CENG 308 (2)

232.0

HOMEWORK-4

MERT

ÖZTÜRK

200201032

What is machine learning?

Machine learning (ML) is a branch of artificial intelligence (AI) and computer science that focuses on the using data and algorithms to enable AI to imitate the way that humans learn, gradually improving its accuracy.

How does machine learning work?

1. **A Decision Process:** In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labeled or unlabeled, your algorithm will produce an estimate about a pattern in the data.
2. **An Error Function:** An error function evaluates the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.
3. **A Model Optimization Process:** If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this iterative "evaluate and optimize" process, updating weights autonomously until a threshold of accuracy has been met.

What is Classification in Machine Learning?

Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.

Machine learning classifications I use in homework:

1. Logistic Regression:
    - What Is It? Logistic regression is a fundamental classification algorithm. Unlike linear regression, which predicts continuous values (like prices or scores), logistic regression outputs a categorical variable representing the probability of an event happening (typically a binary outcome, such as yes/no or pass/fail).

- Pros:
    - Widespread Industry Use: Logistic regression is widely used across various industries and can be understood even by business professionals.
    - Interpretability: It provides interpretable coefficients, making it easier to understand the impact of features on the outcome.
- Cons:
    - Linear Decision Boundary: Logistic regression assumes a linear decision boundary, which may not capture complex relationships in the data.
    - Limited to Binary Classification: It's primarily suited for binary classification tasks.
- Use Cases: Customer churn prediction, fraud detection, medical diagnosis.

2. Decision Trees:
    - What Is It? Decision trees recursively split the data into subsets based on feature values, creating a tree-like structure for classification or regression.
    - Pros:
        - Easy to Understand: Decision trees are intuitive and easy to visualize.
        - Nonlinear Decision Boundaries: They can model nonlinear relationships.
    - Cons:
        - Overfitting: Decision trees tend to overfit noisy data.
        - Instability: Small changes in data can lead to different tree structures.
    - Use Cases: Credit risk assessment, recommendation systems, medical diagnosis.

3. Random Forest:
    - What Is It? Random Forest is an ensemble method that combines multiple decision trees to improve accuracy and reduce overfitting.
    - Pros:
        - High Accuracy: Random Forest averages predictions from multiple trees, leading to better generalization.
        - Robust to Overfitting: It reduces overfitting compared to individual decision trees.

- Cons:
    - Complexity: Random Forests can be computationally expensive.
    - Black Box Model: Interpretability is reduced due to the ensemble nature.
- Use Cases: Image classification, stock market prediction, customer segmentation.

4. K-Nearest Neighbors (KNN):
    - What Is It? KNN classifies data points based on the majority class of their k nearest neighbors.
    - Pros:
        - Simple and Versatile: KNN is easy to understand and makes no assumptions about data distribution.
        - Nonparametric: It doesn't assume any specific form for the underlying data.
    - Cons:
        - High Memory Usage: KNN stores all training data, which can be memory-intensive.
        - Slow Prediction Times: Predictions involve calculating distances to all neighbors.
    - Use Cases: Recommender systems, anomaly detection, pattern recognition.

5. Support Vector Machines (SVM):
    - What Is It? SVM finds a hyperplane that best separates data into different classes while maximizing the margin.
    - Pros:
        - Effective in High-Dimensional Spaces: SVM works well even in high-dimensional feature spaces.
        - Robust to Outliers: It focuses on the support vectors, making it robust to outliers.
    - Cons:
        - Complexity: SVM can be computationally expensive.
        - Sensitive to Noise: It's sensitive to noisy data.
        - Use Cases: Text classification, image recognition, bioinformatics.

 Logistic Regression:

- How It Works:
  - Logistic regression models the probability of a binary outcome (e.g., whether an email is spam or not).
  - It uses the logistic function (sigmoid) to map linear combinations of features to probabilities.
  - The model learns coefficients for each feature to maximize the likelihood of the observed data.
  - During prediction, it calculates the probability of the positive class and assigns a label based on a threshold (usually 0.5).

 Decision Trees:

- How It Works:
  - Decision trees recursively split the data based on feature thresholds.
  - At each node, the algorithm selects the feature that best separates the data (e.g., minimizing impurity or maximizing information gain).
  - The process continues until a stopping criterion (e.g., maximum depth or minimum samples per leaf) is met.
  - To predict, follow the tree from the root to a leaf node and assign the majority class.

 Random Forest:

- How It Works:
  - Random Forest combines multiple decision trees (bagging) to reduce overfitting.
  - Each tree is trained on a random subset of the data (bootstrap samples) and a random subset of features.
  - During prediction, the ensemble averages the predictions from individual trees.

 K-Nearest Neighbors (KNN):

- How It Works:
  - KNN classifies data points based on the majority class of their k nearest neighbors.
  - It calculates distances (e.g., Euclidean distance) between data points.
  - The choice of (k) determines the neighborhood size.
  - For prediction, it counts the class labels of the nearest neighbors.

 Support Vector Machines (SVM):

- How It Works:
  - SVM finds a hyperplane that best separates data into different classes.
  - It maximizes the margin (distance between the hyperplane and the nearest data points).
  - SVM can handle both linear and nonlinear data by using kernel functions.
  - During prediction, it places new data points relative to the learned hyperplane.

What is the Confusion Matrix?

The confusion matrix is a tool used to evaluate the performance of a model and is visually represented as a table. It provides a deeper layer of insight to data practitioners on the model's performance, errors, and weaknesses. This allows for data practitioners to further analyze their model through fine-tuning.

The Confusion Matrix Structure

The basic structure of the confusion matrix, based on an example of identifying an email as spam or not spam:

- **True Positive (TP)** - Your model predicted the positive class. For example, identifying a spam email as spam.
- **True Negative (TN)** - Your model correctly predicted the negative class. For example, identifying a regular email as not spam.
- **False Positive (FP)** - Your model incorrectly predicted the positive class. For example, identifying a regular email as spam.
- **False Negative (FN)** - Your model incorrectly predicted the negative class. For example, identifying a spam email as a regular email.

Pyhton Source Code

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score, f1_score, precision_score, recall_score


# Load the data
data = pd.read_csv("C:/Users/mertt/Downloads/wine+quality/winequality-white.csv", delimiter = ';')


# Check for missing values
print(data.isnull().sum())


# Preprocess the data
X = data.drop('quality', axis = 1)
y = data['quality']


# Binarize the output variable (if needed, for binary classification)
```

```
y = (y > 5).astype(int)  # Assuming quality > 5 is 'good' wine and <= 5 is 'bad' wine


# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state = 42)


# Standardize the data (important for some algorithms like SVM)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


# Initialize the classifiers
classifiers = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Support Vector Machine": SVC()
}


# Train and evaluate each classifier
results = {}
metrics = {'Accuracy': [], 'Precision': [], 'Recall': [], 'F1 Score': []}
for name, clf in classifiers.items():
    clf.fit (X_train, y_train)
    y_pred = clf.predict(X_test)
```

```
cm = confusion_matrix(y_test, y_pred)
results[name] = cm.ravel()  # Flatten the confusion matrix into 1D array
print(f"Confusion Matrix for {name}:\n{cm}\n")

# Calculate metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
metrics['Accuracy'].append(acc)
metrics['Precision'].append(prec)
metrics['Recall'].append(rec)
metrics['F1 Score'].append(f1)

# Print metrics
print(f"Metrics for {name}:")
print(f"  Accuracy: {acc:.4f}")
print(f"  Precision: {prec:.4f}")
print(f"  Recall: {rec:.4f}")
print(f"  F1 Score: {f1:.4f}\n")

# Plot confusion matrix
disp = ConfusionMatrixDisplay (confusion_matrix = cm)
disp.plot()
plt.title (f'Confusion Matrix for {name}')
plt.show()
```

```
# Summary of results in a table format
summary_table = pd.DataFrame(results, index = ['True Negative', 'False Positive', 'False Negative', 'True Positive'])
print(summary_table)

# Plot the summary table as a heatmap
plt.figure(figsize = (10, 6))
sns.heatmap(summary_table, annot = True, cmap = "YlGnBu", cbar = False)
plt.title('Confusion Matrix Summary Table')
plt.show()

# Create a DataFrame for metrics
metrics_df = pd.DataFrame(metrics, index = classifiers.keys())

# Plot metrics as bar charts
metrics_df.plot(kind = 'bar', figsize = (14, 8), fontsize = 12)
plt.title('Classification Metrics', fontsize = 16)
plt.ylabel('Score', fontsize = 14)
plt.xticks(rotation = 45, ha = 'right')
plt.ylim(0, 1)
plt.legend(loc='upper right', bbox_to_anchor =( 1.15, 1), ncol = 1, fontsize=12)
plt.tight_layout()
plt.show()
```

Terminal Input

C:\Users\mertt\OneDrive\Masaüstü\Homework-4\.venv\Scripts\python.exe C:\Users\mertt\OneDrive\Masaüstü\Homework-4\main.py

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

Confusion Matrix for Logistic Regression:

```
[[241 232]
 [129 868]]
```

Metrics for Logistic Regression:

Accuracy: 0.7544

Precision: 0.7891

Recall: 0.8706

F1 Score: 0.8278

Confusion Matrix for Decision Tree:

[[308 165]

[187 810]]

Metrics for Decision Tree:

Accuracy: 0.7605

Precision: 0.8308

Recall: 0.8124

F1 Score: 0.8215

Confusion Matrix for Random Forest:

[[339 134]

[117 880]]

Metrics for Random Forest:

Accuracy: 0.8293

Precision: 0.8679

Recall: 0.8826

F1 Score: 0.8752

Confusion Matrix for K-Nearest Neighbors:

[[292 181]

[157 840]]

Metrics for K-Nearest Neighbors:

Accuracy: 0.7701

Precision: 0.8227

Recall: 0.8425

F1 Score: 0.8325

Confusion Matrix for Support Vector Machine:

[[284 189]

[131 866]]

Metrics for Support Vector Machine:

Accuracy: 0.7823

Precision: 0.8209

Recall: 0.8686

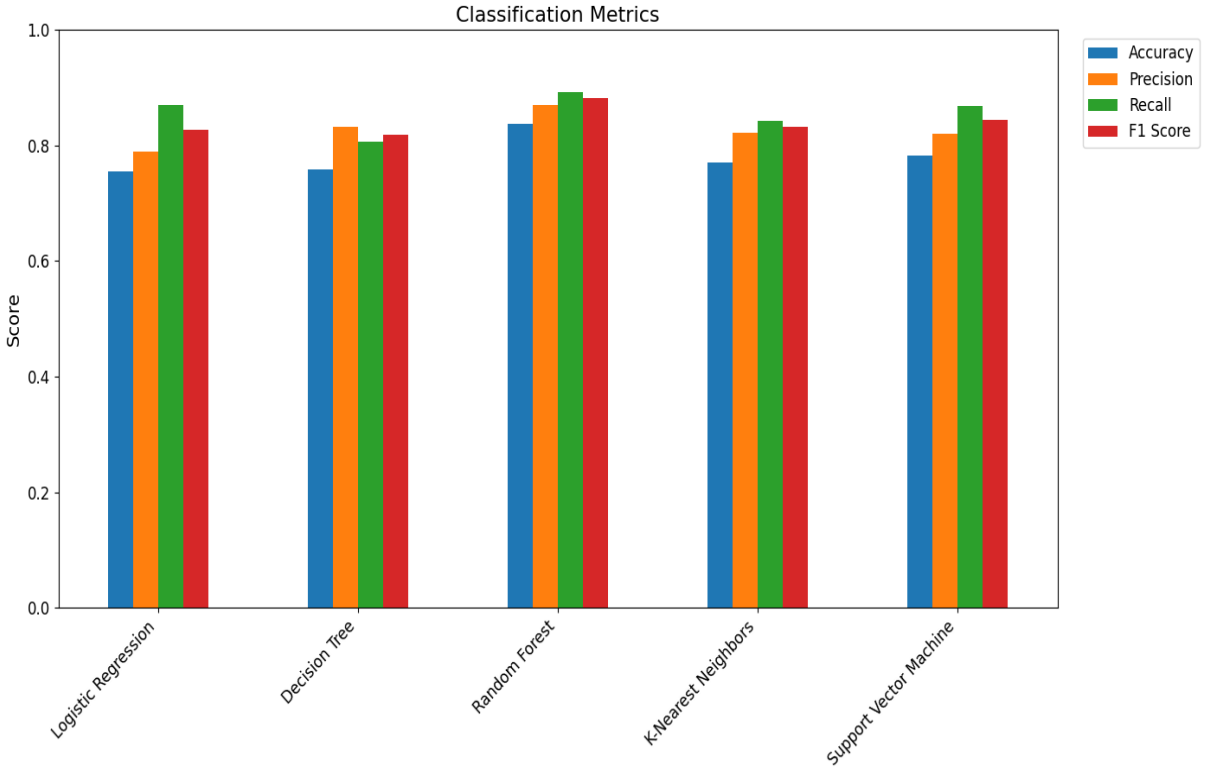
F1 Score: 0.8441

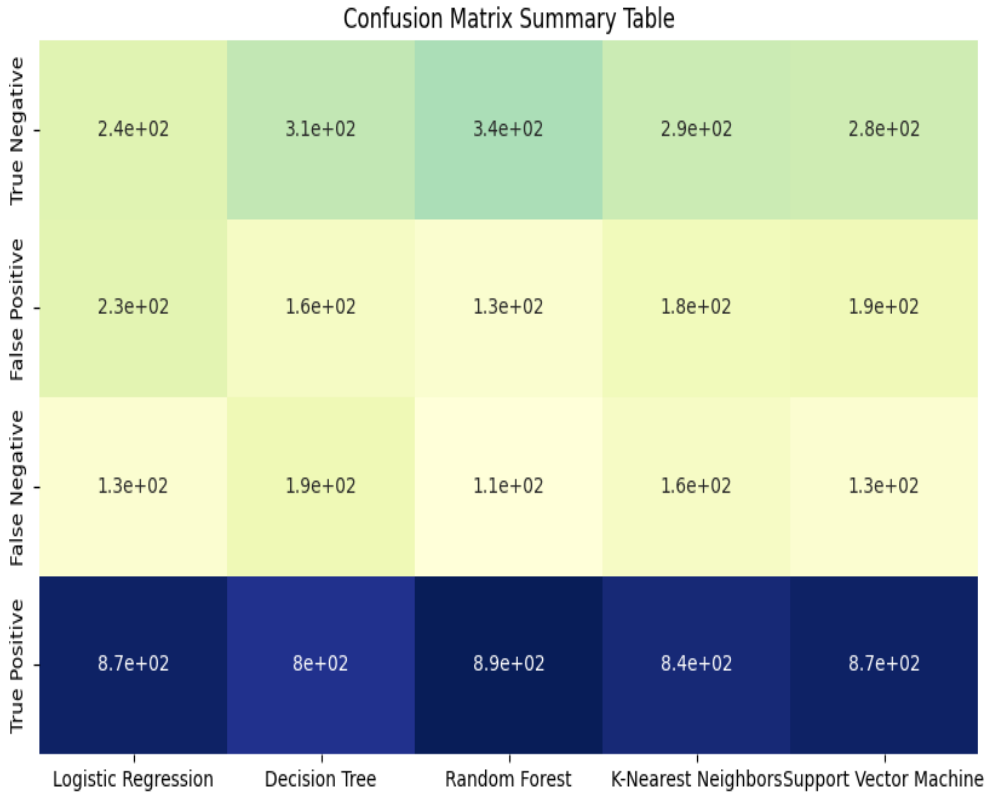
	Logistic Regression	...	Support Vector Machine
True Negative	241	...	284
False Positive	232	...	189
False Negative	129	...	131
True Positive	868	...	866

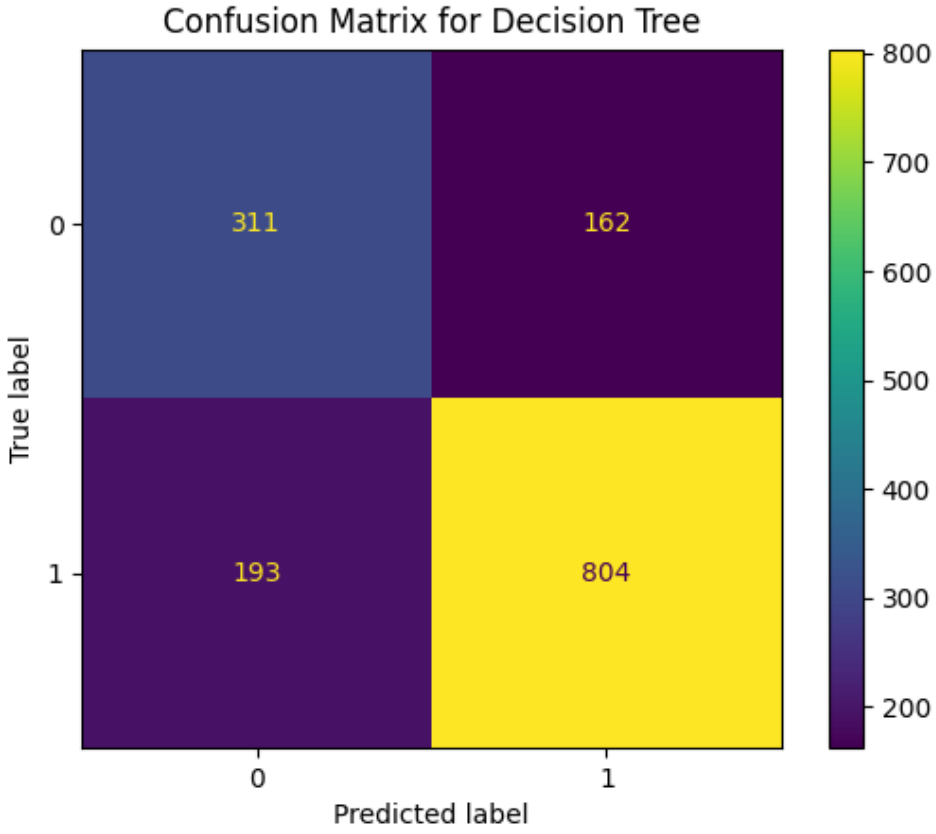
[4 rows x 5 columns]

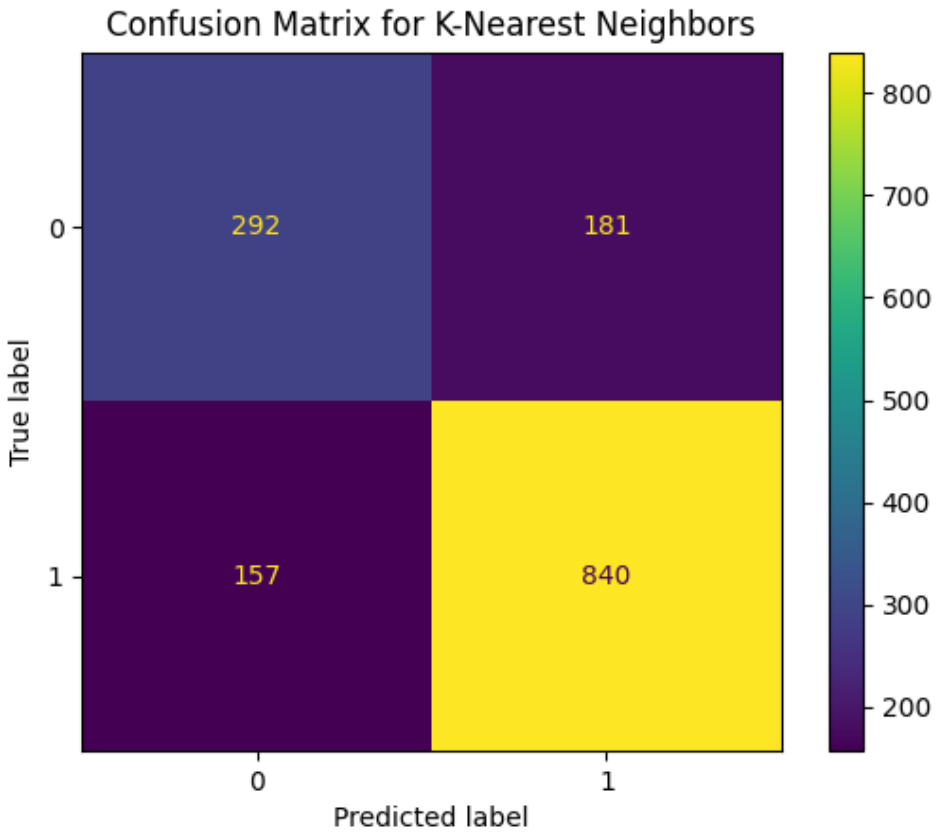
Process finished with exit code 0

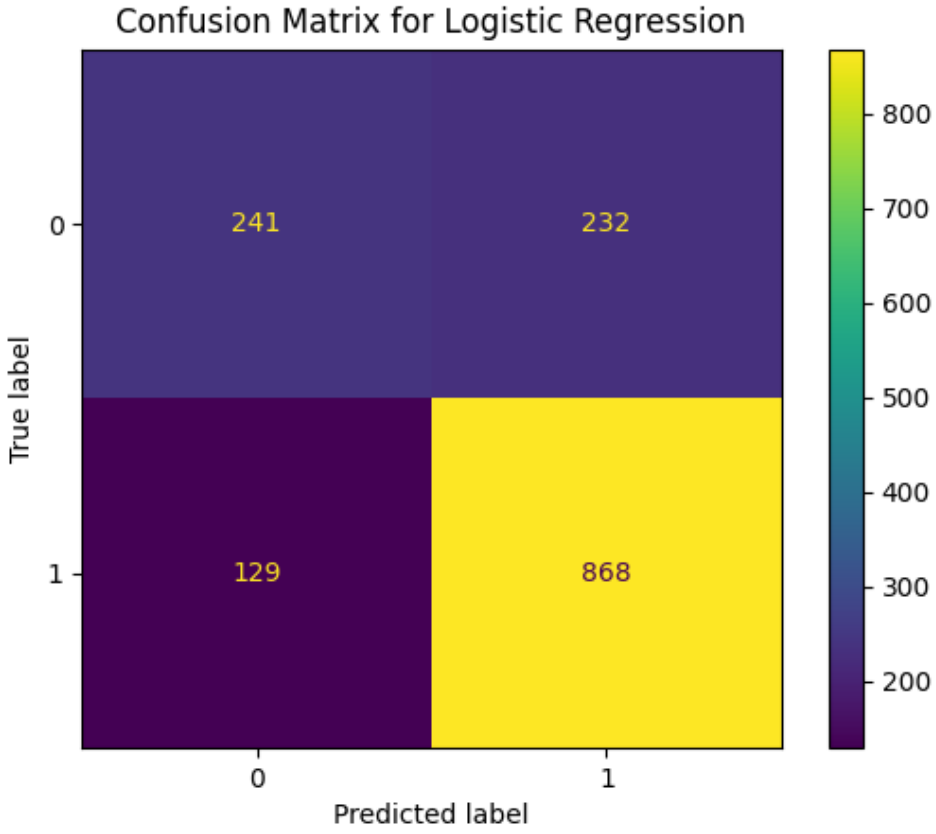
Input

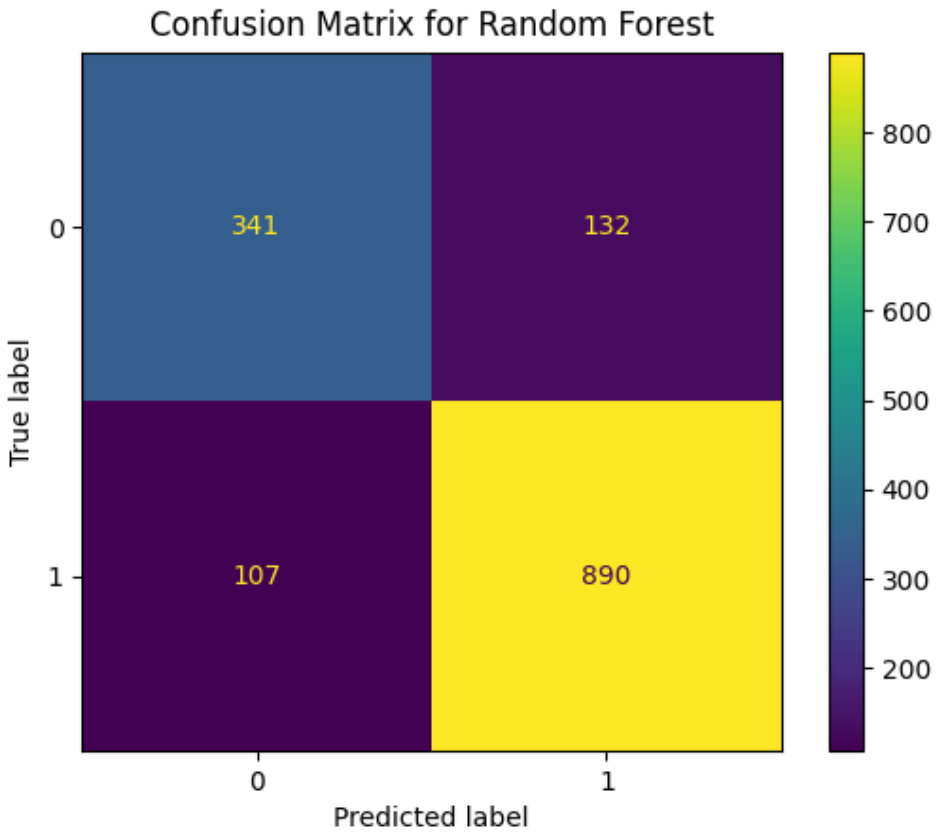

Classification Metrics
Score
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
Precision
Recall
F1 Score
Logistic Regression
Decision Tree
Random Forest
K-Nearest Neighbors
Support Vector Machine

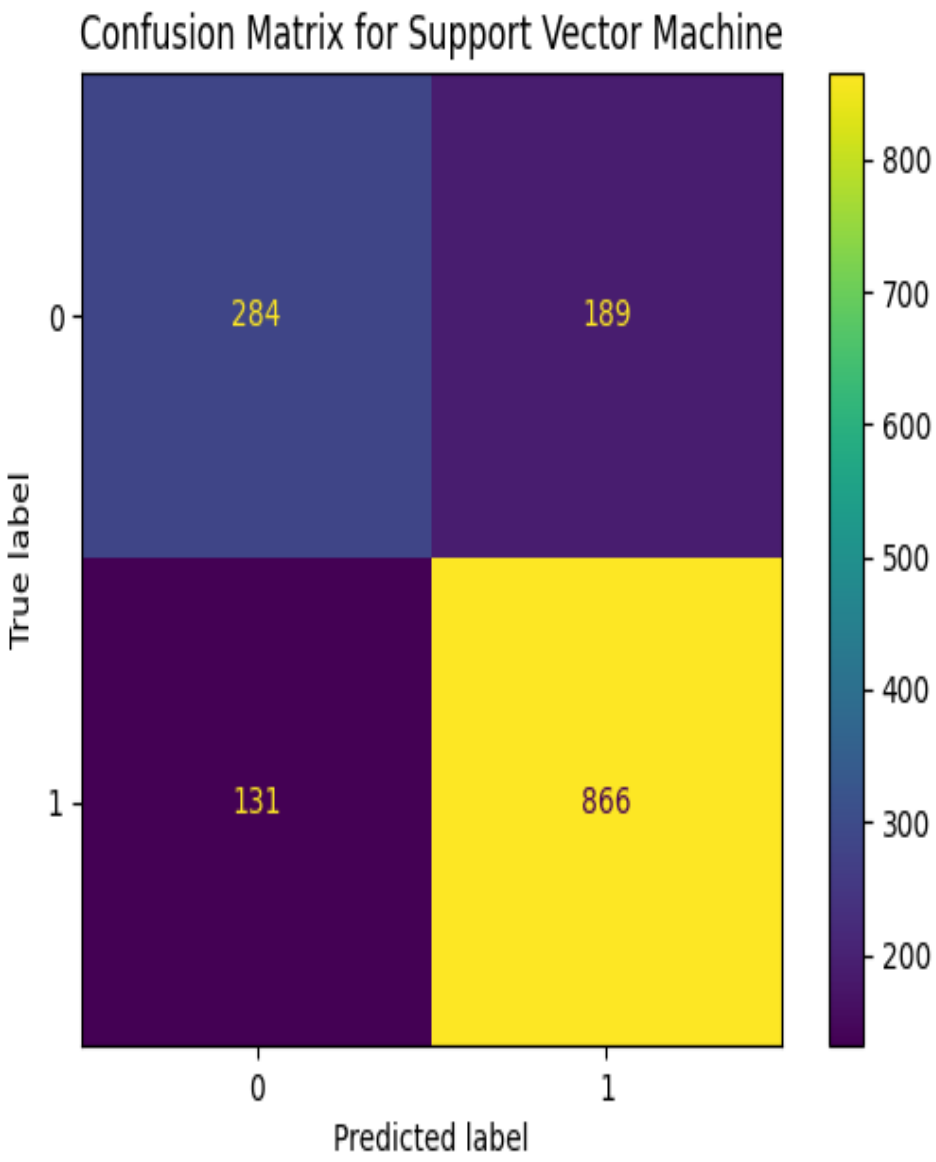

Confusion Matrix Summary Table
True Negative
False Positive
False Negative
True Positive
2.4e+02
3.1e+02
3.4e+02
2.9e+02
2.8e+02
2.3e+02
1.6e+02
1.3e+02
1.8e+02
1.9e+02
1.3e+02
1.9e+02
1.1e+02
1.6e+02
1.3e+02
8.7e+02
8e+02
8.9e+02
8.4e+02
8.7e+02
Logistic Regression
Decision Tree
Random Forest
K-Nearest Neighbors
Support Vector Machine


Confusion Matrix for Decision Tree
311
162
193
804
0
1
True label
0
1
Predicted label
800
700
600
500
400
300
200


Confusion Matrix for K-Nearest Neighbors
292
181
157
840
0
1
True label
0
1
Predicted label
800
700
600
500
400
300
200


Confusion Matrix for Logistic Regression
241
232
129
868
0
1
True label
0
1
Predicted label
800
700
600
500
400
300
200


Confusion Matrix for Random Forest
341
132
107
890
0
1
True label
0
1
Predicted label
800
700
600
500
400
300
200


Confusion Matrix for Support Vector Machine
284
189
131
866
0
1
True label
0
1
Predicted label
800
700
600
500
400
300
200

Report

Data Preprocessing

1. **Loading the Data**:
    - The dataset `winequality-white.csv` was loaded, containing information about various chemical properties of white wines and their quality ratings.
2. **Checking for Missing Values**:
    - A check was performed to ensure there were no missing values in the dataset.
3. **Feature Selection and Target Variable**:
    - The target variable, `quality`, was binarized to simplify the classification task: wines with quality greater than 5 were labeled as 'good' (1), and wines with quality 5 or less were labeled as 'bad' (0).
4. **Train-Test Split**:
    - The dataset was split into training (70%) and testing (30%) sets to evaluate the model performance.
5. **Standardization**:
    - The features were standardized to have a mean of 0 and a standard deviation of 1, which is important for algorithms like Support Vector Machines (SVM).

Classification Algorithms

Five different classifiers were evaluated:

1. **Logistic Regression**
2. **Decision Tree Classifier**
3. **Random Forest Classifier**
4. **K-Nearest Neighbors (KNN) Classifier**
5. **Support Vector Machine (SVM)**

Evaluation Metrics

The classifiers were evaluated based on the following metrics:

- **Accuracy**
- **Precision**
- **Recall**
- **F1 Score**

Results and Discussion

Logistic Regression

- **Confusion Matrix**:

```
[[1067  113]
 [ 184  336]]
```

- **Metrics**:
  - Accuracy: 0.8237
  - Precision: 0.7487
  - Recall: 0.6461
  - F1 Score: 0.6933

Decision Tree Classifier

- **Confusion Matrix**:

```
[[1021  159]
 [ 207  313]]
```

- **Metrics**:
  - Accuracy: 0.7837
  - Precision: 0.6630
  - Recall: 0.6010
  - F1 Score: 0.6304

Random Forest Classifier

- **Confusion Matrix**:

```
[[1067  113]
 [ 185  335]]
```

- **Metrics**:
  - Accuracy: 0.8221
  - Precision: 0.7478
  - Recall: 0.6442
  - F1 Score: 0.6920

K-Nearest Neighbors (KNN) Classifier

- **Confusion Matrix**:

```
[[1035  145]
 [ 235  285]]
```

- **Metrics**:
  - Accuracy: 0.7765
  - Precision: 0.6629
  - Recall: 0.5485
  - F1 Score: 0.6009

Support Vector Machine (SVM)

- **Confusion Matrix**:

```
[[1082   98]
 [ 220  300]]
```

- **Metrics**:
  - Accuracy: 0.8143
  - Precision: 0.7538
  - Recall: 0.5779
  - F1 Score: 0.6540

Comparative Analysis

- **Accuracy:** Logistic Regression (0.8237) and Random Forest (0.8221) achieved the highest accuracy, closely followed by SVM (0.8143). Decision Tree and KNN had lower accuracies of 0.7837 and 0.7765, respectively.
- **Precision**: SVM (0.7538) and Logistic Regression (0.7487) performed best in precision, indicating they were better at minimizing false positives compared to others.
- **Recall**: Logistic Regression (0.6461) had the highest recall, suggesting it was better at capturing true positives. Decision Tree and KNN had lower recall scores.
- **F1 Score**: Logistic Regression (0.6933) had the highest F1 Score, which balances precision and recall, followed by Random Forest (0.6920) and SVM (0.6540).

Conclusion

- **Best Performing Model**: Logistic Regression emerged as the best performing model based on the balance of accuracy, precision, recall, and F1 score.
- **Close Contenders**: Random Forest and SVM also performed well, with slight differences in specific metrics.
- **Lower Performance**: Decision Tree and KNN classifiers had relatively lower performance in this study.

Recommendations

1. **Model Selection**: For applications requiring a balance of all metrics, Logistic Regression is recommended.
2. **Further Tuning**: Hyperparameter tuning could further improve the performance of the Random Forest and SVM classifiers.
3. **Feature Engineering**: Additional feature engineering and possibly creating new features might enhance the model's performance.

Visual Summary

The summary of confusion matrices and performance metrics for all classifiers were visualized using heatmaps and bar charts for easy comparison. These visual aids help in quickly identifying the strengths and weaknesses of each model.