ridge

February 26, 2024

1 Régression Régularisées : RIDGE

1.0.1 ZWANEB

0. Importez les libairies usuelles

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1. Chargez en mémoire le dataset des prix des logements à boston grâce à la commande :

```
from sklearn.datasets import load_boston
boston=load_boston()
boston_df=pd.DataFrame(boston.data,columns=boston.feature_names)
```

[3]:
```
from sklearn.datasets import load_boston
boston=load_boston()
boston_df=pd.DataFrame(boston.data,columns=boston.feature_names)
boston_df.head()
```

[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	\
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	

	PTRATIO	B	LSTAT
0	15.3	396.90	4.98
1	17.8	396.90	9.14
2	17.8	392.83	4.03
3	18.7	394.63	2.94
4	18.7	396.90	5.33

2. Créez un dataframe contenant les variables explicatives et un autre contenant uniquement la variable cible qui est le prix des maisons

[8]:
```
boston_df['PRICE'] = boston.target
X = boston_df.drop('PRICE',axis = 1)
y = boston_df['PRICE']
y = y.to_frame()
y.head()
```

[8]:

	PRICE
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2

3. Utilisez la commande `train_test_split` du package `sklearn.model_selection` afin de créer un échantillon d'entraînement contenant 70% des observations et un échantillon test contenant 30% des observations.

[79]:
```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y,
                                                 test_size = 0.3)
```

[80]:
```
boston_df.shape
```

[80]:
```
(506, 14)
```

[81]:
```
X_train.shape
```

[81]:
```
(354, 13)
```

[82]:
```
y_train
```

[82]:

	PRICE
133	18.4
99	33.2
367	23.1
433	14.3
450	13.4
..	…
210	21.7
25	13.9
342	16.5
384	8.8
126	15.7

[354 rows x 1 columns]

4. Générer un modèle de régression linéaire classique, un modèle ridge où alpha vaut 0.01 et un modèle ridge où alpha vaut 100.

[83]:
```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression

lin_regressor=LinearRegression().fit(X_train, y_train)


from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV

ridge1=Ridge(alpha=100, copy_X=True, fit_intercept=True, max_iter=None,
   normalize=False, random_state=None, solver='auto', tol=0.001)
ridge2=Ridge(alpha=0.01,copy_X=True, fit_intercept=True, max_iter=None,
   normalize=False, random_state=None, solver='auto', tol=0.001)
ridge1.fit(X_train,y_train)
ridge2.fit(X_train,y_train)
```

```
/Users/User/.local/lib/python3.8/site-
packages/sklearn/linear_model/_base.py:148: FutureWarning: 'normalize' was
deprecated in version 1.0 and will be removed in 1.2. Please leave the normalize
parameter to its default value to silence this warning. The default behavior of
this estimator is to not do any normalization. If normalization is needed please
use sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
/Users/User/.local/lib/python3.8/site-
packages/sklearn/linear_model/_base.py:148: FutureWarning: 'normalize' was
deprecated in version 1.0 and will be removed in 1.2. Please leave the normalize
parameter to its default value to silence this warning. The default behavior of
this estimator is to not do any normalization. If normalization is needed please
use sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
```

[83]:
```
Ridge(alpha=0.01, normalize=False)
```

5. Entraînez ces modèles sur les données sur les données d'apprentissage

[84]:
```
y_train_pred_lin = lin_regressor.predict(X_train)
y_train_pred_ridge_1 = ridge1.predict(X_train)
y_train_pred_ridge_2 =ridge2.predict(X_train)
```

[ ]:
```

```

6. Produisez les scores de performance de c'est trois modèles sur l'échantillon d'apprentissage et de validation grâce à l'attribut .score

[85]:
```
from sklearn.model_selection import cross_val_score

print("Linear Regression score :", lin_regressor.score(X_test, y_test))
print("Ridge with small Alpha score :",ridge1.score(X_test, y_test) )
print("Ridge with large Alpha score:", ridge2.score(X_test, y_test) )
```

```
Linear Regression score : 0.7272424835800392
Ridge with small Alpha score : 0.6817591826230354
Ridge with large Alpha score: 0.727129554760964
```

7. Comparez les coefficients des trois modèle à l'aide d'un graphique, que remarquez vous ?

[86]:
```
df = pd.DataFrame(lin_regressor.coef_, index = ['Item_1'])
abs(df.iloc[0])
```

[86]:
```
0      0.112694
1      0.046821
2      0.044750
3      2.907119
4     15.311814
5      3.607487
6      0.007125
7      1.365356
8      0.267880
9      0.010000
10     1.012241
11     0.009451
12     0.564692
Name: Item_1, dtype: float64
```

[ ]:

[ ]:

[87]:
```
lr_all = lin_regressor
df_lin = pd.DataFrame(lin_regressor.coef_, index = ['Item_1'])
df_ridge = pd.DataFrame(ridge2.coef_)
df_ridge2 = pd.DataFrame(ridge1.coef_)
lr_all_coeffcients = pd.DataFrame([X_train.columns,df_lin.iloc[0],abs(df_lin.
 ↪iloc[0]),df_ridge.iloc[0],abs(df_ridge.iloc[0]),df_ridge.
 ↪iloc[0],abs(df_ridge.iloc[0])]).T
lr_all_coeffcients = lr_all_coeffcients.rename(columns={0: 'features', 1:␣
 ↪'coef_linear_regressor', 2: 'coef_abs_linear_regressor',
                                                       3:␣
 ↪'coef_ridge_small_alpha',4:'coef_abs_ridge_small_alpha',
                                                        5:
 ↪'coef_ridge_large_alpha',6:'coef_abs_ridge_large_alpha'
                                                       })
lr_all_coeffcients.head()
```

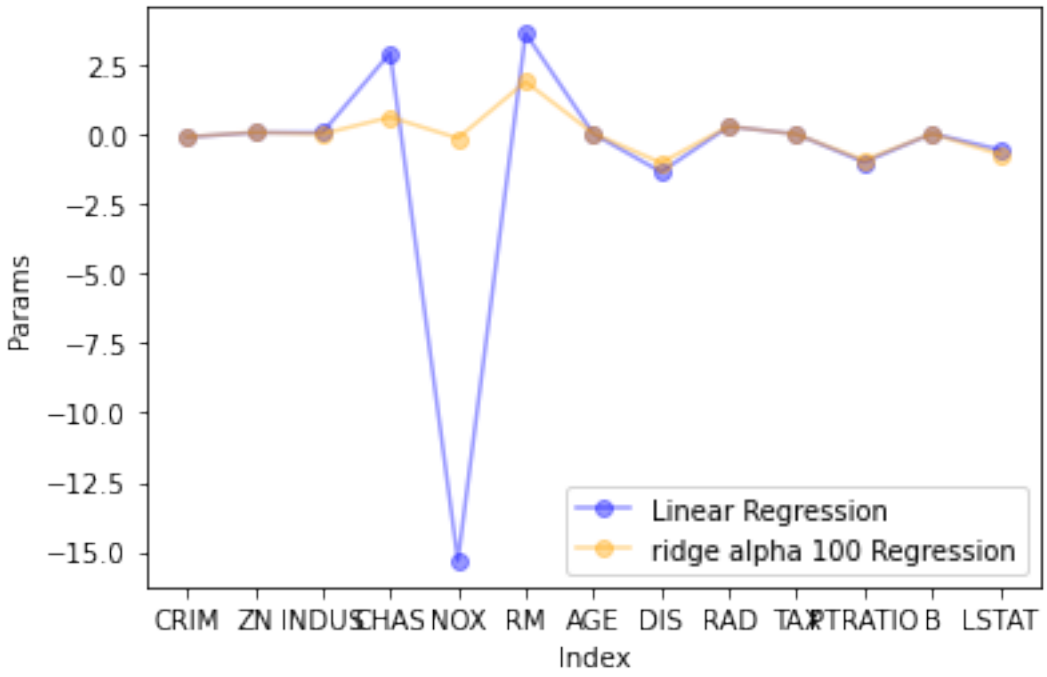
[87]:
```
  features coef_linear_regressor coef_abs_linear_regressor  \
0     CRIM             -0.112694                   0.112694
1       ZN              0.046821                   0.046821
2    INDUS               0.04475                    0.04475
```

```
3     CHAS              2.907119                   2.907119
4      NOX            -15.311814                  15.311814

  coef_ridge_small_alpha coef_abs_ridge_small_alpha coef_ridge_large_alpha  \
0              -0.112616                   0.112616              -0.112616
1               0.046842                   0.046842               0.046842
2               0.044211                   0.044211               0.044211
3               2.907098                   2.907098               2.907098
4             -15.163059                  15.163059             -15.163059

  coef_abs_ridge_large_alpha
0                   0.112616
1                   0.046842
2                   0.044211
3                   2.907098
4                  15.163059
```

[88]:
```
import matplotlib.pyplot as plt
plt.plot(lr_all_coeffcients['features'],df_lin.T,alpha=0.
 ↪4,marker='o',color='blue',label='Linear Regression')
plt.plot(lr_all_coeffcients['features'],df_ridge2.iloc[0].T,alpha=0.
 ↪4,marker='o',color='orange',label='ridge alpha 100 Regression')
plt.xlabel('Index')
plt.ylabel('Params')
plt.legend()
plt.show()
```



[ ]:

[ ]: