# Aula 4 - Pré processamento de dados

UNIFESP

Universidade Federal de São Paulo

1933

# AGENDA

- Motivação
- Tratando dados faltantes
- Variáveis categóricas
- Normalização

# Motivação

- A etapa de pré-processamento dos dados é importante pois a qualidade do resultado obtido por meio dos algoritmos de ML dependerá da qualidade dos dados de entrada
- Na grande maioria dos casos os dados coletados contém certos problemas que podem prejudicar o resultado almejado
    - Diferença de escala entre os atributos numéricos
    - Exemplos atributos faltantes
    - Atributos textuais

# Tratando dados faltantes

- Alguns dataset podem conter linhas dados com atributos sem valor
- Existem algumas técnicas que podem ser utilizadas para contornar esses defeitos
    - Excluir a coluna do atributo - pode ocasionar grande perda de informações
    - Excluir as linhas com atributos faltantes - perda de exemplos
    - Preencher os valores faltantes com a média

# Tratando dados faltantes - Excluir coluna

- Recomenda-se a utilização dessa opção quando o atributo exercer pouca influência sobre o conjunto de dados.
- Para a execução desta opção basta se utilizar a função *df.dropna(axis=1, inplace=True)* da biblioteca Pandas

# Tratando dados faltantes - Excluir coluna

- No exemplo abaixo a coluna “property tax” do dataset foi removida

```
dataset = pd.read_csv('houses_to_rent_v2.csv')
```

```
[37] dataset.head(5)
```

| | Unnamed: 0 | Unnamed: 0.1 | city | area | rooms | bathroom | parking spaces | floor | animal | furniture | hoa | rent amount | property tax | fire insurance | total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 1 | 240 | 3 | 3 | 4 | - | acept | furnished | R$0 | R$8,000 | NaN | R$121 | R$9,121 |
| **1** | 1 | 1 | 0 | 64 | 2 | 1 | 1 | 10 | acept | not furnished | R$540 | R$820 | R$122 | R$11 | R$1,493 |
| **2** | 2 | 2 | 1 | 443 | 5 | 5 | 4 | 3 | acept | furnished | R$4,172 | R$7,000 | NaN | R$89 | R$12,680 |
| **3** | 3 | 3 | 1 | 73 | 2 | 2 | 1 | 12 | acept | not furnished | R$700 | R$1,250 | R$150 | R$16 | R$2,116 |
| **4** | 4 | 4 | 1 | 19 | 1 | 1 | 0 | - | not acept | not furnished | R$0 | R$1,200 | R$41 | R$16 | R$1,257 |

```
[38] dataset.dropna(axis=1, inplace=True)
```

```
[39] dataset.head()
```

| | Unnamed: 0 | Unnamed: 0.1 | city | area | rooms | bathroom | parking spaces | floor | animal | furniture | hoa | rent amount | fire insurance | total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 1 | 240 | 3 | 3 | 4 | - | acept | furnished | R$0 | R$8,000 | R$121 | R$9,121 |
| **1** | 1 | 1 | 0 | 64 | 2 | 1 | 1 | 10 | acept | not furnished | R$540 | R$820 | R$11 | R$1,493 |
| **2** | 2 | 2 | 1 | 443 | 5 | 5 | 4 | 3 | acept | furnished | R$4,172 | R$7,000 | R$89 | R$12,680 |
| **3** | 3 | 3 | 1 | 73 | 2 | 2 | 1 | 12 | acept | not furnished | R$700 | R$1,250 | R$16 | R$2,116 |
| **4** | 4 | 4 | 1 | 19 | 1 | 1 | 0 | - | not acept | not furnished | R$0 | R$1,200 | R$16 | R$1,257 |

# Tratando dados faltantes - Excluir linhas

- Recomenda-se a utilização dessa opção quando existem poucas linhas com dados faltantes no *dataset*
- Esta operação pode ser realizada utilizando-se a função *df.dropna(axis=0, inplace=True)*

# Tratando dados faltantes - Excluir linhas

- No exemplo abaixo foram removidas as linhas que possuíam pelo menos um atributo com valor faltante

```
[42] dataset.head()
```

| | Unnamed: 0 | Unnamed: 0.1 | city | area | rooms | bathroom | parking spaces | floor | animal | furniture | hoa | rent amount | property tax | fire insurance | total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 1 | 240 | 3 | 3 | 4 | - | acept | furnished | R$0 | R$8,000 | NaN | R$121 | R$9,121 |
| **1** | 1 | 1 | 0 | 64 | 2 | 1 | 1 | 10 | acept | not furnished | R$540 | R$820 | R$122 | R$11 | R$1,493 |
| **2** | 2 | 2 | 1 | 443 | 5 | 5 | 4 | 3 | acept | furnished | R$4,172 | R$7,000 | NaN | R$89 | R$12,680 |
| **3** | 3 | 3 | 1 | 73 | 2 | 2 | 1 | 12 | acept | not furnished | R$700 | R$1,250 | R$150 | R$16 | R$2,116 |
| **4** | 4 | 4 | 1 | 19 | 1 | 1 | 0 | - | not acept | not furnished | R$0 | R$1,200 | R$41 | R$16 | R$1,257 |

```
[43] dataset.dropna(axis=0, inplace=True)
```

```
[44] dataset.head()
```

| | Unnamed: 0 | Unnamed: 0.1 | city | area | rooms | bathroom | parking spaces | floor | animal | furniture | hoa | rent amount | property tax | fire insurance | total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 0 | 64 | 2 | 1 | 1 | 10 | acept | not furnished | R$540 | R$820 | R$122 | R$11 | R$1,493 |
| **3** | 3 | 3 | 1 | 73 | 2 | 2 | 1 | 12 | acept | not furnished | R$700 | R$1,250 | R$150 | R$16 | R$2,116 |
| **4** | 4 | 4 | 1 | 19 | 1 | 1 | 0 | - | not acept | not furnished | R$0 | R$1,200 | R$41 | R$16 | R$1,257 |
| **5** | 5 | 5 | 1 | 13 | 1 | 1 | 0 | 2 | acept | not furnished | R$0 | R$2,200 | R$42 | R$28 | R$2,270 |
| **10** | 10 | 10 | 0 | 60 | 1 | 1 | 0 | 6 | acept | not furnished | R$480 | R$720 | R$20 | R$10 | R$1,230 |

# Tratando dados faltantes - Preenchendo com a média

- Nesta solução, menos drástica que as anteriores, os valores faltantes podem ser preenchidos com o valor médio do atributo no *dataset*

```
[27] dataset = pd.read_csv('houses_to_rent_v3.csv')
     #calcula a média da coluna 'property tax'
     property_tax_mean = dataset['property tax'].mean()
     #atribui a média calculada para todos o NaN
     dataset['property tax'].fillna(property_tax_mean,inplace = True)
```

# Variáveis categóricas

- Determinados atributos podem ser do tipo nominal sendo necessário convertê-los para valores numéricos
- Isso pode ser feito alterando-se o tipo da coluna para *category* e utilizar o código das categorias

```
[67] dataset = pd.read_csv('houses_to_rent_v3.csv')
     dataset["animal"] = dataset["animal"].astype('category')
     dataset["animal_numerico"]  = dataset["animal"].cat.codes
     dataset
```

| | Unnamed: 0 | Unnamed: 0.1 | Unnamed: 0.1.1 | city | area | rooms | bathroom | parking spaces | floor | animal | furniture | hoa | rent amount | property tax | fire insurance | total | animal_numerico |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 1 | 240 | 3 | 3 | 4 | - | acept | furnished | 0 | 8000 | NaN | 121 | 9121 | 0 |
| **1** | 1 | 1 | 1 | 0 | 64 | 2 | 1 | 1 | 10 | acept | not furnished | 540 | 820 | 122.0 | 11 | 1493 | 0 |
| **2** | 2 | 2 | 2 | 1 | 443 | 5 | 5 | 4 | 3 | acept | furnished | 4172 | 7000 | NaN | 89 | 12680 | 0 |
| **3** | 3 | 3 | 3 | 1 | 73 | 2 | 2 | 1 | 12 | acept | not furnished | 700 | 1250 | 150.0 | 16 | 2116 | 0 |
| **4** | 4 | 4 | 4 | 1 | 19 | 1 | 1 | 0 | - | not acept | not furnished | 0 | 1200 | 41.0 | 16 | 1257 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **6075** | 6075 | 6075 | 6075 | 1 | 50 | 2 | 1 | 1 | 2 | acept | not furnished | 420 | 1150 | 0.0 | 15 | 1585 | 0 |
| **6076** | 6076 | 6076 | 6076 | 1 | 84 | 2 | 2 | 1 | 16 | not acept | furnished | 768 | 2900 | 63.0 | 37 | 3768 | 1 |
| **6077** | 6077 | 6077 | 6077 | 0 | 48 | 1 | 1 | 0 | 13 | acept | not furnished | 250 | 950 | 42.0 | 13 | 1255 | 0 |
| **6078** | 6078 | 6078 | 6078 | 1 | 160 | 3 | 2 | 2 | - | not acept | not furnished | 0 | 3500 | 250.0 | 53 | 3803 | 1 |
| **6079** | 6079 | 6079 | 6079 | 1 | 60 | 2 | 1 | 1 | 4 | acept | furnished | 489 | 1900 | 0.0 | 25 | 2414 | 0 |

# Normalização

- Alguns *datasets* possuem atributos numéricos que possuem diferentes escalas. Ex: idade, salário, altura, etc.
- Essa diferença de escala dificulta o treinamento dos modelos de ML, de modo que um atributo pode influenciar mais do que outro.
- Para solucionar esse problema aplica-se a normalização
- A biblioteca Scikit Learn fornece diferentes funções que podem ser aplicadas sobre os dados de acordo com suas características.
- MinMaxScaler, StandardScaler, RobustScaler

# Normalização - MinMaxScaler

- Fórmula = ( valor — Min) / (Max — Min)
- Reescala os valores para um intervalo entre 0 e 1 ou -1 e 1 caso existam valores negativos dentro do conjunto.
- É aplicada apenas dentro da coluna.
- Recomendada quando os dados não possuem distribuição normal e baixo desvio padrão.
- Sintaxe: *x_normalizado = MinMaxScaler().fit_transform(x)*

# Normalização - MinMaxScaler

- Exemplo:

```
[53] rent_amount = np.array(dataset['rent amount']).reshape(-1,1)

     from sklearn.preprocessing import MinMaxScaler

     rent_amount_normalizado  = MinMaxScaler().fit_transform(rent_amount)

     print("Dados originais:")
     print(rent_amount)
     print("Dados normalizados:")
     print(rent_amount_normalizado)
```

```
Dados originais:
[[8000]
 [ 820]
 [7000]
 ...
 [ 950]
 [3500]
 [1900]]
Dados normalizados:
[[0.1700314 ]
 [0.00897263]
 [0.14759982]
 ...
 [0.01188874]
 [0.06908928]
 [0.03319874]]
```

# Normalização - StandardScaler

- Fórmula: (valor — média) / desvio padrão
- Mais recomendado para dados com distribuição normal
- O resultado é um conjunto de dados com desvio padrão e variância iguais à 1
- Sintaxe: *x_normalizado = StandardScaler().fit_transform(x)*

# Normalização - StandardScaler

- Exemplo:

```
[55] rent_amount = np.array(dataset['rent amount']).reshape(-1,1)

     from sklearn.preprocessing import StandardScaler

     rent_amount_normalizado  = StandardScaler().fit_transform(rent_amount)

     print("Dados originais:")
     print(rent_amount)
     print("Dados normalizados:")
     print(rent_amount_normalizado)
```

```
Dados originais:
[[8000]
 [ 820]
 [7000]
 ...
 [ 950]
 [3500]
 [1900]]
Dados normalizados:
[[ 1.00776786]
 [-0.9998517 ]
 [ 0.72815511]
 ...
 [-0.96350204]
 [-0.25048952]
 [-0.69786992]]
```

# Normalização - RobustScaler

- Fórmula: (valor — média) / (Amplitude Interquartil)
- Recomendada para dados que possuem valores *outliers*
- Mantém os *outliers* porém devido a mudança de escala sua influência negativa é atenuada
- Sintaxe: *x_normalizado = RobustScaler().fit_transform(x)*

# Normalização - RobustScaler

- Exemplo:

```
[56] rent_amount = np.array(dataset['rent amount']).reshape(-1,1)

     from sklearn.preprocessing import RobustScaler

     rent_amount_normalizado  = RobustScaler().fit_transform(rent_amount)

     print("Dados originais:")
     print(rent_amount)
     print("Dados normalizados:")
     print(rent_amount_normalizado)
```

```
Dados originais:
[[8000]
 [ 820]
 [7000]
 ...
 [ 950]
 [3500]
 [1900]]
Dados normalizados:
[[ 1.17736303]
 [-0.55171583]
 [ 0.93654425]
 ...
 [-0.52040939]
 [ 0.09367851]
 [-0.29163155]]
```

# GRATIDÃO!

UNIFESP

UNIVERSIDADE FEDERAL DE SÃO PAULO

1933