

Cristian César Martins – RA: 799714  
Rodrigo Pavão Coffani Nunes – RA: 800345  
Vinícius de Oliveira Guimarães – RA: 802431  
Vitor Gabriel Orsin – RA: 801575

# **Trabalho Prático 1**

## **Aprendizado de Máquina 1**

São Carlos/SP, Brasil

27 de Junho, 2023

Cristian César Martins – RA: 799714  
Rodrigo Pavão Coffani Nunes – RA: 800345  
Vinícius de Oliveira Guimarães – RA: 802431  
Vitor Gabriel Orsin – RA: 801575

## **Trabalho Prático 1**

### **Aprendizado de Máquina 1**

Relatório científico de pesquisa do primeiro trabalho da disciplina de Aprendizado de Máquina 1, com o dataset sobre o tema "Qualidade da Água"

Universidade Federal de São Carlos – UFSCar  
Departamento de Computação  
Bacharelado em Ciência da Computação

São Carlos/SP, Brasil  
27 de Junho, 2023

# LISTA DE ILUSTRAÇÕES

Figura 1 – Verificação de dados duplicados . . . . .	7
Figura 2 – <i>Output</i> de <code>data.info()</code> . . . . .	7
Figura 3 – Removendo dados com valores não-numéricos . . . . .	8
Figura 4 – Boxplot para visualização de outliers . . . . .	9
Figura 5 – Heatmap de correlação dos atributos . . . . .	10
Figura 6 – Exemplo de Árvore de Decisão - Surf or Don't surf . . . . .	11
Figura 7 – Resultados da aplicação da Árvore de Decisão . . . . .	12
Figura 8 – Resultados da aplicação da Árvore de Decisão (dados normalizados) . .	13
Figura 9 – Resultados da aplicação de Arvore de Decisão com Cross-Validation k-fold	14
Figura 10 – Resultados da aplicação de Arvore de Decisão com Cross-Validation k-fold (dados normalizados) . . . . .	15
Figura 11 – Aplicação do método de classificação KNN . . . . .	16
Figura 12 – Aplicação do método de classificação KNN (dados normalizados) . . . .	17
Figura 13 – Aplicação de KNN com Cross-Validation k-fold . . . . .	18
Figura 14 – Aplicação de KNN com Cross-Validation k-fold (dados normalizados) .	19
Figura 15 – Aplicação de MLP . . . . .	20
Figura 16 – Aplicação de MLP (dados normalizados) . . . . .	21
Figura 17 – Aplicação de MLP com Cross-Validation . . . . .	22
Figura 18 – Aplicação de MLP com Cross-Validation (dados normalizados) . . . . .	23

# LISTA DE TABELAS

Tabela 1 – Colunas presentes no <i>dataset</i> . . . . .	6
Tabela 2 – Valores obtidos a partir da Figura 7 . . . . .	12
Tabela 3 – Valores obtidos a partir da Figura 8 . . . . .	13
Tabela 4 – Valores obtidos a partir da Figura 9 . . . . .	14
Tabela 5 – Valores obtidos a partir da Figura 10 . . . . .	15
Tabela 6 – Valores obtidos a partir da Figura 11 . . . . .	16
Tabela 7 – Valores obtidos a partir da Figura 12 . . . . .	17
Tabela 8 – Valores obtidos a partir da Figura 13 . . . . .	18
Tabela 9 – Valores obtidos a partir da Figura 14 . . . . .	19
Tabela 10 – Valores obtidos a partir da Figura 15 . . . . .	20
Tabela 11 – Valores obtidos a partir da Figura 16 . . . . .	21
Tabela 12 – Valores obtidos a partir da Figura 17 . . . . .	22
Tabela 13 – Valores obtidos a partir da Figura 18 . . . . .	23
Tabela 14 – Tabela comparativa de acurácia e f1_score dos algoritmos executados .	24

# SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>5</b>
<b>1.1</b>	<b>Enunciado do trabalho . . . . .</b>	<b>5</b>
<b>1.2</b>	<b><i>Dataset</i> escolhido . . . . .</b>	<b>5</b>
1.2.1	Colunas . . . . .	5
<b>1.3</b>	<b>Problemas a serem enfrentados . . . . .</b>	<b>6</b>
<b>2</b>	<b>DISCUSSÃO E RESULTADOS . . . . .</b>	<b>7</b>
<b>2.1</b>	<b>Abordagens iniciais . . . . .</b>	<b>7</b>
2.1.1	Removendo valores problemáticos . . . . .	8
<b>2.2</b>	<b>Seleção e redução de atributos . . . . .</b>	<b>8</b>
2.2.1	Boxplot . . . . .	8
2.2.2	Heatmap . . . . .	9
2.2.3	Normalização de dados . . . . .	9
<b>2.3</b>	<b>Métodos de classificação . . . . .</b>	<b>10</b>
2.3.1	Árvore de Decisão . . . . .	11
2.3.1.1	Árvore de Decisão com Cross-Validation . . . . .	13
2.3.2	KNN . . . . .	15
2.3.2.1	KNN com Cross-Validation . . . . .	18
2.3.3	Rede Neural . . . . .	20
2.3.3.1	Rede Neural com Cross-Validation . . . . .	22
<b>3</b>	<b>CONCLUSÃO . . . . .</b>	<b>24</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>25</b>

# 1 INTRODUÇÃO

Grandes conjuntos de dados estão cada vez mais presentes em nosso dia a dia. Dessa forma, a análise exploratória destes dados também se torna uma questão importante, possibilitando um grande uso para a área de aprendizado de máquina.

## 1.1 Enunciado do trabalho

Para a realização deste trabalho, é necessário escolher um conjunto de dados (*dataset*) que seja: relevante, desafiador e complexo. Após a escolha, deve-se explorá-lo e então solucionar eventuais problemas de atributos, como selecioná-los, reduzi-los e criar visualizações. É preciso deixar o conjunto de dados em condições ideais para aplicação dos métodos, sendo necessário uma explicação e justificativa de cada técnica aplicada.

Com o conjunto de dados devidamente preparado, devem-se ser aplicados diferentes métodos de classificação, justificando a escolha, o ajuste de parâmetros, e comparando de forma adequada os resultados. Ao final, apresentar gráficos que ilustrem os resultados encontrados.

## 1.2 *Dataset* escolhido

O conjunto de dados escolhidos para ser explorado foi encontrado no Kaggle e seu tema é a qualidade da água<sup>1</sup>. É composto por dados de elementos presentes na água e disponibilizado com uma breve descrição de cada elemento.

O *dataset* possui 21 colunas, sendo vinte atributos e uma de classificação. Com 8000 linhas, foi considerado um bom conjunto a ser escolhido pela quantidade satisfatória de amostras e a presença de diversos atributos a serem analisados.

### 1.2.1 Colunas

As colunas presentes no *dataset* e suas descrições fornecidas são as estão presentes na Tabela 1. Observando as descrições providas, é intuitivo de imaginar que a classificação seria tão simples quanto conferir os valores de todos os atributos e, caso algum passasse de seu *threshold*<sup>2</sup>, classificá-lo como "não seguro".

Entretanto, esse não é o caso. Os elementos não são independentes entre si, ou seja, os seus valores de forma isolada não são suficientes para definir sua classificação final.

<sup>1</sup> Obtido em <<https://www.kaggle.com/datasets/mssmartypants/water-quality>>.

<sup>2</sup> Limite, limiar.

Tabela 1 – Colunas presentes no *dataset*

<b>coluna</b>	<b>(tradução) descrição</b>
aluminium	(alumínio) perigoso se maior que 2.8
ammonia	(amônia) perigoso se maior que 32.5
arsenic	(arsênio) perigoso se maior que 0.01
barium	(bário) perigoso se maior que 2
cadmium	(cádmio) perigoso se maior que 0.005
chloramine	(cloraminas) perigoso se maior que 4
chromium	(crômio) perigoso se maior que 0.1
copper	(cobre) perigoso se maior que 1.3
flouride	(fluoreto) perigoso se maior que 1.5
bacteria	(bactérias) perigoso se maior que 0
viruses	(vírus) perigoso se maior que 0
lead	(chumbo) perigoso se maior que 0.015
nitrate	(nitratos) perigoso se maior que 10
nitrite	(nitritos) perigoso se maior que 1
mercury	(mercúrio) perigoso se maior que 0.002
perchlorate	(perclorato) perigoso se maior que 56
radium	(rádio) perigoso se maior que 5
selenium	(selênio) perigoso se maior que 0.5
silver	(prata) perigoso se maior que 0.1
uranium	(urânio) perigoso se maior que 0.3
is_safe	(seguro) atributo de classe (0 - não seguro, 1 - seguro)

Fonte: os autores

### 1.3 Problemas a serem enfrentados

Os problemas a serem enfrentados neste trabalho são os mesmos que ocorrem com grandes conjuntos de dados: entender corretamente os atributos e suas correlações, assim como filtrar quais são os mais importantes quando se trata de atribuir uma classificação correta.

A partir de análises presentes em etapas seguintes deste relatório, será possível determinar se existem amostras com dados problemáticos, quais atributos possuem uma grande influência na classificação ou quais podem ser considerados redundantes. Assim como, após a redução de atributos e aplicação de filtros, quais são os métodos mais adequados para analisar os dados uma vez já pré-processados.

## 2 DISCUSSÃO E RESULTADOS

### 2.1 Abordagens iniciais

Após carregar os dados, utilizando a biblioteca pandas<sup>1</sup>, iremos estudar o *dataset* e buscar por dados nulos ou duplicados.

Figura 1 – Verificação de dados duplicados

```
[ ] 1 data.duplicated().any()
False
```

Fonte: os autores

Figura 2 – *Output* de data.info()

```
0s 1 data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7999 entries, 0 to 7998
Data columns (total 21 columns):
#   Column                Non-Null Count  Dtype
---  -
0   aluminium             7999 non-null   float64
1   ammonia                7999 non-null   object
2   arsenic                7999 non-null   float64
3   barium                 7999 non-null   float64
4   cadmium                7999 non-null   float64
5   chloramine             7999 non-null   float64
6   chromium               7999 non-null   float64
7   copper                 7999 non-null   float64
8   fluoride               7999 non-null   float64
9   bacteria               7999 non-null   float64
10  viruses                 7999 non-null   float64
11  lead                   7999 non-null   float64
12  nitrates               7999 non-null   float64
13  nitrites               7999 non-null   float64
14  mercury                7999 non-null   float64
15  perchlorate            7999 non-null   float64
16  radium                 7999 non-null   float64
17  selenium               7999 non-null   float64
18  silver                 7999 non-null   float64
19  uranium                7999 non-null   float64
20  is_safe                7999 non-null   object
dtypes: float64(19), object(2)
memory usage: 1.3+ MB
```

Fonte: os autores

<sup>1</sup> Disponível em: <<https://pandas.pydata.org/>>

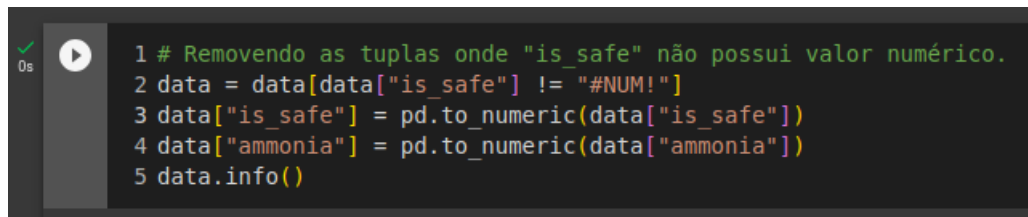


Analisando o resultado apresentado na Figura 1 podemos afirmar que não existem dados duplicados. E, com base na Figura 2, vemos que em cada coluna existem três amostras onde a coluna `is_safe` possui um valor não-numérico, o que não ajuda nos algoritmos de classificação.

### 2.1.1 Removendo valores problemáticos

Buscando quais são as colunas onde isso ocorre, encontramos que elas também possuem a coluna `ammonia` com o mesmo dado. Dessa forma, por serem uma porcentagem pequena em comparação ao conjunto inteiro, não há problema em remover essas amostras para que não causem erros ou inconsistências nos métodos a serem aplicados.

Figura 3 – Removendo dados com valores não-numéricos



```
1 # Removendo as tuplas onde "is_safe" não possui valor numérico.
2 data = data[data["is_safe"] != "#NUM!"]
3 data["is_safe"] = pd.to_numeric(data["is_safe"])
4 data["ammonia"] = pd.to_numeric(data["ammonia"])
5 data.info()
```

Fonte: os autores

Com a remoção das amostras com dados não-consistentes, é necessário converter os tipos das colunas para dados numéricos para facilitar futuros métodos.

## 2.2 Seleção e redução de atributos

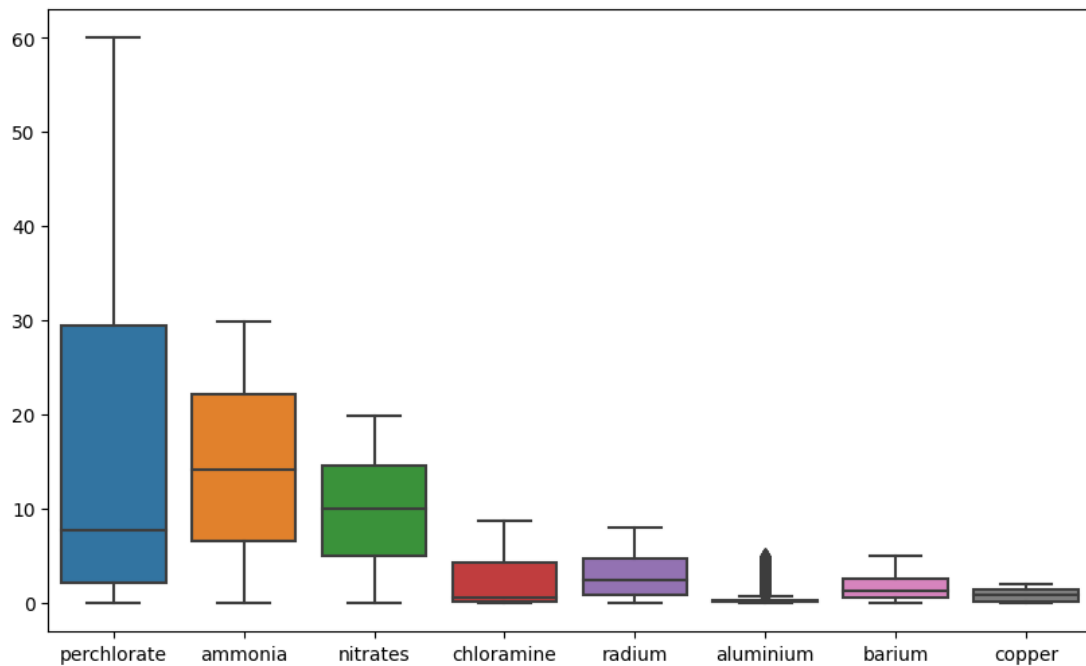
Para melhorar o conjunto de dados, deve-se eliminar atributos que não interferem com a classificação final da amostra. Assim, tornar a análise do *dataset* mais eficiente no sentido de capacidade computacional necessária para processar os dados, além de eliminar o risco de estudar atributos que se mostram desnecessários.

### 2.2.1 Boxplot

A utilização do gráfico de caixas (*boxplot*) tem como objetivo facilitar a visualização de amostras com dados *outliers*. Estes são, de acordo com [Hoppen e Prates \(2017\)](#), dados que se diferenciam drasticamente de todos os outros, com valores que fogem da normalidade e que podem causar anomalias nos resultados obtidos por meio de algoritmos e sistemas de análise.

A Figura 4 ordena as colunas de maneira onde os 8 atributos com as maiores dispersões são apresentados. Portanto, como um dos atributos possui uma variância muito

Figura 4 – Boxplot para visualização de outliers



Fonte: os autores

grande com outlier, preferimos utilizar o z-score para realizar a normalização ao invés da normalização min-max.

### 2.2.2 Heatmap

Com o uso do método Heatmap<sup>2</sup>, é possível analisar a correlação dos atributos visualmente, através de um gráfico que apresenta o nível de correlação vinculado com uma escala de cores, onde a cor mais escura apresenta maior correlação. Assim, a diagonal principal da tabela dispensa análise, uma vez que apresenta correlação de um atributo com ele mesmo (sempre igual 100%).

Este gráfico pode ser observado na Figura 5.

### 2.2.3 Normalização de dados

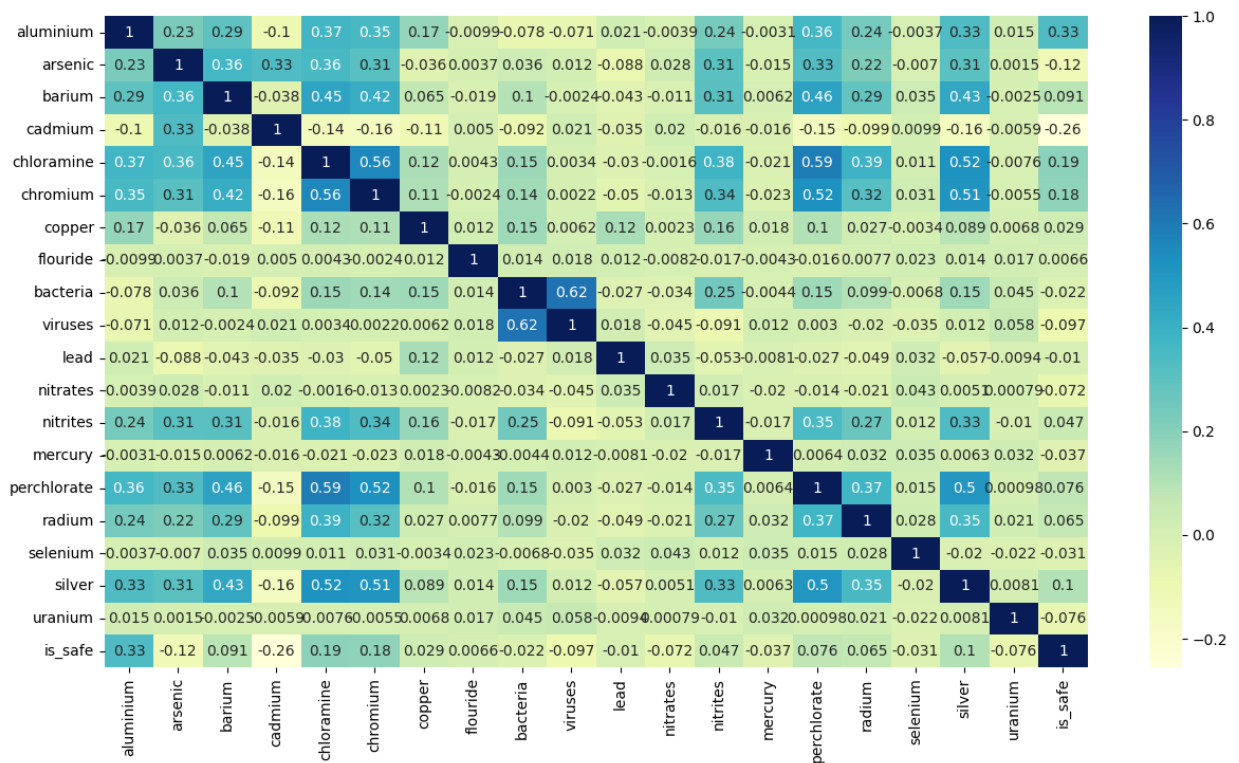
A normalização de dados coloca os dados dentro de um intervalo controlado, normalmente entre 0 e 1, ou -1 e 1 caso exista valores negativos. Entretanto, isso acontece sem distorcer as diferenças nas faixas de valores. Ou seja, sem retirar *outliers*.

Pode ser essencial fazer normalização em alguns algoritmos específicos, onde é necessário que os dados estejam na mesma escala, como KNN (*K-Nearest Neighbours*<sup>3</sup>),

<sup>2</sup> Mapa de calor

<sup>3</sup> K-Vizinhos Mais Próximos

Figura 5 – Heatmap de correlação dos atributos



Fonte: os autores

Redes Neurais, Regressão Linear, entre outros.

Porém, também existem alguns algoritmos que não obrigatoriamente necessitam dos dados na mesma escala, como é o caso das Árvores de Decisão, Random Forest ou Naïve Bayes.

## 2.3 Métodos de classificação

Existem diversos métodos para a classificação de resultados assim como diferentes métricas para definição de qual método é mais adequado para cada caso em específico, como acurácia, precisão, entre outras. Portanto, é parte essencial da pesquisa descobrir qual método melhor se adapta ao conjunto de dados selecionados para estudo.

Durante o processo de treinamento, utilizamos dois *datasets* inicialmente iguais, mas com diferentes processamentos, o primeiro com as seguintes características:

- Sem normalização
- Sem realização de amostragem

- Sem redução de dimensões (Fusão ou seleção)
- Tuplas colocadas aleatoriamente dentro do dataset

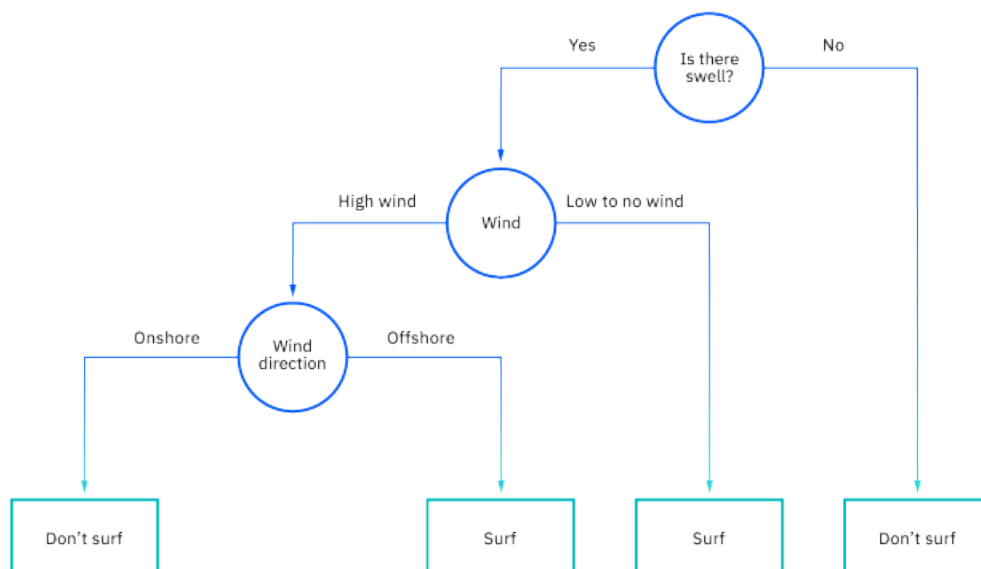
e o segundo da seguinte forma:

- Normalização com z-score (Para que outliers não influenciem radicalmente nos dados).
- Separação de amostragem de 1000 tuplas aleatórias com class is\_safe sendo 0. (Feito para diminuir o desbalanceamento entre as classes)
- Seleção de 15 atributos através de Árvore de Decisão, para separar os que mais influenciam no resultado.
- Tuplas distribuídas de forma aleatória dentro do dataset.

### 2.3.1 Árvore de Decisão

Árvore de Decisão, ou *decision tree*, é um algoritmo de aprendizado supervisionado não-paramétrico, ou seja, não faz nenhuma suposição sobre os dados. Pode ser usado tanto para realizar trabalhos de classificação, como também de regressão. Possui uma arquitetura hierárquica, em formato de árvore, com nó-raiz, galhos, nós internos e nós-folhas, como exemplificado pela Figura 6.

Figura 6 – Exemplo de Árvore de Decisão - Surf or Don't surf



Fonte: [IBM \(2023\)](#)

Um dos principais desafios quando se trata de Árvore de Decisão, é a escolha de quais atributos possuirão maior relevância para classificação. Entretanto, utilizando a biblioteca *sklearn*, é possível executar o algoritmo sem maiores esforços.

Para o primeiro conjunto de dados, foram elaborados os seguintes resultados com o uso de Árvore de Decisão:

Figura 7 – Resultados da aplicação da Árvore de Decisão

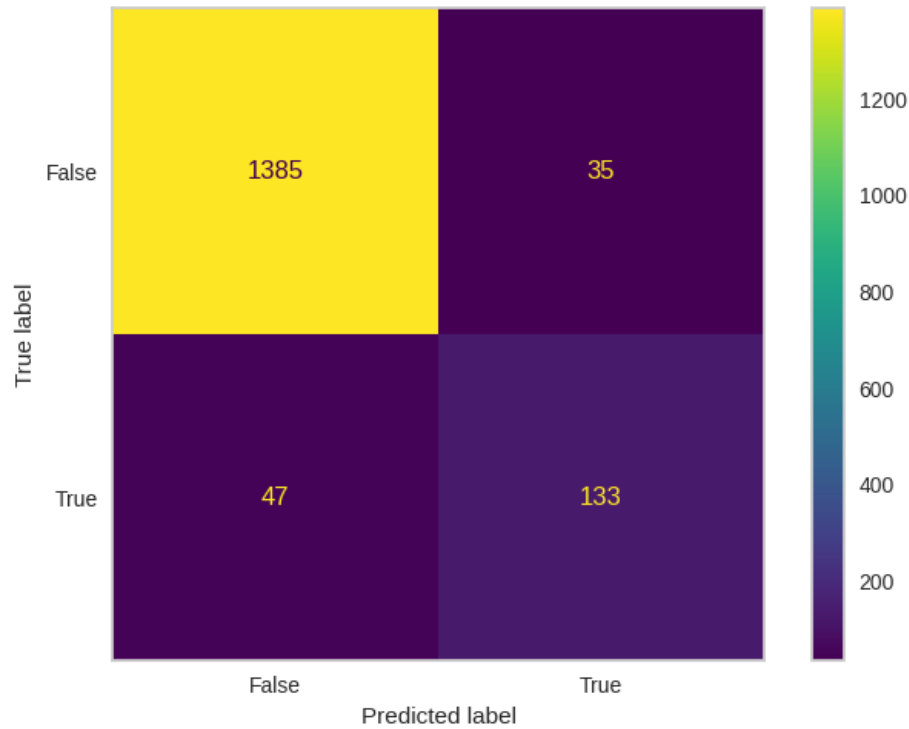


Tabela 2 – Valores obtidos a partir da Figura 7

	precision	recall	f1_score	support
<b>0</b>	0.97	0.98	0.97	1420
<b>1</b>	0.79	0.74	0.76	180
<b>macro avg</b>	0.88	0.86	0.87	1600
<b>weighted avg</b>	0.95	0.95	0.95	1600
<b>accuracy</b>	0.95			1600

Fonte: os autores

Pode-se observar que o recall para a classe `is_safe = 1` é de 0.74, bem menor do que o recall da classe `is_safe = 0`. Isso se dá pois como há muito mais tuplas da classe 0, o nosso modelo tende a classificar tuplas como sendo da classe `is_safe = 0`, fazendo com que a quantidade de falsos negativos aumente, diminuindo o recall.

E para o segundo conjunto, agora com os dados normalizados, foram obtidos os seguintes resultados:

Figura 8 – Resultados da aplicação da Árvore de Decisão (dados normalizados)

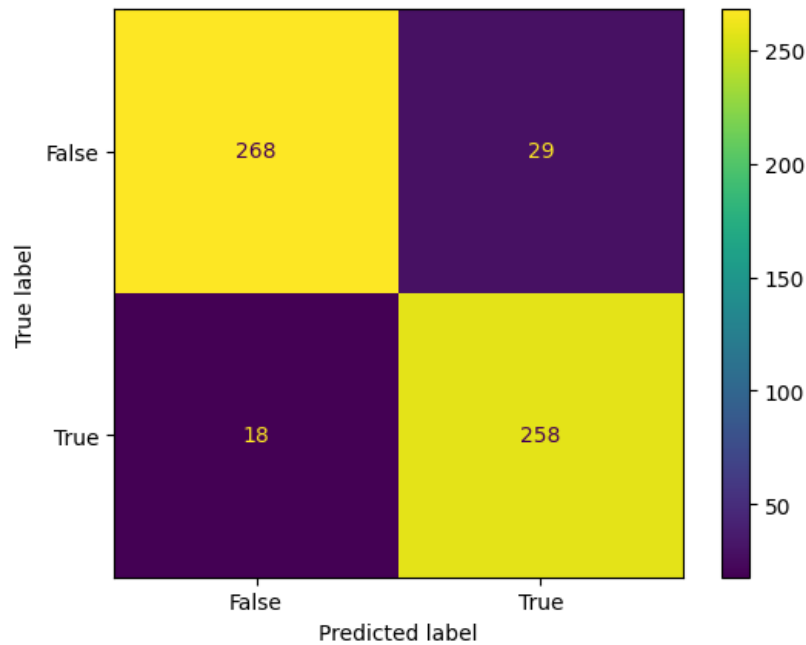


Tabela 3 – Valores obtidos a partir da Figura 8

	precision	recall	f1_score	support
<b>0</b>	0.94	0.90	0.92	297
<b>1</b>	0.90	0.93	0.92	276
<b>macro avg</b>	0.92	0.92	0.92	573
<b>weighted avg</b>	0.92	0.92	0.92	573
<b>accuracy</b>	0.92			573

Fonte: os autores

### 2.3.1.1 Árvore de Decisão com Cross-Validation

A fim de explorar mais técnicas de aprendizado de máquina, também será considerada a ideia de *cross-validation*<sup>4</sup>. Na validação cruzada k-fold, os dados de entrada são divididos em subconjuntos de dados (*folds*)  $k$  para serem analisados.

Nesse método, um modelo é treinado em todos os conjuntos, menos em um ( $k-1$ ). Em seguida, esse modelo é avaliado no conjunto de dados que não foi usado para treinamento. Esse processo é repetido  $k$  vezes, com um subconjunto diferente reservado para avaliação (e excluído do treinamento) a cada vez.

Para o primeiro conjunto de dados, surgem os seguintes resultados:

<sup>4</sup> Validação cruzada

Figura 9 – Resultados da aplicação de Arvore de Decisão com Cross-Validation k-fold

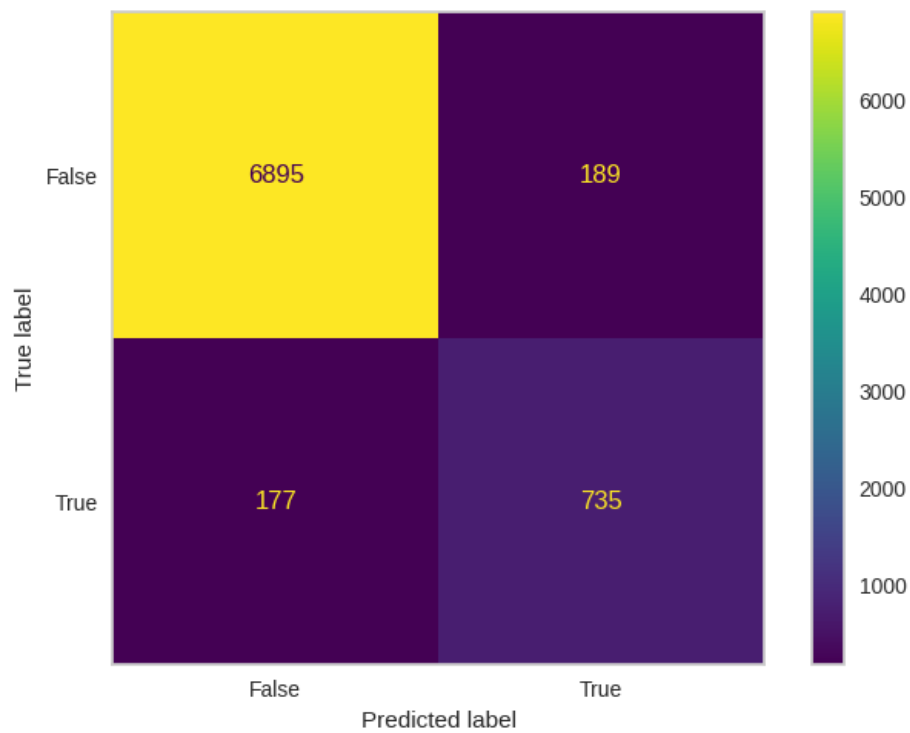


Tabela 4 – Valores obtidos a partir da Figura 9

	precision	recall	f1_score	support
<b>0</b>	0.97	0.97	0.97	7084
<b>1</b>	0.80	0.81	0.80	912
<b>macro avg</b>	0.89	0.89	0.89	7996
<b>weighted avg</b>	0.95	0.95	0.95	7996
<b>accuracy</b>	0.95			7996

Fonte: os autores

Assim, para o segundo conjunto, os resultados são os apresentados na Figura 10

Figura 10 – Resultados da aplicação de Arvore de Decisão com Cross-Validation k-fold (dados normalizados)

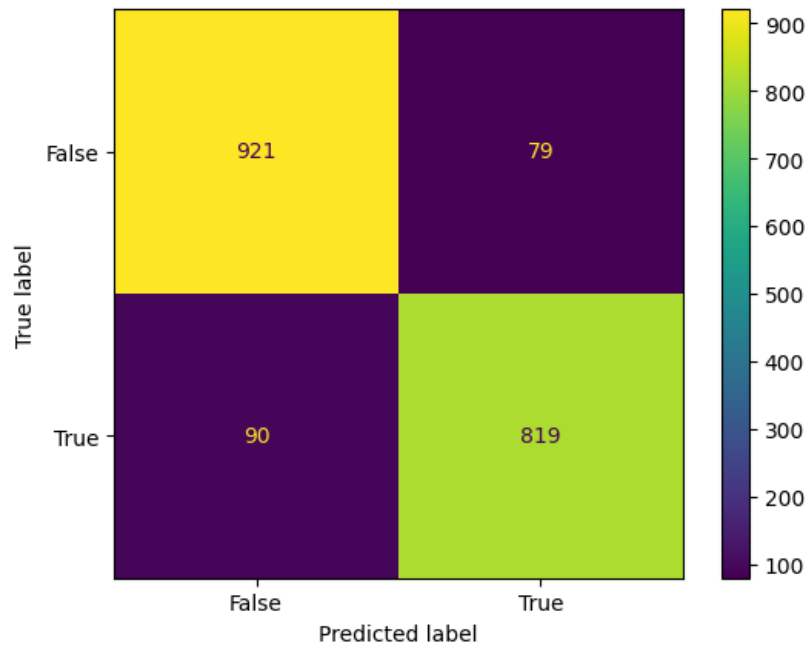


Tabela 5 – Valores obtidos a partir da Figura 10

	precision	recall	f1_score	support
<b>0</b>	0.91	0.92	0.92	1000
<b>1</b>	0.91	0.90	0.91	909
<b>macro avg</b>	0.91	0.91	0.91	1909
<b>weighted avg</b>	0.91	0.91	0.91	1909
<b>accuracy</b>	0.91			1909

Fonte: os autores

### 2.3.2 KNN

O KNN (K-Vizinhos Mais Próximos) é um dos primeiros estudados quando se trata de algoritmos de classificação. É considerado básico e essencial para o aprendizado de máquina, principalmente ao campo do aprendizado supervisionado e seu uso é intenso com o reconhecimento de padrões.

O algoritmo possui grande possibilidade de aplicação em cenários do dia a dia e, assim como a Árvore de Decisão, é não-paramétrico. Dessa forma, é necessário informar dados prévios (chamados de dados de treinamento), que já possuem uma classificação. Assim, o papel do algoritmo é assumir que coisas semelhantes existem nas proximidades. Em outras palavras, coisas semelhantes estão próximas umas das outras. (VAZ, 2021)

Assim, para o primeiro *dataset*, os resultados são como apresentado na Figura 11.



Figura 11 – Aplicação do método de classificação KNN

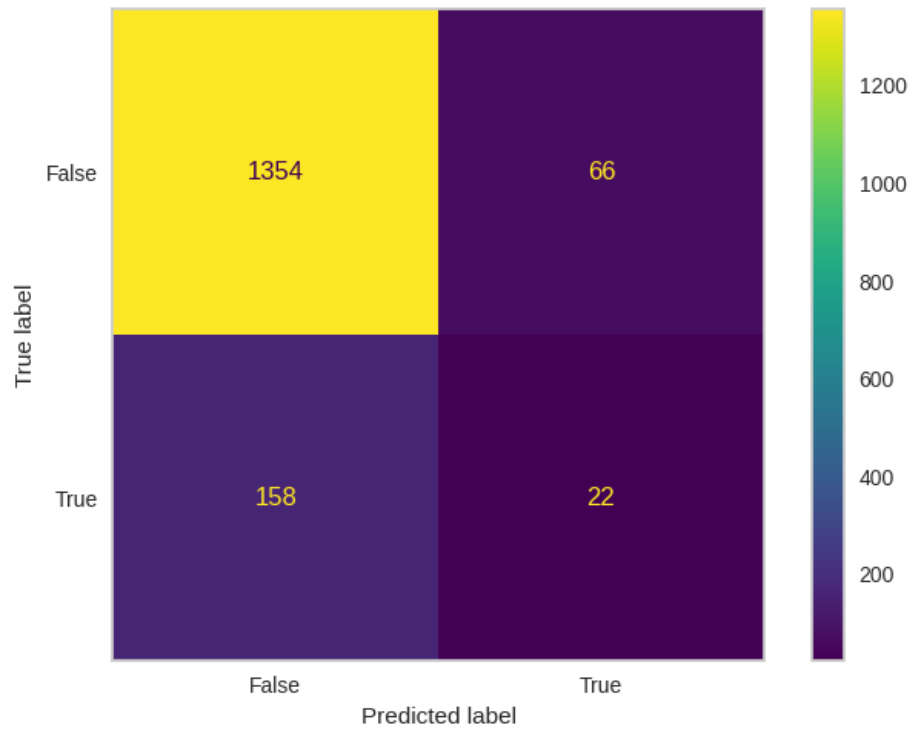


Tabela 6 – Valores obtidos a partir da Figura 11

	precision	recall	f1_score	support
<b>0</b>	0.90	0.95	0.92	1420
<b>1</b>	0.25	0.12	0.16	180
<b>macro avg</b>	0.57	0.54	0.54	1600
<b>weighted avg</b>	0.82	0.86	0.84	1600
<b>accuracy</b>	0.86			1600

Fonte: os autores

Para o segundo, por sua vez, os resultados são apresentados na Figura 12.

Figura 12 – Aplicação do método de classificação KNN (dados normalizados)

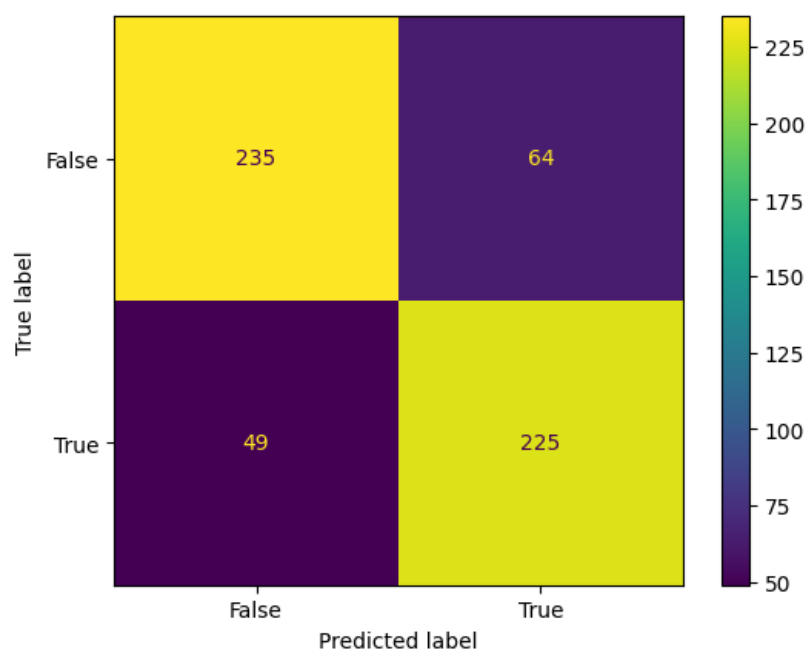


Tabela 7 – Valores obtidos a partir da Figura 12

	precision	recall	f1_score	support
<b>0</b>	0.83	0.79	0.81	299
<b>1</b>	0.78	0.82	0.80	274
<b>macro avg</b>	0.80	0.80	0.80	573
<b>weighted avg</b>	0.80	0.80	0.80	573
<b>accuracy</b>	0.80			573

Fonte: os autores

### 2.3.2.1 KNN com Cross-Validation

Assim como feito com o algoritmo de Árvore de Decisão, também é possível utilizar *cross-validation* k-fold combinado ao método KNN. Para o primeiro conjunto, os resultados são apresentados na Figura 13.

Figura 13 – Aplicação de KNN com Cross-Validation k-fold

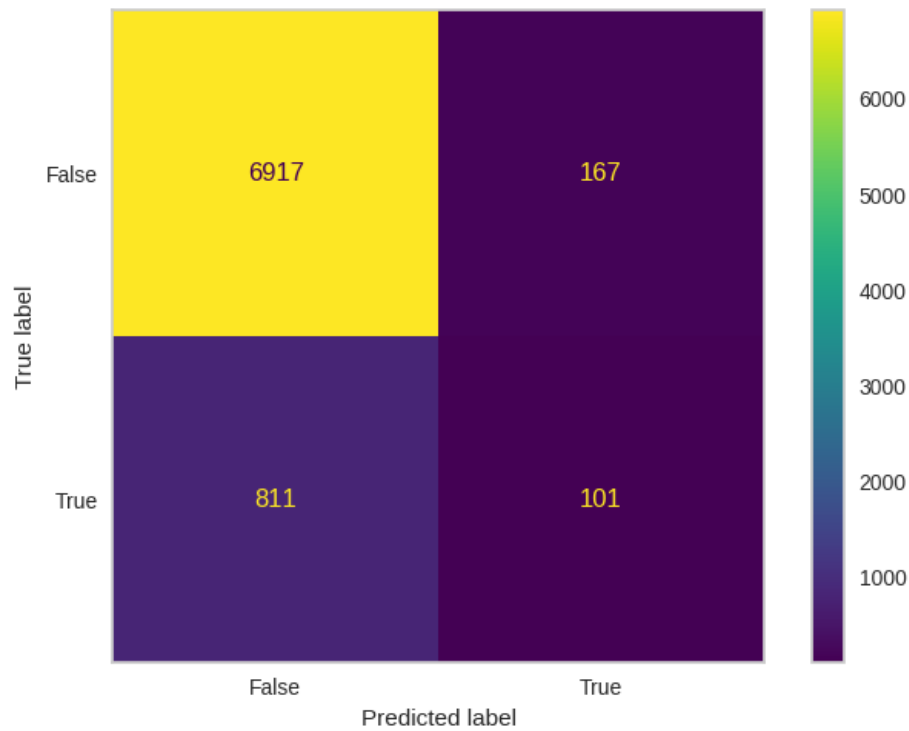


Tabela 8 – Valores obtidos a partir da Figura 13

	precision	recall	f1_score	support
<b>0</b>	0.90	0.98	0.93	7084
<b>1</b>	0.38	0.11	0.17	912
<b>macro avg</b>	0.64	0.54	0.55	7996
<b>weighted avg</b>	0.84	0.88	0.85	7996
<b>accuracy</b>	0.88			7996

Fonte: os autores

Pode-se observar que, os valores de precision, recall e f1\_score são muito baixos para a class is\_safe e isso se deve a grande quantidade de tuplas com is\_safe = 0. Com isso, pode acontecer de que o fold pego para treinar tenha apenas classes de 1 tipo, nesse caso, com maior chance da classe is\_safe = 0 aparecer mais.

Para o segundo conjunto de dados, os resultados são exibidos na Figura 14.

Figura 14 – Aplicação de KNN com Cross-Validation k-fold (dados normalizados)

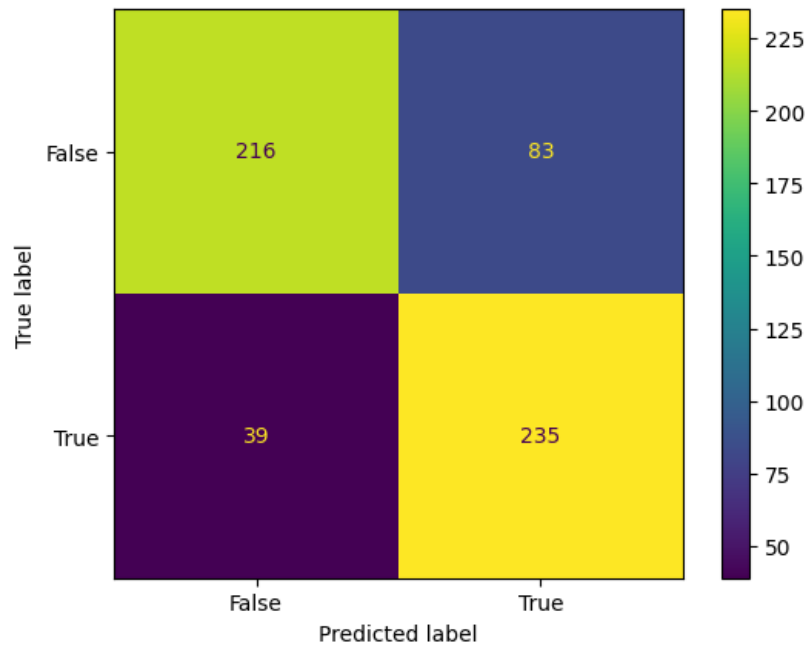


Tabela 9 – Valores obtidos a partir da Figura 14

	precision	recall	f1_score	support
<b>0</b>	0.85	0.72	0.78	299
<b>1</b>	0.74	0.86	0.79	274
<b>macro avg</b>	0.79	0.79	0.79	573
<b>weighted avg</b>	0.80	0.79	0.79	573
<b>accuracy</b>	0.79			573

Fonte: os autores

Observa-se também nesse caso, que a normalização, organização aleatória das tuplas e seleção de atributos mais importantes faz com que as eficiências sejam melhores do que o dataset não processado.

### 2.3.3 Rede Neural

Utilizamos a rede neural com 3 camadas, cada uma com 8 neurônios e, além disso, com um total máximo de 500 iterações.

Com relação ao primeiro conjunto de dados, os resultados apresentados foram os seguintes:

Figura 15 – Aplicação de MLP

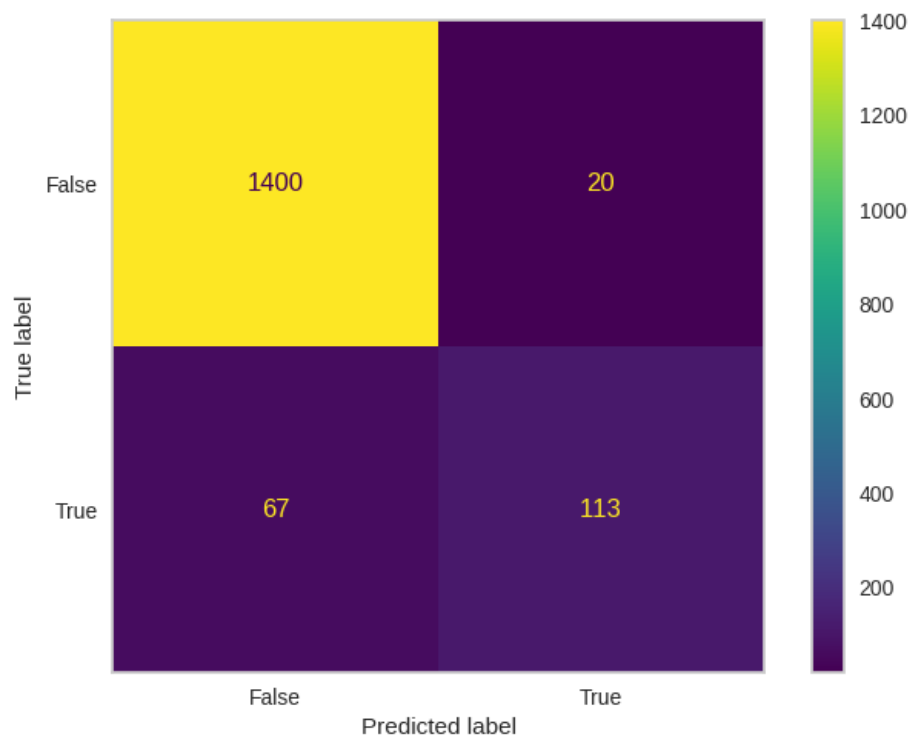


Tabela 10 – Valores obtidos a partir da Figura 15

	precision	recall	f1_score	support
<b>0</b>	0.90	0.98	0.93	7084
<b>1</b>	0.38	0.11	0.17	912
<b>macro avg</b>	0.64	0.54	0.55	7996
<b>weighted avg</b>	0.84	0.88	0.85	7996
<b>accuracy</b>	0.88			7996

Fonte: os autores

Já em relação ao segundo conjunto de dados, foram apresentados:

Figura 16 – Aplicação de MLP (dados normalizados)

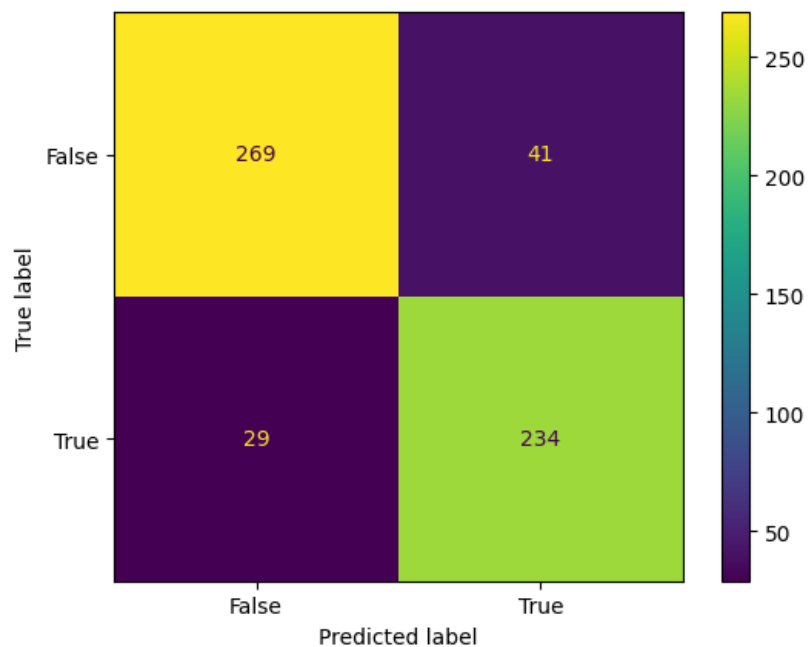


Tabela 11 – Valores obtidos a partir da Figura 16

	precision	recall	f1_score	support
<b>0</b>	0.90	0.87	0.88	310
<b>1</b>	0.85	0.89	0.87	263
<b>macro avg</b>	0.88	0.88	0.88	573
<b>weighted avg</b>	0.88	0.88	0.88	573
<b>accuracy</b>	0.88			573

Fonte: os autores

### 2.3.3.1 Rede Neural com Cross-Validation

No primeiro *dataset*, foram obtidos os resultados expostos na Figura 17

Figura 17 – Aplicação de MLP com Cross-Validation

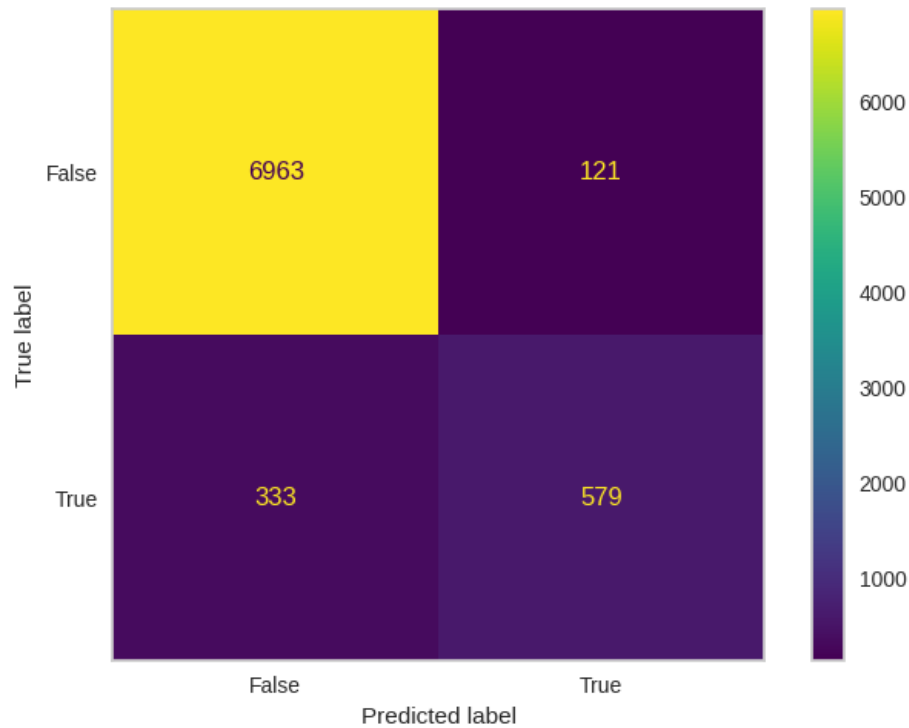


Tabela 12 – Valores obtidos a partir da Figura 17

	precision	recall	f1_score	support
<b>0</b>	0.90	0.98	0.93	7084
<b>1</b>	0.38	0.11	0.17	912
<b>macro avg</b>	0.64	0.54	0.55	7996
<b>weighted avg</b>	0.84	0.88	0.85	7996
<b>accuracy</b>	0.88			7996

Fonte: os autores

Já em relação ao segundo *dataset*, os resultados foram os seguintes:

Figura 18 – Aplicação de MLP com Cross-Validation (dados normalizados)

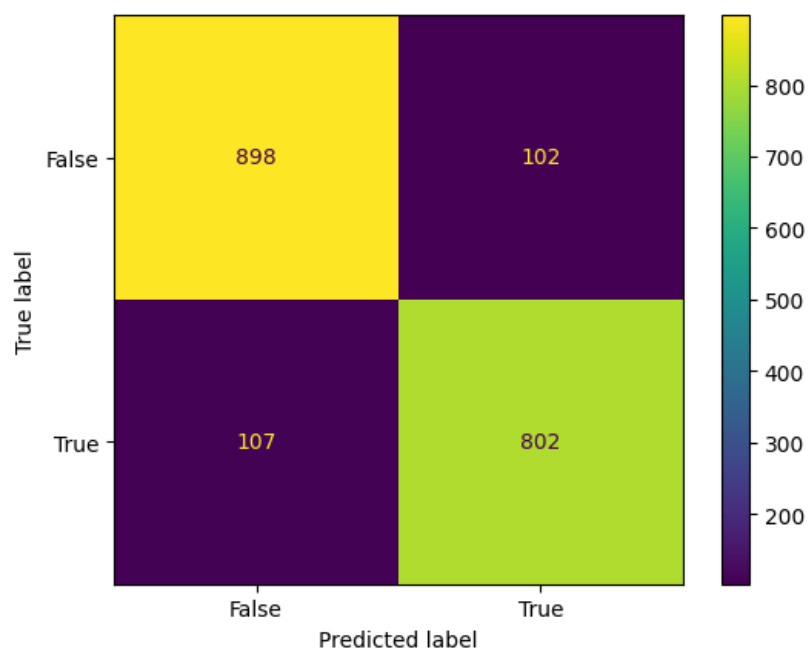


Tabela 13 – Valores obtidos a partir da Figura 18

	precision	recall	f1_score	support
<b>0</b>	0.89	0.90	0.90	1000
<b>1</b>	0.89	0.88	0.88	909
<b>macro avg</b>	0.89	0.89	0.89	1909
<b>weighted avg</b>	0.89	0.89	0.89	1909
<b>accuracy</b>	0.89			1909

Fonte: os autores



### 3 CONCLUSÃO

Assim, com o uso de diversos os algoritmos e métodos para pré-processamento e análise dos dados, podemos compará-los para definir quais obtiveram os melhores resultados.

Tabela 14 – Tabela comparativa de acurácia e f1\_score dos algoritmos executados

<b>algoritmo</b>	<b>acurácia</b>	<b>f1_score</b>
Árvore de Decisão	0.95	0.87
Árvore de Decisão (norm.)	0.92	0.92
Árvore de Decisão com Validação Cruzada	0.95	0.89
Árvore de Decisão com Validação Cruzada (dados norm.)	0.91	0.91
KNN	0.86	0.54
KNN (dados norm.)	0.80	0.80
KNN com Validação Cruzada	0.88	0.55
KNN com Validação Cruzada (dados norm.)	0.79	0.79
Rede Neural	0.88	0.55
Rede Neural (dados norm.)	0.88	0.88
Rede Neural com Validação Cruzada	0.88	0.55
Rede Neural com Validação Cruzada (dados norm.)	0.89	0.89

Dado os resultados mostrados anteriormente, podemos dizer que a normalização e processamentos dos dados realmente importaram para que os resultados sejam mais corretos com uma grande variedade de informação. Com isso, a normalização z-score, a seleção de atributos de acordo com o modelo Decision Tree e amostragem da classe `is_safe = 0` fizeram com que o f1\_score tivesse um aumento considerável para todos os modelos (KNN, Decision Tree e Neural Network).

Pode-se perceber também que para todos os datasets mais balanceados, o resultado do f1\_score foi maior do que para datasets não balanceados, uma vez que, como o modelo foi treinado com quantidades similares de ambas as classes, não fica enviesado quando comparado com datasets desbalanceados.

Além disso, observa-se uma consequência de datasets desbalanceados nos valores da acurácia, onde os mesmos são bem altos. Tal fato é resultado da grande quantidade de tuplas de classe 0, fazendo com que o modelo seja enviesado (Ou seja, tende a classificar as tuplas de teste como sendo da classe `is_safe = 0`). Por conta disso, mesmo classificando todas as tuplas como sendo da classe `is_safe = 0`, teremos um valor alto para a acurácia.

# REFERÊNCIAS

HOPPEN, J.; PRATES, W. Outliers, o que são e como tratá-los em uma análise de dados? *Aquarela*, set. 2017. Disponível em: <<https://www.aquare.la/o-que-sao-outliers-e-como-trata-los-em-uma-analise-de-dados/>>.

IBM. What is a decision tree? *IBM*, 2023. Disponível em: <<https://www.ibm.com/topics/decision-trees>>.

VAZ, A. L. Knn —k-nearest neighbor, o que é? *DataHackers, Medium*, mar. 2021. Disponível em: <<https://medium.com/data-hackers/knn-k-nearest-neighbor-o-que-é-ál-aebe0f833eb>>.