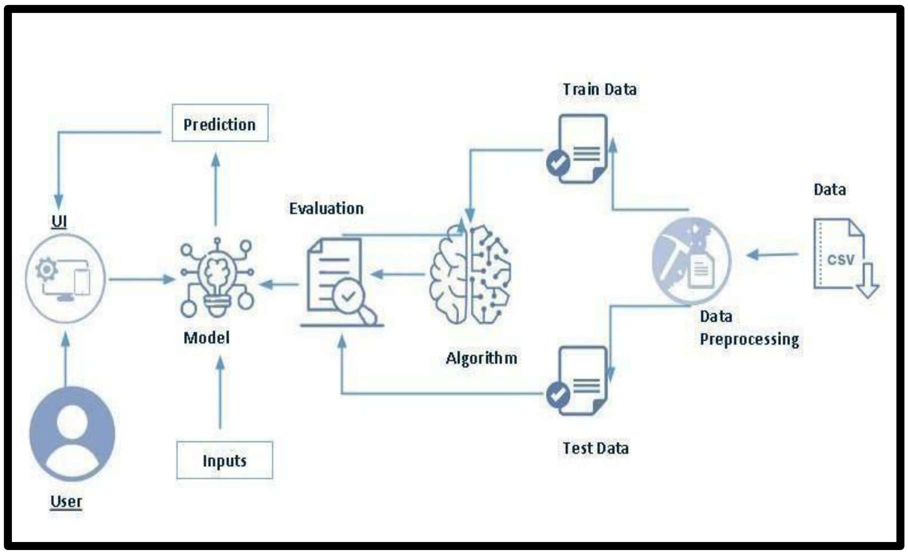
Analysis of Medium App Reviews from Google Play Store

SmartInternz
www.smartinternz.com

Analysis of Medium App Reviews from Google Play Store

Sentiment Insights: Analyzing Medium App Reviews from Google Play Store" is a sophisticated analytical tool designed to evaluate user feedback on Medium apps available on the Google Play Store. This project focuses on leveraging natural language processing and sentiment analysis techniques to classify reviews as positive, negative, or neutral. By analyzing user sentiments, the tool provides valuable insights into user experiences, identifies common issues or praises, and helps developers and stakeholders understand user perceptions and areas for improvement. The ultimate goal is to enhance app development and user satisfaction by providing actionable insights based on comprehensive review analysis.

Technical Architecture:



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements ○ Literature Survey
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation
- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis
- Model Building
  - Training the model in multiple algorithms
  - Testing the model
- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning

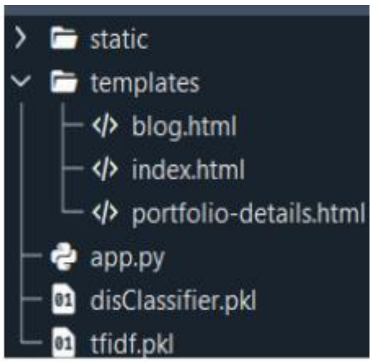
- Model Deployment
  - Save the best model
  - Integrate with Web Framework
- Project Demonstration & Documentation
  - Record explanation Video for project end to end solution
  - Project Documentation-Step by step project development procedure

Prior Knowledge:

- ML Concepts
- Supervised learning: https://www.javatpoint.com/supervised-machine-learning
- Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
- Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification- algorithm
- Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
- KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
- Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understands-the-math-behind-xgboost/
- Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model- evaluation-error-metrics/
- NLP:-https://www.javatpoint.com/nlp
- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



```
> static
v templates
    blog.html
    index.html
    portfolio-details.html
  app.py
  disClassifier.pkl
  tfidf.pkl
```

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- Dtc_model.pkl is our saved model. Further we will use this model for flask integration.
- Data Folder contains the Dataset used
- The Notebook file contains procedure for building th model.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Refer Project Description

Activity 2: Business requirements

A sentiment analysis system for app reviews should fulfill the following business needs:

- **Accurate Classification**: The system must reliably identify positive, negative, and neutral sentiments from textual reviews to support data-driven decisions.
- **Scalability**: Capable of processing thousands of reviews regularly without compromising speed or performance.
- **Adaptability**: Able to accommodate new reviews, app versions, and language styles without extensive retraining.
- **Compliance**: Must align with privacy and ethical standards for processing user-generated content.
- **User Interface**: Provide an intuitive dashboard or web interface for non-technical stakeholders to access sentiment insights and visual reports.

Activity 3: Literature Survey (Student Will Write)

The literature survey for this project will review NLP-based sentiment classification techniques with a focus on app reviews. It will include works on TF-IDF and word embeddings for feature representation, as well as deep learning models like LSTM and BiLSTM for capturing contextual meaning. Previous studies will be analyzed to understand the benefits and limitations of these approaches, and to identify opportunities for real-time visualization and deployment.**Activity**

4: Social or Business Impact.

- **Social Impact**: Automating the interpretation of user feedback can significantly improve response times for addressing bugs, frustrations, or suggestions, enhancing the user experience and community satisfaction.

- **Business Impact**: By understanding sentiment trends, product teams can prioritize features and fixes more effectively. This can improve app ratings, reduce churn, and increase overall engagement and competitive advantage.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

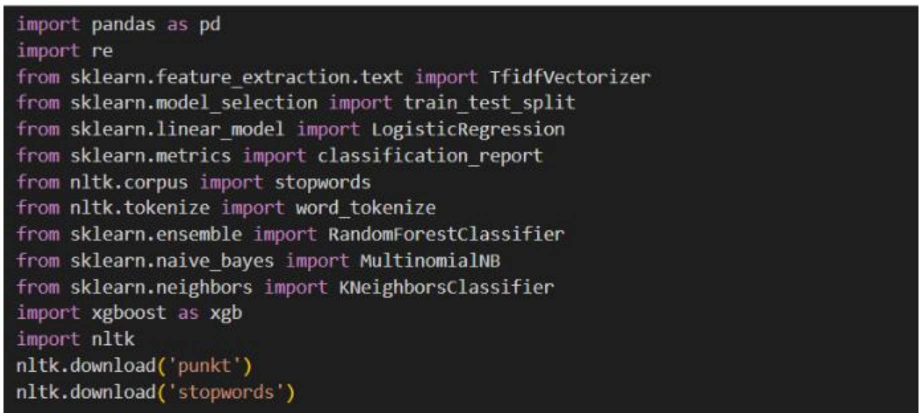
Link: https://www.kaggle.com/datasets/raqhea/medium-app-reviews-from-google-play-store

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.



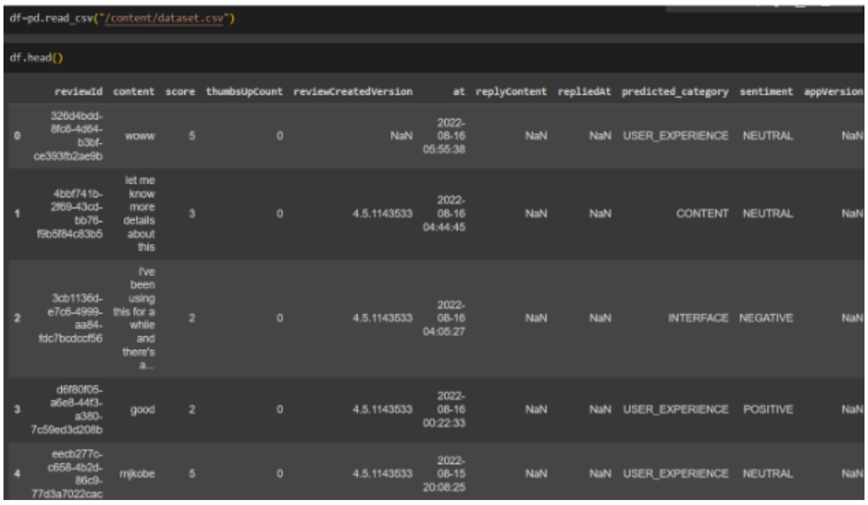
```python
import pandas as pd
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
import xgboost as xgb
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

- For checking the null values, df.isna().any( ) function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.



```python
df=pd.read_csv("/content/dataset.csv")
```

```python
df.head()
```

	reviewId	content	score	thumbsUpCount	reviewCreatedVersion	at	replyContent	repliedAt	predicted_category	sentiment	appVersion
0	326d4bdd-8fc6-4d64-b3bf-ce393fb2ae9b	woww	5	0	NaN	2022-08-16 05:55:38	NaN	NaN	USER_EXPERIENCE	NEUTRAL	NaN
1	4bbf741b-2f69-43cd-bb76-f9b5f84c83b5	let me know more details about this	3	0	4.5.1143533	2022-08-16 04:44:45	NaN	NaN	CONTENT	NEUTRAL	NaN
2	3cb1136d-e7c6-4999-aa84-fdc7bcdccf56	I've been using this for a while and there's a...	2	0	4.5.1143533	2022-08-16 04:05:27	NaN	NaN	INTERFACE	NEGATIVE	NaN
3	d6f80f05-a6e8-44f3-a380-7c59ed3d208b	good	2	0	4.5.1143533	2022-08-16 00:22:33	NaN	NaN	USER_EXPERIENCE	POSITIVE	NaN
4	eecb277c-c658-4b2d-86c9-77d3a7022cac	mjkobe	5	0	4.5.1143533	2022-08-15 20:08:25	NaN	NaN	USER_EXPERIENCE	NEUTRAL	NaN

Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data.

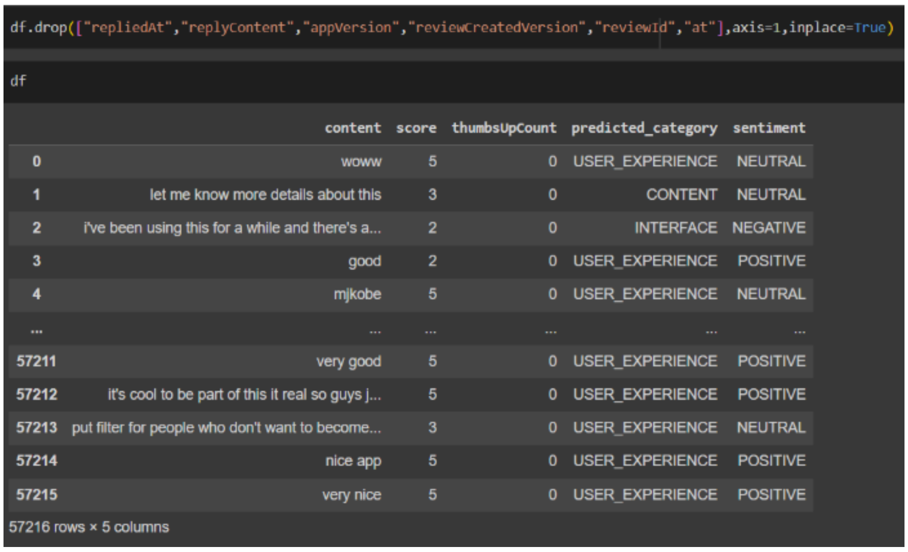
The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling Outliers

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Handling missing values

- For checking the null values, df.isna().any( ) function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.



```
df.drop(["repliedAt","replyContent","appVersion","reviewCreatedVersion","reviewId","at"],axis=1,inplace=True)
```

```
df
```

```
                                                content  score  thumbsUpCount  predicted_category  sentiment
0                                                  woww      5              0     USER_EXPERIENCE    NEUTRAL
1                       let me know more details about this      3              0             CONTENT    NEUTRAL
2             i've been using this for a while and there's a...      2              0           INTERFACE   NEGATIVE
3                                                  good      2              0     USER_EXPERIENCE   POSITIVE
4                                                mjkobe      5              0     USER_EXPERIENCE    NEUTRAL
...                                                 ...    ...            ...                 ...        ...
57211                                         very good      5              0     USER_EXPERIENCE   POSITIVE
57212       it's cool to be part of this it real so guys j...      5              0     USER_EXPERIENCE   POSITIVE
57213  put filter for people who don't want to become...      3              0     USER_EXPERIENCE    NEUTRAL
57214                                          nice app      5              0     USER_EXPERIENCE   POSITIVE
57215                                         very nice      5              0     USER_EXPERIENCE   POSITIVE
57216 rows × 5 columns
```

Activity 2.2: Text Processing

Text pre-processing (python packages)

Text preprocessing is a crucial step in Natural Language Processing (NLP) and Information Retrieval (IR) tasks. The goal is to convert raw text into a more meaningful and manageable representation for further analysis.

In Python, several packages provide support for text pre-processing operations. Some of the most common ones are:

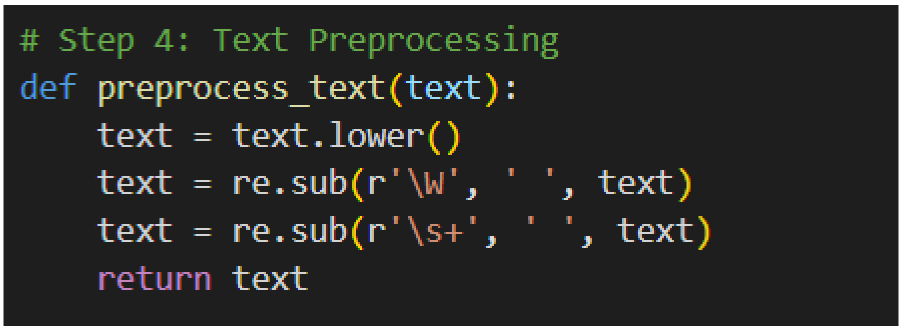
NLTK (Natural Language Toolkit)-

It is one of the most widely used NLP libraries in Python. It provides tools for tokenization, stemming, lemmatization, stop-word removal, and more.

In text preprocessing, we first convert all text to lowercase for uniformity. Then, we tokenize the

text into smaller units like words or sentences. Special characters and numbers are removed to
eliminate noise. Common words, known as stopwords, are also removed. Lemmatization further refines the text by reducing words to their base forms. Tokens are rejoined into text, and empty rows are handled. Finally, the preprocessed data is ready for analysis.

- Lowercasing: Converting all text to lowercase to ensure consistency in word representations.
- Tokenization: Breaking down the text into smaller units such as words, phrases, or sentences.
- Removing Stopwords: Removing common words like "and", "the", and "is" that occur
- frequently in the language but usually don't contribute much to the meaning of the text.
- Stemming or Lemmatization: Reducing words to their base or root form to normalize variations. Stemming chops off prefixes or suffixes, while lemmatization maps words to their dictionary form.
- Removing special characters and numbers: Dealing with numerical values by either replacing them with placeholders or converting them into text representations, Addressing special characters, emojis, or symbols that might not be relevant to the analysis.

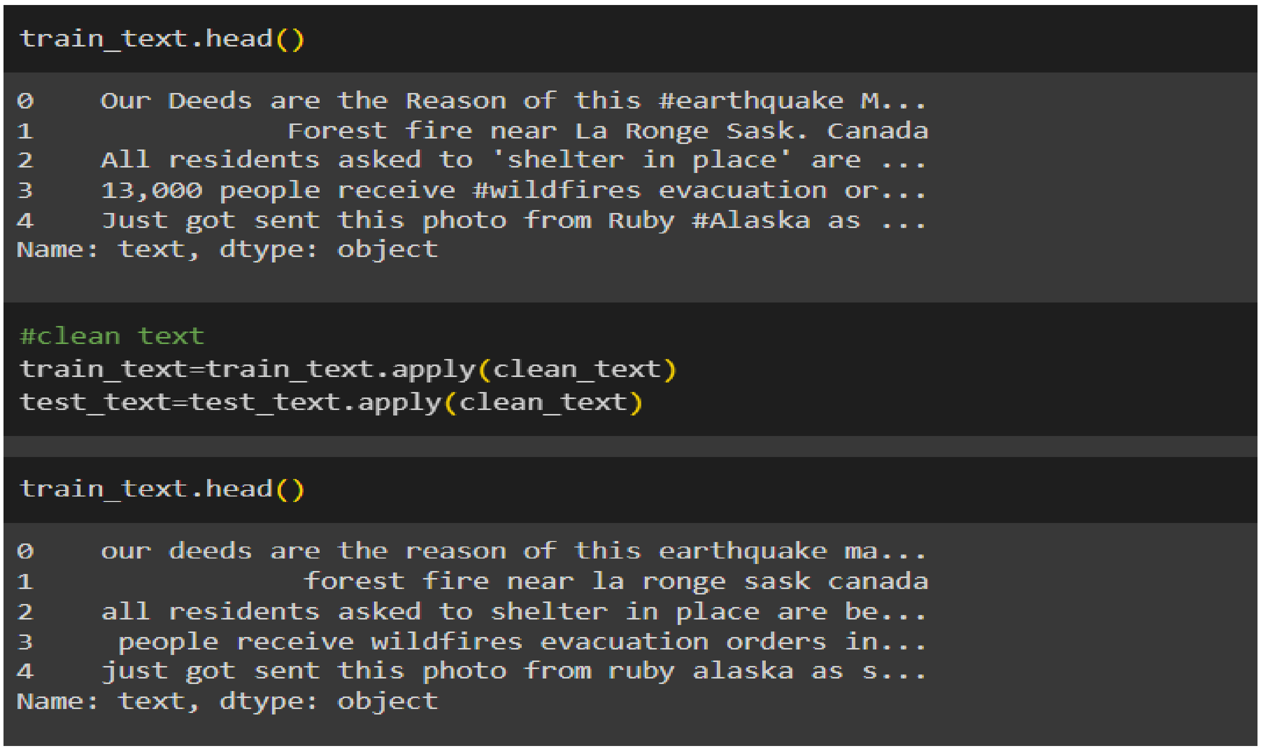


```
# Step 4: Text Preprocessing
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'\W', ' ', text)
    text = re.sub(r'\s+', ' ', text)
    return text
```

The clean_text function is designed to process and clean a given text string by removing specific unwanted elements such as hashtags, punctuation, and other non-alphabetic characters. Here's a detailed explanation of the function:

1. Lowercasing the Text: The first step in the function is to convert the entire text to lowercase using text.lower(). This ensures that the text is uniform in terms of letter casing, which can be helpful for further processing, such as comparison or analysis.
2. Removing Hashtags: The function then uses the re.sub method to remove hashtags from the text. The regular expression # is used to find any hashtag characters, and they are replaced with an empty string (''). This step specifically targets and removes the # symbol.
3. Removing Non-Alphabetic Characters: The next re.sub method call uses the regular expression [^a-zA-Z] to identify and remove any character that is not an alphabet letter (a-z or A-Z) or a space. This effectively strips the text of punctuation marks, numbers, and other special characters, leaving only letters and spaces.
4. Returning the Cleaned Text: Finally, the cleaned text, now free of hashtags, punctuation, and non-alphabetic characters, is returned.

Overall, the function provides a straightforward way to clean text data, making it more suitable for tasks that require analysis of alphabetic content, such as natural language processing or text mining.



```
train_text.head()
```

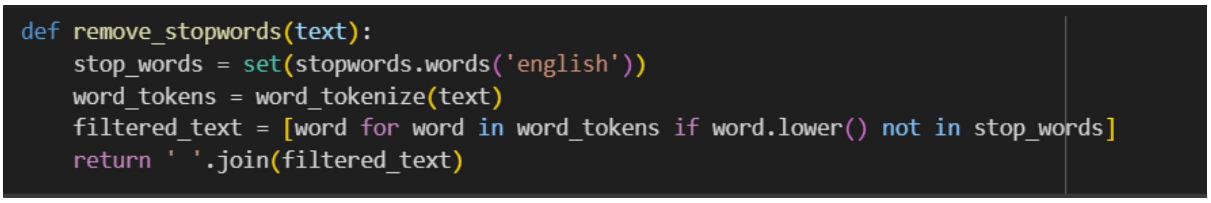
```
0    Our Deeds are the Reason of this #earthquake M...
1                 Forest fire near La Ronge Sask. Canada
2    All residents asked to 'shelter in place' are ...
3    13,000 people receive #wildfires evacuation or...
4    Just got sent this photo from Ruby #Alaska as ...
Name: text, dtype: object
```

```
#clean text
train_text=train_text.apply(clean_text)
test_text=test_text.apply(clean_text)
```

```
train_text.head()
```

```
0    our deeds are the reason of this earthquake ma...
1                  forest fire near la ronge sask canada
2    all residents asked to shelter in place are be...
3     people receive wildfires evacuation orders in...
4    just got sent this photo from ruby alaska as s...
Name: text, dtype: object
```

In this code snippet, the `train_text` data, which consists of textual entries, is processed using the previously defined `clean_text` function. Initially, the `head()` method is used to display the first few entries of `train_text`, showing text with hashtags and punctuation. The `clean_text` function is then applied to both `train_text` and `test_text` using the `apply` method, resulting in the removal of hashtags, punctuation, and other non-alphabetic characters, as well as converting all text to lowercase. The cleaned version of `train_text` is displayed again using the `head()` method, revealing the text in a more uniform and simplified format, ready for further analysis or processing.



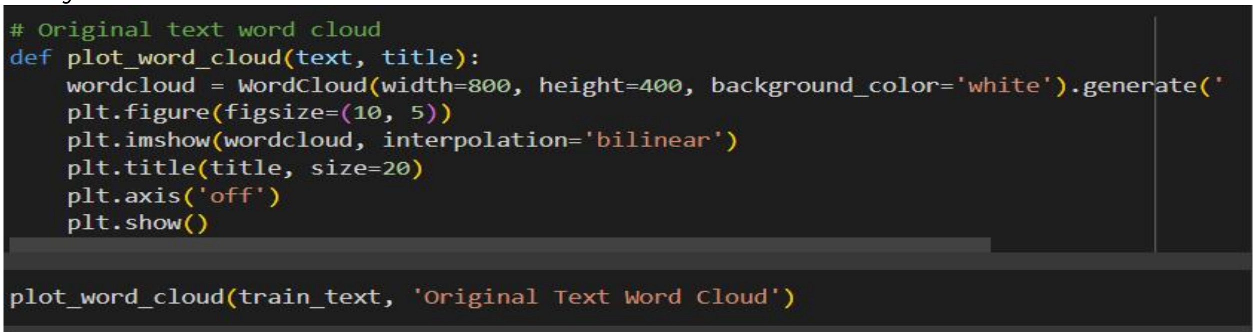
```
def remove_stopwords(text):
    stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(text)
    filtered_text = [word for word in word_tokens if word.lower() not in stop_words]
    return ' '.join(filtered_text)
```

In this code snippet, text preprocessing is extended by tokenizing and lemmatizing the text data from both `train_text` and `test_text`. The `WordNetLemmatizer` is initialized to perform lemmatization, which converts words to their base or root form. For each entry in `train_text`, the text is tokenized into words using `nltk.word_tokenize`, and stopwords (common words like 'the', 'is', 'in') are removed using a predefined list from `nltk.corpus.stopwords`. The remaining words are lemmatized and then rejoined into a cleaned sentence, which is appended to the `train_sequence` list. This process is repeated for `test_text` to create `test_sequence`. The lengths of `train_sequence` and `test_sequence` are confirmed to be 7613 and 3263, respectively, ensuring that all entries have been processed. The first processed entry in `train_sequence` is displayed as "deed reason earthquake may allah forgive u", demonstrating the effectiveness of the preprocessing steps.

Milestone 3: Data Analysis

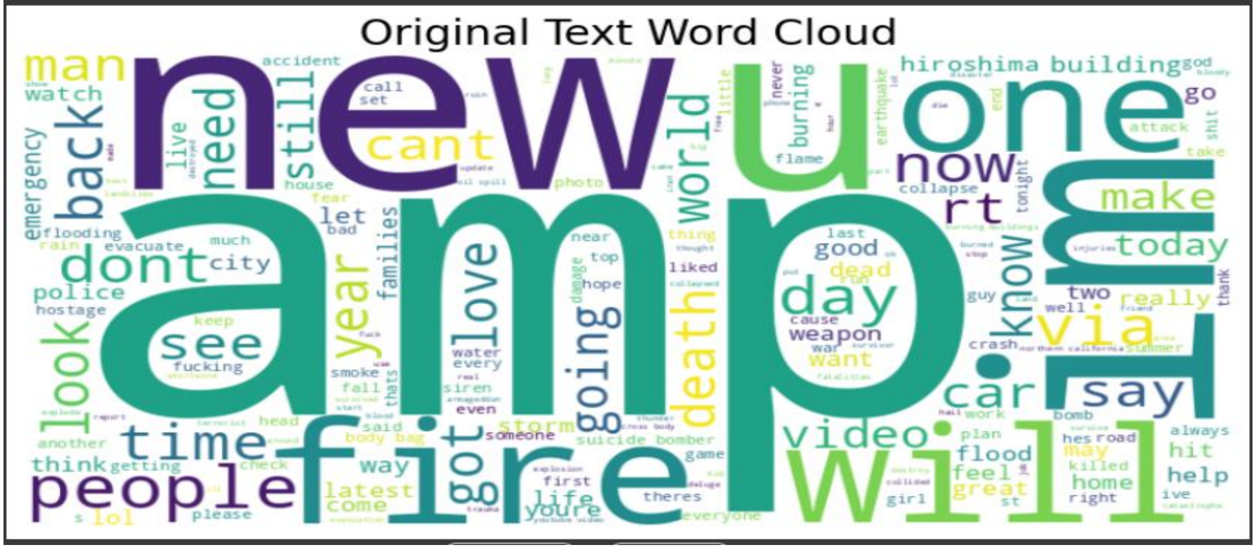
Activity 1: Word Cloud

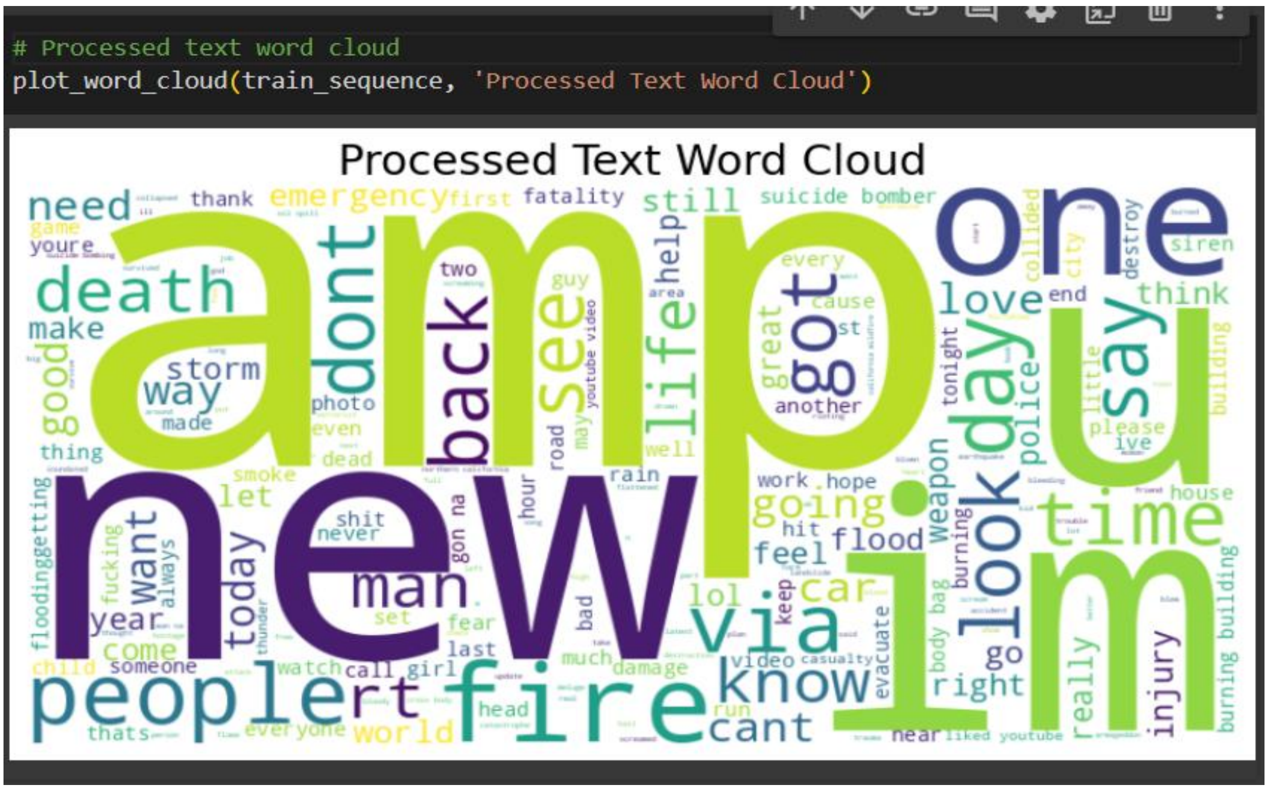
To create a word cloud for the "generated" column across two categories, we segment the data,
tokenize the text, remove noise like special characters and numbers, eliminate stopwords, and apply lemmatization. Then, we calculate word frequencies for each category and generate separate word clouds. This visualizes prevalent themes within each category, analysis



```
# Original text word cloud
def plot_word_cloud(text, title):
    wordcloud = WordCloud(width=800, height=400, background_color='white').generate('
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(title, size=20)
    plt.axis('off')
    plt.show()
```

```
plot_word_cloud(train_text, 'Original Text Word Cloud')
```



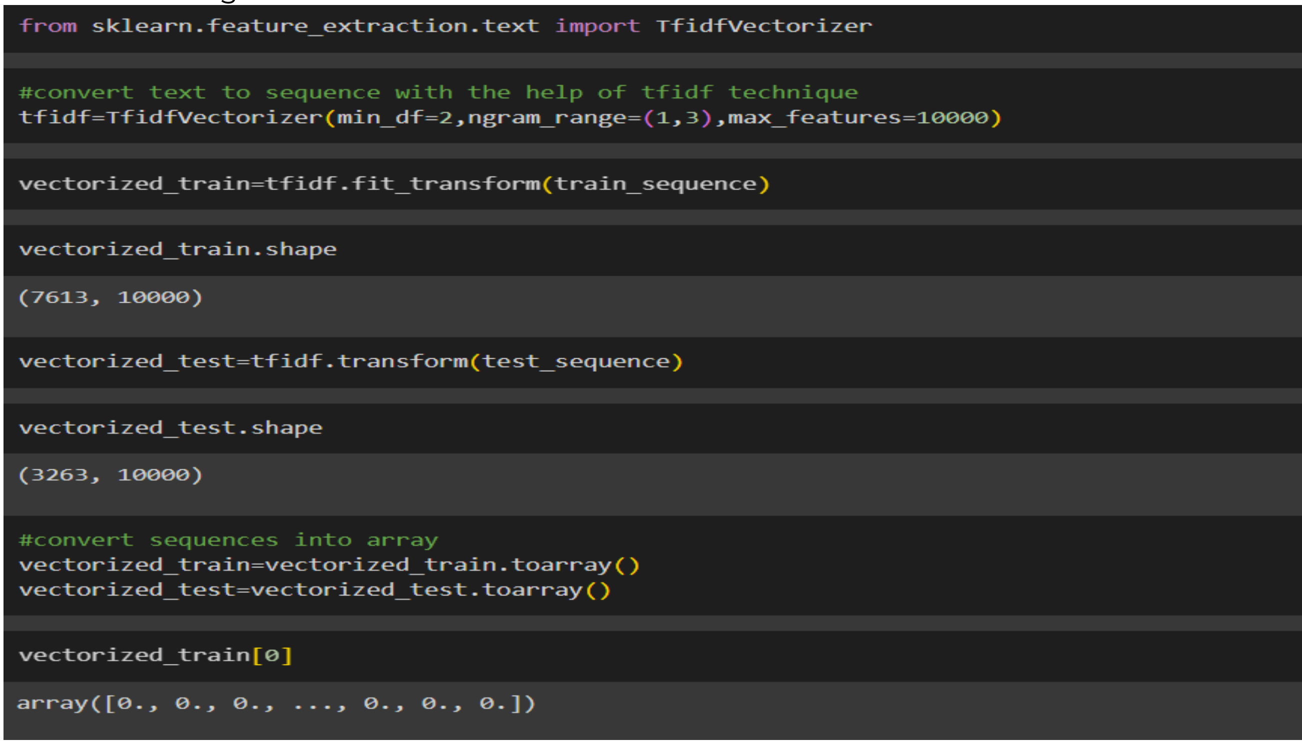


```
# Processed text word cloud
plot_word_cloud(train_sequence, 'Processed Text Word Cloud')
```

Milestone 4: Model Building

Activity 1: Vectorize Text Data

TF-IDF vectorization converts text data into numerical format. It assigns weights to words based on their frequency in a document and across all documents. This process captures the significance of words within individual documents while considering their prevalence in the entire dataset. The resulting numerical representation facilitates the application of machine learning algorithms to analyze and extract insights from textual data



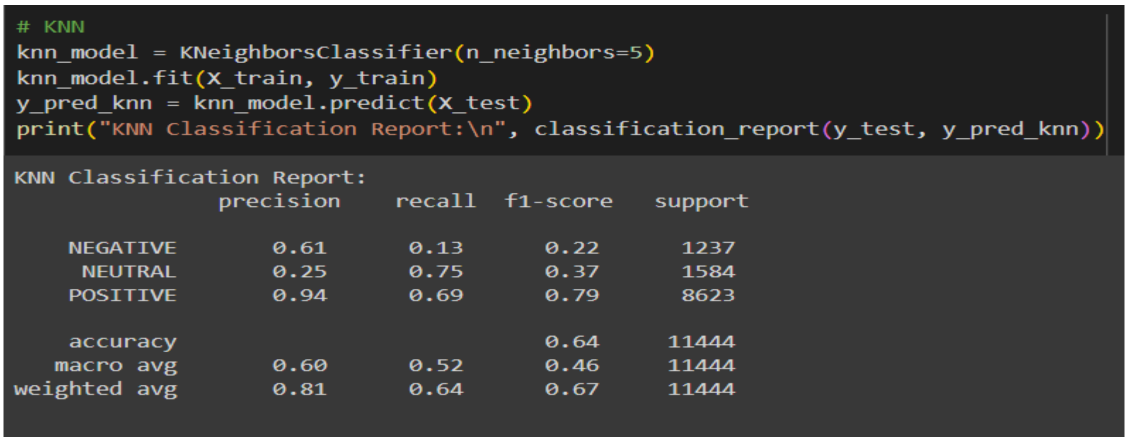
```
from sklearn.feature_extraction.text import TfidfVectorizer

#convert text to sequence with the help of tfidf technique
tfidf=TfidfVectorizer(min_df=2,ngram_range=(1,3),max_features=10000)

vectorized_train=tfidf.fit_transform(train_sequence)

vectorized_train.shape

(7613, 10000)

vectorized_test=tfidf.transform(test_sequence)

vectorized_test.shape

(3263, 10000)

#convert sequences into array
vectorized_train=vectorized_train.toarray()
vectorized_test=vectorized_test.toarray()

vectorized_train[0]

array([0., 0., 0., ..., 0., 0., 0.])
```

Activity 2: Training the model

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project, we are applying three classification algorithms. The best model is saved based on its performance.

Activity 1.1:KNN

A function named knn_model is created, which takes training and testing data as parameters. Inside the function, the k-Nearest Neighbors algorithm is initialized with a specified number of neighbors (k). The training data is passed to the model using the .fit() function. The model then predicts the labels of the test data using the .predict() function, and the predictions are saved in a new variable. To evaluate the model's performance, a confusion matrix and accuracy score are calculated, providing insights into the model's classification ability.

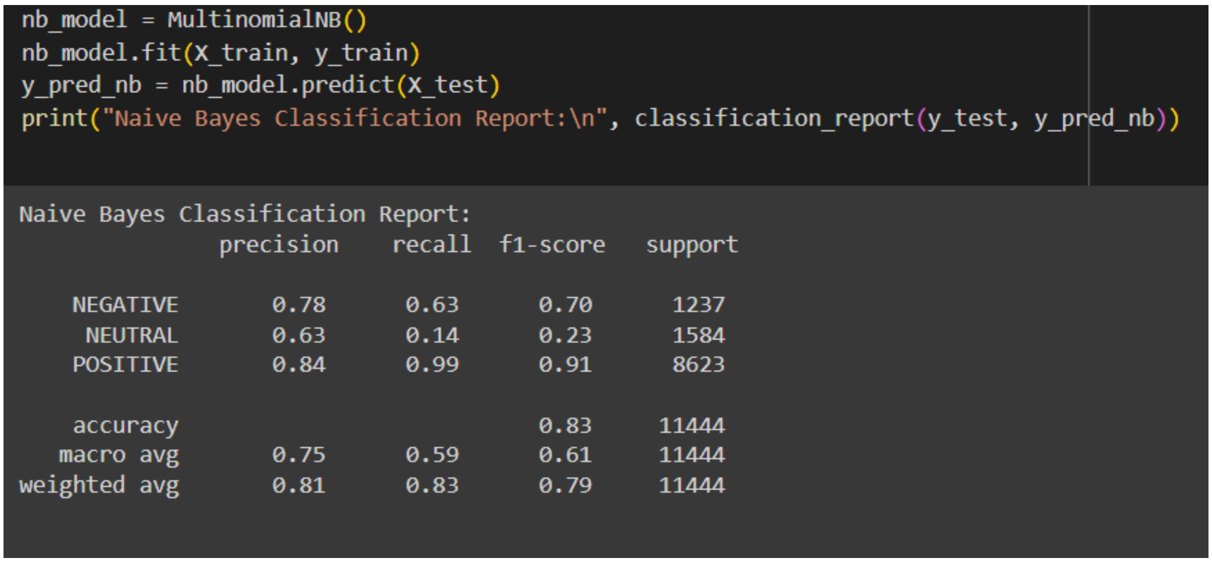


```
# KNN
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
y_pred_knn = knn_model.predict(X_test)
print("KNN Classification Report:\n", classification_report(y_test, y_pred_knn))
```

```
KNN Classification Report:
              precision    recall  f1-score   support

    NEGATIVE       0.61      0.13      0.22      1237
     NEUTRAL       0.25      0.75      0.37      1584
    POSITIVE       0.94      0.69      0.79      8623

    accuracy                           0.64     11444
   macro avg       0.60      0.52      0.46     11444
weighted avg       0.81      0.64      0.67     11444
```

Activity 1.2:Naive Bayes

A function named naive_bayes_model is created, which takes train and test data as parameters. Inside the function, the GaussianNB algorithm from the sklearn.naive_bayes module is initialized. The training data is passed to the model using the .fit() function. The test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and accuracy score are calculated.

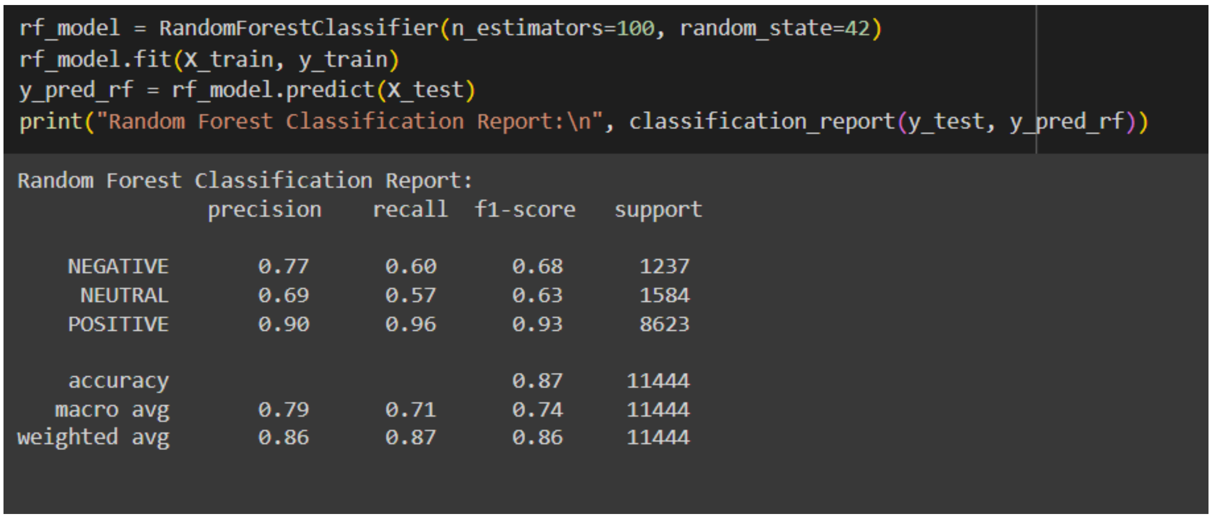


```
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
y_pred_nb = nb_model.predict(X_test)
print("Naive Bayes Classification Report:\n", classification_report(y_test, y_pred_nb))
```

```
Naive Bayes Classification Report:
              precision    recall  f1-score   support

    NEGATIVE       0.78      0.63      0.70      1237
     NEUTRAL       0.63      0.14      0.23      1584
    POSITIVE       0.84      0.99      0.91      8623

    accuracy                           0.83     11444
   macro avg       0.75      0.59      0.61     11444
weighted avg       0.81      0.83      0.79     11444
```

Activity 1.3:Random Forest

A function named random_forest_model is created, which takes training and test data as parameters. Inside the function, the RandomForestClassifier algorithm is initialized, and the training data is passed to the model using the .fit() function. The test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and accuracy score are calculated.

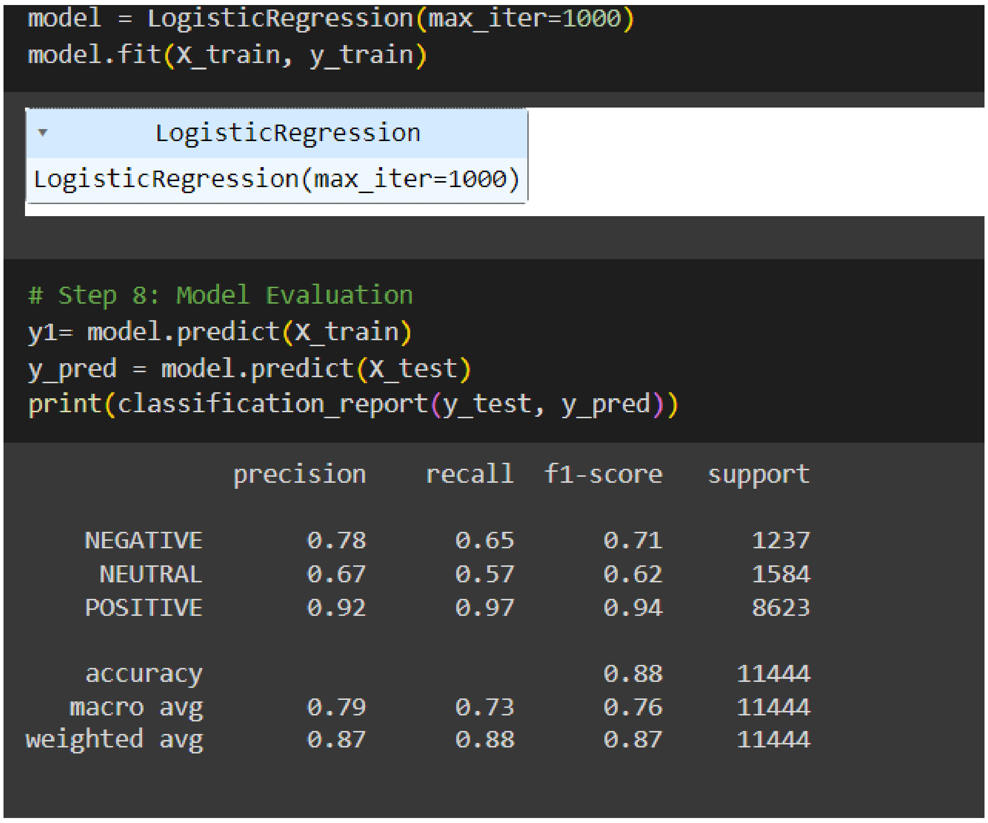


```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
print("Random Forest Classification Report:\n", classification_report(y_test, y_pred_rf))
```

```
Random Forest Classification Report:
               precision    recall  f1-score   support

    NEGATIVE       0.77      0.60      0.68      1237
     NEUTRAL       0.69      0.57      0.63      1584
    POSITIVE       0.90      0.96      0.93      8623

    accuracy                           0.87     11444
   macro avg       0.79      0.71      0.74     11444
weighted avg       0.86      0.87      0.86     11444
```

Activity 1.4: logistic regression

A function named logistic_regression_model is created, which takes train and test data as parameters. Inside the function, the LogisticRegression algorithm is initialized, and the training data is passed to the model using the .fit() function. The test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and accuracy score are calculated.

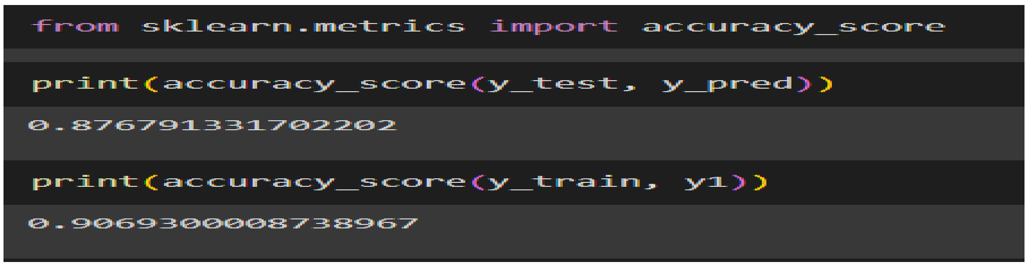


```
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

```
▾      LogisticRegression
LogisticRegression(max_iter=1000)
```

```
# Step 8: Model Evaluation
y1= model.predict(X_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

    NEGATIVE       0.78      0.65      0.71      1237
     NEUTRAL       0.67      0.57      0.62      1584
    POSITIVE       0.92      0.97      0.94      8623

    accuracy                           0.88     11444
   macro avg       0.79      0.73      0.76     11444
weighted avg       0.87      0.88      0.87     11444
```



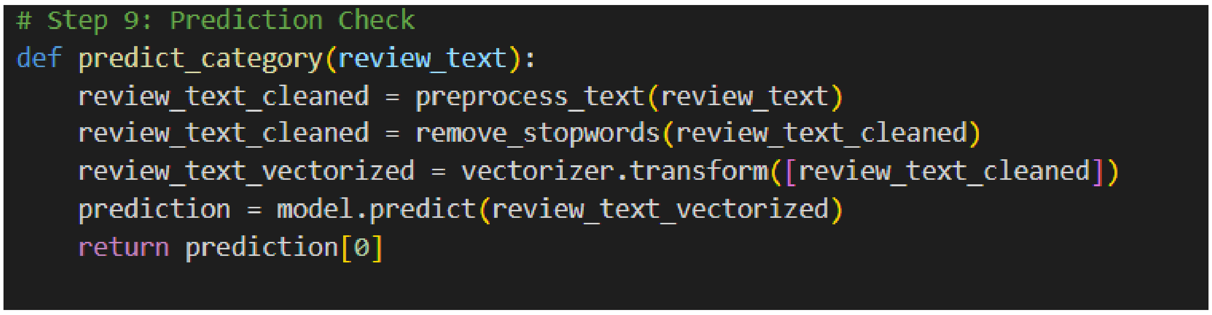
```
from sklearn.metrics import accuracy_score
```

```
print(accuracy_score(y_test, y_pred))
0.876791331702202
```

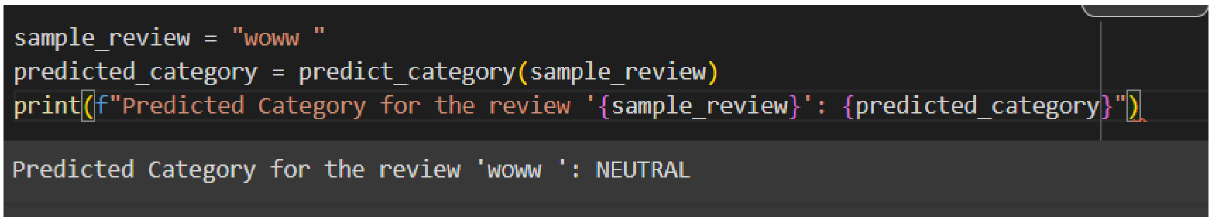
```
print(accuracy_score(y_train, y1))
0.9069300008738967
```

Activity 3: Testing the model

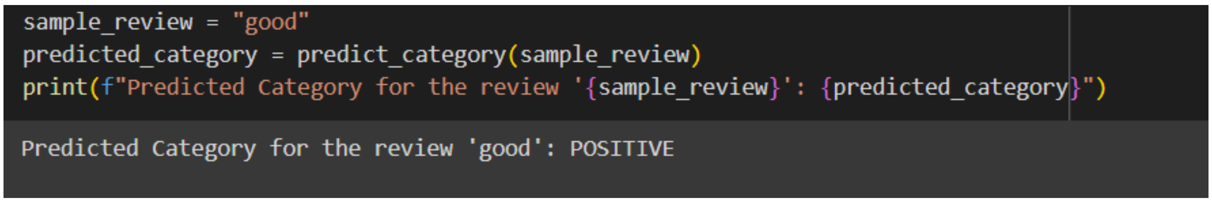
Here we have tested . With the help of the predict() function.



```
# Step 9: Prediction Check
def predict_category(review_text):
    review_text_cleaned = preprocess_text(review_text)
    review_text_cleaned = remove_stopwords(review_text_cleaned)
    review_text_vectorized = vectorizer.transform([review_text_cleaned])
    prediction = model.predict(review_text_vectorized)
    return prediction[0]
```



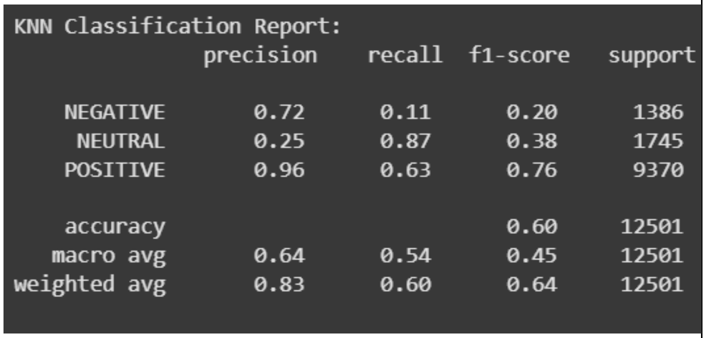
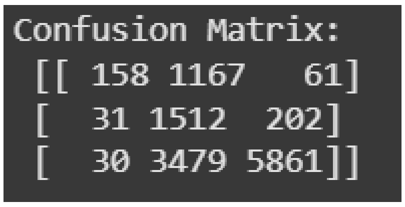
```
sample_review = "woww "
predicted_category = predict_category(sample_review)
print(f"Predicted Category for the review '{sample_review}': {predicted_category}")

Predicted Category for the review 'woww ': NEUTRAL
```



```
sample_review = "good"
predicted_category = predict_category(sample_review)
print(f"Predicted Category for the review '{sample_review}': {predicted_category}")

Predicted Category for the review 'good': POSITIVE
```

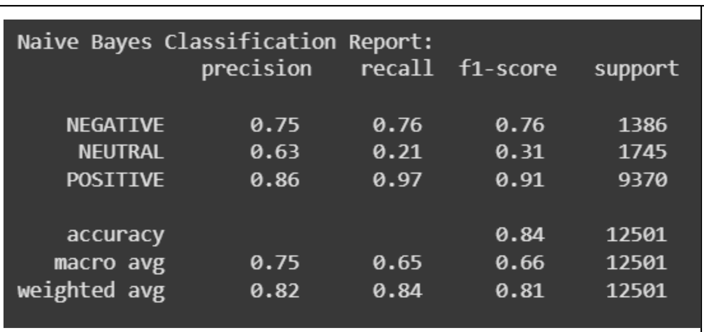
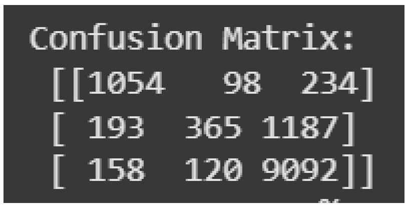
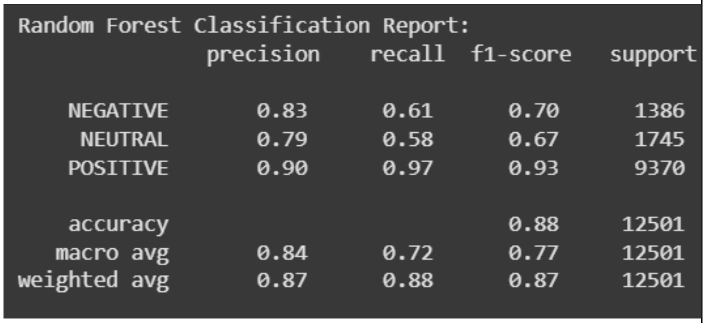
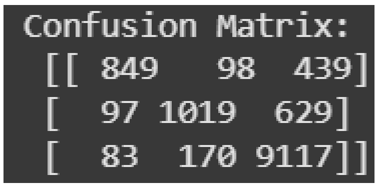
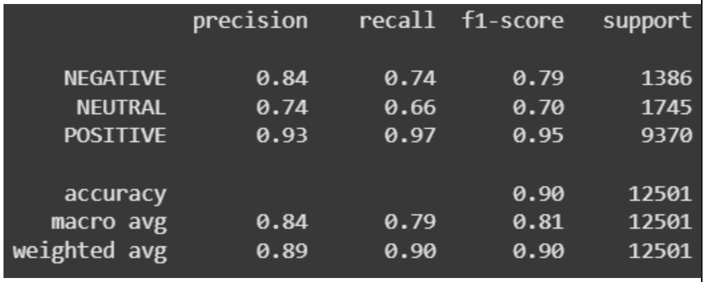
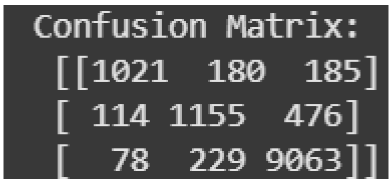
Milestone 5: Performance Testing & Hyperparameter Tuning

Activity 1: Model Validation and Evaluation Report

Model	Classification Report	F1 Score	Confusion Matrix
KNN		44.54%	

```
KNN Classification Report:
              precision    recall  f1-score   support

    NEGATIVE       0.72      0.11      0.20      1386
     NEUTRAL       0.25      0.87      0.38      1745
    POSITIVE       0.96      0.63      0.76      9370

    accuracy                           0.60     12501
   macro avg       0.64      0.54      0.45     12501
weighted avg       0.83      0.60      0.64     12501
```

```
Confusion Matrix:
 [[ 158 1167   61]
 [  31 1512  202]
 [  30 3479 5861]]
```

Naïve Bayes		**66.11%**	
Random Forest		**76.92**	
Logistic Regression		**81.1%**	

```
Naive Bayes Classification Report:
              precision    recall  f1-score   support

    NEGATIVE       0.75      0.76      0.76      1386
     NEUTRAL       0.63      0.21      0.31      1745
    POSITIVE       0.86      0.97      0.91      9370

    accuracy                           0.84     12501
   macro avg       0.75      0.65      0.66     12501
weighted avg       0.82      0.84      0.81     12501
```

```
Confusion Matrix:
 [[1054   98  234]
 [ 193  365 1187]
 [ 158  120 9092]]
```

```
Random Forest Classification Report:
              precision    recall  f1-score   support

    NEGATIVE       0.83      0.61      0.70      1386
     NEUTRAL       0.79      0.58      0.67      1745
    POSITIVE       0.90      0.97      0.93      9370

    accuracy                           0.88     12501
   macro avg       0.84      0.72      0.77     12501
weighted avg       0.87      0.88      0.87     12501
```

```
Confusion Matrix:
 [[ 849   98  439]
 [  97 1019  629]
 [  83  170 9117]]
```

```
              precision    recall  f1-score   support

    NEGATIVE       0.84      0.74      0.79      1386
     NEUTRAL       0.74      0.66      0.70      1745
    POSITIVE       0.93      0.97      0.95      9370

    accuracy                           0.90     12501
   macro avg       0.84      0.79      0.81     12501
weighted avg       0.89      0.90      0.90     12501
```

```
Confusion Matrix:
 [[1021  180  185]
 [ 114 1155  476]
 [  78  229 9063]]
```

Activity 2: Hyperparameter Tuning Documentation

Model	**Tuned Hyperparameters**
K-Nearest Neighbors (KNN)	**- n_neighbors: Number of neighbors to use (affects bias-variance tradeoff). - weights: Uniform or distance-based influence of neighbors. - metric: Distance metric used (e.g., 'euclidean', 'manhattan').**
Naive Bayes	**- var_smoothing: Portion added to variance to avoid zero division errors (for GaussianNB). Typically tuned using log-scale values (e.g., 1e-9, 1e-8).**
Random Forest	**- n_estimators: Number of trees in the forest. - max_depth: Maximum depth of each tree. - min_samples_split: Minimum samples required to split an internal node. - max_features: Number of features to consider for best split.**

Activity 3: Final Model Selection

Final Model	Reasoning
Random Forest (Example)	**Random Forest was chosen as the final model due to its superior performance in terms of accuracy and robustness on the validation set. It handles non-linear relationships and feature interactions effectively and showed the highest F1-score among all models.**

Milestone 6: Model Deployment

Activity 1: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

Activity 2.1: Building Html Page:

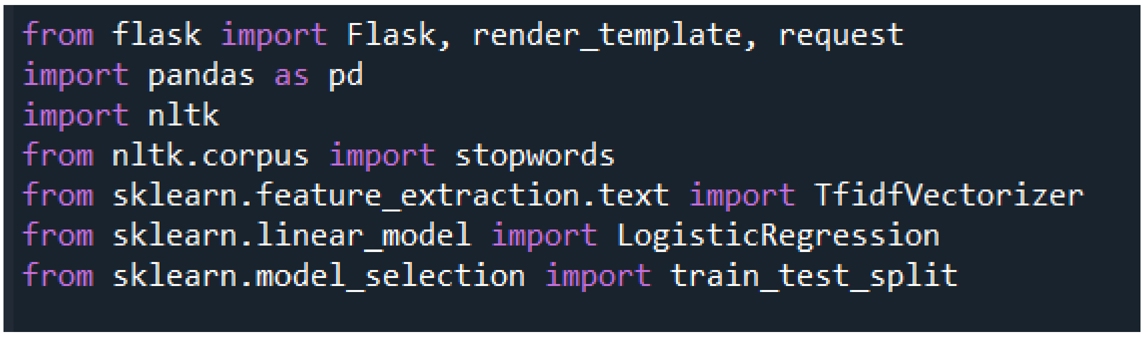
For this project create HTML file namely

- index.html

and save them in the templates folder. Refer this link for templates.

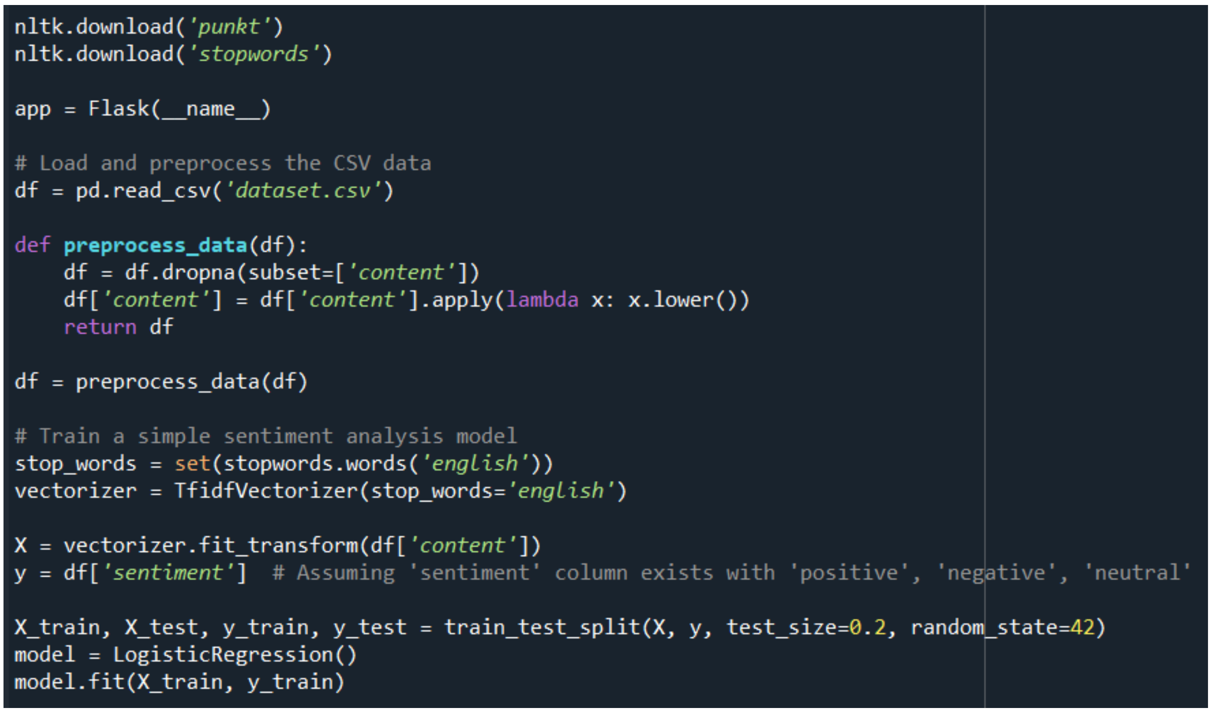
Activity 2.2: Build Python code:

Import the libraries



```python
from flask import Flask, render_template, request
import pandas as pd
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

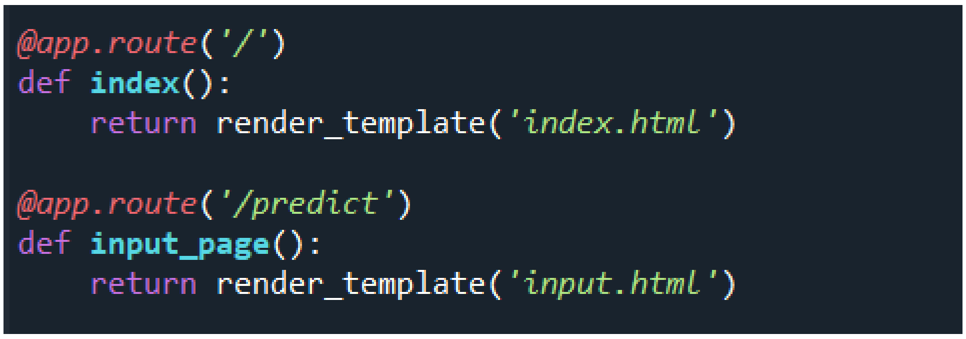
Load the saved model. Importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. The Flask constructor takes the name of the current module (name) as an argument.



```python
nltk.download('punkt')
nltk.download('stopwords')

app = Flask(__name__)

# Load and preprocess the CSV data
df = pd.read_csv('dataset.csv')

def preprocess_data(df):
    df = df.dropna(subset=['content'])
    df['content'] = df['content'].apply(lambda x: x.lower())
    return df

df = preprocess_data(df)

# Train a simple sentiment analysis model
stop_words = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words='english')

X = vectorizer.fit_transform(df['content'])
y = df['sentiment']  # Assuming 'sentiment' column exists with 'positive', 'negative', 'neutral'

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
```

Render HTML page:

Here we will be using a declared constructor to route to the HTML page that we have created earlier.

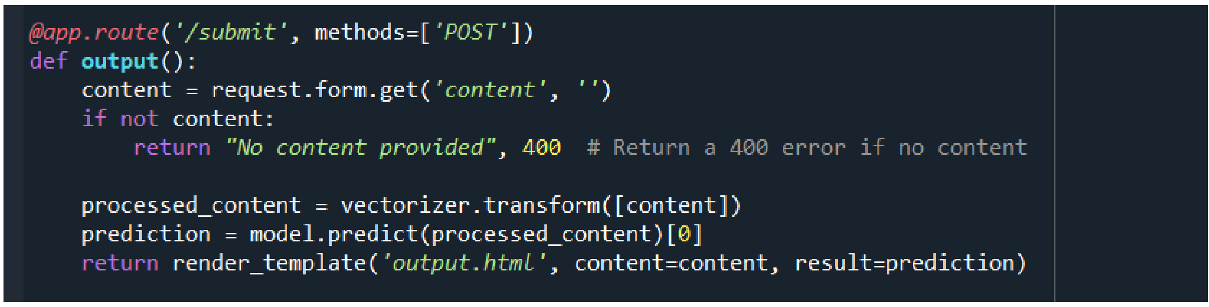


```python
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict')
def input_page():
    return render_template('input.html')
```

In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the HTML page will be rendered.

Whenever you enter the values from the html page the values can be retrieved using POST Method.

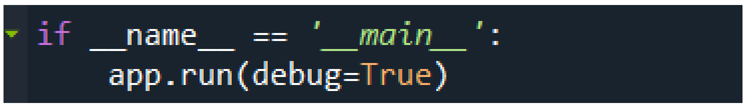
Retrieves the value from UI:



```
@app.route('/submit', methods=['POST'])
def output():
    content = request.form.get('content', '')
    if not content:
        return "No content provided", 400  # Return a 400 error if no content

    processed_content = vectorizer.transform([content])
    prediction = model.predict(processed_content)[0]
    return render_template('output.html', content=content, result=prediction)
```

Here we are routing our app to predict the () function. This function retrieves all the values from the HTML page using a Post request. That is stored in an array. This array is passed to the model. predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the input.html page earlier.

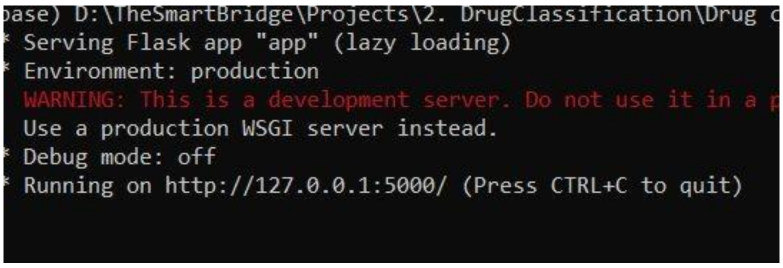
Main Function:



```
if __name__ == '__main__':
    app.run(debug=True)
```

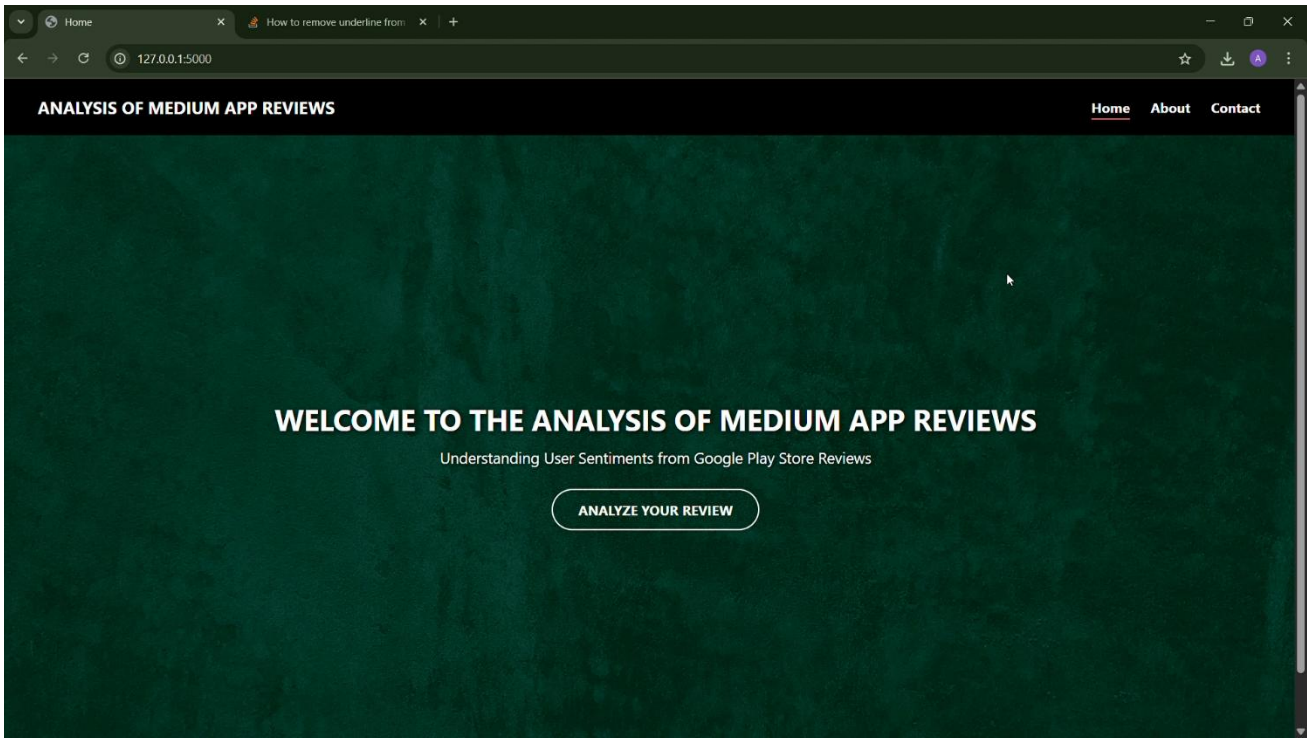
Activity 2.3: Run the web application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

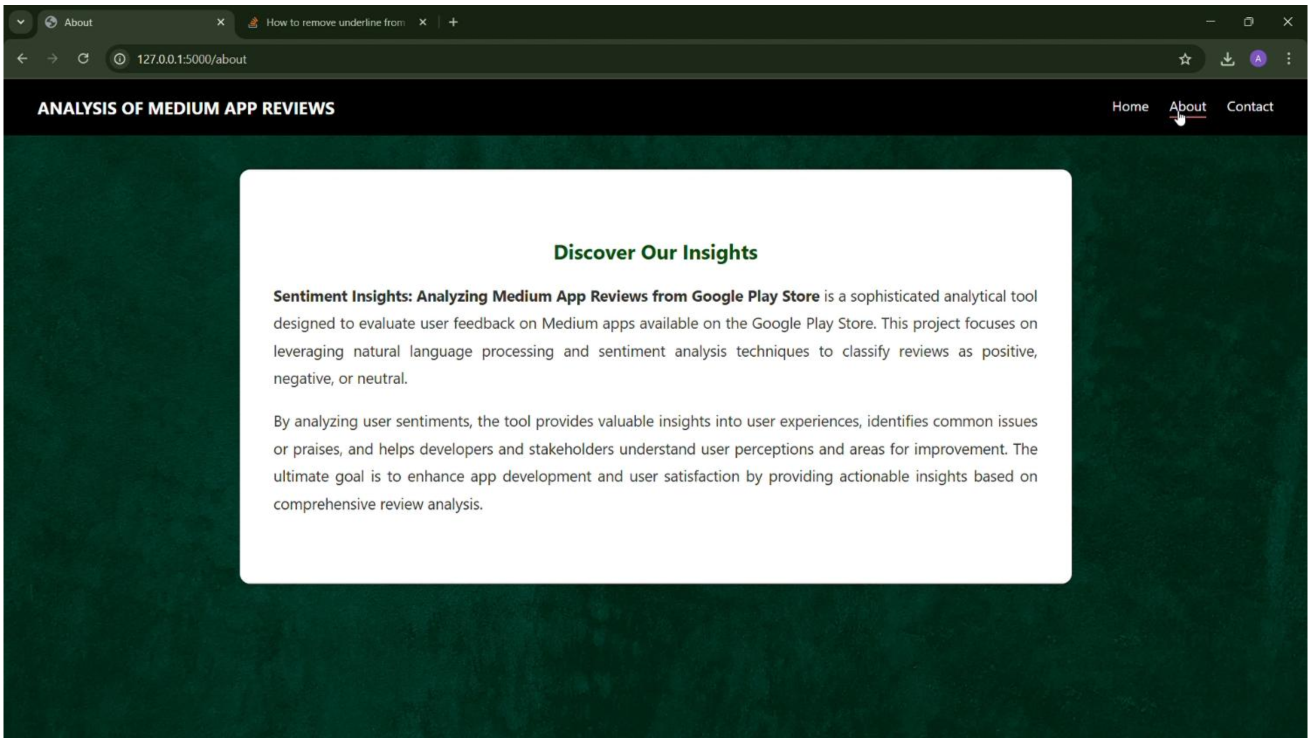


```
base) D:\TheSmartBridge\Projects\2. DrugClassification\Drug c
 Serving Flask app "app" (lazy loading)
 Environment: production
 WARNING: This is a development server. Do not use it in a p
 Use a production WSGI server instead.
 Debug mode: off
 Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result



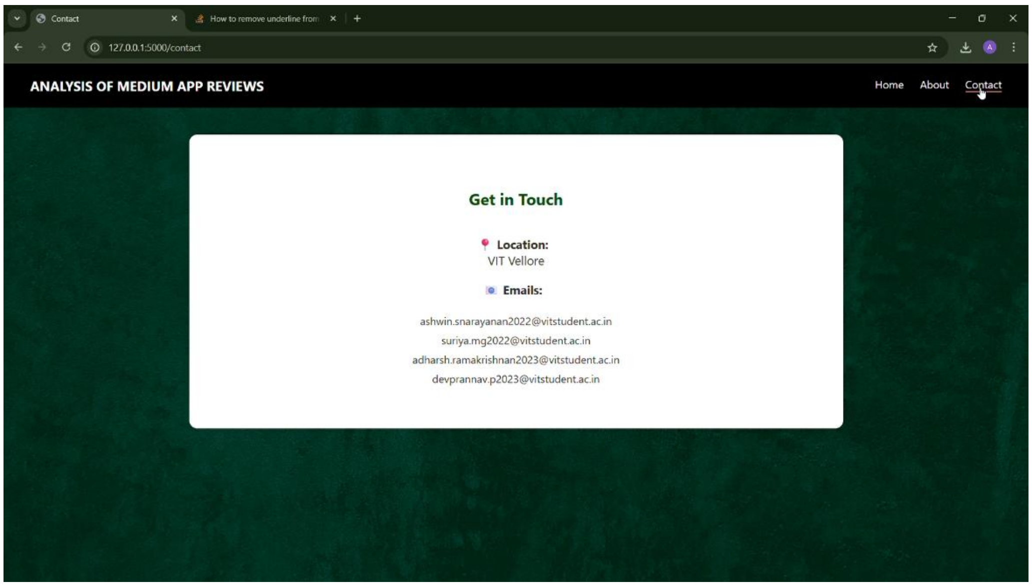
Home
How to remove underline from
127.0.0.1:5000
ANALYSIS OF MEDIUM APP REVIEWS
Home
About
Contact
WELCOME TO THE ANALYSIS OF MEDIUM APP REVIEWS
Understanding User Sentiments from Google Play Store Reviews
ANALYZE YOUR REVIEW



Discover Our Insights

Sentiment Insights: Analyzing Medium App Reviews from Google Play Store is a sophisticated analytical tool designed to evaluate user feedback on Medium apps available on the Google Play Store. This project focuses on leveraging natural language processing and sentiment analysis techniques to classify reviews as positive, negative, or neutral.

By analyzing user sentiments, the tool provides valuable insights into user experiences, identifies common issues or praises, and helps developers and stakeholders understand user perceptions and areas for improvement. The ultimate goal is to enhance app development and user satisfaction by providing actionable insights based on comprehensive review analysis.



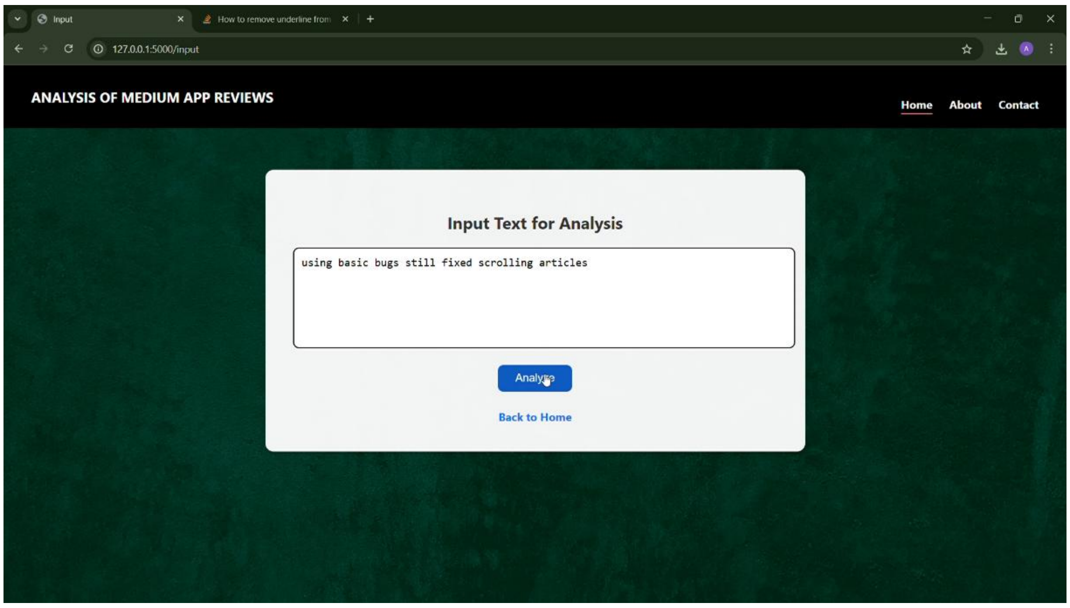
ANALYSIS OF MEDIUM APP REVIEWS

Get in Touch

Location:
VIT Vellore

Emails:

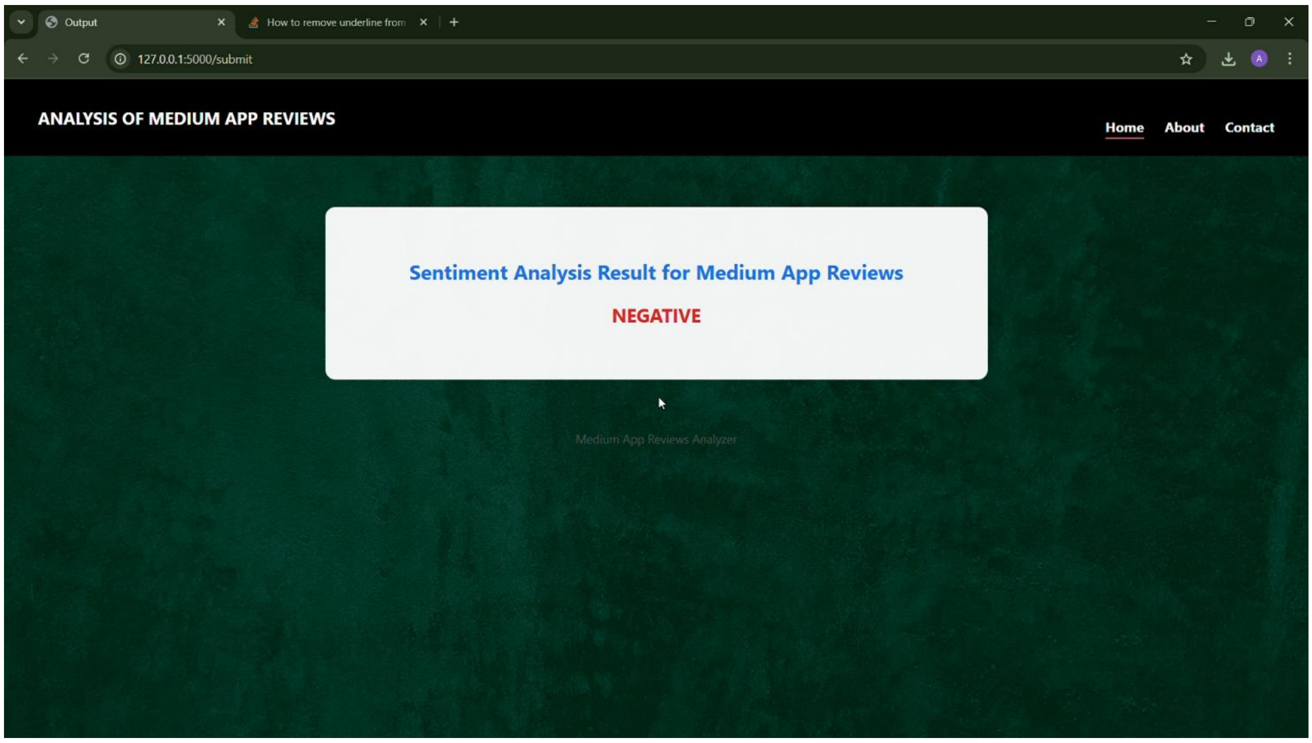
ashwin.snarayanan2022@vitstudent.ac.in
suriya.mg2022@vitstudent.ac.in
adharsh.ramakrishnan2023@vitstudent.ac.in
devprannav.p2023@vitstudent.ac.in



ANALYSIS OF MEDIUM APP REVIEWS

Input Text for Analysis

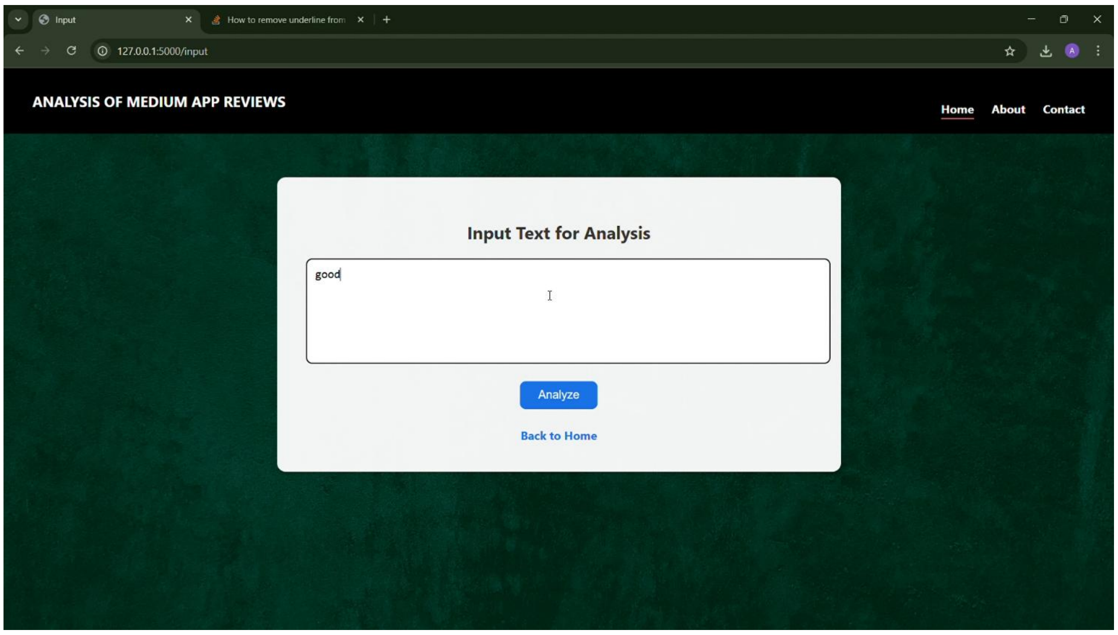
```
using basic bugs still fixed scrolling articles
```

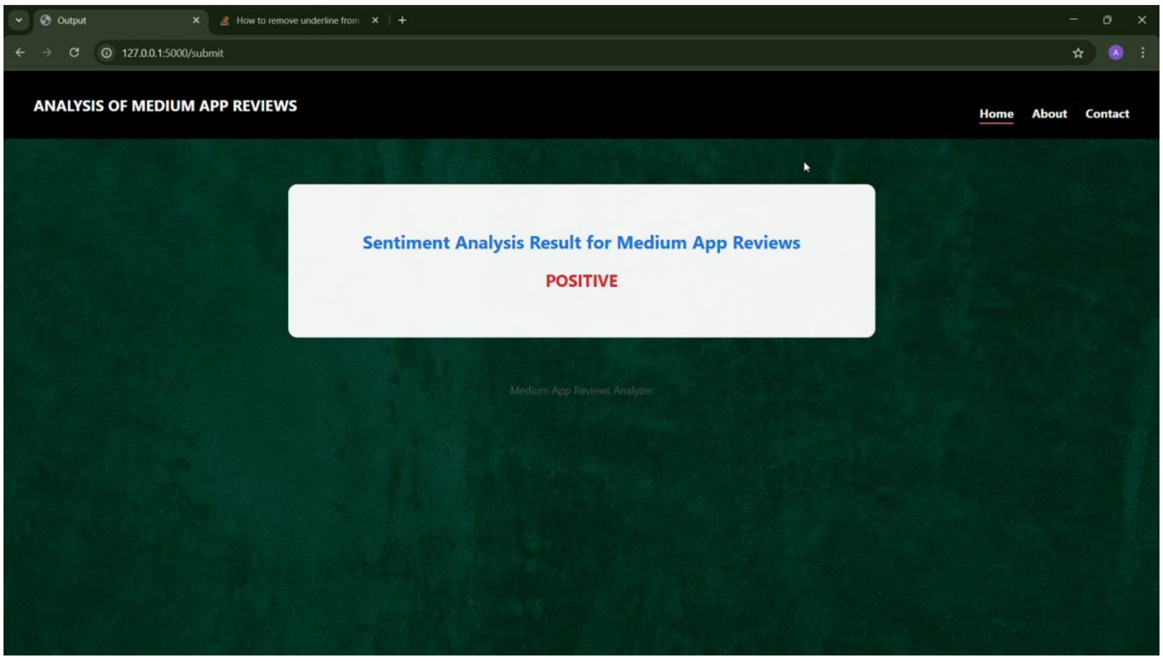


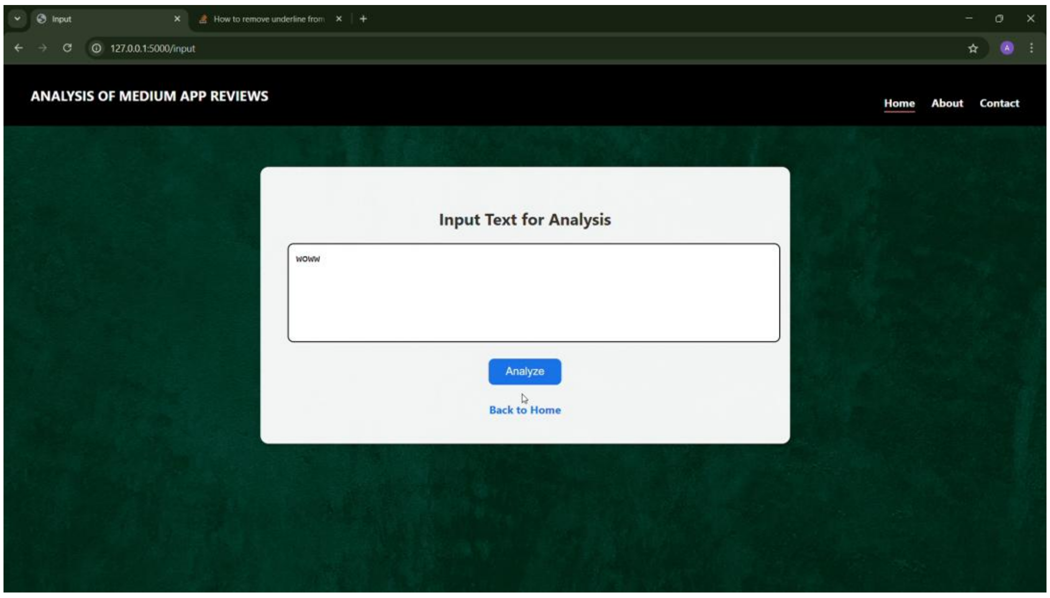
ANALYSIS OF MEDIUM APP REVIEWS

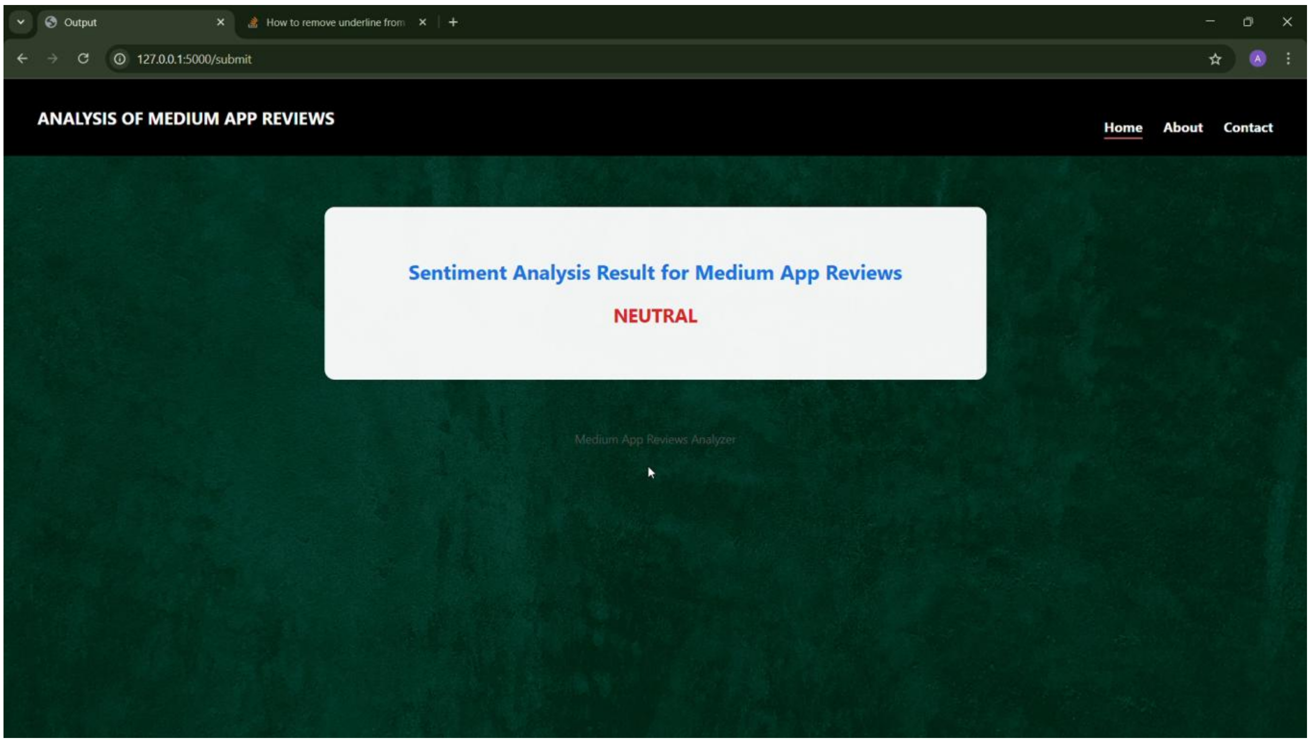
Sentiment Analysis Result for Medium App Reviews

NEGATIVE


Input
How to remove underline from
127.0.0.1:5000/input
ANALYSIS OF MEDIUM APP REVIEWS
Home
About
Contact
Input Text for Analysis
good
Analyze
Back to Home


Output
How to remove underline from
127.0.0.1:5000/submit
ANALYSIS OF MEDIUM APP REVIEWS
Home
About
Contact
Sentiment Analysis Result for Medium App Reviews
POSITIVE
Medium App Reviews Analyzer


Input
How to remove underline from
127.0.0.1:5000/input
ANALYSIS OF MEDIUM APP REVIEWS
Home
About
Contact
Input Text for Analysis
woww
Analyze
Back to Home


Output
How to remove underline from
127.0.0.1:5000/submit
ANALYSIS OF MEDIUM APP REVIEWS
Home
About
Contact
Medium App Reviews Analyzer

Sentiment Analysis Result for Medium App Reviews

NEUTRAL

Milestone 7: Project Demonstration & Documentation

GitHub Repository Link : https://github.com/Ashwin-S-Narayanan-2005/analysis-of-medium-app-reviews-from-google-play-store