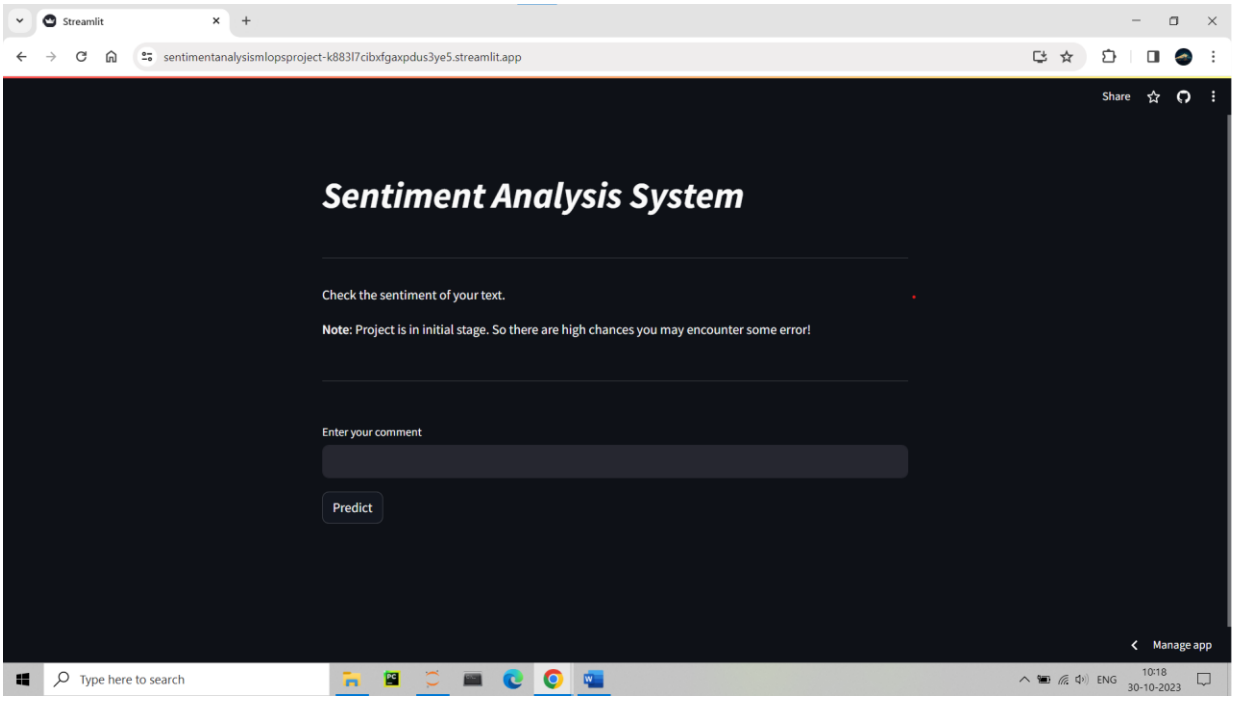
Sentiment Analysis System



Date: 29/10/2023

By: Yash Keshari

Document Version Control

Date Issued	Version	Description	Author
29/10/2023	1	Documentation- V 0.1	Yash Keshari

Content :

1 Introduction

Sentiment analysis is often referred as opinion mining. It is a natural language processing (NLP) technique that aims to determine the emotional tone or sentiment expressed in a piece of text or comment, such as a review, social media post, or comment. It involves the use of machine learning and NLP algorithms to classify the sentiment of text as positive, negative, or neutral.

Sentiment analysis has many applications, including customer feedback analysis, brand monitoring, market research, and social media sentiment tracking. It provides valuable insights into public opinion and can help businesses and organizations to make data-driven decisions based on the sentiment of their customers or the general public.

2 General Description

2.1 Product Perspective

This Sentiment Analysis model operates within a machine learning-based prediction model. It serves as a tool for sentiment classification and plays a crucial role in understanding user sentiments from comments, reviews, or any textual data. The model's insights can be instrumental in making informed decisions related to customer feedback analysis, brand monitoring, and market research.

2.2 Problem statement

The primary objective of this AI solution is to perform sentiment analysis effectively on text data. The specific use cases include:

- Accurately classify sentiments within textual data as Negative or Positive class
- Provide businesses with insights into public opinion and feedback.
- Enhancing customer experience by understanding and addressing sentiments within user comments.

2.3 Proposed solution

Our Sentiment Analysis model uses Multinomial Naive Bayes algorithm to classify comments into Negative and Positive sentiments. The model processes textual data through various NLP techniques, making it possible to understand sentiment accurately.

By employing this model, businesses and organizations can gain a deeper understanding of their customers sentiments. This information is invaluable for improving customer satisfaction, making data-driven decisions, and enhancing overall product and service quality.

3. Tools used

Tools and Technologies Used -

- **Programming Language**: Python
- **Python Libraries and Frameworks**:
  - NumPy
  - Pandas
  - Scikit-learn
  - Natural language processing library - NLTK
- **Integrated Development Environment (IDE)**: Vs-code
- **Data Visualization**:
  - Seaborn
- **Cloud Computing and Deployment**:
  - Streamlit cloud
- **Front-end Development**:
  - Streamlit.
- **GitHub is used as version control**

4. Data Preprocessing:

- **Data Loading:**

The data was loaded from a CSV file from the notebook/data folder.

- **Removing Duplicate Entries:**

Duplicate rows were removed from the dataset. This step ensures that the dataset contains unique comments.

- **Filtering Out Irrelevant Labels:**

Rows with label values tagged as 'O' were removed from the dataset. The condition df['label'] == 'O' was applied, and rows where the condition is True were dropped. This step is useful for filtering out irrelevant data. In this case 'O' was very low in number, just 36 rows compared to our dataset size of 41k rows. Which is very small, hence it was removed.

- **Label Encoding:**

The label values 'N' and 'P' were encoded into numeric values '0' and '1', respectively. This encoding is essential for building machine learning models, which requires numeric labels.

- **NLP Text Conversion:**

The 'comment' column was transformed using an NLP (Natural Language Processing) function called "nlp_transform". The transformed text was stored in a new column "lemma_transform" in the DataFrame. This step involved task such as tokenization, lemmatization, and other text preprocessing techniques to prepare the text data for analysis.

- **Final Columns Drop:**

Several columns were dropped from the DataFrame to retain only the relevant columns for the analysis. Columns 'Unnamed: 0,' 'comment,' and 'label' were removed.

- **Logging and Data Storage:**

Logging statements were used to record the progress of the data transformation process. The transformed data was stored in a CSV file. This step ensures that the transformed data is saved for future use.

- **Subsampling for Model Training:**

A subset of the data was selected for model training. Positive ('P') and negative ('N') samples were separately subsampled to balance the class distribution. This step involved random subsampling and resulted in a DataFrame called 'subsampled_df.'

- **Shuffling Data:**

The entire DataFrame was shuffled to randomize the order of samples, reducing any potential bias in the data.

- **Splitting Data:**

The final data was split into feature data (X) and target data (y) for model training. 'lemma_transform' was used as the feature (X), and 'encoded_label' was used as the target label (y). This step prepared the data for training machine learning models.

5. Process Flow/Model Architecture.

Sentiment analysis Process Flow:

- **Data Collection**:
  Gather data from sources. (hate.csv)

- **Data Preprocessing**:
  Clean and prepare the data, handle missing values, and transform features for model training, transforming Text data into numbers via NLTK, Vectorization.

- **Machine Learning Model Development:**
  Select and implement machine learning algorithms to build a classification model. Train the model on the preprocessed data to establish the relationship between comments and labels. For this purpose we used Mutlinominal naïve bayes algorithm.

- **Model Evaluation:**
  Assess the model performance using metrics like Precision, recall, f1-score.

- **User Interface:**
  Design and develop a user-friendly interface, such as a web or mobile app, for users to input their comments, This is done using streamlit library.

- **Model Deployment:**
  Deploy the trained model on a cloud platform (e.g., streamlit cloud or AWS) to make it accessible to users.

- **User Interaction:**
  Users input their comments via the user interface.

- **Prediction Generation:**
  The system processes the user input through the trained model to generate sentiment prediction.

- **Output Presentation:**

  Display the fare estimate to the user, allowing them to make informed booking decisions.

6. Model Training.

- **Model Selection:**

  In the training phase, multiple machine learning models were considered for sentiment analysis. These models include Logistic Regression and Naive Bayes models such as Multinomial Naive Bayes, Bernoulli Naive Bayes, Gaussian Naïve bayes.

- **Training and Evaluation:**

  Each model is trained on the preprocessed data, which includes the text features transformed using NLP techniques.
  After training, the models are evaluated on a validation dataset, and their performance is assessed using relevant metrics such as Precision, recall and f1-score.

- **F1 Score Consideration:**

  The primary focus during model selection was achieving a balance between precision and recall. The F1 score, which combines both precision and recall, was used as the evaluation metric.
  A model with a moderate F1 score is chosen to ensure a trade-off between minimizing false positives and false negatives.

- **Model Selection Outcome:**

  After evaluation, the Multinomial Naive Bayes model is selected based on its performance achieve the balance between precision and recall.

7. Evaluation results and analysis

Model: Multinomial Naive Bayes (Alpha_Value = 20)

Performance Metrics:

- Accuracy: 65.03%
- Precision: 67.15%
- F1-Score: 62.75%
- Confusion Matrix:

- True Negatives (TN): 3735
- False Positives (FP): 1513
- False Negatives (FN): 2159
- True Positives (TP): 3093

Analysis:

The sentiment analysis model gives a reasonable level of accuracy at approximately 65.03%. It correctly classifies the majority of instances, making it suitable for general sentiment analysis tasks.

The F1-Score of 62.75% suggests that the model maintains a good balance between precision and recall, which is important for minimizing false positives and false negatives.

The model can be improved further with proper time and analysis.

8. Project Folder Architecture

```
|- artifacts/ (Include trained models or other important artifacts)
|
|- notebooks/
| |- notebook.ipynb (Jupyter Notebook for the project)
| |- data/ (Data files or datasets)
|
|- logs/ (Log files or logs directory)
|
|- src/
| |- logger.py (Logging utilities)
| |- exception.py (Custom exception handling)
| |- utils.py (Utility functions)
| |- __init__.py
|
| |- components/
|       -__init__.py
|       - data_ingestion.py (Data loading functions)
|       - data_transformation.py (Data preprocessing, feature engineering)
|       - model_training.py (Machine learning model training)
|
| |- pipelines/
|       - __init__.py
|       - prediction_pipeline.py (prediction pipeline)
|       - training_pipeline.py (Model training pipeline)
|
|- requirements.txt (List of project dependencies)
|
|- setup.py (Setup script for the project)
|
|- app.py (Main application script)
|
|- .gitignore (Specify files or directories to ignore in version control)
```

9. References

- Google
- Documentations – Scikit learn, pandas, streamlit, ect.