Documento auxiliar – Como utilizar os pacotes mais importantes para as aulas práticas.

1. Entendendo a base de dados – Pandas

[...]. Geralmente, os datasets são armazenadas em arquivos com diferentes extensões, sendo os mais comuns '**csv**' e '**txt**'. Usando o Pandas, o arquivo pode ser carregado usando a função **read_csv**. Para as atividades práticas da disciplina, existem alguns argumentos que podem ser úteis, exemplificados no trecho do código abaixo.

```
import pandas as pd

dataset = pd.read_csv('class_breast.csv', header=None, sep = ',')
```

- **Header**: Nome das colunas. Caso não exista um nome na base de dados, é possível especificar (passando uma lista de ***strings***) ou ignorar (passando **None**).
- **Sep**: separador de colunas. Geralmente é default, mas algumas opções que podem ocorrer, dependendo da base, são ";" e " "(espaço).

	ID	radius	texture	perimeter	area	smoothness	compacity	concavity	pts_concavity	symmetry
0	842302	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419
1	842517	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812
2	84300903	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069
3	84348301	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597
4	84358402	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809
...	...	...	...	...	...	...	...	...	...	...
564	926424	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	926682	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	926954	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	927241	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	92751	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

569 rows × 32 columns

Figura 1 – Exemplo de base de dados 'Breast Cancer'

Nem todos os datasets são perfeitamente preenchidos. Na verdade, é bastante comum que existam problemas durante o processo de geração da base de dados que levem a erros de preenchimento, como erro de leitura do dado, formatos diferentes de dados ou até negligência durante a manipulação de dados.

Antes de conversarmos sobre algumas ideias que podem ser úteis para avaliar a base de dados, é interessante mostrar como podemos gerar melhores visualizações da tabela. Para acessar

algumas características específicas, basta passar para o ***DataFrame*** os nomes das colunas. Também podemos selecionar quais linhas queremos analisar, usando a propriedade **iloc**.

```
#Para selecionar colunas do dataset
sel_col = ['radius','perimeter','area']
dataset[sel_col]

#Para selecionar linhas
dataset.iloc[5:10,:]
```

1. Valores ausentes

A identificação de valores ausentes é bem simples, caso o **pandas** tenha reconhecido corretamente o caractere utilizado para representar a ausência de dados. O mais comum (default de reconhecimento de valor não preenchido pelo **pandas**) é o preenchimento vazio. Contudo, o sistema pode utilizar caracteres especiais, como '?' ou *strings* 'NA', 'NaN', que não são reconhecidos como valores ausentes. A Figura 2 mostra um exemplo da base de dados em que existem dados faltantes, mas não são reconhecidos adequadamente.

temperature of extremities	peripheral pulse	mucous membranes	capillary refill time	pain	peristalsis
3	3	?	2	5	4
?	?	4	1	3	4
1	1	3	1	3	3
4	1	6	2	2	4
?	?	6	2	?	?
...	...	...	...	...	...
4	?	4	2	2	4
3	2	4	2	4	3
4	3	4	1	4	4
3	3	3	1	3	3
?	?	?	?	?	?

Figura 2 – Tabela com valores ausentes não reconhecidos pelo pandas.

Para evitar este problema, podemos carregar a base de dados indicando quais strings representam estes dados faltantes, ou então utilizar o método replace. Estas duas estratégias alteram o conteúdo da tabela, como é possível observar na Figura 3.

```
df.replace('?',np.nan)
#Caso 2 - Leitura da base com a representação de NA values
df = pd.read_csv('https://raw.githubusercontent.com/jbrownlee/Datasets/ma
ster/horse-colic.csv',na_values='?')
```

temperature of extremities	peripheral pulse	mucous membranes	capillary refill time	pain	peristalsis
3.0	3.0	NaN	2.0	5.0	4.0
NaN	NaN	4.0	1.0	3.0	4.0
1.0	1.0	3.0	1.0	3.0	3.0
4.0	1.0	6.0	2.0	2.0	4.0
NaN	NaN	6.0	2.0	NaN	NaN
...	...	...	...	...	...
4.0	NaN	4.0	2.0	2.0	4.0
3.0	2.0	4.0	2.0	4.0	3.0
4.0	3.0	4.0	1.0	4.0	4.0
3.0	3.0	3.0	1.0	3.0	3.0
NaN	NaN	NaN	NaN	NaN	NaN

Figura 3 – Tabela com valores ausentes.

Com os valores ausentes já reconhecidos, podemos também identificar a proporção de dados ausentes, seja em relação aos atributos ou registros. Para isso, usamos o método isna, como mostrado abaixo.

```
Para os atributos
df3.isna().sum(axis=0)
#Para os registros
df3.isna().sum(axis=1)
```

2. Dados inconsistentes

Por mais que não seja comum, uma tabela pode conter dados preenchidos incorretamente. Problemas como preenchimento incorreto de formulários ou erros de aquisição do sensor, a presença de dados inconsistentes pode prejudicar o treinamento do modelo de Machine Learning. Para investigar se certo atributo contém valores condizentes, podemos visualizar algumas métricas estatísticas ou, em um primeiro momento, avaliar os valores únicos. Deste modo, conseguimos investigar se existem valores digitados incorretamente (e.g. letras em um atributo puramente numérico) ou "defeituosos" (e.g. sensor de temperatura medindo -273 ºC em um ambiente externo do Rio de Janeiro).

```
df[atributo].describe()
df[atributo].unique()
```

3. Tratamento da base de dados

Até o momento, sabemos apenas identificar os problemas na base de dados. Mas como resolvê-los? O **pandas** permite a manipulação da tabela para o tratamento de dados, seja no preenchimento de valores ou remoção de registros/atributos. Alguns comandos que podem ser úteis:

```
#Preenchimento simples
df.fillna(method='pad')
#Remoção de todos os valores nulos
df.dropna()
#Remoção de colunas específicas
df.drop(labels=['pulse'], axis=1)
```

2. Manipulando vetores e matrizes – Numpy

Na área de Ciência de Dados, a manipulação de vetores e matrizes é fundamental. Para a linguagem de programação Python, a biblioteca **Numpy** é a mais conhecida por implementar a representação e cálculos vetoriais e matriciais. Todas as operações mais conhecidas já estão implementadas no **Numpy**, basta apenas pesquisar como funciona. Para as funções que mostraremos a seguir, denotaremos **numpy** como **np** (usando **import numpy as np**) Alguns exemplos:

- Produto escalar: **np.dot**
- Concatenação: **np.concatenate**
- Valor máximo: **np.max**
- Argumento do valor máximo: **np.argmax**
- Tamanho da matriz: **np.shape**

Os valores das matrizes geralmente são acessados entre colchetes, onde o primeiro índice é 0 (diferente de outras linguagens que iniciam em 1, como o MATLAB). Portanto, se **x** for um vetor (**np.array**), o primeiro elemento é acessado como x[0]. Podemos acessar também uma parcela de valores do vetor, através de uma lista de índices ou por slices. Por exemplo, se **x** for um vetor com shape 10, podemos acessar os índices 0 e 2 da forma **x[[0, 2]]**. Caso queiramos manipular uma sequência de índices, podemos fazer uso de slices, denotados por dois pontos ":". Seguindo o exemplo anterior, se quisermos acessar os 4 primeiros índices do vetor **x**, podemos escrever simplesmente **x[:4]**.

Na prática, o uso de *slices* para acessar parte dos dados é bastante comum. Além disso, existem índices negativos em **Numpy**, funcionando como um caminho "cíclico" do vetor. Por exemplo, x[-1] indica o último valor do vetor **x**. Combinando estas duas ideias, podemos selecionar todos os valores de um vetor **x**, exceto o último, da forma **x[:-1]**.

A manipulação de matrizes segue a mesma ideia lógica de vetores. A maior diferença é que, no caso matricial, precisamos de dois índices para acessar um elemento. Sendo **X** uma matriz (**np.ndarray**), o primeiro elemento é acessado da forma **X[0,0]**.

3. Utilizando técnicas conhecidas – **Scikit-learn**

O **scikit-learn** é, provavelmente, a biblioteca mais conhecida na área de Ciência de Dados. Diferentes modelos de *Machine Learning*, técnicas de pré-processamento e manipulação de dados estão presentes no **scikit-learn**, proporcionando um maior acesso e divulgação de novas metodologias criadas nesta área. Neste documento, nos atentaremos a uma etapa específica, de pré-processamento e avaliação dos resultados. Nas aulas práticas, podemos utilizar as ferramentas citadas abaixo. Para cada função descrita, existe um tutorial do **scikit-learn** mostrando o passo-a-passo sobre a sua utilização.

- Pré-processamento
    - Codificação de atributos categóricos – **LabelEncoder**
    - Normalização min-max – **MinMaxScaler**
    - Normalização padrão – **StandardScaler**
    - Preenchimento de dados faltantes – **SimpleImputer, KNNImputer**

- Avaliação de resultados
    - Métricas de avaliação – **classification_report**
    - Matriz de Confusão – **confusion_matrix**
    - Resultado da acurácia – **accuracy_score**

- Seleção de modelos
    - Divisão da base de dados – **train_test_split**
    - K-Fold – **KFold**