Istraživanje podataka 1

Uvod

Ne postoji stroga definicija **istraživanja podataka**, ali se najčešće koriste:

- proces koji uključuje prikupljanje podataka, njihovo čišćenje, obradu, analizu i dobijanje korisnih saznanja o njima
- pronalaženje skrivenih informacija u bazi podataka
- netrivijalno izdvajanje implicitnih, prethodno nepoznatih i potencijalno korisnih informacija iz baza podataka
- integralni deo otkrivanja znanja u bazama podataka (Knowledge Discovery in Databases)

Istraživanje podataka je **potrebno** jer:

- stalno se prikupljaju velike količine podataka
- postoji velika količina ravnih podataka (raw data) za obradu
- tradicionalne metode za analizu nisu pogodne zbog količine i prostorno-vremenske prirode podataka, a često veliki deo podataka nikada i ne stiže do analize
- postoje sakrivene informacije koje nisu odmah ili lako uočljive
- omogućava dobijanje konciznih i upotrebljivih informacija za neki cilj
- ima primenu u različitim oblastima kao što su nauka, medicina, inženjerstvo, poslovanje
- računari su snažniji

Poreklo istraživanja podataka može se naći u statistici, veštačkoj inteligenciji, mašinskom učenju, prepoznavanju obrazaca, tehnologijama baza podataka i paralelnom i distribuiranom računarstvu. IP je **zasnovano na algoritmima**. Svaki algoritam pokušava da ukalupi podatke u neki model. Bira se model koji je najbliži karakteristikama podataka. Neke od oblasti koje su bliske IP su Big data, Predictive analytics, Data Science i slično.

Faze procesa IP su skladištenje podataka, njihova transformacija, prečišćavanje, ponovno skladištenje i upotreba, istraživanje. **Izazovi i problemi u procesu IP** su razni:

- nema gotovih recepata, već imamo veliki broj problema i mogućih rešenja
- veliki broj različitih formata i tipova podataka, kao i složeni i heterogeni podaci
- interpretacija i vizuelizacija rezultata
- velika količina ulaznog materijala (algoritmi moraju biti skalabilni)
- veliki broj atributa (dimenzionalnost)
- kvalitet podataka (nedostajući podaci, irelevantni podaci, ...)
- dostupnost ljudima koji nisu stručnjaci
- narušavanje privatnosti
- neautorizovano korišćenje podataka
- profilisanje (pogrešno kvalifikovanje)

Podaci

Podaci (data set) predstavljaju skup **objekata** i njihovih **atributa**. Atributi su svojstva ili karakteristike objekta. Skup atributa opisuje objekat. Vrednosti atributa su brojevi ili simboli koji su pridruženi atributu. Isti atribut može biti preslikan u različite vrednosti atributa (npr. dužina u metrima ili kilometrima). Različiti atributi mogu da budu preslikani u isti skup vrednosti, pri čemu osobine atributa mogu da budu različite (npr. godine i težina su celobrojne vrednosti, ali težina može da se smanjuje, a godine ne). Prema broju vrednosti koje mogu da sadrže, atributi mogu biti:

- **diskretni** - konačan ili prebrojivo beskonačan skup vrednosti. Najčešće se prikazuju kao celobrojne promenljive. **Binarni** atributi su specijalan slučaj diskretnih atributa koji imaju tačno dve vrednosti.
- **kontinuirani (neprekidni, kontinualni)** - skup vrednosti čine realni brojevi. Najčešće se prikazuju kao realni brojevi u pokretnom zarezu.
- **asimetrični** - kod njih je jedino bitno prisustvo ne-nula vrednosti. **Binarni asimetrični** atributi imaju tačno dve vrednosti, od kojih je samo jedna bitna. Na primer, za svakog studenta sa 0 ili 1 beležimo da li je slušao neki kurs i želimo da izmerimo sličnost između studenata po broju kurseva koje zajedno slušaju. Fakultet sadrži veliki broj kurseva i dva studenta će onda biti slični po svim kursevima koje ne sluša ni jedan od njih, pa su nam zbog toga zapravo bitni samo kursevi koje su oba studenta slušala.

Prema operacijama koje se mogu koristiti, atributi mogu biti (svaki sledeći tip podržava operacije prethodnog tipa):

Tip	Operacije	Dodatne operacije	Primer
Imenski	**različitost** $(=,\ \neq)$	**modus, entropija, korelacija,** $\mathcal{X}^2$ **test**	**pol, poštanski kod, boja očiju**
Redni	**uređenje** $(<,\ \leq,\ >,\ \geq)$	**medijana, percentil, korelacija ranga**	**ocene, redni brojevi nečega**
Intervalni	**aditivnost** $(+,\ -)$	**srednja vrednost, standardna devijacija**	**datumi, temperatura**
Razmerni	**multiplikativnost** $(\cdot,\ /)$	**geometrijska sredina, harmonijska sredina**	**dužina, masa, količina**

Kvalitativni (kategorički) atributi su imenski i redni, a **kvantitativni (numerički) atributi** su intervalni i razmerni. Kvalitativni atributi nemaju većinu svojstava brojeva.

Nezavisni (tabelarni) podaci su podaci koji međusobno nisu povezani. Najčešće su to multidimenzionalni ili tekstualni podaci. Za njihovo čuvanje pogodne su baze podataka. Sastoje se od slogova (objekata), a svaki slog se sastoji od polja (atributa). Datoteke se u bazama mogu čuvati tako što se u redovima čuvaju dokumenti, u kolonama reči, a u poljima broj pojavljivanja određene reči u određenom dokumentu. **Skup multidimenzionalnih podataka** je skup od n slogova $\overline{X_1}$, ..., $\overline{X_n}$ takvih da svaki od slogova $\overline{X_i}$ sadrži skup od d osobina označenih sa $(x_i^1,\ \ldots,\ x_i^d)$. **Retki podaci** su podaci gde postoji mali broj podataka koji su značajni (ne-nula) podaci. Na primer, to je čest slučaj pri čuvanju dokumenata kroz tabele jer se mnoge reči pojavljuju samo u jednom ili par dokumenata, a u ostalima ne. **Zavisni podaci** su podaci kod kojih postoji **implicitna** ili **eksplicitna zavisnost**. Na primer, implicitna zavisnost se uočava kod senzora koji se ponašaju slično i ako se u nekom trenutku uoči veliko odstupanje ono je od interesa za istraživanje. Eksplicitna zavisnost je npr. prisutna kod grafova poseta veb sajtu, uticaja lekova na druge lekove ili bolesti i slično.

Podaci sa poretkom su podaci gde atributi imaju odnose koji podrazumevaju vremenski ili prostorni redosled. **Vremenske serije** imaju implicitnu zavisnost od prethodnih merenja. To su na primer EKG, temperatura i slično. **Sekvencijalni podaci** sastoje se od skupa koji predstavlja sekvencu objekata, npr. sekvenca slova ili sekvenca reči. Redosled određuje pozicija u sekvenci. To su npr. DNK i RNK sekvence. **Prostorni podaci** određuju prostorne lokacije. Kao kod vremenskih serija, ovde imamo prostornu/fizičku korelaciju. Svi ovi podaci sadrže više atributa koji prikazuju ponašanje, kao i jedan ili više **kontekstualnih atributa** (vreme, lokacija ili pozicija) koji određuju sam kontekst. Postoje i **prostorno-vremenski podaci** gde razlikujemo dva tipa:

- i prostorni i vremenski atributi mogu biti kontekstualni (npr. merenje varijacije temperature mora kroz vreme)
- vremenski atribut je kontekstualan, a prostorni modelira ponašanje (npr. analiza trajektorija)

Grafovski (mrežni) podaci su podaci kod kojih su vrednosti pridružene čvorovima u grafu. Objekte predstavljamo kao čvorove, a njihove međusobne odnose kao grane. Granama možemo pridružiti i usmerenja i težine za dodatni opis odnosa. Koriste se npr. za predstavljanje veb stranica ili hemijskih jedinjenja.

Najznačajniji gradivni blokovi u IP su:

1. **podaci -** najčešće su prethodno prikupljani iz nekog drugog razloga. Te podatke možemo smestiti u matricu sa n redova i d kolona, gde je n broj slogova, a d broj atributa. Potrebno je izvršiti i **preprocesiranje**, tj. pripremu i prečišćavanje podataka. **Otkrivanje anomalija** je određivanje redova u matrici koji su jako različiti od ostatka redova. **Element van granica (outlier)** je podatak koji je u značajnoj meri različit od ostalih podataka. To su npr. spam poruke, upadi u računarski sistem, zloupotreba kartica i slično.
2. **pravila pridruživanja -** tehnike koje otkrivaju odnose među podacima. Na primer, imamo podatke o transakcijama gde su u vrstama transakcije, a u kolonama stavke u obliku **retke binarne baze** (koristimo 0 i 1 gde 1 znači da je u transakciji i kupljena stavka j). U datoj binarnoj matrici posmatrajmo podskupove kolona A takve da sve vrednosti u tim kolonama u jednoj vrsti imaju vrednost 1. Tada A nazivamo skupom stavki, a sa $\#(A)$ označavamo broj pojavljivanja skupa stavki A. Sa $A \to B$ označavamo da je skup stavki B pridružen skupu stavki A, odnosno u našem primeru to bi značilo "ako korisnik kupi stavke A, verovatno će kupiti i stavke B". U opštem slučaju, elementi matrice ne moraju nužno biti binarne vrednosti. Zadatak je naći pravila pridruživanja koja povezuju atribute pri čemu su nam interesantna pravila koja zadovoljavaju neki nivo određenih svojstava. Svojstva koja opisuju pravilo pridruživanja $A \to B$ su:
   - **podrška (support)**: količnik broja transakcija koje sadrže A i B u odnosu na ukupan broj transakcija N. Opisuje koliko često važi posmatrano pravilo pridruživanja.

$$sup(A \to B) = \frac{\#(A \cup B)}{N}$$

   - **pouzdanost (confidence)**: količnik broja transakcija koje sadrže A i B u odnosu na broj transakcija koje sadrže A. Opisuje verovatnoću da se desi B ako već važi A.
3. **klasifikacija -** određivanje relacija između kolona. Na osnovu atributa objektima dodeljujemo klase. Naziva se i **klasifikacija pod nadzorom**.
4. **klasterovanje** - grupisanje vrsta po sličnosti. Naziva se i **klasifikacija bez nadzora**.
5. **analiza i vizuelizacija rezultata**

Mere sličnosti

Mere sličnosti pružaju način za određivanje sličnosti ili različitosti objekata, atributa i slično. Podaci mogu biti različitog tipa, strukture, raspodele, dimenzionalnosti i tako dalje. **Blizina (proximity)** označava i sličnost i različitost dva objekta. **Sličnost** predstavlja numeričku meru koliko su dva objekta ili atributa slična i najčešće uzima vrednosti iz intervala [0, 1], gde 0 označava da objekti nisu nimalo slični, a 1 da su objekti isti. **Različitost (rastojanje)**

predstavlja numeričku meru koliko su dva objekta ili atributa različiti i najčešće uzima vrednosti iz intervala [0, $+\infty$), gde 0 označava da su objekti isti, a vrednosti veće od nule opisuju u kojoj meri su oni različiti. Primer nekih funkcija sličnosti i različitosti atributa p i q:

Tip	Sličnost	Različitost
Nominalni	$s = \begin{cases} 1, & p = q \\ 0, & p \neq q \end{cases}$	$d = \begin{cases} 1, & p \neq q \\ 0, & p = q \end{cases}$
Redni - vrednosti se preslikavaju u skup [0, $n-1$] gde je n broj vrednosti	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$	$d = \frac{\lvert p-q \rvert}{n-1}$
Intervalni ili razmerni	$s = -d$, $s = \frac{1}{1+d}$, $s = 1 - \frac{d-d_{min}}{d_{max}-d_{min}}$	$d = \lvert\ p - q\ \rvert$

Funkcija rastojanja d je **metrika** ako važi:

1. **pozitivna određenost**: $(\forall\ p,\ q)\ d(p,\ q) \geq 0$ i $d(p,\ q) = 0\ \Leftrightarrow\ p = q$
2. **simetrija**: $(\forall\ p,\ q)\ d(p,\ q) = d(q,\ p)$
3. **nejednakost trougla**: $(\forall\ p,\ q,\ r)\ d(p,\ r) \leq d(p,\ q) + d(q,\ r)$

Ako je funkcija d metrika i važi $(\forall\ p,\ q,\ r)\ d(p,\ r) \leq \max\{d(p,\ q),\ d(q,\ r)\}$ onda je d **ultrametrika**.

Kod kvantitativnih podataka računamo rastojanje tačaka $X = (x_1,\ \ldots,\ x_n)$ i $Y = (y_1,\ \ldots,\ y_n)$ u n-dimenzionalnom prostoru. Najčešće korišćena mera je **rastojanje Minkovskog** $(L_p mera)$:

$$d(X,\ Y) = (\sum_{i=1}^{n} \lvert\ x_i - y_i\ \rvert^p)^{\frac{1}{p}}$$

Specijalni slučajevi rastojanja Minkovskog:

- **Hamingovo (Menhetn, taksi blok) rastojanje** - za $p = 1$ odnosno

$$d(X,\ Y) = \sum_{i=1}^{n} q_i,\ q_i = \begin{cases} 1, & x_i \neq y_i \\ 0, & x_i = y_i \end{cases}$$

- **Euklidskog rastojanje** - za $p = 2$ odnosno

$$d(X,\ Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- **Supremum rastojanje** - za $p \to \infty$ odnosno:

$$d(X,\ Y) = \max_{1 \leq i \leq n} \lvert\ x_i - y_i\ \rvert$$

Rastojanje Minkovskog nije pogodno za upotrebu kod retkih višedimenzionalnih podataka sa nepoznatom raspodelom, šumovima, kao ni ako postoje lokalno irelevantni atributi zbog šuma koji se akumulira pri izračunavanju. **Mahalanobisovo rastojanje** je:

$$d(X,\ Y) = \sqrt{(X-Y)\Sigma^{-1}(X-Y)^T}$$

gde je Σ matrica kovarijansi podataka. Ovo rastojanje je korisno kada su atributi u korelaciji, imaju različite opsege vrednosti i raspodela podataka je približno normalna. Sa druge strane, računanje samog rastojanja je dosta skupo. Ukoliko želimo nekim atributima da dodelimo veću važnost, koristimo težine. **Rastojanje Minkovskog sa težinama**:

$$d(X,\ Y) = (\sum_{i=1}^{n} a_i \mid x_i - y_i \mid^p)^{\frac{1}{p}}$$

Kod binarnih atributa moguće je koristiti posebne mere bliskosti. Označimo sa M_{01} broj atributa koji su 0 u X i 1 u Y, sa M_{10} broj atributa koji su 1 u X i 0 u Y, sa M_{00} broj atributa koji su 0 u X i 0 u Y i sa M_{11} broj atributa koji su 1 u X i 1 u Y. Neke od mera sličnosti su:

- **Jednostavno uparivanje koeficijenata (SMC)**:

$$SMC = \frac{M_{11} + M_{00}}{M_{00} + M_{01} + M_{10} + M_{11}}$$

- **Žakardov koeficijent** - kod asimetričnih atributa:

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

  Prošireni Žakardovi koeficijenti (koeficijenti Tanimotoa) su varijanta Žakardovih koeficijenata za atribute sa prebrojivim i neprekidnim vrednostima:

$$T(X,\ Y) = \frac{X \circ Y}{||X||^2 + ||Y||^2 - X \circ Y}$$

- **Kosinusna sličnost** - kada je M_{00} veliko, ali može da se koristi i kod nebinarnih vektora:

$$\cos(X, Y) = \frac{X \circ Y}{||X|| \cdot ||Y||}$$

Korelacija dva objekta koji imaju binarne ili neprekidne atribute je mera linearnog odnosa između njihovih atributa. **Pirsonov koeficijent korelacije**:

$$\rho_{X,\ Y} = \frac{cov_{X,\ Y}}{\sigma_X \sigma_Y},\ cov_{X,\ Y} = \frac{\sum_{i=1}^{n}(x_k - \overline{X})(y_k - \overline{Y})}{n-1}$$

Ako korelacija ima vrednost 1 (-1) objekti imaju **perfektno pozitivan (negativan) linearni odnos**, odnosno $X = aY + b$.

Kod kategoričkih podataka sličnost dva objekta možemo definisati kao:

$$s(X,\ Y) = \sum_{i=1}^{n} S(x_i,\ y_i),$$

gde je $S(x_i,\ y_i)$ neka mera sličnosti nad pojedinačnim atributima. Najjednostavnija mera bi bila $S(x_i,\ y_i) = \begin{cases} 1,\ x_i = y_i \\ 0,\ x_i \neq y_i \end{cases}$. Ova mera ne uzima u obzir relativnu frekvenciju atributa. Označimo sa $p_i(x)$ broj slogova u kojima i-ti atribut uzima vrednost x. Mere koje uključuju učestalost su:

- **Inverzna učestalost pojavljivanja**

$$S(x_i,\ y_i) = \begin{cases} \frac{1}{p_i(x_i)^2},\ x_i = y_i \\ 0,\ x_i \neq y_i \end{cases}$$

- **Pojavljivanje je dobro**

$$S(x_i,\ y_i) = \begin{cases} 1 - p_i(x_i)^2,\ x_i = y_i \\ 0,\ x_i \neq y_i \end{cases}$$

Sličnost dva dokumenta najbolje se ocenjuje ako se koriste reči koje su zajedničke. Za normalizaciju uparivanja reči u slučajevima kada ima reči koje se retko javljaju i koje se često javljaju koristi se funkcija $id_i = \log \frac{n}{n_i}$, gde je n_i broj dokumenata u kojima se javlja reč i, a n ukupan broj dokumenata. Za smanjenje mogućnosti da pojava neke česte reči utiče na sličnost dokumenata mogu da se koriste i funkcije $f(x_i) = \sqrt{x_i}$ i $f(x_i) = \log x_i$.
Normalizovana frekvencija za i-tu reč se zatim definiše kao $h(x_i) = f(x_i)id_i$. Nad ovim frekvencijama je moguće koristiti kosinusno ili Žakardovo rastojanje kao i u slučaju binarnih podataka:

$$\cos(X,\ Y) = \frac{\sum_{i=1}^{n} h(x_i) \times h(y_i)}{\sqrt{\sum_{i=1}^{n} h(x_i)^2} \times \sqrt{\sum_{i=1}^{n} h(y_i)^2}}$$

$$J(X,\ Y) = \frac{\sum_{i=1}^{n} h(x_i) \times h(y_i)}{\sum_{i=1}^{n} h(x_i)^2 + \sum_{i=1}^{n} h(y_i)^2 - \sum_{i=1}^{n} h(x_i) \times h(y_i)}$$

Sličnost dva sloga $X = (X_n, X_c)$ i $Y = (Y_n, Y_c)$ sa **mešanim atributima** je:

$$S(X,\ Y) = \lambda S_{Num}(X_n, Y_n) + (1 - \lambda) S_{Cat}(X_c,\ Y_c)$$

gde λ određuje relativnu važnost kategoričkih i numeričkih atributa, a S_{Num} i S_{Cat} su neke mere sličnosti za numeričke i kategoričke atribute.

Mere kod diskretnih podataka:

- **Edit rastojanje** - različitost dve niske po tome koliko je potrebno za transformaciju niske $X = (x_1,\ \ldots,\ x_m)$ u nisku $Y = (y_1,\ \ldots y_n)$. Za prvih i simbola X i prvih j simbola Y cena transformacije je:

$$Edit(i,\ j) = \min \begin{cases} Edit(i-1,\ j) + C_b \\ Edit(i,\ j-1) + C_u \\ Edit(i-1,\ j-1) + I_{ij} C_z \end{cases}$$

gde su $C_b,\ C_u,\ C_z$ redom cena brisanja, umetanja i zamene, a I_{ij} indikator jednakosti i-tog simbola X i j-tog simbola Y.

- **Najduža zajednička podniska (LCSS)** - sličnost na osnovu najduže zajedničke podniske. Za prvih i simbola X i prvih j simbola Y najduža zajednička podniska je:

$$LCSS(i,\ j) = \max \begin{cases} LCSS(i-1,\ j-1) + 1,\ x_i = y_j \\ LCSS(i-1,\ j),\ x_i\ nije\ upareno \\ LCSS(i,\ j-1),\ y_j\ nije\ upareno \end{cases}$$

Neke mere su zasnovane na teoriji informacija. **Entropija** događaja X sa n mogućih ishoda sa verovatnoćama ishoda $p_1,\ \ldots,\ p_n$ je:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Pripada intervalu $[0,\ \log_2 n]$ i meri nepredvidljivost nekog događaja. Mala entropija znači veća sigurnost, a velika entropija znači veća nepredvidljivost. **Zajedničke informacije** za događaje X i Y definišemo sa:

$$I(X,\ Y) = H(X) + H(Y) - H(X,\ Y)$$

Mere zasnovane na gustini mere stepen bliskosti objekata u nekoj oblasti. Najčešće se koriste:

- **Euklidska gustina** - broj tačaka po jedinici površine ili zapremine
- **Gustina verovatnoće** - procena distribucije podataka na osnovu izgleda
- **Graf zasnovane gustine** - povezanost

Priprema podataka

Priprema (preprocesiranje) podataka je bitno jer kvalitet podataka direktno utiče na rezultate. Izvorni podaci mogu biti u različitim formatima, mogu postojati nedostajući i nekonzistentni podaci i slično.

Izdvajanje karakteristika je tehnika prepocesiranja koja iz ravnih podataka izdvaja karakteristike. Na primer, izdvajanje nekih piksela koji su značajni na slikama kako bismo razlikovali pse i mačke.

Prenosivost tipova podataka je tehnika preprocesiranja koja predstavlja transformaciju podataka iz jednog tipa u drugi. Potrebna je jer neke karakteristike onemogućavaju primenu gotovih alata, a pojedini algoritmi rade samo sa određenim tipovima podataka. U ovom procesu moguć je gubitak informacija. **Diskretizacija** je transformacija neprekidnih u kategoričke atribute. Obično se primenjuje na atribute u klasifikaciji ili pravilima pridruživanja. Prvo se bira broj kategorija n. Zatim se interval brojeva deli na n podintervala i sve vrednosti

iz jednog podintervala preslikavaju se u istu kategoričku vrednost. Između dobijenih vrednosti ne postoji uređenje. Intervali se mogu izabrati na više načina:

- **jednake širine intervala** - ako vrednosti atributa upadaju u interval $[m,\ M]$ on se deli na n jednakih delova. Ovakva podela je nekorektna ako je distribucija elemenata po intervalima neravnomerna.
- **jednaki** $\log$**-intervali** - ako su podintervali oblika $[a,\ b]$ onda je vrednost $\log b - \log a$ jednaka za svaki interval. Ovakva podela je nekorektna ako je distribucija elemenata po intervalima neravnomerna. Ako distribucija elemenata može da se modelira funkcijom f mogu se birati intervali $[a,\ b]$ tako da je vrednost $f(b) - f(a)$ jednaka za svaki interval.
- **jednak broj elemenata u intervalu** - vrednosti atributa se prebroje i dobijeni broj k se podeli sa n. Vrednosti atributa se sortiraju i u svaki interval se smešta $\frac{k}{n}$ elemenata. Čvor Binning u SPSS Modeleru radi na ovaj način.

Binarizacija je transformacija neprekidnih i diskretnih atributa u binarne. Obično se primenjuje na atribute u pravilima pridruživanja. Ako kategorički atribut ima n vrednosti formira sa n binarnih atributa tako da svaki od njih odgovara jednoj vrednosti kategoričke promenljive.

Predstavljanje tekstualnih podataka preko retkih numeričkih vektora nije pogodno za najveći broj metoda IP, a ograničen je i broj mera koje se mogu koristiti. **Latentna semantička analiza (LSA)** prevodi tekst u neretku reprezentaciju manje dimenzije. Dokument $X = (x_1,\ \ldots,\ x_d)$ se skalira funkcijom:

$$\frac{X}{\sqrt{\sum_{i=1}^{d} x_i^2}}$$

Za ovako dobijene podatke moguće je primeniti Euklidsko rastojanje.

Podaci se iz vremenskih serija transformišu u diskretne niske **SAX algoritmom (simbolička aproksimacija agregata)**. U prvom koraku se serija deli u prozore veličine w za koje se računa prosečna vrednost atributa. U drugom koraku se srednje vrednosti vremenskih serija diskretizuju pomoću tehnike sa intervalima koji imaju isti broj elemenata. Pretpostavka je da vrednosti u vremenskim serijama imaju normalnu raspodelu. Na osnovu toga računaju se srednja vrednost i standardna devijacija vrednosti vremenskih serija iz prozora, a za određivanje granica intervala koriste se kvantili normalne raspodele. Diskretizacija se najčešće vrši u 3 do 10 intervala.

Podaci se iz vremenskih serija transformišu u numeričke podatke putem **diskretne transformacije talasićima (DWT)**, a može da se koristi i **diskretna Furijeova transformacija (DFT)**. Na ovaj način se omogućava upotreba algoritama koji rade sa multidimenzionalnim podacima. Osobina ovih metoda je da dobijeni koeficijenti nisu zavisni kao u originalnim podacima.

Diskretne niske mogu se transformisati u numeričke podatke u dva koraka:

- Diskretne niske se konvertuju u skup binarnih vremenskih serija čiji je broj jednak broju različitih simbola.
- Svaka serija se konvertuje u multidimenzionalni vektor pomoću transformacije talasićima. Osobine iz ovih vektora se kombinuju i formira se multidimenzionalni slog.

Čišćenje podataka je tehnika preprocesiranja koja obuhvata:

- rad sa **nedostajućim podacima** - podaci mogu da nedostaju jer informacije nisu prikupljene ili neki atributi nisu primenljivi u svim slučajevima. Moguće je obraditi slog sa nedostajućim podacima na više načina:
  - Ceo slog se odbacuje.
  - Nedostajuća vrednost se procenjuje i unosi - **imputacija**.
  - Neki algoritmi mogu da obrađuju slogove sa nedostajućim vrednostima, pa ih nije potrebno menjati ili izbacivati.
- rad sa **nekorektnim podacima** - neki podaci mogu biti nekonzistentni.
- rad sa **dupliranim podacima** - najčešće se javljaju kod spajanja podataka iz više izvora. Ovakvi podaci se najčešće eliminišu, ali ne uvek.
- **skaliranje** - transformacija promenljive označava transformaciju koja se primenjuje na sve vrednosti te promenljive. U statistici se često koriste funkcije $\sqrt{x}$, $\log x$ i $\frac{1}{x}$ radi transformacije podataka koji nemaju normalnu raspodelu u podatke koji je imaju. U IP postoje i drugi razlozi, npr. primena $\log$ funkcije na vrednosti iz opsega [1, 1000000000] kako bi se dobili bolji odnosi kod poređenja. Primenom nekih transformacija moguće je promeniti prirodu podataka, npr. sa $\frac{1}{x}$. **Standardizacija** je skaliranje tako da srednja vrednost bude 0, a standardna devijacija 1. Ako atribut a ima srednju vrednost μ_a i standardnu devijaciju σ_a, onda se njegove vrednosti standardizuju izrazom $\frac{x-\mu_a}{\sigma_a}$. Za normalnu raspodelu dobijene vrednosti najčešće se nalaze u intervalu [-3, 3]. **Normalizacija** je skaliranje vrednosti u određeni opseg. Za svođenje na interval [0, 1] primenjuje se **min-maks skaliranje**: $\frac{x-min}{max-min}$.

Redukcija podataka je tehnika preprocesiranja koja za cilj ima smanjenje količine podataka, što omogućava efikasniju primenu algoritama. Postoje različite tehnike redukcije:

1. **Agregacija** je kombinovanje dva ili više atributa (ili objekata) u jedan. Na ovaj način smanjuje se broj atributa ili objekata, menja se skala i dobijaju stabilniji podaci. Nedostatak agregacije je mogući gubitak nekih informacija.
2. **Uzimanje uzorka** koristi se jer obrada kompletnog skupa podataka koji je od interesa može biti skup ili vremenski zahtevan. Uzorak je **reprezentativan** ako ima aproksimativno iste osobine kao i originalni skup podataka. Korišćenjem uzoraka koji su reprezentativni dobija se efekat skoro isti kao da je rađeno na kompletnom skupu podataka. Veličina uzorka treba da bude dovoljno velika da se ne naruši struktura objekta ili uklone interesantne osobine. **Prost (jednostavan) slučajan uzorak** je uzorak gde svaki element ima jednaku verovatnoću da bude izabran. Uzorkovanje može biti **sa vraćanjem** ili **bez vraćanja**. **Pristrasno uzorkovanje** podrazumeva da su neki podaci

važniji od drugih, odnosno imaju veću verovatnoću da budu izabrani. **Stratifikovano uzorkovanje** podrazumeva da se podaci dele u više delova, a zatim se bira slučajni uzorak iz svakog od tih delova.

3. **Izbor karakteristika** je proces eliminacije redundantnih i irelevantnih karakteristika. Često se formiraju i novi atributi koji uključuju važne karakteristike zbog efikasnije obrade.
4. **Redukcija pomoću rotacije osa** predstavlja automatsko uklanjanje koordinatnih osa pomoću rotacije. Koriste se algoritmi **PCA (Principal Component Analysis)** i **SVD (Singular Value Decomposition)**. PCA je tehnika linearne algebre koja pronalazi nove atribute koji su linearne kombinacije originalnih atributa, koji su ortogonalni jedni na druge i obuhvataju maksimalno raznovrsne podatke. Korisna je jer smanjuje broj dimenzija, nalazi obrasce u podacima velike dimenzionalnosti i omogućava vizuelizaciju podataka velike dimenzionalnosti. Osnovna ideja je da se podaci rotiraju u sistem sa osama gde je najveći broj varijansi pokriven najmanjim brojem dimenzija. Novi sistem sa osama zavisi od korelacije između atributa. Najčešće se primenjuje posle oduzimanja srednje vrednosti od svake tačke. Za matricu podataka D reda $m \times n$ može da se formira matrica kovarijansi C gde je c_{ij} kovarijansa i-te i j-te kolone. Kovarijansa je mera kako se atributi menjaju u paru, a ako je $i = j$ tada je kovarijansa jednaka varijansi atributa. Cilj PCA je nalaženje transformacije podataka za koju važi:

   - Svaki par novodobijenih atributa ima kovarijansu 0.
   - Atributi su uređeni u opadajućem redosledu u odnosu na veličinu varijanse koja je pokrivena od strane atributa.
   - Između atributa postoji ortogonalnost, tako da svaki naredni atribut pokriva što je moguće veći broj preostalih varijansi.

   Transformacija se vrši upotrebom sopstvenih vrednosti matrice kovarijansi C. Neka važi:

   - λ_i su nenegativne sopstvene vrednosti C uređene u redosledu $\lambda_1 \geq \ \ldots \ \geq \lambda_n$.
   - $U = [u_1, \ \ldots, \ u_n]$ je matrica sopstvenih vektora C uređena tako da i-ti vektor odgovara i-toj najvećoj sopstvenoj vrenosti.
   - Matrica D je prethodno pripremljena tako da je srednja vrednost svakog od atributa jednaka 0. U tom slučaju važi $C = D^T D$.

   Tada važi:
   - Matrica $D' = DU$ je tražena transformacija matrice podataka.
   - Novi atribut je linearna kombinacija starih atributa, a težina linearne kombinacije i-tog atributa su komponente i-tog sopstvenog vektora.
   - Varijansa novog i-tog atributa je λ_i. Zbir varijansi originalnih jednak je zbiru varijansi novih atributa.
   - Novi atributi nazivaju se **glavne komponente**. Prvi novi atribut je prva glavna komponenta, i tako dalje.

Vizuelizacija

Vizuelizacija je konverzija podataka u vizuelni ili tabelarni format. Njome dobijamo mogućnost analize vizuelnog prikaza velike količine podataka, otkrivanje opštih obrazaca, elemenata van granica i slično. Način i vrsta prikaza zavise od podataka. Ako imamo podatke koji imaju visoku dimenzionalnost, možemo eliminisati ili smanjiti naglašenost pojedinih atributa ili podskupa atributa. Najčešće se primenjuje dimenziona redukcija radi smanjenja broja dimenzija na dve ili tri ili se posmatraju parovi atributa posebno. Prvi korak vizuelizacije je **preslikavanje** objekata, atributa i njihovih odnosa u grafičke elemente kao što su tačke, linije, oblici, boje. Na primer, objekte možemo da prikažemo kao tačke, vrednosti atributa kao pozicije tih tačaka ili neke karakteristike kao što su boja, veličina ili oblik. Ako koristimo poziciju, lako možemo da uočimo odnose posmatrajući grupe, oblike i elemente van granica. Preslikavanje zavisi i od tipa tributa. Na primer, ako imamo numeričke atribute lako možemo koristiti veličinu da predstavimo vrednost atributa, dok kod imenskih to i nema mnogo smisla. Sam način smeštanja vizuelnih elemenata takođe utiče na razumevanje, jer sa jednim načinom uređenja možda ne možemo ništa da zaključimo, a sa drugim možemo da primetimo neke obrasce. Izbor tehnike vizuelizacije zavisi i od toga ko je korisnik kome prezentujemo rezultate. Neki značajni tipovi:

- **Histogram** - prikazuje raspodelu vrednosti nekog atributa po podeocima. Za kategoričke podatke svaka vrednost može biti jedan podeok, a za neprekidne atribute možemo opseg podeliti na podeoke jednake širine. Površina pravougaonika podeoka treba da bude proporcionalna broju vrednosti koje pripadaju podeoku. Ako su svi podeoci jednake širine, onda se govori o visini, umesto o površini.
- **Kućice (box plot)** - takođe prikazuje distribuciju podataka, ali koristeći percentile. Donji i gornji kraj kućice poklapa se sa 25. i 75. percentilom, a posebno se obeležavaju i 10. i 90. percentil. Elementi van granica su elementi izvan krajeva.
- **Šeme sa raspršenim elementima** - vrednosti atributa određuju poziciju. Najčešće se koriste 2D, ali postoje i 3D verzije. Oblici ili boje koriste se za predstavljanje nekih dodatnih atributa. Najčešće se koriste za analizu odnosa između dva atributa, a ako imamo i oznake klasa, možemo da zaključimo na koji način ta dva atributa razdvajaju klase.
- **Šeme sa konturama** - korisne su kada se neprekidni atributi mere na prostornoj osnovi, npr. prikaz temperature, padavina, vazdušnog pritiska, nadmorske visine i slično. Dele ravan u regione sa sličnim vrednostima. Konturne linije koje čine granice regiona povezuju tačke sa istim vrednostima.
- **Šeme sa matricama (toplotne mape)** - korisne kada su objekti sortirani u klase. Atributi se obično normalizuju da bi se izbegla dominacija jednog atributa nad drugim.
- **Paralelne koordinate** - koriste se za prikaz atributa višedimenzionalnih podataka. Koordinate ose su paralelne i odgovaraju atributima. Svaki objekat se predstavlja linijom tako što se vrednosti njegovih atributa predstavljaju tačkama na svakoj odgovarajućoj koordinatnoj osi, pri čemu se tačke povezuju linijom. Redosled osa određuje način na koji grafik izgleda. Potrebno je urediti ose tako da prikaz bude prikladniji.

- **Crtanje zvezda** - slično kao paralelne koordinate, ali se ose granaju iz jedne tačke u centru. Tačke objekata su spojene poligonalnom linijom.
- **Černofova lica** - vrednosti atributa se preslikavaju u lica, tako da je svaki atribut povezan sa nekom karakteristikom lica, npr. širina usana, veličina nosa, oblik glave, nagib obrva i slično.
- **Dijagrami Voronoi-a** - za dati skup tačaka prostor se deli u regione unutar kojih su sve tačke bliže centru regiona nego bilo kom drugom centru.
- **Slike i tabele** - uobičajeni grafikoni i tabele. Slike se mogu kombinovati, grafikoni porediti i slično.
- **Čvorovi u mreži** - prikaz međusobne povezanosti.
- **Mreža povezanosti** - prikaz međusobne povezanosti.
- **Animacije**

Klasifikacija

Ulazni podaci za klasifikaciju su slogovi oblika $S_i = (x_i^1, x_i^2, \ldots, x_i^k, y_i)$, gde je atribut y od posebnog interesa i cilj je uočiti da li se na osnovu vrednosti atributa X može odrediti vrednost atributa y. Dakle, zadatak klasifikacije je odrediti funkciju (**klasifikacioni model**) koja preslikava svaki skup atributa X u jednu od predefinisanih vrednosti y. Ako je y kategorički atribut radi se o **klasifikaciji**, a ako je numerički radi se o **regresiji**. Dobijeni model se primenjuje na podatke kod kojih nije poznat ciljni atribut y.

Ulazni podaci se obično dele u dva disjunktna dela:

- **podaci za trening** - podaci uz pomoć kojih se formira model i inicijalna tačnost modela.
- **podaci za testiranje** - podaci za proveru ispravnosti modela.

Pri proveri se upoređuju predviđene vrednosti klasa sa stvarnim vrednostima. Primeri klasifikacije su analiza ćelija tumora (da li su benigne ili maligne), provera ispravnosti korišćenja kreditnih kartica, predviđanje sekundardne strukture proteina, određivanje tipa tekstova (sport, politika, ...) i slično.

Drveta odlučivanja

Osnovna ideja ove tehnike klasifikacije je da se model formira u obliku **drveta**. Ulazni podaci se dele na osnovu vrednosti atributa i pitanja koja se nalaze u unutrašnjim čvorovima. Listovi drveta određuju oznake klasa. Unutrašnji čvorovi imaju jednu ulaznu granu, a proizvoljan broj izlaznih grana. Koren stabla nema ulazne grane. **Izazovi** ovog pristupa su brojni:

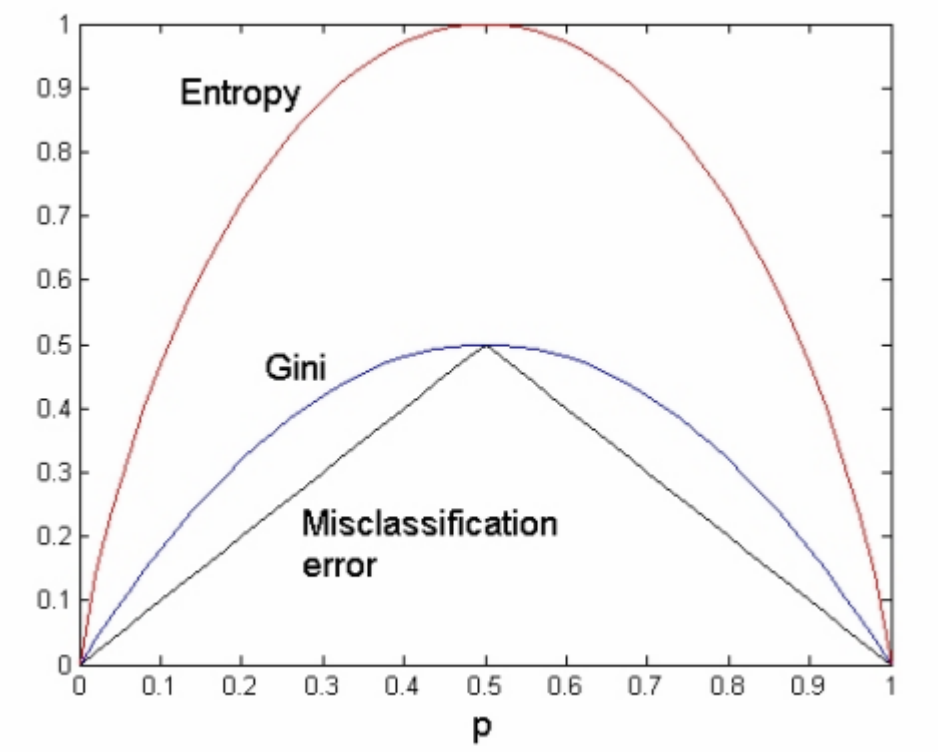
- Kako odabrati atribute po kojima se vrši podela?
- Kako formirati upit za različite tipove atributa?
- Kako odrediti najbolju podelu?
- Na koji način induktivno primeniti prethodne kriterijume na drvo po dubini?
- Kada stati sa konstrukcijom drveta?

- Kako raditi sa nedostajućim vrednostima?
- Koji je kriterijum za procenu greške u generalizaciji?
- Kako oceniti cenu i performanse modela?

Opšti model je opšta procedura za konstrukciju drveta. Neka je D_t skup slogova za trening koji se nalaze u čvoru t i neka su $y = \{y_1, \ldots, y_k\}$ oznake klasa. Opšti postupak je:

- Ako D_t sadrži samo slogove koji pripadaju istoj klasi y_t, tada je taj čvor list, označen po nazivu klase y_t.
- Ako D_t sadrži slogove koji se nalaze u više od jedne klase, tada se na osnovu test atributa vrši podela podataka u manje podskupove na koje se rekurzivno primenjuje kompletna procedura.

Atribut za podelu podataka se bira **strategijom pohlepe (greedy)**, tj. slogovi se dele prema atributu koji optimizuje neki kriterijum. Najbolja podela se određuje pomoću **mere nečistoće** - mera homogenosti distribucije klasa u čvoru. Neke od mera su:



- **Ginijev indeks**

$$Gini(t) = 1 - \sum_{i=1}^{c} p(i \mid t)^2,$$

gde je $p(j \mid t)$ relativna učestalost klase j u čvoru t i c broj klasa.

- **Entropija**

$$Entropija(t) = -\sum_{i=1}^{c} p(i \mid t) \log_2 p(i \mid t),$$

pri čemu važi $0 \cdot \log_2 0 = 0$.

- **Greška klasifikacije**

$$GK(t) = 1 - \max_i p(i \mid t)$$

Minimalna vrednost mera je 0, što označava da svi slogovi pripadaju jednoj klasi. Maksimalna vrednost označava da su slogovi ravnomerno raspoređeni u svim klasama. Maksimum za Ginijev indeks i grešku klasifikacije je $1 - \frac{1}{n_c}$, a za entropiju $\log_2 n_c$. Kao kriterijum podele, posmatra se razlika mere nečistoće u roditelj čvoru i deci čvorovima - **dobit**. Što je dobit veća, to je podela kvalitetnija. Neka je N ukupan broj slogova u roditelj čvoru, a $N(v_j)$ broj slogova u dete čvoru v_j i k broj dete čvorova. Tada su dobiti za Gini i entropiju:

$$\Delta = Gini(roditelj) - \sum_{j=1}^{k} \frac{N(v_j)}{N} Gini(v_j),\ \Delta = Entropija(roditelj) - \sum_{j=1}^{k} \frac{N(v_j)}{N} Entropija(v_j)$$

Uslovi testiranja atributa zavise od tipa atributa, kao i od toga da li delimo na dve ili na više grana.

- Binarni atributi se uvek dele na dve grane.
- Imenski atributi se mogu podeliti na više grana (onoliko koliko atribut ima različitih vrednosti), a mogu se podeliti i na dve grane i u tom slučaju treba naći optimalnu podelu.
- Redne atribute je bolje podeliti na dve grane, jer se na taj način može očuvati redosled (npr. za veličine majica, jedna grana su XS, S i M, a druga L, XL i XXL).
- Kod neprekidnih atributa, postoji više načina:
    - Diskretizacijom se formiraju redni atributi pa se koristi pristup iznad.
    - Podela na dve grane (npr. $A < v$ ili $A \geq v$). Ovaj pristup je računarski zahtevan jer se moraju razmatrati podele za svako moguće v.
    - Podela na k intervala (npr. $v_i \leq A < v_{i+1}$, $i = 1, \ldots, k$). Potrebno je razmatrati sve moguće intervale, pa je i ovaj pristup računarski zahtevan.

    Za efikasnije izračunavanje Gini indeksa, svaki atribut se sortira po vrednostima, a onda se dobijene vrednosti linearno prolaze, ažurira se broj slogova i računa Gini indeks.

Mere kao što su Ginijev indeks i entropija favorizuju atribute sa velikim brojem različitih vrednosti, kao što je npr. broj indeksa. Problem se može rešiti upotrebom isključivo binarnih podela ili uključivanjem broja mogućih vrednosti u ocenu podele:

$$\text{Odnos dobiti za podelu} = \frac{\Delta_{\text{podela}}}{split\_info},\ split\_info = -\sum_{i=1}^{k} P(v_i) \log_2 P(v_i),$$

gde je k broj različitih vrednosti atributa. Ako atribut ima mnogo različitih vrednosti i **split information** će biti veliki, pa je dobit mala.

Karakteristike drveta odlučivanja:

- Jeftina su za konstruisanje.
- Brzo klasifikuju nepoznate podatke.
- Lako se interpretiraju, ako su manje veličine.
- Preciznost se može porediti sa drugim tehnikama klasifikacije.
- Izbor mere nečistoće nema veliki uticaj na performanse.
- Primenljiva su na sve tipove podataka.
- Izražajna su, tj. mogu da predstave svaku funkciju diskretnih atributa.
- Omogućavaju rad sa nedostajućim vrednostima, kao i irelevantnim i redundantnim atributima.
- Ako je kriterijum podele jedan atribut, interval je pravougaonog oblika. Ako je kriterijum kombinacija više atributa, onda se mogu obraditi i kosi podaci.
- Način potkresivanja značajno utiče na rezultate.
- Granice podele mogu da se iskrive zbog postojanja šuma.
- Imaju problem preprilagođavanja i potprilagođavanja.

Potprilagođavanje (underfitting) podrazumeva da je model previše jednostavan i da su greške i pri treniranju i pri generalizaciji visoke. Može nastati preranim potkresivanjem drveta. **Preprilagođavanje (overfitting)** podrazumeva da model isuviše dobro klasifikuje podatke za trening i da onda može imati lošije karakteristike od modela koji ima veću grešku u procesu treninga. Može da se javi zbog nepostojanja reprezentativnih primera. Može se javiti i kod drveta koja su složenija nego što je potrebno. U tom slučaju greška pri treningu ne daje korektnu procenu ponašanja drveta na prethodno nepoznate podatke, pa je potrebno naći način za procenu greške na nepoznatnim, test podacima.

Neka je T drvo, t čvor, $e(t)$ broj pogrešno klasifikovanih slogova u t i $e(T)$ ukupan broj grešaka klasifikacije po drvetu T. Bira se onaj model koji ima najmanju grešku pri treniranju, tj. $\min\left(\sum e(t)\right)$. Greška pri generalizaciji (testiranju) će biti $\sum e'(t)$. Metode procene greške u generalizaciji su:

- **optimistički pristup**: $e'(t) = e(t)$
- **pesimistički pristup**: $e'(t) = e(t) + 0.5$ za list. Za celo drvo:

$$e'(T) = \frac{\sum_{i=1}^{k}[e(t_i) + \Omega]}{\sum_{i=1}^{k} n(t_i)} = e(T) + \frac{k\Omega}{N},$$

  gde je k broj listova, Ω cena pridružena svakom listu t, $n(t_i)$ broj trening slogova u listu t_i i N ukupan broj trening slogova. Kao što je već rečeno, složeniji model ima veće šanse za preprilagođavanje.

Okamov princip (žilet) kaže da od dva modela sa sličnom greškom generalizacije treba izabrati onaj koji je jednostavniji. Kod složenih modela veća je šansa da dođe do preprilagođavanja i zbog grešaka u podacima. Smanjenje greške se može postići potkresivanjem.

Princip najmanje dužine opisa (MDL) kaže da je najbolji model onaj koji daje najkraći ukupni opis podataka i samog modela. Uključuje složenost modela u ocenu kvaliteta. Neka je dat skup slogova sa poznatim atributom X. Osoba A poznaje sve vrednosti klasa atributa y. Osoba B nema tu informaciju i može da pošalje zahtev A da prosledi sve oznake klasa. Ovo bi zahtevalo vreme $O(n)$, gde je n broj slogova. Alternativno, A može da napravi model, kodira ga i pošalje B. Ako je model 100% tačan cena prenosa je jednaka ceni kodiranja modela, a ako nije, potrebne su i informacije o nekorektno klasifikovanim slogovima. Dakle, cena(model, podaci) = cena(model) + cena(podaci | model), gde je cena broj bitova potreban za enkodiranje. Prema MDL-u traži se model koji minimizuje ukupnu cenu.

Prepotkresivanje (pravilo ranijeg zaustavljanja) podrazumeva da se algoritam zaustavlja pre nego što drvo naraste do maksimalne veličine. Tipični uslovi zaustavljanja:

- Sve instance pripadaju istoj klasi.
- Vrednosti svih atributa su iste.
- Broj instanci je manji od neke zadate granice.
- Distribucija instanci je nezavisna od raspoloživih atributa.
- Širenje tekućeg čvora ne poboljšava meru čistoće.

Potkresivanje po završetku podrazumeva da drvo odlučivanja raste do krajnjih granica, a onda se neki čvorovi sa kraja iseku. Ako se greška generalizacije poboljša posle otsecanja, poddrvo se menja čvorom koji je list. Oznaka klase tog lista se određuje na osnovu klase većine instanci poddrveta. Za potkresivanje po završetku se može koristiti i MDL.

Uticaj **nedostajućih vrednosti** na drvo odlučivanja:

- Šta ako atribut po kome se čvor deli ima NV? Kako računati meru čistoće?
- Kako rasporediti instance sa NV na decu čvorove?
- Kako klasifikovati test instancu sa NV?

Matrica konfuzije je tabela koja prikazuje koliko je model tačno ili pogrešno klasifikovao primere svake klase. U redovima se nalaze stvarne klase, a u kolonama predviđene. Primer za binarnu klasifikaciju (klase "Da" i "Ne"):

	Da	**Ne**
Da	**TP**	**FN**
Ne	**FP**	**TN**

Najčešće korišćena metrika je **tačnost**.

$$\frac{TP + TN}{TP + FN + FP + TN}$$

Loša strana ove metrike je to što nije dobra kada podaci nisu balansirani po klasama.

Na primer, ako imamo 9990 primeraka klase "Da" i 10 primeraka klase "Ne", a model uvek predviđa klasu "Da", tačnost će biti 99.9%, ali je jasno da je model loš.

Matrica cene je slična kao matrica konfuzije, ali računa cenu pogrešne klasifikacije. Vrednost $\text{cena}(a \mid b)$ je cena pogrešne klasifikacije sloga klase b kao sloga klase a.

	Da	**Ne**
Da	**cena(Da $\mid$ Da)**	**cena(Ne $\mid$ Da)**
Ne	**cena(Da $\mid$ Ne)**	**cena(Ne $\mid$ Ne)**

Mere osetljive na cenu:

- **Preciznost**

$$p = \frac{TP}{TP + FP}$$

- **Odziv**

$$r = \frac{TP}{TP + FN}$$

- **Težinska preciznost**

$$\frac{w_1 TP + w_4 TN}{w_1 TP + w_2 FN + w_3 FP + w_4 TN}$$

- F **mera**

$$F = \frac{2rp}{r + p}$$

- F_β **mera**

$$F_\beta = (1 + \beta^2) \cdot \frac{rp}{\beta^2 p + r}$$

  F mera je specijalni slučaj F_β mere za $\beta = 1$. Za $\beta = 0$ dobija se preciznost, a za $\beta \to \infty$ dobija se odziv. Manje β daje veću težinu preciznosti, a veće β daje veću težinu odzivu.

Algoritmi drveta odlučivanja

ID3 (Iterative Dichotomiser 3)

Kao meru koristi entropiju, a kao kriterijum podele informacionu dobit. Kriterijum zaustavljanja je kada svi podaci u čvoru pripadaju istoj klasi ili kada je najveća informaciona dobit negativna. Razmatra samo jedan atribut po odlučivanju i formira binarno drvo. Ne vrši se potkresivanje drveta. Mana je to što nema mogućnost da radi sa NV i numeričkim atributima, a sklon je i preprilagođavanju.

C4.5

Predstavlja proširenje ID3 algoritma. To je skup algoritama (C4.5, C4.5 bez potkresivanja i C4.5-pravila). Ako je atribut binaran, test indukuje dve grane, a ako je kategorički, test može da ima više vrednosti, čime se pravi više grana. Dozvoljeno je grupisanje više različitih vrednosti u manji skup opcija sa jednom klasom predviđenom za svaku opciju. Ako je atribut numerički, test je opet binaran, oblika $\leq \theta$ ili $\geq \theta$, gde je θ odgovarajući prag za taj atribut.

Kao meru koristi informacionu dobit za entropiju i **odnos informacione dobiti (gain ratio)**, koja predstavlja način da se smanji favorizacija testova sa višestrukim ishodima: $\text{odnos inf. dobiti}(v) = \frac{\text{inf. dobit}(v)}{Entropija(v)}$. Pohlepno se bira test sa najboljim kriterijumima.

Za binarne i kategoričke atribute, vrednosti testa su različite moguće vrednosti tog atributa. Za numeričke atribute, prag se dobija sortiranjem po tom atributu i izborom podele između uzastopnih vrednosti tako da gorenavedeni kriterijumi budu najbolje zadovoljeni. Ne moraju da se razmatraju sve uzastopne vrednosti: za dve uzastopne vrednosti numeričkog atributa v_i i v_{i+1}, ako sve instance koje imaju vrednost tog atributa v_i ili v_{i+1} pripadaju istoj klasi, onda podela između njih nikako ne može poboljšati informacionu dobit ili odnos informacione dobiti.

Uslov zaustavljanja predstavlja situacija kada je čvor čist ili kada je broj instanci u čvoru premali, tj. ispod neke definisane granice. U drugom slučaju, list ima oznaku većinske klase.

Neka je D skup vrednosti, a el minimalni broj podataka u čvoru. Ideja je da se počne od čitavog skupa podataka i da se on rekurizvno deli na manje podskupove testiranjem određenog atributa na svakom čvoru, sve dok podskupovi ne postanu čisti. Algoritam:

```
C4.5(D, el)
begin
    T = {}
    if svi elementi D pripadaju istoj klasi ili ih ima manje od el
        then završi algoritam
    za svaki atribut x iz D
        izračunati informacionu dobit (ID) u slučaju podele po x
    formirati čvor Y u kome se za podelu koristi atribut x sa najvećom ID
    za svaki podskup Dy
        Ty = C4.5(Dy, el)
        dodati Ty kao dete čvora Y u drvetu T
    return T
end
```

Kako bi se izbeglo preprilagođavanje, koristi se naknadno potkresivanje drveta. Postoje tri načina potkresivanja:

1. **Smanjenje nivoa greške (reduced error prunning)** - koristi se potpuno indukovano drvo i poseban skup test podataka koje treba klasifikovati. Za svako podstablo u indukovanom stablu, vrši se procena da li je korisno zameniti podstablo najboljim mogućim listom. Ako tako dobijeno drvo daje manji broj grešaka, a zamenjeno podstablo ne sadrži drugo podstablom sa istim ovim svojstvom, onda se to stablo zamenjuje. Proces se nastavlja sve dok ovakvo potkresivanje dovodi do manjeg broja grešaka.
2. **Pesimistička procena greške (pesimistic prunning)** - ne zahteva poseban test skup, već se greška procenjuje na osnovu količine pogrešnih klasifikacija na trening podacima. Greška čvora se rekurzivno procenjuje na osnovu grešaka njegovih grana. Ako je N broj instanci u listu, a E broj grešaka u listu, onda je **empirijska greška lista** $\frac{E+0.5}{N}$. Za poddrvo sa L listova, $\sum E$ grešaka i $\sum N$ instanci, **greška podstabla** je $\frac{\sum E+L\cdot 0.5}{\sum N}$. Pretpostavimo da je podstablo zamenjeno svojim najboljim listom i da je J broj pogrešno klasifikovanih trening instanci u tom slučaju. Pesimističko poktresivanje zamenjuje podstablo ovim listom ako je $J+0.5$ unutar jedne standardne devijacije od $\sum E + L \cdot 0.5$.
3. **Intervali poverenja (prune based on desired confidence intervals)** - stopu greške na listovima e modelujemo kao Bernulijevu slučajnu promenljivu i za dati prag pouzdanosti CI, gornja granica e_{max} može da se odredi tako da je $e < e_{max}$ sa verovatnoćom $1 - CI$. Algoritam C4.5 koristi $CI = 0.25$. Za veliko N, e možemo modelovati i normalnom raspodelom.

Stablo možemo modelovati kao disjunktnu kombinaciju konjuktivnih **pravila**, gde svako pravilo odgovara putanji u stablu od korena do lista. Leva strana pravila su uslovi odlučivanja duž putanje, a desna strana je predviđena klasa. Karakteristika C4.5 je to što može da vrši potkresivanje na osnovu pravila izvedenih iz stabla. Za svaku klasu se prvo formiraju skupovi pravila iz nepotkresanog drveta. Za svako pravilo se vrši pretraga odozdo-naviše, da bi se ustanovilo da li se neki od prethodnika može ukloniti. Pošto je uklanjanje prethodnika slično izbacivanju čvorova u indukovanom stablu, koristi se C4.5 pesimističko potkresivanje. Za svaku klasu se bira skup pojednostavljenih pravila. Broj rezultujućih pravila je obično mnogo manji od broja listova u originalnom stablu. S obzirom da se svi prethodnici razmatraju za uklanjanje, moguće je da čvorovi blizu vrha budu potkresani, pa se rezultujuća pravila ne mogu sabiti nazad u kompaktno stablo.

Rad sa nedostajućim vrednostima:

- Šta ako atribut po kome se čvor deli ima NV? Smanjiti informacionu dobit/odnos informacione dobiti za atribut.
- Kako rasporediti instance sa NV na decu čvorove? Pridružiti instancu svakoj grani procentualno prema broju poznatih vrednosti atributa u granama.
- Kako klasifikovati test instancu sa NV? Distribuirati je u grane prema relativnoj verovatnoći dobijenoj kombinovanjem rezultata simultanog pretraživanja za sve moguće ishode testiranja. Konačna klasa je klasa sa najvećom verovatnoćom u ukupnoj distribuciji u drvetu.

Naslednici C4.5 algoritma:

- J4.8 (Java)
- **C5.0** - koristi manji skup pravila sa istom preciznošću. Poboljšanje je zasnovano na **dodatnom podsticanju (boosting)** koje podrazumeva kombinovanje različitih klasifikatora radi povećanja preciznosti. Korišćenje memorije je manje za 90%, a i preko 5 puta je brži od C4.5 algoritma. Ima niži nivo greške i manje drvo odlučivanja. Radi sa težinama atributa.

CART (Classification And Regression Trees)

U pitanju je sveobuhvatan, teorijski zasnovan algoritam. U slučaju klasifikacije, predviđa se klasa koja je kategorički atribut na osnovu neprekidnih i/ili kategirčkih atributa. U slučaju regresije, predviđa vrednost atributa koja je neprekidna na osnovu neprekidnih i/ili kategoričkih atributa. Koristi binarni rekurzivni postupak particionisanja. Počinje se od korenog čvora, podaci se dele na 2 deteta i deca se dele rekurzivno. Drvo raste do maksimalne veličine bez kriterijuma zaustavljanja, tj. dok više razdvajanja nisu moguća zbog nedostatka podataka, a zatim se potkresuje metodom **poktresivanja rezanjem troškova (cost-complexity prunning)**. Ona podrazumeva da je sledeća podela koja se potkresuje ona koja najmanje doprinosi ukupnim performansama stabla na trening podacima. CART proizvodi niz ugnježdenih potkresanih drveta, od kojih je svako kandidat da bude optimalno drvo. Najoptimalnije drvo se onda nalazi procenom predviđanja svakog drveta na nezavisnim test podacima. Dakle, za razliku od C4.5 on ne koristi neku internu meru kvaliteta.

Pravila razdvajanja (splitting rules) su uvek u formi: instanca ide levo ako važi pravilo, a desno ako ne važi. Za neprekidne atribute pravilo je oblika "atribut $X \leq C$", a za kategoričke i nominalne atribute pravilo se izražava kao pripadanje listi vrednosti. Postoje 4 kriterijuma pravila razdvajanja za klasifikaciju:

- **Gini** - za binarnu ciljnu promenljivu:

  $$G(t) = 1 - p(t)^2 - (1 - p(t))^2,$$

  gde je $p(t)$ relativna učestalost klase 1 u čvoru t. Dobit podelom čvora t na levi i desni čvor je

  $$I(t) = G(t) - qG(L) - (1 - q)G(R),$$

  gde je q procenat instanci koje idu levo.
- **Simetrični Gini** - Gini prilagođen da bude simetričan u odnosu na distribuciju klasa u poddrvetima.
- **Twoing** - bazira se na direktnom poređenju distribucije u dete-čvorovima.

  $$I(split) = \frac{1}{4} p_L p_R (\sum_{k=1}^{K} |p_{k \mid L} - p_{k \mid R}|)^2,$$

  gde su p_L i p_R procenat instanci koje idu levo/desno, a $p_{k \mid L}$ i $p_{k \mid R}$ procenat instanci

klase k koje idu u levo/desno. Faktor ispred zagrade kažnjava neuravnotežene splitove. Što je twoing veći, split je bolji, tj. skup se deli na otprilike dva jednaka dela.
- **Ordered Twoing** - tretira klase kao da su uređene i pokušava da odvoji niskorangirane ciljne klase od visokorangiranih. To radi tako što grupiše instance sa susednim klasama u uređenju.

CART omogućava da se podaci mogu modelovati takvi kakvi jesu, bez ikakvih prethodnih obrada, jer se koristi mehanizam **prethodnih verovatnoća (prethodnika)**. Prethodnici su kao težine ciljnih klasa, ali ne utiču na izračunavanja. Koriste se za određivanje kvaliteta podele. CART računa frekvencije klasa u bilo kom čvoru u odnosu na frekvenciju klasa u roditelju. Za binarnu ciljnu klasu, čvor se klasifikuje kao klasa 1 akko:

$$\frac{N_1(dete)}{N_1(roditelj)} > \frac{N_0(dete)}{N_0(roditelj)}$$

CART omogućava potpuno automatizovan i efikasan mehanizam rukovanja sa NV.

- Šta ako atribut po kome se čvor deli ima NV? Prve verzije su ocenjivale podelu na podskupu koji ne sadrži NV, a kasnije verzije uvode kazne za podele po čvoru koji sadrži NV.
- Kako rasporediti instance sa NV na decu čvorove?
- Kako klasifikovati test instancu sa NV?

Za rešavanje druga dva problema, CART za svaki čvor određuje **surogat**, tj. zamenski razdvajač, bilo da taj čvor sadrži NV ili ne. Na taj način su surogati dostupni za test podatke čak i kada u trening podacima ne postoje NV. Surogat atribut je atribut koji može da predvidi podelu koju vrši originalni spliter, pri čemu je i surogat u formi binarnog delitelja. Za svaki split surogati se rangiraju prema **oceni asocijacije** koja meri prednost surogata u odnosu na default pravilo. Da bi se neko pravilo kvalifikovalo kao surogat, mora da nadmaši default pravilo. **Default pravilo** je pravilo koje kaže da sve instance idu ka većem čvoru. Dakle, kada se pojavi instanca sa NV, ona se pomera levo ili desno u skladu sa najbolje rangiranim surogatom.

CART računa **važnost atributa** kao zbir dobiti svih čvorova u kojima se atribut koristi za podelu, pomnožen sa procentom trening podataka u tom čvoru. Surogati su takođe uključeni u procenu, pa i atributi koji se ne koriste ni za jednu podelu mogu biti ocenjeni kao važni. Na ovaj način možemo otkriti **prikrivanje atributa** (kada dominantni atribut sakrije važnost drugih korelisanih atributa) i nelinearne korelacije među atributima.

Čitav matematiči aparat koji opisuje CART postavljen je u terminima pogrešne klasifikacije. **Trošak pogrešne klasifikacije** instance klase i kao instanca klase j je $C(i,\ j)$. Vrednost je podrazumevano 1, osim za $C(i,\ i)$ gde je 0. Kompletan skup troškova predstavljen je u **matrici cene** gde imamo red i kolonu za svaku klasu. **Učenje sa osetljivošću na cene (cost-senstive learning)** može se ostvariti na dva načina. Prvi način podrazumeva da pogrešno klasifikovane instance imaju veće težine, pri čemu se ista težina koristi za instance jedne klase. To određuje koliko je "opasno" pogrešno klasifikovati instance te klase. Drugi

način je da se unosi u matrici cena sumiraju u svakom redu da bi se dobile relativne težine klasa koje približno odražavaju troškove.

CHAID (Chi-Squared Automatic Interaction Detector)

Atribut ciljne klase može biti samo kategorički, dok ostali atributi mogu biti imenski, redni, kategorički i neprekidni. Drvo klasifikacije se formira uzastopnom primenom sledećih koraka na svaki čvor, počev od korenog:

1. **Uparivanje** - vrednosti atributa se uparuju kako bi se od njih kreirale kategorije. Ako dve vrednosti imaju sličnu distribuciju ciljne klase, spajaju se u jednu kategoriju. Numerički atributi se prvo diskretizuju pre primene ovog koraka.
2. **Podela** - računa se p-vrednost u odnosu na atribut ciljne klase. Za podelu se bira atribut sa najmanjom p-vrednošću statističkog testa, tj. onaj koji najbolje objašnjava ciljni atribut. Test zavisi od tipa atributa.
3. **Zaustavljanje** - proces se završava kada nema kategorija koje se mogu spojiti.

Za rad sa nedostajućim vrednostima postoji više opcija:

- Instanca se ne koristi ako se podela vrši na osnovu atributa koji sadrži NV.
- Instanca se ignoriše ako svi atributi imaju NV.
- NV se tretiraju kao posebna kategorija.

Exhaustive CHAID je modifikacija koja proverava sve moguće podele atributa. Zahteva više vremena nego CHAID.

QUEST (Quick, Unbiased, Efficient Statistical Trees)

Podržava podele na osnovu jednog atributa, kao i linearne kombinacije podela na osnovu više atributa. Atribut klase može biti samo kategorički. Vrši binarnu podelu i koristi post-potkresivanje drveta. Za vezu između atributa klase i atributa pri podeli koristi različite testove u zavisnosti od tipa atributa.

SLIQ (Supervised Learning in Quest)

Varijanta QUEST algoritma koja se koristi za klasifikaciju velikih trening podataka. Radi efikasno sa podacima koji ne mogu da stanu u memoriju računara. Koristi se **tehnika rasta drveta prvo u širinu**. Podržava rad sa kategoričkim i numeričkim atributima, a podela se vrši na osnovu Gini indeksa. Koristi vertikalni format podataka - podaci se sortiraju i smeštaju u listu. Lista klasa čuva oznake klasa svih instanci. Za potkresivanje se vrši tehnika zasnovana na MDL.

SPRINT (Scallable PaRallelizable INduction of Trees)

Modifikacija SLIQ algoritma koja uklanja memorijska ograničenja. Podržava kategoričke i numeričke atribute. Oznake klasa pridružuju se **identifikatorima instanci**.

Dodatne metode klasifikacije

Klasifikatori zasnovani na pravilima

Slogovi se klasifikuju pomoću skupa pravila "ako ... onda ...". **Pravilo** je oblika:

$$(uslov) \rightarrow y,$$

gde je $uslov$ konjunkcija atributa, a y oznaka klase. Leva strana pravila se naziva **preduslov (uslov)**, a desna **posledica**. **Odziv pravila** je procenat broja instanci koje zadovoljavaju levu stranu pravila. **Preciznost pravila** je procenat broja instanci koje zadovoljavaju desnu stranu pravila od slogova koji zadovoljavaju levu stranu pravila. Pravilo **pokriva** (obuhvata) instancu ako vrednost atributa instance zadovoljava uslov pravila. Može se desiti da neku instancu pokriva više pravila, a da neke instance ne pokriva ni jedno pravilo. Ograničenja skupa pravila:

- **uzajamno isključiva pravila** - ne postoje dva pravila koja pokrivaju istu instancu, tj. svaka instanca je pokrivena najviše jednim pravilom. Ako nije ispunjeno, postoje dve mogućnosti:
  1. Pravilima se dodeljuju prioriteti i ona seređaju po tom prioritetu. Tako uređen skup pravila naziva se **skup odluka**. Instanca se onda klasifikuje pravilom sa najvišim prioritetom. Prioritet može biti i neka mera kvaliteta pravila. Uređenje može biti zasnovano i na klasama, tj. pravila koja pripadaju istoj klasi se grupišu zajedno.
  2. Koristimo **glasački sistem** koji podrazumeva da primenjujemo sva pravila i rezultat svakog pravila koristimo kao glas za tu klasu. Pravilima se mogu dodeliti i težine na osnovu preciznosti i odziva.
- **pravila koja pokrivaju sve mogućnosti** - postoji pravilo za sve moguće kombinacije vrednosti atributa, tj. svaka instanca je pokrivena bar jednim pravilom. Ako nije ispunjeno, onda se jedna klasa definiše kao podrazumevana i instanca se klasifikuje kao instanca te klase. Uglavnom se uzima najčešća klasa.

Ako skup pravila ima obe karakteristike, to znači da je svaka instanca pokrivena tačno jednim pravilom.

Metode za **formiranje pravila** za klasifikaciju:

- **indirektne metode** - pravila se izdvajaju iz drugih klasifikacionih modela, npr. stabla odlučivanja. Pitanja u čvorovima stabla nam služe da formiramo pravila koja vode do listova. Prednost je to što dobijamo pravila koja su međusobno isključiva i pokrivaju sve mogućnosti. Mana je to što možemo dobiti veliki broj pravila koja se samo malo razlikuju. Najpoznatiji je algoritam C4.5rules koji formira pravila na osnovu stabla odlučivanja koje daje C4.5 algoritam.
- **direktne metode** - pravila se izdvajaju iz podataka. **Sekvencijalno pokrivanje** je metoda koja počinje od praznog skupa pravila, a zatim izdvaja pravila za narednu klasu. Slogovi koji pripadaju toj klasi se posmatraju kao pozitivni, a ostali kao negativni. Koristi

se funkcija koja uči jedno pravilo i skup pravila se proširuje dobijenim pravilima. Svi pozitivni slogovi se onda uklanjaju i korak se ponavlja za sve ostale klase, dok se ne ispuni kriterijum zaustavljanja. Ako ne bismo izbacivali pozitivne slogove, uvek bismo dobijali ista pravila. Neki algoritmi:

1. **1R (One rule)** - za svaki atribut u skupu podataka formira jedno pravilo i zatim bira pravilo sa najvećom preciznošću. Radi sa diskretnim atributima, a NV tretira kao izdvojene vrednosti u skupu vrednosti atributa.
2. **CN2** - formira uređen skup pravila, prema kvalitetu. Koristi sekvencijalno pokrivanje, bez fiksiranja klase unapred. Kao posledica, pravila za različite klase su pomešana. NV zamenjuje sa najčešćim/srednjim vrednostima.
3. **RIPPER** - formira uređen skup pravila. Koristi sekvencijalno pokrivanje, sa fiksiranjem klase unapred. Na narednu klasu se prelazi tek kada se kompletira prethodna klasa. Prvo se određuju pravila za najmanje brojnu klasu.

Osobine sekvencijalnog pokrivanja:

- Porast skupa pravila - eksponencijalno mnogo pravila. Da bi se izbegla eksponencijalna eksplozija, koristi se pohlepna strategija. Pristup **opšte ka pojedinačnom** na početku formira pravilo $r : \{\} \rightarrow y$, koje pokriva sve instance i jako je lošeg kvaliteta. Dodaju se novi konjukti i povećava kvalitet, dok se ne postigne uslov zaustavljanja. Pristup **pojedinačno ka opštem** podrazumeva da se bira jedna pozitivna instanca kao inicijalno pravilo, a onda se konjukti uklanjaju tako da pravilo pokriva što više pozitivnih instanci.
- Mere kvaliteta pravila:
  - **Preciznost**: $\frac{n_+}{n}$,
  - **Laplas**: $\frac{n_+ + 1}{n + k}$,
  - **M-procena**: $\frac{n_+ + kp_+}{n + k}$,
    gde su n broj instanci pokrivenih pravilom, n_+ broj pozitivnih instanci pokrivenih pravilom, k ukupan broj klasa i p_+ prethodna verovatnoća za pozitivnu klasu.
- Uprošćavanje pravila
- Kriterijum zaustavljanja - računamo dobit i ako nije značajna, novo pravilo se odbacuje.
- Potkresivanje pravila - slično potkresivanju drveta odlučivanja. Izbacujemo jedan konjukt i upoređujemo novo pravilo sa starim. Ako je kvalitet bolji, izbacujemo taj konjukt.

Klasifikatori zasnovani na pravilima imaju istu izražajnu moć kao i drveta odlučivanja, a i performanse su slične. Jednostavno se formiraju i interpretiraju.

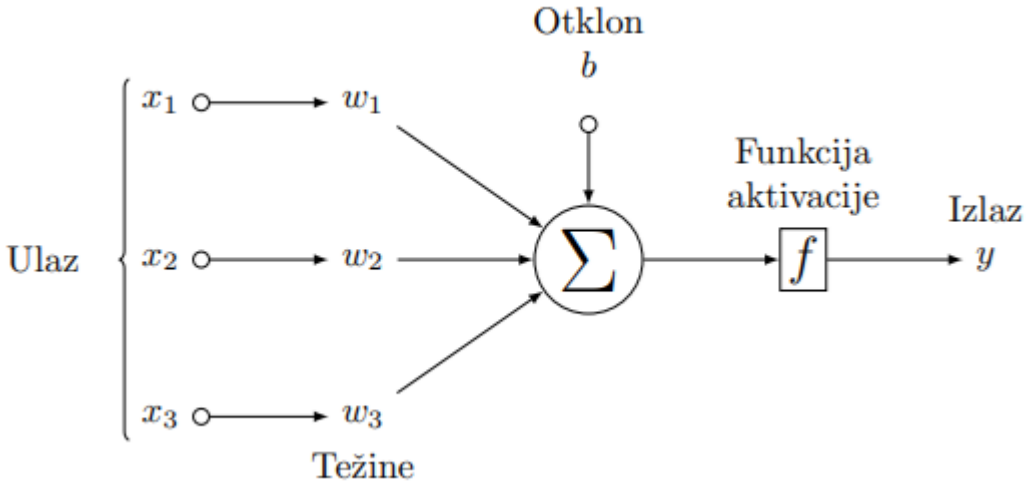
Klasifikatori zasnovani na instancama

Ne postoji model u klasičnom smislu, već model čine svi trening slogovi koji se čuvaju i ponovo koriste u obradi svaki put kada se klasifikuje nepoznata instanca. Ako je trening skup jako veliki, brzina klasifikacije je manja. Kaže se da je klasifikator **lenj** ako procesiranje trening podataka odlažemo sve do klasifikacije test podataka. Primeri:

- **učenje napamet** - sprovodi klasifikaciju samo ako se atributi test slogova potpuno poklope sa atributima trening slogova.
- **najbliži susedi** - koristi k najbližih tačaka za klasifikaciju. Osnovna ideja je "s kim si, takav si". Potreban nam je skup sačuvanih slogova, metrika za računanje rastojanja i vrednost k koja predstavlja broj najbližih suseda koje treba razmatrati. Računamo rastojanja do ostalih slogova, odredimo k najbližih i koristeći njihove oznake, odredimo klasu test instance. Ako je k jako malo, klasifikacija je osetljiva na šum, a ako je jako veliko, susedi mogu da uključe tačke iz drugih klasa. Ako je $k = 1$, tada se podela prostora za klasifikaciju predstavlja preko dijagrama Voronoi-a. Za metriku se može koristiti Euklidsko rastojanje, mada ono nije uvek pogodno (npr. kod asimetričnih atributa). Atributi se mogu i skalirati da bi se sprečilo da jedan atribut dominira nad drugim. Kod biranja klasa, može se birati klasa koja je većinska kod suseda, a rastojanjima se mogu pridružiti i težine.

Veštačke neuronske mreže

Veštačke neuronske mreže simuliraju rad bioloških nervnih sistema. Čine ih čvorovi i veze između njih. **Perceptron** je model neurona. Predstavlja parametrizovanu funkciju koja računa linearnu kombinaciju svojih argumenata. Postoje **ulazni** i **izlazni** čvorovi koji su povezani **vezama sa težinama**. Težine simuliraju jačinu sinaptičke veze kod bioloških neurona. **Treniranje** perceptrona podrazumeva promenu vrednosti težina i traje dok se ne sinhronizuju ulazno-izlazne zavisnosti podataka.



Perceptron računa izlaznu vrednost $\overline{y}$ kao težinsku sumu ulaznih vrednosti uz oduzimanje **otklona (bias)** i proverava znak rezultata. Matematički, izlaz je

$$\overline{y} = sign(w \cdot x - b) = sign(w_d x_d + w_{d-1} x_{d-1} + \ldots + w_1 x_1 - b)$$

Neka je $D = \{(x_i,\ y_i) \mid i = 1,\ \ldots,\ n\}$ trening skup. Algoritam za obučavanje perceptrona inicijalizuje težine slučajnim vrednostima $w^{(0)}$. Dok se ne dostigne kriterijum zaustavljanja, za svaki trening primer $(x_i,\ y_i) \in D$ računa se predviđeni izlaz $\overline{y_i^{(k)}}$, a zatim se svaka težina w_j ažurira: $w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \overline{y_i^{(k)}})x_{ij}$. Vrednost $w^{(k)}$ je težina pridružena i-toj ulaznoj grani posle k-te iteracije, $\lambda \in [0,\ 1]$ **brzina učenja** i x_{ij} vrednost j-tog atributa u i-tom trening

primeru. Nova vrednost težine je kombinacija stare vrednosti i vrednosti proporcionalne grešci predviđanja. Ako je $y = 1$ i $\overline{y} = -1$, tada je greška 2 i da bi se otklonila povećava se vrednost predviđenog izlaza povećanjem težina u svim vezama sa pozitivnim i smanjenjem težina u vezama sa negativnim ulazom. Ako je $y = -1$ i $\overline{y} = 1$, tada je greška -2 i da bi se otklonila smanjuje se vrednost predviđenog izlaza smanjenjem težina u svim vezama sa pozitivnim i povećanjem težina u vezama sa negativnim ulazom. Ako je brzina učenja blizu nule, nove vrednosti težina najviše zavise od starih vrednosti težina, a ako je blizu jedinice, nove vrednosti težina su osetljive na podešavanja u tekućoj iteraciji. Prethodni model perceptora je linearan po parametrima w i atributima x. Vrednost $\overline{y}$ = 0 predstavlja linearnu hiperravan koja razdvaja podatke u dve klase. Algoritam obučavanja perceptrona garantuje konvergenciju ka optimalnom rešenju za linearno razdvojive klasifikacione probleme.

VNM sa više slojeva podrazumevaju da imamo jedan ulazni sloj, jedan izlazni sloj i nula ili više skrivenih slojeva. Mogu biti VNM sa propagacijom unapred, propagacijom unazad, rekurentne, RBF, samoorganizujuće mape, asocijativne itd. Mogu da se koriste različite **funkcije aktivacije** (linearna, sigmoidna, signum, hiperbolički tangens, ...). Cilj obučavanja VNM je odrediti skup težina w koji minimizuje kvadratnu grešku:

$$E(w) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \overline{y_i})^2$$

Zamenom $\overline{y} = w \cdot x$ vidimo da je funkcija greške kvadratna, pa se može odrediti globalni minimum. Ako funkcija aktivacije nije linearna, izlaz iz VNM će biti nelinearna funkcija parametara, za koju ne možemo naći globalni minimum. U tom slučaju, za rešavanje optimizacionog problema koristi se **metoda gradijentnog spusta**. Ona ne garantuje nalaženje globalnog minimuma, tj. može vratiti neki lokalni minimum.

Support Vector Machine (SVM)

Metod potpornih vektora (SVM) zasniva se na statističkoj teoriji učenja. Tehnika je zasnovana na ideji vektorskih prostora. Primenljiva je i na klasifikaciju i na regresiju, a pogotovo kada je broj dimenzija podataka veliki. Model u ovom slučaju je **formula**, a klasa se izračunava na osnovu nje. Upotrebljiva je za numeričke podatke, a kategorički atributi se transformišu uvođenjem promenljive za svaku vrednost kategoričkog atributa. Ideja je u vektorskom prostoru, u kome su podaci, naći razdvajajuću hiperravan tako da su svi podaci iz jedne klase sa iste strane ravni. SVM određuje optimalno rešenje koje maksimizuje razdaljinu (**marginu**) između hiperravni i tačaka koje su blizu potencijalne linije razdvajanja. Kao rezultat, razdvajajuća hiper-ravan je potpuno određena specifičnim podskupom trening podataka, koji se zovu **potporni vektori**. Pretpostavlja se da su podaci linearno razdvojivi. Model zapravo predstavlja jednačinu hiperravni, a na osnovu rastojanja od hiperravni određujemo klasu.

Neka imamo binarni klasifikacioni problem sa N primera za trening. Svaka instanca je predstavljena torkom $(x_i,\ y_i)$, gde su x_i atributi, a y_i oznaka klase. Jednačina hiperravni je $w \cdot x + b = 0$. Za tačke x_a i x_b koje joj pripadaju važi $w \cdot x_a + b = 0$ i $w \cdot x_b + b = 0$, tj.

$w \cdot (x_b - x_a) = 0$. Oznaku klase za tačku z određujemo na sledeći način:

$$y = \begin{cases} 1, \ w \cdot z + b > 0 \\ -1, \ w \cdot z + b < 0 \end{cases}$$

Jednačine granica margina su $h_{i1} : w \cdot x + b = 1$ i $h_{i2} : w \cdot x + b = -1$. Margina d se dobija kao $w \cdot (x_1 - x_2) = 2$, odnosno $d = \frac{2}{||w||}$. Parametri w i b moraju da zadovolje $w \cdot x_i + b \geq 1$ ako je $y_i = 1$, odnosno $w \cdot x_i + b \leq -1$ ako je $y_i = -1$. Obe nejednakosti se mogu zapisati kao $y_i(w \cdot x_i + b) \geq 1$. Zahtev da margina bude maksimalna je sad ekvivalentan:

$$\max_w \frac{2}{||w||}, \ y_i(w \cdot x_i + b) \geq 1, \ i = 1, \ 2, \ \ldots, \ N$$

Odnosno:

$$\min_w \frac{||w||^2}{2}, \ y_i(w \cdot x_i + b) \geq 1, \ i = 1, \ 2, \ \ldots, \ N$$

U pitanju je kvadratni optimizacioni problem sa linearnim ograničenjima, koji ima jedinstveni minimum. Za njegovo izračunavanje koriste se **Lagranžovi množioci**. Prednost ovog pristupa je to što će ograničenje biti zamenjeno ograničenjima Lagranžovih množilaca, što je lakše za rad. Takođe, u novoj formi trening podaci se javljaju jedino u obliku skalarnog proizvoda, što omogućava generalizaciju rešenja na nelinearno razdvojive podatke. Neka je $f(x_1, \ \ldots, \ x_n) : R^n \to R$ diferencijabilna funkcija čiji se minimum traži, uz ograničenje $g(x_1, \ \ldots, \ x_n) = 0$. Gradijent funkcije se menja na isti način kao i gradijent ograničenja, pa važi $\nabla f(x) = \lambda \nabla g(x)$, pri čemu je λ **koeficijent promene (Lagranžov množilac)**. Kombinacijom jednačine i uslova formira se **Lagranžijan**:

$$L(x, \ \lambda) = f(x) - \lambda g(x)$$

Vrednost λ se određuje tako da je $\nabla L(x, \ \lambda) = 0$, pa se zamenom vrednosti λ i x dobija minimum funkcije. Funkcija koja se minimuzuje zapisuje se u obliku Lagranžijana:

$$L_p = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \lambda_i(y_i(w \cdot x_i + b) - 1) = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \lambda_i y_i(w \cdot x_i + b) + \sum_{i=1}^{N} \lambda_i \ (*)$$

Svi λ_i su pozitivni jer su sva ograničenja uvek ≥ 0. Za minimizaciju Lagranžijana treba odrediti:

$$\frac{dL_p}{dw} = 0 \ \Rightarrow \ w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

$$\frac{dL_p}{db} = 0 \ \Rightarrow \ 0 = \sum_{i=1}^{N} \lambda_i y_i$$

Ako su ograničenja jednakosti, direktno se dobijaju w, b i λ. Kako su ograničenja nejednakosti, a važi $\lambda_i \geq 0$, moguća je transformacija u ograničenja sa jednakostima. **Karush-Kuhn-Tucker uslovi**:

1. $\frac{dL_p}{dw} = w - \sum_{i=1}^{N} \lambda_i y_i x_i$
2. $\frac{dL_p}{db} = -\sum_{i=1}^{N} \lambda_i y_i$
3. $y_i(w \cdot x + b) - 1 \geq 0$
4. $\lambda_i \geq 0$
5. $\lambda_i(y_i(w \cdot x + b) - 1) = 0$

Važi $\lambda_i = 0$, osim ako je $y_i(w \cdot x + b) = 1$. Instance kod kojih je $\lambda_i > 0$ su potporni vektori. Ograničenja funkcije se zamenjuju ograničenjima Lagranžovih množilaca. Zamenom $w = \sum_{i=1}^{N} \lambda_i y_i x_i$ i $0 = \sum_{i=1}^{N} \lambda_i y_i$ umesto b u $(*)$ dobija se **dualni problem**:

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i \cdot x_j$$

Vrednosti λ_i se određuju uzimanjem izvoda po λ i izjednačavanjem sa nulom: $\sum_{i=1}^{N} \lambda_i y_i = 0$. Dobija se $w = \sum_{i=1}^{N} \lambda_i y_i x_i$ i $b = avg_{i,\, \lambda_i > 0} b_i$. Za nepoznatu tačku z, klasa se određuje na osnovu znaka:

$$f(z) = sign(w \cdot z + b) = sign((\sum_{i=1}^{N} \lambda_i y_i x_i \cdot z) + b)$$

Ako skup za treniranje nije linearno razdvojiv, uvodimo veličine koje mogu da olabave granice margine, tako da se tolerišu male greške pri klasifikaciji. Sada mora da važi $w \cdot x_i + b \geq 1 - \xi_i$ ako $y_i = 1$, odnosno $w \cdot x_i + b \leq -1 + \xi_i$ ako $y_i = -1$. Primenjuje se isti postupak kao i ranije. Ograničenje veličine ξ ograničava propusnost granice. Funkcija koja treba da se minimizuje je:

$$f(w) = \frac{||w||^2}{2} + C \sum_{i=1}^{N} \xi_i^k,$$

gde su C i k konstante koje određuju cenu pogrešne klasifikacije. Konstanta C se naziva **konstanta regularizacije** i kontroliše odnos između maksimizacije margine i minimizacije gubitka. Drugi sabirak predstavlja gubitak, tj. procenu devijacije od slučaja sa razdvojivim podacima. Kada $C \to 0$ gubitak nestaje i margina se maksimizuje, a kada $C \to \infty$ margina nestaje i potrebno je minimizovati gubitak. Konstanta k pokriva oblik gubitka i najčešće se postavlja na 1 ili 2. Ako je $k = 1$ (**hinge loss**) minimizuje se zbir ξ_i, a ako je $k = 2$ (**quadratic loss**) minimizuje se zbir kvadrata ξ_i. Funkcija koja se minimizuje je:

$$L_p = \frac{1}{2}||w||^2 + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \lambda_i (y_i(w \cdot x_i + b) - 1 + \xi_i) - \sum_{i=1}^{N} \beta_i \xi_i$$

Ograničenja sa nejednakostima mogu da se transformišu u ograničenja sa jednakostima koristeći KKT uslove:

1. $\xi_i \geq 0,\ \lambda_i \geq 0,\ \beta_i \geq 0,\ \forall\, i$
2. $\lambda_i(y_i(w \cdot x_i + b) - 1 + \xi_i) = 0$
3. $\beta_i \xi_i = 0$

Dualni Lagranžijan se dobija izjednačavanjem prvog izvoda po w, b i ξ sa 0:

$$\frac{dL_p}{dw} = 0 \;\Rightarrow\; w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

$$\frac{dL_p}{db} = 0 \;\Rightarrow\; 0 = \sum_{i=1}^{N} \lambda_i y_i$$

$$\frac{dL_p}{d\xi_i} = 0 \;\Rightarrow\; \lambda_i + \beta_i = C$$

Zamenom vrednosti u L_p dobijamo da je ciljna funkcija:

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i \cdot x_j$$

Tražimo njen maksimum po λ_i uz ograničenja $0 \leq \lambda_i C$, $\forall\, i$ i $\sum_{i=1}^{N} \lambda_i y_i = 0$.

Ideja **nelinearnog SVM** je odrediti funkciju $\Phi : R^m \to R^n$ tako da hiperravan $w \cdot \Phi(x) + b = 0$ razdvaja transformisane podatke. Odnosno, projektujemo podatke u višedimenzionalni prostor kako bi oni postali razdvojivi. Analogno prethodnom postupku problem se svodi na dualni problem. Međutim, nejasno je kako odrediti funkciju Φ. **Kerneli** podrazumevaju da se sličnost u transformisanom prostoru računa koristeći originalni skup atributa. Na ovaj način, ne moramo eksplicitno trenirati n-dimenzioni prostor. **Kernel funkcija** K je funkcija sličnosti koja se računa pomoću originalnog skupa atributa. **Mercerova teorema**: Kernel funkcija K može se predstaviti kao $K(x,\, y) = \Phi(x) \circ \Phi(y)$ akko važi:

$$(\forall\, g(x)) \int g(x)^2 dx \text{ je konačan } \Rightarrow \int K(x,\, y) g(x) g(y) dx dy \geq 0$$

Primeri kernela:

- **linearni kernel**: $K(x,\, y) = x^T \cdot y$
- **polinomijalni kernel**: $K(x,\, y) = (x^T \cdot y + r)^d$
- **Gausov RBF kernel**: $K(x,\, y) = e^{-\gamma ||x-y||^2}$, $\gamma = \frac{1}{2\sigma^2}$
- **eksponencijalni RBF kernel**: $K(x,\, y) = e^{-\frac{||x-y||}{2\sigma^2}}$

Klasifikacija metodom ansambla

Osnovna ideja je da ne postoji algoritam koji se najbolje ponaša u svim mogućim situacijama, pa se koristi kombinovanje skupa modela koji rešavaju isti problem. Cilj je dobijanje boljeg globalnog modela. Dobija se veća preciznost i pouzdanost u odnosu na svaki pojedinačni model. **Kondorset teorema žirija**: neka grupa ljudi nezavisno jedan od drugog bira između dve mogućnosti od kojih je samo jedna ispravna, i neka je p verovatnoća da su izabrali ispravnu mogućnost. Njihovi glasovi se kombinuju po pravilu većine i neka M označava verovatnoću da je većina napravila korektan izbor. Ako je $p > 0.5$ tada $M \to 1$ ako broj glasanja teži beskonačnosti. Kao posledica, važi da je gomila pametnija od pojedinca,

pod relativno slabim uslovima da svaki pojedinac donosi odluku nezavisno i da ispravno sudi sa verovatnoćom većom od 0.5.

Jaki klasifikatori su klasifikatori čija je greška klasifikacije proizvoljno mala. **Slabi klasifikatori** su klasifikatori koji su nešto bolji od običnog slučajnog nagađanja. Ideja je da se umesto korišćenja jednog jakog klasifikatora, formira veliki skup slabih klasifikatora čiji se izlazi kombinuju u jedno finalno rešenje. Prema Kondorset teoremi, greška će biti proizvoljno blizu nule, ako se obezbede odgovarajući uslovi. Takođe, lakše je napraviti više slabijih, nego jedan jak klasifikator. Metode za konstrukciju ansambla:

- **Promena skupa za trening** - formira se više skupova za trening na osnovu početnog trening skupa. Distribucija i izbor elemenata može da se menja pri svakom izboru. Klasifikator se formira primenom (istog) algoritma klasifikacije na svaki od skupova za trening. Primer su algoritmi sa dodatnim pojačavanjem (**boosting**) i pakovanjem (**bagging**).
- **Promena skupa ulaznih atributa** - za svaki skup podataka za trening bira se podskup ulaznog skupa atributa. Radi dobro kada ulazni skup sadrži redundantne podatke. Primer je nasumična šuma (**random forest**).
- **Promena skupa oznaka klasa** - trening skup se transformiše u binarni problem slučajnim grupisanjem klasa u dva disjunktna skupa. Uzastopnim grupisanjem postiže se efekat ansambla. Ako klasifikator predvidi klasu, tada sve klase u grupi kojoj pripada dobijaju jedan glas. Klasa koja dobije najviše glasova se dodeljuje test primeru. Koristi se kada je skup klasa dovoljno veliki.
- **Promena algoritma za klasifikaciju** - različiti algoritmi primenjeni nad istim podacima daju različite modele.

Pakovanje (bagging) je tehnika koja formira podatke uzastopnim uzorkovanjem (sa ponavljanjem) podataka iz početnog skupa u skladu sa uniformnom distribucijom verovatnoća. Svaki od formiranih inicijalnih skupova ima istu kardinalnost kao i originalni skup. Zbog ponavljanja, neki slogovi mogu da se jave više puta, a neki se možda ne pojavljuju. Za veliko N, uzorkovani skupovi u proseku sadrže 63% originalnog skupa, jer je svaki uzorak biran sa verovatnoćom $1 - (1 - \frac{1}{N})^N$, što konvergira ka $1 - \frac{1}{e}$. Na svakom od skupova treniramo osnovni klasifikator, a krajnja klasa je klasa koja dobije najveći broj glasova.

Pojačavanje (boosting) je tehnika adaptivne promene distribucije trening podataka u zavisnosti od prethodnih grešaka klasifikacije. Inicijalno, svakom od N slogova se dodeli jednaka težina. Težina se menja na kraju svakog ciklusa, tako što se težina slogova koji su pogrešno klasifikovani povećava, a težina tačno klasifikovanih slogova se smanjuje. Finalni klasifikator kombinuje glasove svih klasifikatora u ciklusu. Jedan od najpopularnijih algoritama je AdaBoost.

Nasumična šuma (random forest) je metoda za ansambl drveta odlučivanja. Konstruiše se više nepotkresanih drveta odlučivanja. Svaki osnovni klasifikator konstruiše novi podskup atributa iz originalnih podataka. Konačan rezultat se dobija glasanjem. Pri biranju atributa za

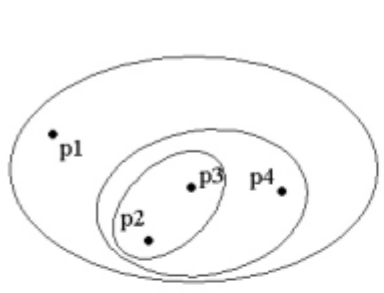
podelu, ne razmatraju se svi atributi, već samo neki. Slučajni vektor atributa može da se formira na više načina:

- Slučajno se bira F atributa za podelu u svakom čvoru drveta. Primer je ForestRI (Random Input).
- U svakom čvoru formira se novi atribut na osnovu slučajno izabranih L atributa. On predstavlja linearnu kombinaciju izabranih atributa sa koeficijentima generisanim uniformnom raspodelom u intervalu [-1, 1]. U svakom čvoru se generiše F takvih slučajno kombinovanih atributa, od kojih se najbolji bira za podelu u čvoru. Primer je ForestRC.
- U svakom čvoru najbolja podela se dobija slučajnim izborom između F najboljih atributa umesto između svih atributa.

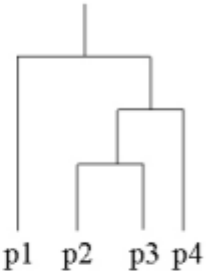
Klaster analiza

Cilj je izvršiti podelu skupa objekata $X = \{x_1, \ldots, x_n\}$ na grupe (podskupove) tako da je objekat x_i koji pripada grupi G_i sličniji po nekom kriterijumu objektima x_j koji pripadaju istoj grupi, nego nekom objektu x_k koji pripada nekoj drugoj grupi G_m. Svaka grupa G naziva se **klaster**, a celokupan postupak naziva se **klasterovanje**. Izbor metode klasterovanja zavisi od samih podataka. Broj klastera zavisi od posmatranog kriterijuma. Neki algoritmi zahtevaju da se unapred zada traženi broj klastera, a neki algoritmi sami određuju broj klastera. U zavisnosti od karakteristika klasterovanja, klasterovanje može biti:

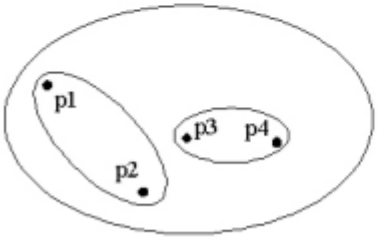
- **Particiono** - podaci se dele u disjunktne klastere, tako da svaki podatak pripada tačnom jednom klasteru.
- **Hijerarhijsko** - element može da pripada više klastera, na različitim nivoima hijerarhije. Prikaz hijerarhije klastera se naziva **dendrogram**. Na x-osi se prikazuju instance, a na y-osi se prikazuje mera sličnosti klastera.



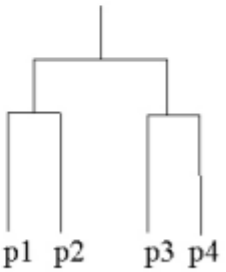
Tradicionalno hijerarhijsko klasterovanje



Tradicionalni dendogram



Netradicionalno hijerarhijsko klasterovanje



Netradicionalni dendogram

Netradicionalni pristup je poželjniji, jer imamo jasniju sliku kako se instance grupišu u klastere, dok u tradicionalnom pristupu dodajemo po jednu instancu u svakom koraku.

Tipovi klasterovanja na osnovu drugih kriterijuma:

- **ekskluzivno i neekskluzivno** - jedna instanca pripada samo jednom klasteru (ekskluzivno) ili više klastera (neekskluzivno).
- **rasplinuto i nerasplinuto** - element svakom klasteru pripada sa stepenom između 0 ili 1, pri čemu je zbir težina elementa u svim klasterima 1 (rasplinuto).
- **delimično i kompletno** - da li se klasteruje deo podataka ili kompletan skup.
- **heterogeno i homogeno** - klasteri su različite veličine, oblika i/ili gustine (heterogeno).

Tipovi klastera:

- **Dobro razdvojeni klasteri** - elementi koji im pripadaju su bliži bilo kom drugom elementu iz tog klastera, nego ostalim elementima koji nisu u klasteru.
- **Klasteri zasnovani na centru** - elementi klastera su bliži prototipu (centru) klastera u odnosu na prototipove ostalih klastera. Centar klastera je **centroid** (prosek svih tačaka u klasteru) ili **medoid** (najreprezentativnija tačka u klasteru).
- **Klasteri zasnovani na grafovima** - instance su predstavljene čvorovima, a čvorovi su povezani ako su u istom klasteru. Neke definicije dopuštaju da između klastera, tj. podgrafova, postoje veze ali u mnogo manjem broju ili sa mnogo većim rastojanjem nego između elemenata jednog podgrafa.
- **Klasteri zasnovani na susedstvu** - zasnovani su na grafovima, ali dva elementa pripadaju istom klasteru akko su na rastojanju koje je manje od unapred definisanog praga. Posledica je da za svaki element koji pripada ovom tipu klastera postoji element iz istog klastera kome je on bliži nego bilo kom elementu koji pripada drugom klasteru.
- **Klasteri zasnovani na gustini** - klasteri su oblasti sa velikom gustinom tačaka, razdvojene oblastima sa malom gustinom tačaka. Koristi se kada su klasteri nepravilni ili isprepleteni, a postoje i šum i elementi van granica.
- **Konceptualni klasteri** - klasteri su skup elementa koji imaju istu zajedničku karakteristiku (npr. oblik).

Algoritmi zasnovani na reprezentativnim predstavnicima

Za skup od n tačaka X_1, ..., X_n u d-dimenzionalnom prostoru cilj je pronaći k reprezentativnih tačaka Y_1, ..., Y_k, gde k minimizuje ciljnu funkciju:

$$O = \sum_{i=1}^{n} \min_{j} Dist(X_i,\ Y_j),$$

gde je $Dist(A,\ B)$ neka funkcija rastojanja. Imamo skup podataka D i željeni broj predstavnika k. Algoritam:

```
klasterovanje_sa_repr_preds(D, k)
begin
    inicijalni izbor skupa repr. preds. S = {Y1, ..., Yk};
    repeat
        formiraj klastere (C1, ..., Ck) dodelom svake tačke iz D najbližem
            predstavniku iz S koristeći funkciju rastojanja Dist(xi, Yj);
        ponovo formiraj S određivanjem novog predstavnika Yj za svaki Cj
            koji minimizuje prethodnu funkciju O;
    until došlo je do konvergencije
    return (C1, ..., Ck);
end
```

Algoritam k-sredina

Algoritam k-sredina je jedan od najznačajnijih predstavnika ovih algoritama. Prototip se definiše kao centroid. Mana algoritma je što zahteva da se unapred navede broj k. Inicijalno svaka tačka se dodeljuje klasteru sa najbližim centroidom. U narednom koraku se vrši izračunavanje novih centroida i ponovno računanje pripadnosti svake tačke klasteru. Za meru rastojanja može se koristiti Euklidsko rastojanje, a i neke druge mere. Uslov zaustavljanja mogu biti situacije kada broj tačaka koje promene klaster bude manji od nekog praga ili ako je promena funkcije koja se minimizuje u dve uzastopne iteracije manja od nekog praga. Jedan od nedostataka je biranje početnih centroida na slučajan način. Pametnije bi bilo da prvog predstavnika izaberemo slučajno, a onda svaki sledeći tako da je udaljen od svih ostalih (**kmeans++**). Može se koristiti i hijerarhijsko klasterovanje da se izaberu početni centroidi. Može se izabrati i veći broj ($> k$) centroida i da se od njih izabere k najboljih.

Vremenska složenost algoritma je $O(nKld)$, gde je n broj tačaka, K broj klastera, l broj iteracija i d broj atributa. Prostorna složenost je $O((n + K)d)$. Jasno je da je algoritam neefikasan ako imamo jako veliki broj tačaka.

Postoji više načina za ocenu kvaliteta klasterovanja. Za podatke u Euklidskom prostoru najčešće se koristi **zbir kvadrata grešaka (sum of squared erros - SSE)**.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i,\ x)^2,$$

gde je x tačka u klasteru C_i, a c_i centroida klastera C_i.

Za klasterovanje dokumenata, podatke čuvamo preko matrice termova. Za dokumente se kao mera koristi kosinusno rastojanje. Stepen sličnosti dokumenata u klasteru sa centroidom naziva se **kohezija klastera**. Ukupna kohezija računa se kao:

$$\sum_{i=1}^{K} \sum_{x \in C_i} \cos(c_i,\ x)$$

Suboptimalno klasterovanje je situacija gde je izvršeno klasterovanje, ali nije dobijeno globalno najbolje rešenje, koje minimizuje SSE, već neki lokalni optimum. Najvažniji je izbor početnih predstavnika. Ako postoji k realnih klastera, verovatnoća da se izabere po jedan centroid u svakom od njih je relativno mala. Jednaka je količniku broja načina za izbor centroida u svakom klasteru i broja načina za izbor k centroida. Jedna mogućnost je da algoritam pokrenemo više puta i da onda izaberemo najbolje rešenje.

Elementi van granica mogu znatno da utiču na rezultate klasterovanja. Problem se može rešiti uklanjanjem elemenata van granica, ali često ipak želimo da te elemente zadržimo. Druga opcija je neka vrsta **postprocesiranja**:

- **Eliminacija malih klastera** sa elementima van granica.
- **Podela klastera** sa visokim SSE.
- **Spajanje klastera** koji su "blizu" i imaju relativno mali SSE.

Prazan klaster je klaster u kome se nalazi samo centroid, bez ijednog elementa. Treba izbaciti prazan klaster i umesto njega formirati neki novi (kako bi ostalo k klastera). Moguća rešenja:

- Izabrati tačku koja najviše učestvuje u SSE i proglasiti je za novi centroid.
- Izabrati tačku koja je najdalje od tekućih centroioda.
- Izabrati tačku iz klastera sa najvećim SSE.

Algoritam bisekcije k-sredina je varijanta algoritma k-sredina koja može da proizvede particiono ili hijerarhijsko klasterovanje. Ideja je da se skup svih tačaka podeli u dva klastera, onda izabere jedan od njih za narednu podelu i postupak ponavlja dok ne dobijemo k klastera. Možemo podeliti najveći klaster, klaster sa najvećim SSE ili koristiti kombinaciju ova dva kriterijuma. Često se centroidi dobijeni ovim algoritmom koriste kao početni centroidi za algoritam k-sredina.

Nedostaci algoritma:

- Funkcioniše samo za klastere globularnog oblika.
- Funkcioniše samo za klastere istih gustina.
- Osetljiv je na elemente van granica.
- Biranje inicijalnih predstavnika i broja k nije jednostavno.

Prednosti algoritma:

- Dobro funkcioniše sa globularnim podacima.
- Jednostavno se implementira i primenjuje.
- Može prepoznati klastere različitih gustina, ako se koristi Mahalanobisovo rastojanje.

Jedan način da se prevaziđu ograničenja jeste da se poveća broj k, čime dobijamo manje klastere. Na taj način nećemo mešati instance iz prirodnih klastera, ali će one biti podeljene u par manjih klastera.

Algoritam k-medijana

Sličan je kao algoritam k-sredina, ali kao centroid koristi medijanu. Za rastojanje koristi **taksi blok rastojanje**. Manje je osetljiv na elemente van granica.

Algoritam k-medoida

Podrazumeva da se izbor centroida uvek vrši iz inicijalnog skupa tačaka. Koristan je za složenije tipove podataka, kao što su vremenske serije. I ovde se koristi taksi blok rastojanje. Pri računanju novih prototipova, gleda se da li postoji instanca iz klastera tako da zamena prototipa tom instancom smanjuje ciljnu funkciju.

Hijerarhijsko klasterovanje

Osnovna ideja je formiranje skupa ugnježdenih klastera koji su organizovani u obliku hijerarhije po nivoima. Rezultati se vizuelizuju pomoću dendrograma ili dijagrama sa ugnježdenim klasterima, gde vidimo redosled formiranja klastera. Karakteristika ovih algoritama je da ne zadajemo broj klastera, već se klasterovanje vrši do kraja, a onda gledamo koji broj klastera nam najviše odgovara. Postoje dve grupe algoritama:

- **algoritmi sakupljajućeg klasterovanja** - hijerarhija se formira odozdo-naviše. Na početku je svaka tačka zaseban klaster, a klasteri se onda spajaju sve dok ne dobijemo jedan klaster. Neke implementacije dozvoljavaju **presecanje** - ranije zaustavljanje algoritma ako se dođe do k klastera ili do l-tog nivoa hijerarhije. Glavna razlika između pojedinačnih algoritama je u računanju sličnosti dva klastera, od čega će zavisiti i izgled dobijene hijerarhije.
- **algoritmi razdvajajućeg klasterovanja** - hijerarhija se formira odozgo-naniže. Na početku imamo jedan klaster u kom su sve tačke, a u svakom koraku se neki klaster deli na dva klastera, sve dok ne dobijemo da je svaka tačka u jednom klasteru. Primer je algoritam bisekcije k-sredina.

Sakupljajuće klasterovanje

Neka su nam na početku poznati skup podataka D i **matrica sličnosti (rastojanja)** M, dimenzije $n \times n$. Algoritam sakupljajućeg klasterovanja:

```
sakupljajuće_klasterovanje(D)
begin
    inicijalizacija matrice rastojanja M na osnovu podataka D;
    repeat
        uzeti najbliži par klastera i i j koristeći M;
```

```
        kombinovati klastere i i j;
        obrisati redove i kolone klastera i i j iz M;
        formirati novi red i kolonu u M za novodobijeni klaster;
        uneti novi red i kolonu u M;
    until zadovoljen kriterijum zastavljanja
    return skup klastera;
end
```

Metode za računanje sličnosti klastera:

- **najbolja (min, single) veza** - najmanje rastojanje između bilo koje dve tačke koje pripadaju klasterima.
- **najgora (max, complete) veza** - najveće rastojanje između bilo koje dve tačke.
- **prosečna (average) veza** - prosečno rastojanje svih parova tačaka. Rastojanjima se mogu pridružiti i težine. Moguća je modifikacija gde se posmatra prosečno rastojanje svih parova tačaka koje su prisutne u oba klastera.
- rastojanje između klastera jednako je **rastojanju centroida**. Ovde se ne pravi razlika između spajanja klastera različitih veličina. Takođe, može se desiti da je rastojanje centroida na nivou k manje nego rastojanje centroida nekih klastera na nivou $k-1$. Umesto centroida, mogu se koristiti i medijane.
- **minimizacija promene varijanse** - novi klaster se formira spajanjem dva klastera čijim spajanjem se minimizuje promena varijanse unutar novodobijenog klastera. Promene u varijansi posle spajanja klastera mogu se izraziti formulom

  $$\triangledown SE_{ij} = SE_{ij} - SE_i - SE_j,$$

  gde SE_x označava prosečnu kvadratnu grešku klastera x.
- **ward** - umesto varijanse koristi zbir kvadrata grešaka, tj. spajaju se dva klastera čijim spajanjem se minimizuje povećanje zbira kvadrata grešaka unutar novog klastera.

$$E = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - c_k||^2$$

Nedostaci sakupljajućeg hijerarhijskog klasterovanja:

- Klasteri se ne mogu razdvojiti nakon kombinovanja. Može se prevazići tako što se na početku primeni particiono klasterovanje za formiranje manjih klastera, a onda se nad njim vrši hijerarhijsko klasterovanje.
- Ne postoji globalna funkcija koja se direktno minimizuje.
- Konkretni nedostaci u zavisnosti od tipa veze:
  - single - pogodna je za neeliptičke klastere, ali je osetljiva na šum i elemente van granica.
  - complete - otporna je na šum i elemente van granica, ali preferira globularne klastere i razbija velike klastere.

- average - kompromis između single i complete. Manje je osetljiva na šum i elemente van granica od single, ali ima naklonost ka globularnim klasterima.
- ward - slično kao average.

Prostorna složenost algoritma je $O(n^2)$, gde je n broj tačaka, zbog čuvanja matrice sličnosti. Vremenska složenost je $O(n^3)$ jer imamo n koraka u kojima računamo sve elemente matrice što je $O(n^2)$. Cena se može smanjiti na $O(n^2 \log n)$ ako se rastojanja za svaki klaster čuvaju u obliku sortirane liste.

Lance-Williams formula za sličnost klastera opisuje sve pomenute veze. Neka je klaster R dobijen spajanjem klastera A i B i neka je p funkcija sličnosti. Sličnost klastera R i Q je jednaka:

$$p(R,\ Q) = \alpha_A p(A,\ Q) + \alpha_B p(B,\ Q) + \beta p(A,\ B) + \gamma(p(A,\ Q) - p(B,\ Q))$$

Za različite vrste veze koriste se različiti koeficijenti. Formula nam omogućava da ne moramo da čuvamo originalne tačke, već je moguće da se matrica sličnosti ažurira kod svakog spajanja. Takođe, omogućava da sličnost računamo isto za sve veze, koristeći samo drugačije koeficijente.

Razdvajajuće klasterovanje

Neka je D skup podataka, a A neki algoritam razdvajanja klastera.

```
razdvajajuće_klasterovanje(D, A)
begin
    inicijalizovati drvo T tako da koren sadrži D;
    repeat
        izabrati list drveta L na osnovu neke strategije;
        koristeći algoritam A razdvojiti L na L1, ..., Lk;
        dodati L1, ..., Lk kao decu čvora L u drvetu T;
    until zadovoljen kriterijum zastavljanja
end
```

Algoritmi zasnovani na mrežama i gustini

Koriste se kada treba odrediti klastere sa visokom gustinom tačaka koji su razdvojeni regionima sa manjom gustinom tačaka. Mogu efikasno da odrede klastere proizvoljnog oblika. Osnovna ideja je sledeća:

- Prostor se podeli na celine (ćelije) pomoću rešetke ili graničnika drugog oblika. Ćelije mogu biti pravougaone, kružne, trouglaste ili da imaju neki nepravilan oblik. Rezultati zavise od oblika i veličine ćelija.
- Prebroje se tačke koje se nalaze u svakoj od ćelija. Ukoliko je broj tačaka u ćeliji veći od unapred zadate veličine k, ta ćelija je **gusta**.

- Guste ćelije koje imaju zajedničke ivice ili zajedničku tačku formiraju klaster. Retke ćelije ne pripadaju ni jednom klasteru.

DBSCAN je algoritam kod koga se zadaju rastojanje oko svake instance $\mathcal{E}$ i minimalan broj suseda t. Tačka pripada **jezgru** ako se u krugu poluprečnika $\mathcal{E}$ nalazi bar t drugih tačaka. Tačka je **na granici** ako se u krugu poluprečnika $\mathcal{E}$ nalazi manje od t drugih tačaka, ali se nalazi bar jedna tačka jezgra. Tačka je **šum** ako nije ni u jezgru ni na granici. Instance koje se nalaze u jezgru i nalaze se na rastojanju do $\mathcal{E}$ povezujemo u klastere. Tačke na granici se dodaju u klaster kojem pripada tačka jezgra koja se nalazi u njihovoj okolini. Tačke koje su šum se ne dodeljuju ni jednom klasteru. Prednost algoritma je u tome što pronalazi klastere proizvoljnog oblika. Nije pogodan ako želimo da svaka instanca pripada nekom klasteru. Problem nastaje i kada su klasteri različite gustine. Ako želimo da pronađemo gušće regione, onda retki regioni lako postaju šum.

Provera korektnosti klasterovanja

Kod klasterovanja provera korektnosti nije jednostavna, jer nemamo informaciju o stvarnim oznakama tačaka, kao što je to bio slučaj u klasifikaciji. Svaki algoritam klasterovanja će pronaći klastere, čak i ako u podacima uopšte ne postoje klasteri. U opštem slučaju, ne postoji najbolji algoritam za klasterovanje, već treba probati različite algoritme i videti kakva rešenja daju. Ako u svim slučajevima dobijamo loše klastere, to je znak da ne postoje prirodni klasteri. Drugi način je korišćenje statističkih testova pre samog klasterovanja. **Hopkinsova statistika**: neka je dat skup od n instanci. Odaberemo broj $p < n$ i generišemo p tačaka koje su slučajno raspoređene u prostoru koji pokriva ulazni skup i uzmemo p tačaka iz originalnog skupa. Za oba skupa tačaka, određujemo rastojanje do najbližeg suseda u originalnom skupu. Neka su v_i i o_i rastojanja do najbližih suseda za veštački generisan skup i za skup uzorkovan iz originalnog skupa. Hopkinsova statistika je:

$$H = \frac{\sum_{i=1}^{p} v_i}{\sum_{i=1}^{p} o_i + \sum_{i=1}^{p} v_i}$$

Ako tačke iz oba skupa imaju slična rastojanja, dobićemo vrednost blizu 0.5. To znači da su podaci najverovatnije slučajni i da ne postoji prirodno klasterovanje. Ako $H \to 1$ to znači da postoji velika verovatnoća da se tačke mogu klasterovati, jer će vrednosti o_i biti male.

Unutrašnji kriterijumi provere se koriste kada nemamo informacije o klasterima podataka. Neke mere su:

- **mere kohezije** koje pokazuju koliko su elementi istog klastera blizu i **mere razdvajanja** koje pokazuju koliko su elementi različitih klastera razdvojeni. Može se koristiti zbir kvadrata međusobnog rastojanja tačaka ili zbir kvadrata rastojanja tačaka od centroida. Ove mere su bolje za algoritme koji se zasnivaju na rastojanjima (npr. k-sredina), a manje odgovaraju algoritmima sa mrežama i gustinom.
- **odnos rastojanja unutar/van klastera** - iz ulaznog skupa se uzme p parova tačaka. P označava skup parova koji pripadaju istom klasteru nađenom od strane algoritma, a Q

skup parova tačaka koji pripadaju različitim klasterima. Računamo:

$$u\ klasteru = \frac{\sum_{(X_i,\ X_j) \in P} dist(X_i,\ X_j)}{|P|},\ van\ klastera = \frac{\sum_{(X_i,\ X_j) \in Q} dist(X_i,\ X_j)}{|Q|}$$

Rastojanje unutar/van klastera će biti $\frac{u\ klasteru}{van\ klastera}$. Što je vrednost manja, klasterovanje je bolje.

- **koeficijent senke (silhouette)** - neka važe sledeće oznake: $AvgDist_i^{in}$ je prosečno rastojanje X_i do tačaka unutar klastera kome pripada X_i, $AvgDist_i^{out}$ je prosečno rastojanje X_i do tačaka klastera kome ne pripada X_i, a $MinDist_i^{out}$ = $\min\{AvgDist_i^{out}\}$. Tada se koeficijent senke za i-tu tačku definiše kao:

  $$S_i = \frac{MinDist_i^{out} - AvgDist_i^{in}}{\max\{MinDist_i^{out},\ AvgDist_i^{in}\}}$$

  Uzima vrednosti iz intervala [-1, 1]. Negativne vrednosti označavaju loše klasterovanje, a vrednost bliska 1 znači da su klasteri dobro razdvojeni. Dobra osobina ove mere je to što apsolutna vrednost nosi informaciju o kvalitetu klasterovanja.
- **verovatnosna mera**

Spoljašnji kriterijumi provere se koriste kada postoje tačne informacije o klasterima u podacima. Najčešće ovo znamo ako sami generišemo skup podataka. Za proveru se najčešće koriste matrica konfuzije, Ginijev indeks, entropija, preciznost, odziv, F-mera i tako dalje.

Klasterovanje kategoričkih podataka

Jedan od načina da koristimo pomenute algoritme na kategoričkim atributima jeste da konvertujemo kategoričke atribute u binarne, što i nije baš dobro rešenje ako imamo mnogo različitih vrednosti atributa. Možemo da izvršimo mapiranje kategoričkih u numeričke, ali najčešće ne postoji neka prirodna uređenost kategoričkih atributa. Zbog toga, postoje verzije algoritama za kategoričke atribute.

Algoritam k**-modalno klasterovanje** za svaki od atributa određuje modalnu vrednost, tj. vrednost sa najvećom frekvencijom. Modalna vrednost svakog atributa se određuje nezavisno od ostalih atributa, tako da centroid ne mora da pripada skupu podataka. Radi dobro kada su vrednosti atributa ravnomerno raspoređene. Kada to nije slučaj, normalizuju se frekvencije pojavljivanja. k**-medoid klasterovanje** je varijacija koja podrazumeva da je prototip tačka iz skupa podataka.

Klasterovanje po potprostorima

Podrazumeva da se podaci klasteruju prema podskupu atributa. Različiti klasteri se javljaju u zavisnosti od različitih podskupova. Neki algoritmi:

- **CLIQUE** (CLusteing In QUEst) - zasnovan na sličnom principu kao Apriori. Kombinuje klasterovanje zasnovano na gustini i mrežama za pronalaženje potprostora pogodnih za

klasterovanje. Pronalaze se ćelije sa gustinom većom od praga, a guste ćelije se spajaju ako se dodiruju ivicom, čime formiraju klastere. Određuje klastere različitih oblika i veličina.

- **MAFIA** (Merging of Adaptive Finite IntervAls) - varijanta CLIQUE algoritma koja za klasterovanje bira potprostore obuhvaćene atributima čija je entropija manja od praga. Može da odredi klastere sa jako malom gustinom, ali ima visoku cenu izračunavanja.
- **ENCLUS** (ENtropy-based CLUStering) - varijanta CLIQUE algoritma koja koristi kriterijum za izbor potprostora zasnovan na entropiji. Za određivanje broja ćelija u svakoj dimenziji koristi histograme. Omogućava paralelizaciju i može da obradi veliku količinu podataka.
- **DENCLUE** (DENsity CLUstEring) - zasnovan na funkciji uticaja koja modelira uticaj tačaka na svoje susede. Gustina u nekoj tački se procenjuje kao zbir uticaja svih ostalih tačaka na nju. Tačka privlačenja je tačka koja odgovara lokalnom maksimumu funkcije gustine. Klaster se definiše kao skup svih tačaka povezanih sa nekom tačkom privlačenja, pri čemu je vrednost funkcije gustine u tački privlačenja veća od nekog praga.

Klasterovanje skalabilnih podataka

Klasterovanje skalabilnih podataka je klasterovanje veoma velikih skupova podataka korišćenjem algoritama koji mogu efikasno da rade i kada obim podataka značajno raste. Omogućava klasterovanje podataka koji ne mogu da stanu u memoriju. Algoritmi:

- **CLARA** (Clusteing LARge Application) i **CLARANS** (Clustering Large Application on RANdomized Search) - uopštenje algoritma k-medoida.
- **CURE** (Clustering Using Representatives) - sakupljajući algoritam za hijerarhijsko klasterovanje.
- **BIRCH** (Balanced Iterative Reducing and Clustering using Hierarchy) - uopštenje algoritma k-sredina na sakupljajuće hijerarhijsko klasterovanje.

Samoorganizujuće mape

Samoorganizujuće mape (SOM) su slične algoritmu k-sredina. Uključuju topografsku organizaciju centroida. Najčešće se koriste 2D mape. Pored veličine mape, određujemo i kako će izgledati ćelije, tj. da li će biti pravougaonik, šestougao i slično, kao i koliko su dva centroida udaljena da bismo rekli da su bliski. Svaka ćelija je opisana centroidom. Na početku inicijalizujemo centroide. Kao i kod k-sredina, instancu poredimo sa svim centroidima i dodeljujemo ćeliji čijem centroidu je najbliža, a onda ažuriramo centroid tako da on bude malo bliži instanci koju smo mu pridružili. Istovremeno, ažuriraju se i centroidi koji su mu u blizini po topografskoj orijentaciji, ali slabije nego što se ažurira sam centroid. Algoritam se zaustavlja kada se centroidi ne menjaju ili kada je dostignut neki drugi prag. Algoritam se ponaša dobro kada imamo veliki broj instanci. Neka su m_1, ..., m_k centroidi, $p(t)$ tekući objekat u trenutku t i neka je njemu najbliža centroida m_j. U trenutku $t+1$, j-ti centroid se ažurira na:

$$m_j(t+1) = m_j(t) + h_j(t)(p(t) - m_j(t))$$

Vrednost $h(t)$ određuje efekat razlike. Na početku obično uzimamo veću vrednost kako bi centroidi konvergirali ka instancama, a pri kraju je smanjujemo kako instance ne bi mnogo menjale centroide. Prednosti SOM su to što su susedni klasteri više u relaciji od nesusednih, pogodne su za vizuelizaciju, a mogu se koristiti i za kompresiju podataka u prostor niže dimenzije. Nedostaci:

- Odabir parametara (veličina mape, oblik ćelije, prag susedstva) nije jednostavan.
- SOM klaster često ne odgovara prirodnom klasteru, tj. SOM klaster može da sadrži više prirodnih klastera, a jedan prirodni klaster može biti razbijen na više SOM klastera.
- Ne postoji funkcija objekta kojom može da se izrazi postupak.
- Nema garancije za konvergenciju, ali u praksi algoritam najčešće konvergira.

Otkrivanje anomalija

Anomalije (elementi van granica) su podaci čije su vrednosti različite od vrednosti ostalih podataka u materijalu. **Hokinsova definicija anomalije**: anomalija je opservacija koja se toliko razlikuje od ostalih opservacija da se javlja sumnja da je nastala pomoću drugačijeg mehanizma. Anomalije su relativno retke, ali mogu značajno da utiču na rezultat istraživanja. Anomalije mogu nastati kao mehaničke ili ljudske greške, greške u instrumentima koji formiraju podatke, posledica podataka koji su sakupljeni iz različitih izvora, posledica promena u ponašanju sistema itd. Neki algoritmi su osetljivi na anomalije, a neki su otporni. Anomalije se otklanjaju u fazi pripreme podataka. **Šum** je pogrešna vrednost ili događaj sa greškom. Na primer, težina je pogrešno zapisana. U pitanju je slučajan događaj koji ne mora da proizvede neuobičajene vrednosti. Šum nije od interesa u istraživanju. Animalije jesu od interesa ako nisu rezultat šuma.

Na proces otkrivanja utiče i broj atributa. Kada imamo više atributa, teže je otkriti anomalije. Možemo otkrivati anomalije samo u odnosu na neke atribute. Može se desiti da ni jedan od atributa pojedinačno nema anomaliju, ali da anomaliju možemo otkriti njihovom kombinacijom (npr. visina i težina). **Globalna i lokalna perspektiva** posmatranja odnose se na to da neke instance mogu biti anomalije u zavisnosti kako to posmatramo. Na primer, može se reći da čovek od 2m+ predstavlja anomaliju, ali ako gledamo košarkaše to nije anomalija. **Veličina anomalije** podrazumeva da ne određujemo samo da li je instanca anomalija ili nije, već da odredimo neki skor. U ovom pristupu moramo da odredimo prag. Sve instance čiji je skor veći od praga su anomalije. Najčešće varijante problema otkrivanja anomalija su:

- Za dati skup D naći sve tačke $x \in D$ čija ja veličina anomalije iznad praga t.
- Za dati skup D naći sve tačke $x \in D$ koje imaju n najvećih vrednosti veličine anomalije.
- Za dati skup D koji najvećim delom sadrži normalne, ali neoznačene tačke i test tačku x, odrediti veličinu anomalije tačke x u odnosu na skup D.

Kod tehnika za otkrivanje anomalija, pretpostavka je da u posmatranim podacima postoji značajno veći broj "normalnih" podataka nego anomalija. Tehnike mogu biti:

- **zasnovane na formiranju modela** - pravimo model koji će da određuje da li je instanca normalna ili anomalija. Retko unapred znamo da li je instanca anomalija ili ne. Kod ovih tehnika imamo dva koraka. U prvom koraku se pravi model. Kod nadgledanih modela, anomalije su tačke koje se ne uklapaju dobro u karakteristike i tačke koje narušavaju izgled modela. Kod nenadgledanih modela, anomalije su tačke koje pripadaju retkim klasterima. U drugom koraku se nalaze podaci koji odskaču, koristeći napravljeni model. Ako imamo veliki skup, za pravljenje modela možemo uzeti neki podskup, pri čemu treba biti pažljiv da ne bismo isključili anomalije.
- **zasnovane na statistici** - anomalija je element koji ima manju verovatnoću pojavljivanja u odnosu na distribuciju verovatnoća u modelu podataka. Mana je što moramo da poznajemo distribuciju podataka, a nekada podaci imaju mešanu distribuciju. Ako posmatramo jedan atribut, u pitanju je **unimodalna statistika**. Na primer, ako imamo normalnu raspodelu, možemo da posmatramo očekivanu vrednost i standardnu devijaciju i da kažemo da su anomalije vrednosti koje su izvan opsega $[m - 2\sigma,\ m + 2\sigma]$. Za normalnu raspodelu, 95.4% nalazi se u ovom opsegu. Ako uzmemo 3 standardne devijacije, preko 99% podataka nalazi se u tom opsegu. Kada je broj elemenata mali, za procenu m i σ se koristi **Grubov metod**, gde se umesto normalne koristi studentova raspodela sa n stepeni slobode. Možemo koristiti i **izgledne verovatnoće**. Pretpostavka je da skup D sadrži primerke sa mešavinom dve raspodele: M je raspodela normalnih podataka, a A je raspodela podataka sa anomalijom. Inicijalno kažemo da sve instance imaju raspodelu M i računamo njihovu verovatnoću za tu raspodelu. Zatim svaku instancu prebacujemo u drugi skup i računamo verovatnoću sa drugom raspodelom. Računamo razliku druge i prve verovatnoće i ako je razlika veća od nekog praga, tada je instanca anomalija i ona ostaje u skupu sa raspodelom A, a inače u skupu sa raspodelom M. Tehnike zasnovane na statistici imaju veliku efikasnost i strogo su matematički zasnovane. Daju dobre rezultate ako je poznata raspodela. Problemi se javljaju kada imamo višedimenzionalne podatke.
- **sa vizuelizacijom** - posmatranje udaljenosti i drugih karakteristika instanci. Tehnika je zgodna ako imamo mali broj atributa. Mana ovog pristupa je što su tehnike podložne subjektivnoj oceni podataka.
- **zasnovane na određivanju rastojanja** - instanca je anomalija ako je cela ili neki njen deo udaljen više od predviđene granice od ostalih instanci. Tehnika k-najbližih suseda podrazumeva da se posmatra rastojanje k najbližih suseda. Ako uzmemo suviše malo k može se desiti da anomalije proglasimo normalnim instancama, a ako uzmemo previše veliko k može se desiti da normalne instance proglasimo anomalijama. Ove tehnike su jednostavne za primenu, ali su računarski zahtevne i osetljive na promenu parametra. Imaju problem kada su podaci višedimenzionalni. Pored Euklidskog rastojanja, može se koristiti i Mahalanobisovo rastojanje, koje uzima u obzir i gustinu između tačaka za koje meri rastojanje.

- **zasnovane na određivanju gustine** - veličina anomalije instance je obrnuto proporcionalna gustini elemenata u njenom okruženju. Može se koristiti tehnika k-najbližih suseda, gde se posmatra inverzno rastojanje do k najbližih suseda. Možemo posmatrati i inverz prosečnog rastojanja do k suseda. Ove tehnike imaju problem kada imamo regione sa različitim gustinama. Umesto samo gustinu instance, možemo gledati i gustine susednih instanci, tj. **relativnu gustinu**. Ako je gustina instance manja od prosečne gustine njenih suseda, onda je instanca anomalija. **LOF (Local Outline Factor)** tačke je odnos prosečne gustine susednih tačaka te tačke i gustine same tačke. Ako je LOF veliki u pitanju je element van granica. Tehnike zasnovane na gustini su jednostavne, ali računarski zahtevne i osetljive na promene parametara. Takođe, postoji problem sa određivanjem gustine u višedimenzionalnom prostoru.
- **zasnovane na klasterovanju** - objekat je anomalija ako je očigledno da ne pripada ni jednom klasteru. Ako koristimo metode zasnovane na prototipovima, instanca je anomalija ako nije blizu centra ni jednog klastera. Ako koristimo metode zasnovane na gustini, instanca je anomalija ako je njena gustina mala. Ako koristimo metode zasnovane na grafovima, instanca je anomalija ako nije dobro povezana. Možemo koristiti i relativno rastojanje, tj. da posmatramo rastojanja suseda. Problem klasterovanja je to što neke metode formiraju klastere sa malim brojem elemenata, a problematično je i određivanje tehnike klasterovanja i broja klastera.

Pravila pridruživanja

Svaki slog sastoji se od jedne **transakcije**. Jednu transakciju opisuje njen **ID** i niz **stavki** koje se pojavljuju u transakciji. Određivanje **pravila pridruživanja** je proces u kome se za dati skup transakcija pronalaze pravila koja predviđaju pojavljivanje stavke na osnovu pojavljivanja ostalih stavki u transakcijama. Pravila su oblika:

$$telo \rightarrow glava$$

Ovo znači da ako se u transakciji nađe telo, verovatno će se naći i glava. Klasičan primer je potrošačka korpa, koja sadrži skup artikala koje je kupac kupio u supermarketu. Svaka potrošačka korpa sadrži jednu transakciju. Neki od ciljeva nalaženja pravila pridruživanja na osnovu potrošačkih korpi su:

- Utvrditi koji se artikli kupuju zajedno. Smestiti artikle koji se kupuju zajedno jedne pored drugih.
- Ako kupac kupi neke proizvode, utvrditi koje će još proizvode verovatno kupiti. U tom slučaju ne želimo da istovremeno damo popust na sve te artikle.
- Pronaći neke neuobičajene trendove.
- Predvideti prodaju i blagovremeno naručivati zalihe.

Osnovni pojmovi:

- **Skup stavki (itemset)** sadrži jednu ili više stavki. k**-skup stavki** je skup koji sadrži k stavki.

- **Dužina transakcije** je broj stavki koje se nalaze u transakciji.
- **Podrška skupa stavki** A, u oznaci $\#(A)$ je broj transakcija u kojima se javlja taj skup stavki. Skup stavki je **čest** ako je njegova podrška veća od praga podrške koji se zadaje unapred ($minsup$).
- **Podrška pravila pridruživanja** $A \to B$ je količnik broja transakcija koje sadrže A i B u odnosu na ukupan broj transakcija N.

$$\sup(A \to B) = \frac{\#(A \cup B)}{N}$$

- **Pouzdanost pravila pridruživanja** $A \to B$ je količnik broja transakcija koje sadrže A i B u odnosu na broj transakcija koje sadrže A.

$$conf(A \to B) = \frac{\#(A \cup B)}{\#(A)}$$

Cilj pravila pridruživanja je naći sva pravila čija je podrška veća od minimalnog praga podrške i čija je pouzdanost veća od minimalnog praga pouzdanosti. Može se desiti da neko pravilo ispunjava ove uslove, ali generalno nije mnogo bitno. U slučaju jako velikih baza podataka, minimalni prag pouzdanosti je obično visok (80%), a minimalni prag podrške je znatno niži (5-10%). Visoka podrška znači da se pravilo često pojavljuje i da je manje verovatno da se slučajno pojavilo, mada nekada mogu biti značajna i pravila sa jako niskom podrškom. Visoka pouzdanost označava da je pravilo jako, tj. da ako se javi telo vrlo verovatno će se javiti i glava.

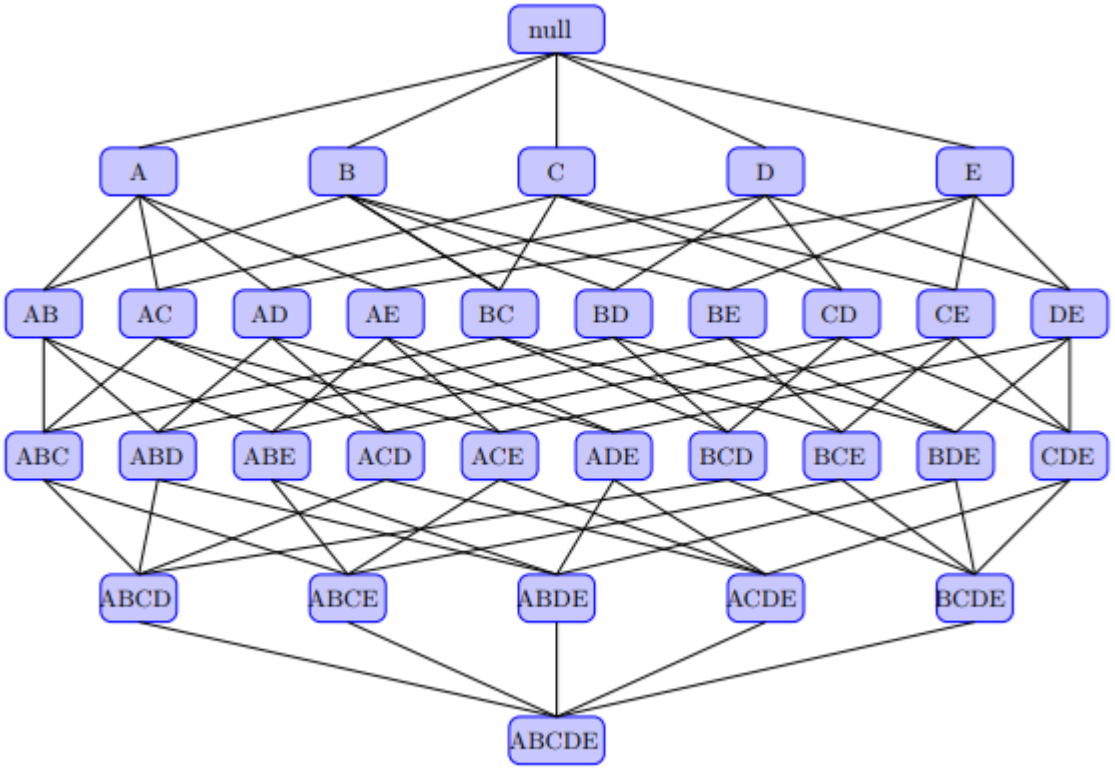
Proces određivanja pravila pridruživanja se može podeliti na dva koraka:

1. Tražimo sve česte skupove stavki.
2. Formiramo pravila iz svakog skupa čestih stavki sa pouzdanošću većom od minimalnog praga pouzdanosti.

U praksi, imamo stotine hiljada različitih stavki. Za d artikala imamo $2^d - 1$ mogućih skupova, za koje bismo morali da proverimo da li su česti ili ne. Potrebno je i često primenjivati ovo, jer se transakcije stalno menjaju. Za proveru da li su svi skupovi česti treba $O(NMw)$ vremena, gde je N broj transakcija, $M = 2^d - 1$ broj skupova kandidata i w maksimalna dužina transakcije. Stavke iz skupa transakcija se najčešće predstavljaju u obliku **rešetke**.

Efikasnost nalaženja čestih skupova se može smanjiti:

- Smanjenjem broja kandidata (M) - koriste se tehnike potkresivanja (npr. Apriori).
- Smanjenjem broja transakcija (N) - sa povećanjem veličine skupova kandidata, smanjuje se broj transakcija koje sadrže skupove te veličine.
- Smanjenjem broja poređenja (NM) - koriste se efikasnije strukture podataka radi čuvanja skupova kandidata i/ili transakcija, pa nije potrebno uparivati svaki skup kandidat sa svakom transakcijom.



Apriori princip podrazumeva da ako je A čest skup stavki, onda su česti i svi njegovi podskupovi. Mera f poseduje **osobinu antimonotonosti** ako za sve skupove stavki X, Y važi:

$$X \subset Y \ \Rightarrow \ f(Y) \leq f(X)$$

Podrška ima osobinu antimonotonosti. Kao posledica, ako neki skup stavki A nije čest, tada ni bilo koji njegov nadskup nije čest, tj. ne mora da se razmatra. Za pouzdanost važi da ako su X_1, X_2 i I skupovi stavki takvi da važi $X_1 \subset X_2 \subset I$, tada važi:

$$conf(X_2 \rightarrow I - X_2) \geq conf(X_1 \rightarrow I - X_1)$$

Kao posledica, možemo da uklonimo redundantna pravila.

Algoritmi za određivanje čestog skupa stavki

Metoda grube sile

Svaki skup stavki u rešetki je kandidat da bude čest i računamo podršku za svaki kandidat. Svaka transakcija se uparuje sa svakim kandidatom. Algoritam služi da se teorijski pokaže da je moguće dobiti skup pravila pridruživanja koja ispunjavaju uslov da imaju veću podršku i pouzdanost od zadatih pragova. U praksi je gotovo nemoguće izvršiti ovaj algoritam za veće skupove transakcija. Složenost je $O(NMw)$. Ukupan broj mogućih pravila pridruživanja za d stavki je:

$$\sum_{k=1}^{d-1}[\binom{d}{k}\sum_{j=1}^{d-k}\binom{d-k}{j}]$$

Apriori algoritam

Koristi Apriori princip i antimonotonost da smanji broj kandidata za skupove čestih stavki. Algoritam:

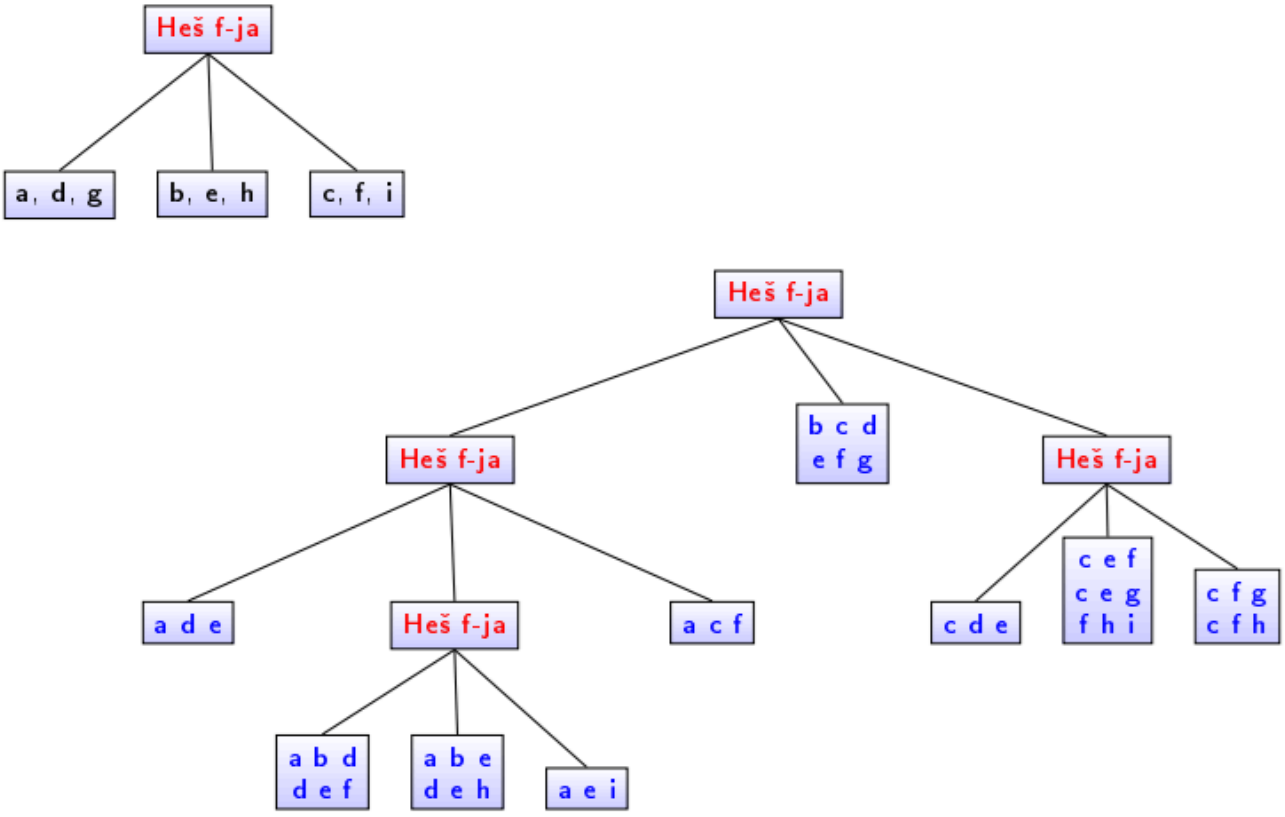
- Elementi skupova stavki se uređuju u leksikografski poredak.
- Prebroje se sve 1-stavke i odbace one koje nisu česte.
- Na osnovu čestih skupova stavki dužine k, formiraju se skupovi dužine $k+1$. Skupovi k-stavki se spajaju i formiraju $k+1$-skup stavki akko su im jednaki prvih $k-1$ elemenata.
- Odbacuju se skupovi $k+1$-stavki ako neki od podskupova k-stavki nije čest.
- Za formirane skupove dužine $k+1$ određuje se podrška.
- Odbacuju se retki skupovi $k+1$-stavki i ne razmatraju se njihovi nadskupovi.

Na osnovu čestih skupova stavki koji su određeni, vrši se formiranje pravila pridruživanja:

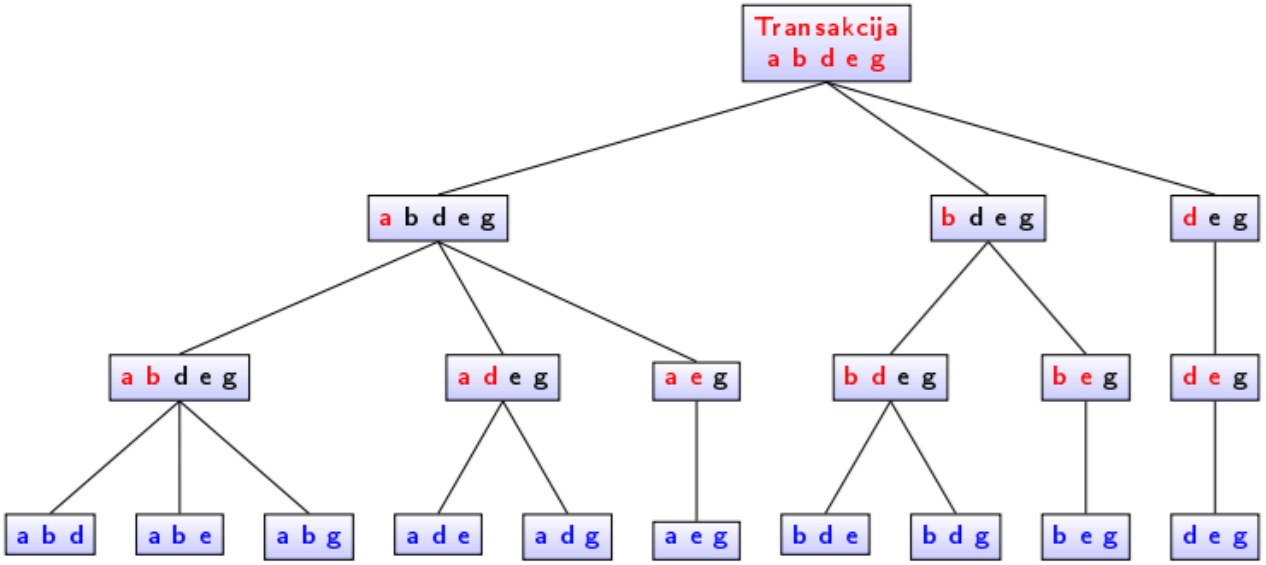
- Svaki čest k-skup može da proizvede do 2^k-2 pravila pridruživanja, jer ignorišemo pravila sa praznom levom ili desnom stranom.
- Pravilo se izdvaja deljenjem skupa stavki Y na dva neprazna podskupa X i $Y-X$, takve da $X \to Y-X$ ima veću pouzdanost od zadatog praga pouzdanosti.

Postoje dve metode za formiranje skupova dužine k. Prva metoda je $F_{k-1} \times F_{k-1}$ koja podrazumeva da su elementi $k-1$-skupa stavki uređeni u leksikografskom poretku i da će se spojiti u novi ako im je prvih $k-2$ stavki jednako. Druga metoda je $F_{k-1} \times F_1$ gde se na $k-1$-skup stavki dodaje 1-skup stavki. I ovde se koristi leksikografsko uređenje, da bismo izbegli da se više puta formira isti skup stavki. To znači da se $k-1$-skup stavki proširuje sa 1-skupom stavki samo ako je ta stavka leksikografski posle svih stavki iz $k-1$-skupa stavki. Prvi metod je bolji, jer kombinuje samo česte skupove stavki.

Da bi smanjio broj poređenja, apriori algoritam kandidatske stavke smešta u **heš drvo**, a zatim primenjuje funkciju koja na osnovu heš drveta i jedne transakcije računa podršku svih kandidata te transakcije. Za smeštanje skupova stavki u heš drvo, heš funkcija se na i-tom nivou primenjuje na i-tu stavku skupa stavki. Za svaki čvor u drvetu postoji limit broja skupova koji se u njemu mogu naći. Radi efikasnosti svi skupovi stavki čuvaju stavke u rastućem leksikografskom uređenju. Primer heš drveta kandidatskih stavki dužine 3, skupa stavki {a, b, c, d, e, f, g, h, i}:



Stavke iz transakcije se takođe grupišu u **korpe** i upoređuju sa kandidatskim stavkama. Primer heš drveta transakcije {a, b, d, e, g}:



Stavke iz transakcija se porede sa sadržajem svoje kandidatske korpe, umesto se celim skupom kandidata, čime se smanjuje broj poređenja. U primeru se dobijaju 3-stavke {ade}, {abd} i {abe}.

U praksi, broj čestih stavki može biti jako veliki, pa je potreban jednostavan način za čuvanje svih čestih skupova. Skup stavki je **maksimalno čest** ako ni jedan od njegovih neposrednih nadskupova nije čest. Ako je skup stavki maksimalno čest, tada su i svi njegovi podskupovi česti. Odavde sledi da je za čuvanje svih čestih skupova dovoljno čuvati samo maksimalno česte skupove. Maksimalni skupovi stavki ne čuvaju informaciju o podršci. Skup stavki je **zatvoren** ako ni jedan od njegovih neposrednih nadskupova nema istu podršku kao sam taj skup. Kao posledica, svi podskupovi zatvorenog skupa stavki imaju istu ili veću podršku.

Presekom skupa maksimalnih i skupa zatvorenih skupova stavki dobijaju se **maksimalni zatvoreni skupovi stavki**. **Kompaktno predstavljanje čestih stavki** podrazumeva da je dovoljno čuvati samo maksimalno zatvorene skupove stavki i njihove podrške.

Obilazak rešetke **nivo po nivo (u širinu)** može biti:

- **od opšteg ka posebnom** - obilazak na dole. Bolji je kada imamo transakcije koje sadrže manji broj stavki.
- **od posebnog ka opštem** - obilazak na gore. Bolji je kada imamo transakcije u kojima se javlja većina stavki. Dobar je i kada tražimo maksimalne skupove stavki.
- **dvosmeran** - obilazak iz oba smera.

Obilazak može biti i po klasama koje su zasnovane na **prefiksnim** ili **sufiksnim delovima** skupa stavki. Pored obilaska u širinu, može se koristiti i obilazak **u dubinu**. Koristan je kada želimo da odredimo maksimalne skupove stavki.

Transakcije su uobičajeno predstavljene **horizontalno**, tj. kao lista stavki koje ih čine. Mogu se predstaviti i **vertikalno**, gde se stavke nalaze u kolonama, a vrednosti u poljima su 1 ako se ta stavka pojavljuje u datoj transakciji. U tom slučaju primenjuje se **vertikalni Apriori**:

```
vertikalni_apriori(T, minsup)
begin
    k = 1;
    F1 = {sve česte 1-stavke};
    formirati vertikalne liste transakcija za stavke (tr_id);
    while (Fk nije prazno) do
        generisati C(k+1) spajanjem parova stavki iz Fk;
        odbaciti stavke iz C(k+1) koje ne zadovoljavaju zatvorenje na niže;
        formirati tr_id liste svake kandidatske stavke iz C(k+1) kao presek
            tr_id listi para stavki iz Fk korišćenih za formiranje C(k+1);
        odrediti podršku stavki u C(k+1) računanjem dužine liste;
        F(k+1) = česte stavke C(k+1) zajedno sa tr_id listama;
        k = k + 1;
    end while
    return (unija svih Fi, i = 1, ..., k);
end
```

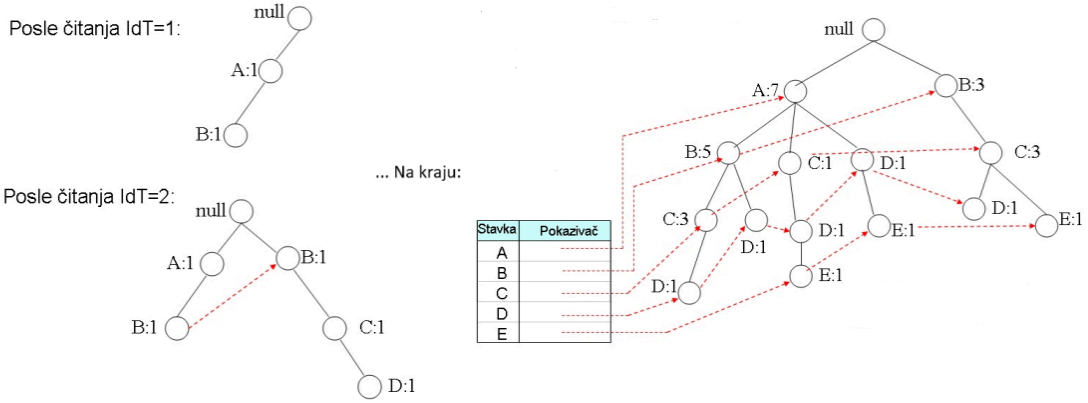
Vertikalni Apriori je brži jer se podrška računa kao dužina liste, ali zahteva veću potrošnju memorije zbog čuvanja tih listi.

Algoritam FP rasta

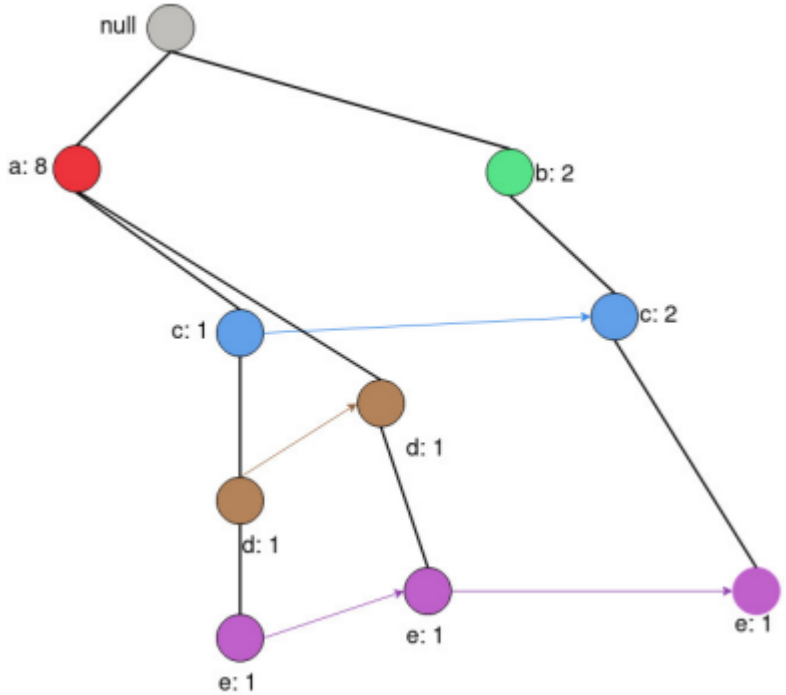
Algoritam FP rasta koristi komprimovanu reprezentaciju baze podataka pomoću **FP (Frequent Pattern) drveta**. Što je veći broj transakcija koje imaju zajedničke početne delove, to je veći stepen kompresije koji se postiže ovim algoritmom. Algoritam može biti i višestruko brži od algoritama koji konstruišu i obilaze rešetku po širini, ali je memorijski zahtevniji.

Stavke u transakcijama se prvo sortiraju opadajuće po frekvenciji pojavljivanja. Konstrukcija FP drveta počinje od korena koji ne sadrži ni jednu stavku i označen je sa $null$. U čvorovima drveta koji se konstruišu upisuje se stavka i broj koji predstavlja koliko transakcija je prošlo tim čvorom. Čvor može sadržati i **vezu** - pokazivač na sledeći čvor sa istim imenom u FP-drvetu ili $null$ ako takav čvor ne postoji. Formira se i **tabela** sa skupom pokazivača na liste istoimenih stavki u različitim granama drveta. Primer (prag je 2):

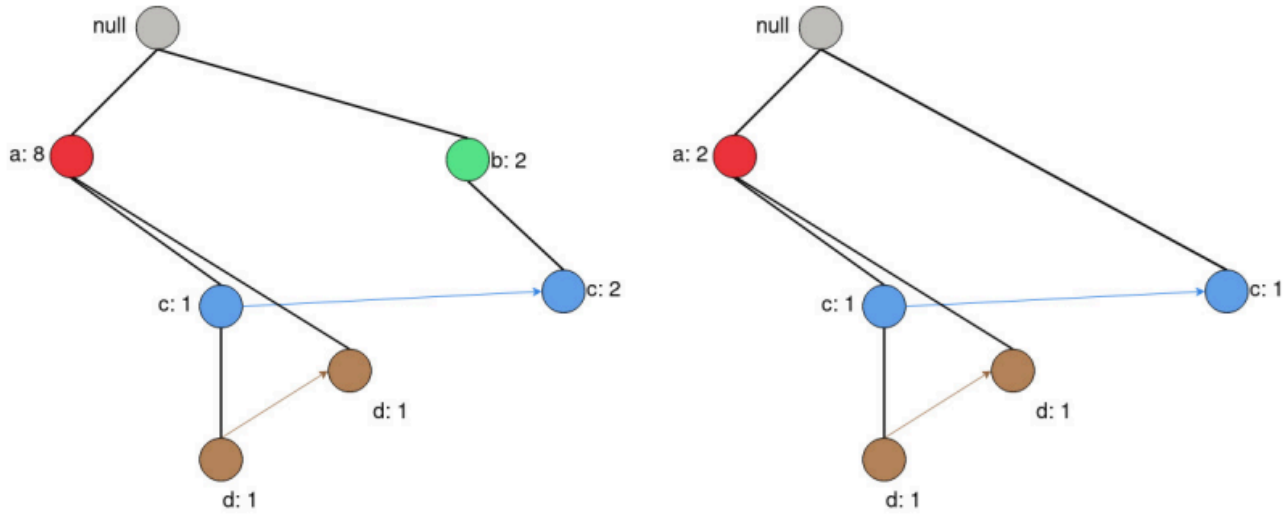
IdT	Stavke
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}



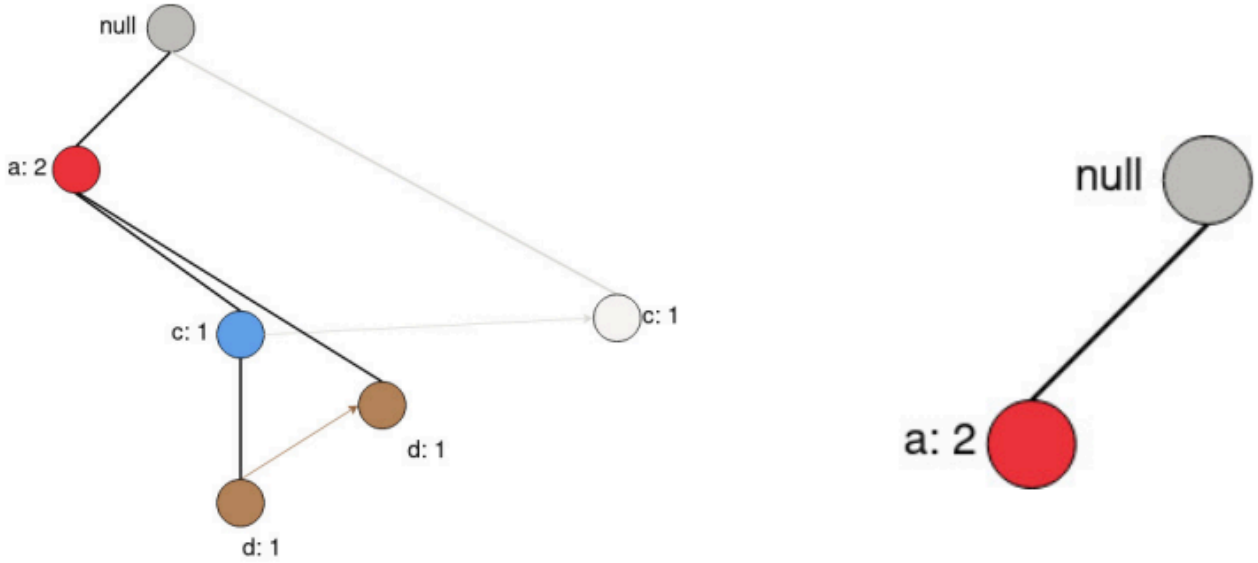
Česti skupovi stavki se određuju odozdo naviše. U primeru, prvo se određuju česte stavke koje se završavaju na e, pa one koje se završavaju na d, c, b i na kraju one koje se završavaju na a. Do čestih stavki koje imaju sufiks e dolazi se ispitivanjem samo onih grana koje imaju list e.



Prvo ispitamo da li je skup stavki $\{e\}$ čest skup. Ako jeste, određujemo česte stavke koje se završavaju redom na de, ce, be i ae. To radimo tako što eliminišemo čvorove koji sadrže e (*levo*) i ažuriramo broj pojavljivanja stavki na višim granama drveta. U primeru, broj pojavljivanja b i c se smanjuje za 1 jer predstavljaju transakciju $\{b,\ c\}$ koja ne sadrži e. Eliminišu se čvorovi koji imaju broj pojavljivanja u poddrvetu manji od praga i dobija se **uslovno FP-drvo** za e (*desno*).



Dobijamo da je $\{d,\ e\}$ česta stavka. Analogno, postupak ponavljamo za de tako što izbacujemo čvor d (*desno*). Dobijamo da je $\{a,\ d,\ e\}$ česta stavka.



Unijom skupova za de, ce, be i ae dobijaju se sve česte stavke koje imaju sufiks e. Isti postupak se ponavlja za stavke koje se završavaju sufiksom d, c, b i a i unija dobijenih skupova predstavlja skup svih čestih stavki.

Mere interesantnosti pravila

U praksi se često dobija jako veliki broj pravila pridruživanja. To često nije korisno za dalju analizu, a može da dovede i do toga da se previdi neko od interesantnih pravila. Ne postoji mera koja je najbolja u svim slučajevima, već svaka ima svoje prednosti i nedostatke, kao i slučajeve u kojima pogrešno procenjuje interesantnost pravila. **Tabela kontigenata** sadrži učestalost pojavljivanja stavki i koristi se za ilustraciju različitih mera. Tabela kontigenata za par binarnih promenljivih A i B:

	B	$\overline{B}$	
A	f_{11}	f_{10}	f_{1+}
$\overline{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

A označava da je stavka prisutna u transakciji, a $\overline{A}$ da nije. Vrednosti f su brojači:

- f_{10} je broj transakcija koje sadrže samo A.
- f_{11} je broj transakcija koje sadrže i A i B.
- f_{01} je broj transakcija koje sadrže samo B.
- f_{00} je broj transakcija koje ne sadrže ni A ni B.
- f_{1+} je broj transakcija koje sadrže A.
- f_{0+} je broj transakcija koje sadrže $\overline{A}$.
- f_{+1} je broj transakcija koje sadrže B.
- f_{+0} je broj transakcija koje sadrže $\overline{B}$.

Ograničenje mera podrške i pouzdanosti je to što ne uzimaju u obzir podršku desne strane pravila. Na primer, možemo dobiti pravilo oblika $X \to Y$ koje ima visoku pouzdanost, ali se Y javlja u svakoj transakciji, pa to ipak nije interesantno.

Činjenice koje se koriste nadalje:

- Podrška $\mathrm{sup}(A)$ meri verovatnoću pojavljivanja A, tj. $\mathrm{sup}(A) = \frac{f_{1+}}{N}$.
- Podrška $\mathrm{sup}(A,\ B)$ meri verovatnoću da se A i B zajedno pojavljuju, tj. $\mathrm{sup}(A,\ B) = \frac{f_{11}}{N}$.
- Ako su A i B nezavisni, onda je $P(A,\ B) = P(A)P(B)$, tj. $\mathrm{sup}(A,\ B)$ = $\mathrm{sup}(A) \cdot \mathrm{sup}(B)$ = $\frac{f_{1+}}{N}$. $\frac{f_{+1}}{N}$.
- Odstupanje $\mathrm{sup}(A,\ B)$ od $\mathrm{sup}(A) \cdot \mathrm{sup}(B)$ je znak statističke zavisnosti A od B. Pouzdanost meri odstupanje $\mathrm{sup}(A,\ B)$ od $\mathrm{sup}(A)$, ali ne i od $\mathrm{sup}(B)$.

Lift je mera koja uzima u obzir i podršku desnog dela pravila.

$$Lift(A,\ B) = \frac{conf(A \to B)}{\mathrm{sup}(B)}$$

Vrednosti veće od 1 označavaju da se B češće pojavljuje u transakcijama koje sadrže i A, nego u transakcijama koje je ne sadrže. Vrednosti manje od 1 označavaju pravila čija je pouzdanost manja od očekivane. Dakle, zanimljiva su pravila koja imaju Lift meru veću od 1.

Pravilo koje ima manju podršku i veći Lift može u nekim slučajevima da bude manje interesantno od pravila koje ima veću podršku i manji Lift, pošto je takvo pravilo primenljivo na veći deo podataka. **Piatetsky-Shapiro mera (mera poluge)** je mera koja uzima u obzir i veličinu i jačinu efekta pravila pridruživanja.

$$PS = \mathrm{sup}(A,\ B) - \mathrm{sup}(A)\,\mathrm{sup}(B)$$

Ako je vrednost 0, A i B su međusobno nezavisni, a ako je vrednost pozitivna/negativna, onda su A i B pozitivno/negativno korelisani.

Odnos kamata se za skup stavki $i_1,\ \ldots,\ i_k$ definiše kao

$$I(i_1,\ \ldots,\ i_k) = \frac{\sup(i_1,\ \ldots,\ i_k)}{\Pi_{j=1}^{k} \sup(i_j)}$$

U slučaju binarnih promenljivih, odnos kamate se poklapa sa Lift merom:

$$I(A,\ B) = \frac{\sup(A,\ B)}{\sup(A)\sup(B)} = Lift(A,\ B)$$

Ako je vrednost 1, A i B su međusobno nezavisni, a ako je vrednost veća/manja od 1, onda su A i B pozitivno/negativno korelisani. Mera daje netačne rezultate kada je neka stavka ekstremno retka. Tada ona može biti uparena sa bilo kojom stavkom sa kojom se javlja zajedno i taj skup stavki će imati jako veliki odnos kamate.

Deployability (mogućnost širenja) je procenat skupa podataka koji zadovoljava uslov na levoj strani pravila, ali ne i posledični deo na desnoj strani pravila.

$$D = \frac{\sup(A) - \sup(A,\ B)}{N}$$

Poželjne su vrednosti bliže 0, što znači da je pravilo veoma pouzdano.

Koeficijent korelacije (Pitersonov koeficijent) je normalizovana verzija PS mere, tj. uzima vrednosti iz segmenta [-1, 1].

$$\rho = \frac{\sup(A,\ B) - \sup(A)\sup(B)}{\sqrt{\sup(A)\sup(B)(1-\sup(A))(1-\sup(B))}}$$

Njegova mana je to što ne ostaje invarijantan sa proporcionalnom promenom veličine ulaznih podataka.

$\mathcal{X}^2$ **mera** je simetrična mera koja na isti način tretira prisustvo i odsustvo stavki. Neka su O_i i E_i osmotrena i očekivana vrednost apsolutne podrške stavki u stanju i. $\mathcal{X}^2$ mera skupa stavki X je

$$\mathcal{X}^2(X) = \sum_{i=1}^{2^{|X|}} \frac{(O_i - E_i)^2}{E_i}$$

Vrednosti koje su blizu 0 označavaju statističku nezavisnost između stavki. Veće vrednosti označavaju zavisnost između stavki, ali ne nose informaciju da li je ta zavisnost pozitivna ili negativna. $\mathcal{X}^2$ test zadovoljava osobinu zatvorenja naviše, što kao posledicu ima mogućnost konstrukcije efikasnog algoritma za određivanje interesantnih k-skupova stavki.

IS mera je alternativna mera koja se primenjuje u slučaju asimetričnih binarnih promenljivih.

$$IS(A,\ B) = \sqrt{I(A,\ B)\sup(A,\ B)} = \frac{\sup(A,\ B)}{\sqrt{\sup(A)\sup(B)}}$$

IS mera raste kada rastu odnos kamate i podrška. Ako dva pravila imaju isti odnos kamate, IS daje prednost onom sa većom podrškom. IS mera je ekvivalentna kosinusnoj meri za binarne promenljive.

Kosinusna mera je simetrična mera koja može da se primeni na kolone radi računanja sličnosti među stavkama u tim kolonama. Računa se koristeći vertikalnu reprezentaciju listi odgovarajućih binarnih vektora. Ako su A i B par vektora tada je $A \circ B$ = $\sup(A,\ B)$, a $|A|$ = $\sqrt{\sup(A)}$ je veličina vektora A, pa se dobija:

$$IS(A,\ B) = \frac{\sup(A,\ B)}{\sqrt{\sup(A)\sup(B)}} = \frac{A \circ B}{|A| \cdot |B|} = \cos(A,\ B)$$

Pouzdanost svih je mera koja se odnosi na skupove stavki, a ne na pravila. Podržava zatvorenje naniže i računa se po formuli:

$$all\_confidence(X) = \frac{\sup(X)}{\max_{x \in X} \sup(X)}$$

Mera kaže da sva pravila koja mogu da se izvedu iz skupa stavki X imaju podršku jednaku bar $all\_confidence(X)$.

Prema autorima PS mere, dobra mera mora da zadovoljava osobine:

1. $M(A,\ B) = 0$ ako su A i B statistički nezavisne.
2. $M(A,\ B)$ se monotono povećava sa porastom $P(A,\ B)$ kada $P(A)$ i $P(B)$ ostaju nepromenjene.
3. $M(A,\ B)$ se monotono smanjuje sa porastom $P(A)$ kada $P(A,\ B)$ i $P(B)$ ostaju nepromenjene. Analogno za $P(B)$.

Osobine mera:

- Mera M je **simetrična** ako važi $M(A,\ B) = M(B,\ A)$
- Kako se mera ponaša prilikom skaliranja vrednosti u redu ili koloni?
- Kako se mera ponaša kod inverzija (npr. kod vektora binarnih vrednosti pralazak 0 u 1 i obrnuto)
- Kako se mera ponaša kada dodajemo prazne slogove? Prazni slogovi ne sadrže A i B.

Simpsonov paradoks: Posmatramo skup transakcija za kupovinu HDTV i trake za trčanje. Dobijamo da je veća šansa da kada korisik kupi HDTV, da će kupiti i traku za trčanje. Ako u analizu uključimo informaciju o tome da li je korisnik student ili zaposleni, za obe grupe dobijamo da je veća šansa da ako kupe HDTV, da neće kupiti traku za trčanje. Bez obzira na posmatranu meru, kada kombinujemo podatke (studenti i zaposleni), HDTV i trake za trčanje su pozitivno korelisani, a kada ih posmatramo odvojeno, oni su negativno korelisani.

Dodatne tehnike pravila pridruživanja

Prethodni algoritmi su primenljivi na podatke predstavljene u obliku asimetričnih binarnih atributa. Ako su podaci predstavljeni kategoričkim, neprekidnim ili simetričnim binarnim atributima, potrebne su određene modifikacije.

Kategorički atributi

Kategorički atributi se transformišu u asimetrične binarne atribute. Pri ovom postupku mogu nastati razni problemi. Jedan od njih su atributi sa velikim brojem mogućih vrednosti, što kao posledicu može da ima malu podršku. Moguće rešenje je da se neke vrednosti agregiraju. Drugi problem je kada raspodela vrednosti atributa jako odskače na jednu stranu. U tom slučaju možemo izbrisati stavke sa visokom učestalošću.

Neprekidni atributi

Postoje različite metode:

- **Metode zasnovane na diskretizaciji** - diskretizacija može biti **bez nadzora**. U tom slučaju možemo uzeti delove iste širine, delove iste dubine ili izvršiti klasterovanje vrednosti atributa. Diskretizacija **pod nadzorom** je diskretizacija koja pokušava da što bolje razdvoji klase na osnovu vrednosti atributa. Problem ovih metoda je to što veličina intervala utiče na pouzdanost i podršku. Ako je interval jako mali, podrška može biti nedovoljna, a ako je jako veliki, pouzdanost može biti nedovoljna. Jedno rešenje podrazumeva da isprobamo sve moguće intervale, ali to je neefikasno. Ako interval sadrži n vrednosti, u proseku postoji $O(n^2)$ mogućih uređenja.
- **Statistički zasnovane metode** - glava pravila je neka statistika neprekidne promenljive (sredina, medijana, ...). Prvo izdvajamo ciljni atribut. Generišemo česte stavke na osnovu ostalih atributa, a onda za svaku čestu stavku računamo statistiku ciljnog atributa. Za procenu zanimljivosti pravila koristimo neki statistički test. To radimo tako što poredimo statistiku dela populacije pokrivenog pravilom u odnosu na deo populacije koji nije pokriven pravilom. Za $A \rightarrow B$ dobijamo μ, a za $\overline{A} \rightarrow B$ dobijamo μ'. Nulta hipoteza podrazumeva da važi $\mu' = \mu + \Delta$, a alternativna da važi $\mu' > \mu + \Delta$. Koristi se Z statistika za dva uzorka. Ako je $Z > Z_0$, gde je Z_0 kritična vrednost za određeni nivo pouzdanosti, onda se nulta hipoteza odbacuje.
- **Metode zasnovane na nediskretizaciji** - zanimljivije je naći vezu između neprekidnih atributa, nego njihovih diskretnih intervala. Podrazumeva da podaci sadrže samo neprekidne atribute istog tipa. Na primer, na osnovu TF matrice možemo da zaključimo da li neke reči imaju tendenciju da se zajedno pojavljuju u istom dokumentu. Prvo se vektor reči normalizuje, tako da svaka reč ima podršku jednaku 1.

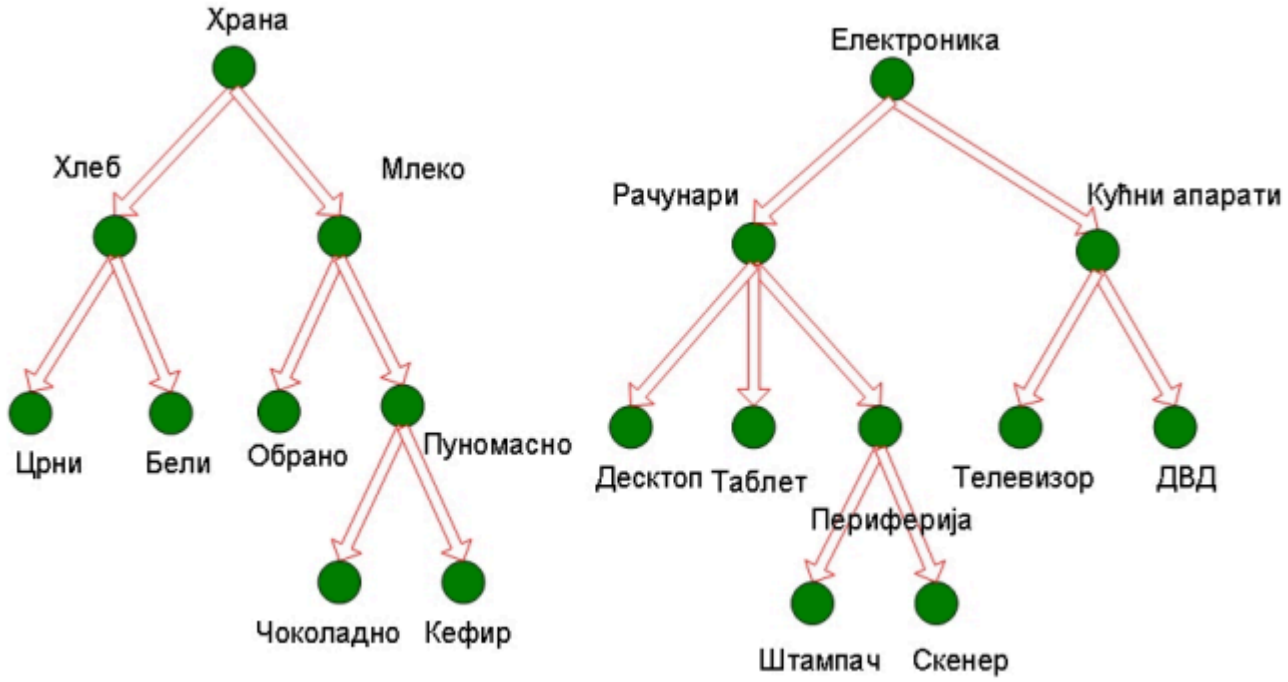
	W_1	W_2	W_3	W_4	W_5			W_1	W_2	W_3	W_4	W_5
D_1	2	2	0	0	1	$\longrightarrow$	D_1	0.40	0.33	0.00	0.00	0.17
D_2	0	0	1	2	2	$\longrightarrow$	D_2	0.00	0.00	0.33	1.00	0.33
D_3	2	3	0	0	0	$\longrightarrow$	D_3	0.40	0.50	0.00	0.00	0.00
D_4	0	0	1	0	1	$\longrightarrow$	D_4	0.00	0.00	0.33	0.00	0.17
D_5	1	1	1	0	2	$\longrightarrow$	D_5	0.20	0.17	0.33	0.00	0.33

Stavka je skup reči. Podrška u ovom slučaju predstavlja meru koliko su reči pridružene jedna drugoj. Podrška skupa reči C u skupu dokumenata T računa se kao

$$\sup(C) = \sum_{i \in T} \min_{j \in C} D(i,\ j)$$

Ova mera podrške naziva se **Min-Apriori** i ima osobine da podrška:
- monotono raste kako raste normalizovana frekvencija reči.
- monotono raste kako raste broj dokumenata koji sadrže reč.
- monotono opada kako raste broj reči u skupu stavki (antimonotonost).

Pravila pridruživanja između više nivoa



Hijerarhija se uključuje jer neka pravila na nižim nivoima možda nemaju dovoljnu podršku da se jave u čestim skupovima podataka. To se dešava jer ta pravila mogu biti previše specifična. Na primer, pravila obrano mleko $\rightarrow$ beli hleb, punomasno mleko $\rightarrow$ crni hleb i kefir $\rightarrow$ crni hleb, mogu biti indikatori pravila pridruživanja između mleka i hleba. Ako se obilazi **drvo hijerarhije koncepata**, tada važi:

- Ako je X roditelj stavka za X_1 i X_2, tada važi $\sup(X) \geq \sup(X_1) + \sup(X_2)$.
- Ako je $\sup(X_1 \cup Y_1) \geq minsup$ i X je roditelj od X_1 i Y je roditelj od Y_1, tada važi i $\sup(X \cup Y_1) \geq minsup$, $\sup(X_1 \cup Y) \geq minsup$ i $\sup(X \cup Y) \geq minsup$.
- Ako je $conf(X_1 \rightarrow Y_1) \geq minconf$, tada važi $conf(X_1 \rightarrow Y) \geq minconf$.

Pravila pridruživanja možemo proširivati na dva načina:

1. Tekuće pravilo proširujemo stavkom višeg nivoa. Na primer, transakciju {obrano mleko, beli hleb} proširimo na {obrano mleko, beli hleb, mleko, hleb, hrana}. Problem kod ovog pristupa je što stavke višeg nivoa imaju mnogo veći nivo podrške od onih na manjem nivou. Druga mana je što se povećava dimenzionalnost podataka.
2. Česte obrasce prvo formiramo na najvišem nivou, a onda redom i na sledećim nivoima. Mana ovog pristupa je to što se UI zahtevi drastično povećavaju, zbog potrebe višestrukog prolaženja kroz podatke, a moguće je i izgubiti neke potencijalno zanimljive veze između obrazaca na različitim nivoima.

Istraživanje sekvencijalnih obrazaca

Niz podataka podrazumeva da imamo podatke koji su poređani u niz. Na primer, za svakog kupca imamo transakcije, ali i vreme obavljanja tih transakcija. Još neki primeri su podaci o nekim događajima, DNK niske vrsta, pregledanje aktivnosti posetilaca Veba i slično.

Niska je uređena lista **elemenata** ($S = \ < e_1e_2e_3... >$). Predstavlja jednu transakciju. Svaki element sadrži skup događaja, tj. stavki ($e_i = \{i_1i_2...i_k\}$). Svakom elementu se dodeljuje i određeno vreme ili mesto. Broj elemenata u nisci s određuje **dužinu niske** ($|s|$). k**-niska** je niska koja sadrži k stavki. Niska $a = \ < a_1a_2...a_n >$ je sadržana u nisci $b = \ < b_1b_2...b_m >$, $m \geq n$, ako postoje celi brojevi $i_1 < i_2 <... < i_n$ takvi da važi $a_1 \subset b_{i_1}$, $a_2 \subset b_{i_2}$, ..., $a_n \subset b_{i_n}$. Kažemo da je niska a **podniska** niske b.

Istraživanje sekvencijalnih obrazaca podrazumeva nalaženje svih čestih podniski u nekoj bazi sa niskama. Podniska je **česta** ako je njena podrška veća od minimalne, korisnički definisane podrške. **Podrška niske** w je količnik broja niski koje sadrže w u odnosu na ukupan broj niski. Imamo veliki broj mogućih podniski. Na primer, ako imamo n događaja i_1, ..., i_n, kandidati su:

- 1-podniske: $< \{i_1\} >$, $< \{i_2\} >$, ..., $< \{i_n\} >$
- 2-podniske: $< \{i_1,\ i_2\} >$, $< \{i_1,\ i_3\} >$, ..., $< \{i_1,\ i_1\} >$, $< \{i_1\}\ \{i_2\} >$, ..., $< \{i_{n-1}\}\ \{i_n\} >$
- 3-podniske: $< \{i_1,\ i_2,\ i_3\} >$, $< \{i_1,\ i_2,\ i_4\} >$, ..., $< \{i_1,\ i_2\}\ \{i_1\} >$, $< \{i_1\}\ \{i_2\}\ \{i_2\} >$, ..., $< \{i_1\}\ \{i_1,\ i_2\} >$, $< \{i_1\}\ \{i_1,\ i_3\} >$, ..., $< \{i_1\}\ \{i_1\}\ \{i_1\} >$, $< \{i_1\}\ \{i_1\}\ \{i_2\} >$, ...

GSP (Generalized Sequential Patterns) algoritam:

1. Iz baze niski D formiramo sve česte 1-podniske.
2. Ponavljamo postupak dok god ima novih čestih podniski:
   - Formiranje kandidata: spajamo parove čestih podniski nađenih u $(k-1)$-om prolazu radi formiranja kandidatskih niski koje sadrže k stavki.
   - Potkresivanje spiska kandidata: potkresamo skup kandidatskih k-niski koje sadrže retke $(k-1)$-podniske.
   - Izračunavanje podrške: napravimo novi prolaz kroz bazu niski D i računamo podršku za preostale kandidatske niske.

- Uklanjanje kandidata: uklanjamo kandidatske k-niske čija je podrška manja od minimalne definisane podrške.

Česta $(k-1)$-niska w_1 se spaja sa drugom čestom $(k-1)$-niskom w_2 i formira se kandidatska k-niska ako je podniska dobijena uklanjanjem prvog događaja iz w_1 ista kao i podniska dobijena uklanjanjem poslednjeg događaja iz w_2. Rezultujuća niska se dobija proširivanjem niske w_1 poslednjim događajem niske w_2. Ako poslednja dva događaja iz w_2 pripadaju istom elementu, tada poslednji događaj iz w_2 postaje deo poslednjeg elementa u w_1. U suprotnom, poslednji događaj iz w_2 postaje poseban element dodat na kraj w_1.

Kao jedan od uslova da bi niska bila česta, može se postaviti **vremensko ograničenje** u kome se niska pojavljuje. **Maksimalni razmak** je maksimalni dozvoljeni razmak između prvog i poslednjeg događaja jedne niske. **Minimalni jaz** je minimalni dozvoljeni razmak između susednih elemenata. **Maksimalni jaz** je maksimalni dozvoljeni razmak između dva susedna elementa. Postoje dva pristupa za nalaženje čestih niski sa vremenskim ograničenjima:

1. Primenimo algoritam za istraživanje sekvencijalnih obrazaca bez vremenskih ograničenja, a onda naknadno obrađujemo pronađene obrasce.
2. Modifikujemo prethodne algoritme tako da direktno potkresuju kandidate koji krše vremenska ograničenja. U ovom slučaju, maksimalni jaz može da naruši Apriori princip.

Važenje Apriori principa se prevazilazi uvođenjem koncepta neprekidnih podniski. Niska s je **neprekidna podniska** od $w = \ < e_1 e_2 \dots e_k >$ ako važi:

- s je dobijeno iz w brisanjem događaja ili iz e_1 ili iz e_k ili
- s je dobijeno iz w brisanjem događaja iz nekog elementa e_i koji sadrži najmanje dva događaja ili
- s je neprekidna podniska od t i t je neprekidna podniska od w (rekurzivna definicija)

Modifikovani Apriori princip: Ako je k-niska česta, tada su i sve njene neprekidne $(k-1)$-podniske česte. Sada, pri istraživanju sekvencijalnih obrazaca potkresujemo kandidatske niske:

- bez ograničenja na veličinu maksimalnog jaza - razmatramo sve $(k-1)$-podniske i kandidatska k-niska se potkresuje ako je bar jedna od njenih $(k-1)$-podniski retka.
- sa ograničenjem na veličinu maksimalnog jaza - razmatraju se samo neprekidne $(k-1)$ podniske i kandidatska k-niska se potkresuje ako je bar jedna neprekidna $(k-1)$-podniska retka.

Još jedno dodatno ograničenje može biti **veličina prozora** koja se definiše kao najveći dozvoljeni vremenski razmak između prvog i poslednjeg pojavljivanja događaja u elementima sekvencijalnog obrasca. Prozor veličine 0 označava da se svi događaji u istom elementu dešavaju istovremeno.

Prebrojavanje koliko puta se podniska sadrži u niski može se vršiti na više načina:

- COBJ - broji se samo jedno pojavljivanje, tj. podniska se ili sadrži ili se ne sadrži u niski.
- CWIN - jedno pojavljivanje po pomičnom prozoru zadate veličine.
- CMINWIN - broj najmanjih prozora u kojima se podniska pojavljuje.
- CDIST_O - broje se sva pojavljivanja, sa mogućnošću vremenskog preklapanja.
- CDIST - sva pojavljivanja, ali bez vremenskog preklapanja.

Retki i negativni obrasci

Redak obrazac je skup stavki ili pravilo čija je podrška manja od zadatog praga $minsup$. Na primer, istovremeno upisivanje izbornih predmeta Analiza 4, Algebra 2 i Diskretne strukture 3.

Neka je $I = \{i_1, \ldots, i_d\}$ skup stavki. **Negativna stavka** $\overline{i_k}$ označava odsustvo stavke i_k iz date transakcije. **Negativan skup stavki** X je skup stavki koji ima sledeće osobine:

- $X = A \cup \overline{B}$, gde je A skup pozitivnih stavki, a $\overline{B}$ skup negativnih stavki, pri čemu je $|\overline{B}| \geq 1$.
- $\sup(X) \geq minsup$

Negativno pravilo pridruživnja je pravilo pridruživanja koje ime sledeće osobine:

- Pravilo je izdvojeno iz negativnog skupa stavki.
- Podrška pravila je $\geq minsup$.
- Pouzdanost pravila je $\geq minconf$.

Skup stavki $X = \{x_1, \ldots, x_k\}$ je **negativno korelisan** ako

$$\sup(X) < \Pi_{j=1}^{k} \sup(x_j)$$

Pravilo pridruživanja $X \to Y$ je **negativno korelisano** ako

$$\sup(X \cup Y) < \sup(X)\sup(Y),$$

gde su X i Y disjunktni skupovi stavki. Pun uslov:

$$\sup(X \cup Y) < \Pi_j \sup(X_j) \cdot \Pi_j \sup(Y_j)$$

Ako su stavke X i Y međusobno pozitivno korelisane, praktičnije je koristiti delimični uslov umesto punog uslova.

Mnogi retki obrasci imaju odgovarajuće negativne obrasce. Mnogi negativno korelisani obrasci takođe imaju odgovarajuće negativne obrasce. Što je manja podrška za $X \cup Y$, obrazac je više negativno korelisan. Retki negativno korelisani obrasci su interesantniji od čestih negativno korelisanih obrazaca. Ako su X i Y negativno korelisani, tada $X \cup \overline{Y}$, $\overline{X} \cup Y$ ili oba moraju da imaju relativno visoku podršku.

Tehnike za istraživanje negativnih obrazaca:

- **Tehnika zasnovana na simetričnim binarnim promenljivim**

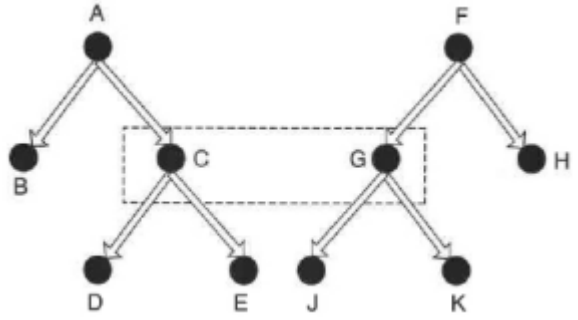
TID	Items
1	{A,B}
2	{A,B,C}
3	{C}
4	{B,C}
5	{B,D}

Original Transactions

TID	A	$\overline{A}$	B	B	C	$\overline{C}$	D	D
1	1	0	1	0	0	1	0	1
2	1	0	1	0	1	0	0	1
3	0	1	0	1	1	0	0	1
4	0	1	1	0	1	0	0	1
5	0	1	1	0	0	1	1	0

Transactions with Negative Items

  Problem ovog pristupa je veliki broj stavki, jer umesto rešetke sa 2^d stavki imamo rešetku sa 2^{2d} stavki, kao i to što se dužina transakcije povećava kada se uključe negativne stavke. Takođe, potkresivanje na osnovu podrše ne može da se primeni jer za svaku stavku ili x ili $\overline{x}$ ima podršku veću od 50%.
- **Tehnike zasnovane na hijerarhiji koncepata**



  Na primer, ako je stavka $\{C,\ G\}$ česta, a $\{D,\ J\}$ nije, tada D i J formiraju interesantan obrazac.
- **Tehnike zasnovane na indirektnom pridruživanju** - par stavki a i b je indirektno pridružen preko medijatora Y ako važi
  - $\sup(\{a,\ b\}) < t_s$
  - $\exists\, Y \neq \emptyset$ tako da:
    - $\sup(\{a\} \cup Y) \geq t_f$ i $\sup(\{b\} \cup Y) \geq t_f$
    - $d(\{a\} \cup Y) \geq t_d$ i $d(\{b\} \cup Y) \geq t_d$, gde je d mera pridruživanja.

Klasifikacija pomoću pravila pridruživanja

Pravila pridruživanja se koriste za klasifikaciju tako što se koristi skup pravila sa velikom pouzdanošću, pri čemu se u desnom delu pravila nalazi samo jedna stavka koja predstavlja klasu koja se predviđa, a stavke u levom delu pravila čine atribute na osnovu kojih se vrši predviđanje.