Preparação dos Dados

Departamento de Engenharia Informática (DEI/ISEP)
Fátima Rodrigues
mfc@isep.ipp.pt

Motivação

Em aplicações reais os dados tendem a ser inconsistentes, incompletos e/ou errados

O que acontece quando os dados não são correctos ?

Pode o conhecimento extraído dos dados ser de confiança ?

Obstáculos para a Descoberta de Conhecimento → **Dados de má qualidade**

Velho ditado da computação: ***Garbage in Garbage out***

Preparação dos Dados - Objectivos

- Compreender a natureza dos dados
- Resolver problemas inerentes a características intrínsecas e/ou à recolha dos dados
- Saber que informação útil existe em determinado conjunto de dados por forma a que esta seja preservada quando dela se formam subconjuntos aleatórios de amostras
- Adaptar os dados de acordo com os algoritmos de DM
- Proporcionar análises de dados mais significativas
- Extrair conhecimento com mais significado a partir dos dados
- Estima-se que a Preparação dos Dados tome **70-80%** de todo o esforço de desenvolvimento de um projecto de Data Mining

Problemas nos Dados

Problemas nos Dados

- Muitos Dados
  - dados corrompidos
  - dados com ruído
  - tabelas muito grandes
- Poucos Dados
  - falta de atributos de interesse
  - falta de valores nos atributos
  - pequenas amostras
- Dados Inconsistentes
  - dados incompatíveis
  - diferentes fontes de dados
  - dados com diferentes níveis de granularidade

Fase de Seleção

Fase de Seleção

- Conhecimento e compreensão da área em estudo
- Análises multidimensionais dão suporte à criação e uso de relações quantitativas
- Escolha dos atributos de acordo com os objetivos de Descoberta
  - selecionar todos os atributos relacionados com o objetivo de descoberta
  - a decisão final de inclusão ou não de um atributo na mostra de dados é novamente avaliada na fase de Pré-processamento

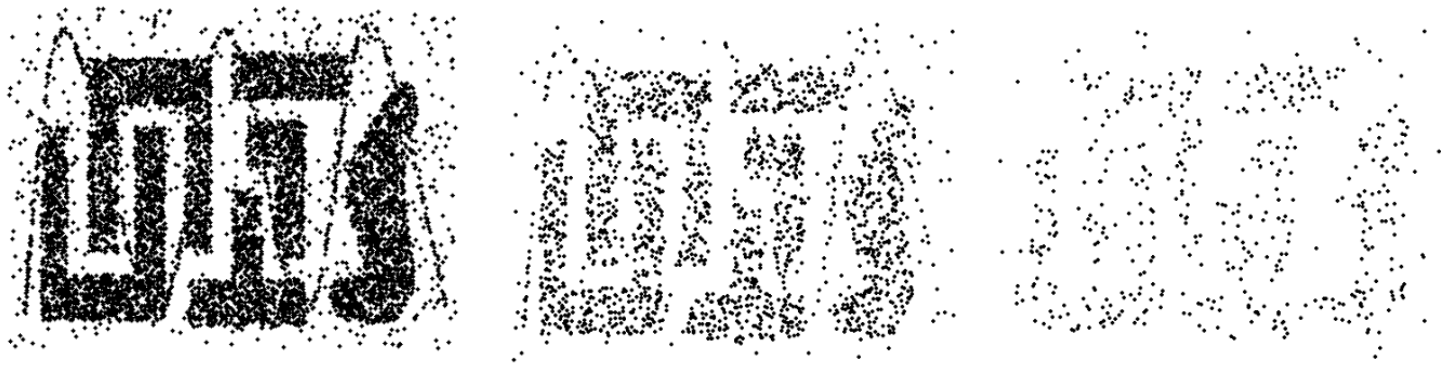
Amostragem

Amostra: é um subconjunto de observações da população que idealmente representa bem a população

Necessária quando a experimentação com a população inteira é impossível ou muito cara

Principio chave de amostragem

- O uso de uma amostra é superior à exploração de toda a BD se a **amostra for representativa**
- Uma **amostra é representativa** se tem aproximadamente as **mesmas propriedades de interesse** que a BD original



8000 points **2000 Points** **500 Points**

Amostragem

Todas as instâncias têm igual probabilidade de serem selecionadas – seleção puramente aleatória

Simple random sampling with replacement (com substituição)

- Cada instância selecionada é devolvida à população inicial – a mesma instância pode ser selecionada várias vezes

Simple random sampling without replacement (sem substituição)

- Os objetos selecionados para a amostra são retirados da população inicial

Amostragem aleatória com substituição

Sampling with replacement using NumPy

```
In [7]: np.random.seed(3)

        np.random.choice(a=12, size=12, replace=True)

Out[7]: array([10,  8,  9,  3,  8,  8,  0,  5,  3, 10, 11,  9])
```

Sampling with replacement using pandas

```
In [9]: dfaux = df.head(5)

        dfaux.sample(n = 5, replace = True, random_state=2)
```

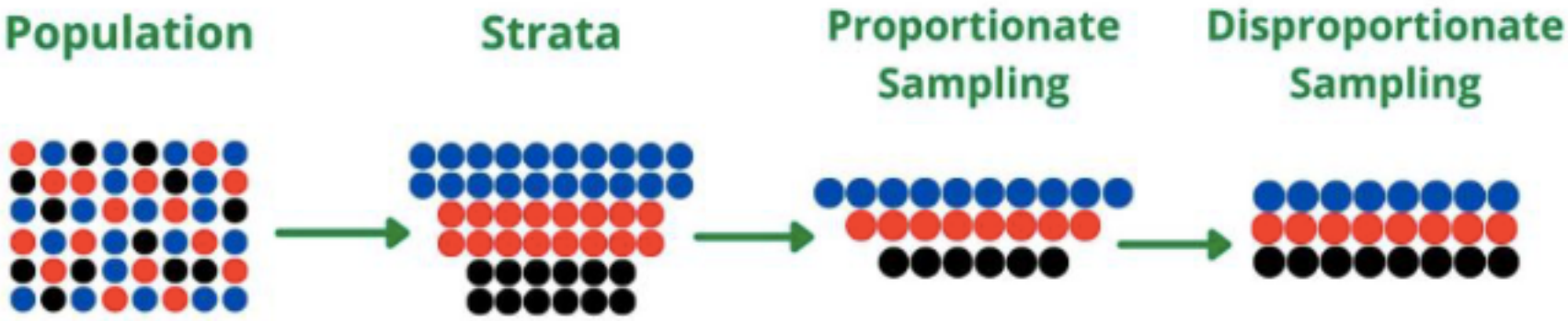
Out[9]:

	category	discipline	phd	service	sex	salary
0	Prof	B	56	49	Male	186960
0	Prof	B	56	49	Male	186960
3	Prof	A	40	31	Male	131205
2	Prof	A	23	20	Male	110515
3	Prof	A	40	31	Male	131205

Amostragem aleatória estratificada

Stratified random sampling (Estratificada)

A amostra é feita de modo a manter a proporção da variável objetivo da população inicial



Amostragem aleatória estratificada

In [21]:
```
counts = df.category.value_counts()
percent100 = df.category.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.DataFrame({'category': counts,'percent': percent100})
```

Out[21]:

	category	percent
Prof	46	59.0%
AsstProf	19	24.4%
AssocProf	13	16.7%

In [26]:
```
from sklearn.model_selection import train_test_split

X = df.iloc[:,1:]  # Select From 2nd to end
y = df.iloc[:, 0]  # Select first column

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

# Verify distribution
print("Train category distribution\n",y_train.value_counts(normalize=True))
print("Test category distribution\n",y_test.value_counts(normalize=True))
```

```
Train category distribution
 Prof          0.592593
AsstProf      0.240741
AssocProf     0.166667
Name: category, dtype: float64
Test category distribution
 Prof          0.583333
AsstProf      0.250000
AssocProf     0.166667
Name: category, dtype: float64
```

Volume de Dados

- Depende da complexidade do problema
  - Nº atributos da amostra
  - Valores distintos dos atributos
  - Quantidade de ruído existente nas amostras de dados

- Depende da operação DM a realizar
  - Classificação: várias instâncias por cada classe
  - Regressão: problema mais complexo, classes contínuas
  - Redes Neuronais: requerem muitos dados, pois amostras pequenas são facilmente memorizadas

Amostragem Incremental: Nº Casos

O treino é realizado em amostras aleatórias com um número de casos cada vez maior, observa-se a tendência e pára-se quando não houver mais progresso

Um padrão típico de tamanhos de amostras pode ser: 10%, 20%, 33%, 50%, 65%

Critérios de paragem:

- O erro diminuiu ?
- A complexidade do modelo aumentou mais do que a queda da taxa de erro ?
- A complexidade da solução actual é aceitável para a interpretação ?

Classes Desequilibradas

Objectivo Descoberta Desequilibrado

- Desequilíbrio de classes ocorre quando uma ou mais classes têm proporções muito baixas nos dados de treino em comparação com as outras classes

- O desequilíbrio de classes ocorre com frequência em aplicações:
  - Previsão de abandono operador: 97% permanecem, 3% abandonam
  - Diagnóstico médico: 90% saudáveis, 10% não saudáveis
  - eCommerce: 99% não compram, 1% compra
  - Segurança: >99.99% das pessoas são não terroristas
  - **....**

- Situação similar ocorre com classes múltiplas

- Nestes casos o Classificador até apresenta uma tx. de acerto elevada (na classe maioritária), mas com pouca ou nenhuma utilidade

↳ Tx. acerto não ocorre na classe em minoria

Previsão de Classes Desequilibradas

Existem várias formas de lidar com classes desequilibradas no sentido de se melhorar a precisão dos modelos, através de:

- Afinação do modelo
- Ajustar as probabilidades à Priori
- Dar pesos desiguais aos casos
- Métodos de Amostragem
- Treino sensível aos Custos

Gestão de Dados Desequilibrados

Duas classes objetivo

- A partir do conjunto inicial de dados, aplicar método de amostragem adequado
- Gerar **conjunto de treino balanceado**
- Construir os modelos usando os **dados balanceados**
- Estimar os resultados finais no conjunto de teste com distribuição de dados inicial - desta forma é apresentada uma **estimativa honesta** do desempenho futuro do modelo

- **Múltiplas classes**
  - **Amostragem estratificada**
    - Garantir que cada classe é representada aproximadamente em iguais proporções no conjunto de treino

Random Undersampling

Cria um novo conjunto de treino incluindo todos os exemplos "positivos" e escolhendo aleatoriamente exemplos "negativos"

Prós:

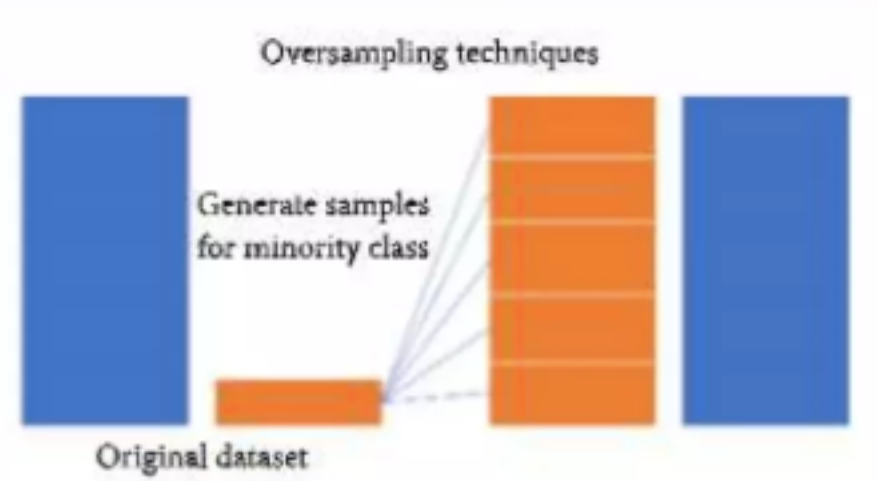
- fácil de implementar
- o treino é mais rápido (conjunto de treino menor)
- para alguns domínios, pode funcionar muito bem

Contra:

- Perda de dados/informações

Random Oversampling

Faz amostragem aumentando a distribuição da classe minoritária



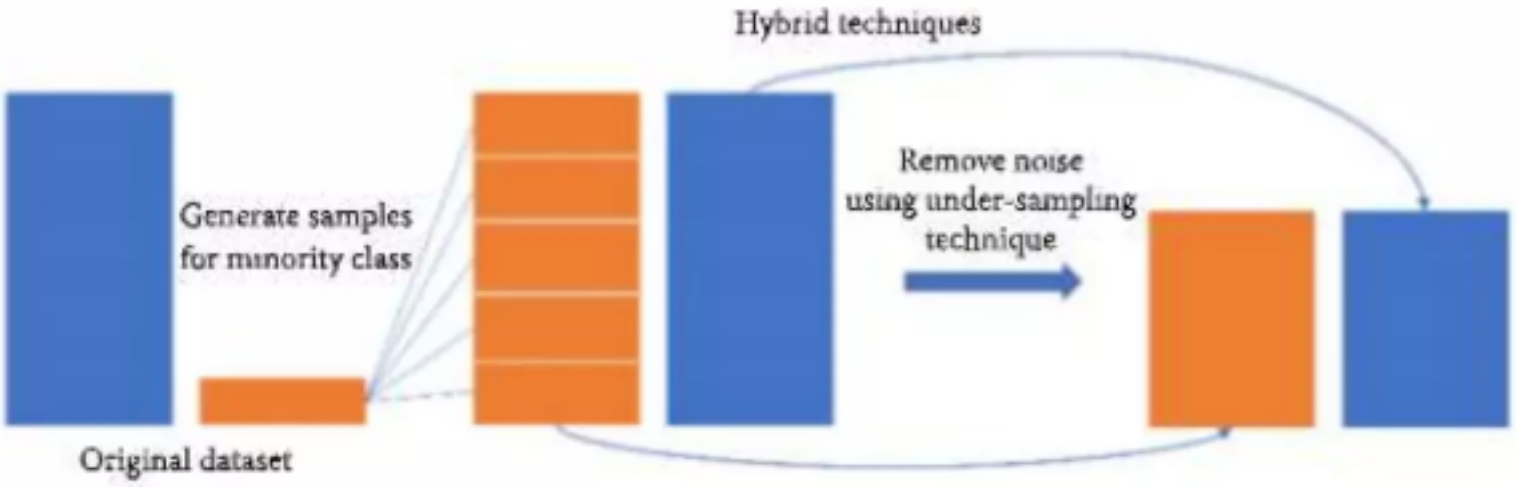
Prós:

- Fácil de implementar
- Utiliza todos os dados de treino
- Tende a ter um bom desempenho num conjunto mais amplo de dados do que a subamostragem

Contra:

- Computacionalmente mais caro para treinar o classificador
- Pode conduzir a *overffiting*

Abordagem híbrida



Algoritmos para balanceamento de dados

- SMOTE (Synthetic Minority Over-sampling Technique)
- ADASYN (Adaptive Synthetic Sampling)
- Tomek Links

```
# Create a synthetic imbalanced dataset
X, y = make_classification(n_classes=2, weights=[0.95, 0.05], n_samples=1000,
                                                              random_state=42)


# Instantiate the SMOTE oversampler
smote = SMOTE(sampling_strategy='auto', random_state=42)

# Apply SMOTE to balance the dataset
X_resampled, y_resampled = smote.fit_resample(X, y)

# Check the class distribution after oversampling
print("\nClass distribution after oversampling:")

unique, counts = np.unique(y_resampled, return_counts=True)
print(pd.DataFrame({'Values': unique,'percent': counts/sum(counts)}))
```

```
Class distribution after oversampling:
   Values  percent
0       0      0.5
1       1      0.5
```

Fase de Limpeza

Fase de Limpeza

Envolve a manipulação física dos dados

Operações mais frequentemente aplicadas:

- **Filtragem** - para tratar dados corrompidos e com ruído
- **Ordenação** - para organizar os dados em localizações próprias tabelas para posterior análise e manuseamento
- **Edição de dados** - aplicada para corrigir dados

Erros nos Dados

Erros sistemáticos

- introduzidos de forma previsível
  - ↪ potencialmente detectáveis e corrigíveis, Ex: calibração incorrecta de equipamento

Erros não sistemáticos

- introduzidos de forma imprevisível
  - ↪ mais difíceis de detectar e corrigir

Exemplos inválidos

- Valores duplicados
- Valores inconsistentes
- Valores ausentes
- Valores isolados

Ruído

- Distorção de um valor com componente espacial ou temporal

Valores Ausentes

Um **valor ausente** pode ser um **valor em falta** no conjunto de dados, mas existente no contexto em que a medida foi realizada

Numa Base de dados os valores em falta são indicados:

- em atributos numéricos por valores negativos ou nulos
- em atributos não numéricos por brancos ou traços
- ás vezes por uma mesma constante

Um valor ausente pode também ser um **valor inaplicável**, ou seja um valor ausente e inexistente no contexto em que a medida foi realizada

Ex: Sexo = Masculino e hemofílico = S/N

Sexo = Feminino e hemofílico = null

A diferenciação entre valores em falta e valores inaplicáveis é mais importante ainda quando não se dispõem de técnicas automáticas que os distingam

Em algumas situações os dados inaplicáveis são altamente informativos

Tratamento de Valores em Falta

- remoção dos registos com valores em falta
- preencher os valores em falta manualmente (se em número reduzido)
- preencher os valores em falta usando os valores de outros atributos
- preencher com valores comuns
  - moda para dados categóricos (apresenta limitações)
  - mediana para valores ordenados
- usar o valor mais provável segundo um modelo baseado em valores de outros atributos:
  - Regressão Linear
  - Aprendizagem Baseada em Instâncias ou
  - Média dos k Vizinhos-Mais-Próximos, para dados numéricos

Conversão de Valores

- Os valores de um mesmo atributo podem diferir segundo as diversas fontes, isto pode acontecer devido a diferenças na representação, escala ou codificação
  - Peso: em libras ou em quilos
  - Altura: valor numérico ou categórico (1,80m, médio, pequeno...)
  - Preço: pode indicar serviços diferentes

- As amostras devem ser representadas à mesma escala

Dados Categóricos → Valores Numéricos

- Atributos binários: **codificados 0/1**

```
dfML[feature] = (dfML[feature].values == 'Yes').astype(int)
```

ou

```
diag_map = {'M':1, 'B':0}
dfML['diagnosis']= df['diagnosis'].map(diag_map)
```

- Atributos com 3 ou mais valores: **codificação dummy**

participant_id	race	asian	black	hispanic	height
1	Asian	1	0	0	67
2	Black	0	1	0	69
3	Hispanic	0	0	1	66
4	White	0	0	0	68

```
dfML = pd.get_dummies(dfML, drop_first=True)
```

Valores Isolados (”outliers”)

São objetos com características bastante diferentes da maioria dos restantes objetos do conjunto de dados, mas **são objetos válidos**

Em muitas aplicações os **valores isolados** correspondem a situações críticas, com elevados custos associados, exigindo por isso ações preventivas e/ou corretivas a serem tomadas

Ex. Aplicações de detecção de fraude

Noutras aplicações os **valores isolados** podem ser considerados **ruído**, ou valores atípicos e são frequentemente tratados como erros e, portanto, devem ser eliminados da análise

Modelos como árvore de decisão e máquinas de suporte vectorial são resistentes aos outliers, no entanto outros modelos como por exemplo baseados no cálculo de distâncias podem ser altamente influenciáveis pelos outliers

Identificação de Valores Isolados

Análise gráfica dos valores da variável - Boxplot

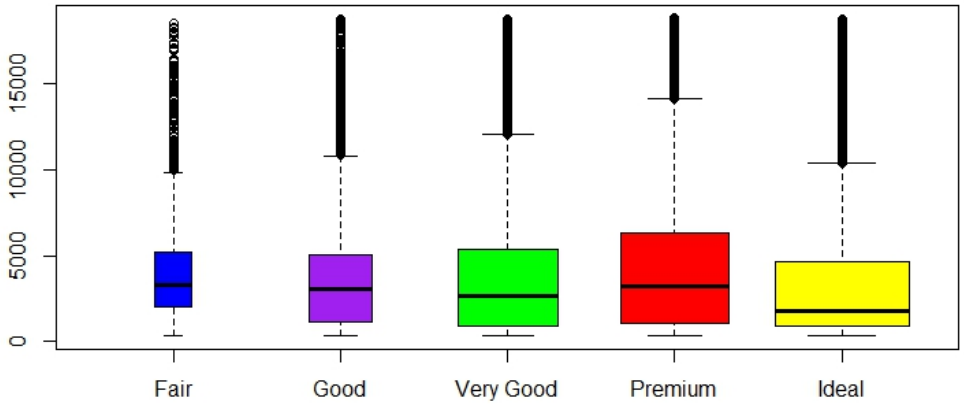
Este método assume uma distribuição quase normal dos valores de uma variável e marca como valores extremos quaisquer valores **fora do intervalo**,

$$[Q1-1.5\times IQR, \ldots, Q3+1.5\times IQR]$$

Sendo

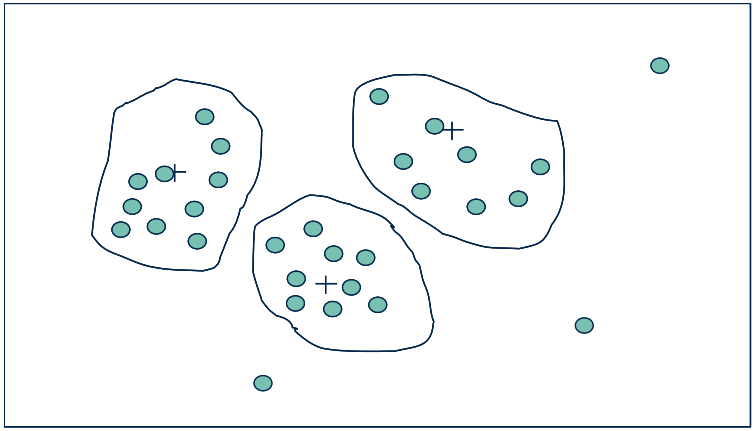
Q1 (Q3) o 1º e 3º quartil

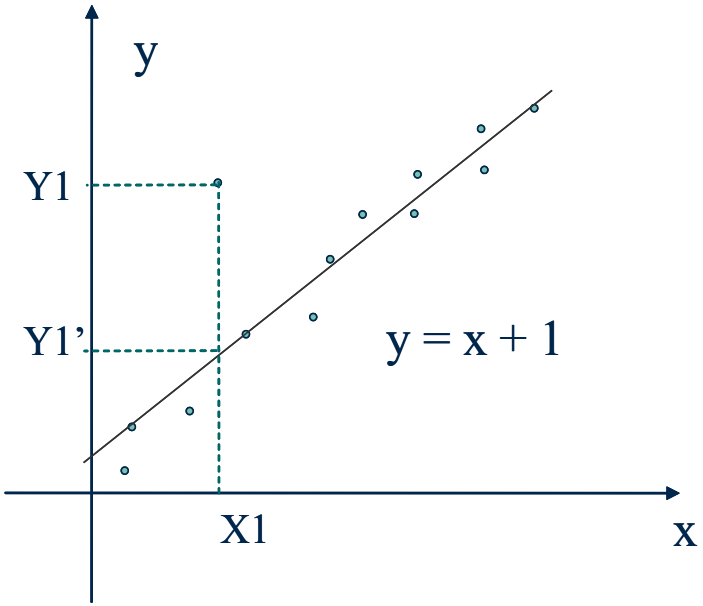
IQR diferença inter-quartil (=Q3-Q1)



Identificação de Valores Isolados

- Clustering
- Regressão





Redução dos dados a uma mesma escala

Redução dos dados a uma mesma escala, para que variáveis, medidas em diferentes escalas, tenham valores comparáveis

É necessária porque certos algoritmos não-paramétricos assumem implicitamente que as distâncias em diferentes direcções do espaço de entrada têm o mesmo peso

- algoritmos da vizinhança-mais-próxima
- Algoritmos de Clustering
- Redes Neuronais/Redes Kohonnen
- Análise dos Componentes Principais
- SVM

Variáveis com grandes valores numéricos podem dominar os efeitos de variáveis com valores menores mas de igual modo importantes na definição do modelo. Exemplo: idade versus salário

Normalização Min-Max

A normalização Min-Max realiza uma transformação linear do conjunto de entrada original para um novo conjunto específico (tipicamente 0-1)

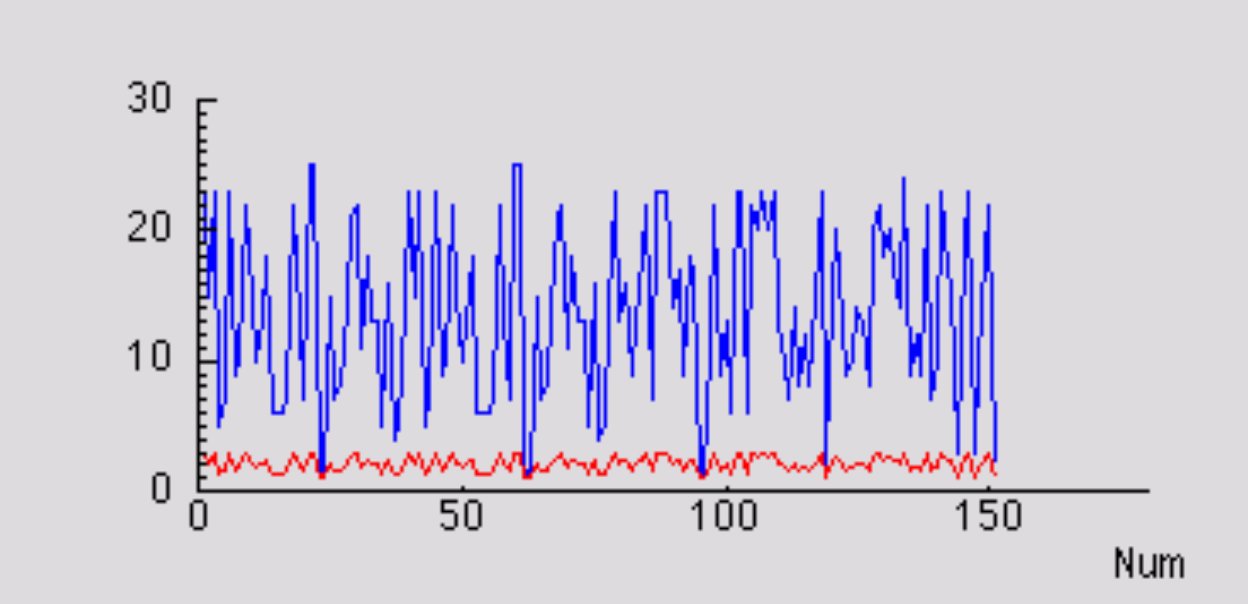
- o antigo mínimo (min_1) é mapeado para um novo mínimo $\rightarrow$ min_2
- o antigo máximo (max_1) é mapeado para um novo máximo $\rightarrow$ max_2

Todos os pontos entre estes dois extremos são mapeados para a nova escala

A fórmula matemática para a normalização Min-Max:

$$y' = \frac{y - \min_y}{\max_y - \min_y}$$

Vantagens da Normalização Min-Max



- Preserva exactamente todas as relações iniciais dos valores dos dados
- Não introduz quaisquer alterações nos dados

  ↪ a forma do histograma é mantida
- **Não funciona bem** em amostras com **valores isolados**

Normalização Zscore

Referida também como média-zero ou normalização standard, consiste em subtrair a média e dividir pelo desvio padrão.

$$y' = \frac{y - média_y}{desvio.padrão_y}$$

Cada valor reflete a distância à média em unidades de desvio padrão

A normalização Zscore funciona bem quando:

- A amostra tem valores isolados que dominam a normalização Min-Max
- A média de uma variável padronizada z-score é sempre zero, e o intervalo de valores é bastante compacto
- Z-score superior a 3 ou inferior a -3 indica valores extremamente raros

Normalização Sigmoidal

Normaliza dados de entrada não-lineares num intervalo [-1,1] usando a função sigmoidal.

A fórmula aplicada por este tipo de normalização é a seguinte:

$$y' = \frac{1 - e^{-\alpha}}{1 + e^{-\alpha}} \qquad \alpha = \frac{y - média}{desvio\ padrão}$$

A normalização sigmoidal é apropriada quando se pretende :

- **incluir pontos isolados** no conjunto de dados a analisar
- evitar que os valores mais comuns sejam comprimidos, sem contudo perder a habilidade de representar valores isolados

Fase de Pré-processamento

Fase de Pré-Processamento

Fase com objetivos contraditórios

- incorporar o máximo de informação possível nas amostras
- reduzir ao máximo o seu tamanho, em linhas e colunas

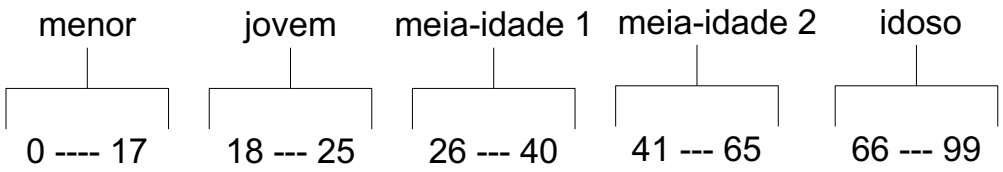
- A necessidade de pré-processamento de dados é determinada pelo tipo de modelo a desenvolver

- Alguns modelos como os baseados em árvores, são notoriamente insensíveis às características dos atributos previsores. Outros, como a regressão linear, não

Discretização dirigida por Classes

- **Discretização de valores numéricos**

  Valores inteiros ou reais em atributos categóricos (discretos)



Discretização dirigida por classes ("*class-driven*")

- Usa classes do atributo objetivo para determinar as fronteiras dos intervalos discretos

Discretização não sensível à classe ("*class-blind*")

- **Intervalos com igual largura**
  - não adequado para lidar com valores isolados
  - Pode criar desbalanceamento dos dados ao nível das classes
- **Intervalos com igual frequência de objetos**

Exemplo: Dados Iris

- Conjunto de dados Iris (150 registos)
  - Três tipos de flores (classes): Setosa, Virginica, Versicolour
  - Quatro atributos:
    - Sepal width, Sepal length
    - Petal width, Petal length

sepal length	sepal width	petal length	petal width	class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor
6.3	3.3	6.0	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
7.1	3.0	5.9	2.1	Iris-virginica
6.3	2.9	5.6	1.8	Iris-virginica

Conjunto de dados Iris

Modelo Inicial

```
default: Iris-setosa
except
 if petal-length>=2.45 and petal-length<5.355 and petal-width<1.75
 then Iris-versicolor
 except if petal-length >= 4.95 and petal-width < 1.55
        then Iris-virginica
        else
            if sepal-length < 4.95 and sepal-width >= 2.45
            then Iris-virginica
            else
                if petal-length >= 3.35
                then Iris-virginica
                except if petal-length < 4.85
                           and sepal-length < 5.95
                       then Iris-versicolor
```

Exemplo: Dados Iris

- **Discretização dos atributos**
  - petal width: [0, 0.75) *low*, [0.75, 1.75) *medium*, [1.75, ∞) *high*
  - petal length : [0, 2.5) *low*, [2.5, 5) *medium*, [5, ∞) *high*

Petal Length	Petal Width	Species Type	Count
low	low	Setosa	46
low	medium	Setosa	2
medium	low	Setosa	2
medium	medium	Versicolour	43
medium	high	Versicolour	3
medium	high	Virginica	3
high	medium	Versicolour	2
high	medium	Virginica	3
high	high	Versicolour	2
high	high	Virginica	44

Conjunto de dados Iris

Modelo Final

```
PetLenght = low => Setosa
PetLenght = high => Virginica
PetLenght = medium
  PetWidht = low => Versicolor
  PetWidht = high
    sepal width <= 3.1 => Virginica
    sepal width   > 3.1 => Versicolor
  PetWidht = medium  => Versicolor
```

Discretização

Algumas Considerações

- Intervalos com igual frequência de objetos regra geral dá melhor resultados
- Discretização dirigida por classes é mais adequada para problemas de classificação – os algoritmos de árvore de decisão fazem-no
- Perda de informação
  - distinção entre objectos de uma mesma categoria
  - amplitude da diferença entre objectos de categorias diferentes
- Maior estabilidade dos dados
- Muitos outros métodos existem …

Redução do Número de Linhas das Amostras

Generalização de Atributos Categóricos

Um atributo é generalizável se contém um número elevado de valores e se existe uma hierarquia entre estes, i.e. se existem conceitos de mais alto nível que resumam os valores desses atributos

As hierarquias largamente utilizadas nos cubos OLAP representam conhecimento útil que deve ser usado no processo de generalização dos dados

Discretização/Generalização

Vantagens

- pode ser usada por vários algoritmos – ganha em generalidade e reutilização
- estende a faixa de algoritmos de Data Mining aplicáveis
- melhora a compreensibilidade do conhecimento extraído - as regras são definidas em termos de conceitos mais genéricos - torna-as mais simples e explícitas para os utilizadores
- reduz o tempo gasto pelo algoritmo de Data Mining no paradigma de indução de regras
- pode reduzir a quantidade de ruído nos dados
- reduz o número de regras geradas

Desvantagens

- reduz a precisão do conhecimento descoberto

Redução do Número de Colunas

O objetivo é selecionar um subconjunto de atributos relevantes para o objetivo descoberta de entre todos os atributos disponíveis

Motivações

- Aumentar o grau de correção
  - ➥ taxa de acerto do conhecimento descoberto
- Reduzir o tempo gasto pelos algoritmos de Data Mining

Redução do Número de Colunas

Mais atributos nas amostras de treino pode resultar em conhecimento menos correto:

- alguns atributos podem ser **ruidosos** – o que confunde os algoritmos de Data Mining
- atributos **irrelevantes** podem impedir a terminação dos algoritmos de Data Mining
- alguns atributos podem ser **redundantes**, resultado da integração de várias fontes
- Amostras com uma grande dimensionalidade é sinónimo de dados **esparsos**

Técnicas de Redução do Nº de Colunas

- Seleção Manual de Atributos pelo Analista
- Eliminação de Falsos Previsores
- Remover as colunas que não apresentem qualquer variação, ou muito pouca variação
- Eliminação de variáveis correlacionadas
- Combinação de Variáveis de Entrada
- Testes estatísticos

- Análise dos Componentes Principais
- Algoritmos Feature Selection:
  - Random Forest
  - Relief algorithm
  - …

Previsores Correlacionados

Dois previsores altamente correlacionados (**colineares**) significa que representam a mesma informação subjacente e frequentemente acrescentam mais complexidade ao modelo do que informação

A remoção de um deles não compromete o desempenho do modelo e pode levar a um modelo de mais fácil interpretação

É possível obter relações de colinearidade entre vários previsores ao mesmo tempo (**multicolinearidade**) – representada por uma matriz de correlação

A correlação (*Pearson's*) entre dois atributos A e B mede a relação linear entre esses dois atributos

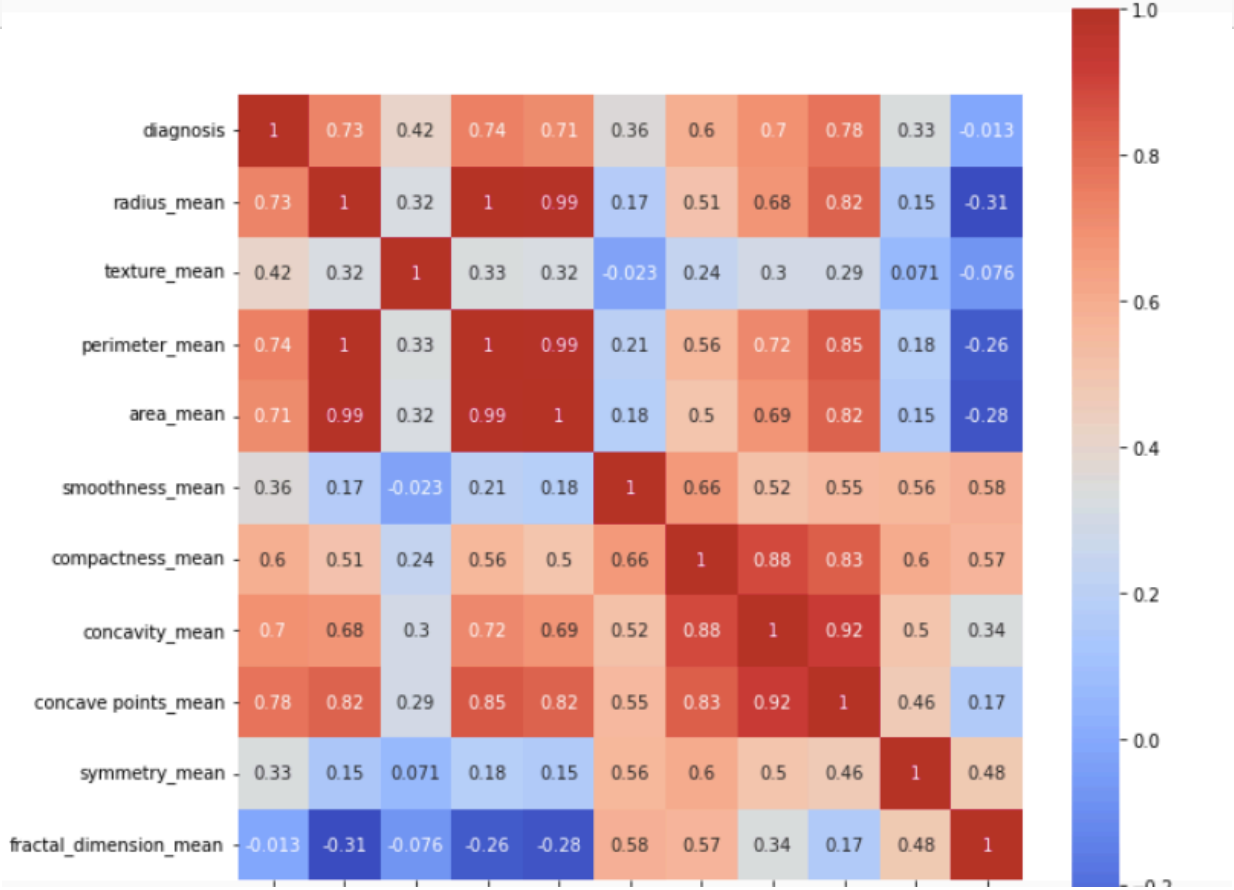
$$r_{A,B} = \frac{\sum (A - \bar{A})(B - \bar{B})}{(n-1)\sigma_A \sigma_B}$$

$$\text{em que } \bar{A} = \frac{\sum A}{n} \quad \text{e} \quad \sigma_A = \sqrt{\frac{\sum (A - \bar{A})^2}{n-1}}$$

Matriz de Correlação

```
# Seaborn to create a heat map of the correlations between the mean features

plt.figure(figsize=(10,10))
sns.heatmap(df[features_mean].corr(), annot=True, square=True, cmap='coolwarm')
plt.show()
```



Combinação de Variáveis

Combinar uma ou mais variáveis independentes numa única variável de entrada

Exemplo

Altura	Comprimento	Largura	Caixa	Altura	Comprimento	Largura	Caixa
2	12	2	Class1	12	4	2	Class2
6	4	2	Class1	4	12	2	Class2
3	8	2	Class1	8	6	2	Class2
4	4	3	Class1	4	8	3	Class2

Possíveis regras de classificação:

```
se (Altura <= 3)
     então Classe 1
 senão
    se (Comprimento <= 4)
       então Classe 1
    senão
       então Classe 2
```

Regras Proposicionais

Combinação de Variáveis

Derivando um novo atributo: A*C*L

Altura	Comprimento	Largura	A*C*L	Caixa	Altura	Comprimento	Largura	A*C*L	Caixa
2	12	2	48	Class1	12	4	2	96	Class2
6	4	2	48	Class1	4	12	2	96	Class2
3	8	2	48	Class1	8	6	2	96	Class2
4	4	3	48	Class1	4	8	3	96	Class2

Duas regras de classificação bem mais simples:

```
Se (A*C*L <= 48)
   então Classe 1
Se (A*C*L > 48)
    então Classe 2
```

Extração de **regras relacionais** só é conseguida com pré-processamento adequado – **combinação de variáveis**

Testes Estatísticos

Para determinar a força do relacionamento entre variáveis

- **Matriz de correlação**
  - contínuo vs contínuo
- **ANOVA test**
  - contínuo vs categórico
- **Chi-Square test**
  - categórico vs categórico

ANOVA teste (categórico vs. contínuo)

A análise de variância (ANOVA) é realizada para verificar se existe alguma relação entre uma variável contínua e uma variável categórica

Hipótese H0: $\mu_1 = \mu_2 = \mu_3 = \ldots = \mu_n$

Hipótese H1: $\mu i \neq \mu j \ (para\ alguns\ i \neq j)$

O teste estatístico ANOVA calcula a estatística F, que é a razão entre a variabilidade entre grupos e a variabilidade dentro do grupo:

$$F = (SSB / (k - 1)) / (SSW / (n - k))$$

k é o número de grupos ou categorias na variável categórica

n é o número total de observações

SSB e SSW são as somas dos quadrados da variabilidade entre grupos e dentro do grupo, respetivamente.

ANOVA teste (categórico vs. contínuo)

A análise de variância (ANOVA) é realizada para verificar se existe alguma relação entre uma variável contínua e uma variável categórica

Hipótese H0: $\mu_1 = \mu_2 = \mu_3 = \ldots = \mu_n$

Hipótese H1: $\mu i \neq \mu j \ (para\ alguns\ i \neq j)$

Teste de hipótese:

Se o **valor p for menor** que o nível de significância escolhido (α = 0.05),
a **hipótese nula (H0) é rejeitada**

=> existe uma relação significativa entre a var. categórica e a var. contínua

Se **o valor p for maior que α**,

a hipótese nula não é rejeitada, sugerindo que não há relação significativa.

Python: ANOVA function

```
def FunctionAnova (inpData, TargetVar, PredictorList):
    from scipy.stats import f_oneway
    SelectedPredictors=[]  # empty list of final selected predictors

    print('##### ANOVA Results ##### \n')
    for predictor in PredictorList:
        CategoryGroupLists=inpData.groupby(TargetVar)[predictor].apply(list)
        AnovaResults = f_oneway(*CategoryGroupLists)

        # If the ANOVA P-Value is < 0.05, that means we reject H0
        if (AnovaResults[1] < 0.05):
            print(predictor, 'is correlated with', TargetVar, '| P-Value:’,
                                                              AnovaResults[1])
            SelectedPredictors.append(predictor)
        else:
            print(predictor, 'is NOT correlated with', TargetVariable, '| P-Value:’,
                                                                  AnovaResults[1])
    return(SelectedPredictors)
```

Feature Selection Algorithms

- Recursive Feature Elimination (RFE)
- Principal Component Analysis (PCA)
- Lasso (Least Absolute Shrinkage and Selection Operator)
- Random Forest Importance

Análise dos Componentes Principais (ACP)

Ideia Geral

- Descrever (a maior parte) da variação de um conjunto de dados, com múltiplas variáveis, usando um conjunto menor (mais conciso) de variáveis

- ACP pode ser pensada como uma forma de **redução de dados** - usa-se um conjunto menor de variáveis que contêm a informação relevante que está nos dados completos

- Antes da aplicação da ACP num **conjunto de dados** estes devem estar **centrados e normalizados**, isto permite que a ACP encontre os relacionamentos subjacentes nos dados sem serem influenciados pelas escalas de medição originais e diferenças de distribuição dos previsores

Análise dos Componentes Principais (ACP)

- A ACP cria um novo conjto de variáveis $y_1, y_2, \ldots, y_q$ em que cada uma é uma combinação linear das variáveis originais $x_1, x_2, \ldots, x_n$ com $q < n$

$$y_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n$$

$$y_2 = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n$$

$$\ldots$$

$$y_q = a_{q1}x_1 + a_{q2}x_2 + \ldots + a_{qn}x_n$$

- As novas variáveis devem ser não correlacionadas por forma a que a informação em y_2, por exemplo, não sobreponha a informação em y_1
- Procuram-se as combinações lineares que "expliquem" a maior parte da "variabilidade" que existe nos dados sobre os eixos originais
- Com "sorte", com apenas alguns desses novos eixos, **idealmente $q << n$ para fácil visualização dos dados**, é possível explicar a maior parte da variabilidade nos dados originais
- Cada observação original é então projetada para estes novos eixos

Análise dos Componentes Principais (ACP)

Algoritmo

- Encontrar uma primeira combinação linear que melhor capta a variabilidade dos dados
- Passar para uma segunda combinação linear de modo a tentar capturar a variabilidade não explicada na primeira combinação linear
- Continuar até que o conjunto de novas variáveis explique a maior parte da variabilidade do conjunto de dados original (normalmente **90% representatividade** do conjunto é suficiente)

Análise dos Componentes Principais (ACP)

- A ACP ao resumir a variabilidade não considera o atributo objetivo - é uma **técnica não supervisionada**

- Se o relacionamento entre os atributos previsores e o atributo objectivo não for descrito pela variabilidade dos previsores, então, os CPs derivados não fornecerão uma relação adequada com atributo objectivo

- Neste caso é necessário usar a técnica supervisionada – *Partial Least Squares* (PLS) que deriva os componentes considerando ao mesmo tempo o correspondente atributo objectivo

Preparação de Dados – Ideias Chave

- Construir amostras de dados representativas do conhecimento que se pretende extrair dos dados
- Explorar graficamente os dados para inspecionar anomalias e erros e perceber tendências nos dados
- Eliminar "falsos positivos"
- Desenvolver componentes de software pequenos e reutilizáveis
- Verificar os resultados após cada passo

↳ Uma boa preparação dos dados é essencial para produzir modelos válidos e úteis