TECHNOHACKS

(LETS GROW TOGETHER)

An PROJECT

on

“MOVIE REVIEWS CLASSIFICATION”

Submitted in partial fulfillment for the INTERNSHIP

BATCH 23

IN

MACHINE LEARNING DOMAIN

Submitted by

AFSHA SULTHANA

Under the guidance of

Mr.SANDIP GAVIT

TECHNOHACKS EDUTECH INTERNSHIP PROGRAM

2023

MOVIE REVIEWS CLASSIFICATION:

use a dataset containing movie reviews to build a model that can classify them as positive or negative.

Certainly! To build a model that can classify movie reviews as positive or negative, you can use a dataset containing labeled movie reviews. In this example, I'll use the IMDb dataset available in the **nltk** library, which is a collection of 50,000 movie reviews labeled as positive or negative.

Code:

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Download the IMDb dataset from nltk
import nltk
from nltk.corpus import movie_reviews
nltk.download('movie_reviews')

# Create a list of documents where each document is a movie review
documents = [(list(movie_reviews.words(fileid)), category)
for category in movie_reviews.categories()
for fileid in movie_reviews.fileids(category)]

# Shuffle the documents
np.random.shuffle(documents)

# Separate the documents into features (X) and target variable (y)
```

```
X = [" ".join(words) for words, category in documents]
y = [category for words, category in documents]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Convert the text data into numerical features using CountVectorizer
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

# Build and train the Naive Bayes classifier
naive_bayes_classifier = MultinomialNB()
naive_bayes_classifier.fit(X_train_vectorized, y_train)

# Make predictions on the test set
y_pred = naive_bayes_classifier.predict(X_test_vectorized)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{class_report}')
```

OUTPUT:

```
[nltk_data] Downloading package movie_reviews to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package movie_reviews is already up-to-date!
Accuracy: 0.8125
Confusion Matrix:
[[180  40]
 [ 35 145]]
Classification Report:
              precision    recall  f1-score   support

         neg       0.84      0.82      0.83       220
         pos       0.78      0.81      0.79       180

    accuracy                           0.81       400
   macro avg       0.81      0.81      0.81       400
weighted avg       0.81      0.81      0.81       400
```

This example uses a simple bag-of-words representation of the text data and a Naive Bayes classifier. You can experiment with other feature extraction techniques and classification algorithms based on your preferences and the characteristics of your dataset.

This code downloads the movie_reviews dataset from nltk and uses it for building and testing the sentiment classification model.