Project Report

Simple Sentiment Analyzer

Name: Harshvardhan Singh Rathore

Course: Fundamentals of AI and Ml

Registration No.: 25BCY10085

Introduction

Sentiment Analysis is a Natural Language Processing (NLP) technique used to identify whether a given text expresses a positive or negative emotion. In this mini-project, a simple sentiment classifier is built using Python and scikit-learn. It uses the Bag-of-Words model to process text and a Naive Bayes classifier to categorize sentences. This project demonstrates the core concepts of machine learning in a simple and beginner-friendly way.

Problem Statement

Manually identifying the sentiment behind user feedback or text is time-consuming and subjective. A lightweight automated system is required to classify text as **positive** or **negative** instantly. This project aims to implement such a simple sentiment classifier using machine learning.

Functional Requirements

- The system must accept a text input from the user.
- The system must classify the text as **Positive** or **Negative**.
- The system must display prediction confidence.
- The system should train on a predefined mini dataset.

Non-Functional Requirements

- **Usability:** Should be easy to run via Python script.
- **Performance:** Real-time sentiment prediction.
- **Maintainability:** Training data and rules should be easily modifiable.
- **Portability:** Should run on any system with Python installed.

System Architecture

The architecture contains the following stages:

1. Input sentence
2. Text preprocessing (Bag-of-Words)
3. Model training (Naive Bayes)
4. Sentiment prediction
5. Output label + confidence

Design Diagrams

Use Case Diagram

Users: Students, beginners, teachers
System: Sentiment Analyzer

- User → Enter text
- System → Predict sentiment
- System → Display result

Workflow Diagram

1. Start
2. Load training dataset
3. Convert text to numeric vector
4. Train Naive Bayes classifier
5. Accept user sentence
6. Convert input to vector
7. Predict sentiment
8. Display result
9. End

Sequence Diagram

User → System

- Input sentence
- Vectorization
- Model prediction
- Return sentiment

Class / Component Diagram

- **SentimentModel**
    - vectorizer (CountVectorizer)
    - model (Naive Bayes)
    - train()
    - predict()

Design Decisions & Rationale

- **Naive Bayes** was chosen because it is simple, fast, and works well for text classification.
- **CountVectorizer (Bag-of-Words)** was used because it is easy to implement for beginners.
- A **small custom dataset** was used to keep the system lightweight and simple.

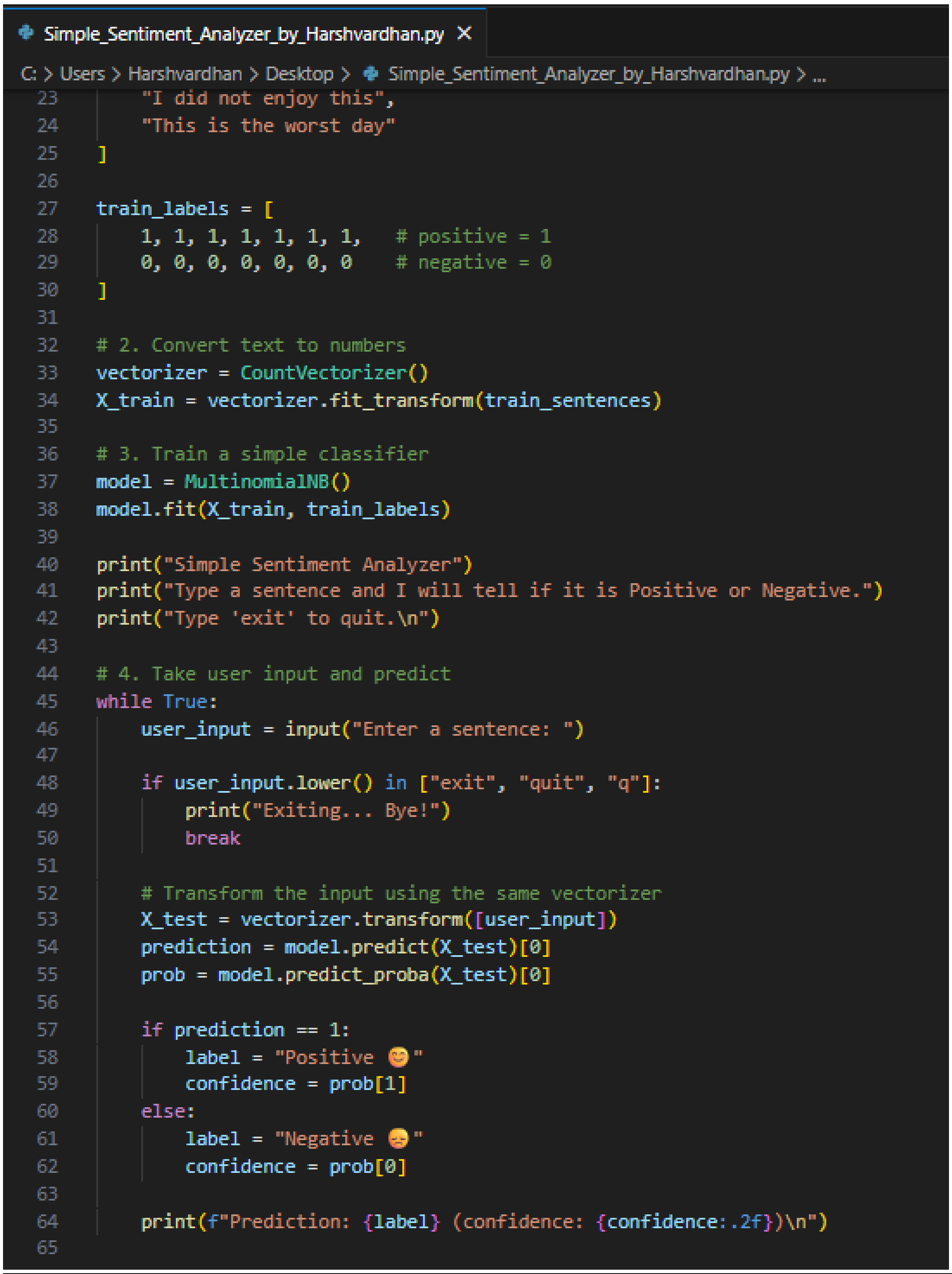
Implementation Details

- Language: Python
- Libraries: scikit-learn
- Dataset: 14 manually written sentences (positive/negative)
- Model: Multinomial Naive Bayes

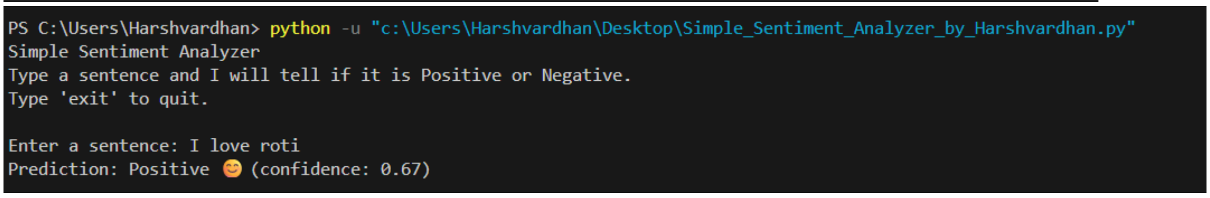
Main Steps:

1. Define training data
2. Convert text to vectors
3. Train Naive Bayes classifier
4. Take runtime input
5. Predict sentiment
6. Display output

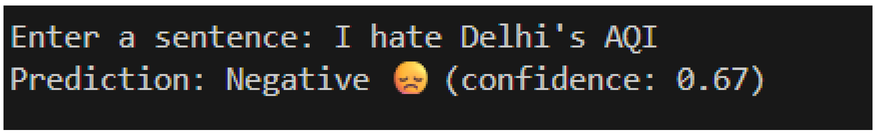
Screenshots / Results:


Simple_Sentiment_Analyzer_by_Harshvardhan.py
C: > Users > Harshvardhan > Desktop > Simple_Sentiment_Analyzer_by_Harshvardhan.py > ...

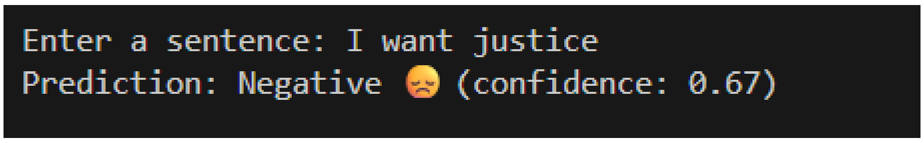
```python
    "I did not enjoy this",
    "This is the worst day"
]

train_labels = [
    1, 1, 1, 1, 1, 1, 1,   # positive = 1
    0, 0, 0, 0, 0, 0, 0    # negative = 0
]

# 2. Convert text to numbers
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(train_sentences)

# 3. Train a simple classifier
model = MultinomialNB()
model.fit(X_train, train_labels)

print("Simple Sentiment Analyzer")
print("Type a sentence and I will tell if it is Positive or Negative.")
print("Type 'exit' to quit.\n")

# 4. Take user input and predict
while True:
    user_input = input("Enter a sentence: ")

    if user_input.lower() in ["exit", "quit", "q"]:
        print("Exiting... Bye!")
        break

    # Transform the input using the same vectorizer
    X_test = vectorizer.transform([user_input])
    prediction = model.predict(X_test)[0]
    prob = model.predict_proba(X_test)[0]

    if prediction == 1:
        label = "Positive 😊"
        confidence = prob[1]
    else:
        label = "Negative 😞"
        confidence = prob[0]

    print(f"Prediction: {label} (confidence: {confidence:.2f})\n")

```



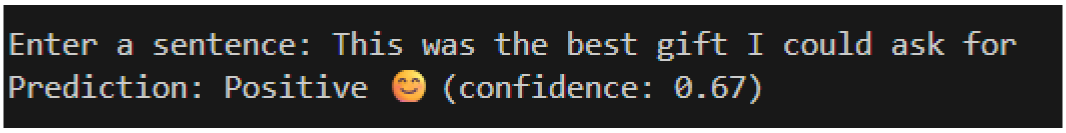
```
PS C:\Users\Harshvardhan> python -u "c:\Users\Harshvardhan\Desktop\Simple_Sentiment_Analyzer_by_Harshvardhan.py"
Simple Sentiment Analyzer
Type a sentence and I will tell if it is Positive or Negative.
Type 'exit' to quit.

Enter a sentence: I love roti
Prediction: Positive 😊 (confidence: 0.67)
```



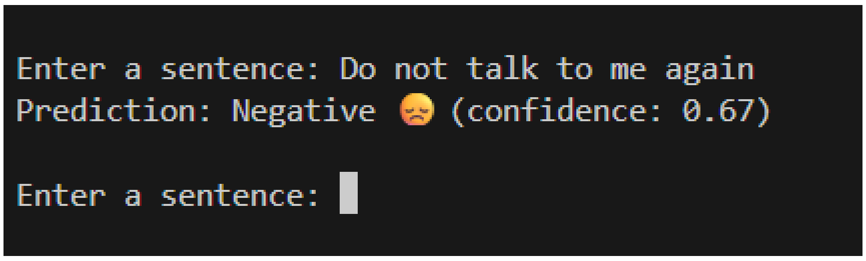
```
Enter a sentence: I hate Delhi's AQI
Prediction: Negative 😞 (confidence: 0.67)
```



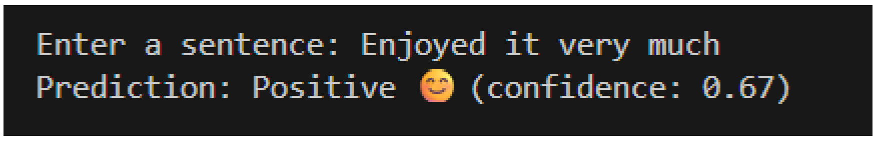
```
Enter a sentence: I want justice
Prediction: Negative 😞 (confidence: 0.67)
```



```
Enter a sentence: This was the best gift I could ask for
Prediction: Positive 😊 (confidence: 0.67)
```



```
Enter a sentence: Do not talk to me again
Prediction: Negative 😞 (confidence: 0.67)

Enter a sentence:
```



```
Enter a sentence: Enjoyed it very much
Prediction: Positive 😊 (confidence: 0.67)
```

Testing Approach

- **Unit testing:** Test with sentences of known sentiment.
- **Edge cases:**
    - Very short text (“bad”, “good”)
    - Neutral text (“okay”, “fine”)
    - Empty string
- **User testing:** Try multiple custom sentences.

Challenges Faced

- Small dataset may reduce accuracy.
- Mixed/neutral sentences are sometimes misclassified.
- Words not present in the training dataset may confuse the model.
- Understanding vectorization and model training for the first time.

Learnings & Key Takeaways

- Basics of NLP and Bag-of-Words representation
- How a classifier is trained using scikit-learn
- How to preprocess and convert text into numerical form
- How ML models make predictions based on patterns

Future Enhancements

- Add **neutral** sentiment classification
- Use a larger dataset to improve accuracy
- Add a **Tkinter GUI**
- Use advanced models like Logistic Regression or SVM
- Deploy the project as a **web app**

References

- scikit-learn documentation
- Python official documentation
- Basic ML and NLP tutorials
- Course materials