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.