Predicting User Churn with PySpark: A Comprehensive Guide

In this blog post, I'll guide you through a complete machine learning project aimed at predicting user churn for the fictional music streaming service, Sparkify. We'll cover every step from project overview to hyperparameter tuning, using PySpark to handle large datasets efficiently.

Project Overview

In the competitive world of music streaming services, retaining users is crucial. Predicting user churn can help businesses take proactive measures to keep their customers engaged. This project aims to build a predictive model to identify users likely to churn based on their activity logs from the Sparkify app.

Problem Statement

We have a JSON log of user actions over two months. Our objective is to predict user churn based on their behavior patterns. This involves cleaning the data, performing exploratory data analysis (EDA), engineering features, and building a machine learning model. Although we work with a small subset for this demonstration, the process is scalable to handle the full dataset.

Metrics

Given that the churned user class is unbalanced, accuracy alone is not sufficient. Instead, we use precision-recall metrics to evaluate our models. Precision measures how many of the predicted churned users actually churned, and recall measures how many actual churned users were correctly identified. We aim for a balance between precision and recall, and thus use the F1-score as our primary metric:

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

Data Understanding & Preprocessing

Dataset Description

We are working with a 128MB subset of the dataset, which represents 1% of the full 12GB dataset. This subset contains 286,500 JSON records. Below is the schema of the dataset and a brief description of each column.

Dataset Schema

```
root
 |-- artist: string (nullable = true)
 |-- auth: string (nullable = true)
 |-- firstName: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- itemInSession: long (nullable = true)
 |-- lastName: string (nullable = true)
 |-- length: double (nullable = true)
 |-- level: string (nullable = true)
 |-- location: string (nullable = true)
 |-- method: string (nullable = true)
 |-- page: string (nullable = true)
 |-- registration: long (nullable = true)
 |-- sessionId: long (nullable = true)
 |-- song: string (nullable = true)
 |-- status: long (nullable = true)
 |-- ts: long (nullable = true)
 |-- userAgent: string (nullable = true)
 |-- userId: string (nullable = true)
```

Columns Description

artist: The artist of the song being played (only provided when a song is played).
auth: The authentication status of the user.
firstName: The user's first name.
gender: The user's gender.
itemInSession: The index of the event within the session.
lastName: The user's last name.
length: The duration of the song (in seconds).
level: The user's subscription level (free or paid).
location: The user's location (city, state).
method: The HTTP method used for the request.
page: The page the user interacted with.
registration: The timestamp when the user registered.
sessionId: The ID of the user session.

song: The name of the song being played.
status: The status code of the HTTP response.
ts: The timestamp of the event.
userAgent: The user agent string of the device used to connect.
userId: The unique identifier of the user.

This dataset provides rich information about user interactions with the Sparkify service, which will be crucial for predicting user churn.

Missing Values

During the data preprocessing step, we identified the following columns with missing values:
artist: 58,392 missing values
firstName: 8,346 missing values
gender: 8,346 missing values
lastName: 8,346 missing values
length: 58,392 missing values
location: 8,346 missing values
registration: 8,346 missing values
song: 58,392 missing values
userAgent: 8,346 missing values
userId: 8,346 missing values

Handling these missing values appropriately is crucial for ensuring the quality of the dataset and the accuracy of the predictive model.
Due to the fact that, the churn prediction is user-based behavior. As a result, rows without userID or seesionID is kind of useless to the prediction, we decide to remove these rows from the dataset. We also drop any duplicated rows.

Exploratory Data Analysis

Defining Churn

To predict user churn, we need to create a label that identifies churned users. In our analysis, we define churn based on the **Cancellation Confirmation** events, which occur when users cancel their service. This event indicates a clear instance of churn for both free and paid users. Additionally, we also consider **Downgrade** events as an indicator of potential churn risk, where users switch from a premium to a free subscription.

Creating Downgrade Column: We add a new column **Downgrade** which is set to 1 if the user submits a downgrade request (**Submit Downgrade** event), otherwise 0. This helps identify users who have downgraded their subscription.

Creating Churn Column: We add another column **Churn** which is set to 1 if the user confirms cancellation of their service (**Cancellation Confirmation** event), otherwise 0. This column serves as our primary label for churn.
Defining Window Bounds for Downgrade and Churn: We use window functions to sum the **Downgrade** and **Churn** events for each user over their entire history. This involves creating windows partitioned by **userId** and calculating the total number of downgrade and churn events per user. The resulting columns, **total_downgrade** and **total_churn**, provide a comprehensive view of the user's downgrade and churn activity.
By defining churn in this manner, we can accurately label churned users and include potential risk factors like downgrades in our predictive model. This approach ensures that our model captures both explicit cancellations and behavioral indicators of churn.

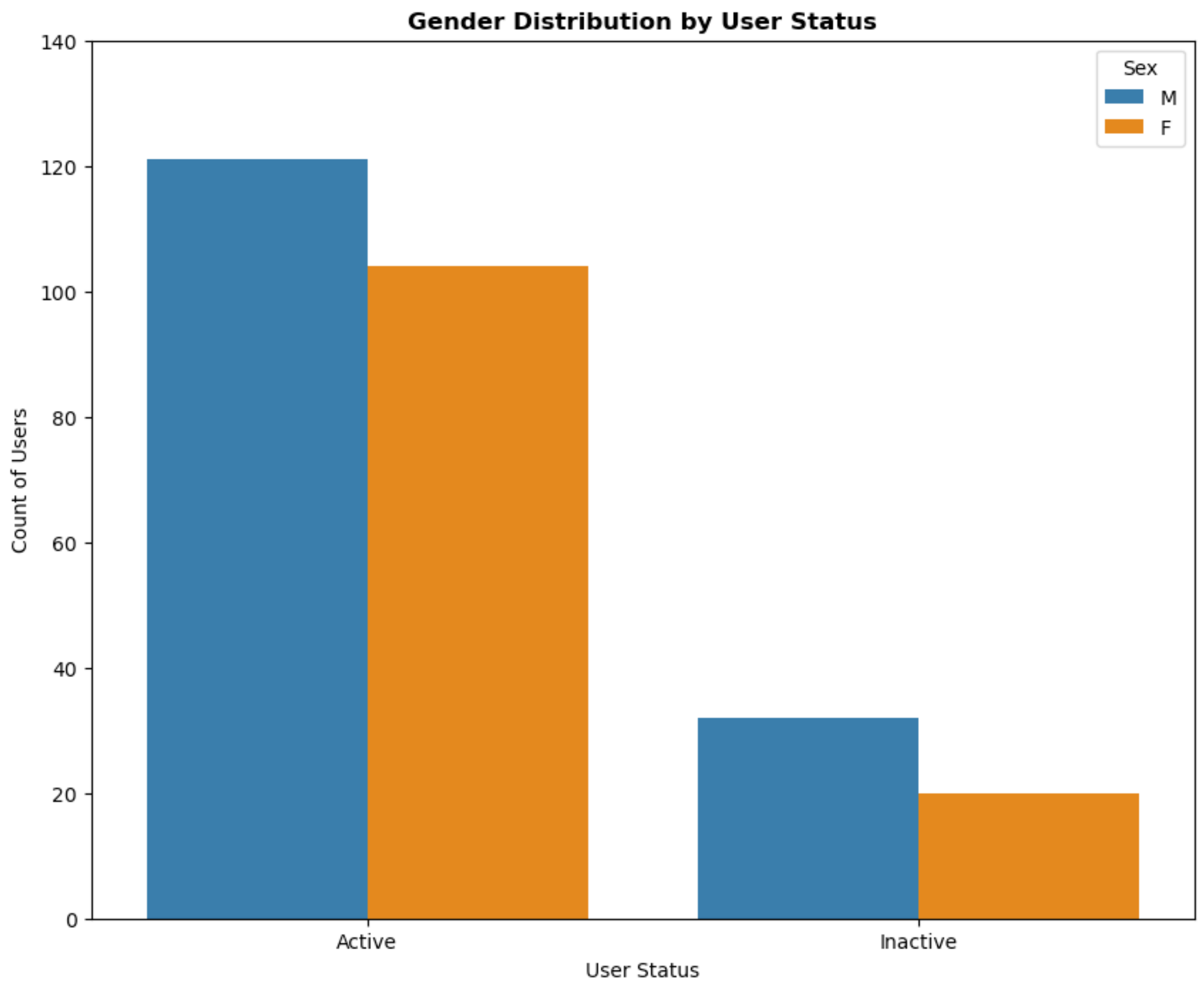
Some basic steps of understanding downgrade/churn

The number of churn vs non-churn in this subset data

Total Churn	Count
Not Churn	173
Churn	52

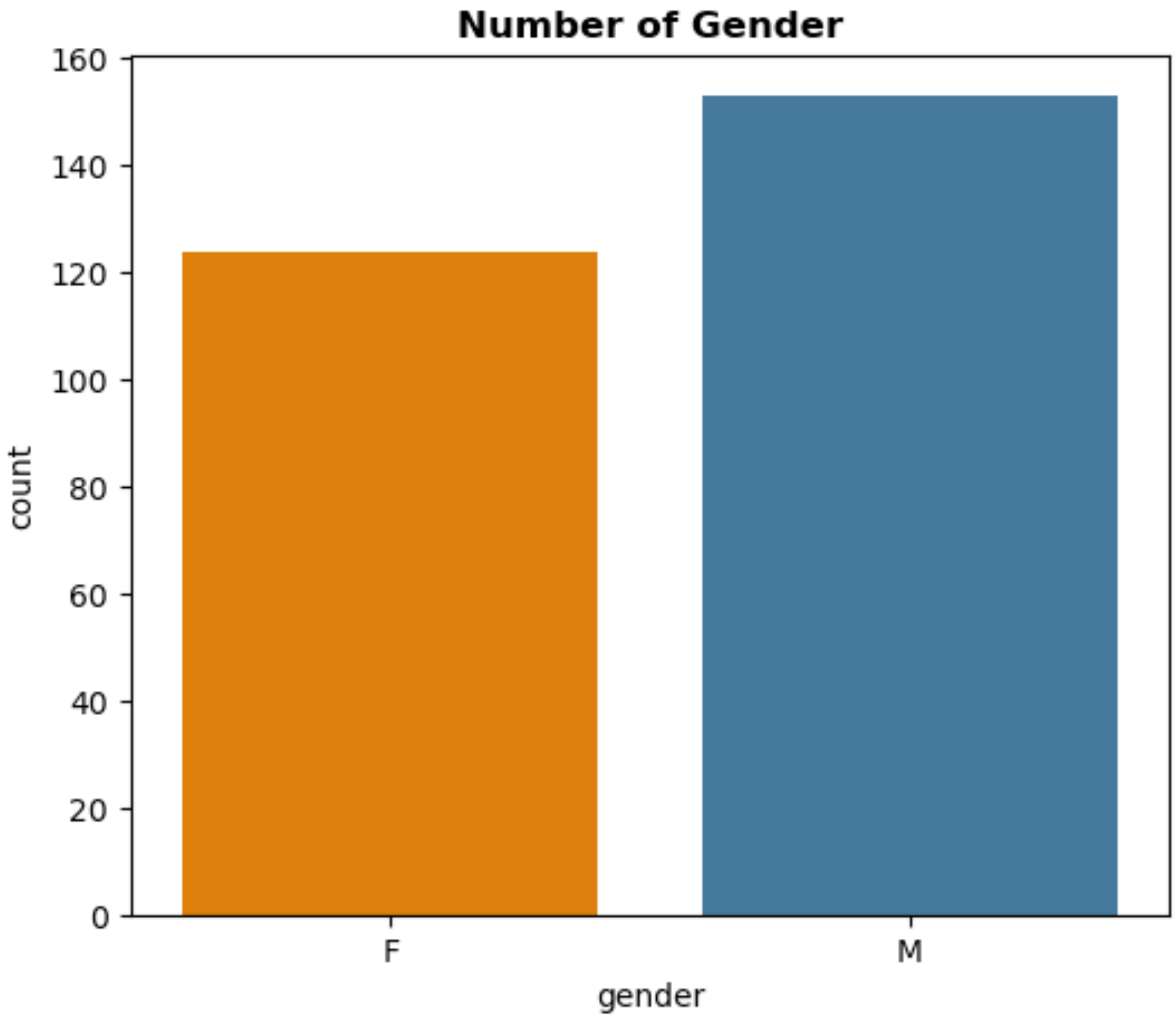
Churn-rate is about 23.11% in this subset data

Gender Distribution by User Status



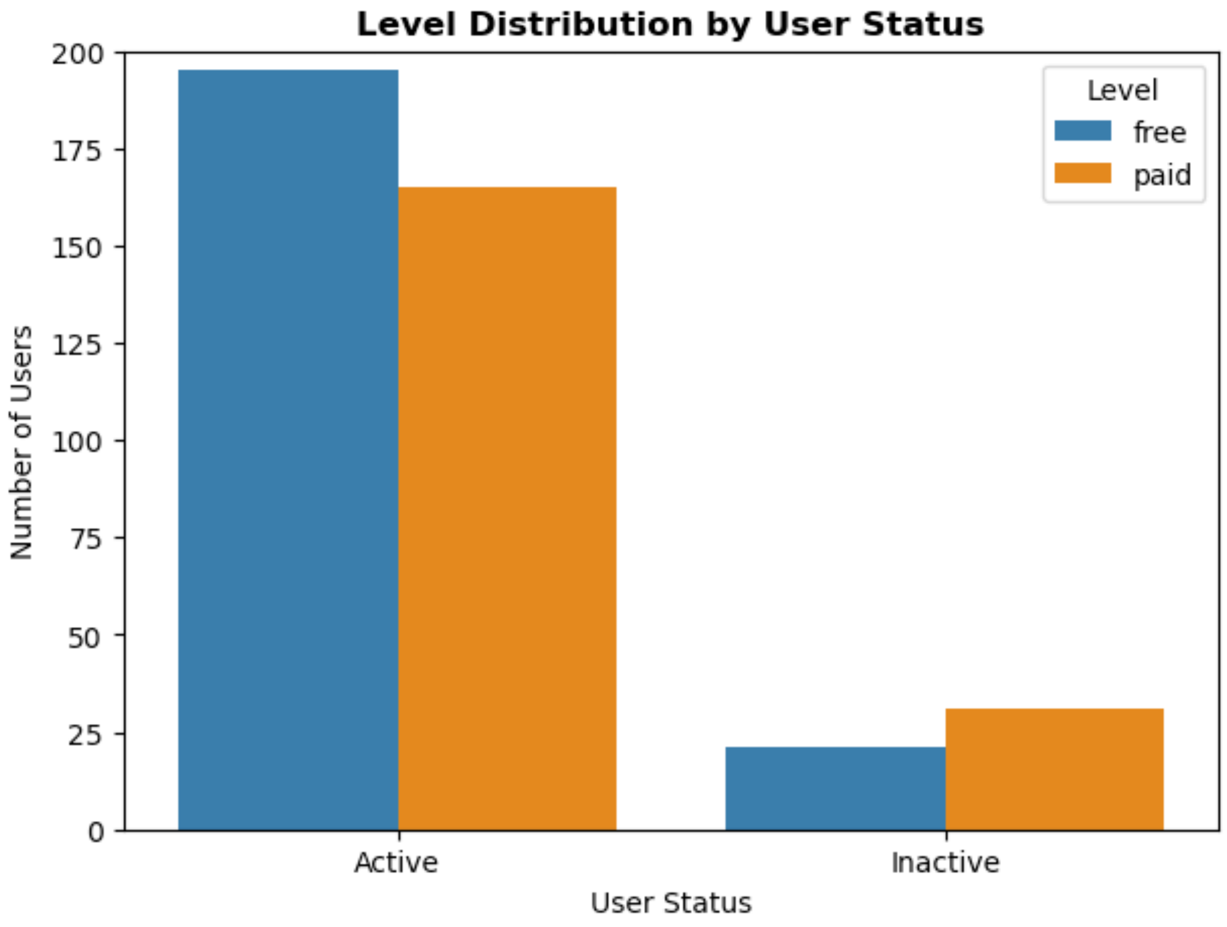
Insight:
The majority of active users are male, with a smaller but significant number of female users. Among inactive users, the trend is similar, but the counts are much lower for both genders. This indicates that both genders are actively using the service, with males being slightly more dominant in both active and inactive categories.

Number of Gender



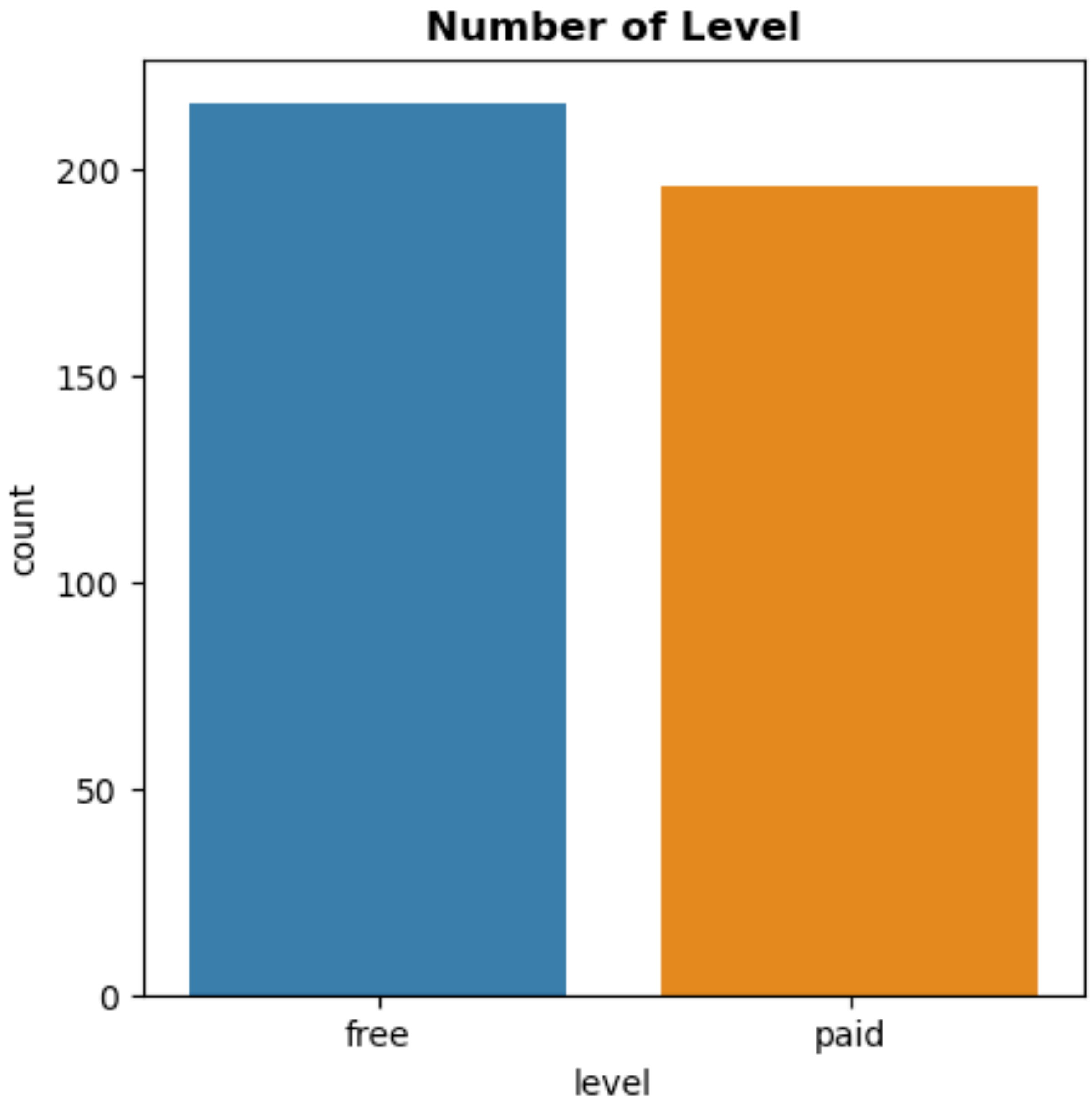
Insight:
There are more male users than female users in the dataset. This could imply that the music streaming service is more popular among males, or it could reflect the overall user base distribution.

Level Distribution by User Status



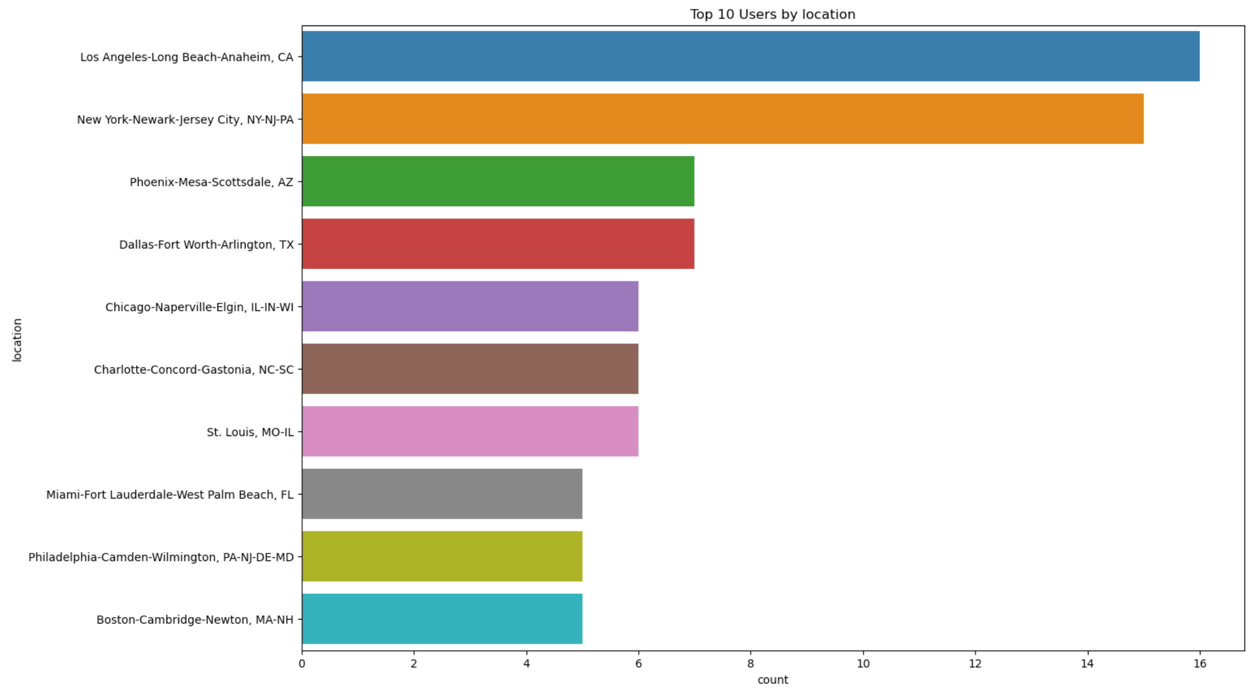
Insight:
Among active users, there are more users with a free subscription than those with a paid subscription. This trend is similar for inactive users, although the numbers are lower.
The data suggests that most users start with a free subscription, and a smaller fraction of these users upgrade to a paid subscription.

Number of Level



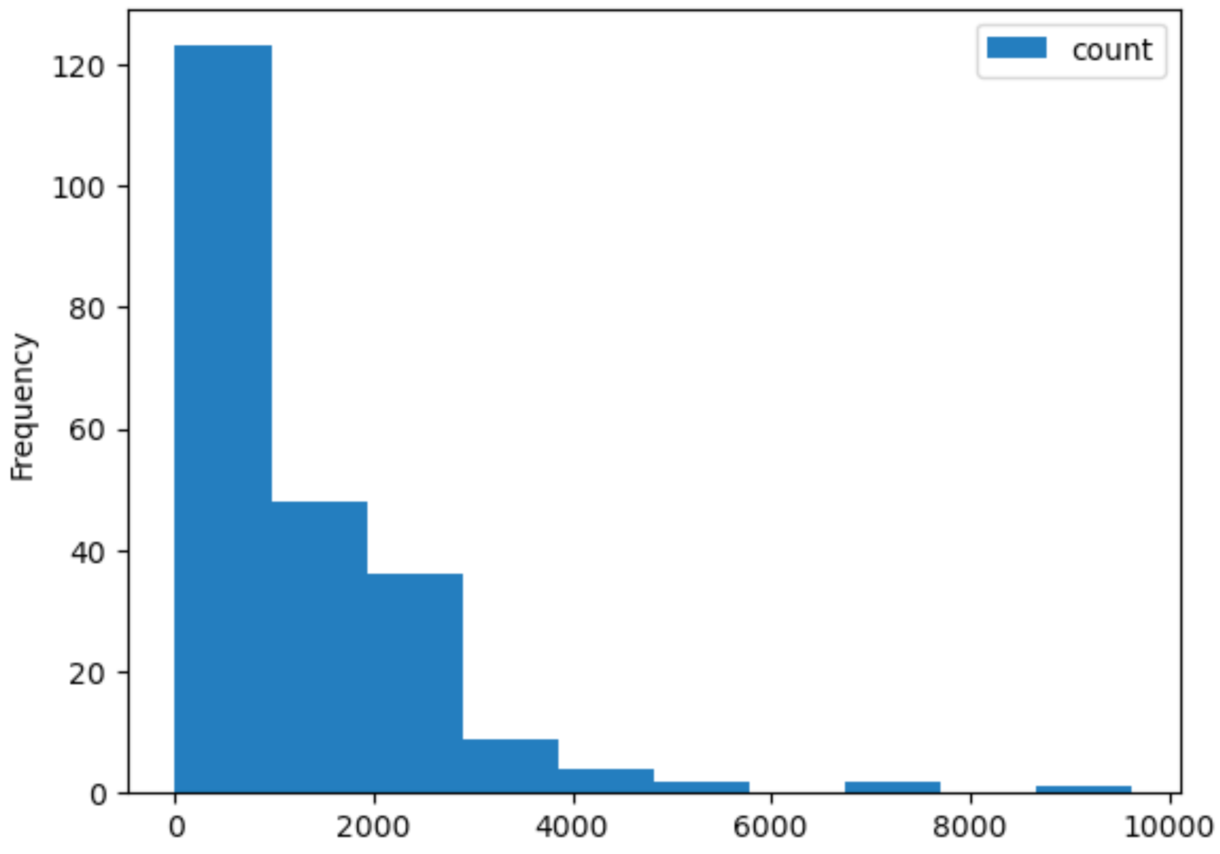
Insight:
There are more users with free subscriptions than paid subscriptions. This supports the observation that users are likely to use the free tier before deciding to pay for the premium service.

Top 10 Users by Location



Insight:
The top locations with the highest user counts are Los Angeles-Long Beach-Anaheim, CA, and New York-Newark-Jersey City, NY-NJ-PA.
Other significant locations include Phoenix-Mesa-Scottsdale, AZ, and Dallas-Fort Worth-Arlington, TX.
This geographic distribution can help target marketing efforts and understand regional popularity.

Session Length Distribution



Insight:
The distribution shows that most users have shorter sessions, with a long tail indicating fewer users having longer sessions.
This suggests that while users often engage in brief sessions, there are instances of prolonged usage, possibly indicating higher engagement levels for certain users.

Summary of Insights

Gender and User Status: There is a higher number of active male users compared to female users, although both genders show significant activity.
Subscription Levels: The majority of users have free subscriptions, with fewer users opting for the paid tier. This trend is consistent across both active and inactive users.
Geographic Distribution: The service has a broad geographic reach, with the highest user counts in major metropolitan areas.
Session Length: Most users engage in short sessions, but there are notable instances of long sessions, indicating varying levels of user engagement.

These insights help us understand the user demographics, subscription preferences, geographic distribution, and engagement patterns, which are crucial for building a predictive model for user churn.

Feature Engineering

Feature engineering involves creating new features from the existing data that can improve the predictive power of the model. Features include:

- Average Song Length (avg_song_length): The average length of all songs played by the user. This feature helps understand how long users typically engage with the content.

- Total Unique Artists (total_artist): The total number of unique artists the user has listened to. This indicates the diversity of the user's listening habits.

- Total Unique Songs (total_song): The total number of unique songs the user has played. This provides a measure of the user's engagement with different tracks.

- Total Sessions (total_session): The total number of actions performed by the user (indicated by the page column). This feature captures the user's overall activity on the platform.

- Total Thumbs Up (total_thumb_up): The total number of times the user has given a thumbs up. This reflects the user's positive feedback and satisfaction.

- Total Thumbs Down (total_thumb_down): The total number of times the user has given a thumbs down. This indicates the user's negative feedback and potential dissatisfaction.

- Total Add Friend (total_add_friend): The total number of times the user has added a friend. This feature can provide insight into the user's social behavior on the platform.

- Total Add to Playlist (total_add_playlist): The total number of times the user has added a song to a playlist. This reflects the user's engagement in curating and managing their personal playlists.

- Gender (gender): The gender of the user, mapped to numerical values (0 for Male, 1 for Female). This demographic feature can help understand the influence of gender on user behavior and churn.

Model Building

We use the following machine learning algorithms to build the churn prediction model:

- Logistic Regression
- Random Forest Classifier
- Gradient Boosted Trees (GBT) Classifier
- Hyper Tunned Random Forest Classifier

Hyperparameter Tuning

Hyperparameter tuning is performed using grid search and cross-validation to find the optimal parameters for the best-performing model of Random Forest. The primary focus is on optimizing the F1 score to balance precision and recall.

Define a parameter grid for hyperparameter tuning

```
paramGrid = ParamGridBuilder() \
  .addGrid(rf.maxDepth,[5, 10]) \
  .addGrid(rf.numTrees, [20, 50]) \
  .addGrid(rf.minInstancesPerNode, [1, 10]) \
  .addGrid(rf.subsamplingRate, [0.7, 1.0]) \
  .build()
```

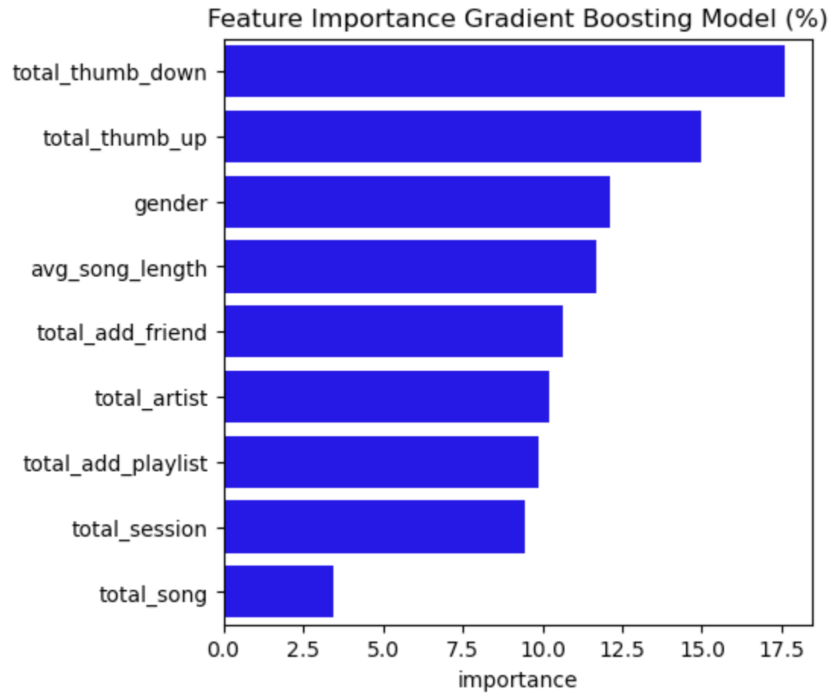
Conclusion

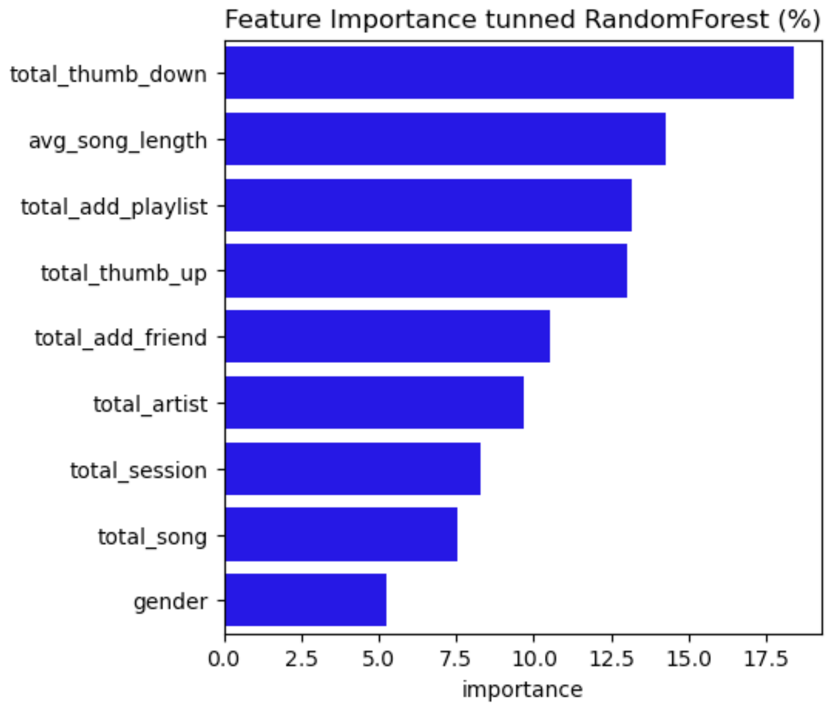
Here is a final result:

Model	F1 Score	Accuracy
Logistic Regression	58.54%	67.74%
Random Forest	62.69%	70.97%
Gradient Boosting	67.05%	69.35%
Random Forest Tuned	61.64%	69.35%

We should go with Gradient Boosting Model where it achieves the highest F1-Score and a roughly balance Accuracy compared to other models. It could be a based model to further improvement with full dataset.

The Feature Importantance of Gradient Boosting and tunned Ramdom Forest Models





Recommendations

Based on the feature importance insights, we can make several recommendations to reduce user churn:

Improve Content Quality: Since the number of thumbs down is the most significant predictor of churn, focus on understanding the reasons behind negative feedback. Improve content quality and personalize recommendations to better match user preferences.

Enhance Positive Engagement: Encourage users to give thumbs up and add songs to playlists by making these features more accessible and engaging. Highlight popular and well-liked content to keep users satisfied.

Targeted Marketing and Offers: Utilize gender information to tailor marketing campaigns and offers. Personalized promotions and content recommendations based on gender-specific preferences can enhance user satisfaction.

Promote Social Interactions: Increase the social features of the platform, such as friend recommendations and social playlists. Encourage users to add friends and interact with each other to create a more engaging and community-driven experience.

Analyze Listening Habits: Pay attention to users' listening habits, such as the average length of songs played. Use this information to curate playlists and suggest content that aligns with user preferences.

Diverse Content Library: Ensure a diverse library of artists and songs to cater to varied musical tastes. Highlighting new and diverse artists can keep users engaged and prevent them from becoming bored with the platform.

User Activity Monitoring: Monitor user activity levels and identify patterns of reduced engagement. Proactively reach out to users showing signs of reduced activity with personalized offers and re-engagement campaigns.

Feedback Loops: Implement feedback mechanisms where users can provide insights into why they may be dissatisfied or considering leaving. Use this feedback to make continuous improvements to the service.

By implementing these recommendations, Sparkify can address the key factors driving user churn and work towards improving user retention and overall satisfaction.

Suggestions

Our sample dataset represents only 1% of the original dataset, which is a very small portion. Using such a small dataset presents several challenges when applying the model to a larger dataset, including:

Challenges with Small Datasets:

High Variance and Overfitting:
Models trained on small datasets are more likely to have high variance, making them sensitive to even minor changes in the data split. This increases the risk of overfitting, where the model performs well on the training data but poorly on unseen data.

Inconsistent Results:
The results obtained from the sample dataset may not be consistent with those from other data splits. Depending on the specific training/test dataset used, the performance of the machine learning models can differ significantly.
Handling Challenges in Larger Datasets:

Scaling Exploratory Data Analysis (EDA):

Approach: Perform EDA more details in big dataset by PySpark.
Benefit: This ensures comprehensive analysis and uncovering of patterns that might be missed in smaller samples.

Feature Engineering Adjustments:

Approach: Implement scalable feature engineering techniques, such as distributed computing for calculating rolling averages or other time-based features.
Benefit: Ensures that feature engineering can handle the volume and velocity of the full dataset without performance degradation.

Model Training and Tuning:

Approach: Use distributed machine learning libraries (e.g., MLlib in PySpark, Dask-ML, or TensorFlow on distributed clusters) to train and tune models on the full dataset.
Benefit: This enables the model to learn from the entire dataset, reducing the risk of overfitting and improving generalizability.

Balancing Computational Resources:

Approach: Optimize resource allocation by using cloud-based solutions (e.g., AWS, GCP, Azure) that offer scalable computing power. Implement resource-efficient techniques like early stopping, checkpointing, and model pruning.
Benefit: Efficient use of resources ensures cost-effectiveness and timely completion of tasks without compromising model performance.

While the small dataset allows for initial model experimentation, the results are not directly translatable to the full dataset. To handle the transition to a larger dataset, we must scale our EDA, adjust feature engineering techniques, and leverage distributed computing for model training and tuning. This approach will help balance computational resources and ensure that our findings are robust and reliable when applied to the full dataset.