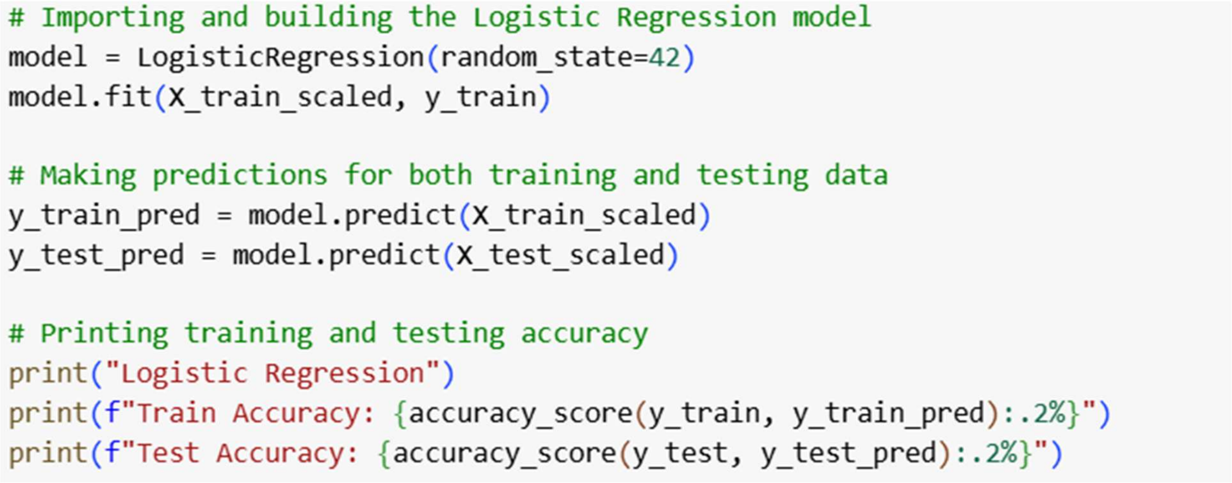
Model Development Phase Template

Date	20 June 2025
Project Title	Rising water: A Machine Learning Approach to Flood Prediction
Maximum Marks	4 Marks

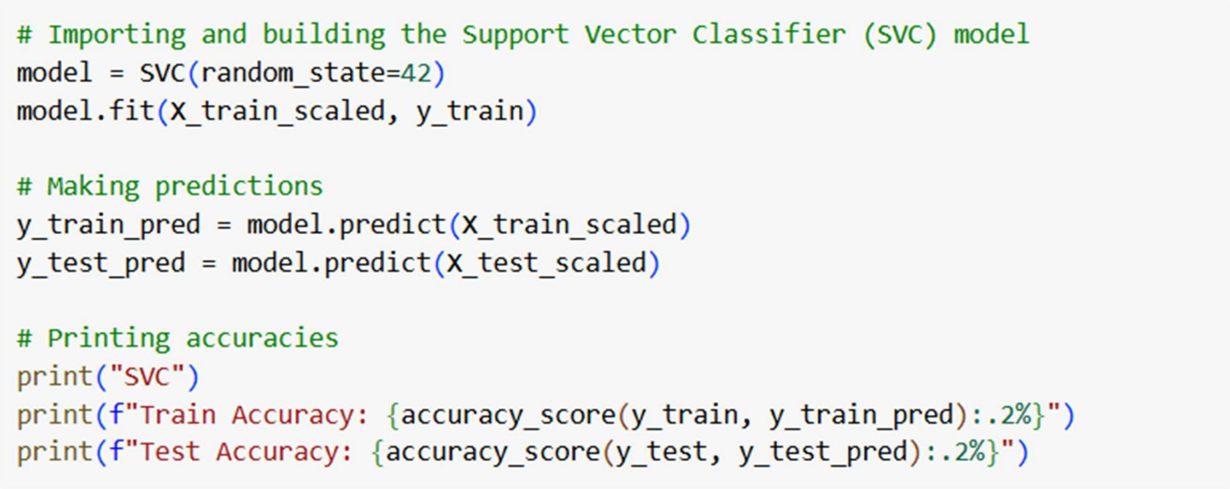
Initial Model Training Code, Model Validation and Evaluation Report

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.

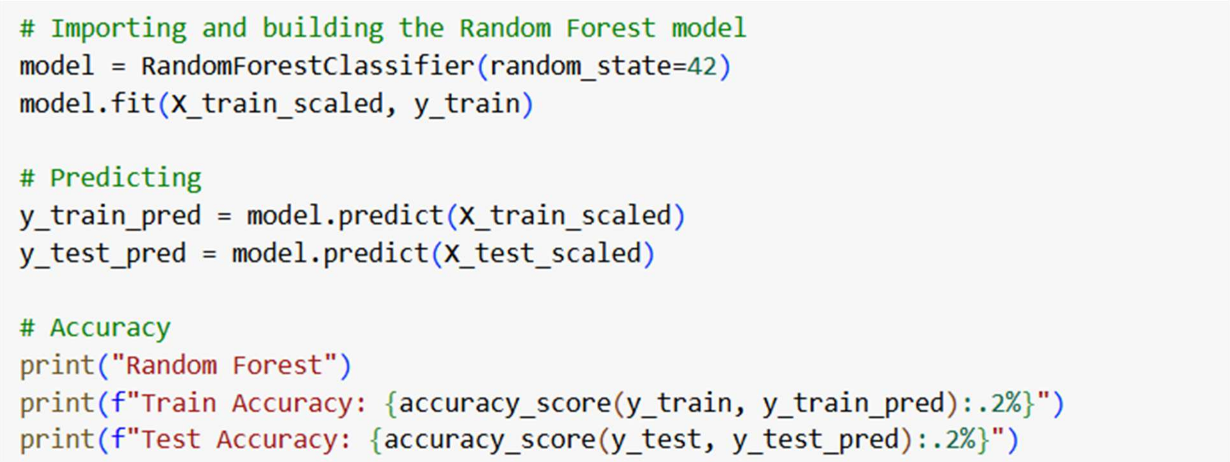
Initial Model Training Code:



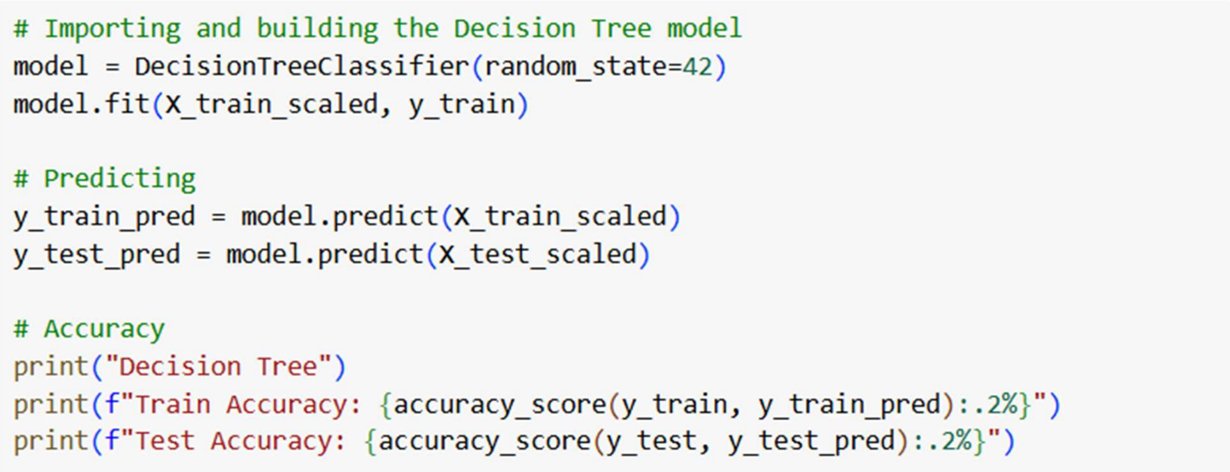
```
# Importing and building the Logistic Regression model
model = LogisticRegression(random_state=42)
model.fit(X_train_scaled, y_train)

# Making predictions for both training and testing data
y_train_pred = model.predict(X_train_scaled)
y_test_pred = model.predict(X_test_scaled)

# Printing training and testing accuracy
print("Logistic Regression")
print(f"Train Accuracy: {accuracy_score(y_train, y_train_pred):.2%}")
print(f"Test Accuracy: {accuracy_score(y_test, y_test_pred):.2%}")
```



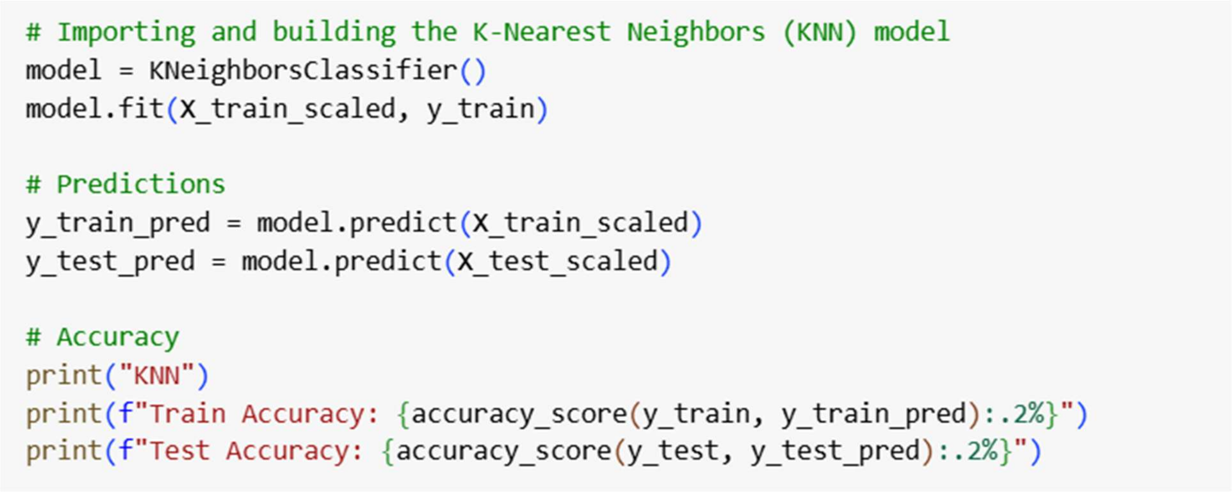
```
# Importing and building the Support Vector Classifier (SVC) model
model = SVC(random_state=42)
model.fit(X_train_scaled, y_train)

# Making predictions
y_train_pred = model.predict(X_train_scaled)
y_test_pred = model.predict(X_test_scaled)

# Printing accuracies
print("SVC")
print(f"Train Accuracy: {accuracy_score(y_train, y_train_pred):.2%}")
print(f"Test Accuracy: {accuracy_score(y_test, y_test_pred):.2%}")
```



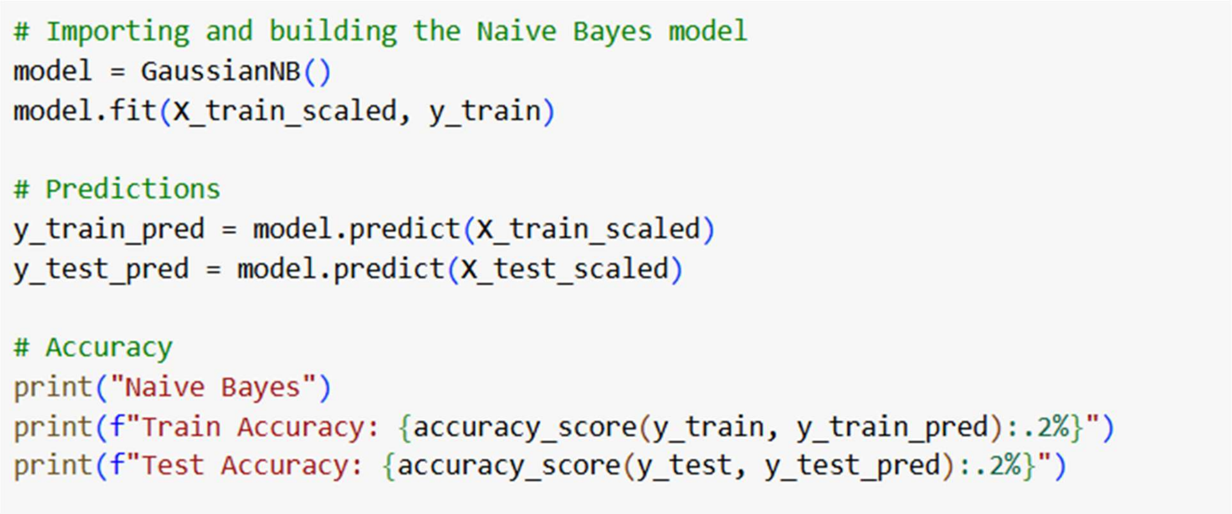
```
# Importing and building the Random Forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train_scaled, y_train)

# Predicting
y_train_pred = model.predict(X_train_scaled)
y_test_pred = model.predict(X_test_scaled)

# Accuracy
print("Random Forest")
print(f"Train Accuracy: {accuracy_score(y_train, y_train_pred):.2%}")
print(f"Test Accuracy: {accuracy_score(y_test, y_test_pred):.2%}")
```



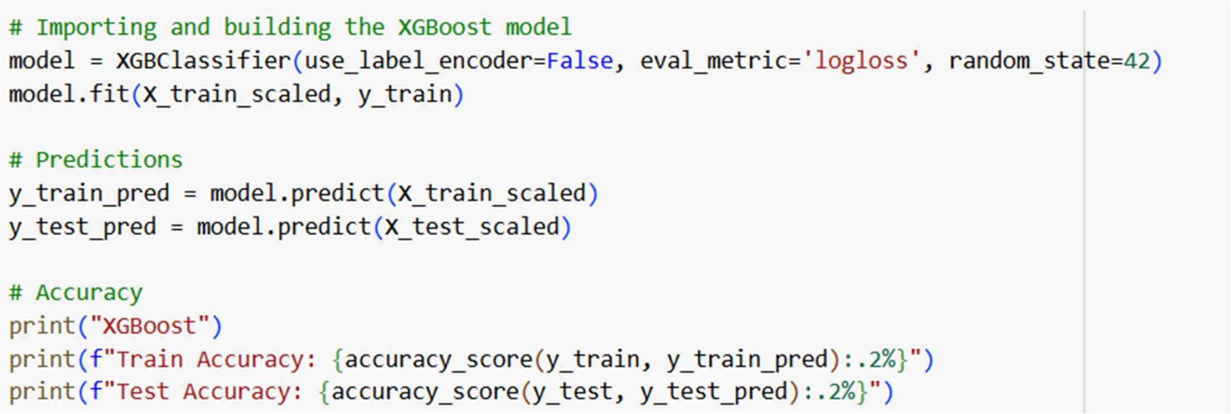
```
# Importing and building the Decision Tree model
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train_scaled, y_train)

# Predicting
y_train_pred = model.predict(X_train_scaled)
y_test_pred = model.predict(X_test_scaled)

# Accuracy
print("Decision Tree")
print(f"Train Accuracy: {accuracy_score(y_train, y_train_pred):.2%}")
print(f"Test Accuracy: {accuracy_score(y_test, y_test_pred):.2%}")
```



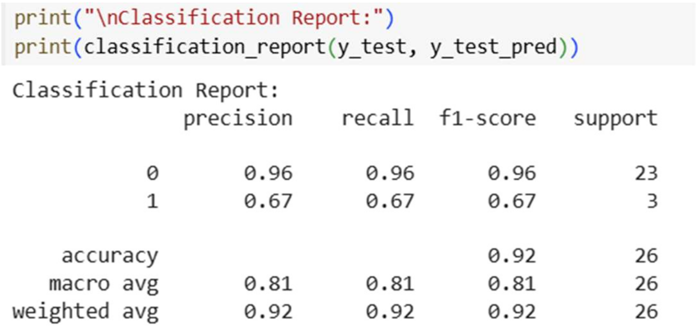
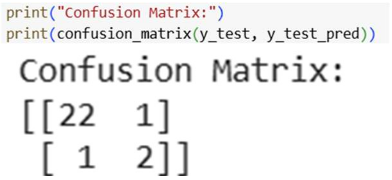
```
# Importing and building the K-Nearest Neighbors (KNN) model
model = KNeighborsClassifier()
model.fit(X_train_scaled, y_train)

# Predictions
y_train_pred = model.predict(X_train_scaled)
y_test_pred = model.predict(X_test_scaled)

# Accuracy
print("KNN")
print(f"Train Accuracy: {accuracy_score(y_train, y_train_pred):.2%}")
print(f"Test Accuracy: {accuracy_score(y_test, y_test_pred):.2%}")
```



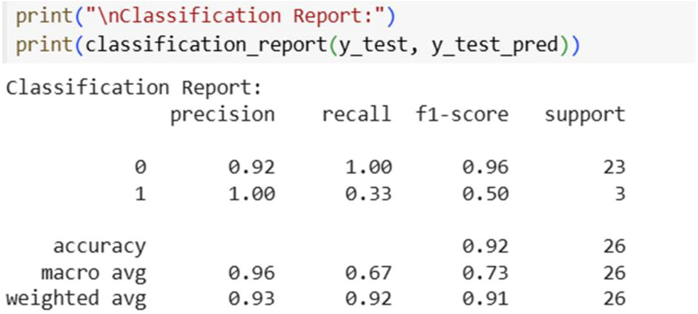
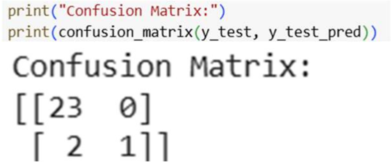
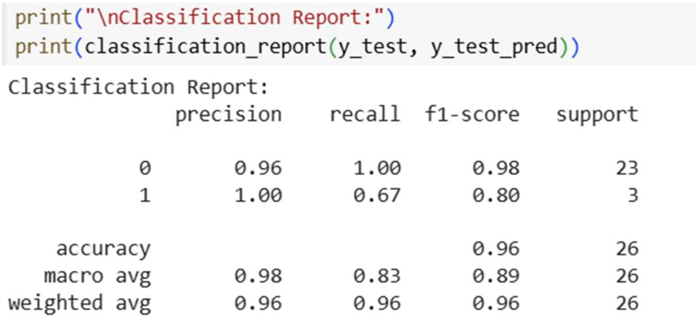
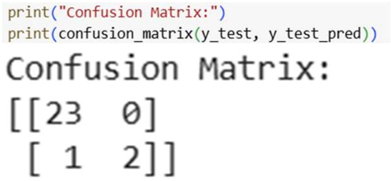
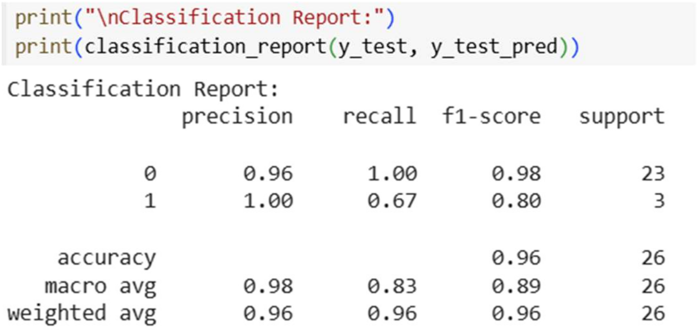
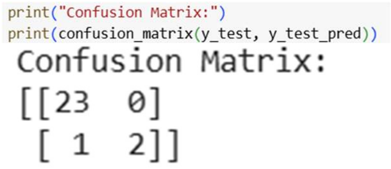
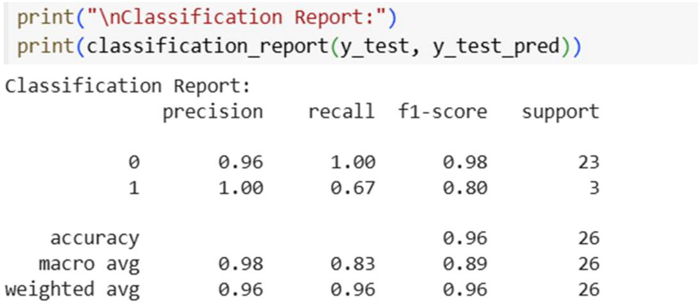
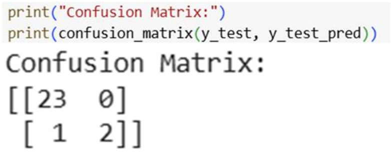
```
# Importing and building the Naive Bayes model
model = GaussianNB()
model.fit(X_train_scaled, y_train)

# Predictions
y_train_pred = model.predict(X_train_scaled)
y_test_pred = model.predict(X_test_scaled)

# Accuracy
print("Naive Bayes")
print(f"Train Accuracy: {accuracy_score(y_train, y_train_pred):.2%}")
print(f"Test Accuracy: {accuracy_score(y_test, y_test_pred):.2%}")
```

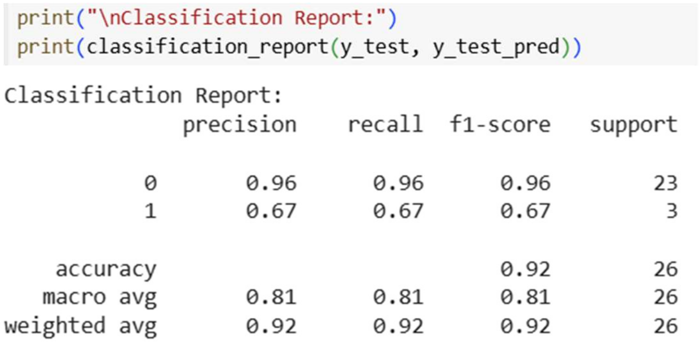
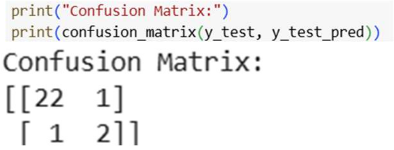
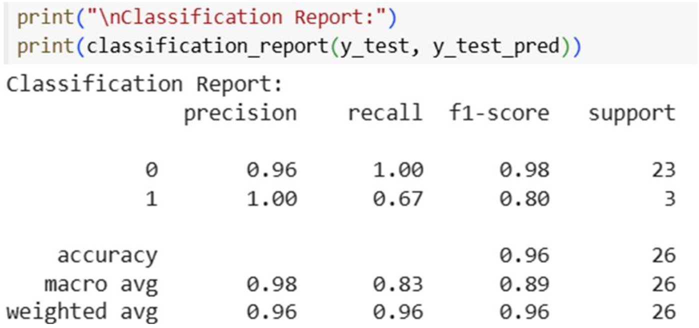
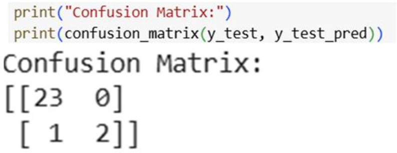


```
# Importing and building the XGBoost model
model = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
model.fit(X_train_scaled, y_train)

# Predictions
y_train_pred = model.predict(X_train_scaled)
y_test_pred = model.predict(X_test_scaled)

# Accuracy
print("XGBoost")
print(f"Train Accuracy: {accuracy_score(y_train, y_train_pred):.2%}")
print(f"Test Accuracy: {accuracy_score(y_test, y_test_pred):.2%}")
```

Model	Classification Report	F1 Score	Confusion Matrix
Logistic Regression	 print("\nClassification Report:") print(classification_report(y_test, y_test_pred)) Classification Report: precision recall f1-score support 0 0.96 0.96 0.96 23 1 0.67 0.67 0.67 3 accuracy 0.92 26 macro avg 0.81 0.81 0.81 26 weighted avg 0.92 0.92 0.92 26	67%	 print("Confusion Matrix:") print(confusion_matrix(y_test, y_test_pred)) Confusion Matrix: [[22 1] [1 2]]

Model Validation and Evaluation Report:

SVC	 print("\nClassification Report:") print(classification_report(y_test, y_test_pred)) Classification Report: precision recall f1-score support 0 0.92 1.00 0.96 23 1 1.00 0.33 0.50 3 accuracy 0.92 26 macro avg 0.96 0.67 0.73 26 weighted avg 0.93 0.92 0.91 26	50%	 print("Confusion Matrix:") print(confusion_matrix(y_test, y_test_pred)) Confusion Matrix: [[23 0] [2 1]]
Decision Tree	 print("\nClassification Report:") print(classification_report(y_test, y_test_pred)) Classification Report: precision recall f1-score support 0 0.96 1.00 0.98 23 1 1.00 0.67 0.80 3 accuracy 0.96 26 macro avg 0.98 0.83 0.89 26 weighted avg 0.96 0.96 0.96 26	80%	 print("Confusion Matrix:") print(confusion_matrix(y_test, y_test_pred)) Confusion Matrix: [[23 0] [1 2]]
Random Forest	 print("\nClassification Report:") print(classification_report(y_test, y_test_pred)) Classification Report: precision recall f1-score support 0 0.96 1.00 0.98 23 1 1.00 0.67 0.80 3 accuracy 0.96 26 macro avg 0.98 0.83 0.89 26 weighted avg 0.96 0.96 0.96 26	80%	 print("Confusion Matrix:") print(confusion_matrix(y_test, y_test_pred)) Confusion Matrix: [[23 0] [1 2]]
K-Nearest Neighbors	 print("\nClassification Report:") print(classification_report(y_test, y_test_pred)) Classification Report: precision recall f1-score support 0 0.96 1.00 0.98 23 1 1.00 0.67 0.80 3 accuracy 0.96 26 macro avg 0.98 0.83 0.89 26 weighted avg 0.96 0.96 0.96 26	80%	 print("Confusion Matrix:") print(confusion_matrix(y_test, y_test_pred)) Confusion Matrix: [[23 0] [1 2]]

Naive Bayes	 print("\nClassification Report:") print(classification_report(y_test, y_test_pred)) Classification Report: precision recall f1-score support 0 0.96 0.96 0.96 23 1 0.67 0.67 0.67 3 accuracy 0.92 26 macro avg 0.81 0.81 0.81 26 weighted avg 0.92 0.92 0.92 26	67%	 print("Confusion Matrix:") print(confusion_matrix(y_test, y_test_pred)) Confusion Matrix: [[22 1] [1 2]]
XGBoost	 print("\nClassification Report:") print(classification_report(y_test, y_test_pred)) Classification Report: precision recall f1-score support 0 0.96 1.00 0.98 23 1 1.00 0.67 0.80 3 accuracy 0.96 26 macro avg 0.98 0.83 0.89 26 weighted avg 0.96 0.96 0.96 26	80%	 print("Confusion Matrix:") print(confusion_matrix(y_test, y_test_pred)) Confusion Matrix: [[23 0] [1 2]]