Isolation Forest for Anomaly Detection

Efficient Outlier Detection Using Random Partitioning

Authors: Nicholas Jebeles, Blake Senn, and Isaiah Barlatier

Motivating Questions

Q: Why do we need anomaly detection?

- Fraud detection in financial transactions
- Network intrusion detection
- Manufacturing quality control
- System health monitoring

Q: How can we detect outliers efficiently?

- Traditional methods struggle with large datasets
- Need for unsupervised approaches
- This project demonstrates a solution!

Algorithm Introduction

- **Core Insight**: Anomalies are "few and different"

Key Properties of Anomalies:

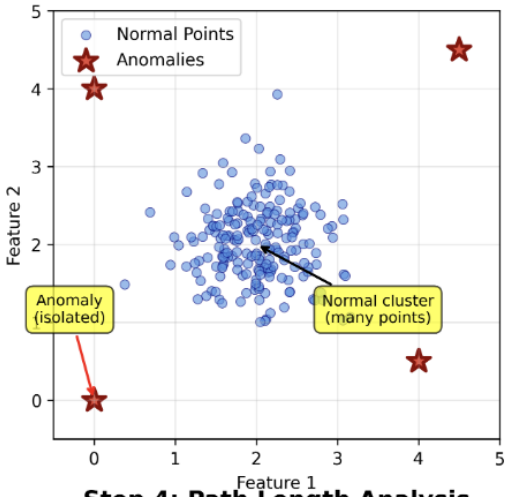
- **Few**: Minority instances in the dataset
- **Different**: Have unusual feature values

The Isolation Forest Approach:

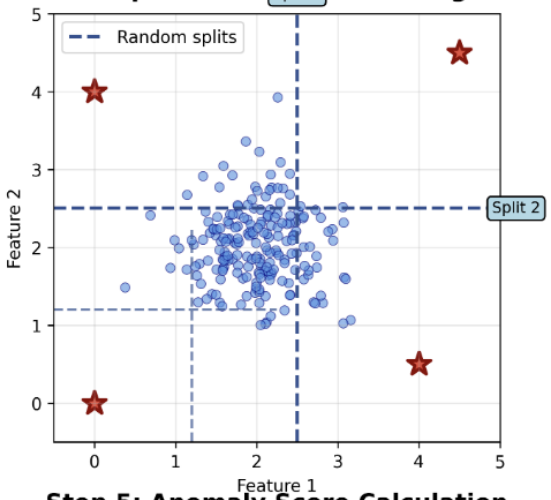
- Uses random partitioning
- Anomalies require fewer splits to isolate
- No distance or density calculations needed

How Isolation Forest Works

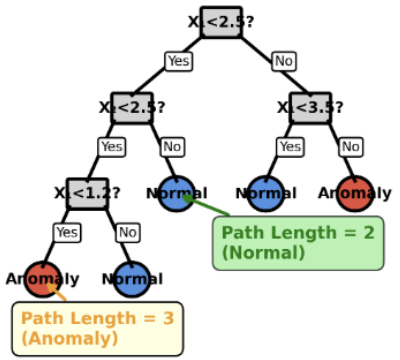
Step 1: Data Distribution



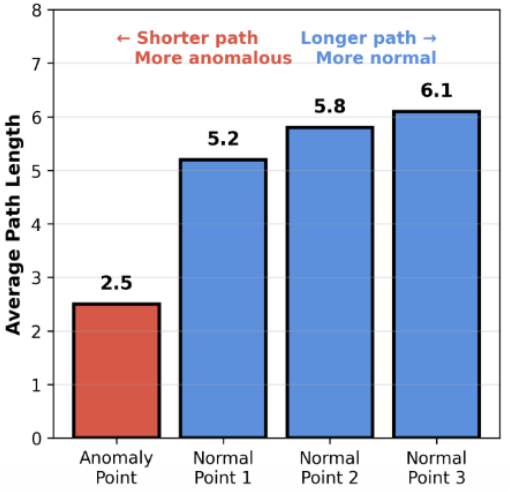
Step 2: Random Partitioning



Step 3: Isolation Tree



Step 4: Path Length Analysis



Step 5: Anomaly Score Calculation

Anomaly Score Formula:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Where:

- E(h(x)) = Average path length
- c(n) = Normalization constant
- n = Number of samples

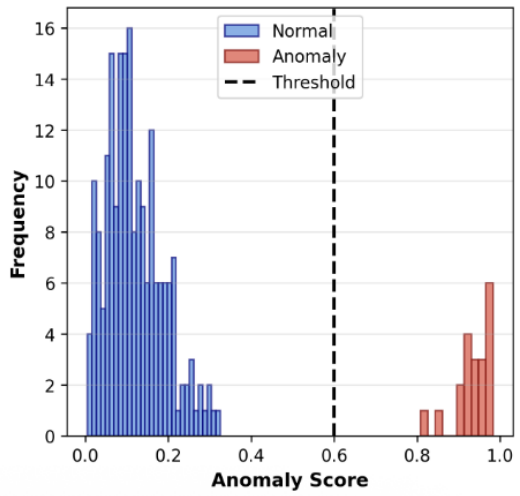
Score Interpretation:

- Score → 1: Anomaly
- Score → 0: Normal
- Score ≈ 0.5: Uncertain

Example Scores:

- **Anomaly: s = 0.85**
- **Normal: s = 0.12**

Step 6: Final Classification



Mathematical Foundation

Path Length Calculation:

$h(x)$ = number of edges from root to isolate point x

$E(h(x))$ = average path length across all trees

Complexity Analysis:

Training: $O(t \times \psi \times \log \psi)$

t = number of trees

ψ = sample size per tree

Prediction: $O(t \times \log \psi)$

Normalization Constant:

$c(n) = 2H(n-1) - 2(n-1)/n$

Where $H(i) = \ln(i) + \gamma$ (Euler's constant ≈ 0.5772)

Anomaly Score:

$s(x, n) = 2^{(-E(h(x)) / c(n))}$

Interpretation:

- $s(x, n) \rightarrow 1$: Definite anomaly
- $s(x, n) \rightarrow 0$: Definitely normal
- $s(x, n) \approx 0.5$: Ambiguous

Dataset & Preprocessing

Dataset: Credit Card Fraud Detection

- 10,000 transactions
- 20 features
- ~3% fraud rate (highly imbalanced)

Preprocessing Steps:

- Feature scaling (StandardScaler)
- Train/test split (70/30)
- No label information used for training

Model Configuration

Key Parameters:

- **n_estimators**: 100 trees
- **contamination**: 0.03 (3% expected outliers)
- **max_samples**: 'auto' (256 samples per tree)
- **random_state**: 42 (reproducibility)

Why These Values?

- 100 trees: Good balance of accuracy and speed
- Contamination matches actual fraud rate
- Max_samples='auto': Efficient for this data size

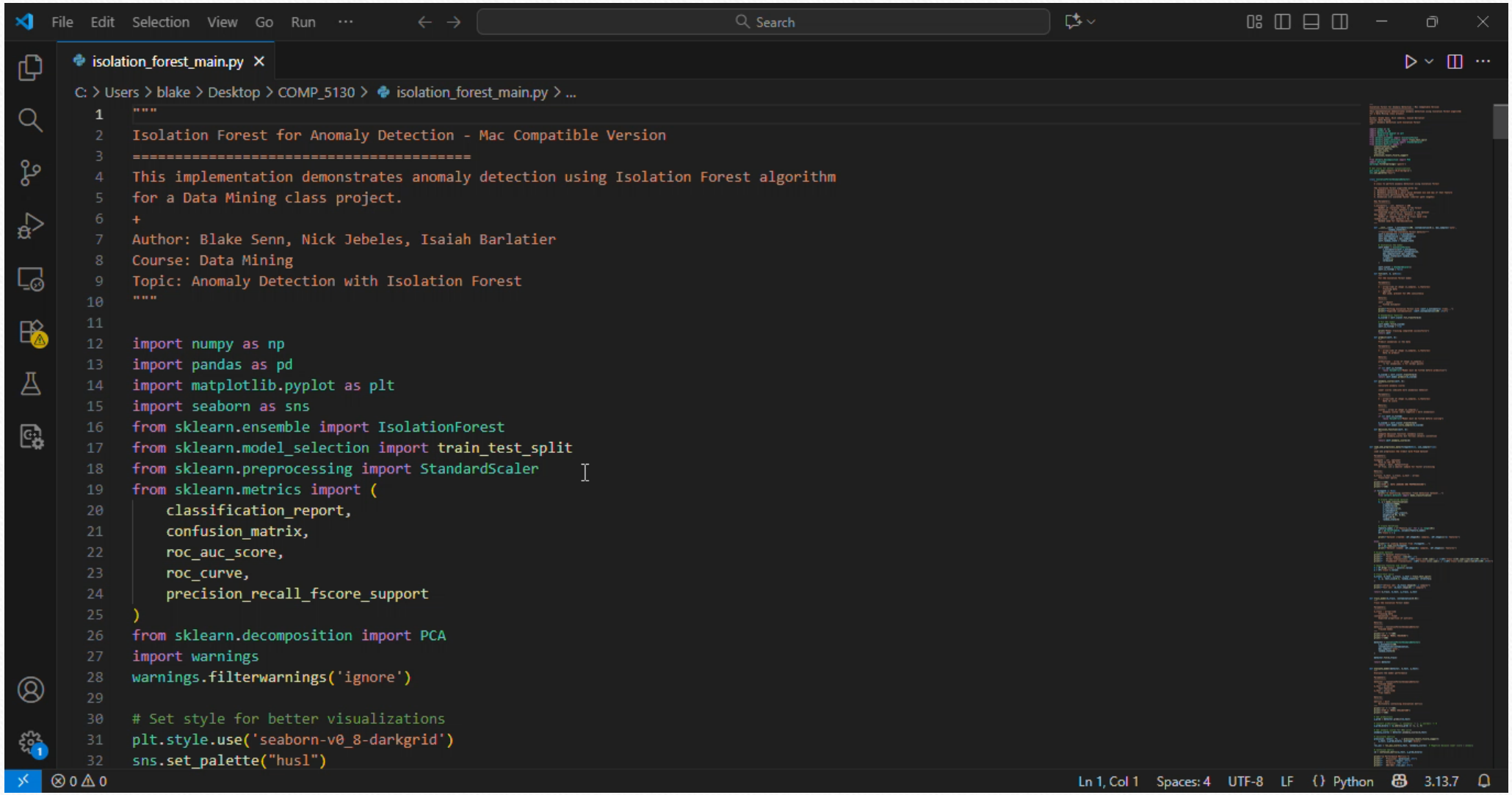
Results - Performance Metrics

Performance:

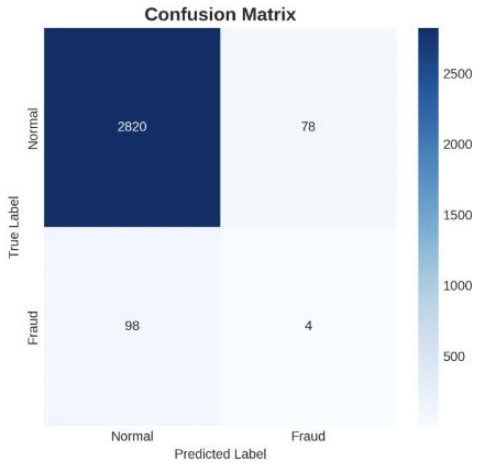
- Precision: 4.9%
- Recall: 3.9%
- F1-Score: 4.3%
- ROC-AUC: 59.2%

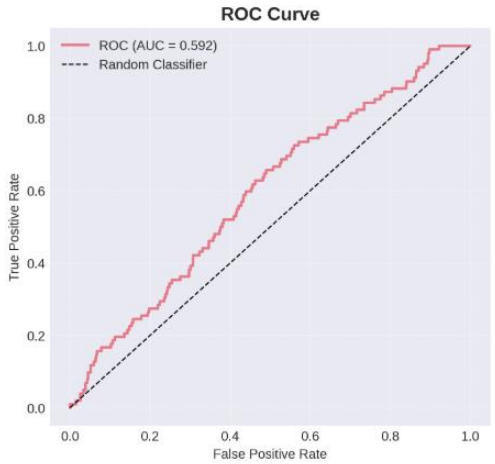
Confusion Matrix

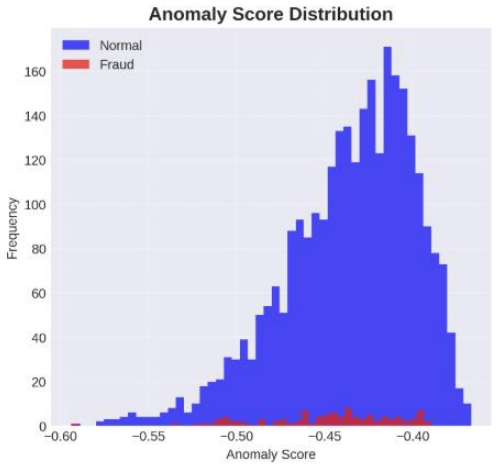
	Predicted Normal	Predicted Fraud
Actual Normal	2,820 (TN)	78 (FP)
Actual Fraud	98 (FN)	4 (TP)

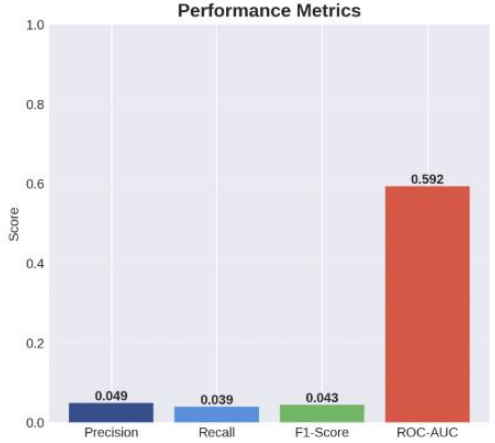

File Edit Selection View Go Run ···
Search
isolation_forest_main.py
C: > Users > blake > Desktop > COMP_5130 > isolation_forest_main.py > ...
"""
Isolation Forest for Anomaly Detection - Mac Compatible Version
==========
This implementation demonstrates anomaly detection using Isolation Forest algorithm
for a Data Mining class project.
+
Author: Blake Senn, Nick Jebeles, Isaiah Barlatier
Course: Data Mining
Topic: Anomaly Detection with Isolation Forest
"""
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import IsolationForest
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
classification_report,
confusion_matrix,
roc_auc_score,
roc_curve,
precision_recall_fscore_support
)
from sklearn.decomposition import PCA
import warnings
warnings.filterwarnings('ignore')
# Set style for better visualizations
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("husl")
0 0
Ln 1, Col 1 Spaces: 4 UTF-8 LF {} Python 3.13.7

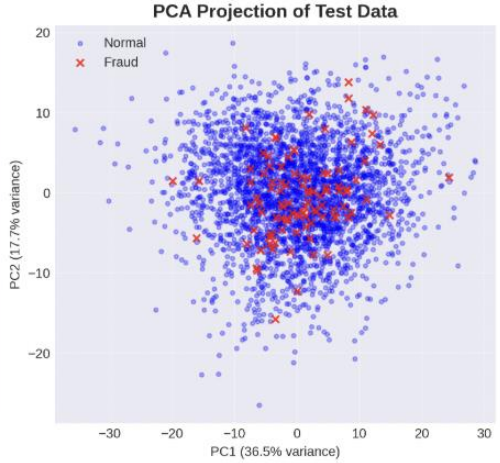
Results - Visualizations

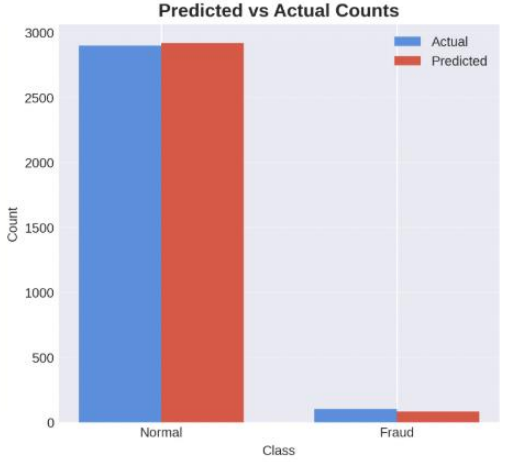












Code Demonstration

```
# Model initialization
detector = IsolationForestAnomalyDetector(
        n_estimators=100,
        contamination=0.03,
        max_samples='auto’
)
# Training (unsupervised)
detector.fit(X_train)
# Prediction
predictions = detector.predict(X_test)
 anomaly_scores = detector.anomaly_scores(X_test)
```

Conclusion & Discussion

- **Key Takeaways:**
- Isolation Forest is efficient and scalable
- No labeled data required
- Strong performance on imbalanced data
- Practical for real-world applications

Limitations:

- May miss local/contextual anomalies
- Requires setting contamination parameter
- Less interpretable than rule-based methods

Future Work:

- Test on multiple datasets
- Compare with other algorithms
- Add feature importance analysis
- Deploy as real-time system