Datawhale 零基础入门数据挖掘
Task5 模型融合

分享人：ML67

目录

contents

Part 1 模型融合

Part 2 Q&A

Part 3 总结

Datawhale

Part 1 模型融合

作者：ML67

模型融合

-冲刺!冲刺!

Datawhale 零基础入门数据挖掘-Task5 模型融合

五、模型融合

Tip:此部分为零基础入门数据挖掘的 Task5 模型融合 部分，带你来了解各种模型结果的融合方式，在比赛的攻坚时刻冲刺Top，欢迎大家后续多多交流。

赛题：零基础入门数据挖掘 - 二手车交易价格预测

地址：https://tianchi.aliyun.com/competition/entrance/231784/introduction?spm=5176.12281957.1004.1.38b02448ausjSX

5.2 内容介绍

模型融合是比赛后期一个重要的环节，大体来说有如下的类型方式。

1. 简单加权融合:
    - 回归（分类概率）：算术平均融合（Arithmetic mean），几何平均融合（Geometric mean）；
    - 分类：投票（Voting)
    - 综合：排序融合(Rank averaging)，log融合

2. stacking/blending:
    - 构建多层模型，并利用预测结果再拟合预测。

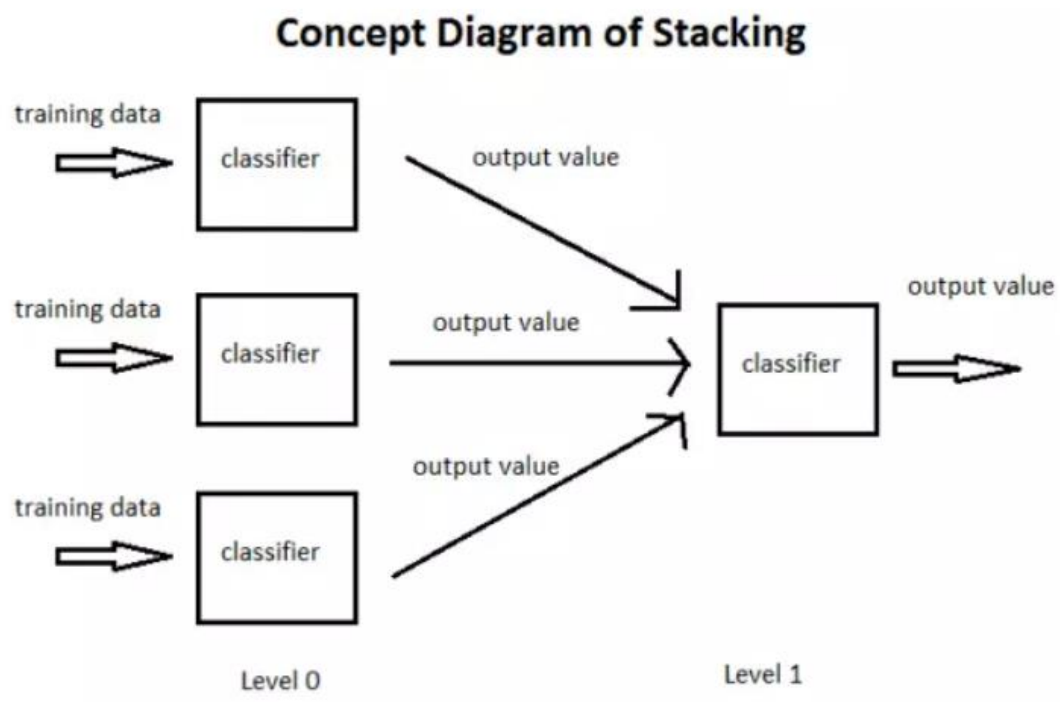
4. boosting/bagging（在xgboost，Adaboost,GBDT中已经用到）：
    - 多树的提升方法

Task 5 模型融合

Datawhale

1) 什么是 stacking

简单来说 stacking 就是当用初始训练数据学习出若干个基学习器后，将这几个学习器的预测结果作为新的训练集，来学习一个新的学习器。



2) 如何进行 stacking

算法示意图如下：

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
初级学习算法 $\mathfrak{L}_1, \mathfrak{L}_2, \ldots, \mathfrak{L}_T$;
次级学习算法 $\mathfrak{L}$.

过程：

```
for t = 1, 2, ..., T do
    h_t = L_t(D);
end for
D' = ∅;
for i = 1, 2, ..., m do
    for t = 1, 2, ..., T do
        z_it = h_t(x_i);
    end for
    D' = D' ∪ ((z_i1, z_i2, ..., z_iT), y_i);
end for
h' = L(D');
```

输出： $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x}))$

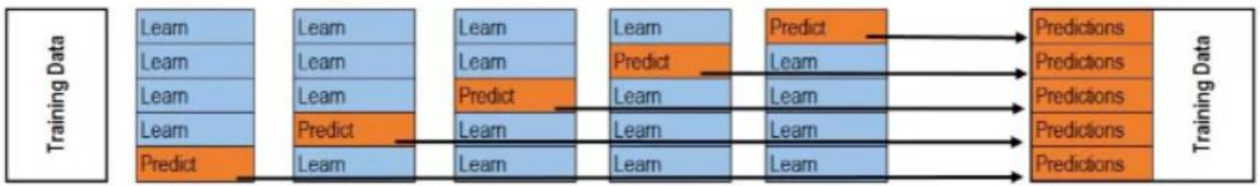
图 8.9 Stacking 算法

> 引用自 西瓜书《机器学习》

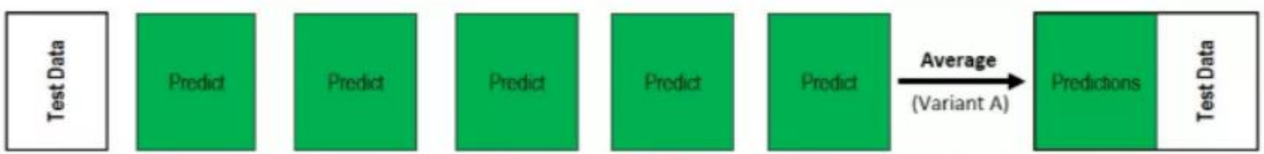
Task 5 模型融合

- 1. 次级模型尽量选择简单的线性模型
- 2. 利用K折交叉验证

K-折交叉验证：训练：



预测：



5.4 代码示例

5.4.1 回归\分类概率-融合：

1）简单加权平均，结果直接融合

In [1]:
```
## 生成一些简单的样本数据，test_prei 代表第i个模型的预测值
test_pre1 = [1.2, 3.2, 2.1, 6.2]
test_pre2 = [0.9, 3.1, 2.0, 5.9]
test_pre3 = [1.1, 2.9, 2.2, 6.0]

# y_test_true 代表第模型的真实值
y_test_true = [1, 3, 2, 6]
```

In [2]:
```
import numpy as np
import pandas as pd

## 定义结果的加权平均函数
def Weighted_method(test_pre1,test_pre2,test_pre3,w=[1/3,1/3,1/3]):
    Weighted_result = w[0]*pd.Series(test_pre1)+w[1]*pd.Series(test_pre2)+w[2]*pd.Series(test_pre3)
    return Weighted_result
```

In [3]:
```
from sklearn import metrics
# 各模型的预测结果计算MAE
print('Pred1 MAE:',metrics.mean_absolute_error(y_test_true, test_pre1))
print('Pred2 MAE:',metrics.mean_absolute_error(y_test_true, test_pre2))
print('Pred3 MAE:',metrics.mean_absolute_error(y_test_true, test_pre3))
```

```
Pred1 MAE: 0.175
Pred2 MAE: 0.075
Pred3 MAE: 0.1
```

In [4]:
```
## 根据加权计算MAE
w = [0.3,0.4,0.3] # 定义比重权值
Weighted_pre = Weighted_method(test_pre1,test_pre2,test_pre3,w)
print('Weighted_pre MAE:',metrics.mean_absolute_error(y_test_true, Weighted_pre))
```

```
Weighted_pre MAE: 0.0575
```

可以发现加权结果相对于之前的结果是有提升的，这种我们称其为简单的加权平均。

还有一些特殊的形式，比如mean平均，median平均

In [5]:
```
## 定义结果的加权平均函数
def Mean_method(test_pre1,test_pre2,test_pre3):
    Mean_result = pd.concat([pd.Series(test_pre1),pd.Series(test_pre2),pd.Series(test_pre3)],axis=1).mean(axis=1)
    return Mean_result
```

In [6]:
```
Mean_pre = Mean_method(test_pre1,test_pre2,test_pre3)
print('Mean_pre MAE:',metrics.mean_absolute_error(y_test_true, Mean_pre))
```

```
Mean_pre MAE: 0.0666666666667
```

In [7]:
```
## 定义结果的加权平均函数
def Median_method(test_pre1,test_pre2,test_pre3):
    Median_result = pd.concat([pd.Series(test_pre1),pd.Series(test_pre2),pd.Series(test_pre3)],axis=1).median(axis=1)
    return Median_result
```

In [8]:
```
Median_pre = Median_method(test_pre1,test_pre2,test_pre3)
print('Median_pre MAE:',metrics.mean_absolute_error(y_test_true, Median_pre))
```

```
Median_pre MAE: 0.075
```

2） Stacking融合(回归)：

In [9]:
```
from sklearn import linear_model

def Stacking_method(train_reg1,train_reg2,train_reg3,y_train_true,test_pre1,test_pre2,test_pre3,model_L2= linear_model.LinearRegression()):
    model_L2.fit(pd.concat([pd.Series(train_reg1),pd.Series(train_reg2),pd.Series(train_reg3)],axis=1).values,y_train_true)
    Stacking_result = model_L2.predict(pd.concat([pd.Series(test_pre1),pd.Series(test_pre2),pd.Series(test_pre3)],axis=1).values)
    return Stacking_result
```

In [10]:
```
## 生成一些简单的样本数据，test_prei 代表第i个模型的预测值
train_reg1 = [3.2, 8.2, 9.1, 5.2]
train_reg2 = [2.9, 8.1, 9.0, 4.9]
train_reg3 = [3.1, 7.9, 9.2, 5.0]
# y_test_true 代表第模型的真实值
y_train_true = [3, 8, 9, 5]

test_pre1 = [1.2, 3.2, 2.1, 6.2]
test_pre2 = [0.9, 3.1, 2.0, 5.9]
test_pre3 = [1.1, 2.9, 2.2, 6.0]

# y_test_true 代表第模型的真实值
y_test_true = [1, 3, 2, 6]
```

In [11]:
```
model_L2= linear_model.LinearRegression()
Stacking_pre = Stacking_method(train_reg1,train_reg2,train_reg3,y_train_true,
                               test_pre1,test_pre2,test_pre3,model_L2)
print('Stacking_pre MAE:',metrics.mean_absolute_error(y_test_true, Stacking_pre))
```

```
Stacking_pre MAE: 0.0421348314607
```

可以发现模型结果相对于之前有进一步的提升，这是我们需要注意的一点是，对于第二层Stacking的模型不宜选取的过于复杂，这样会导致模型在训练集上过拟合，从而使得在测试集上并不能达到很好的效果。

5.4.2 分类模型融合：

对于分类，同样的可以使用融合方法，比如简单投票，Stacking...

In [12]:

```
from sklearn.datasets import make_blobs
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_moons
from sklearn.metrics import accuracy_score,roc_auc_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
```

Task 5 模型融合

Datawhale

1）Voting投票机制：

Voting即投票机制，分为软投票和硬投票两种，其原理采用少数服从多数的思想。

In [16]:
```python
'''
硬投票：对多个模型直接进行投票，不区分模型结果的相对重要度，最终投票数最多的类为最终被预测的类。
'''
iris = datasets.load_iris()

x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

clf1 = XGBClassifier(learning_rate=0.1, n_estimators=150, max_depth=3, min_child_weight=2, subsample=0.7,
                     colsample_bytree=0.6, objective='binary:logistic')
clf2 = RandomForestClassifier(n_estimators=50, max_depth=1, min_samples_split=4,
                              min_samples_leaf=63,oob_score=True)
clf3 = SVC(C=0.1)

# 硬投票
eclf = VotingClassifier(estimators=[('xgb', clf1), ('rf', clf2), ('svc', clf3)], voting='hard')
for clf, label in zip([clf1, clf2, clf3, eclf], ['XGBBoosting', 'Random Forest', 'SVM', 'Ensemble']):
    scores = cross_val_score(clf, x, y, cv=5, scoring='accuracy')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]" % (scores.mean(), scores.std(), label))
```

```
Accuracy: 0.97 (+/- 0.02) [XGBBoosting]
Accuracy: 0.33 (+/- 0.00) [Random Forest]
Accuracy: 0.95 (+/- 0.03) [SVM]
Accuracy: 0.94 (+/- 0.04) [Ensemble]
```

Task 5 模型融合

Datawhale

In [17]:

```python
'''
软投票：和硬投票原理相同，增加了设置权重的功能，可以为不同模型设置不同权重，进而区别模型不同的重要度。
'''
x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

clf1 = XGBClassifier(learning_rate=0.1, n_estimators=150, max_depth=3, min_child_weight=2, subsample=0.8,
                     colsample_bytree=0.8, objective='binary:logistic')
clf2 = RandomForestClassifier(n_estimators=50, max_depth=1, min_samples_split=4,
                              min_samples_leaf=63,oob_score=True)
clf3 = SVC(C=0.1, probability=True)

# 软投票
eclf = VotingClassifier(estimators=[('xgb', clf1), ('rf', clf2), ('svc', clf3)], voting='soft', weights=[2, 1, 1])
clf1.fit(x_train, y_train)

for clf, label in zip([clf1, clf2, clf3, eclf], ['XGBBoosting', 'Random Forest', 'SVM', 'Ensemble']):
    scores = cross_val_score(clf, x, y, cv=5, scoring='accuracy')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]" % (scores.mean(), scores.std(), label))
```

```
Accuracy: 0.96 (+/- 0.02) [XGBBoosting]
Accuracy: 0.33 (+/- 0.00) [Random Forest]
Accuracy: 0.95 (+/- 0.03) [SVM]
Accuracy: 0.96 (+/- 0.02) [Ensemble]
```

Task 5 模型融合

Datawhale

```
#模型融合中使用到的各个单模型
clfs = [LogisticRegression(solver='lbfgs'),
        RandomForestClassifier(n_estimators=5, n_jobs=-1, criterion='gini'),
        ExtraTreesClassifier(n_estimators=5, n_jobs=-1, criterion='gini'),
        ExtraTreesClassifier(n_estimators=5, n_jobs=-1, criterion='entropy'),
        GradientBoostingClassifier(learning_rate=0.05, subsample=0.5, max_depth=6, n_estimators=5)]

#切分一部分数据作为测试集
X, X_predict, y, y_predict = train_test_split(data, target, test_size=0.3, random_state=2020)

dataset_blend_train = np.zeros((X.shape[0], len(clfs)))
dataset_blend_test = np.zeros((X_predict.shape[0], len(clfs)))

#5折stacking
n_splits = 5
skf = StratifiedKFold(n_splits)
skf = skf.split(X, y)

for j, clf in enumerate(clfs):
    #依次训练各个单模型
    dataset_blend_test_j = np.zeros((X_predict.shape[0], 5))
    for i, (train, test) in enumerate(skf):
        #5-Fold交叉训练，使用第i个部分作为预测，剩余的部分来训练模型，获得其预测的输出作为第i部分的新特征。
        X_train, y_train, X_test, y_test = X[train], y[train], X[test], y[test]
        clf.fit(X_train, y_train)
        y_submission = clf.predict_proba(X_test)[:, 1]
        dataset_blend_train[test, j] = y_submission
        dataset_blend_test_j[:, i] = clf.predict_proba(X_predict)[:, 1]
    #对于测试集，直接用这k个模型的预测值均值作为新的特征。
    dataset_blend_test[:, j] = dataset_blend_test_j.mean(1)
    print("val auc Score: %f" % roc_auc_score(y_predict, dataset_blend_test[:, j]))

clf = LogisticRegression(solver='lbfgs')
clf.fit(dataset_blend_train, y)
y_submission = clf.predict_proba(dataset_blend_test)[:, 1]

print("Val auc Score of Stacking: %f" % (roc_auc_score(y_predict, y_submission)))
```

Task 5 模型融合

Datawhale

```
#模型融合中使用到的各个单模型
clfs = [LogisticRegression(solver='lbfgs'),
        RandomForestClassifier(n_estimators=5, n_jobs=-1, criterion='gini'),
        RandomForestClassifier(n_estimators=5, n_jobs=-1, criterion='entropy'),
        ExtraTreesClassifier(n_estimators=5, n_jobs=-1, criterion='gini'),
        #ExtraTreesClassifier(n_estimators=5, n_jobs=-1, criterion='entropy'),
        GradientBoostingClassifier(learning_rate=0.05, subsample=0.5, max_depth=6, n_estimators=5)]

#切分一部分数据作为测试集
X, X_predict, y, y_predict = train_test_split(data, target, test_size=0.3, random_state=2020)

#切分训练数据集为d1,d2两部分
X_d1, X_d2, y_d1, y_d2 = train_test_split(X, y, test_size=0.5, random_state=2020)
dataset_d1 = np.zeros((X_d2.shape[0], len(clfs)))
dataset_d2 = np.zeros((X_predict.shape[0], len(clfs)))

for j, clf in enumerate(clfs):
    #依次训练各个单模型
    clf.fit(X_d1, y_d1)
    y_submission = clf.predict_proba(X_d2)[:, 1]
    dataset_d1[:, j] = y_submission
    #对于测试集，直接用这k个模型的预测值作为新的特征。
    dataset_d2[:, j] = clf.predict_proba(X_predict)[:, 1]
    print("val auc Score: %f" % roc_auc_score(y_predict, dataset_d2[:, j]))

#融合使用的模型
clf = GradientBoostingClassifier(learning_rate=0.02, subsample=0.5, max_depth=6, n_estimators=30)
clf.fit(dataset_d1, y_d2)
y_submission = clf.predict_proba(dataset_d2)[:, 1]
print("Val auc Score of Blending: %f" % (roc_auc_score(y_predict, y_submission)))
```

Task 5 模型融合

Datawhale

利用mlxtend:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier

from mlxtend.classifier import StackingClassifier

from sklearn.model_selection import cross_val_score
from mlxtend.plotting import plot_learning_curves
from mlxtend.plotting import plot_decision_regions

# 以python自带的鸢尾花数据集为例
iris = datasets.load_iris()
X, y = iris.data[:, 1:3], iris.target

clf1 = KNeighborsClassifier(n_neighbors=1)
clf2 = RandomForestClassifier(random_state=1)
clf3 = GaussianNB()
lr = LogisticRegression()
sclf = StackingClassifier(classifiers=[clf1, clf2, clf3],
                          meta_classifier=lr)

label = ['KNN', 'Random Forest', 'Naive Bayes', 'Stacking Classifier']
clf_list = [clf1, clf2, clf3, sclf]
```

Task 5 模型融合

Datawhale

In [22]:

```
def Ensemble_add_feature(train,test,target,clfs):

    # n_flods = 5
    # skf = list(StratifiedKFold(y, n_folds=n_flods))

    train_ = np.zeros((train.shape[0],len(clfs*2)))
    test_ = np.zeros((test.shape[0],len(clfs*2)))

    for j,clf in enumerate(clfs):
        '''依次训练各个单模型'''
        # print(j, clf)
        '''使用第1个部分作为预测，第2部分来训练模型，获得其预测的输出作为第2部分的新特征。'''
        # X_train, y_train, X_test, y_test = X[train], y[train], X[test], y[test]

        clf.fit(train,target)
        y_train = clf.predict(train)
        y_test = clf.predict(test)

        ## 新特征生成
        train_[:,j*2] = y_train**2
        test_[:,j*2] = y_test**2
        train_[:, j+1] = np.exp(y_train)
        test_[:, j+1] = np.exp(y_test)
        # print("val auc Score: %f" % r2_score(y_predict, dataset_d2[:, j]))
        print('Method ',j)

    train_ = pd.DataFrame(train_)
    test_ = pd.DataFrame(test_)
    return train_,test_
```

Task 5 模型融合

Datawhale

In [23]:
```python
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()

data_0 = iris.data
data = data_0[:100,:]

target_0 = iris.target
target = target_0[:100]

x_train,x_test,y_train,y_test=train_test_split(data,target,test_size=0.3)
x_train = pd.DataFrame(x_train) ; x_test = pd.DataFrame(x_test)

#模型融合中使用到的各个单模型
clfs = [LogisticRegression(),
        RandomForestClassifier(n_estimators=5, n_jobs=-1, criterion='gini'),
        ExtraTreesClassifier(n_estimators=5, n_jobs=-1, criterion='gini'),
        ExtraTreesClassifier(n_estimators=5, n_jobs=-1, criterion='entropy'),
        GradientBoostingClassifier(learning_rate=0.05, subsample=0.5, max_depth=6, n_estimators=5)]

New_train,New_test = Ensemble_add_feature(x_train,x_test,y_train,clfs)

clf = LogisticRegression()
# clf = GradientBoostingClassifier(learning_rate=0.02, subsample=0.5, max_depth=6, n_estimators=30)
clf.fit(New_train, y_train)
y_emb = clf.predict_proba(New_test)[:, 1]

print("Val auc Score of stacking: %f" % (roc_auc_score(y_test, y_emb)))
```

Task 5 模型融合

Datawhale

本赛题示例:

```
print('Predict GBDT...')
model_gbdt = build_model_gbdt(x_train,y_train)
val_gbdt = model_gbdt.predict(x_val)
subA_gbdt = model_gbdt.predict(X_test)
```

In [44]:

```
print('predict XGB...')
model_xgb = build_model_xgb(x_train,y_train)
val_xgb = model_xgb.predict(x_val)
subA_xgb = model_xgb.predict(X_test)

print('predict lgb...')
model_lgb = build_model_lgb(x_train,y_train)
val_lgb = model_lgb.predict(x_val)
subA_lgb = model_lgb.predict(X_test)
```

```
predict XGB...
predict lgb...
```

Task 5 模型融合

Datawhale

1）加权融合

In [46]:
```
def Weighted_method(test_pre1,test_pre2,test_pre3,w=[1/3,1/3,1/3]):
    Weighted_result = w[0]*pd.Series(test_pre1)+w[1]*pd.Series(test_pre2)+w[2]*pd.Series(test_pre3)
    return Weighted_result

## Init the Weight
w = [0.3,0.4,0.3]

## 测试验证集准确度
val_pre = Weighted_method(val_lgb,val_xgb,val_gbdt,w)
MAE_Weighted = mean_absolute_error(y_val,val_pre)
print('MAE of Weighted of val:',MAE_Weighted)

## 预测数据部分
subA = Weighted_method(subA_lgb,subA_xgb,subA_gbdt,w)
print('Sta inf:')
Sta_inf(subA)
## 生成提交文件
sub = pd.DataFrame()
sub['SaleID'] = X_test.index
sub['price'] = subA
sub.to_csv('./sub_Weighted.csv',index=False)
```

```
MAE of Weighted of val: 730.877443666
Sta inf:
_min -2816.93914153
_max: 88576.7842223
_mean 5920.38233546
_ptp 91393.7233639
_std 7325.20946801
_var 53658693.7502
```

Task 5 模型融合

Datawhale

2）Starking融合

In [48]:

```
## Starking

## 第一层
train_lgb_pred = model_lgb.predict(x_train)
train_xgb_pred = model_xgb.predict(x_train)
train_gbdt_pred = model_gbdt.predict(x_train)

Strak_X_train = pd.DataFrame()
Strak_X_train['Method_1'] = train_lgb_pred
Strak_X_train['Method_2'] = train_xgb_pred
Strak_X_train['Method_3'] = train_gbdt_pred

Strak_X_val = pd.DataFrame()
Strak_X_val['Method_1'] = val_lgb
Strak_X_val['Method_2'] = val_xgb
Strak_X_val['Method_3'] = val_gbdt

Strak_X_test = pd.DataFrame()
Strak_X_test['Method_1'] = subA_lgb
Strak_X_test['Method_2'] = subA_xgb
Strak_X_test['Method_3'] = subA_gbdt
```

In [50]:

```
## level2-method
model_lr_Stacking = build_model_lr(Strak_X_train,y_train)
## 训练集
train_pre_Stacking = model_lr_Stacking.predict(Strak_X_train)
print('MAE of Stacking-LR:',mean_absolute_error(y_train,train_pre_Stacking))

## 验证集
val_pre_Stacking = model_lr_Stacking.predict(Strak_X_val)
print('MAE of Stacking-LR:',mean_absolute_error(y_val,val_pre_Stacking))

## 预测集
print('Predict Stacking-LR...')
subA_Stacking = model_lr_Stacking.predict(Strak_X_test)
```

```
MAE of Stacking-LR: 628.399441036
MAE of Stacking-LR: 707.673951794
Predict Stacking-LR...
```

后处理:

In [55]:
```
subA_Stacking[subA_Stacking<10]=10  ## 去除过小的预测值

sub = pd.DataFrame()
sub['SaleID'] = X_test.index
sub['price'] = subA_Stacking
sub.to_csv('./sub_Stacking.csv',index=False)
```

In [56]:
```
print('Sta inf:')
Sta_inf(subA_Stacking)
```

```
Sta inf:
_min 10.0
_max: 90849.3729816
_mean 5917.39429976
_ptp 90839.3729816
_std 7396.09766172
_var 54702260.6217
```

3.4 经验总结

比赛的融合这个问题，个人的看法来说其实涉及多个层面，也是提分和提升模型鲁棒性的一种重要方法：

- 1）**结果层面的融合**，这种是最常见的融合方法，其可行的融合方法也有很多，比如根据结果的得分进行加权融合，还可以做Log，exp处理等。在做结果融合的时候，有一个很重要的条件是模型结果的得分要比较近似，然后结果的差异要比较大，这样的结果融合往往有比较好的效果提升。
- 2）**特征层面的融合**，这个层面其实感觉不叫融合，准确说可以叫分割，很多时候如果我们用同种模型训练，可以把特征进行切分给不同的模型，然后在后面进行模型或者结果融合有时也能产生比较好的效果。
- 3）**模型层面的融合**，模型层面的融合可能就涉及模型的堆叠和设计，比如加Staking层，部分模型的结果作为特征输入等，这些就需要多实验和思考了，基于模型层面的融合最好不同模型类型要有一定的差异，用同种模型不同的参数的收益一般是比较小的。

Task 5-模型融合 END.

--- By: ML67

Email: maolinw67@163.com
PS: 华中科技大学研究生，长期混迹Tianchi等，希望和大家多多交流。
github: https://github.com/mlw67 （近期会做一些书籍推导和代码的整理）

Datawhale

Part 2 Q&A

Q&A

Datawhale

木风

路人

木风

路人

木风

旺旺旺

木风

路人

Task5 模型结果融合相关问题

- **问题描述：** 模型融合后，怎么部署上线？进行k折交叉验证后，取结果的均值，这种模型又该如何部署呢？
-

- 回答：

- **问题描述：** 5折交叉验证后，返回了oof_train 和 test预测结果，这种怎么判断模型预测结果好坏？如何评估模型模型的泛化能力？

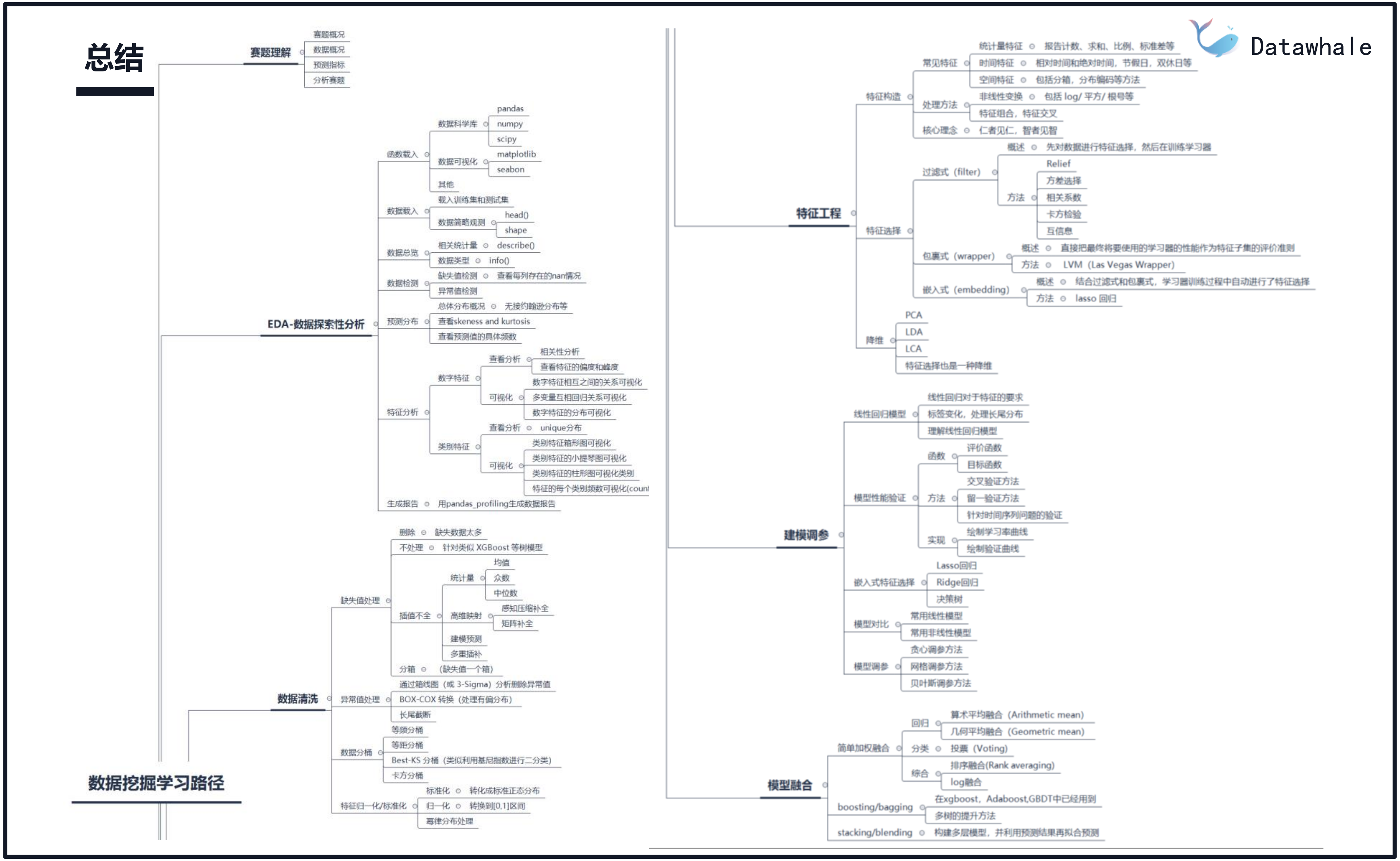
- 回答：

- **问题描述：** 发动机功率是类别型数据还是数值型，为什么不是0-600
-

- 回答：

问题描述：如果发现某个对预测目标影响很大的定类特征，但其在大多数树模型的特征重要度排序中非常靠后，该怎么办？ by8群 lchj

Part 3 总结



总结
Datawhale
数据挖掘学习路径
赛题理解
赛题概况
数据概况
预测指标
分析赛题
EDA-数据探索性分析
函数载入
数据科学库
pandas
numpy
scipy
数据可视化
matplotlib
seabon
其他
数据载入
载入训练集和测试集
数据简略观测
head()
shape
数据总览
相关统计量
describe()
数据类型
info()
数据检测
缺失值检测
查看每列存在的nan情况
异常值检测
预测分布
总体分布概况
无接约翰逊分布等
查看skeness and kurtosis
查看预测值的具体频数
特征分析
数字特征
查看分析
相关性分析
查看特征的偏度和峰度
可视化
数字特征相互之间的关系可视化
多变量互相回归关系可视化
数字特征的分布可视化
类别特征
查看分析
unique分布
可视化
类别特征箱形图可视化
类别特征的小提琴图可视化
类别特征的柱形图可视化类别
特征的每个类别频数可视化(count
生成报告
用pandas_profiling生成数据报告
数据清洗
缺失值处理
删除
缺失数据太多
不处理
针对类似 XGBoost 等树模型
插值不全
统计量
均值
众数
中位数
高维映射
感知压缩补全
矩阵补全
建模预测
多重插补
分箱
（缺失值一个箱）
异常值处理
通过箱线图（或 3-Sigma）分析删除异常值
BOX-COX 转换（处理有偏分布）
长尾截断
数据分桶
等频分桶
等距分桶
Best-KS 分桶（类似利用基尼指数进行二分类）
卡方分桶
特征归一化/标准化
标准化
转化成标准正态分布
归一化
转换到[0,1]区间
幂律分布处理
特征工程
特征构造
常见特征
统计量特征
报告计数、求和、比例、标准差等
时间特征
相对时间和绝对时间，节假日，双休日等
空间特征
包括分箱，分布编码等方法
处理方法
非线性变换
包括 log/ 平方/ 根号等
特征组合，特征交叉
核心理念
仁者见仁，智者见智
特征选择
过滤式（filter）
概述
先对数据进行特征选择，然后在训练学习器
方法
Relief
方差选择
相关系数
卡方检验
互信息
包裹式（wrapper）
概述
直接把最终将要使用的学习器的性能作为特征子集的评价准则
方法
LVM（Las Vegas Wrapper）
嵌入式（embedding）
概述
结合过滤式和包裹式，学习器训练过程中自动进行了特征选择
方法
lasso 回归
降维
PCA
LDA
LCA
特征选择也是一种降维
建模调参
线性回归模型
线性回归对于特征的要求
标签变化，处理长尾分布
理解线性回归模型
模型性能验证
函数
评价函数
目标函数
方法
交叉验证方法
留一验证方法
针对时间序列问题的验证
实现
绘制学习率曲线
绘制验证曲线
嵌入式特征选择
Lasso回归
Ridge回归
决策树
模型对比
常用线性模型
常用非线性模型
模型调参
贪心调参方法
网格调参方法
贝叶斯调参方法
模型融合
简单加权融合
回归
算术平均融合（Arithmetic mean）
几何平均融合（Geometric mean）
分类
投票（Voting）
综合
排序融合(Rank averaging)
log融合
boosting/bagging
在xgboost，Adaboost,GBDT中已经用到
多树的提升方法
stacking/blending
构建多层模型，并利用预测结果再拟合预测

Datawhale

作者

--- By: ML67

Email: maolinw67@163.com
PS: 华中科技大学研究生，长期混迹Tianchi等，希望和大家多多交流。
github: https://github.com/mlw67 （近期会做一些书籍推导和代码的整理）

--- By: AI蜗牛车

PS：东南大学研究生，研究方向主要是时空序列预测和时间序列数据挖掘
公众号： AI蜗牛车
知乎： https://www.zhihu.com/people/seu-aigua-niu-che
github: https://github.com/chehongshu

--- By: 阿泽

PS：复旦大学计算机研究生
知乎：阿泽 https://www.zhihu.com/people/is-aze（主要面向初学者的知识整理）

--- By: 小雨姑娘

PS：数据挖掘爱好者，多次获得比赛TOP名次。
知乎：小雨姑娘的机器学习笔记：https://zhuanlan.zhihu.com/mlbasic