model_Lasso

May 24, 2019

Fit in Lasso with clean data and see its performance. Use grid search to improve performance.

In [3]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [4]:
```
clean_data = pd.read_csv('clean_data.csv')
clean_data.head()
```

Out[4]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	\
0	1	60	65.0	8450	7	5	2003	
1	2	20	80.0	9600	6	8	1976	
2	3	60	68.0	11250	7	5	2001	
3	4	70	60.0	9550	7	5	1915	
4	5	60	84.0	14260	8	5	2000	

	YearRemodAdd	ExterQual	ExterCond	...	SaleType_ConLI	\
0	2003	3	2	...	0	
1	1976	2	2	...	0	
2	2002	3	2	...	0	
3	1970	2	2	...	0	
4	2000	3	2	...	0	

	SaleType_ConLw	SaleType_New	SaleType_Oth	SaleType_WD	\
0	0	0	0	1	
1	0	0	0	1	
2	0	0	0	1	
3	0	0	0	1	
4	0	0	0	1	

	SaleCondition_AdjLand	SaleCondition_Alloca	SaleCondition_Family	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	

```
4                     0                     0                     0

   SaleCondition_Normal  SaleCondition_Partial
0                     1                     0
1                     1                     0
2                     1                     0
3                     0                     0
4                     1                     0

[5 rows x 534 columns]
```

In [5]:
```
y=clean_data['SalePrice']
clean_data=clean_data.drop(['SalePrice'],axis=1) #drop the y in clean_data
X = clean_data.iloc[:,1:] # drop the id column
X.shape
```

Out[5]: (1459, 532)

In [9]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [6]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=
# scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_sc = scaler.fit_transform(X_train)
X_test_sc = scaler.transform(X_test)
```

```
/Users/fanwenyu/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/data.py:625: DataC
  return self.partial_fit(X, y)
/Users/fanwenyu/anaconda3/lib/python3.6/site-packages/sklearn/base.py:462: DataConversionWarni
  return self.fit(X, **fit_params).transform(X)
/Users/fanwenyu/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:8: DataConversionW
```

In [7]:
```
clf = Lasso(random_state=42)
```

In [8]:
```
clf.fit(X_train_sc,y_train)
y_test_pred = clf.predict(X_test_sc)
y_train_pred = clf.predict(X_train_sc)
```

```
/Users/fanwenyu/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/coordinate_descent.
  ConvergenceWarning)
```

In [10]:
```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_test_pred)
from math import sqrt
sqrt(mse)
```

```
Out[10]: 33069.93524702181

In [55]: mse = mean_squared_error(y_train, y_train_pred)
         mse

Out[55]: 200068353.65906683

In [56]: # grid search to get the best hyperparameter.
         from sklearn.model_selection import GridSearchCV
         parameters = { 'max_iter':[1000,2000,5000],'alpha':[1, 10,100,1000,10000]}
         ls = Lasso(random_state=42)
         clf = GridSearchCV(ls, parameters, cv=5)
         clf.fit(X_train_sc,y_train)
         y_test_pred = clf.predict(X_test_sc)
         y_train_pred = clf.predict(X_train_sc)

In [2]: clf.best_params_


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-2-9772bec5683a> in <module>()
    ----> 1 clf.best_params_


        NameError: name 'clf' is not defined


In [1]: clf.coef_


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-1-eaad8d6955f1> in <module>()
    ----> 1 clf.coef_


        NameError: name 'clf' is not defined


In [57]: from math import sqrt
         mse_test = mean_squared_error(y_test, y_test_pred)
         sqrt(mse_test)

Out[57]: 26802.87448312158
```

```
In [58]: mse_train = mean_squared_error(y_train, y_train_pred)
         sqrt(mse_train)

Out[58]: 19794.68865118897

In [59]: plt.scatter(y_test, y_test_pred)

Out[59]: <matplotlib.collections.PathCollection at 0x10f1a3828>
```