Model Optimization and Tuning Phase Report

Date	15 March 2024
Team ID	SWTID1749835773
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
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>
#parameter selection
param_grid = {
    "criterion": ["gini", "entropy"],
    "max_depth":np.arange(3,20)
}
model = DecisionTreeClassifier()
grid_search_dctree = GridSearchCV(model, param_grid, cv=5, verbose=1, n_jobs=-1)
grid_search_dctree.fit(x_train, y_train)
print(f"Best hyperparameters found by Grid Search: {grid_search_dctree.best_params_}")
</pre></td><td><pre>
print(f"Best hyperparameters found by Grid Search: {grid_search_dctree.best_params_}")
model_opt = grid_search_dctree.best_estimator_
y_predict = model_opt.predict(x_test)
accuracy = accuracy_score(y_test, y_predict)
print(f"Accuracy: {accuracy}")

Fitting 5 folds for each of 34 candidates, totalling 170 fits
Best hyperparameters found by Grid Search: {'criterion': 'gini', 'max_depth': np.int64(18)}
Accuracy: 0.727810650887574
</pre></td></tr>
<tr><td>Random
Forest</td><td><pre>
param_grid = {'criterion': ['gini', 'entropy'],
    'n_estimators': [50, 100, 150],
    'max_depth':np.arange(0,21)
    }

rf_model = RandomForestClassifier(random_state=42)
</pre></td><td><pre>
# Make predictions on the test set
y_pred = best_rf_model.predict(x_test)

accuracy_randomforest = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy_randomforest}")

Fitting 5 folds for each of 126 candidates, totalling 630 fits
Best hyperparameters found by Grid Search: {'criterion': 'gini', 'max_depth': np.int64(1), 'n_estimators': 50}
Accuracy: 0.7886178861788617
</pre></td></tr>
</table>

<table>
<tr>
<td>KNN</td>
<td><pre>
#parameter selection

param_grid = {
    'n_neighbors': np.arange(1, 11),   # Number of neighbors to try
    'weights': ['uniform', 'distance'],   # Weight function used in prediction
    'p': [1, 2]   # Power parameter for Minkowski metric (1=Manhattan, 2=Euclidean)
}

model_knn = KNeighborsClassifier()
grid_search = GridSearchCV(model_knn, param_grid, cv=5, scoring='accuracy', n_jobs=-1, verbose=1)
grid_search.fit(X_train_scaled, y_train)
</pre></td>
<td><pre>
print(f"Best hyperparameters found by Grid Search: {grid_search.best_params_}")
print(f"Classification Accuracy Score: {accuracy:.2f}")
print("Classification Report:\n", report)

Best hyperparameters found by Grid Search: {'n_neighbors': np.int64(10), 'p': 1, 'weights': 'uniform'}
Classification Accuracy Score: 0.74
</pre></td>
</tr>
<tr>
<td>XGB
Boost</td>
<td><pre>
    param_grid = {
        'learning_rate': [0.5,0.1, 0.01, 0.05],
        'n_estimators': [50, 100, 200]}

    model = XGBClassifier(random_state=42, eval_metric='logloss')
</pre></td>
<td><pre>
print("Best parameters:", grid_search.best_params_)
best_model = grid_search.best_estimator_

y_pred_xgb = best_model.predict(x_test)
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
print(f"XGBoost (n_estimators=100, learning_rate=0.1) Accuracy: {accuracy_xgb:.4f}")
print("XGBoost Classification Report:")
print(classification_report(y_test, y_pred_xgb))

Best parameters: {'learning_rate': 0.01, 'n_estimators': 50}
XGBoost (n_estimators=100, learning_rate=0.1) Accuracy: 0.7886
</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>
print(classification_report(y_test, y_predict))

              precision   recall f1-score   support

           0   0.76   0.74   0.75   94
           1   0.69   0.71   0.70   75

    accuracy   0.73   169
   macro avg   0.72   0.73   0.73   169
weighted avg   0.73   0.73   0.73   169
</pre></td>
</tr>
</table>

<table>
<tr><td>Random Forest</td><td>

```
print(classification_report(y_test,y_pred))
print(f1_score)
```

```
Accuracy: 0.7886178861788617
              precision    recall  f1-score   support

           0       0.95      0.42      0.58        43
           1       0.76      0.99      0.86        80

    accuracy                           0.79       123
   macro avg       0.85      0.70      0.72       123
weighted avg       0.83      0.79      0.76       123
```

</td></tr>
<tr><td>KNN</td><td>

```
print("Classification Report:\n", report)
```

```
Best hyperparameters found by Grid Search: {'n_neighbors': np.int64(10), 'p': 1, 'weights': 'uniform'
Classification Accuracy Score: 0.74
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.30      0.45        43
           1       0.72      0.97      0.83        80

    accuracy                           0.74       123
   macro avg       0.79      0.64      0.64       123
weighted avg       0.77      0.74      0.70       123
```

</td></tr>
<tr><td>XGBoost</td><td>

```
print("XGBoost Classification Report:")
print(classification_report(y_test, y_pred_xgb))
```

```
Best parameters: {'learning_rate': 0.01, 'n_estimators': 50}
XGBoost (n_estimators=100, learning_rate=0.1) Accuracy: 0.7886
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.42      0.58        43
           1       0.76      0.99      0.86        80

    accuracy                           0.79       123
   macro avg       0.85      0.70      0.72       123
weighted avg       0.83      0.79      0.76       123
```

</td></tr>
</table>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	The random forest model is chosen for its simplicity over a complex model like XGBoost .Tuning of hyperparameters is easier, along with the speed to process clean and simple data.