

**SPARKIFY**

**CHURN PREDICTION WITH PYSPARK**

**UDACITY DATA SCIENCE NANODEGREE – CAPSTONE PROJECT**

Ishank Gupta

|  |  |
| --- | --- |
| **TABLE OF CONTENT** | |
| **Table of Content** | …2 |
| **Problem Overview** | … |
| **Problem Statement** | … |
| **Data Analysis** | … |
| **Feature Engineering** | … |
| **Modelling** | … |
| **Result & Evaluation** | … |
| **Conclusion** | … |

**Problem Overview**

Sparkify is a music service platform like Spotify (Not real) and has a subscription-based model. Where the users have option to either use it as paid (nominal fees) or free subscription (limited features). Users have the option to upgrade/downgrade or even cancel their plan altogethor (left the app). Besides users have various features like adding songs to their playlist, researching songs from their favorrite artists, sharing songs to friends, upvoting/downvotings songs based on their preference and much more.

Also, app has advanced functionality of recommending songs based on user activity (only on paid model). Each time the user interacts with the App, the activity gets recorded with time.

**Problem Statement**

The project's objective is to analyze customer churn and build a model for Sparkify to identify which customers will potentially churn. Using this analysis Sparkify can take proactive action and they can send various incentives to the customers to try to retain them as their customers. Some examples of incentives are:

1. 75% off for the next 3 months
2. Enroll in the family plan pay for 2 get 2 subscriptions free

A Major part of the project is to define what is churn for our project.  
Ideally, churn is defined as those customers that have left your service. So we can expand this definition further by identifying those customers who downgraded their services that is moved from paid to a free model as churn also. But to simplify the analysis for this project we will only define churn as those customers who have permanently left your service (i.e canceled the plan – submit the canceled button).

For this pupose we have received a dataset of 12 gb.

As performing analysis on 12 gb data is computationally expensive so we have divided our project in 2 phases:

1. Phase-1: Analyzing the data, performing feature engineering, and running classifincation on mini dataset (128 mb).
2. Phase -2: Based on our objseration of result on mini dataset using AWS EMR (elastic Map Reduce) to run the models on full 12 gb data.

**Data Analysis**

Schema of Mini-Dataset

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)

It has 286500 rows and 18 columns.

To explore the data more (we created this data summary for our analysis):

Feature #Nulls %Nulls description

0 artist 58392 0.796188 artist name

1 auth 0 1.000000 user's authentication

2 firstName 8346 0.970869 user's first name

3 gender 8346 0.970869 gender f or m

4 itemInSession 0 1.000000 item count

5 lastName 8346 0.970869 user last name

6 length 58392 0.796188 length of the song in seconds

7 level 0 1.000000 plan: free or paid

8 location 8346 0.970869 user's location

9 method 0 1.000000 request method

10 page 0 1.000000 page : various options

11 registration 8346 0.970869 resgistration timestamp

12 sessionId 0 1.000000 session id

13 song 58392 0.796188 song name

14 status 0 1.000000 status

15 ts 0 1.000000 timestamp of event

16 userAgent 8346 0.970869 user's browser

17 userId 0 1.000000 user unique Id

From this we can observe that many columns have around 79.6188% of values Null, after analyzing the cause we found that this is because the there are many users whose userrId and other personal details is empty maybe because they haven’t signed into the app and are using the services as guest.

So, for that purpose we dropped all rows which has userId set as “” from our data.

Let’s understand what each column refers to:

1. Auth

+----------+

| auth|

+----------+

|Logged Out|

| Cancelled|

| Guest|

| Logged In|

+----------+

1. Gender (proportion of female and male users – before cleaning data)

+------+------+

|gender| count|

+------+------+

| F|154578|

| null| 8346|

| M|123576|

+------+------+

1. Level

+-----+

|level|

+-----+

| free|

| paid|

+-----+

1. Location (these are actual address later we transformed this to extract state from this data)
2. Page (there are various page options, and each time a user interacts with the page the activity gets recorded in system with actual timestamp)
3. +--------------------+
4. | page|
5. +--------------------+
6. | Cancel|
7. | Submit Downgrade|
8. | Thumbs Down|
9. | Home|
10. | Downgrade|
11. | Roll Advert|
12. | Logout|
13. | Save Settings|
14. |Cancellation Conf...|
15. | About|
16. | Settings|
17. | Add to Playlist|
18. | Add Friend|
19. | NextSong|
20. | Thumbs Up|
21. | Help|
22. | Upgrade|
23. | Error|
24. | Submit Upgrade|
25. +--------------------

For our analysis pupose we are transformed ts and registration date in date format (MM-DD-YYYY).

Also, after defining churn as those customers who left the system that is have sumbitted a cancellation confirmation, then in our data we have

|  |  |
| --- | --- |
| Unique Customerts | 225 |
| Customers Churned | 52 |
| Customers Active | 173 |

**Analysis-1**

Using the ts column we analyzed the user pattern with the app.

A blue dotted line on a white background

Description automatically generated

From the plot we can see that users are most active from 3pm to around 6pm. There can be various reasons for this for example that is the usual time people leave from the office.

Based on this analysis the Sparkify marketing team can utilize this timeperiod of 3-4 hours to have targeted advertisement to customers to convinve them to switch over to the paid version (for example to listen to the song watch 30 second add or migrate to paid service)

Analysis -2

A blue and white rectangles

Description automatically generated

**Analysis-3**

**A graph of different colored bars

Description automatically generated**

**Analysis-4**

A graph of blue and orange bars

Description automatically generated

**Analysis -5**

A screen shot of a graph

Description automatically generated

**Feature Engineering**

For the classification, we want to identify or develop new features from raw data that might be crucial to developing the classification model for identifying customer churn.

Based on my initial data exploration, data understanding and intuition, I was able to get an idea regarding which all features might be relevant to be included in the model (too many features if used can lead to overfit as well).  
In this project, we created a pipeline for how we are transforming data by use of 3 user-defined functions that are:

1. Data cleaning: Dropping nulls/duplicates etc.
2. Data preparation: Data Transformations such as getting time filed in the correct date format.
3. Feature building: Transforming raw data to get the key features required by the model, such as extracting the state from the location.

The Key features identified and used in the model are:

1. Days\_registered: No. of days since users have been registered on the app
2. Avg\_songs: No. of songs users listen to in 1 session.
3. Last\_level: last level registered is it paid or free?
4. Thumbs-up count: No. of thumbs up given.
5. Thumbs Down count Number of thumbs down given.
6. Num\_friends: Number of friends of the user
7. Avg\_roll\_advertise ment sent to the users.
8. Help\_visit: Number of times users had to visit the help page.
9. Error\_visit: Number of times users had to visit the error page.
10. Total\_playlist: Total number of songs
11. Gender: M/F

I was also planning to include other features such as the last location, the number of days the users have consecutively used the app, and the number of songs shared among other features. (Planning to expand this project further).

To identify the key features relevant to the model performed the feature selection using SelectKBest and f\_regression. Based on these the top 5 features relevant to the model are:

Selected Features:

Index(['days\_registered', 'thumbs\_up\_count', 'num\_friends', 'avg\_roll\_advert',

'total\_playlist'],

dtype='object')

A screenshot of a graph

Description automatically generated

**Modeling**

The first analysis is done on the mini-dataset 128 mb and then the process is replicated over 128 gb, this is done by making a suitable assumption that the mini-dataset is created by properly sampling data from the main dataset.

Using the feature engineering pipelines made (which take the raw input data) and clean, preprocess, transform, and return a dataset with relevant features.   
After getting the data, we first split the dataset into Training (80%) and testing (20%) and then create a data pipeline for the model:

1. Vector assembler: VectorAssembler is a transformer that combines a given list of columns into a single vector column. It is useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like logistic regression and decision trees.
2. Normalizer: the process of translating data into the range [0, 1] (or any other range) or simply transforming data onto the unit sphere.

(It might have been advantageous to use a scaler like min-max scaler – can consider in future)

3 main classification Algorithms considered are:

1. Random Forest
2. Logistic Regression
3. Gradient Boosted trees

The main reason behind focusing on tree-based algorithms was that there is a class imbalance in the data (in the mini dataset only the churned\_customers is 52 and unchurned is 173 io. A Ratio of almost 1:3) Also boosting algorithms works well because) are ideal for imbalanced datasets because higher weight is given to the minority class at each successive iteration. during each interaction in training the weights of misclassified classes are adjusted.

Other methods to explore to reduce the effect of class imbalance that can be implemented in future are:

1. Stratified Train-Test Split
2. Resampling data (Oversampling / undersampling)

Metrics to consider for evaluation:

1. Accuracy – This might not be good enough, because of class imbalance the model will think that by selecting the Major class it is performing correctly. High Accuracy can be misleading.
2. Precision: It basically gives us an idea about how many churned customers were correctly classified.
3. Recall: It gives us an idea regarding the total number of churn customers and how many were classified correctly.
4. F-1 Score: It is the harmonic mean of precision and recall. For imbalance data problems using F-1 score is a better metric as it can be used to find an equal balance btw prediction and recall.

F1=2∗precision∗recall/(precision+recall)

All the feature engineering steps, along with pipeline creation was performed on both mini-dataset and full-sized dataset. Cross-validation was used to identify the best model and the results of both are summarized below.

**Mini-Dataset**

(Train – test split: Training data (169 customers), Testing data (56 customers))

(Results for testing data)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classification Method | Accuracy | Precision | Recall | F-1 Score |
| Random Forest | 0.85714 | 0.83697 | 0.85714 | 0.83965 |
| Logistics Regression | 0.76785 | 0.8227 | 0.76785 | 0.68281 |

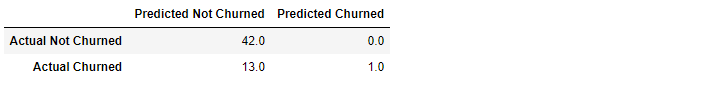
Confusion matrix:

Random forest

A white rectangular object with black text

Description automatically generated

Logistic Regression



**Full-sized dataset(12 Gb):**

Used AWS EMR for this big dataset. After creating aws account setting up MFA authorization has to create an IAM user otherwise it was creating an issue.  
Based on account I checked the limits of the account and as per that I only could use the cluster with instance type of m5x, m52x. Created an EMR cluster with following specifications:

**A screenshot of a computer

Description automatically generated**

With Instance type of m5. large with 1 primary and 6 core nodes.

The same process as that from the mini dataset is used here with an additional gbt classifier and results are summarized below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classification Methods | Accuracy | Precision | Recall | F-1 Score |
| Random Forest | 0.78808 | 0.6703 | 0.788625 | 0.72338 |
| Logistic Regression | 0.8185 | 0.75264 | 0.8185 | 0.7407 |
| GBT | 0.8134 | 0.7267 | 0.8127 | 0.7416 |

Furthermore, cross-validation with 3 folds was used to determine the best model for both random forest and logistic regression. The results of the test data are summarized below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classification Method | Accuracy | Precision | Recall | F-1 Score |
| Random Forest | 0.81851 | 0.6702 | 0.8185 | 0.73707 |
| Logistics Regression | 0.81706 | 0.7555 | 0.817424 | 0.74804 |

Did not hyper-tune the GBT model because of computation and time constraint as running the code on AWS is expensive.

Thus, based on the result we can see that if we hypertuned the GBT model it should give us best result.

**CONCLUSION**

As per the problem statement classification models have been developed which will help the Sparkify team identify the customers who might churn in the future so that they can take proactive action by implementing various methods to try to retain the customers.   
To complete the projects following steps were performed:

1. Data cleaning – Dropping nulls, duplicates transforming data to correct format etc.
2. Data exploration – Exploring data to understand each feature and extract crucial information regarding customer behavior.
3. Feature Engineering – Toughest step of the project as identifying key features which are relevant to the model was crucial. There is a lot of scope for improvement in this step which will lead to the development of a better model.
4. Defining Key Metrics- Identifying the presence of an imbalance in data, choosing correct metrics like F-1 Score, and justifying the selection of boosting and tree-based methods.
5. Hypertuning the model and finding the best model using the cross-validation method.

**Future Improvements**

1. Exploring ways to handle a class imbalance in better manner with the use of stratified train-test split, applying under sampling/ over sampling such as SMOTE.
2. Building better relevant features such as days of continuous usage as it is the most important parameter identifying how actively customer uses the app. Cpaturing of other demographic data like age should help.
3. Hypertuning of parameters for GBT classifier.
4. Can also use an ensamble model.