Exercício 2 - Machine Learning

Data de entrega: 18/09/2020 - Nota: 10

Integrantes do Grupo

- João Marcelo
- Matheus Reis
- Matheus Sena
- Ygor Oliveira
- Thiago Costa

O RMS Titanic foi um navio de passageiros britânico que afundou no Oceano Atlântico Norte nas primeiras horas da manhã de 15 de abril de 1912, após colidir com um iceberg durante sua viagem inaugural de Southampton para a cidade de Nova York. Havia cerca de 2.224 passageiros e tripulantes a bordo do navio, e mais de 1.500 morreram, tornando-o um dos desastres marítimos comerciais mais mortíferos em tempos de paz da história moderna. O RMS Titanic era o maior navio à tona na época em que entrou em serviço e foi o segundo de três transatlânticos da classe Olímpica operados pela White Star Line. O Titanic foi construído pelo estaleiro Harland and Wolff em Belfast. Thomas Andrews, seu arquiteto, morreu no desastre.

In [1]:
```
# linear algebra
import numpy as np

# data processing
import pandas as pd

# data visualization
import seaborn as sns
from matplotlib import pyplot as plt

# algorithms
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC, SVC
```

Dicas

O aluno pode selecionar qual o tipo de modelo ele gostaria de implementar para descobrir de forma preditiva quantas pessoas em um determinado modelo poderá aumentar os sobreviventes.

OBS: Já tem alguns imports que pode ajudar em qual modelo usar.

Carregando dados

In [2]:
```
train_df = pd.read_csv("./dataset/train.csv")
test_df = pd.read_csv("./dataset/test.csv")
```

In [3]:
```
print(f'{"TRAIN":.^40}')
train_df.info()

print(f'\n{"TEST":.^40}')
test_df.info()
```

```
.................TRAIN.................
<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

.................TEST.................
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

Removendo colunas (características) que não serão utilziadas no modelo

In [4]:
```
train_df.drop(['PassengerId', 'Name', 'Cabin', 'Ticket'], axis=1, inplace=True)
test_df.drop(['PassengerId', 'Name', 'Cabin', 'Ticket'], axis=1, inplace=True)
```

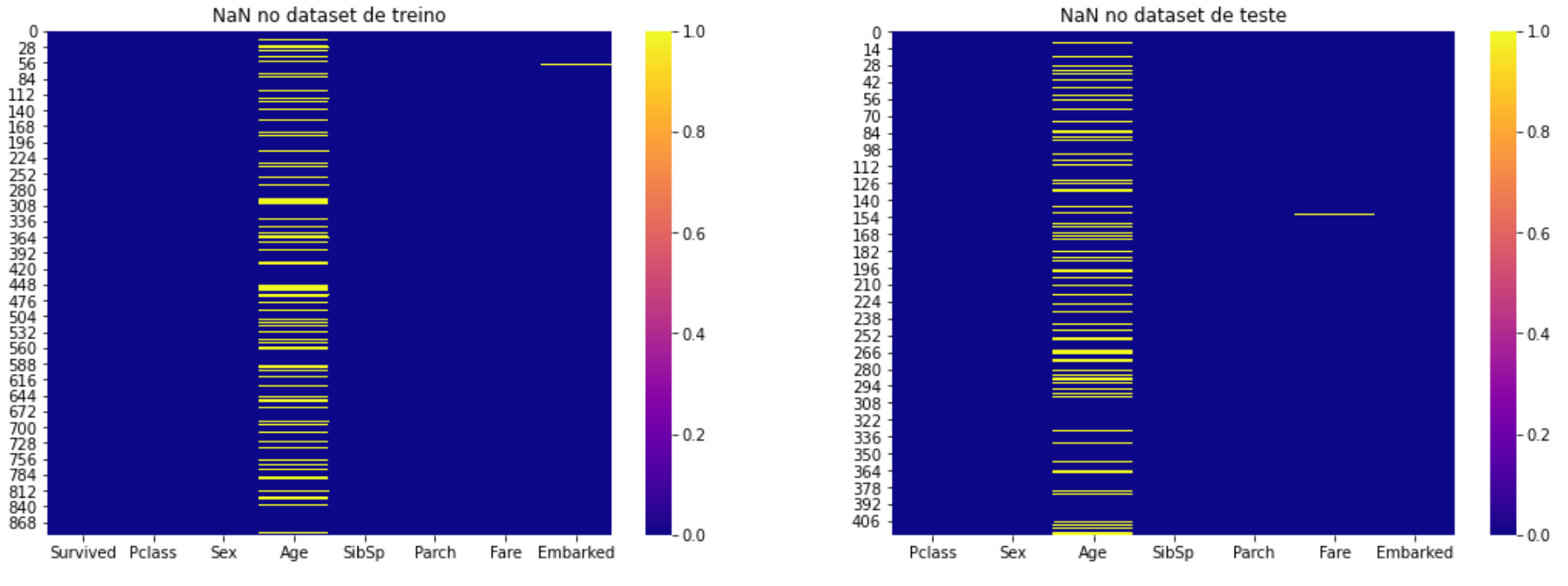
Tratamento de dados

In [5]:
```
# NaN no dataset
fig, axs = plt.subplots(1, 2, figsize=(18, 6))

axs[0].set_title('NaN no dataset de treino')
sns.heatmap(train_df.isna(), cmap='plasma', ax=axs[0])

axs[1].set_title('NaN no dataset de teste')
sns.heatmap(test_df.isna(), cmap='plasma', ax=axs[1])
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbc402f1160>



In [6]:
```
# preenchendo os NaN de Age de acordo com a idade média das pessoas na mesma Pclass
for i in range(1, 4):
    train_pclass = train_df[train_df['Pclass'] == i]['Age']
    train_pclass_age_na = train_df[(train_df['Pclass'] == i) & (train_df['Age'].isna())]

    test_pclass = test_df[test_df['Pclass'] == i]['Age']
    test_pclass_age_na = test_df[(test_df['Pclass'] == i) & (test_df['Age'].isna())]

    train_df.loc[train_pclass_age_na.index, 'Age'] = int(train_pclass.mean())
    test_df.loc[test_pclass_age_na.index, 'Age'] = int(test_pclass.mean())
else:
    del train_pclass, train_pclass_age_na, test_pclass, test_pclass_age_na
```

In [7]:
```
# removendo registros com Fare e Embarked nulos (removendo pois tem poucos dados nulos) (3 linhas no total)
train_df.dropna(axis=0, inplace=True)
test_df.dropna(axis=0, inplace=True)
```

In [8]:
```
# dados em string para dados numérios (one hot encode)

# female, male == 0, 1
train_df['Sex'] = train_df['Sex'].apply(lambda sex: 0 if sex == 'female' else 1)
test_df['Sex'] = test_df['Sex'].apply(lambda sex: 0 if sex == 'female' else 1)

# S, C, Q == 0, 1, 2
train_df['Embarked'] = train_df['Embarked'].apply(lambda emb: 0 if emb == 'S' else 1 if  emb == 'C' else 2)
test_df['Embarked'] = test_df['Embarked'].apply(lambda emb: 0 if emb == 'S' else 1 if emb == 'C' else 2)
```

Separação dos dados de treino e de teste

In [9]:
```
# características e classificações
X = train_df.drop('Survived', axis=1)
Y = train_df['Survived']
```

In [10]:
```
# dados de treino e test
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=101)
```

Modelo

In [11]:
```
classificador = LogisticRegression()
classificador.fit(x_train, y_train)
```

Out[11]: LogisticRegression()

In [12]:
```
pred_test = classificador.predict(x_test)
```

Métricas

In [13]:
```
print(f'Acurácia: {accuracy_score(y_test, pred_test) * 100:.2f}%')
```

```
Acurácia: 83.15%
```

Predição dos dados no dataset de test ¶

In [14]:
```
predicao = classificador.predict(test_df)
```

In [15]:
```
N_SOBREVIVENTES = predicao.sum()
N_NAO_SOBREVIVENTES = len(predicao)
print(f'Quantidade de pessoas que sobreviveram: {N_SOBREVIVENTES} de {N_NAO_SOBREVIVENTES}')
```

```
Quantidade de pessoas que sobreviveram: 151 de 417
```

In [16]:
```
plt.title('Dados do dataset de test')
plt.pie([N_SOBREVIVENTES, N_NAO_SOBREVIVENTES], shadow=True, labels=['Survived', 'Not survived'])

plt.show()
```