

Kaggle初探--房价预测案例之模型建立



超级个体颖颖 (/u/b7f94092fc21) [+ 关注](#)

2017.06.22 14:46* 字数 386 阅读 6743 评论 12 喜欢 16

(/u/b7f94092fc21)

概述

本文数据来源kaggle的House Prices: Advanced Regression Techniques (<https://link.jianshu.com?t=https://www.kaggle.com/c/house-prices-advanced-regression-techniques/discussion>)大赛。

本文接着Kaggle 初探 -- 房价预测案例之数据分析 (<https://www.jianshu.com/p/62716b33e7be>)做模型部分。

```
import pandas as pd
import numpy as np
import seaborn as sns
from scipy import stats
from scipy.stats import skew
from scipy.stats import norm
import matplotlib
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# import warnings
# warnings.filterwarnings('ignore')

%config InlineBackend.figure_format = 'retina' #set 'png' here when working on notebook
%matplotlib inline
```

```
train_df = pd.read_csv("../input/train.csv")
test_df = pd.read_csv("../input/test.csv")
```

特征工程

此处特征的处理根据Kaggle 初探 -- 房价预测案例之数据分析 (<https://www.jianshu.com/p/62716b33e7be>)的分析来做。

```
all_df = pd.concat((train_df.loc[:, 'MSSubClass': 'SaleCondition'], test_df.loc[:, 'MSSubClass': 'SaleCondition']))
all_df['MSSubClass'] = all_df['MSSubClass'].astype(str)
quantitative = [f for f in all_df.columns if all_df.dtypes[f] != 'object']
qualitative = [f for f in all_df.columns if all_df.dtypes[f] == 'object']
```

缺失数据处理



对于缺失数据，我们直接将列删除

```
missing = all_df.isnull().sum()
missing.sort_values(inplace=True, ascending=False)
missing = missing[missing > 0]
```

```
#dealing with missing data
all_df = all_df.drop(missing[missing>1].index,1)
# 对于missing 1 的我们到时候以平均数填充
```

```
all_df.isnull().sum()[all_df.isnull().sum()>0]
```

```
Exterior1st    1
Exterior2nd    1
BsmtFinSF1     1
BsmtFinSF2     1
BsmtUnfSF      1
TotalBsmtSF    1
Electrical     1
KitchenQual    1
GarageCars     1
GarageArea     1
SaleType       1
dtype: int64
```

(/apps/redi
utm_sourc
banner-clc



处理log项

GrLivArea、1stFlrSF、2ndFlrSF、TotalBsmtSF、LotArea、KitchenAbvGr、GarageArea 以上特征我们进行logp处理

```
logfeatures = ['GrLivArea', '1stFlrSF', '2ndFlrSF', 'TotalBsmtSF', 'LotArea', 'KitchenAbvGr', 'GarageArea']
```

```
for logfeature in logfeatures:
    all_df[logfeature] = np.log1p(all_df[logfeature].values)
```

处理Boolean变量

```
all_df['HasBasement'] = all_df['TotalBsmtSF'].apply(lambda x: 1 if x > 0 else 0)
all_df['HasGarage'] = all_df['GarageArea'].apply(lambda x: 1 if x > 0 else 0)
all_df['Has2ndFloor'] = all_df['2ndFlrSF'].apply(lambda x: 1 if x > 0 else 0)
all_df['HasWoodDeck'] = all_df['WoodDeckSF'].apply(lambda x: 1 if x > 0 else 0)
all_df['HasPorch'] = all_df['OpenPorchSF'].apply(lambda x: 1 if x > 0 else 0)
all_df['HasPool'] = all_df['PoolArea'].apply(lambda x: 1 if x > 0 else 0)
all_df['IsNew'] = all_df['YearBuilt'].apply(lambda x: 1 if x > 2000 else 0)
```

```
quantitative = [f for f in all_df.columns if all_df.dtypes[f] != 'object']
qualitative = [f for f in all_df.columns if all_df.dtypes[f] == 'object']
```

对于定性变量的encode



```
all_dummy_df = pd.get_dummies(all_df)
```

对于数值变量进行标准化

```
all_dummy_df.isnull().sum().sum()
```

```
6
```

```
mean_cols = all_dummy_df.mean()
all_dummy_df = all_dummy_df.fillna(mean_cols)
```

```
all_dummy_df.isnull().sum().sum()
```

```
0
```

```
X = all_dummy_df[quantitative]
std = StandardScaler()
s = std.fit_transform(X)
```

```
all_dummy_df[quantitative] = s
```

```
dummy_train_df = all_dummy_df.loc[train_df.index]
dummy_test_df = all_dummy_df.loc[test_df.index]
```

```
y_train = np.log(train_df.SalePrice)
```

模型预测

此处我们先运用多个模型进行预测，最后进行bagging操作

岭回归

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import cross_val_score
```

```
y_train.values
```

```
array([ 12.24769432,  12.10901093,  12.31716669, ...,  12.49312952,
        11.86446223,  11.90158345])
```

(/apps/redi
utm_sourc
banner-clc

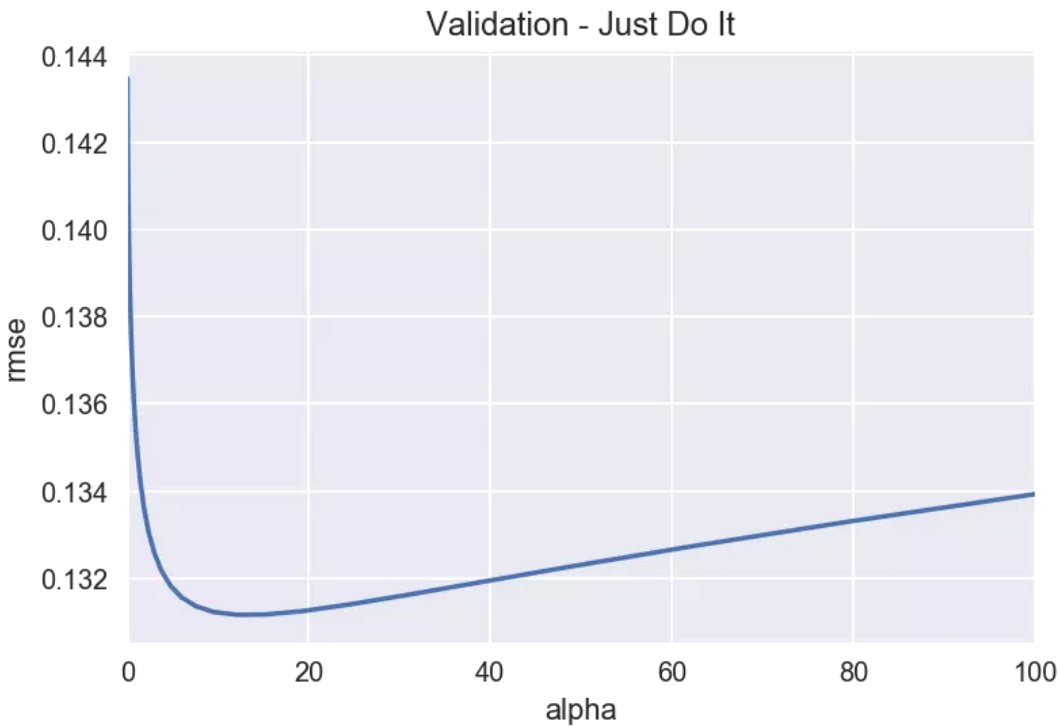


```
def rmse_cv(model):  
    rmse= np.sqrt(-cross_val_score(model, dummy_train_df, y_train.values, scoring="neg_mean_squared_error"))  
    return(rmse)
```

```
alphas = np.logspace(-3, 2, 50)  
cv_ridge = []  
coefs = []  
for alpha in alphas:  
    model = Ridge(alpha = alpha)  
    model.fit(dummy_train_df,y_train)  
    cv_ridge.append(rmse_cv(model).mean())  
    coefs.append(model.coef_)
```

```
import matplotlib.pyplot as plt  
%matplotlib inline  
cv_ridge = pd.Series(cv_ridge, index = alphas)  
cv_ridge.plot(title = "Validation - Just Do It")  
plt.xlabel("alpha")  
plt.ylabel("rmse")  
# plt.plot(alphas, cv_ridge)  
# plt.title("Alpha vs CV Error")
```

<matplotlib.text.Text at 0x118dd0ef0>



output_30_1.png

(/apps/red
utm_sourc
banner-cl

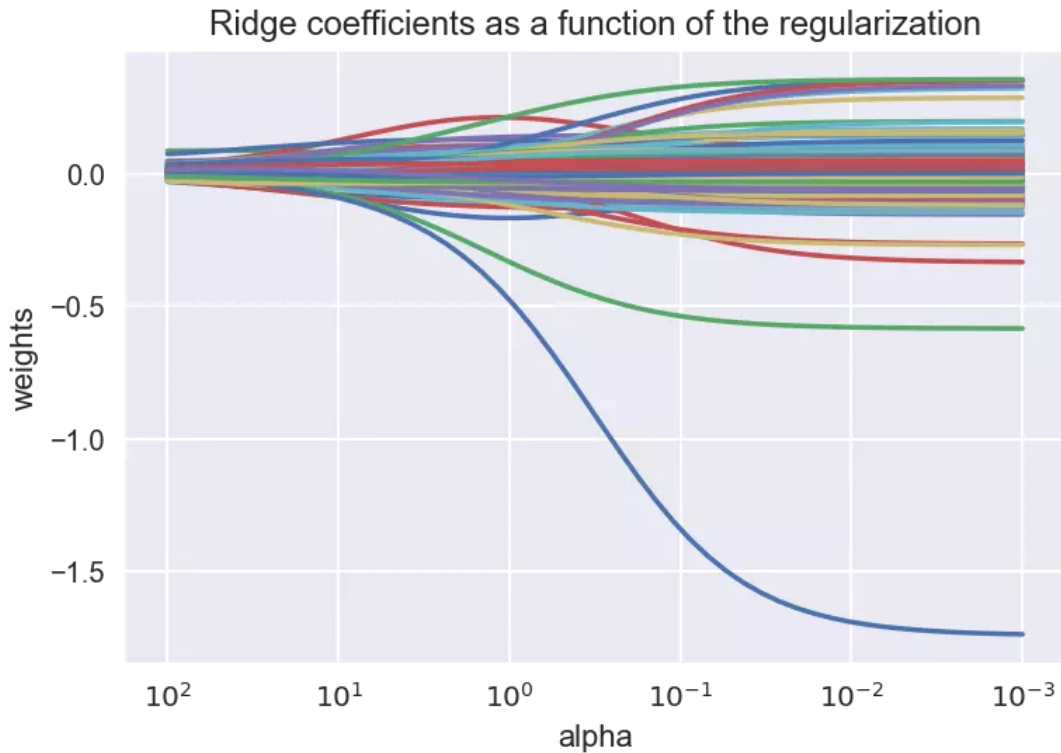


```
# 岭迹图
# matplotlib.rcParams['figure.figsize'] = (12.0, 12.0)
ax = plt.gca()

# ax.set_color_cycle(['b', 'r', 'g', 'c', 'k', 'y', 'm'])

ax.plot(alphas, coefs)
ax.set_xscale('log')
ax.set_xlim(ax.get_xlim()[::-1]) # reverse axis
plt.xlabel('alpha')
plt.ylabel('weights')
plt.title('Ridge coefficients as a function of the regularization')
plt.axis('tight')
plt.show()
```

(/apps/redi
utm_sourc
banner-clip



output_31_0.png

很尴尬的岭迹图，主要是现在feature太多了。看不出什么东西来

Lasso

Lasso能针对上面特征太多的问题，来选择一部分重要的特征

```
from sklearn.linear_model import Lasso, LassoCV
```

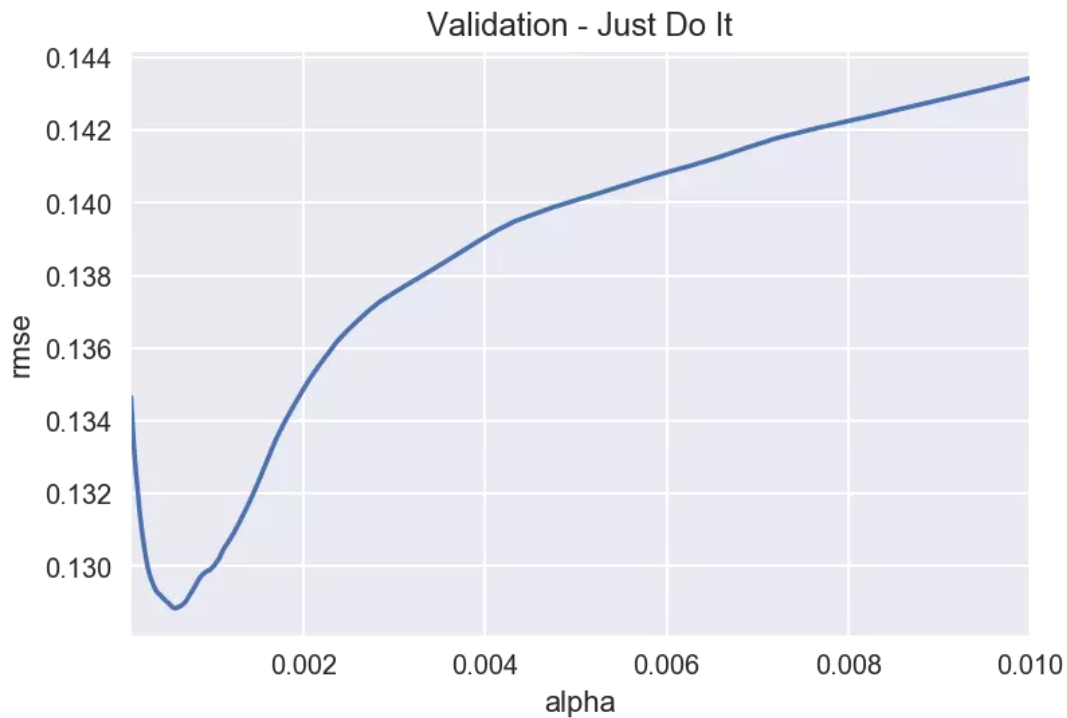
```
# alphas = np.logspace(-3, 2, 50)
# alphas = [1, 0.1, 0.001, 0.0005]
alphas = np.logspace(-4, -2, 100)
cv_lasso = []
coefs = []
for alpha in alphas:
    model = Lasso(alpha = alpha, max_iter=5000)
    model.fit(dummy_train_df, y_train)
    cv_lasso.append(rmse_cv(model).mean())
    coefs.append(model.coef_)
```



```
cv_lasso = pd.Series(cv_lasso, index = alphas)
cv_lasso.plot(title = "Validation - Just Do It")
plt.xlabel("alpha")
plt.ylabel("rmse")
# plt.plot(alphas, cv_ridge)
# plt.title("Alpha vs CV Error")
```

(/apps/redi
utm_sourc
banner-clic

<matplotlib.text.Text at 0x118bca940>



output_36_1.png

```
print(cv_lasso.min(), cv_lasso.argmin())
```

```
0.128843680722 0.000585702081806
```

```
model = Lasso(alpha = 0.00058,max_iter=5000)
model.fit(dummy_train_df,y_train)
```

```
Lasso(alpha=0.00058, copy_X=True, fit_intercept=True, max_iter=5000,
      normalize=False, positive=False, precompute=False, random_state=None,
      selection='cyclic', tol=0.0001, warm_start=False)
```

```
coef = pd.Series(model.coef_, index = dummy_train_df.columns)
```

```
print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the other "
```

```
Lasso picked 84 variables and eliminated the other 142 variables
```

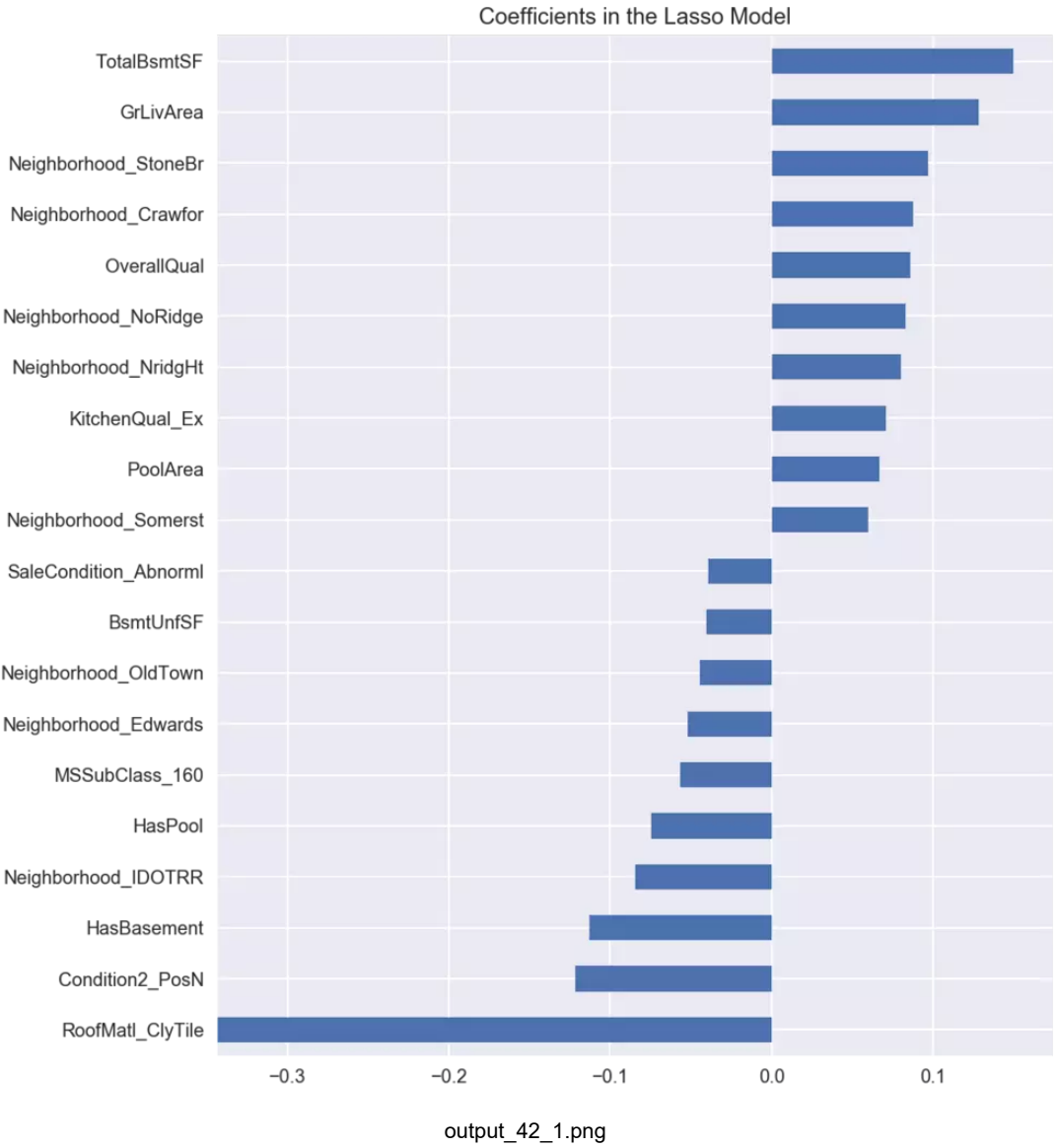


```
imp_coef = pd.concat([coef.sort_values().head(10),
                      coef.sort_values().tail(10)])
```

```
matplotlib.rcParams['figure.figsize'] = (8.0, 10.0)
imp_coef.plot(kind = "barh")
plt.title("Coefficients in the Lasso Model")
```

```
<matplotlib.text.Text at 0x11aa1dbe0>
```

(/apps/redi
utm_sourc
banner-clip



Elastic Net

结合了 Lasso 和 Ridge 两个模型，既能解决 Lasso 的共线问题，又能很好的筛选变量

```
from sklearn.linear_model import ElasticNet,ElasticNetCV
```

```
elastic = ElasticNetCV(l1_ratio=[.1, .5, .7, .9, .95, .99, 1],
                      alphas=[0.001, 0.05, 0.1, 0.3, 1, 3, 5, 10, 15, 30])
```



```
elastic.fit(dummy_train_df, y_train)
```

```
ElasticNetCV(alphas=[0.001, 0.05, 0.1, 0.3, 1, 3, 5, 10, 15, 30, 50, 75],
              copy_X=True, cv=5, eps=0.001, fit_intercept=True,
              l1_ratio=[0.1, 0.5, 0.7, 0.9, 0.95, 0.99, 1], max_iter=5000,
              n_alphas=100, n_jobs=1, normalize=False, positive=False,
              precompute='auto', random_state=None, selection='cyclic',
              tol=0.0001, verbose=0)
```

```
rmse_cv(elastic).mean()
```

```
0.12908591441325348
```

(/apps/redi
utm_sourc
banner-clc

特征二

很尴尬的发现这种提取特征的方式，取得的结果不是很好，所以，此处我们采用

<https://www.kaggle.com/opanichev/ensemble-of-4-models-with-cv-lb-0-11489>

(<https://link.jianshu.com?t=https://www.kaggle.com/opanichev/ensemble-of-4-models-with-cv-lb-0-11489>) 这篇文章的方式来处理特征

```
import utils
```

```
train_df_munged,label_df,test_df_munged = utils.feature_engineering()
```

```
Training set size: (1456, 111)
Test set size: (1459, 111)
Features engineering..
0:00:14.427659
```

```
test_df = pd.read_csv('../input/test.csv')
```

```
from sklearn.metrics import mean_squared_error,make_scorer
from sklearn.model_selection import cross_val_score
# 定义自己的score函数
def my_custom_loss_func(ground_truth, predictions):
    return np.sqrt(mean_squared_error(np.exp(ground_truth), np.exp(predictions)))

my_loss_func = make_scorer(my_custom_loss_func, greater_is_better=False)
```

```
def rmse_cv2(model):
    rmse= np.sqrt(-cross_val_score(model, train_df_munged, label_df.SalePrice, scorir
    return(rmse)
```

L2 岭回归

```
from sklearn.linear_model import RidgeCV,Ridge
```




```
alphas = np.logspace(-3, 2, 100)
model_ridge = RidgeCV(alphas=alphas).fit(train_df_munged, label_df.SalePrice)
```

```
# Run prediction on training set to get a rough idea of how well it does.
pred_Y_ridge = model_ridge.predict(train_df_munged)
print("Ridge score on training set: ", model_ridge.score(train_df_munged, label_df.SalePrice))
```

```
Ridge score on training set: 0.940191172098
```

```
print("cross_validation: ", rmse_cv2(model_ridge).mean())
```

```
cross_validation: 0.111384227695
```

Lasso

```
from sklearn.linear_model import Lasso, LassoCV
```

```
model_lasso = LassoCV(eps=0.0001, max_iter=20000).fit(train_df_munged, label_df.SalePrice)
```

```
# Run prediction on training set to get a rough idea of how well it does.
pred_Y_lasso = model_lasso.predict(train_df_munged)
print("Lasso score on training set: ", model_lasso.score(train_df_munged, label_df.SalePrice))
```

```
Lasso score on training set: 0.940560493411
```

```
print("cross_validation: ", rmse_cv2(model_lasso).mean())
```

```
cross_validation: 0.11036670335
```

Elastic Net

```
from sklearn.linear_model import ElasticNet, ElasticNetCV
```

```
model_elastic = ElasticNetCV(l1_ratio=[.1, .5, .7, .9, .95, .99, 1],
                             alphas=[0.001, 0.05, 0.1, 0.3, 1, 3, 5, 10, 15, 30])
```

```
model_elastic.fit(train_df_munged, label_df.SalePrice)
```

(/apps/redi
utm_sourc
banner-clic



```
ElasticNetCV(alphas=[0.001, 0.05, 0.1, 0.3, 1, 3, 5, 10, 15, 30, 50, 75],
              copy_X=True, cv=5, eps=0.001, fit_intercept=True,
              l1_ratio=[0.1, 0.5, 0.7, 0.9, 0.95, 0.99, 1], max_iter=10000,
              n_alphas=100, n_jobs=1, normalize=False, positive=False,
              precompute='auto', random_state=None, selection='cyclic',
              tol=0.0001, verbose=0)
```

(/apps/redi
utm_sourc
banner-clic

```
# Run prediction on training set to get a rough idea of how well it does.
pred_Y_elastic = model_elastic.predict(train_df_munged)
print("Elastic score on training set: ", model_elastic.score(train_df_munged, label_df.SalePrice))
```

Elastic score on training set: 0.940707195529

```
print("cross_validation: ", rmse_cv2(model_elastic).mean())
```

cross_validation: 0.109106832215

XGBoost

参看: <https://www.kaggle.com/aharless/amit-choudhary-s-kernel-notebook-ified>
(<https://link.jianshu.com?t=https://www.kaggle.com/aharless/amit-choudhary-s-kernel-notebook-ified>)

此处XGBoost怎么进行调优缺失

```
# XGBoost -- I did some "manual" cross-validation here but should really find
# these hyperparameters using CV. ;-)

import xgboost as xgb

model_xgb = xgb.XGBRegressor(
    colsample_bytree=0.2,
    gamma=0.0,
    learning_rate=0.05,
    max_depth=6,
    min_child_weight=1.5,
    n_estimators=7200,
    reg_alpha=0.9,
    reg_lambda=0.6,
    subsample=0.2,
    seed=42,
    silent=1)

model_xgb.fit(train_df_munged, label_df.SalePrice)

# Run prediction on training set to get a rough idea of how well it does.
pred_Y_xgb = model_xgb.predict(train_df_munged)
print("XGBoost score on training set: ", model_xgb.score(train_df_munged, label_df.SalePrice))
```

XGBoost score on training set: 0.990853904354

```
print("cross_validation: ", rmse_cv2(model_xgb).mean())
```



```
cross_validation: 0.11857237109
```

```
print("score: ",mean_squared_error(model_xgb.predict(train_df_munged),label_df.SalePr
```

```
score: 0.0014338471114
```

(/apps/redi
utm_sourc
banner-clic

Ensemble

```
from sklearn.linear_model import LinearRegression
# Create linear regression object
regr = LinearRegression()
```

```
train_x = np.concatenate(
    (pred_Y_lasso[np.newaxis, :].T,pred_Y_ridge[np.newaxis, :].T,
     pred_Y_elastic[np.newaxis, :].T,pred_Y_xgb[np.newaxis, :].T), axis=1)
```

```
regr.fit(train_x,label_df.SalePrice)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
regr.coef_
```

```
array([ 2.28665601, -0.15426296, -2.43483763,  1.30394217])
```

```
print("Ensemble score on training set: ", regr.score(train_x,label_df.SalePrice)) # 0.993716162184
```

```
Ensemble score on training set: 0.993716162184
```

很尴尬的发现通过ensemble操作并没有任何帮助

```
print("score: ",mean_squared_error(regr.predict(train_x),label_df.SalePrice))
```

```
score: 0.000985126664884
```

提交答案

```
model_lasso.predict(test_df_munged)[np.newaxis, :].T
```



```
array([ 11.67407587, 11.95939264, 12.11110308, ..., 12.01706033,
        11.70077616, 12.29221647])

test_x = np.concatenate(
    (model_lasso.predict(test_df_munged)[np.newaxis, :].T,model_ridge.predict(test_df_mur
        model_elastic.predict(test_df_munged)[np.newaxis, :].T, mc
    ,axis=1)

y_final = regr.predict(test_x)

y_final

array([ 11.83896506, 11.95544055, 12.08303061, ..., 12.02530217,
        11.71776755, 12.16714229])

submission_df = pd.DataFrame(data= {'Id' : test_df.Id, 'SalePrice': np.exp(y_final)})

submission_df.to_csv("bag-4.csv",index=False) # 取消index的存储
```

(/apps/redi
utm_sourc
banner-clic



您的每一次打赏，都是对我最大的鼓励，期待我们共同进步

赞赏支持

深度学习基石 (/nb/9393067) 举报文章 © 著作权归作者所有



超级个体颢頔 (/u/b7f94092fc21)


写了 109918 字，被 430 人关注，获得了 430 个喜欢

(/u/b7f94092fc21)

专注大规模分布式系统开发，紧跟人工智能浪潮

+ 关注

喜欢 | 16



更多分享

