CXC Federato

Joey Shum
Nam Ho
Quang Bui
Tung Nguyen

February 24th, 2025

Contents

Summary

In this report, we analyze and develop a recommendation system based on a given dataset. Our approach begins with an in-depth exploration of the original dataset to understand its structure, key features, and potential insights. Based on our findings, we decided to process and aggregate the data into 2 distinct datasets, each designed to build a specialized model.

The first model aims to capture user behavior patterns and predict the likelihood of a user engaging with a given event. The second model focuses on user retention, identifying factors that contribute to sustained engagement over time. The third model is designed to estimate the time users spend on the platform, providing valuable insights into content consumption and engagement duration.

By integrating these three models, we construct an ensemble recommendation system that optimally suggests events. This system is designed to maximize user retention, increase time spent on the platform, and enhance the probability of users selecting recommended events, thereby improving overall user engagement and satisfaction.

Initial Data Processing

Handling event_properties and user_properties

The original format of these two columns (dictionary/JSON) is difficult to process and extract meaningful information from, so we expand them into approximately 100 separate columns. However, since most of these columns are either empty or contain NaN values, we filter out those with more than 50% NaN values. Additionally, we remove columns that are overly dominated by a single category. As a result, we retain only the following columns: slug from event_properties and roles, isInternalUser, referrer from user_properties

Handling event_type

Since the event_type column contains approximately 600 unique categories, analyzing and visualizing it in its raw form would be overwhelming and difficult to interpret. To create more meaningful insights and improve visualization clarity, we have decided to group these categories into broader, more relevant groups based on repeated keywords. This approach helps reduce noise, enhances pattern recognition, and allows for easier comparison across different event types while preserving the overall structure and significance of the data.

Category	event_type keywords
‘Sessions & Navigation’	session_start, session_end, application-window, nav-header, dashboard
‘Account/Policy Mgmt’	account, policy, rating
‘Dashboard & UI’	dashboard, widget, layout, insights, table

‘Action Center’	action-center, task, workflow, take-action
‘Submissions & Forms’	submit-click, form, create, definition, save-click, submissions
‘Filter & Searching’	filter, sort, search, advanced-filters
‘Document & Report’	document, report, download, csv
‘Other’/ System Events	EMPTY, help-page, user-signed-out, everything else

Handling slug

Original Slug Pattern	Mapped Slug
Numeric values	numeric
Starts with "account"	account
Starts with "email-inbox"	email-inbox
Starts with "excess" or "Excess"	excess
Contains "access-control"	access-control
Contains "agency"	agency
Contains "auto"	auto
Contains "aviation" or "fleet"	aviation
Contains "bond"	bond
Contains "classification"	classification
Contains "compliance"	compliance
Contains "dashboard"	dashboard
Contains "financial"	financial
Contains "flood"	flood
Contains "general" or "gen-upsert"	general
Contains "insurance"	insurance
Contains "policy" or "policies"	policy
Contains "property"	property
Contains "submission"	submission
Contains "user"	user
Contains "vehicle"	vehicle
Contains "workers-comp"	workers-comp
Contains "casualty"	specialty-casualty
Contains "rating" or "commodity"	rating-commodity
Contains "fmcsa" or "mvr"	auto
Contains "guidelines" or "document"	documentation

Contains "aviation", "cyber", "terrorism", "marine"	specialty

Due to the column's high cardinality, we have decided to use leave-one-out target encoding. This method encodes each category as the mean target value of all other rows with the same category, excluding the current row. This approach reduces dimensionality compared to one-hot encoding and prevents data leakage, helping the model learn meaningful patterns without overfitting.

Why do we want to overfit the model initially?
Overfitting the training data initially can help us understand the capacity of our model, revealing its upper limits and potential performance on the training set without regularization or other constraints. More specifically, it can help us with the following things:

1. Baseline: Establish a baseline performance for the model.
2. Model Complexity Check: Allowing the model to overfit helps confirm that it has sufficient capacity to capture patterns in the data. If it's unable to overfit the training data, the model may be too simple, or the limitation could stem from preprocessing or data quality issues.

We will attempt to overfit the model by adding more features through feature engineering, guided by our intuition and business insights.

Engineering
Time to Server (time_to_server): Calculated as the difference between client_upload_time and client_event_time, this feature measures the delay between when an event occurs on the client-side and when it is uploaded to the server.

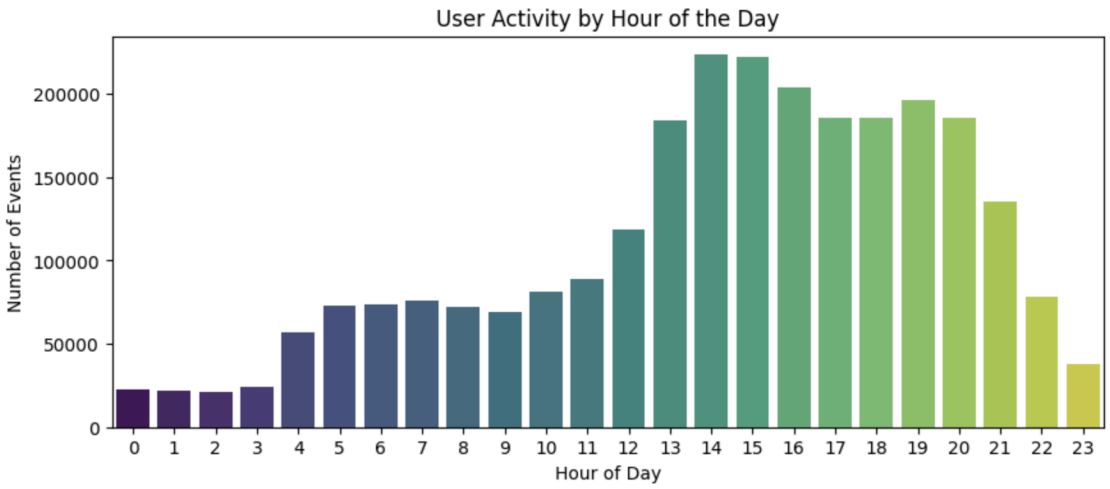
Server to Process (server_to_process): Derived by subtracting server_received_time from server_upload_time, this captures the time taken for the server to acknowledge and begin processing the uploaded data.

Processing Time (processing_time): Computed as the difference between processed_time and server_upload_time, this feature represents the time required for the server to fully process an event.

Initial EDA (EDA on full dataset)

The dataset captures event-based user interactions within the Federato RiskOps platform. The goal of the EDA is to uncover user behavior patterns that influence retention and engagement over a 28-day period.

Event Type Analysis



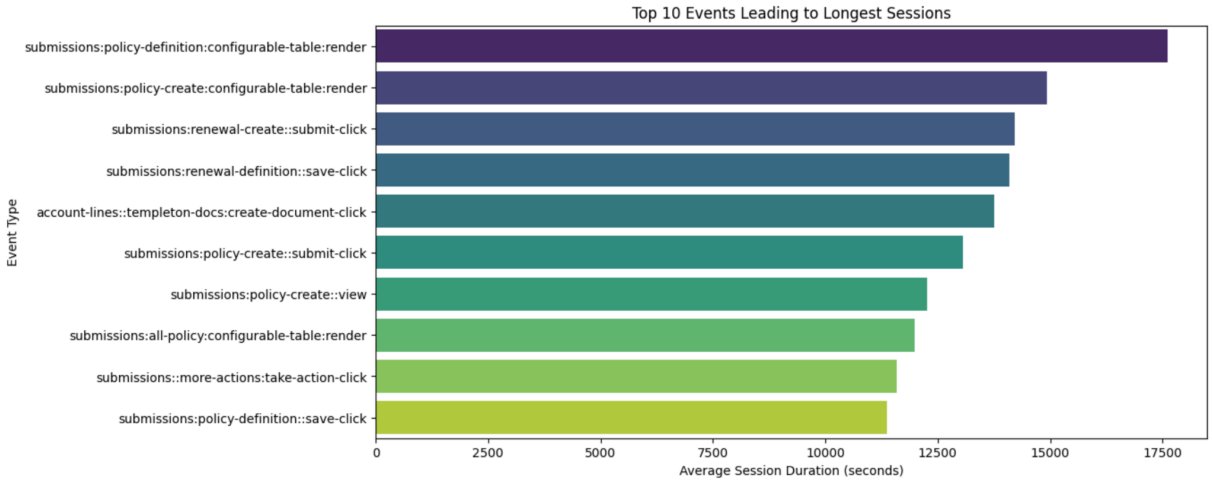
User Activity by Hour of the Day

This visualization presents user activity distribution across different hours of the day. The bar chart indicates that user engagement is relatively low during early morning hours (midnight to 6 AM) but steadily increases as the day progresses. The peak activity period is observed between 1 PM and 4 PM, suggesting that users are most active on the platform during the afternoon hours. After 4 PM, the events gradually decline, with a noticeable drop after 9 PM.

This pattern suggests that the application experiences the highest engagement during working hours, which intuitively makes sense given the nature of the platform. With that said, a secondary peak in the late evening could indicate users revisiting the platform post-work hours.

Implications:

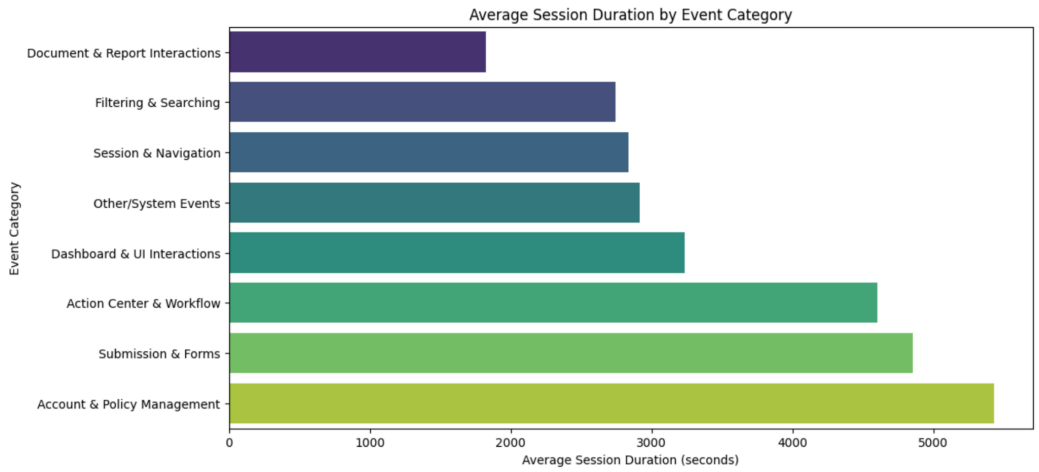
- System performance optimizations should focus on peak usage hours (2 PM – 7 PM).
- Engagement strategies could be more effective if triggered before peak hours.
- A/B testing for new features within the platform should ideally be conducted during peak times to ensure higher participation.



Events Leading to Longest Sessions

Based on the chart above, we can see that **the longest user sessions are strongly associated with form submission events**. This trend suggests that users engage deeply when completing underwriting tasks, likely due to the complexity of decision-making processes and the need to enter detailed information. Unlike high-frequency events that revolve around viewing or navigating, form submission represents a more **intentional** and **high-commitment** action, where users are actively progressing toward a business outcome.

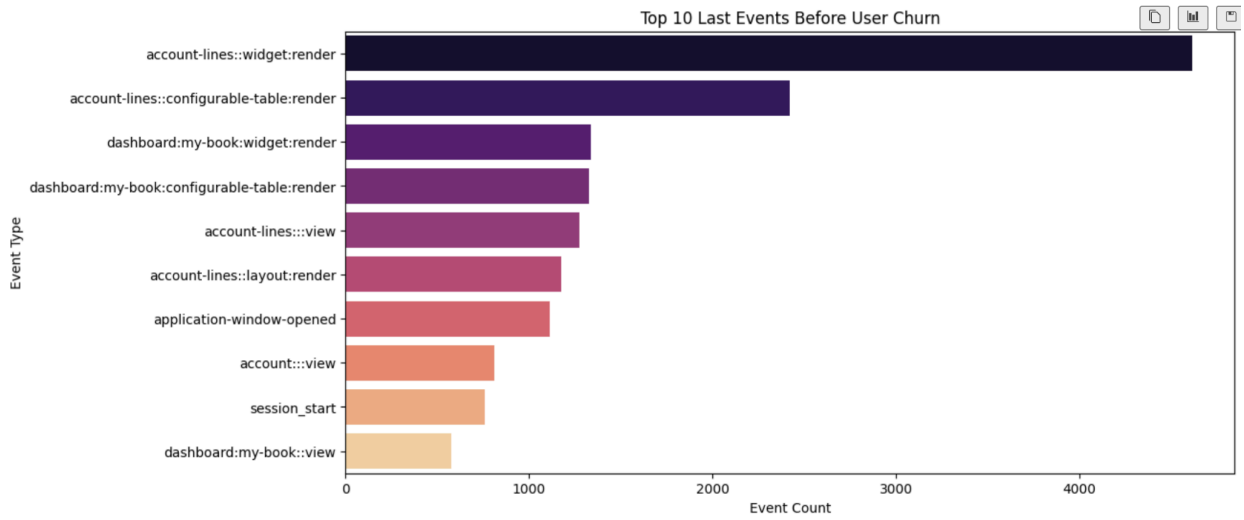
There are several possible explanations for this extended engagement. First, **the cognitive load required for underwriting tasks** may necessitate careful evaluation of multiple data points before finalizing submissions. Users might spend additional time cross-referencing policies, adjusting risk factors, or ensuring that entered information is accurate. Second, lengthy form submissions may indicate **potential usability challenges**, where users take longer than necessary to complete tasks due to **unclear field requirements, complex dependencies, or excessive manual inputs**.



Average Session Duration by Event Category

The chart illustrates the average session duration categorized by event type. Categories such as "Account & Policy Management" and "Submission & Forms" exhibit the highest average session durations, implying that users spend significant time on administrative or form-filling tasks. Conversely, interactions related to "Document & Report Interactions" and "Filtering & Searching" have shorter session durations, suggesting that users can simply quickly access the required information and move on.

These insights are crucial for identifying pain points or areas that may need UX optimization. Long session durations in form submissions, for example, might indicate cumbersome processes that could benefit from streamlining.



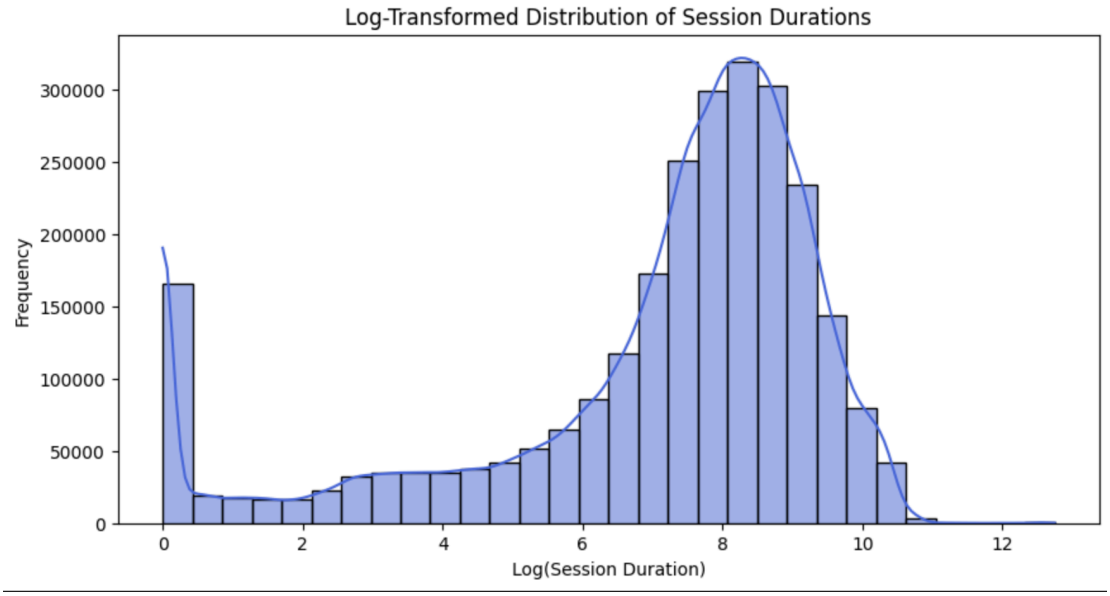
Top 10 Last Events Before User Churn

The following chart highlights the most common last events before user churn. The most frequent event preceding churn is "account-lines::widget:render," followed by similar interactions related to account-line features and dashboard navigation. These events suggest that users might be disengaging after specific interactions, possibly due to frustration, unclear workflows, or unmet expectations.

Analyzing these final interactions can inform targeted interventions, such as improving UI elements, enhancing feature explanations, or triggering retention-focused messaging at critical junctures. Understanding user drop-off points is fundamental for reducing churn and increasing long-term engagement.

Implications:

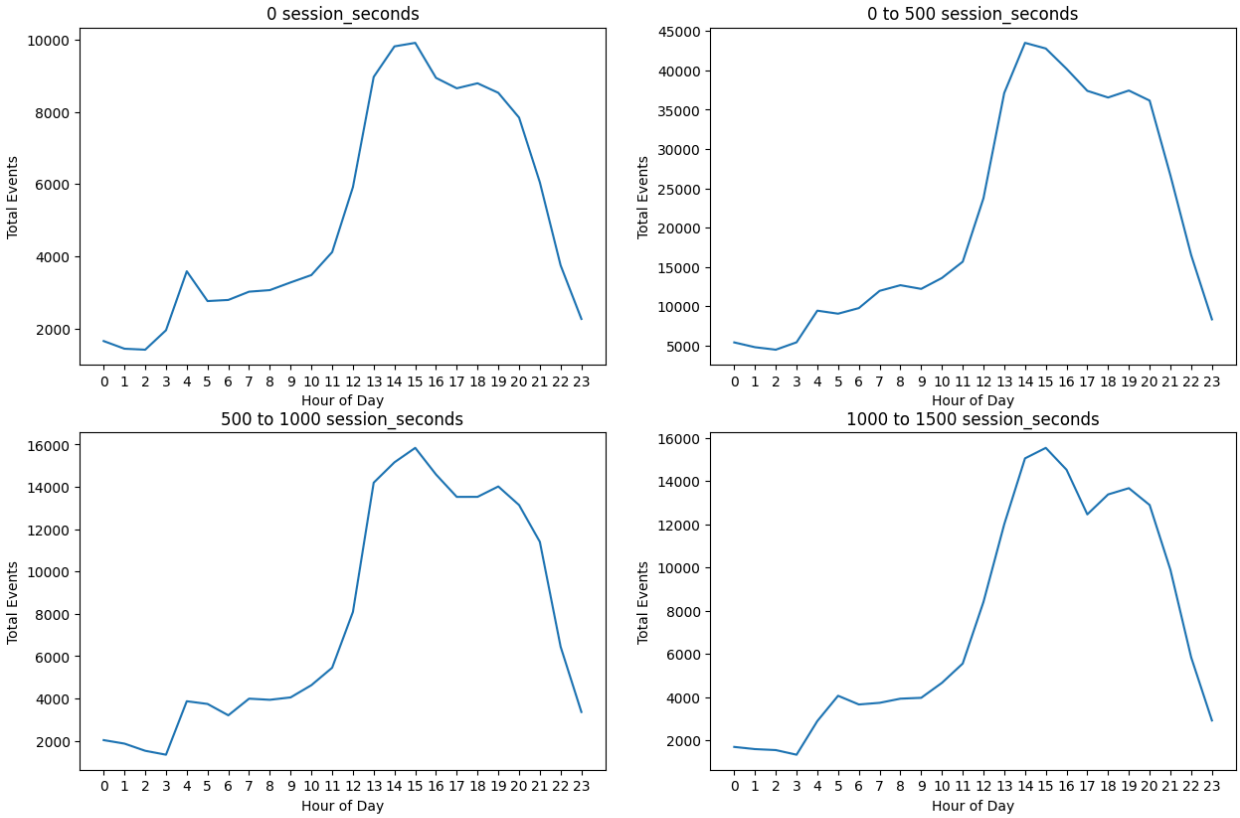
- Users who frequently engage with dashboards but do not proceed to underwriting actions may need **guidance on the next steps**.
- Introducing **personalized recommendations or tooltips** could encourage users to take meaningful actions instead of passively navigating.



Log-Transformed Distribution of Session Durations

The graph illustrates the distribution of how long an user spent on a session using a log transformation. The histogram demonstrates a right-skewed distribution, indicating that a significant number of users have longer session durations, while a smaller proportion engages for short periods, potentially just to check or verify their actions.

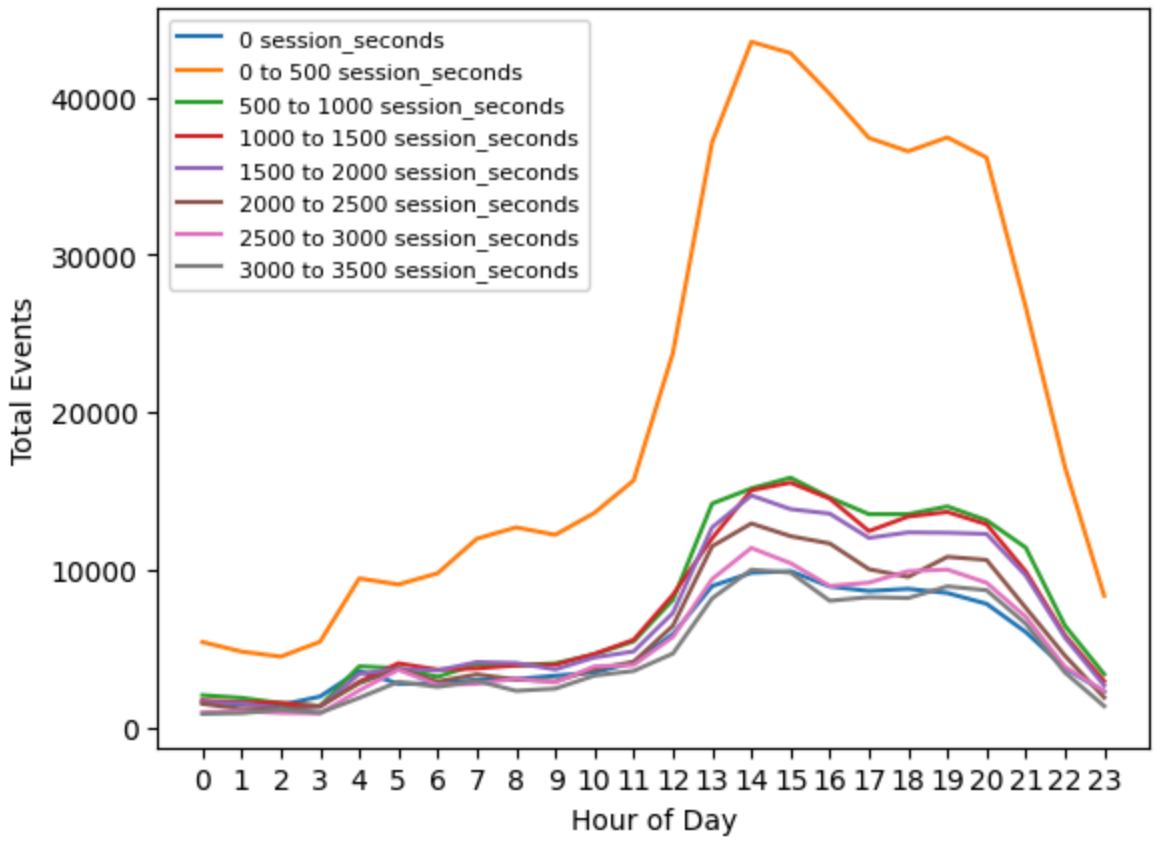
A sharp peak is observed around a log-transformed value of ~8, suggesting a modal session duration within that range. The presence of long-tail values implies that some users have extended engagement, which may correlate with higher retention or deeper interactions within the platform. The log transformation effectively normalizes the data, making it easier to interpret patterns and outliers in session behavior.



Total Events by Hour of Day, Segmented by Session Duration

By segmenting the number of events into different groups of session duration, we aim to identify the differences between sessions. Overall, we see that the shape for all graphs are fairly similar. Small differences include a spike at 4am when session_seconds is 0. Having session_seconds = 0 implies that there is only one action in that section. There is more analysis later on the event types that have a session duration of 0.

Another difference is that the number of total events have a bigger difference from 17h to 19h in the bottom right graph, while other graphs have a flatter curve at those times. In the following graph, we dive deeper into the differences by overlaying the curves on top of each other.

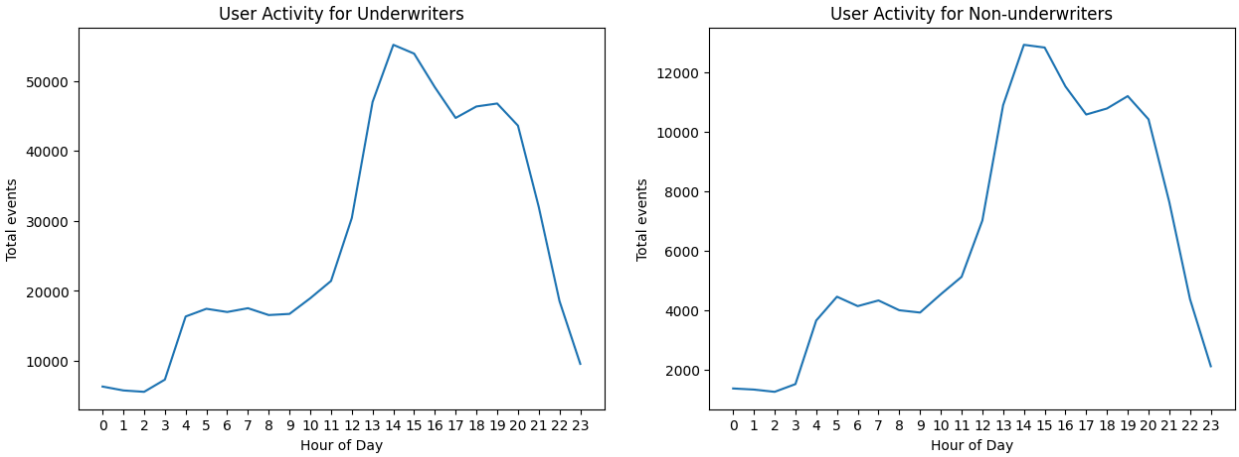


We create buckets of size 500 and plot the curves all on the same graph. This visualization indicates that the total events in small session durations (0 to 500 seconds) have high engagement. Although all curves have a similar shape, this curve sets apart from the others.

event_type	
session_start	26572
session_end	26246
application-window-opened	17115
dashboard:my-book::view	11650
account-lines:::view	5397
account:::view	5240
dashboard:my-book:widget:render	4280
account-lines::widget:render	3515
dashboard:my-book:configurable-table:render	3436
:all-accounts::view	1874
account-lines::configurable-table:render	1572
all-accounts:::view	1449
dashboard:my-book:layout:render	1327
action-center:action-details::view	1191
action-center:::view	1172
action-center:::close-click	1093
all-accounts::accounts-table:account-click	1032
account-lines:::change-rating-click	1003
::nav-header:action-center-click	779
account-lines::layout:render	746
account-property-rating:perils::view	740
submissions:all-policy::view	683
::configurable-table:render	668
account-page-view	617
:all-accounts:configurable-table:render	517
unified-dashboard-page-view	497

Event_type Where Session Duration is 0

As mentioned above, a session duration of 0 implies that there is only one action taken for a particular session id. This insight is important because it shows the causes of low engagement. The table above shows the different categories in event_type and their corresponding value counts, arranged in descending order. After the top 3 values, the rest of the top values are all view or render. They are mostly related to dashboard and account, followed by action-center.



User Activity for Underwriters and Non-underwriters

In these graphs, “underwriters” are defined as any user that has “underwriter” as one of their roles, if they have multiple. The shape of both curves are nearly identical and align with the visualization for “User Activity by Hour of the Day” as presented in an earlier section. However, note that the total number of events for underwriters are significantly higher than that of non-underwriters, as indicated by the scale in the y-axis. This implies that underwriters perform more actions and engage with the Federato RiskOps Platform more.

Data Processing - Creating training datasets

Metrics

From the initial round of EDA, we define our goal is to recommend user’s action that will optimize retention and time spent on the platform. Specifically, we define retention as **“If the user returns to the platform 28 days after they finish a session”** and time spent on the platform as **“Average time spent per session”.**

Goal

Create two datasets to train three models for a recommendation system that optimizes user engagement and retention.

- The first dataset focuses on retention and session duration, where the target variables include whether a user returns within seven days after a session and the time spent on the platform per session. This dataset enables the training of two models: one predicting user retention and another estimating session duration.
- The second dataset is designed to predict the user's most probable action, capturing behavioral patterns that drive engagement.

By integrating these models, the recommendation system can not only anticipate user actions but also suggest content or features that enhance session duration and increase retention, ultimately improving overall user experience.

First Dataset
In the first dataset, we first perform one hot encoding for the categorical columns, we then drop all rows that has more than 50% of missing values, which leaves us with the following columns”

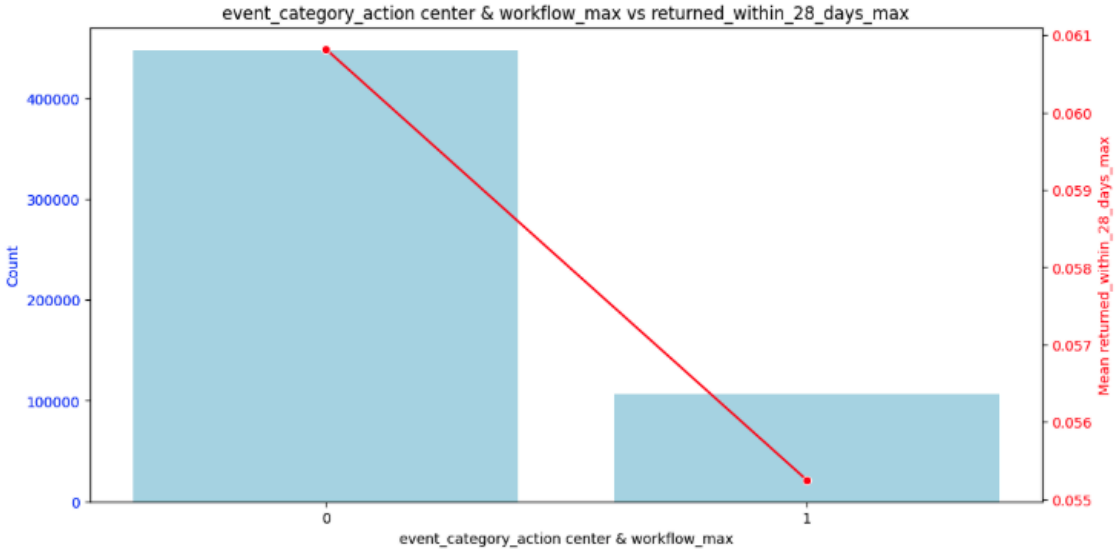
```
columns = [
    'session_id', 'user_id_first', 'device_family_linux_max',
    'device_family_mac os x_max', 'device_family_windows_max',
    'region_grouped_international_max', 'region_grouped_midwest_max',
    'region_grouped_northeast_max', 'region_grouped_south_max',
    'region_grouped_west_max',
    'event_category_account & policy management_max',
    'event_category_action center & workflow_max',
    'event_category_dashboard & ui interactions_max',
    'event_category_other/system events_max',
    'event_category_session & navigation_max',
    'event_category_submission & forms_max', 'returned_within_28_days_max',
    'uw_max', 'admin_max', 'manager_max', 'broker_max', 'google_max',
    'microsoft_max', 'session_seconds_mean', 'client_event_hour_mean',
    'client_upload_hour_mean', 'event_hour_mean',
    'server_received_hour_mean', 'server_upload_hour_mean',
    'time_to_server_mean', 'server_to_process_mean',
    'Processing_time_mean'
]
```

For retention the target column is ‘returned_within_28_days_max’, and for time spent on the platform per session the target column is 'session_seconds_mean'.

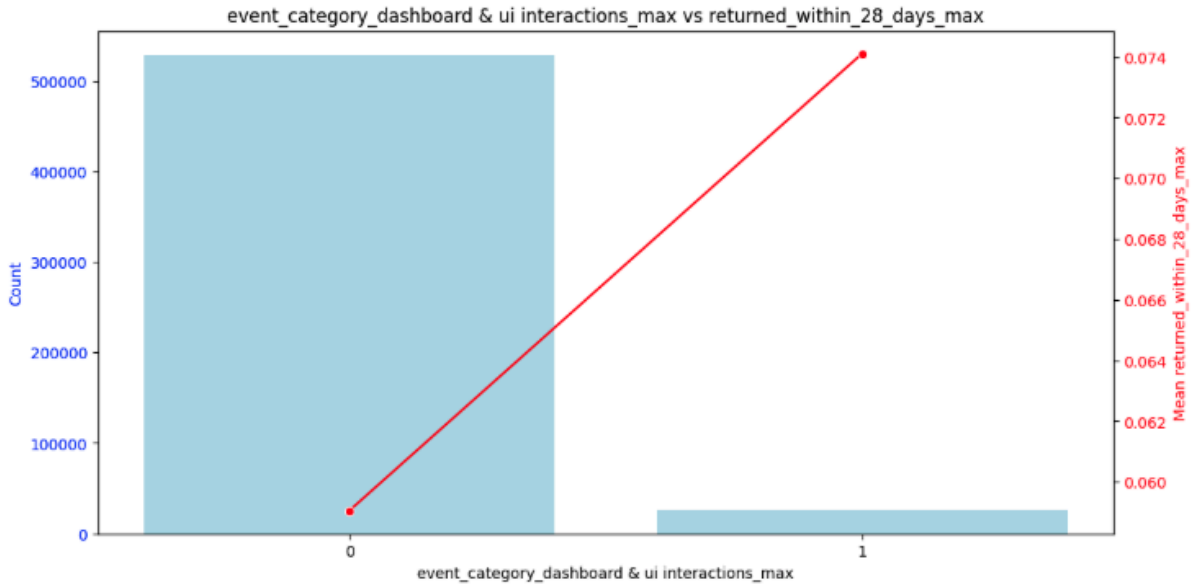
Second Dataset
In the second dataset, we group by user_id and session_id and aggregate sequence of actions taken during that session. Each action is assigned a numerical token ranging from 0 to 740, where 740 represents an empty token. To train the model effectively, we transform each sequence into multiple training samples by progressively hiding the next action in the sequence. This means that for each session, we generate multiple rows where the model is given a partial sequence as input and is trained to predict the next action. This approach helps the model learn action patterns and anticipate what a user is most likely to do next based on their past interactions.

EDA On Retention

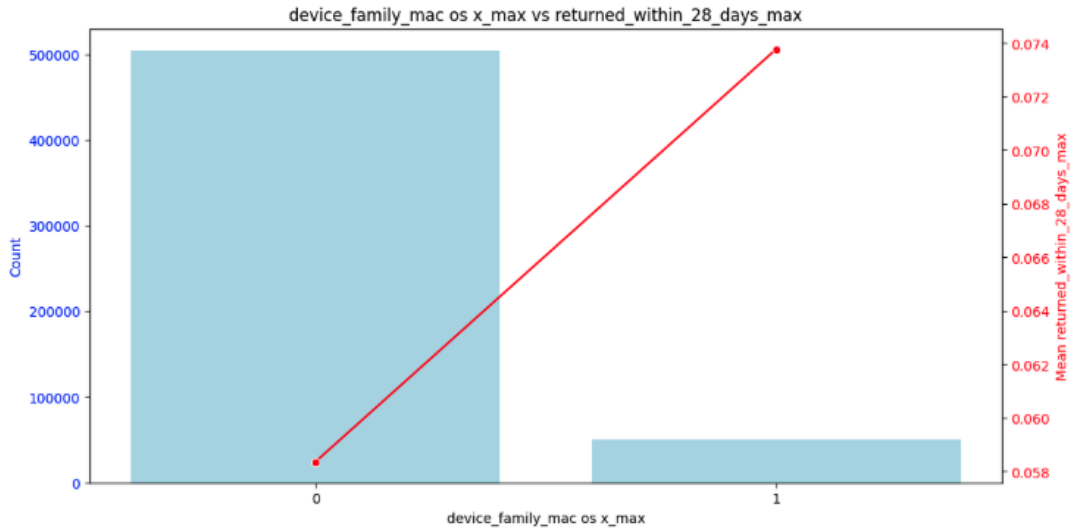
Event_category

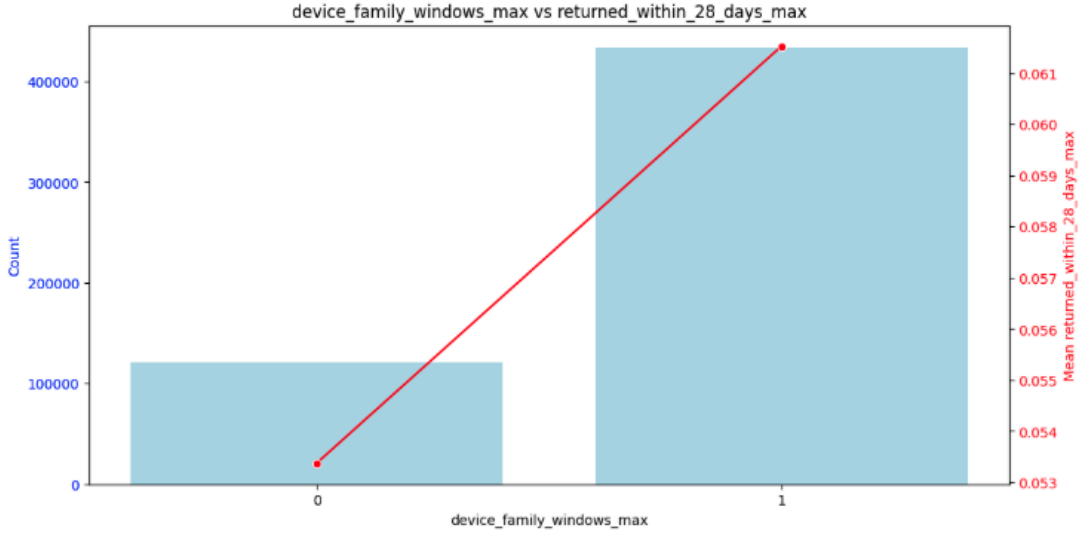

event_category_action center & workflow_max vs returned_within_28_days_max
Count
Mean returned_within_28_days_max
event_category_action center & workflow_max
0
100000
200000
300000
400000
0.055
0.056
0.057
0.058
0.059
0.060
0.061
0
1

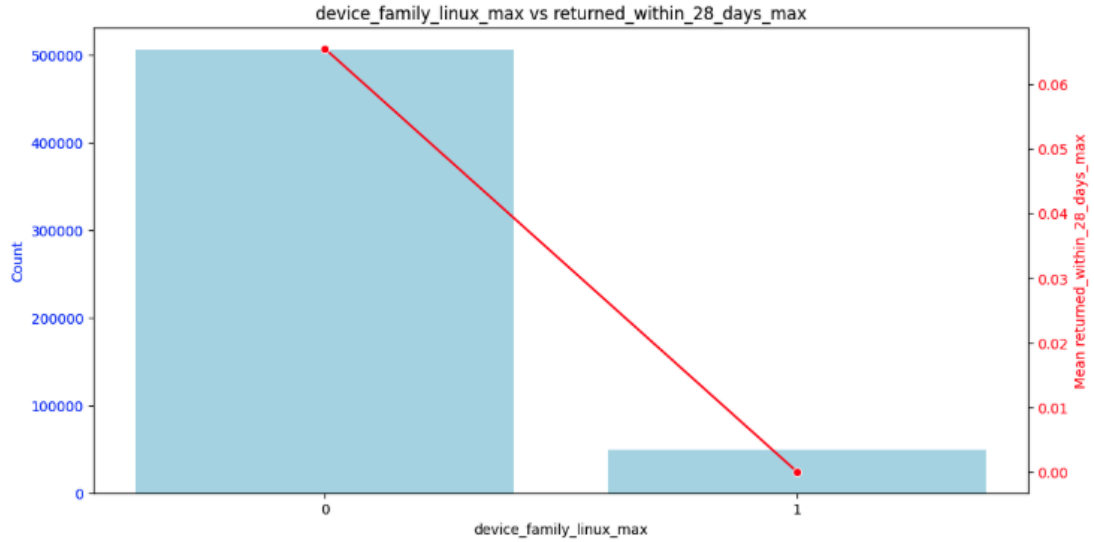

event_category_account & policy management_max vs returned_within_28_days_max
Count
Mean returned_within_28_days_max
event_category_account & policy management_max
0
50000
100000
150000
200000
250000
0.040
0.045
0.050
0.055
0.060
0.065
0.070
0.075
0.080
0
1



The Action Center & Workflow shows high activity correlating with lower return rates, with both metrics declining together. Account & Policy Management maintains steady event counts while return rates increase with activity. Dashboard & UI Interactions shows a dramatic drop in frequency but improved return rates at higher activity levels. These patterns suggest each feature type impacts user retention differently, with account management and dashboard interactions showing more positive correlation with returns despite varying usage levels.

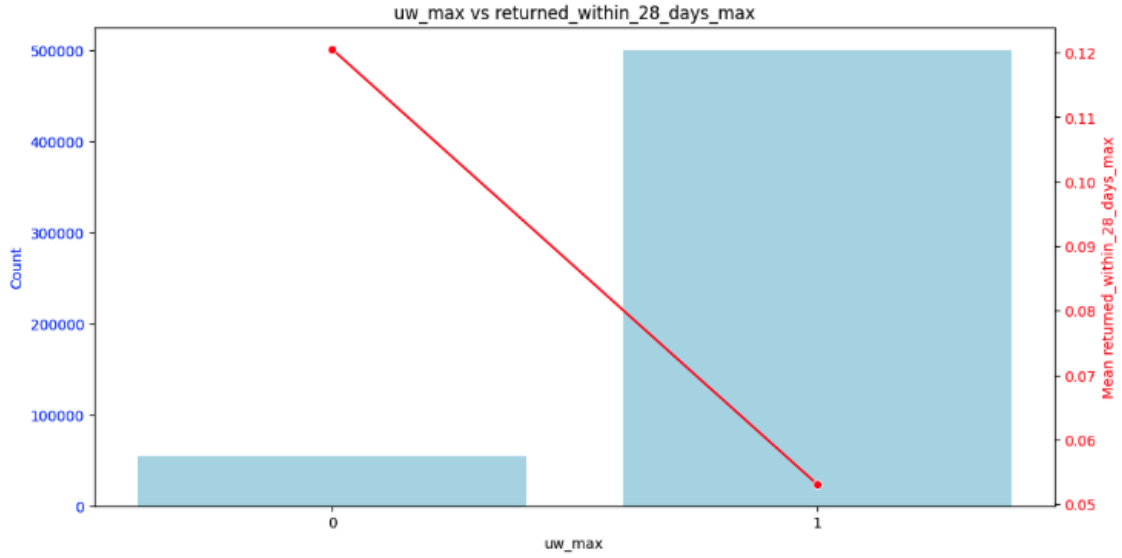

device_family_mac os x_max vs returned_within_28_days_max
Count
Mean returned_within_28_days_max
device_family_mac os x_max
0
1
500000
400000
300000
200000
100000
0
0.074
0.072
0.070
0.068
0.066
0.064
0.062
0.060
0.058

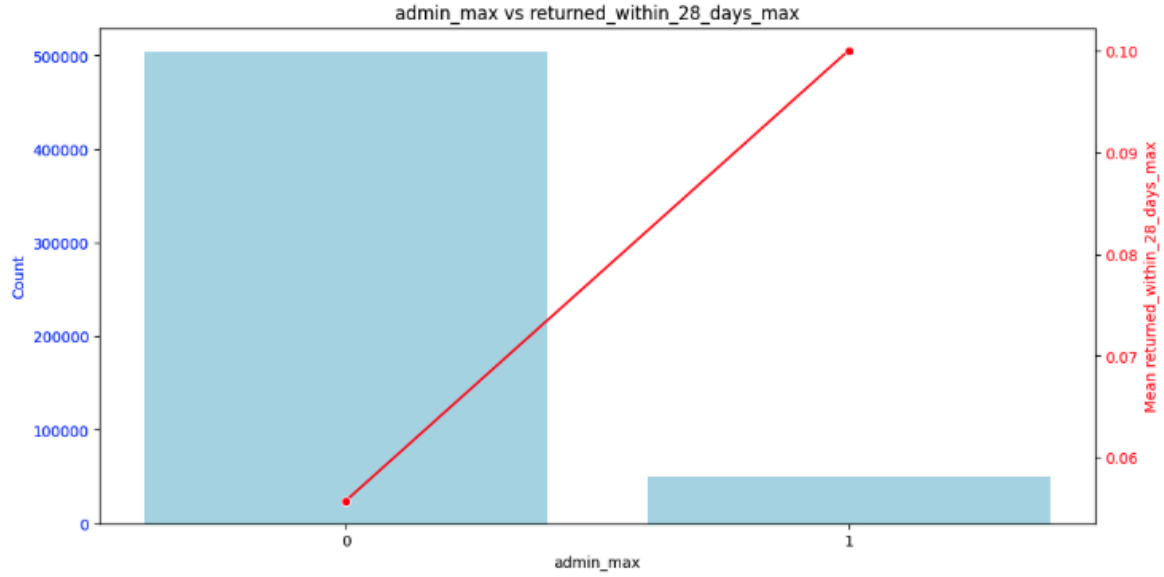

device_family_windows_max vs returned_within_28_days_max
Count
Mean returned_within_28_days_max
device_family_windows_max
0
1
400000
300000
200000
100000
0
0.061
0.060
0.059
0.058
0.057
0.056
0.055
0.054
0.053

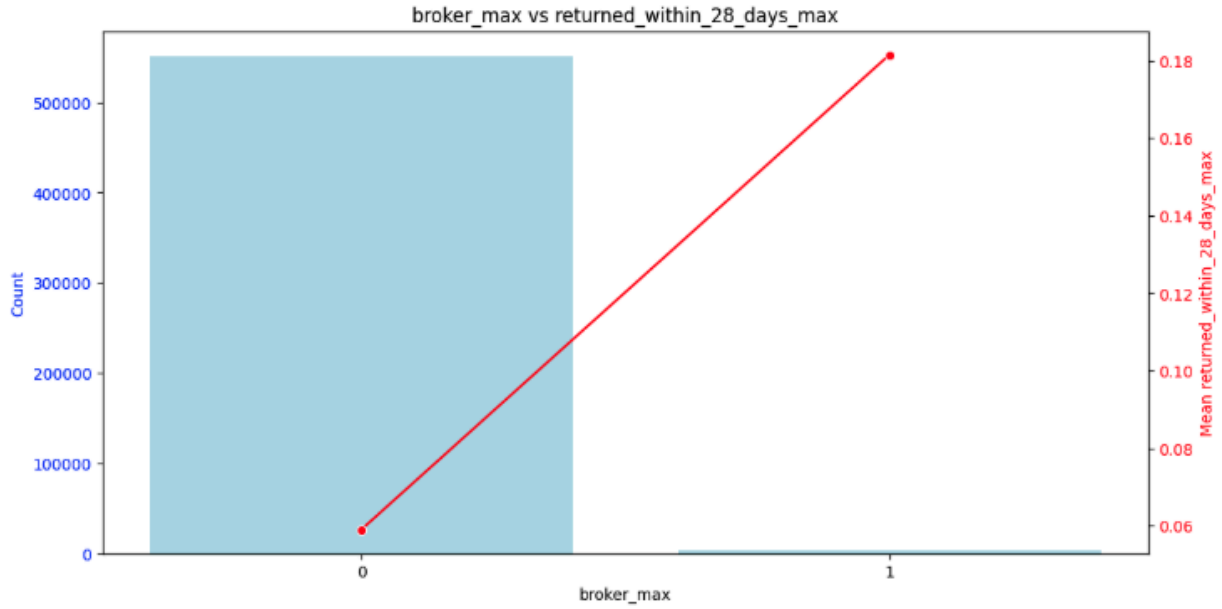

device_family_linux_max vs returned_within_28_days_max
Count
Mean returned_within_28_days_max
device_family_linux_max
0
1
500000
400000
300000
200000
100000
0
0.06
0.05
0.04
0.03
0.02
0.01
0.00

The Mac OS category shows high event volume at state 0, dropping significantly at state 1, though return rates increase. The Windows category displays the opposite pattern, lower counts at state 0 but higher volume at state 1, with return rates rising alongside usage. The Linux category starts with high volume at state 0 but drops sharply at state 1, with return rates declining correspondingly. These patterns indicate that Windows users show increasing engagement and retention, while Linux users trend downward in both metrics. Mac users show an interesting split - lower usage but higher retention at maximum activity levels.

Roles:

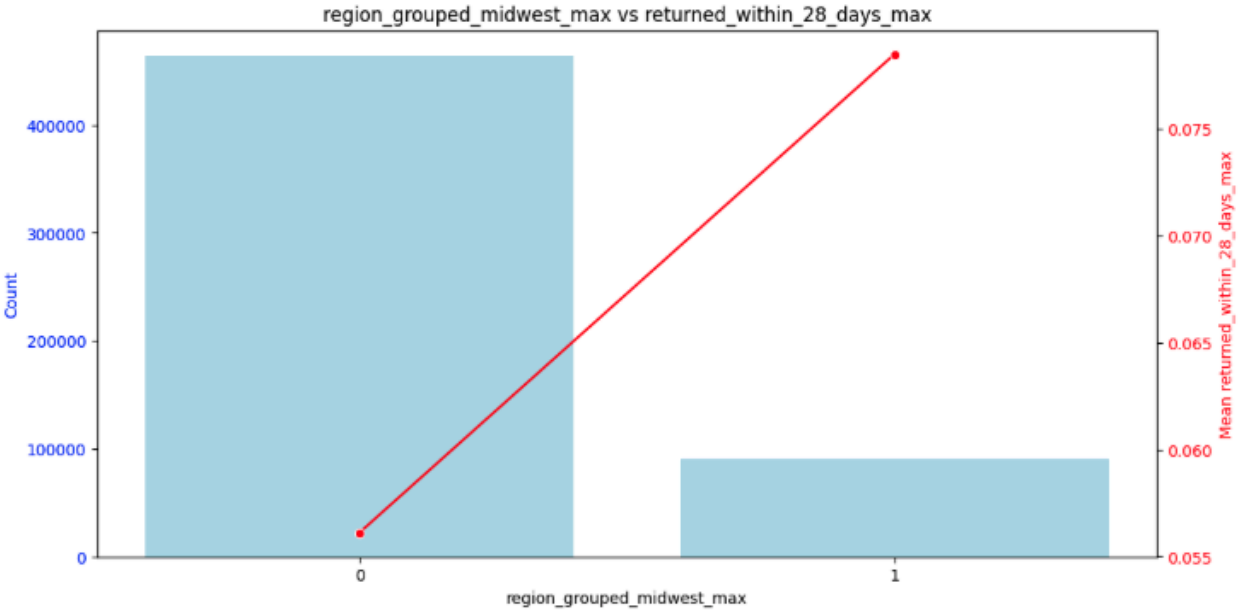


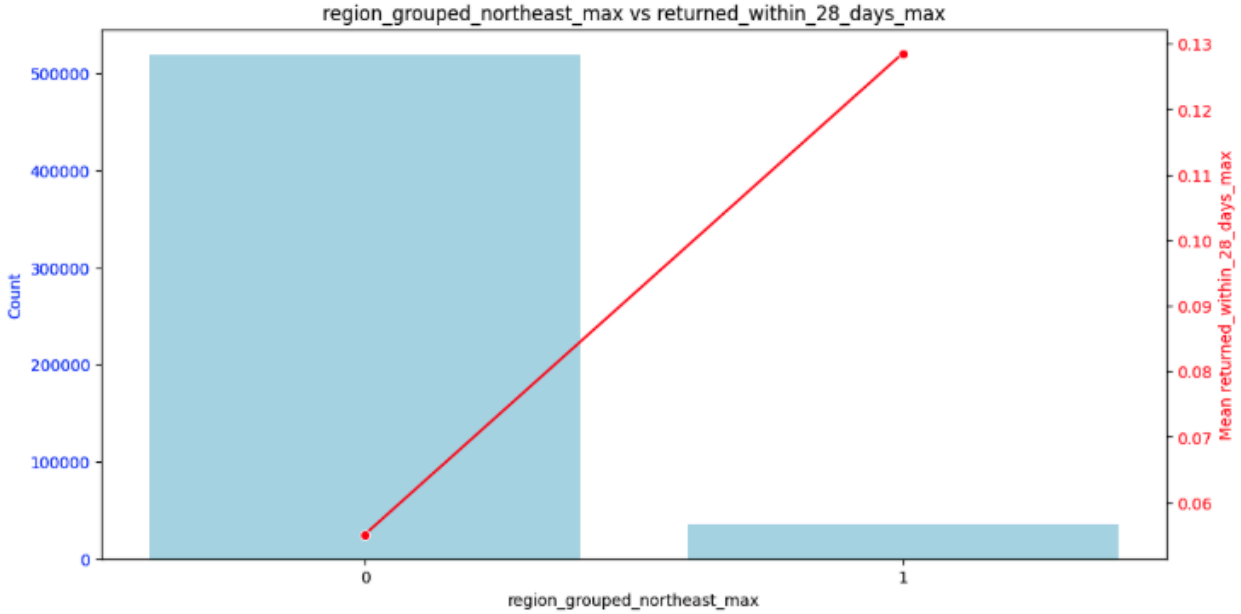


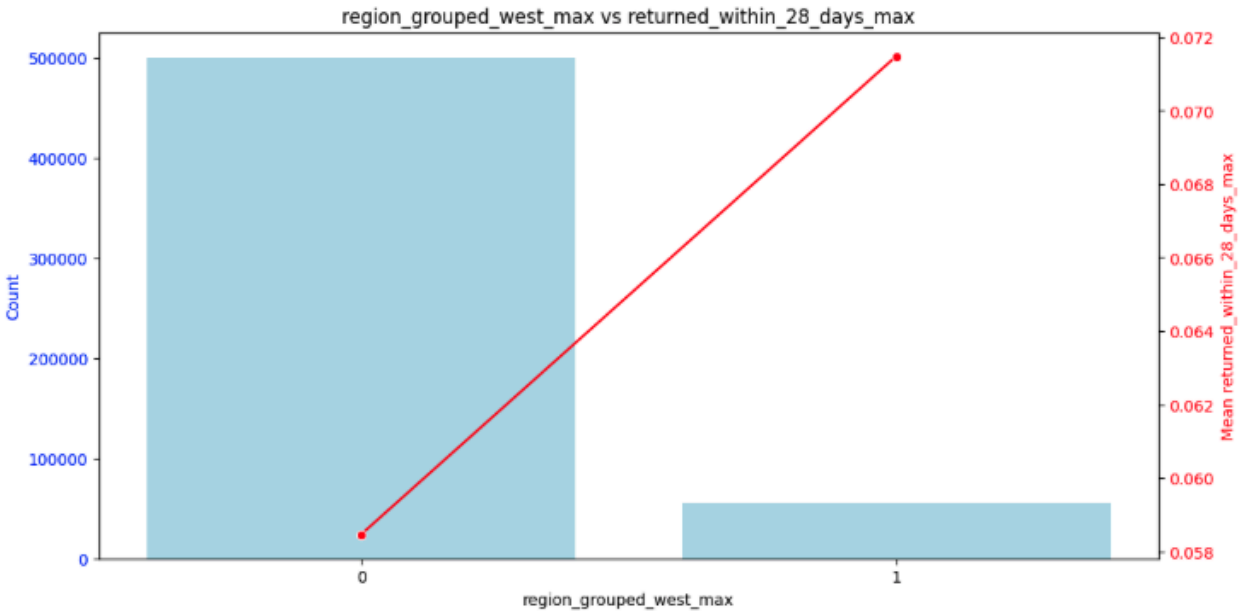


The data shows that Underwriters have much higher activity frequency and increase retention rates. In contrast, admin, manager, and broker roles display an inverse pattern, they show lower activity frequency when the role is active but significantly lower retention. This suggests that while regular users have more frequent interactions with the platform, admin and manager roles engage in less frequent and did not usually return to the platform within 28 days.

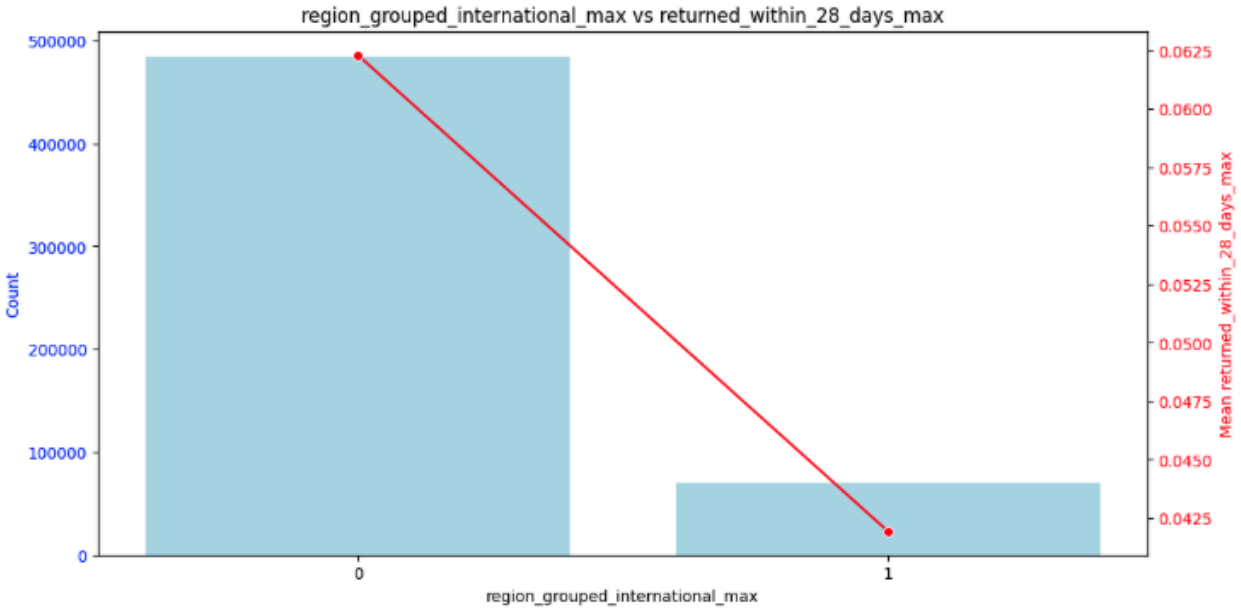
Region:

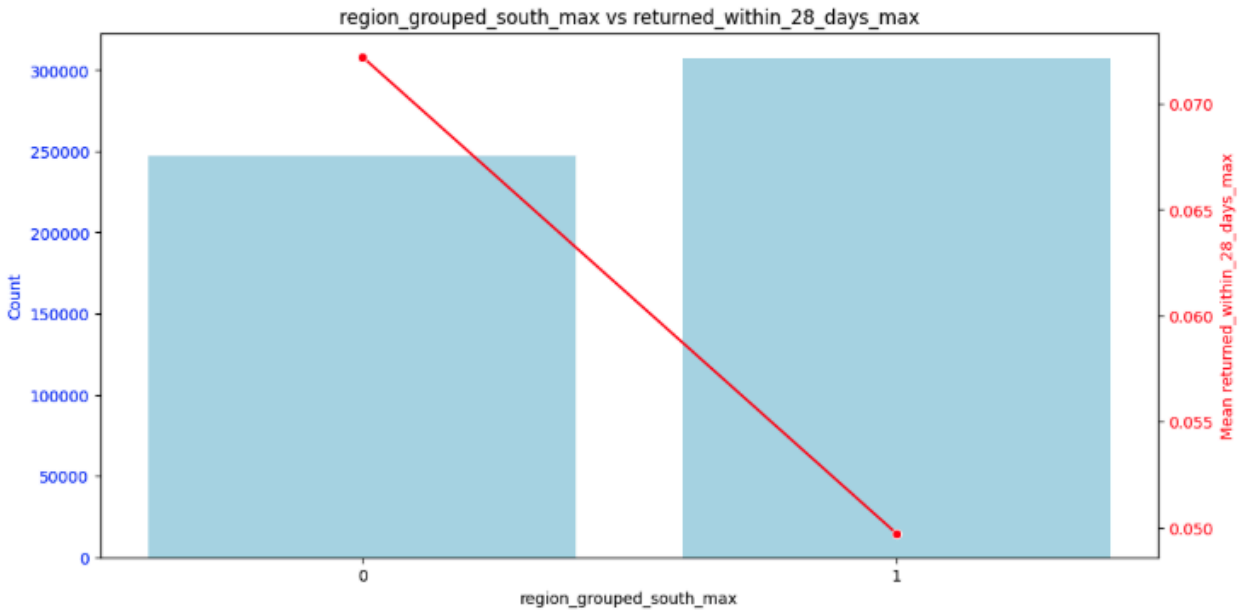

region_grouped_midwest_max vs returned_within_28_days_max
Count
Mean returned_within_28_days_max
region_grouped_midwest_max
0
1
0
100000
200000
300000
400000
0.055
0.060
0.065
0.070
0.075


region_grouped_northeast_max vs returned_within_28_days_max
Count
Mean returned_within_28_days_max
region_grouped_northeast_max
0
1
0
100000
200000
300000
400000
500000
0.06
0.07
0.08
0.09
0.10
0.11
0.12
0.13



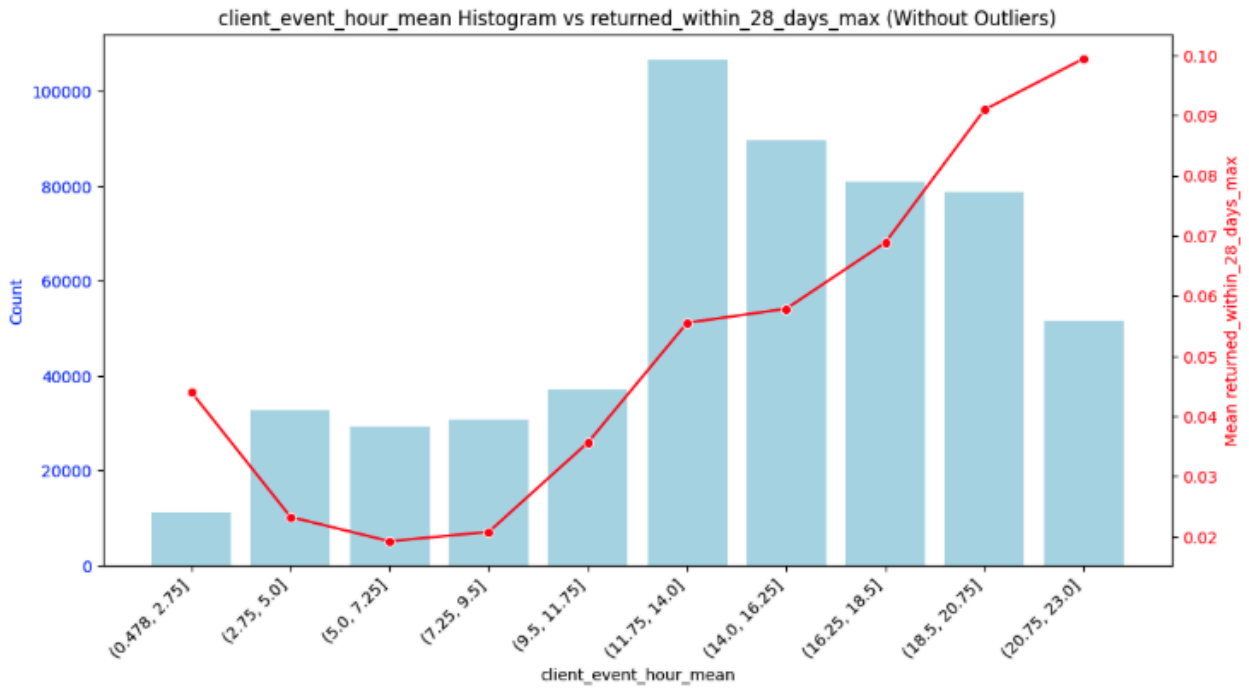
These graphs above indicate a positive correlation between the presence of "West", "Midwest" and “Northeast” regional characteristics and users retention rate. Users categorized within these regions exhibit notably higher 28-day retention rate compared to those without these attributes. This suggests that these regional factors may positively influence user engagement and contribute to extended interaction with the platform or service.





On the other hand, the presence of "International" and “South” regional characteristics appears to correlate with a decrease in users’ 28 day retention. Users from these regions demonstrate lower retention rate, indicating a potential negative association between these regional factors and platform stickines. Understanding these trends may help tailor strategies to improve user retention and engagement within these specific regions.

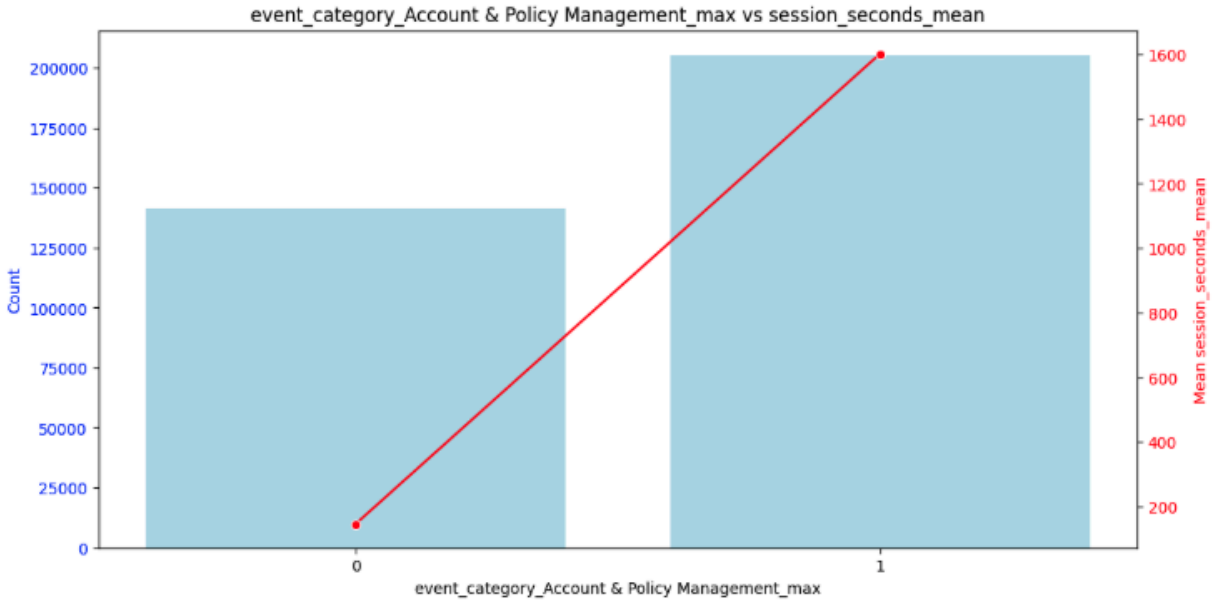
User’s Usage time during the day

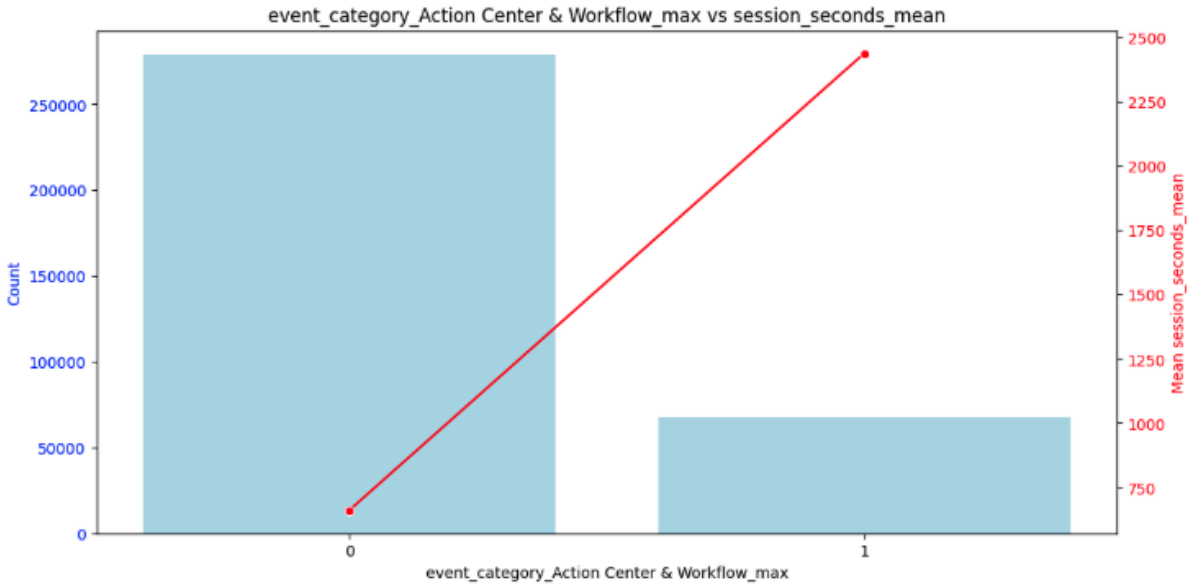


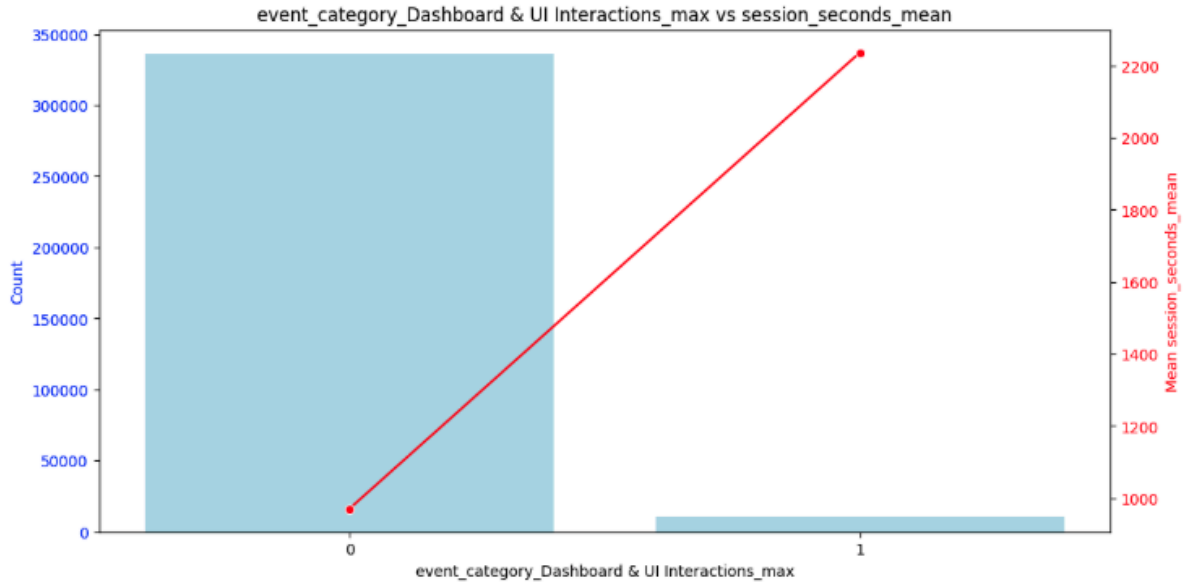
User activity follows a distinct daily pattern, peaking significantly during mid-day hours (around 105k events) and showing lowest activity in early morning hours (10k events). The return rate (red line) shows an interesting trend - it starts relatively high at the earliest hours, dips during early morning, then steadily climbs throughout the day to reach its peak in late evening hours. This suggests that while mid-day sees the highest volume of activity, users who engage during evening hours are more likely to return within 28 days.

EDA On Session Time

Event_category

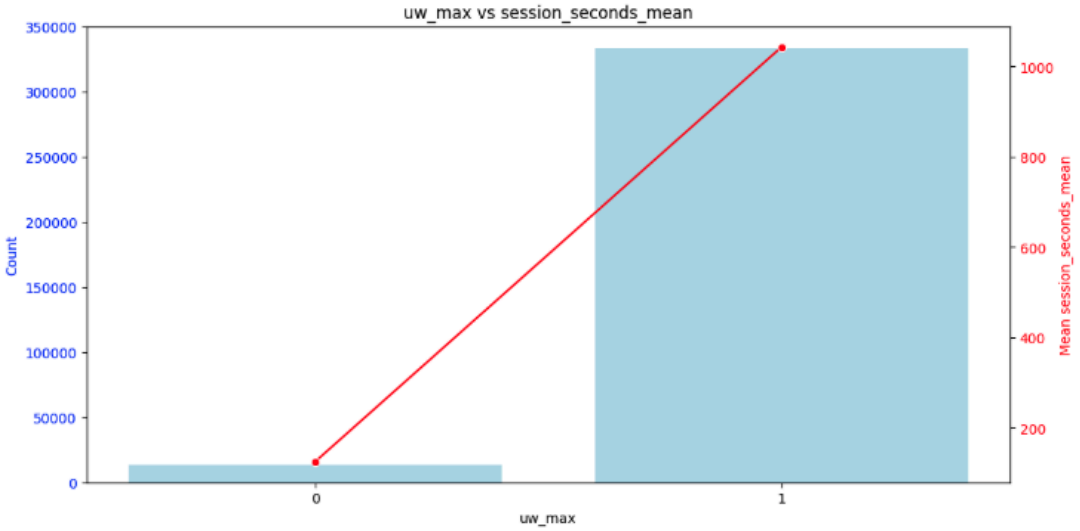

event_category_Account & Policy Management_max vs session_seconds_mean
Count
Mean session_seconds_mean
event_category_Account & Policy Management_max
0
1
0
25000
50000
75000
100000
125000
150000
175000
200000
200
400
600
800
1000
1200
1400
1600

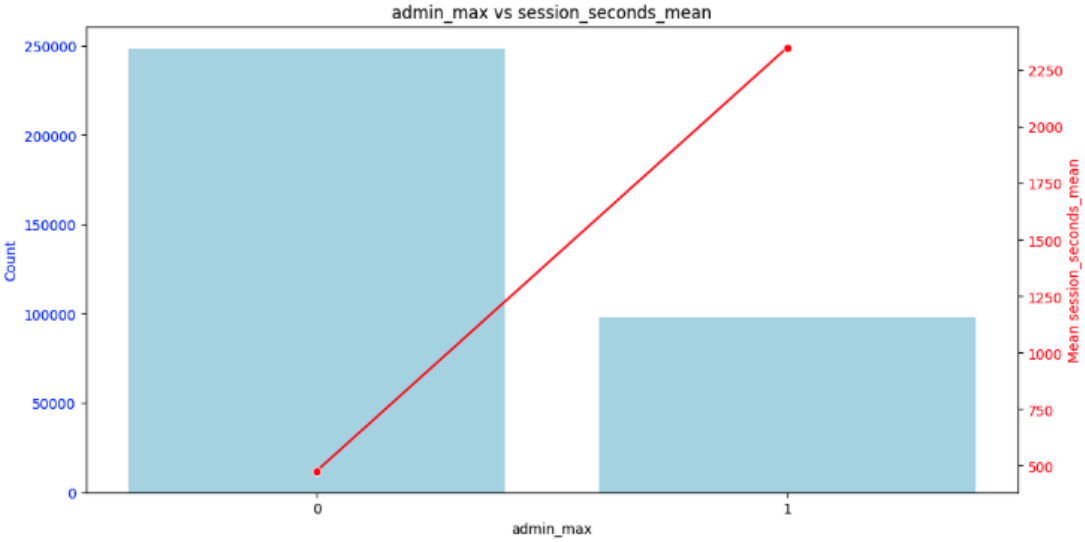


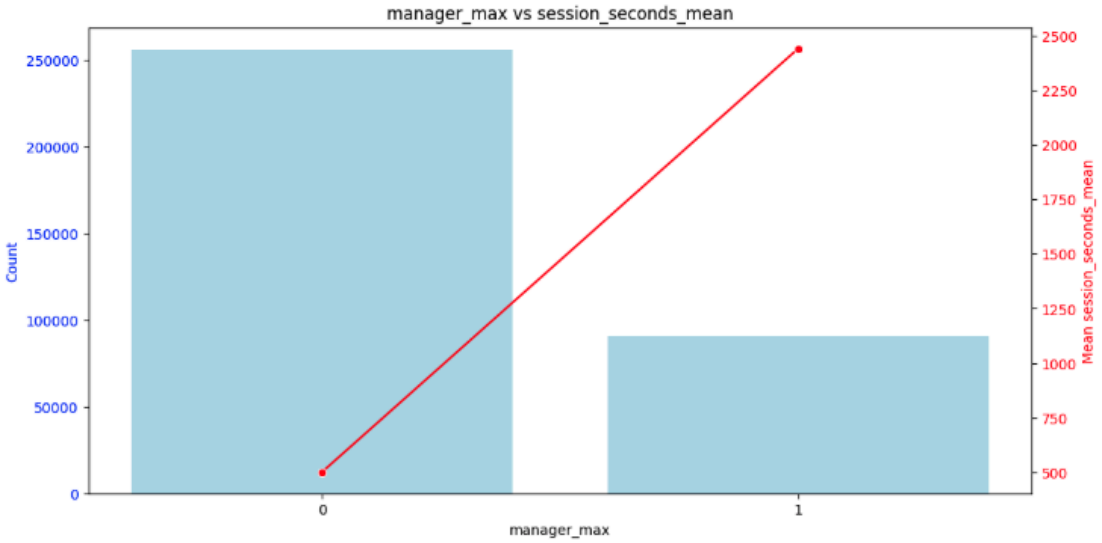


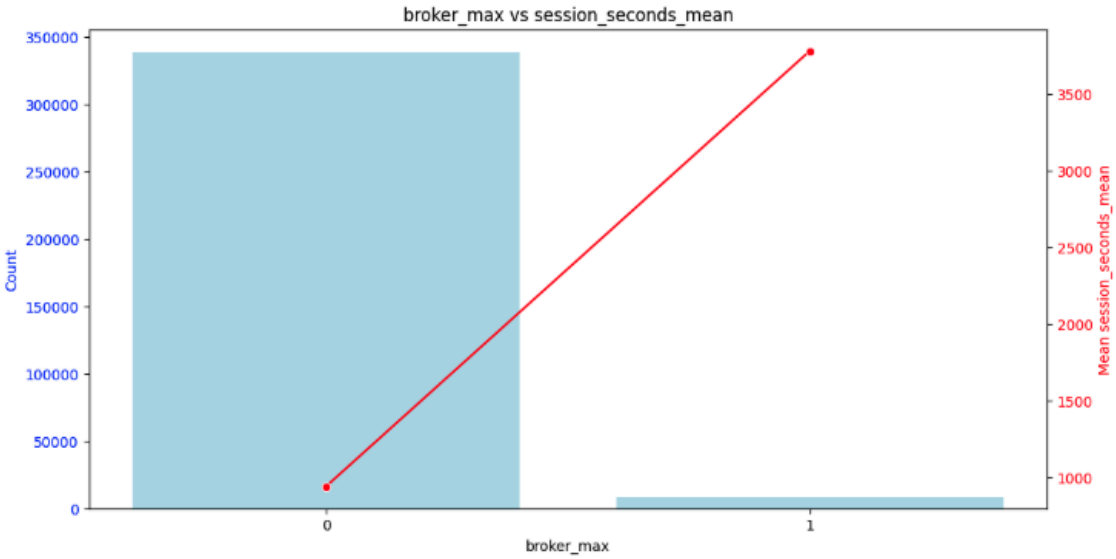
The three graphs reveal distinct patterns in how event categories relate to session duration. Account & Policy Management shows both high frequency and increased session times when active, indicating common but time-intensive operations. In contrast, Action Center & Workflow and Dashboard & UI Interactions display longer session durations but lower frequency when active, suggesting these are less common but more complex interactions. These patterns provide valuable predictive features for modeling session behavior and resource usage, with all categories showing positive correlations to session duration despite their varying frequency patterns.

Roles


uw_max vs session_seconds_mean
Count
Mean session_seconds_mean
uw_max
0
1
350000
300000
250000
200000
150000
100000
50000
0
1000
800
600
400
200

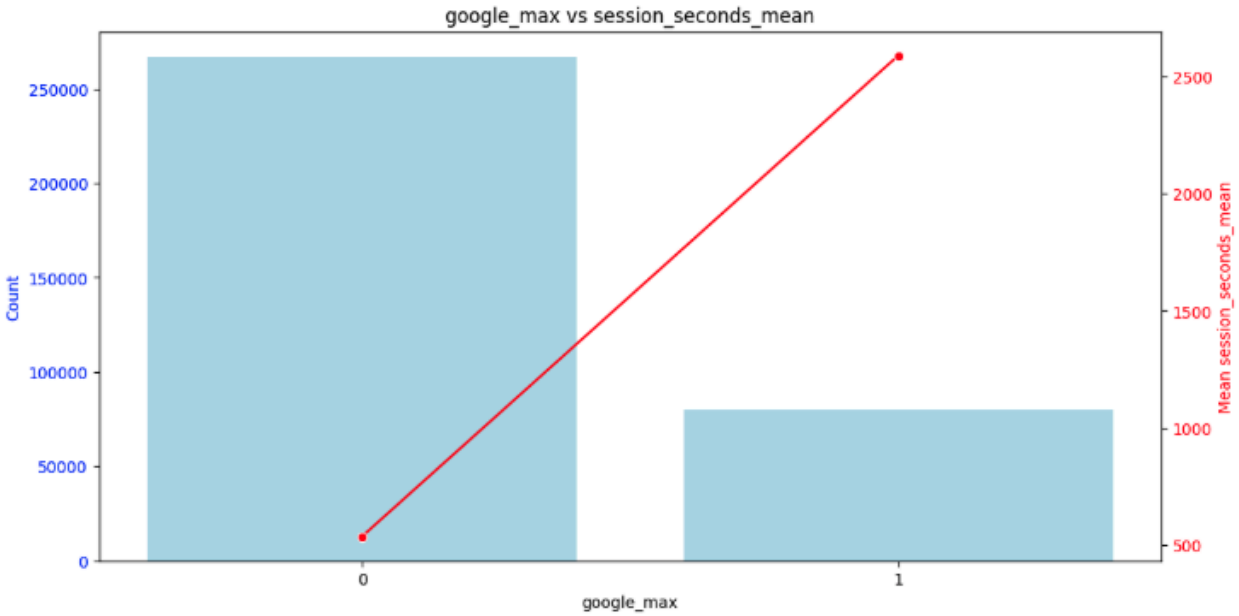

admin_max vs session_seconds_mean
Count
Mean session_seconds_mean
admin_max
0
1
250000
200000
150000
100000
50000
0
2250
2000
1750
1500
1250
1000
750
500

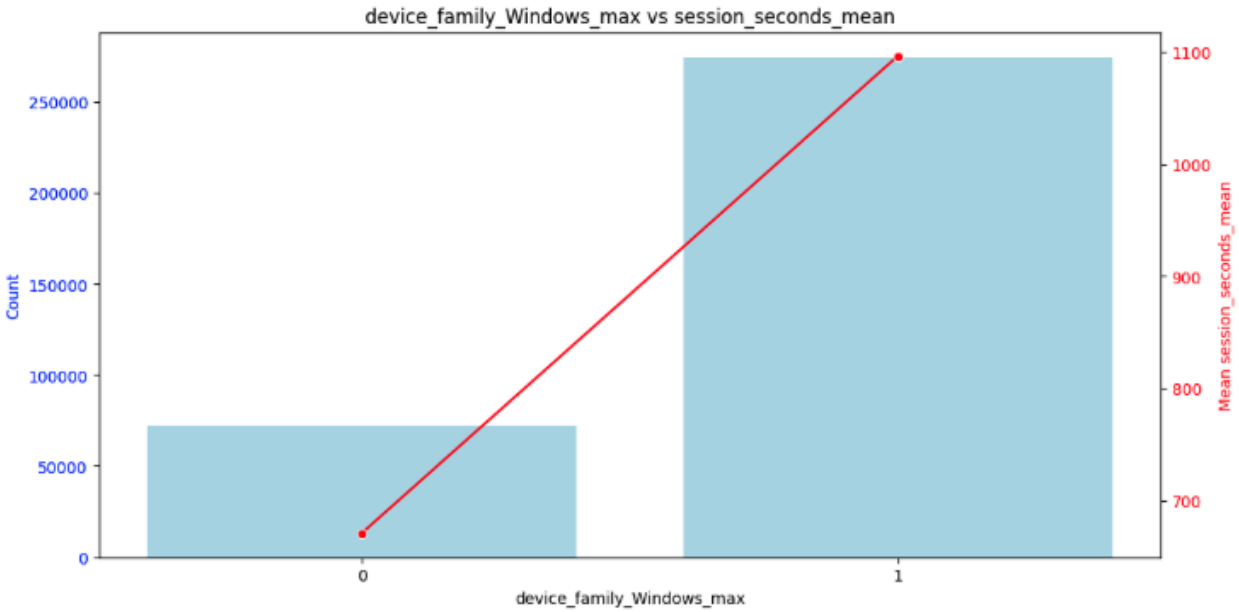

manager_max vs session_seconds_mean
Count
Mean session_seconds_mean
manager_max
0
1
250000
200000
150000
100000
50000
0
2500
2250
2000
1750
1500
1250
1000
750
500



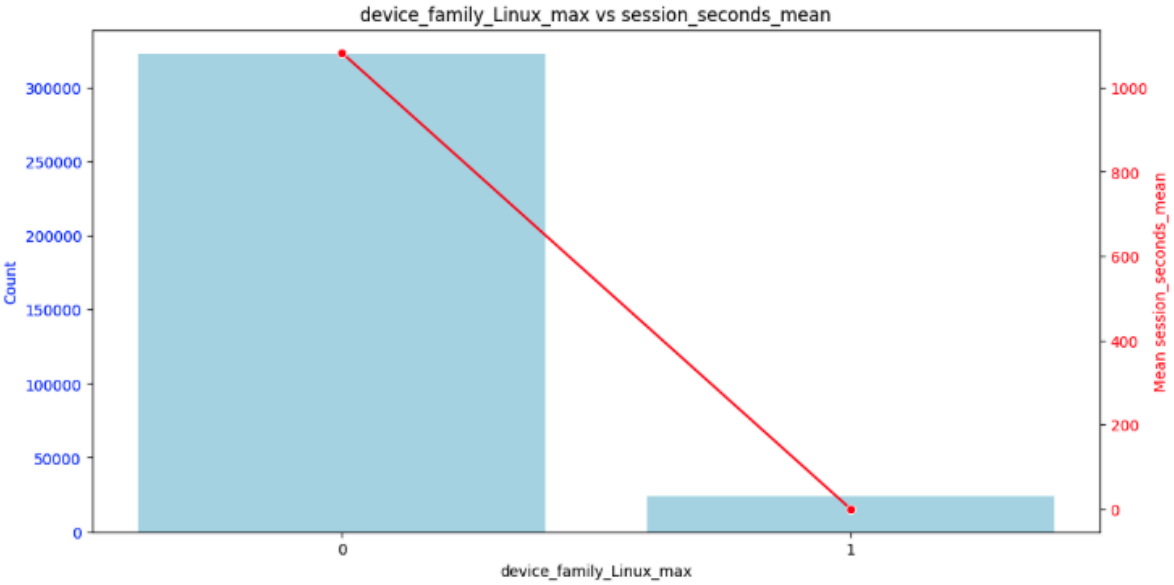
The data shows that Underwriters have much higher activity frequency and increased session durations. In contrast, admin, manager, and broker roles display an inverse pattern, they show lower activity frequency when the role is active but significantly longer session durations. This suggests that while regular users have more frequent but potentially simpler interactions, admin and manager roles engage in less frequent but more time-intensive tasks. All roles demonstrate positive correlations with session duration, making them valuable predictive features for modeling user behavior and resource requirements.

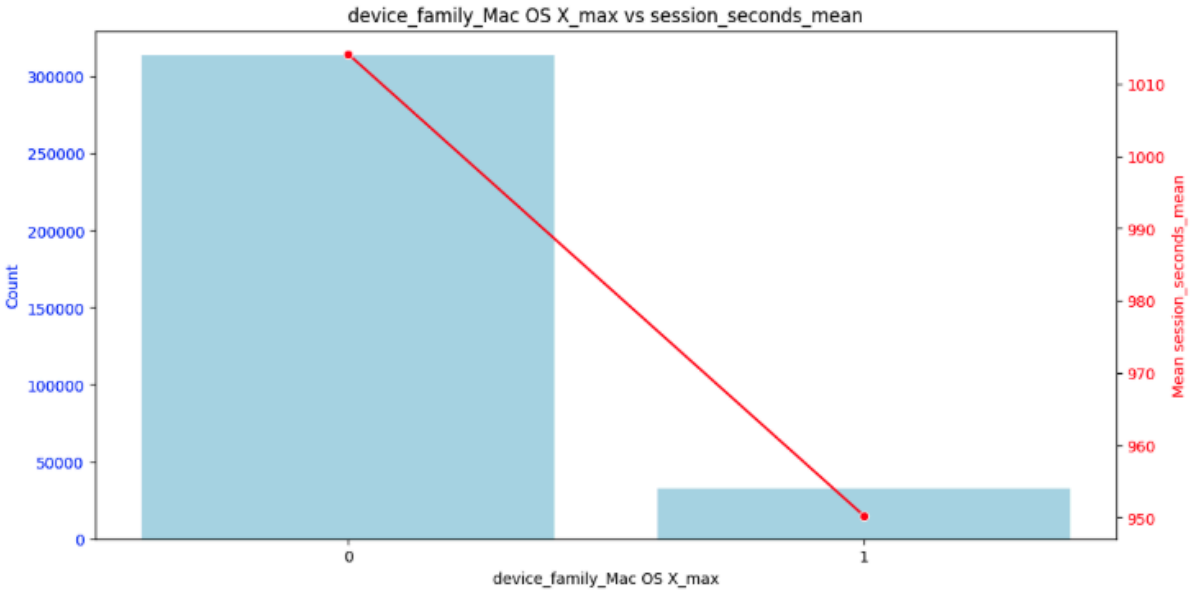
Device family / os





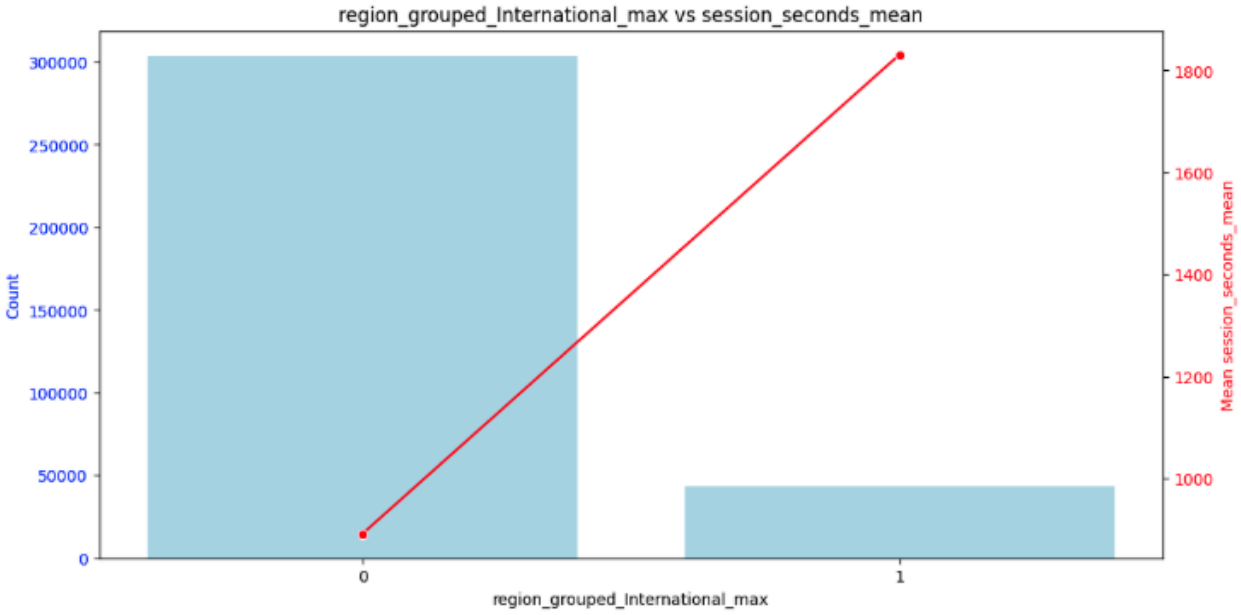
The analysis reveals that when users use Google, session duration is significantly higher, although fewer users fall into this category, indicating that these users tend to have longer sessions. Similarly, Windows users exhibit slightly longer session durations, and their prevalence suggests a meaningful impact on overall session duration predictions.

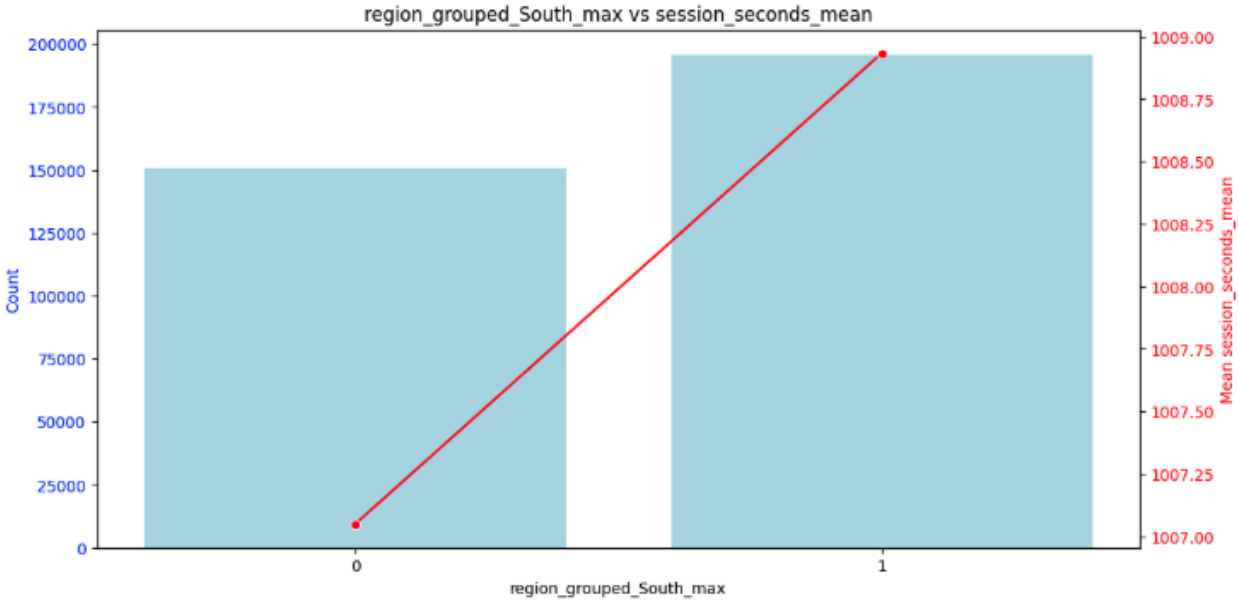




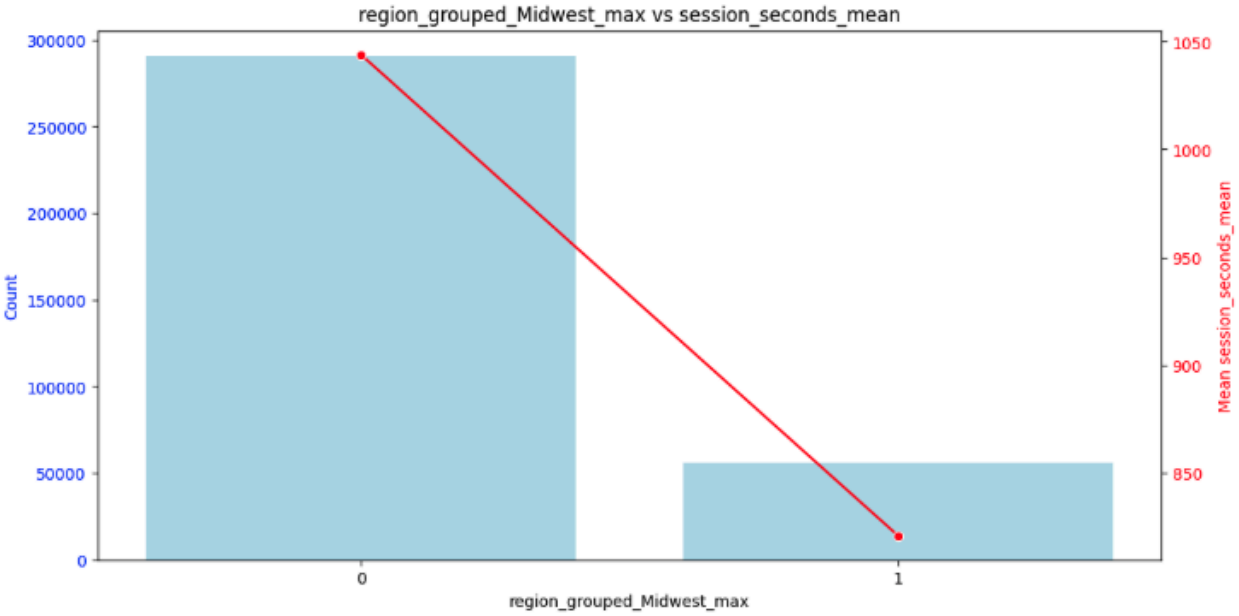
In contrast, Linux and Mac OS users tend to have shorter sessions, with them both showing a marked decrease in session duration. These insights contribute to modeling by highlighting the importance of categorical variables in predicting session duration.

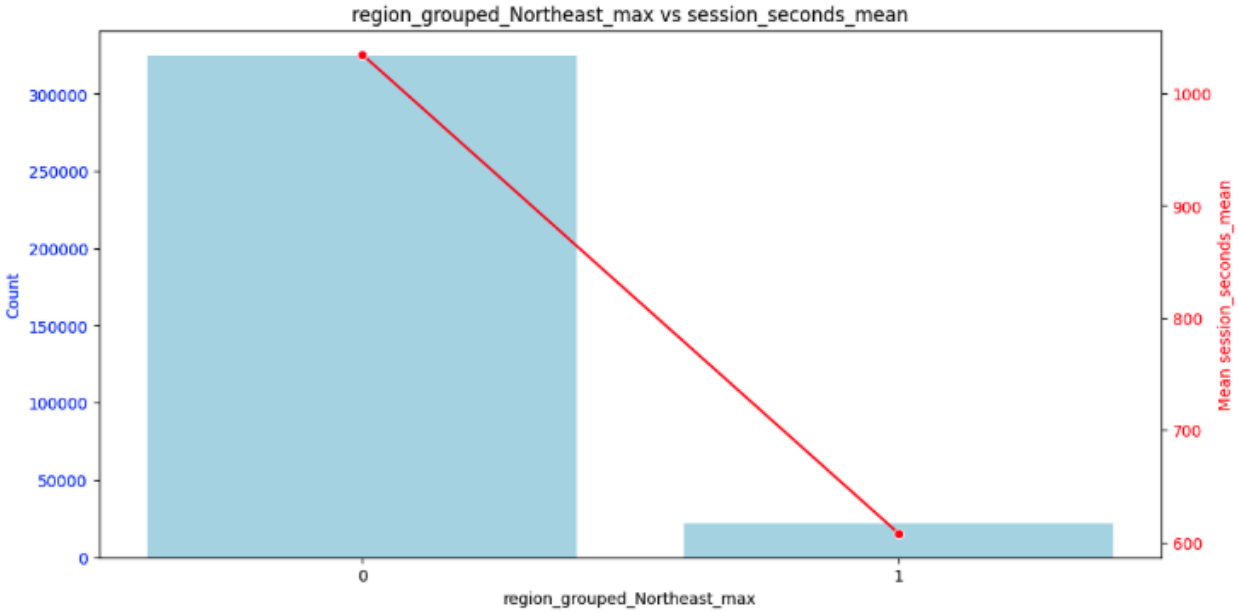
Region

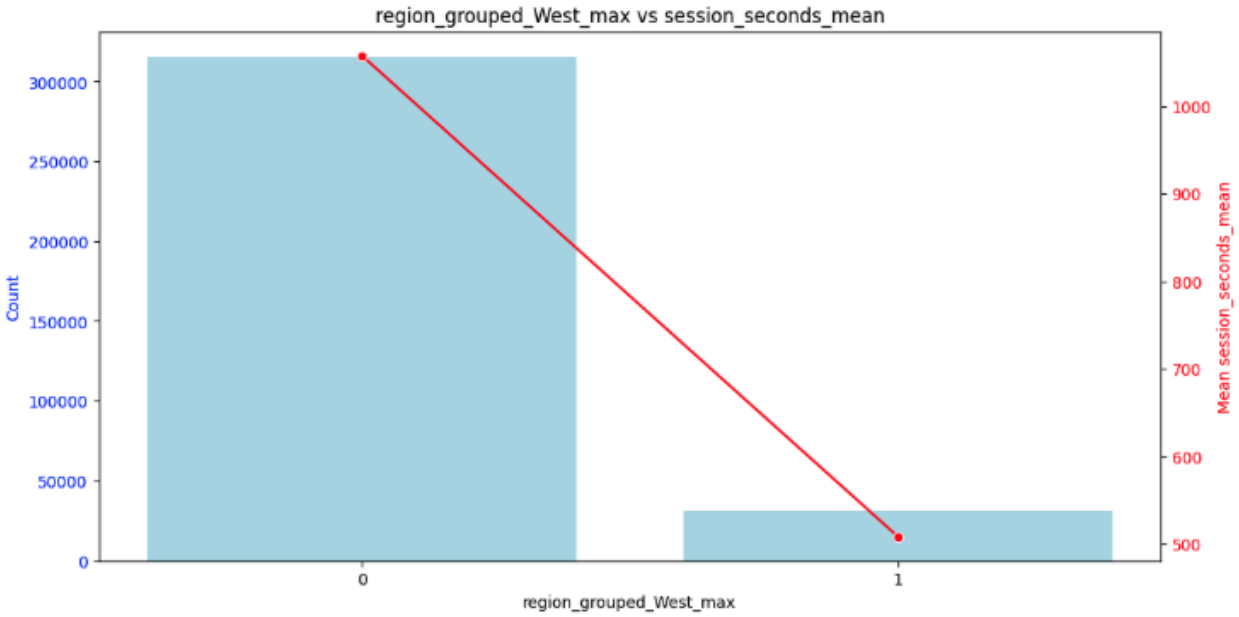




The graphs indicate a positive correlation between the presence of "International" or "South" regional characteristics and mean session duration. Users categorized within these regions exhibit notably longer average session times compared to those without these attributes. This suggests that these regional factors may positively influence user engagement and contribute to extended interaction with the platform or service.

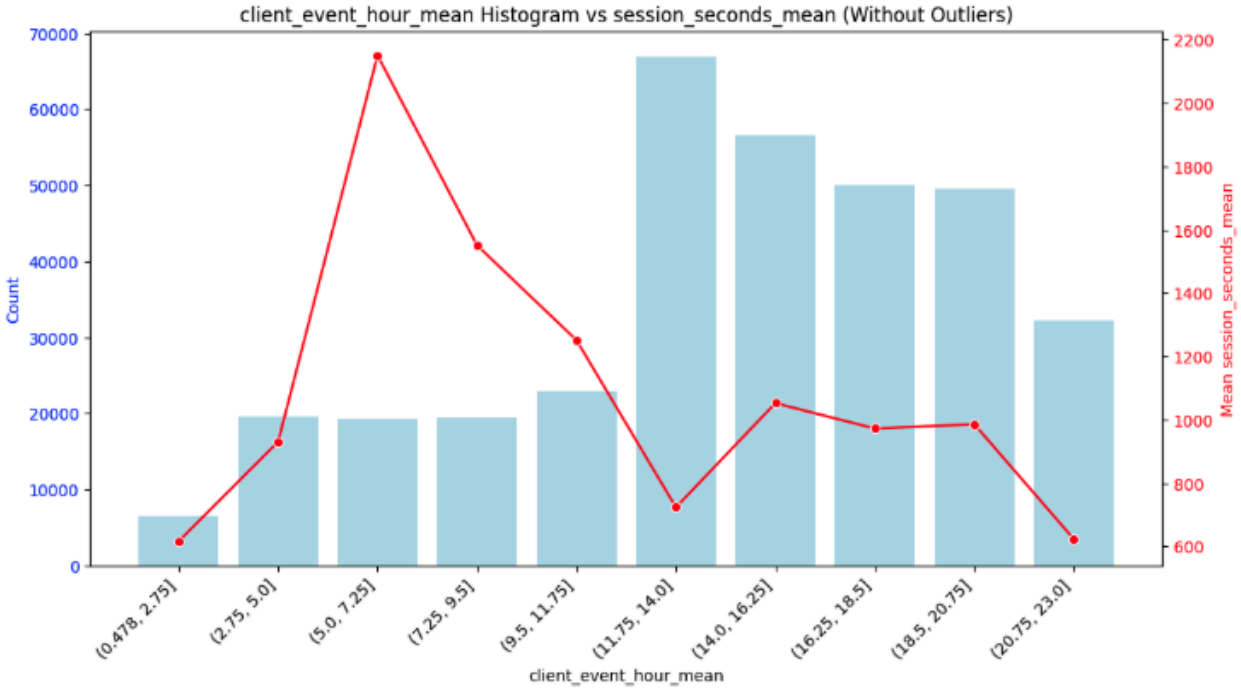






Conversely, the presence of "Midwest," "Northeast," or "West" regional characteristics appears to correlate with a decrease in mean session seconds. Users from these regions demonstrate shorter average session times, indicating a potential negative association between these regional factors and user engagement. Understanding these trends may help tailor strategies to improve user retention and engagement within these specific regions.

User's usage time during the day



The graph reveals a non-linear relationship between the average hour of client events and the mean session duration. Notably, session lengths peak significantly during the early morning hours (5:00 AM to 7:15 AM), while the shortest session durations occur around midday (11:45 AM to 2:00 PM). Although event frequency is highest during the late morning and early afternoon, the substantial variation in session duration across different times of day suggests that client_event_hour_mean is a crucial feature for modeling session_seconds_mean, warranting the consideration of non-linear transformations or encodings to capture its impact effectively or usage of more complex algorithms that can capture non-linear relationship(e.g. XGBoost).

Modeling

1. **User's likely action**

Dataset

Split the dataset into training (80%), validation (10%), and test sets (10%) based on user identifiers. Use a rule to divide the data, ensuring each set represents different user groups or sessions for training, validation, and testing. Extract the sequences and targets from each subset as lists, preparing them for use in creating data batches for the model.

Finally, create a custom dataset class to manage the sequences and targets. This class handles padding shorter sequences to a fixed length and truncating longer ones, ensuring uniformity for model processing. Use a padding value outside the valid range of item identifiers, such as 740, to indicate non-relevant positions in the sequences.

BERT4Rec Structure
The BERT4Rec model builds on a transformer architecture, specifically adapting the BERT model for sequential recommendations. It processes input sequences of item identifiers, padded or truncated to a consistent length, and predicts the next item using a transformer-based encoder.

The model starts with an input layer that accepts sequences of numbers, padded with a special value to maintain uniform length. An attention mask is applied to ignore padded positions, allowing the model to focus on actual interaction data. The core of the model consists of transformer layers, configured with a set number of layers and attention heads, which capture contextual relationships within the sequences. These layers transform the input into hidden representations, capturing patterns and dependencies.

After processing through the transformer layers, the model extracts the hidden state of the last token in each sequence, representing the context of the entire sequence. This hidden state passes through an additional layer to produce output logits, which are numerical scores for each possible item in the vocabulary. The logits correspond to the likelihood of each item being the next action, enabling the model to predict the most probable item.
The model configuration includes parameters such as the vocabulary size, the size of hidden representations, the number of transformer layers, the number of attention heads, and the maximum sequence length. These settings define the model's capacity and behavior, tailored to the dataset's characteristics.

Evaluation
Evaluate the model using accuracy metrics tailored to recommendation systems. Focus on determining whether the correct item is among the top predicted options, such as the top five or ten recommendations. Use the loss value from training and validation to gauge overall model performance, ensuring it aligns with expected outcomes for recommendation tasks.

2. **Retention Model**

Dataset
Split the dataset into training (80%), validation (10%), and test sets (10%) based on user identifiers. This approach ensures that each set represents unique user groups or sessions, maintaining independence between training, validation, and testing data. This division helps the model generalize better by avoiding data leakage across sets.

After splitting the data, extract the input features and target variable (indicating whether a user returns after a session) from each subset. Organize them into lists for efficient batching during model training and evaluation.

XGBoost Structure
The XGBoost model is used for predicting customer retention, leveraging gradient boosting techniques for robust and accurate classification. XGBoost is chosen for its capability to handle

complex patterns and interactions within the features, making it suitable for modeling user behavior and retention.

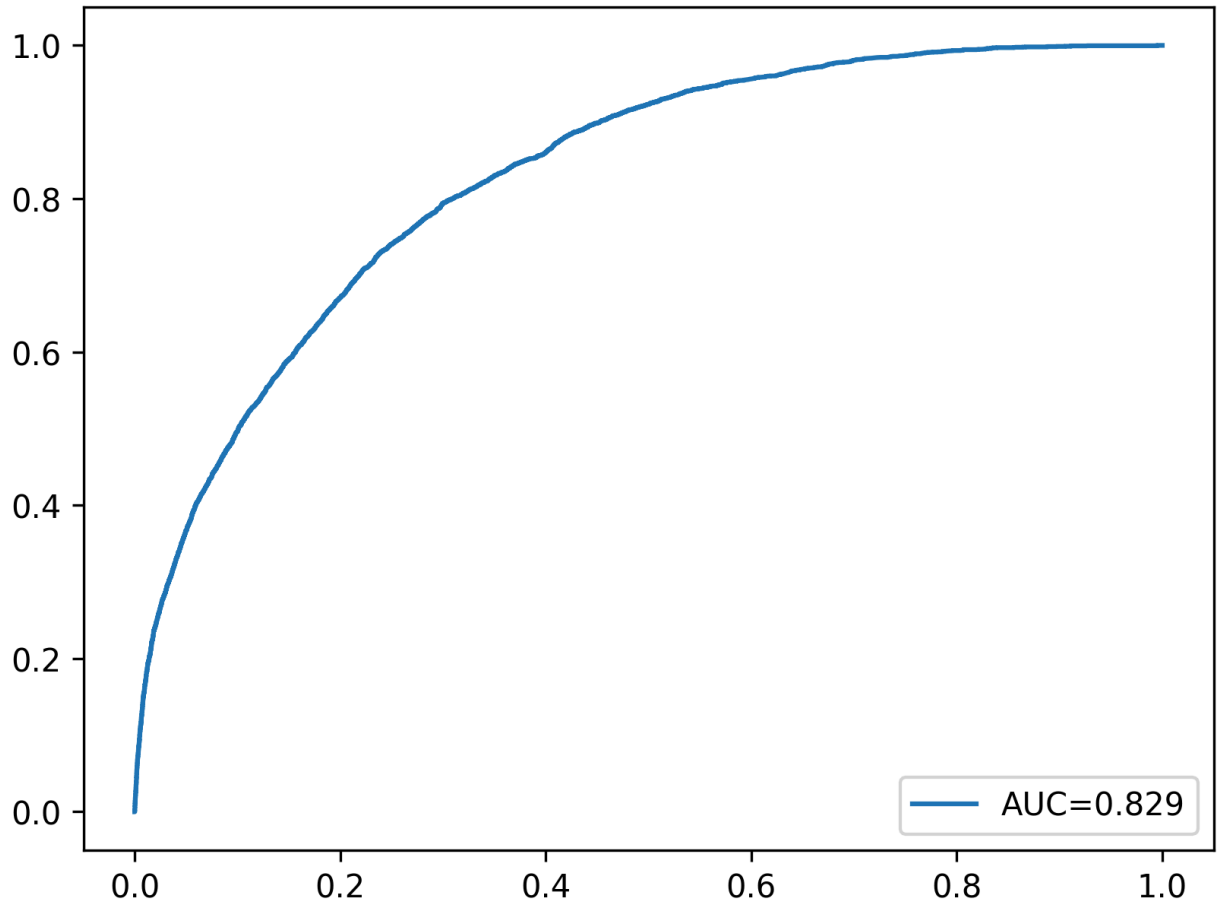
The model begins by taking the input features, standardized to ensure consistent scaling and improved model convergence. It utilizes decision trees as base learners, gradually refining predictions by learning from the residuals of previous iterations. This boosting approach allows XGBoost to effectively capture non-linear relationships and interactions within the dataset.

Key model parameters include the number of trees (estimators), maximum tree depth, learning rate, regularization terms, and subsampling ratios. These settings control the model's complexity and generalization, ensuring it balances bias and variance effectively. Hyperparameter tuning is conducted to optimize these values for the best performance on the validation set.

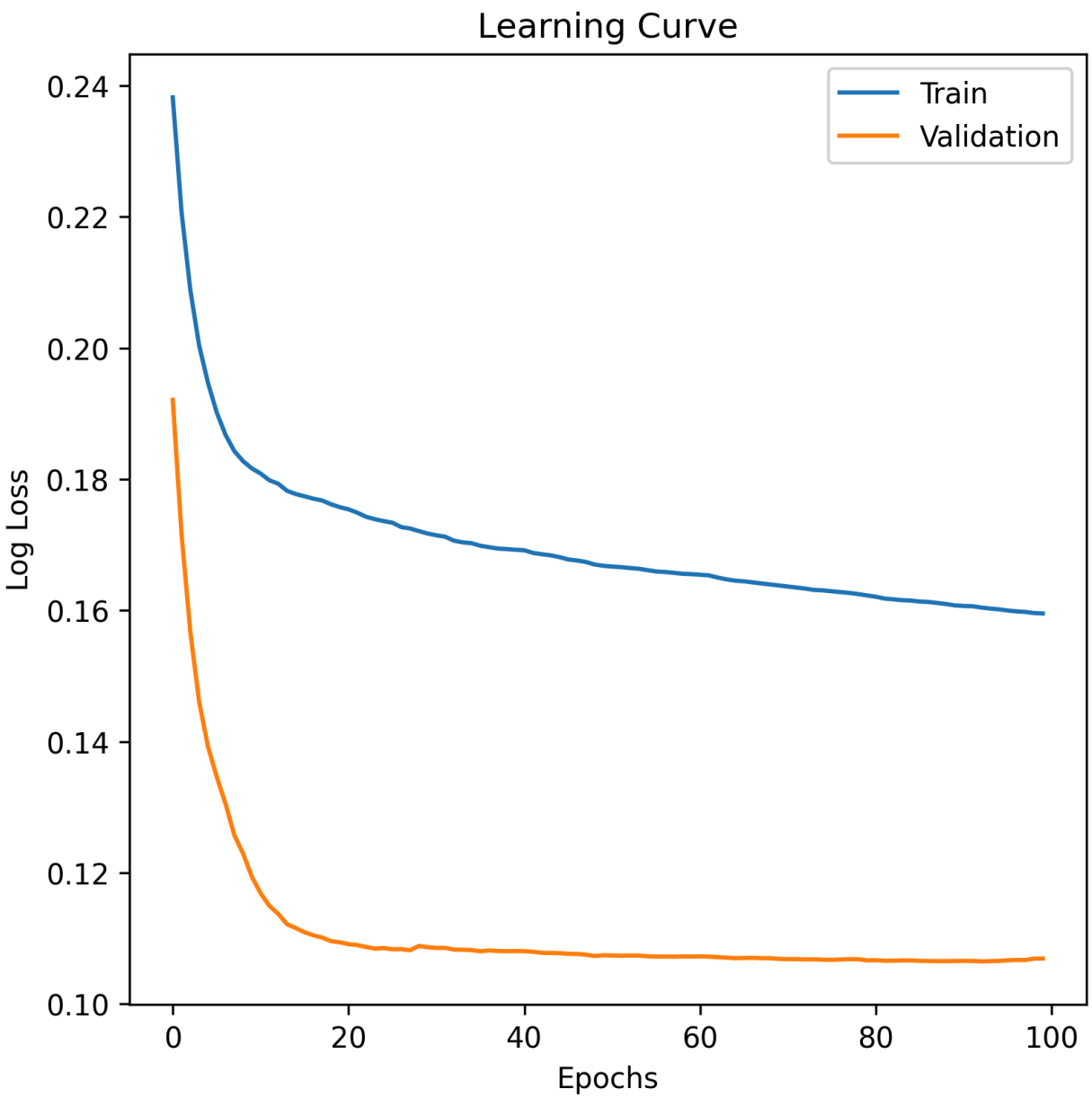
Evaluation

Evaluate the model using the ROC AUC metric, as it measures the model’s ability to distinguish between the two classes (retained vs. not retained). ROC AUC is particularly suitable for this task because it provides a threshold-independent measure of performance, focusing on the model's ranking capabilities rather than accuracy at a specific threshold.

During training, monitor the loss value on both training and validation sets to ensure proper model learning and to detect overfitting. Additionally, track the ROC AUC on the validation set to gauge how well the model generalizes to unseen data. The final evaluation on the test set provides an unbiased estimate of the model's performance in predicting customer retention. The following are the visualization graph of ROC AUC, learning curve and feature importance, respectively.

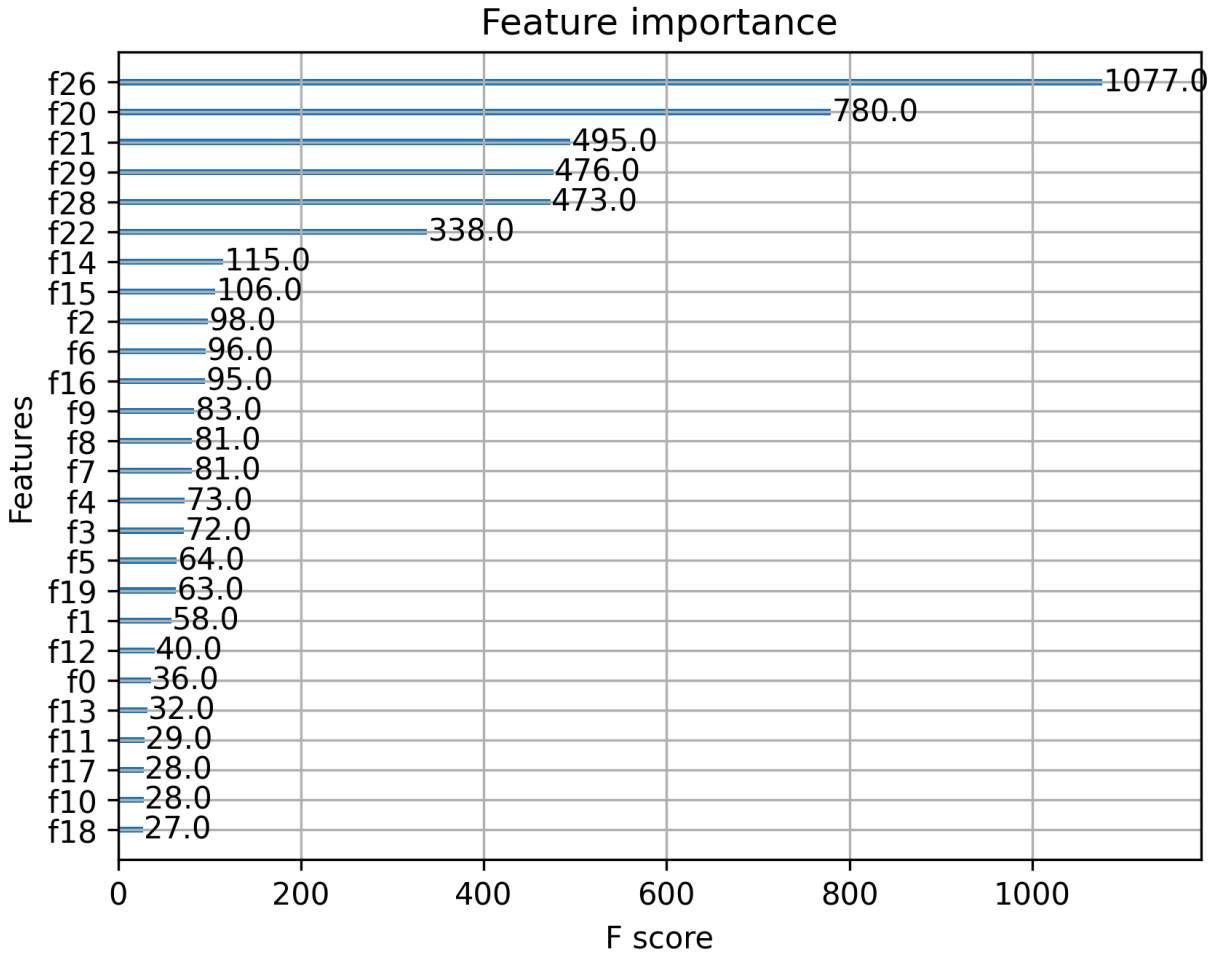


The ROC curve provides a graphical representation of the model's ability to distinguish between classes. With an AUC score of 0.829, the model exhibits strong discriminatory power, effectively differentiating between positive and negative classes. A perfectly random model would yield an AUC of 0.5, while an ideal model would score 1.0. The current AUC score, positioned in the high 0.8 range, indicates that the classifier is performing well. However, minor adjustments to hyperparameters, feature selection, or handling of class imbalances could potentially push the AUC even higher. This performance insight suggests that the model is robust and reliable, making it well-suited for real-world deployment with minimal risk of misclassification.The ROC curve provides a graphical representation of the model's ability to distinguish between classes. With an AUC score of 0.829, the model exhibits strong discriminatory power, effectively differentiating between positive and negative classes. A perfectly random model would yield an AUC of 0.5, while an ideal model would score 1.0. The current AUC score, positioned in the high 0.8 range, indicates that the classifier is performing well. However, minor adjustments to hyperparameters, feature selection, or handling of class imbalances could potentially push the AUC even higher. This performance insight suggests that the model is robust and reliable, making it well-suited for real-world deployment with minimal risk of misclassification.



The learning curve demonstrates the model's training progress, with both training and validation log loss decreasing over time. Initially, both curves show a steep decline, indicating rapid learning in the early epochs. However, after approximately 20 epochs, the validation loss stabilizes at around 0.10, while the training loss continues to decrease gradually. The small gap between the training and validation curves suggests that the model is not overfitting, meaning it generalizes well to unseen data. However, if further improvements are desired, regularization techniques (such as tuning lambda and alpha) or early stopping adjustments could be

considered to fine-tune generalization.



The feature importance plot from the XGBoost classifier highlights the most influential factors driving predictions. The top-ranked features include session duration (session_seconds_mean, f26) with an F-score of 1077, indicating that how long users stay engaged plays a crucial role in classification. Event timing-related features such as client_event_hour_mean (f21) and server_received_hour_mean (f20) are also highly ranked, reinforcing the idea that when users interact with the system significantly affects predictions. Additionally, processing_time_mean (f28), which measures how long it takes to process user interactions, is another major contributor. Interestingly, slug-based encoding (slug_encoded_mean, f29) is also among the top contributors, suggesting that specific event types or user actions correlate with the target variable. The dominance of time-based features in the model emphasizes the importance of user behavior patterns over time, which can be leveraged to further refine predictive capabilities.

Validation Set Metrics:

Accuracy:	F1:	AUC:	Confusion Matrix:
0.9634524228263236	0.2076978939724037 6	0.9097137534273037	[[57235 276] [1906 286]]

Test Set Metrics:

Accuracy:	F1:	AUC:	Confusion Matrix:
0.9318909618111428	0.2254031527450625	0.8273042584585741	[[57870 406] [3869 622]]

3. Time Usage Model

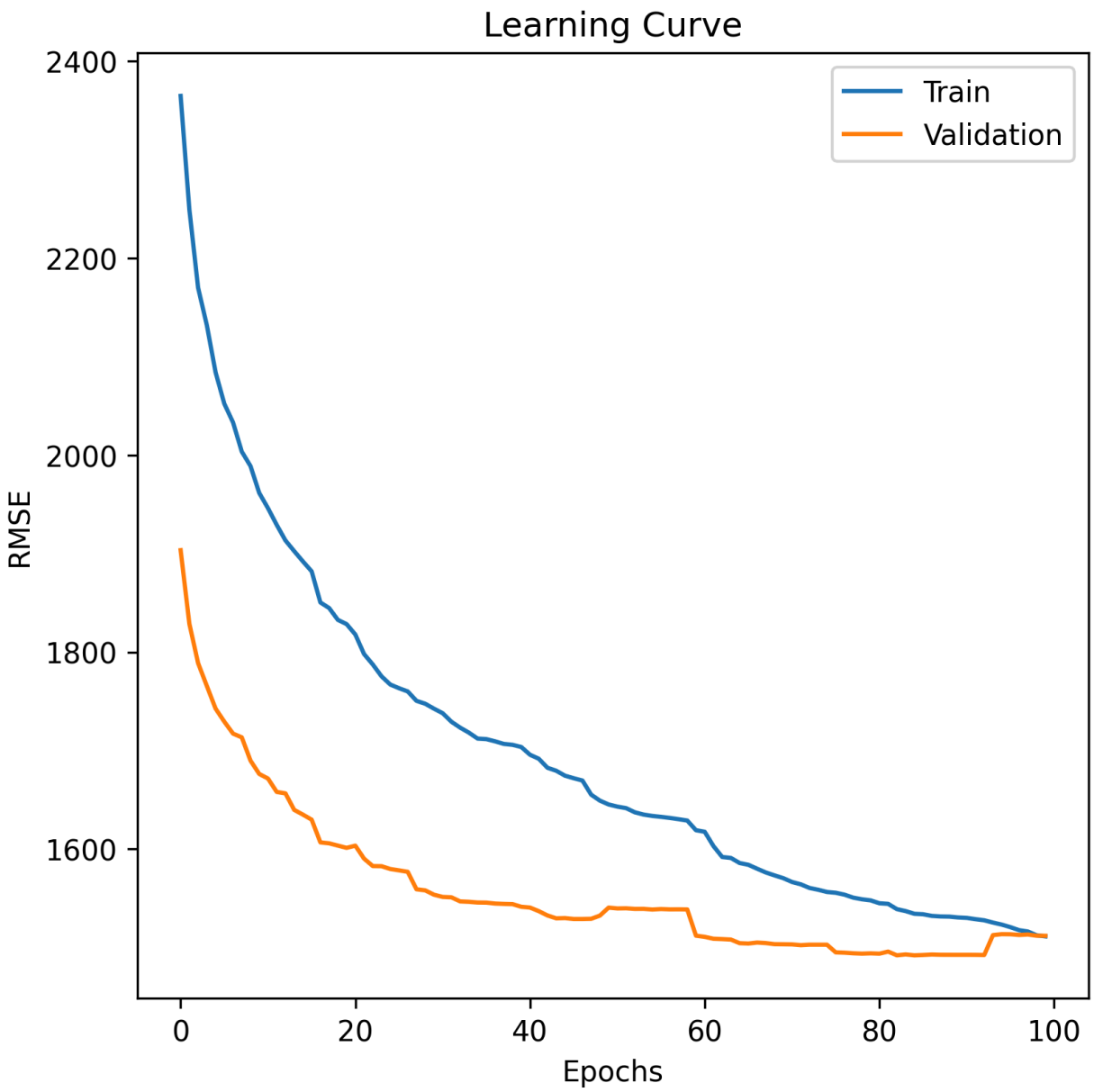
Dataset
We use the same dataset as the retention model but instead of the target variable ‘returned_within_28_days_max’ we use ‘session_seconds_mean’.

XGBoost Structure
The XGBoost regression model is used to predict time usage per session, leveraging gradient boosting to provide accurate and robust continuous value predictions. XGBoost is chosen for its ability to handle complex feature interactions and non-linear relationships, making it suitable for modeling time-based patterns in user behavior.

The model begins by taking the input features. It uses decision trees as base learners, refining predictions by minimizing the residuals from previous iterations. This boosting technique enables the model to capture intricate patterns within the data effectively.

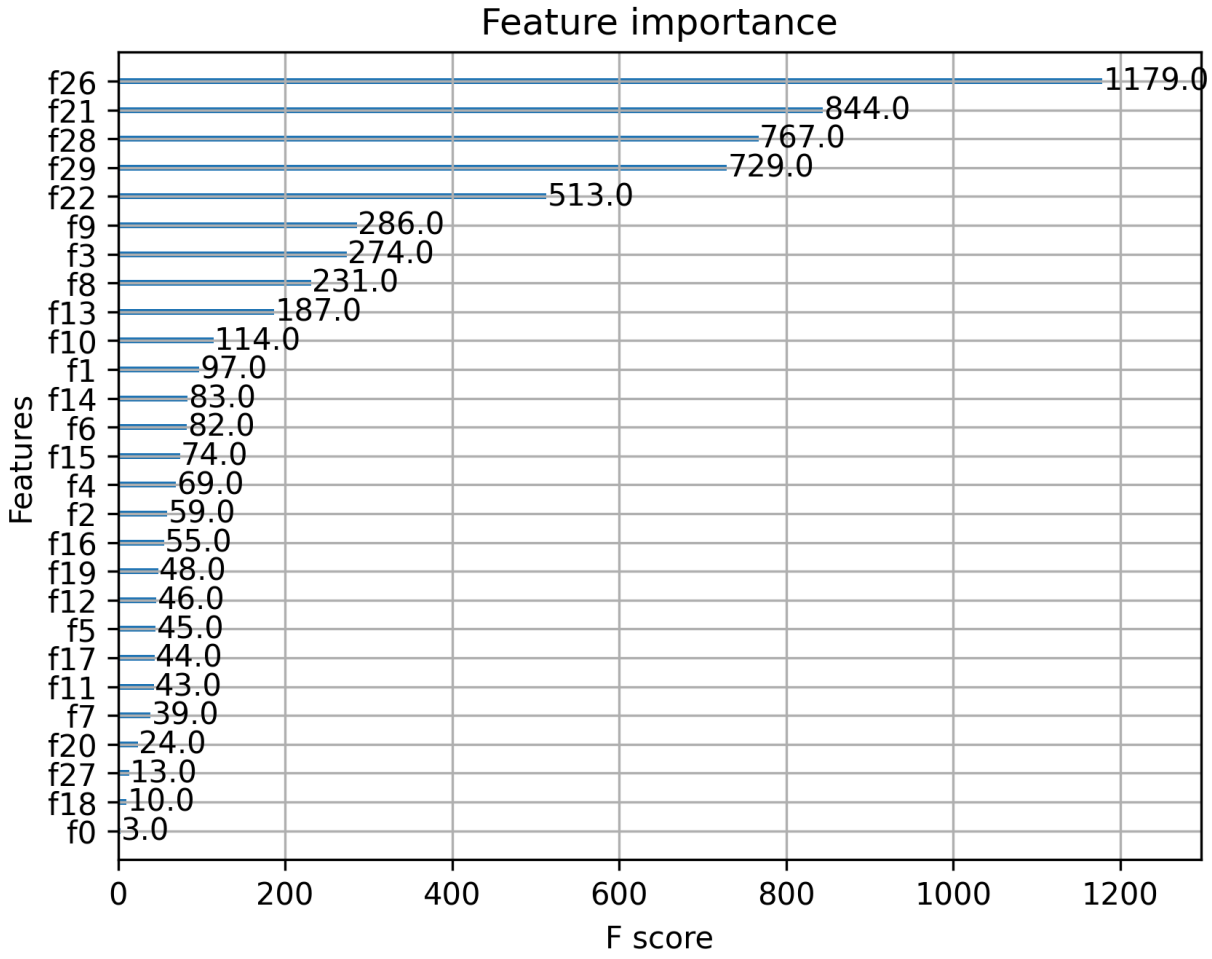
Key model parameters include the number of trees (estimators), maximum tree depth, learning rate, regularization terms, and subsampling ratios. These parameters control the model's complexity and its ability to generalize. Hyperparameter tuning is conducted to optimize these values, ensuring the best performance on the validation set.

Evaluation
Evaluate the model using Root Mean Squared Error (RMSE), which measures the average magnitude of the prediction errors. RMSE is chosen because it penalizes large errors more significantly than smaller ones, making it suitable for assessing the accuracy of time usage predictions.



During training, monitor the loss value on both training and validation sets to ensure the model learns effectively and to detect overfitting. Additionally, track the RMSE on the validation set to evaluate how well the model generalizes to unseen data. The final evaluation on the test set provides an unbiased estimate of the model's performance in predicting session time usage. The learning curve plots the Root Mean Squared Error (RMSE) over 100 epochs for both training and validation sets. Initially, both curves show a steep decline, indicating that the model is learning quickly. However, while the training RMSE keeps decreasing, the validation RMSE flattens around epoch 60-80, suggesting diminishing returns in model improvement. The slight gap between training and validation loss suggests mild overfitting, where the model performs slightly better on training data than on unseen data. While this is not severe, adding

regularization (lambda, alpha) or applying early stopping could improve generalization.



The feature importance plot for the XGBoost regressor highlights the most influential variables affecting the model's predictions. The most important feature is session_seconds_mean(f26), with the highest F-score of 1179, followed by client_event_hour_mean(f21), processing_time_mean(f28), and server_received_hour_mean(f29), each contributing significantly to the model's predictions. These features suggest that the duration of user sessions, the time of events, and the processing time play critical roles in determining the target outcome. Other influential features include slug_encoded_mean, which may capture categorical event encoding patterns. On the other hand, features such as google_max(f27), microsoft_max(f18), and device_family_linux_max(f0) have low importance, indicating they contribute minimally to the model's predictions.

Given the learning curve and feature importance, the model appears to perform reasonably well. The training and validation errors follow a decreasing trend, which means the model is learning from data effectively. However, since the validation RMSE does not decrease significantly after a certain point, it may indicate that further fine-tuning of hyperparameters is needed. Additionally, an analysis of residual errors could help determine whether certain features

contribute to systematic biases in the predictions. By leveraging these insights, the model can be further optimized for better generalization and performance in real-world applications.

Test Set Metrics:

R2:	MSE:	MAE:
0.3668431317573806	2361934.776503033	200.1507568359375

Recommendations

A/B Testing

Based on our comprehensive analysis of user behavior, retention patterns, and session duration metrics, we propose the following specific A/B testing strategies to validate their effectiveness:

1. Enhanced User Onboarding for Non-Underwriters

Given the significant disparity in platform engagement between underwriters and non-underwriters, we recommend implementing role-specific onboarding experiences:

- Create role-specific tutorial workflows
- Implement contextual help features
- Develop quick-access toolsets for common tasks

Potential Approach:

- Treatment Group: Standard onboarding
- Control Group: Role-optimized onboarding experience
- Key Metrics: 28-day retention rate, feature adoption rate, time-to-first-action
- Duration: 6 weeks with a 30/70 split (new users only)
- Success Criteria: 20% improvement in 28-day retention rate for non-underwriters

2. Time-Sensitive Feature Availability

Based on the analysis of user activity patterns throughout the day, we recommend optimizing feature availability and system resources:

- Prioritize system performance during peak hours (2 PM - 7 PM)
- Schedule maintenance during low-activity periods
- Implement time-based feature promotions

A/B Testing Approach:

- Treatment Group: Static feature presentation
- Control Group: Time-optimized feature prominence
- Key Metrics: Feature engagement rate, session duration, user satisfaction scores
- Duration: 3 weeks with 50/50 split
- Success Criteria: 10% increase in feature engagement during target hours

3. Action-Based Retention Interventions

Based on the BERT4Rec model's predictions of user actions, we recommend implementing smart interventions:

- Trigger contextual help based on predicted next actions
- Implement smart defaults based on user behavior patterns
- Develop proactive notification systems

A/B Testing Approach:

- Treatment Group: Standard interface
- Control Group: Predictive assistance interface
- Key Metrics: Task completion rate, time-to-completion, user satisfaction
- Duration: Possibly 75 weeks with 20/80 split
- Success Criteria: 30% reduction in task abandonment rate

Implementation Priority

We recommend implementing these changes in the following order:

1. Role-Specific Onboarding (Critical for non-underwriter retention)
2. Action-Based Interventions (Leverages existing predictive capabilities)
3. Time-Sensitive Features (Operational efficiency)

Monitoring and Iteration

For each A/B test:

- Establish clear baseline metrics before implementation
- Monitor both primary and secondary metrics to catch unexpected effects
- Set up automated alerting for significant metric deviations
- Plan for a gradual rollout of successful features
- Document learnings from failed tests for future iterations

Following these recommendations and carefully executed A/B testing strategies should lead to meaningful improvements in user retention, engagement, and overall platform effectiveness.

Regular review and adjustment of these strategies based on test results will ensure continuous optimization of the user experience.