04. Aprendizagem não Supervisionada

IC 22/23

ISEC - CPereira

Aprendizagem não supervisionada

- A aprendizagem não supervisionada inclui todos os tipos de aprendizagem onde não há saída (*label*) conhecida.
  - O algoritmo gera uma função para identificar estruturas ocultas no conjunto de dados fornecido, de acordo com os padrões, semelhanças e diferenças que existem entre os dados sem qualquer treino anterior.
- Tipos de aprendizagem não supervisionada:
  - **Transformação de Dados** - criar uma nova representação dos dados, mais fáceis para visualizar ou processar. em comparação com a representação original. Exemplos:
    - Redução a duas dimensões para visualização. Determina as componentes que “caracterizam" os dados.
    - Extração de tópicos em coleções de documentos de texto.
  - **Clustering** (Agrupamento) - particionar os dados em grupos de itens semelhantes e distintos entre si. Exemplos:
    - agrupar imagens que mostram a mesma pessoa. Não existindo um “label”, a solução passa por extrair todas as imagens e dividi-las em grupos que aparência semelhante.

Aprendizagem não supervisionada

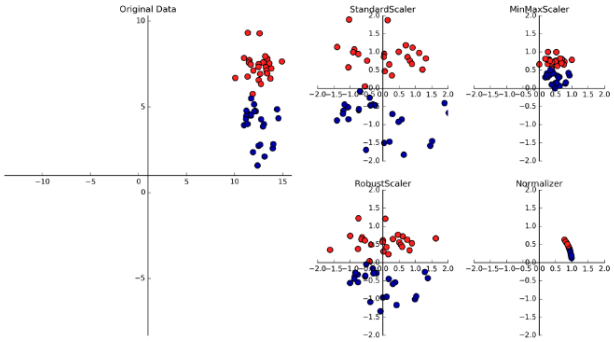
- Desafios
  - Como avaliar se o algoritmo produziu algo útil?
    - Assim, são usados frequentemente num ambiente exploratório, quando se pretende entender os dados.
  - Usados como etapa de pré-processamento para algoritmos supervisionados
    - uma nova representação dos dados pode:
      - melhorar a precisão dos algoritmos supervisionados (como por exemplo redes neuronais)
      - levar a menor consumo de memória
      - tempo de processamento mais reduzido.

Pré-processamento e ajuste de escala

- Alguns algoritmos, como redes neuronais (MLP, RBF) e SVMs, são sensíveis ao dimensionamento (escala) dos dados.
  - No scikit-learn
    - StandardScaler
      - garante que para cada característica-"feature" a média é 0 e a variância (média do quadrado dos desvios) é 1.
    - RobustScaler
      - funciona de forma semelhante ao StandardScaler, no entanto, usa a mediana e os quartis, e assim ignora as amostras muito "distantes" das restantes ("outliers", que podem surgir por erros de medição, ruído, etc…).
    - MinMaxScaler
      - desloca os dados de modo que todas as "features" sejam entre 0 e 1.
    - Normalizer
      - dimensiona cada ponto de modo que o vetor de características tenha um comprimento euclidiano de 1.

Pré-processamento e ajuste de escala

- …



Fonte: Muller and Guido, 2016

Pré-processamento e ajuste de escala

- …
  - https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

```
>>> from sklearn.preprocessing import StandardScaler
>>> data = [[0, 0], [0, 0], [1, 1], [1, 1]]
>>> scaler = StandardScaler()
>>> print(scaler.fit(data))
StandardScaler()
>>> print(scaler.mean_)
[0.5 0.5]
>>> print(scaler.transform(data))
[[-1. -1.]
 [-1. -1.]
 [ 1.  1.]
 [ 1.  1.]]
>>> print(scaler.transform([[2, 2]]))
[[3. 3.]]
```

Pré-processamento e ajuste de escala

- …

```
>>> from sklearn.preprocessing import RobustScaler
>>> X = [[ 1., -2.,  2.],
...      [ -2.,  1.,  3.],
...      [ 4.,  1., -2.]]
>>> transformer = RobustScaler().fit(X)
>>> transformer
RobustScaler()
>>> transformer.transform(X)
array([[ 0. , -2. ,  0. ],
       [-1. ,  0. ,  0.4],
       [ 1. ,  0. , -1.6]])
```

```
>>> from sklearn.preprocessing import MinMaxScaler
>>> data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]
>>> scaler = MinMaxScaler()
>>> print(scaler.fit(data))
MinMaxScaler()
>>> print(scaler.data_max_)
[ 1. 18.]
>>> print(scaler.transform(data))
[[0.   0.  ]
 [0.25 0.25]
 [0.5  0.5 ]
 [1.   1.  ]]
>>> print(scaler.transform([[2, 2]]))
[[1.5 0. ]]
```

Pré-processamento e ajuste de escala

- …
  - Normaliza as amostras individualmente para a unidade.
  - Cada amostra com pelo menos um componente diferente de zero é redimensionada independentemente de outras amostras de modo que sua norma seja igual a um.
  - comum para classificação de texto, onde o produto escalar de dois vetores normalizados é comumente usada para análise de similaridade.
- Outras metodologias e estudo comparativo
  - https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py

```
>>> from sklearn.preprocessing import Normalizer
>>> X = [[4, 1, 2, 2],
...      [1, 3, 9, 3],
...      [5, 7, 5, 1]]
>>> transformer = Normalizer().fit(X)  # fit does nothing.
>>> transformer
Normalizer()
>>> transformer.transform(X)
array([[0.8, 0.2, 0.4, 0.4],
       [0.1, 0.3, 0.9, 0.3],
       [0.5, 0.7, 0.5, 0.1]])
```

Exemplo

- Winscosin data set
  - Aplicar o mesmo método de processamento aos dados de treino!
    - Impacto no classificador:

```
from sklearn.svm import SVC

X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,
                                                    random_state=0)

svm = SVC(C=100)
svm.fit(X_train, y_train)
print("Test set accuracy: {:.2f}".format(svm.score(X_test, y_test)))
```

No conjunto de teste, accuracy=0.63

Exemplo

- …

```
# preprocessing using 0-1 scaling
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# learning an SVM on the scaled training data
svm.fit(X_train_scaled, y_train)

# scoring on the scaled test set
print("Scaled test set accuracy: {:.2f}".format(
    svm.score(X_test_scaled, y_test)))
```

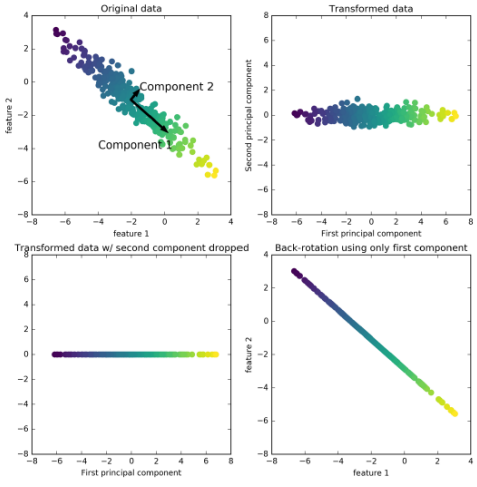
- Accuracy de 0.97!

PCA – Principal Component Analysis

- Princípios
  - Rotação do conjunto de dados de forma a que as “features” após a rotação não se encontrem correlacionadas
  - Esta rotação é frequentemente seguida pela seleção de um subconjunto das novas “features”, de acordo com a sua importância para “explicar” os dados.
    - O algoritmo determina primeiro a direção da variação máxima (com maior informação)- “Componente 1”.
    - De seguida determina a segunda componente – direção que contém mais informação sendo perpendicular com a primeira (note-se que existem várias perpendiculares num espaço multi-dimensional!)
    - As direções encontradas usando este processos são designadas de componentes principais - pois são as direções de maior variância. Em geral, existem tantos componentes principais quanto o número de “features” originais.

PCA

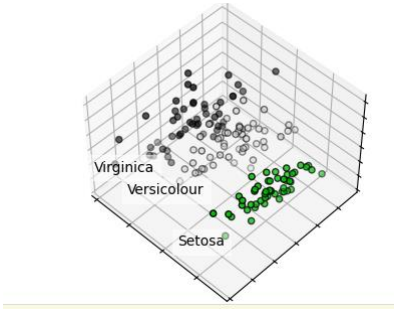
- …
  - (1) dados originais e componentes principais
  - (2) Os mesmos dados, mas agora com rotação, para que o primeiro componente principal se alinha com o eixo x e o segundo componente principal a linha-se com o eixo y
  - (3-4) Podemos efetuar uma redução d dimensionalidade, retendo apenas alguns dos principais componentes - Neste caso, mantemos apenas o primeiro componente principal



Fonte: Muller and Guido, 2016

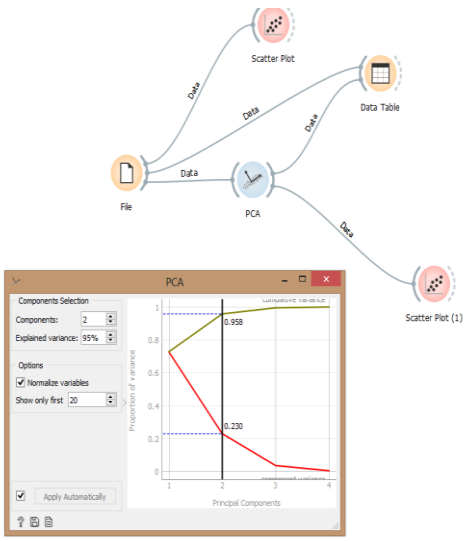
PCA

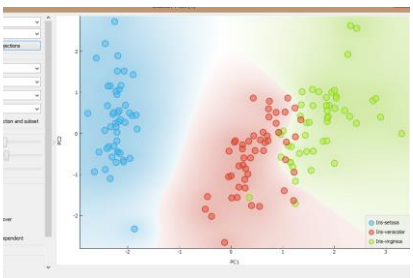
- Exemplo – Iris Dataset
  - https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
  - https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_iris.html#sphx-glr-auto-examples-decomposition-plot-pca-iris-py



PCA

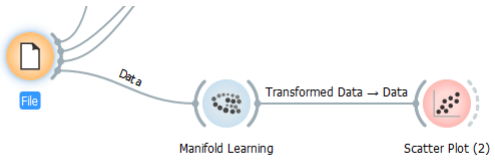
- Iris

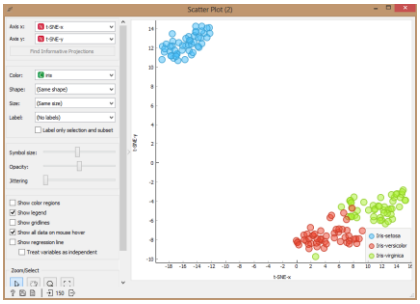




Manifold Learning

- Apenas para análise e visualização de dados
  - A ideia é encontrar uma representação bidimensional dos dados que preserva as distâncias entre os pontos da melhor forma possível.
    - Não se aplica ao conjunto de teste;
    - Para posterior aplicação de um classificador.



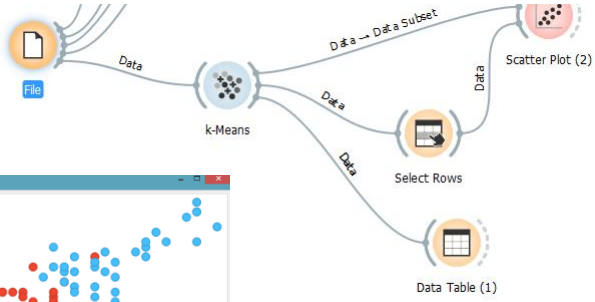


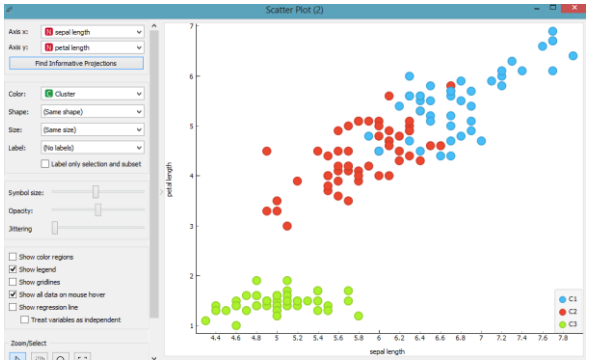
Clustering

- K-means
  - O algoritmo de agrupamento mais simples.
  - Tenta encontrar centros de clusters que sejam representativos de certas regiões dos dados.
    - O algoritmo intercala duas etapas:
      - atribuir cada ponto (instância ou amostra) ao centro de cluster mais próximo;
      - em seguida, definir cada centro de cluster como a média dos pontos que lhe são atribuídos a ele.
    - O algoritmo termina quando a atribuição de instâncias para clusters não se altera.

Clustering

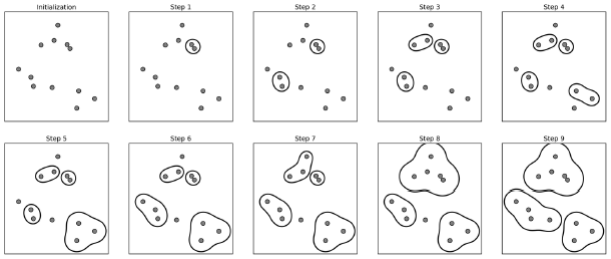
- …





Clustering

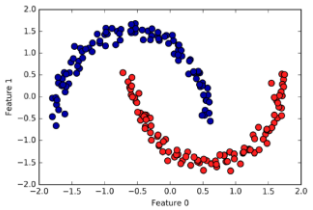
- Clustering aglomerativo
  - Refere-se a um conjunto de algoritmos, que seguem o seguinte princípio:
    - o algoritmo inicia atribuindo a cada ponto o seu próprio cluster;
    - De seguida, aglomera os dois clusters mais semelhantes, até que o critério de paragem seja satisfeito (usualmente o número de clusters pretendido)



Fonte: Muller and Guido, 2016

Clustering

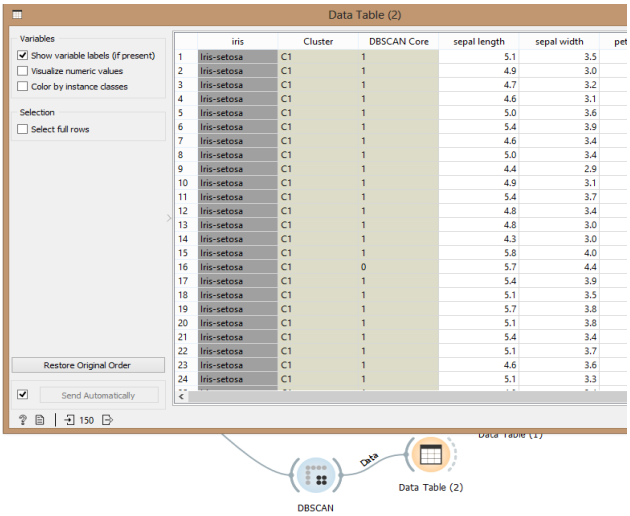
- DBSCAN
  - Não exige a definição prévia do número de clusters.
    - Identifica pontos que estão em regiões “densas“ (lotadas) – regiões onde muitos pontos estão próximos.
    - Baseia-se no princípio de que “os clusters formam regiões densas de pontos, separadas por regiões relativamente vazias”.
      - Se houver pelo menos “min_samples” num raio de “eps” para um determinado ponto, esse ponto é classificado como “core”. Amostras “core” a uma distância inferior a “eps” são colocadas no mesmo cluster.



Fonte: Muller and Guido, 2016

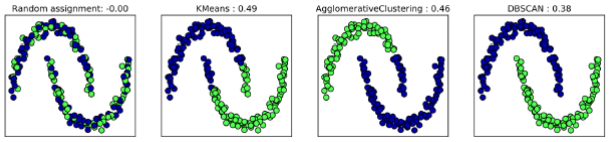
Clustering

- …



Clustering

- Análise comparativa
  - Cada um dos algoritmos tem os seus pontos fortes:
    - K-means é simples e eficiente. Também pode ser visto como uma decomposição, onde cada ponto é representado pelo seu centro de grupo.
    - O agrupamento aglomerativo pode fornecer toda uma hierarquia de possíveis partições dos dados, que podem ser facilmente inspecionadas por meio de dendrogramas.
    - DBSCAN permite a deteção de "pontos de ruído" que não são atribuídos a nenhum cluster, e ajuda a determinar automaticamente o número de clusters. Permite ainda formas de cluster complexas - exemplo de “two_moons dataset”:



Fonte: Muller and Guido, 2016

Referências

- Müller, Andreas C., and Sarah Guido. *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.
- https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html