**Capstone Project Steven Greulich**

**Data Science Engineer Nanodegree February 10, 2019**

**Project Definition**

## **Project Overview**

Millions of users stream heir favorite songs through our service through the free tier that plays advertisements between songs, or using the premium subscription model that plays songs ad free, but pay a monthly fee. Users can upgrade, downgrade or cancel their service at any time. It is crucial that the users love the service. Every time a user interacts with a service (i.e. downgrading a service, playing songs, logging out, liking a song, hearing an ad, etc.) it generates data. All of this data contains the key insights to keeping the users happy and allowing the company to thrive. It is There is a fictional online music streaming site, Sparkify, in which we are interested in seeing if we can predict which users are at risk of cancelling their service. The goal is identify these users before they leave so they can hypothetically receive discounts and incentives to stay.

## **Problem Statement**

The overall goal of this project is to create a predictor system that can determine if a user is at risk of cancelling their subscription.

The tasks in order to do this is as follows:

* Download the “medium-sparkify-event-data.json” data
  + This data is located at the following file location: <https://s3.amazonaws.com/video.udacity-data.com/topher/2018/December/5c1d6681_medium-sparkify-event-data/medium-sparkify-event-data.json>)
* Preprocess the dataset to remove any missing values and to encode non-numerical values
* Train a classifier that can determine users that are at risk of cancelling their subscription
* Predict new users to determine the accuracy

Note: if the metrics of the predictions are low, performing model tuning as necessary.

## **Metrics**

Since this is a binary classifier (0 if the user is not at risk of cancelling their subscription and 1 if they are), we will be using the accuracy metric to determine how are model is doing on both the Training and Testing sets.

Accuracy = (True Positives + True Negatives) / Dataset Size

**Analysis**

## **Data Exploration**

After initial loading of the dataset, I noticed that the dataset has 286,500 log entries with 18 unique columns. These 286,500 entries are not unique users, but rather a totality of transactions that (GET USER COUNT HERE).

The columns in the database are as follows:

* **Artist**: The artist being listened to
* **Auth**: Whether or not the user is logged in
* **FirstName**: First name of the user
* **Gender**: Gender of the user
* **ItemInSession**: Item number in session
* **LastName**: Last name of the user
* **Length**: Length of time for specific log
* **Level**: Free or Paid user
* **Location**: Physical location of user
* **Method**: Get or Put requests (Web calls)
* **Page**: Which page are they on in the site
* **Registration**: Users registration number
* **SessionId**: Session ID
* **Song**: Song currently being played
* **Status**: Web status (200 for OK, etc.)
* **Ts**: Timestamp of current log
* **UserAgent**: Type of browser user is on
* **UserId**: UserID for user

One particular column that we need to pay attention to is the “Page” column. This is where we can see what pages the user is on.

We have the following values that are in the “Page” column.

* Add Friend
* Add to Playlist
* Cancel
* Cancellation Confirmation
* Downgrade
* Error
* Help
* Home
* Logout
* NextSong
* Roll Advert
* Save Settings
* Settings
* Submit Downgrade
* Submit Upgrade
* Thumbs Down
* Thumbs Up
* Upgrade

The pages we never want to see users utilize is the “Cancel” and especially the “Cancellation Confirmation”.

## **Data Visualization**

Figure 1 represents the first row in the dataset just to get a feel of the type of data that are in each column:

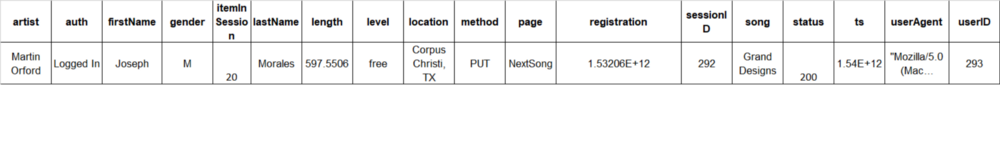


Figure 1 **Note:** To condense the columns into a readable format, the “Registration” and “TS” columns have been condensed.

Now that we had a preliminary look of the columns and what we might expect in each row, let us take a look at how many rows at least one (or more) values that are missing have.

For the original dataset, we had a count of 543,705 records in the data. After running the command to remove any rows that have at least 1 missing value, we still had a total of 543,705.

One other thing that we should explore are the datatypes of each column. Typically, machine-learning algorithms require numeric values. Any columns that we would eventually select must be in numeric value if not already. If the specified columns are not numeric, we will need to convert them utilizing either label encoding or one hot encoding.

* **artist**: string
* **auth:** string
* **firstName:** string
* **gender:** string
* **itemInSession:** long
* **lastName:** string
* **length:** double
* **level:** string
* **location:** string
* **method:** string
* **page:** string
* **registration:** long
* **sessionId:** long
* **song:** string
* **status:** long
* **ts:** long
* **userAgent:** string
* **userId:** string

**Methodology**

## **Data Preprocessing**

Now that we have performed some data exploration, we will need to convert as necessary so that we can feed or data into a machine-learning model.

First, let us tackle missing values. Fortunately for this specific dataset, we did not have any missing values.

Now, let us define and determine the churn column. “Churn” for the purposes of this discussion, shall be referenced as anyone, whether a free or paid user, successfully completes the “Cancellation Confirmation” page. I created a column called “page” that will indicate a 1 if they are a person who has cancelled and 0 if they have not cancelled their subscription.

Another item that we need to look at is which columns are numeric and which are non-numeric. These column data types come into play when we choose the columns we want to put into our Machine Learning model. Any columns that we choose for the model must be of numeric type. If we choose non-numeric columns, we will need to convert them so that they do become numeric (whether through label encoding or one hot encoding).

Listed below are the column and their respective data types:

* **artist**: string
* **auth:** string
* **firstName:** string
* **gender:** string
* **itemInSession:** long
* **lastName:** string
* **length:** double
* **level:** string
* **location:** string
* **method:** string
* **page:** string
* **registration:** long
* **sessionId:** long
* **song:** string
* **status:** long
* **ts:** long
* **userAgent:** string
* **userId:** string

As we can see, we have the following columns that are of non-numeric values: artist, auth, firstName, gender, lastName, level, location, method, page, song, userAgent, userId. In the interest of computing resources, we will not be converting all non-numeric columns into numeric values. We will only be converting the ones that we deem necessary for the model.

## **Implementation**

Given all of the columns above, I believe that the following columns are necessary features for our model:

* Gender
* UserAgent
* Status
* Page

Once the columns were identified, we now have to make sure that they are al in the numeric datatype so that they could be put into the model that we choose. The Gender, UserAgent and page columns had to be converted into numeric values using a combination of String Indexing and One Hot encoding.

For the purposes of this project, I chose LogisticRegression as it is relatively simple to do and we were looking for a simple yes (1) or no (0) that the user might cancel their service.

Now that we have our columns in numeric format and our machine learning algorithm chosen, we were able to build our pipeline to perform all the necessary steps on our data.

At a high level, the pipeline performed the following:

1. Converted non-numeric columns with String Indexer
2. Took the String Indexed columns and one hot encoded them
3. Created Logistic model
4. Fit the data

## **Refinement**

A parameter grid was used to determine the best value of parameters. The various parameters that were chosen was 0.0 and 0.1. I did not choose a large amount of parameters to begin with to quickly determine if I was heading in the right direction or not. I fed the two parameters into the Cross Validator object along with the pipeline and the evaluator.

**Results**

## **Model Evaluation and Validation**

When the model was finished training, I used the Cross Validator Average Metrics method to get the accuracy of both the training and testing sets on our new model. I was amazed at the results! With only using the two parameters (0.0 and 0.1), I received a 99.99% accuracy for the training set and a 99 whereas the testing set received a 99.98% accuracy.

## **Justification**

At the beginning, I was quite shocked by the high level of accuracy that was received during the training of the model. By taking very user specific columns out of the dataset (timestamps, user ID, first name, last name), I believe that the model was able to generalize well and was able to predict with a high level of accuracy on new un-seen data.

**Conclusion**

## **Reflection**

Overall, I would say that this was a very exciting project to work on. This was a real world scenario for many online business that rely on subscriptions (both paid and un-paid). Utilizing the Spark technologies allowed me to get a better feel of Big Data Technologies and all of the potential that it has out in the real world.

## **Improvement**

As with anything in life, there is always room for improvement. One thing that could be looked more into, is to not only predict people that would cancel their subscription altogether, but to predict which paid users might downgrade to a free membership. This is also a concern to Sparkify, as this would lower their subscription fees that they would receive.