**Regressão Lógica**

É um algoritmo de classificação, usamos para estimar valores discretos (valores binários como 0/1, sim/não, verdadeiro/falso) baseado no set dado de variáveis independentes. Em simples palavras, prevê a probabilidade da ocorrência de um evento apropriando os dados para uma função lógica. Consequentemente, é conhecido como regressão lógica. Desde que, prevê a probabilidade dos valores de saída se encontram entre 0 e 1 (como esperado).

Vamos por um simples exemplo:

Vamos dizer que seus amigos te dão um quebra-cabeça para resolver. Tem apenas dois cenários de saída, ou você resolve ou não. Agora imagina, que você tem recebido uma grande quantidade de quebra-cabeças/enigmas em uma tentativa de entender em quais assuntos você é bom. A saída desse estudo vai ser algo tipo isso – Se você receber um problema de décima série baseado em trigonometria, você tem 70% de probabilidade de resolver. Agora, se é uma questão de história de quinta série, a probabilidade de você obter uma resposta é 30%. Isso é o que a Regressão lógica te fornece.

Convertendo para a matemática, a razão de possibilidades (*log odds*) da saída é modelada como uma combinação linear das variáveis preditoras.

Probabilidades = p/(1-p)

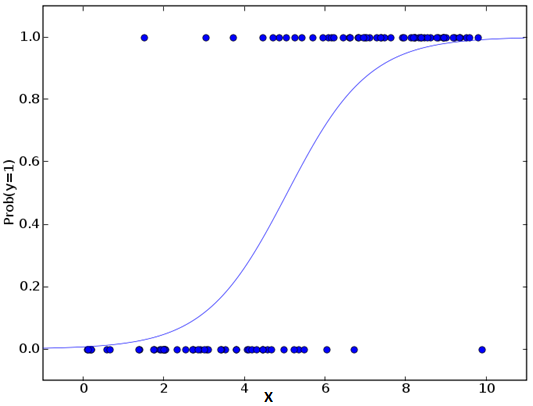
= probabilidade da ocorrência do evento / probabilidade de não ocorrer o evento Ln(probabilidades) =

Ln(p/(1-p)) logit(p)

= ln(p/(1-p)) = b0+b1X1+b2X2+b3X3....+bkXk

Acima, p é a probabilidade da presença da característica do interesse. Escolhe parâmetros que maximiza a probabilidade que observa os valores de exemplo em vez de minimizar a soma dos erros ao quadrado (como em uma regressão ordinária).

Agora você pode perguntar, porquê ter um log odds? Pela simplicidade, vamos dizer que esse é um dos melhores caminhos matemáticos para replicar uma função de passo.



**Código em Python**

#Importar Biblioteca de sklearn.linear\_model

*Import LogisticRegression*

#Digamos que você tenha, X (preditor) e Y (alvo) para treinar um set de dados e x\_test(preditor) de test\_dataset

#Criar um objeto de Regressão Lógica

*model = LogisticRegression()*

#Treinar o modelo usando os sets de treinamento e validar os pontos

*model.fit(X, y)*

*model.score(X, y)*

#Equação coeficiente e interceptor

*print('Coefficient: \n', model.coef\_)*

*print('Intercept: \n', model.intercept\_)*

#Prever saída

*predicted= model.predict(x\_test)*

Além disso, tem mais passos diferentes que pode tentar em ordem que podem melhorar esse modelo:

* Incluir termos de interação
* Remover características
* [Técnicas de regularização](https://www.analyticsvidhya.com/blog/2015/02/avoid-over-fitting-regularization/)
* Usar um modelo não-linear