grebena_regresija

August 23, 2017

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn import metrics
from sklearn import model_selection
%matplotlib inline
```

In [2]:
```
# radimo sa podacima koji se ticu bejzbol igraca
# hteli bismo na osnovu svih statistika o performansama igraca da predvidi
# podaci koji se koriste su preuzeti iz knjige Introduction to Statisctica
# link do knjige je: http://www-bcf.usc.edu/~gareth/ISL/
data = pd.read_csv('hitters.csv')
```

In [3
```
# skup sadrzi 21 atribut (ukljucujuci i platu koju zelimo da predvidimo)
# i 322 informacije koje se ticu pojedinacnih igraca
data.shape
```

Out[3

In [4]:
```
data.head(5)
```

Out[4]:

	Player	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat
0	-Andy Allanson	293							
1	-Alan Ashby	3							
2	-Alvin Davis	479	13						
3									
4	-Andres Galarraga	3							

	CHits	...	CRuns	CRBI	CWalks	League	Division	PutOuts	Assists
0	66	...	3						
1	83								
2	457	...	224	266	263				
3									
4	101	...	48	46	3				

	Errors	Salary	NewLeague
0	20	NaN	A

```
1      10   475.0          N
2      14   480.0          A
3
4       4    91.5          N

[5 rows x 21 columns]
```

```
In [5]: # ispitujemo prirodu podataka koji se nalaze u tabeli
        data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3
Data columns (total 21 columns):
Player       3
AtBat        3
Hits         3
HmRun        3
Runs         3
RBI          3
Walks        3
Years        3
CAtBat       3
CHits        3
CHmRun       3
CRuns        3
CRBI         3
CWalks       3
League       3
Division     3
PutOuts      3
Assists      3
Errors       3
Salary       263
NewLeague    3
dtypes: float64(1), int64(16), object(4)
memory usage: 52.9+ KB
```

```
In [6]: # neke od vrsta zadrze nedostajuce vrednosti tako da nam nisu podesne u mod
        data = data.dropna()
```

```
In [7]: data.shape
```

```
Out[7]: (263
```

```
In [8]: # informacije o platama su nam potrebne za predvidjanje kao posebna velicin
        Y = data['Salary']
```

```
In [10]: # neki atributi nam nisu informativni - kategorickog su tipa, a ne numeric
          # to su Player, League, Division, NewLeague
```

```
        # Salary takodje treba ukloniti
        X = data.drop(['Player', 'League', 'Division', 'NewLeague', 'Salary'], axi
        print(X)
```

```
     AtBat  Hits  HmRun  Runs  RBI  Walks  Years  CAtBat  CHits  CHmRun  \
1        3
2      479    13
3
4        3
5      594   169      4    74   51      3
6      185     3
7      298    73
8        3
9      401    92     17    49   66     65     13
10     574   159     21   107   75     59     10    463
11     202    53
12     418   113
13
14     196    43
16     568   158     20    89   75     73
17     190    46      2    24    8     15      5     479    102       5
19     127     3
20     413
21     426   109      3
23
24     629   168     18    73
25     587   163
26       3
27     474   129     10    50   56     40     10    23
28     550   152      6    92   3
29     513
3
3
3
3
..     ...   ...    ...   ...  ...    ...    ...     ...    ...     ...
287    687   213
288      3
289    263
290    642   211     14   107   59     52      5    23
291    265    68      8    26   3
293
294    520   120     17    53
295     19     4      1     2    3
296    205    43
297    193
299    213
3
```

3
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3

	CRuns	CRBI	CWalks	PutOuts	Assists	Errors
1	3					
2	224	266	263			
3						
4	48	46	3			
5	501	3				
6	3					
7	41	3				
8	3					
9	784	890	866	0	0	0
10	702	504	488	23		
11	192	186	161	3		
12	205	204	203			
13						
14	3					
16	1045	993				
17	65	23				
19	67	82	56	202	22	2
20	72	48	65	280	9	5
21	55	43				
23						
24	1008	1072	402	1067	157	14
25	442	198	3			
26	291	108	180	222	3	
27	246	3				
28	3					
29	763					

3
3

```
3
3
..     ...   ...     ...      ...      ...     ...
287    196   13
288    207   162     198      209      246      3
289     83
290    3
291    13
293
294    106    80      52       70      144      11
295      2     3
296    105    99      71      13
297    129   13
299    488   491     244      178       45       4
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3

[263


In [13

Out[13

In [11]: # posto su skale vrednosti podataka neujednacene, ima smisla raditi normal

In [12]: # delimo skup podataka na:
         # trening skup
         # validacioni skup koji koristimo za ucenje metaparametara
         # test skup na kojem ocenjujemo uspesnost modela
```

```
X_train_validation, X_test, Y_train_validation, Y_test = model_selection.t
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_
```

In [13
```
# baseline - linearna regresija
linreg = linear_model.LinearRegression(normalize=True)
linreg.fit(X_train_validation, Y_train_validation)
Y_predicted = linreg.predict(X_test)
linreg_score = metrics.mean_squared_error(Y_test, Y_predicted)
linreg_score
```

Out[13

In [14]:
```
# grebena regresija sa metaparametrom 4
reg_1= linear_model.Ridge(alpha=4, normalize=True)
reg_1.fit(X_train_validation, Y_train_validation)
Y_predicted = reg_1.predict(X_test)
reg_score_1 = metrics.mean_squared_error(Y_test, Y_predicted)
reg_score_1
# na osnovu ovog rezultata zakljucujemo da ima smisla raditi regularizaci
```

Out[14]: 126928.00669545057

In [15]:
```
# grebena regresija sa metaparametrom 10^10
reg_2= linear_model.Ridge(alpha=10**10, normalize=True)
reg_2.fit(X_train_validation, Y_train_validation)
Y_predicted = reg_2.predict(X_test)
reg_score_2 = metrics.mean_squared_error(Y_test, Y_predicted)
reg_score_2
# na osnovu ovog rezultata zakljucujemo da ima smisla odrediti i precizno
# jer ocigledno nisu sve regularizacije dobre
```

Out[15]: 19483

In [45]:
```
# skup vrednosti iz kojeg trazimo najbolju vrednost za parametar alfa
# imamo i male i velike vrednosti u skupu
alphas = 10**np.linspace(10, -2, 100)*0.5
```

In [46]:
```
alphas
```

Out[46]:
```
array([  5.00000000e+09,   3
         2.1643
         9.3
         4.05565415e+08,   3
         1.75559587e+08,   1.3
         7.59955541e+07,   5.74878498e+07,   4.3
         3
         1.42401793
         6.16423
         2.6683
```

```
In [50]: # trazimo koja je to vrednost

In [51]: type(errors)

Out[51]: list

In [52]: erros = np.array(errors)

In [53

Out[53

In [54]: alpha = alphas[74]
         alpha

Out[54]: 5.3

In [55]: # greska modela
         errors[74]

Out[55]: 124988.0740587797

In [56]: # sada kada znamo najbolju vrednost metaparametra alfa mozemo da naucimo i
         # da damo konacnu procenu greske
         # ne zaboraviti da se u ovom koraku koriste i trening i validacioni skup

         regfin = linear_model.Ridge(alpha=5.3              True)
```

```
regfin.fit(X_train_validation, Y_train_validation)
Y_predicted = regfin.predict(X_test)
regfin_score = metrics.mean_squared_error(Y_test, Y_predicted)
regfin_score
```

Out[56]: 13

In [ ]: