Model Optimization and Tuning Phase Report

Date	10s July 2024
Team ID	740092
Project Title	Credit card approval prediction using ML
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>Decision Tree</td><td><pre>
# Define the Decision Tree classifier
dt_classifier = DecisionTreeClassifier()

# Define the hyperparameters and their possible values for tuning
param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
</pre></td><td><pre>
# Evaluate the performance of the tuned model
accuracy = accuracy_score(y_test, y_pred)
print(f'Optimal Hyperparameters: {best_params}')
print(f'Accuracy on Test Set: {accuracy}')

Optimal Hyperparameters: {'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'best'}
Accuracy on Test Set: 0.7159763313609467
</pre></td></tr>
</table>

<table>
<tr>
<td>Random Forest</td>
<td><pre>
# Define the Random Forest classifier
rf_classifier = RandomForestClassifier()

# Define the hyperparameters and their possible values for tuning
param_grid = {
    'n_estimators': [50, 100, 200],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
}
</pre></td>
<td><pre>
# Evaluate the performance of the tuned model
accuracy = accuracy_score(y_test, y_pred)
print(f'Optimal Hyperparameters: {best_params}')
print(f'Accuracy on Test Set: {accuracy}')

Optimal Hyperparameters: {'criterion': 'entropy', 'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Accuracy on Test Set: 0.7751479289940828
</pre></td>
</tr>
</table>

<table>
<tr>
<td>Logistic Regression</td>
<td><pre>
lr_classifier = LogisticRegressionclassifier()
#define hyperparameters and their possible values for tuning
param_grid_lr = {
    'penalty': ['l1', 'l2', 'elasticnet', 'none'],
    'C': [0.01, 0.1, 1, 10, 100],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'max_iter': [100, 200, 300],
    'fit_intercept': [True, False]
}
</pre></td>
<td><pre>
# Evaluate the performance of the tuned model
accuracy = accuracy_score(y_test, y_pred)
print(f'Optimal Hyperparameters: {best_params}')
print(f'Accuracy on Test Set: {accuracy}')

Optimal Hyperparameters: {'n_neighbors': 9, 'p': 1, 'weights': 'distance'}
Accuracy on Test Set: 0.7218934911242604
</pre></td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td><pre>
# Define the Gradient Boosting classifier
gb_classifier = GradientBoostingClassifier()

# Define the hyperparameters and their possible values for tuning
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'subsample': [0.8, 1.0]
}
</pre></td>
<td><pre>
# Evaluate the performance of the tuned model
accuracy = accuracy_score(y_test, y_pred)
print(f'Optimal Hyperparameters: {best_params}')
print(f'Accuracy on Test Set: {accuracy}')

Optimal Hyperparameters: {'learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100, 'subsample': 0.8}
Accuracy on Test Set: 0.7928994082840237
</pre></td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric

Decision Tree

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      2692
           1       1.00      0.99      1.00      2335

    accuracy                           1.00      5027
   macro avg       1.00      1.00      1.00      5027
weighted avg       1.00      1.00      1.00      5027
```

```
print(classification_report (ytest, ypred))
```

```
print("Classification report")
```

```
Confusion matrix
[[2685    7]
 [  15 2320]]
```

Random Forest

```
              precision    recall  f1-score   support

Not Approved       0.80      0.85      0.82       500
Approved           0.83      0.78      0.80       500

    accuracy                           0.81      1000
   macro avg       0.81      0.81      0.81      1000
weighted avg       0.81      0.81      0.81      1000
```

```
print(confusion_matrix(ytest,ypred))
```

```
Confusion matrix
[[2617   75]
 [ 199 2136]]
```

Logistic Regression

```
Classification report
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      2692
           1       0.97      0.91      0.94      2335

    accuracy                           0.95      5027
   macro avg       0.95      0.94      0.94      5027
weighted avg       0.95      0.95      0.95      5027
```

```
confusion_matrix(y_test,ypred)
```

```
array([[43, 32],
       [29, 65]])
```

```
print(classification_report(ytest, ypred))
```

Gradient Boosting	print(classification_report(ytest,ypred)) Classification report precision recall f1-score support 0 1.00 1.00 1.00 2692 1 1.00 1.00 1.00 2335 accuracy 1.00 5027 macro avg 1.00 1.00 1.00 5027 weighted avg 1.00 1.00 1.00 5027 confusion_matrix(y_test,ypred) array([[63, 12], [26, 68]])

Final Model Selection Justification (2 Marks):

Final Model	Reasoning

Gradient Boosting	The Gradient Boosting 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.