# Programação Orientada a Objetos para Dados

**POO para análise exploratória de dados**

Prof. Ms. Leonardo Rocha

- Unidade de Ensino: 4
- Competência da Unidade: Compreender a estruturação de classes para tratamento de dados.
- Resumo: Conceito de classe, construção e utilização para tratamento de dados.
- Palavras-chave: classe, métodos, bibliotecas, dados.
- Título da Teleaula: POO para análise exploratória de dados
- Teleaula nº: 4

## Contextualização

Classes e métodos
Estrutura de classes
Bibliotecas para tratamento de dados
Utilizando uma classe em Python

## Machine learning

## O que é?

Consiste em um sub-campo da ciência da computação que evoluiu a partir da identificação de padrões. Sugere-se que, com os padrões identificados através dos dados, é possível, com auxília de algoritmos, criar meios para que uma máquina aprenda algo. Dai o termo Aprendizagem de máquina ou inteligência Artificial.
Utiliza-se de conhecimentos matemáticos e estatísticos para cumprir esse propósito.

## Dia-a-dia

Quando você vai assistir filmes via streaming, encontra diversos títulos recomendados para você de acordo com seu perfil de interesse, essas recomendações foram feitas utilizando Machine Learning. Quando você entra na sua caixa de e-mail e percebe que algumas mensagens foram automaticamente classificadas como spam, adivinhe só? Isso também foi feito utilizando Machine Learning. Existem diversos outros exemplos nos mais variados temas.

## Campos de uso

A utilização de machine Learning é ampla. Está em diversas áreas atualmente.

Biotecnologia por exemplo: criação de um algoritmo para identificar através de análise de dados de fita simples, um vírus como SARS-CoV-2 e apresentar uma vacina em pouquíssimo tempo.

## Possibilidades

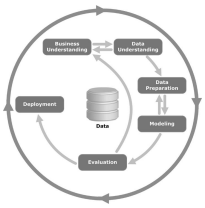
**Prever as vendas dos próximos meses** para adequar o estoque da empresa de forma que não haja muitas sobras e nem falte produto;

**Segmentar os clientes** de acordo com suas características para potencializar a estratégia de comunicação de uma empresa, adequando sua linguagem a cada perfil de cliente;

**Recomendar produtos** para os clientes de acordo seu perfil de compra, potencializando as vendas;

**Criar um chatbot** para automatizar parte do atendimento aos clientes, reduzindo o volume de trabalho dos call centers;

## Processo para mineração de dados




Cross-industry standard process for data mining. In: Wikipédia: a enciclopédia livre. Disponível em: https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining . Acesso em: 17 abr 2021.

## Datasets

## Dataset

Para análise de conjunto de dados é necessário utilizar o que chamamos de dataset. Existem datasets de várias áreas, por exemplo, biologia.

Vamos baixar um dataset disponível em:

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

## Iniciando a análise

```
>>> import pandas as pd
>>> df = pd.read_csv('/tmp/train.csv')
>>> df
     PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0              1         0       3  ...   7.2500   NaN         S
1              2         1       1  ...  71.2833   C85         C
2              3         1       3  ...   7.9250   NaN         S
3              4         1       1  ...  53.1000  C123         S
4              5         0       3  ...   8.0500   NaN         S
..           ...       ...     ...  ...      ...   ...       ...
886          887         0       2  ...  13.0000   NaN         S
887          888         1       1  ...  30.0000   B42         S
888          889         0       3  ...  23.4500   NaN         S
889          890         1       1  ...  30.0000  C148         C
890          891         0       3  ...   7.7500   NaN         Q

[891 rows x 12 columns]
```

## Conhecendo o dataset

**PassengerId:** Um identificador único para cada passageiro.

**Survived:** A resposta que queremos prever. 1 = Sobreviveu, 2 = Não sobreviveu.

**Pclass:** Identifica a classe do bilhete do passageiro. 1 = Primeira Classe, 2 = Segunda Classe, 3 = Terceira Classe. Esta variável tem grande relação com o status socioeconômico do passageiro. Primeira Classe = Alto, Segunda Classe = Médio, Terceira Classe = Baixo.

**Name:** Nome do passageiro.

**Sex:** Sexo do passageiro.

## Conhecendo o dataset

**Age:** Idade em anos. Se a idade do passageiro for menor que 1, então a idade terá um valor fracional (ex.: 0.75). Se a idade do passageiro for estimada, então ela terá o formato XX.5 (ex.: 35.5 para uma idade estimada de 35 anos)

**SibSp:** Quantidade de irmãos e cônjuges do passageiro que também estão abordo.

**Parch:** Quantidade de pais e filhos do passageiro que também estão abordo.

**Ticket:** Número do bilhete do passageiro.

**Fare:** Valor da tarifa paga pelo passageiro.

**Cabin:** Número da cabine do passageiro.

**Embarked:** Porto onde o passageiro embarcou. C = Cherbourg, Q = Queenstown, S = Southampton

## Analisando

O dataset precisa ser analisado. Para treinar um algoritmo, é preciso que todos os dados sejam válidos. Campos vazidos precisam ser descarados. vamos olhar:

```
In [4]: df.isna().sum() / df.shape[0]

Out[4]: PassengerId    0.000000
        Survived       0.000000
        Pclass         0.000000
        Name           0.000000
        Sex            0.000000
        Age            0.198653
        SibSp          0.000000
        Parch          0.000000
        TIcket         0.000000
        Fare           0.000000
        Cabin          0.771044
        Embarked       0.002245
        dtype: float64
```

## O que fazer?

Note que 3 campos apresentam dados vazios:

| | | |
|---|---|---|
| Age | 0.198653 | 19,8% |
| Cabin | 0.771044 | 77,1% |
| Embarked | 0.002245 | -1% |

É preciso tomar decisão sobre o que fazer. Dá para descartar ou para inserir dados. Como Cabin é o campo que mais está prejudicado. Vamos descartá-lo. Os demais, vamos imputar dados.

## Imputando dados

Para o campo Age, vamos descobrir a mediana de idade com o seguinte comando:

```
In [5]: df['Age'].median()

Out[5]: 28.0
```

A idade tem uma grande relação com o sexo e a classe do passageiro. Assim vamos agrupar os dados usando o groupby()

## Imputando dados

Utilizando o groupby() para agrupar os dados por classe (Pclass) e sexo (Sex). Teremos:

```
In [15]: df_idade = df.groupby(['Pclass', 'Sex'], as_index=False)['Age'].median()
         df_idade

Out[15]:
```

| | Pclass | Sex | Age |
|---|---|---|---|
| 0 | 1 | female | 35.0 |
| 1 | 1 | male | 40.0 |
| 2 | 2 | female | 28.0 |
| 3 | 2 | male | 30.0 |
| 4 | 3 | female | 21.5 |
| 5 | 3 | male | 25.0 |

## Como encontrar média de idade por sexo?

### Solução

```
>>> df_idade_media = df.groupby(['Sex'], as_index=False)['Age'].mean()
>>> df_idade_media
      Sex        Age
0  female  27.915709
1    male  30.726645
```

## Dúvidas

## Gráficos

### Dados para gráficos

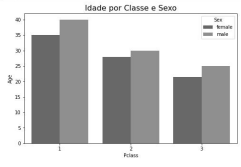
Agora que organizamos nossos dados e agrupamos todos corretamente, vamos trabalhar com gráficos para plotagem desses dados.

Para isso, vamos importar a biblioteca matplotlib e, em seguida, vamos montar a figura do gráfico para exibir os dados tratados.

### Gerando gráfico

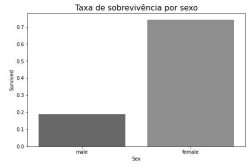
```
In [7]: 1 import matplotlib.pyplot as plt
        2 import seaborn as sns

In [8]: 1 fig, ax = plt.subplots(figsize=(8, 5))
        2 sns.barplot(x='Pclass', y='Age', hue='Sex', data=df_idade)
        3 ax.set_title('Idade por Classe e Sexo', fontsize=16)
        4 plt.show()
```



## Sobrevivência por sexo

```
In [19]: 1 df_plot = df.groupby('Sex', as_index=False)['Survived'].mean()
         2 fig, ax = plt.subplots(figsize=(6, 5))
         3 sns.barplot(x='Sex', y='Survived', data=df_plot)
         4 ax.set_title("Taxa de sobrevivência por sexo", fontsize=16)
         5 ax.set_xticklabels(['male', 'female'])
         6 plt.show()
```




## Gráficos complexos

## Ampliando a análise

Agora que entendemos como tratar os dados e manipulá-los afim de gerar gráficos e informação. Vamos fazer análises com gráficos de maneira um pouco mais complexa. Vamos cruzar mais dados para obter mais informações.

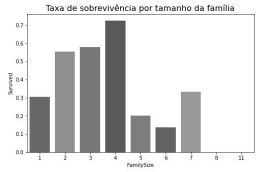
## Ampliando a análise

Vamos tentar encontrar a resposta para a seguinte dúvida:

Qual é a relação entre o tamanho da família e a taxa de sobrevivência? Sobrevive em maior número as famílias com maior ou menor quantidade de pessoas?

## Vamos à prova

```
In [24]: 1 df_plot = df.groupby('FamilySize', as_index=False)['Survived'].mean()
         2 fig, ax = plt.subplots(figsize=(8, 5))
         3 sns.barplot(x='FamilySize', y='Survived', data=df_plot)
         4 ax.set_title("Taxa de sobrevivência por tamanho da família", fontsize=16)
         5 plt.show()
```




## Classe para tratamento de dados

## POO na análise

Utilizar POO não só organiza nosso código evitando que nós façamos algo repetidamente mas também otimiza o esforço e nos possibilita reutilizar esse código criado.

Agora, vamos montar uma classe para tratar os dados com os quais trabalhamos até aqui. Veja:

## Classe DataPrep

DataPrep será o nome da nossa classe. Ela terá a seguinte estrutura:

| **DataPrep** |
| --- |
| +data |
| +tratar_nulos()<br>+tratar_variaveis_categóricas()<br>+criar_variaveis()<br>+remover_variaveis()<br>+ normalizar_dados()<br>+separar_treino_teste()<br>+preparar_dados() |

## Implementando métodos

Para iniciar, é preciso instalar e importar algumas bibliotecas do Python. São elas:

sklearn

matplotlib

pandas

## Classe disponível para download

bit.ly/classPython_dataprep

## Estrutura

```
import pandas as pd
from sklearn.model_selection import train_test_split

class DataPrep:
    def __init__(self, data: pd.DataFrame) -> None:
        "Inicializa o classe DataPrep com base nos dados do dataset fitanic."
        self.data = data

    def remover_variaveis(self) -> None:
        "Remove as variáveis que não serão utilizadas pelo modelo."
        colunas_para_remover = [
            'PassengerId',
            'Name',
            'Ticket',
            'Cabin'     #Variável com quantidade elevada de dados ausentes
        ]
        self.data.drop(columns=colunas_para_remover, inplace=True)

    def tratar_nulos(self) -> None:
        "Trata as variáveis nulas, imputando valores adequados"
        #Imputa mediana das idades por classe e sexo
        self.data['Age'] = self.data.groupby(['Pclass', 'Sex'])['Age'] \
            .apply(lambda x: x.fillna(x.median()))

        #Imputa local de embarque
        self.data['Embarked'] = self.data['Embarked'].fillna('S')
```

## Estrutura

```
def tratar_variaveis_categoricas(self) -> None:
    "Trata as variáveis categóricas"
    #Label encoding da variável Sex
    sexo = {'male': 0, 'female': 1}
    self.data['Sex'] = self.data['Sex'].map(sexo)

    #One Hot Encoding da variável Embarked
    embarked_dummies = pd.get_dummies(self.data['Embarked'])
    self.data = pd.concat([self.data, embarked_dummies], axis=1)

def normalizar_dados(self) -> None:
    variaveis = self.data.drop(columns='Survived')
    var_cols = variaveis.columns
    resposta = self.data['Survived']

    scaler = MinMaxScaler()
    variaveis = scaler.fit_transform(variaveis)

def criar_variaveis(self) -> None:
    # Soma a quantidade de irmãos e conjuges,
    # a quantidade de pais e filhos,
    # e +1 para considerar o próprio passageiro
    self.data['FamilySize'] = self.data['SibSp'] + self.data['Parch'] + 1
```

## Estrutura

```
    def separar_treino_teste(self) -> tuple[pd.DataFrame, pd.DataFrame]:
        "Separa a base de dados entre conjunto de treinamento e teste"
        treino, teste = train_test_split(self.data, test_size=0.3, random_state=2021)
        return treino, teste

    def preparar_dados(self) -> tuple[pd.DataFrame, pd.DataFrame]:
        "Executa todas as etapas de tranformação de dados"
        self.tratar_nulos()
        self.tratar_variaveis_categorias()
        self.criar_variaveis()
        self.remover_variaveis()
        treino, teste = self.separar_treino_teste()
        return treino, teste
```

## Como executar a classe para tratamento dos dados?

## Solução

```
>>> import pandas as pd
>>> from dataprep import DataPrep
>>> df = pd.read_csv('/tmp/train.csv')
>>> dp = DataPrep(df)
>>> df_treino, df_teste = dp.preparar_dados()
>>> df_treino.head()
     Survived  Pclass  Sex   Age  SibSp  ...  Embarked  C  Q  S  FamilySize
725         0       3    0  20.0      0  ...         S  0  0  1           1
52          1       1    1  49.0      1  ...         C  1  0  0           2
883         0       2    0  28.0      0  ...         S  0  0  1           1
344         0       2    0  36.0      0  ...         S  0  0  1           1
450         0       2    0  36.0      1  ...         S  0  0  1           4

[5 rows x 12 columns]
```

## Dúvidas

## Recapitulando

- Classes e métodos em Python
- Importação de bibliotecas para construção de classe
- Estrutura de métodos
- Convocação de classe para tratamento de dados