1. 数据预处理

1.1 异常值处理

- 箱线法：适用于大部分数据
- 拉以达准则：假设一组检测数据只含有随机误差，对其进行计算处理得到标准偏差，按一定概率确定一个区间，认为凡超过这个区间的误差，就不属于随机误差而是粗大误差，含有该误差的数据应予以剔除

  适用于数据量大，服从或近似服从正态分布的数据。

  误差范围可取3σ（3σ原则）

1.2 数据无量纲化（归一化or标准化）

线性的无量纲化包括**中心化**（Zero-centered或者Mean-subtraction）处理和**缩放处理**（Scale）。中心化的本质是让所有记录减去一个固定值，即让数据样本数据平移到某个位置。缩放的本质是通过除以一个固定值，将数据固定在某个范围之中，取对数也算是一种缩放处理

- **数据归一化（python库：preprocessing.MinMaxScaler）**

  当数据(x)按照最小值中心化后，再按极差（最大值 - 最小值）缩放，数据移动了最小值个单位，并且会被收敛到[0,1]之间，而这个过程，就叫做**数据归一化**(Normalization，又称Min-Max Scaling)。归一化后的数据服从正态分布

  $$x^* = \frac{x - min(x)}{max(x) - min(x)}$$

  - feature_range: 控制我们希望把数据压缩到的范围，默认是[0, 1]
  - 当x中的特征数量非常多是，fit会报错，并表示数据量太大我计算不了。此时应使用partial_fit作为训练接口
  - 数据归一化另一种实现（BONUS：使用numpy来实现）

  ```
  X_nor = (X - X.min(axis=0)) / (X.max(axis=0)) - X.min(axis=0)
  ```

- **数据标准化（python库：preprocessing.StandardScaler）**

  当数据(x)按均值(μ)中心化后，再按标准差(σ)缩放，数据就会服从为均值为0，方差为1的正态分布（即标准正态分布），而这个过程，就叫做**数据标准化**(Standardization，又称Z-score normalization)

  $$x^* = \frac{x - \mu}{\sigma}$$

```
# 标准化
for i in range(X.shape[-1]):
    sigma_ = pow(( sum( (X[:, i]-np.mean(X[:,i]))**2 ) / (X.shape[0]-1) ),
0.5)
    X[:, i] = (X[:, i]-np.mean(X[:,i])) / sigma_

# 或
from scipy.stats import zscore
X = zscore(X, axis=0)
```

    - 或者直接使用SPSS（分析-描述统计-描述：在描述列表的方框左下角，看到“将标准化得分另存为变量（Z）之后点击打勾，然后确定。）

- 看情况。大多数机器学习算法中，会选择StandardScaler来进行特征缩放，因为MinMaxScaler对异常值非常敏感。在PCA，聚类，逻辑回归，支持向量机，神经网络这些算法中，StandardScaler往往是最好的选择。

  MinMaxScaler在不涉及距离度量、梯度、协方差计算以及数据需要被压缩到特定区间时使用广泛，比如数字图像处理中量化像素强度时，都会使用MinMaxScaler将数据压缩于[0,1]区间之中。

  建议先试试看StandardScaler，效果不好换MinMaxScaler。

1.3 缺失值

其实有两种可能，第一种是缺失某个值，另一种是缺失某条记录，但大多数时候都是指前者

1.3.1 python库：impute.SimpleImputer

> - **impute.SimpleImputer**
>
> *class* `sklearn.impute.SimpleImputer` (*missing_values=nan, strategy='mean', fill_value=None, verbose=0, copy=True*)
>
> 在讲解随机森林的案例时，我们用这个类和随机森林回归填补了缺失值，对比了不同的缺失值填补方式对数据的影响。这个类是专门用来填补缺失值的。它包括四个重要参数：

参数	含义&输入
missing_values	告诉SimpleImputer，数据中的缺失值长什么样，默认空值np.nan
strategy	我们填补缺失值的策略，默认均值。 输入“mean”使用均值填补（仅对数值型特征可用） 输入“median"用中值填补（仅对数值型特征可用） 输入"most_frequent”用众数填补（对数值型和字符型特征都可用） 输入“constant"表示请参考参数“fill_value"中的值（对数值型和字符型特征都可用）
fill_value	当参数startegy为”constant"的时候可用，可输入字符串或数字表示要填充的值，常用0
copy	默认为True，将创建特征矩阵的副本，反之则会将缺失值填补到原本的特征矩阵中去。

BONUS：用Pandas和Numpy进行填补其实更加简单

```
#判断缺失值
df[df["名字"].isnull()]
# 填补缺失值
data_.loc[:,"Age"] =
data_.loc[:,"Age"].fillna(data_.loc[:,"Age"].median())#.fillna 在DataFrame里面直接进行填补
# 删除缺失值
data_.dropna(axis=0,inplace=True)
```

其它填补缺失值的方法

- 随机森林
- knn
- 多重插补
- 三次样条插值

1.4 分类型特征

1.4.1 preprocessing.LabelEncoder：标签专用，能够将分类转换为分类数值

```
from sklearn.preprocessing import LabelEncoder
data.iloc[:,-1] = LabelEncoder().fit_transform(data.iloc[:,-1])
```

1.4.2 preprocessing.OrdinalEncoder：特征专用，能够将分类特征转换为分类数值

接口categories对应*LabelEncoder*的接口*classes*，一模一样的功能

1.4.3 preprocessing.OneHotEncoder：独热编码，创建哑变量

对于一些变量不能简单用OrdinalEncoder这一类，有些变量是“有你就没有我”的不等概念。如性别、舱门等

编码与哑变量	功能	重要参数	重要属性	重要接口
.LabelEncoder	分类标签编码	N/A	.classes_：查看标签中究竟有多少类别	fit, transform, fit_transform, inverse_transform
.OrdinalEncoder	分类特征编码	N/A	.categories_：查看特征中究竟有多少类别	fit, transform, fit_transform, inverse_transform
.OneHotEncoder	独热编码，为名义变量创建哑变量	**categories**: 每个特征都有哪些类别，默认"auto"表示让算法自己判断，或者可以输入列表，每个元素都是一个列表，表示每个特征中的不同类别 **handle_unknown**：当输入了categories，且算法遇见了categories中没有写明的特征或类别时，是否报错。默认"error"表示请报错，也可以选择"ignore"表示请无视。如果选择"ignore"，则未再categories中注明的特征或类别的哑变量会全部显示为0。在逆转(inverse transform)中，未知特征或类别会被返回为None。	.categories_：查看特征中究竟有多少类别，如果是自己输入的类别，那就不需要查看了	fit, transform, fit_transform, inverse_transform, get_feature_names：查看生成的哑变量的每一列都是什么特征的什么取值

1.5 处理连续性特征（二值化与分段）

1.5.1 sklearn.preprocessing.Binarizer（二值化）

根据阈值将数据二值化（将特征值设置为0或1），用于处理连续型变量。大于阈值的值映射为1，而小于或等于阈值的值映射为0。默认阈值为0时，特征中所有的正值都映射到1

1.5.2 preprocessing.KBinsDiscretizer（分箱）

将连续型变量划分为分类变量的类，能够将连续型变量排序后按顺序分箱后编码。

参数	含义&输入
n_bins	每个特征中分箱的个数，默认5，一次会被运用到所有导入的特征
encode	编码的方式，默认“onehot” "onehot"：做哑变量，之后返回一个稀疏矩阵，每一列是一个特征中的一个类别，含有该类别的样本表示为1，不含的表示为0 “ordinal”：每个特征的每个箱都被编码为一个整数，返回每一列是一个特征，每个特征下含有不同整数编码的箱的矩阵 "onehot-dense"：做哑变量，之后返回一个密集数组。
strategy	用来定义箱宽的方式，默认"quantile" "uniform"：表示等宽分箱，即每个特征中的每个箱的最大值之间的差为(特征.max() - 特征.min())/(n_bins) "quantile"：表示等位分箱，即每个特征中的每个箱内的样本数量都相同 "kmeans"：表示按聚类分箱，每个箱中的值到最近的一维k均值聚类的簇心得距离都相同

2. 特征工程

当数据预处理完成后，我们就要开始进行特征工程了

- 特征提取
- 特征创造
- 特征选择

> 这里主要讲特征选择

有四种方法可以用来选择特征：过滤法，嵌入法，包装法，和降维算法。

2.1 过滤法

过滤方法通常用作预处理步骤，特征选择完全独立于任何机器学习算法。它是根据各种统计检验中的分数以及相关性的各项指标来**选择特征。**

类	说明	超参数的选择
VarianceThreshold	方差过滤，可输入方差阈值，返回方差大于阈值的新特征矩阵	看具体数据究竟是含有更多噪声还是更多有效特征 一般就使用0或1来筛选 也可以画学习曲线或取中位数跑模型来帮助确认
SelectKBest	用来选取K个统计量结果最佳的特征，生成符合统计量要求的新特征矩阵	看配合使用的统计量
chi2	卡方检验，专用于分类算法，捕捉相关性	追求p小于显著性水平的特征
f_classif	F检验分类，只能捕捉线性相关性 要求数据服从正态分布	追求p小于显著性水平的特征
f_regression	F检验回归，只能捕捉线性相关性 要求数据服从正态分布	追求p小于显著性水平的特征
mutual_info_classif	互信息分类，可以捕捉任何相关性 不能用于稀疏矩阵	追求互信息估计大于0的特征
mutual_info_regression	互信息回归，可以捕捉任何相关性 不能用于稀疏矩阵	追求互信息估计大于0的特征

2.1.1 方差过滤（sklearn.feature_selection.VarianceThreshold）

VarianceThreshold有重要参数**threshold**，表示方差的阈值，表示舍弃所有方差小于threshold的特征，不填默认为0，即删除所有的记录都相同的特征。

> 当特征是二分类时，特征的取值就是伯努利随机变量，这些变量的方差可以计算为：
>
> $$Var[X] = p(1-p)$$
>
> 其中X是特征矩阵，p是二分类特征中的一类在这个特征中所占的概率。
>
> ```
> #若特征是伯努利随机变量，假设p=0.8，即二分类特征中某种分类占到80%以上的时候删除特征
> X_bvar = VarianceThreshold(.8 * (1 - .8)).fit_transform(X)
> X_bvar.shape
> ```

减少特征对KNN算法很有用，而对随机森林效果不大：这其实很容易理解，无论过滤法如何降低特征的数量，随机森林也只会选取固定数量的特征来建模；而最近邻算法就不同了，特征越少，距离计算的维度就越少，模型明显会随着特征的减少变得轻量。因此，过滤法的**主要对象**是：**需要遍历特征或升维的算法们**，而过滤法的**主要目的**是：**在维持算法表现的前提下，帮助算法们降低计算成本**

这里的方差阈值，其实相当于是一个超参数，要选定最优的超参数，我们可以画学习曲线，找模型效果最好的点。但现实中，我们往往不会这样去做，因为这样会耗费大量的时间。我们**只会使用阈值为0或者阈值很小的方差过滤，**来为我们优先消除一些明显用不到的特征，然后我们会选择**更优的特征选择方法**继续削减特征数量

2.1.2 相关性过滤

我们希望选出与标签相关且有意义的特征，因为这样的特征能够为我们提供大量信息。

- χ^2过滤

  卡方过滤是专门针对离散型标签（即分类问题）的相关性过滤。卡方检验类**feature_selection.chi2**计算每个非负特征和标签之间的卡方统计量，并依照卡方统计量由高到低为特征排名

- F过滤

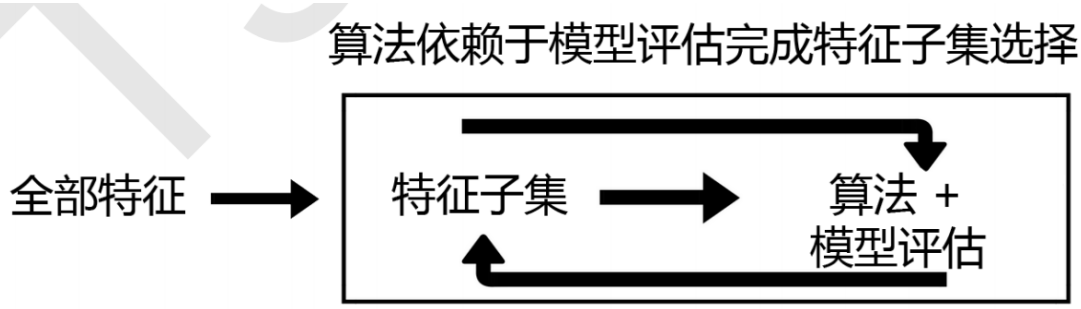
  F检验，又称ANOVA，方差齐性检验，是用来捕捉每个特征与标签之间的线性关系的过滤方法。它即**可以做回归也可以做分类**，因此包含**feature_selection.f_classif**（F检验分类）和**feature_selection.f_regression**（F检验回归）两个类。其中F检验分类用于标签是离散型变量的数据，而F检验回归用于标签是连续型变量的数据

- 互信息法

  互信息法是用来捕捉每个特征与标签之间的任意关系（包括线性和非线性关系）的过滤方法。和F检验相似，它既可以做回归也可以做分类，并且包含两个类**feature_selection.mutual_info_classif**（互信息分类）和**feature_selection.mutual_info_regression**（互信息回归）。这两个类的用法和参数都和F检验一模一样，不过互信息法比F检验更加强大，F检验只能够找出线性关系，而互信息法可以找出任意关系。

  互信息法不返回p值或F值类似的统计量，它返回“每个特征与目标之间的互信息量的估计”，这个估计量在[0,1]之间取值，**为0则表示两个变量独立，为1则表示两个变量完全相关。**所有特征的互信息量估计都大于0，则所有特征都与标签相关

5.2 嵌入法



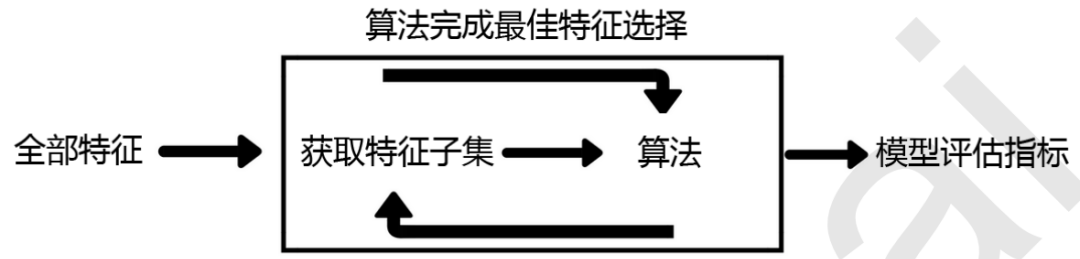
SelectFromModel是一个元变换器，可以与任何在拟合后具有coef，*feature_importances*属性或参数中可选惩罚项的评估器一起使用（比如随机森林和树模型就具有属性feature_importances_，逻辑回归就带有l1和l2惩罚项，线性支持向量机也支持l2惩罚项）。

需要自己输入阈值

参数	说明
estimator	使用的模型评估器，只要是带feature_importances_或者coef_属性，或带有l1和l2惩罚项的模型都可以使用
threshold	特征重要性的阈值，重要性低于这个阈值的特征都将被删除
prefit	默认False，判断是否将实例化后的模型直接传递给构造函数。如果为True，则必须直接调用fit和transform，不能使用fit_transform，并且SelectFromModel不能与cross_val_score，GridSearchCV和克隆估计器的类似实用程序一起使用。
norm_order	k可输入非零整数，正无穷，负无穷，默认值为1 在评估器的coef_属性高于一维的情况下，用于过滤低于阈值的系数的向量的范数的阶数。
max_features	在阈值设定下，要选择的最大特征数。要禁用阈值并仅根据max_features选择，请设置threshold = -np.inf

tip：因此，比起要思考很多统计量的过滤法来说，嵌入法可能是更有效的一种方法。然而，在算法本身很复杂的时候，过滤法的计算远远比嵌入法要快，所以大型数据中，我们还是会优先考虑过滤法。

5.3 包装法



注意，在这个图中的“算法”，指的不是我们最终用来导入数据的分类或回归算法（即不是随机森林），而是专业的数据挖掘算法，即我们的目标函数。这些数据挖掘算法的核心功能就是选取最佳特征子集。

最典型的目标函数是递归特征消除法（Recursive feature elimination, 简写为RFE）。它是一种贪婪的优化算法，

- **feature_selection.RFE**

class `sklearn.feature_selection.RFE` (*estimator*, *n_features_to_select=None*, *step=1*, *verbose=0*)

参数**estimator**是需要填写的实例化后的评估器，**n_features_to_select**是想要选择的特征个数，**step**表示每次迭代中希望移除的特征个数。除此之外，RFE类有两个很重要的属性，**.support_**：返回所有的特征的是否最后被选中的布尔矩阵，以及**.ranking_**返回特征的按数次迭代中综合重要性的排名。类feature_selection.RFECV会在交叉验证循环中执行RFE以找到最佳数量的特征，增加参数cv，其他用法都和RFE一模一样。

包装法的计算成本是最高的

包装法的效果是所有特征选择方法中最利于提升模型表现的，它可以使用很少的特征达到很优秀的效果。除此之外，在特征数目相同时，包装法和嵌入法的效果能够匹敌，不过它比嵌入法算得更见缓慢，所以也不适用于太大型的数据。相比之下，包装法是最能保证模型效果的特征选择方法。

5.4 特征选择方法的总结

过滤法更快速，但更粗糙。包装法和嵌入法更精确，比较适合具体到算法去调整，但计算量比较大，运行时间长。当数据量很大的时候，优先使用方差过滤和互信息法调整，再上其他特征选择方法。使用逻辑回归时，优先使用嵌入法。使用支持向量机时，优先使用包装法。迷茫的时候，从过滤法走起，看具体数据具体分析

注意：以上方法与PCA和SVD算法的区别：

- 以上方法仍属于特征选择的范围。
- 而PAC(或SVD)算法将已存在的特征进行压缩，降维完毕后的特征不是原本的特征矩阵中的任何一个特征，而是通过某些方式组合起来的新特征
- 故PAC(或SVD)算法应该属于特征创造
- PCA**一般不适用于**探索特征和标签之间的关系的模型（如**线性回归**），因为无法解释的新特征和标签之间的关系不具有意义。在线性回归模型中，我们使用**特征选择**