AULA 7 - MODELOS DE REGRESSÃO

por **Cibele Russo**

ICMC/USP - São Carlos SP

PROGRAMA

- Modelos lineares.
- Regressão múltipla.
- Regressão multivariada.
- A qualidade do ajuste.
- Seleção de modelos.
- Análise de diagnóstico.

Referências:

- Draper, N. R., & Smith, H. (1998). Applied regression analysis. 3rd edition. Wiley.
- James, Gareth et al. (2013) An introduction to statistical learning. New York: Springer.
- Dobson, A. J.; Barnett, Adrian G. (2018). An introduction to generalized linear models. CRC press.

section*Modelos de regressão

Objetivos

Predizer Y a partir do conhecimento de variáveis em $X = x$.

Em notação matricial, um modelo linear geral é dado por

$$Y = f(X, \beta) + \epsilon,$$

em que

- Y é a **variável resposta** (vetor de variáveis aleatórias observáveis),
- X contém **variáveis preditoras** (matriz conhecida, ou seja, não-aleatória),
- β é um **vetor de parâmetros de interesse**, que queremos estimar,
- f é uma função das variáveis preditoras e dos parâmetros de interesse,
- ϵ é o **erro aleatório** (vetor de erros aleatórios não observáveis).

section*Modelos lineares

Objetivos

Predizer Y a partir do conhecimento de variáveis em $X = x$ utilizando uma função linear dos parâmetros β.

Em notação matricial, um modelo linear geral é dado por

$$Y = X\beta + \epsilon,$$

em que

- Y é a **variável resposta** (vetor de variáveis aleatórias observáveis),
- X contém **variáveis preditoras** (matriz conhecida, ou seja, não-aleatória),
- β é um **vetor de parâmetros de interesse**, que queremos estimar,
- ϵ é o **erro aleatório** (vetor de erros aleatórios não observáveis).

MODELO DE REGRESSÃO LINEAR SIMPLES

Suponha que são observados os pares $(X_1, Y_1), \ldots, (X_n, Y_n)$ de variável preditora e resposta, respectivamente.

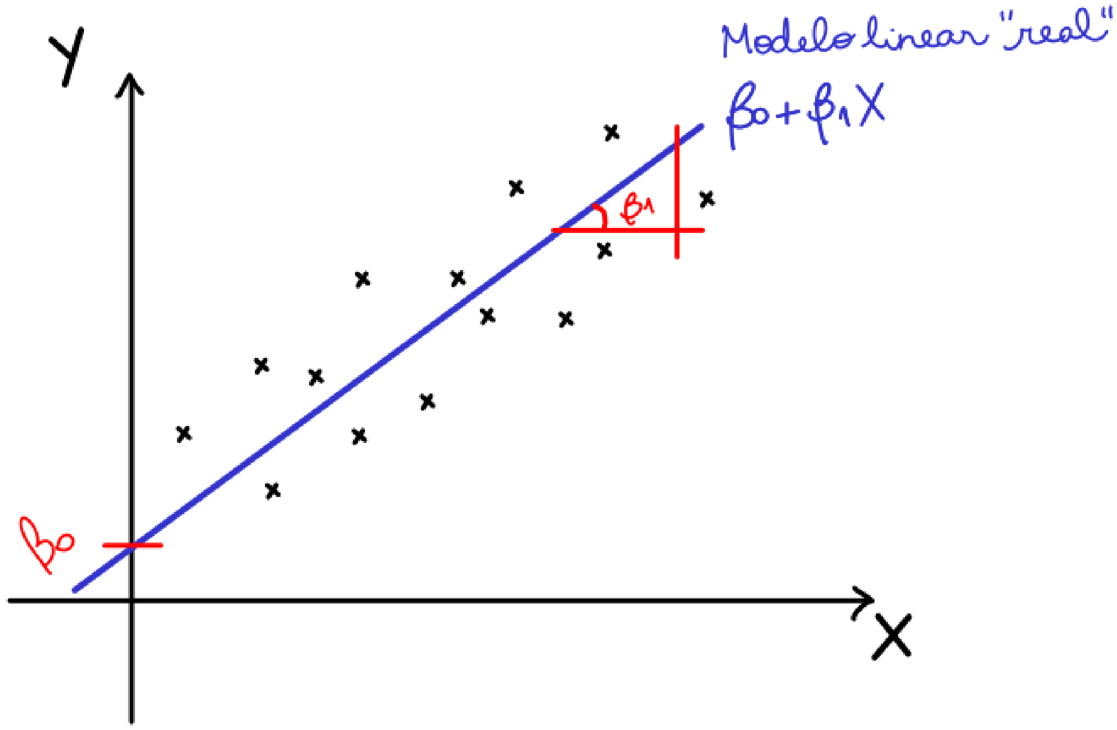
Um **modelo linear simples** para explicar a variabilidade de Y usando a variabilidade de X seria:

$$Y_j = \beta_0 + \beta_1 X_j + \epsilon_j, \text{ para } j = 1, \ldots, n.$$

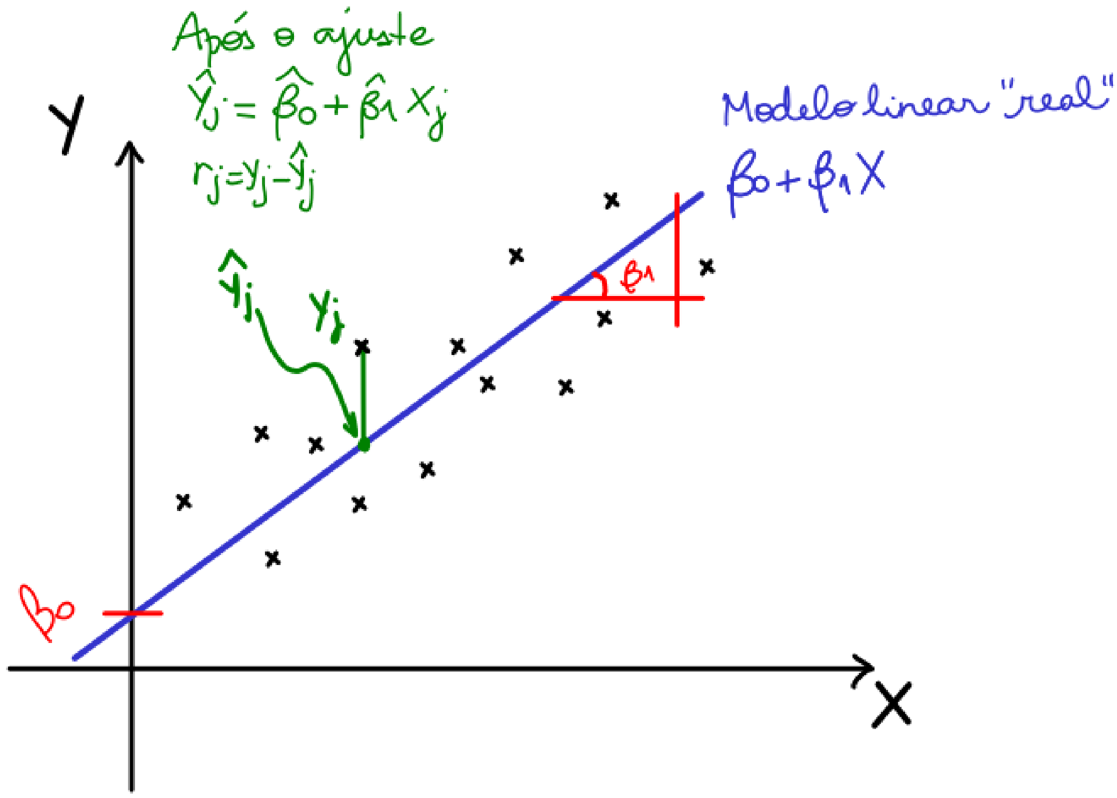
Nomenclatura: - Y_j: j-ésima observação da variável resposta (dependente, aleatória), - β_0 e β_1: parâmetros desconhecidos e que queremos estimar (fixo e desconhecido), - X_j: j-ésima observação da variável preditora (fixa, ou seja, não aleatória), - ϵ_j: j-ésimo erro aleatório não observável.

Suposições:

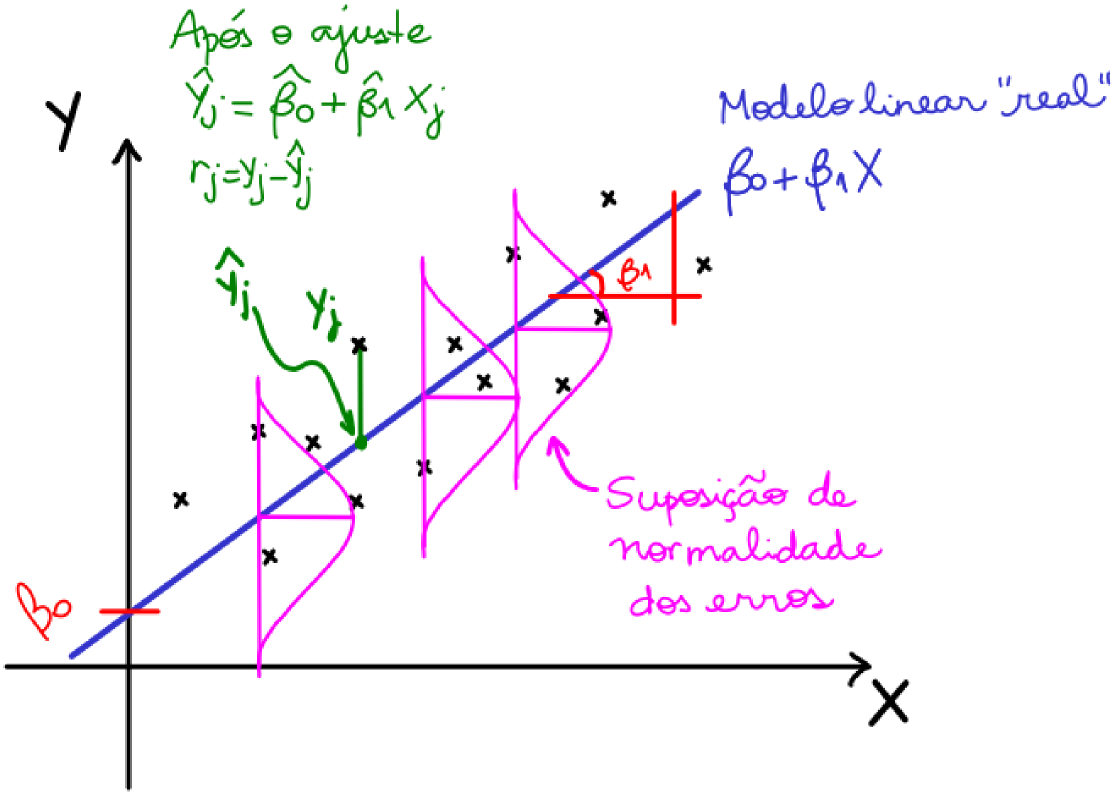
- $E(\epsilon_j) = 0$ para $j = 1, \ldots, n$,
- $Var(\epsilon_j) = \sigma^2$ para $j = 1, \ldots, n$,
- $Cov(\epsilon_i, \epsilon_j) = 0$ para $i, j = 1, \ldots, n$ e $i \neq j$.



Modelo linear "real"
$\beta_0+\beta_1 X$
β_1
β_0
Y
X



Após o ajuste
$\hat{y}_j = \hat{\beta}_0 + \hat{\beta}_1 x_j$
$r_j = y_j - \hat{y}_j$
Modelo linear "real"
$\beta_0+\beta_1 X$
$\hat{y}_j$
y_j
β_1
β_0
Y
X



INTERPRETAÇÃO DOS PARÂMETROS

- β_0: valor esperado de Y quando X é zero.
- β_1: aumento esperado em Y quando X é acrescido de uma unidade.

MODELO DE REGRESSÃO LINEAR MÚLTIPLA

Motivação: Deseja-se construir um modelo para explicar

- Y_j: valor de mercado de uma casa utilizando variáveis explicativas
- X_{1j}: área
- X_{2j}: localização
- X_{3j}: valor da casa no ano anterior
- X_{4j}: qualidade da construção

Um possível modelo linear (nos parâmetros) seria:

$$Y_j = \beta_0 + \beta_1 X_{1j} + \beta_2 X_{2j} + \beta_3 X_{3j} + \beta_4 X_{4j} + \epsilon_j.$$

Nomenclatura: - Y_j: variável resposta (dependente), - β_i: parâmetros desconhecidos, - X_{ij}: variáveis explicativas (covariáveis, variáveis independentes), - ϵ_j: erro aleatório.

Suposições:

- $E(\epsilon_j) = 0$ para $ j=1,...,n$,
- $Var(\epsilon_j) = \sigma^2$ para $ j=1,...,n$,

- $Cov(\epsilon_i, \epsilon_j) = 0$ para $ i,j=1,...,n$ e $i \neq j$.

Poderíamos estender esse modelo para p covariáveis,

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \ldots + \beta_p X_{pi} + \epsilon_i, i = 1, \ldots, n.$$

Note que a variável resposta Y_i é unidimensional.

Poderíamos "empilhar" os dados de n indivíduos em linhas. Teríamos então matricialmente

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}, X = \begin{bmatrix} 1 & X_{11} & \ldots & X_{1p} \\ 1 & X_{21} & \ldots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & \ldots & X_{np} \end{bmatrix}, \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}, \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

ou seja,

$$Y_{n\times 1} = X_{n\times(p+1)}\beta_{(p+1)\times 1} + \epsilon_{n\times 1}.$$

INTERPRETAÇÃO DOS PARÂMETROS

- β_0: valor esperado de Y quando $X_{1i}, X_{2i}, \ldots, X_{pi}$ são todas zero.
- β_k: aumento esperado em Y quando X_k é acrescido de uma unidade e todas as outras são mantidas fixadas, $k = 1, \ldots, p$.

ESTIMAÇÃO DOS PARÂMETROS

Alguns métodos podem ser usados para estimar os parâmetros, por exemplo

- Método de mínimos quadrados ordinários (EMQ ou MQO)
- Método de máxima verossimilhança (EMV)

No modelo linear geral

$$Y = X\beta + \epsilon.$$

com as suposições

- $E(\epsilon) = 0,$
- $Var(\epsilon) = \sigma^2 I,$

o **estimador de mínimos quadrados** que minimiza a soma de quadrados dos resíduos, é dado por

$$\widehat{\beta} = (X^\top X)^{-1} X^\top Y.$$

Se $\epsilon \sim N(0, \sigma^2 I)$, então o **estimador de máxima verossimilhança** de β é dado (também) por

$$\widehat{\beta} = (X^\top X)^{-1} X^\top Y.$$

Nesse caso,

$$\widehat{\beta} \sim N\left(\beta, \sigma^2 (X^\top X)^{-1}\right)$$

e é comum estimar σ^2 com

$$\widehat{\sigma}^2 = MSE.$$

Observação

O EMV de β é não-viesado e consistente, que são boas propriedades estatísticas.

VALOR AJUSTADO DE Y

O valor ajustado de Y, para um determinado $X = x$ é obtido fazendo

$$\hat{Y} = X\hat{\beta}.$$

O erro quadrático médio, MSE, é usado para estimar σ^2, fazendo

$$\hat{\sigma}^2 = \frac{SQE}{n-p} = \frac{\sum_{i=1}^{n} (Y - \hat{Y})^\top (Y - \hat{Y})}{n-p}$$

COEFICIENTE DE DETERMINAÇÃO DO MODELO

O coeficiente de determinação, ou coeficiente de explicação do modelo, é dado por

$$R^2 = 1 - \frac{SQE}{SQT},$$

em que $SQT = Y^\top Y - \frac{1}{n} Y^\top \not{1}^\top \not{1} Y$, em que $\not{1}$ indica um vetor de uns de mesma dimensão de Y.

Para levar em conta o aumento da explicação da variabilidade da resposta quando aumentamos o número de covariáveis, é comum considerar o coeficiente de determinação do modelo ajustado:

$$R^2_{ajustado} = 1 - \frac{n-1}{n-p} \frac{SQE}{SQT}.$$

Tanto R^2 quanto $R^2_{ajustado}$ estão entre 0 e 1, e pode ser usado como um **indício** de qualidade do ajuste, quanto maior o coeficiente de determinação, melhor é o modelo linear.

MODELO DE REGRESSÃO LINEAR MULTIVARIADA

Considere agora que, para cada indivíduo, sejam observadas m variáveis respostas, e que cada uma delas tenha uma relação linear com as p covariáveis.

Assim, teriamos m modelos de regressão:

$$\begin{gathered}
Y_1 = \beta_{01} + \beta_{11}X_1 + \beta_{21}X_2 + \beta_{31}X_3 + \ldots + \beta_{p1}X_p + \epsilon_1 \\
Y_2 = \beta_{02} + \beta_{12}X_1 + \beta_{22}X_2 + \beta_{32}X_3 + \ldots + \beta_{p2}X_p + \epsilon_2 \\
\vdots \\
Y_m = \beta_{0m} + \beta_{1m}X_1 + \beta_{2m}X_2 + \beta_{3m}X_3 + \ldots + \beta_{pm}X_p + \epsilon_m
\end{gathered}$$

Para cada um dos n indivíduos, vamos observar as m variáveis resposta e as p covariáveis. Assim, podemos definir um modelo de regressão multivariado

$$Y_{n\times m} = X_{n\times(p+1)}\beta_{(p+1)\times m} + \epsilon_{n\times m}$$

em que

$$Y = \begin{bmatrix} Y_{11} & Y_{12} & \ldots & Y_{1m} \\ Y_{21} & Y_{22} & \ldots & Y_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{n1} & Y_{n2} & \ldots & Y_{nm} \end{bmatrix}, \; X = \begin{bmatrix} 1 & X_{11} & \ldots & X_{1p} \\ 1 & X_{21} & \ldots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & \ldots & X_{np} \end{bmatrix},$$

$$\beta = \begin{bmatrix} \beta_{01} & \beta_{02} & \ldots & \beta_{0m} \\ \beta_{11} & \beta_{12} & \ldots & \beta_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{p1} & \beta_{p2} & \ldots & \beta_{pm} \end{bmatrix}, \; \epsilon = \begin{bmatrix} \epsilon_{11} & \epsilon_{12} & \ldots & \epsilon_{1m} \\ \epsilon_{21} & \epsilon_{22} & \ldots & \epsilon_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon_{n1} & \epsilon_{n2} & \ldots & \epsilon_{nm} \end{bmatrix}$$

MATERIAL COMPLEMENTAR

Para as suposições e demais desenvolvimentos no modelo de regressão linear multivariado, veja a aula de Regressão multivariada em https://youtu.be/9QIh71MQ2xQ

ANÁLISE DE DIAGNÓSTICO

Os resíduos contém indicativos de adequabilidade das suposições do modelo

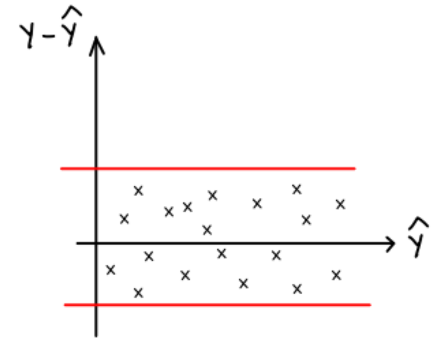
Os **resíduos ordinários** do modelo são dados por

$$e = Y - \widehat{Y}.$$

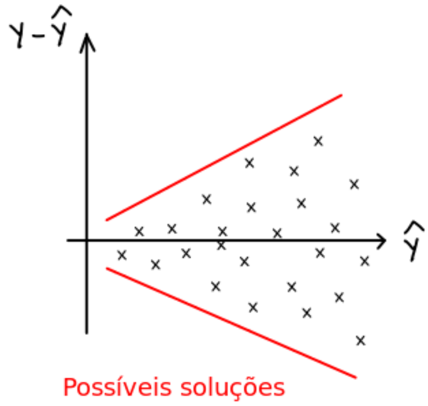
É comum construir gráficos dos resíduos ordinários contra a ordem das observações, os valores ajustados $\widehat{Y}$ e X_i, para algumas das variáveis preditoras de interesse.

Espera-se que os resíduos sejam aleatoriamente distribuídos em torno de zero.

Alguns padrões de resíduos são ilustrados a seguir:



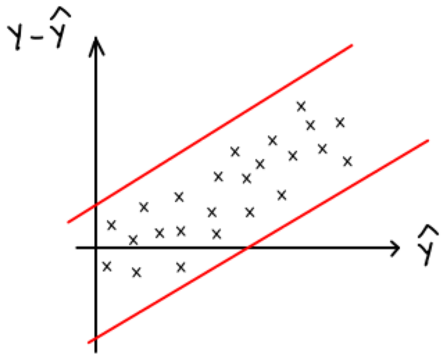
Ok!



Possíveis soluções

- mínimos quadrados ponderados

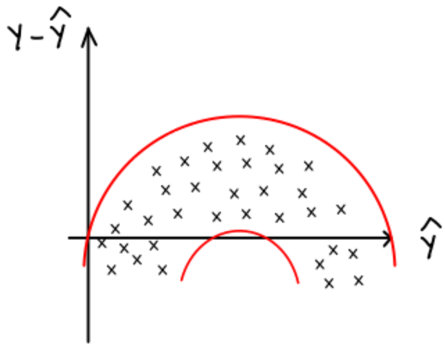
- transformação em Y



Rever modelo!

Se o padrão for observado no gráfico contra o tempo, pode faltar termo linear no tempo.

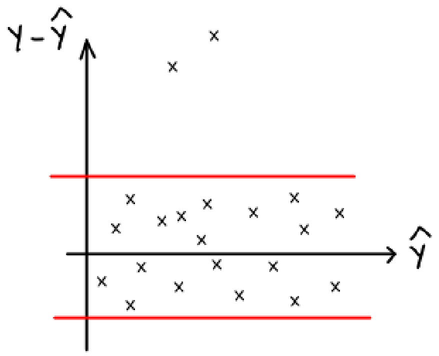
Se o padrão for observado no gráfico contra X, rever preditores.

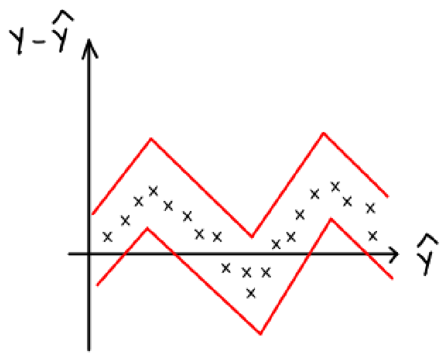


Possíveis soluções:

- Adicionar termos extras ou transformar Y

Se o padrão for observado no gráfico contra o tempo, testar a inclusão de termos lineares ou quadráticos no tempo





Existem algumas propostas para a padronização de resíduos:

Temos que

$$e = Y - \hat{Y} = Y - X(X^\top X)^{-1}X^\top Y = (I - X(X^\top X)^{-1}X^\top)Y = (I - H)Y,$$

em que $H = X(X^\top X)^{-1}X^\top$ é a matriz hat (matriz chapéu). Seja h_{ii} o i-ésimo elemento da diagonal de H.

Pode-se mostrar que

$$Var(e_i) = (1 - h_{ii})\sigma^2$$

Assim, podemos definir dois novos resíduos:

- **Resíduo Studentizado internamente:**

$$s_i = \frac{e_i}{s\sqrt{1 - h_{ii}}},$$

em que s é uma estimativa de σ, usualmente a raiz do MSE.

- **Resíduo Studentizado externamente**

$$t_i = \frac{e_i}{s(i)\sqrt{1 - h_{ii}}},$$

em que $s(i)$ é uma estimativa para σ^2 sem a observação i.

Observação: Com h_{ii} é possível identificar **pontos de alavanca**, que são outliers no espaço dos X e não necessariamente são ponto influentes, ou seja, não necessariamente mudam a inferência do modelo.

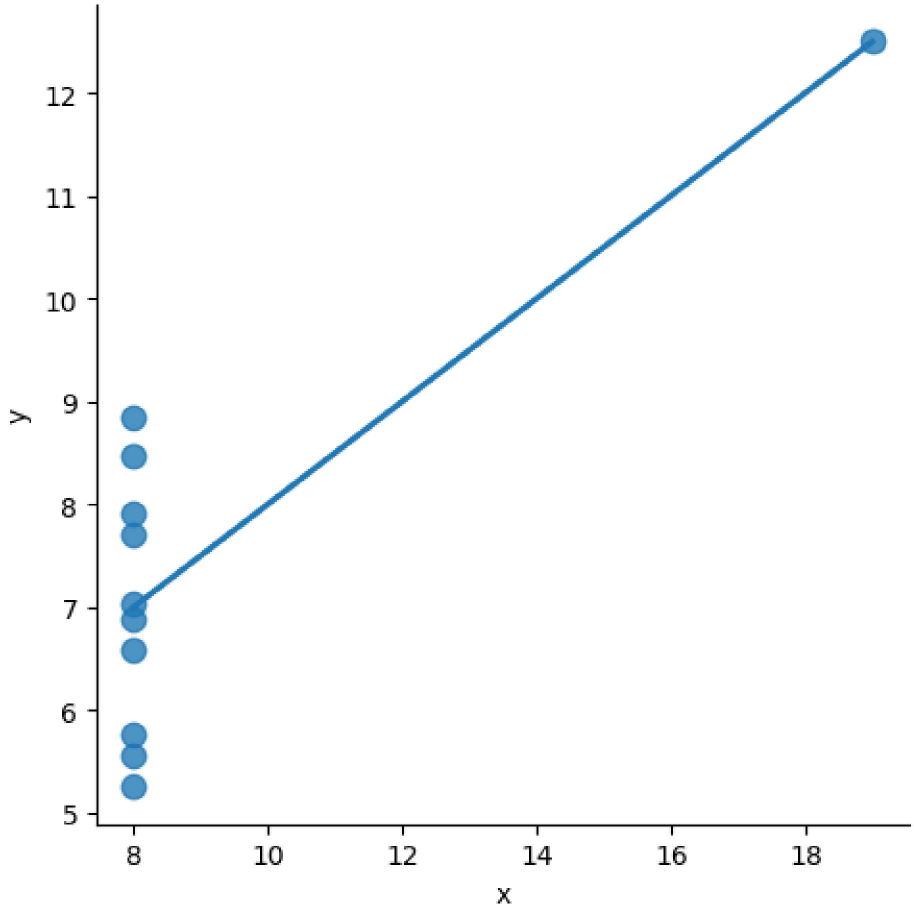
Existem técnicas para identificar **pontos influentes**, como DFFITS e Distância de Cook, que veremos na prática.

ILUSTRAÇÃO DE RESÍDUOS E PONTOS DE ALAVANCA

Conjuntos de dados de Anscombe: https://pt.wikipedia.org/wiki/Quarteto_de_Anscombe

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

[2]:
```
anscombe = sns.load_dataset("anscombe")

sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'IV'"),
           robust=True, ci=None, scatter_kws={"s": 80});
```



[3]:
```
# Gráficos de diagnóstico para o modelo linear simples com estimadores MQO

# Fonte: https://stackoverflow.com/questions/46607831/
↪python-linear-regression-diagnostic-plots-similar-to-r
# Fonte: https://stackoverflow.com/questions/46304514/
↪access-standardized-residuals-cooks-values-hatvalues-leverage-etc-easily-i/
↪55764402#55764402
```



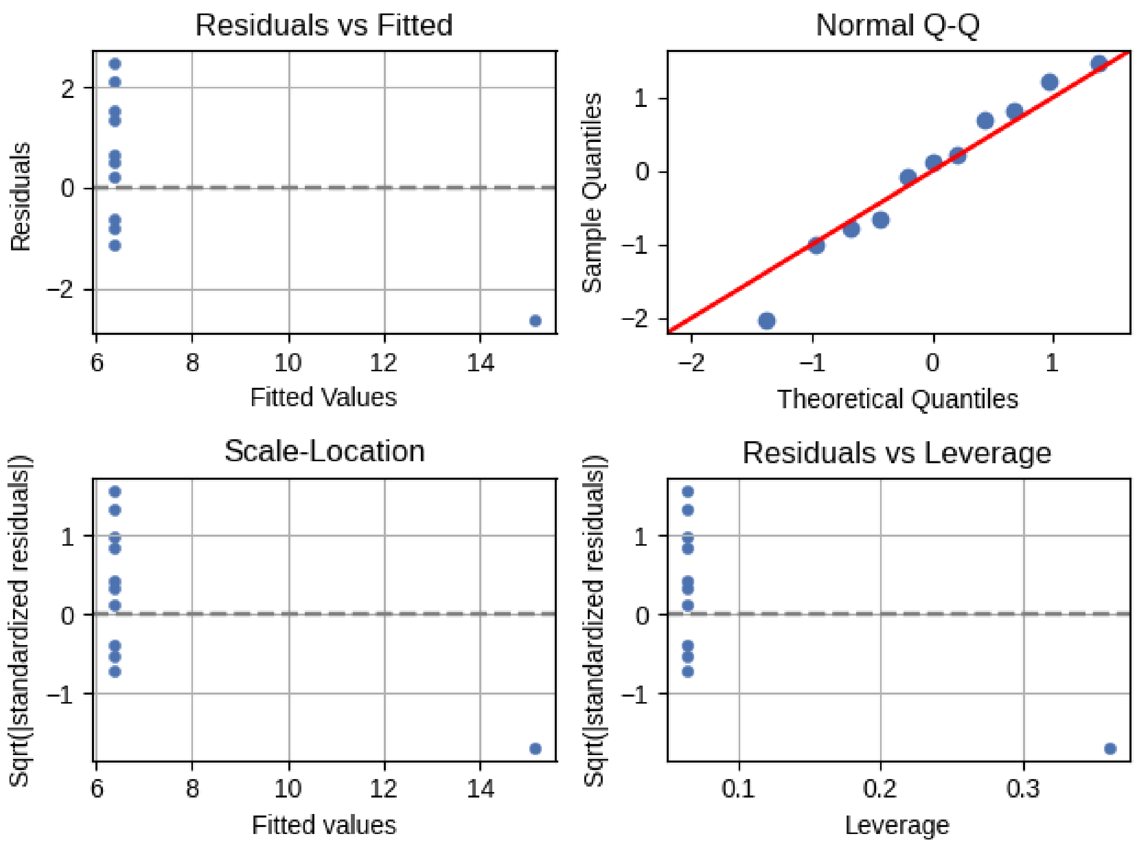
```python
from matplotlib import pyplot as plt
from pandas.core.frame import DataFrame
import scipy.stats as stats
import statsmodels.api as sm


def linear_regression(df: DataFrame) -> DataFrame:
    """Perform a univariate regression and store results in a new data frame.

    Args:
        df (DataFrame): orginal data set with x and y.

    Returns:
        DataFrame: another dataframe with raw data and results.
    """
    mod = sm.OLS(endog=df['y'], exog=df['x']).fit()
    influence = mod.get_influence()

    res = df.copy()
    res['resid'] = mod.resid
    res['fittedvalues'] = mod.fittedvalues
    res['resid_std'] = mod.resid_pearson
    res['leverage'] = influence.hat_matrix_diag
    return res


def plot_diagnosis(df: DataFrame):
    fig, axes = plt.subplots(nrows=2, ncols=2)
    plt.style.use('seaborn')

    # Residual against fitted values.
    df.plot.scatter(
        x='fittedvalues', y='resid', ax=axes[0, 0]
    )
    axes[0, 0].axhline(y=0, color='grey', linestyle='dashed')
    axes[0, 0].set_xlabel('Fitted Values')
    axes[0, 0].set_ylabel('Residuals')
    axes[0, 0].set_title('Residuals vs Fitted')

    # qqplot
```

```python
    sm.qqplot(
        df['resid'], dist=stats.t, fit=True, line='45',
        ax=axes[0, 1], c='#4C72B0'
    )
    axes[0, 1].set_title('Normal Q-Q')

    # The scale-location plot.
    df.plot.scatter(
        x='fittedvalues', y='resid_std', ax=axes[1, 0]
    )
    axes[1, 0].axhline(y=0, color='grey', linestyle='dashed')
    axes[1, 0].set_xlabel('Fitted values')
    axes[1, 0].set_ylabel('Sqrt(|standardized residuals|)')
    axes[1, 0].set_title('Scale-Location')

    # Standardized residuals vs. leverage
    df.plot.scatter(
        x='leverage', y='resid_std', ax=axes[1, 1]
    )
    axes[1, 1].axhline(y=0, color='grey', linestyle='dashed')
    axes[1, 1].set_xlabel('Leverage')
    axes[1, 1].set_ylabel('Sqrt(|standardized residuals|)')
    axes[1, 1].set_title('Residuals vs Leverage')

    plt.tight_layout()
    plt.show()
```

[4]:
```python
df = data=anscombe.query("dataset == 'IV'")
df = df.drop('dataset', axis=1)
```

[5]:
```python
df = linear_regression(df)
```

[6]:
```python
plot_diagnosis(df)
```



section*Seleção de modelos

Algumas formas de avaliar e selecionar modelos, além da análise de diagnóstico, são

- Validação com bases de treinamento e teste
- Avaliação de métricas de qualidade do ajuste
- Seleção de variáveis
- Validação cruzada (K-fold Cross-Validation): ver https://scikit-learn.org/stable/modules/cross_validation.html

MÉTRICAS PARA AVALIAR A QUALIDADE DO AJUSTE

ERRO ABSOLUTO MÉDIO (EAM, MAE)

O erro quadrático médio (EQM) ou mean absolute error (MAE) é a média do valor absoluto dos erros.

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \widehat{y}_j|$$

ERRO QUADRÁTICO MÉDIO (EQM, MSE)

O erro quadrático médio (EQM) ou mean squared error (MSE) é a média dos erros quadráticos.

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{n}(y_j - \widehat{y}_j)^2$$

RAIZ DO ERRO QUADRÁTICO MÉDIO (REQM, RMSE)

A raiz do erro quadrático médio (REQM) ou root mean squared error (RMSE) é a raiz da média dos erros quadráticos.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \widehat{y}_j)^2}$$

APLICAÇÃO

Suponha que desejamos predizer o valor de venda de uma casa utilizando variáveis preditoras como número de quartos, número de banheiros, tamanho da sala, número de andares, entre outras.

Fonte e alguns desenvolvimentos adicionais: ver https://www.kaggle.com/burhanykiyakoglu/predicting-house-prices

[7]:
```
!pip install folium
```

[8]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import PolynomialFeatures
from sklearn import metrics
from sklearn.model_selection import cross_val_score



import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
import folium
from folium.plugins import HeatMap
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

evaluation = pd.DataFrame({'Model': [],
                           'Details':[],
                           'Root Mean Squared Error (RMSE)':[],
                           'R-squared (training)':[],
```

```python
                                    'Adjusted R-squared (training)':[],
                                    'R-squared (test)':[],
                                    'Adjusted R-squared (test)':[],
                                    '5-Fold Cross Validation':[]})

# Faça a leitura dos dados localmente
#df = pd.read_csv('/hdd/MBA/ECD/Data/kc_house_data.csv')

# Ou faça a leitura direto do github
df = pd.read_csv('https://raw.githubusercontent.com/cibelerusso/
 ↪Estatistica-Ciencia-Dados/main/Data/kc_house_data.csv')
df.head()
```

[8]:

```
           id             date     price  bedrooms  bathrooms  sqft_living  \
0  7129300520  20141013T000000  221900.0         3       1.00         1180
1  6414100192  20141209T000000  538000.0         3       2.25         2570
2  5631500400  20150225T000000  180000.0         2       1.00          770
3  2487200875  20141209T000000  604000.0         4       3.00         1960
4  1954400510  20150218T000000  510000.0         3       2.00         1680

   sqft_lot  floors  waterfront  view  …  grade  sqft_above  sqft_basement  \
0      5650     1.0           0     0  …      7      1180.0              0
1      7242     2.0           0     0  …      7      2170.0            400
2     10000     1.0           0     0  …      6       770.0              0
3      5000     1.0           0     0  …      7      1050.0            910
4      8080     1.0           0     0  …      8      1680.0              0

   yr_built  yr_renovated  zipcode      lat     long  sqft_living15  \
0      1955             0    98178  47.5112 -122.257           1340
1      1951          1991    98125  47.7210 -122.319           1690
2      1933             0    98028  47.7379 -122.233           2720
3      1965             0    98136  47.5208 -122.393           1360
4      1987             0    98074  47.6168 -122.045           1800

   sqft_lot15
0        5650
1        7639
2        8062
3        5000
4        7503
```

```
[5 rows x 21 columns]
```

[9]:
```python
df.describe()
```

[9]:

	id	price	bedrooms	bathrooms	sqft_living
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000

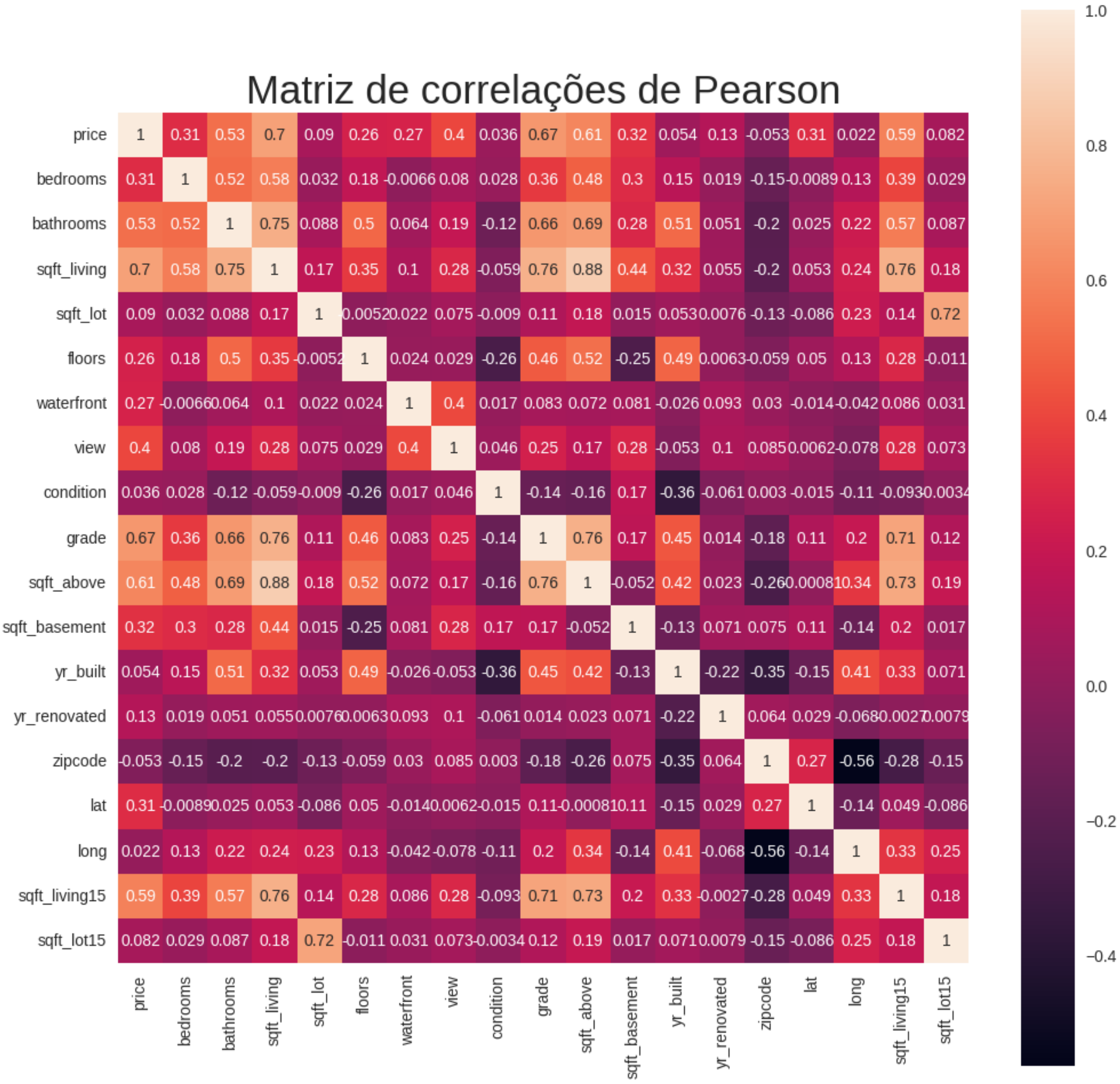
	sqft_lot	floors	waterfront	view	condition
count	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000
mean	1.510697e+04	1.494309	0.007542	0.234303	3.409430
std	4.142051e+04	0.539989	0.086517	0.766318	0.650743
min	5.200000e+02	1.000000	0.000000	0.000000	1.000000
25%	5.040000e+03	1.000000	0.000000	0.000000	3.000000
50%	7.618000e+03	1.500000	0.000000	0.000000	3.000000
75%	1.068800e+04	2.000000	0.000000	0.000000	4.000000
max	1.651359e+06	3.500000	1.000000	4.000000	5.000000

	grade	sqft_above	sqft_basement	yr_built	yr_renovated
count	21613.000000	21611.000000	21613.000000	21613.000000	21613.000000
mean	7.656873	1788.396095	291.509045	1971.005136	84.402258
std	1.175459	828.128162	442.575043	29.373411	401.679240
min	1.000000	290.000000	0.000000	1900.000000	0.000000
25%	7.000000	1190.000000	0.000000	1951.000000	0.000000
50%	7.000000	1560.000000	0.000000	1975.000000	0.000000
75%	8.000000	2210.000000	560.000000	1997.000000	0.000000
max	13.000000	9410.000000	4820.000000	2015.000000	2015.000000

	zipcode	lat	long	sqft_living15	sqft_lot15
count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000
mean	98077.939805	47.560053	-122.213896	1986.552492	12768.455652
std	53.505026	0.138564	0.140828	685.391304	27304.179631
min	98001.000000	47.155900	-122.519000	399.000000	651.000000
25%	98033.000000	47.471000	-122.328000	1490.000000	5100.000000
50%	98065.000000	47.571800	-122.230000	1840.000000	7620.000000

75%	98118.000000	47.678000	-122.125000	2360.000000	10083.000000
max	98199.000000	47.777600	-121.315000	6210.000000	871200.000000

[10]:
```python
fig, ax = plt.subplots(figsize=(12,12))
sns.heatmap(df.drop(['id', 'date'], axis=1).corr(),annot=True, square=True)

plt.title('Matriz de correlações de Pearson',fontsize=25);
```



MODELO LINEAR SIMPLES

[11]:
```python
from statsmodels.formula.api import ols

#Ajusta o modelo de regressão linear simples para o preço das casas
mod = ols('price~sqft_living',data=df)
```

```
res = mod.fit()
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.493
Model:                            OLS   Adj. R-squared:                  0.493
Method:                 Least Squares   F-statistic:                 2.100e+04
Date:                Thu, 16 May 2024   Prob (F-statistic):               0.00
Time:                        01:32:52   Log-Likelihood:            -3.0027e+05
No. Observations:               21613   AIC:                         6.005e+05
Df Residuals:                   21611   BIC:                         6.006e+05
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept   -4.358e+04   4402.690     -9.899      0.000   -5.22e+04    -3.5e+04
sqft_living   280.6236      1.936    144.920      0.000     276.828     284.419
==============================================================================
Omnibus:                    14832.490   Durbin-Watson:                   1.983
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           546444.713
Skew:                           2.824   Prob(JB):                         0.00
Kurtosis:                      26.977   Cond. No.                     5.63e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.63e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Modelo ajustado

$\widehat{Y}_i = -43580 + 280.62$ sqft_living

MODELO LINEAR MÚLTIPLO

[12]:
```
X =␣
↪df[['bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','view','condition
↪values
```



```python
y = df['price'].values
```

[13]:
```python
from statsmodels.formula.api import ols

#Ajusta o modelo de regressão linear múltipla para o preço das casas
modelo = ols('price ~ bedrooms + bathrooms + sqft_living + sqft_lot + floors +␣
↪waterfront + view + condition + grade',data=df)

res = modelo.fit()
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.605
Model:                            OLS   Adj. R-squared:                  0.605
Method:                 Least Squares   F-statistic:                     3673.
Date:                Thu, 16 May 2024   Prob (F-statistic):               0.00
Time:                        01:32:53   Log-Likelihood:            -2.9757e+05
No. Observations:               21613   AIC:                         5.952e+05
Df Residuals:                   21603   BIC:                         5.952e+05
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   -6.827e+05   1.73e+04    -39.509      0.000   -7.17e+05   -6.49e+05
bedrooms    -3.367e+04   2159.240    -15.594      0.000   -3.79e+04   -2.94e+04
bathrooms   -1.142e+04   3449.874     -3.309      0.001   -1.82e+04   -4653.384
sqft_living   196.3657      3.454     56.855      0.000     189.596     203.135
sqft_lot       -0.3462      0.039     -8.931      0.000      -0.422      -0.270
floors      -1.312e+04   3575.901     -3.669      0.000   -2.01e+04   -6109.502
waterfront   5.783e+05   1.99e+04     29.128      0.000    5.39e+05    6.17e+05
view         6.327e+04   2348.558     26.939      0.000    5.87e+04    6.79e+04
condition    5.499e+04   2521.782     21.805      0.000       5e+04    5.99e+04
grade        1.006e+05   2231.983     45.064      0.000    9.62e+04    1.05e+05
==============================================================================
Omnibus:                    15533.601   Durbin-Watson:                   1.983
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           900296.321
Skew:                           2.871   Prob(JB):                         0.00
Kurtosis:                      34.093   Cond. No.                     5.59e+05
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.59e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

MÉTRICAS

[14]:
```python
# R2 ajustado de
# https://www.kaggle.com/burhanykiyakoglu/predicting-house-prices

def adjustedR2(r2,n,k):
    return r2-(k-1)/(n-k)*(1-r2)
```

[15]:
```python
# Fonte: https://www.kaggle.com/burhanykiyakoglu/predicting-house-prices

train_data,test_data = train_test_split(df,train_size = 0.8,random_state=3)

features =␣
 ↪['bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront', 'view',␣
 ↪'condition', 'grade']
complex_model_1 = linear_model.LinearRegression()
complex_model_1.fit(train_data[features],train_data['price'])

print('Intercept: {}'.format(complex_model_1.intercept_))
print('Coefficients: {}'.format(complex_model_1.coef_))

pred = complex_model_1.predict(test_data[features])
rmsecm = float(format(np.sqrt(metrics.
 ↪mean_squared_error(test_data['price'],pred)),'.3f'))
rtrcm = float(format(complex_model_1.
 ↪score(train_data[features],train_data['price']),'.3f'))

artrcm = float(format(adjustedR2(complex_model_1.
 ↪score(train_data[features],train_data['price']),train_data.
 ↪shape[0],len(features)),'.3f'))
rtecm = float(format(complex_model_1.
 ↪score(test_data[features],test_data['price']),'.3f'))
```

```
artecm = float(format(adjustedR2(complex_model_1.
 ↪score(test_data[features],test_data['price']),test_data.
 ↪shape[0],len(features)),'.3f'))
cv = float(format(cross_val_score(complex_model_1,df[features],df['price'],cv=5).
 ↪mean(),'.3f'))

r = evaluation.shape[0]
evaluation.loc[r] = ['Multiple Regression-1','selected␣
 ↪features',rmsecm,rtrcm,artrcm,rtecm,artecm,cv]
evaluation.sort_values(by = '5-Fold Cross Validation', ascending=False)
```

```
Intercept: -682602.5354521909
Coefficients: [-3.37785908e+04 -1.09025075e+04  1.99448654e+02 -3.50854472e-01
 -1.48240739e+04  5.45051297e+05  6.50285793e+04  5.58998443e+04
  9.95878298e+04]
```

[15]:

```
                  Model           Details  Root Mean Squared Error (RMSE)  \
0  Multiple Regression-1  selected features                      221680.867

   R-squared (training)  Adjusted R-squared (training)  R-squared (test)  \
0                 0.602                          0.602             0.617

   Adjusted R-squared (test)  5-Fold Cross Validation
0                      0.616                    0.601
```

SELEÇÃO DE VARIÁVEIS (FEATURE SELECTION)

[16]:

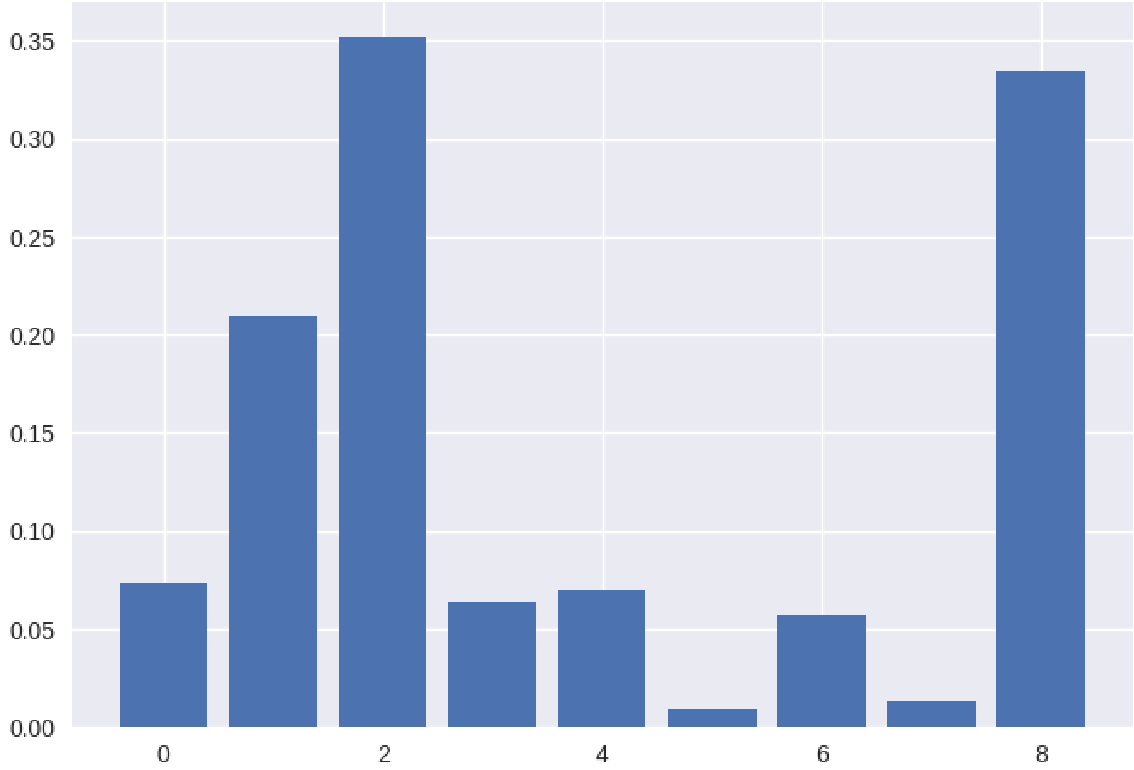
```
X =␣
 ↪df[['bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','view','condition

y = df['price']


X_treino, X_teste, y_treino, y_teste = train_test_split(X, y, test_size=0.2,␣
 ↪random_state = 1)
```

[17]:

```
# https://www.kaggle.com/burhanykiyakoglu/predicting-house-prices

# load and summarize the dataset
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
```

```python
from sklearn.feature_selection import mutual_info_regression
from matplotlib import pyplot


# feature selection
def select_features(X_treino, y_treino, X_teste):
        # configure to select all features
        fs = SelectKBest(score_func=mutual_info_regression, k='all')
        # learn relationship from treinoing data
        fs.fit(X_treino, y_treino)
        # transform treino input data
        X_treino_fs = fs.transform(X_treino)
        # transform teste input data
        X_teste_fs = fs.transform(X_teste)
        return X_treino_fs, X_teste_fs, fs

# split into treino and teste sets
X_treino, X_teste, y_treino, y_teste = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=1)

# feature selection
X_treino_fs, X_teste_fs, fs = select_features(X_treino, y_treino, X_teste)

# what are scores for the features
for i in range(len(fs.scores_)):
        print('Feature %d: %f' % (i, fs.scores_[i]))

# plot the scores
pyplot.bar([i for i in range(len(fs.scores_))], fs.scores_)
pyplot.show()
```

```
Feature 0: 0.073221
Feature 1: 0.209489
Feature 2: 0.352033
Feature 3: 0.063594
Feature 4: 0.070467
Feature 5: 0.008940
Feature 6: 0.056564
Feature 7: 0.013337
Feature 8: 0.334368
```



[18]:
```
# Fonte: https://www.kaggle.com/burhanykiyakoglu/predicting-house-prices

train_data,test_data = train_test_split(df,train_size = 0.8,random_state=3)

features = ['sqft_living','grade']
complex_model_1 = linear_model.LinearRegression()
complex_model_1.fit(train_data[features],train_data['price'])

print('Intercept: {}'.format(complex_model_1.intercept_))
print('Coefficients: {}'.format(complex_model_1.coef_))

pred = complex_model_1.predict(test_data[features])
rmsecm = float(format(np.sqrt(metrics.
 ↪mean_squared_error(test_data['price'],pred)),'.3f'))
rtrcm = float(format(complex_model_1.
 ↪score(train_data[features],train_data['price']),'.3f'))

artrcm = float(format(adjustedR2(complex_model_1.
 ↪score(train_data[features],train_data['price']),train_data.
 ↪shape[0],len(features)),'.3f'))
```

```
rtecm = float(format(complex_model_1.
↪score(test_data[features],test_data['price']),'.3f'))
artecm = float(format(adjustedR2(complex_model_1.
↪score(test_data[features],test_data['price']),test_data.
↪shape[0],len(features)),'.3f'))
cv = float(format(cross_val_score(complex_model_1,df[features],df['price'],cv=5).
↪mean(),'.3f'))

r = evaluation.shape[0]
evaluation.loc[r] = ['Multiple Regression-3','selected␣
↪features',rmsecm,rtrcm,artrcm,rtecm,artecm,cv]
evaluation.sort_values(by = '5-Fold Cross Validation', ascending=False)
```

```
Intercept: -598022.8027257539
Coefficients: [   186.877963  97878.7708715]
```

[18]:

```
                   Model            Details  Root Mean Squared Error (RMSE)  \
0  Multiple Regression-1  selected features                      221680.867
1  Multiple Regression-3  selected features                      242366.385

   R-squared (training)  Adjusted R-squared (training)  R-squared (test)  \
0                 0.602                          0.602             0.617
1                 0.533                          0.533             0.542

   Adjusted R-squared (test)  5-Fold Cross Validation
0                      0.616                    0.601
1                      0.542                    0.533
```

ANÁLISE DE RESÍDUOS

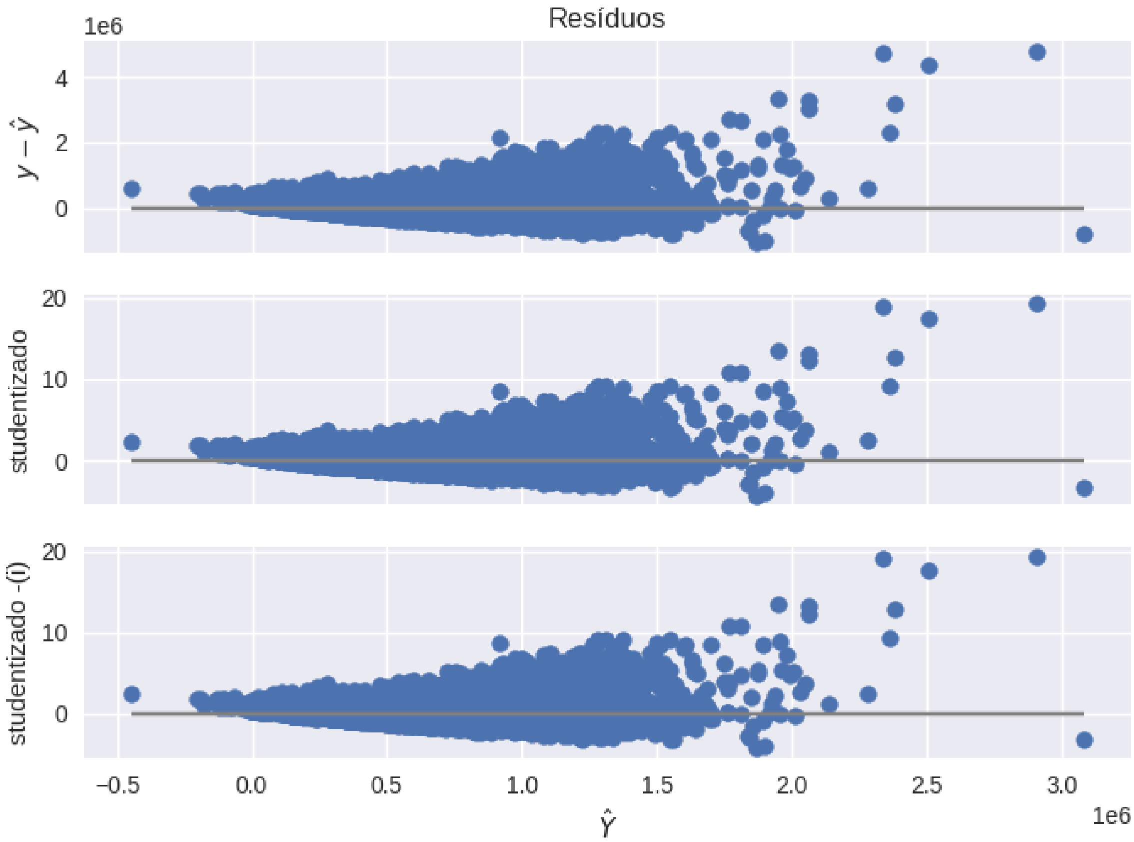
[19]:

```
# Ajusta o modelo de regressão linear múltipla para o preço das casas com duas␣
↪preditoras
modelo = ols('price ~ sqft_living + grade',data=df)
res = modelo.fit()

# valores ajustados de E(Y)
ypred=res.fittedvalues

# resíduo=observado-ajustado
residuo = res.resid
```

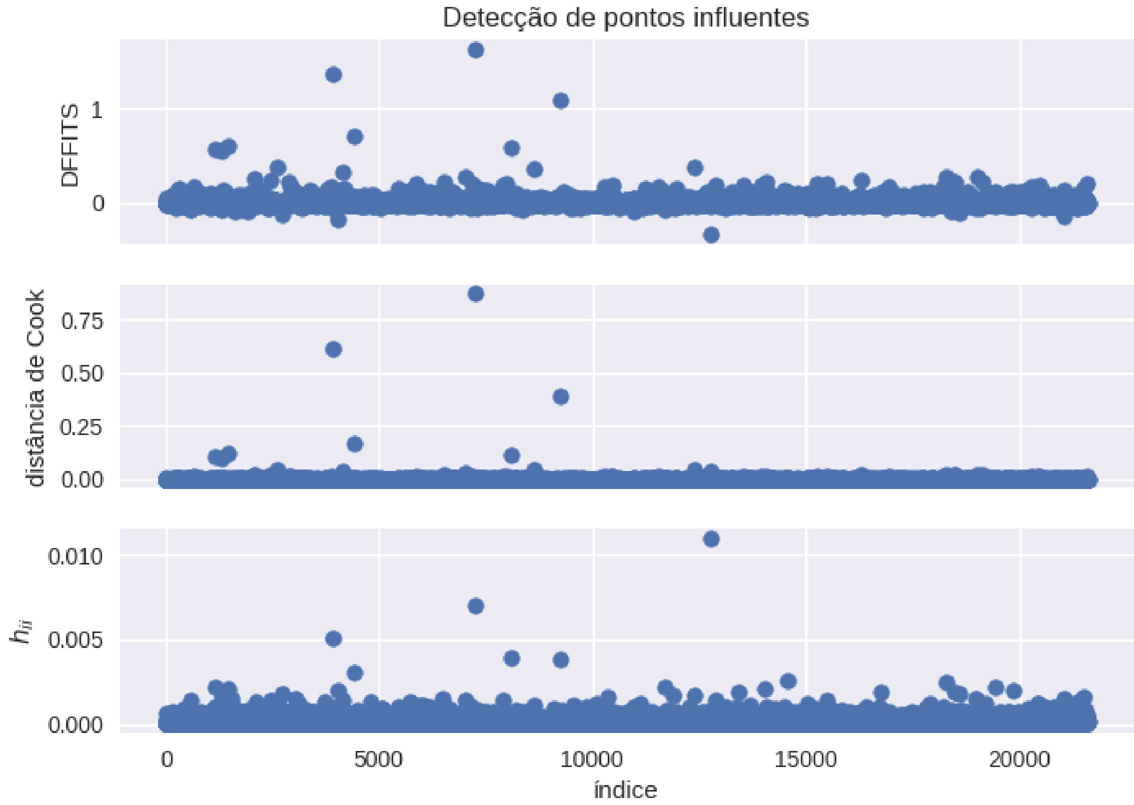
```python
# objeto para a análise de pontos influentes
infl = res.get_influence()

# diagonal da matriz hat
hii = infl.hat_matrix_diag

# resíduo studentizado (internamente)
res_stud = infl.resid_studentized_internal

# resíduo studentizado com i-ésima observação deletada (externamente)
res_stud_del = infl.resid_studentized_external

# DFFITS
(dffits,p) = infl.dffits

# Distância de Cook
(cook,p) = infl.cooks_distance
```

[20]:

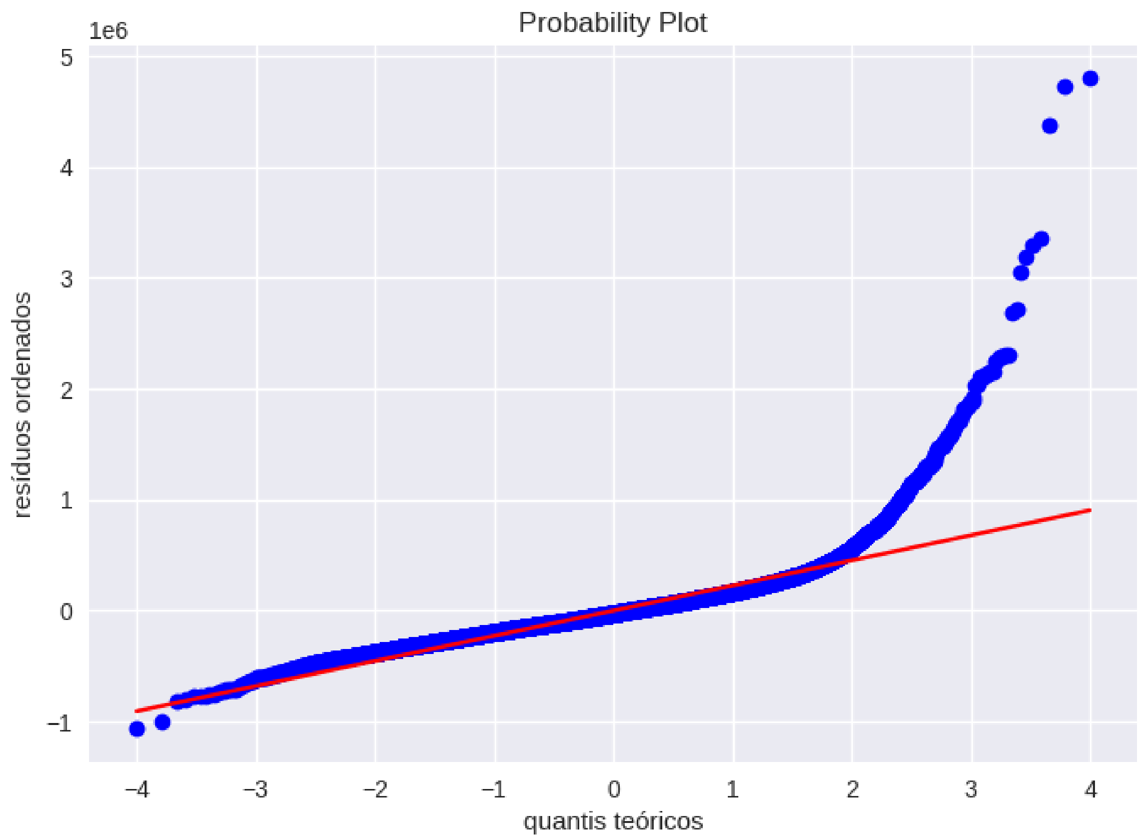
```python
fig, (ax1, ax2, ax3) = plt.subplots(3)
ax1.scatter(ypred, residuo)
ax1.set_ylabel('$y-\hat{y}$')
ax1.set_title('Resíduos')
ax1.hlines(0,xmin=min(ypred),xmax=max(ypred),color='gray')
ax2.scatter(ypred, res_stud)
ax2.set_ylabel('studentizado')
ax2.hlines(0,xmin=min(ypred),xmax=max(ypred),color='gray')
ax3.scatter(ypred, res_stud_del)
ax3.set_ylabel('studentizado -(i)')
ax3.hlines(0,xmin=min(ypred),xmax=max(ypred),color='gray')
ax3.set_xlabel('$\hat{Y}$')

for ax in fig.get_axes():
    ax.label_outer()
```



[21]:
```
fig, (ax1, ax2, ax3) = plt.subplots(3)
ax1.scatter(df.index, dffits)
ax1.set_ylabel('DFFITS')
ax1.set_title('Detecção de pontos influentes')
#ax1.hlines(0,xmin=1,xmax=102,color='gray')

ax2.scatter(df.index, cook)
ax2.set_ylabel('distância de Cook')

ax3.scatter(df.index, hii)
ax3.set_ylabel('$h_{ii}$')
ax3.set_xlabel('índice')

for ax in fig.get_axes():
    ax.label_outer()
```



[22]:
```
# Verificando a suposição de distribuição Normal dos resíduos

stats.probplot(residuo, plot=plt)
plt.xlabel('quantis teóricos')
plt.ylabel('resíduos ordenados')
plt.show()
```



TRANSFORMAÇÃO EM Y

Transformação de Box Cox: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.boxcox.html

[23]:
```
stats.boxcox(df['price'])
```

[23]:
```
(array([4.0334758 , 4.07834265, 4.02144527, …, 4.06460556, 4.06434971,
        4.05395248]),
 -0.23401853749997595)
```

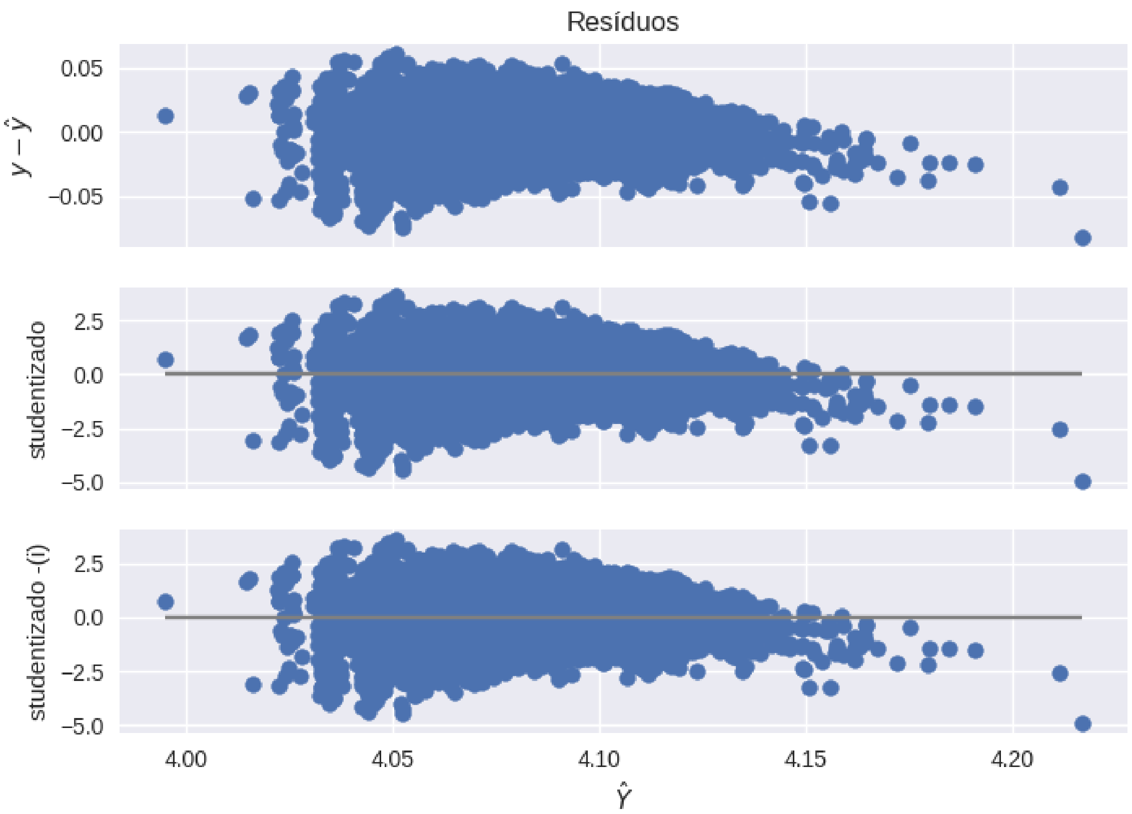
[24]:
```
df['price_transformado'] = (pow(df['price'],-0.234) - 1)/(-0.234)
```

[25]:
```
# Ajusta o modelo de regressão linear múltipla para o preço das casas com duas␣
 ↪preditoras
modelo = ols('price_transformado ~ sqft_living + grade',data=df)
res = modelo.fit()

# valores preditos de E(Y)
ypred=res.fittedvalues

# resíduo=observado-ajustado
```

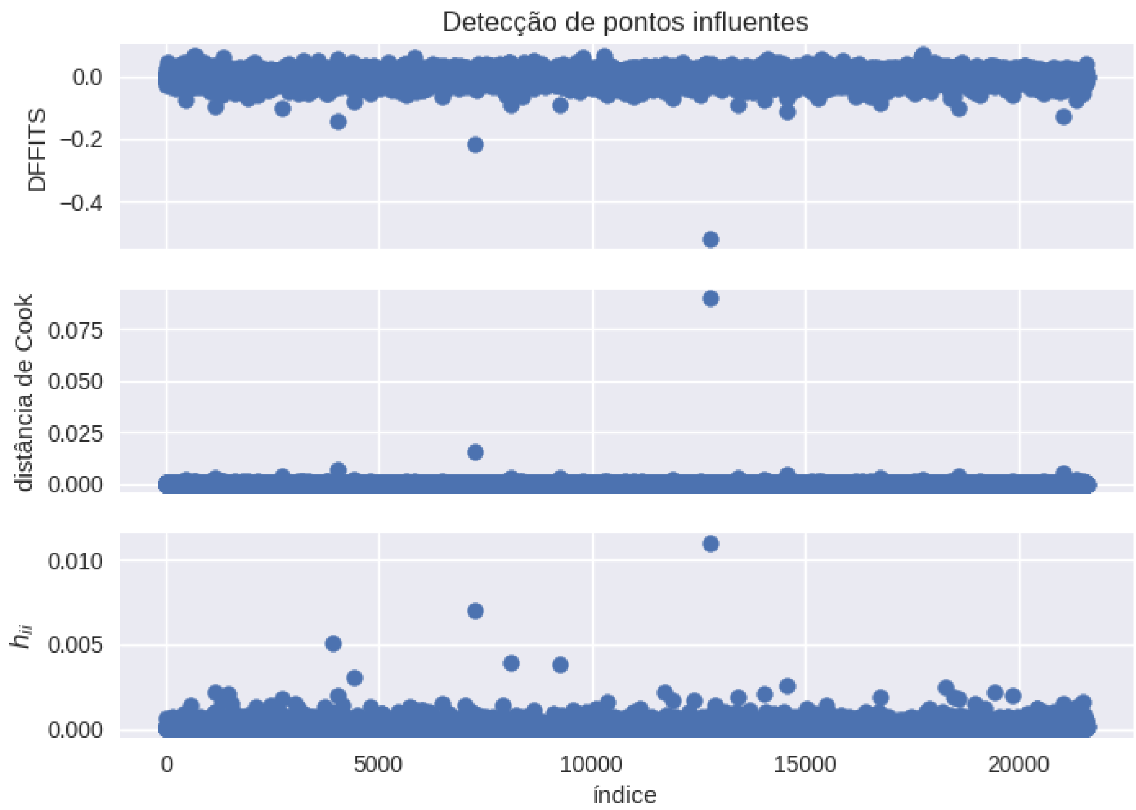
```python
residuo = res.resid

# objeto para a análise de pontos influentes
infl = res.get_influence()

# diagonal da matriz hat
hii = infl.hat_matrix_diag

# resíduo studentizado (internamente)
res_stud = infl.resid_studentized_internal

# resíduo studentizado com i-ésima observação deletada (externamente)
res_stud_del = infl.resid_studentized_external

# DFFITS
(dffits,p) = infl.dffits

# Distância de Cook
(cook,p) = infl.cooks_distance
```

[26]:

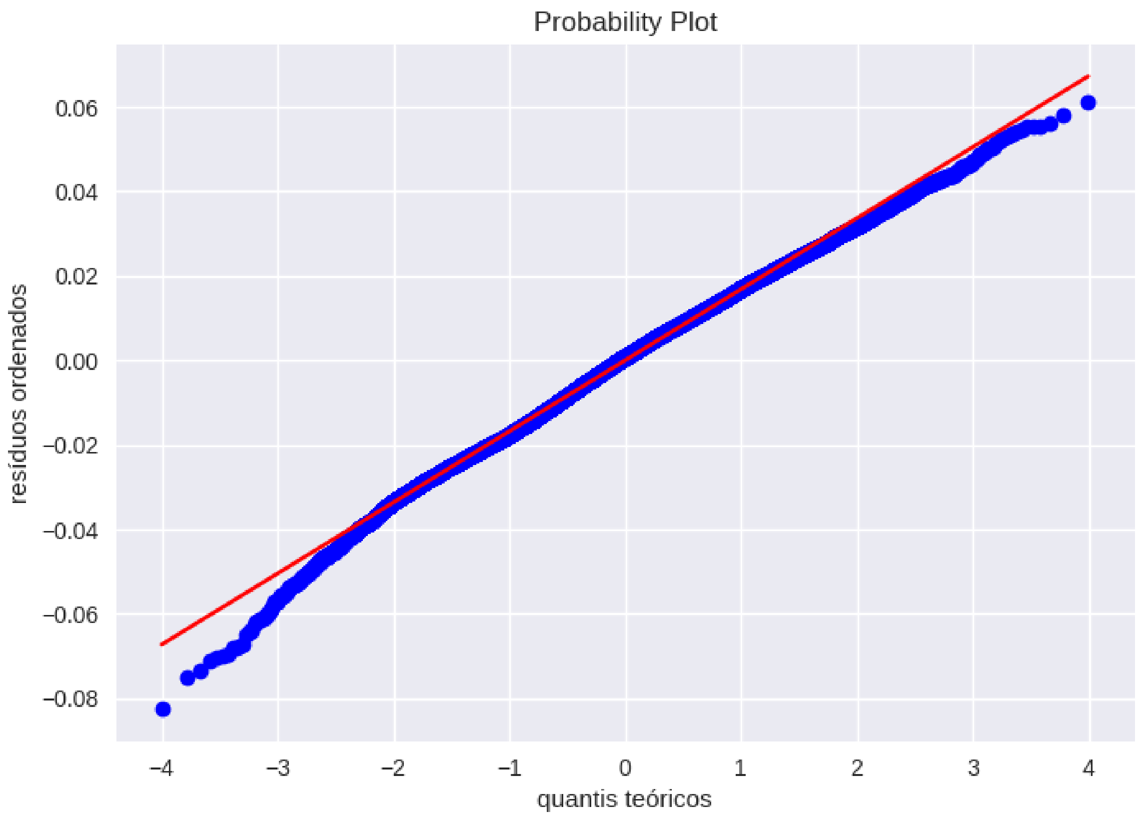
```python
fig, (ax1, ax2, ax3) = plt.subplots(3)
ax1.scatter(ypred, residuo)
ax1.set_ylabel('$y-\hat{y}$')
ax1.set_title('Resíduos')
#ax1.hlines(0,xmin=min(ypred),xmax=max(ypred),color='gray')

ax2.scatter(ypred, res_stud)
ax2.set_ylabel('studentizado')
ax2.hlines(0,xmin=min(ypred),xmax=max(ypred),color='gray')

ax3.scatter(ypred, res_stud_del)
ax3.set_ylabel('studentizado -(i)')

ax3.hlines(0,xmin=min(ypred),xmax=max(ypred),color='gray')
ax3.set_xlabel('$\hat{Y}$')

for ax in fig.get_axes():
    ax.label_outer()
```



[27]:
```
fig, (ax1, ax2, ax3) = plt.subplots(3)
ax1.scatter(df.index, dffits)
ax1.set_ylabel('DFFITS')
ax1.set_title('Detecção de pontos influentes')

#ax2.hlines(0,xmin=1,xmax=102,color='gray')
ax2.scatter(df.index, cook)
ax2.set_ylabel('distância de Cook')

ax3.scatter(df.index, hii)
ax3.set_ylabel('$h_{ii}$')
ax3.set_xlabel('índice')

for ax in fig.get_axes():
    ax.label_outer()
```



[28]:
```
# Verificando a suposição de distribuição Normal dos resíduos


stats.probplot(residuo, plot=plt)
plt.xlabel('quantis teóricos')
plt.ylabel('resíduos ordenados')
plt.show()
```



[29]:
```
#!pip install plotly
```

[30]:
```
# x and y given as DataFrame columns
import plotly.express as px

fig = px.scatter(x = df.index, y=cook)
fig.show()
```

[31]:
```
df['grade'].describe()
```

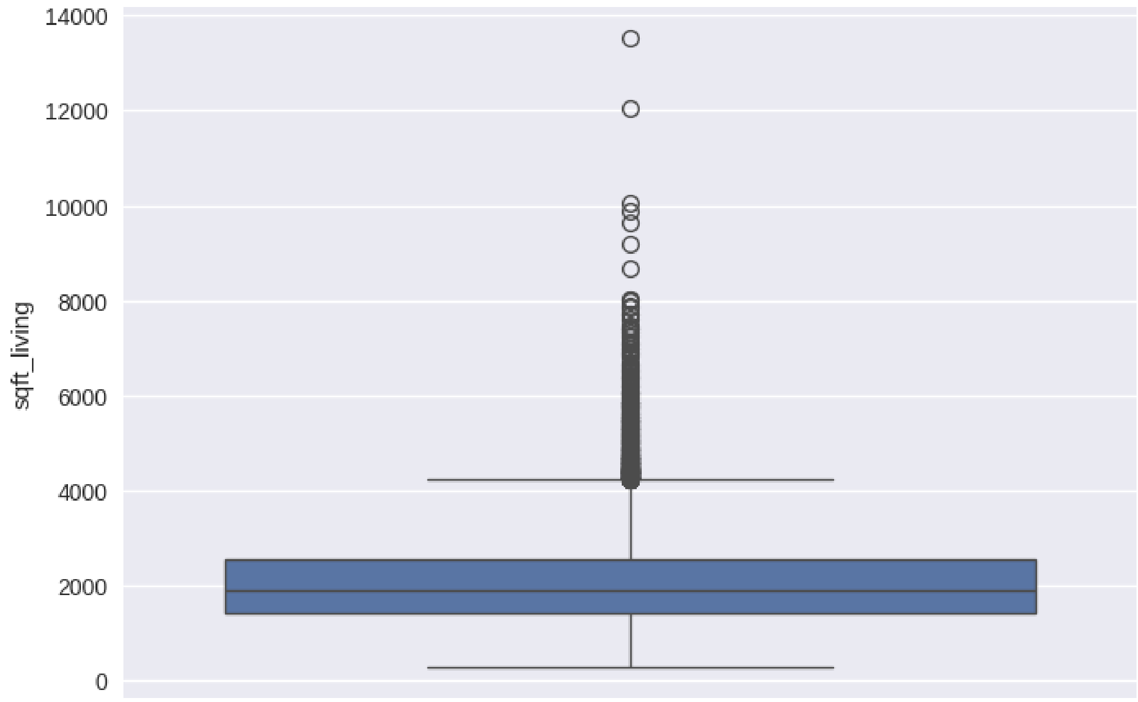
[31]:
```
count    21613.000000
mean         7.656873
std          1.175459
min          1.000000
25%          7.000000
50%          7.000000
75%          8.000000
max         13.000000
Name: grade, dtype: float64
```

[32]:
```python
df.iloc[12777,:]

#2.280.000
```

[32]:
```
id                        1225069038
date                 20140505T000000
price                      2280000.0
bedrooms                           7
bathrooms                        8.0
sqft_living                    13540
sqft_lot                      307752
floors                           3.0
waterfront                         0
view                               4
condition                          3
grade                             12
sqft_above                    9410.0
sqft_basement                   4130
yr_built                        1999
yr_renovated                       0
zipcode                        98053
lat                          47.6675
long                        -121.986
sqft_living15                   4850
sqft_lot15                    217800
price_transformado            4.1345
Name: 12777, dtype: object
```

[33]:
```python
sns.boxplot(df['sqft_living'])
```

[33]:
```
<Axes: ylabel='sqft_living'>
```



[34]:
```
px.scatter(y = df['price'], x=df['sqft_living'])
```

[35]:
```
px.scatter(y = df['price_transformado'], x=df['sqft_living'])
```

[36]:
```
res.params
```

[36]:
```
Intercept      3.983499
sqft_living    0.000009
grade          0.008750
dtype: float64
```

[37]:
```
res.predict()
```

[37]:
```
array([4.055955  , 4.06915421, 4.04331154, …, 4.05443567, 4.06869342,
       4.05443567])
```

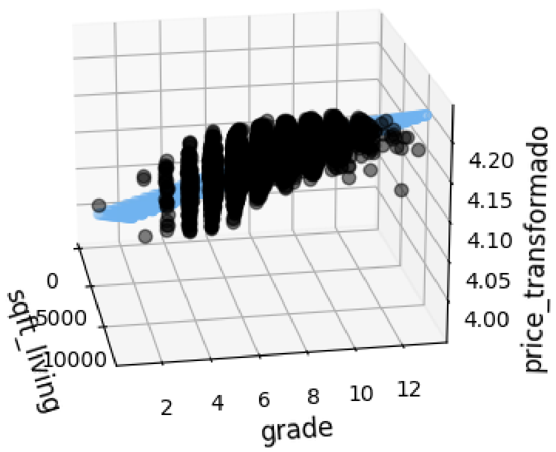
[38]:
```
X = df[['sqft_living', 'grade']].values.reshape(-1,2)
Y = df['price']

x = X[:, 0]
y = X[:, 1]
z = Y
```

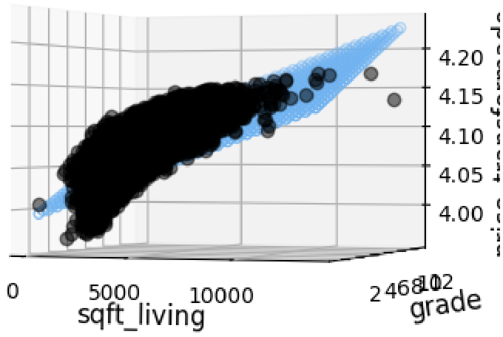
[39]:
```
# Visualização do modelo de regressão múltipla em Python
```

```python
## Fonte: https://aegis4048.github.io/
↪mutiple_linear_regression_and_visualization_in_python


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from mpl_toolkits.mplot3d import Axes3D

######################################## Preparação dos dados␣
↪########################################


X = df[['sqft_living', 'grade']].values.reshape(-1,2)
Y = df['price_transformado']

######################## Preparação para a visualização␣
↪###############################

x = X[:, 0]
y = X[:, 1]
z = Y

x_pred = np.linspace(290, 13540, 30)   # grade de valores para x
y_pred = np.linspace(1, 13, 30)  # grade de valores para y
xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)
model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T

################################################ Treinamento do modelo␣
↪############################################

ols = linear_model.LinearRegression()
model = ols.fit(X, Y)
predicted = model.predict(model_viz)

############################################## Avaliação␣
↪############################################

r2 = model.score(X, Y)
```

```
############################################### Plota o gráfico␣
↪###############################################

plt.style.use('default')

fig = plt.figure(figsize=(12, 4))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')


axes = [ax1, ax2]

for ax in axes:
    ax.plot(x, y, z, color='k', zorder=15, linestyle='none', marker='o', alpha=0.
↪5)
    ax.scatter(xx_pred.flatten(), yy_pred.flatten(), predicted,␣
↪facecolor=(0,0,0,0), s=20, edgecolor='#70b3f0')
    ax.set_xlabel('sqft_living', fontsize=12)
    ax.set_ylabel('grade', fontsize=12)
    ax.set_zlabel('price_transformado', fontsize=12)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')


ax1.view_init(elev=20, azim=-10)
ax2.view_init(elev=0, azim=290)


fig.suptitle('$R^2 = %.2f$' % r2, fontsize=20)

fig.tight_layout()
```


$R^2 = 0.53$





[40]:
```
for ii in np.arange(0, 180, 1):
    ax1.view_init(elev=20, azim=(2*ii))
    ax2.view_init(elev=0, azim=(2*ii))
    fig.savefig('gif_image%d.png' % ii)
```

MÉTRICAS COM BASES DE TREINO E TESTE

[41]:
```
# Ajusta o modelo de regressão linear múltipla para o preço das casas com duas␣
↪preditoras
from statsmodels.formula.api import ols

modelo = ols('price ~ sqft_living + grade', data=train_data)
res = modelo.fit()
```

[42]:
```
previsao = res.predict(X_teste)
```

[43]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mean_absolute_error(previsao, y_teste)
```

[43]: 168694.27817451468

[44]:
```
mean_squared_error(previsao, y_teste)
```

[44]: 79633647247.60545

Exercício

É possível melhorar esse modelo?

Testar outros métodos para a seleção de variáveis.