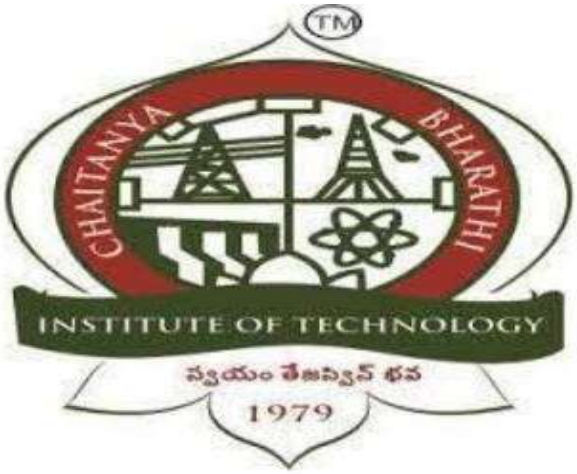
INTRODUCTION TO MACHINE LEARNING-22AMC03

ASSIGNMENT -2

REPORT

ON

“Twitter Sentiment Analysis”



Submitted to

Dr. Y C A PADMANABHA REDDY

Department of Artificial Intelligence and machine learning

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)
(Affiliated to Osmania University)

Gandipet, Hyderabad- 500075

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

An Autonomous Institute I Affiliated to Osmania University

Kokapet Village, Gandipet Mandal, Hyderabad, Telangana-500075, www.cbit.ac.in

Approved by | Affiliated to | UGC Autonomous | 10 Programs Accredited by | Grade A++ in | All India Ranking 151-200 Band

COMMITTED TO RESEARCH, INNOVATION AND EDUCATION 46 years

SUBMITTED BY:

Ch Laxmi Prashasthi -160123729303

Kotla Geethika -160123729304

Mukassir Ahmed Farooqui- 160123729305

Abstract

This project is about analyzing the sentiment of tweets using machine learning. The goal is to classify tweets as **positive** or **negative**. First, tweets are cleaned and processed using natural language processing methods. Then, we use a TF-IDF vectorizer to convert the text into numbers that a machine learning model can understand. A **Logistic Regression** model is used to predict the sentiment. The app is built using **Streamlit**, where users can input their own text or use sample tweets to see the results. This project shows how machine learning can be used in a simple and effective way to analyze public opinion on social media.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to all those who contributed to the completion of this Twitter Sentiment Analysis project. Special thanks to my faculty member **Mr. Y C A PADMANABHA REDDY** for his valuable guidance and support throughout the analysis process. I also appreciate the creators of the open-source tools and libraries that made data processing, model training, and visualization possible. Lastly, I would like to acknowledge the contributions made toward preparing and refining the dataset, which played a crucial role in improving the accuracy and reliability of the sentiment predictions.

INTRODUCTION

With the rapid growth of social media, understanding public sentiment has become an important aspect of decision-making for businesses and analysts. Twitter, in particular, is a valuable platform for gauging public opinion on various topics.
This project is built around the task of **Twitter Sentiment Analysis**, where we analyze the emotions expressed in tweets. The core of the system relies on a **machine learning model** trained on labeled tweet data. The steps include:

- **Text Preprocessing**: Cleaning the tweet by removing special characters, converting to lowercase, and filtering out stopwords.
- **Feature Extraction**: Using TF-IDF (Term Frequency-Inverse Document Frequency) to convert the processed text into numerical format.
- **Model Prediction**: Using a pre-trained logistic regression model to classify the tweet sentiment as positive or negative.
- **Deployment**: The entire functionality is wrapped in a user-friendly web interface using **Streamlit**, with support for both manual text input and preloaded sample tweets from a CSV.

This simple yet powerful machine learning application demonstrates how sentiment analysis can provide insights from unstructured textual data and can be extended to many real-world scenarios like customer feedback analysis, brand monitoring, and more.

PROGRAM:

1.Twitter_Sentiment_Analysis.ipyb

```
import pandas as pd
import numpy as np
import re
import nltk
import pickle

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from nltk.corpus import stopwords
from sklearn.metrics import classification_report, accuracy_score

# Download stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

# Load dataset
df = pd.read_csv("twitter_training.csv")
df = df[['text', 'label']]

# Basic preprocessing
def preprocess_text(text):
    text = re.sub(r"[^a-zA-Z]", " ", text)  # Remove non-alphabetic characters
    text = text.lower()
    tokens = text.split()
    tokens = [word for word in tokens if word not in stop_words]
    return " ".join(tokens)

df['cleaned_text'] = df['text'].apply(preprocess_text)
```

```
# Convert labels to numerical
df['label'] = df['label'].map({"Positive": 1, "Negative": 0})

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(df['cleaned_text'], df['label'], test_size=0.2, random_state=42)

# TF-IDF Vectorizer
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vect = vectorizer.fit_transform(X_train)
X_test_vect = vectorizer.transform(X_test)

# Model
model = LogisticRegression()
model.fit(X_train_vect, y_train)

# Predictions and evaluation
y_pred = model.predict(X_test_vect)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Save model and vectorizer
with open("model.pkl", "wb") as f:
    pickle.dump(model, f)

with open("vectorizer.pkl", "wb") as f:
    pickle.dump(vectorizer, f)
```

2.Twitter_client.py

```
import pandas as pd
import random

def fetch_user_tweets(username: str = None, max_tweets: int = 5):
    """
    Load tweets from a local CSV file.
    Ignores the `username` and just returns random samples from the file.
    """
    # Load the CSV file with sample tweets
    df = pd.read_csv("sample_tweets.csv")

    # Randomly select tweets
    sampled = df.sample(n=max_tweets).reset_index(drop=True)

    # Convert to list of dicts (same format as original)
    tweets = []
    for _, row in sampled.iterrows():
        tweets.append({
            "text": row["text"],
            "created_at": None  # Not available in sample data
        })
    return tweets
```

3.app.py

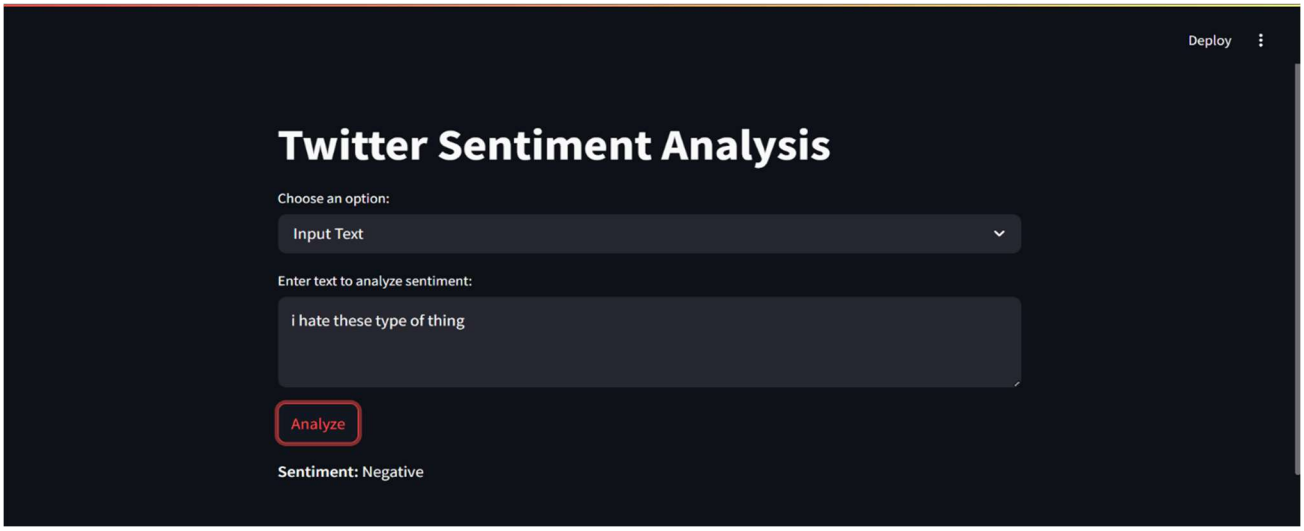
```
import streamlit as st
import pickle
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
import nltk
from twitter_client import fetch_user_tweets  # Now fetches from CSV

# Download stopwords once
def load_stopwords():
    nltk.download('stopwords')
    return stopwords.words('english')

# Load model and vectorizer once
@st.cache_resource
def load_resources():
    stop_words = load_stopwords()
    with open('model.pkl', 'rb') as model_file:
        model = pickle.load(model_file)
    with open('vectorizer.pkl', 'rb') as vectorizer_file:
        vectorizer = pickle.load(vectorizer_file)
    return stop_words, model, vectorizer

# Sentiment prediction
def predict_sentiment(text, model, vectorizer, stop_words):
    text = re.sub('[^a-zA-Z]', ' ', text)
    text = text.lower().split()
    text = [word for word in text if word not in stop_words]
    text = ' '.join(text)
```

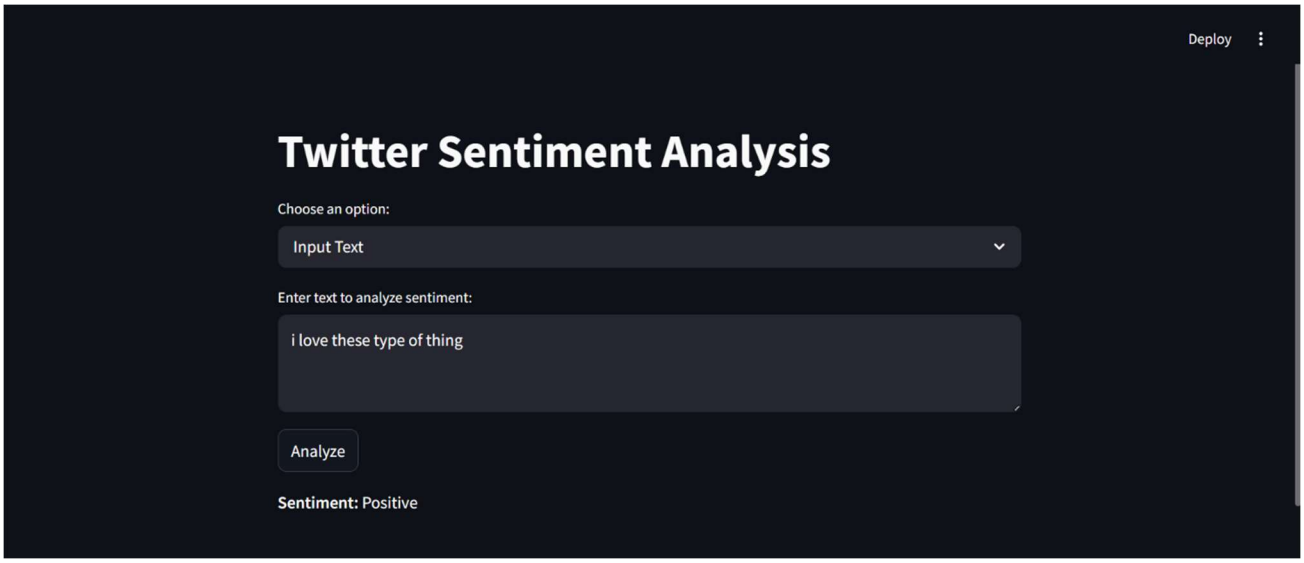
```
    vect_text = vectorizer.transform([text])
    label = model.predict(vect_text)[0]
    return "Negative" if label == 0 else "Positive"

# Colored tweet card
def display_card(tweet_text, sentiment):
    color = "#2ecc71" if sentiment == "Positive" else "#e74c3c"
    st.markdown(
        f"""
        <div style="background-color: {color}; padding: 12px; border-radius: 8px; margin: 8px 0;">
            <strong style="color: white;">{sentiment} Sentiment</strong><br>
            <span style="color: white;">{tweet_text}</span>
        </div>
        """,
        unsafe_allow_html=True
    )

# Main App
def main():
    st.title("Twitter Sentiment Analysis")

    # Load model/vectorizer/stopwords
    stop_words, model, vectorizer = load_resources()

    # User selects mode
    option = st.selectbox("Choose an option:", ["Input Text", "Use Sample Tweets"])

    if option == "Input Text":
        text_input = st.text_area("Enter text to analyze sentiment:")
        if st.button("Analyze"):
            if text_input.strip():
                sentiment = predict_sentiment(text_input, model, vectorizer, stop_words)
```

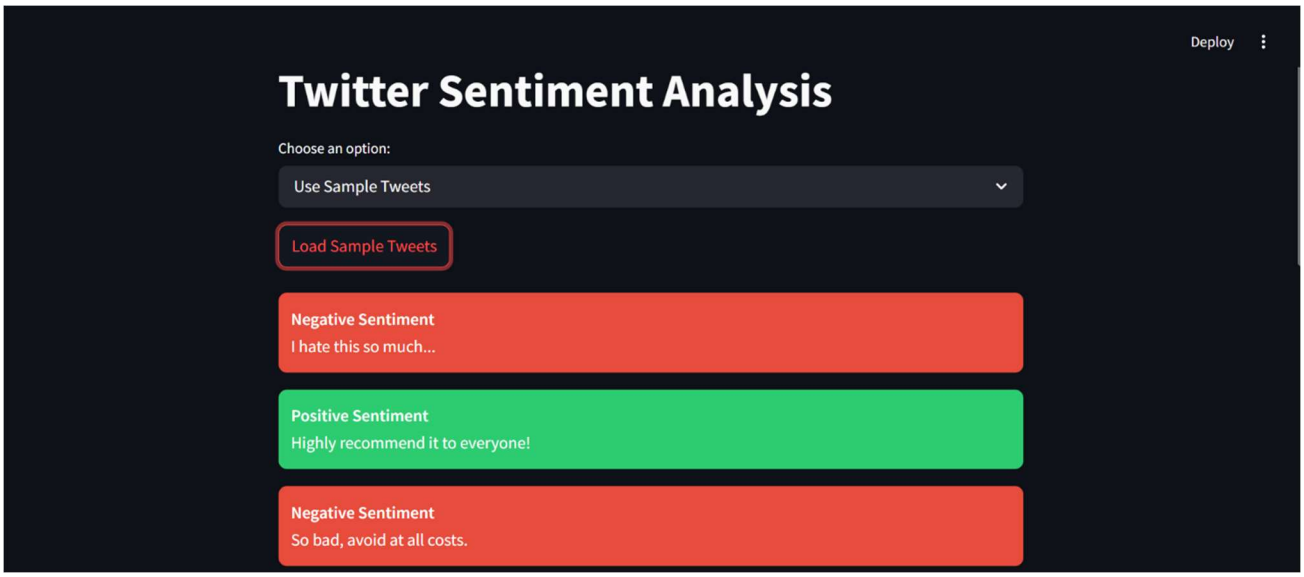
```
            st.write(f"**Sentiment:** {sentiment}")
        else:
            st.warning("Please enter some text.")
    else:
        if st.button("Load Sample Tweets"):
            try:
                tweets = fetch_user_tweets(max_tweets=10)  # you can change this
                for tweet in tweets:
                    sentiment = predict_sentiment(tweet['text'], model, vectorizer, stop_words)
                    display_card(tweet['text'], sentiment)
            except Exception as e:
                st.error(f"Error loading sample tweets: {e}")

if __name__ == "__main__":
    main()
```

OUTPUT:

Accuracy: 0.775615625







Sentiment Analysis using TF-IDF and Logistic Regression

This project focuses on binary sentiment classification of tweets using **Logistic Regression**, a reliable and interpretable machine learning algorithm. The goal is to identify whether a tweet expresses a **positive** or **negative** sentiment, based on its textual content.

Key Steps in the ML Pipeline:

1. Text Preprocessing

- Remove unwanted characters like punctuation and digits using regular expressions.
- Convert all text to lowercase for consistency.
- Remove common English **stopwords** (e.g., *is, the, this, was*) using NLTK.
- This reduces noise and focuses the model on informative words.

2. TF-IDF Vectorization

- **TF-IDF (Term Frequency-Inverse Document Frequency)** transforms raw text into meaningful numeric features.
- It highlights words that are important in individual tweets but not too common in the full dataset.
- This representation helps the logistic model distinguish sentiment-driving words effectively.

3. Logistic Regression Model

- A **supervised learning** algorithm that learns the probability of a class (positive/negative) from input features.
- Works well for linearly separable data and is fast to train and predict.
- Provides high interpretability and can be deployed easily in real-time applications.

4. Model Evaluation

- The dataset is split into training and test sets (e.g., 80% train / 20% test).
- Performance is measured using **accuracy** and optionally precision/recall or a confusion matrix.
- Logistic Regression delivered solid results with fast computation and minimal overfitting.

5. Saving and Using the Model

- The trained model and vectorizer are saved as `.pkl` files using Python's `pickle` module.
- These files are loaded in the Streamlit app for real-time sentiment analysis of user input or sample tweets.

Conclusion

The Twitter Sentiment Analysis project successfully demonstrates how machine learning can be used to analyze and classify the emotional tone of social media content. By leveraging text preprocessing techniques and TF-IDF vectorization, the project was able to convert unstructured tweet data into meaningful numerical features suitable for machine learning.

The Logistic Regression model, chosen for its simplicity and efficiency, performed well in predicting whether tweets express positive or negative sentiments. The model's accuracy and speed make it an excellent choice for real-time applications, especially in web-based environments.

Deploying the model using **Streamlit** further enhanced the project by providing an interactive interface that allows users to input custom text or analyze sample tweets with just a few clicks. The system’s ability to provide immediate visual feedback makes it both user-friendly and informative.

Overall, the project highlights the effectiveness of combining basic NLP techniques with a well-tuned logistic model to extract useful insights from short, informal te