

# 6. Klasifikacija primjenom algoritma K najbližih susjeda

## 6.1 Cilj Vježbe

Upoznati se s algoritmom  $K$  najbližih susjeda. Upoznati se sa strojevima s potpornim vektorima. Upoznati se s načinom određivanja hiperparametara navedenih metoda te odgovarajućom podrškom u scikit-learn biblioteci.

## 6.2 Teorijska pozadina

U ovoj vježbi studenti se upoznaju s metodama koje omogućavaju modeliranje složenih odnosa između ulazne veličine i izlazne veličine: algoritam  $K$  najbližih susjeda (engl. *K Nearest Neighbours* – KNN) i strojevi s potpornim vektorima (engl. *support vector machine*). Analizira se binarni klasifikacijski problem u kojem podatkovni primjeri nisu linearni odvojivi te se daje uvid kako navedene metode generiraju nelinearnu granicu odluke u ulaznom prostoru. Razmatra se problem poduskađivanja (engl. *underfitting*) i pretjeranog uskađivanja na podatke (engl. *overfitting*) te kako se ovaj problem rješava podešavanjem hiperparametara modela  $k$ -struktom unakrsnom validacijom (engl. *k-fold cross validation*).

### 6.2.1 Algoritam $K$ najbližih susjeda (KNN)

Algoritam  $K$  najbližih susjeda (KNN) je relativno jednostavan algoritam strojnog učenja koji se može primijeniti i na regresijske i na klasifikacijske probleme. KNN pripada grupi neparametarskih metoda. U takvim modelima ne modelira se izravno aproksimirajuća funkcija između ulaznih veličina i izlazne veličine na temelju skupa za učenje kao što je to na primjer u slučaju logističke regresije gdje je aproksimirajuća funkcija parametrirana konačnim vektorom parametara  $\theta$ . U KNN algoritmu, predikcija se za neki novi podatkovni primjer  $x$  određuje izravno na temelju skupa za učenje. Često se naziva i lijeni algoritam (engl. *lazy learner*) jer pamti sve podatke za učenje. U KNN algoritmu, primjer se klasificira na temelju klase njemu najbližih  $K$  susjeda. Pri tome se kao mjeru udaljenosti mogu koristiti različite metrike (euklidska udaljenost, Manhattan udaljenost i sl.). U slučaju numeričkih veličina najčešće se koristi Euklidska udaljenost  $d$  između dva podatkovna

## Poglavlje 6. Klasifikacija primjenom algoritma K najbližih susjeda i primjenom strojeva s potpornim vektorima.

primjera:

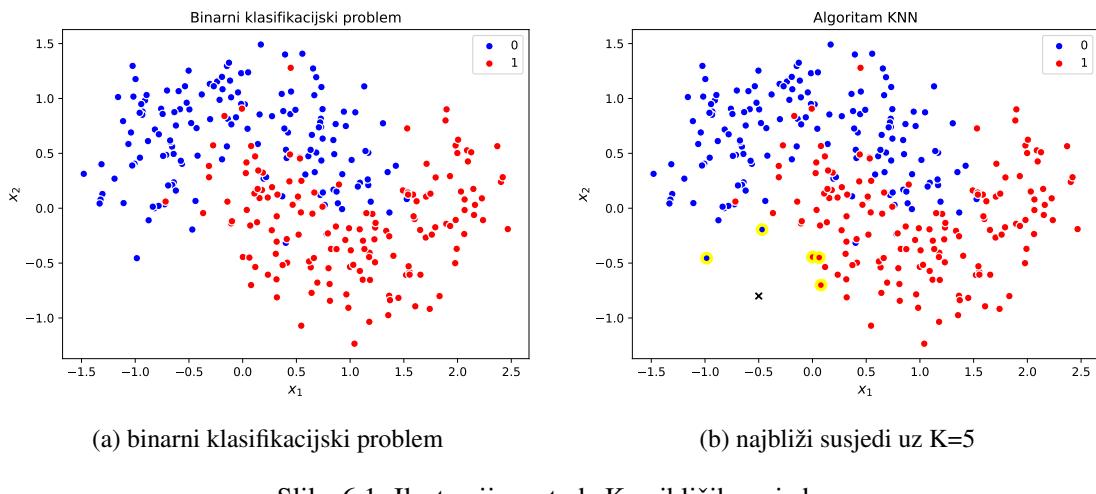
$$d(x, \tilde{x}) = \sqrt{\sum_{j=1}^m (x_j - \tilde{x}_j)^2}, \quad (6.1)$$

pri čemu je  $m$  broj ulaznih veličina. Vjerovatnost pripadanja klasi  $j$  algoritmom KNN definira se na način:

$$p(y = j | x) = \frac{1}{K} \sum_{k \in S} I(y^{(k)} = j), \quad (6.2)$$

gdje skup  $S$  sadrži  $K$  najbližih susjeda primjeru  $x$ , a  $I(a = b)$  je funkcija koja je jednaka 1 ako je  $a = b$ , a u suprotnom je jednaka 0. Pojednostavljeno, novi uzorak se klasificira s obzirom na većinsku klasu  $K$  najbližih susjeda. Broj susjeda  $K$  utječe na rezultate klasifikacije i naziva se hiperparametrom modela jer se mora unaprijed definirati.

Na slici 6.1 prikazan je binarni klasifikacijski problem s dvije ulazne veličine pri čemu su podaci za učenje obojani u ovisnosti o klasi (crveno klasa „1“, plava klasa „0“). Novi podatkovni primjer je prikazan crnom bojom te se metodom najbližih susjeda (uz  $K = 5$ ) klasificira kao klasa „1“ uz  $p(y = 1 | x) = 3/5$ .

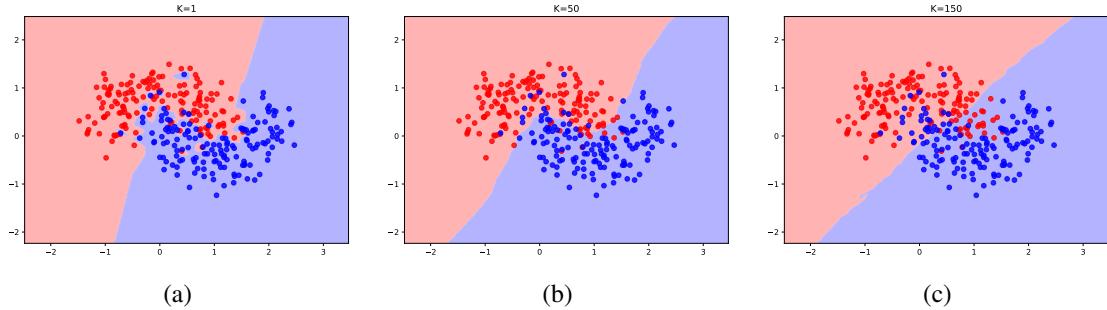


Slika 6.1: Ilustracija metode K najbližih susjeda.

$K$  najbližih susjeda stvara nelinearnu granicu odluke u ulaznom prostoru i može se koristiti za složene probleme klasifikacije. Na slici 6.2 prikazan je binaran klasifikacijski problem s dvije ulazne veličine pri čemu boja pozadine prikazuje izlaz KNN modela uz različit broj  $K$ . Očito da broj  $K$  utječe na složenost dobivene granice odluke. Vrlo je važno skalirati podatke prije rada s KNN algoritmom. Mane ovog algoritma su memorijska i računalna zahtjevnost osobito prilikom rada sa skupovima koji imaju velik broj podatkovnih primjera.

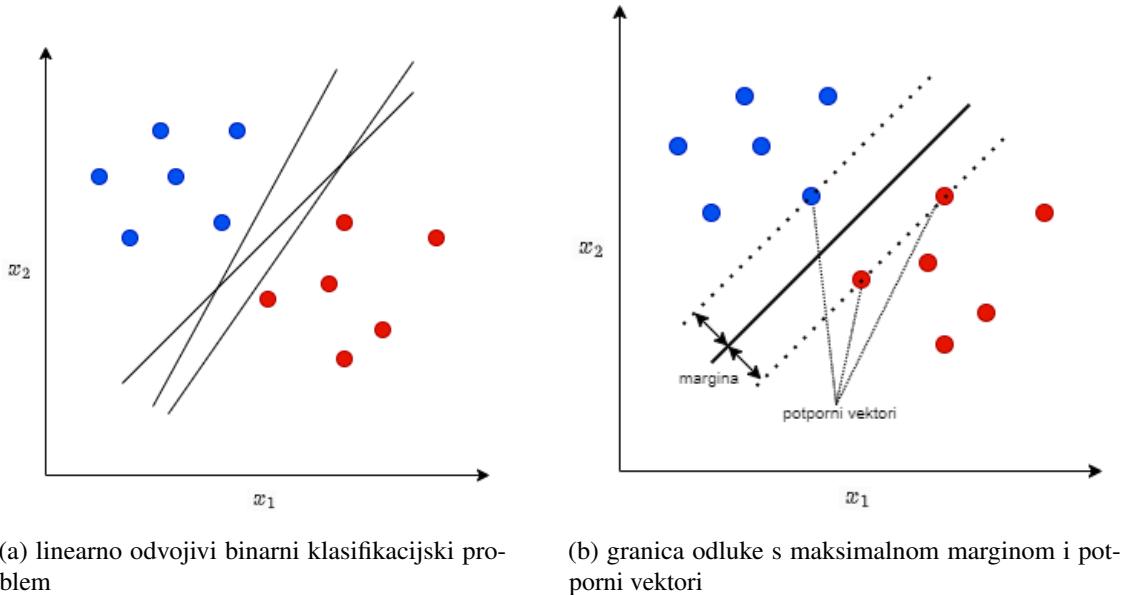
### 6.2.2 Strojevi s potpornim vektorima (SVM)

Strojevi s potpornim vektorima (engl. *Support Vector Machines* - SVM) su popularni algoritam koji se može primijeniti i klasifikacijske i na regresijske probleme. Algoritam se temelji na maksimizaciji



Slika 6.2: Primjena metode K najbližih susjeda na binarni klasifikacijski problem uz različit broj K.

**margine** koja predstavlja udaljenost granice odluke od podataka za učenje koji su najbliži granici odluke jer taj pristup daje najbolju generalizaciju. Potonji podatkovni primjeri iz skupa za učenje se nazivaju **potporni vektori**. Princip algoritma je ilustriran na jednostavnom **linearno odvojivom** binarnom klasifikacijskom problemu na slici 6.3a gdje su prikazane tri proizvoljne granice odluke. Kod SVM, od svih mogućih granica odluke odabire se ona koja maksimizira marginu, tj. granica odluke je najviše udaljena od primjera iz dviju klasa kao što prikazuje slika 6.3b.



(a) linearno odvojivi binarni klasifikacijski problem

(b) granica odluke s maksimalnom marginom i potporni vektori

Slika 6.3: Linearni SVM.

SVM je efikasan u slučajevima kada je ulazni prostor visokodimenzionalan. Za nove podatkovne primjere se prilikom predikcije koriste samo potporni vektori pa je algoritam značajno memorijski efikasniji nego KNN algoritam. Matematička formulacija optimizacijskog problema izlazi van okvira ovog priručnika, a može se pronaći u [1].

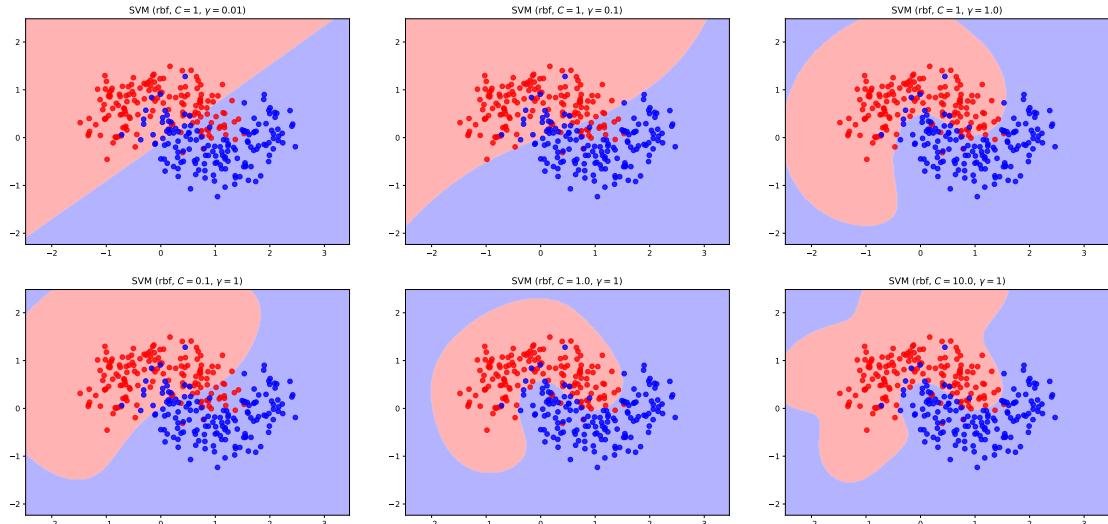
U stvarnosti podatkovni primjeri često nisu linearne odvojivi npr. zbog prisutnosti šuma ili pogrešaka. Kako bi se i u tom slučaju mogla odrediti granica odluke (jer će postojati pogreške klasifikacije), u SVM se uvodi hiperparametar C koji kontrolira utjecaj pogrešne klasifikacije podataka za učenje

na konačno rješenje. Veće vrijednosti parametra C znači jače penaliziranje pogrešne klasifikacije dok manje vrijednosti hiperparametra C znači manje penaliziranje pogrešne klasifikacije. Praktički to znači da C definira kompromis između širine margine i količine pogrešno klasificiranih primjera. Ovo se često naziva regularizacija i jedan je od načina sprječavanja pretjeranog usklađivanja na podatke.

Nadalje, kako bi se mogli rješavati problemi klasifikacije u kojima je prisutna nelinearna granica odluke, nelinearni SVM projicira podatkovne primjere u visokodimenzionalni prostor gdje se onda rješava problem pomoću linearog SVM-a. U praksi se ovo svodi na korištenje kernel funkcija od kojih je najpopularnija radikalno bazna funkcija (engl. *radial basis function* - RBF):

$$\kappa(x, \tilde{x}) = \exp\left(-\frac{\|x - \tilde{x}\|^2}{2\sigma^2}\right) = \exp(-\gamma \|x - \tilde{x}\|^2) \quad (6.3)$$

pri čemu je  $\gamma$  hiperparametar koji je veći od nule. Kernel funkcija praktički daje mjeru sličnosti između dva podatkovna primjera. Na slici 6.4 prikazana je granica odluke u slučaju rješavanja binarnog klasifikacijskog problema pomoću nelinearnog SVM uz RBF kernel i uz različite vrijednosti hiperparametara  $\gamma$  i  $C$ .



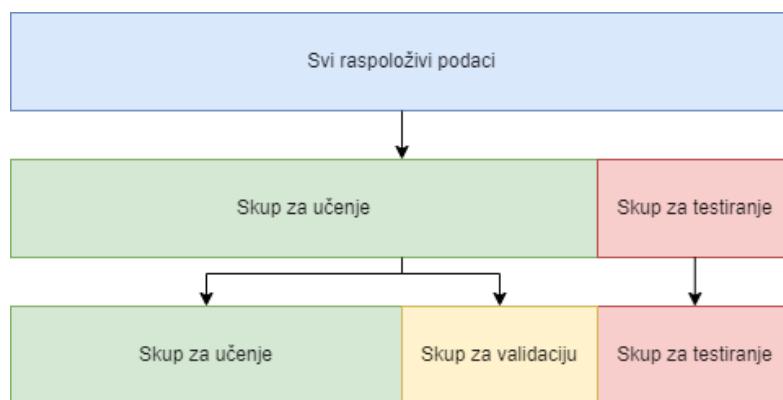
Slika 6.4: Primjena SVM na na binarni klasifikacijski problem uz različite vrijednosti  $\gamma$  i  $C$ .

### 6.2.3 Odabir optimalne dimenzije modela

Nerijetko izgrađeni model može imati dobre performanse na podacima za učenje, ali loše performanse na podacima za testiranje. Ovaj problem naziva se **pretjerano usklađivanje modela na podatke za učenje** (engl. *overfitting*), a posljedica je pretjerane složenosti modela za dane podatke. S druge pak strane, model može biti prejednostavan i ne može opisati odnose koji su prisutni u podacima pa će imati loše performanse i na skupu podataka za učenje i na skupu podataka za testiranje. Potonji problem se nazive **podusklađivanje** (engl. *underfitting*). Za KNN model na slici 6.2a vidi se pretjerano usklađivanje na podatke za učenje, dok se na slici 6.2c vidi problem podusklađivanja. Kako bi model imao zadovoljavajuće performanse i na testnim podacima, potrebno je pronaći

optimalnu dimenziju modela (vidi sliku 6.2b). Ovo se svodi na odabir odgovarajućih vrijednosti hiperparametara modela, ali bez upotrebe skupa podataka za testiranje jer on služi za konačnu procjenu generalizacijskih performansi izgrađenog modela.

Jedan od načina procjene optimalnih hiperparametara modela je korištenje **skupa podataka za validaciju** (validacijski skup) kako je prikazano na slici 6.5. Konkretno, raspoloživi podaci se dijele u tri skupa: skup podataka za učenje, skup podataka za validaciju i skup podataka za testiranje. Na skupu podataka za učenje se procjenjuju parametri modela te se zatim procjenjuju performanse modela na skupu podataka za validaciju. Na temelju rezultata na validacijskom skupu se odabire optimalni model. Konačno, odabrani model se evaluira na testnom skupu kako bi se dobila procjena generalizacijskih performansi modela.



Slika 6.5: Podjela podataka na skup za učenje, skup za validaciju i skup za testiranje.

Nedostatak pristupa s validacijskim skupom je osjetljivost na način podjele podataka. Procjena performansi modela može značajno varirati s podjelom podataka. Također, dijeljenjem podataka na tri skupa se značajno smanjuje količina podataka dostupnih za učenje što može biti problem kod manjih skupova podataka. Stoga se često koristi *k*-struka **unakrsna validacija** (engl. *k-fold cross-validation*). Kod ovog pristupa se skup podataka za učenje dijeli u *k* podskupa, pri čemu se *k* – 1 podskupova koristi za učenje modela, a jedan podskup za validaciju modela. Procedura se ponavlja *k* puta te se dobiva *k* modela s pripadnom procjenom pogreške na temelju koje se računa prosječna pogreška. Princip unakrsne validacije prikazan je na slici 6.6 kada se koristi 5 podskupova.

*k*-struka unakrsna validacija može se ponoviti za različite vrijednosti određenog hiperparametra, primjerice broj susjeda. Na slici 6.6 prikazana je prosječna točnost klasifikacije za različit broj susjeda *K* koja se dobiva metodom unakrsne validacije. Na temelju dobivenog grafa odabire se optimalni broj susjeda.

## 6.3 KNN i SVM u scikit-learn biblioteci

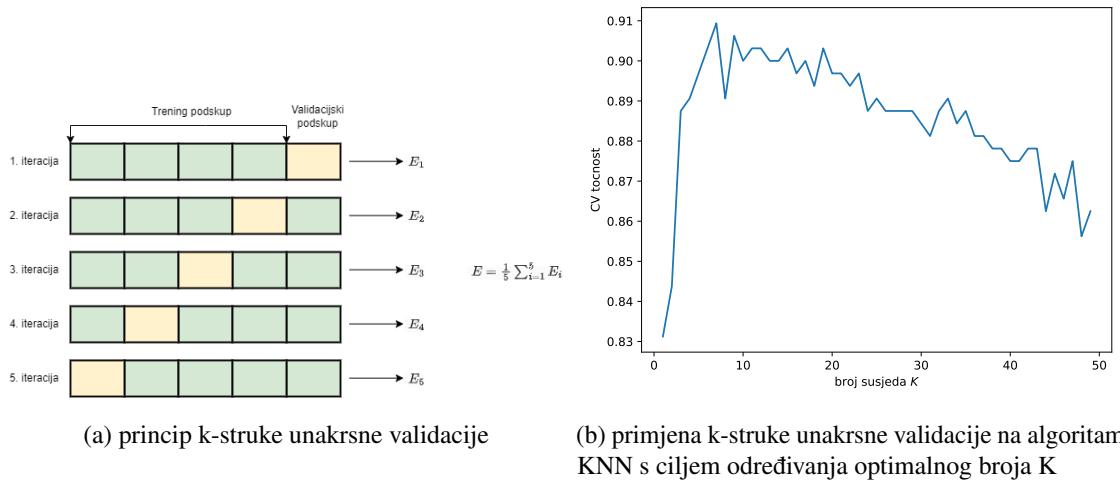
Algoritam K najbližih susjeda za klasifikaciju je u scikit-learn biblioteci implementiran je u obliku klase:

```

class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *,
weights='uniform', algorithm='auto', leaf_size=30, p=2,
metric='minkowski', metric_params=None, n_jobs=None)
  
```

Najvažniji parametri su:

## Poglavlje 6. Klasifikacija primjenom algoritma K najbližih susjeda i primjenom strojeva s potpornim vektorima.



Slika 6.6: Unakrsna validacija.

- `n_neighbors` – broj susjeda  
cjelobrojna vrijednost, `default=5`
- `p` – potencija na Minkowski metriku udaljenosti. Kada je  $p = 1$  koristi se Manhattan udaljenost, kada je  $p = 2$  koristi se Euklidska udaljenost  
cjelobrojna vrijednost, `default=2`

Najvažnije metode ove klase su:

- `.fit(X,y)` koja memorira sve podatkovne primjere za učenje
- `.predict(X)` za izračunavanje izlaza modela na temelju ulaznih vrijednosti ulaznih veličina
- `.predict_proba(X)` za izračunavanje vjerojatnosti klase na temelju ulaznih vrijednosti ulaznih veličina
- `.kneighbors(X=None, n_neighbors=None, return_distance=True)` za izračunavanje indeksa i udaljenost najbližih susjeda za dane primjere u  $X$

SVM algoritam za klasifikaciju u scikit-learn biblioteci implementiran je u obliku klase:

```
class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3,
gamma='scale', coef0=0.0, shrinking=True, probability=False,
tol=0.001, cache_size=200, class_weight=None, verbose=False,
max_iter=-1, decision_function_shape='ovr',
break_ties=False, random_state=None)
```

Najvažniji parametri su:

- $C$  – regularizacijski parametar („jačina“ regularizacije obrnuto je proporcionalna vrijednosti  $C$ ).  
pozitivna decimalna vrijednost, `default=1.0`
- `kernel` – definicija kernel funkcije koja se koristi  
`'linear'`, `'poly'`, `'rbf'`, `'sigmoid'`, `'precomputed'`, `default='rbf'`
- `degree` – stupanj polinomskog kernela ako se on koristi  
cjelobrojna vrijednost, `default=3`
- `gamma` – kernel koeficijent za kernele tipa `'rbf'`, `'poly'` i `'sigmoid'`.  
`'scale'`, `'auto'` ili float vrijednost, `default='scale'`

Najvažnije metode ove klase su:

- `.fit(X,y)` za procjenu parametara modela na temelju podataka za učenje
- `.predict(X)` za izračunavanje izlaza modela na temelju ulaznih vrijednosti ulaznih veličina
- `.predict_proba(X)` za izračunavanje vjerojatnosti klase na temelju ulaznih vrijednosti ulaznih veličina

Primjer 6.1 prikazuje isječak koda koji inicijalizira KNN algoritam i SVM algoritam, podešava njihove parametre i radi predikciju za testne podatke.

■ **Primjer 6.1**

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm

# inicijalizacija i ucenje KNN modela
KNN_model = KNeighborsClassifier(n_neighbors = 5)
KNN_model.fit(X_train_n, y_train)

# inicijalizacija i ucenje SVM modela
SVM_model = svm.SVC(kernel='rbf', gamma = 1, C=0.1)
SVM_model.fit(X_train_n, y_train)

# predikcija na skupu podataka za testiranje
y_test_p_KNN = KNN_model.predict(X_test)
y_test_p_SVM = SVM_model.predict(X_test)
```

### 6.3.1 Metoda unakrsne validacije u scikit-learn biblioteci

Procjena određene metrike za dani model primjenom k-struke unakrsne validacije implementirana je u obliku funkcije:

```
sklearn.model_selection.cross_val_score(estimator, X, y=None,
*, groups=None, scoring=None, cv=None, n_jobs=None, verbose=0,
fit_params=None, pre_dispatch='2*n_jobs', error_score=nan)
```

Najvažniji parametri su:

- `estimator` – scikit-learn model (koji implementira određeni *scoring*)
- `X` – vrijednosti ulaznih veličina
- `y` – vrijednosti izlaznih veličina
- `scoring` – string koji definira metriku za evaluaciju modela ako se ne koristi *scoring* modela
- `cv` – cjelobrojna vrijednost koja definira broj podskupova na koji se dijeli skup za učenje a funkcija vraća:
  - `scores` – polje koje sadrži rezultate za svaku iteraaciju unakrsne validacije.

Primjer 6.2 prikazuje način vrjednovanja modela pomoću metode unakrsnog validacije pri čemu se skup podataka za učenje dijeli u 5 podskupova.

■ **Primjer 6.2**

```
from sklearn.model_selection import cross_val_score

model = svm.SVC(kernel='linear', C=1, random_state=42)
scores = cross_val_score(clf, X_train, y_train, cv=5)
print(scores)
```

## Poglavlje 6. Klasifikacija primjenom algoritma K najbližih susjeda i primjenom strojeva s potpornim vektorima.

Uobičajeno se koristi  $k$ -struka unakrsna validacija kako bi se pronašla optimalna vrijednost hiperparametra određene metode. Često metode imaju više od jednog hiperparametra pa se procjena optimalnih vrijednosti hiperparametara radi „rešetkastom“ pretrage, tj. evaluiraju se modeli za sve kombinacije vrijednosti hiperparametara modela koje su dane od strane korisnika. Ova pretraga je implementirana u obliku klase:

```
class sklearn.model_selection.GridSearchCV(estimator,
param_grid, *, scoring=None, n_jobs=None, refit=True, cv=None,
verbose=0, pre_dispatch='2*n_jobs', error_score=nan,
return_train_score=False)
```

Najvažniji parametri su:

- `estimator` – scikit-learn model (koji implementira određeni scoring)
- `param_grid` – rječnik ili lista ječnika koja sadrži vrijednosti hiperparametara za koje je potrebno izvršiti evaluaciju modela
- `scoring` - string koji definira metriku za evaluaciju modela tijekom unakrsne validacije ako se ne koristi `scoring` modela
- `cv` – cjelobrojna vrijednost koja definira broj podskupova na koji se dijeli skup za učenje

Najvažnije metode ove klase je `.fit(X, y)` kojom se izvršava unakrsna validacija za različite vrijednosti hiperparametara. Rezultati se pohranjuju u atribut ove klase pod nazivom `cv_results`.

Uobičajeno se prilikom traženja optimalnih hiperparametara koristi cjevovod (eng. *pipe*) koji predstavlja niz operacija kako bi se od vrijednosti ulaznih veličina dobila vrijednost izlazne veličine. Primjer 6.3 ilustrira princip traženja optimalnih hiperparametara SVC modela pri čemu se koristi rešetkasta pretraga i cjevovod koji se sastoji od skaliranja podataka (standardizacije) i željenog modela (SVC).

### ■ Primjer 6.3

```
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV

param_grid = {'model__C': [10, 100, 100],
'model__gamma': [10, 1, 0.1, 0.01]}

svm_gscv = GridSearchCV(pipe, param_grid, cv=5, scoring='accuracy',
n_jobs=-1)

svm_gscv.fit(X_train, y_train)

print(svm_gscv.best_params_)
print(svm_gscv.best_score_)
print(svm_gscv.cv_results_)
```

## 6.4 Priprema za vježbu

1. Proučite poglavljje 6.2.
2. Po potrebi dodatno proučite dokumentaciju:
  - a. KNN algoritam u scikit-learn biblioteci

- b. Klasifikacija pomoću strojeva s potpornim vektorima u sciki-learn biblioteci
- c. Postupak unakrsne validacije u scikit-learn biblioteci

## 6.5 Rad na vježbi

1. Isprobajte Python primjere iz poglavlja 6.3 u Visual Studio Code IDE.
2. Riješite dane zadatke.

**Zadatak 6.5.1** Skripta `zadatak_1.py` učitava `Social_Network_Ads.csv` skup podataka [2]. Ovaj skup sadrži podatke o korisnicima koji jesu ili nisu napravili kupovinu za prikazani oglas. Podaci o korisnicima su spol, dob i procijenjena plaća. Razmatra se binarni klasifikacijski problem gdje su dob i procijenjena plaća ulazne veličine, dok je kupovina (0 ili 1) izlazna veličina. Za vizualizaciju podatkovnih primjera i granice odluke u skripti je dostupna funkcija `plot_decision_region` [1]. Podaci su podijeljeni na skup za učenje i skup za testiranje modela u omjeru 80%-20% te su standardizirani. Izgrađen je model logističke regresije te je izračunata njegova točnost na skupu podataka za učenje i skupu podataka za testiranje. Potrebno je:

1. Izradite algoritam KNN na skupu podataka za učenje (uz  $K=5$ ). Izračunajte točnost klasifikacije na skupu podataka za učenje i skupu podataka za testiranje. Usporedite dobivene rezultate s rezultatima logističke regresije. Što primjećujete vezano uz dobivenu granicu odluke KNN modela?
2. Kako izgleda granica odluke kada je  $K = 1$  i kada je  $K = 100$ ?

**Zadatak 6.5.2** Pomoću unakrsne validacije odredite optimalnu vrijednost hiperparametra  $K$  algoritma KNN za podatke iz Zadatka 1.

**Zadatak 6.5.3** Na podatke iz Zadatka 1 primijenite SVM model koji koristi RBF kernel funkciju te prikažite dobivenu granicu odluke. Mijenjajte vrijednost hiperparametra  $C$  i  $\gamma$ . Kako promjena ovih hiperparametara utječe na granicu odluke te pogrešku na skupu podataka za testiranje? Mijenjajte tip kernela koji se koristi. Što primjećujete?

**Zadatak 6.5.4** Pomoću unakrsne validacije odredite optimalnu vrijednost hiperparametra  $C$  i  $\gamma$  algoritma SVM za problem iz Zadatka 1.

## 6.6 Izvještaj s vježbe

Kao izvještaj s vježbe prihvata se web link na repozitorij pod nazivom OSU\_LV.

### Literatura

- [1] Sebastian Raschka, Vahid Mirjalili. Python Machine Learning, Third Edition, 2019.
- [2] <https://www.kaggle.com/datasets/rakeshrau/social-network-ads>