Name-Bharti Kohli
SAP ID-500096937
Enroll no-R2142211290
Course- Blockchain Technology

Project Report

Movie Review Classification using Naive Bayes

Introduction:

This project aims to develop a movie review classification system utilizing the Naive Bayes classifier. The primary objective is to classify movie reviews as positive or negative based on their sentiment. The Naive Bayes algorithm is chosen due to its simplicity and efficiency in text classification tasks.

Dataset:

The dataset comprises movie reviews sourced from Kaggle(IMDB Dataset), consisting of 2 features the review and sentiment labeled as positive or negative to denote the sentiment. Preprocessing steps, such as removing irrelevant information like HTML tags, punctuation, and stopwords, were applied to enhance data quality.

Methodology:

- Preprocessing: Initial preprocessing involved tokenization, removal of stopwords, and stemming to extract meaningful features.
- Feature Extraction: We adopted a bag-of-words model to convert reviews into numerical feature vectors.
- Naive Bayes Classifier: Trained on the preprocessed dataset, the Naive Bayes classifier calculates conditional probabilities of each class (positive or negative) given the input features.
- Model Evaluation: Performance metrics including accuracy, precision, recall, and F1-score were used to assess the classifier's effectiveness.

Implementation:

Programming Language:

- Python

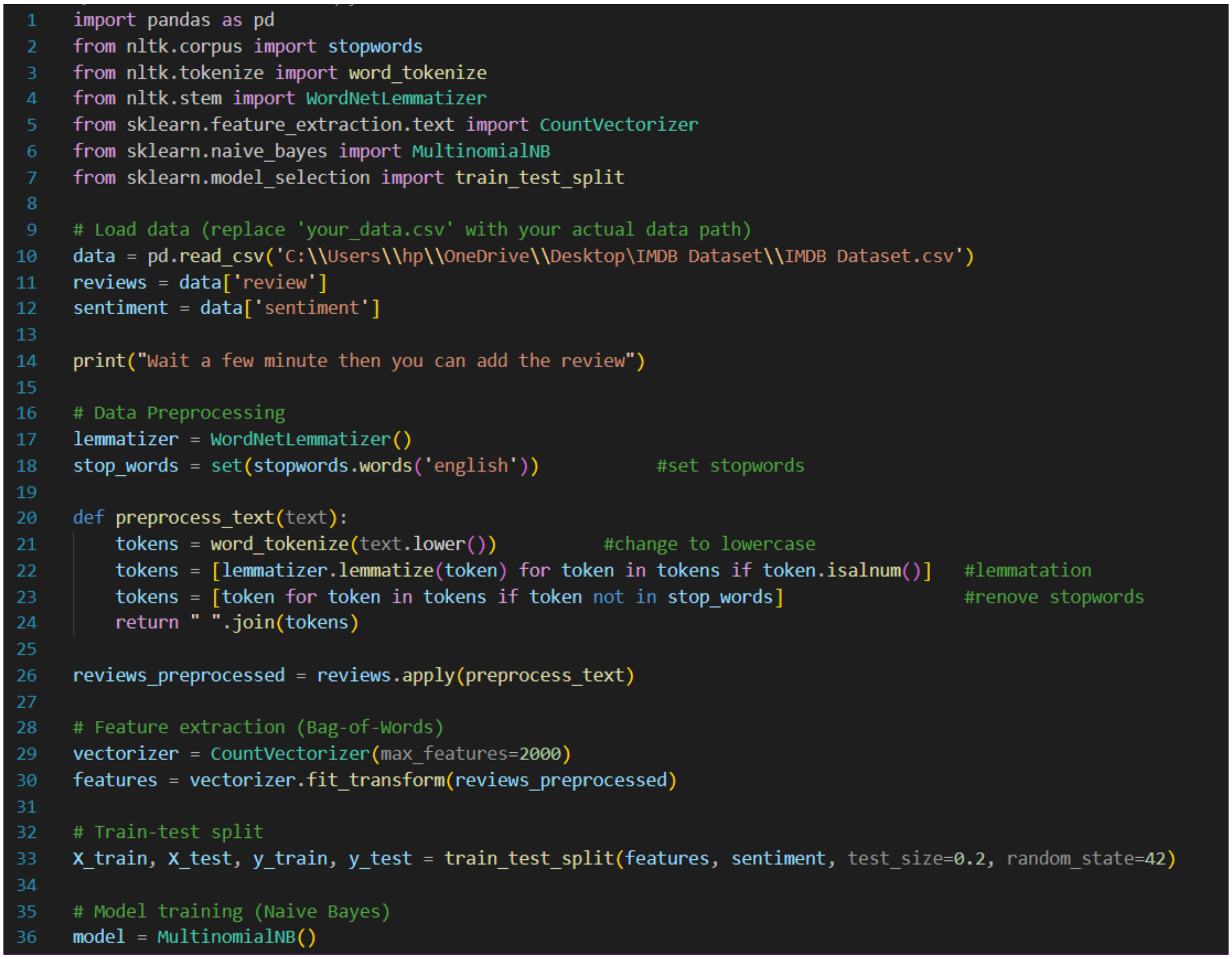
Libraries Used:

- scikit-learn: for Naive Bayes classifier implementation and evaluation metrics.
- NLTK (Natural Language Toolkit): for text preprocessing.

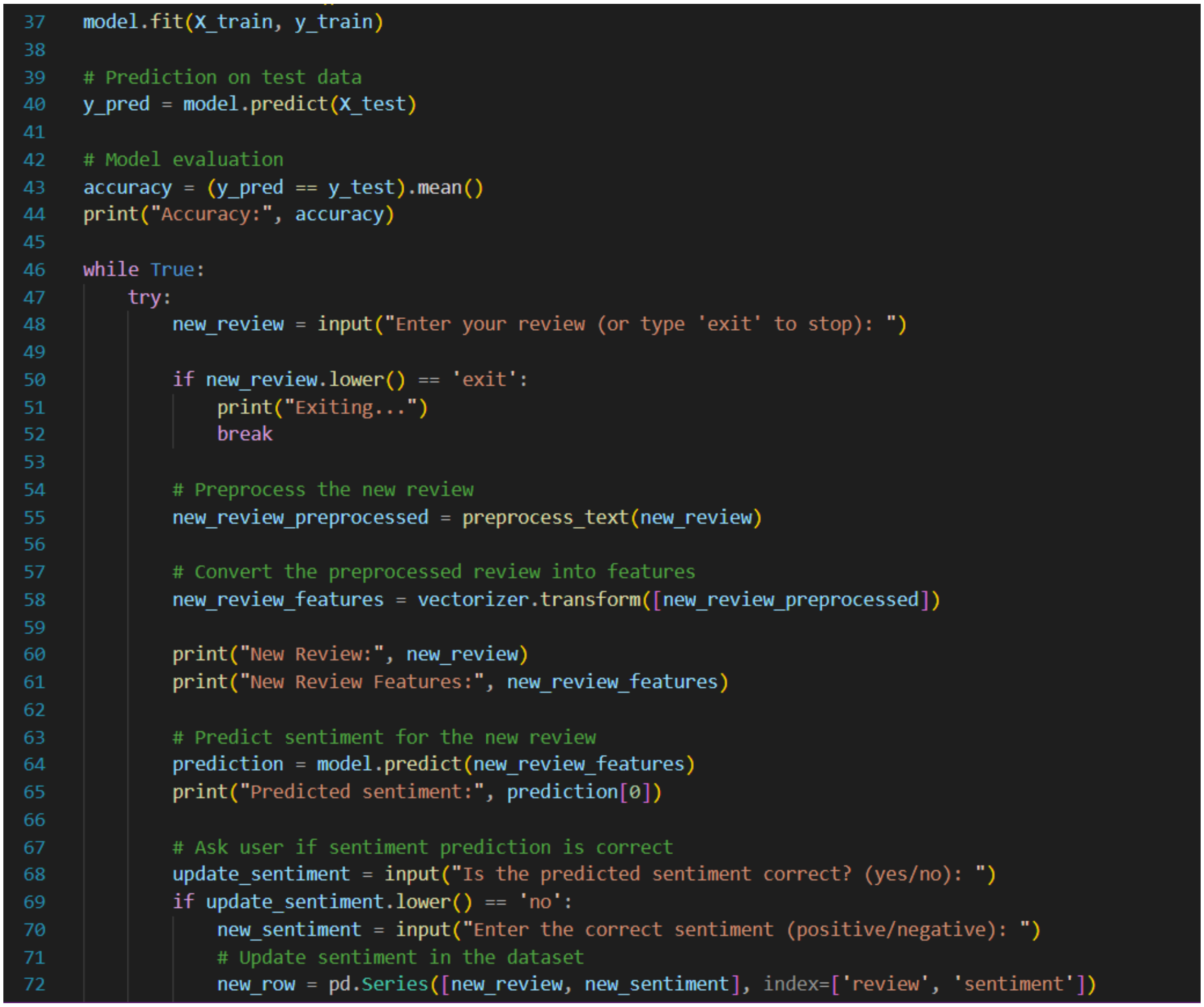
Steps:

- Data loading and preprocessing
- Feature extraction (bag-of-words)
- Dataset split into training and testing sets
- Training Naive Bayes classifier
- Evaluation on test set

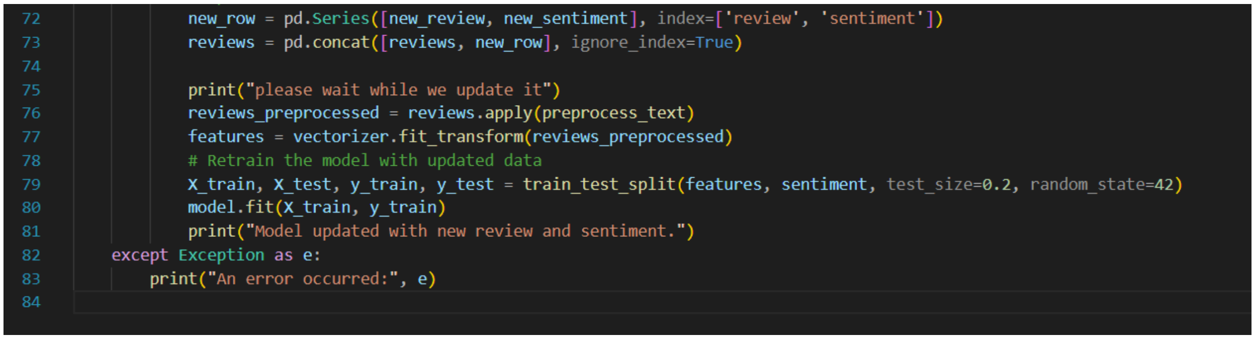
Code:



```python
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split

# Load data (replace 'your_data.csv' with your actual data path)
data = pd.read_csv('C:\\Users\\hp\\OneDrive\\Desktop\IMDB Dataset\\IMDB Dataset.csv')
reviews = data['review']
sentiment = data['sentiment']

print("Wait a few minute then you can add the review")

# Data Preprocessing
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))           #set stopwords

def preprocess_text(text):
    tokens = word_tokenize(text.lower())          #change to lowercase
    tokens = [lemmatizer.lemmatize(token) for token in tokens if token.isalnum()]   #lemmatation
    tokens = [token for token in tokens if token not in stop_words]                #renove stopwords
    return " ".join(tokens)

reviews_preprocessed = reviews.apply(preprocess_text)

# Feature extraction (Bag-of-Words)
vectorizer = CountVectorizer(max_features=2000)
features = vectorizer.fit_transform(reviews_preprocessed)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(features, sentiment, test_size=0.2, random_state=42)

# Model training (Naive Bayes)
model = MultinomialNB()
```

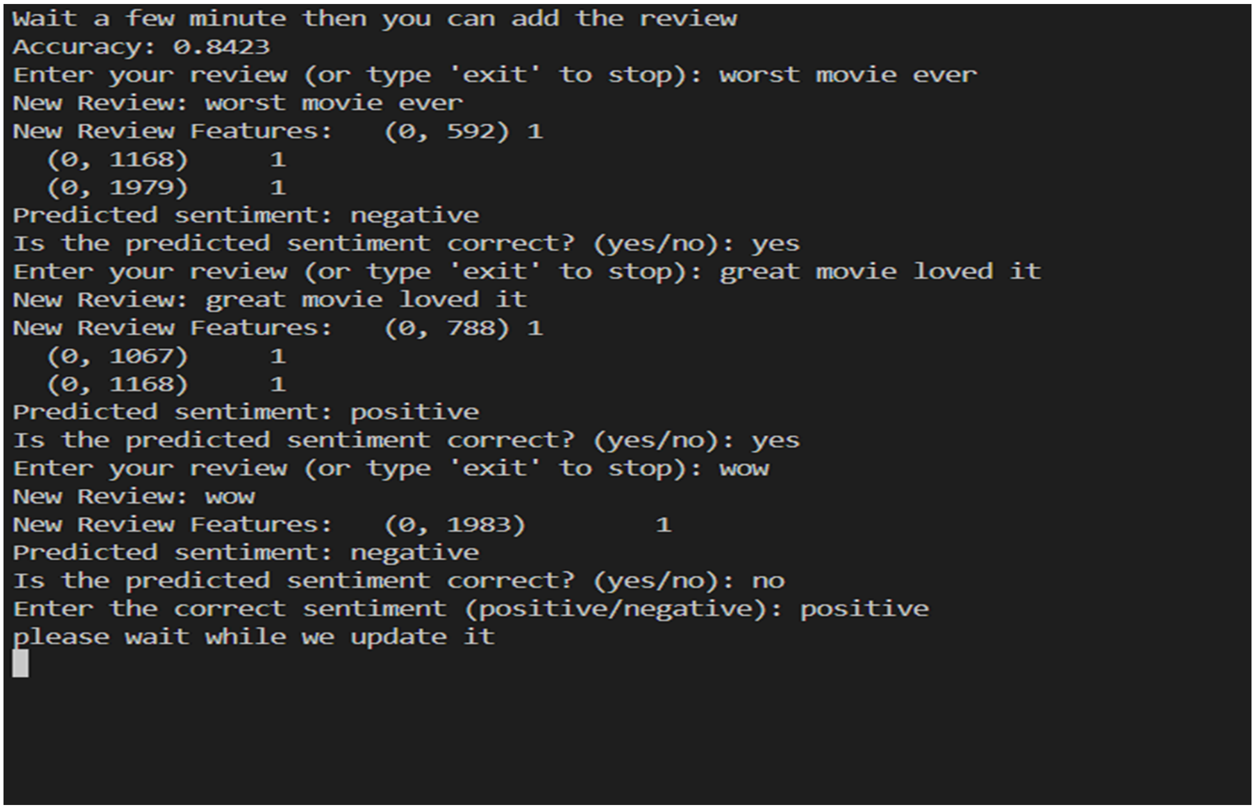


```python
model.fit(X_train, y_train)

# Prediction on test data
y_pred = model.predict(X_test)

# Model evaluation
accuracy = (y_pred == y_test).mean()
print("Accuracy:", accuracy)

while True:
    try:
        new_review = input("Enter your review (or type 'exit' to stop): ")

        if new_review.lower() == 'exit':
            print("Exiting...")
            break

        # Preprocess the new review
        new_review_preprocessed = preprocess_text(new_review)

        # Convert the preprocessed review into features
        new_review_features = vectorizer.transform([new_review_preprocessed])

        print("New Review:", new_review)
        print("New Review Features:", new_review_features)

        # Predict sentiment for the new review
        prediction = model.predict(new_review_features)
        print("Predicted sentiment:", prediction[0])

        # Ask user if sentiment prediction is correct
        update_sentiment = input("Is the predicted sentiment correct? (yes/no): ")
        if update_sentiment.lower() == 'no':
            new_sentiment = input("Enter the correct sentiment (positive/negative): ")
            # Update sentiment in the dataset
            new_row = pd.Series([new_review, new_sentiment], index=['review', 'sentiment'])
```



```python
            new_row = pd.Series([new_review, new_sentiment], index=['review', 'sentiment'])
            reviews = pd.concat([reviews, new_row], ignore_index=True)

            print("please wait while we update it")
            reviews_preprocessed = reviews.apply(preprocess_text)
            features = vectorizer.fit_transform(reviews_preprocessed)
            # Retrain the model with updated data
            X_train, X_test, y_train, y_test = train_test_split(features, sentiment, test_size=0.2, random_state=42)
            model.fit(X_train, y_train)
            print("Model updated with new review and sentiment.")
    except Exception as e:
        print("An error occurred:", e)

```

Results:



```
Wait a few minute then you can add the review
Accuracy: 0.8423
Enter your review (or type 'exit' to stop): worst movie ever
New Review: worst movie ever
New Review Features:   (0, 592) 1
  (0, 1168)     1
  (0, 1979)     1
Predicted sentiment: negative
Is the predicted sentiment correct? (yes/no): yes
Enter your review (or type 'exit' to stop): great movie loved it
New Review: great movie loved it
New Review Features:   (0, 788) 1
  (0, 1067)     1
  (0, 1168)     1
Predicted sentiment: positive
Is the predicted sentiment correct? (yes/no): yes
Enter your review (or type 'exit' to stop): wow
New Review: wow
New Review Features:   (0, 1983)        1
Predicted sentiment: negative
Is the predicted sentiment correct? (yes/no): no
Enter the correct sentiment (positive/negative): positive
please wait while we update it
```