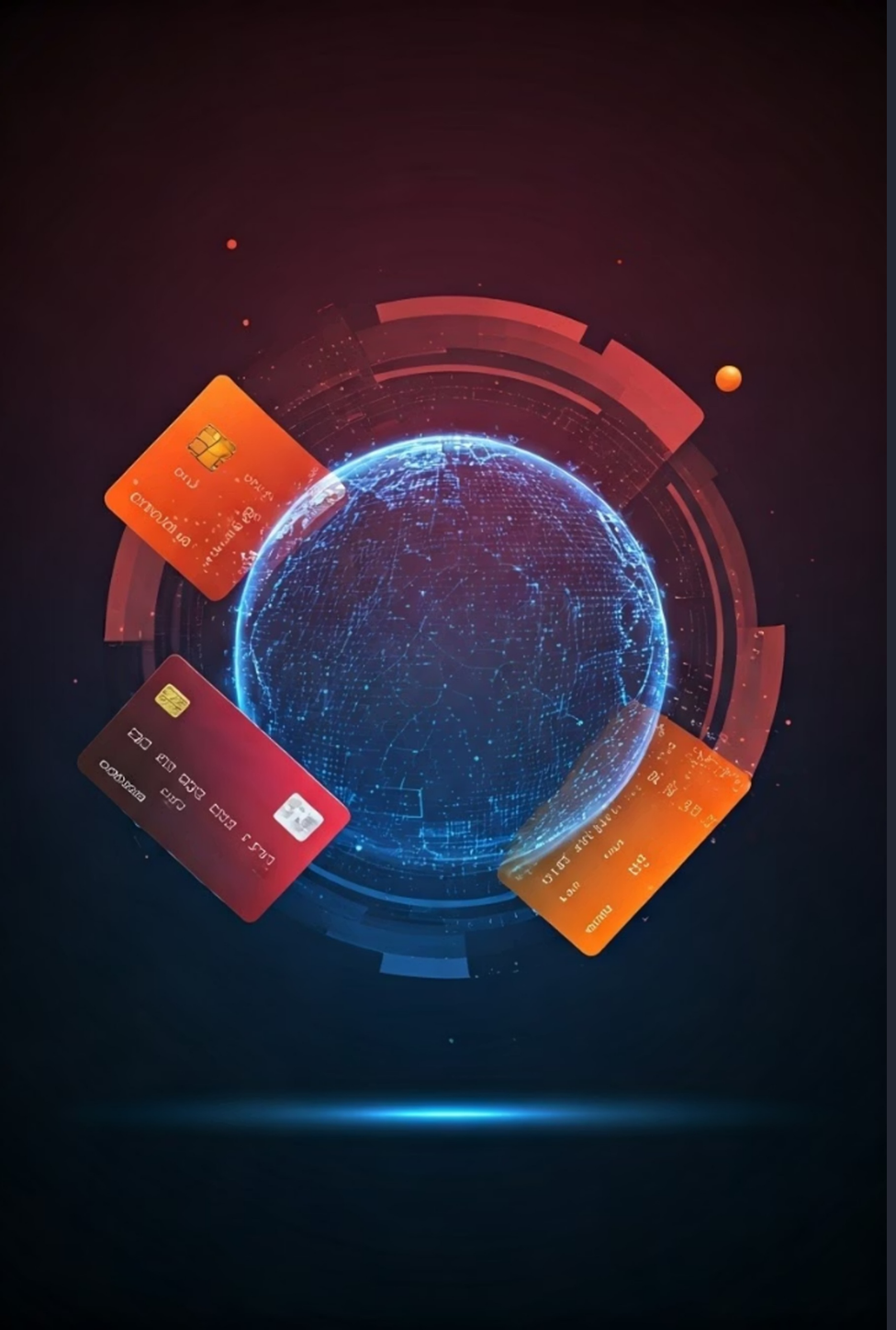


Credit Card Fraud Detection

This presentation explores machine learning approaches to detect credit card fraud, a critical financial security challenge costing billions annually.

Course: CS 513 – A (Knowledge Discovery and Data Mining)

Team: Final Project 10

Meet Our Team

Parth Gadekar

Ankit Dhandharia

Priti Nagdive

Jeel Patel

Introduction to Credit Card Fraud

Financial Impact

Credit card fraud costs billions of dollars annually.

Machine Learning Solution

ML can detect patterns that humans might miss.

Key Challenge

Extremely imbalanced data with very few fraud cases.

Dataset Overview

Transaction Volume

284,807 credit card transactions with 31 features.

Feature Types

Time, V1-V28 (PCA transformed), Amount, and Class.

Data Quality

No missing values in the dataset.

Class Imbalance Challenge

284,315

Normal Transactions

99.83% of all transactions

492

Fraudulent Transactions

Only 0.17% of all transactions

578:1

Imbalance Ratio

Creates significant modeling challenges

Data Visualization Insights

Amount Distribution

Fraudulent transactions generally involve smaller amounts. Fraudsters often test small amounts before larger transactions.

Time Distribution

Normal transactions show cyclical day/night patterns. Fraudulent transactions are more evenly distributed across time periods.

Data Preparation Techniques

Feature Scaling

Used RobustScaler for 'Amount' and 'Time' features to handle outliers.

Handling Class Imbalance

Applied SMOTE to create synthetic fraud examples.

Train-Test Split

Stratified 80/20 split to maintain class distribution.

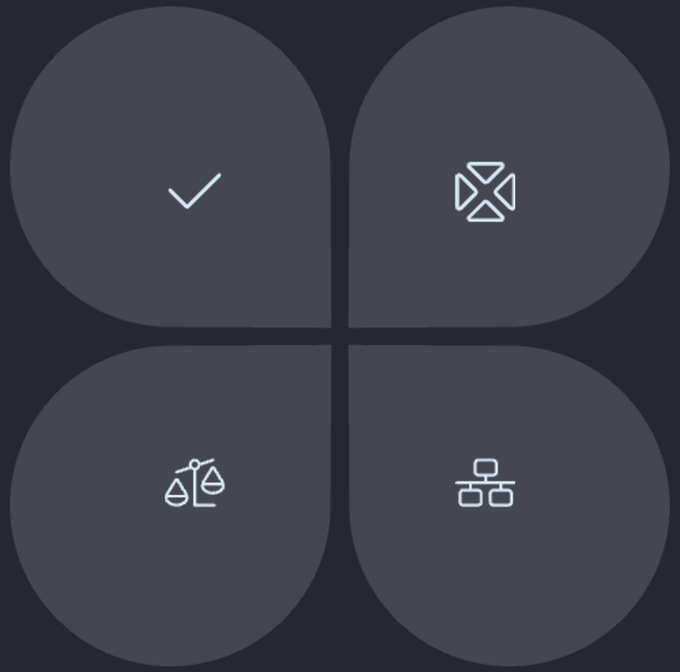
Model Evaluation Metrics

Accuracy

Overall correct predictions

F1 Score

Harmonic mean of precision and recall



Precision

Proportion of predicted frauds that are actually fraudulent

Recall

Proportion of actual frauds correctly identified

Logistic Regression

Overview

- Logistic Regression is a linear model used for binary classification tasks.
- It estimates the probability of a class using a sigmoid activation on a linear combination of input features.
- Works best when classes are linearly separable and data is scaled.
- Easy to implement, fast to train, and interpretable.
- Often serves as a baseline for comparing more complex models.

Application:

- RobustScaler was used to scale numerical features.
- SMOTE was applied to address class imbalance.
- Model trained on the SMOTE-balanced dataset.
- LogisticRegression parameters:
  - penalty = 'l2'
  - solver = 'liblinear'
  - class_weight = 'balanced'
  - random_state = 42
- Evaluated using ROC-AUC and precision-recall curves.

Outcome Highlights

Random Forest and XGBoost outperformed Logistic Regression, but it served well as a baseline model.

Performance Metrics:

- **Accuracy:** 0.9747
- **Precision:** 0.0591
- **Recall:** 0.9184
- **F1-Score:** 0.1110
- **AUC-ROC Score:** 0.9712

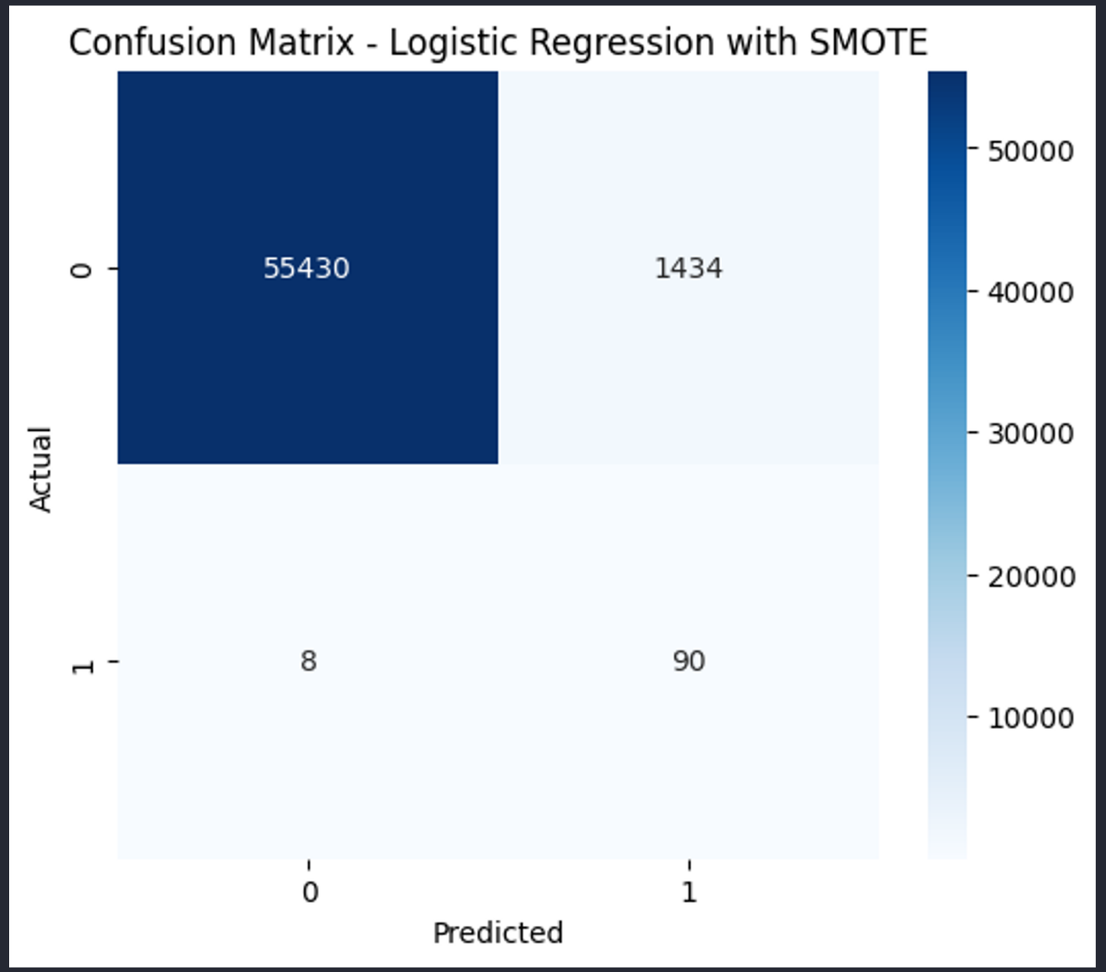
Strengths:

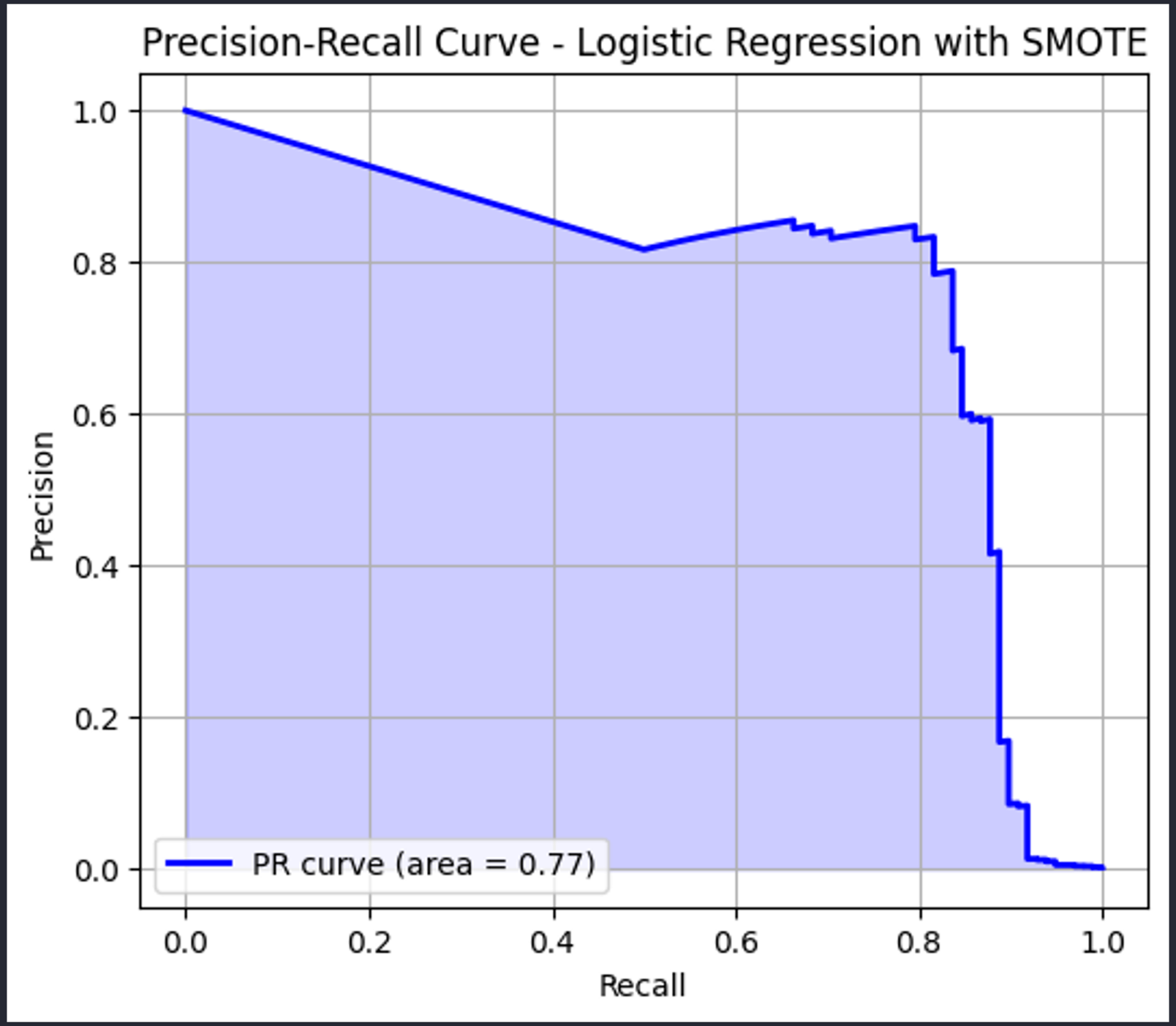
- Simple and interpretable model with well-calibrated probabilities.
- Balanced class weighting supported better fraud detection.
- Fast training and efficient for large datasets.

Logistic Regression with SMOTE Performance Metrics:
Accuracy: 0.9747
Precision: 0.0591
Recall: 0.9184
F1 Score: 0.1110
AUC-ROC Score: 0.9712
Average Precision Score: 0.7235

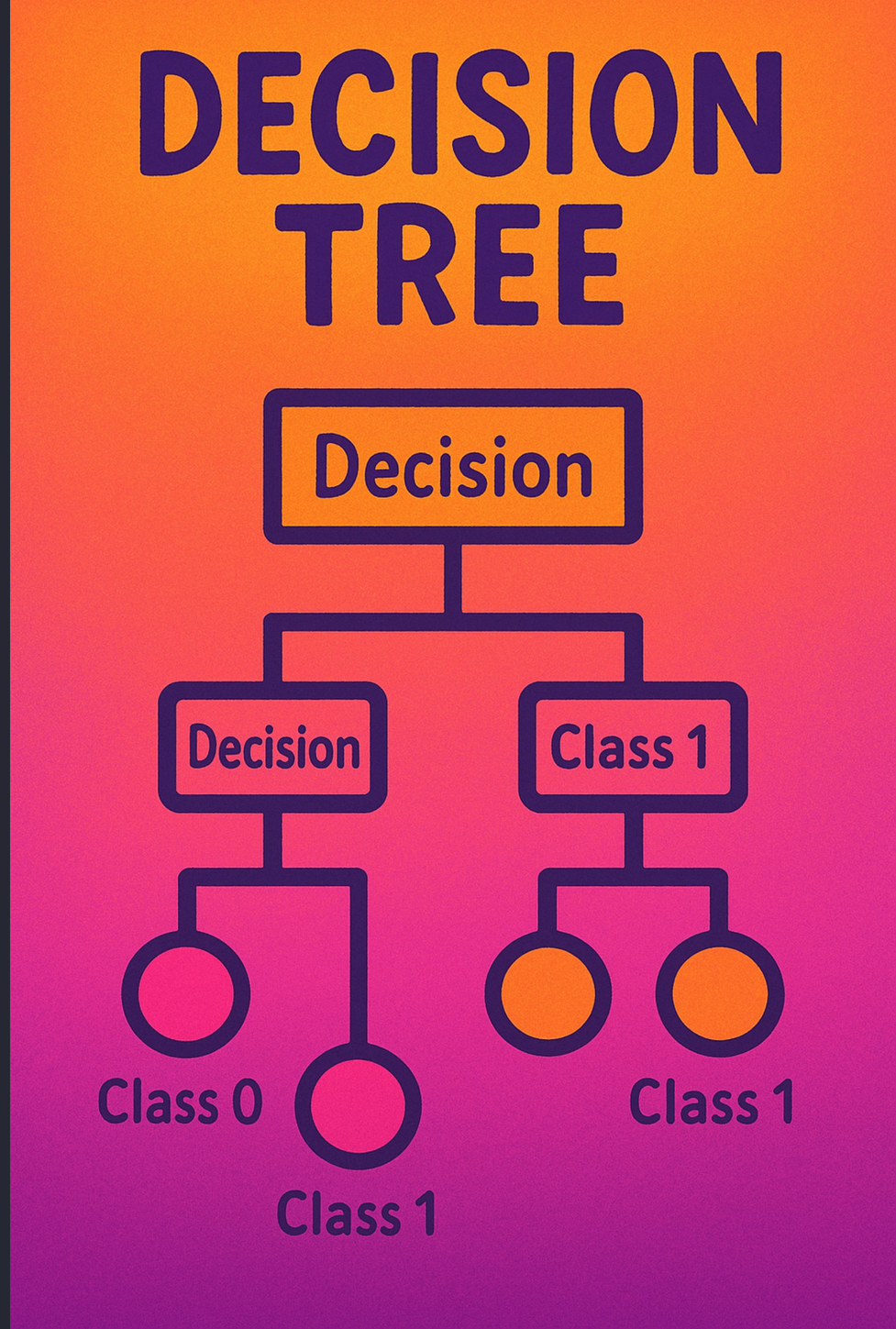
Classification Report:

	precision	recall	f1-score	support
0	1.00	0.97	0.99	56864
1	0.06	0.92	0.11	98
accuracy			0.97	56962
macro avg	0.53	0.95	0.55	56962
weighted avg	1.00	0.97	0.99	56962




Precision-Recall Curve - Logistic Regression with SMOTE
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
PR curve (area = 0.77)

Decision Tree



Overview

- A simple, interpretable model that splits data based on feature thresholds.
- Can overfit easily but provides insight into decision paths.
- Not robust to imbalance without class weighting.
- Serves as the base estimator in ensemble models like AdaBoost.

Application:

- Used as a standalone classifier and inside AdaBoost.
- Parameters:
  - class_weight = 'balanced'
  - random_state = 42
- Preprocessing: RobustScaler + SMOTE.
- Evaluated to understand baseline tree behavior on imbalanced data.

Outcome Highlights

Lowest performance among ensemble methods; used for comparison.

Performance Metrics:

- **Accuracy:** 0.9810
- **Precision:** 0.0705
- **Recall:** 0.8265
- **F1-Score:** 0.1299
- **AUC-ROC Score:** 0.8704

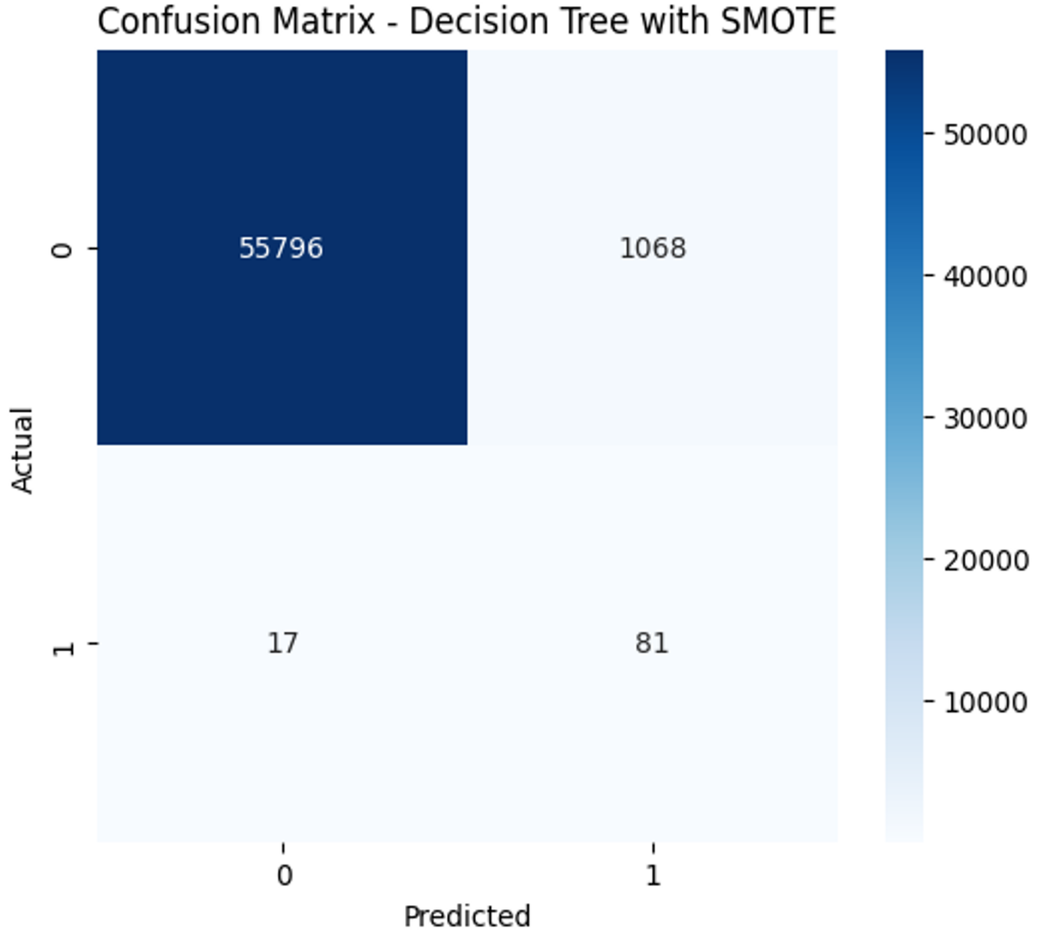
Strengths:

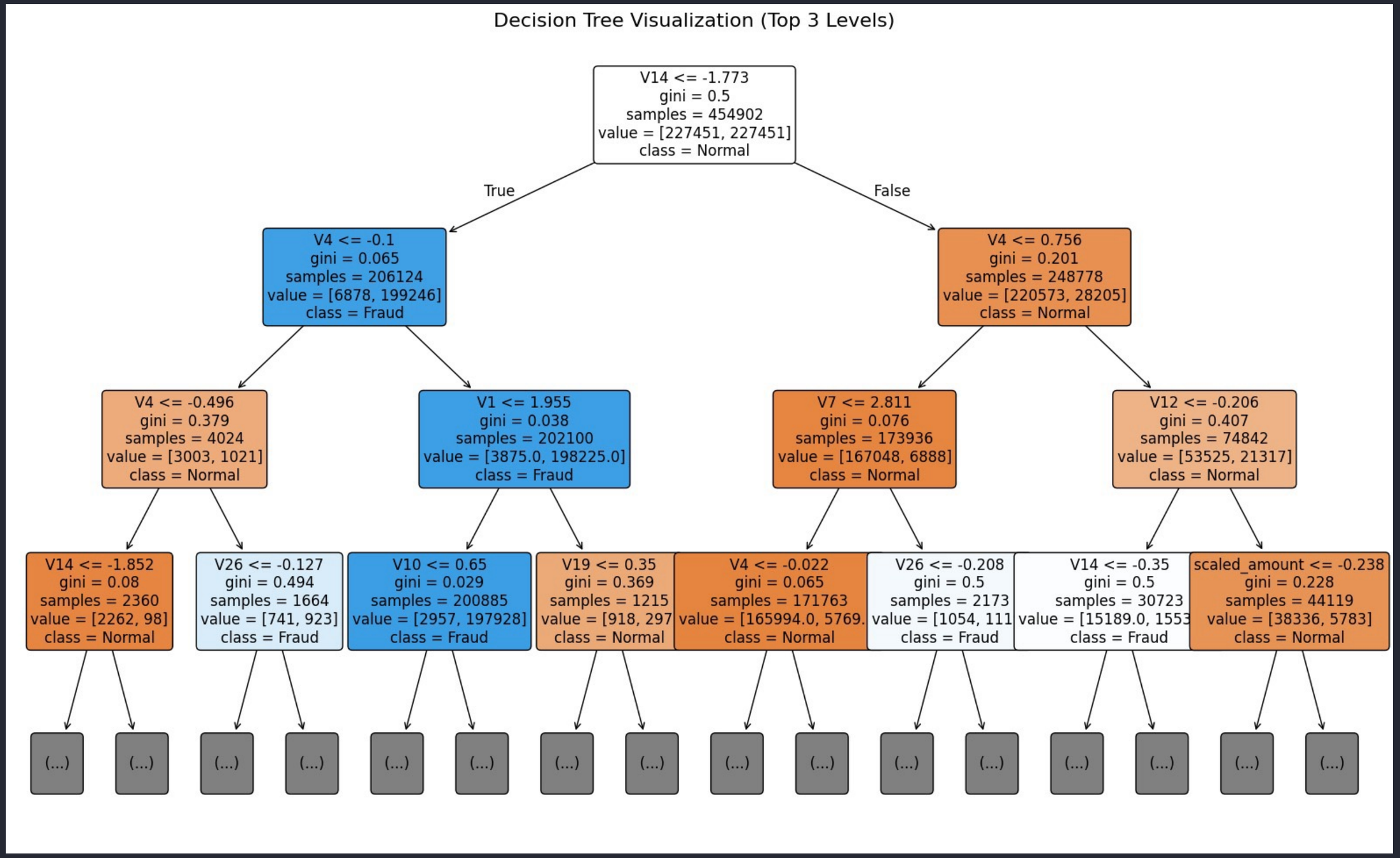
- Simple and interpretable.
- Forms the basis for boosting and bagging models.
- Quick inference but prone to overfitting.

Decision Tree with SMOTE Performance Metrics:
Accuracy: 0.9810
Precision: 0.0705
Recall: 0.8265
F1 Score: 0.1299
AUC-ROC Score: 0.8704
Average Precision Score: 0.4408

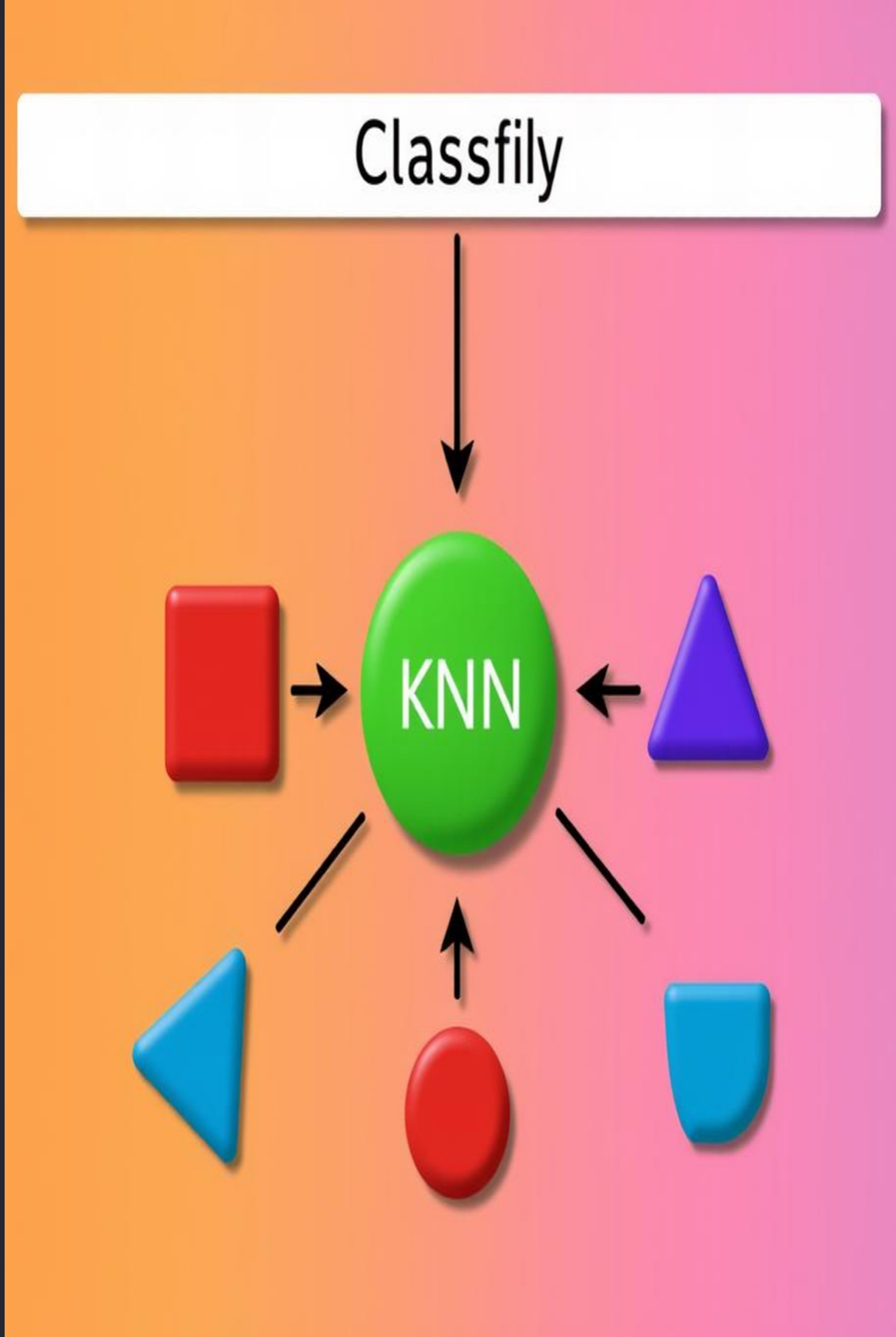
Classification Report:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	56864
1	0.07	0.83	0.13	98
accuracy			0.98	56962
macro avg	0.54	0.90	0.56	56962
weighted avg	1.00	0.98	0.99	56962




Decision Tree Visualization (Top 3 Levels)
V14 <= -1.773
gini = 0.5
samples = 454902
value = [227451, 227451]
class = Normal
True
False
V4 <= -0.1
gini = 0.065
samples = 206124
value = [6878, 199246]
class = Fraud
V4 <= 0.756
gini = 0.201
samples = 248778
value = [220573, 28205]
class = Normal
V4 <= -0.496
gini = 0.379
samples = 4024
value = [3003, 1021]
class = Normal
V1 <= 1.955
gini = 0.038
samples = 202100
value = [3875.0, 198225.0]
class = Fraud
V7 <= 2.811
gini = 0.076
samples = 173936
value = [167048, 6888]
class = Normal
V12 <= -0.206
gini = 0.407
samples = 74842
value = [53525, 21317]
class = Normal
V14 <= -1.852
gini = 0.08
samples = 2360
value = [2262, 98]
class = Normal
V26 <= -0.127
gini = 0.494
samples = 1664
value = [741, 923]
class = Fraud
V10 <= 0.65
gini = 0.029
samples = 200885
value = [2957, 197928]
class = Fraud
V19 <= 0.35
gini = 0.369
samples = 1215
value = [918, 297
class = Normal
V4 <= -0.022
gini = 0.065
samples = 171763
value = [165994.0, 5769.
class = Normal
V26 <= -0.208
gini = 0.5
samples = 2173
value = [1054, 111
class = Fraud
V14 <= -0.35
gini = 0.5
samples = 30723
value = [15189.0, 1553
class = Fraud
scaled_amount <= -0.238
gini = 0.228
samples = 44119
value = [38336, 5783]
class = Normal
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)

K-NEAREST NEIGHBORS (KNN)



Overview

- KNN is a non-parametric, instance-based machine learning algorithm.
- It classifies samples based on the majority vote of their k-nearest neighbors in the feature space.
- Distance-based method: Similar points are likely to belong to the same class.
- In fraud detection, rare minority class (fraud) makes KNN sensitive to data imbalance.
- Drawback: Prediction time grows with dataset size, making it computationally expensive for large datasets.

Application:

- RobustScaler was applied to scale 'Amount' and 'Time' features, reducing the effect of outliers.
- SMOTE technique was used to balance the training data (since fraud cases were rare).
- Key KNN Settings:
  - n_neighbors = 5 (5 nearest neighbors)
  - weights = 'distance' (closer neighbors have more impact)
  - algorithm = 'kd_tree' (faster search for neighbors)
- Training strategy:
  - Dataset had 264k+ entries.
  - KNN was trained on a random subset of 20,000 samples (to reduce prediction time).
- KNN was evaluated on the full test set (~49,000+ samples).
- Predicted probabilities were used to calculate metrics like AUC and F1-Score.

Outcome Highlights

KNN served as a reasonable baseline model but faced challenges due to the high computation cost.

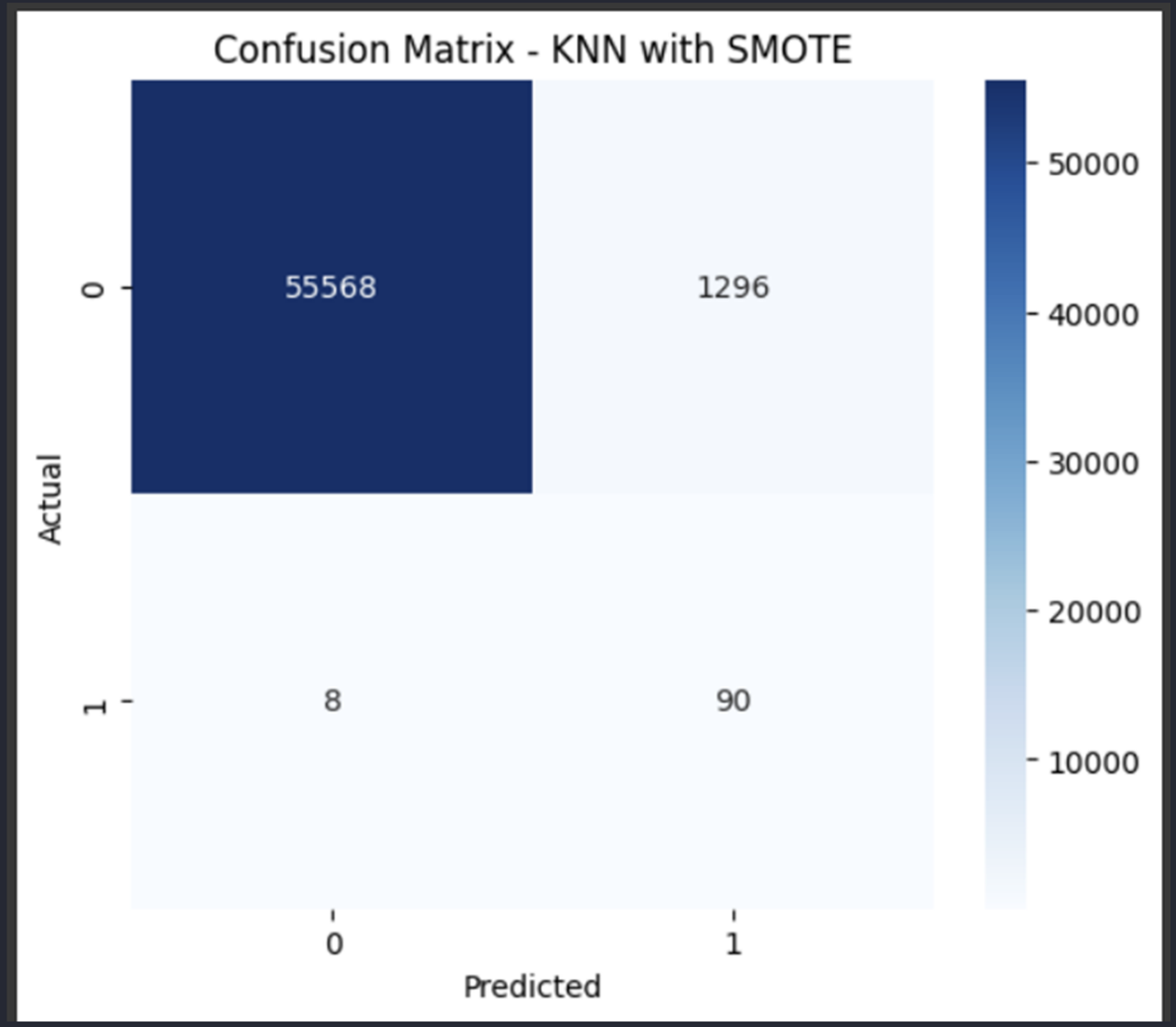
Performance Metrics:

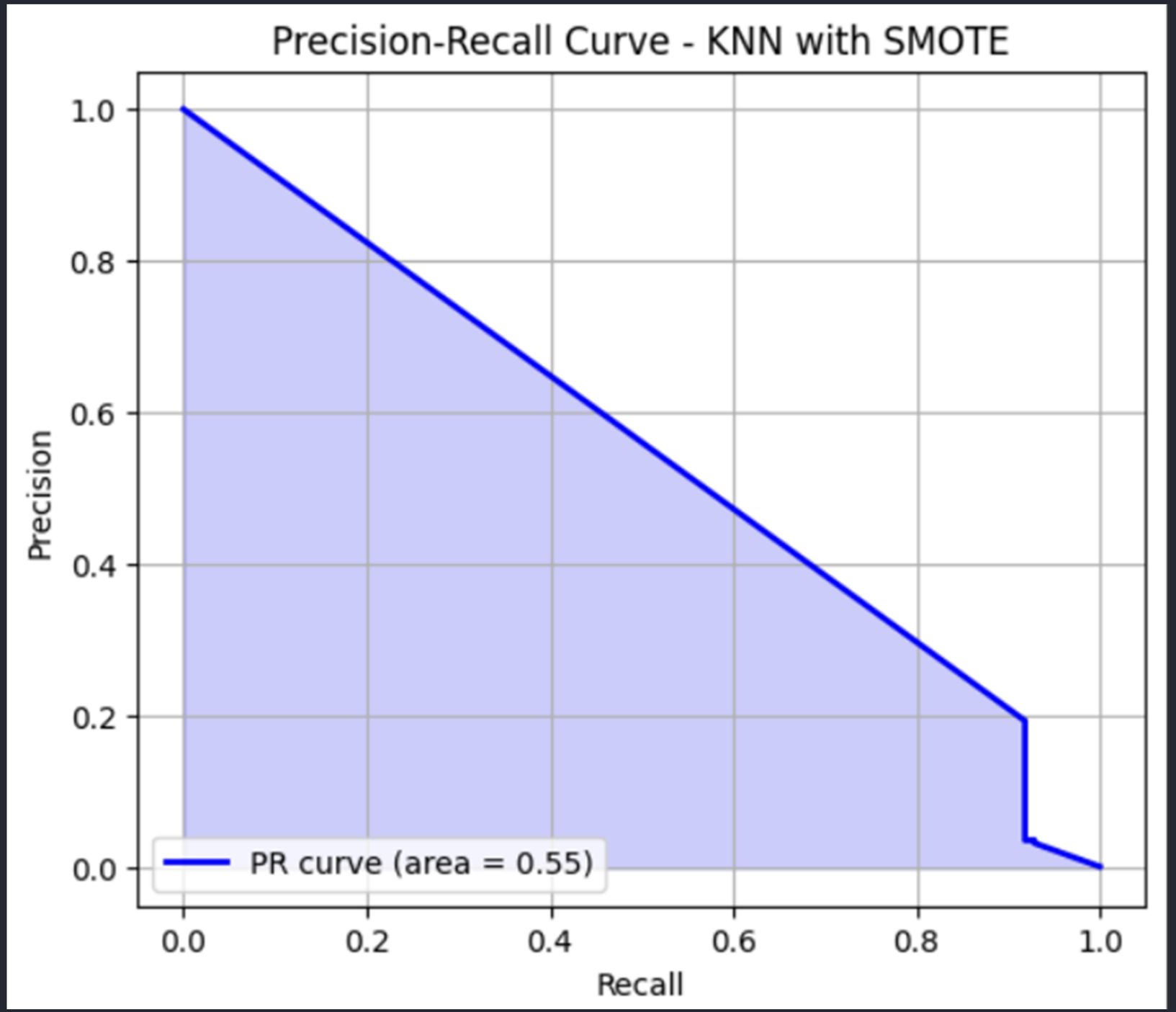
- **Accuracy:** 0.9771
- **Precision:** 0.0649
- **Recall:** 0.9184
- **F1-Score:** 0.1213 (very low, because of high class imbalance)
- **AUC-ROC Score:** 0.9591

Limitation:

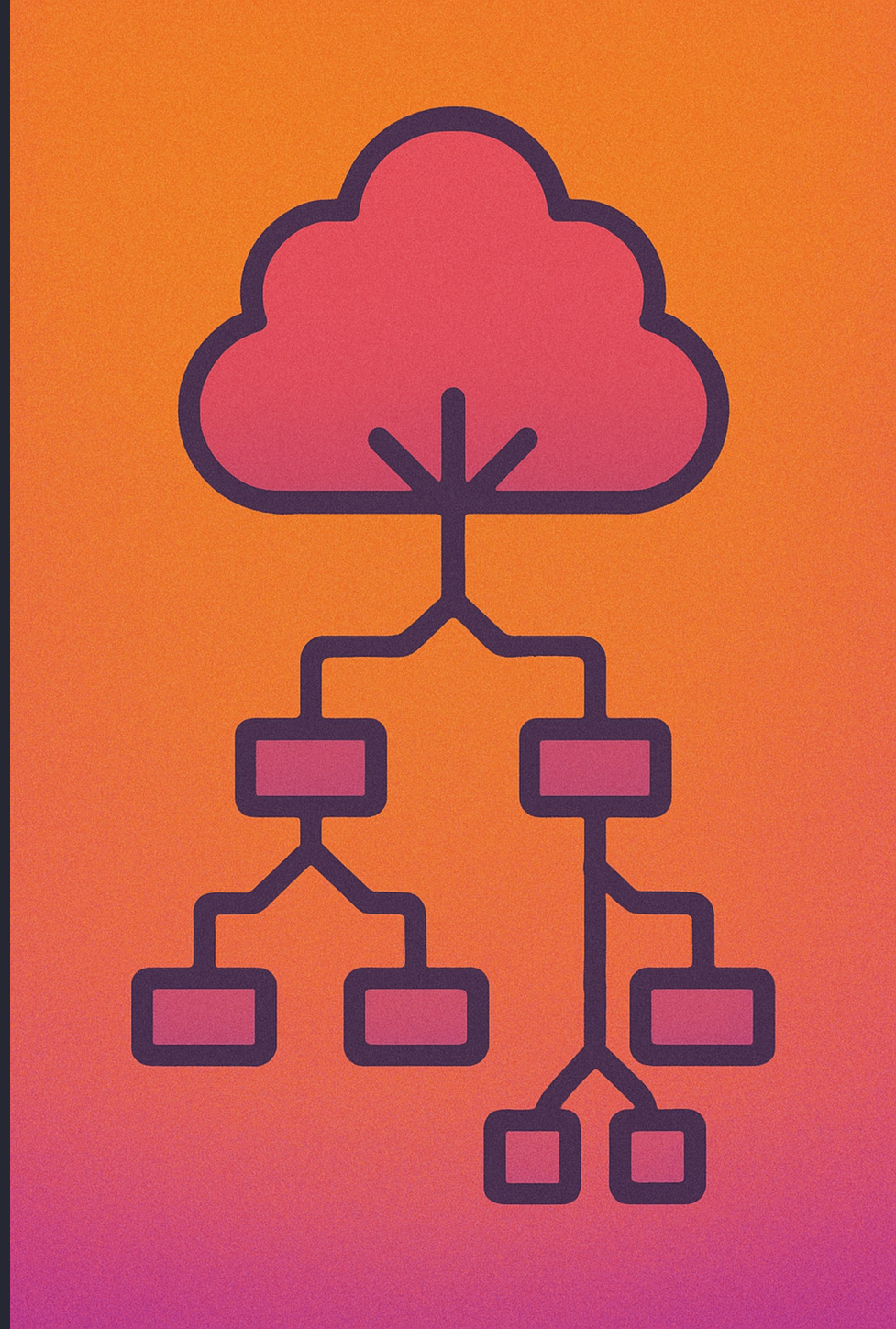
- Slow prediction on large datasets.
- Struggled with minority class (fraud) even after SMOTE.

```
KNN with SMOTE Performance Metrics:
Accuracy: 0.9771
Precision: 0.0649
Recall: 0.9184
F1 Score: 0.1213
AUC-ROC Score: 0.9591
Average Precision Score: 0.1786

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     56864
           1       0.06      0.92      0.12        98

    accuracy                           0.98     56962
   macro avg       0.53      0.95      0.55     56962
weighted avg       1.00      0.98      0.99     56962
```





RANDOM FOREST CLASSIFIER



Overview

- Random Forest is an ensemble method that builds many decision trees and aggregates their predictions.
- It reduces overfitting and improves generalization by averaging multiple trees.
- Random Forests are highly robust against class imbalance and scale well to large datasets.
- Well-suited for fraud detection where complex patterns need to be captured.

Application:

- RobustScaler was used for preprocessing, same as for KNN.
- SMOTE was applied to rebalance the dataset for training.
- Training strategy:
  - Model trained on the full SMOTE-balanced training set.
- Key Random Forest Settings:
  - n_estimators = 100 (100 trees)
  - class_weight = 'balanced' (giving equal importance to minority class)
  - random_state = 42 (reproducibility)
- Feature importance was extracted to analyze top contributing features.

Outcome Highlights

Random Forest outperformed KNN significantly in all metrics.

Key Feature Importances: 'V14', 'V17', 'V12', and 'V10' were major predictors.

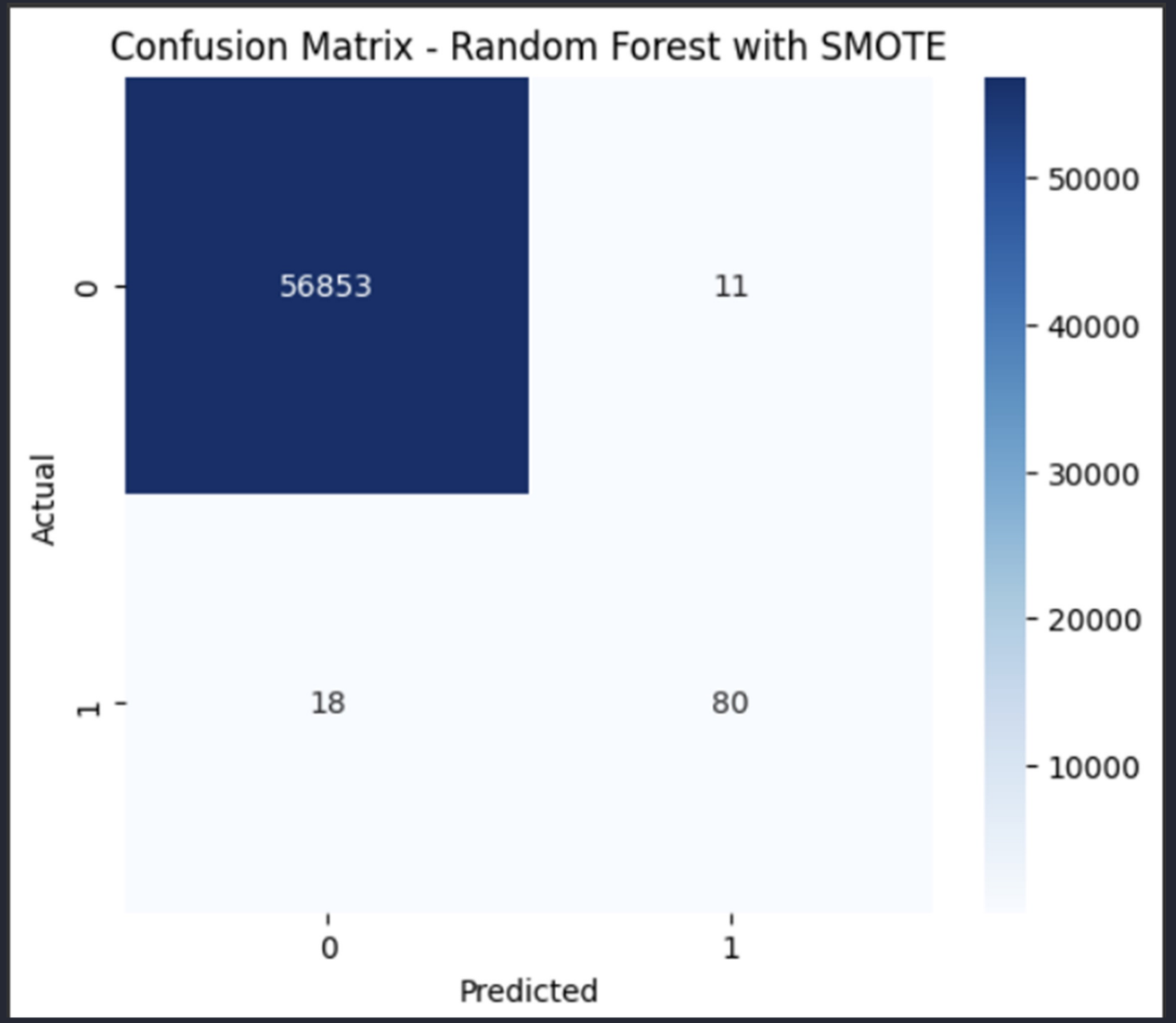
Performance Metrics:

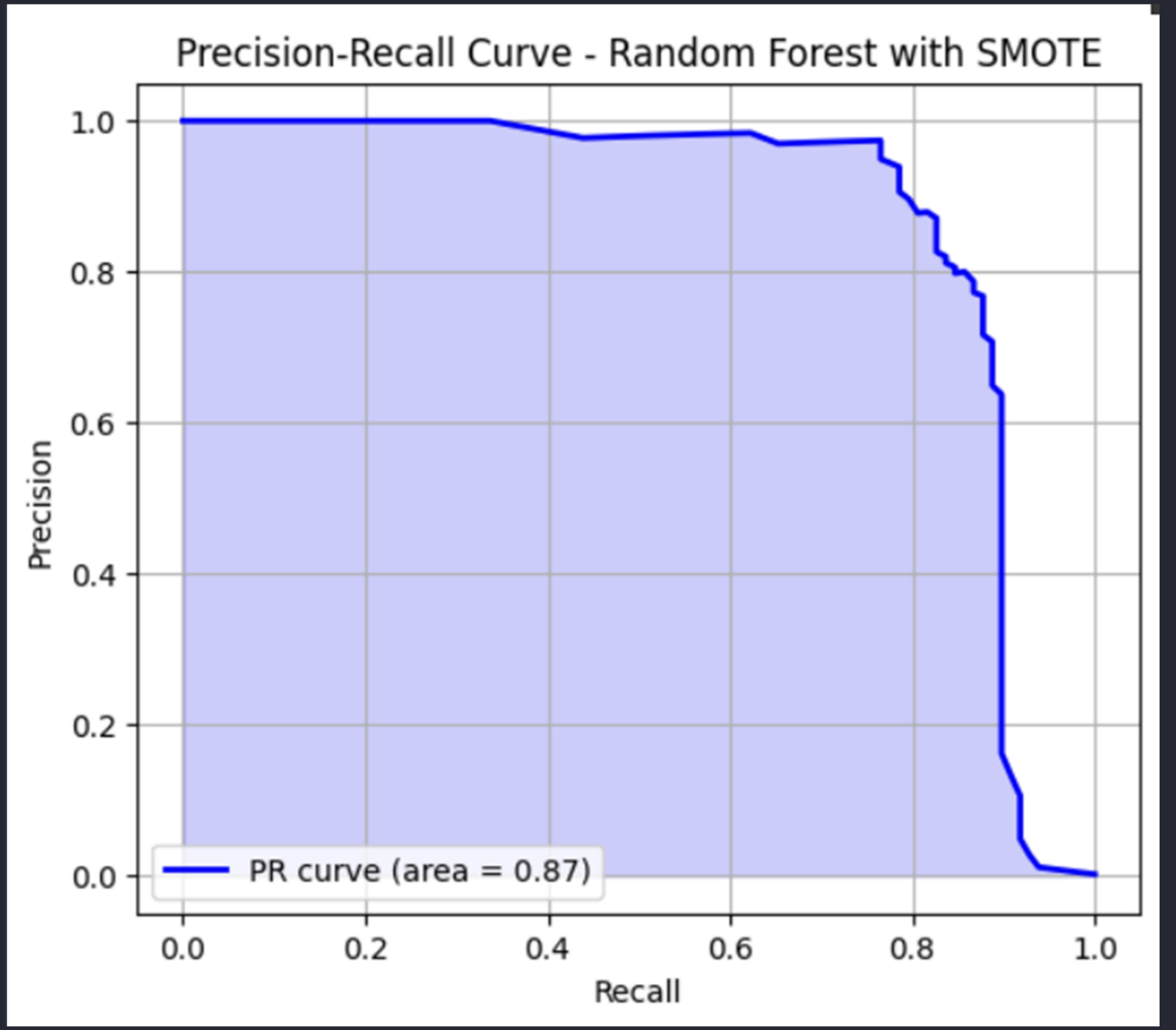
- **Accuracy:** 0.9991
- **Precision:** 0.8791
- **Recall:** 0.8163
- **F1-Score:** 0.8466
- **AUC-ROC Score:** 0.9632

Strengths:

- Very high precision and recall even for the minority class (fraud).
- Fast prediction time and excellent generalization.

```
Random Forest with SMOTE Performance Metrics:
Accuracy: 0.9995
Precision: 0.8791
Recall: 0.8163
F1 Score: 0.8466
AUC-ROC Score: 0.9632
Average Precision Score: 0.8680

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.88      0.82      0.85        98

    accuracy                           1.00     56962
   macro avg       0.94      0.91      0.92     56962
weighted avg       1.00      1.00      1.00     56962
```




Precision-Recall Curve - Random Forest with SMOTE
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
PR curve (area = 0.87)

Gradient Boosting

Overview

- Gradient Boosting builds additive models in a sequential manner to minimize prediction error.
- It adjusts weights based on the gradient of the loss function for better learning.
- Handles non-linear patterns well and can be tuned for high performance.
- More prone to overfitting if not properly regularized.
- Works well with imbalanced datasets using class weights or sampling.

Application:

- Used RobustScaler and SMOTE-preprocessed data.
- Trained on full SMOTE-balanced set.
- GradientBoostingClassifier parameters:
  - n_estimators = 100
  - learning_rate = 0.1
  - random_state = 42
- Compared across precision, recall, F1-score, and ROC-AUC.

Outcome Highlights

Performed close to XGBoost but with slightly higher training time.

Performance Metrics:

- **Accuracy:** 0.9870
- **Precision:** 0.1085
- **Recall:** 0.9082
- **F1-Score:** 0.1939
- **AUC-ROC Score:** 0.9799

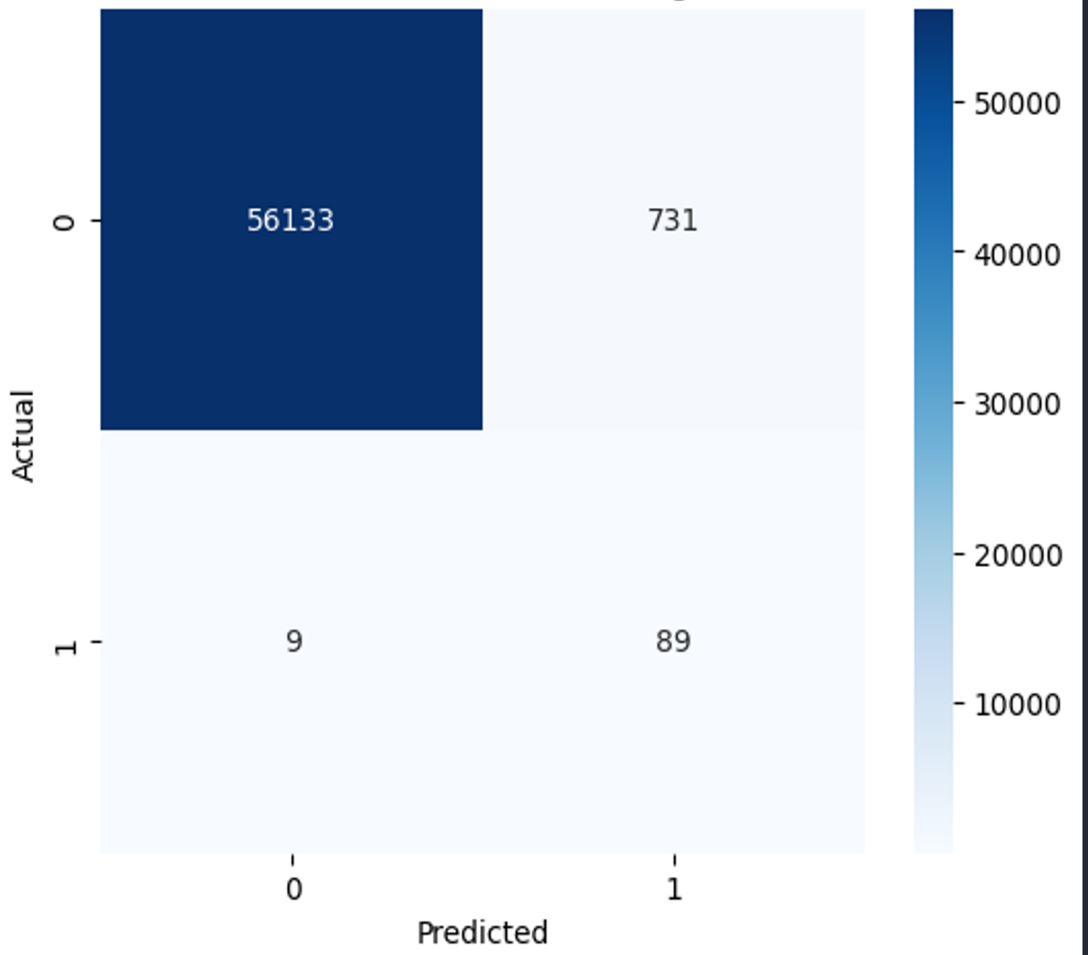
Strengths:

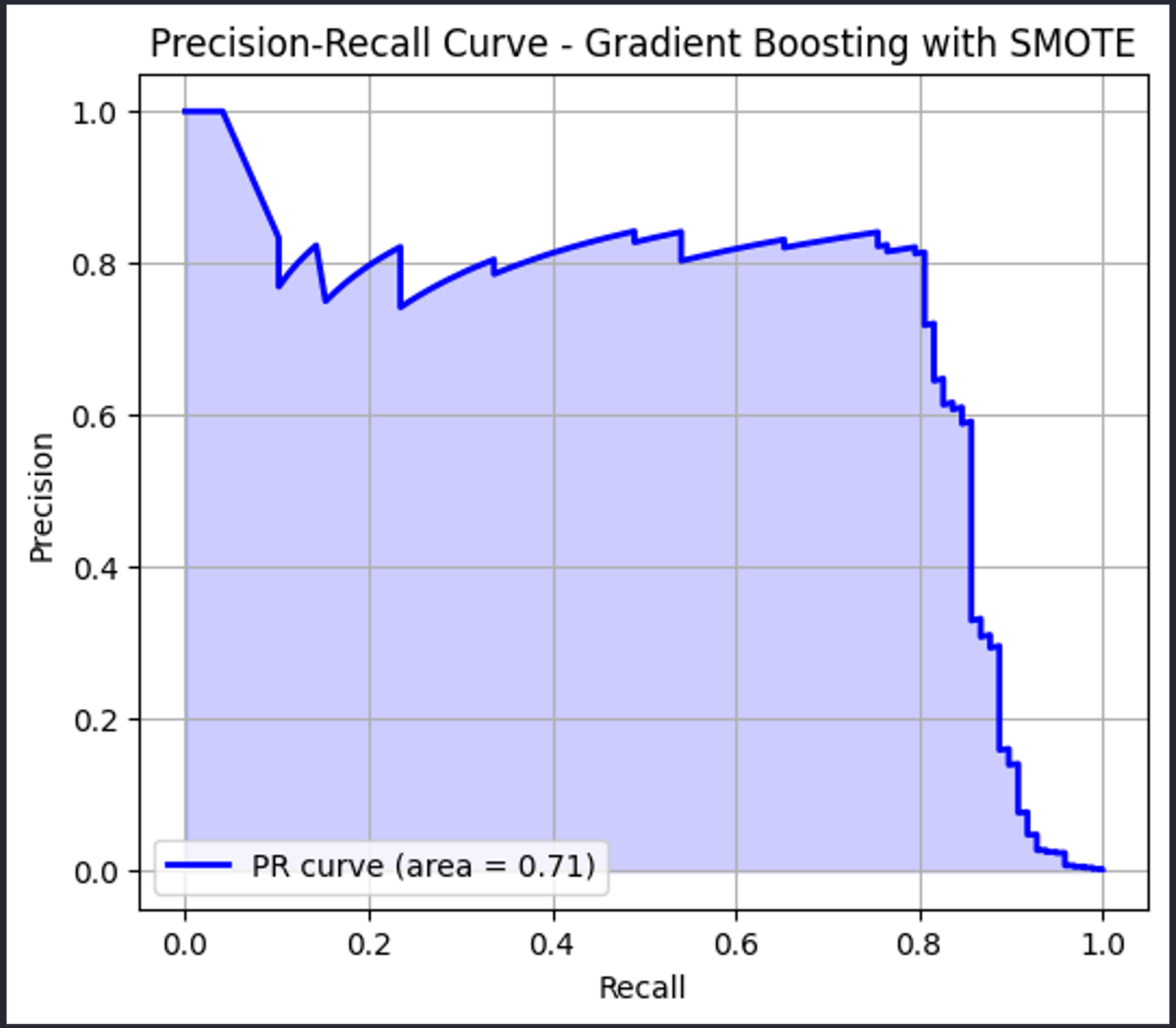
- High precision and recall for fraud detection.
- Captured complex patterns in transaction behavior.
- Lower overfitting compared to single tree models.

Gradient Boosting with SMOTE Performance Metrics:
Accuracy: 0.9870
Precision: 0.1085
Recall: 0.9082
F1 Score: 0.1939
AUC-ROC Score: 0.9799
Average Precision Score: 0.7108

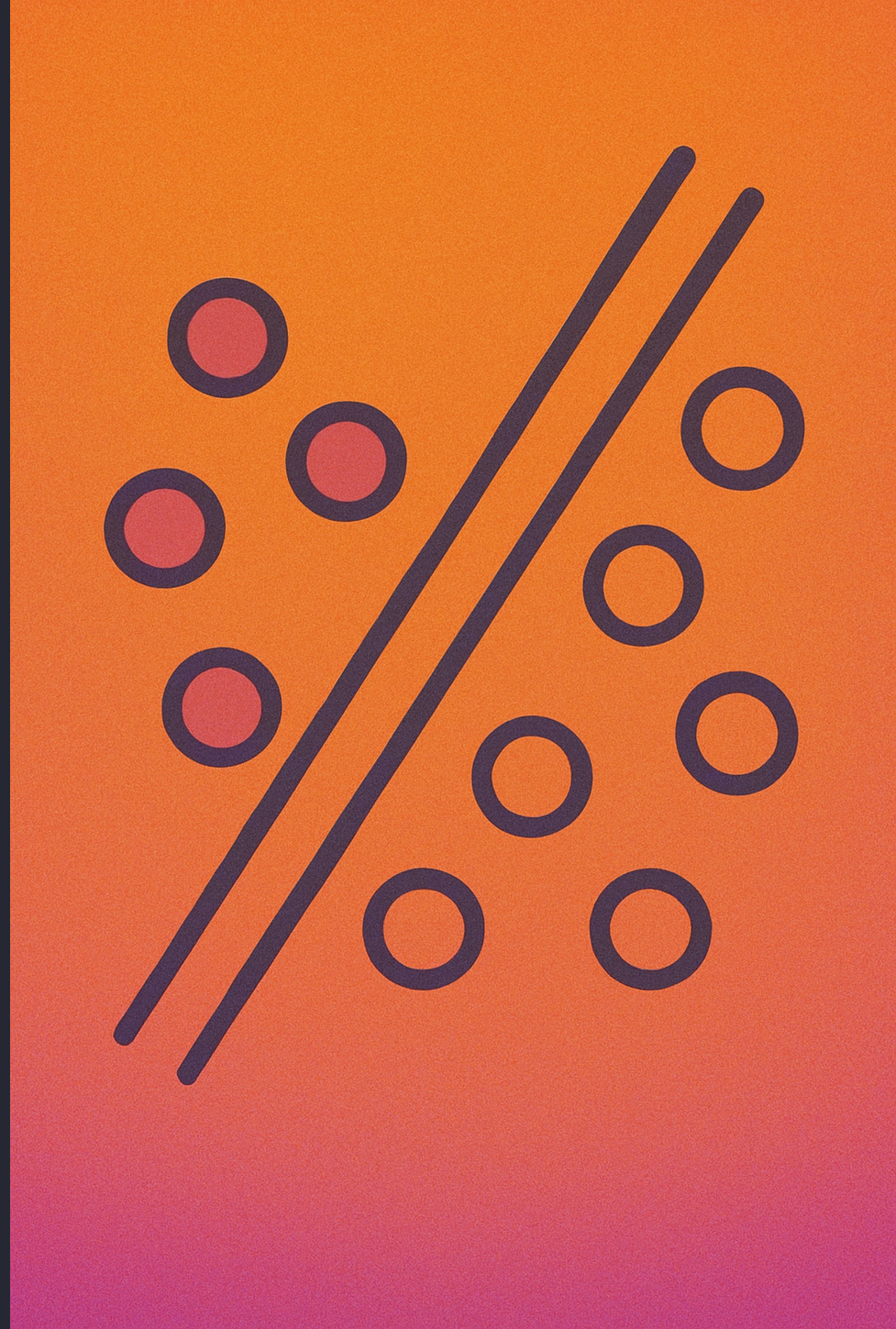
Classification Report:

	precision	recall	f1-score	support
0	1.00	0.99	0.99	56864
1	0.11	0.91	0.19	98
accuracy			0.99	56962
macro avg	0.55	0.95	0.59	56962
weighted avg	1.00	0.99	0.99	56962




Precision-Recall Curve - Gradient Boosting with SMOTE
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
PR curve (area = 0.71)

SUPPORT VECTOR MACHINE



Overview

- Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification tasks.
- It constructs an optimal hyperplane that maximizes the margin between different classes.
- SVMs are effective in high-dimensional spaces and are robust to overfitting, especially in low-sample, high-feature settings.
- Kernel trick (e.g., RBF, linear) allows SVM to handle non-linear decision boundaries.
- Well-suited for fraud detection where class separation in imbalanced data is critical.

Application:

- Used SVC from sklearn.svm with class weights balanced for fairness to minority class.
- Preprocessing: RandomUnderSampler was applied to rebalance class distribution.
- Training Strategy:
- Model trained on the full under-sampled training dataset.
- Key SVM Settings:
- class_weight = 'balanced' (automatically balances minority class)
- probability = True (enabled to generate prediction probabilities)
- random_state = 42 (ensures reproducibility)
- Model evaluated using metrics like accuracy, recall, precision, F1-score, AUC, and average precision.

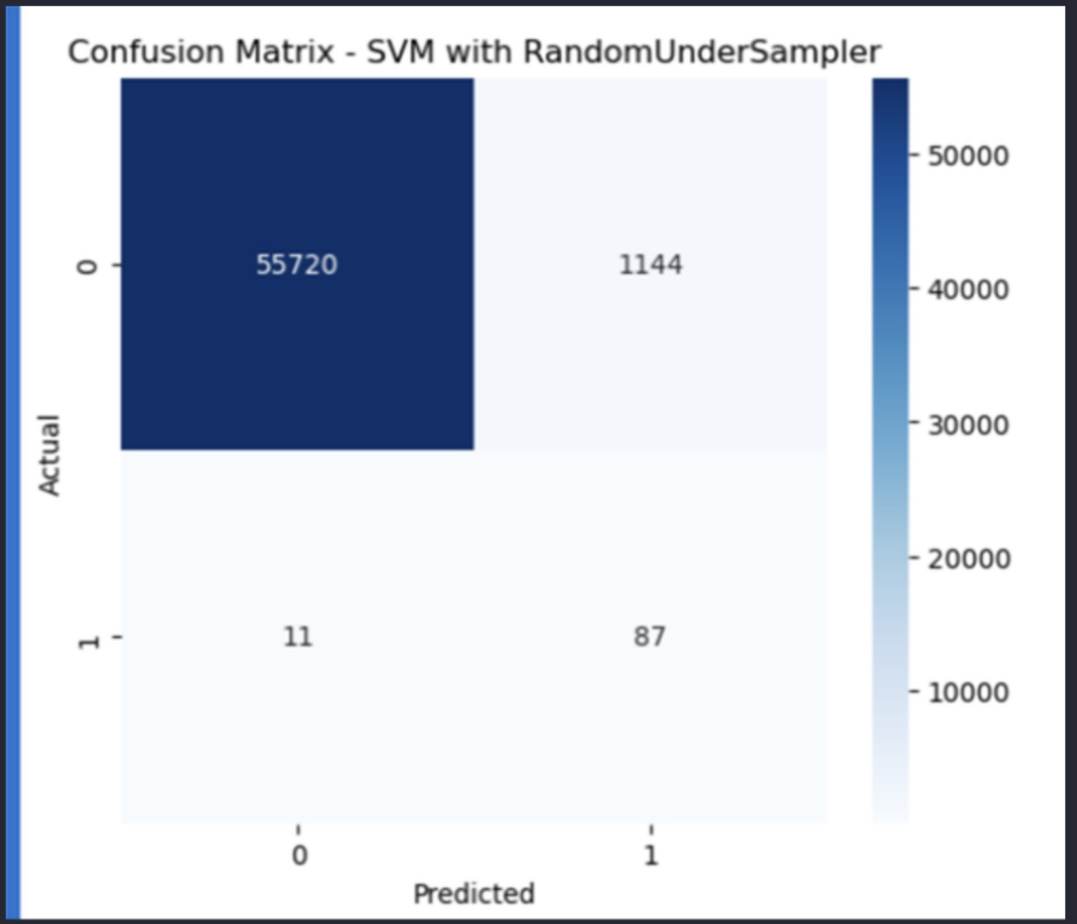
Outcome Highlights

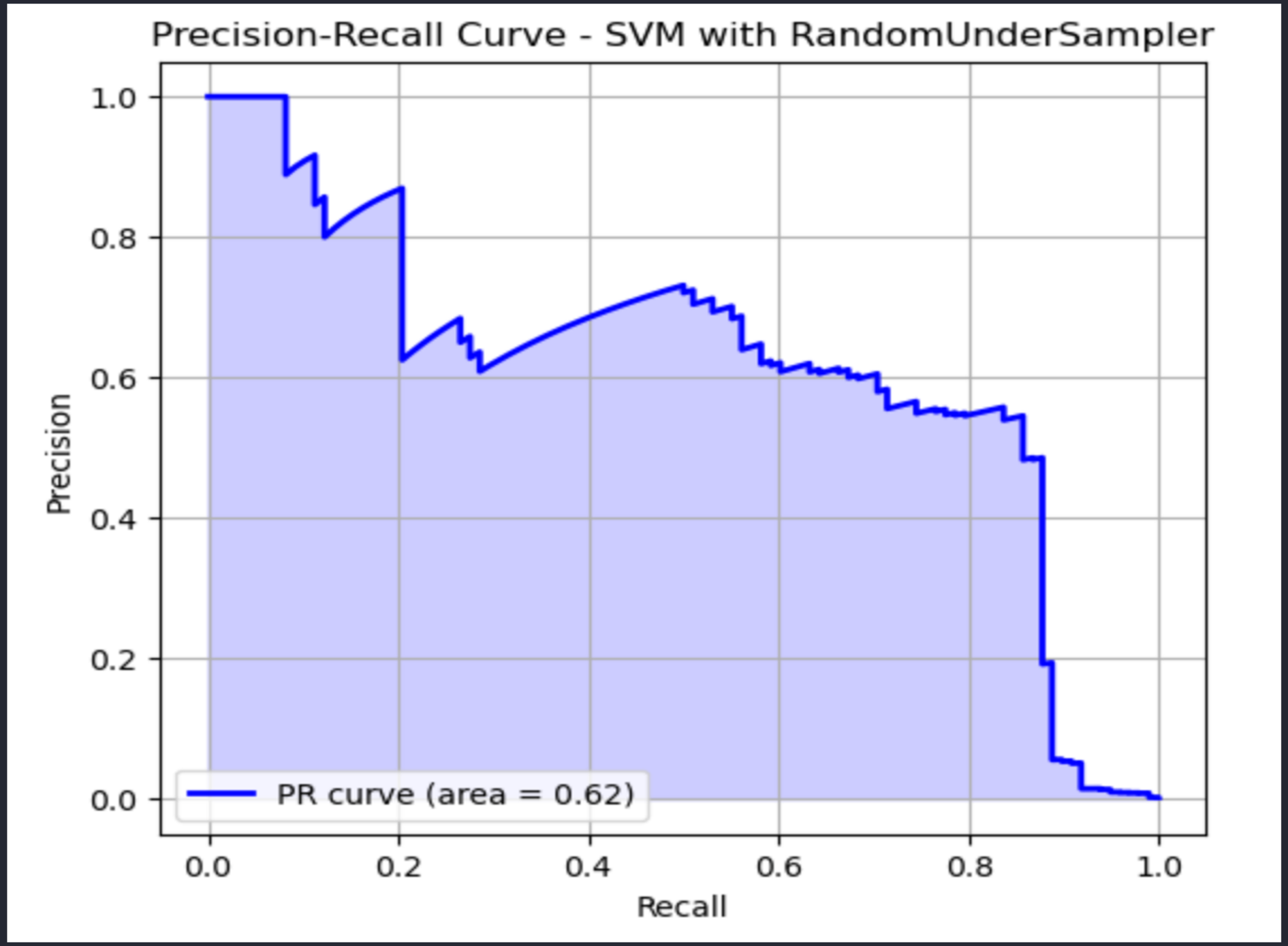
- **Performance Metrics:**
- **Accuracy: 0.9797**
- **Precision: 0.0707**
- **Recall: 0.8878**
- **F1 Score: 0.1309**
- **AUC-ROC Score: 0.9786**
- **Average Precision Score: 0.6174**
- **Classification Report:**
- **Class 0 (Non-Fraud): Precision = 1.00, Recall = 0.98, F1-Score = 0.99**
- **Class 1 (Fraud): Precision = 0.07, Recall = 0.89, F1-Score = 0.13**
- **Macro Avg: Precision = 0.54, Recall = 0.93, F1-Score = 0.56**
- **Weighted Avg: Precision = 1.00, Recall = 0.98, F1-Score = 0.99**

SVM with RandomUnderSampler Performance Metrics:
Accuracy: 0.9797
Precision: 0.0707
Recall: 0.8878
F1 Score: 0.1309
AUC-ROC Score: 0.9786
Average Precision Score: 0.6174

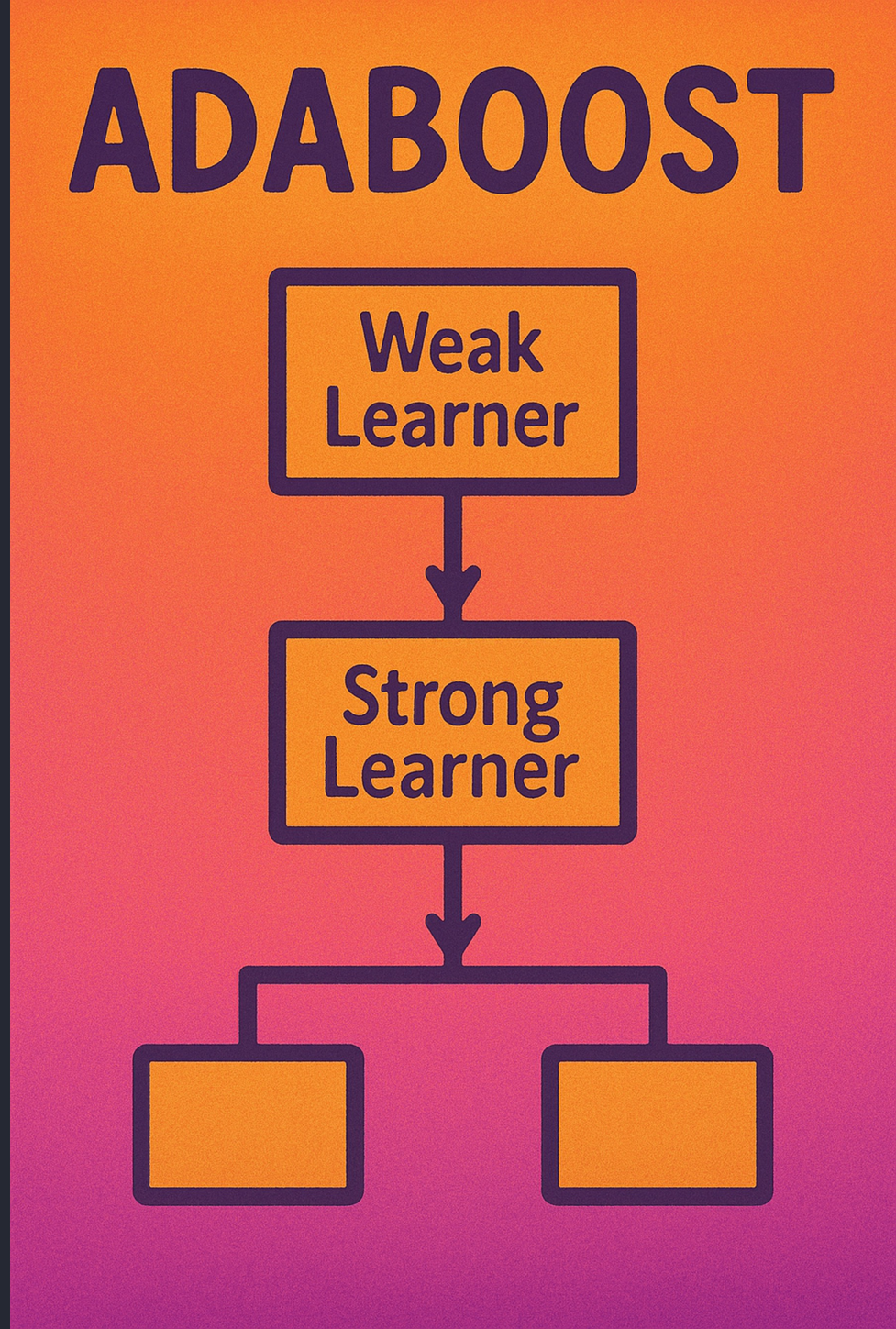
Classification Report:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	56864
1	0.07	0.89	0.13	98
accuracy			0.98	56962
macro avg	0.54	0.93	0.56	56962
weighted avg	1.00	0.98	0.99	56962




Precision-Recall Curve - SVM with RandomUnderSampler
Precision
Recall
PR curve (area = 0.62)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

AdaBoost



Overview

- AdaBoost builds a strong classifier by combining many weak learners (typically decision trees).
- Adjusts weights on misclassified samples to focus subsequent learners on harder cases.
- Less prone to overfitting than standard decision trees.
- Works well for datasets with noise and moderate imbalance.

Application:

- Preprocessing: RobustScaler + SMOTE.
- AdaBoostClassifier with base estimator: decision tree (max_depth=1).
- Parameters:
  - n_estimators=50
  - learning_rate=1.0
  - random_state=42
- Evaluated for precision and recall on the minority (fraud) class.

Outcome Highlights

Performed moderately well, better than standalone decision tree.

Performance Metrics:

- **Accuracy:** 0.9808
- **Precision:** 0.0757
- **Recall:** 0.9082
- **F1-Score:** 0.1398
- **AUC-ROC Score:** 0.9747

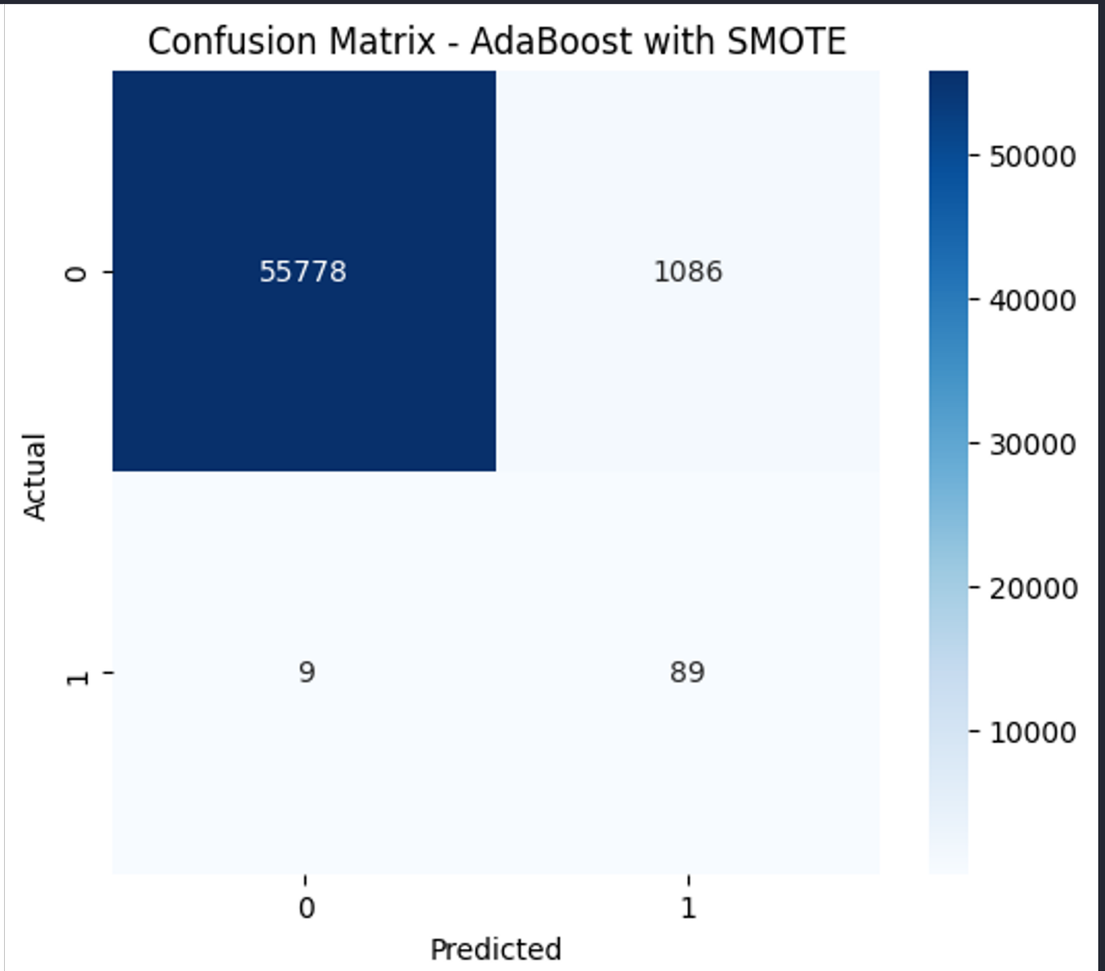
Strengths:

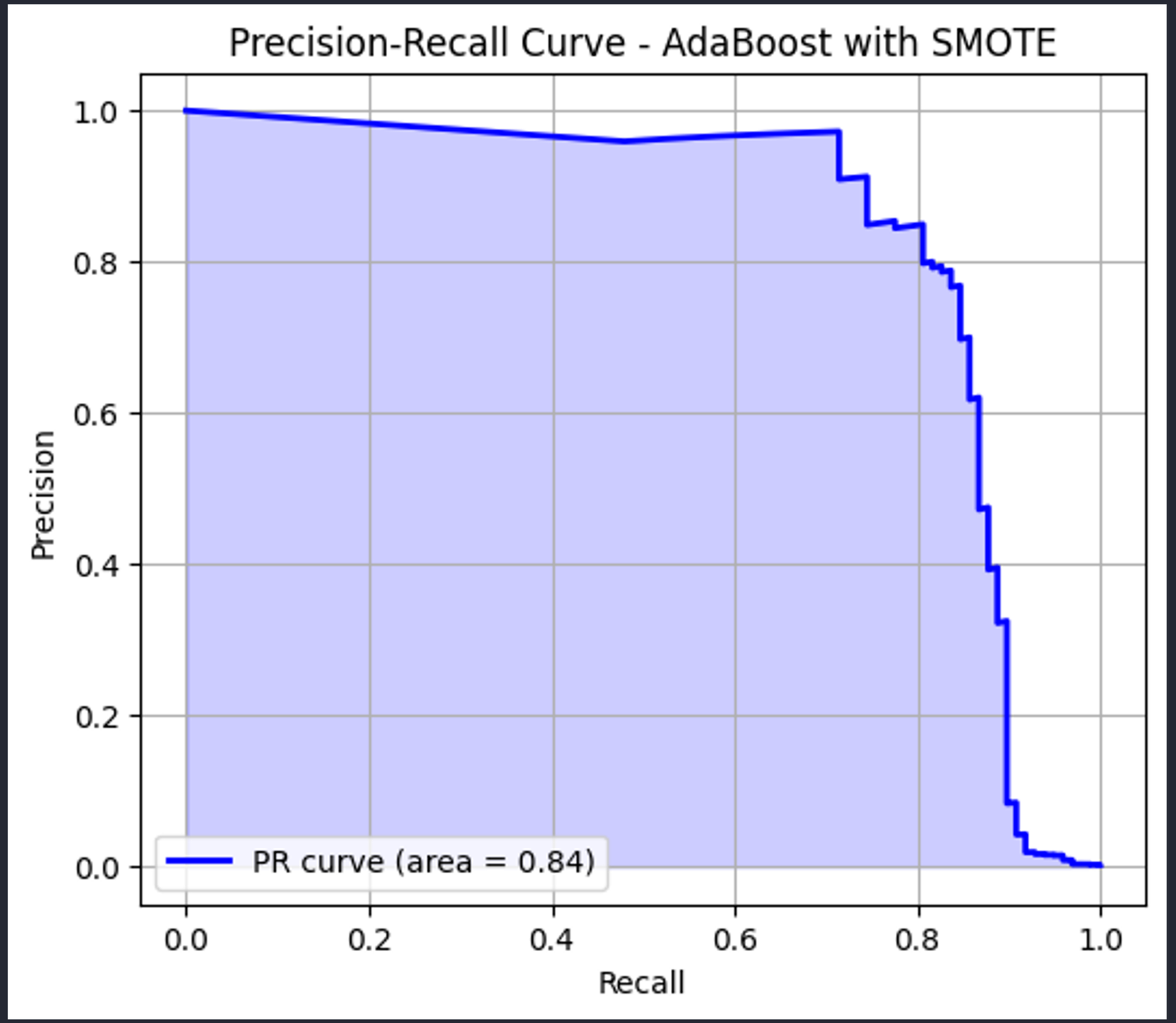
- Improved learning on misclassified samples.
- Lightweight model with fast inference.
- Good tradeoff between performance and complexity.

AdaBoost with SMOTE Performance Metrics:
Accuracy: 0.9808
Precision: 0.0757
Recall: 0.9082
F1 Score: 0.1398
AUC-ROC Score: 0.9747
Average Precision Score: 0.8269

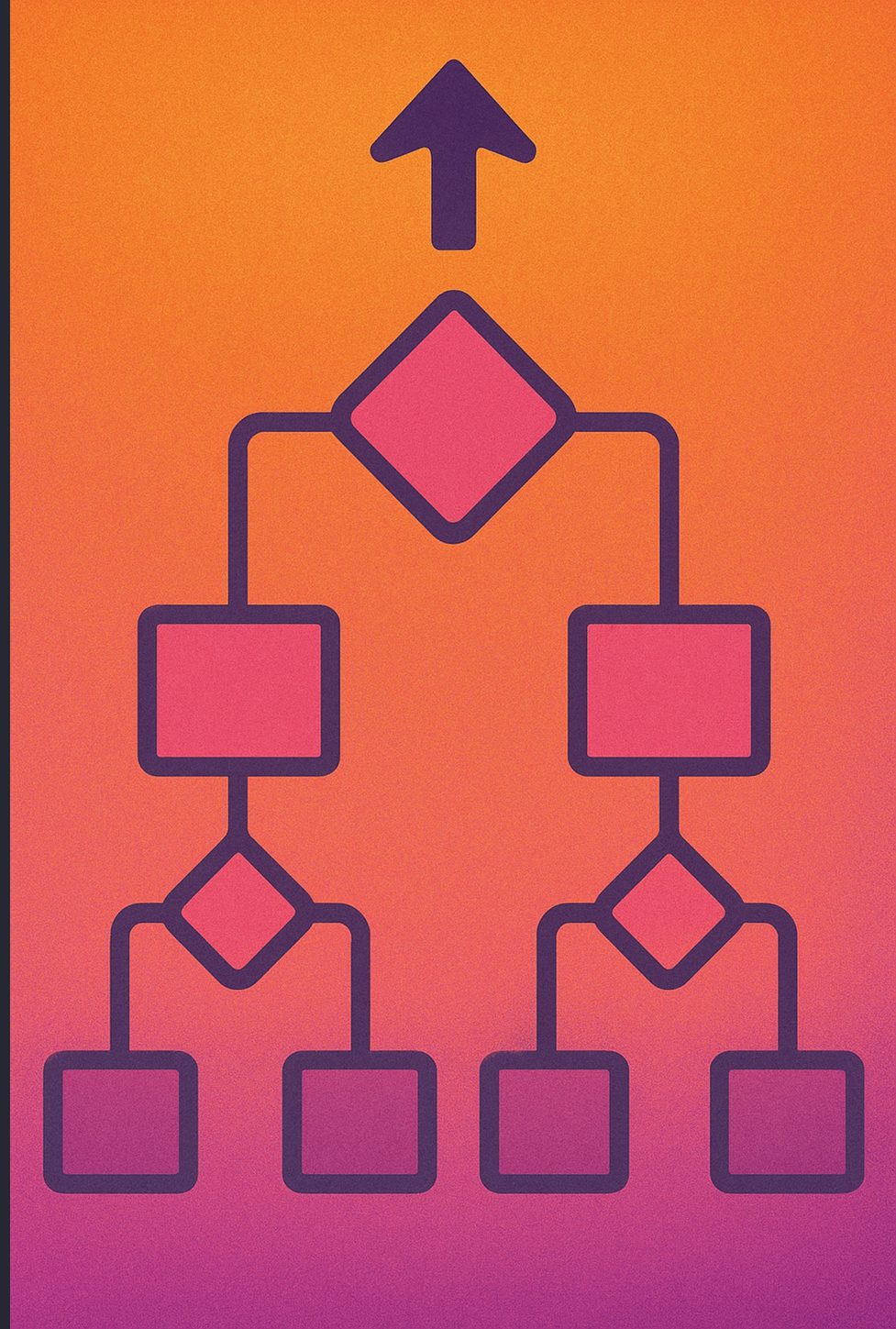
Classification Report:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	56864
1	0.08	0.91	0.14	98
accuracy			0.98	56962
macro avg	0.54	0.94	0.57	56962
weighted avg	1.00	0.98	0.99	56962




Precision-Recall Curve - AdaBoost with SMOTE
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
PR curve (area = 0.84)

XGBoost



Overview

- An optimized and regularized version of gradient boosting.
- Incorporates both L1 and L2 regularization to prevent overfitting.
- Highly efficient and scalable for large datasets.
- Handles missing values and imbalance effectively with scale_pos_weight.

Application:

- SMOTE was used to balance training data.
- Parameters:
  - learning_rate=0.01
  - n_estimators=500
  - max_depth=5
  - subsample=0.8
  - colsample_bytree=0.8
  - random_state=42
  - use_label_encoder=False
  - eval_metric='logloss'
- Feature importance was visualized to interpret key predictors.
- Delivered best performance across most metrics (precision, recall, ROC-AUC).

Outcome Highlights

Top performer across all metrics and fraud detection ability.

Key Feature Importances: ‘V14’, ‘V10’, ‘V12’, ‘V17’ contributed significantly.

Performance Metrics:

- **Accuracy:** 0.9912
- **Precision:** 0.1525
- **Recall:** 0.8980
- **F1-Score:** 0.2607
- **AUC-ROC Score:** 0.9767

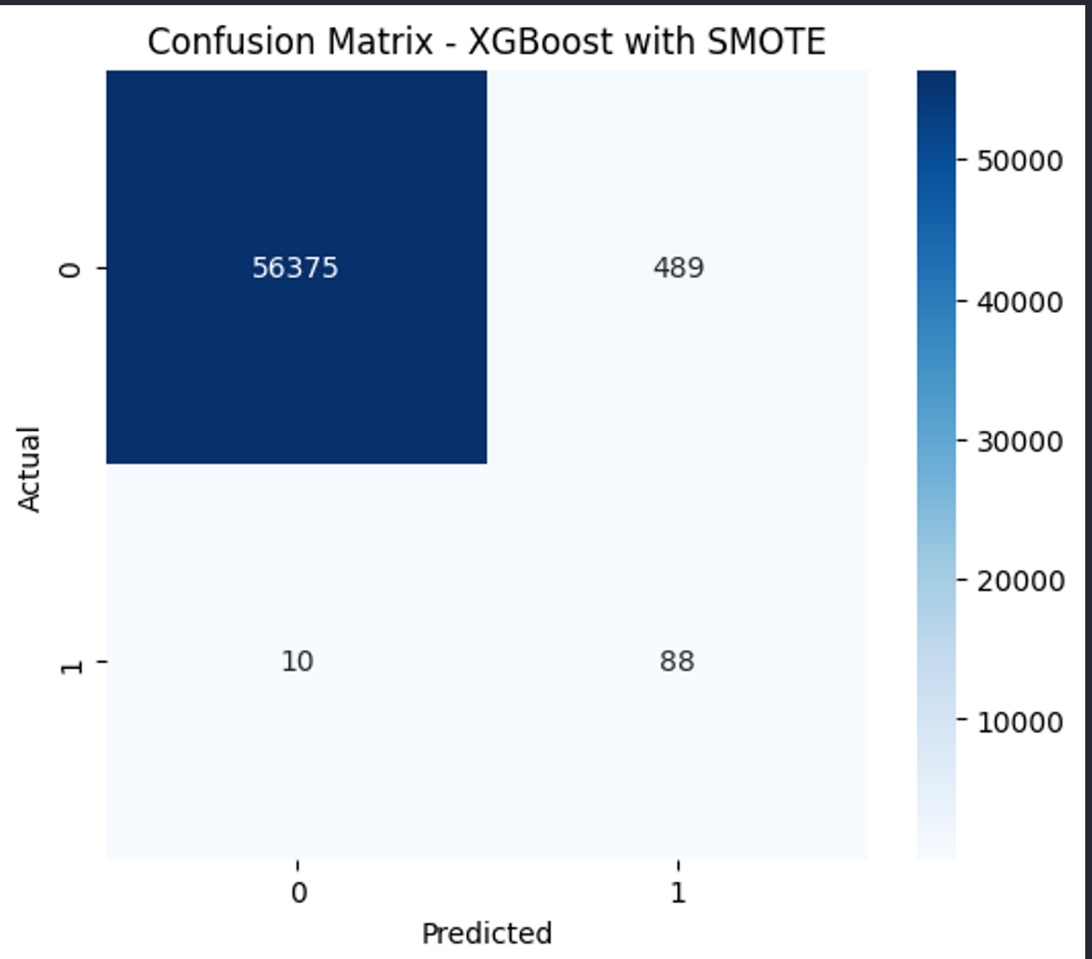
Strengths:

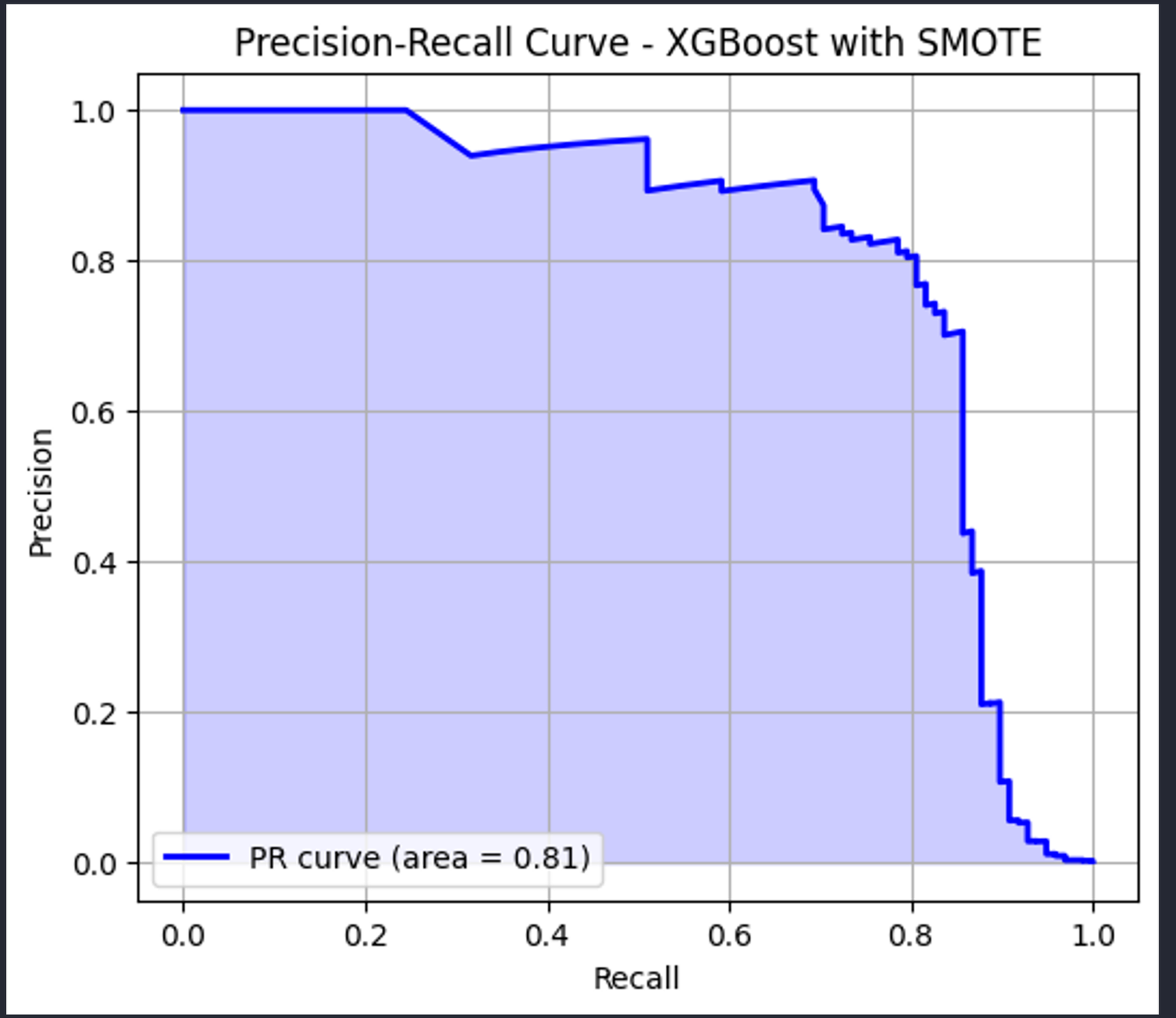
- Excellent handling of class imbalance with scale_pos_weight.
- Fast training and highly regularized to prevent overfitting.
- Most balanced model in terms of precision and recall.

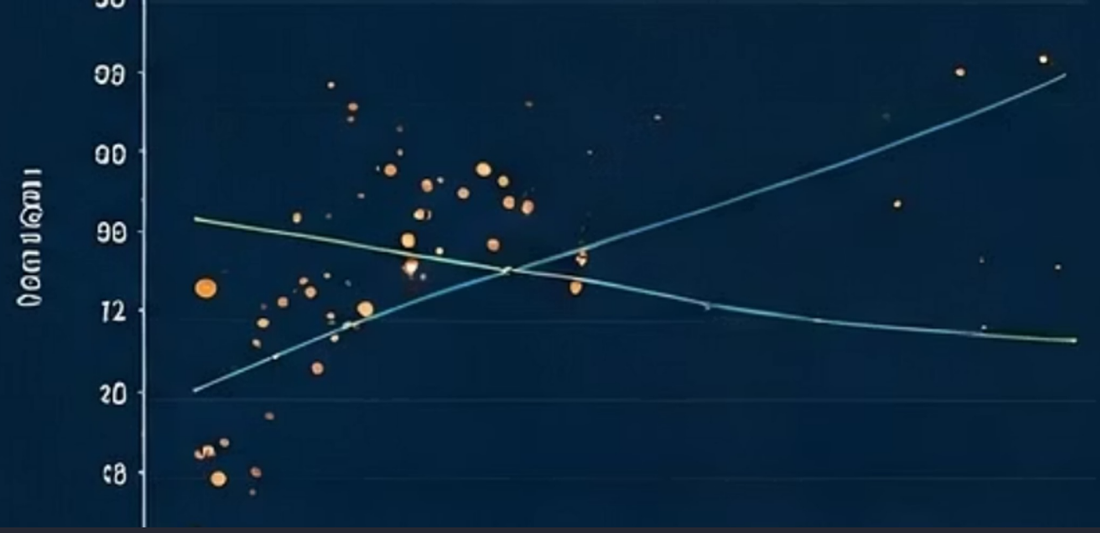
XGBoost with SMOTE Performance Metrics:
Accuracy: 0.9912
Precision: 0.1525
Recall: 0.8980
F1 Score: 0.2607
AUC-ROC Score: 0.9767
Average Precision Score: 0.8088

Classification Report:

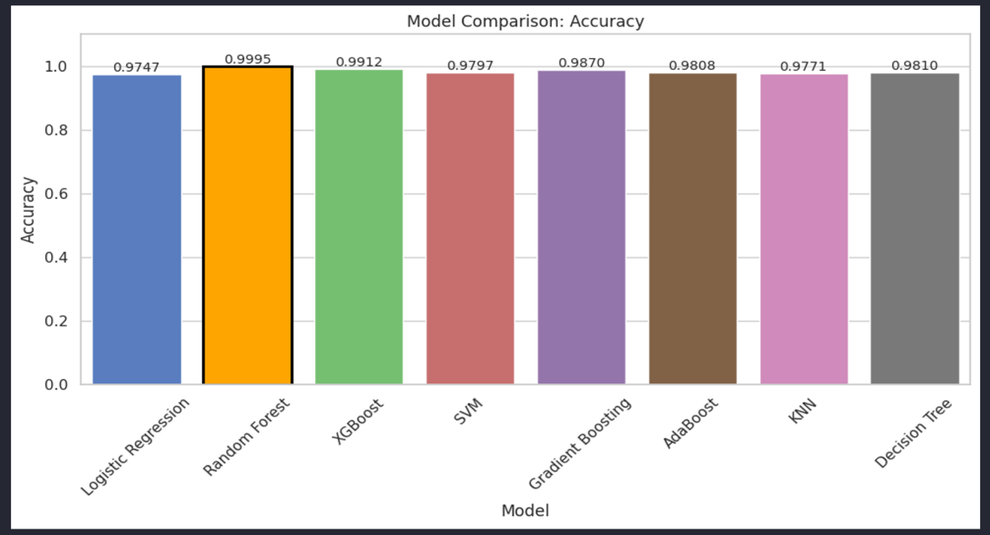
	precision	recall	f1-score	support
0	1.00	0.99	1.00	56864
1	0.15	0.90	0.26	98
accuracy			0.99	56962
macro avg	0.58	0.94	0.63	56962
weighted avg	1.00	0.99	0.99	56962



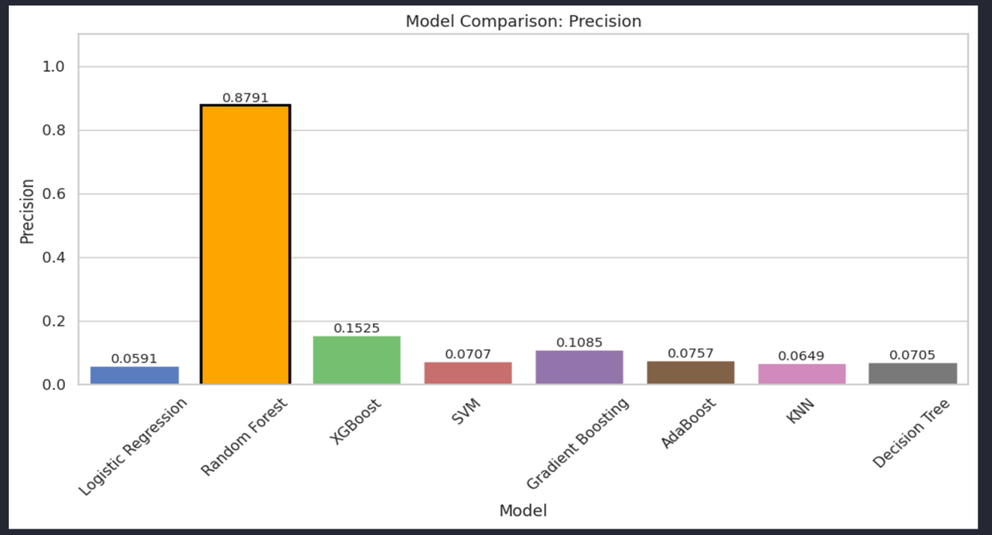

Precision-Recall Curve - XGBoost with SMOTE
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
PR curve (area = 0.81)



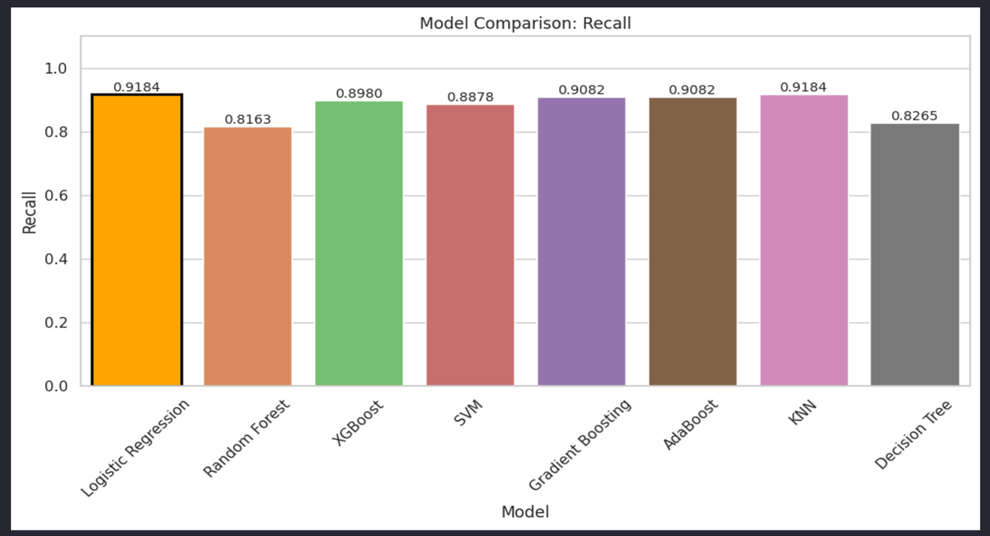
Model Performance Comparison



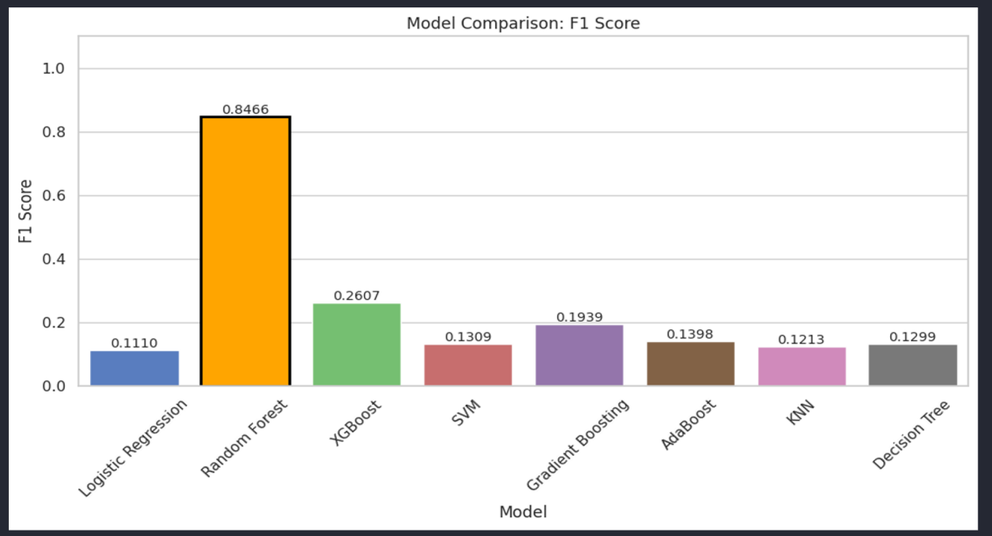
Accuracy



Precision

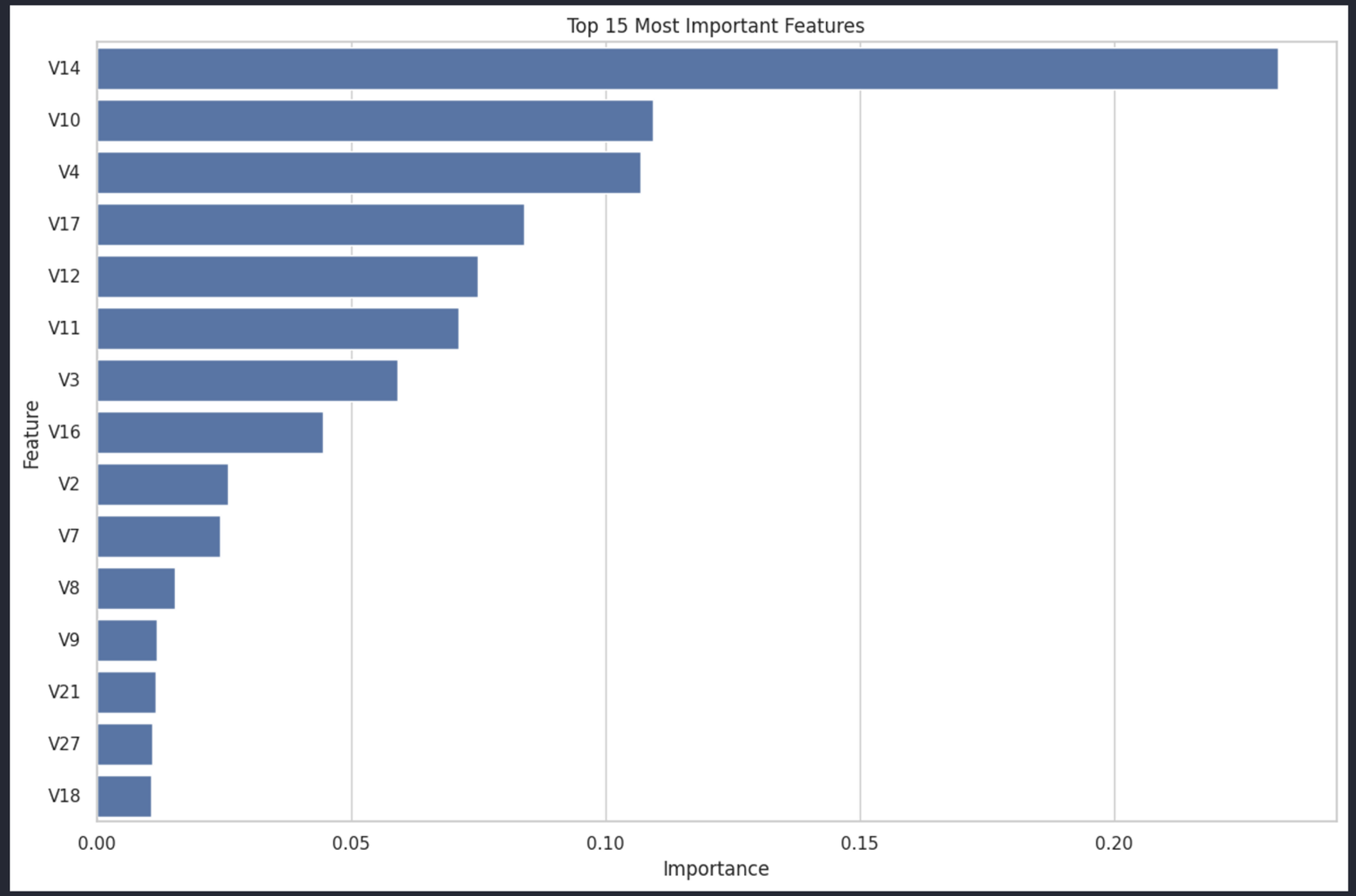


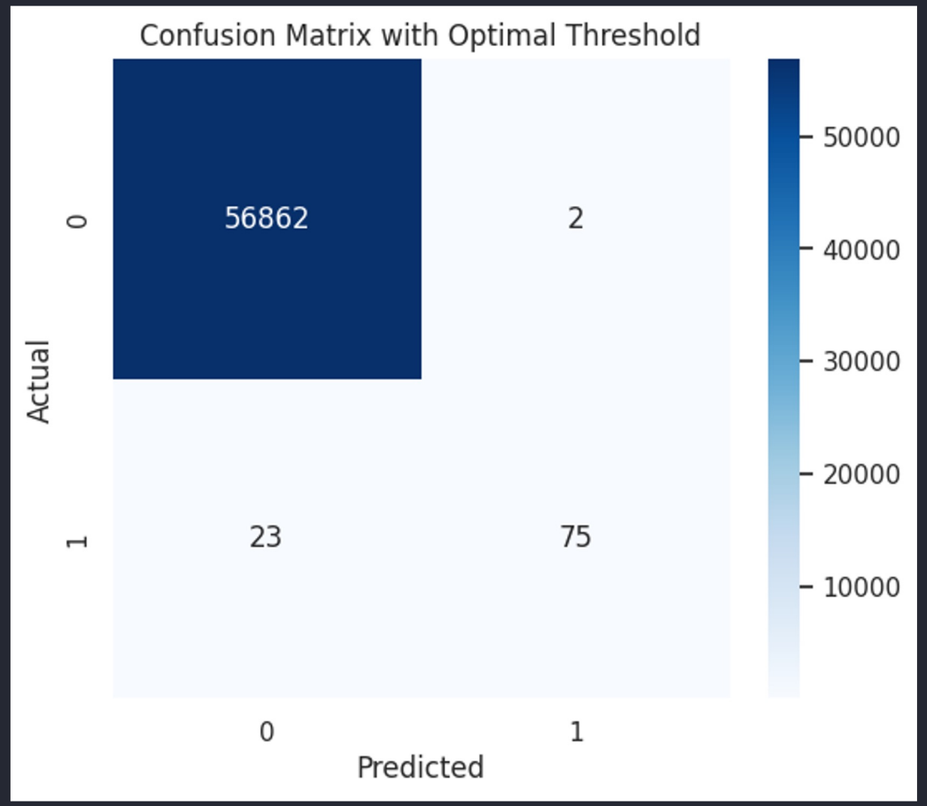
Recall



F1 Scores

Features and Optimal Threshold





Random Forest overall proved to be best suited for fraud detection on SMOTE data.

Thank You