

Model Development Phase Template

Date	20 July 2025
Team ID	SWUID20250184320
Project Title	Online Payment Fraud Detection
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:

Paste the screenshot of the model training code

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Random Forest Classifier	<pre>from sklearn.ensemble import RandomForestClassifier from sklearn.metrics import classification_report # Train Model 1 (Random Forest) model1 = RandomForestClassifier(random_state=42) model1.fit(X_train, y_train) y_pred1 = model1.predict(X_test) # Classification Report print("Classification Report (Model 1):") print(classification_report(y_test, y_pred1))</pre>	99.99%	<pre>from sklearn.metrics import confusion_matrix import seaborn as sns import matplotlib.pyplot as plt # Confusion Matrix for Model 1 conf_matrix1 = confusion_matrix(y_test, y_pred1) plt.figure(figsize=(8,4)) sns.heatmap(conf_matrix1, annot=True, fmt='d', cmap='Blues') plt.title('Confusion Matrix - Model 1') plt.xlabel('Predicted') plt.ylabel('Actual') plt.tight_layout() plt.savefig('confusion_matrix_model1.png') # Save as image plt.show()</pre>

Logistic Regression	<pre> from sklearn.linear_model import LogisticRegression from sklearn.metrics import classification_report # Train Model 2 (Logistic Regression) model2 = LogisticRegression(max_iter=1000) model2.fit(x_train, y_train) y_pred2 = model2.predict(x_test) # Classification Report print('Classification Report (Model 2):') print(classification_report(y_test, y_pred2)) </pre>	100%	<pre> conf_matrix2 = confusion_matrix(y_test, y_pred2) plt.figure(figsize=(6,4)) sns.heatmap(conf_matrix2, annot=True, fmt='d', cmap='seismic') plt.title('Confusion Matrix - Model 2') plt.xlabel('Predicted') plt.ylabel('Actual') plt.tight_layout() plt.savefig('confusion_matrix_model2.png') # Save as image plt.show() </pre>
Extra Trees Classifier	<pre> # Extra Trees Classifier from sklearn.ensemble import ExtraTreesClassifier et = ExtraTreesClassifier(random_state=0) et.fit(x_train, y_train) y_pred_et = et.predict(x_test) # Accuracy print('Accuracy (Extra Trees):', accuracy_score(y_test, y_pred_et)) # Classification Report print('Classification Report (Extra Trees):', classification_report(y_test, y_pred_et)) </pre>	100%	<pre> conf_matrix_et = confusion_matrix(y_test, y_pred_et) plt.figure(figsize=(6,4)) sns.heatmap(conf_matrix_et, annot=True, fmt='d', cmap='seismic', sticklabels=False, cbar=False) plt.title('Confusion Matrix - Extra Trees') plt.xlabel('Predicted') plt.ylabel('Actual') plt.tight_layout() plt.savefig('conf_matrix_et.png') plt.show() </pre>
Support Vector Machine Classifier	<pre> # Support Vector Machine from sklearn.svm import SVC svm = SVC(random_state=0) svm.fit(x_train, y_train) y_pred_svm = svm.predict(x_test) # Accuracy print('Accuracy (SVM):', accuracy_score(y_test, y_pred_svm)) # Classification Report print('Classification Report (SVM):', classification_report(y_test, y_pred_svm)) </pre>	99.59%	<pre> conf_matrix_svm = confusion_matrix(y_test, y_pred_svm) plt.figure(figsize=(6,4)) sns.heatmap(conf_matrix_svm, annot=True, fmt='d', cmap='seismic', sticklabels=False, cbar=False) plt.title('Confusion Matrix - SVM') plt.xlabel('Predicted') plt.ylabel('Actual') plt.tight_layout() plt.savefig('conf_matrix_svm.png') plt.show() </pre>
XG Boost Classifier	<pre> # XGBoost Classifier from xgboost import XGBClassifier xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=0) xgb.fit(x_train, y_train) y_pred_xgb = xgb.predict(x_test) # Accuracy print('Accuracy (XGBoost):', accuracy_score(y_test, y_pred_xgb)) # Classification Report print('Classification Report (XGBoost):', classification_report(y_test, y_pred_xgb)) </pre>	100%	<pre> conf_matrix_xgb = confusion_matrix(y_test, y_pred_xgb) plt.figure(figsize=(6,4)) sns.heatmap(conf_matrix_xgb, annot=True, fmt='d', cmap='seismic', sticklabels=False, cbar=False) plt.title('Confusion Matrix - XGBoost') plt.xlabel('Predicted') plt.ylabel('Actual') plt.tight_layout() plt.savefig('conf_matrix_xgb.png') plt.show() </pre>