Roteiro da Aula

1. Correlação vs Regressão Linear
2. Regressão Linear: Conceitos Iniciais
3. Entendendo a regressão linear na prática
4. Como encontrar a reta de regressão ideal? (Minimizando o erro)
5. Exemplo prático: Peso e Altura
    - Conhecendo as bibliotecas: `sklearn` e `statsmodels`
    - Modelo com a presença de outliers
6. Características do Erro
7. Métricas para Análise dos Erros

---

1. Correlação vs Regressão Linear

Correlação	Regressão
Mede o grau de relação entre duas variáveis	Uma variável afeta a outra
Grau de interrelação	Baseada em causalidade (~Relação~ $\rightarrow$ Causa e efeito)
$\rho(x,y)$ = $\rho(y,x)$	Unidirecional
Único ponto	Linha

$\Rightarrow$ Correlação não implica causalidade!

In [ ]:
```
y_est = beta_0 + beta_1 * x
y_verdadeiro = y_est + erro
```

In [6]:
```
# Importação das bibliotecas
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('ggplot') # tema
```

2. Regressão Linear: Conceitos iniciais

In [7]:
```
x = np.arange(6)
y = x
```

In [8]:
```
x
```

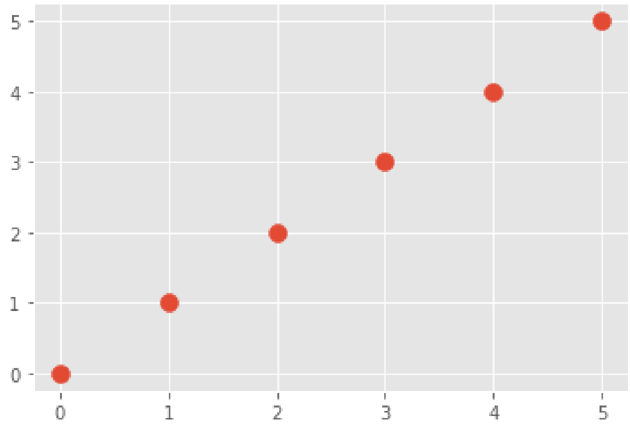
Out[8]:
```
array([0, 1, 2, 3, 4, 5])
```

In [9]:
```
y
```

Out[9]:
```
array([0, 1, 2, 3, 4, 5])
```

In [10]:
```
# Visualização
plt.scatter(x, y, s=100)
```

Out[10]:
```
<matplotlib.collections.PathCollection at 0x22cbe75ec50>
```



Equação da reta de regressão

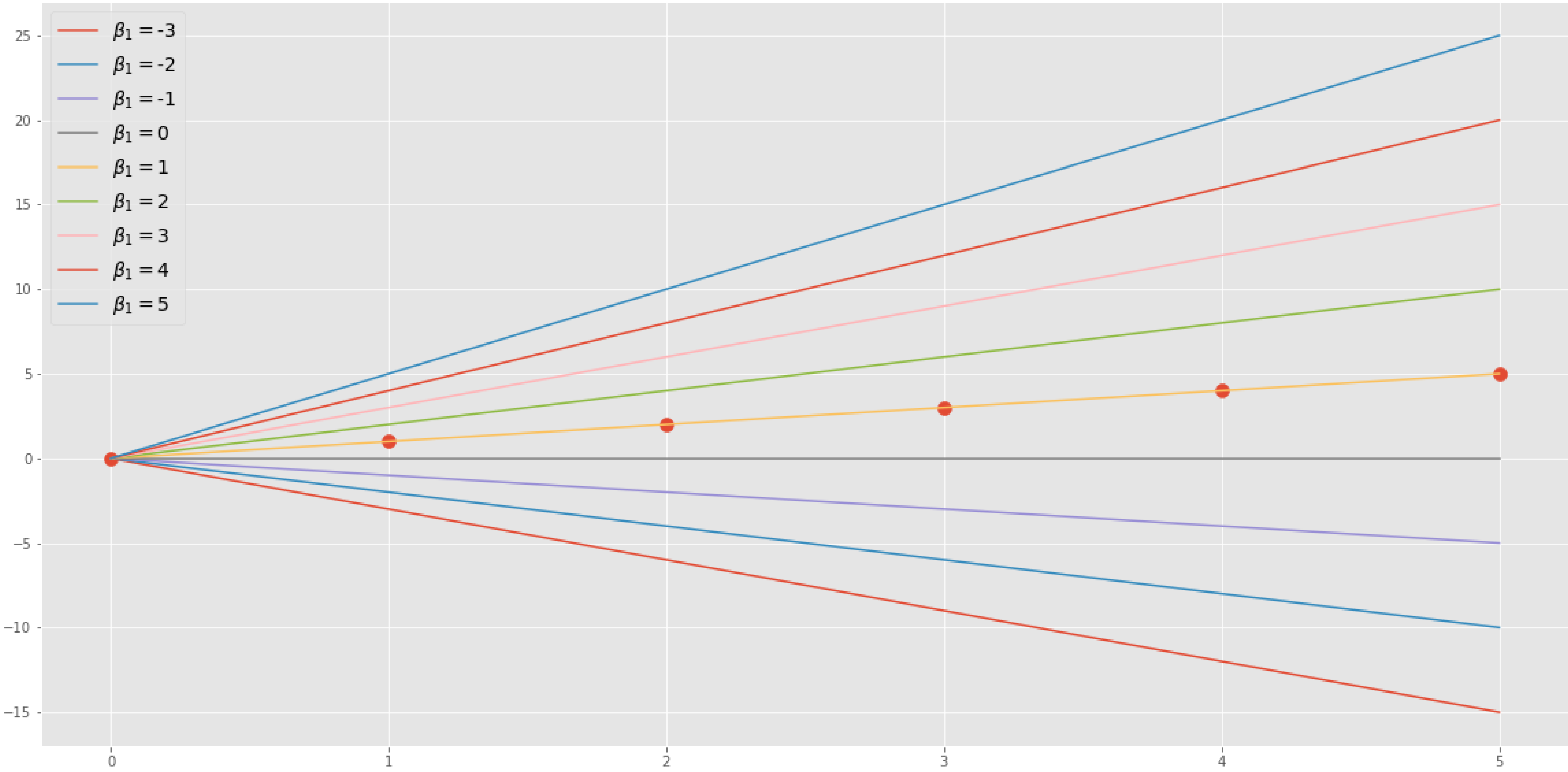
y = b0 + b1 * x

In [11]:
```
# Coeficiente x EQM
b0 = 0

plt.figure(figsize=(20,10))
plt.scatter(x, y, s=100)

coeficiente_erro = []

for b1 in range(-3, 6):
```

```
    y_est = b0 + b1 * x
    plt.plot(x, y_est, label='$\\beta_1 = $' + str(b1))

    erro = sum((y_est - y)**2) / 6 # Erro médio quadrático

    coeficiente_erro.append([b1, erro])

plt.legend(fontsize=14)
```

Out[11]:

```
<matplotlib.legend.Legend at 0x22cc08a0c40>
```



In [12]:

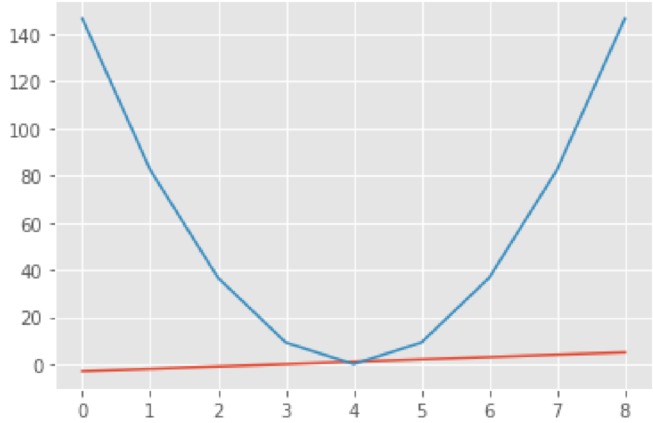
```
coeficiente_erro
```

Out[12]:
```
[[-3, 146.66666666666666],
 [-2, 82.5],
 [-1, 36.666666666666664],
 [0, 9.166666666666666],
 [1, 0.0],
 [2, 9.166666666666666],
 [3, 36.666666666666664],
 [4, 82.5],
 [5, 146.66666666666666]]
```

No nosso caso, $y = x$, então, $\beta_0 = 0$, $\beta_1 = 1$.

In [13]:
```
plt.plot(coeficiente_erro)
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x22cc0e36fe0>,
 <matplotlib.lines.Line2D at 0x22cc0e36f80>]
```



In [14]:
```
coef_erro = np.array(coeficiente_erro)
```

In [15]:
```
coef_erro[:,0]
```

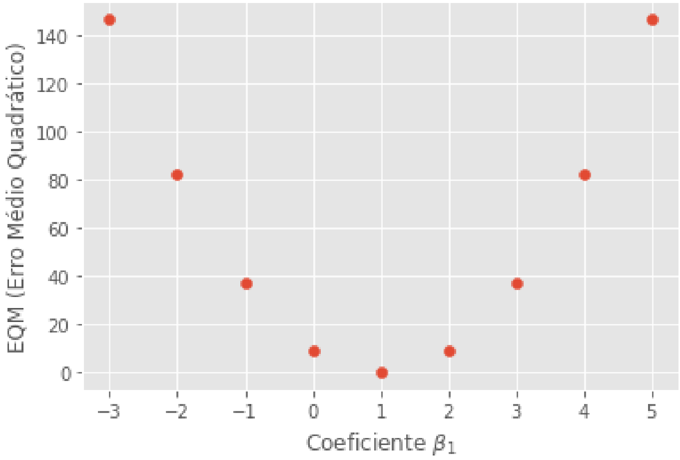
Out[15]:
```
array([-3., -2., -1.,  0.,  1.,  2.,  3.,  4.,  5.])
```

In [16]:
```
coef_erro[:,1]
```

Out[16]:
```
array([146.66666667,  82.5       ,  36.66666667,   9.16666667,
         0.        ,   9.16666667,  36.66666667,  82.5       ,
       146.66666667])
```

In [17]:
```
plt.scatter(coef_erro[:,0], coef_erro[:,1])
plt.xlabel('Coeficiente $\\beta_1$')
plt.ylabel('EQM (Erro Médio Quadrático)')
```

Out[17]:
```
Text(0, 0.5, 'EQM (Erro Médio Quadrático)')
```



Calculando os coeficientes β_0 e β_1 por meio das equações

$$ \beta_1 = \sum_{i=1}^N \frac{(x_i - \bar{x}) \cdot (y_i - \bar{y})}{(x_i - \bar{x})^2} $$$$ \beta_0 = \bar{y} - \beta_1\bar{x} $$

In [18]:
```
x
```
Out[18]:
```
array([0, 1, 2, 3, 4, 5])
```

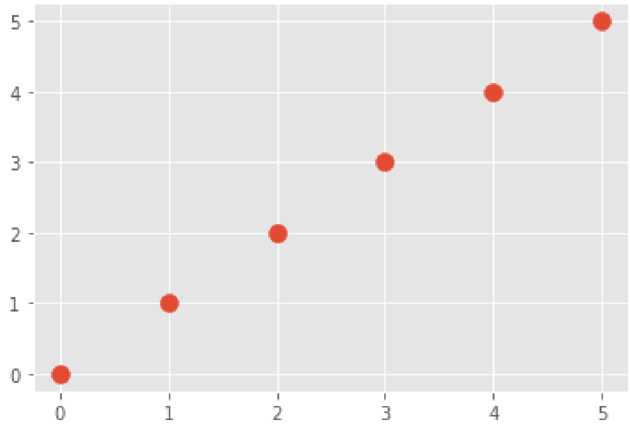
In [19]:
```
y
```
Out[19]:
```
array([0, 1, 2, 3, 4, 5])
```

In [20]:
```
x_mean = x.mean()
y_mean = y.mean()

x_mean, y_mean
```
Out[20]:
```
(2.5, 2.5)
```

In [21]:
```
x - x_mean
```
Out[21]:
```
array([-2.5, -1.5, -0.5,  0.5,  1.5,  2.5])
```

In [22]:
```
y - y_mean
```
Out[22]:
```
array([-2.5, -1.5, -0.5,  0.5,  1.5,  2.5])
```

In [23]:
```
(y - y_mean) * (y - y_mean)
```
Out[23]:
```
array([6.25, 2.25, 0.25, 0.25, 2.25, 6.25])
```

In [24]:
```
sum((y - y_mean) * (y - y_mean))
```
Out[24]:
```
17.5
```
In [25]:
```
sum(x - x_mean)
```
Out[25]:
```
0.0
```
In [26]:
```
b1 = sum( (x - x_mean) * (y - y_mean) ) / sum( (x - x_mean)**2 )
b1
```
Out[26]:
```
1.0
```
In [27]:
```
b0 = y_mean - (b1 * x_mean)
b0
```
Out[27]:
```
0.0
```
In [28]:
```
plt.scatter(x, y, s=100)
#plt.plot(x, b0 + b1 * x, color='blue')
```
Out[28]:
```
<matplotlib.collections.PathCollection at 0x22cc11c9f30>
```


Regressão Linear Simples: Altura $\Rightarrow$ Peso

In [30]:
```
df = pd.read_csv('..\weight-height.csv')
```
In [31]:
```
df.head()
```

Out[31]:

	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801

In [32]:
```
df.describe().T
```

Out[32]:

	count	mean	std	min	25%	50%	75%	max
Height	10000.0	66.367560	3.847528	54.263133	63.505620	66.318070	69.174262	78.998742
Weight	10000.0	161.440357	32.108439	64.700127	135.818051	161.212928	187.169525	269.989699

Conversão das unidades

In [33]:
```
df['altura'] = df['Height'] * 2.54 # pol => cm
```

In [34]:
```
df['peso'] = df['Weight'] * 0.453592 # libras => kg
```

In [35]:
```
df.head()
```

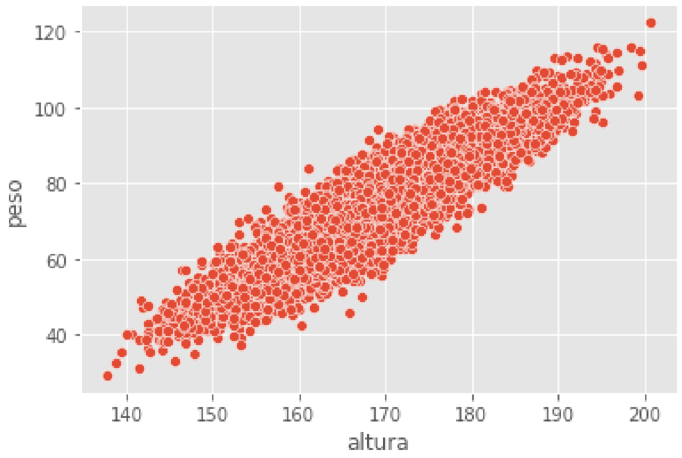
Out[35]:

	Gender	Height	Weight	altura	peso
0	Male	73.847017	241.893563	187.571423	109.720985
1	Male	68.781904	162.310473	174.706036	73.622732
2	Male	74.110105	212.740856	188.239668	96.497550
3	Male	71.730978	220.042470	182.196685	99.809504
4	Male	69.881796	206.349801	177.499761	93.598619

Visualização gráfica

In [36]:
```
sns.scatterplot(data=df, x='altura', y='peso')
```

Out[36]:
```
<AxesSubplot:xlabel='altura', ylabel='peso'>
```



Calculando a correlação entre as features

In [37]:
```
from scipy.stats import pearsonr
```

In [38]:
```
pearsonr(df['altura'], df['peso'])
```

Out[38]:
```
(0.9247562987409151, 0.0)
```

- Correlação muito forte: [-1, -0.8] or [0.8, 1]

$$ \beta_1 = \sum_{i=1}^N \frac{(x_i - \bar{x}) \cdot (y_i - \bar{y})}{(x_i - \bar{x})^2} $$$$ \beta_0 = \bar{y} - \beta_1\bar{x} $$

In [39]:
```
x = df['altura']
```

In [40]:
```
y = df['peso']
```

In [41]:
```
x_mean = x.mean()
y_mean = y.mean()

x_mean, y_mean
```

Out[41]:
```
(168.57360177724598, 73.22805433651739)
```

In [42]:
```
b1 = sum( (x - x_mean) * (y - y_mean) ) / sum( (x - x_mean)**2 )
b1
```

Out[42]:
```
1.3781495809287967
```

In [43]:
```
b0 = y_mean - (b1 * x_mean)
```

```
b0
```

Out[43]:

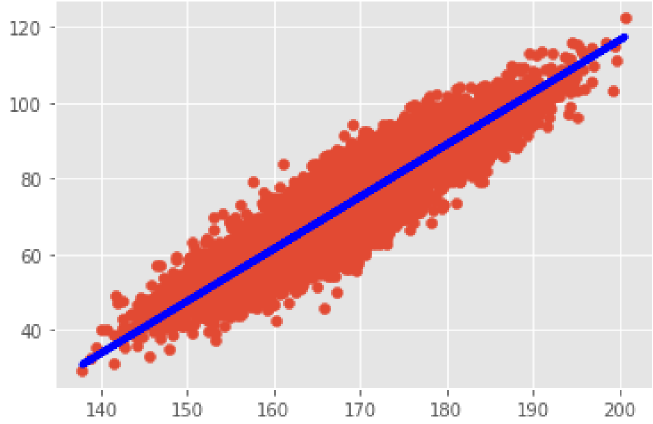
```
-159.091584308452
```

In [44]:

```
plt.scatter(df['altura'], df['peso'])
plt.plot(x, b0 + b1 * x, color='blue', linewidth=4)
```

Out[44]:

```
[<matplotlib.lines.Line2D at 0x22cc1c1b400>]
```



In [45]:

```
y_est = b0 + b1 * x
```

In [46]:

```
y
```

Out[46]:

```
0       109.720985
1        73.622732
2        96.497550
3        99.809504
4        93.598619
           ...
9995     62.041159
9996     77.504315
9997     58.275377
9998     74.322166
9999     51.550324
Name: peso, Length: 10000, dtype: float64
```

In [47]:

```
erro = y - y_est
erro
```

```
Out[47]: 0        10.311091
1        -8.056735
2        -3.833285
3         7.806803
4         8.068981
           ...
9995    -10.504621
9996      1.827329
9997     -6.202942
9998     -8.240614
9999     -6.193921
Length: 10000, dtype: float64
```

Scikit Learning

```
In [49]: from sklearn.linear_model import LinearRegression
```

```
In [50]: lr = LinearRegression()
```

```
In [51]: lr
```

```
Out[51]: ▾ LinearRegression
LinearRegression()
```

```
In [52]: df['altura']
```

```
Out[52]: 0       187.571423
1       174.706036
2       188.239668
3       182.196685
4       177.499761
           ...
9995    168.078536
9996    170.350573
9997    162.224700
9998    175.346978
9999    157.338385
Name: altura, Length: 10000, dtype: float64
```

```
In [53]: X = df['altura'].values
y = df['peso'].values
```

```
In [54]: X
```

```
Out[54]: array([187.57142322, 174.70603628, 188.2396677 , ..., 162.22470022,
       175.34697755, 157.33838453])
```

```
In [55]: y
```

Out[55]:
```
array([109.72098511,  73.62273185,  96.49755015, ...,  58.2753768 ,
        74.32216565,  51.55032378])
```

Entendendo o método `reshape`

In [56]:
```python
array = np.ones(25)
array
```

Out[56]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1.])
```

In [57]:
```python
array.reshape(5, 5)
```

Out[57]:
```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

In [58]:
```python
X_reshape = X.reshape(-1, 1)
```

In [59]:
```python
X_reshape.shape # Vetor coluna
```

Out[59]:
```
(10000, 1)
```

In [60]:
```python
X.shape        # Vetor linha
```

Out[60]:
```
(10000,)
```

In [61]:
```python
# Treinamento do modelo: calcular os parâmetros do modelo

lr.fit(X.reshape(-1, 1), y)
```

Out[61]:
```
▾ LinearRegression
LinearRegression()
```

In [62]:
```python
lr.coef_ # coeficientes
```

Out[62]:
```
array([1.37814958])
```

In [63]:
```python
lr.intercept_ # intercepto
```

Out[63]:
```
-159.09158430845105
```

Statsmodels

In [67]:
```
import statsmodels.api as sm
```

In [68]:
```
x = sm.add_constant(X)

model = sm.OLS(y, x).fit()
```

In [69]:
```
model.summary()
```

Out[69]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.855
Model:	OLS	**Adj. R-squared:**	0.855
Method:	Least Squares	**F-statistic:**	5.904e+04
Date:	Tue, 14 Jun 2022	**Prob (F-statistic):**	0.00
Time:	10:01:33	**Log-Likelihood:**	-31313.
No. Observations:	10000	**AIC:**	6.263e+04
Df Residuals:	9998	**BIC:**	6.265e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-159.0916	0.958	-166.109	0.000	-160.969	-157.214
x1	1.3781	0.006	242.975	0.000	1.367	1.389

Omnibus:	2.141	**Durbin-Watson:**	1.677
Prob(Omnibus):	0.343	**Jarque-Bera (JB):**	2.150
Skew:	0.036	**Prob(JB):**	0.341
Kurtosis:	2.991	**Cond. No.**	2.92e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.92e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Influência dos Outliers

In [70]:
```
df.head()
```

Out[70]:

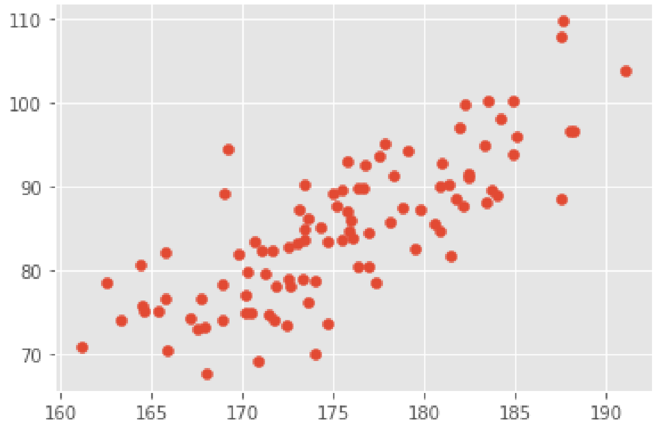
	Gender	Height	Weight	altura	peso
0	Male	73.847017	241.893563	187.571423	109.720985
1	Male	68.781904	162.310473	174.706036	73.622732
2	Male	74.110105	212.740856	188.239668	96.497550
3	Male	71.730978	220.042470	182.196685	99.809504
4	Male	69.881796	206.349801	177.499761	93.598619

Obtendo apenas 100 amostras (como array)

In [71]:
```
X = df['altura'].values[:100]
y = df['peso'].values[:100]
```

In [72]:
```
# Visualizando
plt.scatter(X, y)
```

Out[72]:
```
<matplotlib.collections.PathCollection at 0x22cc5ad1f00>
```



Modelo sem a presença dos outliers

In [73]:
```
lr = LinearRegression()
```

In [74]:
```
lr.fit(X.reshape(-1, 1), y)
```

Out[74]:

```
▾ LinearRegression
LinearRegression()
```

In [75]:
```python
lr.coef_[0]
```

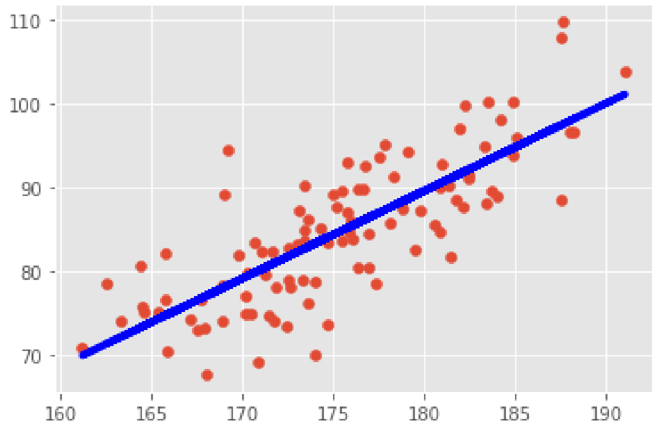
Out[75]:
```
1.0449527098060838
```

In [76]:
```python
lr.intercept_
```

Out[76]:
```
-98.52075866207376
```

In [77]:
```python
plt.scatter(X, y)
plt.plot(X, lr.intercept_ + lr.coef_[0] * X, color='blue', linewidth=4)
```

Out[77]:
```
[<matplotlib.lines.Line2D at 0x22cc5b4ac20>]
```



In [78]:
```python
lr.score(X.reshape(-1, 1), y)
```

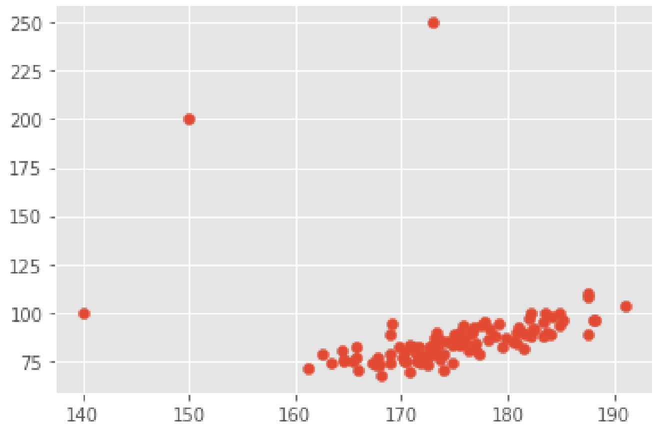
Out[78]:
```
0.6214261131940761
```

Modelo com a presença dos outliers

In [79]:
```python
X2 = np.append(X, [150, 140, 173])
y2 = np.append(y, [200, 100, 250])
```

In [80]:
```python
plt.scatter(X2, y2)
```

Out[80]:
```
<matplotlib.collections.PathCollection at 0x22cc5bb6440>
```



In [81]:
```
lr2 = LinearRegression() # Modelo com os outliers
```

In [82]:
```
lr2.fit(X2.reshape(-1, 1), y2)
```

Out[82]:
```
▾ LinearRegression
LinearRegression()
```

In [83]:
```
lr2.coef_[0]
```

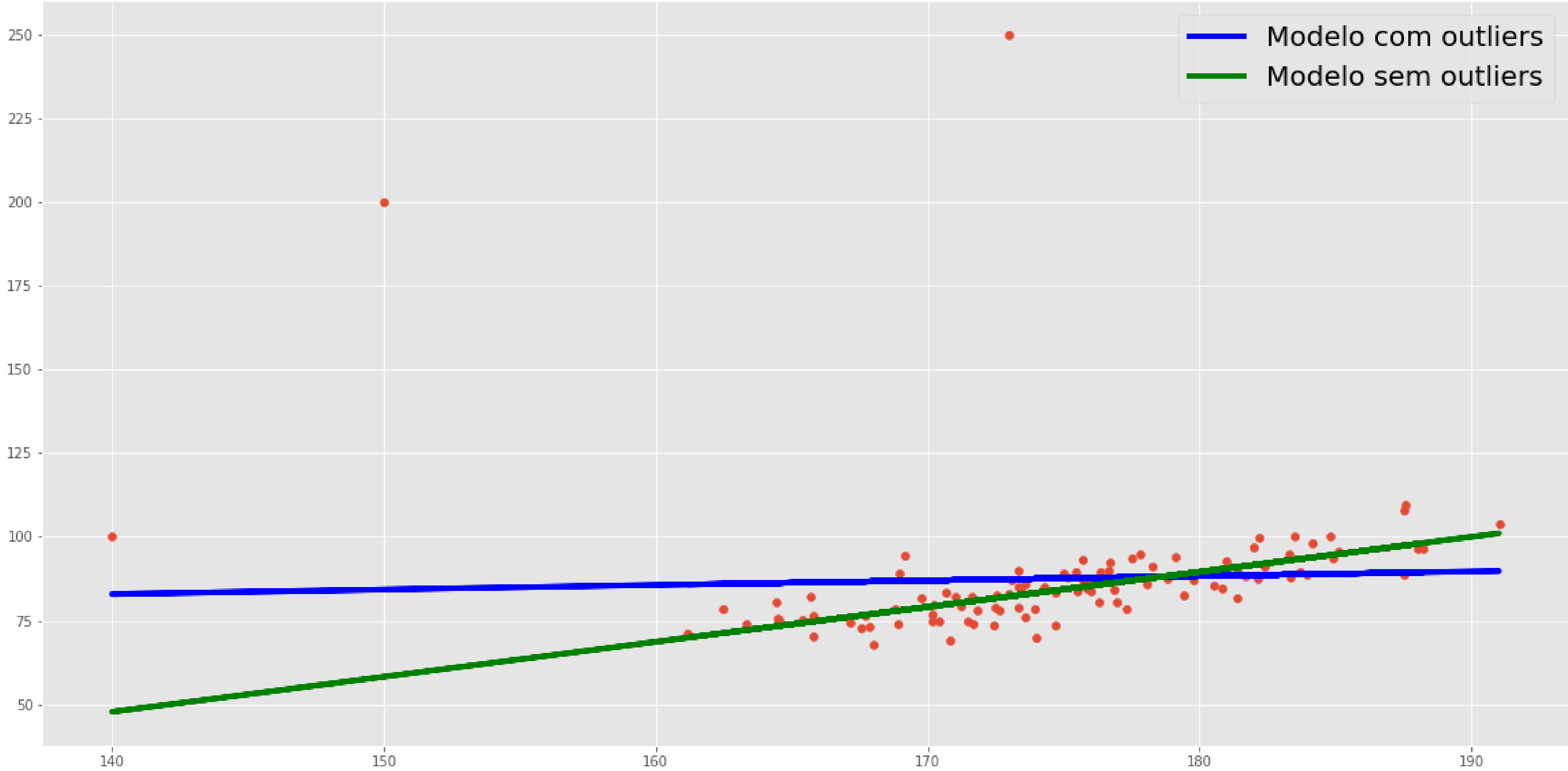
Out[83]:
```
0.1348777878056209
```

In [84]:
```
lr2.intercept_
```

Out[84]:
```
64.05035131391783
```

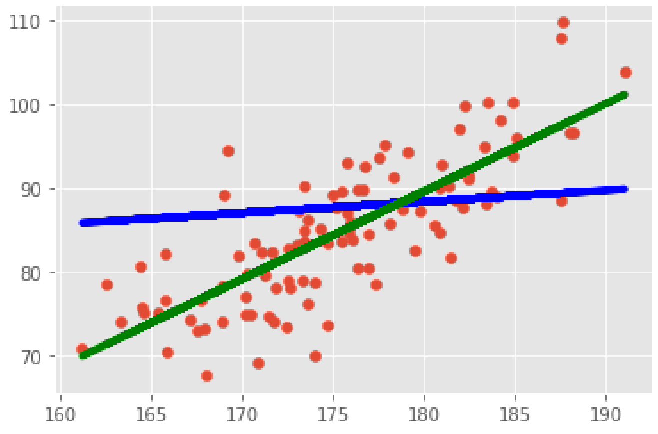
In [85]:
```
plt.figure(figsize=(20, 10))

# plot de dispersão dos dados originais com os 3 outliers
plt.scatter(X2, y2)

# plot da reta de regressão do modelo treinado com os dados + outliers
plt.plot(X2, lr2.intercept_ + lr2.coef_[0] * X2, color='blue', linewidth=4, label='Modelo com outliers')
plt.plot(X2, lr.intercept_ + lr.coef_[0] * X2, color='green', linewidth=4, label='Modelo sem outliers')

plt.legend(fontsize=20)
```

Out[85]:
```
<matplotlib.legend.Legend at 0x22cc5bdc520>
```



In [86]:
```
# plot de dispersão dos dados originais com os 3 outliers
plt.scatter(X, y)

# plot da reta de regressão do modelo treinado com os dados + outliers
plt.plot(X, lr2.intercept_ + lr2.coef_[0] * X, color='blue', linewidth=4, label='Modelo com outliers')
plt.plot(X, lr.intercept_ + lr.coef_[0] * X, color='green', linewidth=4, label='Modelo sem outliers')
```

Out[86]:
```
[<matplotlib.lines.Line2D at 0x22cc5cbe560>]
```



Características dos Resíduos

1. Não devem ser correlacionados

Sem outliers

In [87]:
```
y1_est = lr.intercept_ + lr.coef_[0] * X # sem outliers
```

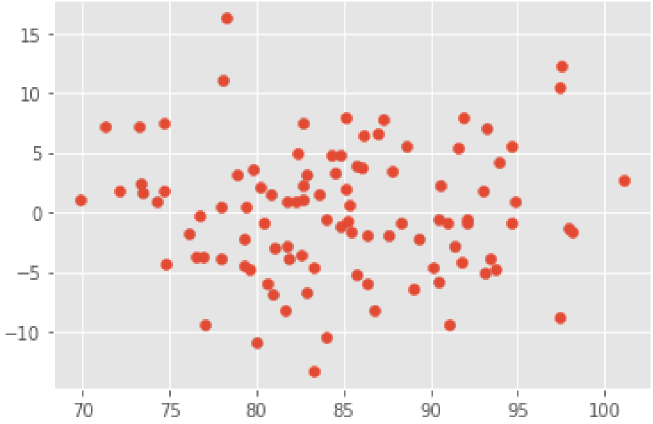
In [88]:
```
residuos = y - y1_est
```

In [89]:
```
residuos
```

Out[89]:
```
array([ 12.23847679, -10.41605551,  -1.68324204,   7.94334299,
         6.64052067, -10.93862649,  -0.61900139,  -6.69814429,
         0.44059742,   1.03745092,  -5.80209135,   5.32445947,
         2.42695011,   0.56230009,  -0.6995153 ,  -2.92027033,
        -4.77351824,   7.13994933,  -1.95565208,   0.95364938,
        -2.76040881,   3.90538528,  -9.38700051,   2.67509001,
        -8.19507602,   4.89992669,  -1.19723156,  -0.89016356,
        -0.95171279, -13.27246275,   1.44755404,  -1.40390926,
        -2.86292813,   7.97381595,   6.45756184,  -0.83647932,
         4.7592617 ,   3.55224787,  -1.82181775,   1.83036849,
        -3.81602721,  -5.06552018,   1.76518136,  -3.77840911,
        -5.97519411,  -1.85630938,   2.21727739,   1.7942771 ,
         2.0011387 ,   0.97298052,   1.6962389 ,   3.44560591,
        -4.60414973,   2.0976939 ,  -2.29359128,  -3.90511334,
         3.15048432,  -2.18109307,  -0.59787555,   7.73223693,
        -0.84027929,   3.76451322,  -8.82465138,  -4.48405706,
        -9.45186959,  -4.3218979 ,   1.00529284,   5.48617176,
         0.9219087 ,   4.76525105,  -1.65497362,   5.47467014,
        -6.00237573,  -6.90475905,   4.14602364,   2.23261243,
        -0.93777685,   6.96836941,  11.09236542,  -0.23805277,
         1.45526328,   7.49207415,  10.49006559,  -8.25740867,
        -3.66288666,  -4.58696462,  16.26641358,   3.25160357,
        -3.63755472,  -3.89629064,  -0.63676411,   7.47547717,
        -6.36261256,   7.24550945,  -4.75298696,   0.8551966 ,
        -4.20124643,   3.09104347,   0.4628156 ,  -5.21066612])
```

In [90]:
```python
plt.scatter(y1_est, residuos)
```

Out[90]:
```
<matplotlib.collections.PathCollection at 0x22cc5d21960>
```



Com outliers

In [91]:
```python
y2_est = lr2.intercept_ + lr2.coef_[0] * X2
```

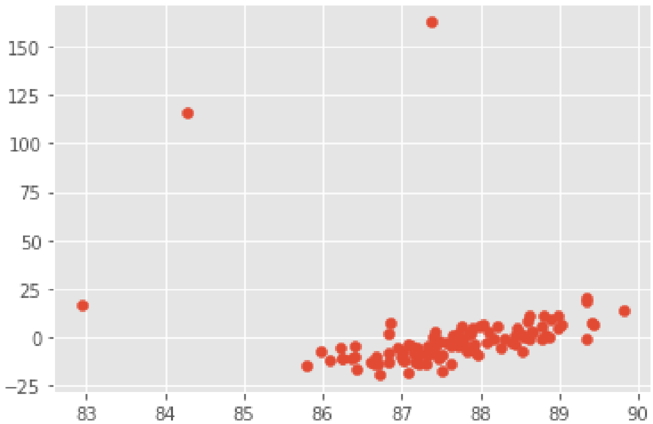
In [92]:
```python
residuos2 = y2 - y2_est
```

In [93]:
```
residuos2
```

Out[93]:
```
array([ 2.03714152e+01, -1.39915832e+01,  7.05784888e+00,  1.11848670e+01,
        5.60749229e+00, -1.80483176e+01, -4.18718463e+00, -1.12754887e+01,
       -7.21015847e+00, -1.48482430e+01, -3.79864611e+00,  8.35753983e+00,
       -1.04309416e+01, -1.85473696e+00, -3.20748539e+00, -9.12242185e+00,
        5.68501780e-02, -7.54872974e+00, -3.54747497e+00, -4.57725063e+00,
       -8.33974740e+00,  1.84981107e+00, -1.90484186e+01,  1.39493793e+01,
       -1.38249318e+01, -1.50097935e-01, -4.06141957e+00,  4.82445030e+00,
       -7.67253297e+00, -1.75079233e+01, -2.50659706e+00,  7.16081248e+00,
       -9.01076042e-02,  5.31889545e+00,  4.72043809e+00, -6.69484836e-01,
        1.86395908e+00, -3.69496533e+00, -1.22733751e+01, -1.20958044e+01,
       -9.25832078e+00, -7.64253209e-01, -9.92763090e+00, -1.35761060e+01,
       -1.24900996e+01, -2.37387533e+00, -2.58354939e+00,  6.05690158e+00,
       -6.54483323e-01,  6.84828835e+00, -1.11240805e+01,  3.11010921e+00,
       -8.85108754e+00, -4.83212039e+00, -1.00090846e+01, -1.27565087e+01,
       -1.41636280e+00, -1.15064984e+00,  1.45305297e+00,  7.00334330e+00,
        1.64351428e+00,  1.96914369e+00, -7.57552776e-01, -1.21858226e+01,
       -6.93274643e+00, -1.59940106e+01, -3.74385825e+00,  1.11187853e+01,
       -4.23133536e+00,  1.45509866e+00, -4.05712434e+00,  5.92436203e+00,
       -7.53949408e+00, -1.32233012e+01,  9.18398154e+00,  4.36921296e+00,
        2.48453318e+00,  1.13984833e+01,  2.27120049e+00, -1.01770881e+01,
       -4.90515301e+00,  2.67907029e+00,  1.85747148e+01, -9.47381898e+00,
       -1.37624763e+01, -2.85250690e+00,  7.63842567e+00,  6.81934165e-02,
       -8.45542866e+00,  6.99471001e-01,  2.79995535e+00, -4.29430181e+00,
       -5.64175826e+00, -5.69190909e+00, -1.22081788e+01, -1.11909668e+01,
       -1.02847921e+00, -4.98983340e+00, -8.45393913e+00, -7.33076524e+00,
        1.15717981e+02,  1.70667584e+01,  1.62615791e+02])
```

In [94]:
```
plt.scatter(y2_est, residuos2)
```

Out[94]:
```
<matplotlib.collections.PathCollection at 0x22cc5d9cc40>
```



2. Ter média zero

Sem outliers

In [95]:
```
residuos.mean()
```
Out[95]:
```
3.467448550509289e-14
```

Com outliers

In [96]:
```
residuos2.mean()
```
Out[96]:
```
1.2969129545912509e-14
```

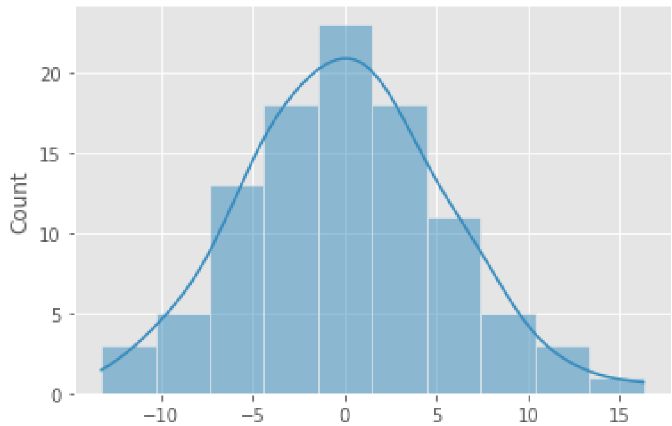
4. Distribuição normal dos resíduos

Sem outliers

In [97]:
```
from scipy.stats import normaltest
```

In [98]:
```
normaltest(residuos)
```
Out[98]:
```
NormaltestResult(statistic=0.9490854261141337, pvalue=0.6221695014128703)
```

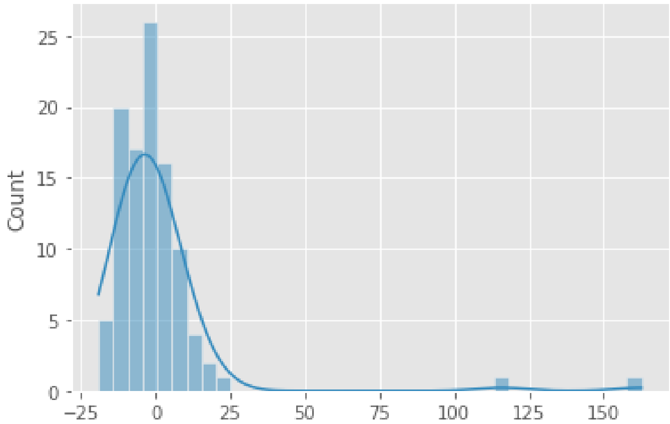
In [99]:
```
sns.histplot(residuos, kde=True)
```
Out[99]:
```
<AxesSubplot:ylabel='Count'>
```



Com outliers

In [100…
```
sns.histplot(residuos2, kde=True)
```

Out[100]: <AxesSubplot:ylabel='Count'>



In [101…
```
model
```

Out[101]: <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x22cc0cc91e0>

In [102…
```
model.resid
```

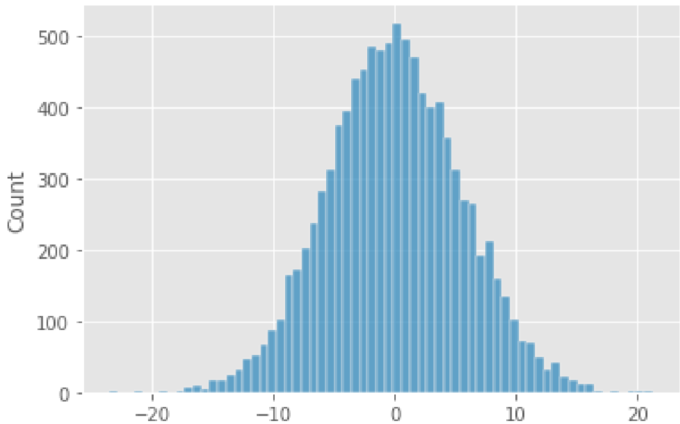
Out[102]:
```
array([10.31109111, -8.05673452, -3.83328469, ..., -6.20294152,
       -8.24061367, -6.19392062])
```

In [103…
```
normaltest(model.resid)
```

Out[103]: NormaltestResult(statistic=2.1407382479464676, pvalue=0.3428819281169143)

In [104…
```
sns.histplot(model.resid)
```

Out[104]: <AxesSubplot:ylabel='Count'>



Métricas para Análise dos Erros

$$ SQE = \sum_{i=1}^N \epsilon_i^2 = \frac{1}{N} \sum_{i=1}^N \left[y_i - \hat{y}_i\right]^2 = \frac{1}{N} \sum_{i=1}^N \left[y_i - (\beta_0 + \beta_1x_i)\right]^2 $$
$$ SQR = \sum_{i=1}^N \left(\hat{y}_i - \overline{y}\right)^2 $$
$$ SQT = \sum_{i=1}^N \left(y_i - \overline{y}\right)^2 $$

1. R-Quadrado (R^2)

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

In [105…
```
from sklearn.metrics import r2_score
```

In [106…
```
lr.score(X.reshape(-1, 1), y)
```

Out[106]:
```
0.6214261131940761
```

In [107…
```
r2_score(y, y1_est)
```

Out[107]:
```
0.6214261131940761
```

2. MAE

In [108…
```
from sklearn.metrics import mean_absolute_error
```

In [109…
```
mean_absolute_error(y, y1_est)
```

Out[109]:
```
4.260493303495109
```

In [110…
```
mean_absolute_error(y2, y2_est)
```

Out[110]:
```
9.640274384813585
```

3. MSE

In [111…
```
from sklearn.metrics import mean_squared_error
```

In [112…
```
mean_squared_error(y, y1_est)
```

Out[112]:
```
28.728718184843036
```

In [113…
```
mean_squared_error(y2, y2_est)
```

Out[113]:
```
460.66295293941755
```

4. RMSE

In [114…
```
np.sqrt(mean_squared_error(y, y1_est))
```
Out[114]:
```
5.359917740492202
```

In [115…
```
np.sqrt(mean_squared_error(y2, y2_est))
```
Out[115]:
```
21.4630601951217
```

In [ ]:
```

```

5. RMSLE

In [116…
```
from sklearn.metrics import mean_squared_log_error
```

In [117…
```
mean_squared_log_error(y, y1_est)
```
Out[117]:
```
0.003965422358995947
```

In [118…
```
mean_squared_log_error(y2, y2_est)
```
Out[118]:
```
0.027940307257655395
```

In [ ]:
```

```