

# Projeto Final A3: Desenvolvimento de um Agente Inteligente

## ✓ 1.2 Conhecendo o Dataset

---

### ✓ Importando a biblioteca pandas

<https://pandas.pydata.org/>

```
import pandas as pd
```

## O Dataset e o Projeto

---

### Descrição:

O objetivo principal do projeto é desenvolver um sistema de avaliação imobiliária utilizando análises preliminares e adaptando o modelo de acordo com o aprendizado de máquina, machine learning.

Nosso \*dataset\* é uma amostra aleatória de tamanho 5000 de imóveis disponíveis para venda no município do Rio de Janeiro.

### Dados:

- **Valor** - Valor (R\$) de oferta do imóvel
- **Area** - Área do imóvel em m<sup>2</sup>
- **Dist\_Praia** - Distância do imóvel até a praia (km) (em linha reta)
- **Dist\_Farmacia** - Distância do imóvel até a farmácia mais próxima (km) (em linha reta)


### ✓ Leitura dos dados

```
from google.colab import drive
drive.mount('/content/drive')
dados = pd.read_csv("/content/drive/MyDrive/Colab_Notebooks/DataScience/reg_linear_II/Dad
```



 Mounted at /content/drive

Visualizar os dados

```
dados.head()
```



	Valor	Area	Dist_Praia	Dist_Farmacia
0	4600000	280	0.240925	0.793637
1	900000	208	0.904136	0.134494
2	2550000	170	0.059525	0.423318
3	550000	100	2.883181	0.525064
4	2200000	164	0.239758	0.192374



Próximas etapas:

Gerar código com dados

☐ Ver gráficos recomendados

New interactive sheet

Verificando o tamanho do dataset

```
dados.shape
```



(5000, 4)

1.3 Análises Preliminares

Estatísticas descritivas

```
dados.describe().round(2)
```



	Valor	Area	Dist_Praia	Dist_Farmacia
<b>count</b>	5000.00	5000.00	5000.00	5000.00
<b>mean</b>	1402926.39	121.94	3.02	0.50
<b>std</b>	1883268.85	90.54	3.17	0.29
<b>min</b>	75000.00	16.00	0.00	0.00
<b>25%</b>	460000.00	70.00	0.44	0.24
<b>50%</b>	820000.00	93.00	1.48	0.50
<b>75%</b>	1590000.00	146.00	5.61	0.75
<b>max</b>	25000000.00	2000.00	17.96	1.00



## ✓ Matriz de correlação

O **coeficiente de correlação** é uma medida de associação linear entre duas variáveis e situa-se entre **-1** e **+1** sendo que **-1** indica associação negativa perfeita e **+1** indica associação positiva perfeita.

```
dados.corr().round(4)
```



	Valor	Area	Dist_Praia	Dist_Farmacia
<b>Valor</b>	1.0000	0.7110	-0.3665	-0.0244
<b>Area</b>	0.7110	1.0000	-0.2834	-0.0310
<b>Dist_Praia</b>	-0.3665	-0.2834	1.0000	0.0256
<b>Dist_Farmacia</b>	-0.0244	-0.0310	0.0256	1.0000



## ✓ 2.1 Comportamento da Variável Dependente (Y)

### ✓ Importando biblioteca seaborn

<https://seaborn.pydata.org/>

O Seaborn é uma biblioteca Python de visualização de dados baseada no matplotlib. Ela fornece uma interface de alto nível para desenhar gráficos estatísticos.

```
import seaborn as sns
```

## ✓ Configurações de formatação dos gráficos

```
# palette -> Accent, Accent_r, Blues, Blues_r, BrBG, BrBG_r, BuGn, BuGn_r, BuPu, BuPu_r,
sns.set_palette("Accent")
```

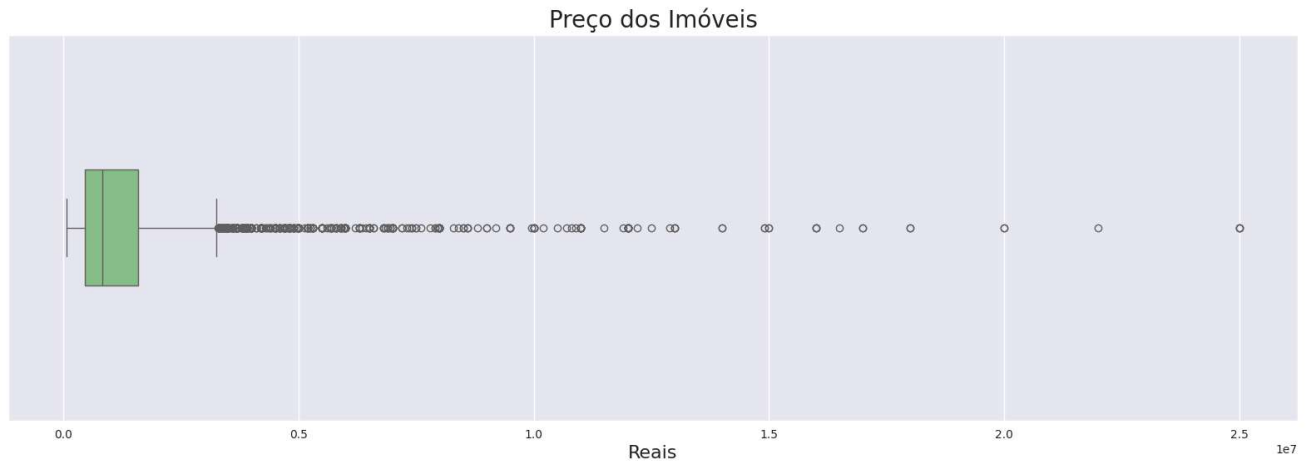
```
# style -> white, dark, whitegrid, darkgrid, ticks
sns.set_style("darkgrid")
```

## ✓ Box plot da variável *dependente* (y)

<https://seaborn.pydata.org/generated/seaborn.boxplot.html?highlight=boxplot#seaborn.boxplot>

```
ax = sns.boxplot(data = dados.Valor, orient = "h", width = 0.3)
ax.figure.set_size_inches(20, 6)
ax.set_title('Preço dos Imóveis', fontsize=20)
ax.set_xlabel('Reais', fontsize=16)
ax
```

```
<Axes: title={'center': 'Preço dos Imóveis'}, xlabel='Reais'>
```



## ✓ 2.2 Distribuição de Frequências

### ✓ Distribuição de frequências da variável *dependente* (y)

<https://seaborn.pydata.org/generated/seaborn.distplot.html?highlight=distplot#seaborn.distplot>

```
ax = sns.distplot(dados.Valor)
ax.figure.set_size_inches(20, 6)
ax.set_title('Distribuição de Frequências', fontsize=20)
ax.set_xlabel('Preço dos Imóveis (R$)', fontsize=16)
ax
```



<ipython-input-10-c142b211f70c>:1: UserWarning:

``distplot`` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either ``displot`` (a figure-level function with similar flexibility) or ``histplot`` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
ax = sns.distplot(dados.Valor)
<Axes: title={'center': 'Distribuição de Frequências'}, xlabel='Preço dos Imóveis (R$)', ylabel='Density'>
```



## ✓ 2.3 Dispersão Entre as Variáveis

---

Gráficos de dispersão entre as variáveis do dataset

### ✓ seaborn.pairplot

<https://seaborn.pydata.org/generated/seaborn.pairplot.html?highlight=pairplot#seaborn.pairplot>

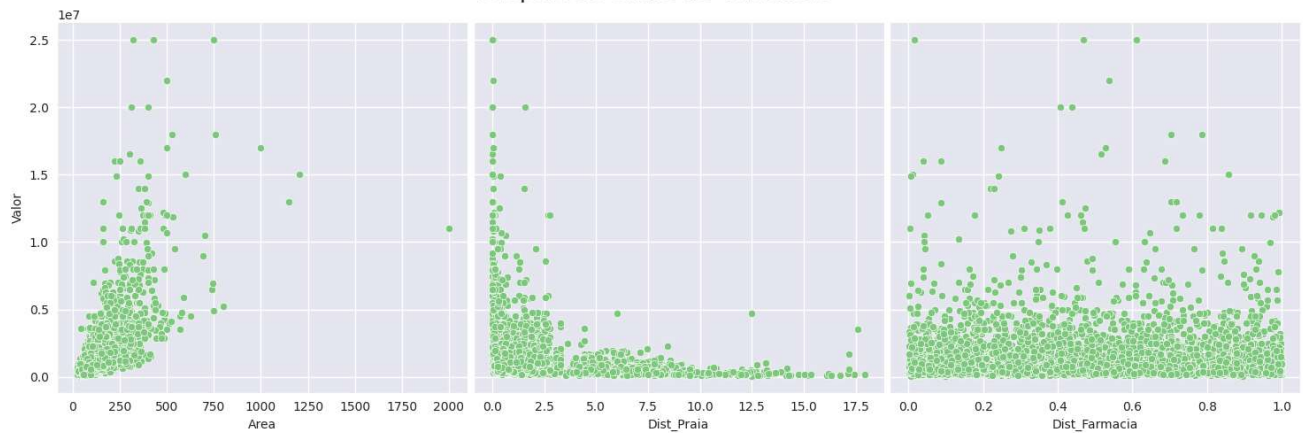
Plota o relacionamento entre pares de variáveis em um dataset.

```
ax = sns.pairplot(dados, y_vars = "Valor", x_vars = ["Area", "Dist_Praia", "Dist_Farmacia"]  
ax.fig.suptitle('Dispersão entre as Variáveis', fontsize=20, y=1.05)  
ax
```



<seaborn.axisgrid.PairGrid at 0x7d9c2850a890>

Dispersão entre as Variáveis

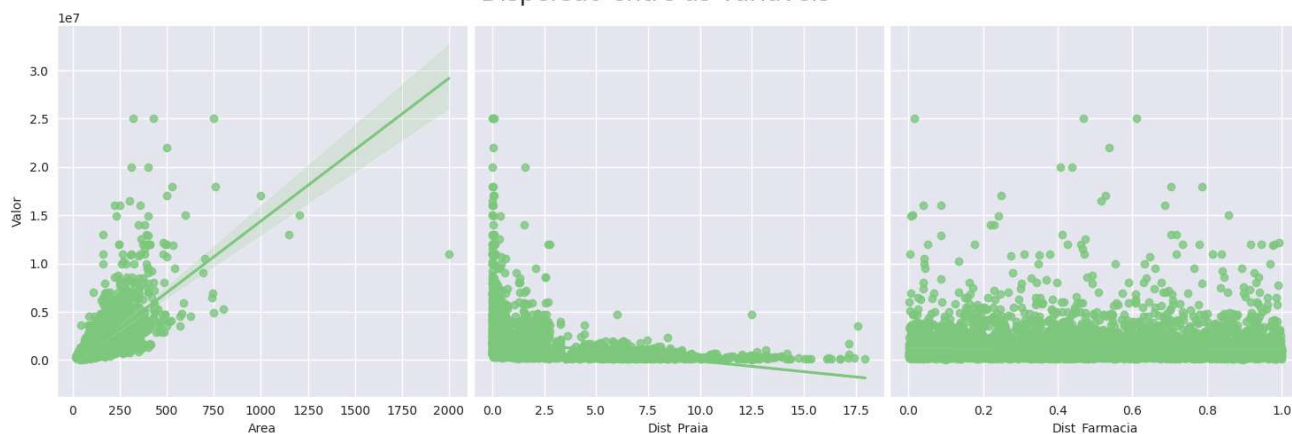


```
ax = sns.pairplot(dados, y_vars = "Valor", x_vars = ["Area", "Dist_Praia", "Dist_Farmacia"]  
ax.fig.suptitle('Dispersão entre as Variáveis', fontsize=20, y=1.05)  
ax
```



&lt;seaborn.axisgrid.PairGrid at 0x7d9be4cb0610&gt;

Dispersão entre as Variáveis



## ✓ 3.1 Transformando os Dados

### Distribuição Normal

#### Por quê?

Testes paramétricos assumem que os dados amostrais foram coletados de uma população com distribuição de probabilidade conhecida. Boa parte dos testes estatísticos assumem que os dados seguem uma distribuição normal (t de Student, intervalos de confiança etc.).

## ✓ Importando biblioteca numpy

```
import numpy as np
```



## ✓ Aplicando a transformação logarítmica aos dados do *dataset*

<https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.log.html>

```
np.log(0)
```

```
↳ <ipython-input-14-f6e7c0610b57>:1: RuntimeWarning: divide by zero encountered in log
  np.log(0)
-inf
```

```
dados["log_Valor"] = np.log(dados.Valor)
dados["log_Area"] = np.log(dados.Area)
dados["log_Dist_Praia"] = np.log(dados["Dist_Praia"] + 1)
dados["log_Dist_Farmacia"] = np.log(dados["Dist_Farmacia"] + 1)
```

```
dados.head()
```

```
↳
```

	Valor	Area	Dist_Praia	Dist_Farmacia	log_Valor	log_Area	log_Dist_Praia	log
0	4600000	280	0.240925	0.793637	15.341567	5.634790	0.215857	
1	900000	208	0.904136	0.134494	13.710150	5.337538	0.644028	
2	2550000	170	0.059525	0.423318	14.751604	5.135798	0.057821	
3	550000	100	2.883181	0.525064	13.217674	4.605170	1.356655	
4	2200000	164	0.239758	0.192374	14.603968	5.099866	0.214916	

Próximas etapas:

Gerar código com dados



Ver gráficos recomendados

New interactive sheet

## ✓ Distribuição de frequências da variável *dependente transformada* (y)

```
ax = sns.distplot(dados.log_Valor)
ax.figure.set_size_inches(12, 6)
ax.set_title('Distribuição de Frequências', fontsize=20)
ax.set_xlabel('log do Preço dos Imóveis', fontsize=16)
ax
```



<ipython-input-17-a6184c629882>:1: UserWarning:

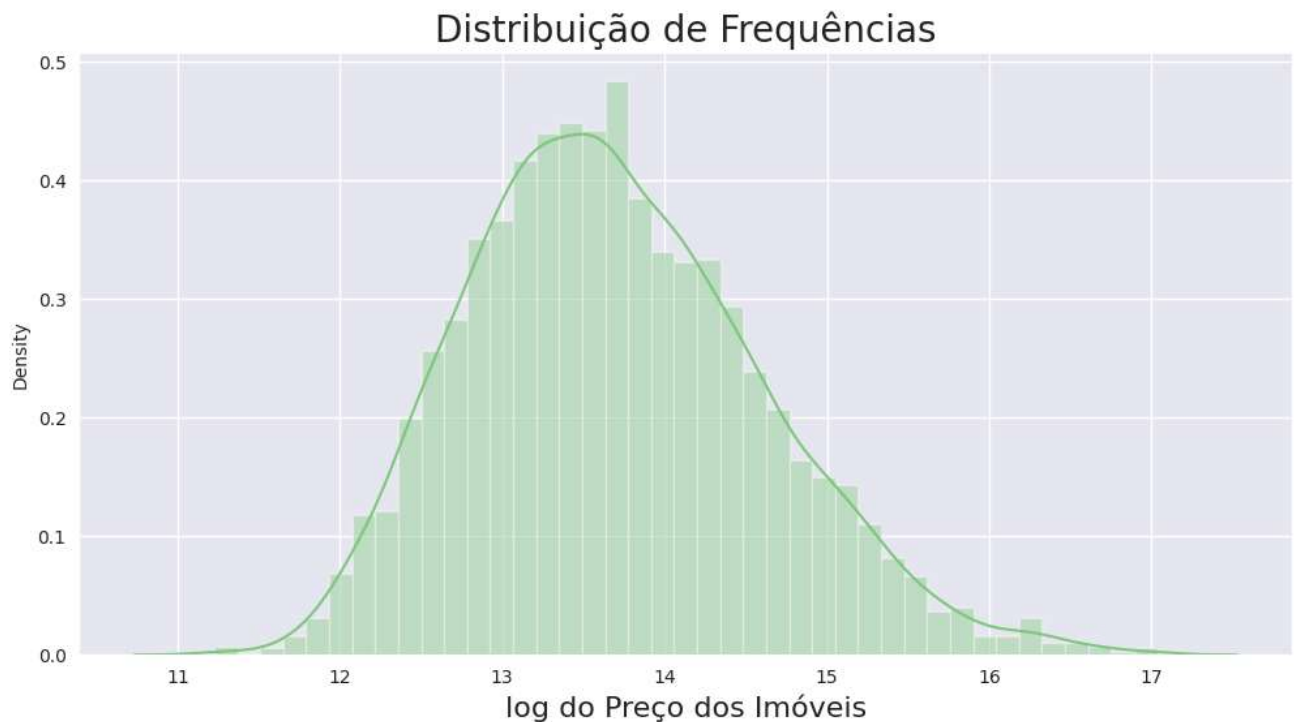
``distplot`` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either ``displot`` (a figure-level function with similar flexibility) or ``histplot`` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
ax = sns.distplot(dados.log_Valor)
<Axes: title={'center': 'Distribuição de Frequências'}, xlabel='log do Preço dos Imóveis', ylabel='Density'>
```



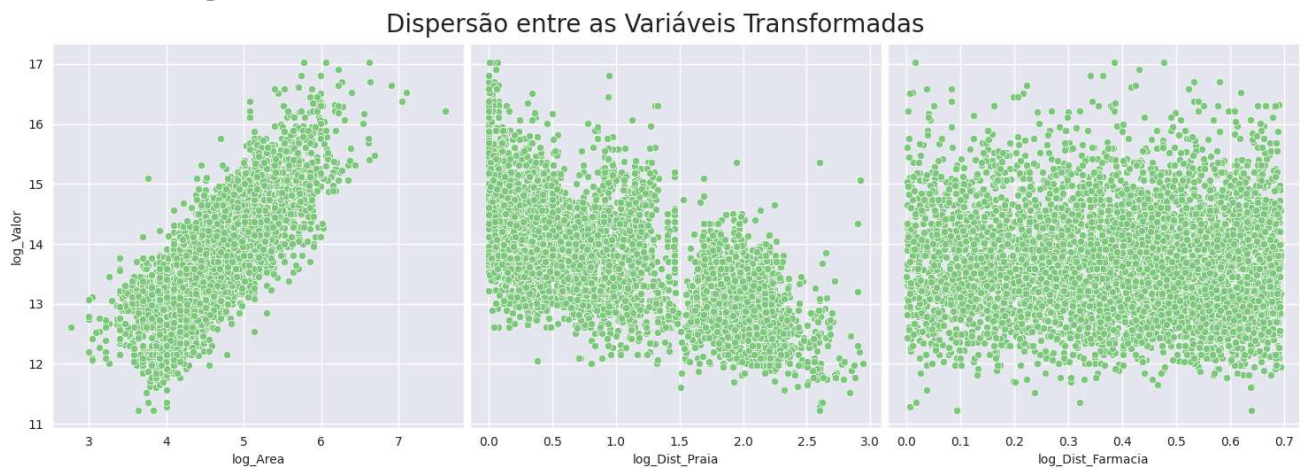
## ✓ 3.2 Verificando Relação Linear

### ✓ Gráficos de dispersão entre as variáveis transformadas do dataset

```
ax = sns.pairplot(dados, y_vars = "log_Valor", x_vars = ["log_Area", "log_Dist_Praia", "log_Dist_Farmacia"])
ax.fig.suptitle('Dispersão entre as Variáveis Transformadas', fontsize=20, y=1.05)
```



<seaborn.axisgrid.PairGrid at 0x7d9be1a064a0>



## ✓ 4.1 Criando os *Datasets* de Treino e Teste

### ✓ Importando o *train\_test\_split* da biblioteca *scikit-learn*

[https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

```
from sklearn.model_selection import train_test_split
```

### ✓ Criando uma Series (pandas) para armazenar o Preço dos Imóveis (y)

```
y = dados.log_Valor
```

- ✓ Criando um DataFrame (pandas) para armazenar as variáveis explicativas (X)

```
x = dados[["log_Area", "log_Dist_Praia", "log_Dist_Farmacia"]]
```

- ✓ Criando os datasets de treino e de teste

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state =
```

- ✓ Regressão Linear

---

A análise de regressão diz respeito ao estudo da dependência de uma variável (a variável **dependente**) em relação a uma ou mais variáveis, as variáveis explanatórias, visando estimar e/ou prever o valor médio da primeira em termos dos valores conhecidos ou fixados das segundas.

scikit-learn (<https://scikit-learn.org/stable/>)

- ✓ Importando a biblioteca statsmodels

<https://www.statsmodels.org/stable/index.html>

```
import statsmodels.api as sm
```

- ✓ Estimando o modelo com statsmodels

```
x_train_com_constate = sm.add_constant(x_train)
```

```
modelo_statsmodels = sm.OLS(y_train, x_train_com_constate, hasconst = True).fit()
```

- ✓ 4.2 Avaliando o Modelo Estimado

## ✓ Avaliando as estatísticas de teste do modelo

```
print(modelo_statsmodels.summary())
```



### OLS Regression Results

=====						
Dep. Variable:	log_Valor	R-squared:	0.805			
Model:	OLS	Adj. R-squared:	0.805			
Method:	Least Squares	F-statistic:	5495.			
Date:	Wed, 11 Dec 2024	Prob (F-statistic):	0.00			
Time:	01:15:27	Log-Likelihood:	-2044.9			
No. Observations:	4000	AIC:	4098.			
Df Residuals:	3996	BIC:	4123.			
Df Model:	3					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]
-----						
const	9.3417	0.060	154.734	0.000	9.223	9.460
log_Area	1.0580	0.012	89.320	0.000	1.035	1.081
log_Dist_Praia	-0.4905	0.009	-56.690	0.000	-0.508	-0.474
log_Dist_Farmacia	-0.0167	0.032	-0.521	0.603	-0.080	0.046
=====						
Omnibus:	64.751	Durbin-Watson:	1.971			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	106.858			
Skew:	0.136	Prob(JB):	6.25e-24			
Kurtosis:	3.753	Cond. No.	47.6			
=====						

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly spec



## ✓ 4.3 Modificando o Modelo e Avaliando Novamente o Ajuste

### ✓ Criando um novo conjunto de variáveis explicativas (X)

```
x = dados[["log_Area", "log_Dist_Praia"]]
```

### ✓ Criando os datasets de treino e de teste

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state =
```

## ✓ Estimando o modelo com o statsmodels

```
x_train_com_constate = x_train_com_constate = sm.add_constant(x_train)
modelo_statsmodels = sm.OLS(y_train, x_train_com_constate, hasconst = True).fit()
```

## ✓ Avaliando as estatísticas de teste do novo modelo

```
print(modelo_statsmodels.summary())
```



### OLS Regression Results

```
=====
Dep. Variable:          log_Valor    R-squared:                0.805
Model:                  OLS          Adj. R-squared:           0.805
Method:                 Least Squares  F-statistic:             8244.
Date:                  Wed, 11 Dec 2024  Prob (F-statistic):       0.00
Time:                  01:15:29        Log-Likelihood:          -2045.1
No. Observations:      4000          AIC:                    4096.
Df Residuals:          3997          BIC:                    4115.
Df Model:               2
Covariance Type:       nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
const	9.3349	0.059	158.353	0.000	9.219	9.450
log_Area	1.0581	0.012	89.345	0.000	1.035	1.081
log_Dist_Praia	-0.4906	0.009	-56.709	0.000	-0.508	-0.474

```
=====
Omnibus:                65.115    Durbin-Watson:           1.972
Prob(Omnibus):           0.000    Jarque-Bera (JB):        107.712
Skew:                   0.136    Prob(JB):                4.08e-24
Kurtosis:                3.757    Cond. No.                 46.1
=====
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly spec



## ✓ 5.1 Estimando o Modelo com os Dados de Treino

### ✓ Importando *LinearRegression* e *metrics* da biblioteca *scikit-learn*

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LinearRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)

<https://scikit-learn.org/stable/modules/classes.html#regression-metrics>

```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

## ✓ Instanciando a classe *LinearRegression()*

```
modelo = LinearRegression()
```

## ✓ Utilizando o método *fit()* do objeto "modelo" para estimar nosso modelo linear utilizando os dados de TREINO (y\_train e X\_train)

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LinearRegression.html#sklearn.linear\\_model.LinearRegression.fit](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression.fit)

```
modelo.fit(x_train, y_train)
```



```
▼ LinearRegression ⓘ ?
LinearRegression()
```

## ✓ Obtendo o coeficiente de determinação ( $R^2$ ) do modelo estimado com os dados de TREINO

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LinearRegression.html#sklearn.linear\\_model.LinearRegression.score](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression.score)

### Coeficiente de Determinação - $R^2$

O coeficiente de determinação ( $R^2$ ) é uma medida resumida que diz quanto a linha de regressão ajusta-se aos dados. É um valor entra 0 e 1.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1} (y_i - \bar{y}_i)^2}$$

```
print(f"R² = {modelo.score(x_train, y_train)}")
```



```
R² = 0.8048773977172844
```

- ✓ Gerando previsões para os dados de TESTE ( $X_{\text{test}}$ ) utilizando o método *predict()* do objeto "modelo"

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LinearRegression.html#sklearn.linear\\_model.LinearRegression.predict](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression.predict)

```
y_previsto = modelo.predict(x_test)
```

- ✓ Obtendo o coeficiente de determinação ( $R^2$ ) para as previsões do nosso modelo

[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2\\_score.html#sklearn.metrics.r2\\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html#sklearn.metrics.r2_score)

```
print(f"R² (previsões) = {metrics.r2_score(y_test, y_previsto)}")
```

```
➞ R² (previsões) = 0.7904430508603385
```

## ✓ 5.2 Obtendo Previsões Pontuais

---

### ✓ Dados de entrada

```
entrada = x_test[0:1]
entrada
```

```
➞
```

	log_Area	log_Dist_Praia
1006	5.273	1.282769

### ✓ Gerando previsão pontual

```
modelo.predict(entrada)[0]
```

```
➞ 14.284820061847878
```



## ✓ Invertendo a transformação para obter a estimativa em R\$

<https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.exp.html>

```
np.exp(modelo.predict(entrada)[0])
```

```
→ 1598889.7847794362
```

## ✓ Criando um simulador simples

```
Area = 250
Dist_Praia = 1
entrada = [[np.log(Area), np.log(Dist_Praia + 1)]]
print(f"O valor do imóvel é: R${np.exp(modelo.predict(entrada)[0])}")
```

```
→ O valor do imóvel é: R$2777292.8403123408
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not
  warnings.warn(
```

## ✓ 5.3 Interpretação dos Coeficientes Estimados

### ✓ Obtendo o intercepto do modelo

O **intercepto** representa o efeito médio em  $\hat{Y}$  (Preço do Imóveis) tendo todas as variáveis explicativas excluídas do modelo. No caso do modelo log-linear este coeficiente deve ser transformado com o uso da função exponencial para ser apresentado em R\$.

```
modelo.intercept_
```

```
→ 9.33491640980033
```

```
print(np.exp(modelo.intercept_))
```

```
→ 11326.681428069862
```

### ✓ Obtendo os coeficientes de regressão

Os **coeficientes de regressão**  $\beta_2$  e  $\beta_3$  são conhecidos como **coeficientes parciais de regressão** ou **coeficientes parciais angulares**.

Um aspecto interessante do modelo log-linear, que o tornou muito utilizado nos trabalhos aplicados, é que os coeficientes angulares  $\beta_2$  e  $\beta_3$  medem as elasticidades de  $Y$  em relação a  $X_2$  e  $X_3$ , isto é, a variação percentual de  $Y$  correspondente a uma dada variação percentual (pequena) em  $X_2$  e  $X_3$ .

```
modelo.coef_
```

```
array([ 1.05807818, -0.49061226])
```

## ✓ Confirmando a ordem das variáveis explicativas no DataFrame

```
x.columns
```

```
Index(['log_Area', 'log_Dist_Praia'], dtype='object')
```

## ✓ Criando uma lista com os nomes das variáveis do modelo

```
index = ["log da area", "log da area", "log da distância até a praia"]
```

## ✓ Criando um DataFrame para armazenar os coeficientes do modelo

<https://docs.scipy.org/doc/numpy/reference/generated/numpy.append.html?#numpy.append>

```
pd.DataFrame(data = np.append(modelo.intercept_, modelo.coef_), index = index, columns =
```

```
array([ 1.05807818, -0.49061226])
```

	Parâmetros
log da area	9.334916
log da area	1.058078
log da distância até a praia	-0.490612

## Interpretação dos Coeficientes Estimados

**Intercepto** → Excluindo o efeito das variáveis explicativas ( $X_2 = X_3 = 0$ ) o efeito médio no Preço dos Imóveis seria de **R\$ 11.326,68** ( $\exp[9.334916]$ ).

**Área (m<sup>2</sup>)** → Mantendo-se o valor de  $X_3$  (Distância até a Praia) constante, um acréscimo de 1% na Área de um imóvel gera, em média, um acréscimo de **1.06%** no Preço do Imóvel.

**Distância até a Praia (km)** → Mantendo-se o valor de  $X_2$  (Área) constante, um acréscimo de 1% na Distância de um imóvel até a praia gera, em média, um decréscimo de **0.49%** no Preço do Imóvel.

## ✓ 5.4 Análises Gráficas dos Resultados do Modelo

---

### ✓ Gerando as previsões do modelo para os dados de TREINO

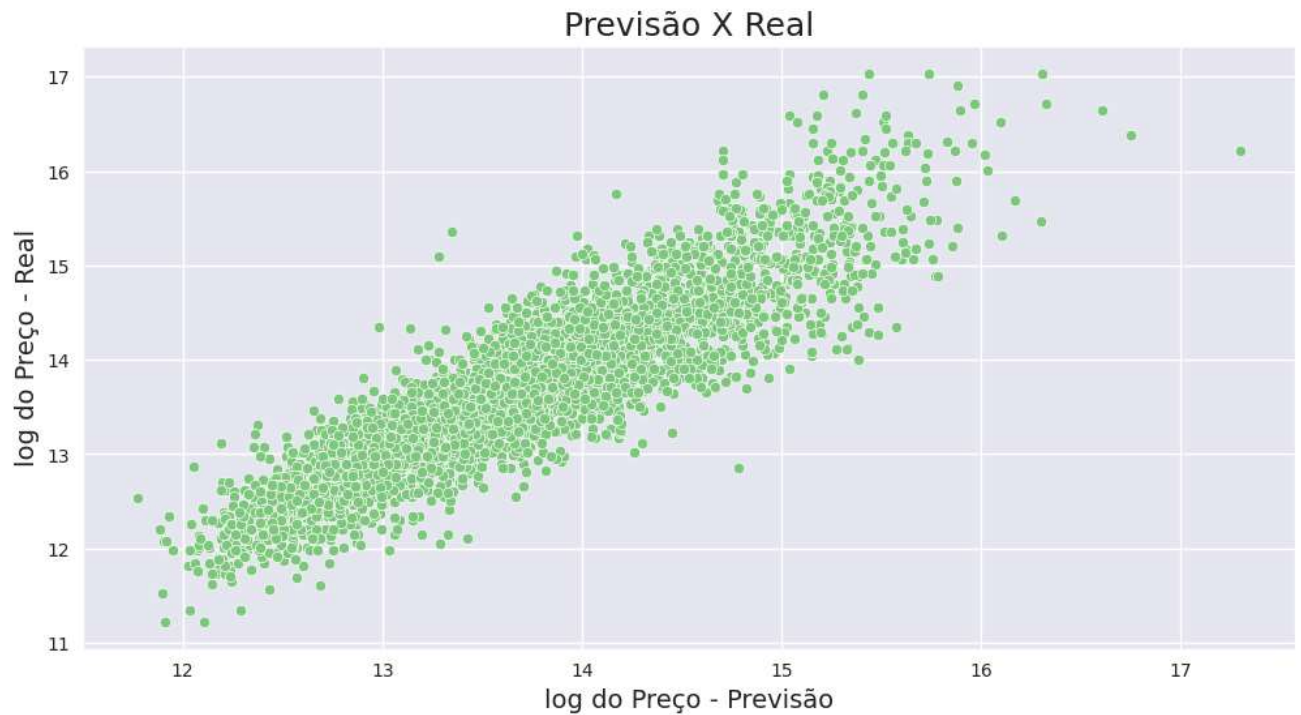
```
y_previsto_train = modelo.predict(x_train)
```

### ✓ Gráfico de dispersão entre valor estimado e valor real

<https://seaborn.pydata.org/generated/seaborn.scatterplot.html>

```
ax = sns.scatterplot(x = y_previsto_train, y = y_train)
ax.figure.set_size_inches(12, 6)
ax.set_title('Previsão X Real', fontsize=18)
ax.set_xlabel('log do Preço - Previsão', fontsize=14)
ax.set_ylabel('log do Preço - Real', fontsize=14)
ax
```

```
<Axes: title={'center': 'Previsão X Real'}, xlabel='log do Preço - Previsão',  
ylabel='log do Preço - Real'>
```



## ✓ Obtendo os resíduos

```
residuo = y_train - y_previsto_train
```

## ✓ Plotando a distribuição de frequências dos resíduos

```
ax = sns.distplot(residuo)  
ax.figure.set_size_inches(12, 6)  
ax.set_title('Distribuição de Frequências dos Resíduos', fontsize=18)  
ax.set_xlabel('log do Preço', fontsize=14)  
ax
```

 <ipython-input-50-9d07fa519467>:1: UserWarning:

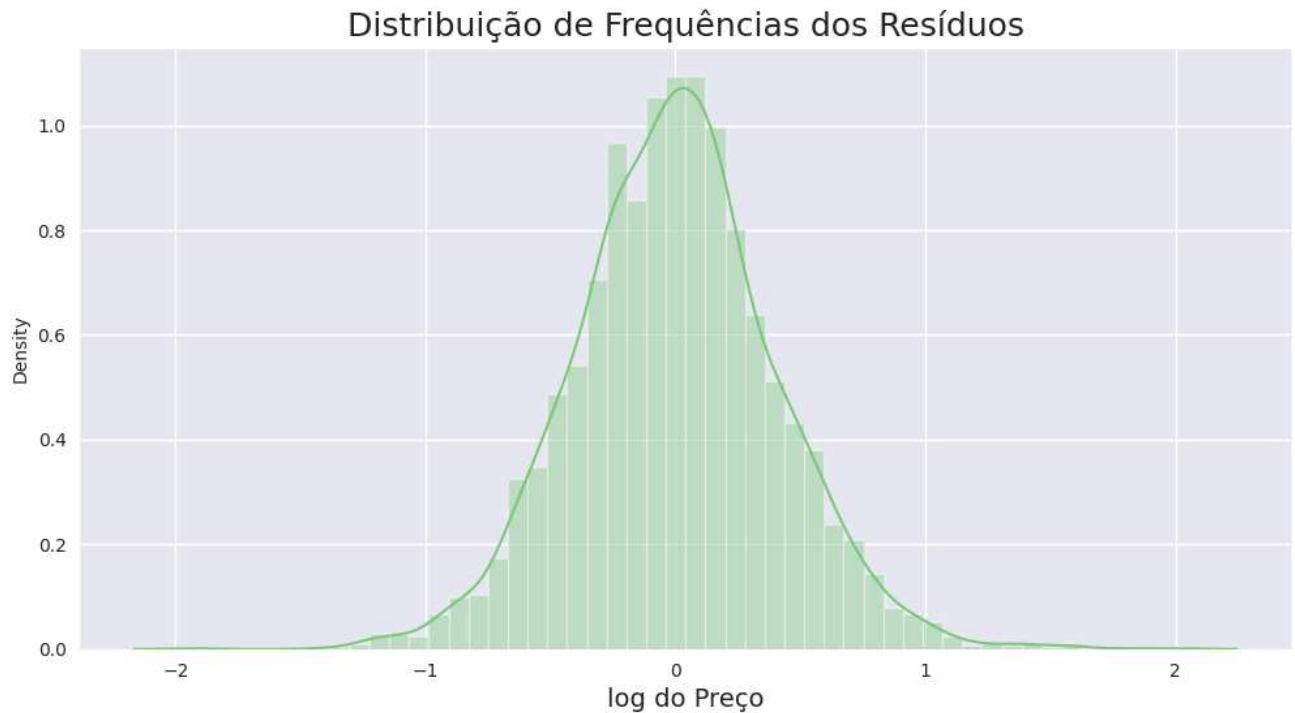
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
ax = sns.distplot(residuo)
<Axes: title={'center': 'Distribuição de Frequências dos Resíduos'}, xlabel='log do Preço', ylabel='Density'>
```



## ✓ Gráfico de dispresão entre valor estimado e resíduos

```
ax = sns.scatterplot(x = y_previsto_train, y = residuo, s = 150)
ax.figure.set_size_inches(20, 8)
ax.set_title("Resíduos X Previsão", fontsize = 18)
ax.set_xlabel("Consumo de cerveja - Previsão", fontsize = 14)
ax.set_ylabel("Resíduos", fontsize = 14)
ax
```

 <Axes: title={'center': 'Resíduos X Previsão'}, xlabel='Consumo de cerveja - Previsão', ylabel='Resíduos'>

