Recommendations

|  |  |
| --- | --- |
| **Summary** | In this codelab, we demonstrate content-based model and collaborative filtering for recommendations |
| **URL** | <https://codelabs-preview.appspot.com/?file_id=1o68Pa17h-kSY75WN28O1A3yWCBnY2ars421qp40y4Ds#0> |
| **Category** | Data Modeling |
| **Environment** | web, kiosk, |
| **Status** | Published |
|  |  |
| **Author** | Dingling Ge, Zefan Feng |
|  |  |
|  |  |

[Introduction](#_ok7k5uux6)

[Summary of the project and data](#_x0me2ipv51gl)

[Instructions](#_21yzqg98x7h6)

[DataPreprocess](#_bwo0af2iwk90)

[Content-Based Model--DNN Model](#_rfoc73tu5bp6)

[Collaborative Filtering](#_8t47rdksfmsb)

[Conclusion](#_jdmzwkcs72o)

[Appendix](#_ot43tirxvs6w)

# Introduction

**Last Updated:** 2019-11-24

## **Summary of the project and data**

The rapid growth of data collection has led to a new era of information. Data is being used to create more efficient systems and this is where Recommendation Systems come into play. Recommendation Systems are a type of **information filtering systems** as they improve the quality of search results and provides items that are more relevant to the search item or are related to the search history of the user.

We create recommenders using content- based and collaborative filtering. These two approaches are proved to be almost complimentary.

For the Dataset:

This dataset contains social networking, tagging, and music artist listening information

from a set of 2K users from Last.fm online music system.

http://www.last.fm

The dataset is released in the framework of the 2nd International Workshop on

Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011)

http://ir.ii.uam.es/hetrec2011

at the 5th ACM Conference on Recommender Systems (RecSys 2011)

http://recsys.acm.org/2011

1892 users

17632 artists

12717 bi-directional user friend relations, i.e. 25434 (user\_i, user\_j) pairs

avg. 13.443 friend relations per user

92834 user-listened artist relations, i.e. tuples [user, artist, listeningCount]

avg. 49.067 artists most listened by each user

avg. 5.265 users who listened each artist

11946 tags

186479 tag assignments (tas), i.e. tuples [user, tag, artist]

avg. 98.562 tas per user

avg. 14.891 tas per artist

avg. 18.930 distinct tags used by each user

avg. 8.764 distinct tags used for each artist

## **Instructions**

* Load data
* Data preprocess

1. Artist file
2. User file
3. Friend feature
4. Feature crosses

* Content-based Recommendation

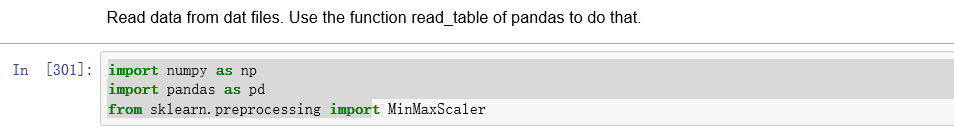
1.DNN model

* Collaborative Filtering for Recommendation
  + - 1. Surprise
      2. KNN

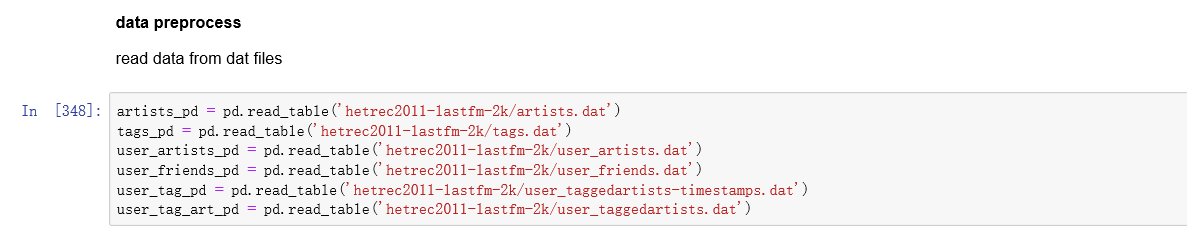
# DataPreprocess

Duration: 15:00

Read data



Read data from dat files



For Artist profile

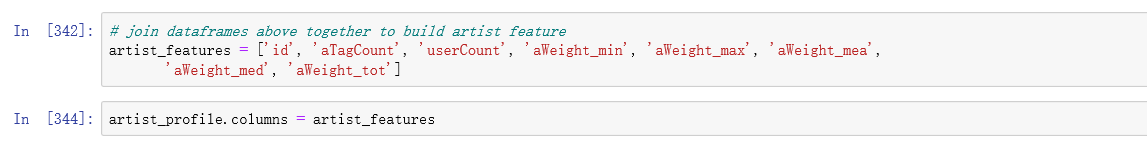
We created features of artists’ tags, artists’ tags user count( the amount of users who use the tag to artist), artists’ listened weight min(the minimum weight of being listened to by users), artists’ listened weight max, artists’ listened weight mean, artists’ listened weight median and artists’ listened weight sum.







Then join the above data frames together to build artist feature

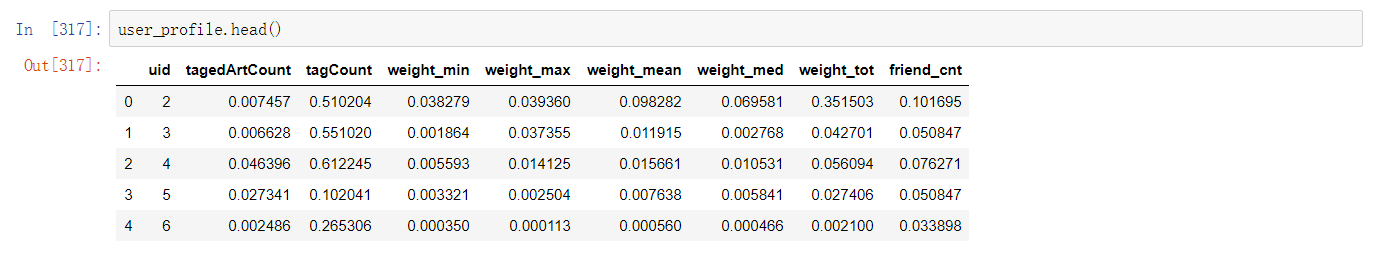


And it would be the same process with User file and Friend Features.

For User profile

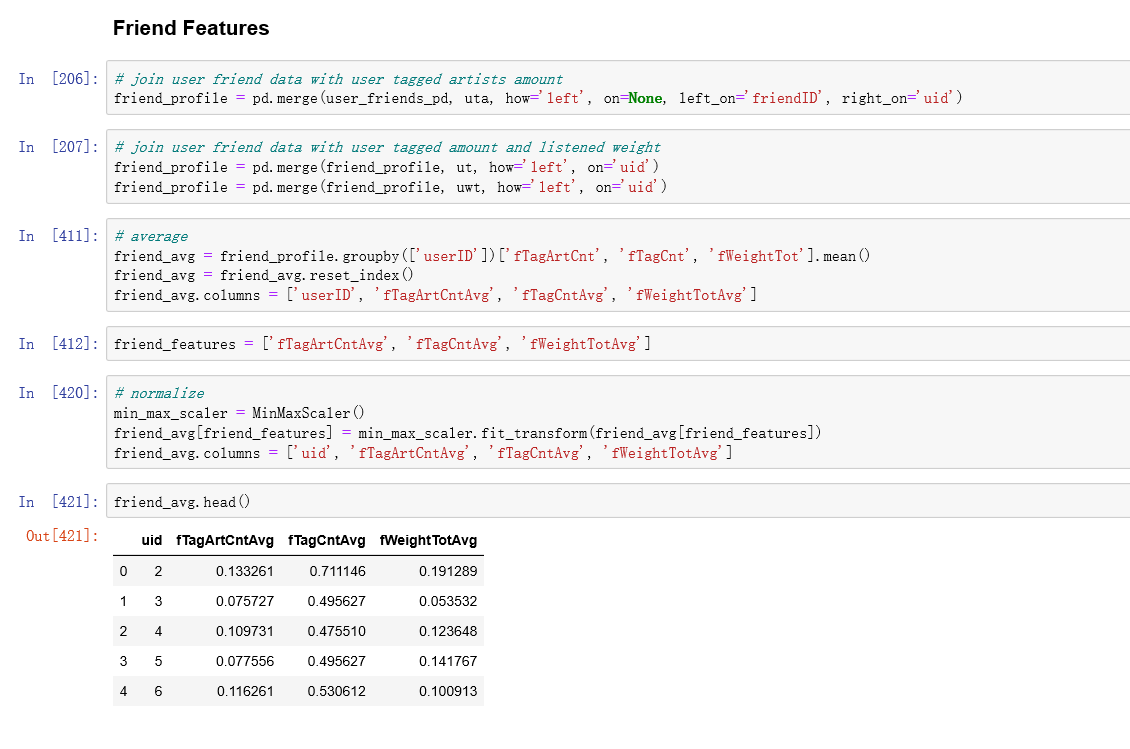
We created features of the amount of artists user tagged, the amount of tags user tagged, user listened artists' weight min, user listened artists' weight max, user listened artists' weight mean, user listened artists' weight median, user listened artists' weight sum, and users' friends amount. And the next step we join the dataframe above together to build the user feature.

Head the user profile we can see this:



For Friend Feature, we join the user friend data and user tagged artist amount, join user friend data with user tagged amount and listened weight ,average and then normalize it.

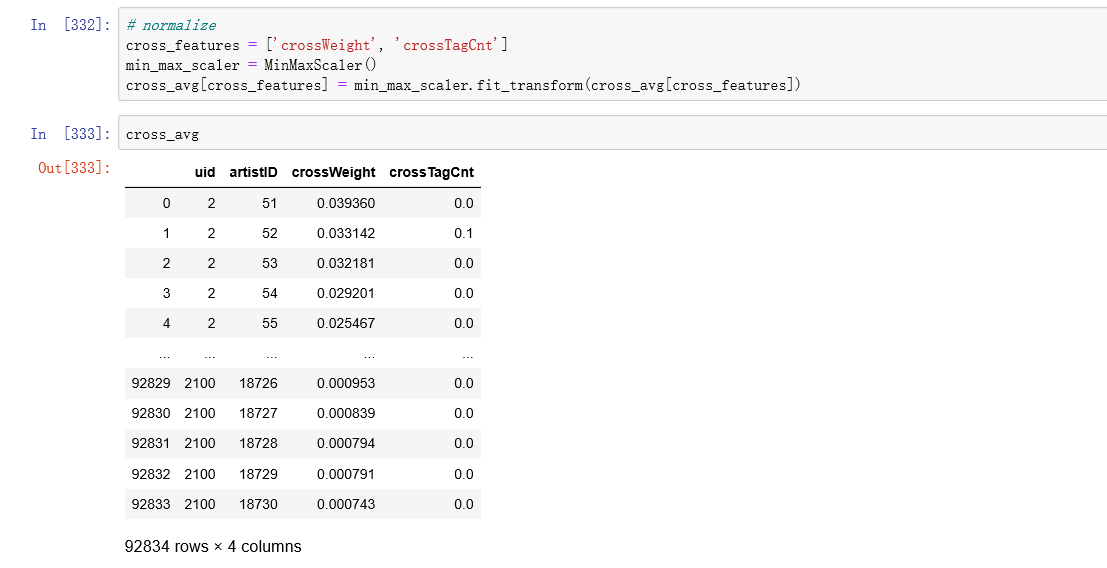
The following is based on the user's friend relationship and user characteristics. The user's friend characteristics are the average number of marks, artist number of marks and average weight of all the users' friends.



The last step is to do the feature crosses. A feature cross is a synthetic feature formed by multiplying (crossing) two or more features. Crossing combinations of features can provide predictive abilities beyond what those features can provide individually, and cross features have better performance on NON-linear Models.

In our step, the cross between user and artist is the average number of users marked artist





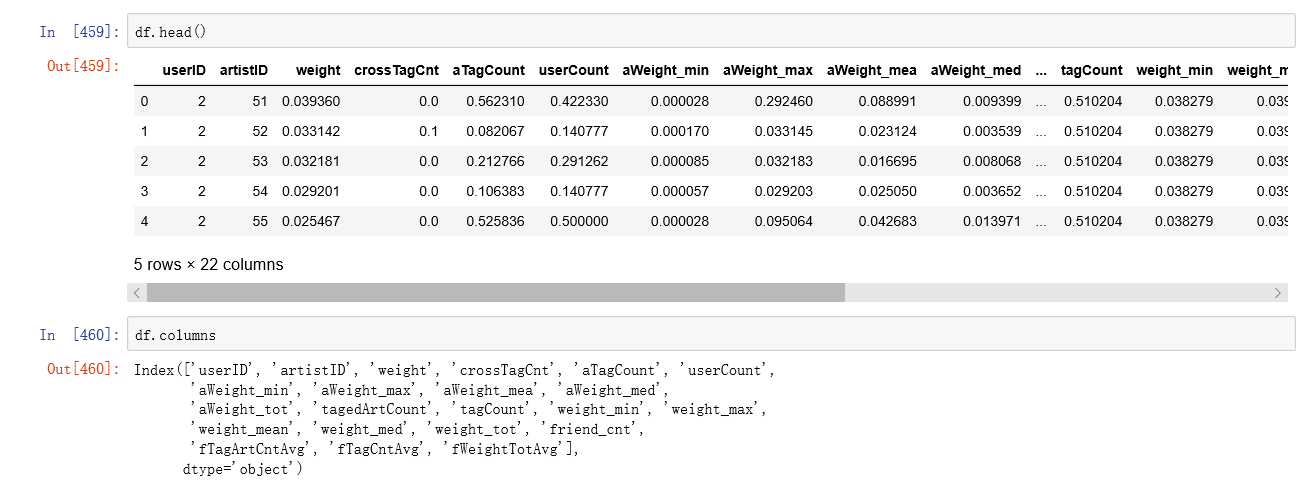
After all these characteristics are obtained, they are all joined together to form a large table and then be normalized. Since that, we are able to begin to train models. The features of the model are above, the label of the model is the weight of users under artists, which is the extent of interests the users have in artists.

# Content-Based Recommendation--DNN Model

Duration: 15:00

* the example from the 'recommander' Github link was similar to our processes, the whole idea and logic is the same. But it used a linear model, which might not work very well. So, we choose a different model, which can performance better.
* Use DNN model to predict the weight.The bigger the weight is, the more interest of user.
* Deep neural network (DNN) models can address the limitations of matrix factorization. DNNs can easily incorporate query features and item features (due to the flexibility of the input layer of the network), which can help capture the specific interests of a user and improve the relevance of recommendations.
* In this recommender system the content of the artist(tag) is used to find its similarity with other artists. Then the artists that are most likely to be similar are recommended.

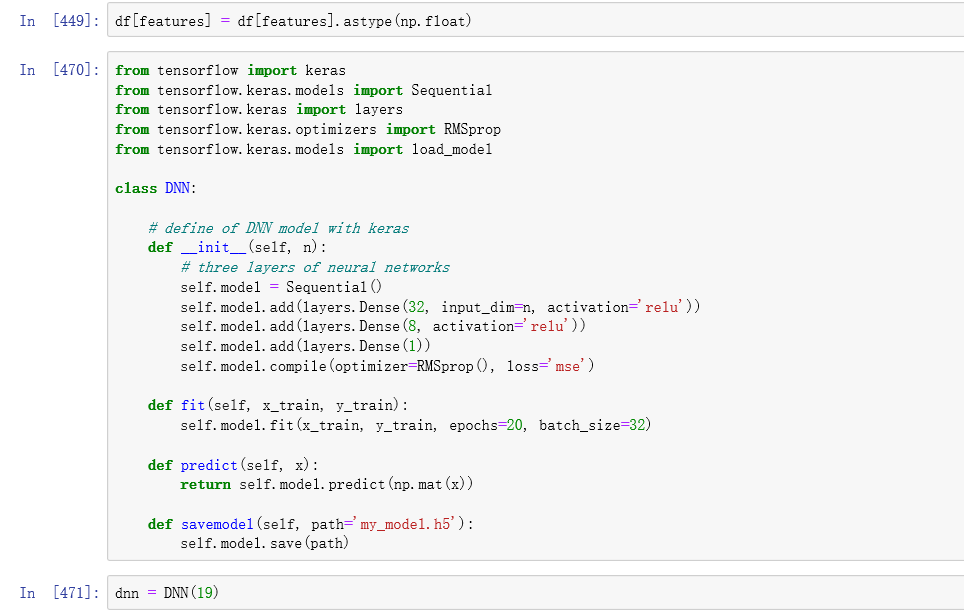


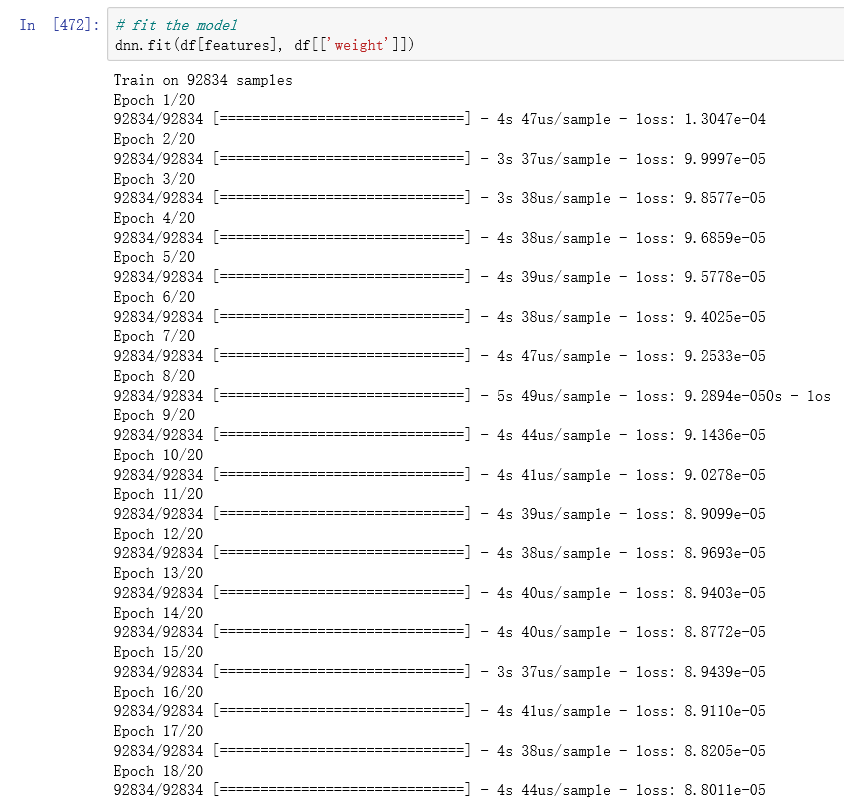




Since the features and outcome may not be linear, we choose the DNN model which has better performance at Non-linear models. (the refer link well be listed at the end of the report)

And we define 3 layers in DNN model, then train the model.



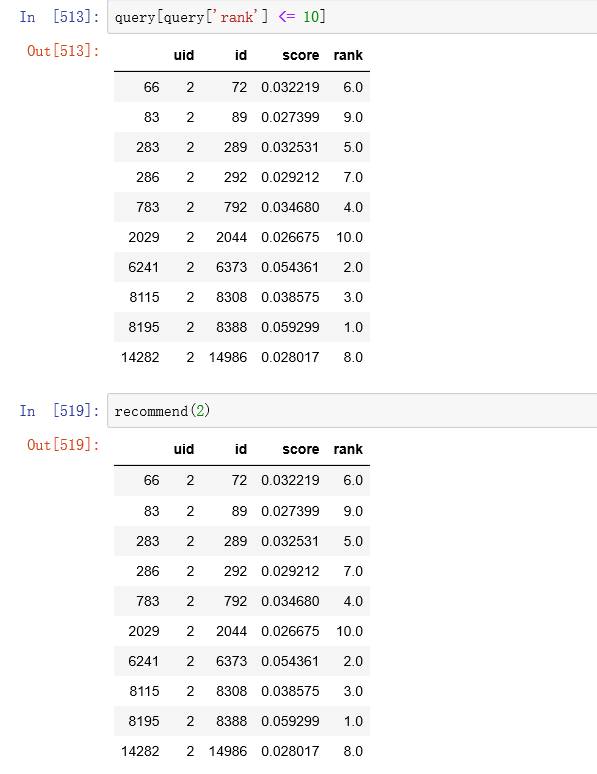


In the next step, we recall all the artists, use the model to predict and score all the recalls (recall is the ratio of accurate record and all records. The outcome more close to 1,the more accurate).



We are going to define a function that takes in a userID as an input and outputs a list of the 10 recommended artistID. In other words, the whole process is to get user characteristics through uid and join with all other characteristics. And then you throw it into the model to predict, and you get the results of the predict and sort them.

Sort by score and return top 10.So all together, given a user id, return the recommended 10 artists, like below:



# Collaborative Filtering for Recommendation

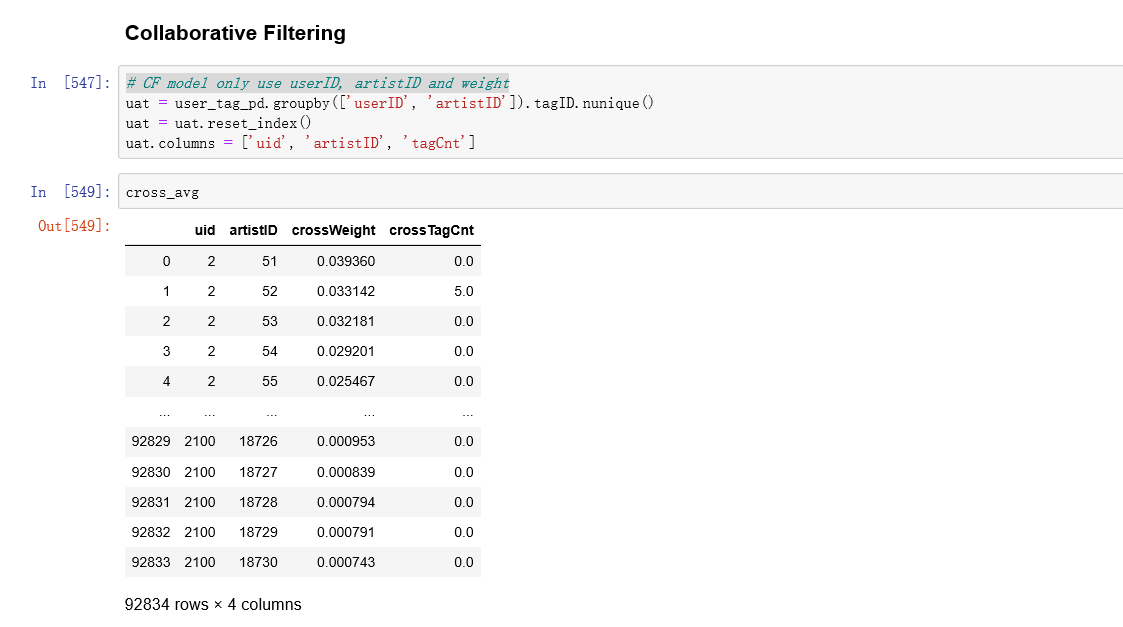
Duration: 15:00

Our content based engine suffers from some severe limitations. It is only capable of suggesting music which are close to a certain music. Also, the engine that we built is not really personal in that it doesn't capture the personal tastes and biases of a user. Anyone querying our engine for recommendations based on a music will receive the same recommendations for that music, regardless of who she/he is.

Therefore, in this section, we will use Collaborative Filtering to makerecommendations to Music listeners.

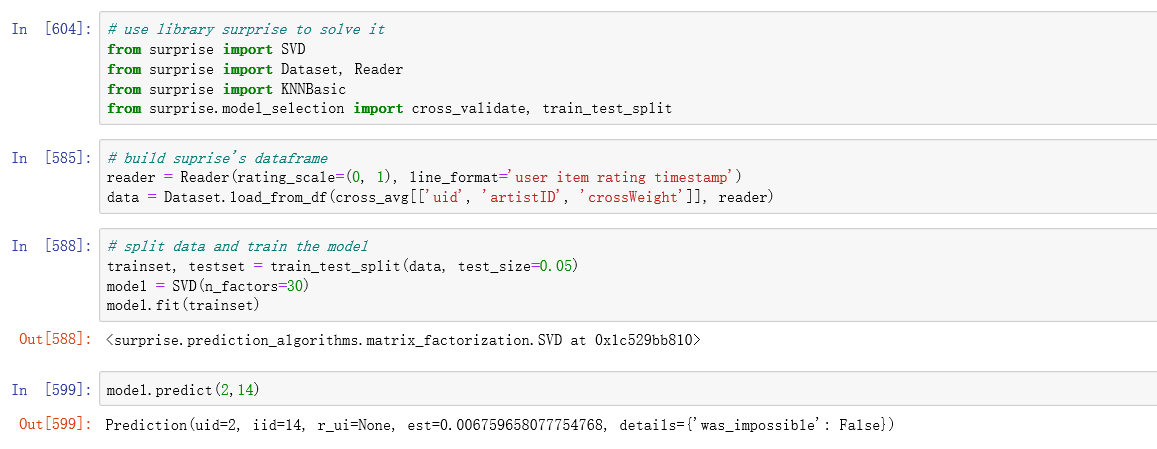
**Collaborative Filtering** doesn’t need anything else except users’ historical preference on a set of items. Because it’s based on historical data, the core assumption here is that the users who have agreed in the past tend to also agree in the future.

We only use userID, artistID and weight features in CF Model. (the refer link well be listed at the end of the report)

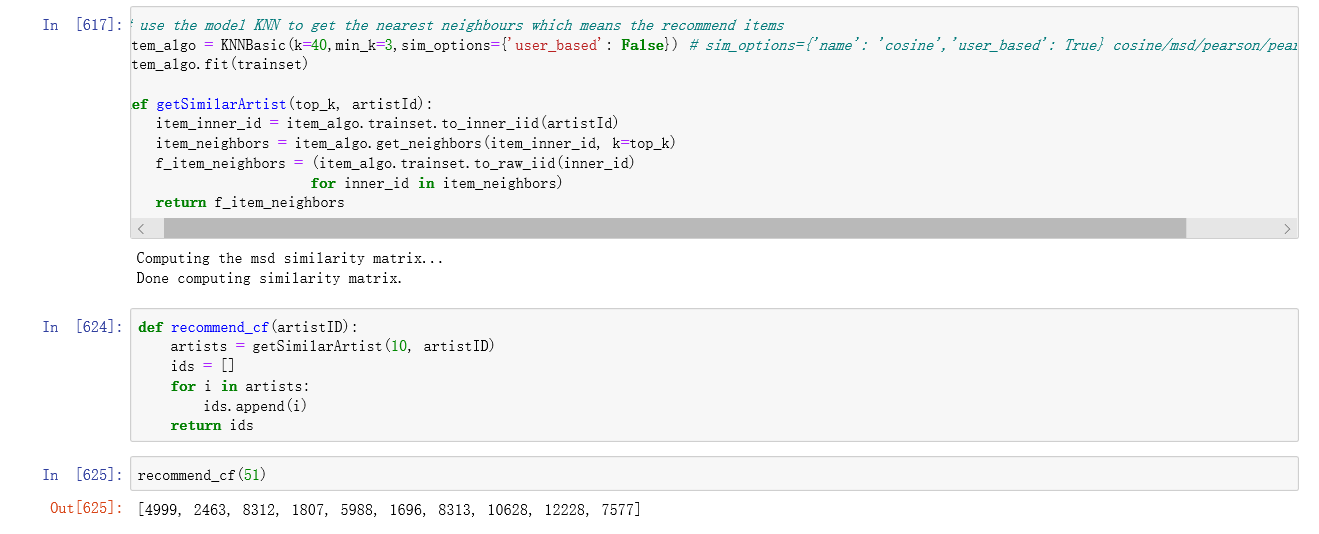


In the next step we use [Surprise library](http://surpriselib.com/) to solve it. Surprise library is a library used for recommender systems.

We use SVD ( a CF model) to train the model and get an artist\*user matrix from which we are able to calculate the similarity between users and users, artists and artists.



Then we use the KNN model to get the nearest neighbours which is the recommended artist in this project. The k-nearest neighbors (**KNN**) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand. Essentially, we want to turn the recommendation problem into an optimization problem. We can view it as how good we are in predicting the rating for items given a user.



Here we define a function that takes in an artistID as an input and outputs a list of the 10 recommended artistID.

# Conclusion

Duration: 1:30

We create recommenders using content- based and collaborative filtering. These two approaches are proved to be almost complimentary. This model was very baseline and only provides a fundamental framework to start with.

We can find that, for nonlinear characteristics, the effect of the neural network is very good, although the job is simply use a very basic neural network model, but we can see its well performance , We think It has the high value to study. In the future, we will study it further, and we believe that it will have a better performance in other fields.

We would like to mention some excellent references that we learned from:

1. <https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75>
2. <https://www.kaggle.com/rounakbanik/movie-recommender-systems>
3. 模型学习和源码引用的参考链接：
4. For DNN：
5. <https://keras.io/getting-started/functional-api-guide/> (English)  
   <https://blog.csdn.net/BF02jgtRS00XKtCx/article/details/85812754> (foreign language）  
   <https://blog.csdn.net/jclian91/article/details/83024344> (foreign language）
6. for surprise：
7. <http://surpriselib.com/> (English)  
   [https://blog.csdn.net/mycafe\_/article/details/79146764#12-音乐预测的例子](https://blog.csdn.net/mycafe_/article/details/79146764#12-%E9%9F%B3%E4%B9%90%E9%A2%84%E6%B5%8B%E7%9A%84%E4%BE%8B%E5%AD%90) (foreign language)

# Appendix

Duration: 3:00

You can look up the detailed code on [Github](https://github.com/Featuretools/predict-next-purchase/blob/master/Tutorial.ipynb)

References:

1. DNN Model: <https://keras.io/>
2. Surprise: <http://surpriselib.com/>