Cell 1: Install Libraries

```
## Notebook for Testing Fine-tuned Sentiment Model

# @title Cell 1: Install Libraries
!pip install transformers torch pandas emoji -q
print("Libraries installed.")
```

```
Libraries installed.
```

Cell 2: Import Libraries

```
# @title Cell 2: Import Libraries
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch
import os
import zipfile
import pandas as pd
import emoji
from google.colab import drive

print("Libraries imported.")
```

```
Libraries imported.
```

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

Cell 3: Get Model from Google Drive

```
# @title Cell 3: Get Model from Google Drive

# --- Mount Google Drive ---
try:
    print("Mounting Google Drive...")
    drive.mount('/content/drive', force_remount=True) # force_remount can help if Drive connection issues occur
    print("Google Drive mounted successfully.")
except Exception as e:
    print(f"Error mounting Google Drive: {e}")
    raise SystemExit("Mounting failed.")

# --- *** IMPORTANT: SET THIS PATH *** ---
# Path to the ZIP file WITHIN your Google Drive.
# Example: /content/drive/MyDrive/Model07/your_model_file.zip
zip_path_in_drive = "/content/drive/MyDrive/Model07/distilbert-base-uncased_50000subset_3epochs.zip" # <<< CHANGE to the correct path in

# Path where the unzipped model folder should be created IN COLAB
# This path will be used in the next cell to load the model
model_path_in_colab = "/content/fine_tuned_sentiment_model"
# --- ---

# --- Unzip the model from Drive to Colab ---
try:
    if not os.path.exists(zip_path_in_drive):
        print(f"ERROR: Zip file not found at: {zip_path_in_drive}")
        print("Please ensure you saved a copy to your Drive and the path is correct.")
        raise FileNotFoundError("Zip file not found in Drive.")

    print(f"Unzipping model from {zip_path_in_drive} to {model_path_in_colab}...")

    # Create target directory if it doesn't exist
    os.makedirs(model_path_in_colab, exist_ok=True)

    # Unzip using the zipfile module for better control
    with zipfile.ZipFile(zip_path_in_drive, 'r') as zip_ref:
        zip_ref.extractall(model_path_in_colab)

    # --- Check if the expected output directory exists after unzipping ---
    extracted_folders = [f for f in os.listdir(model_path_in_colab) if os.path.isdir(os.path.join(model_path_in_colab, f))]
    if len(extracted_folders) == 1:
```

```
        # Assume the zip contained one folder with the model files
        potential_model_path = os.path.join(model_path_in_colab, extracted_folders[0])
        # Check if this subdirectory actually contains the config file
        if os.path.exists(os.path.join(potential_model_path, "config.json")):
             model_path_in_colab = potential_model_path
             print(f"Model files found in subdirectory: {model_path_in_colab}")
    elif not os.path.exists(os.path.join(model_path_in_colab, "config.json")):
         print(f"WARNING: config.json not found directly in {model_path_in_colab}.")
         print("Please check the unzipped contents and adjust 'model_path_in_colab' if needed before running the next cell.")
         print(f"Contents of {model_path_in_colab}: {os.listdir(model_path_in_colab)}")


    print(f"Model successfully unzipped to: {model_path_in_colab}")

except FileNotFoundError:
    # Error message printed above
    pass
except Exception as e:
    print(f"An error occurred during unzipping: {e}")
    raise SystemExit("Unzipping failed.")
```

```
Mounting Google Drive...
Mounted at /content/drive
Google Drive mounted successfully.
Unzipping model from /content/drive/MyDrive/Model07/distilbert-base-uncased_50000subset_3epochs.zip to /content/fine_tuned_sentiment_mod
Model files found in subdirectory: /content/fine_tuned_sentiment_model/sentiment_model_amazon_csv_finetuned
Model successfully unzipped to: /content/fine_tuned_sentiment_model/sentiment_model_amazon_csv_finetuned
```

Cell 4: Load Model and Tokenizer

```
# @title Cell 4: Load Model and Tokenizer

print("--- Loading Model & Tokenizer ---")

if 'model_path_in_colab' not in locals():
    # Default if Cell 3 wasn't run or variable got lost - adjust as needed
    model_path_in_colab = "/content/fine_tuned_sentiment_model/sentiment_model_amazon_csv_finetuned" # Example path
    print(f"Warning: 'model_path_in_colab' not found, defaulting to {model_path_in_colab}. Ensure this is correct.")
    # raise SystemExit("Variable 'model_path_in_colab' not set. Please run Cell 3 first.") # Option to halt instead

saved_model_path = model_path_in_colab
# --- ---

try:
    if not os.path.isdir(saved_model_path):
        print(f"ERROR: Directory not found: {saved_model_path}")
        print("Please ensure Cell 3 ran correctly, unzipped the file, and set the path correctly.")
        raise FileNotFoundError("Model directory not found.")

    print(f"Loading tokenizer from: {saved_model_path}")
    tokenizer = AutoTokenizer.from_pretrained(saved_model_path)

    print(f"Loading model from: {saved_model_path}")
    model = AutoModelForSequenceClassification.from_pretrained(saved_model_path)

    # Check if GPU is available and move model
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model.to(device)
    print(f"Model moved to device: {device}")

    # Set model to evaluation mode
    model.eval()
    print("Model and tokenizer loaded successfully.")

except FileNotFoundError:
     # Error message printed above
     raise SystemExit("Loading failed.")
except Exception as e:
    print(f"An error occurred loading the model/tokenizer: {e}")
    print(f"Please check if the path '{saved_model_path}' contains the necessary model files (config.json, model weights, tokenizer file
    raise SystemExit("Loading failed.")
```

```
--- Loading Model & Tokenizer ---
Loading tokenizer from: /content/fine_tuned_sentiment_model/sentiment_model_amazon_csv_finetuned
Loading model from: /content/fine_tuned_sentiment_model/sentiment_model_amazon_csv_finetuned
Model moved to device: cpu
```

```
Model and tokenizer loaded successfully.
```

Cell 5: Prediction Function (Emojis, No Probabilities - Modified)

```python
# @title Cell 5: Prediction Function (Emojis, No Probabilities - Modified)

# Define sentiment mapping (using the Amazon model's 1-5 score)
sentiment_map = {1: "Score 1 (Very Negative)", 2: "Score 2 (Negative)", 3: "Score 3 (Neutral)", 4: "Score 4 (Positive)", 5: "Score 5 (Ve

# MODIFIED: Function now takes model and tokenizer as input arguments
# MODIFIED: Added emoji conversion step
# MODIFIED: REMOVED probability calculation and printing
def predict_sentiment(text, model_to_use, tokenizer_to_use):
    """Converts emojis, tokenizes text, predicts sentiment, and returns score/label."""

    print(f"\nOriginal Input Text: '{text}'")
    # Basic check for valid text input
    if not isinstance(text, str) or not text.strip():
        print("Invalid input text provided.")
        return None, None
    try:
        # --- Convert emojis to text aliases ---
        text_no_emoji = emoji.demojize(text, language='alias')
        if text != text_no_emoji:
             print(f"Text after demojize: '{text_no_emoji}'") # Show converted text if emojis were present
        # --- ---

        # Tokenize using the provided tokenizer
        inputs = tokenizer_to_use(text_no_emoji, return_tensors="pt", truncation=True, padding=True, max_length=512)

        # Move inputs to the same device as the provided model
        inputs = {k: v.to(model_to_use.device) for k, v in inputs.items()}

        # Perform prediction using the provided model
        with torch.no_grad(): # Disable gradient calculation for inference
            logits = model_to_use(**inputs).logits

        # Get the predicted class index
        predicted_class_id = torch.argmax(logits, dim=-1).item() # Argmax directly on logits

        # Map back to original sentiment score (0-4 -> 1-5)
        predicted_sentiment_score = predicted_class_id + 1
        predicted_label = sentiment_map.get(predicted_sentiment_score, 'Unknown')

        print(f"Predicted Sentiment Score (1-5): {predicted_sentiment_score}")
        print(f"Predicted Sentiment Label: {predicted_label}")

        # --- Probabilities section removed ---

        return predicted_sentiment_score, predicted_label

    except Exception as e:
        print(f"An error occurred during prediction: {e}")
        return None, None

print("Prediction function defined (converts emojis, requires model/tokenizer arguments")
```

```
Prediction function defined (converts emojis, requires model/tokenizer arguments
```

Cell 5b: Load Test Dataset (Modified)

```python
# @title Cell 5b: Load Test Dataset (Modified)

# --- Configuration ---
test_csv_path = "/content/Social Media comments.csv"

test_text_column = "Text"

num_samples_to_test = 10

test_samples = []
df_test = None

# --- Inspect and Load Test Data ---
print(f"--- Processing Test Data ---")
```

```
try:
    # Ensure pandas (pd) is available (Cell 2 should have run)
    if 'pd' not in globals(): raise NameError("'pd' is not defined. Please run Cell 2 first.")

    if os.path.exists(test_csv_path):
        print(f"Inspecting Test data: {test_csv_path}")
        df_test_head = pd.read_csv(test_csv_path, nrows=5)
        print("Test Data Columns:", df_test_head.columns.tolist())
        print("Test Data Head:\n", df_test_head.head())
        print("-" * 30)

        if test_text_column not in df_test_head.columns:
             print(f"ERROR: Column '{test_text_column}' not found in {test_csv_path}.")
             print("Please UPDATE 'test_text_column' in this cell and rerun.")
        else:
            print(f"Loading Test samples from column '{test_text_column}'...")
            # Load full file (or handle large files differently if needed)
            df_test = pd.read_csv(test_csv_path)
            df_test = df_test.dropna(subset=[test_text_column])
            # Take random samples
            test_samples = df_test[test_text_column].sample(n=min(num_samples_to_test, len(df_test)), random_state=101).tolist()
            print(f"Loaded {len(test_samples)} Test samples.")
    else:
        print(f"Warning: Test file not found at {test_csv_path}")

except NameError as ne:
    print(f"ERROR: {ne}")
    print("Import failed? Run Cell 2.")
except FileNotFoundError:
    print(f"ERROR: File not found: {test_csv_path}")
except Exception as e:
    print(f"Error inspecting/loading Test data: {e}")


# --- Report loaded samples ---
print(f"\nTotal Test samples loaded: {len(test_samples)}")
if not test_samples:
     print("\nWarning: No samples loaded from the test dataset.")
print("Ready for Cell 6.")
```

```
--- Processing Test Data ---
Inspecting Test data: /content/Social Media comments.csv
Test Data Columns: ['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator', 'HelpfulnessDenominator', 'Time', 'Summary', 'Te
Test Data Head:
        Id   ProductId          UserId             ProfileName  \
0  165257  B000EVG8J2  A1L01D2BD3RKVO  B. Miller "pet person"
1  231466  B0000BXJIS  A3U62RE5XZDP0G                   Marty
2  427828  B008FHUFAU   AOXC0JQQZGGB6         Kenneth Shevlin
3  433955  B006BXV14E  A3PWPNZVMNX3PA             rareoopdvds
4   70261  B007I7Z3Z0  A1XNZ7PCE45KK7                 Og8ys1

   HelpfulnessNumerator  HelpfulnessDenominator        Time  \
0                     0                       0  1268179200
1                     0                       0  1298937600
2                     0                       2  1224028800
3                     0                       1  1335312000
4                     0                       2  1334707200

                                     Summary  \
0  Crunchy & Good Gluten-Free Sandwich Cookies!
1                           great kitty treats
2                                COFFEE TASTE
3              So the Mini-Wheats were too big?
4                           Great Taste . . .

                                          Text
0  Having tried a couple of other brands of glute...
1  My cat loves these treats. If ever I can't fin...
2  A little less than I expected.  It tends to ha...
3  First there was Frosted Mini-Wheats, in origin...
4  and I want to congratulate the graphic artist ...
------------------------------
Loading Test samples from column 'Text'...
Loaded 10 Test samples.

Total Test samples loaded: 10
Ready for Cell 6.
```

Cell 6: Test Predictions on Loaded Samples (Modified)

```
# @title Cell 6: Test Predictions on Loaded Samples (Modified)

print("\n--- Running Test Predictions on Loaded Samples ---")

# Check if required variables exist before proceeding
run_predictions = True
if 'test_samples' not in locals():
     print("ERROR: 'test_samples' list not found. Please run Cell 5b first.")
     run_predictions = False
elif not test_samples:
     print("Warning: 'test_samples' list is empty. No predictions to run.")
     run_predictions = False
elif 'predict_sentiment' not in globals():
     print("ERROR: predict_sentiment function not defined. Please run Cell 5 first.")
     run_predictions = False
elif 'model' not in locals() or model is None:
     print("ERROR: 'model' not loaded or is None. Please run Cell 4 successfully.")
     run_predictions = False
elif 'tokenizer' not in locals() or tokenizer is None:
     print("ERROR: 'tokenizer' not loaded or is None. Please run Cell 4 successfully.")
     run_predictions = False

if run_predictions:
    # Proceed if all checks pass
    print(f"Using loaded model and tokenizer for predictions on {len(test_samples)} samples...")
    for i, comment in enumerate(test_samples):
        print(f"\n--- Sample {i+1} ---")
        # Pass model and tokenizer to the function
        predict_sentiment(comment, model, tokenizer)
else:
    print("Cannot run predictions due to missing functions or variables.")


print("\n--- Testing Finished ---")
```

```
--- Running Test Predictions on Loaded Samples ---
Using loaded model and tokenizer for predictions on 10 samples...

--- Sample 1 ---

Original Input Text: 'I love coffee, I drink it two times per day/every day. I expected this coffee not to be so good because of a few
Predicted Sentiment Score (1-5): 4
Predicted Sentiment Label: Score 4 (Positive)

--- Sample 2 ---

Original Input Text: 'I read about it, I was really excited, I wanted so badly to like this stuff!<br />Truth is, it's just plain awfu
Predicted Sentiment Score (1-5): 1
Predicted Sentiment Label: Score 1 (Very Negative)

--- Sample 3 ---

Original Input Text: 'These are wonderful for newborns, who don't drink large amounts.  Small nipples, so they fit the newborn mouths
Predicted Sentiment Score (1-5): 5
Predicted Sentiment Label: Score 5 (Very Positive)

--- Sample 4 ---

Original Input Text: 'My first time experience Coconut Water. I heard it is good for inflammation. I am very happy with my choice of c
Predicted Sentiment Score (1-5): 5
Predicted Sentiment Label: Score 5 (Very Positive)

--- Sample 5 ---

Original Input Text: 'This is the strangest thing I've ever seen.  Imagine your roll on deodorant container  being filled with some ba
Predicted Sentiment Score (1-5): 1
Predicted Sentiment Label: Score 1 (Very Negative)

--- Sample 6 ---

Original Input Text: 'I'm not a big cereal eater, but I love this cereal.  It is perfect with Vanilla Coconut milk.  I bought it for m
Predicted Sentiment Score (1-5): 5
Predicted Sentiment Label: Score 5 (Very Positive)

--- Sample 7 ---

Original Input Text: 'My boxer likes these, but he doesn't get really excited about them.  But if they work as advertised and clean hi
```

```
Predicted Sentiment Score (1-5): 4
Predicted Sentiment Label: Score 4 (Positive)

--- Sample 8 ---

Original Input Text: 'Disgusting, yuck, no thanks! The first three words I thought of after biting into one of these chocolate fruit t
Predicted Sentiment Score (1-5): 1
Predicted Sentiment Label: Score 1 (Very Negative)

--- Sample 9 ---

Original Input Text: 'The Best Cat Litter For Your Money<br /><br />I don't buy this from Amazon as a certain chain brick 'n' mortar h
Predicted Sentiment Score (1-5): 5
Predicted Sentiment Label: Score 5 (Very Positive)
```

```
print("--- Manual Tests ---")
predict_sentiment("The product is okay, not great but not terrible either.", model, tokenizer)
predict_sentiment("I guess it's fine for the price.", model, tokenizer)
predict_sentiment("A bit disappointed with the quality.", model, tokenizer)
predict_sentiment("Service could be improved.", model, tokenizer)
```

```
--- Manual Tests ---

Original Input Text: 'The product is okay, not great but not terrible either.'
Predicted Sentiment Score (1-5): 3
Predicted Sentiment Label: Score 3 (Neutral)

Original Input Text: 'I guess it's fine for the price.'
Predicted Sentiment Score (1-5): 4
Predicted Sentiment Label: Score 4 (Positive)

Original Input Text: 'A bit disappointed with the quality.'
Predicted Sentiment Score (1-5): 3
Predicted Sentiment Label: Score 3 (Neutral)

Original Input Text: 'Service could be improved.'
Predicted Sentiment Score (1-5): 5
Predicted Sentiment Label: Score 5 (Very Positive)
(5, 'Score 5 (Very Positive)')
```