Feasibility Stage

In this stage, we set up the environment, load datasets, explore data structure, identify shared emotion labels, and run basic visualizations to understand class distributions and potential preprocessing needs.

```
!pip install -U "transformers==4.37.2" "peft==0.10.0" "accelerate==0.27.2" --quiet
```

```
import pandas as pd
import kagglehub
from datasets import load_dataset
import os
os.environ["WANDB_DISABLED"] = "true"
```

```
# Load the GoEmotions dataset from uploaded CSV
goemotions_df = pd.read_csv("go_emotions_dataset.csv")
print("GoEmotions Sample:")
print(goemotions_df.sample(1))
```

```
GoEmotions Sample:
             id                                               text  \
24482  edjin31  Except the ones trying to zipper by flying by ...

       example_very_unclear  admiration  amusement  anger  annoyance  \
24482                 False           0          0      0          0

       approval  caring  confusion  ...  love  nervousness  optimism  pride  \
24482         0       1          0  ...     0            0         0      0

       realization  relief  remorse  sadness  surprise  neutral
24482            0       0        0        0         0        0

[1 rows x 31 columns]
```

```
import tarfile
import os

# Extract the tar.gz archive
with tarfile.open("empatheticdialogues.tar.gz", "r:gz") as tar:
    tar.extractall("empatheticdialogues")

# Confirm extraction
print("Files extracted to:", os.listdir("empatheticdialogues"))
```

```
Files extracted to: ['empatheticdialogues']
```

```
# Load the Cleaned EmpatheticDialogues dataset with error skipping
empathetic_df = pd.read_csv("empatheticdialogues/empatheticdialogues/train.csv", on_bad_lines="skip")
print("EmpatheticDialogues Sample:")
print(empathetic_df.sample(1))
```

```
EmpatheticDialogues Sample:
                    conv_id  utterance_idx context  \
33807  hit:5408_conv:10816              2  caring

                                              prompt  speaker_idx  \
33807  My grandfather was diagnosed with cancer recen...          540

                              utterance     selfeval tags
33807  Oh my God_comma_ that's terrible.  5|5|5_4|4|4  NaN
```

```
# Show all emotion columns
emotion_columns = goemotions_df.columns[3:]  # skip 'id', 'text', 'example_very_unclear'
print("GoEmotions Emotion Labels:")
print(emotion_columns.tolist())

goemotions_label_counts = goemotions_df[emotion_columns].sum().sort_values(ascending=False)
print("\nTop 10 Emotions by Frequency:")
print(goemotions_label_counts.head(10))
```

```
GoEmotions Emotion Labels:
['admiration', 'amusement', 'anger', 'annoyance', 'approval', 'caring', 'confusion', 'curiosity', 'desire', 'disappointment'

Top 10 Emotions by Frequency:
neutral        55298
approval       17620
admiration     17131
annoyance      13618
gratitude      11625
disapproval    11424
curiosity       9692
amusement       9245
realization     8785
optimism        8715
dtype: int64
```

```
# Check unique labels in the 'context' column
print("EmpatheticDialogues Emotion Labels:")
print(empathetic_df['context'].unique())

print("\nLabel Frequencies:")
print(empathetic_df['context'].value_counts().head(10))
```

```
EmpatheticDialogues Emotion Labels:
['sentimental' 'afraid' 'proud' 'faithful' 'terrified' 'joyful' 'angry'
 'sad' 'jealous' 'grateful' 'prepared' 'embarrassed' 'excited' 'annoyed'
 'lonely' 'ashamed' 'guilty' 'surprised' 'nostalgic' 'confident' 'furious'
 'disappointed' 'caring' 'trusting' 'disgusted' 'anticipating' 'anxious'
 'hopeful' 'content' 'impressed' 'apprehensive' 'devastated']

Label Frequencies:
context
surprised    3956
excited      2935
angry        2740
proud        2719
annoyed      2642
sad          2634
afraid       2510
lonely       2503
grateful     2487
terrified    2487
Name: count, dtype: int64
```

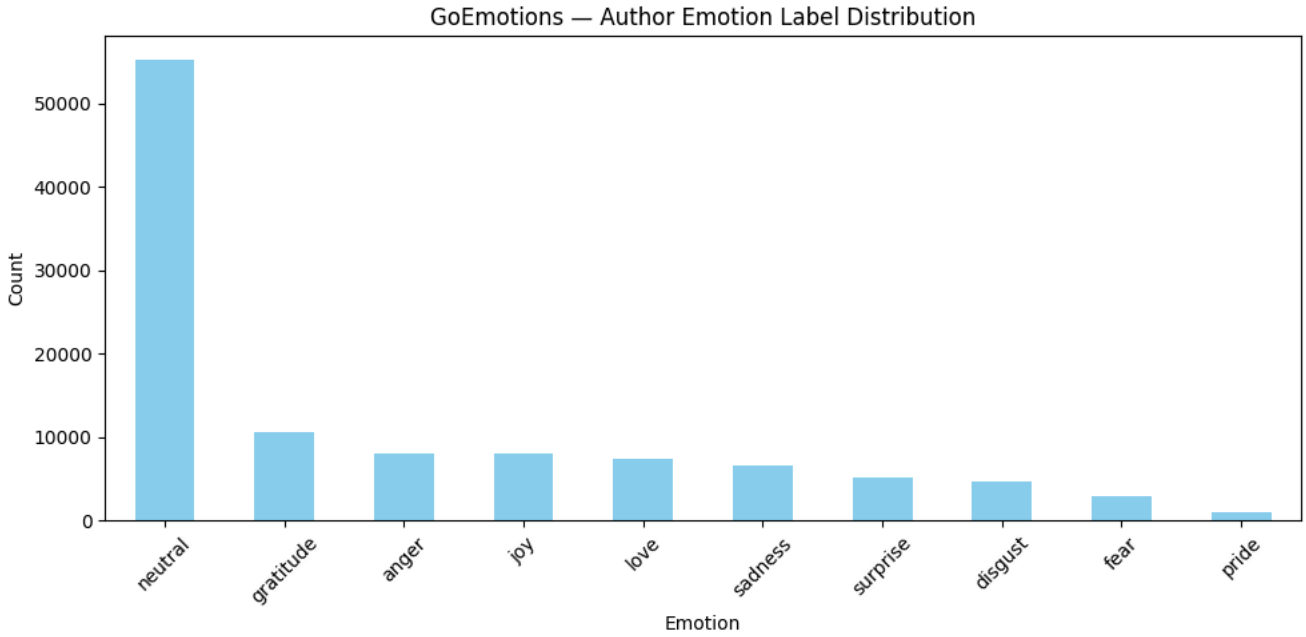
```
# Define shared label set
shared_emotions = [
    "joy", "anger", "sadness", "fear", "surprise",
    "disgust", "love", "gratitude", "neutral", "pride"
]

# Filter only rows with at least one shared emotion
filtered = goemotions_df[goemotions_df[shared_emotions].sum(axis=1) > 0].copy()

# For each row, pick the highest scoring (dominant) emotion
filtered["label"] = filtered[shared_emotions].idxmax(axis=1)

# Keep only text and new label
goemotions_clean = filtered[["text", "label"]].copy()
print("GoEmotions (Cleaned) Sample:")
print(goemotions_clean.sample(3))
```

```
GoEmotions (Cleaned) Sample:
                                                     text      label
30815                             What a time to be alive        joy
205185           Thanks for reminding me not to have kids  gratitude
12140   I've been getting it off from the West Wing si...    neutral
```

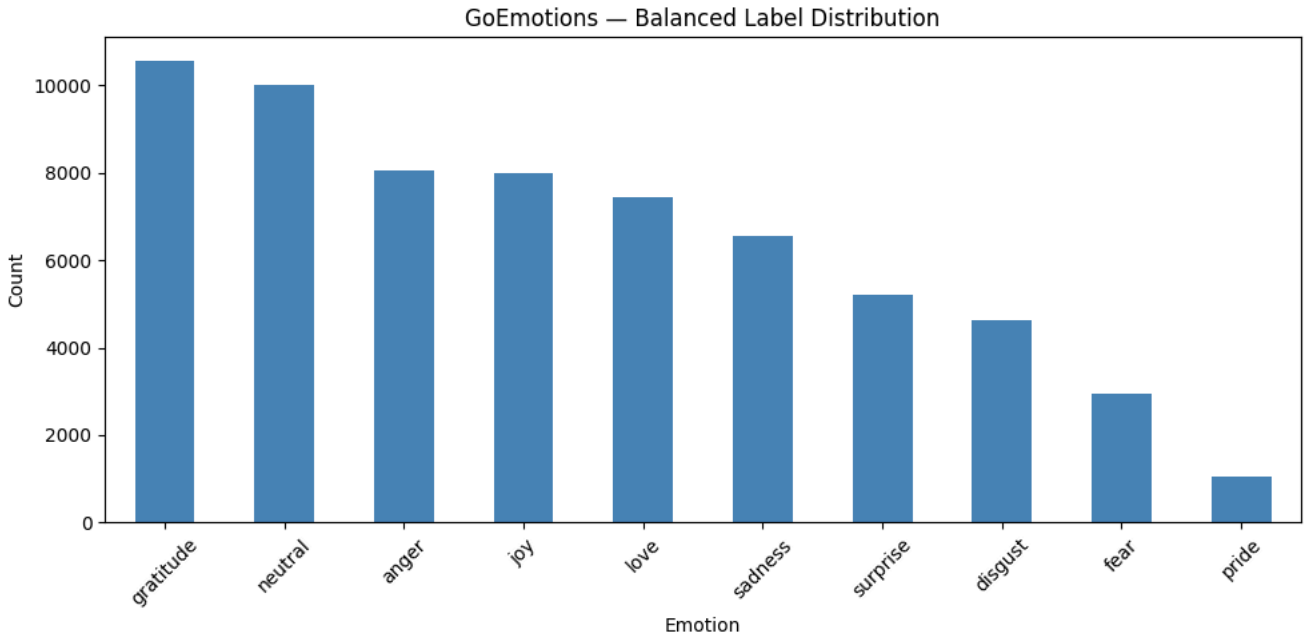
```
# Define mapping from EmpatheticDialogues context → shared emotions
context_mapping = {
    "joyful": "joy",
    "excited": "joy",
    "content": "joy",
    "hopeful": "joy",

    "proud": "pride",
    "grateful": "gratitude",

    "angry": "anger",
    "furious": "anger",
    "annoyed": "anger",

    "afraid": "fear",
    "terrified": "fear",
    "apprehensive": "fear",

    "sad": "sadness",
    "lonely": "sadness",
    "devastated": "sadness",

    "surprised": "surprise",

    "disgusted": "disgust",

    "sentimental": "love",
    "nostalgic": "love"
}

# Map the context column
empathetic_df["label"] = empathetic_df["context"].map(context_mapping)

# Drop rows where mapping failed
empathetic_clean = empathetic_df.dropna(subset=["label"])[["utterance", "label"]].rename(columns={
    "utterance": "text"
})

# Preview sample
print("EmpatheticDialogues (Cleaned) Sample:")
print(empathetic_clean.sample(3))
```

```
EmpatheticDialogues (Cleaned) Sample:
                                                    text label
34481  It is worth a look for the new one_comma_ it l...   joy
43130  You have done such a great job! You should be ...  fear
25868            it was late and the street wasnt lit up  fear
```

```
import matplotlib.pyplot as plt

# Plot label distribution for GoEmotions
goemotions_clean["label"].value_counts().plot(kind="bar", figsize=(10,5), color="skyblue")
plt.title("GoEmotions — Author Emotion Label Distribution")
plt.xlabel("Emotion")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



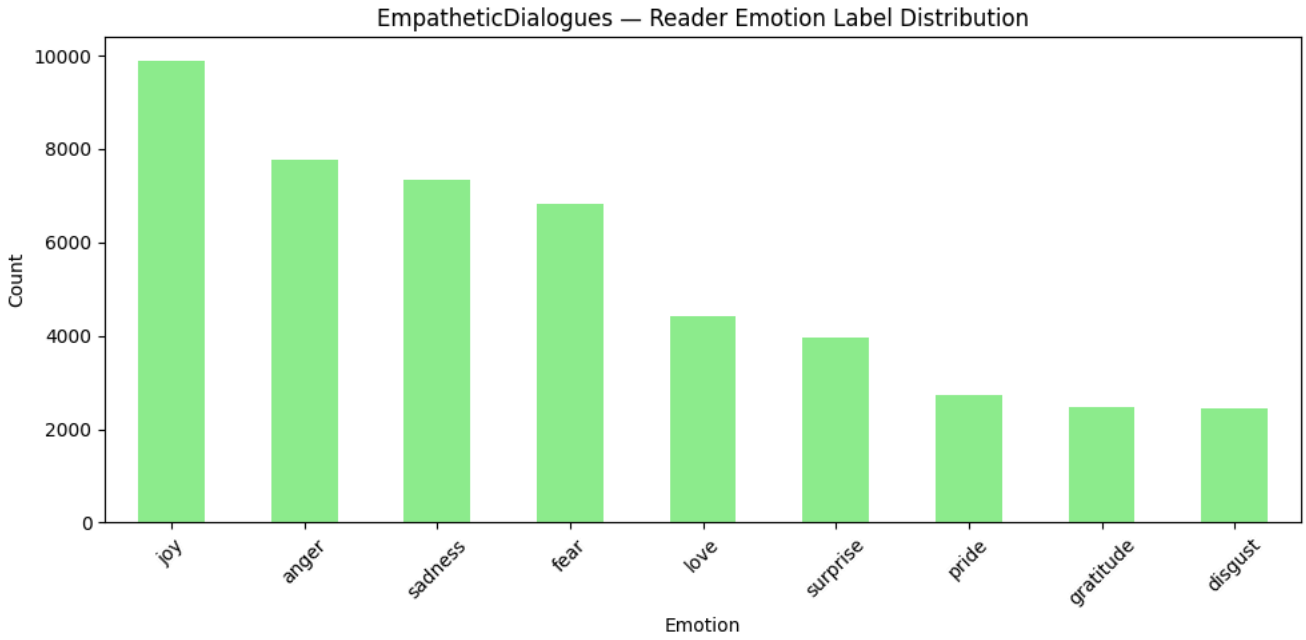

```
# Cap for "neutral"
neutral_cap = 10000

# Split neutral and non-neutral
neutral_rows = goemotions_clean[goemotions_clean["label"] == "neutral"].sample(n=neutral_cap, random_state=42)
non_neutral_rows = goemotions_clean[goemotions_clean["label"] != "neutral"]

# Combine and shuffle
goemotions_balanced = pd.concat([neutral_rows, non_neutral_rows])
goemotions_balanced = goemotions_balanced.sample(frac=1, random_state=42).reset_index(drop=True)

goemotions_clean = goemotions_balanced

# Plot to confirm
goemotions_clean["label"].value_counts().plot(kind="bar", figsize=(10,5), color="steelblue")
plt.title("GoEmotions — Balanced Label Distribution")
plt.xlabel("Emotion")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Balancing the Dataset

The "neutral" label was heavily overrepresented, with over 55k samples. To avoid bias in our model, we capped it at 10,000 and kept all other labels unchanged. Then we shuffled the combined data to remove any ordering. The updated plot shows a much more balanced distribution, which helps ensure fairer learning across all emotion categories.

```
# Plot label distribution for EmpatheticDialogues
empathetic_clean["label"].value_counts().plot(kind="bar", figsize=(10,5), color="lightgreen")
plt.title("EmpatheticDialogues — Reader Emotion Label Distribution")
plt.xlabel("Emotion")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```
# Show one example per emotion for GoEmotions
print("GoEmotions: One Example Per Emotion\n")
for label in goemotions_clean["label"].unique():
    example = goemotions_clean[goemotions_clean["label"] == label]["text"].iloc[0]
    print(f"{label.upper()}: {example}")
```

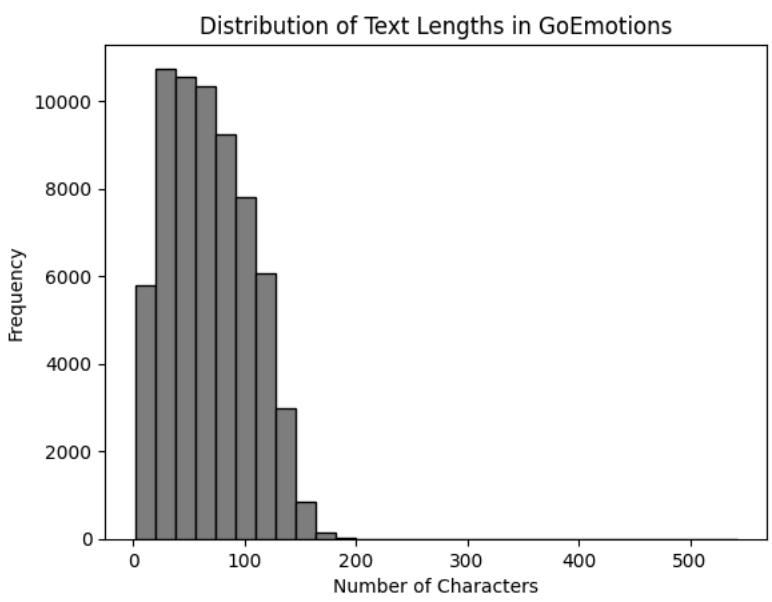
```
GoEmotions: One Example Per Emotion

NEUTRAL: Oh, [NAME], don't bring THAT story up!
GRATITUDE: Thank you so much! That’s a helpful list with a great variety!
LOVE: I like this rule because in life there are always exceptions. The problem is when you start feeling entitled to *being
DISGUST: I’m always weirded out whenever I see him without a beard because in my game I got the longest beard I could
JOY: Im glad youre going to be a perfect parent who never makes any mistakes. Make sure you write a book!
SURPRISE: Really? I thought this joke was a bit plane
ANGER: [NAME] is so pathetic, 40 year old man jealous of a 26 year old DJ
SADNESS: I’m sorry for your loss bro ❤️
PRIDE: No problem! I've been chillin in capture the flag for couple of days now and she is super strong and fun in this mode
FEAR: My dosage was increased 2 weeks ago and I've noticed the issues getting worse. With your comment, I'm really leaning t
```

```
# Show one example per emotion for EmpatheticDialogues
print("\nEmpatheticDialogues: One Example Per Emotion\n")
for label in empathetic_clean["label"].unique():
    example = empathetic_clean[empathetic_clean["label"] == label]["text"].iloc[0]
    print(f"{label.upper()}: {example}")
```

```
EmpatheticDialogues: One Example Per Emotion

LOVE: I remember going to see the fireworks with my best friend. It was the first time we ever spent time alone together. Al
FEAR:  it feels like hitting to blank wall when i see the darkness
```

```
PRIDE: Hi how are you doing today
JOY: Hi_comma_ this year_comma_ I was the first over 300 students at my enginering school
ANGER: I lost my job last year and got really angry.
SADNESS: During christmas a few years ago_comma_ I did not get any presents.
GRATITUDE: Hi_comma_ I went to a park and I set on a bench. I didn't notice that my wallet felt. A man came to me from behin
SURPRISE: When I was working my first job_comma_ my parents picked me up in my new car_comma_ I was very surprised_comma_ i
DISGUST: I hate to be tired and see a store crowded with customers and only one or two checkouts open.
```

```
# Plot distribution of text lengths for GoEmotions
goemotions_clean["length"] = goemotions_clean["text"].apply(len)
goemotions_clean["length"].plot(kind="hist", bins=30, color="gray", edgecolor="black")
plt.title("Distribution of Text Lengths in GoEmotions")
plt.xlabel("Number of Characters")
plt.ylabel("Frequency")
plt.show()
```



Prototype Stage

Here, we implement the core model pipeline: encoding labels, tokenizing text, creating dataset classes, loading a base transformer model, and running initial training and evaluation to test feasibility at scale.

```
# Encode labels
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# Drop missing or non-string labels
goemotions_clean = goemotions_clean[goemotions_clean["label"].notna()]
goemotions_clean = goemotions_clean[goemotions_clean["label"].apply(lambda x: isinstance(x, str))]

# Encode the cleaned labels
label_encoder = LabelEncoder()
goemotions_clean["label_encoded"] = label_encoder.fit_transform(goemotions_clean["label"])

# Split into train/test sets
g_train_texts, g_test_texts, g_train_labels, g_test_labels = train_test_split(
    goemotions_clean["text"],
    goemotions_clean["label_encoded"],
    test_size=0.2,
    random_state=42,
    stratify=goemotions_clean["label_encoded"]
)

# Reset index and convert labels to NumPy arrays
g_train_labels = g_train_labels.reset_index(drop=True).to_numpy()
g_test_labels = g_test_labels.reset_index(drop=True).to_numpy()

from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("BAAI/bge-small-en")
```