Enhancing Tweet Sentiment Analysis with Emoji Integration

Shivam Vishwakarma

Roll No. - 210102080

Date: 8th May 2025

Repository Link: https://github.com/v-shivam/Emoji-Enhanced-Chat-Sentiment.git

Motivation & Objectives

Why this topic?

- Tweets are rich in emojis, which carry strong emotional cues.
- Text-only sentiment models often miss or misinterpret these cues.

Objectives:

1. Assign each emoji a fixed binary polarity (0/1) using historical sentiment data.
2. Clean and vectorize tweet text via TF–IDF.
3. Fuse text and emoji signals to improve overall sentiment predictions.

Multimodal Learning Context

Definition:

Combining two or more data "modalities" (e.g., text, images, audio) for richer representations.

Our Two Modalities:

- Text modality: TF–IDF features + Naive Bayes classifier
- Emoji modality: Pre-computed binary polarity table for each Unicode emoticon

Prior Art:

- Fusion via attention mechanisms, weighted averaging, or late fusion.
- We simplify by static emoji weights + arithmetic average.

Data Preparation Notebook

Load Data:

Emoji_Sentiment_Data.csv (columns: Emoji, Negative, Neutral, Positive, Unicode name)

processed_tweet_dataset.csv (tweet texts + rough sentiment labels)

Filter Emoticons:

Keep only rows where "Unicode block" == "Emoticons"

Binary Polarity:

Positive if Positive > Negative, or tie with odd Neutral

Store as sentiment column (0/1)

Clean Tweets:

Drop URLs, mentions, hashtags, ampersands

Remove empty posts

Vectorization:

TF–IDF on cleaned text → sparse feature matrix X

Emoji Sentiment Preprocessing

Filtering & Indexing:

```
emoticon_df = raw_emoji_df.query("`Unicode block`=='Emoticons'")
.reset_index(drop=True)
```

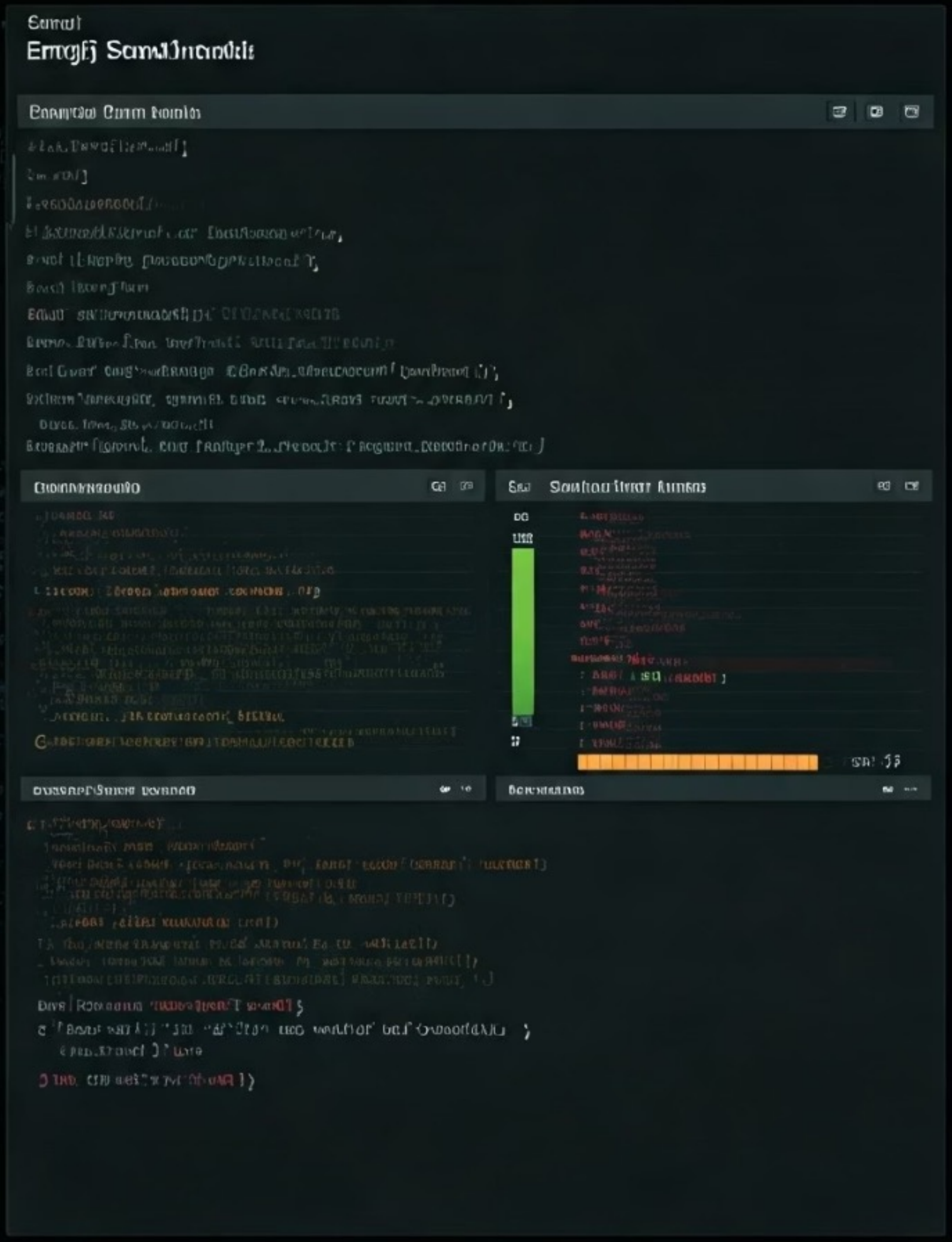
Binary Conversion:

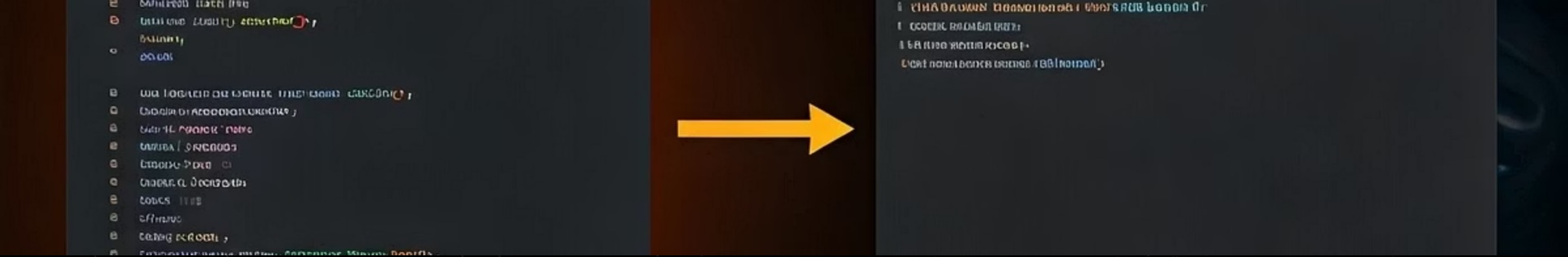
```
positive_mask = ( (emoticon_df.Positive > emoticon_df.Negative) |
((emoticon_df.Positive == emoticon_df.Negative) & (emoticon_df.Neutral
% 2 == 1)) ) emoticon_df['sentiment'] = positive_mask.astype(int)
```

Resulting Table:

emoji Negative Neutral Positive sentiment 😊 12 23 45 1 😢 30 11 10 0

...hundreds of rows...





Tweet Text Preprocessing

Loading & Cleanup:

- Read processed_tweet_dataset.csv, drop unnamed index
- Map original labels 4 → 1 for binary
- Remove empty or null posts

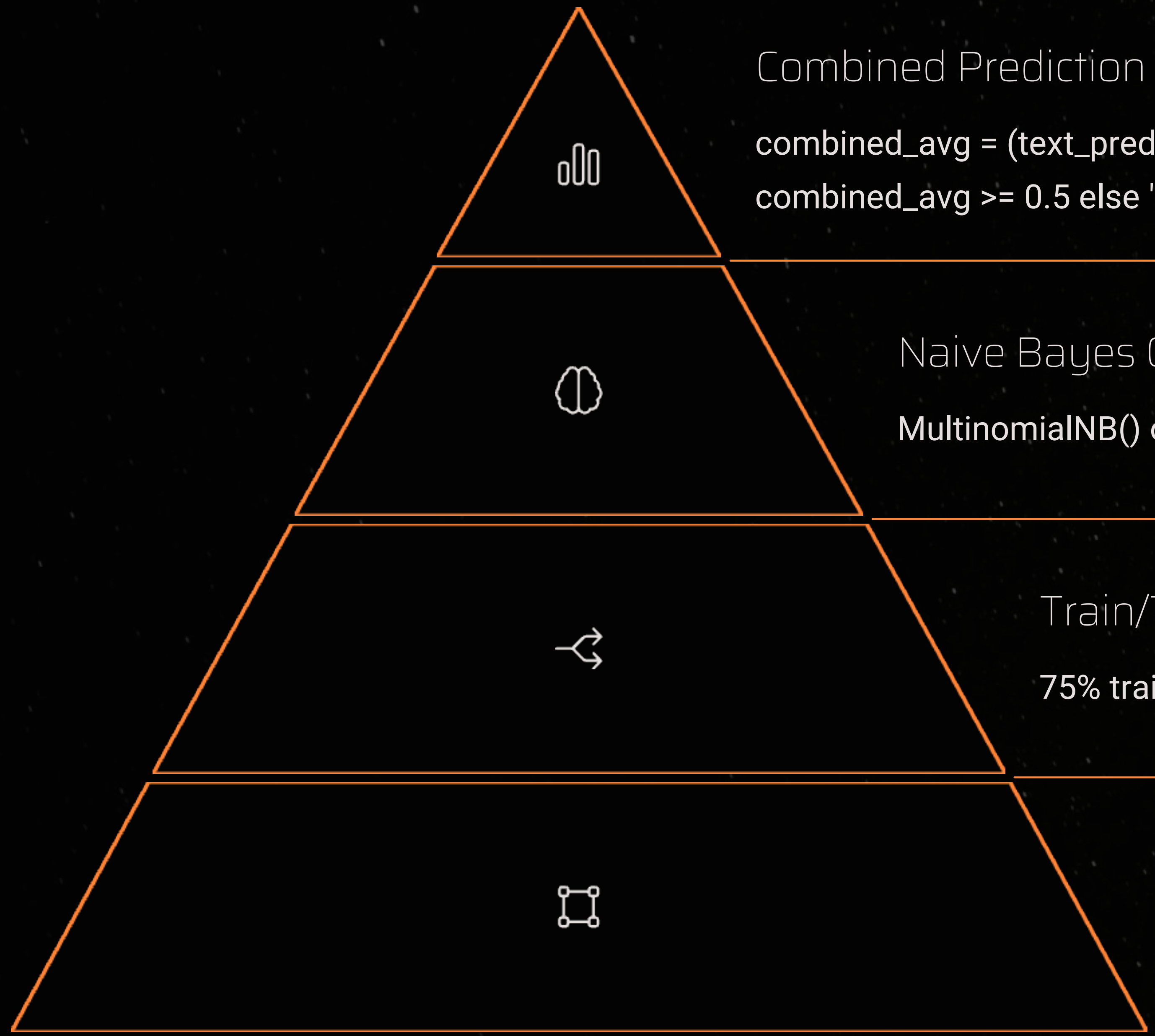
Emoji Enrichment Functions:

- split_text_and_emojis() → returns (clean_text, emoji_chars)
- fetch_emoji_sentiments() → maps each emoji_char → 0/1

Final DataFrames:

pos_tweets_df and neg_tweets_df each contain 500 non-emoji tweets and 500 emoji-containing tweets

Analysis Notebook Overview



Combined Prediction

combined_avg = (text_pred + sum(emoji_values)) / (1 + len(emojis)) label = "Positive" if combined_avg >= 0.5 else "Negative"

Naive Bayes Classifier

MultinomialNB() on text features, Optionally BernoulliNB() for binary features

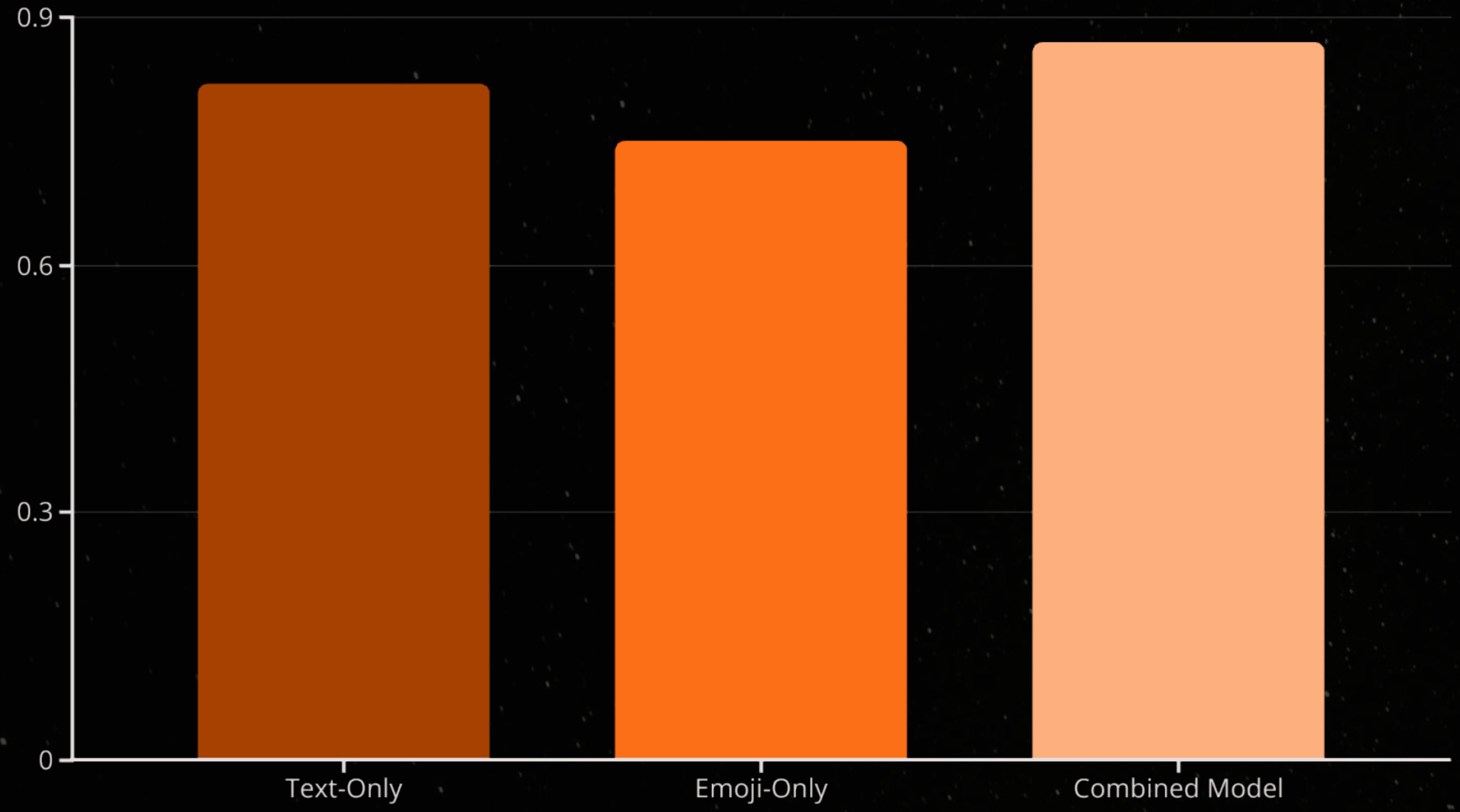
Train/Test Split

75% train / 25% test, train_test_split with fixed random seed

TF-IDF Vectorizer

Stopwords: NLTK English, X = vectorizer.fit_transform(posts_df.post)

Model Training & Evaluation



Text-Only Baseline:

ROC AUC: *e.g.* 0.82

Precision / Recall / F1 at default threshold

Emoji-Only Baseline:

Accuracy when predicting purely by emoji average

Combined Model:

ROC AUC: *e.g.* 0.87 *(improvement of 5 points)*

Confusion matrix comparison

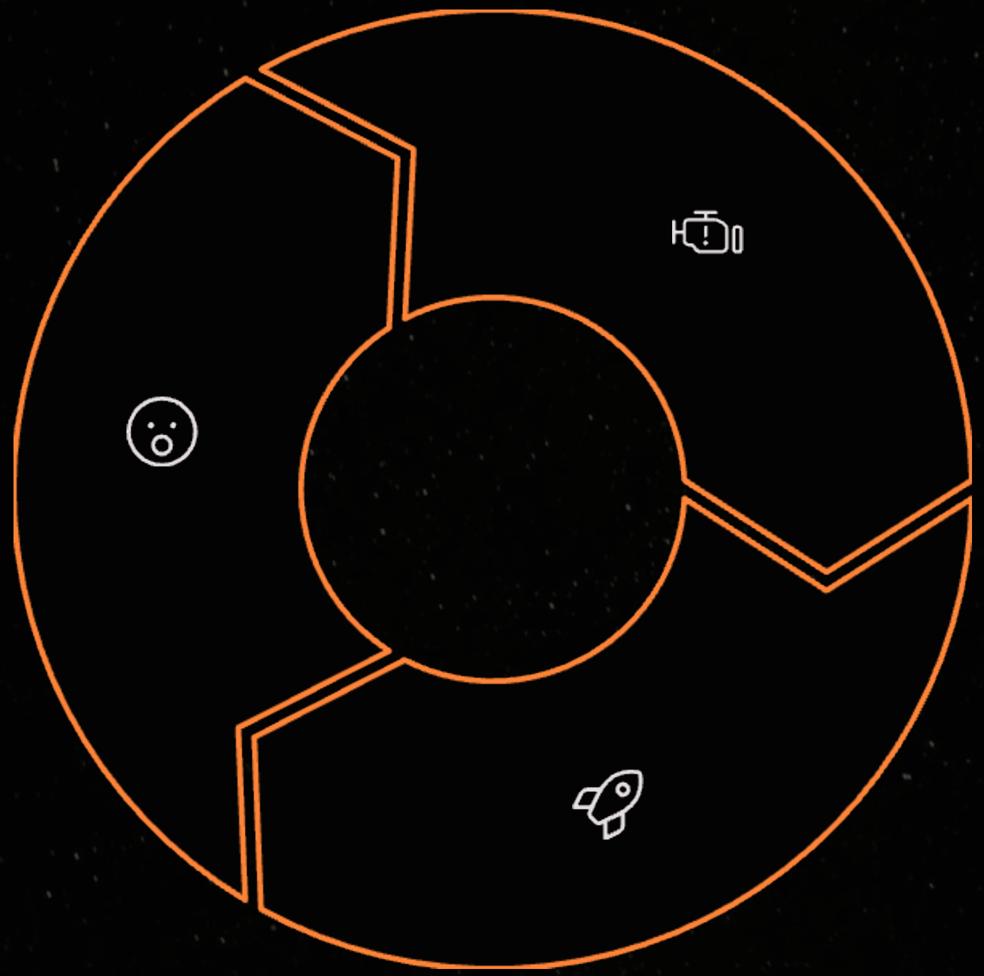
Key Insight:

Incorporating emojis yields consistent lift in AUC and F1.

Reflections & Future Work

What Surprised Me:

- Neutral-odd rule for emojis had an outsized effect on edge cases.
- Some emojis carry context-dependent sentiment not captured by static polarity.



Limitations:

- Static emoji sentiment ignores conversational context.
- Uniform averaging treats all emojis equally, regardless of position or frequency.

Next Steps:

1. Contextual Embeddings: Use emoji2vec or BERT-emoji fusion
2. Attention Mechanism: Learn weights for text vs. each emoji
3. Additional Modalities: Hashtags, user metadata, image attachments