Inteligência Artificial

2024/2

Profa. Dra. Juliana Félix

jufelix16@gmail.com

PUC GOIÁS

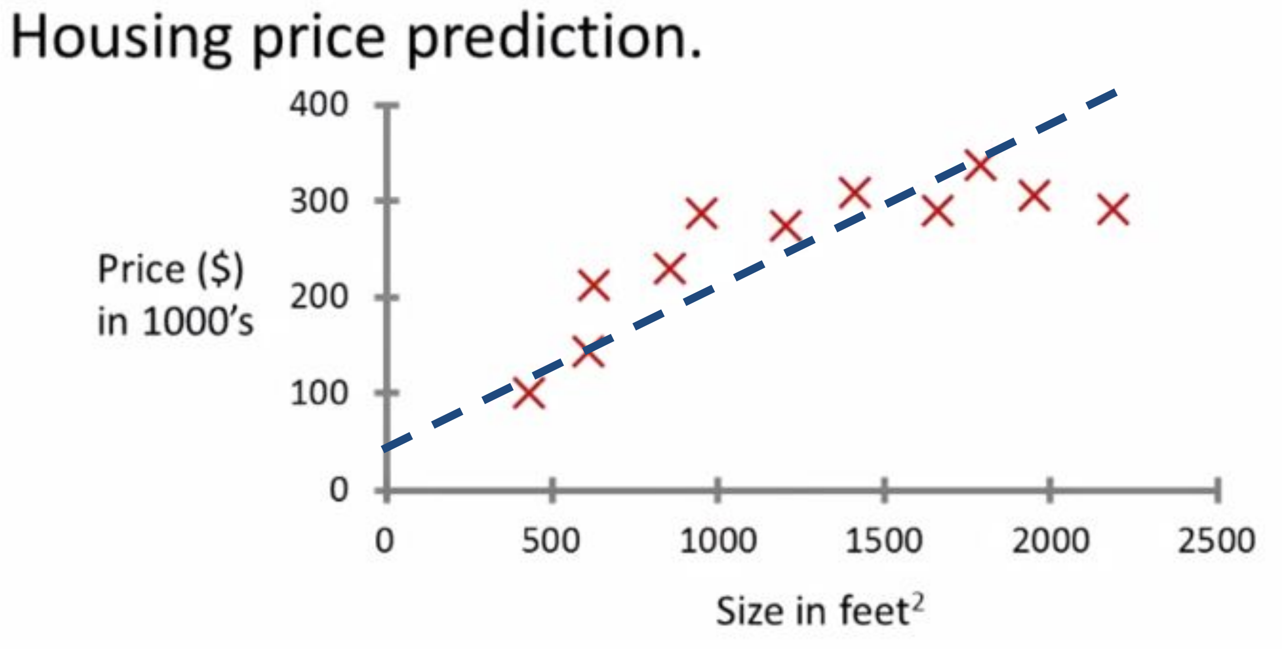
Regressão

Regressão

A **regressão linear** é uma técnica de análise de dados que permite prever o valor de dados desconhecidos usando valores de dados relacionados e conhecidos. Para isso modela-se, matematicamente, uma **equação linear** que relaciona:

- uma variável desconhecida, ou dependente, muitas vezes chamada de **variável de resultado**.
- e uma ou mais variáveis independentes, frequentemente chamados de **preditores**, covariáveis, recursos, ou **features**.

Regressão



Housing price prediction.
Price ($)
in 1000's
400
300
200
100
0
0
500
1000
1500
2000
2500
Size in feet²

Regressão

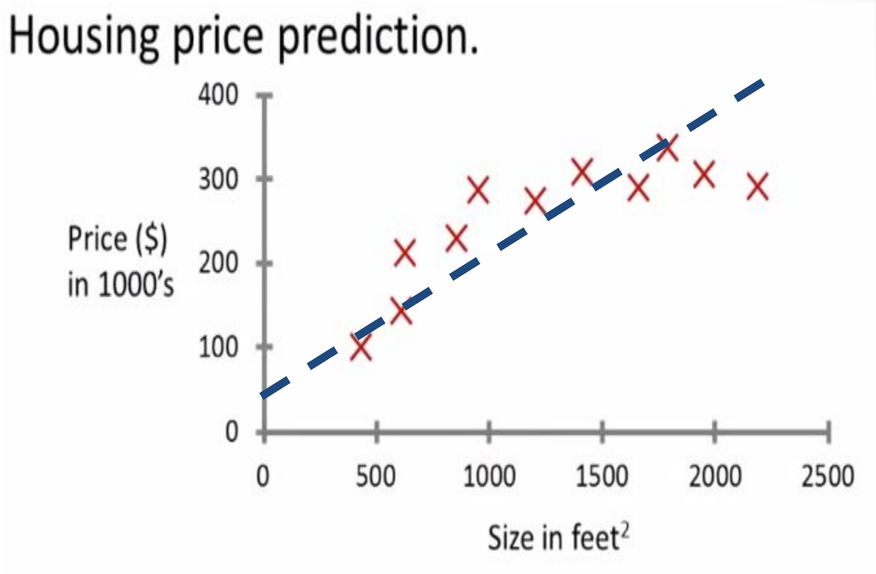
A regressão é um modelo baseado em aprendizado.

- Analizamos os dados.
- Supomos que eles seguem algum padrão.
- Utilizamos o padrão para predizer dados futuros.

Regressão

Equação da reta:

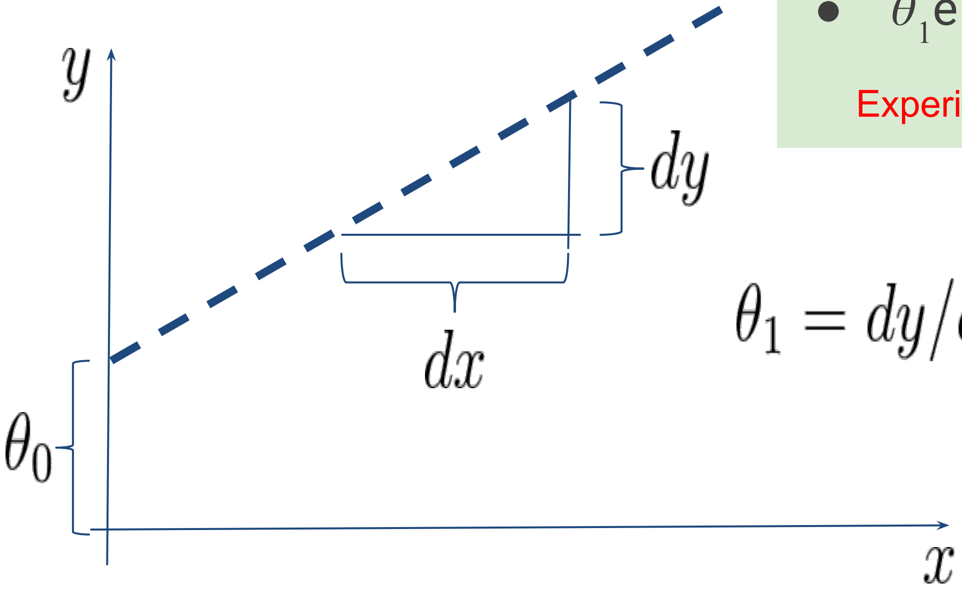
$$y = \theta_0 + \theta_1 x$$



Regressão

- θ_0 é o deslocamento no eixo y
- θ_1 é a inclinação da reta

Experimente algumas visualizações aqui.

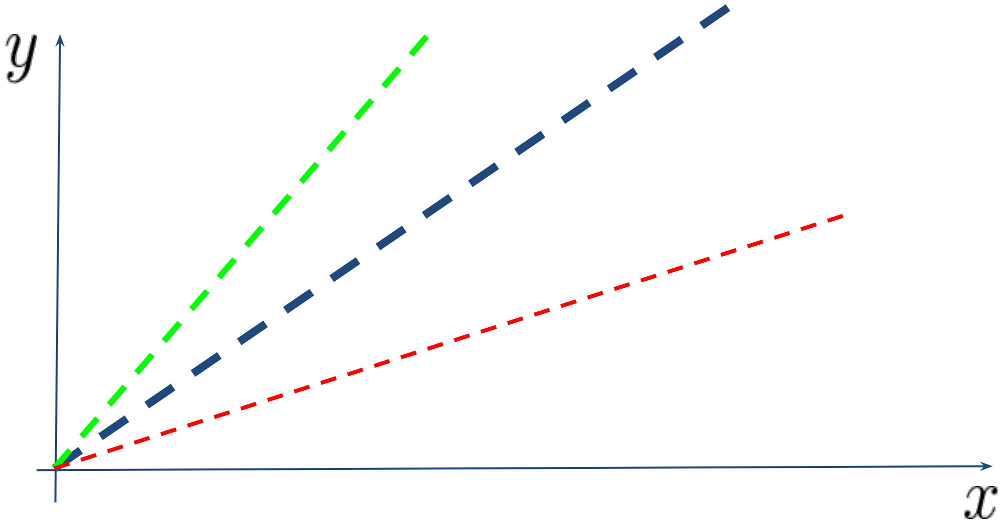


$$\theta_1 = dy/dx$$

$$y = \theta_0 + \theta_1 x$$

Regressão

- Se θ_0 é zero, a reta passa na origem



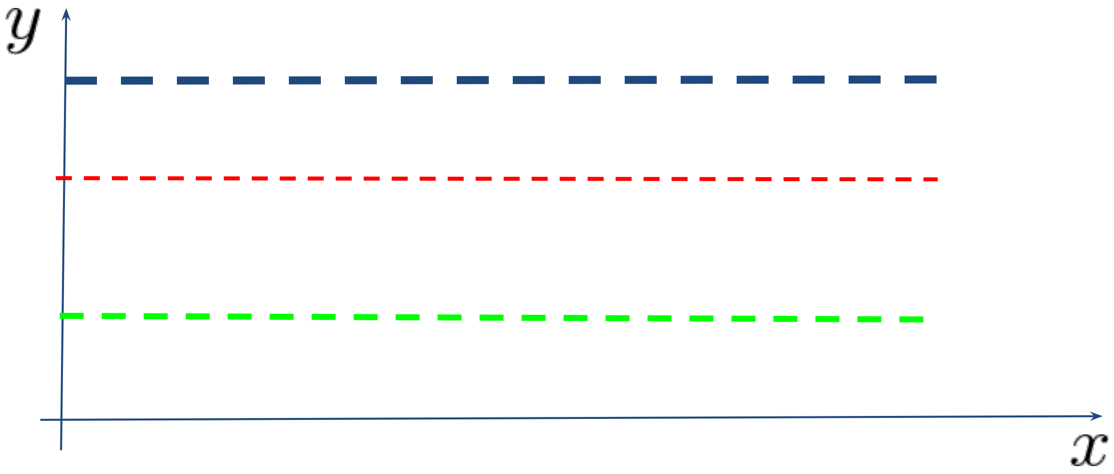
$$\theta_0 = 0 \qquad y = \theta_1 x$$

Regressão

```
import matplotlib.pyplot as plt
import numpy as np

#coeficiente angular da reta y = theta1*x + theta0 (ou y = mx + n, como muitos conhecem)
theta1 = 0.8

x = np.linspace(0, 1000, 1000)
#print(x)

y = theta1 * x + 0

plt.plot(x, y, '-r')
plt.xlim(0, 1000)
plt.ylim(0, 2000)
plt.xticks(np.arange(0,1100, step=100))

plt.xlabel('Área em $m^2$')
plt.ylabel('Preço em 1000\'s R$')
plt.title('Preço estimado do imóvel')
plt.grid()
plt.show()
```

Regressão

- Se θ_1 é zero, a reta será paralela ao eixo x



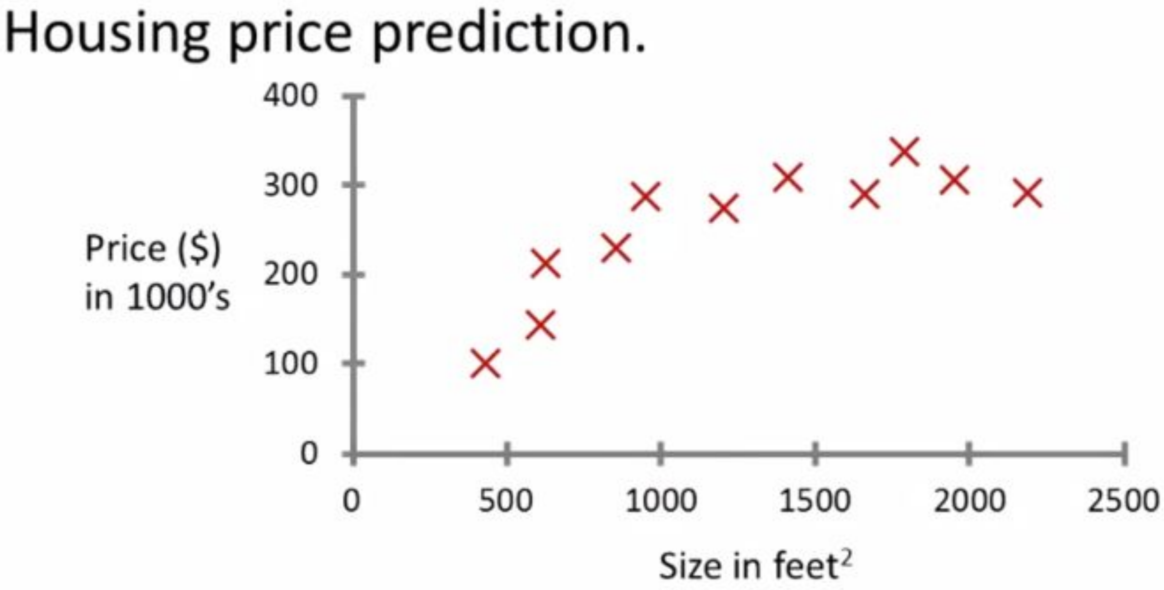
$\theta_1 = 0$ $\qquad$ $y = \theta_0$

Regressão

```
import matplotlib.pyplot as plt
import numpy as np

#parâmetro para uma reta sem inclinação y = theta1*x + theta0 (ou, y= mx + n)
theta0 = 700

x = np.linspace(0, 1000, 1000)
y = np.ones(1000) * theta0

plt.plot(x, y, '-r')
plt.xlim(0, 1000)
plt.ylim(0, 2000)
plt.xticks(np.arange(0,1100, step=100))
plt.xlabel('Área em $m^2$')
plt.ylabel('Preço em 1000\'s R$')
plt.title('Preço estimado do imóvel')
plt.grid()
plt.show()
```

Regressão

Voltando para o exemplo original... queremos estimar uma reta que melhor se ajuste aos pontos de observação.



Regressão

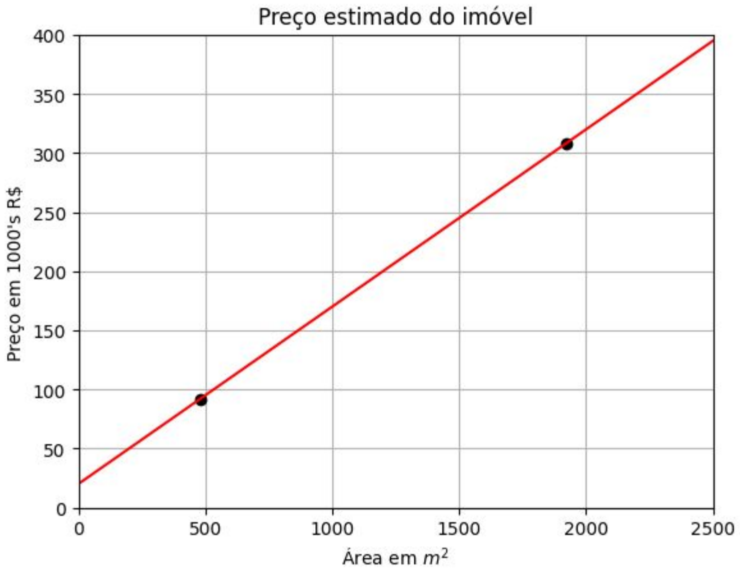
Se a correlação linear é forte....

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([480, 510, 520, 850, 960, 1200, 1400, 1650, 1700, 1920, 2350])
y = np.array([ 92, 96.5, 98, 147.5, 164, 200, 230, 267.5, 275, 308, 372.5])

plt.plot(x, y, 'o', color='black'); #plota os pontos no gráfico

theta1 = 0.15 # inclinacao da reta
theta0 = 20   # deslocamento no eixo y

x_entrada = np.linspace(0, 2500, 2500)
y_predito = theta1 * x_entrada + theta0

plt.plot(x_entrada, y_predito, '-r')
plt.xlim(0,2500)
plt.ylim(0,400)

plt.xlabel('Área em $m^2$')
plt.ylabel('Preço em 1000\'s R$')
plt.title('Preço estimado do imóvel')
plt.grid()
plt.show()
```

Regressão

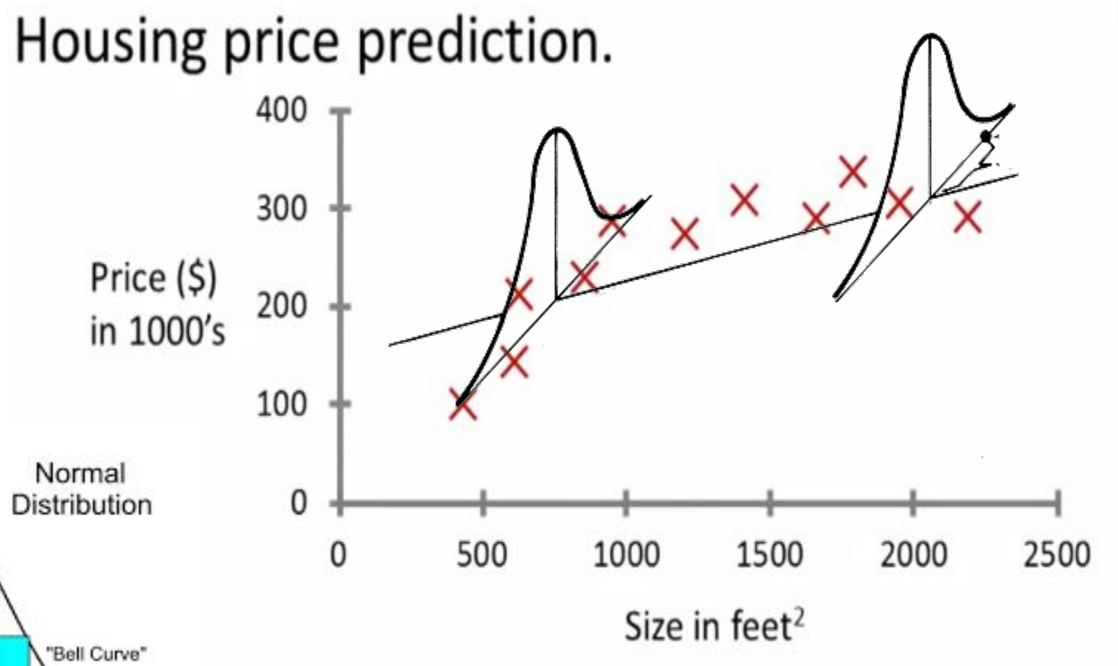
Se a correlação linear é forte…
Apenas 2 pontos quaisquer são suficientes para se encontrar os valores que definem uma reta

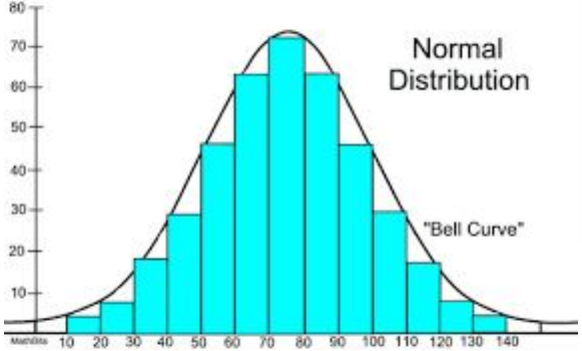
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([480, 1920])
y = np.array([ 92, 308])

plt.plot(x, y, 'o', color='black'); #plota os pontos no gráfico

# y = theta0 + theta1*x)
theta1 = (y[1] - y[0])/(x[1]-x[0]) # inclinacao da reta
theta0 = y[1] - theta1 * x[1]    # deslocamento no eixo y

x_entrada = np.linspace(0, 2500, 2500)
y_predito = theta1 * x_entrada + theta0

plt.plot(x_entrada, y_predito, '-r')
plt.xlim(0,2500)
plt.ylim(0,400)

plt.xlabel('Área em $m^2$')
plt.ylabel('Preço em 1000\'s R$')
plt.title('Preço estimado do imóvel')
plt.grid()
plt.show()
```



Regressão

Mas na vida real...





Exercício 1

Considerando os valores x e y fornecidos abaixo, tente encontrar uma reta que melhor se ajuste aos dados:

```
x = np.array([480, 510, 520, 850, 960, 1200, 1400, 1650, 1700, 1920, 2350])
y = np.array([98, 110, 200, 210, 280, 265, 300, 287, 325, 300, 290])
```

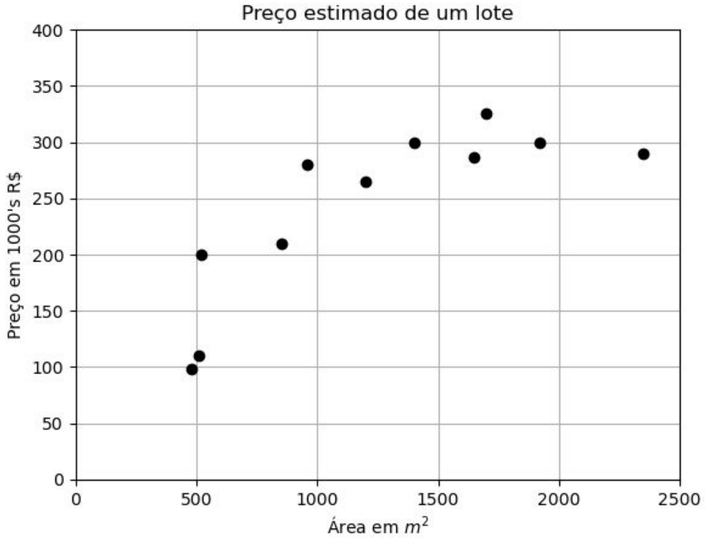
Faça isso utilizando:

a) Apenas θ_0
b) Apenas θ_1
c) Atribuindo valores para θ_0 e θ_1

Exercício 1

Tente ajustar, manualmente, uma reta que se ajuste aos dados fornecidos!

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([480, 510, 520, 850, 960, 1200, 1400, 1650, 1700, 1920, 2350])
y = np.array([98, 110, 200, 210, 280, 265, 300, 287, 325, 300, 290])

plt.plot(x, y, 'o', color='black');

plt.xlim(0, 2500)
plt.ylim(0, 400)

plt.xlabel('Área em $m^2$')
plt.ylabel('Preço em 1000\'s R$')
plt.title('Preço estimado de um lote')
plt.grid()
plt.show()
```



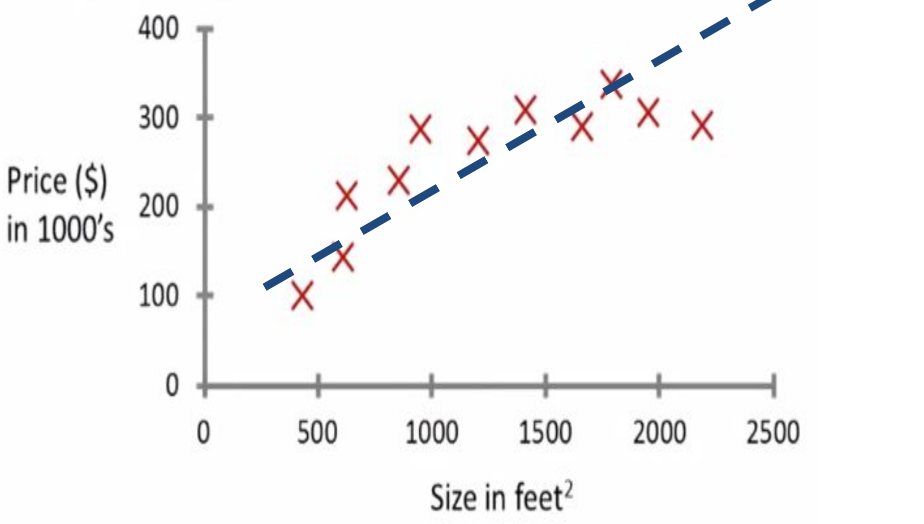
Regressão Linear

Regressão Linear

Modelo

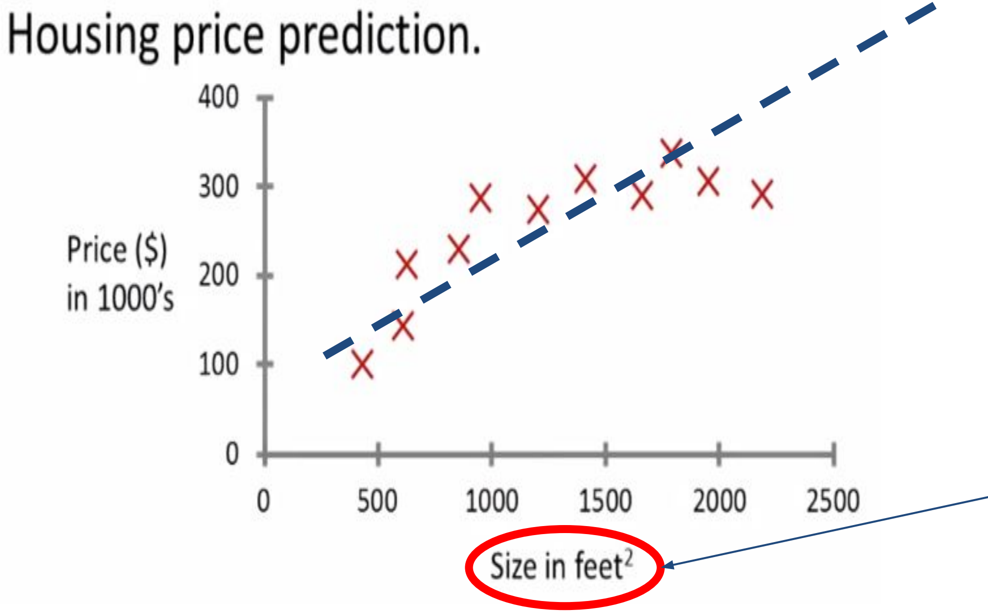
$$\boxed{y = \theta_0 + \theta_1 x}$$

Housing price prediction.



Regressão Linear

$$y = \theta_0 + \theta_1 x$$

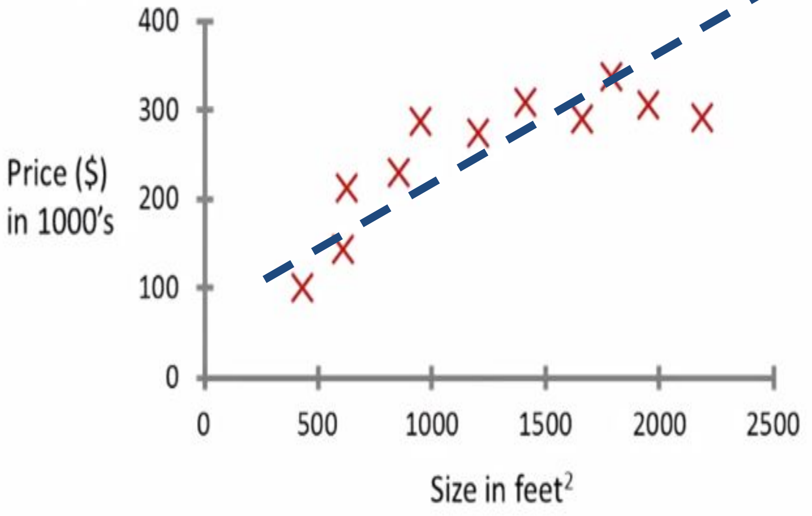


Características (*features*)

Regressão Linear

$$y = \theta_0 + \theta_1 x$$

Housing price prediction.

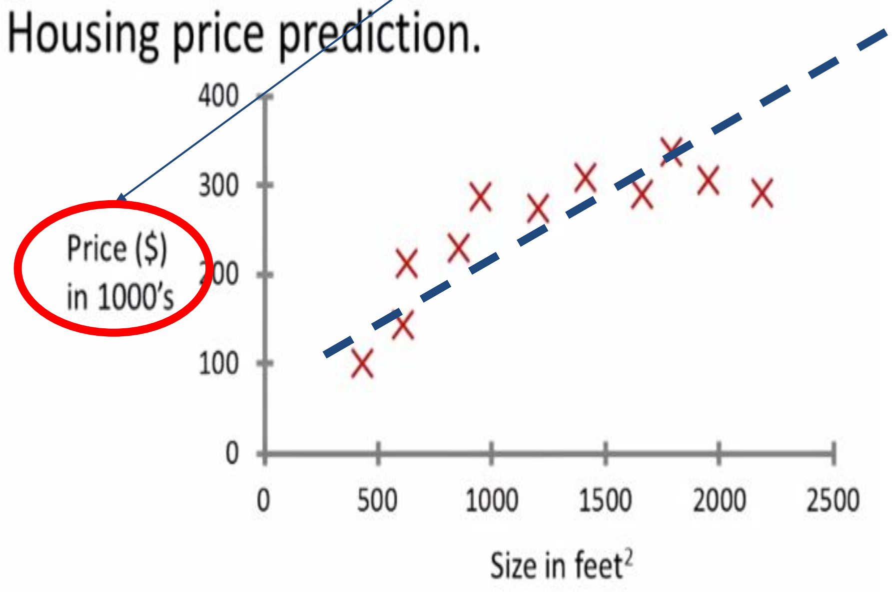


Parâmetros

Regressão Linear

Saída (output)

$$y = \theta_0 + \theta_1 x$$



Exemplo

Conjunto de dados (*Dataset*)

Dataset é um conjunto de dados que combina amostras com

- Valores ou variáveis de **entrada** (features, características) e
- Valores de **saída** (outcome, labels) utilizados no aprendizado supervisionado.

Dataset

Exemplo:

Preço de imóveis na 'Terra tão tão distante'	
Tamanho do imóvel (em m^2) - X	Preço (R$) - Y
2104	399.900
1600	329.900
2400	369.000
...	...

Total amostras = 47

Feature (característica)

Outcome (saída)

Notação

Podemos pensar no problema anterior como um problema que tem:

- Um total de $\mathbf{m}$ amostras/*samples* ($\mathbf{m}$ = 47)
- Cada amostra tem 1 única *feature*/característica (tamanho do imóvel)
  - Costumamos representar uma variável de entrada por $\mathbf{x}$
- Para cada amostra, temos uma única saída (preço do imóvel).
  - Costumamos representar uma variável de saída por $\mathbf{y}$
- Cada amostra pode ser representada por um par, ou tupla $\mathbf{(x,y)}$
  - Uma tupla $\mathbf{(x^i, y^i)}$ representa a i-ésima amostra do problema, com $\mathbf{1 \leq i \leq m}$

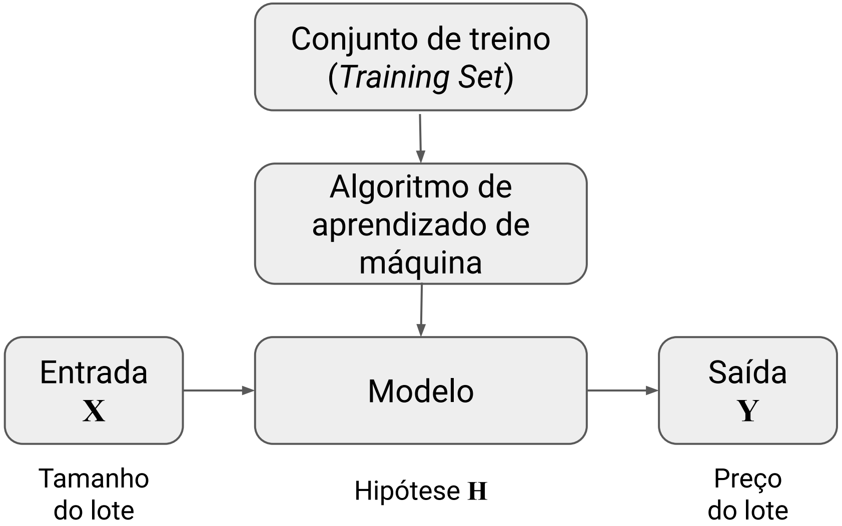
Notação

- O par (x^1, y^1) refere-se aos dados $(2104, 399900)$.
- No entanto, se a linguagem de programação que estiver utilizando considerar índices iniciados em zero (caso do Python), o par (x^1, y^1) pode referir-se, na verdade, aos dados $(1600, 329900)$.

Preço de lotes na 'Terra tão tão distante'	
Tamanho do lote (em m^2) - X	**Preço do lote (R$) - Y**
2104	399.900
1600	329.900
2400	369.000
...	...

Processo básico de Machine Learning

A base de qualquer processo de machine learning consiste em mapear um dado de entrada $\mathbf{X}$ em um dado de saída $\mathbf{Y}$.



$$h(x) = \theta_0 + \theta_1 x$$

PUC GOIÁS

Desvio

Quando fazemos a predição de um valor, o **desvio** é a diferença entre o **valor esperado** (conhecido) e o **valor predito** pelo modelo construído.

$$\text{desvio}^i = \mathbf{Y}^i - \mathbf{h}(\mathbf{x}^i)$$

$$\text{desvio}^i = \mathbf{Y}^i - \hat{\mathbf{Y}}^i$$

MSE

O ***Mean Square Error*** (MSE - Erro Médio Quadrático) é a **média** do **quadrado** dos **erros** obtidos pelo modelo.

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(Y^i - h(x^i)\right)^2$$

MSE

O ***Mean Square Error*** (*MSE* - Erro Médio Quadrático) é a **média** do **quadrado** dos **erros** obtidos pelo modelo.

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} \left(Y^i - h(x^i)\right)^2$$

Na prática, a média é dividida pela metade (1/2) como uma conveniência para o cálculo do ***gradiente descendente***, método utilizado na regressão linear, que cancelará o termo 1/2.

Em outras palavras, dividir por 1/m ou 1/2m não traz diferenças significativas para o cálculo dos valores analisados.

Leia mais aqui: https://math.stackexchange.com/questions/884887/why-divide-by-2m

MSE

$$MSE = \frac{1}{2m}\sum_{i=1}^{m}\left(Y^i - h(x^i)\right)^2$$

Considerando os seguintes valores preditos, podemos calcular o *MSE* do modelo.

Preço de lotes na 'Terra tão tão distante'				
x	**y**	**h(x)**	**desvio**	**desvio^2**
2104	399.900	399.800	100	10.000
1600	329.900	339.900	-10.000	10.000.000
2400	369.000	367.000	2.000	4.000.000
			Soma	14.010.000
			MSE	**2.335.000**

PUC GOIÁS

Exercício 2

Considerando os valores x e y fornecidos, tente encontrar, manualmente, uma reta que melhor se ajuste aos dados abaixo:

```
x = np.array([480, 510, 520, 850, 960, 1200, 1400, 1650, 1700, 1920, 2350])
y = np.array([98, 110, 200, 210, 280, 265, 300, 287, 325, 300, 290])
```

Faça isso utilizando:

a) Apenas θ_0
b) Apenas θ_1
c) Atribuindo valores para θ_0 e θ_1

Para cada reta, calcule o respectivo MSE. Plote as retas e o MSE encontrado em todos os casos.

PUC GOIÁS

Leitura recomendada

Calculadora gráfica: [Desmos | Calculadora Gráfica]

Regressão linear: [Explicação sobre o modelo de regressão linear]