

转

手把手教写出XGBoost实战程序

2018年08月14日 21:46:21

九城风雪

阅读数: 422

标签: XGBOOST 更多

0

**简单介绍:**

这是一个真实的比赛。赛题来源是天池大数据的 "商场中精确定位用户所在店铺"。原数据有114万条，计算起来非常困难。为了让初学者有一个更加基础，我将数据集缩小了之后放在这里，密码：ndfd。供大家下载。

在我的数据中，数据是这样子的：**train.csv**

user_id	用户的id	time_stamp	时间戳
latitude	纬度	wifi_strong 1-10	十个wifi的信号强度
longitude	经度	wifi_id 1-10	十个wifi的id
shop_id	商店的id	con_sta 1-10	十个wifi连接状态

test.csv

user_id	用户的id	time_stamp	时间戳
latitude	纬度	wifi_id 1-10	十个wifi的id
longitude	经度	con_sta 1-10	十个wifi连接状态
row_id	行标	wifi_strong 1-10	十个wifi的信号强度
shop_id	商店的id		

这个题目的意思是，我们在商场中，由于不同层数和GPS精度限制，我们并不能仅根据经纬度准确知道某用户具体在哪一家商店中。我们通过手机与附近10的连接情况，来精准判断出用户在哪个商店中。方便公司根据用户的位置投放相应店家的广告。

开始实战

准备实战之前，当然要对整个XGBoost有一个基本了解，对这个模型不太熟悉的朋友，建议看我之前的文章《XGBoost》。

实战的流程一般是先将数据预处理，成为我们模型可处理的数据，包括丢失值处理，数据拆解，类型转换等等。然后将其导入模型运行，最后根据结果调整参数，反复调参数达到最优。

我们在机器学习实战的时候一定要脱离一个思维惯性——一切都得我们思考周全才可以运行。这是一个很有趣的思维惯性，怎么解释呢？比如这道赛题学通信出身的，看到十个wifi强度值，就想找这中间的关系，然后编程来求解人的确切位置。这本质上还是我们的思维停留在显式编程的层面上，觉得程序5楚才可达到预定的目标。但其实大数据处理并不是这个原理。决策树不管遇到什么数据，无论是时间还是地理位置，都是一样的按照一定规则生成树，最后按照这个树走一遍得到预测的结果。也就是说我们不必花很多精力去考虑每个数据的具体物理意义，只要把他们放进模型里面就可以了。(调参需要简单地意义来给各个数据以权重，这个以后再说)

分析一下数据

我们的数据的意义都在上面那张表里面，我们有用用户的id、经纬度、时间戳、商店id、wifi信息。我们简单思考可以知道：

- 1. user\_id并没有什么实际意义，仅仅是一个代号而已
- 2. shop\_id是我们预测的目标，我们题目要求就是我们根据其他信息来预测出用户所在的shop\_id,所以 shop\_id 是我们的训练目标
- 3. 经纬度跟我们的位置有关，是有用的信息
- 4. wifi\_id 让我们知道是哪个路由器，这个不同的路由器位置不一样，所以有用
- 5. wifi\_strong是信号强度，跟我们离路由器距离有关，有用

7. 我们看test.csv总体差不多，就多了个row\_id,我们输出结果要注意对应上就可以

[登录](#)[注册](#)[×](#)

python库准备

```
1 import pandas as pd
2 import xgboost as xgb
3 from sklearn import preprocessing
4 复制代码
```

咱这个XGBoost比较简单，所以就使用了最必要的三个库，pandas数据处理库，xgboost库，从大名鼎鼎的机器学习库sklearn中导入了preprocessing库，这pandas库对数据的基本处理有很多封装函数，用起来比较顺手。想看例子的戳这个[链接](#)，我写的pandas.DataFrame基本拆解数据的方法。

先进行数据预处理

咱得先导入一份数据：

```
1 train = pd.read_csv(r'D:\XGBoost_learn\mall_location\train2.csv')
2 tests = pd.read_csv(r'D:\XGBoost_learn\mall_location\test_pre.csv')
3 复制代码
```

我们使用pandas里面的read\_csv函数直接读取csv文件。csv文件全名是Comma-Separated Values文件，就是每个数据之间都以逗号隔开，比较简洁，也是比赛常用的格式。我们需要注意的是路径问题，windows下是\,linux下是/，这个有区别。并且我们写的路径经常会与库里的函数字段重合，所以在路径最前来自禁止与库里匹配，重合报错。r是raw的意思，生的，大家根据名字自行理解一下。

我们的time\_stamp原来是一个str类型的数据，计算机是不会知道它是什么东西的，只知道是一串字符串。所以我们进行转化成datetime处理：

```
1 train['time_stamp'] = pd.to_datetime(pd.Series(train['time_stamp']))
2 tests['time_stamp'] = pd.to_datetime(pd.Series(tests['time_stamp']))
3 复制代码
```

train和tests都要处理。这也体现了pandas的强大。接下来我们看time\_stamp数据的样子：2017/8/6 21:20，看数据集可知，是一个十分钟为精确度(粒度)的感觉这个数据包含太多信息了呢，放一起很浪费(其实是容易过拟合，因为一个结点会被分的很细)，我们就将其拆开吧：

```
1 train['Year'] = train['time_stamp'].apply(lambda x: x.year)
2 train['Month'] = train['time_stamp'].apply(lambda x: x.month)
3 train['weekday'] = train['time_stamp'].dt.dayofweek
4 train['time'] = train['time_stamp'].dt.time
5 tests['Year'] = tests['time_stamp'].apply(lambda x: x.year)
6 tests['Month'] = tests['time_stamp'].apply(lambda x: x.month)
7 tests['weekday'] = tests['time_stamp'].dt.dayofweek
8 tests['time'] = tests['time_stamp'].dt.time
9 复制代码
```

细心的朋友可能会发现，这里采用了两种写法，一种是.apply(lambda x: x.year)，这是什么意思呢？这其实是采用了一种叫匿名函数的写法。匿名函数就是要写一个函数，但并不想费神去思考这个函数该如何命名，这时候我们就需要一个匿名函数，来实现一些小功能。我们这里采用的是.apply(lambda x: x.year)是调用了apply函数，是加这一列的意思，加的列的内容就是x.year。我们要是觉得这样写不直观的话，也可以这样写：

```
1 YearApply(x):
2     return x.year
3
4 train['Year'] = train['time_stamp'].apply(YearApply)
5 复制代码
```

这两种写法意义都是一样的，在调用weekday和datetime的时候，我们使用的是sklearn里面的函数，由于代码冗余，其实这也可以这样写，

数。为什么采用weekday呢，因为星期几比几号对于购物来说更加有特征性。接下来我们将这个time\_stamp丢掉，因为已经有了year、month那些：

```
1 train = train.drop('time_stamp', axis=1)
2 tests = tests.drop('time_stamp', axis=1)
3 复制代码
```

再丢掉缺失值，或者补上缺失值。

```
1 train = train.dropna(axis=0)
2 tests = tests.fillna(method='pad')
3 复制代码
```

我们看到我对训练集和测试集做了两种不同方式的处理。训练集数据比较多，而且缺失值比例比较少，于是就将所有缺失值使用dropna函数，tests为测试集，不能丢失一个信息，哪怕数据很多缺失值很少，所以我们用各种方法来补上，这里采用前一个非nan值补充的方式（method=“pad”），当然也有其他方式，如用这一列出现频率最高的值来补充。

```
1 class DataFrameImputer(TransformerMixin):
2     def fit(self, X, y=None):
3         for c in X:
4             if X[c].dtype == np.dtype('O'):
5                 fill_number = X[c].value_counts().index[0]
6                 self.fill = pd.Series(fill_number, index=X.columns)
7             else:
8                 fill_number = X[c].median()
9                 self.fill = pd.Series(fill_number, index=X.columns)
10        return self
11
12    def transform(self, X, y=None):
13        return X.fillna(self.fill)
14
15 train = DataFrameImputer().fit_transform(train)
16 复制代码
```

这一段代码有一点拗口，意思是对于X中的每一个c，如果X[c]的类型是object（‘O’表示object）的话就将X[c].value\_counts().index[0]传给空值，X[c].value\_counts().index[0]表示的是重复出现最多的那个数，如果不是object类型的话，就传回去X[c].median()，也就是这些数的中位数。

在这里我们可以使用print来输出一下我们的数据是什么样子的。

```
1 print(train.info())
2 复制代码
```

```
1 <class 'pandas.core.frame.DataFrame' at 0x0000024527C50D08>
2 Int64Index: 467 entries, 0 to 499
3 Data columns (total 38 columns):
4 user_id          467 non-null object
5 shop_id          467 non-null object
6 longitude        467 non-null float64
7 latitude         467 non-null float64
8 wifi_id1         467 non-null object
9 wifi_strong1     467 non-null int64
10 con_sta1         467 non-null bool
11 wifi_id2         467 non-null object
12 wifi_strong2     467 non-null int64
13 con_sta2         467 non-null object
```

```

14 | wifi_id3          467 non-null object
    |                  15 | wifi_strong3      467 non-null float64
16 | con_sta3          467 non-null object
17 | wifi_id4          467 non-null object
18 | wifi_strong4      467 non-null float64
19 | con_sta4          467 non-null object
20 | wifi_id5          467 non-null object
21 | wifi_strong5      467 non-null float64
22 | con_sta5          467 non-null object
23 | wifi_id6          467 non-null object
24 | wifi_strong6      467 non-null float64
25 | con_sta6          467 non-null object
26 | wifi_id7          467 non-null object
27 | wifi_strong7      467 non-null float64
28 | con_sta7          467 non-null object
29 | wifi_id8          467 non-null object
30 | wifi_strong8      467 non-null float64
31 | con_sta8          467 non-null object
32 | wifi_id9          467 non-null object
33 | wifi_strong9      467 non-null float64
34 | con_sta9          467 non-null object
35 | wifi_id10         467 non-null object
36 | wifi_strong10     467 non-null float64
37 | con_sta10         467 non-null object
38 | Year              467 non-null int64
39 | Month            467 non-null int64
40 | weekday          467 non-null int64
41 | time             467 non-null object
42 | dtypes: bool(1), float64(10), int64(5), object(22)
43 | memory usage: 139.1+ KB
44 | None
45 | 复制代码

```

我们可以清晰地看出我们代码的结构，有多少列，每一列下有多少个值等等，有没有空值我们可以根据值的数量来判断。我们在缺失值处理之前加入这个 `print(train.info())` 就会得到：

```

1 | <class 'pandas.core.frame.DataFrame' at 0x000001ECFA6D6718>
2 | RangeIndex: 500 entries, 0 to 499
3 | 复制代码

```

这里面就有500个值，处理后就只剩467个值了，可见丢弃了不少。同样的我们也可以将test的信息输出一下：

```

1 | <class 'pandas.core.frame.DataFrame' at 0x0000019E13A96F48>
2 | RangeIndex: 500 entries, 0 to 499
3 | 复制代码

```

500个值一个没少。都给补上了。这里我只取了输出信息的标题，没有全贴过来，因为全信息篇幅很长。我们注意到这个数据中有bool、float、int、obje型，我们XGBoost是一种回归树，只能处理数字类的数据，所以我们要转化。对于那些字符串类型的数据我们该如何处理呢？我们采用LabelEncoder方法：

```

1 | for f in train.columns:
2 |     if train[f].dtype=='object':
3 |         if f != 'shop_id':
4 |             print(f)
5 |             lbl = preprocessing.LabelEncoder()
6 |             train[f] = lbl.fit_transform(list(train[f].values))
7 | for f in tests.columns:
8 |     if tests[f].dtype == 'object':
9 |         print(f)
10 |         lbl = preprocessing.LabelEncoder()
11 |         tests[f] = lbl.fit_transform(list(tests[f].values))
12 | 复制代码

```

这段代码的意思是调用sklearn中preprocessing里面的LabelEncoder方法，对数据进行标签编码，作用主要就是使其变成数字类数据，有的进行归一化处理，行更快等等。我们看这段代码，lbl只是LabelEncoder的简写，`lbl = preprocessing.LabelEncoder()`，这段代码只有一个代换显得一行不那么长而已，没什么。第二句`lbl.fit_transform(list(train[f].values))`是将train里面的每一个值进行编码，我们在其前后输出一下`train[f].values`就可以看出来：

```
1 print(train[f].values)
2 train[f] = lbl.fit_transform(list(train[f].values))
3 print(train[f].values)
4 复制代码
```

我加上那一串0和/的目的是分隔开输出数据。我们得到：

```
1 user_id
2 ['u_376' 'u_376' 'u_1041' 'u_1158' 'u_1654' 'u_2733' 'u_2848' 'u_3063'
3  'u_3063' 'u_3063' 'u_3604' 'u_4250' 'u_4508' 'u_5026' 'u_5488' 'u_5488'
4  'u_5602' 'u_5602' 'u_5602' 'u_5870' 'u_6429' 'u_6429' 'u_6870' 'u_6910'
5  'u_7037' 'u_7079' 'u_7869' 'u_8045' 'u_8209']
6 [ 7  7  0  1  2  3  4  5  5  5  6  8  9 10 11 11 12 12 12 13 14 14 15 16 17
7  18 19 20 21]
8 复制代码
```

我们可以看出，LabelEncoder将我们的str类型的数据转换成数字了。按照它自己的一套标准。对于tests数据，我们可以看到，我单独将shop\_id给避开了。理的原因就是shop\_id是我们提交的数据，不能有任何编码行为，一定要保持这种str状态。

接下来需要将train和tests转化成matrix类型，方便XGBoost运算：

```
1 feature_columns_to_use = ['Year', 'Month', 'weekday',
2  'time', 'longitude', 'latitude',
3  'wifi_id1', 'wifi_strong1', 'con_sta1',
4  'wifi_id2', 'wifi_strong2', 'con_sta2',
5  'wifi_id3', 'wifi_strong3', 'con_sta3',
6  'wifi_id4', 'wifi_strong4', 'con_sta4',
7  'wifi_id5', 'wifi_strong5', 'con_sta5',
8  'wifi_id6', 'wifi_strong6', 'con_sta6',
9  'wifi_id7', 'wifi_strong7', 'con_sta7',
10 'wifi_id8', 'wifi_strong8', 'con_sta8',
11 'wifi_id9', 'wifi_strong9', 'con_sta9',
12 'wifi_id10', 'wifi_strong10', 'con_sta10',]
13 train_for_matrix = train[feature_columns_to_use]
14 test_for_matrix = tests[feature_columns_to_use]
15 train_X = train_for_matrix.as_matrix()
16 test_X = test_for_matrix.as_matrix()
17 train_y = train['shop_id']
18 复制代码
```

待训练目标是我们的shop\_id,所以train\_y是shop\_id。

导入模型生成决策树

```
1 gbm = xgb.XGBClassifier(silent=1, max_depth=10, n_estimators=1000, learning_rate=0.05)
2 gbm.fit(train_X, train_y)
3 复制代码
```

这两句其实可以合并成一句，我们也就是在XGBClassifier里面设定好参数，其所有参数以及其默认值(缺省值)我写在这,内容来自XGBoost源代码：

- **max\_depth=3**, 这代表的是树的最大深度，默认值为三层。max\_depth越大，模型会学到更具体更局部的样本。
- **learning\_rate=0.1**, 学习率，也就是梯度提升中乘以的系数，越小，使得下降越慢，但也是下降的越精确。

- **n\_estimators=100**,也就是弱学习器的最大迭代次数,或者说最大的弱学习器的个数。一般来说n\_estimators太小,容易欠拟合, n\_estimators太大,量会太大,并且n\_estimators到一定的数量后,再增大n\_estimators获得的模型提升会很小,所以一般选择一个适中的数值。默认是100。
- **silent=True**,是我们训练xgboost树的时候后台要不要输出信息, True代表将生成树的信息都输出。
- **objective="binary:logistic"**,这个参数定义需要被最小化的损失函数。最常用的值有:
  1. **binary:logistic** 二分类的逻辑回归, 返回预测的概率(不是类别)。
  2. **multi:softmax** 使用softmax的多分类器, 返回预测的类别(不是概率)。在这种情况下, 你还需要多设一个参数: num\_class(类别数目)。
  3. **multi:softprob**和**multi:softmax**参数一样, 但是返回的是每个数据属于各个类别的概率。
- **nthread=-1**,多线程控制, 根据自己电脑核心数,想用几个线程就可以设定几个, 如果你想用全部核心, 就不要设定, 算法会自动识别
- **gamma=0**,在节点分裂时, 只有分裂后损失函数的值下降了, 才会分裂这个节点。Gamma指定了节点分裂所需的最小损失函数下降值。这个参数大, 算法越保守。这个参数的值和损失函数息息相关, 所以是需要调整的。
- **min\_child\_weight=1**,决定最小叶子节点样本权重和。和GBM的 **min\_child\_leaf** 参数类似, 但不完全一样。XGBoost的这个参数是最小样本权重和而GBM参数是最小样本总数。这个参数用于**避免过拟合**。当它的值较大时, 可以避免模型学习到局部的特殊样本。但是如果这个值过高, 会导致欠拟合。这个参数需要使用CV来调整
- **max\_delta\_step=0**, 决定最小叶子节点样本权重和。和GBM的 **min\_child\_leaf** 参数类似, 但不完全一样。XGBoost的这个参数是最小样本权重和而GBM参数是最小样本总数。这个参数用于**避免过拟合**。当它的值较大时, 可以避免模型学习到局部的特殊样本。但是如果这个值过高, 会导致欠拟合。这个参数需要使用CV来调整。
- **subsample=1**, 和GBM中的subsample参数一模一样。这个参数**控制对于每棵树, 随机采样的比例**。减小这个参数的值, 算法会更加保守, 避免过拟合, 但是, 如果这个值设置得过小, 它可能会导致欠拟合。典型值: 0.5-1
- **colsample\_bytree=1**, 用来控制每棵树随机采样的列数的占比(每一列是一个特征)。典型值: 0.5-1
- **colsample\_bylevel=1**,用来控制树的每一级的每一次分裂, 对列数的采样的占比。其实subsample参数和colsample\_bytree参数可以起到相似的作用
- **reg\_alpha=0**,权重的L1正则化项。(和Lasso regression类似)。可以应用在很高维度的情况下, 使得算法的速度更快。
- **reg\_lambda=1**, 权重的L2正则化项这个参数是用来控制XGBoost的正则化部分的。这个参数越大就越可以惩罚树的复杂度
- **scale\_pos\_weight=1**,在各类别样本十分不平衡时, 把这个参数设定为一个正值, 可以使

- **base\_score**=0.5, 所有实例的初始化预测分数, 全局偏置; 为了足够的迭代次数, 改变这个值将不会有太大的影响。
- **seed**=0, 随机数的种子设置它可以复现随机数据的结果, 也可以用于调整参数

数据通过树生成预测结果

```
1 predictions = gbm.predict(test_X)
2 复制代码
```

将tests里面的数据通过这生成好的模型, 得出预测结果。

```
1 submission = pd.DataFrame({'row_id': tests['row_id'],
2                             'shop_id': predictions})
3 print(submission)
4 submission.to_csv("submission.csv", index=False)
5 复制代码
```

将预测结果写入到csv文件里。我们注意写入文件的格式, row\_id在前, shop\_id在后。index=False的意思是不写入行的名称。改成True就把每一行的行标标了。

---

## 附录

参考资料

1. 机器学习系列(12)\_XGBoost参数调优完全指南 (附Python代码) [http://blog.csdn.net/han\\_xiaoyang/article/details/52665396](http://blog.csdn.net/han_xiaoyang/article/details/52665396)
2. Kaggle比赛: 泰坦尼克之灾: <https://www.kaggle.com/c/titanic>

---

完整代码


```
1 import pandas as pd
2 import xgboost as xgb
3 from sklearn import preprocessing
4
5
6 train = pd.read_csv(r'D:\mall_location\train.csv')
7 tests = pd.read_csv(r'D:\mall_location\test.csv')
8
9 train['time_stamp'] = pd.to_datetime(pd.Series(train['time_stamp']))
10 tests['time_stamp'] = pd.to_datetime(pd.Series(tests['time_stamp']))
11
12 print(train.info())
13
14 train['Year'] = train['time_stamp'].apply(lambda x: x.year)
15 train['Month'] = train['time_stamp'].apply(lambda x: x.month)
16 train['weekday'] = train['time_stamp'].apply(lambda x: x.weekday())
17 train['time'] = train['time_stamp'].dt.time
18 tests['Year'] = tests['time_stamp'].apply(lambda x: x.year)
19 tests['Month'] = tests['time_stamp'].apply(lambda x: x.month)
20 tests['weekday'] = tests['time_stamp'].dt.dayofweek
21 tests['time'] = tests['time_stamp'].dt.time
22 train = train.drop('time_stamp', axis=1)
23 train = train.dropna(axis=0)
24 tests = tests.drop('time_stamp', axis=1)
25 tests = tests.fillna(method='pad')
26 for f in train.columns:
```

```
27 |         if train[f].dtype=='object':28 |             if f != 'shop_id':
29 |                 print(f)
30 |                 lbl = preprocessing.LabelEncoder()
31 |                 train[f] = lbl.fit_transform(list(train[f].values))
32 | for f in tests.columns:
33 |     if tests[f].dtype == 'object':
34 |         print(f)
35 |         lbl = preprocessing.LabelEncoder()
36 |         lbl.fit(list(tests[f].values))
37 |         tests[f] = lbl.transform(list(tests[f].values))
38 |
39 |
40 | feature_columns_to_use = ['Year', 'Month', 'weekday',
41 | 'time', 'longitude', 'latitude',
42 | 'wifi_id1', 'wifi_strong1', 'con_sta1',
43 | 'wifi_id2', 'wifi_strong2', 'con_sta2',
44 | 'wifi_id3', 'wifi_strong3', 'con_sta3',
45 | 'wifi_id4', 'wifi_strong4', 'con_sta4',
46 | 'wifi_id5', 'wifi_strong5', 'con_sta5',
47 | 'wifi_id6', 'wifi_strong6', 'con_sta6',
48 | 'wifi_id7', 'wifi_strong7', 'con_sta7',
49 | 'wifi_id8', 'wifi_strong8', 'con_sta8',
50 | 'wifi_id9', 'wifi_strong9', 'con_sta9',
51 | 'wifi_id10', 'wifi_strong10', 'con_sta10',]
52 |
53 | big_train = train[feature_columns_to_use]
54 | big_test = tests[feature_columns_to_use]
55 | train_X = big_train.as_matrix()
56 | test_X = big_test.as_matrix()
57 | train_y = train['shop_id']
58 |
59 | gbm = xgb.XGBClassifier(silent=1, max_depth=10,
60 |                         n_estimators=1000, learning_rate=0.05)
61 | gbm.fit(train_X, train_y)
62 | predictions = gbm.predict(test_X)
63 |
64 | submission = pd.DataFrame({'row_id': tests['row_id'],
65 |                           'shop_id': predictions})
66 | print(submission)
67 | submission.to_csv("submission.csv",index=False)
```

作者：香橙云子  
链接：<https://juejin.im/post/5a1bb29e51882531ba10aa49>  
来源：掘金  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。  
转载：<https://juejin.im/post/5a1bb29e51882531ba10aa49>

Python爬虫全栈教学，零基础教你成编程大神

零基础学爬虫，你要掌握学习那些技能？



想对作者说点什么

- XGboost数据比赛实战之调参篇(完整流程) - HuangQinJian

这篇博客的内容是在上一篇博客Scikit中的特征选择，XGboost进行回归预测，模型优化的实战的基...

3183

来自：HuangQinJian
- xgboost 实战以及源代码分析 - 木东的博客

1.序 距离上一次编辑将近10个月，幸得爱可可老师（微博）推荐，访问量陡增。最近毕业论文与...

4900

来自：木东的博客



机器学习案例**实战第六课-PCA与Xgboost**

学院

适合人群: 所有人,章节: Xgboost实例演示

**不用离家，区块链开发八周学会！**

区块链DApp开发学习大纲免费领

**XGBoost使用教程（纯xgboost方法） — - 飘过的春风** 👁 1.8万

一、导入必要的工具包# 导入必要的工具包 import xgboost as xgb # 计算分类正确率 from sklearn.me... 来自: [飘过的春风](#)

**xgboost入门与实战（实战调参篇） - hczheng的专栏** 👁 4.3万

xgboost入门与实战（实战调参篇）前言前面几篇博文都在学习原理知识，是时候上数据上模型跑一跑... 来自: [hczheng的专栏](#)

**XGBoost实战与调优 - 法相的博客** 👁 2129

首先，python和Anaconda都没有自带xgboost。windows下安装xgboost非常方便。在前面的文章中， ... 来自: [法相的博客](#)

**xgboost算法原理与实战 - JasonZhangOO的博客** 👁 2.5万

xgboost算法原理与实战之前一直有听说GBM，GBDT（Gradient Boost Decision Tree）渐进梯度决策... 来自: [JasonZhangOO的博客](#)

**XGBoost调参技巧（二）Titanic**实战**Top9% - c2a2o2的专栏** 👁 2348

学习Kaggle的第一个比赛就是Titanic，断断续续的半年时间，从小白到杀入9%。XGBoost果真是Kag... 来自: [c2a2o2的专栏](#)

**蜂蜜的功效和作用？正宗的蜂蜜多少钱？养蜂人教你如何区分！**

智数互动 · 顶新

**机器学习xgboost**实战**—手写数字识别 - Eddy\_zheng的博客** 👁 1.7万

1、xgboost 安装安装问题这里就不再做赘述，可参考前面写的博文： [http://blog.csdn.net/eddy\\_zhen...](http://blog.csdn.net/eddy_zhen...) 来自: [Eddy\\_zheng的博客](#)

**相关热词** [vuejs手把手](#) [手把手微信支付](#) [手把手 fpga](#) [手把手web](#) [手把手阿里云](#)

**lgbm和xgboost使用教程 - owenfy的博客** 👁 3884

# coding: utf-8 # pylint: disable = invalid-name, C0111 import lightgbm as lgb import pandas as pd f... 来自: [owenfy的博客](#)



白马负金羁

[关注](#) 365篇文章



宿永杰

[关注](#) 672篇文章



androidstarjack

[关注](#) 218篇文章

**xgboost学习样例解析之binary classification - xunileida的专栏** 👁 824


玩kaggle时听说这个库很牛逼，准确率很高，于是打算学学看看。xgboost是extreme gradient boostin... 来自: [xunileida的专栏](#)

**Android 滚动事件 OnScrollListener - koterror的专栏** 👁 2495

From 网站:<http://blog.csdn.net/qeqeqe236/article/details/7289112> SCROLL\_STATE\_FLING是指手指... 来自: [koterror的专栏](#)

**Android测试之设备化测试（Instrumented Tests） - Java上下求索的专栏** 👁 2234

当我们需要使用到安卓框架的时候，也就是android.jar里面的api的时候，使用本地单元测试的方式就... 来自: [Java上下求索的专栏](#)



**发现了一个免费的云服务器,号称是永久的**

百度广告

**Vue.js入门 - ☐安装 - 辰煦媛的前端日常** 👁 3622

Vue.js入门系列，安装教程 来自: [辰煦媛的前端日常](#)

**求大神解决，已困扰两天，python，unittest测试结果为Ran 0 tests in 0.000s - qq\_21854...** 👁 1399

testadd\_run.py #coding=utf-8 import unittest from match\_ import Match class Test\_match(unittest.... 来自: [qq\\_21854029的博客](#)

XGBoost**实战完全总结（下）** - xiaoliuhexiaolu的博客


第二部分：实战xgboost既可以通过自己本身的接口进行训练，也可以借助sklearn的接口训练，下面... 来自： xiaoliuhexiaolu的博客

xgboost**实战与导论**

xgboost文件中含有推导，代码等，不过如果想要完整研究，建议看陈天奇完整论文。此方法在数据量较小时，就目前我所知，就xgbo...

xgboost**实战讲义** - refuil

xgboost实战课程，数据分析机器学习GDBT xgboost实战ppt



**走出抑郁的泥潭**  
百度广告

xgboost**导读与实战** - nyzwt

适用初学者，快速入门xgboost。主要内容包含传统GBDT理论和调参训练

xgboost**入门与实战** - qq\_34218221的博客

https://blog.csdn.net/sb19931201/article/details/52557382 来自： qq\_34218221的博客

**手把手教你React Native 实战之开山篇《一》 - 这个时代，作为程序员可能要学习小程序**

先说一下我为什么学习RN 18年3月29号，随着自己内心的欲望和冲动，任务交接了一下，正式离开一... 来自： 这个时代，作为程序...

下载

**React Native 视频教程-电商项目**实战****

02-22

React Native 视频教程-电商项目实战 React Native 视频教程-电商项目实战 React Native 视频教程-电商项目实战

下载

**React Native 学习教程及电商项目**实战**全套视频教程 - 夕ゝ**

05-08

React Native 学习教程及电商项目实战全套视频教程。。

**恨不在北京！11月起，北京上班族可在职申请提升大学学历！**  
尚德学历中心·顶新

**手把手教你在Excel使用VLOOKUP来进行列查找 - 白马负金羁**

VLOOKUP函数是Excel中的一个纵向查找函数，它与LOOKUP函数和HLOOKUP函数属于一类函数， ... 来自： 白马负金羁

下载

**手把手教你学arm入门篇和提高篇》**

08-11

《手把手教你学arm入门篇和提高篇》种子下载，详细了解arm，及学习arm

**xgboost入门与实战（实战调参篇）** 标签： **xgboostpythonkaggle机器学习** - u011089523... 7531

xgboost入门与实战（实战调参篇） 原文地址 前言 前面几篇博文都在学习原理知识，是时候上数据上... 来自： u011089523的博客

**xgboost入门与实战（原理篇）** - hczheng的专栏 15.1万

xgboost入门与实战（原理篇）前言： xgboost是大规模并行boosted tree的工具，它是目前最快最好... 来自： hczheng的专栏

**史上最详细的XGBoost**实战** - 学习AI算法，请关注微信公众号：机器学习算法全栈工程师.....**

0. 环境介绍 Python 版本： 3.6.2 操作系统 Windows 集成开发环境： PyCharm 1. 安装Python... 来自： 学习AI算法，请关注...



**参加自主招生需要满足哪些条件**  
百度广告

**史上最详细的XGBoost**实战**（上） - 燕哥带你学算法**

作者：章华燕 编辑：祝鑫泉 零环境介绍： Python版本： 3.6.2 · 操作系统：Windows · 集成开发环境... 来自： 燕哥带你学算法

**机器学习系列(12)\_XGBoost参数调优完全指南（附Python代码） - 寒小阳**

这篇文章主要讲了如何提升XGBoost模型的表现。首先，我们介绍了相比于GBM，为何XGBoost可以... 来自： 寒小阳

东方耀

手把手教React Native实战开发视频教程共237集 - 蒂花之秀

358

东方耀老师的全套react native 开发视频 高清 ReactNative 官方视 频 教 程 需 要 的 联 系QQ:277487...

来自: 蒂花之秀

手把手教“MFC版贪吃蛇教程” - huang3838438的专栏

8540

写在前面的话 本次贪吃蛇教程主要知识点包括以下几个方面 1 CView类中的消息响应 2 控...

来自: huang3838438的专栏

Python机器学习之XGBoost从入门到实战(基本理论说明) - 雪域枫蓝的博客

416

Xgboost从基础到实战XGBoost:eXtreme Gradient Boosting \* 应用机器学习领域的一个强有力的工具 \*...

来自: 雪域枫蓝的博客



英国大学综合排名

百度广告

执行代码提示: Ran 0 tests in 0.000s - yihongyuantufei的博客

286

执行结果提示: Ran 0 tests in 0.000s, 而看不到打开网页执行代码过程解决方法: unittest框架中, m...

来自: yihongyuantufei的博客

东方耀 手把手教React Native实战开发视频教程+源码笔记 1-237集 - 凌云社区

180

课程序号标题第0课0、手把手教React Native实战之开山篇\_视频第1课1、手把手教React Native实战...

来自: 凌云社区

xgboost原理实战【转】 - wang603603的专栏

54

https://www.cnblogs.com/-Sai-/p/6723556.html 原理 xgboost的原理首先看xgboost的作者陈天奇的ppt...

来自: wang603603的专栏

手把手教你串口仪器控制（基础篇）Labview串口通信详解 - 豚

490

http://www.51hei.com/bbs/dpj-94933-1.html

来自: 豚

下载

c++新手必看，手把手教你c++

03-04

C++实验指导书。通过实验过程教会你c++的使用，很基础

恨不在北京！11月起，北京上班族可在职申请提升大学学历！

尚德学历中心 · 顶新

Python机器学习之XGBoost从入门到实战(代码实现) - 雪域枫蓝的博客

717

# -\*- coding: utf-8 -\*- \_\_author\_\_ = 'gerry' XGBoost案例之蘑菇是否有毒 任务：根据蘑菇的22个特征...

来自: 雪域枫蓝的博客

揭秘Kaggle神器xgboost - 维尼弹着肖邦的夜曲

2867

http://geek.csdn.net/news/detail/201207 XGBoost : eXtreme Gradient Boosting 项目地址: https://gi...

Pandas使用DataFrame进行数据分析比赛进阶之路（一） - HuangQinJian

1527

这篇文章中使用的数据集是一个足球球员各项技能及其身价的csv表，包含了60多个字段。数据集下载...

来自: HuangQinJian

下载

xgboost源代码python - 陈同学喜欢吃面包

11-27

根据我的课程设计写了一个xgboost代码，效果不错。希望能帮助跟我一样的初学者。

xgboost特征选择 - 芜湖小老板的BLOG

2.5万

Xgboost在各大数据挖掘比赛中是一个大杀器，往往可以取得比其他各种机器学习算法更好的效果。数...

来自: 芜湖小老板的BLOG

我们对天发誓：这正版传奇爆率9.8，送VIP，卸载算我输！

贪玩游戏 · 顶新

XGBoost训练流程 - 小菜鸡的博客

86

1.调用流程 import xgboost as xgb from xgboost.sklearn import XGBClassifier from sklearn.grid\_sear...

来自: 小菜鸡的博客

java对象存储方式 - u010507622的专栏

937

1.寄存器。这是最快的存储区，因为它是直接存储在处理器内部，由于寄存器的数量有限，寄存器会...

来自: u010507622的专栏

- 工作笔记3.手把手教你搭建SSH(struts2+hibernate+spring)环境 - chestnut

本文以搭建SSH(struts2+hibernate+spring)框架为例，通过3个独立配置、2个整合，基本完成SSH框...

来自：chestnut
- ubuntu10.04编译安装QT可能出现的问题和解决办法 - situzhuge的专栏

ubuntu10.04编译安装QT可能出现的问题和解决办法2010-06-21 14:13./configure --prefix=/usr 如出现...

来自：situzhuge的专栏
- xgboost优化 - jinruoyanxu的博客

译注：文内提供的代码和运行结果有一定差异，可以从这里下载完整代码对照参考。另外，我自己跟...

来自：jinruoyanxu的博客

我们对天发誓：这正版传奇爆率9.8，送VIP，卸载算我输！

贪玩游戏 · 顶新

- PyCharm中出现No tests were found的一种解决方法及PyCharm建立测试类的官方原文及...

惭愧，将近一年没进这博客了。一年多虽然学了不少，但是根本无心把东西放在这里。有时候想想也...

来自：DROIDEYE



九城风雪

关注

原创9

粉丝11

喜欢5

评论0

等级：博客 2

积分：252

访问：8280

排名：33万+

最新文章

leetcode 20. Valid Parentheses

1.twosum

pandas ix & iloc & loc 的联系和区别

pandas 重新索引

pandas中关于set\_index和reset\_index的用法

个人分类

数据处理11篇

天池大赛2篇

机器学习算法11篇

Ubuntu4篇

linux1篇

展开

https://blog.csdn.net/weixin\_42029738/article/details/81675234

12/13

归档

2018年10月

2篇

2018年9月

4篇

2018年8月

14篇

2018年7月

13篇

2018年5月

15篇

展开

种头发危害



联系我们



扫码联系客服



下载CSDN APP

 QQ客服

 kefu@csdn.net

 客服论坛

 400-660-0108

工作时间 8:00-22:00

关于我们 招聘 广告服务 网站地图

 百度提供站内搜索 京ICP证09002463号

©2018 CSDN版权所有

网络110报警服务 经营性网站备案信息

北京互联网违法和不良信息举报中心

中国互联网举报中心