Akademia Ekonomiczno-Humanistyczna w Warszawie

SPRAWOZDANIE

INTELIGENTNA ANALIZA DANYCH

LAB1
KWANTYFIKATORY

23.10.2021

JOANNA PRAJZENDANC
36358

MIŁOSZ SAKOWSKI
36381

1. Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z nowymi pojęciami używanymi podczas analizy danych: klasyfikatory, metryki, macierz pomyłek (tablica pomyłek, ang. *confusion matrix*), czułość (ang. *sensitivity*, *recall*), swoistość (ang. *specificity*, *SPC*), precyzja (ang. *precision*) i dokładność (ang. *accuracy*, *ACC*) oraz F1-score.

2. Analiza kodu i wyjaśnienie pojęć jako zadanie 1

```
# 1-chory, 0-zdrowy
y_real = [0, 1, 1, 0, 1, 1, 1, 0, 0, 0] # rzeczywistosć
y_pred = [1, 1, 1, 1, 0, 0, 1, 1, 0, 1] # wyniki testu
```

```
TP = 0 # true positive, jest: chory, wynik: chory, y_real=1, y_pred=1
FP = 0 # false positive, jest: zdrowy, wynik: chory, y_real=0, y_pred=1
TN = 0 # true negative, jest: zdrowy, wynik: zdrowy, y_real=0, y_pred=0
FN = 0 # false nagative, jest: chory, wynik: zdrowy, y_real=1, y_pred=0
for i in range(len(y_pred)):
    if y_real[i]==1 and y_pred[i]==1:
        TP += 1
    if y_real[i]==0 and y_pred[i]==1:
        FP += 1
    if y_real[i]==0 and y_pred[i]==0:
        TN += 1
    if y_real[i]==1 and y_pred[i]==0:
        FN += 1
print("TP: " + str(TP))
print("FP: " + str(FP))
print("TN: " + str(TN))
print("FN: " + str(FN))
```

```
TP: 3
FP: 4
TN: 1
FN: 2
```

Obraz 1: Fragment kodu z wynikami TP, FP, TN, FN

Powyższy zrzut kodu przedstawia zaproponowane 10 obserwacji oraz 10 predykcji na podstawie których możemy stwierdzić wyniki TP, FP, TN, FN i utworzyć macierz pomyłek.

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_real, y_pred, labels=[1,0])
```

```
array([[3, 2],
       [4, 1]], dtype=int64)
```

Obraz 2: Fragment kodu z macierzą pomyłek

Przedstawiony zrzut ekranu przedstawia **macierz pomyłek**, która pozwala dokonać oceny jakości klasyfikacji binarnej. W takiej tablicy dane są oznaczone etykietami: „pozytywną”

(TP - *true positive*, TN - *true negative*) oraz „negatywną” (FP - *false positive*, FN - *false negative*).

		Klasa rzeczywista	
		pozytywna	**negatywna**
Klasa predykowana	**pozytywna**	prawdziwie pozytywna (TP)	fałszywie pozytywna (FP)
	negatywna	fałszywie negatywna (FN)	prawdziwie negatywna (TN)

Obraz 3: Schemat tworzenia macierzy pomyłek. Źródło: Wikipedia

Obliczając sumę elementów tego samego typu można zauważyć, że wartości „negatywne” (FP + FN) mają wyższą sumę

$$2 + 4 = 6$$

niż wartości „pozytywne” (TP + TN)

$$3 + 1 = 4$$

co nie jest do końca dobrym wynikiem, ponieważ powinno dążyć się do jak najwyższej wartości TP + TN i do stosunkowo jak najniższej wartości FP + FN

Licząc teraz **czułość** (odsetek prawdziwie pozytywnych):

$$\frac{TP}{TP + FN} = \frac{3}{3 + 2} = 0.6$$

```
recall = TP / (TP + FN)
print(recall)
```

```
0.6
```

Obraz 4: Fragment kodu z wynikiem czułości

Czułość określa, jaki jest udział prawidłowo prognozowanych przypadków pozytywnych (TP) ze wszystkich przypadków pozytywnych. Powinna dążyć do 1.

Oraz **swoistość** (odsetek prawdziwie negatywnych):

$$\frac{TN}{TN + FP} = \frac{1}{1 + 4} = 0.2$$

Swoistość bada, jaki jest udział prawidłowo przewidzianych przypadków negatywnych (TN) ze wszystkich przypadków negatywnych. Powinna dążyć do 1.

Precyzję:

$$\frac{TP}{TP + FP} = \frac{3}{3 + 4} \approx 0.43$$

```
precision = TP / (TP + FP)
print(precision)
```

```
0.42857142857142855
```

Obraz 5: Fragment kodu z wynikiem precyzji

Precyzja wskazuje, że wartość oznaczona jako dodatnia jest rzeczywiście dodatnia. Im wyższa precyzja tym mniejsza ilość FP.

Dokładność:

$$\frac{TP + TN}{TP + FN + TN + FP} = \frac{3 + 1}{7 + 3} = 0.4$$

```
accuracy = (TP + TN) / (TP + FP + TN + FN)
print(accuracy)
```

```
0.4
```

Obraz 6: Fragment kodu z wynikiem obliczeń dokładności

Widzimy, że predykcja jest niewystarczająca, ma zbyt małą dokładność. Dokładność to metryka, który pozwala ocenić jakość klasyfikacji testu i powinna dążyć do 1. Z obliczeń możemy dowiedzieć się jaka część testów ze wszystkich zaklasyfikowanych, została zaklasyfikowana poprawnie.

Możemy również policzyć średnią skuteczność modelu w tym zbiorze danych, tzw. **F1-score**:

$$\frac{2}{\frac{1}{precyzja} * \frac{1}{czułość}} = \frac{2 * precyzja * czułość}{precyzja + czułość} = \frac{2 * 0.43 * 0.6}{0.43 + 0.6} \approx 0.5$$

```
f1 = (2 * precision * recall) / (precision + recall)
print(f1)
```

```
0.5
```

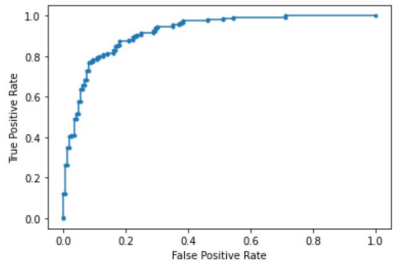
Obraz 7: Fragment kodu z wynikiem wskaźnika F1-score

Im bliżej wartości 1, tym lepiej to świadczy o algorytmie klasyfikującym. Wskaźnik F1-score przyjmuje wartość 1, kiedy mamy do czynienia z idealną czułością i precyzją.

Na koniec zaprezentowany został przykład sztucznej inteligencji, która na podstawie modelu rozkładu normalnego uzyskała dokładność 0.919.

In [19]:
```
# 0.1 0.2 0.2 0.8
# 0-1
# thr=0.15
# 0 1 1 1
# thr=0.5
# 0 0 0 1

from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot

x, y = make_classification(n_samples=1000, n_classes=2, random_state=1)
train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.3, random_state=2)

model = LogisticRegression(solver='lbfgs')
model.fit(train_x, train_y)

probs = model.predict_proba(test_x)
probs = probs[:, 1]
auc = roc_auc_score(test_y, probs)
print('AUC=%.3f' % (auc))

# ROC curve
fpr, tpr, thresholds = roc_curve(test_y, probs)
# fpr = fp / (fp + tn)
# tpr = tp / (tp + fn)
pyplot.plot(fpr, tpr, marker='.')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.show()
```

```
AUC=0.919
```



Obraz 8: Fragment kodu z przykładem sztucznej inteligencji rozwiązującej ten sam problem

3. Zadanie nr 2

Zadanie:

Jesteśmy programistami sterowników do samochodów z systemem automatycznego sterowania. Nasz system ma na celu jak najbezpieczniejszą jazdę. Po otrzymaniu sygnału z kamery musimy sklasyfikować, czy obraz przedstawia obiekt który należy ominąć (np. pies) czy inny, niegroźny obiekt (np. cień). Przyjmijmy, że obiekt do ominięcia reprezentuje klasę pozytywną. Proszę opisać w 5 zdaniach czym powinien charakteryzować się nasz system (np. wysokie/niskie TP,FP,TN,FN) i dlaczego?

Odpowiedź:
Parametr true-negative (TN) w takim systemie powinien być najważniejszy- najlepiej gdyby przyjmował wartość 1. Parametr true-positive(TP) również powinien dążyć do jak najwyższej wartości (1). Parametr false-negative(FN), który w naszym przypadku określałby wypadek powinien być jak najniższy- najlepiej 0. Parametr false-positive(FP) również powinien być jak najniższy aczkolwiek gdyby się pojawił, pojazd zatrzymałby się w sytuacji braku przeszkody

4. Zadanie nr 3

Zadanie:

Mamy dwie macierze pomyłek:

a) 9 1

0 4

b) 4 6

3 1

Która macierz pomyłek reprezentuje lepszy model i dlaczego?

Odpowiedź:

Macierz a) reprezentuje lepszy model, ponieważ suma TP i TN jest większa od sumy FP + FN, czyli suma jest wyższa dla wartości sklasyfikowanych jako przypadki pozytywne niż dla przypadków sklasyfikowanych jako negatywne.