Using Spark to Predict Sparkify

Churn Rate

**Project Definition**

**Project Overview**

Sparkify is a fictional music streaming app created by Udacity for this project. The fictional users have 2 service plans to choose:

* + ‘Free - tier’ asks no fee from users but place advertisements between songs;
  + ‘Premium subscription plan’ where the stream musics are free but users pay a monthly flat rate.

Users can shift plan between ‘Free - tier’ and ‘Premium’ freely. They can even ‘Cancel’(Churn) the service if they don’t love it.

In this project, I use Spark MLlib to build machine learning models with large datasets to predict which users are at risk to churn (cancel) their service before they leave.

**Problem Statement**

The goal is to predict the churn rate with the large datasets shared by Udacity. The tasks involved are the following:

* 1. Load large datasets into Spark and manipulate them using Spark SQL and Spark Data frames.
  2. Build out the features with the data
  3. Use the machine learning APIs within Spark ML to build and tune models that can determine if the feature determines “Churn”

The final application is expected to be useful for predicting the user’s “Churn”.

**Metrics**

Accuracy

is a common metric for binary classifiers; it takes into account both true positives and true negatives with equal weight

Accuracy = (true positives + true negatives)/ dataset size

Our problem is to predict which users that are likely to churn which not, so essentially this is binary classification.This metric was used when evaluating the binary classification.

f1 score

Since the churned users are a fairly small subset, accuracy is not enough to explain the model’s robustness. I used F1 score as the metric to evaluate the model.

f1 = 2\*precision\*recall / (precision + recall)

F1 score is a better measure for the imbalance class distribution in our project.

**Analysis**

**Data exploration**

The full dataset is 12GB, of which I analyzed a tiny subset (128MB).

The mini subset (I used) has 286500 rows and 18 columns which has the information of users(gender, name, etc.) and API events(login, playing next song, etc.) The number of customers that churn is 52 in this dataset and total number of customers is 225, about 23 % of total users cancel the service!

Some of the “userId" data are either missing or empty, which means these data points must be discarded during preprocessing.

**Exploratory Visualization**

The tables below shows how the ‘Gender’, ’Premium Usage’, ‘Usage Time’ are distributed between the different classes (Churn and Stay users).

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

|Churn|sex|count|

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

| 0 | 1| 89|

| 0 | 0| 84|

| 1 | 1| 32|

| 1 | 0| 20|

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

Fig 1. Sex distribution(sex = 1: Male, Sex =0: F, Churn = 1: churn user)

It can be seen that: about 50% more male users cancel the service. While in the ‘Stay’ set the number of females and males are very closed.

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

|Churn| avg(chgrd)|

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

| 0. |1098.0173410404625|

| 1 | 624.5384615384615|

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

Fig 2. Premium Usage distribution ( chgrd: numbers of paid API events )

It can be seen that: average premium utilization of ‘Stay’ users is almost double the premium utilization of ‘Churn’ users.

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

|Churn| avg(time\_gap)|

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

| 0 |4060011.456647399|

| 1 |2031665.576923077|

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

Fig 3. Utilization time distribution(time\_gap: max(ts) - min (ts) by user)

It can be seen that: average utilization time of ‘Stay’ users is almost double the utilization time of ‘Churn’ users. (Here, the ‘time\_gap’ keeps date time format. )

**Methodology**

**Data Preprocessing**

The preprocessing does in the Jupiter notebook(‘Sparkify.ipynb’ section ‘Load and Clean Dataset’ )consists of the following steps:

* 1. Transform ‘gender’ data into numeric data ‘sex’ (0 or 1)
  2. Flag Churn classes as numeric data ‘Churn’ (0 or 1)
  3. Compute premium utilization, utilization time as ‘chgrd' ,'time\_gap'
  4. The data are divided in a train set (rest) and validation set

There are also some preprocessing steps which are done as the data get loaded into model before training:

* 1. Three columns (‘sex’, ‘chgrd' , 'time\_gap') are converted to resemble vector
  2. The vector get transformed to standardized vector
  3. Transform the 3-dim standardized vector into 2-dim
  4. Each of the classes gets a label, which is numeric data ( 0 or 1)

**Implementation**

The implementation process can be split into 2 main stages:

* 1. The model training stage
  2. The application prediction stage

During the first stage, the model was trained on the preprocessed training data. This was done in the Jupiter notebook(‘Sparkify.ipynb’ section ‘Modeling’ ).

**Refinement**

The dataset trained with logistic model achieved a good accuracy, around 84.7%. However since the churned users are a fairly small subset, the accuracy result cannot explain the model’s validation explicitly. In addition I took F1 score, even the average F1 was 0.85 to both classes, the F1 score and precision are 0.6,0.6 correspondingly to the ‘Churn’ set. I cannot say it’s a perfect result as expected, but it’s much better than the initial result.

To get the initial result, the accuracy is around 76%, and F1 from ‘Churn’ group is 0.33. This was improved upon by with techniques like PCA to reduced the correlated noise and standard scale the vector to make the model more scaleable.

**Results**

**Model Evaluation and Validation**

During development, a validation set was used to evaluate the model.

The final model architecture and hyper parameters were chosen because they performed the best among the tried combinations.

For a complete description of the final model and the training process : the training runs for 10 max iterations, and the regParameter is 0.0.

**Justification**

Using the final architecture, with CPA and standard scaled vector improve the model’s validation. CPA transformation helps to reduced the correlated noise and make low- dimension features independent. In the other hand, standard scaling vector is also applied. This project is against a mass dataset, standard scaling the feature is an effective method to make the model scalable.

**Conclusion**

**Reflection**

The process used for this project can be summarized using the following steps:

* 1. Load a mass dataset into spark and familiarize the data
  2. Build out the features, clean and transform the data
  3. Train and tune the model

I found steps 2 and 3 the most difficult, as I had to familiarize myself with the [Python PyData Stack](https://pydata.org/downloads.html) to manipulate spark data frame, used Spark MLlib to build machine learning models and train model, both of which were technologies that I was not familiar with before the project.

As for the most interesting aspects of the project, I’m very glad that I did a project on using machine learning technology, as I’m sure it’ll be useful for later projects/experiments. I’m also happy about getting to use spark, as I believe spark ecosystem will be more and more popular to be used to manipulate big dataset in the data science world.

**Improvement**

The evaluation of the model is not perfect at all. The model could be improved significantly by using more other feature like  daily and monthly averages of page events; and other different classification methods and algorithms like Random Forest Classification and Gradient Boosting Classifier.