UNIVEM - CENTRO UNIVERSITÁRIO “EURÍPIDES SOARES DA ROCHA” DE MARÍLIA

ISABELLE RIBEIRO ORNELAS COELHO LIMA

# TITANIC: MACHINE LEARNING FROM DISASTER

Projeto final solicitado pelo Prof. Fagner Christian Paes, para a disciplina de Qualificação I, do curso de Ciência de Dados.

Marília
2020

## SUMÁRIO

## 1 OBJETIVO

Este projeto tem como objetivo reproduzir o desafio de *Data Science* (Ciência de Dados) do *Titanic: Machine Learning from Disas*ter, uma das competições de *Machine Learning* mais conhecidas dos cientistas de dados, promovida pelo site Kaggle. No desafio é preciso prever quais passageiros sobreviveram ao naufrágio do navio Titanic que afundou no meio do Atlântico em 15/04/1912, matando 1502 dos 2224 passageiros.

Para a realização do projeto foi utilizado o *dataset* (o conjunto de dados) disponibilizado no site do Kaggle, contendo um *dataset* para treinamento ("train.csv") com 891 dados de passageiros, incluindo a informação se sobreviveram ou não, e um outro dataset para teste ("test.csv") com o objetivo de prever o destino de 418 passageiros, cujo a resposta se sobreviveram ou não é incerta.

## 2 INTRODUÇÃO

Este projeto foi proposto como trabalho final para a disciplina de Qualificação I, ministrada pelo Prof. Fagner Christian Paes, do curso de Ciência de Dados, do UNIVEM - Centro Universitário “Eurípides Soares da Rocha”, campus de Marília/SP.

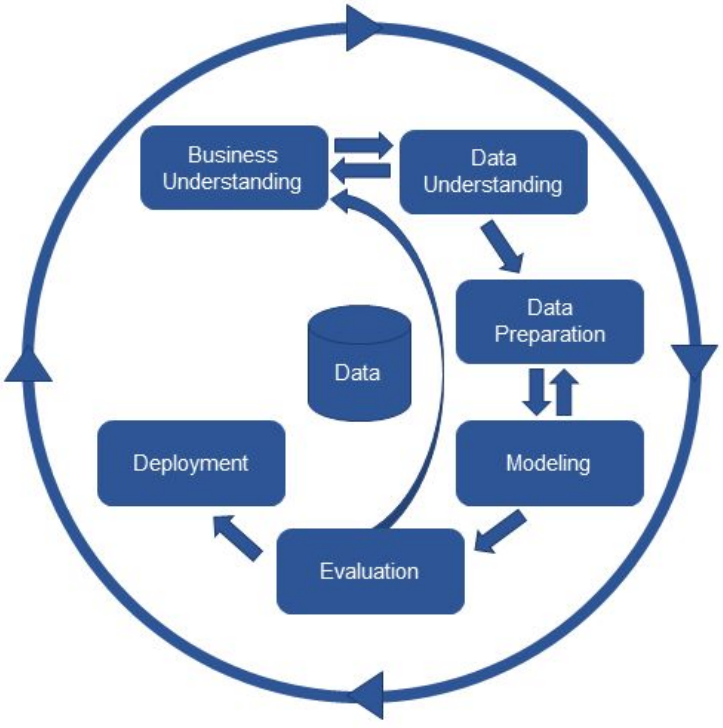
O trabalho possui a finalidade de, a partir do desafio reproduzido, obter um melhor entendimento da área de Ciência de Dados e como a mesma funciona. Colocando em prática os temas visto na disciplina Qualificação I e também vários outros temas abordados nas disciplinas de Algoritmo e Programação, Estatística, Infraestrutura de Tecnologia da Informação e por fim Business Strategy & Data Science, ministradas no primeiro semestre do curso.

A Ciência de Dados é uma área de estudo interdisciplinar e intensivamente computacional, que surgiu com o intuito de “lidar com o grande volume e variedade de dados tanto estruturados, quanto não-estruturados, produzidos diariamente, de modo a subsidiar melhores decisões estratégicas” (CURTY; CERVANTES, 2016, p. 1). Assim, a Ciência de Dados nada mais é que o uso de diversas tecnologias, modelos e metodologias para capturar, armazenar e processar informações, gerando valor a um negócio.

O Kaggle é uma das maiores plataformas de competições de Machine Learning do mundo, permitindo que os usuários encontrem e publiquem conjuntos de dados, explorem e construam modelos em um ambiente de ciência de dados baseado na Web, trabalhem com outros cientistas de dados e engenheiros de aprendizado de máquina e participem de competições para solucionar desafios de ciência de dados.

A metodologia utilizada no projeto foi a CRISP-DM (*Cross Industry Standard Process for Data Minin*g). Segundo o site Semantix (2016) o CRISP-DM é uma "metodologia que fornece uma abordagem estruturada para processos de mineração de dados, sendo amplamente utilizada devido à sua poderosa praticidade, flexibilidade e utilidade ao usar a análise para resolver problemas comerciais complexos". Esse método define o ciclo de vida do projeto em 6 fases:

Figura 1 - Fases CRISP-DM




Fonte: https://medium.com/matgonz/crisp-dm-na-pr%C3%A1tica-65be0ee92ada

Pela definição do autor Gonzalez (2019) o *Business Understanding*, primeira fase, consiste em identificar as reais necessidades do projeto. A fase *Data Understanding* coleta, organiza e documenta todos os dados que se encontram

disponíveis para realizar a análise exploratória. A fase *Data Preparation* deve tratar os dados para certificar que as informações estão de acordo com o esperado. Consistência de erros e ausência de valores deverão ser resolvidos para que se possa selecionar amostras aleatórias e utilizá-las para treino e testes. A fase *Modeling* ocorre a definição de quais modelos testar.

Já a fase *Evaluation* avalia os resultados e levanta todas as possibilidades de variações que os dados possam ter, analisando se existe algum fator que tenha sido negligenciado e até que ponto o modelo atende aos objetivos do negócio. Caso o modelo não esteja performando como esperado, devemos retornar a primeira etapa para entender o negócio e os dados. E por fim a fase *Deployment*, onde o modelo é enviado para produção e implementado em protótipo.

# 3 EXPERIMENTO

## 3.1 Business Understanding

Como falado anteriormente a reprodução do desafio foi divida em 6 fases começando pelo Business Understanding, que consistia na definição do problema do *dataset*. No site do kaggle você já encontra a definição do desafio e um pouco da história do naufrágio.

> O naufrágio do Titanic é um dos mais infames naufrágios da história. Em 15 de abril de 1912, durante sua viagem inaugural, o amplamente considerado "inafundável" RMS Titanic afundou após colidir com um iceberg. Infelizmente, não havia barcos salva-vidas suficientes para todos a bordo, resultando na morte de 1502 dos 2224 passageiros e tripulantes. Embora houvesse algum elemento de sorte envolvido na sobrevivência, parece que alguns grupos de pessoas eram mais propensos a sobreviver do que outros.
>
> Nesse desafio, pedimos que você construa um modelo preditivo que responda à pergunta: "que tipo de pessoas têm mais probabilidade de sobreviver?" usando dados de

passageiros (nome, idade, sexo, classe socioeconômica etc.) (KAGGLE, 2020).

### 3.2 Data Understanding

Através de uma análise exploratória dos dados, esta fase consistia em coletar, organizar e documentar todos os dados para entender os atributos e suas características. Para iniciar foi carregado as bibliotecas e o dataset, logo em seguido feito a análise das colunas existem nele.

**Carregar bibliotecas**

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
import os as os
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

**Carregar dataset**

In [2]:
```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

**Entender os atributos**

- **PassengerId:** Número de identificação do passageiro;
- **Survived:** Indica se o passageiro sobreviveu ao desastre. É atribuído o valor de 0 para aqueles que não sobreviveram, e 1 para quem sobreviveu;
- **Pclass:** Classe na qual o passageiro viajou. É informado 1 para primeira classe; 2 para segunda; e 3 para terceira;
- **Name:** Nome do passageiro;
- **Sex:** Sexo do passageiro;
- **Age:** Idade do passageiro em anos;
- **SibSp:** Quantidade de irmãos e cônjuges a bordo ;
- **Parch:** Quantidade de pais e filhos a bordo;
- **Ticket:** Número da passagem;
- **Fare:** Preço da passagem;
- **Cabin:** Número da cabine do passageiro;
- **Embarked:** Indica o porto no qual o passageiro embarcou. Há apenas três valores possíveis: Cherbourg, Queenstown e Southampton, indicados pelas letras “C”, “Q” e “S”, respectivamente

Além disso também foi verificado os tipos de variáveis, observado as amostra dos dados, descrito estatisticamente os dados, visualizado a distribuição dos dados em um histograma, analisado a probabilidade de sobrevivência pelo Sexo, e plotado os gráficos para Survived vs. Sex, Pclass e Embarked.

**Verificar tipos dos atributos**

In [3]:
```
print("Variáveis:\t{}\nEntradas:\t{}".format(train.shape[1], train.shape[0]))
```

```
Variáveis:      12
Entradas:       891
```

In [4]:
```
train.dtypes
```

Out[4]:
```
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
```

**Observar amostra dos dados**

In [5]:
```
train.head()
```

Out[5]:

| | 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 |

**Descrever estatisticamente os dados**

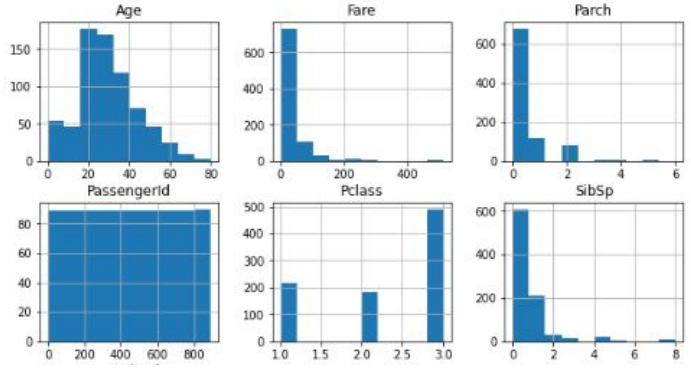
In [6]:
```
train.describe()
```

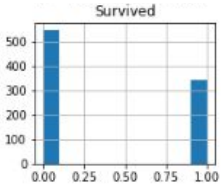
Out[6]:

| | 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 |

**Visualizar distribuição dos dados**

In [7]:
```
train.hist(figsize=(10,8));
```






**Analisar a probabilidade de sobrevivência pelo Sexo**

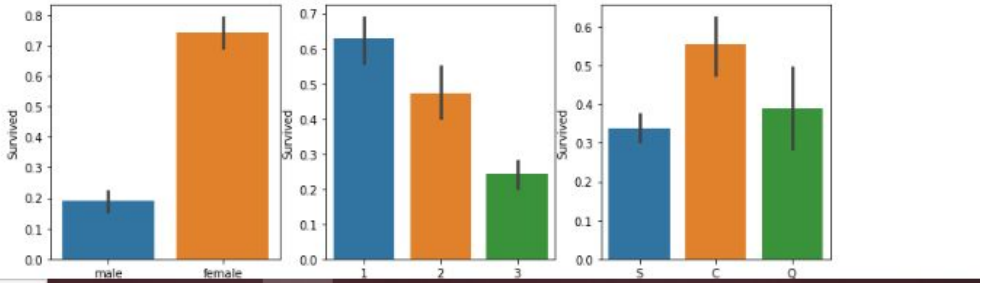
```
In [8]: train[['Sex', 'Survived']].groupby(['Sex']).mean()
```

Out[8]:

| | Survived |
|---|---|
| Sex | |
| female | 0.742038 |
| male | 0.188908 |

**Plotar os gráficos para Survived vs. Sex, Pclass e Embarked**

```
In [9]: fig, (axis1, axis2, axis3) = plt.subplots(1,3, figsize=(12,4))

sns.barplot(x='Sex', y='Survived', data=train, ax=axis1)
sns.barplot(x='Pclass', y='Survived', data=train, ax=axis2)
sns.barplot(x='Embarked', y='Survived', data=train, ax=axis3);
```




### 3.3 Data Preparation

A terceira fase houve o tratamento dos dados para certificar que as informações estão de acordo com o esperado. Começando pela junção dos conjuntos de dados de treino e teste, em seguida a retirada de dados irrelevantes, verificação da presença de valores nulos, preparação das variáveis para o modelo de Machine Learning e por fim recuperar os conjunto de dados de treino e teste.

**Juntar os conjuntos de dados de treino e teste**

Salvar os índices dos datasets para recuperação posterior

In [10]:
```
train_idx = train.shape[0]
test_idx = test.shape[0]
```

Salvar PassengerId para submissao ao Kaggle

In [11]:
```
passengerId = test['PassengerId']
```

Extrair coluna 'Survived' e excluir ela do dataset treino

In [12]:
```
target = train.Survived.copy()
train.drop(['Survived'], axis=1, inplace=True)
```

Concatenar treino e teste em um único DataFrame

In [13]:
```
df_merged = pd.concat(objs=[train, test], 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 11)
```

**Retirar dados irrelevantes**

In [14]:
```
df_merged.drop(['PassengerId', 'Name', 'Ticket', 'Cabin', 'Fare'], axis=1, inplace=True)

display(df_merged.head())
```

| | Pclass | Sex | Age | SibSp | Parch | Embarked |
|---|---|---|---|---|---|---|
| 0 | 3 | male | 22.0 | 1 | 0 | S |
| 1 | 1 | female | 38.0 | 1 | 0 | C |
| 2 | 3 | female | 26.0 | 0 | 0 | S |
| 3 | 1 | female | 35.0 | 1 | 0 | S |
| 4 | 3 | male | 35.0 | 0 | 0 | S |

**Verificar presença de valores nulos**

In [15]:
```
df_merged.isnull().sum()
```

Out[15]:
```
Pclass        0
Sex           0
Age         263
SibSp         0
Parch         0
Embarked      2
dtype: int64
```

**Tratar valores nulos**

São tratamentos comuns para valores nulos:

- Exclusão do atributo (caso ele seja nulo para grande parte das instâncias);
- Exclusão da instância (caso ela seja nula para grande parte dos atributos);
- Imputação por estatísticas simples, como média, mediana ou moda (podem ser calculadas para sub-amostras);
- Imputação por regressão e modelos preditivos.

Age: calcular o valor da mediana

In [16]:
```
age_median = df_merged['Age'].median()
df_merged['Age'].fillna(age_median, inplace=True)
```

Embarked: adicionar o valor com maior frequência

In [17]:
```
embarked_top = df_merged['Embarked'].value_counts()[0]
df_merged['Embarked'].fillna(embarked_top, inplace=True)
```

**Preparar as variáveis para o modelo de Machine Learnig**

Converter 'Sex' em 0 e 1

In [18]:
```
df_merged['Sex'] = df_merged['Sex'].map({'male': 0, 'female': 1})
```

Dummie variables para 'Embaked'

In [19]:
```
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 | Embarked_914 | Embarked_C | Embarked_Q | Embarked_S |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 3 | 0 | 22.0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 38.0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 3 | 1 | 26.0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 1 | 1 | 35.0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 4 | 3 | 0 | 35.0 | 0 | 0 | 0 | 0 | 0 | 1 |

Recuperar os conjunto de dados de treino e teste

In [20]:
```
train = df_merged.iloc[:train_idx]
test = df_merged.iloc[train_idx:]
```

### 3.4 Modeling e Evaluation

Criação de um modelo de Machine Learning utilizando os algoritmos de Regressão Logística e Árvore de Decisão.

O algoritmo de Regressão Logística

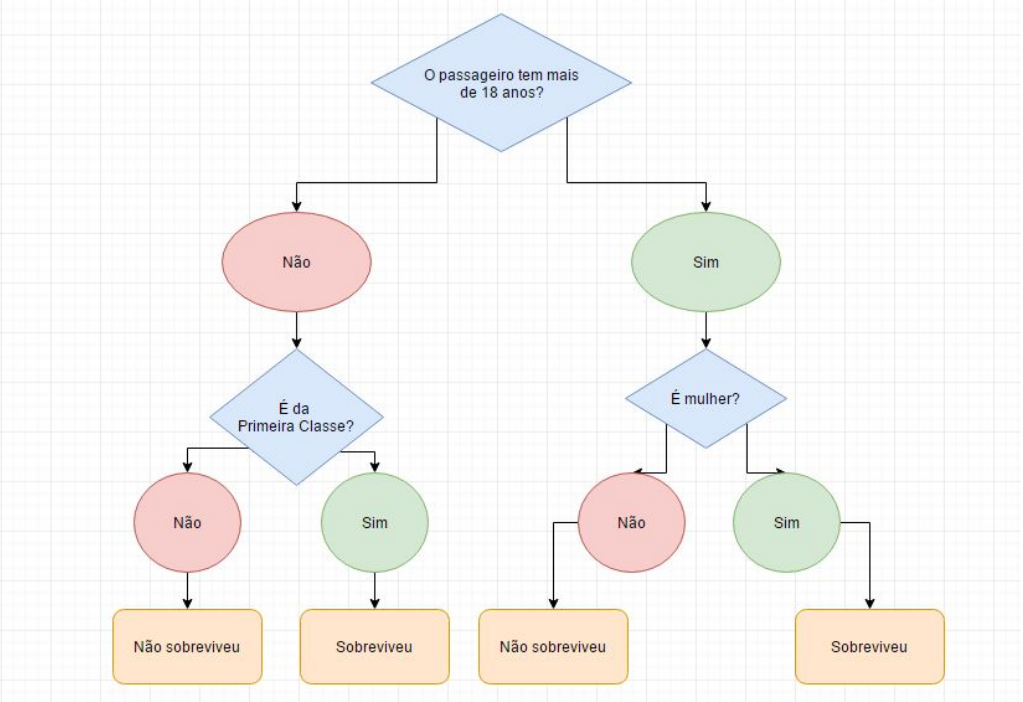
> trata-se de uma classificação, não de um algoritmo de regressão. Ela é usada para estimar valores discretos (valores binários como 0/1, sim/não, verdadeiro/falso) baseado em um grupo de variáveis independentes. Em palavras simples, ela prevê a probabilidade da ocorrência de um evento, ajustando os dados a uma função logística. Por isso, também é conhecida como regressão logística. Como prevê a probabilidade, seus valores de saída são algo esperado entre 0 e 1 (Vooo – Insights, 2006).

Já a Árvore de Decisão é

> um tipo de algoritmo de aprendizado supervisionado mais usado para problemas de classificação. Surpreendentemente, ele funciona tanto para variáveis dependentes categóricas quanto contínuas. Nestes algoritmos, nós dividimos a população em dois ou mais grupos homogêneos. Isso é feito

baseado nos atributos ou variáveis independentes mais significantes para tornar os grupos o mais distintos possível (Vooo – Insights, 2006).

Figura 2 - Estrutura de uma Árvore de Decisão




Fonte: https://paulovasconcellos.com.br/competicao-kaggle-titanic-tutorial-5b11993774f7

Toda a estrutura do fluxograma que você está vendo acima é chamado de “Árvore”, e toda árvore começa com uma pergunta inicial — no nosso exemplo, se o passageiro tem mais de 18 anos. Cada pergunta que você vê dentro dos losangos é chamado de nó; e o valor que ele retorna — por exemplo, “Sim” ou “Não” e “Não sobreviveu” ou “Sobreviveu” — é chamado de folha.

O algoritmo da árvore só termina quando todas as folhas forem puras, ou seja, quando todas as perguntas são respondidas. Existem casos onde nós podemos definir um grau de profundidade para a árvore, a fim de não deixá-la muito extenso e nosso modelo lento. Poderíamos, por exemplo, fazer nossa árvore responder até quatro perguntas antes de retornar o resultado.

Criar um modelo de Machine Learning de Regressão Logística

In [21]:
```
from sklearn.linear_model import LogisticRegression

lr_model = LogisticRegression(solver='liblinear')
lr_model.fit(train, target)

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

```
Acurácia do modelo de Regressão Logística: 80.02
```

Criar um modelo de Machine Learning de Árvore de Decisão

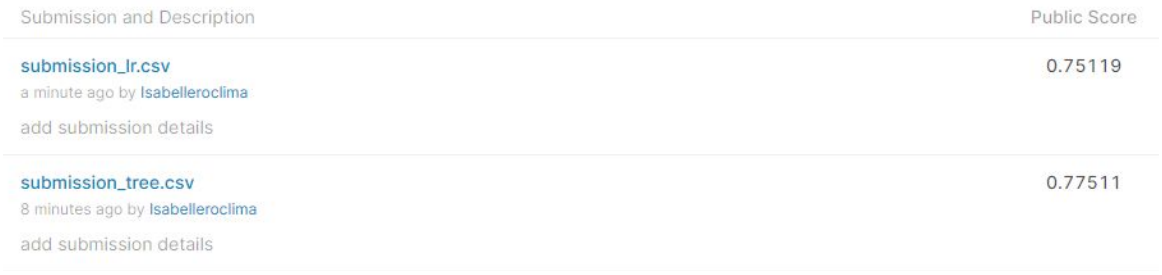
In [22]:
```
from sklearn.tree import DecisionTreeClassifier

tree_model = DecisionTreeClassifier(max_depth=3)
tree_model.fit(train, target)

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

```
Acurácia do modelo de Árvore de Decisão: 82.83
```

# 4 RESULTADOS OBTIDOS

## 4.1 Submetendo os resultados ao Kaggle

Para finalizar foi submetido o arquivo csv na página da competição no site Kaggle, onde informa a posição em que fiquei e o meu score. O resultado do jogo for inferior ao do treino, pois o modelo foi treinado 100% em cima do conjunto de treino (significa que ele tem um fit bem melhor ao dataset train). Pelo algoritmo de Regressão Logística o score foi de 0.75119 e pela Árvore de Decisão 0.77511.




## 4.2 Eu sobreviveria ao Titanic?

Depois que o modelo foi treinado e com boa acurácia, foi feito a análise se eu sobreviveria ao naufrágio do Titanic, indicando como passageira viajando na 1ª

classe, idade 24 anos, sem maridos e sem filhos e embarque no porto de Queenstown (Nova Zelândia). Sendo classificada como sobrevivente.

Árvore de Decisão

```
In [23]: isabelle_lima = np.array([1, 1, 24, 0, 1, 0, 0, 1, 0]).reshape((1, -1))
         print("Isabelle Lima:\t{}".format(tree_model.predict(isabelle_lima)[0]))
         Isabelle Lima:  1
```

Regressão Logística

```
In [24]: isabelle_lima = np.array([2, 1, 24, 0, 1, 0, 0, 1, 0]).reshape((1, -1))
         print("Isabelle Lima:\t{}".format(lr_model.predict(isabelle_lima)[0]))
         Isabelle Lima:  1
```

## 5 CONCLUSÃO

A metodologia CRISP-DM foi essencial para a exploração de dados deste dataset, pois, auxiliou no entendimento de como funciona um projeto desse tipo, gerando um conhecimento que com certeza irá contribuir para resolução de outros tipos de problemas em projetos futuros.

## BIBLIOGRAFIA


CURTY, Renata Gonçalves; CERVANTES, Brígida Maria Nogueira. Data science: ciência orientada a dados. **Informação & Informação**. Londrina, v. 21, n. 2, p. 1 – 3, maio/ago., 2016. Disponível em: https://brapci.inf.br/index.php/res/download/45442. Acesso em: 01 jun. 2020.

FUNDAMENTOS dos algoritmos de machine learning (com código python e r). Vooo – Insights. 2016. Disponível em: https://www.vooo.pro/insights/fundamentos-dos-algoritmos-de-machine-learning-com-codigo-python-e-r/. Acesso em: 01 jun. 2020.

GONZALEZ, Matheus. **CRISP-DM na prática**. Medium, 2019. Disponível em: https://medium.com/matgonz/crisp-dm-na-pr%C3%A1tica-65be0ee92ada. Acesso em: 01 jun. 2020.

JESUS, Márcio Ozório de.**Titanic**: análise de dados. GITHUB. 2017. Disponivel em: https://github.com/marcioozorio/titanic-investigando-a-base-de-dados/blob/master/Titanic_Final_V1.1.ipynb. Acesso em: 01 jun. 2020.

KAGGLE. **Titanic**: machine learning from disaster. Disponível em: https://www.kaggle.com/c/titanic. Acesso em: 01 jun. 2020.

MELO, Carlos. **Titanic**: machine learning from disaster. GITHUB. 2019. Disponível em: https://github.com/carlosfab/data_science/blob/master/Titanic.ipynb. Acesso em: 01 jun. 2020.

MELO, Carlos. **Data Science**: Investigando o naufrágio do Titanic. SIgmoidal, 2019. Disponível em: https://sigmoidal.ai/data-science-titanic-python-1/. Acesso em: 01 jun. 2020.

SEMANTIX. **Como explorar e gerenciar dados com o CRISP-DM**. Semantix, 2018. Disponível em: https://semantix.com.br/como-explorar-e-gerenciar-dados-com-o-crisp-dm/. Acesso em: 01 jun. 2020.

VASCONCELLOS, Paulo. **Como participar de um competição de Machine Learning no Kaggle**. 2017 Disponível em: https://paulovasconcellos.com.br/competicao-kaggle-titanic-tutorial-5b11993774f7. Acesso em: 01 jun. 2020.