Churn Prediction from Music Streaming Service Logs

Python for Data Science Project Report – December 2025

Maximilien Lucille, Oliver Smith

1 Introduction

The objective of this project is to predict which users of a music streaming service will cancel their subscription within 10 days after November 20, 2018. A user is considered to have churned when they visit the "Cancellation Confirmation" page.

Data: The training set contains 19140 users with 17+ million log entries, while the test set contains 2,904 users whose churn status we must predict. Each log entry contains user information, session data, page visits (NextSong, Thumbs Up/Down, Help, Settings, etc.), and content metadata.

Evaluation Metric: Balanced accuracy $= \frac{1}{2}(\text{Recall}_0 + \text{Recall}_1)$, which equally weights both classes.

2 Exploratory Data Analysis

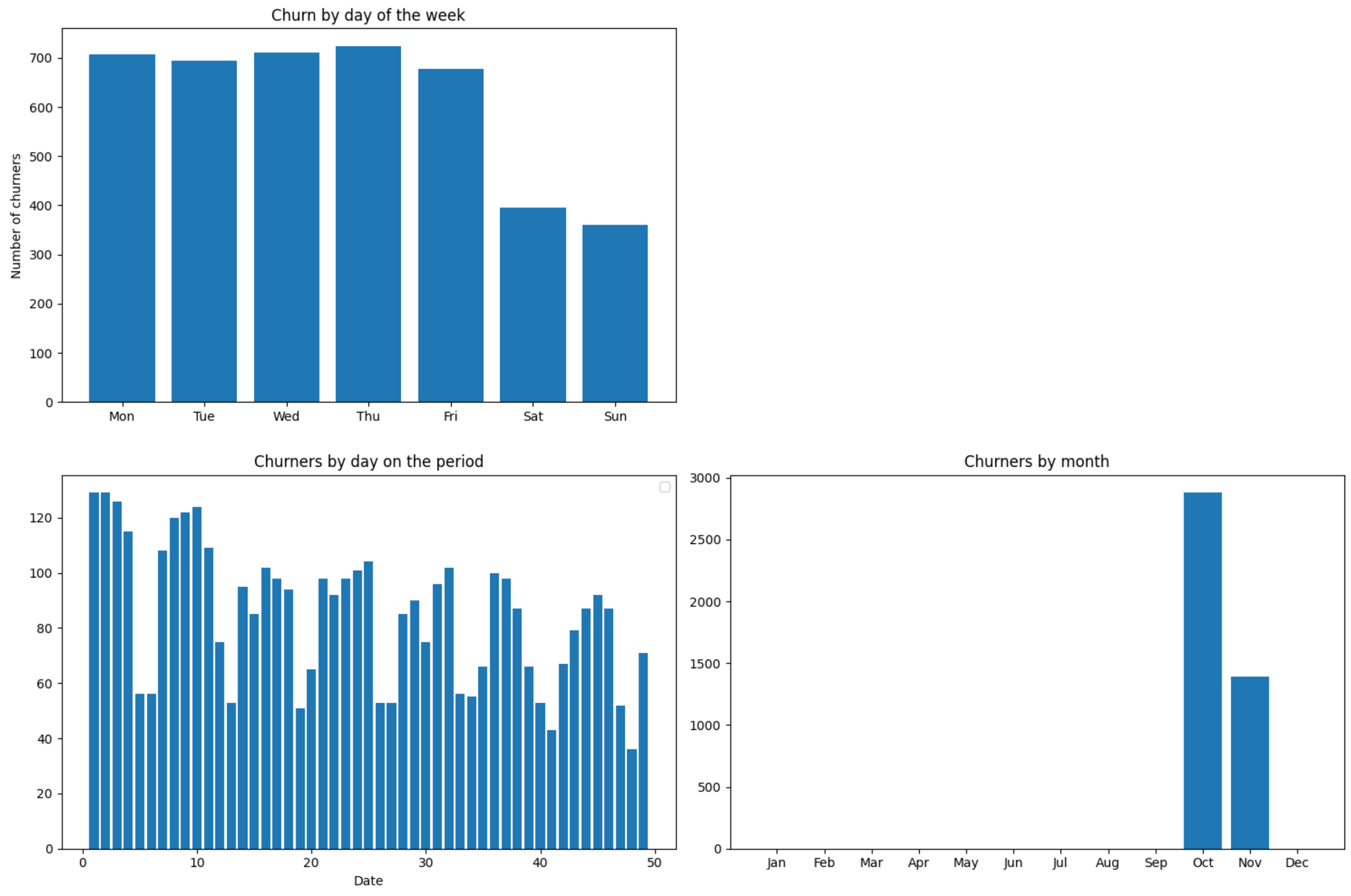


Figure 1: Temporal patterns in churn events: (top-left) churns by day of week showing weekend dips, (bottom-left) daily churn count over the observation period, (bottom-right) monthly churn totals.

Our temporal analysis revealed that churns occur less frequently on weekends (Saturday/Sunday), suggesting users tend to manage subscriptions during workdays. The daily pattern shows consistent churn throughout the observation period with no major structural breaks.

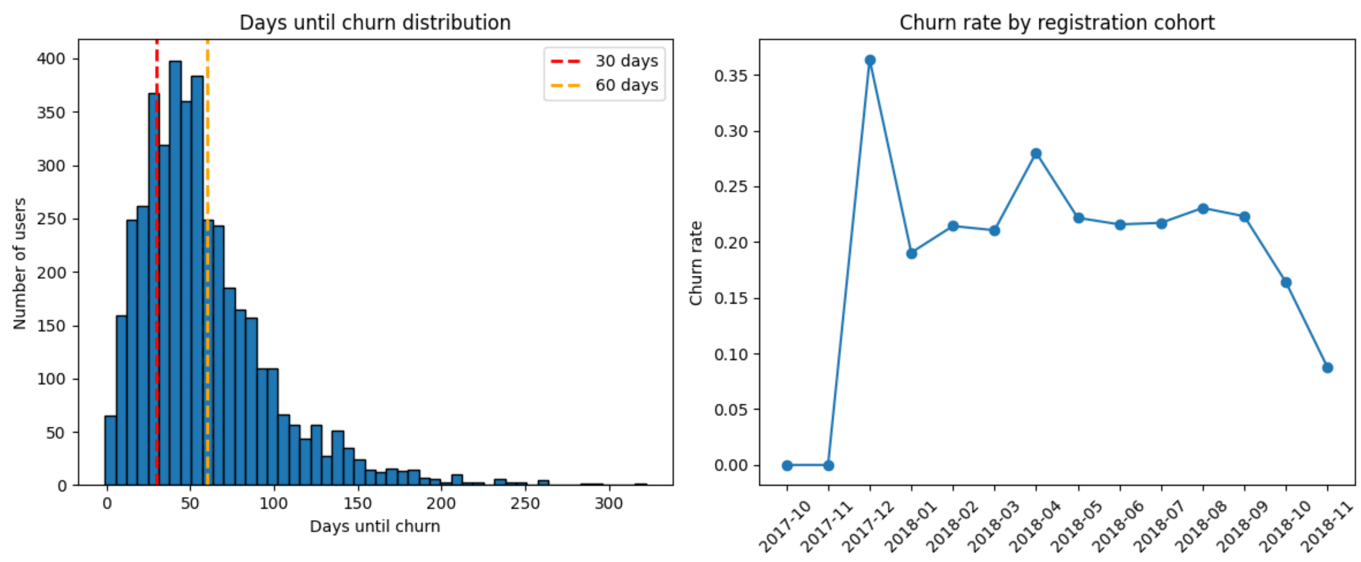


Figure 2: (Left) Distribution of days until churn showing most churns occur 30-60 days after registration. (Right) Churn rate by registration cohort showing stable rates around 20-22% after initial volatility.

The time-to-churn distribution peaks around 30-50 days, with reference lines at 30 and 60 days. The cohort analysis shows churn rates stabilize around 20-22% after the initial registration period.

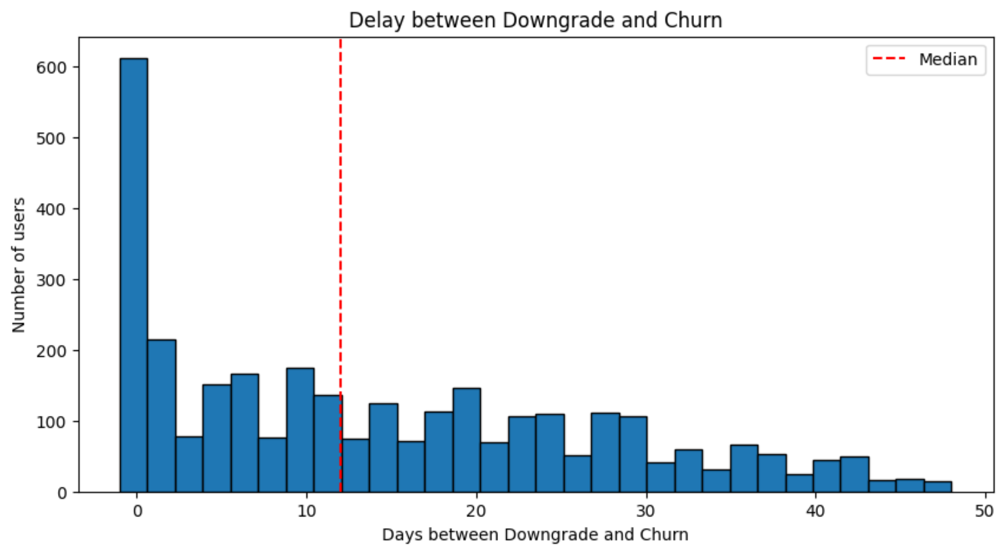


Figure 3: Days between downgrade event and churn. The large spike at day 0 indicates many users churn immediately after downgrading, making downgrade a strong predictive signal.

A key insight from EDA: users who downgrade often churn immediately (large spike at day 0), making downgrade actions a strong predictive feature.

3 Feature Engineering

We engineered 24 features at the session level using a 60-day lookback window:

Engagement metrics: `count_nextsong`, `count_thumbs_up`, `count_thumbs_down`, `count_add_playlist`, `count_add_fr`

Support-seeking: `count_help`, `count_settings`, `count_error`

Subscription actions: `count_downgrade`, `count_upgrade`, `count_submit_downgrade`, `count_submit_upgrade`

Activity metrics: `total_actions`, `num_sessions`, `count_roll_advert`

Computed ratios: `skip_rate` (skips/songs), `like_ratio` (likes/songs)

Temporal variations: `activity_variation_pct`, `music_consumption_variation_pct` comparing current vs previous week

Binary flags: `has_downgrade_action`, `has_upgrade_action`, `has_negative_engagement`

4 Data Preparation

Several filtering steps were applied to create clean training data:

1. **Temporal cutoff:** Sessions after November 10 excluded to ensure we train only on behavior observed before the 10-day prediction window.
2. **Early churners removal:** Users who churned within the first week of the dataset were excluded, as we considered there was insufficient behavioral data to learn meaningful patterns
3. **Consistent user filtering:** For churning users, kept only sessions with `will_churn=1`; for non-churners, kept all sessions. This removed 17,939 sessions (13%)

Final training set: 119,755 sessions with 12,901 positive (10.8%) and 106,854 negative labels.

5 Modeling

5.1 Models Tested

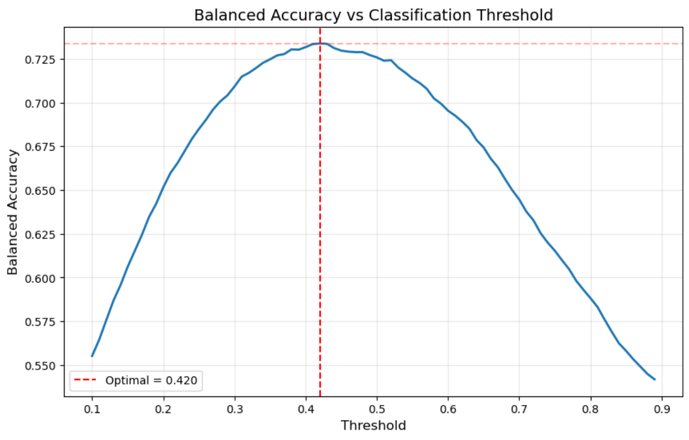
Random Forest with SMOTE: Used SMOTE oversampling to address class imbalance, with `max_depth=20`.

XGBoost: Used `scale_pos_weight` parameter (ratio ≈ 8.3) instead of resampling to handle imbalance. Parameters: `n_estimators=200`, `max_depth=6`, `learning_rate=0.01`.

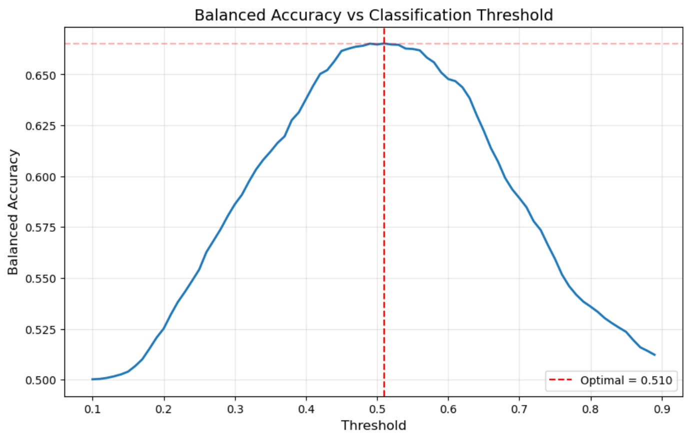
Logistic Regression: Baseline model with `class_weight='balanced'`.

5.2 Threshold Optimization

Since balanced accuracy depends on the classification threshold, we optimized it on validation data:



(a) Random Forest: optimal threshold = 0.42



(b) XGBoost: optimal threshold = 0.51

Figure 4: Balanced accuracy vs classification threshold for both models. The curves show smooth optima around 0.4-0.5.

The threshold curves show that RandomForets achieves higher balanced accuracy (0.73) compared to XGBoost (0.67) on validation data, with optimal thresholds at 0.51 and 0.42 respectively.

5.3 Model Calibration

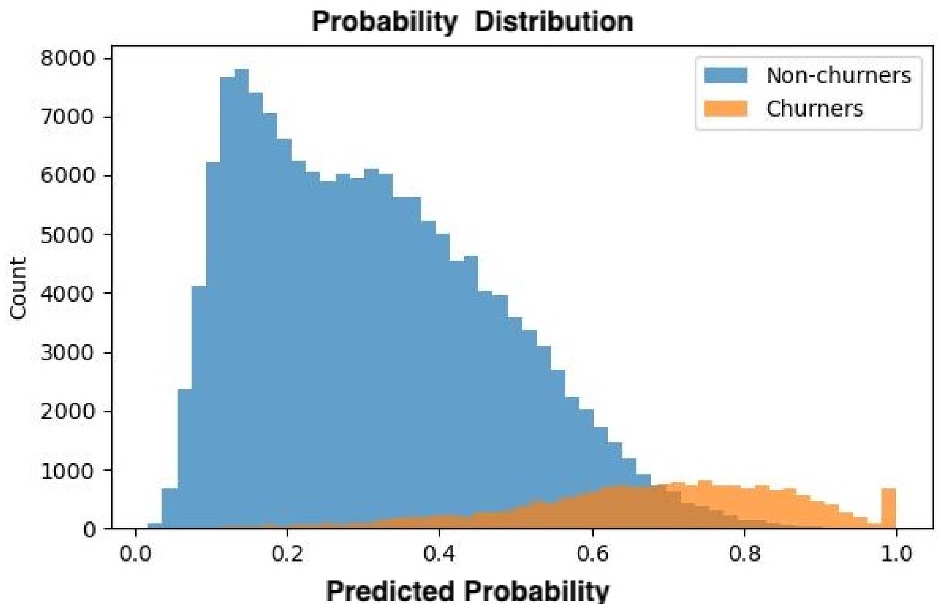


Figure 5: Distribution of predicted probabilities for churners vs non-churners

The probability distribution shows that the model somewhat separates the two classes, with non-churners concentrated at lower probabilities and churners spread across higher values.

6 Results

Model	Validation Bal. Acc.	Kaggle Score
Random Forest + SMOTE	0.73	0.61
XGBoost	0.67	0.60

Table 1: Model performance comparison

Our best submission achieved **0.61 balanced accuracy** on Kaggle using RandomForest with threshold calibration.

Key insights:

- Downgrade actions are the strongest churn predictors (immediate churn after downgrade)
- Activity variation features capture declining engagement patterns
- Threshold optimization is critical for balanced accuracy metric
- Distribution shift between train and test data limits generalization

6.1 Alternative Approaches

Beyond the final model, we tested several alternative approaches that were ultimately discarded due to the lack of performance improvement. These included adding music-related features (favorite artists, preferred songs, and favorite genres), segmenting churners into different usage profiles (very intensive versus very occasional users), and enriching the dataset with contextual variables such as paid versus free status and device-related features (desktop or mobile, search engine used). We also attempted a deep learning model based on TAPFN, which is designed for tabular data, but it could not be run on our local machines. Overall, none of these approaches improved the results and most led to lower predictive performance, likely due to overfitting on the training data.

7 Conclusion

We developed a churn prediction pipeline achieving 0.61 balanced accuracy. The most effective approach combined session-level feature engineering with XGBoost and careful threshold calibration. The analysis revealed that subscription downgrade events and activity decline patterns are the strongest predictors of churn. Future improvements could include sequential modeling to capture user behavior trajectories over time.