

# 音乐推荐系统 项目总结

李倩

2020 年 3 月

# 摘要

本项目属于 Kaggle 的一个音乐推荐比赛，项目要求基于 KKbox 音乐网站提供的一段时间内用户歌曲数据，实现对用户在首次播放歌曲之后一段时间内再次播放的概率预测。本文针对数据集特点和音乐推荐方面的背景知识，进行了较为充分的特征工程，并在此基础上融合 LightGBM 和神经网络模型，实现了较好的预测效果。

在数据探索方面：研究了训练集和测试集的数据时间分布，确定了时序性数据的验证集划分方法；基于数据规模和实际硬件配置，确定了 LightGBM 模型采用部分数据分批训练的训练策略。

在特征工程方面：基于对数据集中字段含义的理解，对数据集中的缺失和异常值进行了合理填充；采用统计方法（均值、标准差等）和概率方法（如后验概率）等方法对特征维数进行了大幅扩充，使特征维数从 20 个增加到 296 个，有效提取了数据集中数据之间的内在联系。

在模型选择方面：考虑到数据规模，选择了能够有效处理大数据、速度快的 LightGBM 模型，分批次采用训练集中的部分数据训练多个模型并进行加权融合；考虑到神经网络模型优秀的泛化能力，构建了能够有效融合不同类别特征的网络结构，并对基于不同超参训练得到的模型进行融合；最后融合 LightGBM 和神经网络，优势互补，提高了模型在测试集中的预测效果。

基于以上工作，本文实现了测试集上较为优异的预测效果，在 LeadBoard 上的 Public 分数位列第九位。

# 目录

摘要.....	2
目录.....	3
图目录.....	6
表目录.....	8
1 数据探索.....	9
1.1 引言.....	9
1.2 数据集简介.....	9
1.2.1 train.csv 简介.....	9
1.2.2 test.csv 简介.....	9
1.2.3 members.csv 简介.....	10
1.2.4 songs.csv 简介.....	10
1.2.5 song_extra_info.csv 简介.....	10
1.3 数据集基本信息.....	11
1.3.1 用户表.....	11
1.3.2 歌曲表.....	12
1.3.3 训练集.....	14
1.3.4 测试集.....	16
1.4 数据统计.....	18
1.4.1 无关数据统计与清理.....	18
1.4.2 统计冷启动数据.....	18
1.4.3 统计歌曲被重复收听的规律.....	18
1.4.4 缺失值和异常值处理.....	19
1.4.5 个别字段针对性处理.....	19
1.4.6 时序特性.....	21
1.4.7 统计用户和其他信息的相关规律.....	22
2 特征工程.....	25
2.1 引言.....	25
2.2 原始类别型特征的处理.....	25
2.3 songs 表和 song_extra_info 表的特征处理.....	25
2.3.1 流派特征的处理.....	25

2.3.2 歌手、作词、作曲特征的处理.....	25
2.3.3 song_extra 字段处理 .....	26
2.3.4 数值型特征的处理.....	26
2.3.5 歌曲播放次数及歌曲被重复播放的概率.....	26
2.3.6 歌曲热度特征.....	27
2.4 用户表数据处理 .....	27
2.4.1 用户的相应特征.....	27
2.4.2 用户-歌曲以及用户-歌手关系 .....	27
2.4.3 用户活跃度.....	28
2.5 基于特征交叉的隐含特征提取 .....	28
2.6 各表数据整合 .....	29
3 传统机器学习模型与训练 .....	30
3.1 引言 .....	30
3.2 LogisticRegression 模型 .....	30
3.3 SVM 模型.....	30
3.4 RandomForest 模型.....	31
3.5 LightGBM 模型.....	31
3.5.1 模型训练过程.....	31
3.5.2 结果分析.....	33
4 神经网络模型 .....	35
4.1 引言 .....	35
4.2 网络总体结构 .....	35
4.2.1 基于 field-aware 的特征融合.....	35
4.2.2 基于多维特征融合的分类网络.....	37
4.3 模型训练 .....	37
4.4 模型融合 .....	38
5 LightGBM 和神经网络融合 .....	39
5.1 引言 .....	39
5.2 融合方法 .....	39
6 项目感想与收获 .....	40
6.1 引言 .....	40
6.2 遇到的问题 .....	40

6.2.1 超大的数据集规模造成的影响.....	40
6.2.2 算法实现中遇到的问题.....	40
6.3 项目过程中的收获和感想 .....	41

# 图目录

图 1.1 用户表的字段缺失值比例.....	11
图 1.2 年龄数据分布.....	12
图 1.3 歌曲表部分数据.....	12
图 1.4 language 数据分布 .....	13
图 1.5 song length 数据分布 .....	13
图 1.6 训练集中不同标签样本分布.....	14
图 1.7 训练集 source_system_tab 数据分布 .....	15
图 1.8 训练集 source_screen_name 数据分布 .....	15
图 1.9 训练集 source_type 数据分布 .....	16
图 1.10 测试集 source_system_tab 数据分布 .....	16
图 1.11 测试集 source_screen_name 数据分布 .....	17
图 1.12 测试集 source_type 数据分布 .....	17
图 1.13 训练集歌曲 plays 和 repeat_chance 关系图 .....	19
图 1.14 歌手数和歌曲数的关系图.....	20
图 1.15 作词人数量和歌曲数的关系图.....	20
图 1.16 作曲人数量和歌曲数的关系图.....	21
图 1.17 歌曲年份在训练集和测试集中的分布图.....	21
图 1.18 用户播放歌曲次数分布图.....	22
图 1.19 会员时长分布图.....	23
图 1.20 用户注册年份和播放歌曲数目关系图.....	23
图 1.21 注册年份和歌曲重复播放概率关系图.....	24
图 3.1 LightGBM 模型 A 的排名 .....	32
图 3.2 LightGBM 模型融合策略.....	32
图 3.3 LightGBM 融合模型的排名.....	33
图 3.4 LightGBM 模型输出的模型重要性排序.....	33
图 4.1 User Field 网络结构图.....	35
图 4.2 Song Field 网络结构图 .....	36

图 4.3 Context Field 网络结构图.....	36
图 4.4 分类网络结构图.....	37
图 4.5 融合之后神经网络模型的排名.....	38

# 表目录

表 1.1 train 表的字段简介 .....	9
表 1.2 members 表的字段简介 .....	10
表 1.3 songs 表的字段简介 .....	10
表 1.4 song_extra_info 表的字段简介 .....	11
表 4.1 不同个数的模型融合效果 .....	38
表 5.1 不同融合方法的效果 .....	39



# 1 数据探索

## 1.1 引言

本章首先介绍数据集的基本概况，分别对 train、test、members、songs 和 song\_extra\_info 五个数据表的各个字段含义进行介绍；然后从用户表、歌曲表、训练集和测试集四个方面对数据集的基本信息进行探索；最后深入到各个特征中，通过统计方法研究特征的数据特点和特性。

## 1.2 数据集简介

该数据集是 2018 年的 Kaggle 竞赛——KKbox 音乐推荐挑战赛提供的，用来预测用户在第一次听过某个歌曲之后一段时间内，再次收听的概率。数据集包含了训练集 train.csv、测试集 test.csv、用户相关属性集 members.csv、歌曲相关集 songs.csv 和歌曲额外信息集 song\_extra\_info.csv 五个文件。下面将介绍各个文件的基本情况及其字段含义。

### 1.2.1 train.csv 简介

train.csv 包含六个字段，主要为用户、歌曲、来源信息和标签，具体信息如表 1.1 所示。

表 1.1 train 表的字段简介

字段名称	字段含义
msno	用户 ID
song_id	歌曲 ID
source_system_tab	用户获取歌曲的方式
source_screen_name	用户播放时显示的播放路径
source_type	用户第一次播放歌曲的获取方式
target	标签，1 表示用户一个月内重复收听

### 1.2.2 test.csv 简介

test.csv 包含六个字段，分别为 msno、song\_id、source\_system\_tab、

source\_screen\_name、source\_type 以及 id，与 train.csv 的区区别仅在于多了 id，少了 target。id 用来在最后提交预测结果时区分样本。

### 1.2.3 members.csv 简介

members.csv 包含七个字段，主要为用户的各方面信息，具体信息如表 1.2 所示。

表 1.2 members 表的字段简介

字段名称	字段含义
msno	用户 ID
city	所在城市
bd	年龄，包含异常值
gender	性别
registered_via	会员注册途径
registration_init_time	会员注册时间
expiration_date	会员过期时间

### 1.2.4 songs.csv 简介

songs.csv 包含七个字段，主要为歌曲的各方面信息，具体信息如表 1.3 所示。

表 1.3 songs 表的字段简介

字段名称	字段含义
song_id	歌曲 ID
song_length	歌曲长度，单位为 ms
genre_ids	编码之后的流派信息，多个流派之间用‘ ’分割
artist_name	歌手
composer	作曲
lyricist	作词
language	语言

### 1.2.5 song\_extra\_info.csv 简介

song\_extra\_info.csv 包含三个字段，主要为歌曲的一些额外信息,具体信息如下表 1.4 所示。

表 1.4 song\_extra\_info 表的字段简介

字段名称	字段含义
song_id	歌曲 ID
song_name	歌曲名
isrc	国际标准音像制品编码

## 1.3 数据集基本信息

下面主要深入到各个数据表内部，通过一些统计方法进行数据探索。

### 1.3.1 用户表

#### (1) 缺失值统计

用户表包含 34403 条数据，查看各个字段缺失值的比例，结果如图 1.1 所示。

```
msno          0.000000
city          0.000000
bd            0.000000
gender        0.578496
registered_via 0.000000
registration_init_time 0.000000
expiration_date 0.000000
dtype: float64
```

图 1.1 用户表的字段缺失值比例

可以发现性别信息的缺失比例超过了一半，可以考虑用单独的一个值（如 other）进行填充。

#### (2) 年龄

对年龄数据 bd 进行统计，结果如图 1.2 所示，可以发现年龄数据存在一些异常值（处于 1000 附近的数据），可以将这些异常值归为一类。

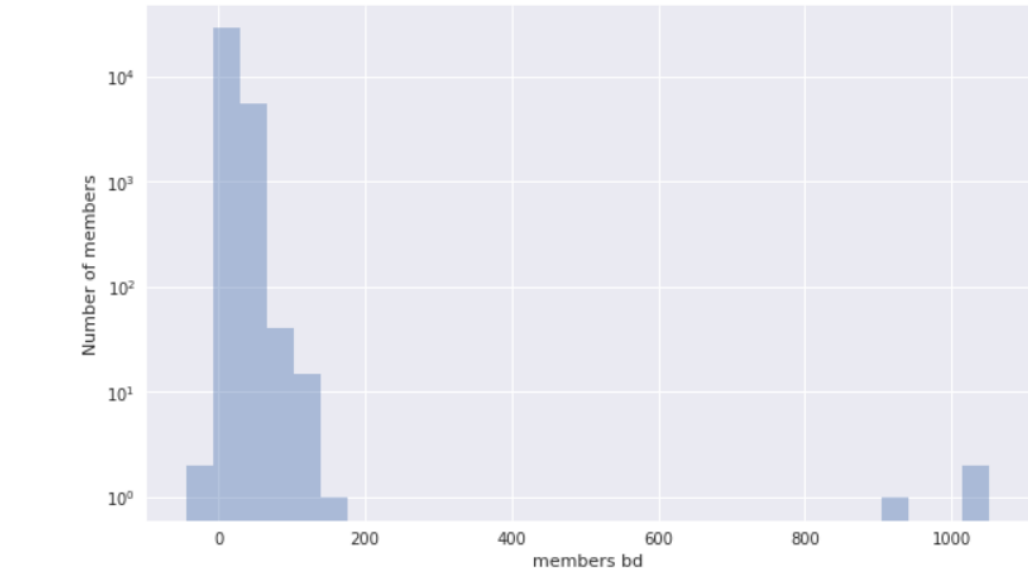


图 1.2 年龄数据分布

### 1.3.2 歌曲表

#### (1) 总体信息

歌曲表包含 2296320 条数据，数据量非常大，展示第 10-18 条数据看一下基本信息，如图 1.3 所示。

	song_id	song_length	genre_ids	artist_name	composer	lyricist	language
9	btcG03OHY3GNKWccPP0auvtSbhxog/killlOx5grE/k=	232629	352 1995	Kodaline	Stephen Garrigan  Mark Prendergast  Vincent Ma...	Stephen Garrigan  Mark Prendergast  Vincent Ma...	52.0
10	HuIM/OaHgD5kUyJNQjDUf8VZdsy7h4EJUiff79Cifwo=	272544	2157	D.L 羅時豐 (Daniel Lo)	陳偉強	陳偉強	10.0
11	wypPzqFNdUJAqyBVxmFGaK4z7krUNWr5YqA0q0wi9eE=	254880	465	白安 (Ann)	白安	白安	3.0
to hide output	dfQaL G76a6Ei4alt1eSjBM9rshQkiQEC6+n+y08=	180871	726	LittleSong	Michael William Balfe	NaN	-1.0
13	tqBIH4r/q1Tf6C5+C6ucjGILjMbfu5yjqB6iffRzy5dc=	257602	458	蔡旻佑 (Evan Yo)	蔡旻佑	陳信延	3.0
14	an6Edlr+Z+KbqIVQIXn5PKkcXncefQ7hhWONseRuub4=	282697	359	Coldplay	Chris Martin  Guy Berryman  Jonny Buckland  Mi...	NaN	52.0
15	J2MFmy8iF94mExWfRWE3KxsMZB+ZledV5liqZoSrERQ=	221518	359	Maggie Rogers	Maggie Rogers  Nicholas Das	NaN	52.0
16	MrRilXQwoUAcoAf0N3RT82qX2/us/wEhYDXE+ZTIW5o=	199157	458	小男孩樂園 (Men Envy Children)	Skot Suyama陶山/"蔡詩藝 Dominique Tsai"	程懷昌 Vince Cheng	3.0
17	OcG4Ya7iXmVMCMY24C5wxDMtr9w6WQZIFaN0uq6zdTk=	169430	465	BIGBANG	TAEYANG  TEDDY	G-DRAGON  T.O.P  TEDDY	3.0
18	JcHlgDP5ivyqYIn7RxIXM13eWwOzcokWosSauz6RbU=	374027	726	White Noise	NaN	NaN	-1.0

图 1.3 歌曲表部分数据

可以发现，genre\_ids/artist\_name/composer/lyricist 这几个字段都是类别型变量，且值都存在不同的分隔符，代表了多个值。

统计各个字段缺失值的比例，发现 composer 和 lyricist 缺失值较多。

#### (2) language 数据分布

统计 language 数据的总体分布，结果如图 1.4 所示。

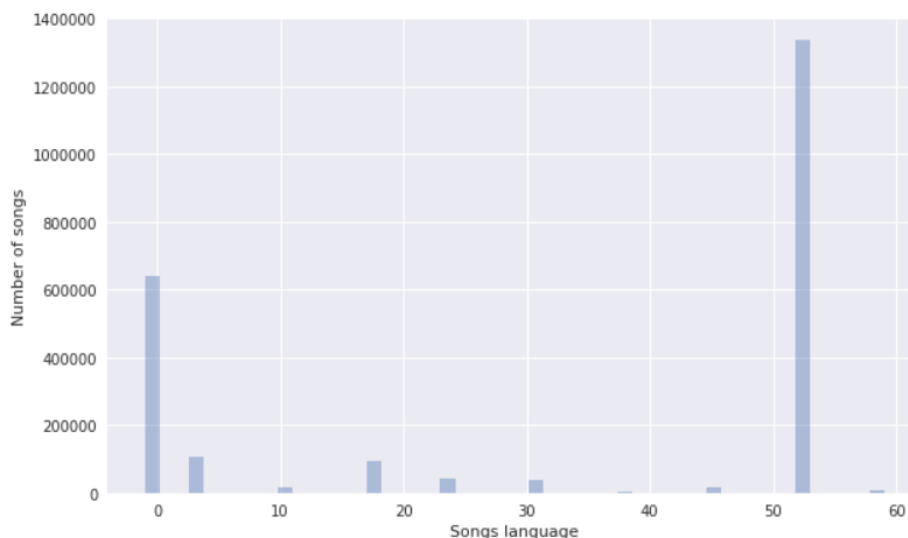


图 1.4 language 数据分布

可以发现歌曲在 language 上的分布不均衡，多数集中于 0 附近和 50+附近的语言类型。

### (3) song length 数据分布

统计 song length 的极值，可以发现最短的歌曲只有 1.393 秒，最长的竟然有 12173 秒（202 分钟）。画出 length 数据的直方图如图 1.5 所示，可以发现大多数歌曲处在正常的长度范围（100s 到 1000s 之间），只有少数是极短和极长的歌曲。

找到这条长度最长的 song，通过在搜索引擎中搜索歌手名，发现此歌手的音乐风格是 software，大概判断这首歌是舒缓型歌曲，符合歌曲长度长的特点。因此不作为异常点继续分析。

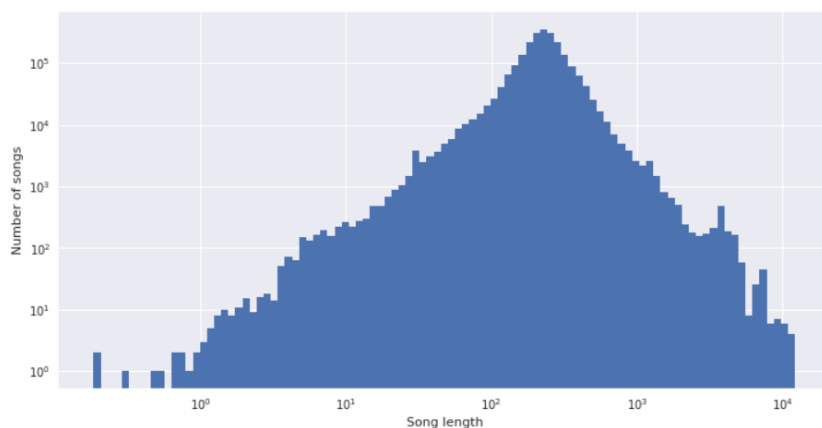


图 1.5 song length 数据分布

### 1.3.3 训练集

#### (1) 总体信息

训练集包含 7377418 条数据，数据量非常庞大。统计各个字段的缺失值，结果如下。

msno	0
song_id	0
source_system_tab	24849
source_screen_name	414804
source_type	21539
target	0

可以发现 `source_screen_name` 缺失值很多，达到 40 万+。

统计训练集中 `target` 为 0 和为 1 的样本数量，如图 1.6 所示，可见训练集中不同类别的样本分布均衡。

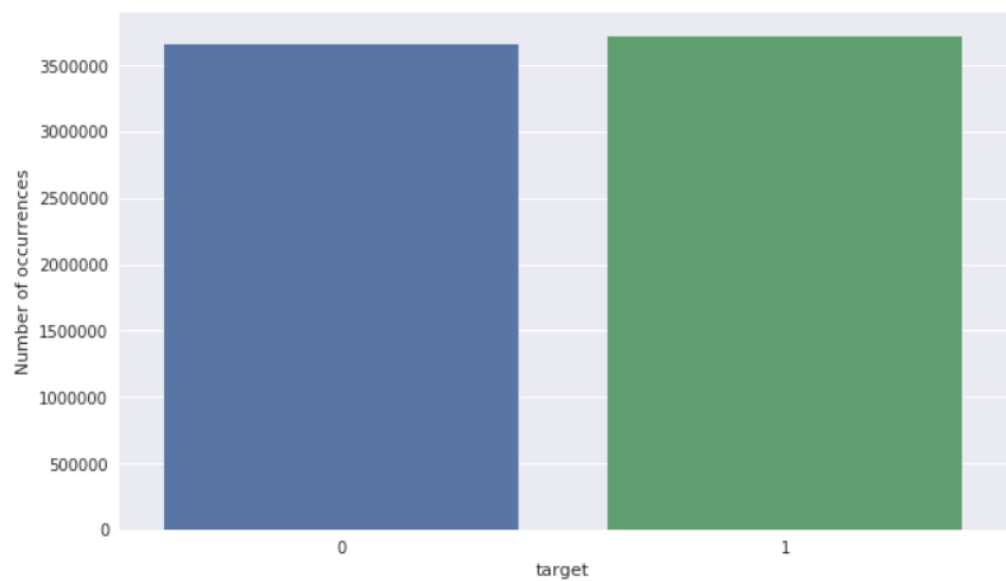


图 1.6 训练集中不同标签样本分布

#### (2) `source_system_tab` 字段

统计类别型变量 `source_system_tab` 字段的分布情况，结果如图 1.7 所示。

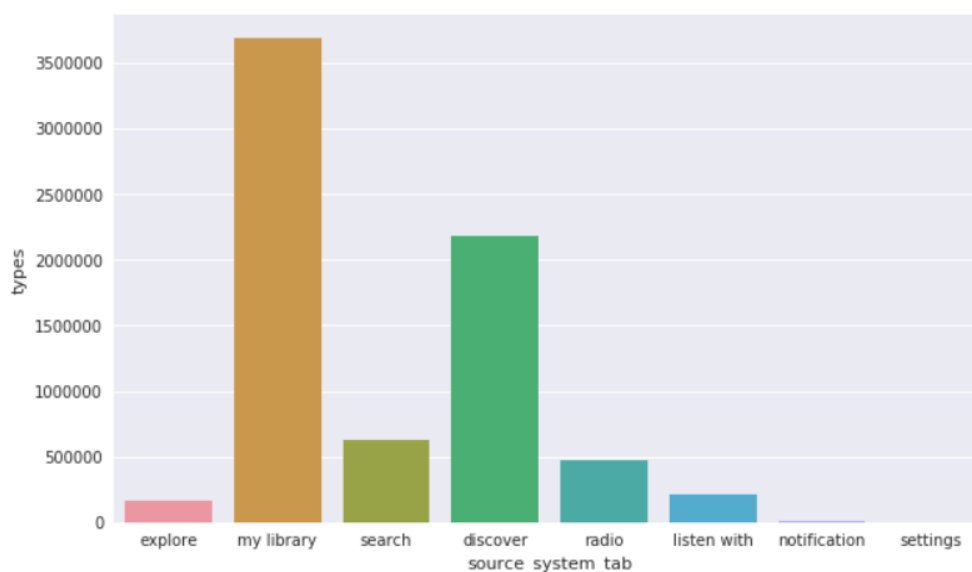


图 1.7 训练集 source\_system\_tab 数据分布

从上图看出训练集 source\_system\_tab 字段取值有 8 种，其中 mylibrary 最多，setting 值比较少。

### (3) source\_screen\_name 字段

统计 source\_screen\_name 字段的分布情况，结果如图 1.8 所示。

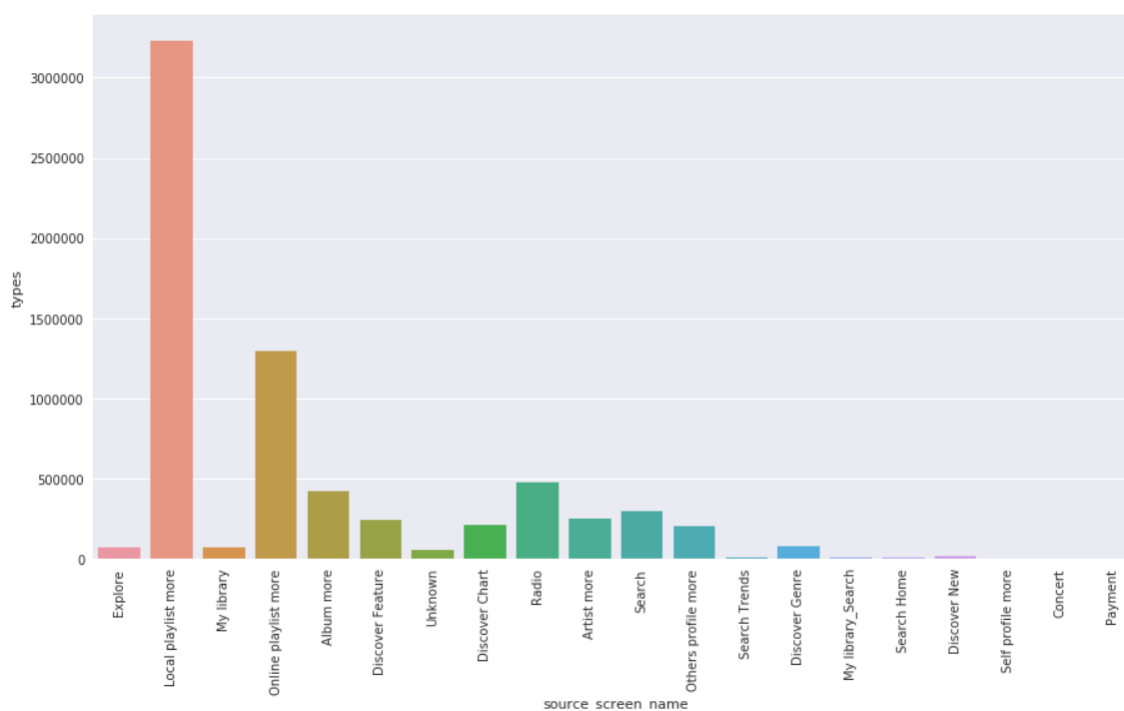


图 1.8 训练集 source\_screen\_name 数据分布

从上图看出训练集上 source\_screen\_name 字段取值共有 20 种，其中 Local playlist more 最多，Payment 最少。花钱听的歌，重复次数少符合常理。

#### (4) source\_type 字段

统计 source\_type 字段的分布情况，结果如图 1.9 所示。

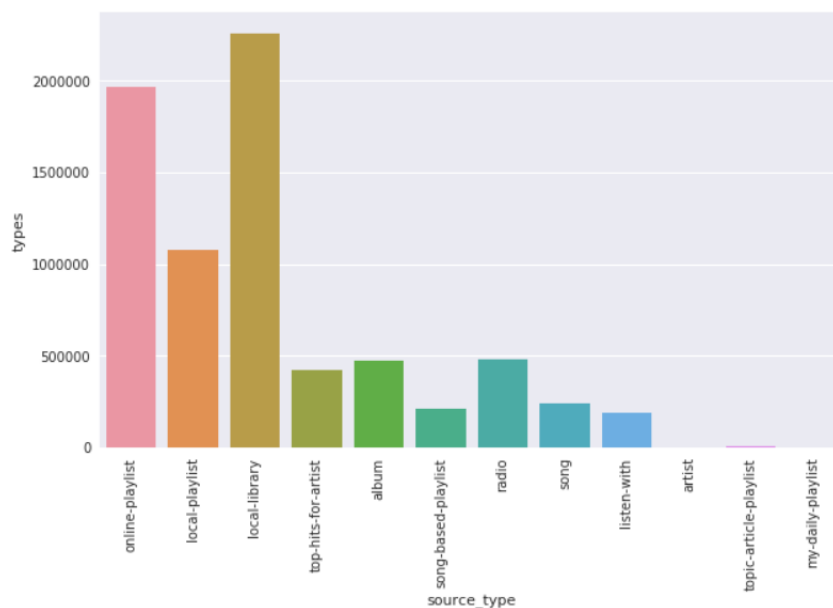


图 1.9 训练集 source\_type 数据分布

从上图看出训练集上 source\_type 字段取值共有 12 种，其中 local-library 最多，my-daily-playlist 最少。

### 1.3.4 测试集

测试集包含 2556790 条数据，除 ID 外其他都是类别型变量。

#### (1) source\_system\_tab 字段

统计该字段的分布情况，结果如图 1.10 所示。

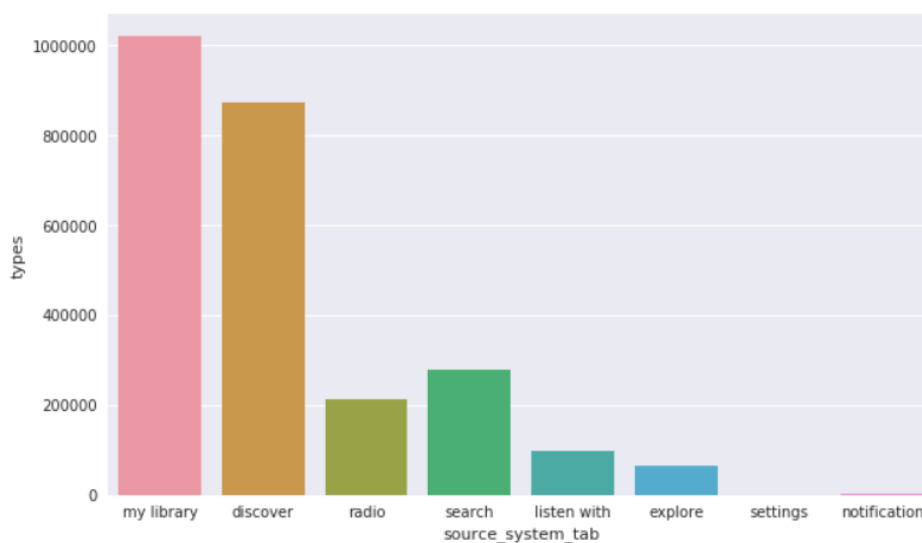


图 1.10 测试集 source\_system\_tab 数据分布



总体上仍旧是 my library 类型的数据量最多。

## (2) source\_screen\_name 字段

统计该字段的分布情况，结果如图 1.11 所示。

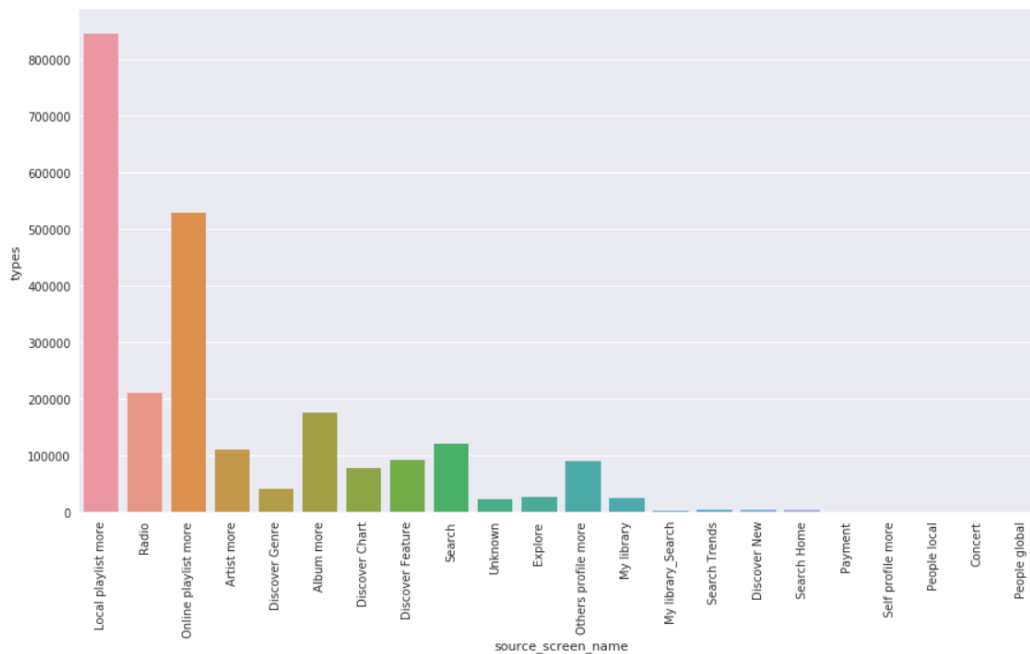


图 1.11 测试集 source\_screen\_name 数据分布

## (3) source\_type 字段

统计该字段的分布情况，结果如图 1.12 所示。

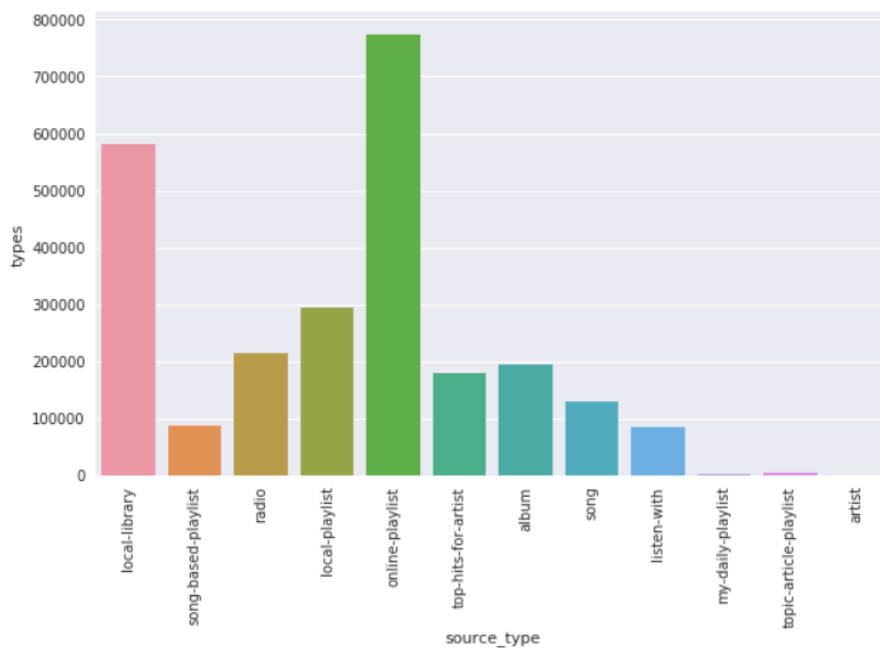


图 1.12 测试集 source\_type 数据分布

## 1.4 数据统计

### 1.4.1 无关数据统计与清理

Songs 和 song\_extra 中，存在很多歌曲和用户在训练集和测试集中都没有出现，属于无关数据，清除这部分数据可以大大减小数据量。

清除无关数据后，songs 表剩下 419781 条数据（清除前为 2296320 条）。Song\_extra 剩下 419661 条 songs 额外信息（原数据有 2295971 条信息）。

### 1.4.2 统计冷启动数据

针对用户表，统计出现在用户表中但没出现在训练集的用户数量，有 3648 个，占总用户数的 10.6%，属于新用户。

针对歌曲表，统计出现在歌曲表中但没出现在训练集的歌曲数量，有 59867 个，占总各歌曲数的 14.26%，属于新歌曲。

针对歌曲表中 artist 数据，统计出现在歌曲表中但没出现在训练集的数量，有 5790 个，占总数的 12.48%。

### 1.4.3 统计歌曲被重复收听的规律

训练集中 target=0 代表歌曲被收听过一次，target=1 代表用户在第一次收听歌曲后一个月内再次收听。训练集中某首歌曲被听取的总次数为 plays，某首歌曲 target=1 的样本个数为它被重复收听的次数 repeated\_plays。repeated\_plays 和 plays 的比值，代表歌曲被重复收听的概率 repeat\_chance。

统计整个训练集所有歌曲 plays 和 repeat\_chance 的关系，曲线如图 1.13 所示。

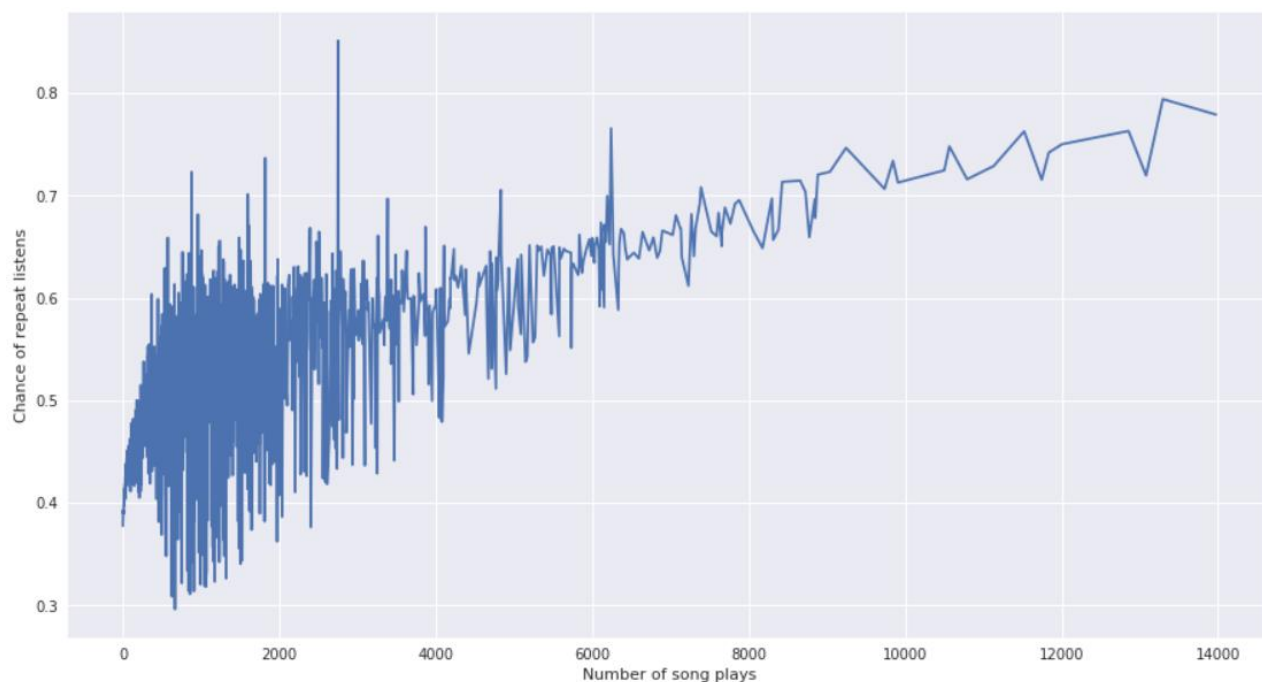


图 1.13 训练集歌曲 plays 和 repeat\_chance 关系图

横轴代表播放次数 plays，纵轴代表该播放次数下的重复播放概率。可以发现，大体上遵循播放次数越多，重复播放概率越大。但也有一些歌曲播放次数少，重复播放概率大。

#### 1.4.4 缺失值和异常值处理

各个表中都存在一定的缺失值和异常值，处理策略如下：

songs 表中 language 字段的缺失值，通过演唱者其他歌曲的 language 进行填充；songs 表中 genre\_ids 字段的缺失值用 0 进行填充；artist\_name 的缺失值用‘no\_artist\_name’进行填充，composer、lyricist 的填补策略相似。

members 表中 bd 字段存在一些异常值（小于 0 和大于等于 80 的），对该部分数值用 nan 进行填充，并增加一个 bd\_missing 新特征来反映这种情况；gender 字段的缺失值用‘other’填充。

训练集和测试集中 source 相关的三个字段，缺失值采用相应字段的众数进行填充。

#### 1.4.5 个别字段针对性处理

(1) songs 表的 genre\_ids 字段

该字段代表歌曲流派，一般会存在一首歌有多个流派，用'|'分割的情况，因此首先通过分隔符对字段数据进行分割，得到前三个流派信息以及总的流派个数，由此产生 first\_genre\_id、second\_genre\_id、third\_genre\_id、genre\_id\_cnt 四个新特征，抛弃原有的 genre\_ids 特征。

(2) songs 表的 artist、lyricist、composer 字段

这些字段代表演唱、作词、作曲信息，也存在一首歌对应多个演唱、作词、作曲的情况，因此采用相应的分隔符进行分割，统计每首歌对应的歌手数目、作词数目和作曲数目，形成新特征，同时只保留第一个歌手、作词和作曲者。

统计歌手、作词、作曲人数与歌曲个数的分布关系，结果如图 1.14-1.16 所示。

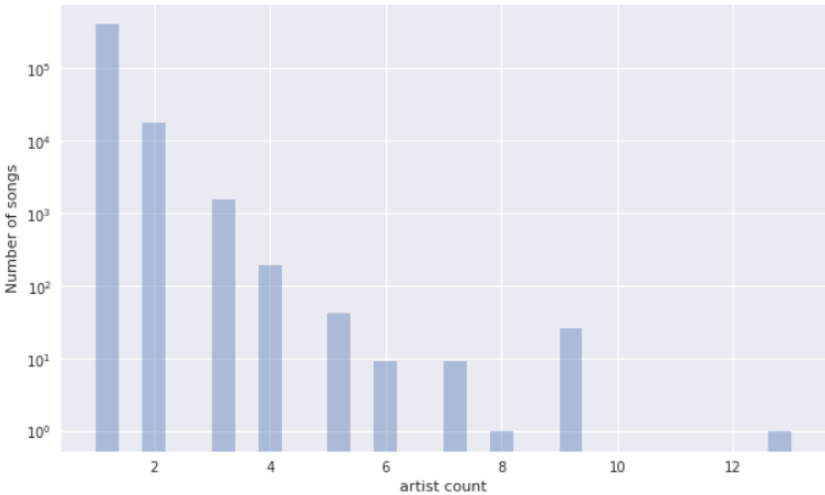


图 1.14 歌手数和歌曲数的关系图

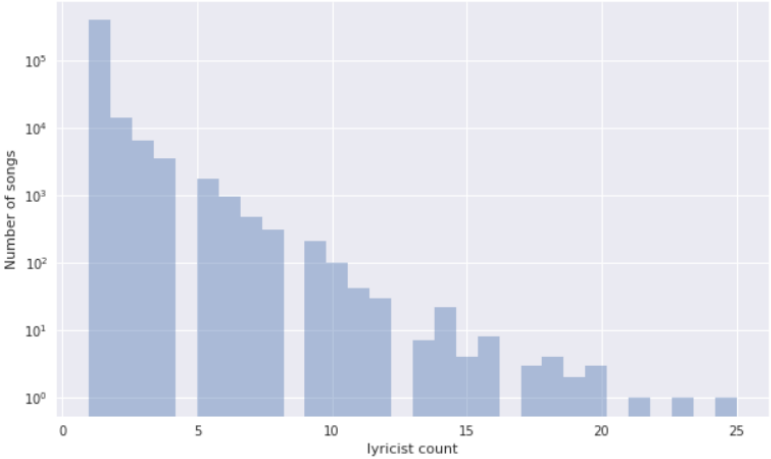


图 1.15 作词人数量和歌曲数的关系图

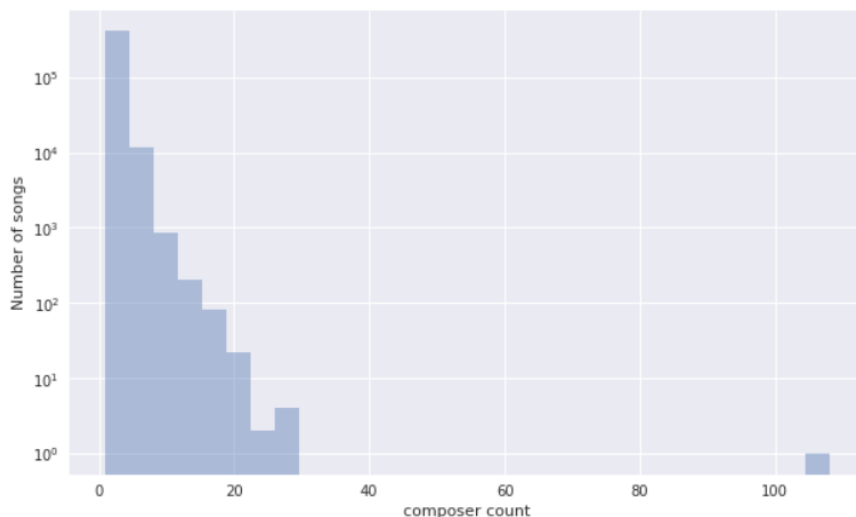


图 1.16 作曲人数量和歌曲数的关系图

从分布情况看出，作曲人数目方面，大致都集中在 30 以下，有一首歌存在 100 多个作曲人，可以作为离群点或者噪声处理。

### (3) songs\_extra 表中的字段处理

该表中 isrc 字段中包含了歌曲国家、出版以及年份等信息，通过对该字段进行分割，得到歌曲的年份信息并处理成 4 位年份形式。另外新增加一个特征代表该字段是否缺失。

## 1.4.6 时序特性

根据歌曲的年份信息，以 50000 的窗口大小，对训练集和测试集中从前到后的 2017 年歌曲出现的频率进行统计，结果如图 1.17 所示。

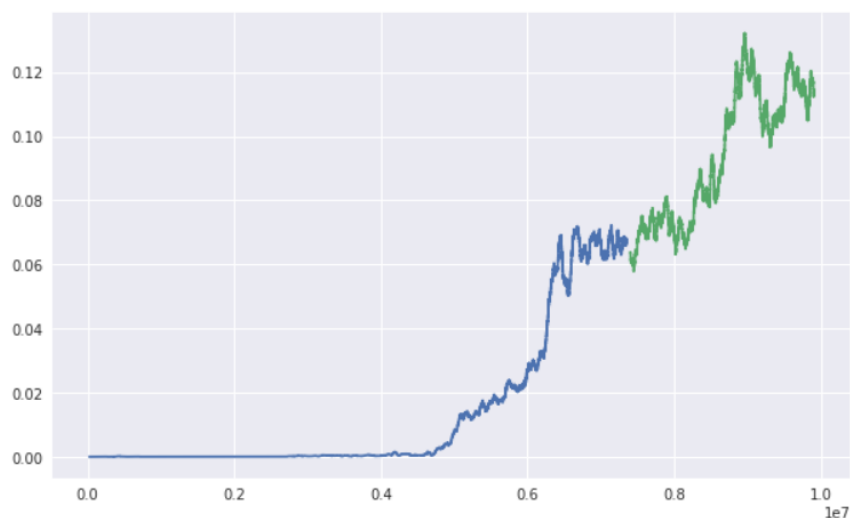


图 1.17 歌曲年份在训练集和测试集中的分布图

上图中横轴是在数据集中的索引，纵轴是 2017 年歌曲在窗口中出现的频率。蓝色部分是训练集的曲线，绿色部分是测试集的曲线。可以看出，训练集和测试集都是越往后 2017 年的歌曲出现频率越高，同时测试集和训练集的曲线衔接较为流畅。可以得出结论，训练集和测试集的样本具有一定的时序顺序。因此后续考虑选择训练集末尾的 10%-20% 的数据作为验证集，数据量是 70 万到 150 万之间。

### 1.4.7 统计用户和其他信息的相关规律

#### (1) 用户和歌曲播放次数的关系

统计训练集和测试集每个用户播放歌曲的次数，结果如图 1.18 所示。

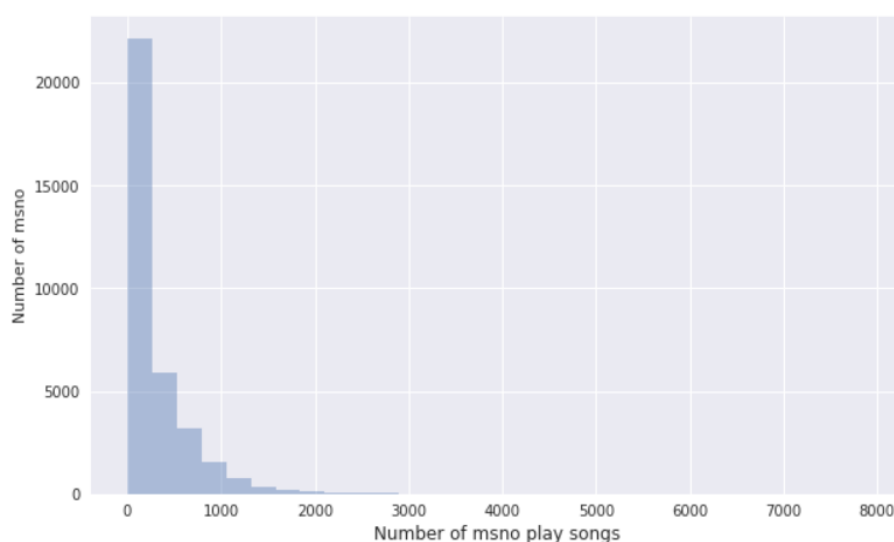


图 1.18 用户播放歌曲次数分布图

图中横坐标是每个用户播放的歌曲数目，纵坐标是用户数量。可以看出这是个长尾分布，大部分人都集中在 0-1000 之间，考虑加 log 变换。

#### (2) 分解字段

用户注册时间 `registration_init_time` 原来是一个 `yyyymmdd` 日期，将其拆成年、月、日，并作为三个新特征加入到 `members` 中；同时将用户会员到期时间 `expiration_date` 也拆成年、月、日，作为三个新特征加入到 `members` 中。

#### (3) 统计用户会员时长分布

通过会员到期时间减去用户的注册时间，计算得出会员时间 membership\_days 作为一个新特征加入到用户表中。统计不同会员时间下的用户人数分布，结果如图 1.19 所示。

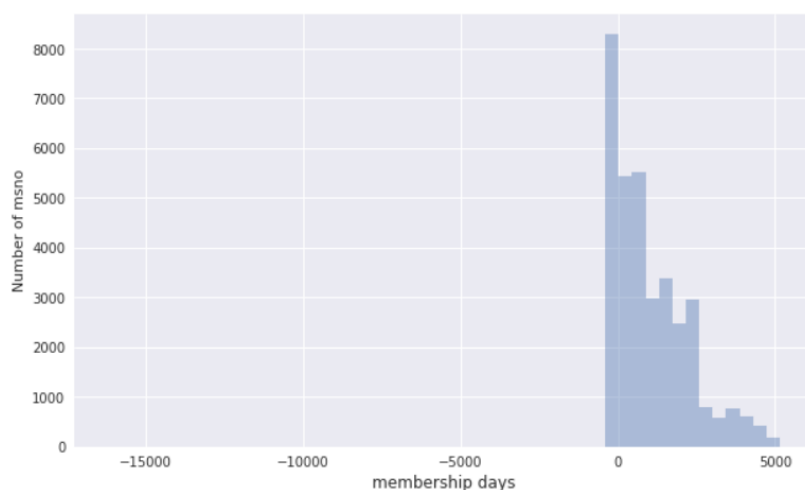


图 1.19 会员时长分布图

图中横轴是会员时长，纵轴是对应的用户个数。由于存在一个过期时间小于注册时间的异常值，因此在时长上存在负值。观察大于 0 的部分，可以发现基本符合现实规律，即大部分用户的会员时长较短，随着时长增加用户数逐渐下降。

#### (4) 统计用户注册年份和播放歌曲数目的关系

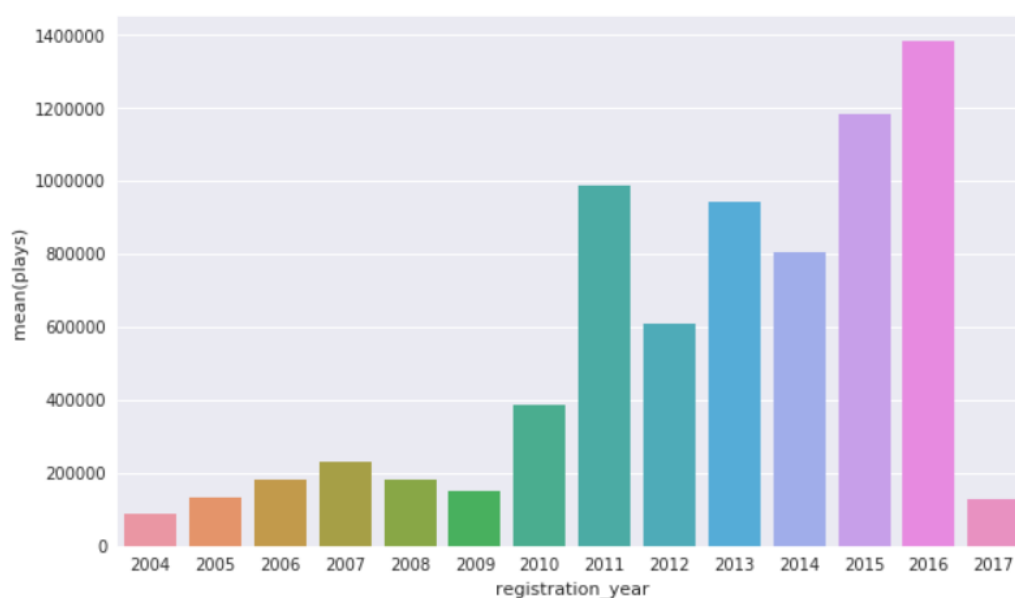


图 1.20 用户注册年份和播放歌曲数目关系图

图 1.20 横轴是注册年份，纵轴是播放次数。可以看出随着时间增加，播放次数增加；2017 年的次数很少，可能是因为这个训练集是对 2017 年 2 月份之前的播放做统计，因此是合理的。

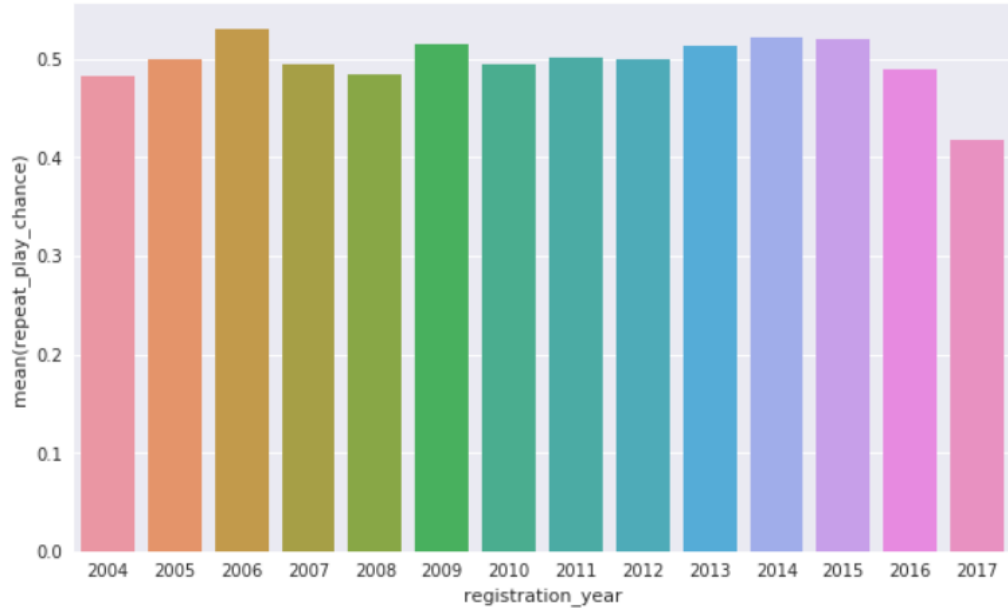


图 1.21 注册年份和歌曲重复播放概率关系图

图 1.21 横轴是用户注册的年份，纵轴是在训练集中当前年份歌曲被重复播放的概率。可以看到概率与时间没有必然的联系，这符合规律。



## 2 特征工程

### 2.1 引言

本章在第一章数据探索的基础上，进行特征工程方面的工作，重点在于以音乐方面的背景知识为依托，分析数据集原始特征之间的关联，从而抽取隐含信息提取新的特征，对数据集在特征维数上进行扩展。

### 2.2 原始类别型特征的处理

数据集中包含很多的类别型特征，主要包括：`msno`、`song_id`、`source_system_tab`、`source_screen_name`、`source_type`、`city`、`gender`、`registered_via`。采用 `sklearn` 中的 `LabelEncoder` 方法对这些特征进行编码。

### 2.3 songs 表和 song\_extra\_info 表的特征处理

#### 2.3.1 流派特征的处理

`songs` 表中的流派特征 `genre_ids`，包含了采用‘|’分隔符分割的多种流派信息，在处理中通过分割得到前三个流派，保存为 `first_genre_id`、`second_genre_id`、`third_genre_id`，以及总的流派个数 `genre_id_cnt`。

得到新的特征之后，对三个流派特征进行 `LabelEncoder` 编码，同时抛弃原有的 `genre_ids` 特征。

#### 2.3.2 歌手、作词、作曲特征的处理

在数据探索中发现，歌曲存在有多个歌手、作词、作曲人的现象，因此统计了每首歌曲的歌手数目 `artist_cnt`、作曲人数目 `composer_cnt`、作词人数目 `lyricist_cnt` 作为新的特征，另外根据歌曲是否具有多个歌手，产生新的特征 `is_featured`。

在完成以上处理后，对原有的歌手 `artist_name`、作词 `lyricist`、作曲 `composer` 特征，只保留第一个人的信息，并对特征进行 `LabelEncoder` 编码。

另外针对每个歌手，统计了每个歌手演唱的歌曲数目，产生 `artist_song_cnt` 特征；针对每个作曲人，统计编曲数目，产生 `composer_song_cnt` 特征；针对每个作词人，统计作词数目，产生 `lyricist_song_cnt` 特征；针对歌曲流派，统计每个流派对应的歌曲数目，产生 `genre_song_cnt` 特征。

针对歌曲、歌手、作词、作曲、流派和用户的关系，统计了每首歌被播放的用户数目 `song_rec_cnt`，每个歌手被收听的用户数目 `artist_rec_cnt`，每个作曲人被收听的用户数目 `composer_rec_cnt`，每个作词人被收听的用户数目 `lyricist_rec_cnt`，每个流派被收听的用户数目 `genre_rec_cnt`。

### 2.3.3 song\_extra 字段处理

该表中的 `isrc` 字段是一种国际通用的编码，包含了国家、出版者、年份等信息，因此根据编码规范，可以从中提取出 `cn`、`xxx`、`year` 特征，分别表示歌曲的国家、出版者、年份信息，并对三个新特征进行 `LabelEncoder` 编码。另外根据 `isrc` 是否缺失，增加 `isrc_missing` 特征。

针对 `cn`、`xxx`、`year`，分别统计对应的歌曲数目，产生 `cn_song_cnt`、`xxx_song_cnt`、`year_song_cnt` 特征；分别统计对应的用户数目，产生 `cn_rec_cnt`、`xxx_rec_cnt`、`year_rec_cnt` 特征，代表有多少用户收听了 `cn`、`xxx`、`year` 对应的歌曲。

### 2.3.4 数值型特征的处理

对于数值型特征，如 `song_length`、`song_rec_cnt` 等，一般存在长尾分布的现象，为了降低较大取值的影响，对这些特征进行 `log1p` 变换。

### 2.3.5 歌曲播放次数及歌曲被重复播放的概率

统计数据集中每首歌曲出现的次数，即为其被播放的总次数，产生新特征 `plays`；统计数据集中每首歌曲对应 `target` 为 1 的数量，即为其被重复播放的次数 `replay_counts`，计算它和 `plays` 的比值，即可得到代表重复播放概率的新特征 `repeat_play_chance`。

### 2.3.6 歌曲热度特征

数据集中的样本分布具有一定的时序性，从前往后分别统计固定时间窗口的某个歌曲出现的次数，其变化规律可以体现歌曲在数据集中固定时间段内热度的变化规律。分别采用 10、100、1000、5000、20000 的窗口大小，统计歌曲热度，得到 `song_10_before_cnt`、`song_10_after_cnt` 等特征。

从前往后，统计每个时间之前歌曲的出现次数，其变化规律可以体现歌曲在数据集中总的热度的变化规律，得到 `song_till_now_cnt` 特征。

对以上新特征进行 `log1p` 变换。

## 2.4 用户表数据处理

### 2.4.1 用户的相应特征

统计每个用户 `id` 下的样本数量，即可得到代表用户收听歌曲数目的新特征 `msno_rec_cnt`，并对该特征进行 `log1p` 变换。

根据用户的注册时间 `registration_init_time`，会员到期时间 `expiration_date`，可以得到代表会员时长的新特征 `membership_days`。另外注册时间和过期时间进行拆分，可以得到 `registration_year`、`registration_month`、`registration_date`、`expiration_year`、`expiration_month`、`expiration_date` 特征。

对用户收听来源方面的三个特征 `source_system_tab`、`source_screen_name`、`source_type` 特征进行 one-hot 编码；针对单个特征，统计该特征对应的用户总数以及每种取值对应的用户数，计算二者比值可以得到每种取值的概率，得到一组新的特征。根据每种取值的概率，计算得到用户在收听某个歌曲时来源信息的后验概率，得到一组后验概率特征。

### 2.4.2 用户-歌曲以及用户-歌手关系

根据用户是否收听过某个歌曲，可以构建用户-歌曲关系矩阵；根据用户是否收听过某个歌手演唱的歌曲，可以构建用户-歌手关系矩阵。

对用户-歌曲关系矩阵进行 SVD 分解，取奇异值个数为 48，得到奇异值矩阵 `S`、代表用户对歌曲 48 个隐含特征偏好的 `U` 矩阵，以及歌曲对 48 个隐含特

征拥有成都的 V 矩阵，从而得到一组新的特征 `member_component_x` 和 `song_component_x`（x 取值为 0 到 47）。计算数据集中每个样本中该用户对应的 `member_component_x*S*song_component_x`，得到代表该样本中用户对歌曲喜爱程度的新特征 `song_embeddings_dot`。

对用户-歌手关系矩阵进行 SVD 分解，取奇异值个数为 16，得到奇异值矩阵 S、代表用户对歌手 16 个隐含特征偏好的 U 矩阵，以及歌手对 16 个隐含特征拥有成都的 V 矩阵，从而得到一组新的特征 `member_artist_component_x` 和 `artist_component_x`（x 取值为 0 到 15）。计算数据集中每个样本中该用户对应的 `member_artist_component_x*S*artist_component_x`，得到代表该样本中用户对歌手喜爱程度的新特征 `artist_embeddings_dot`。

### 2.4.3 用户活跃度

数据集中的样本分布具有一定的时序性，从前往后分别统计固定时间窗口的某个用户出现的次数，其变化规律可以体现用户在数据集中固定时间段内活跃度的变化规律。分别采用 10、100、1000、5000、20000 的窗口大小，统计用户活跃度，得到 `msno_10_before_cnt`、`msno_10_after_cnt` 等特征。

从前往后，统计每个时间之前用户的出现次数，其变化规律可以体现用户在数据集中总的活跃度的变化规律，得到 `msno_till_now_cnt` 特征。

对以上新特征进行 `log1p` 变换。

## 2.5 基于特征交叉的隐含特征提取

基于 pandas 的 `groupby` 方法，分别统计同一个用户 id 下的 `song_length`（歌曲长度）、`artist_song_cnt`（歌手活跃度）、`artist_rec_cnt`（歌手热度）、`song_rec_cnt`（歌曲热度）、`yy`（歌曲年份）、`language`（歌曲语言）六种特征的均值和标准差，可以得到用户对这六种特征的偏好。如某用户 id 对应的所有歌曲长度的均值，代表了用户对歌曲长度的偏好，标准差代表了该偏好的稳定度；同理用户 id 对应的所有歌手热度的均值，代表了用户对热门歌手的偏好程度。

一个用户可能在数据集中多次出现，针对该用户在数据集中的某条样本，

统计该用户在该样本之前上一次出现时对应的歌曲 `id`、来源信息、歌曲时间等信息，以及在该样本之后第一次出现时对应的上述信息，可以有效表征用户的兴趣变化。

一首歌曲在数据集中一般也会多次出现，每条样本对应了不同的播放来源。统计某首歌曲和某种来源信息在数据集中配对出现的次数，可以得到隐含在数据中的歌曲来源特征。

## 2.6 各表数据整合

完成以上的特征工程之后，各表都出现了一些新的特征。最终的训练将只会用到 `train` 和 `test`，因此需要基于 `msno` 和 `song_id`，分别将 `songs` 表和 `members` 表中的数据整合进 `train` 和 `test` 表中。

## 3 传统机器学习模型与训练

### 3.1 引言

在完成第二章的特征工程后，数据集的特征维数扩充到了 296 个。本章在此基础上，主要研究传统机器模型在数据集上的表现与性能，确定最终所采用的模型以及相应的参数。在模型方面，主要尝试了 LogisticRegression、SVM、RandomForest 和 LightGBM 四种模型。由于样本数据量非常大(700 多万)，特征维数有 296 维，用全量数据很难完成训练，因此在训练中根据模型的实际情况采用，逐步增加数据量的方式，逼近能够使用的最大数据量和最好结果。

### 3.2 LogisticRegression 模型

LogisticRegression 的特点是实现简单，速度很快，存储资源低，还可以添加 L1 或者 L2 正则化。

(1) 以 10%的训练数据进行训练，设置正则参数  $C=10$ ，penalty 选择 L1，solver 选择 saga;

(2) 以 10%的训练数据进行训练，设置正则参数  $C=10$ ，penalty 选择 L2，solver 选择 liblinear。

利用以上训练好的模型对验证集进行预测。根据预测结果和验证集的标签，计算出相应 AUC 的分值都在 0.5323 左右。从结果可以看出，由于训练数据的特征维数很高并且样本数量庞大，逻辑回归的性能不是很好，容易欠拟合。

### 3.3 SVM 模型

Sklearn 中 SVM 有三种方式实现分类，LinearSVC、带核函数的 SVC 和基于随机梯度下降的 SGDClassifier。由于样本数很多，所有选择 LinearSVC 做模型训练。

以 5%的训练数据进行模型训练和调参，penalty 为 l2 情况下调试正则参数 C 的值。选择参数最优的模型对验证集进行预测，根据预测结果和验证集的标

签，计算出相应 AUC 的分值都在 0.5053 左右。从结果看出，SVM 的性能一般。从原理上来说，LinearSVC 跟 LogisticRegression 差不多，但随着训练样本数目增加，对计算和存储资源的需求也快速增长，所以不适合数据量大的样本。

### 3.4 RandomForest 模型

随机森林是多重决策树的组合，能有效的处理大数据集和高维特征的样本，同时处理分类和数值特征。

以 10% 的训练数据进行模型训练和参数调优，通过调试基学习器数目，树的深度 max\_depth、max\_features（最大特征数目）、min\_samples\_leaf 等，得到一个最佳模型。通过此模型对验证集进行预测，根据预测结果和验证集的标签，计算出相应 AUC 的分值都在 0.6759 左右。

从结果看出，AUC 的分值比 LR 和 SVM 提高了很多，说明随机森林模型针对大数据集还是有比较好的效果。但是算法复杂，计算成本高，训练时间很漫长。

### 3.5 LightGBM 模型

LightGBM 使用基于直方图的算法，所有训练速度更快且内存占用低，准确率高，能够处理大数据集。涉及的超参数个数多，因为叶子优先的分裂方法，所以重点关注 num\_leaves 和 min\_data\_leaf 调优过程。

#### 3.5.1 模型训练过程

参数调优方面，通过在验证集上的训练选择最优的超参数，然后以最优参数在所有的训练集中再次训练模型，得到最终用来预测的最优模型。详细训练过程如下：

- 1、选择验证集上 AUC 分数最高的参数进行训练，对应的验证集上的 AUC 为 0.76442

- 2、对训练集 70%-100% 位置的数据进行训练，得到模型 A。基于模型 A 对测试集进行预测，将预测结果提交到 Kaggle 上，如图 3.1 所示 public 分数是

0.70386，排名为第 37 位。






35	Mikhail_Kamalov		0.70576	58	2y
36	k1414st		0.70484	73	2y
37	ximibbb		0.70385	101	2y
38	Hungry Cat		0.70383	47	2y
39	cybertails		0.70355	53	2y

图 3.1 LightGBM 模型 A 的排名

3、对训练集 50%-70%位置的数据进行训练，得到模型 B

4、对训练集 30%-50%位置的数据进行训练，得到模型 C

5、对训练集 10%-30%位置的数据进行训练，得到模型 D

考虑到四个模型应该具有一定的互补性，因此可以将四个模型进行融合，融合策略如图 3.2 所示。

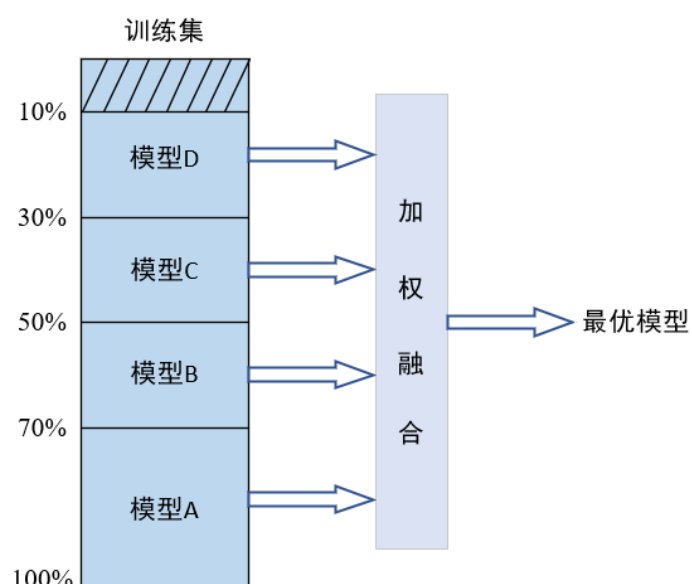


图 3.2 LightGBM 模型融合策略

在融合方法上，采用的是加权平均。首先尝试了四个模型权重相等的平均方法，在测试集上的分数为 0.70290，低于模型 A 的分数。因此需要提高模型 A 的比重，根据四个模型的数据在训练集上的时间先后，最终确定的四个模型权重分别为 A 模型 0.6，B 模型 0.2，C 模型和 D 模型 0.1。以此权重进行加权平均，如图 3.3 所示在测试集上的分数为 0.70987，排名为第 22 位。








20	Lab418		0.71034	91	2y
21	ifengc		0.71001	23	2y
22	mim-solutions.pl		0.70954	69	2y
23	Ye Jingyi		0.70911	139	2y
24	NikitaLukashev		0.70883	108	2y

图 3.3 LightGBM 融合模型的排名

### 3.5.2 结果分析

根据 LightGBM 的训练结果，输出特征的重要性如图 3.4 所示。

	name	importance
1	timestamp	8640
2	msno_artist_name_prob	5504
3	msno_till_now_cnt	5022
4	song_embeddings_dot	4807
5	msno_50000_before_cnt	4647
6	msno_50000_after_cnt	4640
7	msno_source_prob	4341
8	msno_xxx_prob	4198
9	song_source_screen_name_prob	4153
10	song_source_type_prob	4146
11	msno_source_type_prob	4136
12	song_id	4111
13	song_length	3927
14	msno	3917
15	msno_timestamp_std	3757
16	msno_year_prob	3756
17	song_source_system_tab_prob	3740
18	msno_source_screen_name_prob	3622
19	repeat_play_chance	3578
20	msno_source_system_tab_prob	3531
21	membership_days	3511
22	member_component_44	3486
23	member_component_28	3479

图 3.4 LightGBM 模型输出的模型重要性排序

可以看出，排在前 20 的特征大多都是特征工程产生的新特征，也从侧面验

证了特征工程的重要性和有效性。

另外从四个模型的预测准确性上，可以看出训练集上数据的时序性确实对数据的重要性有一定影响，越往靠近测试集的时间区间，数据越重要；但早期数据同样能够在某些程度上对模型的性能进行弥补，所以四个模型的加权融合能够取得更好的性能。

## 4 神经网络模型

### 4.1 引言

本章主要研究基于多类特征交叉融合的神经网络模型构建、参数调优和模型融合，重点是选择合适的特征进行拼接和融合，提高神经网络模型的性能。

### 4.2 网络总体结构

#### 4.2.1 基于 field-aware 的特征融合

在网络结构设计中，主要依据 field-aware 思想：即数据集中的不同特征主要从属于不同的 field，如有些特征属于用户特征，有些特征属于歌曲特征，有些特征主要为包含上下文信息的上下文特征，因此可以将特征划分为 member field、song field 和 context field 三个 field。同一个 field 内部的所有特征经过 Embedding 处理和标准化处理之后拼接在一起，经过一个全连接网络，其输出作为该 field 的输出。不同 field 的特征经过多种方法的拼接后，最终进入一个包含多层全连接层的分类网络并得到最终的分类结果。

##### (1) User Field 网络构建

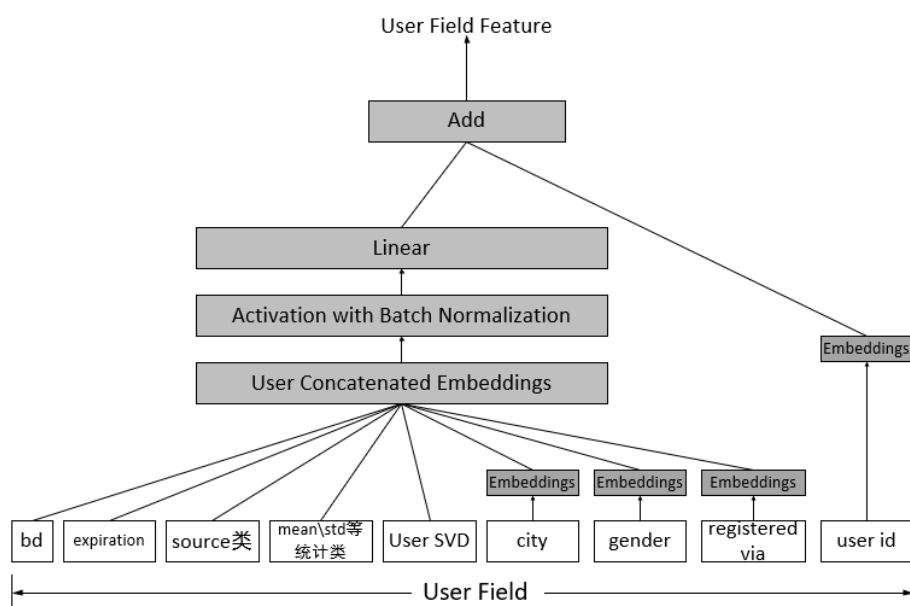


图 4.1 User Field 网络结构图

网络结构如图 4.1 所示。对 User Field 中的 city、gender、registered via 和 user id 特征采用 Keras 中的 Embedding 层进行编码，权重可以和整个网络同步训练；经过 Embedding 之后，除 user id 外，其余特征经过 concat 拼接作为输入，经过一个可选 BatchNormal 的全连接层和简单全连接层，输出和经过 Embedding 的 user id 特征进行 add 运算，最终得到 User Field Feature。

## (2) Song Field 网络构建

网络结构如图 4.2 所示。对 Song Field 中的 artist、composer、lyricist 等特征采用 Keras 中的 Embedding 层进行编码，编码输出和其余特征进行 concat 拼接后，经过一个可选 BatchNormal 的全连接层和简单全连接层，最终得到 Song Field Feature。

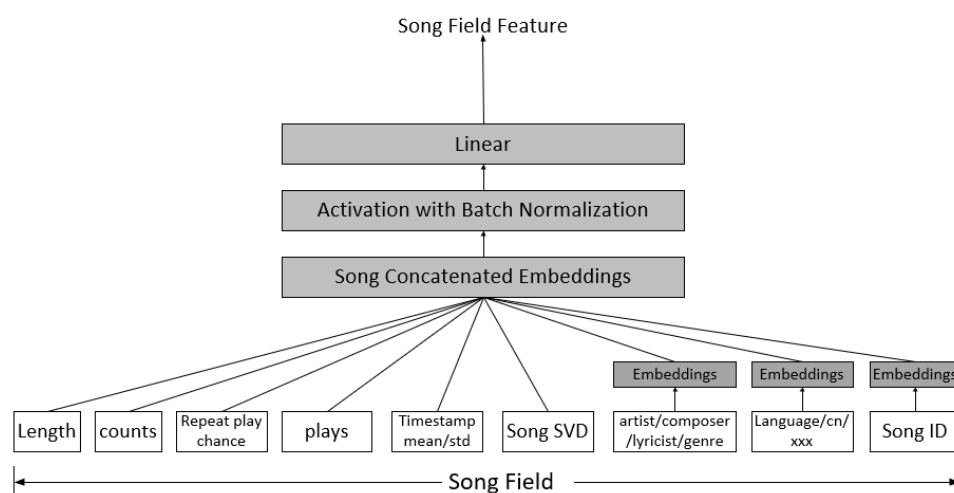


图 4.2 Song Field 网络结构图

## (3) Context Field 网络构建

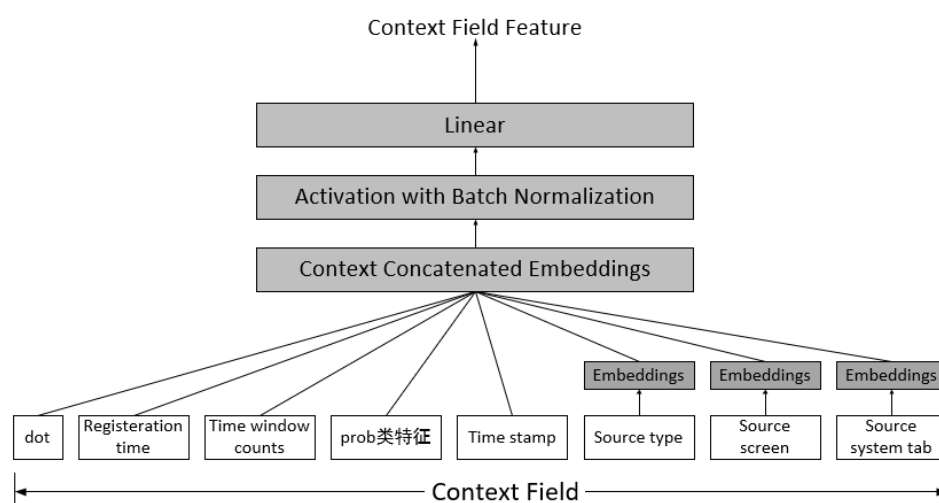


图 4.3 Context Field 网络结构图

网络结构如图 4.3 所示。对 Context Field 中的 Source type、Source screen、Source system tab 等特征采用 Keras 中的 Embedding 层进行编码，编码输出和其余特征进行 concat 拼接后，经过一个可选 BatchNormal 的全连接层和简单全连接层，最终得到 Context Field Feature。

#### 4.2.2 基于多维特征融合的分类网络

不同 field 的特征，首先进行包含 dot、concat 等操作的融合，之后作为输入送入一个分类网络，该分类网络的结构借鉴了 DenseNet 的多维特征融合思想，结构如图 4.4 所示：User Field 的特征和 Song Field 的特征经过 Dot 运算之后得到 Dot 输出，然后将 Dot 输出和三个 Field 的输入进行 concat 操作，作为分类网络的输入。

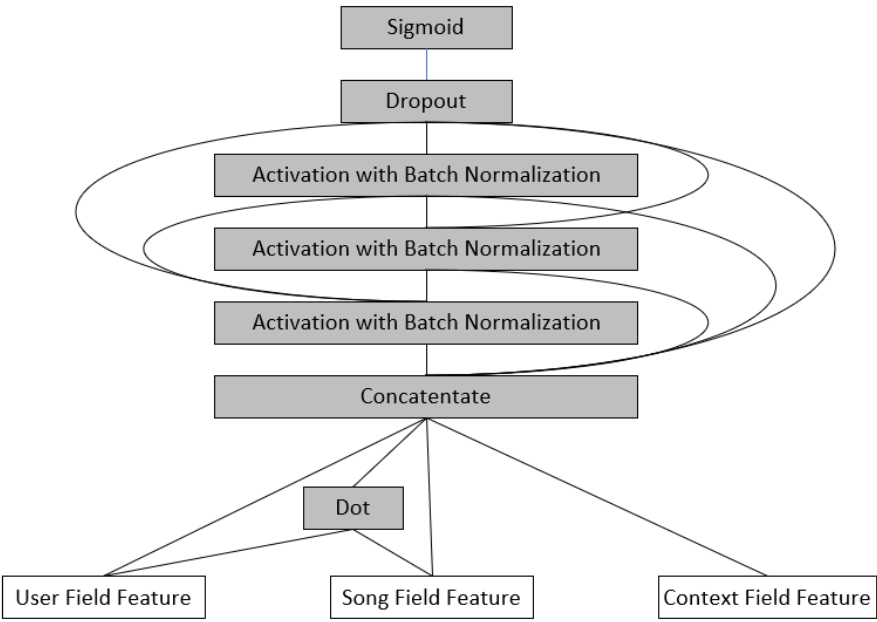


图 4.4 分类网络结构图

### 4.3 模型训练

训练使用的 GPU 为 Tesla V100，显存为 16G。整个网络需要确定的超参数主要包括：Embedding 层的输出维数、全连接层的激活函数、正则参数、是否包含 Batch Normalization、初始学习率、学习率衰减比例。

在训练时，为了获得以上超参数的最优值，采用策略为：多次训练，每次训练时在一定范围内随机初始化以上超参，训练 40 个 epoch，保存该组参数以

及验证集上最佳的 AUC 分数。由于时间和计算力的限制，本文一共进行了 35 组不同参数的训练，得到了 35 组超参数。最后根据在验证集上的 AUC 降序排序，选择一定数量的参数在整个训练集上进行训练。

### 4.4 模型融合

虽然模型结构一样，但是不同的超参数仍旧可以训练得到不同性能的模型，各个模型在验证集上的 AUC 不同，但是模型之间存在一定的互补性，可以通过融合多个模型提高模型的泛化能力。本文主要尝试了不同个数的模型融合在测试集上的预测效果，结果如表 4.1 所示。

表 4.1 不同个数的模型融合效果

模型数量	测试集上的 AUC
35	0.70585
18	0.70735
16	0.70889
12	0.71075
8	<b>0.71127</b>
5	0.70975

由于不同参数的模型 AUC 不同，因此当模型数量过多的时候差的模型会降低整个模型的性能；但是模型数量过少又无法有效起到融合的作用，因此最佳的融合数量是 8 个模型，在测试集上的 AUC 达到了 0.71127，在 LeaderBoard 上排名为第 19 名。






17	MrOoijer		0.71179	145	2y
18	rob_med		0.71164	29	2y
19	adorable zombies		0.71041	86	2y
20	Lab418		0.71034	91	2y
21	ifengc		0.71001	23	2y

图 4.5 融合之后神经网络模型的排名

从结果上看，神经网络模型的泛化能力优于 LightGBM 模型，在测试集上取得了更好的效果。

# 5 LightGBM 和神经网络融合

## 5.1 引言

在完成了两种模型内部的融合之后，本章主要研究 LightGBM 和神经网络模型的最佳融合方法。

## 5.2 融合方法

考虑到神经网络模型的性能略高于 LightGBM，因此在融合时神经网络模型应该具有更高的权重，不过本文仍旧尝试了不同权重下的融合效果；另外考虑到可能最优的两个模型融合不一定能够得到最优的效果，因此本文尝试了其 他非最优模型的融合。

实现结果如表 5.1 所示，其中 LightGBM 模型采用的是最优模型即 AUC 为 0.70987 的模型。

表 5.1 不同融合方法的效果

神经网络模型	AUC	权重（LightGBM: NN）	融合后 AUC
18 个模型	0.70735	6:4	0.71756
		5:5	<b>0.71854</b>
		3:7	0.71804
16 个模型	0.70889	3:7	0.71807
		5:5	<b>0.71860</b>
		7:3	0.71624
8 个模型	0.71127	5:5	0.71898
		4:6	<b>0.71965</b>
		3:7	0.71950
		2:8	0.71823

从上表可以看出，融合 8 个模型的神经网络模型性能最高，融合后取得的 AUC 也最高，达到了 0.71965，在 LeaderBoard 上排名为第 9 位。

# 6 项目感想与收获

## 6.1 引言

本章主要阐述自己在整个项目过程中遇到的一些问题，产生的一些感想和收获。

## 6.2 遇到的问题

### 6.2.1 超大的数据集规模造成的影响

700 万+的训练集规模和 200 万+的测试集规模，是整个课程学习过程中从未见到过的。超大规模的数据集在整个项目过程中造成的影响也是贯穿始终，主要有以下两个方面：

#### （1）对硬件要求极高

最初是在自己的电脑上做，结果发现由于内存不够，Macbook Air 连数据集都读不出来，因此决定在云端做项目，尝试过百度云和阿里云，性能都差不多，但是最多的时候都需要用到 12 核、32G 内存才能够完成某些数据处理。

#### （2）数据处理和训练耗时长

即使是 32G 内存，在做特征工程时，某些数据处理操作也要耗费五六个小时的时间，甚至有时会由于内存溢出直接报错；LightGBM 训练时，5000 轮的训练需要三四个小时（200 万-300 万数据），且无法直接用全量数据；神经网络模型由于可以单次只处理一个 Batch 数据，因此可以用全量数据，使用 Tesla V100 训练一个 epoch 需要 3-5 分钟，一个模型训练 40 个 epoch 大概需要 1 到 2 个小时。

### 6.2.2 算法实现中遇到的问题

剩下的问题主要是自己编码能力不足和对理论知识理解不够造成在项目实践过程中走了一些弯路，主要包含以下方面：

（1）在做 LabelEncoder 编码时，如果要编码的字段里存在 NAN 或者填补缺失值时用了不同的类型（比如 int64 不小心加了引号）都会报错；



(2) 参数调优时由于数据量大无法用 `gridSearchCV`，需要对所有参数逐个进行调试，效率比较低；

(3) 在特征维数或者数据量多的时候，查看数据信息会出现显示不全的问题，之后通过 `pandas` 的 `display` 方法解决；

(4) 模型训练完成后，对测试集做预测时，由于数据量比较大导致内存不足。之后采用少量多次的方法进行预测，再对多次预测结果进行拼接；

(5) 做了某些操作后导致 `dataframe` 中数据索引不连续，需要做 `reset_index` 操作；

(6) 用神经网络训练模型时，发现 `train_loss` 一直是 `nan`，`train_acc` 直接到 0。在检查了代码之后没发现问题，后来查出来是因为数据里存在空值；

(7) 模型训练过程中，发现 `train_loss` 一直在下降，但是 `train_acc` 基本不变，保持在 50% 左右。后来查出来是模型设计有问题，特征分类不合适。

## 6.3 项目过程中的收获和感想

最大的收获就是在整个项目过程中，完整的实现了推荐系统中的数据探索、特征工程、模型选择和训练过程，在大量的实际动手操作中收获了很多单纯理论学习无法获取的经验，对理论知识的体会也更加深刻，同时提高了编码能力。

最大的感想就是自己依旧还是个小白，对于推荐系统方向常用的方法、经典思想和模型仍旧不甚了解，还需要多多进行实际比赛和项目的锻炼，才能够获得更大的提高。

最后感谢整个课程过程中 CSDN 的老师、班班和助教对我的帮助，祝训练营越办越好！