1. Load Data

gift_df = pd.read_csv("amazon_reviews_us_Gift_Card_v1_00.tsv", sep="\t")
empathetic_df = pd.read_csv("emotion-emotion_69k.csv")

Two primary data sources are loaded:

- Amazon gift card reviews for sentiment analysis
- Empathetic dialogues for emotion detection and response generation

2. Column Selection and Renaming

empathetic_df = empathetic_df[['Situation', 'emotion', 'empathetic_dialogues', 'labels']]
empathetic_df.rename(columns={...}, inplace=True)

Selected only relevant columns and renamed them for consistency (situation, emotion, text, target).

3. Text Cleaning

def clean_text(text):
*   ...*
empathetic_df['text'] = ...

Applied a regex-based cleaning function to:

- Remove HTML
- Strip links
- Remove non-alphabetic characters
- Convert to lowercase

4. Filter Valid Emotion Labels

valid_emotions = {...}
empathetic_df = empathetic_df[empathetic_df['emotion'].isin(valid_emotions)]

Retained only samples with predefined emotion labels to ensure clean, label-consistent training data.

5. Visualize Emotion Label Distribution

sns.countplot(...)

Displayed the sample distribution across 30+ fine-grained emotion labels.

6. Merge Emotions into 8 Major Groups

emotion_map = {...}
empathetic_df['emotion_group'] = empathetic_df['emotion'].map(emotion_map)

Merged fine-grained emotions into 8 main emotion groups (e.g., positive, negative, fear) to simplify classification and analysis.

7. Text Length Feature Creation

empathetic_df['text_len'] = empathetic_df['text'].apply(lambda x: len(x.split()))

Calculated the word/token count of each user utterance – useful for:

- Padding/truncating sequences
- Understanding length variance across emotions

8. Missing Value Check

empathetic_df.isnull().sum()

Identified and removed samples with missing emotion_group due to unmatched mappings.

9. Boxplot: Emotion Group vs Text Length

sns.boxplot(x='emotion_group', y='text_len', ...)

Visual comparison of text length distributions across emotion groups to highlight linguistic differences.

10. Correlation Heatmap

empathetic_df['target_encoded'] = LabelEncoder().fit_transform(...)
sns.heatmap(...)

Computed correlation between text_len and target (encoded), indicating whether emotional length influences intended response.

11. Top Words by Emotion Group

def generate_top_words_by_group(...)

Collected the most frequent words per emotion group to identify emotional keywords.

12. WordCloud Visualization

plot_group_wordcloud(...)

Generated and displayed separate word clouds for each emotion group using their top words – highly interpretable for analysis and presentations.