



**FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA  
ROCHA”**

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM

## **Relatório Científico com Análise do Acidente do Titanic**

João Victor Fontes Ribeiro  
Luiz Andre Junqueira Mendes  
Luiz César Benedito  
Marcos Antonio Fernandes Junior  
Michel Ikeda Francisco  
Paulo Sérgio Bonfim Cristanello

Projeto sobre aprendizado de máquina onde o modelo preveja quais passageiros sobreviveram ao naufrágio do RMS Titanic, Fundação de Ensino “Eurípedes Soares da Rocha, submetido ao professor Fagner Christian Paes.

## Sumário

1.	Introdução.....	3
2.	Objetivo.....	3
3.	Obtenção dos Dados.....	4
3.1	Definição da Base de Dados.....	4
3.2	Carregando os arquivos CSV.....	4
3.3	Importando as bibliotecas necessárias.....	4
4.	Exploração dos Dados.....	5
4.1	Verificando quantidade de variáveis (Colunas) e o tamanho (Linhas) do conjunto treinamento (train.csv).....	5
4.2	Exibir o tipo de cada feature.....	5
4.3	Exibir as 5 primeiras entradas do conjunto de dados.....	6
4.4	Exibir a distribuição estatística dos dados.....	7
4.5	Exibir histograma das features numéricas.....	8
4.5.1	Qual a porcentagem dos passageiros sobreviventes?.....	8
4.5.2	Qual a faixa etária dos passageiros do Titanic?.....	9
4.5.3	As Crianças, sobreviveram mais do que os Adultos??.....	10
5.	Preparação do dados.....	11
5.1	Juntar os conjuntos de dados (Treino e Teste).....	11
5.2	Selecionar as features.....	11
5.3	Exibir os valores faltantes nos datasets (Treino e Teste).....	12
5.4	Completar os valores faltantes.....	12
5.5	Preparar features para o modelo de Machine Learning.....	13
5.6	Recuperar os conjuntos de dados (Treino e Teste).....	13
6.	Modelagem & Avaliação.....	14
6.1	Importar biblioteca do modelo de Machine Learning (Regressão Logística).....	14
6.2	Avaliação do modelo utilizando a Regressão Logística.....	14
6.3	Importar biblioteca do modelo de Machine Learning (Árvore de Decisão).....	14
6.4	Avaliação do modelo utilizando Árvore de Decisão.....	15
7.	Conclusão.....	15

## **1. Introdução**

O projeto consiste em analisar os dados sobre o acidente do RMS Titanic, construindo um estudo para prever a probabilidade de quais passageiros sobreviveriam ao naufrágio e o que influenciou na morte ou não dos mesmos.

Palavras chave: Regressão Logística, Arvore de decisão, Titanic, Machine Learning.

## **2. Objetivo**

O RMS Titanic teve sua construção iniciada em março de 1909, levando 2 anos para ficar pronto, foi lançado ao mar em maio de 1911 e afundou no Oceano Atlântico em 15 de abril de 1912, quando colidiu contra um iceberg, o número de mortos é incerto, pois alguns passageiros desistiram de embarcar de última hora e outros não pagaram. O navio partiu de Southampton com direção a Nova York, com mais de 1500 pessoas a bordo. O Titanic foi feito para ser considerado o navio mais luxuoso e seguro daquela época.

Embora aqueles que escaparam com vida tiveram sua boa dose de sorte, alguns grupos de pessoas eram mais propensos a escaparem da morte do que outros. Por exemplo, mulheres, crianças e passageiros da 1ª Classe. Assim, nota-se que existe algum padrão que pode ser extraído dos dados brutos.

Com isso, o objetivo deste trabalho é aplicar os conhecimentos adquiridos em sala de aula e apresentar um modelo para análise dos dados referente ao acidente. Utilizando métodos e testes estatísticos para prever a probabilidade de o passageiro ter sobrevivido (1) ou não (0), analisando os resultados e escolhendo um bom modelo que seja satisfatório.

### 3. Obtenção dos Dados

#### 3.1 Definição da Base de Dados.

Os dados utilizados foram retirados do site: [www.kaggle.com/c/titanic/data](http://www.kaggle.com/c/titanic/data). Esse site hospeda diversos *datasets* para competições de *Data Science*. O *dataset* contém 12 informações referente os passageiros como idade, nome, sexo e classe de viagem, entre outras.

#### 3.2 Carregando os arquivos CSV.

Figura 1 - Carregando Arquivos CSV

```
[ ] dadostraincsv = pd.read_csv("train.csv")
    dadostestcsv = pd.read_csv("test.csv")
```

Fonte: Desenvolvido pelos autores.

- Conjunto de treinamento (train.csv)
- Conjunto de teste (test.csv)

#### 3.3 Importando as bibliotecas necessárias.

O comando *import* é utilizado para realizar a importação das bibliotecas que contém comandos uteis para o desenvolvimento do projeto.

Figura 2 - Importação das bibliotecas

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
```

Fonte: Desenvolvido pelos autores.

- O pacote *numpy* possui funções de álgebra linear.
- O pacote *pandas* possui funções de *data processing*, CSV file I/O.
- O pacote *matplotlib* possui funções para criação de gráficos e visualizações de dados.
- O pacote *seaborn* possui funções para análise de dados estatísticos.

## 4. Exploração dos Dados

### 4.1 Verificando quantidade de variáveis (Colunas) e o tamanho (Linhas) do conjunto treinamento (train.csv).

Figura 3 - Verificando conjunto de treinamento (train.csv)

```
[ ] print("Variáveis:\t{}\nEntradas:\t{}".format(dadostraincsv.shape[1], dadostraincsv.shape[0]))
```

Variáveis: 12  
Entradas: 891

Fonte: Desenvolvido pelos autores.

A Figura 3, mostra que temos 12 variáveis com um total de 891 linhas de informações referente os passageiros no dataset de treinamento.

### 4.2 Exibir o tipo de cada feature.

Figura 4 - Exibição do tipo das variáveis

```
[ ] display(dadostraincsv.dtypes)
```

PassengerId int64  
Survived int64  
Pclass int64  
Name object  
Sex object  
Age float64  
SibSp int64  
Parch int64  
Ticket object  
Fare float64  
Cabin object  
Embarked object  
dtype: object

Fonte: Desenvolvido pelos autores.

Podemos verificar na Figura 4, que temos 3 tipos diferentes de features no conjunto de dados:

- 5 do tipo int64 - Números inteiros;
- 5 do tipo object - Objetos
- 2 do tipo float64 - Números de ponto flutuante.

E as features (atributos) dos passageiros:

- PassengerId: Número de identificação do passageiro
- Survived: Informa se o passageiro sobreviveu ao desastre
- Pclass: Classe em que estava viajando
- Name: Nome
- Sex: Sexo
- Age: Idade
- SibSp: Quantidade de cônjuges e irmãos a bordo
- Parch: Quantidade de pais e filhos a bordo
- Ticket: Número do bilhete
- Fare: Preço do bilhete
- Cabin: Número da cabine do passageiro
- Embarked: Porto no qual o passageiro embarcou

#### 4.3 Exibir as 5 primeiras entradas do conjunto de dados.

Figura 5 - Exibindo as 5 primeiras linhas dos dados

```
[ ] dados.traincsv.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Fonte: Desenvolvido pelos autores.

Na Figura 5, temos as 5 primeiras linhas do nosso conjunto de dados. Para exibir uma quantidade maior de entradas, deve-se adicionar o valor como parâmetro da função *head()*. Por exemplo: *dadostrainscsv.head(50)*

#### 4.4 Exibir a distribuição estatística dos dados

Utilizamos o método *describe()* para ter uma visão estatística dos dados e dos valores faltantes.

Figura 6 - Visualizando estatísticas dos dados

```
[ ] dadostrainscsv.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Fonte: Desenvolvido pelos autores.

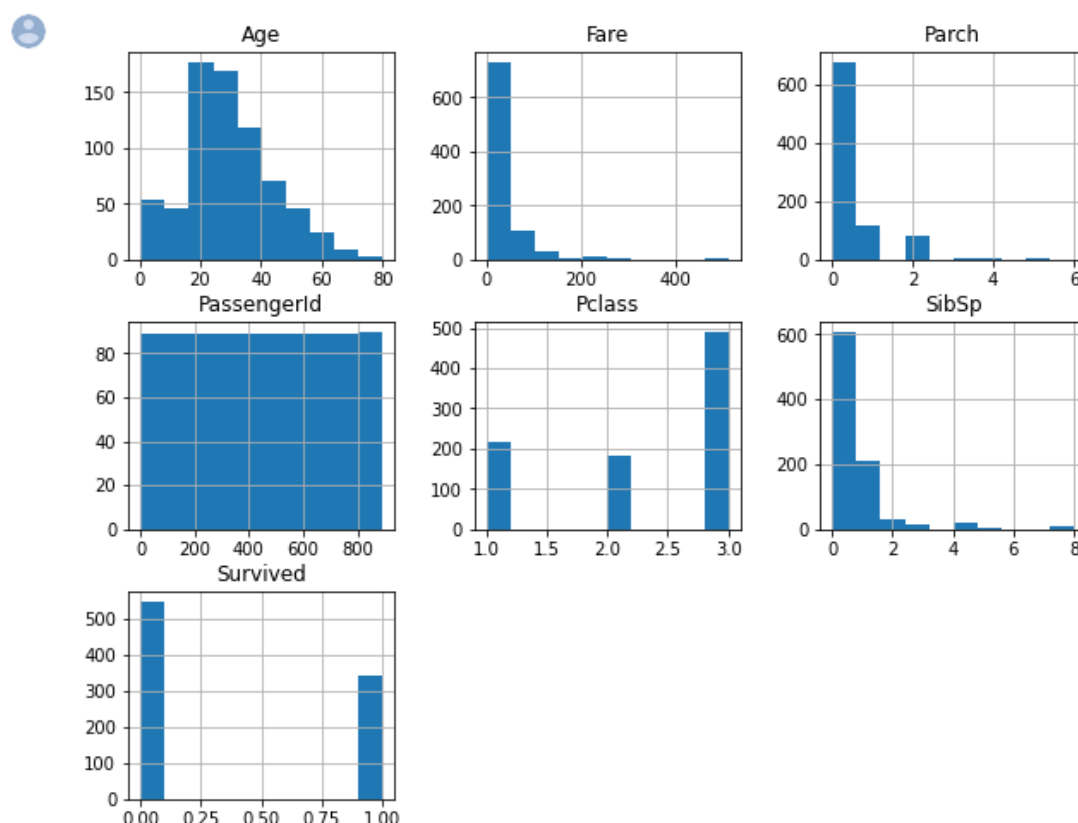
Acima na Figura 6 podemos identificar numa análise superficial, que algumas idades estão faltando, que a porcentagem de sobreviventes foi de aproximadamente 38% e que as idades dos passageiros variam de 0,4 a 80.

## 4.5 Exibir histograma das features numéricas

Na Figura 7 logo abaixo, geramos gráficos das features numéricas, para auxiliar na análise de possíveis valores faltantes ou inconsistentes.

Figura 7 – Histogramas das features numéricas

```
[ ] dadostrainscv.hist(figsize=(10,8));
```



Fonte: Desenvolvido pelos autores.

### 4.5.1 Qual a porcentagem dos passageiros sobreviventes?

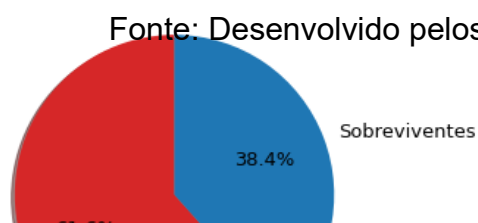
Figura 8 - Quantidade de Sobreviventes e Não Sobreviventes

```
[ ] dadostrainscv.idade = dadostrainscv.conv()
```

Figura 9 - Gráfico com porcentagem de sobreviventes e não sobreviventes

```
[ ] dadostrainscv['Survived'].value_counts().plot.pie(colors=('tab:red', 'tab:blue'),
    title='Qual a porcentagem dos passageiros sobreviventes?',
    fontsize=13, shadow=True, startangle=90, autopct='%1.1f%%',
    labels=('Não sobreviventes', 'Sobreviventes'),
    figsize=(5,5)).set_ylabel('')
```

Text(0, 0.5, '')  
Qual a porcentagem dos passageiros sobreviventes?



Fonte: Desenvolvido pelos autores.



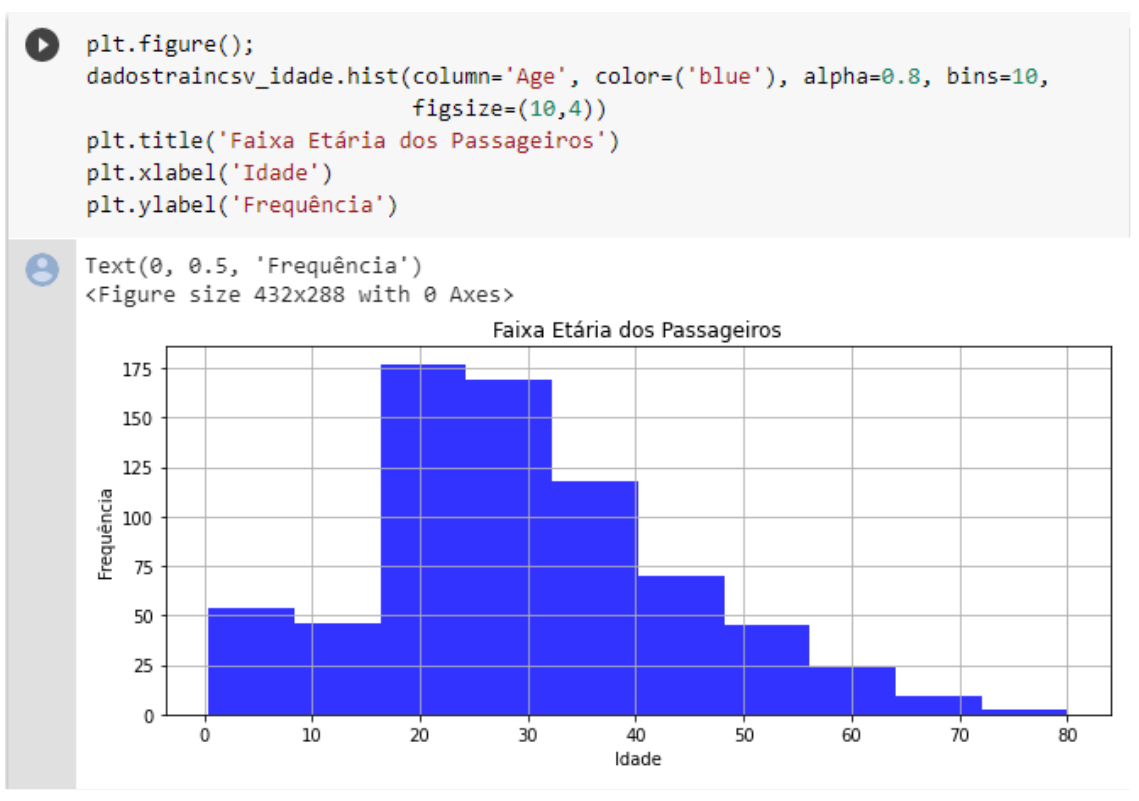
Do total de 891 passageiros, 342 sobreviveram e 549 não sobreviveram como mostrado na Figura 8.

Para melhor entender, na Figura 9 temos um gráfico com a porcentagem de sobreviventes (38.4%) e não sobreviventes (61.6%)

#### 4.5.2 Qual a faixa etária dos passageiros do Titanic?

Como citado anteriormente, podemos confirmar pelo gráfico da Figura 10, que a idade dos passageiros está entre 0.4 anos a 80, com uma quantidade maior de pessoas na faixa dos 20 – 30 anos de idade.

Figura 10 - Gráfico com faixa etária dos passageiros



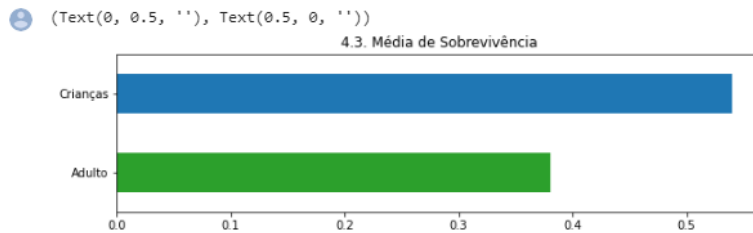
Fonte: Desenvolvido pelos autores.

### 4.5.3 As Crianças, sobreviveram mais do que os Adultos??

Figura 11 – Gráfico comparativo entre sobrevivência de adultos e crianças

```
[ ] dadostrainsv_idade['Crianca_Adulto'] = 0
    dadostrainsv_idade.loc[dadostrainsv_idade['Age'] < 18].index, 'Crianca_Adulto'] = 'Crianças'
    dadostrainsv_idade.loc[dadostrainsv_idade['Age'] >= 18].index, 'Crianca_Adulto'] = 'Adulto'
    dadostrainsv_idade.groupby('Crianca_Adulto')[['Survived']].mean()

    dadostrainsv_idade.groupby('Crianca_Adulto')['Survived'].mean().plot.barh(title='4.3. Média de Sobrevivência',
                                     figsize=(10,2.5),
                                     color=('tab:green','tab:blue')).set_ylabel(''),
    plt.xlabel('')
```



Fonte: Desenvolvido pelos autores.

Para analisar se as crianças sobreviveram mais que os adultos, definimos “crianças” todos os passageiro com menos de 18 anos. Como mostra o gráfico, as crianças tem uma taxa de sobrevivência maior que a dos adultos.

## 5. Preparação do dados

### 5.1 Juntar os conjuntos de dados (Treino e Teste)

Figura 12 – Código para Concatenar treino e teste em um único DataFrame

```
[ ] dadostrainsv_idx = dadostrainsv.shape[0]
    dadostestcsv_idx = dadostestcsv.shape[0]

    passengerId = dadostestcsv['PassengerId']

    target = dadostrainsv.Survived.copy()
    dadostrainsv.drop(['Survived'], axis=1, inplace=True)

    df_merged = pd.concat(objs=[dadostrainsv, dadostestcsv], axis=0).reset_index(drop=True)

    print("df_merged.shape: ({} x {})".format(df_merged.shape[0], df_merged.shape[1]))

df_merged.shape: (1309 x 12)
```

Fonte: Desenvolvido pelos autores.

## 5.2 Selecionar as features

Como qualquer conjunto de dados, pode existir a necessidade de retirar algumas features por falta de informações ou por não serem úteis.

Optamos por retirar as features: *PassengerId*, *Name*, *Cabin*, *Ticket* e *Not Survived*, por não conter informações que irão influenciar no projeto.

Figura 13 – Linha de comando onde são retirados algumas features

```
df_merged.drop(['PassengerId', 'Name', 'Cabin', 'Ticket', 'Not Survived'],  
               axis=1, inplace=True)
```

Fonte: Desenvolvido pelos autores.

## 5.3 Exibir os valores faltantes nos datasets (Treino e Teste)

Figura 14 – Verificação de valores faltantes

```
[ ] df_merged.isnull().sum()
```

Pclass	0
Sex	0
Age	263
SibSp	0
Parch	0
Fare	1
Embarked	2
dtype: int64	

Fonte: Desenvolvido pelos autores.

## 5.4 Completar os valores faltantes

Depois de verificado os dados faltantes, iremos imputar essas informações substituindo os valores nulos.

Para o preenchimento da idade (age) e preço do bilhete (fare), optamos por calcular o valor da mediana para ter uma precisão melhor, já no local de embarque (embarked) adicionamos com o valor de maior frequência.

Figura 15 – Preenchimento de valores faltantes

```
[ ] # age
    age_median = df_merged['Age'].median()
    df_merged['Age'].fillna(age_median, inplace=True)

    # fare
    fare_median = df_merged['Fare'].median()
    df_merged['Fare'].fillna(fare_median, inplace=True)

    # embarked
    embarked_top = df_merged['Embarked'].value_counts()[0]
    df_merged['Embarked'].fillna(embarked_top, inplace=True)
```

Fonte: Desenvolvido pelos autores.

## 5.5 Preparar features para o modelo de Machine Learning

Convertemos a feature embarked para *dummies*, depois utilizamos o comando *get\_dummies* que realiza a análise dos dados contidos e criar o número apropriado de features, como resultado obtivemos: *Embarked\_C*, *Embarked\_Q* e *Embarked\_S*.

A feature sex será utilizada de forma binária, sendo (1) mulher e (0) homem.

Figura 16 – Preparação das features

```
[ ] df_merged['Sex'] = df_merged['Sex'].map({'male': 0, 'female': 1})

embarked_dummies = pd.get_dummies(df_merged['Embarked'], prefix='Embarked')
df_merged = pd.concat([df_merged, embarked_dummies], axis=1)
df_merged.drop('Embarked', axis=1, inplace=True)

display(df_merged.head())
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked_914	Embarked_C	Embarked_Q	Embarked_S
0	3	0	22.0	1	0	7.2500	0	0	0	1
1	1	1	38.0	1	0	71.2833	0	1	0	0
2	3	1	26.0	0	0	7.9250	0	0	0	1
3	1	1	35.0	1	0	53.1000	0	0	0	1
4	3	0	35.0	0	0	8.0500	0	0	0	1

Fonte: Desenvolvido pelos autores.

## 5.6 Recuperar os conjuntos de dados (Treino e Teste)

Figura 17 – Linha de comando para recuperação dos datasets

```
[ ] dadostraincsv = df_merged.iloc[:dadostraincsv_idx]
    dadostestcsv = df_merged.iloc[dadostraincsv_idx:]
```

Fonte: Desenvolvido pelos autores.

## 6. Modelagem & Avaliação

### 6.1 Importar biblioteca do modelo de Machine Learning (Regressão Logística)

Figura 18 – Importação da biblioteca de Regressão Logística

```
[ ] from sklearn.linear_model import LogisticRegression
```

Fonte: Desenvolvido pelos autores.


### 6.2 Avaliação do modelo utilizando a Regressão Logística

Utilizando a regressão logística, obtivemos uma acurácia de 80.13%

Figura 19 - Avaliação do modelo

```
[ ] lr_model = LogisticRegression(solver='liblinear')
    lr_model.fit(dadostraincsv, target)

    acc_logReg = round(lr_model.score(dadostraincsv, target) * 100, 2)
    print("Acurácia do modelo de Regressão Logística: {}".format(acc_logReg))
```

 Acurácia do modelo de Regressão Logística: 80.13

Fonte: Desenvolvido pelos autores.

### 6.3 Importar biblioteca do modelo de Machine Learning (Árvore de Decisão)

Figura 20 – Importação da biblioteca de Árvore de Decisão

```
[ ] from sklearn.tree import DecisionTreeClassifier
```

Fonte: Desenvolvido pelos autores.

### 6.4 Avaliação do modelo utilizando Árvore de Decisão

Utilizando o modelo de árvore de decisão, obtivemos uma acurácia de 82.72%

Figura 21 - Avaliação do modelo

```
[ ] tree_model = DecisionTreeClassifier(max_depth=3)
    tree_model.fit(dadostraincsv, target)

    acc_tree = round(tree_model.score(dadostraincsv, target) * 100, 2)
    print("Acurácia do modelo de Árvore de Decisão: {}".format(acc_tree))
```

 Acurácia do modelo de Árvore de Decisão: 82.72

Fonte: Desenvolvido pelos autores.

## 7. Conclusão

Podemos concluir que o projeto alcançou o seu objetivo.

Todo o projeto ocorreu como o esperado, esse Desafio do dataset Titanic nos deu uma excelente oportunidade de testar na prática, os conhecimentos adquiridos em sala de aula. Suas amostras são confirmados pela própria história do ocorrido, e sua acurácia se mostrou na média entre diversos projetos avaliados na competição da Kaggle. Evidentemente, para que se obtenha um modelo com uma acurácia maior, aconselha-se analisar melhor os dados e realizar mais testes com outras bibliotecas de Machine Learning. Para este

projeto, o algoritmo de Árvore de Decisão foi o mais eficaz, retornando um bom resultado. Para projetos posteriores, será essencial o estudo mais aprofundado da exploração e preparação dos dados, assim como de outras bibliotecas de Machine Learning, a fim conseguir melhores resultados do que os obtidos neste projeto.