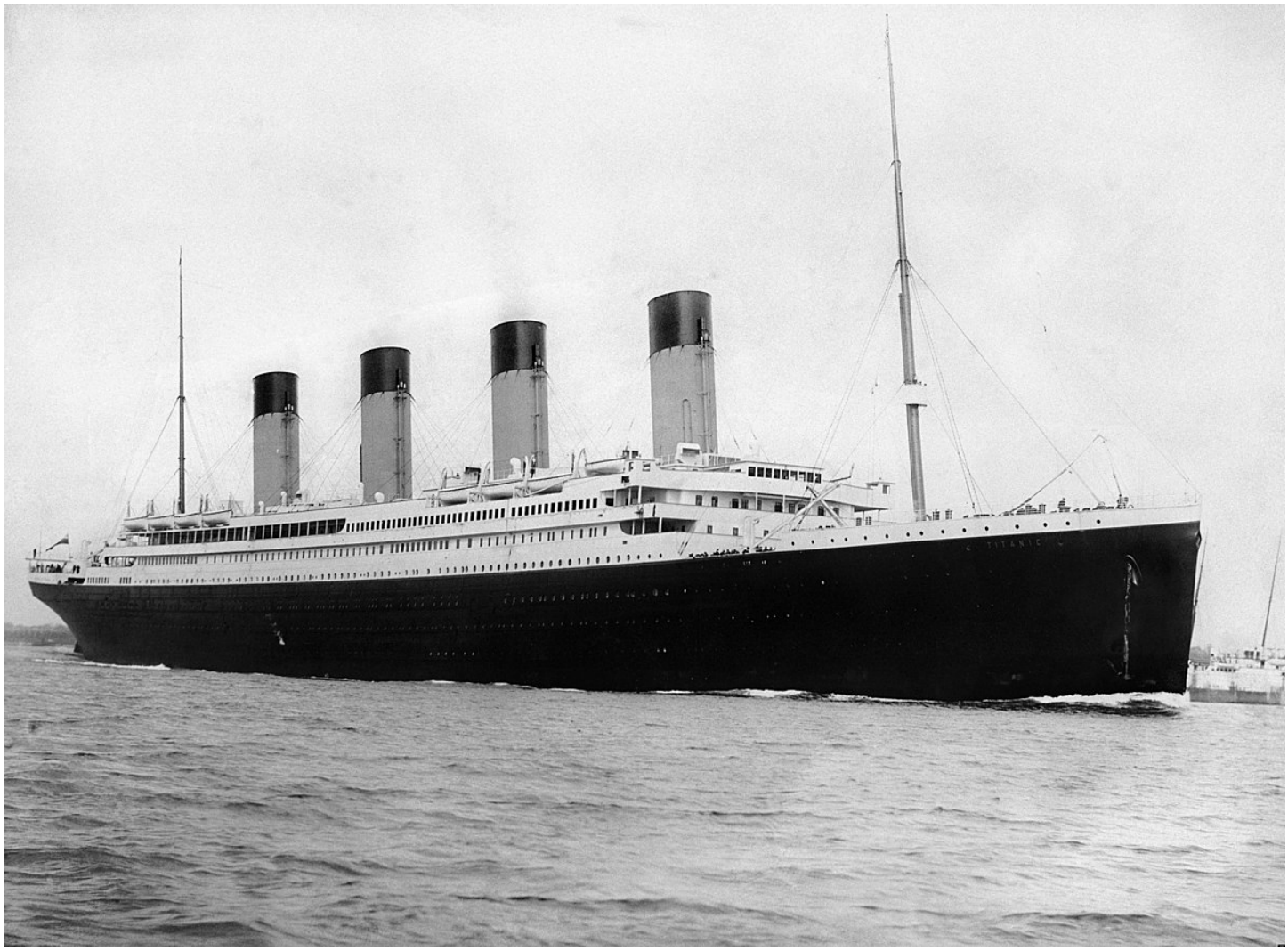
PUC-Rio - BI-Master - DM
Relatório de Data Mining
Trabalho de Conclusão da Disciplina
Professora Manoela Kohler

Camilo Lopes Costa
João Gabriel Evelin d'Oliveira

11.04.2021



Introdução:
Este trabalho tem por objetivo realizar a análise exploratória, tratamento de *missing*, identificar atributos desnecessários realizar balanceamento dos atributos utilizados assim como verificar os modelos de classificação aplicados à base.

Para este trabalho, a base utilizadas foi a do Titanic (fonte: `https://www.kaggle.com/c/titanic/`. Nesta base foram aplicados modelos de classificação, tais como Regressão Logística, KNN (*K-Nearest Neighbors*), SVM (*Support Vector Machine*), Árvore de Decisão e Random Forest. Cada um destes modelos serão descritos mais abaixo.

Para realização destes procedimentos foram utilizados Python e o RapidMiner.

Legenda dos atributos:

survival - Sobreviveu: 0 = No , 1 = Yes.

pclass - Classe da passagem: 1 = 1st, 2 = 2nd, 3 = 3rd.

sex - Sexo: Male/Female.

age - Idade: Número inteiro em anos.

sibsp - Número de irmãos/noivo(a) : Número inteiro.

parch - Número de parentes e filhos: Número inteiro.

ticket - Número da passagem fare - Custo da passagem: Número inteiro.

cabin - Cabine/Quarto.

embarked - Porto embarcado: C = Cherbourg, Q = Queenstown, S = Southampton.

1. **Análise exploratória, atributos desnecessários e *missing values***

- **Análise Exploratória (Python)**:
  Antes de realizarmos qualquer tipo de ação, devemos importar as bibliotecas básicas que serão utilizadas ao longo do códigos. Para isso utilizamos as seguintes bibliotecas:
  - **pandas**: Biblioteca utilizada para criação de um *Dataframe* assim como alguns tratamentos do mesmo.
  - **numpy**: Biblioteca utilizada para operações numéricas e matemáticas.
  - **matplotlib.pyplot** e **seaborn**: Bibliotecas para plotagem gráficos.

  Para conseguirmos fazer uma análise mais ampla, foi utilizado a biblioteca "*Profiling*"pois ela retorna uma análise de todos os atributos assim como missing, relação, entre outros. Com isso teremos o código inicial e um overview dos atributos conforme abaixo.

```
# %% Importacao das bibliotecas padrao
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# %% Importacao da base de dados e criacao do Dataframe
df = pd.read_csv('./titanic/train.csv')
df.head()
# %% Analise Exploratoria
import pandas_profiling as pp
pp.ProfileReport(df=df)
```

Com o resultado obtido através do profiling (Figura 1), podemos ver que há 7 variáveis categóricas e 6 númericas além da quantidade de variáveis, se há ou não duplicatas, entre outros. O atributo que será monitorado, ou seja, nossa saída, é a variável "Survived".

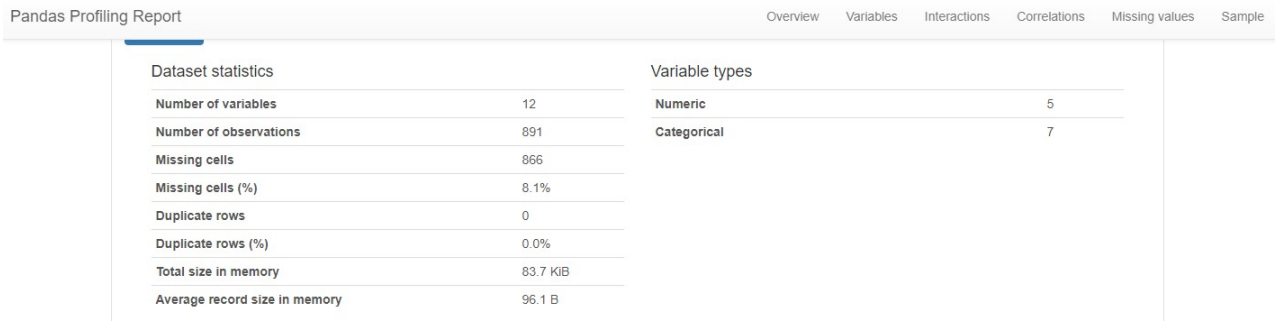


Figura 1: Overview obtido do Pandas Profiling

- **Atributos Desnecessários (Python)**
  Analisando todos os atributos do dataframe, iremos utilizar como parâmetros de entrada apenas: gênero (homem/mulher), classe (1/2/3) e idade pois, conforme pequisa realizada, foram os principais atributos para determinação dos primeiros a serem embarcados nos botes salva-vidas.
- **Missing Values (Python)**
  Agora, analisando a quantidade de missing dos atributos que iremos utilizar na aplicação dos modelos (Figura 2), podemos observar que o atributo 'Idade' (Age) possui 177 missing, totalizando 714 registros. Sendo assim, foram realizados duas abordagens: excluindo as linhas que possuem o missing neste parâmetro do dataframe e outra substituindo pela média das idades. A primeira abordagem foi uma forma rápida porém não é muito bom realizar esse tratamento uma vez que a quantidade de missing equivale a, aproximadamente, 20% do total da base.

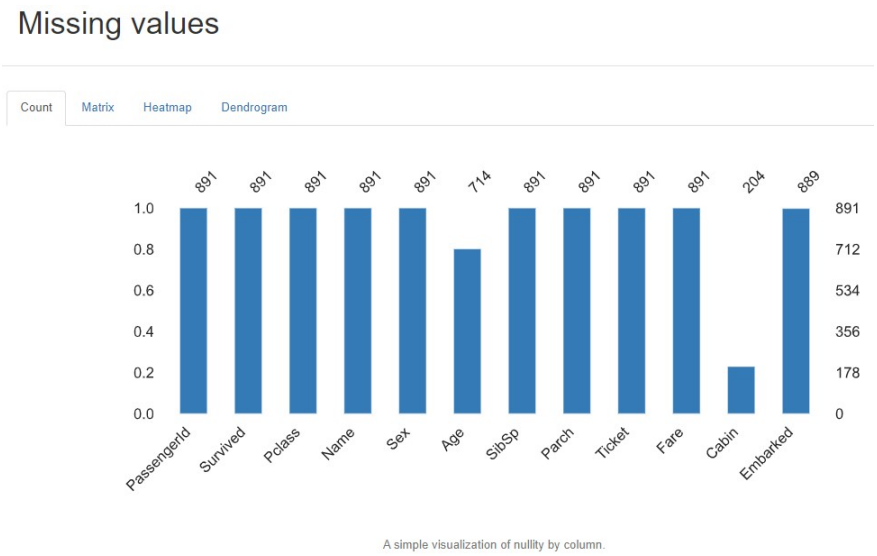


Figura 2: Missing Values - Profiling

Para exclusão, foi utilizado o trecho de código abaixo e obtemos as quantidade de 'NaN' nos atributos de entrada (X).

```
# %% Separando em Entrada e Saida
X = df.iloc[:,[2,4,5]]
y = df.Survived
# %% Verificacao de Missing
print(f"A entrada X possui missing? {X.isnull().values.any()}")
    True
print(f"Quantos? {X.isnull().sum().sum()}")
    177
print(f"Quais colunas possuem missing? \n {X.isnull().sum()}")
    Pclass      0
    Sex         0
    Age       177
# %% Exclusao das linhas com missing
df = df.dropna(subset=['Age'])
df = df.reset_index(drop=True)
```

2. **Balanceamento e Transformação de atributos categóricos**

- **Transformação:**
  Antes de realizarmos o balanceamento, deve-se transformar o atributo de gênero para valores binários (1/0). Para isso, separamos todo o dataset em saída e entradas, de forma "macro". Após essa divisão, substituímos o 'male/female' por '1/0' para facilitar o treino do modelo. Uma vez realizada essa substituição, essas bases de entrada e saída são dividades em bases de treino e testes.

```
# %% Separando em Entrada e Saida
X = df.iloc[:,[2,4,5]].values
y = df.Survived.values
# %% Alterando Genero
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,1] = le.fit_transform(X[:,1])
# %% Separando em base de treino e de teste
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ...
    random_state=0)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

- **Balanceamento:**
  Após realizar essa divisão, é realizado o escalonamento dos atributos de entrada. Primeiro é realizado o 'fit_transform', que é responsável por realizar o escalonamento de acordo com atributos passados e em seguida aplicar este escalonamento nestes atributos. Este processo é feito na base de treino da entrada (X_train) e, uma vez definido e aplicado o escalonamento nesta base, o mesmo escalonamento é aplicado na base de entrada de testes (X_test) através do 'transfom'.

```
# %% Escalonamento dos atributos
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

3. **Testando Modelos**

(a) **Python**
No Python, após determinarmos quais seriam os parâmetros de entrada e saída e realizar os devidos tratamentos descritos acima, foi executado o código para que verificasse a precisão de cada modelo. Para isso foi dividido em duas etapas por conta do tratamento de *missings* distintos. Na Tabela 1 são os resultados com a exclusão dos registros com *missing* enquanto na Tabela 2 apresenta os resultados obtidos pela substituição da média.

Modelo	**ExclusaoMissing**	**GridSearch**	**Acurácia**	**Kappa**	**F1**
Regressão Logística	X	-	0.846154	0.688884	0.828125
KNN	X	-	0.804196	0.600479	0.770492
KNN_GS	X	X	0.748252	0.493905	0.727273
SVM	X	-	0.776224	0.530763	0.692308
Árvore Decisão	X	-	0.846154	0.687959	0.825397
Árvore Decisão_GS	X	X	0.832168	0.659591	0.809524
Random Forest	X	-	0.790210	0.577007	0.769231

Tabela 1: Comparação do resultado dos modelos - Exclusão de *Missing*

Modelo	**ExclusaoMissing**	**GridSearch**	**Acurácia**	**Kappa**	**F1**
Regressão Logística	-	-	0.804469	0.586168	0.744526
KNN	-	-	0.821229	0.616446	0.757576
KNN_GS	-	X	0.821229	0.614328	0.753846
SVM	-	-	0.810056	0.592474	0.742424
Árvore Decisão	-	-	0.832402	0.636425	0.765625
Árvore Decisão_GS	-	X	0.793296	0.560138	0.725926
Random Forest	-	-	0.832402	0.636425	0.765625

Tabela 2: Comparação do resultado dos modelos - Substituição pela Média

Desta forma é possível observar que alguns resultados sofreram uma alteração significativa uma vez que a quantidade de registros alteraram e, desta forma, foi possível obter uma quantidade maior de treinamento do modelo e, consequentemente, um melhor resultado para alguns modelos.

(b) **Rapidminer**

Pela agilidade proporcionada pelo Rapidminer pudemos ser mais criativos na seleção de atributos. Primeiro selecionamos mais atributos e o exploramos na árvore de decisão. O atributo "Survived" foi escolhido como "label" pois nosso objetivo é prever os sobreviventes no acidente do Titanic. O nome do passageiro foi utilizado como identificação (id) e utilizamos sex, age, pclass, fare, sibsp, parch como atributos de entrada.

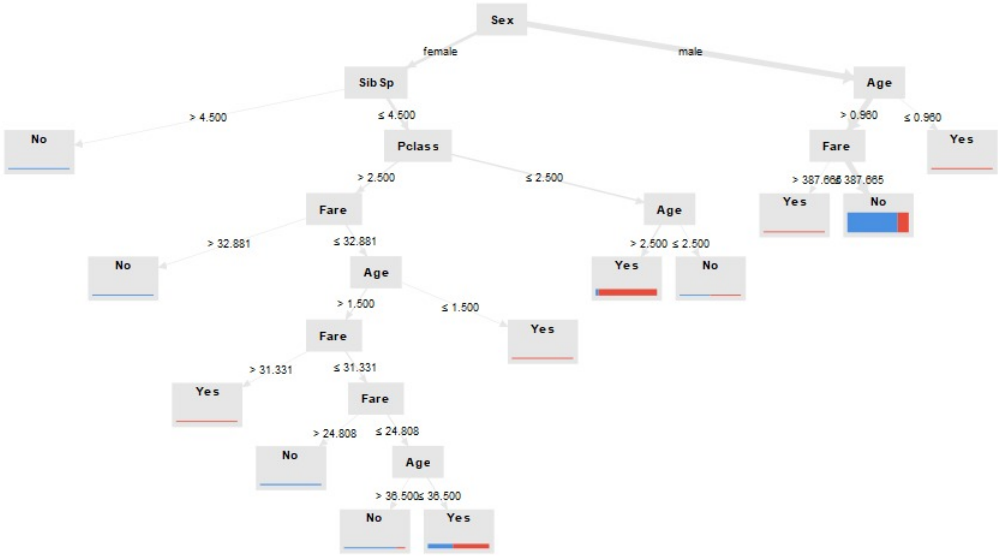


Figura 3: Árvore de Decisão

Segundo, visualizamos a árvore de decisão, isso nos permitiu perceber sutilezas não identificadas na primeira análise. Por exemplo, apesar de mulheres terem grandes chances de sobrevivência, esse não era o caso para mulheres que embarcaram na terceira classe ou que pagaram pouco na passagem e com muitos irmãos.

Por último, aplicamos o modelo treinado de Árvore de Decisão, Random Forest e KNN (k=8) pela natureza pois a natureza do problema de previsão de sobrevivência do Titanic é de classificação. Os resultados estão relatados abaixo.

accuracy: 79.02%

	true No	true Yes	class precision
pred. No	81	26	75.70%
pred. Yes	4	32	88.89%
class recall	95.29%	55.17%	

Figura 4: Precisão - Árvore de Decisão

accuracy: 73.03%

	true No	true Yes	class precision
pred. No	89	27	76.72%
pred. Yes	21	41	66.13%
class recall	80.91%	60.29%	

Figura 5: Precisão - KNN para k=8

accuracy: 87.41%

	true No	true Yes	class precision
pred. No	80	13	86.02%
pred. Yes	5	45	90.00%
class recall	94.12%	77.59%	

Figura 6: Precisão - Random Forest

Conclusão

Apesar das tentativas de aperfeiçoar o modelo pela seleção de atributos, ficou claro como o maior peso estiverem nos atributos de sexo, idade e preço da passagem.

Tanto no modelo em Python quanto no Rapidminer obtivemos uma acurácia entre 70% e 90%, com uma pequena diferença entre as duas formas devido a execução da biblioteca Python x RapidMiner. Enquanto o modelo que possui os melhores indicadores no Python foram a Árvor de Decisão e o Random Forest, na modelagem utilizando o RapidMiner o mais preciso foi o do Random Forest chegando a uma acurácia de 87%, ambos utilizando a substituição de missing pela média.

Considerações Finais

Ao longo do desenvolvimento do trabalho, foi possível obter um conhecimento mais prático do conteúdo apresentado em sala de aula assim como foi possível verificar a diferença entre as ferramentas utilizadas para realização da mesma análise.