Music Streaming Churn Prediction

A Journey Through Temporal Leakage and Distribution Shift

Guust Goossens & Litong Hou | Python for Data Science | December 2025

Abstract

This report documents our approach to predicting user churn for a music streaming service using event-level interaction data. What began as a straightforward classification problem evolved into an exploration of temporal data leakage and distribution shift. Our initial models achieved 88% cross-validation accuracy but collapsed to random-chance performance on the test set. Through systematic investigation, we identified that cumulative features inadvertently encoded the churn outcome. By implementing a fixed-window approach ensuring equal observation time for all users, we developed models achieving 62.3% test accuracy, demonstrating that understanding data characteristics matters more than algorithmic sophistication.

1 Introduction

Customer churn prediction is one of the most important challenges facing subscription-based businesses. With streaming services and subscription models becoming increasingly dominant across industries, this is a problem we may well encounter in our future careers. Research consistently shows that acquiring new customers costs five to seven times more than retaining existing ones, making proactive churn prevention a critical business capability. For music streaming services operating in a competitive market, identifying users likely to cancel enables targeted retention interventions such as personalized offers or improved recommendations.

This project tackles predicting which users will churn, defined operationally as visiting the "Cancellation Confirmation" page. While this appears to be a standard binary classification problem, it exhibits characteristics more typical of survival analysis, where the temporal dynamics of *when* users churn fundamentally affect prediction.

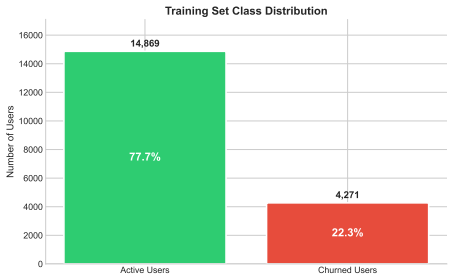

Figure 1: Training set class distribution (22.3% churn rate).

1.1 Dataset Characteristics

The dataset consists of event-level data capturing user interactions: 17.5M training events (19,140 users) and 4.4M test events (2,904 users) spanning October–November 2018. Each event contains: `userId`, `page` (interaction type), `time`, `sessionId`, `level` (subscription tier), `song`, `artist`, and demographic information.

Critical constraint: There is zero user overlap between training and test sets. This means models must learn generalizable behavioral patterns rather than memorizing individual user characteristics. This is a realistic requirement for production deployment, but one that creates significant validation challenges.

The training set exhibits moderate class imbalance with 4,271 churned users (22.31%) among 19,140 total users (Figure 1). This imbalance is manageable with standard techniques and does not require aggressive resampling.

2 Feature Engineering

The core technical challenge was transforming 17.5 million event records into meaningful user-level features. A vectorized pipeline using pandas `groupby` operations achieved approximately 100x speedup over naive iteration. The pipeline produced **43 features** across seven categories:

- **Engagement (8):** total_events, total_songs, total_sessions, avg_session_length
- **Page Counts (12):** page_nextsong, page_thumbs_up, page_thumbs_down, page_downgrade
- **Behavioral Ratios (6):** thumbs_up_ratio, error_rate, ad_ratio
- **Temporal (5):** days_active, events_per_day, activity_trend
- **Subscription (4):** is_paid, level_changes, has_downgrade
- **Content (4):** unique_songs, unique_artists, song_repeat_ratio

2.1 Key Design Decisions

Several findings shaped the approach. First, **activity levels show no notable difference**: the mean event count for churned users (917.8) is nearly identical to non-churned users (913.3). This suggests churn is not simply driven by disengagement, requiring other behavioral signals.

Second, **downgrade behavior is a key predictor**: 72.9% of churned users visited the Downgrade page versus 61.0% of non-churners, representing a strong signal of intent to reduce platform commitment.

Third, cancel page events were carefully handled, excluding events within 12 hours of churn confirmation to prevent obvious leakage while preserving legitimate behavioral signals.

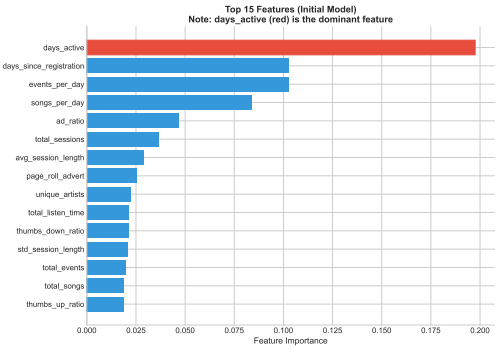

Figure 2: Initial feature importance.

2.2 Initial Results

With feature engineering complete, baseline models achieved promising cross-validation results: Logistic Regression at 81.8% and Random Forest at 87.1%. The `days_active` feature emerged as the dominant predictor by a substantial margin (Figure 2). Confidence was high. **This was wrong.**

3 The Leakage Crisis

The first test submission revealed catastrophic failure: the model achieving **88% CV accuracy collapsed to ∼50% test accuracy**, essentially random guessing. A 38-point gap demanded immediate investigation. This was not simple overfitting; complete collapse to chance indicated a fundamental flaw.

3.1 Root Cause: Temporal Leakage

Systematic investigation identified the culprit: **temporal leakage** through the strongest feature, `days_active`. This feature measured days with recorded activity, but the problem becomes clear when considering what it actually measures:

- A user who churns on day 10 has at most 10 days of data
- A user who never churns has data spanning the entire 50-day period
- Therefore, `days_active` directly encodes whether/when the user churned

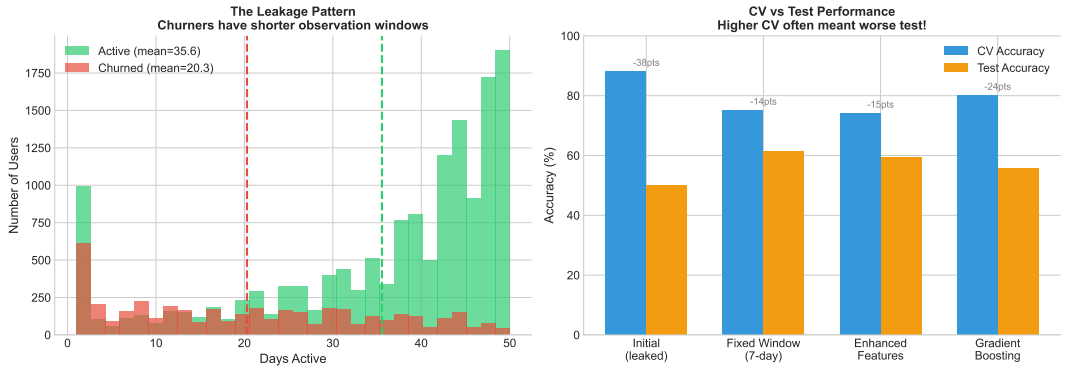

Figure 3: Left: Distribution of `days_active` by churn status. Right: CV vs test accuracy.

The model learned the tautology "low activity days = churner"; trivially true in training because churners left early, but meaningless for prediction. The contamination extended beyond `days_active`: all cumulative features were affected (`total_events`, `total_sessions`, page counts), and even rate features like `events_per_day` as ratios of contaminated totals. In the test set, where users are observed *before* knowing if they will churn, all users have similar observation windows, rendering the leaked features useless.

3.2 The Solution: Fixed-Window Approach

Temporal leakage was addressed by restricting feature computation to a **fixed window of the first 7 days** for every user, regardless of when they eventually churned. This ensures: (1) equal observation time for everyone, (2) no future leakage, as features cannot encode "how long they stayed", and (3) production-realistic prediction from early behavior only.

```
def filter_to_window(df, window_days=7):
    """Filter events to first N days per user, eliminating temporal leakage."""
    user_first_event = df.groupby('userId')['time'].min()
    df = df.copy()
    df['user_first_event'] = df['userId'].map(user_first_event)
    df['days_since_start'] = (df['time'] - df['user_first_event']).dt.total_seconds() /
    86400
    return df[df['days_since_start'] <= window_days]
```

The 7-day window was chosen based on experimentation: shorter windows (3 days) had insufficient data; longer windows (14+ days) reintroduced leakage effects.

4 Model Development and Results

With fixed-window features, multiple modeling approaches were systematically evaluated. Table 1 summarizes the key experiments.

Table 1: Model comparison after leakage fix

Window	Model	CV Accuracy	Test Accuracy
Full (leaked)	Random Forest	87%	~50%
7-day	Random Forest	74.7%	61.3%
7-day	Gradient Boosting	79.8%	55.7%
7-day	Enhanced Features	74.3%	59.2%
7-day	LDA (GDA)	73.1%	50.0%

A striking pattern emerged: **lower CV accuracy correlated with better test performance**. Models that "learned less" generalized better because they couldn't exploit leakage patterns. This inversion reflects the distribution shift between datasets.

4.1 Feature Importance After Fix

With leakage eliminated, feature importance rankings changed substantially. Top features: `ads` (0.077), `sessions_in_window` (0.073), `ads_per_song` (0.070), `thumbs_down` (0.052), `downgrades` (0.051). These tell a coherent story: users who churn are frustrated by ads, signal dissatisfaction, and consider downgrading. This aligns with domain knowledge about streaming churn.

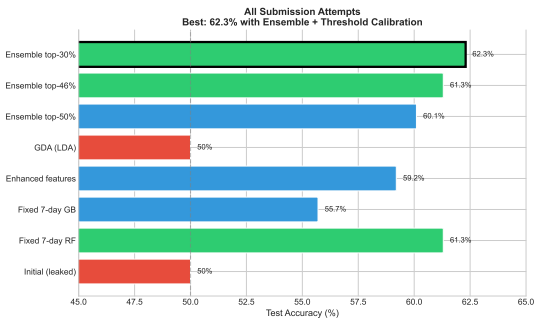

Figure 4: Submission attempts over time.

4.2 Final Model Configuration

The best submission achieved **62.3% test accuracy** using an ensemble approach with threshold calibration:

```
# Ensemble: 65% Gradient Boosting + 35% Random
    Forest
proba = 0.65 * gb.predict_proba
(X)[:,1] + 0.35 * rf.predict_proba(X)[:,1]

# Threshold calibration: predict top 30% as
    churners (not p > 0.5)
n_churn = int(len(proba) * 0.30)
predictions = np.zeros(len(proba), dtype=int)
predictions[np.argsort(proba)[-n_churn:]] = 1
```

The threshold calibration (predicting top 30% by probability rather than using 0.5) accounts for potential base rate differences between train and test sets. Figure 4 shows the progression through different approaches.

5 Discussion and Lessons Learned

Why this problem is hard:

Temporal leakage is subtle and devastating. The strongest feature was computed *from* the outcome. Feature importance correctly identified it as highly predictive, because it was essentially the target in disguise. This type of leakage is easy to introduce and hard to detect without careful temporal reasoning.

Cross-validation can mislead. Standard CV assumes train and test come from the same distribution. When violated, CV scores become unreliable. The results showed consistent *negative* correlation between CV and test accuracy.

Data quality trumps model complexity. Six architectures, four window sizes, and three feature sets were tested. All converged to ∼61%. The ceiling is in the data, not the modeling.

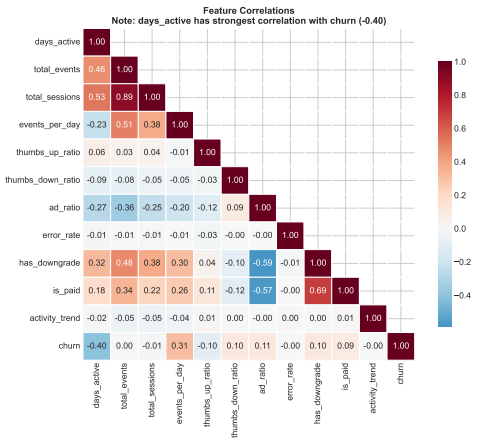

Figure 5: Feature correlations with churn.

5.1 Key Insights

Several principles emerged: (1) **Simpler models generalize better** under distribution shift; Random Forest outperformed Gradient Boosting despite lower CV. (2) **This is fundamentally a survival analysis problem**; binary classification ignores *when* users churn, and a user churning on day 5 differs from one churning on day 45. (3) **Feature engineering requires causal reasoning**, not just statistics; statistically predictive features can be useless if they encode information unavailable at prediction time.

5.2 Future Directions

Future work could explore: **Survival Analysis** (Cox Proportional Hazards) for proper time-to-event modeling with censoring; **Sequence Models** (RNN/LSTM) on raw events to learn temporal patterns directly; and **Change-Point Detection** to identify behavioral regime shifts preceding churn.

6 Conclusion

This project demonstrated that **understanding your data beats tuning your model**. The journey from 88% CV / 50% test to 75% CV / 62.3% test illustrates a fundamental principle: careful attention to data leakage matters more than algorithmic complexity. The final 62.3% represents genuine predictive signal, far more valuable than inflated metrics built on leaked information.

Table 2: Project Summary

Approach	CV	Test	Key Learning
Full features	88%	50%	Temporal leakage in cumulative features
Fixed 7-day window	74.7%	61.3%	Equal observation time eliminates leakage
Gradient Boosting	79.8%	55.7%	Complex models overfit to distribution shift
Ensemble + top-30%	—	62.3%	Threshold calibration provides final gain