大体规划

2020年5月4日

21:31

**✔数据挖掘流程：**

（一）数据读取：

* 1. 读取数据，并进行展示
  2. 统计数据各项指标
  3. 明确数据规模与要完成任务

（二）特征理解分析

* 1. 单特征分析，逐个变量分析其对结果的影响
  2. 多变量统计分析，综合考虑多种情况影响
  3. 统计绘图得出结论

（三）数据清洗与预处理

* 1. 对缺失值进行填充
  2. 特征标准化/归一化
  3. 筛选有价值的特征
  4. 分析特征之间的相关性

（四）建立模型

* 1. 特征数据与标签准备
  2. 数据集切分
  3. 多种建模算法对比
  4. 集成策略等方案改进

**✔数据挖掘具体流程：**

* 1. EDA（探索性数据分析）
  2. 载入各种数据科学以及可视化库:
     + 数据科学库 pandas、numpy、scipy；
     + 可视化库 matplotlib、seabon；
     + 其他；
  3. 载入数据：
     + 载入训练集和测试集；
     + 简略观察数据(head()+shape)；
  4. 数据总览:
     + 通过describe()来熟悉数据的相关统计量
     + 通过info()来熟悉数据类型
  5. 判断数据缺失和异常
     + 查看每列的存在nan情况
     + 异常值检测
  6. 了解预测值的分布
     + 总体分布概况（无界约翰逊分布等）
     + 查看skewness and kurtosis
     + 查看预测值的具体频数
  7. 特征分为类别特征和数字特征，并对类别特征查看unique分布

数字特征分析

* 1. 相关性分析
  2. 查看几个特征得 偏度和峰值
  3. 每个数字特征得分布可视化
  4. 数字特征相互之间的关系可视化
  5. 多变量互相回归关系可视化

类型特征分析

* 1. unique分布
  2. 类别特征箱形图可视化
  3. 类别特征的小提琴图可视化
  4. 类别特征的柱形图可视化类别
  5. 特征的每个类别频数可视化(count\_plot)
  6. 用pandas\_profiling生成数据报告

* + 特征工程
  + 异常处理：
    - 通过箱线图（或 3-Sigma）分析删除异常值；
    - BOX-COX 转换（处理有偏分布）；
    - 长尾截断；
  + 特征归一化/标准化：
    - 标准化（转换为标准正态分布）；
    - 归一化（抓换到 [0,1] 区间）；
    - 针对幂律分布，可以采用公式： log(1+x1+median)log(1+x1+median)
  + 数据分桶：
    - 等频分桶；
    - 等距分桶；
    - Best-KS 分桶（类似利用基尼指数进行二分类）；
    - 卡方分桶；
  + 缺失值处理：
    - 不处理（针对类似 XGBoost 等树模型）；
    - 删除（缺失数据太多）；
    - 插值补全，包括均值/中位数/众数/建模预测/多重插补/压缩感知补全/矩阵补全等；
    - 分箱，缺失值一个箱；
  + 特征构造：
    - 构造统计量特征，报告计数、求和、比例、标准差等；
    - 时间特征，包括相对时间和绝对时间，节假日，双休日等；
    - 地理信息，包括分箱，分布编码等方法；
    - 非线性变换，包括 log/ 平方/ 根号等；
    - 特征组合，特征交叉；
    - 仁者见仁，智者见智。
  + 特征筛选
    - 过滤式（filter）：先对数据进行特征选择，然后在训练学习器，常见的方法有 Relief/方差选择发/相关系数法/卡方检验法/互信息法；
    - 包裹式（wrapper）：直接把最终将要使用的学习器的性能作为特征子集的评价准则，常见方法有 LVM（Las Vegas Wrapper） ；
    - 嵌入式（embedding）：结合过滤式和包裹式，学习器训练过程中自动进行了特征选择，常见的有 lasso 回归；
  + 降维
    - PCA/ LDA/ ICA；
    - 特征选择也是一种降维。

* + 模型融合
  + 简单加权融合:
    - 回归（分类概率）：算术平均融合（Arithmetic mean），几何平均融合（Geometric mean）；
    - 分类：投票（Voting)
    - 综合：排序融合(Rank averaging)，log融合
  + stacking/blending:
    - 构建多层模型，并利用预测结果再拟合预测。
  + boosting/bagging（在xgboost，Adaboost,GBDT中已经用到）:
    - 多树的提升方法

**✔近期的大体学习方案**

* + 复现冠军方案，结合自己的思考
  + 复现，认真揣摩大佬的比赛笔记和比赛感悟视频，结合自己的思考

方法

**数据读取和认识**

2020年5月4日

21:45

# 初始化

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

plt.style.use('ggplot')

color = sns.color\_palette()

import warnings

warnings.filterwarnings('ignore')

%matplotlib inline

Imo

27个隐藏的空间

#绘画风格

# set plot style

plt.style.use('ggplot') #模仿R语言绘图风格

plt.style.use(‘fivethirtyeight’) #线条变粗

Plt.style.use (‘ seaborn’) #seaborn绘图风格

#得到plt.styte的值

print(plt.style.available)

# 颜色

color = sns.color\_palette()

#查看数据格式

Path='路径'

open(path).readline()#读取一行

Tips：批量导入多个有规律命名的文件

pieces=[]

for year in range(1880,2011):

path='pydata-book-2nd-edition/datasets/babynames/yob%d.txt' % year

frame= pd.read\_csv(path,names=columns)

frame['year']=year

pieces.append(frame)

Tip:

#被数量不同的空白字符间隔开的，正则表达式表达为\s+

#无表头数据加两个属性：header=None表头, names=列表

#两种形式的json文件

文件多行格式

records = [json.loads(line) for line in open(path)]

文件是一行而已

records =json.load(path)

#### *h5文件*

##### #读取写入

import h5py #导入工具包

f = h5py.File('HDF5\_FILE.h5','w') #创建一个h5文件，文件指针是f

f['labels'] = range(100) #将数据写入文件的主键labels下面

f.close() #关闭文件

f = h5py.File('HDF5\_FILE.h5','r') #打开h5文件

f.keys() #可以查看所有的主键

##### #转化为dataframe

train =pd.read\_hdf(TRAIN\_PATH,key='df')

# 数据读取

* 标准数据读取

data=pd.read\_csv('train.csv',sep='default=',' ' encoding = "gbk")

users = pd.read\_table(文件路径, sep='分隔符',

header=None表头, names=列表)

* [pd.read\_csv参数设置](https://blog.csdn.net/weixin_44056331/article/details/89366105)
  + 中文乱码设定encoding = "gbk"
  + nrows： int 读取的行数
  + header: 指定第几行作为列名(忽略注解行)，如果没有指定列名，默认header=0; 如果指定了列名header=None
  + sep: 指定分割符，默认是’,’C引擎不能自动检测分隔符，但Python解析引擎可以
* json数据读取

import json

records = [json.loads(line) for line in open(path)]

frame = pd.DataFrame(records)

* 查看数据

data.head()

/data.tail()

* 去除打印的省略号：np.set\_printoptions(threshold = 1e6)
* 设置dataframe的显示格式
  + value的显示长度为100，默认为50

pd.set\_option('max\_colwidth',100)

* 显示所有列

pd.set\_option('display.max\_columns', None)

* 显示所有行

pd.set\_option('display.max\_rows', None)

[pd.set\_option参数设置](https://www.cnblogs.com/cxxBoo/p/12531437.html)

X\_data[['regionCode']]（dataframe）

149995 
149996 
149997 
149998 
regionCode 
1046 
4366 
2806 
4576 
2826 
3302 
150000 rows x 1 columns 

X\_data['regionCode']（serial）

计算机生成了可选文字:
0
2
3
4
149995
149996
149997
149998
149999
1046
4366
2806
434
6g77
4576
2826
3302
1877
235
Name．regionCode
Length：
150000，
dtype
int16

# 训练集和测试集融合

#merge train\_set and test\_set

data\_train["oringin"]="train"

data\_test["oringin"]="test"

data\_all=pd.concat([data\_train,data\_test],axis=0,ignore\_index=True)

# 数据初略认识

Train\_data.head().append(Train\_data.tail())#查看数据前五行和后五行

data.describe() #表格数据初略计算，平均值等

data.info() #表格基本信息

data.isnull().sum() #查看缺失值、

data.columns #查看列名

#数据特征基本理解，举例如下：

离散值：性别（男，女） 登船地点（S,Q,C）

连续值：年龄，船票价格

#数据分布初步认识（如正态分布等）

sns.distplot(df\_train['SalePrice'])

0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 
0 

# 解决中文显示问题

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

区分连续型和离散型特征

#普通区分方法（只适用于没有直接label coding的数据）

方法一

#数字特征

numeric\_features = Train\_data.select\_dtypes(include=[np.number])

numeric\_features.columns

#类型特征

categorical\_features = Train\_data.select\_dtypes(include=[np.object])

categorical\_features.columns

方法二

numerical\_fea = list(data\_train.select\_dtypes(exclude=['object']).columns)

category\_fea = list(filter(lambda x: x not in numerical\_fea,list(data\_train.columns)))

# 单变量分析（绘图）

方法一：列名\_counts = frame['列名'].value\_counts()#查看某栏值的个数和分布情况

方法二：直接用图描绘出所有所有列对应值的情况，举例如下：

## #手工绘制连续变量：

##### 方法一（df.hist）

colnm = df.columns.tolist()

plt.figure(figsize = (10, 8))

for i in range(12):

plt.subplot(4,3,i+1)

plt.xlabel(colnm[i],fontsize = 12)

df[colnm[i]].hist(bins = 100, color = color[0])

plt.ylabel('Frequency')

plt.tight\_layout()

##### 方法二（sns.kdeplot）

# Explore feature distibution

#fig = plt.figure(figsize=(6, 6))

for column in data\_all.columns[0:-2]:

g = sns.kdeplot(data\_all[column][(data\_all["oringin"] == "train")], color="Red", shade = True)

g = sns.kdeplot(data\_all[column][(data\_all["oringin"] == "test")], ax =g, color="Blue", shade= True)

g.set\_xlabel(column)

g.set\_ylabel("Frequency")

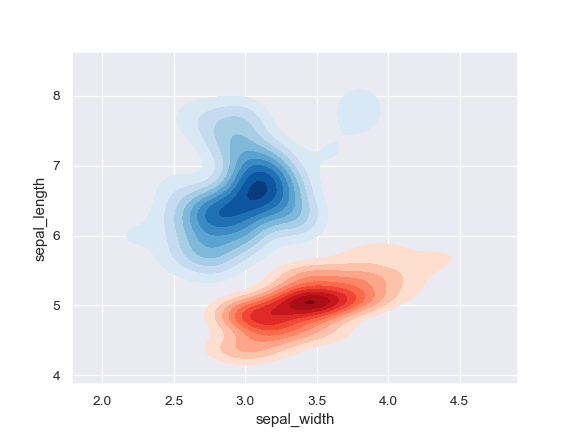
g = g.legend(["train","test"])

plt.show()

main 
06 
05 
04 
02 

##### [拓展(官方文档）](https://www.cntofu.com/book/172/docs/25.md)

iris = sns.load\_dataset("iris")  
setosa = iris.loc[iris.species == "setosa"]  
virginica = iris.loc[iris.species == "virginica"]  
ax = sns.kdeplot(setosa.sepal\_width, setosa.sepal\_length,  
 cmap="Reds", shade=**True**, shade\_lowest=**False**)  
ax = sns.kdeplot(virginica.sepal\_width, virginica.sepal\_length,  
 cmap="Blues", shade=**True**, shade\_lowest=**False**)



## #手工绘制离散变量：

colnms = df.columns.tolist()

plt.figure(figsize = (10, 8))

for i in range(12):

plt.subplot(4,3,i+1)

sns.countplot(df[colmns[i]], color = color[0])

plt.xlabel(colnm[i],fontsize = 12)

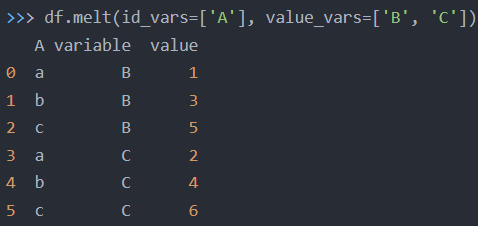
plt.ylabel('Frequency')

plt.tight\_layout()

## [#结构化多绘图网格](https://blog.csdn.net/weixin_42398658/article/details/82960379)

### 举例一

#如果说 df.pivot() 将长数据集转换成宽数据集，[df.melt()](https://blog.csdn.net/mingkoukou/article/details/82867218) 则是将宽数据集变成长数据集



f = pd.melt(Train\_data, value\_vars=numeric\_features)

g = sns.FacetGrid(f, col="variable", col\_wrap=2, sharex=False, sharey=False)

g = g.map(sns.distplot, "value")

variable = pnce 
000020 
00001 s 
000010 
variable = kilometer 
variable V I 
variable = v 3 
0 0025 
00020 
00015 
00010 
00005 
0150 
0100 
0075 
0050 
0025 
variable = power 
variable = v O 
variable v 2 
variable = v 4 

### 举例二

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

for i in range(len(data\_all.columns)-2):

g = sns.FacetGrid(data\_all, col='oringin')

g = g.map(sns.distplot, data\_all.columns[i])

06 
04 
02 
08 
06 
oringin = train 
oringin • train 
oringin = test 
oringin • test 

拓展：

ordered\_days = tips.day.value\_counts().index

g = sns.FacetGrid(tips, row="day", row\_order=ordered\_days,

height=1.7, aspect=4)

g.map(sns.distplot, "total\_bill", hist=False, rug=True)

day • sat 
day • Sun 
0023 
day • 
btal_biH 

# 分析连续变量在测试集和训练集的分布

#### *举例一*

for column in data\_all.columns[0:-2]:

    g = sns.kdeplot(data\_all[column][(data\_all["oringin"] == "train")], color="Red", shade = True)

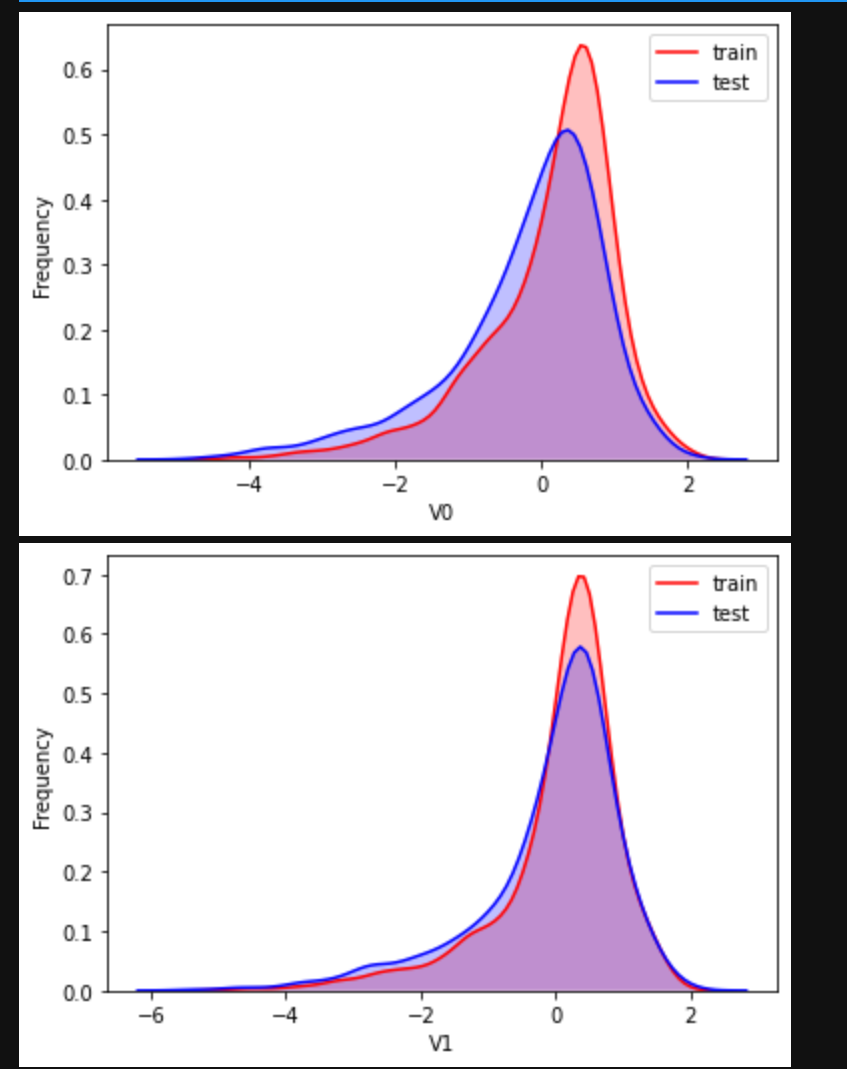
    g = sns.kdeplot(data\_all[column][(data\_all["oringin"] == "test")], ax =g, color="Blue", shade= True)

    g.set\_xlabel(column)

    g.set\_ylabel("Frequency")

    g = g.legend(["train","test"])

    plt.show()



#### *举例二*

fig = plt.figure(figsize=(10, 10))

for i in range(len(data\_all.columns)-2):

    g = sns.FacetGrid(data\_all, col='oringin')

    g = g.map(sns.distplot, data\_all.columns[i])

oringin = train 
orinqin train 
orinqin train 
or •nq'n = test 
orinqn test 
oringm • test 

## sns.distplot和sns.regplot分析单变量分布（连续变量）

### 资料：

[sns.distplot()用法](https://www.sohu.com/a/158933070_718302)

[Python可视化 | Seaborn5分钟入门(一)——kdeplot和distplot](https://blog.csdn.net/qq_39949963/article/details/79362501)

[sns.regplot()的用法](https://www.cnblogs.com/cgmcoding/p/13293395.html)

### 举例：

# figure parameters

data\_train1=data\_all[data\_all["oringin"]=="train"].drop("oringin",axis=1)

fcols = 2

frows = len(data\_train.columns)

plt.figure(figsize=(5\*fcols,4\*frows))

i=0

for col in data\_train1.columns:

i+=1

ax=plt.subplot(frows,fcols,i)

sns.regplot(x=col, y='target', data=data\_train, ax=ax,

scatter\_kws={'marker':'.','s':3,'alpha':0.3},

line\_kws={'color':'k'});

plt.xlabel(col)

plt.ylabel('target')

i+=1

ax=plt.subplot(frows,fcols,i)

sns.distplot(data\_train[col].dropna() , fit=stats.norm)

plt.xlabel(col)

07 
06 
05 
04 
03 
02 
οι 
οο 
08 
07 
06 
05 
04 
03 
02 
οο 
-4 
-3 

# 数据分布分析 （偏度和峰值）

## #label总体分布概况（无界约翰逊分布等）

import scipy.stats as st

y = Train\_data['price']

plt.figure(1); plt.title('Johnson SU')

sns.distplot(y, kde=False, fit=st.johnsonsu)

plt.figure(2); plt.title('Normal')

sns.distplot(y, kde=False, fit=st.norm)

plt.figure(3); plt.title('Log Normal')

sns.distplot(y, kde=False, fit=st.lognorm)

plt.figure(4); plt.title('Log transform')

sns.distplot(log(y))



## #得到所有特征的偏度和峰值

data\_skew=Train\_data.skew()

data\_krut=Train\_data.kurt()

data\_skew\_kurt = pd.DataFrame({'feature':Train\_data.columns,'Skewness':data\_skew.values,'Kurtosis':data\_krut.values}).round(2)

data\_skew\_kurt

## #画出所有特征的偏度和峰值的分布图

plt.figure(1); plt.title('skewness')

sns.distplot(data\_skew,color='blue',axlabel ='Skewness')

plt.figure(2); plt.title('Kurtness')

sns.distplot(data\_krut,color='orange',axlabel ='Kurtness')

# 特征分析

## #连续特征分析

#相关性分析

price\_numeric = Train\_data[numeric\_features]

correlation = price\_numeric.corr()

correlation['price'].sort\_values(ascending = False).rename('price\_corr').to\_frame().style.background\_gradient(cmap='summer')

price 
v_12 
power 
v_ll 
kilometer 
price_c orr 
1 000000 
0692823 
0685798 
0628397 
0.21 
o. 16431 
o, 085 
00689 
00609 
0.0359 
_0013 
-0053 
-0147 
-0206 
-02461 
-0275 
4405 
-0730946 

df\_temp=df[df[colmn[i]].value\_counts()

sns.barplot(x=df\_temp.index,y=df\_temp.values)

df\_temp=df[colmn[i]]

sns.distplot(df\_temp)

特征工程

2020年11月24日

9:57

[数据预处理](https://www.jianshu.com/p/e2d705e9f38d)

2020年5月6日

15:01

# 基本认识

# 缺失值填充

## 缺失值可视化

## 注意object类型，可能会有空值

## 对数据分组，再用均值来进行填充

## 根据实际情况填充缺失值

## 缺失值填充常用方法

# 时间格式处理

# 对象类型特征转换到数值

## 过滤数值型类别特征

## object型特征转化为数值

# 离散化连续变量（分箱操作）

## 分箱的意义

## 无监督分箱

### 例一 loc条件填充

data['Age\_band']=0

data.loc[data['Age']<=16,'Age\_band']=0

data.loc[(data['Age']>16)&(data['Age']<=32),'Age\_band']=1

data.loc[(data['Age']>32)&(data['Age']<=48),'Age\_band']=2

data.loc[(data['Age']>48)&(data['Age']<=64),'Age\_band']=3

data.loc[data['Age']>64,'Age\_band']=4

data.head(2)

### 例二 [pd.qcut](https://www.jb51.net/article/164739.htm)等频离散化变量

data['Fare\_Range']=pd.qcut(data['Fare'],4,labels=False)

import pandas as pd

data = pd.Series([0,8,1,5,3,7,2,6,10,4,9])

print(pd.qcut(data,[0, 0.1, 0.2, 0.3, 1],labels=['first 10%','second 10%','third 10%','70%']))

first 
first 
third 
second 
10 
10k 
70k 
dtype: category 
Categories (4, object) 
[first < second < third < 70±1 

### 例三[pd.cut](https://www.cnblogs.com/sench/p/10128216.html)等距离散化变量(labels=False去除空的分箱

data['Age\_band']=pd.cut(data['Age'],5,labels=[0,1,2,3,4])

bin = [i\*10 for i in range(31)]

data['power\_bin'] = pd.cut(data['power'], bin, labels=False)

例四np.floor\_divide等距分箱

#原理：data/q得到一个数，在向下取整，相当于距离为q的等距分箱

# 通过除法映射到间隔均匀的分箱中，每个分箱的取值范围都是loanAmnt/1000

data['loanAmnt\_bin1'] = np.floor\_divide(data['loanAmnt'], 1000)

例五np.floor等数量级分箱

#通过对数函数映射到指数宽度分箱

data['loanAmnt\_bin2'] = np.floor(np.log10(data['loanAmnt']))

## 有监督分箱

## CountEncoder（文本特征编码）

## labelcount编码（比CountEncoder更具有优势）

## target\_encode（最有优势）

## [拓展](https://blog.csdn.net/ssswill/article/details/90271293)

## beta target encoding （[kaggle编码categorical feature总结](https://zhuanlan.zhihu.com/p/40231966)）

## [MeanEncoder平均数编码（大杀器）](https://blog.csdn.net/juzexia/article/details/78581462)

适用范围：

某一个特征是定性的（categorical），而这个特征的可能值非常多（高基数）

高基数定性特征的例子：IP地址、电子邮件域名、城市名、家庭住址、街道、产品号码

源码：

import numpy as np

import pandas as pd

from sklearn.model\_selection import [StratifiedKFold](https://blog.csdn.net/wqh_jingsong/article/details/77896449)

from itertools import product

class MeanEncoder:

def \_\_init\_\_(self, categorical\_features, n\_splits=5, target\_type='classification', prior\_weight\_func=None):

"""

:param categorical\_features: list of str, the name of the categorical columns to encode

:param n\_splits: the number of splits used in mean encoding

:param target\_type: str, 'regression' or 'classification'

:param prior\_weight\_func:

a function that takes in the number of observations, and outputs prior weight

when a dict is passed, the default exponential decay function will be used:

k: the number of observations needed for the posterior to be weighted equally as the prior

f: larger f --> smaller slope

"""

self.categorical\_features = categorical\_features

self.n\_splits = n\_splits

self.learned\_stats = {}

if target\_type == 'classification':

self.target\_type = target\_type

self.target\_values = []

else:

self.target\_type = 'regression'

self.target\_values = None

if isinstance(prior\_weight\_func, dict):

self.prior\_weight\_func = eval('lambda x: 1 / (1 + np.exp((x - k) / f))', dict(prior\_weight\_func, np=np))

elif callable(prior\_weight\_func):

self.prior\_weight\_func = prior\_weight\_func

else:

self.prior\_weight\_func = lambda x: 1 / (1 + np.exp((x - 2) / 1))

@staticmethod

def mean\_encode\_subroutine(X\_train, y\_train, X\_test, variable, target, prior\_weight\_func):

X\_train = X\_train[[variable]].copy()

X\_test = X\_test[[variable]].copy()

if target is not None:

nf\_name = '{}\_pred\_{}'.format(variable, target)

X\_train['pred\_temp'] = (y\_train == target).astype(int) # classification

else:

nf\_name = '{}\_pred'.format(variable)

X\_train['pred\_temp'] = y\_train # regression

prior = X\_train['pred\_temp'].mean()

col\_avg\_y = X\_train.groupby(by=variable, axis=0)['pred\_temp'].agg({'mean': 'mean', 'beta': 'size'})

col\_avg\_y['beta'] = prior\_weight\_func(col\_avg\_y['beta'])

col\_avg\_y[nf\_name] = col\_avg\_y['beta'] \* prior + (1 - col\_avg\_y['beta']) \* col\_avg\_y['mean']

col\_avg\_y.drop(['beta', 'mean'], axis=1, inplace=True)

nf\_train = X\_train.join(col\_avg\_y, on=variable)[nf\_name].values

nf\_test = X\_test.join(col\_avg\_y, on=variable).fillna(prior, inplace=False)[nf\_name].values

return nf\_train, nf\_test, prior, col\_avg\_y

def fit\_transform(self, X, y):

"""

:param X: pandas DataFrame, n\_samples \* n\_features

:param y: pandas Series or numpy array, n\_samples

:return X\_new: the transformed pandas DataFrame containing mean-encoded categorical features

"""

X\_new = X.copy()

if self.target\_type == 'classification':

skf = StratifiedKFold(self.n\_splits)

else:

skf = KFold(self.n\_splits)

if self.target\_type == 'classification':

self.target\_values = sorted(set(y))

self.learned\_stats = {'{}\_pred\_{}'.format(variable, target): [] for variable, target in

[product](https://blog.csdn.net/happyday_d/article/details/86005024)(self.categorical\_features, self.target\_values)}

for variable, target in product(self.categorical\_features, self.target\_values):

nf\_name = '{}\_pred\_{}'.format(variable, target)

X\_new.loc[:, nf\_name] = np.nan

for large\_ind, small\_ind in skf.split(y, y):

nf\_large, nf\_small, prior, col\_avg\_y = MeanEncoder.mean\_encode\_subroutine(

X\_new.iloc[large\_ind], y.iloc[large\_ind], X\_new.iloc[small\_ind], variable, target, self.prior\_weight\_func)

X\_new.iloc[small\_ind, -1] = nf\_small

self.learned\_stats[nf\_name].append((prior, col\_avg\_y))

else:

self.learned\_stats = {'{}\_pred'.format(variable): [] for variable in self.categorical\_features}

for variable in self.categorical\_features:

nf\_name = '{}\_pred'.format(variable)

X\_new.loc[:, nf\_name] = np.nan

for large\_ind, small\_ind in skf.split(y, y):

nf\_large, nf\_small, prior, col\_avg\_y = MeanEncoder.mean\_encode\_subroutine(

X\_new.iloc[large\_ind], y.iloc[large\_ind], X\_new.iloc[small\_ind], variable, None, self.prior\_weight\_func)

X\_new.iloc[small\_ind, -1] = nf\_small

self.learned\_stats[nf\_name].append((prior, col\_avg\_y))

return X\_new

def transform(self, X):

"""

:param X: pandas DataFrame, n\_samples \* n\_features

:return X\_new: the transformed pandas DataFrame containing mean-encoded categorical features

"""

X\_new = X.copy()

if self.target\_type == 'classification':

for variable, target in product(self.categorical\_features, self.target\_values):

nf\_name = '{}\_pred\_{}'.format(variable, target)

X\_new[nf\_name] = 0

for prior, col\_avg\_y in self.learned\_stats[nf\_name]:

X\_new[nf\_name] += X\_new[[variable]].join(col\_avg\_y, on=variable).fillna(prior, inplace=False)[

nf\_name]

X\_new[nf\_name] /= self.n\_splits

else:

for variable in self.categorical\_features:

nf\_name = '{}\_pred'.format(variable)

X\_new[nf\_name] = 0

for prior, col\_avg\_y in self.learned\_stats[nf\_name]:

X\_new[nf\_name] += X\_new[[variable]].join(col\_avg\_y, on=variable).fillna(prior, inplace=False)[

nf\_name]

X\_new[nf\_name] /= self.n\_splits

return X\_new

class\_list = ['model','brand','name','regionCode']+date\_cols

MeanEnocodeFeature = class\_list#声明需要平均数编码的特征

ME = MeanEncoder(MeanEnocodeFeature,target\_type='regression') #声明平均数编码的类

X\_data = ME.fit\_transform(X\_data,Y\_data)#对训练数据集的X和y进行拟合

#x\_train\_fav = ME.fit\_transform(x\_train,y\_train\_fav)#对训练数据集的X和y进行拟合

X\_test = ME.transform(X\_test)#对测试集进行编码

# 异常数据处理

## 根据训练好的模型删除无法拟合的数据

## 删除严重倾斜的特征

## [异常值清洗](https://blog.csdn.net/qq_29339467/article/details/105636205)

## 检查缺失值是否清理干净

# [数据标准化](https://blog.csdn.net/u012768474/article/details/99871942)

## 基本认识

## #纯手工对数变换标准化正态分布

## [#Box-Cox 转换](https://blog.csdn.net/sinat_26917383/article/details/77864582)

## #Z-Score标准化 （可在scale中直接实现）

## #StandardScaler()标准化均值为0，方差为1的正太分布

## #Z-Score和StandardScaler()的区别

## #MinMaxScaler()标准化数据，映射到 [ 0，1 ]

## #RobustScaler()标准化数据（MinMaxScaler思想的延伸，避免outliers的影响）

# 数据归一化（Normalization）

## 根据某一列或几列排列

## 按列进行差分（可以做时间差，位移量，速度，加速度）

## [rank对某列排序](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.rank.html#pandas-series-rank)

tips：

用dataframe.loc[index,colums]进行离散化变量

#独热编码（One-Hot Encoding）和 标签编码

离散特征的编码分为两种情况：

1、离散特征的取值之间没有大小的意义，比如color：[red,blue],那么就使用one-hot编码

2、离散特征的取值有大小的意义，比如size:[X,XL,XXL],那么就使用数值的映射{X:1,XL:2,XXL:3}

使用pandas可以很方便的对离散型特征进行one-hot编码

#特征工程实质

用线性模型，就得通过离散化将非线性特征转化为线性的，这就是所谓特征工程。用非线性模型，比如Tree, DNN就不需要。

Tips:dp.qcut&pd.cut

6 
print( *30) 
print(pd .qcut(factors, 5) 
(-1.917, 1.215) 
(-1.215, 
(-0.516, 
-o. 516] 
0.182] 
(0.182, 0.881] 
(0.881, 1.579] 
dtype: int6đ 
(-1.914, 
(-0.929, 
( -e. 464, 
(-0.0559, 
-0.464] 
-o. 0559] 
0.506) 
4 
le 
3 
6 
6 
6 
(0.506, 1.579] 
dtype: inta 

特征分析

2020年5月4日

22:00

# 探索性数据分析（EDA)

## 资料

[一文带你探索性数据分析(EDA)](https://www.jianshu.com/p/9325c9f88ee6)

[什么是多维度数据分析？](https://www.sohu.com/a/215001437_109461)

## 探索性数据分析（EDA）与传统统计分析（Classical Analysis）的区别

传统的统计分析方法通常是先假设样本服从某种分布，然后把数据套入假设模型再做分析。但由于多数数据并不能满足假设的分布，因此，传统统计分析结果常常不能让人满意。

探索性数据分析方法注重数据的真实分布，强调数据的可视化，使分析者能一目了然看出数据中隐含的规律，从而得到启发，以此帮助分析者找到适合数据的模型。“探索性”是指分析者对待解问题的理解会随着研究的深入不断变化。

## 一张图搞定

# 多变量分析（离散变量）

## #pd.crosstab分析目标特征与其他特征的关系，举例如下：

pd.crosstab(index,columns,margins=True).style.background\_gradient(cmap='summer\_r')

In : 
Out : 
1 pd. crosstab(data. Pclass, data. Survived, margins—True) . style. background_gradient (cmap=' sunmer_r' ) 
Survived 
Pclass 
2 
3 
All 
80 
97 
549 
136 
87 
119 
342 
All 
216 
184 
891 

## #sns.barplot分析目标特征与其他特征的关系，举例如下：

categorical\_features = ['model',

'brand',

'bodyType',

'fuelType',

'gearbox',

'notRepairedDamage']

for c in categorical\_features:#离散变量转化为category类型变量，[原因请点击这里](https://blog.csdn.net/cFarmerReally/article/details/105648222)

Train\_data[c] = Train\_data[c].astype('category')

if Train\_data[c].isnull().any():

Train\_data[c] = Train\_data[c].cat.add\_categories(['MISSING'])

Train\_data[c] = Train\_data[c].fillna('MISSING')

def bar\_plot(x, y, \*\*kwargs):

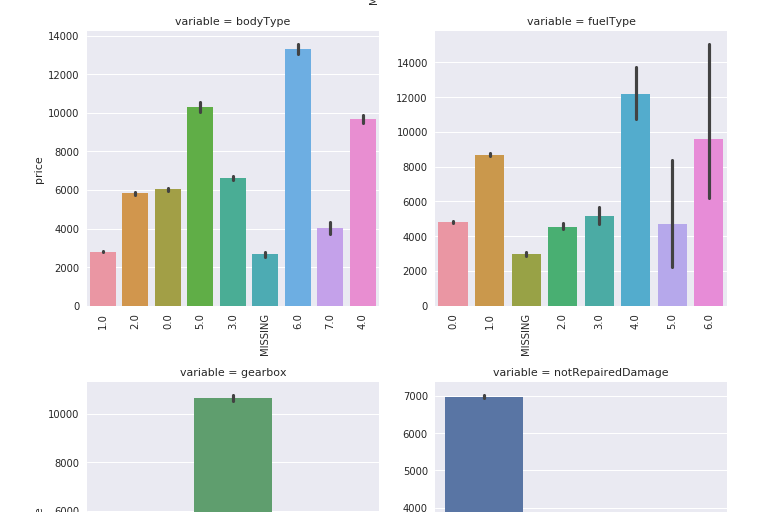
sns.barplot(x=x, y=y)

x=plt.xticks(rotation=90)

f = pd.melt(Train\_data, id\_vars=['price'], value\_vars=categorical\_features)

g = sns.FacetGrid(f, col="variable", col\_wrap=2, sharex=False, sharey=False, size=5)

g = g.map(bar\_plot, "value", "price")



## #sns.countplot分析目标特征与其他特征的关系，举例如下：

f,ax=plt.subplots(1,2,figsize=(18,8))

sns.countplot('Pclass',data=data,ax=ax[1])

ax[0].set\_title('Number Of Passengers By Pclass')

sns.countplot('Pclass',hue='Survived',data=data,ax=ax[1])

ax[1].set\_title('Pclass:Survived vs Dead')

plt.show()

计算机生成了可选文字:
Number Of passengers BY pclass 
Pclass:Survived vs Dead 
Survived 
350 
300 
250 
c 200 
150 
100 
Pclass 

## #sns.factorplot分析目标特征与其他特征的关系，举例如下：

sns.factorplot('Pclass','Survived',hue='Sex',data=data)

plt.show()

计算机生成了可选文字:
08 
pc\ass 

## #sns.violinplot分析目标特征与其他特征的关系，举例如下：

plt.figure(figsize=(6,3))

sns.barplot('SibSp','Survived',data=data)

plt.title('SibSp vs Survived')

## #sns.violinplot分析目标特征与其他特征的关系，举例如下：

### #纯手工一对二离散变量关系

f,ax=plt.subplots(1,2,figsize=(18,8))

sns.violinplot("Pclass","Age", hue="Survived", data=data,split=True,ax=ax[0])

ax[0].set\_title('Pclass and Age vs Survived')

ax[0].set\_yticks(range(0,110,10))

sns.violinplot("Sex","Age", hue="Survived", data=data,split=True,ax=ax[1])

ax[1].set\_title('Sex and Age vs Survived')

ax[1].set\_yticks(range(0,110,10))

plt.show()

计算机生成了可选文字:
Pclass 

### #一对多离散变量关系

categorical\_features = ['model',

'brand',

'bodyType',

'fuelType',

'gearbox',

'notRepairedDamage']

for c in categorical\_features:#离散变量转化为category类型变量，[原因请点击这里](https://blog.csdn.net/cFarmerReally/article/details/105648222)

Train\_data[c] = Train\_data[c].astype('category')

if Train\_data[c].isnull().any():

Train\_data[c] = Train\_data[c].cat.add\_categories(['MISSING'])

Train\_data[c] = Train\_data[c].fillna('MISSING')

catg\_list = categorical\_features

target = 'price'

for catg in catg\_list :

sns.violinplot(x=catg, y=target, data=Train\_data)

plt.show()

计算机生成了可选文字:
100000 
80000 
40000 
100000 
80000 
40000 
20 
20 
30 
4.0 so 
body Type 
30 
fuelType 
60 
so 
7.0 
60 
MISSING 
MISSING 

## #sns.boxplot分析目标特征与其他特征的关系，举例如下：

### #分析装修质量和房子价格的关系（一对一离散变量关系）

### var = 'OverallQual'

data = pd.concat([df\_train['SalePrice'], df\_train[var]], axis=1)

f, ax = plt.subplots(figsize=(8, 6))

fig = sns.boxplot(x=var, y="SalePrice", data=data)

fig.axis(ymin=0, ymax=800000);

计算机生成了可选文字:
800000 
700000 
600000 
500000 
400000 
300000 
200000 
100000 
1 
2 
3 
4 
5 
6 
7 
8 
9 
10 

### #一对多离散变量关系

# 因为 name和 regionCode的类别太稀疏了，这里我们把不稀疏的几类画一下

def boxplot(x, y, \*\*kwargs):

sns.boxplot(x=x, y=y)

x=plt.xticks(rotation=90)

f = pd.melt(Train\_data, id\_vars=['price'], value\_vars=categorical\_features)

g = sns.FacetGrid(f, col="variable", col\_wrap=2, sharex=False, sharey=False, size=5)

g = g.map(boxplot, "value", "price")

计算机生成了可选文字:
variable = body Type 
variable = fuelType 
100000 
80000 
40000 
100000 
80000 
40000 
variable = gearbox 
100000 
80000 
40000 
100000 
80000 
40000 
variable = notRepairedDamage 

# 多变量分析（连续变量）

## [#sns.pairplot分析目标特征与其他特征的关系，举例如下：](https://www.jianshu.com/p/c50cb4f1029f)

参数介绍：

kind：用于控制非对角线上的图的类型，可选"scatter"与"reg"

diag\_kind：控制对角线上的图的类型，可选"hist"与"kde"

代码示例：

sns.set()

cols = ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', 'FullBath', 'YearBuilt']

sns.pairplot(df\_train[cols],size = 2 ,kind ='scatter',diag\_kind='kde')

plt.show()



## #sns.regplot分析目标特征与其他特征的关系，举例如下：

# 查看GRE Score, TOEFL Score, CGPA两两之间的相关性

fig, axes = plt.subplots(3, 1, figsize=(10, 15))

sns.regplot(x='GRE Score', y='TOEFL Score', data=df, ax=axes[0])

sns.regplot(x='GRE Score', y='CGPA', data=df, ax=axes[1])

sns.regplot(x='TOEFL Score', y='CGPA', data=df, ax=axes[2])

计算机生成了可选文字:
Ore 

## [#sns.heatmap](https://blog.csdn.net/m0_38103546/article/details/79935671)

#### *普通热力图*

#### [*三角热力图*](https://www.jianshu.com/p/7c5693a646ce)

# 找出相关程度

plt.figure(figsize=(20, 16)) # 指定绘图对象宽度和高度

colnm = data\_train1.columns.tolist() # 列表头

mcorr = data\_train1[colnm].corr(method="spearman") # 相关系数矩阵，即给出了任意两个变量之间的相关系数（[pandas相关系数-DataFrame.corr()参数详解](https://blog.csdn.net/walking_visitor/article/details/85128461)）

mask = np.zeros\_like(mcorr, dtype=np.bool) #[np.zeros\_like](https://www.jianshu.com/p/b78991793538) 构造与mcorr同维数矩阵 为bool型

mask[np.triu\_indices\_from(mask)] = True #[np.triu\_indices\_from(mask)](https://blog.csdn.net/yeizisn/article/details/53037328) 返回（n，m）数组**上三角**的索引。角分线右侧为True

cmap = sns.diverging\_palette(220, 10, as\_cmap=True) # 返回matplotlib colormap对象

g = sns.heatmap(mcorr, mask=mask, cmap=cmap, square=True, annot=True, fmt='0.2f') # 热力图（看两两相似度）

plt.show()

0.05 006 0.18 
• • '-ο.16 014006016007006002 
'-423 -006 0.04 010012 017 025 0.19 .3000600301 
.34η.29ηΙ3 ι 
• 007002 
006 022 029 002 004-0.12.ο. 
V25 -0.05 001 017 
008018 Ι 053 0.02 0.15 052 025 013019 040 
V24 
v26 -014006 003 014 002 αοι οιο 020 009 
'427 • 
target 
020 034 013 οοο 001 
-033 0.24006 
003 0.03 
0.09 423 
036 004 005 015 0.24 οοο 
016 017 
-0.13 
-03 
-οο 
040 036 030 0.29 00202' 015 039 027 0.13 σοι 024001005 0.13 
003 022 014 0.12 018 004 002-0,06 008 016 009 017 016 
Ι. 045 047 007 ΟΟΟ 011 Ι 0.09 4.18 
0.12 042 0.13 
v32 -005001008007 008 013 011001 016 009 οιο οοο οοο 012013 002 
'433 -006 002 002 0.06 019 0.09 008 ΟΟΟ 004 0.05 001 013014018000005 002 006003 0.02 002 010 
"34 -003005002 0.04 016 007 036 009005000008 027 004 0.08 007 0.09 005023008 0.02 
04 η.Ι9 
'435 -004 0.03 0.11 018 0.26 019 0.13 
0.13 006 0.17 002 η.ΙΙ 008 0.03 013 0.24 
74 
V36 -021 0.19 Ι. 
008 047 007 οιο 026 018 016 002 019 005 030 006 039-0.05 003 002-0,06 
0.00 0232010 002 '030 031 002 η.15ή% 002'0.27 022018030008 006000 003 002 
-013 015 034 020009 0.06 
w -008 008 024016 006 014 
.33 
02195 
ν16 
. 0.06 029 a 013 009012 0.16 047 0.02 036 
041 -0.02 003 015 0.22 
018 0.20 034 0.12 004 006013 035 —ηλ 
003 027 038 0.06 011005 0.08 028 003 0290 013 020 028 0.11 ΟΟΟ 0.27 028-0,15 • Ι 
w 
νι 
035 0.37 011 οοο 
037 
054 018 001 
νιο 
'112 'ν'13 ν14 
016 -0.17 
• 042 003 004 
0.05 , 
'Λ8 'ν'19 W 'ΩΟ 'ΩΙ 'Ω3 'Ω4 '125 'Ω6 
0.05 040 0.17 
0.20 α 
012 
'129 ν3 
ν31 
002 008 004 0.26 
003 009 003 029 
ν32 ν33 ν34 '135 
ν36 
0.45 036 0.28 
05 
026 016 
ν4 
W target 

#### *去除与目标变量相关性低的特征*

# Threshold for removing correlated variables

threshold = 0.1

# Absolute value correlation matrix

corr\_matrix = data\_train1.corr().abs()

drop\_col=corr\_matrix[corr\_matrix["target"]<threshold].index

data\_all.drop(drop\_col,axis=1,inplace=True)

## #sns.distplot（绘制某分类里，连续变量的密度分布直方图和曲线图）

f,ax=plt.subplots(1,3,figsize=(20,8))

sns.distplot(data[data['Pclass']==1].Fare,ax=ax[0])

ax[0].set\_title('Fares in Pclass 1')

sns.distplot(data[data['Pclass']==2].Fare,ax=ax[1])

ax[1].set\_title('Fares in Pclass 2')

sns.distplot(data[data['Pclass']==3].Fare,ax=ax[2])

ax[2].set\_title('Fares in Pclass 3')

plt.show()

计算机生成了可选文字:
Fares 
Fares in pclass 2 
Fares in pclass 1 
0014 — 
1012 
molo 

# 时间变量的转换

#开始时间减去结束时间，并用天数表示时间段

data['used\_time'] = (pd.to\_datetime(data['creatDate'], format='%Y%m%d', errors='coerce') -

pd.to\_datetime(data['regDate'], format='%Y%m%d', errors='coerce')).dt.days

# 但是这里不建议删除，因为删除缺失数据占总样本量过大，7.5%

# 我们可以先放着，因为如果我们 XGBoost 之类的决策树，其本身就能处理缺失值，所以可以不用管；

data['used\_time'].isnull().sum()

# [可视化报告](https://blog.csdn.net/dujiahei/article/details/100173837)

from pandas\_profiling import ProfileReport

pfr =ProfileReport(Train\_data)

pfr.to\_file("./example.html")

# 数据分组分列抽取与处理

#使用df.columns.str.extract和正则表达式分割处理某列字符串

计算机生成了可选文字:
In 
dtype: object 
[25] : 
df [ ' Latitude' ] . sty. extract ( ' ( Latitude' ] . sty. extract ( ' ) 
D: py:l: FutureWarning: curr 
ently extract (expand=None) means expand=Fa1se (return Index/Series/DataFrame) but in a 
future version of pandas this will be changed to expand—True (return DataFrame) 
*Entry point for launching an IPython kernel. 
Out [25] : 
abde 
1 
2 
abab 
3 
ab e 
4 
5 
abae 
Name: Latitude, 

tz\_counts = frame['tz'].value\_counts()统计相同value的个数

agg\_counts = by\_tz\_os.size()计算每个组的大小

#清洗某列的缺失值

clean\_tz = frame.tz.fillna(‘Missing’)

#清洗某列的空字符

clean\_tz[clean\_tz == ''] = 'Unknown'

#高级分组平均数填充缺失值

data['Initial']=data.Name.str.extract('([A-Za-z]+)\.')

#Checking the Initials with the Sex

pd.crosstab(data.Initial,data.Sex).T.style.background\_gradient(cmap='summer\_r')

计算机生成了可选文字:
Initial 
female 
male 
capt 
o 
Col 
o 
Countess 
Don 
Dr 
Jonkheer 
Lady 
Major 
Master 
Miss 
182 
Mlle 
2 
Mme 
Mr 
Mrs 
o 
Rev 
O 
Sir 
O 
o 

data['Initial'].replace(['Mlle','Mme','Ms','Dr','Major','Lady','Countess','Jonkheer','Col','Rev','Capt','Sir','Don'],['Miss','Miss','Miss','Mr','Mr','Mrs','Mrs','Other','Other','Other','Mr','Mr','Mr'],inplace=True)

data.groupby('Initial')['Age'].mean() #lets check the average age by Initials

#条件筛选dataframe

boys = top1000[top1000.sex == 'M']

fec\_mrbo = fec[fec.cand\_nm.isin(['Obama, Barack','Romney, Mitt'])] #筛选出某列包含字符的那一行

frame[frame['c'].notnull() [& frame.c.notnull()]] #筛选掉c那一框里是na值的那一行

active\_title= ratings\_by\_title[ratings\_by\_title >= 250]

subset = total\_births.loc[['John', 'Harry', 'Mary', 'Marilyn']]#根据行名取行

subset\_lesl=names[names.name.str.lower().str.contains('lesl')]

data.loc[(data.Age.isnull())&(data.Initial=='Mr'),'Age']=33

 #根据列名取列

subset = total\_births[['John', 'Harry', 'Mary', 'Marilyn']]

#二级列名取列

subtable = table.reindex(columns=[1910, 1960, 2010], level='year')

#对某行应用函数，并根据结果大小进行排列

indexer=ostz.sum(1).argsort() #默认由小到大排列

counts\_subset= ostz.take(indexer[-10:])

#取出感兴趣的数据组成新的dataframe

data.pivot\_table直接从数据表中提取主要数据变成表格，举例如下：

mean\_ratings= data.pivot\_table('rating',index='title',columns='gender',aggfunc='mean')

# dataframe的修改与增添

#np.where(给dataframe添加布尔类型的列）

cframe['os'] = np.where(cframe['a'].str.contains('Windows'),'Windows', 'Not Windows')

创建一个框，名字叫os，np.where(condiction,value,orvalue)

#concat（物理合并多个表格）

sum\_table=pd.concat([df1,df2], axis = 1)#行合并两个表axis = 1，列合并最好设置ignore\_index=True

#merge（融合多个特征）

df\_temp = pd.merge(df\_temp,df\_temp2, on = ('Topic'), how = 'left')

df\_temp.head(5)

#map单列运算

df['col2'] = df['col1'].map(lambda x: x\*\*2 [对应关系])可以是函数名，或者是字典

df['type'] = df['type'].map({'围网': 0, '刺网': 1, '拖网': 2, 'unknown': -1})

#apply多列运算，分组运算，改变行个数

df['col3'] = df.apply(lambda x: x['col1'] + 2 \* x['col2'], axis=1)

#transform分组运算，增加列个数

df['col3'] = df.groupby('col1')['col2'].transform(lambda x: (x.sum() - x) / x.count())

#applymap对整个dataframe进行运算

df1.applymap(lambda x : 1 if x>0 else 0)

#agg增加分组后某特征的统计量

group\_df = df.groupby('ship')['type'].([['label', 'mean'], ['cnt','count']]).reset\_index()

group\_df

计算机生成了可选文字:
cnt 
o 
2 
3 
4 
8995 
8996 
8997 
8998 
8999 
ship 
o 
2 
3 
4 
8995 
8996 
8997 
8998 
8999 
label 
NaN 
NaN 
NaN 
NaN 
NaN 
NaN 
NaN 
NaN 
NaN 
NaN 
9000 rows x 3 columns 

Tips:

空值：在pandas中，空值就是空字符串 “”

缺失值：np.nan（缺失数值），pd.naT（缺失时间），或None（缺失字符串）

fillna函数inplace=True替换源数据

Tips：

Annotate解释

首先要注意的是，只有数值特征进行比较

正相关：如果特征A的增加导致特征b的增加，那么它们呈正相关。值1表示完全正相关。

负相关：如果特征A的增加导致特征b的减少，则呈负相关。值-1表示完全负相关。

现在让我们说两个特性是高度或完全相关的，所以一个增加导致另一个增加。这意味着两个特征都包含高度相似的信息，并且信息很少或没有变化。这样的特征对我们来说是没有价值的！

那么你认为我们应该同时使用它们吗？。在制作或训练模型时，我们应该尽量减少冗余特性，因为它减少了训练时间和许多优点。

现在，从上面的图，我们可以看到，特征不显著相关。

#花里胡哨的热力图属性设置

plt.figure(figsize=(6,3))

sns.heatmap(data.[cols].corr(), cbar=True, annot=True, square=True, fmt='.2f', annot\_kws={'size': 10},cmap='YlGnBu') #说明的字体大小

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

plt.tight\_layout()

##### [#data.corr().nlargest筛选与tabel相关最高的特征](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.nlargest.html)

k = 10 #number of variables for heatmap

cols = data.corr().nlargest(k, 'SalePrice')['SalePrice'].index

#方法一

cm = [np.corrcoef](https://blog.csdn.net/qq_39514033/article/details/88931639)(data[cols].values.T)

sns.set(font\_scale=1.25)

sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot\_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values,cmap='YlGnBu')

#方法二

sns.set(font\_scale=1.25)

sns.heatmap(data.[cols].corr(), cbar=True, annot=True, square=True, fmt='.2f', annot\_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values,cmap='YlGnBu')

特征工程与特征选择

2020年5月29日

15:25

## 构造时间特征

#### *例一*

# 使用时间：data['creatDate'] - data['regDate']，反应汽车使用时间，一般来说价格与使用时间成反比

# 不过要注意，数据里有时间出错的格式，所以我们需要 errors='coerce'

data['used\_time1'] = (pd.to\_datetime(data['creatDate'], format='%Y%m%d', errors='coerce') -

pd.to\_datetime(data['regDate'], format='%Y%m%d', errors='coerce')).dt.days

#### *例二 时刻构造分组时间特征（Top1比赛方案）*

#### *时间格式转换*

df['time'] = df['time'].apply(lambda x: '2019-' + x.split(' ')[0][:2] + '-' + x.split(' ')[0][2:] + ' ' + x.split(' ')[1])#切片构造标准时间样式



转化为



df['time'] = pd.to\_datetime(df['time']) #转化为时间格式

df['diff\_minutes'] = df.groupby('ship')['time'].diff(-1).dt.seconds // 60 #用diff得到时间间隔，转为秒，又转为分，是需要四舍五入

#### *根据日期构造，年月日星期的特征*

def date\_tran(df,fea\_col):

for f in tqdm(fea\_col):

df[f] = pd.to\_datetime(df[f].astype('str').apply(date\_proc))

df[f + '\_year'] = df[f].dt.year

df[f + '\_month'] = df[f].dt.month

df[f + '\_day'] = df[f].dt.day

df[f + '\_dayofweek'] = df[f].dt.dayofweek

return (df)

###### *#例三*

#转化成时间格式 issueDateDT特征表示数据日期离数据集中日期最早的日期（2007-06-01）的天数

data\_all['issueDate'] = pd.to\_datetime(data\_all['issueDate'],format='%Y-%m-%d')

startdate = datetime.datetime.strptime('2007-06-01', '%Y-%m-%d')

data\_all['issueDateDT'] = data\_all['issueDate'].apply(lambda x: x-startdate).dt.days

## #特征简单合并与筛选

### #按经验去除去除无用或重复的特征

data.drop(['Name','Age','Ticket','Fare','Cabin','Fare\_Range','PassengerId'],axis=1,inplace=True)

### #按经验筛选有用的特征

data=data[[columns]]

### #通过heatmap判断，合并相关性较高的特征，或者合并属性差不多的特征。举例如下：

观察他们之间的相关性，相关性较高的可以生成新的特征

df['GREAndToefl'] = df['GRE Score'] + df['TOEFL Score'] + df['CGPA']

df['test'] = df['SOP'] + df['LOR']

df['compatable'] = df['University Rating']+ df['Research']

## #特征转换

### #条件转换

df['anchor'] = df.apply(lambda x: 1 if x['y\_diff'] < 0.01 and x['x\_diff'] < 0.01

and x['v'] < 0.1 and x['diff\_minutes'] <= 10 else 0, axis=1)

all\_features['haspool'] = all\_features['PoolArea'].apply(lambda x: 1 if x > 0 else 0)

### #对数转换

def logs(res, ls):

for l in ls:

res[l+'\_log' ] = res[l].apply(lambda x:[np.log](https://blog.csdn.net/YZXnuaa/article/details/79520323)(1.01+x))

return res

log\_features = ['PoolArea','MiscVal','YearRemodAdd','TotalSF']

all\_features = logs(all\_features, log\_features)

### #平方转换

def logs(res, ls):

for l in ls:

res[l+'\_sque' ] = res[l].apply(lambda x:np.square(x))

return res

log\_features = ['PoolArea','MiscVal','YearRemodAdd','TotalSF']

all\_features = logs(all\_features, log\_features)

### #分组增添特征的统计量

#### *#方法一*

# 计算某品牌的销售统计量，同学们还可以计算其他

# 这里要以 train 的数据计算统计量

Train\_gb = Train\_data.groupby("brand")

all\_info = {}

for kind, kind\_data in Train\_gb:

info = {}

kind\_data = kind\_data[kind\_data['price'] > 0]

info['brand\_amount'] = len(kind\_data)

info['brand\_price\_max'] = kind\_data.price.max()

info['brand\_price\_median'] = kind\_data.price.median()

info['brand\_price\_min'] = kind\_data.price.min()

info['brand\_price\_sum'] = kind\_data.price.sum()

info['brand\_price\_std'] = kind\_data.price.std()

info['brand\_price\_average'] = round(kind\_data.price.sum() / (len(kind\_data) + 1), 2)

all\_info[kind] = info

brand\_fe = pd.DataFrame(all\_info).T.reset\_index().rename(columns={"index": "brand"})

data = data.merge(brand\_fe, how='left', on='brand')

#### *#方法二（墙裂推荐）（Top1比赛方案）*

def q10(x):

    return x.quantile(0.1)

def q20(x):

    return x.quantile(0.2)

stat\_functions = ['min', 'max', 'mean', 'median', 'nunique', q10, q20, q30, q40, q60, q70, q80, q90]

stat\_ways = ['min', 'max', 'mean', 'median', 'nunique', 'q\_10', 'q\_20', 'q\_30', 'q\_40', 'q\_60', 'q\_70', 'q\_80', 'q\_90']

stat\_cols = ['y', 'x', 'v', 'd']

group\_tmp = df.groupby('ship')[stat\_cols].agg(stat\_functions).reset\_index()

o 
ship 
o 
min 
5, 124873e•06 
5.042857+06 
5.130781e+06 
5.112874+06 
5.130494e*06 
5094050e+06 
median 
5.130672e•06 
5.112760e*06 
nunique 
23 
138 
min 
6.118352e+06 
6049472+06 
median 
6118352e•06 
min 
6, 152038+06 
6.102450+06 
6.119351e+06 
6091460+06 
max 
39 
1047 

group\_tmp.columns = ['ship'] + ['{}\_{}'.format(i, j) for i in stat\_cols for j in stat\_ways]

o 
ship 
o 
y_min 
5.124873e+06 
5.042857e+06 
Y—max 
5.130781€+06 
5.112874é+06 
y—mean 
5.130494e+06 
5.094050e+06 
y—median 
5.130672e+06 
5112760é+06 
y—nunique 
23 
138 
x-min 
6.1183529+06 
6.049472e+06 
x—max 
6_102450e+06 
X—mean 
6.1193519+06 
6.091460e+06 
x_median 
6.118352e+06 
6.102450e+06 
v_min 
0.0 
0.0 

#同理，排除df中位移为零的数据再进行统计特征刻画；排除df中速度为零的数据再进行统计特征刻画

y\_x\_neq\_zero = df[(df['y\_diff'] != 0) & (df['x\_diff'] != 0)]

y\_x\_neq\_group = y\_x\_neq\_zero.groupby('ship')[stat\_cols].agg(stat\_functions).reset\_index()#对['y', 'x', 'v', 'd']进行进行统计特征刻画

y\_x\_neq\_group.columns = ['ship'] + ['pos\_neq\_zero\_{}\_{}'.format(i, j) for i in stat\_cols for j in stat\_ways]#改变columns名称

v\_neg\_zero = df[df['v\_diff'] != 0]

v\_neg\_zero\_group = v\_neg\_zero.groupby('ship')[stat\_cols].agg(stat\_functions).reset\_index()

v\_neg\_zero\_group.columns = ['ship'] + ['v\_neq\_zero\_{}\_{}'.format(i, j) for i in stat\_cols for j in stat\_ways]

#融合数据

group\_df = group\_df.merge(group\_tmp, on='ship', how='left')

group\_df = group\_df.merge(y\_x\_neq\_group, on='ship', how='left')

group\_df = group\_df.merge(v\_neg\_zero\_group, on='ship', how='left')

#以'ship', 'y', 'x'分组，做count统计量的特征刻画，筛选前三频次位置信息，并添加这些位置信息到数据中

mode\_df = df.groupby(['ship', 'y', 'x'])['time'].agg({'mode\_cnt': 'count'}).reset\_index()

mode\_df['rank'] = mode\_df.groupby('ship')['mode\_cnt'].rank(method='first', ascending=False)

    for i in range(1, 4):

    tmp\_df = mode\_df[mode\_df['rank'] == i]

    del tmp\_df['rank']

    tmp\_df.columns = ['ship', 'rank{}\_mode\_y'.format(i), 'rank{}\_mode\_x'.format(i), 'rank{}\_mode\_cnt'.format(i)]

    group\_df = group\_df.merge(tmp\_df, on='ship', how='left')

## #定义交叉特征统计（类别交叉连续）

用数值特征对类别特征做统计刻画

连续变量统计量：'min'， 'median'，'max'

def cross\_cat\_num(df,num\_col,cat\_col):

for f1 in tqdm(cat\_col):

g = df.groupby(f1, as\_index=False)

for f2 in tqdm(num\_col):

feat = g[f2].agg({

'{}\_{}\_max'.format(f1, f2): 'max', '{}\_{}\_min'.format(f1, f2): 'min',

'{}\_{}\_median'.format(f1, f2): 'median',

})

df = df.merge(feat, on=f1, how='left')

return(df)

## # 类别特征的二阶交叉（类别交叉类别）

离散变量统计量： 'nunique'，'count' ，比例偏好， entropy

from scipy.stats import entropy

def cross\_qua\_cat\_num(df):

for f\_pair in tqdm([

['model', 'brand'], ['model', 'regionCode'], ['brand', 'regionCode']

]):

### 共现次数

df['\_'.join(f\_pair) + '\_count'] = df.groupby(f\_pair)['SaleID'].transform('count')

### n unique、熵

df = df.merge(df.groupby(f\_pair[0], as\_index=False)[f\_pair[1]].agg({

'{}\_{}\_nunique'.format(f\_pair[0], f\_pair[1]): 'nunique',

'{}\_{}\_ent'.format(f\_pair[0], f\_pair[1]): lambda x: entropy(x.value\_counts() / x.shape[0])

}), on=f\_pair[0], how='left')

df = df.merge(df.groupby(f\_pair[1], as\_index=False)[f\_pair[0]].agg({

'{}\_{}\_nunique'.format(f\_pair[1], f\_pair[0]): 'nunique',

'{}\_{}\_ent'.format(f\_pair[1], f\_pair[0]): lambda x: entropy(x.value\_counts() / x.shape[0])

}), on=f\_pair[1], how='left')

### 比例偏好

df['{}\_in\_{}\_prop'.format(f\_pair[0], f\_pair[1])] = df['\_'.join(f\_pair) + '\_count'] / df[f\_pair[1] + '\_count']

df['{}\_in\_{}\_prop'.format(f\_pair[1], f\_pair[0])] = df['\_'.join(f\_pair) + '\_count'] / df[f\_pair[0] + '\_count']

return (df)

## #四种自动化特征筛选 （训练集）

### 1、[单变量分析（SelectPercentile）](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectPercentile.html" \l "sklearn.feature_selection.SelectPercentile)

from sklearn.datasets import load\_digits  
from sklearn.feature\_selection import SelectPercentile, chi2,f\_classif, f\_regression  
X, y = load\_digits(return\_X\_y=True)  
X.shape

(1797, 64)

X\_new = SelectPercentile(chi2, percentile=10).fit\_transform(X, y)  
X\_new.shape

(1797, 7)

拓展：score\_func的参数选取

f\_regression:基于线性回归分析来计算统计指标。适用于回归问题

chi2:计算卡方统计量，适用于分类问题

f\_classif（default）:根据方差分析（ANOVA）的原理，以F-分布为依据，利用平方和与自由度所计算的祖居与组内均方估计出F值，适用于分类问题

### [2、基于模型的特征选择（SelectFromModel）](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html#sklearn.feature_selection.SelectFromModel)

#假如某个特征和响应变量之间的关系是非线性的，可以用基于树的方法（决策树、随机森林）、或者扩展的线性模型等

#回归模型，SVM，决策树，随机森林等等。

#根据某种模型估计a coef\_ attribute or a feature\_importances\_ attribute,然后删除特征重要性小于给定阈值的特征

from sklearn.ensemble import RandomForestClassifier

select = SelectFromModel(RandomForestClassifier(n\_estimators=100, random\_state=42), threshold="median")

#threshold参数设置根据特征重要性筛选特征的阈值“median”，“1.25\*mean”，Lasso

select=select.fit(X\_train, y\_train)

X\_train\_l1 = select.transform(X\_train)

print(X\_train.shape)

print(X\_train\_l1.shape)

#Output:

#(284, 80)

#(284, 40)

#一行代码搞定

X\_new =SelectFromModel(RandomForestClassifier(n\_estimators=100, random\_state=42), threshold="median").fit\_transform(X, y)

拓展：max\_feature参数选择

The maximum number of features to select. To only select based on max\_features, set threshold=-np.inf.

### [3、逐步特征删除（RFE）](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html#sklearn.feature_selection.RFE)

from sklearn.feature\_selection import RFE

#根据a coef\_ attribute or a feature\_importances\_ attribute逐渐删除重要性最低的特征子集,直到特征达到指定需要个数为止

select = RFE(RandomForestClassifier(n\_estimators=100, random\_state=42), n\_features\_to\_select=40).fit(X\_train, y\_train)

#select = RFE(LogisticRegression(penalty="l1"), n\_features\_to\_select=40).fit(X\_train, y\_train)

#n\_features\_to\_select参数设置要选择的特征数目,默认选一半

# visualize the selected features:

mask = select..support\_

plt.matshow(mask.reshape(1, -1), cmap='gray\_r')

<zatplotlib. image. Axeslzage at Ox121b95SO> 
So 
70 

### [4、序列化特征选择（SequentialFeatureSelector）](http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/)

from mlxtend.feature\_selection import SequentialFeatureSelector

#http://rasbt.github.io/mlxtend/

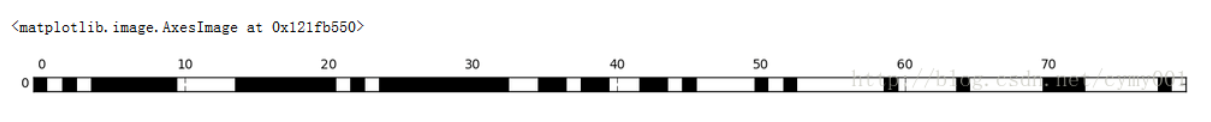
sfs = SequentialFeatureSelector(LogisticRegression(), k\_features=40, forward=True, scoring='accuracy',cv=5).fit(X\_train, y\_train)

# visualize the selected features:

mask = np.zeros(X\_train.shape[1], dtype='bool')

mask[np.array(sfs.k\_feature\_idx\_)] = True

plt.matshow(mask.reshape(1, -1), cmap='gray\_r')



## 深度学习的特征挖掘

* 1. LR
  2. Poly2
  3. FM
  4. FFM
  5. MLP
  6. WDL(DLRS16)
  7. DeepFM(IJCAI17)
  8. NFM(IJCAI17)
  9. AFM(IJCAI17)
  10. xDeepFM(KDD19)
  11. TFNET(SIGIR20)
  12. ONN/NFFM(Arxiv19)
  13. AoAFFM(AAAI20)
  14. AutoFIS(KDD20)

### [CTR深度交叉特征](https://zhuanlan.zhihu.com/p/257895631?utm_source=ZHShareTargetIDMore)

### [FM(因子分解机），DeepFM，XDeepFM](https://blog.csdn.net/qq_33761777/article/details/109961673)

xDeepFM更像是Deep&Cross的改进版，结合了其中显式特征组合的思想，同时克服了它的两个缺点。

### [利用深度学习网络显示挖掘高阶特征交互的CTR模型](https://blog.csdn.net/guoyuhaoaaa/article/details/108882578)

### [自动特征组合——微软deep crossing模型](https://blog.csdn.net/dpengwang/article/details/106074511)

# 数据导出

# 目前的数据其实已经可以给树模型使用了，所以我们导出一下

data.to\_csv('data\_for\_tree.csv', index=0)

特征选择

2020年10月19日

10:25

# 资料

[（3.1）特征选择---过滤法（特征相关性分析）](https://blog.csdn.net/dili8870/article/details/101506788)

**【1】**[特征选择： 卡方检验、F 检验和互信息](https://www.cnblogs.com/massquantity/p/10486904.html)

【2】[特征工程总结（三）特征相关性分析](https://blog.csdn.net/qq_36874480/article/details/80272047)

【3】[P值解释和误区](https://www.cnblogs.com/webRobot/p/6772744.html)

【4】[机器学习特征选择之卡方检验与互信息](https://blog.csdn.net/yihucha166/article/details/50646615)

【5】[Maximal Information Coefficient (MIC)最大互信息系数详解与实现](https://zhuanlan.zhihu.com/p/53092905)

【6】[结合Scikit-learn介绍几种常用的特征选择方法](https://blog.csdn.net/bryan__/article/details/51607215)

【7】[Sklearn中的f\_classif和f\_regression](https://blog.csdn.net/jetFlow/article/details/78884619)

【8】[特征选择：方差选择法、卡方检验、互信息法、递归特征消除、L1范数、树模型](https://www.cnblogs.com/wzdLY/p/9689547.html)

【9】[结合Scikit-learn介绍几种常用的特征选择方法](https://www.cnblogs.com/hhh5460/p/5186226.html)（优秀）

【10】[Relief 特征选择算法简单介绍](https://blog.csdn.net/coffee_cream/article/details/61420732)

【11】[Relief特征选择算法Python实现](https://andy1314chen.github.io/2017/08/22/ReliefF%E7%89%B9%E5%BE%81%E9%80%89%E6%8B%A9%28python%29/)

【12】[浅谈关于特征选择算法与Relief的实现](https://cloud.tencent.com/developer/article/1087796)>

[（3.2）特征选择---包装法](https://blog.csdn.net/dili8870/article/details/101506793)

[（3.3）特征选择---嵌入法（特征重要性评估](https://blog.csdn.net/dili8870/article/details/101506791?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.add_param_isCf&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.add_param_isCf)

【1】[树模型特征重要性评估方法](https://blog.csdn.net/tinkle181129/article/details/80231871)

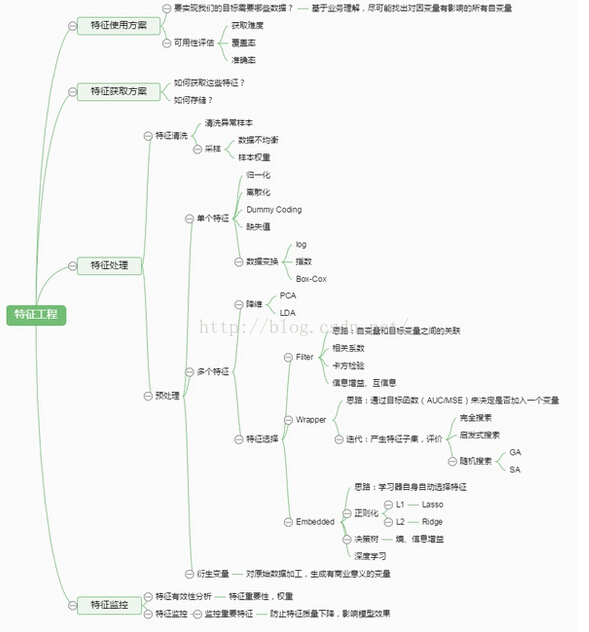
【2】[用xgboost模型对特征重要性进行排序](https://blog.csdn.net/waitingzby/article/details/81610495)

【3】[xgboost特征重要性源代码](https://github.com/zengfanxi/xgboost/blob/master/python-package/xgboost/core.py)

【4】[机器学习的特征重要性究竟是怎么算的](https://zhuanlan.zhihu.com/p/64759172)（知乎）

【5】[特征工程](https://daiwk.github.io/posts/ml-feature-engineering.html)

[机器学习之（四）特征工程以及特征选择的工程方法](https://blog.csdn.net/boon_228/article/details/51749646?utm_medium=distribute.pc_relevant.none-task-blog-title-4&spm=1001.2101.3001.4242)



·

# 自动特征工程

skopt 
hyperopt 
ATM 
Oxnet 
PYTbRCH 
H20 
AutoML 
FAR.HO 
auto-sklearn 

hyperparameter

[HyperparameterHunter 3.0：一文教你学会自动化特征工程](https://mp.weixin.qq.com/s/hNh1PXKZqeSZ9gDt1uZKwA)

[Github地址](https://github.com/HunterMcGushion/hyperparameter_hunter)

[HyperparameterHunter官方文档](https://hyperparameter-hunter.readthedocs.io/en/latest/why_use_hyperparameter_hunter.html)

featuretools

[AutoML之自动化特征工程](https://www.cnblogs.com/wkang/p/10380500.html)

[手把手教你用Python实现自动特征工程Python特征工程库Featuretools](https://zhuanlan.zhihu.com/p/43630912)

[Feature Tools：可自动构造机器学习特征的Python库](https://blog.csdn.net/hellozhxy/article/details/80772872)

[浅谈自动特征构造工具Featuretools](https://www.cnblogs.com/dogecheng/p/12659605.html)

[机器学习特征工程3-自动特征构造](https://www.jianshu.com/p/71782dbe2e1e)

[项目地址](https://featuretools.alteryx.com/en/stable/)

[代码地址](https://github.com/WillKoehrsen/automated-feature-engineering/blob/master/walk_through/Automated_Feature_Engineering.ipynb)

【前言】众多特征工程中实际上只是另一种需要优化的超参数而已——但是并没有工具去优化这些「超参数』

loans. sample (10) 
client_id loan_type 
254 
196 
45 
415 
26 
417 
137 
200 
180 
8 
29841 
26326 
41480 
49624 
49545 
49624 
32726 
26326 
48177 
46109 
credit 
credit 
credit 
home 
loan—mount 
4718 
10902 
12581 
8621 
4131 
11219 
7499 
5275 
6521 
11062 
repaid 
o 
o 
o 
o 
o 
loan_id 
11175 
11559 
11461 
10454 
10939 
11132 
10285 
11072 
11235 
11611 
loan_start 
2014-03-08 
2014 _09_06 
2009-10-29 
20140-26 
2008-10-13 
201m&12 
2012.W07 
2014-11-11 
20140-12 
201209-12 
loan_end 
2015-09-13 
201611-14 
2011-07-08 
2015-12-29 
2010-02-26 
201&03-15 
201503-12 
201007-17 
201&02-22 
2014-03-14 
rate 
2.74 
6.73 
1.94 
3.18 
4.07 
3.77 
2.72 
1.45 
6.17 
5.48 

payments. sample (10) 
loan_id payment_amount 
payment_date 
2009-07-14 
2009-11 o6 
2012-03-18 
2013-10-04 
2005-1103 
20070-20 
2014-0T23 
2001-03-20 
200Tw02 
missed 
965 
1285 
1664 
1074 
2521 
2909 
44 
1581 
1344 
11221 
11264 
10769 
10106 
10868 
10537 
11141 
10109 
10624 
10262 
1511 
218 
210 
1003 
349 
1292 
1220 
1176 
770 

1 clients. head () 
0 
2 
3 
4 
client_id 
46109 
49545 
41480 
46180 
25707 
joined 
2002-04-16 
2007-11-14 
2013-03-11 
2001-11-06 
200&10-06 
income 
172677 
104564 
122607 
43851 
211422 
credit_score 
527 
770 
562 
621 

# 基本知识

特征选择的目的是筛选出能训练出更好的模型的数据。

特征选择的方法有：

1、过滤思想：利用特征与标注的距离（可以是相关性、欧氏距离、假设检验等）剔除最不相关的特征。[sklearn.feature\_selection.SelectKBest](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html)

2、包裹思想：通过构造简单模型，根据系数、逐步过滤特征集合（比如RFE算法）。[sklearn.feature\_selection.RFE](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html)

3、嵌入思想：通过所有特征构建的模型（如线性回归）拟合当前的标注，最后根据各个特征的权重筛选掉权重小的特征。

数据降维

2020年5月30日

22:47

# [sklearn.decomposition.PCA的使用笔记](https://www.jianshu.com/p/115f8b3708c1)

from sklearn import decomposition

pca = decomposition.PCA(n\_components=146)

all\_pca = pca.fit\_transform(all\_data)

X\_pca = all\_pca[:len(x\_train)]

test = all\_pca[len(x\_train):]

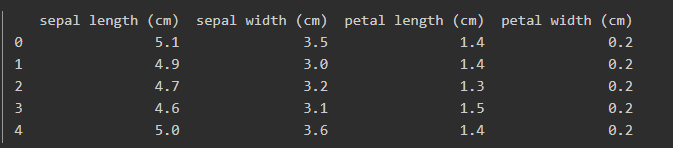
y = Train\_data['price'].values

拓展：

如果n\_components == 'mle'且svd\_solver == 'full'，将采用Minka’s MLE方法得出最终的维度

举例如下

print(df\_iris.head())



from sklearn.decomposition import PCA

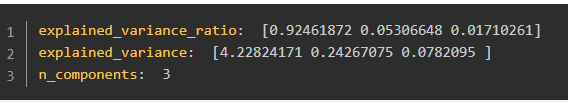
mle\_pca = PCA(n\_components='mle', svd\_solver='full')

mle\_pca.fit(df\_iris)

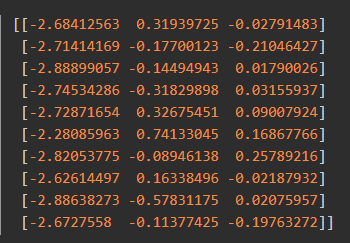
print('explained\_variance\_ratio: ', mle\_pca.explained\_variance\_ratio\_)

print('explained\_variance: ', mle\_pca.explained\_variance\_)

print('n\_components: ', mle\_pca.n\_components\_)



print(X\_pca)



建立模型

2020年5月7日

14:26

# 初始化

# 分类模型

[#(Support Vector Machines)支持向量机](https://blog.csdn.net/weixin_41990278/article/details/93137009)

①Radial Support Vector Machines(rbf-SVM)

model=svm.SVC(kernel='rbf',C=1,gamma=0.1)

model.fit(train\_X,train\_Y)

prediction1=model.predict(test\_X)

print('Accuracy for rbf SVM is ',metrics.accuracy\_score(prediction1,test\_Y))



②Linear Support Vector Machine(linear-SVM)

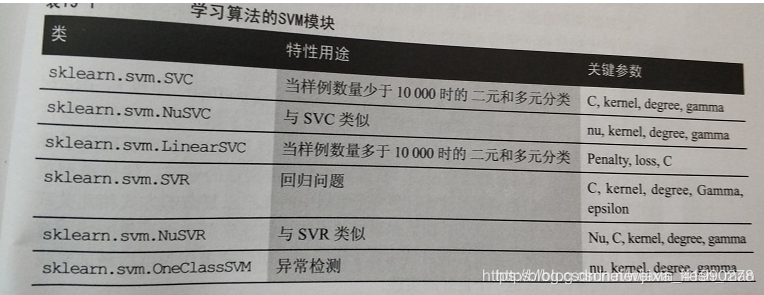
model=svm.SVC(kernel='linear',C=0.1,gamma=0.1)

model.fit(train\_X,train\_Y)

prediction2=model.predict(test\_X)

print('Accuracy for linear SVM is',metrics.accuracy\_score(prediction2,test\_Y))





tips：

#rbf: Radial Basis Function

径向基函数

#参数详解

C：C-SVC的惩罚参数C?默认值是1.0  
C越大，相当于惩罚松弛变量，希望松弛变量接近0，即对误分类的惩罚增大，趋向于对训练集全分对的情况，这样对训练集测试时准确率很高，但泛化能力弱。C值小，对误分类的惩罚减小，允许容错，将他们当成噪声点，泛化能力较强。

kernel ：核函数，默认是rbf，可以是‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’  
– 线性：u’v  
– 多项式：(gamma\*u’v + coef0)^degree  
– RBF函数：exp(-gamma|u-v|^2)  
–sigmoid：tanh(gammau’\*v + coef0)

degree ：多项式poly函数的维度，默认是3，选择其他核函数时会被忽略。

gamma ： ‘rbf’,‘poly’ 和‘sigmoid’的核函数参数。默认是’auto’，则会选择1/n\_features

coef0 ：核函数的常数项。对于‘poly’和 ‘sigmoid’有用。

#利用[GridSearchCV](https://www.cnblogs.com/mdevelopment/p/9634629.html)求最优参数

import numpy as np from sklearn.model\_selection import GridSearchCV parameters={'kernel':['linear','rbf','sigmoid','poly'],'C':np.linspace(0.1,10,20),'gamma':np.linspace(0.1,10,20)}

svc = svm.SVC()

model = GridSearchCV(svc,parameters,cv=5,[scoring](https://blog.csdn.net/u014652309/article/details/84979497)='accuracy')

model.fit(X\_train,y\_train)

model.best\_params\_

model.score(X\_test,y\_test)

# [LogisticRegression逻辑回归模型](https://blog.csdn.net/Monk_donot_know/article/details/90720366?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-3&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-3)

model = LogisticRegression() #实例化逻辑回归模型

model.fit(train\_X,train\_Y) #拟合模型

prediction3=model.predict(test\_X) #预测结果

print('The accuracy of the Logistic Regression is',metrics.accuracy\_score(prediction3,test\_Y)) #衡量并得出结果准确度



#[Decision Tree决策树](https://blog.csdn.net/qq_41577045/article/details/79844709?ops_request_misc=&request_id=&biz_id=102&utm_term=decision%20tree%E5%8F%82%E6%95%B0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduweb~default-0-79844709)

model=DecisionTreeClassifier()

model.fit(train\_X,train\_Y)

prediction4=model.predict(test\_X)

print('The accuracy of the Decision Tree is',metrics.accuracy\_score(prediction4,test\_Y))



#看看是否过拟合

print("train score:", model.score(X\_train, y\_train))

print("test score:", model.score(X\_test, y\_test))

#优化模型参数，举例优化max\_depth参数

def cv\_score(d): #创建一个函数使用不同的模型深度训练模型，并计算评分数据

clf = DecisionTreeClassifier(max\_depth=d)

clf.fit(X\_train, y\_train)

return(clf.score(X\_train, y\_train), clf.score(X\_test, y\_test))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

depths = np.arange(1,10)

scores = [cv\_score(d) for d in depths]

tr\_scores = [s[0] for s in scores]

te\_scores = [s[1] for s in scores]

# 找出交叉验证数据集评分最高的索引

tr\_best\_index = [np.argmax](https://www.jianshu.com/p/54a21e3100bb)(tr\_scores)

te\_best\_index = np.argmax(te\_scores)

print("bestdepth:", te\_best\_index+1, " bestdepth\_score:", te\_scores[te\_best\_index], '\n')

#把模型参数和对应的评分画出来

depths = np.arange(1,10)

plt.figure(figsize=(6,4), dpi=120)

plt.grid()

plt.xlabel('max depth of decison tree')

plt.ylabel('Scores')

plt.plot(depths, te\_scores, label='test\_scores')

plt.plot(depths, tr\_scores, label='train\_scores')

plt.legend()

#[K-Nearest Neighbours(KNN)](https://www.cnblogs.com/YukiNote/p/11381246.html)

model=KNeighborsClassifier()

model.fit(train\_X,train\_Y)

prediction5=model.predict(test\_X)

print('The accuracy of the KNN is',metrics.accuracy\_score(prediction5,test\_Y))



#找出n\_neighbours最优值

a\_index=list(range(1,11))

a=pd.Series()

x=[0,1,2,3,4,5,6,7,8,9,10]

for i in list(range(1,11)):

model=KNeighborsClassifier(n\_neighbors=i)

model.fit(train\_X,train\_Y)

prediction=model.predict(test\_X)

a=a.append(pd.Series(metrics.accuracy\_score(prediction,test\_Y)))

plt.plot(a\_index, a)

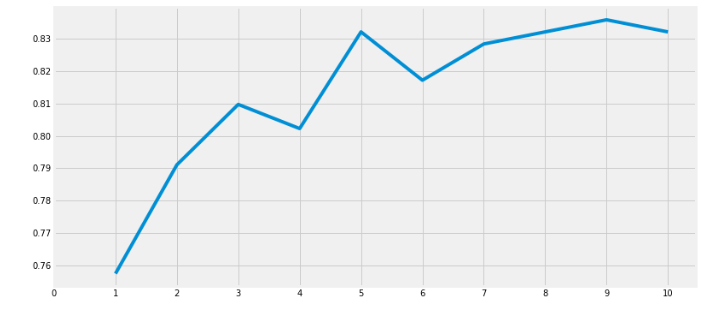
plt.xticks(x)

fig=plt.gcf()

fig.set\_size\_inches(12,6)

plt.show()

print('Accuracies for different values of n are:',a.values,'with the max value as ',a.values.max()



[#NaiveBayes（朴素贝叶斯）](https://blog.csdn.net/brucewong0516/article/details/78798359)

model=GaussianNB()

model.fit(train\_X,train\_Y)

prediction6=model.predict(test\_X)

print('The accuracy of the NaiveBayes is',metrics.accuracy\_score(prediction6,test\_Y))

[#Random Forests（随机森林）](https://www.jianshu.com/p/a493ebb90cce)

model=RandomForestClassifier(n\_estimators=100)

model.fit(train\_X,train\_Y)

prediction7=model.predict(test\_X)

print('The accuracy of the Random Forests is',metrics.accuracy\_score(prediction7,test\_Y))

# 模型和特征评估

# [集成方法](https://www.jianshu.com/p/c4bf8821af19)

# 回归模型

X = all\_features.iloc[:len(train\_labels), :]

X\_test = all\_features.iloc[len(train\_labels):, :]

X.shape, train\_labels.shape, X\_test.shape

# Setup cross validation folds

from sklearn.model\_selection import [KFold](https://blog.csdn.net/weixin_43685844/article/details/88635492)

kf = KFold(n\_splits=12, random\_state=42, shuffle=True)

# Define error metrics

def rmsle(y, y\_pred):

return np.sqrt([mean\_squared\_error](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html)(y, y\_pred))

def cv\_rmse(model, X=X):

rmse = np.sqrt(-cross\_val\_score(model, X, train\_labels, scoring="neg\_mean\_squared\_error", cv=kf))

return (rmse)

[#LGBMRegressor](https://blog.csdn.net/ssswill/article/details/85235074)

lightgbm = LGBMRegressor(objective='regression',

num\_leaves=6,

learning\_rate=0.01,

n\_estimators=7000,

max\_bin=200,

bagging\_fraction=0.8,

bagging\_freq=4,

bagging\_seed=8,

feature\_fraction=0.2,

feature\_fraction\_seed=8,

min\_sum\_hessian\_in\_leaf = 11,

verbose=-1,

random\_state=42)

# XGBoost Regressor

xgboost = XGBRegressor(learning\_rate=0.01,

n\_estimators=6000,

max\_depth=4,

min\_child\_weight=0,

gamma=0.6,

subsample=0.7,

colsample\_bytree=0.7,

objective='reg:linear',

nthread=-1,

scale\_pos\_weight=1,

seed=27,

reg\_alpha=0.00006,

random\_state=42)

## # [Ridge Regressor](https://blog.csdn.net/denghecsdn/article/details/78542583)

##### #[RidgeCV](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeCV.html)（回归）

from sklearn.linear\_model import RidgeCV

ridge\_alphas = [1e-15, 1e-10, 1e-8, 9e-4, 7e-4, 5e-4, 3e-4, 1e-4, 1e-3, 5e-2, 1e-2, 0.1, 0.3, 1, 3, 5, 10, 15, 18, 20, 30, 50, 75, 100]

ridge = [make\_pipeline](https://blog.csdn.net/elma_tww/article/details/88427695)(RobustScaler(), RidgeCV(alphas=ridge\_alphas, cv=kf))

##### [#RidgeClassifierCV](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifierCV.html#sklearn.linear_model.RidgeClassifierCV)（分类）

class sklearn.linear\_model.RidgeClassifierCV(alphas=(0.1, 1.0, 10.0), \*, fit\_intercept=True, normalize=False, scoring=None, cv=None, class\_weight=None, store\_cv\_values=False)[source]

# Support Vector Regressor

svr = make\_pipeline(RobustScaler(), SVR(C= 20, epsilon= 0.008, gamma=0.0003))

# Gradient Boosting Regressor

gbr = GradientBoostingRegressor(n\_estimators=6000,

learning\_rate=0.01,

max\_depth=4,

max\_features='sqrt',

min\_samples\_leaf=15,

min\_samples\_split=10,

loss='huber',

random\_state=42)

# Random Forest Regressor

rf = RandomForestRegressor(n\_estimators=1200,

max\_depth=15,

min\_samples\_split=5,

min\_samples\_leaf=5,

max\_features=None,

oob\_score=True,

random\_state=42)

[#StackingCVRegressor](https://blog.csdn.net/zhuiqiuuuu/article/details/84866502)

stack\_gen = StackingCVRegressor(regressors=(xgboost, lightgbm, svr, ridge, gbr, rf),

meta\_regressor=xgboost,

use\_features\_in\_secondary=True)

# [Keras构造深度学习模型](https://keras-cn.readthedocs.io/en/latest/)

# sklearn中提供了Pipeline(管道操作)

# 模型参数优化，评估

from sklearn.preprocessing import StandardScaler

def train\_model(model, param\_grid=[], X=[], y=[],

                splits=5, repeats=5):

    # get unmodified training data, unless data to use already specified

    if len(y)==0:

        X,y = get\_trainning\_data\_omitoutliers()

        #poly\_trans=PolynomialFeatures(degree=2)

        #X=poly\_trans.fit\_transform(X)

        #X=MinMaxScaler().fit\_transform(X)

    # create cross-validation method

    rkfold = RepeatedKFold(n\_splits=splits, n\_repeats=repeats)

    # perform a grid search if param\_grid given

    if len(param\_grid)>0:

        # setup grid search parameters

        gsearch = GridSearchCV(model, param\_grid, cv=rkfold,

                               scoring="neg\_mean\_squared\_error",

                               verbose=1, return\_train\_score=True)

        # search the grid

        gsearch.fit(X,y)

        # extract best model from the grid

        model = gsearch.best\_estimator\_

        best\_idx = gsearch.best\_index\_

        # get cv-scores for best model

        grid\_results = pd.DataFrame(gsearch.cv\_results\_)

        cv\_mean = abs(grid\_results.loc[best\_idx,'mean\_test\_score'])

        cv\_std = grid\_results.loc[best\_idx,'std\_test\_score']

    # no grid search, just cross-val score for given model

    else:

        grid\_results = []

        cv\_results = cross\_val\_score(model, X, y, scoring="neg\_mean\_squared\_error", cv=rkfold)

        cv\_mean = abs(np.mean(cv\_results))

        cv\_std = np.std(cv\_results)

    # combine mean and std cv-score in to a pandas series

    cv\_score = pd.Series({'mean':cv\_mean,'std':cv\_std})

    # predict y using the fitted model

    y\_pred = model.predict(X)

    # print stats on model performance

    print('----------------------')

    print(model)

    print('----------------------')

    print('score=',model.score(X,y))

    print('rmse=',rmse(y, y\_pred))

    print('mse=',mse(y, y\_pred))

    print('cross\_val: mean=',cv\_mean,', std=',cv\_std)

    # residual plots

    y\_pred = pd.Series(y\_pred,index=y.index)

    resid = y - y\_pred

    mean\_resid = resid.mean()

    std\_resid = resid.std()

    z = (resid - mean\_resid)/std\_resid

    n\_outliers = sum(abs(z)>3)

    plt.figure(figsize=(15,5))

    ax\_131 = plt.subplot(1,3,1)

    plt.plot(y,y\_pred,'.')

    plt.xlabel('y')

    plt.ylabel('y\_pred');

    plt.title('corr = {:.3f}'.format(np.corrcoef(y,y\_pred)[0][1]))

    ax\_132=plt.subplot(1,3,2)

    plt.plot(y,y-y\_pred,'.')

    plt.xlabel('y')

    plt.ylabel('y - y\_pred');

    plt.title('std resid = {:.3f}'.format(std\_resid))

    ax\_133=plt.subplot(1,3,3)

    z.plot.hist(bins=50,ax=ax\_133)

    plt.xlabel('z')

    plt.title('{:.0f} samples with z>3'.format(n\_outliers))

    return model, cv\_score, grid\_results

# places to store optimal models and scores

opt\_models = dict()

score\_models = pd.DataFrame(columns=['mean','std'])

# no. k-fold splits

splits=5

# no. k-fold iterations

repeats=5

model = 'Ridge'

opt\_models[model] = Ridge()

alph\_range = np.arange(0.25,6,0.25)

param\_grid = {'alpha': alph\_range}

opt\_models[model],cv\_score,grid\_results = train\_model(opt\_models[model], param\_grid=param\_grid,

                                              splits=splits, repeats=repeats)

cv\_score.name = model

score\_models = score\_models.append(cv\_score)

plt.figure()

plt.errorbar(alph\_range, abs(grid\_results['mean\_test\_score']),

             abs(grid\_results['std\_test\_score'])/np.sqrt(splits\*repeats))

plt.xlabel('alpha')

plt.ylabel('score')

Fitting 25 folds for 
25) 
each of 23 candidates, totalling 
Using backend SequentialBackend 
Done 575 out of 575 | elapsed: 
575 fits 
with 1 concurrent workers. 
57.7s finished 
e .8926884448727849 
rmse= e. 324664e78e75827g 
mse= e. 19540676359282725 
cross val: mean: e.1e92%88e81827798 
Text(e, e.5, 
' score ' ) 
corr = 0.945 
std: e.øø892783ø1eøøøø986 
std resid = 0.325 
10 samples with 

goto 
019 
EIIO 
VttO 
SILO 

#### *拓展：*

[plt.errorbar官网介绍](https://matplotlib.org/gallery/lines_bars_and_markers/errorbar_limits_simple.html#sphx-glr-gallery-lines-bars-and-markers-errorbar-limits-simple-py)

[matplotlib可视化番外篇errorbar()--误差棒图](https://www.jianshu.com/p/5973680a1542)

[sklearn.model\_selection.GridSearchCV](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV)

#### *保存模型(*[*python3中pickle模块的dump()方法和load()方法*](https://blog.csdn.net/jpc20144055069/article/details/106004172?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.channel_param&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.channel_param)*):*

# Dictionary of best parameters

best\_pars = grid.best\_params\_

# Best XGB model that was found based on the metric score you specify

best\_model = grid.best\_estimator\_

# Save model

pickle.dump(grid.best\_estimator\_, open("xgb\_log\_reg.pickle", "wb"))

# load()方法

pkl\_file = open('data.pkl', 'rb')

data2 = pickle.load(pkl\_file)

pprint.pprint(data2)

pprint.pprint(data3)

|  |  |
| --- | --- |
| data3 = pickle.load(p | kl\_file) |

pkl\_file.close()

# 模型预测

Tips：

#利用[GridSearchCV](https://blog.csdn.net/weixin_41988628/article/details/83098130)求最优min\_impurity\_decrease参数

from sklearn.model\_selection import GridSearchCV

thresholds = np.linspace(0, 0.2, 50)

param\_grid = {'min\_impurity\_decrease':thresholds}

clf = GridSearchCV(DecisionTreeClassifier(), param\_grid, cv=5)

clf.fit(X\_train,y\_train)

print("best\_parms:{0}\nbest\_score:{1}".format(clf.best\_params\_, clf.best\_score\_))

tips:[朴素贝叶斯](https://blog.csdn.net/Dorisi_H_n_q/article/details/82783895)

朴素：独立性假设

一般来说，如果样本特征的分布大部分是连续值，使用GaussianNB会比较好。如果如果样本特征的分大部分是多元离散值，使用MultinomialNB比较合适。而如果样本特征是二元离散值或者很稀疏的多元离散值，应该使用BernoulliNB。

定义：

一个测试集看起来不太够呀，多轮求均值是一个好的策略！

1）的交叉验证的工作原理是首先将数据集分成k-subsets。

2）假设我们将数据集划分为（k＝5）部分。我们预留1个部分进行测试，并对这4个部分进行训练。

3）我们通过在每次迭代中改变测试部分并在其他部分中训练算法来继续这个过程。然后对衡量结果求平均值，得到算法的平均精度。

这就是所谓的交叉验证。

tipe:

tips：集成方法

集成方法通常包括如下两个类别:

平均方法（averaging methods）

该方法的原则是独立地构建多个学习器，然后将它们的预测结果进行平均。通常提供过组合之后的学习器要比单个的学习器性能更好，因为通过组合之后的模型降低了方差。

该法方法的代表有：

bagging(装袋)

随机森林（是一种以决策树为基学习器的Bagging算法）

提升方法（boosting methods)

与上一个方法所不同的是，提升方法的学习器是依次构建的，该方法试图去降低学习器的偏差。

该方法的代表有：

AdaBoost（自适应增强）

Gradient Tree Boosting（梯度提升树）

tips:bagging

对于分类问题我们可以采用baggingclassifier对于回归问题我们可以采用baggingregressor，通过设置参数max\_samples和max\_features我们可以指定子集大小和用于训练的特征的比例。

tips：

* + [adaboost、bagging、boosting的区别](https://blog.csdn.net/liulina603/article/details/78742614)：

Bagging和Boosting都是将已有的分类或回归算法通过一定方式组合起来，形成一个性能更加强大的分类器，更准确的说这是一种分类算法的组装方法。即将弱分类器组装成强分类器的方法。

* + Adaboost(adaptive boosting)是一种基于boost思想的一种自适应的迭代式算法。是boosting方法有许多不同的变形

tips：

[三种常见的集成学习框架：bagging，boosting和stacking](https://blog.csdn.net/zwqjoy/article/details/80431496)

用于减少方差的bagging

用于减少偏差的boosting

用于提升预测结果的stacking（大杀器）

tips：

[AdaBoost算法的比较(Discrete AdaBoost, Real AdaBoost, LogitBoost, Gentle Adaboost)](https://blog.csdn.net/weixin_30437847/article/details/99217423?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1.nonecase)

tips

[Gradient Tree Boosting（梯度提升树](https://blog.csdn.net/weixin_39777626/article/details/79859562)）两个模型：

GradientBoostRegressor GradientBoostingClassifier

tips：

#全称：Extreme Gradient Boosting

[#当GridSearch遇上XGBoost 一段代码解决调参问题](https://juejin.im/post/5b7669c4f265da281c1fbf96)

[#xgboost可以通过get\_fscore获取特征重要性](https://blog.csdn.net/sujinhehehe/article/details/84201415?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1.nonecase)

[#XGBoost参数调优完全指南（附Python代码）](https://blog.csdn.net/han_xiaoyang/article/details/52665396)

模型参数优化

2020年5月24日

15:43

# [常见的10大超参数优化库](https://blog.csdn.net/weixin_40255359/article/details/82193887?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.nonecase)

# cross\_val\_score 交叉验证与 K折交叉验证

## 资料

[cross\_val\_score 交叉验证与 K折交叉验证](https://www.bbsmax.com/A/mo5krv44Jw/)

[【Sklearn】数据划分方法](https://blog.csdn.net/weixin_41990278/article/details/94030724)

## 作用

sklearn cross\_val\_score 交叉验证，这个函数没有洗牌功能，添加K 折交叉验证，可以用来选择模型，也可以用来选择特征

## 参数

sklearn.model\_selection.cross\_val\_score(estimator, X, y=None, groups=None, scoring=None, cv=None, n\_jobs=1, verbose=0, fit\_params=None, pre\_dispatch=‘2\*n\_jobs’)

#### *scoring*

‘precision’（检测的准确率），‘recall’（正类被分类正确的部分），‘f1’（准确率和召回率的调和平均数）'neg\_mean\_squared\_error‘（回归）

#### *cv*

cv=5表示cross\_val\_score采用的是k-fold cross validation的方法，重复5次交叉验证

实际上，cross\_val\_score可以用的方法有很多，如kFold, leave-one-out, ShuffleSplit等，举例而言：

* cv=StratifiedKFold(n\_splits=5)其实就等价于cv=5（分层采样交叉切分，确保训练集，测试集中各类别样本的比例与原始数据集中相同。）
* cv=ShuffleSplit(n\_splits=3,test\_size=0.3,random\_state=0)

cross\_val\_score(model, X,y, cv=cv)

* cv=KFold(n\_splits=3, shuffle=False, random\_state=None)

cross\_val\_score(model, X,y, cv=cv)

## 举例一

def rmsle\_cv(model=None,X\_train\_head=None,y\_train=None):

n\_folds = 5

kf = KFold(n\_folds, shuffle=True, random\_state=seed).get\_n\_splits(X\_train\_head)

rmse= -cross\_val\_score(model, X\_train\_head, y\_train, scoring="neg\_mean\_squared\_error", cv = kf)

return(rmse)

# [贝叶斯优化参数](https://blog.csdn.net/ningyanggege/article/details/89003698)

* 与常规的网格搜索或者随机搜索的区别

贝叶斯调参采用高斯过程，考虑之前的参数信息，不断地更新先验；网格搜索未考虑之前的参数信息

贝叶斯调参迭代次数少，速度快；网格搜索速度慢,参数多时易导致维度爆炸

贝叶斯调参针对非凸问题依然稳健；网格搜索针对非凸问题易得到局部优最

## #BayesianOptimization

from sklearn.cross\_validation import cross\_val\_score

from bayes\_opt import BayesianOptimization

#黑盒子函数，输出越小越好

def rf\_cv(n\_estimators, min\_samples\_split, max\_features, max\_depth):

val = cross\_val\_score(

RandomForestClassifier(n\_estimators=int(n\_estimators),

min\_samples\_split=int(min\_samples\_split),

max\_features=min(max\_features, 0.999), # float

max\_depth=int(max\_depth),

random\_state=2

),

x, y, scoring='roc\_auc', cv=5

).mean()

return val

#实例化一个bayes优化对象

rf\_bo = BayesianOptimization(

rf\_cv,

{'n\_estimators': (10, 250),

'min\_samples\_split': (2, 25),

'max\_features': (0.1, 0.999),

'max\_depth': (5, 15)}

#运行bayes优化

rf\_bo.maximize()

lmt.ahzat.on 
Stepl Time I 
1 | oomoosl 
2 | oomoosl 
31 
4 | OOm02s I 
5 | OOrn03s I 
Value I max_depth I 
096787 | 
0.97010 | 
097150 | 
13,9545 | 
59750 | 
134843 | 
61950 | 
14.8414 | 
Bayesian Optimization 
Stepl Time I 
6 | 00m10s I 
7 | OOm07s I 
81 OOm10sl 
Valuel max_depth I 
5.8199 | 
0967181 5.4943 | 
148575 | 
097268 | 
max_features I 
0.3534 | 
04846 | 
0.9235 | 
0.6731 | 
0.8682 | 
max_features I 
04952 | 
0.4241 | 
07446 | 
6.3168 | 
21.6560 | 
7.39731 
12.5059 | 
22.1919 | 
2.3110 | 
2.1376 | 
22085 | 
I n_estimat 
31.0603 | 
13,2925 | 
108.9650 | 
174.1166 | 
220.9189 | 
I n_estimat 
249.739 | 
10.0733 | 
2018059 | 

#查看当前最优的参数和结果

rf\_bo.res['max']

{'max_params': 
{ ' max_depth ' : 
5 .819908283575526, 
•max features': ø.ag517d56Ø35Ø9127, 
2. 311€)014720414958, 
•n estimators 
249.73529231990733}, 
'max_val' : e.9774ø794e79ae794} 

#三组较优的超参数，让其在该参数基础上进行explore

rf\_bo.explore(

{'n\_estimators': [10, 100, 200],

'min\_samples\_split': [2, 10, 20],

'max\_features': [0.1, 0.5, 0.9],

'max\_depth': [5, 10, 15]

}

)

#修改高斯过程的参数，高斯过程主要参数是核函数(kernel)，还有其他参数可以参考[sklearn.gaussianprocess](http://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessRegressor.html)

gp\_param={'kernel':None}

rf\_bo.maximize(\*\*gp\_param)

'max features': 0.4951745603509127, 
'min samples split : 
' 2.3110014720414958, 
'n estimators': 
249.73529231990733), 
'max : 

## [#Hyperopt（自动贝叶斯调参）](https://blog.csdn.net/linxid/article/details/81189154)

* Hyheropt四个重要的因素：

指定需要最小化的函数，搜索的空间，采样的数据集(trails database)（可选），搜索的算法（可选）。

* 资料

[如何保存最好的hyperopt优化keras模型及其权重？](http://www.voidcn.com/article/p-secfhiik-bvr.html)

* [例一 xgboost参数优化](https://www.cnblogs.com/yangruiGB2312/p/9374377.html)

#coding:utf-8

import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

import xgboost as xgb

from random import shuffle

from xgboost.sklearn import XGBClassifier

from sklearn.cross\_validation import cross\_val\_score

import pickle

import time

from hyperopt import fmin, tpe, hp,space\_eval,rand,Trials,partial,STATUS\_OK

def loadFile(fileName = "E://zalei//browsetop200Pca.csv"):

data = pd.read\_csv(fileName,header=None)

data = data.values

return data

data = loadFile()

label = data[[:,-1]](https://blog.csdn.net/weixin_43291997/article/details/87185618)

attrs = data[:,:-1]

labels = label.reshape((1,-1))

label = labels.tolist()[0]

minmaxscaler = MinMaxScaler()

attrs = minmaxscaler.fit\_transform(attrs)

index = range(0,len(label))

shuffle(index)

trainIndex = index[:int(len(label)\*0.7)]

print len(trainIndex)

testIndex = index[int(len(label)\*0.7):]

print len(testIndex)

attr\_train = attrs[trainIndex,:]

print attr\_train.shape

attr\_test = attrs[testIndex,:]

print attr\_test.shape

label\_train = labels[:,trainIndex].tolist()[0]

print len(label\_train)

label\_test = labels[:,testIndex].tolist()[0]

print len(label\_test)

print np.mat(label\_train).reshape((-1,1)).shape

#定义需要最小化的函数（由于fmin是求最小值，因此求-auc的最小值）

def GBM(argsDict):

max\_depth = argsDict["max\_depth"] + 5

n\_estimators = argsDict['n\_estimators'] \* 5 + 50

learning\_rate = argsDict["learning\_rate"] \* 0.02 + 0.05

subsample = argsDict["subsample"] \* 0.1 + 0.7

min\_child\_weight = argsDict["min\_child\_weight"]+1

print "max\_depth:" + str(max\_depth)

print "n\_estimator:" + str(n\_estimators)

print "learning\_rate:" + str(learning\_rate)

print "subsample:" + str(subsample)

print "min\_child\_weight:" + str(min\_child\_weight)

global attr\_train,label\_train

gbm = xgb.XGBClassifier(nthread=4, #进程数

max\_depth=max\_depth, #最大深度

n\_estimators=n\_estimators, #树的数量

learning\_rate=learning\_rate, #学习率

subsample=subsample, #采样数

min\_child\_weight=min\_child\_weight, #孩子数

max\_delta\_step = 10, #10步不降则停止

objective="binary:logistic")

metric = cross\_val\_score(gbm,attr\_train,label\_train,cv=5,scoring="roc\_auc").mean()

print metric

return -metric

#搜索的空间

space = {"max\_depth":hp.randint("max\_depth",15),

"n\_estimators":hp.randint("n\_estimators",10), #[0,1,2,3,4,5] -> [50,]

"learning\_rate":hp.randint("learning\_rate",6), #[0,1,2,3,4,5] -> 0.05,0.06

"subsample":hp.randint("subsample",4),#[0,1,2,3] -> [0.7,0.8,0.9,1.0]

"min\_child\_weight":hp.randint("min\_child\_weight",5), #

}

#搜索算法

from hyperopt import tpe

# Algorithm

tpe\_algorithm = tpe.suggest

best = fmin(GBM,space,algo=tpe\_algorithm,max\_evals=4)

print best

'max features': 0.4951745603509127, 
'min samples split : 
' 2.3110014720414958, 
'n estimators': 
249.73529231990733), 
'max : 

print GBM(best)

|  |  |
| --- | --- |
|  | [例二 lightgbm超参数搜索](https://blog.csdn.net/linxid/article/details/81189154) |

from hyperopt import fmin, tpe, hp, STATUS\_OK, Trials

import lightgbm as lgb

from hyperopt import STATUS\_OK

N\_FOLDS = 10

*# Create the dataset*

train\_set = lgb.Dataset(train\_features, train\_labels)

#### *#1、目标函数*

def objective(params, n\_folds=N\_FOLDS):

'''Objective function for Gradient Boosting Machine Hyperparameter Tuning'''

# Perform n\_fold cross validation with hyperparameters

# Use early stopping and evalute based on ROC AUC

cv\_results = lgb.cv(params, train\_set, nfold=n\_folds, num\_boost\_round=10000,

early\_stopping\_rounds=100, metrics='auc', seed=50)

# Extract the best score

best\_score = max(cv\_results['auc-mean'])

# Loss must be minimized

loss = 1 - best\_score

# Dictionary with information for evaluation

return {'loss': loss, 'params': params, 'status': STATUS\_OK}

#### *#2、搜索的空间*

from hyperopt import hp

space = {

'class\_weight': hp.choice('class\_weight', [None, 'balanced']),

'boosting\_type': hp.choice('boosting\_type',

[{'boosting\_type': 'gbdt',

'subsample': hp.uniform('gdbt\_subsample', 0.5, 1)},

{'boosting\_type': 'dart',

'subsample': hp.uniform('dart\_subsample', 0.5, 1)},

{'boosting\_type': 'goss'}]),

'num\_leaves': hp.quniform('num\_leaves', 30, 150, 1),

'learning\_rate': hp.loguniform('learning\_rate', np.log(0.01), np.log(0.2)),

'subsample\_for\_bin': hp.quniform('subsample\_for\_bin', 20000, 300000, 20000),

'min\_child\_samples': hp.quniform('min\_child\_samples', 20, 500, 5),

'reg\_alpha': hp.uniform('reg\_alpha', 0.0, 1.0),

'reg\_lambda': hp.uniform('reg\_lambda', 0.0, 1.0),

'colsample\_bytree': hp.uniform('colsample\_by\_tree', 0.6, 1.0)

#### *#3、采样的数据集*

from hyperopt.pyll.stochasti import sample

# Sample from the full space

example = sample(space)

# Dictionary get method with default

subsample = example['boosting\_type'].get('subsample', 1.0)

# Assign top-level keys

example['boosting\_type'] = example['boosting\_type']['boosting\_type']

example['subsample'] = subsample

example

{'boosting\_type': 'gbdt',

'class\_weight': 'balanced',

'colsample\_bytree': 0.8111305579351727,

'learning\_rate': 0.16186471096789776,

'min\_child\_samples': 470.0,

'num\_leaves': 88.0,

'reg\_alpha': 0.6338327001528129,

'reg\_lambda': 0.8554826167886239,

'subsample\_for\_bin': 280000.0,

'subsample': 0.6318665053932255}

#### *#4、优化算法*

from hyperopt import tpe

# Algorithm

tpe\_algorithm = tpe.suggest

#### *#5、保存历史结果（可选）*

from hyperopt import Trials

# Trials object to track progress

bayes\_trials = Trials()

import csv#写入cvs文件

#### *# File to save first results*

out\_file = 'gbm\_trials.csv'

of\_connection = open(out\_file, 'w')

writer = csv.writer(of\_connection)

#### *# Write the headers to the file*

writer.writerow(['loss', 'params', 'iteration', 'estimators', 'train\_time'])

of\_connection.close()

#### *#每次迭代时添加行写入csv*

# Write to the csv file ('a' means append)

of\_connection = open(out\_file, 'a')

writer = csv.writer(of\_connection)

writer.writerow([loss, params, iteration, n\_estimators, run\_time])

of\_connection.close()

#6、优化

from hyperopt import fmin

MAX\_EVALS = 500

# Optimize

best = fmin(fn = objective, space = space, algo = tpe.suggest,

max\_evals = MAX\_EVALS, trials = bayes\_trials)

{ 'boosting_type': 'gbdt' , 
• class_weight• : 'balanced • , 
•cols.le_bytree• : ø.7125187W5392453, 
'learning_rate• : ø.ø22592570862øaa956, 
• : 250, 
•num leaves': 49, 
'reg_alpha ' : ø.2ø352116431øa735, 
•reg_lambda• : e.6d5513171592gø91, 
'subsanvle' : ø.983566228e71919, 
subsanvle_for bin' : 2Wøøø} 

#拓展：

choice：类别变量

quniform：离散均匀（整数间隔均匀）

uniform：连续均匀（间隔为一个浮点数）

loguniform：连续对数均匀（对数下均匀分布）

# boosting type domain

boosting\_type = {'boosting\_type': hp.choice('boosting\_type',

[{'boosting\_type': 'gbdt',

'subsample': hp.uniform('subsample', 0.5, 1)},

{'boosting\_type': 'dart',

'subsample': hp.uniform('subsample', 0.5, 1)},

{'boosting\_type': 'goss',

'subsample': 1.0}])}

搜索算法：

①随机搜索(hyperopt.rand.suggest)

②模拟退火(hyperopt.anneal.suggest)

③TPE算法（hyperopt.tpe.suggest，算法全称为Tree-structured Parzen Estimator Approach）

1. options) where options should be a python list or tuple. 100] , "L*choice , fiFfiiZRquniform 
2. upper) , , qunitorm 
3 low, nigh) where low and nigh are the lower and upper bounds on the range. 
Ilow,highl 
4. low, high) where low and nigh are the lower and upper bounds on the range. , [ 
1.0 , , 
5. IOW, [dou , ehigh] , , , 
6. low, high, q) 
7. mu, sigma) where mu and sigma are the mean and standard deviation a , respectively. , 
& mu, sigma, q) 
9. mu, sigma) 
10 mu, sigma, q) 

[python使用hyperopt工具进行自动调参](https://www.jianshu.com/p/e1bda6355452)

[多种模型调参举例](https://blog.csdn.net/u012735708/article/details/84820101)

## [Grid SearchCV（网格搜索）](https://www.cnblogs.com/wj-1314/p/10422159.html)

from sklearn.model\_selection import GridSearchCV

param\_grid = [

{'n\_estimators': [3, 10, 30], 'max\_features': [2, 4, 6, 8]},

{'bootstrap': [False], 'n\_estimators': [3, 10], 'max\_features': [2, 3, 4]},

]

forest\_reg = RandomForestRegressor()

grid\_search = GridSearchCV(forest\_reg, param\_grid, cv=5,

                          scoring='neg\_mean\_squared\_error')

grid\_search.fit(housing\_prepared, housing\_labels)

#返回最优的训练器

# extract best model from the grid

model = grid\_search.best\_estimator\_

best\_idx = grid\_search.best\_index\_

# get cv-scores for best model

grid\_results = pd.DataFrame(grid\_search.cv\_results\_)

cv\_mean = abs(grid\_results.loc[best\_idx,'mean\_test\_score'])

cv\_std = grid\_results.loc[best\_idx,'std\_test\_score']

print(best\_estimator)

[模型融合](https://tianchi.aliyun.com/notebook-ai/detail?spm=5176.12586969.1002.9.1cd866c2JJX57l&postId=95535)

2020年5月27日

15:25

* + 模型融合
    1. 简单加权融合:
       - 回归（分类概率）：算术平均融合（Arithmetic mean），几何平均融合（Geometric mean）；
       - 分类：投票（Voting)
       - 综合：排序融合(Rank averaging)，log融合
    2. stacking/blending:
       - 构建多层模型，并利用预测结果再拟合预测。
    3. boosting/bagging（在xgboost，Adaboost,GBDT中已经用到）:
       - 多树的提升方法

# 简单加权融合

## Stacking

### [StackingCVClassifier](https://blog.csdn.net/LAW_130625/article/details/78573736)

* + [模型融合—— stacking详细讲解](https://blog.csdn.net/u011630575/article/details/81302994)
  + [StackingCVClassifier代码举例](https://blog.csdn.net/LAW_130625/article/details/78573736)
  + [StackingCVClassifier参数和举例](https://www.pianshen.com/article/4747167101/)
  + [模型融合---Stacking调参总结](https://www.cnblogs.com/nxf-rabbit75/p/10596180.html)

评价指标

2020年9月23日

10:45

# 分类型模型评判的指标

## 相关资料

[分类模型评判指标（一）](https://blog.csdn.net/Orange_Spotty_Cat/article/details/80520839)

[类模型评判指标（二）](https://blog.csdn.net/Orange_Spotty_Cat/article/details/80499031?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.channel_param&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.channel_param)

## 混淆矩阵

Positive 
Positive 
Negative 
Type Il 
Negative 
TN 

#### *代码实现*

## 混淆矩阵

import numpy as np

from sklearn.metrics import confusion\_matrix

y\_pred = [0, 1, 0, 1]

y\_true = [0, 1, 1, 0]

print('混淆矩阵:\n',confusion\_matrix(y\_true, y\_pred))

#### *二级指标*

TN 
ACC 
TPR 
TNR 
Accuracy 
Precision 
Sensitivity 
Specificity 
Recall 
TN 
TN + FP 

#### *三级指标*

2PR 
Fl Score 

##### 代码实现

## accuracy

from sklearn.metrics import accuracy\_score

y\_pred = [0, 1, 0, 1]

y\_true = [0, 1, 1, 0]

print('ACC:',accuracy\_score(y\_true, y\_pred))

## Precision,Recall,F1-score

from sklearn import metrics

y\_pred = [0, 1, 0, 1]

y\_true = [0, 1, 1, 0]

print('Precision',metrics.precision\_score(y\_true, y\_pred))

print('Recall',metrics.recall\_score(y\_true, y\_pred))

print('F1-score:',metrics.f1\_score(y\_true, y\_pred))

## ROC曲线

受试者工作特征曲线（Receiver Operating Characteristic Curve）

• Support Vector Classifier 
• LOA 
00.0 
0.4 

横轴写着“False positive rate”，纵轴写着“True positive rate”

#### *代码实现*

## ROC曲线

from sklearn.metrics import roc\_curve

y\_pred = [0, 1, 1, 0, 1, 1, 0, 1, 1, 1]

y\_true = [0, 1, 1, 0, 1, 0, 1, 1, 0, 1]

FPR,TPR,thresholds=roc\_curve(y\_true, y\_pred)

plt.title('ROC')

plt.plot(FPR, TPR,'b')

plt.plot([0,1],[0,1],'r--')

plt.ylabel('TPR')

plt.xlabel('FPR')

## AUC（Area Under Curve）

为ROC曲线下与坐标轴围成的面积。

#### *代码实现*

## AUC

import numpy as np

from sklearn.metrics import roc\_auc\_score

y\_true = np.array([0, 0, 1, 1])

y\_scores = np.array([0.1, 0.4, 0.35, 0.8])

print('AUC socre:',roc\_auc\_score(y\_true, y\_scores))



## KS值

KS不同代表的不同情况，一般情况KS值越大，模型的区分能力越强

20 以 下 
41 ． 50 
51 ． 匐 
61 一 75 
75 以 上 
非 幫 强 
于 高 ， 疑 似 存 在 石 

#### *代码实现*

## KS值 在实际操作时往往使用ROC曲线配合求出KS值

from sklearn.metrics import roc\_curve

y\_pred = [0, 1, 1, 0, 1, 1, 0, 1, 1, 1]

y\_true = [0, 1, 1, 0, 1, 0, 1, 1, 1, 1]

FPR,TPR,thresholds=roc\_curve(y\_true, y\_pred)

KS=abs(FPR-TPR).max()

print('KS值：',KS)

## 相关知识

混淆矩阵（Confuse Matrix）

# 回归评价指标

## 资料

[回归评价指标MSE、RMSE、MAE、R-Squared](https://www.jianshu.com/p/9ee85fdad150)

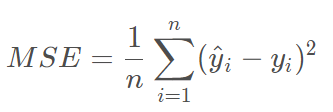
[sklearn.metrics.make\_scorer详解](https://blog.csdn.net/qq_43327574/article/details/105601819)

score func 
greater_is_better 
needs_proba 
gnu True, 
needs threshold 
nsklearn.metrics.accuracy_score$, 
s klearn. metri .mean_squa red_error$ 
iEfBhTrue ; 

[sklearn中的模型评估-构建评估函数](https://www.cnblogs.com/harvey888/p/6964741.html)

[预测评价指标RMSE、MSE、MAE、MAPE、SMAPE](https://blog.csdn.net/guolindonggld/article/details/87856780?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.add_param_isCf&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.add_param_isCf)

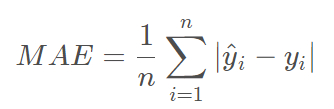
MSE 均方误差（Mean Square Error）



RMSE 均方根误差（Root Mean Square Error）

RMSE - 
E(ji - Yi)2 

MAE 平均绝对误差（Mean Absolute Error）



# coding=utf-8

import numpy as np

from sklearn import metrics

# MAPE和SMAPE需要自己实现

def mape(y\_true, y\_pred):

    return np.mean(np.abs((y\_pred - y\_true) / y\_true)) \* 100

def smape(y\_true, y\_pred):

    return 2.0 \* np.mean(np.abs(y\_pred - y\_true) / (np.abs(y\_pred) + np.abs(y\_true))) \* 100

y\_true = np.array([1.0, 5.0, 4.0, 3.0, 2.0, 5.0, -3.0])

y\_pred = np.array([1.0, 4.5, 3.5, 5.0, 8.0, 4.5, 1.0])

# MSE

print(metrics.mean\_squared\_error(y\_true, y\_pred)) # 8.107142857142858

# RMSE

print(np.sqrt(metrics.mean\_squared\_error(y\_true, y\_pred))) # 2.847304489713536

# MAE

print(metrics.mean\_absolute\_error(y\_true, y\_pred)) # 1.9285714285714286

# MAPE

print(mape(y\_true, y\_pred)) # 76.07142857142858，即76%

# SMAPE

print(smape(y\_true, y\_pred)) # 57.76942355889724，即58%

## 代码实现

from sklearn.model\_selection import GridSearchCV, RepeatedKFold, cross\_val\_score,cross\_val\_predict,KFold

from sklearn.metrics import make\_scorer,mean\_squared\_error

# metric for evaluation

def rmse(y\_true, y\_pred):

diff = y\_pred - y\_true

sum\_sq = sum(diff\*\*2)

n = len(y\_pred)

return np.sqrt(sum\_sq/n)

def mse(y\_ture,y\_pred):

return mean\_squared\_error(y\_ture,y\_pred)

# scorer to be used in sklearn model fitting

rmse\_scorer = make\_scorer(rmse, greater\_is\_better=False)

mse\_scorer = make\_scorer(mse, greater\_is\_better=False)

必备封装代码

2020年5月24日

11:08

**目录**

[内存优化](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={DB75A166-F0A3-4567-ADDD-62F690D34010}&99&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one)

[特征工程](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={2248AC01-0E63-437E-B5E4-12AFC346ACE0}&A&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one)

[#等距分桶操作](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={8ACA6BB0-FEE4-48C3-9AA1-D9E829FEBEA9}&D&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one)

[# count编码](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={8ACA6BB0-FEE4-48C3-9AA1-D9E829FEBEA9}&19&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one)

[#定义交叉特征统计（类别交叉连续）](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={8ACA6BB0-FEE4-48C3-9AA1-D9E829FEBEA9}&23&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one)

[# 类别特征的二阶交叉（类别交叉类别）](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={8ACA6BB0-FEE4-48C3-9AA1-D9E829FEBEA9}&39&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one)

[测试和训练集的处理](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={DB75A166-F0A3-4567-ADDD-62F690D34010}&8E&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one)

[模型验证](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={DB75A166-F0A3-4567-ADDD-62F690D34010}&93&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one)

[模型融合](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={DB75A166-F0A3-4567-ADDD-62F690D34010}&6E&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one)

# 随机生成Dataframe

#### [*np.random的各种生成随机数方法*](https://blog.csdn.net/weixin_39374967/article/details/103655190)

ar = np.random.randint(1,20,30).reshape(10,3)

print(ar)

df1 = pd.DataFrame(ar)

df2 = pd.DataFrame(ar, columns = ["one", "two", "three"])

# 数据预处理

# 数据转化和存取

# tqdm可视化迭代

# 运行时间

# 内存优化 ([页首](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={5E592C28-1627-4767-84A7-9236DE75CAD1}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one))

# 特征工程 ([页首](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={5E592C28-1627-4767-84A7-9236DE75CAD1}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one))

# 测试和训练集的处理 ([页首](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={5E592C28-1627-4767-84A7-9236DE75CAD1}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one))

# 模型验证 ([页首](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={5E592C28-1627-4767-84A7-9236DE75CAD1}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one))

# 根据训练好的模型删除无法拟合的数据

# 多模型评估指标存储

# 模型融合 ([页首](onenote:#必备封装代码&section-id={F318C780-999A-4125-A0F2-725AFBFFDF25}&page-id={43E255B5-D069-4C2F-9CF6-EB6DF976F603}&object-id={5E592C28-1627-4767-84A7-9236DE75CAD1}&F&base-path=https://d.docs.live.net/04abe269f00f36a1/文档/数据挖掘/数据挖掘实战.one))

debug

2020年5月28日

22:07

#SpecificationError: nested renamer is not supported

import pandas as pd

import numpy as np

df = pd.DataFrame({'A': ['a', 'b', 'a', 'c', 'a', 'c', 'b', 'c'],

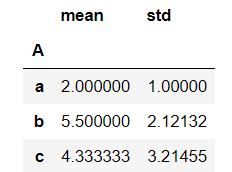
'B': [2, 7, 1, 3, 3, 2, 4, 8],

'C': [100, 87, 96, 130, 105, 87, 96, 155]})

df.groupby('A', as\_index=False)['B'].agg({'mean':np.mean, 'std': np.std})



df.groupby('A')['B'].agg([('mean',np.mean), ('std', np.std)])



问题

2020年5月15日

21:41

什么是bias和variance

k折交叉验证中k取值多少有什么关系

分类模型和回归模型的区别是什么

为什么会产生过拟合

如何解决过拟合

写出信息增益，信息增益比和基尼系数的公式

写出LR的目标函数和损失函数

LR和SVM的异同

推导SVM

SVM在哪个地方引入核函数

常用的核函数有哪些，说说它们的特点及调参方式

随机森林的学习过程

随机森林的优缺点

GBDT和随机森林的区别

如何判断决策树过拟合

线性分类器与非线性分类器的区别及优劣

特征比数据量还大时，选择什么样的分类器

L1和L2正则的区别，如何选择L1和L2正则

多分类怎么处理

如何进行特征选择

吐槽一个推荐算法，并提出修改意见

知识速查

2020年5月5日

9:20

# 魔法命令

## 资料

[安利十二个常用的IPython魔法命令](https://blog.csdn.net/weixin_38753213/article/details/108414023)

# [os库](https://www.cnblogs.com/Lengjie/p/9391390.html)

#### *三个作用*

路径操作：os.path子库，处理文件路径及信息

进程管理：启动系统中其他程序

环境参数：获得系统软硬件信息等环境参数

#### *导入*

import os.path as op

import os.system as ost

# 数字

## Series排列

[pandas.Series.rank](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.rank.html#pandas-series-rank)

## 迭代对象排列

## 随机数生成五种方法

np.random.rand(X,Y) #返回一个或一组服从标准正态分布的随机样本值。

np.random.randn(X,Y) #一个或一组服从“0~1”均匀分布的随机样本值。随机样本取值范围是[0,1)，不包1

numpy.random.randint(下限, 上限, size=矩阵规格) #返回整数

range(下限, 上限（不包括）)

np.arange(1, 5, .5) #输出：array([ 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])

## [python 求均值、中位数、众数的方法](https://zhuanlan.zhihu.com/p/46661241)

nums = [0,4,5,8,8]

#求均值和中位数均可以使用numpy库的方法：  
import numpy as np  
#均值  
np.mean(nums)  
#中位数  
np.median(nums)

#求均值和中位数均也可以使用panda库的方法：  
import panda as pd  
gm = pd.Series(data=nums)  
#均值  
gm.mean()  
#中位数  
gm.median()

求众数方法一

import numpy as np

#获取平铺后每个索引位置值在原始数列中出现的次数  
counts = np.bincount(nums)

#返回众数 返回最大值在数列中的索引位置  
np.argmax(counts)

求众数方法二——直接利用scipy下stats模块

from scipy import stats

nums = [0,4,5,8,8]

stats.mode(nums)[0][0]

求众数方法三——panda下mode()方法直接获取众数

import pandas as pd

v = [3,5,6,2,6,9]

gm = pd.Series(data=v)

gm.mode()[0]

#### [*分组求众数*](https://blog.csdn.net/Yvettre/article/details/79726396)

# 绘图

## numpy自带的绘图接口

DataFrame.plot(x=None, y=None, kind='line', ax=None, subplots=False,

sharex=None, sharey=False, layout=None,figsize=None,

use\_index=True, title=None, grid=None, legend=True,

style=None, logx=False, logy=False, loglog=False,

xticks=None, yticks=None, xlim=None, ylim=None, rot=None,

xerr=None,secondary\_y=False, sort\_columns=False, \*\*kwds

## matplotlib函数大全

plt.plot(x,y ,fmt)  ：绘制坐标图

plt.boxplot(data,notch, position): 绘制箱形图

plt.bar(left,height, width, bottom) : 绘制条形图

plt.barh(width,bottom, left, height) : 绘制横向条形图

plt.polar(theta,r) : 绘制极坐标图

plt.pie(data,explode) : 绘制饼图

plt.scatter(x,y) :绘制散点图

plt.hist(x,bings, normed) : 绘制直方图

plt.xlabel()：对x轴增加文本标签

plt.ylabel()：同理

plt.title(): 对图形整体增加文本标签

## [seaborn绘图](https://blog.csdn.net/weixin_42983055/article/details/82661864)

#### *相关资料：*

[seaborn简介和实例](https://blog.csdn.net/qq_34264472/article/details/53814653?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1.nonecase)

#### *显示中文*

import matplotlib as mpl

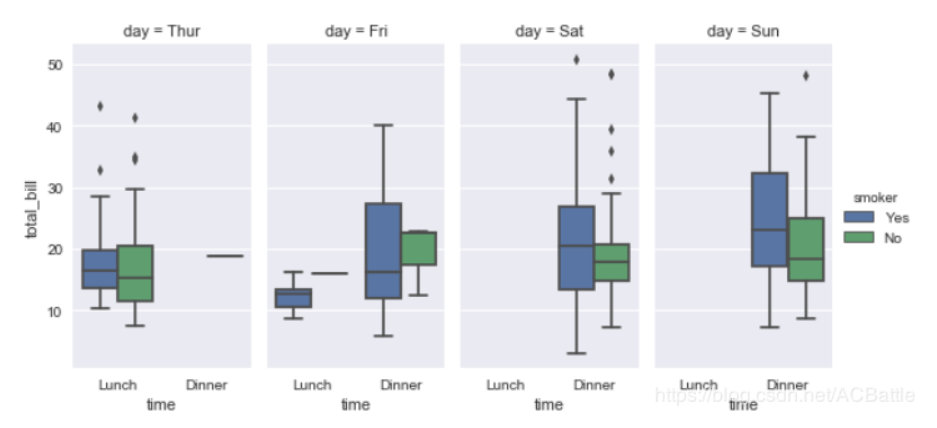
mpl.rcParams['font.sans-serif'] = ['SimHei'] # 指定默认字体

mpl.rcParams['axes.unicode\_minus'] = False # 解决保存图像是负号'-'显示为方块的问题

#### [*sns.factorplot参数*](https://blog.csdn.net/acbattle/article/details/86490450)

sns.factorplot(x='time',y='total\_bill',hue='smoker', col='day',data=tips,kind='box',size=4,aspect=0.5)

参数kind：point默认，bar柱形图，count频次，box箱体，violin提琴，strip散点，swarm分散点



# 字符串

## 相关资料：

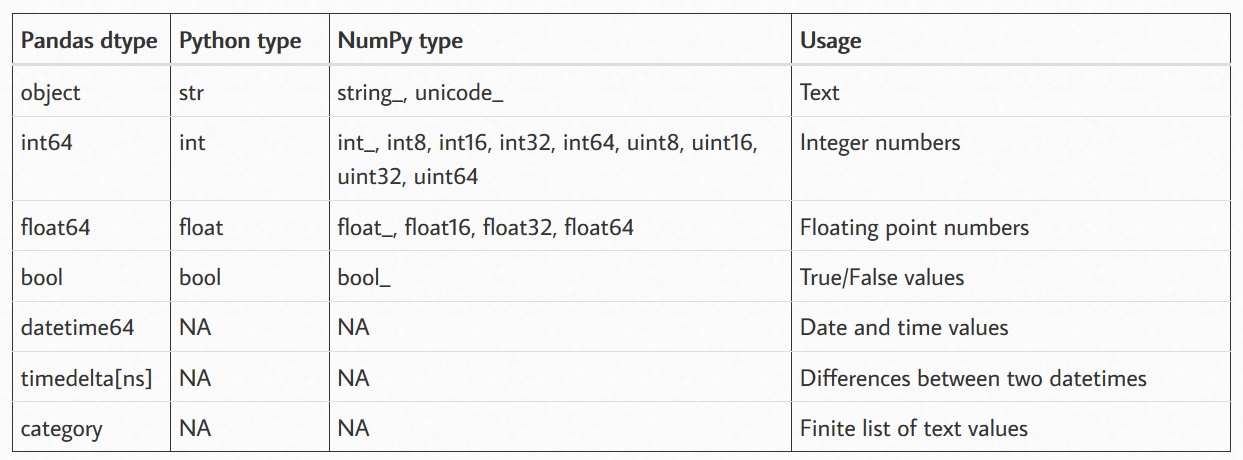
#### [*#Python format 格式化函数*](https://www.runoob.com/python/att-string-format.html)

# 其他表格

## scoring

Scoring 
Classification 
'accuracy' 
'average_precision' 
Tl micro' 
fl macro' 
fl _wevgnted' 
Tl _samples' 
'neg_log loss' 
precision' etc. 
'recall' etc. 
'roc auc' 
Clustering 
•adjusted _ rand_score' 
Regression 
'neg_mean_absolute_error' 
'neg_mean_squared_error' 
neg_median_absolute_error' 
'r2' 
Function 
—t rics- 
•t rics- f I smre 
•t rics. f I 
—t rics. f I sa»re 
—t rics- f smre 
•t rics- f I smre 
rics. 
—t rics- re:aII sa»re 
•t rics- ra: •c •:ore 
rics. *bust ed_r 
•t rics- 
rics- 
rics- 
—t rics. 
Comment 
tor binary targets 
micro-averaged 
macro-averaged 
weighted average 
by multilabel sample 
requires predict_prcba support 
suffixes appty as with fl' 
suffixes appty as with fl ' 

#### [*数据类型转换*](https://zhuanlan.zhihu.com/p/35287822)



先 简 单 介 生 一 下 且 个 常 用 。 racle 中 的 犟 言 函 麴 。 
O cle 中 合 函 数 名 称 
Count 
Max 
Min 
Sum 
Variance 
Stddev 
函 数 的 作 用 
用 来 求 有 效 数 据 的 数 量 
用 来 求 给 定 数 撫 中 最 大 的 那 一 个 数 据 
用 宗 求 给 定 数 据 中 最 小 的 那 一 个 数 据 
用 求 给 定 数 据 的 十 均 值 
用 来 求 绉 定 数 据 的 总 和 
来 求 给 定 数 的 标 准 差 
用 来 求 给 定 数 据 的 方 差 

### print的花里胡哨

print(f"response: {response[变量]}")



# 疑难杂症

[#如何解决Python下 pip install module 下载慢解决方法？](https://blog.csdn.net/u010899985/article/details/79187141)

, restore\_best\_weights=True

[python中分组函数groupby和分组运算函数agg小结](https://blog.csdn.net/weixin_37536446/article/details/82109431)

√group\_df = df.groupby('ship')['type'].agg([['label', 'mean'], ['cnt','count']]).reset\_index()

×group\_df = df.groupby('ship')['type'].agg({'label': 'mean', 'cnt': 'count'}).reset\_index()

## RuntimeError: This event loop is already running?

import nest\_asyncio

nest\_asyncio.apply()

[Python 解决写入csv中间隔一行空行问题](https://www.cnblogs.com/littlebob/p/9293931.html)

解决办法就是写入后面加上newline=''

with open(birth\_weight\_file,'w',newline='') as f:

writer=csv.writer(f)

writer.writerow(birth\_header)

writer.writerows(birth\_data)

f.close()

[关于运行时：matplotlib摆脱max\_open\_warning输出](https://www.codenong.com/27476642/)

import matplotlib as plt

plt.rcParams.update({'figure.max\_open\_warning': 0})

或

import matplotlib as mpl

mpl.rc('figure', max\_open\_warning = 0)