Problem Statement

SiriusAI

DATA ANOMALY DETECTION SYSTEM

ML + RULE-BASED + GENAI

By :- Parth Tyagi

June 17 , 2025

What is Anomaly Detection?

Anomaly detection is the process of identifying data points, events, or observations that deviate significantly from the normal patterns in a dataset.

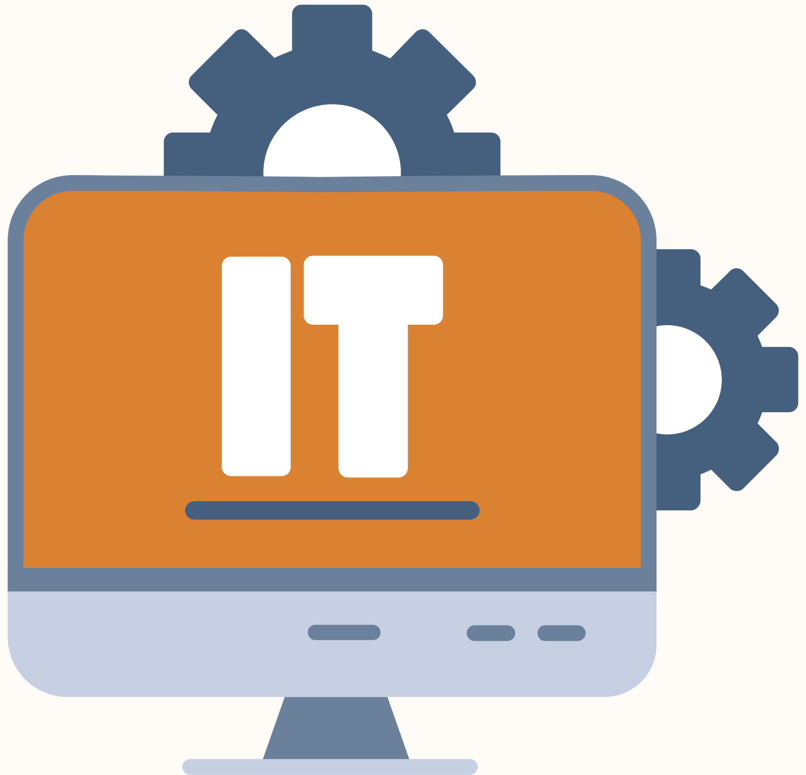
Types of Anomalies		
Type	**Description**	**Example**
Point Anomaly	A single data point is far from the rest	A transaction of $10,000 vs avg $1000
Contextual	An anomaly based on context (e.g., time)	Login at 3 AM from unusual location
Collective	A sequence of events is unusual together	Series of failed logins from same IP

Industry Applications

SiriusAI

User usually spends ₹1000/day → sudden ₹50,000 transaction = anomaly

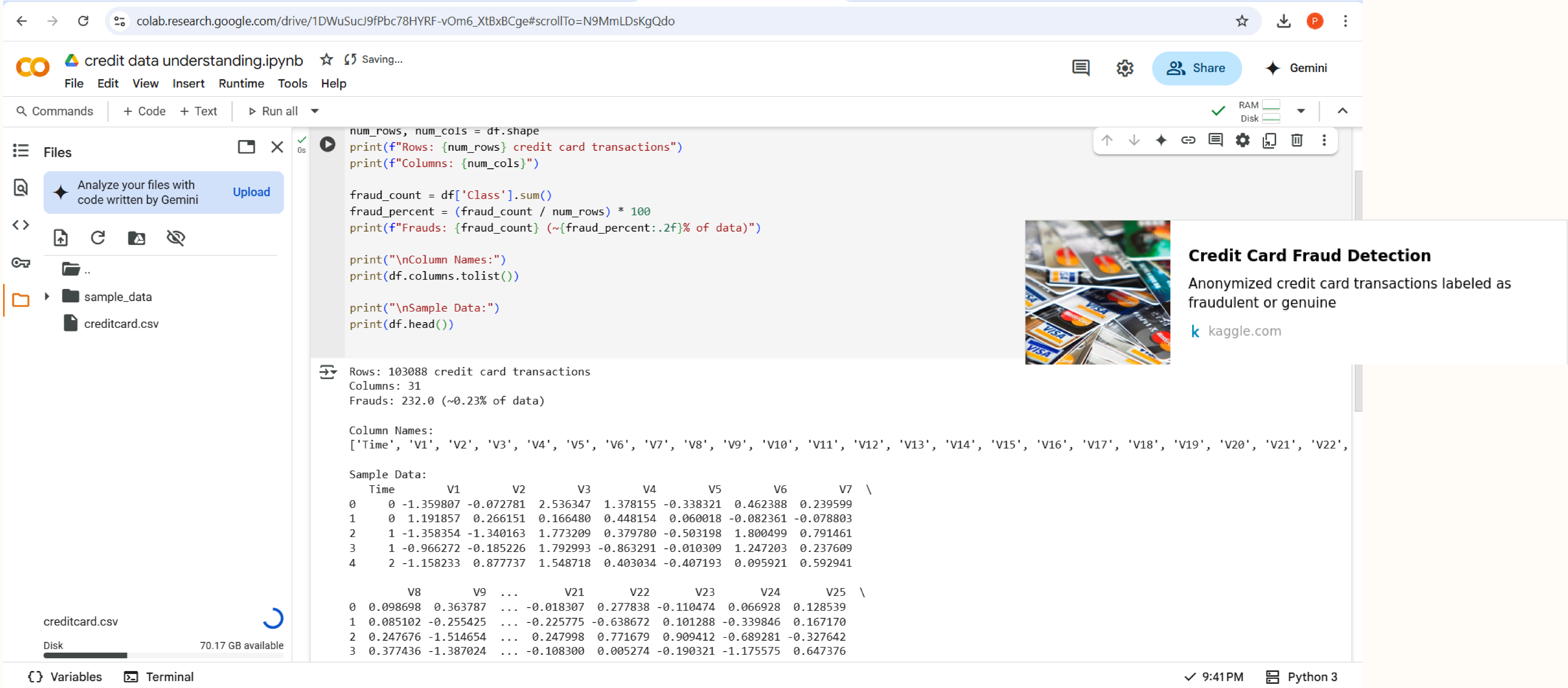
Irregular heart rate or glucose spike in wearable data



Sudden drop in website traffic or spike in 500 errors

A customer ordering the same expensive product 20 times in an hour.

Dataset Description (Credit Card Fraud)



colab.research.google.com/drive/1DWuSucJ9fPbc78HYRF-vOm6_XtBxBCge#scrollTo=N9MmLDsKgQdo
credit data understanding.ipynb
Saving...
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
Files
Analyze your files with code written by Gemini
Upload
..
sample_data
creditcard.csv
creditcard.csv
Disk
70.17 GB available
Variables
Terminal
9:41 PM
Python 3

```python
num_rows, num_cols = df.shape
print(f"Rows: {num_rows} credit card transactions")
print(f"Columns: {num_cols}")

fraud_count = df['Class'].sum()
fraud_percent = (fraud_count / num_rows) * 100
print(f"Frauds: {fraud_count} (~{fraud_percent:.2f}% of data)")

print("\nColumn Names:")
print(df.columns.tolist())

print("\nSample Data:")
print(df.head())
```

```
Rows: 103088 credit card transactions
Columns: 31
Frauds: 232.0 (~0.23% of data)

Column Names:
['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21', 'V22',

Sample Data:
   Time        V1        V2        V3        V4        V5        V6        V7  \
0     0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1     0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2     1 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3     1 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4     2 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9  ...       V21       V22       V23       V24       V25  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128539
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167170
2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647376
```

Credit Card Fraud Detection

Anonymized credit card transactions labeled as fraudulent or genuine

kaggle.com

The features V1 to V28 were generated using PCA (Principal Component Analysis).

PCA is a dimensionality reduction technique that:

Takes correlated input features (like transaction time, merchant ID, etc.)

Converts them into new uncorrelated features (called components)

These components are labeled V1, V2, ..., V28

V14 < -9

That transaction has an extremely unusual pattern in the underlying data feature captured by V14.

A value this low is very rare — it's far from the mean, indicating strong deviation from normal.

Isolation Forest – Concept

What is Isolation Forest?

A machine learning algorithm for unsupervised anomaly detection

Based on a simple idea:

"Anomalies are few and different, so they can be isolated faster."

Anomaly Score

Calculated from average path length over many trees

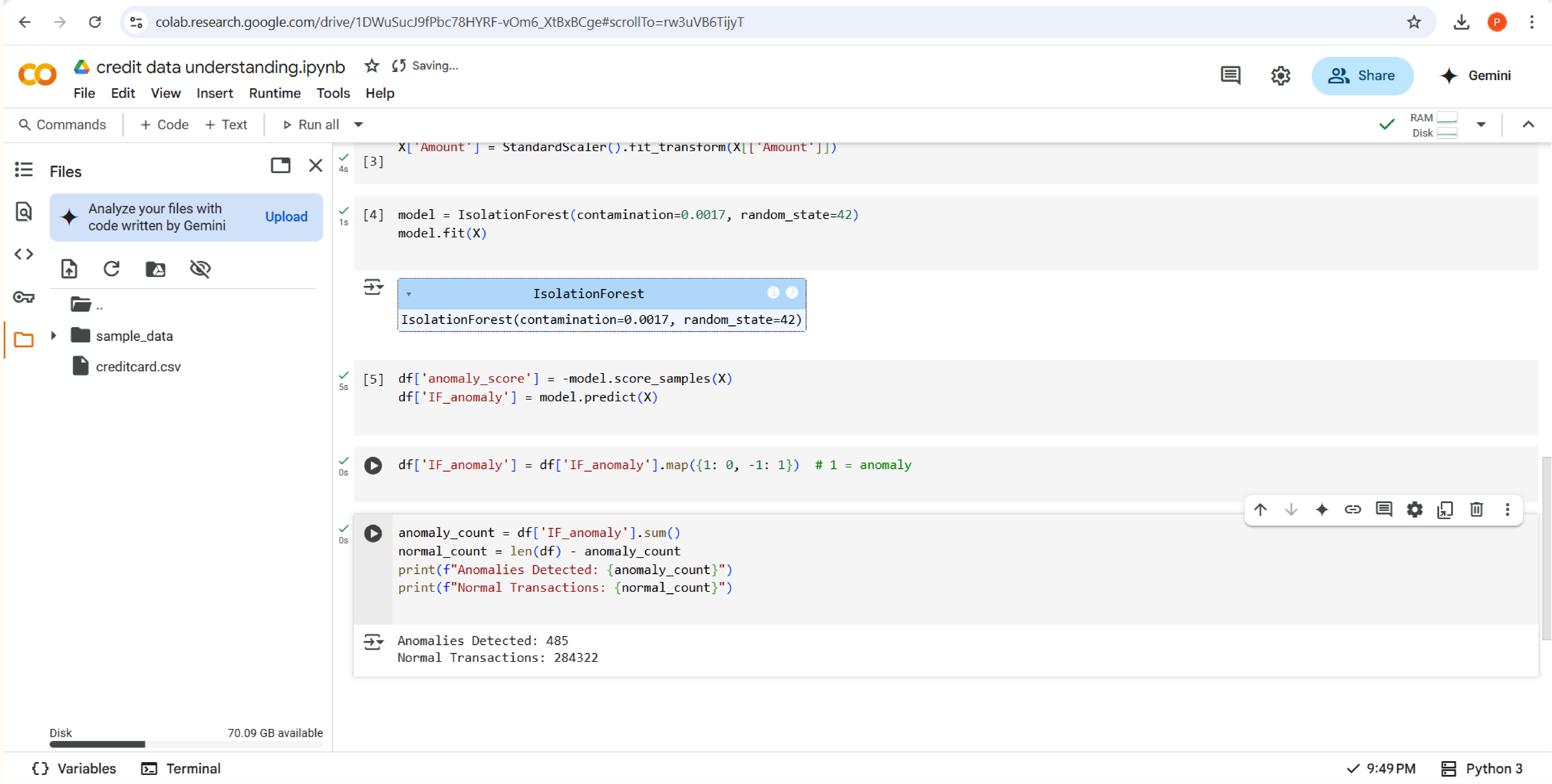
Shorter path = higher anomaly score

Predictions:

-1 = Anomaly

1 = Normal

Isolation Forest – Implementation



```
X['Amount'] = StandardScaler().fit_transform(X[['Amount']])
```

[4]
```
model = IsolationForest(contamination=0.0017, random_state=42)
model.fit(X)
```

```
IsolationForest
IsolationForest(contamination=0.0017, random_state=42)
```

[5]
```
df['anomaly_score'] = -model.score_samples(X)
df['IF_anomaly'] = model.predict(X)
```

```
df['IF_anomaly'] = df['IF_anomaly'].map({1: 0, -1: 1})  # 1 = anomaly
```

```
anomaly_count = df['IF_anomaly'].sum()
normal_count = len(df) - anomaly_count
print(f"Anomalies Detected: {anomaly_count}")
print(f"Normal Transactions: {normal_count}")
```

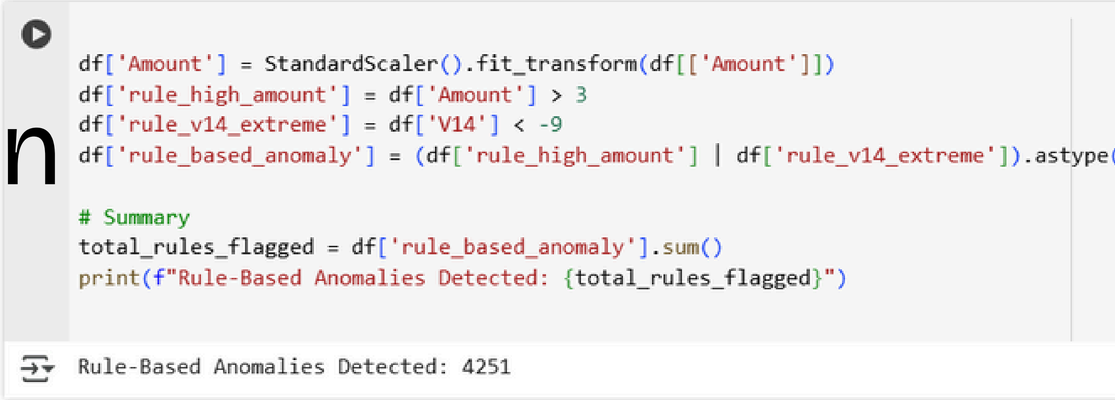
```
Anomalies Detected: 485
Normal Transactions: 284322
```

Rule-Based Detection

What is Rule-Based Anomaly Detection?

A system that uses if–then logic to flag anomalies based on known thresholds or patterns

Created using domain expertise or observed patterns



```
df['Amount'] = StandardScaler().fit_transform(df[['Amount']])
df['rule_high_amount'] = df['Amount'] > 3
df['rule_v14_extreme'] = df['V14'] < -9
df['rule_based_anomaly'] = (df['rule_high_amount'] | df['rule_v14_extreme']).astype(

# Summary
total_rules_flagged = df['rule_based_anomaly'].sum()
print(f"Rule-Based Anomalies Detected: {total_rules_flagged}")
```

```
Rule-Based Anomalies Detected: 4251
```

Rule	Description
IF Amount > 2000 THEN flag	High-value transaction
IF V14 < -9 AND Amount > 1000	Rare pattern from known fraud
IF transactions from same user < 5 sec apart	Possible bot/fraud attack

Adding a GenAI Layer

What is GenAI (Generative AI)?

Uses Large Language Models (LLMs) like GPT-4 to understand and generate patterns

Can read anomaly patterns and suggest human-readable rules

- Helps automate rule discovery from complex datasets
- Explains why a transaction may be suspicious
- Assists analysts by summarizing or validating anomalies

```
import openai
openai.api_key = "your-api-key"
prompt = "Found anomalies with V14 < -9 and Amount >
1000.\nSuggest a rule to detect similar cases."
response = openai.ChatCompletion.create(
    model="gpt-4",
    messages=[{"role": "user", "content": prompt}]
)
print(response.choices[0].message.content)
```

Example Prompt to GPT

We found 5 transactions with:

- V14 < -9

- Amount > 1000

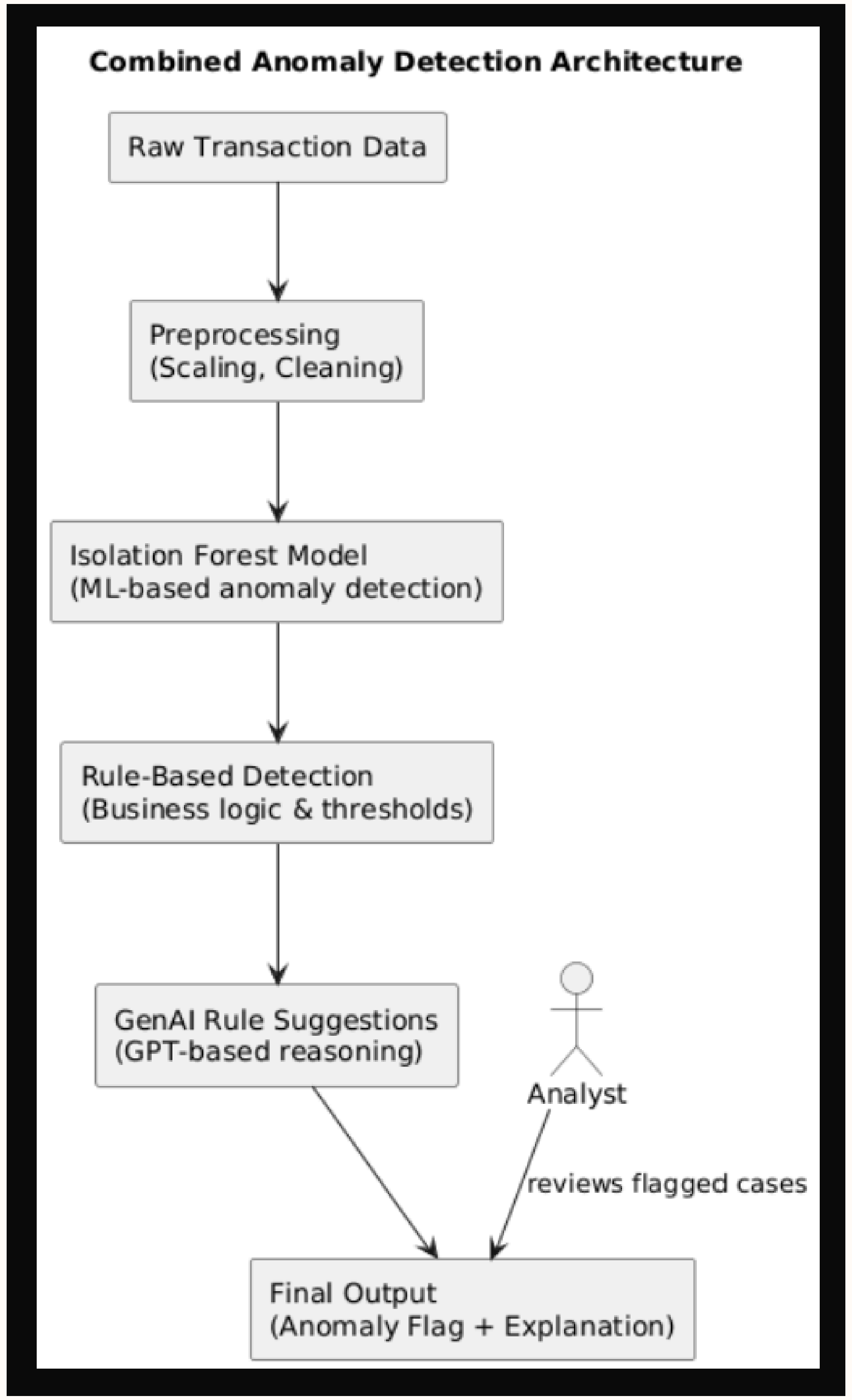
Normal transactions do not follow this pattern.

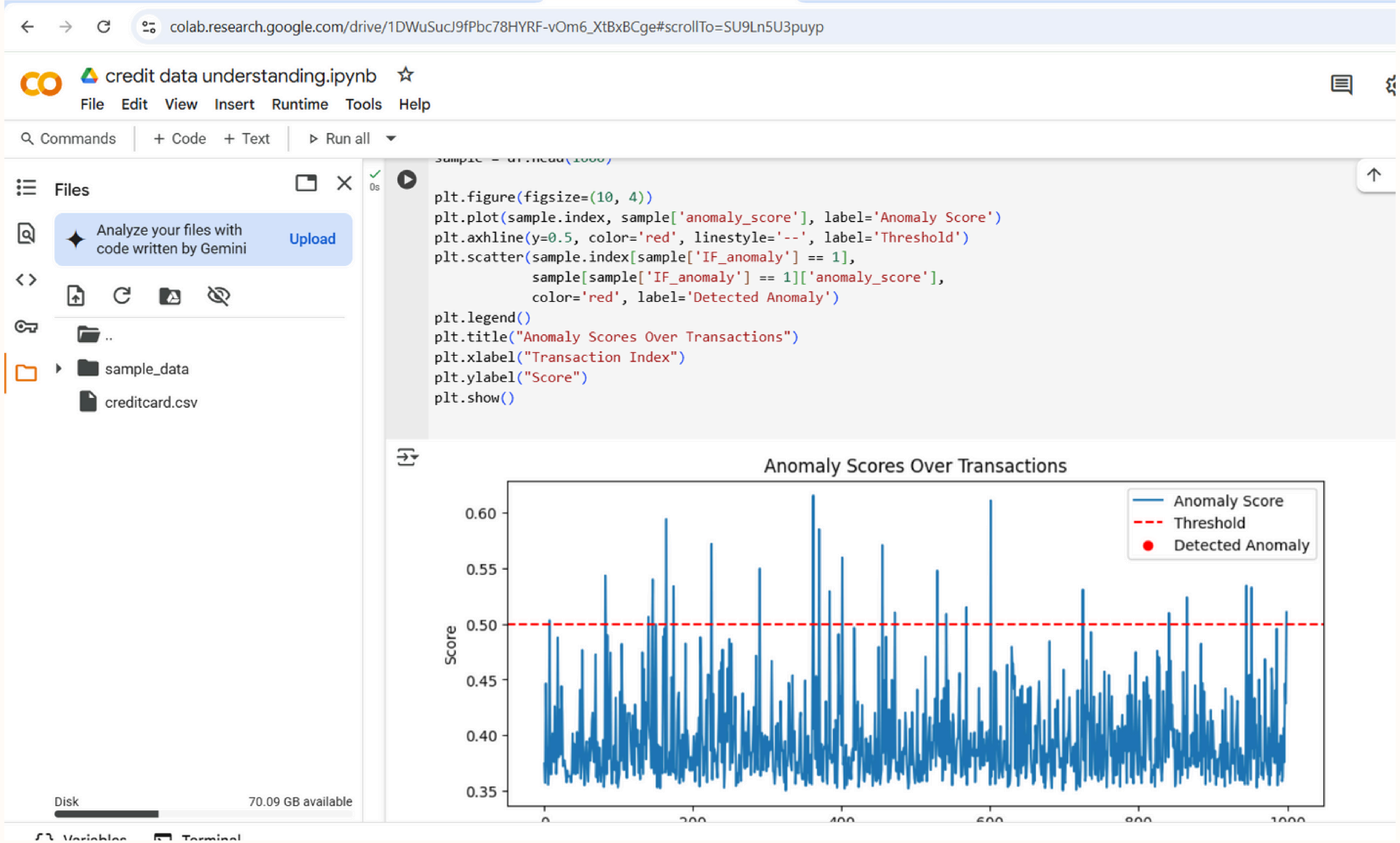
Suggest a rule to detect similar anomalies.

Example Output from GPT:

“Flag transactions where V14 < -9 and Amount > 1000 as potentially fraudulent.”

Combined Architecture


Combined Anomaly Detection Architecture
Raw Transaction Data
Preprocessing (Scaling, Cleaning)
Isolation Forest Model (ML-based anomaly detection)
Rule-Based Detection (Business logic & thresholds)
GenAI Rule Suggestions (GPT-based reasoning)
Analyst
reviews flagged cases
Final Output (Anomaly Flag + Explanation)


colab.research.google.com/drive/1DWuSucJ9fPbc78HYRF-vOm6_XtBxBCge#scrollTo=SU9Ln5U3puyp
credit data understanding.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Commands
+ Code
+ Text
Run all
Files
Analyze your files with code written by Gemini
Upload
..
sample_data
creditcard.csv
Disk
70.09 GB available
Variables
Terminal

```
sample = df.head(1000)

plt.figure(figsize=(10, 4))
plt.plot(sample.index, sample['anomaly_score'], label='Anomaly Score')
plt.axhline(y=0.5, color='red', linestyle='--', label='Threshold')
plt.scatter(sample.index[sample['IF_anomaly'] == 1],
            sample[sample['IF_anomaly'] == 1]['anomaly_score'],
            color='red', label='Detected Anomaly')
plt.legend()
plt.title("Anomaly Scores Over Transactions")
plt.xlabel("Transaction Index")
plt.ylabel("Score")
plt.show()
```

Thank you