



Credit Card Fraud Detection

Jiban-Ul Azam Chowdhury Shafin

Submitted to: Prof. Dr. Stefan Henkler

Electronic Engineering, Hochschule Hamm-LIPPSTADT



Understanding Credit Card Fraud Detection

Definition

- ❑ Credit card fraud detection identifies unauthorized transactions,
- ❑ which involve misusing credit information or cards without consent.
- ❑ This includes application fraud (false info for new cards) and behavioral fraud (misusing legitimate card details).

Data & Approach

- ✓ Transactional data, with up to a hundred variables per transaction, is logged to build detection models.
- ✓ Fraud detection can be approached as a binary classification problem or an anomaly detection problem, flagging deviations from normal behavior.

Challenges: Imbalance and Cost of Errors

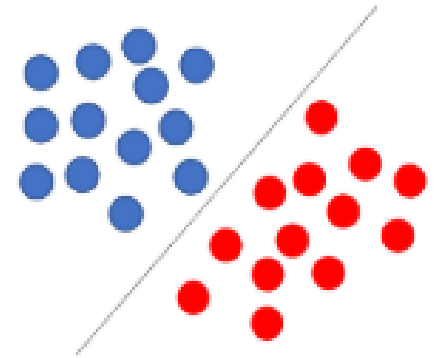
Extreme Class Imbalance

Fraudulent transactions are rare, often less than 0.5% of all transactions. This makes fraud detection a "needle-in-a-haystack" problem, as analytical models struggle to learn patterns of the minority class.

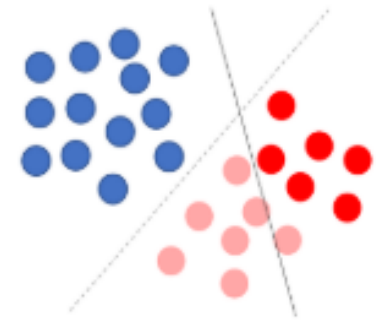
Cost of Errors

False negatives (missed fraud) lead to significant financial loss. False positives (legitimate transactions flagged as fraud) harm customer experience. Cost-sensitive learning assigns higher misclassification costs to critical errors.

$$Totalcost = C(-, +) \times FN + C(+, -) \times FP$$



Classifier with
balanced data



Classifier with
imbalanced data

Isolation Forest Algorithm

1

Core Idea

Isolation Forest (iForest) detects anomalies by explicitly isolating them, rather than profiling normal data. Anomalies are "few and different," making them easier to separate with fewer recursive partitions.

2

Tree Construction

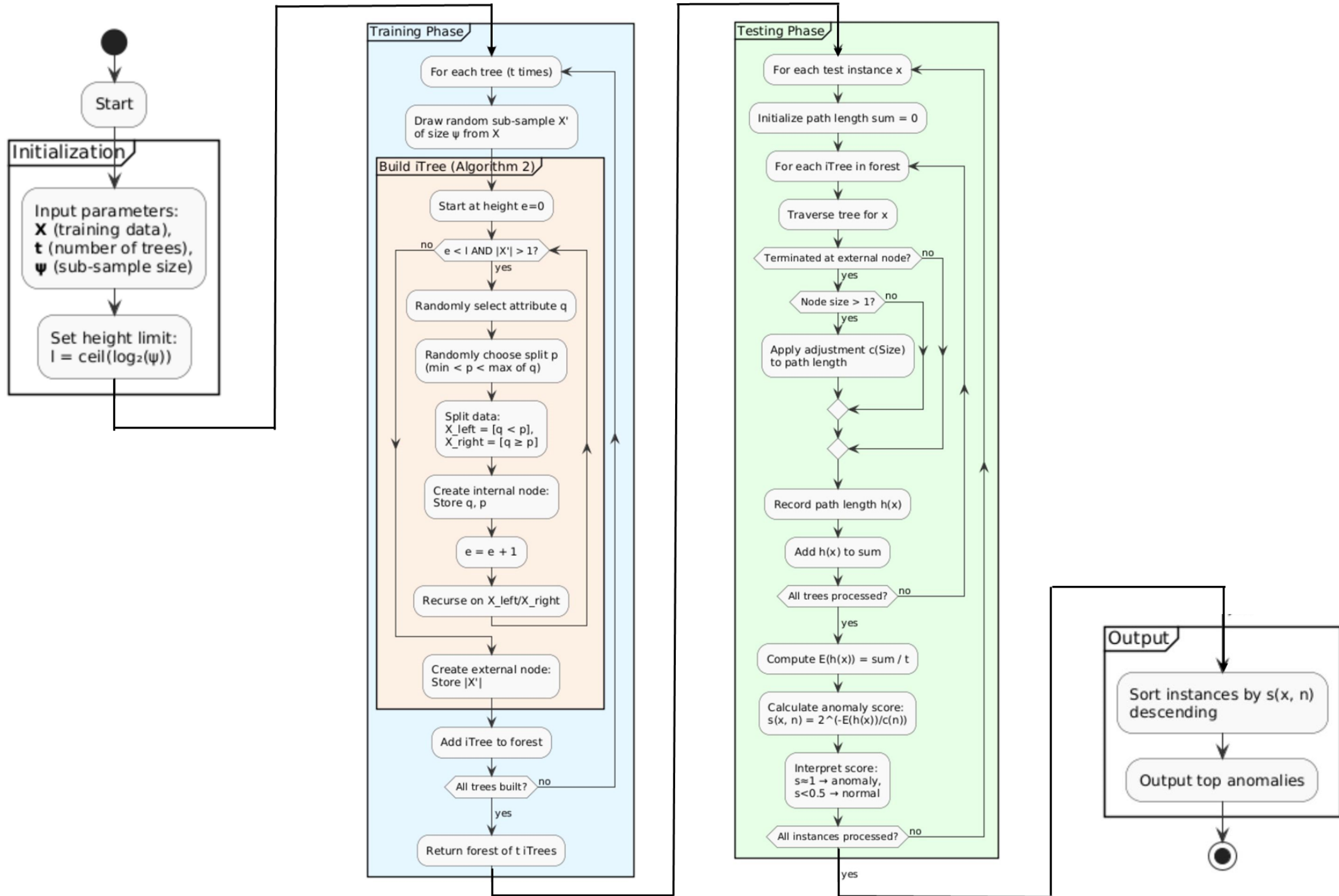
The algorithm builds an ensemble of random Isolation Trees (iTrees). Each tree recursively partitions data by randomly selecting an attribute and a split value, continuing until a stopping condition is met.

3

Anomaly Score

An anomaly score $s(x, n)$ is computed for each point based on its average path length across all iTrees. Shorter path lengths indicate higher anomaly likelihood.

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



Implementation Steps

Data Loading & Exploration

```
import pandas as pd

# Replace path with your correct path
df = pd.read_csv('/content/drive/MyDrive/mldataset/creditcard.csv')

# Check the data
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...
5 rows × 31 columns											
...		V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
...	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
...	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
...	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
...	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
...	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

- ❑ The Credit Card Fraud Detection dataset (284,807 European transactions, 492 fraudulent) was loaded and explored.
- ❑ It includes 28 anonymized principal components (V1-V28), transaction amount, and time.

Preprocessing with StandardScaler

```
[ ] from sklearn.preprocessing import StandardScaler

# Drop 'Time' and 'Class' to create features
features = df.drop(columns=['Time', 'Class'])

# Save 'Class' separately for evaluation
labels = df['Class']

# Apply StandardScaler to features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)

print("Shape of scaled data:", X_scaled.shape)
```

```
➞ Shape of scaled data: (284807, 29)
```

- ❑ The 'Time' and 'Class' columns were removed.
- ❑ Remaining features were standardized using StandardScaler, ensuring they are centered around zero with unit variance,
- ❑ which is vital for Isolation Forest.










Training Isolation Forest

```
from sklearn.ensemble import IsolationForest

# Define Isolation Forest model
iso_forest = IsolationForest(n_estimators=100, max_samples='auto', contamination='auto', random_state=42)

# Fit model on scaled data
iso_forest.fit(X_scaled)
```



IsolationForest ⓘ ?	
▼ Parameters	
 n_estimators	100
 max_samples	'auto'
 contamination	'auto'
 max_features	1.0
 bootstrap	False
 n_jobs	None
 random_state	42
 verbose	0
 warm_start	False



Confusion Matrix:

```
[[274629   9686]
 [    90    402]]
```

Precision: 0.039849325931800156

Recall: 0.8170731707317073

F1 Score: 0.07599243856332703

Accuracy: 0.9656750009655661

Anomaly Scoring & Prediction

```
# Get anomaly scores (higher = more normal, lower = more anomaly)
anomaly_scores = iso_forest.decision_function(X_scaled)

# Print example scores
print("Anomaly scores example:", anomaly_scores[:10])
```



```
Anomaly scores example: [ 0.12540493  0.14272779  0.05345558  0.10818359  0.12636705  0.14446305
 0.13256987 -0.00329639  0.1164912   0.1351488 ]
```



```
# Predict anomalies
y_pred = iso_forest.predict(X_scaled)

# Convert: -1 → 1 (fraud), 1 → 0 (normal)
y_pred = np.where(y_pred == -1, 1, 0)

# Print example predictions
print("Predictions example:", y_pred[:10])
```

```
Predictions example: [0 0 0 0 0 0 0 1 0 0]
```

- ❑ Anomaly scores were computed using the `decision_function()` method, with lower scores indicating anomalies.
- ❑ Predictions were then generated using `predict()`, mapping -1 to fraud (1) and 1 to normal (0).

Model Evaluation Metrics

Confusion Matrix

- 1 Organizes predictions against actual outcomes: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

Accuracy & Error Rate

- 2 Accuracy is the proportion of correct predictions. Error Rate is its complement.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Recall & Precision

Recall captures correctly detected fraud cases. Precision reflects truly fraudulent predicted cases. High Recall is crucial for fraud detection.

- 3
$$Recall = \frac{TP}{TP + FN}$$

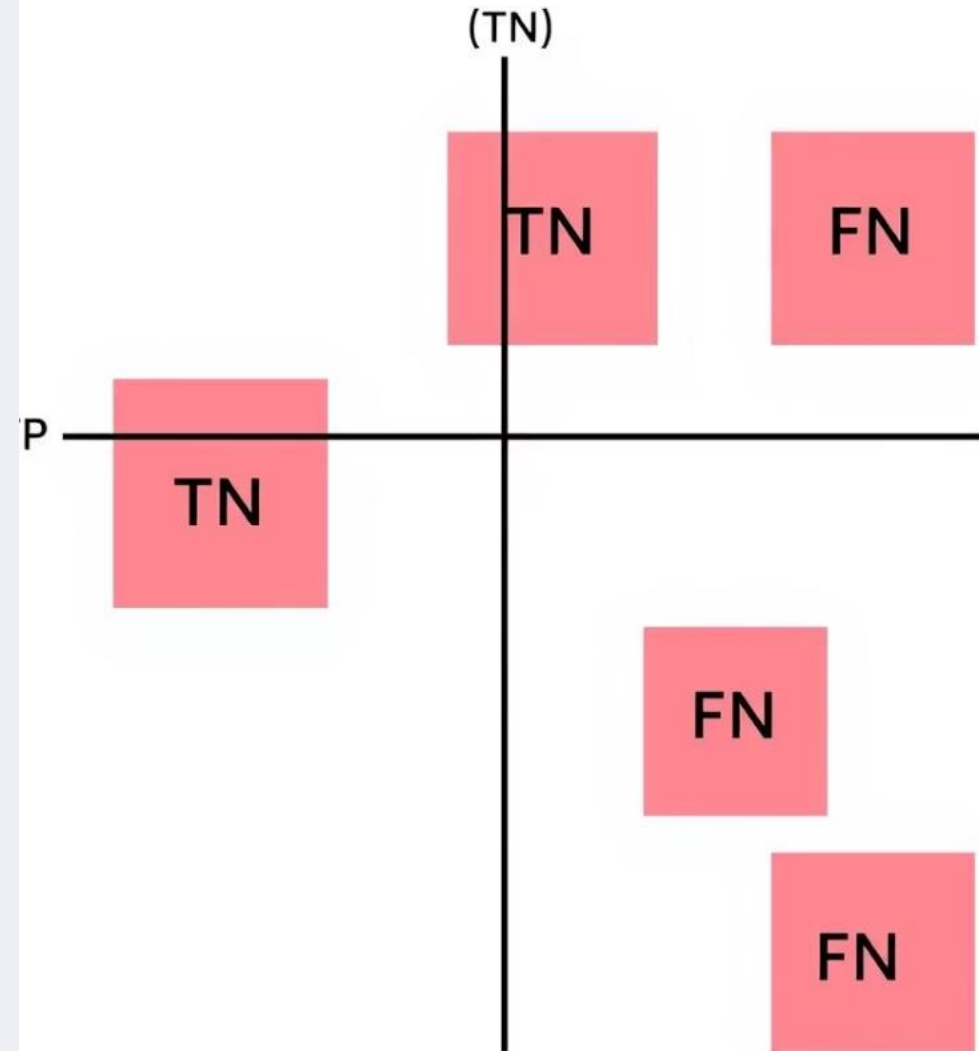
$$Precision = \frac{TP}{TP + FP}$$

F1-Measure & Specificity

- 4 F1-Measure balances Precision and Recall. Specificity measures correct classification of legitimate transactions.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Confusion Matrix



Results and Conclusion

0.17%

Fraud Rate

Only 0.17% of transactions were fraudulent, highlighting the extreme class imbalance challenge.

High

Recall Achieved

The model successfully detected most fraudulent transactions, demonstrating high recall.

Lower

Precision

Precision was lower, a common outcome in highly imbalanced fraud detection problems.

Isolation Forest effectively identifies anomalous transactions without labeled data, proving suitable for large-scale, unsupervised fraud detection. While achieving high recall, precision remains a challenge due to class imbalance. Future work should consider complementary techniques to improve overall performance and manage false positive rates.



Questions?

Thank you for your attention.

I'm happy to answer any questions or feedback you may have.