Model Optimization and Tuning Phase Report

Date	01 May 2024
Team ID	Team-738315
Project Title	Online Payment Fraud Detection using Machine Learning
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th><th>Optimal Values</th></tr>
<tr><td>Random Forest</td><td><pre>
def RandomForest(X_train, X_test, y_train, y_test):
    # Initialize the Random Forest classifier
    model = RandomForestClassifier()

    # Train the model
    model.fit(X_train, y_train)

    # Predictions on the training set
    y_train_pred = model.predict(X_train)
    train_accuracy = accuracy_score(y_train, y_train_pred)
    print("Train Accuracy:", train_accuracy)

    # Predictions on the test set
    y_test_pred = model.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    print("Test Accuracy:", test_accuracy)
</pre></td><td><pre>
# Get the best parameters and the best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print("Best Parameters:", best_params)
print("Best F1 Score:", best_score)

Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Best F1 Score: 0.538755980861244
</pre></td></tr>
<tr><td>Decision Tree</td><td><pre>
def DecisionTree(X_train, X_test, y_train, y_test):
    # Initialize the Decision Tree classifier
    model = DecisionTreeClassifier()

    # Train the model
    model.fit(X_train, y_train)

    # Predictions on the training set
    y_train_pred = model.predict(X_train)
    train_accuracy = accuracy_score(y_train, y_train_pred)
    print("Train Accuracy:", train_accuracy)

    # Predictions on the test set
    y_test_pred = model.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    print("Test Accuracy:", test_accuracy)
</pre></td><td><pre>
# Get the best parameters and the best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print("Best Parameters:", best_params)
print("Best F1 Score:", best_score)

Best Parameters: {'max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 2}
Best F1 Score: 0.5677569786535304
</pre></td></tr>
</table>

<table>
<tr>
<td>SVM</td>
<td><pre>
def SVM(X_train, X_test, y_train, y_test):
    # Initialize the SVM classifier
    model = SVC()

    # Train the model
    model.fit(X_train, y_train)

    # Predictions on the training set
    y_train_pred = model.predict(X_train)
    train_accuracy = accuracy_score(y_train, y_train_pred)
    print("Train Accuracy:", train_accuracy)

    # Predictions on the test set
    y_test_pred = model.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    print("Test Accuracy:", test_accuracy)
</pre></td>
<td></td>
</tr>
<tr>
<td>XG Boosting</td>
<td><pre>
def XGBoost(X_train, X_test, y_train, y_test):
    # Initialize the XGBoost classifier
    model = XGBClassifier()

    # Train the model
    model.fit(X_train, y_train)

    # Predictions on the training set
    y_train_pred = model.predict(X_train)
    train_accuracy = accuracy_score(y_train, y_train_pred)
    print("Train Accuracy:", train_accuracy)

    # Predictions on the test set
    y_test_pred = model.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    print("Test Accuracy:", test_accuracy)
</pre></td>
<td><pre>
print("Best Parameters:", best_params)
print("Best F1 Score:",best_score)

Best Parameters: {'gamma': 0, 'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}
Best F1 Score: 0.6770698287220027
</pre></td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

<table>
<tr>
<th>Model</th>
<th>Optimized Metric</th>
</tr>
<tr>
<td>Decision Tree</td>
<td><pre>
classification_rep=classification_report(y_test, y_pred)
print(classification_rep)

              precision   recall f1-score   support

         0.0   1.00   1.00   1.00   120053
         1.0   0.58   0.61   0.60   70

    accuracy   1.00   120123
   macro avg   0.79   0.81   0.80   120123
weighted avg   1.00   1.00   1.00   120123

from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)

0.9982184775505403
array([[5641,   1],
       [8,   10]])
</pre></td>
</tr>
</table>

Random Forest

```
# Generate the classification report
report = classification_report(y_test, y_pred)
print(report)

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00    120053
         1.0       0.96      0.61      0.75        70

    accuracy                           1.00    120123
   macro avg       0.98      0.81      0.87    120123
weighted avg       1.00      1.00      1.00    120123
```

```
confusion_matrix(y_test, y_pred)

array([[120051,      2],
       [    27,     43]])
```

SVM

```
report = classification_report(y_test, y_pred)
print(report)

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      5642
         1.0       0.00      0.00      0.00        18

    accuracy                           1.00      5660
   macro avg       0.50      0.50      0.50      5660
weighted avg       0.99      1.00      1.00      5660
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)

array([[5642,    0],
       [  18,    0]])
```

XG Boosting

```
report = classification_report(y_test, y_pred)
print(report)

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00    111795
         1.0       0.89      0.60      0.71        42

    accuracy                           1.00    111837
   macro avg       0.95      0.80      0.86    111837
weighted avg       1.00      1.00      1.00    111837
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)

array([[111792,      3],
       [    17,     25]])
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Decision Tree	The Decision Tree model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.