VJEŽBA 5: RJEŠAVANJE KLASIFIKACIJSKIH PROBLEMA. VREDNOVANJE KLASIFIKACIJSKIH MODELA.

I. Cilj vježbe:

Upoznati se s klasifikacijskim problemima te primijeniti logističku regresiju, metodu K najbližih susjeda te stabla odlučivanja za njihovo rješavanje. Upoznati se s načinom vrjednovanja klasifikacijskih modela.

II. Opis vježbe:

II.1. Nadgledano učenje. Problem klasifikacije.

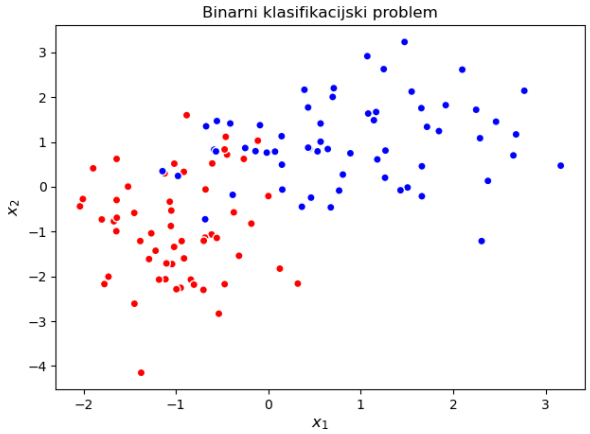
U ovoj vježbi razmatra se problem nadgledanog učenja gdje je cilj odrediti nepoznatu funkcionalnu ovisnost između *m* ulaznih veličina $X = [x_1, x_2, \dots, x_m]$ i izlazne veličine *y* na temelju podatkovnih primjera. Promatra se slučaj kada je izlazna veličina *y* oznaka klase ili kategorije (klasifikacija). Pri tome su podatkovni primjeri parovi koji se sastoje od vektora ulaznih veličina (atributi ili značajke) i vrijednosti izlazne veličine (ciljne varijable), stoga se *i*–ti podatkovni primjer ili uzorak može prikazati kao uređeni par $(\mathbf{x}^{(i)}, y^{(i)})$. Vektor ulaznih veličina zapisuje u obliku:

$$\mathbf{x}^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \dots, x_m^{(i)}\right]^T \tag{5-1}$$

Skup koji se sastoji od *n* raspoloživih podatkovnih primjera $\left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), \dots, (\mathbf{x}^{(n)}, y^{(n)})$ može se zapisati u matričnom obliku:

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_m^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_m^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \dots & x_m^{(n)} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}. \tag{5-2}$$

Ako izlazna veličina ima dvije moguće vrijednosti (npr. klase „Spam“ i „Not spam“), tada problem nazivamo binarni klasifikacijski problem. U suprotnom radi se o višeklasnoj klasifikaciji (npr. klase „Low“, „Medium“, i „High“). Kod binarne klasifikacije koristimo 0/1 kodiranje izlaznih vrijednosti, a u slučaju višeklasne klasifikacije 1-od-*K* kodiranje (engl. *one-hot encoding*). Primjer binarnog klasifikacijskog problema s dvije ulazne veličine dan je na slici 5.1.



Sl. 5.1. Primjer binarnog klasifikacijskog problema.

II.2. Algoritmi za klasifikaciju

II.2.1. Logistička regresija

Logistička regresija (engl. *logistic regression*) je osnovni algoritam strojnog učenja koji se koristi za klasifikaciju. Iako se često naziva "regresijom", koristi se za predviđanje kategoričkih varijabli, što ga čini klasifikacijskim algoritmom, a ne regresijskim. Koristi se za modeliranje vjerojatnosti da će neki ulazni primjer pripadati klasi 1 u slučaju binarne klasifikacije, koristeći logističku funkciju kako bi se izlaz ograničio na raspon [0, 1]. U scikit-learn biblioteci se koristi na sličan način kao i model linearne regresije (na raspolaganju su metode *.fit* i *.predict*). Izgradnja modela logističke regresije pomoću scikit-learn biblioteke dana je u primjeru 5.1. Provjerite parametre klase sklearn.linear_model.LogisticRegression.

Primjer 5.1.

```
import numpy as np
from sklearn.linear_model import LogisticRegression

# Umjetni podaci
np.random.seed(42)
X = np.random.rand(100, 2)  # 100 podatkovnih primjera, 2 atributa
y = np.random.randint(0, 2, 100)  # Oznake klase (0 ili 1)

# Kreiraj i istreniraj model logističke regresije
logreg = LogisticRegression()
logreg.fit(X, y)

# Predikcija modela
y_pred = logreg.predict(X)
```

II.2.2. Metoda *K* najbližih susjeda

Osnovna pretpostavka metode *K*-najbližih susjeda (engl. *K-nearest neighbours* – KNN) je da slični primjeri imaju slične izlazne vrijednosti. Za svaki neoznačeni primjer tj. novi primjer kojeg je potrebno klasificirati, KNN traži *K* najbližih označenih primjera u skupu podataka. Kao mjera udaljenosti između primjera najčešće se koristi Euklidska udaljenost. Kada je pronađeno *K* najbližih susjeda, određuje se klasa novog primjera na temelju klasa susjeda. Na primjer, ako 4 od 5 susjeda pripadaju klasi 1, tada je vjerojatnost pripadanja klasi 1 jednaka 4/5. Izgradnja algoritma KNN pomoću scikit-learn biblioteke dana je u primjeru 5.2. Provjerite parametre klase sklearn.neighbors.KNeighborsClassifier.

Primjer 5.2.

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

# Umjetni podaci
np.random.seed(42)
X = np.random.rand(100, 2)  # 100 podatkovnih primjera, 2 atributa
y = np.random.randint(0, 2, 100)  # Oznake klase (0 ili 1)

# Kreiraj i istreniraj KNN
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X, y)

# Predikcija modela
y_pred = knn.predict(X)
```

II.2.3. Stabla odlučivanja

Stablo odlučivanja (engl. *decision tree*) gradi hijerarhijsku strukturu stabla na temelju ulaznih veličina odnosno značajk. Svaki unutarnji čvor stabla postavlja pitanje o određenoj značajki (if else uvjet), a grane iz čvora odgovaraju mogućim odgovorima na to pitanje (True ili False). Stablo završava u listovima koji predstavljaju klase ili kategorije. Podatkovni primjer prolazi kroz stablo počevši od korijenskog čvora, prolazeći kroz čvorove na temelju odgovora na pitanja, sve dok

ne dođe do lista, koji označava klasu kojoj primjer pripada. Izgradnja stabla odlučivanja i njegov prikaz pomoću scikit-learn biblioteke dana je u primjeru 5.3. Provjerite parametre klase sklearn.tree.DecisionTreeClassifier.

Primjer 5.3.

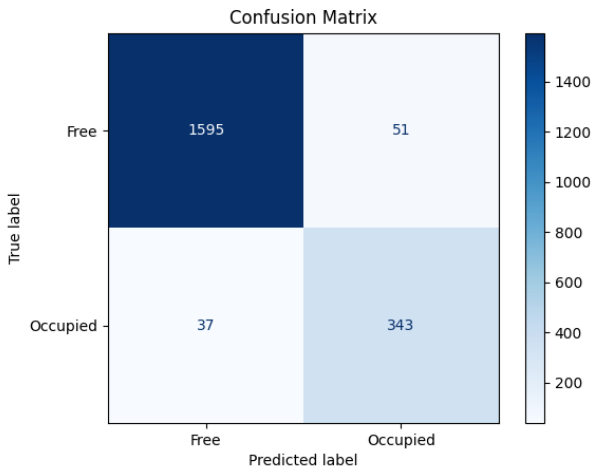
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, plot_tree

# Umjetni podaci
np.random.seed(42)
X = np.random.rand(100, 2)  # 100 podatkovnih primjera, 2 atributa
y = np.random.randint(0, 2, 100)  # Oznake klase (0 ili 1)

# Kreiraj i istreniraj stablo odlucivanja
dt = DecisionTreeClassifier()
dt.fit(X, y)

# Predikcija modela
y_pred = dt.predict(X)

# Vizualizacija stabla odlucivanja
plt.figure(figsize=(12, 8))
plot_tree(dt, filled=True)
plt.show()
```

II.4. Vrjednovanje modela

Testiranje predikcijskih sposobnosti odnosno sposobnosti generalizacije izgrađenog klasifikatora potrebno je provesti na zasebnom skupu podataka koji se naziva skup za testiranje. U slučaju binarne klasifikacije podatkovni primjer $(\mathbf{x}^{(i)}, y^{(i)})$ može biti pozitivan $y^{(i)} = 1$ ili negativan $y^{(i)} = 0$. Stoga, moguća su četiri slučaja prilikom uspoređivanja rezultata koje daje klasifikator sa stvarnom vrijednosti:

- istinito pozitivan rezultat (engl. *true positive* – TP) – pozitivni primjer kojeg je klasifikator klasificirao kao pozitivan primjer,
- istinito negativan rezultat (eng. *true negative* – TN) – negativni primjer kojeg je klasifikator klasificirao kao negativan primjer,
- lažno pozitivan rezultat (eng. *false positive* – FP) – negativan primjer kojeg je klasifikator klasificirao kao pozitivan primjer, i
- lažno negativan rezultat (eng. *false negative* – FN) – pozitivan primjer kojeg je klasifikator klasificirao kao negativan primjer.

Očito da su TP i TN primjeri koje klasifikator točno klasificira dok u su FP i FN primjeri koje klasifikator pogrešno klasificira. Vrlo je korisno prikazati ove rezultate u obliku matrice zabune (engl. *confusion matrix*). Ova matrica pokazuje ukupan broj rezultata za sva četiri slučaja. Primjer matrice zabune dan je na slici 5.2.



Sl. 5.2. Primjer matrice zabune.

Za vrjednovanje klasifikacijskog modela se osim matrice zabune mogu koristiti sljedeće metrike:

- točnost (engl. *accuracy*) - predstavlja udio točno klasificiranih primjera:

$$točnost = \frac{TP+TN}{TP+TN+FP+FN} \tag{5-3}$$

- preciznost (engl. *precision*) - predstavlja udio točno klasificiranih primjera u skupu koje klasifikator klasificira kao pozitivne primjere:

$$preciznost = \frac{TP}{TP+FP} \tag{5-4}$$

- odziv (engl. *recall*) - predstavlja udio točno klasificiranih primjera u skupu pozitivnih primjera:

$$odziv = \frac{TP}{TP+FN} \tag{5-5}$$

Iznosi ovih metrika su u intervalu od 0 do 1 pri čemu je 1 najbolja vrijednost. Način korištenja metrika u okviru scikit-learn biblioteke dan je u primjeru 5.4.

Primjer 5.4.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
precision_score, recall_score, accuracy_score, classification_report

# Stvarne vrijednosti i predikcije
y_true = np.array([0, 1, 0, 1, 1, 0, 1, 0, 0, 1])
y_pred = np.array([0, 1, 0, 1, 0, 1, 1, 0, 1, 1])

# Izracunaj matricu zabune i prikazi ju
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Class 0',
'Class 1'])
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()

# Izracunaj preciznost
precision = precision_score(y_true, y_pred)

# Izracunaj odziv
recall = recall_score(y_true, y_pred)

# Izracunaj tocnost
accuracy = accuracy_score(y_true, y_pred)

# Report
print(classification_report(y_true, y_pred))
```

III. Priprema za vježbu:

Nema posebne pripreme za vježbu.

IV. Rad na vježbi:

1. Isprobajte Python primjere iz II. Opis vježbe u Visual Studio Code IDE. Razmislite o svakoj liniji programskog koda i što je njen rezultat. Pokrenite primjere u *Debug* modu i pogledajte u *Explorer*-u kako izgleda svaka od varijabli u danim primjerima.

2. Riješite dane zadatke.

Zadatak 1

Skripta 5.1. učitava skup podataka koji se nalazi u csv datoteci `occupancy_processed.csv`. Ova datoteka sadrži podatke koji su prikupljeni u prostoriji veličine 6m x 4.6m tijekom 4 dana [1]. Zbog jednostavnosti skup sadrži samo dva atributa: mjerenja dobivena sa senzora temperature i mjerenja sa senzora CO2. Izlazna (ciljna) veličina je zauzetost prostorije (0 – prazna prostorija, 1 – u prostoriji se nalazi barem jedna osoba). Cilj je izgraditi klasifikator koji će na temelju trenutnih mjerenja dobivenih sa senzora temperature i sa senzora CO2 procijeniti zauzetost prostorije.

a) Pokrenite skriptu i pogledajte dobiveni dijagram raspršenja. Što primjećujete?
b) Koliko podatkovnih primjera sadrži učitani skup podataka?
c) Kakva je razdioba podatkovnih primjera po klasama?

Zadatak 2

Izgradite i evaluirajte algoritam *K* najbližih susjeda. Slijedite ovaj redoslijed:

a) Podijelite podatke na skup za učenje i skup za testiranje (omjer 80%-20%) pomoću funkcije `train_test_split`. Koristite opciju `stratify=y`.
b) Pomoću `StandardScaler` skalirajte ulazne veličine.
c) Pomoću klase `KNeighborsClassifier` izgradite algoritam *K* najbližih susjeda.
d) Evaluirajte izgrađeni klasifikator na testnom skupu podataka:
    a. prikažite matricu zabune
    b. izračunajte točnost klasifikacije
    c. izračunajte preciznost i odziv po klasama
e) Što se događa s rezultatima ako se koristi veći odnosno manji broj susjeda?
f) Što se događa s rezultatima ako ne koristite skaliranje ulaznih veličina?

Zadatak 3

Umjesto algoritma *K* najbližih susjeda koristite stablo odlučivanja te ponovite korake a) do d) iz prethodnog zadatka.
a) Vizualizirajte dobiveno stablo odlučivanja.
b) Što se događa s rezultatima ako mijenjate parametar `max-depth` stabla odlučivanja?
c) Što se događa s rezultatima ako ne koristite skaliranje ulaznih veličina?

Zadatak 4

Po uzoru na prethodne zadatke izgradite model logističke regresije. Što primjećujete kod vrednovanja ovog modela? Što je uzrok dobivenim rezultatima?

V. Izvještaj s vježbe

Kao izvještaj s vježbe prihvaća se web link na repozitorij pod nazivom `PSU_LV`.

V. Literatura

[1] https://archive.ics.uci.edu/dataset/864/room+occupancy+estimation