CxC 2025

RiskOps PathFinder for Federato

By Shami-uz Zaman, Tony Ngo, Roah Cho, Angad Ahluwalia

Table of Contents

1. Introduction

1.1 Challenge Overview

Federato’s RiskOps Platform generates a large volume of event-based data capturing how underwriters interact with the system. The challenge is to use this data to understand user behavior, identify which actions drive engagement and retention, and ultimately recommend the next best actions to maximize session length.

1.2 Objectives and Goals

- **Understand User Behavior:** Determine which features or actions have the highest impact on keeping users engaged.
- **Enhance Retention:** Identify patterns and sequences that lead to higher retention rates.
- **Develop a Recommendation Framework:** Build models that suggest the next action for users to keep them on the platform longer.
- **Deliver Actionable Insights:** Translate data findings into concrete recommendations for platform improvements.

---

2. Data Overview

2.1 Key Data Columns and Their Descriptions

Column	Description
insert_id	Unique identifier for each event
amplitude_id	Platform-specific identifier
event_id	Numeric identifier for each event
user_id	Unique user identifier
session_id	Identifier grouping events into sessions
event_time	Timestamp when the event occurred
client_event_time	Time when the event was recorded on the client
event_type	Type of user interaction (e.g., page view, click)
event_properties	Additional details about the event (as a string)
device_type, country, etc.	Device and location context for segmentation

2.2 Data Context

The dataset provides:

- **User Context:** Demographic information and user properties.
- **Session Context:** Information on session duration, frequency, and action counts.
- **Event Context:** Detailed records of each interaction.
- **Device/Location Context:** Used to tailor the platform experience based on device type or geography.

---

3. Data Ingestion & Cleaning

3.1 Data Loading

Data was loaded from multiple CSV chunks stored in the "2025_csv" folder and then concatenated.

Example Output: Combined DF shape: (1,850,109, 30)

```
import pandas as pd
import glob

# Load CSV files from a directory
csv_files = glob.glob("2025_csv/*.csv")
dataframes = [pd.read_csv(file) for file in csv_files]

# Combine all data
data = pd.concat(dataframes, ignore_index=True)
print("Data Loaded. Shape:", data.shape)
```

3.2 Data Inspection

Columns were reviewed, and descriptive statistics were computed to understand the dataset's structure.

```
# Display column names and data types
print(data.info())

# Display summary statistics
print(data.describe())

# Check for missing values
print(data.isnull().sum())
```

3.3 Data Cleaning & Feature Engineering

3.3.1 Handling Missing Values

- **Action:** Fill missing categorical values (e.g., country, region, city) with "Unknown".

- **Result:** Ensured no empty values in key segments.

```
# Fill missing categorical values with 'Unknown'
data.fillna({"country": "Unknown", "region": "Unknown", "city":
"Unknown"}, inplace=True)
```

3.3.2 Timestamp Conversions

- **Action:** Convert columns like `event_time` to datetime objects.
- **Result:** Enabled proper time-based calculations.

```
# Convert timestamps to datetime format
data["event_time"] = pd.to_datetime(data["event_time"])
data["client_event_time"] = pd.to_datetime(data["client_event_time"])
```

3.3.3 Session Aggregation and Derived Metrics

- **Session Grouping:** Events were grouped by `session_id` to derive session start and end times and count the number of actions.
- **Derived Metrics:** Session duration (in minutes) and time since the last login were computed.

```
# Compute session duration
session_data = data.groupby("session_id").agg(
    session_start=("client_event_time", "min"),
    session_end=("client_event_time", "max"),
    action_count=("event_id", "count")
)
session_data["session_duration"] = (session_data["session_end"] -
session_data["session_start"]).dt.total_seconds() / 60
```

4. Exploratory Data Analysis (EDA)

4.1 Descriptive Statistics

A summary of the data (min, max, mean, quartiles) was computed. Key outputs include:

Statistic	Event Time	Event ID
Minimum	2025-01-01 01:03:57.37	0
25th Percentile	2025-01-08 21:29:16.43	7,033
Median	2025-01-15 14:57:45.77	17,507
75th	2025-01-22 17:26:52.41	38,027

Percentile		
Maximum	2025-01-28 21:00:00.16	197,743

```
# Basic statistics
print(data.describe(include="all"))
```

4.2 Visualizations

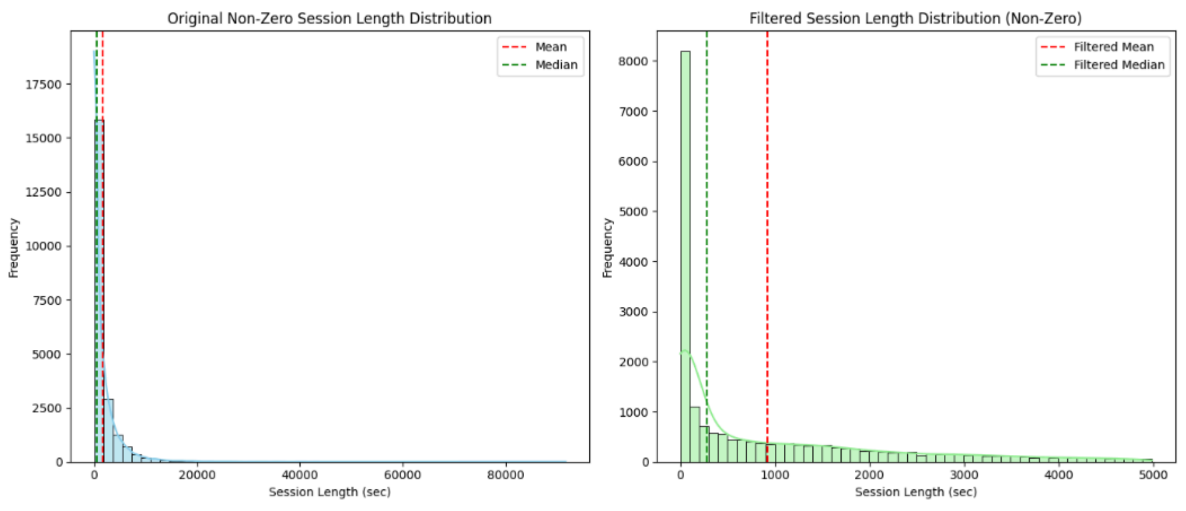
4.2.1 Session Duration Histograms

Histograms of session durations (in seconds) were created. Outlier handling (removing zero durations and applying the IQR method) resulted in:

Metric	Original (sec)	Filtered (sec)
Average Session Length	~1049	~918
Median Session Length	~13	~281

```
import matplotlib.pyplot as plt

plt.hist(session_data["session_duration"], bins=50, edgecolor="black")
plt.xlabel("Session Duration (minutes)")
plt.ylabel("Frequency")
plt.title("Distribution of Session Durations")
plt.show()
```

Note: The extreme difference between mean and median indicates skewness due to a few very long sessions.



4.2.2 Heatmaps and Co-Occurrence Plots

Daily and weekly trends were plotted to reveal temporal patterns in user activity.

```python
import seaborn as sns

# Compute correlation matrix
corr_matrix = data.corr()

# Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Feature Correlation Heatmap")
plt.show()
```

4.3 Engagement and Retention Analysis

4.3.1 User and Session Metrics

A retention flag was defined: users with any event ≥7 days after their first event were marked as "retained." Additional user metrics included:

- **Total Events per User**
- **Distinct Features Used per User**
- **Average and Total Session Duration**

4.3.2 Segment Comparisons

These metrics can be further segmented (by device, location, etc.) to tailor the user experience. For example, comparing average session durations across different regions can inform regional engagement strategies.

---

5. Optimization Framework for Next-Action Recommendations

5.1 Predictive Modeling

5.1.1 Sequence Modeling and Feature Extraction

- **Session Sequences:** The sequence of event types for each session was extracted.
- **Last N Actions:** The last three actions in each session were identified.
- **User Role Extraction:** User roles were optionally extracted from `user_properties` to add additional context.

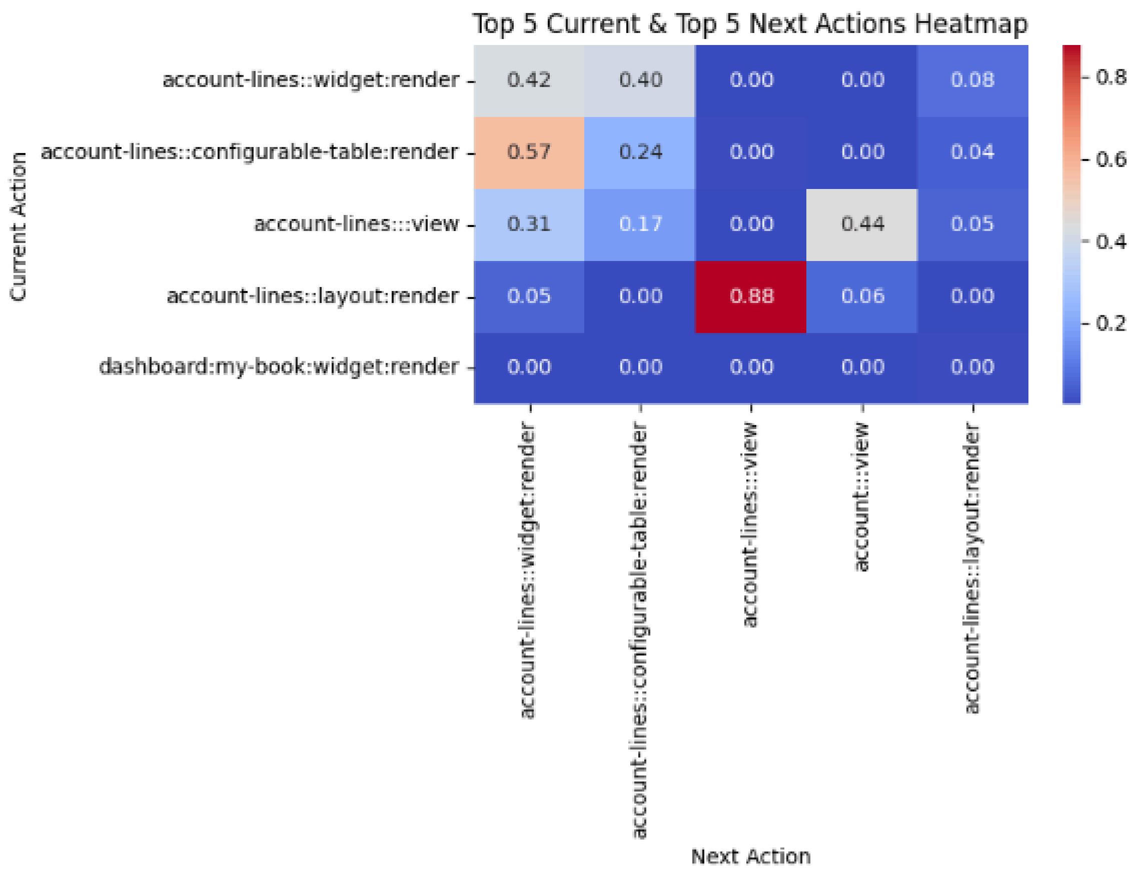
5.1.2 Model Approaches (Markov Chains)

A Markov chain model was built by counting transitions between consecutive event types and converting these counts into probabilities.

```
import numpy as np

actions = ["login", "view", "click", "logout"]
transition_matrix = np.array([
    [0.1, 0.6, 0.2, 0.1],
    [0.2, 0.2, 0.5, 0.1],
    [0.3, 0.3, 0.1, 0.3],
    [0.0, 0.0, 0.0, 1.0]
])

def recommend_next_action(current_action, actions, matrix):
    idx = actions.index(current_action)
    return actions[np.argmax(matrix[idx])]

print("Next action after 'login':", recommend_next_action("login",
actions, transition_matrix))
```

5.1.3 Model Evaluation and Metrics

A full transition probability heatmap was produced. To simplify interpretation, we filtered the matrix to include only the top 5 current actions and top 5 next actions, resulting in a 5×5 heatmap. For example:



Current Action	account-lines::widget:render	account-lines::configurable-table:render	account-lines:::view	account:::view	account-lines::layout:render
account-lines::widget:render	0.42	0.40	0.00	0.00	0.08
account-lines::configurable-table:render	0.57	0.24	0.00	0.00	0.04
account-lines:::view	0.31	0.17	0.00	0.44	0.05
account-lines::layout:render	0.05	0.00	0.88	0.06	0.00
dashboard:my-book:widget:render	0.00	0.00	0.00	0.00	0.00

5.2 Graph-Based Analysis

5.2.1 Graph Construction (Nodes & Edges)

A directed graph was built using a combination of `event_type` and `event_properties` (formatted as "event_type | event_properties"). This graph represents the sequence in which users interact with features.

5.2.2 Network Metrics (Betweenness Centrality, PageRank)

Key network metrics were computed to identify “power nodes.” For instance, the node corresponding to "account-lines::widget:render" consistently ranked high in both betweenness centrality and PageRank, suggesting it is a crucial transition point.

5.2.3 Sankey Diagram

A Sankey diagram was created (using a smaller sample) to visually map the user journey between key actions. This diagram helps identify the flow of user interactions and highlights areas where users drop off or repeatedly cycle through certain features.

5.3 Causal Inference

5.3.1 Propensity Score Matching

We defined a treatment variable for sessions where the `event_properties` equal "AdvancedAnalytics." Logistic regression was used to compute propensity scores based on session duration and action count. The resulting distributions (visualized via KDE plots and box plots) indicate that sessions with "AdvancedAnalytics" may have longer durations.

5.3.2 Difference-in-Differences Approach

While not fully implemented here, a difference-in-differences approach would compare engagement metrics before and after a feature rollout to determine the causal impact.

---

6. Advanced Challenge: User Journey Mapping

6.1 Data Preparation and Sequence Extraction

Session sequences are extracted and enhanced with user role information, allowing us to see how different user segments navigate the platform.

```python
import networkx as nx
import matplotlib.pyplot as plt

G = nx.DiGraph()
G.add_edges_from([("login", "view"), ("view", "click"), ("click",
"logout")])
```

```
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_size=2000, node_color="skyblue")
plt.title("User Journey Graph")
plt.show()
```

6.2 Visualization Techniques (Sankey Diagrams)

A Sankey diagram (constructed from transition data) provides a visual map of the flow between actions. This helps pinpoint bottlenecks and opportunities for intervention.

6.3 Segment Analysis by User Role and Company Type

The framework is set up to further segment journeys by user role, enabling targeted recommendations based on user experience levels.

---

7. Actionable Insights & Recommendations

7.1 Real-Time Next-Action Suggestions

- **Trigger Points:** Analysis of session sequences and transition probabilities reveals common flows. For example, a high probability exists for users to transition from "account-lines::widget:render" to "account-lines::configurable-table:render."
- **Next-Action Engine:** A recommendation model can leverage these patterns to suggest the next action. For instance, when a user completes an "account-lines::widget:render" action, the system can prompt: “Would you like to view the configurable table next?”

7.2 Engagement Strategies and Feature Adoption

- **In-App Prompts:** Tailored prompts based on user role and previous interactions can

guide users to high-value features.
- **Gamification:** Introducing rewards or badges for exploring key features can motivate users.
- **Causal Impact:** Causal inference results support investing in features like "AdvancedAnalytics" since they are associated with longer session durations.

---

8. Implementation Details & Coding Workflow

8.1 Project Setup and Folder Structure

Data is stored in structured folders (raw data, processed data, notebooks, scripts, models, visualizations, reports) to streamline analysis and deployment.

/raw_data   # Original CSV/JSON files

/processed_data # Cleaned and combined data

/notebooks   # Jupyter notebooks for analysis

/scripts   # Python scripts for data processing

/models   # Trained models and results

/visualizations # Generated plots and graphs

/reports   # Final reports and documentation

8.2 Key Tools and Libraries

The analysis uses:

- **Pandas and NumPy** for data manipulation.
- **Matplotlib, Seaborn, and Plotly** for visualization.
- **NetworkX** for graph analysis.
- **Scikit-learn** for predictive modeling and causal inference.

8.3 Continuous Iteration and A/B Testing

The recommendations and models are intended for continuous refinement. A/B testing is crucial to validate that the next-action suggestions drive engagement and improve retention.

---

9. Conclusion & Next Steps

9.1 Summary of Findings

Key Finding	Summary
Data Integrity	Over 1.85 million records from 862 users were ingested, with data filtered for 2025.
User Engagement	High diversity in feature usage correlates with higher retention (Logistic Regression AUC ~0.94).
Natural Workflows	Frequently observed transitions (e.g., from "widget:render" to "configurable-table:render") indicate a natural user flow.
Power Nodes in User Journey	Graph analysis identifies key nodes such as "account-lines::widget:render" as critical hubs.
Causal Impact of Features	Sessions using "AdvancedAnalytics" tend to be longer, suggesting a causal impact on engagement.

9.2 Future Enhancements and Recommendations

- **Advanced Sequence Models:** Explore RNNs, LSTMs, or transformers for improved next-action predictions.
- **Real-Time Deployment:** Develop microservices to deliver real-time prompts based on identified trigger points.
- **Segmentation Analysis:** Further segment data by user roles and geographic regions to tailor recommendations.
- **Enhanced Causal Analysis:** Apply a difference-in-differences approach to validate feature impact post-rollout.

10. Appendix

10.1 Code Snippets and Examples

All the detailed code used for data ingestion, cleaning, EDA, predictive modeling, graph-based analysis, causal inference, and next-action recommendations is provided in the attached code script.

10.2 Additional Resources and References

Resource	Link
Pandas Documentation	pandas.pydata.org
Scikit-Learn Documentation	scikit-learn.org
Plotly Documentation	plotly.com
NetworkX Documentation	networkx.org