# PREPROCESSING

Prof. Nielsen Rechia
nielsen.machado@uniritter.edu.br

# Dados (dataset)



**Colunas (M):** Atributos, características — Classe, Rótulo, Label

**Linhas (N):**
Instâncias
Objetos
Exemplos
Tuplas
Amostras
Casos
Registros

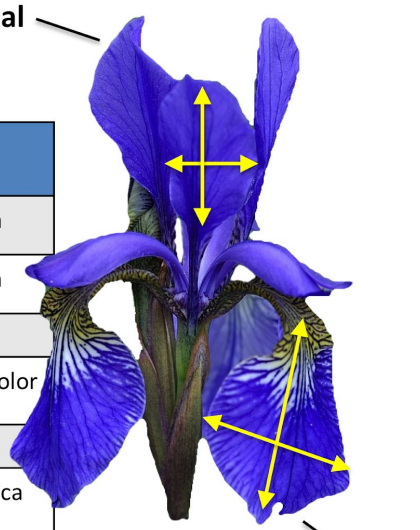
| | $\mathbf{X^1}$ | $\mathbf{X^2}$ | ... | $\mathbf{X^m}$ | $\mathbf{y}$ |
|---|---|---|---|---|---|
| $\mathbf{x}_1$ | $x_1^{(1)}$ | $x_1^{(2)}$ | ... | $x_1^{(m)}$ | $y_1$ |
| $\mathbf{x}_2$ | $x_2^{(1)}$ | $x_2^{(2)}$ | ... | $x_2^{(m)}$ | $y_2$ |
| . | . | . | . | | . |
| . | . | . | . | ... | . |
| . | . | . | . | | . |
| $\mathbf{x}_n$ | $x_n^{(1)}$ | $x_n^{(2)}$ | ... | $x_n^{(m)}$ | $y_n$ |

# Exemplo



**Samples**
(instances, observations)

| | Sepal length | Sepal width | Petal length | Petal width | Class label |
|---|---|---|---|---|---|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | Setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | Setosa |
| ... | | | | | |
| 50 | 6.4 | 3.5 | 4.5 | 1.2 | Versicolor |
| ... | | | | | |
| 150 | 5.9 | 3.0 | 5.0 | 1.8 | Virginica |




**Features**
(attributes, measurements, dimensions)

**Class labels**
(targets)

# Atributos



| ID | Nome | Temperatura | Enjôo | Mancha | Dor | Salário | Diagnóstico |
|---|---|---|---|---|---|---|---|
| 01 | Ana | 37.7 | sim | pequena | sim | 1000 | doente |
| 02 | Marcia | 37 | não | pequena | não | 1100 | saudável |
| 03 | José | 38.2 | sim | grande | não | 600 | saudável |
| 04 | Pedro | 39 | não | pequena | sim | 2000 | doente |
| 05 | Paulo | 37.3 | não | grande | sim | 1800 | saudável |
| 06 | Juliana | 37.7 | não | grande | sim | 900 | doente |

Nominal | Intervalar | Ordinal | Racional

# Exemplo



| | Tipo de Atributo | Descrição | Exemplo |
|---|---|---|---|
| Categórico (qualitativo) | Nominal | Valores são nomes diferentes, compara-se com = e ≠ | Estado Civil, CEP, CPF, Sexo, ... |
| | Ordinal | Valores possuem informações para ordenação, compara-se com =, ≠, > e < | Grau de Instrução, Número de endereço, patente militar, Grau de educação... |
| Numérico (quantitativo) | Intervalar | A diferença entre valores faz sentido. Existe unidade de medida com referência (zero) arbitrário. Compara-se com todas as operações anteriores incluindo + e - | Datas, temperatura em Fahrenheit |
| | Racional | A razão entre valores tem sentido (zero é absoluto). Compara-se com todas as anteriores e também com * e / | Contagens, Massa, Largura, Corrente Elétrica, Valores Monetários |

# Exemplo



**Atributos contínuos:**

Quantidade incontável de valores (número infinito de valores em um determinado intervalo.

ex: peso, temperatura, distância, tempo ...

**Atributos Discretos:**

Número contável de valores (se tem noção da quantidade de valores)

ex: cores elementares, n° de anos, n° de filhos ...

**Atributos Simétricos / Assimétricos:**

O valor indica que instância “não possui” / ”possui” determinada característica (valor do atributo “não importa” / “importa”)

# Exploração dos dados



**Entender as características dos datasets**

Inconsistências, ruídos, redundâncias, outliers, ausentes ...

**Permitir a definição das técnicas de pré-processamento ou aprendizado mais apropriadas**

Frequência dos dados, distribuição ou formato, Informações estatísticas

# Exploração dos dados



## Frequência

Proporção de vezes que um atributo assume um dado valor

Normalmente utilizada para a verificação de atributos categóricos

| ID | Nome | Temperatura | Enjôo | Mancha | Dor | Salário | Diagnóstico |
|---|---|---|---|---|---|---|---|
| 01 | Ana | 37.7 | sim | pequena | sim | 1000 | doente |
| 02 | Marcia | 37 | não | pequena | não | 1100 | saudável |
| 03 | José | 38.2 | sim | grande | não | 600 | saudável |
| 04 | Pedro | 39 | não | pequena | sim | 2000 | doente |
| 05 | Paulo | 37.3 | não | grande | sim | 1800 | saudável |
| 06 | Juliana | 37.7 | não | grande | sim | 900 | doente |

50% dos pacientes possuem mancha grande

(histograma)

# Exploração dos dados



**Medidas de localidade**

- Para dados categóricos:
  - Moda
- Para dados numéricos:
  - Média
  - Mediana
  - Percentil

# Exploração dos dados



Moda para o atributo Dor: Sim
Média de temperatura: 37.2
Mediana de temperatura: 37.7

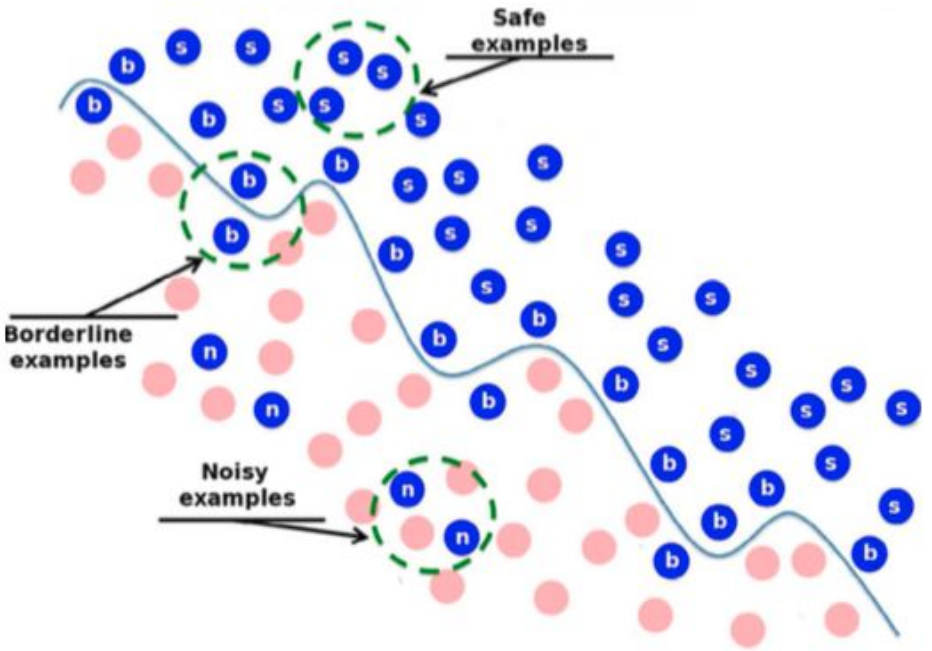
| ID | Nome | Temperatura | Enjôo | Mancha | Dor | Salário | Diagnóstico |
|---|---|---|---|---|---|---|---|
| 01 | Ana | 37.7 | sim | pequena | sim | 1000 | doente |
| 02 | Marcia | 37 | não | pequena | não | 1100 | saudável |
| 03 | José | 38.2 | sim | grande | não | 600 | saudável |
| 04 | Pedro | 39 | não | pequena | sim | 2000 | doente |
| 05 | Paulo | 37.3 | não | grande | sim | 1800 | saudável |
| 06 | Juliana | 37.7 | não | grande | sim | 900 | doente |

# Exploração dos dados



**Podemos ter:**

- ruídos
- ausentes
- inconsistentes
- redundantes
- outliers (exceções)



# Exploração dos dados



**Média**

facilmente calculada, porém é sensível a $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

**Mediana**

Menos sensível a outliers

Necessário ordenar valores

$$mediana(x) = \begin{cases} x_{(r+1)} & \text{se n é ímpar} \\ \frac{1}{2}(x_r + x_{(r+1)}) & \text{se n é par} \end{cases}$$

$r = \frac{n}{2}$ ⟶ divisão inteira

# Exercício



**Dado um atributo x = [1,2,3,4,5,80], calcular em Python:**

Média

Mediana

```
In [1]: import numpy as np

x = [1,2,3,4,5,80]

mean = np.mean(x)
print mean

median = np.median(x)
print median
```

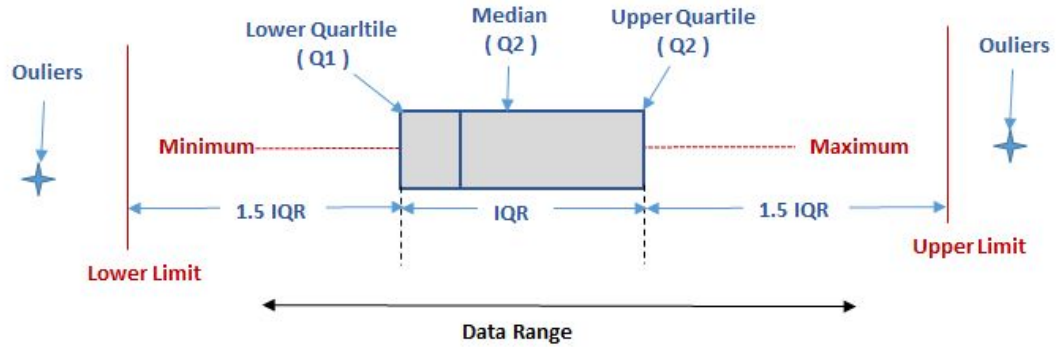
```
15.8333333333
3.5
```

# Distribuição dos dados



## Boxplot

Resumo das informações em gráfico



# Exploração dos dados



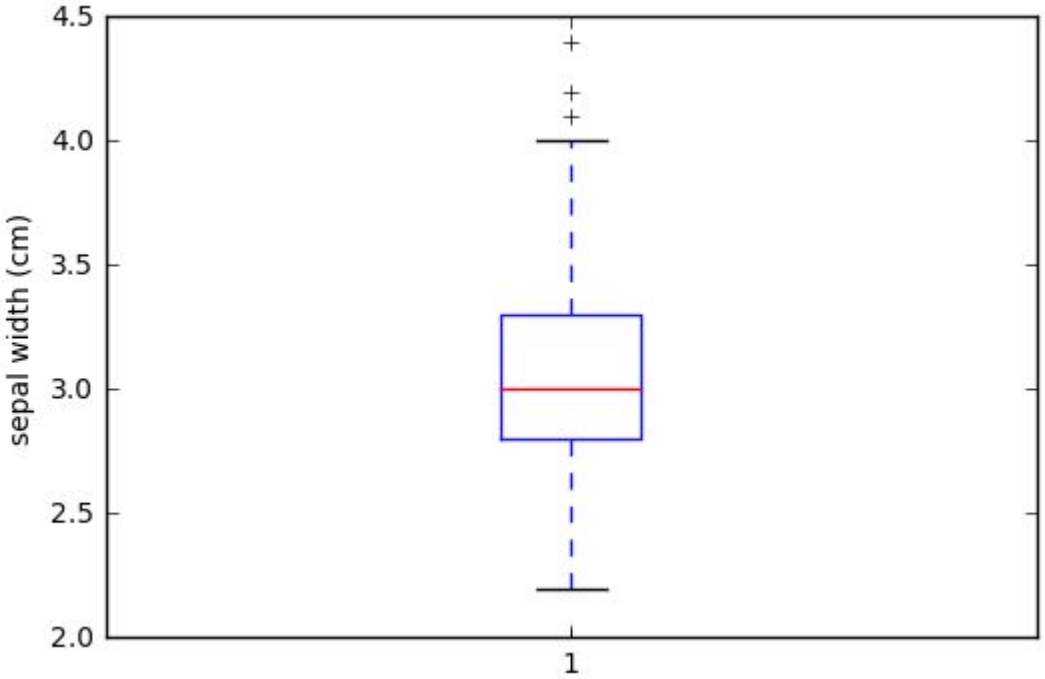
## Box plot feature sepal do dataset Iris

In [2]:
```
# encoding=utf-8
from matplotlib import pyplot as plt
from sklearn.datasets import load_iris

dataset = load_iris()
feature = 1  # pegando largura da sépala

plt.boxplot(
dataset.data[:, feature]
)
plt.ylabel(
dataset.feature_names[feature]
)

plt.show()
```



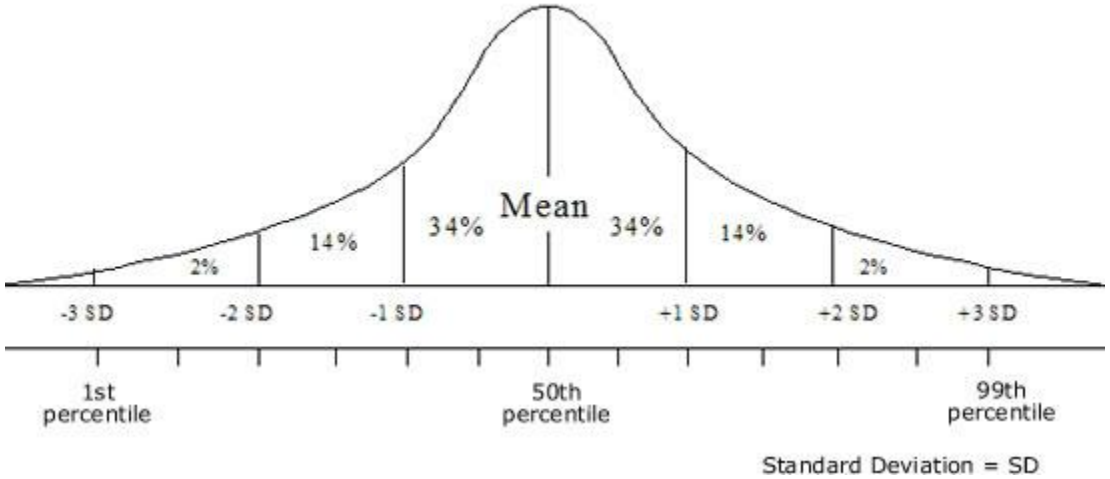
# Exploração dos dados



## Medidas de espelhamento

Medem a dispersão de um conjunto de valores

Intervalo, Variância e Desvio padrão



# Exploração dos dados



**Intervalo**

Ruim para atributos com valores concentrados ao redor de um ponto, com poucos valores extremos

$$r(x) = \max(x) - \min(x)$$

**Variância**

Medida preferida

**Desvio padrão**

Raiz quadrada da Variância.

$$\sigma^2(x) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Outras métricas: **Obliquidade** e **Curtose**. (ver com histogramas)

# Exploração dos dados



## Dados multivariados

### Covariância

Mede o grau com que os atributos variam juntos

Atributos  independentes têm covariância zero

Não indica com clareza a correlação entre atributos

$$\mathrm{cov}(x,y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

### Correlação Pearson

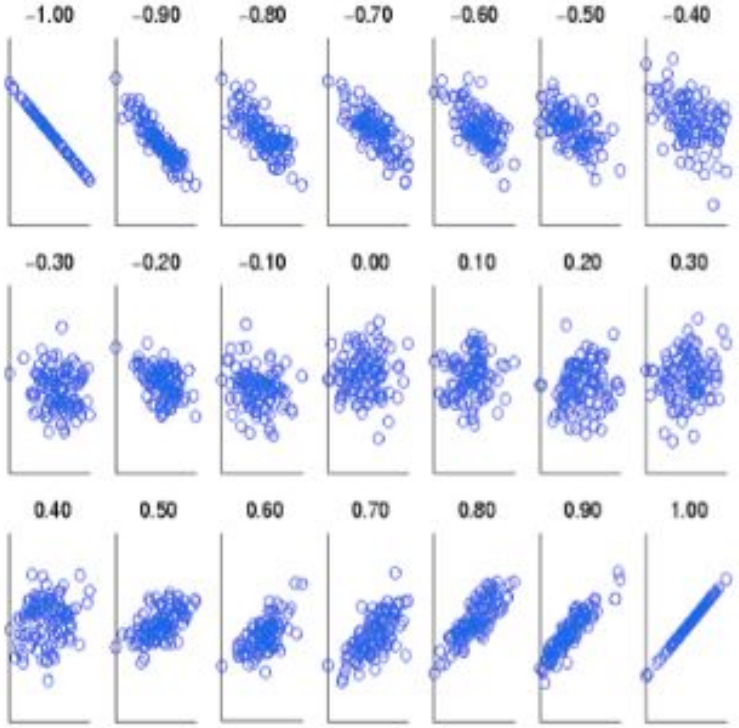
Mede o grau de correlação linear, positiva (1) ou negativa (-1), entre dois atributos

$$corr(x,y) = \frac{\mathrm{cov}(x,y)}{\sigma_x * \sigma_y}$$

# Exploração dos dados



## Correlação



# Exercícios



**Com o dataset pesos_alturas_english.csv, faça:**

- Análise de boxplot para os atributos (plt.boxplot)
- Análise de Distribuição (plt.scatter)
- **Calcule**:
  - Correlação entre os atributos
  - Intervalos
  - Desvio padrão
  - Variâncias
  - Médias
  - Medianas

**Existe correlação?**

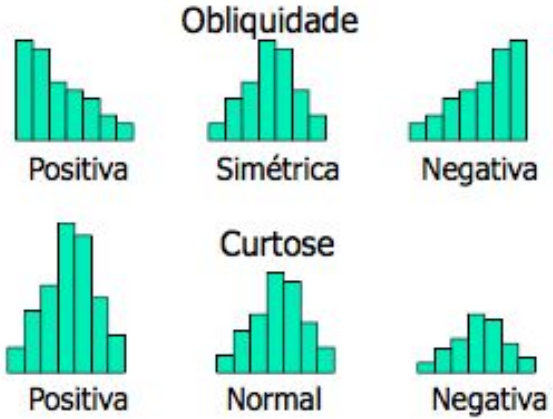
# Exploração dos dados



## Histogramas

Forma de visualização gráfica da distribuição dos dados.

Podemos visualizar, por exemplo, frequência, obliquidade e curtose.



# Exercício



Com o mesmo dataset de peso e altura, plotar histogramas dos atributos.

```
In [73]: dataset.hist()
         plt.show()

         plt.hist(dataset.values)
         plt.show()
```

# Transformação de dados



**Vários algoritmos de ML trabalham somente com dados numéricos.**

Redes Neurais, SVM, etc.

**Necessário converter atributos**

Necessário verificar ordem

**Como converter, por exemplo:** (Comum é associar inteiros crescentes = {1,2,3,4})

```
In [ ]: cores = ['amarelo', 'vermelho', 'verde', 'azul', 'branco']
        temperaturas = ['quente', 'frio', 'morno']
        avaliacoes = ['bom', 'ruim', 'otimo', 'pessimo']
```

# Transformação de dados



Não existe ordem, então transformamos em inteiro e depois binário assimétrico.

| Valor | Valor Inteiro | A1 | A2 | A3 |
|---|---|---|---|---|
| **Amarelo** | 0 | 0 | 0 | 0 |
| **Vermelho** | 1 | 0 | 0 | 1 |
| **Verde** | 2 | 0 | 1 | 0 |
| **azul** | 3 | 0 | 1 | 1 |
| **branco** | 4 | 1 | 0 | 0 |

# Transformação de dados



In [76]:
```
from sklearn import preprocessing

X = [[ 1., -1.,  2.], [ 2.,  0.,  0.], [ 0.,  1., -1.]]
print X

binarizer = preprocessing.Binarizer(threshold=0.0).fit(X)
binarizer.transform(X)
```

```
[[1.0, -1.0, 2.0], [2.0, 0.0, 0.0], [0.0, 1.0, -1.0]]
```

Out[76]:
```
array([[ 1.,  0.,  1.],
       [ 1.,  0.,  0.],
       [ 0.,  1.,  0.]])
```

In [79]:
```
enc = preprocessing.OneHotEncoder()
enc.fit([[0], [1], [2], [0]])
enc.transform([[0], [1], [2]]).toarray()
```

Out[79]:
```
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

# Exercícios



**Com o dataset de acidentes 2005, transforme as colunas TIPO_ACID e REGIAO em binário.**

In [95]:
```
import pandas as pd

df = pd.read_csv('acidentes-2005.csv', sep=';')
print df
x = df['TIPO_ACID'].unique()
print x
pd.get_dummies(df['TIPO_ACID'])
```

In [94]:
```
import pandas as pd

df = pd.read_csv('acidentes-2005.csv', sep=';')
x = df['REGIAO'].unique()
print x
pd.get_dummies(df['REGIAO'])
```

# Tranformação de Dados



**Vários algoritmos de ML trabalham somente com dados categóricos.**

Regras de Associação, etc.

**Necessário converter atributos**

Necessário discretizar em intervalos

**Como converter, por exemplo:** (Comum é identificar intervalos):

```
In [ ]: tempos = [1.0, 1.1, 1.5, 1.7, 2.0, 3.0, 5.0, 7.0, 8.0]
        utilizacoes = [345.5, 345.6, 564.5, 566.6, 78989.0, 90798.1]
```

# Discretização



**Transformação de atributos contínuos em intervalos**

Atributo se transforma em categórico ordinal

**Necessário definir número de intervalos** (na maioria dos casos pelo próprio usuário)

**Necessário definir como mapear os valores contínuos para os intervalos** (categorias)

**Tipos:**

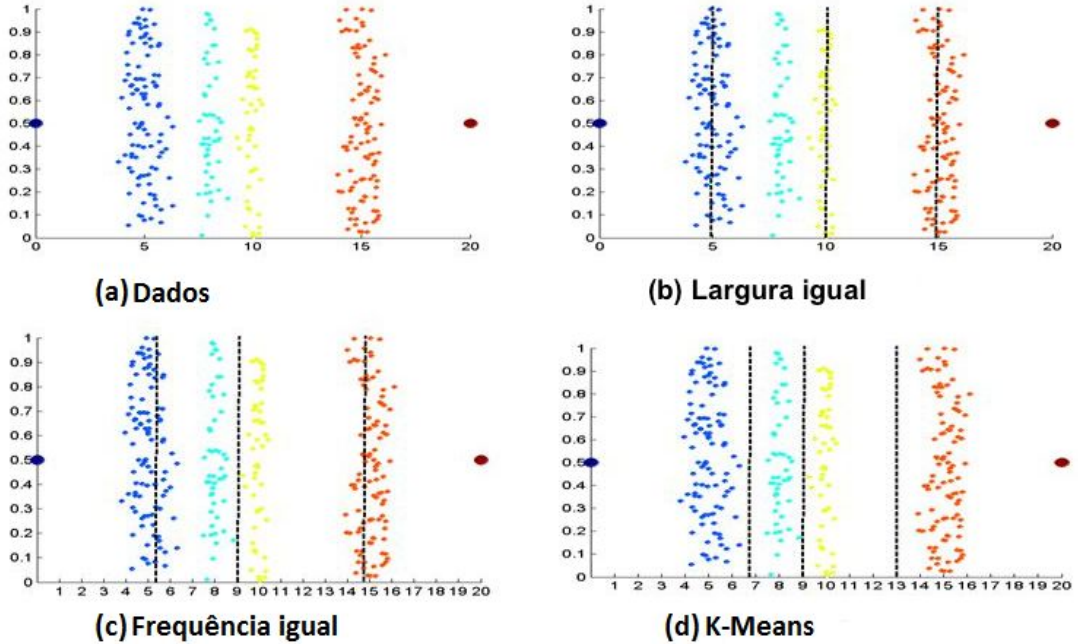
Supervisionado (utilizam valores e classes das instâncias)

Não-supervisionado (Somente os valores das instâncias)

# Discretização



## Discretização Não-supervisionada



# Exercício



**Discretizar os seguintes atributos numéricos com as 3 formas de discretização não supervisionada** (sugestão 3 intervalos):

a = [0, 1, 2, 6, 6, 9, 10, 10, 10, 13, 18, 20, 21, 21, 25]

x = [0, 1, 2, 2, 6, 6, 7, 8, 9, 10, 10, 10, 13, 18, 20, 21, 21, 25]

**Qual das abordagens é mais afetada por outliers?**

```
largura = [[0,1,2,6,6], [9,10,10,10,13], [18,20,21,21,25]]
frequencia = [[0,1,2,6,6], [9,10,10,10,13], [18,20,21,21,25]]
```

```
largurax = [[0,1,2,2,6,6,7,8], [9,10,10,13], [18,20,21,21,25]]
frequenciax = [[0,1,2,2,6,6], [7,8,9,10,10,10], [13,18,20,21,21,25]]
```

# Discretização



## Discretização supervisionada

Objetivo é escolher os intervalos considerando a pureza (máxima ou mínima) das classes.

Para isso, utilizamos a **Entropia.**

Intervalo de dados $S_i$

Probabilidade de que o intervalo i seja da classe j

$$E(S_i) = -\sum_{j=1}^{c} p_{ij} \log_2 p_{ij}$$

Para cada classe *j* no conjunto de todas as classes *c*

A definição de entropia $E(S_i)$ calcula a pureza de um único intervalo

# Discretização



**Para a discretização devemos mensurar (e minimizar) a entropia de todos os n intervalos**

**Nesse caso, a entropia total de um conjunto de intervalos S1, S2, ... , Sn é a média das entropias individuais ponderada pelas respectivas quantidades de elementos mi:**

$$E = \frac{\sum_{i=1}^{n} m_i E(S_i)}{\sum_{i=1}^{m} m_i} = \sum_{i=1}^{n} \frac{m_i}{m} E(S_i)$$

Número total de partições

Número de instâncias na partição i

Entropia da partição i

Número total de instâncias

# Discretização



## Calcular entropia com 3 partições

| ID | Nome | Temperatura | Enjôo | Mancha | Dor | Salário | Diagnóstico |
|---|---|---|---|---|---|---|---|
| **03** | José | 38.2 | sim | grande | não | 600 | saudável |
| **06** | Juliana | 37.7 | não | grande | sim | 900 | doente |
| **01** | Ana | 37.7 | sim | pequena | sim | 1000 | doente |
| **02** | Marcia | 37 | não | pequena | não | 1100 | saudável |
| **05** | Paulo | 37.3 | não | grande | sim | 1800 | saudável |
| **07** | Maria | 37.5 | sim | pequena | sim | 2000 | doente |
| **04** | Pedro | 39 | não | pequena | sim | 2000 | doente |

# Discretização



## Calcular entropia com 3 partições

| ID | Nome | Temperatura | Enjôo | Mancha | Dor | Salário | Diagnóstico |
|---|---|---|---|---|---|---|---|
| **03** | José | 38.2 | sim | grande | não | 600 | saudável |
| **06** | Juliana | 37.7 | não | grande | sim | 900 | doente |
| **01** | Ana | 37.7 | sim | pequena | sim | 1000 | doente |
| **02** | Marcia | 37 | não | pequena | não | 1100 | saudável |
| **05** | Paulo | 37.3 | não | grande | sim | 1800 | saudável |
| **07** | Maria | 37.5 | sim | pequena | sim | 2000 | doente |
| **04** | Pedro | 39 | não | pequena | sim | 2000 | doente |

# Discretização



**Se um intervalo possuir apenas uma classe a entropia é 100% pura (zero ou 1)**

**Se a distribuição é 50 - 50 acontece a impureza máxima (0.5)**

Exemplo: (2 classes)

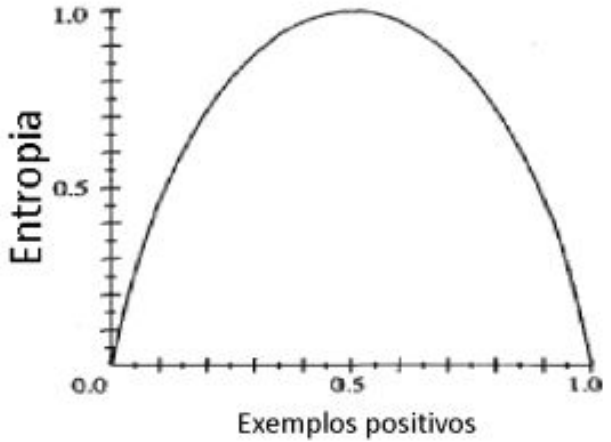
-pi1 = 1 - pi2

impureza mínima

pi1 ou pi2 = 0 (ou 1)

impureza máxima

pi1 = pi2 = 0.5



# Transformação dos dados



**Muita vezes é necessário que os dados originais sejam transformados ou consolidados em diferentes formatos (uma mesma escala).**

Medidas de similaridade e algoritmos são muitas vezes influenciados.

**Tipos:**

Re-escalar (normalization, minmax)

Padronização (standardization, z-score)

Transformação logarítmica (logarithmic transformation)

# Transformação dos dados



**Re-escalar (minmax):**

Dados de um atributo x o são individualmente reescalados em um pequeno intervalo (0 e 1 ou -1 e +1).

$$x' = \frac{(x - min_x)}{(max_x - min_x)}$$

In [111]:
```
import numpy as np
a = np.arange(20)
print 'maximo antigo:', a.max()
print 'minimo antigo:', a.min()

b = np.array(map(lambda x: (x - a.min())/(a.max() - a.min()), a))
print 'maximo novo:', b.max()
print 'minimo novo:', b.min()
```

```
maximo antigo: 19
minimo antigo: 0
maximo novo: 1
minimo novo: 0
```

# Transformação dos dados



**Re-escalar (minmax):**

Dados de um atributo x o são individualmente reescalados em um pequeno intervalo (0 e 1 ou -1 e +1).

$$x' = \frac{(x - min_x)}{(max_x - min_x)}$$

In [116]:
```
from sklearn.preprocessing import MinMaxScaler as minmax

scaler = minmax()
scaler.fit(a.reshape(-1,1))
c = scaler.transform(a.reshape(-1,1))
print 'maximo novo:', c.max()
print 'minimo novo:', c.min()
```

```
maximo novo: 1.0
minimo novo: 0.0
```

http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

# Transformação dos dados



**Padronização (z-score):**

Dados de um atributo x são individualmente padronizados com base na média e no desvio-padrão.

$$x' = \frac{(x - \mu_x)}{(\sigma_x)}$$

In [120]:
```
import numpy as np
d = np.arange(20)
print 'media antiga:', d.mean()
print 'desvio padrao antigo:', d.std()

e = np.array(map(lambda x: (x - d.mean())/d.std(), d))
print 'media nova:', e.mean()
print 'desvio padrao novo:', e.std()
```

```
media antiga: 9.5
desvio padrao antigo: 5.76628129734
media nova: 0.0
desvio padrao novo: 1.0
```

# Transformação dos dados



**Padronização (z-score):**

Dados de um atributo x são individualmente padronizados com base na média e no desvio-padrão.

$$x' = \frac{(x - \mu_x)}{(\sigma_x)}$$

```
In [121]: from sklearn.preprocessing import StandardScaler as zscore

          scaler = zscore()
          scaler.fit(d.reshape(-1,1))
          f = scaler.transform(d.reshape(-1,1))

          print 'media nova:', f.mean()
          print 'desvio padrao novo:', f.std()
```

```
media nova: 0.0
desvio padrao novo: 1.0
```

http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler

# Transformação dos dados



**Transformação logarítmica (logarithmic transformation):**

Dados de um atributo x são individualmente transformados por meio da aplicação de logaritmo.

$$x' = \log(x)$$

```
In [130]: import numpy as np
          g = np.array([5,6,7,8,9,10,11,12,13,14])
          print 'maximo novo:', g.max()
          print 'minimo novo:', g.min()

          e = np.array(map(lambda x: np.log(x), g))
          print 'maximo novo:', e.max()
          print 'minimo novo:', e.min()
```

```
maximo novo: 14
minimo novo: 5
maximo novo: 2.63905732962
minimo novo: 1.60943791243
```

# Transformação dos dados



## Centralizar valores na média

In [140]:
```
import numpy as np
from sklearn.preprocessing import scale

X = np.array([[ 1., -1.,  2.], [ 2.,  0.,  0.], [ 0.,  1., -1.]])
print X
X_scaled = scale(X)
print X_scaled
mean = X_scaled.mean(axis=0)
print mean
std = X_scaled.std(axis=0)
print std
```

```
[[ 1. -1.  2.]
 [ 2.  0.  0.]
 [ 0.  1. -1.]]
[[ 0.         -1.22474487  1.33630621]
 [ 1.22474487  0.         -0.26726124]
 [-1.22474487  1.22474487 -1.06904497]]
[ 0.  0.  0.]
[ 1.  1.  1.]
```

http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.scale.html

# Exercícios



| Idade | Salário | Re-escalar | Padronização |
|---|---|---|---|
| 20 | 2.000,00 | | |
| 19 | 845,00 | | |
| 32 | 7.580,00 | | |
| 55 | 19.430,00 | | |
| 49 | 12.300,00 | | |
| 37 | 3.500,00 | | |
| 26 | 4.950,00 | | |

# Transformação dos dados



**Em geral, qualquer função decrescente pode ser utilizada para a transformação de atributos em diferentes escalas**

- Média e desvio padrão são influenciados por outliers
- Pode-se utilizar mediana e desvio padrão absoluto
- Pode-se utilizar Agregação e Generalização

# Exercícios



Modifique 3 atributos nominais do dataset de acidentes em categóricos numéricos

In [157]:
```
import pandas as pd
base = pd.read_csv('acidentes-2005.csv', sep=';')
base['colunal'] = base['colunal'].astype('category')
categories = base.select_dtypes(['category']).columns
base[categories] = base[categories].apply(lambda x: x.cat.codes)
```

# Exercícios



Modifique 3 atributos nominais do dataset de acidentes em categóricos numéricos

In [165]:
```
import pandas as pd
df = pd.read_csv('acidentes-2005.csv', sep=';')
# print df
df['LOCAL'] = df['LOCAL'].astype('category')
df['TIPO_ACID'] = df['TIPO_ACID'].astype('category')
df['REGIAO'] = df['REGIAO'].astype('category')
categories = df.select_dtypes(['category']).columns
# print df[categories]
df[categories] = df[categories].apply(lambda x: x.cat.codes)
print df[categories]
```

| | LOCAL | TIPO_ACID | REGIAO |
|---|---|---|---|
| 0 | 1 | 3 | 1 |
| 1 | 1 | 4 | 4 |
| 2 | 1 | 2 | 1 |
| 3 | 1 | 3 | 1 |
| 4 | 1 | 3 | 4 |
| 5 | 1 | 3 | 3 |

# Transformação de Dados



Um problema bastante comum é a ausência de valores em determinados atributos, ou seja, registros com dados incompletos, seja por falhas no processo de seleção ou de revisão.

**Imputação de dados faltantes (como?)**

Técnicas de imputação de valores

Média dos valores

Exclusão da instância ou atributo

# Transformação de Dados



In [176]:
```
import numpy as np
from sklearn.preprocessing import Imputer
X = np.array([[np.nan, 2], [6, np.nan], [7, 6]])
print X
imp = Imputer(missing_values='NaN', strategy='mean')
imp.fit(X)
print imp.transform(X)
```

```
[[ nan   2.]
 [  6.  nan]
 [  7.   6.]]
[[ 6.5  2. ]
 [ 6.   4. ]
 [ 7.   6. ]]
```

5]:
```
import numpy as np
from sklearn.preprocessing import Imputer
X = np.array([[np.nan, 2], [6, np.nan], [7, 6]])
print X
T = np.array([[1, 2], [np.nan, 3], [7, 6]])
print T
imp = Imputer(missing_values='NaN', strategy='mean')
imp.fit(T)
print imp.transform(X)
```

```
[[ nan   2.]
 [  6.  nan]
 [  7.   6.]]
[[  1.   2.]
 [ nan   3.]
 [  7.   6.]]
[[ 4.          2.        ]
 [ 6.          3.66666667]
 [ 7.          6.        ]]
```

Pandas: df.dropna() e df.fillna()

https://pandas.pydata.org/pandas-docs/stable/missing_data.html

# Dis(similaridade)



**Alguns algoritmos de Agrupamento e também outras tarefas de Aprendizado de Máquina necessitam de um valor de similaridade entre instâncias que compõem um conjunto de dados.**

Similaridade

- Medida que indica nível de semelhança entre dois objetos
- Quanto mais semelhantes, maior o seu valor
- Geralmente valor $\in [0, 1]$

Dissimilaridade

- Medida que indica o quanto dois objetos são diferentes
- Quanto mais diferentes, maior o seu valor
- Geralmente valor $\in [0, dmax]$ ou $[0, +\infty]$

Calcular a distância entre as características dos objetos

- **Distâncias**: Euclidiana, SMC, Jaccard (variações), Manhattan, Minkowski e Cosseno.

# Dis(similaridade)



## Medidas de proximidade e distância

Geralmente entre 0 e 1

| Attribute Type | Dissimilarity | Similarity |
|---|---|---|
| Nominal | $d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$ | $s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$ |
| Ordinal | $d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where $n$ is the number of values) | $s = 1 - \frac{\lvert p-q \rvert}{n-1}$ |
| Interval or Ratio | $d = \lvert p - q \rvert$ | $s = -d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$ |

# Dis(similaridade)



**Distância Euclidiana (uma das mais utilizadas)**

para valores numéricos

$$Dist\_E(X_i, X_j) = \sqrt{\sum_{k=1}^{m} (X_{i,k} - X_{j,k})^2}$$

Número de atributos

Valor do k-ésimo atributo na i-ésima instância

# Dis(similaridade)



## Distância Euclidiana

In [183]:
```
from sklearn.metrics.pairwise import euclidean_distances
A = [3, 2, 0, 5, 0, 0, 0, 2, 0, 0]
B = [1, 0, 0, 0, 0, 0, 0, 1, 0, 2]
# distance between rows of X
print euclidean_distances(np.array(A).reshape(1,-1), np.array(B).reshape(1,-11))
```

```
[[ 6.164414]]
```

# Dis(similaridade)



**Para valores binários**

Exemplo $M_{pq}$:

$M_{01}$ = atributos onde p é 0 e q é 1
$M_{10}$ = atributos onde p é 1 e q é 0
$M_{00}$ = atributos onde p é 0 e q é 0
$M_{11}$ = atributos onde p é 1 e q é 1

# Dis(similaridade)



**Simple Matching (SMC)**

**SMC** = número de valores combinados / número de valores

= $(M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00})$

exemplo: verdadeiro/falso

**Jaccard Index (J)**

**J** = 1 - (número de valores combinados **não ambos zero** / número de **valores não ambos zero)**

= $1 - ((M_{11}) / (M_{01} + M_{10} + M_{11}))$

exemplo: valores de ações executadas

# Exercícios



**Qual a similaridade das instâncias (SMC e J):**

p = 1 0 0 0 0 0 0 0 0

q = 0 0 0 0 0 1 0 0 1

$M_{00} = 7$

$M_{01} = 2$

$M_{10} = 1$

$M_{11} = 0$

**SMC** = $(M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00}) = (0 + 7) / (2 + 1 + 0 + 7) = 0.7$

**J** = $1 - ((M_{11}) / (M_{01} + M_{10} + M_{11})) = 1 - (0 / (2 + 1 + 0)) = 1$

# Dis(similaridade)



**Vetores assimétricos podem ter magnitudes diferentes ou ainda serem não-binários.**

Contagem de palavras em um dataset textual

Lista de produtos comprados

Nestes casos, utilizamos Variação do **Jaccard** ou **similaridade cosseno**

# Similaridade



**Para listas (números e textos)**

Exemplo:

A = [23, 45, ‘frio’]

B = [23, 50, 75, ’frio’]

**Jaccard Index (J)**

$$Dist\_J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} = 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

A intersecção B = [23, ‘frio’] = 2

A união B = [23, 45, 50, 75, ‘frio’] = 5

**J** = 1 - (2 / 5) = 1 - 0.4 = 0.6

# Dis(similaridade)



## Jaccard

In [185]:
```
from sklearn.metrics.pairwise import euclidean_distances
import numpy as np
A = [23, 45, 'frio']
B = [23, 50, 75, 'frio']

union = float(len(np.union1d(A, B)))
intersection = float(len(np.intersect1d(A, B)))
j = 1 - (intersection/union)
print j
```

```
0.6
```

# Dis(similaridade)



**Coseno**

$$\cos(\mathbf{d}_1, \mathbf{d}_2) = (\mathbf{d}_1 \bullet \mathbf{d}_2) / \|\mathbf{d}_1\| \|\mathbf{d}_2\|$$

Calcule a similaridade cosseno entre os dois documentos:

d1 = 3 2 0 5 0 0 0 2 0 0

d2 = 1 0 0 0 0 0 0 1 0 2

d1 · d2= 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5

||d1|| = raiz(3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0) = raiz(42) = 6.48

||d2|| = raiz(1*1+0*0+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2) = raiz(6) = 2.44

cos( d1, d2 ) = 0,316

# Exemplo



In [177]:
```
from sklearn.metrics.pairwise import cosine_similarity
A = [3, 2, 0, 5, 0, 0, 0, 2, 0, 0]
B = [1, 0, 0, 0, 0, 0, 0, 1, 0, 2]
print cosine_similarity(A, B)
```

```
[[ 0.31497039]]
```

# Exercícios



Para o datatset peso e altura, calcule a distância euclidiana e o cosseno entre os atributos Peso e Altura

Métodos para calcular manualmente

- np.corrcoef(x, y)[0, 1]
- np.dot(x, y)
- np.linalg.norm(x)
- np.sum()
- np.cov(x, y)
- np.std(x)

# Conlusão



**Leitura recomendada:**

Capítulo 2 e 3 de Introduction to Data Mining