Naive Bayes Classifier Implementation

Raghda Altaei

2024

1 Introduction

In this report, we utilize a Naive Bayes classifier to categorize news articles based on user ratings. The assumption is that a user's score for a news article is independent of their scores for other articles. Our objective is to evaluate the performance of the model by reporting the training and testing accuracy, as well as plotting the ROC curve. This code was written using PYTHON in Google Colab.

2 Problem Statement

Given a dataset where each row represents a user's ratings for different news articles, we aim to classify these ratings into categories (e.g., relevant or not relevant) using a Naive Bayes classifier.

3 Methodology

The following steps were implemented in the solution:

1. Load the training and test datasets.
2. Preprocess the data to extract features and target variables.
3. Train the Naive Bayes model using the training dataset.
4. Evaluate the model's accuracy on both the training and test datasets.
5. Generate and plot the ROC curve to visualize the model's performance.

4 Implementation

The following Python code was used to implement the Naive Bayes classifier:

```
# Import Required Libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
from scipy.io import loadmat

# Load Data
train_data = loadmat('/content/drive/MyDrive/Train_data.mat')
test_data = loadmat('/content/drive/MyDrive/Test_Data.mat')

# Extract features and target
X_train = train_data['train'][:, :-1]  # Features from training data
y_train = train_data['train'][:, -1]    # Target from training data
X_test = test_data['test'][:, :-1]      # Features from test data
y_test = test_data['test'][:, -1]        # Target from test data

# Train Naive Bayes Model
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)

# Evaluate the Model
y_train_pred = nb_model.predict(X_train)
y_test_pred = nb_model.predict(X_test)

# Calculate Accuracy
train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

print(f"Training accuracy: {train_accuracy:.4f}")
print(f"Test accuracy: {test_accuracy:.4f}")

# Generate ROC Curve
y_test_proba = nb_model.predict_proba(X_test)[:, 1]  # Get probabilities for class 1
fpr, tpr, thresholds = roc_curve(y_test, y_test_proba)
roc_auc = roc_auc_score(y_test, y_test_proba)

# Plot ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Naive Bayes Classifier')
```

```
plt.legend(loc='lower right')
plt.grid()
plt.show()
```

5 Results

The performance of the Naive Bayes classifier was evaluated, yielding the following results:

- Training accuracy: 0.8031
- Test accuracy: 0.7250

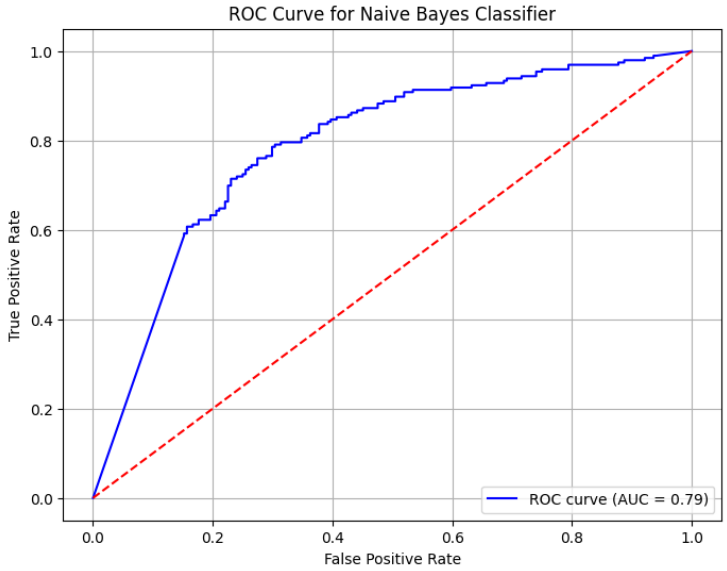

Figure 1: ROC Curve for Naive Bayes Classifier

6 Conclusion

The Naive Bayes classifier achieved a training accuracy of 80.31% and a test accuracy of 72.50%. The ROC curve indicates the trade-off between sensitivity and specificity, showcasing the classifier's performance across different thresholds.