Anomaly Detection Lab: Following Along with the Demo

Student Handout

Before We Start

- Open JupyterLab and create a new Python notebook
- Make sure you can access SAS Viya
- Have this handout ready to follow along

What We're Building Today

A real-time fraud detection system that monitors transactions and flags suspicious activity—just like your bank does thousands of times per second.

Code Breakdown: Understanding Each Step

📍 Cell 1: Setting Up Our Tools

```
python

import saspy
import pandas as pd
import numpy as np
import time
from IPython.display import display, clear_output

sas = saspy.SASsession()
```

What's happening here?

- `saspy`: Lets Python talk to SAS (like a translator)
- `pandas`: Handles our data in Python
- `numpy`: Does the math for random number generation
- `time`: Creates pauses for our animation
- `display/clear_output`: Makes the chart update live

Business translation: We're plugging in our monitoring equipment.

📍 Cell 2: Creating "Normal" Baseline Data

```
python

DATA WORK.BASELINE;
   DO time = 1 TO 50;
      amount = RAND('NORMAL', 50, 15);
      OUTPUT;
   END;
RUN;
```

What's happening here?

- Creating 50 fake transactions
- Average transaction: $50
- Standard deviation: $15 (most fall between $35-$65)

Why these numbers?

- $50 = typical coffee shop office order
- $15 spread = natural variation (solo coffee vs. team lunch)

Your turn: What would you change for a gas station? A jewelry store?

📍 Cell 3: Building Our "Normal Detector"

```
python

PROC SHEWHART DATA=WORK.BASELINE;
   IRCHART amount * time /
      NOCHART
      OUTLIMITS=WORK.CONTROL_LIMITS;
RUN;
```

What's happening here?

- `PROC SHEWHART`: SAS's quality control procedure
- `IRCHART`: Individual & Moving Range chart
- `NOCHART`: Don't display yet (just calculate)
- `OUTLIMITS`: Save the boundaries to use later

The magic: SAS calculates three numbers:

- Lower limit: ~$5 (below this = weird)
- Center line: ~$50 (expected average)
- Upper limit: ~$95 (above this = investigate!)

Think about it: Why save limits separately instead of recalculating each time?

📍 Cell 4: The Live Monitoring Loop

```
python

for i in range(1, 21):
   if i == 15:
      new_amount = 300  # THE FRAUD!
   else:
      new_amount = np.random.normal(50, 15)
```

What's happening here?

- Loop 20 times (20 new transactions)
- Transaction #15 is our planted fraud ($300)
- Others are normal (random around $50)

The streaming process:

1. Generate new transaction (Python)
2. Add to dataset (append, don't replace!)
3. Apply existing limits (don't recalculate!)
4. Refresh the chart
5. Wait 1 second (for dramatic effect)

What to Look For During the Demo

Before Transaction #15

- ✅ Points scattered between the lines

- ✅ Natural variation (some high, some low)
- ✅ All within "normal" bounds

At Transaction #15

- 🚨 Point shoots way above upper limit
- 🚨 SAS automatically marks it
- 🚨 Clear visual flag for investigation

After Transaction #15

- ✅ System continues monitoring
- ✅ Limits stay the same (no recalculation)
- ✅ Normal transactions look normal again

Try It Yourself: Modifications

Easy Modification

Change the fraud amount from $300 to $150. Does it still get caught?

Medium Challenge

Add a second anomaly at transaction #10 that's unusually LOW ($5). What happens?

Advanced Extension

Create seasonal variation: Make morning transactions smaller ($30) and afternoon larger ($70).

Key Concepts to Remember

1. Control Limits ≠ Specification Limits

- **Control limits:** Statistical (what's normal?)
- **Specification limits:** Business rules (what's allowed?)
- Example: $200 transaction might be statistically weird but business-acceptable

2. Why Not Just Use Fixed Rules?

Instead of: "Flag anything over $100"

- Different stores have different normals
- Customer patterns change
- Fraudsters learn your fixed rules

3. The Power of Visual Monitoring

- Patterns visible to humans
- Trends before they become problems
- Context for individual alerts

Quick Quiz: Check Your Understanding

1. **Why do we use historical data for baseline?**
   - Think: How else would we know what's "normal"?
2. **What happens if we include the fraud in our baseline calculation?**
   - Think: Would future frauds get detected?
3. **Why animate the chart instead of showing final result?**
   - Think: How do real monitoring systems work?
4. **When would this simple approach fail?**
   - Think: Black Friday, new product launches, system upgrades

Connecting to Your Future Job

If You Work in:

- **Banking:** This is exactly how fraud alerts work
- **E-commerce:** Monitor cart abandonment spikes
- **Healthcare:** Track patient vital signs
- **Marketing:** Detect campaign performance issues
- **Operations:** Catch supply chain disruptions

Skills You're Building:

- ✅ Statistical process control
- ✅ Real-time data processing
- ✅ Threshold-based alerting
- ✅ Visual analytics design

After Class: Keep Learning

Try These Datasets:

1. Your credit card statement (look for your anomalies)
2. Stock prices (market crashes are anomalies)
3. Weather data (find unusual temperatures)

Google These Terms:

- "Western Electric Rules" (advanced pattern detection)
- "CUSUM charts" (detect small shifts)
- "Isolation Forest" (ML approach we'll cover next week)

Challenge Question:

Your coffee shop has happy hour (3-5 PM) with 50% off. How do you prevent false anomaly alerts during this time without missing actual fraud?

Remember: You're Not Just Learning Statistics

You're learning to protect businesses, catch problems early, and make systems smarter. Every major company uses these techniques. Now you know how they work!

Next week: We'll add machine learning to catch clever fraudsters who stay within "normal" amounts but have weird patterns.