Model Optimization and Tuning Phase Report

Date	19 May 2025
Team ID	SWTID1750233055
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase focuses on improving the performance of machine learning models by refining their configurations. This stage ensures that the chosen algorithm not only performs well on the training data but also generalizes effectively to unseen data.

Model	Grid Search	Optimal Values

Decision Tree	`param_grid = {`
	`'n_estimators': [10, 25, 50, 75, 100],`
	`'learning_rate': [0.5, 1.0, 1.5],`
	`}`
	
	`# GridSearchCV setup`
	`grid = GridSearchCV(estimator=abc, param_grid=param_grid, cv=5, n_jobs=-1)`
	`grid.fit(X_train, y_train)`
		`# Best model`
		`print("Best Parameters:", grid.best_params_)`
		
		`# Evaluate`
		`y_pred_grid = grid.predict(X_test)`
		`print("Grid Search Accuracy:", accuracy_score(y_test, y_pred_grid))`
		
		`Best Parameters: {'learning_rate': 0.5, 'n_estimators': 75}`
		`Grid Search Accuracy: 0.7553191489361702`

<table>
<tr>
<td>Ada Boost</td>
<td><pre>param_grid = {
    'n_estimators': [10, 25, 50, 75, 100],
    'learning_rate': [0.5, 1.0, 1.5],

}

# GridSearchCV setup
grid = GridSearchCV(estimator=abc, param_grid=param_grid, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)</pre></td>
<td><pre># Best model
print("Best Parameters:", grid.best_params_)

# Evaluate
y_pred_grid = grid.predict(X_test)
print("Grid Search Accuracy:", accuracy_score(y_test, y_pred_grid))</pre>
<pre>Best Parameters: {'learning_rate': 0.5, 'n_estimators': 75}
Grid Search Accuracy: 0.7553191489361702</pre></td>
</tr>
</table>

<table>
<tr>
<td>KNN</td>
<td><pre>param_grid = {
    'n_estimators': [10, 25, 50, 75, 100],
    'learning_rate': [0.5, 1.0, 1.5],

}

# GridSearchCV setup
grid = GridSearchCV(estimator=abc, param_grid=param_grid, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)</pre></td>
<td><pre># Best model
print("Best Parameters:", grid.best_params_)

# Evaluate
y_pred_grid = grid.predict(X_test)
print("Grid Search Accuracy:", accuracy_score(y_test, y_pred_grid))</pre>
<pre>Best Parameters: {'learning_rate': 0.5, 'n_estimators': 75}
Grid Search Accuracy: 0.7553191489361702</pre></td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td><pre>param_grid = {
    'n_estimators': [10, 25, 50, 75, 100],
    'learning_rate': [0.5, 1.0, 1.5],

}

# GridSearchCV setup
grid = GridSearchCV(estimator=abc, param_grid=param_grid, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)</pre></td>
<td><pre># Best model
print("Best Parameters:", grid.best_params_)

# Evaluate
y_pred_grid = grid.predict(X_test)
print("Grid Search Accuracy:", accuracy_score(y_test, y_pred_grid))</pre>
<pre>Best Parameters: {'learning_rate': 0.5, 'n_estimators': 75}
Grid Search Accuracy: 0.7553191489361702</pre></td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric

Decision Tree	`dec=DecisionTreeClassifier(random_state=49)`
	`base = DecisionTreeClassifier(max_depth=3)`
	`dec.fit(X_train,y_train)`
	`pred_abc = dec.predict(X_test)`
	`print(classification_report(y_test,pred_abc))`
	`print(confusion_matrix(y_test,pred_abc))`
	
	`              precision    recall  f1-score   support`
	
	`           0       0.68      0.66      0.67       184`
	`           1       0.68      0.70      0.69       192`
	
	`    accuracy                           0.68       376`
	`   macro avg       0.68      0.68      0.68       376`
	`weighted avg       0.68      0.68      0.68       376`
	
	`[[121  63]`
	` [ 58 134]]`

Ada Boost	`abc = AdaBoostClassifier(random_state=99)`
	`base = DecisionTreeClassifier(max_depth=3)`
	`abc.fit(X_train,y_train)`
	`pred_abc = abc.predict(X_test)`
	`print(classification_report(y_test,pred_abc))`
	`print(confusion_matrix(y_test,pred_abc))`
	
	`C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_w`
	`nd will be removed in 1.6. Use the SAMME algorithm to circumve`
	`  warnings.warn(`
	`              precision    recall  f1-score   support`
	
	`           0       0.72      0.75      0.74       184`
	`           1       0.75      0.72      0.74       192`
	
	`    accuracy                           0.74       376`
	`   macro avg       0.74      0.74      0.74       376`
	`weighted avg       0.74      0.74      0.74       376`
	
	`[[138  46]`
	` [ 53 139]]`

KNN	``` kn=KNeighborsClassifier() base = DecisionTreeClassifier(max_depth=3) kn.fit(X_train,y_train) pred_abc = kn.predict(X_test) print(classification_report(y_test,pred_abc)) print(confusion_matrix(y_test,pred_abc)) ```
Gradient Boosting	``` gb=GradientBoostingClassifier(random_state=49) base = DecisionTreeClassifier(max_depth=3) gb.fit(X_train,y_train) pred_abc = gb.predict(X_test) print(classification_report(y_test,pred_abc)) print(confusion_matrix(y_test,pred_abc)) ```

KNN output:

```
              precision    recall  f1-score   support

           0       0.65      0.70      0.67       184
           1       0.69      0.65      0.67       192

    accuracy                           0.67       376
   macro avg       0.67      0.67      0.67       376
weighted avg       0.67      0.67      0.67       376

[[128  56]
 [ 68 124]]
```

Gradient Boosting output:

```
              precision    recall  f1-score   support

           0       0.73      0.76      0.74       184
           1       0.76      0.72      0.74       192

    accuracy                           0.74       376
   macro avg       0.74      0.74      0.74       376
weighted avg       0.74      0.74      0.74       376

[[140  44]
 [ 53 139]]
```

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Ada Boost	The **AdaBoost model** was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to focus on difficult-to-classify instances, reduce variance, and maintain strong generalization makes it well-suited for this project. AdaBoost's effectiveness in handling imbalanced and noisy data, along with its lightweight nature and ease of integration into real-time systems, aligns with the project objectives — justifying its selection as the final model.