

Universidade Federal do Paraná (UFPR)  
Especialização em Engenharia Industrial 4.0

Boas Maneiras  
Aprendizado Não Supervisionado  
Regressão

David Menotti

[www.inf.ufpr.br/menotti/am-182](http://www.inf.ufpr.br/menotti/am-182)

# Hoje

- Boas Maneiras
  - Métricas de Avaliação & Téc. de Validação
  - Avaliação & Comparação de Classificação
- Aprendizado não supervisionado (*Clustering*)
  - k-Means
  - DBScan
- Regressão
  - Linear ( e Múltipla )
  - Não-Linear ( Exponencial / Logística )

# Boas Maneiras

## Agenda

- Introdução
- Métricas de Avaliação
- Técnicas de Validação
- Avaliação de Classificadores
- Comparando Classificadores

# Introdução

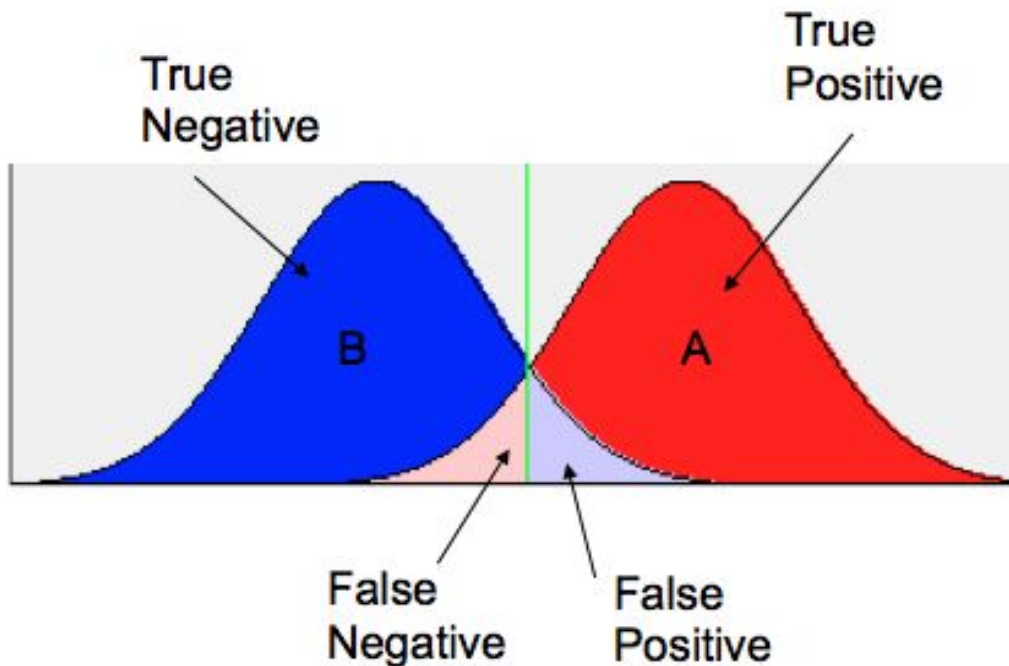
- Considere por exemplo, um problema de classificação binário com 99 exemplos na classe positiva e 1 exemplo na classe negativa.
- Considere que o classificador *classificou* corretamente todos os exemplos da classe positiva e errou somente o exemplo da classe negativa.
- Nesse caso temos uma acurácia de 99%.
- **Atenção:** Acurácia de um classificador nem sempre é a melhor medida para se avaliar um classificador, principalmente em problemas **desbalanceados**.

# Avaliação

- Dado um classificador binário, as saídas possíveis são as seguintes:

		<u>True class</u>	
		<b>p</b>	<b>n</b>
<u>Hypothesized class</u>	<b>Y</b>	True Positives	False Positives
	<b>N</b>	False Negatives	True Negatives
Column totals:		<b>P</b>	<b>N</b>

# Avaliação

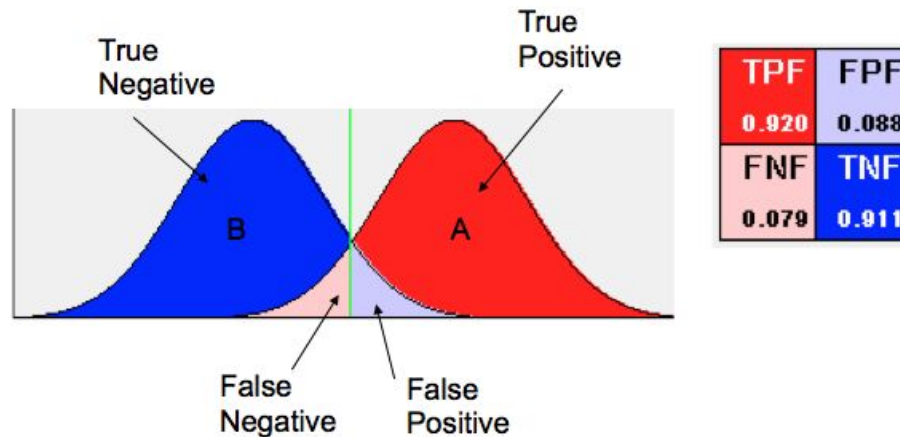


<b>TPF</b>	<b>FPF</b>
<b>0.920</b>	<b>0.088</b>
<b>FNF</b>	<b>TNF</b>
<b>0.079</b>	<b>0.911</b>

TP – Classe é A e classificamos como A  
TN – Classe é B e classificamos como B  
FP – Classe é B e classificamos como A  
FN – Classe é A e classificamos como B

# Tipos de Erro

- **Erro Tipo I** ( $\alpha$ -erro, falso positivo):
  - Aceita-se como genuíno algo que é falso - **FP**
- **Erro Tipo II** ( $\beta$ -erro, falso negativo):
  - Rejeita-se algo que deveria ter sido aceito - **FN**



TP – Classe é A e classificamos como A  
TN – Classe é B e classificamos como B  
FP – Classe é B e classificamos como A  
FN – Classe é A e classificamos como B

# Métricas

- FP Rate
  - $FP / N$

- TP Rate ou hit rate
  - $TP / P$

- Accuracy
  - $(TP + TN) / (P + N)$

- $P = TP + FN$
- $N = TN + FP$

- *Precision* (Pr)
  - $TP / (TP + FP)$

Tende a 1 se FP tende a 0

- *Recall* (R) (revogação)
  - $TPR = TP / (TP + FN)$

Tende a 1 se FN tende a 0

- F-Measure
  - $2 / (1 / Pr + 1 / R)$
  - $2 Pr \cdot R / (Pr + R)$

Média harmônica de Pr e R, tendo em vista que são grandezas inversamente proporcionais



# Métricas

No scikit-learn, essas métricas estão implementadas no método `classification_report` (classe `metrics`)

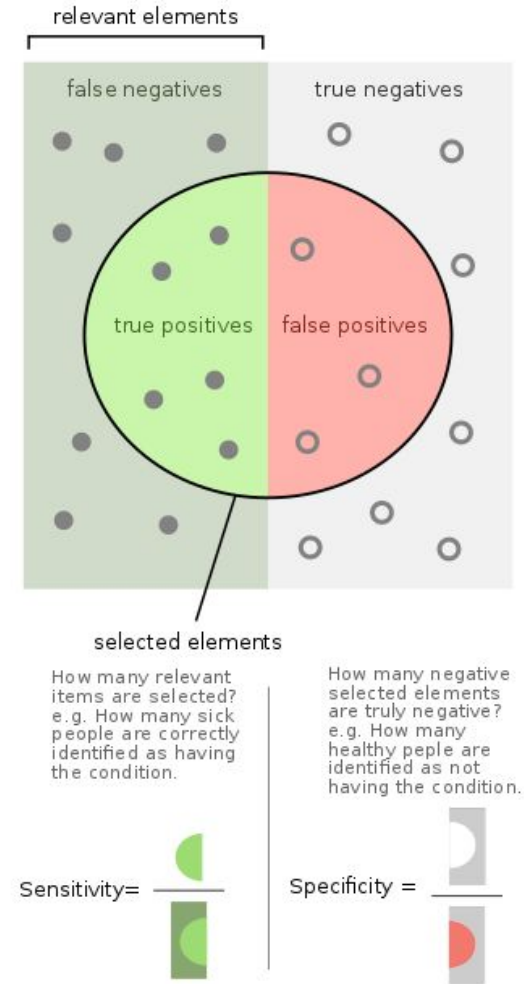
```
>>> from sklearn.metrics import classification_report
>>> y_true = [0, 1, 2, 2, 0]
>>> y_pred = [0, 0, 2, 2, 0]
>>> target_names = ['class 0', 'class 1', 'class 2']
>>> print(classification_report(y_true, y_pred, target_names=target_names))
```

	precision	recall	f1-score	support
class 0	0.67	1.00	0.80	2
class 1	0.00	0.00	0.00	1
class 2	1.00	1.00	1.00	2
avg / total	0.67	0.80	0.72	5

# Métricas

- Especificidade / *Specificity*
  - **TN** / N
- Sensitividade / *Sensitivity*
  - **TP** / P
  - \* *Recall*

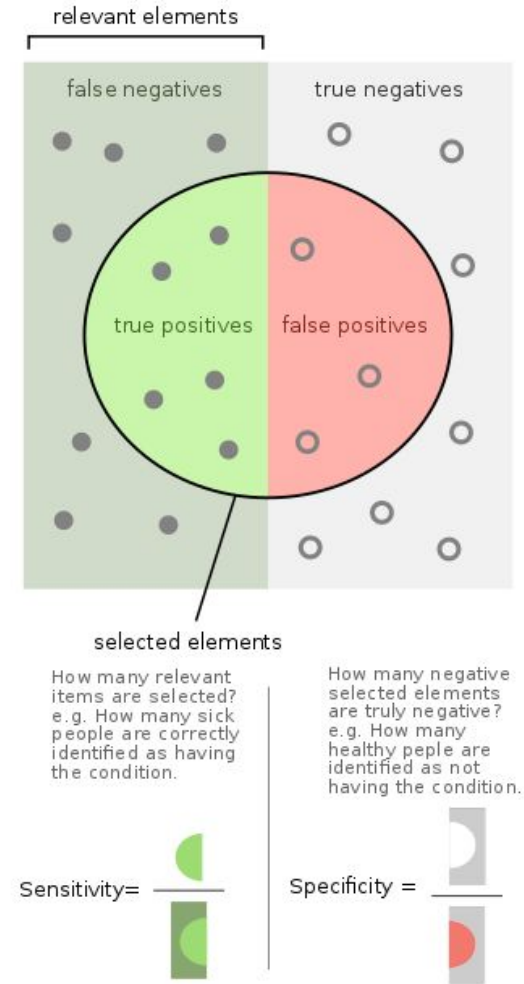
- $P = TP + FN$
- $N = TN + FP$



# Métricas

- Especificidade / *Specificity*
  - **TN** / N
- Sensibilidade / *Sensibility*
  - **TP** / P
  - \* *Recall*

- $P = TP + FN$
- $N = TN + FP$



# Técnicas de Validação

- Resubstitution
- Hold-out
- K-fold cross-validation
- LOOCV
- Random subsampling
- Bootstrapping

# Resubstitution

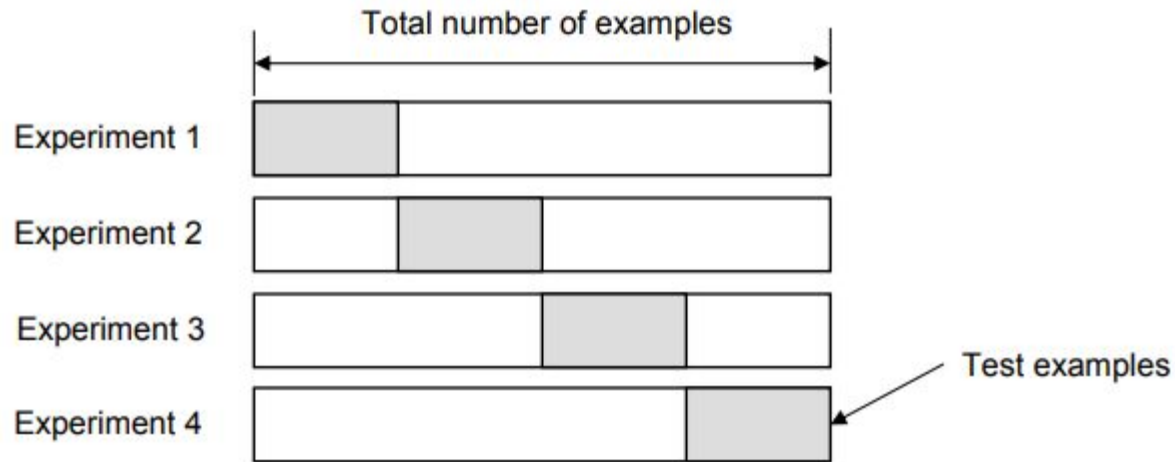
- Se todos os dados forem usados para treinar o modelo e a taxa de erro for avaliada com base no resultado versus valor real do mesmo conjunto de dados de treinamento, esse erro será chamado de **erro de resubstituição** (resubstitution error validation).
- Por exemplo:
  - Análise de *Malware*
  - Aplicações onde necessitam de:
    - *Black list*
    - *White list*

# Hold-out

- Para evitar o erro de resubstituição, os dados são divididos em dois conjuntos de dados diferentes rotulados como um conjunto de treinamento e de teste.
- A divisão pode ser: 60% / 40% ou 70% / 30% ou 80% / 20%, etc.
  - Avaliar classificador em diferentes cenários:
    - 5% de train? ou 90% de train?
- Muito provavelmente haverá distribuição desigual das classes (alvos/metapas) nos conjunto de dados de treinamento e teste.
  - os conjuntos de dados de treinamento e teste são criados com distribuição igual de diferentes classes de dados.
    - Esse processo é chamado de **estratificação**.

# *K-fold cross-validation*

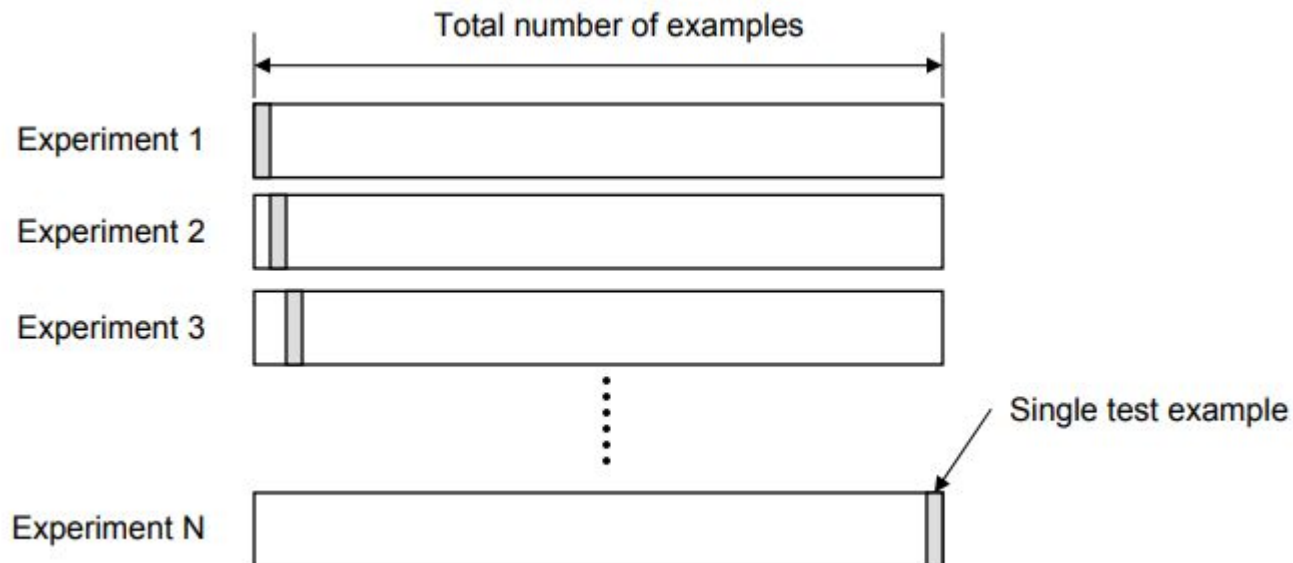
- Nesta técnica,  $k-1$  *folds* são usadas para **treinamento** e o restante (1 *fold*) é usado para **teste**.



- A vantagem é que todos os dados são usados para treinamento.
  - Cada amostra é usada apenas uma vez para **teste**.
- A taxa de erro do modelo é a média
- Pode-se ver esta técnica como um *hold-out* repetido

# *Leave-One-Out Cross-Validation (LOOCV)*

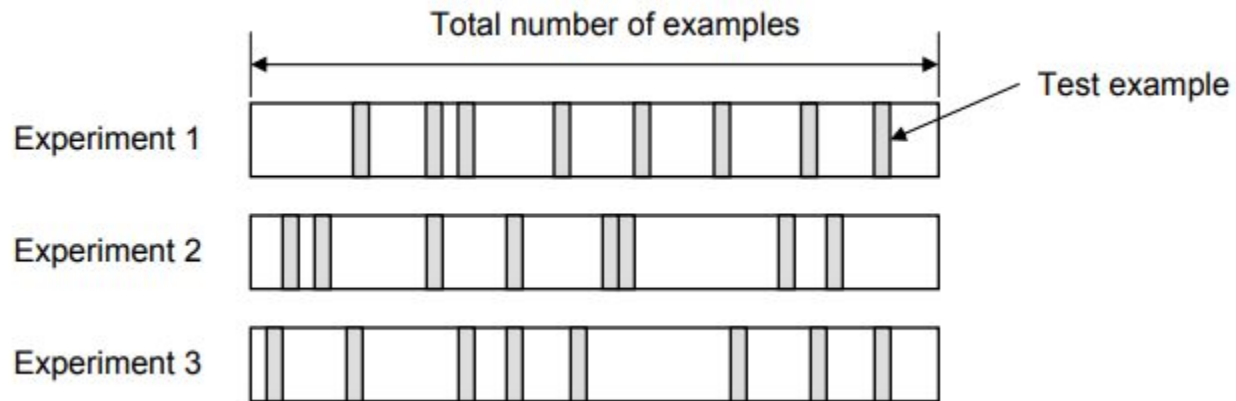
- Todos os dados, exceto um registro, são usados para treinamento
  - Um registro é usado para teste.
- Processo é repetido por N vezes se houver registros N.
  - A vantagem é que  $(N-1)/$  são usados para treinamento
- Desvantagem: custo computacional: N **rodadas**





# Random subsampling

- Amostras são escolhidos aleatoriamente para formar o **test set**.
- Os dados restantes formam o **train set**
- Bem utilizada na prática: pelo menos 30 execuções/rodadas



# Bootstrapping

- Train set é selecionado aleatoriamente com **substituição / repetição**.
- Os exemplos restantes que não foram selecionados para treinamento são usados para teste.
- Diferente da validação cruzada de K-fold,
  - é provável que o valor mude de fold para fold.



# Avaliação de Classificadores

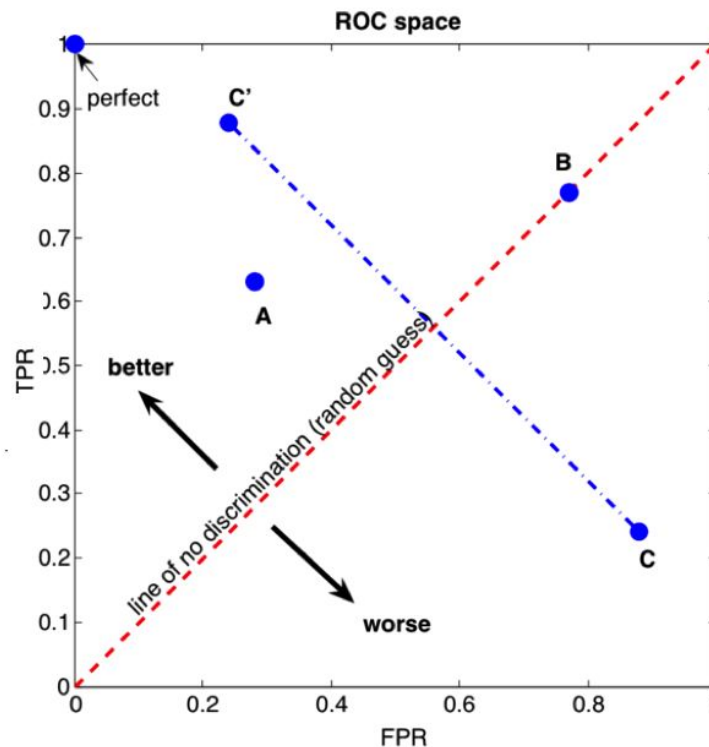
# Exemplos de Classificadores

A			B			C			C'		
TP=63	FN=37	100	TP=77	FN=23	100	TP=24	FN=76	100	TP=76	FN=24	100
FP=28	TN=72	100	FP=77	TN=23	100	FP=88	TN=12	100	FP=12	TN=88	100
91	109	200	154	46	200	112	88	200	88	112	200
TPR = 0.63			TPR = 0.77			TPR = 0.24			TPR = 0.76		
FPR = 0.28			FPR = 0.77			FPR = 0.88			FPR = 0.12		
F1 = 0.66			F1 = 0.61			F1 = 0.22			F1 = 0.81		
ACC = 0.68			ACC = 0.50			ACC = 0.18			ACC = 0.82		

# Espaço ROC

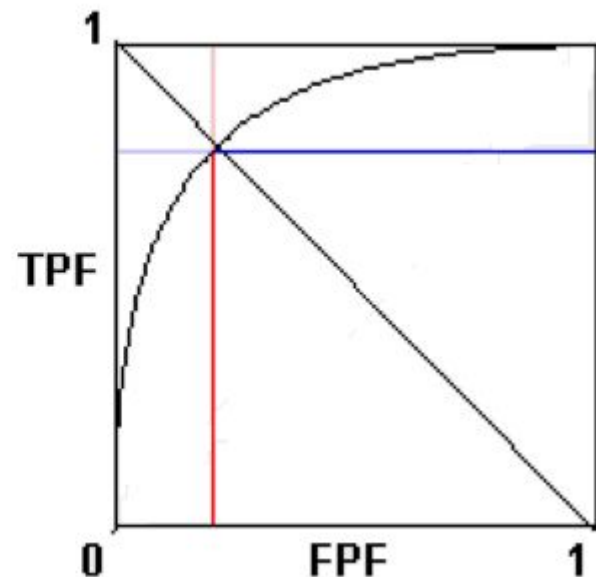
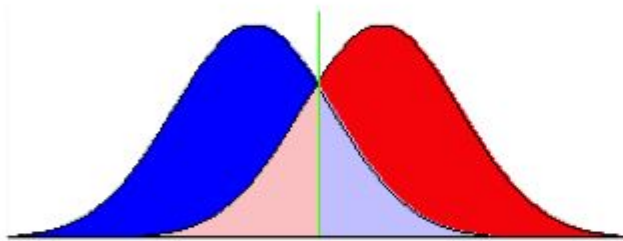
ROC (*Receiver Operating Characteristic*) é uma representação gráfica que ilustra o desempenho de um classificador binário em um determinado ponto de operação.

- Classificadores do exemplo anterior no espaço **ROC**.



# Curva ROC

- A curva ROC mostra todos os pontos de operação do classificador (relação entre TP e FP)
- Para cada ponto, existe um limiar associado.
- EER (*Equal Error Rate*):
  - Ponto no gráfico no qual FPR é igual a 1-TPR
- Quando não existe um ponto operacional específico, usamos EER.



# Curva ROC

- Considere 20 amostras, 10 classificadas corretamente (+) e 10 classificadas incorretamente (-) com suas respectivas probabilidades.

#	Classe	Score	#	Classe	Score
1	+	0.90	11	-	0.70
2	+	0.80	12	-	0.53
3	+	0.60	13	-	0.52
4	+	0.55	14	-	0.505
5	+	0.54	15	-	0.39
6	+	0.51	16	-	0.37
7	+	0.40	17	-	0.36
8	+	0.38	18	-	0.35
9	+	0.34	19	-	0.33
10	+	0.30	20	-	0.10

# Curva ROC

## Exemplo

- Considere 20 amostras, 10 classificadas corretamente (+) e 10 classificadas incorretamente (-) com suas respectivas probabilidades.

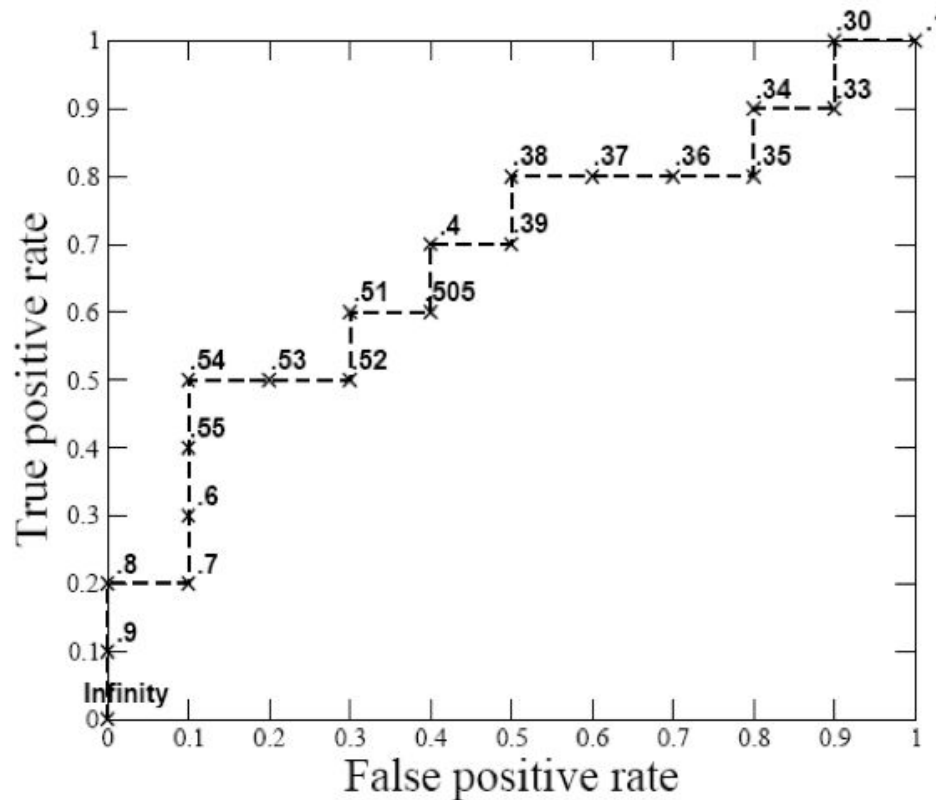
#	Classe	Score	#	Classe	Score
1	+	0.90	11	-	0.70
2	+	0.80	12	-	0.53
3	+	0.60	13	-	0.52
4	+	0.55	14	-	0.505
5	+	0.54	15	-	0.39
6	+	0.51	16	-	0.37
7	+	0.40	17	-	0.36
8	+	0.38	18	-	0.35
9	+	0.34	19	-	0.33
10	+	0.30	20	-	0.10



# Curva ROC

## Exemplo

- Após ordenar os dados usando as probabilidades, temos a seguinte curva ROC



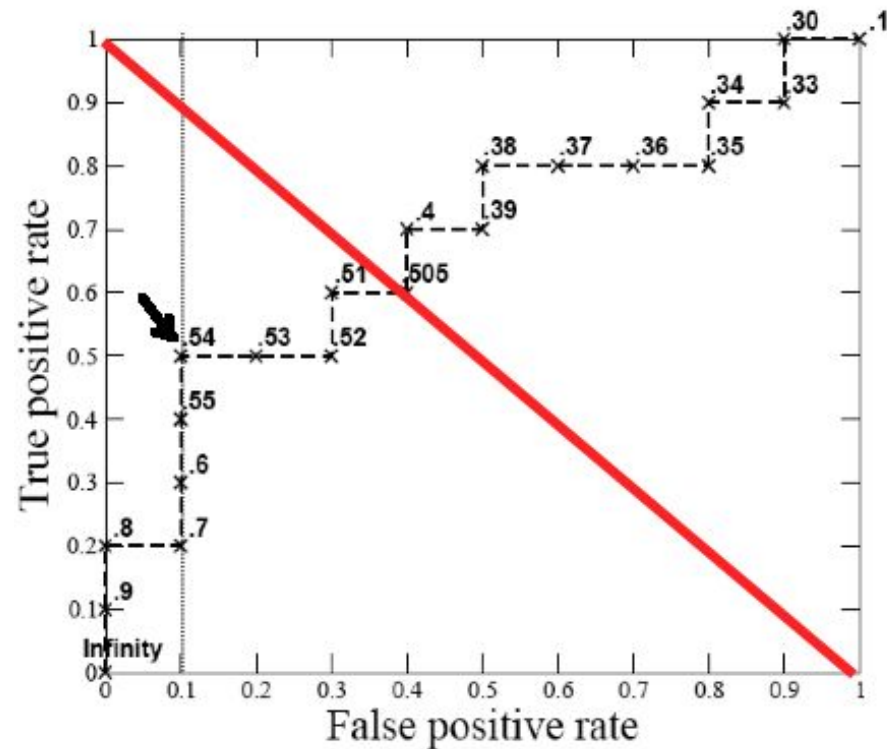
# Curva ROC

## Exemplo

- Suponha que a especificação do seu sistema diga que o máximo **FPR** permitido é de 10%
- Qual seria o ponto de operação do sistema (limiar) ? **[0,7 ; 0,54]**
- Para esse limiar, qual seria a taxa de acerto do sistema?

$$\frac{(\text{Pos} \times \text{TPR}) + (\text{Neg} - (\text{FPR} \times \text{Neg}))}{N}$$

- 70% para um limiar de 0,54



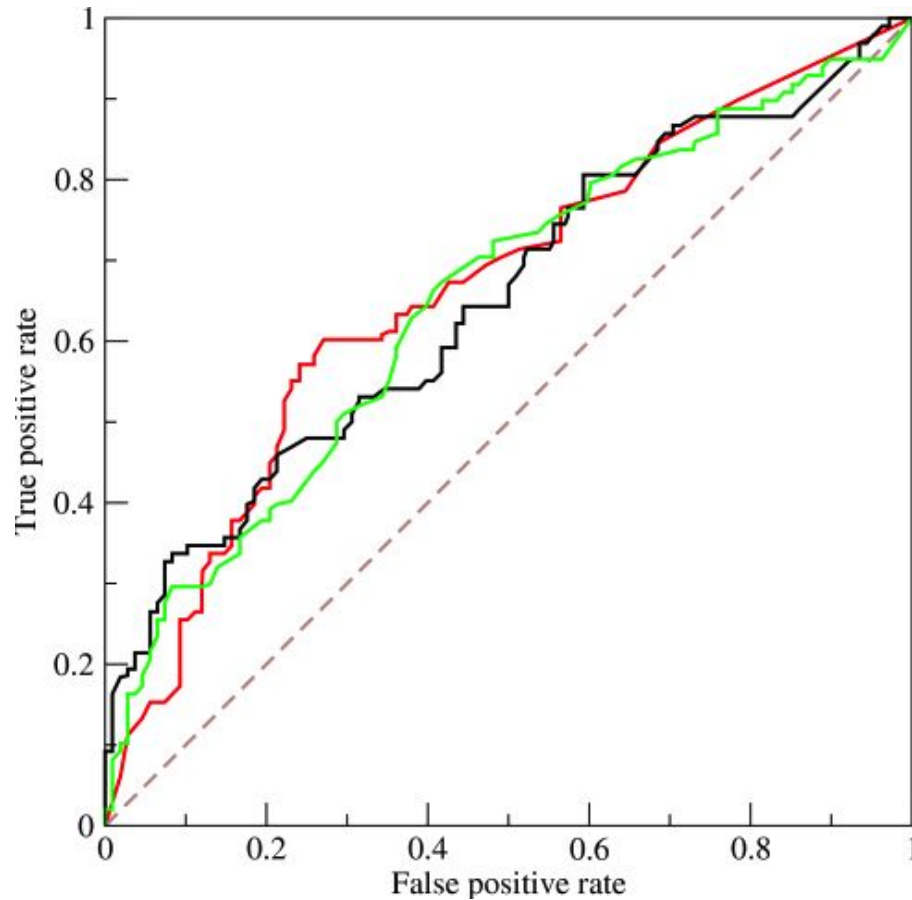
# Curva ROC

## Classes desbalanceadas

- Curvas ROC têm uma propriedade interessante:
  - não são sensíveis a mudanças de distribuição das classes
- Se a proporção de instâncias negativas e positivas na base de teste muda, a curva continua a mesma.
  - A curva é baseada em TP e FP.
- Isso permite uma fácil visualização do desempenho dos classificadores independentemente da distribuição das classes.

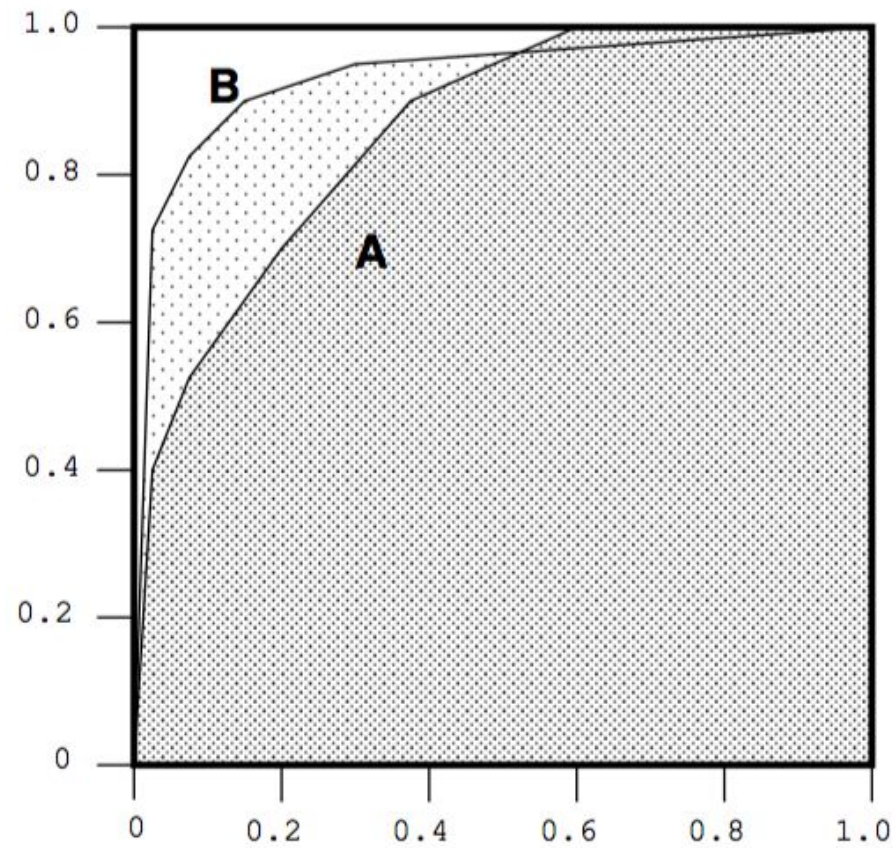
# Comparando Classificadores

Qual seria o melhor classificador?



# AUC

## Area Under the Curve



# Comparando Clasificadores

# Comparando Classificadores

- Suponha que você tenha dois classificadores que produziram as seguintes taxas de erro nos 10 folds de validação da sua base de dados.

$$C_1 = [10, 15, 7, 6, 13, 11, 12, 9, 17, 14]$$

e

$$C_2 = [17, 14, 12, 16, 23, 18, 10, 8, 19, 22]$$

- O erro médio de  $C_1 = 11,4\%$  enquanto o erro médio de  $C_2 = 15,9\%$
- Podemos afirmar que  $C_1$  é melhor do que o  $C_2$ ?
- A média é bastante sensível a outliers.
  - O desempenho muito bom em um fold pode compensar o desempenho ruim em outro.

# Comparando Classificadores

- Uma ferramenta bastante útil nesses casos é o teste de hipóteses.
  - t-Teste
    - Requer que as diferenças entre as duas variáveis comparadas sejam distribuídas normalmente.
    - Tamanho das amostras seja grande o suficiente ( $\sim 30$ ).
- A natureza do nosso problema não atende nenhum dos critério acima.
- Alternativa é o ***Wilcoxon signed-rank test***



# Wilcoxon Signed-Rank test

- Teste não paramétrico que ordena as diferenças de desempenho de dois classificadores, ignorando os sinais, e compara a posição das diferenças (positivas e negativa).
- O teste consiste em:
  - Definir as hipóteses  
( $H_0$  - Não existe diferença e  $H_1$  - Existe diferença)
  - Calcular as estatísticas do teste
  - Definir o nível de significância do teste ( $\alpha$ )
  - Rejeitar ou aceitar  $H_0$ , comparando o valor da estatística calculada com o valor crítico tabelado.
- **Nível de significância:** Em aprendizagem de máquina,  $\alpha = 0,05$  é um valor bastante utilizado. Isso quer dizer que existe 5 chances em 100 da hipótese ser rejeitada quando deveria ter sido aceita, ou seja, uma probabilidade de erro de  $0,05 = 5\%$ .

# Wilcoxon Signed-Rank test

- Considere o exemplo anterior dos classificadores  $C_1$  e  $C_2$

C_1	C_2	Diff
10	17	7
15	14	1
7	12	5
6	16	10
13	23	10
11	18	7
12	10	2
9	8	1
17	19	2
14	22	8

Diferenças	1	1	2	2	5	7	7	8	10	10
	1	2	3	4	5	6	7	8	9	10
Posição	1.5	1.5	3.5	3.5	5	6.5	6.5	8	9.5	9.5
Sinal	+	+	+	-	-	-	-	-	-	-

- Estatísticas:  $N = 10$ ,
- $W^+ = 6,5$  (Soma das posições com sinal +)
- $W^- = 48,5$  (Soma das posições com sinal -)
- Comparar os valores das estatísticas calculadas com os valores tabelados.

# Wilcoxon Signed-Rank test

## Valores críticos bi-caudais para o teste de Wilcoxon

Rejeição de  $H_0$  para  $\sum R^-$  ou  $\sum R^+$  fora do intervalo dado pelos valores delimitados entre Inferior e Superior

		$\alpha$							
		0,1		0,05		0,02		0,01	
número de pares com diferenças não-nulas		Inferior	Superior	Inferior	Superior	Inferior	Superior	Inferior	Superior
	1								
2									
3									
4									
5	0	15							
6	2	19	0	21					
7	3	25	2	26	0	28			
8	5	31	3	33	1	35	0	36	
9	8	37	5	40	3	42	1	44	
10	10	45	8	47	5	50	3	52	
11	13	53	10	56	7	59	5	61	
12	17	61	13	65	9	69	7	71	
13	21	70	17	74	12	79	9	82	
14	25	80	21	84	15	90	12	93	
15	30	90	25	95	19	101	15	105	

- Comparar os valores das estatísticas calculadas ( $W^+ = 6,5$  e  $W^- = 48,5$ ) com os valores tabelados.
- Os valores críticos de  $W$  para  $N=10$  e  $\alpha = 0,05$  são 8 e 47.
- Como os valores calculados estão fora do intervalo crítico, rejeita-se a hipótese que não há diferença ( $H_0$ ), ou seja, aceita-se  $H_1$  (existe diferença estatística entre os dois classificadores).

# Wilcoxon Signed-Rank test

O teste de Wilcoxon pode ser encontrado em diversas ferramentas

- Matlab (`signedrank`)
- Python Scipy (`from scipy import stats`)

```
Python
```

```
>>> c1 =[10, 15, 7, 6, 13, 11, 12, 9, 17, 14]
>>> c2 =[17, 14, 12, 16, 23, 18, 10, 8, 19, 22]
>>> from scipy.stats import wilcoxon
>>> wilcoxon(a,b)
(6.5, 0.031865021445197136)
```

```
MATLAB
```

```
>> [p,h,stats] = signrank(c1,c2)
p = 0.0332
h = 1
stats = signedrank: 6.5000
```

# Wilcoxon Signed-Rank test

- Um teste **não paramétrico** utilizado para comparar diversos classificadores é o teste de **Friedman**
  - Semelhante ao ANOVA

Aprendizado  
Não Supervisionado

# Agenda

- Aprendizado não supervisionado
  - Clustering
    - Particionamento: k-Means
    - Densidade: DBScan

# Aprendizagem Não Supervisionada

- Em problemas de aprendizagem supervisionada, contamos como a tupla  $[X,y]$ , em que o objetivo é classificar  $y$  usando o vetor de características  $X$ .
- Na aprendizagem não supervisionada, temos somente o vetor  $X$ .
- Nesse caso, o objetivo é descobrir alguma coisa a respeito dos dados
  - Por exemplo, como eles estão agrupados.
- Mais subjetiva que a aprendizagem supervisionada uma vez que não existe um objetivo simples como a classificação.
- **Dados não rotulados:**
  - Obtenção de dados não rotulados não é custosa!



# Noção de Grupos pode ser Ambígua

•



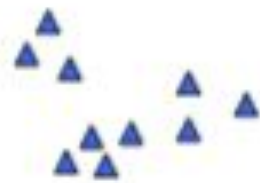
Quantos clusters?



Seis Clusters



Dois Clusters



Quatro Clusters



# *Clustering*

- Refere-se a um conjunto de técnicas utilizada para encontrar grupos (ou **clusters**) em um conjunto de dados.
- **Cluster**: Uma coleção de objetos similares entre si e diferentes dos objetos pertencentes a outros clusters.
- Métodos de *clustering* podem ser divididos em:
  - Particionamento
  - Hierárquico
  - Densidade

# Particionamento

## k-Means

- Separar os dados em um número pré-determinado de clusters
- k-Means Clustering

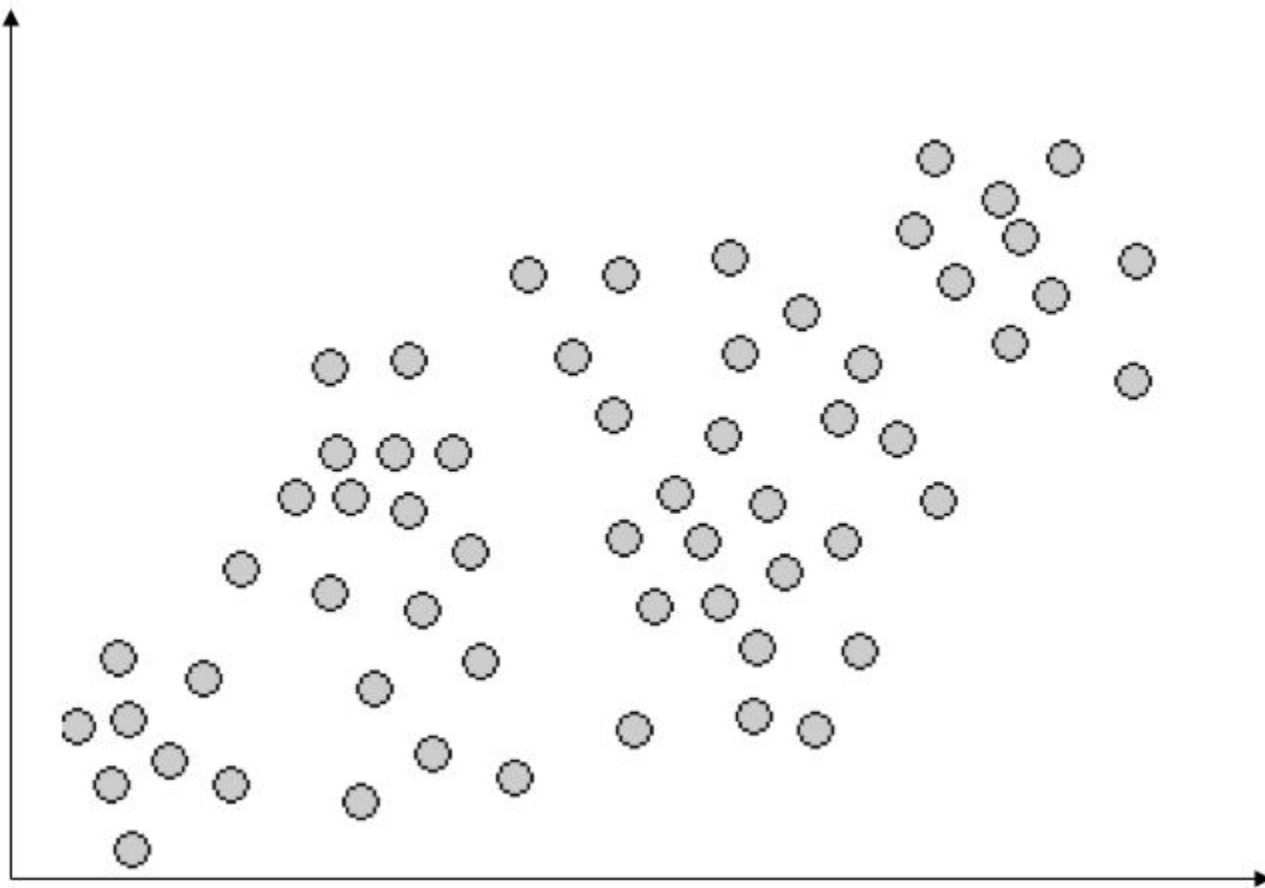
## Algoritmo

Entrada:  $k$ , dados ( $X$ )

Saída: dados agrupados em  $k$  grupos

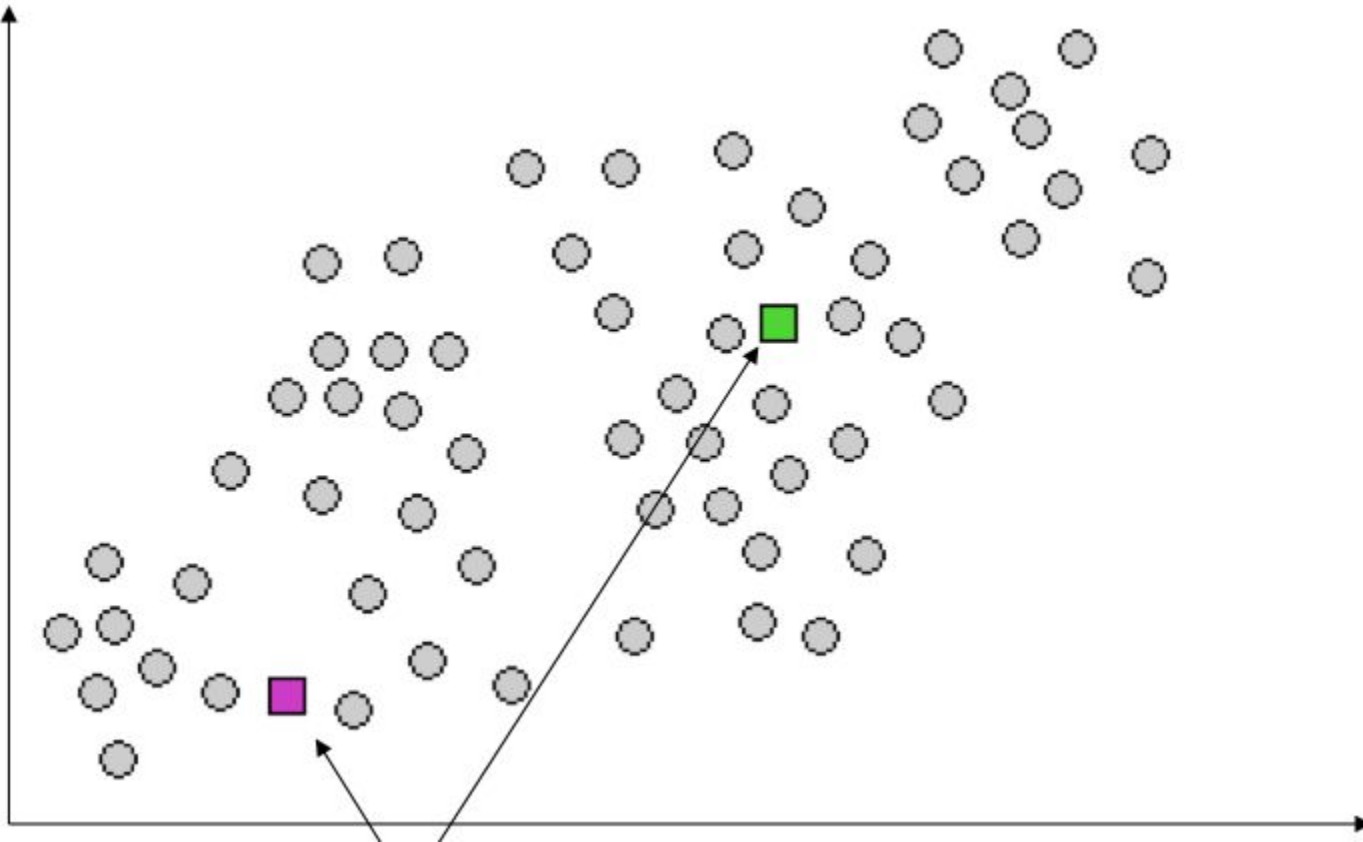
1. Determinar os  $k$  centróides
2. Atribuir a cada objeto do grupo o centróide mais próximo.
3. Após atribuir um centróide a cada objeto, recalcular os centróides
4. Repetir os passos 2 e 3 até que os centróides “não sejam” modificados

# k-Means



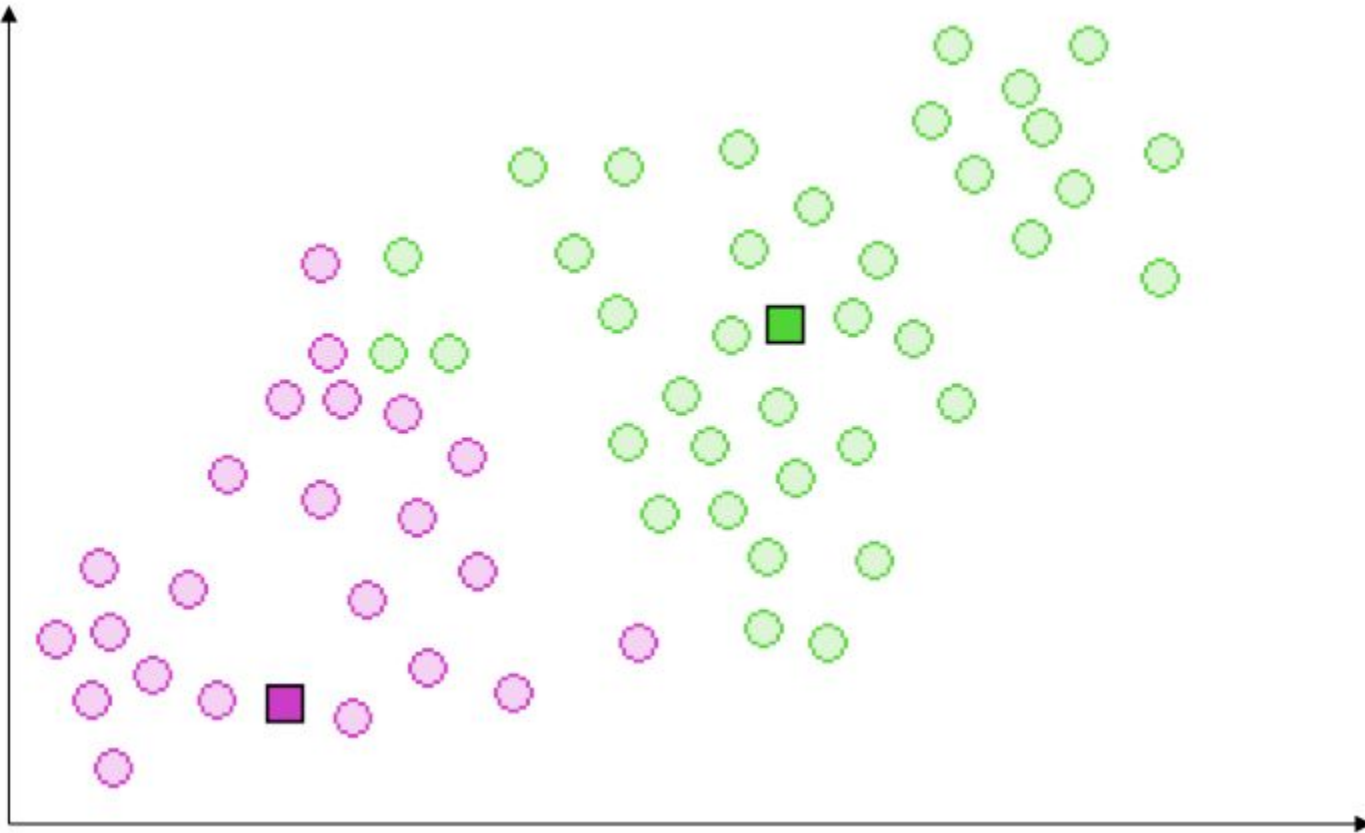
Dados em duas dimensões

# k-Means



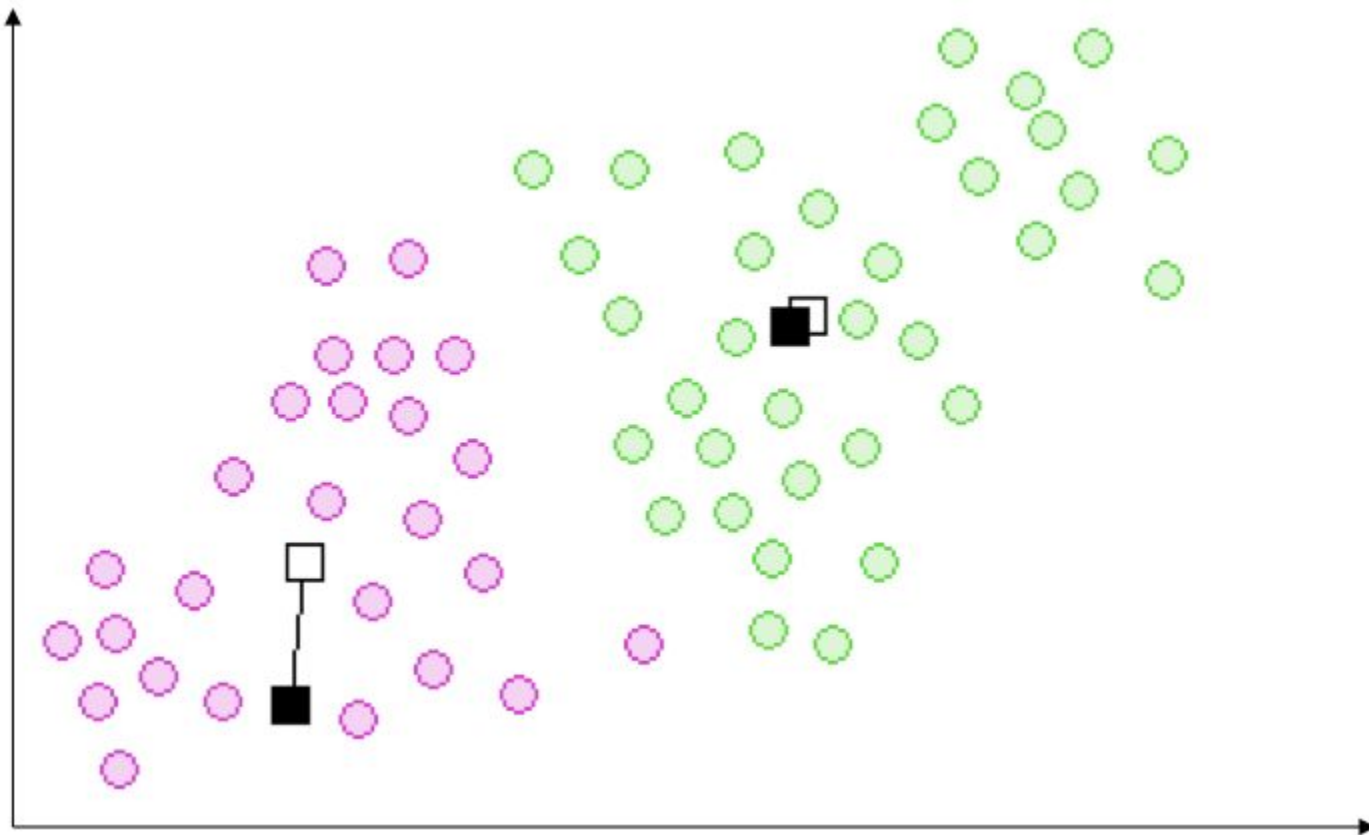
Passo 1: Centróides definidos aleatoriamente

# k-Means



Passo 2: Atribuir a cada objeto o centróide mais próximo

# k-Means



Passo 3: Recalcular os centróides

# k-Means

## Obsevações

- Relativamente eficiente (escalável)
- Necessidade de definir o número de clusters *a priori*.
- Ineficiente para lidar com ruídos e/ou *outliers*.
  - Todo ponto será atribuído a um cluster mesmo que ele não pertença a nenhum.
  - Em alguns domínios de aplicação (detecção de anomalias) isso causa problemas.
- Inadequado para descobrir *clusters* com formato **não convexo**.

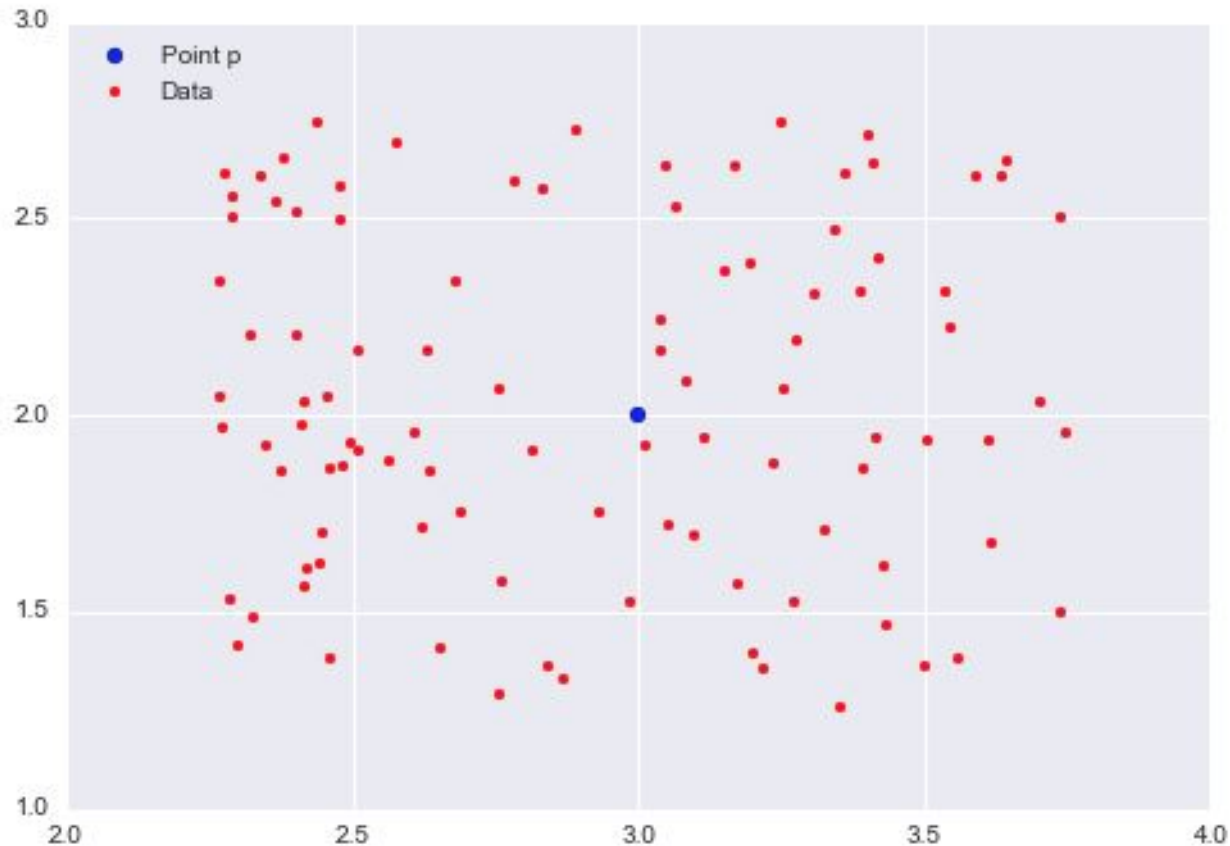


# Métodos baseados em Densidade

- Diferentemente do k-Means, os métodos baseados em densidade identificam regiões densas, permitindo a formação de clusters com diferentes formatos.
- Resistentes a presença de ruídos.
- Baseado no conceito de  $\epsilon$ -vizinhança.
- Em um espaço bi-dimensional, a  $\epsilon$ -vizinhança de um ponto  $p$  is o conjunto de pontos contido num círculo de raio  $\epsilon$ , centrado em  $p$ .

# $\epsilon$ -vizinhança

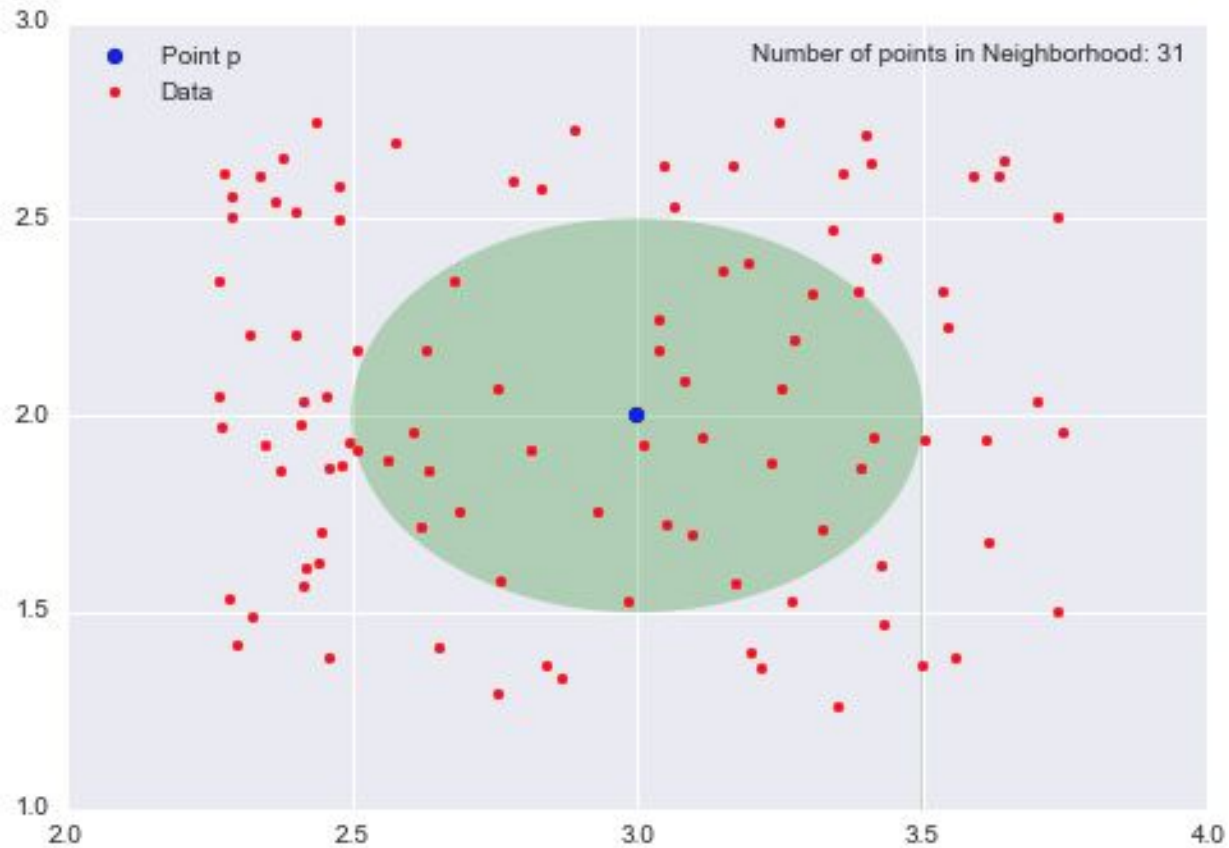
## Exemplo



100 pontos no intervalo  $[1,3] \times [2,4]$  e  $p = (3,2)$

# $\epsilon$ -vizinhança

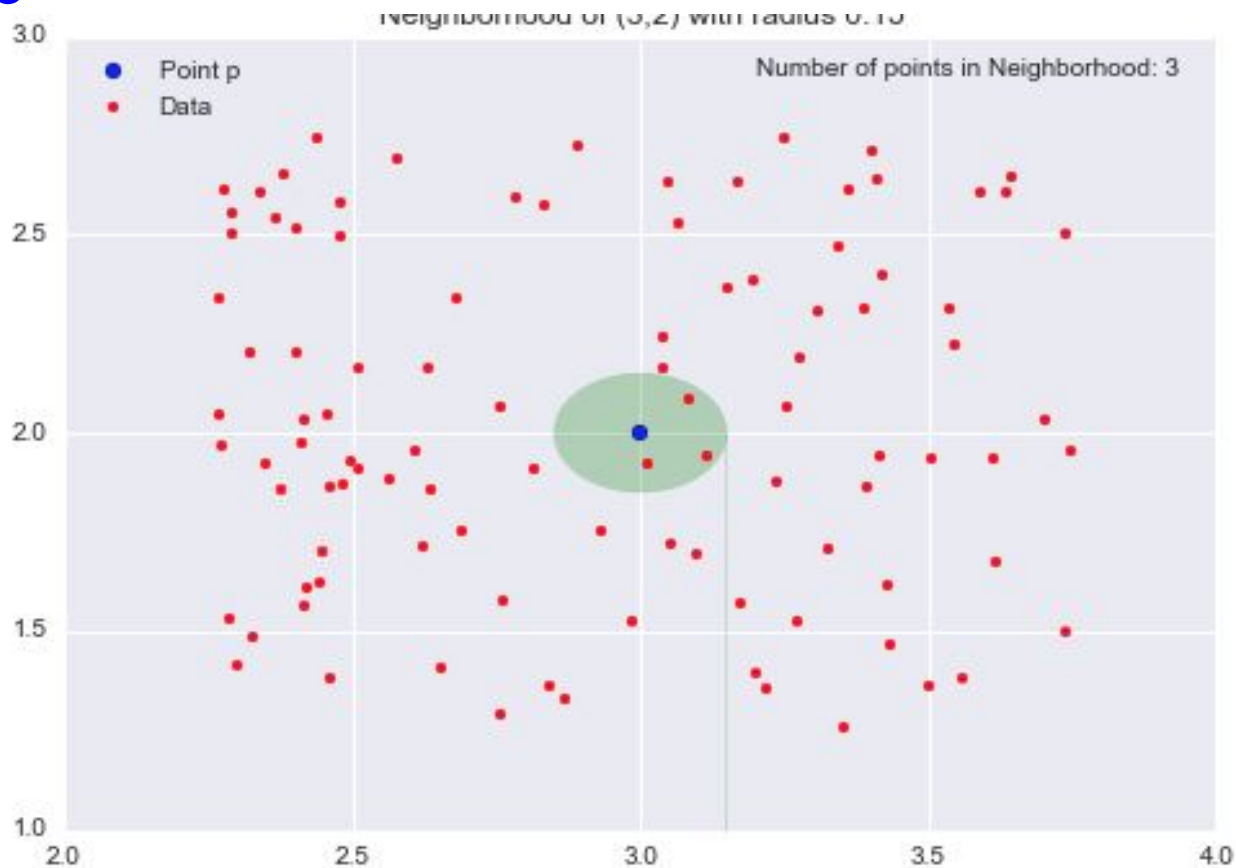
## Exemplo



Vizinha de  $p$  com raio 0,5 ( $\epsilon = 0,5$ ) e 31 pontos de 100

# $\epsilon$ -vizinhança

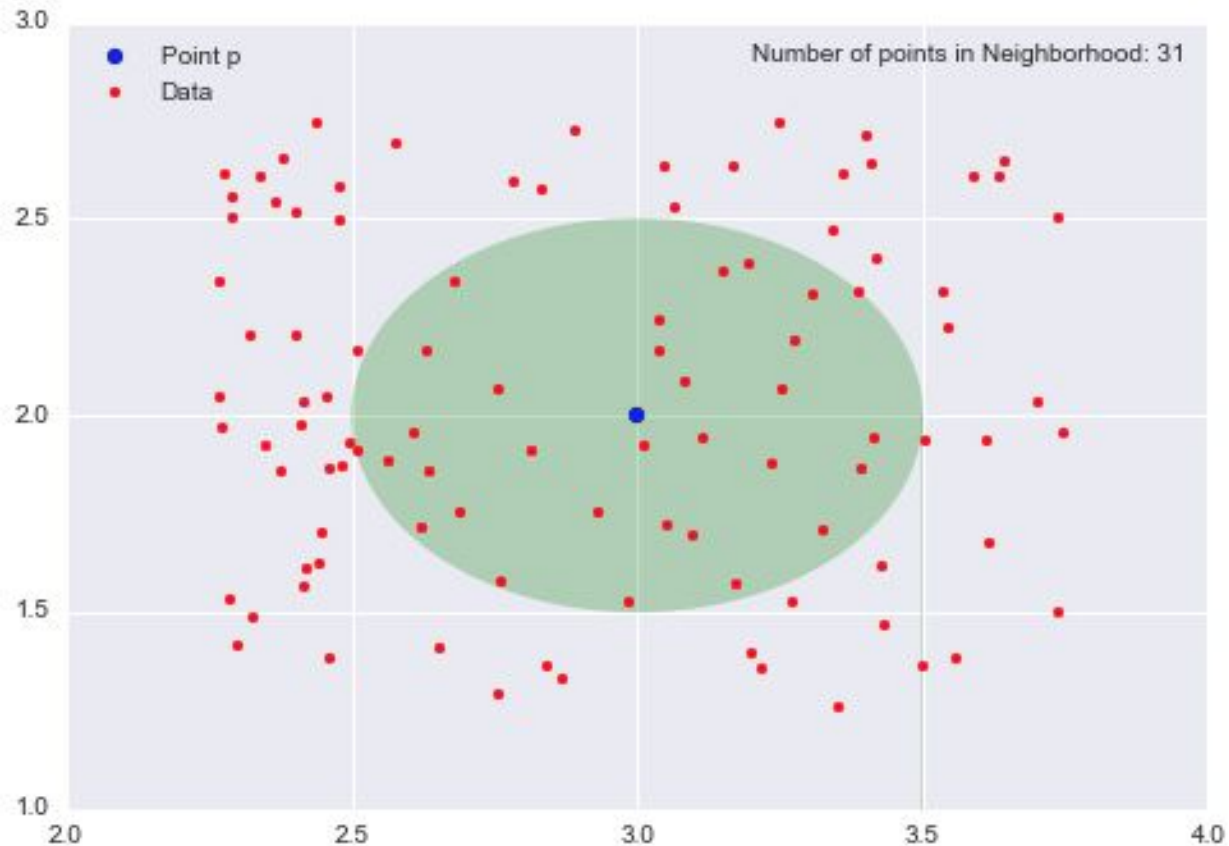
## Exemplo



Vizinha de  $p$  com raio 0,15 ( $\epsilon = 0,15$ ) e apenas 3 vizinhos.

# Densidade

Densidade = Massa / Volume

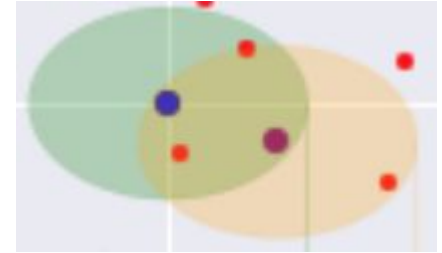


Para o caso de  $\epsilon = 0,15$ , temos o volume como a área do círculo, logo  $\pi(0,5)^2 = \pi/4$ .  
Desta forma, massa/volume = 31 pontos /  $(\pi/4) \approx 39,5$

# Densidade

- A ideia é usar o valor de densidade para agrupar aqueles pontos que possuem valores similares
- Identificar **vizinhanças** densas nas quais a maioria dos pontos está contida.
- Essa é a ideia do algoritmo DBSCAN

# DBSCAN



- Possui dois parâmetros:
  - $\epsilon$ : Raio da vizinhança
  - **MinPts**: Número mínimo de pontos na vizinhança para definir um *cluster*.
- Com bases nesses dois parâmetros, DBSCAN classifica os **pontos** em três categorias:
  - **core point**: Se a vizinhança de **p** com raio  $\epsilon$  contém pelo menos **MinPts**.
  - **border point**: Se a vizinhança de **q** com raio  $\epsilon$  contém menos de **MinPts**, mas pode ser alcançado por um **core point p**.
  - **outlier**: Nenhum dos casos acima.

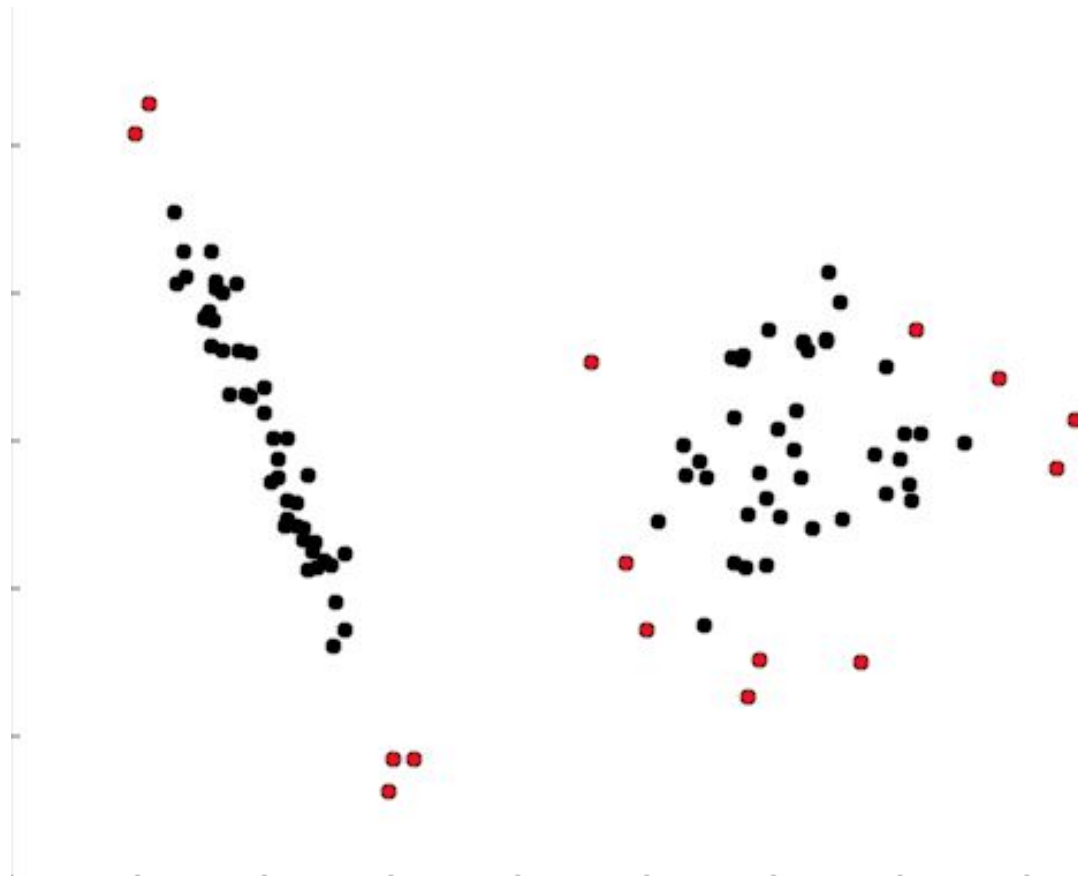
# DBSCAN

1. Selecione um ponto aleatoriamente que não tenha sido atribuído a nenhum *cluster* e que não seja um **outlier**. Calcule sua  $\epsilon$ -vizinhança e determine se ele é um **core point**.
  - Se sim, comece um cluster ao redor desse ponto.
  - Senão, rotule como **outlier**.
2. Depois de ter encontrado **core point**, expanda-o adicionando todos os pontos alcançáveis.
3. Repita os dois passos acima até que todos os pontos sejam atribuídos a um cluster ou sejam rotulados como **outliers**.



# DBSCAN

## Exemplo



Outliers marcados em **vermelho**

# Avaliação em Aprendizagem Não Supervisionada

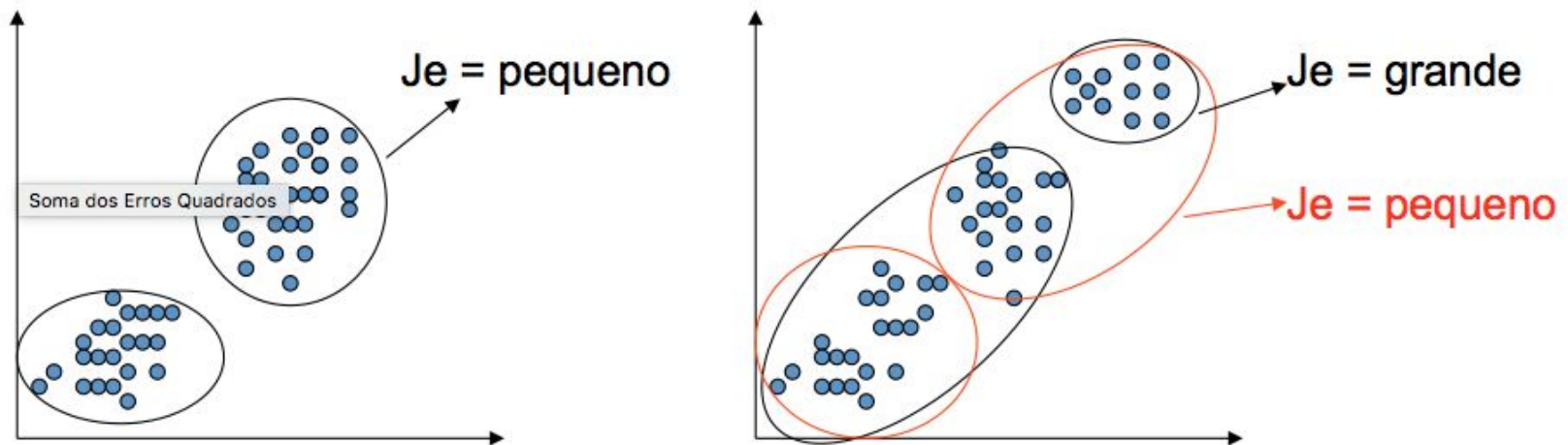
- Diferentemente da aprendizagem supervisionada, na qual podemos utilizar diferentes métricas de avaliação **objetivas**, a avaliação em aprendizagem não supervisionada é mais subjetiva.
- Porque é importante avaliar *clusters*?
  - Comparar algoritmos de *clustering*.
  - Comparar *clusters* gerados por mais de um algoritmo.

# Avaliação em Aprendizagem Não Supervisionada

- O que é um bom cluster ? x Coesão e separação.
- A média (centróide) de cada cluster  $D_i$

$$- m_i = \frac{1}{n_i} \sum_{x \in D_i} x$$

- Erro<sup>2</sup>  $J_e = \sum_{x=1}^c \sum_{x \in D_i} (x - m_i)^2$

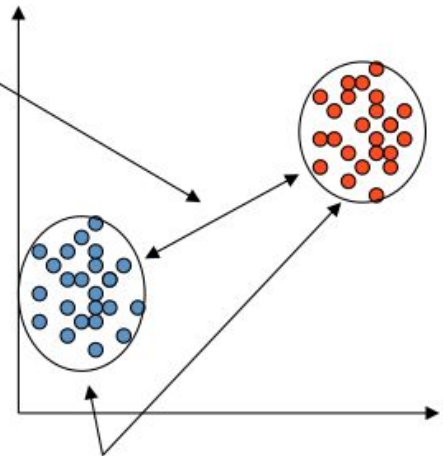


- **Outliers** podem afetar bastante os vetores médios.

# Avaliação em Aprendizagem Não Supervisionada

- A coesão (*within-cluster*) pode ser medida pela soma dos erros quadrados dentro de cada cluster
  - $S_w = \sum_{x \in D_i} (x - m_i)^2$
- A separação (*between-cluster*) é medida pela soma dos quadrados entre clusters
  - $S_b = \sum_{x=1}^c n_i (m_i - m)^2$
  - Em que  $m$  é o vetor médio total (centróide geral)

**Alto between ( $S_b$ )**  
Clusters distantes um do outro.



**Baixo within ( $S_w$ )**  
(boa compactação)

# Avaliação em Aprendizagem Não Supervisionada

## Coeficiente Silhouette

- O coeficiente Silhouette combina coesão e separação e pode ser calculado seguindo quatro passos:
  - a. Para um dado objeto do cluster  $i$ , calcule a distância média ( $a_i$ ) para todos os outros objetos de  $i$ .
  - b. Seja  $b_i$  a menor distância média de  $i$  para todos os outros *clusters*, dos quais  $i$  não seja membro.
  - c.  $s_i = (b_i - a_i) / \max(a_i, b_i)$
  - d. O coeficiente  $s_k$  para todos os objetos  $I$  numa partição de  $k$  *clusters* é dados por  $s_k = \frac{1}{I} \sum_{i=1}^I s_i$

# Avaliação em Aprendizagem Não Supervisionada

## Coeficiente *Silhouette*

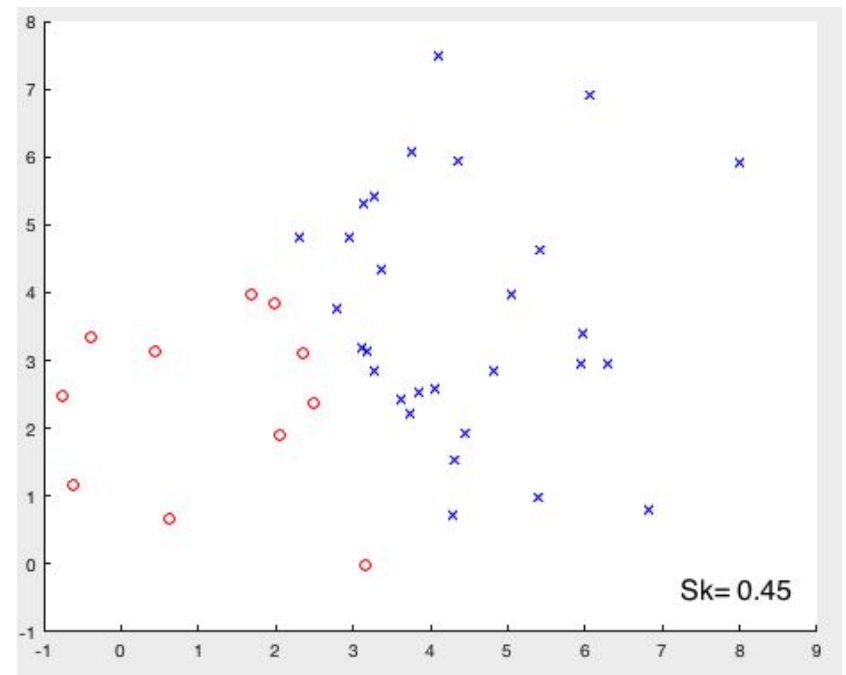
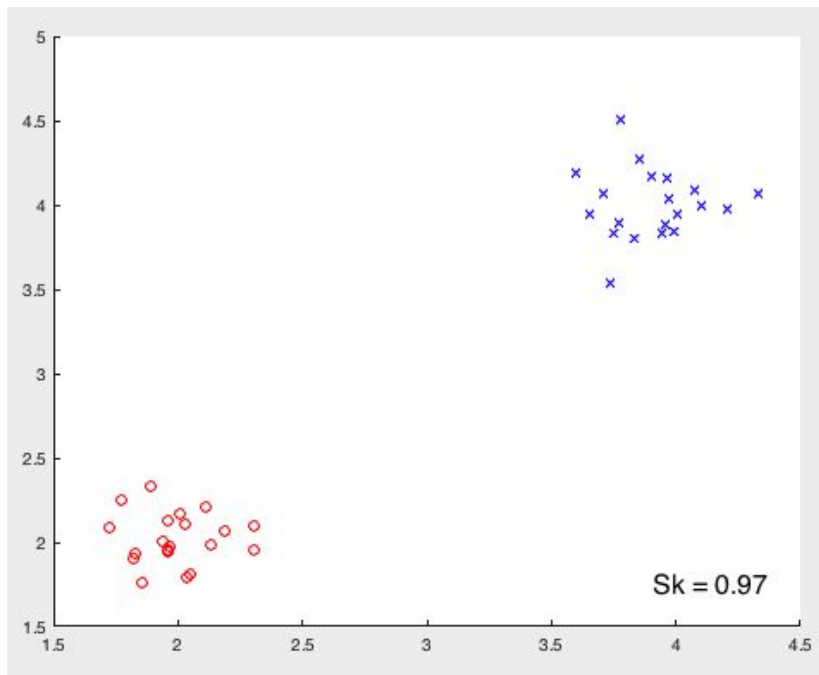
- Coesão e separação

$$s_k = \frac{1}{I} \sum_{i=1}^I s_i \qquad s_i = (b_i - a_i) / \max(a_i, b_i)$$

- O valor do coeficiente pode variar entre -1 e 1.
- Um valor negativo não é desejável pois nesse caso o valor de  $a_i$  seria maior do que  $b_i$
- É desejável que o coeficiente seja positivo (  $a_i < b_i$  ) o que indica clusters compactos ( $a_i$  próximo de zero).

# Avaliação em Aprendizagem Não Supervisionada

## Coeficiente *Silhouette*



# Avaliação em Aprendizagem Não Supervisionada

## Coeficiente Silhouette

- Prática comum: Utilize mais de um índice de clustering
- Voto entre os índices pode ser um bom indicativo de qual número de clusters você deve escolher.

```
*****
```

```
* Among all indices:
```

```
* 3 proposed 2 as the best number of clusters
```

```
* 4 proposed 3 as the best number of clusters
```

```
* 19 proposed 4 as the best number of clusters
```

```
* 1 proposed 5 as the best number of clusters
```

```
***** Conclusion *****
```

```
* According to the majority rule, the best number of clusters is 4
```



# Regressão

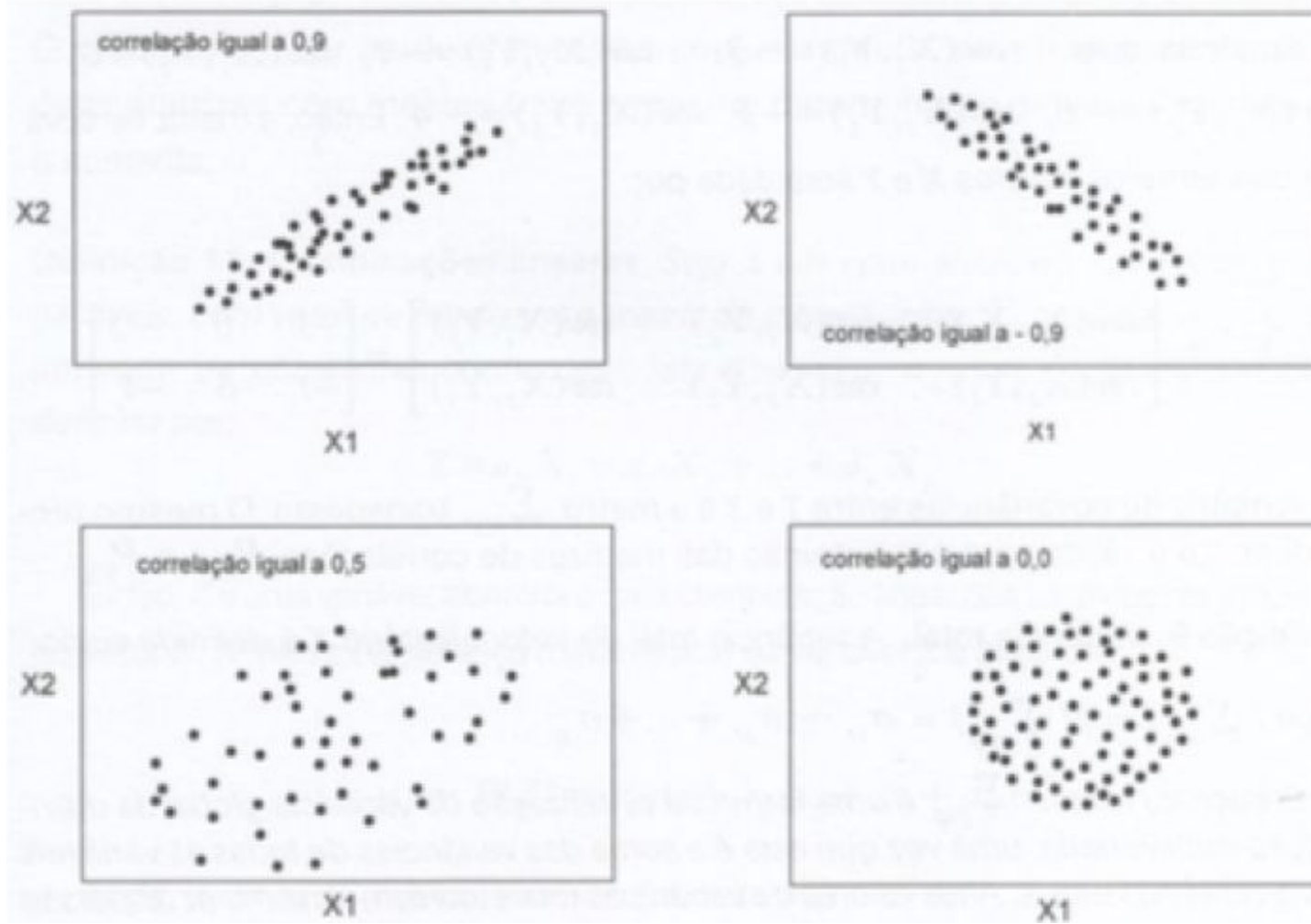
# Agenda

- Regressão
  - Correlação
  - Análise de Regressão
    - Linear
    - Não-linear

# Correlação

- Indica a força e a direção do relacionamento linear entre dois atributos.
- Trata-se de uma medida de **relação** entre dois atributos, embora correlação não implique **causalidade**
  - Duas variáveis podem estar altamente **correlacionadas** e não existir relação de causa e efeito entre elas.
- Em muitas aplicações duas ou mais variáveis estão relacionadas, sendo necessário explorar a natureza desta relação
  - Correlação muito próxima de 1 ou de -1 indica relação linear entre dois atributos.
  - Nesse caso é possível ajustar um modelo que expresse tal relação
  - Esse é o objetivo da análise de regressão.

# Correlação



# Análise da Regressão

**Objetivo:** Determinar o modelo que expressa esta relação (modelo de regressão) a qual é ajustada aos dados

- Permite construir um modelo matemático que representa dois atributos (  $x$  e  $y$  )
- $y = f(x)$ , em que  $f(.)$  é a função que relaciona  $x$  e  $y$
- $x$  é a variável independente da equação
- $y$  é a variável dependente das variações de  $x$

# Análise da Regressão

- Esse modelo pode ser usado para prever o valor de  $y$  para um dado valor de  $x$ 
  - Realizar previsões sobre um comportamento futuro de algum fenômeno da realidade.
  - Extrapolar para o futuro relações de causa-efeito entre as variáveis, já observadas no passado.
  - **Cuidado** com a extrapolação

*Mas em toda a minha experiência nunca estive em nenhum acidente... de qualquer tipo digno de menção. Só vi uma única embarcação em perigo em todos os meus anos no mar. Nunca vi um naufrágio nem nunca naufraguei, tampouco enfrentei qualquer contratempo que ameaçasse terminar em qualquer tipo de desastre.*

E. J. Smith, 1907, capitão, RMS

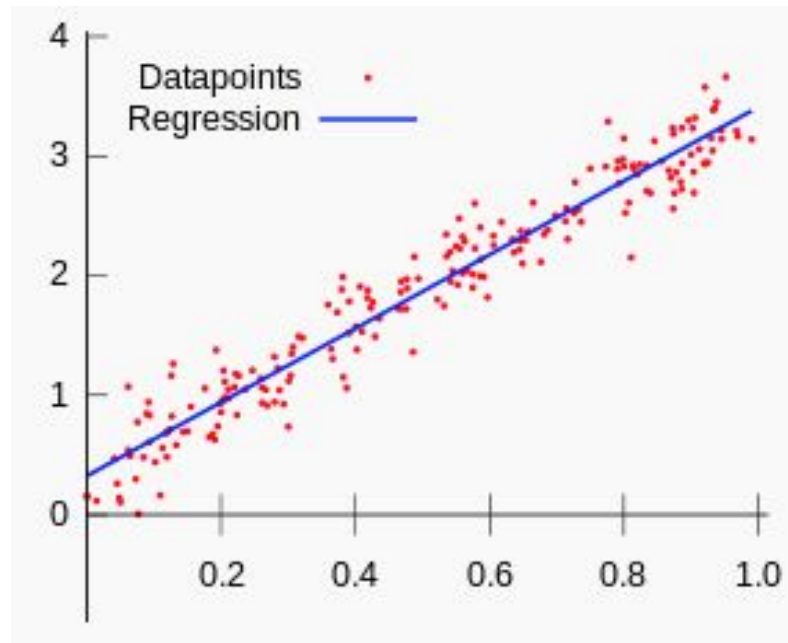
*Titanic*

# Análise da Regressão

- A análise de regressão compreende dois tipos básicos:
  - Linear
  - Linear Múltipla
  - Não Linear
    - Modelos exponenciais
    - Modelos logísticos

# Regressão Linear

- Considera que a relação da resposta às variáveis é uma função linear de alguns parâmetros



- Modelos de regressão linear são frequentemente ajustados usando a abordagem dos mínimos quadrados.



# Método dos Mínimos Quadrados

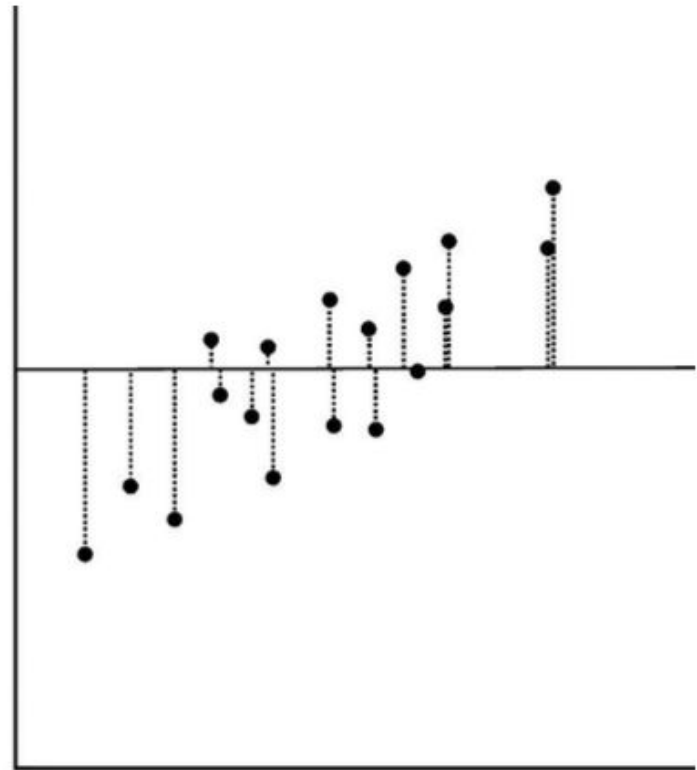
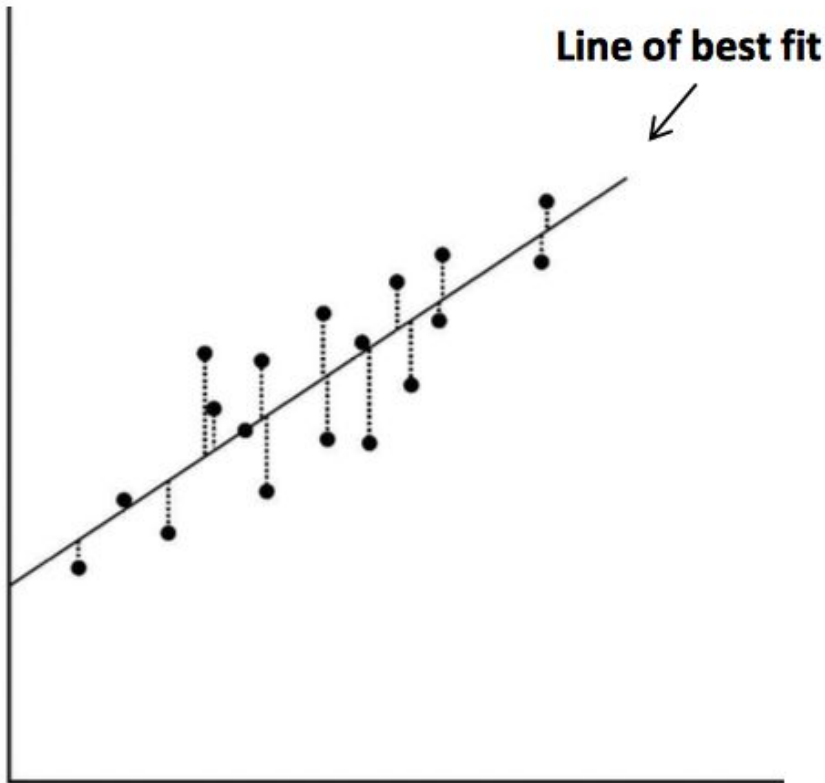
- Seja a equação da reta  $y = ax + b$  (ou  $y = \alpha + \beta x$ ) que modela a relação entre uma variável independente e uma variável dependente.
- Seja um conjunto de  $n$  pontos de dados conhecidos:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

- O objetivo é encontrar os coeficientes  $a$  e  $b$  para o modelo que minimizem a medida de erro quadrático RSS (Residual Sum of Squares)

$$RSS = \sum_{i=1}^n (y - y_i)^2$$

# Regressão Linear



# Método dos Mínimos Quadrados

- Os parâmetros da reta podem ser estimados através do conjunto de dados **D** usando as seguintes equações:

$$a = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}$$

$$b = \frac{\sum_{i=1}^n x_i^2 \sum_{i=1}^n y_i - \sum_{i=1}^n x_i y_i \sum_{i=1}^n x_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}$$

# Método dos Mínimos Quadrados

## Exemplo

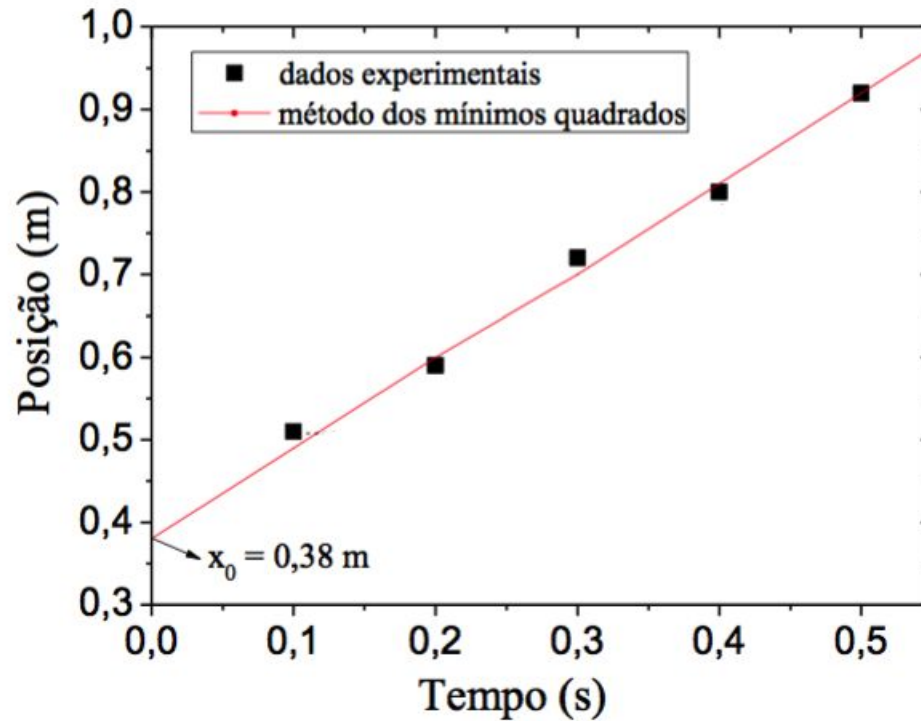
- Considere os dados da tabela abaixo. Encontre os coeficientes  $a$  e  $b$  e calcule a posição ( $y$ ) no tempo  $x = 0,6$

X - tempo (s)	Y - posição (m)
0,100	0,51
0,200	0,59
0,300	0,72
0,400	0,80
0,500	0,92

- Da tabela acima temos,  $\sum x = 1.5$        $\sum xy = 1.17$   
 $\sum y = 3.54$        $\sum x^2 = 0.55$
- Usando as equações anteriores, temos  $a = 1,08$  e  $b = 0,38$
- Desta forma, temos o modelo de regressão  $y = 1,08x + 0,38$

# Método dos Mínimos Quadrados

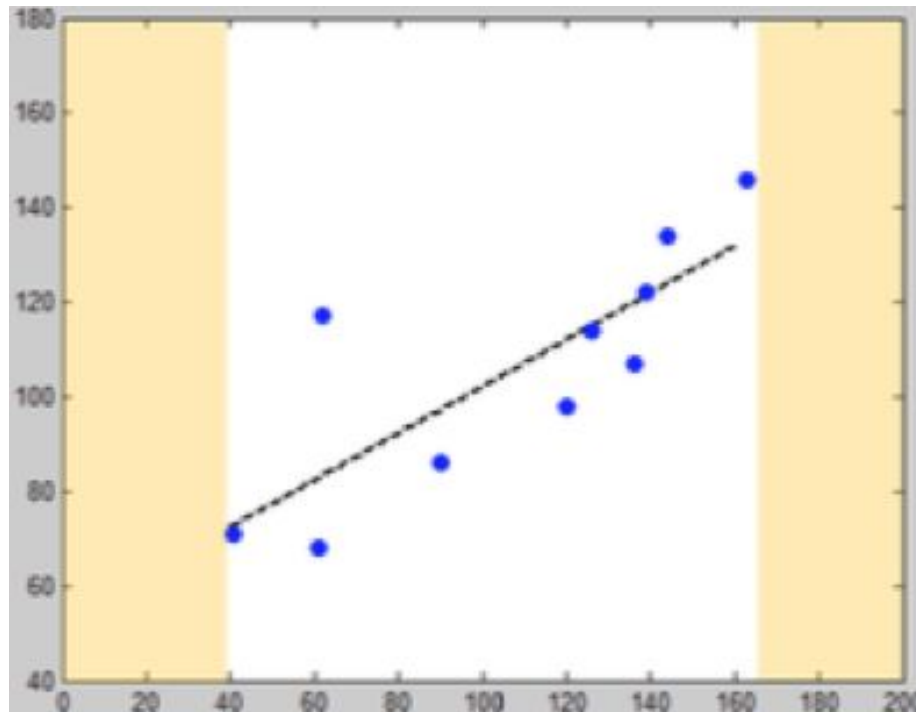
## Exemplo



- Respondendo a questão anterior, para  $x = 0,6$ ,  $y = 1,02$

# Método dos Mínimos Quadrados

- Modelos de regressão linear não costumam ser válidos para fins de **extrapolação**, ou seja, predizer um valor fora do domínio dos dados



# Análise de resíduos

- Como avaliar a qualidade do modelo?
  - Os erros têm distribuição normal?
  - Existem “outliers” no conjunto de dados?
  - O modelo gerado é adequado?
- Podemos responder essas perguntas analisando os resíduos, o qual é dado pela diferença entre o  $y_i$  e sua estimativa  $\hat{y}_i$

$$e_i = y_i - \hat{y}_i$$

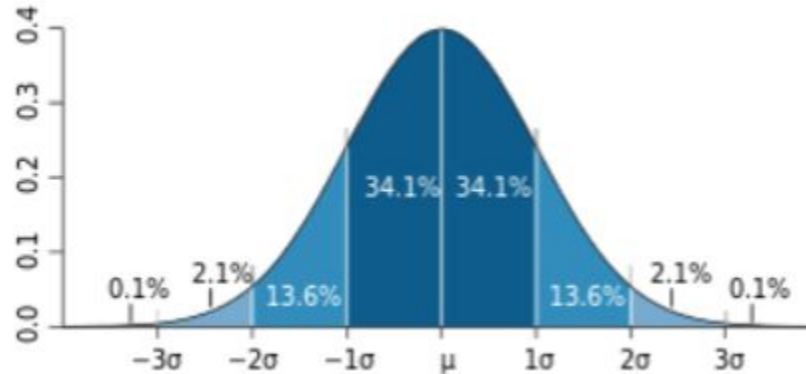
# Análise de resíduos

## Outliers

- Construir um histograma de frequência dos resíduos
- Normalizar de modo a ter média zero e desvio 1

$$Z = \frac{e - \mu}{\sigma}$$

- O histograma de resíduos deve ser semelhante a uma normal.

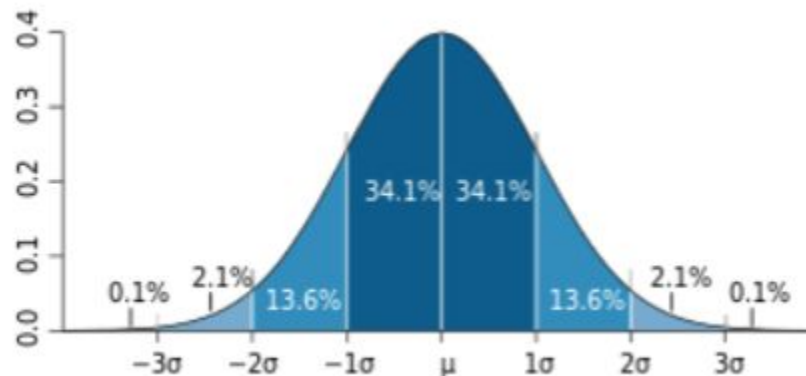




# Análise de resíduos

## Outliers

- Se os erros tiverem uma distribuição normal,
  - Aproximadamente, 95% dos resíduos estarão no intervalo de um desvio padrão da média
- Caso contrário, deve existir a presença de “outlier”,
  - ou seja, valores atípicos ao restante dos dados.
- O que fazer com os outliers?

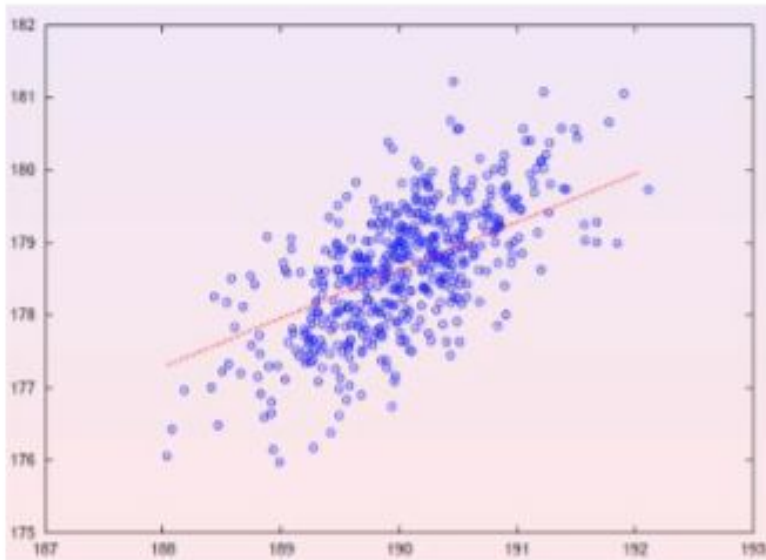


# Coeficiente de Determinação ( $R^2$ )

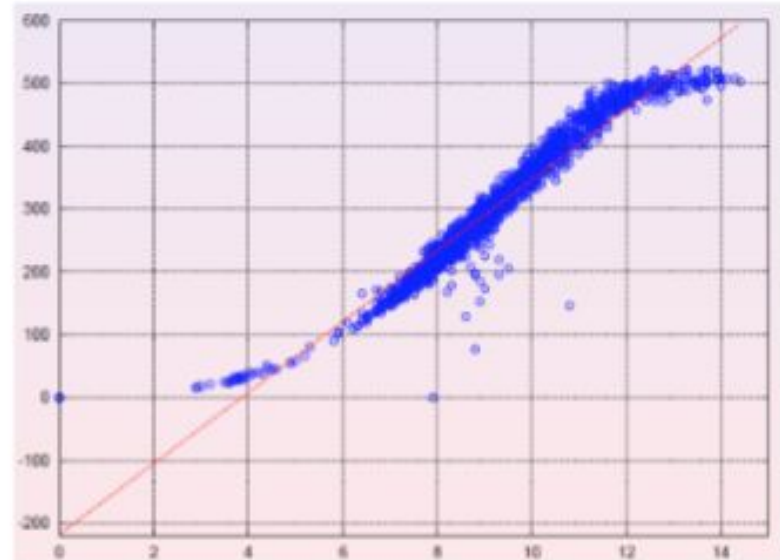
- Uma forma de avaliar a **qualidade** de ajuste do modelo.
- Indica a quantidade de **variabilidade** dos dados que o modelo de regressão é capaz de explicar.
- Varia entre 0 e 1, indicando quanto o modelo consegue explicar os valores observados.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2}$$

# Coeficiente de Determinação ( $R^2$ )



$$R^2 = 0.44$$



$$R^2 = 0.93$$

# Regressão Linear

## Exercício

- Programa exemplo de regressão usando scikit-learn: **Linear Regression Example**[1](#)

```
print(__doc__)

# Code source: Jaques Grobler
# License: BSD 3 clause

import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Load the diabetes dataset
diabetes = datasets.load_diabetes()

# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes.target[:-20]
diabetes_y_test = diabetes.target[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()
```

```
# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)

# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(diabetes_y_test, diabetes_y_pred))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(diabetes_y_test, diabetes_y_pred))

# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test, color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)

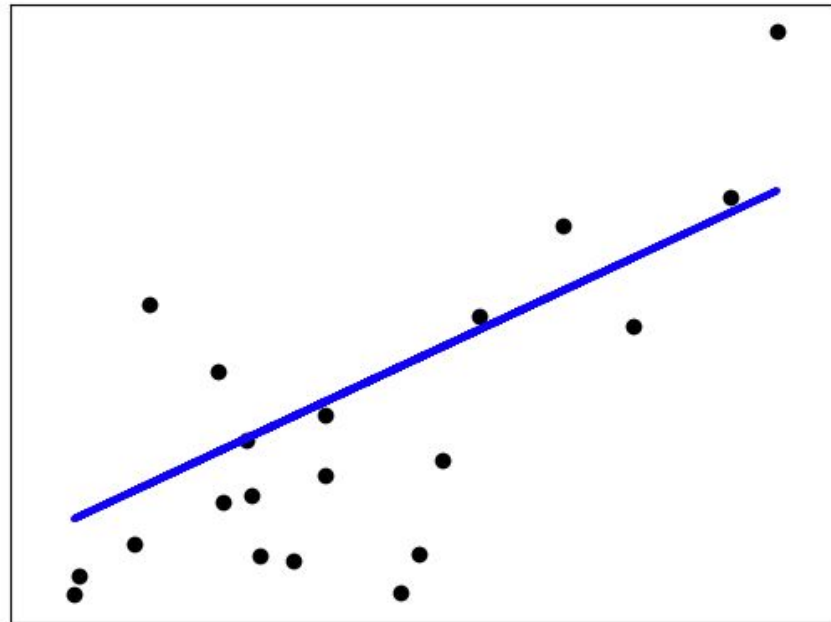
plt.xticks(())
plt.yticks(())

plt.show()
```

# Regressão Linear

## Exercício

The coefficients, the residual sum of squares and the variance score are also calculated.



```
Out: Coefficients:  
      [938.23786125]  
Mean squared error: 2548.07  
Variance score: 0.47
```

# Regressão Linear Múltipla

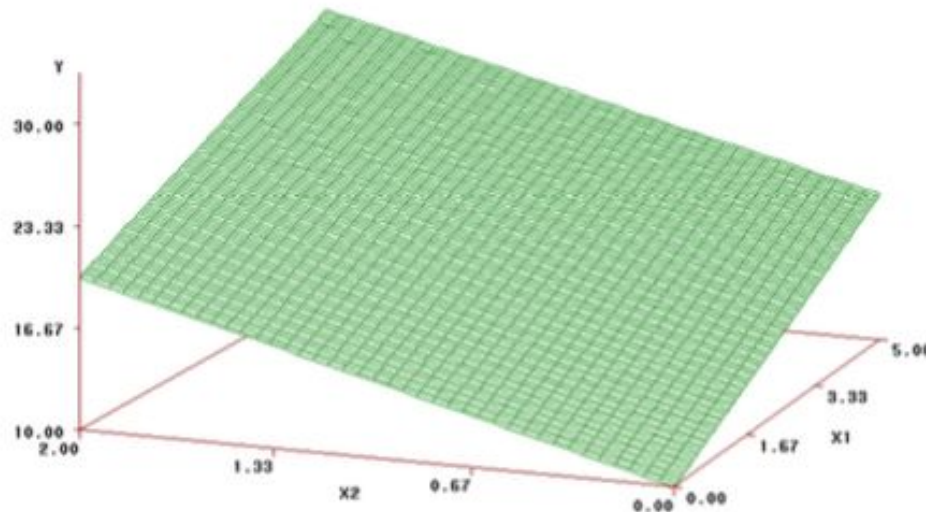
- A regressão múltipla funciona de forma parecida com a regressão simples
- Leva em consideração diversas variáveis de entrada  $x_i$ ,  $i = 1, \dots, p$ , influenciando ao mesmo tempo uma única variável de saída,  $y$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p + \epsilon$$

# Regressão Linear Múltipla

**Exemplo:**

$$y = 10 + 2x_1 + 5x_2$$



- A função de regressão na regressão múltipla é chamada de superfície de resposta
- Descreve um hiperplano no espaço p-dimensional das variáveis de entrada  $x$

# Regressão Linear Múltipla

Como calcular a superfície de regressão?

- Usar o método dos mínimos quadrados como feito na regressão linear simples
- A diferença é que agora temos um elevado número de parâmetros na forma

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p + \epsilon$$

- **Solução:** Expressar as operações matemáticas utilizando notação matricial

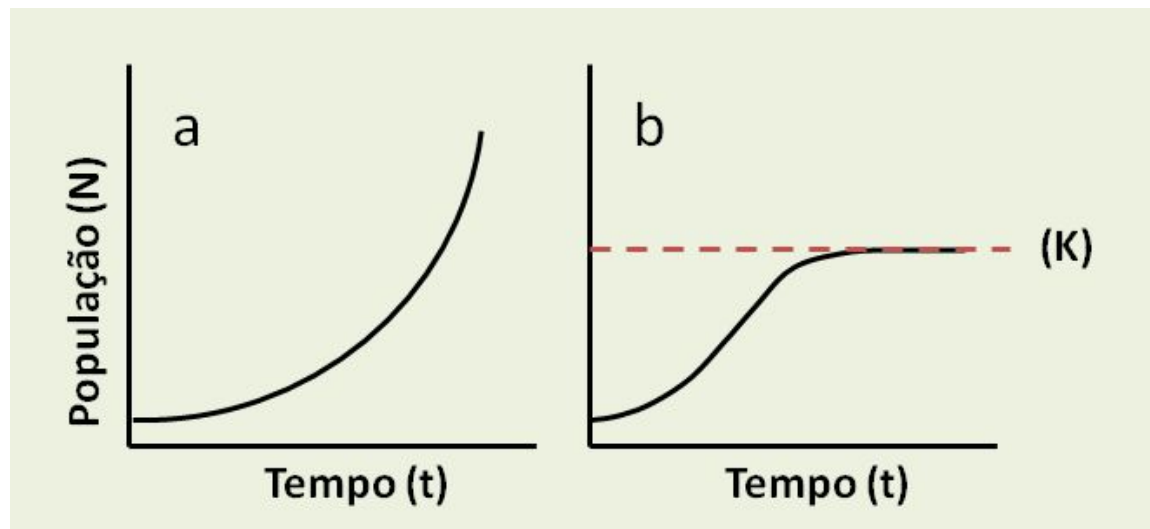
$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} \dots & x_{p1} \\ 1 & x_{12} \dots & x_{p2} \\ \dots & \dots \dots & \dots \\ 1 & x_{1p} \dots & x_{pn} \end{pmatrix} * \begin{pmatrix} \beta_0 \\ \beta_1 \\ \dots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \dots \\ \epsilon_n \end{pmatrix}$$

- $y = X\beta + \epsilon$



# Regressão Não Linear

- Em alguns casos o modelo linear pode não ser o mais adequado.
- Muitas aplicações biológicas são modeladas por meio de relações não lineares.
- Por exemplo, padrões de crescimento podem seguir modelos:
  - Exponenciais: onde a população aumenta sem limites
  - Logísticos: onde a população cresce rapidamente no início, desacelera e se mantém estável.



# Método dos Mínimos Quadrados

## Caso Exponencial

- O método dos mínimos quadrados pode ser facilmente **adaptado** para o caso exponencial, usando logaritmos neperianos.
- Nesse caso,  $y' = \ln(y)$
- A equação que modela a relação entre uma variável independente e uma variável dependente é dada por  $y = be^{ax}$

$$a = \frac{n \sum_{i=1}^n x_i y'_i - \sum_{i=1}^n x_i \sum_{i=1}^n y'_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}$$

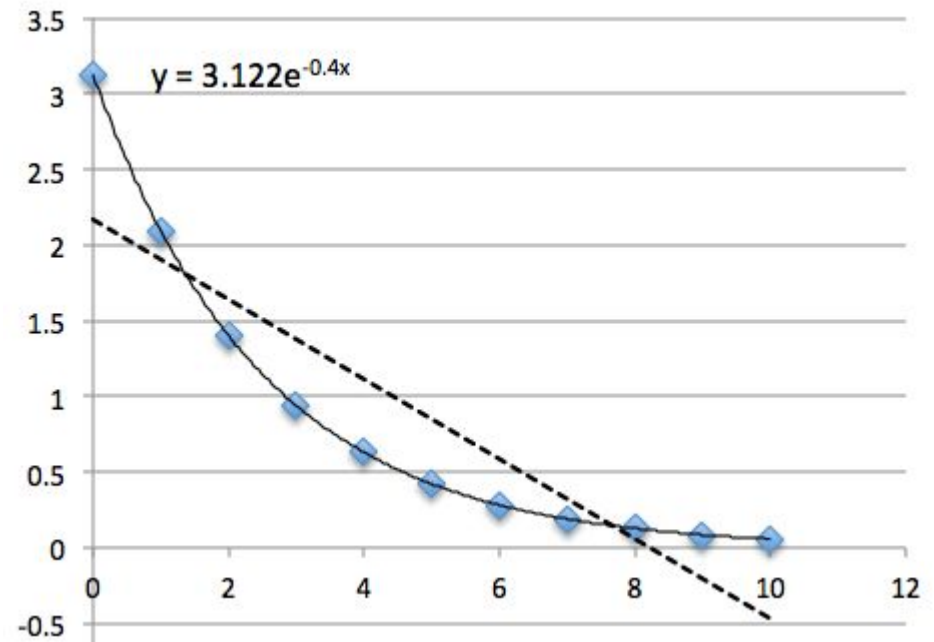
$$b = e^{\bar{y}' - a\bar{x}}$$

# Método dos Mínimos Quadrados

## Caso Exponencial

### Exemplo

x	y	$\ln(y) = y'$	$x^2$	$x \cdot y'$	
0	3.12	1.1378	0	0	
1	2.091	0.7376	1	0.738	
2	1.402	0.3379	4	0.676	
3	0.94	-0.062	9	-0.186	
4	0.63	-0.462	16	-1.848	
5	0.422	-0.863	25	-4.314	
6	0.283	-1.262	36	-7.574	
7	0.19	-1.661	49	-11.63	
8	0.127	-2.064	64	-16.51	
9	0.085	-2.465	81	-22.19	
10	0.057	-2.865	100	-28.65	
SUM	55	9.347	-9.49	385	-91.47
AVG	5		-0.863		-8.316



# Regressão Logística

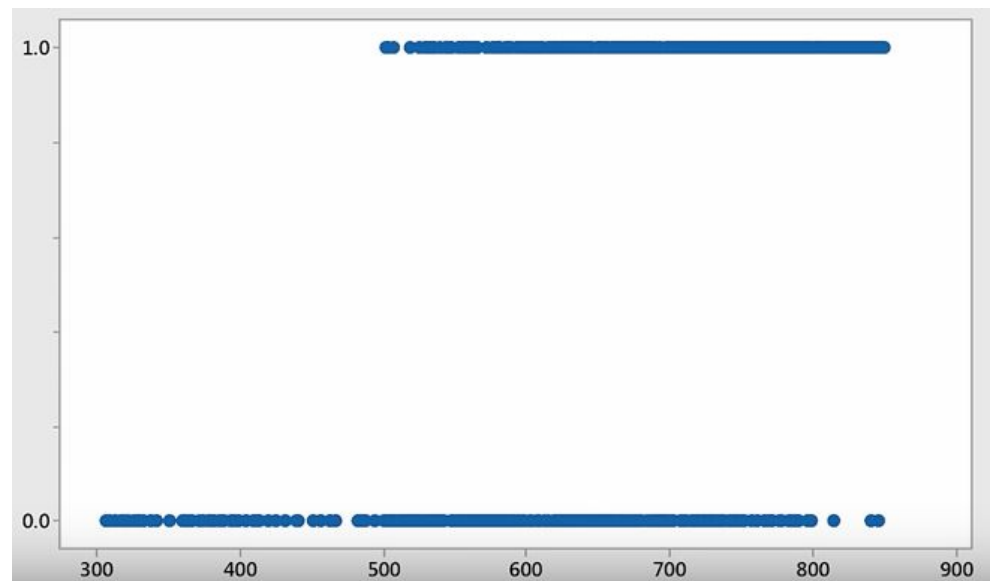
## Exemplo

- Considere que alguém queira comprar uma casa e está na busca por um financiamento.
- Os bancos mantêm para cada cliente um **score**, que varia de 300 a 850
- Suponha que um dado cliente tem um score de 720 e gostaria de saber qual é a probabilidade de ter seu crédito aprovado pela instituição financeira.
- Esse cliente encontrou ainda dados de 1000 clientes com seus respectivos escores, que tiveram seus pedidos aprovados ou não.
- Notem que nesse caso a variável **y** é dicotômica
  - Aprovado (1)
  - Negado (0)

# Regressão Logística

## Exemplo

creditScore	approved
655	0
692	0
681	0
663	1
688	1
693	1
699	0
699	1
683	1
698	0
655	1
703	0
704	1
745	1
702	1



- Note que a distribuição dos dados é diferente (**binomial**).
- O modelos de regressão vistos até então não funcionam nesse caso.
- Solução: Regressão Logística

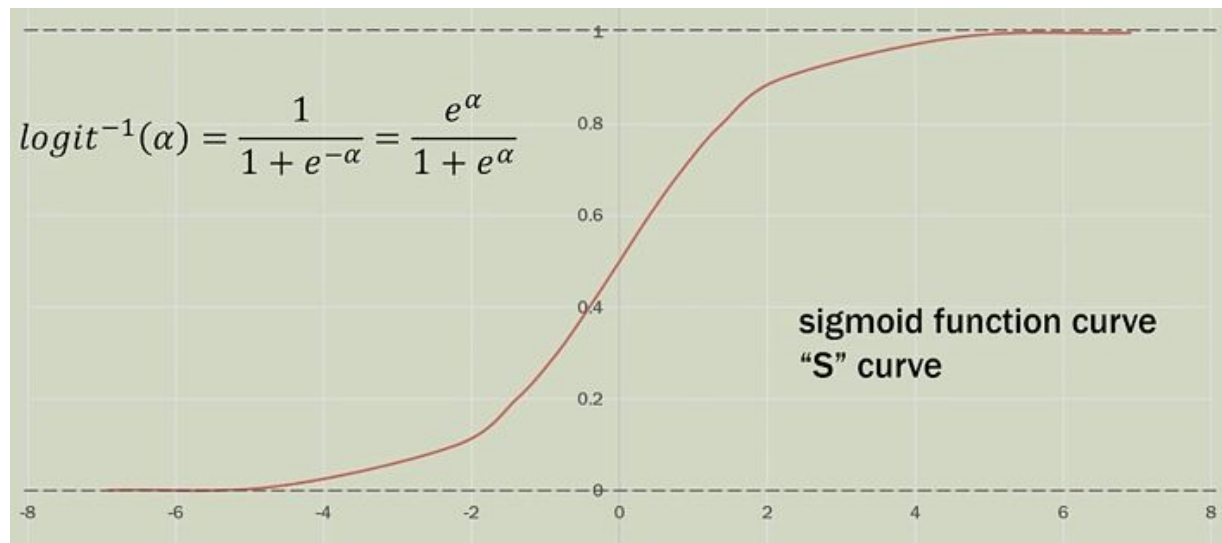
# Regressão Logística

A regressão logística tem como objetivo:

- Modelar a probabilidade de um evento ocorrer dependendo dos valores das variáveis independentes.
- Estimar a probabilidade de um evento ocorrer (e também de não ocorrer) para uma dada observação.
- Pode ser usada como um classificador, atribuindo uma classe ao padrão de entrada.
  - No nosso exemplo, crédito aprovado ou reprovado.

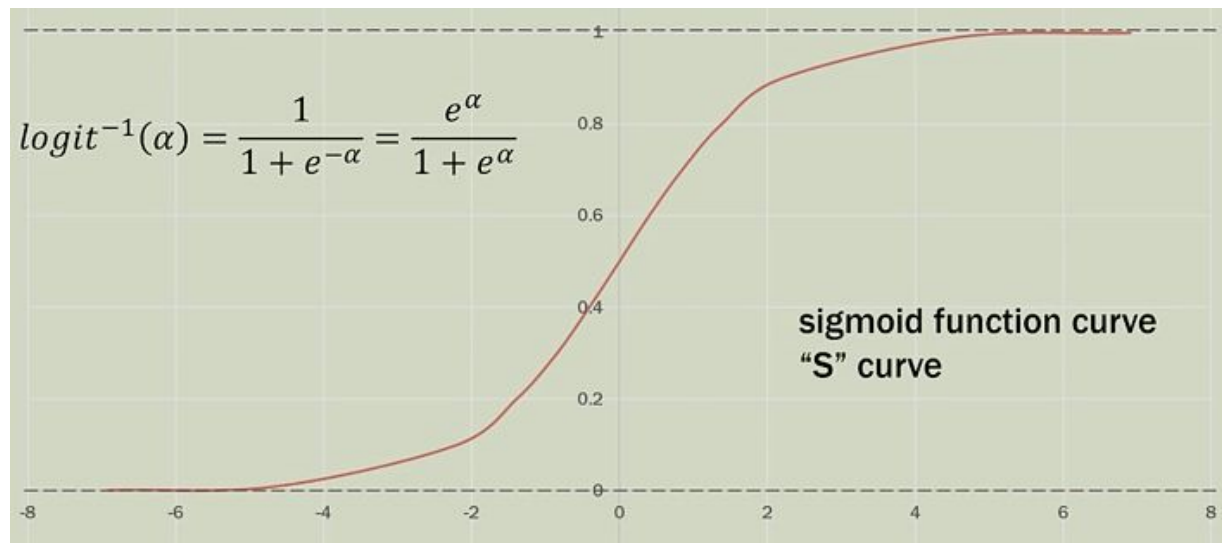
# Regressão Logística

- A variável dependente na regressão logística segue a distribuição de Bernoulli.
- Distribuição discreta de espaço amostral  $\{0,1\}$  que tem probabilidade de sucesso  $p$  e falha  $q = 1 - p$
- Na regressão logística estimamos  $p$  para qualquer combinação linear das variáveis independentes.



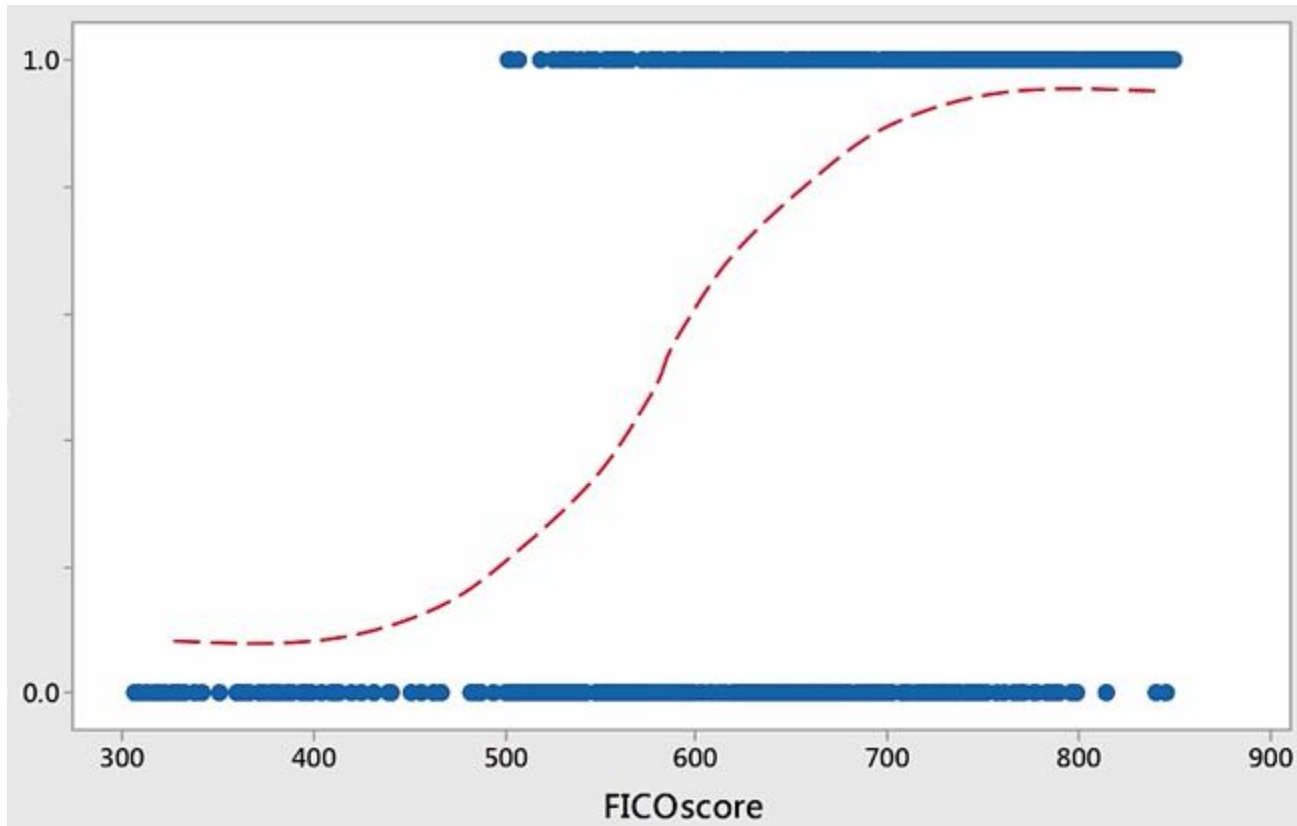
# Regressão Logística

- Isso pode ser alcançado ajustando o seguinte modelo (função logística)
- MLE (*Maximum Likelihood Estimation*) é usada para estimar os coeficiente do modelo.





# Regressão Logística



- Esse modelo diz basicamente que a probabilidade de conseguir crédito sobre em função do *score*

# Regressão Logística

## Exercício

- Programa exemplo de regressão usando `scikit-learn`: **Logistic Function**

```
print(__doc__)

# Code source: Gael Varoquaux
# License: BSD 3 clause

import numpy as np
import matplotlib.pyplot as plt

from sklearn import linear_model

# this is our test set, it's just a straight line
# Gaussian noise
xmin, xmax = -5, 5
n_samples = 100
np.random.seed(0)
X = np.random.normal(size=n_samples)
y = (X > 0).astype(np.float)
X[X > 0] *= 4
X += .3 * np.random.normal(size=n_samples)

X = X[:, np.newaxis]
# run the classifier
clf = linear_model.LogisticRegression(C=1e5)
clf.fit(X, y)

# and plot the result
plt.figure(1, figsize=(4, 3))
plt.clf()
plt.scatter(X.ravel(), y, color='black', zorder=20)
X_test = np.linspace(-5, 10, 300)

def model(x):
    return 1 / (1 + np.exp(-x))
loss = model(X_test * clf.coef_ + clf.intercept_).ravel()
plt.plot(X_test, loss, color='red', linewidth=3)

ols = linear_model.LinearRegression()
ols.fit(X, y)
plt.plot(X_test, ols.coef_ * X_test + ols.intercept_, linewidth=1)
plt.axhline(.5, color='.5')

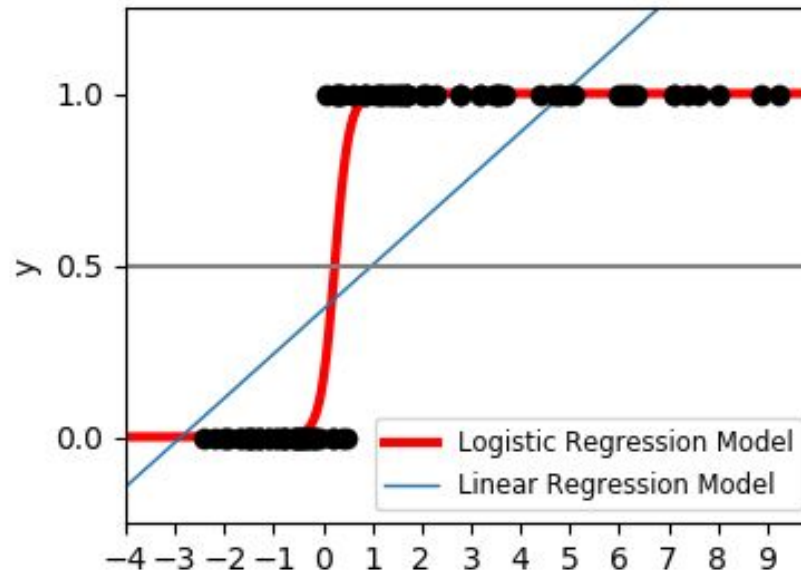
plt.ylabel('y')
plt.xlabel('X')
plt.xticks(range(-5, 10))
plt.yticks([0, 0.5, 1])
plt.ylim(-.25, 1.25)
plt.xlim(-4, 10)
plt.legend(('Logistic Regression Model', 'Linear Regression Model'),
          loc="lower right", fontsize='small')

plt.show()
```

# Regressão Logística

## Exercício

- Programa exemplo de regressão usando `scikit-learn`: **Logistic Function**





# Referências

- J. Demsar,  
Statistical Comparisons of Classifiers over Multiple Data Sets,  
J. of M. Learning Research, 7:1-20, 2006.
- S. Salzberg,  
On Comparing Classifiers: Pitfalls to avoid and recommended  
approach,  
Data Mining and Knowledge Discovery, 1:317-327, 1997.
- Luiz E. S. Oliveira,  
**Avaliando Classificadores,**  
Notas de Aulas, DInf / UFPR, 2017.

# Referências

- Charrad et al.,  
**NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set,**  
Journal of Statistical Software, 61, 2014.
- Luiz E. S. Oliviera,  
**Aprendizado Não-supervisionado,**  
Notas de Aulas, DInf / UFPR, 2017.
- Luiz E. S. Oliviera,  
**Regressão,**  
Notas de Aulas, DInf / UFPR, 2017.

# Referências

- Wikipedia, **Sensitivity & Specificity**  
[https://en.wikipedia.org/wiki/Sensitivity\\_and\\_specificity](https://en.wikipedia.org/wiki/Sensitivity_and_specificity)
- DZone - Ai Zone  
**Machine Learning: Validation Techniques**  
<https://dzone.com/articles/machine-learning-validation-techniques>