1.特征选择

特征工程是数据分析中最耗时间和精力的一部分工作，它不像算法和模型那样是确定的步骤，更多是工程上的经验和权衡。因此没有统一的方法。这里只是对一些常用的方法做一个总结。

1. 1 特征的来源

在做数据分析的时候，特征的来源一般有两块：

- 一块是业务已经整理好各种特征数据，我们需要去找出**我们问题需要**的特征；
- 另一块是我们从业务特征中自己去寻找高级数据特征。

1.2 选择合适的特征

我们首先看当业务已经整理好各种特征数据时，我们如何去找出适合我们问题需要的特征，此时特征数可能成百上千，哪些才是我们需要的呢?

第一步是找到**该领域懂业务的专家**，让他们给一些建议。比如我们需要解决一个药品疗效的分类问题，那么先找到领域专家，向他们咨询哪些因素（特征）会对该药品的疗效产生影响，较大影响的和较小影响的都要。这些特征就是我们的特征的第一候选集。

这个特征集合有时候也可能很大，在尝试降维之前，我们有必要用特征工程的方法去选择出较重要的特征结合，这些方法不会用到领域知识，而仅仅是统计学的方法。

最简单的方法就是**方差筛选**。方差越大的特征，那么我们可以认为它是比较有用的。如果方差较小，比如小于1，那么这个特征可能对我们的算法作用没有那么大。最极端的，如果某个特征方差为0，即所有的样本该特征的取值都是一样的，那么它对我们的模型训练没有任何作用，可以直接舍弃。**在实际应用中，我们会指定一个方差的阈值，当方差小于这个阈值的特征会被我们筛掉。** sklearn中的VarianceThreshold类可以很方便的完成这个工作。

特征选择方法有很多，一般分为三类：

- 第一类**过滤法**比较简单，它按照特征的发散性或者相关性指标对各个特征进行评分，设定评分阈值或者待选择阈值的个数，选择合适特征。上面我们提到的方差筛选就是过滤法的一种。
- 第二类是**包装法**，根据目标函数，通常是预测效果评分，每次选择部分特征，或者排除部分特征。
- 第三类**嵌入法**则稍微复杂一点，它先使用某些机器学习的算法和模型进行训练，得到各个特征的权值系数，根据权值系数从大到小来选择特征。类似于过滤法，但是它是通过机器学习训练来确定特征的优劣，而不是直接从特征的一些统计学指标来确定特征的优劣。下面我们分别来看看3类方法。

1.2.1 过滤法选择特征（方差、相关系数、假设检验）

- 上面我们已经讲到了使用特征方差来过滤选择特征的过程。除了特征的**方差**这第一种方法，还有其他一些统计学指标可以使用。
- 第二个可以使用的是**相关系数**。这个主要用于输出连续值的监督学习算法中。我们分别计算所有训练集中各个特征与输出值之间的相关系数，设定一个阈值，选择相关系数较大的部分特征。

- 第三个可以使用的是**假设检验**，比如卡方检验。卡方检验可以检验**某个特征分布和输出值分布之间的相关性**。个人觉得它比比粗暴的方差法好用。在sklearn中，可以使用chi2这个类来做卡方检验得到所有特征的卡方值与显著性水平P临界值，我们可以给定卡方值阈值， 选择**卡方值较大的部分特征**。

  除了卡方检验，我们还可以使用**F检验和t检验**，它们都是使用假设检验的方法，只是使用的统计分布不是卡方分布，而是F分布和t分布而已。在sklearn中，有F检验的函数f_classif和f_regression，分别在分类和回归特征选择时使用。

- 第四个是**互信息**，即从**信息熵的角度分析各个特征和输出值之间的关系评分**。在决策树算法中我们讲到过互信息（信息增益）。互信息值越大，说明该特征和输出值之间的相关性越大，越需要保留。在sklearn中，可以使用mutual_info_classif(分类)和mutual_info_regression(回归)来计算各个输入特征和输出值之间的互信息。

以上就是过滤法的主要方法，个人经验是，在没有什么思路的时候，可以优先使用卡方检验和互信息来做特征选择

1.2.2 包装法选择特征

包装法的解决思路没有过滤法这么直接，它会选择一个目标函数来一步步的筛选特征。

最常用的包装法是**递归消除特征法(recursive feature elimination,以下简称RFE)**。递归消除特征法使用一个机器学习模型来进行多轮训练，每轮训练后，消除若干权值系数的对应的特征，再基于新的特征集进行下一轮训练。在sklearn中，可以使用RFE函数来选择特征。

我们下面以经典的 **SVM-RFE 算法**来讨论这个特征选择的思路。这个算法以支持向量机来做RFE的机器学习模型选择特征。它**在第一轮训练的时候，会选择所有的特征来训练，得到了分类的超平面 $wx^{\cdot}+b$=0 后，如果有n个特征，那么 RFE-SVM 会选择出 w 中分量的平方值 $$w^2i$$ *最小的那个序号 $$i$$ 对应的特征，将其排除，在第二轮的时候，特征数就剩下n-1个了，我们继续用这n-1个特征和输出值来训练 SVM，同样的，去掉 $$w^2i$$* 最小的那个序号i对应的特征。以此类推，直到剩下的特征数满足我们的需求为止。

1.2.3 嵌入法选择特征

嵌入法也是用机器学习的方法来选择特征，但是它和 RFE 的区别是它不是通过不停的筛掉特征来进行训练，而是使用的都是特征全集。在sklearn中，使用 SelectFromModel 函数来选择特征。

最常用的是**使用L1正则化和L2正则化**来选择特征。在用scikit-learn 和 pandas 学习 Ridge 回归中，我们说到**正则化惩罚项越大，那么模型的系数就会越小**。当正则化惩罚项大到一定的程度的时候，部分特征系数会变成0，当正则化惩罚项继续增大到一定程度时，所有的特征系数都会趋于0. 但是我们会发现一部分特征系数会更容易先变成0，这部分系数就是可以筛掉的。也就是说，我们选择特征系数较大的特征。常用的L1正则化和L2正则化来选择特征的基学习器是逻辑回归。

此外也可以使用决策树或者GBDT。那么是不是所有的机器学习方法都可以作为嵌入法的基学习器呢? 也不是，**一般来说，可以得到特征系数coef或者可以得到特征重要度(feature importances)的算法才可以做为嵌入法的基学习器。**

1.3 寻找高级特征

在我们拿到已有的特征后，我们还可以根据需要寻找到更多的高级特征。比如有车的路程特征和时间间隔特征，我们就可以得到车的平均速度这个二级特征。根据车的速度特征，我们就可以得到车的加速度这个三级特征，根据车的加速度特征，我们就可以得到车的加加速度这个四级特征。。。也就是说，高级特征可以一直寻找下去。

在Kaggle之类的算法竞赛中，高分团队主要使用的方法除了**集成学习算法，剩下的主要就是在高级特征**上面做文章。
所以寻找高级特征是模型优化的必要步骤之一。当然，在第一次建立模型的时候，我们可以先不寻找高级特征，得到以后基准模型后，再寻找高级特征进行优化。

寻找高级特征最常用的方法有：

- 若干项特征加和： 我们假设你希望根据每日销售额得到一周销售额的特征。你可以将最近的7天的销售额相加得到。
- 若干项特征之差： 假设你已经拥有每周销售额以及每月销售额两项特征，可以求一周前一月内的销售额。
- 若干项特征乘积： 假设你有商品价格和商品销量的特征，那么就可以得到销售额的特征。
- 若干项特征除商： 假设你有每个用户的销售额和购买的商品件数，那么就是得到该用户平均每件商品的销售额。

当然，寻找高级特征的方法远不止于此，它需要你**根据你的业务和模型需要**而得，而不是随便的两两组合形成高级特征，这样容易导致特征爆炸，反而没有办法得到较好的模型。

个人经验是，**聚类的时候高级特征尽量少一点，分类回归的时候高级特征适度的多一点**。

1.4 特征选择小结

特征选择是特征工程的第一步，它关系到我们机器学习算法的上限。因此原则是尽量不错过一个可能有用的特征，但是也不滥用太多的特征。

2.特征表达

在特征选择中，我们讲到了特征选择的一些要点。我们继续讨论特征工程，不过会重点关注于特征表达部分，即如果对某一个特征的具体表现形式做处理。主要包括**缺失值处理，特殊的特征处理比如时间和地理位置处理，离散特征的连续化和离散化处理，连续特征的离散化**处理几个方面。

2.1 缺失值处理

特征有缺失值是非常常见的，大部分机器学习模型在拟合前需要所有的特征都有值，不能是空或者NULL。那么如果有缺失值我们需要怎么处理呢?

首先我们会看是该特征是连续值还是离散值。

- 如果是连续值，那么一般有两种选择，
    - 一是选择所有有该特征值的样本，然后取**平均值**，来填充缺失值，
    - 另一种是取**中位数**来填充缺失值。
- 如果是离散值
    - 一般会选择所有有该特征值的样本中**最频繁出现的类别值**，来填充缺失值。

在sklearn中，可以使用preprocessing.Imputer来选择这三种不同的处理逻辑做预处理。

2.2 特殊的特征处理

有些特征的默认取值比较特殊，一般需要做了处理后才能用于算法。比如日期时间，比如显示20180519，这样的值一般没办法直接使用。那么一般需要如何变换呢?

- 对于**时间原始特征**，处理方法有很多，这里只举例几种有代表性的方法。
    - 第一种是使用连续的时间差值法，即计算出所有样本的时间到某一个未来时间之间的数值差距，这样这个差距是UTC的时间差，从而将时间特征转化为连续值。
    - 第二种方法是根据时间所在的年，月，日，星期几，小时数，将一个时间特征转化为若干个离散特征，这种方法在分析具有明显时间趋势的问题比较好用。
    - 第三种是权重法，即根据时间的新旧得到一个权重值。比如对于商品，三个月前购买的设置一个较低的权重，最近三天购买的设置一个中等的权重，在三个月内但是三天前的设置一个较大的权重。当然，还有其他的设置权重的方法，这个要根据要解决的问题来灵活确定。
- 对**地理特征**，比如“广州市天河区XX街道XX号”，这样的特征我们应该如何使用呢？处理成离散值和连续值都是可以的。
    - 如果是处理成离散值，则需要转化为多个离散特征，比如城市名特征，区县特征，街道特征等。
    - 如果我们需要判断用户分布区域，则一般处理成连续值会比较好，这时可以将地址处理成经度和纬度的连续特征。

2.3 离散特征的连续化处理

有很多机器学习算法只能处理连续值特征，不能处理离散值特征，比如线性回归，逻辑回归等。那么想使用逻辑回归，线性回归时这些值只能丢弃吗？当然不是。我们可以将离散特征连续化处理。

- 局部表示：
    - 局部表示有两个优点：
        - 1）这种离散的表示方式具有很好的解释性，有利于人工归纳和总结特征，并通过特征组合进行高效的特征工程；
        - 2）通过多种特征组合得到的表示向量通常是稀疏的二值向量，当用于线性模型时计算效率非常高.
    - 局部表示有两个不足之处：
        - 1）one-hot向量的维数很高，且不能扩展。如果有一种新的颜色，我们就需要增加一维来表示；
        - 2）不同颜色之间的相似度都为0，即我们无法知道“红色”和“中国红”的相似度要高于“红色”和“黑色”的相似度.
- 分布式表示：示通常可以表示为低维的稠密向量；解决了局部表示的缺点。

- 最常见的离散特征连续化的处理方法是**独热编码one-hot encoding【局部表示】**。处理方法其实比较简单，比如某特征的取值是高，中和低，那么我们就可以创建三个取值为0或者1的特征，将高编码为1,0,0；中编码为0,1,0；低编码为0,0,1这样三个特征。也就是说，之前的一个特征被我们转化为了三个特征。sklearn的OneHotEncoder可以帮我们做这个处理。
- 第二个方法是特征**嵌入embedding【分布式表示】**。这个一般用于深度学习中。比如对于用户的ID这个特征，如果要使用独热编码，则维度会爆炸，如果使用特征嵌入就维度低很多了。对于每个要嵌入的特征，我们会有一个特征嵌入矩阵，这个矩阵的行很大，对应我们该特征的数目。比如用

> 户ID，如果有100万个，那么嵌入的特征矩阵的行就是100万。但是列一般比较小，比如可以取20。这样每个用户ID就转化为了一个20维的特征向量。进而参与深度学习模型。在tensorflow中，我们可以先随机初始化一个特征嵌入矩阵，对于每个用户，可以用tf.nn.embedding_lookup找到该用户的特征嵌入向量。特征嵌入矩阵会在反向传播的迭代中优化。

此外，在自然语言处理中，我们也可以用word2vec将词转化为词向量，进而可以进行一些连续值的后继处理。

2.4 离散特征的离散化处理

离散特征有时间也不能直接使用，需要先进行转化。比如最常见的，如果特征的取值是高，中和低，那么就算你需要的是离散值，也是没法直接使用的。

对于原始的离散值特征，最常用的方法也是独热编码，方法在第三节已经讲到。

第二种方法是虚拟编码dummy coding，它和独热编码类似，但是它的特点是，如果我们的特征有N个取值，它只需要N-1个新的0,1特征来代替，而独热编码会用N个新特征代替。比如一个特征的取值是高，中和低，那么我们只需要两位编码，比如只编码中和低，如果是1，0则是中，0,1则是低。0,0则是高了。目前**虚拟编码**使用的没有独热编码广，因此一般有需要的话还是使用独热编码比较好。

此外，有时候我们可以对特征进行研究后做一个更好的处理。比如，我们研究商品的销量对应的特征。里面有一个原始特征是季节春夏秋冬。我们可以将其转化为淡季和旺季这样的二值特征，方便建模。当然有时候转化为三值特征或者四值特征也是可以的。

对于分类问题的特征输出，我们一般需要用sklearn的LabelEncoder将其转化为0,1,2，...这样的类别标签值。

2.5 连续特征的离散化处理

对于连续特征，有时候我们也可以将其做离散化处理。这样特征变得高维稀疏，方便一些算法的处理。

对常用的方法是根据阈值进行分组，比如我们根据连续值特征的分位数，将该特征分为高，中和低三个特征。将分位数从0-0.3的设置为高，0.3-0.7的设置为中，0.7-1的设置为高。

当然还有高级一些的方法。比如使用GBDT。在**LR+GBDT**的经典模型中，就是使用GDBT来先将连续值转化为离散值。那么如何转化呢？比如我们用训练集的所有连续值和标签输出来训练GBDT，最后得到的GBDT模型有两颗决策树，第一颗决策树有三个叶子节点，第二颗决策树有4个叶子节点。如果某一个样本在第一颗决策树会落在第二个叶子节点，在第二颗决策树落在第4颗叶子节点，那么它的编码就是0,1,0,0,0,0,1，一共七个离散特征，其中会有两个取值为1的位置，分别对应每颗决策树中样本落点的位置。在sklearn中，我们可以用GradientBoostingClassifier的 apply方法很方便的得到样本离散化后的特征，然后使用独热编码即可。

> 具体的一个示例代码如下：

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import OneHotEncoder
X, y = make_classification(n_samples=10)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
gbc = GradientBoostingClassifier(n_estimators=2)
one_hot = OneHotEncoder()
gbc.fit(X_train, y_train)
X_train_new = one_hot.fit_transform(gbc.apply(X_train)[:, :, 0])
print (X_train_new.todense())
```

输出是：

[[0. 1. 1. 0.]
 [1. 0. 0. 1.]
 [1. 0. 0. 1.]
 [1. 0. 0. 1.]
 [0. 1. 1. 0.]]

2.6 小结

本文总结了特征表达的一些具体方法， 但是特征表达的方法便不止于上文中的方法，毕竟这是工程实践。但是上文中的方法是比较普遍的，希望可以给大家一些帮助和启发。

3.特征与处理

主要包括特征的归一化和标准化，异常特征样本清洗与样本数据不平衡问题的处理。

3.1 特征的标准化和归一化

由于标准化和归一化这两个词经常混用，所以本文不再区别标准化和归一化，而通过具体的标准化和归一化方法来区别具体的预处理操作。

z-score标准化：这是最常见的特征预处理方式，基本所有的线性模型在拟合的时候都会做 z-score标准化。具体的方法是求出样本特征x的均值mean和标准差std，然后用 **(x-mean)/std** 来代替原特征。这样特征就变成了均值为0，方差为1了。
在sklearn中，我们可以用StandardScaler来做z-score标准化。当然，如果我们是用pandas做数据预处理，可以自己在数据框里面减去均值，再除以方差，自己做z-score标准化。

max-min标准化：也称为离差标准化，预处理后使特征值映射到[0,1]之间。具体的方法是求出样本特征x的最大值max和最小值min，然后用
(x-min)/(max-min) 来代替原特征。如果我们希望将数据映射到任意一个区间[a,b]，而不是[0,1]，那么也很简单。用(x-min)(b-a)/(max-min)+a来代替原特征即可。在sklearn中，我们可以用MinMaxScaler来做max-min标准化。这种方法的问题就是如果测试集或者预测数据里的特征有小于min，或者大于max的数据，会导致max和min发生变化，需要重新计算。所以实际算法中， 除非你对特征的取值区间有需求，否则max-min标准化没有 z-score标准化好用。

L1/L2范数标准化：如果我们只是为了统一量纲，那么通过L2范数整体标准化也是可以的，具体方法是求出每个样本特征向量$\vec{x}$的L2范数$||\vec{x}||2$,然后用$\vec{x}/||\vec{x}||2$代替原样本特征即可。当然L1范数标准化也是可以的，即用$\vec{x}/||\vec{x}||_1$代替原样本特征。通常情况下，范数标准化首选L2范数标准化。在sklearn中，我们可以用Normalizer来做L1/L2范数标准化。

此外，经常我们还会用到**中心化**，主要是在PCA降维的时候，此时我们求出特征x的平均值mean后，用**x-mean代替原特征**，也就是特征的均值变成了0, 但是方差并不改变。这个很好理解，因为PCA就是依赖方差来降维的。

虽然大部分机器学习模型都需要做标准化和归一化，也有不少模型可以不做做标准化和归一化，**主要是基于概率分布的模型，比如决策树大家族的CART，随机森林等**。当然此时使用标准化也是可以的，大多数情况下对模型的泛化能力也有改进。

3.2 异常特征样本清洗

我们在实际项目中拿到的数据往往有不少异常数据，有时候不筛选出这些异常数据很可能让我们后面的数据分析模型有很大的偏差。那么如果我们没有专业知识，如何筛选出这些异常特征样本呢？常用的方法有两种。

- 第一种是聚类，比如我们可以用KMeans聚类将训练样本分成若干个簇，如果某一个簇里的样本数很少，而且簇质心和其他所有的簇都很远，那么这个簇里面的样本极有可能是异常特征样本了。我们可以将其从训练集过滤掉。
- 第二种是异常点检测方法，主要是使用 iForest 或者 one class SVM ，使用异常点检测的机器学习算法来过滤所有的异常点。

当然，某些筛选出来的异常样本是否真的是不需要的异常特征样本，最好找懂业务的再确认一下，防止我们将正常的样本过滤掉了。

3.3 处理不平衡数据

这个问题其实不算特征预处理的部分，不过其实它的实质还是训练集中各个类别的样本的特征分布不一致的问题，所以这里我们一起讲。

我们做分类算法训练时，如果训练集里的各个类别的样本数量不是大约相同的比例，就需要处理样本不平衡问题。也许你会说，不处理会怎么样呢？如果不处理，那么拟合出来的模型对于训练集中少样本的类别泛化能力会很差。举个例子，我们是一个二分类问题，如果训练集里A类别样本占90%，B类别样本占10%。 而测试集里A类别样本占50%， B类别样本占50%， 如果不考虑类别不平衡问题，训练出来的模型对于类别B的预测准确率会很低，甚至低于50%。

如何解决这个问题呢？一般是两种方法：权重法或者采样法。

- 权重法是比较简单的方法，我们可以对训练集里的每个类别加一个权重class weight。如果该类别的样本数多，那么它的权重就低，反之则权重就高。如果更细致点，我们还可以对每个样本加权重sample weight，思路和类别权重也是一样，即样本数多的类别样本权重低，反之样本权重高。sklearn中，绝大多数分类算法都有class weight和 sample weight可以使用。

如果权重法做了以后发现预测效果还不好，可以考虑采样法。

- 采样法常用的也有两种思路，
    - 一种是对类别样本数多的样本做子采样, 比如训练集里A类别样本占90%，B类别样本占10%。那么我们可以对A类的样本子采样，直到子采样得到的A类样本数和B类别现有样本一致为

止，这样我们就只用子采样得到的A类样本数和B类现有样本一起做训练集拟合模型。
    - 第二种思路是对类别样本数少的样本做过采样, 还是上面的例子，我们对B类别的样本做过采样，直到过采样得到的B类别样本数加上B类别原来样本一起和A类样本数一致，最后再去拟合模型。

上述两种常用的采样法很简单，但是都有个问题，就是采样后改变了训练集的分布，可能导致泛化能力差。所以有的算法就通过其他方法来避免这个问题，比如SMOTE算法通过人工合成的方法来生成少类别的样本。方法也很简单，对于某一个缺少样本的类别，它会随机找出几个该类别的样本，再找出最靠近这些样本的若干个该类别样本，组成一个候选合成集合，然后在这个集合中不停的选择距离较近的两个样本，在这两个样本之间，比如中点，构造一个新的该类别样本。举个例子，比如该类别的候选合成集合有两个样本($x1$,y),($x2$,y),那么SMOTE采样后，可以得到一个新的训练样本(($x1$+$x2$)/2,y),通过这种方法，我们可以得到不改变训练集分布的新样本，让训练集中各个类别的样本数趋于平衡。我们可以用imbalance-learn这个Python库中的SMOTEENN类来做SMOTE采样。

总结：
三种方法：

- 对较多的那个类别进行**欠采样**(under-sampling)，舍弃一部分数据，使其与较少类别的数据相当
- 对较少的类别进行**过采样**(over-sampling)，重复使用一部分数据，使其与较多类别的数据相当
- 阈值调整（threshold moving），将原本默认为0.5的阈值调整到 较少类别/（较少类别+较多类别）即可

前面2种的优缺点：

- 明显的改变数据分布，训练数据假设不再是真实数据的无偏表述。
- 在第一种方法中，我们浪费了很多数据。
- 第二类方法中有无中生有或者重复使用了数据，会导致过拟合的发生

但不难看出，其实这样的过程是需要花时间处理数据和编程的，对于很多知识和能力有限的人来说难度比较大。

特此推荐两个简单易行且效果中上的做法：

- 简单的调整阈值，不对数据进行任何处理。此处特指将分类阈值从0.5调整到正例比例
- 使用现有的集成学习分类器，如随机森林或者xgboost，并调整分类阈值

提出这样建议的原因有：

- 首先，简单的阈值调整从经验上看往往比过采样和欠采样有效。
- 其次，如果你对统计学知识掌握有限，而且编程能力一般，在集成过程中更容易出错，还不如使用现有的集成学习并调整分类阈值。

3.4 结语

特征工程终于写完了，这个系列的知识比较零散，更偏向工程方法，所以不像算法那么紧凑，写的也不是很好，希望大家批评指正。如果有其他好的特征工程方法需要补充的，欢迎留言评论。