Dokumentacja programu

Ogólny opis działania

Program ma za zadanie na podstawie otrzymanego zbioru testowego, który zawiera informacje o poszczególnych cechach komórek i rodzaju raka (łagodny lub złośliwy), przewidzieć w zbiorze testowym, który składa się tylko z cech komórek rodzaj raka. Do zaimplementowania powyższej funkcjonalności program używa algorytmu randomForest.

Specyfikacja wejściowa

**Dołączone biblioteki:**

* Biblioteka Pandas używana do wczytywania i wyprowadzania danych do plików
* Bibilioteka sklearn, która zawiera w sobie odpowiednie funkcje, które pozwalają zaimplementować algorytm randomForest

Na wejściu program pobiera trzy pliki:

* Plik *„Data/cancerTrain.csv”* zawierający w sobie zbiór na którym program będzie się uczył
* Plik *„Data/cancerTest.csv”* zawierający zbiór dla którego program będzie przewidywał rodzaj raka
* Plik *„Data/toCheck.csv”* zawierający prawidłowe wartości, które program miał przewidzieć dla pliku testowego. Używany do określenia macierzy pomyłek

Specyfikacja wyjściowa

Program zwraca plik *„predictions.csv”* zawierający w sobie dane wynikowe w postaci ID pacjenta i rodzaju raka, który został wyznaczony przez program.

Ponadto w terminalu poleceń zostaje wydrukowana macierz pomyłek dla powyższych danych

Specyfikacja wewnętrzna

**Funkcje:**

* create\_submission(…) – jak parametry pobiera obiekt zwracany przez funkcję RandomForestClassifier(), obiekt z danymi treningowymi, obiekt z danymi testowymi, obiekt z danymi do porównania, obiektem z nazwami kolumn, które mają zostać wykorzystane do prognozowania, nazwę pliku, do którego mają zostać wyeksportowane wyznaczone dane. Sama funkcja na podstawie przekazanych parametrów przewiduje czy ktoś ma raka i dane te eksportuje do odpowiedniego pliku.

**Wydruk programu z komentarzami:**

Na samym początku program wczytuje potrzebne biblioteki i importuje z nich odpowiednie funkcje (KFold – do 10-fold cross validation, confusion\_matrix – do wyznaczania macierzy pomyłek, RandomForestClassifier – do budowania drzew w lesie), później wczytuje odpowiednie dane z plików za pomocą funkcji read\_csv() z biblioteki pandas oraz wczytuje do obiektu nazwy kolumn na podstawie których program będzie przewidywał:

1. print('Importing libraries...')
2. import pandas as pd
3. from sklearn.cross\_validation import KFold
4. from sklearn.metrics import confusion\_matrix
5. from sklearn.ensemble import RandomForestClassifier
7. train\_data = pd.read\_csv("Data/cancerTrain.csv", delimiter=';', )
8. test\_data  = pd.read\_csv("Data/cancerTest.csv", delimiter=';', )
9. to\_check   = pd.read\_csv("Data/toCheck.csv", delimiter=';', )
10. (...)
11. predictors = ["ClumpThickness", "UniformityCellSize", "CellShape", "MarginalAdhesion", "CellSize", "BareNuclei", "BlandChromatin", "NormalNucleoli", "Mitoses"]

Następnie program stara się znaleźć najlepszy pod względem trafności prawidłowych przewidzeń współczynnik n, który określa liczbę drzew w lesie.

Przechodzi on w pętli for po różnych wartościach tegoż współczynnika (w naszym przypadku od 1 do 100, przy wyższych wartościach program działa bardzo wolno) i dla każdej z nich za pomocą funkcji KFold, wykonuje on 10-fold cross validation na zbiorze treningowym. W następnej pętli for każdy z tak otrzymanych podzbiorów staje się wewnętrznym zbiorem testowym, a reszta treningowym. Dla każdej z takich kombinacji zostaje wyznaczony rodzaj raka, a następnie za pomocą metody score() zostaje porównana trafność przewidzenia. Dla każdej kombinacji program wyznacza średnią arytmetyczną sumy tych wyników, a następnie porównuje z poprzednich najwyższym wynikiem. n, dla którego wynik ten był najwyższy zostaje ostatecznym n, które zostanie użyte podczas testowania właściwego zbioru testowego. Na końcu tego bloku kodu program wypisuje znalezioną wartość n i wynik jaki został dla niego osiągnięty:

1. print('Finding best n\_estimators for RandomForestClassifier...')
2. max\_score = 0
3. best\_n = 0
4. for n in range(1,100):
5. rfc\_scr = 0.
6. rfc = RandomForestClassifier(n\_estimators=n)
7. for train, test in KFold(len(train\_data), n\_folds=10, shuffle=True):
8. rfc.fit(train\_data[predictors].T[train].T, train\_data["Class"].T[train].T)
9. rfc\_scr += rfc.score(train\_data[predictors].T[test].T, train\_data["Class"].T[test].T)/10
10. if rfc\_scr > max\_score:
11. max\_score = rfc\_scr
12. best\_n = n
13. print(best\_n, max\_score)

W następnych linijkach program buduje las drzew decyzyjnych, i przekazuje odpowiednie parametry do funkcji create\_submission(), która przeprowadza analizę zbioru testowego i zwraca wynik swojego działania w postaci pliku:

1. rfc = RandomForestClassifier(n\_estimators=best\_n)
2. print('Creating submission...')
3. create\_submission(rfc, train\_data, test\_data, to\_check, predictors, "predictions.csv")
4. print('Submitted.')

Funkcja create\_submission() za pomocą metody fit() uczy się przewidywać kolumnę „Class”, w której są informację o rodzaju raka, na podstawie odpowiedniej listy kolumn. W kolejnym kroku program przewiduje rodzaj raka dla zbioru testowego, dalej program drukuje w terminalu poleceń macierz pomyłek. Następnie za pomocą metody DataFrame() z biblioteki pandas zostaje utworzony obiekt danych, w postaci << „PatientID”: …, „Class”: … >>, który następnie zostaje zapisany do pliku za pomocą metody to\_csv() również z biblioteki pandas.