Churn prediction from streaming service logs

Team members	Fang Zhang, Zixiao Guo
Jury	Felix Lefebvre flefebv
Module	Python for Data Science
Date	December 15, 2025

1. Introduction

Customer churn refers to the decision of a user to stop using a service. In subscription-based digital platforms, churn is a key business concern, as retaining existing users is typically less costly than acquiring new ones. Predicting churn can therefore help identify high-risk users and support targeted retention strategies.

In this project, we study churn prediction using detailed user activity logs. Each user generates a sequence of timestamped events reflecting listening behavior, platform interactions, and account-related actions. Churn is observed when a user explicitly confirms cancellation. The objective is to predict future churn based on past observed behavior.

This task presents several practical challenges. User behavior is observed over time, while the churn decision occurs at a specific moment, which makes the temporal framing of the prediction problem critical. In addition, user activity levels vary widely, and churn events represent only a small fraction of users.

A central challenge is therefore to design a temporal modeling strategy that captures meaningful behavioral signals while avoiding information leakage from actions occurring too close to the churn decision. The following sections explore two alternative approaches that address this issue in different ways.

2. Temporal Modeling Strategies

An important challenge in churn prediction is defining when the prediction is made. Unlike static classification tasks, churn is a time-dependent phenomenon: user behavior evolves over time, while the churn decision occurs at a specific moment. The choice of temporal framing therefore directly affects label definition, feature construction, and the risk of information leakage.

We consider two alternative temporal modeling strategies. Both rely on similar behavioral features but differ fundamentally in how observation windows and prediction targets are defined.

Method A — User-Specific Anchor Date (Event-Driven Window)

Our first approach models churn as a user-specific event, anchored on each user's own behavioral timeline.

For users who churn, the anchor date is defined as the timestamp of their first Cancellation Confirmation event. For non-churn users, a fixed global cutoff date (2018-11-20) is used. From this anchor date, we extract behavioral features over a backward-looking window of fixed length (empirically chosen as 9 days). Only events strictly before the anchor date are retained.

This framing preserves heterogeneity in churn timing and allows each user to be observed relative to their own outcome. However, it raises concerns about near-outcome information leakage. An analysis of user behavior shows that approximately 95% of churners visit the Cancel page within six minutes prior to confirmation, suggesting that events immediately before confirmation may encode almost deterministic information about the label.

To address this, we experiment with introducing a short buffer before the anchor date, excluding events occurring shortly before confirmation. In practice, this adjustment removes highly predictive signals, which motivates exploring an alternative formulation that avoids user-specific anchor dates altogether.

Method B — Fixed Risk-Set Window (Global Time Framing)

Our second approach reframes churn prediction as a risk-set classification problem at a fixed point in time.

We select a global reference date (2018-11-20) and define a churn window immediately preceding it (e.g. the last 10 days). Users who churned before this window are removed from the dataset. The remaining churners are those who churn within the window, while all others are treated as non-churn at that time.

For all remaining users, features are constructed from a backward-looking window ending at the start of the churn window. In practice, we find that a look-back period of around 40 days provides stable performance. This design ensures a clear temporal separation between observed behavior and future churn, eliminating near-outcome leakage by construction.

Compared to Method A, this strategy avoids user-specific anchor dates, defines a consistent prediction task across users, and yields a cleaner temporal setup, at the cost of discarding early churners and reducing the effective sample size.

3. Modeling Choice & Feature Engineering

3.1 Modeling Choice

Logistic Regression is chosen as the baseline model for this churn prediction task because it provides a well-defined probabilistic framework for binary outcomes and behaves robustly under moderate feature dimensionality. The model directly estimates the probability that a user will churn, producing outputs bounded between 0 and 1, which aligns well with the objective of predicting individual churn risk rather than hard class labels.

From a modeling perspective, this probabilistic interpretation suits the churn setting, where users exhibit different levels of churn risk and classification thresholds may vary depending on how the predicted probabilities are used in practice. In addition, the linear structure of Logistic Regression offers the practical advantage that each feature contributes additively to the log-odds of churn, making the direction and relative magnitude of effects transparent. This facilitates analysis of behavioral patterns associated with higher or lower churn risk which complements the model's predictive role.

3.2 Feature Engineering

Behavior Counts

This category includes 18 page-level interaction counts, capturing the total number of visits to each distinct page type during the observation window. These features are:

count_Add_Friend, count_Add_to_Playlist, count_Downgrade, count_Error, count_Help, count_Home, count_NextSong, count_Roll_Advert, count_Thumbs_Down, count_Thumbs_Up, count_Upgrade, along with the remaining page-specific count variables. Together, they quantify explicit user actions and directly reflect engagement, dissatisfaction, or friction with the platform.

Activity Intensity

These features summarize how actively a user interacts with the service in aggregate terms. They include length_sum (total listening duration in seconds), num_sessions (number of unique sessions), total_interactions (total number of logged actions across all pages), and active_days (number of distinct days with activity). This group distinguishes consistently active users from those with sparse or irregular usage patterns.

Engagement Extent and Dynamics

This category captures the quality and temporal structure of engagement beyond raw volume. It includes last_3_days_count (number of actions in the final three days of the observation window), songs_per_session (average number of songs played per session), avg_daily_listen_time (average listening duration per active day), and recent_trend (ratio of

actions in the last three days relative to total interactions). These features describe short-term dynamics and intensity shifts in user behavior.

Static Encoded Features

Finally, time-invariant user attributes provide contextual information that frames observed behavior. These include tenure_days (days since user registration), gender_code (binary encoding of gender), and level_code (binary encoding of subscription status: paid versus free).

Overall, this feature construction strategy balances expressiveness and parsimony, producing a comprehensive yet manageable representation of user behavior suitable for downstream modeling and feature selection.

3.3 Feature Selection

The initial feature set constructed from user logs is relatively large and contains several highly correlated variables, such as total listening duration (length_sum) and song-related activity counts (e.g., count_NextSong). Including all candidate features would increase the risk of overfitting and lead to unstable coefficient estimates, particularly given the linear nature of Logistic Regression. Strong correlations among inputs can inflate variance and obscure the individual contribution of each feature, reducing both interpretability and robustness.

Recursive Feature Elimination (RFE)

To address this, we apply Recursive Feature Elimination (RFE) with Logistic Regression as the base estimator. RFE is a wrapper-based selection method that iteratively fits the model, ranks features according to their coefficient magnitude, and removes the least informative ones at each step. By embedding feature selection directly within the chosen model, RFE ensures that the retained features are those that contribute most effectively to predictive performance in the Logistic Regression setting. Using this procedure, we retain a final subset of 18 features:

count_Add_Friend, count_Add_to_Playlist, count_Downgrade, count_Error, count_Help, count_Home, count_NextSong, count_Roll_Advert, count_Thumbs_Down, count_Thumbs_Up, count_Upgrade, active_days, total_interactions, length_sum, tenure_days, songs_per_session, level_code, and avg_daily_listen_time.

This reduced feature set balances predictive power and model stability while preserving interpretability.

4. Evaluation and Results

Models are evaluated on a stratified validation split at the user level. Given class imbalance and the asymmetric cost of churn prediction errors, we focus on precision, recall, and confusion matrices, with particular attention to the churn class (label = 1), rather than accuracy alone.

Method A — User-Specific Anchor Date

Under Method A, validation performance without any buffer appears exceptionally strong, with near-perfect confusion matrices and very high precision and recall for churners. At first glance, this suggests a highly effective model. However, such results are likely too optimistic given the temporal proximity between observed behavior (Cancel page visits occurring minutes before confirmation) and the churn event.

```
# Method A — No buffer before churn confirmation

Confusion Matrix (Validation Set)

                Predicted 0    Predicted 1
Actual 0            2155              6
Actual 1              19            835
```

```
# Method A — 6-minute buffer before churn confirmation

Confusion Matrix (Validation Set)

                Predicted 0    Predicted 1
Actual 0            1620            540
Actual 1             177            673
```

When a short buffer is introduced before the churn confirmation event, validation performance deteriorates sharply: precision and recall for churners drop substantially, and the confusion matrix becomes noticeably noisier. This change reveals that a large share of Method A's predictive power is driven by user actions occurring immediately before confirmation, which may encode near-deterministic signals of churn.

Therefore, validation metrics under Method A become highly sensitive to the exact anchor date definition and buffer length. This sensitivity makes it difficult to reliably assess model quality or to tune temporal parameters based on standard validation results alone.

Method B — Fixed Risk-Set Window

Method B yields more conservative but substantially more stable validation results. Precision for churners remains low, reflecting both the rarity of churn and the structure of the risk-set design. By removing users who churned before the prediction window, the remaining validation set becomes more imbalanced, which naturally increases the number of false positives observed in the confusion matrix.

Importantly, these performance patterns are stable across different temporal specifications. As the look-back window used for feature extraction increases from 10 days, recall improves and overall performance stabilizes around a 40-day window. Extending the look-back period beyond this point does not materially change the confusion matrix, suggesting limited additional information in older user behavior. Based on this observation, a 40-day look-back window is selected.

Given the high number of false positives, we further adjust the decision threshold. Empirically, a threshold around 0.56 provides a more balanced trade-off between precision and recall for this setting.

Overall, Method A serves as a useful reference but exhibits strong sensitivity to temporal design choices. Method B, by contrast, offers a more stable evaluation framework and clearer separation between observed behavior and future churn, which makes it better suited for robust churn risk modeling.

```
# Method B — Fixed Risk-Set Window

Confusion Matrix (Validation Set)

                Predicted
              Non-Churn   Churn
Actual
Non-Churn          2121       802
Churn                48        74
```

5. Conclusion

This study shows that churn prediction is highly sensitive to how the prediction time is defined.

Among the two temporal formulations we explored, the Fixed Risk-Set Window (Method B) provides the most reliable and interpretable results, as it avoids near-outcome leakage by construction and yields stable validation performance.

The selected features suggest that churn is primarily associated with patterns of disengagement and friction: increased exposure to adverts, negative feedback signals, and downgrade-related actions are indicative of higher churn risk, while positive engagement signals and social interactions are associated with retention. These findings are consistent with intuitive user behavior and remain stable across different look-back window lengths.

Overall, this work highlights the importance of temporal problem framing in churn modeling and shows that conservative, leakage-aware designs can offer more trustworthy insights than seemingly high-performing but fragile alternatives.