SLE LAB

Kaggle Pro 银牌计划

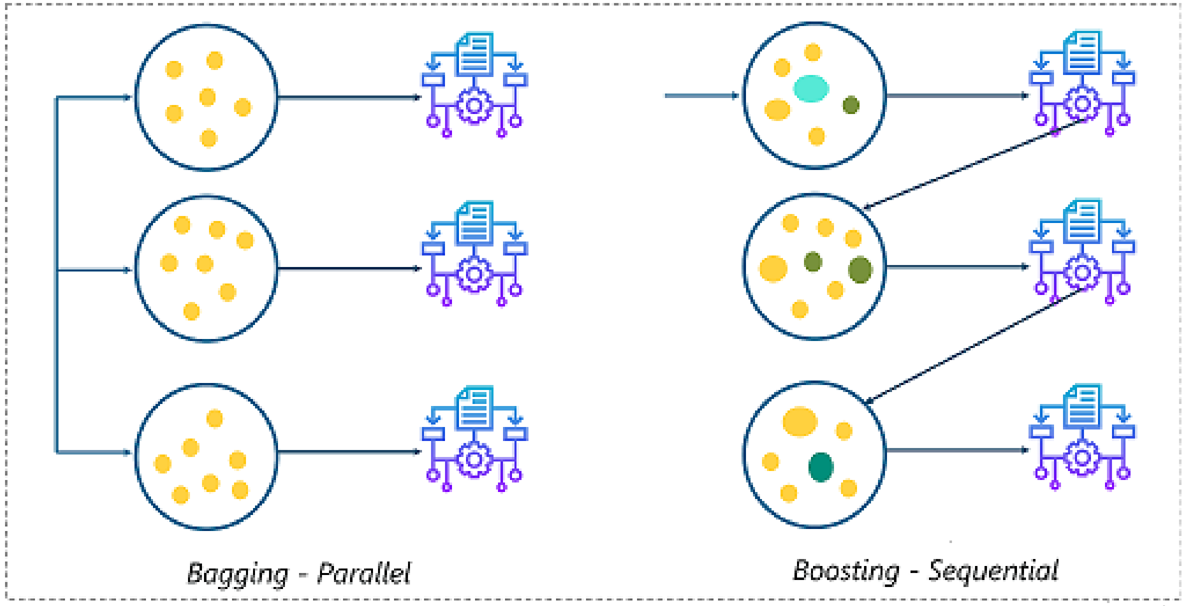
集成树模型与特征工程基础

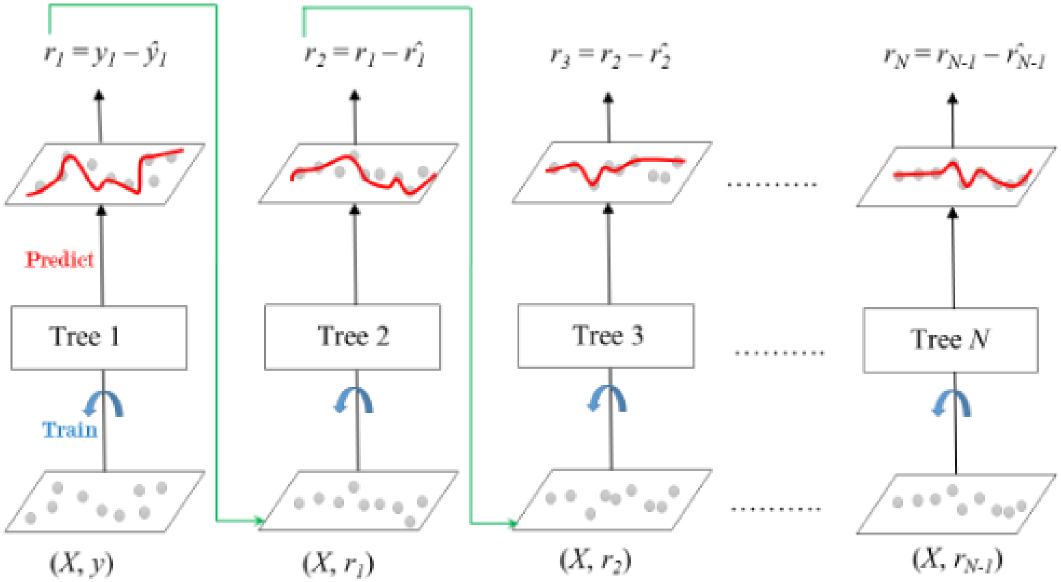
主讲人：黄老师

梯度提升决策树

SLE LAB

Boosting - Gradient Boosting Decision Tree





随机森林

GBDT

XGBoost

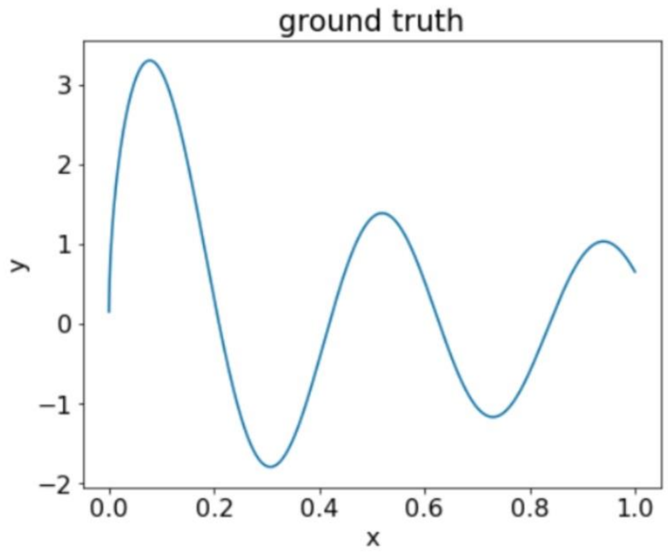
LightGBM

CatBoost

⋮

梯度提升决策树

GBDT流程示意



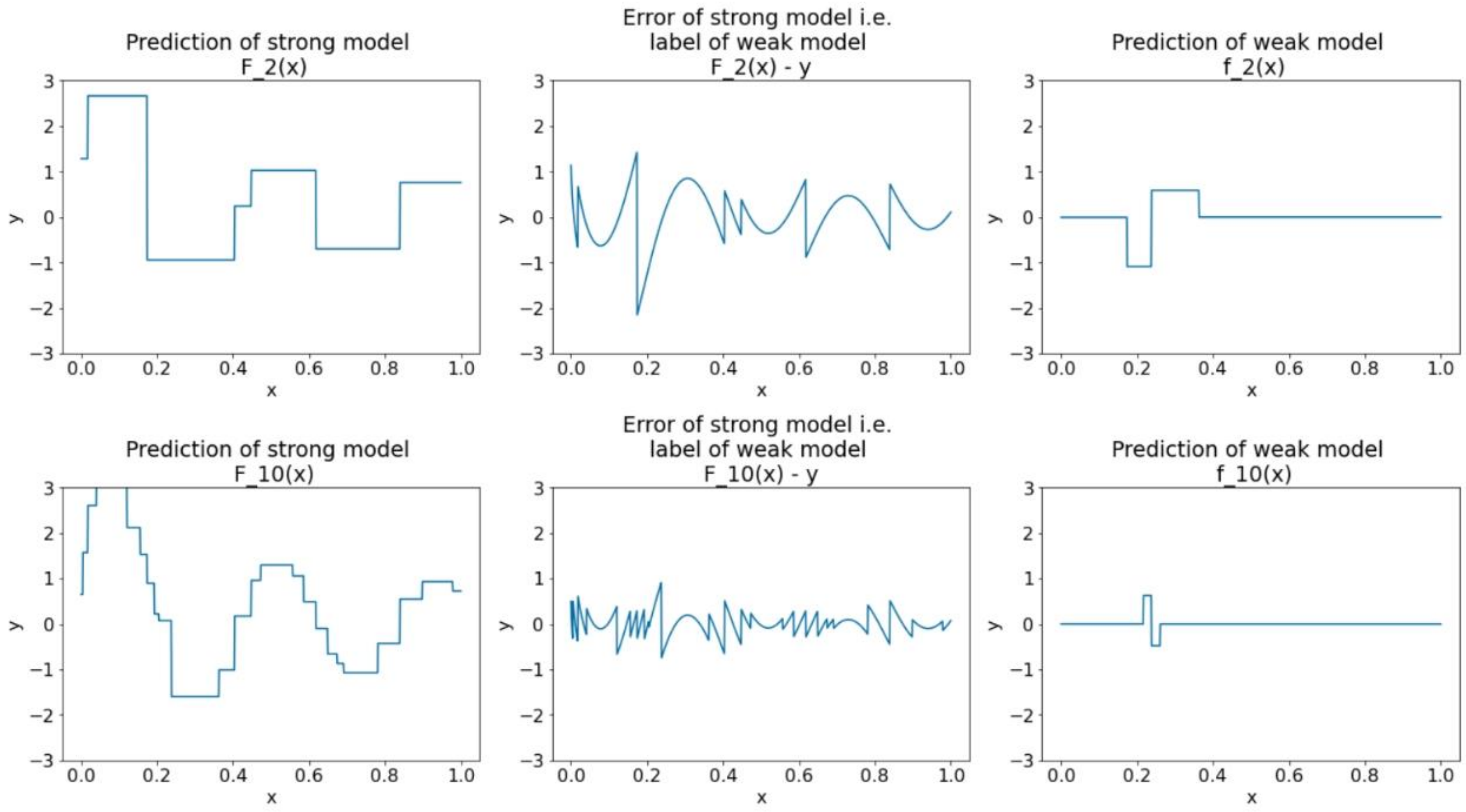


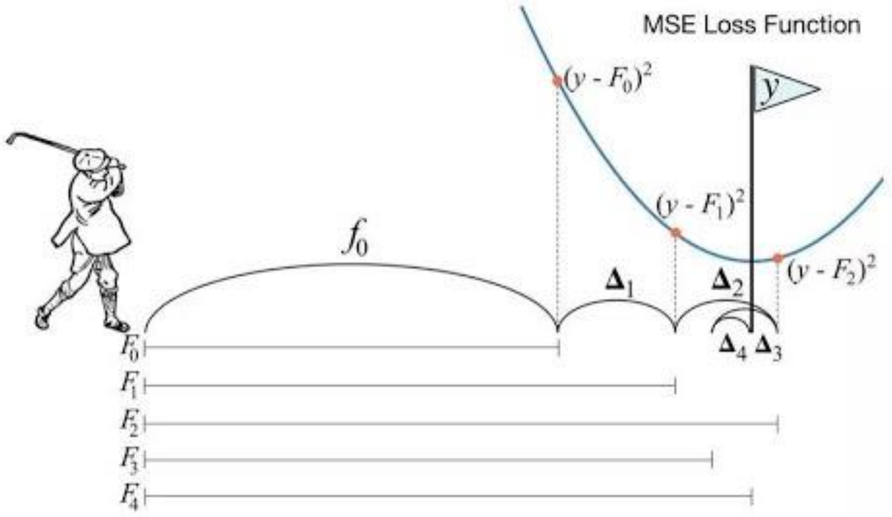
图 28. 第三次迭代和第十次迭代之后的三张图表。

https://developers.google.com/machine-learning/decision-forests/intro-to-gbdt?hl=zh-cn

梯度提升决策树

SLE LAB

原理：高尔夫球



$$\text{Target} : \min L(\mathbf{y}, f(\mathbf{x}))$$

无约束优化迭代法： $f_{n+1}(\mathbf{x}) = f_n(\mathbf{x}) + \Delta_n$

梯度下降法： $\Delta_n = -\alpha \cdot \nabla_{f_n(\mathbf{x})} L(\mathbf{y}, f_n(\mathbf{x}))$

以MSE为例：

$$L = \frac{1}{2}\|\mathbf{y} - f(\mathbf{x})\|_2^2 = \frac{1}{2} f(\mathbf{x})^T f(\mathbf{x}) - \mathbf{y}^T f(\mathbf{x}) + \frac{1}{2}\mathbf{y}^T\mathbf{y}$$

$$-\nabla_{f(\mathbf{x})} L(\mathbf{y}, f(\mathbf{x})) = \boxed{\mathbf{y} - f(\mathbf{x})}$$

GBDT的本质是梯度下降!

- 输入为向量的泰勒级数展开

$$f(\mathbf{x}_k + \delta) \approx f(\mathbf{x}_k) + \mathbf{g}^T(\mathbf{x}_k)\,\delta + \boxed{\frac{1}{2}\delta^T \mathbf{H}(\mathbf{x}_k)\,\delta}$$

XGBoost

SLE LAB

当代数据挖掘算法奠基人

Tianqi Chen
University of Washington
Verified email at cs.washington.edu - Homepage
Machine Learning Systems

Xgboost: A scalable tree boosting system
T Chen, C Guestrin - Proceedings of the 22nd acm sigkdd international ..., 2016 - dl.acm.org
... Tree boosting is a highly effective and widely used machine learning method. In this paper, we describe a scalable endto-end tree boosting system called XGBoost, ... for approximate tree ...
☆ 保存 99 引用 被引用次数：21911 相关文章 所有 42 个版本

XGBoost: A Scalable Tree Boosting System

Tianqi Chen
University of Washington
tqchen@cs.washington.edu

Carlos Guestrin
University of Washington
guestrin@cs.washington.edu

ABSTRACT

Tree boosting is a highly effective and widely used machine learning method. In this paper, we describe a scalable end-to-end tree boosting system called XGBoost, which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges. We propose a novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning. More importantly, we provide insights on cache access patterns, data compression and sharding to build a scalable tree boosting system. By combining these insights, XGBoost scales beyond billions of examples using far fewer resources than existing systems.

Keywords

Large-scale Machine Learning

problems. Besides being used as a stand-alone predictor, it is also incorporated into real-world production pipelines for ad click through rate prediction [15]. Finally, it is the de-facto choice of ensemble method and is used in challenges such as the Netflix prize [3].

In this paper, we describe XGBoost, a scalable machine learning system for tree boosting. The system is available as an open source package[2]. The impact of the system has been widely recognized in a number of machine learning and data mining challenges. Take the challenges hosted by the machine learning competition site Kaggle for example. Among the 29 challenge winning solutions [3] published at Kaggle's blog during 2015, 17 solutions used XGBoost. Among these solutions, eight solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles. For comparison, the second most popular method,

```
# 在此进行参数调节
params = {'objective': 'reg:linear',
          'eta': 0.01,
          'max_depth': 11,
          'subsample': 0.5,
          'colsample_bytree': 0.5,
          'silent': 1,
          'seed': 1
          }
num_trees = 10000
```

XGBoost

数学原理

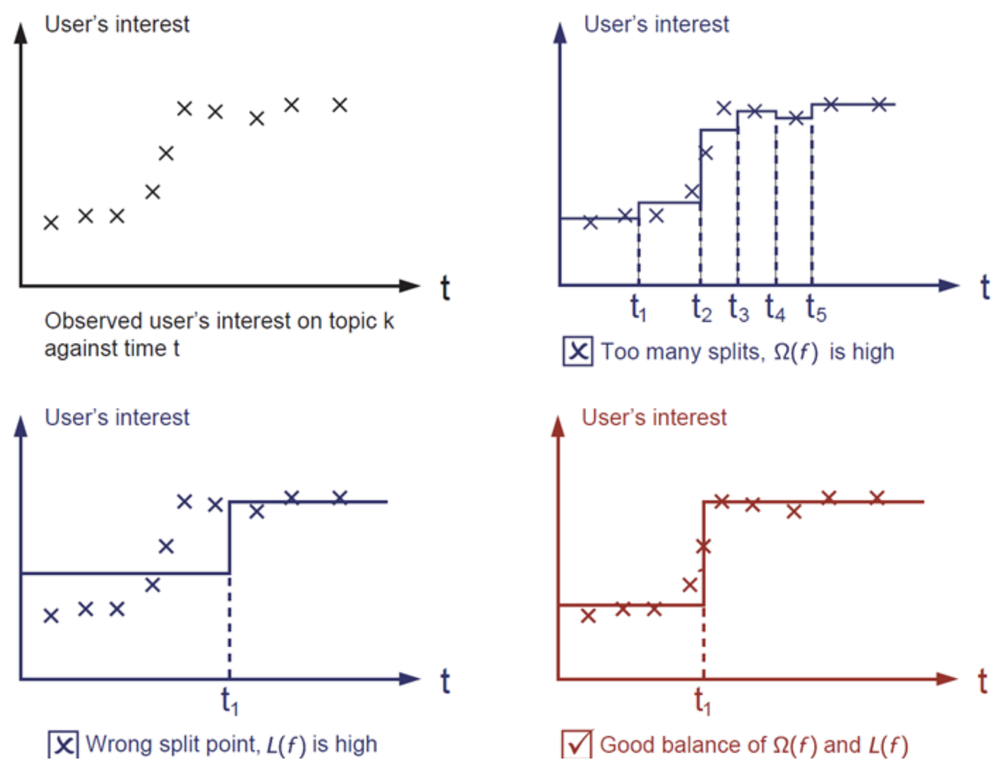
$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i)$$

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)$$

损失函数：

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \boxed{\Omega(f_t)}$$

二阶泰勒展开



XGBoost

SLE LAB

2.2 Gradient Tree Boosting

The tree ensemble model in Eq. (2) includes functions as parameters and cannot be optimized using traditional optimization methods in Euclidean space. Instead, the model is trained in an additive manner. Formally, let $\hat{y}_i^{(t)}$ be the prediction of the i-th instance at the t-th iteration, we will need to add f_t to minimize the following objective.

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

This means we greedily add the f_t that most improves our model according to Eq. (2). Second-order approximation can be used to quickly optimize the objective in the general setting [12].

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ are first and second order gradient statistics on the loss function. We can remove the constant terms to obtain the following simplified objective at step t.

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t) \tag{3}$$

Define $I_j = \{i | q(\mathbf{x}_i) = j\}$ as the instance set of leaf j. We can rewrite Eq (3) by expanding Ω as follows

$$\begin{aligned} \tilde{\mathcal{L}}^{(t)} &= \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T \end{aligned} \tag{4}$$

For a fixed structure $q(\mathbf{x})$, we can compute the optimal weight w_j^* of leaf j by

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}, \tag{5}$$

and calculate the corresponding optimal value by

$$\boxed{\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T.} \tag{6}$$

XGBoost版本的“基尼指数”

Eq (6) can be used as a scoring function to measure the quality of a tree structure q. This score is like the impurity score for evaluating decision trees, except that it is derived for a wider range of objective functions. Fig. 2 illustrates how this score can be calculated.

Normally it is impossible to enumerate all the possible tree structures q. A greedy algorithm that starts from a single leaf and iteratively adds branches to the tree is used instead. Assume that I_L and I_R are the instance sets of left and right nodes after the split. Lettting $I = I_L \cup I_R$, then the loss reduction after the split is given by

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda}\right] - \gamma \tag{7}$$

代码模板

SLE LAB

```
# 在此进行参数调节
params = {'objective': 'reg:linear',
          'eta': 0.01,
          'max_depth': 11,
          'subsample': 0.5,
          'colsample_bytree': 0.5,
          'silent': 1,
          'seed': 1
          }
num_trees = 10000
```

```
dtrain = xgb.DMatrix(X_train[features], np.log1p(X_train.Sales))
dvalid = xgb.DMatrix(X_test[features], np.log1p(X_test.Sales))
dtest = xgb.DMatrix(test[features])

watchlist = [(dtrain, 'train'),(dvalid, 'eval')]
gbm = xgb.train(params, dtrain, num_trees, evals=watchlist, early_stopping_rounds=50, feval=rms
pe_xg, verbose_eval=False)
```

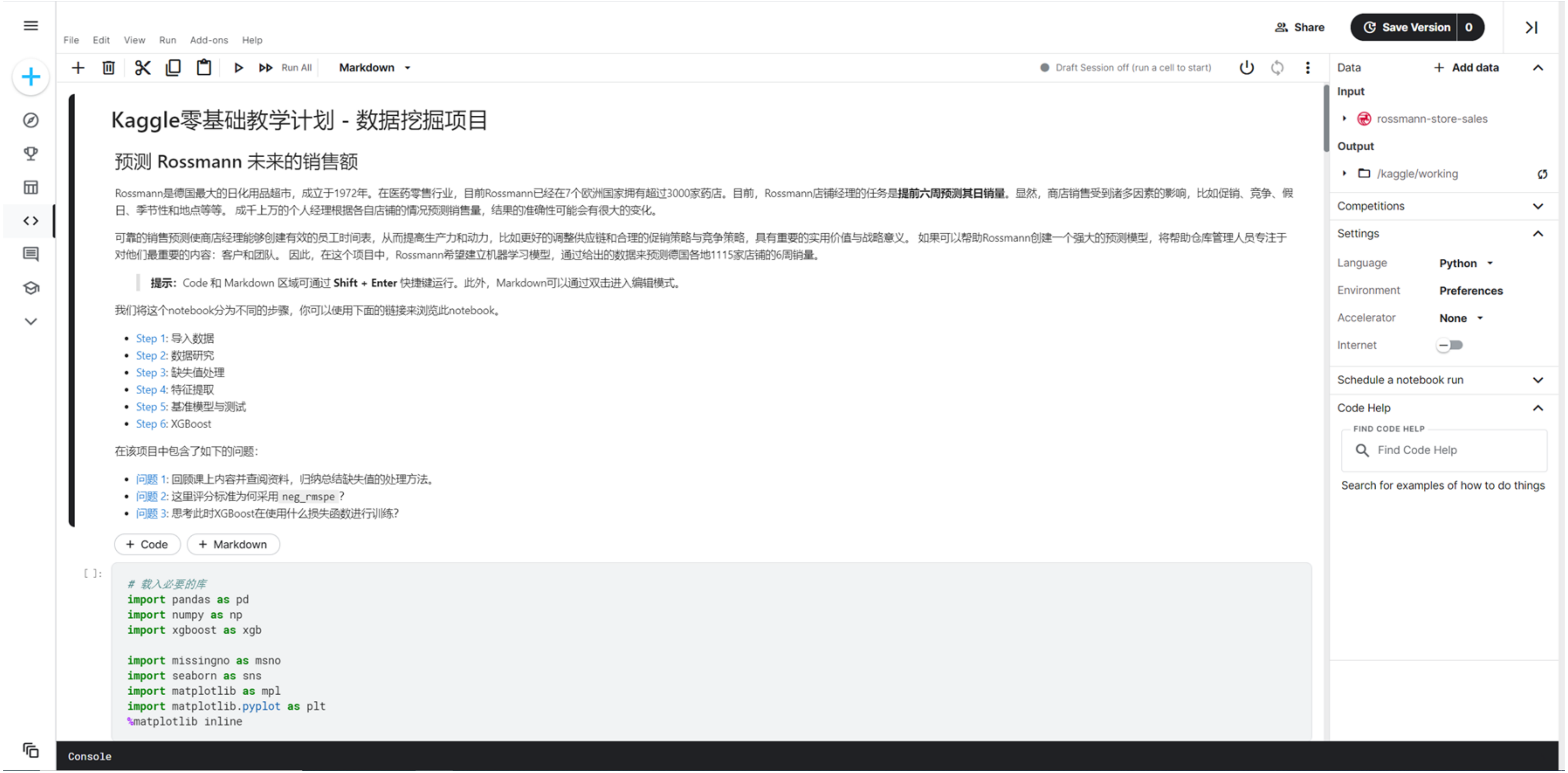
参考链接：

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

https://xgboost.readthedocs.io/en/latest/parameter.html

https://www.kaggle.com/code/sunlightsedu/sunlightsedu-rossmann

代码模板

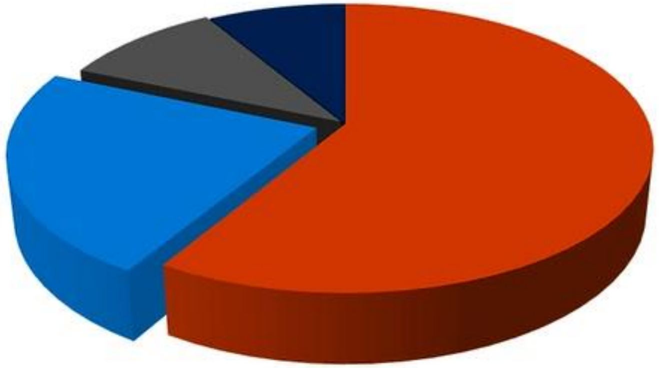

File Edit View Run Add-ons Help
Share
Save Version 0
Run All
Markdown
Draft Session off (run a cell to start)
Kaggle零基础教学计划 - 数据挖掘项目
预测 Rossmann 未来的销售额
Rossmann是德国最大的日化用品超市，成立于1972年。在医药零售行业，目前Rossmann已经在7个欧洲国家拥有超过3000家药店。目前，Rossmann店铺经理的任务是提前六周预测其日销量。显然，商店销售受到诸多因素的影响，比如促销、竞争、假日、季节性和地点等等。成千上万的个人经理根据各自店铺的情况预测销售量，结果的准确性可能会有很大的变化。
可靠的销售预测使商店经理能够创建有效的员工时间表，从而提高生产力和动力，比如更好的调整供应链和合理的促销策略与竞争策略，具有重要的实用价值与战略意义。如果可以帮助Rossmann创建一个强大的预测模型，将帮助仓库管理人员专注于对他们最重要的内容：客户和团队。因此，在这个项目中，Rossmann希望建立机器学习模型，通过给出的数据来预测德国各地1115家店铺的6周销量。
提示：Code 和 Markdown 区域可通过 Shift + Enter 快捷键运行。此外，Markdown可以通过双击进入编辑模式。
我们将这个notebook分为不同的步骤，你可以使用下面的链接来浏览此notebook。
Step 1: 导入数据
Step 2: 数据研究
Step 3: 缺失值处理
Step 4: 特征提取
Step 5: 基准模型与测试
Step 6: XGBoost
在该项目中包含了如下的问题：
问题 1: 回顾课上内容并查阅资料，归纳总结缺失值的处理方法。
问题 2: 这里评分标准为何采用 neg_rmspe？
问题 3: 思考此时XGBoost在使用什么损失函数进行训练？
+ Code
+ Markdown
[ ]:
# 载入必要的库
import pandas as pd
import numpy as np
import xgboost as xgb
import missingno as msno
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
Console
Data
Add data
Input
rossmann-store-sales
Output
/kaggle/working
Competitions
Settings
Language
Python
Environment
Preferences
Accelerator
None
Internet
Schedule a notebook run
Code Help
FIND CODE HELP
Find Code Help
Search for examples of how to do things

LightGBM

SLE LAB

LightGBM

最近微软DMTK团队在github上开源了性能超越其他boosting decision tree工具LightGBM，三天之内star了1000+次，fork了200+次。知乎上有近千人关注“如何看待微软开源的LightGBM？”问题，被评价为“**速度惊人**”，“**非常有启发**”，“**支持分布式**”，“**代码清晰易懂**”，“**占用内存小**”等。

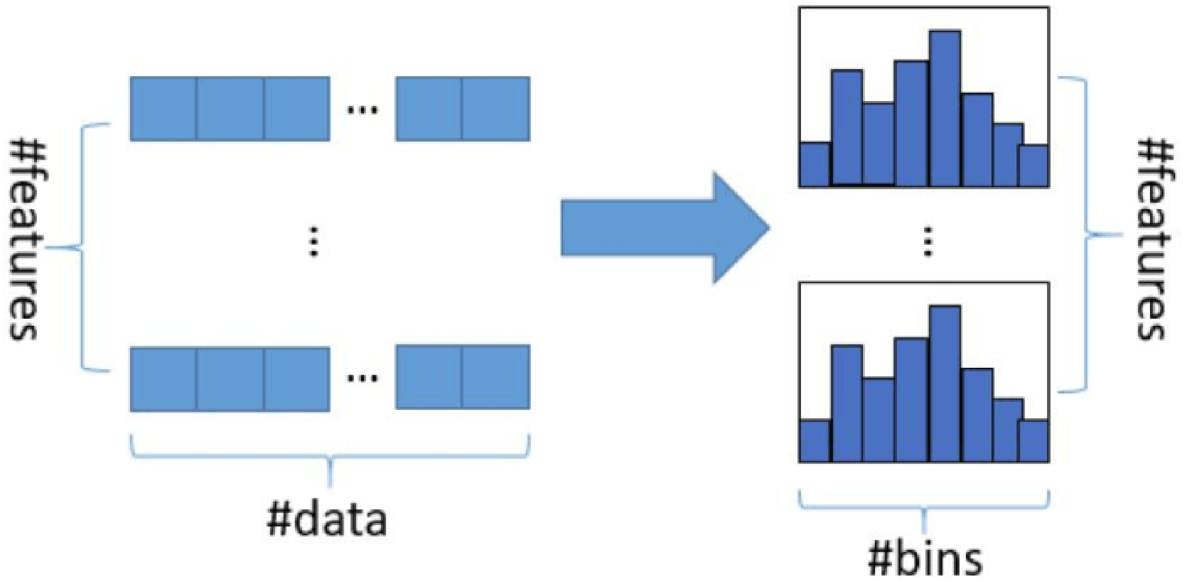


且听来自MSRA开发团队的解读

https://github.com/microsoft/LightGBM

Histogram

把数据的某一个特征转化成一个直方图，然后再以直方图中的某一个bin作为分隔点，计算loss。



相比传统CART分裂方法的优势:

储存空间更小：传统的方法排序的是连续值，而histogram是将连续值离散化，所以离散数据可以用更小的内存来存储。比方说，连续数据可能是4.234252131，但是改成离散值可能就是4.2;

计算复杂度更低：传统方法，需要计算多少次增益呢？特征值乘上样本数量。现在histogram只需要计算特征值乘上直方图bin的数量，一般会设置为一个常数。

Histogram直方图法后来XGB也支持使用了，所以目前来说LGB和XGB都可以用这个方法。

LightGBM

SLE LAB

GOSS

Gradient-based One-Side Sampling，**基于梯度的单边采样。**

简单的说，因为数据太多了，所以从大量数据中采样出一些对训练影响比较大的**大梯度数据**，这样大数据变成小数据，可以提高速度，因为分布相同，所以精度减少的不会多。

Algorithm 2: Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models $\leftarrow$ {}, fact $\leftarrow \frac{1-a}{b}$
topN $\leftarrow$ a $\times$ len(I) , randN $\leftarrow$ b $\times$ len(I)
for $i = 1$ ***to*** d **do**
  preds $\leftarrow$ models.predict(I)
  g $\leftarrow$ $loss$(I, preds), w $\leftarrow$ {1,1,...}
  sorted $\leftarrow$ GetSortedIndices(abs(g))
  topSet $\leftarrow$ sorted[1:topN]
  randSet $\leftarrow$ RandomPick(sorted[topN:len(I)], randN)
  usedSet $\leftarrow$ topSet + randSet
  w[randSet] $\times=$ fact ▷ Assign weight $fact$ to the small gradient data.
  newModel $\leftarrow$ L(I[usedSet], − g[usedSet], w[usedSet])
  models.append(newModel)

- 先根据模型计算出一个预测值preds；
- 计算这个preds和真实值的损失，可能是均方误差或者是其他的，这个损失计算出来；
- 然后对这个损失进行排序，这里需要注意的是，这个损失是一个数组，每一个样本的预测值和真实值都会有一个损失，并不是像是神经网络中的一个batch损失一样把多个样本的损失加和。假设有100个样本，那么就会有100个损失，对这个100个损失进行排序，排序后的数组叫做sorted；
- 选取sorted中前面topN个样本，就是选取100个样本中预测效果最差的topN个样本，叫做**大梯度数据**（large gradient data）
- 然后随机从剩下的预测比较准确的样本（小梯度数据）中选取一些，选取比率就是上面图片中提到的b。假设有100个样本，a为0.2，b为0.3，那么就会让损失最大的20个样本作为大梯度数据，然后在剩下的80个样本中随机选取30个样本作为小梯度数据；
- 将小梯度的样本乘上一个权重系数（1-a）/b，
- 然后用选出取来的大梯度数据和小梯度数据，还有这个权重，来训练一个新的弱学习器。
- 最后把这个弱学习器加到models里面；然后再来一遍整个流程。这里可以看到，在第一步中 根据模型 得到预测值的这个模型，就是models，其实是当前已经训练的所有弱分类器共同得到的一个预测值。

LightGBM

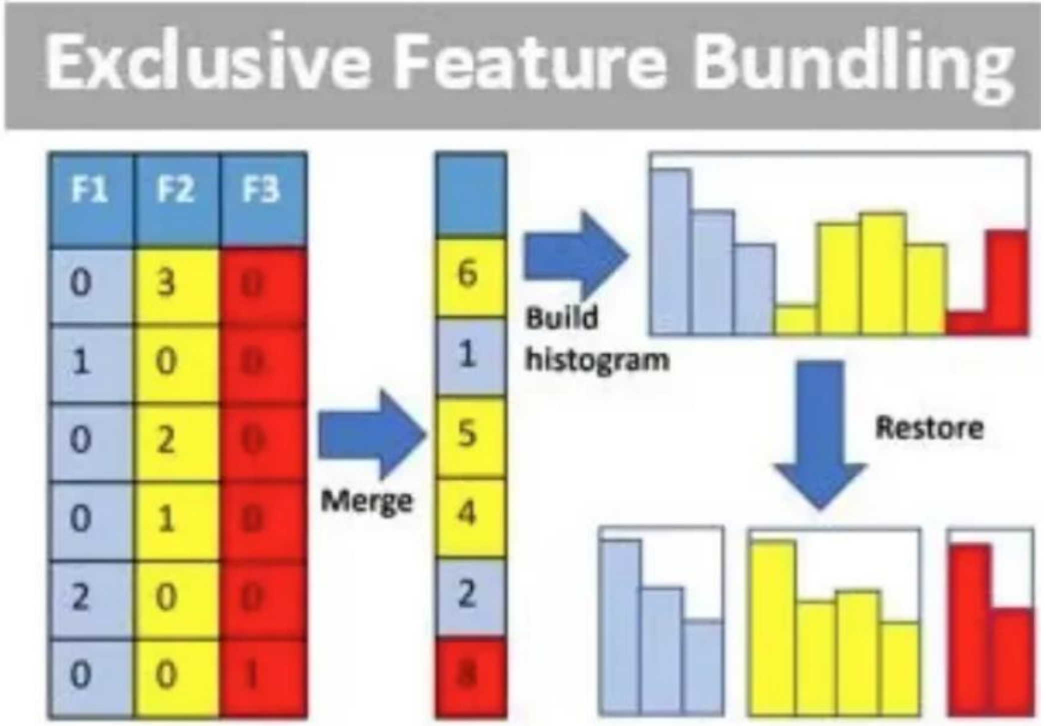
SLE LAB

EBF

Exclusive Feature Bundling **互斥特征捆绑**

LightGBM进行特征抽样，将互斥特征合并，让数据的规模进一步的减小。

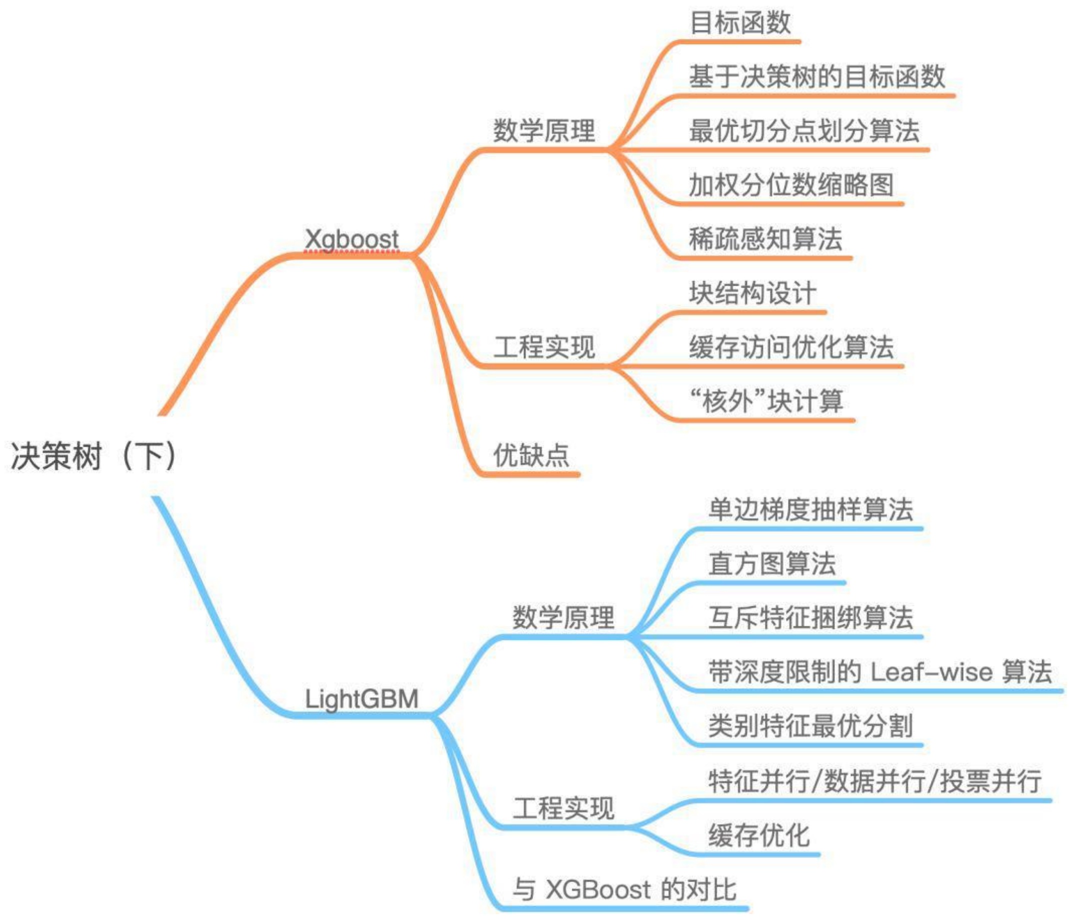
feature1	feature2	feature_bundle
0	2	+ 4 = 6
0	1	+ 4 = 5
0	2	+ 4 = 6
1	0	1
2	0	2
3	0	3
4	0	4



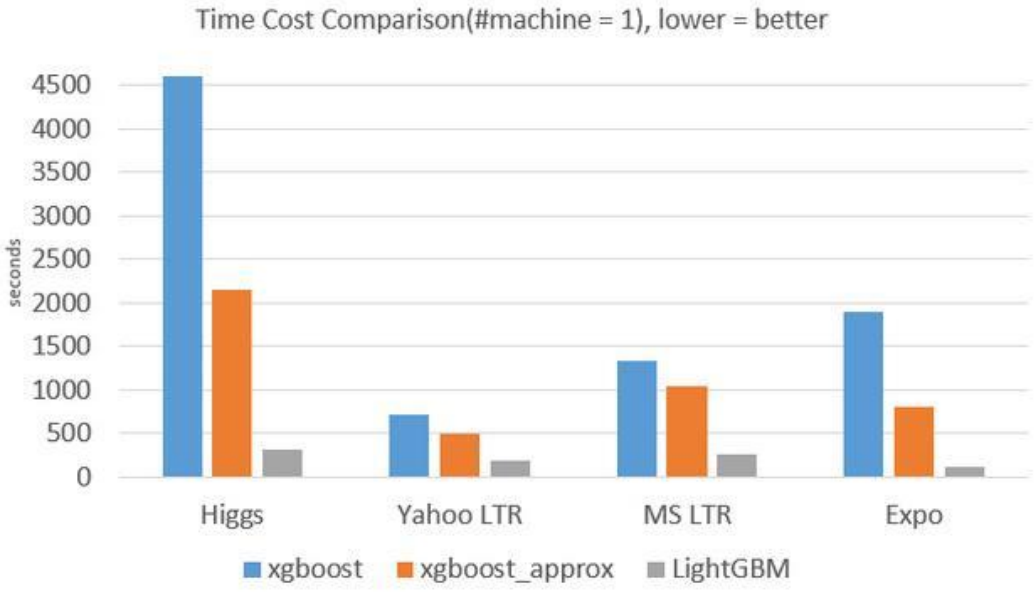
LightGBM

SLE LAB

与XGBoost的比较



训练速度方面



https://www.biaodianfu.com/lightgbm.html

LightGBM两种代码格式

1. 与XGBoost相同的字典格式：

In [17]:
```
# 在此进行参数调节
params = {'objective': 'reg:linear',
          'eta': 0.01,
          'max_depth': 11,
          'subsample': 0.5,
          'colsample_bytree': 0.5,
          'silent': 1,
          'seed': 1
          }
num_trees = 10000
```

6.2 模型训练

In [18]:
```
# 随机划分训练集与验证集
from sklearn.model_selection import train_test_split

X_train, X_test = train_test_split(train, test_size=0.2, random_state=2)

dtrain = xgb.DMatrix(X_train[features], np.log1p(X_train.Sales))
dvalid = xgb.DMatrix(X_test[features], np.log1p(X_test.Sales))
dtest = xgb.DMatrix(test[features])

watchlist = [(dtrain, 'train'),(dvalid, 'eval')]
gbm = xgb.train(params, dtrain, num_trees, evals=watchlist, early_stopping_rounds=50, feval=rmspe_xg, verbose_eval=False)
```

```
/opt/conda/lib/python3.6/site-packages/xgboost/core.py:587: FutureWarning: Series.base is deprecated and will be removed in a future version
  if getattr(data, 'base', None) is not None and \
```

```
params = {
    'task': 'train',
    'boosting_type': 'gbdt',
    'objective': 'regression',
    'metric': 'rmse',
    'num_leaves': 40,
    'subsample':0.8,
    'learning_rate': 0.03,
    'verbose': 1,
    'lambda_l2':3
}

num_trees = 1000
```

```
d_train = lgb.Dataset(train_features, train_target, categorical_feature=categorical_features)
d_eval = lgb.Dataset(test_features,test_target, categorical_feature=categorical_features)
print("Building model meter :",val,'fold:',t)

md = lgb.train(params, d_train, num_boost_round=num_trees, valid_sets=(d_train, d_eval),
               early_stopping_rounds=200,verbose_eval=20)
```

https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html

https://lightgbm.readthedocs.io/en/latest/Parameters.html

代码模板

2. 调用函数格式：

lightgbm.LGBMRegressor

class **lightgbm.LGBMRegressor**(*boosting_type='gbdt', num_leaves=31, max_depth=-1, learning_rate=0.1, n_estimators=100, subsample_for_bin=200000, objective=None, class_weight=None, min_split_gain=0.0, min_child_weight=0.001, min_child_samples=20, subsample=1.0, subsample_freq=0, colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0, random_state=None, n_jobs=None, importance_type='split', **kwargs*) [source]

Bases: `RegressorMixin`, `LGBMModel`

LightGBM regressor.

__init__(*boosting_type='gbdt', num_leaves=31, max_depth=-1, learning_rate=0.1, n_estimators=100, subsample_for_bin=200000, objective=None, class_weight=None, min_split_gain=0.0, min_child_weight=0.001, min_child_samples=20, subsample=1.0, subsample_freq=0, colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0, random_state=None, n_jobs=None, importance_type='split', **kwargs*)

Construct a gradient boosting model.

Parameters:

- **boosting_type** (*str, optional (default='gbdt')*) – 'gbdt', traditional Gradient Boosting Decision Tree. 'dart', Dropouts meet Multiple Additive Regression Trees. 'goss', Gradient-based One-Side Sampling. 'rf', Random Forest.
- **num_leaves** (*int, optional (default=31)*) – Maximum tree leaves for base learners.
- **max_depth** (*int, optional (default=-1)*) – Maximum tree depth for base learners, <=0 means no limit.
- **learning_rate** (*float, optional (default=0.1)*) – Boosting learning rate. You can use `callbacks` parameter of `fit` method to shrink/adapt learning rate in training using `reset_parameter` callback. Note, that this will ignore the `learning_rate` argument in training.
- **n_estimators** (*int, optional (default=100)*) – Number of boosted trees to fit.
- **subsample_for_bin** (*int, optional (default=200000)*) – Number of samples for constructing bins.

```python
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=2019)
clf = LGBMClassifier(
    learning_rate=0.05,
    n_estimators=10000,
    subsample=0.8,
    subsample_freq=1,
    colsample_bytree=0.8,
    random_state=2019
)
amt_oof = np.zeros(train_num)
prob_oof = np.zeros((train_num, 33))
test_pred_prob = np.zeros((test_values.shape[0], 33))
for i, (trn_idx, val_idx) in enumerate(skf.split(train_values, clf_labels)):
    print(i, 'fold...')
    t = time.time()

    trn_x, trn_y = train_values[trn_idx], clf_labels[trn_idx]
    val_x, val_y = train_values[val_idx], clf_labels[val_idx]
    val_repay_amt = amt_labels[val_idx]
    val_due_amt = train_due_amt_df.iloc[val_idx]

    clf.fit(
        trn_x, trn_y,
        eval_set=[(trn_x, trn_y), (val_x, val_y)],
        early_stopping_rounds=100, verbose=5
    )
```

LightGBM

SLE LAB

LightGBM

latest

Search docs

CONTENTS:

- Installation Guide
- Quick Start
- Python Quick Start
- Features
- Experiments
- Parameters
  - Parameters Format
  - Core Parameters
  - Learning Control Parameters
  - IO Parameters
  - Objective Parameters
  - Metric Parameters
  - Network Parameters
  - GPU Parameters
  - Others
- Parameters Tuning
- C API
- Python API
- R API

`blacklist`

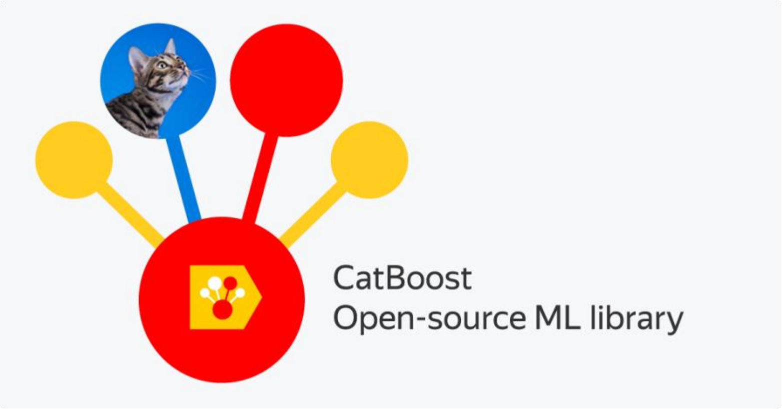
- used to specify some ignoring columns in training
- use number for index, e.g. `ignore_column=0,1,2` means column_0, column_1 and column_2 will be ignored
- add a prefix `name:` for column name, e.g. `ignore_column=name:c1,c2,c3` means c1, c2 and c3 will be ignored
- **Note**: works only in case of loading data directly from text file
- **Note**: index starts from `0` and it doesn't count the label column when passing type is `int`
- **Note**: despite the fact that specified columns will be completely ignored during the training, they still should have a valid format allowing LightGBM to load file successfully

• `categorical_feature`, default = `""`, type = multi-int or string, aliases: `cat_feature`, `categorical_column`, `cat_column`, `categorical_features`

- used to specify categorical features
- use number for index, e.g. `categorical_feature=0,1,2` means column_0, column_1 and column_2 are categorical features
- add a prefix `name:` for column name, e.g. `categorical_feature=name:c1,c2,c3` means c1, c2 and c3 are categorical features
- **Note**: all values will be cast to `int32` (integer codes will be extracted from pandas categoricals in the Python-package)
- **Note**: index starts from `0` and it doesn't count the label column when passing type is `int`
- **Note**: all values should be less than `Int32.MaxValue` (2147483647)
- **Note**: using large values could be memory consuming. Tree decision rule works best when categorical features are presented by consecutive integers starting from zero
- **Note**: all negative values will be treated as **missing values**
- **Note**: the output cannot be monotonically constrained with respect to a categorical feature
- **Note**: floating point numbers in categorical features will be rounded towards 0

```
d_train = lgb.Dataset(X_train, label=y_train, categorical_feature=cat_features)
d_valid = lgb.Dataset(X_valid, label=y_valid, categorical_feature=cat_features)
watchlist = [d_train, d_valid]

print('training LGB:')
model = lgb.train(params,
                  train_set=d_train,
                  num_boost_round=num_rounds,
                  valid_sets=watchlist,
                  verbose_eval=verbose_eval,
                  early_stopping_rounds=early_stop)
```

CatBoost：LightGBM的平替




```
cbr = CatBoostRegressor(n_estimators=100,
                        use_best_model=True,
                        objective='RMSE',
                        loss_function='RMSE',
                        boosting_type='Plain',
                        eval_metric='RMSE',
                        l2_leaf_reg=1,
                        learning_rate=0.03,
                        max_depth=8,
                        task_type = 'CPU'
                       )

#设置分类变量
categorical_features=['building_id', 'site_id', 'primary_use', 'had_air_temperature', 'had_cloud_coverage',
                      'had_dew_temperature', 'had_precip_depth_1_hr','had_sea_level_pressure', 'had_wind_direction',
                      'had_wind_speed', 'tm_day_of_week', 'tm_hour_of_day']
```

- 对GPU支持最好，可多GPU训练
- 接口统一，代码不混乱
- 算法原理区别于XGB和LGB，方便集成

```
train_features = X1.iloc[train_index]
train_target = y[X1.iloc[train_index].index]

test_features = X1.iloc[test_index]
test_target = y[X1.iloc[test_index].index]

print("Building model meter :",val,'fold:',t)
md = cbr.fit(train_features, train_target,eval_set=(test_features, test_target),
             cat_features=categorical_features,early_stopping_rounds=500, verbose=10)
```

https://catboost.ai/docs

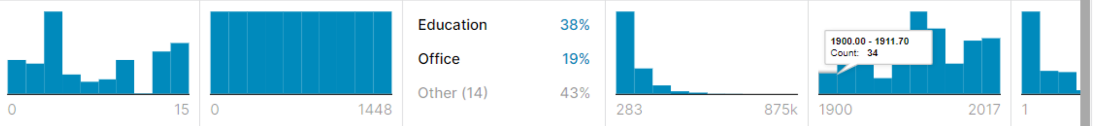
https://www.biaodianfu.com/catboost.html

数据挖掘竞赛通用流程

SLE LAB

1. 数据预处理与特征抽取 （特征工程）
2. 模型训练
3. 结果后处理

building_metadata.csv (44.46 KB)   6 of 6 columns   Views



	site_id	building_id	primary_use	square_feet	year_built	floor_
1	0	0	Education	7432	2008	
2	0	1	Education	2720	2004	
3	0	2	Education	5376	1991	
4	0	3	Education	23685	2002	
5	0	4	Education	116607	1975	
6	0	5	Education	8000	2000	
7	0	6	Lodging/residential	27926	1981	
8	0	7	Education	121074	1989	
9	0	8	Education	60809	2003	
10	0	9	Office	27000	2010	
11	0	10	Entertainment/public assembly	370773	1991	
12	0	11	Education	49073	1968	
13	0	12	Lodging/residential	37100	1999	
14	0	13	Education	99380	2000	
15	0	14	Education	86250	2013	
16	0	15	Office	83957	1974	
17	0	16	Education	54644	1996	

```python
train_df = pd.read_csv('dataset/train.csv', parse_dates=['auditing_date', 'due_date', 'repay_date'])
train_df['repay_date'] = train_df[['due_date', 'repay_date']].apply(
    lambda x: x['repay_date'] if x['repay_date'] != '\\N' else x['due_date'], axis=1
)
train_df['repay_amt'] = train_df['repay_amt'].apply(lambda x: x if x != '\\N' else 0).astype('float32')
train_df['label'] = (train_df['repay_date'] - train_df['auditing_date']).dt.days
train_df.loc[train_df['repay_amt'] == 0, 'label'] = 32
clf_labels = train_df['label'].values
amt_labels = train_df['repay_amt'].values
del train_df['label'], train_df['repay_amt'], train_df['repay_date']
train_due_amt_df = train_df[['due_amt']]
train_num = train_df.shape[0]
test_df = pd.read_csv('dataset/test.csv', parse_dates=['auditing_date', 'due_date'])
sub = test_df[['listing_id', 'auditing_date', 'due_amt']]
df = pd.concat([train_df, test_df], axis=0, ignore_index=True)


listing_info_df = pd.read_csv('dataset/listing_info.csv')
del listing_info_df['user_id'], listing_info_df['auditing_date']
df = df.merge(listing_info_df, on='listing_id', how='left')

# 表中有少数user不止一条记录，因此按日期排序，去重，只保留最新的一条记录。
user_info_df = pd.read_csv('dataset/user_info.csv', parse_dates=['reg_mon', 'insertdate'])
user_info_df.rename(columns={'insertdate': 'info_insert_date'}, inplace=True)
user_info_df = user_info_df.sort_values(by='info_insert_date', ascending=False).drop_duplicates('user_id').reset_index(drop=True)
df = df.merge(user_info_df, on='user_id', how='left')

# 同上
user_tag_df = pd.read_csv('dataset/user_taglist.csv', parse_dates=['insertdate'])
user_tag_df.rename(columns={'insertdate': 'tag_insert_date'}, inplace=True)
user_tag_df = user_tag_df.sort_values(by='tag_insert_date', ascending=False).drop_duplicates('user_id').reset_index(drop=True)
df = df.merge(user_tag_df, on='user_id', how='left')

# 历史记录表能做的特征远不止这些
repay_log_df = pd.read_csv('dataset/user_repay_logs.csv', parse_dates=['due_date', 'repay_date'])
# 由于题目任务只预测第一期的还款情况，因此这里只保留第一期的历史记录。当然非第一期的记录也能提取很多特征。
repay_log_df = repay_log_df[repay_log_df['order_id'] == 1].reset_index(drop=True)
repay_log_df['repay'] = repay_log_df['repay_date'].astype('str').apply(lambda x: 1 if x != '2200-01-01' else 0)
repay_log_df['early_repay_days'] = (repay_log_df['due_date'] - repay_log_df['repay_date']).dt.days
repay_log_df['early_repay_days'] = repay_log_df['early_repay_days'].apply(lambda x: x if x >= 0 else -1)
for f in ['listing_id', 'order_id', 'due_date', 'repay_date', 'repay_amt']:
    del repay_log_df[f]
group = repay_log_df.groupby('user_id', as_index=False)
repay_log_df = repay_log_df.merge(
    group['repay'].agg({'repay_mean': 'mean'}), on='user_id', how='left'
)
repay_log_df = repay_log_df.merge(
    group['early_repay_days'].agg({
        'early_repay_days_max': 'max', 'early_repay_days_median': 'median', 'early_repay_days_sum': 'sum',
        'early_repay_days_mean': 'mean', 'early_repay_days_std': 'std'
    }), on='user_id', how='left'
)
repay_log_df = repay_log_df.merge(
    group['due_amt'].agg({
        'due_amt_max': 'max', 'due_amt_min': 'min', 'due_amt_median': 'median',
        'due_amt_mean': 'mean', 'due_amt_sum': 'sum', 'due_amt_std': 'std',
        'due_amt_skew': 'skew', 'due_amt_kurt': kurtosis, 'due_amt_ptp': np.ptp
    }), on='user_id', how='left'
)
del repay_log_df['repay'], repay_log_df['early_repay_days'], repay_log_df['due_amt']
repay_log_df = repay_log_df.drop_duplicates('user_id').reset_index(drop=True)
df = df.merge(repay_log_df, on='user_id', how='left')

cate_cols = ['gender', 'cell_province', 'id_province', 'id_city']
for f in cate_cols:
    df[f] = df[f].map(dict(zip(df[f].unique(), range(df[f].nunique())))).astype('int32')

df['due_amt_per_days'] = df['due_amt'] / (train_df['due_date'] - train_df['auditing_date']).dt.days
date_cols = ['auditing_date', 'due_date', 'reg_mon', 'info_insert_date', 'tag_insert_date']
for f in date_cols:
```

特征工程

SLE LAB

数据概览：探索性数据分析 （EDA）

Votes	Title	Age	Score	Tags	Output	Language	Comments
799	⚡ASHRAE -Start Here: A GENTLE Introduction	3mo ago		beginner, data cleaning, data visualization, starter code, utility script		Py	220
395	EDA for ASHRAE	5mo ago				Py	50
373	ASHRAE: Half and Half	5mo ago	1.1	beginner, gradient boosting, regression, starter code		Py	73
276	A deep dive EDA into ALL variables	5mo ago		eda, data cleaning, data visualization, feature engineering, starter code		Rmd	41
281	ASHRAE: Training LGBM by meter type	5mo ago	1.121	tutorial, ensembling, feature engineering, gradient boosting, starter code		Py	89
226	ASHRAE- KFold LightGBM - without leak (1.08)	4mo ago	1.08	data cleaning, ensembling, feature engineering, regression, starter code		Py	47
217	Starter EDA and Feature selection ASHRAE3	5mo ago	1.24	data visualization, feature engineering, regression, starter code		Py	29
199	Ashrae: simple data cleanup (LB 1.08 no leaks)	4mo ago	1.08			Py	43

- 了解竞赛数据类型：

  时间戳、字符特征、分类特征 ...
- 数据分布概览
- 特征关联性分析
- 特征重要性判断

  ...

热门竞赛：不需要自己做

特征工程

SLE LAB

常见特征工程方法

1. 数据清洗:
- 缺失值处理：可以选择填充缺失值（如用平均数、中位数、众数等），或者删除含有缺失值的行或列。
- 异常值处理：通过箱线图、Z分数等方法检测并处理异常值。

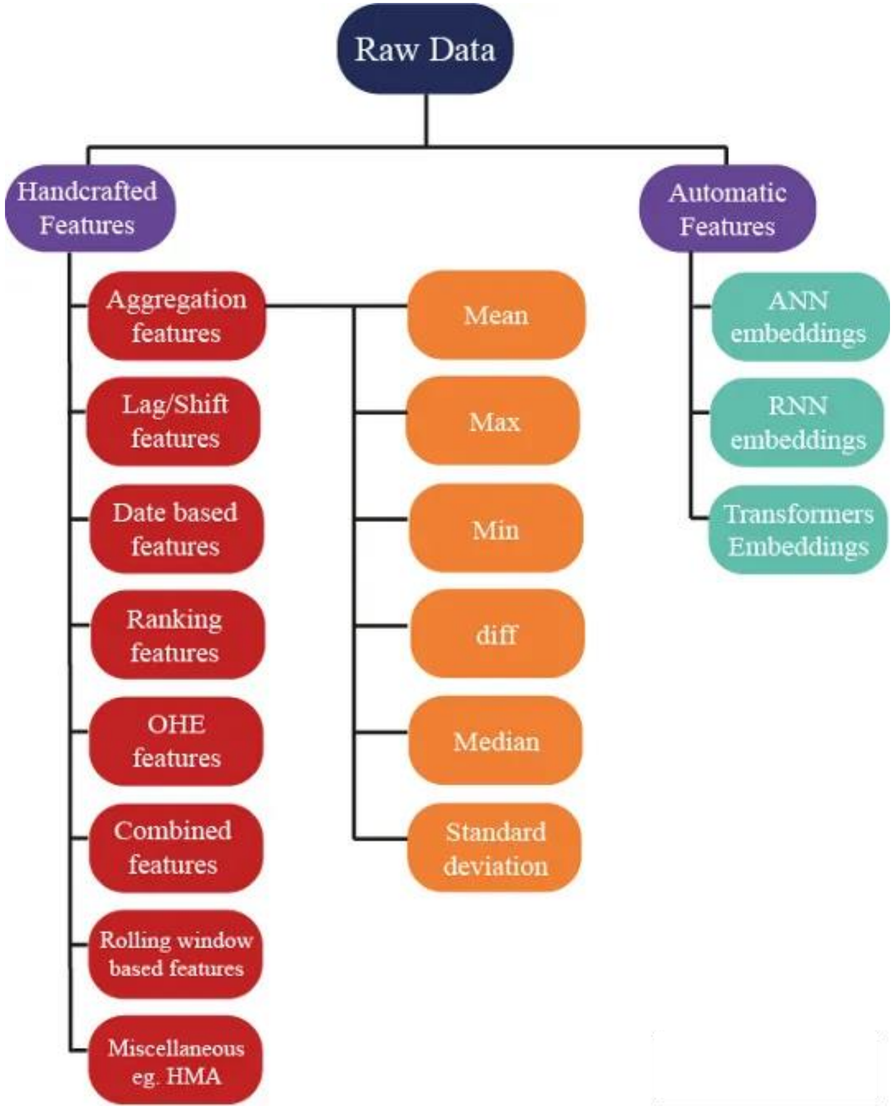
2. 特征选择:
- 过滤法（Filter）：根据统计测试的结果（如相关系数、卡方检验）选择特征。
- 包裹法（Wrapper）：如递归特征消除，通过一系列模型的性能来选择特征。
- 嵌入法（Embedded）：如LASSO回归，利用模型自身的特性选择特征。

3. 特征构造:
- 交互特征：将两个或多个特征相乘或组合，创建新的交互特征。
- 多项式特征：通过特征的高次项和交互项来增加模型的复杂度。

4. 特征转换:
- 标准化/归一化：如Z分数标准化、最小-最大归一化，使得不同量纲的特征具有可比性。
- 离散化/分箱：将连续特征离散化成几个区间，以简化模型的复杂度。
- 对数变换、平方根变换等：用于处理偏态分布的数据，使其更接近正态分布。

5. 特征编码:
- 独热编码（One-Hot Encoding）：将分类变量转换为一系列的二进制列，每个数值对应一个列。
- 标签编码（Label Encoding）：将每个分类分配一个唯一的整数。
- 目标编码（Target Encoding）：根据目标变量的平均值对分类特征进行编码。

6. 特征抽取:
- 文本数据：使用词袋模型、TF-IDF、Word2Vec等方法将文本数据转换为数值型特征。
- 图像数据：使用边缘检测、颜色直方图、深度学习模型（如CNN）来提取图像特征。



http://blog.wya1.com/article/2068800324/details/39029

特征工程

SLE LAB

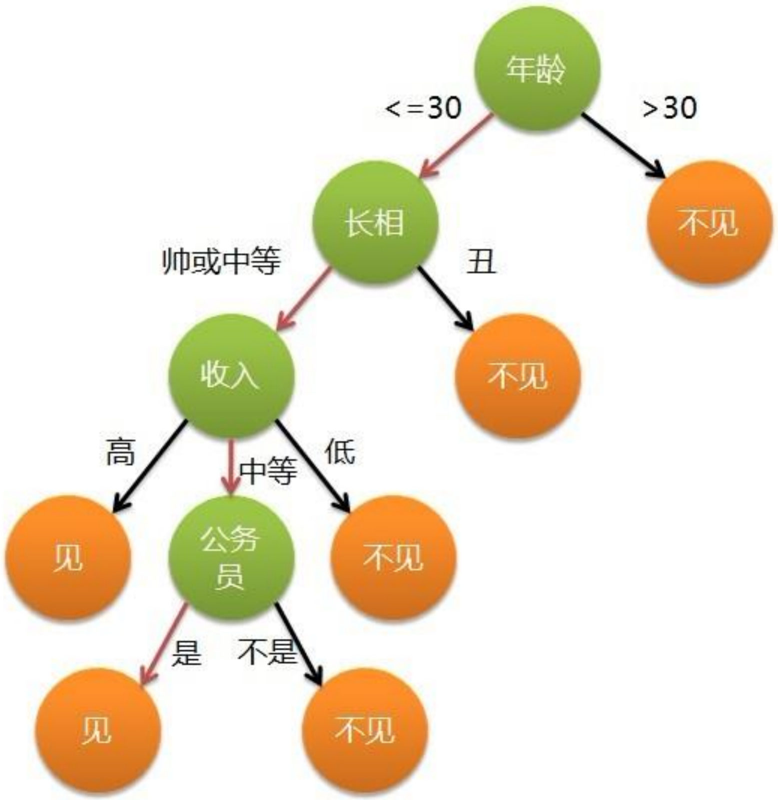
缺失值处理

- 不存在该值
- 尊重资料提供者的意愿
- 无法取得该值

https://zhuanlan.zhihu.com/p/137175585

决策树为什么常用

决策树可以直接处理缺失值



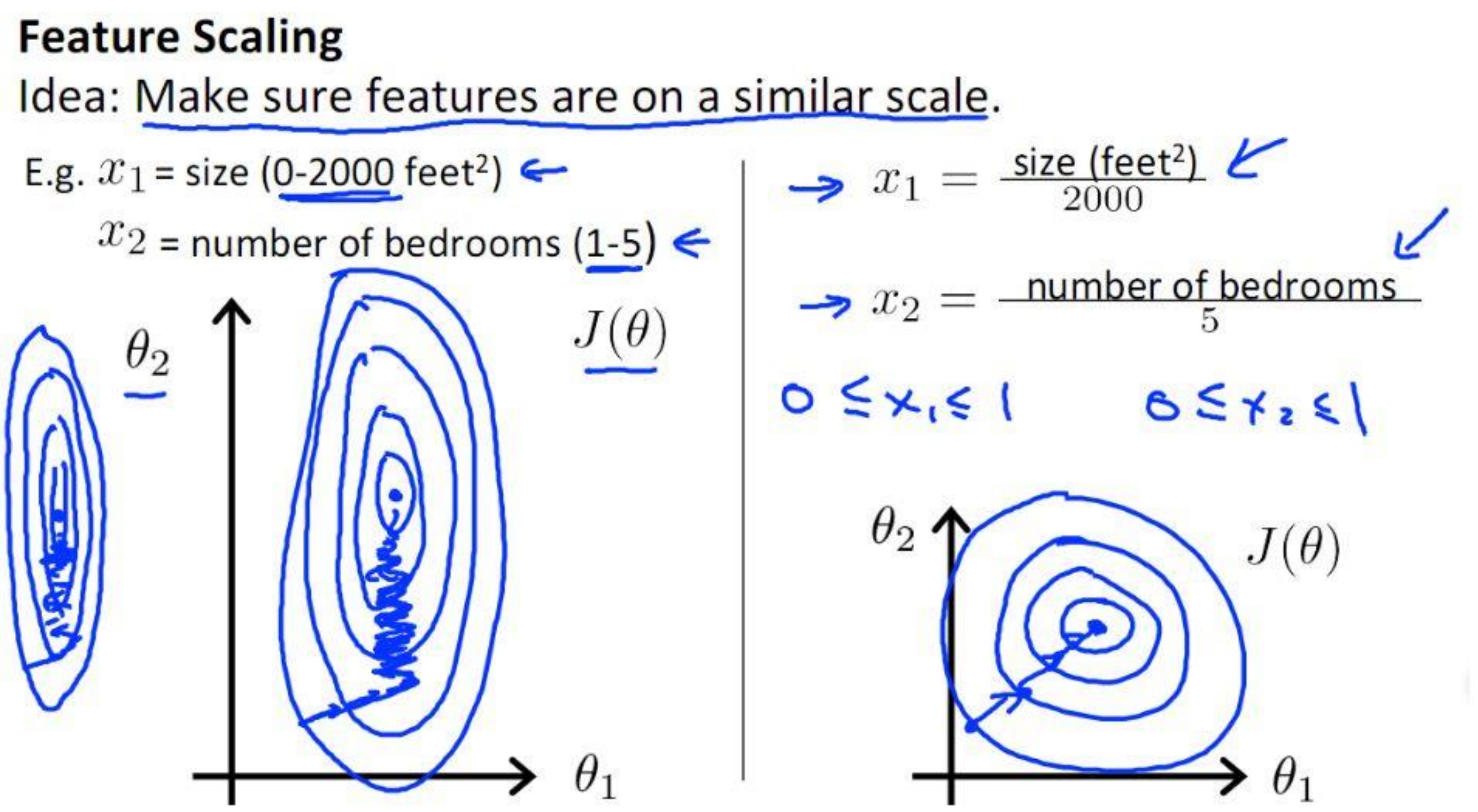
- 对于连续特征，可以直接将缺失值样本分别划分到左边或右边，比较信息增益或基尼指数
- 对于离散特征，可以直接将缺失的样本逐一划分到各个子集中，比较信息增益或基尼指数

TomHall
请填写个人介绍
128 人赞同了该回答
在xgboost里，在每个结点上都会将对应变量是缺失值的数据往左右分支各导流一次，然后计算两种导流方案对Objective的影响，最后认为对Objective降低更明显的方向（左或者右）就是缺失数据应该流向的方向，在预测时在这个结点上将同样变量有缺失值的数据都导向训练出来的方向。

https://www.zhihu.com/question/34867991

特征工程

数值转换



- **Min-Max 转换**

$$x' = \frac{x - x_{min}}{x_{max} - x_{\min}}$$

- **标准化转换**

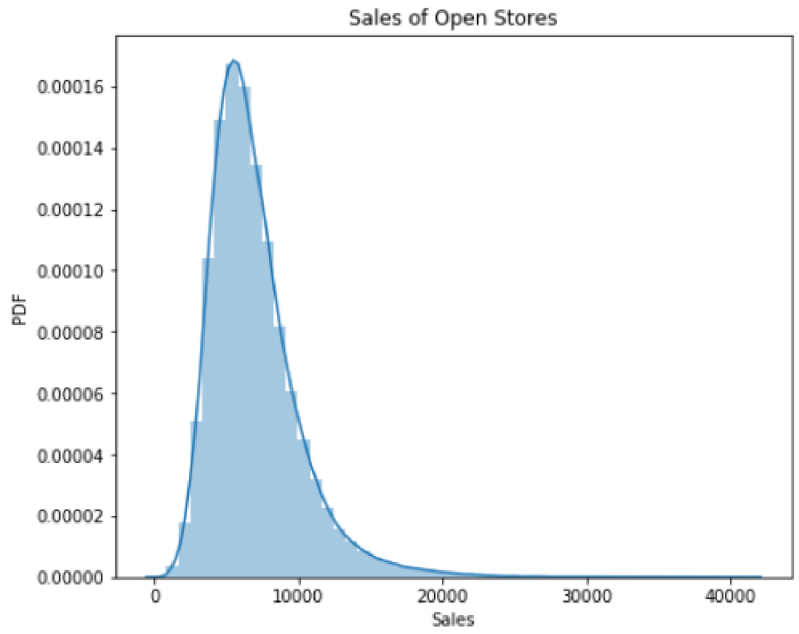
$$x' = \frac{x - \mu}{\sigma}$$

思考：选用决策树算法时，是否需要对特征进行标准化?

数值转换

对数转换：np.log1p() $f(x) = \ln(x+1)$

对于偏度大于0.75的标签数据，一般有必要进行对数转换



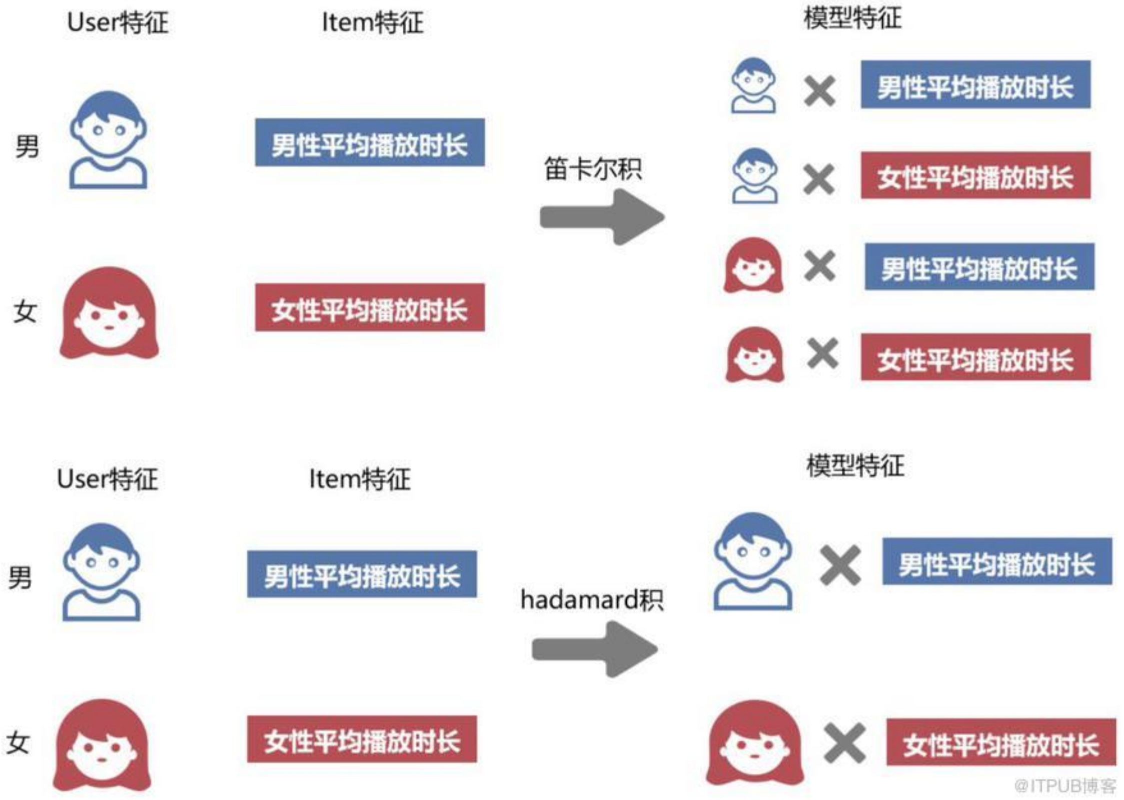

```
# 随机划分训练集与验证集
from sklearn.model_selection import train_test_split

X_train, X_test = train_test_split(train, test_size=0.2, random_state=2)

dtrain = xgb.DMatrix(X_train[features], np.log1p(X_train.Sales))
dvalid = xgb.DMatrix(X_test[features], np.log1p(X_test.Sales))
dtest = xgb.DMatrix(test[features])

watchlist = [(dtrain, 'train'),(dvalid, 'eval')]
gbm = xgb.train(params, dtrain, num_trees, evals=watchlist, early_stopping_rounds=50, feval=rms
pe_xg, verbose_eval=False)
```

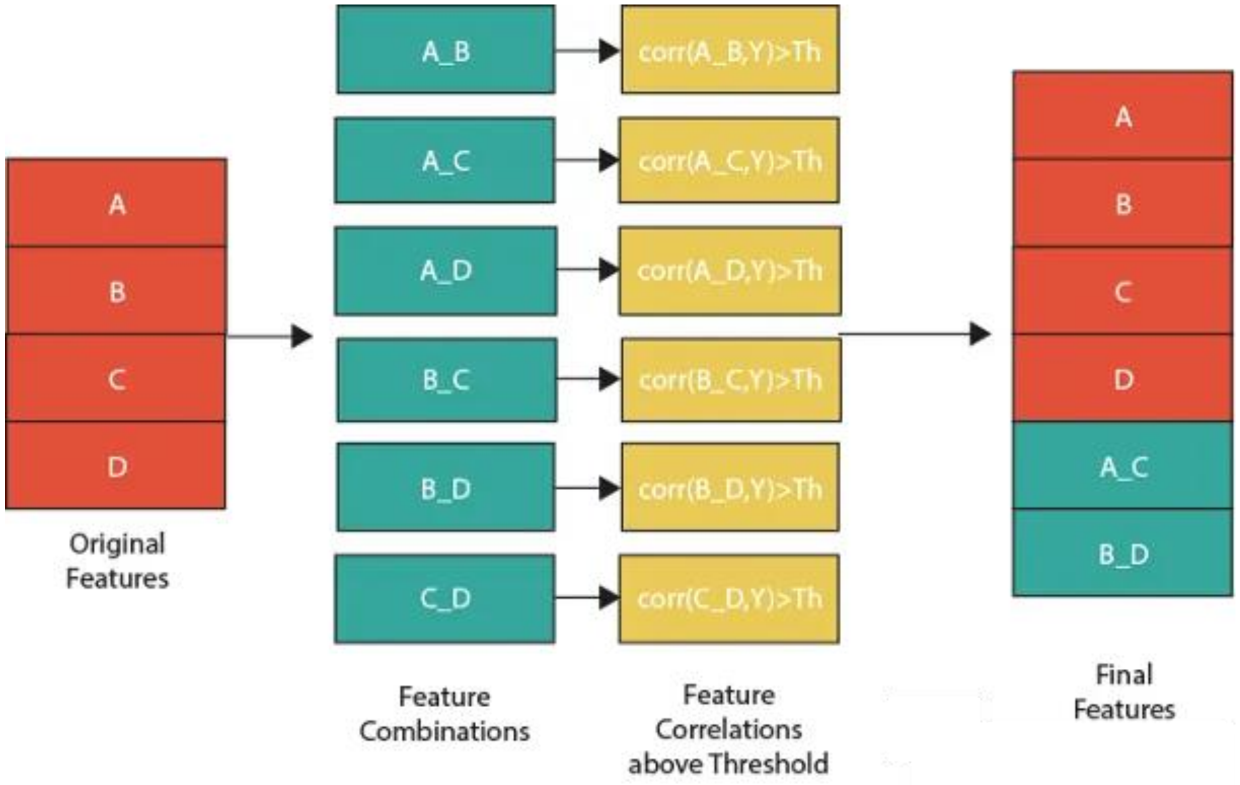
https://www.zhihu.com/question/22012482

特征工程

SLE LAB

特征组合 & 特征交叉





特征工程

特征预处理技巧

1. 独热编码：one-hot encoding 常用于处理字符分类特征

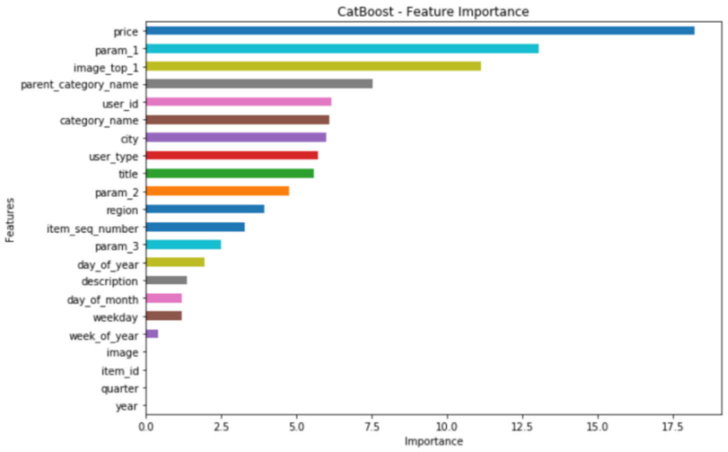
Color
Red
Red
Yellow
Green
Yellow

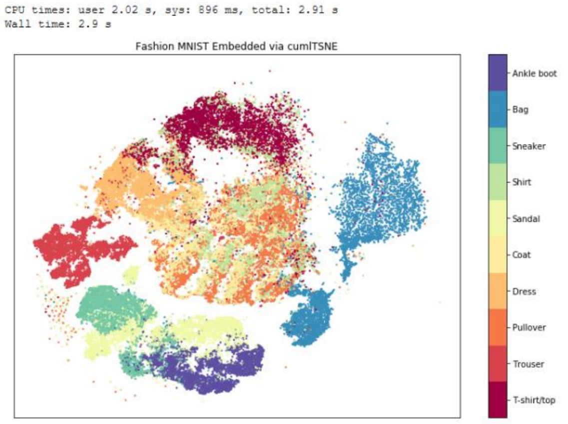
Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

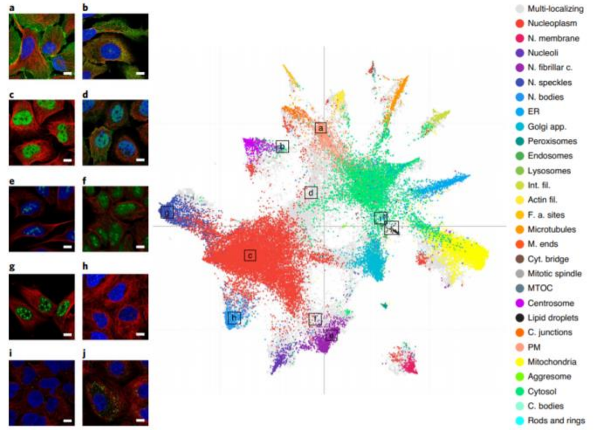
2. 时间戳处理

```
data['Year'] = data.Date.dt.year
data['Month'] = data.Date.dt.month
data['Day'] = data.Date.dt.day
data['DayOfWeek'] = data.Date.dt.dayofweek
data['WeekOfYear'] = data.Date.dt.weekofyear
```

3. 降维与可视化（PCA，LGB & CAT， t-SNE，UMAP）







作业要求

ASHRAE · FEATURED PREDICTION COMPETITION · 4 YEARS AGO

Late Submission

ASHRAE - Great Energy Predictor III

How much energy will a building consume?

Overview Data Code Models Discussion Leaderboard Rules

- **将ASHRAE代码中LightGBM代码部分改为直接使用LGBMRegressor**
- **对于每一个meter-type，分别绘制出LGB模型对应的feature importance**
- **在本地GPU上测试ASHRAE项目中的CatBoost部分（可选）**

https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.plot_importance.html