
Validation Accuracy: 0.7743

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

           0       0.79      0.75      0.77    119907
           1       0.76      0.80      0.78    120093

    accuracy                           0.77    240000
   macro avg       0.77      0.77      0.77    240000
weighted avg       0.77      0.77      0.77    240000
```

Challenges and Solutions

One major challenge was cleaning noisy Twitter data, which often includes slang, emojis, URLs, and user mentions. This was addressed through a series of preprocessing steps involving regular expressions, stopword removal, and lemmatization.

Another challenge was choosing the right model. While Random Forest and other ensemble methods provided decent results, they were slow to train and not ideal for large-scale deployment. Logistic Regression, when combined with TF-IDF, proved to be a fast and effective solution with minimal hyperparameter tuning.

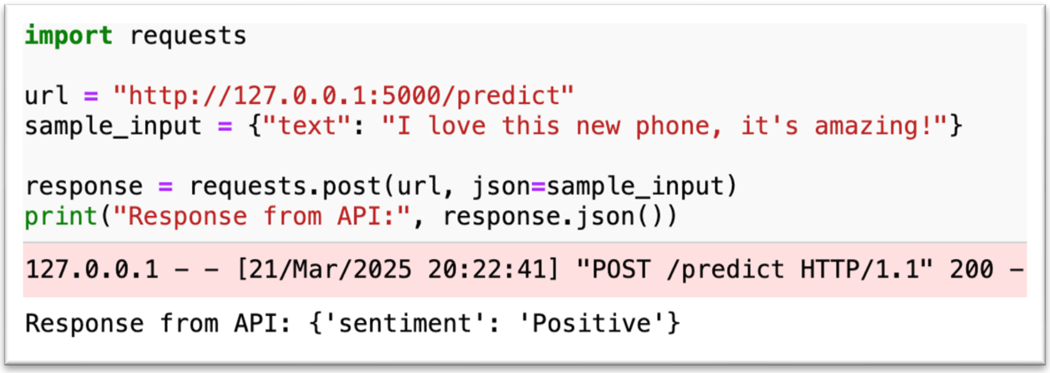
Deploying the model inside Jupyter Notebook was another hurdle, as traditional Flask servers block execution. This was resolved using Python's threading module, which allowed the Flask server to run in the background within the notebook itself.

Model Deployment and API Usage

Once the model was trained and validated, it was saved using Python's pickle module along with the TF-IDF vectorizer to ensure that any incoming tweet could be preprocessed and transformed in the same way as the training data. This step was essential to maintain consistency and accuracy during deployment.

To make the model accessible for real-time predictions, a simple RESTful API was developed using Flask. The API consists of a single endpoint (/predict) which accepts POST requests containing a tweet. Upon receiving input, the API applies the same preprocessing pipeline used during training—this includes cleaning the text, removing noise, lemmatizing, and vectorizing using the saved TF-IDF model. The processed input is then fed into the trained logistic regression model to generate a sentiment prediction, which is returned as either "Positive" or "Negative".

The API can be run locally either from the terminal or directly inside a Jupyter Notebook. When running inside Jupyter, a background thread is used to prevent the Flask server from blocking further execution. To test the API, users can send a sample tweet using tools like Postman or Python's requests library and receive a JSON response indicating the predicted sentiment. This setup enables seamless and interactive use of the model and lays the foundation for scaling it into larger applications or integrating it with user-facing platforms.



```
import requests

url = "http://127.0.0.1:5000/predict"
sample_input = {"text": "I love this new phone, it's amazing!"}

response = requests.post(url, json=sample_input)
print("Response from API:", response.json())
```

```
127.0.0.1 - - [21/Mar/2025 20:22:41] "POST /predict HTTP/1.1" 200 -
Response from API: {'sentiment': 'Positive'}
```

Conclusion

This project effectively showcases the establishment of a comprehensive machine learning pipeline for analyzing sentiment on Twitter, encompassing all stages from data preprocessing to the deployment phase. Utilizing the Sentiment140 dataset and leveraging natural language processing methods, we cleaned and organized the data, converted it into a numerical format through TF-IDF vectorization, and trained a Logistic Regression model that achieved a commendable accuracy of 77%. Employing visualizations such as sentiment distribution and confusion matrix played a crucial role in affirming the model's efficacy and providing clear insights into its classification performance.

In addition to developing the model, the project focused on practical deployment by embedding the trained model into a Flask-based API. This enables users to submit any tweet and promptly receive a sentiment prediction, rendering the solution both interactive and scalable. While traditional machine learning approaches were adequate for this scenario, future improvements could explore the use of deep learning architectures like LSTM or BERT for enhanced contextual comprehension. Overall, this project not only underscores technical proficiency in machine learning and NLP but also presents a functional, real-world application for sentiment analysis.

https://github.com/VishwPatel304/Twitter-Sentiment-Analysis - Github Repository

References:

KazAnova, &Mu;&alpha;&rho;&iota;&omicron;&sigmaf; &Mu;&iota;&chi;&alpha;&eta;&lambda;&iota;&delta;&eta;&sigmaf; (2017, September 13). Sentiment140 dataset with 1.6 million tweets. Kaggle. https://www.kaggle.com/datasets/kazanova/sentiment140/data

GeeksforGeeks. (2022, June 15). Deploy machine learning model using flask. https://www.geeksforgeeks.org/deploy-machine-learning-model-using-flask/

Stanford. (n.d.-f). https://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.