**Experiment 10: Sentiment analysis on imdb data set supervised**

**#Objective:**

The goal of this experiment is to perform sentiment analysis on the IMDB dataset using supervised machine learning techniques, specifically applying logistic regression. The dataset consists of movie reviews, and the task is to predict the sentiment of each review (positive or negative).

**#Code**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report, accuracy\_score

import re

from nltk.corpus import stopwords

# Load data

data = pd.read\_csv('IMDB Dataset.csv')

# Preprocess the text data (remove HTML tags, punctuation, etc.)

def preprocess\_text(text):

# Remove HTML tags

text = re.sub(r'<.\*?>', '', text)

# Remove non-alphanumeric characters and convert to lowercase

text = re.sub(r'[^a-zA-Z\s]', '', text)

# Convert to lowercase

text = text.lower()

# Remove stop words

stop\_words = set(stopwords.words('english'))

text = ' '.join([word for word in text.split() if word not in stop\_words])

return text

# Apply text preprocessing to the review column

data['cleaned\_review'] = data['review'].apply(preprocess\_text)

# Split data into features and labels

X = data['cleaned\_review']

y = data['sentiment']

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Convert the text data into numerical format using TF-IDF

vectorizer = TfidfVectorizer(max\_features=5000)

X\_train\_tfidf = vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = vectorizer.transform(X\_test)

# Train a Logistic Regression model

model = LogisticRegression()

model.fit(X\_train\_tfidf, y\_train)

# Make predictions

y\_pred = model.predict(X\_test\_tfidf)

# Evaluate the model

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

***#Output:***

***Accuracy: 0.8882666666666666***

***Classification Report:***

***precision recall f1-score support***

***negative 0.89 0.88 0.89 7411***

***positive 0.88 0.90 0.89 7589***

***accuracy 0.89 15000***

***macro avg 0.89 0.89 0.89 15000***

***weighted avg 0.89 0.89 0.89 15000***

**#Conclusion:**

The Minimum Edit Distance algorithm is a fundamental technique in computer science with applications in various fields. By understanding the recursive approach, we can efficiently calculate the minimum number of operations required to transform one sequence into another.