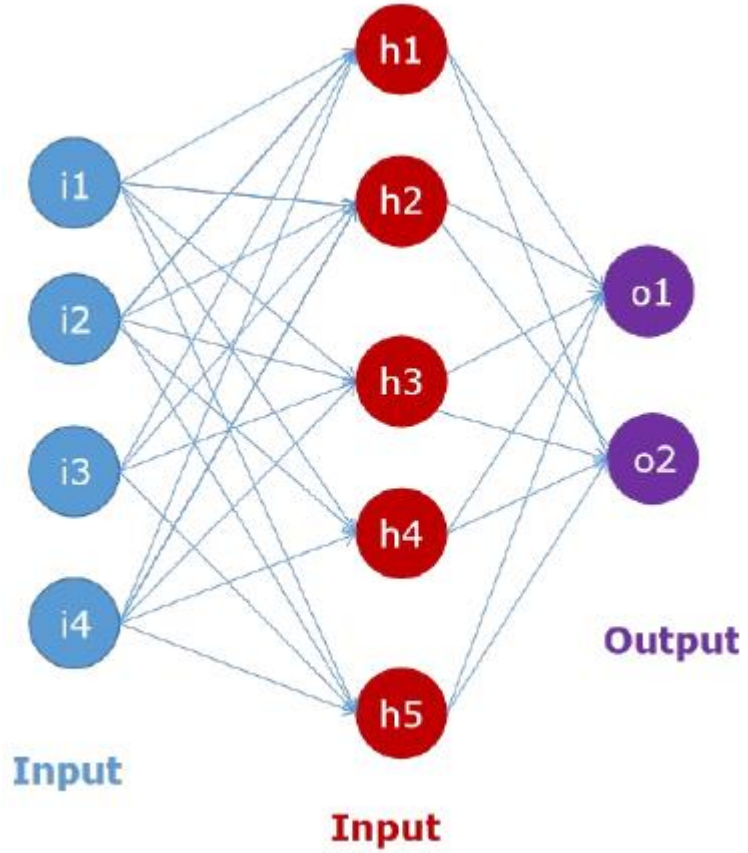
Redes Neurais Artificiais

Camila Nori Dias Kubota

GEIA (Hygor Santiago Lara)

Definição:

Rede Neural



Definição:

Rede Neural

- *“According to a simplified account, the human brain consists of about ten billion neurons -- and a neuron is, on average, connected to several thousand other neurons. By way of these connections, neurons both send and receive varying quantities of energy. One very important feature of neurons is that they don't react immediately to the reception of energy. Instead, they sum their received energies, and they send their own quantities of energy to other neurons only when this sum has reached a certain critical threshold. The brain learns by adjusting the number and strength of these connections. Even though this picture is a simplification of the biological facts, it is sufficiently powerful to serve as a model for the neural net.”*

Site da IBM (Visitado em: 14/09/2023)[1]

Em resumo:

Rede Neural

- Uma rede neural pode ter seu funcionamento aproximado ao de uma regressão linear.

$$\hat{y} = W^1 x^1 + W^2 x^2 + \ldots + W^n x^n + b$$

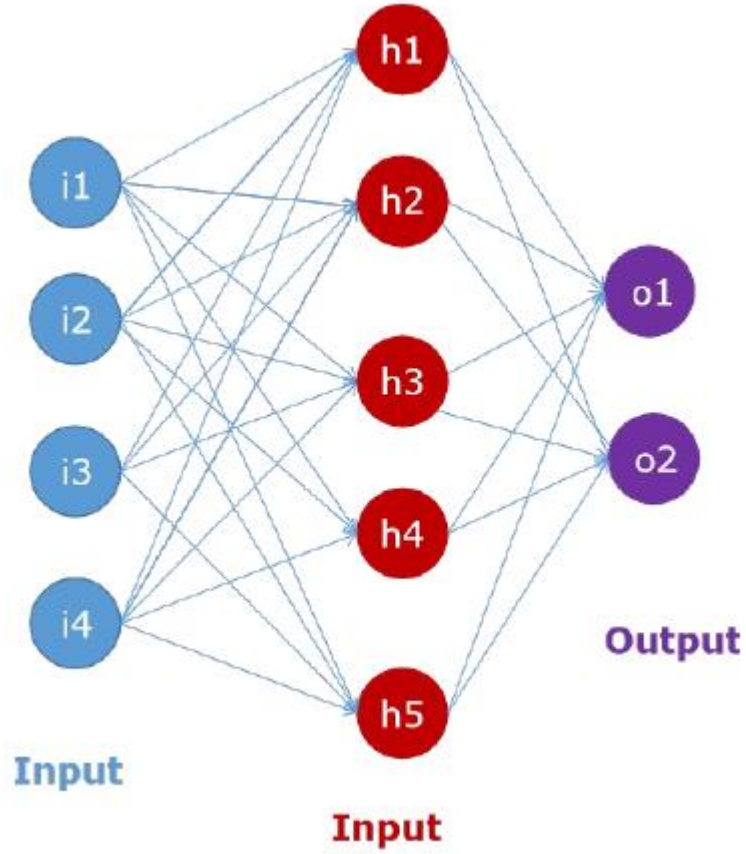
$$\hat{y} = W^T X + b$$

Representação Matricial

W – Pesos
b – Bias

Definição:

Rede Neural - Arquitetura



Input Layer – Inserção dos dados

Hidden Layer – Processamento dos dados

Output Layer – Camada de saída, previsões

Definição:

Regressão Linear

- *“Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.”*

Site da IBM (Visitado em: 14/09/2023)[2]

Definição:

Regressão Linear

$$\hat{y} = W^1 x^1 + W^2 x^2 + \ldots + W^n x^n + b$$

$$\hat{y} = W^T X + b$$

Representação Matricial

W – Pesos
b – Bias

$$\hat{y} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \ddots & \cdots & \vdots \\ x_{s1} & x_{s2} & \cdots & x_{sn} \end{pmatrix} + (b)$$

Exemplo:

Regressão Linear

Em uma pesquisa teórica, a altura e o peso do participante (2 *features*) poderiam prever o quanto ele gastaria em roupas durante um período específico, e essas características se correlacionam linearmente

$$Qtde\ dinheiro = W_{altura} \cdot altura + W_{peso} \cdot peso + b$$

Definição:

Ou seja:

- É um modelo voltado a problemas de previsão de valores.

    Ex: Prever o valor do preço de uma casa com base em suas características (quantos quartos, banheiros...).

Principal ponto que diferencia na hora de decidir utilizar Regressão Linear ou Regressão Logística

Ponto importante:

Algumas vezes as correlações entre os dados são muito complexas para uma regressão linear .

Algumas vezes os inputs das camadas lineares necessitam ser processados por outras funções por possuírem correlações não lineares ou necessidades de outputs específicos. Essas outras funções são chamadas funções de ativação.

Definição:

Regressão Logística

- *“This type of statistical model (also known as logit model) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn’t vote, based on a given dataset of independent variables.”*

Site da IBM (Visitado em: 19/04/2023)[1]

Definição:

Ou seja:

- É um modelo voltado a problemas de classificação, definições binárias.

Ex: Prever se um cliente irá ou não comprar seu produto.

Principal ponto que diferencia na hora de decidir utilizar Regressão Linear ou Regressão Logística

Tipos:

- Binária:

  Somente dois resultados possíveis.

  Ex: 0 e 1

- Multinomial

  De 3 a mais resultados possíveis, porém não ordenados

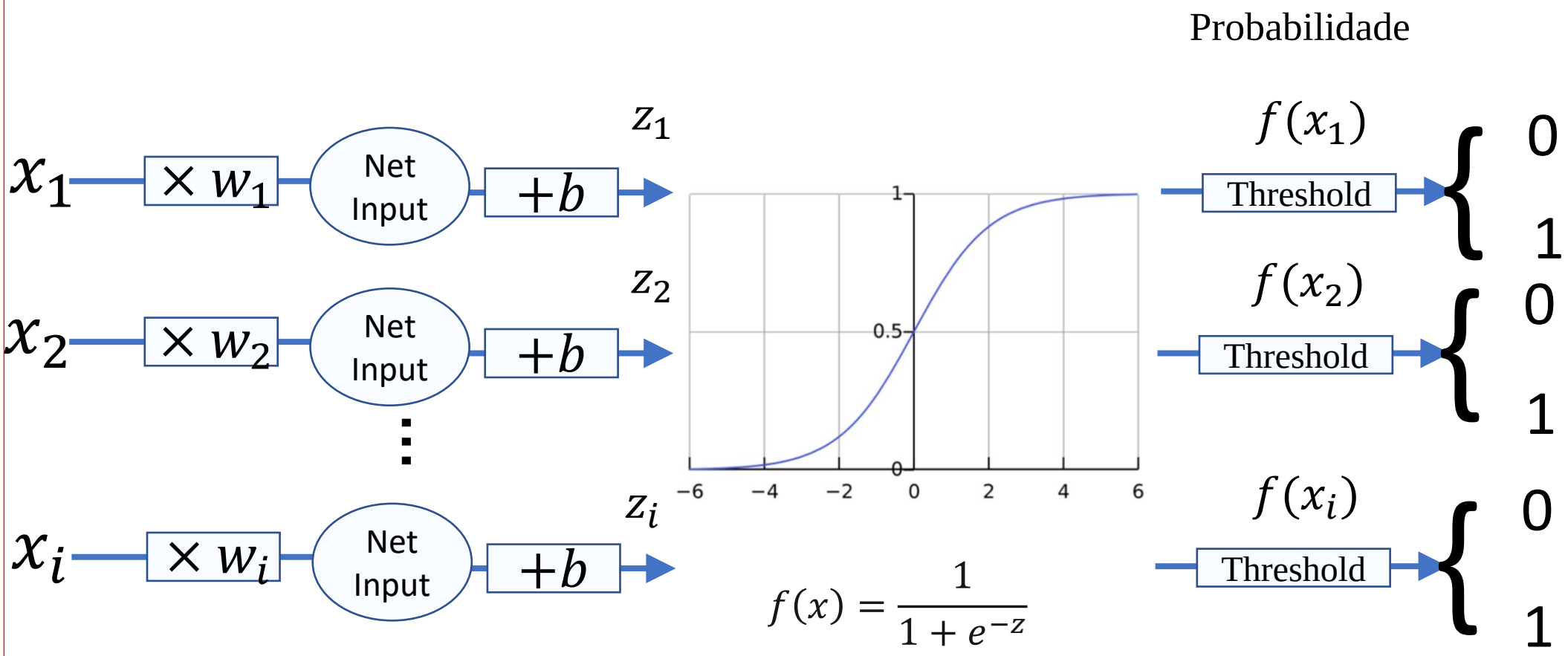
  Ex: Um estúdio deseja estudar a influência da idade, sexo e status de relacionamento em comparação ao gênero de filme favorito

- Ordinal:

  De 3 a mais resultados possíveis, mas ordenados

  Ex: Escala de 1 a 10 ou de A até F

Funcionamento:



Funcionamento:

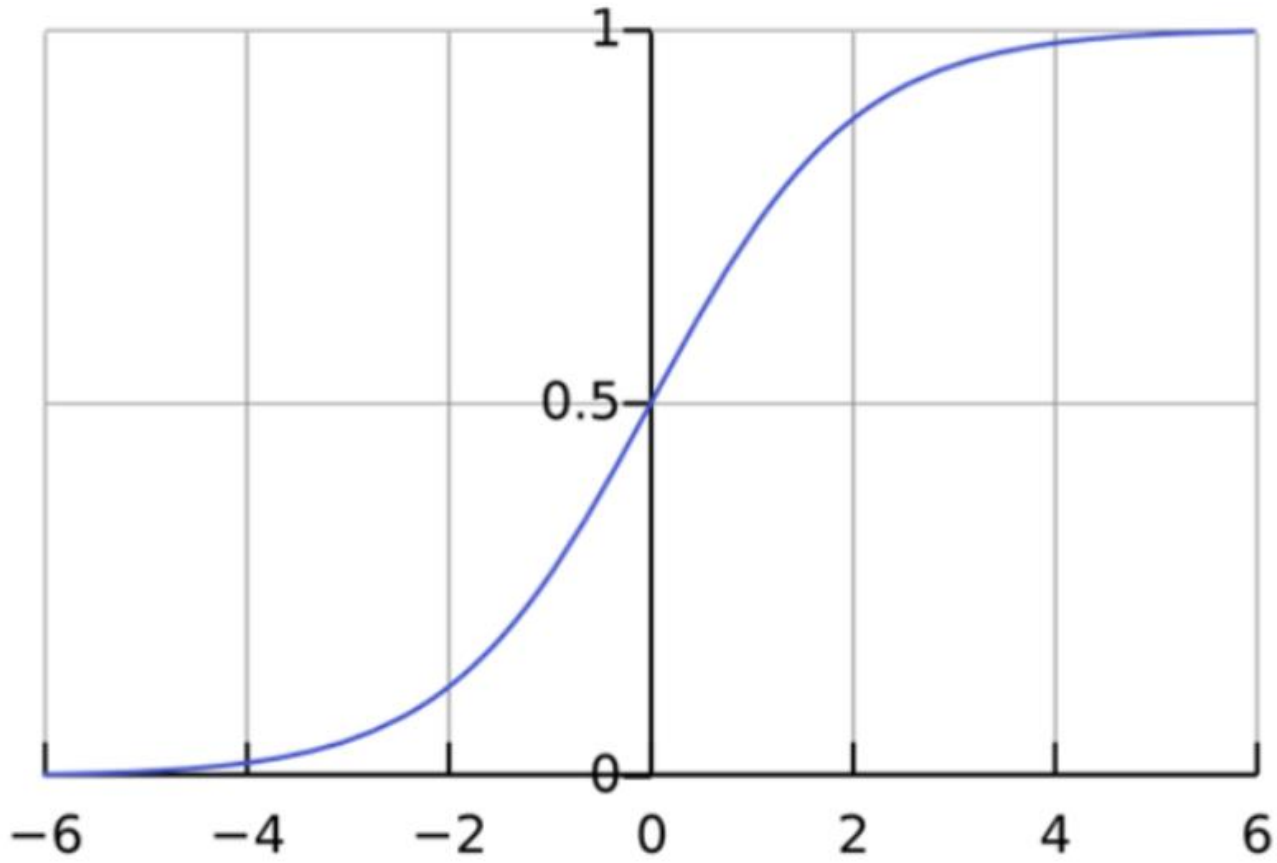
$$f(x) = \frac{1}{1 + e^{-z}}$$

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b$$

Função de ativação:

$$f(x) = \frac{1}{1 + e^{-z}}$$

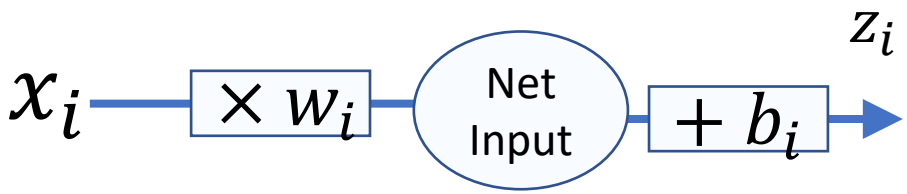
Função: Sigmoide



$$f(x) = \frac{1}{1 + e^{-z}}$$

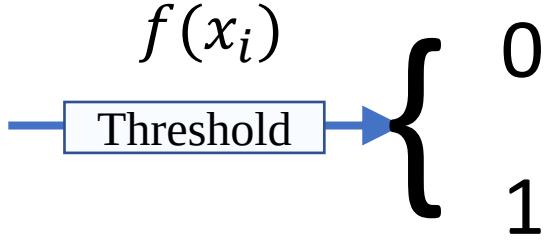
Retirado do site AWS Amazon[2]

Funcionamento:



$$\chi_i$$

Probabilidade



$$f(x) = \frac{1}{1 + e^{-z}}$$

Threshold:

É o “limiar”, valor definido em que:

- Caso a probabilidade esteja acima dele → Definido como o primeiro valor
- Caso esteja abaixo → Definido como o segundo valor

Exemplo:

$z \geq 0{,}75 \quad \rightarrow \quad = 1$

$z < 0{,}75 \quad \rightarrow \quad y = f(x) = 0$

Loss function:

Comumente utiliza-se como *loss function* a função denominada Cross Entropy para classificação e Mean Squared Error para regressão linear.

De forma resumida:

Cross Entropy

$$loss = -p_{observado} \times \ln(p_{prevista})$$

Mean Squared Error (MSE)

$$loss = \frac{1}{n}\Sigma_{i=1}^{n}(Y_i - \hat{Y})^2$$

Ponto importante:

Não necessariamente uma rede com mais camadas expressa bem a correlação dos seus dados.

Demonstração:

$$H = W_{(1)}X + b_{(1)}$$

$$Y = W_{(2)}H + b_{(2)}$$

$$Y = W_{(2)}H + b_{(2)} \Rightarrow Y = W_{(2)}(W_{(1)}X + b_{(1)}) + b_{(2)}$$

$$Y = \underbrace{W_{(1)}W_{(2)}}_{W}X + \underbrace{b_{(1)}W_{(2)} + b_{(2)}}_{b} \Rightarrow Y = WX + b$$

Ponto importante:

É uma boa prática normalizar os inputs

Deixar todas as variáveis latentes com o mesmo peso e manter seus valores entre 0 e 1, otimizando o gradiente descente.

Ponto importante:

Em certas situações otimizar e limpar seus dados pode ser mais eficiente na melhora do modelo do que alterar parâmetros da rede.

Um método de normalização:

Autoescalamento

$$x'_i = \frac{x_i - \bar{x}_i}{\sigma_i}$$

Um método de normalização:

Autoescalamento

$$\bar{x}_i = \frac{\left(\sum_{j=1}^{n} x_i^{(j)}\right)}{n}$$

$$\sigma_i = \sqrt{\frac{1}{n}\left(\sum_{i=1}^{n} x_i^{(j)} - \bar{x}_i\right)^2}$$

Bibliotecas

- Numpy
- Scikit-Learn
- Tensorflow (Keras)

Prática – Regressão Linear:

Banco de Dados

Foi utilizado o banco de dados do Auto mpg no site da Universidade de Irvine [3].

https://www.kaggle.com/datasets/uciml/autompg-datasetpetitions/titanic/data

Exemplos:

Numpy

```
#%% numpy
import numpy as np
'''
Esta etapa define a classe Regressão linear e os métodos fit e predict.
'''
class LinearRegression():

    def __init__(self,lr=0.001,n_iters=1000):
        self.lr= lr
        self.n_iters=n_iters
        self.weights=None
        self.bias=None

    def fit(self, X, y):
        n_samples, n_features= X.shape
        self.weights=np.zeros(n_features)
        self.bias=0

        for _ in range(self.n_iters):
            linear_predictions=np.dot(X, self.weights)+self.bias
            predictions= linear_predictions
            dw=(1/n_samples)*np.dot(X.T,(predictions-y))
            db=(1/n_samples)*np.sum(predictions-y)

            self.weights=self.weights-self.lr*dw
            self.bias=self.bias-self.lr*db
```

```
    def predict(self,X):
        linear_predictions=np.dot(X, self.weights)+self.bias
        y_pred=linear_predictions
        return y_pred
```

```
def precision(y_pred,y_test):
    prec=[((a-b)**2) for  a, b in zip(y_pred, y_test)]
    return np.mean(prec)
```

```
regr=LinearRegression(lr=0.01,n_iters=1000)
regr.fit(X_train,y_train)
y_pred=regr.predict(X_val)
```

```
prec=precision(y_pred, y_val)
print(f"Erro Quadrado Médio: {prec}")
```

Exemplos:

Scikit-Learn

```
#%%  Scikit-learn
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

model = LinearRegression()  # Inicializar o modelo de Regressão Linear
model.fit(X_train,y_train)  # Treinar o modelo

linear_pred = model.predict(X_val)  # Fazer predições com o modelo treinado

# Calcular o erro quadrado médio
print('Erro Quadrado Médio: {}'.format(mean_squared_error(y_val,linear_pred)))
```

Exemplos:

Tensorflow (Keras)

```
#%%  Tensorflow
import tensorflow as tf
from tensorflow import keras
import numpy as np
#Construindo o modelo
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=X_train.shape[1:]),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)
])

optimizer = tf.keras.optimizers.RMSprop(0.001)

model.compile(loss='mse',
              optimizer=optimizer,
              metrics=['mae', 'mse'])

model.summary()
```

```
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
history = model.fit(X_train, y_train, validation_data=(X_val, y_val),
                    epochs=1000, batch_size=32, callbacks=[early_stop])

predc=model.predict(X_val)

import numpy
prec=[((a-b)**2) for  a, b in zip(predc, y_val)]
print('Erro Quadrado Médio: ' +  str(np.mean(prec)))
```

Prática – Regressão Logística:

Banco de Dados

Foi utilizado o banco de dados do Titanic no Kaggle [3].

https://www.kaggle.com/competitions/titanic/data

Exemplos:

Numpy

```
#%% Numpy
'''
Definição da classe Regressão Logística e seus métodos
'''
import numpy as np

def sigmoid(x):
    return (1/(1+np.exp(-x)))

class LogisticRegression():

    def __init__(self,lr=0.001,n_iters=1000):
        self.lr= lr
        self.n_iters=n_iters
        self.weights=None
        self.bias=None

    def fit(self, X, y):
        n_samples, n_features= X.shape
        self.weights=np.zeros(n_features)
        self.bias=0

        for _ in range(self.n_iters):
            linear_predictions=np.dot(X, self.weights)+self.bias
            predictions=sigmoid(linear_predictions)

            dw=(1/n_samples)*np.dot(X.T,(predictions-y))
            db=(1/n_samples)*np.sum(predictions-y)

            self.weights=self.weights-self.lr*dw
            self.bias=self.bias-self.lr*db

    def predict(self,X):
        linear_predictions=np.dot(X, self.weights)+self.bias
        y_pred=sigmoid(linear_predictions)
        class_pred=[0 if y<=0.5 else 1  for y in y_pred]
        return class_pred


def accuracy(y_pred,y_test):
    return np.sum((y_pred==y_test)/len(y_test))


regr=LogisticRegression(lr=0.05,n_iters=700)
regr.fit(x_train,y_train)
y_pred=regr.predict(x_test)


acc=accuracy(y_pred, y_test)
print(f'Accuracy {acc}')
```

Exemplos:

Scikit-Learn

```
#%% Scikit-learn
from sklearn.linear_model import LogisticRegression
import numpy as np

Previs_titanic=LogisticRegression()  # Definindo o modelo
Previs_titanic.fit(X_train,y_train)  #Treinando o modelo


testing=Previs_titanic.predict(X_test)

print('Acurácia: {}'.format(np.sum((testing==y_test)/len(y_test))))
```

Exemplos:

Tensorflow (Keras)

```
#%% Tensorflow
import tensorflow as tf
from tensorflow import keras

# Definindo o modelo
model=keras.Sequential([keras.layers.Dense(1,input_shape=(len(labels),),
                                           activation='sigmoid')])
model.compile(optimizer='adam',loss='binary_crossentropy')


# Treinando o modelo
hist=model.fit(X_train,y_train,epochs=1000,validation_split=0.2)


predictions=model.predict(X_test)


# Definindo o threshold
for i in range(len(predictions)):
    predictions[i]=1 if predictions[i]>=0.75 else 0
predictions = [int(x[0]) for x in predictions]


acc=np.sum((y_pred==y_test)/len(y_test))
print('Acurácia:' + acc)
```

Referências:

[1] What are Neural Networks? | IBM. https://www.ibm.com/topics/neural-networks.

[2] About Linear Regression | IBM. https://www.ibm.com/topics/linear-regression.

[3] https://www.ibm.com/topics/logistic-regression#:~:text=Resources-,What%20is%20logistic%20regression%3F,given%20dataset%20of%20independent%20variables.

[4] https://aws.amazon.com/pt/what-is/logistic-regression/#:~:text=A%20regress%C3%A3o%20log%C3%ADstica%20%C3%A9%20uma,resultados%2C%20como%20sim%20ou%20n%C3%A3o.

[5]https://edisciplinas.usp.br/pluginfile.php/3769787/mod_resource/content/1/09_RegressaoLogistica.pdf

[6] https://statquest.org/video-index/

[7] Jurafsky D., Martin J. H., "*Speech And Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, And Speech Recognition*" , 3ª edição,