

# T319 - Introdução ao Aprendizado de Máquina: *Regressão Linear (Parte V)*



***Inatel***

Felipe Augusto Pereira de Figueiredo  
felipe.figueiredo@inatel.br

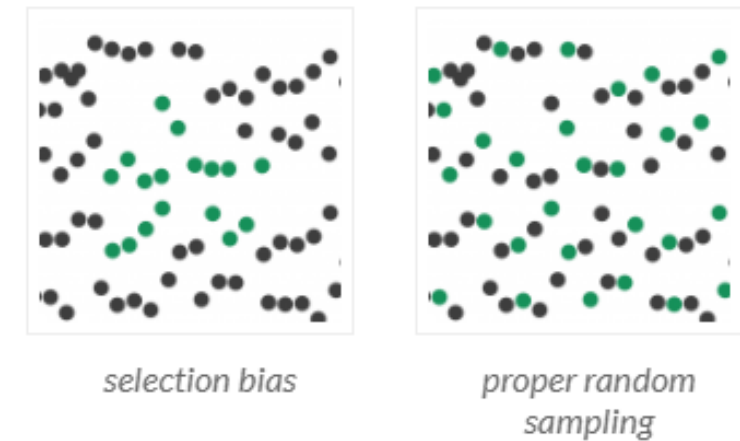
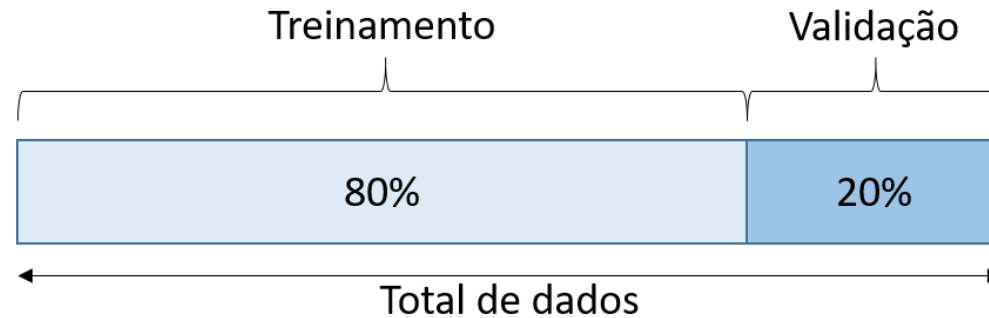
# Recapitulando

- Vimos que o ***escalonamento de atributos*** ajuda a acelerar o aprendizado do algoritmo do gradiente descendente quando os atributos têm intervalos de variação muito diferentes.
- Aprendemos que ***funções hipótese polinomiais*** podem ser utilizadas para aproximar funções que não têm um mapeamento linear.
- Porém, ***precisamos encontrar a ordem ideal para o polinômio aproximador***.
  - Polinômios de ordem muito baixa podem não ter flexibilidade o suficiente para aproximar os dados, o que causa ***subajuste***.
  - Polinômios de ordem muito alta podem ser tão flexíveis que acabam memorizando os dados de treinamento, o que causa ***sobreajuste***.
- Na sequência, veremos como escolher a ***ordem da função hipótese polinomial*** quando não conhecemos o ***mapeamento verdadeiro***.

# Validação cruzada

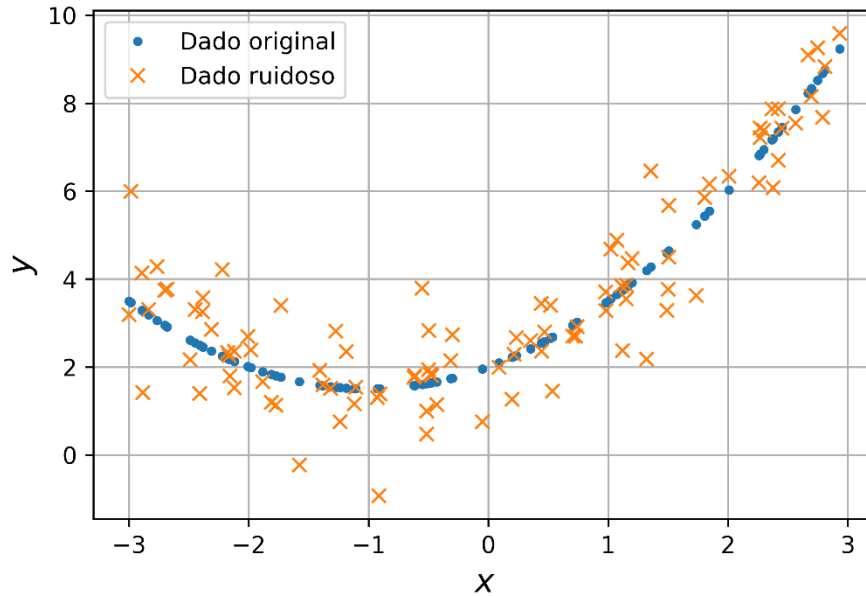
- **Validação cruzada** é uma forma de se avaliar **quantitativamente** o sobreajuste ou subajuste de um modelo e, com isso, **encontrar sua ordem ideal**.
  - Ou seja, podemos verificar quais ordens do polinômio fazem o modelo se ajustar demais ou insuficientemente aos exemplos de treinamento.
- Para realizar a **validação cruzada**, nós dividimos o conjunto total de exemplos em dois outros conjuntos, o de treinamento e o de validação (ou teste) do modelo.
- O objetivo da **validação cruzada** é encontrar um ponto de equilíbrio entre a **flexibilidade** e o **grau de generalização** da **função hipótese polinomial**.
  - Flexibilidade o suficiente para se ajustar à função verdadeira (**medida através do erro de treinamento**).
  - Grau de generalização: capacidade de gerar saídas próximas às verdadeiras para exemplos não vistos durante o treinamento (**medido através do erro de validação**).
- As estratégias para validação cruzada mais utilizadas são:
  - Holdout
  - k-fold
  - Leave-p-out

# Holdout

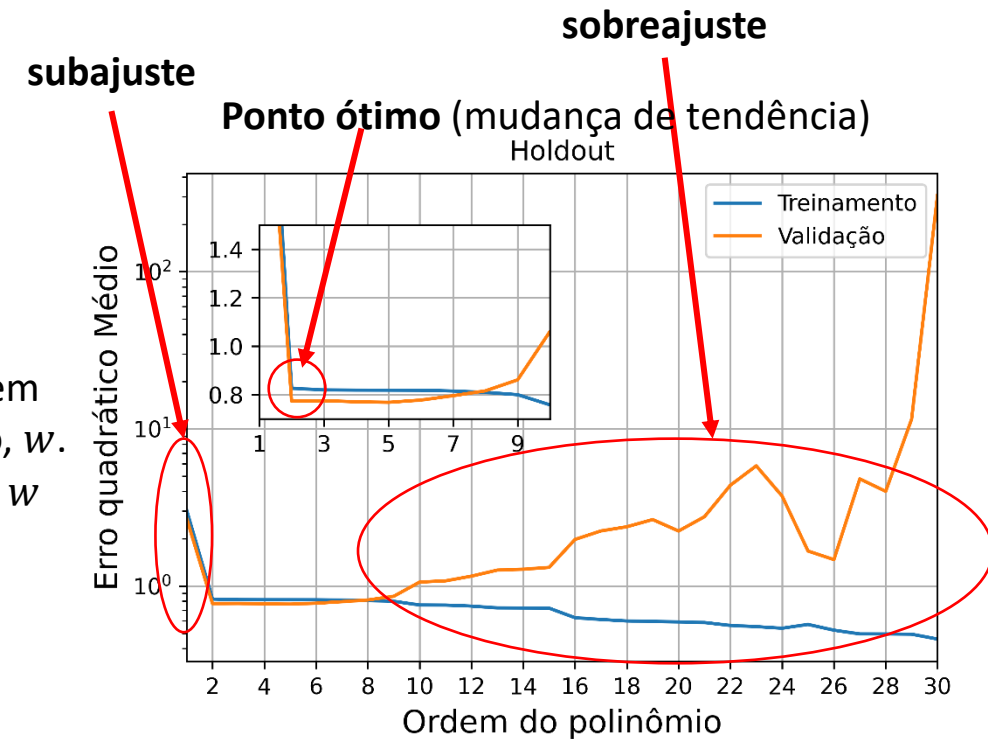


- É a estratégia ***mais simples*** das três e apresenta a menor complexidade computacional, pois realiza-se ***apenas um treinamento e uma validação***.
- Divide-se ***aleatoriamente*** o conjunto total de dados em  $p$  % para treinamento e  $(100 - p)$  % para validação.
  - Normalmente, divide-se o conjunto total de dados em 70/80% para treinamento e 30/20% para validação.
- Entretanto, devemos nos assegurar que os conjuntos de treinamento e validação sejam suficientemente ***representativos do mapeamento verdadeiro*** que se pretende aproximar.
- **Desvantagem**
  - Pode sofrer com o problema do ***viés de seleção***: a qualidade do modelo pode depender muito de quais exemplos vão para o conjunto de treinamento e quais vão para o conjunto de validação.
  - Portanto, o desempenho do modelo pode ser significativamente diferente dependendo de como a divisão é feita, ou seja, os resultados podem depender de uma escolha aleatória particular dos exemplos dos conjuntos de treinamento e validação.

# Holdout: Exemplo



Função observável é um polinômio de segunda ordem mais ruído Gaussiano branco,  $w$ .  
 $y_{noisy} = 2 + x + 0.5x^2 + w$



- 70% para conjunto de treinamento e 30% para conjunto de validação.
- Tempo médio para execução com  $N = 100$  é de aproximadamente 160 ms.
- Erro de treinamento **diminui** conforme a ordem do polinômio aumenta.
- Erro de validação **aumenta** conforme a ordem do polinômio aumenta.
- Qual ordem escolher?
  - O ponto onde **ambos** os erros sejam mínimos (balanço entre flexibilidade e grau de generalização) e com menor complexidade.

# k-Fold

- Estratégia mais elaborada que o Holdout e que nos **fornece indicações mais claras**.
- Consiste em **embaralhar** (opcional) e **dividir o conjunto total de dados em  $k$  subconjuntos** (ou *folds*) de tamanhos iguais (se possível) e realizar  **$k$  treinamentos distintos**, onde cada um dos  **$k$  treinamentos** considera  **$k-1$  folds** para treinamento e **1 fold** para validação.

**$k = 5$**

	Total de dados				
Treinamento 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Treinamento 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Treinamento 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Treinamento 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Treinamento 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Treinamento

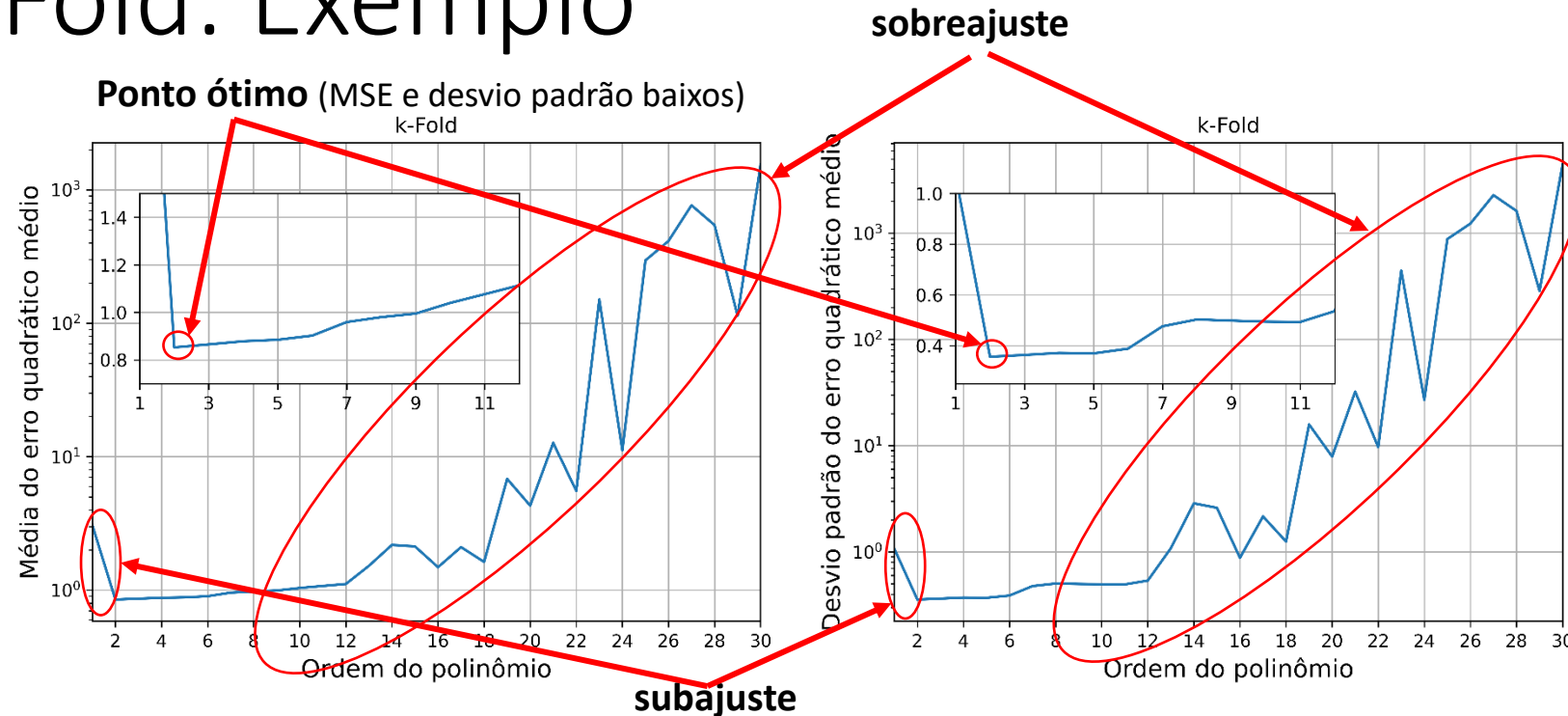
Validação

- Cada exemplo entra em um conjunto de validação exatamente **1** vez e em um conjunto de treinamento  **$k-1$**  vezes.
- O desempenho do modelo é dado pela **média dos erros de validação** calculados para cada um dos  **$k$  folds** usados para validação do modelo.
- Uma alta **variância do erro de validação** é um forte indicador de sobreajuste.

# k-Fold

- Como regra geral, normalmente, se utiliza  $k = 5$  ou  $10$ .
- Porém, tenha em mente que o valor de  $k$  deve ser escolhido de forma que os conjuntos de treinamento e validação sejam grandes o suficiente para serem ***estatisticamente representativos*** do mapeamento verdadeiro.
- O k-Fold é bastante útil quando se tem conjuntos de dados pequenos a moderados.
- **Vantagem**
  - Reduz significativamente o problema do ***viés de seleção*** em relação ao ***holdout***.
    - Pois faz-se a avaliação do modelo através de uma média de  $k$  avaliações.
    - Todos os exemplos do conjunto total de dados aparecem nos conjuntos de treinamento e validação.
- **Desvantagem**
  - O treinamento deve ser executado novamente do zero  $k$  vezes, o que significa que leva-se aproximadamente  $k$  vezes mais tempo que o ***holdout*** para se realizar a avaliação do modelo (treinamento + validação).

# k-Fold: Exemplo



Conforme o modelo se **sobreajusta** aos dados de treinamento, a **variância do erro de validação aumenta, devido a redução de seu grau de generalização** (modelo aprendido se distancia muito do modelo gerador).

Modelos com altíssimo grau de flexibilidade (maior do que o necessário) apresentam variância do erro de treinamento muito baixa e variância do erro de validação muito alta (**sobreajuste**).


Modelos com **baixíssimo grau de flexibilidade** (menor do que o necessário) têm ambas as variâncias altas (**subajuste**).

- Usa-se a mesma função observável do exemplo anterior.
- $k = 10$  folds: 10 iterações com 9 grupos para treinamento e 1 para teste.
- Tempo médio para execução com  $N = 100$  exemplos é de aproximadamente 1.5 s.
- Gráficos mostram a média e desvio padrão do MSE para as 10 etapas de treinamento/validação.
- Média e desvio padrão do MSE aumentam com a ordem do polinômio.
- Qual ordem escolher?
  - O ponto onde **ambos**, média e desvio padrão do MSE, sejam mínimos.



# Leave-p-out

- Valida um modelo usando ***todas as combinações possíveis*** de ***p*** exemplos como ***conjunto de validação*** e os ***N-p*** exemplos restantes como ***conjunto de treinamento***.
- Para um conjunto de dados com ***N*** amostras e um valor ***p***, essa estratégia produz

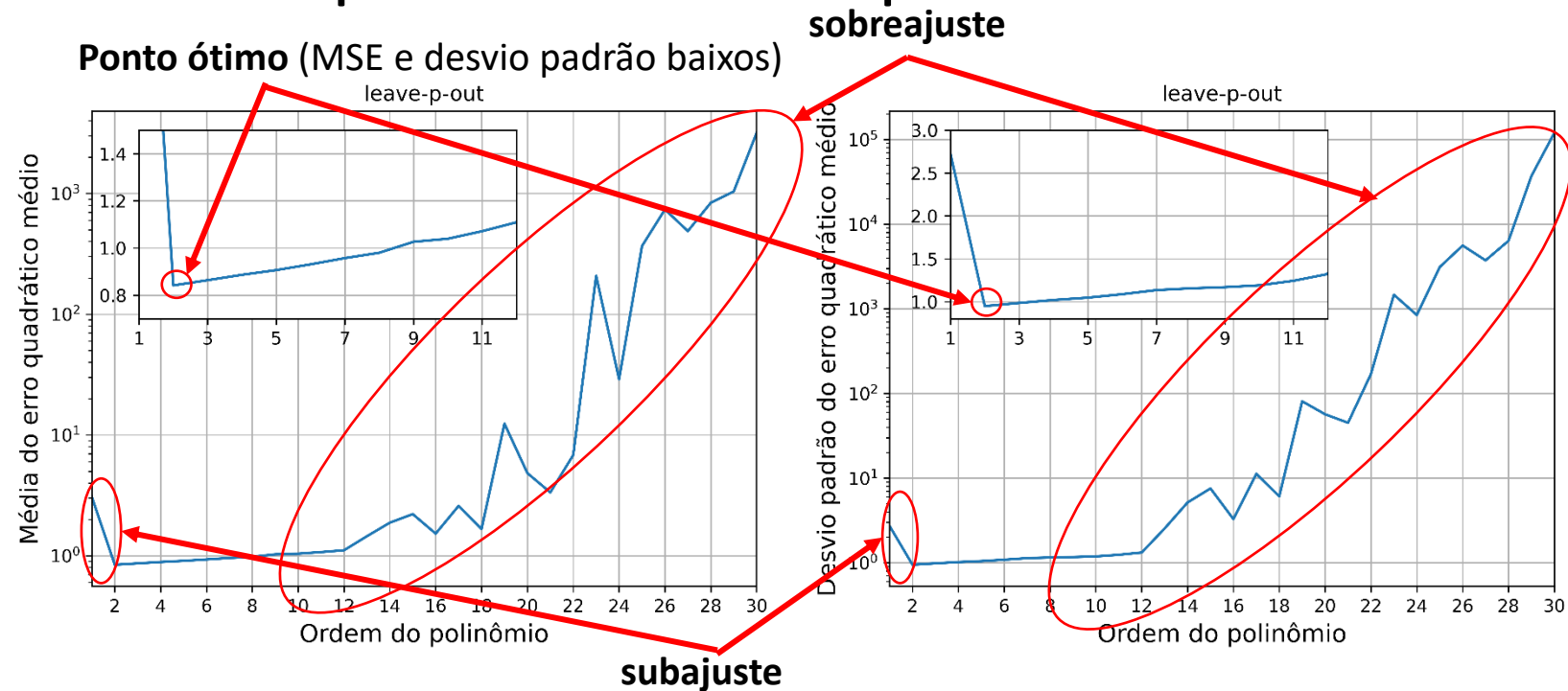
Quantos subconjuntos de *p* exemplos  
podemos criar a partir de *N* exemplos? 

$$\binom{N}{p} = \frac{N!}{p!(N-p)!},$$

pares de conjuntos treinamento/validação, portanto, a ***complexidade computacional desta estratégia aumenta drasticamente com o aumento de p***.

- Exemplos para um número total de amostras, ***N***, igual a 100:
  - ***p*** = 1 -> 100 combinações, ou seja, 100 treinamentos e validações.
  - ***p*** = 2 -> 4.950 combinações, ou seja, 4.950 treinamentos e validações.
  - ***p*** = 5 -> 75.287.520 combinações, ou seja, 75.287.520 treinamentos e validações.
- ***Fornece estimativas de erro e desvio padrão muito mais precisas do que as abordagens anteriores***, pois tem-se mais etapas de treinamento/validação.
- **Desvantagem**
  - É uma ***estratégia exaustiva***, pois treina e valida o modelo para todas as combinações possíveis e, para uma base de dados grande e um valor de ***p*** moderadamente grande, pode se tornar inviável computacionalmente.

# Leave-p-out: Exemplo



- Para ordem igual a 1, a média e desvio padrão são elevados: **subajuste**.
- Conforme a ordem aumenta, ambos diminuem, atingindo o **ponto ótimo** quando igual a 2.
- Porém, conforme a ordem continua a aumentar, ambos aumentam, indicando **sobreajuste**.

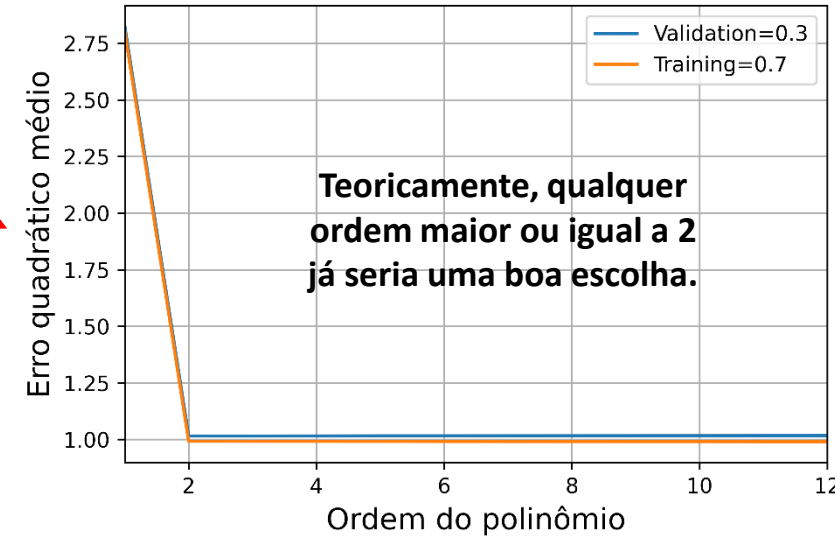
- Usa-se a mesma função observável do exemplo anterior.
- $p = 2$ : 4950 combinações possíveis com 98 exemplos para treinamento e 2 para validação.
- Tempo médio para execução com  $N = 100$  é de aproximadamente 700 [s] ( $\approx 12$  [m]).
- Gráficos mostram a média e desvio padrão do MSE para as 4950 etapas de treinamento/validação.
- Média e desvio padrão do MSE aumentam com a ordem do polinômio.
- Qual ordem escolher?
  - O ponto onde **ambos**, média e desvio padrão do MSE, sejam mínimos.

# Qual estratégia utilizar?

- Dentre as três estratégias, o ***leave-p-out*** dá indicações mais claras de ***qual ordem usar***, pois usa um número maior de pares treinamento/validação, ***aumentando a confiabilidade dos valores da média e do desvio padrão do MSE***.
- Porém, ele é bastante custoso em relação ao tempo necessário para se executá-lo, mesmo com uma base de 100 amostras leva-se quase 5 minutos.
- Portanto, deve-se utilizá-lo com bases de dados relativamente pequenas.
- Para bases maiores, o ***k-fold*** é uma opção melhor e mais eficiente do que o ***holdout*** e ***leave-p-out***.
- Para bases muito grandes, o ***holdout*** já daria boas indicações sobre qual ordem utilizar, pois a probabilidade dos conjuntos serem representativos é maior.

# Qual ordem escolher para o modelo?

- Nos exemplos anteriores foi fácil definir a ordem, mas *qual ordem escolher se os erros de treinamento e validação são pequenos, similares e praticamente constantes para várias ordens de polinômio?*
  - Isso ocorre quando o número de amostras é muito maior do que a complexidade (i.e., ordem) do modelo.
- A resposta é aplicar o princípio da **navalha de Occam**.
- A **navalha de Occam** é um **princípio lógico** que afirma que de múltiplas explicações possíveis para um fenômeno, a explicação mais simples é geralmente a mais provável de ser a correta.
  - Ou seja, deve-se preferir explicações mais simples às mais complicadas.
- Portanto, usando a **navalha de Occam** escolhemos a função hipótese *menos complexa (i.e., a mais simples), mas que se ajusta bem aos dados*.



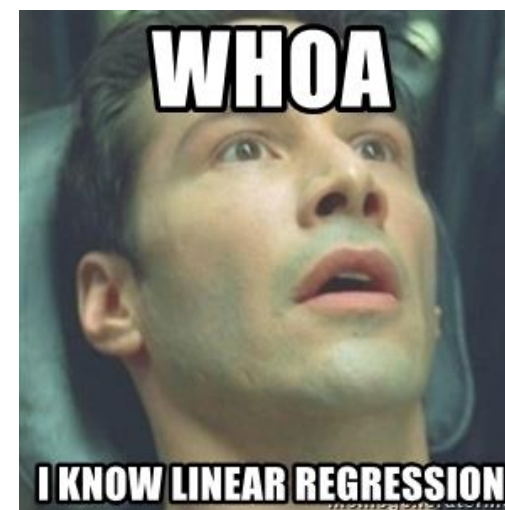
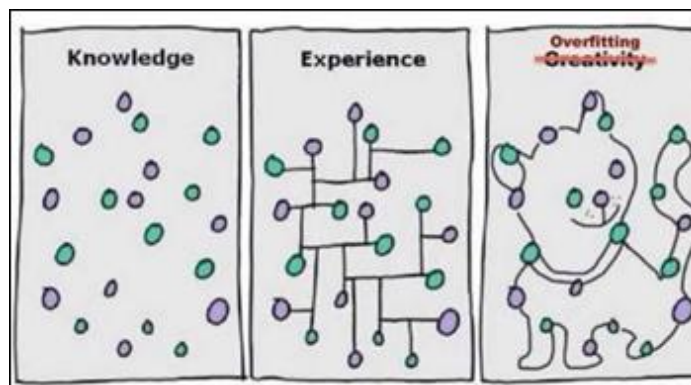
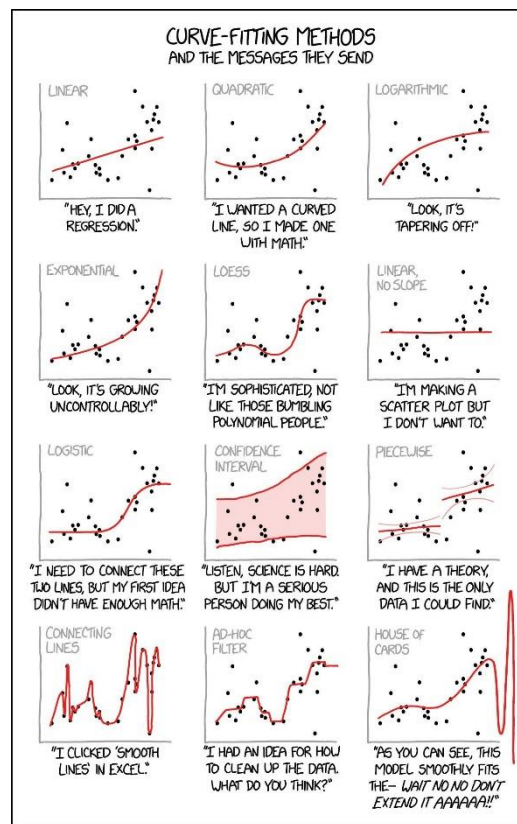
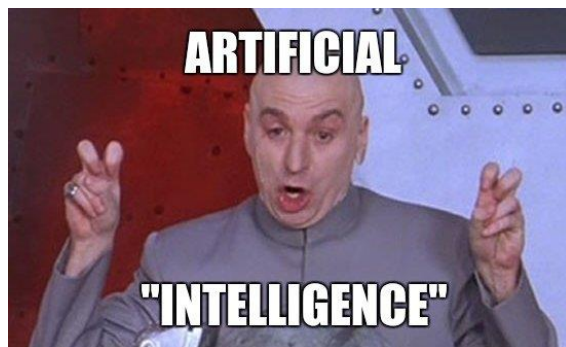
- Mesma função observável dos exemplos anteriores.
- Base de dados com **10000 exemplos**:
  - Evita o sobreajuste, pois os modelos têm complexidade muito menor do que o número de exemplos.
- Holdout com 30% para validação.
- **Qual ordem escolher?**

# Tarefas

- **Quiz:** “*T319 - Quiz - Regressão: Parte V*” que se encontra no MS Teams.
- **Exercício Prático:** [Laboratório #6](#).
  - Pode ser acessado através do link acima (Google Colab) ou no GitHub.
  - Vídeo explicando o laboratório: Arquivos -> Material de Aula -> Laboratório #6
  - [Instruções para resolução e entrega dos laboratórios](#).
- **Projeto Final**
  - Projeto pode ser feito em grupos de no máximo 3 alunos.
  - **Entrega: 10/12/2023 até às 23:59.**
  - Leiam os enunciados do trabalho atentamente.

Obrigado!





FIGURAS



