Sumário

[Modelo de Regressão Linear 2](#_Toc83029971)

[Introdução 2](#_Toc83029972)

[Função Custo 4](#_Toc83029973)

[método dos mínimos quadrados 4](#_Toc83029974)

[método do gradiente 4](#_Toc83029975)

[Testes nos resíduos 6](#_Toc83029976)

[Métricas para avaliar o modelo 7](#_Toc83029977)

[Seleção de variável: 9](#_Toc83029978)

[Classificação: Regressão Logística 10](#_Toc83029979)

[Introdução às Tarefas de Classificação 10](#_Toc83029980)

[Regressão Logística 10](#_Toc83029981)

[Transformação 11](#_Toc83029982)

[Logística 11](#_Toc83029983)

[Chances (Odds) 12](#_Toc83029984)

[Aplicação 13](#_Toc83029985)

[Estimação e Interpretação do Modelo 14](#_Toc83029986)

[Métricas de avaliação 14](#_Toc83029987)

[Acurácia 15](#_Toc83029988)

[Matriz de Confusão 15](#_Toc83029989)

[Medidas de Desempenho Derivadas da Matriz de Confusão 16](#_Toc83029990)

[F-Score 17](#_Toc83029991)

[Análise ROC e Coeficiente de Gini 17](#_Toc83029992)

[Árvore de Decisão 19](#_Toc83029993)

[Introdução 19](#_Toc83029994)

[Entropia e Ganho de Informação 21](#_Toc83029995)

[Implementações Práticas 23](#_Toc83029996)

[Algoritmos de Clusterização 25](#_Toc83029997)

[Introdução 25](#_Toc83029998)

[Métricas de Proximidade 27](#_Toc83029999)

[Atributos Quantitativos 27](#_Toc83030000)

[Atributos Qualitativos 28](#_Toc83030001)

[K-Means 28](#_Toc83030002)

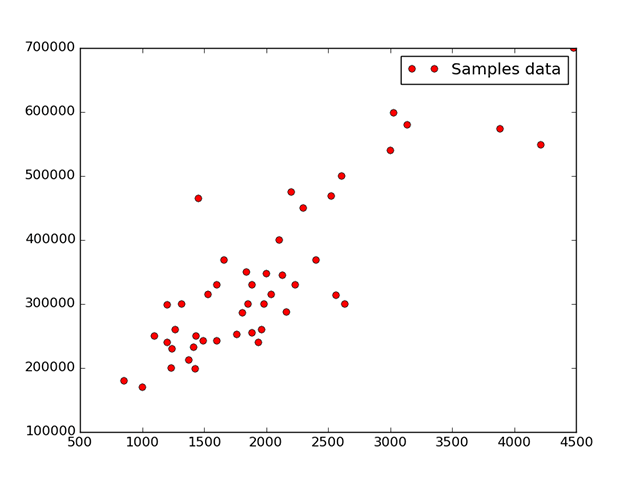
# Modelo de Regressão Linear

## Introdução

Estamos interessados em avaliar, de modo exploratório, um dos modelos estatíısticos mais utilizados na prática, conhecido como modelo de regressão. O exemplo mais simples serve para a an´alise de dados pareados (x1,y1),...,(xn,yn)(x1,y1),...,(xn,yn) de duas variáveis contínuas X e Y num contexto em que sabemos a priori que a distribuicão de frequências de Y pode depender de X, ou seja, a variável explicativa e Y e a variável resposta.

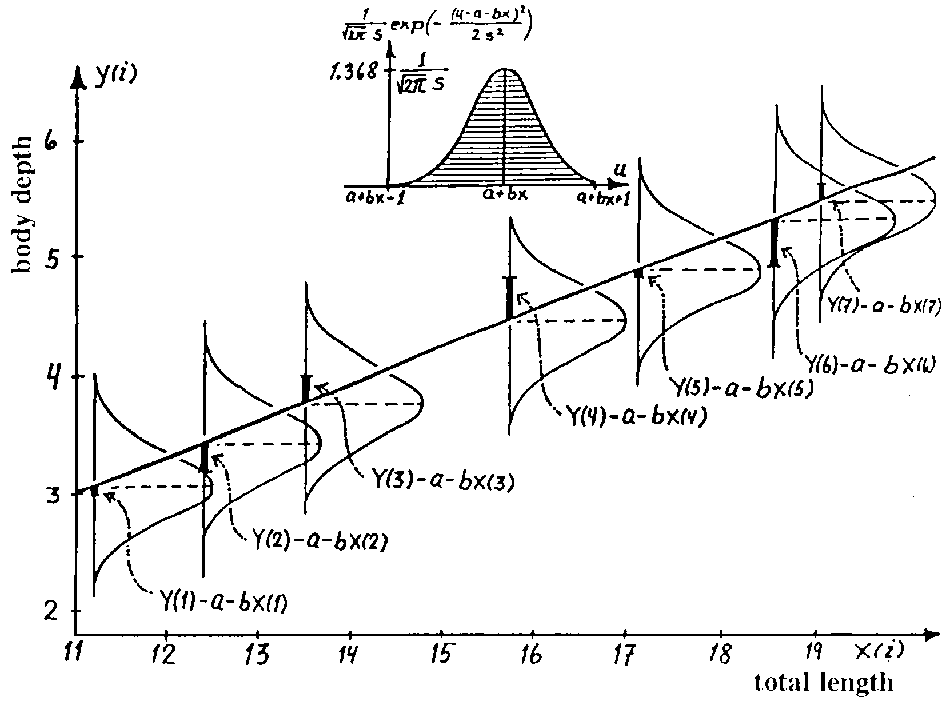
No caso geral em que temos n pares de dados, o modelo de regressão utilizado para essa quantificação:

yi=θ0+θ1xi+ϵi,i=1,...,nyi=θ0+θ1xi+ϵi,i=1,...,n



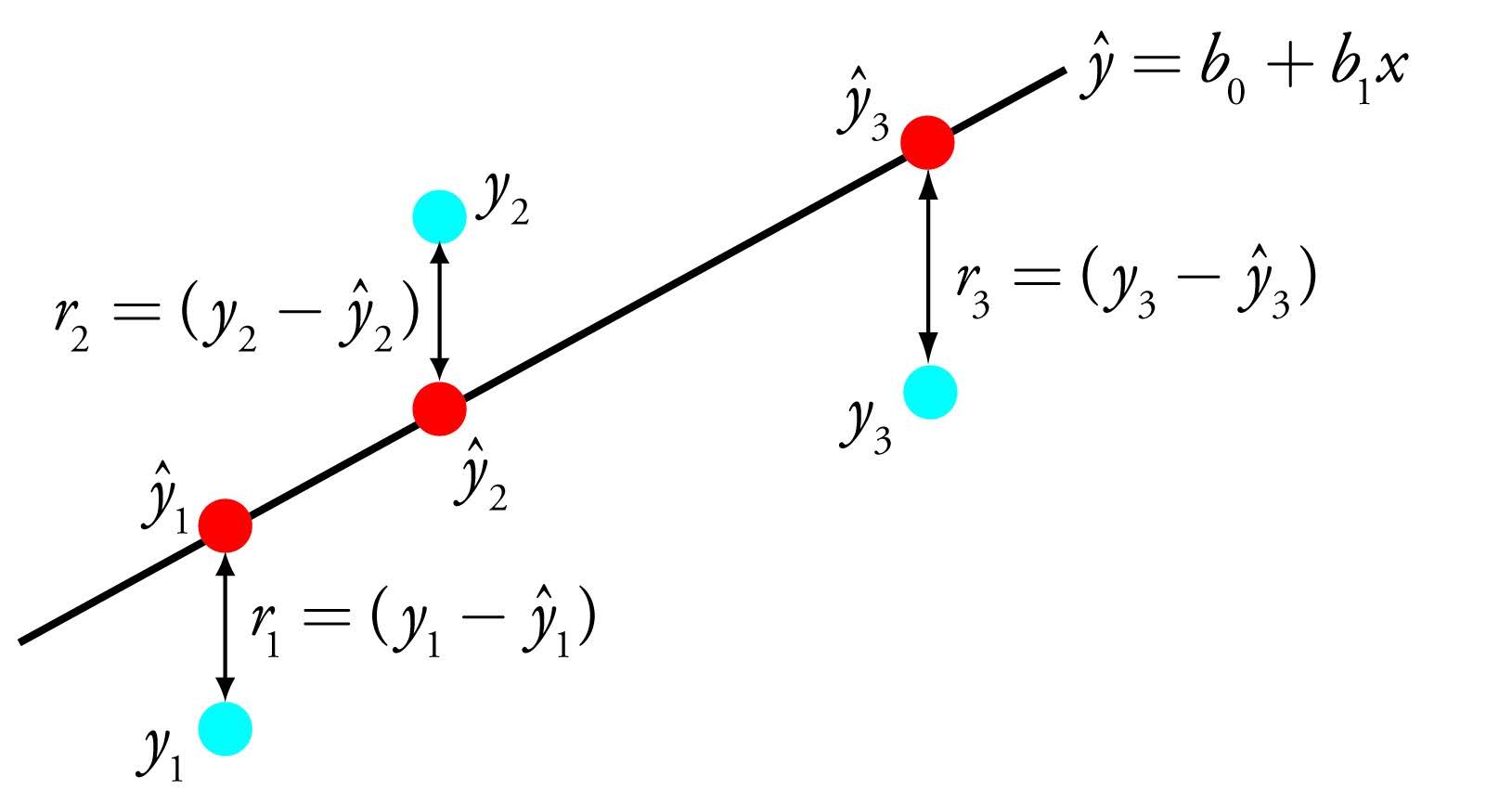
os comportamentos do erro, fazemos a assunção de que eles tem um comportamento normal, isto é, e N(0,σ²)e N(0,σ²)

* comportamento de erros:



* qual função para definir a função de custo ?

Precisamos definir uma função em que conseguimos definir os parâmetros αα**e**ββ, qual melhor função que podemos escolher com que minimizamos o erro que vamos cometer :



*podemos observar que podemos minimizar os erros da nossa****reta estimada****com os pontos observados, mas se consideramos os pontos abaixo da reta esse valor estariámos negativando nossa função, e gostariamos de apenas saber o valor da "distância" obtida, uma forma de contornar isso é tornando esse erro****quadrático***

## Função Custo

Podemos medir a precisão de nossa função de hipótese usando uma função de custo. Isso leva a uma diferença média (na verdade, uma versão mais sofisticada de uma média) de todos os resultados da hipótese com entradas de x e a saída real y.

J(θ0,θ1)=12m∑i=1m(y^i−y)2J(θ0,θ1)=12m∑i=1m(y^i−y)2

Sabemos que a derivada é uma função de taxa de variação, se encontrarmos a derivada = 0 encontramos exatamente o ponto em que maximizamos ou minimizamos nossa função

## método dos mínimos quadrados

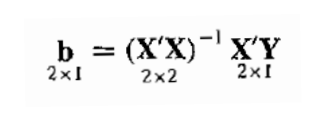
podemos obter de forma **fechada** nossos parâmetros por meio das derivadas parciais da nossa função custo acima obtendo os seguintes parâmetros

* **parâmetros**:

θ0=y¯¯¯+θ1∗x¯¯¯θ0=y¯+θ1∗x¯

θ1=∑i(xi−x¯¯¯)(yi−y¯¯¯)∑i(xi−x¯¯¯)2θ1=∑i(xi−x¯)(yi−y¯)∑i(xi−x¯)2

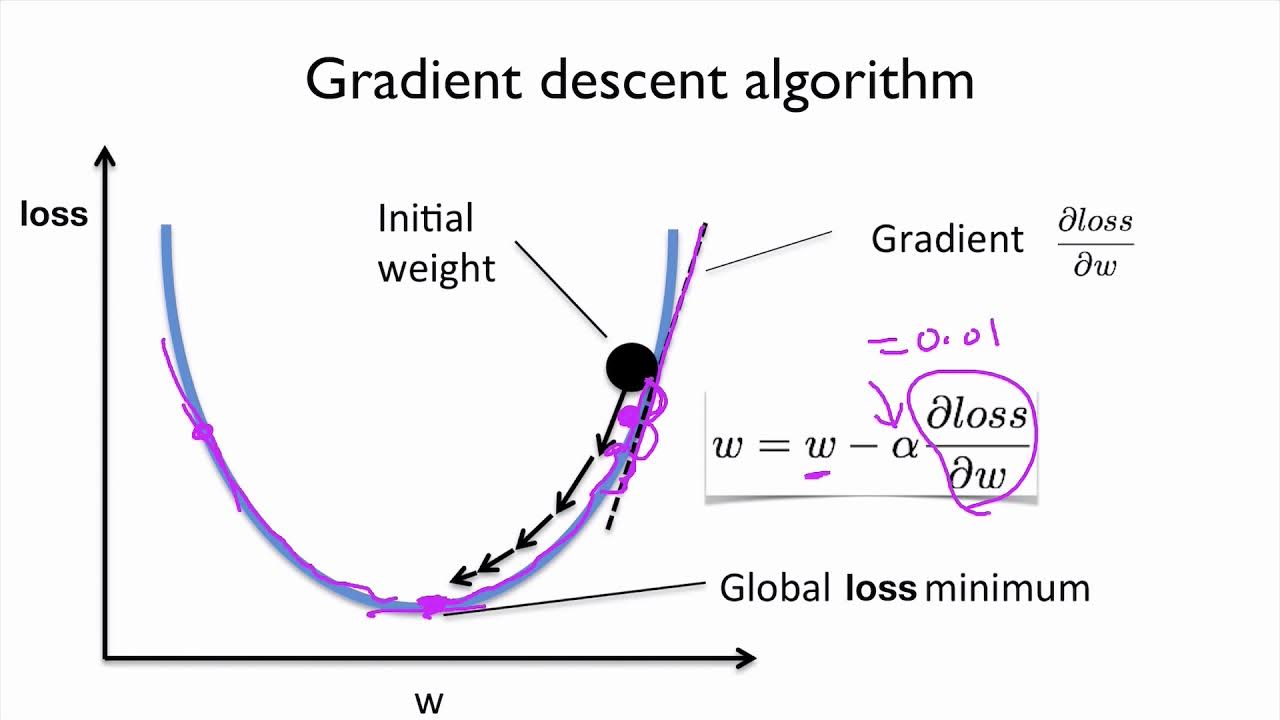
*forma matricial :*



## método do gradiente

A forma do gradiente utiliza as derivadas ( que são a taxa e direção aonde nossa função cresce) e métodos iterativos, para atualizar de forma dinâmica nossos coeficientes até que convirgam para um lugar comum, nesse contexto temos o que chamamos de **tamanho do passo ou learning rate**

* caminhando no sentido em que minimiza as funções:



* tamanho do passo

O passo podemos cometer alguns erros no **aprendizado**, ele pode ser :

*pequeno demais: quando definimos um passo tão pequeno que ele precisa caminha milhares de vezes e torna o código lento*

*passo ideal : o passo tem uma convergência "ótima" para os valores que minimizam a função.*

*passo largo : nesse o passo constuma ser tão grande que passa o mínimo e fica indo e voltando sem conseguir chegar no mínimo da nossa função*



## Testes nos resíduos

Existem muitos testes estatísticos que podemos usar para quantificar se uma amostra de dados parece ter sido extraída de uma distribuição gaussiana.

Cada teste faz suposições diferentes e considera diferentes aspectos dos dados.

Cada teste retornará pelo menos duas coisas:

**Estatística**: uma quantidade calculada pelo teste que pode ser interpretada no contexto do teste, comparando-a com valores críticos da distribuição da estatística do teste.

**Valor-p**: usado para interpretar o teste, neste caso, se a amostra foi retirada de uma distribuição gaussiana.

Na implementação SciPy desses testes, você pode interpretar o valor p da seguinte maneira.

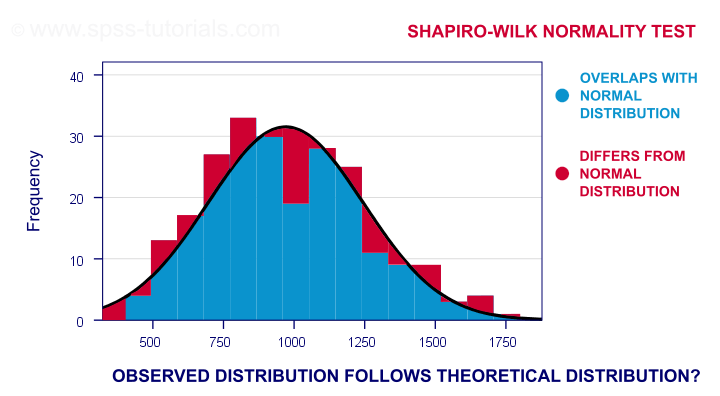
-- Nosso teste esta pautado então na seguinte hipótese

H0=NormalH0=Normal

H1=NaoNormalH1=NaoNormal

**p <= alfa:** rejeita H0, não normal.`

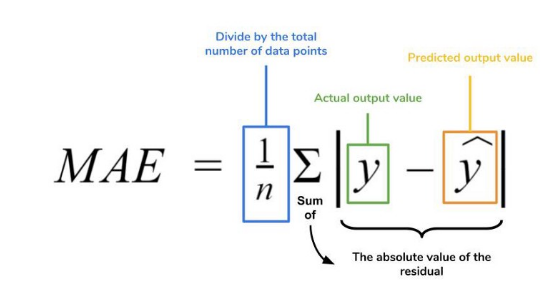
**p> alfa:** não rejeita H0,logo nossa distribuição é normal.`



## Métricas para avaliar o modelo

* **MAE**

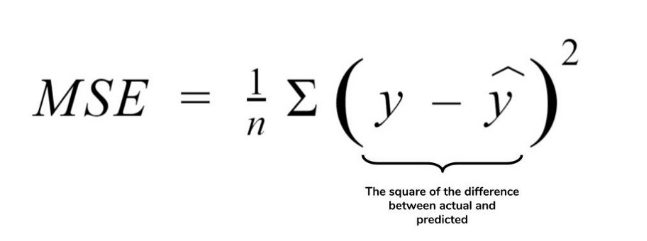
O MAE mede a magnitude média dos erros em um conjunto de previsões, sem considerar sua direção. É a média na amostra de teste das diferenças absolutas entre previsão e observação real, em que todas as diferenças individuais têm peso igual.



O erro médio absoluto usa a mesma escala que os dados. Isso é conhecido como uma medida de precisão dependente da escala e, portanto, não pode ser usado para fazer comparações entre séries usando escalas diferentes.

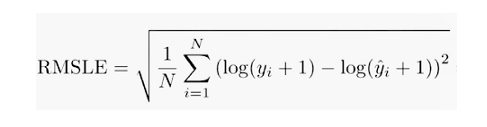
* **MSE**

MSE é uma função de risco, correspondente ao valor esperado da perda de erro ao quadrado. O fato de o MSE quase sempre ser estritamente positivo (e não zero) é por causa da aleatoriedade ou porque o estimador não leva em consideração informações que possam produzir uma estimativa mais precisa. O MSE é uma medida da qualidade de um estimador - é sempre não negativo e os valores mais próximos de zero são melhores.



“Minimizar o MSE é um critério-chave na seleção de estimadores: veja erro médio quadrático mínimo. Entre os estimadores imparciais, minimizar o MSE é equivalente a minimizar a variação, e o estimador que faz isso é o estimador imparcial de variância mínima.

* **RMSE: Root mean square error**
  + O RMSE é uma regra de pontuação quadrática que também mede a magnitude média do erro. É a raiz quadrada da média das diferenças quadráticas entre previsão e observação real.



* *Comparação RMSE vs MAE*

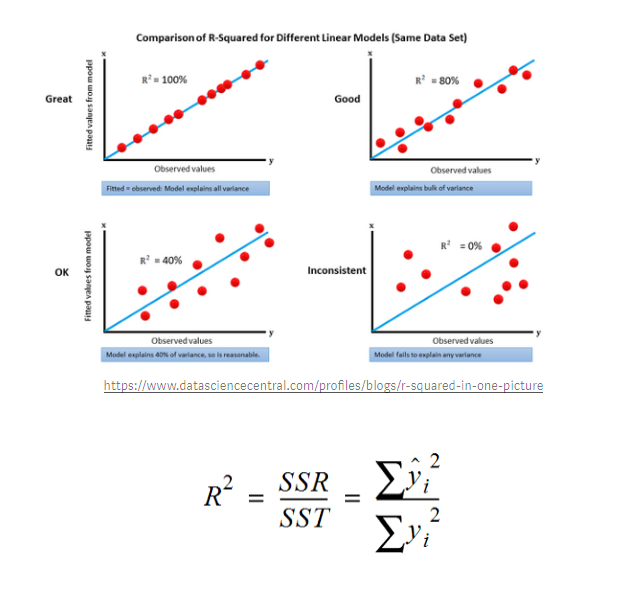
*Semelhanças:*

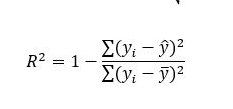
* + Expresse o erro médio de previsão do modelo nas mesmas unidades da variável de interesse. Pode variar de 0 a ∞ e são indiferentes à direção dos erros. Valores mais baixos são melhores.

*Diferença:*

* + Tomando a raiz quadrada antes da média, o RMSE atribui um peso relativamente alto a erros grandes; portanto, o RMSE deve ser útil quando erros grandes são indesejáveis.
* R²R²

R² e R-quadrado nos ajudam a saber quão bom é o nosso modelo de regressão em comparação com um modelo muito simples que apenas prevê o valor médio do alvo do trem definido como previsões.





## Seleção de variável:

* Stepwise

O método Stepwise para a seleção de variáveis é muito usado em regressão linear.

Qualquer procedimento para seleção or exclusão de variáveis de um modelo é baseado em um algoritmo que checa a importância das variáveis, incluindo ou excluindo-as do modelo se baseando em uma regra de decisão. A importância da variável é definida em termos de uma medida de significância estatística do coeficiente associado à variável para o modelo. Essa estatística depende das suposições do modelo.

Temos a seguir o algoritmo do método de Stepwise passo a passo;

**Passo 0**: Suponha que temos p variáveis explicativas candidatas ao modelo. Esse passo começa com o ajuste apenas do intercepto e seja L0L0 o log da verossimilhança desse ajuste. Após isso, ajustamos os p modelos com apenas uma variável explicativa. L(0)jLj(0) é o log da verossimilhança do modelo contendo a variável xjxj. O valor do teste da Razão de Verossimilhança do modelo contendo xjxj versus o modelo com apenas o intercepto é G(0)j=−2(L0−L(0)j)Gj(0)=−2(L0−Lj(0))

**Passo 1**: Ajustamos agora o modelo contendox1x1. Seja L(1)e1Le1(1) o log da verossimilhança desse modelo. Para verificar se p-1 variáveis são importantes para o modelo, uma vez que x1x1 está nele, ajustamos p-1 modelos de regressão contendo x1x1 e xjxj.

**Passo 2**: O passo 2 começa com o ajuste do modelo contendo x1x1 e x2x2. É possível que, com a inclusão de x2x2, x1x1 passa a ser não significativa para o modelo. Por isso, nesse passo testamos a significância de uma das variáveis dado que a outra está no modelo.

e novas variáveis entram e as estatisticas **F** são cálculadas para saber se a inclusão de novas variáveis

# Classificação: Regressão Logística

## ****Introdução às Tarefas de Classificação****

Problemas de Classificação: deseja-se determinar a que **categoria** (ou **classe**) dentro de um **conjunto de categorias** uma dada observação pertence, com base em suas features. Nesse tipo de problema, o target é uma **classe**.

Exemplos de problemas de classificação:

* Detecção de e-mails SPAM: um e-mail é SPAM ou não?;
  + Features: palavras contidas no corpo do e-mail; remetente; assunto;
* Detecção de doenças: que codição médica a pessoa tem?
  + Features: sintomas fisiológicos; resultados de exames (medidas de variáveis biológicas);
* Detecção do tipo de documento: secreto, confidencial ou não-sensível?
  + Features: palavras no corpo do texto; título;
* Detecção de fraudes de cartão de crédito: uma operação é fraudulenta ou não?;
  + Features: histórico de transações; hora, local e frequência das transações; tipo de compra;
* Modelo de risco de crédito: qual é a chance de determinada pessoa não pagar seu empréstimo?
  + Features: histórico de pagamento; score de crédito;

Para resolver problemas de classificação, construímos um **classificador**: modelo que tem como input as features (contínuas ou discretas) e como output uma entre as classes (discretas).

### ****Regressão Logística****

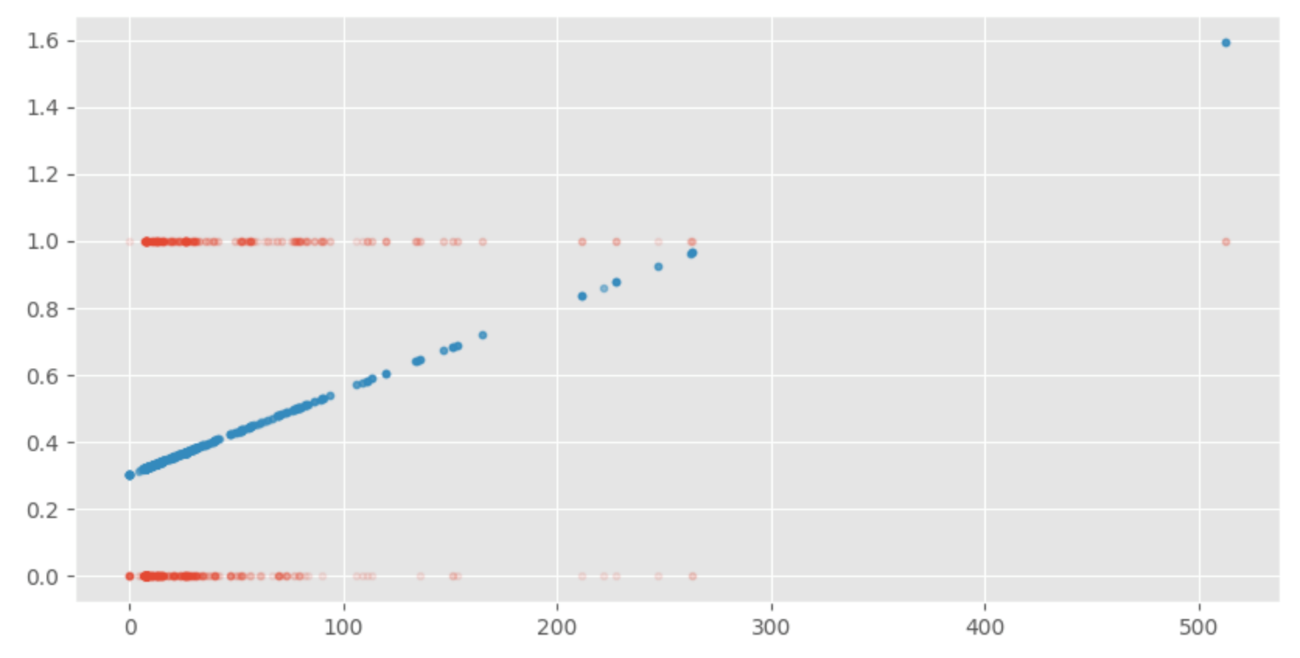
Uma Regressão Linear Múltipla buscar entender os relacionamentos entre diversas variáveis explicativas (x) com a variável resposta (y), que, de forma matemática, busca achar os coeficiente (b) da equação:

y=β0+β1x1+β2x2+...+βnxny=β0+β1x1+β2x2+...+βnxn

No caso das Regressões Lineares, a variável resposta é contínua. Dessa forma, ao tentar estimar o preço de um imóvel usando sua área em m2m2, quantidade de quartos e quantidade de banheiros, a Regressão Linear pode ser uma boa opção.

Porém, como proceder nos casos que a variável resposta não é contínua, mas binária? Imagine que, dado o valor pago por um passageiro em uma cabine no Titanic, queremos saber se ele sobreviveu ou não.

Matematicamente, podemos representar a sobrevivência como 1 e a não sobrevivência como 0, e a partir daí fazer a modelagem usando Regressão Linear. Entretanto, ao modelar dessa forma, não estamos restringindo o valor que a variável resposta pode assumir; logo, valores estranhos podem aparecer, como no gráfico.



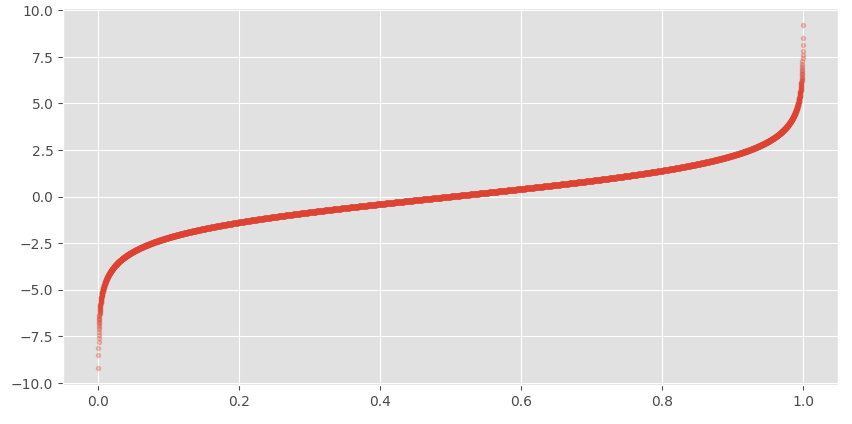
O eixo Y indica se a pessoa sobreviveu (1) ou não (0). Já o eixo X indica o valor pago pela cabine. É possível perceber que quanto maior o valor pago pela cabine, maior a chance de sobreviver. Porém, chance é sinônimo de probabilidade (estatísticos, me perdoem), que é um valor entre 0 e 1. Dessa forma, o ponto azul na extremidade superior direita nos dá uma intuição de algo errado, afinal probabilidade de sobrevivência de 1,6 não existe.

## **Transformação**

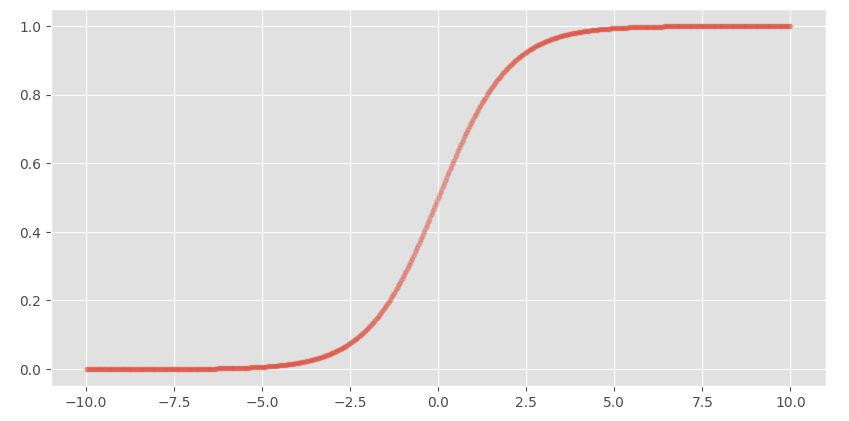
### Logística

A variável resposta predita ao utilizar Regressão Linear para modelar o problema estará vaga, num espectro de valores infinitamente amplo. Nesse cenário de variável resposta binária, se faz necessária a aplicação de algumas transformações para que possamos ajustar um modelo de forma linear.

Para isso, utilizaremos o logito (logit), que é uma função que mapeia a probabilidade de pertencimento a uma classe com amplitude de ±∞±∞, ao invés de 0 e 1. Matematicamente, é uma função do tipo f(x)=log(x1−x)f(x)=log⁡(x1−x), com o gráfico descrito abaixo. Seu gráfico está descrito na imagem a seguir e podemos vê-la sendo limitada com assíntotas verticais em 0 e 1.



Porém, na forma como o gráfico está descrito, o que está sendo limitado é o eixo X, que **no padrão seguido nas aulas** é a nossa variável explicativa. Entretanto, o que precisamos é que a nossa variável resposta, eixo Y, esteja limitada entre 0 e 1. Dessa forma, precisamos do inverso do logito, que é a função logística, dada por f(x)=ex1+e−xf(x)=ex1+e−x. Aplicando a função logística, temos:



Como queremos que nossa variável reposta seja uma probabilidade pp entre 0 e 1 (redundância proposital), podemos aplicar a função logística na função de Regressão Linear, obtendo:

y=p=11+e−(β0+β1x1+β2x2+...+βnxn)y=p=11+e−(β0+β1x1+β2x2+...+βnxn)

### Chances (Odds)

Porém, seria interessante que mantivéssemos a equação em um formato linear, como é na Regressão Linear, afinal, única coisa que queremos modificar é a resposta. Para tirar a função exponencial do denominador da função, usaremos chances (OddsOdds) ao invés de probabilidades. Chamaremos de OddsOdds a proporção entre sobrevivência (1) e não sobrevivência (0), ou, em outras palavras, a probabilidade de um evento ocorrer sobre a probabilidade dele não ocorrer. Representamos da seguinte forma:

Odds(Y=1)=p1−pOdds(Y=1)=p1−p

Isolando o pp na OddsOdds, temos:

Odds=p1−p⇒(1−p)×Odds=p⇒Odds−(p×Odds)=pOdds=p1−p⇒(1−p)×Odds=p⇒Odds−(p×Odds)=p

⇒Odds=p+(p×Odds)⇒Odds=p×(1+Odds)⇒Odds=p+(p×Odds)⇒Odds=p×(1+Odds)

p=Odds1+Oddsp=Odds1+Odds

Fazendo as devidas manipulações algébricas, chegamos em:

log(Odds)=β0+β1x1+β2x2+...+βnxnlog⁡(Odds)=β0+β1x1+β2x2+...+βnxn

Dessa forma, estamos mapeando em uma probabilidade pp qualquer valor (−∞,∞)(−∞,∞), usando um modelo linear para prever essa probabilidade. Por sua vez, podemos mapear esse valor para definir a sobrevivência ou não sobrevivência a partir de um ponto de corte: qualquer valor de probabilidade acima de 0.5, por exemplo, pode ser definido como sobrevivência.

## Aplicação

Agora, aplicando uma Regressão Logística aos dados do Titanic, temos:

from sklearn.linear\_model import LogisticRegression

x = df.Fare.values.reshape(-1, 1)

y = df.Survived

lr = LogisticRegression()

lr.fit(x, y)

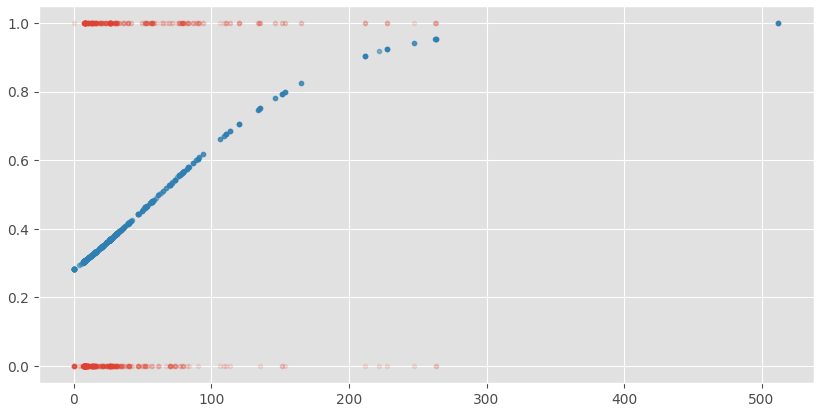
y\_pred = lr.predict\_proba(x)[:, 1]

plt.figure(figsize = (10, 5), dpi = 100)

plt.plot(x, y, '.', alpha = 0.1)

plt.plot(x, y\_pred, '.', alpha = 0.5)

plt.show()



Vale salientar que a resposta da equação é o logarítimo das chances, e não uma probabilidade em si. Por "sorte", o sklearn já faz as transformações necessárias ao usar o método .predict\_proba().

## Estimação e Interpretação do Modelo

Uma vantagem da regressão logística é a sua rápida estimação em novos casos, sem necessidade de grandes cálculos, afinal, basta usarmos as variáveis explicativas, multiplicarmos pelo seus respectivos coeficientes e fazer a transformação em probabilidade.

Outra grande vantagem é em relação à interpretação do modelo. Ora, a OddsOdds é a chance de um evento ocorrrer sobre as chances de ele não ocorrrer. Logo, se Odds=2Odds=2, podemos interpretar como Odds=21=chance do evento ocorrrerchance do evento não ocorrrerOdds=21=chance do evento ocorrrerchance do evento não ocorrrer, e temos que a chance do evento ocorrer é duas vezes maior que ele não ocorrer.

Podemos checar os coeficientes da equação utilizando lr.coef\_.

Até agora estamos modelando as chances/probabilidades de sobrevivência em termos do valor pago na cabine. Porém, da mesma forma que na Regressão Linear, na Regressão Logística também podemos fazer uma análise multivariada e modelar com NN variáveis. Dessa forma, a vantagem da interpretabilidade do modelo fica ainda mais forte.

# ****Métricas de avaliação****

Como podemos mensurar a qualidade do modelo? Ou, se treinarmos vários modelos diferentes, como determinar qual é o melhor?

Para responder a estas perguntas, utilizamos as **métricas de avaliação** de modelos de classificação.

Existem diversas formas de realizar tais avaliações, cada uma observando o problema de um ponto de vista diferente.

## Acurácia

Também conhecida como **taxa de acerto**, essa medida de desempenho traz a proporção de acertos sobre o total de observações.

Assumindo que, dado um conjunto de features xx associadas a um conjunto de target yy um modelo MM foi treinado, temos que a acurácia do modelo MM (ac(M)ac(M)) pode ser descrita matematicamente como:

ac(M)=1n∑i=1nI(yi=M(xi))ac(M)=1n∑i=1nI(yi=M(xi))

A taxa de acerto é um número **limitado entre 0 e 1**. Quanto maior for o seu valor, mais acurado é o modelo MM.

De forma similar, podemos obter a **taxa de erro** com:

err(M)=1−ac(M)err(M)=1−ac(M)

Nesse caso, quanto menor a taxa de erro, mais acurado é o modelo MM.

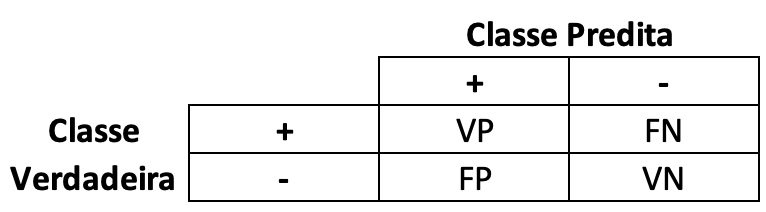
## Matriz de Confusão

Uma alternativa para visualizar o desempenho de um modelo é analisar sua **matriz de confusão**, que ilustra o número de predições corretas e incorretas para cada classe do modelo.

* As **linhas** dessa matriz representam as classes verdadeiras;
* As colunas representam as classes preditas pelo modelo.

Logo, casa elemento mijmij de uma matriz de confusão MMCMMC apresenta o número de exemplos da classe ii classificados como classe jj.

Dessa forma, os elementos na **diagonal principal** indicam as classificações feitas de forma **correta**, enquanto os outros elementos são os classificados de forma **incorreta**.



Por meio dela, temos as medidas quantitativas de quais classes possuem maior dificuldade de serem corretamente classificadas, se existe alguma "confusão" recorrente entre duas classes e mais uma série de medidas quantitativas sobre o modelo, como veremos a seguir.

### Medidas de Desempenho Derivadas da Matriz de Confusão

Dada a matriz de confusão mostrada no item anterior, podemos extrair, entre outras, as seguintes medidas de desempenho:

#### **Precisão**

É a proporção de **exemplos positivos classificados corretamente** entre todos aqueles preditos como positivos pelo modelo MM.

prec(M)=VPVP + FPprec(M)=VPVP + FP

Pode ser vista como uma medida de **exatidão** do modelo.

Uma precisão de 1 para uma determinada classe C1C1 significa que cada item predito como pertencene a essa classe de fato pertence ela; porém, não nos trás informações sobre as predições das classes C2C2.

#### **Sensibilidade**

É a **taxa de acerto na classe positiva**, também conhecida como **revocação** ou **taxa de verdadeiros positivos (TVP)**.

sens(M)=VPVP + FNsens(M)=VPVP + FN

TVP(M)=sens(M)TVP(M)=sens(M)

Pode ser vista como uma medida de **completude** do modelo.

Uma sensibilidade de 1 para uma determinada classe C1C1 significa que todos os itens que deveriam ser previstos como tal, de fato foram; mas não nos trás informações sobre as outras predições erradas dentro da própria classes C2C2.

#### **Especificidade**

É a **taxa de acerto na classe negativa**, sendo o complementar a taxa de falsos positivos (TFP).

esp(M)=VNVN + FPesp(M)=VNVN + FP

TFP(M)=1−esp(M)TFP(M)=1−esp(M)

#### **Generalização para problemas multiclasse**

Essas medidas podem facilmente ser expandidas para problemas de classificação não binários ao considerar cada classe como positiva em relação ao conjunto das demais classes, sendo obtido um valor de desempenho para cada classe.

## F-Score

Uma forma de unificar a exatidão (precisão) e completude (sensibilidade) de um modelo é por meio do cálculo do **F-Score**, que é uma **média harmônica ponderada da precisão e sensibilidade**, dada por:

Fw(M)=(w+1)×sens(M)×prec(M)sens(M)+w×prec(M)Fw(M)=(w+1)×sens(M)×prec(M)sens(M)+w×prec(M)

No geral, é comum usar w=1w=1, dando o mesmo grau de importância para a precisão e sensiblidade. Dessa forma, temos o **F1-Score**, dado por:

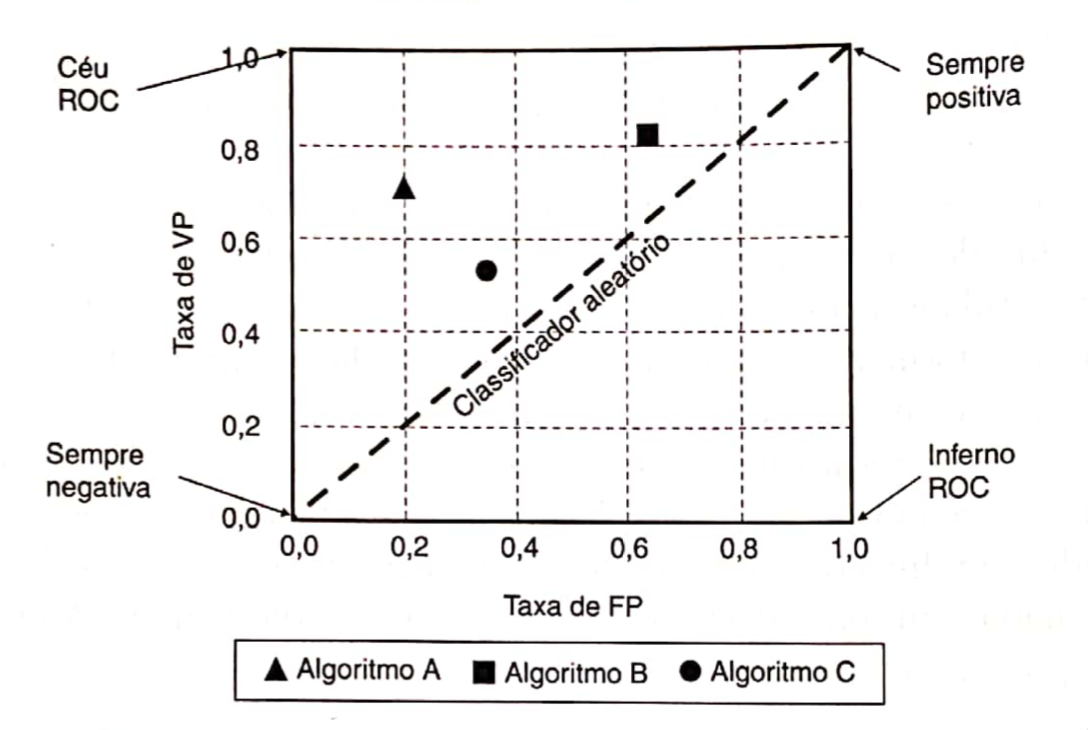
F1(M)=2×sens(M)×prec(M)sens(M)+prec(M)F1(M)=2×sens(M)×prec(M)sens(M)+prec(M)

## Análise ROC e Coeficiente de Gini

Uma forma alternativa e comum de avaliar classificadores em problemas binários é por meio do uso das curvas ROC (Receiving Operating Characteristics).

Seu gráfico é bidimensional: **no eixo X está a TFP e no Y a TVP.**

Na próxima figura, temos um exemplo desse tipo de análise:

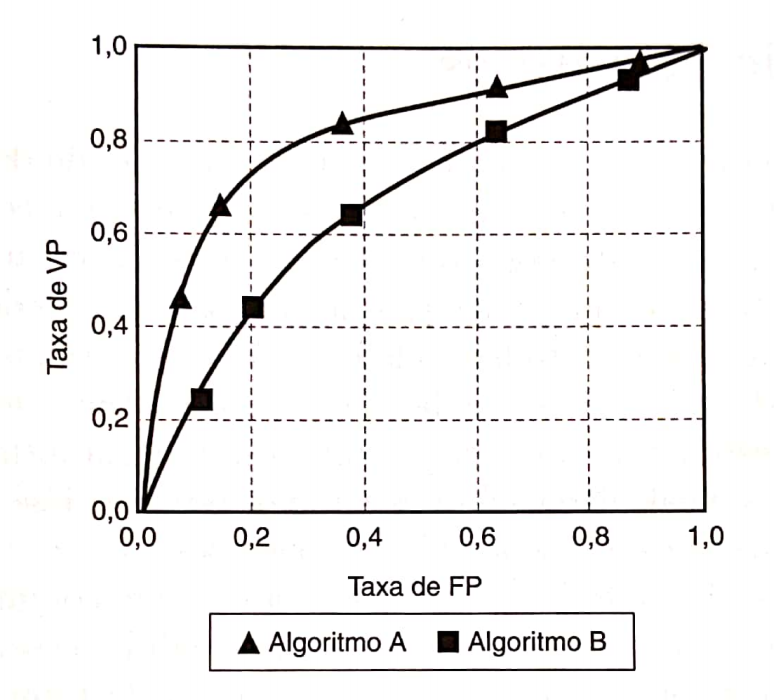


Se um modelo se encontra na diagonal, dizemos que ele possui comportamento similar ao lançamento de uma moeda não viciada.

Modelos **abaixo dessa linha** são **piores que o aleatório**, enquanto que **acima** são modelos **melhores que o aleatório**.

* Se um modelo está na **ponta superior esquerda** (chamada de céu ROC), dizemos que é um modelo perfeito;
* Se está na **ponta superior direita** ou **inferior esqueda**, o modelo sempre classificará novos itens como positivos ou negativos, respectivamente;
* Se está na **ponta inferior direita** (chamada de inferno ROC), esse modelo estará sempre errando.

Tomemos o seguinte exempplo: apesar da variável resposta ser binária em uma Regressão Logística, sua resposta é dado em um valor contínuo entre 0 e 1, que depois é aplicado um limiar de corte para definir se aquele caso pertence a classe positiva ou negativa; logo, temos um valor de TVP e TFP para cada ponto limiar, gerando assim uma curva para cada modelo de classificação, no formato das curvas na próxima figura:



Quando **não há interseções entre as curvas de dois modelos**, significa que o modelo que possui sua curva mais próxima do céu ROC é o que oferece melhor desempenho.

Ao **existir cruzamentos**, cada um terá um desempenho melhor que o outro de acordo com a região.

Entretanto, o mais comum é a determinação da **área abaixo da curva ROC (AUC-ROC)** para cada modelo e compará-los com essa medida única, que é compreendida enre 0 e 1:

* Valores próximos de 1 são considerados os melhores;
* Valores próximos a 0,5 são considerados aleatórios.

A AUC-ROC trás duas grandes vantagens:

* Análise única independente do limiar;
* Robustez contra o desbalanceamento.

Uma outra medida bastante comum, especialmente em econometria, é o **coeficiente de Gini**.

Originalmente, esta medida foi criada para calcular a distribuição da renda de uma determinada população, mas pode ser aplicado também em análises de modelos preditivos por meio da fórmula:

Gini=(2×AUC)−1

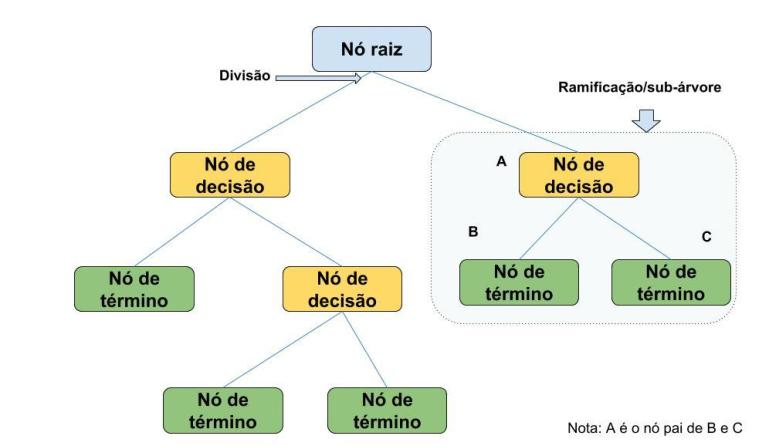
# Árvore de Decisão

## Introdução

Da mesma forma que uma uma regressão logística, uma árvore de decisão representa uma função que toma como entrada uma série de atributos/features e retorna uma **decisão**: um valor de saída único.

Uma boa analogia é com os manuais de conserto de eletrodomésticos dos anos 90: se uma situação A acontecer, você seguirá os passos descritos pelo caminho A+; caso contrário, seguirá pelo caminho A-; cada caminho levará a diferentes verificações e, consequentemente, a diferentes conclusões. chegando à sua conclusão por meio da execução de uma série de testes, em geral de "verdadeiro-falso". De forma simples e direta, é um conjunto de regras if-then-else, sendo de fácil interpretação e implementação.

Olhando para o exemplo da imagem a seguir, podemos ilustrar essa sequência: a raiz e cada nó interno na árvore correspondem ao testes de um atributo de entrada do modelo; as folhas representam a decisão para esse caminho que foi percorrido.



Porém, os testes que ocorrem nos atributos não são aleatórios: devem ser entendidos como funções de custo que buscam otimizar a homogeneidade de suas subpopulações. Queremos fazer com que cada subpopulação de cada nó esteja cada vez mais homogêneo possível, até chegar a folha, usando para isso uma estratégia de dividir para conquistar.

Vamos tomar como base uma amostra da tabela do Titanic, a qual pode ser gerada utilizando df.sample(15, random\_state = 1).

| **PassengerId** | **Survived** | **Sex** | **Parch** |
| --- | --- | --- | --- |
| 863 | 1 | female | 0 |
| 224 | 0 | male | 0 |
| 85 | 1 | female | 0 |
| 681 | 0 | female | 0 |
| 536 | 1 | female | 2 |
| 624 | 0 | male | 0 |
| 149 | 0 | male | 2 |
| 4 | 1 | female | 0 |
| 35 | 0 | male | 0 |
| 242 | 1 | female | 0 |
| 795 | 0 | male | 0 |
| 3 | 1 | female | 0 |
| 7 | 0 | male | 0 |
| 18 | 1 | male | 0 |
| 369 | 1 | female | 0 |

Intuitivamente, observamos que entre Sex e Parch, a primeora é uma variável que discrimina melhor a chance de sobrevivência (Survived). Isso porque dentro das possibilidades de Sex, temos que quando essa variável é igual a male, a probabilidade de sobrevivência é igual a 0.14, e quando igual a female, a probabilidade de sobrevivência é igual a 0.87. Em contrapartida, quando Parch igual a 2, tempos uma probabilidade de sobrevivência igual a 0.54, enquanto que quando Parch igual a 0, a probabilidade de sobrevivência é igual a 0.5. Ou seja, Sex discrimna melhor a probabilidade de sobrevivência.

## Entropia e Ganho de Informação

Em árvores de decisão, estamos buscando separar as instâncias de forma a obter o máximo de homogeneidade possível nas folhas, observano a variável resposta. Para chegar nesse ponto, olhamos para o conjunto de dados, escolhemos uma variável e um limiar e dividimos em dois subconjuntos baseados nisso. A partir daí podemos repetir o passo anterior para cada subconjunto, e para cada subconjunto gerado nos subconjuntos, fazendo um processo recursivo até que tenhamos nosso conjunto de dados perfeitamente separado.

Porém, precisamos escolher um critério para escolher uma variável e limiar para separar em dois subconjuntos. Para isso, usaremos um conceito da Teoria da Informação chamado Entropia, que mede a aleatoriedade de uma variável aleatória, sendo dada por:

H(VA)=−∑ipi×log2piH(VA)=−∑ipi×log2⁡pi

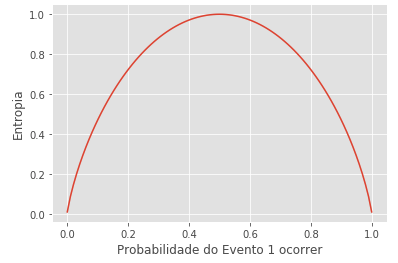
Exemplificando com uma moeda MM não viciada, podemos citar dois eventos:

* 1, cara, com probabilidade p=0.5p=0.5
* 2, coroa, com probabilidade p=0.5p=0.5

Dessa forma, a entropia H(M)H(M) é:

H(M)=−(0.5×log20.5+0.5×log20.5)=1H(M)=−(0.5×log2⁡0.5+0.5×log2⁡0.5)=1

Logo, a entropia é máxima (casos binário), significando que essa variável possui um alto nível de aleatoriedade. O gráfico da equação mostrada é:



Em casos binários, no qual uma variável aleatória assume o valor 0 ou 1, podemos escrever a entropia como

H(VA)=B(pp+n)=−[p×log2p+(1−p)×log2(1−p)]H(VA)=B(pp+n)=−[p×log2⁡p+(1−p)×log2⁡(1−p)]

O que queremos obter com a divisão do conjunto de dados são subconjuntos que diminuam ao máximo a aleatoriedade da variável resposta do conjunto original. Logo, para cada atributo do conjunto de dados, queremos calcular o **Ganho de Informação** que mede a redução na entropia das partições obtidas de acordo os valores desse atributo. Isso é dado pela diferença entre a entropia do conjunto de exemplos original e a soma ponderada da entropia das partições, sendo a construção da árvore orientada pelo objetivo de reduzir a entropia.

Ou seja, um atributo AA com dd valores possíveis para ele gera SS subconjuntos. Cada subconjunto SkSk possui pkpk exemplos positivos e nknk exemplos negativos, nos dando uma entropia de

B(pkpk+nk)B(pkpk+nk)

Ponderando pela quantidade de exemplos dos subconjuntos, temos um conteúdo de informação esperado do atributo AiAi de

E(Ai)=∑ipk+nkp+kB(pkpk+nk)E(Ai)=∑ipk+nkp+kB(pkpk+nk)

O Ganho de Informação do atributo AiAi é a redução na entropia do estado anterior para o estado posterior, na forma

Ganho(Ai)=B(pp+n)−E(Ai)Ganho(Ai)=B(pp+n)−E(Ai)

Podemos definir uma função com a seguir para calcular o ganho de informação. Nela, o parâmetro df é o Data Frame a ser lido; col é o atributo no qual estamos calculando o ganho de informação; e v\_resp é em sobre qual variável resposta que estamos querendo verificar a redução de entropia.

def entropia\_binaria(p):

if p == 0 or p == 1:

return 0

n = 1 - p

return -(p\*np.log2(p) + n\*np.log2(n))

def ganho\_informacao(df, col, v\_resp):

# Armazena todas as possibilidades

seq\_poss = df[col].unique()

# Calcula a entropia inicial

ent\_inicial = entropia\_binaria(df[v\_resp].value\_counts(1)[0])

# Separa os conjuntos baseado nas possibilidades de valor do atributo

conj = [df[df[col] == i] for i in seq\_poss]

# Calcula a entropia binária da variável resposta em cada conjunto

ent = [entropia\_binaria(i[v\_resp].value\_counts(1)[0]) for i in conj]

# Calcula a quantidade de elementos em cada conjunto para fazer a

# média ponderada

n\_elem = [i.shape[0] for i in conj]

total = sum(n\_elem)

# Caula a média ponderada e retorna o ganho de informação

ent\_atr = 0

for i in range(len(conj)):

ent\_atr += ent[i]\*n\_elem[i]/total

return ent\_inicial - ent\_atr

## **Implementações Práticas**

Podemos utilizar a implementação pronta do pacote scikit-learn, que está dentro do módulo tree. Com o modelo instanciado em uma variável, podemos ajustá-lo aos dados usando o método .fit(), e fazer as predições com o .predict(), lembrando de deixar todos os dados no formato numérico.

from sklearn.tree import DecisionTreeClassifier

arv = DecisionTreeClassifier()

x = df\_amostra.loc[:, ['Pclass',

'Sex',

'SibSp',

'Parch',

'fare\_categorizado']]

y = df\_amostra.loc[:, 'Survived']

x.Sex = x.Sex.apply(lambda x: 1 if x == 'male' else 0)

x.fare\_categorizado.replace({'medio': 0,

'baixo': -1,

'alto': 1}, inplace = True)

arv.fit(x, y)

y\_pred = arv.predict(x)

Podemos veriricar a acurácia do modelo na forma:

from sklearn.metrics import accuracy\_score

accuracy\_score(y, y\_pred)

Podemos também, ao invés de verificar a classe predita usando a moda, podemos enxergar a probabilidade de uma determinada instância pertencer a uma classe ou outra utilizando o .predict\_proba(). A partir daí, também podemos calcular o AUC ou Gini, já que estamos tratando de um caso binário.

from sklearn.metrics import roc\_auc\_score

y\_pred\_proba = arv.predict\_proba(x)

print('AUC:', roc\_auc\_score(y, y\_pred\_proba[:, 1]))

print('Gini:', 2\*roc\_auc\_score(y, y\_pred\_proba[:, 1]) - 1)

Além disso, podemos também ter uma ótima forma de visualização da árvore utilizando o graphviz. Dessa forma, conseguimos interpretar com perfeição o modelo, entendo onde ocorrem as quebras, quais são as mais importantes e qual a hierarquia entre elas.

import sklearn.tree as tree

import pydotplus

from sklearn.externals.six import StringIO

from IPython.display import Image

dot\_data = StringIO()

tree.export\_graphviz(arv,

out\_file = dot\_data,

class\_names = ['MORREU', 'SOBREVIVEU'],

feature\_names = x.columns.to\_list(),

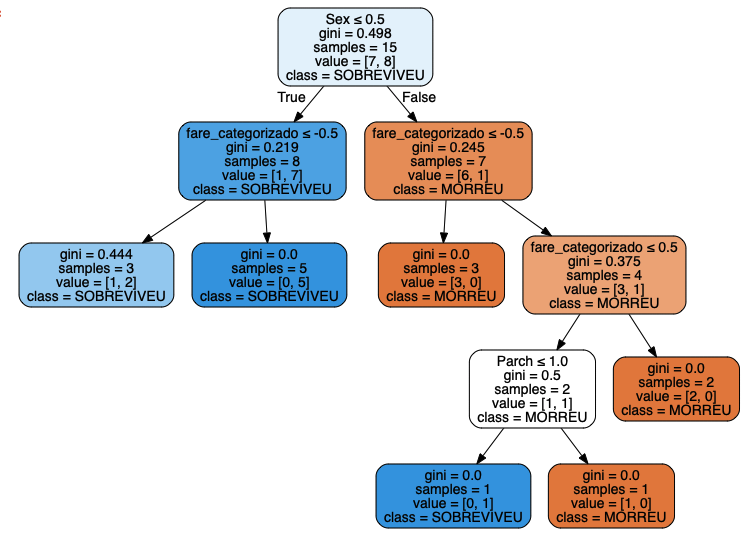
filled = True,

rounded = True,

special\_characters = True)

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

Image(graph.create\_png())

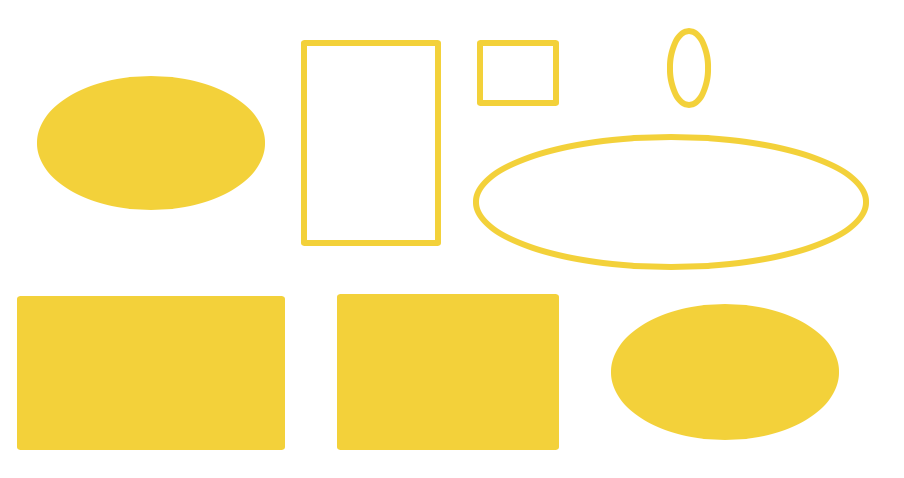


# Algoritmos de Clusterização

## Introdução

Existem situações que o Cientista de Dados se depara que o foco é segmentar os dados. Por exemplo, é necessário identificar quais são os grupos distintos de clientes que existem dentro de uma empresa, para criar estruturas de marketing específicas para cada grupo desse. Então, algoritmos de agrupamento podem ser utilizados para realizar tal modelagem. O objetivo desse tipo de técnica é encontrar estruturas de clusters nos dados em que os objetos pertencentes a cada cluster compartilham característica ou propriedades relevantes para o domínio do problema em estudo.

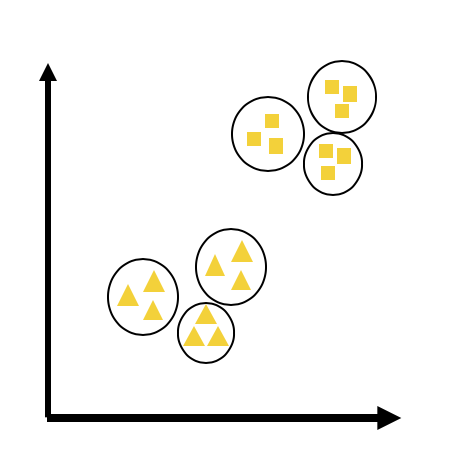
O mesmo conjunto de dados pode ser dividido em N formas diferentes, capturando em cada forma dessa caraterísticas específicas. Ao tomar como exemplo a figura a seguir, podemos segmentar as figuras geométricas por (i) forma, (ii) preenchimento e (iii) forma e preenchimento.



Dessa forma, os agrupamentos realizados em um mesmo conjunto de dados podem ser diferentes a depender das variáveis usadas ou por algoritmo utilizado para criação dos clusters. Existem 3 grandes critérios sobre os algoritmos de agrupamento:

1. Compactação: os clusters desse tipo são criados baseando-se em uma variação intracluster pequena, criando, em sua maioria, grupos de formatos esféricos.
2. Encadeamento: baseia-se na ideia de que objetos vizinhos devem compartilhar o mesmo cluster, sendo assim bastante efetivo na identificação de formatos arbitrários de clusters.
3. Separação Espacial: considera as distâncias entre os clusters, fornecendo pouca orientação durante o processo de agrupamento e normalmente levando a soluções triviais.

É importante salientar também diversos tipos de refinamento podem ser feitos no processo de determinação dos clusters. Na figura a seguir, por exemplo, podemos claramente ter 2 grupos distintos: o grupo dos quadrados, no canto superior direito; e o grupo dos triângulos, no canto inferior esquerdo. Porém, podemos também identificar 6 grupos separadas espacialmente, identificados pelos círculos pretos. Esse tipo de refinamento dependerá da configuração dos hiperparâmetros escolhidos.



## Métricas de Proximidade

Em uma modelagem por agrupamento, os passos a serem seguidos são parecidos com o que é feito em uma modelagem supervisionada, ou seja, partimos do entendimento do problema; depois vamos a preparação dos dados; aplicamos o algoritmo; e validamos os clusters criados. Porém, dois passos a mais são importantes: o primeiro se refere a necessidade de entendimentos dos clusters. Ora, não basta separar os grupos, é fundamental a interpretação do significado de cada cluster encontrado, sendo de grande importância o apoio de um especialista no cenário que está sendo tratado. O outro passo importante é a escolha da métrica de proximidade a ser utilizada. Todos os algoritmos de agrupamento utilizam técnicas de proximidade, sendo possível inclusive a implementação do mesmo algoritmo utilizando técnicas distintas.

### Atributos Quantitativos

As métricas mais utilizadas em atributos quantitativos são as referentes à distância de Minkowski, dada por

d(x¯i,x¯j)=(∑l=1d(|xli−xlj|)p)1pd(x¯i,x¯j)=(∑l=1d(|xil−xjl|)p)1p

Ao utilizar p=1p=1, temos a Distância de Manhattan; com p=2p=2, temos a famosa distância euclidiana.

Outras duas métricas bastante utilizadas para atributos quantitativos são o cosseno e correlação de Pearson.

cosseno(x¯i,x¯j)=∑dl=1xlixlj∑dl=1(xli)2∑dl=1(xlj)2−−−−−−−−−−−−−−−−√cosseno(x¯i,x¯j)=∑l=1dxilxjl∑l=1d(xil)2∑l=1d(xjl)2

pearson(x¯i,x¯j)=covariância(x¯i,x¯j)variância(x¯i)variância(x¯j)pearson(x¯i,x¯j)=covariância(x¯i,x¯j)variância(x¯i)variância(x¯j)

### Atributos Qualitativos

Em atributos qualitativos, uma métrica bastante utilizada é a distância de Hamming, que em linhas gerais verifica a quantidade de atributos iguais dois objetos compartilham.

h(x¯i,x¯j)=∑l=1da(xli−xlj)h(x¯i,x¯j)=∑l=1da(xil−xjl)

a(xli−xlj)={1,se xli≠xlj0,se xli=xlja(xil−xjl)={1,se xil≠xjl0,se xil=xjl

## K-Means

O K-Means é um dos algoritmos de agrupamentos mais tradicionais. Seu objetivo é separar os exemplos em N grupos com igual variância, sendo necessária a especificação da quantidade de grupos a serem criados. De forma prática, existem 3 passos no algoritmo:

1. Escolha aleatória de N exemplos do conjunto de dados que serão os centróides iniciais;
2. Por meio de uma métrica de proximidade, assimila cada instância do conjunto de dados a um centróide;
3. Cria novos N centróides baseado na média de todos os exemplos associados a um centróide específico.

O passo 1 é feito apenas uma vez, enquanto os passos 2 e 3 são repetidos até que a diferença entre os centróides na iteração tt e os respectivos centróides na iteração t+1t+1 não seja significativa.

O sklearn possui uma implementação desse algoritmo:

kmeans = KMeans(n\_clusters = i, max\_iter = 300, n\_init = 10)

kmeans.fit(X)

Porém, como a quantidade de cluster é um hiperparâmetro, é imporante que seja ajustada. Dessa forma, para fazer o ajuste, podemos avaliar a soma dos quadrados intracluster (WCSS), que é uma medida que avalia a média quadrada das distâncias de todos os pontos do cluster até o centróide, em um gráfico de cotovelo, no qual o eixo x é a quantidadde de clusters e o eixo y o WCSS. Fazendo isso de forma iterativa, temos o seguinte código, gerando a figura na sequência.

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, max\_iter = 300, n\_init = 10)

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 11), wcss)

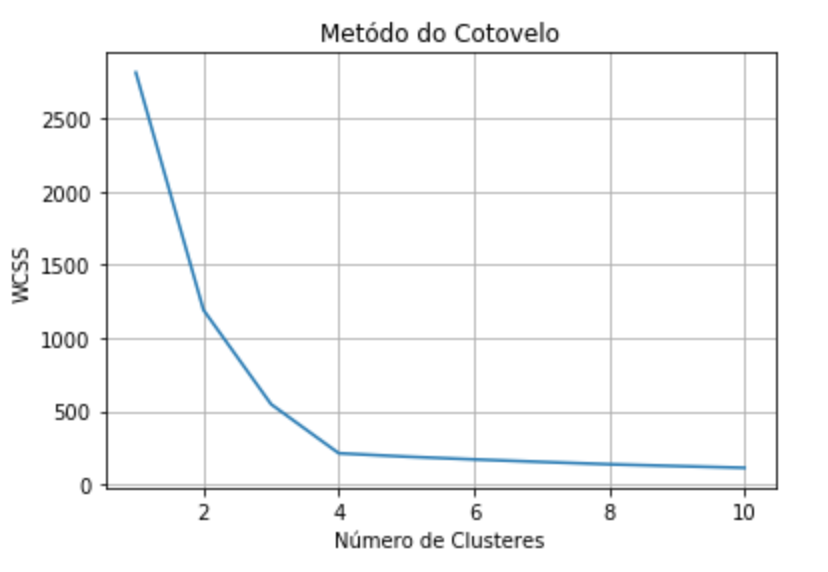
plt.title('Metódo do Cotovelo')

plt.xlabel('Número de Clusters')

plt.ylabel('WCSS')

plt.grid()

plt.show()



Por meio desse gráfico, podemos definir que a quantidade ideal de clusters é 4, dado que nesse ponto, a variação do WCSS passa a ser muito pequena.