AY: 2025-26

Class:	BE	**Semester:**	VII
Course Code:	CSDOL7011	**Course Name:**	Natural Language Processing

Name of Student:	Konisha Jayesh Thakare
Roll No. :	71
Experiment No.:	9
Title of the Experiment:	Training and Evaluating a Text Classification Model Using Proper Experimental Methodology
Date of Performance:	09.09.2025
Date of Submission:	16.09.2025

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations(BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Dr. Tatwadarshi P. Nagarhalli

Signature :

Date :

Aim: To train and evaluate a text classification model using proper experimental methodology, including dataset preparation, preprocessing, feature extraction, model training, and performance evaluation.

Theory:

Text Classification is the process of assigning predefined categories to text documents. Applications include:

- Spam detection
- Sentiment analysis
- Topic classification
- Email filtering

A proper experimental methodology ensures reliable and reproducible results:

1. Data Collection: Select or download a labeled dataset.
2. Data Preprocessing: Clean text by lowercasing, removing punctuation, stopwords, and applying tokenization or stemming/lemmatization.
3. Feature Engineering: Convert text into numerical vectors using Bag-of-Words, TF-IDF, or embeddings.
4. Model Selection: Choose an appropriate classifier (e.g., Logistic Regression, Naïve Bayes, SVM).
5. Training and Testing: Split the dataset into training and testing sets to avoid overfitting.
6. Evaluation: Assess model performance using metrics like accuracy, precision, recall, F1-score, and confusion matrix.

This methodology ensures that the results are valid, comparable, and reproducible.

Procedure:

1. Collect or load a labeled text dataset (e.g., IMDB reviews, SMS spam dataset).
2. Preprocess the text data: lowercase, remove punctuation, stopwords, tokenize.
3. Convert the text to numerical features using TF-IDF or Bag-of-Words.
4. Split the dataset into training and testing sets (e.g., 80:20).
5. Train a machine learning model (e.g., Logistic Regression, Naïve Bayes).
6. Evaluate model performance on the test set using accuracy, precision, recall, F1-score, and confusion matrix.
7. Analyze and interpret the results.

Algorithm:

1. Load the dataset.
2. Preprocess the text.
3. Apply feature extraction (TF-IDF).
4. Split data into training and test sets.
5. Train a classifier on training data.
6. Predict labels on test data.
7. Compute evaluation metrics.
8. Interpret results and conclude.

Implementation:

```
import nltk
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import requests
from io import StringIO

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

url =
"https://raw.githubusercontent.com/justmarkham/pycon-2016-tutorial/master/
data/sms.tsv"
response = requests.get(url)
data = pd.read_csv(StringIO(response.text), sep='\t', header=None,
names=['label','text'])

data['label'] = data['label'].map({'ham':0, 'spam':1})
stop_words = set(stopwords.words('english'))
ps = PorterStemmer()

def preprocess(text):
    text = text.lower()
```

```
    text = text.translate(str.maketrans('', '', string.punctuation))
    tokens = nltk.word_tokenize(text)
    tokens = [ps.stem(word) for word in tokens if word not in stop_words]
    return " ".join(tokens)

data['clean_text'] = data['text'].apply(preprocess)
tfidf = TfidfVectorizer()
X = tfidf.fit_transform(data['clean_text'])
y = data['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
model = MultinomialNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

Output:

```
Accuracy: 0.9659192825112107

Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       966
           1       0.99      0.75      0.85       149

    accuracy                           0.97      1115
   macro avg       0.98      0.88      0.92      1115
weighted avg       0.97      0.97      0.96      1115


Confusion Matrix:
 [[965   1]
 [ 37 112]]
PS C:\Users\Konisha Thakare\OneDrive\Desktop\Python>
```

Interpretation:

- High accuracy indicates the model is effective at distinguishing spam and ham messages.
- Precision, recall, and F1-score confirm the model's reliability.

Conclusion:

This experiment demonstrated the complete methodology for training and evaluating a text classification model. By preprocessing text, converting it into numerical features using TF-IDF, and applying a Naïve Bayes classifier, the model was able to accurately classify messages into categories. Proper experimental methodology, including dataset splitting, preprocessing, and evaluation using multiple metrics, ensures the results are valid, reproducible, and interpretable. Such an approach forms the foundation for a wide range of NLP applications, including spam detection, sentiment analysis, and topic classification.