Predicting Customer Churn for a Music Subscription Service Using Spark

Joshua Lindsay

March 14, 2024

Abstract

A project analysing the actions of customers of a music subscription service and using machine learning techniques to try and predict if a customer will churn or not. Although only a small subset of the total data will be analysed to allow it to be done on local mode on a Jupyter Notebook Workspace, pySpark will be used in this project as it's context lends itself to working with big data. the machine learning techniques will be compare using chosen metric and the best one chosen and applied to a validation set.

1 Introduction

1.1 Project Overview

In a world where data is more omnipresent than ever, it is getting harder and harder to be able to analyse, model and manipulate the data a business collects from its customers. As businesses can now record and analyse every single customer interaction, they acquire larger and larger datasets which single computers with normal hardware struggle to analyse. This "big data" requires different approaches to be able to accurately model for different purposes, one of which is spark. Spark uses cluster computing to try and overcome this difficulty where multiple computers work in tandem and splits the data and tasks between them to bypass any bottlenecks which normal computing creates and processes the data more quickly.

Spark requires a slightly different form of programming than normal python due to it's purpose. It can also be used via 'local mode' where a single computer is used but the spark language is required. We can demonstrate this via a fake company and it's data called 'Sparkify'. This is a music streaming company which records every single customer interaction so would normally be an example of a business or area where you may encounter 'big data'. However as we want to be able to use local mode to better demonstrate the use of the spark coding language via a Jupyter notebook only a smaller subset of the data is used in this project. However the spark coding language is the same.

The use of spark means we can analyse a number of questions a business may have on it's customers from their behaviours to their potential future actions. This leads to the problem statement which a business may want to analyse that we are going to explore today.

How can we use Spark's Machine Learning techniques to be able to predict which of our customers may churn in the future?
This would provide great value to a business as they will be able to send offers to try and retain these potential churners and therfore increase their revenue.

So how can we attempt to do this? We will devise a new binary variable to keep track of the customers who have churned or cancelled their subscriptions and comparing the characteristics of both groups will allow us to identify potential variables which may signify a customer could potentially churn. Using these characteristics we can then compare a range of different machine learning techniques to try and predict if a customer will churn in the future using a "test" subset of the data. A metric we can then use to compare the three models and choose the most appropriate is the F-score which measures the positive prediction ability of the model in question. We are using F score as what we are predicting (customers who churn) is a relatively small proportion of the dataset in question so accuracy would

not be an appropriate measure as simply predicting all customers will not churn would lead to a high accuracy score, but provide no value to the business.

2 Data Analysis

2.1 Initial Analysis

Firstly we shall explore the subset of Sparkify's data we have been given and try and gain insight into what is relevant for answering the problem statement we have set ourselves. As previously stated, each record within the datset describes a customer interaction with the sparkify app or website. There are a number of variables used to describe the interactions, including:

- **artist / song:** details of the song the user is listening to if applicable.
- **auth:** checks if the user is logged in or not.
- **firstname / lastname / gender / location :** user details.
- **userId:** unique identifier of each customer the business interacts with.
- **sessionId / iteminSession:** describes the number of sessions a customer has with the app and how many interactions or actions they make within each session.
- **level:** checks if a customer is on a free or paid subscription.
- **page:** Identifies the page a customer is on on the app or website i.e new song, cancel, home.

There are 286500 interactions in total within the dataset. However there will be multiple interactions recorded for the same customer. If we try to count the distinct users within the dataset we see there are only 226 users within the dataset which shows that certain users may have a lot of interactions with the company - however this makes sense as each individual song a customer listens to will count as a separate interaction.

2.2 Adding Churn

Looking at the variables described in the previous section we can see there is none which directly relate to a customer churning, something which we are looking at in our problem statement. We will need to create and identifier for both when customers churn and also a marker for customers who will later churn. This marker will later be used as the target variable for our classification models to try and predict. Correctly predicting these customers mean the company is able to try and retain these customers via offers and deals and therefore increase revenue.

To create an identifier for when churn occurs we will use the **page** variable. One such option for pages is the **cancellation confirmation** page which a customer will access when they want to cancel their account, be that free or paid. A new binary variable **churn** will be created which is 1 when a customer accesses this page and 0 when not.

The next stage is then marking the customers who will churn using this variable. A variable called **will cancel** can be created which is again binary and is 1 if the userId of the interaction in question will churn in the future and 0 if not. This will be our target variable.

2.3 Comparing Churned and Non-Churned Customers

Before we use modelling techniques to predict churn, we can use exploratory analysis to compare the two groups (customers who will churn in the future and those who won't). This will lead to multiple benefits such as as being able to identify any issues with the data before modelling begins as well as helping us decide which variables to include.

Firstly we start with looking at the number of customers who churned compared to the number who haven't and both groups total number of interactions. Of the 226 total customers, 52 of them went to the cancel confirmation page and therefore are recorded to have churned. This means there are 174 who have not. Looking at the number of interactions which these groups gathered we can

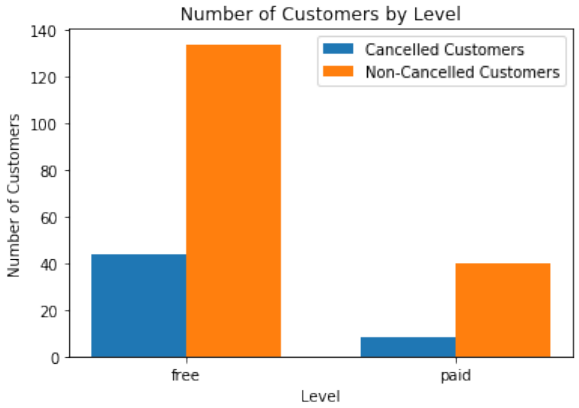

Figure 1: A graph comparing the split between paid and free subscriptions for customers who churned and those who didn't.

see those who churned gathered 44,864 total interactions while those who did not gathered 241,636. This shows that those who did not churn gathered over five times as many interactions as those who churned with only just over 3 times as many individual customers. This makes sense as those who churned would spend less time on the app and therefore gather less total interactions. This makes the problem statement even more important as even before they churn we are not maximising the value of these customers.

Another aspect of explanatory analysis which we want to look at are variables where these two groups differ as this would indicate that these variables could be used as predictors in our future models. We have already noticed that the value churning customers provide to the business is less than that of those who do not. We can further assess if this is the case by looking at what subscriptions both of these groups possess. Figure 1 shows this split. It shows that whilst for both groups the majority of customers are on the free subscription this split is even more apparent for the churning customers where there are over 5 times as many free customers than paying ones. The same statistic for non-churning customers is just over three times. This shows the level of subscription the customer is on could be a predictor for our models. We can also look at the number of songs which customers who churn listen to and how this compares to those who don't. As those who churn have less interactions we would expect them to listen to less songs on average. The average number of songs which users who churned listened to is 863 approximately. The average for those who did not churn is 1388. It is clear to see that number of songs has a relationship with customers churning. One way this can be included in the model is to look at the "sessionId" and the "iteminsession" variables which take into account the extent of the customer's usage of the app.

Another area where we may want to look at is what pages customers may be more likely to frequent before they cancel their account to try and stop them before they do so. If we look at the last twenty pages which customers who cancel their account visit and compare these to the overall distribution of pages which customers visit we can see if there are any clear differences. Figure 2 and 3 show these distributions of pages. Both clearly show that for all customers, next song is the most common page. However Figure 2 does show there a number of pages which customers who are going to cancel are more likely to frequent. As well as the self explanatory pages such as "cancel" and "cancel confirmation" other pages which are more common for soon to be churning customers include "thumbs down", "settings" and "downgrade". If sparkify was to send an offer to customers when they access these pages they may be more likely to stay. This shows that the page a customer is on should be included within our model as a predictor.

3 Methods

3.1 Preprocessing

To answer our problem statement we will be using ML techniques to predict a binary variable of "will cancel". There are a number of methods which spark allows to do so, but all require the inputted data

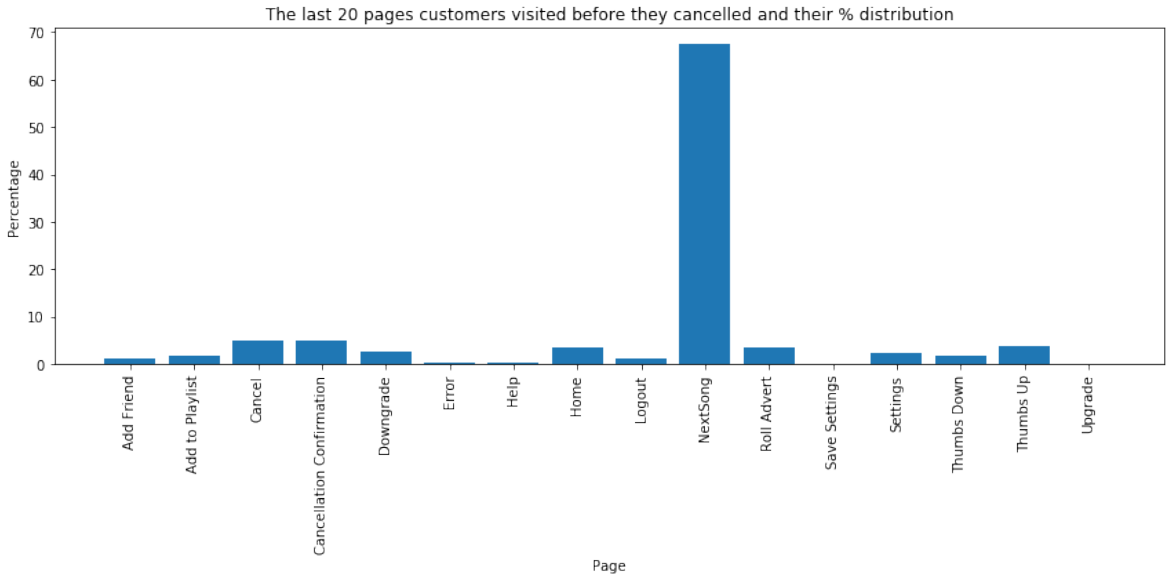

Figure 2: A graph showing the last twenty pages customers visited before they cancelled their account and their % distribution.

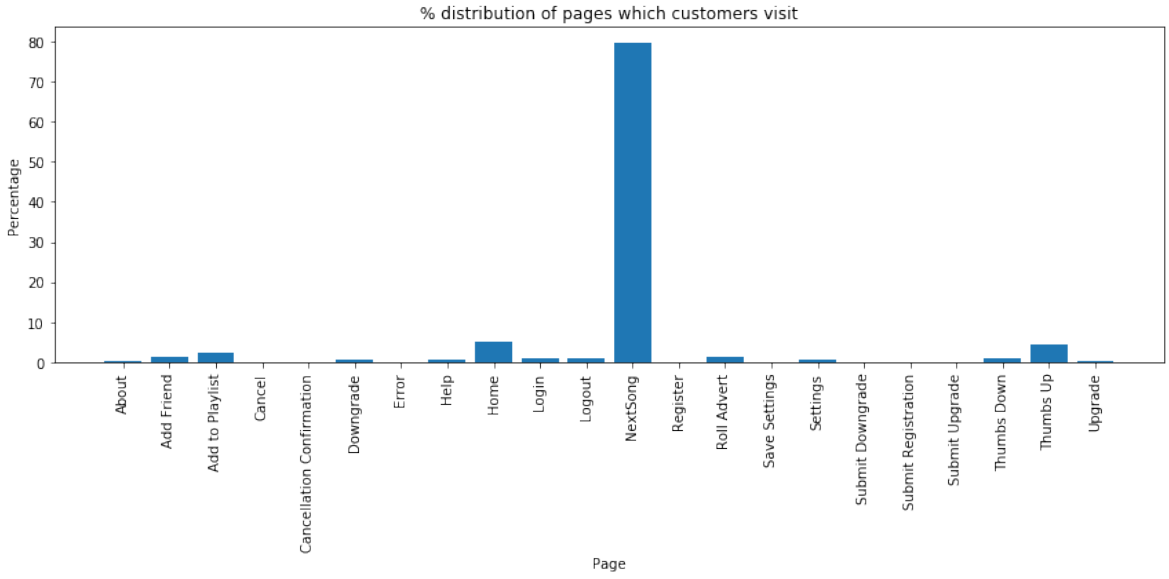

Figure 3: A graph showing the % distribution of pages which Sparkify customers visit.

to be of a certain format. They require all of the input variables to be of vector format and therefore to be numeric. In order to run these models when we choose our variables we will then have to convert them to this form.

3.1.1 Choosing Variables

Firstly we need to select our predictor variables which we will want to use. From our explanatory analysis we encountered a number of variables which we want to include within our final models. We saw that "page" could have an impact as customers who are about to churn often interacted with different pages as those who didn't. We also wanted to include the "level"of subscription a customer was on for a similar reason. To include the number of interactions and number of songs a customer listens to (which we saw from explanatory analysis may have an impact) we are going to include the "sessionId " and the "itemInSession" variables. Finally we will include the "gender" variable to see if it has an impact.

3.1.2 Variable Format

So are all of our predictor variables numeric as we desire? Our session variables are so they can be left for now. Our gender and subscription variables are text but are binary with only two options allowed for both. Therefore we shall simply convert these to numeric binary to allow us to use these. 1 will indicate male and paid respectively and likewise 0 for female and free. This simple conversion allows them to be used.

However the page variable is a text categorical variable and therefore will have to manipulate it into a numeric format. Firstly we use the tokenizer function to split it into it's respective words. We then use count vectorizer and the inverse document frequency functions to try and find the importance of the word and the impact it would have in being able to distinguish between pages. This numeric value which we get from using the idf function is what we shall use to represent our page variable.

All that is left for us to do is to use the Vector Assembler function to combine these numeric variables as a vector which we can use as an input for our models. This vector will represent all of our predictor variables.

3.2 Methods and Refinement

The next stage is to model itself. Before we do that however we split our dataset into training sets to build our models on, test sets to compare the results and metrics of our models and a validation set which we will run our chosen model on to get final results. The split we shall use is 70% for the training set, 20% for the test set and 10% for the validation set.

In terms of methods there are four common supervised classification techniques which are used within spark. They are

- Logistic regression
- Decision Tree
- Random Forests
- Gradient-Boosted Trees

We shall attempt all of these and compare our chosen metric for each (F-Score) to choose a final model.

3.2.1 Logistic Regression

Logistic Regression uses the logistic or sigmoid function to model the relationship between features and the target variable being in one of two classes so is therefore used in binary categorical classification. Maps values to a probability between 0 and 1. The equations to produce these probabilities are below

$$p = \frac{1}{1 + e^{-t}} \tag{1}$$

where

$$t = \beta_0 x_0 + \beta_1 x_1 + ... + \beta_t x_t \tag{2}$$

Where x_t are our explanatory variables and β_t their respective parameters. We can then apply a threshold value of 0.5 and using the probability produced use this to classify results as either a 1 for will cancel their account or 0 if not.

3.2.2 Decision Tree

Decision Trees are also used in classification problems. As their name suggests, they have a tree like structure with each branch being a decision rule which you follow down to the leaf or predicted outcome. It uses Gini impurity to select the features used to classify the outcome and then splitting the data into subsets based on these features recursively until a stopping criterion is met.

This could be all instances in the subset belonging in the same outcome or a maximum depth of tree is reached. One area which you can refine when it comes to decision trees is the maxDepth hyperparameter. This is the maximum number of levels and therefore the complexity of the tree. A larger value means more splits and therefore may be better at classifying more complex data. However this may mean it is prone to overfitting where it performs extremely well on the training data but this means it struggles with new test data. We will compare three different vales of maxDepth and the F score they produce to decide what we shall use for this parameter; 10,20, 30. 30 is the maximum which sparks decision tree package allows. The table shows us that using a maxDepth of 20 provides

MaxDepth	F-Score
10	0.775
20	0.809
30	0.807

the greatest F-score and will therefore be used when doing model comparison in the next step.

3.2.3 Random Forests

Random Forests are an ensemble technique which uses multiple decision trees from the previous section to make predictions. In the collection of decision trees, every tree is trained on a different subset of data and features using bootstrapping. Each tree is then developed as normal using the process described above and used to make predictions on the unseen data. Therefore at the end of this stage there will be a number of decision tress which will have different predictions. The final predictions which the forest as a whole will make will be done via majority voting, with the most common predicted class selected.

The advantages which this possesses over using individual decision trees is that random forests are less prone to overfitting, able to fit more complex data better and can provide importance of features.

Similarly to individual decision trees, random forests have a range of hyperparameters which can be refined for better metrics. One of these is the number of trees to include within the forest. More trees means errors made by individual trees have less impact and therefore provides more stable results. Unlike individual trees there is no real danger of overfitting by using more trees, however there is the drawback of using more computational power and time to do so. The error produced by random forests will often decrease to a minimum and then plateau and it is best practice to select a size of forest where this plateau starts. As before we will test using three different sizes of forest and try to select a value which is at the start of plateau for error; 10, 500, 100. The table below shows the impact these three sizes of forest has on the F-Score. As shown in this example the plateau is reached early

numTrees	F-Score
10	0.775
50	0.775
100	0.775

when the forest is only 10 trees large and therefore we shall use this size of forest in the next section.

3.2.4 Gradient Boosting Trees (GBTs)

GBTs are another form of ensemble classification, similar to random forests. However it's method for making predictions is different. It will start with a single classifier such as a decision tree which will make mistakes when making predictions. Future classifiers are then built to try and correct the errors produced by the previous ones. Each tree is built to minimise the difference between the target values and the predictions which the classifiers have made so far. This is done until a stopping criteria such as the number of trees has been met and then predictions are made using weighted voting. They provide similar advantages over singular classifiers that random forests do.

We can refine our hyperparameters for this classification method as well. As in the decision tree, the maxDepth parameter can be refined to find the maximum depth of each individual tree used in the method. We will compare the F-Score produced by the same three values of this hyperparameter as in the singular decision tree; 10,20,30. The table below shows this comparison. Using F score we

MaxDepth	F-Score
10	0.807
20	0.815
30	0.814

can see that the optimal maxDepth to use is 20 and therefore this is the form of GBT we will use in the next section in model comparison.

4 Results

4.1 Model Comparison

As stated in the previous section, we have identified four modelling techniques which we can use to try and predict customer churn and have used them for this purpose. For each we have refined one of their hyperparamters to try and find the best possible form of this model. Now we have got four optimal models we can then compare them to find the best one which we will then apply to the validation set for our final results.

Figure 4 shows these four F scores obtained by our different methods. It shows that all four obtained relatively high F scores - all scoring between 0.75 to 0.85. Our logistic regression and random forest have slightly lower scores, implying that decision trees and GBT are the better method for this particular issue. In terms of an overall best model to use, the GBT model is the best for the chosen metric and is therefore the model to apply to the validation set.

4.2 Final Results on Subset

Now we have chosen our final model we can apply it to our validation set and look at some final metrics and how it performs.

The main metric we have been using so far in this project has been F-score. It is given by the equation

$$F = \frac{2}{recall^{-1} + precision^{-1}} \tag{3}$$

Where recall is the number of classified positives divided by all those which should have been classified as positive and precision the number which should have been classified as positive divided by the number which were. It ranges from 0 to 1 where 1 indicates perfect precision and recall. The reason we have used this throughout is due to it being a stronger metric than accuracy for unbalanced datasets where there is significantly less positive results than negative ones. The F score obtained by our final model on the validation set is **0.816** which is relatively strong.

There a number of other metrics we can look at for our final model. Although not as strong, accuracy can also be looked at. It is measured by the number of correct classifications divided by all of the total classifications. For our model the accuracy is measured as 0.853 which again shows that our model is relatively strong at predicting. This shows just some of the qualities which our relatively rudimentary model possesses.

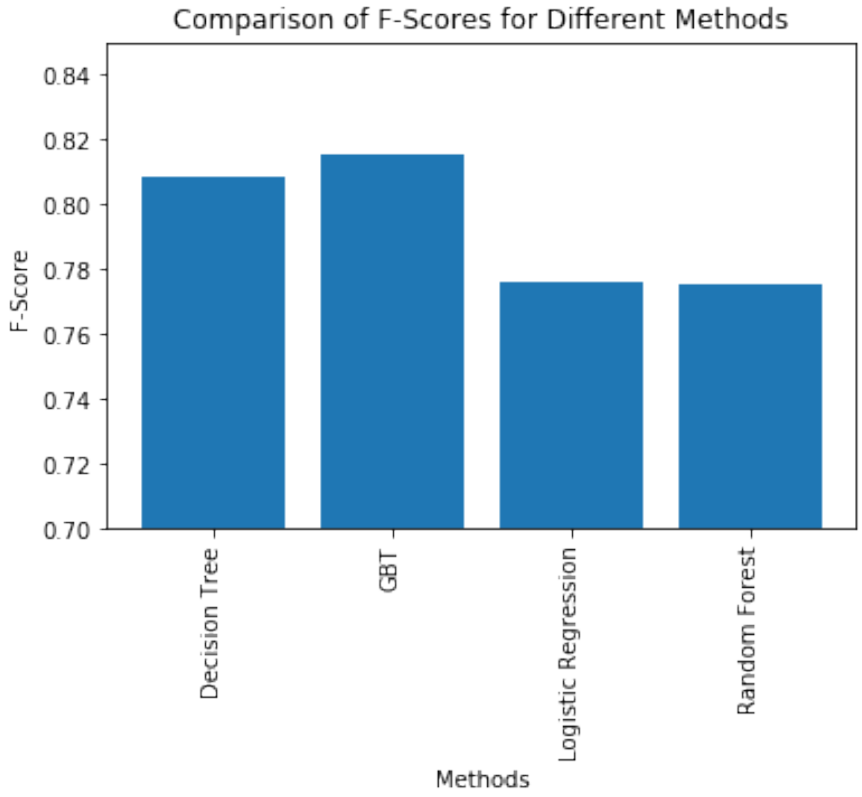

Figure 4: A graph comparing the F score obtained by four different classification techniques used to predict customer churn for Sparkify.

5 Conclusions

5.1 Reflections

In an age where the amount of data is getting greater and greater we have to be able to use different techniques and capabilities to derive insights from it. One such way to do this is to use cluster computing such as via spark which reduces the time by using multiple computers to solves problems in regards to large datasets.

One such real life business problem that may require the use of spark is to identify customers who may churn and predicting this. By doing so, they can send offers to the customer to try and retain them therefore increasing their revenue. Often a business will have to look at customer interactions in order to do this and as their can often be millions of these, this will require the use of spark or another form of cluster computing.

We have shown how this problem could be addressed, albeit via local mode on a standalone computer but this can be easily extended using IBM or AWS. We have shown that spark offers a number of different techniques which can be used to model big data for classification such as decision trees, random forests, logistic regression and gradient boosted trees. Using data from a fake music streaming company - sparkify - to try and compare these techniques and found that gradient boosted trees were the best performing in our chosen metric.

Now we have seen the techniques which can be used to solve real life issues with big data, we can expand their use and use them to solve different questions and make even more insights in our world of big data.

5.2 Improvements

This project is merely intended to show one way which we can solve issues with big data and how to model it. However there are certain areas where improvements could have been made if working with

real life data.

- As stated before we have been using local mode on a standalone computer. Whilst this has shown the techniques which are used and followed the spark coding language, it has had to use a subset of the available data to overcome this. To better illustrate why spark has to be used in certain situations a larger dataset could have been used and the use of cluster computing to analyse this. A number of services offer this capability such as IBM and AWS and would have shown exactly how big data should be modelled.

- The choosing of predictors to include in the model was rudimentary looking at what seemed to differ between customers who churned and those who didn't. A more comprehensive approach could have been taken. This could have been comparing metrics before and after each variables had been added in regards to a threshold value in a forward selection like method. This would have taken into account both predictive power whilst still ensuring the model is as simple as possible.

- Likewise the hyperparameters chosen for each model where either default or in some certain instances of refinement, done so in a rather basic fashion. One way in which we could have chosen the best hyperparamters for each of the methods in turn could have been through the use of GridSearchCV. However whilst this may have led to stronger models, it would have required significant computational time and effort.