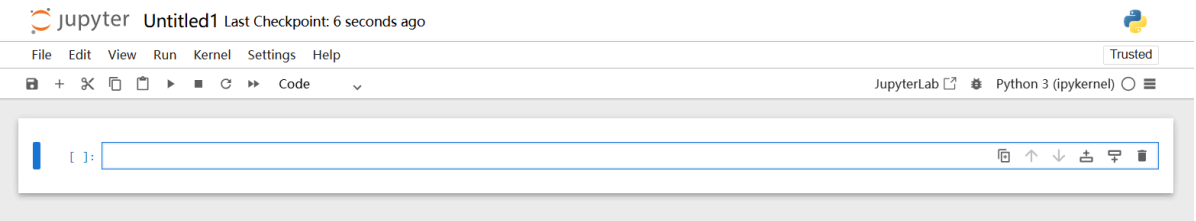
实验二：数据预处理的基本方法



打开 Jupyter Notebook，后续的 python 代码都是输入到蓝色框中，按 Shift + Enter 运行的

1. 缺失值的检测与缺失值处理

第一题有两问，我是分开做的，因为只有知道缺失值的分布情况，才好思考怎么处理

```
# 缺失值的检测
import os
import pandas as pd
df=pd.read_csv("train.csv")
r,c=df.shape
mc=df.isna().sum()
mr=(df.isna().mean()*100).round(2)
tot=int(mc.sum())
rat=round(tot/(r*c)*100,2)
with open("1a.txt","w",encoding="utf-8") as f:
    f.write(f"shape:{r} x {c}\n\n")
    f.write("per col count:\n")
    for k,v in mc.sort_values(ascending=False).items(): f.write(f"{k}:{v}\n")
    f.write("\nper col rate(%):\n")
    for k,v in mr.sort_values(ascending=False).items(): f.write(f"{k}:{v}\n")
    f.write(f"\ntotal:{tot}\n")
    f.write(f"total rate(%):{rat}\n")
print("done")
```

把 CSV文件 读成一个 DataFrame，变量名叫 df

其中 isna() 会得到一个和 df 同大小的布尔表，缺失为 True，sum() 是按列求和，mean() 是按列求平均值

这样，mc 和 mr 作为两个 Series，分别表示了每一列的缺失值个数、每一列的缺失值比例

再对 mc 求和，就得到了总的缺失值个数

输出的时候按照缺失值个数从大到小的顺序：

```
shape:1460 x 81

per col count:
PoolQC:1453
MiscFeature:1406
Alley:1369
Fence:1179
MasVnrType:872
FireplaceQu:690
LotFrontage:259
GarageQual:81
GarageFinish:81
GarageType:81
GarageYrBlt:81
GarageCond:81
BsmtFinType2:38
BsmtExposure:38
BsmtCond:37
BsmtQual:37
BsmtFinType1:37
MasVnrArea:8
Electrical:1
Condition2:0
BldgType:0
Neighborhood:0
LandSlope:0
LotConfig:0
Condition1:0
LandContour:0
LotShape:0
Street:0
LotArea:0
```

```

per col rate(%):
PoolQC:99.52
MiscFeature:96.3
Alley:93.77
Fence:80.75
MasVnrType:59.73
FireplaceQu:47.26
LotFrontage:17.74
GarageQual:5.55
GarageFinish:5.55
GarageType:5.55
GarageYrBlt:5.55
GarageCond:5.55
BsmtFinType2:2.6
BsmtExposure:2.6
BsmtCond:2.53
BsmtQual:2.53
BsmtFinType1:2.53
MasVnrArea:0.55
Electrical:0.07
Condition2:0.0
BldgType:0.0
Neighborhood:0.0
LandSlope:0.0
LotConfig:0.0
Condition1:0.0
LandContour:0.0
LotShape:0.0
Street:0.0
LotArea:0.0
MSSubClass:0.0
```

```
2ndFlrSF:0.0
1stFlrSF:0.0
CentralAir:0.0
HeatingQC:0.0
BsmtFinSF2:0.0
Fireplaces:0.0
TotRmsAbvGrd:0.0
KitchenQual:0.0
KitchenAbvGr:0.0
Functional:0.0
FullBath:0.0
HalfBath:0.0
GarageCars:0.0
GarageArea:0.0
BedroomAbvGr:0.0
PavedDrive:0.0
EnclosedPorch:0.0
3SsnPorch:0.0
OpenPorchSF:0.0
WoodDeckSF:0.0
PoolArea:0.0
ScreenPorch:0.0
MiscVal:0.0
MoSold:0.0
YrSold:0.0
SaleType:0.0
SaleCondition:0.0
SalePrice:0.0

total:7829
total rate(%):6.62

```

观察到缺失值的比例只有 6.62%，属于很小的部分

第二问：缺失值的处理

百度了一下，选取KNN 可以理解为在数据里找**距离最近的 k 个样本**，用邻居的信息来填补缺失值

对数值列可以取邻居的平均值，但是非数值列怎么取平均值？所以改成用众数

```python
# 缺失值处理(KNN)
import os,pandas as pd,numpy as np
from sklearn.impute import KNNImputer
df=pd.read_csv("train.csv")
num=df.select_dtypes(include=[np.number]).columns.tolist()
cat=df.select_dtypes(exclude=[np.number]).columns.tolist()
imp=KNNImputer(n_neighbors=5)
df[num]=imp.fit_transform(df[num])
for k in cat:
    m=df[k].mode(dropna=True)
    val=m.iloc[0] if len(m)>0 else "Missing"
    df[k]=df[k].fillna(val)
out="trainKNN.csv"
df.to_csv(out,index=False)
print("done")
```

num 表示所有数值列，cat 表示所有非数值列

建一个 KNN 插补器，设定邻居数为 5，对数值列做 KNN 插补

对于非数值列，逐个遍历这些列，如果能取到众数，就用众数填充，否则填字符串"Missing"

为了验证缺失值填充是否成功，我们可以对 trainKNN.csv 再跑一遍缺失值检测：

```
KitchenQual:0.0
TotRmsAbvGrd:0.0
Functional:0.0
Fireplaces:0.0
FireplaceQu:0.0
GarageType:0.0
GarageYrBlt:0.0
GarageFinish:0.0
GarageCars:0.0
GarageArea:0.0
GarageQual:0.0
GarageCond:0.0
PavedDrive:0.0
WoodDeckSF:0.0
OpenPorchSF:0.0
EnclosedPorch:0.0
3SsnPorch:0.0
ScreenPorch:0.0
PoolArea:0.0
PoolQC:0.0
Fence:0.0
MiscFeature:0.0
MiscVal:0.0
MoSold:0.0
YrSold:0.0
SaleType:0.0
SaleCondition:0.0
SalePrice:0.0

total:0
total rate(%):0.0

```

可见没有缺失值了，缺失值填充成功

2. 异常值检测

```python
# 异常值检测(IQR)
import pandas as pd
df=pd.read_csv("train.csv")
num=df.select_dtypes(include=[float,int]).columns.tolist()
with open("2a.txt","w",encoding="utf-8") as f:
    f.write("outlier detect\n")
    for k in num:
        q1=df[k].quantile(0.25); q3=df[k].quantile(0.75); i=q3-q1
        lo=q1-1.5*i; hi=q3+1.5*i
        n=((df[k]<lo)|(df[k]>hi)).sum()
        f.write(f"{k}:{n}\n")
print("done")
```

对于每一个数值列，q1 是第一四分位数，q3 是第三四分位数，lo = q1 - 1.5*(q3-q1)，hi = q3 + 1.5*(q3-q1)

系数 1.5 是箱线图的常用经验值，能较稳地标出异常值；数据波动更大时可调大，想更敏感就调小

统计出这一列中小于 lo 或者 大于 hi 的数值个数

```
outlier detect
Id:0
MSSubClass:103
LotFrontage:88
LotArea:69
OverallQual:2
OverallCond:125
YearBuilt:7
YearRemodAdd:0
MasVnrArea:96
BsmtFinSF1:7
BsmtFinSF2:167
BsmtUnfSF:29
TotalBsmtSF:61
1stFlrSF:20
2ndFlrSF:2
LowQualFinSF:26
GrLivArea:31
BsmtFullBath:1
BsmtHalfBath:82
FullBath:0
HalfBath:0
BedroomAbvGr:35
KitchenAbvGr:68
TotRmsAbvGrd:30
Fireplaces:5
GarageYrBlt:0
GarageCars:5
GarageArea:21
WoodDeckSF:32
OpenPorchSF:77
EnclosedPorch:208
```

为了符合实际问题，处理异常值的方式为选取平均值

```
# 异常值处理(均值替换)
import pandas as pd
df=pd.read_csv("train.csv")
num=df.select_dtypes(include=[float,int]).columns.tolist()
for k in num:
    s=df[k]
    q1=s.quantile(0.25); q3=s.quantile(0.75); i=q3-q1
    lo=q1-1.5*i; hi=q3+1.5*i
    m=s.mean()
    df[k]=s.mask((s<lo)|(s>hi),m)
df.to_csv("trainMean.csv",index=False)
print("done")
```

s.mask((s<lo)|(s>hi),m) 会把这一列中所有异常值替换为平均值，并返回替换后的这一列

3. 特征间的相关性分析

```
# 相关性矩阵+去冗余
import pandas as pd
df=pd.read_csv("train.csv")
cols=[k for k in df.columns if pd.api.types.is_numeric_dtype(df[k])]
cor=df[cols].corr().abs()
order=df[cols].var().sort_values(ascending=False).index.tolist()
keep=[]
for k in order:
    ok=True
    for j in keep:
        if cor.loc[k,j]>0.4: ok=False; break
    if ok: keep.append(k)
cor_keep=cor.loc[keep,keep].round(3)
```

```
cor_keep.to_csv("3a.csv")
print("done")
```

cols 只取数值型列名

cor 计算数值列与数值列的皮尔逊相关矩阵，并取绝对值

如何删去冗余特征?

可以按顺序考虑每个特征，如果当前特征与某个关键特征相关性大于 0.3，剔除掉

而这个顺序就是 order

order 按方差从大到小排序列名，方差大的列优先考虑

		SalePrice	LotArea	MiscVal	BsmtFinSF1	2ndFlrSF	Id	BsmtFinSF
1	SalePrice	1.0	0.264	0.021	0.386	0.319	0.022	0.01
2	LotArea	0.264	1.0	0.038	0.214	0.051	0.033	0.11
3	MiscVal	0.021	0.038	1.0	0.004	0.016	0.006	0.00
4	BsmtFinSF1	0.386	0.214	0.004	1.0	0.137	0.005	0.0
5	2ndFlrSF	0.319	0.051	0.016	0.137	1.0	0.006	0.09
6	Id	0.022	0.033	0.006	0.005	0.006	1.0	0.00
7	BsmtFinSF2	0.011	0.111	0.005	0.05	0.099	0.006	1
8	WoodDeckSF	0.324	0.172	0.01	0.204	0.092	0.03	0.06
9	OpenPorchSF	0.316	0.085	0.019	0.112	0.208	0.0	0.00
10	EnclosedPorch	0.129	0.018	0.018	0.102	0.062	0.003	0.03
11	ScreenPorch	0.111	0.043	0.032	0.062	0.041	0.001	0.08
12	LowQualFinSF	0.026	0.005	0.004	0.065	0.063	0.044	0.01
13	MSSubClass	0.084	0.14	0.008	0.07	0.308	0.011	0.06
14	PoolArea	0.092	0.078	0.03	0.14	0.081	0.057	0.04
15	3SsnPorch	0.045	0.02	0.0	0.026	0.024	0.047	0.0
16	MoSold	0.046	0.001	0.006	0.016	0.035	0.021	0.01
17	YrSold	0.029	0.014	0.005	0.014	0.029	0.001	0.03
18	OverallCond	0.078	0.006	0.069	0.046	0.029	0.013	0.0
19	BsmtHalfBath	0.017	0.048	0.007	0.067	0.024	0.02	0.07
20	KitchenAbvGr	0.136	0.018	0.062	0.081	0.059	0.003	0.04

最终保留了 20 个关键特征

4. 对price属性进行标准化

```
# price标准化(对数+Z)
import pandas as pd,numpy as np
df=pd.read_csv("train.csv")
col="price" if "price" in df.columns else ("SalePrice" if "SalePrice" in
df.columns else None)
if col is None: raise SystemExit("no price")
s=df[col].astype(float)
x=np.log1p(s)
z=(x-x.mean())/x.std(ddof=0)
df[col]=z.round(3)
df.to_csv("trainPriceStd.csv",index=False)
print("done")
```

合理的标准化:

① 对价格做对数变换，以减小右偏、稳定方差

② 对变换后的值做 Z 标准化，做完后：均值 0，标准差 1

s=df[col].astype(float)：取价格列并转成浮点型，便于后续数值计算

x=np.log1p(s)：对价格做对数变换

z=(x-x.mean())/x.std(ddof=0)：对变换后的值做 Z 标准化

df[col]=z.round(3)：为了方便观察，四舍五入保留3位小数

	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
1440			0	11	2007	WD	Normal	0.418
1441			0	9	2008	WD	Normal	0.341
1442			0	5	2008	WD	Normal	-0.276
1443			0	4	2009	WD	Normal	1.553
1444			0	5	2009	WD	Normal	-0.803
1445			0	11	2007	WD	Normal	0.186
1446			0	5	2007	WD	Normal	-0.642
1447			0	4	2010	WD	Normal	-0.136
1448			0	12	2007	WD	Normal	0.912
1449	GdWo		0	5	2007	WD	Normal	-0.996
1450			0	8	2006	WD	Abnorml	-1.489
1451			0	9	2009	WD	Normal	-0.51
1452			0	5	2009	New	Partial	1.361
1453			0	5	2006	WD	Normal	-0.35
1454			0	7	2006	WD	Abnorml	-1.702
1455			0	10	2009	WD	Normal	0.261
1456			0	8	2007	WD	Normal	0.121
1457	MnPrv		0	2	2010	WD	Normal	0.578
1458	GdPrv	Shed	2500	5	2010	WD	Normal	1.175
1459			0	4	2010	WD	Normal	-0.4
1460			0	6	2008	WD	Normal	-0.307

5. 根据price属性进行离散化

```
# price离散化(四分位)
import pandas as pd
df=pd.read_csv("train.csv")
col="price" if "price" in df.columns else ("SalePrice" if "SalePrice" in
df.columns else None)
if col is None: raise SystemExit("no price")
s=df[col].astype(float)
b=pd.qcut(s,4,labels=[1,2,3,4],duplicates="drop")
df[col+"Bin"]=b.astype(int)
df.to_csv("trainPriceBin.csv",index=False)
print("done")
```

b=pd.qcut(s,4,labels=[1,2,3,4],duplicates="drop")：用**分位数分箱**把所有价格切成 4 个等频区间

指定每个箱子的标签为 1~4（从低到高），如果价格取值太少导致某些箱边界重复，自动合并重复区间，避免报错

df[col+"Bin"]=b.astype(int)：在表格右侧新增一列，表示离散化后的价格

6. 找出与price相关性最高的三个特征，并给出合理的解释

```
# 与price最相关的三个特征
import pandas as pd,numpy as np
df=pd.read_csv("train.csv")
col="price" if "price" in df.columns else ("SalePrice" if "SalePrice" in
df.columns else None)
if col is None: raise SystemExit("no price")
y=df[col].astype(float)
num=[k for k in df.columns if pd.api.types.is_numeric_dtype(df[k]) and k!=col]
cat=[k for k in df.columns if not pd.api.types.is_numeric_dtype(df[k])]
s={}
if num:
    v=df[num+[col]].corr()[col].abs().drop(col)
    for k in v.index: s[k]=v.loc[k]
for k in cat:
```

```
    d=pd.get_dummies(df[k])
    m=0.0
    for j in d.columns:
        t=pd.concat([d[j],y],axis=1).corr().iloc[0,1]
        if pd.notna(t): m=max(m,abs(t))
    s[k]=m
top=sorted(s.items(),key=lambda x:x[1],reverse=True)[:3]
for k,v in top: print(k,round(v,3))
```

num 表示所有数值列，**除了价格列**

cat 表示所有非数值列

对于数值列，可以借用第三题的皮尔逊相关矩阵

v=df[num+[col]].corr()[col].abs().drop(col)：

① 取出所有数值列，再加上price列，计算皮尔逊相关矩阵

② [col] 选取与price的那一列，abs取绝对值，drop(col)去掉price这一行

for k in v.index: s[k]=v.loc[k]：把序列 v 里的每个特征名及其相关系数写入字典

对于非数值列，对每个非数值列先做 one-hot，再把每列与 price 计算皮尔逊相关系数，取绝对值最大的那个作为该字段的相关强度

最后与数值列的相关度一起排序选出前三

```
[20]: # 与price最相关的三个特征
import pandas as pd,numpy as np
df=pd.read_csv("train.csv")
col="price" if "price" in df.columns else ("SalePrice" if "SalePrice" in df.columns else None)
if col is None: raise SystemExit("no price")
y=df[col].astype(float)
num=[k for k in df.columns if pd.api.types.is_numeric_dtype(df[k]) and k!=col]
cat=[k for k in df.columns if not pd.api.types.is_numeric_dtype(df[k])]
s={}
if num:
    v=df[num+[col]].corr()[col].abs().drop(col)
    for k in v.index: s[k]=v.loc[k]
for k in cat:
    d=pd.get_dummies(df[k])
    m=0.0
    for j in d.columns:
        t=pd.concat([d[j],y],axis=1).corr().iloc[0,1]
        if pd.notna(t): m=max(m,abs(t))
    s[k]=m
top=sorted(s.items(),key=lambda x:x[1],reverse=True)[:3]
for k,v in top: print(k,round(v,3))

OverallQual 0.791
GrLivArea 0.709
GarageCars 0.64

[ ]:
```

最终的前三名：

OverallQual: Rates the overall material and finish of the house

GrLivArea: Above grade (ground) living area square feet

GarageCars: Size of garage in car capacity

合理的解释：

OverallQual（整体做工与用材等级）：

高品质材料与更精细的施工直接提升耐久性、舒适度与美观度，买家愿意支付溢价

GrLivArea（地上居住面积，平方英尺）：

居住面积是最直观的“可用空间”，直接决定功能性与舒适度

GarageCars（车库可停车辆数）：

更大的车库提供额外收纳与车辆停放便利，车位是很多家庭的刚需