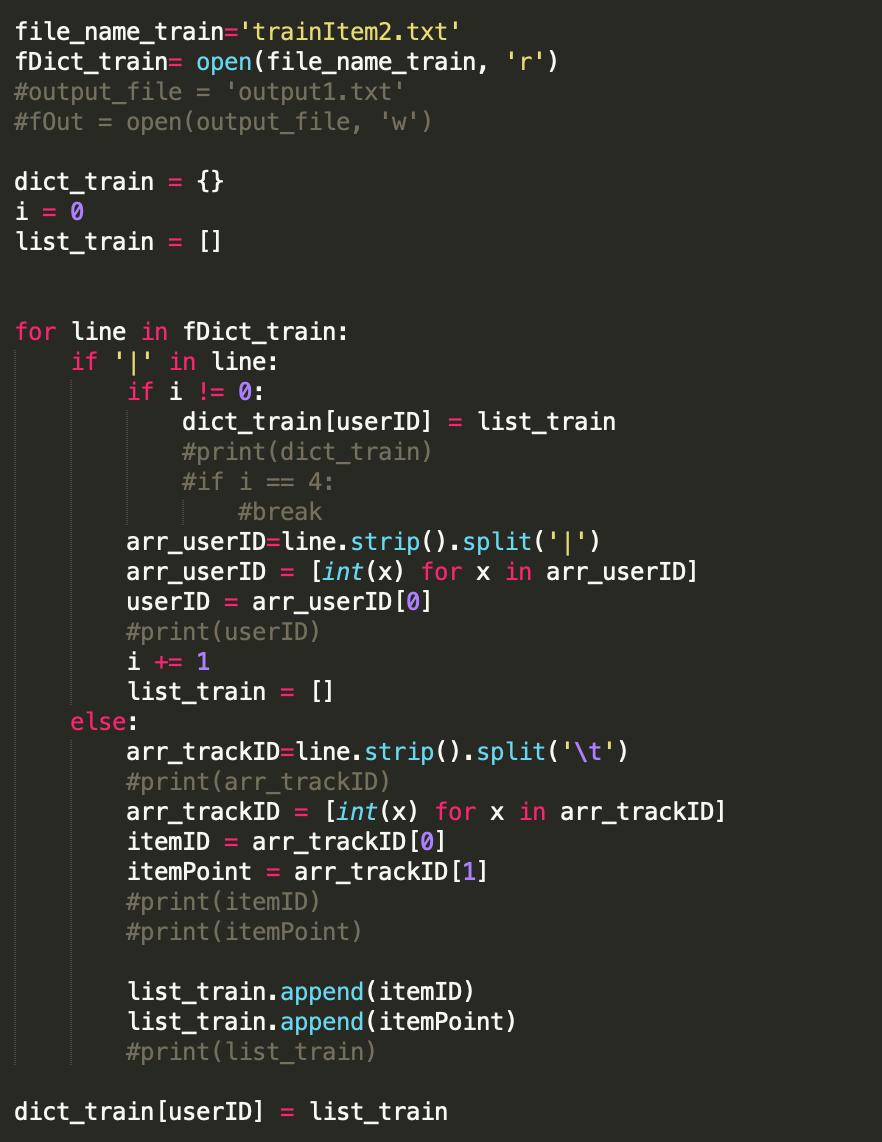
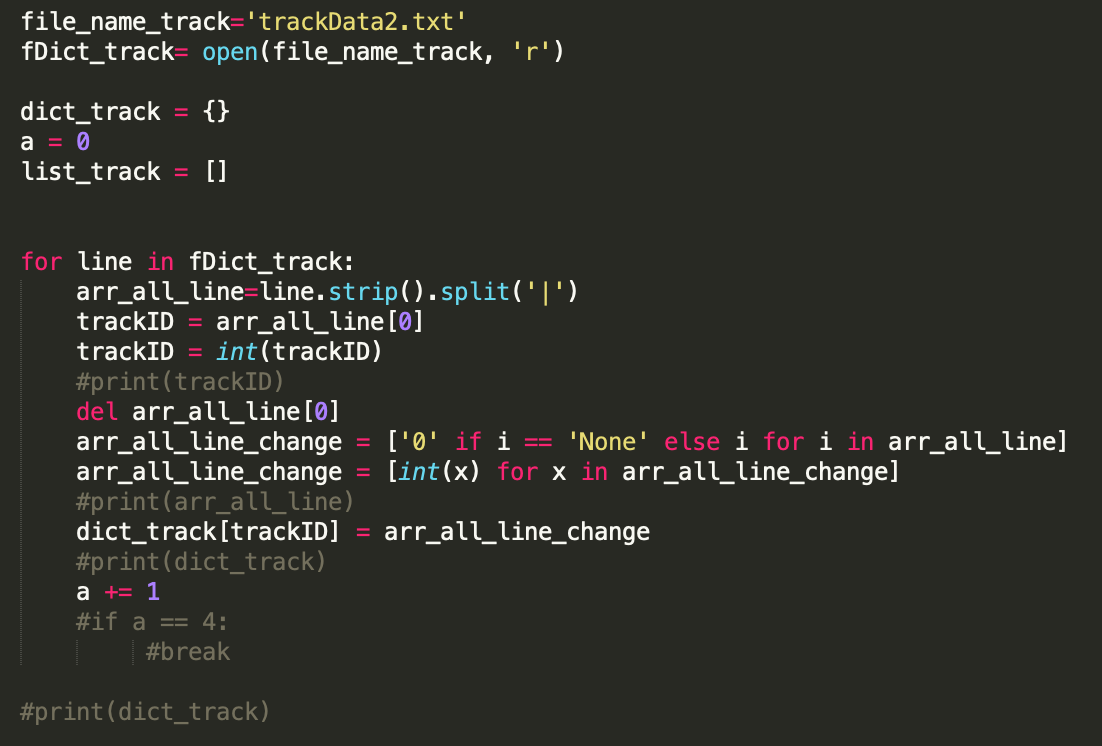
Organize Data（可以换个说法）

In order to process the data, we have to read all the data and put them together in one file or some data structure. At first, I planned to put them in some dictionary. Dictionary is in a good performance for us to edit and view our data in a short time. For this project, we build three dictionaries.

The first one is training data dictionary from trainitem2.txt. In this dictionary, keys are userID and values are lists. In every list, the odd index of value is itemID and the even index of data is score of the previous item. Like (userID, [itemID1, score1, itemID2, score2, itemID3, score3….]).

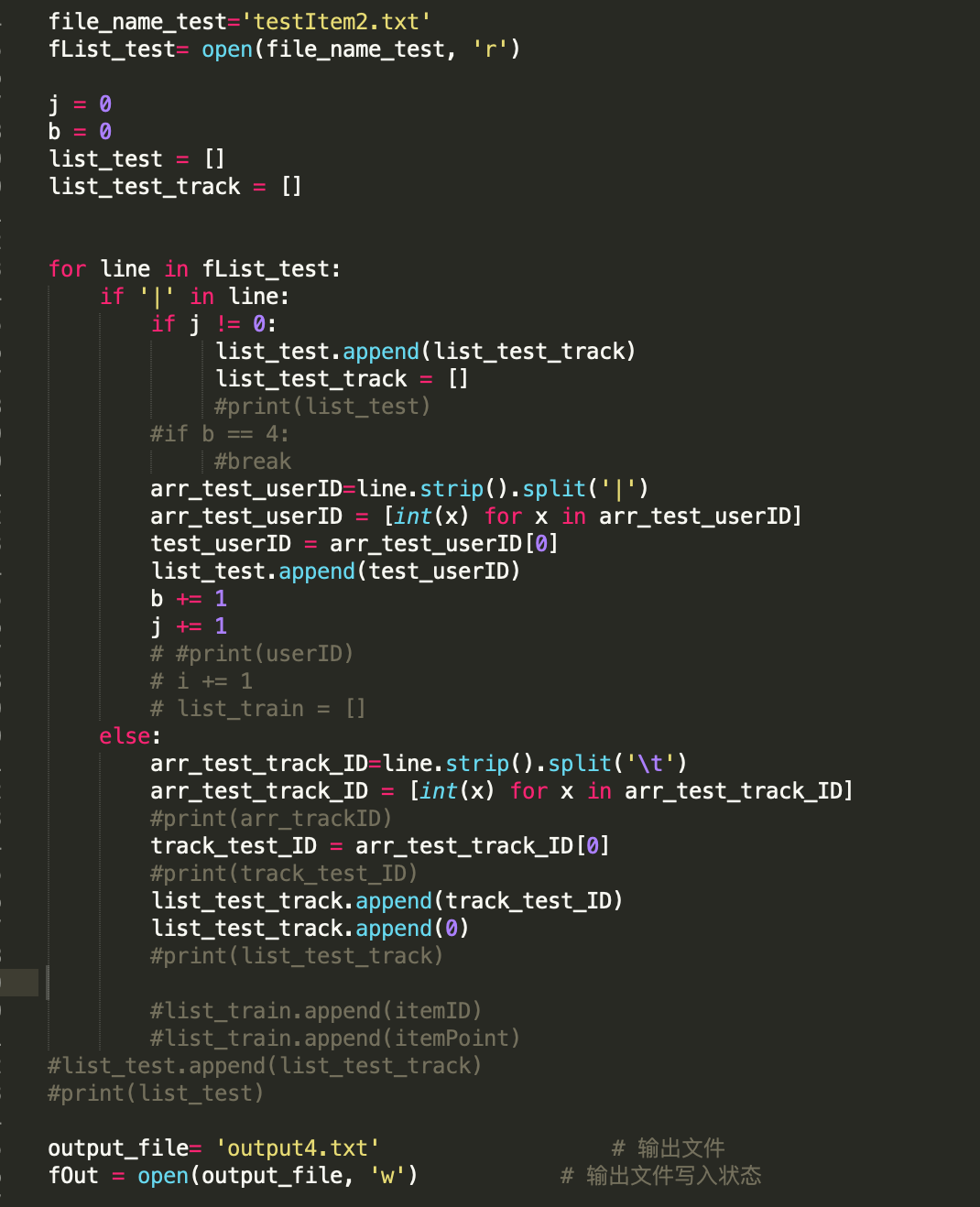


The second one is dictionary for track data from trackData2.txt. This dictionary contains all the relationship of all tracks and their album, artist, genres. Keys in this dictionary is trackID and values are albumID, artistID and genres.



The third one is a big list from testItem2.txt. In this list, the odd indexes are userID and the even indexes are small lists. In these small lists, odd indexes are trackID and even indexes are scores for the previous track.

Like [userID1, [trackID1, score1-1, trackID1-2, score1-2…. trackID1-6, score1-6], userID2, [trackID2-1, score2-1, trackID2-2, score2-2…. trackID2-6, score2-6] ….].



Now, all the data structures have been built. We can easily get every data we want from these three dictionaries and lists but not from those files every time we run our code. Also, it’s a good way to write all these data in our data structure into one file and only read this file at the begin of our code.

Simply Change Weight

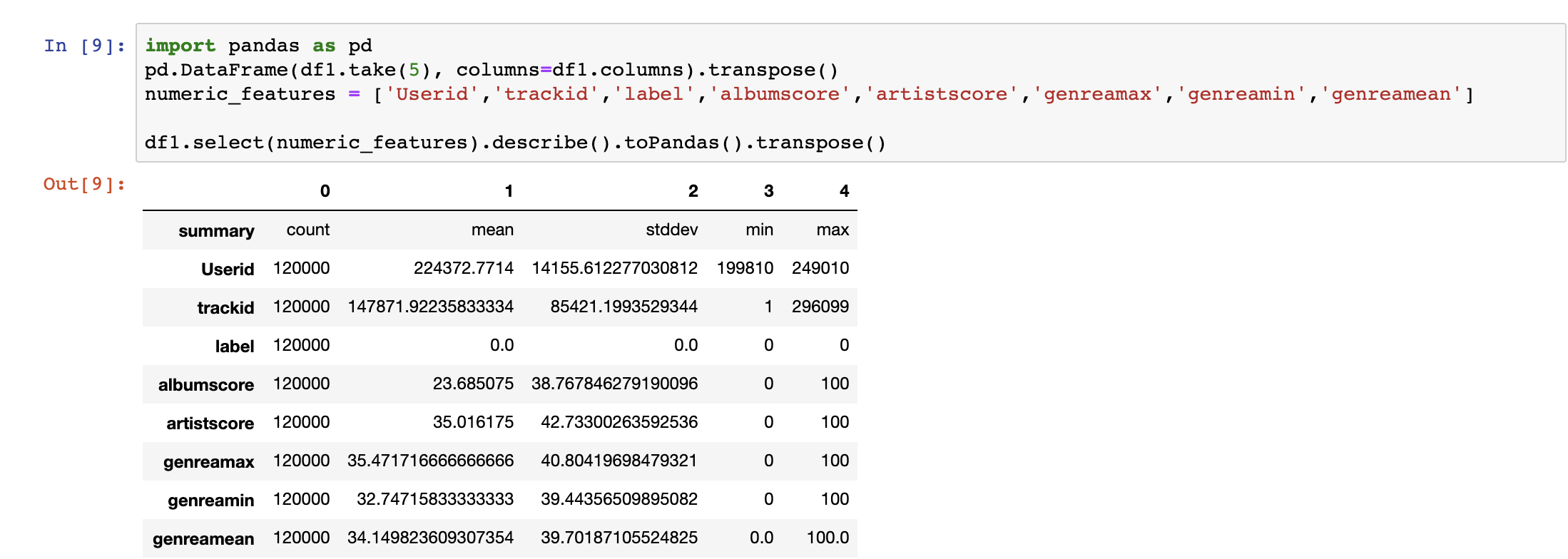
The simplest way to score all the track is just change the weight of all the feature we already have. We tried a lot of combination of weight. In the beginning, we limited the sum of all the proportions to 1. The best result we get in Kaggle is 0.86679. The combination is

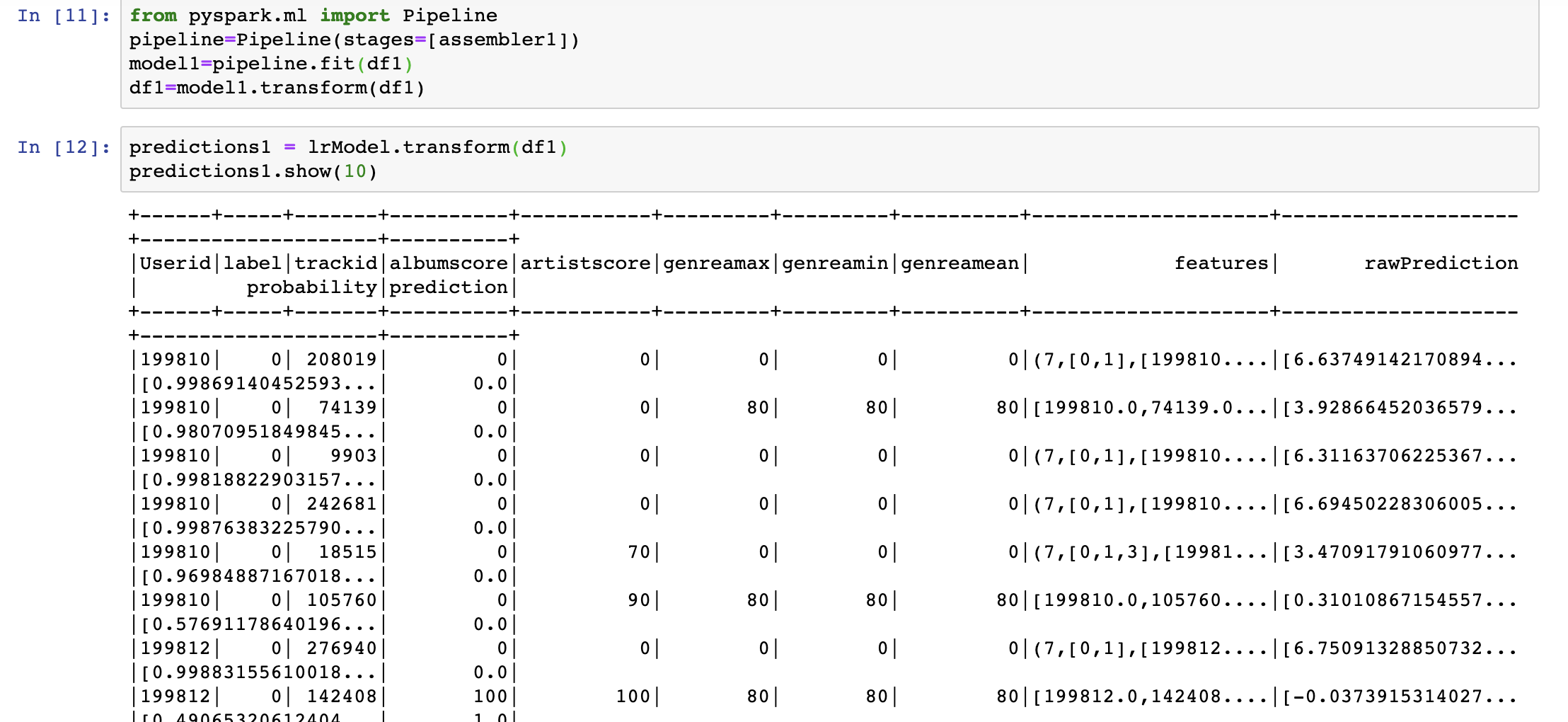
*Predict = Track + Album \* 0.6 + Artist \* 0.35 + Genre \* 0.05.*

By accident we find that we may get better score when the sum of these weight is not 1. Then we tried some new combination then we get 0.86841 in Kaggle with the following combination.

*Predict = Track + Album 0.912 + Artist \* 0.591 + Genre \* 0.05.*

Logistic Regression



Beside album, artist and genre score, we get some other numeric feature like the max, min and mean of genre. In order to build our logistic regression model by spark pipeline, we have to transform all our data type from string to integer. Then we just call the function in pipeline to build our logic regression model. Using this model to get a new feature column with those numeric features. After that we can preview some output of feature.

Then get the probability as the basis for scoring. The last step is scoring every element with the probability we get. After uploading our output of logic regression, we get 0.83256 in Kaggle.