Kaggle Twitter Sentiment Classification Project

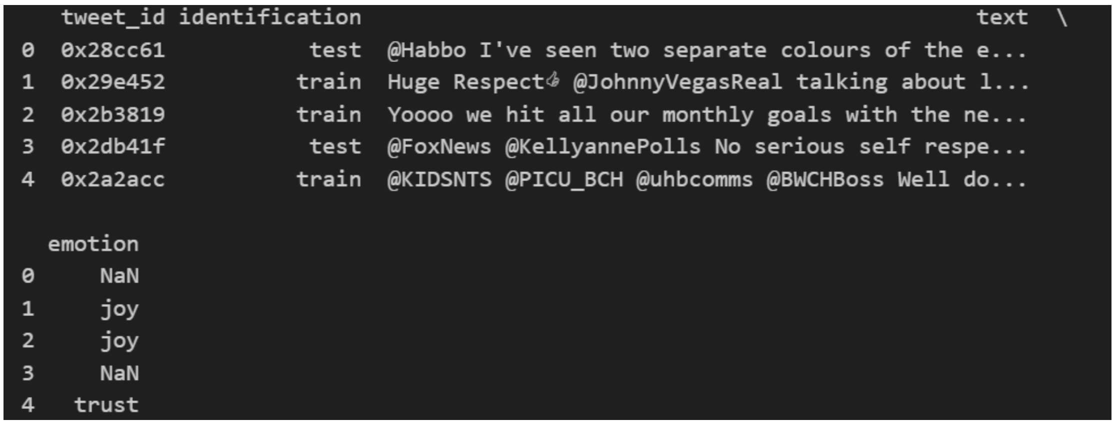
110704054 謝弘廷

(1) Loading and Merging Data

I began by loading the required datasets. I imported two CSV files using pandas: data_identification.csv, which maps tweet IDs and their identification type (train or test), and emotion.csv, which contains emotion labels for the tweets. I also processed a JSON file (tweets_DM.json) that stores the tweet content.

To handle the JSON file, I opened it line by line, parsing each JSON object using json.loads. From there, I extracted the tweet_id and the tweet's text, appending them to a list. After processing all the lines, I converted this list into a DataFrame for easier handling.

Finally, I merged these three datasets (df_id, df_emotion, and the tweets DataFrame) using tweet_id as the key. I used a left join to ensure that all rows from the primary dataset were preserved, even if some tweets or labels were missing.



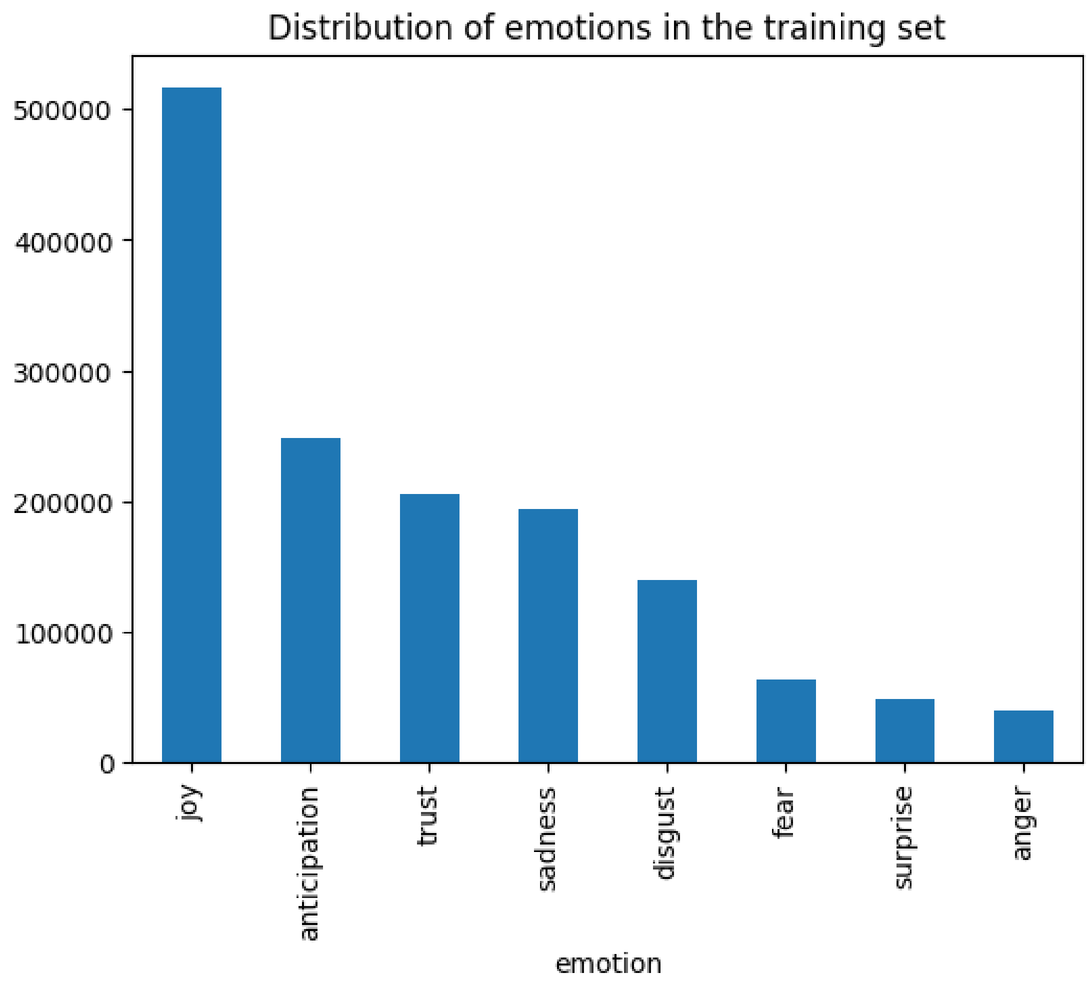
```
  tweet_id identification                                              text  \
0  0x28cc61           test  @Habbo I've seen two separate colours of the e...
1  0x29e452          train  Huge Respect👌 @JohnnyVegasReal talking about l...
2  0x2b3819          train  Yoooo we hit all our monthly goals with the ne...
3  0x2db41f           test  @FoxNews @KellyannePolls No serious self respe...
4  0x2a2acc          train  @KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...

  emotion
0     NaN
1     joy
2     joy
3     NaN
4   trust
```

(2) Exploratory Data Analysis (EDA)

Next, I conducted a simple EDA to understand the dataset better. I checked the number of rows, column types, and any missing values. This step was crucial to ensure data integrity before proceeding further.

I also explored the distribution of the identification column to see how many records were for training versus testing. Then, focusing on the training data, I filtered out rows without emotion labels and analyzed the distribution of emotions. I used a bar chart with Matplotlib to visualize this distribution, which helped me quickly grasp how balanced or imbalanced the dataset was.

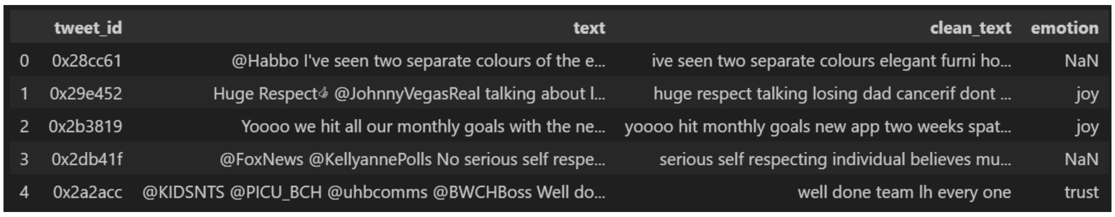


(3) Text Preprocessing and Cleaning

For text cleaning, I created a function called clean_text. Here's how I approached it:

1. Converted all text to lowercase for consistency.
2. Removed URLs and mentions (like @username) since they don't add value to emotion prediction.
3. Stripped punctuation, numbers, and special characters, retaining only alphabetical characters and spaces.
4. Compressed multiple spaces into single spaces.
5. Tokenized the text into individual words using nltk's word_tokenize and filtered out stopwords (common words like "and," and "the") to reduce noise.

This process ensured that the cleaned text retained only meaningful words, improving the model's performance later.



	tweet_id	text	clean_text	emotion
0	0x28cc61	@Habbo I've seen two separate colours of the e...	ive seen two separate colours elegant furni ho...	NaN
1	0x29e452	Huge Respect👍 @JohnnyVegasReal talking about l...	huge respect talking losing dad cancerif dont ...	joy
2	0x2b3819	Yoooo we hit all our monthly goals with the ne...	yoooo hit monthly goals new app two weeks spat...	joy
3	0x2db41f	@FoxNews @KellyannePolls No serious self respe...	serious self respecting individual believes mu...	NaN
4	0x2a2acc	@KIDSNTS @PICU_BCH @uhbcomms @BWCHBoss Well do...	well done team lh every one	trust

(4) Feature Engineering

For feature extraction, I used TF-IDF vectorization to transform the cleaned text into numerical features. I configured the vectorizer to include unigrams and bigrams, allowing the model to capture both single words and short phrases. I also set an upper limit on the number of features to manage memory and computation.

(5) Model Building and Validat**ion**

For the baseline model, I chose logistic regression due to its simplicity and reliability. I split the training data into a smaller training set and a validation set, ensuring balanced class distribution using stratified sampling. The logistic regression model was trained on the TF-IDF vectors and evaluated on the validation set using metrics like precision, recall, and F1-score. This provided a benchmark for how well the model could distinguish between emotion classes.

(6) Final Model and Submission

Once I validated the model, I retrained it on the full training set to maximize its performance. Using the final model, I predicted emotion labels for the test set. These predictions, along with the tweet IDs, were saved to a CSV file in the required submission format. By following this process, I ensured that the model was trained on all available labeled data and that the predictions were formatted correctly for a Kaggle-style competition.