Machine Learning Package

Portfolio de algoritmos de Machine Learning

Sumário

- A validação cruzada permite avaliar de maneira credível o desempenho de um modelo num determinado *dataset*
- Iremos implementar a validação cruzada *k-fold* – *cross_validate*
- A otimização de parâmetros consiste num processo de seleção dos melhores modelos que minimizam o erro
- Iremos implementar a procura em grelha – *grid_search_cv*

Datasets

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

cross_validate

- Adiciona o módulo *cross_validate.py* ao sub-package *model_selection*
- *def cross_validate:*
  - assinatura/argumentos:
    - model – modelo a validar
    - dataset – dataset de validação
    - scoring – função de score
    - cv – número de *folds*
    - test_size – tamanho do dataset de teste
  - ouput esperado:
    - Um dicionário com os scores de treino e teste
  - algoritmo:
    - Ver slide seguinte

cross_validate

- O algoritmo do *cross_validate*:
  1. Obtém uma seed/random_state usando o np.random.randint
  2. Divide o dataset em treino e teste usando a seed gerada anteriormente e o tamanho do dataset de teste
  3. Treina o modelo
  4. Obtém o score do modelo no dataset de treino. Usa a função de score
  5. Obtém o score do modelo no dataset de teste. Usa a função de score
  6. Repete os passos anteriores para todos os folds (cv)

- O *cross_validate* deve retornar um dicionário com as seguintes chaves:
  - *seeds*: as seeds geradas para cada fold
  - *train*: os scores do modelo no dataset de **treino** para cada fold
  - *test*: os scores do modelo no dataset de **teste** para cada fold

Teste *cross_validate*

- *cross_validate*:
  1. Usa o dataset *breast-bin.csv*
  2. Usa o *sklearn.preprocessing.StandardScaler* para standardizar os dataset.
     breast_dataset.X = StandardScaler().fit_transform(breast_dataset.X)
  3. Cria o modelo *LogisticRegression*
  4. Realiza uma validação cruzada com 5 folds
  5. Quais os scores obtidos?

grid_search_cv

- Adiciona o módulo *grid_search.py* ao sub-package *model_selection*
- *def grid_search_cv:*
  - assinatura/argumentos:
    - model – modelo a validar
    - dataset – dataset de validação
    - parameter_grid – os parâmetros para a procura. Dicionário com nome do parâmetro e valores de procura
    - scoring – função de score
    - cv – número de *folds*
    - test_size – tamanho do dataset de teste
  - ouput esperado:
    - Uma lista de dicionários com a combinação dos parâmetros e os scores de treino e teste
  - algoritmo:
    - Ver slide seguinte

grid_search_cv

- O algoritmo do *grid_search*:
  1. Verifica se os parâmetros fornecidos existem no modelo.
     Podes usar a função do python *hasattr*.
  2. Obtém o produto cartesiano dos parâmetros fornecidos (todas as combinações possíveis).
     Podes usar o *itertools.product* para obter todas as combinações.
  3. Altera os parâmetros do modelo com uma combinação.
     Podes usar a função do python *setattr.*
  4. Realiza o *cross_validate* com esta combinação
  5. Guarda a combinação de parâmetros e os scores obtidos.
  6. Repete os passos 3, 4 e 5 para todas as combinações.

- O *grid_search* deve retornar uma lista de dicionários. Os dicionários devem conter as seguintes chaves:
  - parameters: a combinação de parâmetros
  - seeds: as seeds geradas para cada fold
  - train: os scores do modelo no dataset de **treino** para cada fold
  - test: os scores do modelo no dataset de **teste** para cada fold

Teste *grid_search_cv*

- *grid_search_cv*:
  1. Usa o dataset *breast-bin.csv*
  2. Usa o *sklearn.preprocessing.StandardScaler* para standardizar os dataset.
     breast_dataset.X = StandardScaler().fit_transform(breast_dataset.X)
  3. Cria o modelo *LogisticRegression*
  4. Realiza uma procura em grelha com os seguintes parâmetros:
     - l2_penalty: 1, 10
     - alpha: 0.001, 0.0001
     - max_iter: 1000, 2000
  5. Podes usar 3 folds para o *cross_validate*
  6. Quais os scores obtidos?

Avaliação

- Exercício 8: Adiciona o método *randomized_search_cv*.
  - O método *randomized_search_cv* implementa uma estratégia de otimização de parâmetros de usando Nº combinações aleatórias. O *randomized_search_cv* avalia apenas um conjunto aleatório de parâmetros retirados de uma distribuição ou conjunto de valores possíveis.
  - 8.1) Considera a estrutura e algoritmo do *randomized_search_cv* apresentados nos slides seguintes
  - 8.2) Valida a tua implementação seguindo o protocolo:
    1. Usa o dataset *breast-bin.csv*
    2. Usa o *sklearn.preprocessing.StandardScaler* para standardizar os dataset.
       breast_dataset.X = StandardScaler().fit_transform(breast_dataset.X)
    3. Cria o modelo *LogisticRegression*
    4. Realiza uma procura aleatória com as seguintes distribuições de parâmetros:
       - l2_penalty: distribuição entre 1 e 10 com 10 intervalos iguais (e.g., *np.linspace(1, 10, 10)*)
       - alpha: distribuição entre 0.001 e 0.0001 com 100 intervalos iguais (e.g., *np.linspace(0.001, 0.0001, 100)*)
       - max_iter: distribuição entre 1000 e 2000 com 200 intervalos iguais (e.g., *np.linspace(1000, 2000, 200)*)
    5. Podes usar *n_iter* de 10 e 3 folds para o *cross_validate*.
    6. Quais os scores obtidos?

randomized_search_cv

- Adiciona o módulo *randomized_search.py* ao sub-package *model_selection*
- *def randomized_search_cv:*
  - assinatura/argumentos:
    - model – modelo a validar
    - dataset – dataset de validação
    - parameter_distribution – os parâmetros para a procura. Dicionário com nome do parâmetro e distribuição de valores
    - scoring – função de score
    - cv – número de *folds*
    - n_iter – número de combinações aleatórias de parâmetros
    - test_size – tamanho do dataset de teste
  - ouput esperado:
    - Uma lista de dicionários com a combinação dos parâmetros e os scores de treino e teste
  - algoritmo:
    - Ver slide seguinte

randomized_search_cv

- O algoritmo do *randomized_search*:
  1. Verifica se os parâmetros fornecidos existem no modelo.
  2. Obtém *n_iter* combinações de parâmetros. Ou seja, se *n_iter* for igual a 10 deves obter 10 combinações dos parâmetros fornecidos. Podes usar a função *np.random.choice* do *numpy* para retirar um valor aleatório da distribuição de valores de cada parâmetro.
  3. Altera os parâmetros do modelo com uma combinação.
  4. Realiza o *cross_validate* com esta combinação.
  5. Guarda a combinação de parâmetros e os scores obtidos.
  6. Repete os passos 3, 4 e 5 para todas as combinações.

- O *randomized_search* deve retornar uma lista de dicionários. Os dicionários devem conter as seguintes chaves:
  - *parameters*: a combinação de parâmetros
  - *seeds*: as seeds geradas para cada fold
  - *train*: os scores do modelo no dataset de **treino** para cada fold
  - *test*: os scores do modelo no dataset de **teste** para cada fold