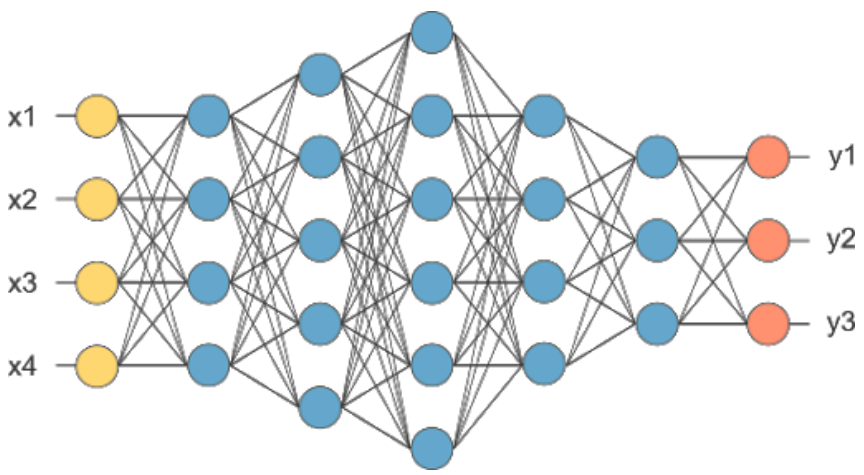
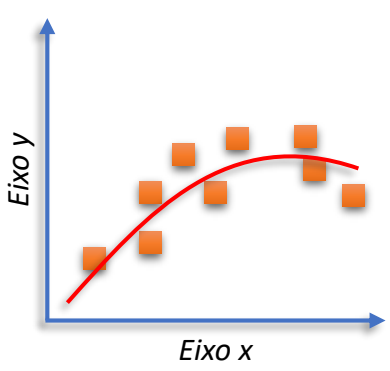
@Odemir Depieri Jr

# Um guia sobre Regressão Linear Polinomial com Redes Neurais Artificiais



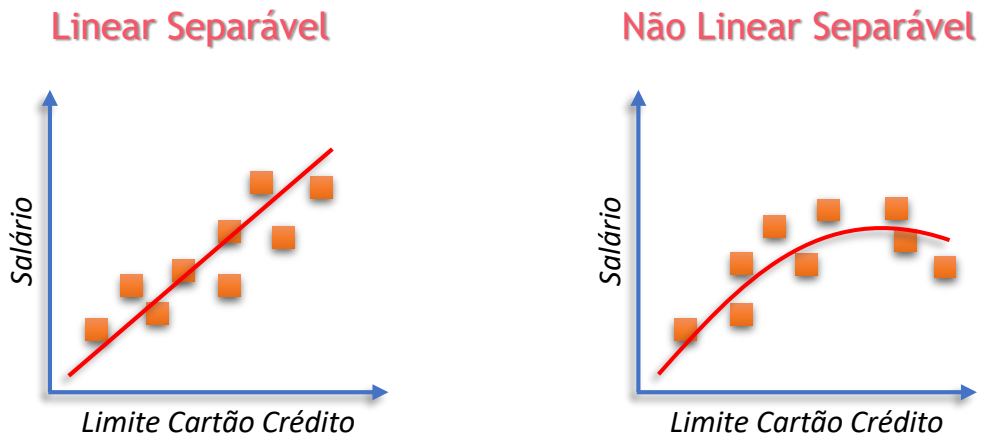

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_p x^p + \epsilon$$

```
[49] print('Ola Mundão!')

Ola Mundão!
```

# O que é Regressão Polinomial

## Regressão Polinomial

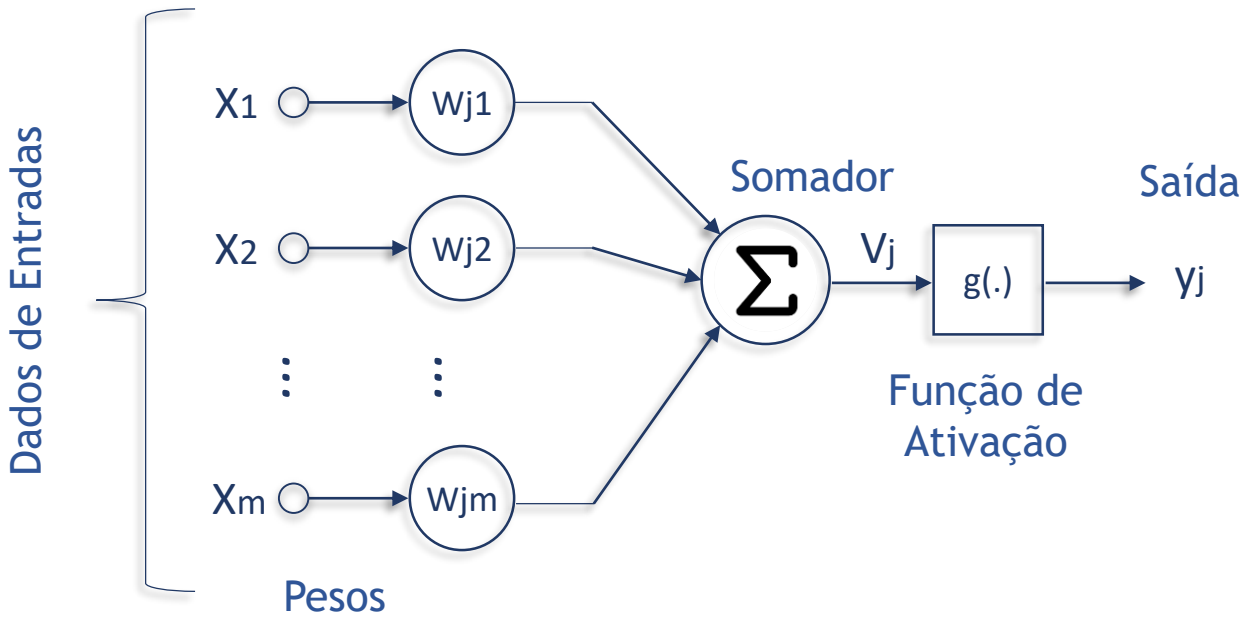
Os modelos de **regressão linear** ou **logística** são ótimos para aprender **fenômenos simples**, nos quais as variáveis interagem de forma linear.
As **regressões polinomial** são utilizadas para problemas **que não são** linearmente separáveis.
*Vamos ilustrar graficamente ...*




# O que são Redes Neurais

## Redes Neurais

Resumidamente **Redes Neurais Artificiais** são técnicas computacionais que apresentam um **modelo matemático** inspirado na **estrutura neural** de organismos inteligentes e que adquirem conhecimento através da experiência.
*Vamos ilustrar a estrutura da rede neural artificial ...*



# Referências para Estudo

## Modelos de Regressão

http://rstudio-pubs-static.s3.amazonaws.com/385563_aff45a3836e149669652a12dc102a64b.html

## Regressão Polinomial – Modelo Teórico

https://www.youtube.com/watch?v=YBHfIVsRZCE&list=PLgmg2_hBFC-2FbJF3MBjNMaK1lLMhnJnm

## Regressão Polinomial – Modelo Teórico

https://www.youtube.com/watch?v=2RgPsGuDD6I

## Redes Neurais – Modelo Teórico

https://medium.com/brasil-ai/entendendo-o-funcionamento-de-uma-rede-neural-artificial-4463fcf44dd0

## Redes Neurais – Modelo Teórico

https://www.youtube.com/watch?v=FCRStdk9hRg

## Redes Neurais – Tutorial em Python

https://www.youtube.com/watch?v=NFZwmiKTFfI

## Redes Neurais – **camadas escondidas e quantos neurônios incluir numa rede neural**

https://iaexpert.academy/2020/05/04/quantas-camadas-escondidas-e-quantos-neuronios-incluir-numa-rede-neural-artificial/

## Redes Neurais – **Descida do Gradiente**

https://www.deeplearningbook.com.br/aprendizado-com-a-descida-do-gradiente/

# Mão na Massa

## Vamos prever o valor de um apartamento nesse guia.

### Vamos importar as bibliotecas necessárias

```
[110] # Biblioteca para modelagem de dados
      import pandas as pd

      # Biblioteca para recursos matemáticos
      import numpy as np

      # Biblioteca para recursos Graficos
      import matplotlib.pyplot as plt
      import seaborn as sns
```

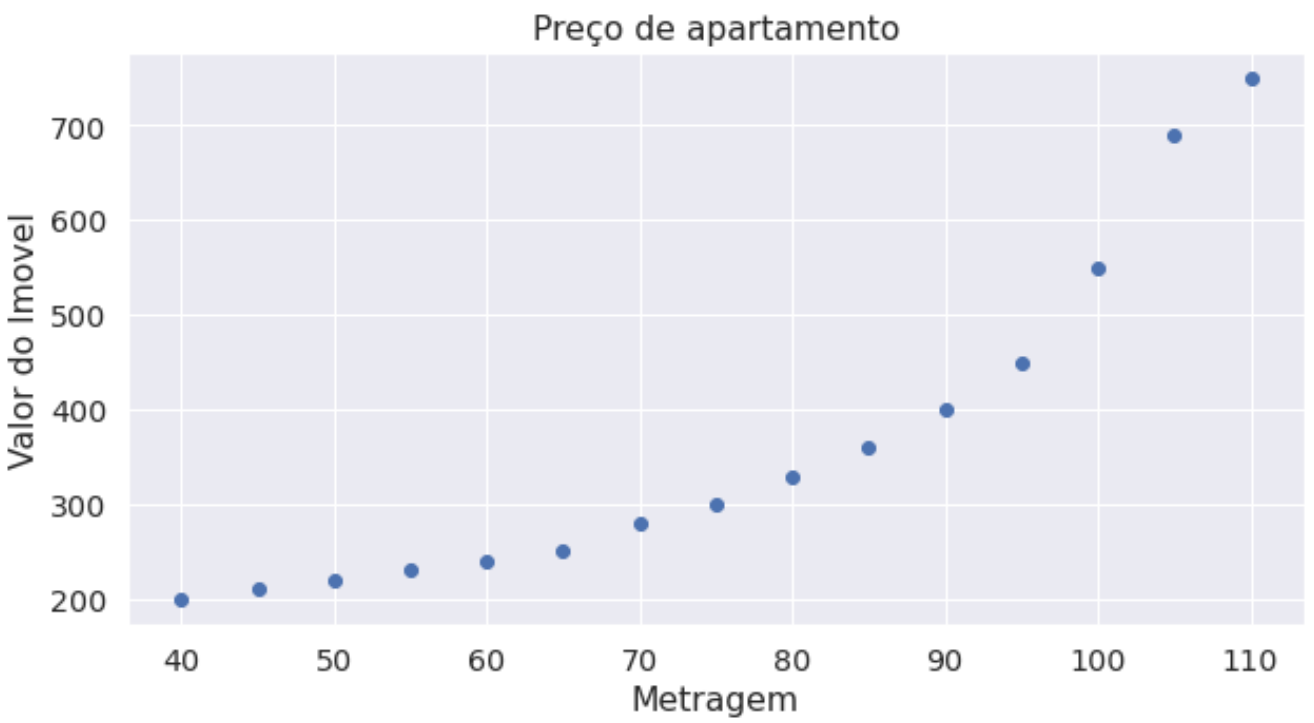
### Vamos gerar alguns números para podemos criar uma base de dados fictícia

```
[111] # Criando nossa base de dados
      # ---------------- Base de Preço de apartamentos Fictícia --------

      # Criando lista com os valores
      Metragem = [40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90,
                  95, 100, 105, 110]
      Valor = [200, 210, 220, 230, 240, 250, 280, 300, 330, 360,
               400, 450, 550, 690, 750]

      # Organizando os valores em um Dicionário
      Dicionario = {
          'Metragem' : Metragem,
          'Valor Imovel' : Valor
      }

      # Lendo o Dicionário com o Pandas
      DataFrame = pd.DataFrame( data=Dicionario )

      # Verificando as primeiras linhas
      DataFrame.head()
```

| | Metragem | Valor Imovel |
|---|---|---|
| **0** | 40 | 200 |
| **1** | 45 | 210 |
| **2** | 50 | 220 |
| **3** | 55 | 230 |
| **4** | 60 | 240 |

## Vamos gerar um gráfico para entender como ficou nossos dados

```
[112]  # -------------- Plotagem dos dados --------------

       # Ajustando o tamanho do Gráfico
       plt.figure( figsize=(10,5) )
       # Passando os valores para o grafico
       plt.scatter( DataFrame['Metragem'].values,
                    DataFrame['Valor Imovel'].values )
       # Definindo um titulo
       plt.title('Preço de apartamento')
       # Definindo o nome do eixo x
       plt.xlabel('Metragem')
       # Definindo o nome do eixo y
       plt.ylabel('Valor do Imovel');
```

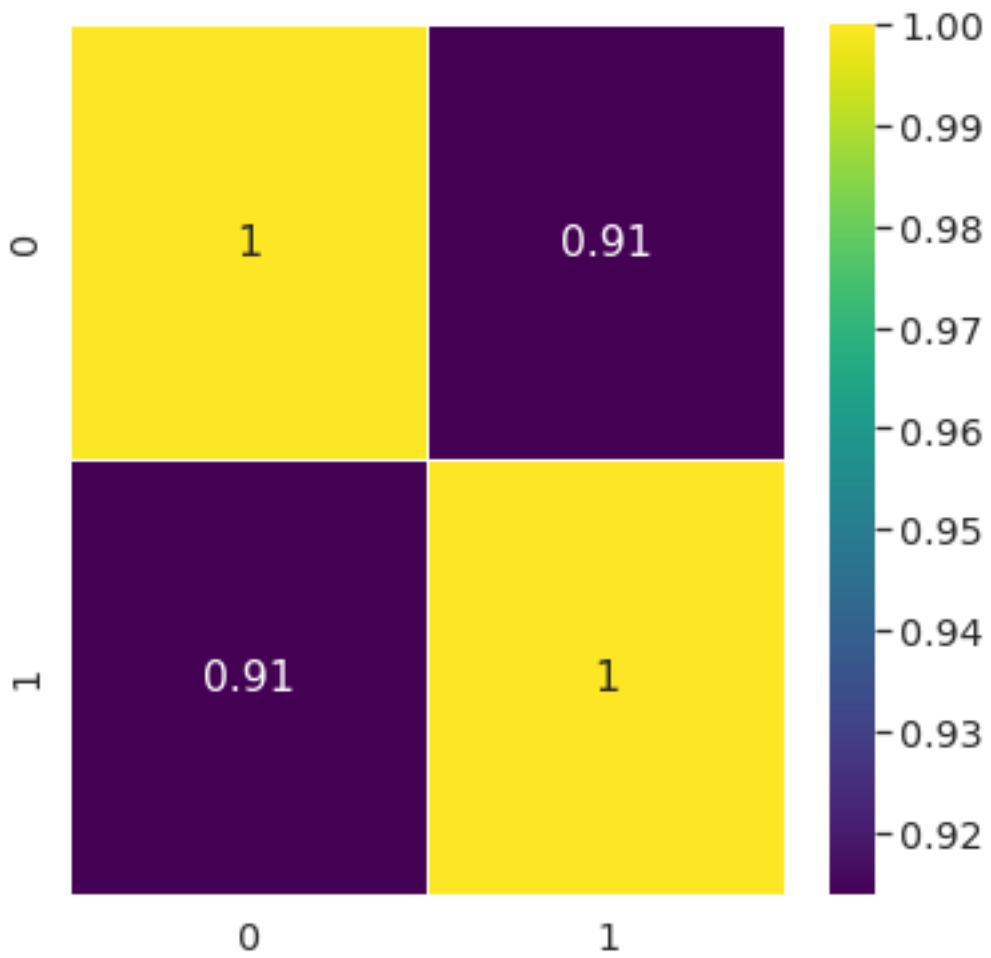



## Agora vamos separar os dados em eixos x e y
## Depois calculamos a correlação das variáveis.

```
[113] # Separando os dados no eixo x e y
      Eixo_x = DataFrame.iloc[:,0].values
      Eixo_y = DataFrame.iloc[:,1].values

      # Calculando a correlação entre os dados usando o Numpy
      Correlacao = np.corrcoef( Eixo_x, Eixo_y )
      Correlacao

      array([[1.        , 0.91379598],
             [0.91379598, 1.        ]])
```

## Vamos fazer um gráfico para ilustrar melhor a correlação

```
[114] # --- Analisando as correlações em uma plotagem

      # Definindo Tamanho do Gráfico
      plt.figure(figsize=(6,6))
      # Fazendo o plot do gráfico
      sns.heatmap(Correlacao, linewidths=.1, cmap='viridis', annot=True);
```




## Vamos converter os dados para um formato de Matriz

```
[115] # Convertendo os Dados para formato de Matriz
      # -1 quer dizer para não mexer nas linhas,
      # 1 quer dizer para incluir uma coluna

      Eixo_x = Eixo_x.reshape(-1, 1)
      Eixo_y = Eixo_y.reshape(-1, 1)
```

## Agora, vamos transformar os dados em escalas próximas

```
[116] # Função para fazer o escolanemento do dados
      from sklearn.preprocessing import StandardScaler

      # Definindo a função para cada eixo
      Escala_x = StandardScaler()
      Escala_y = StandardScaler()

      # Fazendo o escalonamento
      x = Escala_x.fit_transform( Eixo_x )
      y = Escala_y.fit_transform( Eixo_y )
```

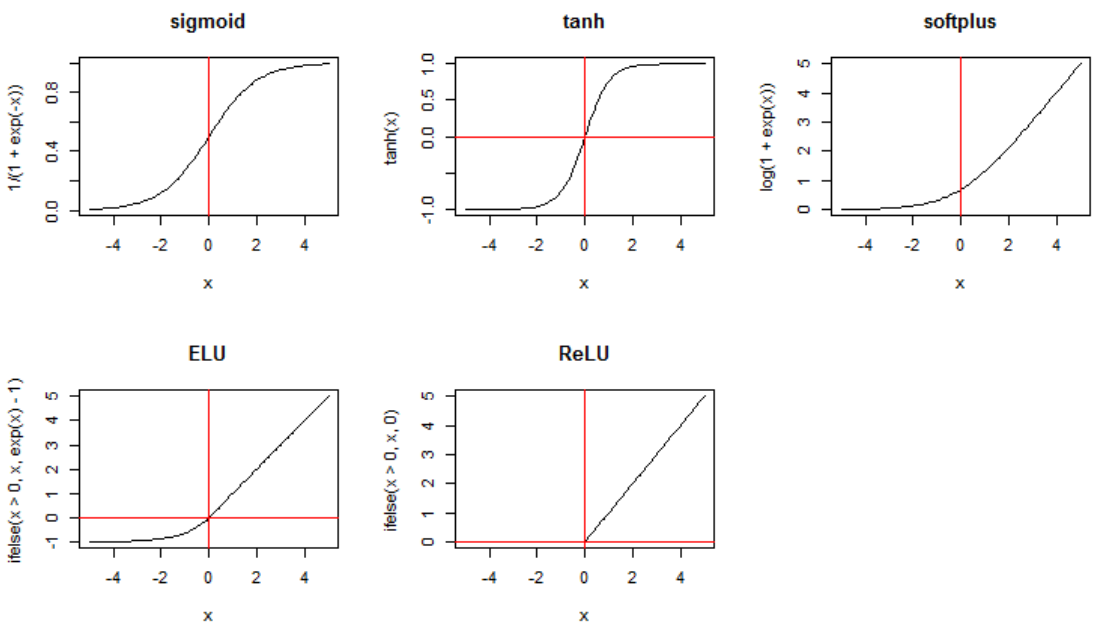
## Hora de treinar o modelo

```
[137] # Importando a função da Rede Nerual
      from sklearn.neural_network import MLPRegressor

      # Definindo os neuronios da rede
      Regresao_Neural = MLPRegressor( hidden_layer_sizes=(4, 4),
                                      activation='relu',
                                      solver='adam',
                                      alpha=0.0001,
                                      tol=0.0001,
                                      max_iter=1000,
                                      verbose=False )

      # Treinando o modelo
      print( Regresao_Neural.fit( x, y ) )

      # Calculando o Score da Regressão
      print('\n', Regresao_Neural.score( x, y ) )
```

```
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
             beta_2=0.999, early_stopping=False, epsilon=1e-08,
             hidden_layer_sizes=(4, 4), learning_rate='constant',
             learning_rate_init=0.001, max_fun=15000, max_iter=1000,
             momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
             power_t=0.5, random_state=None, shuffle=True, solver='adam',
             tol=0.0001, validation_fraction=0.1, verbose=False,
             warm_start=False)

 0.9391715861991892
```

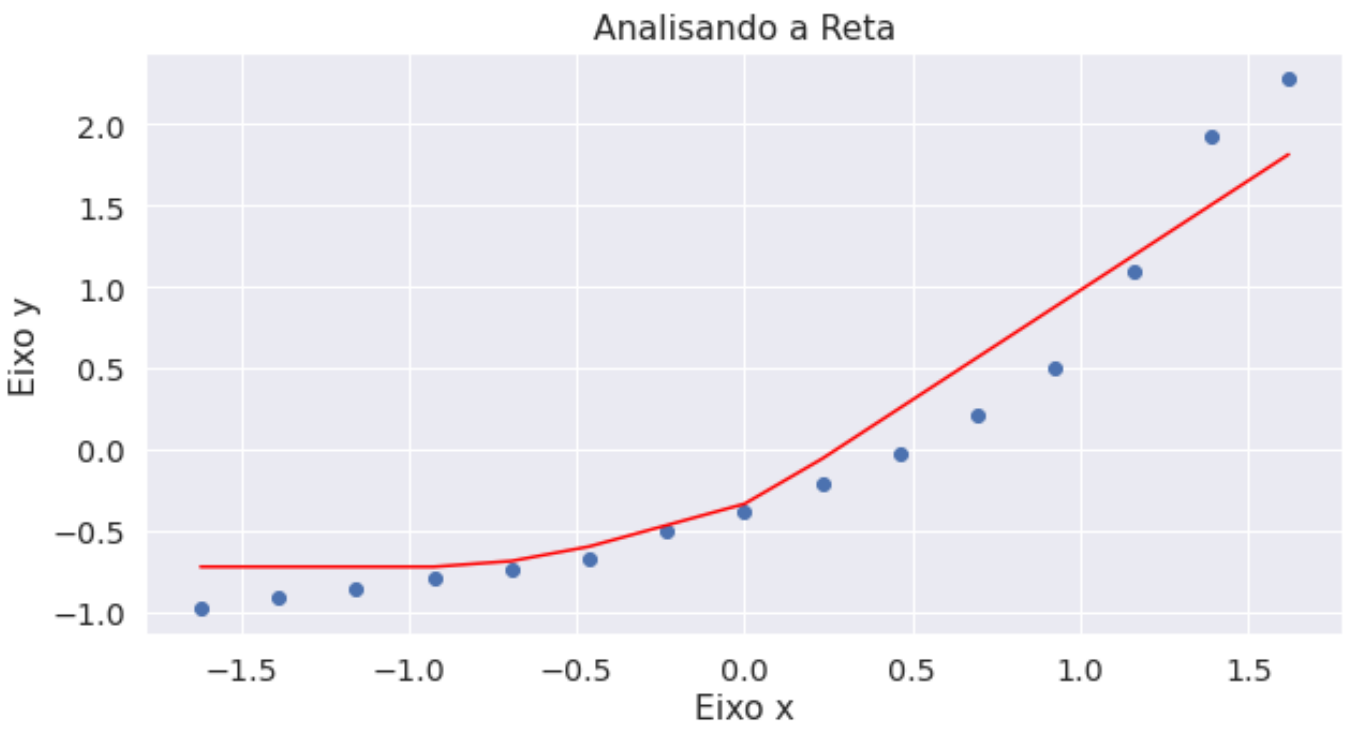
Essa mensagem mostra todos os **parâmetros** que utilizamos na rede neural.
No script fiz alguns ajustes na Rede para **melhorar a performance** dela.
Não vou abordar todos os conceitos porque poderíamos escrever um livro sobre esses parâmetros.

**Para exemplificar**, vamos mostrar alguns tipos de ativação



## Vamos plotar a reta do modelo

```
[138]  # -------------- Plotagem da Reta --------------

       # Ajustando o tamanho do Gráfico
       plt.figure( figsize=(10,5) )
       # Passando os valores para o grafico
       plt.scatter( x, y )
       # Plotando a reta gerada pela regressão
       plt.plot( x, Regresao_Neural.predict(x), color='red' )
       # Definindo um titulo
       plt.title('Analisando a Reta')
       # Definindo o nome do eixo x
       plt.xlabel('Eixo x')
       # Definindo o nome do eixo y
       plt.ylabel('Eixo y');
```




Veja que a reta se adaptou muito próximo ao dados e **fez aquela curva** que queríamos.

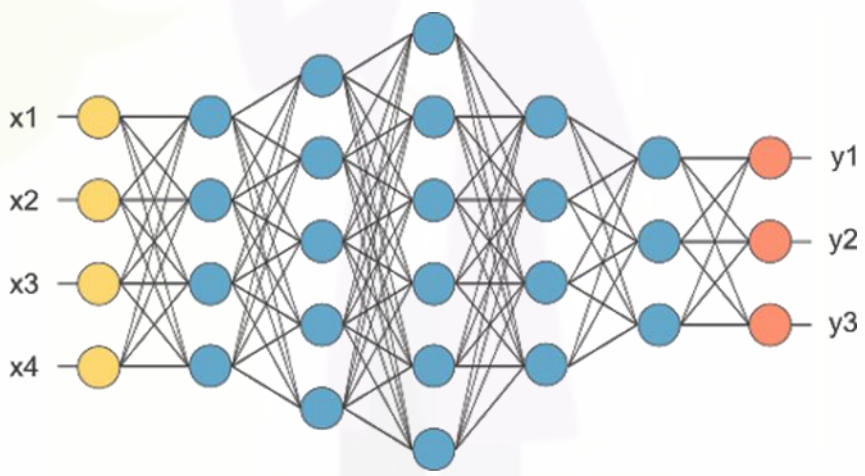
```
[139] # ------- Fazendo previsões -------
      # Vamos prever o valor de um apartamento de 50 metros
      Qual_Tamanho_Apartamento = [[ 50 ]]

      Escalando_Previsao = Escala_y.transform( Qual_Tamanho_Apartamento )
      Previsao = Regresao_Neural.predict( Escalando_Previsao )
      Transformando_Previsao = Escala_y.inverse_transform( Previsao )


      print('Um apartamento de:',
            Qual_Tamanho_Apartamento[0][0], 'metros' )
      print('Usando o modelo para prever o valor, custaria: R$',
            Transformando_Previsao[0] )
```

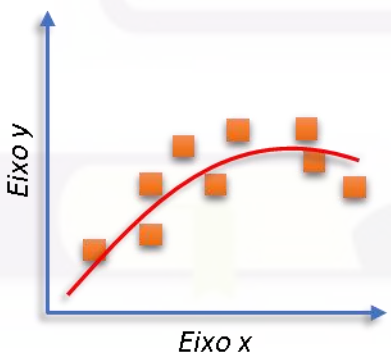
```
Um apartamento de: 50 metros
Usando o modelo para prever o valor, custaria: R$ 242.67653399135352
```

# Final

Esse guia é sobre como realizar uma regressão polinomial usando redes neurais.

Link da documentação, caso queira mais detalhes.
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html




**Odemir Depieri Jr**

Software Engineer Sr
Tech Lead
Specialization AI