Model Optimization and Tuning Phase Report

Date	03 October 2024
Team ID	LTVIP2024TMID24947
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):

Model	Tuned Hyperparameters	Optimal Values

Decision Tree	`# Hyperparameter tuning for Decision Tree`
	`param_grid_dt = {`
	`    'max_depth': [None, 5, 10],  # Limit max depth`
	`    'min_samples_split': [2, 5]  # Limit minimum samples to split`
	`}`
	
	`dt_grid = GridSearchCV(estimator=DecisionTreeClassifier(), param_grid=param_grid_dt, cv=3, scoring='accuracy', n_jobs=-1)`
	`dt_grid.fit(x_train_scaled, y_train)`
	
	`print("Best Decision Tree Parameters:", dt_grid.best_params_)`
	`print("Best Decision Tree Accuracy:", dt_grid.best_score_)`
		`Best Random Forest Parameters: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 100}`
		`Best Random Forest Accuracy: 0.9795041642814564`
		`Best KNN Parameters: {'n_neighbors': 7, 'weights': 'distance'}`
		`Best KNN Accuracy: 0.918009713656467`
		`Best Decision Tree Parameters: {'max_depth': None, 'min_samples_split': 2}`
		`Best Decision Tree Accuracy: 0.981259833367022`
		`Best XGBoost Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}`
		`Best XGBoost Accuracy: 0.9830165311147149`

Random Forest	`# Hyperparameter tuning for Random Forest`
	`param_grid_rf = {`
	`    'n_estimators': [50, 100],  # Reduced number of estimators`
	`    'max_depth': [None, 10],  # Limit max depth`
	`    'min_samples_split': [2, 5]  # Limit minimum samples to split`
	`}`
	
	`rf_grid = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_grid_rf, cv=3, scoring='accuracy', n_jobs=-1)`
	`rf_grid.fit(x_train_scaled, y_train)`
	
	`print("Best Random Forest Parameters:", rf_grid.best_params_)`
	`print("Best Random Forest Accuracy:", rf_grid.best_score_)`
		`Best Random Forest Parameters: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 100}`
		`Best Random Forest Accuracy: 0.9795041642814564`
		`Best KNN Parameters: {'n_neighbors': 7, 'weights': 'distance'}`
		`Best KNN Accuracy: 0.918009713656467`
		`Best Decision Tree Parameters: {'max_depth': None, 'min_samples_split': 2}`
		`Best Decision Tree Accuracy: 0.981259833367022`
		`Best XGBoost Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}`
		`Best XGBoost Accuracy: 0.9830165311147149`

KNN	`# Hyperparameter tuning for KNN`
	`param_grid_knn = {`
	`    'n_neighbors': [3, 5, 7],  # Reduced number of neighbors`
	`    'weights': ['uniform', 'distance']  # Explore different weighting schemes`
	`}`
	
	`knn_grid = GridSearchCV(estimator=KNeighborsClassifier(), param_grid=param_grid_knn, cv=3, scoring='accuracy', n_jobs=-1)`
	`knn_grid.fit(x_train_scaled, y_train)`
	
	`print("Best KNN Parameters:", knn_grid.best_params_)`
	`print("Best KNN Accuracy:", knn_grid.best_score_)`
		`Best Random Forest Parameters: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 100}`
		`Best Random Forest Accuracy: 0.9795041642814564`
		`Best KNN Parameters: {'n_neighbors': 7, 'weights': 'distance'}`
		`Best KNN Accuracy: 0.918009713656467`
		`Best Decision Tree Parameters: {'max_depth': None, 'min_samples_split': 2}`
		`Best Decision Tree Accuracy: 0.981259833367022`
		`Best XGBoost Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}`
		`Best XGBoost Accuracy: 0.9830165311147149`

XG Boost	`# Hyperparameter tuning for XGBoost`
	`param_grid_xgb = {`
	`    'n_estimators': [50, 100],  # Reduced number of estimators`
	`    'max_depth': [3, 5],  # Limit max depth`
	`    'learning_rate': [0.1, 0.01]  # Explore different learning rates`
	`}`
	
	`xgb_grid = GridSearchCV(estimator=xgb.XGBClassifier(), param_grid=param_grid_xgb, cv=3, scoring='accuracy', n_jobs=-1)`
	`xgb_grid.fit(x_train_scaled, y_train)`
	
	`print("Best XGBoost Parameters:", xgb_grid.best_params_)`
	`print("Best XGBoost Accuracy:", xgb_grid.best_score_)`
		`Best Random Forest Parameters: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 100}`
		`Best Random Forest Accuracy: 0.9795041642814564`
		`Best KNN Parameters: {'n_neighbors': 7, 'weights': 'distance'}`
		`Best KNN Accuracy: 0.918009713656467`
		`Best Decision Tree Parameters: {'max_depth': None, 'min_samples_split': 2}`
		`Best Decision Tree Accuracy: 0.981259833367022`
		`Best XGBoost Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}`
		`Best XGBoost Accuracy: 0.9830165311147149`

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric

Decision Tree	`Best Decision Tree Confusion Matrix:`
	`[[310   6]`
	` [  8 530]]`
	`Best Decision Tree Classification Report:`
	`              precision    recall  f1-score   support`
	
	`           0       0.97      0.98      0.98       316`
	`           1       0.99      0.99      0.99       538`
	
	`    accuracy                           0.98       854`
	`   macro avg       0.98      0.98      0.98       854`
	`weighted avg       0.98      0.98      0.98       854`

Random Forest	Best Random Forest Confusion Matrix: [[308 8] [14 524]] Best Random Forest Classification Report: precision recall f1-score support 0 0.96 0.97 0.97 316 1 0.98 0.97 0.98 538 accuracy 0.97 854 macro avg 0.97 0.97 0.97 854 weighted avg 0.97 0.97 0.97 854
KNN	Best KNN Confusion Matrix: [[288 28] [43 495]] Best KNN Classification Report: precision recall f1-score support 0 0.87 0.91 0.89 316 1 0.95 0.92 0.93 538 accuracy 0.92 854 macro avg 0.91 0.92 0.91 854 weighted avg 0.92 0.92 0.92 854
XG Boost	Best XGBoost Confusion Matrix: [[311 5] [10 528]] Best XGBoost Classification Report: precision recall f1-score support 0 0.97 0.98 0.98 316 1 0.99 0.98 0.99 538 accuracy 0.98 854 macro avg 0.98 0.98 0.98 854 weighted avg 0.98 0.98 0.98 854

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Random forest	After evaluating the models based on several metrics such as **accuracy**, **precision**, **recall**, and **F1-score**, all models demonstrated good performance. However, **Random Forest (RF)** was selected as the final model due to its combination of accuracy, robustness, and interpretability. While XGBoost showed competitive performance, RF was easier to interpret and required less computational overhead for deployment, making it more suitable for this application. **Final Choice**: **Random Forest** was chosen as the model for predicting loan eligibility because of its high performance and the ability to generalize well to new, unseen data.