www.cnblogs.com

# python进行机器学习（一）之数据预处理 - 光彩照人 - 博客园

*光彩照人 关注 - 7 粉丝 - 117 +加关注*

4-5 minutes

---

**一、加载数据**

houseprice=pd.read_csv('../input/train.csv') #加载后放入dataframe里

all_data=pd.read_csv('a.csv', header=0,parse_dates=['time'],usecols=['time','LotArea','price'])  #可以选择加载哪几列houseprice.head()  #显示前5行数据

houseprice.info()  #查看各字段的信息

houseprice.shape  #查看数据集行列分布，几行几列

houseprice.describe()  #查看数据的大体情况

```
df.describe()
```

| | Id | MSSubClass | LotFrontage | LotArea | OverallQual | OverallCond | YearBuilt |
|---|---|---|---|---|---|---|---|
| **count** | 1460.000000 | 1460.000000 | 1201.000000 | 1460.000000 | 1460.000000 | 1460.000000 | 1460.000000 |
| **mean** | 730.500000 | 56.897260 | 70.049958 | 10516.828082 | 6.099315 | 5.575342 | 1971.267808 |
| **std** | 421.610009 | 42.300571 | 24.284752 | 9981.264932 | 1.382997 | 1.112799 | 30.202904 |
| **min** | 1.000000 | 20.000000 | 21.000000 | 1300.000000 | 1.000000 | 1.000000 | 1872.000000 |
| **25%** | 365.750000 | 20.000000 | 59.000000 | 7553.500000 | 5.000000 | 5.000000 | 1954.000000 |
| **50%** | 730.500000 | 50.000000 | 69.000000 | 9478.500000 | 6.000000 | 5.000000 | 1973.000000 |
| **75%** | 1095.250000 | 70.000000 | 80.000000 | 11601.500000 | 7.000000 | 6.000000 | 2000.000000 |
| **max** | 1460.000000 | 190.000000 | 313.000000 | 215245.000000 | 10.000000 | 9.000000 | 2010.000000 |

8 rows × 38 columns

**二、分析缺失数据**

houseprice.isnull()  #元素级别的判断，把对应的所有元素的位置都列出来，元素为空或者NA就显示True，否则就是False

| | Id | MSSubClass | MSZoning | LotFrontage | LotArea | Street | Alley |
|---|---|---|---|---|---|---|---|
| 0 | False | False | False | False | False | False | True |
| 1 | False | False | False | False | False | False | True |
| 2 | False | False | False | False | False | False | True |
| 3 | False | False | False | False | False | False | True |
| 4 | False | False | False | False | False | False | True |
| 5 | False | False | False | False | False | False | True |
| 6 | False | False | False | False | False | False | True |
| 7 | False | True | False | True | False | False | True |
| 8 | False | False | False | False | False | False | True |
| 9 | False | False | False | False | False | False | True |

```
houseprice.isnull().any()                    #列级别的判断，只要该列有为空或者NA的元素，就为True，否则False
```

```
Id             False
MSSubClass      True
MSZoning       False
LotFrontage     True
LotArea        False
Street         False
Alley           True
dtype: bool
```

```
missing=houseprice.columns[houseprice.isnull().any()].tolist()          #将为空或者NA的列找出来
```

```
>>>
['MSSubClass', 'LotFrontage', 'Alley']
>>>
```

```
houseprice[missing].isnull().sum()                    #将列中为空或者NA的个数统计出来
```

```
...
MSSubClass      1
LotFrontage     1
Alley          10
dtype: int64
>>>
```

```
# 将某一列中缺失元素的值，用value值进行填充。处理缺失数据时，比如该列都是字符串，不是数值，可以将出现次数最多的字符串填充缺失值。
def cat_imputation(column, value):
        houseprice.loc[houseprice[column].isnull(),column] = value
```

```
houseprice[['LotFrontage','Alley']][houseprice['Alley'].isnull()==True]      #从LotFrontage 和Alley 列中进行选择行，选择Alley中数据为空的行。主要用来看两个列的关联程度，是不是大多同时为空。
```

```
houseprice['Fireplaces'][houseprice['FireplaceQu'].isnull()==True].describe()
#对筛选出来的数据做一个描述，比如一共多少行，均值、方差、最小值、最大值等等。
```

**三、统计分析**

```
houseprice['MSSubClass'].value_counts()          #统计某一列中各个元素值出现的次数
```

```
print("Skewness: %f" % houseprice['MSSubClass'].skew())     #列出数据的偏斜度
```

```
print("Kurtosis: %f" % houseprice['MSSubClass'].kurt())
```

#列出数据的峰度

```
houseprice['LotFrontage'].corr(houseprice['LotArea'])          #计算两个列的相关度

    houseprice['SqrtLotArea']=np.sqrt(houseprice['LotArea'])    #将列的数值求根，并
```

赋予一个新列

```
houseprice[['MSSubClass', 'LotFrontage']].groupby(['MSSubClass'],
```

as_index=False).mean()  #跟MSSubClass进行分组，并求分组后的平均值

| | MSSubClass | LotFrontage |
|---|---|---|
| 0 | 20 | 77.500000 |
| 1 | 50 | 68.000000 |
| 2 | 60 | 72.333333 |
| 3 | 70 | 60.000000 |

**四、数据处理**

**1)删除相关**

```
del houseprice['SqrtLotArea']       #删除列

 houseprice['LotFrontage'].dropna()    #去掉为空值或者NA的元素

houseprice.drop(['Alley'],axis=1)      #去掉Alley列，不管空值与否

df.drop(df.columns[[0,1]],axis=1,inplace=True)  #删除第1，2列，inplace=True表示直
```

接就在内存中替换了，不用二次赋值生效。

```
    houseprice.dropna(axis=0)          #删除带有空值的行

    houseprice.dropna(axis=1)         #删除带有空值的列
```

**2）缺失值填充处理**

```
houseprice['LotFrontage']=houseprice['LotFrontage'].fillna(0)       #将该列中的空
```

值或者NA填充为0

```
all_data.product_type[all_data.product_type.isnull()]=all_data.product_type.dropna().mode().val
```

#如果该列是字符串的，就将该列中出现次数最多的字符串赋予空值,mode()函数就是取出现次数最多的元素。

```
houseprice['LotFrontage'].fillna(method='pad')                              #使用
```

前一个数值替代空值或者NA，就是NA前面最近的非空数值替换

```
houseprice['LotFrontage'].fillna(method='bfill',limit=1)  #使用后一个数值替代空值
```

或者NA，limit=1就是限制如果几个连续的空值，只能最近的一个空值可以被填充。

```
houseprice['LotFrontage'].fillna(houseprice['LotFrontage'].mean())      #使用平均
```

值进行填充

```
houseprice['LotFrontage'].interpolate()                # 使用插值来估计NaN 如果index是数字，可以设置参数method='value' ，如果是时间，可以设置method='time'
houseprice= houseprice.fillna(houseprice.mean())  #将缺失值全部用该列的平均值代替，这个时候一般已经提前将字符串特征转换成了数值。
```

注：在kaggle中有人这样处理缺失数据，如果数据的缺失达到15%，且并没有发现该变量有多大作用，就删除该变量！

**3）字符串替换**

```
houseprice['MSZoning']=houseprice['MSZoning'].map({'RL':1,'RM':2,'RR':3,}).astype(int)
    #将MSZoning中的字符串变成对应的数字表示
```

**4）数据连接**

```
merge_data=pd.concat([new_train,df_test])  #讲训练数据与测试数据连接起来，以便一起进行数据清洗

    all_data = pd.concat((train.loc[:,'MSSubClass':'SaleCondition'], test.loc[:,'MSSubClass':'SaleCondition']))  #另一种合并方式，按列名字进行合并。

     res = pd.merge(df1, df2,on=['time'])     #将df1,df2按照time字段进行合并，两个df中都含有time字段
```

**5）数据保存**

```
merge_data.to_csv('merge_data.csv', index=False)    #index=False，写入的时候不写入列的索引序号
```

**6）数据转换**

```
houseprice["Alley"] = np.log1p(houseprice["Alley"])     #采用log(1+x)方式对原数据进行处理，改变原数据的偏斜度，使数据更加符合正态曲线分布。

    numeric_feats =houseprice.dtypes[houseprice.dtypes != "object"].index     #把内容为数值的特征列找出来
```

**#下面几行代码将偏斜度大于0.75的数值列做一个log转换，使之尽量符合正态分布，因为很多模型的假设数据是服从正态分布的**

```
skewed_feats = train[numeric_feats].apply(lambda x: skew(x.dropna())) #compute skewness
```

```
skewed_feats = skewed_feats[skewed_feats > 0.75]

skewed_feats = skewed_feats.index

all_data[skewed_feats] = np.log1p(all_data[skewed_feats])
```

houseprice= pd.get_dummies(houseprice)  #另外一种形式数据转换，将字符串特征列中的内容分别提出来作为新的特征出现，这样就省去了将字符串内容转化为数值特征内容的步骤了。

```
   Id  MSSubClass  LotFrontage  LotArea  Alley  MSZoning_RL  MSZoning_RM  \
0   1          60           65     8450      1            1            0
1   2          20           80     9600      2            1            0
2   3          60           68    11250      3            1            0
3   4          70           60     9550      4            0            1
4   5          60          NaN    14260      5            1            0
5   6          50          NaN    14115      8            0            0
6   7          20           75    10084      3            1            0
7   8         NaN          NaN    10382      2            0            0
8   9          50           51     6120      1            0            1
```

**6）数据标准化**

我们都知道大多数的梯度方法（几乎所有的机器学习算法都基于此）对于数据的缩放很敏感。因此，在运行算法之前，我们应该进行标准化，或所谓的规格化。标准化包括替换所有特征的名义值，让它们每一个的值在0和1之间。而对于规格化，它包括数据的预处理，使得每个特征的值有0和1的离差。Scikit-Learn库已经为其提供了相应的函数。

```
from sklearn import preprocessing

# normalize the data attributes

normalized_X = preprocessing.normalize(X)

# standardize the data attributes

standardized_X = preprocessing.scale(X)
```