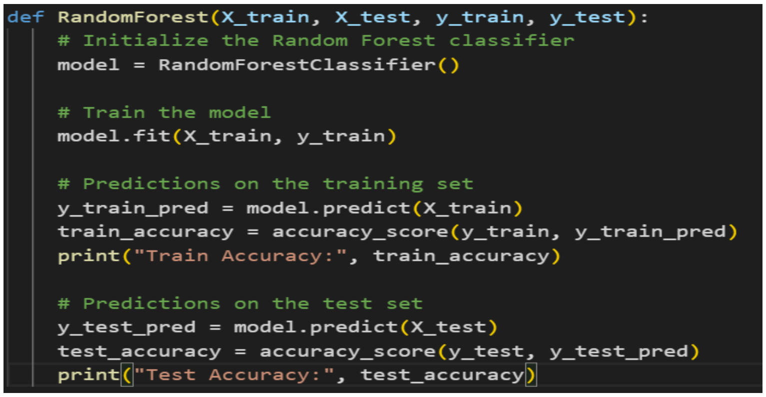
Model Development Phase Template

Date	30 April 2024
Team ID	Team-738315
Project Title	Online Payment Fraud Detection using Machine Learning
Maximum Marks	4 Marks

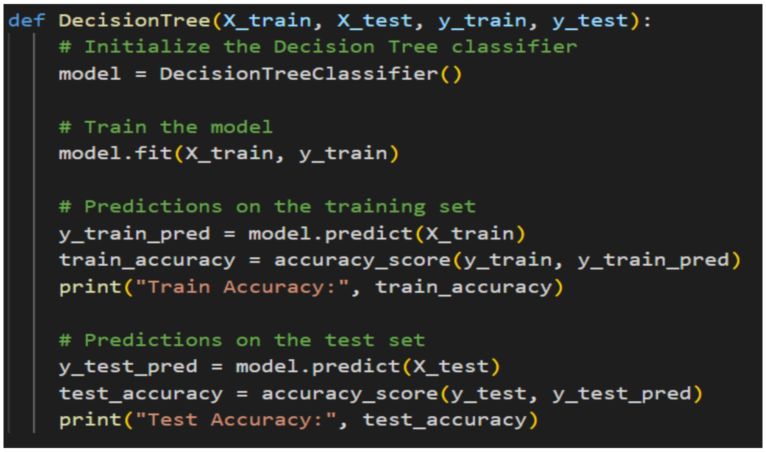
Initial Model Training Code, Model Validation and Evaluation Report

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.

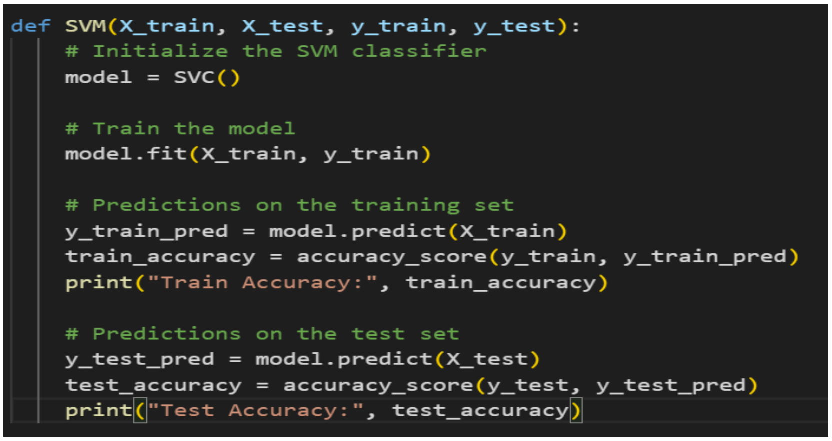
Initial Model Training Code:



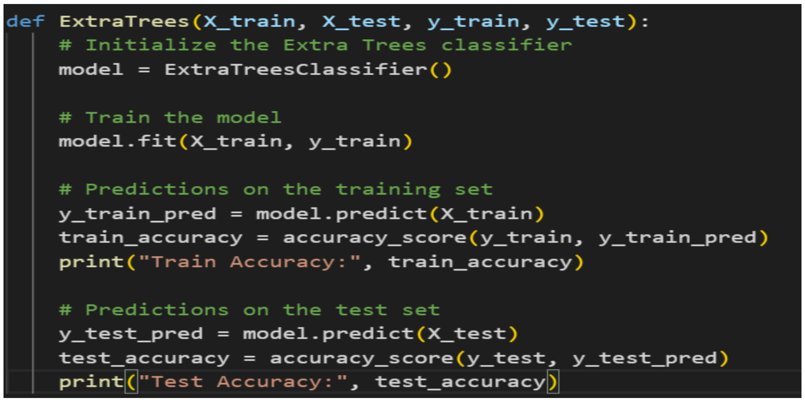
```python
def RandomForest(X_train, X_test, y_train, y_test):
    # Initialize the Random Forest classifier
    model = RandomForestClassifier()

    # Train the model
    model.fit(X_train, y_train)

    # Predictions on the training set
    y_train_pred = model.predict(X_train)
    train_accuracy = accuracy_score(y_train, y_train_pred)
    print("Train Accuracy:", train_accuracy)

    # Predictions on the test set
    y_test_pred = model.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    print("Test Accuracy:", test_accuracy)
```



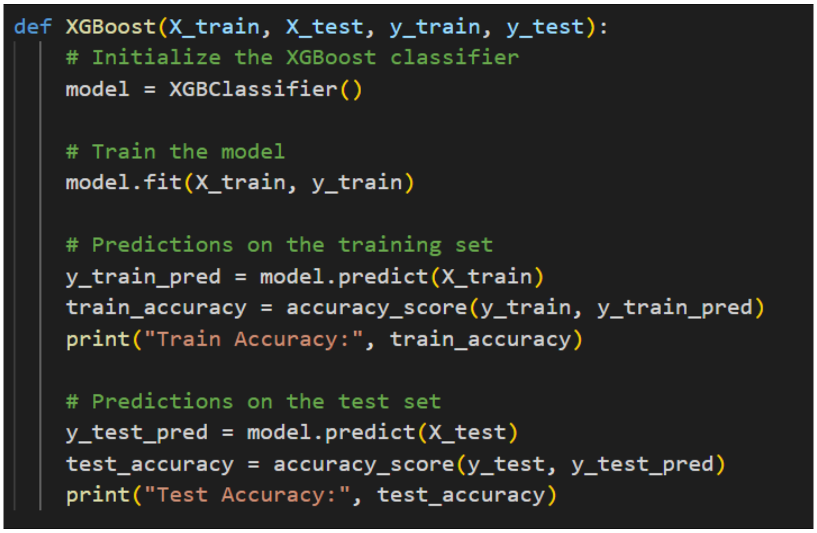
```python
def DecisionTree(X_train, X_test, y_train, y_test):
    # Initialize the Decision Tree classifier
    model = DecisionTreeClassifier()

    # Train the model
    model.fit(X_train, y_train)

    # Predictions on the training set
    y_train_pred = model.predict(X_train)
    train_accuracy = accuracy_score(y_train, y_train_pred)
    print("Train Accuracy:", train_accuracy)

    # Predictions on the test set
    y_test_pred = model.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    print("Test Accuracy:", test_accuracy)
```



```
def SVM(X_train, X_test, y_train, y_test):
    # Initialize the SVM classifier
    model = SVC()

    # Train the model
    model.fit(X_train, y_train)

    # Predictions on the training set
    y_train_pred = model.predict(X_train)
    train_accuracy = accuracy_score(y_train, y_train_pred)
    print("Train Accuracy:", train_accuracy)

    # Predictions on the test set
    y_test_pred = model.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    print("Test Accuracy:", test_accuracy)
```



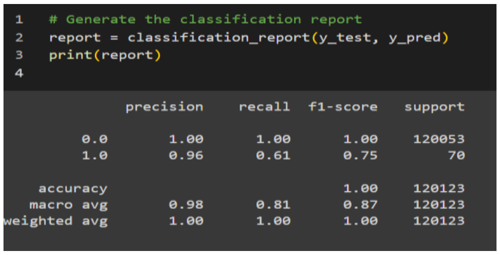
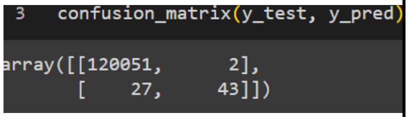
```
def ExtraTrees(X_train, X_test, y_train, y_test):
    # Initialize the Extra Trees classifier
    model = ExtraTreesClassifier()

    # Train the model
    model.fit(X_train, y_train)

    # Predictions on the training set
    y_train_pred = model.predict(X_train)
    train_accuracy = accuracy_score(y_train, y_train_pred)
    print("Train Accuracy:", train_accuracy)

    # Predictions on the test set
    y_test_pred = model.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    print("Test Accuracy:", test_accuracy)
```

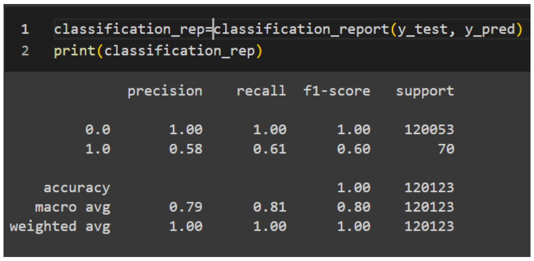
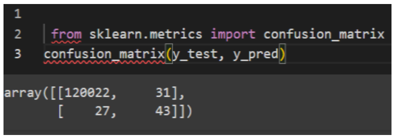
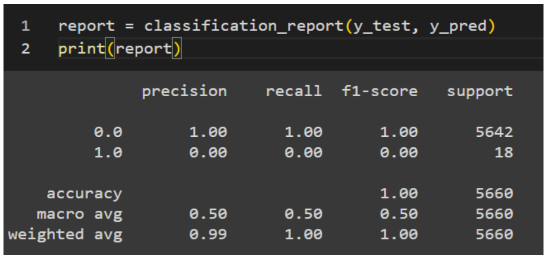
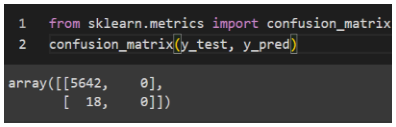
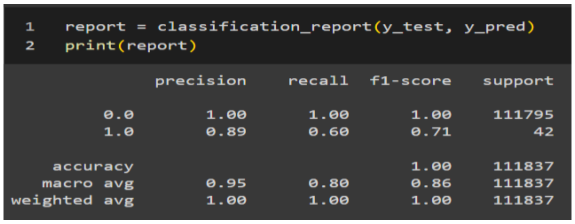
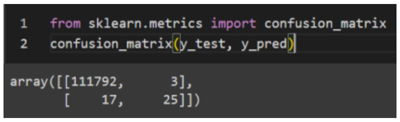


```
def XGBoost(X_train, X_test, y_train, y_test):
    # Initialize the XGBoost classifier
    model = XGBClassifier()

    # Train the model
    model.fit(X_train, y_train)

    # Predictions on the training set
    y_train_pred = model.predict(X_train)
    train_accuracy = accuracy_score(y_train, y_train_pred)
    print("Train Accuracy:", train_accuracy)

    # Predictions on the test set
    y_test_pred = model.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    print("Test Accuracy:", test_accuracy)
```

Model Validation and Evaluation Report:

Model	Classification Report	F1 Score	Confusion Matrix

Random Forest	
	# Generate the classification report
	report = classification_report(y_test, y_pred)
	print(report)
	precision recall f1-score support
	0.0 1.00 1.00 1.00 120053
	1.0 0.96 0.61 0.75 70
	accuracy 1.00 120123
	macro avg 0.98 0.81 0.87 120123
	weighted avg 1.00 1.00 1.00 120123
		66%	
			confusion_matrix(y_test, y_pred)
			array([[120051, 2],
			[27, 43]])

Decision Tree	 classification_rep=classification_report(y_test, y_pred) print(classification_rep) precision recall f1-score support 0.0 1.00 1.00 1.00 120053 1.0 0.58 0.61 0.60 70 accuracy 1.00 120123 macro avg 0.79 0.81 0.80 120123 weighted avg 1.00 1.00 1.00 120123	99.8%	 from sklearn.metrics import confusion_matrix confusion_matrix(y_test, y_pred) array([[120022, 31], [27, 43]])
SupportVectorMachine	 report = classification_report(y_test, y_pred) print(report) precision recall f1-score support 0.0 1.00 1.00 1.00 5642 1.0 0.00 0.00 0.00 18 accuracy 1.00 5660 macro avg 0.50 0.50 0.50 5660 weighted avg 0.99 1.00 1.00 5660	99.5%	 from sklearn.metrics import confusion_matrix confusion_matrix(y_test, y_pred) array([[5642, 0], [18, 0]])
XG Boosting	 report = classification_report(y_test, y_pred) print(report) precision recall f1-score support 0.0 1.00 1.00 1.00 111795 1.0 0.89 0.60 0.71 42 accuracy 1.00 111837 macro avg 0.95 0.80 0.86 111837 weighted avg 1.00 1.00 1.00 111837	99.8%	 from sklearn.metrics import confusion_matrix confusion_matrix(y_test, y_pred) array([[111792, 3], [17, 25]])