Curso de Data Science

Prof. MSc. Eng. Marcelo Bianchi

Trabalho 1

Grupo 4

Integrantes: Daniel Moreira, Lia Morimoto, Raphael Bezerra, Thainan Abreu

Parte 2: Regressão Linear Múltipla

Predição: Preço das casas

Importando as bibliotecas

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

1) Importar o Dataset e aplicar a técnica **Missing Data**

In [2]:

```
df2 = pd.read_csv('dataset2.csv')
df2.head()
```

Out[2]:

	bedrooms	bathrooms	floors	condition	grade	yr_built	yr_renovated	zipcode	Location	House price
0	3	1	1	3	7	1955	0	98178	Paulista	221900
1	3	225	2	3	7	1951	1991	98125	Aclimação	538000
2	2	1	1	3	6	1933	0	98028	Campo Belo	180000
3	4	3	1	5	7	1965	0	98136	Mooca	604000
4	3	2	1	3	8	1987	0	98074	Paulista	510000

In [3]:

```
# renomeando as colunas para português
df2.columns = ["quartos",'banheiros',"andares","condicao","avaliacao","construcao","ano_reforma","CEP","local","preco"]
df2.head()
```

Out[3]:

	quartos	banheiros	andares	condicao	avaliacao	construcao	ano_reforma	CEP	local	preco
0	3	1	1	3	7	1955	0	98178	Paulista	221900
1	3	225	2	3	7	1951	1991	98125	Aclimação	538000
2	2	1	1	3	6	1933	0	98028	Campo Belo	180000
3	4	3	1	5	7	1965	0	98136	Mooca	604000
4	3	2	1	3	8	1987	0	98074	Paulista	510000

3) Aplicar Feature Scaling (Se for aplicável, se não for então justificar)

In [4]:

```
# Ajustando a escala da variável dependente por um fator de 10.000
# por simplicidade e para manter a coerencia entre ordens de grandeza das variáveis independentes e da saída
df2["preco"] = df2["preco"]/10000
df2 = df2.rename(columns={"preco":"preco(*10k)"})
df2.head()
```

Out[4]:

	quartos	banheiros	andares	condicao	avaliacao	construcao	ano_reforma	CEP	local	preco(*10k)
0	3	1	1	3	7	1955	0	98178	Paulista	22.19
1	3	225	2	3	7	1951	1991	98125	Aclimação	53.80
2	2	1	1	3	6	1933	0	98028	Campo Belo	18.00
3	4	3	1	5	7	1965	0	98136	Mooca	60.40
4	3	2	1	3	8	1987	0	98074	Paulista	51.00

In [5]:
```
#descartando os dois valores extremos na variável dependente

df2_precos_out = df2[df2['preco(*10k)'] > 100 ]
df2 = df2.drop(df2_precos_out.index, axis=0)
```

A coluna CEP não traz a princípio nenhuma informação extra e seu tratamento como variável categórica geraria diversas outras dummy variables. Logo sera dropada.

In [6]:
```
df2.drop(columns = ['CEP'], inplace = True)
df2.head()
```

Out[6]:

	quartos	banheiros	andares	condicao	avaliacao	construcao	ano_reforma	local	preco(*10k)
0	3	1	1	3	7	1955	0	Paulista	22.19
1	3	225	2	3	7	1951	1991	Aclimação	53.80
2	2	1	1	3	6	1933	0	Campo Belo	18.00
3	4	3	1	5	7	1965	0	Mooca	60.40
4	3	2	1	3	8	1987	0	Paulista	51.00

A coluna "ano_reforma" não oferece, a princípio, nenhuma informação útil. Dessa forma, propõe-se um indicador "anos_sem_reforma" tomando como base nosso ano atual (2021) e o ano da última reforma ou, caso não haja, o ano de contrução

In [7]:
```
df2["ano_reforma"] = 2021- (df2[["ano_reforma", "construcao"]].max(axis=1))
df2["ano_reforma"].head()
```

Out[7]:
```
0    66
1    30
2    88
3    56
4    34
Name: ano_reforma, dtype: int64
```

O indicador "anos_sem_reforma" substitui a coluna "ano_reforma"

In [8]:
```
df2 = df2.rename(columns={"ano_reforma":"anos_sem_reforma"})
df2.head()
## construção do indicador relativo ao número de anos desde a última reforma
```

Out[8]:

	quartos	banheiros	andares	condicao	avaliacao	construcao	anos_sem_reforma	local	preco(*10k)
0	3	1	1	3	7	1955	66	Paulista	22.19
1	3	225	2	3	7	1951	30	Aclimação	53.80
2	2	1	1	3	6	1933	88	Campo Belo	18.00
3	4	3	1	5	7	1965	56	Mooca	60.40
4	3	2	1	3	8	1987	34	Paulista	51.00

In [9]:
```
X = df2.iloc[:, :-1]  #variáveis independentes
y = df2.iloc[:, 8]    # variável dependente
```

In [10]:
```
X.head()
```

Out[10]:

	quartos	banheiros	andares	condicao	avaliacao	construcao	anos_sem_reforma	local
0	3	1	1	3	7	1955	66	Paulista
1	3	225	2	3	7	1951	30	Aclimação
2	2	1	1	3	6	1933	88	Campo Belo
3	4	3	1	5	7	1965	56	Mooca
4	3	2	1	3	8	1987	34	Paulista

---

4) Aplicar Dummy Variable (Se for aplicável, se não for então justificar)

In [11]:
```
# codificando a variável categórica 'local'

from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
ct = ColumnTransformer([("Location",OneHotEncoder(),[7])], remainder = 'passthrough')
X = ct.fit_transform(X)

# Avoiding the Dummy Variable Trap
X = X[:, 1:]
```

---

2) Dividir o dataset entre o Training Set e o Test Set

In [12]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

5) Aplicar a Multiple Linear Regression

In [13]:

```
# Construindo o modelo ótimo para a regressão múltipla usando eliminação retroativa
# import statsmodels.formula.api as sm
import statsmodels.regression.linear_model as sm

X= np.append(arr = np.ones((26,1)).astype(int), values = X, axis =1)
X_opt = X[:,[0,1,2,3,4,5,7,8,9,10]]
X_opt = np.array(X_opt, dtype=float)

regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
#Eliminar o maior P-valor do sumário a cada etapa até restar o index e a coluna de interesse (index+1) em X
```

Out[13]:

OLS Regression Results

Dep. Variable:	preco(*10k)	**R-squared:**	0.564
Model:	OLS	**Adj. R-squared:**	0.319
Method:	Least Squares	**F-statistic:**	2.300
Date:	Tue, 16 Mar 2021	**Prob (F-statistic):**	0.0701
Time:	18:52:14	**Log-Likelihood:**	-101.94
No. Observations:	26	**AIC:**	223.9
Df Residuals:	16	**BIC:**	236.5
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	421.4887	307.228	1.372	0.189	-229.806	1072.783
x1	2.6982	13.286	0.203	0.842	-25.467	30.863
x2	-7.7939	12.753	-0.611	0.550	-34.829	19.241
x3	-4.1555	12.453	-0.334	0.743	-30.556	22.245
x4	-1.3162	11.236	-0.117	0.908	-25.135	22.502
x5	4.1977	5.388	0.779	0.447	-7.225	15.620
x6	-0.2337	0.918	-0.255	0.802	-2.180	1.713
x7	1.4377	5.253	0.274	0.788	-9.698	12.574
x8	17.0969	5.891	2.902	0.010	4.609	29.585
x9	-0.2653	0.156	-1.697	0.109	-0.597	0.066

Omnibus:	0.214	**Durbin-Watson:**	2.061
Prob(Omnibus):	0.899	**Jarque-Bera (JB):**	0.059
Skew:	0.101	**Prob(JB):**	0.971
Kurtosis:	2.885	**Cond. No.**	1.97e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.97e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [14]:

```
#Eliminada a coluna 4
X_opt = X[:,[0,1,2,3,5,7,8,9,10]]
X_opt = np.array(X_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

Out[14]:

OLS Regression Results

Dep. Variable:	preco(*10k)	**R-squared:**	0.564
Model:	OLS	**Adj. R-squared:**	0.358
Method:	Least Squares	**F-statistic:**	2.745
Date:	Tue, 16 Mar 2021	**Prob (F-statistic):**	0.0381
Time:	18:52:14	**Log-Likelihood:**	-101.95
No. Observations:	26	**AIC:**	221.9
Df Residuals:	17	**BIC:**	233.2
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]

	coef	std err	t	P>|t|	[0.025	0.975]
const	416.3096	295.079	1.411	0.176	-206.253	1038.873
x1	3.7040	9.840	0.376	0.711	-17.057	24.465
x2	-6.9128	9.995	-0.692	0.499	-28.001	14.175
x3	-3.3342	9.990	-0.334	0.743	-24.410	17.742
x4	4.2962	5.166	0.832	0.417	-6.602	15.195
x5	-0.2223	0.886	-0.251	0.805	-2.092	1.647
x6	1.3393	5.033	0.266	0.793	-9.279	11.958
x7	17.0983	5.718	2.991	0.008	5.035	29.161
x8	-0.2631	0.151	-1.747	0.099	-0.581	0.055

Omnibus:	0.190	**Durbin-Watson:**	2.072
Prob(Omnibus):	0.909	**Jarque-Bera (JB):**	0.039
Skew:	0.076	**Prob(JB):**	0.981
Kurtosis:	2.887	**Cond. No.**	1.95e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.95e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [15]:

```
#Eliminada a coluna 7
X_opt = X[:,[0,1,2,3,5,8,9,10]]
X_opt = np.array(X_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

Out[15]:

OLS Regression Results

Dep. Variable:	preco(*10k)	**R-squared:**	0.562
Model:	OLS	**Adj. R-squared:**	0.392
Method:	Least Squares	**F-statistic:**	3.300
Date:	Tue, 16 Mar 2021	**Prob (F-statistic):**	0.0194
Time:	18:52:14	**Log-Likelihood:**	-102.00
No. Observations:	26	**AIC:**	220.0
Df Residuals:	18	**BIC:**	230.1
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	377.6323	244.957	1.542	0.141	-137.002	892.267
x1	3.1639	9.348	0.338	0.739	-16.477	22.804
x2	-7.9269	8.900	-0.891	0.385	-26.625	10.771
x3	-3.9968	9.380	-0.426	0.675	-23.703	15.710
x4	3.9720	4.869	0.816	0.425	-6.258	14.202
x5	0.9979	4.718	0.212	0.835	-8.913	10.909
x6	16.9909	5.551	3.061	0.007	5.329	28.653
x7	-0.2420	0.122	-1.989	0.062	-0.498	0.014

Omnibus:	0.455	**Durbin-Watson:**	2.120
Prob(Omnibus):	0.797	**Jarque-Bera (JB):**	0.146
Skew:	0.183	**Prob(JB):**	0.930
Kurtosis:	2.967	**Cond. No.**	1.66e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.66e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [16]:

```
#Eliminada a coluna 1
X_opt = X[:,[0,2,3,5,8,9,10]]
X_opt = np.array(X_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

Out[16]:

OLS Regression Results

Dep. Variable:	preco(*10k)	**R-squared:**	0.559
Model:	OLS	**Adj. R-squared:**	0.420
Method:	Least Squares	**F-statistic:**	4.018
Date:	Tue, 16 Mar 2021	**Prob (F-statistic):**	0.00913
Time:	18:52:14	**Log-Likelihood:**	-102.08
No. Observations:	26	**AIC:**	218.2
Df Residuals:	19	**BIC:**	227.0
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	389.7734	236.602	1.647	0.116	-105.440	884.987
x1	-8.9494	8.174	-1.095	0.287	-26.058	8.159
x2	-4.6361	8.971	-0.517	0.611	-23.413	14.141
x3	3.7066	4.692	0.790	0.439	-6.115	13.528
x4	1.1937	4.572	0.261	0.797	-8.375	10.762
x5	16.7093	5.359	3.118	0.006	5.493	27.926
x6	-0.2467	0.118	-2.090	0.050	-0.494	0.000

Omnibus:	0.234	**Durbin-Watson:**	2.039
Prob(Omnibus):	0.890	**Jarque-Bera (JB):**	0.023
Skew:	0.065	**Prob(JB):**	0.989
Kurtosis:	2.934	**Cond. No.**	1.65e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.65e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [17]:

```
#Eliminada a coluna 8
X_opt = X[:,[0,2,3,5,9,10]]
X_opt = np.array(X_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

Out[17]:

OLS Regression Results

Dep. Variable:	preco(*10k)	**R-squared:**	0.558
Model:	OLS	**Adj. R-squared:**	0.447
Method:	Least Squares	**F-statistic:**	5.043
Date:	Tue, 16 Mar 2021	**Prob (F-statistic):**	0.00377
Time:	18:52:14	**Log-Likelihood:**	-102.13
No. Observations:	26	**AIC:**	216.3
Df Residuals:	20	**BIC:**	223.8
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	418.9774	203.580	2.058	0.053	-5.684	843.639
x1	-8.7708	7.953	-1.103	0.283	-25.361	7.820
x2	-3.9597	8.386	-0.472	0.642	-21.454	13.534
x3	3.5584	4.548	0.782	0.443	-5.929	13.046
x4	16.6311	5.224	3.183	0.005	5.733	27.529
x5	-0.2590	0.106	-2.452	0.024	-0.479	-0.039

Omnibus:	0.184	**Durbin-Watson:**	2.037
Prob(Omnibus):	0.912	**Jarque-Bera (JB):**	0.026
Skew:	0.057	**Prob(JB):**	0.987
Kurtosis:	2.896	**Cond. No.**	1.45e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.45e+05. This might indicate that there are strong multicollinearity or other numerical problems.

In [18]:

```
#Eliminada a coluna 3
X_opt = X[:,[0,2,5,9,10]]
X_opt = np.array(X_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

Out[18]:

OLS Regression Results

Dep. Variable:	preco(*10k)	**R-squared:**	0.553
Model:	OLS	**Adj. R-squared:**	0.468
Method:	Least Squares	**F-statistic:**	6.487
Date:	Tue, 16 Mar 2021	**Prob (F-statistic):**	0.00146
Time:	18:52:14	**Log-Likelihood:**	-102.27
No. Observations:	26	**AIC:**	214.5
Df Residuals:	21	**BIC:**	220.8
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	453.4832	186.462	2.432	0.024	65.715	841.252
x1	-7.3035	7.184	-1.017	0.321	-22.244	7.637
x2	2.4646	3.841	0.642	0.528	-5.523	10.452
x3	17.3287	4.918	3.524	0.002	7.102	27.555
x4	-0.2781	0.096	-2.904	0.008	-0.477	-0.079

Omnibus:	0.073	**Durbin-Watson:**	2.054
Prob(Omnibus):	0.964	**Jarque-Bera (JB):**	0.042
Skew:	0.001	**Prob(JB):**	0.979
Kurtosis:	2.802	**Cond. No.**	1.35e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.35e+05. This might indicate that there are strong multicollinearity or other numerical problems.

In [19]:

```
#Eliminada a coluna 5
X_opt = X[:,[0,2,9,10]]
X_opt = np.array(X_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

Out[19]:

OLS Regression Results

Dep. Variable:	preco(*10k)	**R-squared:**	0.544
Model:	OLS	**Adj. R-squared:**	0.482
Method:	Least Squares	**F-statistic:**	8.747
Date:	Tue, 16 Mar 2021	**Prob (F-statistic):**	0.000524
Time:	18:52:14	**Log-Likelihood:**	-102.52
No. Observations:	26	**AIC:**	213.0
Df Residuals:	22	**BIC:**	218.1
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	456.8788	183.878	2.485	0.021	75.540	838.218
x1	-6.1352	6.856	-0.895	0.381	-20.354	8.084
x2	18.5605	4.466	4.156	0.000	9.298	27.823
x3	-0.2806	0.094	-2.972	0.007	-0.476	-0.085

Omnibus:	0.222	**Durbin-Watson:**	2.080
Prob(Omnibus):	0.895	**Jarque-Bera (JB):**	0.004
Skew:	-0.008	**Prob(JB):**	0.998
Kurtosis:	2.945	**Cond. No.**	1.35e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.35e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [20]:

```
#Eliminada a coluna 2
X_opt = X[:,[0,9,10]]
X_opt = np.array(X_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

Out[20]:

OLS Regression Results

Dep. Variable:	preco(*10k)	**R-squared:**	0.527
Model:	OLS	**Adj. R-squared:**	0.486
Method:	Least Squares	**F-statistic:**	12.83
Date:	Tue, 16 Mar 2021	**Prob (F-statistic):**	0.000181
Time:	18:52:15	**Log-Likelihood:**	-102.99
No. Observations:	26	**AIC:**	212.0
Df Residuals:	23	**BIC:**	215.8
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	413.6313	176.642	2.342	0.028	48.219	779.044
x1	19.7819	4.234	4.672	0.000	11.022	28.541
x2	-0.2637	0.092	-2.863	0.009	-0.454	-0.073

Omnibus:	0.379	**Durbin-Watson:**	2.005
Prob(Omnibus):	0.827	**Jarque-Bera (JB):**	0.258
Skew:	0.224	**Prob(JB):**	0.879
Kurtosis:	2.805	**Cond. No.**	1.30e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.3e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [21]:

```
#Eliminada a coluna 10
X_opt = X[:,[0,9]]
X_opt = np.array(X_opt, dtype=float)
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

Out[21]:

OLS Regression Results

Dep. Variable:	preco(*10k)	**R-squared:**	0.359
Model:	OLS	**Adj. R-squared:**	0.332
Method:	Least Squares	**F-statistic:**	13.44
Date:	Tue, 16 Mar 2021	**Prob (F-statistic):**	0.00122
Time:	18:52:15	**Log-Likelihood:**	-106.95
No. Observations:	26	**AIC:**	217.9
Df Residuals:	24	**BIC:**	220.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-84.6800	34.319	-2.467	0.021	-155.511	-13.849
x1	17.3302	4.728	3.666	0.001	7.572	27.088

Omnibus:	2.454	**Durbin-Watson:**	1.607
Prob(Omnibus):	0.293	**Jarque-Bera (JB):**	1.497
Skew:	0.583	**Prob(JB):**	0.473
Kurtosis:	3.141	**Cond. No.**	84.0

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

A coluna 9 corresponde à coluna de X relativa às "avaliacoes": X_test[:,8] e X_train[:,8]

In [22]:

```
# Após a Backward Elimination, o modelo se reduz a uma regressão linear simples onde a variável independente foi a selecionada anteriormente
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train[:,8].reshape(-1,1), y_train)


# Predicting the Test set results
y_pred = regressor.predict(X_test[:,8].reshape(-1,1))
```

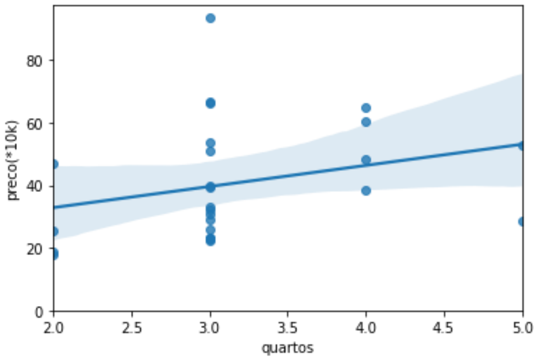
---

6) Construir o Gráfico (Scatter Plot)

In [23]:

```
# visualização exploratória das relações entre as variáveis independentes e o preço
import seaborn as sns

sns.regplot(x="quartos", y="preco(*10k)", data=df2)
plt.ylim(0,)
```

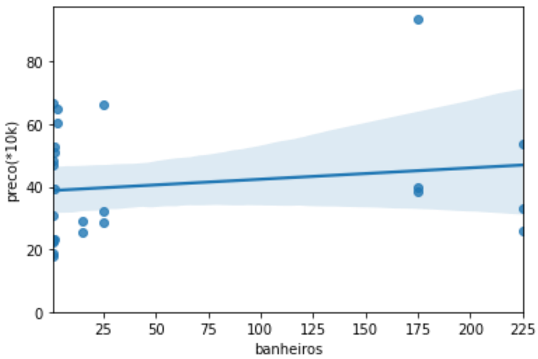
Out[23]: (0.0, 97.485)



In [24]:

```
sns.regplot(x="banheiros", y="preco(*10k)", data=df2)
plt.ylim(0,)
```

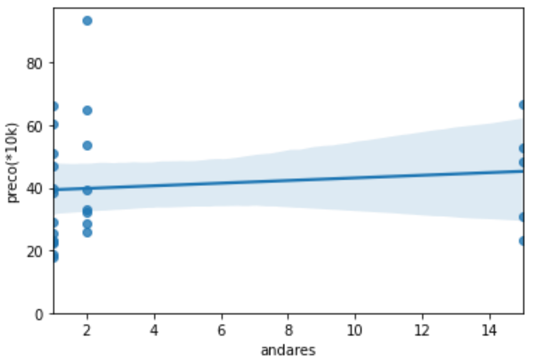
Out[24]: (0.0, 97.485)



In [25]:

```
sns.regplot(x="andares", y="preco(*10k)", data=df2)
plt.ylim(0,)
```

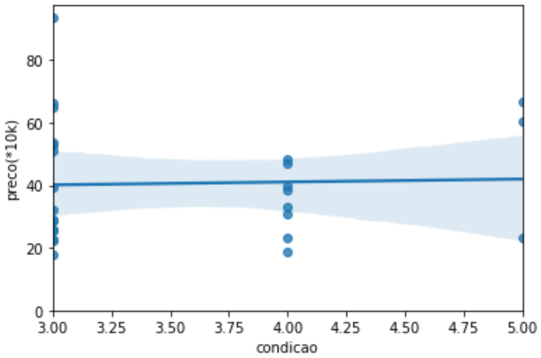
Out[25]: (0.0, 97.485)



In [26]:

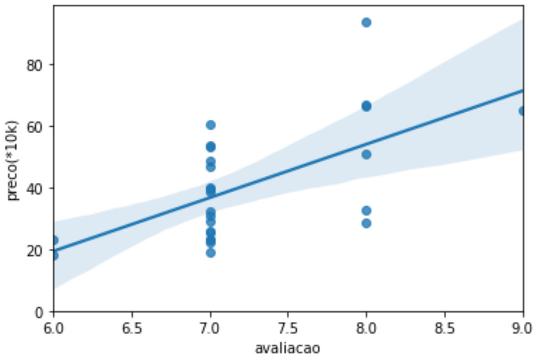
```
sns.regplot(x="condicao", y="preco(*10k)", data=df2)
plt.ylim(0,)
```

Out[26]: (0.0, 97.485)



In [27]:
```
sns.regplot(x="avaliacao", y="preco(*10k)", data=df2)
plt.ylim(0,)
```

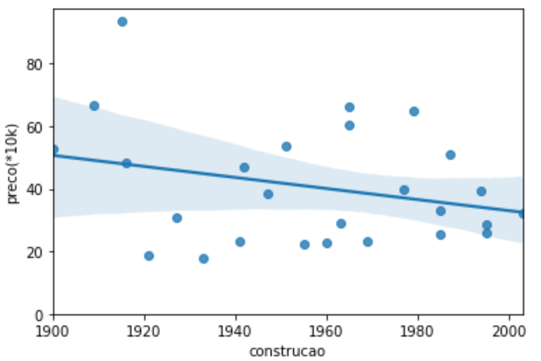
Out[27]: (0.0, 98.97129444071706)



"avaliacao" confirma-se um bom candidato à variável de interesse para o modelo de regressão

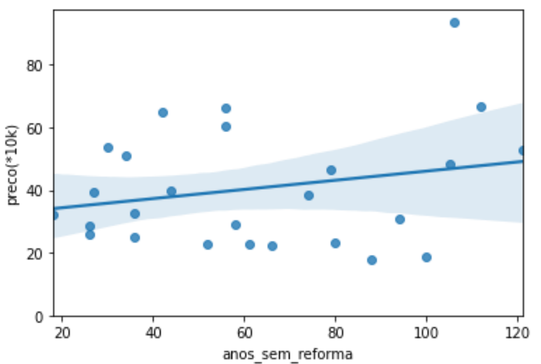
In [28]:
```
sns.regplot(x="construcao", y="preco(*10k)", data=df2)
plt.ylim(0,)
```

Out[28]: (0.0, 97.485)

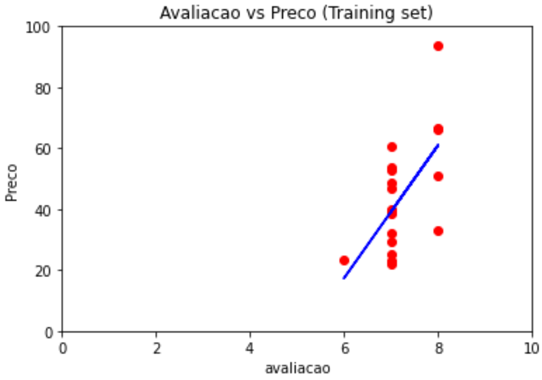


In [29]:
```
sns.regplot(x="anos_sem_reforma", y="preco(*10k)", data=df2)
plt.ylim(0,)
```

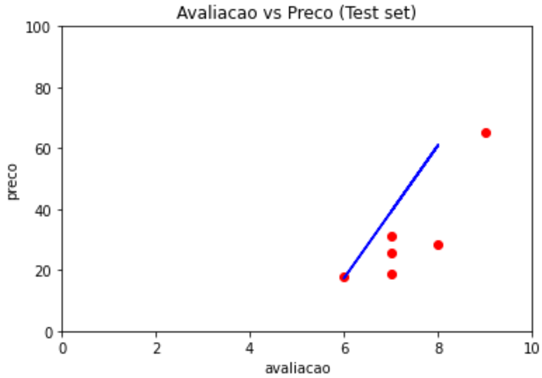
Out[29]: (0.0, 97.485)



In [30]:
```
# Visualising the Training set results
plt.scatter(X_train[:,8], y_train, color = 'red')
plt.plot(X_train[:,8], regressor.predict(X_train[:,8].reshape(-1,1)), color = 'blue')
plt.xlim(0, 10)
plt.title('Avaliacao vs Preco (Training set)')
plt.xlabel('avaliacao')

plt.ylim(0, 100)
plt.ylabel('Preco')
plt.show()
```



In [31]:
```
# Visualising the Test set results
plt.scatter(X_test[:,8], y_test, color = 'red')
plt.plot(X_train[:,8], regressor.predict(X_train[:,8].reshape(-1,1)), color = 'blue')
plt.xlim(0, 10)
plt.title('Avaliacao vs Preco (Test set)')
plt.xlabel('avaliacao')

plt.ylim(0, 100)
plt.ylabel('preco')

plt.show()
```



---

7) Criar a tabela no banco de dados SQLite

In [32]:
```
import sqlite3 as sq3
```

In [33]:
```
connection = sq3.connect('database2.db')
```

In [34]:
```
df2.to_sql(name = 'precos', con = connection, if_exists = 'append', index = False)
```

---

8) Aplicar uma consulta em linguagem SQL que irá trazer uma listagem da tabela

In [35]:
```
pd.read_sql('select * from precos', connection)
```

Out[35]:

	quartos	banheiros	andares	condicao	avaliacao	construcao	anos_sem_reforma	local	preco(*10k)
0	3	1	1	3	7	1955	66	Paulista	22.190
1	3	225	2	3	7	1951	30	Aclimação	53.800
2	2	1	1	3	6	1933	88	Campo Belo	18.000
3	4	3	1	5	7	1965	56	Mooca	60.400
4	3	2	1	3	8	1987	34	Paulista	51.000
5	3	225	2	3	7	1995	26	Mooca	25.750
6	3	15	1	3	7	1963	58	Aclimação	29.185
7	3	1	1	3	7	1960	61	Centro	22.950
8	3	25	2	3	7	2003	18	Mooca	32.300
9	3	25	1	3	8	1965	56	Paulista	66.250
10	2	1	1	4	7	1942	79	Campo Belo	46.800
11	3	1	15	4	7	1927	94	Centro	31.000
12	3	175	1	4	7	1977	44	Paulista	40.000
13	5	2	15	3	7	1900	121	Centro	53.000
14	4	3	2	3	9	1979	42	Aclimação	65.000

	quartos	banheiros	andares	condicao	avaliacao	construcao	anos_sem_reforma	local	preco(*10k)
15	3	2	2	3	7	1994	27	Campo Belo	39.500
16	4	1	15	4	7	1916	105	Mooca	48.500
17	2	1	1	4	7	1921	100	Paulista	18.900
18	3	1	1	4	7	1969	52	Paulista	23.000
19	4	175	1	4	7	1947	74	Centro	38.500
20	5	25	2	3	8	1995	26	Mooca	28.500
21	2	15	1	3	7	1985	36	Centro	25.270
22	3	225	2	4	8	1985	36	Mooca	32.900
23	3	2	15	5	6	1941	80	Centro	23.300
24	3	175	2	3	8	1915	106	Paulista	93.700
25	3	1	15	5	8	1909	112	Campo Belo	66.700

```
In [ ]:
```