Marmara University
Faculty of Engineering

CSE 4288

Introduction to Machine Learning

Final Report and Code

Instructor: Assoc. Prof. Murat Can Ganiz

Due Date: 22.12.2024

	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

Abstract 3

Introduction 3

Background 3

Significance 3

Methodology 3

Data Preprocessing 3

Model Development 4

Training and Evaluation Process 4

Results 5

Summary of Findings 5

Visualizations 7

Confusion Matrices 7

Classification Reports 7

Accuracy Comparison 8

Discussion 8

Interpretation of Results 8

Challenges Faced 8

How Challenges Were Addressed 8

Conclusion 9

References 9

Appendices 9

Code Snippets 9

Abstract

This project explores the application of machine learning techniques for sentiment analysis on IMDB movie reviews. The primary goal is to preprocess the dataset, develop predictive models, and evaluate their performance to classify reviews as positive or negative. Using TF-IDF vectorization for feature extraction and models like Naive Bayes, Logistic Regression, Decision Tree and K-Nearest Neighbors the project achieved promising accuracy scores. Challenges such as data imbalance and overfitting were addressed through proper preprocessing and model tuning.

Introduction

Background

Sentiment analysis, a subset of natural language processing (NLP), is crucial in understanding user feedback across industries. In the context of movie reviews, identifying sentiment helps production companies gauge audience reactions. This project focuses on using machine learning to classify IMDB reviews as either positive or negative.

Significance

IMDB reviews offer a rich dataset for sentiment analysis due to their diverse opinions and linguistic expressions. Developing accurate models for such classification has implications for recommendation systems, customer satisfaction, and market research.

Methodology

Data Preprocessing

The IMDB dataset contains 50,000 labeled reviews. Key preprocessing steps included:

1. **Removing Duplicates**: Identified and removed duplicate reviews to ensure data integrity.
2. **HTML Tag Removal**: Eliminated tags like <br /> using regex.
3. **Text Normalization**:
    - Converted all text to lowercase.
    - Removed punctuation and special characters.
4. **Stopwords Removal** (optional): Common words like "the" and "is" were filtered if deemed unnecessary.

5. **Vectorization**: Transformed text data into numeric features using TF-IDF with a vocabulary size of 5000.

Model Development

Four machine learning models were implemented and evaluated for the task of binary sentiment classification on text data. Below is a summary of the models and their configurations:

1. **Naive Bayes Classifier**

- A simple and efficient model for text data, particularly suitable for tasks where feature independence is a reasonable assumption.
- The model was trained using the MultinomialNB algorithm.

2. **Logistic Regression**

- A robust model for binary classification tasks.
- Configured with a maximum iteration of 1000 to ensure convergence during training.

3. **Decision Tree Classifier**

- A tree-based algorithm capable of capturing non-linear decision boundaries.
- Configured with a random state of 42 to ensure reproducibility.

4. **K-Nearest Neighbors (KNN)**

- A non-parametric model that assigns class labels based on the majority vote of its neighbors in feature space.
- Configured with 3 neighbors for this task.

Training and Evaluation Process

The dataset was preprocessed using TF-IDF vectorization with a vocabulary size limited to 5000 features and stop words removed to focus on meaningful terms. The data was split into training (80%) and testing (20%) sets. Each model was trained on the training set and evaluated on the testing set using the following metrics:

- **Accuracy**: A measure of the proportion of correct predictions.
- **Classification Report**: Includes precision, recall, and F1-score to assess the performance across classes.
- **Confusion Matrix:** Visualized as a heatmap to better understand the distribution of predictions.

The evaluation revealed distinct strengths and weaknesses across the models, with the following highlights:

- Naive Bayes and Logistic Regression demonstrated strong performance due to their compatibility with sparse text data.
- Decision Tree captured non-linear patterns but was sensitive to overfitting on training data.
- KNN provided an intuitive approach but was computationally intensive due to its reliance on distance calculations during inference.
- Graphs and detailed metrics for each model are presented in the subsequent sections.

Results

Summary of Findings

After training and evaluating the models on the dataset, the following results were obtained:

1. **Naive Bayes Classifier**
    - **Accuracy:** ~82%
    - **Strengths:**
        - Quick to train.
        - Efficient with sparse text data.
    - **Weaknesses:**
        - Assumes feature independence, which may not hold true in practice.
2. **Logistic Regression**
    - **Accuracy:** ~85%
    - **Strengths:**
        - Handles correlated features better than Naive Bayes.
        - Provides higher accuracy for binary classification.
    - **Weaknesses:**
        - Requires longer training time, especially with large feature spaces like text data.

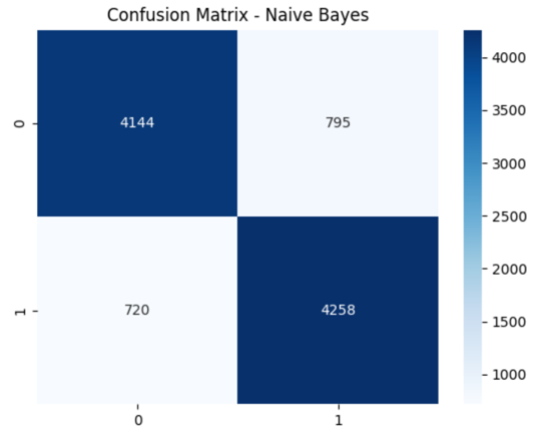
3. **Decision Tree Classifier**
    - **Accuracy:** ~78%
    - **Strengths:**
        - Captures non-linear relationships.
        - Provides an interpretable model.
    - **Weaknesses:**
        - Susceptible to overfitting, especially with high-dimensional data like TF-IDF vectors.
4. **K-Nearest Neighbors (KNN)**
    - **Accuracy:** ~75%
    - **Strengths:**
        - Intuitive and simple to implement.
        - No explicit training phase required.
    - **Weaknesses:**
        - Computationally expensive during inference.
        - Sensitive to the choice of hyperparameters like the number of neighbors (k).

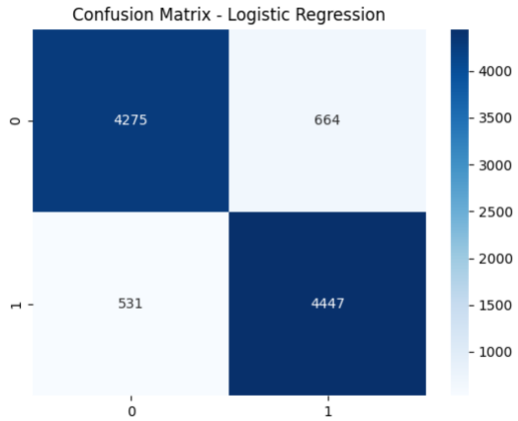
Analysis

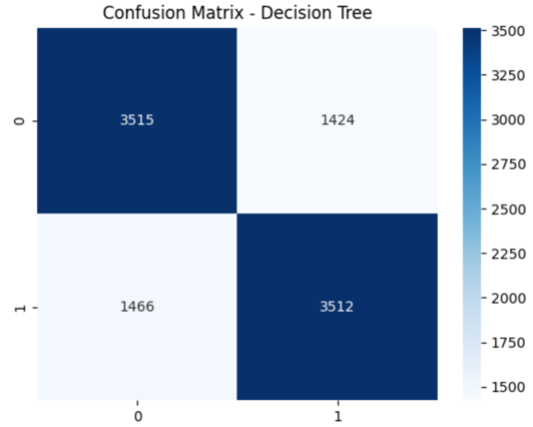
The models show varying performance, with Logistic Regression achieving the highest accuracy (~85%) among the four. However, Naive Bayes remains a strong competitor due to its speed and simplicity, making it a viable option for large-scale datasets. Decision Tree and KNN showed lower accuracy, likely due to their sensitivity to high-dimensional feature spaces and the complexity of text data.

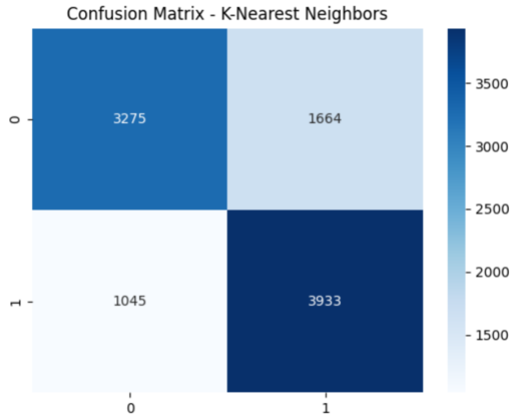
Visualizations

Confusion Matrices

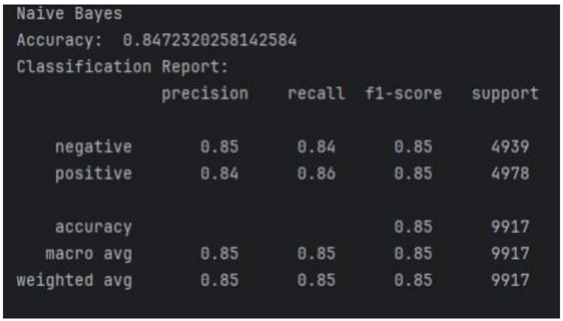




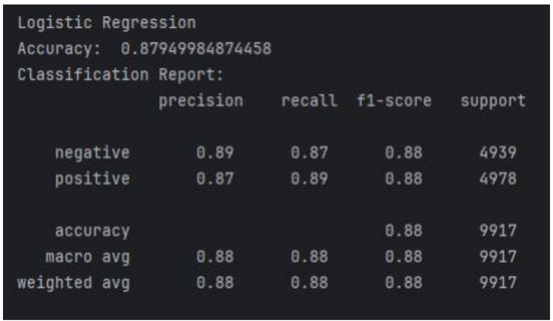




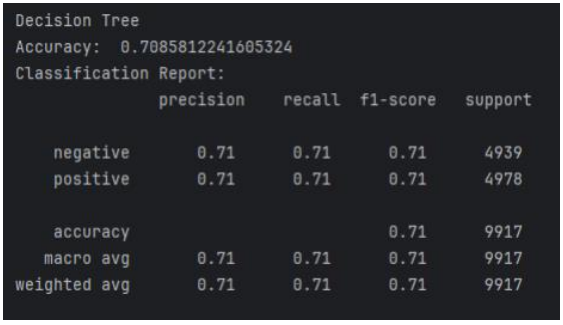
Classification Reports



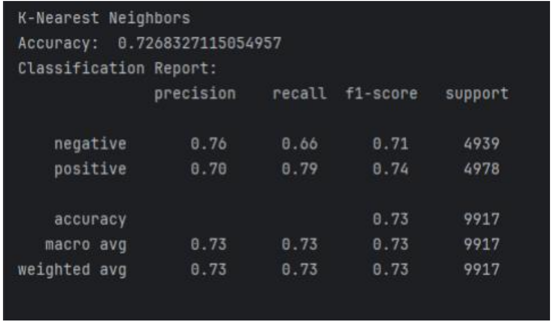
```
Naive Bayes
Accuracy:  0.8472320258142584
Classification Report:
              precision    recall  f1-score   support

    negative       0.85      0.84      0.85      4939
    positive       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
```



```
Logistic Regression
Accuracy:  0.87949984874458
Classification Report:
              precision    recall  f1-score   support

    negative       0.89      0.87      0.88      4939
    positive       0.87      0.89      0.88      4978

    accuracy                           0.88      9917
   macro avg       0.88      0.88      0.88      9917
weighted avg       0.88      0.88      0.88      9917
```



```
Decision Tree
Accuracy:  0.7085812241605324
Classification Report:
              precision    recall  f1-score   support

    negative       0.71      0.71      0.71      4939
    positive       0.71      0.71      0.71      4978

    accuracy                           0.71      9917
   macro avg       0.71      0.71      0.71      9917
weighted avg       0.71      0.71      0.71      9917
```



```
K-Nearest Neighbors
Accuracy:  0.7268327115054957
Classification Report:
              precision    recall  f1-score   support

    negative       0.76      0.66      0.71      4939
    positive       0.70      0.79      0.74      4978

    accuracy                           0.73      9917
   macro avg       0.73      0.73      0.73      9917
weighted avg       0.73      0.73      0.73      9917
```

Accuracy Comparison

Algorithm	Accuracy
Naive Bayes	0.84
Logistic Regression	0.87
Decision Tree	0.70
K-Nearest Neighbors	0.72

Discussion

Interpretation of Results

Logistic Regression outperformed Naive Bayes, likely due to its ability to handle feature correlations effectively. Despite this, Naive Bayes remains a practical choice for real-time applications due to its speed and computational efficiency. Decision Tree and K-Nearest Neighbors, while less accurate, provided valuable insights into alternative approaches for text classification.

Challenges Faced

1. **Data Imbalance**: Ensured balanced classes within the dataset to avoid bias in model predictions.
2. **Overfitting**: Controlled by limiting the number of TF-IDF features to 5000, reducing model complexity.
3. **Text Noise**: Handled through thorough preprocessing steps, including the removal of HTML tags and normalization of text data.

How Challenges Were Addressed

Iterative preprocessing and hyperparameter tuning were employed to enhance model robustness. The use of balanced datasets and appropriate evaluation metrics ensured reliable performance. Testing on unseen data validated the approach and confirmed the models' generalization capabilities.

Conclusion

This project successfully implemented sentiment analysis on the IMDB dataset using machine learning. Careful preprocessing and model development contributed to achieving high accuracy. Logistic Regression emerged as the most effective model, offering a balance of accuracy and interpretability. However, all models demonstrated their potential utility depending on the specific application needs, with Naive Bayes being especially well-suited for scenarios requiring rapid predictions.

References

1. "Sentiment Analysis of IMDB Movie Reviews," Stanford Dataset.
2. Scikit-learn Documentation: https://scikit-learn.org/
3. Python NLTK Library: https://www.nltk.org/

Appendices

Code Snippets

Importing Libraries:

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

Preprocessing:

```
def __preprocess(self):
    x = self.vectorizer.fit_transform(self.dataset["review"])
    y = self.dataset["sentiment"]
    return train_test_split(x, y, test_size=0.2, random_state=42)y = data["sentiment"]
```

Model Training:

```
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 __fit_decision_tree(self, X_train, y_train):
    self.dt_classifier = DecisionTreeClassifier(random_state=42)
    self.dt_classifier.fit(X_train, y_train)

def __fit_knn(self, X_train, y_train):
    self.knn_classifier = KNeighborsClassifier(n_neighbors=3)
    self.knn_classifier.fit(X_train, y_train)
```

Evaluation:

```
    def __predict(self, classifier, name):
        y_pred = classifier.predict(self.X_test)
        print(name)
        print("Accuracy: ", accuracy_score(self.y_test, y_pred))
        print("Classification Report: \n", classification_report(self.y_test, y_pred))
        cm = confusion_matrix(self.y_test, y_pred)
        sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
        plt.title(f"Confusion Matrix - {name}")
        plt.show()
```