Machine Learning Package

Portfolio de algoritmos de Machine Learning

Sumário

- Feature selection/filtragem consiste em selecionar/reduzir o nº de variáveis no dataset.
- No nosso portefólio, métodos de feature selection podem seguir a estrutura de um ***Transformer***.
- Arquitetura de um ***Transformer***:
  - Parâmetros – conjunto de parâmetros definidos pelo utilizador
  - Parâmetros estimados – conjunto de parâmetros/atributos estimados a partir dos dados
  - Fit – método responsável por estimar parâmetros a partir dos dados
  - Transform – método responsável por transformar os dados

Datasets

- Os datasets estão disponíveis em:
  - https://www.dropbox.com/sh/oas4yru2r9n61hk/AADpRunbqES44W49gx9deRN5a?dl=0

Objeto VarianceThreshold

- Na pasta *feature_selection*, adiciona o modulo *variance_threshold.py* que deve conter o objeto *VarianceThreshold.*

- *class VarianceThreshold:*
  - Parâmetros:
    - threshold – linha de corte/valor de corte
  - Parâmetros estimados:
    - variance – a variância de cada feature
  - Métodos:
    - fit – estima/calcula a variância de cada feature; retorna o self (ele próprio)
    - transform – seleciona todas as features com variância superior ao threshold e retorna o X selecionado
    - fit_transform – corre o fit e depois o transform

statistics sub-package

- Adiciona agora outro sub-package chamado *statistics* com o módulo chamado *f_classification.py.* Vamos adicionar métodos para analisar a variância do nosso dataset.

- *def f_classification*
  - assinatura/argumentos:
    - dataset – o dataset
  - ouput esperado:
    - Tuplo com valores de F + Tuplo com valores de p
  - algoritmo:
    - agrupa as samples/exemplos por classes. Podes usar o método get_classes() do dataset e depois selecionar as samples de cada classe numa lista
    - usa a função *scipy.stats.f_oneway*. Esta função retorna os valores de F e os valores de p

Objeto SelectKBest

- Na pasta *feature_selection*, adiciona o modulo *select_k_best.py* que deve conter o objeto *SelectKBest.*

- *class SelectKBest:*
  - Parâmetros:
    - score_func – função de análise da variância (*f_classification*)
    - k – número de features a selecionar
  - Parâmetros estimados:
    - F – o valor de F para cada feature estimado pela score_func
    - p – o valor de p para cada feature estimado pela score_func
  - Métodos:
    - fit – estima o F e p para cada feature usando a scoring_func; retorna o self (ele próprio)
    - transform – seleciona as k features com valor de F mais alto e retorna o X selecionado
    - fit_transform – corre o fit e depois o transform

Avaliação

- Exercício 3: Implementar o *SelectPercentile*
  - 3.1) Adiciona o objeto *SelectPercentile* ao sub-package *feature_selection*. Deves criar um módulo chamado *select_percentile.py* para implementar este objeto
  - 3.2) A class *SelectPercentile* tem uma arquitetura semelhante à classe *SelectKBest*. Considera a estrutura apresentada no diapositivo seguinte.
  - 3.3) Podes testar a class SelectPercentile num jupyter notebook usando o dataset iris.csv (classificação)

Objecto *SelectPercentile*

- *class SelectPercentile:*
  - Parâmetros:
    - score_func – função de análise da variância (*f_classification*)
    - percentile – percentil para as features a selecionar
  - Parâmetros estimados:
    - F – o valor de F para cada feature estimado pela score_func
    - p – o valor de p para cada feature estimado pela score_func
  - Métodos:
    - fit – estima o F e p para cada feature usando a scoring_func; retorna o self (ele próprio)
    - transform – seleciona as features com valor de F mais alto até ao percentil indicado. Por exemplo, para um dataset com 10 features e um percentil de 50%, o teu transform deve selecionar as 5 features com valor de F mais alto
    - fit_transform – corre o fit e depois o transform