exo01

September 27, 2022

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
import pandas as pd
```

[2]:
```
reg = linear_model.LinearRegression()
```

[3]:
```
reg.fit([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
```

[3]:
```
LinearRegression()
```

[4]:
```
print(reg.intercept_, reg.coef_)
```

```
1.1102230246251565e-16 [0.5 0.5]
```

[5]:
```
df = pd.read_csv("https://www-lisic.univ-littoral.fr/~verel/TEACHING/22-23/
↪apprentissage-automatique-avance-M2/data01/cars.csv")
```

[6]:
```
df.head()
```

[6]:

	speed	dist
0	4	2
1	4	10
2	7	4
3	7	22
4	8	16

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   speed   50 non-null     int64
 1   dist    50 non-null     int64
dtypes: int64(2)
memory usage: 928.0 bytes
```

[8]:
```
# basic descriptive statistics
df.describe()
```

[8]:

	speed	dist
count	50.000000	50.000000
mean	15.400000	42.980000
std	5.287644	25.769377
min	4.000000	2.000000
25%	12.000000	26.000000
50%	15.000000	36.000000
75%	19.000000	56.000000
max	25.000000	120.000000

[9]:
```
# co-variance matrix
np.cov(df.speed, df.dist)
```

[9]:
```
array([[ 27.95918367, 109.94693878],
       [109.94693878, 664.06081633]])
```

[10]:
```
# co-variance matrix from pandas
df.cov()
```

[10]:

	speed	dist
speed	27.959184	109.946939
dist	109.946939	664.060816

[11]:
```
# correlation matrix
df.corr()
```

[11]:

	speed	dist
speed	1.000000	0.806895
dist	0.806895	1.000000

[12]:
```
reg_cars = linear_model.LinearRegression()
```

[17]:
```
#x = [ [x] for x in df.speed ]
X = df.drop(['dist'], axis = 1)
y = np.array(df.dist)
```

[18]:
```
print(X, y)
```

```
    speed
0       4
1       4
2       7
3       7
4       8
```

```
5        9
6       10
7       10
8       10
9       11
10      11
11      12
12      12
13      12
14      12
15      13
16      13
17      13
18      13
19      14
20      14
21      14
22      14
23      15
24      15
25      15
26      16
27      16
28      17
29      17
30      17
31      18
32      18
33      18
34      18
35      19
36      19
37      19
38      20
39      20
40      20
41      20
42      20
43      22
44      23
45      24
46      24
47      24
48      24
49      25 [  2  10   4  22  16  10  18  26  34  17  28  14  20  24  28  26  34
34
  46  26  36  60  80  20  26  54  32  40  32  40  50  42  56  76  84  36
  46  68  32  48  52  56  64  66  54  70  92  93 120  85]
```

[19]:
```
reg_cars.fit(X, y)
```

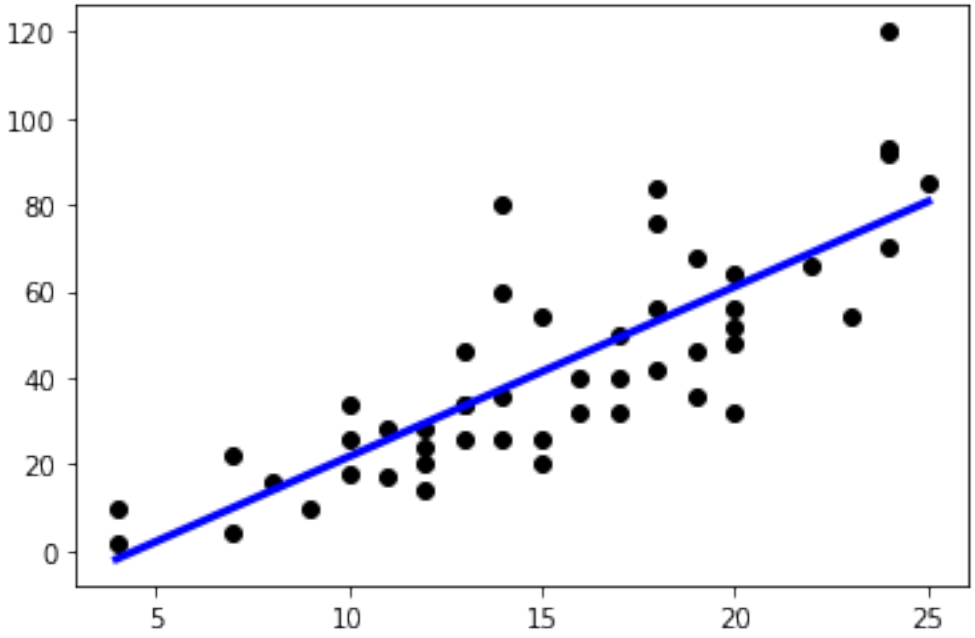
[19]:
```
LinearRegression()
```

[20]:
```
print(reg_cars.intercept_, reg_cars.coef_)
y_pred = reg_cars.predict(X)
```

```
-17.57909489051095 [3.93240876]
```

[22]:
```
# Plot outputs
plt.scatter(X, y,  color='black')
plt.plot(X, y_pred, color='blue', linewidth=3)

plt.show()
```



[23]:
```
dfb = pd.read_csv("https://www-lisic.univ-littoral.fr/~verel/TEACHING/22-23/
↪apprentissage-automatique-avance-M2/data01/basketData09.csv", sep = ' ')
```

[24]:
```
dfb.head()
```

[24]:

	height	weight	fielGoal	freeThrows	points
0	6.8	225	0.442	0.672	9.2
1	6.3	180	0.435	0.797	11.7
2	6.4	190	0.456	0.761	15.8
3	6.2	180	0.416	0.651	8.6
4	6.9	205	0.449	0.900	23.2

[25]:
```
dfb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54 entries, 0 to 53
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   height      54 non-null     float64
 1   weight      54 non-null     int64
 2   fielGoal    54 non-null     float64
 3   freeThrows  54 non-null     float64
 4   points      54 non-null     float64
dtypes: float64(4), int64(1)
memory usage: 2.2 KB
```

[26]:
```
dfb.describe()
```

[26]:

	height	weight	fielGoal	freeThrows	points
count	54.000000	54.000000	54.000000	54.000000	54.000000
mean	6.587037	209.907407	0.449111	0.741852	11.790741
std	0.458894	30.265036	0.056551	0.100146	5.899257
min	5.700000	105.000000	0.291000	0.244000	2.800000
25%	6.225000	185.000000	0.415250	0.713000	8.150000
50%	6.650000	212.500000	0.443500	0.753500	10.750000
75%	6.900000	235.000000	0.483500	0.795250	13.600000
max	7.600000	263.000000	0.599000	0.900000	27.400000

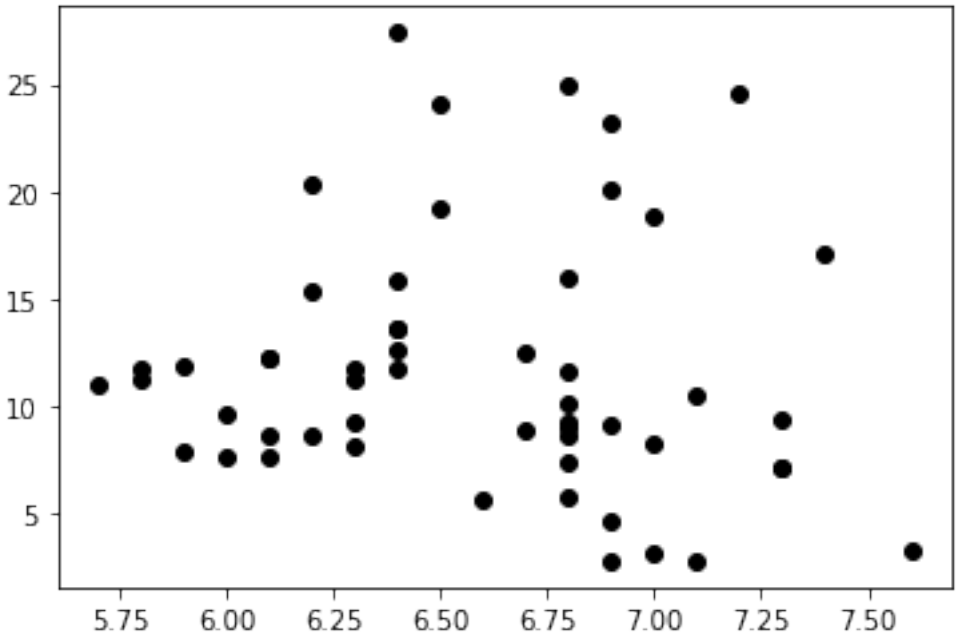
[27]:
```
dfb.corr()
```

[27]:

	height	weight	fielGoal	freeThrows	points
height	1.000000	0.834324	0.495546	-0.259271	-0.068906
weight	0.834324	1.000000	0.516051	-0.290159	-0.009844
fielGoal	0.495546	0.516051	1.000000	-0.018570	0.338760
freeThrows	-0.259271	-0.290159	-0.018570	1.000000	0.244852
points	-0.068906	-0.009844	0.338760	0.244852	1.000000

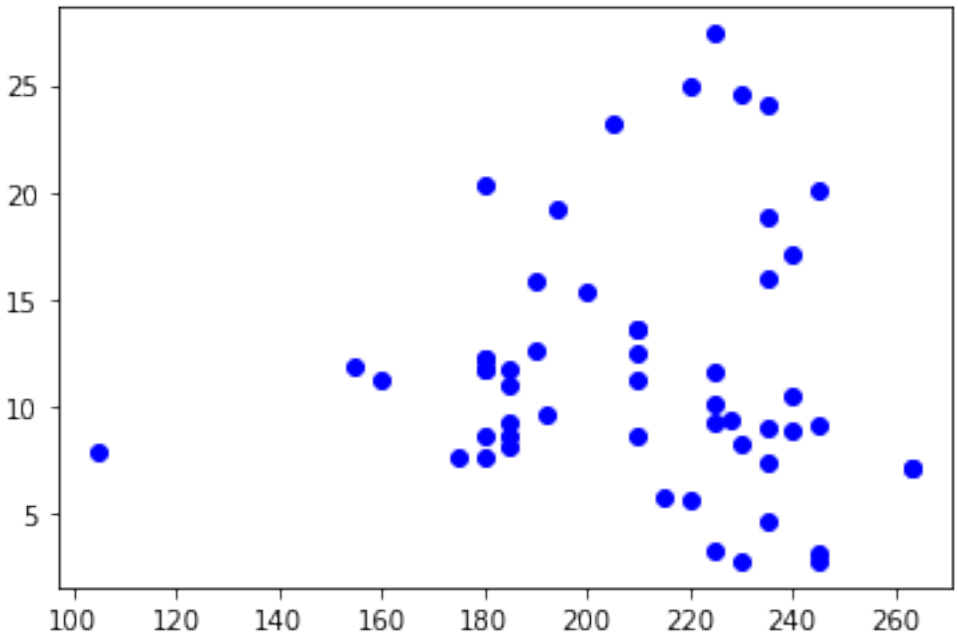
[28]:
```
plt.scatter(dfb.height, dfb.points,  color='black')

plt.show()
```



[29]:
```
plt.scatter(dfb.weight, dfb.points,  color='blue')

plt.show()
```



[30]:
```
reg_b = linear_model.LinearRegression()
```

[31]:
```
#x = [ [x1, x2] for x1, x2 in zip(dfb.weight, dfb.height) ]
X = dfb.drop(['fielGoal', 'freeThrows', 'points'], axis=1)
y = np.array(dfb.points)
```

[32]:
```
reg_b.fit(X, y)
```

[32]:
```
LinearRegression()
```

[33]:
```
y_pred = reg_b.predict(X)
```

[34]:
```
# Root mean square error of the multilinear model:
np.sqrt(np.mean((y - y_pred)**2))
```

[34]:
```
5.808565800733839
```

[45]:
```
# mean absolute error of the multilinear model:
np.mean(np.abs((y - y_pred)))
```

[45]:
```
4.435710332863232
```

[35]:
```
print(reg_b.intercept_, reg_b.coef_)
```

```
22.287426326843295 [-2.56737625  0.03055974]
```

Notice that: increasing the average points decreases with the increasing of the height, and the opposite for weight feature. But, it is not possible to compare the value of the coefficients directement. They must be rescale.

[36]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

[37]:
```
pipe = Pipeline([('scaler', StandardScaler()), ('linear', linear_model.
↪LinearRegression())])
```

[38]:
```
pipe.fit(X, y)
```

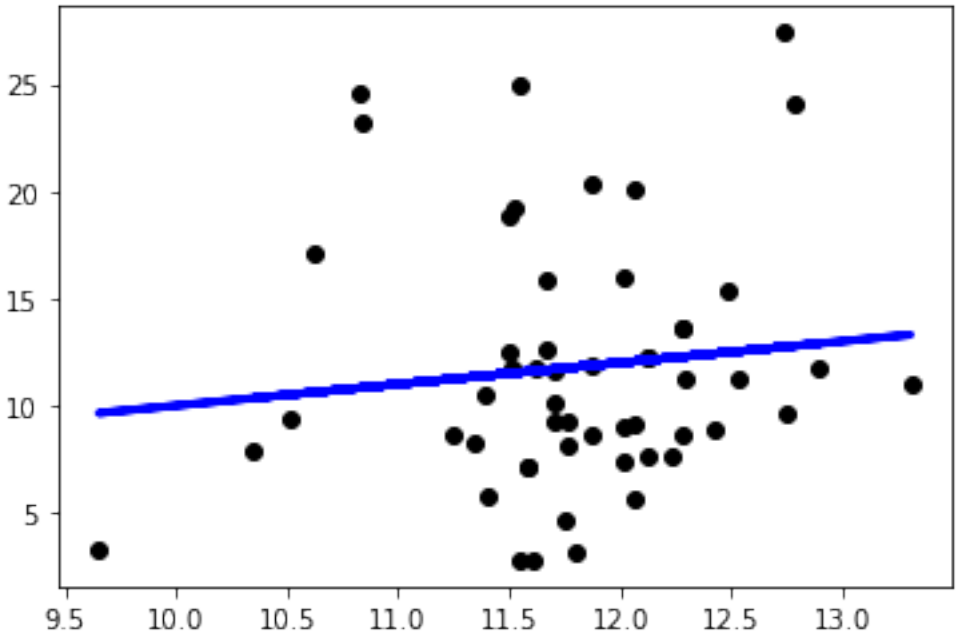
[38]:
```
Pipeline(steps=[('scaler', StandardScaler()), ('linear', LinearRegression())])
```

[39]:
```
y_pred = pipe.predict(X)
print(np.sqrt(np.mean((y - y_pred)**2)))
```

```
5.808565800733839
```

[49]:
```
plt.scatter(y_pred, y,  color='black')
plt.plot(y_pred, y_pred, color='blue', linewidth=3) # y = x line
```

```
plt.show()
```



[40]:
```
print(pipe[-1].intercept_, pipe[-1].coef_)
```

```
11.790740740740738 [-1.16719321  0.91628795]
```

Of course, with a linear transformation of predictors, the quality is exactly the same. But now, it is possible to fairly compare the value of coefficients. The variable ”height” has nearly the same impact of the prediction than the variable ”weight”. They only differ from factor of 1 - 0.91628795 / 1.16719321 = 21.5

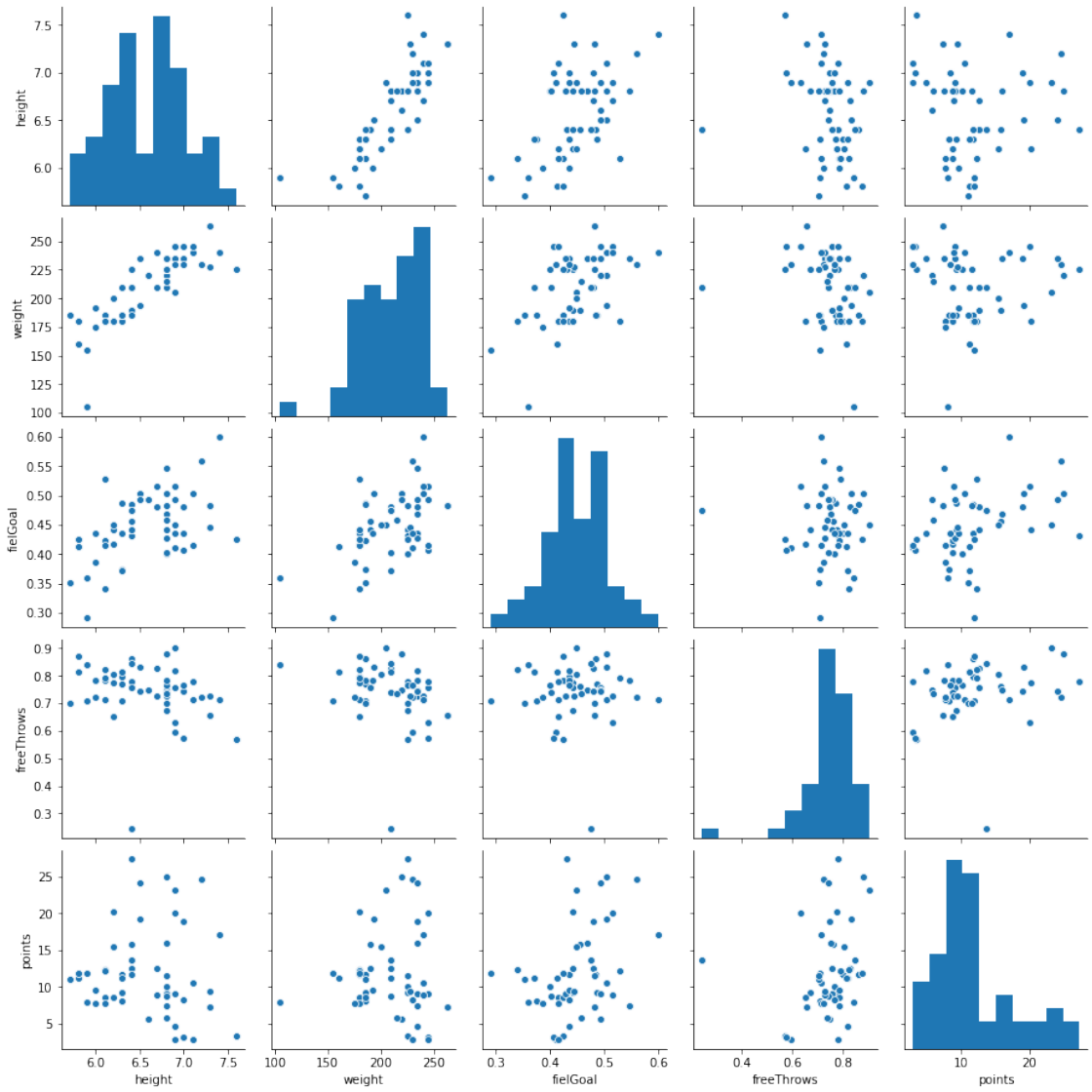
[41]:
```
0.91628795 / 1.16719321
```

[41]:
```
0.7850353670237681
```

Extra: scatterplot matrix of features (pair plot). Have fun with it by modifying the style...

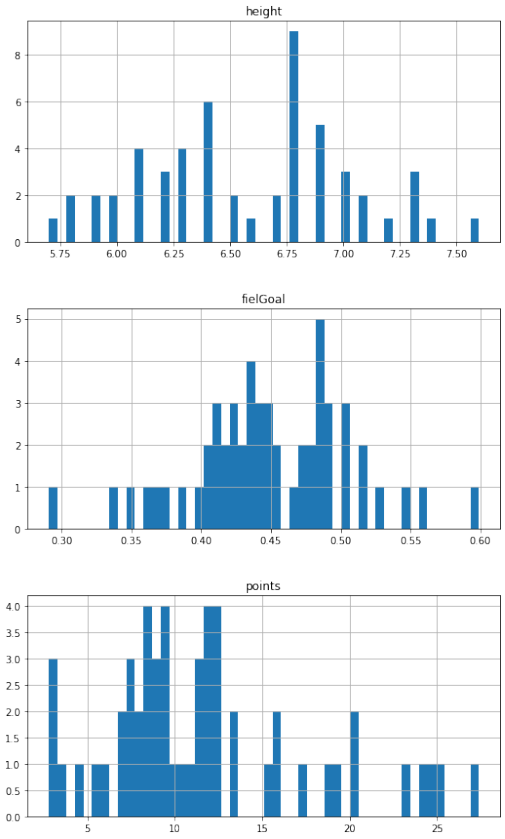
[42]:
```
# see: https://seaborn.pydata.org/generated/seaborn.pairplot.html
import seaborn as sns
sns.pairplot(dfb)
```

[42]:
```
<seaborn.axisgrid.PairGrid at 0x7fe8b01869a0>
```

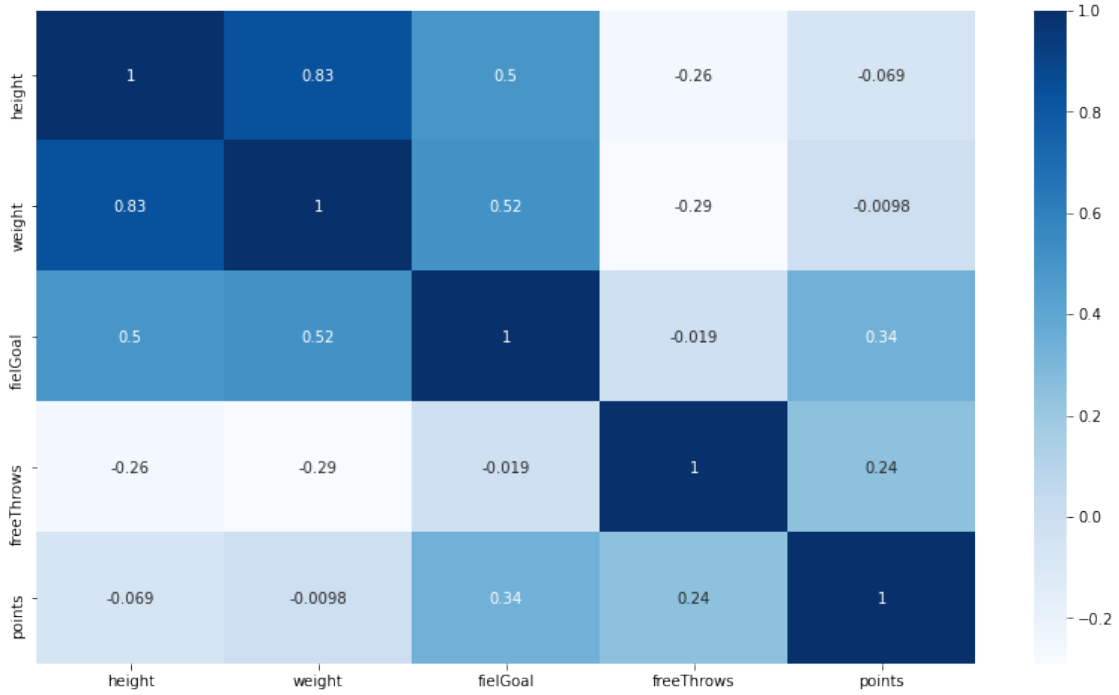


[43]:
```
dfb.hist(bins=50, figsize=(20,15))
```

[43]:
```
array([[<AxesSubplot:title={'center':'height'}>,
        <AxesSubplot:title={'center':'weight'}>],
       [<AxesSubplot:title={'center':'fielGoal'}>,
        <AxesSubplot:title={'center':'freeThrows'}>],
       [<AxesSubplot:title={'center':'points'}>, <AxesSubplot:>]],
      dtype=object)
```



[44]:
```
plt.figure(figsize=(14,8))
corr = dfb.corr()
heatmap = sns.heatmap(corr, annot=True, cmap="Blues")
```


height
weight
fielGoal
freeThrows
points
1
0.83
0.5
-0.26
-0.069
0.83
1
0.52
-0.29
-0.0098
0.5
0.52
1
-0.019
0.34
-0.26
-0.29
-0.019
1
0.24
-0.069
-0.0098
0.34
0.24
1
height
weight
fielGoal
freeThrows
points
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

[ ]: