Model Development Phase Template

Date	15 March 2024
Team ID	SWTID1749835773
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

Decision Tree:

```
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_}")
model_opt = grid_search_dctree.best_estimator_
y_predict = model_opt.predict(x_test)
print (f'actual values:  \n{y_test.values}')
print (f"predicted values: \n{y_predict}")
```

Random Forest:

```
rf_model = RandomForestClassifier(random_state=42)

grid_search = GridSearchCV(
    estimator=rf_model,
    param_grid=param_grid,
    cv=5,                # 5-fold cross-validation
    n_jobs=-1,           # Use all CPU cores
    scoring='accuracy',  # Or 'roc_auc' for ROC-AUC
    verbose=2
)

grid_search.fit(x_train, y_train)

print(f"Best hyperparameters found by Grid Search: {grid_search.best_params_}")
best_rf_model = grid_search.best_estimator_


# Make predictions on the test set
y_pred = best_rf_model.predict(x_test)
```

KNN:

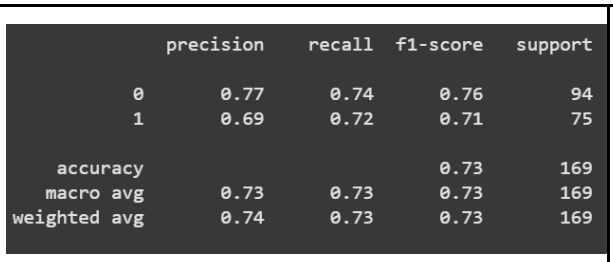
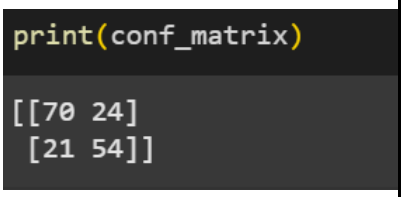
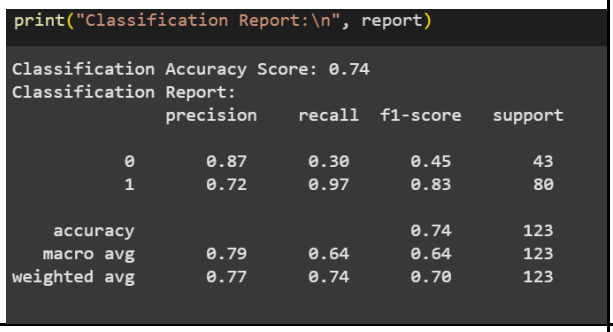
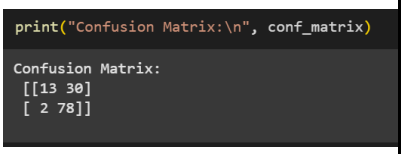
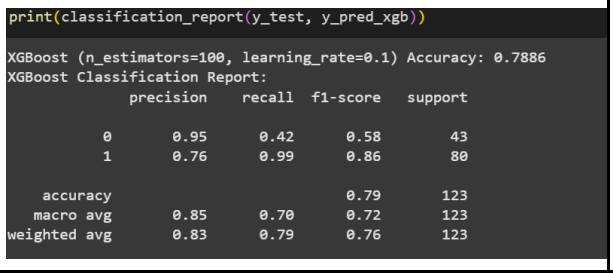
```
model_knn = KNeighborsClassifier()
grid_search = GridSearchCV(model_knn, param_grid, cv=5, scoring='accuracy', n_jobs=-1, verbose=1)
grid_search.fit(X_train_scaled, y_train)

print(f"Best hyperparameters found by Grid Search: {grid_search.best_params_}")

model_knn_best= grid_search.best_estimator_
model_knn_best.fit(X_train_scaled, y_train)
y_pred = model_knn_best.predict(X_test_scaled) # Use scaled data for prediction
```

XGBoost:

```
model = XGBClassifier(random_state=42, eval_metric='logloss')

grid_search = GridSearchCV(
    estimator=model,
    param_grid=param_grid,
    scoring='roc_auc',    # Or 'accuracy', 'f1', etc.
    cv=5,                 # 5-fold cross-validation
    n_jobs=-1,            # Use all CPU cores
    verbose=1
)

grid_search.fit(x_train, y_train)
print("Best score:", grid_search.best_score_)
print("Best parameters:", grid_search.best_params_)
best_model = grid_search.best_estimator_

y_pred_xgb = best_model.predict(x_test)
```

Model Validation and Evaluation Report:

Model	Classification Report	F1 Score	Confusion Matrix
Random Forest	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	79%	print(confusion_matrix(y_test,y_pred)) [[18 25] [1 79]]

Decision Tree		73%	
KNN		74%	
XGBoost		79%