AI - Fatec Aula 2018-10-30 Prof. Walmir

In [ ]:

```
"""
INSTRUÇÕES
~~~~~~~~~~
cd ~/workspacepy
git clone https://github.com/huogerac/machinelearning-class.git

# Cria/ativa virtual env para não zuar o ambiente local da máquina
# --> https://virtualenv.pypa.io/en/latest/userguide/
virtualenv linearregression_env
source linearregression_env/bin/activate

# Instala as dependencias para rodar este Jupyter Notebook
~/workspacepy/machinelearning-class
pip install -r requirements.txt
jupyter notebook
--> só navegar e abrir este notebook linearregression/LinearRegression_01_aula_3
0out_houses.ipynb
"""
```

In [1]:

```
import pandas as pd
import seaborn as sns
import numpy as np

from sklearn import linear_model
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

```
/home/roger/.ve/fatec_decisiontree/lib/python3.5/site-packages/sklea
rn/utils/fixes.py:313: FutureWarning: numpy not_equal will not check
object identity in the future. The comparison did not return the sam
e result as suggested by the identity (`is`)) and will change.
  _nan_object_mask = _nan_object_array != _nan_object_array
```

In [2]:

```
# dados originais
houses = pd.read_csv('house.csv')
houses
```

Out[2]:

	houseSize	lotSize	bedrooms	granite	bathroom	sellingPrice
0	3529	9191	6	0	0	205000
1	3247	10061	5	1	1	224900
2	4032	10150	5	0	1	197900
3	2397	14156	4	1	0	189900
4	2200	9600	4	0	1	195000
5	3536	19994	6	1	1	325000
6	2983	9365	5	0	1	230000

In [3]:

```
# um pouco mais de informações sobre os dados
houses.describe()
```

Out[3]:

	houseSize	lotSize	bedrooms	granite	bathroom	sellingPrice
count	7.000000	7.000000	7.000000	7.000000	7.000000	7.000000
mean	3132.000000	11788.142857	5.000000	0.428571	0.714286	223957.142857
std	655.120854	4000.294049	0.816497	0.534522	0.487950	47052.837574
min	2200.000000	9191.000000	4.000000	0.000000	0.000000	189900.000000
25%	2690.000000	9482.500000	4.500000	0.000000	0.500000	196450.000000
50%	3247.000000	10061.000000	5.000000	0.000000	1.000000	205000.000000
75%	3532.500000	12153.000000	5.500000	1.000000	1.000000	227450.000000
max	4032.000000	19994.000000	6.000000	1.000000	1.000000	325000.000000

In [4]:

```
# coeficiente de relação (quanto mais próximo de 1.0 ou -1.0, melhor atributo)
houses.corr()
```

Out[4]:

	houseSize	lotSize	bedrooms	granite	bathroom	sellingPrice
houseSize	1.000000	0.080843	0.768985	-0.102805	0.176225	0.330205
lotSize	0.080843	1.000000	0.277027	0.689550	0.019578	0.785860
bedrooms	0.768985	0.277027	1.000000	0.000000	0.000000	0.629471
granite	-0.102805	0.689550	0.000000	1.000000	-0.091287	0.450142
bathroom	0.176225	0.019578	0.000000	-0.091287	1.000000	0.384840
sellingPrice	0.330205	0.785860	0.629471	0.450142	0.384840	1.000000

In [5]:

```
# Campos relavantes para o treinamento
cols = ['houseSize', 'lotSize', 'bedrooms', 'bathroom']
# Campo para predição
cols_target = ['sellingPrice']

regression = linear_model.LinearRegression()
x_train, x_test, y_train, y_test = train_test_split(
    houses[cols], houses[cols_target], test_size=0.2, random_state=4)
```

In [6]:

```
x_train
```

Out[6]:

	houseSize	lotSize	bedrooms	bathroom
3	2397	14156	4	0
0	3529	9191	6	0
1	3247	10061	5	1
5	3536	19994	6	1
2	4032	10150	5	1

In [7]:

```
x_test
```

Out[7]:

	houseSize	lotSize	bedrooms	bathroom
4	2200	9600	4	1
6	2983	9365	5	1

In [8]:

```
y_train
```

Out[8]:

	sellingPrice
3	189900
0	205000
1	224900
5	325000
2	197900

In [9]:

```
y_test
```

Out[9]:

	sellingPrice
4	195000
6	230000

In [10]:

```
# Executa Treinamento com 80% dos dados disponíveis
regression.fit(x_train, y_train)

# Faz previsão dos 20% dos dados que não entraram no treinamento
output = regression.predict(x_test)
output
```

Out[10]:

```
array([[ 214624.13699031],
       [ 229537.48206201]])
```

In [11]:

```
# Verifica a qualidade da previsão
score = r2_score(y_test, output)
score
```

Out[11]:

```
0.37090501966120515
```

In [12]:

```
# Usa o treinamento para fazer uma previsão de um dado novo
df_new_house_info = pd.DataFrame(
    [(2100, 9600, 3, 1)], columns=['houseSize', 'lotSize', 'bedrooms','bathroom'
])

output2 = regression.predict(df_new_house_info)
output2
```

Out[12]:

```
array([[ 174135.0822226]])
```