IA - Regressão

Prof. MSc. Bruno Santos
email: bruno.santos@saojudas.br

AVALIAÇÃO A3

DOCUMENTAÇÃO DO PROJETO:

- **0. Definição do problema;**
- **1. Obtenção dos dados;**
- **2. Análise exploratória dos dados;**
- **3. Preparação dos dados*;**
- **4. Modelagem*;**
- **5. Avaliação: Acurácia, RMSE*;**
- 6. Metodologia de montagem do comitê;
- 7. Resultados obtidos pelo comitê;
- 8. Discussão dos resultados obtidos; e
- 9. Conclusão.

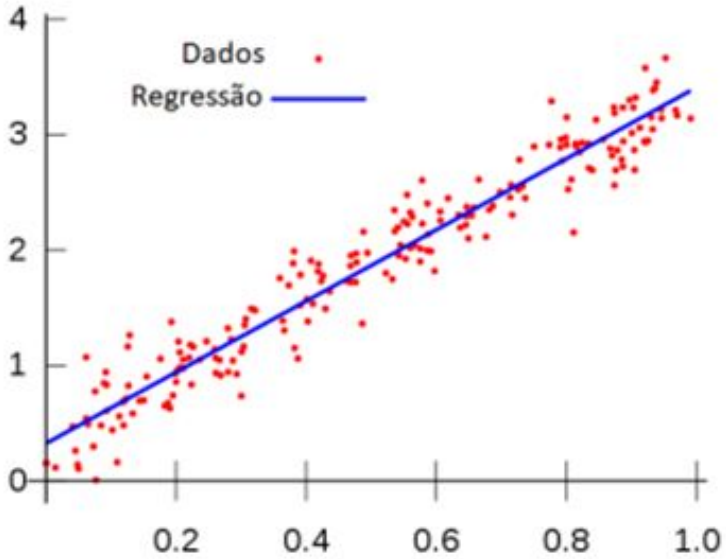
* Para todas as IAs escolhidas.

AVALIAÇÃO A3

- Apresentação final (7/12) = total de 15 pontos; e
- Expo USJT (~11-15) = total de 10 pontos.
- 15 pontos para quem fizer o curso e obter o certificado de **Microsoft AI fundamentals** (https://docs.microsoft.com/pt-br/learn/certifications/exams/dp-900) ou **IBM Watson Academy Badges** (https://www.ibm.com/training/badge/d7937817-0a3e-461f-af8d-7e381b6b03e8).

REVISÃO TEÓRICA - Regressão Linear

O objetivo da regressão linear é encaixar uma linha na distribuição, mais próxima da maioria dos pontos do conjunto de dados, minimizando os erros, ou seja, a distância entre os pontos e a reta.



PROJETO COM REGRESSÃO LINEAR

Objetivo: Utilizaremos a regressão linear para prever o valor de um imóvel com base no valor de imóveis semelhantes na mesma região.

Para esta aula, trabalharemos com os dados de habitação disponibilizados pela Kaggle.

https://www.kaggle.com/bumba5341/advertisingcsv

0. DEFINIÇÃO DO PROBLEMA - USA-Housing-Dataset

Sua vizinha é uma corretora de imóveis e deseja ajuda para prever os preços da habitação em regiões nos EUA. Seria ótimo se você pudesse, de alguma forma, criar um modelo para ela que lhe permitisse incluir algumas características de uma casa e retornar uma estimativa de preço que a casa venderia.

Ela perguntou se você poderia ajudá-la com suas novas habilidades em ciência de dados. Você diz que sim e decide que a regressão linear pode ser um bom caminho para resolver esse problema!

Sua vizinha fornece algumas informações sobre várias casas em regiões dos Estados Unidos, estão todas no conjunto de dados: USA_Housing.csv.

1. OBTENÇÃO DOS DADOS

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import seaborn as sns
5  %matplotlib inline
```

```
7  # lendo o arquivo USA_Housing.csv
8  df = pd.read_csv('USA_Housing.csv')
9
10 # Exibindo as colunas
11 df.columns
```

1. OBTENÇÃO DOS DADOS

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

O dataframe contém as seguintes colunas:

- 'Avg. Area Income': A renda média dos residentes da cidade onde a casa está localizada.
- 'Avg. Area House Age': Idade média das casas na mesma cidade.
- 'Avg. Area Number of Rooms': Número Médio de Cômodos para Casas na mesma cidade.
- 'Avg. Area Number of Bedrooms': Número Médio de Quartos para Casas na mesma cidade.
- 'Area Population': a população da cidade onde está localizada as Casas.
- 'Price': preço em que a casa foi vendida.
- 'Address': Endereço da casa.

2. ANÁLISE EXPLORATÓRIA DOS DADOS

```
13  # verificando as primeiras instâncias
14  df.head()
15  # identificar o tipo de cada variável
16  df.dtypes
```

```
18  # excluir a coluna endereço
19  df.drop('Address',axis=1,inplace=True)
20  # Normalizando o dataframe
21  normalized_df=(df-df.min())/(df.max()-df.min())
22  print(normalized_df)
```

2. ANÁLISE EXPLORATÓRIA DOS DADOS

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

```
Avg. Area Income                float64
Avg. Area House Age             float64
Avg. Area Number of Rooms       float64
Avg. Area Number of Bedrooms    float64
Area Population                 float64
Price                           float64
Address                          object
dtype: object
```

2. ANÁLISE EXPLORATÓRIA DOS DADOS

	Avg. Area Income	Avg. Area House Age	...	Area Population	Price
0	0.686822	0.441986	...	0.329942	0.425210
1	0.683521	0.488538	...	0.575968	0.607369
2	0.483737	0.468609	...	0.528582	0.425192
3	0.506630	0.660956	...	0.491549	0.507384
4	0.469223	0.348556	...	0.376988	0.250702
...	...	...	...	...	...
4995	0.475738	0.754359	...	0.326351	0.425683
4996	0.675097	0.633450	...	0.366362	0.597881
4997	0.507135	0.670026	...	0.476515	0.413672
4998	0.558419	0.420389	...	0.611282	0.482127
4999	0.530715	0.486997	...	0.667088	0.523011

[5000 rows x 6 columns]

3. PREPARAÇÃO DOS DADOS

Os dados já estão prontos para serem usados, dispensando um pré-processamento.

4. MODELAGEM

Variáveis preditoras:

- Avg. Area Income
- Avg. Area House Age
- Avg. Area Number of Rooms
- Avg. Area Number of Bedrooms
- Area Population

Variável de resposta:

- Price

4. MODELAGEM - Aplicando o Modelo de Regressão Linear

Vamos agora começar a treinar o modelo de regressão!

Primeiro, precisamos dividir nossos dados em uma matriz X que contém os recursos para treinamento e uma matriz y com a variável de destino, neste caso a coluna Preço.

Jogaremos fora a coluna Endereço, pois ela possui apenas informações de texto que o modelo de regressão linear não pode usar.

4. MODELAGEM - Aplicando o Modelo de Regressão Linear

```
24  #Matriz X e y -> X - variáveis preditoras, y - variável de resposta
25  X = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms','Avg. Area Number of Bedrooms', 'Area Population']]
26  y = df['Price']
27
28  X.head()
29  y.head()
```

4. MODELAGEM - Aplicando o Modelo de Regressão Linear

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
0	79545.458574	5.682861	7.009188	4.09	23086.800503
1	79248.642455	6.002900	6.730821	3.09	40173.072174
2	61287.067179	5.865890	8.512727	5.13	36882.159400
3	63345.240046	7.188236	5.586729	3.26	34310.242831
4	59982.197226	5.040555	7.839388	4.23	26354.109472

```
0    1.059034e+06
1    1.505891e+06
2    1.058988e+06
3    1.260617e+06
4    6.309435e+05
Name: Price, dtype: float64
```

4. MODELAGEM - Aplicando o Modelo de Regressão Linear

```
32  #Dividindo os dados em Treino e Teste
33  from sklearn.model_selection import train_test_split
34  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=50)
35
36  #Criando e Treinando o Modelo
37  from sklearn.linear_model import LinearRegression
38
39  # Aqui criamos o modelo preditor
40  modelopreditor = LinearRegression()
41
42  # Aqui ocorre o aprendizado
43  modelopreditor.fit(X_train,y_train)
44
45  # Imprimindo os coeficientes com os nomes das colunas
46  coeff = pd.DataFrame(modelopreditor.coef_,X.columns,columns=['Coeficiente'])
47  coeff
```

4. MODELAGEM - Aplicando o Modelo de Regressão Linear

	Coeficiente
Avg. Area Income	21.525435
Avg. Area House Age	166415.114396
Avg. Area Number of Rooms	119802.717039
Avg. Area Number of Bedrooms	1783.166098
Area Population	15.387540

- O aumento de 1 unidade na Avg. Area Income aumenta o preço em 21,57 USD.
- O aumento de 1 unidade na Avg. Area House Age aumenta o preço em 166.552,47 USD.
- O aumento de 1 unidade na Avg. Area Number of Rooms aumenta o preço em 119.512.53 USD.
- O aumento de 1 unidade na Avg. Area Number of Bedrooms aumenta o preço em 2.758.95 USD.
- O aumento de 1 unidade na Area Population aumenta o preço em 15.29 USD.

5. AVALIAÇÃO

```
49  # fazendo predições com os dados de teste
50  predictions = modelopreditor.predict(X_test)
51
52  # imprimindo as primeiras variáveis preditoras
53  X_test.head()
54
55  # Predições realizadas
56  predictions
57
58  # Valores reais para acertar
59  y_test.head()
```

5. AVALIAÇÃO

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
3431	50570.864807	5.828143	4.851423	4.36	40580.092291
2042	65314.720112	6.918945	6.245656	2.42	36565.029831
79	64419.252638	6.954422	8.516160	6.16	39318.170755
4663	60390.502855	5.195406	8.368913	6.27	37921.720586
3640	73068.518101	7.271422	5.685408	3.14	42929.876157

```
array([ 631305.0584258 , 1231990.56161572, 1539664.80551512, ...,
       1251986.62330105, 1036432.95159371, 1553580.80137947])
```

```
3431    6.129387e+05
2042    1.347083e+06
79      1.492011e+06
4663    1.223915e+06
3640    1.368692e+06
Name: Price, dtype: float64
```

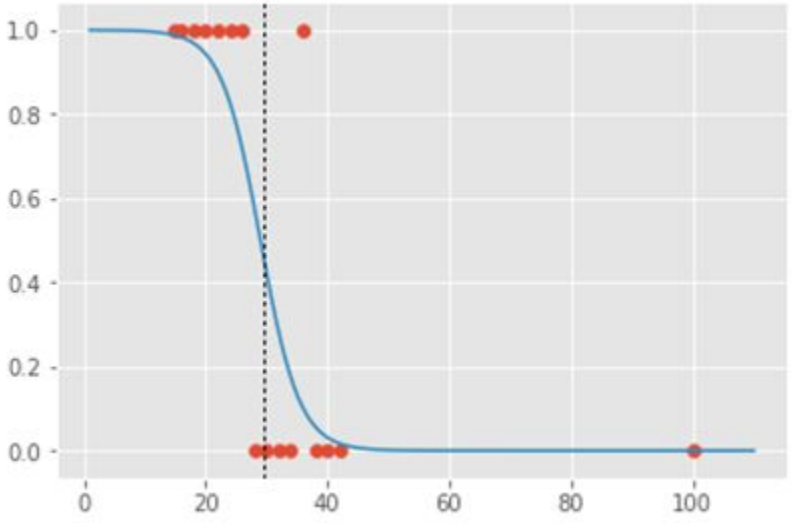
5. AVALIAÇÃO

```
61  from sklearn import metrics
62  print('MAE:', metrics.mean_absolute_error(y_test, predictions))
63  print('MSE:', metrics.mean_squared_error(y_test, predictions))
64  print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 80728.9338454242
MSE: 10077066685.875519
RMSE: 100384.5938671643
```

REVISÃO TEÓRICA - Regressão Logística

A regressão logística é um recurso que nos permite estimar a probabilidade associada à ocorrência de determinado evento em face de um conjunto de variáveis explanatórias.



PROJETO COM REGRESSÃO LOGÍSTICA

Objetivo: Utilizar os dados disponíveis para medir a probabilidade de sobrevivência dos passageiros do Titanic.

Para esta aula, trabalharemos com os dados do RMS (Royal Mail Ship) Titanic disponibilizados pela Kaggle. Titanic Data Set from Kaggle.

https://www.kaggle.com/c/titanic

0. DEFINIÇÃO DO PROBLEMA - RMS Titanic Dataset

Vamos criar um modelo de predição para classificar os passageiros como: sobrevivente ou morto, no naufrágio do Titanic, ocorrido entre 14 a 15 de abril de 1912. Neste naufrágio morreram 1502 pessoas de um total de 2224 passageiros.

Aqueles que sobreviveram à tragédia contaram com muita sorte, mas será que alguns grupos de passageiros não tiveram “mais sorte” que outros grupos? Será que crianças e mulheres realmente tiveram mais chances de sobreviver? Será que o "Jack" teve menos chances de escapar do seu destino trágico que a "Rose", só pelo fato de ele ter embarcado na 3ª Classe?

1. OBTENÇÃO DOS DADOS

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import seaborn as sns
5  %matplotlib inline
```

```
 7  # lendo titanic_data.csv
 8  td = pd.read_csv('titanic_data.csv')
 9
10  # Exibindo as colunas
11  td.columns
```

1. OBTENÇÃO DOS DADOS

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

Entendendo o significado dos dados:

- PassengerId: Número de identificação do passageiro
- Survived: Informa se o passageiro sobreviveu ao desastre (0 = Não e 1 = Sim)
- Pclass: Classe do bilhete (1 = 1ª Classe, 2 = 2ª Classe e 3 = 3ª Classe)
- Name: Nome do passageiro
- Sex: Sexo do passageiro
- Age: Idade do passageiro
- SibSp: Quantidade de cônjuges e irmãos a bordo
- Parch: Quantidade de pais e filhos a bordo
- Ticket: Número da passagem
- Fare: Preço da Passagem
- Cabin: Número da cabine do passageiro
- Embarked: Porto no qual o passageiro embarcou
- (C = Cherbourg, Q = Queenstown e S = Southampton)

2. EXPLORAÇÃO DOS DADOS

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

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

3. PREPARAÇÃO DOS DADOS

Como qualquer conjunto de dados do mundo real, você vai se deparar sempre com dados que não servem para nada e outros que não tem peso ou significância nenhuma no seu modelo.

Muitas vezes nosso julgamento pode ser equivocado, mas, infelizmente, é papel seu, como cientista de dados, escolher quais features serão usadas para o modelo de Machine Learning.

No nosso caso, vamos desconsiderar as variáveis, pois aparentemente não parecem relevantes: **PassengerId; Name; Ticket; e Cabin.**

3. PREPARAÇÃO DOS DADOS

```
19  td.columns
20
21  # Excluindo as colunas ['PassengerId', 'Name', 'Ticket', 'Cabin']
22  td.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'],axis=1,inplace=True)
23
24  # exibindo as colunas (features) relevantes que sobraram
25  td.columns
```

3. PREPARAÇÃO DOS DADOS

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
Index(['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',
       'Embarked'],
      dtype='object')
```

3. PREPARAÇÃO DOS DADOS

```
27  # contando os valores faltantes
28  td.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

3. PREPARAÇÃO DOS DADOS

Normalmente, há duas abordagens mais utilizadas quando a gente encontra missing values (valores faltantes):

- Preencher esses valores arbitrariamente (média, mediana, valor mais frequente);
- Excluir a linha inteira;

3. PREPARAÇÃO DOS DADOS

Cada caso é um caso e novamente você, cientista de dados, é quem vai tomar a decisão sobre qual passo seguir. Na maioria das vezes, não é desejável jogar informação de uma linha inteira só por causa de um campo faltando. Sempre que possível é melhor você preencher o campo, e é isso que vamos fazer.

- Para a variável Age (idade), vamos colocar a idade média por Classe;
- Para a variável Fare (tarifa), vamos colocar o valor da mediana; e
- Para a variável do Embarked (porto de embarque), vamos colocar o valor com maior frequência.

3. PREPARAÇÃO DOS DADOS

```
32  # Exibindo a média por classe
33  print (td.groupby('Pclass'). mean())
```

```
35  # criando função para inserir idade faltante
36  def insert_age(cols):
37      Age = cols[0]
38      Pclass = cols[1]
39
40      if pd.isnull(Age):
41          if Pclass == 1:
42              return 38
43          elif Pclass == 2:
44              return 30
45          else:
46              return 25
47      else:
48          return Age
```

...

3. PREPARAÇÃO DOS DADOS

```
50  # aplicando a função aos dataFrame
51  td['Age'] = td[['Age','Pclass']].apply(insert_age,axis=1)
52  td['Age'].median()
```

```
54  # Fare
55  fare_median = td['Fare'].median()
56  td['Fare'].fillna(fare_median, inplace=True)
```

```
58  # embarked
59  embarked_top = td['Embarked'].value_counts()[0]
60  td['Embarked'].fillna(embarked_top, inplace=True)
```

3. PREPARAÇÃO DOS DADOS

Pclass	Survived	Age	SibSp	Parch	Fare
1	0.629630	38.233441	0.416667	0.356481	84.154687
2	0.472826	29.877630	0.402174	0.380435	20.662183
3	0.242363	25.140620	0.615071	0.393075	13.675550

```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S
5	0	3	male	25.0	0	0	8.4583	Q
6	0	1	male	54.0	0	0	51.8625	S
7	0	3	male	2.0	3	1	21.0750	S
8	1	3	female	27.0	0	2	11.1333	S
9	1	2	female	14.0	1	0	30.0708	C

3. PREPARAÇÃO DOS DADOS

```
#Precisamos converter os dados das colunas Sex e Embarked para valores numéricos.
#Isso é feito usando variáveis dummy em pandas.

# exibindo os tipos de dados
td.info()

# criando o dataframe sex com a coluna 'Sex' convertida para valores numéricos
sex = pd.get_dummies(td['Sex'],drop_first=True)
print (sex)

# criando o dataframe embark com a coluna 'Embarked' convertida para valores numéricos
embark = pd.get_dummies(td['Embarked'],drop_first=True)
print (embark)

# Eliminando as colunas Sex e Embarked
td.drop(['Sex','Embarked'],axis=1,inplace=True)

# concatendo df com sex e embark
td = pd.concat([td,sex,embark],axis=1)

td.head()
td.info()
```

3. PREPARAÇÃO DOS DADOS

```
[891 rows x 3 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Age       891 non-null    float64
 3   SibSp     891 non-null    int64
 4   Parch     891 non-null    int64
 5   Fare      891 non-null    float64
 6   male      891 non-null    uint8
 7   C         891 non-null    uint8
 8   Q         891 non-null    uint8
 9   S         891 non-null    uint8
dtypes: float64(2), int64(4), uint8(4)
memory usage: 45.4 KB
```

4. MODELAGEM

```
#Vamos separar nosso dataframe em dois conjuntos: teste e treino
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(td.drop('Survived',axis=1), td['Survived'], test_size=0.40, random_state=101)
```

```
#Treinando
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression(solver='lbfgs',max_iter=1000)
logmodel.fit(X_train,y_train)
```

```
#Fazendo predições
predictions = logmodel.predict(X_test)
predictions
```

4. MODELAGEM

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

5. AVALIAÇÃO

```
101  #Usando classification report vamos verificar: precision, recall, f1-score
102  from sklearn.metrics import classification_report
103  print(classification_report(y_test,predictions))
```

```
105  #Imprimindo a Matriz de Confusão
106  from sklearn.metrics import confusion_matrix
107  conf_mat = confusion_matrix(y_test, predictions)
108  print(conf_mat)
```

5. AVALIAÇÃO

```
              precision    recall  f1-score   support

           0       0.79      0.87      0.83       207
           1       0.80      0.67      0.73       150

    accuracy                           0.79       357
   macro avg       0.79      0.77      0.78       357
weighted avg       0.79      0.79      0.79       357

[[181  26]
 [ 49 101]]
```

Você sobreviveria ao Titanic?

EXEMPLO:

- Pclass: 1 (viajando na primeira classe);
- Age: 25 (anos);
- SibSP: 0 (acompanhado da esposa);
- Parch: 0 (não acompanhado dos filhos);
- Fare: 350.00 (valor da passagem);
- male: 1 (sexo masculino); e
- CQS 001 (embarcado no porto de Southampton).

Você sobreviveria ao Titanic?

```
111  # exemplo
112  EXEMPLO = np.array([1,25,0,0,350,1,0,0,1]).reshape((1,-1))
113
114  # resultado: 0 = não sobreviveu, 1 = sobreviveu
115  print("EXEMPLO: {}".format(logmodel.predict(EXEMPLO)[0]))
```

```
EXEMPLO: 1
```