

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



Inatel

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

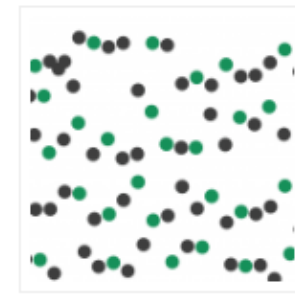
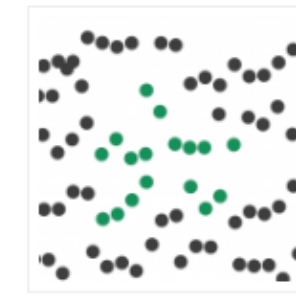
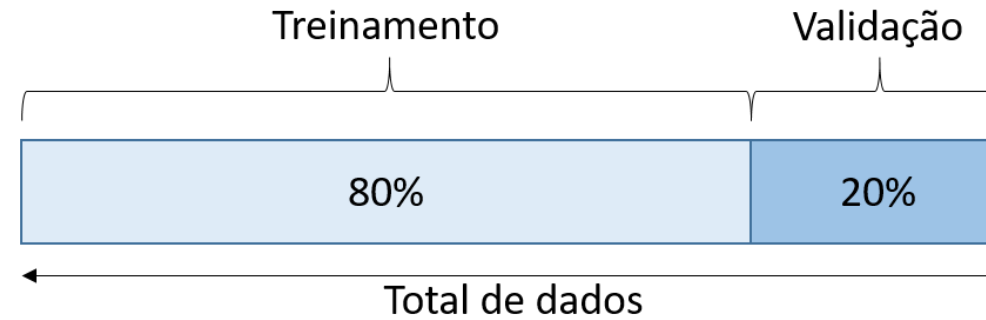
Resumindo

- 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 dados que não são lineares.
- Porém, precisamos encontrar a ordem ideal para o polinômio aproximador.
 - Polinômios de ordem baixa podem não ter flexibilidade o suficiente para aproximar os dados, o que causa **subajuste**.
 - Polinômios de ordem alta podem ser tão flexíveis que acabam memorizando os dados de treinamento, o que causa **sobreajuste**.
- Hoje veremos como escolher a ordem da **função hipótese polinomial** quando não conhecemos o **mapeamento verdadeiro**.

Validação cruzada

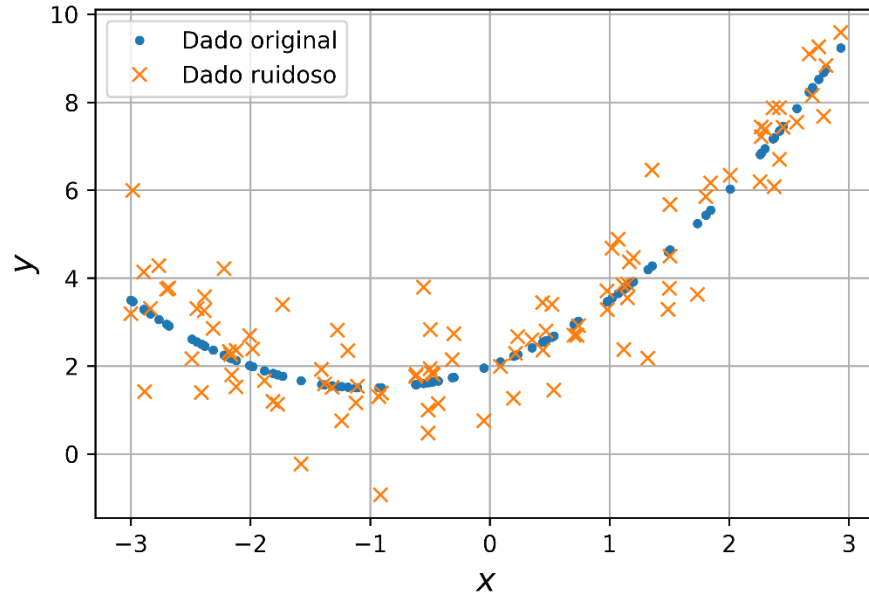
- **Validação cruzada** é uma das formas de se avaliar quantitativamente o sobreajuste ou subajuste de um modelo e, com isso, **encontrar sua ordem ótima**.
- Ou seja, podemos verificar quais ordens fazem o modelo se ajustar demais ou insuficientemente aos exemplos de treinamento.
- Na **validação cruzada**, nós dividimos o conjunto de exemplos em 2 outros conjuntos, o de treinamento e o de validação (ou teste) do modelo.
- O objetivo é testar a capacidade do modelo em prever as saídas para exemplos que não foram utilizados durante o treinamento (conjunto de validação), ou seja, a capacidade do modelo em **generalizar**.
- As estratégias para validação cruzada mais utilizadas são:
 - Holdout
 - k-fold
 - Leave-p-out

Holdout

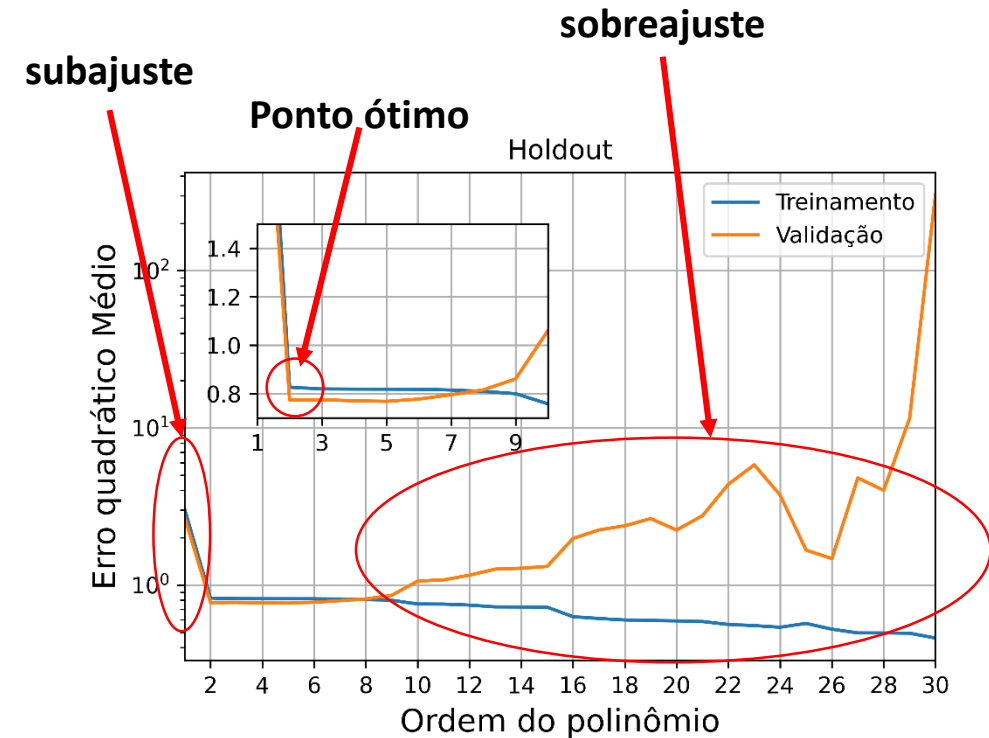


- É a estratégia mais simples das 3 e não acarreta em aumento da complexidade computacional, pois tem-se apenas um único par de conjuntos de treinamento e validação.
- Devemos nos assegurar que os conjuntos de treinamento e validação sejam suficientemente **representativos** do mapeamento verdadeiro que se pretende aproximar.
- 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.
- **Desvantagem**
 - Pode sofrer com o problema do **viés de seleção**: a validação 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 .
 $y = 2 + x + 0.5x^2 + w$



- 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).

k-Fold

- Estratégia mais elaborada que a anterior.
- Consiste em dividir o conjunto total de dados em **k** folds (subconjuntos) 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.

	← 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

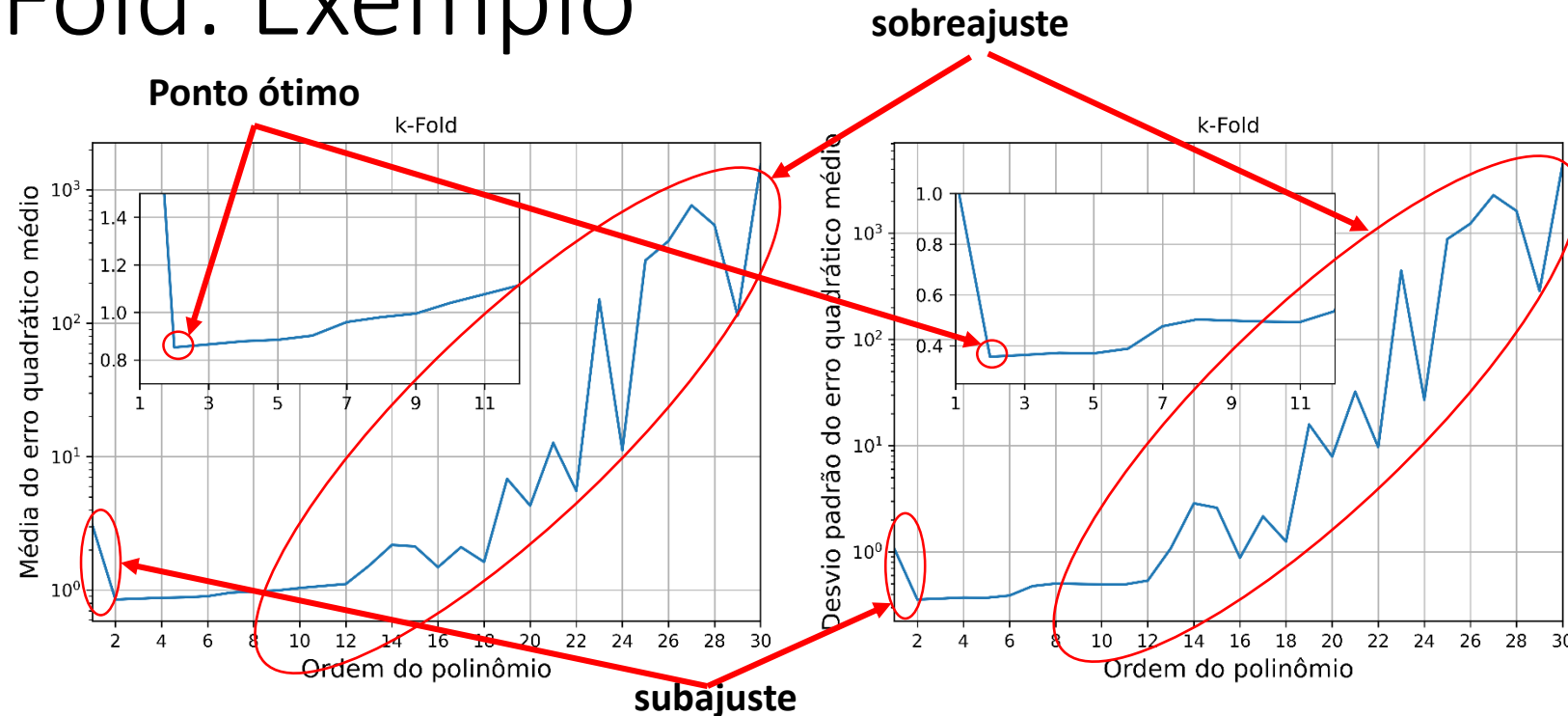
Validação

- 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.

k-Fold

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

k-Fold: Exemplo



Conforme o modelo se **sobreajusta** aos dados de treinamento, sua variância aumenta, devido a redução de seu grau de generalização.

Em teoria, a variância deve ser igual a 0 para modelos com alto grau de generalização.

- 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 1.9 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.

[Exemplo: validacao_cruzada.ipynb](#)

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, essa estratégia produz

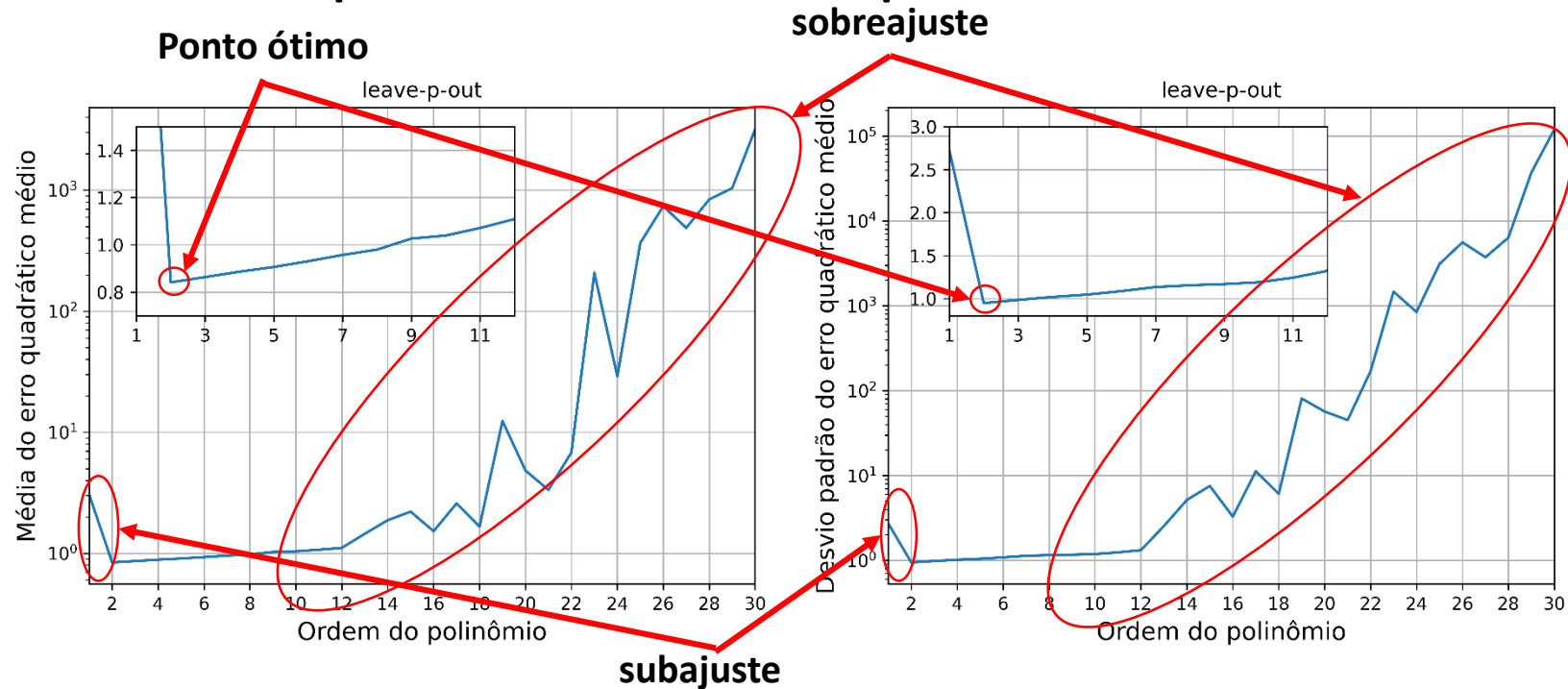
$$\binom{N}{p} = \frac{N!}{p!(N-p)!},$$

Quanto subconjuntos de p exemplos posso criar a partir de N exemplos?

pares de conjuntos treinamento/teste, portanto, a complexidade computacional desta estratégia aumenta drasticamente com o aumento de ***p***. Exemplos para $N = 100$:

- $p = 1 \rightarrow 100$ combinações
- $p = 2 \rightarrow 4.950$ combinações
- $p = 5 \rightarrow 75.287.520$ combinações
- Fornece estimativas de erro e desvio padrão mais precisas do que as abordagens anteriores, pois tem-se mais etapas de treinamento/validação.
- **Desvantagem**
 - É uma estratégia exaustiva no sentido de que ela 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.
- No caso do k-Fold, quando fazemos ***k=N*** (número folds igual ao número total de exemplos), então o k-Fold é equivalente à estratégia do leave-one-out, ou seja, ***p*** = 1.

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**.

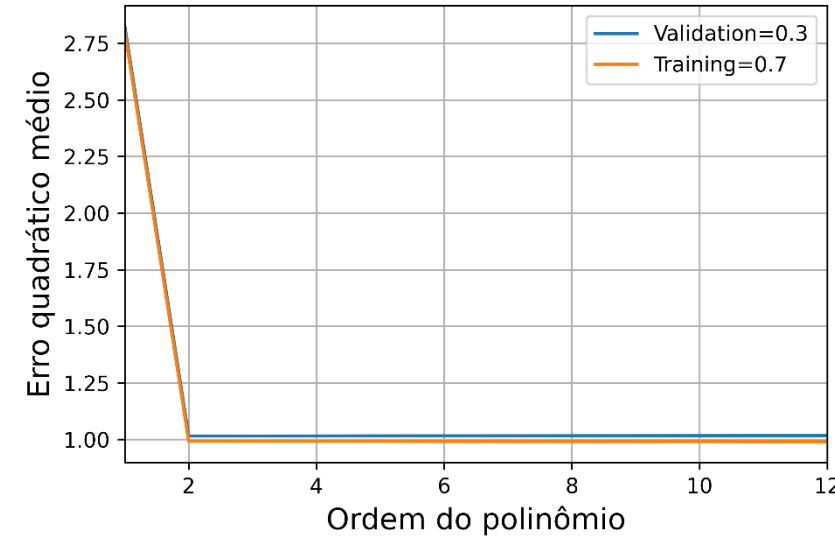
- Usa-se a mesma função observável do exemplo anterior.
- $p = 2$: 4950 combinações possíveis com 98 exemplos para treinamento e 2 para validação.
- Tempo médio para execução com $N = 100$ é de aproximadamente 810 [s] (+ de 13 [m]).
- Gráficos mostram a média e desvio padrão do MSE para as 4950 etapas 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?

- O **leave-p-out** dá indicações mais claras de qual ordem usar, pois usa um maior número de pares treinamento/validação, aumentando a confiabilidade 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 mais de 13 minutos!
- Portanto, deve-se utilizá-lo com bases relativamente pequenas.
- Para bases maiores, o **k-fold** é uma opção melhor e mais eficiente do que o **holdout**.
- Para bases muito grandes, o **holdout** já daria boas indicações sobre qual ordem utilizar.

Qual ordem escolher para o modelo?

- E se os erros de treinamento e validação são pequenos e praticamente constantes para várias ordens de polinômio?
- Uma resposta é usar o princípio da **navalha de Occam**.
- A **navalha de Occam** é um princípio lógico que diz que deve-se preferir explicações mais simples às mais complicadas.
- Portanto, escolhemos modelos usando a **navalha de Occam**: escolhemos a **função hipótese** menos complexa que se ajusta bem aos dados.

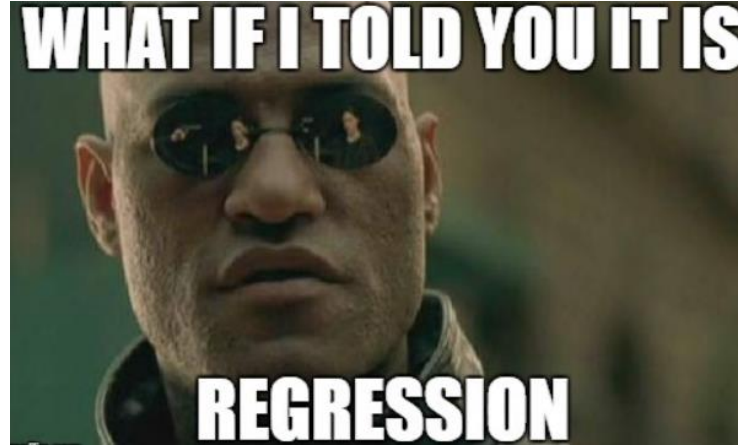
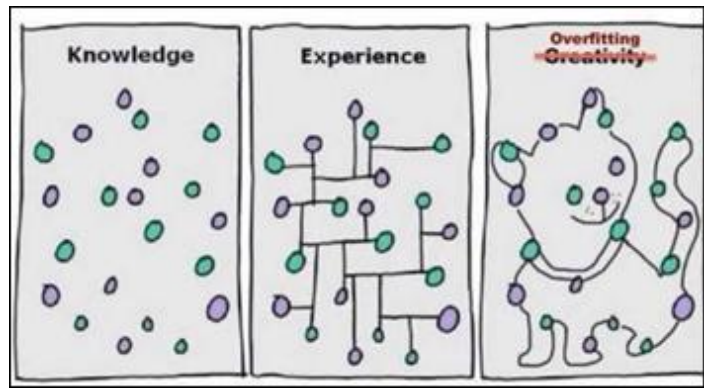
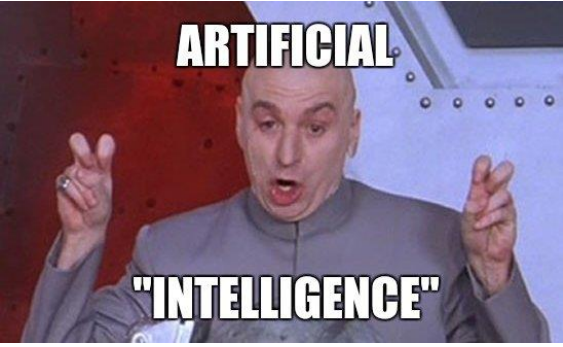
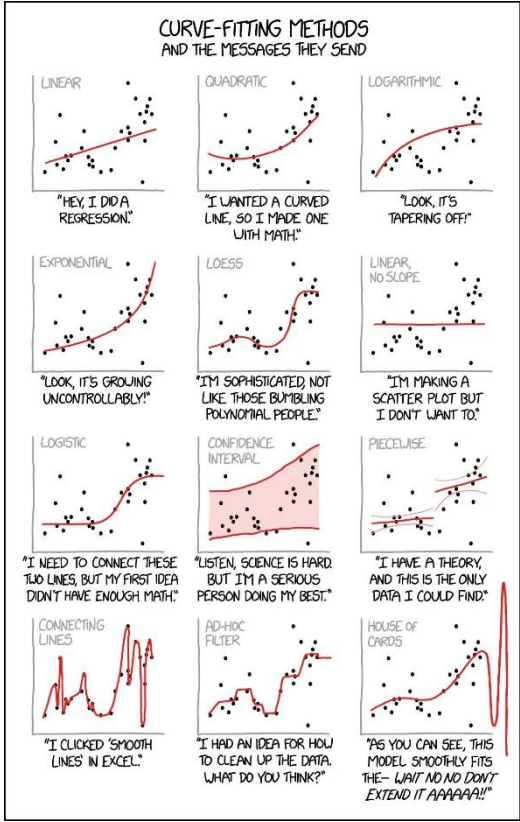
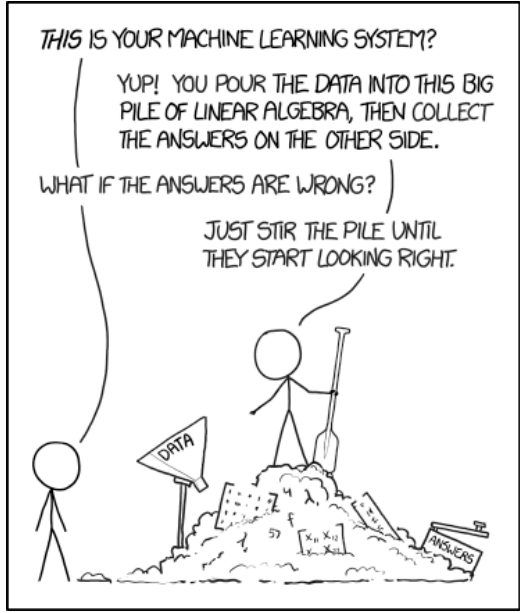


- Mesma função observável dos exemplos anteriores.
- Base de dados com **10000 exemplos**.
- Holdout com 30% para validação.
- Vejam que teoricamente, qualquer ordem maior ou igual a 2 já seria uma boa escolha.
- **Qual ordem escolher?**

Tarefas

- **Quiz:** “*T319 - Quiz - Regressão: Parte V (1S2021)*” que se encontra no MS Teams.
- **Exercício Prático:** [Laboratório #6](#).
 - Pode ser baixado do MS Teams ou do GitHub.
 - Pode ser respondido através do link acima (na nuvem) ou localmente.
 - [Instruções para resolução e entrega dos laboratórios](#).
 - **Laboratórios podem ser feitos em grupo, mas as entregas devem ser individuais.**

Obrigado!



FIGURAS

