

**PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO  
CIENTÍFICA E TECNOLÓGICA**

**UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY  
RIBEIRO**  
*Centro CCT*  
*Labotatório LCMAT*

*Relatório do período: Março 2021 - Janeiro 2022*

**Relatório Anual PIBIC-10**

**Bolsista:** Daniel Brito dos Santos  
**Matricula:** 00119110393  
**Orientadora:** Prof. Dra. Annabell Del Real Tamariz  
**Curso:** Bacharelado em Ciência da Computação

**Titulo do Projeto:** Project-driven *Data Science*: Aprendendo e Mapeando  
**Título do Plano de Trabalho:** Ponta do Iceberg: primeiros passos na Ciência de dados  
**Fonte financiadora:** PIBIC/UENF

# 1 Introdução

Em 1962, John Tukey publicou o artigo "The Future of Data Analysis" na revista científica The Annals of Mathematical Statistics [W.62]. Sendo o principal veículo para publicações de estatística matematicamente avançada, os outros artigos na revista apresentavam definições, teoremas e provas. Enquanto isso, o artigo de Tukey era basicamente uma confissão pública, explicando porquê ele achava esse tipo de pesquisa restritiva, possivelmente inútil e danosa. [Don17] Para ele, o escopo da pesquisa estatística deveria ser drasticamente ampliado e redirecionado. Nesse sentido, Tukey introduziu o termo "data analysis" para nomear o trabalho dos estatísticos práticos, diferenciando-o da inferência estatística formal. Assim, seu argumento central é de que a estatística é parte de uma entidade maior, entidade esta que não seria apenas um braço da matemática, mas sim uma nova ciência. Uma ciência definida pelo problema onipresente que ela busca resolver ao invés de se definir pelo seu objeto concreto [Don17].

Não é surpreendente que a resposta ao seu artigo tenha sido tímida, e hostil.[Don17] No próprio artigo, Tukey admite que extendeu o termo "data analysis" muito além de sua filologia, a tal ponto de na verdade englobar toda a estatística e ainda mais. Ele admite também que é um campo difícil, tanto a sua formalização quanto sua prática, e portanto deve se adaptar ao que as pessoas podem e precisam fazer com dados. Em suas palavras [W.62]:

Assim como biologia é mais complexa que física, ciência do comportamento é mais complexa que ambas, é provável que os problemas gerais de *data analysis* sejam mais complexos que os problemas de cada uma dessas três disciplinas. De modo que é pedir demais esperar uma orientação próxima e eficaz de uma estrutura altamente formalizada, agora ou em um futuro próximo.

Entretanto, o desafio que essa área apresenta é proporcional ao seu potencial disruptivo, como ficaria claro ao longo das décadas seguintes. [Don17] argumenta que "em última instância, o que tornou essa atividade chamada 'data analysis' concreta foi código, e não palavras". De modo que a partir do compartilhamento dos scripts computacionais para análise de dados, principalmente potencializado pelo que o autor chama de *Common Task Framework* (CTF), e todo o contexto de profusão de dados, crescimento do poder computacional e da mudança de paradigma que ela representou, o que atualmente chamamos de ciência de dados se consolidou como um dos principais campos de pesquisa e trabalho da atualidade. [Res]

As CTFs foram as competições abertas nas quais grupos diferentes poderiam submeter seus scripts para resolver um desafio de dados, de modo a premiar e implementar as melhores soluções. Esse fenômeno foi resultado direto da observação de Tukey sobre a mudança de paradigma de otimizar o valor gerado no mundo real ao invés da elegância matemática.

Esse foco na geração de valor torna essa nova ciência extremamente relevante no contexto das necessidades empresariais e sociais que temos: aumentar eficiência, entregar produtos melhores, criar soluções relevantes, analisar impacto de políticas, organizar a torrente continua de informação coletada o tempo inteiro.

Logo, esse enfoque trouxe muitos resultados, por exemplo o sucesso de algumas das empresas mais valiosas do mundo como a Amazon<sup>1</sup>, Google<sup>2</sup>, Facebook<sup>3</sup> e Netflix<sup>4</sup>, nas quais o seu produto é fruto direto de seu processamento de dados [SL20]. Tanto que nelas foi inaugurado o cargo "cientista de dados". Que logo foi denominada "a profissão mais sexy do século XXI" pela revista Harvard Business Review<sup>5</sup>.

Nesse contexto se consolidou a Ciência de Dados (DS) que pode ser definida como o campo de conhecimento fundamentalmente interdisciplinar; responsável por aplicar, organizar e expandir o conjunto de ferramentas, técnicas e conceitos necessários no processo de transformação de dados brutos em informações relevantes. Normalmente, utiliza-se modelos preditivos para modelar matematicamente relações entre variáveis de entrada de modo que permita inferir novos valores a partir de novas variáveis de entrada.

Entretanto, considerando a amplitude de sua "caixa de ferramentas", a necessidade de repertório para aplicá-las, a diversidade dos problemas abordados e a velocidade com que todo o campo se desenvolve, a formação de cientista de dados é um desafio inerente à própria natureza desse campo,[CFGT20, Don17].

---

<sup>1</sup>[www.amazon.com](http://www.amazon.com)

<sup>2</sup>[google.com](http://google.com)

<sup>3</sup>[facebook.com](http://facebook.com)

<sup>4</sup>[netflix.com](http://netflix.com)

<sup>5</sup>[hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century](http://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century)

Assim, nosso projeto se insere como um primeiro passo na tentativa de mapear essa nova ciência. Elencar os seus principais conceitos e ferramentas. Para tanto, revisamos a diversa literatura disponível tanto em livros quanto artigos e até mesmo sites interativos. Bem como executamos um projeto representativo para exemplificar e consolidar as principais técnicas e ferramentas utilizadas em cada etapa de um projeto de dados.

O projeto selecionado foi o Projeto Titanic, disponível na plataforma Kaggle<sup>6</sup>. Essa plataforma é um dos principais recursos para a prática e aprendizado da ciência de dados. Pois nela encontramos fóruns de discussão, minicursos, e principalmente competições de dados semelhantes aos CTFs mencionados por [Don17], onde é apresentado um problema e um conjunto de dados, de modo que cada usuário pode construir e submeter a sua solução, que será automaticamente avaliada e *rankeada* junto às outras soluções submetidas.

O grande impacto que essa estrutura oferece é a possibilidade de competição colaborativa, pois ao mesmo tempo que cada um desenvolve a sua rotina de análise, os notebooks podem ser publicados e discutidos, e dessa maneira muitos problemas inéditos foram resolvidos [BM21, Gra15, IMO17, NSR11, PRB14, TH14, YZT<sup>+</sup>18].

Tais soluções são construídas em notebooks, análogos aos Jupyter Notebooks, ou seja células interativas que podem conter código ou texto e que, segundo [CFG120], se tornaram o principal ambiente de desenvolvimento dos cientistas de dados. Os notebooks do Kaggle ainda oferecem processamento e armazenamento online gratuitos.

Finalmente, selecionamos o Titanic por ser a competição sugerida como introdução tanto à plataforma, quanto a projetos de dados.

## 2 Etapas propostas no plano de trabalho

O plano de trabalho original consistiu em desenvolver o Projeto Titanic de acordo com as etapas canônicas de um projeto de dados. Desse modo, em cada passo executamos o projeto, e utilizamos seu contexto para o estudo direcionado dos conceitos e ferramentas necessárias àquela etapa. Foram elas:

1. Bases gerais
2. Definição de problema
3. Obtenção de dados
4. Data Analysis
5. Aprendizado de Máquina ou Machine Learning
6. Data Visualization
7. Deployment
8. Elaboração do relatório final.

Executamos todas as etapas do projeto. Identificamos, entretanto, que as etapas de Aprendizado de Máquina, Data Visualization e Deployment trazem consigo uma complexidade maior em função de serem áreas autárquicas com seus próprios corpus de conhecimentos. O que inspira futuros desenvolvimentos para atingirmos o objetivo principal de cartografar o campo da ciência de dados.

## 3 Objetivos

Nosso objetivo nesse primeiro ano foi mapear e delinear os contornos do atual campo conhecido como Ciência de Dados. Explicitar a sua definição, seus principais métodos, conceitos e ferramentas. Bem como consolidar e exemplificar o aprendizado por meio da execução de um projeto representativo.

---

<sup>6</sup>[www.kaggle.com/c/titanic](http://www.kaggle.com/c/titanic)

## 4 Metodologia

Utilizamos uma metodologia diversificada para abordar os diferentes aspectos do projeto. De modo que o separamos em subprojetos de acordo com as etapas canônicas de um projeto de dados, principalmente inspiradas nos cinco passos definidos por [Ozd16]:

- Construir uma pergunta interessante
- Obter os dados
- Explorar os dados
- Modelar os dados
- Comunicar e visualizar os resultados

Assim, a cada etapa, buscamos o ferramental necessário e executamos os passos correspondentes no projeto Titanic.

Além dessa pesquisa teórica, o bolsista também se familiarizou com a plataforma Kaggle. Especialmente no que diz respeito ao projeto Titanic por meio do estudo da [página do projeto](#)<sup>7</sup>, tendo em vista compreende-lo para aplicar o que foi estudado teoricamente sobre a estruturação e execução de um projeto de dados.

### 4.1 Bases Gerais

O primeiro dos subprojetos foi o estudo da área em si. Iniciamos o processo de familiarização a partir da leitura dos livros [Ozd16] e [OS13]. Posteriormente, percebemos que há um importante debate quanto a definição de ciência de dados como um campo. Portanto, entendemos necessário buscar referencial teórico na literatura para definirmos Ciência de Dados, de modo a delimitarmos o escopo de sua prática e teoria, modelo de trabalho e pressupostos.

Nesse sentido pesquisamos o termo *"Data Science"* na plataforma de busca *"Google Scholar"*<sup>8</sup>, filtramos apenas os artigos do tipo revisão. Percorremos as primeiras 100 páginas de resultados avaliando manualmente seus títulos e resumos (*abstracts*). Assim, selecionamos os artigos que têm por objeto a caracterização rigorosa da Ciência de Dados, seu campo, teoria e prática. Uma vez selecionados, os artigos foram lidos sistematicamente de modo a construir a partir dos mesmos um panorama dessa área.

### 4.2 Definição de problemas de dados

Nessa etapa voltamos aos livros [Klo19], [Ozd16] e principalmente ao modelo de trabalho definido por [CFG19], encontrado como resultado da pesquisa na etapa anterior (seção 4.1). De modo a identificar e compreender de forma mais ampla as ferramentas e preocupações que cientistas de dados devem ter ao iniciar um projeto.

### 4.3 Obtenção de Dados

Iniciamos essa etapa estudando a obtenção de dados, usando o livro [Klo19], e seguindo as diretrizes do modelo [CFG19]. Nesse contexto, também buscamos elencar os principais recursos para aprendizagem da linguagem SQL [CB74]. Isto porque bancos de dados relacionais são uma das mais importantes fontes de dados para um cientista de dados, portanto dominar a sua língua franca é essencial para um profissional dos dados [Gru19, Klo19]. Não utilizamos SQL no Titanic devido ao fato de ser um projeto introdutório no qual os dados já foram fornecidos no formato de uma tabela simples e organizada, portanto não foi necessário utilizarmos bancos de dados.

Nesse sentido, para obtermos os dados para o projeto Titanic utilizamos as instruções do Kaggle bem como a documentação oficial da biblioteca Pandas<sup>9</sup>. Essa biblioteca é a principal ferramenta para obtenção e análise de dados em Python. Ela oferece estruturas de dados especiais para trabalhar com conjunto de dados, sendo o DataFrame a sua principal estrutura. Com ela podemos repartir, agrupar, ler diversos tipos de arquivo, criar visualizações rápidas, aplicar funções a colunas, dentre várias outras funcionalidades. Na sequencia apresentam-se os passos seguidos:

---

<sup>7</sup><https://www.kaggle.com/c/titanic>

<sup>8</sup>[scholar.google.com](http://scholar.google.com)

<sup>9</sup>[pandas.pydata.org](http://pandas.pydata.org)

- Criar um novo notebook para o projeto, em sua [página principal](#)<sup>10</sup> selecionamos a aba CODE e clicamos no botão NEW NOTEBOOK, vide Figura 1.

The screenshot shows the Kaggle homepage for the 'Titanic - Machine Learning from Disaster' competition. On the left, there's a sidebar with links like 'Create', 'Home', 'Competitions', 'Datasets', 'Code', 'Discussions', 'Courses', and 'More'. The main area features a large image of the Titanic ship. Below it, the competition title 'Titanic - Machine Learning from Disaster' is displayed along with the tagline 'Start here! Predict survival on the Titanic and get familiar with ML basics'. A 'Kaggle' logo indicates 13,658 teams are ongoing. At the bottom of the main area, there are tabs for 'Overview', 'Data', 'Code' (which is underlined), 'Discussion', 'Leaderboard', 'Rules', and 'Team'. A prominent 'New Notebook' button is located at the bottom right of the main content area. Below the main content, there's a search bar for 'Search notebooks' and a 'Filters' button.

Figura 1: Criação de um novo notebook.

- Como se mostra na Figura 2, nosso novo notebook já tem o código para a listar os arquivos disponíveis, bem para carregar as bibliotecas normalmente utilizadas nesse contexto, dentre elas utilizaremos a Pandas. A outra biblioteca carregada é a Numpy, é a principal ferramenta para computação numérica e álgebra linear em Python, não será utilizada diretamente nesse projeto, mas é fundamental em muitas operações numéricas mais complexas.

The screenshot shows a Kaggle Notebook titled 'Titanic' in a 'Draft saved' state. The interface includes a top navigation bar with 'File', 'Edit', 'View', 'Run', 'Add-ons', 'Help', and a 'Code' dropdown. To the right of the code editor are buttons for 'Share', 'Save Version' (with a count of 0), and a 'Run' button. The code editor itself contains the following Python script:

```

# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session

/kaggle/input/titanic/train.csv
/kaggle/input/titanic/test.csv
/kaggle/input/titanic/gender_submission.csv

```

Below the code editor, there are buttons for '+ Code' and '+ Markdown'.

Figura 2: Notebook criado.

- Assim, utilizamos a função `READ_CSV()` da biblioteca Pandas (importada com o "apelido" de "pd", vide Figura 2), com o endereço do arquivo desejado. Assim, carregamos na memória e armazenamos em variáveis os três arquivos disponíveis no Titanic:

- `train.csv`, arquivo principal que utilizaremos para analisar e processar.
- `test.csv`, arquivo que será utilizado para gerar nossa solução final que será submetida para avaliação da plataforma Kaggle.
- `gender_submission.csv`, arquivo que exemplifica a submissão final à plataforma. Nesse caso gender por exemplificar utilizando o gênero como única variável para determinar a predição.

para carregarmos a tabela de treinamento e a tabela de teste respectivamente na memória, também criamos uma cópia do DATAFRAME TRAIN para a variável DF, tendo em vista podemos alterar DF, mantendo em memória o DATAFRAME original "train". Apresenta-se, na Figura 3, os comandos executados para esse fim.

<sup>10</sup><https://www.kaggle.com/c/titanic>

```

train = pd.read_csv("/kaggle/input/titanic/train.csv")
test = pd.read_csv("/kaggle/input/titanic/test.csv")
gender_submission = pd.read_csv("/kaggle/input/titanic/gender_submission.csv")
df = train.copy()

```

Figura 3: Carregamento das tabelas.

## 4.4 Data Analysis

Nessa etapa, iniciamos construindo um repertório de técnicas e conceitos a partir das ideias de [Klo19], [CFG20], e simultaneamente exploramos os dados desenvolvendo a análise do projeto Titanic. Assim como utilizamos o ecossistema construído, aliado aos referenciais teóricos para ganhar fluência no ecossistema da linguagem Python para análise de dados. Isto é, utilizando a biblioteca Pandas para ler arquivos CSV, processar tabelas e transformar dados.

### 4.4.1 Técnicas e conceitos

- Utilizamos a função `.INFO()` da biblioteca Pandas para nos apresentar um resumo da tabela: lista de suas colunas, quantidades de linha, número de entradas em cada linha. Por exemplo, usando o comando `DF.INFO()`, obtém-se um resumo da tabela de treinamento `DF`. Maiores detalhes podem ser observados na Figura 10 na seção 5.
- Utilizamos a função `.DESCRIBE()` para observarmos o resumo estatístico dos dados, apresentando dados como média, desvio padrão, quartis.
- Utilizamos a função `.MAP()` para mapear cada entrada da coluna a um dicionário. Desse modo, a função percorre cada entrada de uma determinada coluna e caso seu valor esteja presente como chave do dicionário, a função o substitui pelo valor do dicionário. Assim podemos por exemplo ter uma coluna "Sex" com valores 'male' e 'female' e um dicionário {'male':0,'female':1}, ao aplicar o dicionário na coluna, as strings serão substituídas pelos inteiros correspondentes. Assim podemos fazer análises numéricas dessa coluna também.
- Utilizamos a função `GROUPBY()` para relacionar duas colunas e observar seu comportamento de acordo com a outra. Por exemplo, podemos observar a média da idade dos sobreviventes, e comparar com a média de idade dos que não sobreviveram. Nesse caso agrupamos todas as variáveis numéricas de acordo com a sobrevivência, de modo que temos a média de cada coluna em cada destino.
- `.HIST()` nos apresenta o histograma da coluna selecionada. Como parâmetros opcionais utilizamos o `BINS=` para determinar quantas barras serão utilizadas, `SHAREX=True` para ambos os histogramas compartilharem o eixo x, `SHAREY=False` para que a escala de y possa ser de acordo com cada histograma, assim podemos ver a diferença percentual e não apenas absoluta.
- Utilizamos a função `.PLOT()` com o parâmetro `KIND="BARS"` para construirmos um gráfico de barras representando a relação de duas variáveis. Podemos, por exemplo visualizar a incidência de sobrevivência para cada número de familiares a bordo.

Já na função final de preprocessamento temos as seguintes técnicas:

- Utilizamos a função `.DROP()` para remover as colunas que não utilizaremos como entrada. Com os parâmetros `AXIS=1` para selecionar colunas e `INPLACE=True` para que modifique o próprio *dataset* ao invés de retornar uma cópia.
- Criamos a função `LABEL_ENCONDER_CONVERTER()` para transformar as variáveis textuais em numéricas.
- Criamos a função `TRANSFORM_COLUMN()` para ser aplicada no *dataframe* de modo que preencha a coluna AGE de acordo com a mediana da classe e sexo de cada pessoa com esse campo vazio.

Tais técnicas foram utilizadas na construção da função WRANGLE() apresentada na Listagem 1. Utilizada para efetuar o preprocessamento nos dados obtidos a partir do que foi estudado na etapa de *Profiling*. Desse modo, sempre que tivermos dados análogos aos dados de teste eles serão inicialmente transformados por meio dessa função antes de serem processados pelo modelo que será desenvolvido na etapa de Aprendizado de Máquina.

```
def Wrangle(df):
    """Recebe e transforma o dataframe"""
    dropped_columns = ["PassengerId", "Name", "Ticket", "Cabin", "Sex_encoded"]
    df.drop(dropped_columns, axis=1, inplace=True)

    def label_encoder_converter(df):
        """Codifica em números as variáveis categóricas"""
        df["Embarked"] = df["Embarked"].map({"C":3, "Q":2, "S":1, np.nan:0}).astype(int)
        df["Sex"] = df["Sex"].map({"male":0, "female":1}).astype(int)

    def Transform_column(column, age_table):
        """Preenche os dados faltantes da coluna Age a partir da classe e sexo de cada passageiro"""
        Age = column[0]
        Sex = column[1]
        Pclass = column[2]

        if(pd.isna(Age)):
            return age_table.loc[Sex, Pclass]
        else:
            return Age

    age_table = df.pivot_table(index="Sex", columns="Pclass", values="Age", aggfunc="median")
    df["Age"] = df[["Age", "Sex", "Pclass"]].apply(Transform_column, axis = 1, age_table=age_table)
    df["Family"] = df["SibSp"] + df["Parch"]
    label_encoder_converter(df)
```

Listing 1: Função WRANGLE()

## 4.5 Aprendizado de Máquina ou *Machine Learning*

Seguindo e estudando as diretrizes do modelo apresentado em [CFGT20], dos minicursos disponíveis no Kaggle<sup>11</sup> <sup>12</sup> <sup>13</sup>, da documentação oficial da biblioteca *Scikit Learn*[PVG<sup>+</sup>11]<sup>14</sup>, assim como os livros [Gru19] e [Ozd16]; buscou-se então definir um referencial teórico dos principais conceitos da etapa de aprendizado de máquina.

Os conceitos foram aplicados no Titanic da seguinte forma:

- A estrutura geral do Aprendizado de Máquina se dá por meio de um conjunto de dados, também chamado de *DataFrame*. Inicialmente precisamos de um *DataFrame* de treino, isto é, com uma coluna chamada TARGET com a variável que queremos predizer, e outras colunas que serão utilizadas para efetuar essa predição, denominadas de FEATURES. Dessa forma, um modelo preditivo é treinado com um *DataFrame* de treino e pode posteriormente ser utilizado para prever a variável TARGET em *DataFrames* inéditos para o modelo. No problema aqui apresentado sobre o projeto Titanic, o arquivo train.csv seria o *DataFrame* de teste, apresentado anteriormente na Figura 3 com a variável SURVIVED como o TARGET.
- Importamos os módulos relevantes da biblioteca *Scikit Learn*:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

Em relação a esses módulos, trouxemos uma breve explicação deles na sequência:

1. Função "TRAIN\_TEST\_SPLIT()" para separar aleatoriamente os nossos dados de teste de modo que possamos utilizar parte para treinar o modelo e outra parte para validar e avaliá-lo.
2. "LogisticRegression", "DecisionTreeClassifier", "RandomForestClassifier", "KNeighborsClassifier" e "SVC" são as implementações dos respectivos modelos de Aprendizado de Máquina disponíveis na biblioteca.

- Construímos um dicionário para armazenar os modelos:

```
models = {"Logistic Regression": LogisticRegression(),
          "Decision Tree Classifier": DecisionTreeClassifier(),
          "Random Forest Classifier": RandomForestClassifier(random_state = 0),
          "KNN": KNeighborsClassifier(),
          "Support Vector Classifier": SVC()}
```

- Criamos a função EVALUATE\_MODELS() que recebe o dicionário de modelos e o *DataFrame* com os dados de treino para automaticamente treinar e avaliar cada modelo conforme a Listagem 2.
- Após a primeira avaliação, experimentamos aplicar a técnica normalização e comparar os resultados. Nesse sentido:

1. Importamos a função escaladora normal da biblioteca *Scikit Learn* usando a seguinte linha de comando:

```
from sklearn.preprocessing import StandardScaler
```

2. Copiamos o DF, modificamos suas duas colunas AGE e FARE para suas versões normalizadas.

```
df_2 = df.copy()
df_2[["Age", "Fare"]] = StandardScaler().fit_transform(df[["Age", "Fare"]])
```

<sup>11</sup><https://www.kaggle.com/learn/intro-to-machine-learning>

<sup>12</sup><https://www.kaggle.com/learn/intermediate-machine-learning>

<sup>13</sup><https://www.kaggle.com/learn/feature-engineering>

<sup>14</sup>[scikit-learn.org](http://scikit-learn.org)

```

def Evaluate_models(models,df):
    """Recebe o dicionario de modelos e o dataframe."""
    """Treina e avalia cada modelo retornando suas respectivas acuráncias"""

    X = df.drop(["Survived"],axis=1)
    #remove a coluna Survived do dataset
    #de modo que X armazena as Features
    y = df["Survived"]
    #y armazena apenas a variável Target
    #arma

    acc = {}
    #cria dicionario vazio, acc como sigla para accuracy.
    X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size = 0.2, random_state = 0)
    #separa e armazena em y 20 por cento das linhas do df para validarmos o modelo

    for name,model in models.items():
        #para cada item do dicionario de modelos:
        model.fit(X_train, y_train)
        #treina o modelo
        y_pred = model.predict(X_valid)
        #utiliza o modelo treinado para predizer o y que conhecemos
        acc[name] = model.score(X_valid, y_valid)
        #armazena no dicionario de resultados a pontuação do modelo

    results = pd.DataFrame.from_dict(acc,orient="index",columns=["Score"])
    #transforma o dicionario de resultados em uma tabela pandas
    return results

```

Listing 2: Definição da Função EVALUATE\_MODELS

## 4.6 Data Visualization

Nesse primeiro ano utilizamos apenas os recursos mais simples das bibliotecas e conceitos de visualização, de modo que bastou a documentação oficial da biblioteca Pandas para utilizarmos a função de histograma (*hist()*), plotagem (*plot()*) bem como a construção de tabelas especiais para visualizarmos relações entre variáveis com a função *groupby()* conforme explicado na seção 4.4.

Também estudamos o artigo [CFGT20], escrito por pesquisadoras da Tableau, principal empresa de software para visualização interativa de dados e Business Intelligence, termo que se refere a ciência de dados aplicada especificamente na construção de soluções baseada em dados para empresas [NG08].

## 4.7 Deployment

O Deployment de uma solução de dados diz respeito ao processo de tornar o modelo construído acessível ao usuário final em seu ambiente “real”, também conhecido como ambiente de “produção”. Ou seja, onde poderá ser utilizado na prática para cumprir o seu objetivo.

No caso do nosso projeto Titanic, a solução final diz respeito a aplicar nosso melhor modelo no conjunto de dados denominado *test.csv*, para gerar a predição de sobrevivência de cada tripulante nele presente. Esse arquivo com as predições é a nossa resposta final que será submetida ao Kaggle para avaliação automática da nossa solução.

Desse modo, construímos a função PIPELINE() apresentada na Listagem 3, que recebe o *Data-Frame* do arquivo *test.csv* e retorna suas respectivas predições em um Array de zeros e uns para representar a sobrevivência predita de cada passageiro conforme mostrado na Figura 4.

```

def Pipeline(test):
    """Recebe um dataset apenas com as features e prediz o target"""
    X_test = Wrangle(test)
    #Aplica a função de pré-processamento ao dataset recebido
    rf = models["Random Forest Classifier"]
    #criamos uma variável para o melhor modelo treinado
    y_predict = rf.predict(X_test)
    #utilizamos o modelo treinado para predizer a variável target
    return y_predict

```

Listing 3: Descrição da Função PIPELINE()

```

preds_test = Pipeline(test)
preds_test

array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1])

```

Figura 4: Predição do resultado final.

Na sequência, construímos o arquivo final *submission.csv* conforme a listagem 4. Seguindo as especificações demonstrados pelo arquivo de exemplo *gender-submission.csv* como podemos observar na Figura 5 ambos com as mesmas colunas e passageiros.

```

pyreds_test = Pipeline(test)
#armazenamos o Array com as predições geradas pela função Pipeline

output = pd.DataFrame({'PassengerId': test.PassengerId,
                       'Survived': preds_test})
#transformamos o Array resultante em um dataframe
output.to_csv('submission.csv', index=False)
#geramos um csv a partir do dataframe

```

Listing 4: Gerando o arquivo *submission.csv*

A partir da criação do arquivo *submission.csv* basta selecionarmos, na plataforma Kaggle, a opção "Competitions" no menu à direita de nosso notebook e clicarmos na opção "submit" apresentada na Figura 6. A plataforma automaticamente identificará o arquivo e avaliará a solução.

gender_submission			pd.read_csv("/kaggle/working/submission.csv")		
	PassengerId	Survived		PassengerId	Survived
0	892	0	0	892	0
1	893	1	1	893	0
2	894	0	2	894	0
3	895	0	3	895	0
4	896	1	4	896	0
...	...	...	...	...	...
413	1305	0	413	1305	0
414	1306	1	414	1306	1
415	1307	0	415	1307	0
416	1308	0	416	1308	0
417	1309	0	417	1309	0
418 rows × 2 columns			418 rows × 2 columns		

Figura 5: Comparação entre nossa solução e o exemplo.

DBS97\_Titanic Draft saved

File Edit View Run Add-ons Help

Share Save Version 3

[118]: pd.read\_csv("/kaggle/working/submission.csv")

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	0
...	...	...
413	1305	0
414	1306	1

Competitions

COMPETITION  
Titanic - Machine Learning  
from Disaster



LATEST SCORE  
0.77751 V3

BEST SCORE  
0.77751 V3

DAILY SUBMISSIONS  
1 / 10 used

Submit

Figura 6: Submetendo o arquivo resultando ao Kaggle.

## 5 Resultados

Nesta seção apresentamos os primeiros resultados obtidos nesta pesquisa usando o Projeto Titanic da Plataforma Kaggle.

### 5.1 Bases Gerais

Nesse primeiro subprojeto buscamos estruturar uma definição para a ciência de dados a partir dos artigos estudados.

#### 5.1.1 O que é Ciência de Dados?

[CFG20] define a ciência de dados como o campo multidisciplinar que busca novas ideias a partir de dados do mundo real através da aplicação estruturada de técnicas primariamente estatísticas e computacionais. O artigo também afirma que na literatura ainda há considerável debate quanto aos contornos desse campo, porém, os especialistas tendem a concordar que a necessidade de integrar diversas disciplinas, aliada aos desafios de criar uma infraestrutura analítica robusta, introduziram um conjunto único de desafios que são melhor enfrentados por profissionais chamados "cientistas de dados". Ao que [Don17] acrescenta:

"Ciência de Dados" é o melhor campo para acomodar pesquisa de grande impacto que seria difícil de classificar de outro modo, como por exemplo Tukey 1977 EDA e o trabalho de Wickham em Tidy Data e Grammar of Graphics.

Essa nova ciência, conclamada em 1962 por John Tukey, foi recebida com timidez e controvérsia entre os estatísticos. Mas, meio século depois, com a popularização de linguagens e ambientes de programação quantitativos como o R[IG96], ela se tornou auto-evidente. Isto porque o que antes era uma descrição em prosa de uma análise, agora poderia ser precisamente descrito em um SCRIPT com alto nível de abstração. Esses chamados *workflows* podem ser compartilhados, reexecutados com outros dados, facilmente modificados e quantificados. Logo, se tornaram evidentes objetos de pesquisa, [Don17].

Ambos os artigos [CFG20, Don17] observaram na literatura a precisão da profecia de Tukey, especialmente em sua afirmativa sobre a Ciência de Dados ser uma disciplina orientada pela prática. Ou seja, seu sucesso deriva do valor gerado ao resolver problemas do mundo real, maximizar resultados, e a sua efetividade em gerar valor.

#### 5.1.2 Processos que constituem a Ciência de dados

Outra dimensão ressaltada por Tukey e observada por [Don17] especialmente nas ideias de John Chambers[M.93] and Bill Cleveland[Cle01] é a ciência que se forma no entorno do "aprender com dados do mundo real" e tudo que esse processo engloba. Especialmente a possibilidade da Ciência de Dados construir e oferecer ferramental que possibilite o avanço de todas as outras ciências, ao aprimorar todo o processo de transformar dados em informação, desde ferramentas de processamento computacional, até bases epistemológicas desse processo.

[Don17] chama essa disciplina expandida de *Greater Data Science (GDS)*, e a divide em seis sub-áreas:

1. Data Gathering, Preparation, and Exploration
2. Data Representation and Transformation
3. Computing with Data
4. Data Modeling
5. Data Visualization and Presentation
6. Science about Data Science

Já [CFG20] traz uma abordagem observacional, processando o que já está registrado na literatura com pesquisas de mercado. O artigo sintetiza o Modelo de Trabalho ou *Data Work Model*, constituído de quatro processos de alta ordem, divididos em 14 subprocessos da seguinte forma:

- **Preparation:** Defining Needs, Data Gathering, Data Creation, Profiling, and Data Wrangling

- **Analysis:** Experimentation, Exploration, Modeling, Verification, and Interpretation.
- **Deployment:** Monitoring and Refinement
- **Communication:** Dissemination and Documentation

O artigo também identifica mais dois processos de alta ordem emergindo:

- **Pedagogy**
- **Collaboration**

Podemos observar que ambas as definições são compatíveis e adequadas à sua proposta: a GDS [Don17] como orientação teórica de pesquisa e organização da literatura, enquanto o Data Work Model [CFG20] oferece uma excelente estrutura para abordar um projeto de dados.

## 5.2 Definição de problemas de dados

### 5.2.1 Principais conceitos

O livro [Klo19] ressalta a importância fundamental de se entender o "problema do negócio" e circunscreve-lo em uma definição matemática. O que muitas vezes significa ter um papel fundamental na definição do "que será feito e como será feito". Portanto, a ciência de dados requer um diálogo imprescindível entre o cientista e pessoas especializadas na área do "negócio".

Nesse sentido, segundo o modelo [CFG20], a definição do problema de dados é o processo de traduzir objetivos analíticos, geralmente definidos pelas partes interessadas, em um conjunto de requerimentos viáveis. Dentre tais requerimentos estão:

- Dados necessários
- Planos para analisá-los
- Definir quais serão as entregas, como por exemplo: relatórios, modelos e infraestrutura computacional.

### 5.2.2 Titanic

No projeto Titanic, e nas competições Kaggle em geral, já temos o "problema de negócio" definido. Nesse caso, nosso problema é determinar a partir dos seus dados, quais passageiros sobreviveriam ao trágico acidente, tendo em vista, especialmente um olhar crítico para os resultados, analisando quais foram as características dos passageiros que aumentaram suas chances de sobrevivência.

A partir desse problema devemos pensar em quais dados precisamos obter para abordá-lo. Nesse caso, precisamos de uma amostra com os dados de diversos passageiros e o seu destino. Podemos utilizar tal amostra para treinar um modelo preditivo que possa receber outro conjunto análogo apenas com as informações e predizer o destino de cada um. Dessa forma, no processo de construção desse modelo esperamos compreender quais foram os fatores que mais contribuíram para o desfecho de cada pessoa à bordo.

Já temos nessa etapa da definição do problema, conceitos iniciais de *Aprendizado de Máquina*. Isto porque temos uma variável a ser predita, também chamada de variável alvo (**target**), variável resposta ou **y**. Para predizê-la, temos um conjunto de **features**, isto é, as características de cada passageiro em nosso conjunto de dados, também chamado de *DataFrame* ou *DataSet*. A partir das **features**, geramos um conjunto de variáveis **X** para prever nossa variável **y**. Dessa forma, precisamos de um *dataset* com **features** e variável alvo. Estes dados de treino são inseridos em um modelo que tentará modelar matematicamente as relações de **X** para **y**, de modo que posteriormente para qualquer outro conjunto análogo a **X**, o modelo possa prever um valor de **y**.

Assim, sabemos que devemos construir um modelo preditivo de classificação, que para cada conjunto de dados  $X_i$  de um passageiro  $i$ , ele possa gerar um  $y_i$  de valor zero ou um, representando a sua sobrevivência. Também sabemos que se faz necessário, portanto, obtermos um *dataset* com **X** e **y** para treinarmos nosso modelo. O que também já nos aponta para o uso da biblioteca Pandas para estruturar esse **X** além do uso da biblioteca *Scikit Learn* para criar, treinar e utilizar um modelo de Aprendizado de Máquina em português.

## 5.3 Obtenção de Dados

### 5.3.1 Conceitos principais

Uma vez definidos os requerimentos, o passo seguinte de acordo com o modelo [CFGT20], é a obtenção de dados. Para tanto, o modelo ressalta três possibilidades:

- Identificar o conjunto de dados adequado dentre vários candidatos.
- Gerar novos dados a partir da integração e transformação de dois ou mais conjuntos existentes.
- Desenvolver critérios e mecanismos para coletar dados que ainda não existem.

Tanto a primeira, quanto a segunda situação se referem aos casos nos quais o "cliente" do serviço de dados já dispõem de um banco de dados estruturado. Nesse caso, normalmente, a obtenção envolve escrever uma *query* na linguagem SQL. Do contrário, quando o cliente dispõem dos dados porém em formato inadequado, é necessário primariamente adequá-los ao primeiro caso, o que normalmente também envolverá a linguagem SQL, provavelmente *scripts* e possivelmente certa organização manual de dados.

No terceiro caso, há uma variedade maior de possibilidades, desde a realização de pesquisa como questionários, até a criação de *scripts* para capturar da internet as informações relevantes, processo que chamamos de *Webscraping*[Mit18].

O livro [Klo19] também ressalta a importância da documentação do conjunto de dados obtidos. Nela devem estar as principais informações sobre o *dataset*, tais como a explicação de suas colunas e seus conteúdos, definições de termos, seu contexto, dentre outras.

### 5.3.2 SQL

Conforme mencionamos na metodologia, a linguagem SQL não foi necessária no projeto Titanic, entretanto muitos autores como [Gru19, Klo19] ressaltam a sua importância, especialmente no ferramental de um cientista de dados. [DeB22] a define da seguinte maneira:

SQL é uma linguagem de programação amplamente utilizada que permite definir e consultar bancos de dados. Seja você um analista de marketing, um jornalista ou um pesquisador mapeando neurônios no cérebro de uma mosca, você se beneficiará do uso do SQL para gerenciar objetos de banco de dados, bem como criar, modificar, explorar e resumir dados.

Nesse sentido, elencamos recursos para aprender essa habilidade que esperamos nos aprofundar nos próximos trabalhos. Identificamos os livros [DeB22, SKS19] como as referências interessantes. Além deles, acreditamos que a forma mais eficiente de se aprender uma habilidade prática é por meio da prática direcionada, nesse sentido, os seguintes sites se mostraram os mais eficientes em apresentar exercícios práticos:

- [SQLbolt](#)<sup>15</sup>
- [Hacker Rank](#)<sup>16</sup>
- [PostgreSQL Exercises](#)<sup>17</sup>

### 5.3.3 Titanic

À partir da definição do problema, buscamos obter os dados disponíveis ou criá-los. No caso de nosso projeto Titanic, como em todas as competições Kaggle, os dados são disponibilizados pela plataforma e podem ser obtidos conforme descrito na metodologia. Temos, portanto, os *DataFrames* para o treinamento e validação do modelo, predição e submissão à plataforma Kaggle e exemplo de submissão derivados dos arquivos *train.csv*, *test.csv*, e *gender\_submission.csv* respectivamente apresentados nas Figuras 7, 8 e 9 respectivas.

---

<sup>15</sup>[sqlbolt.com/](http://sqlbolt.com/)

<sup>16</sup>[www.hackerrank.com/domains/sql](http://www.hackerrank.com/domains/sql)

<sup>17</sup>[pgexercises.com](http://pgexercises.com)

[3]: df

	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... Heikkinen, Miss. Laina	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0	0	0	STON/O2, 3101282	7.9250	NaN	S
3	4	1	1	Allen, Mr. William Henry	male	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Montvila, Rev. Juozas	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
887	888	1	1	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W/C. 6607	23.4500	NaN	S
888	889	0	3	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
889	890	1	1	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q
890	891	0	3									

891 rows × 12 columns

Figura 7: *DataFrame* treinamento lido do arquivo train.csv

[4]: test

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	Oliva y Ocana, Dona. Felmina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
417	1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

418 rows × 11 columns

Figura 8: *DataFrame* do arquivo test.csv, que será predito e submetido à plataforma

► gender\_submission

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
...	...	...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	0

418 rows × 2 columns

+ Code + Markdown

Figura 9: *DataFrame* gender\_submission.csv submetido à plataforma Kaggle

- *train.csv*: uma tabela com 891 linhas e 12 colunas, respectivamente representando cada passageiro, e suas *features*, inclusive se ele sobreviveu. Exatamente o *dataset* descrito em

nosso requerimento, portanto será o nosso principal *DataFrame* de trabalho, que iremos analisar, transformar e utilizar para treinar nosso modelo preditivo.

- *test.csv*: muito semelhante ao arquivo anterior, contem outros 418 passageiros, mas dessa vez apenas 11 colunas. Isto porque a coluna "SURVIVED" está ausente, exatamente a coluna que nosso modelo se propõe a preencher. Este, é o conjunto de dados cuja resposta será utilizada na avaliação automática que o kaggle faz em nosso modelo.
- *gender\_submission.csv*: é um exemplo de submissão a competição kaggle, ela contém as mesmas 418 linhas e a mesma coluna "PASSENGERID" que o arquivo *test.csv* contém, acrescida da coluna "SURVIVED", nesse caso, preenchida a título de exemplo de modo que todas as passageiras sobreviveram e todos os passageiros morreram. Esse é um exemplo do arquivo que será submetido a competição e utilizado para avaliar nossa solução para o problema.

## 5.4 Data Analysis

### 5.4.1 Conceitos principais

[CFGT20] aponta que esse processo, denominado de Data Analysis pelo livro [Klo19], também pode ser encontrado com outros nomes na literatura como "pré-processamento", "tidying", "limpeza", "wrangling", "Exploratory Data Analysis (EDA)" ou ainda "Data Understanding". Nesse sentido, as autoras defendem dividi-lo entre "Profiling" e "Data Wrangling" e agrupa-los junto aos passos anteriores, de modo a constituir o macro processo de "Preparation".

Assim, de acordo com o modelo apresentado, temos as seguintes definições:

- **Profiling**: é o processo de avaliar atributos dos dados para entender a distribuição de seus valores, identificar valores faltantes e examinar a sua associação à outros atributos. Bem como compreender os dados e seu conteúdo.
- **Data Wrangling**: processo de utilizar o que foi aprendido no Profiling para conformar e transformar os dados de modo a adequá-los para os próximos passos.

Nesse sentido, dentro do ecossistema Python, a biblioteca Pandas é a ferramenta absoluta para manipulação de tabelas, aliada a NumPy, que traz um robusto repertório de implementações matemáticas e aplicações de álgebra linear.[Ozd16]

### 5.4.2 Técnicas

Segundo [CFGT20, Don17, Klo19] Essa etapa é sabidamente a mais dispendiosa de um projeto de dados. Devido principalmente ao tempo necessário e dificuldade de automatizar. Dessa forma, é também uma das etapas mais difíceis de se ensinar justamente em função da intervenção humana e seu processo quase artesanal para compreender adequar os dados.

Dessa forma, segundo [Klo19], devemos analisar os dados tanto diretamente quanto por meio de sumários gráficos e numéricos, mas principalmente analisar criticamente se nossos achados fazem sentido, e se estão de acordo com a documentação do dataset. Nesse sentido compilamos a seguinte lista de perguntas para nortear o processo de *Profiling*:

1. Quantas colunas? (podem ser features, variável resposta, ou metadados)
2. Quantas linhas? (checkar se cada linha é uma amostra)
3. Quais são os tipos de features? Quais são categóricas e quais são numéricas?
4. Como são esses dados? (em numéricas podemos examinar range de valores, ou frequência de diferentes classes em categóricas por exemplo)
5. Temos valores faltantes?

Se o dataset for pequeno o bastante podemos examinar individualmente cada feature. Já quando o dataset tem centenas de features devemos explorar técnicas de redução de dimensionalidade, que condensam informação em um número menor de features derivadas ou métodos de feature selection que podem auxiliar a encontrar features importantes dentre muitas candidatas. Nesse sentido, [Klo19] afirma:

"Tempo dedicado analisando critica e detalhadamente se um dataset cumpre seu propósito é um tempo bem gasto."

### 5.4.3 Titanic

No projeto Titanic, portanto, iniciamos o *Profiling* do nosso arquivo principal *train.csv*, por meio da variável ”DF” orientados pela lista de perguntas que construímos.

Como podemos observar na Figura 10, temos 12 colunas ou *features* e 891 linhas ou *dados*. Também logo observamos que as colunas AGE, CABIN e EMBARKED tem dados faltantes. As outras 9 *features* estão totalmente preenchidas. Observamos por meio do atributo *Dtype* que as colunas NAME, SEX, TICKET, CABIN e EMBARKED são variáveis categóricas (definidas como tipo *object*), enquanto as outras são numéricas. Podemos confirmar tal fato observando os conteúdos da tabela de treinamento mostrada na Figura 7 onde podemos ver as primeiras e últimas entradas desta tabela de treinamento.

```
[6]: df.info()  
  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 891 entries, 0 to 890  
Data columns (total 12 columns):  
 #   Column      Non-Null Count  Dtype     
---    
 0   PassengerId 891 non-null    int64    
 1   Survived     891 non-null    int64    
 2   Pclass       891 non-null    int64    
 3   Name         891 non-null    object    
 4   Sex          891 non-null    object    
 5   Age          714 non-null    float64   
 6   SibSp        891 non-null    int64    
 7   Parch        891 non-null    int64    
 8   Ticket       891 non-null    object    
 9   Fare          891 non-null    float64   
 10  Cabin         204 non-null    object    
 11  Embarked     889 non-null    object    
dtypes: float64(2), int64(5), object(5)  
memory usage: 83.7+ KB
```

Figura 10: Informação sobre a tabela de treinamento, usando o comando DF.INFO()

Logo em seguida observamos a tabela DF e estudamos a documentação do *DataSet* para compreender cada uma das 12 colunas/features. Na sequência apresenta-se o significado de cada uma destas colunas no Projeto Titanic:

- PASSENGERID: o número que representa cada passageiro.
- SURVIVED: 0 ou 1, para representar se determinado passageiro sobreviveu.
- PCLASS: classe do ticket do passageiro. Equivalente a classe econômica 1-alta;2-média;3-baixa
- NAME: nome do passageiro com sobrenome e pronome de tratamento.
- SEX: masculino ou feminino.
- AGE: idade em anos, em bebês abaixo de 1 ano é uma fração estimada.
- SIBSP: # irmãos e cônjuges a bordo.
- PARCH: # parentes e filhos a bordo.
- TICKET: número do ticket
- FARE: Tarifa paga pelo passageiro.
- CABIN: Número da cabine.
- EMBARKED: Porto de embarque. C = Cherbourg, Q = Queenstown, S = Southampton

Na sequência, analisamos suas principais métricas (média, desvio padrão, quartis). Na Figura 11 se apresenta o resultado da execução do comando DF.DESCRIBE().



```
df.describe()
```

[7]:	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

Figura 11: Estatística descritiva dos dados do treinamento.

A partir dessas observações podemos desenvolver as seguintes suposições:

- PASSENGERID pode ser desconsiderada, uma vez que é apenas um artefato computacional para denominar as colunas.
- NAME a princípio também será desconsiderado, uma vez que é pouco provável que tenha uma relação direta com a sobrevivência portanto a complexidade de quantificá-la não parece valer.
- SEX pode ser convertida em 0 e 1.
- AGE tem um número considerável de linhas com dados faltantes, poderíamos excluí-la, porém faria sentido a idade ter alguma relação com a sobrevivência, nesse caso iremos lidar com os valores faltantes. Também podemos criar uma *Feature* com intervalos de idade.
- SIBSP e PARCH podem ser combinados em uma terceira *feature* que combina os familiares à bordo, seria interessante saber se deveríamos somar ou multiplicar esses números.
- TICKET também pode ser, a princípio, desconsiderado, uma vez que não temos na documentação a explicação de seu código alfa numérico, e assim como NAME, essa *feature* não parece ter direta relação com a sobrevivência.
- FARE, parece bem promissora, uma vez que não apresenta dados faltantes, já é numérica, e parece ter uma relação clara com a sobrevivência.
- CABIN, assim como TICKET, não temos uma tradução numérica, poderíamos construir uma *Feature* como a contagem de cabines, entretanto temos muitos valores faltantes, portanto, é provável que desconsiderá-la seja a melhor opção.
- EMBARKED, pode ter alguma relação com a sobrevivência, provavelmente a trataremos.

Podemos utilizar a pivotagem pra nos indicar a importância relativa de nossas *features* numéricas, ou seja, agrupar cada uma delas de acordo com a sobrevivência por meio de suas médias. Assim teremos para cada coluna o seu valor médio entre os sobreviventes e entre os não sobreviventes, desse modo, variáveis que foram importantes como mostram as Figuras 12 e 13.

```

df["Sex_encoded"] = df["Sex"].map({"male":0,"female":1})
#Criamos a coluna "Sex_encoded" para armazenar os sexos como 0 e 1.
df["Family+"] = df["SibSp"] + df["Parch"]
df["Family*"] = df["SibSp"] * df["Parch"]
#Criamos a coluna "Family" combinando o número SibSp e Parch
df.groupby(["Survived"]).mean()
#Agrupamos a média de cada coluna em relação a sobrevivência.

```

[12...]	PassengerId	Pclass	Age	SibSp	Parch	Fare	Sex_encoded	Family+	Family*
	Survived								
0	447.016393	2.531876	30.626179	0.553734	0.329690	22.117887	0.147541	0.883424	0.688525
1	444.368421	1.950292	28.343690	0.473684	0.464912	48.395408	0.681287	0.938596	0.374269

Figura 12: Agrupando cada coluna em média de acordo com a sobrevivência.

```
df[ [ "Embarked", "Survived" ] ].groupby( [ "Embarked" ] ).mean()
```

Survived	
Embarked	
C	0.553571
Q	0.389610
S	0.336957

Figura 13: Agrupando a sobrevivência de acordo com os portos de embarque.

A partir dela podemos constatar que a média de classe, sexo e custo da passagem foi bem diferente entre os que sobreviveram e os que não sobreviveram, o que sugere serem variáveis importantes. O número de irmãos e cônjuges, bem como o número de pais e filhos apresentaram uma diferença mais sutil, assim como a idade. Pode ser entretanto que haja alguma relação de grupos dessas *features* com a sobrevivência, conforme observamos na Figura 14 ao compararmos os histogramas das idades dos sobreviventes com o histograma das idades das vitimas. Também podemos observar nas Figuras 15 e 16 a existência de algum padrão de sobrevivência de acordo com a quantidade de parentes a bordo.

```
df[["Age"]].hist(by=df["Survived"], bins=10, sharey=False, sharex=True)
```

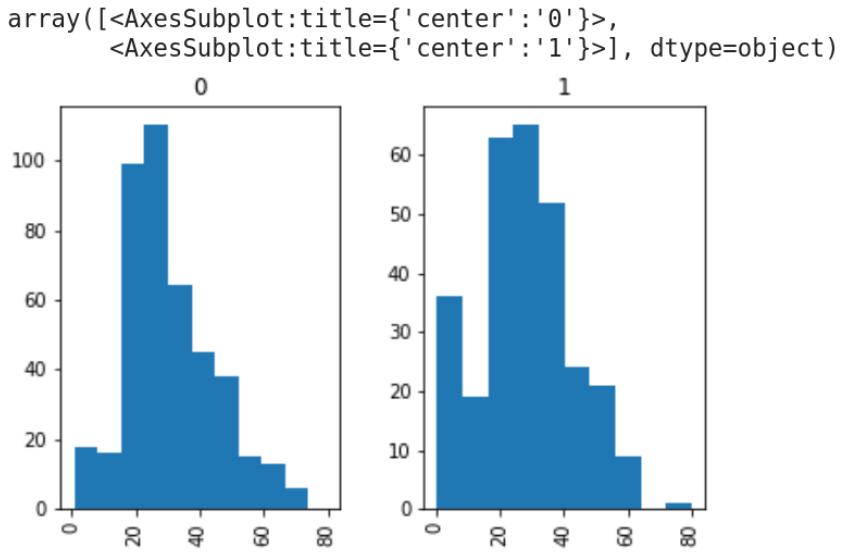


Figura 14: Histograma das idades dos sobreviventes e das vítimas.

```
df["Family"] = df["SibSp"]+df["Parch"]
df[["Survived", "Family"]].groupby(["Family"]).mean()
```

Survived	
Family	
0	0.303538
1	0.552795
2	0.578431
3	0.724138
4	0.200000
5	0.136364
6	0.333333
7	0.000000
10	0.000000

Figura 15: Índice de sobrevivência para cada quantidade de parentes a bordo.

```
df[["Survived", "Family"]].groupby(["Family"]).mean().plot(kind="bar")
```

```
<AxesSubplot:xlabel='Family'>
```

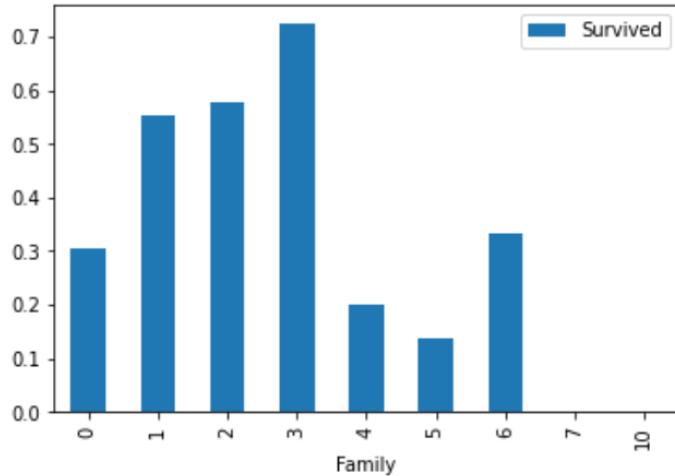


Figura 16: Gráfico do índice de sobrevivência para cada quantidade de parentes a bordo.

Assim, concluímos o seguinte:

- Removeremos as colunas: *PassengerId*, *Name*, *Ticket* e *Cabin*.
- Pclass já pode ser utilizada conforme se encontra.
- Modificaremos a feature Sex codificando os valores male e female para 0 e 1 respectivamente.
- Completaremos os valores faltantes da Age considerando a mediana da o sexo e a classe do passageiro do passageiro.
- Criaremos a feature Family combinando Parch e SibSp e a codificaremos numericamente.
- Fare, assim como Pclass já está pronta para ser utilizada.
- Codificaremos os portos em Embarked.

Finalmente, uma vez que estabelecemos um plano, podemos aplicá-lo ao nosso dataframe DF. Um ponto muito importante é que devemos estruturar as transformações de modo que possam ser aplicadas tanto nos dados de treino quanto nos dados que o modelo receberá em produção. Desse modo, obtemos nosso dataset tratado, conforme temos na Figura 17.

```
Wrangle(df)
```

```
df
```

	<b>Survived</b>	<b>Pclass</b>	<b>Sex</b>	<b>Age</b>	<b>SibSp</b>	<b>Parch</b>	<b>Fare</b>	<b>Embarked</b>	<b>Family</b>
<b>0</b>	0	3	0	22.0	1	0	7.2500	1	1
<b>1</b>	1	1	1	38.0	1	0	71.2833	3	1
<b>2</b>	1	3	1	26.0	0	0	7.9250	1	0
<b>3</b>	1	1	1	35.0	1	0	53.1000	1	1
<b>4</b>	0	3	0	35.0	0	0	8.0500	1	0
...	...	...	...	...	...	...	...	...	...
<b>886</b>	0	2	0	27.0	0	0	13.0000	1	0
<b>887</b>	1	1	1	19.0	0	0	30.0000	1	0
<b>888</b>	0	3	1	21.5	1	2	23.4500	1	3
<b>889</b>	1	1	0	26.0	0	0	30.0000	3	0
<b>890</b>	0	3	0	32.0	0	0	7.7500	2	0

891 rows × 9 columns

Figura 17: Aplicação da função *Wrangle()* para preprocessar os dados de treinamento.

## 5.5 Aprendizado de Máquina

### 5.5.1 Conceitos Principais

No modelo [CFGT20], após o macroprocesso de *Preparation* temos o macroprocesso *Analysis*. Dessa forma, essa etapa que chamamos de *Aprendizado de Máquina* é equivalente aos seguintes processos da pipeline:

- **Modeling:** aplicar técnicas computacionais e estatísticas para derivar informação que oriente possíveis ações a partir dos dados.
- **Verification:** é o processo avaliativo para confirmar a robustez dos resultados do código e dos modelos. É interessante notar que tem crescido a importância também de considerar a transparência e imparcialidade (fairness) dos processos de dados.
- **Interpretation:** processo de compreensão dos resultados, da exploração e do modelo no que diz respeito as suas aplicações no mundo real.

As principais ferramentas desse etapa no ecossistema Python são:

- Bibliotecas Pandas e NumPy utilizadas, assim como na etapa anterior, para manipulação de tabelas e operações com colunas.
- **Scikit Learn:** é a principal biblioteca para aprendizado de máquina em Python. Podemos utilizá-la para facilmente criar modelos dentro dos diversos tipos oferecidos. Ela também oferece ferramentas para suportar todo o processo de treiná-los, avalia-los, ajustá-los e utilizá-los. Além ser open source e de oferecer diversas outras funcionalidades como a criação de pipelines de processamento automatizado. [citar a própria página oficial do projeto]

Nesse contexto, quando falamos de *Aprendizado de Máquina* (ML), nos referimos a modelos preditivos que ”aprendem” a partir de dados. Normalmente, os modelos encontram as relações e correlações entre colunas de uma tabela. Nesse sentido, podemos definir um modelo como a especificação de uma relação matemática ou probabilística entre variáveis, [Gru19] [Ozd16].

[Ozd16] Ressalta as seguintes premissas universais a todos os modelos de ML:

- Os dados de entrada devem ser limpos e pré processados. Poucos modelos toleram dados sujos, com valores faltantes ou variáveis categóricas. Nesse sentido, parte importante da preparação dos dados envolve lidar com essas questões.
- Em geral, são muito sensíveis a dados ruidosos.
- Cada linha da tabela representa uma única observação do ambiente que tentamos modelar.
- Deve existir alguma relação entre variáveis.
- Modelos são geralmente considerados semi-automáticos, isto é, necessitam que humanos tomem decisões. Seu output é normalmente uma sequência de números. Um ser humano é necessário para analisar esses resultados com a perspectivas adequadas e comunica-los.

Nesse sentido, [Gru19] elenca os seguintes conceitos fundamentais à modelagem de dados:

- Overfitting e Underfitting: é a medida do quanto o modelo é capaz de aprender e generalizar determinado conjunto de dados. Overfitting ocorre quando o modelo ”decora” os dados de treino, portanto ele tem um excelente rendimento com dados conhecidos e péssimo com dados novos. Underfitting quando ele não aprende o suficiente.
- Bias-Variance trade off: analoga ao over e underfitting, um modelo com Bias alta performa mal até mesmo nos dados de treino (underfitting), enquanto uma variança alta significa que dados diferentes levariam a modelos muito diferentes, o que corresponde ao overfitting.
- Correctness: Existem diversas métricas para se avaliar um modelo. Normalmente se utiliza Acurácia, Precision and Recall, muitas vezes a sua média harmônica denominada F1-score.
- Feature selection, extraction e engineering: um dos conceitos mais importantes de um modelo são as suas features, a seguir detalhamos seus principais conceitos e técnicas.

### 5.5.2 Feature Engineering

O objetivo do Feature Engineering é aumentar a performance preditiva do modelo, diminuir a necessidade computacional e de dados, e finalmente melhorar a interpretabilidade dos resultados. Nessa sessão abordamos os principais fundamentos, e a práticas na preparação das features de um modelo.

Nesse sentido, [Gru19] ressalta os seguintes pontos centrais:

- Features são quaisquer inputs do nosso modelo.
- Quando nossos dados têm poucas features o modelo tende ao underfit. Do mesmo modo que o excesso de features faz com o que o modelo overfite.
- Existem três tipos principais: booleano (0 ou 1), numérico e categórico (uma escolha dentro um conjunto discreto de opções).
- O tipo das features pode restringir o tipo de modelo que pode ser usado. Como por exemplo os modelos de regressão requererem dados numéricos, enquanto árvores de decisão conseguem lidar tanto com dados numéricos quanto categóricos.

Dentre as práticas mais importantes nesse processo de construção das features de um modelo podemos citar:

- Compreender as features, estudar o dataset e sua documentação, pesquisar o problema a ser solucionado tanto com profissionais da área quanto por meio de livros e artigos. Relações numéricas entre features são normalmente expressas por fórmulas matemáticas. Frequentemente as descobrimos quando pesquisamos a área do problema.
- Estudar outras soluções para problemas similares, ou anteriores.

- Usar a visualização de dados. A visualização muitas vezes pode revelar patologias em uma distribuição, ou inspirar possíveis simplificações para relações complicadas, ou ainda sugerir algum tipo de transformação como por exemplo aplicar uma função logarítmica.
- Podemos utilizar uma função para medir a relação de determinada feature com a variável alvo e desse modo rankeá-las para facilitar a busca das features mais interessantes. Mutual Information é um exemplo de métrica utilizada nesse sentido, semelhante a correlação estatística, ela tem a vantagem de conseguir medir qualquer relação, não apenas linear entre variáveis.
- A utilidade de uma feature, entretanto, é diretamente proporcional a capacidade do modelo aprender a relação dela com a variável alvo. Desse modo, mesmo features com alto índice de relação com a variável alvo, podem precisar serem transformadas para expor ao modelo a sua associação a variável alvo. Quanto mais complexa a combinação, mais difícil para o modelo aprender a relação.

Além das práticas podemos citar as seguintes principais técnicas para abordar esse processo:

- Aplicação de uma potência, raiz, ou transformar valores absolutos em percentual ou proporções
- Normalização. Isto é, deslocar a distribuição dos dados de modo que sua média seja zero e escalar seus valores de modo que seu desvio padrão seja unitário.
- Criação de uma feature de contagem. Podemos agregar o número de fatores de risco, o número de ingredientes, etc.
- Frequentemente temos features complexas que podem ser separadas em features menores, como números de telefone (operadora e número), endereço (rua, cidade, país). Da forma semelhante podemos ter features que poderiam ser combinadas como modelo e marca de carro, ou tipo e nível de um seguro.
- Group Transforms. Podemos usar groupby para criar features como "média salarial no estado daquela pessoa", ou "Proporção de filmes lançados em determinado dia da semana, por gênero", ou "frequencia que o estado aparece no dataset"

### 5.5.3 Principais modelos

Apesar de existirem muitas classificações, modelos são geralmente divididos entre ([Gru19, Ozd16]):

- Supervisionados: são treinados com dados classificados, para fazer previsões sobre novos dados não-classificados.
- Não-supervisionados: não consideram nenhuma classificação, e sim busca agrupar os dados.

É importante considerar as características de cada tipo de modelo ao criar features para eles:

- Modelos lineares naturalmente aprendem somas e diferenças, mas não conseguem aprender nada mais complexo.
- Razões são normalmente difíceis para a maioria dos modelos, logo criar features com combinações de razões frequentemente levam a ganhos de performance.
- Modelos lineares e redes neurais geralmente funcionam melhor com features normalizadas. Modelos de árvores normalmente não apresentam tanto ganho de performance ao escalar os valores das features para valores próximos de zero (normalizar).
- Modelos de árvores podem aprender quase qualquer combinação de features, mas quando uma combinação é particularmente importante, pode ser benéfico criar uma feature explícita, especialmente quando temos dados limitados.
- Contagem são especialmente úteis para modelos do tipo árvore. Isso porque esses modelos não tem uma forma natural de agregar informação entre muitas features de uma vez.

Nesse projeto buscamos utilizar os modelos mais representativos que se adequassem ao problema apresentado [Van16, Aur17]. Gostaríamos de nos aprofundar nos mecanismos de cada um deles em pesquisas futuras. Utilizamos os seguintes modelos:

- Logistic Regression
- Deciosion Tree Classifier
- Random Forest Classifier
- KNN
- Suppor Vector Classifier

#### 5.5.4 Titanic

A partir do que foi visto na teoria, construímos uma estrutura para criar os cinco modelos estudados, automaticamente treiná-los e avaliá-los com os dados preprocessados. Desse modo, encontramos os seguintes resultados que serão apresentados na Figura 18.

```
results_1 = Evaluate_models(models, df)
results_1
```

Score	
<b>Logistic Regression</b>	0.810056
<b>Decision Tree Classifier</b>	0.826816
<b>Random Forest Classifier</b>	0.832402
<b>KNN</b>	0.731844
<b>Support Vector Classifier</b>	0.720670

Figura 18: Score dos diferentes modelos de Aprendizado de Máquina.

Como pode-se observar, modelos do tipo árvore foram os que melhor desempenho tiveram; seguidos da regressão logística, e finalmente os modelos não-supervisionados apresentaram o pior desempenho.

Após essa observação, aplicamos a técnica de normalização que vimos na seção 5.5.2, em nossas features AGE e FARE conforme mostrado na Figura 19. O resultado experimental confirma o que estudamos: modelos baseados em árvore e regressão logística são menos sensíveis à normalização. Constatamos na Figura 20 que de fato, sua pontuação não foi alterada enquanto os modelos não supervisionados apresentaram uma melhora considerável de desempenho.

```

from sklearn.preprocessing import StandardScaler
df_2 = df.copy()
df_2[["Age", "Fare"]] = StandardScaler().fit_transform(df[["Age", "Fare"]])
df_2

```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Family	Family*
0	0	3	0	-0.534891	1	0	-0.502445	1	1	0
1	1	1	1	0.668392	1	0	0.786845	3	1	0
2	1	3	1	-0.234070	0	0	-0.488854	1	0	0
3	1	1	1	0.442776	1	0	0.420730	1	1	0
4	0	3	0	0.442776	0	0	-0.486337	1	0	0
...	...	...	...	...	...	...	...	...	...	...
886	0	2	0	-0.158865	0	0	-0.386671	1	0	0
887	1	1	1	-0.760507	0	0	-0.044381	1	0	0
888	0	3	1	-0.572494	1	2	-0.176263	1	3	2
889	1	1	0	-0.234070	0	0	-0.044381	3	0	0
890	0	3	0	0.217161	0	0	-0.492378	2	0	0

891 rows × 10 columns

Figura 19: Comparaçāo da performance apōs a normalizaçāo de Age e Fare.

```

results = results_1.join(results_2, lsuffix='1')
results.columns = ["Results_1", "Results_2"]
results

```

	Results_1	Results_2
<b>Logistic Regression</b>	0.810056	0.810056
<b>Decision Tree Classifier</b>	0.810056	0.826816
<b>Random Forest Classifier</b>	0.832402	0.832402
<b>KNN</b>	0.731844	0.810056
<b>Support Vector Classifier</b>	0.720670	0.810056

Figura 20: Comparaçāo da performance apōs a normalizaçāo de Age e Fare.

Assim, concluimos que modelos de arvore funcionaram melhor, especialmente o Random Forest. Também observamos que os tripulantes mais abastados, mulheres, crianças e idosos tiveram mais chance de sobreviver.

## 5.6 Data Visualization

O artigo [CFG20] busca definir a ciência de dados e seu modelo de trabalho justamente tendo em vista localizar o papel da visualização de dados dentro desse campo. Concluindo que apesar de pouco reconhecido entre cientistas de dados, o uso da visualização percorre a maior parte das etapas de um projeto de dados.

Nesse ano de trabalho, a visualização de dados foi usada em diversos momentos das duas etapas anteriores (Análise de Dados e Aprendizado de Máquina), além da apresentação dos resultados. Tanto os gráficos que utilizamos, quanto as tabelas, e até todas as ferramentas, como o próprio conceito de notebook de código estão intimamente entrelaçados com conceitos de apresentação visual de informação, comunicação, arte e fundamentalmente design. Assim, um seguimento muito importante desse nosso primeiro ano envolve nos aprofundarmos e mapearmos essa área tão rica e diversa.

## 5.7 Deployment

### 5.7.1 Titanic

Conforme apresentamos na metodologia, essa etapa final de um projeto de dados envolve transferir a solução construída para o seu ambiente final. Desse modo, para o Titanic, criamos e submetemos o arquivo final com as previsões de nosso modelo para cada passageiro presente nos dados do arquivo test.csv. Tais dados são 418 novas entradas para as mesmas 11 colunas de features que o nosso arquivo de treino. Assim, após o treinamento dos modelos, aplicamos o modelo de melhor performance para predizer a sobrevivência desses 418 passageiros.

Nossa solução obteve o score de aproximadamente 78% no Kaggle, como mostra a Figura 21. Tal desempenho é compatível com os números encontrados na validação do modelo. Assim, uma vez submetido, podemos permitir que nosso caderno seja acessível ao público.<sup>18</sup>

---

<sup>18</sup><https://www.kaggle.com/danielbritodossantos/dbs97-titanic>

# Competitions



## COMPETITION

### Titanic - Machine Learning from Disaster



## LATEST SCORE

0.77751 V3

## BEST SCORE

0.77751 V3

## DAILY SUBMISSIONS

1 / 10 used

Submit

Figura 21: Pontuação final no Kaggle.

### 5.7.2 Aprofundamentos

Considerando que a entrega final de um projeto de dados pode ser um relatório, o deployment de um modelo ou de uma infra-estrutura computacional, como vimos na seção 5.2.1, nesse ano demonstramos a entrega de um modelo, neste relatório também podemos encontrar aspectos de um relatório de dados.

Entretanto, o deployment de infra estrutura envolve um considerável corpo de conhecimentos. Portanto, nesse ano identificamos os contornos dessa área, esperamos nas próximas etapas desenvolver seus principais conceitos. Tais como redes, aplicações e serviços web, técnicas para armazenagem de funções e modelos treinados, frameworks dentre outras.

## 6 Conclusão

Assim, podemos afirmar que nosso objetivo principal foi concluído. Uma vez que foi possível percorrer um vasto terreno, compilar diversas técnicas e conceitos, compreender a sua estrutura e como muitas dessas partes dialogam entre si, além de encontrarmos diversas oportunidades de aprofundamento.

Assim, concluímos citando Tukey [W.62] em tradução livre:

O futuro da [ciência de dados] pode envolver grande progresso, superar dificuldades reais, e prestar um grande serviço para todos os campos de ciência e tecnologia. Isso

vai acontecer? Depende de nós, de nossa vontade de percorrer o caminho difícil dos problemas reais ao invés do caminho suave das premissas irreais, critérios arbitrários e resultados abstratos sem impacto no mundo real. Quem aceita o desafio?

## 7 Perspectiva de continuidade

A partir da construção dos fundamentos da ciência de dados, identificamos diversas oportunidades de desenvolvimentos futuros, principalmente nas áreas de Data Visualization, Bancos de Dados e Aprendizado de Máquina. Especialmente na possibilidade de nos aprofundarmos em seus corpus de conhecimento e aplicá-los em projetos reais.

## 8 Participação em congressos e trabalhos publicados ou submetidos e outras atividades acadêmicas e de pesquisa

O bolsista apresentou a pesquisa em vídeo para o CONFICT 2021<sup>19</sup> (Figuras 23 e 22) [dS21].



Figura 22: Certificado de congressista.

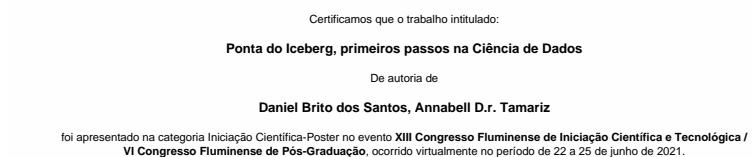


Figura 23: Certificado de apresentação.

<sup>19</sup><https://proceedings.science/confict-conpg-2021/papers/ponta-do-iceberg-primeiros-passos-na-ciencia-de-dados>

Também apresentamos os certificados dos minicursos oferecidos pela plataforma Kaggle nas figuras 24 e 25.



Figura 24: Certificado de conclusão do curso introdutório ao aprendizado de máquina.



Figura 25: Certificado de conclusão ao curso intermediário de aprendizado de máquina.

## 9 Data e assinatura do bolsista (assinatura digitalizada)

*Daniel Brito dos Santos*

24/01/2022

## 10 Data e assinatura do orientador (assinatura digitalizada)

*Annabell D.R. Tamariz*

28/01/2022

### Referências

- [Aur17] Géron Aurélien. Hands-on machine learning with scikit-learn & tensorflow. *Geron Aurelien*, 2017.
- [BM21] Casper Solheim Bojer and Jens Peder Meldgaard. Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, 37(2):587–603, 2021.
- [CB74] Donald D Chamberlin and Raymond F Boyce. Sequel: A structured english query language. In *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*, pages 249–264, 1974.
- [CFGT20] Anamaria Crisan, Brittany Fiore-Gartland, and Melanie Tory. Passing the data baton: A retrospective analysis on data science work and workers. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1860–1870, 2020.
- [Cle01] William S Cleveland. Data science: an action plan for expanding the technical areas of the field of statistics. *International statistical review*, 69(1):21–26, 2001.
- [DeB22] Anthony DeBarros. *Practical SQL: A Beginner’s Guide to Storytelling with Data*. No Starch Press, 2022.
- [Don17] David Donoho. 50 years of data science. *Journal of Computational and Graphical Statistics*, 26(4):745–766, 2017.
- [dS21] Daniel Brito dos Santos. Ponta do iceberg, primeiros passos na ciência de dados. *XIII CONFICT - VI CONPG.*, 2021.
- [Gra15] Ben Graham. Kaggle diabetic retinopathy detection competition report. *University of Warwick*, 2015.
- [Gru19] J. Grus. *Data Science from Scratch: First Principles with Python*. O’Reilly Media, 2019.
- [IG96] Ross Ihaka and Robert Gentleman. R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314, 1996.
- [IMO17] Vladimir Iglovikov, Sergey Mushinskiy, and Vladimir Osin. Satellite imagery feature detection using deep convolutional neural network: A kaggle competition. *arXiv preprint arXiv:1706.06169*, 2017.
- [Klo19] S. Klosterman. *Data Science Projects with Python: A case study approach to successful data science projects using Python, pandas, and scikit-learn*. Packt Publishing, 2019.
- [M.93] Chambers J. M. Greater or lesser statistics: A choice for future research. *Statistics and Computing*, 3:182, 1993.
- [Mit18] Ryan Mitchell. *Web scraping with Python: Collecting more data from the modern web*. “O’Reilly Media, Inc.”, 2018.
- [NG08] Solomon Negash and Paul Gray. Business intelligence. In *Handbook on decision support systems 2*, pages 175–193. Springer, 2008.

- [NSR11] Arvind Narayanan, Elaine Shi, and Benjamin IP Rubinstein. Link prediction by de-anonymization: How we won the kaggle social network challenge. In *The 2011 International Joint Conference on Neural Networks*, pages 1825–1834. IEEE, 2011.
- [OS13] C. O’Neil and R. Schutt. *Doing Data Science: Straight Talk from the Frontline*. Listen Alaska. O’Reilly Media, 2013.
- [Ozd16] S. Ozdemir. *Principles of Data Science*. Packt Publishing, 2016.
- [PRB14] Antti Puurula, Jesse Read, and Albert Bifet. Kaggle lshtc4 winning solution. *arXiv preprint arXiv:1405.0546*, 2014.
- [PVG<sup>+</sup>11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- [Res] Grand View Research. Data science platform market size, 2020-2027.
- [SKS19] Abraham Silberschatz, Henry Korth, and S Sudarshan. *Database System Concepts*. McGraw-Hill, New York, NY, 7 edition, April 2019.
- [SL20] Cristian Santesteban and Shayne Longpre. How big data confers market power to big tech: Leveraging the perspective of data science. *The Antitrust Bulletin*, 65(3):459–485, 2020.
- [TH14] Souhaib Ben Taieb and Rob J Hyndman. A gradient boosting approach to the kaggle load forecasting competition. *International journal of forecasting*, 30(2):382–394, 2014.
- [Van16] Jake VanderPlas. *Python data science handbook: Essential tools for working with data*. “O’Reilly Media, Inc.”, 2016.
- [W.62] Tukey J. W. The future of data analysis,. *The Annals of Mathematical Statistics*, 33:1, 1962.
- [YZT<sup>+</sup>18] Xulei Yang, Zeng Zeng, Sin G Teo, Li Wang, Vijay Chandrasekhar, and Steven Hoi. Deep learning for practical image recognition: Case study on kaggle competitions. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 923–931, 2018.