Marmara University
Faculty of Engineering

CSE 4288

Introduction to Machine Learning

Progress Report and Code

Instructor: Assoc. Prof. Murat Can Ganiz Due Date: 15.12.2024

	Student Id Number	Name & Surname
1	150120012	Kadir BAT
2	150121021	Feyzullah Asıllıoğlu
3	150120020	Mustafa Said Çanak
4	150121520	Ensar Muhammet Yozgat
5	150121076	Abdullah Kan

Table of Contents

1. Introduction

The goal of this project is to perform sentiment analysis on IMDB movie reviews using two machine learning models: Naive Bayes and Logistic Regression. The analysis focuses on classifying reviews as positive or negative based on their text. This report outlines the code structure, preprocessing steps, model training, evaluation metrics, and results.

2. Dataset Overview

- **Dataset**: IMDB movie reviews dataset
- **Features**:
    - review: The text of the movie review.
    - sentiment: The label indicating whether the review is "positive" or "negative".
- **Size**: 50,000 reviews (before preprocessing)
- **Data Format**: CSV

3. Code Structure

The code is encapsulated in a Python class MovieReviewSentimentAnalysis. Below are the key components:

3.1 Initialization

- **Purpose**: Load the dataset from the given path.
- **Code**:

```
def __init__(self, dataset_path):
  self.dataset = pd.read_csv(dataset_path)
```

3.2 Training

- **Purpose**: Preprocess data and train both Naive Bayes and Logistic Regression models.
- **Code**:

```
def train(self, x_train, y_train):
   self.__preprocess()
   self.__fit_naive_bayes(x_train, y_train)
   self.__fit_logistic_regression(x_train, y_train)
```

3.3 Model Fitting

- **Naive Bayes**:
    - **Classifier**: MultinomialNB
    - **Code**:

```
def __fit_naive_bayes(self, x_train, y_train):
    self.nb_classifier = MultinomialNB()
    self.nb_classifier.fit(x_train, y_train)
```

- **Logistic Regression**:
    - **Classifier**: LogisticRegression
    - **Code**:

```
def __fit_logistic_regression(self, x_train, y_train):
    self.lr_classifier = LogisticRegression(max_iter=1000)
    self.lr_classifier.fit(x_train, y_train)
```

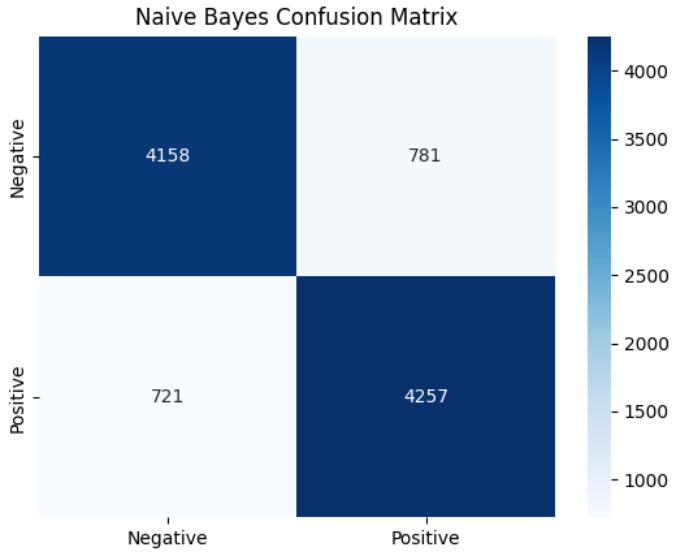
4. Observations and Results

4.1 Naive Bayes

```
Naive Bayes
Accuracy:  0.8485429061208026
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.84      0.85      4939
           1       0.84      0.86      0.85      4978

    accuracy                           0.85      9917
   macro avg       0.85      0.85      0.85      9917
weighted avg       0.85      0.85      0.85      9917
```

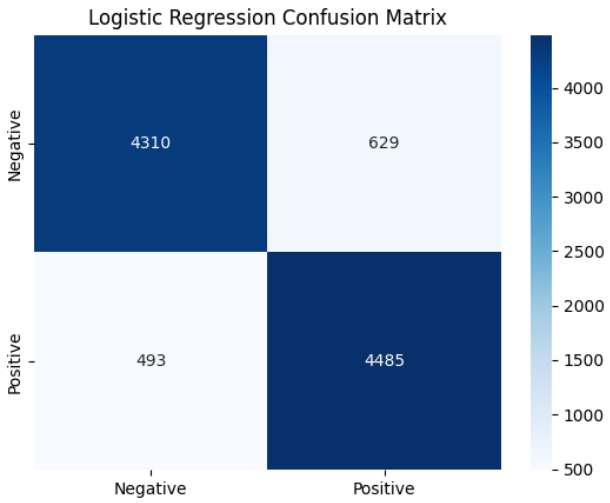


4.2 Logistic Regression

```
Logistic Regression
Accuracy:  0.8868609458505596
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.87      0.88      4939
           1       0.88      0.90      0.89      4978

    accuracy                           0.89      9917
   macro avg       0.89      0.89      0.89      9917
weighted avg       0.89      0.89      0.89      9917
```



5. Improvements and Future Work

- Optimize hyperparameters for better model performance.
- Experiment with other classifiers like SVM or deep learning models.
- Use additional text preprocessing techniques like stemming or lemmatization.
- Address class imbalance if observed in the dataset.

6. Conclusion

This project demonstrates the use of supervised machine learning models (Naive Bayes and Logistic Regression) for sentiment analysis on IMDB reviews. While the implementation covers the essential steps, further enhancements could lead to improved results and robustness.

7. Appendix: Complete Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
```

```
class MovieReviewSentimentAnalysis:
    def __init__(self, dataset_path):
        self.dataset = pd.read_csv(dataset_path)

    def train(self, x_train, y_train):
        self.__preprocess()
        self.__fit_naive_bayes(x_train, y_train)
        self.__fit_logistic_regression(x_train, y_train)

    def __fit_naive_bayes(self, x_train, y_train):
        self.nb_classifier = MultinomialNB()
        self.nb_classifier.fit(x_train, y_train)

    def __fit_logistic_regression(self, x_train, y_train):
        self.lr_classifier = LogisticRegression(max_iter=1000)
        self.lr_classifier.fit(x_train, y_train)

    def __predicit_naive_bayes(self, x_test, y_test):
        y_pred = self.nb_classifier.predict(x_test)
        print("Naive Bayes")
        print("Accuracy: ", accuracy_score(y_test, y_pred))
        print("Classification Report: ", classification_report(y_test, y_pred))
        cm = confusion_matrix(y_test, y_pred)
        sns.heatmap(cm, annot=True)
        plt.show()

    def __predict_logistic_regression(self, x_test, y_test):
        y_pred = self.lr_classifier.predict(x_test)
        print("Logistic Regression")
        print("Accuracy: ", accuracy_score(y_test, y_pred))
        print("Classification Report: ", classification_report(y_test, y_pred))
        cm = confusion_matrix(y_test, y_pred)
```

```
        sns.heatmap(cm, annot=True)
        plt.show()


    def evaluate(self, x_test, y_test):
        self.__predicit_naive_bayes(x_test, y_test)
        self.__predict_logistic_regression(x_test, y_test)
```