

Machine Learning



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



Avaliação de Modelos Preditivos

Responsável pelo Conteúdo:

Prof. Me. Orlando da Silva Junior

Revisão Textual:

Prof.^a Dr.^a Selma Aparecida Cesarin

UNIDADE

Avaliação de Modelos Preditivos



- Introdução à Avaliação de Modelos Preditivos;
- Métricas de Erros;
- Matriz de Confusão e Medidas de Desempenho;
- Técnicas de Amostragem;
- Aplicações em R;
- Aplicações em Python.



OBJETIVOS DE APRENDIZADO

- Identificar estratégias para avaliação de modelos preditivos;
- Avaliar métodos de aprendizagem supervisionada.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:



Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Introdução à Avaliação de Modelos Preditivos

Como você deve ter observado até o momento, não existe apenas uma única técnica universal para a solução de problemas de *Machine Learning*.

Você estudou diferentes métodos de indução de modelos preditivos e deverá experimentar essas diferentes abordagens para resolver os problemas do mundo real.

Apesar disso, o estudo das características dos métodos pode nos ajudar a selecionar melhor quais modelos deverão ser experimentados no nosso problema.

Por exemplo, o Algoritmo 1-NN não parece ser uma boa opção em um problema cujo conjunto de dados apresente altas dimensões. Nesse caso, o mais adequado é descartá-lo da nossa lista de algoritmos candidatos e selecionar um outro que tenha maior chance de resolver o problema.

Perceba ainda que, como você deverá realizar diversos experimentos para encontrar não apenas o melhor método, mas também a melhor configuração de hiperparâmetros para ele, será necessário planejar a realizar esses experimentos a fim de que se demonstre a sua eficácia e se permita a sua reprodução em qualquer momento futuro.

Podemos avaliar um experimento por muitos aspectos, mas, nesta Unidade, nós nos concentraremos em avaliar o desempenho preditivo dos métodos que estão sendo avaliados.

Iremos conhecer, também, algumas formas de realizar essa avaliação e as principais medidas usadas para interpretar os resultados obtidos. Ao final da Unidade, você vai aprender a avaliar os principais métodos de *Machine Learning* nas linguagens *R* e *Python*.

Métricas de Erros

Quando avaliamos o desempenho de um algoritmo supervisionado, estamos analisando o desempenho da função preditora gerada por esse algoritmo.

Sabemos que essa função será utilizada na rotulação dos novos exemplos, aqueles que não foram apresentados na etapa de treinamento.

Embora problemas de classificação e regressão sejam tratados como problemas supervisionados, o rótulo de cada tipo de problema exige que trabalhemos com métricas diferentes.

Em problemas de classificação, usaremos a taxa de erro. Em regressão, usaremos o erro quadrático médio.

Vamos considerar que a taxa de predições incorretas de um classificador \hat{f} é a taxa de erro desse preditor.

A avaliação desse classificador será realizada para um conjunto de dados com n exemplos. A taxa de erro corresponde à seguinte equação (FACELLI *et al.*, 2011):

$$erro(\hat{f}) = \frac{\sum_{i=1}^n I(y_i \neq \hat{f}(x_i))}{n}$$

Onde:

$$I(a) = \begin{cases} 0, & \text{se } a \text{ falso} \\ 1, & \text{se } a \text{ verdadeiro} \end{cases}$$

Na equação, realiza-se uma comparação entre o valor real y_i e o valor prevido por $\hat{f}(x_i)$.

Se a resposta dessa comparação for verdadeira, ou seja, os valores comparados forem diferentes, a função computa +1 para a taxa de erro. Caso contrário, ou seja, os valores comparados sejam iguais, a função soma 0 à taxa de erro, deixando inalterado o valor da taxa de erro final.

A taxa de erro é uma medida que varia entre 0 e 1, sendo os valores mais próximos de 0 melhores para a avaliação do modelo preditivo. A taxa de acerto (ou acurácia) pode ser medida pelo complemento da taxa de erro:

$$acurácia(\hat{f}) = 1 - erro(\hat{f})$$

No caso da acurácia, quanto mais próximo de 1 estiver o resultado da equação, melhor será o desempenho preditivo.

Já no caso dos problemas de regressão, o erro do regressor \hat{f} pode ser computado por meio da distância entre o valor real y_i e o valor prevido por $\hat{f}(x_i)$.

Uma das medidas mais conhecidas que realiza este cálculo é o Erro Quadrático Médio (*MSE – Mean Squared Error*). O MSE é sempre positivo e quanto mais baixo seu valor, melhor é o resultado do modelo.

Ele pode ser calculado por meio da seguinte equação:

$$MSE(\hat{f}) = \frac{\sum_{i=1}^n (y_i - \hat{f}(x_i))^2}{n}$$

Matriz de Confusão e Medidas de Desempenho

Uma alternativa às métricas de erro apresentadas anteriormente para problemas de classificação é a matriz de confusão, que apresenta a quantidade de previsões corretas e incorretas de cada classe.

A Figura 1 ilustra a construção de uma matriz de confusão para um problema de duas classes, A e B.

		Classe predita	
		A	B
Classe verdadeira	A	74	26
	B	18	32

Figura 1 – Exemplo de matriz de confusão para 150 exemplos

Supondo que essa matriz tenha sido obtida a partir da aplicação de um determinado algoritmo de *Machine Learning* em um conjunto de dados com 150 exemplos, sendo 100 da classe A e 50 da classe B, podemos interpretar a matriz da seguinte forma:

- 74 exemplos da classe A foram preditos corretamente como sendo da classe A;
- 26 exemplos da classe B foram preditos incorretamente como sendo da classe B;
- 18 exemplos da classe B foram preditos incorretamente como sendo da classe A; e
- 32 exemplos da classe B foram preditos corretamente como sendo da classe B.

Por meio da matriz de confusão, conseguimos realizar uma análise quantitativa de quais classes o método de aprendizado aplicado teve maior dificuldade. Observe que, no exemplo apresentado, dos 150 exemplos avaliados, 106 foram corretamente preditos, enquanto 44 exemplos tiveram sua classificação realizada incorretamente pelo algoritmo.

Muitas vezes, em problemas com apenas duas classes, consideramos os termos “classe positiva” (+) e “classe negativa” (-) para designar as classes que estão sendo avaliadas.

Por exemplo, em um problema de análise de crédito para o Mercado Financeiro, os “bons pagadores” fariam parte da classe positiva, enquanto os “maus pagadores” estariam compondo a classe negativa.

Em problemas com mais de duas classes, a matriz de confusão também poderá ser construída, bastando observar que a matriz deverá ter a dimensão $k \times k$, sendo k a quantidade de classes avaliadas.

Uma característica interessante da matriz de confusão é que ela nos permite obter medidas de desempenho específicas a respeito do preditor. Para compreender essas medidas, vamos voltar à matriz de confusão e descrever os nomes de cada valor quantitativo que podemos obter.

A Figura 2 ilustra os nomes desses valores, em que:

- VP corresponde à quantidade de verdadeiros positivos;
- FP corresponde à quantidade de falsos positivos;
- FN corresponde à quantidade de falsos negativos; e
- VN corresponde à quantidade de verdadeiros negativos.

		Classe predita	
		+	-
Classe verdadeira	+	VP	FN
	-	FP	VN

Figura 2 – Matriz de confusão

Agora sim, a partir da matriz de confusão com os nomes técnicos de cada célula, você poderá conhecer as principais medidas de desempenho para a avaliação de preditores supervisionados.

Sabendo que $N = VP + FP + FN + VN$, as principais medidas de desempenho são (FACELLI *et al.*, 2011):

- **Taxa de erro na classe positiva:** também conhecida como taxa de falsos negativos (TFN):

$$err_+(\hat{f}) = \frac{FN}{VP + FN}$$

- **Taxa de erro na classe negativa:** também conhecida como taxa de falsos positivos (TFP):

$$err_-(\hat{f}) = \frac{FP}{VN + FP}$$

- **Taxa de erro total:** corresponde à soma da diagonal secundária dividida pela quantidade total de objetos avaliados:

$$err(\hat{f}) = \frac{FP + FN}{N}$$

- **Acurácia:** pode ser entendida como um valor que mede o grau de conformidade de um valor obtido com seu valor real:

$$acurácia(\hat{f}) = \frac{VP + VN}{N}$$

- **Precisão:** proporção de exemplos positivos preditos corretamente entre todos aqueles preditos como positivos:

$$precisão(\hat{f}) = \frac{VP}{VP + FP}$$

- **Sensibilidade (ou revocação ou recall):** corresponde à taxa de acerto na classe positiva:

$$recall(\hat{f}) = \frac{VP}{VP + FN}$$

- **Especificidade:** corresponde à taxa de acerto na classe negativa:

$$especificidade(\hat{f}) = \frac{VN}{VN + FP}$$

Todas as medidas de desempenho apresentadas trabalham com valores entre 0 e 1, inclusive. Além disso, elas podem ser generalizadas para problemas com mais de duas classes.

Nesse caso, você deverá considerar uma medida para cada classe. Por exemplo, em um problema com 5 classes, você deverá calcular a taxa de erro para cada uma delas, tendo, ao final, 5 valores.

Técnicas de Amostragem

A utilização de técnicas de amostragem para a seleção de parte dos dados é uma etapa importante na avaliação de modelos preditivos. Utilizar o mesmo conjunto de dados para as fases de treinamento e avaliação do preditor pode prejudicar nossa análise do desempenho do algoritmo, vez podermos produzir estimativas otimistas acerca dos resultados.

Existem duas técnicas de amostragem bastante comuns em *Machine Learning* e que vamos conhecer e praticar agora!

A primeira delas é conhecida como *holdout*. A Figura 3 ilustra a aplicação da Técnica. Nela, dividimos o conjunto de dados em uma proporção de k para treinamento e $1-k$ para teste.

Em geral, usamos $k=\frac{2}{3}$. Por exemplo, se o nosso conjunto de dados possui 300 exemplos, usaremos $\frac{2}{3}$ para treinamento (200 exemplos) e $\frac{1}{3}$ para teste (100 exemplos).

Se o nosso conjunto de dados é suficientemente grande, essa abordagem é uma das melhores para a experimentação de algoritmos de *Machine Learning*.

Um ponto negativo a respeito dessa técnica é que a nossa avaliação do algoritmo pode ser afetada pela seleção dos exemplos.

Para contornar esse problema e permitir que os resultados fiquem mais independentes da seleção dos exemplos, você pode construir diversas partições aleatórias e obter a média e o desvio padrão dos resultados.

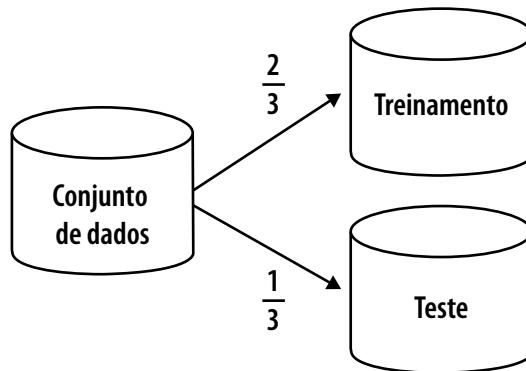


Figura 3 – Amostragem *holdout*

A segunda técnica que você vai conhecer chama-se validação cruzada (*cross validation*). Nesse método, o conjunto de dados é dividido em k subconjuntos de tamanho aproximadamente igual.

Os exemplos dos $k-1$ subconjuntos são utilizados para treinamento, e o subconjunto restante é utilizado para teste.

Esse procedimento é repetido k vezes, sendo que, a cada nova iteração, um subconjunto diferente é utilizado para teste.

A Figura 4 ilustra a aplicação desse método quando $k=10$, que é o número padrão em experimentos de *Machine Learning*.

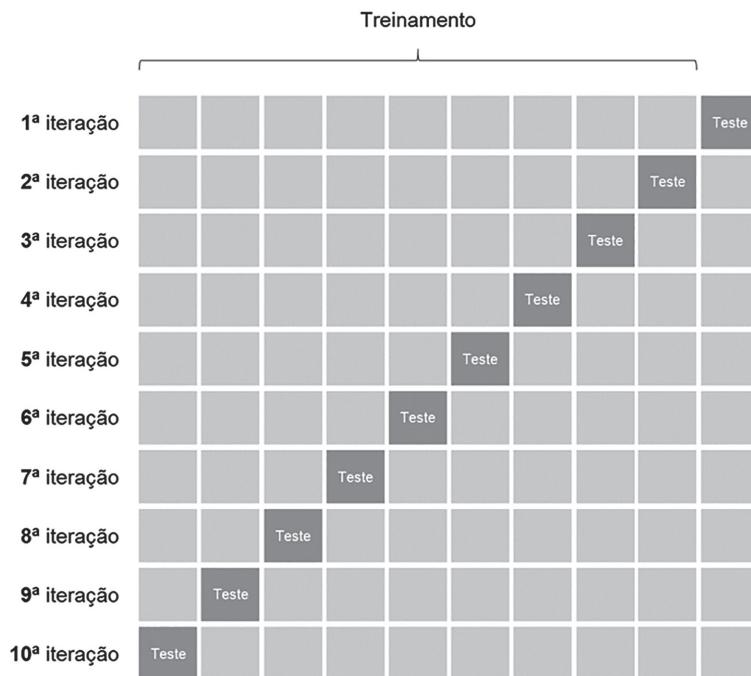


Figura 4 – Amostragem de validação cruzada com 10 subconjuntos

Uma das principais vantagens do método de validação cruzada é que ele pode ser utilizado em conjuntos de dados pequenos ou que não sejam tão grandes a ponto de permitir uma separação única entre conjuntos de treinamento e teste.

Por outro lado, quando $k > 2$, uma parte dos dados passa a ser compartilhada entre os subconjuntos de treinamento, impedindo a completa independência dos subconjuntos.

Embora existam muitas outras técnicas de amostragem e seleção de dados, as duas técnicas que você acabou de conhecer servem para a maior parte dos problemas de ciência de dados que envolvem algoritmos de *Machine Learning*.

Aplicações em R

Agora que você já conhece como funcionam as métricas de erro e as principais medidas de desempenho para a avaliação de modelos preditivos, você vai aprender a construir modelos de *Machine Learning* na linguagem *R*.

Para realizar os experimentos, você precisará do ambiente *R* e do ambiente de desenvolvimento *RStudio*. Ambos os softwares são gratuitos e facilmente instaláveis no *Microsoft Windows*.



Ambiente *R*, disponível em: <https://bit.ly/2EM5UmN>

Ambiente de Desenvolvimento *RStudio*, disponível em: <https://bit.ly/3juu9F2>

A Figura 5 apresenta a interface do software *RStudio*.

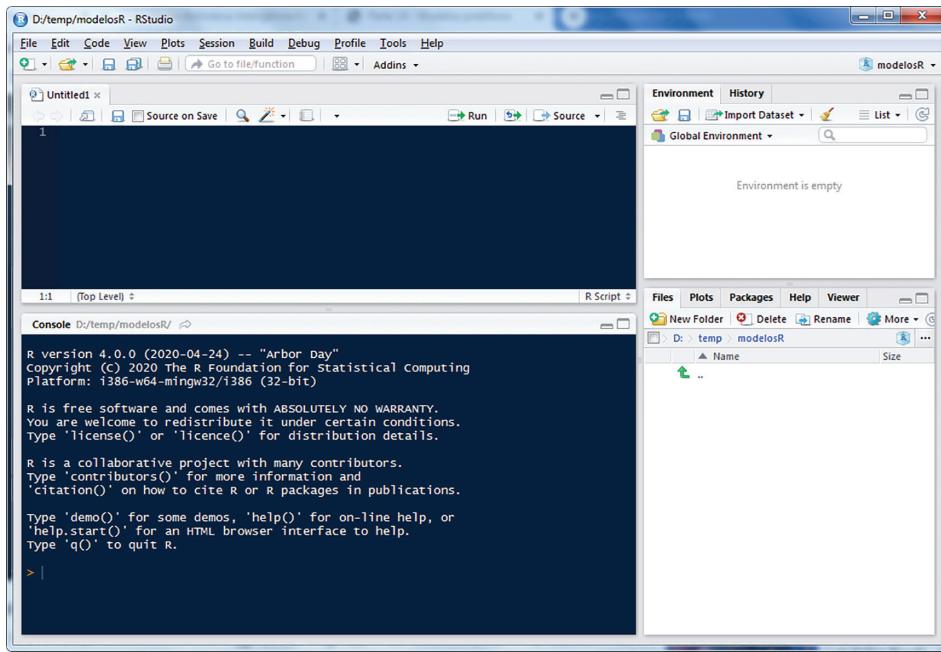


Figura 5 – Interface inicial do software Rstudio

Inicialmente, devemos identificar os principais métodos de *Machine Learning* dentro da linguagem *R*. A Tabela 1 indica os nomes das bibliotecas e as funções que deverão ser carregadas e utilizadas, respectivamente.

Tabela 1 – Bibliotecas da linguagem *R* para *Machine Learning*

Método	Pacote	Função
k-NN	class	knn
Árvore de Decisão	rpart	rpart
Naïve Bayes	e1071	naiveBayes
Redes Neurais	RSNNS	mlp
SVM	e1071	svm

Como exemplo, vamos usar a função *knn()*, que é parte do pacote *class* e requer quatro entradas:

- Uma matriz contendo os preditores associados aos dados de treinamento;
- Uma matriz contendo os preditores associados com os dados os quais queremos realizar previsões;
- Um vetor contendo os rótulos das classes para as observações de treinamento; e
- Um valor para K , correspondente ao número de vizinhos próximos a ser usado pelo classificador.

O seguinte código treina e avalia um modelo de *Machine Learning* em Linguagem *R* usando a base de dados *iris*:

```
# Carrega o pacote class
library(class)

# Define a semente do experimento
set.seed(42)

# Separa os exemplos de treinamento aleatoriamente:
#   2/3 para treinamento
#   1/3 para teste
train <- sample(nrow(iris), nrow(iris)*2/3)

# Conjunto de treinamento
iris_train <- iris[train, 1:4]
iris_train_labels <- iris[train, 5]

# Conjunto de teste
iris_test <- iris[-train, 1:4]
iris_test_labels <- iris[-train, 5]

# Treinamento do modelo com 1-NN
pred <- knn(iris_train, iris_test, iris_train_labels, k = 1)

# Avaliação
table(pred, iris_test_labels)
```

O resultado preditivo é apresentado na Figura 6, que mostra a matriz de confusão para os 50 exemplos selecionados para o conjunto de teste.

Usando o algoritmo 1-NN:

- Dos 17 exemplos da classe setosa, 17 foram classificados corretamente como sendo da classe setosa;
- Dos 18 exemplos da classe versicolor, 15 foram classificados corretamente como sendo da classe versicolor e 3 foram classificados incorretamente como sendo da classe virginica;
- Dos 15 exemplos da classe virginica, 14 foram classificados corretamente como sendo da classe virginica e apenas 1 foi classificado incorretamente como sendo da classe versicolor.

		iris_test_labels		
pred		setosa	versicolor	virginica
setosa	17	0	0	
versicolor	0	15	3	
virginica	0	1	14	

Figura 6 – Matriz de confusão para 1-NN

Por meio da matriz de confusão você pode encontrar as diversas métricas de avaliação de desempenho. Nesse exemplo, o modelo gerado pelo algoritmo 1-NN alcançou 92% de acurácia preditiva.

Aplicações em Python

Para construir e validar modelos de *Machine Learning* usando a Linguagem *Python*, você precisará de um ambiente que execute os programas que iremos construir.

A distribuição Anaconda é uma das mais populares para Ciência de Dados e pode ser utilizada para essa finalidade, vez que ela permite a reprodução dos experimentos por meio da construção de *notebooks* usando *Jupyter*.



Distribuição Anaconda: <https://bit.ly/3cMY88F>

De modo semelhante ao que fizemos com a linguagem R, vamos construir e avaliar uma aplicação de *Machine Learning* usando a Linguagem *Python*. Após a execução do *Jupyter* em seu computador (ou utilizando o *Google Colab*), crie um novo *notebook* para começarmos o experimento.

Comece importando as bibliotecas, classes e funções que utilizaremos para a aplicação.

A Figura 7 apresenta quais são elas e como você deverá fazer a importação usando o *Jupyter Notebook*.

Das importações, usaremos:

- A biblioteca Pandas, destinada à manipulação de dados;
- A função **load_wine()**, correspondente à base de dados que usaremos;
- A função **train_test_split()**, destinada à reamostragem dos dados;
- A classe *DecisionTreeClassifier*, referente à indução de Árvore de Decisão; e
- As funções **accuracy_score()** e **precision_score()**, usadas nesse projeto para calcular o desempenho preditivo do modelo.

```

1 import pandas as pd
2 from sklearn.datasets import load_wine
3 from sklearn.model_selection import train_test_split
4 from sklearn.tree import DecisionTreeClassifier
5 from sklearn.metrics import accuracy_score, precision_score

```

Figura 7 – Bibliotecas do Python

Em seguida, você deverá carregar o conjunto de dados e separá-lo em dois subconjuntos disjuntos: treinamento e teste.

A Figura 8 mostra como você pode fazer isso em *Python*. Neste caso, 70% do conjunto de dados original será destinado ao treinamento do modelo e 30% será utilizado para teste.

A função **`train_test_split()`** devolverá quatro subconjuntos de dados:

- **`X_train`:** subconjunto de entrada de treinamento;
- **`X_test`:** subconjunto de entrada de teste;
- **`y_train`:** subconjunto de saída de treinamento; e
- **`y_test`:** subconjunto de saída de teste.

```

1 features, target = load_wine(return_X_y=True)
2 X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.30, random_state=42)

```

Figura 8 – Carregamento e separação do conjunto de dados

Após a identificação dos subconjuntos de treinamento e teste, você utilizará o subconjunto de treinamento para gerar o modelo da árvore de decisão, como mostra a Figura 9:

```

1 model = DecisionTreeClassifier(criterion='entropy', random_state=42)
2 model.fit(X_train, y_train)

DecisionTreeClassifier(criterion='entropy', random_state=42)

```

Figura 9 – Geração do modelo de árvore de decisão

Para calcular o desempenho preditivo do modelo gerado, você deverá aplicar o modelo no subconjunto de teste, como mostra a Figura 10:

```

1 y_pred = model.predict(X_test)

1 print("Acurácia: %.3f" % accuracy_score(y_test, y_pred))
2 print("Precisão: %.3f" % precision_score(y_test, y_pred, average='macro'))

Acurácia: 0.852
Precisão: 0.853

```

Figura 10 – Predição e cálculo do desempenho preditivo

Observe que utilizamos as métricas de acurácia e precisão para este projeto, cujo objetivo é classificar diferentes tipos de vinhos (classificação multiclasse). Por meio desse exemplo, simples você aprendeu as principais tarefas de *Machine Learning* em *Python*.

O que você achou?

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:



Livros

Inteligência Artificial: Uma abordagem de aprendizado de máquina

FACELI, K. et al. **Inteligência Artificial: Uma abordagem de aprendizado de máquina**. Rio de Janeiro: LTC, 2011



Leitura

Machine learning para análises preditivas em saúde: exemplo de aplicação para prever óbito em idosos de São Paulo, Brasil

SANTOS, H. G. dos. et al. **Machine learning para análises preditivas em saúde: exemplo de aplicação para prever óbito em idosos de São Paulo, Brasil**. **Cadernos de Saúde Pública**, Rio de Janeiro, v. 35, n. 7, e00050818, 2019.

<https://bit.ly/30mc0Hc>

Avaliação de Modelos de Predição e Previsão Construídos por Algoritmos de Aprendizado de Máquina em Problemas de Cidades Inteligentes

SAMPAIO, I. G. B. et al. **Avaliação de Modelos de Predição e Previsão Construídos por Algoritmos de Aprendizado de Máquina em Problemas de Cidades Inteligentes**. **Simpósio Brasileiro de Sistemas de Informação**, Porto Alegre, 2019.

<https://bit.ly/2HLNt2T>

Modelo de classificação de risco de crédito de empresas

BRITO, G. A. S.; ASSAF NETO, A. **Modelo de classificação de risco de crédito de empresas**. **Revista Contabilidade & Finanças**, São Paulo, v. 19, n. 46, p. 18-29, Abril, 2008.

<https://bit.ly/347vlba>

Referências

- CASTRO, L. N. de; FERRARI, D. G. **Introdução à mineração de dados:** conceitos básicos, algoritmos e aplicações. São Paulo: Saraiva, 2016.
- DA SILVA, L. A.; PERES, S. M.; BOSCAROLI, C. **Introdução à mineração de dados:** com aplicações em R. São Paulo: Grupo GEN LTC, 2017.
- FACELI, K. *et al.* **Inteligência Artificial:** uma abordagem de aprendizado de máquina. Rio de Janeiro: LTC, 2011.
- GRUS, J. **Data Science do zero:** Primeiras regras com o Python. Rio de Janeiro: Alta Books, 2019.
- MITCHELL, T. M. **Machine learning.** New York: McGraw-Hill, 1997.



Cruzeiro do Sul
Educacional