# Approaching (Almost) Any Machine Learning Problem | Abhishek Thakur 接近（几乎）任何机器学习问题

原文链接：  
原文链接：

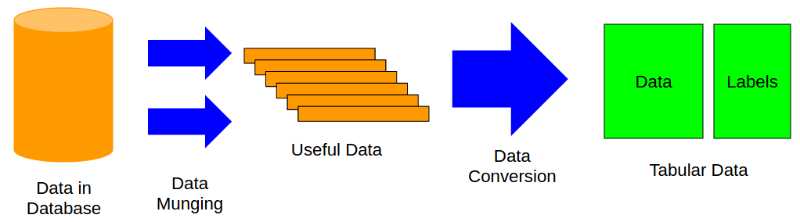
Abhishek Thakur, a Kaggle Grandmaster, originally published this post here on July 18th, 2016 and kindly gave us permission to cross-post on No Free Hunch  
卡格尔大师阿披实克•塔库尔（Abhishek Thakur）最初于2016年7月18日在这里发表了这篇文章，并善意地允许我们在没有自由预感的情况下跨过这篇文章

An average data scientist deals with loads of data daily. Some say over 60-70% time is spent in data cleaning, munging and bringing data to a suitable format such that machine learning models can be applied on that data. This post focuses on the second part, i.e., applying machine learning models, including the preprocessing steps. The pipelines discussed in this post come as a result of over a hundred machine learning competitions that I’ve taken part in. It must be noted that the discussion here is very general but very useful and there can also be very complicated methods which exist and are practised by professionals.  
一个普通的数据科学家每天处理大量的数据。有人说，超过60-70%的时间花在数据清理、咀嚼和将数据转换成合适的格式上，这样机器学习模型就可以应用到这些数据上。本文的重点是第二部分，即应用机器学习模型，包括预处理步骤。本文中讨论的管道是我参加的一百多个机器学习竞赛的结果。必须指出的是，这里的讨论非常一般，但非常有用，也可能是非常复杂的方法，由专业人员存在和实行。

We will be using python!  
我们将使用python！

# Data 数据

Before applying the machine learning models, the data must be converted to a tabular form. This whole process is the most time consuming and difficult process and is depicted in the figure below.  
在应用机器学习模型之前，必须将数据转换为表格形式。整个过程是最耗时、最困难的过程，如下图所示。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_1.png)

The machine learning models are then applied to the tabular data. Tabular data is most common way of representing data in machine learning or data mining. We have a data table, rows with different samples of the data or X and labels, y. The labels can be single column or multi-column, depending on the type of problem. We will denote data by X and labels by y.  
然后将机器学习模型应用于表格数据。表格数据是机器学习或数据挖掘中最常用的数据表示方式。我们有一个数据表，包含不同数据样本的行或X和标签y。根据问题的类型，标签可以是单列或多列。我们将用X表示数据，用y表示标签。

# Types of labels 标签类型

The labels define the problem and can be of different types, such as:  
标签定义了问题，并且可以是不同类型的，例如：

* Single column, binary values (classification problem, one sample belongs to one class only and there are only two classes)  
  单列，二进制值（分类问题，一个样本只属于一个类，只有两个类）
* Single column, real values (regression problem, prediction of only one value)  
  单列，实值（回归问题，仅预测一个值）
* Multiple column, binary values (classification problem, one sample belongs to one class, but there are more than two classes)  
  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）
* Multiple column, real values (regression problem, prediction of multiple values)  
  多列，实值（回归问题，多值预测）
* And multilabel (classification problem, one sample can belong to several classes)  
  和多标签（分类问题，一个样本可以属于多个类）

# Evaluation Metrics 评价指标

For any kind of machine learning problem, we must know how we are going to evaluate our results, or what the evaluation metric or objective is. For example in case of a skewed binary classification problem we generally choose area under the receiver operating characteristic curve (ROC AUC or simply AUC). In case of multi-label or multi-class classification problems, we generally choose categorical cross-entropy or multiclass log loss and mean squared error in case of regression problems.  
对于任何类型的机器学习问题，我们必须知道我们将如何评估我们的结果，或者评估指标或目标是什么。例如，在偏斜的二值分类问题中，我们通常选择接收器工作特性曲线下的区域（ROC-AUC或简称AUC）。对于多标签或多类分类问题，在回归问题中，我们通常选择分类交叉熵或多类对数损失和均方误差。

I won’t go into details of the different evaluation metrics as we can have many different types, depending on the problem.  
我不会详细介绍不同的评估指标，因为根据问题的不同，我们可以有许多不同的类型。

# The Libraries 图书馆

To start with the machine learning libraries, install the basic and most important ones first, for example, numpy and scipy.  
从机器学习库开始，首先安装基本的和最重要的库，例如numpy和scipy。

* To see and do operations on data: pandas ()  
  查看和操作数据：pandas（）
* For all kinds of machine learning models: scikit-learn ()  
  适用于各种机器学习模型：scikit learn（）
* The best gradient boosting library: xgboost ()  
  最佳渐变增强库：xgboost（）
* For neural networks: keras ()  
  对于神经网络：keras（）
* For plotting data: matplotlib ()  
  用于打印数据：matplotlib（）
* To monitor progress: tqdm ()  
  监视进度：tqdm（）

I don’t use Anaconda (). It’s easy and does everything for you, but I want more freedom. The choice is yours. 🙂  
我不用水蟒。这很容易，为你做一切，但我想要更多的自由。选择权在你。🙂

# The Machine Learning Framework 机器学习框架

In 2015, I came up with a framework for automatic machine learning which is still under development and will be released soon. For this post, the same framework will be the basis. The framework is shown in the figure below:  
2015年，我提出了一个自动机器学习框架，该框架仍在开发中，不久将发布。对于这篇文章，同样的框架将是基础。框架如下图所示：

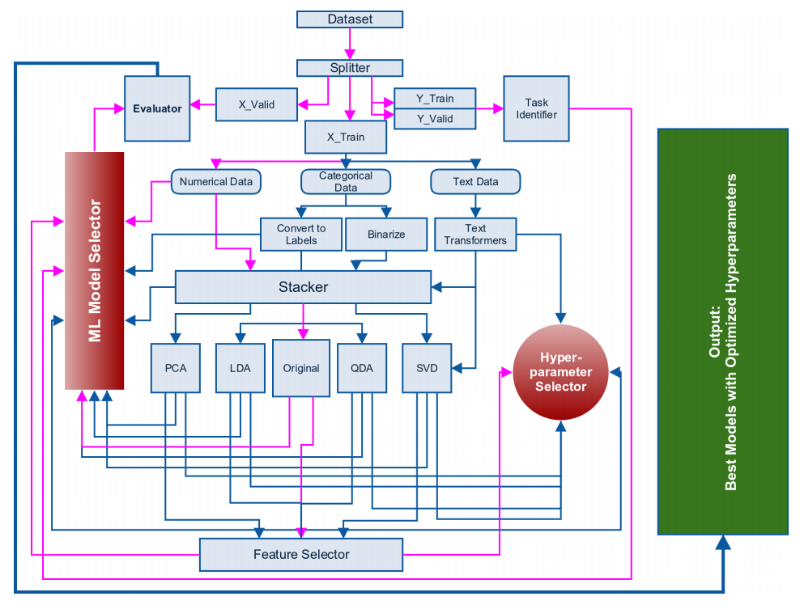
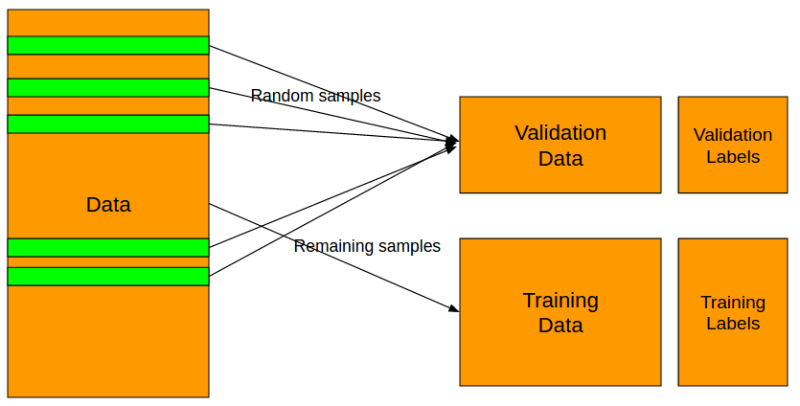


Figure from: A. Thakur and A. Krohn-Grimberghe, AutoCompete: A Framework for Machine Learning Competitions, AutoML Workshop, International Conference on Machine Learning 2015.  
资料来源：A.Thakur和A.Krohn Grimberghe，《自动竞争：机器学习竞赛框架》，AutoML研讨会，2015年机器学习国际会议。

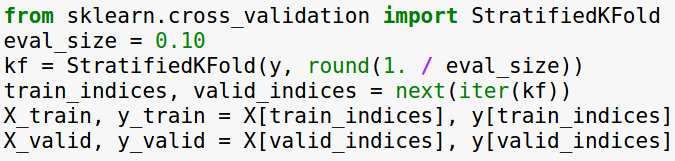
FIGURE FROM: A. THAKUR AND A. KROHN-GRIMBERGHE, AUTOCOMPETE: A FRAMEWORK FOR MACHINE LEARNING COMPETITIONS, AUTOML WORKSHOP, INTERNATIONAL CONFERENCE ON MACHINE LEARNING 2015.  
资料来源：A.THAKUR和A.KROHN-GRIMBERGHE，《自动竞争：机器学习竞赛框架》，AUTOML研讨会，2015年机器学习国际会议。

In the framework shown above, the pink lines represent the most common paths followed. After we have extracted and reduced the data to a tabular format, we can go ahead with building machine learning models.  
在上面所示的框架中，粉红色的线代表了所遵循的最常见的路径。在我们提取数据并将其简化为表格格式之后，我们可以继续构建机器学习模型。

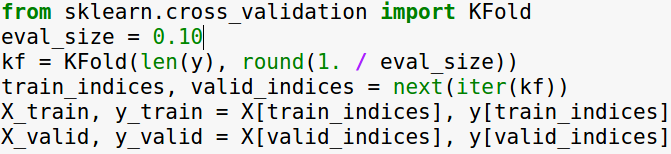
The very first step is identification of the problem. This can be done by looking at the labels. One must know if the problem is a binary classification, a multi-class or multi-label classification or a regression problem. After we have identified the problem, we split the data into two different parts, a training set and a validation set as depicted in the figure below.  
第一步是找出问题所在。这可以通过查看标签来完成。必须知道问题是二元分类、多类或多标签分类还是回归问题。在我们确定了问题之后，我们将数据分成两个不同的部分，一个训练集和一个验证集，如下图所示。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_3.png)

The splitting of data into training and validation sets “must” be done according to labels. In case of any kind of classification problem, use stratified splitting. In python, you can do this using scikit-learn very easily.  
必须根据标签将数据分为训练集和验证集。在任何类型的分类问题，使用分层分裂。在python中，使用scikit learn可以很容易地做到这一点。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_4.png)

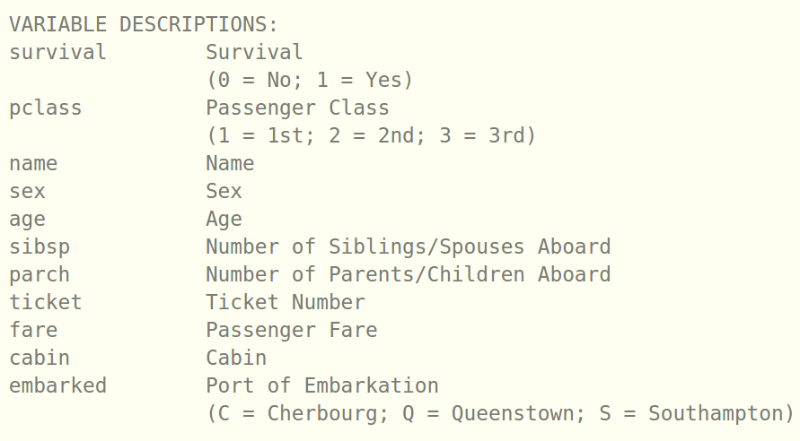
In case of regression task, a simple K-Fold splitting should suffice. There are, however, some complex methods which tend to keep the distribution of labels same for both training and validation set and this is left as an exercise for the reader.  
在回归任务的情况下，简单的K倍分裂就足够了。然而，有一些复杂的方法倾向于保持标签在训练集和验证集中的分布相同，这留给读者作为练习。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_5.png)

I have chosen the eval\_size or the size of the validation set as 10% of the full data in the examples above, but one can choose this value according to the size of the data they have.  
在上面的例子中，我选择了eval\_size或验证集的大小作为完整数据的10%，但是可以根据数据的大小来选择这个值。

After the splitting of the data is done, leave this data out and don’t touch it. Any operations that are applied on training set must be saved and then applied to the validation set. Validation set, in any case, should not be joined with the training set. Doing so will result in very good evaluation scores and make the user happy but instead he/she will be building a useless model with very high overfitting.  
数据拆分完成后，请将此数据保留在外，不要触摸它。必须保存应用于培训集的所有操作，然后将其应用于验证集。在任何情况下，验证集都不应与训练集联接。这样做会得到很好的评价分数，并使用户感到高兴，但相反，他/她将建立一个无用的模型与非常高的过度拟合。

Next step is identification of different variables in the data. There are usually three types of variables we deal with. Namely, numerical variables, categorical variables and variables with text inside them. Let’s take example of the popular Titanic dataset ().  
下一步是识别数据中的不同变量。我们通常处理三种类型的变量。即数值变量、分类变量和文本变量。让我们以流行的泰坦尼克号数据集为例。

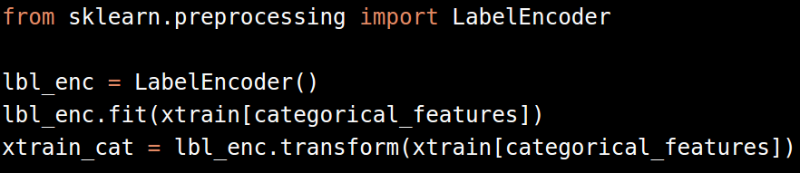
[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_6.png)

Here, survival is the label. We have already separated labels from the training data in the previous step. Then, we have pclass, sex, embarked. These variables have different levels and thus they are categorical variables. Variables like age, sibsp, parch, etc are numerical variables. Name is a variable with text data but I don’t think it’s a useful variable to predict survival.  
在这里，生存是标签。在上一步中，我们已经将标签与培训数据分开。然后，我们开始做爱。这些变量有不同的层次，因此它们是分类变量。年龄、sibsp、parch等变量都是数值变量。Name是一个包含文本数据的变量，但我不认为它是预测生存率的有用变量。

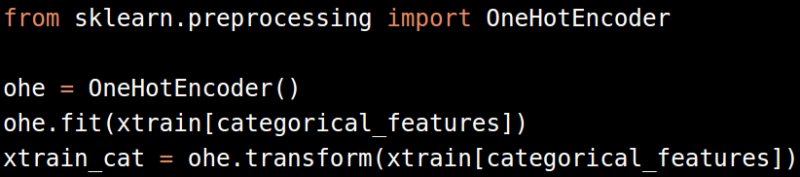
Separate out the numerical variables first. These variables don’t need any kind of processing and thus we can start applying normalization and machine learning models to these variables.  
先把数值变量分开。这些变量不需要任何处理，因此我们可以开始对这些变量应用规范化和机器学习模型。

There are two ways in which we can handle categorical data:  
有两种方法可以处理分类数据：

* Convert the categorical data to labels  
  将分类数据转换为标签

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_7.png)

* Convert the labels to binary variables (one-hot encoding)  
  将标签转换为二进制变量（一个热编码）

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_8.png)

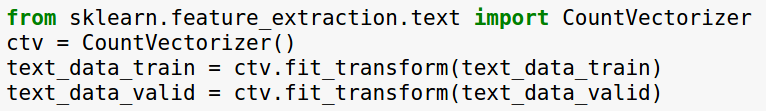
Please remember to convert categories to numbers first using LabelEncoder before applying OneHotEncoder on it.  
请记住，在应用一个hotecoder之前，请先使用LabelEncoder将类别转换为数字。

Since, the Titanic data doesn’t have good example of text variables, let’s formulate a general rule on handling text variables. We can combine all the text variables into one and then use some algorithms which work on text data and convert it to numbers.  
由于泰坦尼克号的数据没有很好的文本变量示例，让我们制定一个处理文本变量的一般规则。我们可以将所有文本变量合并为一个，然后使用一些算法处理文本数据并将其转换为数字。

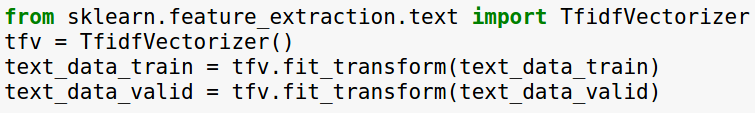
The text variables can be joined as follows:  
文本变量可以按如下方式连接：

[abhishek_9](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_9.png)

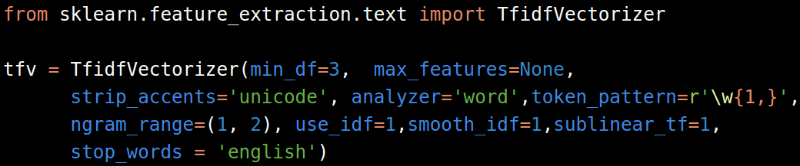
We can then use CountVectorizer or TfidfVectorizer on it:  
然后我们可以在上面使用CountVectorizer或TfidfVectorizer：

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_10.png)

or,  
或者，

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_11.png)

The TfidfVectorizer performs better than the counts most of the time and I have seen that the following parameters for TfidfVectorizer work almost all the time.  
TfidfVectorizer的性能在大多数情况下都优于计数，我已经看到TfidfVectorizer的以下参数几乎一直工作。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_12.png)

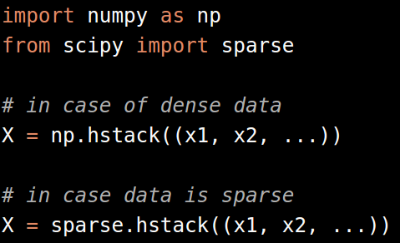
If you are applying these vectorizers only on the training set, make sure to dump it to hard drive so that you can use it later on the validation set.  
如果只在训练集上应用这些矢量器，请确保将其转储到硬盘，以便以后在验证集上使用。

[abhishek_13](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_13.png)

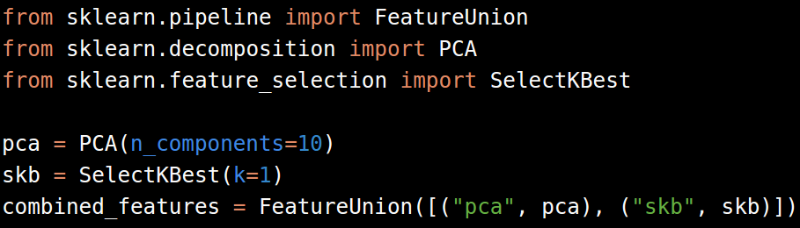
Next, we come to the stacker module. Stacker module is not a model stacker but a feature stacker. The different features after the processing steps described above can be combined using the stacker module.  
接下来，我们来到堆垛机模块。堆垛机模块不是模型堆垛机，而是特征堆垛机。上述处理步骤后的不同特征可以使用堆垛机模块组合。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_14.png)

You can horizontally stack all the features before putting them through further processing by using numpy hstack or sparse hstack depending on whether you have dense or sparse features.  
您可以使用numpy hstack或sparse hstack水平堆叠所有功能，然后再进行进一步处理，具体取决于您是否具有密集或稀疏功能。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_15.png)

And can also be achieved by FeatureUnion module in case there are other processing steps such as pca or feature selection (we will visit decomposition and feature selection later in this post).  
如果还有其他处理步骤，如主成分分析或特征选择（我们将在本文后面访问分解和特征选择），也可以通过特征联合模块来实现。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_16.png)

Once, we have stacked the features together, we can start applying machine learning models. At this stage only models you should go for should be ensemble tree based models. These models include:  
一旦我们将这些特征叠加在一起，我们就可以开始应用机器学习模型了。在这个阶段，您应该选择的模型应该是基于集成树的模型。这些模型包括：

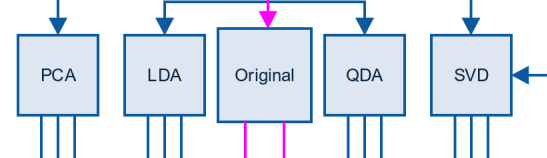
* RandomForestClassifier  
  随机森林分类器
* RandomForestRegressor  
  随机森林回归
* ExtraTreesClassifier  
  树外分类器
* ExtraTreesRegressor  
  树外分频器
* XGBClassifier  
  XGB分类器
* XGBRegressor  
  XGB回归器

We cannot apply linear models to the above features since they are not normalized. To use linear models, one can use Normalizer or StandardScaler from scikit-learn.  
我们不能将线性模型应用于上述特征，因为它们不是标准化的。要使用线性模型，可以使用scikit learn中的Normalizer或StandardScaler。

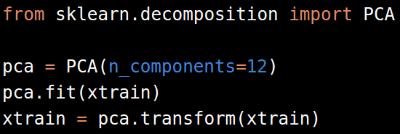
These normalization methods work only on dense features and don’t give very good results if applied on sparse features. Yes, one can apply StandardScaler on sparse matrices without using the mean (parameter: with\_mean=False).  
这些规范化方法只适用于稠密特征，如果应用于稀疏特征，则不会产生很好的效果。是的，可以在不使用平均值的情况下对稀疏矩阵应用StandardScaler（参数：with\_mean=False）。

If the above steps give a “good” model, we can go for optimization of hyperparameters and in case it doesn’t we can go for the following steps and improve our model.  
如果上面的步骤给出了一个“好”的模型，我们可以进行超参数优化，如果没有，我们可以进行下面的步骤并改进我们的模型。

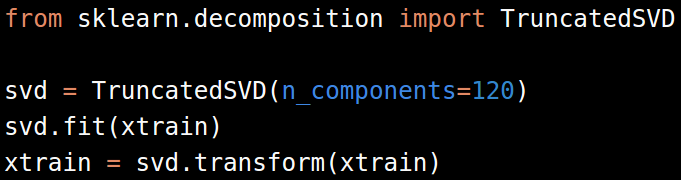
The next steps include decomposition methods:  
接下来的步骤包括分解方法：

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_17.png)

For the sake of simplicity, we will leave out LDA and QDA transformations. For high dimensional data, generally PCA is used decompose the data. For images start with 10-15 components and increase this number as long as the quality of result improves substantially. For other type of data, we select 50-60 components initially (we tend to avoid PCA as long as we can deal with the numerical data as it is).  
为了简单起见，我们将省略LDA和QDA转换。对于高维数据，通常采用PCA对数据进行分解。对于图像，从10-15个分量开始，只要结果的质量有实质性的提高，就增加这个数字。对于其他类型的数据，我们首先选择50-60个分量（只要我们能够处理数值数据，我们倾向于避免PCA）。

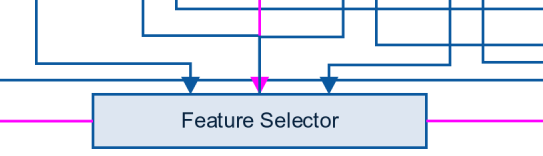
[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_18.png)

For text data, after conversion of text to sparse matrix, go for Singular Value Decomposition (SVD). A variation of SVD called TruncatedSVD can be found in scikit-learn.  
对于文本数据，在将文本转换为稀疏矩阵之后，进行奇异值分解（SVD）。在scikit learn中可以找到一种称为TruncatedSVD的SVD变体。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_decomp.png)

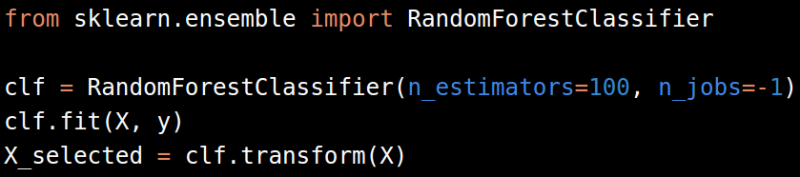
The number of SVD components that generally work for TF-IDF or counts are between 120-200. Any number above this might improve the performance but not substantially and comes at the cost of computing power.  
通常用于TF-IDF或计数的SVD组件数量在120-200之间。上面的任何数字都可能会提高性能，但不会有实质性的提高，而且会以计算能力为代价。

After evaluating further performance of the models, we move to scaling of the datasets, so that we can evaluate linear models too. The normalized or scaled features can then be sent to the machine learning models or feature selection modules.  
在评估模型的进一步性能之后，我们转向数据集的缩放，以便我们也可以评估线性模型。然后，可以将标准化或缩放的特征发送到机器学习模型或特征选择模块。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_19.png)

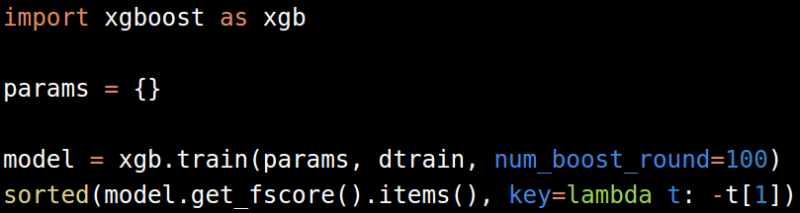
There are multiple ways in which feature selection can be achieved. One of the most common way is greedy feature selection (forward or backward). In greedy feature selection we choose one feature, train a model and evaluate the performance of the model on a fixed evaluation metric. We keep adding and removing features one-by-one and record performance of the model at every step. We then select the features which have the best evaluation score. One implementation of greedy feature selection with AUC as evaluation metric can be found here: . It must be noted that this implementation is not perfect and must be changed/modified according to the requirements.  
有多种方法可以实现特征选择。最常见的方法之一是贪婪的特征选择（向前或向后）。在贪婪特征选择中，我们选择一个特征，训练一个模型，并在一个固定的评价指标上评价模型的性能。我们一个接一个地添加和删除特性，并在每个步骤记录模型的性能。然后选择评价得分最高的特征。贪心特征选择的一个实现，以AUC作为评估指标，可以在这里找到：。必须注意的是，此实现并不完美，必须根据要求进行更改/修改。

Other faster methods of feature selection include selecting best features from a model. We can either look at coefficients of a logit model or we can train a random forest to select best features and then use them later with other machine learning models.  
其他更快的特征选择方法包括从模型中选择最佳特征。我们可以查看一个logit模型的系数，也可以训练一个随机森林来选择最好的特征，然后将它们与其他机器学习模型一起使用。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_20.png)

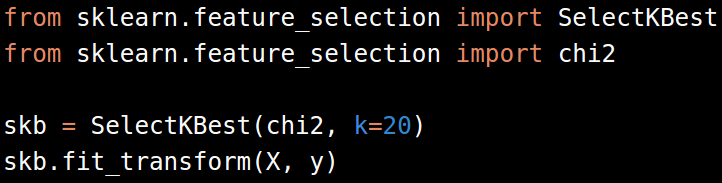
Remember to keep low number of estimators and minimal optimization of hyper parameters so that you don’t overfit.  
记住保持低估计数和超参数的最小优化，这样你就不会过度拟合。

The feature selection can also be achieved using Gradient Boosting Machines. It is good if we use xgboost instead of the implementation of GBM in scikit-learn since xgboost is much faster and more scalable.  
特征选择也可以用梯度助推机来实现。如果我们使用xgboost而不是scikit-learn中的GBM实现，这是很好的，因为xgboost速度更快，可扩展性更强。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_21.png)

We can also do feature selection of sparse datasets using RandomForestClassifier / RandomForestRegressor and xgboost.  
我们还可以使用RandomForestClassifier/RandomForestRegressor和xgboost对稀疏数据集进行特征选择。

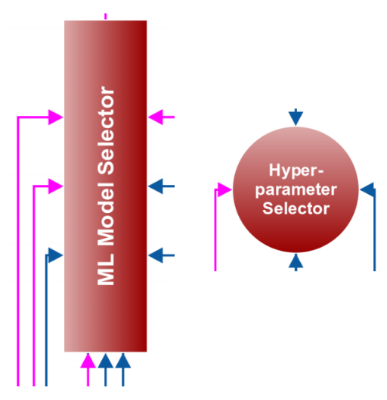
Another popular method for feature selection from positive sparse datasets is chi-2 based feature selection and we also have that implemented in scikit-learn.  
另一种从正稀疏数据集中进行特征选择的流行方法是基于chi-2的特征选择，我们在scikit-learn中也实现了这种方法。

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_22.png)

Here, we use chi2 in conjunction with SelectKBest to select 20 features from the data. This also becomes a hyperparameter we want to optimize to improve the result of our machine learning models.  
在这里，我们使用chi2和SelectKBest从数据中选择20个特性。这也成为我们想要优化的超参数，以改善我们机器学习模型的结果。

Don’t forget to dump any kinds of transformers you use at all the steps. You will need them to evaluate performance on the validation set.  
别忘了扔掉你在所有步骤中使用的任何一种变形金刚。您需要他们评估验证集的性能。

Next (or intermediate) major step is model selection + hyperparameter optimization.  
下一步（或中间）主要是模型选择+超参数优化。

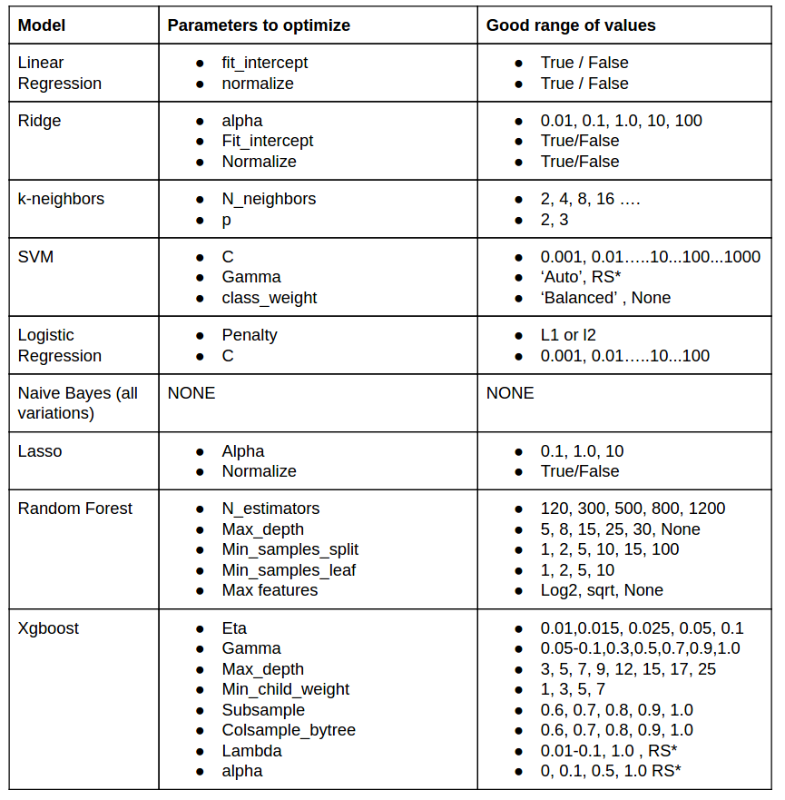
[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_23.png)

We generally use the following algorithms in the process of selecting a machine learning model:  
在选择机器学习模型的过程中，我们通常使用以下算法：

* Classification:  
  分类：
  + Random Forest  
    随机森林
  + GBM  
    GBM公司
  + Logistic Regression  
    对数几率回归
  + Naive Bayes  
    天真的贝耶斯
  + Support Vector Machines  
    支持向量机
  + k-Nearest Neighbors  
    k-最近邻
* Regression  
  回归
  + Random Forest  
    随机森林
  + GBM  
    GBM公司
  + Linear Regression  
    线性回归
  + Ridge  
    山脊
  + Lasso  
    套索
  + SVR  
    SVR公司

Which parameters should I optimize? How do I choose parameters closest to the best ones? These are a couple of questions people come up with most of the time. One cannot get answers to these questions without experience with different models + parameters on a large number of datasets. Also people who have experience are not willing to share their secrets. Luckily, I have quite a bit of experience too and I’m willing to give away some of the stuff.  
我应该优化哪些参数？如何选择最接近最佳参数的参数？这是人们经常提出的几个问题。如果没有在大量数据集上使用不同模型+参数的经验，就无法获得这些问题的答案。有经验的人也不愿意分享他们的秘密。幸运的是，我也有不少经验，我愿意把一些东西送给别人。

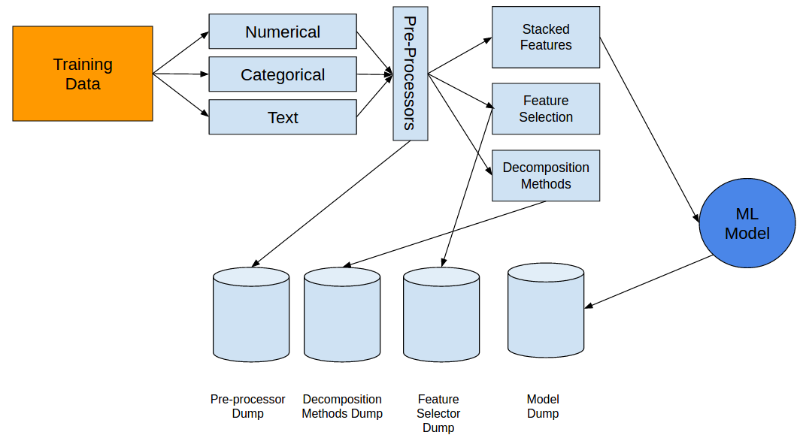
Let’s break down the hyperparameters, model wise:  
让我们分解超参数，模型方面：

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_24.png)

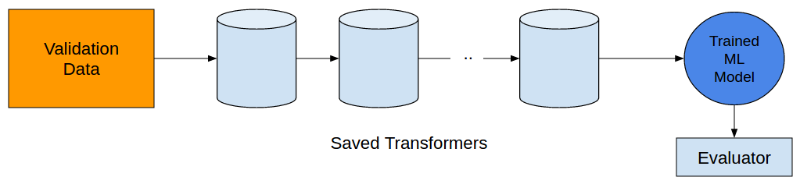
RS\* = Cannot say about proper values, go for Random Search in these hyperparameters.  
RS\*=无法说明正确的值，请在这些超参数中进行随机搜索。

In my opinion, and strictly my opinion, the above models will out-perform any others and we don’t need to evaluate any other models.  
在我看来，严格地说，我的观点是，上述模型将超越任何其他模型，我们不需要评估任何其他模型。

Once again, remember to save the transformers:  
再一次，记住保存变形金刚：

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_25.png)

And apply them on validation set separately:  
并分别应用于验证集：

[](http://s5047.pcdn.co/wp-content/uploads/2016/07/abhishek_26.png)

The above rules and the framework has performed very well in most of the datasets I have dealt with. Of course, it has also failed for very complicated tasks. Nothing is perfect and we keep on improving on what we learn. Just like in machine learning.  
上面的规则和框架在我处理过的大多数数据集中都表现得非常好。当然，对于非常复杂的任务，它也失败了。没有什么是完美的，我们不断地改进我们所学的东西。就像机器学习一样。

Get in touch with me with any doubts: abhishek4 [at] gmail [dot] com