CS 724 High Performance Computing and Big Data

Course Project Design

Herambeshwar Pendyala | 01130541

Title: Customer Churn Prediction using PySpark

1. Objective

The primary objective of the project is to build a scalable model to predict the customer churn for a music streaming service, given features such as user activity on the website, events, user history and other user related data.

2. Description

Sparkify is a fictitious music streaming service, created by Udacity to resemble the real-world datasets generated by companies such as Spotify or Pandora. Millions of users play their favorite songs through music streaming services daily, either through a free tier plan that plays advertisements, or by using a premium subscription model, which offers additional functionalities and is typically ad-free. Users can upgrade or downgrade their subscription plan any time, but also cancel it altogether, so it is very important to make sure they like the service.

Every time a user interacts with a music streaming app, whether playing songs, adding them to playlists, rating them with a thumbs up or down, adding a friend, logging in or out, changing settings, etc., data is generated. User activity logs contain key insights for helping the businesses understand whether their users are happy with the service.

In order to stay on track with its financial goals, it is key for a music streaming business to identify users who are likely to churn, i.e. users who are at risk of downgrading from premium to free tier or cancelling the service. If a music streaming business accurately identifies such users in advance, they can offer them discounts or other similar incentives and save millions in revenues. It is a well-known fact that it is more expensive to acquire a new customer than it is to retain an existing one.

3. Dataset

Sparkify, a fictional Music Streaming Service similar to Spotify, by Udacity, http://udacity-dsnd.s3.amazonaws.com/

The data given is that of user events. Every interaction of every user with the application is provided. This means every time a user goes to the Home page, listens to a song, thumbs up a song, etc. we have an event in the data corresponding to the same. Each record in our data represents an event by a user. Below are the columns in the 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)
```

4. Design

For model development process, most of the steps, such as data understanding, feature engineering and model selection, were performed on a representative sample (1/100) of the full dataset, using Spark in local mode. Models that performed well on the smaller sample were trained and tested also on the full dataset.

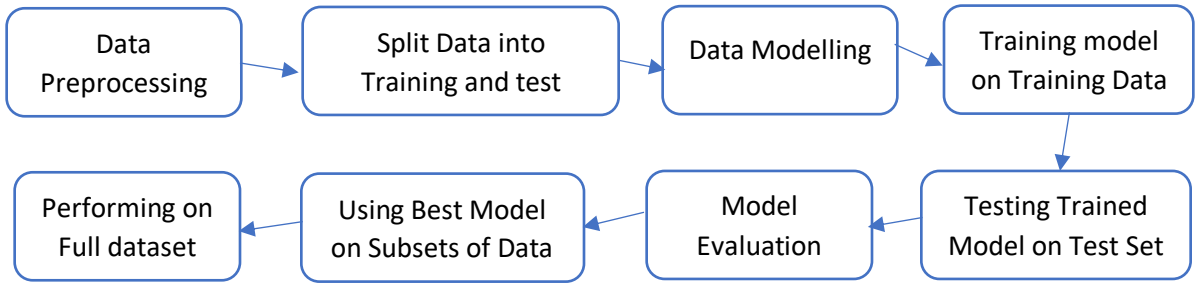

Fig 1. Flow chart of the implementation

4.1. Data Cleaning, Preprocessing, Exploratory Data Analysis

Using SparkSQL for data preprocessing as follows,

To transform the original dataset (one row per user log) to a dataset with user-level information or statistics (one row per user), obtained through mapping (e.g. user's gender, start/end of the observation period, etc.) or aggregation (e.g. song count, advertisement count, etc.)

As no label specifically as 'churn' was given to predict, all the features need to be engineered and used to identify churned users, e.g. aggregated statistic per unit of time, number of plan changes, songs played vs. total activity ratio, thumbs up vs. thumbs down ratio, activity trend, etc.

Defining and calculating the churn, we consider this as binary variable: 1 — users who cancelled their subscription, and 0 — users who kept the service throughout

Analyzing the correlation between engineered features

Performing exploratory data analysis, comparing the engineered statistics for users who stayed vs users who churned.

4.2. Model Building

As this problem is termed as a classification problem, the data is split into a training set and a test set. I am using Logistic regression and Random forest algorithms to build prediction models and see which algorithm performs better on the dataset.

Using PySpark for machine learning,

Logistic Regression

Mathematical expression of the algorithm**:**

For one example $x^{(i)}$:

$$z^{(i)} = w^T x^{(i)} + b$$
$$\widehat{y^{(i)}} = a^{(i)} = sigmoid\left(z^{(i)}\right)$$
$$\mathcal{L}\left(a^{(i)}, y^{(i)}\right) = -y^{(i)} \log\left(a^{(i)}\right) - \left(1 - y^{(i)}\right) \log\left(1 - a^{(i)}\right)$$

The cost is then computed by summing over all training examples:

$$J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(a^{(i)}, y^{(i)}\right)$$

To minimize the cost(*J*), I used gradient descent as an optimization method, the goal is to learn *w* and *b* by minimizing the cost function *J*. For a parameter θ, the update rule is $\theta = \theta - \alpha\, d\theta$, where α is the learning rate.

Using the above mathematical expressions, below is the pseudocode using RDD MapReduce to perform gradient descent.

```
For given number of iterations :
    Gradient = dataRDD.map(calculate cost).reduce(add)
    # update weights
   W = W - α * Gradient
```

After it learn the parameters, they are used to predict the labels for the dataset, by using below equation.

$$\hat{Y} = A = \sigma(w^T X + b)$$

Convert the entries of ***A*** into 0 (if activation <= 0.5) or 1 (if activation > 0.5), stores the predictions in a vector ***Y_prediction***.

Random Forest

I intend to implement this algorithm using RDD and Spark MLlib.

4.3. Model Evaluation

- Perform training on various samples of dataset and also the full dataset and plot ROC (Receiver Operating Characteristics) curve, calculate area under curve to get the best model.

- Also, from the optimum point in the curve, will plot the confusion matrix, calculate true positive rate against false positive rate.
- Calculate Precision, Recall and F1- Score and visualize them over a graph.
- Cost for every iteration is calculated and visualized to observe Cost decrease for every iteration.
- Accuracy calculation is done by comparing the predicted labels to true labels and divided by total.
- Observing the predicted probabilities and coloring with true labels. Visualize decision boundary to look at comparison of predictions and actual labels.

4.4. Scalability Analysis and Results

I will apply the trained algorithm to test on data of different sizes to measure the performance and scalability of the algorithm, I will also compare my model with the MLlib as baseline to measure the efficiency of my algorithm.