0. 数据加载与查看

0.1 数据加载

```
df = pd.read_csv('./train.csv')
df = pd.read_excel('filename.xlsx', sheetname='sn')
```

0.2 数据合并

```
df = pd.concat([df1, df2])
```

0.3 粗略查看

```
df.head(10)
df.tail(10)
df.columns    #查看列名
df.info()     #查看各字段的信息
df.shape      #查看数据集行列分布
df.describe() #查看数据的大体情况，给出数值型列的统计结果：均值、方差、最值
```

0.4 数据统计

对数据做简单的统计。

```
df['feature'].value_counts()    #某列各元素值出现的次数
df['feature'].skew()     #偏斜度
df['feature'].kurt()     #峰度
df['feature1'].corr(df['feature2'])  #两列的相关度

#查看两个变量值的相关性
x = 'LotArea'
y = 'SalePrice'
data = pd.concat([df[y], df[x]], axis = 1)
data.plot.scatter(x = x, y = y, ylim = (0, df[y].max()+100000))

#相关性热力图
import matplotlib.pyplot as plt
import seaborn as sns
corrmat = df.corr()
fig, ax = plt.subplots(figsize = (12, 9))
sns.heatmap(corrmat, square = True)

#找出前k个相关性最高的列
k = 10
cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
cm = np.corrcoef(df[cols].values.T)
sns.set(font_scale = 1.25)
hm = sns.heatmap(cm, cbar = True, annot = True, square = True,
yticklabels=cols.values, xticklabels=cols.values)
plt.show()
```

1. 离群点

1.1 散点图直观查看特征变量与y值的关系

```
fig, ax = plt.subplots()
ax.scatter(x = train['feature1'], y = train['y'])
plt.show()
```

- 作图后根据散点图查看变量与y值之间的关系，如果存在明显的相关关系（比如，线性关系），对于明显不符合该关系的样本点进行剔除。 `drop()`

2. 缺失值

2.1 缺失值情况查看

```
total = train.isnull().sum().sort_values(ascending=False)
all_data_na = (all_data.isnull().sum/len(all_data))*100
all_data_na =
all_data_na.drop(all_data_na[all_data_na==0].index).sort_values(ascending=False)
```

2.2 缺失值处理

- 对于缺失率过高的变量，可以直接剔除
- 用某些值填充

```
data = data.drop(data[data['feature'].isnull()].index)  #剔除缺失值
data['feature'] = data['feature'].fillna("None")
data['feature'] = data['feature'].fillna(0)
data['feature'] = data['feature'].fillna(data['feature'].mode()[0]) #众数
data['feature'] = data['feature'].fillna(data['feature'].median())  #中位数
data['feature'] = data['feature'].fillna(data['feature'].mean())  #平均数
```

3. 相关性

```
#Correlation map to see how features are correlated
import seaborn as sns

corrmat = train.corr()
plt.subplots(figsize = (12,9))
sns.heatmap(corrmat, vmax = 0.9, square = True)
```

4. 变量分布（偏度）

4.1 查看变量偏度

```
numeric_feats = all_data.dtypes[all_data.dtypes != "object"].index
skewed_feats = all_data[numeric_feats].apply(lambda x:
skew(x.dropna())).sort_values(ascending=False)
skewness = pd.DataFrame({'Skew' :skewed_feats})
```

4.2 偏度过大变量进行变换

```
# Box Cox Transformation of (highly) skewed features
skewness = skewness[abs(skewness) > 0.75]
print("There are {} skewed numerical features to Box Cox
transform".format(skewness.shape[0]))
```

```
from scipy.special import boxcox1p
skewed_features = skewness.index

lam = 0.15
for feat in skewed_features:
    all_data[feat] = boxcox1p(all_data[feat], lam)
```

并不是所有的偏离值都需要删除，具体需要在分析之后选择处理方式。这里将偏离值保留下来并不是原封不动保留，而需要做标准化或归一化处理，具体的处理方式可查看最后一节数据转换、标准化、归一化。

5. 数据缩放

5.1 归一化(Rescaling)

将数据缩放到一个指定的最大值和最小值之间(通常为0-1之间)

- 对于方差非常小的属性可以增强其稳定性
- 维持稀疏矩阵中为0的条目

```
min_max_scaler = preprocessing.MinMaxScaler()
X_train_minmax = min_max_scaler.fit_transform(X_train)
# 同样的缩放应用于测试集
X_test_minmax = min_max_scaler.transform(X_test)
```

5.2 标准化（Standardization）

> 该操作会丢失信息，使用时需要判断丢失的信息是否会对之后的操作造成负面影响。

- 对每一列进行标准化
- z-score方法：$(x - mean(x))/std(x)$

```
X_scaled = preprocessing.scale(X)
```

```
# StandardScaler可保存训练集中的参数（均值、标准差），并直接使用其对象转换测试集
scaler = preprocessing.StandardScaler().fit(X_train)
scaler.transform(X_train)
#对测试集转换
scaler.transform(X_test)
```

5.3 正则化(Normalization)

将**每个样本**缩放到单位范数（每个样本的范数为1），对每个样本计算其p-范数，然后对该样本中每个元素除以该范数，这样处理的结果是使得每个处理后样本的p-范数（l1-norm,l2-norm）等于1。

- 该方法主要应用于文本分类和聚类中。例如，对于两个TF-IDF向量的l2-norm进行点积，就可以得到这两个向量的余弦相似性。

p-范数的计算公式：||X||p=(|x1|^p +|x2|^p +...+|xn|^p)^1/p

- 利于使用二次型或其他和方法计算两个样本之间的相似性

1. 使用preprocessing.normalize()函数对指定数据进行转换：

```
X_normalized = preprocessing.normalize(X,norm='l2')
```

2. 使用processing.Normalizer()类实现对训练集和测试集的拟合和转换：

```
normalizer = preprocessing.Normalizer().fit(X) #fit does nothing

normalizer.transform(X_train)
normalizer.transform(X_test)
```

6. 非线性变换

6.1 Mapping to a Uniform distribution

```
quantile_transformer = preprocessing.QuantileTransformer(random_state=0)
X_train_trans = quantile_transformer.fit_transform(X_train)
X_test_trans = quantile_transformer.transform(X_test)
```

6.2 Mapping to a Gaussian distribution

```
#三种变换高斯分布的方法
pt = preprocessing.PowerTransformer(method='box-cox', standardize=False)
pt2 = preprocessing.PowerTransformer(method='yeo-johnson', standardize = False)

qt = preprocessing.QuantileTransformer(output_distribution = 'normal',
random_state = 0)
```

7. 类别特征编码

7.1 一般编码

不改变特征维度，适用于同一特征下不同类别的值之间有距离、大小关系的情况，比如年龄、身高。

```
X = [['male', 'from US', 'uses Safari'], ['female', 'from Europe', 'uses
Firefox']]
enc = preprocessing.OrdinalEncoder().fit(X)
X_encoded = enc.transform(X)
```

7.2 OneHot

使特征维度扩展，适用于同一特征下不同类别的值之间没有距离、大小关系的情况，比如材质、名称等。

```
enc = preprocessing.OneHotEncoder().fit(X)
X_encoded2 = enc.transform(X)
X_encoded2.toarray()
```

8. 多项式特征

Often it's useful to add complexity to the model by considering nonlinear features of the input data. A simple and common method to use is polynomial features, which can **get features' high-order and interaction terms**.

```
from sklearn.preprocessing import PolynomialFeatures
X = np.arange(6).reshape(3,2)
poly = PolynomialFeatures(2)
poly.fit_transform(X)
#The features of X have been transformed from ($X_1,X_2,X_3$) to
($1,X_1,X_2,X_3,X_1X_2,X_1X_3,X_2X_3,X_1X_2X_3$).
```

9. 特征工程

9.1 特征合并

从几个特征值里挖掘一些值来当作新的特征值。

```
df_grouped = df.groupby(['feature1', 'feature2'], as_index=False)
df_grouped.mean()
```

10. 其他操作

10.1 类别分组

```
all_data = pd.get_dummies(all_data)
print(all_data.shape)
```

10.2 保存多个sheet表

```
writer = pd.ExcelWriter('E:\\PythonTestCode\\public opinion_result.xlsx')
data1.to_excel(writer, sheet_name = 'data1', index = False)
data2.to_excel(writer, sheet_name = 'data2', index = False)
writer.save()
writer.close()
```