原文地址：Approaching (Almost) Any Machine Learning Problem

原文翻译与校对：@姜范波 && @黄文畅 && 寒小阳

时间：2016年10月。

出处：http://blog.csdn.net/han\_xiaoyang/article/details/52910022

声明：版权所有，转载请联系作者并注明出

前言

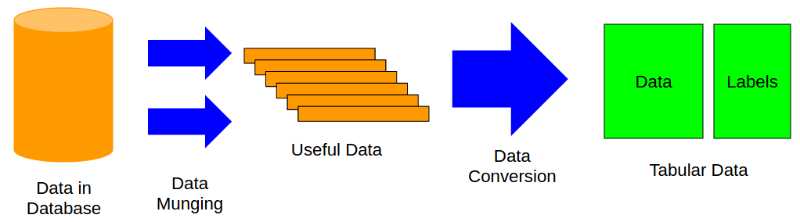
本文由Searchmetrics公司高级数据科学家Abhishek Thakur提供。

一个中等水平的数据科学家每天都要处理大量的数据。一些人说超过60%到70%的时间都用于数据清理、数据处理及格式转化，以便于在之后应用机器学习模型。这篇文章的重点便在后者—— 应用机器学习模型（包括预处理的阶段）。此文讨论到的内容来源于我参加的过的数百次的机器学习竞赛。请大家注意这里讨论的方法是大体上适用的，当然还有很多被专业人士使用的非常复杂的方法。

接下来会使用到python。

数据

在应用机器学习模型之前，所有的数据都必须转换为表格形式。如下图所示，这个过程是最耗时、最困难的部分。



转换完成之后，便可以将这些表格数据灌入机器学习模型。表格数据是在机器学习或是数据挖掘中最常见的数据表示形式。我们有一个数据表，x轴是样本数据，y轴是标签。标签可以是单列可以是多列，取决于问题的形式。我们会用X表示数据，y表示标签。

**标签的种类**

标签会定义你要解决何种问题，有不同的问题类型。例如：

•单列，二进制值（分类问题，一个样本仅属于一个类，并且只有两个类）

•单列，实数值（回归问题，只预测一个值）

•多列，二进制值（分类问题，一个样本属于一个类，但有两个以上的类）

•多列，实数值（回归问题，多个值的预测）

•多个标签（分类问题，一个样本可以属于几个类）

**评估指标**

对于任何类型的机器学习问题，我们都一定要知道如何评估结果，或者说评估指标和目的是什么。举例来说，对于不均衡的二进制分类问题，我们通常选择受试者工作特征曲线下面积（ROC AUC或简单的AUC）；对于多标签或多类别的分类问题，我们通常选择分类交叉熵或多类对数损失；对于回归问题，则会选择均方差。

我不会再深入的讲解不同的评估指标，因为根据问题的不同会有很多不同的种类。

**库**

开始尝试机器学习库可以从安装最基础也是最重要的开始，像numpy和scipy。

•查看和执行数据操作：pandas（http://pandas.pydata.org/）

•对于各种机器学习模型：scikit-learn（http://scikit-learn.org/stable/）

•最好的gradient boosting库：xgboost（https://github.com/dmlc/xgboost）

•对于神经网络：keras（http://keras.io/）

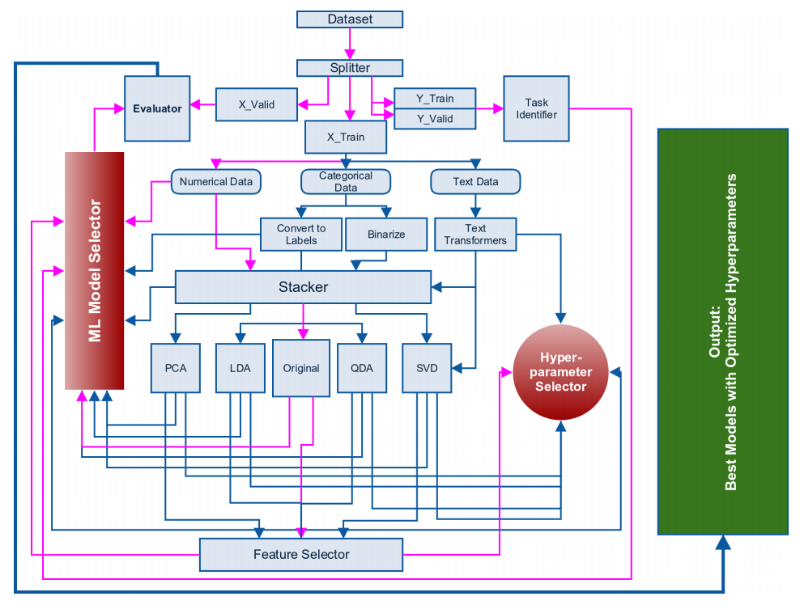
•数据绘图：matplotlib（http://matplotlib.org/）

•监视进度：tqdm（https://pypi.python.org/pypi/tqdm）

Anaconda操作简单而且帮你准备好了这些，可是我没有使用，因为我习惯自己配置和自由使用。当然，决策在你手中。

**机器学习总体框架**

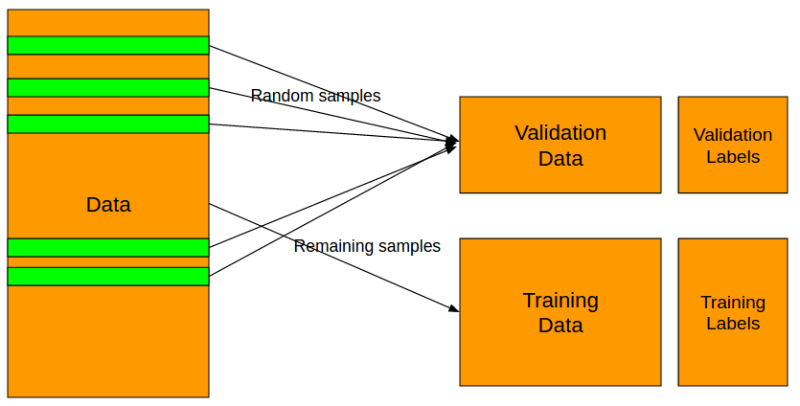
2015起，我开始制作一个自动机器学习框架，还在完善过程中，很快就会发布。下图所示的框架图就是这篇文章中将会提到的基础框架：



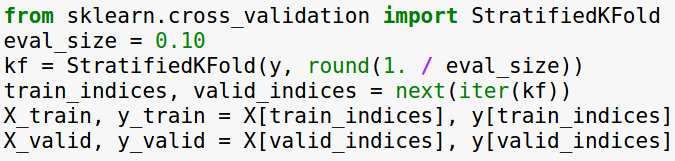
图片来源：A. Thakur and A. Krohn-Grimberghe, AutoCompete: A Framework for Machine Learning Competitions, AutoML Workshop, International Conference on Machine Learning 2015

上面的框架图中，粉色的线代表最常用的路径。结束提取数据并将其转化为表格形式，我们就可以开始建造机器学习模型了。

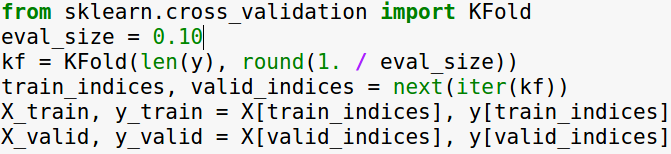
第一步是识别（区分）问题。这个可以通过观察标签解决。你一定要知道这个问题是二元分类，还是多种类或多标签分类，还是一个回归问题。当识别了问题之后，就可以把数据分成训练集和测验集两个部分。如下图所示。



将数据分成训练集和验证集“必须”根据标签进行。遇到分类问题，使用分层分割就对了。在Python中，用scikit-learn很容易就做到了。



遇到回归问题，一个简单的K-Fold分割就可以了。当然，也还有很多复杂的方法能够在维持训练集和验证集原有分布的同时将数据分割开来。这个就留给读者们自己去练习啦。



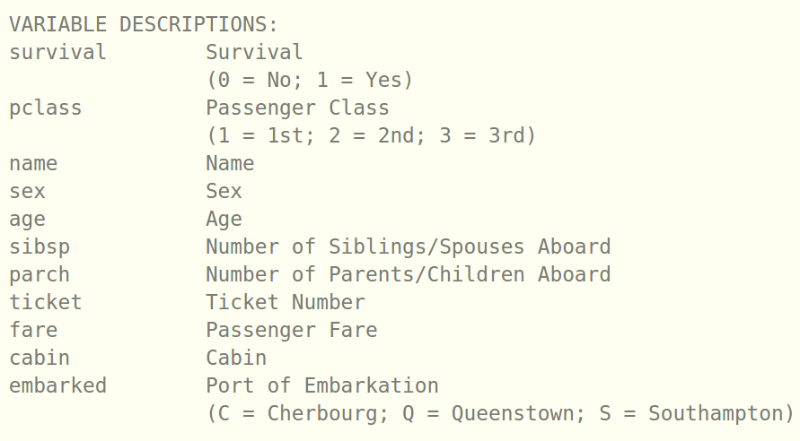
相关阅读: Startup Lessons: This Is Why You Need To Move Quickly

在以上的例子中我选择用全数据的10%作为验证集，当然你可以根据手中具体的数据决定取样的大小。

分好数据之后，就可以把它放在一边不要碰了。任何作用于训练集的运算都必须被保存并应用于验证集。验证集无论如何都不可以和训练集混为一谈。因为混到一起之后虽然回到一个让用户满意的评估指标值，但却会因为模型过拟合而不能使用。

下一步是识别数据中不同的变量。通常有三种变量：数值变量、分类变量和文本变量。让我们用很受欢迎的关于泰坦尼克号的数据集来举个例子。

(https://www.kaggle.com/c/titanic/data).



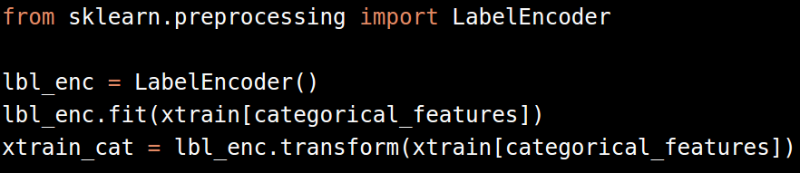
在这里，“survival”（生存）就是标签。在前一个步骤中我们已经把标签从训练集中去掉了。接下来，有pclass，sex, embarked变量这些变量由不同的级别，因此是分类变量。像age, sibsp, parch等就是数值变量。Name是一个含有文本的变量，但我不认为它对预测是否生存有用。

先把数值变量分离出来。这些变量不需要任何形式的处理，所以我们可以开始对其归一并应用机器学习模型。

处理分类变量有两种变法：

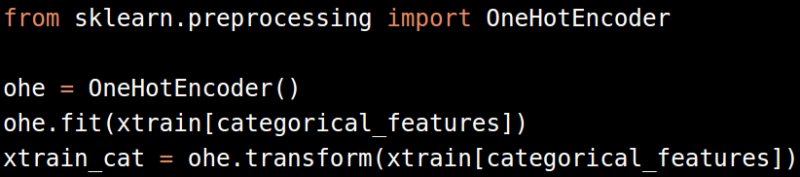
•

把分类变量转化为标签



•

把标签转化为二进制变量



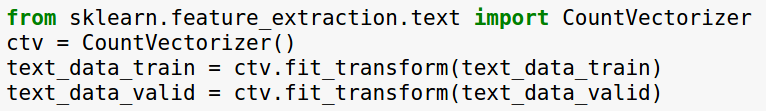
请记住在应用OneHotEncoder之前要用LabelEncoder把分类变量转化为数值变量。

既然泰坦尼克数据里面没有好的关于文本变量的例子，我们就自己制定一个处理文本变量的一般规则。我们可以把所有文本变量整合在一起然后用一些文本分析的算法把他们转换成数字。

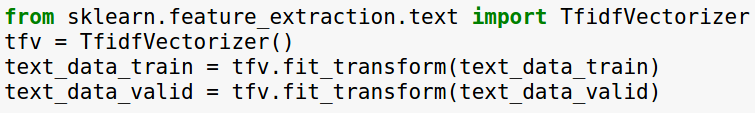
文本变量可以如下这样整合：

IMG_256

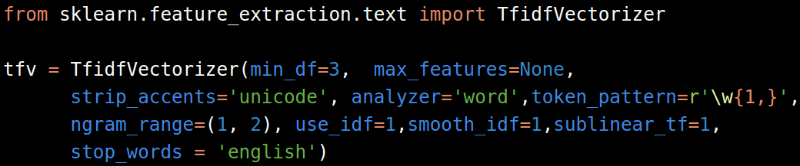
然后就可以应用CoutVectorizer或者TfidfVectorizer在上面啦：



或者



TfidfVectorizer大多数时候比单纯计数效果要好。下面的参数设置在大多数时候都有很好的结果。



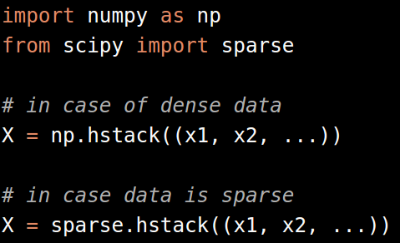
如果你在训练集上做了向量化处理（或者其他操作），请确保将其（相应的处理）转存在硬盘上，以便以后在验证集上应用。

IMG_256

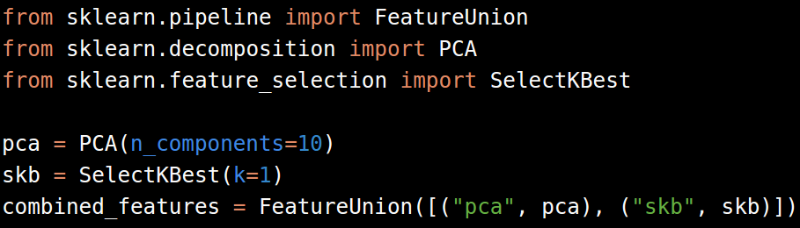
接下来，就是堆叠器模块。堆叠器模块不是模型堆叠而是特征堆叠。上述处理步骤之后得到的不同特征可以通过堆叠器模块整合到一起。



你可以水平地堆叠所有的特征，然后通过使用numpy hstack或sparse hvstack进行进一步处理，具体取决于是否具有密集或稀疏特征。



也可以通过FeatureUnion模块实现，以防万一有其他处理步骤，如PCA或特征选择（我们将在后文提到分解和特征选择）。



一旦我们把特征找齐了，就可以开始应用机器学习模型了。在这个阶段，你只需用到基于树的模型，包括：

•随机森林分类器

•随机森林回归器

•ExtraTrees分类器

•ExtraTrees回归器

•XGB分类器

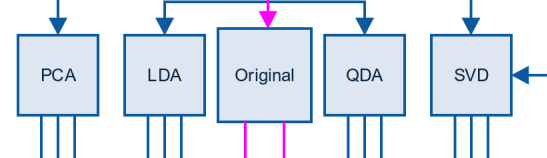
•XGB回归器

由于没有归一化，我们不能将线性模型应用到上述特征上。为了能够应用线性模型，可以从scikit-learn中使用Normalizer或者StandardScaler。

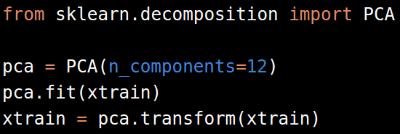
这些归一化的方法仅限于密集特征，对稀疏特征，结果差强人意。当然，也可以在不使用平均值（参数：with\_mean=False）的情况下对稀疏矩阵使用StandardScaler。

如果以上步骤得到了一个“好的”模型，我们就可以进一步做超参数的优化了。得不到的情况下，可以做如下步骤以改进模型。

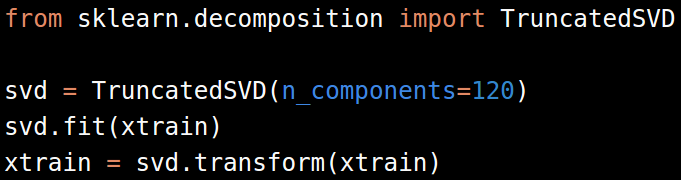
接下来的步骤包括分解模型：



简洁起见，我们跳过LDA和QDA转化。对高维数据，一般而言，PCA可以用来分解数据。对图片而言，从10-15个组分起始，在结果质量持续改进的前提下，逐渐增加组分数量。对其它的数据而言，我们挑选50-60个组分作为起点（对于数字型的数据，只要我们能够处理得了，就不用PCA）

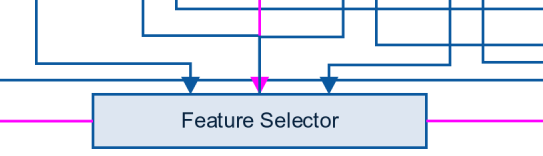


对文本型的数据，把文本转化为稀疏矩阵后，进行奇异值分解（Singular Value Decomposition (SVD)），可以在scikit-learn中找到一个SVD的变异版叫做TruncatedSVD。



一般对TF-IDF有效的奇异值分解成分(components)是120-200个。更多的数量可能效果会有所改进但不是很明显，而计算机资源耗费却很多。

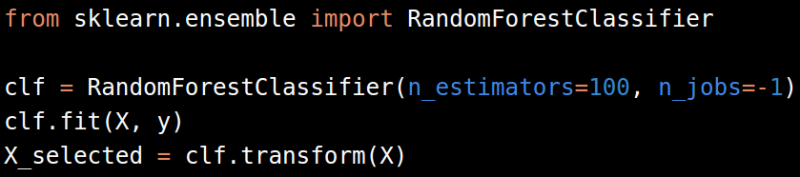
在进一步评价模型的性能以后，我们可以再做数据集的缩放，这样就可以评价线性模型了。归一化或者缩放后的特征可以用在机器学习模型上或者特征选择模块里。



特征选择有很多方法。最常用的方法之一是贪心算法选择（正向或反向）。具体而言，选择一个特征，在一个固定的评价矩阵上训练一个模型，评价其性能，然后一个一个地往里面增加或移除特征，记录每一步的模型性能。最后选择性能得分最高时的那些特征。贪心算法和其评价矩阵的AUC的一个例子见链接： https://github.com/abhishekkrthakur/greedyFeatureSelection。

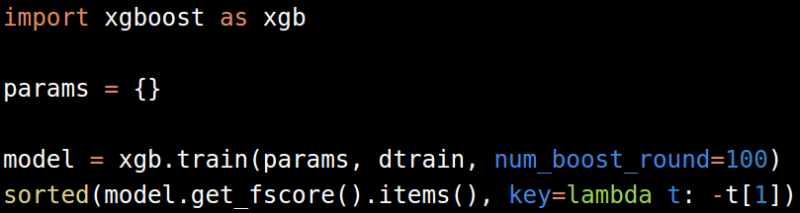
需要注意的是，这个应用并非完美，必须根据要求进行修改。

其他更快的特征选择方法包括从一个模型中选取最好的特征。我们可以根据一个逻辑回归模型的系数，或者训练一个随机森林来选择最好的特征，然后把它们用在其它的机器学习模型里。



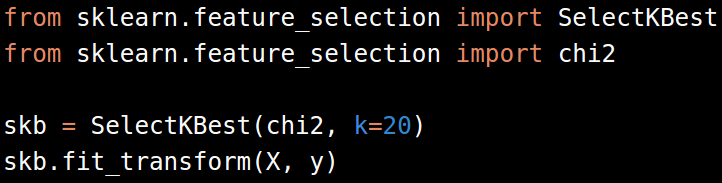
记得把估计值或者超参数的数量控制得尽量少，这样你才不会过拟合。

用Gradient Boosting Machine也可以实现特征选择。如果能用xgboost就不要用GBM，因为前者要快得多，可扩展性更好。



对稀疏数据集，也可以用随机森林分类器/随机森林回归器或xgboost做特征选择。

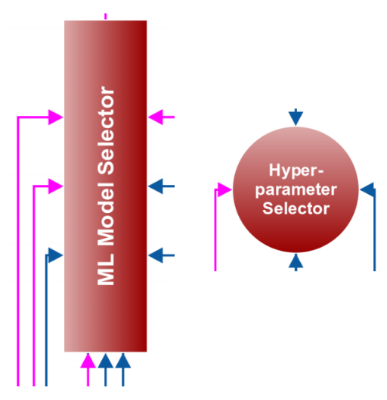
从正性稀疏数据集里选择特征的其它流行方法还有基于卡方的特征选择，scikit-learn中即可应用。



这里，我们用卡方联合SelectKBest的方法从数据中选择了20个特征。这个本身也是我们改进机器学习模型的超参数之一。

别忘了把任何中间过程产生的转化体转存起来。在验证集上你会要用它们来评价性能。

下一步（或者说，紧接着）主要的步骤是模型选择+超参数优化。



一般来说，我们用下面的算法来选择机器学习模型：

•

分类

•随机森林

•GBM

•逻辑回归

•朴素贝叶斯

•支持向量机

•K最近邻法

•

回归

•随机森林

•GBM

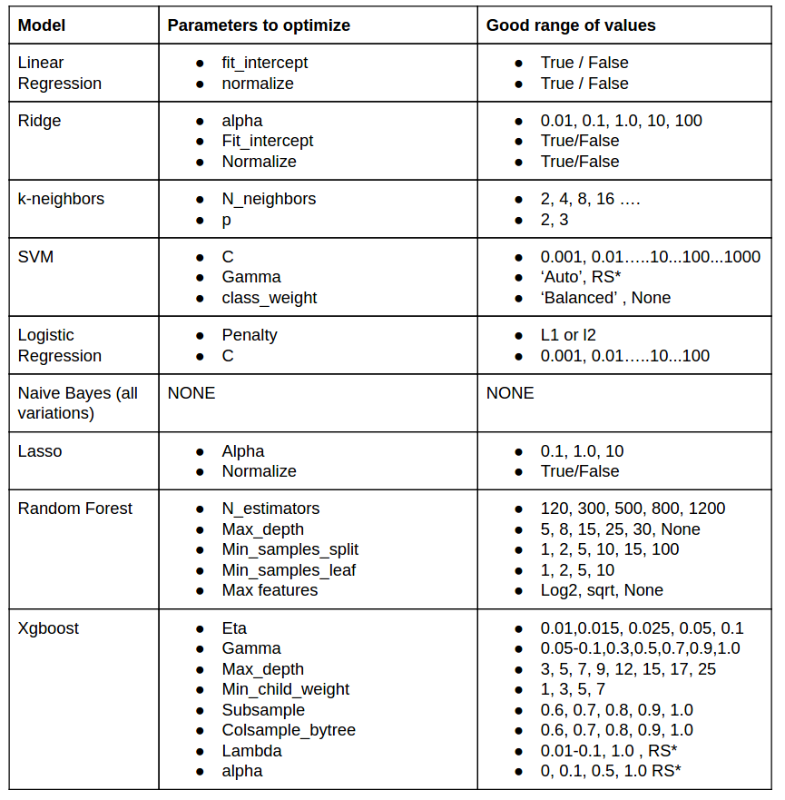
•线性回归

•Ridge

•Lasso

•SVR

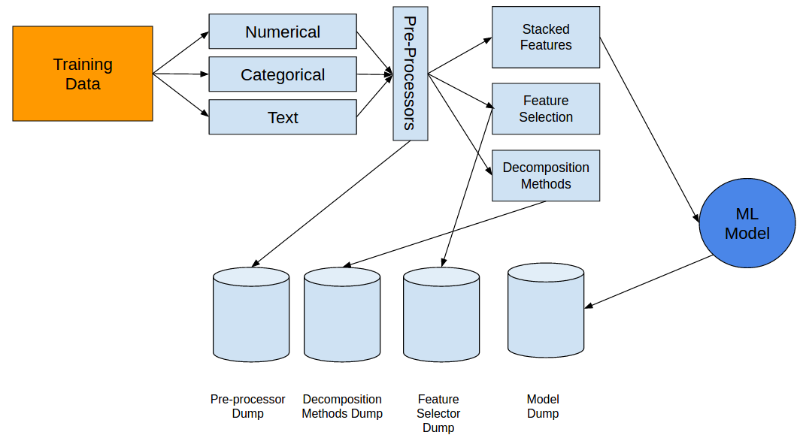
我需要优化哪个参数？如何选择最好的参数？这些是人们经常会遇到的问题。没有大量数据集上不同模型+参数的经验，无法得到这些问题的答案。有经验的人又不愿意把他们的秘诀公之于众。幸运的是，我有丰富的经验，同时愿意分享。让我们来看看不同模型下超参数的秘密：



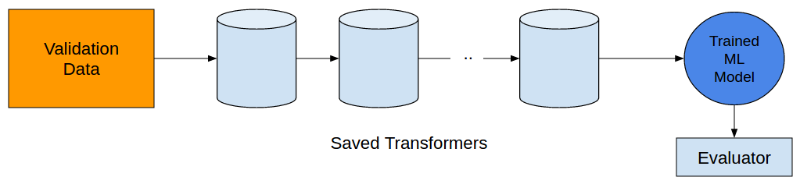
RS\* =不好说合适的值是多少，在这些超参数里随机搜索一下。

以我个人浅见（原文作者个人意见），上述的这些模型比其他模型好，无需评价其它模型。

再说一次，记得保存这些转化体：



然后对验证集做相同的操作。



上面的规则和框架对我遇到的数据集而言运行良好。当然，在特别复杂的情况下也失败过。天下没有完美的东西，我们只能在学习中不断改进，如同机器学习一样。