Modelos Preditivos e Séries Temporais

Bootcamp Analista de Machine Learning

Matheus de Oliveira Mendonça

2020

Modelos Preditivos e Séries Temporais

Bootcamp Analista de Machine Learning

Matheus de Oliveira Mendonça

Sumário

Capítulo 1. Introdução

Introdução à modelagem preditiva

A modelagem preditiva pode ser definida como o conjunto de ferramentas e técnicas de extração de conhecimento a partir de dados históricos para a predição de valores futuros ou de eventos que ainda não aconteceram.

O ciclo de vida de um projeto de predição pode ser resumido em cinco pontos (ver Figura 1):

1. **Definição do problema**: definição do escopo de trabalho. Etapa na qual as seguintes perguntas são realizadas:
    a. Qual evento será modelado?
    b. Como a predição desse evento irá melhorar meu processo?
    c. Possuo dados suficientes? Se sim, onde esses dados estão?
    d. Qual será a frequência de utilização e atualização do modelo preditivo?
2. **Coleta de dados**: etapa de extração de dados, na qual diferentes fontes de dados são utilizadas para a obtenção de um dataset trabalhável.
3. **Tratamento dos dados:** remoção de dados duplicados, tratamento de valores absurdos ou faltantes, entre outras coisas.
4. **Modelagem:** escolha do algoritmo, treinamento e validação de modelos.
5. **Implantação:** implantação do modelo preditivo e monitoramento contínuo do desempenho preditivo do mesmo.

Figura 1 – Ciclo de trabalho da modelagem preditiva.

Neste curso, iremos focar nas etapas 3 e 4 do processo descrito acima, dando ênfase aos seguintes tópicos:

- Tratamento de dados:
  - Tratamento de dados faltantes;
  - Tratamento de ruídos;
  - Codificação de atributos categóricos;
  - Normalização de dados numéricos.
- Modelagem:
  - Modelos de regressão;

– Modelos de classificação de padrões;

– Modelos para séries temporais.

A etapa de modelagem faz uso de técnicas de aprendizado de máquinas para o desenvolvimento de modelos capazes de aprender a partir dos dados sem serem explicitamente programados para tal, diferentemente de técnicas tradicionais de programação que precisam de escrever regras específicas para cada caso (abordagem simbólica).

Como exemplo, considere o seguinte problema de classificação de dígitos (1 ou 2):

O uso da abordagem simbólica para este problema provavelmente resultaria em um algoritmo com o seguinte formato:

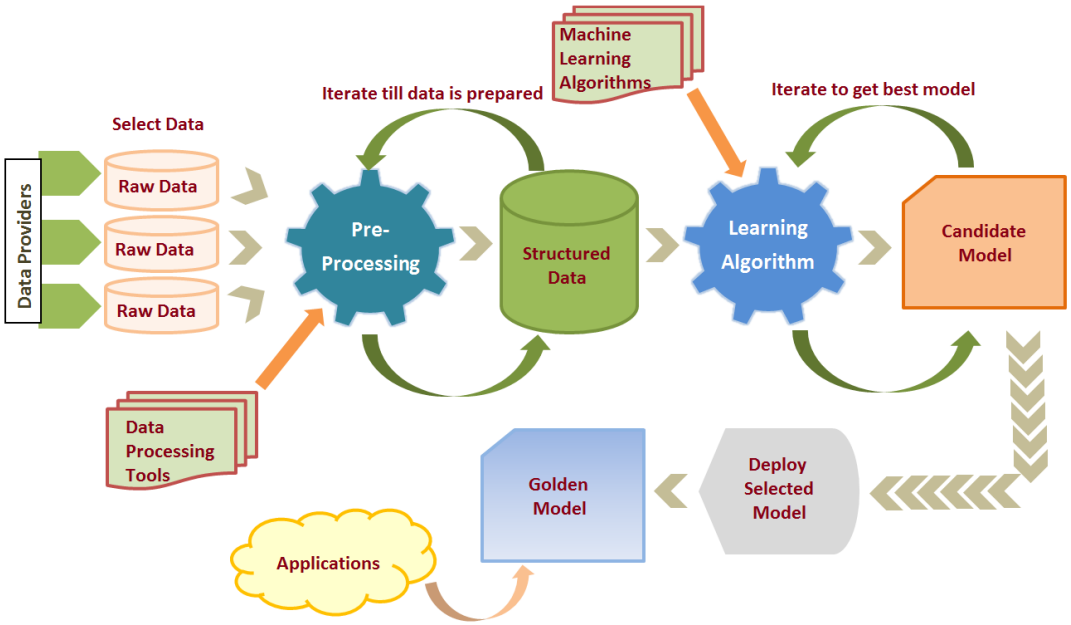
```
SE DÍGITO É COMPOSTO POR UMA RETA ENTÃO “UM”
SE DÍGITO É COMPOSTO POR UMA OU MAIS CURVAS ENTÃO “DOIS”
```

Esta abordagem modela o problema em função de regras fixas (if/else). Do ponto de vista de interpretabilidade, é bem intuitivo entender o processo decisório. Entretanto, na presença de incertezas e informações imprecisas, a capacidade de generalização (predição) desse tipo de abordagem é bem limitada, como é o caso genérico de reconhecimento de dígitos escritos à mão, onde cada pessoa possui uma caligrafia própria (fonte de ruído/incerteza para o modelo):

É nesse sentido que nasce o aprendizado de máquinas, para possibilitar o aprendizado a partir de técnicas de reconhecimento de padrões possibilitando a generalização para novos cenários, sem a necessidade da programação de uma regra explícita para cada possível cenário.

A Figura 2 apresenta um fluxograma descritivo de um projeto de aprendizado de máquinas.

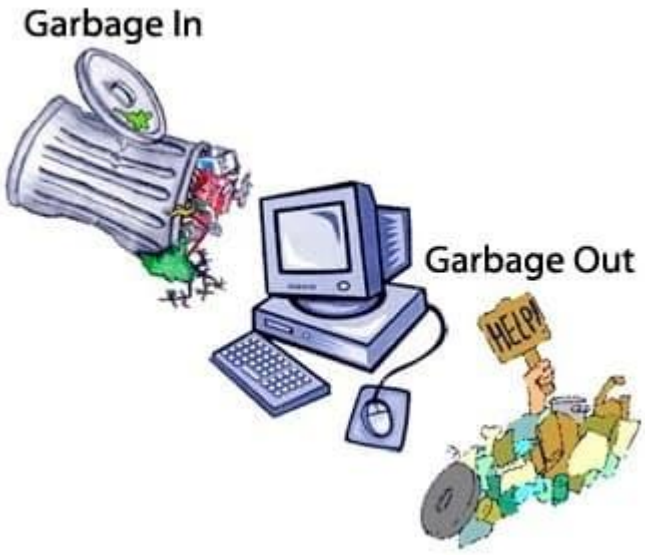
Figura 2 – Pipeline de um projeto de aprendizado de máquinas.



Fonte: https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94.

Note que os **dados históricos são a fonte de todo conhecimento** de um modelo preditivo (primeira etapa da Figura 2) e, por isso, o processo de coleta de dados deve ser feito com muito cuidado para que o modelo seja, de fato, útil. A Figura 3 traz uma ilustração do conceito "**garbage in, garbage out**".

Figura 3 – Um modelo preditivo é tão bom quanto forem os dados de entrada.



Fonte: https://www.fuzzylogx.com.au/fuzzy-friday/fuzzy-friday-part-20/.

Métodos de amostragem de dados

Dada a importância dos dados para qualidade preditiva de um modelo de dados, o processo de coleta dos dados deve ser feito de modo a garantir que:

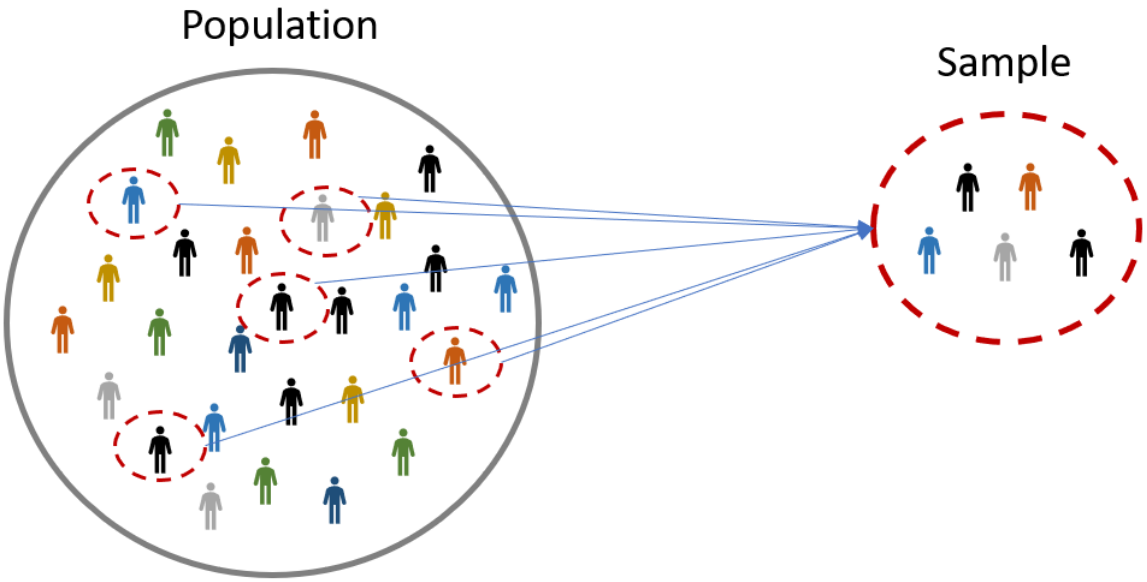
1. A **amostra** coletada seja **representativa da população** (do evento) de interesse;

2. A **amostra não contenha nenhum tipo de viés**.

Assim, a amostragem é o processo de obtenção de um conjunto de dados, que é um subconjunto da grandeza de interesse (ver Figura 4). Nem sempre as amostras refletem a estrutura da população de onde foram retiradas ou são

representativas dessas populações, podendo levar nesses casos a inferências erradas ou ao enviesamento dos resultados.

Figura 4 – População vs. Amostra.

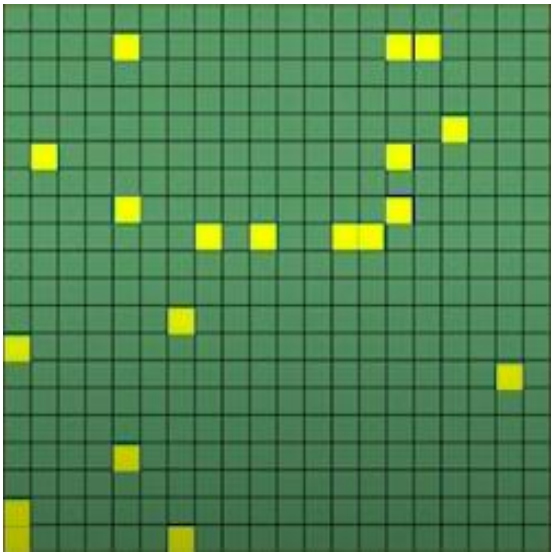


Fonte: https://www.sigmamagic.com/blogs/online-sample-size-calculators/.

A escolha do método de amostragem está diretamente relacionada com a qualidade da amostra, devendo, assim, ser escolhido com adequado. A seguir, alguns tipos de amostragem são apresentadas:

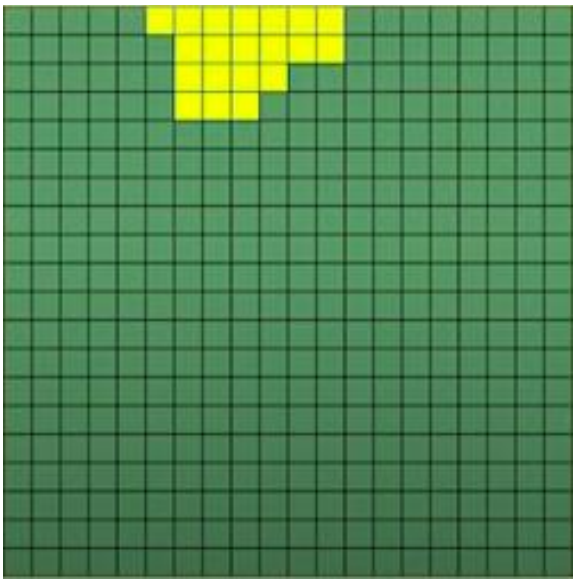
- **Amostragem aleatória simples**: é aquela em que toda amostra possível de mesmo tamanho tem a mesma chance de ser selecionada a partir da população. Este tipo de amostragem é a ideal, não favorece nenhuma observação e produz uma amostra sem viés e representativa da população. Entretanto, ela pode ser difícil de ser executada na prática (tempo e custo):

Figura 5 – Amostragem aleatória simples.



- **Amostragem por conveniência:** é aquela em que a amostra é definida por algum fator de conveniência (exemplo: proximidade da amostra). Este tipo de amostragem tende a produzir uma amostra enviesada e não representativa, mas ela é bem mais simples e rápida de ser executada que uma amostragem aleatória:

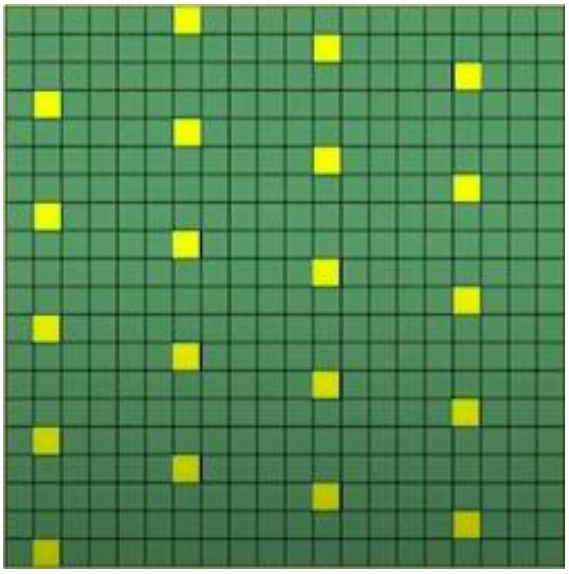
Figura 6 – Amostragem por conveniência.



- **Amostragem aleatória sistemática:** é aquela em que há um fator de aleatoriedade para determinação da primeira observação e, em seguida, uma

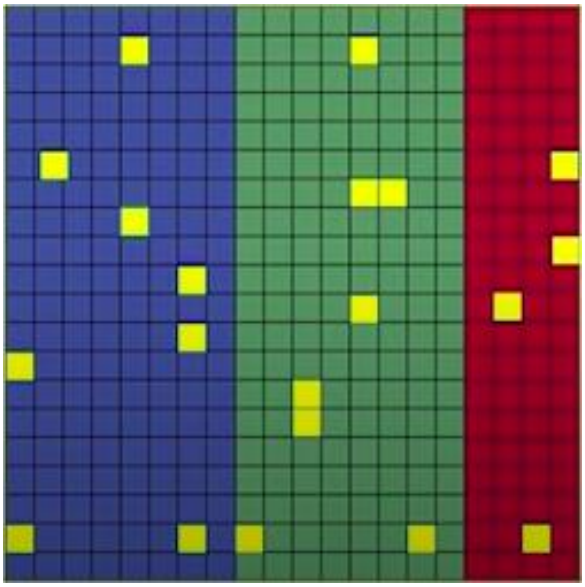
nova observação é introduzida seguindo um determinado padrão (exemplo: um nome a cada 10, 1 peça a cada 30 em uma linha de produção etc.). Esta amostragem pode falhar na presença de um algum padrão sistemático nos dados:

Figura 7 – Amostragem aleatória sistemática.



- **Amostragem aleatória estratificada:** consiste em dividir (estratificar) a população em um certo número de subpopulações que não se sobrepõem e então extrair observações aleatórias de cada estrato.

Figura 8 – Amostragem aleatória estratificada.



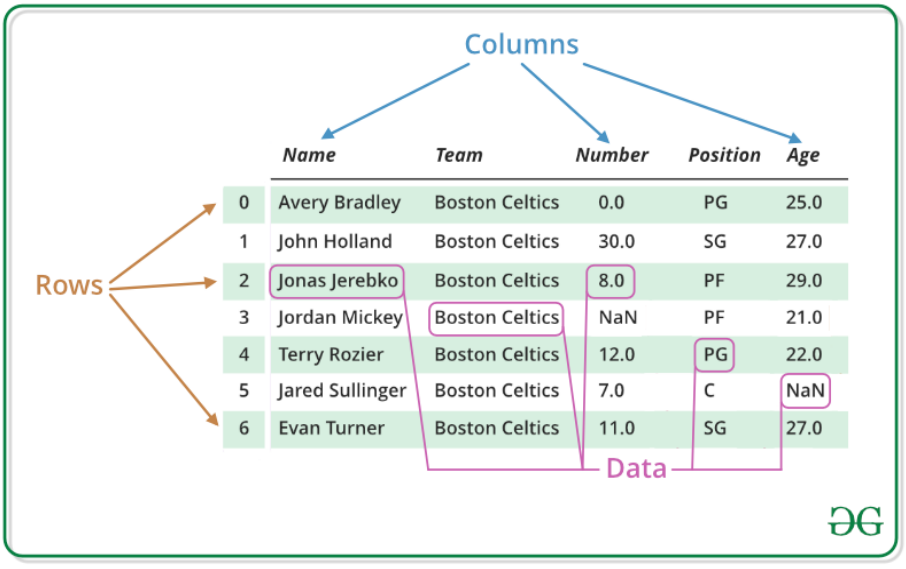
Capítulo 2. Dados estatísticos e pré-processamento

Introdução

Coletamos regularmente dados dos mais variados tipos: numéricos, strings, imagens, sons, vídeos etc. Estes dados podem estar estruturados de forma complexa, por exemplo: atributos individuais de usuários de uma rede social estruturados na forma de um grafo de amizade com arestas conectando seguidores e seguidos. Ou podemos ter dados genéticos de indivíduos organizados como árvores genealógicas em estudos de DNA. Todos estes dados podem (e são) analisados estatisticamente.

Entretanto, o tipo de dado mais comum nas análises estatísticas são aqueles organizados de forma tabular (ver Figura 9). Cada **linha da tabela** corresponde a um caso. Casos também são chamados de **observações**, **instâncias** ou **exemplos**. Cada **coluna** corresponde a uma variável. Uma variável também é chamada de **atributo** ou característica (***feature***, em inglês). A tabela de dados coletados é chamada de **amostra** (***sample***, em inglês).

Figura 9 – Exemplo de um dado tabular (*DataFrame*).

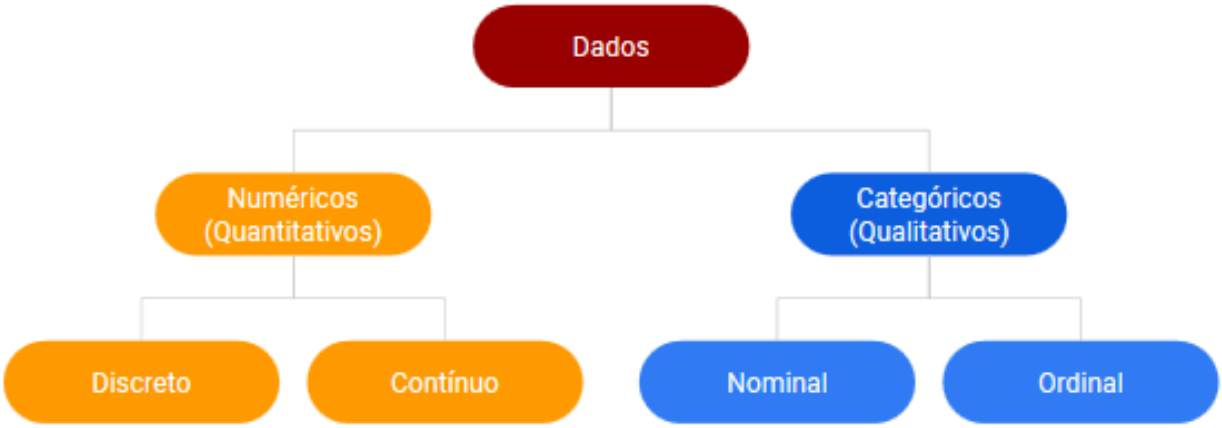


Fonte: https://www.geeksforgeeks.org/python-pandas-dataframe/.

As variáveis podem ser divididas em 4 tipos básicos (ver Figura 10):

- Variáveis numéricas (quantitativas):
  - Discreta;
  - Contínua.
- Variáveis categóricas (qualitativas):
  - Nominal;
  - Ordinal.

Figura 10 – Tipologia dos dados.



Os dados reais podem conter muitos erros, alguns estão listados a seguir:

- **Dados incompletos:** falta de informação, dados apenas agregados.
  - Cargo = ” ”.
- **Ruído**: Ex.: erros ou anomalias.
  - Salário = ”-1500”.
- **Inconsistência local**: dois atributos derivados não batem!
  - Idade = ”18”, data de nascimento=”18/09/1960”.
- I**nconsistência global**: dados indistinguíveis.

  - Todos possuem a mesma data de nascimento.

Desse modo, as principais etapas do pré-processamento dos dados são:

- Limpeza dos dados:
  - Preencher dados ausentes, “alisar” ruído, identificar e/ou remover valores aberrantes;
  - Resolver inconsistências.
- Integração e transformação de dados:
  - Integração de múltiplas bases de dados;
  - Normalização e agregação.
- Redução de dados:
  - Redução no volume de dados com resultados similares.

Algumas dessas etapas serão discutidas a seguir para cada tipo de variável.

Variáveis numéricas

Uma variável é numérica se faz sentido somar seus valores (para obter um total geral, por exemplo), subtrair (para medir a diferença entre dois casos, por exemplo) ou tomar médias de seus valores:

- **Variáveis discretas**: variáveis numéricas discretas assumem apenas alguns valores possíveis. Estes valores podem ser colocados numa lista enumerável. A lista de valores possíveis não precisa ser finita, como no caso dos inteiros, ela precisa ser enumerável. Exemplos:
  - Contagem de páginas de livros;
  - Número de carros em um cruzamento;

  - Tamanho de sapatos;
  - Etc.

- **Variáveis contínuas**: no caso de variáveis numéricas contínuas, seus valores podem assumir qualquer valor no intervalo da reta real. Exemplos:
  - Temperatura;
  - Peso;
  - Duração de um filme;
  - Etc.

Tratamento de dados faltantes

Na presença de dados faltantes e na impossibilidade de coleta de mais observações, algumas estratégias podem ser adotadas para substituição (*imputation,* em inglês) de dados numéricos:

- **"Imputação" unidimensional:**
  - Substituição pela média: a média é fortemente influenciada pela presença de valores extremos;
  - Substituição pela mediana: a mediana é uma estatística mais robusta;

Exemplo:

Figura 11 – Exemplo de tratamento de dados faltantes em atributos numéricos.

Nome	Idade	Sexo
Matheus	26	M
Letícia	22	F
Terezinha	60	F
José	-	M

Substituição pela **média** da coluna:
(26+22+60)/3 = 36

Substituição pela **mediana** da coluna:
26

- **“Imputação” multidimensional:**
  - O atributo com dados faltantes é utilizado como target e os demais são preditores;
  - Processo interativo, pois mais de um atributo pode conter dados faltantes.

Normalização

Para evitar erros de escala entre variáveis, é importante que os dados estejam normalizados. Duas estratégias para normalização de variáveis numéricas estão descritas a seguir:

- **Normalização min-max** (entre 0 e 1): Os valores de um atributo são transformados de acordo com:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

  - **Exemplo:** imagine que o atributo **salário** varia de 900 a 32.000 e queremos transformar os valores para o intervalo [0.0, 1.0]. Neste caso, o valor 10.000 seria transformado para: (10.000-900)/(32.000-900) = 0,292.

- **Normalização z-score**: Os valores de um atributo são transformados de acordo com sua média e desvio padrão:

$$x_{scaled} = \frac{x - \bar{x}}{s_x},$$

Onde $\bar{x}$ é média amostral e s_x é o desvio padrão amostral.

  - **Exemplo**: imagine que o atributo **salário** possua média 3.000 e desvio padrão 1.000. Para este caso, o valor 10.000 seria transformado para: (10.000-3000)/1.000 = 7.

Variáveis categóricas

Como o próprio nome diz, os valores possíveis de variáveis categóricas são categorias. Os valores são apenas rótulos indicando diferentes categorias em que os casos podem ser classificados. Com estas variáveis categóricas, não faz sentido fazer operações aritméticas com seus valores. Assim, em princípio, nós não somamos, subtraímos ou tiramos médias de colunas na tabela que sejam variáveis categóricas:

- **Variáveis categóricas ordinais**: no caso de variáveis categóricas ordinais, o valor é um rótulo para uma categoria dentre k possíveis e as categorias podem ser ordenadas. Existe uma ordem natural nos valores possíveis. Exemplos:
  - Avaliação de filmes (1 a 5);
  - Pesquisas do tipo: em uma escala de (1-10), como você avalia …
  - Avaliações do tipo: avalie em Péssimo - Ruim - Moderado - Bom - Ótimo.
- **Variáveis categóricas nominais**: no caso de variáveis categóricas nominais, os seus valores possíveis são rótulos de categorias que não podem ser ordenadas. Exemplo:
  - Cachorro - Gato - Coelho;
  - Fumante - Não fumante;
  - Comprou - Não comprou.

Tratamento de dados faltantes

Na presença de dados faltantes e na impossibilidade de coleta de mais observações, algumas estratégias podem ser adotadas para substituição (*imputation,* em inglês) de dados categóricos:

- **"Imputação" unidimensional:**
  - Substituição pela moda (valor mais frequente);
  - Substituição por um valor constante.

Exemplo:

Figura 12 – Exemplo de tratamento de dados faltantes em atributos categóricos.

Nome	Idade	Classe Social
Matheus	26	C
Letícia	22	A
Terezinha	60	-
José	70	C

Substituição pela **moda** da coluna: C

Substituição por um valor constante: X

- **"Imputação" multidimensional:**
  - O atributo com dados faltantes é utilizado como target e os demais são preditores (problema de classificação);
  - Processo interativo, pois mais de um atributo pode conter dados faltantes.

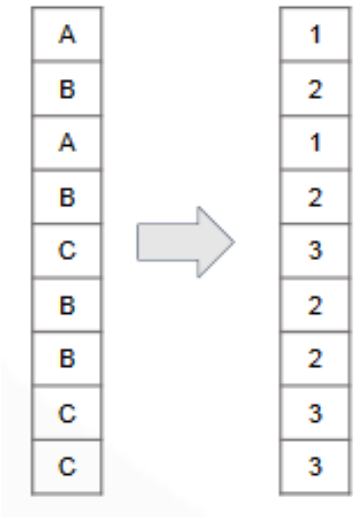
Codificação

Na etapa de pré-processamento, as variáveis categóricas são transformadas em variáveis numéricas para que possam ser inseridas em um modelo preditivo.

A seguir, algumas das técnicas mais utilizadas para este fim serão apresentadas:

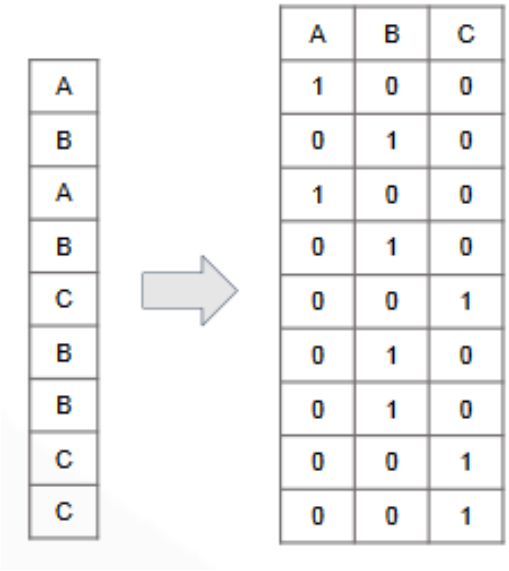
- **Label Encoding**: esta é a codificação mais simples. Os valores categóricos são substituídos por um valor numérico correspondente. Este tipo de codificação é especialmente útil para codificação da variável de resposta. Exemplo:

Figura 13 – Label Encoding.



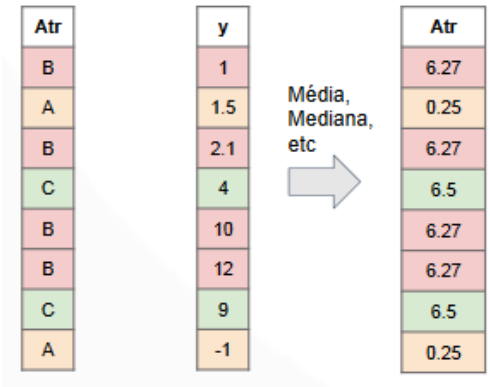
- **One-Hot-Encoding**: codificação que transforma uma variável categórica com **K** categorias distintas em **K** atributos binários. Essas novas variáveis criadas recebem o nome de *dummy variables*. Este tipo de codificação é especialmente útil quando não há uma relação de ordem entre os valores da variável categórica. Para um atributo categórico de alta cardinalidade (**K** elevado), esta pode piorar o desempenho e dificultar o treinamento de um modelo preditivo. Exemplo:

Figura 14 – One-Hot-Encoding.



- **Target Encoding**: substituição dos valores de um atributo categórico com valores extraídos da variável resposta por meio de uma estratégia de agregação (exemplo: média dos valores de y que estão atrelados à categoria A, por exemplo). Esta estratégia é interessante, pois não acrescenta novos atributos ao dataset original (menos parâmetros para treinamento), mas ela pode favorecer o sobreajuste (overfitting) pela utilização de valores da variável de resposta na matriz de atributos. Exemplo de substituição pela média:

Figura 15 – Target Encoding.



Capítulo 3. Regressão

Introdução

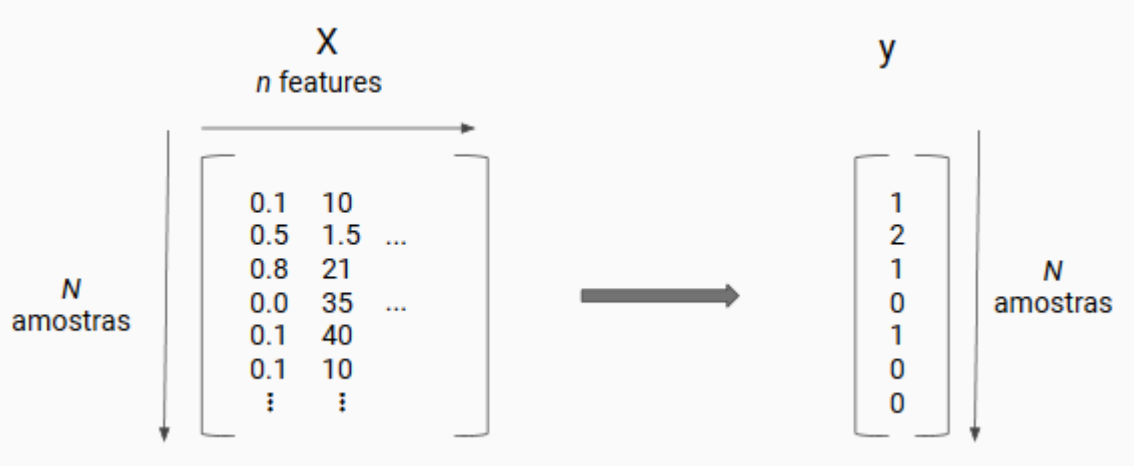
A análise de regressão é uma técnica estatística de modelagem que consiste no relacionamento entre variáveis dependentes/endógenas (**y**) e independentes/exógenas (**X**). Ela é muito usada para modelagem preditiva e de séries temporais.

Nessa análise, a variável dependente (explicada) é estimada como uma função das variáveis independentes (explicativas). A função de estimação é denominada **função de regressão** $f()$.

Existem muitas técnicas para regressão, entretanto, todas elas têm em comum (ver Figura 16):

- A variável independente (**matriz de atributos**): X;
- A **variável de resposta contínua**: y;
- Parâmetros desconhecidos: β.

Figura 16 – Representação de um problema de regressão.



A função $f()$ é especificada de acordo com o problema disponível. Desse modo, de acordo com os dados do problema, são usadas técnicas de regressão, como:

- **Regressão linear**: como próprio nome sugere, essa técnica mapeia linearmente a relação entre X e y;
- **Regressão não linear ou polinomial**: para os casos em que a variável dependente tem uma relação polinomial com a independente.

São diversas as aplicações da análise de regressão, tais como:

- Determinação da temperatura final de um processo de laminação;
- Previsão da temperatura dentro de um forno elétrico;
- Estimativa do preço de imóveis;
- Previsão de demanda.

Regressão linear

Na regressão linear, dada uma matriz de atributos $X = [x_1, x_2, \dots, x_n]$, onde *n* representa a quantidade de atributos do modelo. A estimativa do exemplo y_i é modelada como:

$$\widehat{y_i} = \widehat{\beta_0} + \sum_{j=1}^{n} x_j \widehat{\beta_j}.$$

O termo $\widehat{\beta_0}$ é o coeficiente do termo constante do modelo. Muitas vezes é conveniente incluir a constante 1 no vetor de entradas, incluir $\widehat{\beta_0}$ no vetor de coeficientes e escrever o modelo de regressão linear como um produto interno:

$$\hat{y} = X\hat{\beta}.$$

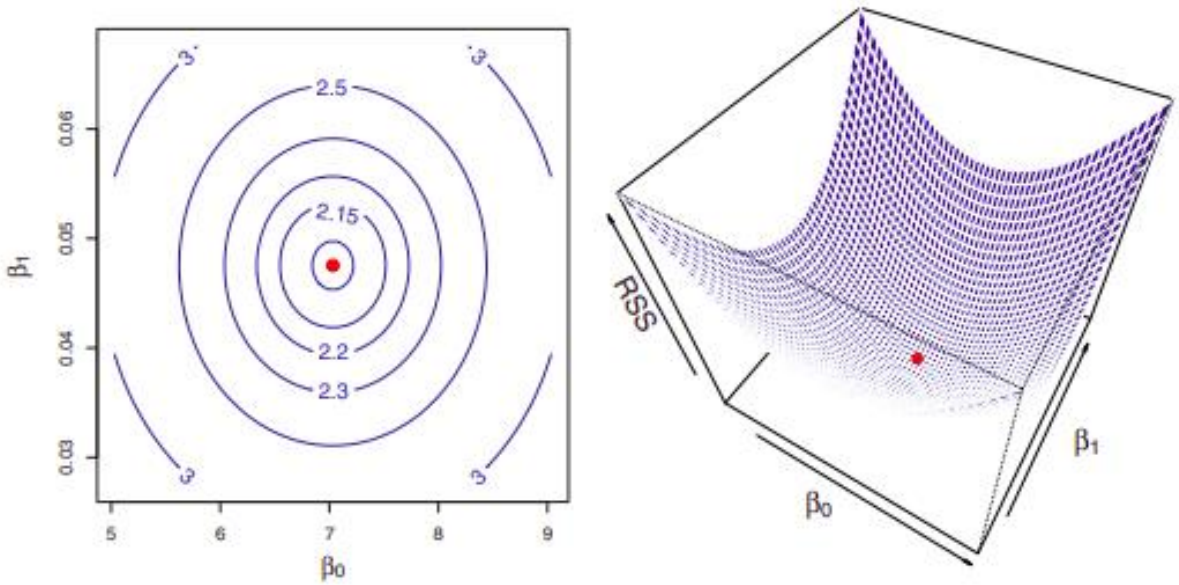
Um dos métodos mais usados para a determinação dos parâmetros β do problema é o Mínimos Quadrados. Basicamente, o problema é determinar os parâmetros β que minimizem o quadrado da soma dos resíduos (RSS, *Residual Sum of Squares*), isto é:

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{N} (y_i - X_i^T \hat{\beta}_i)^2.$$

$RSS(\beta)$ é uma função quadrática dos parâmetros (ver Figura 17) e, desse modo, sempre possui um mínimo, que nem sempre é único. Na forma matricial, é dado por:

$$RSS(\beta) = (y - X\beta)^T (y - X\beta).$$

Figura 17 – Exemplo de função de custo de regressão.

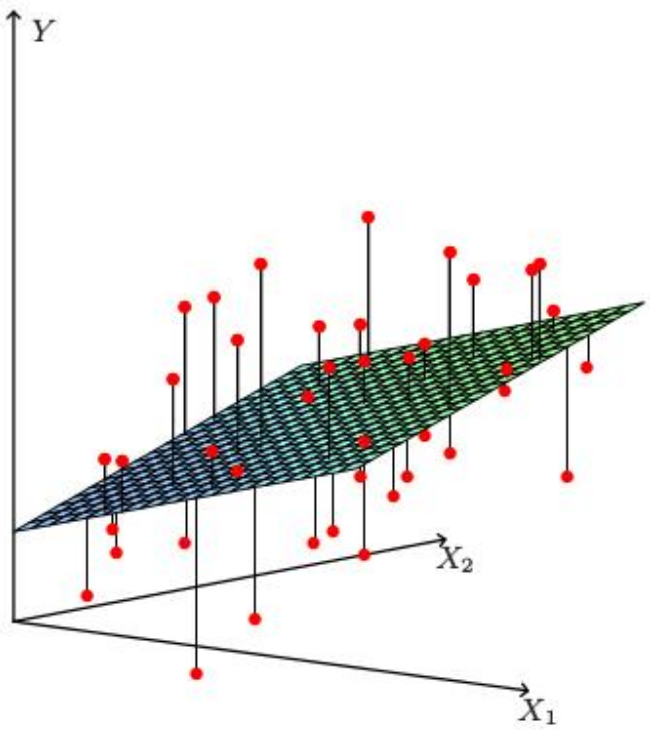


A solução fechada deste problema é dada por:

$$\hat{\beta} = (X^T X)^{-1} X^T y.$$

A imagem a seguir apresenta uma imagem do hiperplano encontrado em um problema de duas dimensões.

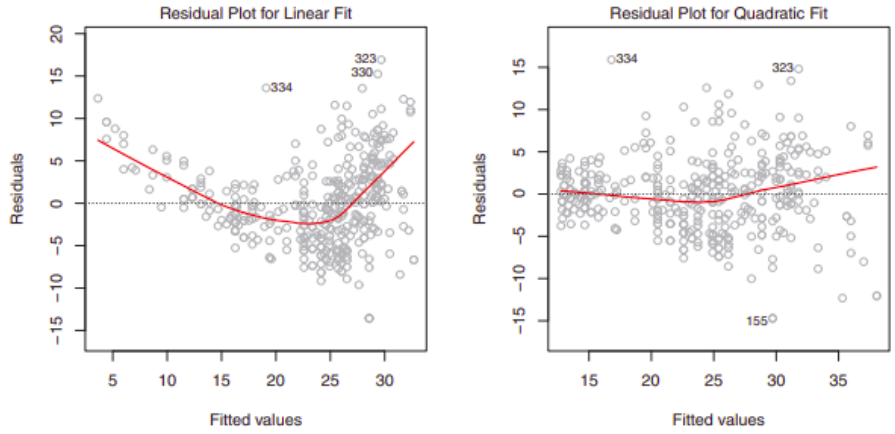
Figura 18 – Exemplo de função de custo de regressão.



No âmbito da análise regressão, algumas suposições são feitas e é importante garantir que a modelagem implementada respeite essas premissas:

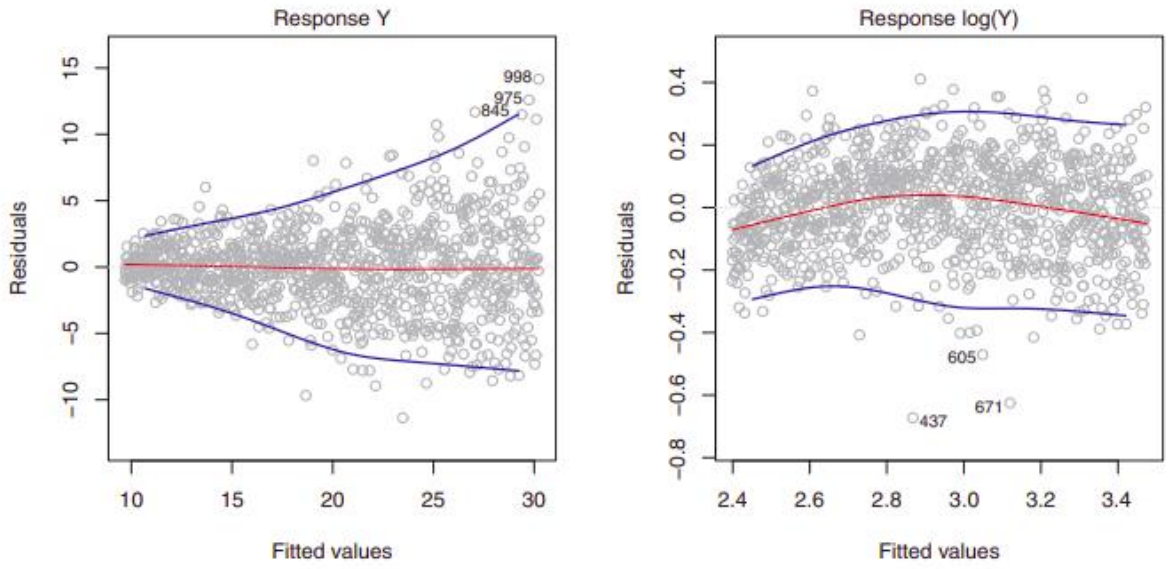
- Relação linear entre as variáveis independentes X e a variável dependente y (ver Figura 19 para exemplo):

Figura 19 – Possível solução para quando isso não ocorre: transformação não linear em X: [X, X^2].



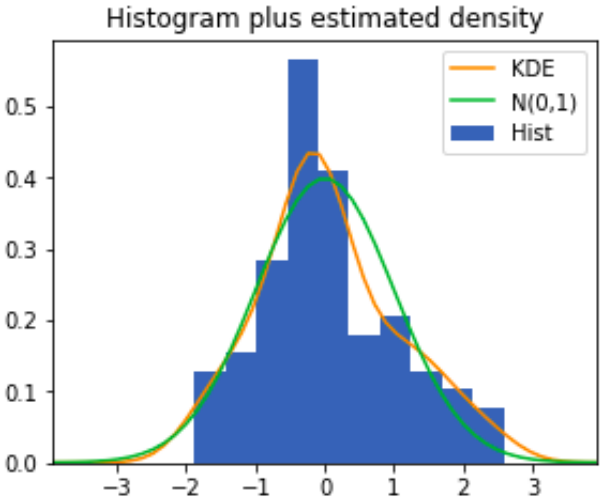
- Variância constante dos resíduos (**homocedasticidade**):

Figura 20 – Possível solução: transformação não linear da variável resposta: log(y).



- Distribuição normal dos resíduos:

Figura 21 – Distribuição normal dos resíduos.



Métricas

O primeiro passo para avaliar a qualidade de um modelo é definir um critério quantificável, uma **métrica** ou **nota**. Na regressão, umas das métricas mais comuns é o **MSE (Mean Squared Error)**, dado por:

$$MSE_{reg} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2.$$

O MSE é representado pela média do quadrado da diferença entre os valores previstos e os valores reais do modelo. Como sua formulação penaliza o quadrado do erro, esta é uma métrica que deve ser utilizada quando erros elevados não são desejáveis. O MSE é muito utilizado, pois é uma função diferenciável e, portanto, mais simples de ser otimizada.

Outra métrica muito utilizada para regressão é o **RMSE**, dado pela raiz quadrada do MSE ($RMSE = \sqrt{MSE}$). O RMSE possui o mesmo comportamento do MSE em termos de penalização de erros elevados, além de dar uma noção da ordem de grandeza da incerteza do modelo, uma vez que é uma métrica que está na mesma unidade de medida da variável de resposta.

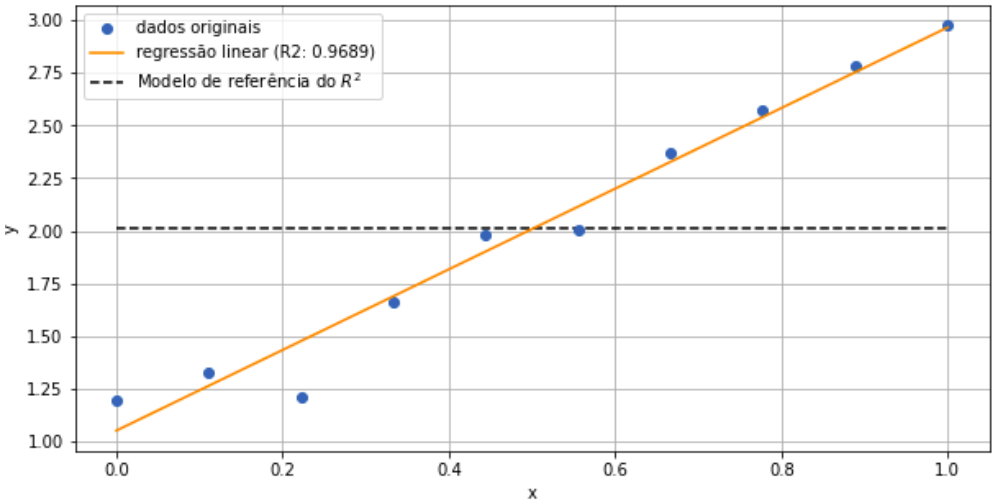
O **coeficiente de determinação R^2** é uma métrica utilizada para avaliação de modelos lineares e que representa o quão melhor o ajuste em um modelo de regressão é melhor em relação a um modelo de referência que prediz sempre a média dos pontos (ver Figura 22). Ou seja, seja o MSE do modelo de referência dado por:

$$MSE_{ref} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2,$$

Onde $\bar{y}$ representa a média dos valores de y. Desse modo, o coeficiente de determinação é dado por:

$$R^2 = 1 - \frac{MSE_{reg}}{MSE_{ref}}.$$

Figura 22 – Intuição sobre o coeficiente de determinação.



Outra métrica muito utilizada é o **MAE (Mean Absolute Error)**, dado por:

$$MAE_{reg} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|.$$

O MAE é uma métrica mais robusta (menos sensível) à presença de valores extremos do que o MSE, uma vez que os erros não estão elevados ao quadrado.

Erros em modelos preditivos

Ao avaliar a qualidade de um modelo (regressão, classificação etc.), ser capaz de medir com precisão seu erro de predição é de importância fundamental. Muitas vezes, no entanto, são utilizadas técnicas de erro de medição que dão resultados enganadores. Isso pode levar ao fenômeno de sobreajuste (overfitting, em inglês), onde um modelo modela os dados de treinamento muito bem, mas para novos dados o resultado não é satisfatório.

O real valor do erro de predição é dependente do erro de treinamento e da complexidade do modelo proposto, isto é:

$$Erro\ de\ Predição = Erro\ de\ Treinamento + Otimismo\ de\ Treino$$

Aqui, $Otimismo\ de\ Treino$ é uma medida do quão pior é o desempenho do modelo em um novo dataset quando comparado com o dataset de treinamento. Quanto mais otimista, menor será o erro do modelo nos novos dados. No entanto,

$$Otimismo\ de\ Treino = f(Complexidade\ do\ Modelo),$$

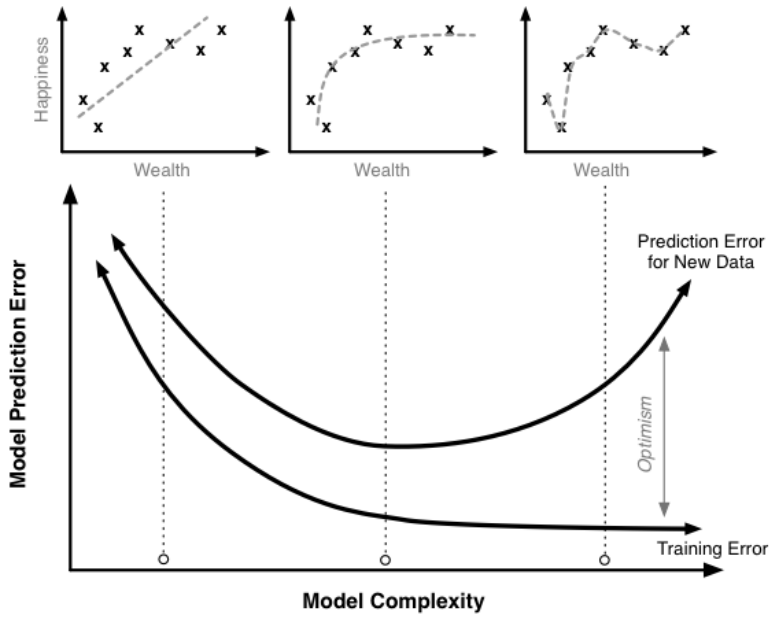
Ou seja, o **erro em novos dados é uma função da complexidade do modelo**. Mas o erro de treinamento também é função da complexidade do modelo.

Para ilustrar o exposto, poderíamos pensar no problema de determinar o nível de felicidade (Happiness) de uma pessoa com base na sua fortuna (Wealth). Esse problema poderia ser modelado de 3 formas distintas, por exemplo:

1. $Happiness = a + bWealth + \varepsilon$
2. $Happiness = a + bWealth + cWealth^2 + \varepsilon$
3. $Happiness = a + bWealth + cWealth^2 + + dWealth^3 + + eWealth^4 + fWealth^5 + \varepsilon$

A imagem a seguir ilustra a relação entre $Erro\ de\ Predição$, $Erro\ de\ Treinamento$ e a $Complexidade\ do\ Modelo$, para um modelo como esse.

Figura 23 – Relação entre erro de predição e complexidade de um modelo.



Fonte: http://scott.fortmann-roe.com/docs/MeasuringError.html.

O aumento da complexidade do modelo irá sempre diminuir o erro dos dados de treinamento, ou seja, para uma complexidade alta o suficiente, o erro de treinamento pode chegar a zero. No entanto, o aumento da complexidade do modelo faz com que o erro em novos dados aumente consideravelmente. O último modelo é um exemplo claro de overfitting e o primeiro é um exemplo de underfitting.

Assim, fica claro a relação de compromisso existente na escolha da complexidade de um modelo que deve ser levada em consideração na construção de modelos preditivos. Este compromisso é denominado "**Dilema Viés e Variância**" (ver Figura 24):

- **Erro devido ao viés**: o erro devido ao viés é a diferença do valor da predição e o valor esperado para a variável.
- **Erro devido à variância**: o erro devido à variância é a variabilidade da predição de um modelo para um determinado ponto. A variância é o quanto o valor de um ponto muda para diferentes realizações do modelo.

Figura 24 – Dilema viés vs. variância.

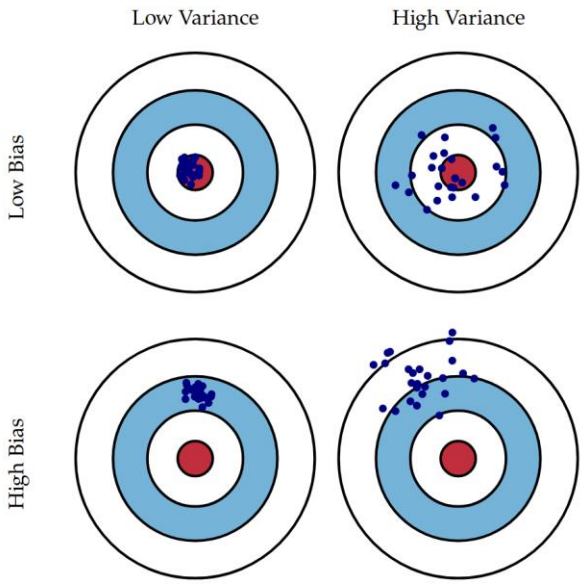
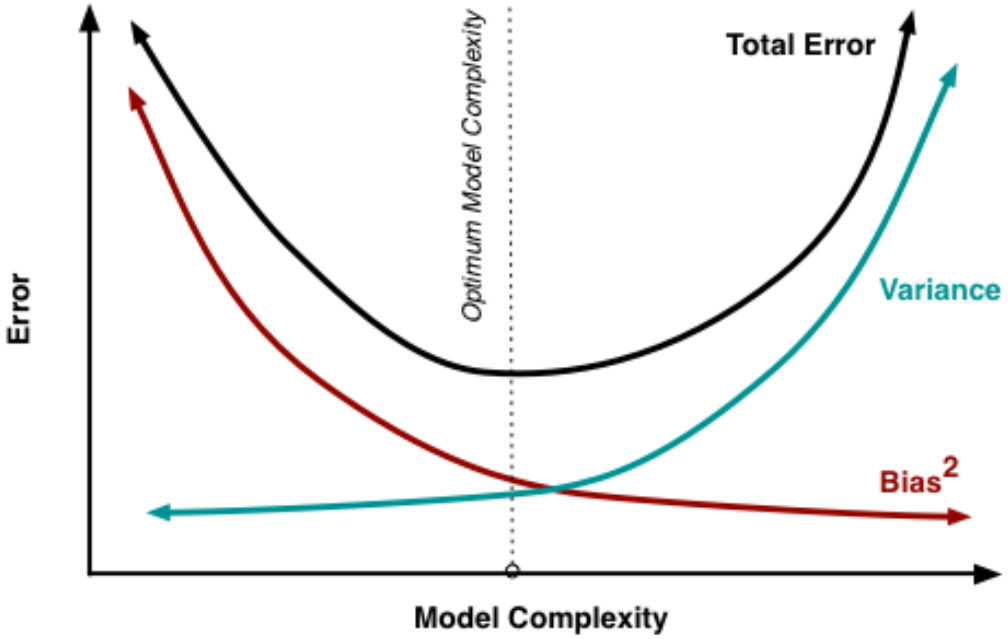


Fig. 1 Graphical illustration of bias and variance.

Fonte: http://scott.fortmann-roe.com/docs/BiasVariance.html.

Lidar com viés e variância é basicamente lidar com over- e under-fitting. O viés reduz e a variância aumenta com a complexidade do modelo, como mostra a Figura 25.

Figura 25 – Dilema viés vs. variância.



Fonte: http://scott.fortmann-roe.com/docs/BiasVariance.html.

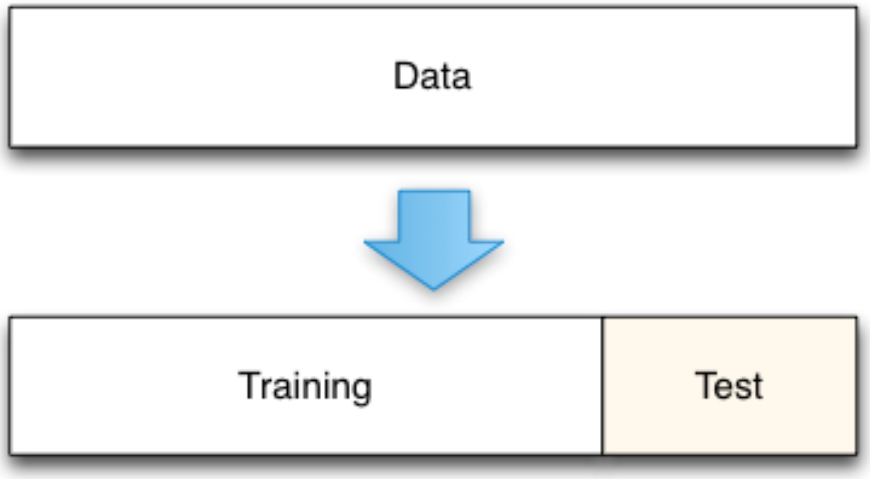
Se a complexidade do modelo excede o ponto ótimo, ocorre o overfitting do modelo, ao passo que se a complexidade é inferior ao ponto ótimo, ocorre o underfitting. Não existe uma forma analítica de determinar esse ponto, o que deve ser feito é o uso de métricas apropriadas e uso de estratégia de validação para a avaliar o desempenho do modelo.

A seguir, algumas técnicas de validação e calibração de modelos preditivos são apresentadas:

- **Validação hold-out**

Esta técnica divide um dataset em duas partes e usa uma delas para treinar e a outra para testar (ver Figura 26), em geral em uma proporção de 70%-30%.

Figura 26 – Validação hold-out.



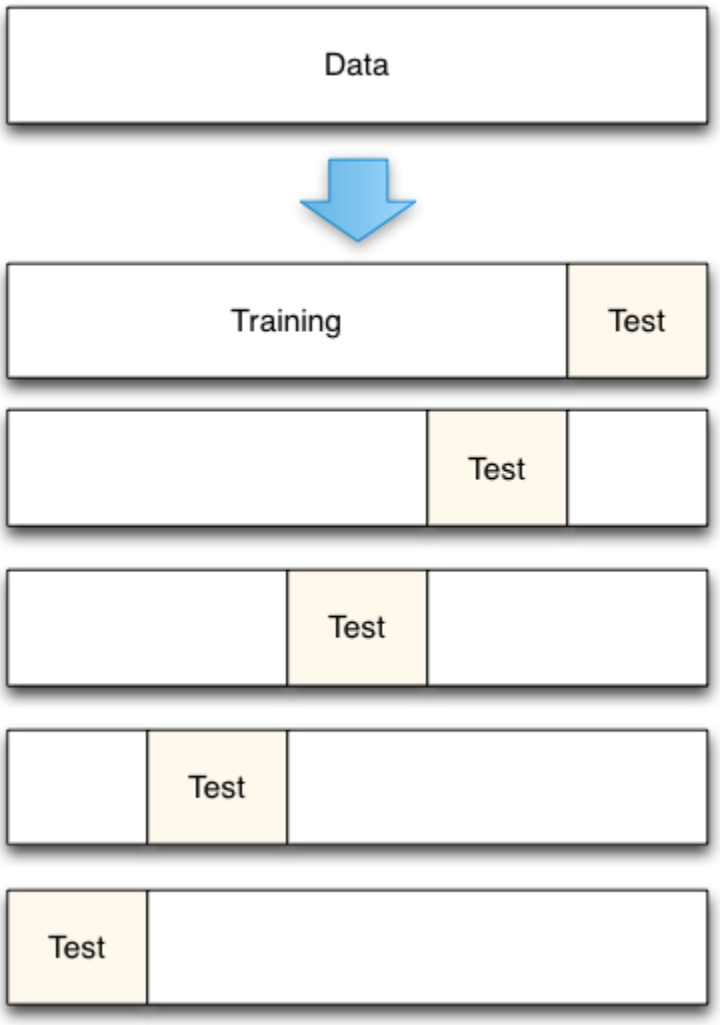
Fonte: http://scott.fortmann-roe.com/docs/MeasuringError.html.

Essa é uma técnica muito boa para mensurar o erro real. No entanto, ela não é apropriada para datasets pequenos. Além disso, como essa técnica treina o modelo com um dataset menor que o original, os erros nos dados de teste tendem a ser maiores (mais conservadores) do que no caso em que se treina o modelo com todo o dataset.

- **Validação cruzada (Cross-Validation)**

O conceito central das técnicas de validação cruzada é o particionamento do conjunto de dados em subconjuntos mutuamente exclusivos e, posteriormente, utiliza-se alguns destes subconjuntos para a estimação dos parâmetros do modelo (dados de treinamento) e o restante dos subconjuntos (dados de validação ou de teste) são empregados na validação do modelo.

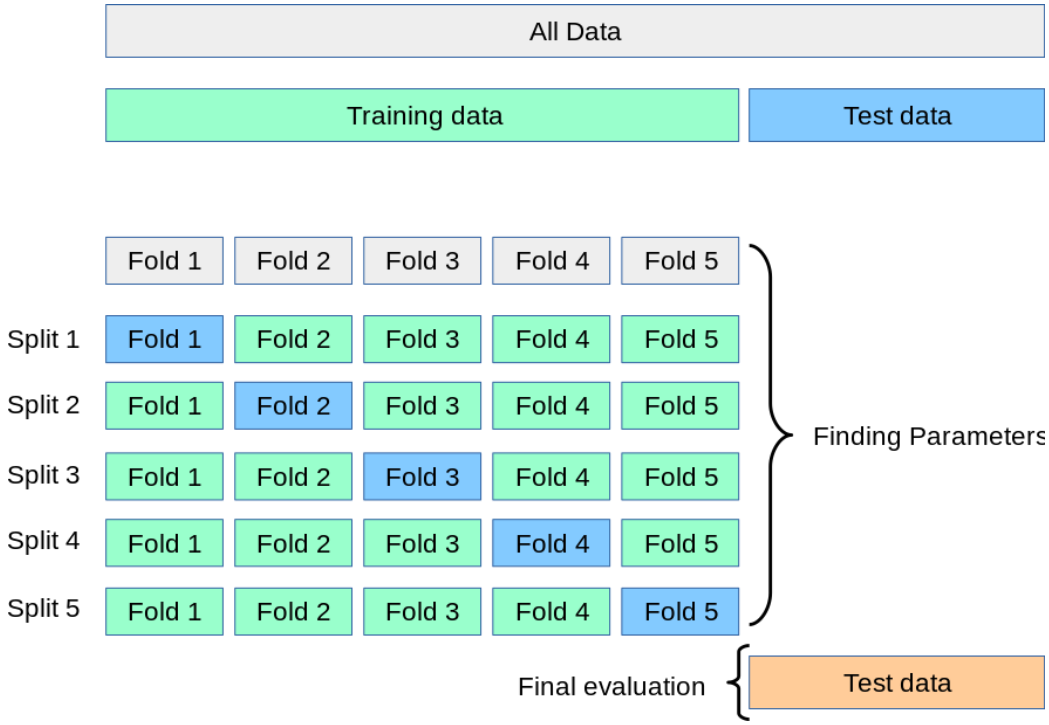
Figura 27 – Validação cruzada.



Fonte: http://scott.fortmann-roe.com/docs/MeasuringError.html.

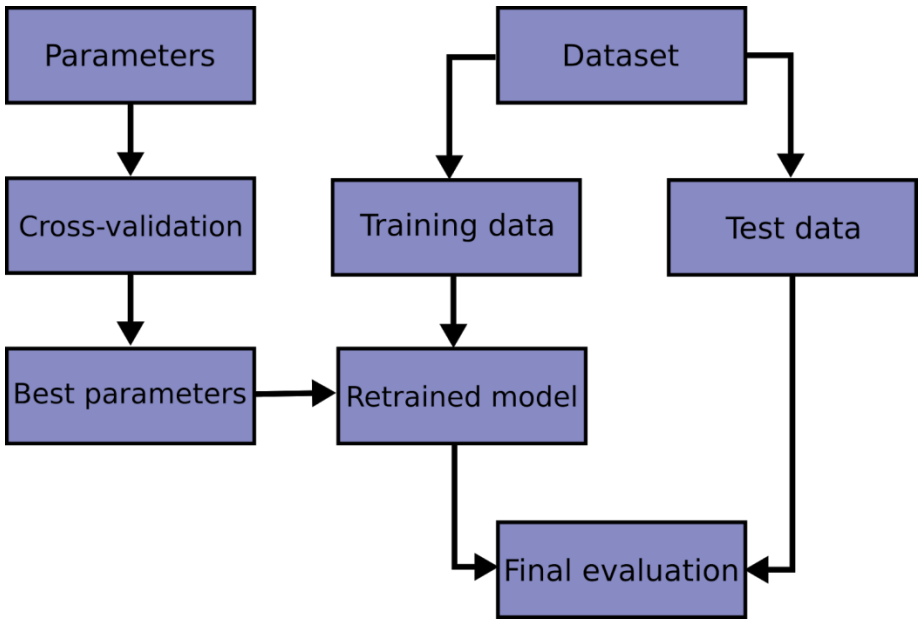
A técnica de validação cruzada é muito utilizada em conjunto com técnicas de calibração de hiperparâmetros, e uma das técnicas mais comuns é o *grid search*, que consiste em treinar diversas instâncias de um mesmo modelo, cada instância com uma combinação de hiperparâmetros distintas. A combinação de parâmetros que tiver o menor erro de validação (cruzada) é escolhida. As figuras 28 e 29 apresentam esquemáticos deste processo.

Figura 28 – Validação cruzada e grid search.



Fonte: https://scikit-learn.org/stable/modules/cross_validation.html.

Figura 29 – Fluxograma para validação cruzada e grid search.



Fonte: https://scikit-learn.org/stable/modules/cross_validation.html.

É importante ressaltar que **as técnicas de validação de modelos** apresentadas nesta seção **são válidas para um problema de classificação** também. A diferença para um problema de classificação está na escolha da métrica de avaliação.

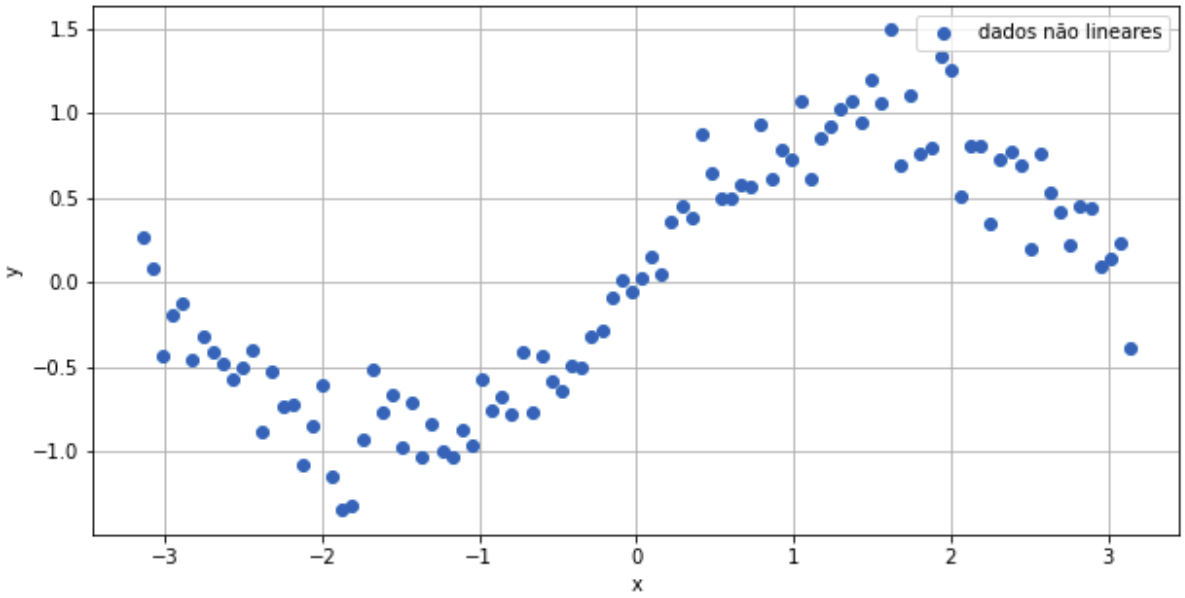
Introdução às redes neurais artificiais

Intuição

As redes neurais são modelos matemáticos inspirados na estrutura de organismos inteligentes. Elas são capazes de realizar tarefas **classificação** e **regressão** através do aprendizado a partir de dados.

Para entendermos a capacidade preditiva de uma rede neural, precisamos entender como resolveríamos o seguinte problema de regressão:

Figura 30 – Problema de regressão não linear.



A abordagem clássica para este tipo de problema é através da utilização de uma **função de base** $\phi()$ capaz de mapear o vetor de entrada para um novo espaço de características, de forma a tornar o problema “simples” de ser resolvido neste novo

espaço. Uma possível função de base é a polinomial, que aplica a seguinte transformação na matriz de atributos:

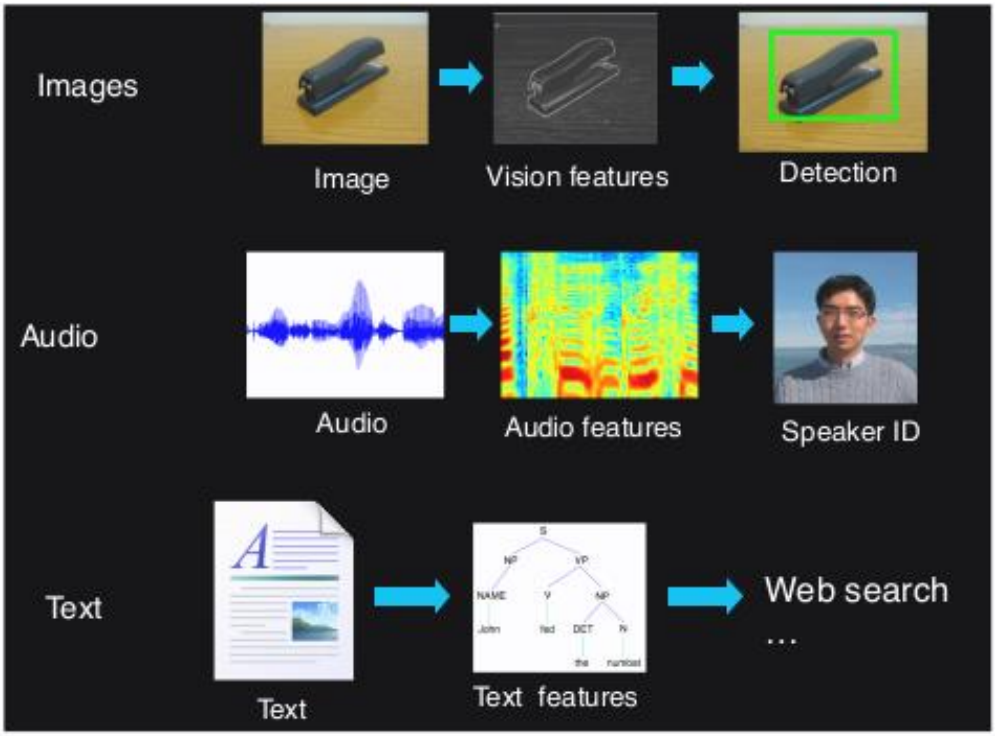
$$\hat{y} = X\beta \rightarrow \hat{y} = \phi(X)\beta,$$

Onde: $\phi(X) = [1, X, X^2, \ldots, X^d]$, sendo *d* a ordem do polinômio.

Do ponto de vista de regressão, tornar o problema "simples" significa conseguir resolvê-lo utilizando um modelo linear em função dos parâmetros. Já do ponto de vista de classificação, significa torná-lo linearmente separável.

Mas qual o problema da abordagem anterior? De maneira geral, essas soluções de mapeamento são baseadas em **conhecimento prévio do problema** ou através de uma etapa de **extração de características**, que em alguns casos pode ser inviável, como é o caso de problema mais complexos mostrados na Figura 31.

Figura 31 – Problemas complexos para extração de características manual.



Fonte: http://cs229.stanford.edu/materials/CS229-DeepLearning.pdf

Uma rede neural de múltiplas camadas é um modelo não linear capaz de **mapear o vetor de entrada** em um **vetor de características**, ou seja, as redes

neurais artificiais estruturalmente possuem a capacidade de extração de características e transformação não lineares nos dados que permite a resolução de problemas complexos.

As redes MLP (Multilayer Perceptron)

Uma rede neural (RN) é uma função matemática do tipo:

$$y = f_{RN}(X),$$

Em que a função f_{RN} tem uma forma particular: são **funções aninhadas**. Para uma rede de 3 camadas, por exemplo:

$$y = f_{RN}(X) = f_3\left(f_2\left(f_1(X)\right)\right),$$

Onde f são funções vetoriais da seguinte forma:

$$f_l(z) = W_l z + b_l,$$

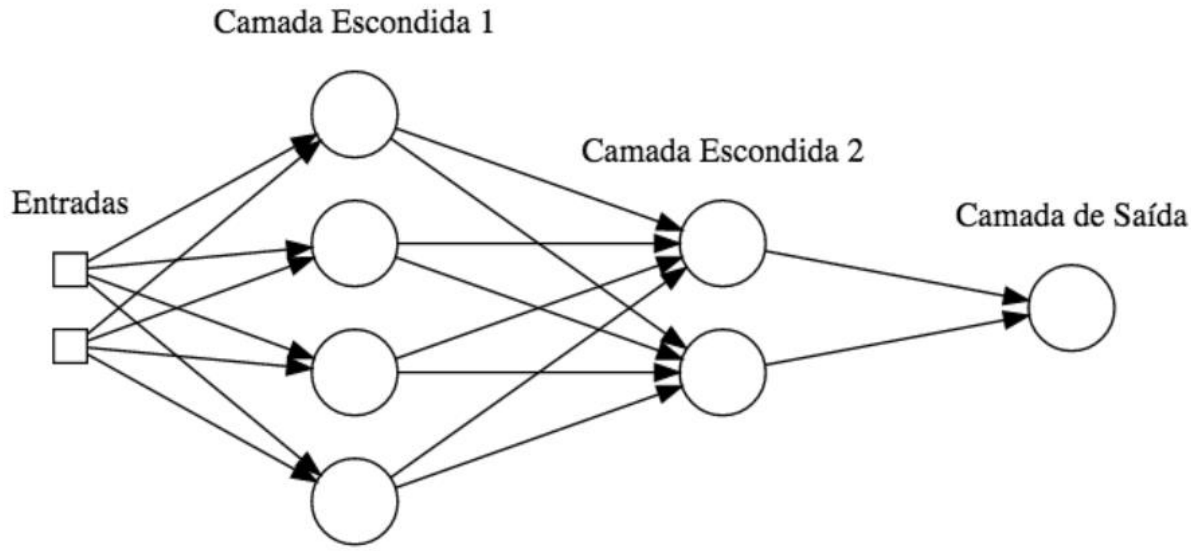
Onde l é o índice da camada e W_l e b_l são os parâmetros da camada aprendidos durante o treinamento (otimização).

Uma rede MLP (Multilayer Perceptron) possui as seguintes características:

- Formadas por neurônios do tipo Perceptron;
- Uma ou mais camadas intermediárias;
- Não existem conexões entre neurônios de uma mesma camada;
- Totalmente conectada entre camadas;
- Cada conexão possui um peso (parâmetro) associado.

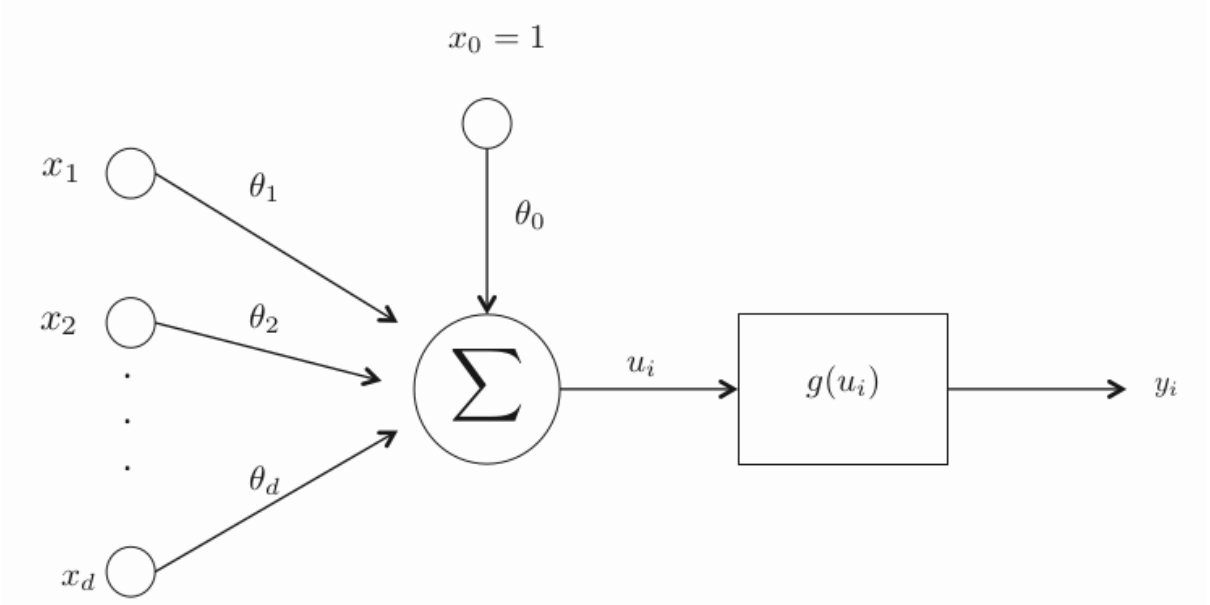
A Figura 32 ilustra uma rede MLP de 2 camadas.

Figura 32 – Exemplo de uma rede MLP de 2 camadas intermediárias (escondidas).



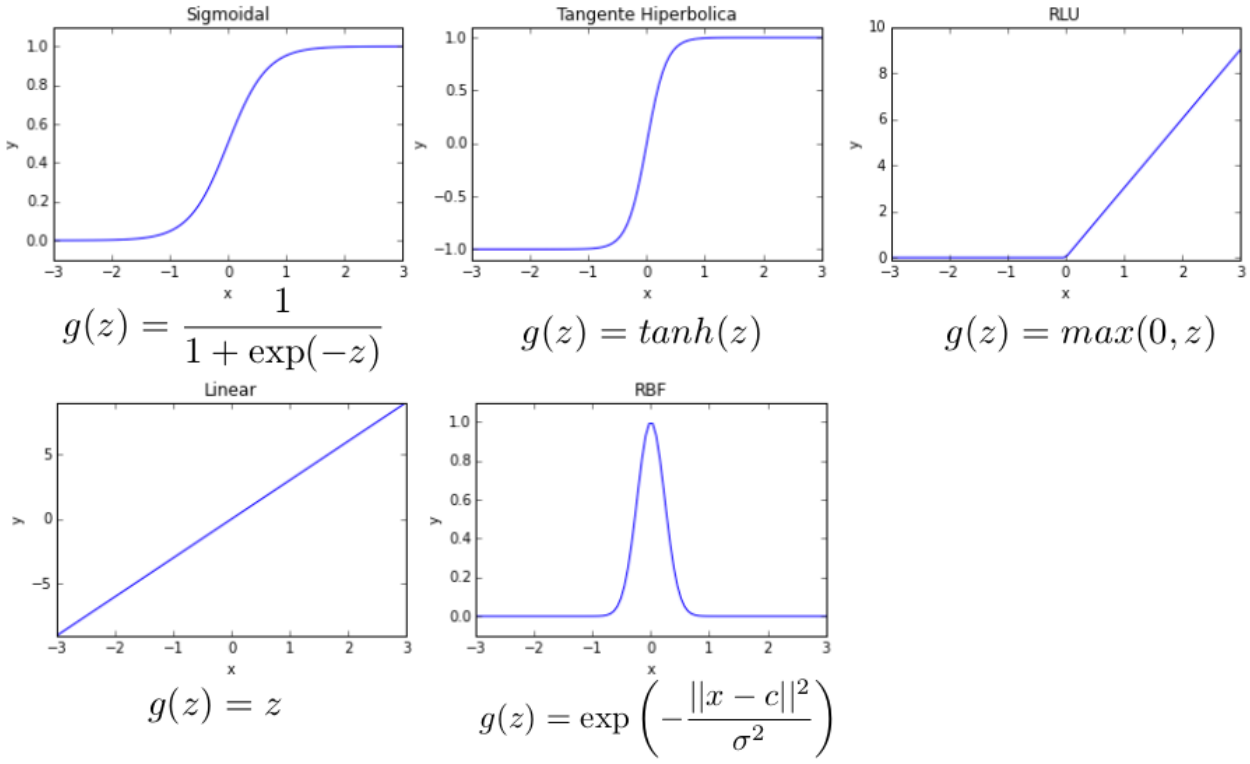
O neurônio de uma rede MLP possui a seguinte estrutura:

Figura 33 – Neurônio MLP: recebe, processa e transmite um sinal.



Os neurônios são estruturas que agregam e ponderam as entradas recebidas, aplicam uma função de ativação $g()$ e transmitem o sinal resultante. A função de ativação é responsável por definir a ativação de saída do neurônio em termos do seu nível de ativação interna. Normalmente, o sinal de ativação do neurônio pertence ao intervalo [0, 1] ou [-1, 1]. A seguir, alguns exemplos de funções de ativação.

Figura 34 – Neurônio MLP: recebe, processa e transmite um sinal.



Em resumo, uma rede neural é uma estrutura matemática que através de mapeamentos consecutivos dos padrões de entradas (camadas escondidas) e de transformações não lineares (função de ativação) é capaz de aproximar diversos tipos de funções.

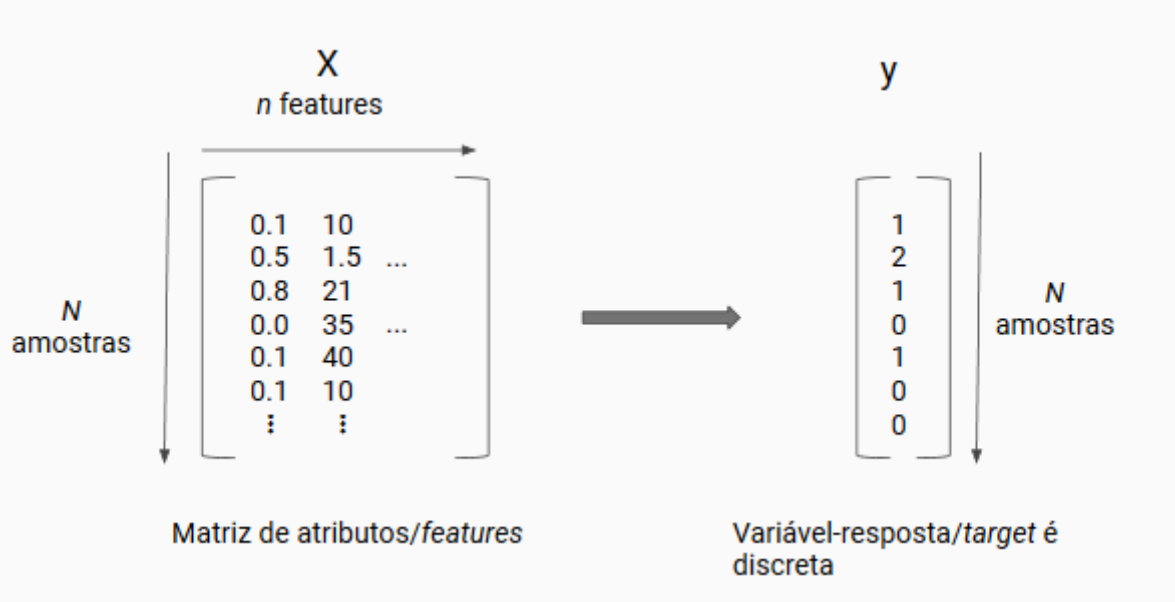
Uma rede com uma camada escondida intermediária pode aproximar qualquer função contínua. Uma rede com duas camadas intermediárias pode aproximar qualquer função (**universal approximation theorem**)!

Capítulo 4. Classificação de padrões

Introdução

No que diz respeito a aprendizado de máquinas, problemas de classificação consistem na identificação de qual grupo (ou classe) uma nova entrada pertence. São análogos aos problemas de regressão, exceto que a variável dependente é categórica (ver Figura 35).

Figura 35 – Estrutura de um problema de classificação.



São diversas as aplicações de classificação, tais como:

- Previsão de inadimplência de cartão de crédito;
- Detecção de fraude em pagamentos por cartão de crédito;
- Reconhecimento de dígitos;
- Recomendação de conteúdo;
- Etc.

Existem vários algoritmos que implementação a classificação, tais como:

– Regressão Logística;

– KNN;

– Redes Neurais;

– Decision Tree Classification;

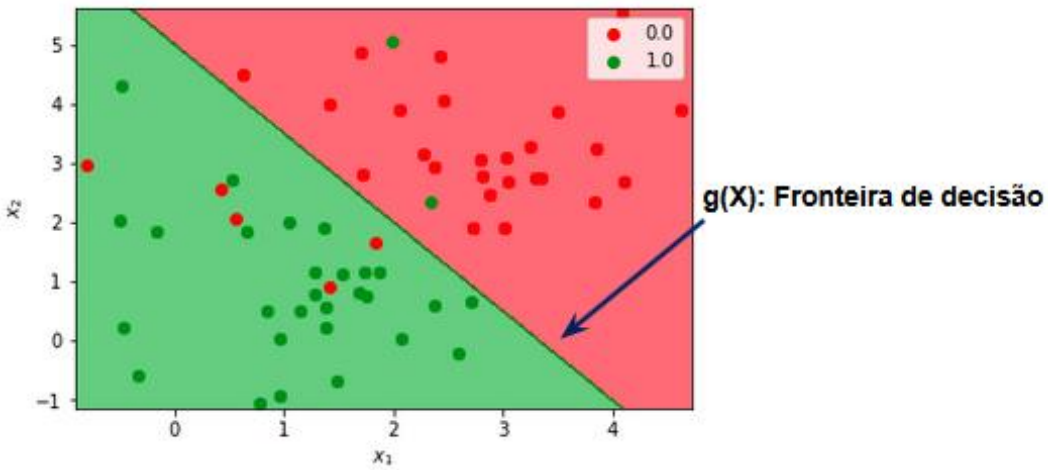
– Random Forest Classification.

Neste capítulo, iremos estudar alguns dos algoritmos de classificação mais utilizados, bem como discutir estratégias para validação e comparação de modelos.

Algoritmos de classificação

Em linhas gerais, os algoritmos de classificação buscam estimar a equação da superfície que melhor discrimina (separa) as classes da variável de resposta. Essa superfície é denominada fronteira de decisão (*boundary decision*, em inglês). A Figura 36 apresenta um exemplo de fronteira de decisão. O que diferencia um modelo do outro, é a forma como a superfície de decisão é aprendida com os dados.

Figura 36 – Exemplo de fronteira de decisão.



Regressão logística

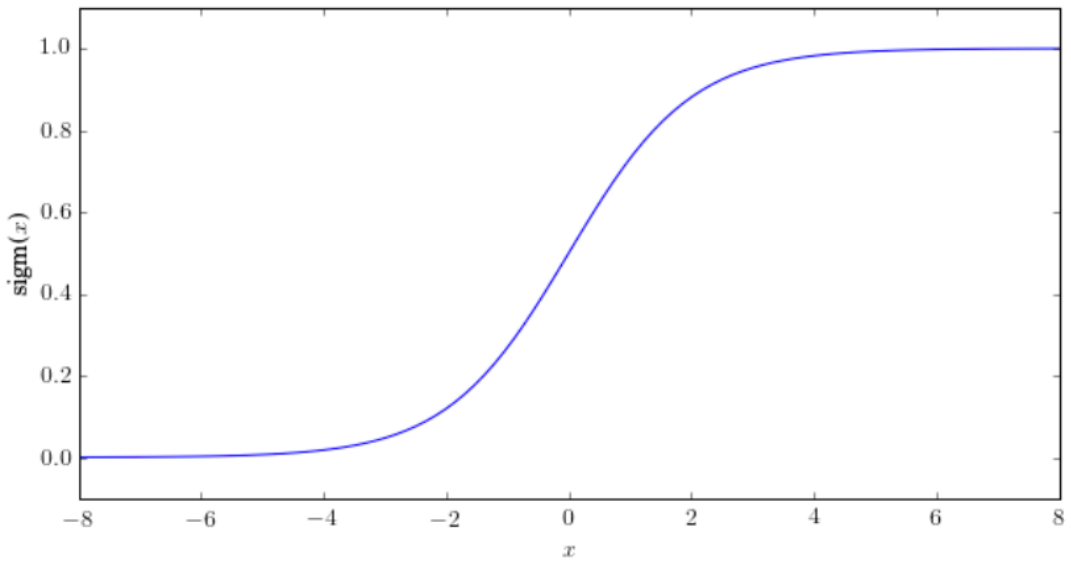
A regressão logística é um **modelo linear** no qual a variável dependente y, estimada a partir de um modelo de regressão linear, é transformada pela função sigmoide e, desse modo, dada uma entrada x, o algoritmo retorna um valor $p \in [0,1]$, que representa a probabilidade da observação pertencer a uma classe. Define-se, então, um limiar l (comumente igual a 0,5) tal que:

$$\hat{y} = \begin{cases} 1, p \geq l \\ 0, p < l \end{cases}.$$

Sabendo que a função sigmoide possui a seguinte equação:

$$g(z) = \frac{1}{1 + e^{(-z)}}$$

Figura 37 – Função sigmoide.



Assim, a formulação da regressão logística pode ser representada pela seguinte equação:

$$\hat{y} = \frac{1}{1 + e^{(-X\beta)}}.$$

Observe que o resultado da equação é contínuo, a característica de classificação é estabelecida ao se aplicar o limiar de classificação l. Note que ao se adotar $l = 0{,}5$, temos que:

$$\hat{y} = \frac{1}{1 + e^{(-X\beta)}} = 0{,}5 \implies e^{(-X\beta)} = 1 \implies X\beta = 0,$$

caracterizando um problema linear muito similar ao da regressão linear tradicional. Esta característica da regressão logística implica que ela é adequada para resolução de problemas linearmente separáveis, isto é, problemas no qual a superfície de decisão ideal é linear.

A seguir, alguns exemplos da aplicação da regressão logística em uma e duas dimensões.

Figura 38 – Regressão logística em uma dimensão.

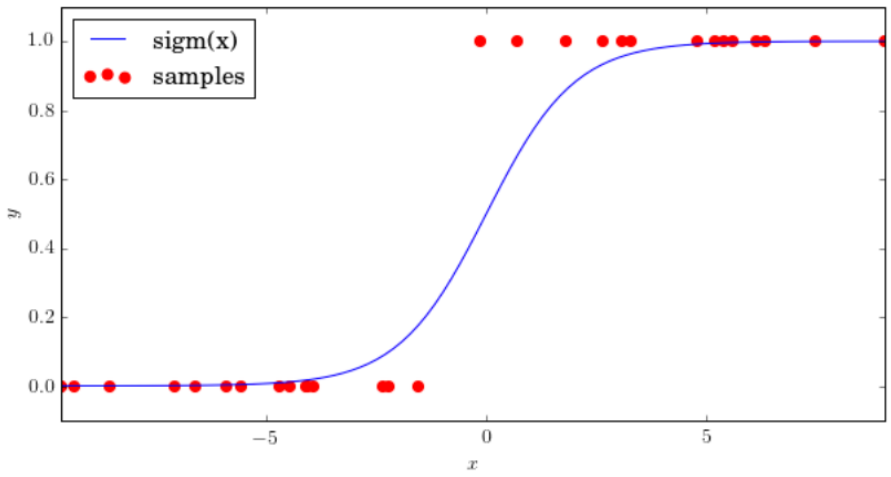
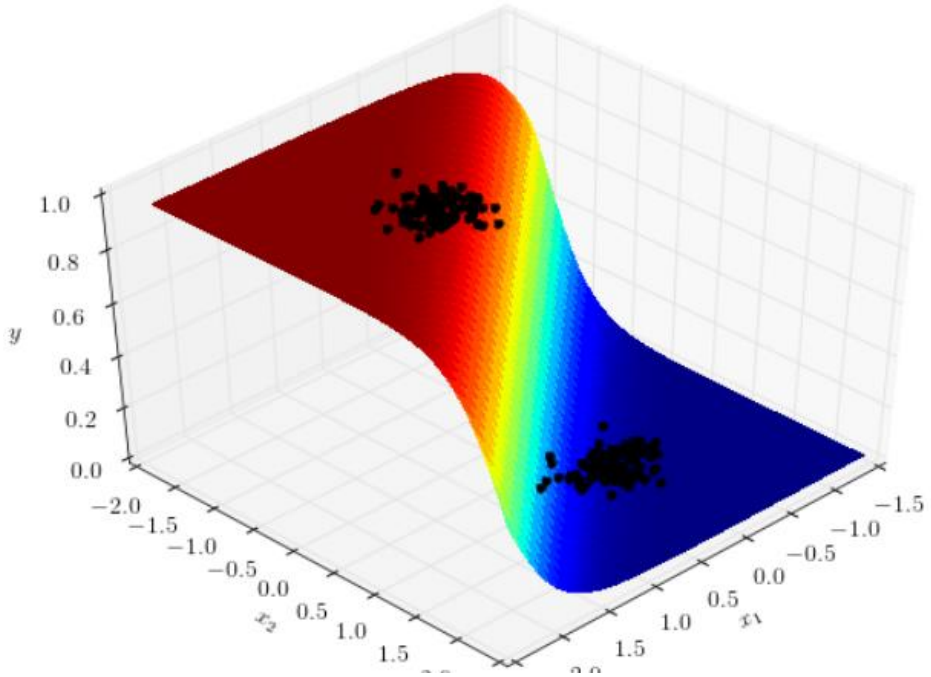


Figura 39 – Regressão logística em duas dimensões.



KNN

Um outro modelo muito usado em aprendizado de máquina é o k vizinhos mais próximos (do inglês: *k nearest neighbors – kNN*). A ideia principal do kNN é determinar o rótulo de classificação de uma amostra baseado nas amostras vizinhas advindas de um conjunto de treinamento. Matematicamente,

$$\hat{y}(X) = \frac{1}{k} \sum_{X_i \in N_k(X)} y_i,$$

Onde $N_k(X)$ é a vizinhança do ponto X definida pelos k vizinhos mais próximos X_i, que são classificados como y_i.

O algoritmo básico do KNN pode ser definido pelos seguintes 4 passos:

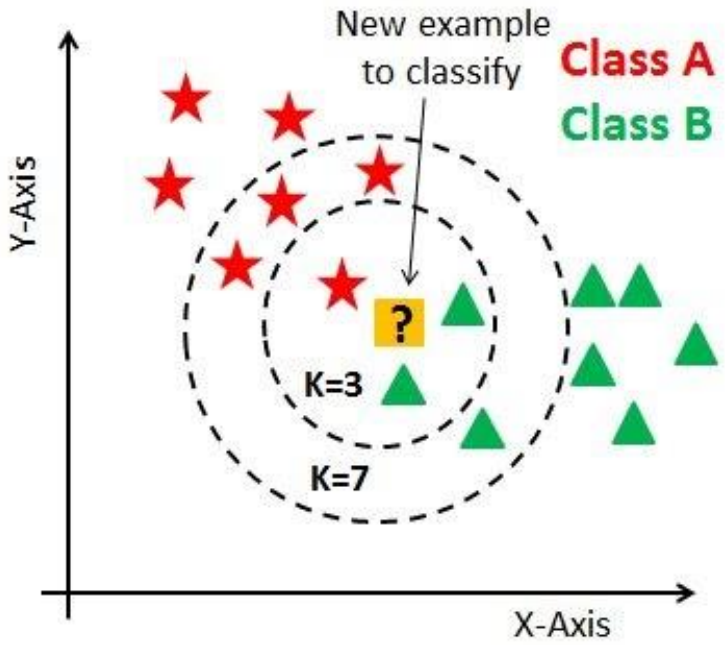
- **Passo 1**: Escolha do número k de vizinhos.
- **Passo 2**: Determinação dos k vizinhos mais próximos a partir de um critério de proximidade (e.g. distância Euclidiana).

- **Passo 3**: Dentre os k vizinhos mais próximos, contar o número de vizinhos em cada classe.

- **Passo 4**: Definir a classificação do novo ponto como sendo a classificação da classe majoritária dentro da vizinhança.

A Figura 40 traz um exemplo gráfico de aplicação do kNN para diferentes valores de k.

Figura 40 – Exemplo do KNN.



Fonte: https://www.computersciencemaster.com.br/2019/02/aula-01-classificacao-com-knn.html.

O kNN é um algoritmo bastante intuitivo, facilmente interpretável e com poucos parâmetros para escolher, e que é capaz de lidar com problemas não linearmente separáveis. Entretanto, como se trata de um método baseado em distância, o custo computacional cresce com quantidade de observações do dataset, além de sofrer com o "mal da dimensionalidade", que é um problema decorrente de uma base de dados com muitos atributos: o conceito de distância, principalmente Euclidiana, é perdido com o aumento do número de dimensões.

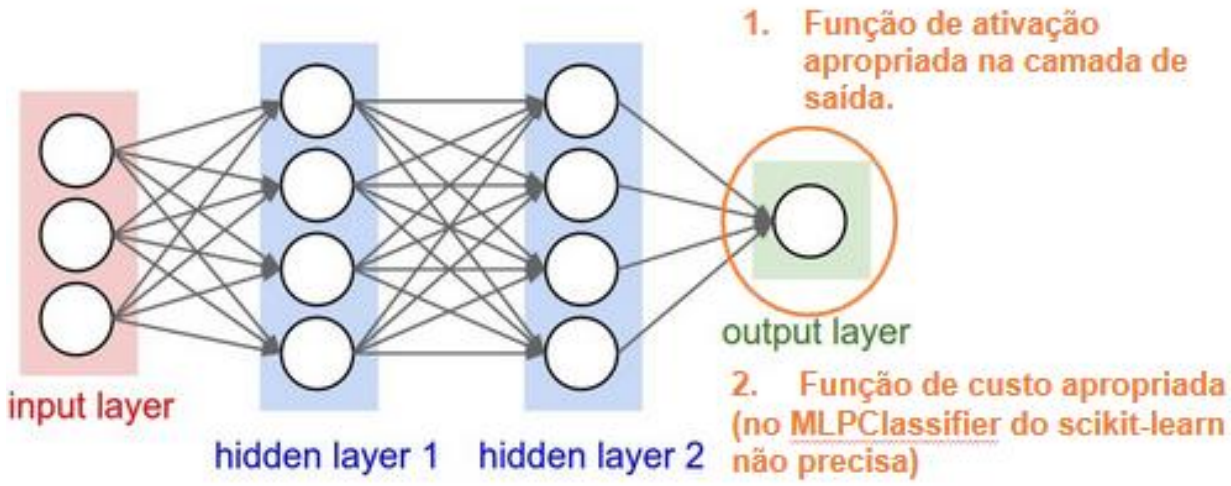
Redes neurais

Assim como na regressão, as redes neurais MLP também conseguem resolver problemas de classificação. Para que isso aconteça, a função de ativação da última camada da rede deve ser adaptada adequadamente. A seguir, alguns exemplos de funções de ativação comumente utilizadas em:

- Classificação binária: sigmoid, tanh;
- Classificação multiclasses: softmax.

Além disso, em alguns frameworks, a função de custo também deve ser adaptada (para maiores detalhes sobre funções de custo: https://keras.io/api/losses/). A Figura 41 apresenta uma ilustração das principais adaptações para uma rede neural atuar como um modelo de classificação.

Figura 41 – Rede MLP utilizada para classificação.



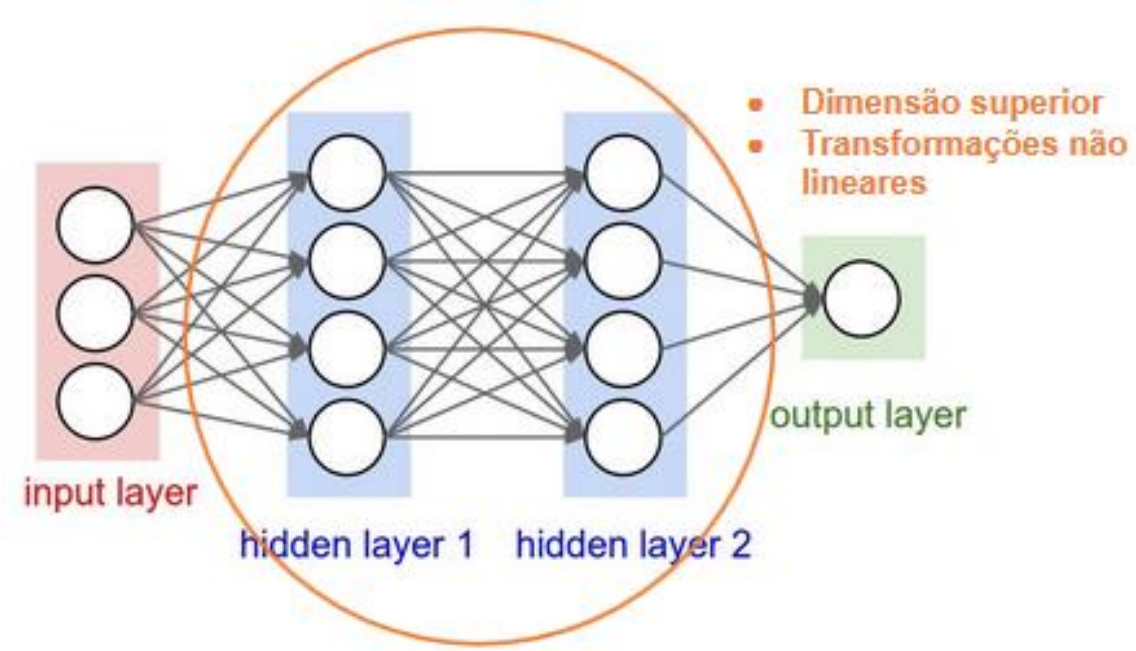
Fonte: Adaptado de http://cs231n.github.io/neural-networks-1/.

Conforme mencionado no Capítulo 3, as redes neurais são modelos capazes de mapear um espaço de entrada em um novo espaço de características de forma a tornar um problema, inicialmente complexo, “simples” de ser resolvido (ver Figura 42 para mais detalhes).

No contexto de classificação, “simples” significa que o problema é linearmente separável. O “truque do kernel”, como ficou conhecido este mapeamento, é uma

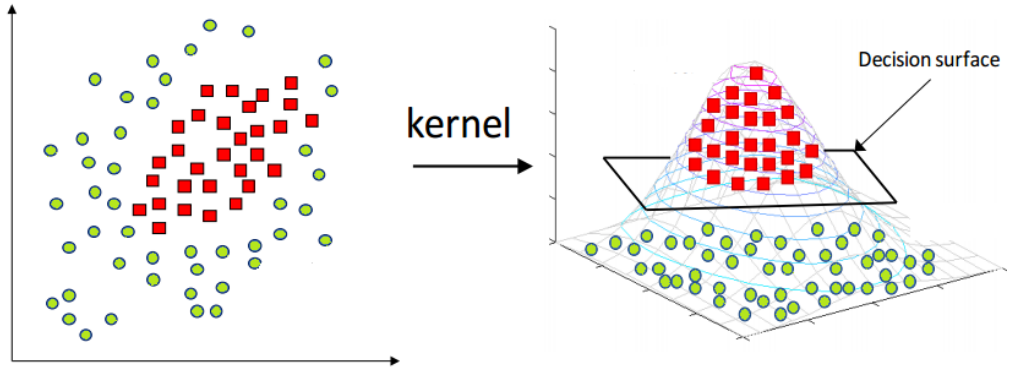
transformação dos dados com o intuito de tornar o problema “simples” de ser resolvido e é amplamente utilizado em algoritmo clássicos, tal como o SVM (Support Vector Machines). As redes neurais são estruturas capazes de fazer algo análogo. A Figura 43 apresenta um esquemático intuitivo do “truque do kernel”, que é um comportamento intrínseco à uma rede neural configurada apropriadamente.

Figura 42 – Codificações dos padrões de entrada realizada por uma rede neural.



Fonte: Adaptado de http://cs231n.github.io/neural-networks-1/.

Figura 43 – O "truque do kernel".



Fonte: https://medium.com/analytics-vidhya/how-to-classify-non-linear-data-to-linear-data-bb2df1a6b781.

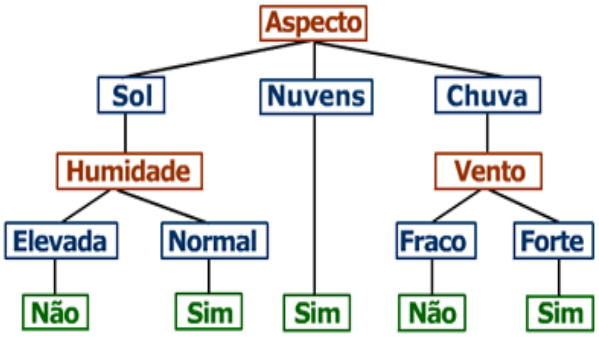
Árvores de decisão

Como o próprio nome sugere, árvores de decisão são modelos matemáticos capazes de realizar classificação e regressão, e que possuem uma estrutura em árvore (grafo), que possibilita o aprendizado de padrões a partir da construção de regras de divisão (if/else/then) consecutivas do dataset em grupos de observações cada vez menores, de forma a torná-los cada vez menores e mais homogêneos em relação à saída desejada. A Figura 44 apresenta um exemplo sintético de uma árvore de decisão para classificação.

Figura 44 – Exemplo de uma árvore de decisão.

Exemplos de Treino

Dia	Aspecto	Temp.	Humidade	Vento	Jogar Ténis
D1	Sol	Quente	Elevada	Fraco	Não
D2	Sol	Quente	Elevada	Forte	Não
D3	Nuvens	Quente	Elevada	Fraco	Sim
D4	Chuva	Ameno	Elevada	Fraco	Sim
D5	Chuva	Fresco	Normal	Fraco	Sim
D6	Chuva	Fresco	Normal	Forte	Não
D7	Nuvens	Fresco	Normal	Fraco	Sim
D8	Sol	Ameno	Elevada	Fraco	Não
D9	Sol	Fresco	Normal	Fraco	Sim
D10	Chuva	Ameno	Normal	Forte	Sim
D11	Sol	Ameno	Normal	Forte	Sim
D12	Nuvens	Ameno	Elevada	Forte	Sim
D13	Nuvens	Quente	Normal	Fraco	Sim
D14	Chuva	Ameno	Elevada	Forte	Não



O aprendizado de uma árvore de decisão pode ser resumido nos seguintes pontos:

- Qual atributo utilizar em cada nó e em qual hierarquia?
- Qual o limiar de decisão escolher para atributos numéricos?
  - Exemplo: sensação térmica > 20, 28 ou 30?

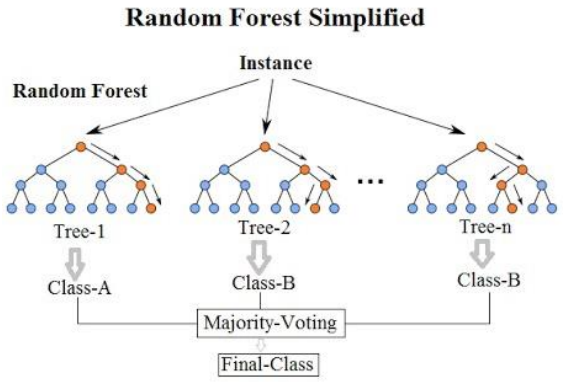
De maneira geral, toda a otimização para determinação da escolha da configuração da árvore é feita baseada em um critério de minimização de “impureza”, isto é, nos terminais (folhas) cada vez mais homogêneos (uma única classe em cada folha é a menor impureza possível).

Florestas aleatórias

Florestas aleatórias (Random Forest, em inglês) são modelos do tipo *ensemble*, isto é, são modelos construídos a partir da agregação dos resultados intermediários de diversos submodelos (weak learners).

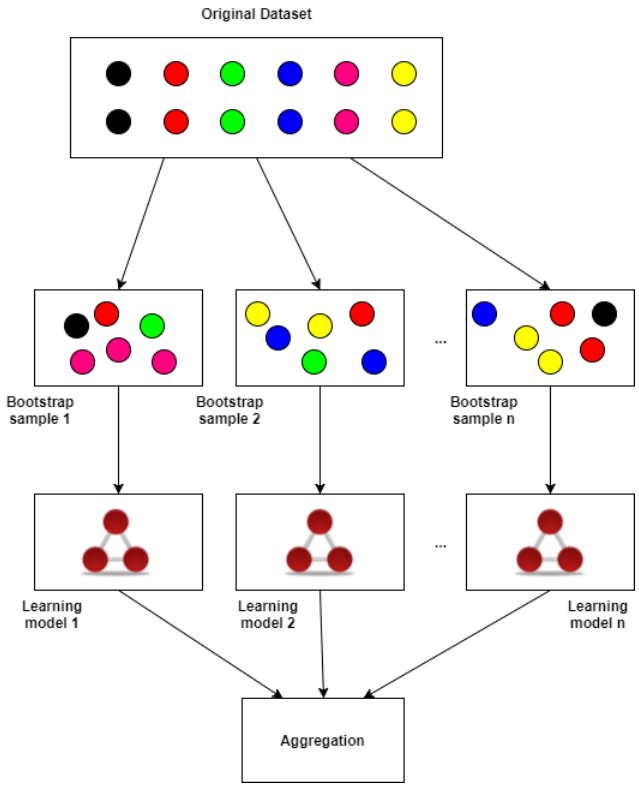
As florestas aleatórias são modelos construídos a partir de diversas árvores de decisão. O resultado final do modelo pode ser dado pelo voto majoritário (no caso de problemas de classificação) ou pela média dos resultados de cada árvore (no caso de problemas de regressão). A Figura 45 apresenta um esquemático das florestas aleatórias.

Figura 45 – Esquemático do Random Forest.



As instâncias subjacentes da floresta são construídas a partir da amostragem com reposição das observações do conjunto de treinamento original (técnica de reamostragem conhecida como bootstrap). Além disso, cada instância utiliza apenas uma parte dos atributos originais. Essas duas estratégias fazem com que este algoritmo minimize o sobreajuste dos dados de treinamento e melhore o poder preditivo final.

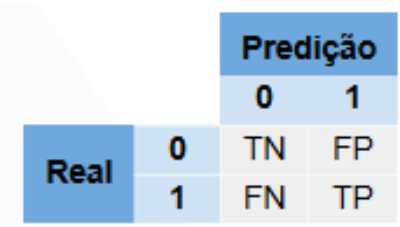
Figura 46 – Esquemático da reamostragem do Random Forest.



Avaliando a qualidade de um modelo de classificação

Em problemas de classificação, uma das principais formas de verificação de desempenho é a análise da matriz de confusão (*confusion matrix*, em inglês). Essa matriz possui o seguinte formato:

Figura 47 – Matriz de confusão em um problema binário.



Onde:

- TN: verdadeiro positivo (true negative);
- FN: falso negativo (false negative);
- FP: falso positivo (false positive);
- TP: verdadeiro positivo (true positive).

A partir desses dados, outras métricas de desempenho podem ser calculadas, tais como:

- Acurácia: razão entre as previsões corretas e todas as previsões realizadas.

$$Acurácia = \frac{TN + TP}{TN + TP + FN + FP}$$

- Taxa de verdadeiro positivo (TPR) ou Recall: proporção de positivos que foi identificada corretamente.

$$TPR = \frac{TP}{TP + FN}$$

- Precisão (precision): proporção entre as observações que de fato são positivas e todas as classificadas como tal.

$$Precisão = \frac{TP}{TP + FP}$$

- Taxa de falso positivo (FPR): proporção entre a quantidade de observações classificadas como positivas e o total de observações da classe negativa.

$$FPR = \frac{FP}{FP + TN}$$

- F1-Score: média harmônica entre Precision e Recall, dado por:

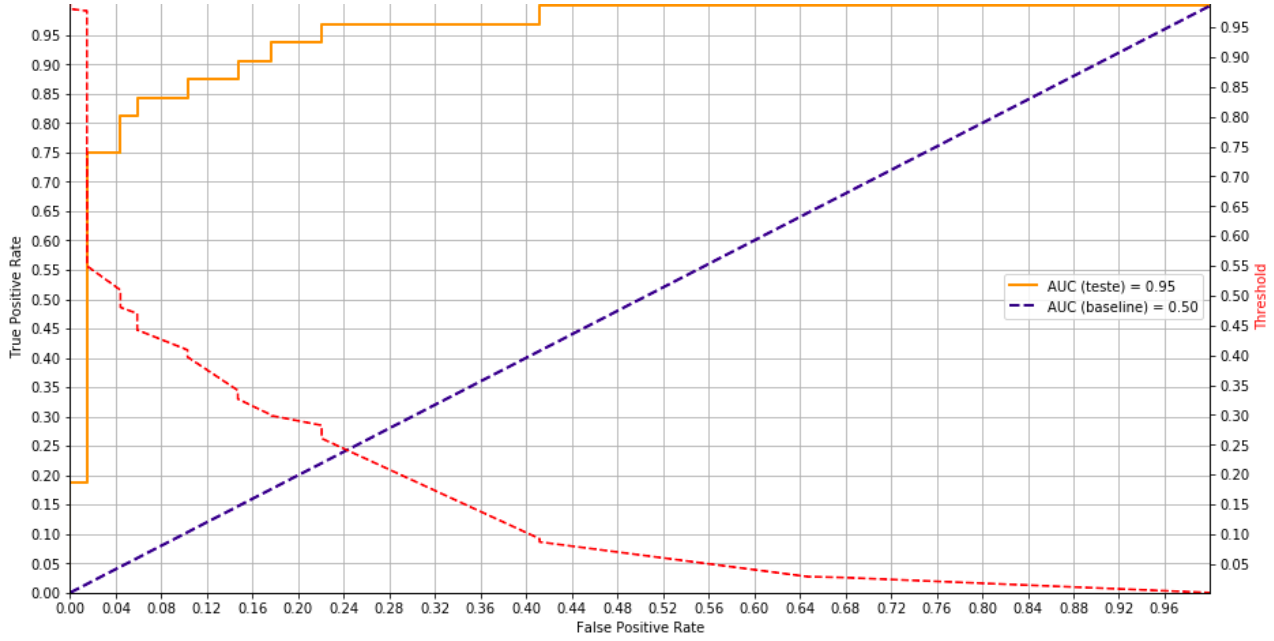
$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Os cálculos acima visam entender o seu modelo sobre os dados positivos, ou seja, para a classe ou evento de interesse. Quando estamos falando de classificação multiclasse (mais de duas classes possíveis de resposta), podemos trabalhar sobre a classe que queremos prever, mas também podemos olhar para cada classe em separado. Para tanto, pode ser útil, na hora de interpretar a matriz de confusão, considerar a classe em questão como a classe positiva, e todo o restante como negativo.

É importante lembrar que as métricas de desempenho acima mencionadas estão relacionadas com a escolha do limiar (*threshold*, em inglês) de probabilidade l, que por padrão é igual a 0,5. Uma forma mais completa de análise de desempenho do modelo é através da construção da curva ROC (Receiver Operating Characteristic Curve). Essa curva plota a taxa de verdadeiros positivos em função da taxa de falsos positivos para uma sequência de limiares diferentes. A área dessa curva (AUC – Area Under the Curve) também é um indicativo de desempenho do modelo proposto, quanto mais próximo de 1, melhor.

Assim como no caso de regressão linear, na classificação também é usado um modelo padrão para comparação de desempenho. Um modelo baseline para sistemas de classificação prediz aleatoriamente 0 ou 1. O baseline apresenta AUC igual a 0.5. A figura a seguir apresenta uma curva ROC de um modelo de classificação.

Figura 48 – Curva ROC e métrica AUC.


True Positive Rate
False Positive Rate
Threshold
AUC (teste) = 0.95
AUC (baseline) = 0.50

Capítulo 5. Séries temporais

Ao longo dos anos, diversas técnicas foram desenvolvidas para análise de séries temporais e predição de eventos e comportamentos futuros. Este capítulo irá discutir algumas dessas técnicas.

Uma série temporal é uma sequência ordenada de observações de um determinado evento. Essas observações são registradas em intervalos regulares ao longo do tempo. Uma característica muito importante desse tipo de dado é que as observações vizinhas são dependentes. Desse modo, o objetivo da análise de série temporal é modelar essa dependência.

A previsibilidade de um determinado evento ou quantidade depende de diversos fatores, tais como:

- Quão bem entendemos os fatores que contribuem para ele?
- Temos dados disponíveis?
- A previsão pode afetar o valor que estamos tentando prever?

Desse modo, na etapa inicial do planejamento da previsão algumas perguntas devem ser respondidas. Por exemplo, suponha que queiramos prever a quantidade de itens produzidos em uma indústria. É necessário saber se iremos prever:

- Cada produto da linha de produção ou grupos de produtos?
- Previsão semanal, mensal ou anual?
- Previsão discriminada por loja, lojas de uma região ou somente o total?
- Qual o horizonte da previsão: dia seguinte, semana seguinte, mês ou ano seguinte?

O planejamento da previsão e o conhecimento dos dados é **fundamental,** pois isto irá responder se é possível realizar a previsão e se algum ajuste no escopo inicial será necessário. E também irá direcionar a escolha do modelo preditivo.

Ao longo do capítulo, serão discutidos os seguintes assuntos:

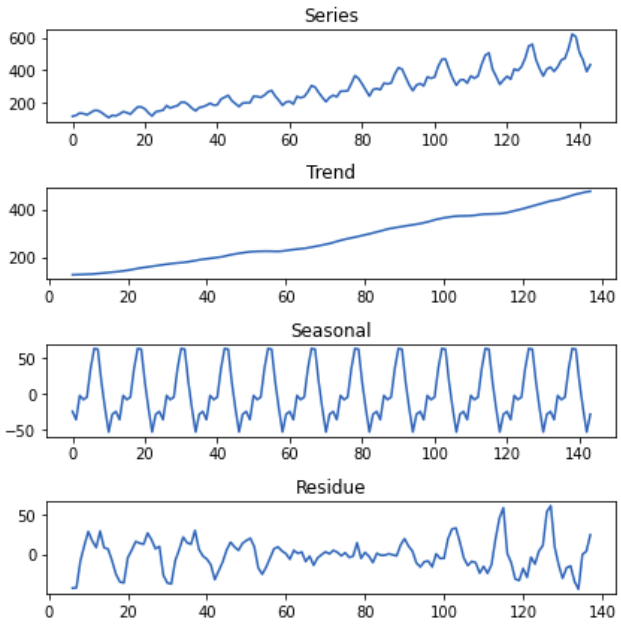
- Introdução de análise de série temporal;
- Previsão de usando abordagens tradicionais (modelos ARIMA);
- Previsão de usando abordagens mais recentes (Redes Neurais Recorrentes).

Introdução

Uma série temporal é a composição de três componentes:

- **Sazonalidade**: componente responsável pelas variações periódicas dos dados;
- **Tendência**: componente responsável por um comportamento de aumento ou diminuição ao longo do tempo;
- **Resíduo**: sinal restante após a subtração dos dois anteriores. Também denominado componente aleatório.

Figura 49 – Componentes de uma série temporal.



Neste curso, iremos focar no modelo aditivo de séries, no qual:

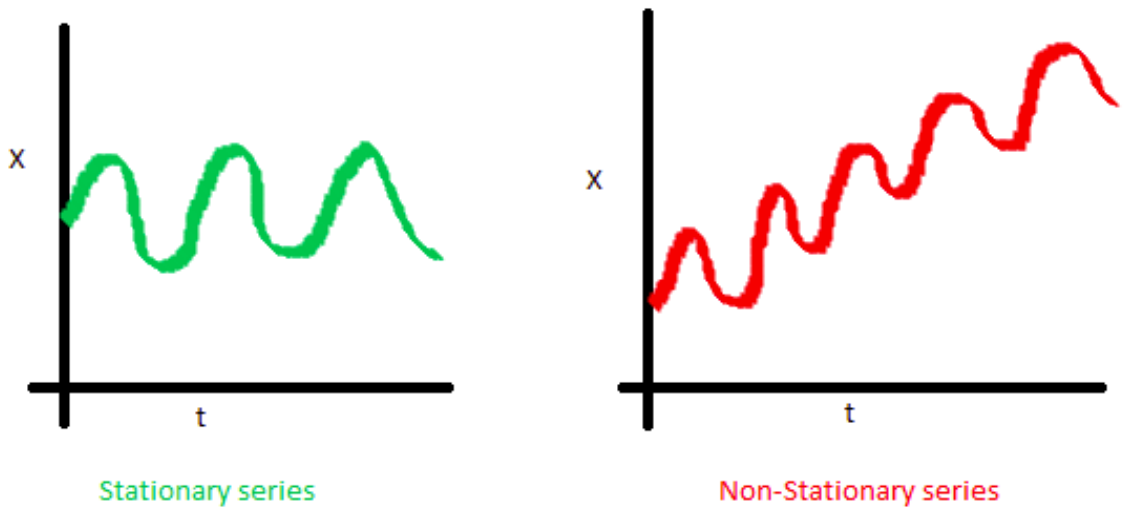
Série Temporal = Tendência + Sazonalidade + Resíduo

Modelos ARIMA

No uso de modelos ARIMA, alguns conceitos básicos devem estar claros:

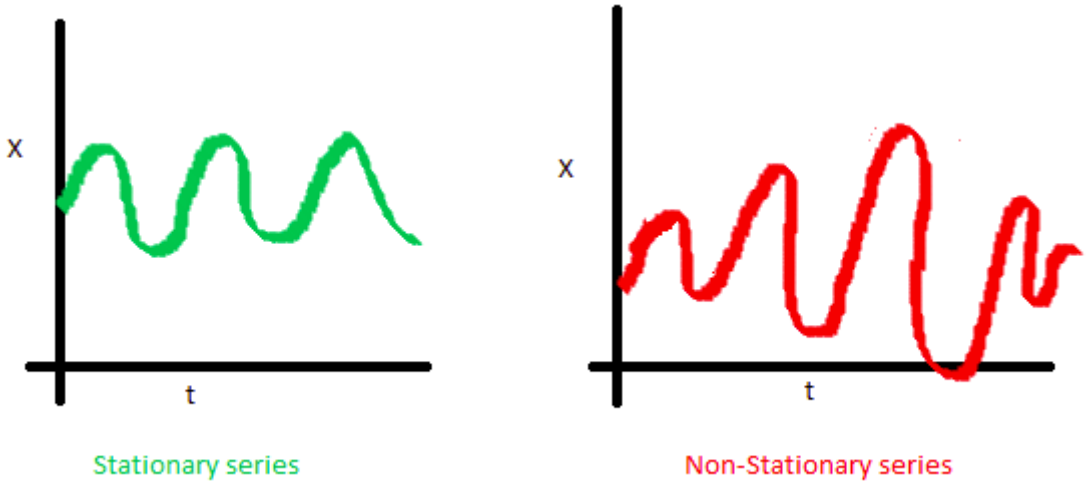
- **Estacionariedade**: Se o processo estocástico que gerou a série de observações é invariante com respeito ao tempo, diz-se que o mesmo é estacionário. Em outras palavras, a média, a variância e a autocorrelação não variam ao longo do tempo. Essa é uma premissa básica feita pelos modelos ARIMA. As figuras 50 e 51 ilustram exemplos de séries estacionárias e não estacionárias.

Figura 50 – Série estacionária (esquerda) vs. Série não estacionária (direita): comparativo da média.



Fonte: https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/.

Figura 51 – Série estacionária (esquerda) vs. Série não estacionária (direita): comparativo da variância.



Fonte: https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/.

- **Diferenciação**: a diferenciação é uma transformação aplicada aos dados de uma série temporal a fim de tornar esta série estacionária. Diversos testes podem ser aplicados para verificar se a série é estacionária ou não, tal como o teste de Dickey Fuller Aumentado (ADF – Augmented Dickey Fuller). Para maiores detalhes, checar a referência: https://nwfsc-timeseries.github.io/atsa-labs/sec-boxjenkins-aug-dickey-fuller.html. A seguir, alguns exemplos de diferenciação de uma série temporal y_t:
  - Diferenciação de primeira ordem:

$$y' = y_t - y_{t-1}.$$

  - Diferenciação de segunda ordem:

$$y'' = y'_t - y''_{t-1}.$$

  - Diferenciação sazonal de ordem *m* (período sazonal):

$$y' = y_t - y_{t-m}.$$

Modelos ARIMA são formulados da seguinte maneira:

- **Modelo AR(p)**: modelo de regressão onde uma observação no instante **t** é a combinação linear de **p** observações anteriores, isto é:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

- **Modelo MA(q)**: também é um modelo de regressão linear que modela o impacto do erro/ruído de **q** observações anteriores na observação atual, isto é:

$$y_t = c + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

A combinação dos modelos AR e MA em uma série diferenciada, resulta no modelo **ARIMA(p,d,q)**, onde **d** é a ordem da diferenciação aplicada.

Uma das maneiras mais usadas para a identificação da ordem do modelo ARMA, são os gráficos ACF (Auto Correlation Function) e PACF (Partial Auto Correlation Function).

O gráfico ACF ajuda a entender a relação entre uma observação e outra anterior. O gráfico ACF é usado para determinar a ordem do modelo de médias móveis, o valor no qual há uma queda brusca é o valor de q.

O gráfico PACF ajuda a entender a relação entre uma observação com outra observação específica, excluída a influência das demais. O gráfico PACF é usado para determinar a ordem do modelo autoregressivo, o valor no qual ele cai é o valor de p.

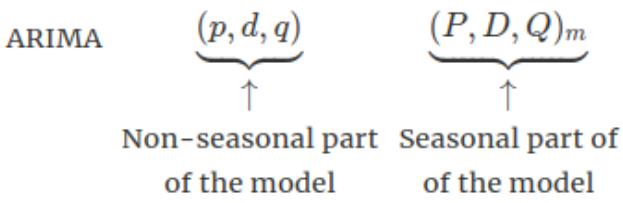
A Tabela 1 apresenta um guia para determinação da ordem do modelo ARMA utilizando os gráficos de ACF e PACF.

Tabela 1 – Tabela-resumo para análise de ACF e PACF.

	AR(p)	MA(q)	ARMA(p,q)
ACF	Decaimento gradual	Declínio acentuado após **q**	Decaimento gradual
PACF	Declínio acentuado após **p**	Decaimento gradual	Decaimento gradual

A sazonalidade de uma série temporal é facilmente incorporada em um modelo ARIMA através da consideração de termos de diferenciação e de ordens AR e MA em relação a ***m*** observações anteriores, onde ***m*** representa a quantidade de observações por ciclo de sazonalidade. Um modelo ARIMA sazonal possui o seguinte formato:

Figura 52 – Modelo SARIMA.



A determinação de ***m*** deve ser feita *a priori* depende de um conhecimento do processo que original a série temporal. A Figura 53 apresenta um guia para determinação deste parâmetro.

Figura 53 – Determinação de *m*: número de observações por ciclo de sazonalidade.

	Frequencies				
Data	**Minute**	**Hour**	**Day**	**Week**	**Year**
Daily				7	365.25
Hourly			24	168	8766
Half-hourly			48	336	17532
Minutes		60	1440	10080	525960
Seconds	60	3600	86400	604800	31557600

Fonte: https://robjhyndman.com/hyndsight/seasonal-periods/.

Redes neurais recorrentes

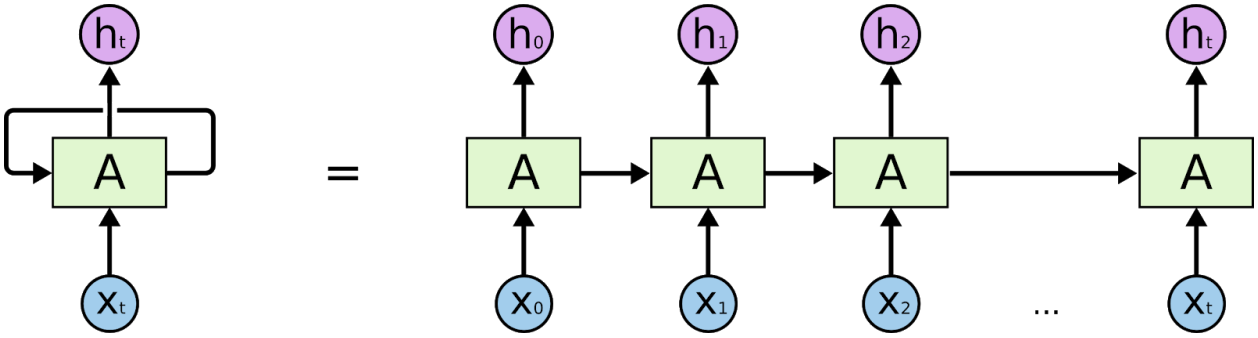
As redes neurais recorrentes (RNN – Recurrent Neural Networks) são um tipo de rede neural artificial projetada para reconhecer padrões em dados sequenciais, tais como (bem como algumas aplicações):

- Séries temporais;
- Reconhecimento de fala;
- Análise de sentimento. Exemplo: “Não gostei do filme!”
- Tradução automática de idiomas;
- Vídeos;
- DNA;
- Etc.

Mas por que não utilizar uma rede neural MLP? De maneira geral, uma rede MLP não tem noção de ordem no tempo, e a única entrada que considera é o exemplo atual a que foi exposta. As redes feedforward são amnésicas em relação ao seu passado recente; uma classificação feita de um exemplo não irá alterar a classificação do exemplo que vem na sequência.

As redes recorrentes são diferenciadas das redes MLP por um laço de recorrência conectado às suas decisões anteriores, ingerindo suas próprias saídas momento após momento como entrada (ver Figura 54).

Figura 54 – Representação de uma rede neural recorrente.



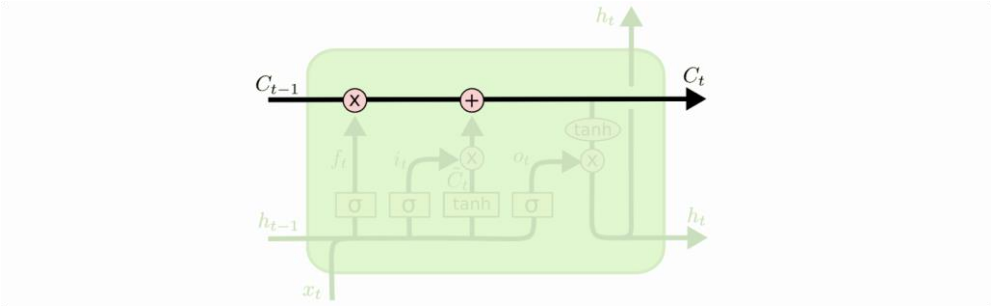
Fonte: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Apesar de serem superiores às redes MLPs no requisito “memória”, isto é, no aprendizado de padrões de dependência de uma sequência ordenada. As RNNs tradicionais podem vir a perder informações de longo prazo em sequências muito longas (“memória curta”).

Em decorrência deste problema, novas arquiteturas de redes recorrentes foram propostas com o intuito de preservar informações de longo prazo aprendidas ao longo do treinamento. As redes LSTM (Long Short Term Memory) são exemplos de arquiteturas desenhadas para preservar relações de longo prazo.

A principal característica da unidade de processamento de uma rede LSTM é a **célula de estado** (ver Figura 55), responsável por propagar informações aprendidas ao longo das unidades de processamento.

Figura 55 – Célula de estado de uma LSTM.

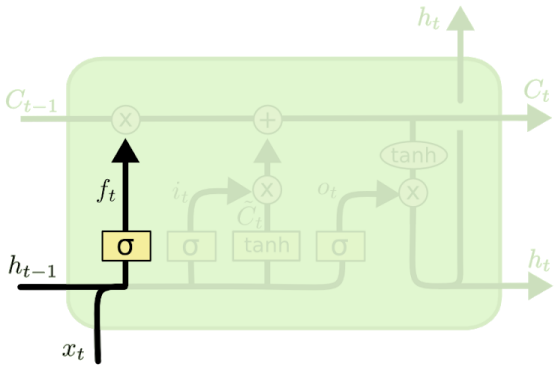


Fonte: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

As atualizações do estado C_t são feitas a partir de dois portões (*gates*, em inglês):

1. **Forget gate**: responsável por decidir qual informação será descartada da célula de estado (ver Figura 56).
2. **Input gate**: responsável por decidir qual informação será adicionada/atualizada na célula de estado.

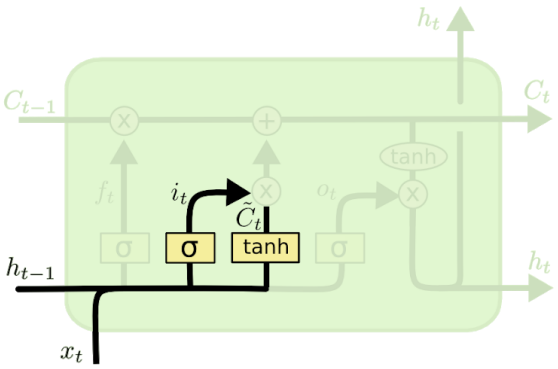
Figura 56 – Forget gate de uma LSTM.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Fonte: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Figura 57 – Input gate de uma LSTM.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Fonte: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

A escolha de uma rede LSTM em detrimento de um modelo mais simples (ARIMA) deve ser feita ponderando a relação de **benefício** que a LSTM pode trazer em termos de performance, ao **custo** de uma complexidade elevada de

desenvolvimento. Assim, alguns pontos devem ser levados em consideração na etapa de planejamento:

- A diferença em performance entre um modelo complexo e um mais simples;
- A valor agregado da performance adicional;
- O custo de implementação de modelos complexos;
- O custo de manter modelos complexos;
- Possível perda de interpretabilidade de modelos complexos.

Referências

BURKOV, Andriy. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019.

CS231N. Quick intro. In: *CS231n Convolutional Neural Networks for Visual Recognition*. Disponível em: <http://cs231n.github.io/neural-networks-1/>. Acesso em: 30 de nov. 2020.

FERNANDO, Yohan. *Garbage In, Garbage Out*. Fuzzy LogX, jan. 2019. Disponível em: <https://www.fuzzylogx.com.au/fuzzy-friday/fuzzy-friday-part-20/>. Acesso em: 30 de nov. 2020.

FORTMANN-ROE, Scott. *Accurately Measuring Model Prediction Error*. Mai. 2012. Disponível em: <http://scott.fortmann-roe.com/docs/MeasuringError.html>. Acesso em: 30 de nov. 2020.

FORTMANN-ROE, Scott. *Understanding the Bias-Variance Tradeoff*. Jun. 2012. Disponível em: <http://scott.fortmann-roe.com/docs/BiasVariance.html>. Acesso em: 30 de nov. 2020.

GEEKS FOR GEEKS. *Python | Pandas DataFrame*. Disponível em: <https://www.geeksforgeeks.org/python-pandas-dataframe/>. Acesso em: 30 de nov. 2020.

HOLMES; E. E.; SCHEURELL, M. D.; WARD, E. J. Dickey-Fuller and Augmented Dickey-Fuller tests. In: *Applied Time Series Analysis*, 2020. Disponível em: <https://nwfsc-timeseries.github.io/atsa-labs/sec-boxjenkins-aug-dickey-fuller.html>. Acesso em: 30 de nov. 2020.

HYNDMAN, Rob J. *Seasonal periods*. Nov. 2014. Disponível em: <https://robjhyndman.com/hyndsight/seasonal-periods/>. Acesso em: 30 de nov. 2020.

HYNDMAN, Rob J.; ATHANASOPOULOS, George. *Forecasting: principles and practice*. OTexts, 2013.

JAMES, Gareth et al. *An introduction to statistical learning*. 1 ed. Nova Iorque: Springer.

KERAS. Losses. Disponível em: <https://keras.io/api/losses/>. Acesso em: 30 de nov. 2020.

PANT, Ayush. *Workflow of a Machine Learning Project*. Towards data science, Medium, jan. 2019. Disponível em: <https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94>. Acesso em: 30 de nov. 2020.

Redes Neurais Recorrentes. In: *Deep Learning Book*. Disponível em: <http://deeplearningbook.com.br/redes-neurais-recorrentes/>. Acesso em: 30 de nov. 2020.

SANTOS, Vinicius dos. *Aula 01 – Classificação com KNN*. Computer Science Master. Disponível em: <https://www.computersciencemaster.com.br/2019/02/aula-01-classificacao-com-knn.html>. Acesso em: 30 de nov. 2020.

SCIKIT LEARN. *3.1. Cross-validation: evaluating estimator performance*. Disponível em: <https://scikit-learn.org/stable/modules/cross_validation.html>. Acesso em: 30 de nov. 2020.

SCIKIT-LEARN. Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 30 de nov. 2020.

SHARMA, Siddhartha. *Kernel Trick in SVM*. Analytics Vidhya, Medium, nov. 2019. Disponível em: <https://medium.com/analytics-vidhya/how-to-classify-non-linear-data-to-linear-data-bb2df1a6b781>. Acesso em: 30 de nov. 2020.

SIGMA MAGIC. *Online Sample Size Calculators - Users Beware*. Disponível em: <https://www.sigmamagic.com/blogs/online-sample-size-calculators/>. Acesso em: 30 de nov. 2020.

SINGH, Aishwarya. *Build High Performace Time Series Models using Auto ARIMA in Python and R*. Analytics Vidhya, ago. 2018. Disponível em: <https://www.analyticsvidhya.com/blog/2018/08/auto-arima-time-series-modeling-python-r/>. Acesso em: 30 de nov. 2020.

SRIVASTAVA, Tavish. *A Complete Tutorial on Time Series Modeling in R*. Analytics Vidhya, dez. 2015. Disponível em: <https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/>. Acesso em: 30 de nov. 2020.

Understanding LSTM Networks. Colah's blog, ago. 2015. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 30 de nov. 2020.