S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Post-lab

Aim: Perform sentiment analysis using pre-labeled datasets like IMDB rating , amazon reviews, twitter data.

Name of Student: _____

Roll No.: _____

Semester/Year: IV Semester /VII Year

Academic Session: _____

Date of Performance: _____

Date of Submission: _____

AIM: Perform sentiment analysis using pre-labeled datasets like IMDB rating , amazon reviews, twitter data.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

1. To understand the concept and workflow of sentiment analysis in Natural Language Processing (NLP).
2. To preprocess textual data using techniques such as tokenization, stop-word removal, and stemming/lemmatization.
3. To train and evaluate sentiment classification models using pre-labeled datasets such as IMDB reviews, Amazon product reviews, and Twitter data.

HARDWARE AND SOFTWARE REQUIRMENTS:

Hardware Requirement:

- Ram -4 Gb
- SSD – 512 Gb
- Processor – Intel Core

Software Requirement:

- Google colab
- Python

THEORY:

Sentiment analysis is a subfield of Natural Language Processing (NLP) that focuses on determining the emotional tone behind a body of text. It is used to analyze user opinions, attitudes, and emotions expressed in sources such as social media posts, reviews, and comments. The primary goal is to classify the sentiment as positive, negative, or neutral, or in some cases, into more specific emotions like joy, anger, sadness, or fear.

The process of sentiment analysis generally involves several steps:

1. Data Collection: A pre-labeled dataset containing text samples and their associated sentiments is used for model training. Examples include the IMDB movie reviews, Amazon product reviews, and Twitter emotion datasets.
2. Data Preprocessing: The text data is cleaned and transformed to remove noise such as punctuation, stopwords, and special symbols. In this project, the data is balanced by sampling equal numbers of examples from each emotion class to ensure fair learning.
3. Feature Extraction (TF-IDF): The TF-IDF (Term Frequency–Inverse Document Frequency) vectorizer converts textual data into numerical features by giving higher importance to rare but

meaningful words. This helps the machine learning model capture the significance of each word in context.

4. Model Training (Logistic Regression): The Logistic Regression algorithm is used as a classifier to learn patterns from the vectorized text and predict the sentiment of unseen data. It is simple, interpretable, and efficient for text classification tasks.
5. Model Evaluation: The trained model is evaluated using metrics such as precision, recall, F1-score, and accuracy. These metrics provide insights into how well the model distinguishes between different emotions or sentiments.
6. Model Deployment and Prediction: The trained model is saved and reloaded using Joblib, allowing it to make real-time predictions. For any new input text, the model predicts the most likely sentiment and provides a confidence score.

This end-to-end pipeline demonstrates a practical machine learning workflow for text-based emotion recognition, integrating NLP techniques with supervised learning. Such systems can be used in real-world applications like customer feedback analysis, brand monitoring, social media analytics, and public opinion mining.

CODE:

```
# Step 1: Setup and Initialization

print(" Step 1: Installing libraries and importing modules...")
import pandas as pd
from google.colab import drive
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report
import joblib

drive.mount('/content/drive', force_remount=True)
print("\nGoogle Drive mounted.")

# Step 2: Load, Diagnose, and Balance the Dataset (Corrected)
print("\nStep 2: Loading and preparing the dataset...")
file_path = '/content/drive/MyDrive/Kunal/tweet_emotions.csv'
try:
    df = pd.read_csv(file_path)
except FileNotFoundError:
    print(f"ERROR: File not found at {file_path}")
    exit()
```

```
print("\nActual column names in your file are:")
print(df.columns.tolist())

TEXT_COLUMN = 'content'
LABEL_COLUMN = 'sentiment'

print(f"\nBalancing the dataset based on the '{LABEL_COLUMN}' column...")
n_samples_per_class = 4000
# THE FIX IS HERE: 'include_groups=False' has been removed from the .apply() call.
balanced_df = df.groupby(LABEL_COLUMN).apply(
    lambda x: x.sample(n=min(len(x), n_samples_per_class))
).reset_index(drop=True)

print("\nDistribution of emotions AFTER balancing:")
print(balanced_df[LABEL_COLUMN].value_counts())

X = balanced_df[TEXT_COLUMN]
y = balanced_df[LABEL_COLUMN]

# Step 3: Train and Evaluate the Improved Model
print("\nStep 3: Training the improved machine learning model...")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42, stratify=y)
model_pipeline = make_pipeline(
    TfidfVectorizer(),
    LogisticRegression(max_iter=1000, random_state=42)
)
model_pipeline.fit(X_train, y_train)
print("\n Improved model trained successfully!")
print("\nNew Model Performance Report:")
y_pred = model_pipeline.predict(X_test)
print(classification_report(y_test, y_pred))

# Step 4: Save the Final Model
print("\nStep 4: Saving the trained model to your Google Drive...")
model_save_path = '/content/drive/MyDrive/Kunal/emotion_classifier_model.joblib'
joblib.dump(model_pipeline, model_save_path)
print(f" Model saved successfully at: {model_save_path}")

# Step 5: Load and Use the Model for Real-Time Predictions
print("\nStep 5: Loading model and preparing for predictions...")
try:
    loaded_model = joblib.load(model_save_path)
    print("Model loaded successfully. Ready to use!")
except FileNotFoundError:
    print(f"ERROR: Saved model not found at {model_save_path}")
```

```
    exit()

def predict_emotion(text_input):
  prediction = loaded_model.predict([text_input])
  probabilities = loaded_model.predict_proba([text_input])
  confidence = probabilities.max()
  print(f'\nInput Text: "{text_input}"')
  print(f'\nPredicted Emotion: {prediction[0]} (Confidence: {confidence:.2%})')

print("\n\n--- Your AI is ready! Let's test it. ---\n")
predict_emotion("I'm not sure how to feel about the news I just received.")
predict_emotion("This is the worst day, everything is going wrong and I feel awful.")
predict_emotion("I just won the lottery! I can't believe it! This is amazing!")
predict_emotion("My presentation is tomorrow morning and I am so nervous.")
```

OUTPUT (SCREENSHOT):

New Model Performance Report:

	precision	recall	f1-score	support
anger	0.00	0.00	0.00	22
boredom	0.00	0.00	0.00	36
empty	0.17	0.01	0.01	165
enthusiasm	0.00	0.00	0.00	152
fun	0.19	0.06	0.10	355
happiness	0.32	0.46	0.38	800
hate	0.52	0.22	0.31	265
love	0.46	0.48	0.47	769
neutral	0.27	0.38	0.31	800
relief	0.29	0.08	0.12	305
sadness	0.33	0.44	0.38	800
surprise	0.23	0.09	0.13	437
worry	0.25	0.34	0.29	800
accuracy			0.32	5706
macro avg	0.23	0.20	0.19	5706
weighted avg	0.30	0.32	0.29	5706

```
Input Text: "I'm not sure how to feel about the news I just received."
→ Predicted Emotion: worry (Confidence: 48.65%)

Input Text: "This is the worst day, everything is going wrong and I feel awful."
→ Predicted Emotion: sadness (Confidence: 34.58%)

Input Text: "I just won the lottery! I can't believe it! This is amazing!"
→ Predicted Emotion: surprise (Confidence: 22.57%)

Input Text: "My presentation is tomorrow morning and I am so nervous."
→ Predicted Emotion: worry (Confidence: 55.38%)
```

CONCLUSION:

We successfully implemented TF-IDF and Logistic Regression to build a practical and high-performing tool that effectively detects emotions from text for real-time sentiment analysis.

REFERENCE:

- *https://medium.com/%40manwill/sentiment-analysis-with-tf-idf-and-logistic-regression-f4cd86f359a1*
- *https://www.tutorialspoint.com/tf-idf-in-sentiment-analysis*
- *https://www.researchgate.net/publication/353514662_Sentimental_Analysis_using_Logistic_Regression*