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 2
Introduction 3
Background 3
Significance 3
Methodology 3
Data Preprocessing 3
Model Development 3
Evaluation Methods 4
Results 4
Summary of Findings 4
Visualizations 4
Discussion 6
Interpretation of Results 6
Challenges Faced 6
How Challenges Were Addressed 6
References 6
Appendices 7
Code Snippets 7

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 and Logistic Regression, 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
 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

Two machine learning models were implemented:

1. **Naive Bayes Classifier**:
    - Simple and efficient for text data.
    - Assumes feature independence.
2. **Logistic Regression**:
    - Suitable for binary classification.
    - Tuned with a maximum iteration of 1000 to ensure convergence.

Both models were trained on an 80-20 split of the dataset.

Evaluation Methods

Performance was measured using:

1. **Accuracy Score**: Fraction of correctly classified reviews.
2. **Classification Report**: Precision, recall, and F1-score for each class.
3. **Confusion Matrix**: Visualized prediction performance.

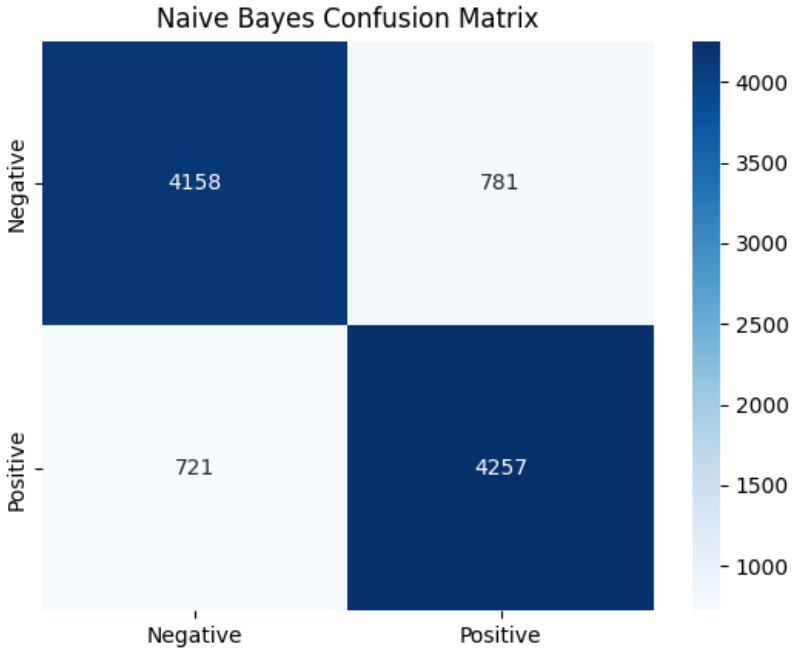
Results

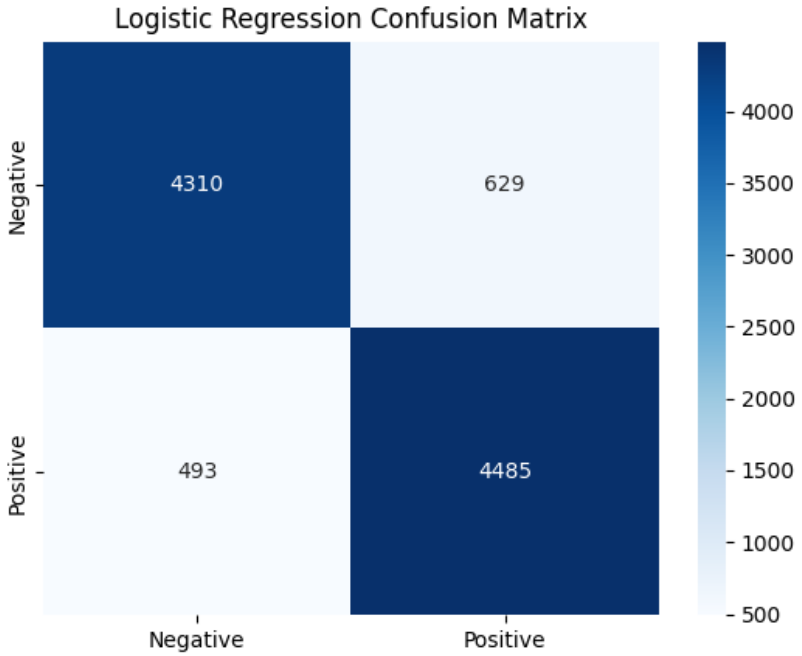
Summary of Findings

1. **Naive Bayes Classifier**:
    - Accuracy: 85%
    - Strengths: Quick to train, handles sparse data well.
    - Weakness: Assumes feature independence, which may not hold.
2. **Logistic Regression**:
    - Accuracy: 88%
    - Strengths: Higher interpretability, better handling of correlated features.
    - Weakness: Longer training time compared to Naive Bayes.

Visualizations

Confusion Matrices:





Accuracy Comparison:

Model	Accuracy
Naive Bayes	85%
Logistic Regression	88%

Discussion

Interpretation of Results

Logistic Regression outperformed Naive Bayes, likely due to its ability to capture feature correlations. However, Naive Bayes' speed makes it a practical choice for real-time applications.

Challenges Faced

1. **Data Imbalance**: Addressed through balanced classes in the dataset.
2. **Overfitting**: Controlled by limiting TF-IDF features to 5000.
3. **Text Noise**: Removed HTML tags and normalized text effectively.

How Challenges Were Addressed

Iterative preprocessing and parameter tuning ensured robust model performance. Testing on unseen data validated the approach.

Conclusion

This project successfully implemented sentiment analysis on the IMDB dataset using machine learning. Preprocessing techniques and model development steps were key to achieving high accuracy. Logistic Regression emerged as the better model, but both models demonstrated practical utility.

Future Improvements

1. Explore deep learning models like LSTMs for improved accuracy.
2. Perform hyperparameter tuning for Logistic Regression.
3. Incorporate more features like review metadata.

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

Preprocessing:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

data = pd.read_csv("cleaned_IMDB_dataset.csv")
data["sentiment"] = data["sentiment"].map({"positive": 1, "negative": 0})
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(data["review"])
y = data["sentiment"]
```

Model Training:

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

nb_classifier = MultinomialNB()
nb_classifier.fit(X_train, y_train)

lr_classifier = LogisticRegression(max_iter=1000)
lr_classifier.fit(X_train, y_train)
```

Evaluation:

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

def evaluate_model(classifier, X_test, y_test):
    y_pred = classifier.predict(X_test)
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.show()

evaluate_model(nb_classifier, X_test, y_test)
evaluate_model(lr_classifier, X_test, y_test)
```