Istraživanje podataka 1 - dopuna za slajdove za vežbe 5, 2020.

1 Unakrsna validacija

Kako algoritmi za klasifikaciju imaju parametre čije se vrednosti zadaju, potrebno je pronaći optimalne vrednosti, tj. vrednosti koje će dati najbolji model za dati skup. Ukoliko se isti trening i test skup koriste pri pravljenju i oceni modela koji se dobijaju promenom vrednosti za jedan ili više parametara istog algoritma (npr. maksimalna dubina drveta odlučivanja ili minimalan broj instanci koji mora biti u čvoru) postoji rizik od preprilagođavanja podacima, jer se optimalna vrednost tih parametara traži tako da se model ponaša dobro i na test skupu. Na taj način, informacije o test skupu se uključuju u proces pravljenja modela, a to ne bi smelo. Kako bi se rešio ovaj problem, pored skupa za treniranje i skupa za testiranje, može da se napravi i skup za proveru (validaciju). Tada se modeli sa različitim vrednostima za prametre prave na osnovu trening skupa, a njihova ocena se vrši na skupu za proveru. Model koji daje najbolje rezultate na skupu za proveru se na kraju ocenjuje sa skupom za testiranje.

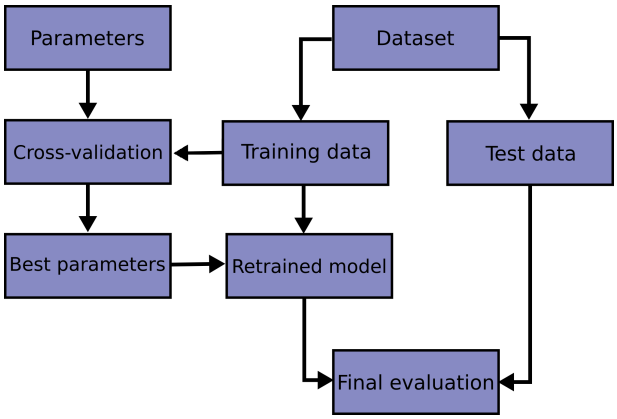
Problem sa podelom skupa na tri dela (trening, test i skup za proveru) je značajno smanjenje broja instanci koje se koriste za učenje modela. Kao rešenje koristi se unakrsna validacija (eng. cross-validation ili CV).

U osnovnom pristupu unakrsne validacije (eng. k-fold CV), skup se deli na trening i test deo, pri čemu se test skup koristi samo za krajnju ocenu modela dobijenog sa optimalnim vrednostima za parametre (slika 1). Trening skup se prvo deli na k manjih delova. Za parametre algoritma se definišu moguće vrednosti (npr. za dubinu drveta moguće vrednosti su 2, 3, 4 ili 5, a za minimalan broj instanci u čvoru 5, 10, 20 ili 25). Za svaku kombinaciju zadatih vrednosti parametara (faza *cross-validation* na slici 1) se procenjuje koliko je dobar model sa tim parametrima tako što se za svaki od k delove dobijenih podelom trening skupa primenjuju sledeći koraci (slika 2):

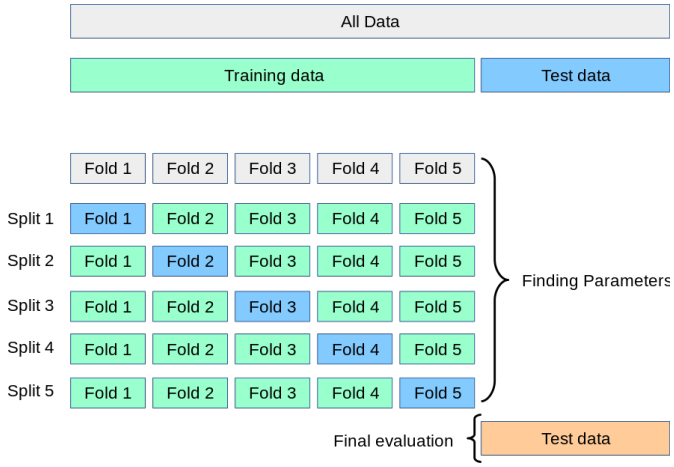
- Pravi se model na osnovu ostalih *k-1* delova.
- Izdvojeni deo, koji nije korišćen za pravljenje modela, koristi se za proveru dobijenog modela, najčešće pomoću preciznosti, a može i druga mera za ocenu.

Procena preciznosti (ili vrednosti neke druge mere za ocenu modela) za korišćenu kombinaciju parametara računa se kao prosečna vrednost preciznosti dobijenih za svaki od k delova kada je korišćen kao skup za proveru. Primetiti da se za jednu kombinaciju vrednosti parametara pravi k modela, svaki model se pravi na *k-1* delova (delovi u jednom redu označeni zelenom bojom na slici 2), a testira na jednom delu (deo označen plavom bojom u jednom redu na slici 2). Svaki od tih modela ima različit skup za treniranje i testiranje. Na ovaj način se obezbeđuje da svaka instanca trening skupa učestvuje u učenju modela sa određenim vrednostima parametara, ali i u proveri modela sa istim vrednostima parametara.

Na osnovu prosečnih preciznosti koje su dobijene za različite vrednosti parametara, izdvajaju se najbolje vrednosti parametara, tj. one za koje je dobijena najveća prosečna preciznost (faza *best parameters* na slici 1) i sa tim parametrima se pravi novi model korišćenjem **celog** skupa za treniranje (faza *retrained model* na slici 1). Primenom tog modela na test skupu izvršava se krajnja ocena modela (faza *final evaluation* na slici 1).



Slika 1: Primena unakrsne validacije za određivanje optimalnih parametara



Slika 2: Podela trening skupa na 5. delova u unakrsnoj validaciji. U svakom koraku se plavi deo koristi za testiranje, a zeleni delovi za pravljenje modela.

1.1 Unakrsna validacija u biblioteci scikit learn

U biblioteci u biblioteci **scikit learn** klasa **sklearn.model_selection.GridSearchCV** obezbeđuje pretragu za najboljim vrednostima za parametre algoritma za klasifikaciju na osnovu zadatih mogućih vrednosti za parametre. Podaci o klasi:

- parametri
  - *estimator* - procenjivač, objekat klase za klasifikaciju (npr. DecisionTreeClassifier)
  - *param_grid* - rečnik ili lista rečnika sa definisanim mogućim vrednostima za parametre procenjivača. Jedan ključ u rečniku odgovara jednom parametu algoritma. Ime ključa mora biti isto kao ime parametra algoritma.
  - *scoring* - mera za ocenu modela
  - *cv* - broj delova za unakrsnu validaciju, default= 3-fold

  - *refit* - da li ponovo napraviti model sa najboljim parametrima nad celim trening skupom. Da bi mogla da se radi predikcija nad drugim skupom potrebno je staviti True. (default=True)

- atributi
  - *cv_results_* - rečnik sa podacima o zadatim parametrima i rezultatima dobijenim za svaku kombinaciju vrednosti parametara. Svakom ključu u rečniku je dodeljen niz vrednosti. Svaka vrednost u tom nizu odgovara rezultatu dobijenom za jednu kombinaciju vrednosti parametara u unakrsnoj validaciji. Neki od ključeva su:
    * *params* kojem je dodeljena lista rečnika. Svaki rečnik u listi sadrži vrednosti parametara u jednoj kombinaciji.
    * *mean_test_score* - srednja vrednost mere za ocenu dobijena na osnovu k test podskupova
    * *std_test_score* - standardna devijacija vrednosti mere za ocenu dobijena na osnovu k test podskupova
  - *best_estimator_* - najbolji klasifikator
  - *best_score_* - najbolji skor
  - *best_params_* - parametri koji daju najbolji rezultat
  - *scorer_* - funkcija za skor, tj. mera za ocenu

- metode
  - *fit(x,y)* - pravljnje modela sa optimalnim parametrima na osnovu skupa (x,y)
  - *predict(x)* - predviđanje klase za test instance

Primer: cv_tree.py

2 Klasifikacija: KNN - K najbližih suseda

U algoritmu K najbližih suseda klasifikacija instance je zasnovana na sličnosti sa drugim instancama. Test instanca se klasifikuje tako što se u trening skupu pronalazi k najbližih instanci (suseda) i test instanci se dodeljuje klasa koja je najzastupljenija među k najbližih suseda.

Osnovni parametri algoritma KNN su:

- k - broj suseda
- *dist* - funkcija za računanje rastojanja između instanci

U najjednostavnijem obliku algoritma KNN, svaki sused ima istu težinu, te klasa koja je najzastupljenija među susedima test instance se dodeljuje test instanci. U nekim okolnostima, bolje je susedima dodeliti težine tako da bliži susedi imaju veći uticaj pri određivanju klase test instance. Tada su težine suseda proporcionalne vrednosti $\frac{1}{d}$, gde je d rastojanje suseda od test instance.

Za računanje rastojanja između instanci najčešće se koriste Euklidsko i Menhetn rastojanje, zbog čega je u okviru preprocesiranja podataka potrebno izvršiti:

- pretvaranje kategoričkih atributa u numeričke
- transformisati numeričke atribute tako da imaju isti ili sličan opseg vrednosti

Algoritam 1 K najbližih suseda

Neka je:

- k zadati broj suseda
- D skup instanci za treniranje; svaka instanca je predstavljena sa (x, y) gde su x vrednosti atributa za predviđanje, a y klasa kojoj instanca pripada
- svaka test instanca z je predstavljena sa (x', y') gde su x' vrednosti atributa za predviđanje, a y' dodeljena klasa

1: za svaku test instancu $z = (x', y')$ uraditi
2: izračunatu $dist(x', x)$, rastojanje između instance z i svake instance $(x, y) \in D$
3: izdvojiti $D_z \subseteq D$, skup k najbližih trening instanci test instanci z
4: $y' = \arg\max_v \sum_{(x_i, y_i) \in D_z} I(v = y_i)$

2.1 Klasifikacija - KNN u scikit-learn biblioteci

2.1.1 Preprocesiranje

- Modul sklearn.preprocessing sadrži klase korisne za preprocesiranje podataka.
- Za standardizacija vrednosti atributa može da se koristi
  - funkcija **scale(x)** - vraća standardizovane vrednosti
  - klasa **StandardScaler** koja ima metode
    - *fit(x)* - računa srednju vrednost i standardnu devijaciju koji se koristite za transformaciju
    - *transform(x)* - izvršava transformaciju vrednosti
    - *inverse_transform(x)* - transformiše u originalne vrednosti
    - *fit_transform(x)* - računa srednju vrednost i standardnu devijaciju i vrši transformaciju vrednosti
- Za transformaciju vrednosti svakog atributa na određeni interval može se koristiti klasa **MinMaxScaler(feature_range=(0, 1))**. Podrazumevani opseg vrednosti koji će imati atributi nakon transformacije je $[0, 1]$ Osnovne metode klase **MinMaxScaler** su kao kod klase **StandardScaler**:
  - *fit(x)*
  - *transform(x)*
  - *fit_transform(x)*
  - *inverse_transform(x)*

2.1.2 KNN

Algoritam KNN je implementiran preko klase **sklearn.neighbors.KNeighborsClassifier** koja ima

- parametre

  - *n_neighbors* - broj suseda za klasifikaciju test instance (default = 5)
  - *p* - stepen ukoliko se koristi rastojanje Minkovskog, koje je podrazumevana mera za rastojanje
  - *metric* - izbor metrike za rastojanje, (default 'minkowski'), moguće vrednosti su definisane u **sklearn.neighbors.DistanceMetric**
  - *weights* - težina suseda
    * *uniform* - svi susedi imaju istu težinu (default)
    * *distance* - težina suseda je obrnuta rastojanju - najbliži sused ima najveću težinu, najudaljeniji sused ima najmanju težinu

- metode
  - *fit(x)* - pravi model
  - *predict(x)* - određuje klase instancama
  - *kneighbors(x)* - vraća rastojanja od test instance do svakog suseda, kao i poziciju suseda u trening skupu

Primer: cv_knn.py

2.2 Klasifikacija - KNN u IBM SPSS Modeleru

Neke od opcija su:

- Preprocesiranje podataka
  - Normalizacija numeričkih atributa (opciono)

$$x' = \frac{2 * (x - x_{min})}{x_{max} - xmin} - 1$$

  - Kodiranje kategoričkih atributa primenom kodiranja *jedan-od-c*, gde je c broj kategorija. Vrednost se kodira kao binarni vektor dimenzije c. Prva kategorija ima vrednost (1,0,...,0), druga (0,1,0,...,0), a poslednja (0,0,...,1).

- Metrike za računanje rastojanja između instance koja se klasifikuje i suseda su:
  - (Težinsko) Euklidsko rastojanje
  - (Težinsko) Menhetn rastojanje
  - težina atributa se računa prema značajnosti atributa

$$w_i = \frac{FI_i}{\sum_{p=1}^{d} FI_p}$$

- Unakrsna validacija za određivanje najboljeg k iz opsega $[k_{min}, k_{max}]$
  - bira se k sa najmanjom prosečnom stopom greške

- Izbor atributa (izbor unapred)
  - Korisnik može da zada koji atributi će se koristi za predviđanje, a može i da izabere opciju da algoritam pronađe najbolje atribute za predviđanje.

  - Algoritam u jednom koraku bira jedan atribut koji nije među već izabranim atributima i koji najviše doprinosi smanjenju greške klasifikacije. Postupak se ponavlja dok se ne zadovolji jedan od kriterijuma za zaustavljanje.

- Instance sa nedostajućim vrednostima se ne uzimaju u obzir.

Primer: ipVezbe52020_knn.str