
Tera

# Estatística e Modelagem de Dados

## Aula 18: Cross-validation & Regularization

# Instrutora

## Cristiane Rodrigues

- **Bacharel em Matemática – UNESP Rio Claro.**
- **Mestre em Estatística – USP Piracicaba**
- Experiências Profissionais:
  - Modelagem de Credito para PF e PJ – Banco Bradesco
  - Experiência com Segmentação e Análise de Series temporais – Atento
  - Consultora Analítica - SAS Institute Brasil
  - Consultora de Pré Vendas - SAS Institute Brasil
  - Professora do curso SAS Academy for Data Science

crisrodrigues_27@hotmail.com

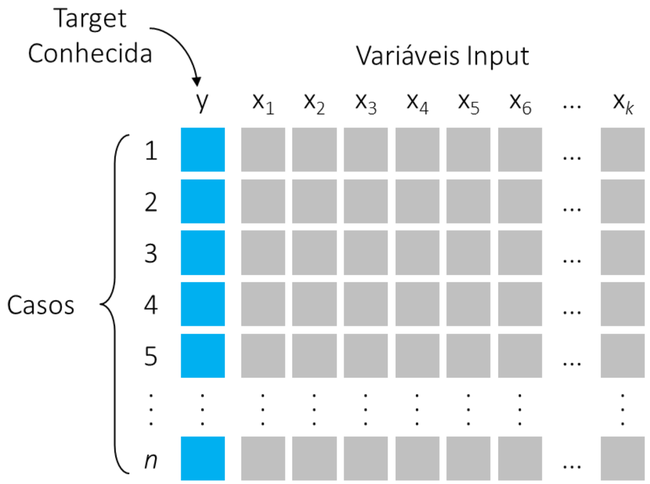
# Cross Validation

# Cross Validation - Índice

- Motivação
- Princípio do otimismo
- Divisão da base
- Overfitting/Underfitting
- Cross Validation : K-Folds Cross Validation
- Cross Validation: Outros métodos

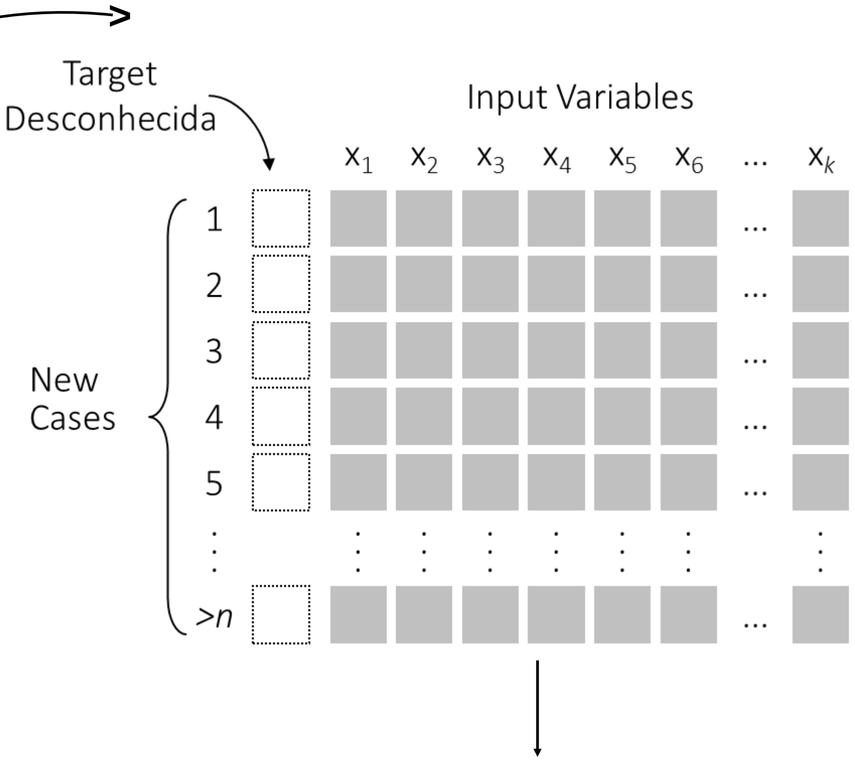
# Motivação

- O objetivo dos modelos preditivos é a generalização, ou seja, a habilidade de prever a variável resposta para novos casos




**Desenvolvimento de um modelo estatístico**

**Obtenção da equação que descreve a relação entre a variável Target e as variáveis Input**

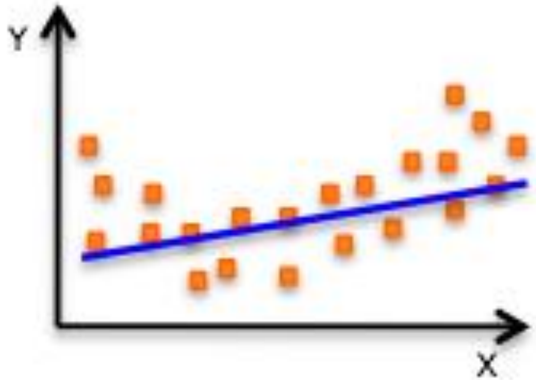



**Generalização: Obtenção da variável resposta para novos casos**

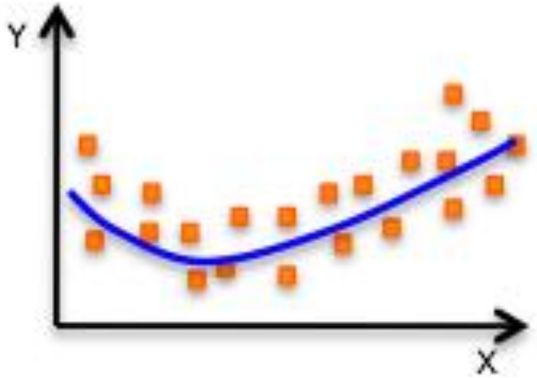
# Motivação

- Não basta apenas ajustar um modelo, precisamos verificar se este modelo está bem ajustado, ou seja, se a generalização está ocorrendo de forma efetiva.

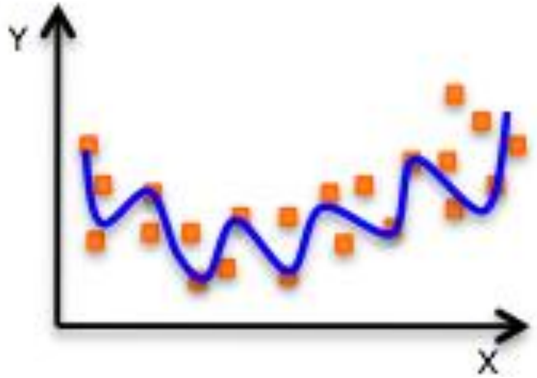
- Como podemos fazer isto?
- Como verificar se o modelo obtido
  - é Simples demais
  - é Complexo demais
  - está OK




**Simples Demais**
**Underfitting**



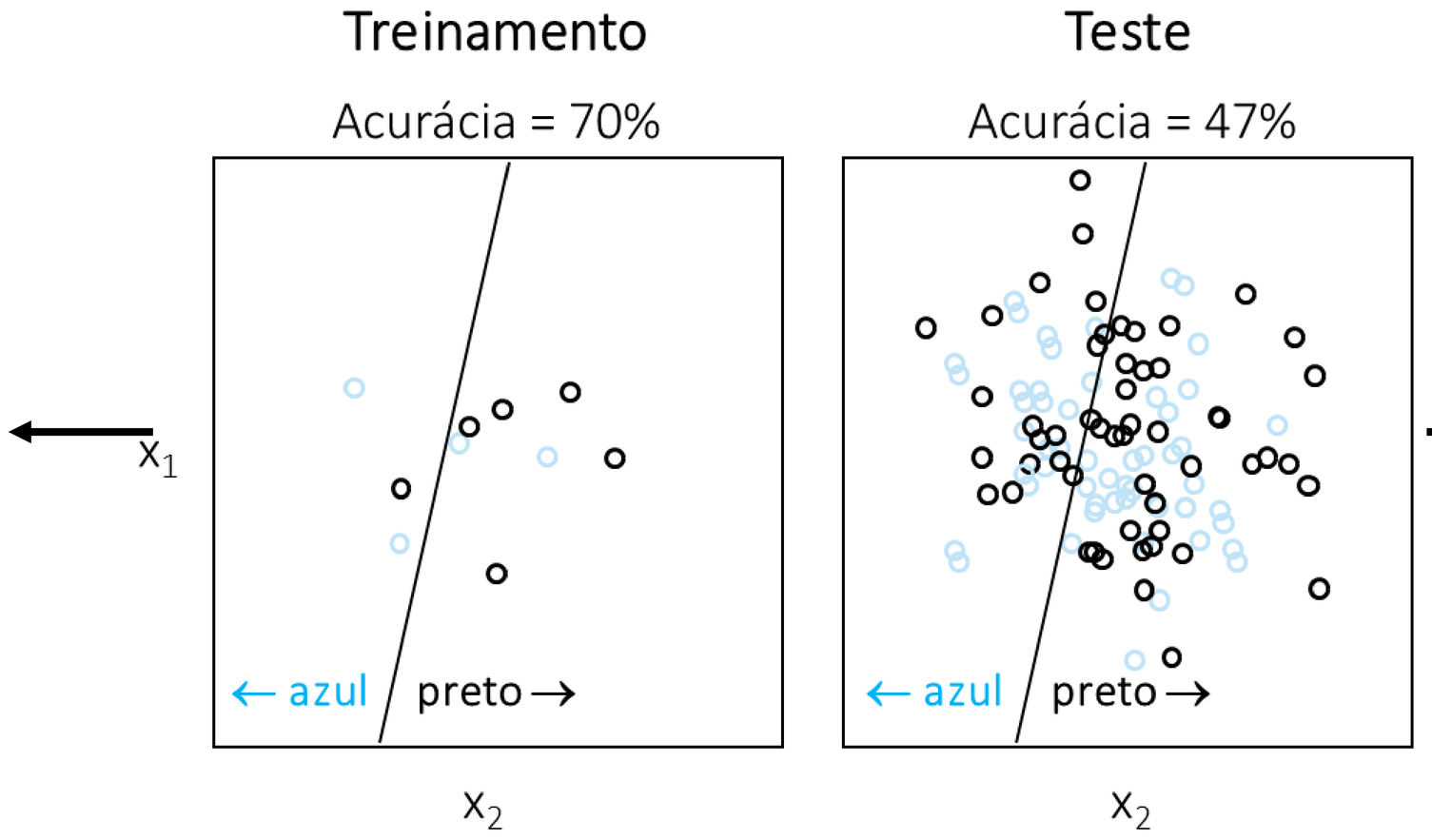

**OK**
**Just Right**




**Complexo Demais**
**Overfitting**

# Princípio do otimismo

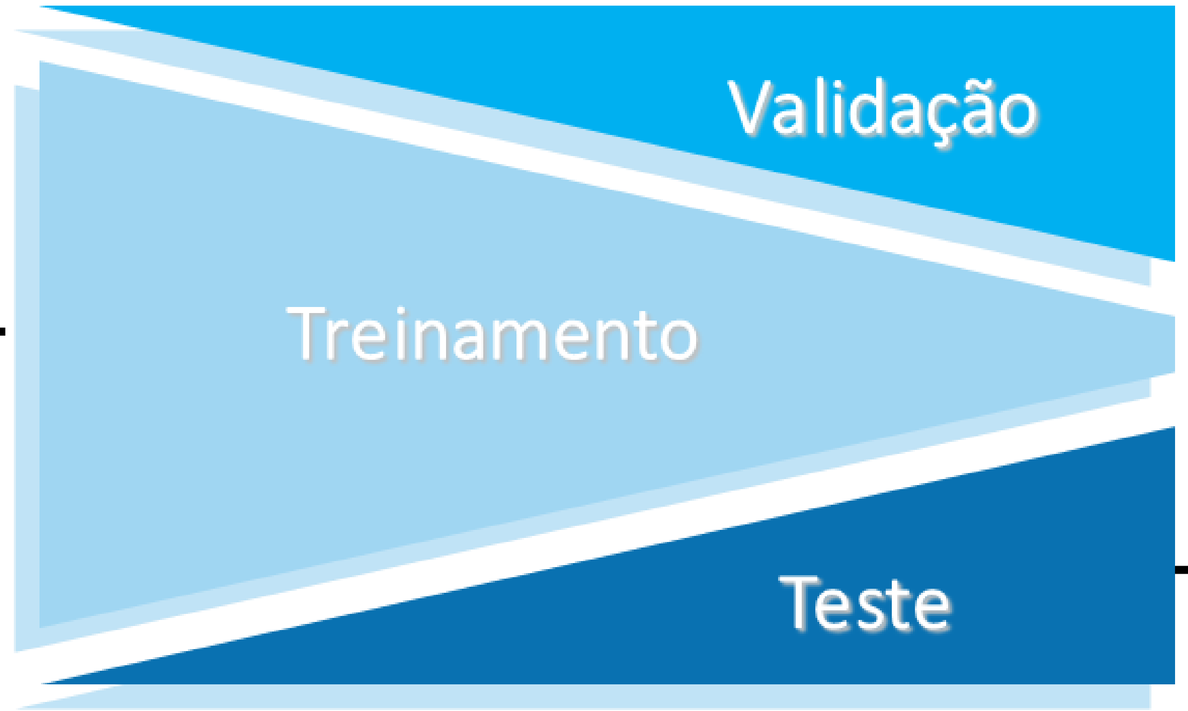
**Base utilizada para desenvolver o modelo**




**Base utilizada para generalizar os resultados do modelo**

# Divisão da Base de Dados

- Para não ter problemas com o princípio do otimismo uma boa prática na modelagem é dividir a base de dados em treinamento, validação e teste.




**Base utilizada ajustar vários modelos**

**Verifica a qualidade do ajuste para cada um dos modelos ajustados na base de treinamento e escolhe o melhor modelo: Comparação, Seleção e Ajuste**

**Valida o modelo escolhido, pela base de validação.**

# O que é Overfitting/Underfitting em um Modelo?

- Quando ajustamos um modelo nos dados de treinamento, uma das duas coisas pode acontecer:
  - superajustar o modelo
  - subajustar o modelo.

- Não queremos que essas coisas aconteçam, pois elas afetam a previsibilidade do modelo. A ideia é encontrar um modelo que se comporta de forma parecida na base de treinamento e de validação.

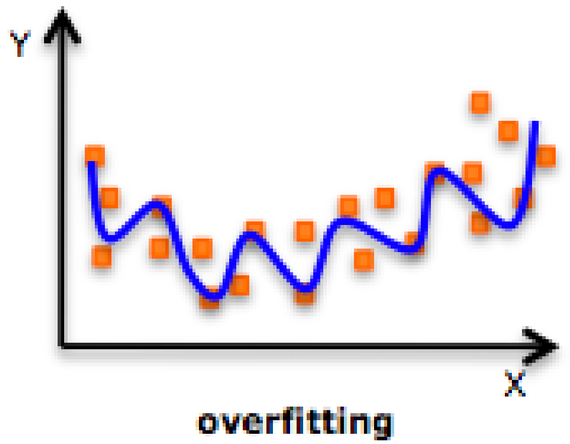
# Overfitting

- O Overfitting significa que o modelo que treinamos treinou "muito bem", ou seja, se ajusta muito bem ao conjunto de dados de treinamento.

- Por que o overfitting ocorre?

  - Geralmente porque o modelo ajustado é muito complexo, ou seja, utiliza muitas características/variáveis em relação ao número de observações. Este modelo é muito preciso nos dados de treinamento, mas provavelmente não será preciso em dados novos.

  - Porque o modelo aprende ou descreve o "ruído" nos dados de treinamento em vez das relações reais entre as variáveis nos dados. Esse ruído, obviamente, não faz parte de nenhum novo conjunto de dados, e não pode ser aplicado a ele.



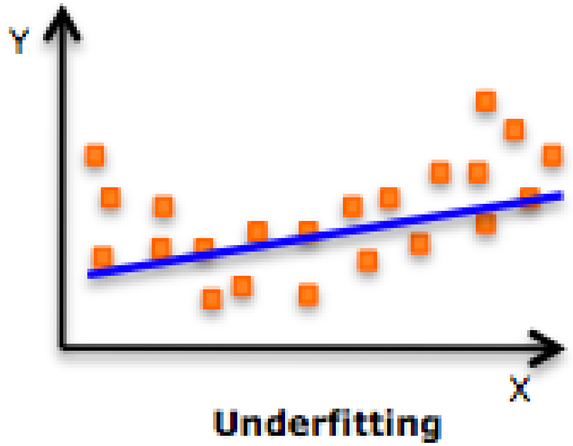
# Underfitting

- O Underfitting significa que o modelo ajustado não corresponde aos dados de treinamento e, portanto, perde as tendências nos dados e não pode ser generalizado para novos dados.

- Por que o underfitting ocorre?

  - Geralmente é o resultado de um modelo muito simples, o qual não tem variáveis preditoras/independentes suficientes.

  - Pode ser resultado do uso de um modelo linear para dados que não são lineares. O que acarretará em uma capacidade de previsão pobre em dados de treinamento e não pode ser generalizado para outros dados.



# Estudo de Caso
# Cross Validation no Python

**Fonte da dados**: 


**Link**: http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html
**Resumo**: 442 pacientes com diabetes foram medidos para 10 variáveis (idade, sexo, bmi (índice de massa corporal), map (pressão arterial média), seis medidas de soro sanguíneo)
**Objetivo**: Ajustar um modelo de previsão, em uma base de treinamento, para a medida da progressão da doença um ano após o ponto de partida (variável resposta), fazer a previsão desta resposta e avaliar a qualidade de ajuste do modelo em uma base de teste.

# Estudo de Caso
# Cross Validation no Python

**Parte_1: Dividindo a base em treinamento e teste**
**Parte_2: Ajustando um modelo linear a base de treinamento**

# Divisão da Base de Dados – Problemas?

- Parece bom, certo?

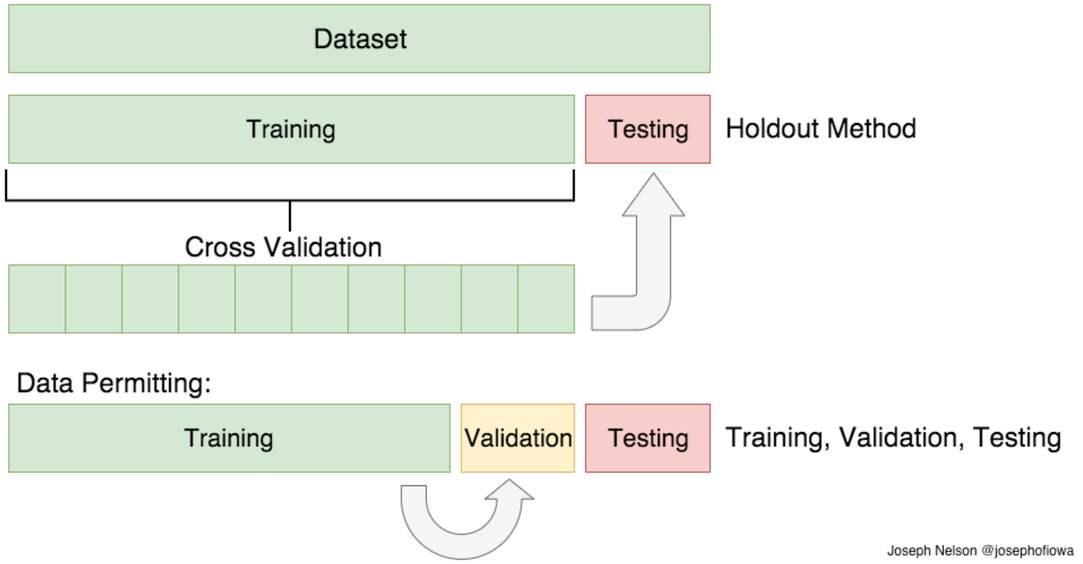
- Mas a divisão em treinamento/validação (teste) tem seus perigos:
  - E se a divisão não for aleatória?
  - E se um subconjunto dos dados tiver apenas pessoas de um determinado estado ou funcionários com um certo nível de renda, mas não outros níveis de renda, apenas mulheres ou apenas pessoas com certa idade?

  Isso pode resultar em overfitting, mesmo que tentemos evitá-lo!

- É aqui que entra a validação cruzada.

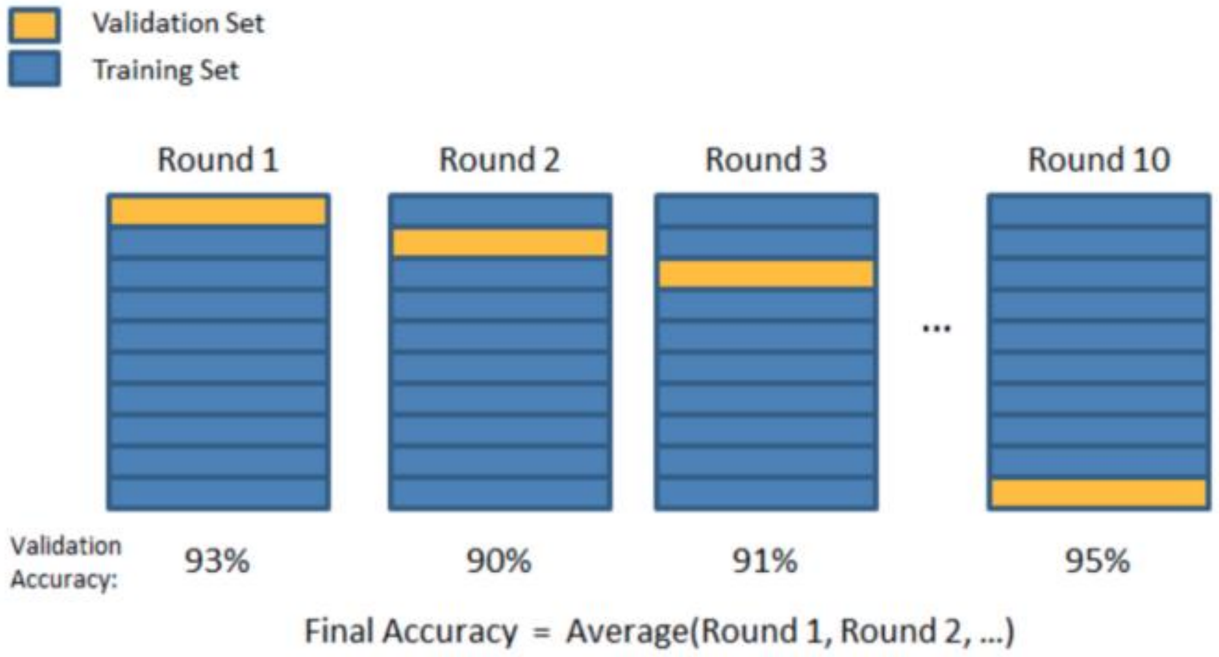
# Cross Validation

- É muito parecida com a divisão de treinamento/teste, mas é aplicado a mais subconjuntos.
  Dividimos os dados em k subconjuntos  e treinamos em k-1 um desses subconjuntos.  Sempre mantendo o último subconjunto para teste.
  A ideia é ir alternando até que todos os subconjuntos tenham sido usados para teste.



# K-Folds Cross Validation

1. Dividir os dados em k subconjuntos diferentes (ou dobras).
2. Usar k-1 subconjuntos para treinar os dados e deixar o último subconjunto (ou a última dobra) como teste.
3. Medir o modelo contra cada uma das dobras e finalizá-lo.
4. Testar o modelo na base de teste.
5. A acurácia final é obtida pela acurácia média de cada um dos modelos ajustado.



# Estudo de Caso
# Cross Validation no Python

## Parte_3: Performando K-Folds Cross Validation

# Cross Validation – Outros métodos

- Leave One Out Cross Validation (LOO)
- K-Fold estratificada
- Leave P Out (LPO)
- Leave One Label Out (LOLO)
- Leave P Label Out (LPLO)

# Desafio Alunos
# Cross Validation no Python

1. Dividir os dados em train e validate
2. Use a base de treinamento para ajustar um modelo (ou mais modelos)
3. Avalie o(s) modelo(s) usando a base de validação
4. Utilize um método de validação cruzada para verificar a qualidade de ajuste no modelo

# Regularization

# Regularization - Índice

- Contextualização
- Regularization
- Métodos
  - Ridge Regression – L2 Norm
  - Lasso Regression – L1 Norm

# Contextualização

- Ter mais variáveis input pode parecer uma maneira perfeita de melhorar a precisão do modelo treinado, pois este será mais flexível e levará em conta mais parâmetros.
- Por outro lado, precisamos ser extremamente cuidadosos para não ter um overfit.
- Cada conjunto de dados é composto de sinal e ruído.
  - Por exemplo, o tamanho da casa não foi medido com precisão ou o preço não está atualizado.
  - As imprecisões/ruídos podem levar a um modelo de baixa qualidade se não forem treinadas com cuidado.
  - O modelo pode memorizar o ruído em vez de aprender o sinal.

- Overfit também pode ocorrer em modelos lineares quando se tem muitas inputs.
  Se as inputs não forem filtradas e exploradas antecipadamente, algumas podem ser mais prejudiciais do que úteis
  - inputs redundantes: informações “repetidas”
  - inputs irrelevantes: adicionam alto ruído ao conjunto de dados.

# Regularization

- Overfit é um problema extremamente comum em machine learning e existem diferentes abordagens para resolvê-lo. O principal conceito para evitar overfit é simplificar os modelos o máximo possível. Modelos simples (geralmente) não geram overfit. Por outro lado, precisamos prestar atenção ao delicado trade-off entre overfitting e underfitting do model.

- Um dos mecanismos mais comuns para evitar overfit é chamado de regurarization, o qual penaliza variáveis que não colaboram significantemente para o ajuste do modelo, fazendo com que seus coeficientes tendam a zero.

- A ideia é encontrar um conjunto ótimo de parâmetros que minimizam uma função de erro, a qual contém outro elemento além do elemento comum da regressão linear, ou seja, vamos penalizar a magnitude dos coeficientes

- Veremos dois métodos de regularização: Ridge e LASSO Regression

# Ridge Regression

- É um método de regularização do modelo que tem como principal objetivo suavizar variáveis que sejam relacionadas umas as outras e que aumentam o ruído no modelo.

- Com a suavização de algumas variáveis, o modelo converge para um resultado muito mais estável sem alterar a acurácia.

- Nessa regressão a ideia é que os coeficientes β sejam pequenos, mas não força que estes sejam zero, isto é, não exclui as variáveis irrelevantes, mas minimiza seu impacto no modelo treinado.

- É uma extensão da regressão linear, mas com um termo de penalidade, o qual é um escalar que deve ser bem calibrado usando métodos de validação cruzada.

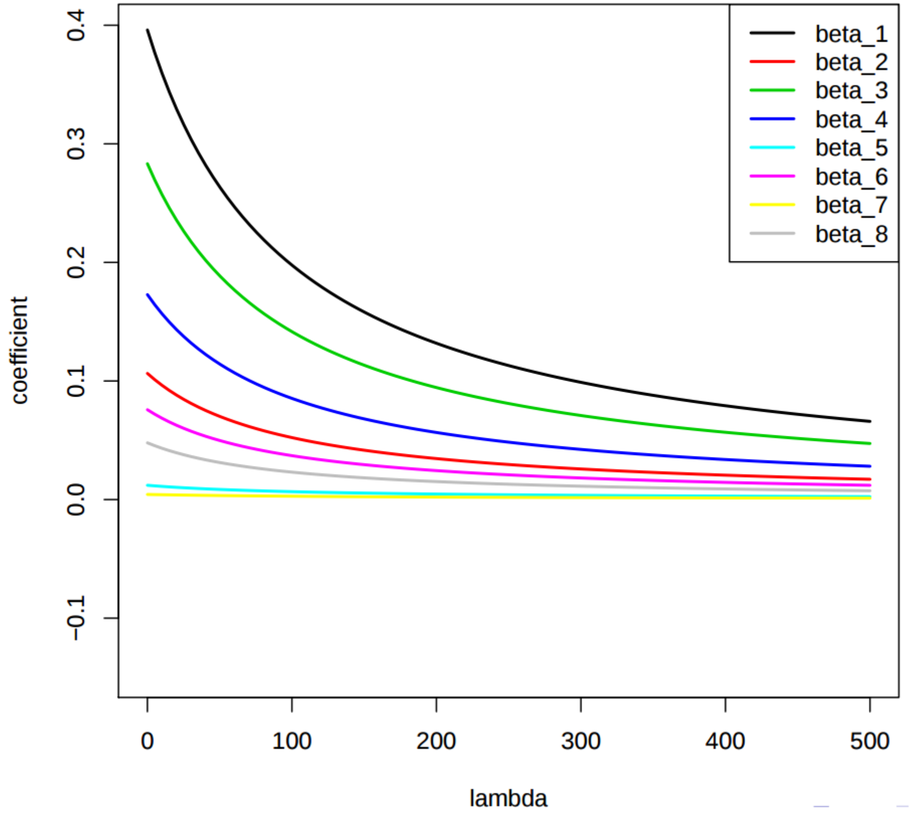
# Ridge Regression

- Como o processo funciona?

A ideia é penalizar os coeficientes utilizando a L2 Norm, ou seja, minimizar a soma dos quadros dos resíduos somado a L2 Norm (soma dos β's ao quadrado multiplicado por λ)

$$\sum(\widehat{Y_i} - Y_i)^2 + \alpha \sum \beta^2$$

em que $\alpha$ é o parâmetro de tuning ou ajuste da penalidade e os β's são os coeficientes estimados.



# Lasso Regression

- LASSO (Least Absolute Shrinkage and Selection Operator) Regression

- Como dito anteriormente também é um método de regularização que evita overfitting penalizando os coeficientes com um alto grau de correlação entre si.

- Nessa regressão o grande diferencial é o fato de poder reduzir alguns dos coeficientes β’s exatamente a zero, o que naturalmente vai eliminar a variável e reduzir a dimensionalidade do modelo, realizando assim uma seleção de variáveis.

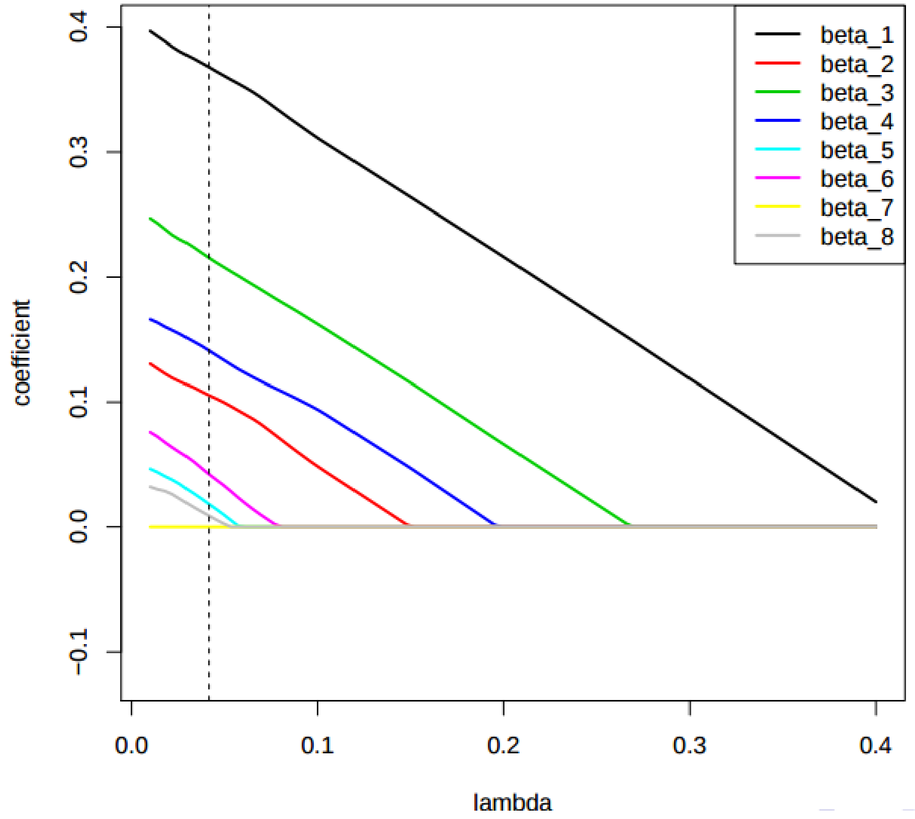
# LASSO Regression

- Como o processo funciona?

A ideia é penalizar os coeficientes utilizando a L1 Norm, ou seja, minimizar a soma dos quadros dos resíduos somado a L1 Norm (soma do valor absoluto dos β's multiplicado por λ)

$$\sum(\widehat{Y_i} - Y_i)^2 + \alpha \sum |\beta|$$

em que $\alpha$ é o parâmetro de tuning ou ajuste da penalidade e os β's são os coeficientes estimados.



# Quando usar Ridge ou Lasso Regression

Agora temos uma boa ideia de como funciona a regressão ridge e lasso.
Vamos tentar consolidar nosso entendimento fazendo uma comparação entre eles

Para isso usaremos os seguintes tópicos:

1. Principais diferenças
2. Caso de Uso
3. Presença de variáveis altamente correlacionadas

# Ridge ou Lasso Regression – Principais Diferenças

• Ridge: inclui todos dos recursos no modelo.
A principal vantagem dessa regressão é a diminuição na magnitude dos coeficientes e a redução da complexidade do modelo.

• Lasso: Junto com a diminuição na magnitude dos coeficientes, o lasso também executa a seleção de variáveis.
Como observado, alguns dos coeficientes se tornam exatamente zero, o que é equivalente a exclusão da variável do modelo.

Tradicionalmente, técnicas como stepwise são usadas para realizar a seleção de variáveis e fazer modelos parcimoniosos.
Com os avanços em Machine Learning, as regressões ridge e lasso fornecem alternativas muito boas, gerando saídas muito melhores, exigindo menos ajustes nos parâmetros e podem ser automatizadas em grande extensão.

# Ridge ou Lasso Regression – Caso de Uso

• Ridge: É usado principalmente para evitar overfitting.
Como inclui todas as variáveis, não é muito útil no caso de termos uma quantidade muito grande de variáveis, digamos em milhões, pois representará desafios computacionais.

• Lasso: Como fornece soluções esparsas, geralmente é escolhido para modelar casos em que o número de variáveis está no ordem de milhões ou mais. Nesse caso, obter uma solução esparsa (sparse solution) é de grande vantagem computacional, já que as variáveis com coeficientes zero são simplesmente ignoradas.

Técnicas de seleção como stepwise tornam-se muito complicadas de implementar em casos de alta dimensionalidade. Nisso, o lasso proporciona uma vantagem significativa.

# Ridge ou Lasso Regression – Variáveis Altamente Correlacionadas

• Ridge: Geralmente funciona bem mesmo na presença de variáveis altamente correlacionadas, pois incluirá todas no modelo, mas os coeficientes serão distribuídos entre eles, dependendo da correlação.

• Laço: Seleciona arbitrariamente qualquer variável entre as altamente correlacionadas e reduz o coeficiente das demais a zero. Além disso, a variável escolhida muda aleatoriamente com a mudança nos parâmetros do modelo. Isso geralmente não funciona tão bem em comparação com a regressão ridge.

Juntamente com o Ridge e o Lasso, o Elastic Net é outra técnica útil que combina a regularização de L1 e L2. Ele pode ser usado para balancear os prós e contras da regressão de ridge e lasso.

# Estudo de Caso
# Risge e Lasso Regression

**Parte_4: Performando LASSO Regression**
**Parte_5: Performando Ridge Regression**

# DÚVIDAS?!

# Referências


1. https://medium.com/towards-data-science/train-test-split-and-cross-validation-in-python-80b61beca4b6

2. http://scikit-learn.org/stable/modules/cross_validation.html

3. http://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/modules/cross_validation.html

4. http://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html

5. https://codingstartups.com/practical-machine-learning-ridge-regression-vs-lasso/

6. https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/

7. https://mineracaodedados.wordpress.com/2015/06/20/qual-a-diferenca-entre-lasso-e-ridge-regression/

8. http://ricardoscr.github.io/como-usar-ridge-e-lasso-no-r.html

# Obrigada

***Cristiane Rodrigues***

***crisrodrigues_27@hotmail.com***