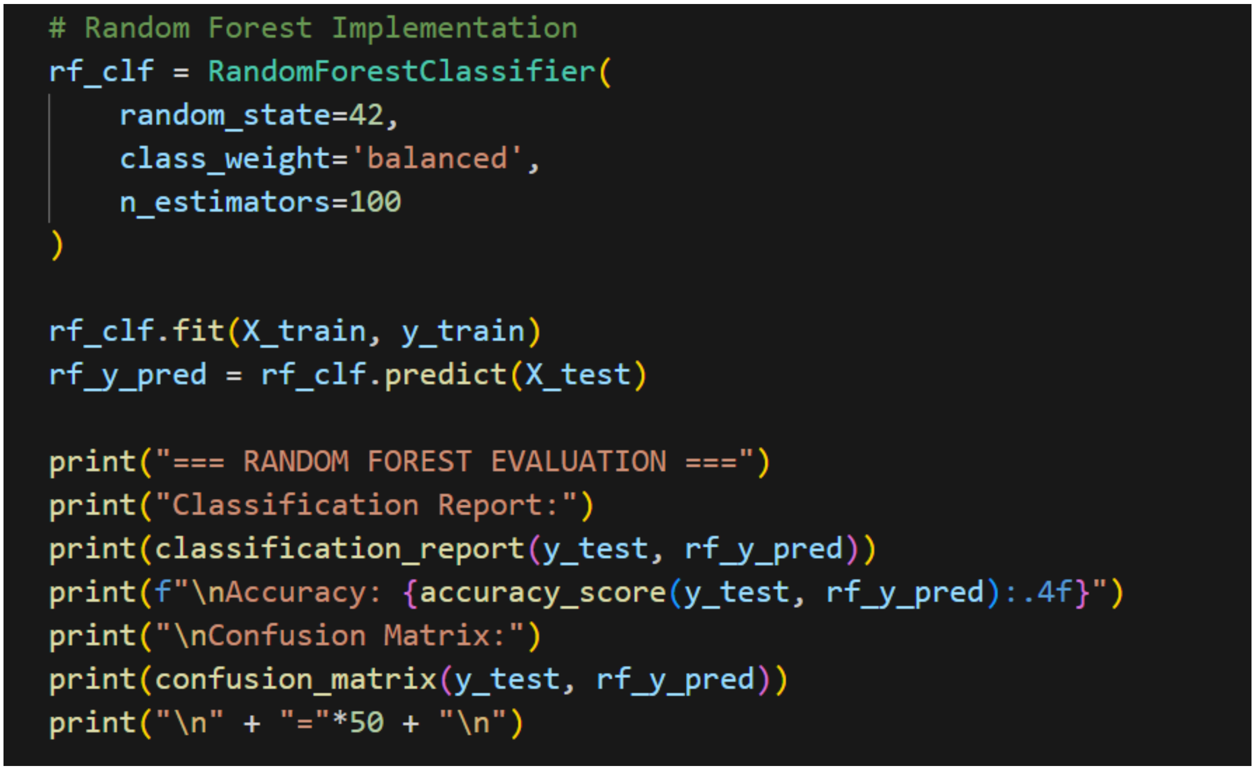
Model Development Phase

Date	19 June 2025
Team ID	SWTID1749710444
Project Title	Online Payment Fraud Detection using ML
Maximum Marks	4 Marks

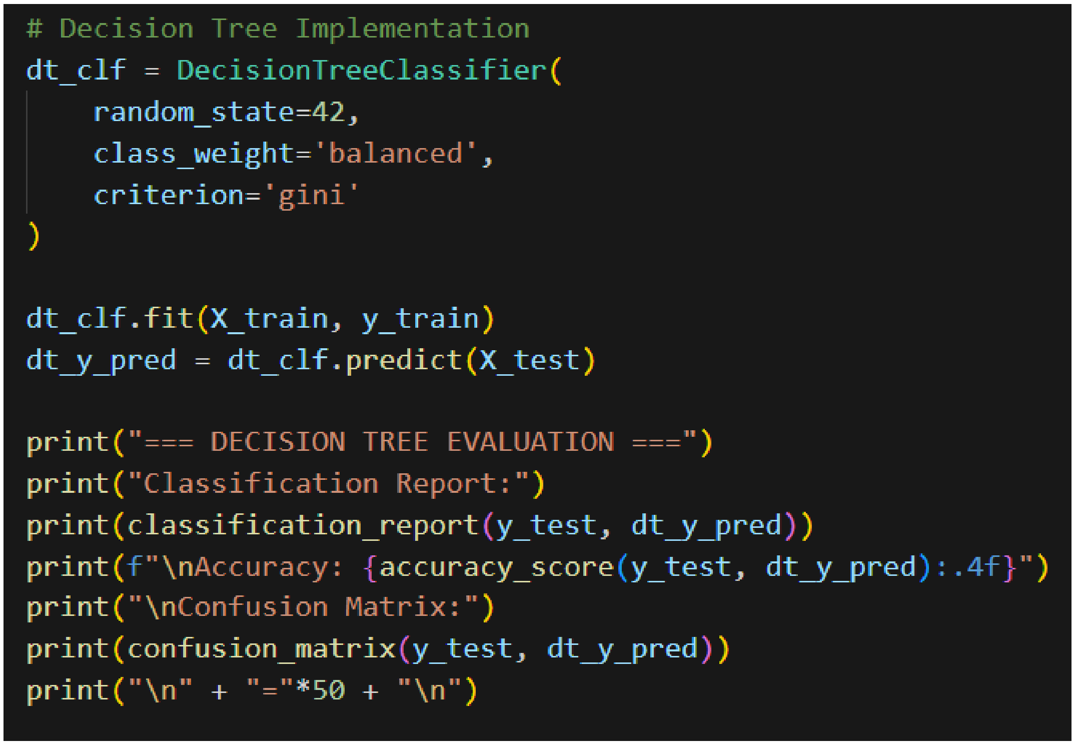
Initial Model Training Code, Model Validation and Evaluation Report

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.

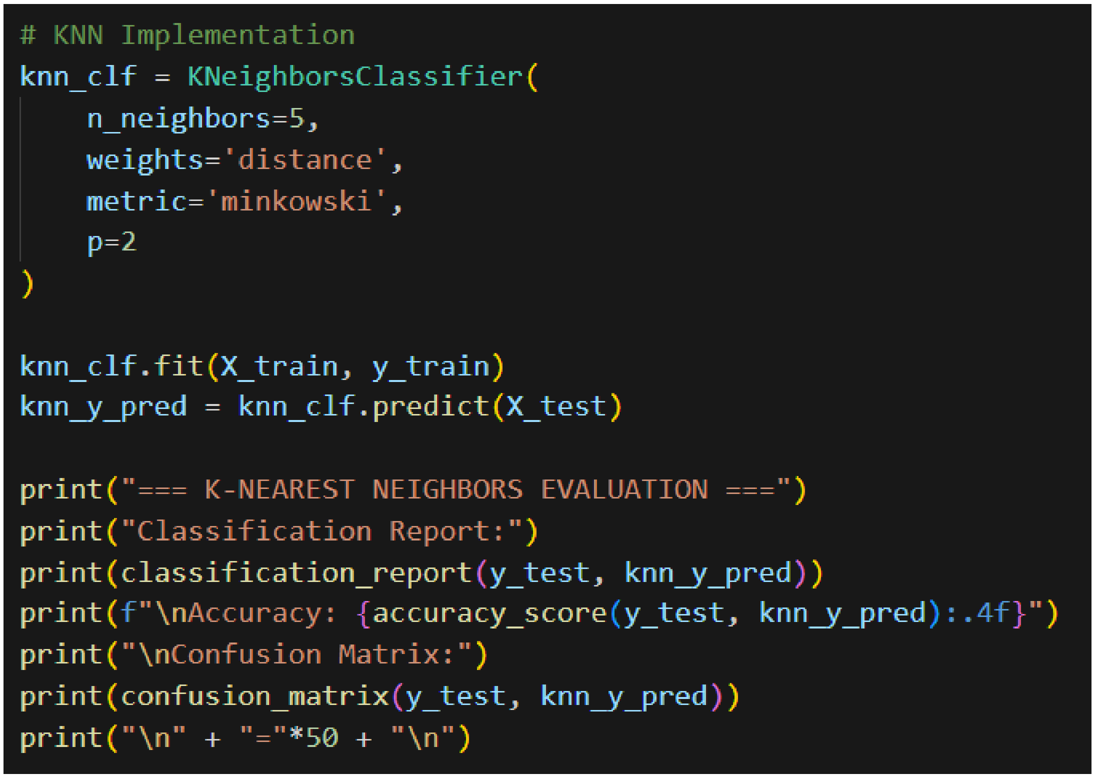
Initial Model Training Code:



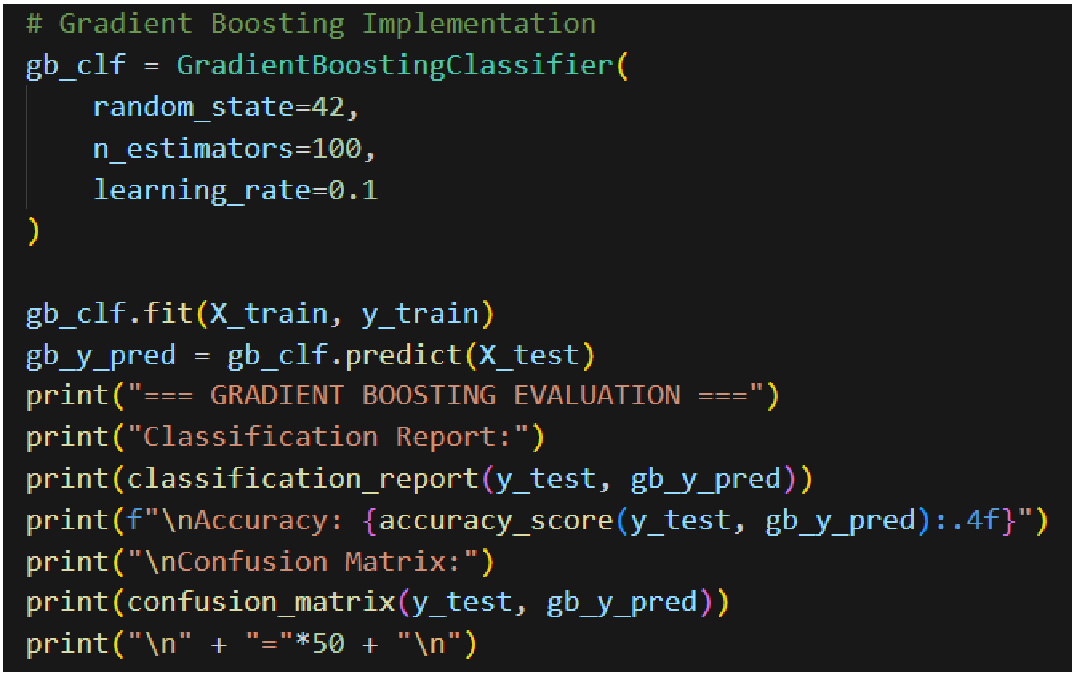
```python
# Random Forest Implementation
rf_clf = RandomForestClassifier(
    random_state=42,
    class_weight='balanced',
    n_estimators=100
)

rf_clf.fit(X_train, y_train)
rf_y_pred = rf_clf.predict(X_test)

print("=== RANDOM FOREST EVALUATION ===")
print("Classification Report:")
print(classification_report(y_test, rf_y_pred))
print(f"\nAccuracy: {accuracy_score(y_test, rf_y_pred):.4f}")
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, rf_y_pred))
print("\n" + "="*50 + "\n")
```



```python
# Decision Tree Implementation
dt_clf = DecisionTreeClassifier(
    random_state=42,
    class_weight='balanced',
    criterion='gini'
)

dt_clf.fit(X_train, y_train)
dt_y_pred = dt_clf.predict(X_test)

print("=== DECISION TREE EVALUATION ===")
print("Classification Report:")
print(classification_report(y_test, dt_y_pred))
print(f"\nAccuracy: {accuracy_score(y_test, dt_y_pred):.4f}")
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, dt_y_pred))
print("\n" + "="*50 + "\n")
```

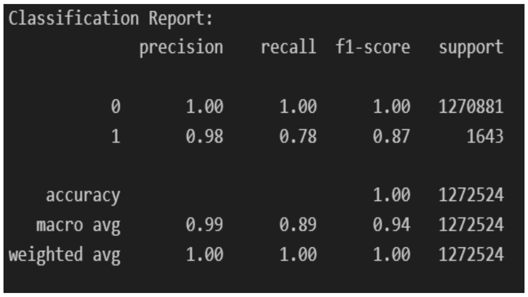


```python
# KNN Implementation
knn_clf = KNeighborsClassifier(
    n_neighbors=5,
    weights='distance',
    metric='minkowski',
    p=2
)

knn_clf.fit(X_train, y_train)
knn_y_pred = knn_clf.predict(X_test)

print("=== K-NEAREST NEIGHBORS EVALUATION ===")
print("Classification Report:")
print(classification_report(y_test, knn_y_pred))
print(f"\nAccuracy: {accuracy_score(y_test, knn_y_pred):.4f}")
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, knn_y_pred))
print("\n" + "="*50 + "\n")
```

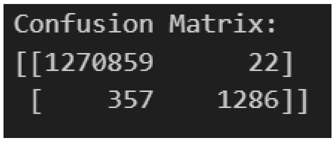
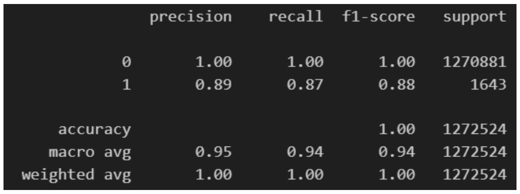


```
# Gradient Boosting Implementation
gb_clf = GradientBoostingClassifier(
    random_state=42,
    n_estimators=100,
    learning_rate=0.1
)

gb_clf.fit(X_train, y_train)
gb_y_pred = gb_clf.predict(X_test)
print("=== GRADIENT BOOSTING EVALUATION ===")
print("Classification Report:")
print(classification_report(y_test, gb_y_pred))
print(f"\nAccuracy: {accuracy_score(y_test, gb_y_pred):.4f}")
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, gb_y_pred))
print("\n" + "="*50 + "\n")
```

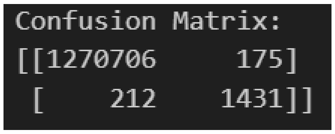
Model Validation and Evaluation Report:

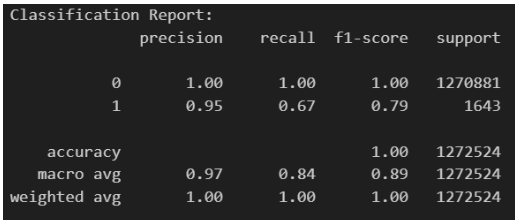
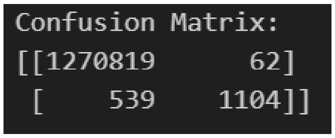
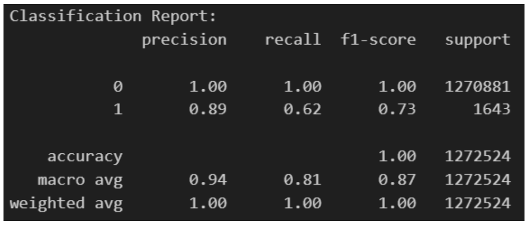
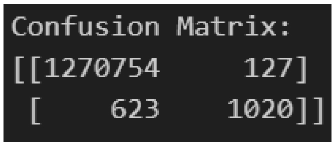
<table>
<tr><th>Model</th><th>Classification Report</th><th>Accuracy</th><th>Confusion Matrix</th></tr>
<tr><td>Random Forest</td><td>
<pre>
Classification Report:
              precision   recall f1-score   support

           0   1.00   1.00   1.00   1270881
           1   0.98   0.78   0.87   1643

    accuracy   1.00   1272524
   macro avg   0.99   0.89   0.94   1272524
weighted avg   1.00   1.00   1.00   1272524
</pre></td><td>0.9997</td><td>
<pre>
Confusion Matrix:
[[1270859   22]
 [357   1286]]
</pre></td></tr>
<tr><td>Decision Tree</td><td>
<pre>
              precision   recall f1-score   support

           0   1.00   1.00   1.00   1270881
           1   0.89   0.87   0.88   1643

    accuracy   1.00   1272524
   macro avg   0.95   0.94   0.94   1272524
weighted avg   1.00   1.00   1.00   1272524
</pre></td><td>0.9997</td><td>
<pre>
Confusion Matrix:
[[1270706   175]
 [212   1431]]
</pre></td></tr>
</table>

KNN	 Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 1270881 1 0.95 0.67 0.79 1643 accuracy 1.00 1272524 macro avg 0.97 0.84 0.89 1272524 weighted avg 1.00 1.00 1.00 1272524	0.9995	 Confusion Matrix: [[1270819 62] [539 1104]]
Gradient Boosting	 Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 1270881 1 0.89 0.62 0.73 1643 accuracy 1.00 1272524 macro avg 0.94 0.81 0.87 1272524 weighted avg 1.00 1.00 1.00 1272524	0.9994	 Confusion Matrix: [[1270754 127] [623 1020]]