

MACHINE LEARNING

Profº Dr. Cao Ji Kan



ASSUNTO

Implementação em Python (Google.colab)
analisando um estudo de caso





Implementação ML

Dados: **Repositórios de dados**

INEP: <https://www.gov.br/inep/pt-br/acao-a-informacao/dados-abertos/microdados>

Google dataset Search: <https://datasetsearch.research.google.com/>

Portal brasileiro de dados abertos: www.dados.gov.br


Kaggle (competições Machine Learning): www.kaggle.com

UCI Machine Learning Repository: <https://archive.ics.uci.edu/ml/index.php>

OMS: <https://www.who.int/>

Paho (organização panamericana de saúde): <https://www.paho.org/en>


DrivenData (competições Ciência de Dados): <https://www.drivendata.org/>



Implementação ML

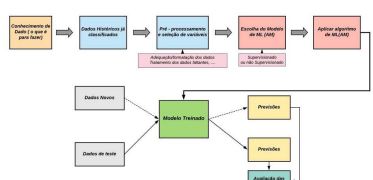
Algoritmos: **Algoritmos de Classificação**


◆ Regressão Logística	◆ Máquinas de Vetor de Suporte
◆ Naive Bayes	◆ XGBoost
◆ Árvore de decisão	◆ LightGBM
◆ Random Forest	◆ CatBoost
◆ KNN	◆ Redes Neurais Artificiais



Implementação ML

Processos: **Processos de Machine Learning (AM - Aprendizagem de Máquina)**





Implementação ML

Treinamentos: **Separação de Dados de Treino e Teste**

Dados de treino: Certa quantidade dos dados (aproximadamente 70%) destinada para treinar o algoritmo.

Dados de teste: Quantidade restante dos dados (aproximadamente 30%) para analisar o desempenho do algoritmo.

Essa separação deve ocorrer de maneira aleatória para evitar problemas nos modelos criados (Exemplo: ter uma quantidade de dados que aparecem em pequena quantidade ou nem aparecem nos dados de teste).

Implementação ML

Validação :

Validação cruzada (Cross validation)

$n_split = 4$
(k-folds)

Implementação ML

Validação :

Validação cruzada

Implementação ML

Validação :

Validação cruzada

Implementação ML

Validação :

Validação cruzada

Implementação ML

Validação :

Atenção a dois problemas no treinamento

1	Underfitting (alto viés)	Algoritmo que não se encaixa com os dados de entrada.
2	Overfitting (alta variância)	Algoritmo ótimo para os dados de entrada e ruim para dados de teste.

Implementação ML

Distribuição normal e a classe Standard Scaler da biblioteca Scikit-learn em Python. Por que padronizar/normlizar os dados?

Distribuição Normal

Podemos ilustrar isso com um exemplo que me fez entender de vez esse tópico: imagine que você tem aula às 10h todos os dias. A quantidade de pessoas que chega exatamente às 10h é grande, por exemplo, 30 pessoas. Se plotarmos um gráfico do fluxo de alunos nessa sala, obteremos exatamente às 10h um pico de alunos.

Implementação ML

Distribuição normal e a classe Standard Scaler da biblioteca Scikit-learn em Python. Por que padronizar/normlizar os dados?

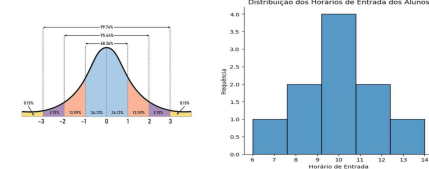
Note também que 10 minutos (09h50) antes do horário marcado (10h), sempre chegam alguns alunos adiantados, digamos 10 alunos (20 a menos que os que chegam no horário). Há também os mais esforçados que chegam 20 minutos (09h40) antes do horário marcado, digamos 5 pessoas.

Agora repetamos o processo só que de 10h pra frente. Chegariam também 10 alunos às 10h10, e também 5 alunos às 10h20.

Implementação ML

Distribuição normal e a classe Standard Scaler da biblioteca Scikit-learn em Python. Por que padronizar/normlizar os dados?

Percebe o padrão aqui? Isso é um exemplo de distribuição normal. Há ainda várias outras situações em que esse tipo de processo aparece.



Implementação ML

Distribuição normal e a classe Standard Scaler da biblioteca Scikit-learn em Python. Por que padronizar/normlizar os dados?

Algumas características são:

- A curva da distribuição é simétrica em relação à sua média
- A área total sob a curva é de 100%
- Ela prolonga-se de $-\infty$ a $+\infty$
- Ela é especificada por uma média μ e um desvio padrão σ
- A distribuição tem um formato de sino
- Em uma distribuição normal, média = moda = mediana

Implementação ML

Distribuição normal e a classe Standard Scaler da biblioteca Scikit-learn em Python. Por que padronizar/normlizar os dados?

Exemplos de distribuição NÃO normal:

Salários numa empresa: a maior parte dos funcionários possuem um salário mais baixo, enquanto poucas pessoas ficam na cadeia de comando ganhando um salário elevado, portanto, o pico do gráfico sino da distribuição normal não existe aqui.

Implementação ML

Distribuição normal e a classe Standard Scaler da biblioteca Scikit-learn em Python. Por que padronizar/normlizar os dados?

Exemplos de distribuição normal:

Notas de uma disciplina de dificuldade média: a maior parte das pessoas ficaria na média, que seria 7, enquanto alguns tirariam notas boas e uns poucos tirariam notas muito boas. Assim como alguns tirariam notas ruins e uns poucos tirariam notas muito ruins.

Implementação ML

O que é um outlier?

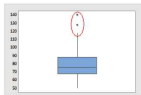
Um **outlier** é uma observação atipicamente grande ou pequeno. Outliers podem ter um efeito desproporcional sobre os resultados estatísticos, como a média, o que pode resultar em interpretações equivocadas. Por exemplo, um conjunto de dados inclui os valores: 1, 2, 3, e 34. O valor médio, 10, que é maior do que a maioria dos dados (1, 2, 3), é muito afetado pelo ponto de dados extremo, 34. Neste caso, o valor médio faz parecer que os valores de dados são mais elevados do que realmente são. Você deve investigar outliers porque eles podem fornecer informações úteis sobre os seus dados ou processo. Muitas vezes, é mais fácil de identificar **outliers** representando-se graficamente os dados.

Implementação ML

O que é um outlier?

Uso de gráficos para identificar outliers

Em boxplots, o Minitab usa um símbolo de asterisco (*) para identificar outliers. Estes outliers são observações que são pelo menos 1,5 vezes o intervalo interquartil (Q3 - Q1) a partir da borda da caixa.

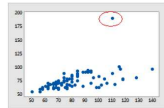


Em boxplots, o Minitab usa um asterisco (*) para identificar outliers.

Implementação ML

O que é um outlier?

Em gráficos de dispersão, pontos que estão muito longe de outros são possíveis outliers.



Este gráfico de dispersão mostra um possível outlier.

Implementação ML

Matriz de confusão

Análise do desempenho: Matriz de confusão

	NEGATIVO	POSITIVO
NEGATIVO	VERDADEIRO NEGATIVO	FALSO POSITIVO
POSITIVO	FALSO NEGATIVO	VERDADEIRO POSITIVO

Implementação ML

Matriz de confusão

Análise do desempenho: Matriz de confusão

Gênero	Estado Civil	Grau de Instrução	Produto
Masculino	Casado	Básico	A
Feminino	Casado	Superior	B
Feminino	Viuva	Básico	A
Masculino	Solteiro	Superior	B
Feminino	Solteira	Superior	B

Previsores (atributos)

Classe (target)

Said	Referência
0	0 1
1	1 1

40% acerto

Implementação ML

MÉTRICAS DE AVALIAÇÃO DE GRUPOS

A *silhueta* considera as distâncias entre as instâncias e calcula o quanto uma instância é mais similar ao seu cluster quando comparada à instância mais próxima de outro. O índice de **Davies-Bouldin** leva em consideração as distâncias para o centroide. Já o índice de **Calinski-Harabasz** é a relação entre a dispersão *inter* e *intracluster*. As três métricas são comumente usadas em conjunto.

Implementação ML

3 Avaliação de grupos

Os grupos formados pelo algoritmo de K-Médias podem variar em quantidade e em relação a quais instâncias pertencem ao mesmo grupo. Diferentes parâmetros resultarão em diferentes níveis de dispersão, tanto entre os grupos (*intercluster*) quanto entre as instâncias (*intracluster*). Esses conceitos são apresentados na Figura 4.

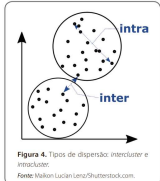



Figura 4. Tipos de dispersão intercluster e intracluster.
Fonte: Wolkon Lucian Lanza/Shutterstock.com.

Implementação ML



Algoritmo Kmeans:

Parâmetro: "algorithm" seleciona entre três formatos de cálculo das distâncias:

1. **full** — padrão dos algoritmos de k-means que calcula a distância de todos os pontos com relação ao centro;
2. **elkan** — converge para os mesmos resultados, mas reduz, a partir de triangulações, a quantidade de distâncias calculadas, eliminando muitas etapas desnecessárias/redundantes;
3. **auto** — recorre ao método convencional para conjuntos de dados esparsos, em que o algoritmo elkan implementado é incapaz de convergir, e ao método elkan para conjuntos de dados densos