Sentiment Guessing Game:
Human vs. AI in Customer Experience Analytics

Sacred Heart University
Final Project

Presented by
Mohammed Haroon Palagiri
palagirim@mail.sacredheart.edu

1. Abstract

This project presents an interactive web-based sentiment analysis game built using Streamlit. The system turns a classic NLP task—classifying the sentiment of text—into a human vs AI competition. On each round, the application displays a review, asks the player to guess its sentiment, and compares that guess to an AI model's prediction and the true sentiment label from the dataset. The app tracks scores across multiple rounds and displays a final scoreboard and winner.

The solution is implemented using **Streamlit** for the user interface, **pandas** for dataset handling, and **TextBlob** (or similar sentiment engine) for AI predictions, as defined in the project's `requirements.txt`. The project demonstrates end-to-end skills in data processing, applied NLP, interactive UI development, and gamification of machine learning concepts.

2. Introduction & Motivation

Sentiment analysis is a core task in Natural Language Processing (NLP) with wide applications in business, such as understanding customer feedback, monitoring brand reputation, and improving products. However, sentiment analysis can also be subjective and sometimes even human readers disagree on whether a review is "positive," "neutral," or "negative."

Instead of building yet another dashboard that only shows metrics, this project aims to **make sentiment analysis experiential and fun**. The idea is to create a **game where humans compete with an AI model** to guess the sentiment of text reviews. This helps:

- Demonstrate how AI models interpret language.
- Show situations where humans and AI agree or disagree.
- Engage non-technical users and make ML concepts more intuitive.

3. Problem Statement

- Traditional sentiment analysis projects focus on building models and reporting accuracy metrics, but they often fail to engage end users.
  Problem:
- How can we make sentiment analysis more interactive and intuitive for end users while still showcasing core machine learning concepts?
  Goal:
- Build a **Streamlit sentiment game** where human players guess the sentiment of text reviews and compete against an AI sentiment model, with transparent feedback each round and a clear final scoreboard.

4. Objectives

1. **Create an interactive game interface** using Streamlit.

2. **Allow users to upload a review dataset** (e.g., CSV with text and sentiment labels).
3. **Randomly select a review each round** and show it to the player.
4. **Capture the human player's sentiment guess** via an intuitive UI (e.g., radio buttons).
5. **Generate an AI sentiment prediction** for the same review using a sentiment model (e.g., TextBlob).
6. **Reveal the true sentiment from the dataset** and compare all three:
    - Human guess
    - AI guess
    - True label
7. **Maintain and display scores** for human vs AI across multiple rounds.
8. **Show a final scoreboard and winner**, with an option to restart the game.

5. Technology Stack

Based on your `requirements.txt`:

- **Python**
- **Streamlit** – Web app framework for building the interactive UI.
- **pandas** – For reading and managing the dataset (CSV files).
- **TextBlob** – For AI sentiment prediction.

These are lightweight but powerful tools that are well-suited for a classroom or demo environment.

6. Dataset Description

The application expects a **CSV file** containing at least:

- A **text column**: e.g., `review`, `text`, or similar.
- A **sentiment label column**: e.g., `sentiment`, `label`.

Typical sentiment labels:

- `positive`
- `negative`
- (optional) `neutral`

The user uploads this dataset through the Streamlit UI. Once loaded, the app:

- Reads the CSV using `pandas.read_csv`.
- Stores it in memory for use throughout the gameplay.
- Randomly samples one review per round.

7. System Design & Game Flow

Your `Readme.txt` describes the game logic in flowchart form:

```
flowchart TD
    A[Start Game] --> B[Load dataset & initialize scores]
    B --> C[Randomly select a review]
    C --> D[Display review to player]
    D --> E[Player selects sentiment via radio button]
    E --> F[AI model predicts sentiment]
    F --> G[Reveal true sentiment from dataset]
    G --> H[Compare guesses with true sentiment]
    H --> I[Update scores & agreement counter]
    I --> J[Increase round number]
    J --> K{More rounds left?}
    K -->|Yes| C
    K -->|No| L[Show final scoreboard & winner]
    L --> M[Offer restart option]
    M -->|Restart| B
    M -->|Exit| N[End]
```

Step-by-Step Explanation

1. **Start Game (A)**
    - The user opens the Streamlit app in the browser.
    - A title and brief description explain the game rules.
2. **Load Dataset & Initialize Scores (B)**
    - The user uploads a CSV file using Streamlit's `file_uploader` widget.
    - Once uploaded, the app:
        - Loads the dataset into a pandas DataFrame.
        - Initializes score variables:
            - `human_score = 0`
            - `ai_score = 0`
            - `agreement_count = 0` (how often human and AI agree)
            - `round_number = 1`
        - Optionally initializes `total_rounds` (user input, e.g., 10 rounds).
3. **Randomly Select a Review (C)**
    - For each round, the app randomly selects a row from the dataset.
    - It extracts the review text and the true sentiment label but only displays the text.
4. **Display Review to Player (D)**
    - The selected review text is displayed clearly, often in a styled container or card in Streamlit.
    - Example: *"The product arrived late and was damaged."*
5. **Player Selects Sentiment via Radio Button (E)**
    - A Streamlit `radio` widget allows the player to choose one sentiment:
        - Positive
        - Neutral (if used)
        - Negative
    - The user submits their choice (often via a button).

6. **AI Model Predicts Sentiment (F)**
    - Once the player submits their guess:
        - The same review text is passed to the **TextBlob** sentiment model (or similar).
        - TextBlob returns a **polarity score** (e.g., -1 to +1).
        - The polarity is converted to a discrete label:
            - polarity > threshold → **positive**
            - polarity < -threshold → **negative**
            - else → **neutral** (if implemented).
    - This becomes the AI's guess.
7. **Reveal True Sentiment from Dataset (G)**
    - Now, the app reveals:
        - Human guess
        - AI prediction
        - True sentiment label from the original dataset
    - This is shown on screen, usually with some styling (e.g., colored text or emojis).
8. **Compare Guesses with True Sentiment (H)**
    - The game checks:
        - If human guess == true label → human gets a point.
        - If AI guess == true label → AI gets a point.
        - If human guess == AI guess → `agreement_count += 1`.
9. **Update Scores & Agreement Counter (I)**
    - Scores are updated and stored in session state variables so they persist across rounds.
    - The current scoreboard is displayed in the UI:
        - Current round number
        - Human score
        - AI score
        - Number of agreements
10. **Increase Round Number (J)**
    - `round_number += 1`
    - The game checks whether the maximum number of rounds has been reached.
11. **More Rounds Left? (K)**
    - If `round_number <= total_rounds`:
        - The flow returns to **C** to select the next random review.
    - If not:
        - The game moves to showing the final results.
12. **Show Final Scoreboard & Winner (L)**
    - At the end of all rounds:
        - Final human score
        - Final AI score
        - Total agreements
        - Win/lose/draw message
        - Optional fun message or animation (e.g., GIFs).
13. **Offer Restart Option (M) → Restart/Exit (N)**
    - A button allows the user to restart:
        - Resets scores and round number.

        - Optionally reshuffles or reloads dataset.
    - If the user chooses not to restart, the game ends.

8. Implementation Details

8.1. Core Libraries

- `streamlit` – For UI, widgets, state management.
- `pandas` – For reading CSV, selecting reviews, handling labels.
- `textblob` – For sentiment polarity and classification.

8.2. App Structure (Typical)

Even if your file is named `sentiment_game_app.py` or `streamlit_app.py`, the structure is usually:

1. **Imports**

```
import streamlit as st
import pandas as pd
from textblob import TextBlob
import random
```

6. **Session State Initialization**
    - Initialize `human_score`, `ai_score`, `agreement_count`, `round_number`, and `total_rounds` using `st.session_state`.
7. **Dataset Upload Section**
    - `st.file_uploader("Upload your reviews CSV", type=["csv"])`
    - Once uploaded, store `df` in session state.
8. **Game Controls & Layout**
    - Sidebar for settings (e.g., number of rounds).
    - Main area for displaying:
        - Current review text
        - Radio button for sentiment
        - Submit button
9. **Prediction Logic**
    - Function `get_ai_sentiment(review_text)` using TextBlob.
    - Mapping polarity to labels.
10. **Scoring & Display**
    - Logic to update scores on each round.
    - Display current and final scoreboards with Streamlit components (e.g., `st.metric`, tables, text).

9. User Interface & Experience

The UI is designed to be simple and intuitive so anyone can play:

- **Top Section:**
  - Title: *"AI vs Human: Sentiment Guessing Game"*
  - Short description: rules, objective, and how scoring works.
- **Upload Section:**
  - A file uploader widget for CSV.
  - Once dataset is loaded, the game starts.
- **Game Area (per round):**
  - Review text displayed clearly.
  - Radio buttons to select sentiment.
  - Submit button.
  - After submission:
    - Feedback text with human guess, AI guess, and true label.
    - Round result (e.g., "You scored!" or "AI scored!").
    - Animated or fun elements if you added GIFs.
- **Scoreboard Section:**
  - Real-time display of:
    - Current round / Total rounds.
    - Human score vs AI score.
    - Number of agreements / total rounds.
- **End Screen:**
  - Final scores.
  - Declaration of winner (Human / AI / Tie).
  - Button to restart the game.

10. Results & Observations

Although this is a game rather than a traditional experiment, some typical observations can be:

- **Agreement Rate:**
  - In many cases, human sentiment and AI sentiment agree, especially on clearly positive/negative reviews.
- **Edge Cases:**
  - Mixed or sarcastic comments often cause disagreements between human and AI.
- **User Engagement:**
  - Turning sentiment analysis into a game makes the concept more understandable and engaging for classmates or non-technical users.

You can optionally add a short summary of any informal "playtests" (e.g., how many classmates played, who won more often, etc.).

11. Limitations

- **Model Simplicity:**
  - TextBlob is rule-based and not as powerful as transformer models (e.g., BERT, RoBERTa).
- **Dataset Dependence:**
  - The quality of the game depends on the dataset's label accuracy.
- **Subjectivity:**
  - Sentiment can be subjective; disagreements are not always model "errors."

12. Future Enhancements

You can mention planned improvements such as:

1. **Use advanced models** such as fine-tuned BERT/RoBERTa for more accurate sentiment predictions.
2. **Multi-class sentiment** (e.g., 1–5 star ratings instead of just positive/negative).
3. **Difficulty levels** based on text complexity or length.
4. **Leaderboard for multiple players** (e.g., save scores to a database).
5. **Explainability** – show why the AI predicted a particular sentiment (keywords, polarity contributions).
6. **Gamification features** like badges, streaks, or time-based scoring.

13. Conclusion

This project successfully transforms a standard NLP task—sentiment analysis—into an interactive game using Streamlit. By allowing human players to compete against an AI model on real review data, the app makes machine learning more accessible and engaging.

The system demonstrates:

- Reading and processing user-provided datasets.
- Applying sentiment analysis with TextBlob.
- Managing game state, scores, and rounds using Streamlit.
- Presenting clear, interactive visuals that reinforce machine learning concepts.

This final project not only meets the technical requirements of building a functional ML-powered web app but also emphasizes user experience, education, and engagement.