

INDEX

1.	INTRODUCTION
2.	CASE STUDY DESCRIPTION
3.	BACKEND CODE OF APP
4.	EXISTING SOLUTIONS
5.	FUNCTIONALITIES
6.	CONCLUSION

Introduction

This project focuses on building a sentiment analysis web application using a machine learning approach to classify text into sentiments such as "Positive," "Negative," "Neutral," and "Irrelevant." The system enables users to input comments, select a machine learning model (Naive Bayes or Random Forest), and get a prediction of the sentiment. The application is developed using Python, Flask for the web interface, and machine learning models trained on pre-processed data. The web-based interface allows non-technical users to interact with and explore machine learning results effectively. This is used in social media platforms to analyze the sentiment of comment to flag it as inappropriate or appropriate.

Case Study Description

a) Problem Definition

Sentiment analysis is essential for determining public perception, particularly for social media platforms such as Twitter, where millions of opinions are shared daily. The goal of this application is to automatically analyze these opinions, allowing businesses, policymakers, and individuals to gain valuable insights. The project uses two primary machine learning models, Naive Bayes and Random Forest, which can handle textual data and classify it into appropriate sentiment categories.

b) System Design

- Frontend (User Interface): The web interface allows users to input a comment and select a model for sentiment prediction. A basic form is created using HTML and styled with CSS to ensure a user-friendly design. The interface provides an option to submit the form, whereupon the backend processes the text and displays the predicted sentiment.
- Backend (Processing and Machine Learning): The core of the system involves natural language processing (NLP) and machine learning techniques. The Flask web framework serves as the backend, handling requests and invoking machine learning models for sentiment prediction.
- Naive Bayes: This is a probabilistic classifier based on Bayes' Theorem. It assumes the features (words in the text) are conditionally independent given the class. It is highly efficient for text classification tasks due to its simplicity and quick convergence.
- Random Forest: This is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes for classification tasks. It's a robust model known for handling complex datasets and improving accuracy through multiple decision paths.

Software Development Model

The Increment and Iterative Development Model was chosen for this project as it allows for continuous improvement and adaptation throughout the development lifecycle. This model is ideal for projects like sentiment analysis, where machine learning models and user interfaces can be enhanced incrementally based on ongoing feedback and testing.

Key Features of the Iterative Model in This Project:

1. Initial Prototype:

- The project began with developing a basic working version of the sentiment analysis system using two machine learning models—Naive Bayes and Random Forest. This initial implementation included data preprocessing, model training, and a simple web interface using Flask.

2. Iterative Improvements:

- After the first iteration, the system was tested and refined based on performance evaluations and user feedback. Changes were made incrementally to the model pipelines, such as tuning hyperparameters, enhancing data preprocessing steps, and improving accuracy.
- The user interface was also enhanced to improve usability and design, based on feedback received during each iteration.

3. Continuous Testing and Feedback Loop:

- After each iteration, the model was re-evaluated using new datasets, and the UI was reviewed for user experience improvements. This cycle of development, testing, and refinement continued until the system met the desired accuracy and usability goals.

4. Flexible Adaptation:

- The iterative model allowed for flexibility, enabling the incorporation of additional features as the project progressed. This included the ability to select different machine learning models and the real-time display of sentiment analysis results in the user interface.

Justification for Using the Iterative Model:

The Iterative Development Model was chosen because it allows for gradual improvement in both the machine learning model performance and the user interface design. This flexibility is critical in

machine learning projects, where model accuracy can be incrementally enhanced through retraining and tuning. Additionally, the iterative process ensures that the user interface evolves based on real-world testing, resulting in a more polished and user-friendly application over time.

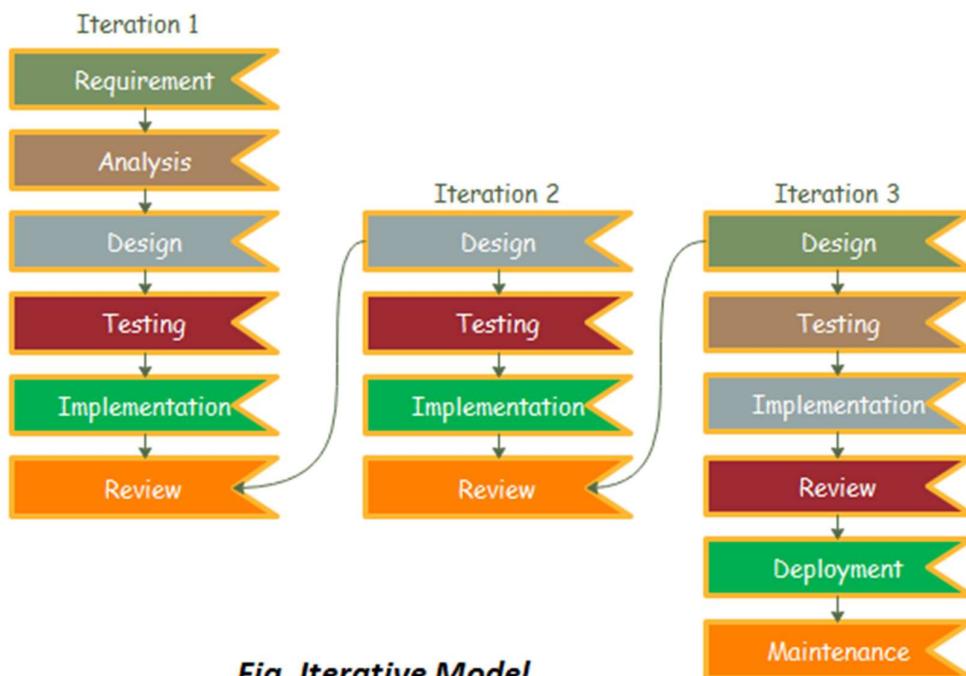


Fig. Iterative Model

Backend Code-

```
import numpy as np
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

```
from sklearn.model_selection import train_test_split  
from sklearn.feature_extraction.text import TfidfVectorizer  
from sklearn.preprocessing import LabelEncoder  
from sklearn.naive_bayes import MultinomialNB  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import accuracy_score, classification_report  
from sklearn.pipeline import Pipeline  
  
import warnings  
warnings.filterwarnings('ignore')  
  
print("Loaded libraries")  
  
nltk.download('stopwords')  
nltk.download('wordnet')  
  
lemmatizer = WordNetLemmatizer()  
stop_words = set(stopwords.words('english'))  
  
df = pd.read_csv("twitter_training.csv")  
print("Data shape:", df.shape)  
print(df.head())  
df.info()  
print("Sentiment counts:\n", df['Sentiment'].value_counts())  
  
for i in range(5):
```

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

```
print(f'{i + 1}: {df['Comment'][i]} - {df['Sentiment'][i]}')

df.dropna(inplace=True)

def preprocess(text):
    # Remove punctuation and convert to lower case
    text = re.sub(r'^\w\s]', " ", text.lower())
    # Tokenize and remove stop words, then lemmatize
    tokens = text.split()
    filtered_tokens = [lemmatizer.lemmatize(token) for token in tokens if token
not in stop_words]
    return " ".join(filtered_tokens)

df['Preprocessed Text'] = df['Comment'].apply(preprocess)

le_model = LabelEncoder()
df['Sentiment'] = le_model.fit_transform(df['Sentiment'])
print(df.head())

x_train, x_test, y_train, y_test = train_test_split(
    df['Preprocessed Text'], df['Sentiment'], test_size=0.2, random_state=42,
    stratify=df['Sentiment'])

print("Shape of X_train:", x_train.shape)
print("Shape of X_test:", x_test.shape)
```

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

```
print("Training Naive Bayes model...")  
nb_pipeline = Pipeline([  
    ('vectorizer', TfidfVectorizer()),  
    ('naive_bayes', MultinomialNB())  
])  
  
nb_pipeline.fit(x_train, y_train)  
y_pred_nb = nb_pipeline.predict(x_test)  
  
print("Naive Bayes Model Accuracy:", accuracy_score(y_test, y_pred_nb))  
print("Naive Bayes Classification Report:\n", classification_report(y_test,  
y_pred_nb))  
  
print("Training Random Forest model...")  
rf_pipeline = Pipeline([  
    ('vectorizer', TfidfVectorizer()),  
    ('random_forest', RandomForestClassifier())  
])  
  
rf_pipeline.fit(x_train, y_train)  
y_pred_rf = rf_pipeline.predict(x_test)  
  
print("Random Forest Model Accuracy:", accuracy_score(y_test, y_pred_rf))  
print("Random Forest Classification Report:\n", classification_report(y_test,  
y_pred_rf))
```

```
test_df = pd.read_csv('twitter_validation.csv')
test_text = test_df['Comment'][7]
print(f'Test Comment: {test_text} ===> True Label: {test_df['Sentiment'][7]}')
```

```
test_text_processed = [preprocess(test_text)]
predicted_label = rf_pipeline.predict(test_text_processed)
```

```
classes = ['Irrelevant', 'Neutral', 'Negative', 'Positive']
print(f'Predicted Label: {classes[predicted_label[0]]}')
```

```

5m  Loaded libraries
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
Data shape: (74682, 4)
   Id      Topic Sentiment \
0 2401 Borderlands Positive
1 2401 Borderlands Positive
2 2401 Borderlands Positive
3 2401 Borderlands Positive
4 2401 Borderlands Positive

                           Comment
0 im getting on borderlands and i will murder yo...
1 I am coming to the borders and I will kill you...
2 im getting on borderlands and i will kill you ...
3 im coming on borderlands and i will murder you...
4 im getting on borderlands 2 and i will murder ...
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74682 entries, 0 to 74681
Data columns (total 4 columns):
 #   Column   Non-Null Count Dtype  
--- 
 0   Id        74682 non-null  int64  
 1   Topic     74682 non-null  object 
 2   Sentiment  74682 non-null  object 
 3   Comment    73996 non-null  object 
dtypes: int64(1), object(3)
memory usage: 2.3+ MB
Sentiment counts:
Sentiment
Negative      22542
Positive       20832
Neutral        18318
Irrelevant     12990
Name: count, dtype: int64
1: im getting on borderlands and i will murder you all , - Positive
2: I am coming to the borders and I will kill you all, - Positive
3: im getting on borderlands and i will kill you all, - Positive
4: im coming on borderlands and i will murder you all, - Positive
5: im getting on borderlands 2 and i will murder you me all, - Positive
   Id      Topic Sentiment \
0 2401 Borderlands      3
1 2401 Borderlands      3
2 2401 Borderlands      3
3 2401 Borderlands      3
4 2401 Borderlands      3

```

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering



```
Comment \\

0 im getting on borderlands and i will murder yo...
1 I am coming to the borders and I will kill you...
2 im getting on borderlands and i will kill you...
3 im coming on borderlands and i will murder you...
4 im getting on borderlands 2 and i will murder ...

Preprocessed Text

0   im getting borderland murder
1   coming border kill
2   im getting borderland kill
3   im coming borderland murder
4   im getting borderland 2 murder
Shape of X_train: (59196,)
Shape of X_test: (14800,)
Training Naive Bayes model...
Naive Bayes Model Accuracy: 0.7177702702702703
Naive Bayes Classification Report:
precision    recall   f1-score   support
          0       0.95      0.39      0.55     2575
          1       0.64      0.91      0.75     4472
          2       0.85      0.61      0.71     3622
          3       0.69      0.81      0.75     4131

accuracy                           0.72      14800
macro avg       0.78      0.68      0.69     14800
weighted avg    0.76      0.72      0.71     14800

Training Random Forest model...
Random Forest Model Accuracy: 0.9060810810810811
Random Forest Classification Report:
precision    recall   f1-score   support
          0       0.97      0.85      0.91     2575
          1       0.92      0.92      0.92     4472
          2       0.86      0.92      0.89     3622
          3       0.89      0.91      0.90     4131

accuracy                           0.91      14800
macro avg       0.91      0.90      0.91     14800
weighted avg    0.91      0.91      0.91     14800

Test Comment: Rocket League, Sea of Thieves or Rainbow Six: Siege? I love playing all three on stream but which is the best? #stream #twitch #RocketLeague #SeaOfThieves #RainbowSixSiege #follow ===> True Label: Positive
Predicted Label: Positive
```

Flask Code-

```
import numpy as np
import pandas as pd
import re
import nltk
import joblib
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
```

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from flask import Flask, render_template, request

import warnings
warnings.filterwarnings('ignore')

app = Flask(__name__)

nltk.download('stopwords')
nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

df = pd.read_csv("twitter_training.csv")
df.dropna(inplace=True)

def preprocess(text):
    text = re.sub(r'^\w\s', " ", text.lower())
    tokens = text.split()
    filtered_tokens = [lemmatizer.lemmatize(token) for token in tokens if token
not in stop_words]
    return " ".join(filtered_tokens)
```

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

```
df['Preprocessed Text'] = df['Comment'].apply(preprocess)
```

```
le_model = LabelEncoder()  
df['Sentiment'] = le_model.fit_transform(df['Sentiment'])
```

```
x_train, x_test, y_train, y_test = train_test_split(  
    df['Preprocessed Text'], df['Sentiment'], test_size=0.2, random_state=42,  
    stratify=df['Sentiment'])  
)
```

```
nb_pipeline = Pipeline([  
    ('vectorizer', TfidfVectorizer()),  
    ('naive_bayes', MultinomialNB())  
)  
nb_pipeline.fit(x_train, y_train)  
joblib.dump(nb_pipeline, 'nb_pipeline.pkl')
```

```
rf_pipeline = Pipeline([  
    ('vectorizer', TfidfVectorizer()),  
    ('random_forest', RandomForestClassifier())  
)  
rf_pipeline.fit(x_train, y_train)  
joblib.dump(rf_pipeline, 'rf_pipeline.pkl')
```

```
nb_pipeline = joblib.load('nb_pipeline.pkl')  
rf_pipeline = joblib.load('rf_pipeline.pkl')
```

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

```
@app.route('/', methods=['GET', 'POST'])

def index():
    prediction = None
    true_label = None

    if request.method == 'POST':
        comment = request.form['comment']
        processed_comment = [preprocess(comment)]
        selected_model = request.form['model']

        if selected_model == 'naive_bayes':
            predicted_label = nb_pipeline.predict(processed_comment)
        else:
            predicted_label = rf_pipeline.predict(processed_comment)

        classes = ['Irrelevant', 'Neutral', 'Negative', 'Positive']
        prediction = classes[predicted_label[0]]

    test_df = pd.read_csv('twitter_validation.csv')
    true_label = test_df['Sentiment'][11]

    return render_template('index.html', prediction=prediction,
                           true_label=true_label)
```

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

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

```
Index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sentiment Analysis</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <h1>Sentiment Analysis</h1>
    <form method="POST" action="/">
        <textarea name="comment" rows="5" cols="50" placeholder="Enter your comment here..."></textarea><br>
        <label for="model">Select Model:</label>
        <select name="model">
            <option value="naive_bayes">Naive Bayes</option>
            <option value="random_forest">Random Forest</option>
        </select><br>
        <input type="submit" value="Analyze">
    </form>
    {{% if prediction %}}
```

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

```
<h2>Predicted Sentiment: <strong>{{ prediction }}</strong></h2>
{%- if true_label %}
<h3>True Label: <strong>{{ true_label }}</strong></h3>
{%- endif %}
{%- endif %}
</body>
</html>
```

Style.css

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: Arial, sans-serif;
    background-color: lightblue;
    color: #333;
    line-height: 1.6;
    padding: 20px;
}

h1 {
    text-align: center;
```

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

```
margin-bottom: 20px;  
color: #4A90E2;  
}  
  
h2 {
```

```
text-align: center;  
margin-top: 20px;  
color: #4CAF50;  
}
```

```
h3 {  
text-align: center;  
color: #FF5722;  
}
```

```
form {  
max-width: 600px;  
margin: 0 auto;  
background: #fff;  
padding: 20px;  
border-radius: 8px;  
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
}
```

```
textarea {  
width: 100%;
```

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

```
padding: 10px;  
border: 1px solid #ccc;  
border-radius: 4px;  
margin-bottom: 15px;  
font-size: 16px;  
}
```

```
label {  
display: block;  
margin-bottom: 5px;  
font-weight: bold;  
}
```

```
select {  
width: 100%;  
padding: 10px;  
border: 1px solid #ccc;  
border-radius: 4px;  
margin-bottom: 15px;  
font-size: 16px;  
}
```

```
input[type="submit"] {  
width: 100%;  
padding: 10px;  
background-color: #4A90E2;
```

```
color: white;  
border: none;  
border-radius: 4px;  
cursor: pointer;  
font-size: 16px;  
}  
  
input[type="submit"]:hover {  
background-color: #357ABD;  
}
```

Software Model :Agile Methodology

The **Agile Methodology** was chosen for this project to ensure flexibility, collaboration, and iterative development throughout the process. Agile allows for continuous feedback, rapid prototyping, and adaptability, making it an ideal choice for projects involving machine learning and user interface design.

Key Features of Agile in This Project:

1. Iterative and Incremental Development:

- The project was broken down into several sprints, each focused on delivering a functional part of the system. Early sprints focused on building and training the sentiment analysis models using Naive Bayes and Random Forest. Later sprints involved enhancing the UI, integrating models, and refining functionality.

2. Collaborative Approach:

- Agile emphasizes teamwork and feedback. Regular check-ins and collaboration allowed continuous refinement of both the machine learning models and the web interface based on stakeholder inputs and testing outcomes.

3. Frequent Feedback and Adaptation:

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

- The system was continuously tested during development, and feedback was gathered at the end of each sprint. This allowed adjustments to be made, such as optimizing model accuracy or improving the user experience in the Flask-based web application.

4. Flexibility for Changes:

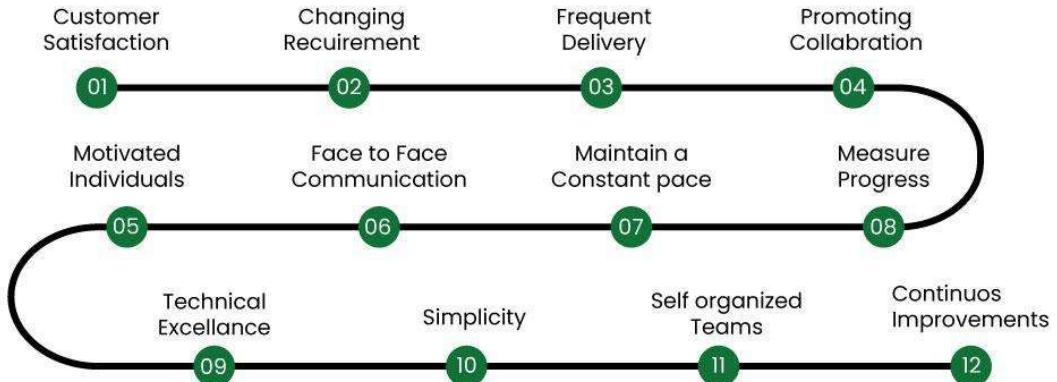
- Agile allowed flexibility to incorporate new features and improvements based on user needs. For example, additional preprocessing steps were added, and model parameters were tuned iteratively based on feedback from test results.

5. Sprints:

- **Sprint 1:** Basic sentiment analysis functionality was implemented using Naive Bayes.
- **Sprint 2:** Random Forest was added as an alternative model, with enhancements in data preprocessing.
- **Sprint 3:** The Flask web application was developed, allowing users to interact with the models.
- **Sprint 4:** Final refinements were made to improve user interface aesthetics and model accuracy.

Justification for Using Agile:

Agile methodology was selected because it offers the flexibility needed for developing a machine learning project, where frequent testing and adaptation are essential. It allowed the project to evolve with continuous feedback, ensuring that both the sentiment analysis models and the user interface met high-quality standards. Agile's iterative nature enabled a functional product to be delivered after each sprint while keeping room for ongoing improvements and adjustments.



12 Principles of Agile Methodology



Existing Solutions

Numerous sentiment analysis tools and platforms are available in the market, offering varying degrees of complexity and user interaction. Some of the most notable solutions include:

1. Google Cloud Natural Language API: Google's Natural Language API allows businesses to analyze the sentiment of text documents quickly. It offers highly scalable and accurate sentiment analysis but requires technical integration via APIs.
2. IBM Watson Natural Language Understanding: IBM Watson's API provides deep insights into the emotions and sentiments behind customer feedback. It uses sophisticated AI techniques but often comes with a high cost for businesses.
3. VADER (Valence Aware Dictionary and sentiment Reasoner): VADER is an open-source sentiment analysis tool specialized in social media sentiment. While useful for its ease of use and accessibility, it is limited in handling complex, long-form text.
4. Monkey Learn: Monkey Learn is a no-code platform offering sentiment analysis for businesses and individuals alike. It is easy to integrate but may lack customizability for specific use cases.

Each of these tools is effective in its own right, but this sentiment analysis solution offers a customized, cost-effective, and flexible alternative. It integrates Naive Bayes and Random Forest classifiers, allowing users to choose between models based on their specific needs.

Functionality

The key functionalities of this sentiment analysis application are:

1. **Text Preprocessing:** The system preprocesses the input text by cleaning, tokenizing, lemmatizing, and removing stopwords, ensuring that the machine learning models work on refined data.
2. **Multiple Model Selection:** Users can choose between two machine learning models—Naive Bayes and Random Forest—depending on their preference for speed or accuracy in predictions.
3. **Real-time Sentiment Prediction:** Users input comments, and the system predicts the sentiment in real-time, classifying the text into Irrelevant, Neutral, Negative, or Positive sentiments.
4. **True Sentiment Comparison:** During testing or evaluation, users can also compare the predicted sentiment with the true sentiment from a pre-labeled dataset.
5. **User-Friendly Interface:** A simple and intuitive web interface developed using Flask allows for easy user interaction without the need for technical expertise.

SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

Screenshots of Output

Sentiment Analysis

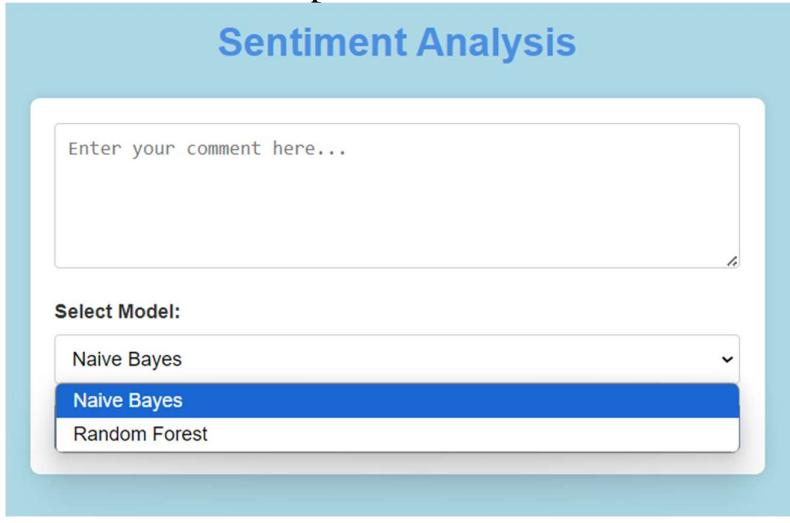
Enter your comment here...

Select Model:

Naive Bayes

Naive Bayes

Random Forest



Sentiment Analysis

So I spent a few hours making something for fun. . . If you don't know I am a HUGE RhondaR fan and Maya is one of my favorite characters. So I decided to make myself a wallpaper for my PC. . Here is the original image versus the creation I made :) Enjoy! pic.twitter.com/mLsI5wf9Jg

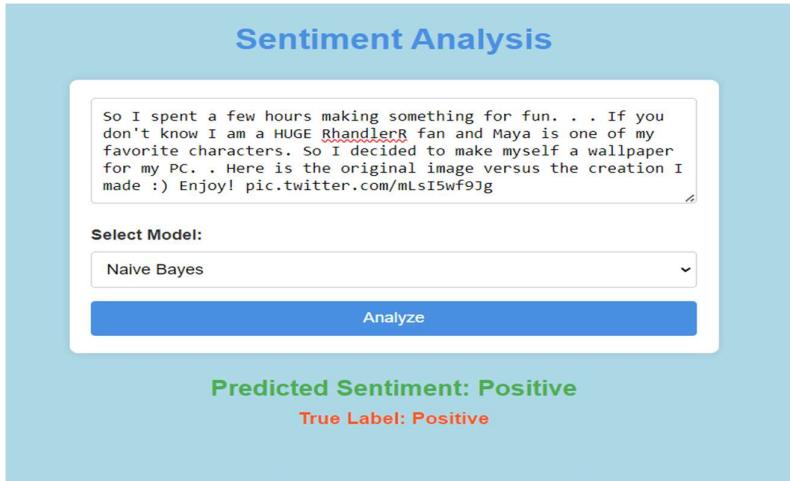
Select Model:

Naive Bayes

Analyze

Predicted Sentiment: Positive

True Label: Positive



SVKM's NMIMS University
School of Technology Management & Engineering
COURSE: Software Engineering

Dataset Description: -

```
>>> pd.read_csv('labeled_tweets.csv').info()
RangeIndex: 74682 entries, 0 to 74681
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Id          74682 non-null   int64  
 1   Topic        74682 non-null   object  
 2   Sentiment    74682 non-null   object  
 3   Comment      73996 non-null   object  
dtypes: int64(1), object(3)
memory usage: 2.3+ MB
Sentiment counts:
 Sentiment
Negative      22542
Positive       20832
Neutral        18318
Irrelevant     12990
Name: count, dtype: int64
```

This explains the imbalance in the classes as well as the datatype of each column to help us understand how the ML model would give the output.

This also shows us which ML model to apply for the given problem.

Link to the dataset: -

<https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis/data>

Conclusion

This sentiment analysis web application effectively demonstrates the integration of natural language processing, machine learning, and web technologies to provide a real-time, interactive user experience. By allowing users to select between two powerful machine learning models, Naive Bayes and Random Forest, the system is flexible and adaptable to different text classification tasks.

The intuitive Flask-based interface ensures that both technical and non-technical users can engage with the system effortlessly. Compared to existing solutions, this application offers a customizable, user-driven

alternative that can be scaled for various use cases like social media sentiment analysis, customer feedback monitoring, and more. Future improvements could include expanding the dataset for more advanced sentiment categories and enhancing the interface for better usability.