Contents

[Project Overview 2](#_Toc76155647)

[Problem statement 2](#_Toc76155648)

[Metrics 2](#_Toc76155649)

[Data Exploration 2](#_Toc76155650)

[Data Pre-processing 4](#_Toc76155651)

[Data Analysis and Visualization 5](#_Toc76155652)

[Data Featuring: 7](#_Toc76155653)

[Initial Model implementation 8](#_Toc76155654)

[Model implementations and Refinement 11](#_Toc76155655)

[Model evaluation and validation 13](#_Toc76155656)

[Justification 15](#_Toc76155657)

[Reflection 15](#_Toc76155658)

[Improvement 15](#_Toc76155659)

[References 16](#_Toc76155660)

[Machine Learning with PySpark and MLlib — Solving a Binary Classification Problem 16](#_Toc76155661)

[How to install PySpark and Jupyter Notebook in 3 Minutes 16](#_Toc76155662)

# Project Overview

Sparkify project aims for predicting churn rate by analysing a large user-activity dataset with Apache Spark technology. The dataset we use is 12GB in size and is located on the Udacity S3 location. We are going to transform the big dataset into user based dataset. We are going to apply a number of classification models to train the dataset for identifying the best model for predicting customer churn.

# Problem statement

Online streaming music provider, Sparkify has quite high churn rate of customers. The reduction of churn rate will make the business more profitable. The problem Sparkify is facing is that it has difficulty to identify the potential customer churn in time.

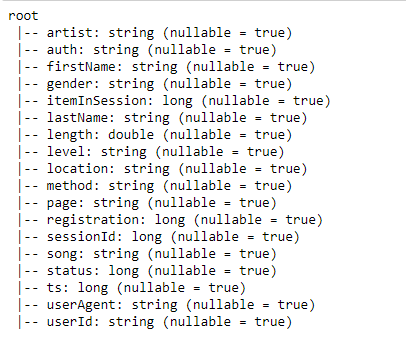
An accurate churn prediction model for identifying potential churn customers will be the key to keep the business profitable and sustainable.

# Metrics

We implement three classification models – Gradient Boosting Trees, Logistic Regression, and Random Forest with a number of defined parameters to train the 80% of dataset. Then we use the rest 20% dataset to test the models to find out the model with highest Test Area under ROC. The reason we choose Test Area under ROC is due to the limitation of Spark ML.

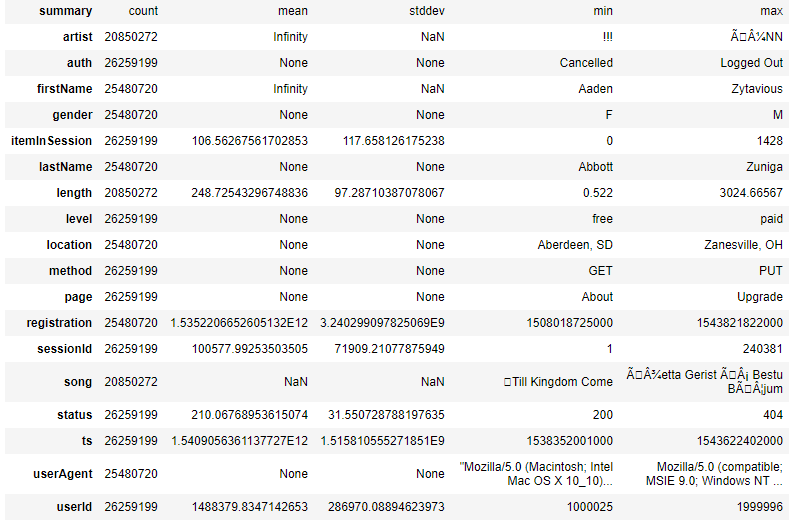
# Data Exploration

After loading the JSON formatted data file, below is the list of data attributes. Each row is a user action on the streaming app on the user’s device. There are approximately **25 million** data points. Each data point has the following attributes.

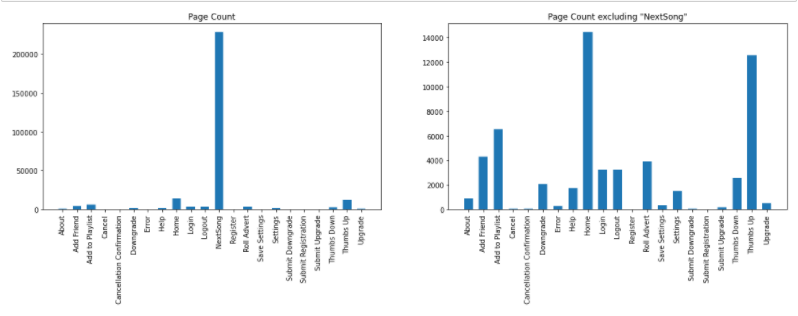




We try to do a descriptive analysis on the data points – see below. However, after calculating mean, max, standard deviations on the attributes, we did not find any interest points we could do further exploration – see the table below.



‘Page’ column indicates the user’s action – when a user click the sparkify streaming apps on its device, the action will be recorded as the data point. A user action can be ‘Play Song’, ‘Log In’, ‘Log Out’, ‘Downgrade’, etc. We analyse ‘Page’ data by grouping each type of page. The number of ‘NextSong’ actions is significantly higher than other actions. After excluding ‘NextSong’, ‘Home’, ‘Thumbs Up’, and ‘Add to Playlist’ seems to be the most popular actions. This page data looks interesting and will be used for our modelling implementation later on.



There are also a number of rows which have no user id. We exclude them. ‘Cancellation Confirmation’ action is used to determine if a user has cancelled its membership.

# Data Pre-processing

In high level, we transform this user-action based large dataset into user-based dataset. After transformation, each record represents one user with all its aggregated data.

These are the steps to the transformation processes:

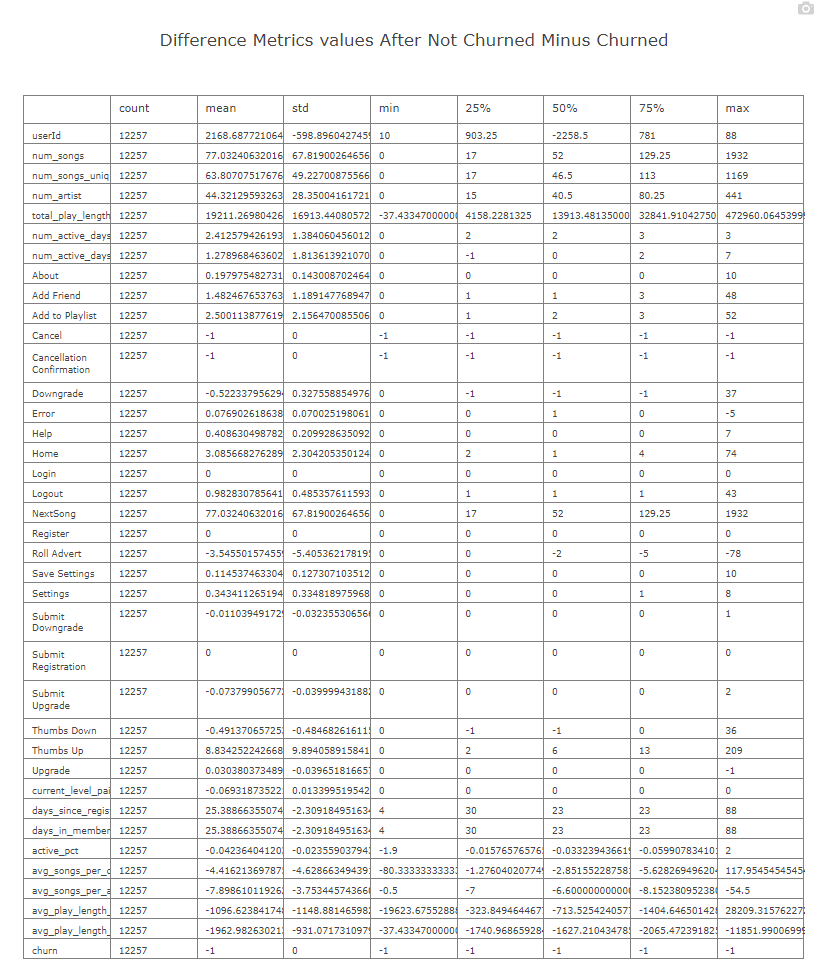
Refer Jupyter Notebook - *sparkify\_data\_engineering* for details

(<https://github.com/zhangshen20/capstone-project-sparkify/blob/master/sparkify_data_engineering.ipynb>)

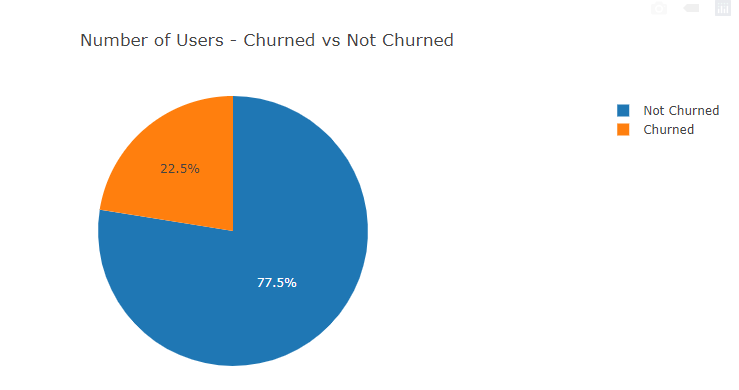
* Load JSON file into Spark Dataframe
* Remove all records with no user id.
* Use ‘Cancellation Confirmation’ as flag to decide if a user has churn. Create a list of users who have ‘Cancellation Confirmation’ action and load into a separated dataframe.
* Convert timestamp format into date format for column ‘ts’ and column ‘registration’.
* Calculate the number of days since a user has registered
* Group data into userId level.
* Calculate
  + number of songs,
  + total play length,
  + number of active days,
  + number of active paid days,
  + number of days in member,
  + active percentage over number of days since registration
  + average songs played per day
  + average songs played per active day
  + average play length per day
  + average play length per active day
  + member level either ‘paid’ or ‘free’
  + number of artists listened
* Transpose User actions (‘Page’) and create action count for each type pf user action.
* Add churn flag to the dataframe by referencing the churn user-list dataframe.
* Create a churn user dataframe
* Create a not churn user dataframe

# Data Analysis and Visualization

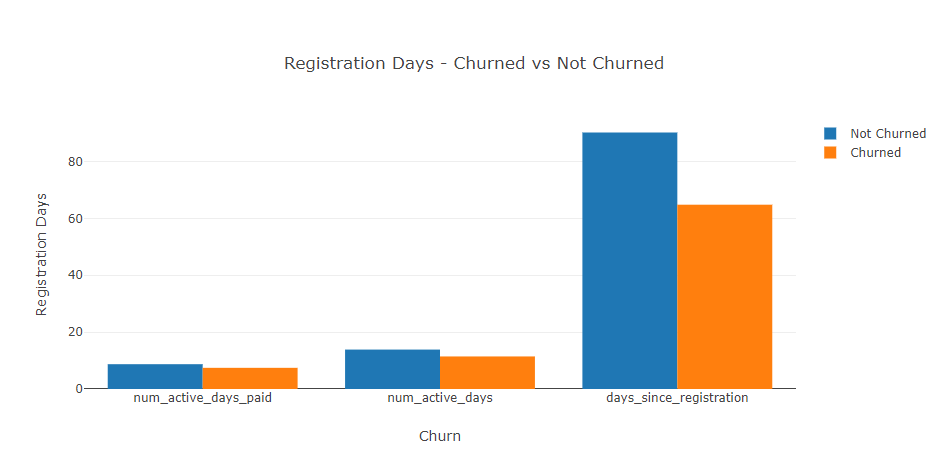
By analysing the dataframe, we identify the difference between 2 groups – see table below



The percentage of users in the 2 groups is shown in the pie chart.

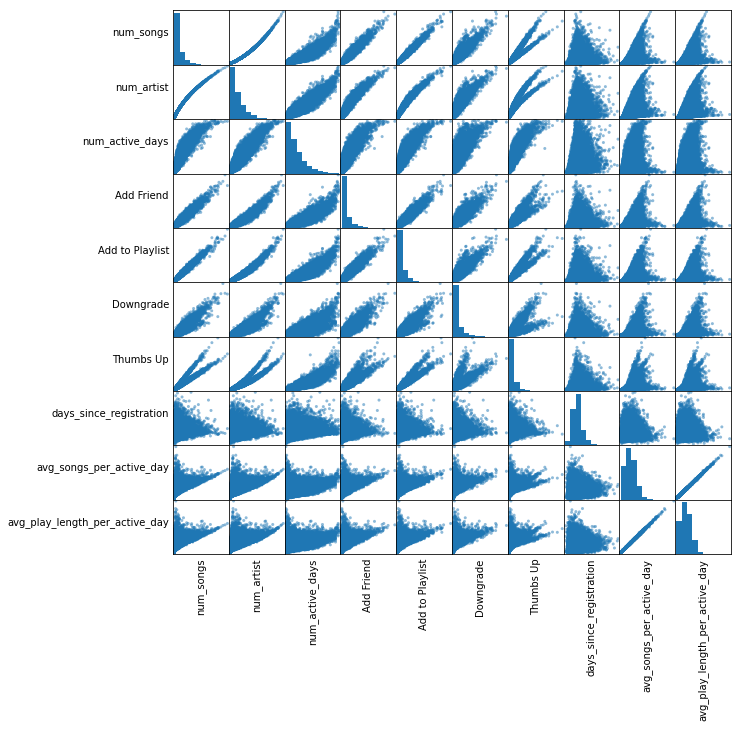


And their days’ metrics is shown in the bar chart below.



# Data Featuring:

We create a scatter matrix to analyse the correlation between each data attributes in the user-level dataframe – see the matrix below

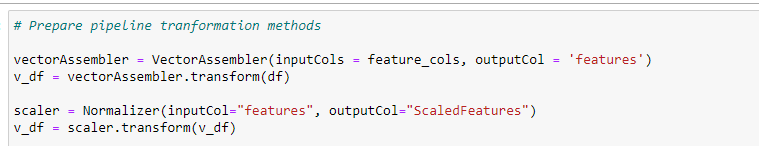


We try to select independent attributes which are:

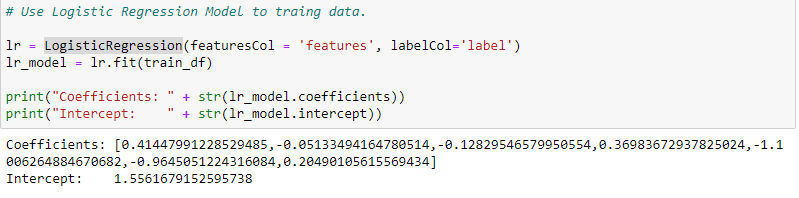
* 'num\_active\_days',
* 'Add Friend',
* 'Add to Playlist',
* 'Downgrade',
* 'Thumbs Up',
* 'days\_since\_registration',
* 'avg\_songs\_per\_active\_day'

# Initial Model implementation

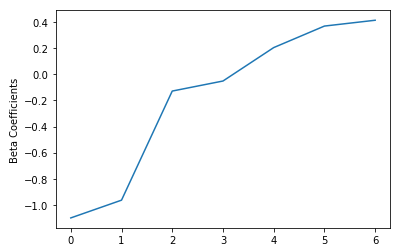
We use ‘vectorAssember’ to vectorize the attributes and use Normalizer to normalise the vectors.



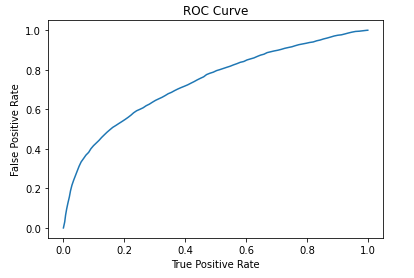
We split data into 80% training data and 20% test data. We use ‘Logistic Regression’ as first classification model.



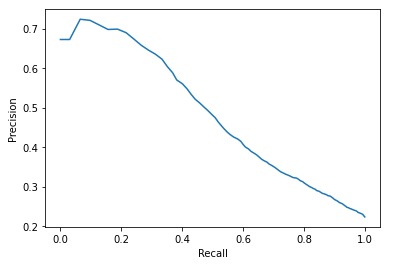
Beta Coefficients:



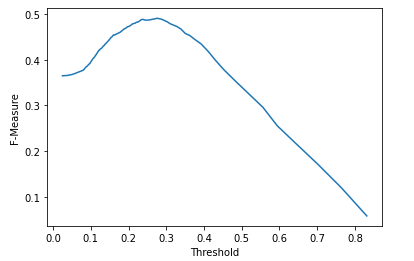
Training under RPC:



Precision:



F-Measure:



We use 20% test data for prediction. The Test Area Under ROC is 0.740



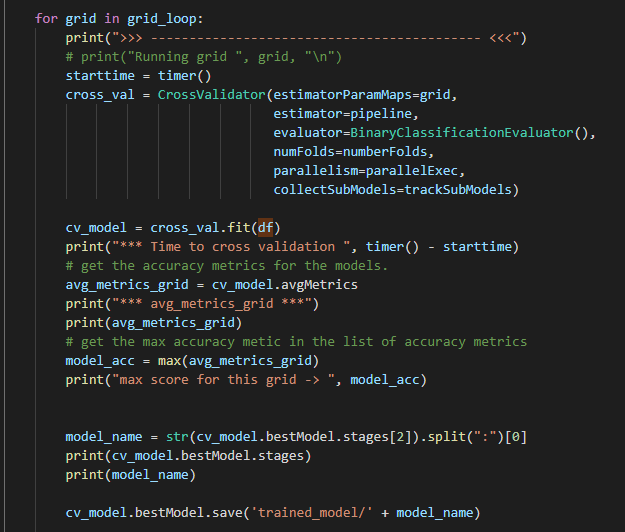
# Model implementations and Refinement

After initial model implementation, we use three classification models – Gradient-Boosted Tree, Logistic Regression, and Random Forest Classifier. We also apply Cross Validation technique with ParameterGrid to optimize each of the models.

In order to efficiently implement models and train models, we build machine learning pipelines under Spark ML frame which save us lots of model-training time.

The two screen shots below are the code to build ML pipeline and apply cross validator to each of 3 models with define parameter grid. Then we save the optimized models of each classification technique into model files.

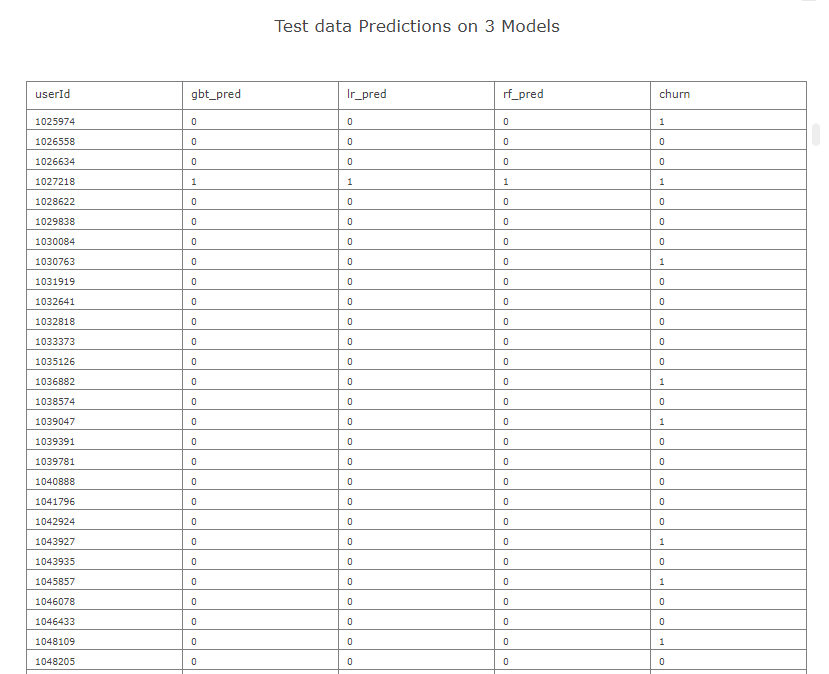


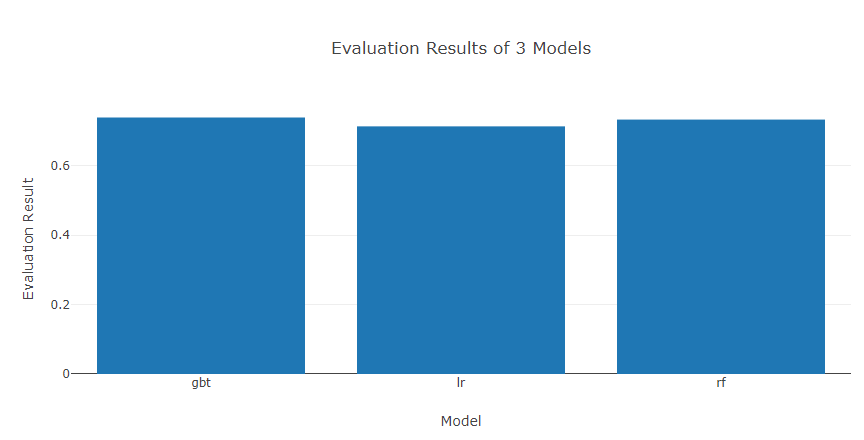


# Model evaluation and validation

We use split data into 80% training data and 20% test data. After we train the models, we apply the test data to evaluate the model results. Below are the tables presenting the results.

The Test Area under ROC of the 3 optimized models are shown in the bar chart below.





We trained each of 3 models using 3-fold cross validation with a number of pre-define parameter grids. The average metrics coming from training results is very close to the Test Area ROC from the test dataset. The test result is aligned to the training results. In addition, we choose the best result from the 3 train models. The Gradient-Boosted Tree model has 0.739 Test Area under ROC comparing Logistic Regression 0.713 and Random Forest 0.733. Therefore, we are confident that the model is robust and well chosen.

# Justification

Base on the test result, the trained gradient-boosted tree model is the best one among the 3 models. However, all of the 3 results all have their Test Area under ROC above 0.7 and they are all very close. We cannot say one model definitely out-performs the others. The model we chose is purely based on the machine learning results and the parameters we defined. Further analysis and modelling work would be required as in future we are having more data points in use. At this time, we only can say that based upon the 28 million points, we use the trained gradient-boosted tree model for customer churn prediction.

# Reflection

In this project, we start with using apache spark to pre-precess data. We collect approximately 25 milliion data points. Each data point is a user’s action on its music streaming device. We transform the data point into user based dataset. Each record represent one user’s summerized activities since he/she joined in the memberhip. We setup machine learning pipelines by using vectorizing and normalizing technique with conjuntion 3 classification modelling techniques for preparing for prediction model. For optimizing models, we use cross validator with 3-fold data split technique and a set of pre-defined parameters to find the best prediction model. At the end, Gradient-Boosted Tree model is the best model for churn prediction. Finally, all the project works are implemented under Spark framework, which saves us significant time and energy.

# Improvement

Further improvement can be done. We can use other classification model technique such as Decision Tree. And we can add more grids for Parameter Grid Builder. For example, to optimize Gradient-Boosted Tree model, we can add more options on *maxDepth, maxBins and maxIter*. I believe once we train model by using more grids, we could be able to identify better prediction models.

# References

# Machine Learning with PySpark and MLlib — Solving a Binary Classification Problem

<https://towardsdatascience.com/machine-learning-with-pyspark-and-mllib-solving-a-binary-classification-problem-96396065d2aa>

# How to install PySpark and Jupyter Notebook in 3 Minutes

<https://www.sicara.ai/blog/2017-05-02-get-started-pyspark-jupyter-notebook-3-minutes>

How to set up Anaconda and Jupyter Notebook the right way

<https://towardsdatascience.com/how-to-set-up-anaconda-and-jupyter-notebook-the-right-way-de3b7623ea4a>

Step-by-step solution with f1-score as a metric

<https://www.kaggle.com/devghiles/step-by-step-solution-with-f1-score-as-a-metric>