Name: Rohit Jagtap
PRN: 202201040048

Lab Assignment 4

NLP Preprocessing And Text Classification

1. Understand and apply NLP preprocessing techniques such as tokenization, stopword removal, stemming, and lemmatization.

2. Implement text vectorization techniques such as TF-IDF and CountVectorizer.

3. Develop a text classification model using a machine learning algorithm.

4. Evaluate the performance of the model using suitable metrics.

Colab File:- Copy of NLP Lab_Assignmnent_4.ipynb

Dataset: Emotion Dataset for Emotion Recognition

Train Dataset:-
https://drive.google.com/file/d/1GctUwtQdJrWIqkqpSVnQtheSezzL3mNf/view?usp=sharing

Test Dataset:-
https://drive.google.com/file/d/1MQm8t97E3nzDuwhIqF5vYHg_b-s_WIL8/view?usp=sharing

Validation Dataset:-
https://drive.google.com/file/d/1wdsCfVtx5PrKWGTzy6xcRcKeHiIPgeXd/view?usp=sharing

Google Colab Lab Assignment -NLP

Course Name: Deep Learning (MDM)_Labs DSG-SSB

Lab Title: NLP Techniques for Text Classification

Student Name: Rohit Jagtap

Student ID:202201040048

Date of Submission: 01/04/2025

Group Members: Parth kulkarni

Rohan Wagh

Rohan Agarwal

Objective The objective of this assignment is to implement NLP preprocessing techniques and build a text classification model using machine learning techniques.

Learning Outcomes:

1. Understand and apply NLP preprocessing techniques such as tokenization, stopword removal, stemming, and lemmatization.
2. Implement text vectorization techniques such as TF-IDF and CountVectorizer.
3. Develop a text classification model using a machine learning algorithm.
4. Evaluate the performance of the model using suitable metrics.

Assignment Instructions:

Part 1: NLP Preprocessing

Dataset Selection:

Choose any text dataset from **Best Datasets for Text** https://en.innovatiana.com/post/best-datasets-for-text-classification Classification, such as SMS Spam Collection, IMDb Reviews, or any other relevant dataset.

Download the dataset and upload it to Google Colab.

Load the dataset into a Pandas DataFrame and explore its structure (e.g., check missing values, data types, and label distribution).

Text Preprocessing:

Convert text to lowercase.

Perform tokenization using NLTK or spaCy.

Remove stopwords using NLTK or spaCy.

Apply stemming using PorterStemmer or SnowballStemmer.

Apply lemmatization using WordNetLemmatizer.

Vectorization Techniques:

Convert text data into numerical format using TF-IDF and CountVectorizer.

```
import pandas as pd

df_train = pd.read_csv('/content/drive/MyDrive/training.csv')
df_test = pd.read_csv('/content/drive/MyDrive/test.csv')
df_valid = pd.read_csv('/content/drive/MyDrive/validation.csv')

print("Training Set:")
print(df_train.info(), "\n")
print(df_train.head(), "\n")

print("Testing Set:")
print(df_test.info(), "\n")
print(df_test.head(), "\n")

print("Validation Set:")
print(df_valid.info(), "\n")
print(df_valid.head(), "\n")

print("Training Set Labels:\n", df_train['label'].value_counts(), "\n")
print("Testing Set Labels:\n", df_test['label'].value_counts(), "\n")
print("Validation Set Labels:\n", df_valid['label'].value_counts(), "\n")
```

```
None

                                                text  label
0  im feeling rather rotten so im not very ambiti...      0
1          im updating my blog because i feel shitty      0
2  i never make her separate from me because i do...      0
3  i left with my bouquet of red and yellow tulip...      1
4    i was feeling a little vain when i did this one      0

Validation Set:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    2000 non-null   object
 1   label   2000 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 31.4+ KB
None

                                                text  label
0  im feeling quite sad and sorry for myself but ...      0
1  i feel like i am still looking at a blank canv...      0
2                     i feel like a faithful servant      2
3                  i am just feeling cranky and blue      3
4  i can have for a treat or if i am feeling festive      1

Training Set Labels:
 label
1    5362
0    4666
3    2159
4    1937
2    1304
5     572
Name: count, dtype: int64 

Testing Set Labels:
 label
1    695
0    581
3    275
4    224
2    159
5     66
Name: count, dtype: int64 

Validation Set Labels:
 label
1    704
0    550
3    275
4    212
2    178
5     81
Name: count, dtype: int64
```

Text Processing

```
import shutil
import os
import nltk
nltk_data_path = os.path.expanduser('~/nltk_data')
shutil.rmtree(nltk_data_path, ignore_errors=True)
```

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

Text Processing

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
import spacy

stop_words = set(stopwords.words('english'))
ps = PorterStemmer()
lemmatizer = WordNetLemmatizer()
nlp = spacy.load('en_core_web_sm')

# Function for text preprocessing
def preprocess(text):
    text = text.lower()  # Lowercase
    tokens = word_tokenize(text)  # Tokenization
    tokens = [word for word in tokens if word.isalnum()]  # Remove punctuation
    tokens = [word for word in tokens if word not in stop_words]  # Stopword removal
    tokens = [ps.stem(word) for word in tokens]  # Stemming
    tokens = [lemmatizer.lemmatize(word) for word in tokens]  # Lemmatization
    return " ".join(tokens)  # Convert back to text

# Apply preprocessing to all datasets
df_train['processed_text'] = df_train['text'].apply(preprocess)
df_test['processed_text'] = df_test['text'].apply(preprocess)
df_valid['processed_text'] = df_valid['text'].apply(preprocess)
```

Vectorization

```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer

# Use TF-IDF
tfidf = TfidfVectorizer()
X_train_tfidf = tfidf.fit_transform(df_train['processed_text'])
X_test_tfidf = tfidf.transform(df_test['processed_text'])
X_valid_tfidf = tfidf.transform(df_valid['processed_text'])

# Use CountVectorizer
cv = CountVectorizer()
X_train_cv = cv.fit_transform(df_train['processed_text'])
X_test_cv = cv.transform(df_test['processed_text'])
X_valid_cv = cv.transform(df_valid['processed_text'])

# Labels (assuming the label column is named 'label')
y_train = df_train['label']
y_test = df_test['label']
y_valid = df_valid['label']
```

Splitting the Data:

Divide the dataset into training and testing sets (e.g., 80% training, 20% testing).

Building the Classification Model:

Train a text classification model using Logistic Regression, Naïve Bayes, or any other suitable algorithm.

Implement the model using scikit-learn.

Model Evaluation:

Evaluate the model using accuracy, precision, recall, and F1-score.

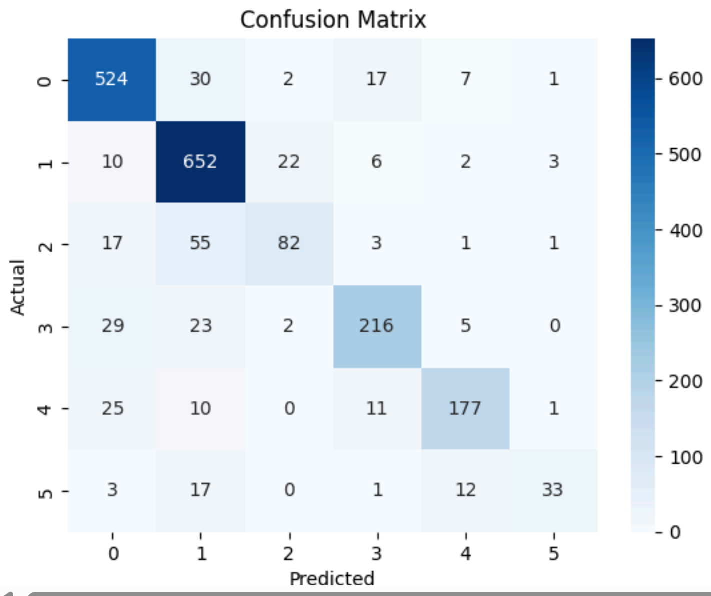
Use a confusion matrix to visualize the results.

```
from sklearn.linear_model import LogisticRegression

# Train classifier
```

```
clf = LogisticRegression()
clf.fit(X_train_tfidf, y_train)

# Predictions
y_pred = clf.predict(X_test_tfidf)
```

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))

# Classification report
print("Classification Report:\n", classification_report(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.842
Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.90      0.88       581
           1       0.83      0.94      0.88       695
           2       0.76      0.52      0.61       159
           3       0.85      0.79      0.82       275
           4       0.87      0.79      0.83       224
           5       0.85      0.50      0.63        66

    accuracy                           0.84      2000
   macro avg       0.84      0.74      0.77      2000
weighted avg       0.84      0.84      0.84      2000
```



	0	1	2	3	4	5
0	524	30	2	17	7	1
1	10	652	22	6	2	3
2	17	55	82	3	1	1
3	29	23	2	216	5	0
4	25	10	0	11	177	1
5	3	17	0	1	12	33

Submission Guidelines:

Google Colab Notebook Submission:

Save your notebook as NLP_Text_Classification_YourName.ipynb.

Ensure all code cells are executed, and the output is visible.

Include proper documentation and comments explaining each step.

Report Submission (Optional):

Prepare a short report (2-3 pages) summarizing your approach, findings, and model performance.

Upload the report along with the Colab Notebook.

Grading Criteria:

Correct implementation of NLP preprocessing (30%)

Effective use of vectorization techniques (20%)

Model accuracy and performance evaluation (30%)

Code clarity, documentation, and presentation (20%)

```
Start coding or generate with AI.
```

Declaration

I, Rohit Jagtap, confirm that the work submitted in this assignment is my own and has been completed following academic integrity guidelines. The code is uploaded on my GitHub repository account, and the repository link is provided below:

GitHub Repository Link: https://github.com/RohitJagtap123/NLP_Text_Classification.git

Signature: Rohit Jagtap

Submission Checklist

✓ Ultralitycs Platform Documentsation Like hel file for Given Task

✓ Code file (Python Notebook or Script)

✓ Dataset or link to the dataset

✓ Visualizations (if applicable)

✓ Screenshots of model performance metrics

✓ Readme File

✓ Evaluation Metrics Details and discussion