Sparkify Project Post

1. Project Definition

This is udacity’s capstone project, using spark to analyze user behavior data from music app Sparkify.

1.1 Project Overview

Sparkify is a music app, this dataset contains two months of sparkify user behavior log. The log contains some basic information about the user as well as information about a single action. A user can contain many entries. In the data, a part of the user is churned, through the cancellation of the account behavior can be distinguished.

1.2 Problem Statement

The job of the project is to find the characteristics of churned users from the behavioral data of these users, and take measures to retain the potential lost users as early as possible.

The difficulty of the project is that the data handed to the model training should be one item for each person. While our data are behavioral data, we need to extract the user characteristics from the behavioral data, and then hand it to the model for training and get the results. These features will be generated from exploratory data analysis and feature engineering. In the case of poor results, the process may iterate many times until the model performs well.

1.3 Metrics

Since churn users is only a small part of all users, the model’s ability to identify churn cannot be measured by the accuracy. So recall rate is an important indicator. On the other hand, accuracy is an important indicator of the cost of retaining customers. Due to the the above goals, F1 is used as the measurement standard for training and evaluation.

2. Analysis

2.1 Data Load and Explore

The original size of the dataset is 12GB which is too large for my preliminary analysis, I used the small dataset (128MB) mini\_sparkify\_event\_data.json to perform data exploration process.

Use df.printSchema() to understand the contens of this dataset.

|-- 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)

First I have sorted out the meaning of all the columns

1. Artist: singer
2. Auth: login status
3. First Name:
4. Gender:
5. Iteminsession:sequence number
6. Last time:
7. Length:
8. Level: paid or free
9. Location
10. Method:PUT/GET
11. Page:page overview
12. Registration:timestamp for register
13. Sessionid:
14. Song:
15. Status:200/404etc.
16. Ts:timestamp for logging time
17. UserAgent:information

2.2 Data Preprocessing

Behavioral data is different from the final training data, and we need to clean up some values that are not in our consideration. UserId and sessionId are keywords whose behavior is difficult to analyze if they are null, so the null value need to be removed.

There are some empty strings in the userId that may be actions of unregistered users from their browsing pages. These users are not related to this churn rate analysis, so they are also removed from the data set.

2.2 Define Churn

I looked at all the page values, which recorded all the possible actions of the user. One things to confirm is Cancellation Confirmation, it means the churn users. And most of the rest of the operations are in the process of listening to the music. Downgrade related activities should also be very important for companies, but this Downgrade is not in our research range.

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

| page|

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

| Cancel|

| Submit Downgrade|

| Thumbs Down|

| Home|

| Downgrade|

| Roll Advert|

| Logout|

| Save Settings|

|Cancellation Conf...|

| About|

| Settings|

| Add to Playlist|

| Add Friend|

| NextSong|

| Thumbs Up|

| Help|

| Upgrade|

| Error|

| Submit Upgrade|

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

I looked at the records of some of churn users. Free users often receive a Roll Advert, which can have an impact on the user. Downgrade pages are sent to subscribers regularly, which may also have an impact. However, a similar situation can be found when looking at non-lost user records, so the impact cannot be determined.

I then added the churn column to the dataset to mark churn users, so now I can observe the difference between what happens to churn users and what doesn’t.

The ratio of males to females is slightly different.

looking at the paid status of the churn users, it seems that the paid users account for a large proportion, and then we can add this feature to see if it helps the model.

This is the frequency of each page, I used the number of operations per 100 to analyze. You can see that the churn users Roll Advert frequency was significantly higher than the non-churn users. In addition, the two operations of Cancel and Cancel Confirmation are used to distinguish the churn users, so these two data cannot be analyzed by the user model and need to be removed before training the model.

The number of operations per session doesn’t seem to make much difference between the two groups.

Finally, I checked the distribution of the two groups according to the time points of listening to the music. There were slight differences between the time and the week, and significant differences between the time points of the month. The difference in monthly time points may be due to the uniform distribution of user churn, resulting in fewer users at the end of the month.

Feature Engineering

The selection of features is based in part on user characteristics, including gender, registration time, whether paid, and how many singers have been heard. Some are based on behavioral data. The selection of behavioral data is mostly based on the session of each user, including the statistical value of listening time of each session, the number of listening songs in each session, and the number of sessions.

# Feature 1: number of days since registration

# Feature 2: avg & min & max time per session

# Feature 3: number of songs per session

# Feature 4: number of sessions

# Feature 5: gender

# Feature 6: whether the user is currently a paid user

# Feature 7: frequency of use of pages

# Feature 8: how many singers have the user heard

The final count is the number of pages per 100 views.I generated a dataframe for each feature separately, and finally made all features into a dataframe for the final training model by left linking.

# put all the features dataframe into a list

features\_list = []

features\_list.append(user\_reg\_days)

features\_list.append(user\_session\_time\_stat)

features\_list.append(user\_session\_songs\_avg)

features\_list.append(user\_session\_count)

features\_list.append(user\_gender)

features\_list.append(user\_recent\_level)

features\_list.append(user\_page\_view\_freq)

features\_list.append(user\_artist\_count)

features\_list.append(user\_churn)

Modeling

First, I confirmed the skew of data distribution.Data skew affects not only our metrics, but also our results, as we’ll see later.

Then I saved the data. In the process of my many times of training, I thought that after restarting the kernel and re-reading the data, the speed of training would be significantly faster.

Before spark training the model, it was necessary to put all the features into vectors. When putting data into vectors, I found that many of my features were string types, so I have to convert them to float.

Although it does not affect the tree model, standardization is important for linear models. I decided to standardize the data.

I use linear model and tree model for training, including logistic model, decision tree and GBT. I select the model with the best performance, divide the test set using 3-fold cross validation and grid search to determine the model parameters for the training set. Finally I use the validation set to evaluate the model performance.

Logistic regression’s hyper-parameters includes elastic network parameters and regularization parameters. This is mainly based on the different regularization effects.

The detailed hyper-parameters are configured as follows:

lr = LogisticRegression()

paramGrid = ParamGridBuilder() \

.addGrid(lr.elasticNetParam,[0.0, 0.1]) \

.addGrid(lr.regParam,[0.0, 0.05]) \

.build()

Decision tree’s hyper-parameters includes impurity and maximum depth. The maximum depth is for the overfitting case.

The detailed hyper-parameters are configured as follows:

dt = DecisionTreeClassifier()

paramGrid = ParamGridBuilder() \

.addGrid(dt.impurity,['entropy', 'gini']) \

.addGrid(dt.maxDepth,[2, 4]) \

.build()

GBT’s hyper-parameters includes max iterations and maximum depth. They are all for the overfitting case.

The detailed hyper-parameters are configured as follows:

gbt = GBTClassifier()

paramGrid = ParamGridBuilder() \

.addGrid(gbt.maxIter,[3, 10]) \

.addGrid(gbt.maxDepth,[2, 4]) \

.build()

Result

The result is not ideal, I decided to undersample the training data, balance the categories of the training set to increase the recall rate and improve f1.

F1 result:

lr: 0.55

lrs: 0.45

​

The best model is logistic regression. According to the results of the model, Registration time has the greatest impact.,the longer the registration time users will not leave.

Besides, the frequency of downgrade pages is also a big factor. I think this is because users hate the downgrade prompt.