《机器学习》 - 数据预处理



华为技术有限公司

目录

[1 实验介绍 2](#_Toc66344286)

[1.1 实验目的 2](#_Toc66344287)

[1.2 实验清单 2](#_Toc66344288)

[2 华为云使用指导 3](#_Toc66344289)

[2.1 简介 3](#_Toc66344290)

[2.2 华为云账号申请与实名认证 3](#_Toc66344291)

[2.2.1 账户用途 3](#_Toc66344292)

[2.2.2 注册步骤 3](#_Toc66344293)

[2.2.3 申请思路 6](#_Toc66344294)

[2.3 Access Key和Secret Access Key的获取 6](#_Toc66344295)

[2.3.1 简介 6](#_Toc66344296)

[2.3.2 AK和SK如何生成 7](#_Toc66344297)

[2.4 华为云常用产品入口 9](#_Toc66344298)

[2.4.1 对象存储服务OBS 9](#_Toc66344299)

[2.5 ModelArts开发平台环境准备 13](#_Toc66344300)

[3 数据预处理实验 21](#_Toc66344301)

[3.1 实验介绍 21](#_Toc66344302)

[3.2 实验目的 21](#_Toc66344303)

[3.3 实验步骤 21](#_Toc66344304)

[3.3.1 数据预处理 21](#_Toc66344305)

[3.3.2 思考题 33](#_Toc66344306)

[3.3.3 数据预处理 - 分类数据 34](#_Toc66344307)

[3.4 实验小结 41](#_Toc66344308)

# 实验介绍

数据的质量涉及多个方面，包括准确性、完整性、一致性、时效性、可解释性等。在真实数据中，往往包含了大量缺失值，噪声，以及人为因素导致的误差或异常点；数据的质量，决定了模型预测和泛化能力的上限。而数据特征太少，不足以描述数据，造成偏差过高；特征太多，一是增大计算成本，二是造成维度灾难（方差过高导致过拟合）。因此在训练模型之前需要数据预处理与特征工程。

本章实验分别使用不同的方法进行数据预处理。

本章实验难度包含：初级

初级实验：数据预处理实验

## 实验目的

掌握数据预处理的常用方法

## 实验清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验 | 简述 | 难度 | 软件环境 | 开发环境 |
| 数据预处理 | 基于google store中各个app下载的信息数据，进行数据预处理 | 初级 | Python3 | 本地PC |

# 华为云使用指导

## 简介

华为云提供了非常丰富的云资源以及云服务，可以满足开发者的需要。本手册主要聚焦于如何申请华为云上的相应资源搭，主要包括了：账号申请与实名认证，AK和SK的获取，华为云主要服务介绍。

## 华为云账号申请与实名认证

### 账户用途

1.用于在华为云官网开通相应服务，确保课程实验部分顺利进行。

2.用于体验华为云提供的各种便捷便宜的云服务，包括深度学习，机器学习，图像识别，语音识别，自然语言处理等。

### 注册步骤

进入华为云官网

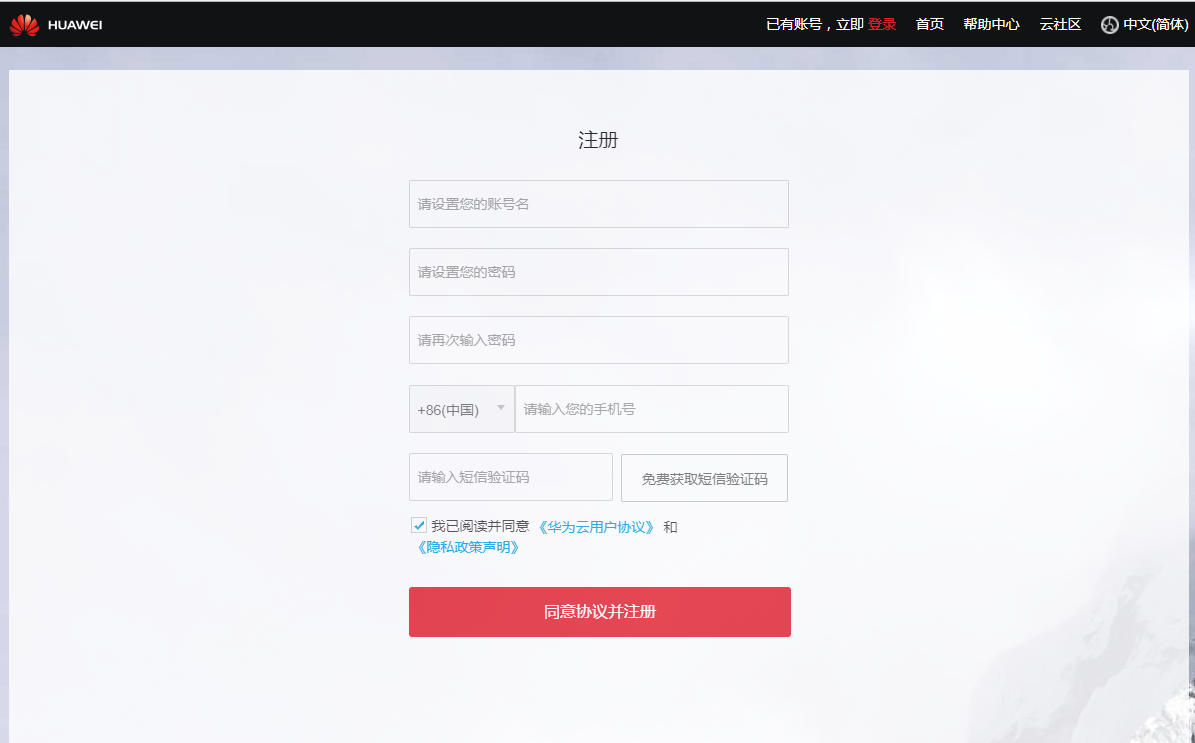
进入华为云官网：<http://www.huaweicloud.com/>，结果如下：



华为云官网

点击首页右上角“注册”

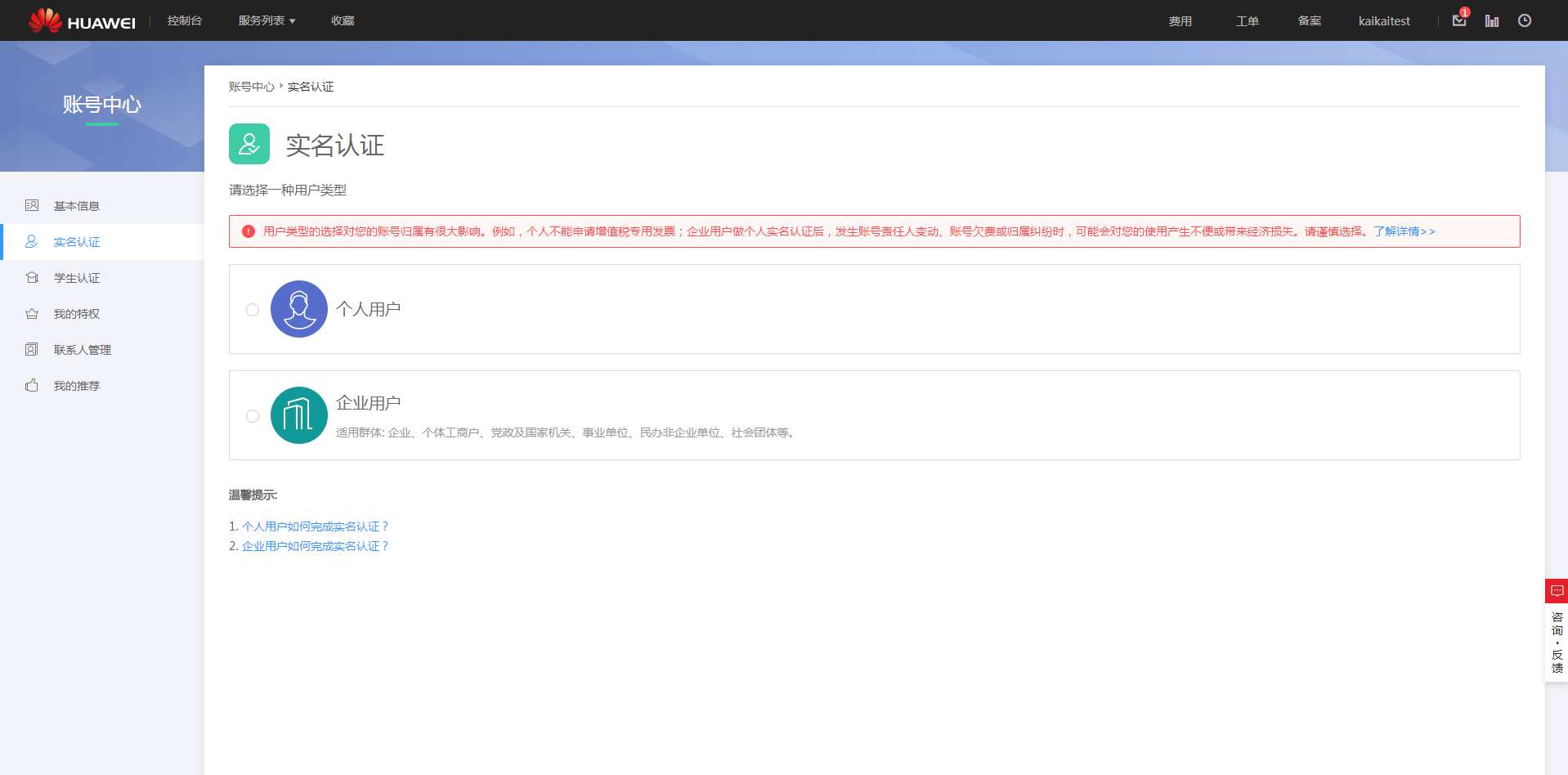
点击右上角的“注册”按钮，进行账号注册，结果如下：



注册账号

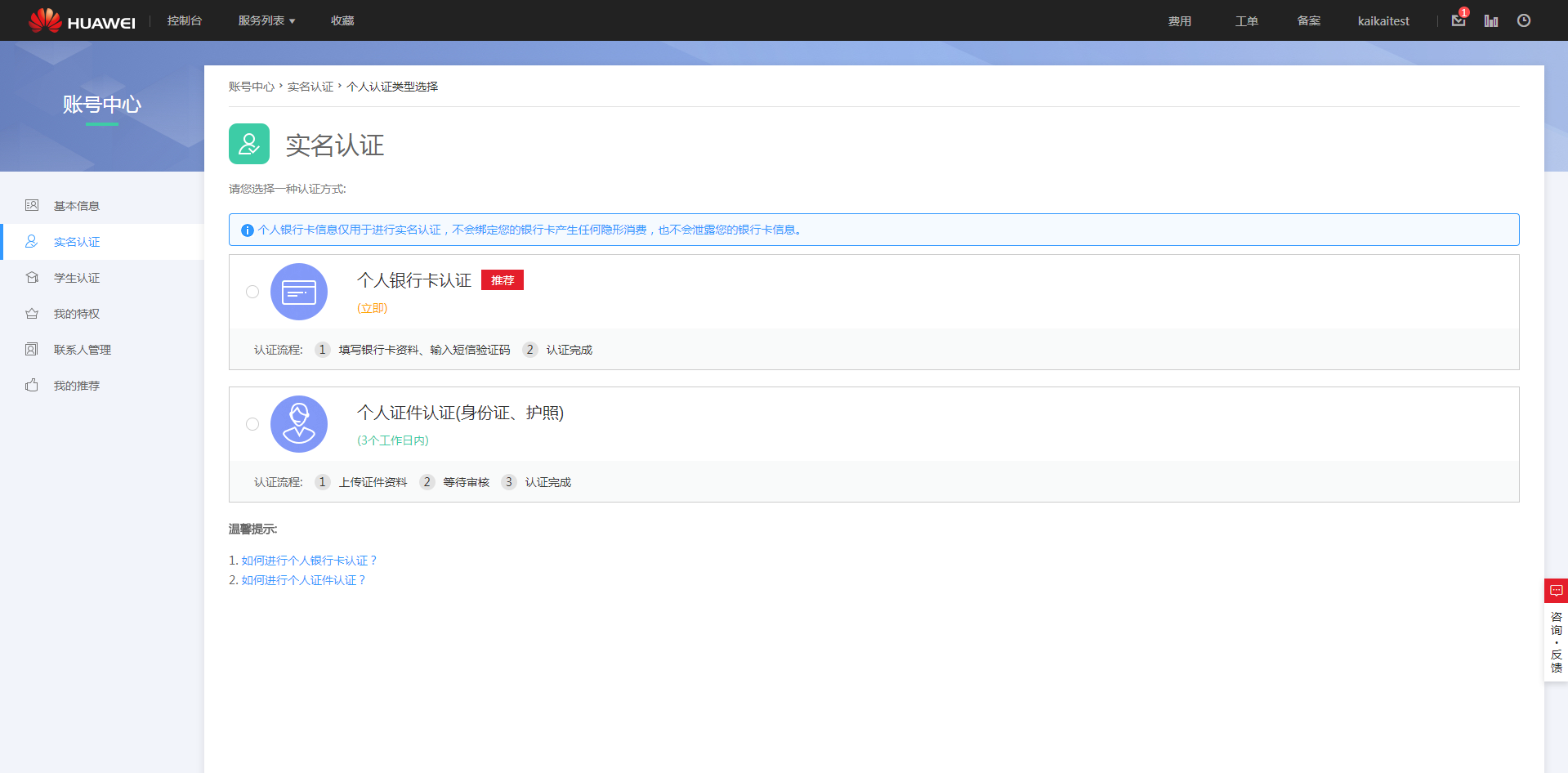
实名认证

点击“实名认证”按钮进行账户的实名认证，选择“个人用户”进入下一项：



实名认证

点击“个人银行卡认证”进入下一项：



个人银行卡实名认证

填写相关信息，并点击“提交认证”确认：



提交认证

至此，已完成实名认证。可进入后续环节。

### 申请思路

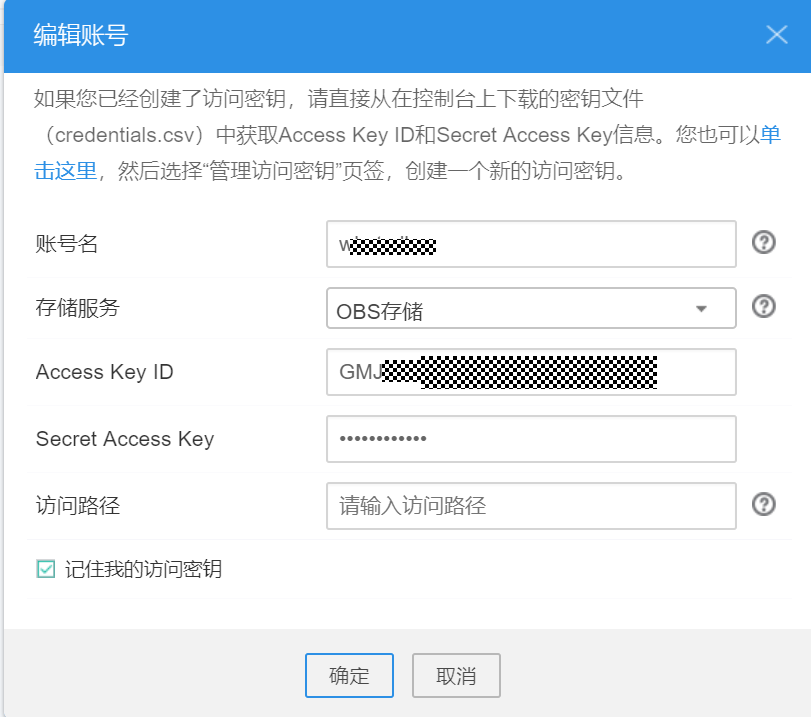
1.华为云提供了大量的云服务，用户可结合自身的任务与需求申请相应的云资源进行相关的操作。

2.云服务的具体申请，可按照对应页面的选项以及自身需求进行相应配置搭建。

## Access Key和Secret Access Key的获取

### 简介

AK和SK是华为云中使用各项服务所需要的“身份代码”，当第一次使用华为云的某些服务时，都会遇到需要输入AK和SK密码的情况，如下图所示：



AK、SK认证页面

图中的Access Key ID就是AK，Secret Access Key就是SK。这是在使用OBS服务时，必须有AK和SK的输入才可以使用。

### AK和SK如何生成

登录好已经注册的华为云账号后，可以从主页的右上角进入账号中心。如下图所示：



进入账号中心

进入账号中心主页后，可以进入“管理我的凭证”。



管理我的凭证

在左侧的列表中，点击“访问秘钥”。



访问密钥

进入访问秘钥后，就可以通过点击“新增访问秘钥”来添加自己的访问秘钥了，也可以通过右侧的操作来编辑、停用、删除访问秘钥。



新增访问密钥

**注意：**新增访问秘钥后会自动下载一个表格文件，该文件存有华为云的访问秘钥，注意保存。

## 华为云常用产品入口

### 对象存储服务OBS

对象存储服务（Object Storage Service）是一款稳定、安全、高效、易用的云存储服务，具备标准Restful API接口，可存储任意数量和形式的非结构化数据。

打开华为云官网并点击OBS服务



OBS入口

OBS服务是华为云中最常用的服务，通常都会搭配ModelArts服务一起使用。

点击管理控制台，进入服务。



进入管理控制台

方式一：使用OBS Browser+上传下载OBS文件

（1）下载OBS Browser+ 并安装：[https://storage.huaweicloud.com/obs/?locale=zh-cn&region=cn-north-4#/obs/manager/buckets](https://storage.huaweicloud.com/obs/?locale=zh-cn&region=cn-north-4" \l "/obs/manager/buckets)



下载OBS Browser+

（2）打开并使用AK/SK登录OBS Browser+



登录OBS Browser+

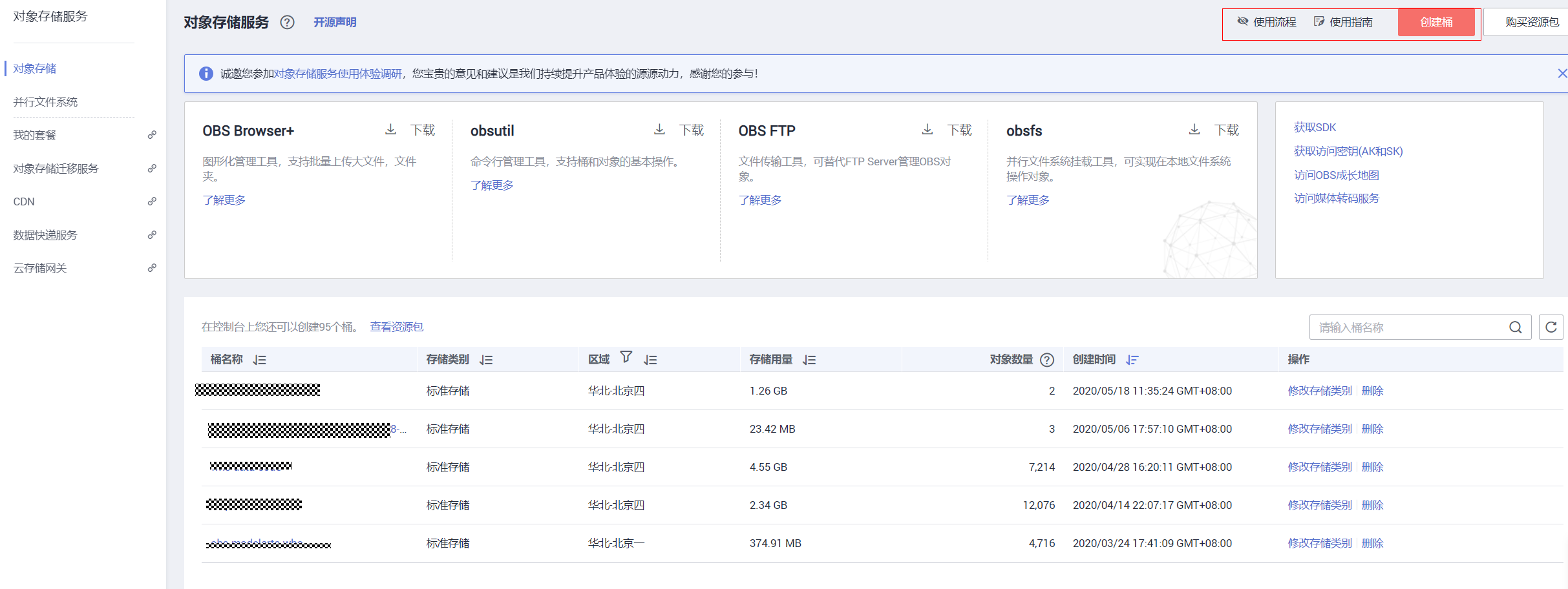
（3）登录后即可创建桶以及在桶内创建文件夹



创建桶

方式二：使用OBS页面来上传下载文件

（1）点击右上角的创建桶，来创建自己的OBS桶。



创建桶

（2）桶的具体配置情况，可以由自己设定，区域自行选择，建议选择“北京四”，桶名称自定义，储存类别为标准储存。桶策略请自行定义，如果需要别人也读取你的文件，请选择“公共读”，如果只供自己使用，建议选择“私有”，之后点击“立即创建”。



配置桶参数

（3）之后，点击进入新创的桶，点击“对象”，进行数据的存储。



管理对象

（4）接下来就可以上传文件或者创建新的文件夹。



上传对象

## ModelArts开发平台环境准备

ModelArts是面向开发者的一站式AI开发平台，为机器学习与深度学习提供海量数据预处理及半自动化标注、大规模分布式Training、自动化模型生成，及端-边-云模型按需部署能力，帮助用户快速创建和部署模型，管理全周期AI工作流。

登录华为云，进入ModelArts控制台



ModelArts平台入口



点击进入控制台

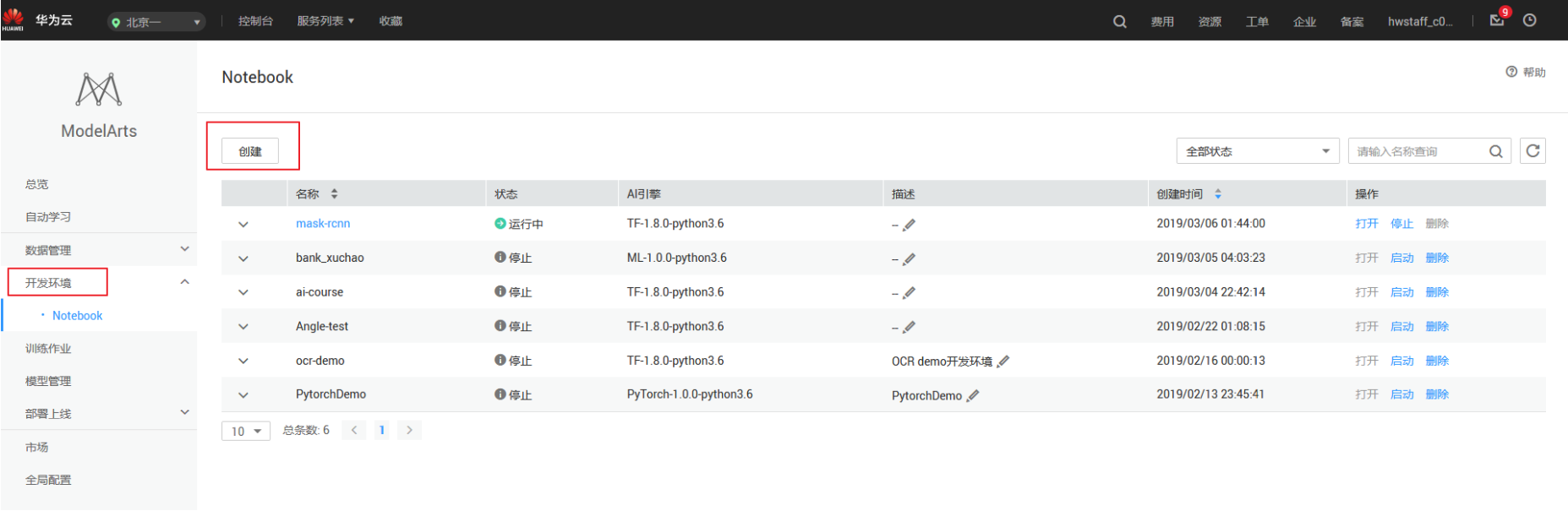


ModelArts主页面

创建ModelArts Notebook

ModelArts Notebook提供网页版的Python开发环境，可以方便的编写、运行代码，并查看运行结果。

（1）在ModelArts服务主界面依次点击“开发环境”、“创建”。首次使用时，可能需要验证AK/SK，按照之前获取的秘钥进行认证即可。



创建ModelArts Notebook

（2）填写Notebook所需的参数，例如：



填写Notebook参数

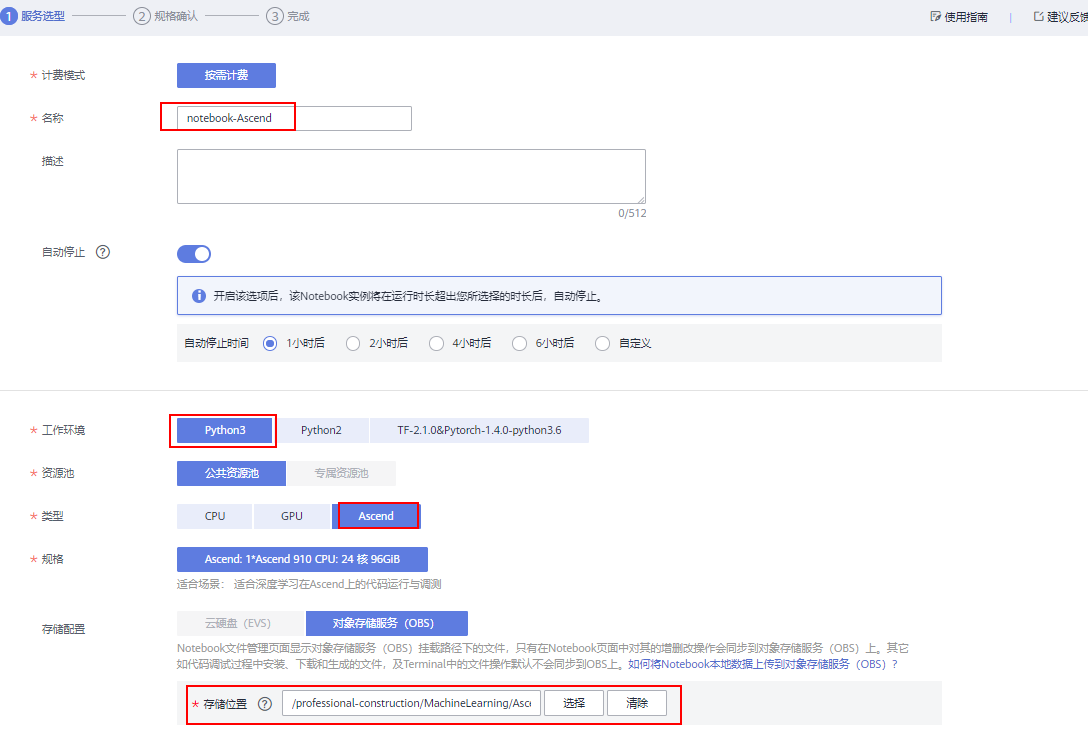
Notebook参数说明

|  |  |
| --- | --- |
| 参数名称 | 说明 |
| “计费方式” | 按需计费。当前仅支持按需计费，无需修改。 |
| “名称” | Notebook的名称。只能包含数字、大小写字母、下划线和中划线，长度不能超过64位且不能为空。 |
| “描述” | 对Notebook的简要描述。 |
| “自动停止” | 默认开启，且默认值为“1小时后”，表示该Notebook实例将在运行1小时之后自动停止，即1小时后停止计费。  开启自动停止功能后，可选择“1小时后”、“2小时后”、“4小时后”、“6小时后”或“自定义”几种模式。选择“自定义”模式时，可指定1~24小时范围内任意整数。 |
| “工作环境” | 当前支持2种工作环境，分别为“Python2”和“Python3”，不同工作环境其对应可使用的AI引擎不同，详细支持列表请参见[支持的AI引擎](https://support.huaweicloud.com/engineers-modelarts/modelarts_23_0033.html" \l "modelarts_23_0033__section191109611479)。  如果需要使用TensorFlow 2.X、PyTorch 1.4.0或者R语言版本的AI框架，则需要选择“TF-2.1.0&Pytorch-1.4.0-python3.6”的工作环境。如果选择此类型的工作环境，暂时无法使用免费规格，建议选择其他规格。  每个工作环境多种AI引擎，可以在同一个Notebook实例中使用所有支持的AI引擎，不同的引擎之间可快速、方便的切换，并且有独立的运行环境。您可以在Notebook实例创建完成后，进入Jupyter页面创建对应AI引擎的开发环境。  **说明：**  ModelArts还支持Keras引擎，详细说明请参见[ModelArts是否支持Keras引擎？](https://support.huaweicloud.com/modelarts_faq/modelarts_05_0042.html" \t "_blank) |
| “资源池” | 可选公共资源池和专属资源池，关于ModelArts专属资源池的介绍和购买，请参见[资源池](https://support.huaweicloud.com/engineers-modelarts/modelarts_23_0076.html)。 |
| “类型” | 支持CPU、GPU和Ascend类型。GPU性能更佳，但是相对CPU而言，费用更高。Ascend类型为公测资源，请提前完成[Ascend 910公测申请](https://console.huaweicloud.com/modelarts/?region=cn-north-4" \l "/dashboard/applyModelArtsAscend910Beta" \t "_blank)。 |
| “规格” | * CPU规格支持：“2核8GiB”、“8核32GiB”；GPU规格支持：“8核64GiB 1\*p100”；只有选择“公共资源池”时，需要选择规格。根据选择的类型不同，可选规格也不同。Ascend规格支持：“Ascend: 1\*Ascend 910 CPU: 24 核 96GiB” * CPU规格支持：“[限时免费]体验规格CPU版”、“2核8GiB”、“8核32GiB”；GPU规格支持：“[限时免费]体验规格GPU版”、“GPU: 1\*v100NV32 CPU: 8 核 64GiB”。 如果选择“限时免费”规格，请仔细阅读界面提示，并勾选“我已阅读并同意以上内容”。 |
| “存储配置” | 存储配置可选“云硬盘”和“对象存储服务”。   * 选择“云硬盘”作为存储位置   根据实际使用量设置磁盘规格。磁盘规格默认5GB。ModelArts提供5GB容量供用户免费使用。超出5GB时，超出部分每GB按“超高IO”类型的收费标准进行按需收费。磁盘规格的取值范围为5GB～4096GB。  选择此模式，用户在Notebook列表的所有文件读写操作都是针对容器中的内容操作，与OBS无关；重启该实例，内容不丢失。   * 选择“对象存储服务”作为存储位置   在“存储位置”右侧单击“选择”，设置用于存储Notebook数据的OBS路径。如果想直接使用已有的文件或数据，可将数据提前上传至对应的OBS路径下。“存储位置”不能设置为OBS桶的根目录，需设置为对应OBS桶下的具体目录。  选择此模式，用户在Notebook列表的所有文件读写操作是基于所选择的OBS路径下的内容操作，与当前实例空间无关。如果您需要将内容同步到实例空间，先选中该内容，单击“Sync OBS”，即可将所选内容同步到当前容器空间，详细操作可参见[与OBS同步文件](https://support.huaweicloud.com/engineers-modelarts/modelarts_23_0038.html)。重启该实例时，内容不丢失。 |
| “Git存储库” | 只有当“存储配置”选择“云硬盘”时，支持此参数。  开启此功能后，可创建一个带有Git存储库的Notebook实例，系统将自动从Github同步代码库。详细配置说明请参见[创建带有Git存储库的Notebook实例](https://support.huaweicloud.com/engineers-modelarts/modelarts_23_0279.html)。 |

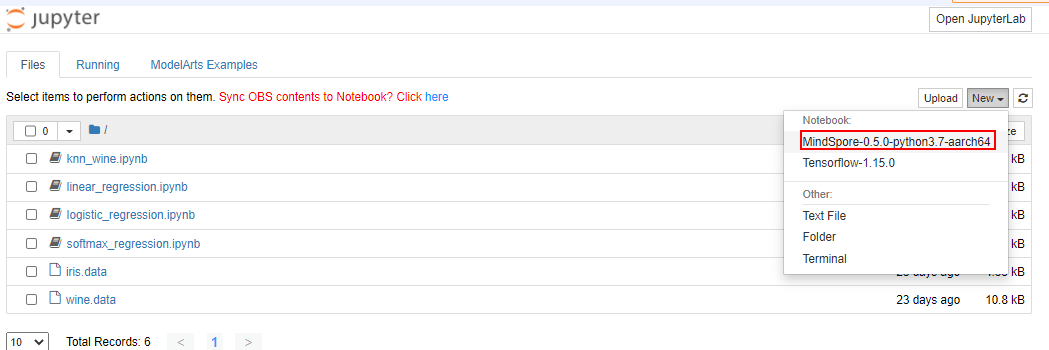
（3）配置好Notebook参数后，点击下一步，进入Notebook信息预览。确认无误后，点击“立即创建”。

（4）创建完成后，返回开发环境主界面，等待Notebook创建完毕后，打开Notebook，进行下一步操作。

**注意：开发框架使用MindSpore的实验，需要选择Notebook的配置为：Python3、Ascend、存储位置默认为OBS，如下图。**



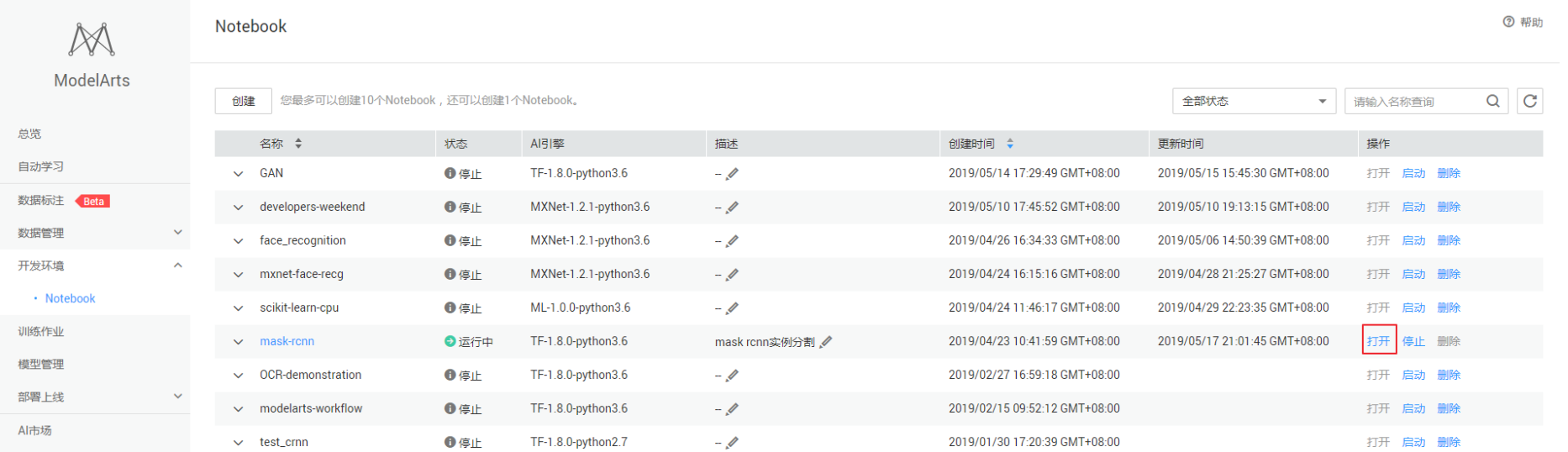
使用MindSpore时所需填写的Notebook参数



创建MindSpore的开发环境

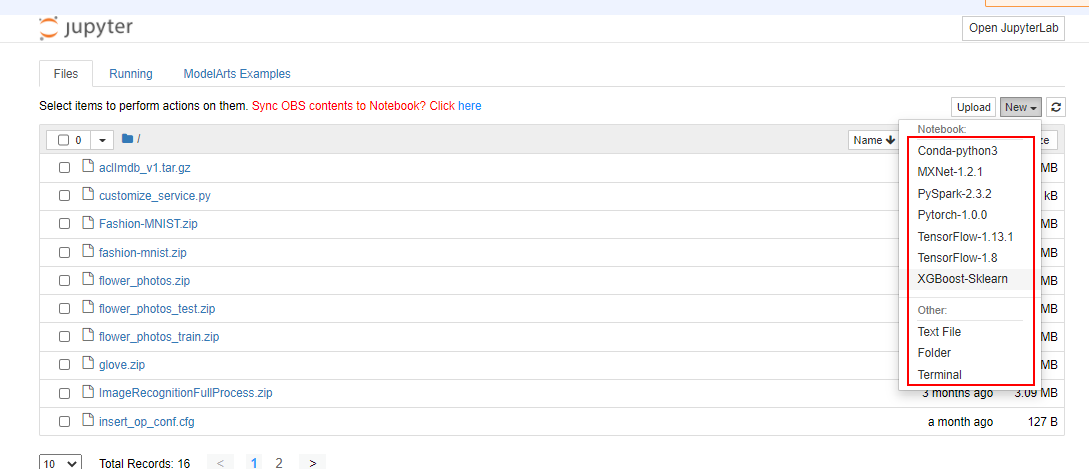
在Notebook中创建开发环境

（1）点击下图所示的“打开”按钮，进入刚刚创建的Notebook。

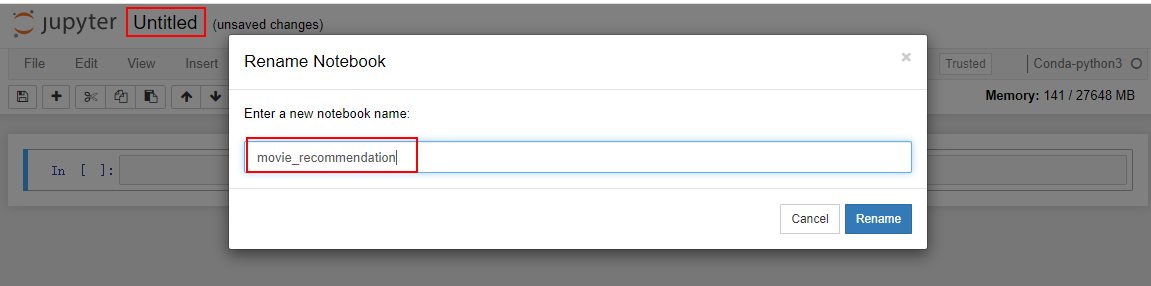


打开notebook

（2）创建一个Python3、CPU环境的Notebook。点击右上角的"New"，可以选择所创建的开发环境。



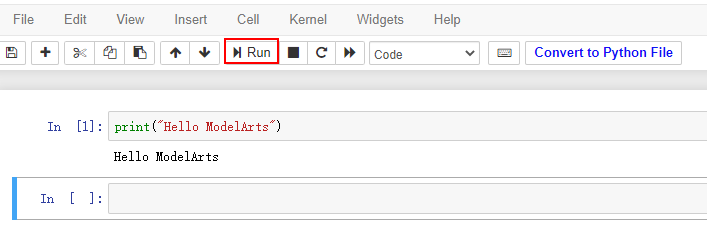
（3）点击左上方的文件名"Untitled"，并输入一个与本实验相关的名称，如"movie\_recommendation。



重命名notebook

在Notebook中编写并执行代码

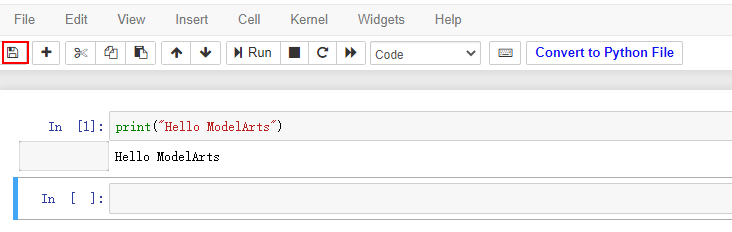
在Notebook中，我们输入一个简单的打印语句，然后点击上方的运行按钮，可以查看语句执行的结果：



测试notebook

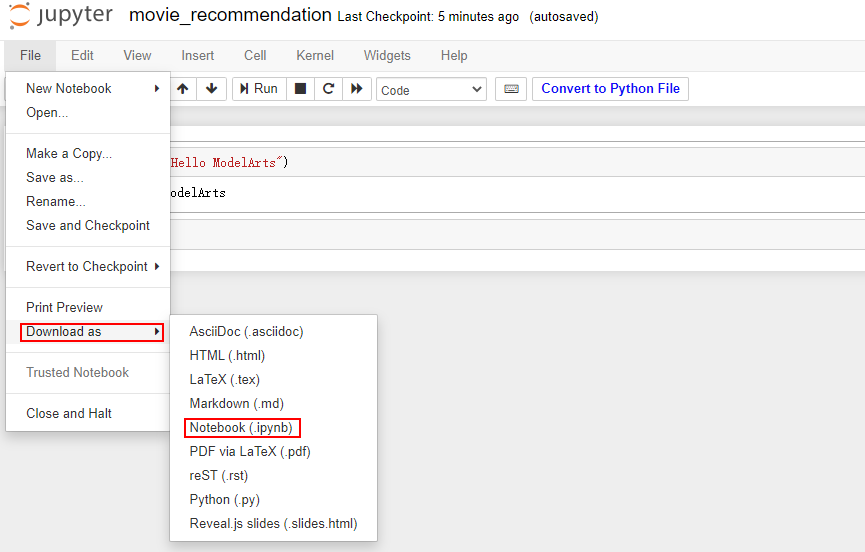
保存代码

代码编写完之后，我们点击下图所示的“保存”按钮，保存代码和代码执行结果。

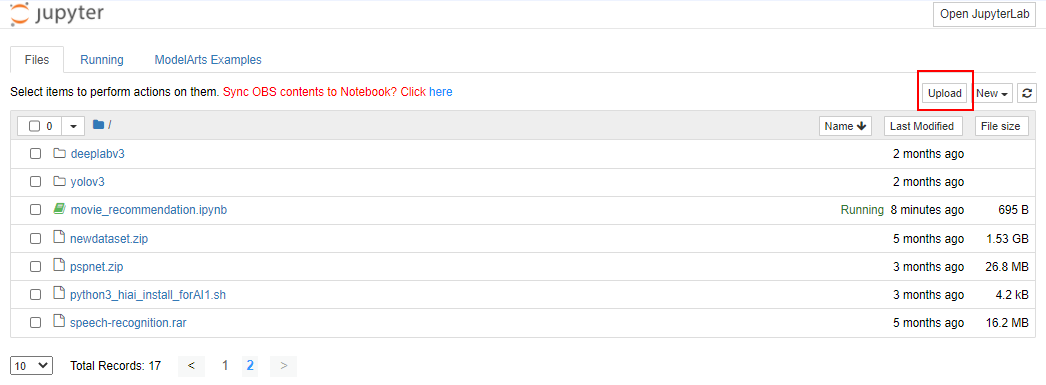


保存代码

从ModelArts下载代码到本地



从本地上传代码到ModelArts



# 数据预处理实验

## 实验介绍

本次实验我们将会对数据集进行相应的预处理工作，因为数据预处理的步骤相对灵活。我们会针对不同特征运用不同的预处理方法。这一个小节的实验，我们将处理一个特征维度的数据，以带领大家学习数据处理的基本语法。掌握了这些基本语法后，大家需要通过代码填空的方式自己动手实践处理数据。

## 实验目的

掌握缺失值的处理方法

掌握异常值的方法

掌握分类数据的处理与特征编码的实现

## 实验步骤

本次实验使用的数据集是google store中，各个app下载的信息。共10841个样本，每个样本13个特征。

实验任务介绍：本次实验我们将利用app的类型（Category）、收费情况(Type)、大小（Size）、评分(Rating)、评价数量(Reviews)和分级(Content Rating)等特征来预测APP的下载量。

首先我们将对整理数据集进行缺失值处理，接下来我们将以Rating、Size和Price这三特征为例，进行数据预处理的教学。

### 数据预处理

#### 数据读取

代码：

#导入相关库

import pandas as pd

import matplotlib.pyplot as plt

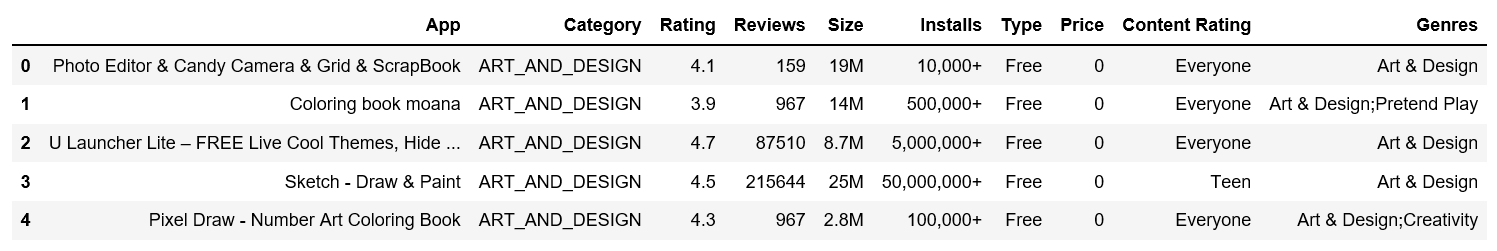
import seaborn as sns

import numpy as np

raw\_df = pd.read\_csv("googleappstorev1.csv",index\_col=0)#读取数据。index\_col=0：读取时不自动添加行号。

raw\_df.head()

输出：



#### 缺失值处理

查看缺失值

pandas中，isnull()是常用的缺失值查看方法：

df.isnull():查看所有数据是否为缺失值，返回bool值。True：缺失值；False：不是缺失值。

df.isnull().any():查看每个特征是否存在缺失值，返回bool值。

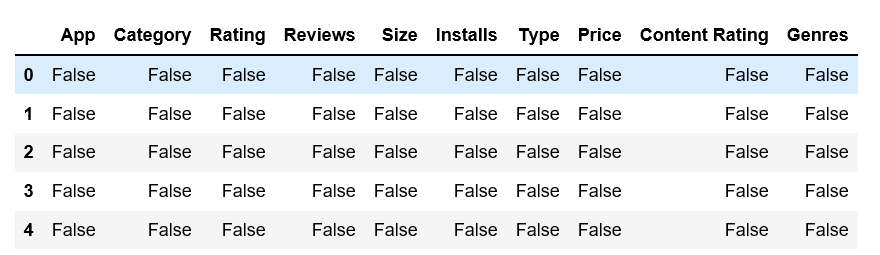
df.isnll().sum():返回每个特征中缺失值的具体数量。

代码：

# 查看前5个样本中，数据是否为缺失值。

raw\_df.head().isnull()

输出：



代码：

# 查看每个特征是否存在缺失值

raw\_df.isnull().any()

输出：

App False

Category False

Rating True

Reviews False

Size False

Installs False

Type True

Price False

Content Rating True

Genres False

dtype: bool

代码：

raw\_df.isnull().sum()

输出：

App 0

Category 0

Rating 1474

Reviews 0

Size 0

Installs 0

Type 1

Price 0

Content Rating 1

Genres 0

dtype: int64

在介绍完三种查看缺失值的方法后，对于本次任务来说，哪种方法更加合适呢？

思路：我们可以选用isnull().any()或isnull().sum()方法，查看哪些特征数据包含缺失值。

#### 缺失值处理

缺失值处理的常见方法有：

1删除：dropna();

2.填充：填充包括统计量填充（众数、均值、中位数）、KNN填充和回归预测填充等。常用方法包括：fillna()与Python的sklearn.preprocessing库中的Imputer类可对缺失值进行众数、均值、中位数填充。

3.不处理

对于本次实验的数据集和任务，应该对于缺失值进行删除还是填充操作呢？

思路：包含缺失值的特征包括：Rating(1474)、Type(1)、Content Rating(1)、Current Ver(8)和Android Ver(3)。除了Rating，其他特征只对应少量缺失数据样本，因此对于这种数据我们可以优先考虑删除缺失数据。而Rating特征对应的缺失数据样本较多，我们优先考虑缺失值填充。

缺失值填充

本次实验我们选择统计量填充方法。统计量填充的选择标准：

对于连续值，推荐使用中位数 ，可以排除一些特别大或者特别小的异常值造成的影响；

对于离散值，推荐使用众数，均值和中位数没有意义，不能使用。

对于评分，我们应该哪种统计量填充方法呢？[¶](http://localhost:8888/notebooks/%E5%BC%80%E5%8F%91%20-%20EBG%E5%9C%A8%E7%BA%BF%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E5%AE%9E%E8%B7%B5/1.%E6%95%B0%E6%8D%AE%E9%A2%84%E5%A4%84%E7%90%86.ipynb" \l "问题3：对于评分，我们应该哪种统计量填充方法呢？)

思路：评分数据的连续的值，所以我们选择中位数填充。

使用Python中pandas库的dropna ( ) 函数，其基本格式如下：

DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None)

关键参数详解：

value: 特定填充值

method: 差值方式：

pad/ffill：用前一个非缺失值去填充该缺失值

backfill/bfill：用下一个非缺失值填充该缺失值

None：指定一个值去替换缺失值（缺省默认这种方式）

axis: 默认为0。axis=0代表d对行数据进行操作，axis=1代表列数据

inplace=True/False：Boolean数据, 默认为False。inplace=True代表直接对原数据集N做出修改。

limit : 阈值。超过阈值才填充该行或该列。

代码：

#将样本按照分类类别进行分组，求出每类中评分的中位数，再填充缺失值。

raw\_df["Rating"] = raw\_df["Rating"].fillna(raw\_df["Rating"].median())

raw\_df["Rating"]

输出：

0 4.1

1 3.9

2 4.7

3 4.5

4 4.3

5 4.4

6 3.8

7 4.1

8 4.4

9 4.7

10 4.4

11 4.4

12 4.2

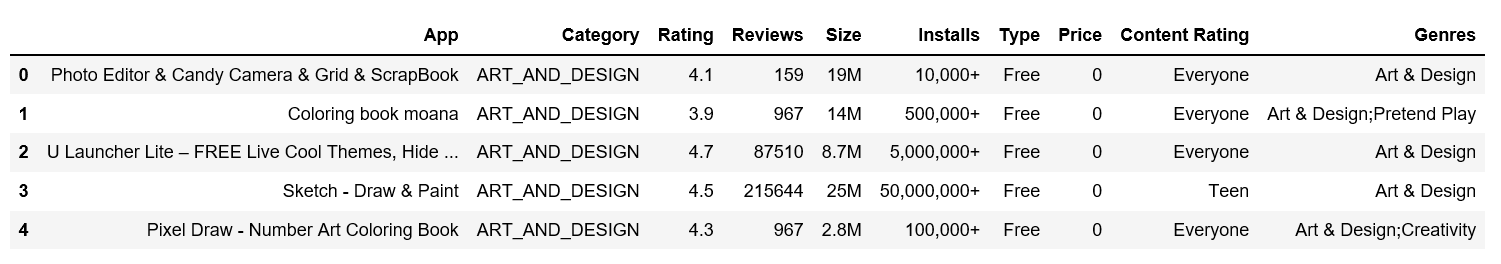
…

代码：

#查看

raw\_df.head()

输出：



缺失值删除

使用Python中pandas库的dropna ( ) 函数，其基本格式如下：

DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)

关键参数详解：

axis=0/1，默认为0。axis=0代表d对行数据进行操作，axis=1代表列数据。

how=any/all，默认为any。how=any代表若某行或某列中存在缺失值，则删除该行或该列。

how=all:若某行或某列中数值全部为空，则删除该行或该列。

thresh=N，可选参数，代表若某行或某列中至少含有N个缺失值，则删除该行或该列。

subset=列名，可选参数，代表若指定列中有缺失值，则删除该行。

inplace=True/False，Boolean数据, 默认为False。inplace=True代表直接对原数据集N做出修改。

inplace=False代表修改后生成新数据集M，原数据集N保持不变。

代码：

#删除新数据集的所有缺失值

non\_na\_df = raw\_df.dropna()

non\_na\_df

输出：



代码：

#当删除掉不需要的行时，行索引会变的不连续，这时候可以重新设计新的索引

non\_na\_df.reset\_index(drop=True,inplace=True)#drop=True：删除原行索引；inplace=True:在数据上进行更新

#检查数据集中是否含有缺失值

non\_na\_df.isnull().any()

输出：

App False

Category False

Rating False

Reviews False

Size False

Installs False

Type False

Price False

Content Rating False

Genres False

dtype: bool

#### 重复值处理

pandas中，除去重复值的常用方法为drop\_duplicate，其基本格式如下：

DataFrame.drop\_duplicates(subset=None, keep='first', inplace=False)：

关键参数详解：

subset:用来指定特定的列，默认所有列；

keep: {‘first’, ‘last’, False}。默认值为‘first’，用于删除重复项并保留第一次出现的项；

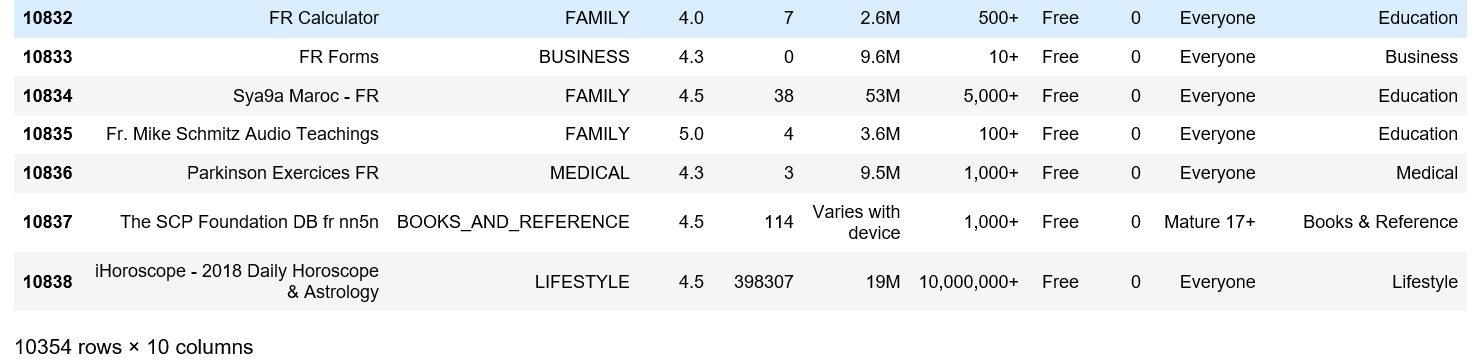
inplace:是直接在原来数据上修改还是保留一个副本

代码：

non\_na\_df.drop\_duplicates(inplace=True)

non\_na\_df

输出：



标签重置

non\_na\_df.reset\_index(drop= True,inplace = True)

异常值处理

异常值检测和处理异常值。 异常值检测的方法主要有：

1. 简单统计分析；

2. 散点图；

3. 箱型图；

4. 3-sigma；

5. 基于模型的异常值检测等。

异常值处理的方法主要包括：

1. 删除；

2. 视为缺失值，进行处理；

3. 不处理：可以直接在具有异常值的数据集上进行数据建模。

下面我们将根据每一特征数据，来处理特征的异常值。处理的常用步骤为：

1.查看特征的值；

2.特征处理；

3.异常值检测及处理。

下面我们一一对各个特征数据进行处理。

#### Rating

查看数据基本信息。主要查看数值的动态范围和数据类型。

代码：

rating\_copy = non\_na\_df.copy()

print(rating\_copy["Rating"].to\_frame().describe())

rating\_copy["Rating"]

输出：

Rating

count 10354.000000

mean 4.203738

std 0.485663

min 1.000000

25% 4.100000

50% 4.300000

75% 4.500000

max 5.000000

评分数据是float32的，数值变化范围为1到5。数据的标准差为0.486，说明数据的扰动不大。

可视化

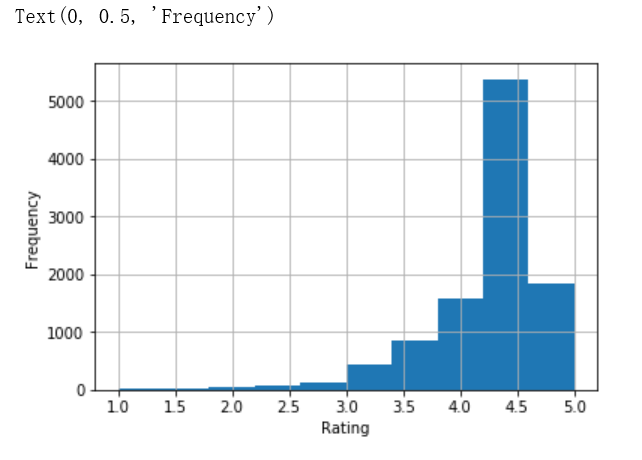
代码：

rating\_copy["Rating"].hist();

plt.xlabel('Rating')

plt.ylabel('Frequency')

输出：



评分与下载频数直方图

通过观察数据的统计数据和可视化，我们没有在评分数据中发现异常值。

#### Size

查看特征信息

代码：

Size\_copy = non\_na\_df.copy()

Size\_copy["Size"].value\_counts()

输出：

Varies with device 1525

11M 188

12M 186

13M 186

14M 182

15M 174

17M 155

26M 145

16M 143

19M 135

10M 133

25M 131

特征处理

size这个特征中，APP大小的单位不一致，且不是数值型的。因此我们首先需要统一单位，并将特征数据转换成数值型。

代码：

#定位单位为MB的数据

k\_indices = Size\_copy['Size'].loc[Size\_copy['Size'].str.contains('k')].index.tolist()

converter = pd.DataFrame(Size\_copy.loc[k\_indices, 'Size'].apply(lambda x: x.strip('k')).astype(float).apply(lambda x: x / 1024).apply(lambda x: round(x, 3)).astype(str))

Size\_copy.loc[k\_indices,'Size'] = converter

Size\_copy['Size'] = Size\_copy['Size'].str.replace('M','')

Size\_copy['Size'].head()

输出

0 19

1 14

2 8.7

3 25

4 2.8

Name: Size, dtype: object

Size特征中，出现最多的是“Varies with device”，我们需要对这种非数值型的字符串进行处理。这里我们将“Varies with device”当做缺失值处理。

代码：

#将Varies with device转化为缺失值

Size\_copy['Size'].replace("Varies with device", np.nan, inplace = True)

Size\_copy['Size']=Size\_copy['Size'].astype("float").apply(lambda x: round(x, 3))

Size\_copy['Size'].head()

输出：

0 19.0

1 14.0

2 8.7

3 25.0

4 2.8

Name: Size, dtype: float64

对于Varies with device，我们应该哪种方式处理缺失值？

思路：缺失值处理的常见方法有：删除：dropna();填充：填充包括统计量填充（众数、均值、中位数）；不处理。根据Varies with device的特点和对应样本的数量，在此我们用均值填充这个缺失值。

代码：

#根据各类别的均值填充数据

Size\_copy['Size'].fillna(Size\_copy.groupby('Category')['Size'].transform('mean'),inplace = True)

#查找检查非数值数据

Size\_copy.applymap(lambda x: isinstance(x,float))['Size'].value\_counts()

输出：

True 10354

Name: Size, dtype: int64

数据全为数值型数据。接下来可以对数据进行异常值检测了。

异常值检测

首先我们可以查看该特征的数值信息并绘制一下散点图，观察特征的数值。

代码：

Size\_copy['Size'].describe()

输出：

count 10354.000000

mean 21.000700

std 21.112004

min 0.008000

25% 5.700000

50% 14.000000

75% 27.930205

max 100.000000

Name: Size, dtype: float64

代码：

plt.figure(figsize = (10,10))

g = sns.jointplot(x="Size", y="Rating",color = 'orangered', data=Size\_copy, size = 8)

Size\_copy.to\_csv('appstorev1.2.csv')

输出



评分数据散点图

通过观察统计数据，发现数据的标准差很大。但是Size这个属性中，有很多APP的大小最大为1e9B，最小为1e6B。数值都在正常范围以内，且我们从散点图中，并未发现明显的异常值。因此这一特征我们认为是没有异常值。

#### Price

查看特征信息

代码：

price\_copy = Size\_copy.copy()

price\_copy["Price"].value\_counts()

输出：

0 9589

$0.99 146

$2.99 125

$1.99 73

$4.99 70

$3.99 60

$1.49 46

$5.99 27

$2.49 25

$9.99 19

$6.99 12

$399.99 12

$4.49 9

$14.99 9

$3.49 7

$7.99 7

数据中，包含非数值符号，需要删除。数据中，有92个不同的价格。

特征处理

**删除非数值符号$。**

**代码：**

**#用replace删除数值当中符号，并把特征中的数据类型转化为数值型。**

**price\_copy["Price"]=price\_copy["Price"].str.replace('$','').astype("float")**

**#price\_copy["Price"].describe()**

**price\_copy["Price"].value\_counts().head(10)**

**输出：**

0.00 9589

0.99 146

2.99 125

1.99 73

4.99 70

3.99 60

1.49 46

5.99 27

2.49 25

9.99 19

Name: Price, dtype: int64

异常值检测

查看统计数据。

代码：

price\_copy["Price"].describe()

输出：

count 10354.000000

mean 1.031099

std 16.280974

min 0.000000

25% 0.000000

50% 0.000000

75% 0.000000

max 400.000000

Name: Price, dtype: float64

从统计数据中我们发现，APP价格的动态范围在[0,400]之间，标准差为16.295。这里我们可以选择查看一下价格比较大APP，通过观察这些样本判断是否为异常值。

代码：

#查看价格大于350美金的样本

price\_copy[price\_copy["Price"]>300]

输出：



这些数据并没有明显异常。

数据可视化，绘制一下散点图，观察数据变换趋势。

代码：

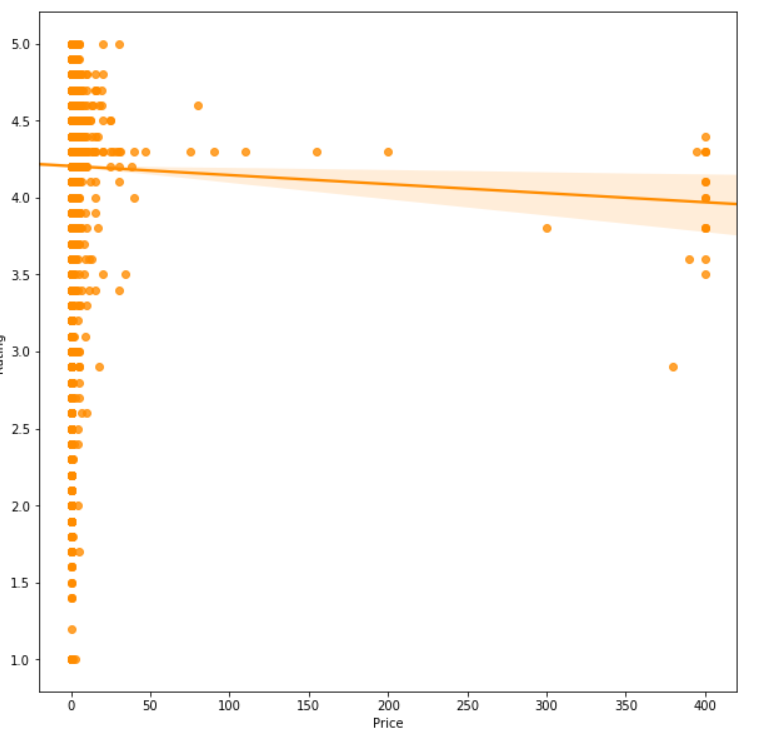
plt.figure(figsize = (10,10))

sns.regplot(x="Price", y="Rating", color = 'darkorange',data= price\_copy)

#保存文件

price\_copy.to\_csv('appstorev1.3.csv')

输出：



价格散点图

从散点图我们可以看出，价格对评分的影响不大，但是如果APP价格设置的过高，会影响评分。

### 思考题

接下来，请大家根据第一节学习内容。对Installs和Reviews进行特征处理、无量纲化(规范化)和异常值处理。 基本步骤如下：

查看特征的值

特征处理

异常值检测及处理

#导入相关库

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

……

…….

### 数据预处理 - 分类数据

上一部分实验，我们完成了缺失值处理和包括Rating、Size、Price、Installs和Reviews在内的5个特征的异常值处理工作。这些都是数值属性的数据。接下来我们对分类属性的特征的处理工作，包括App、Category、Type、Content Rating和Genres。

本次实验中，我们首先Content Rating为例，一起学习处理分类数据的过程。掌握了这些基本语法后，大家需要通过代码填空的方式自己动手实践处理数据。

数据挖掘中，一些算法可以直接计算分类变量，比如决策树模型。但许多机器学习算法不能直接处理分类变量，它们的输入和输出都是数值型数据。因此，把分类变量转换成数值型数据是必要的，可以用独热编码 (One-Hot Encoding) 、哑编码 (Dummy Encoding)和label encoding实现。

* 无序分类变量的离散化方法较为常用方法：独热编码 (One-Hot Encoding)。
* 有序分类变量{低年级，中年级，高年级}，可以使用Label-Encoding直接离散化为{0,1,2}。

代码：

#导入相关库

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

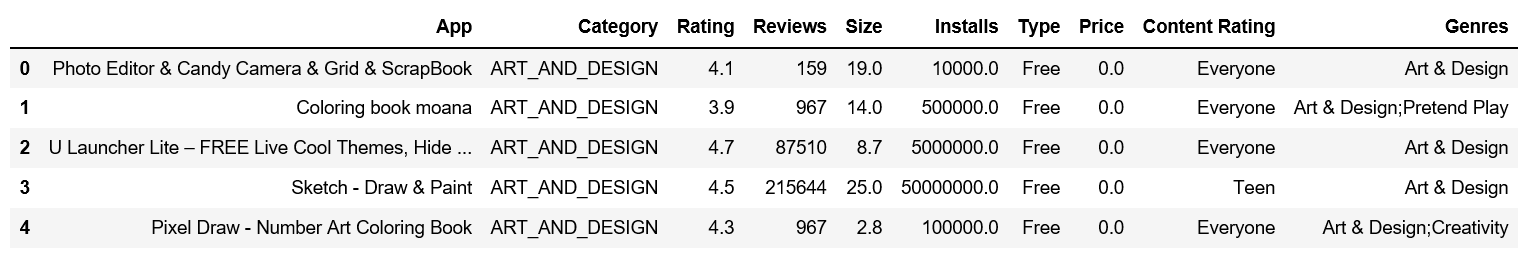
import numpy as np

#数据读取

raw\_df = pd.read\_csv('appstorev1.4.csv',index\_col=0)

raw\_df.head()

输出：



#### Type

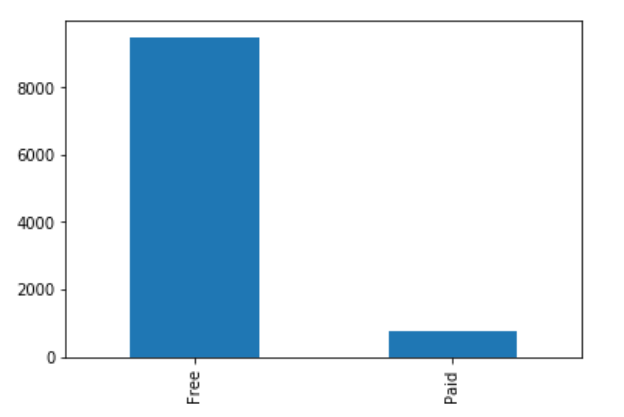
**查看特征信息及可视化**

**代码：**

#查看特征的数值

raw\_df['Type'].value\_counts().plot(kind='bar')

输出：



可视化特征信息

**分类变量数值化**

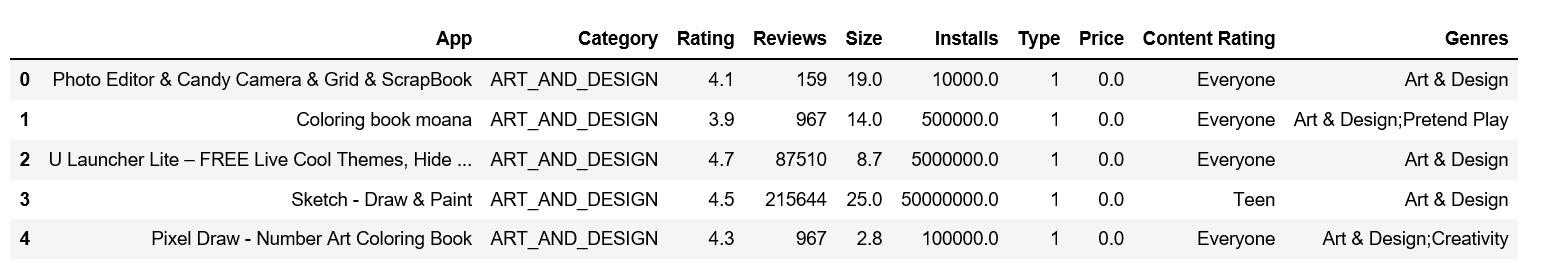
Type数据只有两种可能数值，包括free和paid。因此我们可以用Label-Encoding将这一特征数据离散化。

代码：

#我们可以直接通过map实现分类变量的离散化

raw\_df['Type'] = raw\_df['Type'].map({'Free':1,"Paid":2})

raw\_df.head()

输出：

#### Content Rating

**查看特征信息及可视化。**

代码：

#查看特征的数值

raw\_df['Content Rating'].value\_counts()

输出：

Everyone 8305

Teen 1128

Mature 17+ 445

Everyone 10+ 359

Adults only 18+ 3

Unrated 2

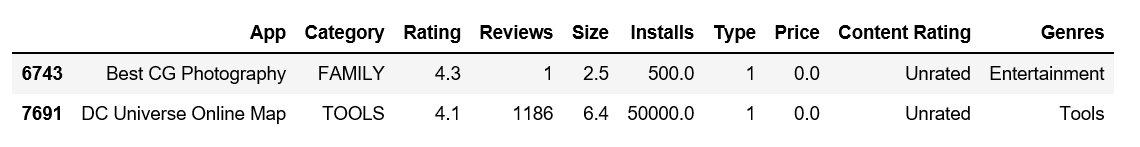
Name: Content Rating, dtype: int64

内容分级数据中，有2个样本没有分级。我们可以单独查看一下这两个样本，观察是否为缺失值。

代码：

raw\_df[raw\_df['Content Rating']=='Unrated']

输出：



可以看出数据的第一条数据只有一条评论和一次下载，第二条数据只有9次下载却有1186条评论。因此我们可以将这两个没有分级的样本给删除。

代码：

raw\_df = raw\_df[raw\_df['Content Rating'] !='Unrated']

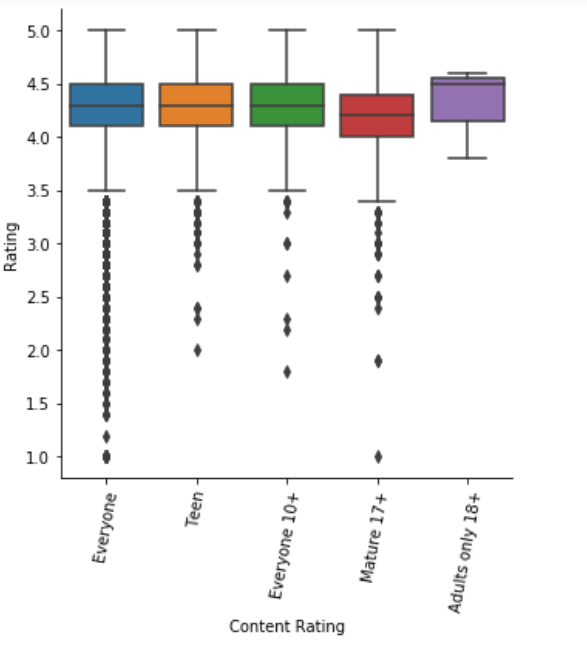
#更新重置样本标签

raw\_df.reset\_index(drop= True,inplace = True)

sns.catplot(x="Content Rating",y="Rating",data=raw\_df, kind="box")

plt.xticks(rotation=80)

输出：



内容分级箱型图

从箱型图中，每个内容分级的均值来看Adults only 18+的平均评分最高。Mature 17+的平均评分最低。

**分类变量数值化**

内容分级这一特征三种取值不是完全独立的，根据限制的程度我们可以排序为Everyone，Everyone 10+，Teen，Mature 17+和Adults only 18。因此分级有高低是有序变量，这里用 LabelEncoder。

代码：

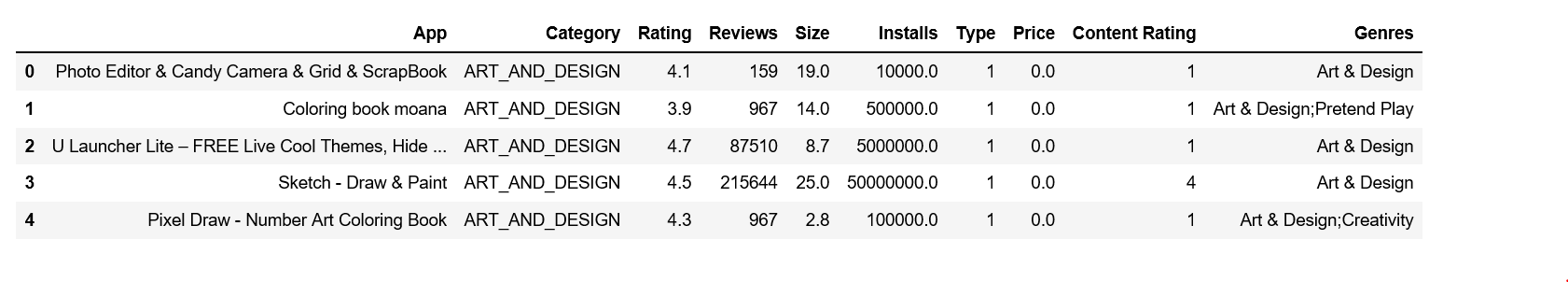
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

raw\_df['Content Rating'] = le.fit\_transform(raw\_df['Content Rating'])

raw\_df.head()

输出：



#### Category

**查看特征信息及可视化。**

代码：

raw\_df["Category"].value\_counts()

输出：

FAMILY 1930

GAME 1081

TOOLS 835

BUSINESS 426

MEDICAL 407

PRODUCTIVITY 406

PERSONALIZATION 387

LIFESTYLE 372

FINANCE 358

SPORTS 349

COMMUNICATION 348

PHOTOGRAPHY 318

HEALTH\_AND\_FITNESS 306

SOCIAL 267

NEWS\_AND\_MAGAZINES 261

TRAVEL\_AND\_LOCAL 233

BOOKS\_AND\_REFERENCE 230

SHOPPING 223

DATING 196

VIDEO\_PLAYERS 172

MAPS\_AND\_NAVIGATION 137

EDUCATION 130

FOOD\_AND\_DRINK 124

ENTERTAINMENT 111

AUTO\_AND\_VEHICLES 85

LIBRARIES\_AND\_DEMO 85

WEATHER 82

HOUSE\_AND\_HOME 80

EVENTS 64

ART\_AND\_DESIGN 64

COMICS 60

PARENTING 60

BEAUTY 53

Name: Category, dtype: int64

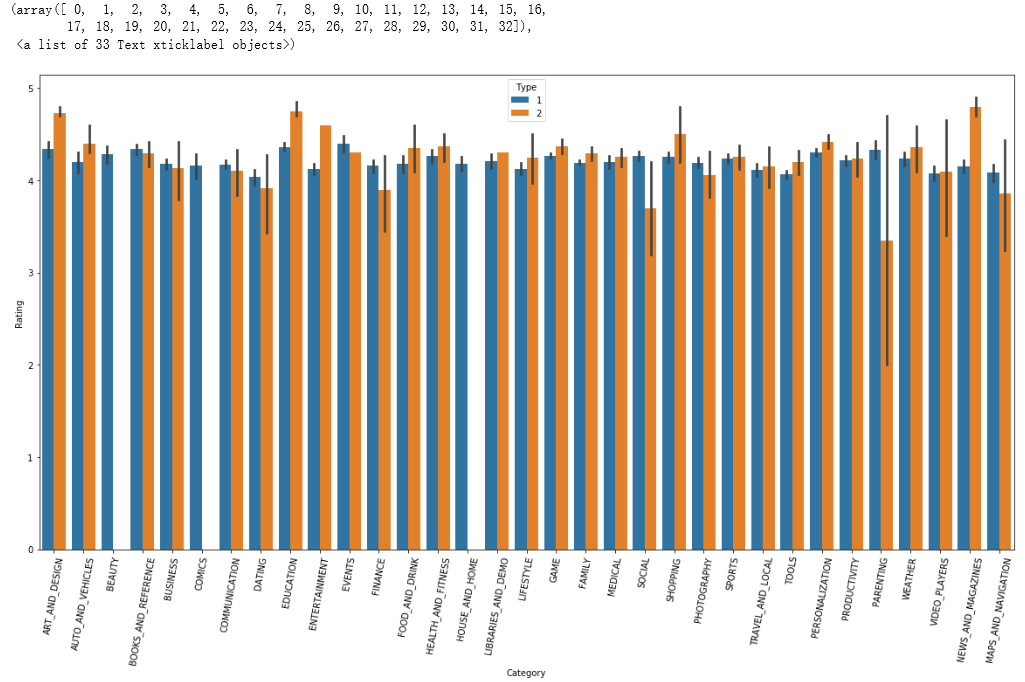
代码：

plt.figure(figsize=(20, 10))

sns.barplot(x="Category", y="Rating",hue="Type", data=raw\_df)

plt.xticks(rotation=80)

输出：



APP类别评分图

对于art\_and\_design、education、entertainment和news\_and\_magnizeszz这几类来说，收费的APP的均分比较高。对于social和parenting来说，APP费用很可能会带来不好的评分。beauty、comics和house\_and\_home这三个类别的所有APP都是免费的。

**分类变量数值化**

经过上面的学习，对类别进行数值离散化，使用哪个方法更合理呢？[¶](http://localhost:8888/notebooks/%E5%BC%80%E5%8F%91%20-%20EBG%E5%9C%A8%E7%BA%BF%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E5%AE%9E%E8%B7%B5/3.%E6%95%B0%E6%8D%AE%E9%A2%84%E5%A4%84%E7%90%86%20-%202.ipynb" \l "问题1：经过上面的学习，对类别进行数值离散化，使用哪个方法更合理呢？)

思路：类别数据是无序的。我们可以采用one-hot编码。one-hot编码的实现方式重要有两种。第一种是利用pandas中的get\_dummy()实现；第二种是利用Sciket learn中的from sklearn.preprocessing import OneHotEncoder实现。由于pandas机制问题，它需要在内存中把数据集都读入进来，要是数据量大的话，太消耗资源，one-hot可以读数组，因此大规模数据集很方便。如果我们选用label-encoding，这样类别数据可能会出现顺序关系，可能会降低模型的精确度。我们选用独热码。

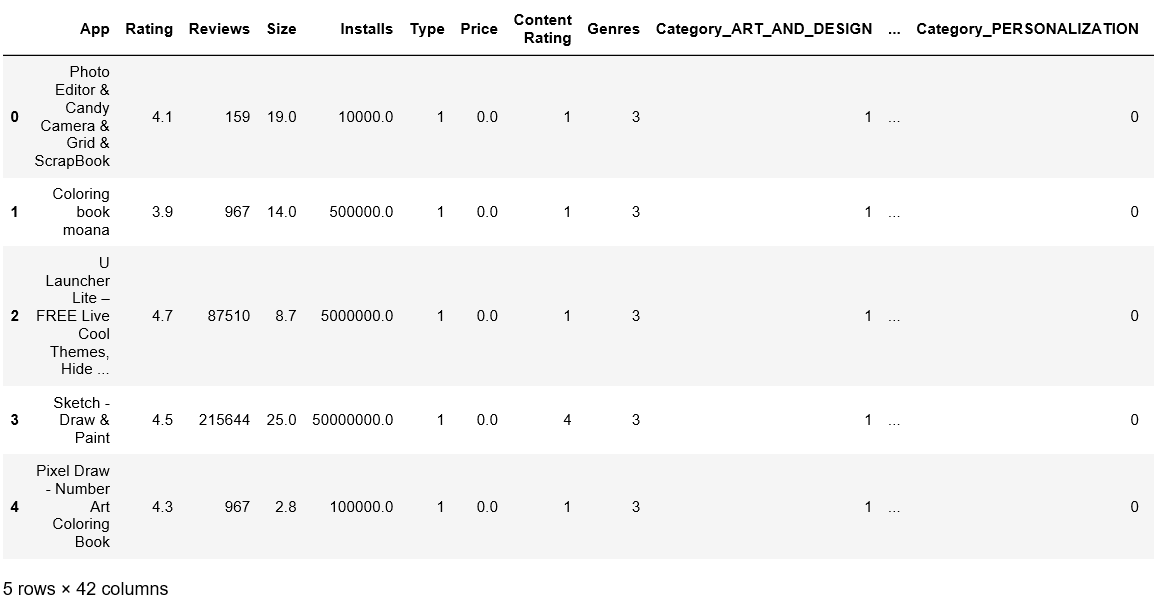
代码：

#用get\_dummies()将Category转换为数值数据。

raw\_df = pd.get\_dummies(raw\_df, columns= ["Category"])

raw\_df.head()

输出：



#### Genres

查看处理特征数据

代码：

raw\_df["Genres"].value\_counts()

输出：

Tools 834

Entertainment 587

Education 525

Business 426

Medical 407

Productivity 406

Personalization 387

Lifestyle 371

Finance 358

Sports 354

Action 354

Communication 348

Photography 318

Health & Fitness 306

Social 267

News & Magazines 261

Travel & Local 232

Books & Reference 230

Shopping 223

Arcade 213

Simulation 199

Dating 196

Casual 173

Video Players & Editors 170

Maps & Navigation 137

Puzzle 136

Food & Drink 124

Role Playing 109

Strategy 97

Racing 94

...

Art & Design;Action & Adventure 2

Entertainment;Pretend Play 2

Video Players & Editors;Creativity 2

Books & Reference;Education 2

Art & Design;Pretend Play 2

Adventure;Education 2

Card;Brain Games 1

Travel & Local;Action & Adventure 1

Adventure;Brain Games 1

Strategy;Education 1

Health & Fitness;Education 1

Racing;Pretend Play 1

Entertainment;Education 1

Music & Audio;Music & Video 1

从输出的类型数据我们可以看出，某些样本的类型数据，其实包含两个类型，比如Video Players & Editors;Creativity，Card;Action & Adventure和Books & Reference;Creativity 。分号前为主要的类型，分号后为次要的类型。有因Genres和Category这个特征表示的意思有重复的地方。这里我们只保留分号前的主要类型。

代码：

sep = ';'

raw\_df['Genres']= raw\_df['Genres'].apply(lambda x: x.split(sep)[0])

raw\_df["Genres"].value\_counts()

输出：

Tools 835

Entertainment 627

Education 608

Business 426

Medical 407

Productivity 406

Personalization 387

Lifestyle 373

Action 369

Finance 358

Sports 358

Communication 349

Photography 318

Health & Fitness 308

Social 267

News & Magazines 261

Casual 242

…

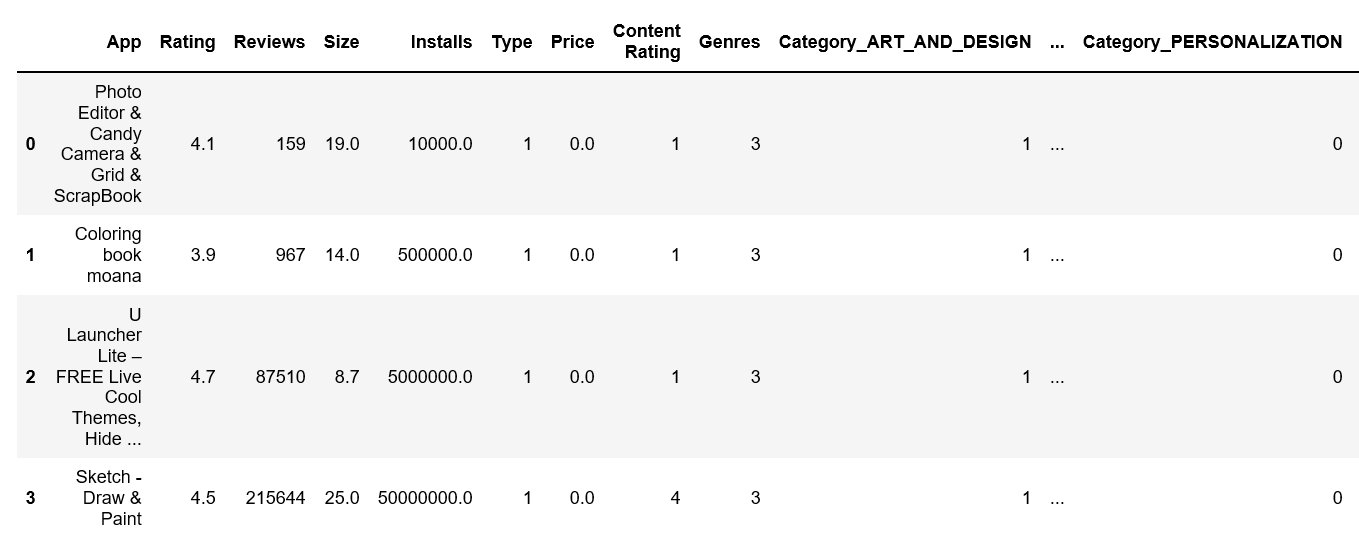
分类变量数值化

代码：

raw\_df['Genres'] = le.fit\_transform(raw\_df['Genres'])

raw\_df.head()

输出：



这里APP的名字我们不做处理，直接删除。

代码：

raw\_df = raw\_df.drop(["App"],axis='columns')

#保存文件

raw\_df.to\_csv("AppDataV2.csv")

## 实验小结

本实验我们对评分数据进行了数据预处理工作，主要的操作包括了缺失值查看和处理方式、异常值查看、异常值处理、特征数据的处理、数值数据处理与分类数据的处理。