The Hong Kong University of Science and Technology

MSBD5003 Big Data Computing

PySpark-based Streaming User Churn Prediction

Student Name	Student ID
TANG Shiyan	20937338
XU Jinwen	20890968
TANG Yinong	20904733
YE Yi Lin	20922644

May, 2023

1 Introduction

1.1 Background

Effective customer management is crucial for the success of the music streaming business. Identifying users who are likely to churn, i.e. those who may downgrade or cancel their paid and free subscriptions, is an important part of this. Offering discounts and other promotions to these users in advance can encourage them to stay and save millions of dollars in lost revenue. Studies suggest that acquiring a new customer can cost five to seven times more than retaining an existing one [1]. Therefore, retaining existing customers is far more cost-effective than acquiring new ones, making it an essential element of any successful business.

Sparkify is an imaginary music streaming enterprise created by Udacity. It is designed to simulate real streaming services like Spotify. Users of Sparkify can choose two plans: a free subscription plan with ads or a paid subscription plan without ads. In addition to listening to music, users can engage with the service by rating songs with a thumbs up or down, adding songs to playlists, or adding friends. Users are also free to modify their subscription plan by upgrading from free to paid, downgrading from paid to free, or cancelling their subscription to use the service.

Each user interaction will be recorded as an activity log, including information such as the event time, session id, user id, subscription plan, user action, and other relevant details.

1.2 Task Description

The objective of this project is to help Sparkify, an fictional music streaming enterprise to accurately identify potential churned users, by building and training a machine learning model based on their past activities and interaction patterns with the service. For large-scale enterprises with a significant user base, traditional methods of data analysis are difficult or time-consuming. Therefore, we aim to leverage big data technology to store and acquire massive data while completing data processing, feature engineering, modelling, and evaluation work on a vast scale.

The project will be conducted from the following three aspects:
1. Data Storage and Connector
2. Feature Engineering, Model Training and Evaluation
3. Data Visualization

2 System Architecture

2.1 Techonologies

1. Data Storage and Connector: MongoDB
MongoDB is used to store the data read by Spark. It is well-suited for handling unstructured data and semi-structured data due to its fast querying and flexible schema design capabilities on large-scale datasets.

2. Data Preprocessing: Spark SQL and DataFrame

Spark SQL and Spark DataFrame are used for big data processing. They offer several advantages, such as supporting distributed computing, handling large datasets efficiently, and providing a rich library of pre-built functions for data manipulation and analysis. In this project, Spark SQL and Spark DataFrame are used to efficiently process and analyze the large volume of user data. By using them, we can perform complex operations such as joining and grouping on the distributed dataset, helping us to find insights into customer behaviour, popular songs, and other metrics.

3. Data Modelling: Spark MLlib and GraphFrame
Spark MLlib is a powerful machine learning library that offers distributed algorithms and data structures to perform data analysis tasks at scale. It provides a wide range of algorithms that can be used for classification, regression, clustering, and collaborative filtering.

GraphFrame, on the other hand, is a distributed graph processing library that allows users to perform graph analytics tasks at scale. It provides a set of graph operators and algorithms that can be used to analyze large-scale graph data. For example, in our project, GraphFrame assists in analyzing the network of users and artists to identify patterns and relationships between them.

Spark MLlib and GraphFrame allow us to process large amounts of data at scale and perform complex analytics tasks that would be difficult or impossible with traditional data analysis tools. Additionally, the integration of MLlib and GraphFrame with Spark allows for seamless integration with other Spark components, such as Spark SQL and Spark Streaming, enabling a unified data processing pipeline.

4. Data Visualization: Matplotlib and Seaborn
Matplotlib and Seaborn are visualization libraries used to interpret and understand the data. Matplotlib is a library that is used to create different types of graphs, charts, and plots. Seaborn is another library that builds on top of Matplotlib. It offers many different types of graphs, including heat maps, distribution plots, and regression plots. Users can easily change colour palettes, font styles and plot aesthetics. Therefore, they are well-suited for a wide range of data visualization tasks.

2.2 System Architecture

We deployed the big data storage, analysis, modelling and visualization techniques mentioned above to complete the task and below is the system architecture with the flow.

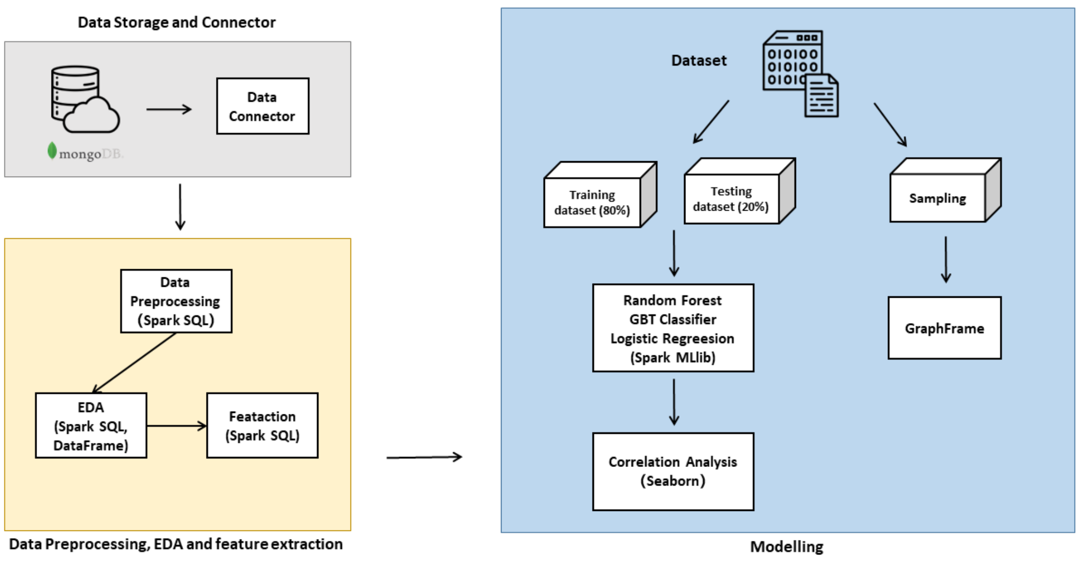
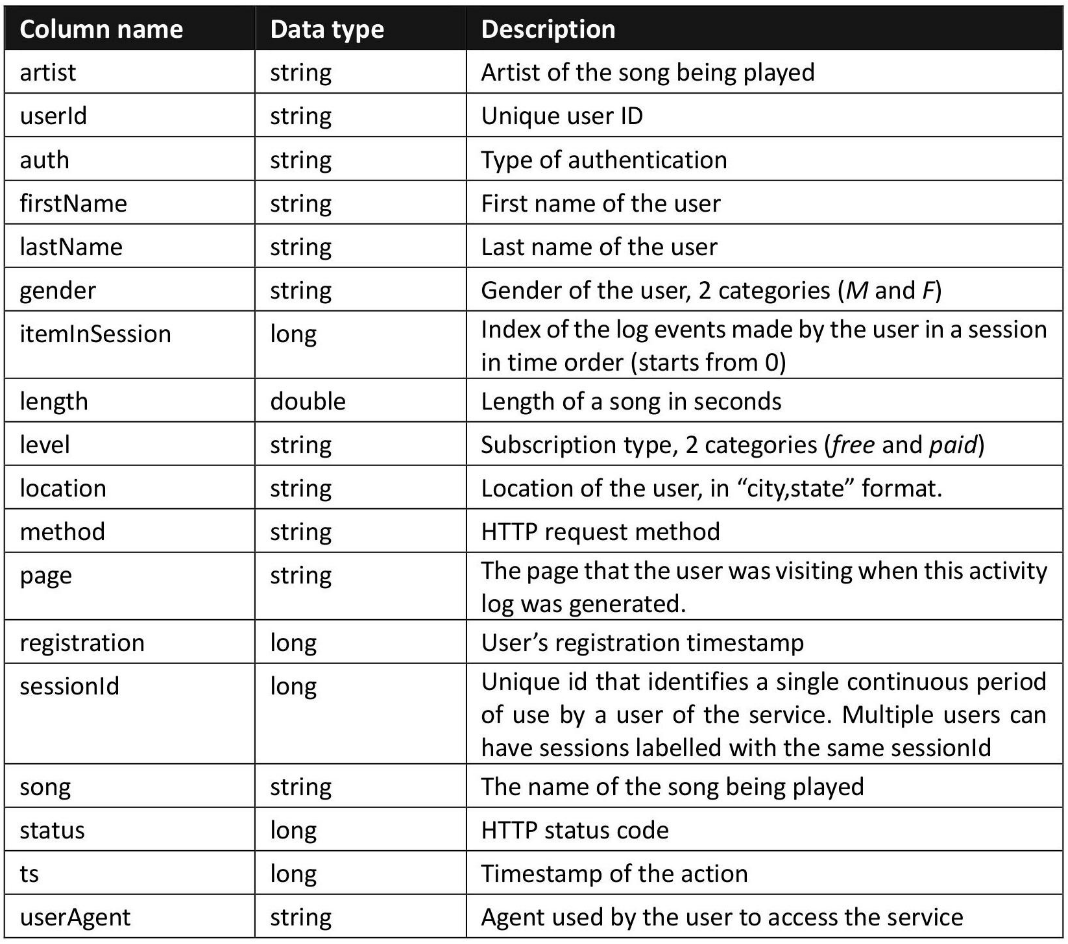


Figure 1: System Architecture

3 Dataset

The dataset used in this project is an activity log of Sparkify users between October 1 and December 3, 2018, in JSON format and is retrieved from the Udacity Amazon S3 bucket. The activity log includes the personal information of the users, songs being played, and actions made by the user, such as logging in/out, playing songs, liking songs, and upgrading/downgrading subscription plans.

Due to the overwhelming size of the original dataset, which contains around 26 million activity logs for 22,278 users and is 12G in size, we decided to sample the activity logs of some subsets of users during the study period to perform the analysis. Two versions of the sampled datasets are available: A larger version (referred to as the "full dataset" in the following part of the report for simplicity) that contains 1,625,786 activity logs for 1,200 users and a smaller version (referred to as the "mini dataset" in the following part of the report for simplicity) that contains 286,500 activity logs for 226 users. Below is an overview of the datasets:



Column name	Data type	Description
artist	string	Artist of the song being played
userId	string	Unique user ID
auth	string	Type of authentication
firstName	string	First name of the user
lastName	string	Last name of the user
gender	string	Gender of the user, 2 categories (*M* and *F*)
itemInSession	long	Index of the log events made by the user in a session in time order (starts from 0)
length	double	Length of a song in seconds
level	string	Subscription type, 2 categories (*free* and *paid*)
location	string	Location of the user, in "city,state" format.
method	string	HTTP request method
page	string	The page that the user was visiting when this activity log was generated.
registration	long	User's registration timestamp
sessionId	long	Unique id that identifies a single continuous period of use by a user of the service. Multiple users can have sessions labelled with the same sessionId
song	string	The name of the song being played
status	long	HTTP status code
ts	long	Timestamp of the action
userAgent	string	Agent used by the user to access the service

Figure 2: Sparkify user log attributes

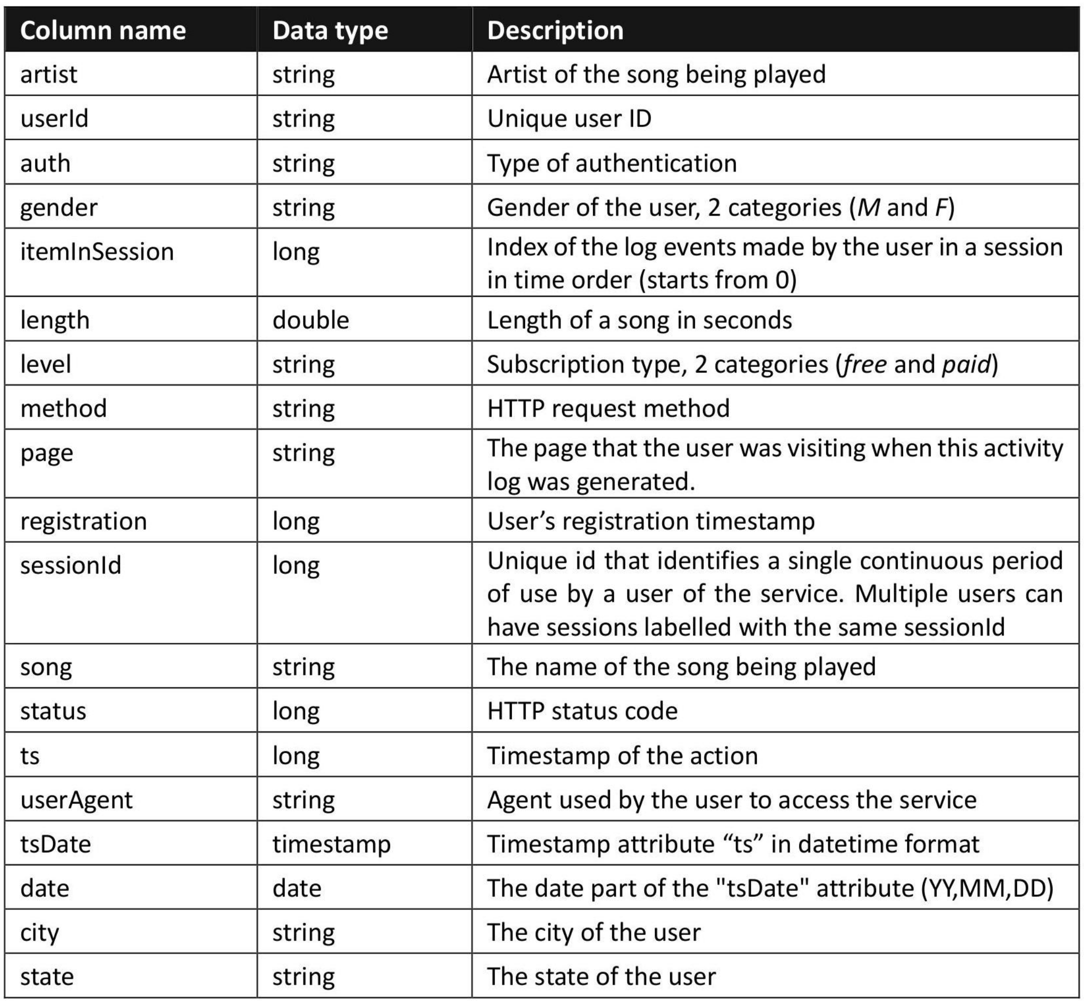
3.1 Data Preprocessing

Step 1: Drop duplicate rows.
Step 2: Identify any unregistered users with empty or null "userId" and remove rows associated with them.
Step 3: Drop two attributes "firstName" and "lastName" from the original dataset. These two columns provide no useful information related to our analysis of churned users and are hence dropped.
Step 4: Add two new attributes "tsDate" and "date".

"tsDate" stores the timestamp "ts" in datetime format. It is created by converting the original timestamp "ts" from its Unix epoch (number of seconds since January 1, 1970) format to a human-readable datetime format.

After creating "tsDate", extract only the date part of the "tsDate" attribute and store it in the "date" attribute. This step is performed using the "to_date" function, which converts a datetime object to a date object.

The resulting "tsDate" attribute will contain the timestamp in datetime format, while the "date" attribute will contain only the date part of the "tsDate" attribute. These attributes can be used for further analysis and visualization tasks.
Step 5: Split the "location" attributes into two attributes: "city" and "state". The "location" column contains the city and state information of the user. We split the composed information into 2 attributes and drop the original attribute.



Column name	Data type	Description
artist	string	Artist of the song being played
userId	string	Unique user ID
auth	string	Type of authentication
gender	string	Gender of the user, 2 categories (*M* and *F*)
itemInSession	long	Index of the log events made by the user in a session in time order (starts from 0)
length	double	Length of a song in seconds
level	string	Subscription type, 2 categories (*free* and *paid*)
method	string	HTTP request method
page	string	The page that the user was visiting when this activity log was generated.
registration	long	User’s registration timestamp
sessionId	long	Unique id that identifies a single continuous period of use by a user of the service. Multiple users can have sessions labelled with the same sessionId
song	string	The name of the song being played
status	long	HTTP status code
ts	long	Timestamp of the action
userAgent	string	Agent used by the user to access the service
tsDate	timestamp	Timestamp attribute “ts” in datetime format
date	date	The date part of the "tsDate" attribute (YY,MM,DD)
city	string	The city of the user
state	string	The state of the user

Figure 3: Preprocessed Sparkify user log attributes

The preprocessed dataset contains 19 attributes and will be used for further analysis.

In this project, some tasks will first be performed on the mini dataset to improve efficiency. After it has been tested, it will be applied to the full dataset.

4 Data Storage and Connector

4.1 MongoDB

MongoDB is a popular NoSQL document-oriented database that is designed for flexibility, scalability, and high availability. It stores data in JSON-like documents, which are flexible and can vary from document to document. This makes MongoDB well-suited for handling semi-structured or unstructured data that may not fit well into a traditional relational database. Additionally, MongoDB provides powerful and flexible querying capabilities, including ad-hoc queries, text search, and real-time aggregation.

4.2 MongoDB Connector

The MongoDB Connector for Apache Spark is a library that allows us to connect Spark and MongoDB, enabling us to utilize Spark's powerful data processing capabilities on MongoDB data. The Connector exposes all of Spark's libraries, including Scala, Java, Python, and R, and maps MongoDB data as DataFrames and Datasets. This makes it easier for us to analyze MongoDB data using Spark's machine learning, graph, streaming, and SQL APIs.

The MongoDB Connector for Apache Spark has several benefits, such as being able to leverage Spark's distributed data processing capabilities on MongoDB data. This can be particularly useful when working with large datasets that require complex processing. Additionally, it allows us to use Spark's machine learning, graph, streaming, and SQL APIs on MongoDB data. This can help us gain deeper insights into the data and make more informed decisions. Finally, the Connector maps MongoDB data as DataFrames and Datasets, which are familiar data structures in the Spark ecosystem. This makes it easier for developers to work with the data and write code that is both efficient and maintainable.

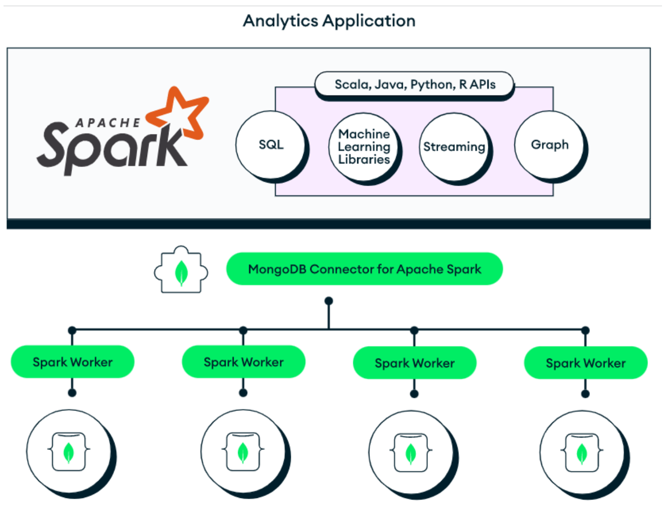


Figure 4: MongoDB Connector for Apache Spark

4.3 Spark Session

To use the MongoDB Connector and Spark in our data project, we first need to create a SparkSession object. This object represents the entry point to programming Spark with the Dataset and DataFrame API. We configure the SparkSession to use the MongoDB Spark Connector by setting the 'spark.mongodb.input.uri' and 'spark.mongodb.output.uri' configurations. These configurations specify the MongoDB database and collection we want to read from and write to, respectively. The code of creating a SparkSession object that is configured to use the MongoDB Spark Connector:

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("Sparkify") \
    .master('local')\
    .config("spark.mongodb.input.uri", "mongodb://@127.0.0.1:27017/test.sparkify1200") \
    .config("spark.mongodb.output.uri", "mongodb://@127.0.0.1:27017/test.sparkify1200") \
    .config('spark.jars.packages', 'org.mongodb.spark:mongo-spark-connector_2.12:3.0.1') \
    .getOrCreate()
```

We create a SparkSession object with the name "Sparkify". We also set the 'spark.mongodb.input.uri' and 'spark.mongodb.output.uri' configurations to specify the MongoDB database and collection we want to work with.

Once we have created a SparkSession object, we can use it to perform various data processing tasks on MongoDB data using Spark's APIs. For example, we can use Spark's DataFrame API to retrieve data from MongoDB and perform data analysis tasks. The code for retrieving data from MongoDB using the SparkSession object:

```
sparkify_df = spark.read\
    .format('com.mongodb.spark.sql.DefaultSource')\
    .option( "uri", "mongodb://127.0.0.1:27017/test.sparkify1200") \
    .load()
```

5 Exploratory Data Analysis

Data mining and machine learning are two methods of data science that help us extract useful and important patterns and trends from the data and apply them to solve real-world or corporate problems. The main purpose of EDA in this project is to analyze the characteristics of different features in the data, and which features have the potential to identify user behavior.

5.1 Definition of Churn

From a business perspective, the term "churn" pertains to customers who terminate their engagement with the company's service within a specified period. In order to prevent such attrition, it is important for businesses to identify and predict the likelihood of users leaving, and take appropriate measures to retain them.

Churned users are referred to as those who cancel the subscription. We identify churned users with *page* attribute equals to 'Cancellation Confirmation'. We also find downgrade users with *page* attribute equals to 'Submit Downgrade' and it is used as a prediction feature later. Churned users are labelled as 1 and unchurned users are labelled as 0.

Among the 1200 users, there are 267 users cancelled their subscriptions and 331 users downgrade their subscriptions. 64 users did both things. Below is the log distribution of users who cancelled their service. It is observed that cancelled users do not use the service often, as the most number of logs created by one user is no more than 100.

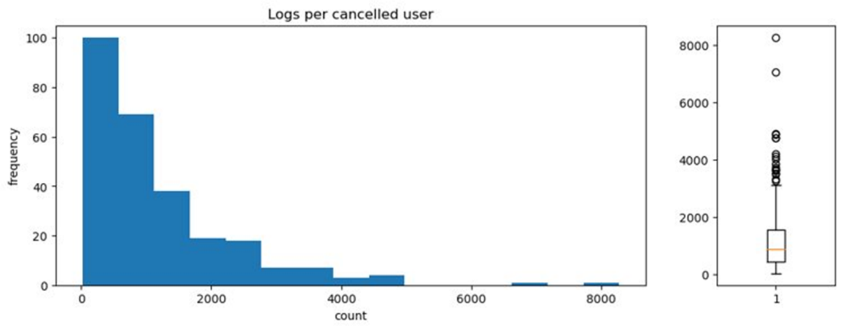


Figure 5: Logs per cancelled user

5.2 Exploratory Data Analysis

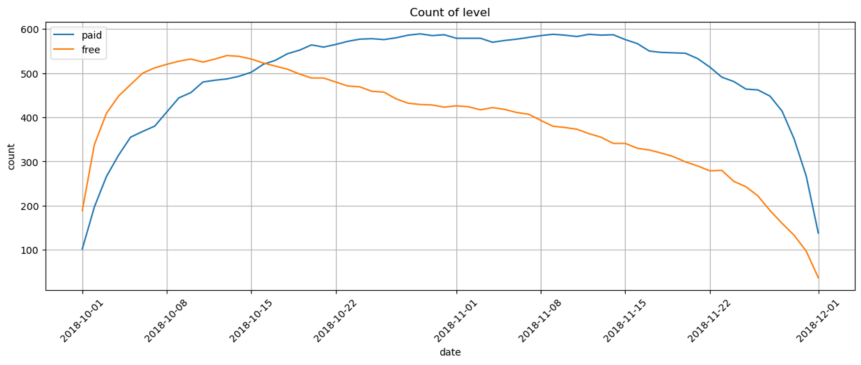
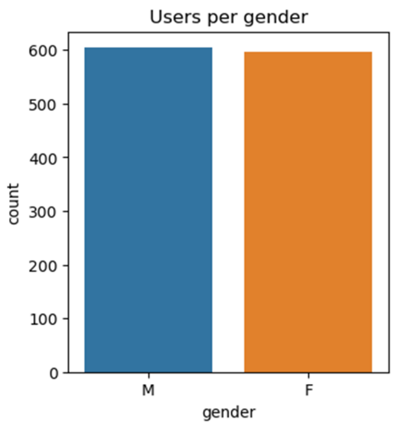


Figure 6: Change of plans during the study period

From Figure 6, we notice that over the course of the study period, there was initially a steady rise in the quantity of paid users, followed by a subsequent decline towards the end of the study period. Conversely, the number of free users exhibited an upward trend for a period of 14 days, after which there was a continuous decrease in their numbers until the conclusion of the study period.



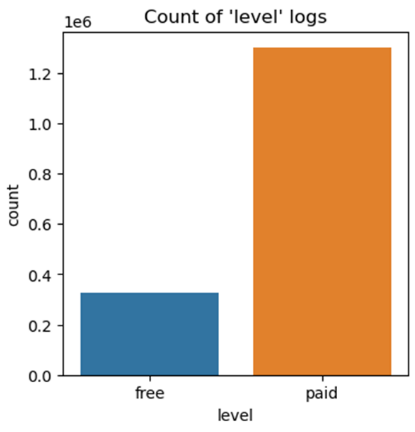


Figure 7: Users per gender and count of logs

The number of logs for paid users is four times more than that of free users and the distribution of user gender is even.

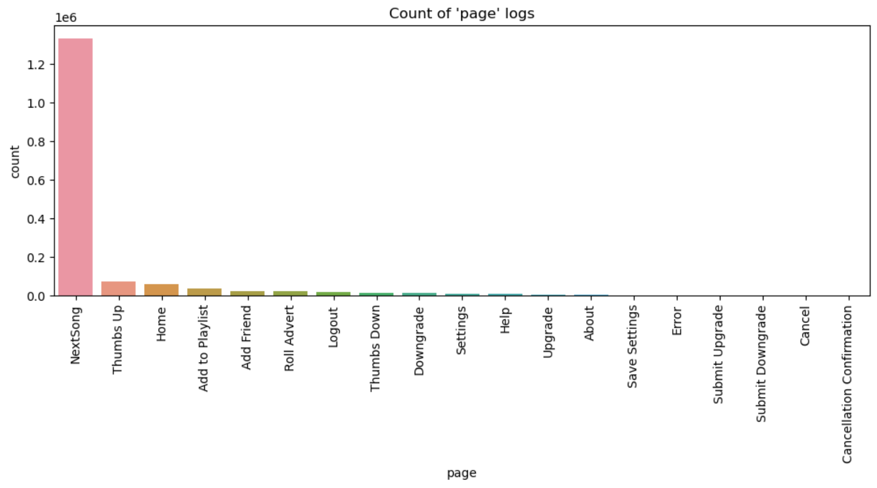


Figure 8: Count log events by page

The most frequent actions made by users are mostly related to songs, such as changing songs, thumbing up and adding to playlists.

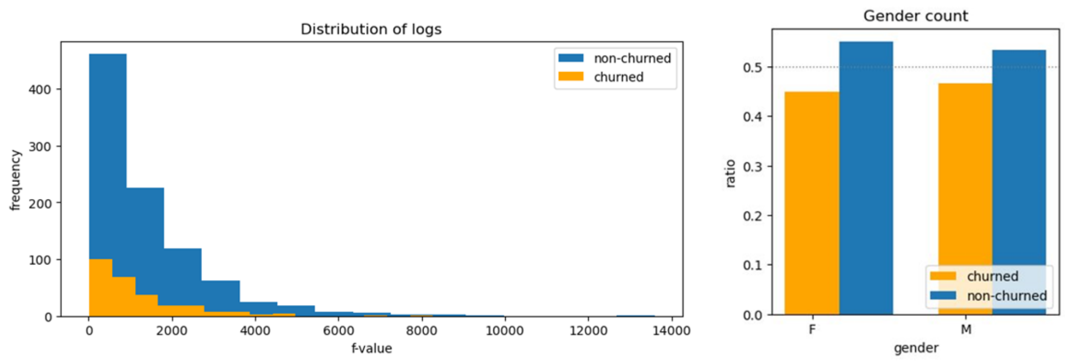


Figure 9: Distribution of logs and gender for churned and unchurned group

The left side plot of Figure 9 suggests that the churned users, on average, have a lower count of logs in comparison to users who do not churn. On the other hand, reft side plot of Figure 9 suggests that the attrition rates for both genders are comparable, indicating that neither gender is more likely to churn than the other.

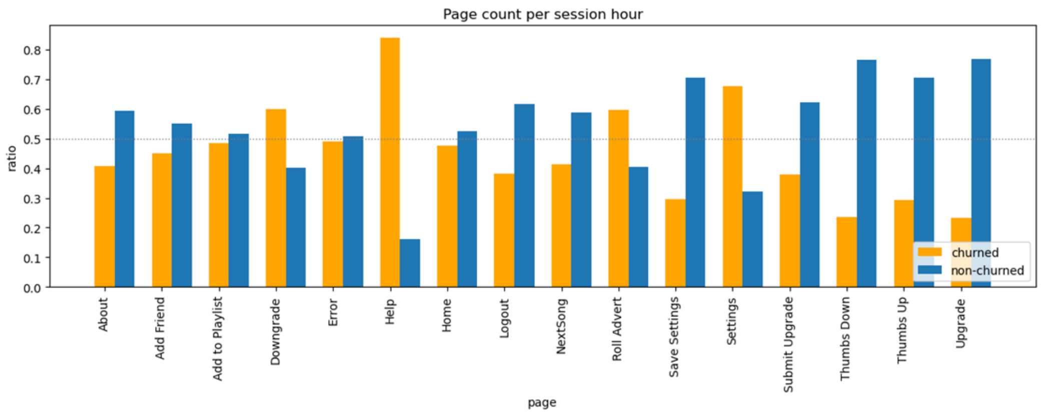


Figure 10: Page count per session hour for churned and unchurned group

Figure 10 reveals that non-churned users tend to engage more frequently in actions such as upgrades and rating content using thumbs up or down. Conversely, churned users exhibit a higher propensity towards activities such as modifying settings and seeking assistance, which may suggest that they encountered difficulties while using the service.

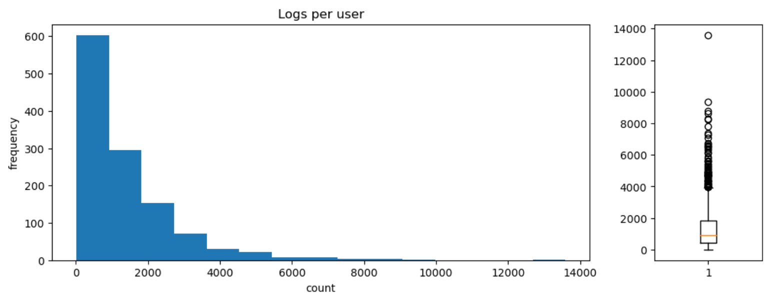


Figure 11: Logs per user

The distribution of logs per user is right-skewed. Some extreme outliers can be detected. In general, we may expect that our statistical model will be negatively affected by these outliers.

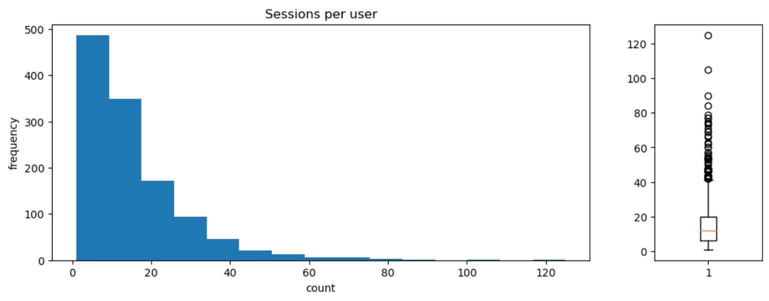


Figure 12: Sessions per user

The distribution of sessions per user is similar to the distribution of logs per user - right-skewed. It is not surprising, both distributions have a lower bound and the data with a lower bound tends to be right-skewed.

6 Data Modelling

6.1 Feature Engineering

6.1.1 General Statistic Method

To perform feature selection for classification, a statistical metric is introduced to evaluate the efficacy of each feature. The feature of a user, represented by $f(x)$, is considered significant only if it is capable of effectively distinguishing between churned and non-churned users. An aggregate function, $g(f, y)$, is employed to compute the mean or sum of the feature for all users belonging to a specific churn group, denoted by y (0 or 1). If the absolute difference between these aggregation values $|g(f, 1) - g(f, 0)|$ is significant, it indicates that feature f is effective in distinguishing between the churned and non-churned user groups. We then normalize the $g(f, y)$ values of two groups to obtain two ratios as follows:

The ratio of churned users (for a given feature f) is calculated as:

- $r_1 = \frac{g_1}{g_1+g_0}$

Similarly, the ratio of non-churned users is calculated as:

- $r_0 = \frac{g_0}{g_1+g_0}$

We use the absolute value of the difference (Δ) between them as the metric. If the absolute value $\Delta = |r_1 - r_0|$ is greater than 0.1, then the feature is considered as important and selected.

6.1.2 Feature Selection

Based on the characteristics of different features, we construct new attributes and apply the procedure described above to evaluate their validity.

Below are some examples of how we construct candidate features.

For features associated with time (e.g. song count), we first group them by time (hour, day, week or month) to find the number of songs each user played during a respective time period and then take an average of each period to extract new features. For the feature describing song count per day, we checked that the absolute delta values of this feature for both churn groups are greater than 0.1. Therefore, we would consider it as a candidate feature for the next step. Following the same procedure, song count per hour is also a candidate feature.

Song count per day of week

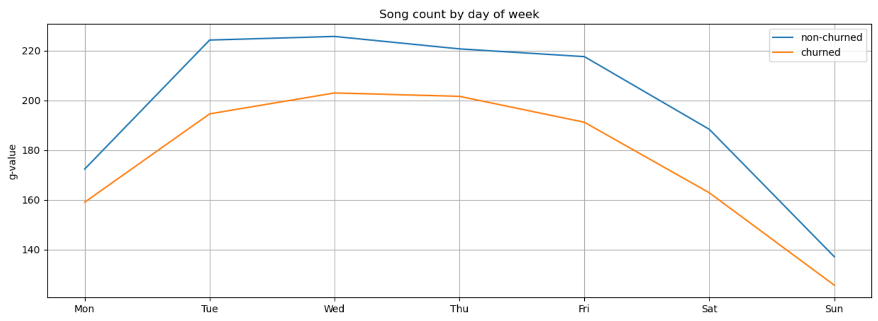
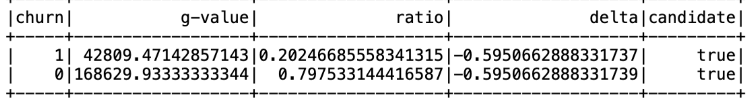


Figure 13: Visualization : Song Count Per Day of Week



```
+-----+------------------+------------------+-------------------+---------+
|churn|           g-value|             ratio|              delta|candidate|
+-----+------------------+------------------+-------------------+---------+
|    1| 42809.47142857143|0.20246685558341315|-0.5950662888331737|     true|
|    0|168629.93333333344|  0.797533144416587|-0.5950662888331739|     true|
+-----+------------------+------------------+-------------------+---------+
```

Figure 14: Statistics : Song Count Per Day of Week

Song count per hour of day

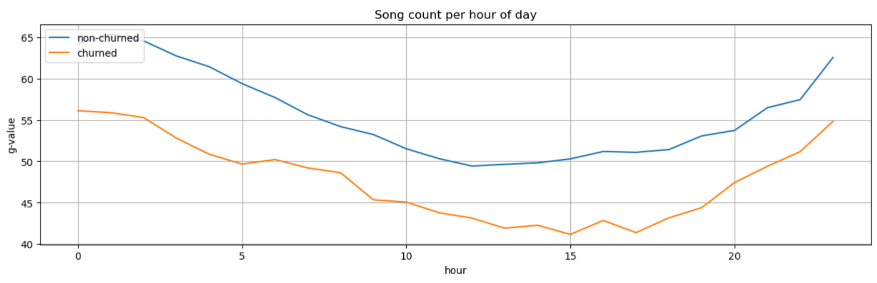
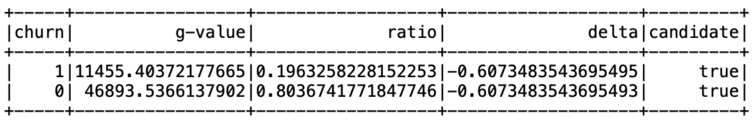


Figure 15: Visualization : Song Count Per Hour of Day



```
+-----+-----------------+------------------+-------------------+---------+
|churn|          g-value|             ratio|              delta|candidate|
+-----+-----------------+------------------+-------------------+---------+
|    1|11455.40372177665|0.1963258228152253|-0.6073483543695495|     true|
|    0| 46893.5366137902|0.8036741771847746|-0.6073483543695493|     true|
+-----+-----------------+------------------+-------------------+---------+
```

Figure 16: Statistics : Song Count Per Hour of Day

Average session gap

A *sessiongap* is a gap of no user activity between the sessions. We are using the following formula to calculate the average session gap per user and found it also important after evaluation:

- Average session gap $= \frac{observationTime - sessionTime}{sessionCount - 1}$

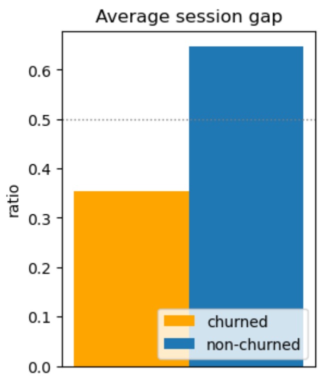
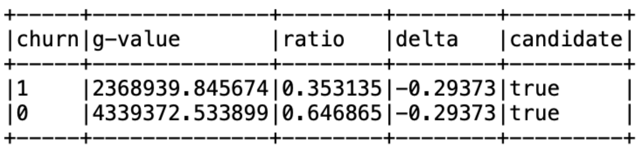


Figure 17: Visualization : Average Session Gap



```
+-----+--------------+--------+--------+---------+
|churn|g-value       |ratio   |delta   |candidate|
+-----+--------------+--------+--------+---------+
|1    |2368939.845674|0.353135|-0.29373|true     |
|0    |4339372.533899|0.646865|-0.29373|true     |
+-----+--------------+--------+--------+---------+
```

Figure 18: Statistics : Average Session Gap

After we obtained the candidate features, we select some of them to perform the modelling. We use different features for different models to achieve the best performance and below is the feature table used for GBTClassifier. The construction of the features could be interpreted from the newly constructed feature names.

Category	Attribute
User	f_Gender
User	f_LastLevel
Log	f_LogCount
Song	f_SongCount
Song	f_NonSongCount
Song	f_AboutCount
Log	f_ThumbsUpCount
Log	f_RollAdvertCount
Session	f_SessionCount
Session	f_AvgSessionLength
Session	f_AvgSessionGap
Session	f_DowngradePerSessionHour
Session	f_ErrorPerSessionHour
Session	f_SettingsPerSessionHour
Session	f_SaveSettingsPerSessionHour
Session	f_LogoutPerSessionHour
Log	f_RollAdvertPerHour
Log	f_ThumbsDownPerHour
Log	f_UpgradePerHour
Log	f_SubmitUpgradePerHour
Session	f_SessionsPerDay
Log	f_AddFriendPerDay
Log	f_RollAdvertPerDay
Log	f_ThumbsDownPerDay
Log	f_ThumbsUpPerDay,
Song	f_TotalSongLength
Song	f_UniqueSongCount
Song	f_UniqueSongShare

Table 1: Feature table for GBTClassifier

6.2 Prediction Model

Firstly, we need to split the full dataset into train, test, and validation sets. Then, we would test several machine learning methods and evaluate the accuracy of each model. The parameters should be tuned to improve the performance of the models. Once we have determined the accuracy of each model, we would select the winning model and report the results on the validation set.

6.2.1 Metrics

Correct measurement is crucial for evaluating machine learning models. In this task, we will evaluate the performance based on the below metrics.

- TP: The predicted result of the classifier is a positive sample, which is also the actual number of positive samples that have been correctly identified.
- FP: The classifier predicts a positive sample, but in reality, it is a negative sample, which is the number of negative samples that are falsely reported.

- TN: The classifier predicts negative samples, but in reality, it is negative samples, which is the number of correctly identified negative samples.
- FP: The classifier predicts a negative sample, but in reality, it is a positive sample, which is the number of missed positive samples.
- $Accuracy = \frac{TP+FP}{numberofsamples}$
- $Precision : \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1 = 2 * \frac{precision*recall}{precision+recall}$

Among them, the F1 score is the indicator we are most concerned about. This is mainly based on two aspects.

First, in the dataset, the scale of churned users is much smaller than that of unchurned users. In the case of class imbalance, accuracy may be misleading and deviate from our goals. F1 score is the indicator we are most concerned about. At the same time, the values of precision and recall will also be referenced, as they are the basis for calculating the F1 score.

Second, Our task is based on an abstract business world. Sparkify is neither a company willing to reduce churn at any cost, nor a company that only focuses on whether our predicted churn includes all churn users, nor a company that pursues high-precision classification. Therefore, we hope to strike a balance between precision and recall indicators.

For these reasons, we will use F1 score as our principal metric.

6.2.2 Procedure

The steps of machine learning experiments are:

- Preliminary exercises: A preliminary analysis was conducted on the mini dataset, utilizing default hyperparameters and a diverse range of feature collections. The objective of these exercises was to obtain initial estimates and assess the efficacy of the selected feature set. Based on the findings, further modifications such as deletion or addition of features were made to refine the feature selection process.
- Cross-validation: Following the preliminary analysis on the mini dataset with various feature collections, hyperparameter tuning was performed using cross-validation. A single feature collection was selected based on the best-performing model. This approach was applied to further refine the model and optimize its performance.
- Correlation analysis: Conduct the correlation analysis between the features of the best model and drop co-linear features with strong correlation. Rebuild the model on the new feature set.
- Final version: The experiments conducted on the mini dataset were subsequently replicated on the full dataset.

6.2.3 Results

Here we would like to show the results of the final version - the experiments on the full dataset. The prediction analysis was done using the following algorithm: Logistic Regression and Ensemble Methods (Random Forest and GradientBoosting DecisionTree)

Logistic Regression - The logistic regression is implemented as a linear model for classification. In this model, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function. The numerical output of the logistic regression, which is the predicted probability, can be used as a classifier by applying a threshold (by default 0.5) to it.

Random Forest - In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. After obtaining the forest, when a new input sample enters, each decision tree in the forest is asked to make a judgment separately to see which class the sample should belong to (for classification algorithms), and then to see which class is selected the most, to predict which class the sample belongs to.

GradientBoosting DecisionTree - It is a generalization of boosting to arbitrary differentiable loss functions. GBDT implement the optimization process of learning using additive models and forward distribution algorithms.

Below is the result that we obtained.

	Accuracy	Precision	Recall	F1-score
GBTClassifier	0.837	0.834	0.837	0.835
Random Forest	0.846	0.866	0.846	0.824
Logistic Regression	0.837	0.831	0.837	0.825

Table 2: Performance of models

	Accuracy	Precision	Recall	F1-score
GBTClassifier	0.867	0.863	0.867	0.863
Random Forest	0.852	0.850	0.852	0.840
Logistic Regression	0.837	0.830	0.837	0.825

Table 3: Performance of best cv models

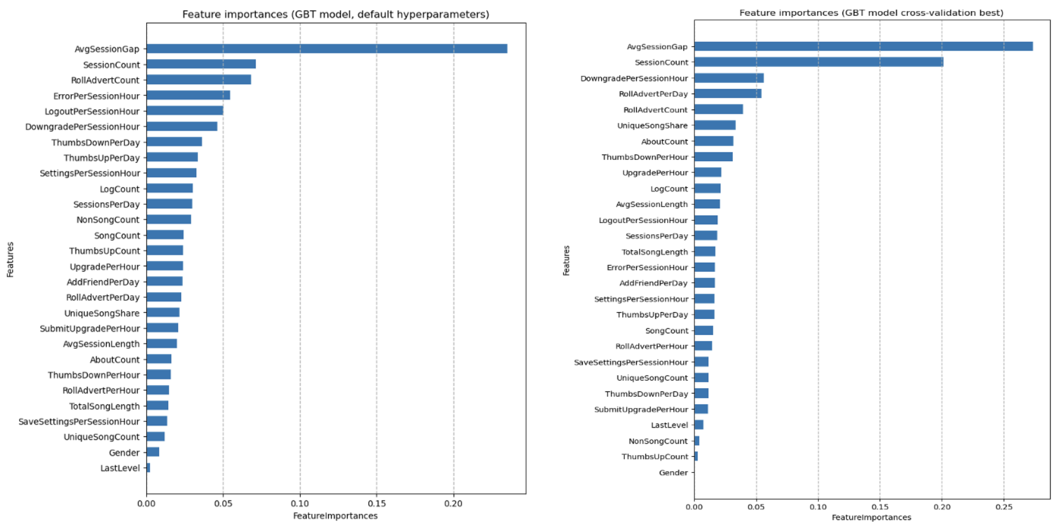


Figure 19: Feature importance of GBTClassifier

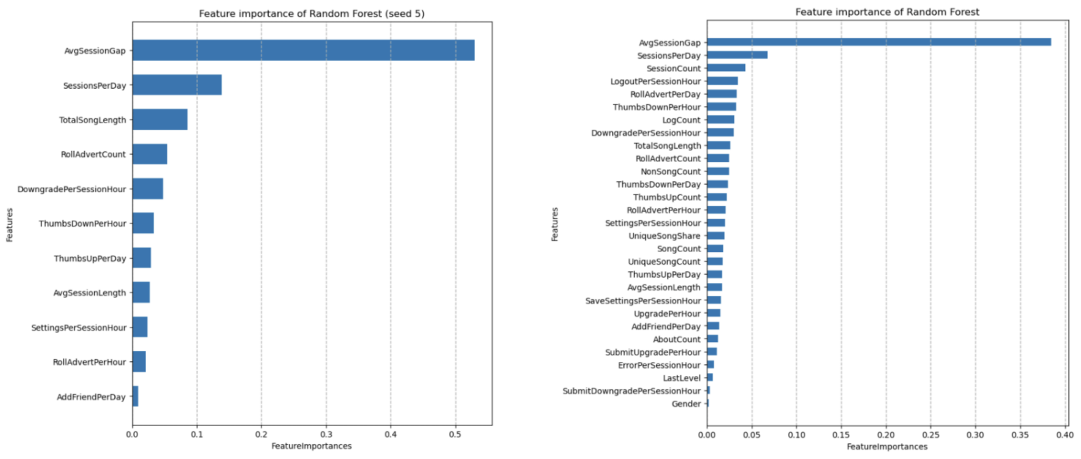


Figure 20: Feature importance of Random Forest

We can see that GBTClassifier has the best performance among them after tuning hyperparameters with cross-validation. Here shows the correlation coefficients of the best model.

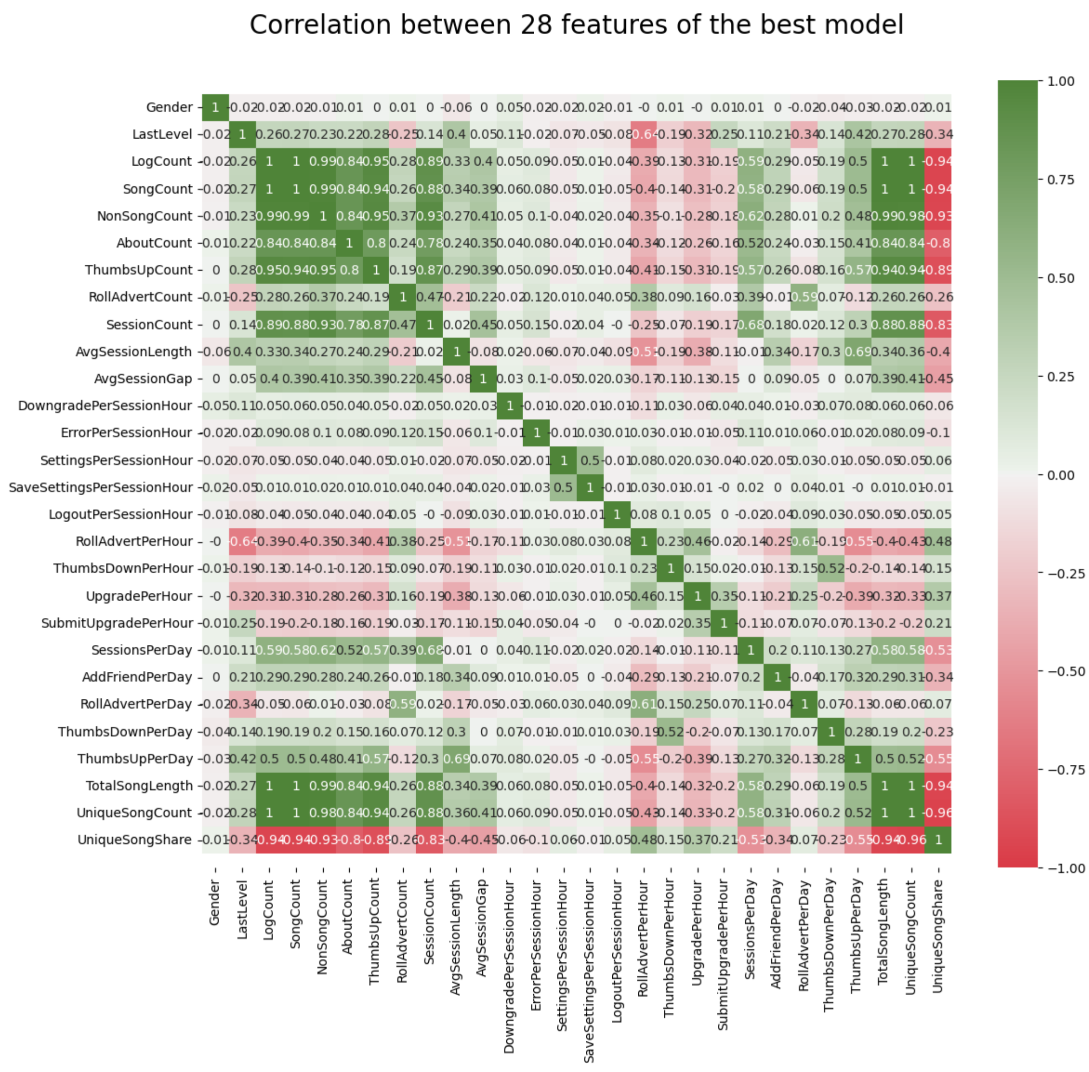


Figure 21: Correlation analysis between features of best model

The green areas in the figure represent a positive correlation: the stronger the green, the stronger the positive correlation between variables. Similarly, areas with a red colour indicate a negative correlation: the stronger the red colour, the stronger the negative correlation between variables. Bright grey indicates areas with weak correlation.

We can find several variables with similar correlation coefficients. These variables have a very strong relationship with each other. After removing them, we rebuilt the GBTClassifier model.

	Accuracy	Precision	Recall	F1-score
GBTClassifier	0.852	0.847	0.852	0.846

Table 4: Performance of best model after removing highly correlated features

We can see the performance slightly downgrades, indicating a potential for further feature engineering but it is still acceptable.

6.3 GraphFrames

GraphFrames is a graph processing library developed by Databricks that provides a high-level API for building and analyzing graphs in Apache Spark. Unlike GraphX, GraphFrames focuses on graph computations over distributed systems while using the DataFrame and SQL APIs to manipulate both vertices and edges of the graph. GraphFrames is built on top of Apache Spark SQL, enabling us to perform graph analytics in conjunction with SQL and DataFrame operations.

Based on our data, we have built an artist influence network using GraphFrames. We used the artist in the mini dataset as the vertices of the network and the number of listeners of an artist as its vertex size. When two artists have the same listenership, we consider that there is an edge between them. The edge is undirected, and we use the number of common listeners as the weight of edges. Several most influential artists are listed in table 5, and the influential relationship between them is shown in table 6.

To ensure the representativeness of the influential network, we selected artists with more than 150 listeners as vertices when constructing the network. Spark SQL is used to process the data. Firstly, we calculated the number of listeners for each musician and selected the musicians with more than 150 listeners. Then, we constructed the set of listeners for each musician. If the sets of listeners of two musicians intersect, we consider that there is an edge between these two musicians, and the length of the intersection (the number of listeners listening to the music of both musicians) is the weight of the edge. Once the artist influence network was constructed, we exported the data and visualised this network using the following tool: cosmograph.app (https://cosmograph.app/).

Artist(vertex)	Number of listeners(size)
Kings Of Leon	199
Coldplay	189
Dwight Yoakam	189
Florence + The Machine	187
The Black Keys	179
BjÃ Â¶rk	179
Justin Bieber	177
Taylor Swift	173
Jack Johnson	173
Harmonia	172

Table 5: Top ten most influential artists

Artist1(src)	Artist2(dst)	Number of common listeners(weight)
Kings Of Leon	Coldplay	181
Kings Of Leon	Dwight Yoakam	181
Kings Of Leon	Florence + The Machine	179
Kings Of Leon	The Black Keys	176
Kings Of Leon	BjÃ Â¶rk	172
Kings Of Leon	Justin Bieber	173
Kings Of Leon	Taylor Swift	170
Kings Of Leon	Jack Johnson	167
Kings Of Leon	Harmonia	166

Table 6: Sample of the influential relationship between artists

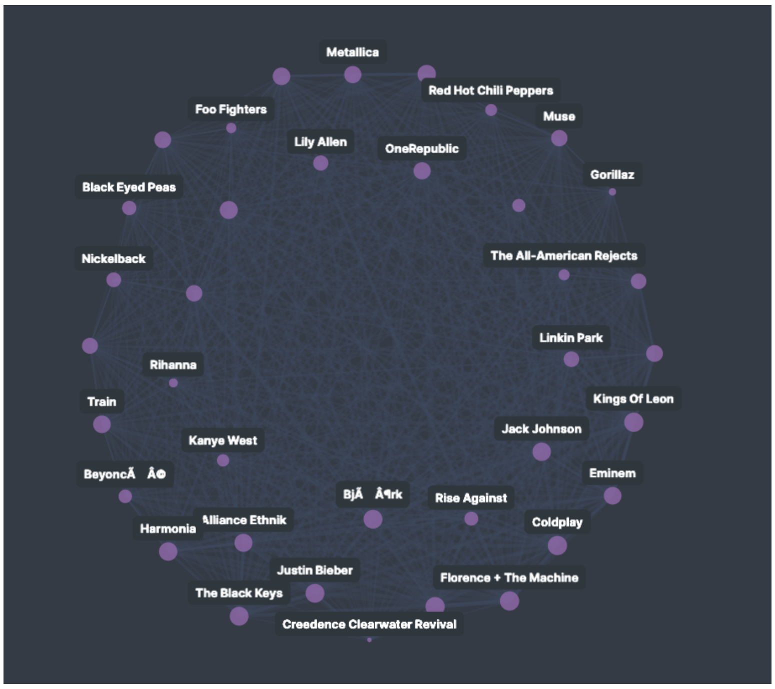


Figure 22: Artist Influence Network

7 Conclusion

Throughout the experiments on the user actions, we found that whether users would churn could be predicted by their behaviour on the platform. Certain features, such as Average Session Gap and Average Session Length, are highly correlated with whether or not churn occurs. Most of such behaviours belong to the session level. This is in line with intuition. If users frequently enter music software and engage in activities, they often have a high level of loyalty to the platform and will not churn. Additionally, some features that describe the behaviour of users after logging in also contribute significantly to the prediction model. For example, ThumbsDownPerhour and UniqueSongShare describe how users interact with each other on the platform. The stronger the user's willingness to interact on the platform, the lower the churn rate.

Considering the performance of models predicting the churn case, the gradient-boosting decision tree classifier has the best performance, with a 0.846 F1-score and 0.852 accuracy on the full dataset. This suggests that machine learning models can identify lost users based on past user activities and interactions with music streaming services with considerably high accuracy and could be applied to enterprises to prevent heavy economic losses.

An artist influence network is also built on the mini dataset, with the artists that have more than 150 listeners as vertices and common listeners between two artists as edges. The number of common listeners is used as edge weight. The resulting network is visualized using cosmograp.app.

8 References

[1] Kumar, S. (2022, December 13). Council post: Customer retention versus customer acquisition. Forbes. Retrieved from https://www.forbes.com/sites/forbes-businesscouncil/2022/12/12/customer-retention-versus-customer-acquisition/?sh=25de82f71c7d

[2] MongoDB. (n.d.). Spark Connector. Retrieved from https://www.mongodb.com/products/spark-connector.

[3] Friedman, J.H.. Stochastic gradient boosting. Computational Statistics & Data Analysis, 38, 367-378. 2002

[4] Breiman, "Random Forests", Machine Learning, 45(1), 5-32, 2001.

[5] Christopher M. Bishop: Pattern Recognition and Machine Learning, Chapter 4.3.4 ' [6] R. Kohavi, A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, Intl. Jnt. Conf. AI

[7] Voron, A. (2019, March 26). Churn Prediction with Sparkify. Retrieved from https://medium.com/@amos.voron/churn-prediction-with-sparkify-6f9127da7235