# EDA

<https://www.kaggle.com/aashita/xgboost-model-with-minimalistic-features>

数值型与数值型 分布依赖图

类别型和数值型 箱图

类别型和类别型 条形图

1. 先画单个变量与Y的关系，找到有关的变量和无关的变量
2. 针对有关的变量，进一步组合画出与Y的关系

比如，存活人中女性比例高，消费水平高的人也高，那是不是因为消费水平高的人都是女性，如果是消费 水平一致前提下，男性的存活比例较高了呢

1. 针对无关的变量，也可以进一步分析其组合是否有新的变量

连续型变量：离散化

离散变量：笛卡尔积模拟新特征

<https://www.kaggle.com/eraaz1/a-comprehensive-guide-to-titanic-machine-learning/comments#380632>

1. P，F,T检验
2. 把训练集和测试集合在一起之后，可以用value\_count()自动过滤nan值

<http://blog.kaggle.com/2016/04/08/homesite-quote-conversion-winners-write-up-1st-place-kazanova-faron-clobber/>

1.对于各种变量可以做的处理

* Categorical ⇒ ID per category
* Categorical ⇒ value count for each category
* Categorical ⇒ out-of-fold likelihood for each category with respect to the target attribute
* Categorical ⇒ one-hot encoding
* Numerical ⇒ as is
* Numerical ⇒ percentile transformation
* One-hot encodings and value counts of all features with distinct values below a threshold

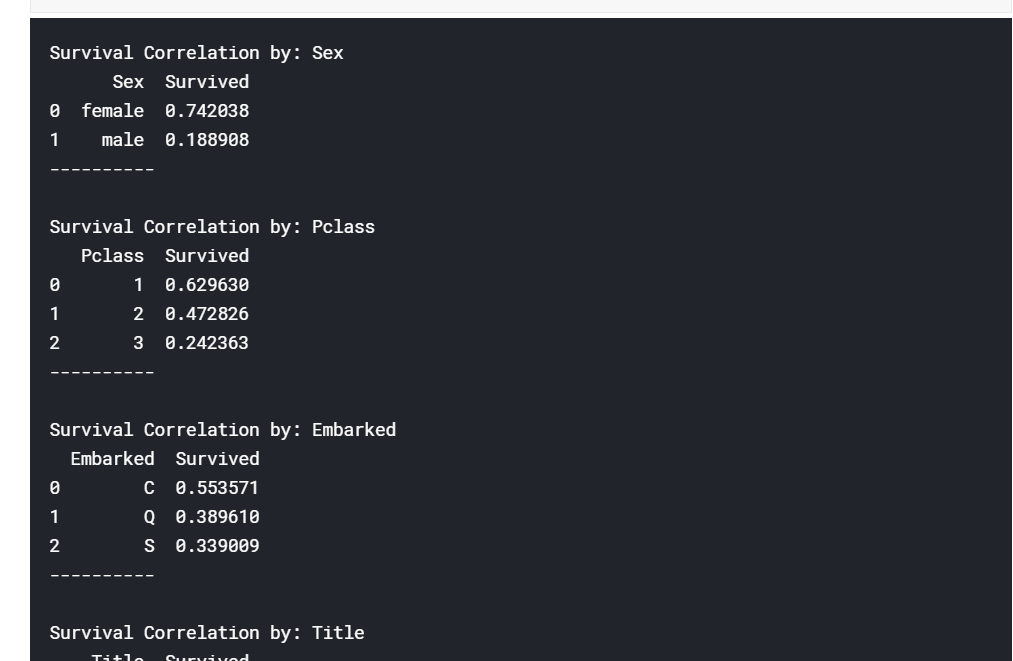
2.研究特征变量之间的相关性

extracted year, month, day & weekday out of Original\_Quote\_Date,

explored summary statistics **2-, 3- & 4-wa**y feature interactions (sums, differences, products and quotients)

<https://www.kaggle.com/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy>

相关性是很重要的东西



# Preprocess

1. 缺失值的处理：
   1. 取平均太简单，做回归又浪费时间(很少数量1，2个的情况下就可以做类别众数，离散均值)
   2. 最好是找到一个相关性比较强的变量，在它的基础上做跳板来做平均值预测

<https://www.kaggle.com/arthurtok/introduction-to-ensembling-stacking-in-python>

* 1. 可以用随机数，拟合分布，起码比用一个数字要好

1. 关于分类变量的处理（哑还是连续）

<https://www.kaggle.com/arthurtok/introduction-to-ensembling-stacking-in-python> （的评论中说到）

* 1. 如果这个变量的类别是有序的，那么可以映射为1234,如果这个变量的类别是无序的，那么则应该映射为哑变量
  2. 对于有经验的人解释道：其实这没有办法很好的评价，所以你应该做两套方案，然后对这两个方案的结果再进行混合

# Feature Engineering

<https://www.kaggle.com/aashita/xgboost-model-with-minimalistic-features>

1. 样本数量：892，样本数量过少会导致选择复杂的模型出现过拟合的情况
2. 相关性分析：

对于决策树来说，变量的相关性是没有没关系，但是对于其他的很多模型都是会影响其表现的。这时候应该用PCA来提取最后的无关变量 。

# Feature Selection

<http://blog.kaggle.com/2016/04/08/homesite-quote-conversion-winners-write-up-1st-place-kazanova-faron-clobber/>

特征选取

In forward selection you start with your null model and add predictors. In backward selection you start with a full model including all your variables and then you drop those you do not need/ are not significant 1 at a time.

* **Forward Selection & Backwards Elimination:** either via brute-force if feasible or greedy in conjunction with feature rankings.
* **Single Feature AUC**: by training a model like XGBoost on each feature seperately or calculating gini coefficients on binned versions of the features.

 In general, it turned out to be more useful to train on many different combinations of feature subsets and feature representations than to tune hyperparameters, which is why several (base) models share the same parameter settings.(多用特征组合比起调整参数要有用的多，所以在大规模的模型作第一层的时候没有浪费时间在调整每一个模型的参数 )

<https://www.cnblogs.com/stevenlk/p/6543628.html>

* **特征是否发散**：如果一个特征不发散，例如方差接近于0，也就是说样本在这个特征上基本上没有差异，这个特征对于样本的区分并没有什么用。
* **特征与目标的相关性**：这点比较显见，与目标相关性高的特征，应当优选选择。除移除低方差法外，本文介绍的其他方法均从相关性考虑。

**这篇文章必须要看**

**Pearson很棒，但是也有一些非线性的缺陷**

**前面用filter中间用wrapper最后用embbeded**

# Model training

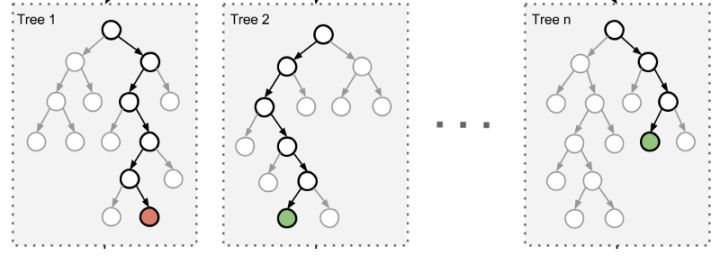
<https://zhuanlan.zhihu.com/p/36161812>

对应的三种情况，可以结合训练集预测结果来看需要什么

* 用于减少方差的bagging
* 用于减少偏差的boosting
* 用于提升预测结果的stacking

在随机森林中，每个树模型都是装袋采样训练的。另外，特征也是随机选择的，最后对于训练好的树也是随机选择的。

这种处理的结果是随机森林的偏差增加的很少，而由于弱相关树模型的平均，方差也得以降低，最终得到一个方差小，偏差也小的模型。



在一个极端的随机树算法中，随机应用的更为彻底：训练集分割的阈值也是随机的，即每次划分得到的训练集是不一样的。这样通常能够进一步减少方差，但是会带来偏差的轻微增加。

<https://www.kaggle.com/arthurtok/introduction-to-ensembling-stacking-in-python>

### ensembling-stacking

第一层模型训练：

·However one cannot simply train the base models on the full training data, generate predictions on the full test set and then output these for the second-level training。（必须使用K-flod的方法，即每次只使用1个K来训练，叫做OOF方法）use from 3 to 10 folds.

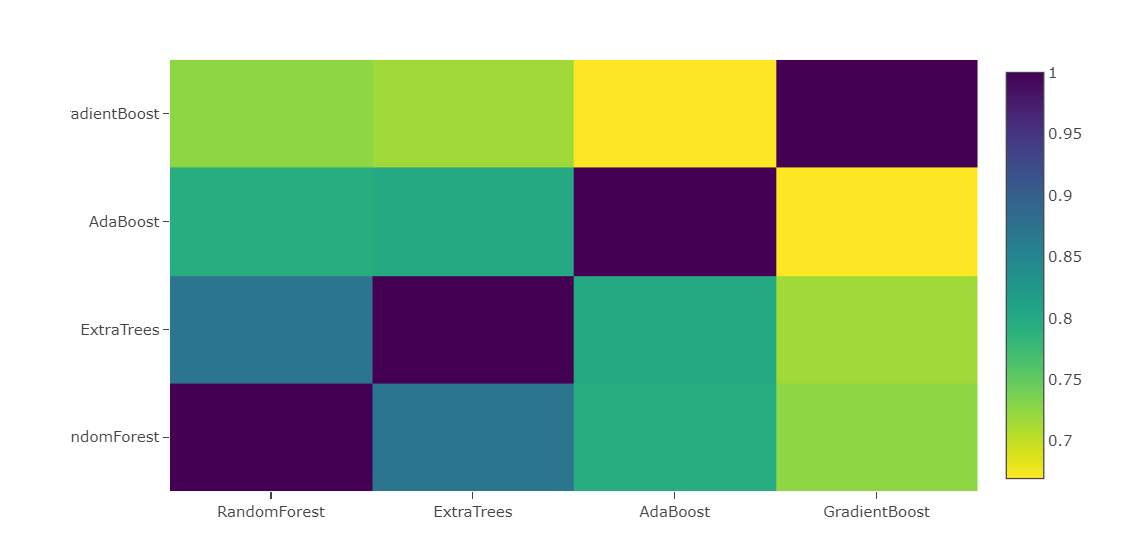
第二层模型训练：

可以看到，他们是以第一层模型的训练结果作为输入



要注意哦，最后预测不是在比赛给的数据上了，而是将比赛的数据经过一层输出的结果哦

然后一定要看热度图，他们之间越不相关，效果就会越好



提升的trick：

1. 构造更多的第一层输出模型
2. 第一层输出之间越不相关，可以学习的东西就越多
3. 输出概率要好过输出类别，连续变量的效果更好

<https://github.com/vecxoz/vecstack/blob/master/examples/00_stacking_concept_pictures_code.ipynb>

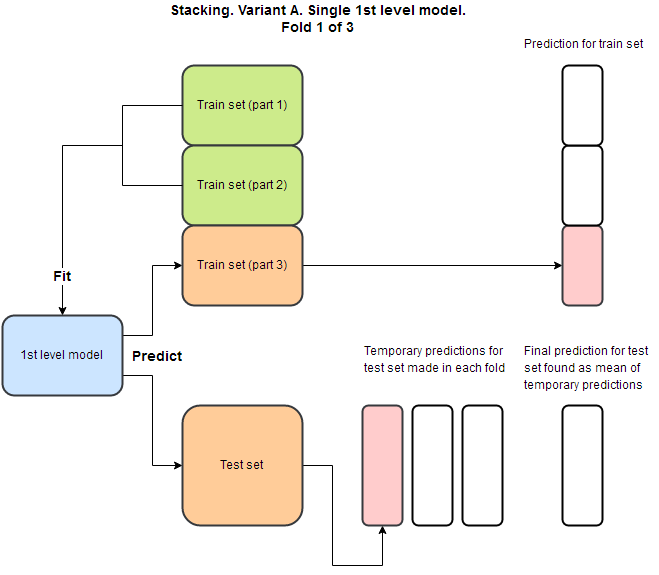
非常详细的流程图

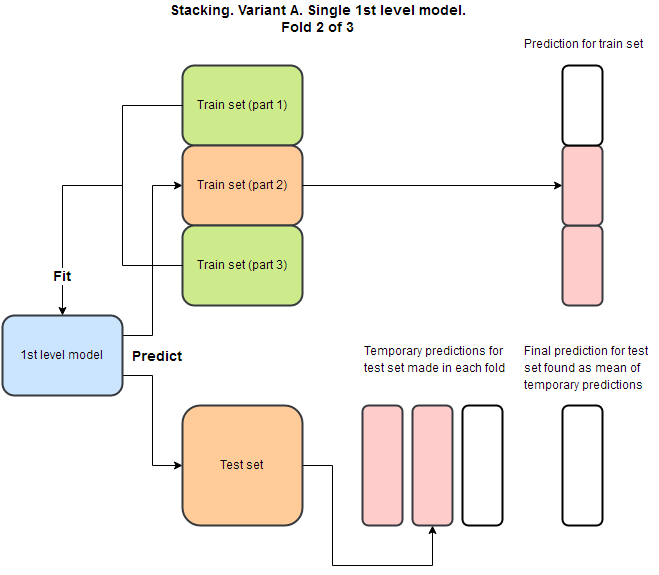
讲解一下：

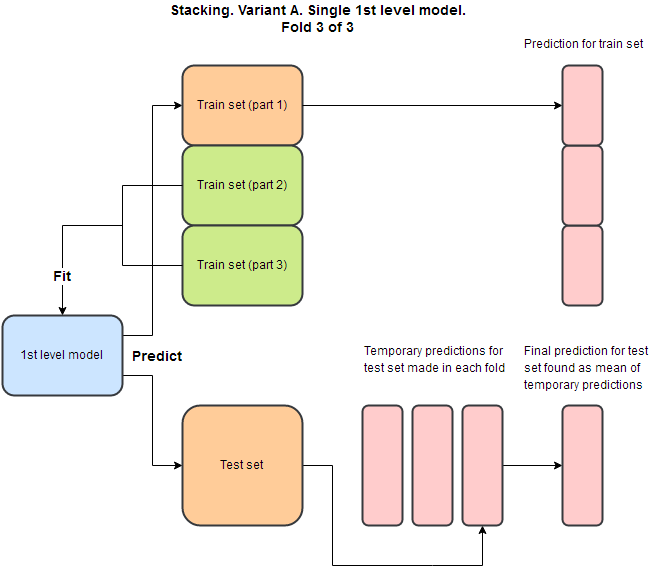
1. 首先下图只是一个基模型的过程，不是第一层的过程
2. 为什么需要OOF，因为如果直接X\_train fit y\_train这样子输出的X\_train的结果会过拟合，因为你已经fit了y\_train，所以才需要在X\_train中切分验证集和训练集，这样predict验证集就不会包含任何之前fit的信息，避免了过拟合
3. 加入train[100,1],test[35,1]使用3个模型

那么第一层的输出就是[100，3]，然后和[100,1]继续fit二层模型

最终用来预测的是第一层的输入[35,3]，然后predict输出最后的[35,1]







**How many model should I built?**

* Reasonable starting point:
  + L1: 2-10 models -> L2: weighted (rank) average or single model
* Then try to add more 1st level models and additional level:
  + L1: 10-50 models -> L2: 2-10 models -> L3: weighted (rank) average
* If you're crunching numbers at Kaggle and decided to go wild:
  + L1: 100-inf models -> L2: 10-50 models -> L3: 2-10 models -> L4: weighted (rank) average

Weighted(rank) average就是指线性加权平均

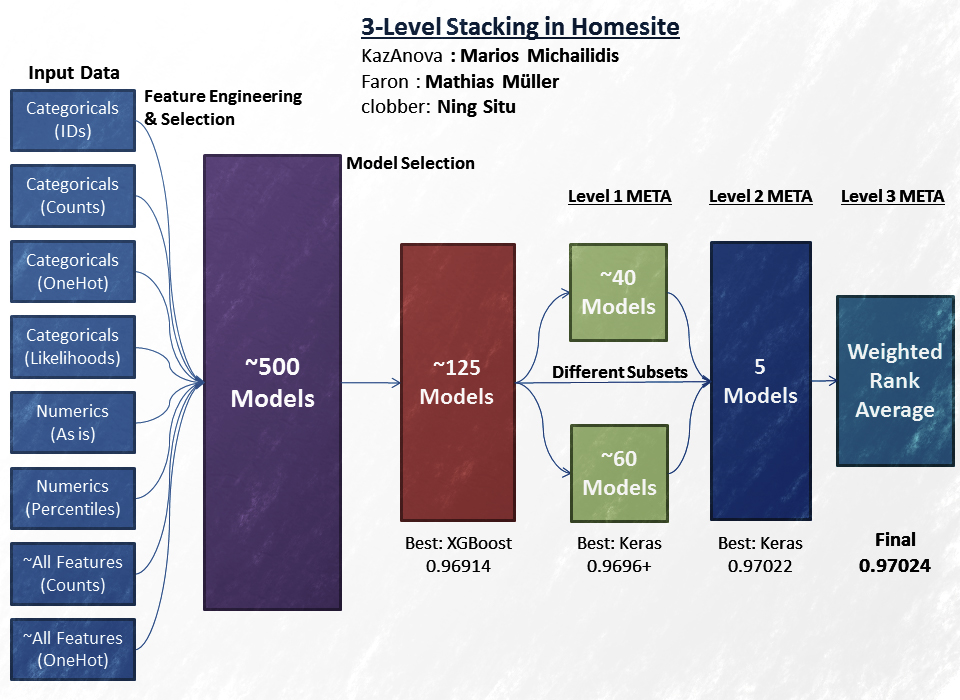
**How many model will I built?**

* Variant A: n\_models\_total = n\_estimators \* n\_folds
* Variant B: n\_models\_total = n\_estimators \* n\_folds + n\_estimators

**How many fold should I choose?**

* Standard approach: 4 or 5 folds.
* If data is big: 3 folds.
* If data is small: you can try more folds like 10 or so.

http://blog.kaggle.com/2016/04/08/homesite-quote-conversion-winners-write-up-1st-place-kazanova-faron-clobber/



# Model selection

<https://baijiahao.baidu.com/s?id=1609468038993670652&wfr=spider&for=pc>

自动机器学习

<https://blog.csdn.net/han_xiaoyang/article/details/52665396>

固定调参法

# Tuning and Evaluation

调参模板

https://www.kaggle.com/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy

# XGBOOST

<https://xgboost.readthedocs.io/en/latest/tutorials/model.html>

简单而详细的介绍了什么是xgboost，一种简单的gradient boosted tree。

Feature Interaction Constraints

When the tree depth is larger than one, many variables interact on the sole basis of minimizing training loss, and the resulting decision tree may capture a spurious relationship (noise) rather than a legitimate relationship that generalizes across different datasets.

也就是说，为了最小化损失，可能会导致一些无关的变量有了相关性，所以这时候就需要对哪些变量可以相互影响做出规定

Control Overfitting

When you observe high training accuracy, but low test accuracy, it is likely that you encountered overfitting problem.

Handle Imbalanced Dataset

For common cases such as ads clickthrough log, the dataset is extremely imbalanced. This can affect the training of XGBoost model, and there are two ways to improve it.

If you care only about the overall performance metric (AUC) of your prediction

Balance the positive and negative weights via scale\_pos\_weight

Use AUC for evaluation

If you care about predicting the right probability

In such a case, you cannot re-balance the dataset

Set parameter max\_delta\_step to a finite number (say 1) to help convergence

There are in general two ways that you can control overfitting in XGBoost:

The first way is to directly control model complexity.

This includes max\_depth, min\_child\_weight and gamma.

The second way is to add randomness to make training robust to noise.

This includes subsample and colsample\_bytree.

You can also reduce stepsize eta. Remember to increase num\_round when you do so.

参数说明：（用的时候到官网上看清楚）

##### 常规参数

nthread：需要多少个线程来训练，默认表示当前所有可用线程

silent：0表示打印过程信息，1表示不打印信息，默认为0

booster ：gbtree和dart使用tree model,gblinear用linear function

##### 用于 Tree 提升的参数

Eta：xgboost在进行完一次迭代后，会将叶子节点的权重乘上该系数，主要是为了削弱每棵树的影响，让后面有更大的学习空间。实际应用中，一般把eta设置得小一点，然后迭代次数设置得大一点，越小越慢，越好。默认0.3

Gamma：在节点分裂时，只有在分裂后损失函数的值下降了，才会分裂这个节点。Gamma指定了节点分裂所需的最小损失函数下降值。这个参数值越大，算法越不易过拟合。默认0

max\_depth：树的最大深度，越深越复杂，越易过拟合。默认6

min\_child\_weight ：决定最小叶子节点样本权重和。当它的值较大时，可以避免模型学习到局部的特殊样本。但如果这个值过高，会导致欠拟合。这个参数需要用cv来调整 。默认1

max\_delta\_step：0表示无限制，如果是正值，则表示模型更conservative。一般不需要该，只有当极度不平衡数据时才需要设置1-10

subsample：这个参数控制对于每棵树，随机采样的比例。减小这个参数的值算法会更加保守，避免过拟合。但是这个值设置的过小，它可能会导致欠拟合。典型值：0.5-1  默认为1

objective：这个参数定义需要被最小化的损失函数。最常用的值有：binary：logistic二分类的逻辑回归，返回预测的概率非类别。multi:softmax使用softmax的多分类器，返回预测的类别。在这种情况下，你还要多设置一个参数：num\_class类别数目。[默认是reg：linear]

 eval\_metric：对于回归问题，默认值是rmse，对于分类问题，默认是error。典型值有：rmse均方根误差；mae平均绝对误差；logloss负对数似然函数值；error二分类错误率；merror多分类错误率；mlogloss多分类损失函数；auc曲线下面积

**n\_jobs** : 需要多少CPU核来训练，-1表示当前电脑所有核

<https://discuss.xgboost.ai/> Xgboost论坛