Klasyfikator wspomagający diagnozę raka (Breast Cancer Wisconsin)

KWD

Karolina Banyś, Artur Kornelski

12.01.2021

Streszczenie

Niniejszy raport ma na celu przedstawienie informacji na temat projektu realizowanego w ramach przedmiotu Komputerowe Wspomaganie Decyzji. Jego tematyka obejmuje zagadnienia związane z nadzorowanym uczeniem maszynowym – klasyfikacją binarną. W dokumencie podano opis problemu, wprowadzenie teoretyczne, badania symulacyjne, podsumowanie oraz kod programu.

Spis treści

Rozdział 1

Wprowadzenie

Uczenie maszynowe to technologia, która - zamiast precyzyjnie programować komputery – uczy je wykonywania zadań na podstawie analizy danych. Uczenie maszynowe dzieli się na nadzorowane, nienadzorowane oraz ze wzmocnieniem.

Uczenie nadzorowane to takie, w którym uczy się pewną funkcję przewidywania wyniku na podstawie danych wejściowych, mając do dyspozycji przykładowe pary: dana wejściowa – wynik.

Algorytmy nienadzorowane uczą się na podstawie danych bez dostępu do poprawnych odpowiedzi. Wykorzystują duże, zróżnicowane zbiory danych do samodoskonalenia.

Uczenie ze wzmocnieniem ma na celu uczenie się wiedzy proceduralnej (umiejętności). Jest to obliczeniowe podejście do uczenia się celowego zachowania na podstawie interakcji.

1.1 Opis problemu

W projekcie tym zajęliśmy się nadzorowanym uczeniem maszynowym, a dokładniej mówiąc klasyfikacją binarną. Celem tego projektu było napisanie klasyfikatora wspomagającego diagnozę raka w języku Python, korzystając z bazy danych z repozytorium o nazwie Breast Cancer Wisconsin. Baza ta zawiera 569 pacjentów, a każdy z nich jest opisany za pomocą 30 etykiet (cech), takich jak np. tekstura, promień, obwód czy powierzchnia zmiany rakowej. Dla każdego pacjenta znany jest również wynik testu na raka, czyli czy jest to złośliwa czy łagodna zmiana rakowa. W celu stworzenia modelu do rozpoznawania zmiany rakowej, bazę 569 pacjentów podzieliliśmy na zbiór uczący (treningowy) oraz testujący, a po uzyskaniu wyników dla zbioru testującego przeprowadziliśmy analizę tych danych.

Rozdział 2

Opis metody

2.1 Wprowadzenie teoretyczne

Na potrzeby tego projektu skupiliśmy się na uczeniu nadzorowanym. Zawiera ono w sobie dwie kategorie zadań: regresję i klasyfikację. Regresja stara się przewidzieć wartość liczbową (ciągłą), np.: wartość nieruchomości czy kurs akcji. Klasyfikacja na bazie danych wejściowych wyznacza kategorie. Na przykład na podstawie obrazka z ręcznie zapisaną cyfrą, określa jaka to cyfra.

Rodzaje klasyfikacji:

- Klasyfikacja binarna - odnosi się do zadań klasyfikacyjnych, które mają dwie etykiety klas.

- Multiklasyfikacja - odnosi się do zadań klasyfikacyjnych, które mają więcej niż dwie etykiety klas.

Regresja logistyczna to uczenie nadzorowane, ale wbrew swojej nazwie nie jest regresją, a metodą klasyfikacyjną. Zakłada ona, że dane mogą być sklasyfikowane (rozdzielone) poprzez linię lub płaszczyznę n-wymiarową, czyli jest modelem liniowym. Innymi słowy klasyfikacja odbywa się poprzez wyliczenie wartości wielomianu pierwszego stopnia o następującej postaci:

y =w1*x1+w2*x2+. . . +wn*xn,

gdzie x jest daną wejściową, w wagą przypisaną do tego parametru, a n ilością parametrów wejściowych. Kolejnym krokiem jest przepuszczenie tak uzyskanego wyniku y przez funkcję logistyczną (np. sigmoid lub tangens hiperboliczny), w celu uzyskania wartości z przedziału (0;1). Po uzyskaniu takiej wartości możemy zaklasyfikować daną wejściową do grupy A lub grupy B na podstawie prostej reguły: jeżeli y jest większe lub równe od 0.5 wtedy klasa A, w przeciwnym wypadku klasa B.

Proces uczenia maszynowego polega tu na takim doborze wag w, aby uzyskać na zbiorze uczącym jak największy odsetek poprawnych klasyfikacji. Uzyskuje się to poprzez wyliczenie funkcji błędu oraz jej minimalizację (czyli minimalizację błędu) z wykorzystaniem algorytmu gradientu prostego.

Najczęściej wykorzystywanym typem regresji logistycznej jest regresja binarna, kiedy klasyfikujemy daną wejściową do jednej z dwóch kategorii (stąd nazwa binarna) - i taki właśnie jest temat naszego projektu, czyli stworzenie klasyfikatora binarnego, który przyporządkowuje pacjenta do jednej z dwóch klas: 0, 1. 0 oznacza, że pacjent ma łagodną zmianę rakową (jest zdrowy), a 1 oznacza, że pacjent ma złośliwą zmianę rakową (jest chory).

2.2 Badania symulacyjne

Suma elementów zbioru treningowego i testującego wynosi 569, a liczba etykiet (cech) wynosi 30.

Po przeprowadzeniu analizy badawczej przy użyciu bibliotek pandas oraz seaborn, doszliśmy do wniosku, że dane jakie posiadamy są bardzo rozbieżne, dlatego też poddaliśmy je standaryzacji, aby dane należały mniej więcej do tej samej dziedziny wartości (do tego samego przedziału liczbowego).

Kolejnym krokiem, który wykonaliśmy było podzielenie danych na zbiór treningowy i testujący. Ostatecznie dane zostały podzielone w następujących proporcjach:

- Liczba elementów zbioru treningowego wynosi 512.

- Liczba elementów zbioru testującego wynosi 57.

W następnej kolejności wytrenowaliśmy model - przykład dla pacjenta numer 5:

- Klasa przewidziana dla pacjenta przez model: 0

- Prawdziwa klasa, do której należy pacjent: 0

Jak widać powyżej model przyporządkował pacjenta do prawidłowej klasy, a mianowicie klasy 0 – pacjent ma łagodną zmianę rakową (jest zdrowy).

Prawdopodobieństwa przynależności do danej klasy:

- Prawdopodobieństwo przynależności pacjenta do klasy 0: 0,999997364

- Prawdopodobieństwo przynależności pacjenta do klasy 1: 0,0000263585905

Następnie dokonaliśmy ewaluacji modeli, która prezentuje się następująco:

- Mean squared error: 0,04

Mean squared error to błąd średniokwadratowy. Jest to średnia różnica kwadratów odchyleń. Błąd średniokwadratowy niewiele nam mówi, natomiast ważne jest, aby był jak najmniejszy.

- Variance score: 0,86

Variance score to wartość wariancji. Wariancja to średnia arytmetyczna kwadratów odchyleń od ich średniej arytmetycznej. Im wartość wariancji jest większa od 0,6 to tym lepiej. W naszym przypadku wynosi ona 0,78, zatem model regresji logistycznej jest dobry.

- Cross – validation dla 4 iteracji: [0,97202797; 0,97887324; 0,97183099; 0,98591549]

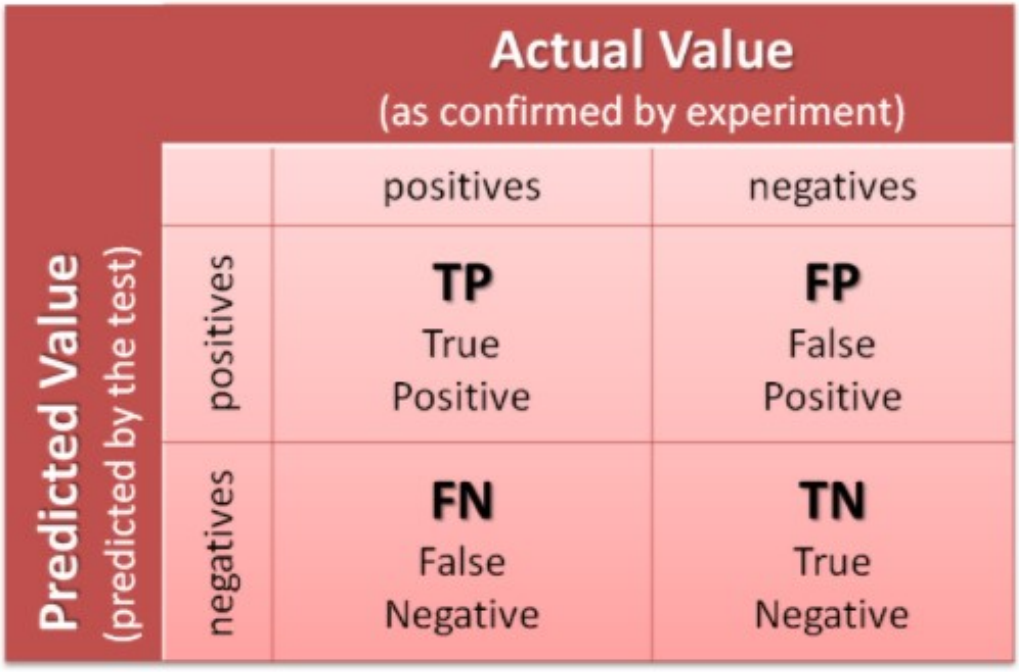
Cross validation to sprawdzian krzyżowy, inaczej kroswalidacja. Idea tej metody polega na podzieleniu zbioru uczącego na możliwie równe podzbiory, a następnie stworzeniu pseudopróby uczącej poprzez usunięcie ze zbioru wejściowego jednego z k-podzbiorów i potraktowanie go jako zbioru testowego. Powtarzając powyższy proces k razy i obliczając statystyki na zbiorach testowych tak zbudowanych modeli możliwa jest estymacja błędu generalizacji modelu. W naszym przypadku wykonywane są 4 iteracje (cv=4) i w każdej z nich z sumy elementów uczących i testujących 3/4 tej sumy są brane jak zbiór uczący, a 1/4 tej sumy jako zbiór testujący. Jak widać powyżej wszystkie cztery wartości są bardzo zblizone do siebie. Największa z nich wynosi: 0.98591549, natomiast najmniejsza: 0.97183099.

- Model accuracy: 0,96

Model accuracy to współczynnik dokładności. Nie wiele on mówi o wytrenowanym modelu, ale ważne jest, aby był jak najbliżej wartości 1. W tym przypadku współczynnik dokładności wynosi 0,96, a więc można powiedzieć, że jest to bardzo dobry wynik.

- Confusion matrix:

[[27 2]
[0 28]]



Confusion matrix to macierz konfuzji (macierz błedu/pomyłek). Macierz konfuzji jest podstawowym narzędziem stosowanym do oceny jakości klasyfikacji. W tym zadaniu brana pod uwagę była klasyfikacja binarna (dwie klasy) i następujące kodowanie klas:

1 - positive: pacjent posiada złośliwą zmianę rakową,

0 - negative: pacjent posiada łagodną zmianę rakową.

Macierz konfuzji powstaje z przecięcia klasy prognozowanej i klasy faktycznie zaobserwowanej - mamy zatem 4 przypadki (2 dla zgodności i 2 dla niezgodności prognozy ze stanem faktycznym). W tym przypadku:

a. 27 - true-positive (TP - prawdziwie pozytywna) - przewidywanie pozytywne, faktycznie zaobserwowana klasa pozytywna (pozytywny wynik testu na raka i faktycznie posiadana złośliwa zmiana rakowa),

b. 2 - false-positive (FP - fałszywie pozytywna) - przewidywanie pozytywne, faktycznie zaobserwowana klasa negatywna (pozytywny wynik testu na raka i faktyczny brak złośliwej zmiany rakowej),

c. 0 - false-negative (FN - fałszywie negatywna) - przewidywanie negatywne, faktycznie zaobserwowana klasa pozytywna (negatywny wynik testu na raka i faktyczne posiadanie złośliwej zmiany rakowej),

d. 28 - true-negative (TN - prawdziwie negatywna) - przewidywanie negatywne, faktycznie zaobserwowana klasa negatywna (negatywny wynik testu na raka i faktycznie brak złośliwej zmiany rakowej).

Czyli w sumie model pomylił się w 2 przypadkach na 57 przypadków (ze zbioru testującego).

Rozdział 3

Podsumowanie

W projekcie tym skupiliśmy się na nadzorowanym uczeniu maszynowym i stworzyliśmy klasyfikator wspomagający diagnozę raka. Do jego stworzenia posłużyliśmy się bazą danych z repozytorium o nazwie Breast Cancer Wisconsin. Zawierała ona informacje o 569 pacjentach kliniki i dla każdego z nich 30 etykiet (cech), takich jak np. promień czy obwód zmiany rakowej. Dane te poddaliśmy procesowi standaryzacji za względu na ich dużą rozbieżność, dokonaliśmy ich podziału na zbiór treningowy i testujący, a następnie wytrenowaliśmy model. Ponadto przeprowadziliśmy ewaluację naszego modelu, która wykazała, że nasz model jest bardzo dobry - pomylił się tylko w 2 na 57 przypadków (ze zbioru testującego). Zrealizowanie tego projektu pozwoliło nam bardziej zagłębić się w dziedzinę uczenia maszynowego, a tym samym pozwoliło nam zdobyć nową wiedzę i umiejętności.

Dodatek A

Kod programu

```
import numpy as np
from sklearn import datasets
patients = datasets.load_breast_cancer()


print(patients.DESCR)


print(patients.data.shape)
print(patients.target.shape)


print("First patient in database")
print(patients['data'][0,:])


print(patients['target'][0])


import pandas as pd
patients_data_p = pd.DataFrame(patients.data, columns=[patients.feature_names])
patients_data_p.head()


patients_data_p.describe()


%matplotlib inline

import matplotlib.pyplot as plt
import seaborn as sb
```

```
#sb.pairplot(patients_data_p, diag_kind="kde")
sb_plot = sb.pairplot(patients_data_p, diag_kind="kde")
sb_plot.savefig("wykresy1.png")


from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(patients['data'])


import pandas as pd
patients_data_p_scaled =
    pd.DataFrame(scaled_data, columns=[patients.feature_names])
patients_data_p_scaled.head()


patients_data_p_scaled.describe()


%matplotlib inline

import matplotlib.pyplot as plt
import seaborn as sb

#sb.pairplot(patients_data_p, diag_kind="kde")
sb_plot_scaled = sb.pairplot(patients_data_p_scaled, diag_kind="kde")
sb_plot_scaled.savefig("wykresy2.png")


from sklearn.model_selection import train_test_split

patients_train_data, patients_test_data, \
patients_train_target, patients_test_target = \
train_test_split(scaled_data, patients['target'], test_size=0.1)


print("Training dataset:")
print("patients_train_data:", patients_train_data.shape)
print("patients_train_target:", patients_train_target.shape)


print("Testing dataset:")
print("patients_test_data:", patients_test_data.shape)
print("patients_test_target:", patients_test_target.shape)
```

```
from sklearn.linear_model import LogisticRegression

logistic_regression = LogisticRegression()
logistic_regression.fit(patients_train_data, patients_train_target)


from sklearn.metrics import mean_squared_error
print("Mean squared error of a learned model: %.2f" %
      mean_squared_error(patients_test_target,
      logistic_regression.predict(patients_test_data)))


from sklearn.metrics import r2_score
print('Variance score: %.2f' % r2_score(patients_test_target,
    logistic_regression.predict(patients_test_data)))


from sklearn.model_selection import cross_val_score
scores = cross_val_score(LogisticRegression(),
    scaled_data, patients['target'], cv=4)
print(scores)


id=5
prediction = logistic_regression.predict(patients_test_data[id,:].reshape(1,-1))
print("Model predicted for patient {0} value {1}".format(id, prediction))

print("Real value for patient \"{0}\" is
    {1}".format(id, patients_test_target[id]))


prediction_probability =
logistic_regression.predict_proba(patients_test_data[id,:].reshape(1,-1))
print(prediction_probability)


from sklearn.metrics import accuracy_score
acc = accuracy_score(patients_test_target,
        logistic_regression.predict(patients_test_data))
print("Model accuracy is {0:0.2f}".format(acc))


from sklearn.metrics import confusion_matrix

conf_matrix = confusion_matrix(patients_test_target,
```

```
logistic_regression.predict(patients_test_data))
print(conf_matrix)
```