

# **OSNOVI GEOINFORMATIKE**

Skripta 2022./2023.

<b>Šta je frontend i koje tehnologije se koriste za njegovu realizaciju?</b>	<b>5</b>
<b>Šta je backend i koje tehnologije se koriste za njegovu realizaciju?</b>	<b>5</b>
<b>Troslojna arhitektura za razvoj geoportala.</b>	<b>5</b>
<b>Šta je GIS?</b>	<b>6</b>
<b>Navesti geografske informacione tehnologije.</b>	<b>6</b>
<b>Šta GIS omogućava?</b>	<b>6</b>
<b>Oblasti primene GIS-a.</b>	<b>7</b>
<b>Komponente GIS-a.</b>	<b>7</b>
<b>Softverska arhitektura GIS-a.</b>	<b>7</b>
<b>Tipovi prostornih fenomena.</b>	<b>7</b>
<b>Vektorski formati podataka.</b>	<b>8</b>
<b>Rasterski formati podataka.</b>	<b>9</b>
<b>Softversko inženjerstvo i softverski proces.</b>	<b>10</b>
<b>Tipovi GIS softvera i njihova primena.</b>	<b>10</b>
Desktop GIS	10
Sistemi za upravljanje prostornim bazama podataka.	10
WebMap serveri	10
Serverski GIS	10
WebGIS Clients	11
Biblioteke i ekstenzije	11
Mobile GIS	11
<b>Primeri tehnologija i alata za različite tipove GIS softvera.</b>	<b>11</b>
<b>Analiza udaljenosti (Proximity Analysis).</b>	<b>11</b>
<b>GeoWorkflow kroz ArcGIS Model Builder ili QGIS Graphical Modeler.</b>	<b>12</b>
<b>Izbor lokacije (Site Selection).</b>	<b>13</b>
<b>Mrežna analiza (Network analysis).</b>	<b>13</b>
<b>Šta je topologija? Vrste topologije.</b>	<b>14</b>
<b>Šta je triangulacija, kako se formira i gde se primjenjuje?</b>	<b>15</b>
<b>Tehnike analize terena (analiza nagiba, analiza aspekta (pravca nagiba, Hillshades, Viewshed)).</b>	<b>16</b>
Analiza nagiba.	16
Analiza pravca nagiba.	16
Hillshades.	17
Viewshed.	18
<b>Šta je map algebra?</b>	<b>18</b>
<b>Mapiranje gustine (Heatmap).</b>	<b>18</b>
<b>Šta je geostatistika i gde se koristi?</b>	<b>19</b>
<b>Objasniti IDW i Kriging i navesti primene.</b>	<b>20</b>

IDW - Inverse Distance Weighting	20
Kriging	20
<b>Objasniti AHP metodu.</b>	<b>20</b>
<b>Šta podrazumeva modelovanje podataka?</b>	<b>21</b>
<b>Šta je UML?</b>	<b>21</b>
<b>Polazni koncepti UML.</b>	<b>22</b>
<b>Tipovi UML dijagrama.</b>	<b>22</b>
Dijagrami strukture	22
Dijagram klasa	22
Dijagram objekata	23
Dijagram komponenti	23
Dijagram paketa	23
Dijagram rasporeda	23
Dijagram kompozitne strukture	23
Dijagrami ponašanja	23
Dijagram slučajeva korišćenja	23
Dijagram aktivnosti	23
Dijagram stanja	23
Dijagram poruka	23
Dijagram sekvenci	23
Dijagram vremena	24
Dijagram interakcije	24
<b>Objasniti dijagram klasa.</b>	<b>24</b>
<b>Tipovi veza između klasa.</b>	<b>26</b>
Asocijacija	26
Agregacija	26
Kompozicija	27
<b>Razlika između agregacije i kompozicije.</b>	<b>27</b>
<b>ISO 19107 standard.</b>	<b>27</b>
<b>Šta je topologija?</b>	<b>28</b>
<b>Geometrijski tipovi prema ISO19107.</b>	<b>28</b>
<b>Topološki tipovi prema ISO19107.</b>	<b>28</b>
<b>Objasniti relacioni model podataka.</b>	<b>29</b>
<b>Objasniti objektno-relacioni model podataka.</b>	<b>30</b>
<b>Navesti i objasniti SQL naredbe.</b>	<b>31</b>
<b>Šta je SUPB i koje su mu prednosti i karakteristike.</b>	<b>31</b>
<b>Upravljanje transakcijama u SUBP.</b>	<b>32</b>
<b>Struktura SUPB.</b>	<b>33</b>
<b>Šta su geoprostorne baze podataka.</b>	<b>33</b>
<b>Navesti primere upotrebe GPB.</b>	<b>34</b>

<b>Prostorni odnosi u GBP.</b>	34
<b>Primarni i sekundarni filter kod selekcije podataka i primeri.</b>	36
<b>Šta je mašinsko učenje?</b>	37
<b>Šta je duboko učenje?</b>	38
<b>Šta je Geo AI? Objasniti vezu između GIS-a i AI.</b>	39
<b>Navesti primere primene geoprostornog mašinskog i dubokog učenja.</b>	40
<b>Šta je AI model i objasniti Geo AI workflow.</b>	40
<b>Prvi korak je prikupljanje i priprema geoprostornih podataka. To može uključivati prikupljanje podataka iz različitih izvora, čišćenje podataka, transformaciju formata i standardizaciju podataka.</b>	41
<b>Navesti alate i programske pakete/jezike koji se koriste za Geo AI.</b>	41
<b>Šta je učenje sa i bez nadzora (supervised i unsupervised) i navesti i objasniti neke algoritme iz svake grupe.</b>	41
<b>Navesti osnovne grupe i primere ML i DL algoritama.</b>	42
<b>Šta je regresija, kakve vrste regresije postoje i gde se primjenjuje?</b>	44
<b>Šta je klasifikacija i gde se primjenjuje? Navesti neke osnovne algoritme za klasifikaciju.</b>	45
<b>Šta je klastering i gde se primjenjuje? Navesti neke osnovne algoritme za klastering?</b>	47
<b>Šta je PCA analiza i za šta se koristi?</b>	48
<b>Šta je scikit-learn?</b>	49
<b>Objasniti regresiju u ArcGIS okruženju upotrebom OLS metode.</b>	50
<b>Šta je kompjuterski vid i gde se koristi?</b>	54
<b>Objasniti klasifikaciju slike i detekciju objekata.</b>	55
<b>Šta je semantička segmentacija, a šta segmentacija instanci?</b>	57
<b>Šta je feature extraction?</b>	58
<b>Objasniti Hot Spot analizu i Density-based klastering.</b>	59
<b>Objasniti F-score metriku za evaluaciju AI modela.</b>	60
<b>Šta je klijent-server arhitektura i kako se vrši komunikacija između klijenta i servera?</b>	61
<b>Šta je servisno-orjentisana arhitektura?</b>	61
<b>Šta je interoperabilnost?</b>	62
<b>Objasniti publish-find-bind model interakcije u SOA.</b>	62
<b>Šta su WSDL i REST servisi i u čemu je razlika?</b>	64
<b>Objasniti REST servise.</b>	64
<b>Objasniti JSON i GeoJSON format podataka.</b>	65
<b>Objasniti troslojnu arhitekturu i navesti primere tehnologija/softvera/programskih jezika koje se koriste za realizaciju svakog sloja.</b>	65
<b>Šta su geoservisi i objasniti OGC WMS, WFS i WCS.</b>	66
<b>Navesti vrste geoprostornih servisa.</b>	67

<b>Šta je TypeScript? TypeScript vs JavaScript.</b>	67
<b>TypeScript osnovni tipovi, primitivni i objektni.</b>	68
<b>Anotacije tipova i zaključivanje tipova.</b>	69
<b>Naredbe za kontrolu toka izvršavanja u TypeScript-u.</b>	70
<b>TypeScript funkcije.</b>	72
<b>TypeScript klase, atributi, metode, vidljivost, konstruktor...</b>	74
<b>Nasleđivanje, interfejsi, apstraktne klase, statički atributi i metode.</b>	77
<b>Objasniti type casting u TypeScript-u.</b>	79
<b>TypeScript generici, generičke klase i interfejsi.</b>	79
<b>TypeScript moduli, import i export.</b>	81
<b>NodeJS i TypeScript. Kojom ključnom reči se zadaju komande u NodeJS? Šta su NodeJS moduli?</b>	82
<b>Šta je Angular, a šta Angular CLI?</b>	83
<b>Komponente u angularu kontrolišu deo ekrana koji se naziva pogled (view). U komponentama se definiše njena aplikativna logika – šta ona radi da podrži pogled (view) – unutar klase.</b>	83
<b>Navesti komande u Angular okruženju za kreiranje projekta, komponente i servisa, pokretanje projekta, building projekta...</b>	84
<b>Za šta se koristi app-routing-module?</b>	85
<b>Data binding u Angular-u i vrste data binding-a.</b>	85
<b>Šta je OpenLayers i kako se prave mape i layer-i?</b>	85
<b>Šta su callback funkcije?</b>	86
<b>Šta je Promise objekat?</b>	86
<b>Šta je FetchAPI i gde se koristi?</b>	86
<b>Objasniti async-await.</b>	87
<b>RxJS Observable, razlika u odnosu na Promise.</b>	87
<b>Šta je CRUD?</b>	88
<b>Šta je Angular Material?</b>	88
<b>Šta je Spring Boot okruženje i kako se kreiraju rest servisi u njemu?</b>	89
<b>Šta je Maven?</b>	90

## Šta je *frontend* i koje tehnologije se koriste za njegovu realizaciju?

**Frontend** je frontalni dio aplikacije - dio koji interreaguje sa korisnikom. Primarni cilj *frontend*-a jeste da učini servis što pristupačnijim i lakšim za korišćenje - izgled aplikacije (*web* sajta), mogućnost lakog kretanja po aplikaciji (*web* sajtu) i sl.

Naziva se i **klijentski sloj**.

Tehnologije koje se najčešće koriste pri razvoju *frontend* dijela aplikacije su **HTML, CSS, JavaScript**.

## Šta je *backend* i koje tehnologije se koriste za njegovu realizaciju?

**Backend** dio aplikacije odnosi se na sve ono što se dešava *iza* frontalnog dijela aplikacije, a to je **obrada zahtjeva** primljenih od klijentske strane, **komunikacija sa slojem podataka (bazom podataka)** te **servisiranje rezultata obrađenih zahtjeva klijentima**.

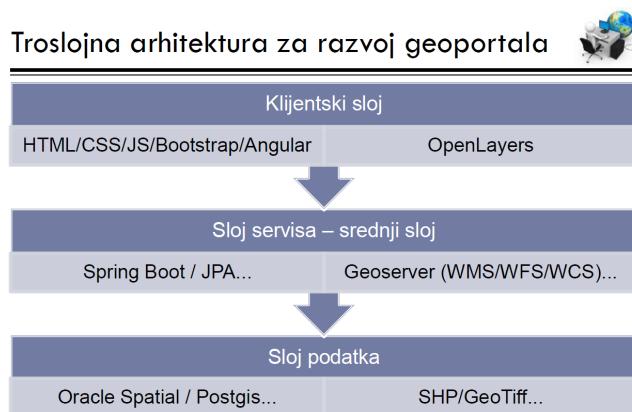
Naziva se i **serverski sloj**.

Tehnologije koje se najčešće koriste pri razvoju *backend* dijela aplikacije su *Java, .Net, Python,...*

## Troslojna arhitektura za razvoj geoportala.

Troslojna arhitektura za razvoj geoportala sastoji se od:

- **Frontend sloja** - opisano gore.
- **Backend sloja** - opisano gore.
- **Sloja podataka** - sloj koji se sastoji od baze podataka i komunicira sa serverskim dijelom radi usluživanja zahtjeva. U ovom sloju su smješteni podaci vezani za dati informacioni sistem (baza podataka - *katastar* i sl.).



## Šta je GIS?

**Geografski informacioni sistemi** su specijalizovani sistemi koji prate ne samo pojave, aktivnosti i događaje, već i gdje se takve pojave, aktivnosti i događaji nalaze. Na taj način, geografska lokacija ili ono gdje se nešto nalazi postaje bitan atribut aktivnosti, rukovođenja, strategija, planova i odluka.

Izvori ulaznih podataka za geoinformacione sisteme su:

- **Globalni sistem pozicioniranja (GPS)** - sistem satelita koji mogu obezbjediti preciznu lokaciju na Zemljinoj površini.
- **Remote sensing** - korišćenje satelita ili aviona za prikupljanje informacija o površini Zemlje.

U geoinformacionim sistemima, karte su povezane sa tabelama alfanumeričkih podataka, tako da **kombinacija karata** zapravo označava **kombinaciju podataka** da bismo dobili željene informacije. Geoinformacioni sistemi omogućuju i postavljanje upita sa mape, odnosno iz baze podataka i sl.

Primjena geoinformacionih sistema je raznorodna:

- 80% aktivnosti samouprave se bazira na prostornim podacima.
- Značajan dio državne uprave ima geografsku komponentu.
- Poslovne firme koriste geoinformacione sisteme za veoma širok spektar primjena, kao što su precizna poljoprivreda, istraživanje prirodnih resursa, urbanizam i sl.
- Izvođenje složenog prostornog modelovanja - planiranje katastrofa, upravljanje resursima i sl.

## Navesti geografske informacione tehnologije.

- **Globalni sistem pozicioniranja (GPS)** - sistem satelita koji mogu obezbjediti preciznu lokaciju na Zemljinoj površini.
- **Remote sensing** - korišćenje satelita ili aviona za prikupljanje informacija o površini Zemlje.
- **Geografski informacioni sistemi (GIS)** - imaju mogućnost unosa, skladištenja, manipulaciju i preuzimanje geografskih informacija.

## Šta GIS omogućava?

**GIS aplikacije** omogućavaju automatizaciju aktivnosti koje uključuju geografske podatke:

- Prozvodnja mapa.
- Računanje površine, dužine ruta,...
- Logistika - planiranje rute, upravljanje saobraćajem,...

Vezivanjem podataka za mape, dopušta se komunikacija složenih prostornih obrazaca - daju odgovor na prostorne upite. Takođe, izvode složeno prostorno modelovanje - planiranje katastrofa, upravljanje resursima i sl.

## Oblasti primene GIS-a.

**Primjena geoinformacionih sistema je raznorodna:**

- 80% aktivnosti samouprave se bazira na prostornim podacima.
- Značajan dio državne uprave ima geografsku komponentu.
- Poslovne firme koriste geoinformacione sisteme za veoma širok spektar primjena, kao što su precizna poljoprivreda, istraživanje prirodnih resursa, urbanizam i sl.
- Izvođenje složenog prostornog modelovanja - planiranje katastrofa, upravljanje resursima i sl.

## Komponente GIS-a.

Geografski informacioni sistemi sadrže **četiri** važne komponente:

- **Kompjuterski hardver.**
- **Niz aplikacionih softverskih modula.**
- **Odgovarajući organizacioni sadržaj koji obuhvata obučene ljudе.**
- **Podatke.**

Takođe, sadrže i **četiri interaktivne komponente**:

- **Podsistem za unos** - vrši konverziju karata i drugih prostornih podataka u digitalni oblik - ***data input***. Takođe, moguće je unijeti i **korisničke upite** - ***query input***.
- **Podsistem za skladištenje i pozivanje podataka** - ***geographic database***.
- **Podsistem za analizu i transformaciju.**
- **Izlazni podsistem za izradu karata, tabela i pružanje odgovora na postavljene upite.**

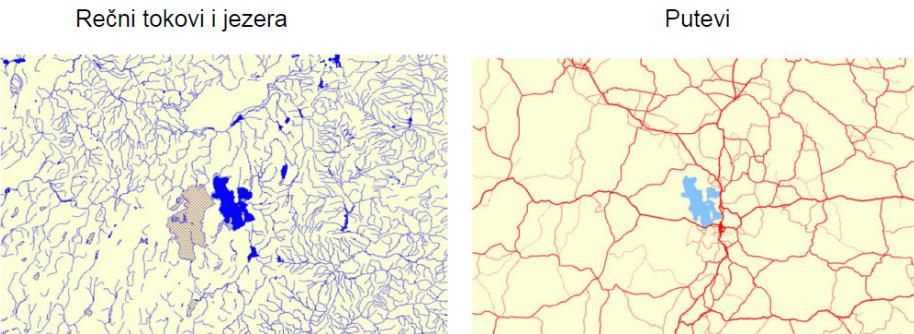
## Softverska arhitektura GIS-a.

Pitanje iznad (**četiri interaktivne komponente**).

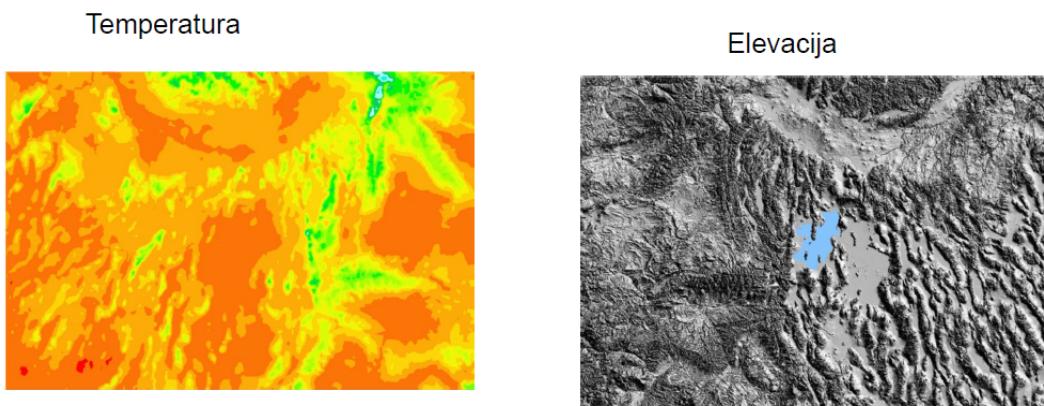
## Tipovi prostornih fenomena.

Potreba za smještanjem fenomena realnog svijeta u model geoprostornih podataka iziskuje poznavanje strukture tih fenomena. Vrši se podjela na dvije vrste:

- **Diskretni fenomeni** - entiteti koji postoje ali se individualno izdvajaju od drugih fenomena sa jasno definisanim granicama. Primjeri su **rijeke, jezera, putevi** i sl.



- **Kontinualni fenomeni** - kontinualni podaci koji se ne mogu izdvojiti kao individualni. Primjeri su **temperatura, elevacija** i sl.



## Vektorski formati podataka.

**Vektorski model podataka** definiše diskretne objekte -  **putevi, jezera, katastarske parcele, zgrade** i sl.

Postoje tri tipa vektorskog modela podataka (svaki tip nosi sa sobom **satelitske** (dodatne) podatke):

- **Tačke** - koriste jedan par koordinata da opišu svoju poziciju. Nemaju dimenziju iako objekti predstavljeni njima u realnom svijetu imaju. Primjeri su **ulične svjetiljke, šahovi** i sl.
- **Linije** - određene setom koordinata, gdje je svaka linija (kriva) određena mnoštvom linijskih segmenata.

- Opisuju se **čvorovima i verteksima**, gdje su čvorovi mesta gdje linija počinje i završava se, dok su verteksi tačke gdje linija mijenja smjer.
- Atributi mogu biti vezani za cijelu liniju ali i za pojedinačnu tačku linije ili linijski segment, te tako jedna linija može imati više redova u tabeli atributa.
- Primjeri su  **putevi, cijevi, elektroenergetski kablovi** i sl.
- **Poligoni** - formiraju se kao set linijskih segmenta u kojem je početna tačka ista kao i krajnja. Poligoni tako predstavljaju **zatvorene forme koje imaju unutrašnje regije**.
  - Atributi su vezani za **centar poligona** bez obzira na to koliko on složen bio.
  - Primjeri su **jezera, gradovi, šume** i sl.

Podaci su tako struktuirani na sledeći način:

- **Prostorni objekat (Feature** - pr. opština)
- **Neprostorni atributi (tabelarni podaci** - pr. naziv opštine, površina, tip plana,...)
- **Prostorni atributi (geometrija** - pr. poligon)

Vektorski podaci se smještaju u **shapefile**-ove.

## Rasterski formati podataka.

**Rasterskim formatom podataka** opisuje se prostorni objekat skupom **pixel-a**, gdje se tako najbolje reprezentuju **kontinualni objekti (temperatura, elevacija i sl.)**.

Može imati oblik seta **ćelija** kao što su **pixel-i** na fotografiji ili da ćelije budu organizovane u **grid pattern (matricu)**.

Svaka ćelija sadrži vrijednost i opisana je dimenzijom (**visinom i širinom**).

Rasteri se smještaju u različite formate datoteka:

- **TIFF**
  - Jedan od najpopularnijih rasterskih formata u vidu **lossless kompresije**.
  - **GeoTIFF** - verzija **TIFF** formata sa ugrađenim geografskim podacima u okviru **tagova TIFF fajla**.
  - Za smještanje i prenos digitalnih satelitskih snimaka, modela terena, aero snimaka, skeniranih mapa i sl.
- **JPEG**
  - **Lossy** tehniku **kompresije**, gdje se veličina fajla redukuje na otprilike 5% od originalne verzije, ali se prilikom kompresije gube detalji.
  - **JGW (JPEG World File)** - koristi se da **georeferencira JPEG** fajl. Sadrži informacije o koordinatama i omogućuje da **JPEG** fajl bude ispravno pozicioniran na mapi.
- **World File**

- Tekstualni fajl kojeg koriste geoinformacioni sistemi za georeferenciranje rastera. Opisuje lokaciju, razmjeru i rotaciju mape.

## Softversko inženjerstvo i softverski proces.

Odnosi se na pitanje dole.

## Tipovi GIS softvera i njihova primena.

### Desktop GIS

Koristi se za kreiranje, uređivanje, upravljanje, analizu i prikaz geografskih podataka.  
Klasificuje se u tri kategorije funkcionalnosti:

- *GIS Viewer*
- *GIS Editor*
- *GIS Analyst*

Primjeri *Desktop GIS* rješenja su **QGIS**, **GRASS GIS**, **MapWindow GIS**, **Autodesk ERDAS, ERSI**,...

### Sistemi za upravljanje prostornim bazama podataka.

Koriste se za čuvanje podataka, ali pružaju i funkcionalnost analize i manipulacije podacima. Primjeri su **PostGIS**, **MySQL Spatial**,...

### WebMap serveri

Za distribuciju mapa preko interneta (**Open Geospatial Consortium**):

- **Web Map Service** - skup specifikacija interfejsa koji daju uniforman pristup od strane web klijenata mapama renderovanim na *map* serveru na internetu.
- **Web Feature Service** - web servis koji dopušta korisniku da objavi geoprostorne objekte na internetu zajedno sa definicijom njihove strukture.
- **Web Coverage Service** - podržava elektronsku razmjenu geoprostornih podataka u formi *coverage-a*, koji je definisan kao digitalna geoprostorna informacija koja predstavlja fenomen koji varira u prostoru.

Primjer rješenja su **GeoServer**, **MapServer**,...

### Serverski GIS

Serverski orijentisan geoinformacioni sistem koji u osnovi pruža istu funkcionalnost kao i *Desktop GIS* rješenja, ali omogućava pristup ovoj funkcionalnosti putem mreža.

## **WebGIS Clients**

Predstavljaju klijentske aplikacije koje se koriste za prikaz podataka i pristupaju funkcijama analize i upita na osnovu serverskog geoinformacionog sistema preko interneta ili intraneta.

Dvije vrste klijenata:

**Thin klijenti** - web browser koji se koristi za prikazivanje Google mapa. Pružaju samo funkciju **prikaza i upita**.

**Thick klijenti** - obezbeđuju dodatne alate za uređivanje podataka, analizu i prikazivanje. Primjeri su *Google Earth*, *Desktop G/S*,...

## **Biblioteke i ekstenzije**

Pružaju dodatne funkcionalnosti koje nisu dio osnovnog GIS softvera. Mogu pokriti alate za analizu terena, čitanje određenih formata podataka ili alate za kartografski prikaz geografskih podataka.

Primjeri su *OpenLayers* (to je open-source AJAX biblioteka za pristup slojevima prostornih podataka), *GeoDjango*, *MapFish*, *OpenMap*,...

## **Mobile GIS**

Aplikacije za mobilne uređaje koje se koriste za prikupljanje podataka na terenu. To je integrisano hardversko i softversko okruženje za pristup geoprostornim podacima i servisima preko mobilnih uređaja kao što su **mobilni telefoni**, **tablet**i i sl. Omogućavaju ažuriranje podataka u realnom vremenu.

## **Primeri tehnologija i alata za različite tipove GIS softvera.**

Navedeno gore kroz tipove GIS softvera.

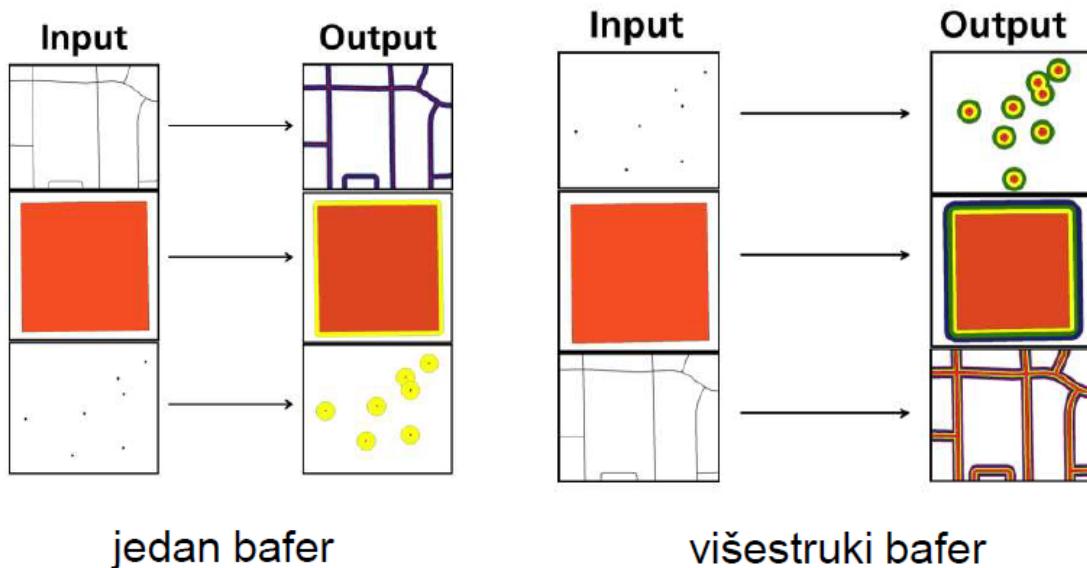
### **Analiza udaljenosti (*Proximity Analysis*).**

Predstavlja tip **vektorske analize** - skup alata koji se koriste za analiziranje relacija između izabranih geoprostornih objekata i njihovih susjeda.

Odgovara na pitanja kao što su **koliko je udaljena najbliža benzinska stanica**, **postoji li automehaničar u blizini do 1km**,...

Alati za udaljenost obuhvataju alate kao što su:

- **Buffer** - jedan bafer oko geo-objekta na specificiranoj udaljenosti od zadatog geo-objekta.
- **Multiple Rings Buffer** - višestruki bafer oko geo-objekta u vidu višestrukih prstenova na nekoliko zadatih udaljenosti od zadatog geo-objekta.



## **GeoWorkflow kroz ArcGIS Model Builder ili QGIS Graphical Modeler.**

**Workflow** predstavlja model koji opisuje ponovljivu sekvencu operacija i u računarstvu se koristi da predstavi interakciju između korisnika i računara. Komponente poslovnog procesa se mogu u osnovi definisati sa tri parametra:

- **ulazi**
- **transformaciona pravila - algoritmi**
- **izlazi**, koji predstavljaju **ulaze za naredne korake**

**Geoprostorni poslovni proces (GeoWorkflow)** predstavlja skup geoprostornih procesa sa kontrolnom strukturom baziranim na ponašanju ulaza i izlaza koji predstavljaju znanje eksperta iz domena. Svi procesi u geoprostornom modelu su organizovani sekvencijalno ili konkurentno, gdje je za svaki par susjednih procesa izvršenje prvog neophodno za izvršenje drugog.

**ModelBuilder** je alat u **ArcGIS**-u koji omogućava kreiranje, izmjenu i upravljanje geoprostornim modelima. On čuva *workflow* koji se može pokretati više puta i ulančava nekoliko alata, omogućavajući da izlaz jednog alata bude ulaz u sledeći alat. Pravljenje modela omogućava benefit od automatizacije procesa, jednostavnog čuvanja, dijeljenja i ponovnog pokretanja modela.

## Izbor lokacije (*Site Selection*).

Postupak kojim se mjeri potreba projekta u odnosu na veći broj ponuđenih lokacija.

Primjer jeste **pronalaženje idealne lokacije za pravljenje novog postrojenja, pronalaženje idealne lokacije za smještanje aerodroma i sl.**

Postupak je:

- **Identifikuju se svi potrebni podaci** (pr. **gradovi, aerodromi**).
- **Identifikuju se potrebni alati** (pr. **select** - za selekciju zadatog grada u *layer-u* gradova, **buffer** - za identifikovanje razdaljine oko zadatog grada, **intersect** - za izdvajanje svih aerodroma iz *layer-a* aerodroma koji se nalaze u 100km baferu oko zadatog grada).

## Mrežna analiza (*Network analysis*).

**Mrežna analiza** predstavlja sistem povezanih tačaka i linija koje predstavljaju moguće putanje od jedne lokacije ka drugoj.

Odgovara na pitanja kao što su **koji je najbrži put do..., koja patrola se nalazi najbliže i sl.**

Pruža informacije pomoću kojih se može napraviti logistička i strateška analiza.

Isključivo se koristi u vektorskom formatu. Sve mreže uključuju i **impedansu** - mjerilo jačine prepreke koje usporavaju vrijeme putovanja (**distanca, trošak za gorivo i sl.**)

Alati mrežne analize su:

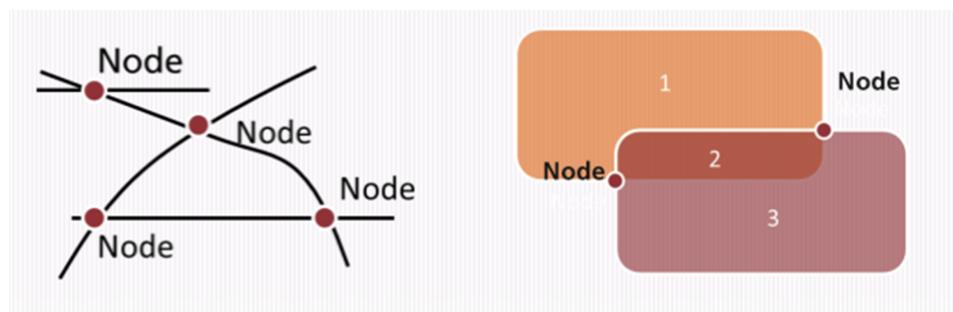
- **Najbliži objekat** - mjeri trošak putovanja između objekata ili određuje koji je objekat najbliži biranom mjestu. Primjer - **uputiti vatrogasno vozilo ka lokaciji požara iz najbliže vatrogasne stanice.**
- **Izbor putanje** - omogućava izvor najbolje putanje bazirane na zadatim kriterijumima. Obično se koristi da nađe putanju uz najmanje troškove koja posjećuje više lokacija. Primjer - **pronaći put koji će potrošiti najmanje vremena i goriva.**
- **Servisne mreže** - identificuje pristupačne ulice unutar zadate impedanse. Primjer - **sva domaćinstva koja su unutar 10km od škole, uzimajući u obzir troškove puta, počevši od polazišta.**
- **Alokacija lokacija** - upotrebom odgovarajućih ulaznih parametara, ovaj alat locira najbolje lokacije za nove objekte uzimanjem u obzir zahtjevane tačke. Primjer - **gdje treba locirati restoran brze hrane da bi se minimizovalo vrijeme puta mušterija do restorana.**

## Šta je topologija? Vrste topologije.

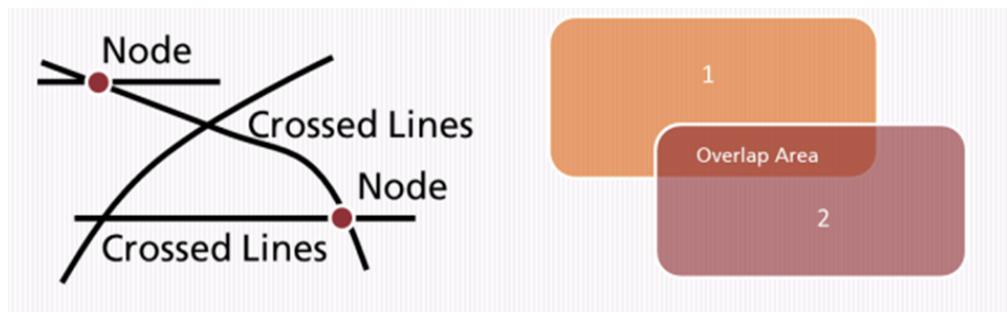
**Topologija** je studija o geometrijskim svojstvima koja se ne mijenjaju kada objekat prolazi kroz transformacije ili modifikacije. Reprezentuje i nameće geometrijske relacije između objekata.

Postoje dvije vrste topologije:

- **Planarna** - smatra da geometrijske reprezentacije postoje na jednoj dvodimenzionalnoj površini.
  - Nisu dozvoljena preklapanja poligona, a da ne prave novi poligon.
  - Ako se linije sijeku, mora da postoji presjek na svakom prelasku linije preko linije.
  - Ako se jedna topologija pomjeri, tada se čvor koji predstavlja presjek takođe pomjera i na taj način se zadržava svojstvo presjeka između dvije linije. Isto važi i za poligone.



- **Neplanarna** - objekti postoje u više ravni sa preklapanjem kod ivica.
  - Nije neophodno da postoji presjek gdje linija prelazi preko linije, ako su one u različitim ravnima. Važi i za poligone.

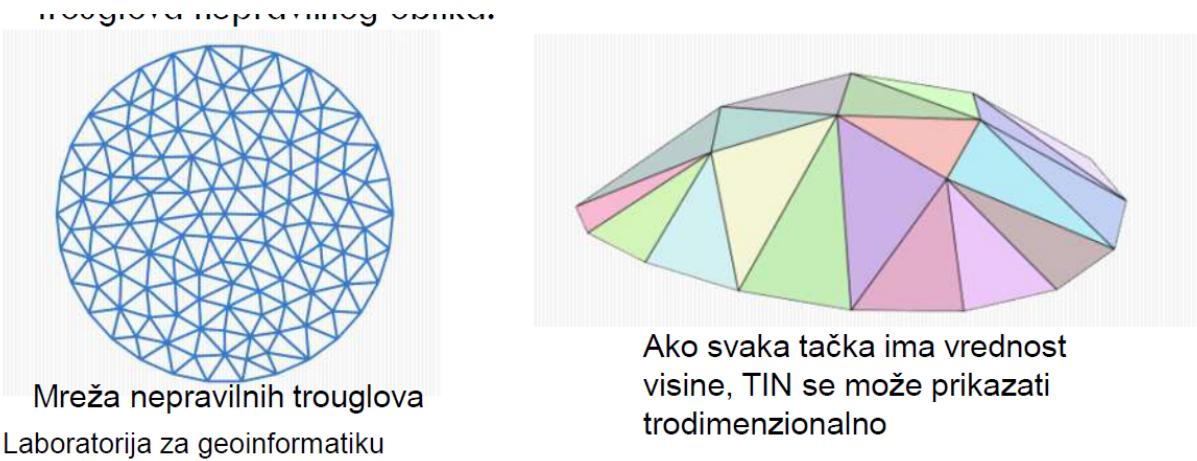


Topologije osiguravaju kvalitet podataka, povezanost linija i obezbeđuju integritet podataka kroz nametanje logičkih pravila. Mana je u tome što to predstavlja vremenski intenzivan proces.

## Šta je triangulacija, kako se formira i gde se primjenjuje?

Triangulacija je mreža nepravilnih trouglova - vektorski bazirana struktura podataka napravljena na bazi triangulacije između tačaka.

Predstavlja **mrežu međusobno povezanih trouglova koja obično predstavlja vrijednosti visina za topografske podatke**.



Prednost mreže nepravilnih trouglova naspram rastera je da može upravljati neuniformnim rastojanjima između ulaznih tačaka, kreirajući time neuniformne veličine trouglova.

Da bi se kreirala mreža nepravilnih trouglova, izvršava se *Delaunay* triangulacija koja koristi sve ulazne tačke - **ni jedna tačka ne smije se nalaziti unutar kruga koji prolazi kroz sva tjemena trougla, niti linije ne smiju da se preklapaju**.



Ako se ulazna tačka nađe unutar kruga, dodaje se novi skup trouglova da obuhvati tu tačku.

Upotreba je u **mapiranju fizičke površine Zemlje, mapiranje dna mora, analiza nagiba terena,...**

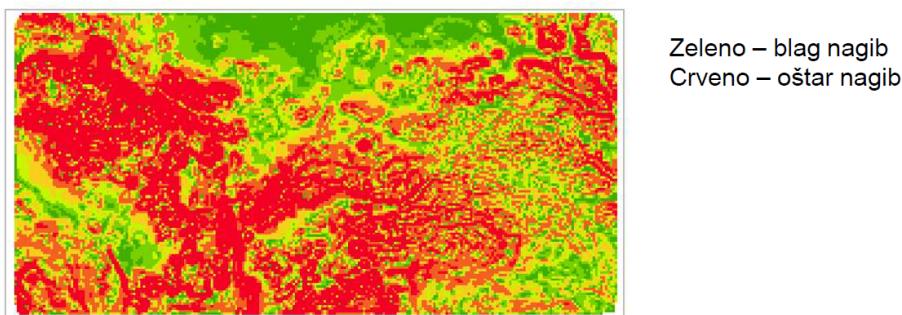
## Tehnike analize terena (analiza nagiba, analiza aspeka (pravca nagiba, *Hillshades*, *Viewshed*)).

### Analiza nagiba.

**Analiza nagiba** kreira površinu koja prikazuje vrijednost nagiba terena. Takva površina prikazuje gdje je teren ravan, umjeren ili jako strm.

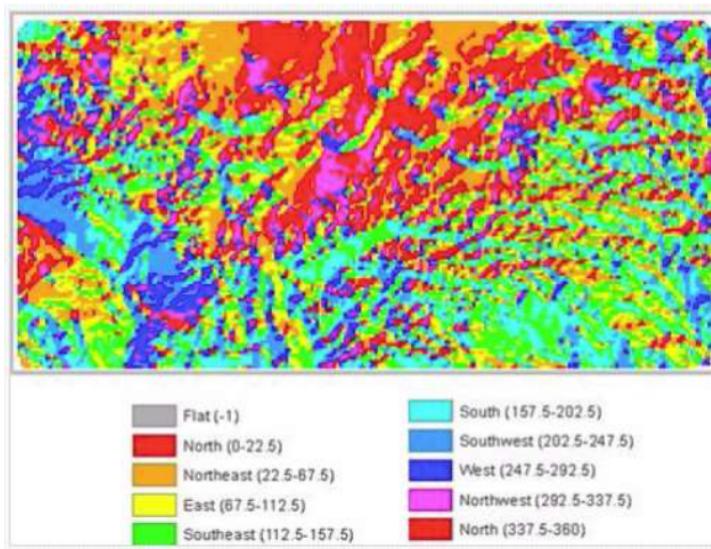
Ovi alati zahtjevaju da ulazi budu rasteri, a izlaz je takođe raster koji definiše jačinu nagiba u svakom pikselu.

Nagib se iskazuje u procentima ili se može klasifikovati da bi se pojednostavio prikaz (ako je nagib veći od 15%, smatrati ga prestrmim).



### Analiza pravca nagiba.

**Analiza pravca nagiba** jeste površinski aspekt koji govori o orijentaciji terena u svakoj tački. Predstavlja usmjerenje nagiba koje pokazuje u kojem se pravcu nagib prostire.



## Hillshades.

**Hillshades** modeli bacaju sjenku na teren za zadatu lokaciju Sunca (reljef sa sijenkom).



U ovom primeru, sunce je locirano u gornjem levom uglu, stoga je teren na severozapadu grebena svetlij od terena na jugoistoku koji je zaklonjen grebenom.

## Viewshed.

**Viewshed** modeli prikazuju područja koja su vidljiva iz određene izvorne tačke. Koriste vrijednosti visina da odrede šta je vidljivo, a šta nije.

Određuju koji objekti smetaju pogledu i koriste ih kod određivanja pravca pogleda da bi se odredilo šta je vidljivo sa pozicije posmatrača.

## Šta je *map algebra*?

**Map algebra** je vrsta **reklasifikacije rastera** koja se bazira na kombinaciji rasterskih lejera u ćeliji kroz proces ćelije. Ona može biti jednostavna operacija kao što je **množenje, sabiranje** i sl. koje mogu ili ne moraju koristiti **algebru skupa** ili **Bulovu algebru**.

1	5	2	3
5	2	6	1
2	6	3	7
3	1	7	4

-

4	2	8	6
3	1	3	5
2	0	7	1
1	9	3	2

=

-3	3	-6	-3
2	1	3	-4
0	6	-4	6
2	-8	4	2

Rezultujući raster je rezultat oduzimanja drugog rastera od prvog na nivou svake pojedinačne ćelije

## Mapiranje gustine (*Heatmap*).

**Mapiranje gustine** omogućava korisniku da vidi gdje se nalazi veliki broj observacija ili viših vrijednosti. Gustine prikazuju **prostorne relacije** među različitim lokacijama podataka.

Ono omogućava da se kreiraju predikcije iz podataka gdje će se druge lokacije ili vrijednosti dogoditi. Primjer za to su **mapiranje lokacija sa najviše ili najmanje bolnica** i sl.

Analiza gustine odvija se u više nivoa:

- **Gustina tačaka** - izračunava gustinu tačaka na određenoj oblasti. To se postiže dodavanjem tačke određenoj oblasti proučavanja i dijeljenjem ukupnog broja površinom zadate oblasti.
- **Gustina linija** - određuje oblast sa većom gustom linija naspram oblasti sa manjom gustom linija.
- **Kernel gistica** - računa gustinu tačaka ili linija **kernel funkcijom** - ova funkcija postavlja veću težinu nekim objektima bazirano na specifičnim vrijednostima (pr. vatrogasni hidranti sa većim pritiskom imaju veći težinski faktor u *kernel* funkciji).

**QGIS plugin Heatmap** omogućava generisanje površine gustine tačaka (bazirane na nekom atributu).

## Šta je geostatistika i gde se koristi?

**Geostatistika** je vrsta statistike koja se koristi da analizira i predviđi vrijednost pridružene prostornim ili prostorno-vremenskim fenomenima. Mnogi geostatički alati su prvobitno bili razvijeni kao praktična sredstva da opišu prostorne obrasce i da **interpoliraju vrijednosti za lokacije na kojima uzorci nisu uzeti**, dok danas daju i **mjeru neizvjesnosti** za ove vrijednosti.

Geostatistika ima široku primjenu:

- **Rudarstvo** - klasifikacija mineralnih resursa i procjena ekonomske izvodljivosti projekta.
- **Ekološke nauke** - procjena nivoa zagađujućih materija.
- **Nauke o zemljištu** - mapiranje nivoa hranjivih materija u tlu.
- **Meteorološka primjena** - predviđanje temperature, padavina i sl.

Geostatičke analize koriste uzorce na različitim lokacijama nekog područja, stvarajući (interpolirajući) kontinualnu površinu. Tok rada je:

- **Mapiranje i proučavanje podataka**
- **Izgradnja geostatičkog modela**
  - **Pre-processing** podataka
  - **Modelovanje prostorne strukture**

- Definisanje strategije pretrage
- Predviđanje vrijednosti na lokacijama koje nisu ispitane
- Kvantifikovanje neodređenosti predikcije
- Provjera modela
- Upotreba informacija u analizama rizika i donošenju odluka

Metode interpolacije mogu biti:

- **Determinističke** - koriste matematičke funkcije za interpolaciju koje određuju glatkoću rezultujuće površine - **IDW**
- **Geostatističke** - bazirane na statističkim modelima koji uključuju **autokorelaciju** - **Kriging**.

## **Objasniti *IDW* i *Kriging* i navesti primene.**

### ***IDW - Inverse Distance Weighting***

Da bi se uzela u obzir udaljenost dvije tačke, vrijednosti bližih tačaka su ponderisane više nego kod onih koje su udaljenije - osnova za **IDW** interpolacionu tehniku - **težina vrijednosti opada sa porastom od lokacije za predikciju**.

Više vrsta polinomijalne interpolacije:

- **Globalna polinomijalna interpolacija:**
  - **Prvog reda** - kontinualni nagib, ne uzima u obzir lokalne varijacije.
  - **Drugog reda** - za modelovanje doline, dozvoljeno jedno savijanje ravni.
- **Lokalna polinomijalna interpolacija:**
  - Uzima u obzir lokalne varijacije.
  - Uklapa mnogo manjih preklapajućih ravni, a zatim koristi centar svake ravni kao predviđanje za svaku lokaciju.
  - Rezultujuća površina postaje fleksibilnija i preciznija.
- **Funkcija radijalne osnove:**
  - **Radial Basis Function.**
  - Omogućava praćenje globalnog trenda uz uzimanje u obzir lokalne varijacije.

### ***Kriging***

**Kriging metoda** predstavlja geostatističku metodu interpolacije koja ponderiše okružujuće izmjerene vrijednosti kako bi izvela predviđanje za svaku lokaciju (slično **IDW**.)

Ponderi nisu samo bazirani na udaljenosti između izmjerениh tačaka i lokacija za predviđanje, već i na cjelokupnom prostornom rasporedu između izmjerениh tačaka.

**Prostorna autokorelacija mora biti kvantifikovana** - prirodni fenomeni je često predstavljaju - vrijednosti uzoraka koji se uzimaju jedan pored drugih sličniji su od onih uzetih daleko jedan od drugih. Kada postoji prostorna autokorelacija, tradicionalne statističke metode ne mogu se pouzdano koristiti.

Primjer - **određivanje vrijednosti visine u tački**.

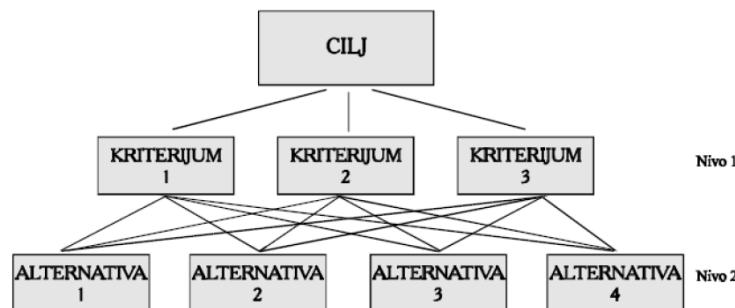
**ESRI ArcGIS Geostatistical Analyst** je alat koji pruža mogućnost modelovanja površine upotrebom determinističkih i geostatističkih metoda u **ESRI ArcGIS** okruženju.

## Objasniti AHP metodu.

**Analitički hijerarhijski model** spada u klasu metoda za meku optimizaciju. U osnovi se radi o alatu za formiranje i analizu **hijerarhija odlučivanja**.

**AHP** najprije omogućava interaktivno kreiranje hijerarhije problema kao pripremu scenarija odlučivanja, a zatim vrijednovanje u parovima elemenata hijerarhije u *top-down* smjeru.

Na kraju se vrši sinteza svih vrijednovanja i po strogom utvrđenom matematičkom modelu određuju težinski koeficijenti svih elemenata hijerarhije. Zbir težinskih koeficijenata elemenata na svakom nivou hijerarhije jednak je 1 što omogućava donosiocu odluka da rangira sve elementa u horizontalnom i vertikalnom smislu.



AHP metod bira najbolju od ponuđenih alternativa na bazi ponderisanih kriterijuma. Kriterijumima su dodeljeni težinski faktori (ponderi) u skladu sa njihovom važnošću. Ponderisanje se radi prema Satjevoj skali.

## Šta podrazumeva modelovanje podataka?

**Model sistema (modelovanje podataka)** jeste prikaz bitnih funkcija nekog stvarnog (ili zamišljenog) sistema koji predstavlja obilježja sistema u upotrebljivom obliku. Modeli se izrađuju u cilju boljeg razumijevanja sistema jer pomažu u vizualizaciji stvarnog ili zamišljenog sistema, opisivajući njegovu strukturu i predstavljajući šablon po kom se

sistem može implementirati. Obično se dokumentuju svi koraci izgradnji jednog takvog sistema.

Izbor modela utiče na pristup problemu i oblikovanju rješenja - **nijedan model sam po sebi nije dobar, potrebno modelovati iz više raznih gledišta.**

Za potrebe modelovanja sistema razvijen je jezik za modelovanje sistema - **UML** (*Unified Modeling Language*).

## Šta je **UML**?

**UML** (*Unified Modeling Language*) jeste jezik razvijen za potrebe modelovanja sistema. Predstavlja **vizualizaciju za opis, izgradnju i dokumentovanje softverske podrške kod analize i izrade prvenstveno softverskog rješenja**. To je **formalan jezik namijenjen za formalno specificiranje, modelovanje, analizu, vizualizaciju, dokumentovanje i razvoj softverskih sistema**.

**UML** olakšava razmjenu informacija među učesnicima projekta budući da je usmjerен na višestruke komunikacije. Nudi skup dobro određenih grafičkih prikaza i dijagrama, razumljivih i projektantima i korisnicima sistema.

Tehnički opis **UML** dijagrama postaje precizan, nedvosmislen i potpun.

Zasnovan je na **principima objektne orijentacije**.

## Polazni koncepti **UML**.

**UML** je zasnovan na **principima objektne orijentacije**, te iz toga ishode polazni koncepti:

- **Klasa** - skup objekata koji imaju iste osobine (**atribute**) i funkcionalnosti (**operacije**), istu **semantiku i zajedničke veze sa drugim objektima**.
- **Objekat** - pojava posmatrane **klase** koja ima određenu ulogu. Opisuje pojedinačni entitet, bio stvaran ili apstraktan, sa dobro definisanom ulogom u modelu problema.
- **Veza** - model mogućih **odnosa** između klasa objekata i/ili samih objekata.  
Osnovne vrste veze:
  - **Asocijacija** (relacija udruživanja)
  - **Agregacija i kompozicija** (relacija sastavljanja)
  - **Zavisnost**
  - **Generalizacija** (relacija uopštavanja)
- **Poruka** - oblik moguće interakcije između objekata.

Dijagrami za modelovanje se razvijaju alatima za vizuelno modelovanje kao što su **Enterprise Architect, Power Designer, IBM Rational Rose,...**

## **Tipovi UML dijagrama.**

Podjela se vrši na osnovu toga da li opisuju statičku strukturu sistema ili ponašanje sistema.

### **Dijagrami strukture**

#### **Dijagram klasa**

Opisuje statički pogled na model kroz klase i veze između njih.

#### **Dijagram objekata**

Specijalni slučaj dijagrama klasa koji naglašava vezu između instanci klasa u nekom trenutku vremena.

#### **Dijagram komponenti**

Pokazuje komponente sistema, osnovne gradivne elemente sistema.

#### **Dijagram paketa**

Koristi se za podjelu modela u logičke cjeline i prikaz veza između tih cjelina.

#### **Dijagram rasporeda**

Modeluje fizički sistem sistemskog mapiranja softverskih elemenata na hardver u kojem se oni izvršavaju, kao i njihovu komunikaciju.

#### **Dijagram kompozitne strukture**

Pokazuje unutrašnju strukturu klase i saradnje koje je omogućavaju i opisuju.

### **Dijagrami ponašanja**

#### **Dijagram slučajeva korišćenja**

Služi za specificiranje poslovne funkcije sistema i predstavlja opis niza akcija koje sistem izvršava prilikom realizacije te poslovne funkcije, a koja se pokreće na zahtjev korisnika.

#### **Dijagram aktivnosti**

Služi da opiše redoslijed izvršavanja aktivnosti u okviru jednog slučaja korišćenja.

## Dijagram stanja

Pokazuje ponašanje sistema kao odgovor na neki ulaz. Prikazuje stanja sistema i način kako taj sistem prelazi iz jednog u drugo stanje.

## Dijagram poruka

Modeluje interakcije između objekata ili dijelova u smislu sekvenciranih poruka.

## Dijagram sekvenci

Pruža grafički prikaz interakcija objekata tokom vremena. Jedan dijagram sekvence obično predstavlja jedna scenario slučaja korišćenja ili tok dijagrama.

## Dijagram vremena

Koristi se za prikaz promjene stanja ili vrijednosti jednog ili više elemenata modela tokom vremena.

## Dijagram interakcije

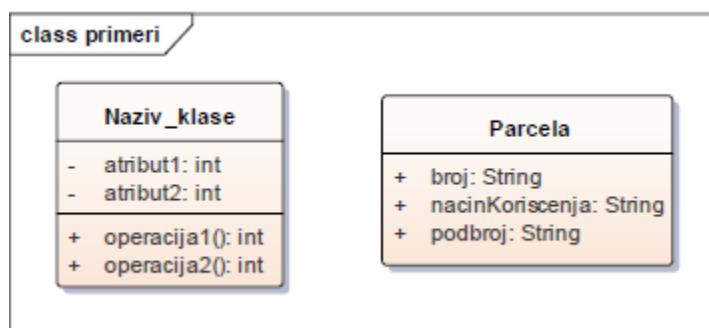
Koristi kontrole iz dijagrama aktivnosti kako bi se kontrolna logika postavila oko dijagrama nižih nivoa.

## Objasniti dijagram klasa.

**Dijagram klasa** opisuje tipove objekata nekog sistema i različite vrste statičkih odnosa koji postoje između tih objekata. Predstavljaju grafičku reprezentaciju statičkog pogleda statičkih elemenata.

Najvažniji elementi *UML* dijagrama klasa su:

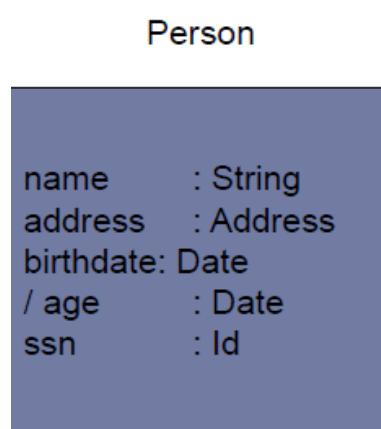
- **Klase** - opisi skupa objekata koji imaju slične attribute, operacije i sl.



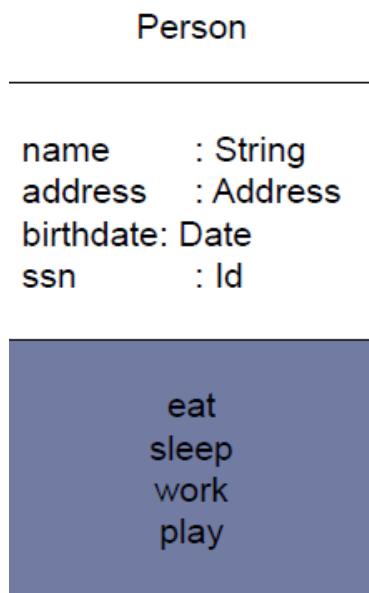
- **Atributi** - osobine objekata klase. **Vidljivost atributa** predstavlja **pravo pristupa** i odnosi se na:
  - **Javni** - pristup neograničen, simbol “+”
  - **Privatni** - pristup ograničen na članove iste klase i podklase, simbol “-”

- **Zaštićeni** - privatno pravo pristupa prošireno na klase izvedene iz date klase, simbol “#”
- **Implementacioni** - jedinstvena identifikacija klase realizovana označavanjem atributa za ključ uvođenjem znaka “\*”

Atribute obično srećemo u paru *attributeName : Type*.



- **Operacije (metode)** - opisuju ponašanje klase i pojavljuju se u trećem dijelu klase.



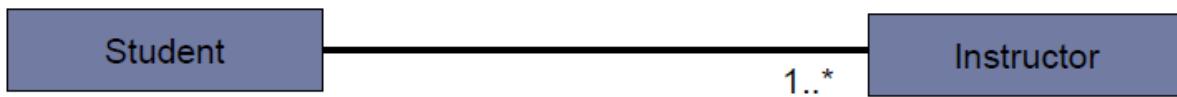
- Odnosi - razlikuju se relacije:
  - Asocijacije
  - Agregacije
  - Kompozicije

## Tipovi veza između klasa.

### Asocijacija

Odnos **asocijacije** specificira da li je objekat neke klase u vezi sa objektom druge (moguće i iste) klase. Može posjedovati sopstveno ime.

Na oba kraja definiše se **višestrukost (multiplikativnost)** - označava broj instanci i ukazuje na to da li je asocijacija obavezna ili ne.



Moguće je nazvati ulogu **relacije** između klasa.

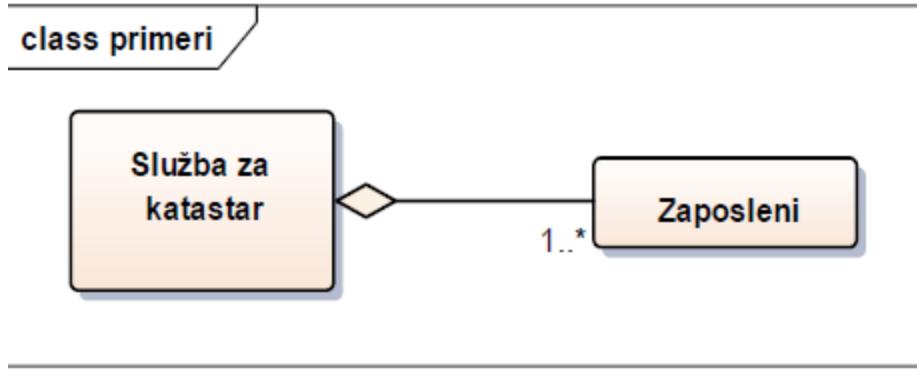


Postoje i **usmjerenе relacije** koje označavaju da se komunikacija između objekata isključivo vrši **u jednom pravcu**.

**Ukoliko su dva objekta nezavisna, onda je relacija asocijacija.**

### Agregacija

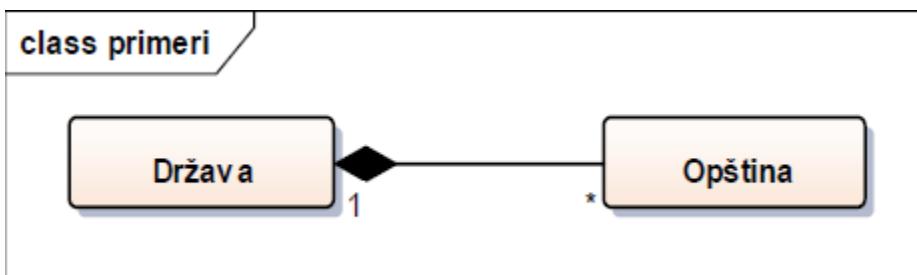
Specijalan oblik asocijacija koji modeluje odnos **cjelina - dio**. Kod agregacije, **dijelovi postoje nezavisno od cjeline**.



Ukoliko su dva objekta usko povezana relacijom **cjelina - dio**, onda je relacija **agregacija**.

#### Kompozicija

Predstavlja jači oblik agregacije, gdje se takođe modeluje odnos **cjelina - dio**, s tim da je **cjelina isključivi i jedini vlasnik njenih dijelova**. Životni vijek **dijela** isključivo zavisi od **cjeline**.



### Razlika između agregacije i kompozicije.

Vidjeti iznad.

### ISO 19107 standard.

**ISO 19107** standard definiše konceptualnu šemu za opisivanje i modeliranje geografskih objekata i njihovih svojstava. Služi za određivanje osnovnih prostornih objekata i operacija koje se mogu koristiti za reprezentaciju geografskih informacija. Standard obuhvata dvodimenzionalne i trodimenzionalne prostorne podatke i pruža osnovu za interoperabilnost između različitih sistema geografskih informacija (GIS). Ključni koncepti definisani u ISO 19107 uključuju:

- **Prostorni objekti** - definiše različite vrste prostornih objekata, kao što su **tačke, krive, površine i tela**.

- **Prostorne atribute** - opisuje **karakteristike ili svojstva** koja su povezana sa prostornim objektima.
- **Prostorne odnose** - definiše **prostorne operacije i odnose između različitih prostornih objekata**, kao što su **presecanje, sadržajnost i povezanost**.
- **Prostorni referentni sistem** - **specificuje koordinatni sistem i prostorni referentni okvir** koji se koristi za precizno lociranje i pozicioniranje prostornih objekata.

## Šta je topologija?

Pitanje već postoji.

## Geometrijski tipovi prema ISO19107.

- **Tačka (Point)** - predstavlja osnovni geometrijski element koji ima tačno jednu poziciju u prostoru.
- **Linija (Curve)** - linija se koristi za reprezentaciju apstraktnih ili stvarnih krivih koje mogu biti **jednostavne** (npr. prava linija) ili **složene**.
- **Površina (Surface)** - površina se koristi za predstavljanje dvodimenzionalnih objekata koji zauzimaju određenu oblast u prostoru.
- **Telo (Solid)** - telo predstavlja trodimenzionalni objekat koji ima zapreminu i definisanu površinu.

Ovi geometrijski tipovi omogućavaju reprezentaciju različitih vrsta prostornih objekata i oblika koji se koriste u **geografskim informacijama**. Takođe je moguće koristiti kombinaciju ovih geometrijskih tipova za reprezentaciju složenijih prostornih entiteta.

## Topološki tipovi prema ISO19107.

- **Unutrašnji odnos (Interior)** - odnos koji se odnosi na unutrašnjost prostornog objekta. Na primer, tačke mogu biti unutar površine, linije mogu biti unutar površine, ili površine mogu biti unutar tela.
- **Granični odnos (Boundary)** - odnos koji se odnosi na granicu između prostornih objekata. Na primer, linije mogu biti na granici površine, tačke mogu biti na granici linije, ili površine mogu biti na granici tela.
- **Eksteriorni odnos (Exterior)** - odnos koji se odnosi na spoljašnjost prostornog objekta. Na primer, tačke mogu biti van površine, linije mogu biti van površine, ili površine mogu biti van tela.
- **Topološka veza (Topological Link)** - veza koja se uspostavlja između dva ili više prostornih objekata na osnovu njihovih topoloških odnosa. Na primer, dve linije mogu biti povezane ako su im granice blizu ili se presecaju.

Korišćenje **topoloških tipova** omogućava precizno definisanje prostornih relacija između geometrijskih objekata. Ovi tipovi su važni za analizu prostornih podataka, kao što su presecanje objekata, provera pripadnosti objekata određenoj oblasti ili određivanje susedstva između objekata.

## Objasniti relacioni model podataka.

Motiv razvoja relacionog modela podataka jeste otklanjanje nedostatka klasičnih modela podataka, čvrsta povezanost logičkih i fizičkih aspekata, strukturalna kompleksnost i navigacioni jezik.

Kod ranijih modela podataka, fizički aspekti baza podataka su bili ugrađeni u programe, stoga je zahtjev kod sadašnjih modela podataka nezavisnost programa od podataka. To se rješava potpunim odvajanjem prezentacionog od formata memorisanja, predstavljanjem relacije kao skup  $n$ -torki.

Opis relacije je  $N(R, K)$ , gdje  $R$  predstavlja skup obilježja, dok  $K$  predstavlja skup ključeva.

### ■ Primer:

- $Fakultet (\{FAK, NAZ, BIP\}, \{FAK\})$
- $r(Fakultet) = \{(PMF, Matematički, 7), (EKF, Ekonomski, 4), (ETF, Elektrotehnički, 9), (MAF, Mašinski, 7)\}$
- Torka ( $EKF, Elektronski, 8$ ), narušava uslov integriteta

Problem selekcije podataka putem proceduralnih jezika kod ranijih modela podataka je to da se svaki podatak pronađe na osnovu naziva relacije, obilježja i vrijednosti ključa. Rješenje toga jeste strukturalna jednostavnost, odnosno da se **podaci predstave pomocu tabele**.

*Fakultet*

FAK	NAZ	BIP
FIL	Filozofski	1
PMF	Matematički	7
ETF	Elektrotehnički	9
EKF	Ekonomski	4
MAF	Mašinski	7

**Deklarativni jezik** je vrsta programskog jezika koja se koristi u relacionom modelu podataka za opisivanje željenih rezultata ili informacija koje se traže iz baze podataka, umesto da se precizno navedu koraci za postizanje tih rezultata.

Najpoznatiji deklarativni jezik je **SQL (Structured Query Language)**.

Osnovni upitni blok SQL-a je

```
SELECT <lista obilježja>
FROM <lista relacija>
WHERE <kvalifikacioni izraz>.
```

■ Primer:

- **SELECT** *ime, prz, bip*  
**FROM** *fakultet, projektant*  
**WHERE** *bip > 5 AND*  
          *fakultet.fak = projektant.fak*

IME	PRZ	BIP
Iva	Ban	7
Ana	Tot	7
Aca	Pap	9
Eva	Tot	9

## Objasniti objektno-relacioni model podataka.

**Objektna orijentacija** je pristup u kome se neki sistem organizuje kao kolekcija međusobno povezanih objekata koji ostvaruju postavljene ciljeve sarađujući međusobno. Objektna orijentacija je danas preovladavajuća u programiranju i programskim jezicima, metodološkim pristupima razvoju softvera i informacionih sistema, a posebno preuzima primat i u sistemima za upravljanje bazom podataka. Jedna od najbitnijih karakteristika sistema za upravljanje bazama podataka je **skup tipova** koje on podržava. Konvencionalni SUPB (hijerarhijski, mrežni i relacioni) su zasnovani na tipu rekorda koji predstavlja agregaciju polja (atributa u relacionom modelu) koja su definisana nad standardnim tipovima.

**Objektni SUPB** su postavili sebi cilj da **stvore mnogo bogatiji skup tipova** i time omoguće prirodniju manipulaciju podacima u aplikacijama koje se razvijaju nad bazom. Bilo koji sistem tada posjeduje mogućnost da se posmatra kao skup međusobno povezanih objekata.

## **Navesti i objasniti SQL naredbe.**

Pored naredbe **SELECT** koja se koristi za **dobavljanje podataka** iz baze, imamo još naredbe:

- **INSERT** - koristi se za dodavanje novih redova u tabelu.
- **UPDATE** - koristi se za ažuriranje postojećih podataka u tabeli.
- **DELETE** - koristi se za brisanje podataka iz tabele.
- **CREATE TABLE** - koristi se za kreiranje novih tabela, indeksa ili drugih objekata baze podataka.
- **ALTER TABLE** - koristi se za izmjenu strukture postojećih tabela, dodavanje ili brisanje kolona.
- **JOIN** - koristi se za povezivanje podataka iz više tabela na osnovu zajedničkih kolona.
- **DROP** - koristi se za brisanje tabela, indeksa ili drugih objekata baze podataka.
- **GROUP BY** - koristi se za grupisanje podataka na osnovu određene kolone.

## **Šta je SUPB i koje su mu prednosti i karakteristike.**

**Sistemi za upravljanje bazama podataka** omogućavaju efikasno i pouzdano formiranje, korišćenje i mijenjanje **baza podataka**. Karakteristike koje jedan sistem za upravljanje bazama podataka mora posjedovati su:

- Mora biti **zasnovan na nekom modelu podataka** (u idealnom slučaju, trebalo bi da podrži sve koncepte i sve karakteristike, tj. prednosti izabranog modela podataka).
- Mora posjedovati **jezik** (ili **jezike**) za obezbjeđenje upravljanja **bazom podataka**.
- **Pouzdanost** - visoka vjerovatnoća bezotkaznog rada u realnom vremenu.
- **Pogodnost korišćenja** - lakoća realizacije predviđenih zadataka i automatizacija postupaka realizacije istih.
- **Pogodnost za održavanje**.
- **Postojanje rječnika (kataloga) baze podataka** - rječnik je **baza podataka** sistema za upravljanje.

## Upravljanje transakcijama u SUBP.

Obrada podataka u BP odvija se iskljucivo putem transakcija. **Transakcija je najmanja jedinica obrade podataka koja prevodi BP iz jednog u drugo** (ne nužno i različito) konzistentno stanje.

Transakcija još predstavlja jedno izvršenje nekog transakcionog programa nad BP (sačinjenog od operacija upita i/ili azuriranja BP) koje u cijelosti uspjeva ili čiji se efekti u cijelosti poništavaju.

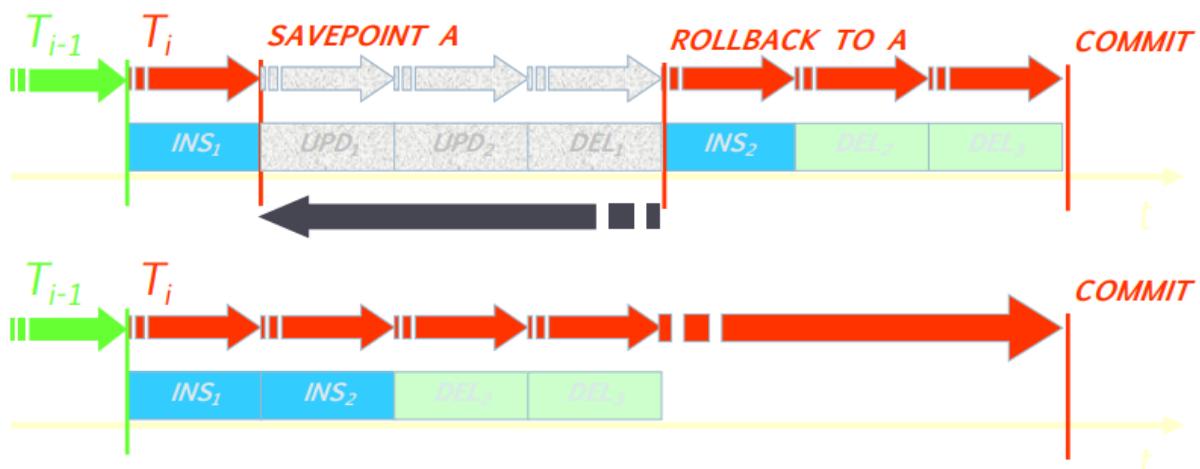
Transakcija ima svoj početak eksplisitno označen sa **BEGIN TRANSACTION** ili implicitno podrazumijevan, nailaskom prve operacije ažuriranja nakon prethodno završene transakcije ili nakon prijavljivanja korisnika za rad sa BP.

Takođe, transakcija ima i svoj kraj označen zahtjevom za potvrđivanje transakcije **COMMIT** ili označen zahtjevom za poništavanje transakcije **ROLLBACK** (**ROLLBACK** može biti i implicitno podrazumijevan, dolaskom do greške u obradi podataka i tada se naziva *implicitni ROLLBACK*).

Transakcija takođe posjeduje djelimično poništavanje kada postavimo određeni **SAVEPOINT** do koga zelimo da ponistimo transakciju (pr. **ROLLBACK TO SAVEPOINT**).

Transakcija će biti **uspjesno izvršena (potvrđena)** ako je to u transakcionom programu eksplisitno zahtjevano (zahtjevom tipa **COMMIT**) i ako SUBP tu potvrdu može uspješno da realizuje. Zatim BP prelazi u novo konzistentno stanje, koje se u opštem slučaju razlikuje od stanja na početku izvođenja transakcije, a moguce je da bude identično prethodnom stanju.

Transakcija će biti u cijelosti poništена u svim opštim slučajevima (npr kada SUBP ne može da je potvrdi, iako je potvrda zahtjevana ili kad pozovemo **ROLLBACK**). BP, u tom slučaju, ostaje u stanju koji je važio na početku transakcije.



Što se tiče vidljivosti, za korisnika koji je pokrenuo transakciju vidljiva je svaka promjena, dok za ostale korisnike sistema vidljivo je stanje koje BP koje je važilo neposredno prije početka izvođenja transakcije, kao i novo stanje BP.

## Struktura SUPB.

Posjeduje komponente i mehanizme za:

- **Upravljanje transakcijama**
- **Upravljanje višekorisničkim režimom rada**
  - Omogućava istovremeni pristup bazi podataka više korisnika. To znači da više korisnika može istovremeno izvršavati upite, izmjene ili druge operacije nad istom bazom podataka.
- **Zaštitu BP od neovlašćenog pristupa**
  - **Preventivni** - sprječavaju neovlašćene načine upotrebe BP.
  - **Korektivni** - omogućavaju evidentiranje i ispitivanje načina upotrebe BP od strane ovlašćenih i neovlašćenih lica.
- **Zaštitu BP od uništenja / oštećenja**
  - **Preventivni** - sprječavaju pokušaje uništenja ili oštećenja BP.
  - **Korektivni** - omogućavaju efikasno vraćanje oštećene ili uništene BP u stanje gotovosti za upotrebu.
- **Upravljanje distribuiranim BP**
  - Globalno upravljanje nazivima objekata i objezbjeđenje lokacijske transparentnosti.
  - Obezbeđenje komunikacije između servera i BP.
  - Upravljanje distribucijom rječnika SUBP.
- **Upravljanje replikacijom u BP**
  - Omogućava kreiranje i održavanje kopija baze podataka na različitim lokacijama.
- **Obezbeđenje performantnog korišćenja BP**
  - Obezbeđivanje zadovoljavajućeg vremena odziva sistema za unaprijed predviđene funkcionalne zahtjeve.

## Šta su geoprostorne baze podataka.

**Geoprostorne baze podataka** su vrsta baza podataka koje su specijalizovane za skladištenje, upravljanje i manipulaciju geoprostornim podacima. Geoprostorne baze podataka spadaju u prostorne sisteme za upravljanje bazama podataka koji koriste prostorne upite - **koja su imena fakulteta koji imaju više od 1000 studenata u krugu od 100km od Beograda?**

Primjer neprostornih podataka su **imena, brojevi telefona, email adrese** i sl., dok je primjer prostornih podataka **popis stanovništva, satelitski snimci, vremenski i klimatski podaci**.

Geoprostorni podaci podržavaju geometrijske tipove podataka kao što su:

- **Point** - predstavlja tačku u koordinatnom prostoru (ima  $(x, y)$  set koordinata).
- **Multipoint** - bezdimenzionali geometrijski skup i sadrži isključivo objekte klase **Point**
- **Curve** - kriva, jednodimenzionala geometrija predstavljena nizom tačaka.  
Podtipovima je moguće definisati kakvu interpolaciju krive želimo.
- **Multicurve** - skup geometrija krivih. Predstavlja apstraktnu klasu.
- **Surface** - apstraktna dvodimenzionala geometrijska klasa za površinu.
- **Polygon** - podtip **Surface**, definisan jednim spoljašnjim ograničenjem sa 0 ili više unutrašnjih ograničenja. Svaka unutrašnja granica predstavlja rupu u geometriji.

## Navesti primere upotrebe GPB.

**Geoprostorne baze podataka (GPB)** koriste se za skladištenje, upravljanje i analizu geografskih podataka. Evo nekoliko primera:

- **Upravljanje infrastrukturom** - geoprostorne baze podataka se koriste za upravljanje različitim vrstama infrastrukture, kao što su putevi, vodovodne mreže, električne mreže, telekomunikacione mreže i druge.
- **Planiranje urbanih područja** - GPB se koriste za analizu urbanih područja i planiranje njihovog razvoja. Podaci o zemljištu, parcelama, zonama namene zemljišta, infrastrukturi, prevozu,...
- **Upravljanje prirodnim resursima** - GPB se koriste za upravljanje prirodnim resursima kao što su šumarstvo, poljoprivreda, vodni resursi, zaštita životne sredine i ostali ekosistemi.
- **Katastar i nekretnine** - GPB se koriste za vođenje kataстра i evidenciju nekretnina.
- **Analiza tržišta i poslovna inteligencija** - geoprostorne baze podataka se koriste u analizi tržišta i poslovnoj inteligenciji.

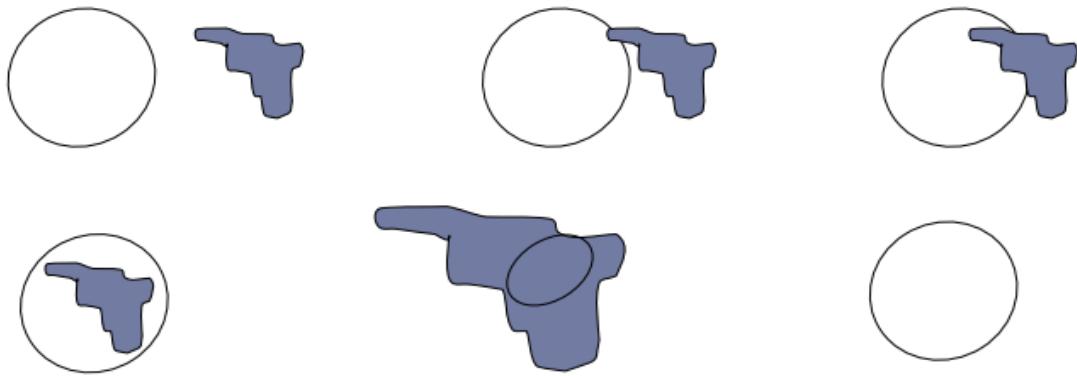
## Prostorni odnosi u GBP.

Prostorni odnosi u geoprostornim bazama podataka (GBP) odnose se na međusobni položaj i interakciju prostornih objekata koji se čuvaju u bazi podataka.

Tipovi prostora i primer operacija: - Topološki - Pored

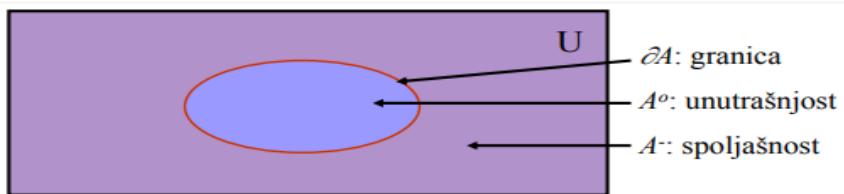
- Mrežni - Najkraći put od
- Direktni - Severno od
- Euklidov - Udaljenost od

Pitanje je kako se moze precizno definisati prostorni odnos:



Devetopresecni(Nine-intersection) model:

Model koji posjeduje 3 komponente:granicu,unutrasnjost i spoljasnost.



$$\Gamma_9(A, B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

Svaki par objekata ima 9 mogucih preseka izmedju njihovih komponenti.Rezultat preseka je prazan(0) skup ili neprazan(1) skup.


Granice objekata-granica se sastoje od tačaka ili linija koje razdvajaju unutrašnjost od spoljašnjosti. Postoje granice linijskih, granice multilinijskih i granice poligona.

## Primarni i sekundarni filter kod selekcije podataka i primeri.

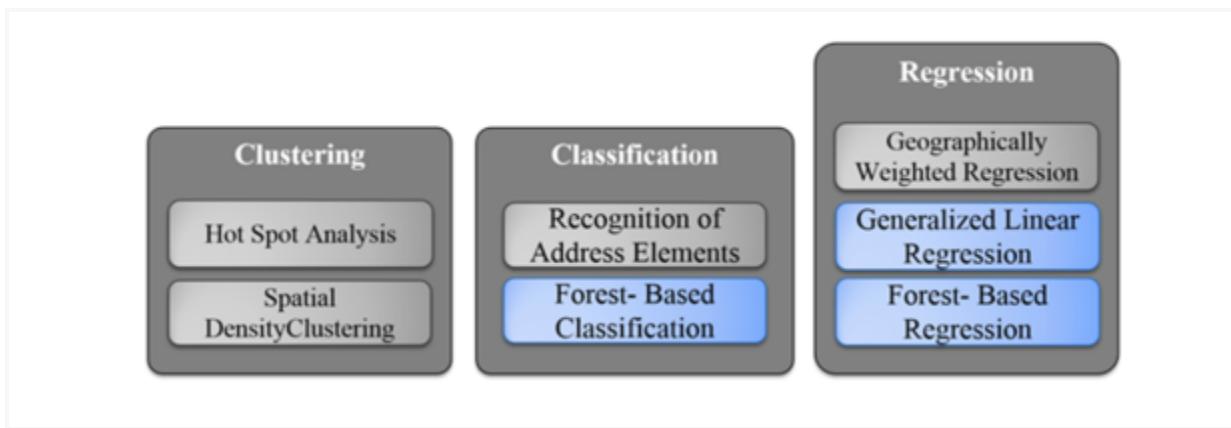
- Primarni filter omogućuje brzu selekciju kandidata datog upita koje zatim prosleđuje sekundarnom filteru. Primarni filter upoređuje aproksimacije geometrija da bi smanjio kompleksnost izračunavanja i smatra se manje zahtevnim filterom. S obzirom da primarni filter upoređuje aproksimacije geometrija, on vraća nadskup tačnog rezultujućeg skupa.

- Sekundarni filter primenjuje tačna izračunavanja na geometrijama koje su rezultat primarnog filtera. Sekundarni filter daje tačan odgovor na prostorni upit. On je zahtevan u pogledu izračunavanja, ali se primenjuje samo na rezultat primarnog filtera, a ne na celokupan skup podataka.



## Šta je mašinsko učenje?

Mašinsko učenje je jezgro veštačke inteligencije. Mašinsko učenje je grana veštačke inteligencije koja se bavi proučavanjem i razvojem algoritama i modela koji omogućavaju računarima da uče iz podataka i donose zaključke bez eksplizitnog programiranja. Kod geoprostornog mašinskog učenja glavni fokus je na primenu mašinskog učenja na geoprostorne podatke. Cilj je da upotrebom GIS alata u koji su integrirani algoritmi mašinskog učenja korisnici mogu da reše raznovrsne probleme u geodomenu, kao što su: prostorni klastering, prostorna klasifikacija, prostorna regresija.



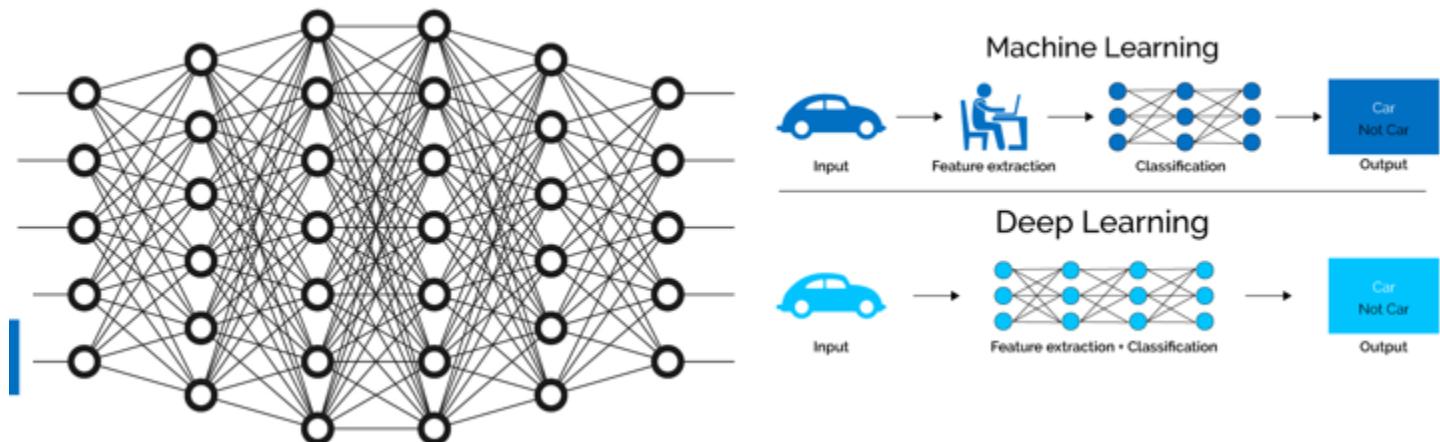
Mašinsko učenje je sve više u upotrebi i utiče na naš svakodnevni život. Primeri upotrebe su: glasovni asistent koji koristi NLP(natural language processing), prepoznavanje govora(computer vision), sistemi za preporuku(Netflix, Youtube, Amazon...), ciljni marketing.

Definicija mašinskog učenja po Tom M. Mitchell-u: "Za računarski program se kaže da uči iz iskustva E u odnosu na neku klasu zadatka T i meru učinka P,ako se njegove performanse na zadacima u T,mereno pomoću P,poboljšavaju iskustvom E". Dok definicija po Arthur Samuel-u glasi "Mašinsko učenje je podoblast računarskih nauka koja daje računarima sposobnost da uče bez da budu eksplizitno programirani".

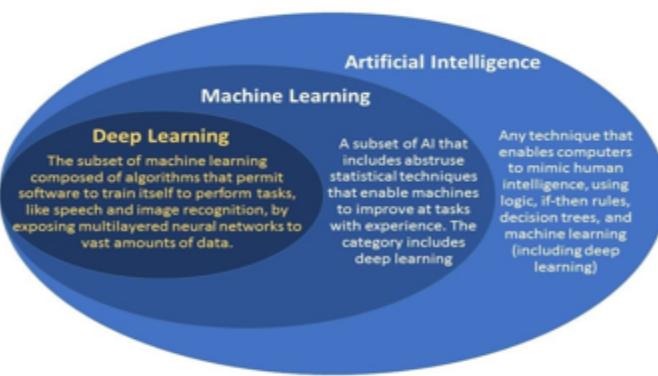
## Šta je duboko učenje?

Najaktuelniji pravac razvoja mašinskog učenja je duboko učenje. Duboko učenje koristi kompjuterski generisane neuronske mreže, koje su inspirisane ljudskim mozgom i koje u širem smislu liče na ljudski mozak, za rešavanje problema i pravljenje predviđanja.

Duboko učenje je podskup mašinskog učenja u kom neuronske mreže sa više slojeva,kombinovane sa velikom procesorskom moći i velikim skupovima podataka mogu da kreiraju moćne ML modele.Zove se duboko učenje jer neuronske mreže imaju različite slojeve(duboke) koji omogućavaju učenje.



AI je u osnovi kada mašine mogu da rade zadatke koje obično zahtevaju ljudsku inteligenciju. Obuhvata mašinsko učenje,gde mašine mogu da uče kroz iskustvo i steknu veštine bez ljudsko učešća,dok duboko učenje je podskup mašinskog učenja,gde veštačke neuronske mreže,algoritmi inspirisani ljudskim mozgom,uče na velikim količinama podataka.



Pojava dubokog učenja može se pripisati trima primarnim razvojima poslednjih godina – dostupnosti podataka, brzom računarstvu i algoritamskim poboljšanjima.

- Podaci:** Sada imamo ogromne količine podataka, zahvaljujući internetu, senzorima svuda oko nas i brojnim satelitima koji svakodnevno snimaju ceo svet.

•**Dostignuća u računarstvu:** Imamo moćne računarske resurse zahvaljujući računarstvu u oblaku (cloud computing) i jedinicama za obradu grafike (GPU) koje su postale moćnije nego ikad i koje su pojeftinile zahvaljujući industriji igara.

•**Algoritamska poboljšanja:** Konačno, istraživači su sada rešili neke od najizazovnijih aspekata obuke dubokih neuronskih mreža kroz algoritamska poboljšanja i mrežne arhitekture.

Neki od primera upotrebe DL su :Virtualni asistent(Cortana)-razumevanje govora,prevodenje sa jednog jezika na drugi(google translate),chat i service botovi(pr.Rea),autonomna vozila(prepoznavanje znakova na putu cak i kad su pokriveni snegom),prepoznavanja lica(sistem nadzora na ulicama),personalizovani shoping i zabava(sistemi preporuke koje koriste Amazon,Netflix,Youtube...).

## Šta je Geo AI? Objasniti vezu između GIS-a i AI.

Geo AI predstavlja primenu veštačke inteligencije (AI) u geoprostornom kontekstu. Ovo područje se bavi korišćenjem AI tehnika i algoritama za analizu, interpretaciju i obradu geoprostornih podataka.

Definicija veštačke inteligencije po Andrew Moore-u je "Veštacka inteligencija(AI) je naučna i inženjerska disciplina programiranja računara da se ponašaju na način za koji se do nedavno mislilo da je potrebna ljudska inteligencija". AI uključuje:Computer Vision,Language Processing,Creativity,Summarization.

Veza između GIS-a i AI leži u tome što AI tehnike i algoritmi mogu biti primjenjeni na geoprostorne podatke kako bi se dobile nove informacije i otkrili skriveni obrazci(primeri:predikcija trajektorije vozila,indoor navigacija,digitalizacija analognih mapa).

Primena metoda veštačke inteligencije u GISu(AI GIS) trenutno predstavljanja važan pravac istraživanja u oblasti geoprostornih podataka.

Poslednjih godina,AI GIS je postepeno postao glavni fokus istraživanja i primene u geonaukama.Međutim većina studija se uglavnom fokusira na pojedinačne scenarije i slučajevе upotrebe,a retko uključuje sveobuhvatno istraživanje tehnološkog sistema koji podrazumeva AI GIS.

Kada se dobiju rezultati na bazi AI algoritama, GIS se tada može koristiti za dalju obradu i prostorne analize takvih dobijenih podataka. Prevashodno se misli na metode vizualizacije geopodataka i geoprostorne analize.(u prevodu-Nakon što su primjenjeni AI algoritmi i dobijeni rezultati, GIS pruža mogućnost da se ti rezultati prikažu na geografskim mapama, što omogućava bolje razumevanje i interpretaciju podataka).

Primena AI u GIS-u može doneti brojne prednosti. Na primer, AI može pomoći u automatskom prepoznavanju i klasifikaciji objekata na satelitskim snimcima, kao što su zgrade ili putevi.

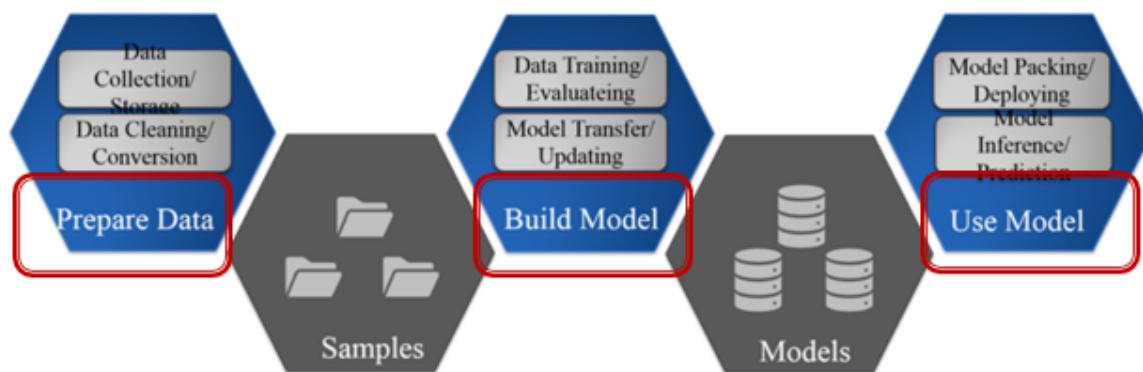
## **Navesti primere primene geoprostornog mašinskog i dubokog učenja.**

Primeri su navedeni u pitanjima šta je mašinsko učenje i šta je duboko učenje.

### **Šta je AI model i objasniti Geo AI workflow.**

AI model je matematički model koji se koristi u veštačkoj inteligenciji kako bi se obavljali određeni zadaci. Model se kreira putem procesa obuke na velikom skupu podataka, pri čemu algoritam mašinskog učenja ili dubokog učenja prilagođava parametre modela kako bi naučio da prepoznaže obrasce, izvlači značajke ili donosi predikcije.

GeoAI tok rada(workflow) uključuju osnovne procese kao što su priprema podataka, kreiranje modela, primenu/korišćenje modela.



Prvi korak je prikupljanje i priprema geoprostornih podataka. To može uključivati prikupljanje podataka iz različitih izvora, čišćenje podataka, transformaciju formata i standardizaciju podataka.

Drugi korak je kreiranje modela. To obuhvata odabir arhitekture modela i obuku modela. Treći korak je primena modela, tj korišćenje, a tu spada pikupljanje novih podataka i predviđanje ponašanja modela nad novim podacima.

## **Navesti alate i programske pakete/jezike koji se koriste za Geo AI.**

Postoje razni alati, programski paketi i jezici koji se koriste za Geo AI u zavisnosti od specifičnih potreba i aplikacija.

Neki od njih su:

### **ArcGIS:**

ArcGIS je jedan od najpoznatijih i najkorišćenijih softverskih paketa za GIS. On pruža širok spektar funkcionalnosti za analizu, vizualizaciju i obradu geoprostornih podataka. ArcGIS Pro, zajedno sa ArcPy bibliotekom, omogućava korisnicima da primene AI algoritme i analiziraju prostorne podatke.

### **SuperMap:**

SuperMap je kineski GIS softver, zamena za ArcGIS, koji pokriva veći deo kineskog tržišta.

### **QGIS:**

QGIS je besplatan i otvorenog koda softver za GIS koji takođe pruža mogućnosti za obradu i analizu geoprostornih podataka. Ovaj softver se može proširiti putem dodataka i pluginova koji podržavaju AI funkcionalnosti.

### **Python:**

Python je veoma popularan jezik za Geo AI zbog bogatog ekosistema biblioteka i paketa za mašinsko učenje i analizu prostornih podataka. Biblioteke poput TensorFlow, Keras, scikit-learn, NumPy, Pandas, GeoPandas i Rasterio pružaju moćne alate za implementaciju i analizu Geo AI rešenja.

## **Šta je učenje sa i bez nadzora (*supervised* i *unsupervised*) i navesti i objasniti neke algoritme iz svake grupe.**

ML algoritmi se klasificuju u dve grupe: algoritmi učenja sa nadzorom (supervised) i algoritmi učenja bez nadzora (unsupervised).

Supervised learning (učenje sa nadzorom) je grana ML koja se odnosi na zaključivanje funkcije koja preslikava ulaz na izlaz na osnovu obučavajućeg skupa podataka. Obučavajući skup se sastoji od para (input, output). Input je obično vektor, a output predstavlja željenu vrednost izlaza koji treba da se dobije kao rezultat funkcije. U zavisnosti od vrste izlaza (output/target), možemo grubo podeliti supervised learning u dve kategorije:

- **Klasifikacija – izlaz je kategorija/diskretna vrednost** (primer: klasifikacija snimaka)
- **Regresija – izlaz je kontinualna vrednost** (image masking)

Neki algoritmi iz grupe supervised learning su: linear regression, logistic regression, k-nearest neighbors, decision tree, random forest, support vector machine.

Unsupervised learning (učenje bez nadzora) zaključuje iz neoznačenih podataka (nema obučavajući skup) funkciju koja opisuje skrivenе strukture u podacima. U ovom tipu učenja mašina mora sama da procesira ulazne podatke i proba da izvede zaključak o izlazu.

Vrste ML algoritama bez učenja su dimension reduction (metode redukcija dimenzije), kao što su PCA, t-SNE (PCA se generalno koristi u preprocesingu, a t-SNE u vizualizaciji podataka), dok napredniju vrstu predstavlja clustering, koji trazi skrivenе obrazce u podacima i na osnovu njih pravi predviđanja (primeri: K-mean clustering, hidden Markov modeli...).

Vrste DL algoritama učenja bez nadzora su: representation learning - ima za cilj da destikuje reprezentativno svojstvo na visokom nivou koje je korisno za neke zadatke, i generative models - nameravaju da reprodukuju ulazne podatke iz nekih skrivenih parametara.

Primeri algoritama učenja bez nadzora: clustering, dimension reduction, density estimation, market basket analysis, generative adversarial networks.

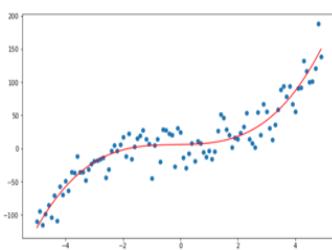
## Navesti osnovne grupe i primere ML i DL algoritama.

Osnovne grupe ML algoritama su: regresija (regression), klasifikacija (classification), clustering (klastering).

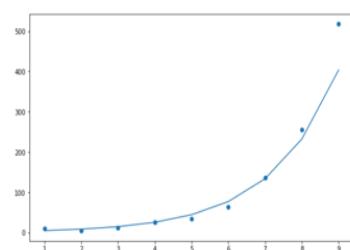
Algoritmi regresije služe za predviđanje kontinualnih vrednosti. Bazirano na matematičkoj funkciji koju koristi postoje algoritmi regresije: linearne, logističke, polinomijalne, eksponencijalne, logaritamske...

Linearna regresija je statistički pristup koji modelira odnos izmedju ulaznih vrednosti (nezavisne promjenljive) i rezultata (zavisne promjenljive).

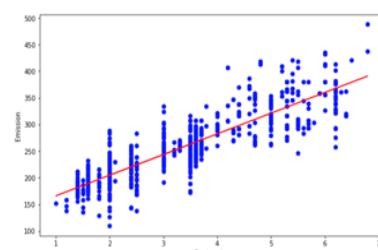
Kada podaci ne prate linearni trend, već polinomski, koristi se polinomska regresija.



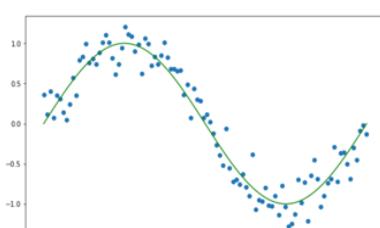
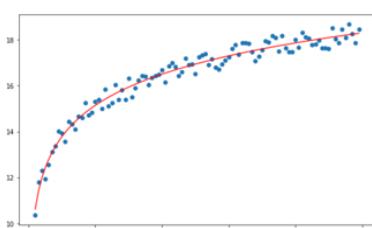
Polinomska



Eksponencijalna



Linearna



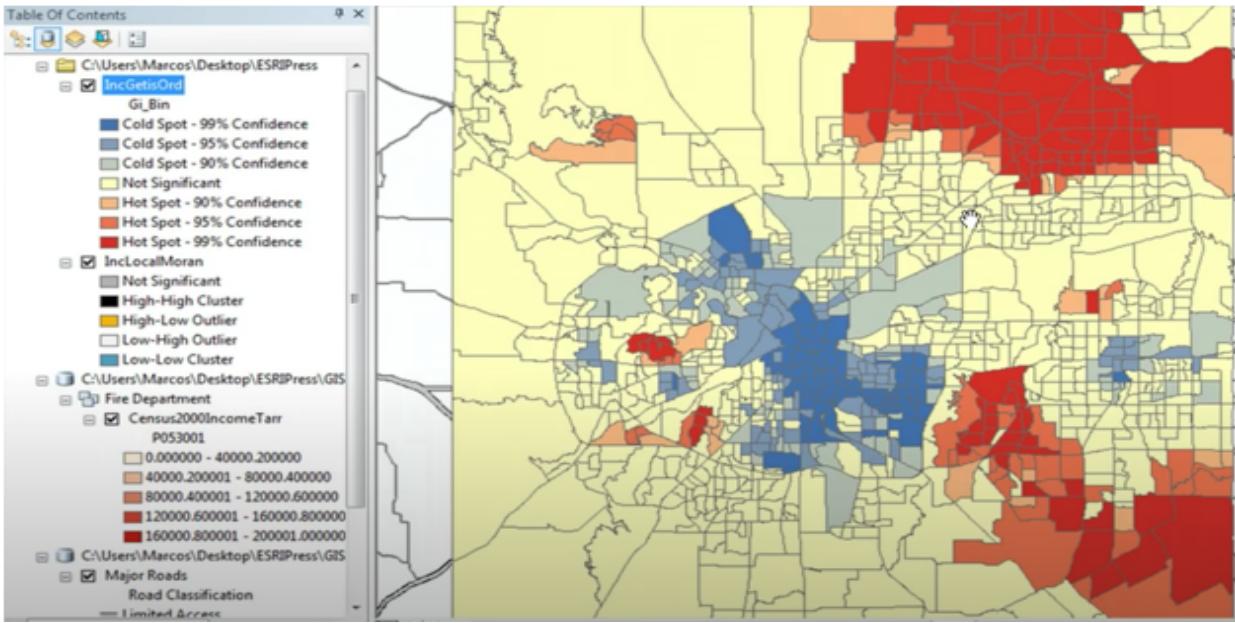
Logarita      Sinusoidalna

U algoritme klasifikacije spadaju:KNN(K-Nearest Neighbors),Decision Tree,Random Forest,Support Vector Machine,Naive Bayes.

Jedan od primera algoritama klasifikacije jeste:ekstrakcija otiska objekata.Jedna od čestih primena DL algoritama za feature extraction je ekstrakcija 2D poligona koji predstavljaju otisak zgrada (Building Footprint Extraction)



I 3.grupa su klastering algoritmi u koje spadaju:K-means,DBSCAN,Mean Shift,Hierarchical.Primer klasteringa jeste analiza grupisanja prema veličini prihoda.



Ostali algoritmi su: Association algoritmi(koriste se za povezivanje dogadjaja ili stvari koji se istovremeno pojavljuju), Anomaly Detection(Otkrivanje anomalija koristimo za otkrivanje abnormalnih aktivnosti i neobičnih slučajeva poput otkrivanja prevara), Sequence Pattern Mining(koristimo sekvensijalno iskopavanje uzoraka za predviđanje sledećih događaja podataka između primera podataka u nizu), Dimensionality Reduction (Smanjenje dimenzionalnosti koristimo za smanjenje veličine podataka da bismo iz skupa podataka izdvojili samo korisne karakteristike), Recommendation Systems(Algoritme za preporuke koristimo za izradu endžina za preporučivanje (primer Netflix, Amazon, YouTube...)).

## Šta je regresija, kakve vrste regresije postoje i gde se primjenjuje?

Definicija regresije i vrste su navedene u prethodnom pitanju. Ovde ćemo navesti 2 tipa linearne regresije:prosta linearna regresija(predikcija na osnovu samo jednog ulaznog parametra) i multivarijabilna linearna regresija(predikcija na osnovu više ulaznih parametara). Neke od primena su:

Finansije: Regresija se koristi za predviđanje cena hartija od vrednosti, indeksa tržišta, kretanja cena akcija ili valuta. Takođe se koristi za analizu rizika, modeliranje kamatnih stopa, vrednovanje imovine i druge finansijske analize.

**Ekonomija:** Regresija se primjenjuje u ekonomskim studijama za analizu odnosa između ekonomskih varijabli kao što su bruto domaći proizvod (BDP), inflacija, nezaposlenost i drugi faktori.

**Marketing i prodaja:** Regresija se koristi za predviđanje potrošačkih preferencija, analizu tržišnih trendova, optimizaciju cena i planiranje prodaje.

**Medicina:** Regresija se koristi za analizu medicinskih podataka, predviđanje bolesti, procenu rizika, modeliranje efikasnosti lekova, prognoziranje pacijentovog stanja i drugih medicinskih istraživanja.

## **Šta je klasifikacija i gde se primjenjuje? Navesti neke osnovne algoritme za klasifikaciju.**

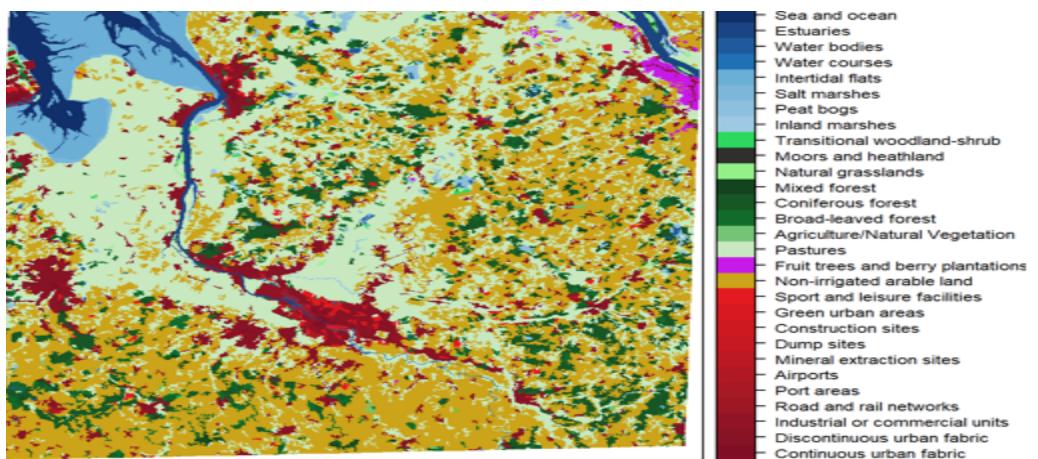
Klasifikacija je tehnika u mašinskom učenju koja se koristi za kategorizaciju objekata ili instanci u unapred definisane klase na osnovu njihovih karakteristika ili atributa.

Algoritmi klasifikacije se koriste za predikciju klase ili kategorije u koje dati entitet spada. U algoritme spadaju: KNN, Decision Tree, Random Forest, Support Vector Machine, Naive Bayes.

Jedan od primera primene algoritama klasifikacije jeste: ekstrakcija otiska objekata. Jedna od čestih primena DL algoritama za feature extraction je ekstrakcija 2D poligona koji predstavljaju otisk zgrada (Building Footprint Extraction).



Primer upotrebe 2: Klasifikacija zemljišnog pokrivača.



- Sea and ocean
- Estuaries
- Water bodies
- Water courses
- Intertidal flats
- Salt marshes
- Peat bogs
- Inland marshes
- Transitional woodland-shrub
- Moors and heathland
- Natural grasslands
- Mixed forest
- Coniferous forest
- Broad-leaved forest
- Agriculture/Natural Vegetation
- Pastures
- Fruit trees and berry plantations
- Non-irrigated arable land
- Sport and leisure facilities
- Green urban areas
- Construction sites
- Dump sites
- Mineral extraction sites
- Airports
- Port areas
- Road and rail networks
- Industrial or commercial units
- Discontinuous urban fabric
- Continuous urban fabric

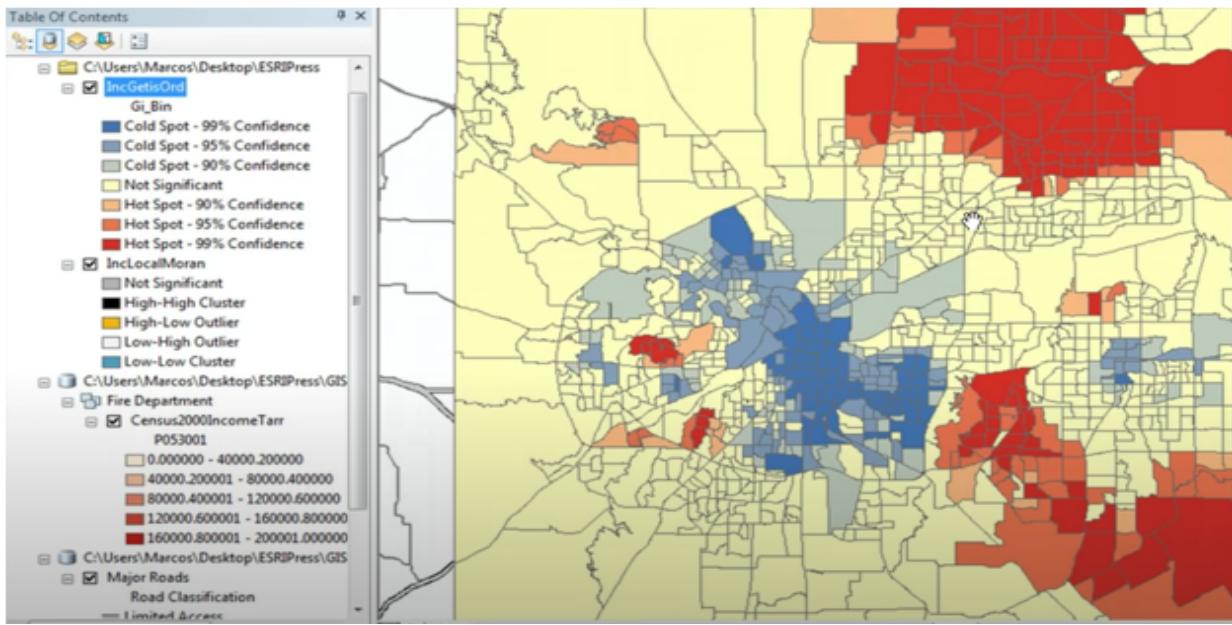
### Primer upotrebe 3:Klasifikacija oblaka tačaka



2D CAD/Vector Layers	3D CAD/Vector Layers
Asfaltni put, poljski put, pešački put, šuma, park, izolovano drveće, žbun, terasa, reka, potok, voda, kanal, ograda, železnica, most, parking, klupa, autobuska stajališta	Kuća, zgrada, stambeno-poslovna zgrada, pomoćna zgrada, rasveta, stub, dalekovodi, saobraćajni znak

## Šta je klastering i gde se primjenjuje? Navesti neke osnovne algoritme za klastering?

Klastering (eng. clustering) je tehnika u mašinskom učenju koja se koristi za grupisanje sličnih objekata ili instanci u skupove, poznate kao klasteri, na osnovu njihovih karakteristika ili atributa. Cilj klasteringa je otkriti prirodne strukture i oblike u skupu podataka bez prethodno definisanih klasa ili oznaka. Klastering algoritmi se koriste za sumiranje ili strukturiranje podataka. Uključuju: K-means, DBSCAN, Mean Shift, Hierarchical. Primer klasteringa jeste analiza grupisanja prema veličini prihoda.



## Šta je PCA analiza i za šta se koristi?

PCA (Principal Component Analysis), ili analiza glavnih komponenti, je tehnika smanjenja dimenzionalnosti koja se koristi za transformaciju skupa podataka sa visokim brojem atributa u manji broj novih varijabli, poznatih kao glavne komponente. Cilj PCA analize je smanjiti kompleksnost podataka, očuvajući što više varijabilnosti i informacija. PCA spada u grupu metoda za preprocesing, odnosno pripremu podataka za regresiju, klasifikaciju i klastering.

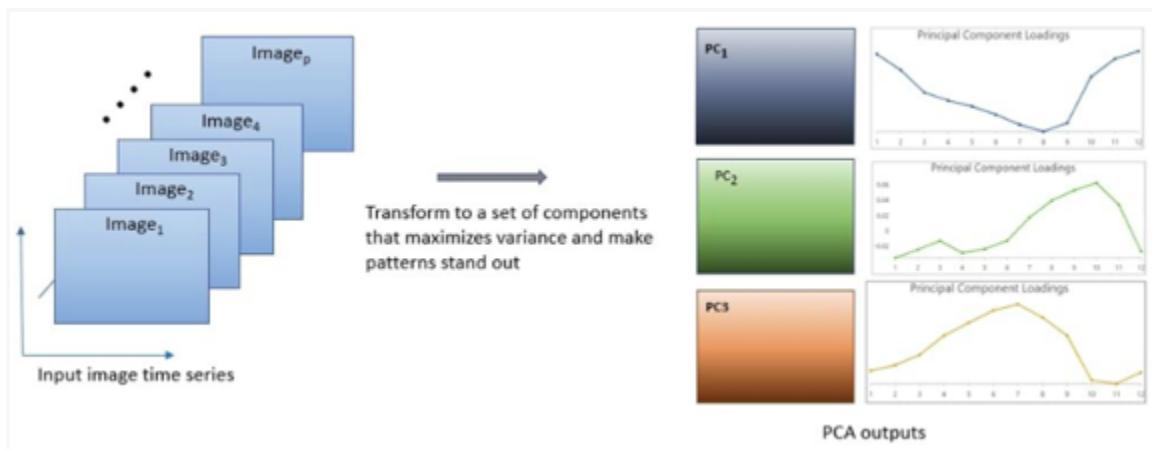
PCA analiza u ArcGis okruženju

ArcGIS sadrži alat multidimenzionalne PCA ([Multidimensional Principal Components tool](#)) u okviru Image Analyst toolboxa. Multidimensional Principal Components tool alat uzima višedimenzionalni raster kao ulaz – to može biti vremenska serija slika ili serija slika zasnovana na Z dimenziji.

Alat pravi tri izlaza analize:

- tabelu sopstvenih vrednosti koja omogućava da se odluči koliko komponenti treba da se zadrži ili odbaci bez gubitka važnih informacija;
- glavne komponente, uskladištene kao multi-band raster, koji predstavljaju preovlađujuće prostorne obrasce u ulaznim podacima;
- i tabelu učitavanja koja pokazuje težine kojima svaki raster doprinosi glavnim komponentama.

Transformacija vremenskih serija slika u PCA izlaz:



Može se koristiti za:

- Smanjenje dimenzionalnosti: PCA se često koristi za smanjenje dimenzionalnosti skupa podataka.
- Identifikacija dominantnih varijabli: PCA može pomoći u identifikaciji dominantnih varijabli ili atributa u skupu podataka.
- Eliminacija koreliranih atributa: Ako imate skup podataka koji sadrži visoko korelirane atribute, PCA može se koristiti za identifikaciju glavnih komponenti koje ne pokazuju korelaciju.
- Prepoznavanje latentnih faktora: PCA se koristi za prepoznavanje latentnih faktora u podacima koji mogu biti teško uočljivi.

## Šta je scikit-learn?

Scikit-learn je besplatna biblioteka za mašinsko učenje za programski jezik Python. Ima većinu algoritama za klasifikaciju, regresiju i klasterisanje i radi sa Python numeričkim bibliotekama kao što su NumPy SciPy.

Scikit-learn python biblioteka podržava algoritme: klasifikacije, regresije i klasteringa.

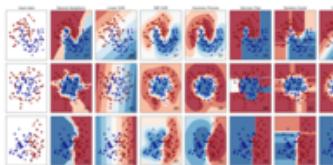
Takođe podržava algoritme: redukcija dimenzija (dimensionality reduction), selekcija modela (model selection) i preprocesiranja (preprocessing).

Takođe podržava metode evaluacije performansi: Tačnost, preciznost, odziv, F1-skora.

### Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.  
**Algorithms:** SVM, nearest neighbors, random forest, and more...

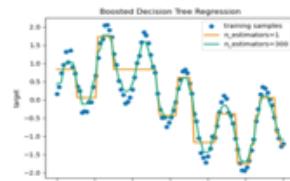


Examples

### Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.  
**Algorithms:** SVR, nearest neighbors, random forest, and more...



Examples

### Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes  
**Algorithms:** k-Means, spectral clustering, mean-shift, and more...

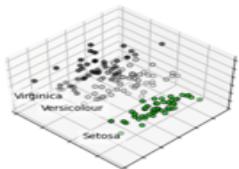


Examples

### Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency  
**Algorithms:** PCA, feature selection, non-negative matrix factorization, and more...

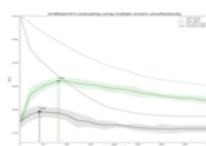


Examples

### Model selection

Comparing, validating and choosing parameters and models.

**Applications:** Improved accuracy via parameter tuning  
**Algorithms:** grid search, cross validation, metrics, and more...

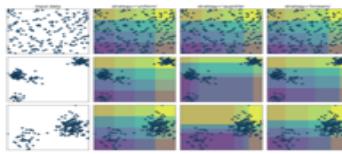


Examples

### Preprocessing

Feature extraction and normalization.

**Applications:** Transforming input data such as text for use with machine learning algorithms.  
**Algorithms:** preprocessing, feature extraction, and more...



Examples

## Objasniti regresiju u ArcGIS okruženju upotrebom OLS metode.

Regresiona analiza u ArcGIS Insights alatu omogućava da se kreira **model regresije** koji koristi **odnos između zavisne promenljive i jedne ili više promenljivih** koje na nju utiču, a zatim koristi taj model za predviđanje vrednosti. Metod regresione analize koju Insights koristi naziva se Obični najmanji kvadrati (**Ordinary Least Squares – OLS**). Obuhvata 4 koraka:

- 1.Učitavanje podataka
- 2.Kako istražiti i izabrati odgovarajuće promenljive
- 3.Kreiranje i evaluacija regresionog modela
- 4.Korišćenje napravljenog modela za predikciju promenljivih u drugim skupovima podataka

1.korak-učitavanje podataka

## Merenja sa bova

Merenja sa plutača na **Velikim jezerima** koje održava Nacionalni centar za plutače NOAA i Odeljenje za ribarstvo i okeane Kanade sumirana su na nekoliko vremenskih skala **za brzinu vetra, smer, srednju visinu i maksimum talasa, dominirajući period talasa, srednji period talasa, srednji pravac talasa, temperatura vazduha i temperatura vode** za svaki zabeleženi period plutače.

mean_monthly_buoy_stats						
+ Field	fx	IF(AtmoPress_Mean<>'9999', VALUE(AtmoPress_Mean))				
MnWvE ↑	AtmoPr ↑	AirTemp ↑	WtrTerr ↑	AirTemp ↑	WaterTemp ↑	AtmoPressure ↑
9999	9999	9999	9999			
9999	9999	9999	9999			
9999	9999	9999	9999			
9999	9999	9999	9999			
9999	9999	9999	9999			
9999	9999	9999	9999			
9999	22.9842519685	9999		22.9843		
9999	21.1492537313	9999		21.1493		
9999	15.4310344828	9999		15.431		
1011.67927928	8.27067669173	9999		8.2707		1,011.6793
1022.17008547	3.30463576159	9999		3.3046		1,022.1701
1014.18275862	-0.450704225...	9999		-0.4507		1,014.1828
1016.5047619	-3.83333333333	9999		-3.8333		1,016.5048
1014.89344262	-5.64462809917	9999		-5.6446		1,014.8934

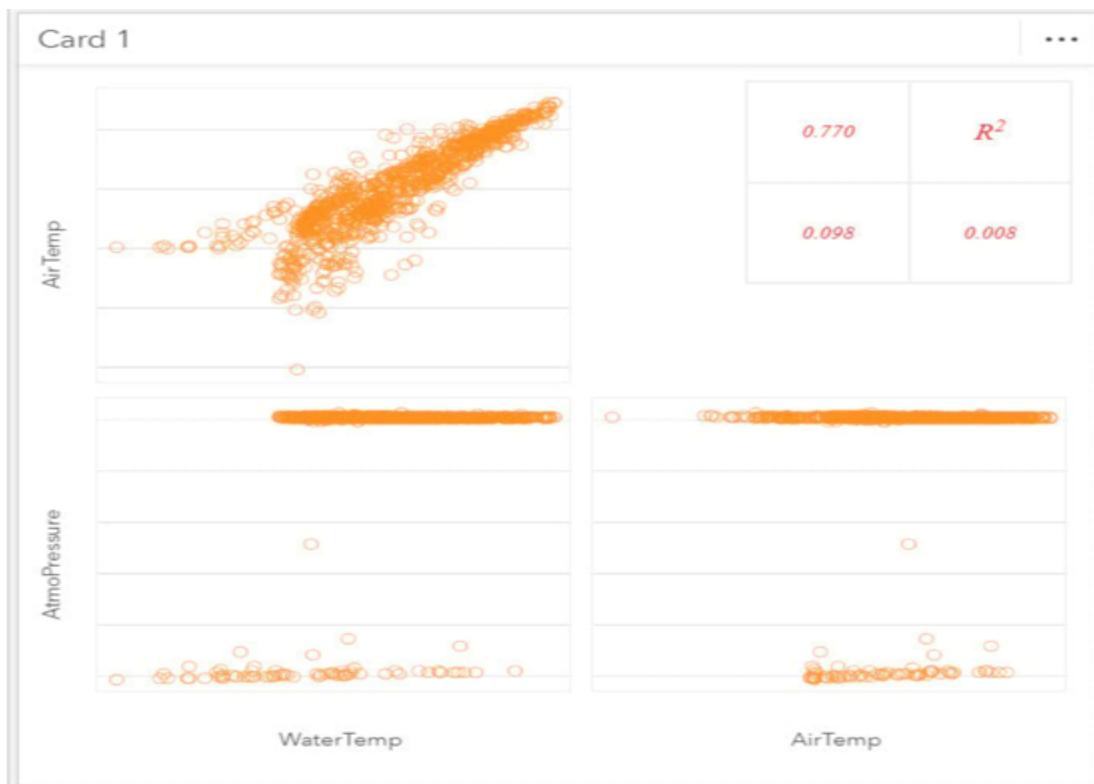
## 2.korak-istraživanje i izbor odgovarajućih promenljivih

Nakon učitavanja podataka, sledeći korak je istraživanje tih podataka da se odredi koje će se promenljive koristiti za regresioni model.Za model regresije potrebna je zavisna varijabla i jedna ili više nezavisnih varijabli koje je objašnjavaju (u vezi su sa njom).Zavisna varijabla će biti kolona koju želimo da objasnimo svojim modelom, a nezavisne varijable će se koristiti za objašnjenje te varijable.Cilj je da se napravi model koji će pomoći u predviđanju temperature vode, stoga **WaterTemp** kolona u tabeli atributa predstavlja zavisnu varijablu.Da bi odredili koje varijable imaju uticaj na zavisnu varijablu može se koristiti **scatter plot dijagram** ili **histogrami**.Mora postojati linearna relacija između zavisne varijable i nezavisnih varijabli.

Na scatter plot matrici može se vizuelno analizirati da li postoji linearna relacija i pogledati koje su  $R^2$  vrednosti.  **$R^2$  vrednost meri jačinu relacije.** Što je ova vrednost bliže 1, to je veza jača.

Takođe, bitno je proveriti da li postoje odstupanja. **Odstupanja (outliers)** štrče od preovlađujućeg obrasca na grafikonu i mogu biti pogrešne mere ili retki događaji koji bi iskrivili rezultate. Stoga je neophodno ukloniti ove podatke koji drastično odstupaju od ostalih kako bi se postigla veća linearost i proizveo bolji model.

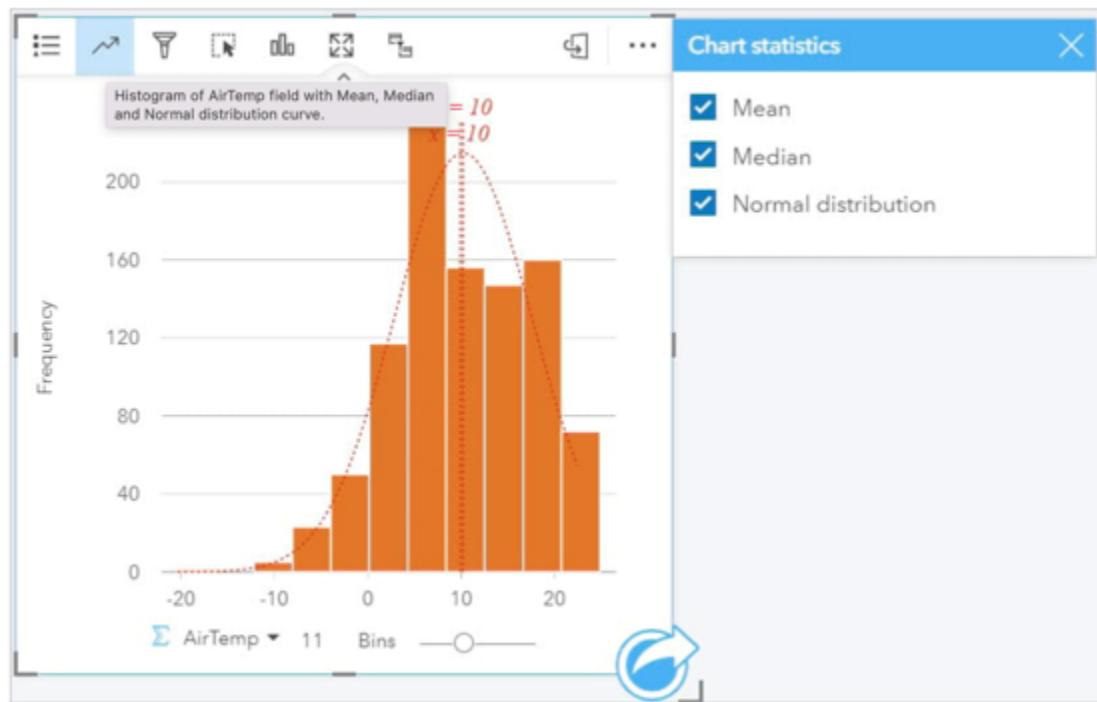
Posmatranjem kreirane scatter plot matrice primećuje se linearna relacija između atributa **Water Temp** i **Air Temp**. Visoka  $R^2$  vrednost potvrđuje da je relacija jaka.



U ovom primeru postoji samo jedna odgovarajuća nezavisna varijabla (Air Temp).

- Međutim, kada ih postoji više potrebno je utvrditi da među njima ne postoji linearna relacija.

Drugi bitan faktor za razmatranje je da varijable imaju normalnu distribuciju, što se može utvrditi sa histograma. U ovom primeru se vidi da varijabla nije savršeno normalno distribuirana, što se potvrđuje dodavanjem krive normalne distribucije na histogram.



### 3.korak-kreiranje i evaluacija modela

Nakon pokretanja modela dešava se:

- Dodat lejer **Avg Standardized Residual**
- Dodat nov skup podataka **Predicted WaterTemp 1**  
Sadrži sve varijable iz prvobitnog skupa podataka i 3 nove: Estimated, Residual, i Standardized Residual.
- Dodat regresioni model na panel

### Evaluacija rezultata modela

Regression Model

- 1 Choose a layer  
mean\_monthly\_buoy\_stats
- 2 Choose a dependent variable  
WaterTemp
- 3 Choose explanatory variables  
1 variable selected

AirTemp

fx Regression Model 1

y = 2.09 + 0.838new\_field  
Prediction equation

R<sup>2</sup>: 0.77039248  
Measures model variability

Adjusted R<sup>2</sup>: 0.7701538  
Predictor adjusted model variability

Durbin-Watson Test: 0.75961391  
Describes model autocorrelation

p-Value: 0  
Determines significance of the model

Residual standard error: 3.373 on 962 df  
Measures model fit

F statistic: 3,227.758 on 1 and 962 df  
Reference value for significance test

[View confidence intervals](#)

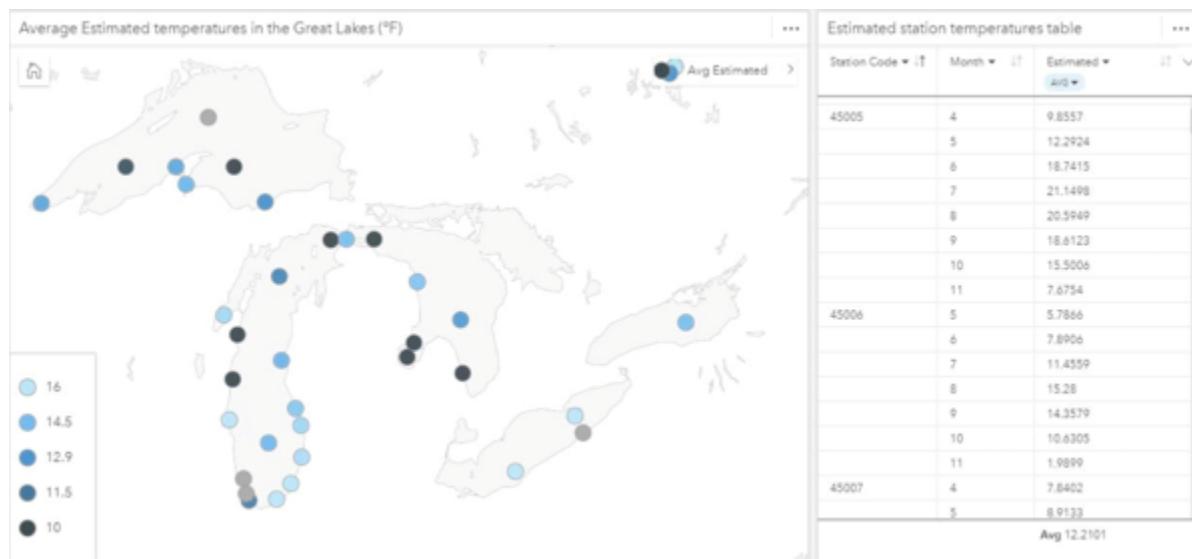
- $R^2$  je 0.77, relativno blizu 1, što je OK.

-Durban-Watson Test je 0.75, što je malo niže nego što treba, jer bi idealno bilo između 1.5 i 2.5.

-p-value modela je 0 što je idealno.

#### 4.korak-predviđanje rezultata

Kreirani regresioni model se može koristiti za drugi skup podataka u kom nedostaje vrednost za temperature vode



## Šta je kompjuterski vid i gde se koristi?

Jedna od oblasti veštačke inteligencije u kojoj se duboko učenje pokazalo izuzetno dobro jeste **kompjuterski vid**, ili **sposobnost računara da vidi**.

Kompjuterski vid je disciplina veštačke inteligencije koja se bavi razumevanjem i interpretacijom vizuelnih podataka pomoću računara. Ova disciplina ima za cilj omogućiti računarima da razumeju i interpretiraju vizuelne informacije na sličan način kao ljudi. Ovo je posebno korisno za GIS, pošto se **satelitski snimci i snimci iz aviona i sa drona**, proizvode brzinom koja onemogućava analizu i izvlačenje uvida tradicionalnim sredstvima.

Primeri upotrebe:

## Deep Learning: Computer Vision Use Cases



Klasifikacija slike je proces dodeljivanja kategorija određenoj slici na osnovu njenih vizuelnih karakteristika. Ova tehnika se koristi u kompjuterskom vidu i mašinskom učenju kako bi se automatizovano identifikovali i klasifikovali objekti ili regioni interesa na slikama.

Kod detekcije objekata, računar treba da pronađe objekte na slici, kao i njihovu lokaciju.

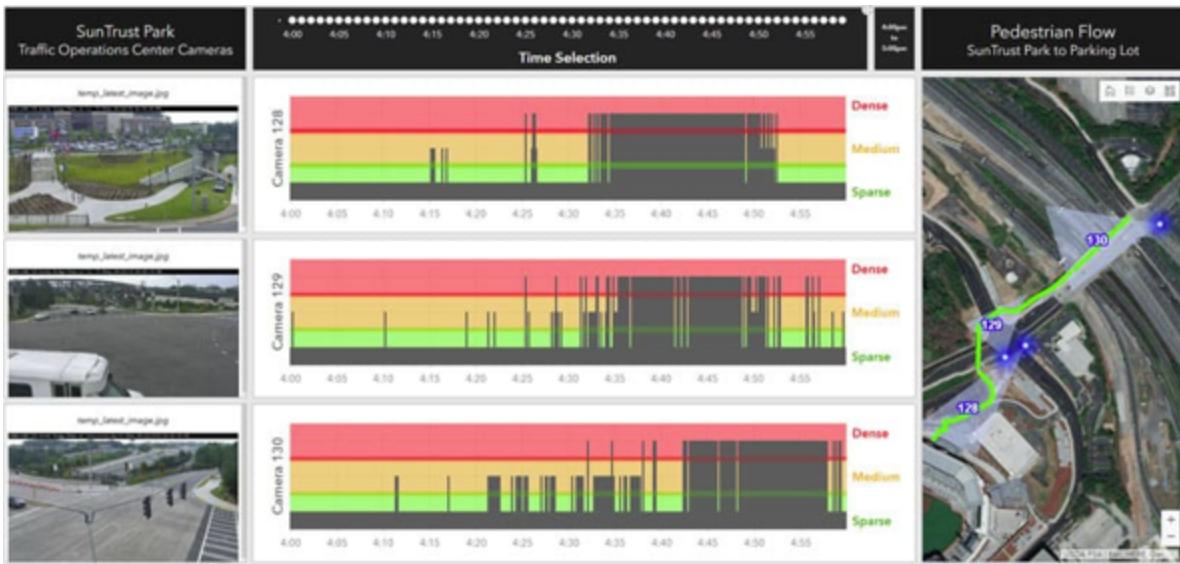
Još jedan važan zadatak kompjuterskog vida je semantička segmentacija, u kojoj se **svaki piksel slike klasifikuje kao pripadajući određenoj klasi**.

Kod segmentacije instance je reč o preciznoj detekciji objekta u kojoj je označena precizna granica svake instance objekta.

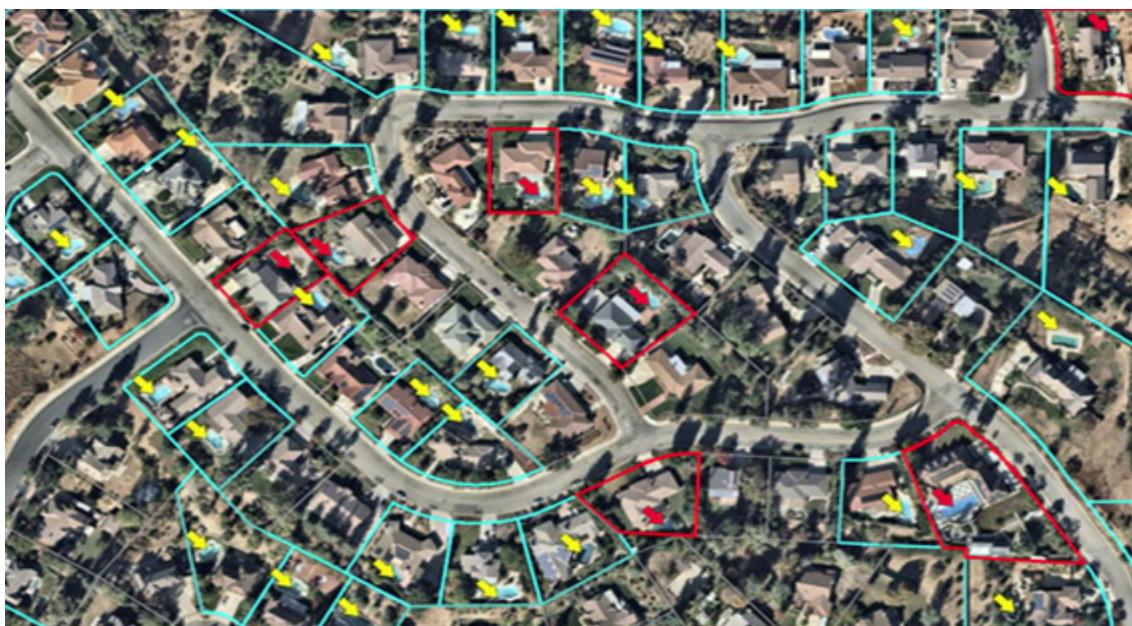
### Objasniti klasifikaciju slike i detekciju objekata.

Klasifikacija slike je proces dodeljivanja kategorija određenoj slici na osnovu njenih vizuelnih karakteristika. Ova tehnika se koristi u kompjuterskom vidu i mašinskom učenju kako bi se automatizovano identifikovali i klasifikovali objekti ili regioni interesa na slikama.

Klasifikacija aktivnosti pešaka može se koristiti za planiranje upravljanja pešacima i saobraćajem tokom javnih događaja.



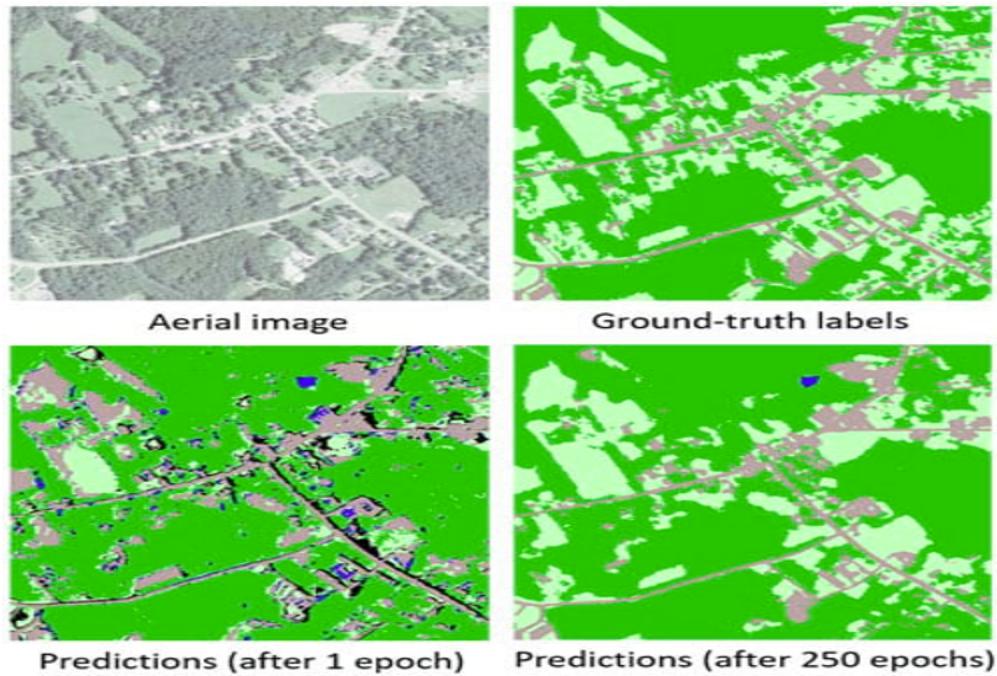
Kod detekcije objekata, računar treba da pronađe objekte na slici, kao i njihovu lokaciju. Ovo je veoma važan zadatak u GIS-u – pronalaženje onoga što se nalazi na satelitskim, vazdušnim snimcima ili snimcima dronom, njihovo lociranje i crtanje na mapi. Ovo se može koristiti za mapiranje infrastrukture, otkrivanje anomalija (**anomaly detection**) i izdvajanje geo-objekata (**feature extraction**).



Primer detekcije bazena

## Šta je semantička segmentacija, a šta segmentacija instanci?

Kod semantičke segmentacije se svaki piksel slike klasificiše kao pripadajući određenoj klasi.U GIS-u, semantička segmentacija se može koristiti za **klasifikaciju zemljišnog pokrivača** ili za **izdvajanje putnih mreža iz satelitskih snimaka**, itd.



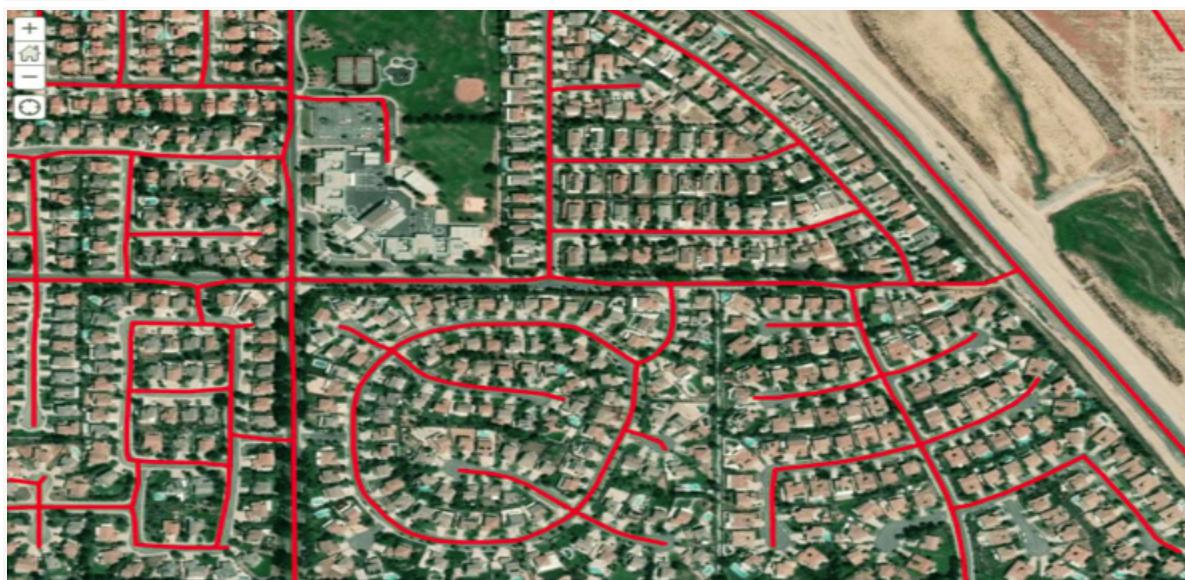
Kod segmentacije instanci je reč **preciznijoj detekciji objekta** u kojoj je **označena precizna granica svake instance objekta**.Segmentacija instance se može koristiti za zadatke kao što je poboljšanje osnovnih mapa (podloga).Na ovaj način se mogu ekstrahovati **2D poligoni koji predstavljaju zgradu** ili rekonstruisti **3D zgrade iz LIDAR podataka**.



## Šta je **feature extraction**?

Feature extraction je proces izdvajanja relevantnih karakteristika ili "obeležja" iz podataka. U kontekstu obrade signala ili analize podataka, feature extraction je tehnika koja se koristi za transformaciju sirovih podataka u reprezentaciju koja je pogodna za dalju analizu ili obradu.

U radu sa satelitskim snimcima, važna primena dubokog učenja je kreiranje digitalnih mapa automatskim izdvajanjem geo-objekata kao što su putne mreže ili 2D objekti (building footprint). Ovo se postiže primenom obučenog modela dubokog učenja na velikom geografskom području. Ovo može biti posebno korisno za zemlje u razvoju koje nemaju visokokvalitetne digitalne mape ili u oblastima u kojima su izgrađeni noviji objekti.



Modeli segmentacije instanci kao što je Mask R-CNN posebno su korisni za segmentaciju **2D otiska zgrada** i mogu pomoći u kreiranju otiska zgrada bez potrebe za ručnom digitalizacijom. Međutim, ovi modeli obično rezultiraju nepravilnim otiscima zgrada koji ne liče na obične zgrade sa ravnim ivicama i pravim uglovima.

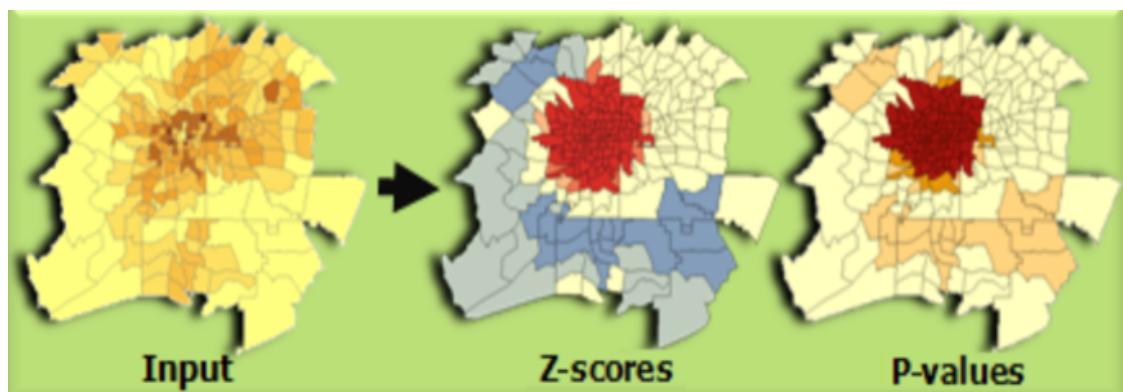
Korišćenje ArcGIS alata **Regularize Building Footprint**, može pomoći u vraćanju ravnih ivica i pravih uglova neophodnih za precizan prikaz 2D otiska zgrade.



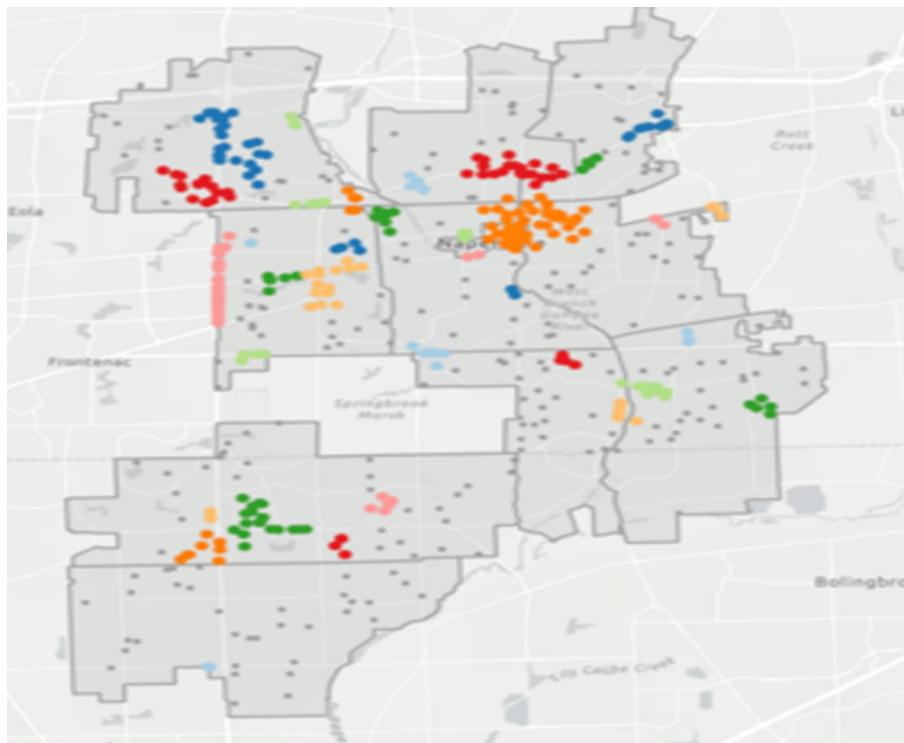
## Objasniti *Hot Spot* analizu i *Density-based* klastering.

Za zadate tačke incidenta ili ponderisane geo-objekte (tačke ili poligoni), kreira mapu statistički značajnih vrućih i hladnih tačaka koristeći Getis-Ord Gi\* statistiku. On procenjuje karakteristike ulazne klase obeležja da bi proizveo optimalne rezultate. Hot Spot analiza izračunava statistiku Getis-Ord Gi\* za svako obeležje u skupu podataka. Rezultirajući z-rezultati i p-vrednosti (p – verovatnoća, z – standardna devijacija) govore gde se prostorno grupišu obeležja sa visokim ili niskim vrednostima. Ovaj alat funkcioniše tako što posmatra svako obeležje u kontekstu susednih obeležja. Obeležje sa visokom vrednošću je zanimljivo, ali možda nije statistički značajna vruća tačka. Da bi bila statistički značajna vruća tačka, obeležje će imati visoku vrednost i biti okruženo drugim obeležjima sa visokim vrednostima. Lokalni zbir za obeležje i njegove susede se poređi proporcionalno zbiru svih obeležja; kada je lokalni zbir veoma različit od očekivanog lokalnog zbira, i kada je ta razlika prevelika da bi bila rezultat slučajnog slučaja, dobija se statistički značajan z-rezultat.

Primer: Hot Spot analiza je standardna praksa koju koriste mnoge policijske agencije da identifikuju grupe incidenata koji se dešavaju u njihovoј nadležnosti. Identifikovanjem klastera (ili vrućih tačaka) aktivnosti, policija je bolje u stanju da fokusira strategije i taktike sprovođenja na ove klastere i efikasnije reaguje na probleme.



Density-Based Clustering (DBC) je tehnika klasterovanja u mašinskom učenju koja se zasniva na gustini podataka. Ova metoda identificiše klasterovske strukture na osnovu gustine podataka u prostoru, što znači da grupiše tačke koje su gusto raspoređene zajedno, a razdvaja tačke koje su razređene.



## Objasniti **F-score** metriku za evaluaciju AI modela.

F-score je često korištena metrika za evaluaciju i ocenu tačnosti AI modela. F-skor (F-score) je mera koja se koristi za procenu performansi klasifikacionih modela, posebno u situacijama kada postoji neuravnoteženost između klasa. Ova mera kombinuje preciznost (precision) i odziv (recall) kako bi pružila sveobuhvatnu procenu tačnosti modela. Preciznost predstavlja procenat tačno klasifikovanih pozitivnih instanci u odnosu na sve instanci koje su model označio kao pozitivne. Odziv, poznat i kao osetljivost ili stopa istinito pozitivnih (TPR), predstavlja procenat tačno klasifikovanih pozitivnih instanci u odnosu na sve pozitivne instance u stvarnom skupu podataka.

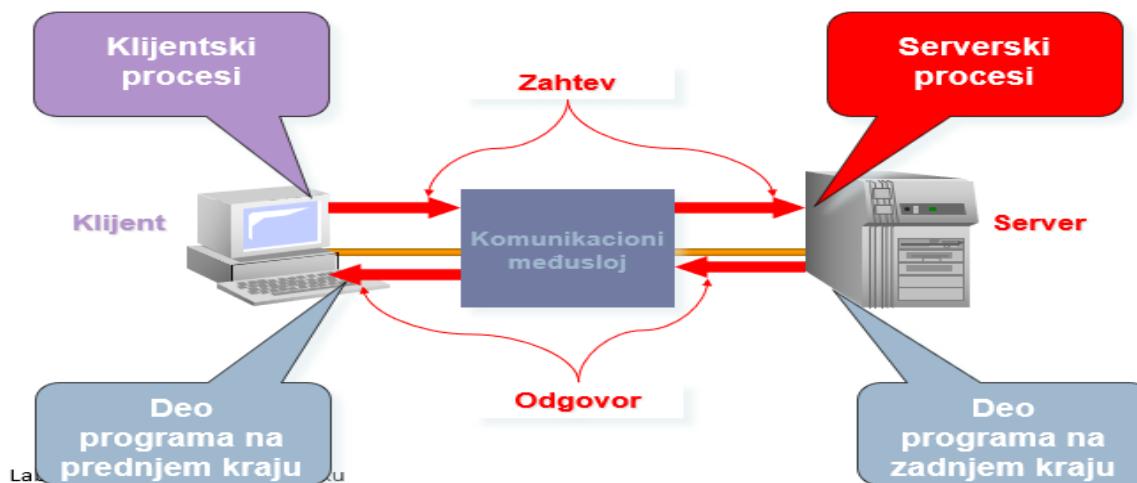
$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \times 100\%, \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100\%$$

$$F1 = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

## Šta je klijent-server arhitektura i kako se vrši komunikacija između klijenta i servera?

K/S model obrade podataka je vrsta distribuirane obrade podataka kod koje se funkcije korisničkog programa raspodjeljuju na najmanje 2 procesa koji međusobno komuniciraju. Mogući tipovi procesa u K/S modelu:-klijentski procesi  
-serverski procesi

Komunikacija između klijenta i servera se odvija putem mreže, koristeći određene protokole i mehanizme komunikacije. Najčešće korišćeni protokol za komunikaciju u klijent-server arhitekturi na webu je HTTP (Hypertext Transfer Protocol). Klijent šalje HTTP zahtev serveru koji sadrži informacije o željenom resursu ili operaciji. Server prima taj zahtev, obrađuje ga i generiše HTTP odgovor koji sadrži tražene informacije ili rezultate operacije. Odgovor se zatim šalje nazad klijentu.



## Šta je servisno-orjentisana arhitektura?

Tradicionalni GIS sistemi su bazirani na aplikacijama koje koriste podatke u određenom formatu koji je vlasništvo nekog proizvođača. Ovi sistemi su "vendor driven", odnosno zavisni su od jednog proizvođača. Rešenje je SOA (Servisno-orientisana arhitektura). Servisno-orientisana arhitektura (SOA, eng. Service-Oriented Architecture) je pristup dizajnu softverskih sistema koji se fokusira na organizaciju funkcionalnosti sistema kao skupa međusobno nezavisnih i ponovno iskoristivih servisa. U SOA arhitekturi, servisi su nezavisne jedinice softvera koje se mogu implementirati, održavati i skalirati nezavisno jedna od druge. Komunikacija između servisa se obično vrši putem mreže, koristeći standardizovane protokole poput HTTP.

## Šta je interoperabilnost?

Interoperabilnost je ključni faktor u troslojnoj arhitekturi jer omogućava efikasnu komunikaciju i integraciju između slojeva sistema.

Troslojna arhitektura se sastoji od tri osnovna sloja: sloj servisa, aplikativni sloj (logički sloj) i sloj podataka (podatkovni sloj). Svaki sloj ima svoju specifičnu ulogu i odgovornosti, ali je neophodno da svi slojevi mogu međusobno komunicirati i razmenjivati podatke kako bi sistem funkcionsao kao celina.

Interoperabilnost je ključna za troslojnu arhitekturu jer omogućava da se prezentacioni sloj komunicira sa aplikativnim slojem, a aplikativni sloj sa slojem podataka. Bez interoperabilnosti, slojevi bi bili izolovani i ne bi mogli efikasno sarađivati, što bi ograničilo funkcionalnosti sistema. Stoga, razumevanje i implementacija interoperabilnosti je važan aspekt prilikom dizajniranja i razvoja sistema zasnovanih na troslojnoj arhitekturi.

Neki od ključnih OGC standarda koji se koriste za postizanje interoperabilnosti u geoprostornim sistemima su: WMS, WFS, WCS (značenja navedena u narednim pitanjima).



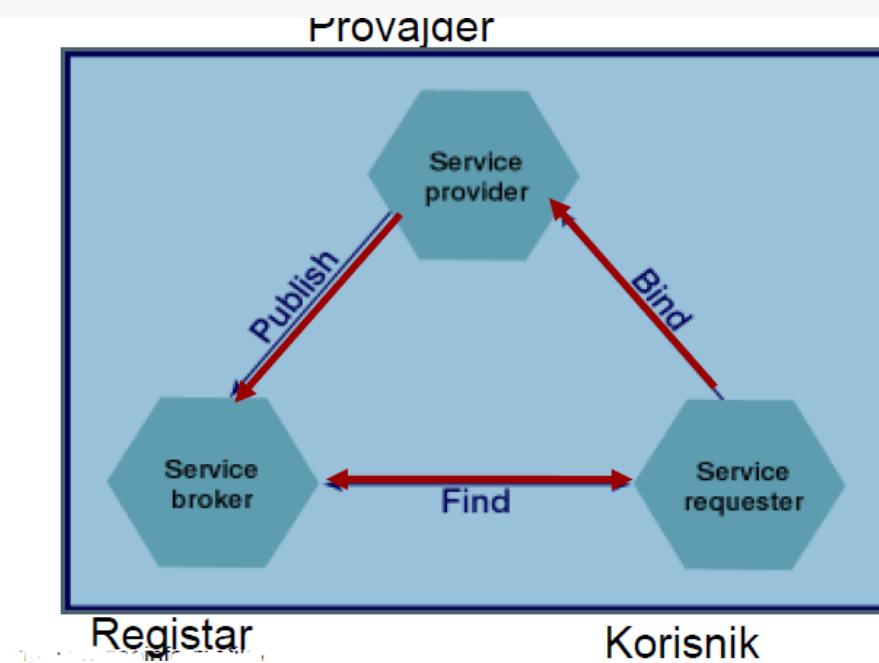
## Objasniti **publish-find-bind** model interakcije u SOA.

Publish-Find-Bind (Objavi-Pronađi-Veži) model interakcije je ključni koncept u servisno-orientisanoj arhitekturi (SOA). Ovaj model opisuje način na koji servisi komuniciraju i međusobno se povezuju u SOA okruženju.

Service Provider (Provajder servisa): Service provider je entitet koji implementira i pruža određene servise. On objavljuje informacije o svojim servisima u registar radi dostupnosti drugim korisnicima. Service provider je odgovoran za implementaciju funkcionalnosti servisa, kao i za pružanje odgovora na zahteve koje dobija od service requestera.

Service Broker (Registrar servisa): Service broker, koji se ponekad naziva i registrar servisa, je centralizovani sistem ili komponenta koja prikuplja, održava i obezbeđuje informacije o dostupnim servisima. Service broker registruje servise koje pružaju service provideri, čuva metapodatke o njima (kao što su opis servisa, interfejsi, podržane operacije itd.) i čini ih dostupnim service requesterima.

Service Requester (Korisnik servisa): Service requester, takođe poznat i kao service consumer, je entitet koji traži i koristi određeni servis. Korisnik servisa pretražuje registrar servisa, koristeći odgovarajuće kriterijume kao što su funkcionalnost, podržane operacije ili druge relevantne informacije. Nakon pronalaska odgovarajućeg servisa, service requester koristi interfejs servisa za slanje zahteva i primanje odgovora od service providera.



## Šta su **WSDL** i **REST** servisi i u čemu je razlika?

WSDL je standardizovani jezik za opisivanje web servisa. On definiše detalje o tome kako pristupiti i koristiti određeni web servis. WSDL koristi XML (format koji koristi tagove) za opisivanje metoda servisa, ulaznih i izlaznih parametara, tipova podataka i komunikacionih protokola.

Representational State Transfer (REST) je softverska arhitektura za distribuirane sisteme kao što je WWW. Iako je WSDL standard namenjen za specifikaciju web

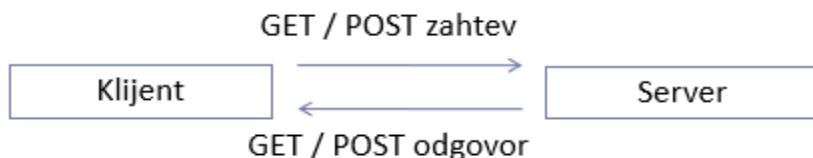
servisa, REST je preuzeo vodeću ulogu na polju geoinformacija, tako da ga je OGC konzorcijum usvojio kao osnovu za razvoj OGC specifikacija.

Osnovna razlika je u tome što WSDL se fokusira na detaljan opis web servisa, koristeći XML za definisanje metoda, tipova podataka i protokola, dok REST se fokusira na jednostavnost i skalabilnost. REST servisi koriste HTTP metode za pristupanje resursima i prenose podatke u formi JSON-a ili XML-a. REST je često povezan sa arhitekturom resursa i omogućava jednostavno deljenje podataka preko mreže.

## Objasniti REST servise.

Representational State Transfer (REST) je softverska arhitektura za distribuirane sisteme kao što je WWW. Iako je WSDL standard namenjen za specifikaciju web servisa, REST je preuzeo vodeću ulogu na polju geoinformacija, tako da ga je OGC konzorcijum usvojio kao osnovu za razvoj OGC specifikacija.

RESTful servisi su servisi bazirani na HTTP protokolu. Koriste HTTP metode kao što su GET i POST za slanje zahtjeva klijenta prema serveru u klijent-server arhitekturi.



```
http://localhost:8080/geoserver/ows?  
service=WFS&  
version=1.0.0&  
request=GetCapabilities
```

REST predstavlja arhitekturni stil, dok RESTful se odnosi na servise koji se pridržavaju principa REST arhitekture.

## Objasniti JSON i GeoJSON format podataka.

JSON (JavaScript Object Notation) je format za razmenu podataka izmedju klijenta i servera-preko REST servisa. Json je tekst, napisan u JavaScript objektnoj notaciji. Univerzalni format na webu: '{ "name": "Petar", "age": 30, "city": "Novi Sad"}'

Kada se razmenjuju podaci između brauzera i servera,podaci mogu biti jedino tekst.Stoga je JSON tekst.Bilo koji JavaScript objekat može da se konvertuje u JSON i pošalje serveru,takođe JSON dobijen sa servera može da se konvertuje u JavaScript objekte.

GeoJSON je format za kodiranje različitih geoprostornih struktura podataka.GeoJSON podržava tipove geometrije kao što su:Point,LineString,Poligon,MultiPoint,MultiLineString i MultiPoligon.

```
{  
  "type": "Feature",  
  "geometry": {  
    "type": "Point",  
    "coordinates": [125.6, 10.1]  
  },  
  "properties": {  
    "name": "Dinagat Islands"  
  }  
}
```

## **Objasniti troslojnu arhitekturu i navesti primere tehnologija/softvera/programskih jezika koje se koriste za realizaciju svakog sloja.**

Troslojna arhitektura je arhitektura koja sadrži 3 sloja:sloj podataka,sloj servisa,aplikativni sloj.

**Sloj podataka** – sadrži podatke u bazi podataka ili fajl sistemu.

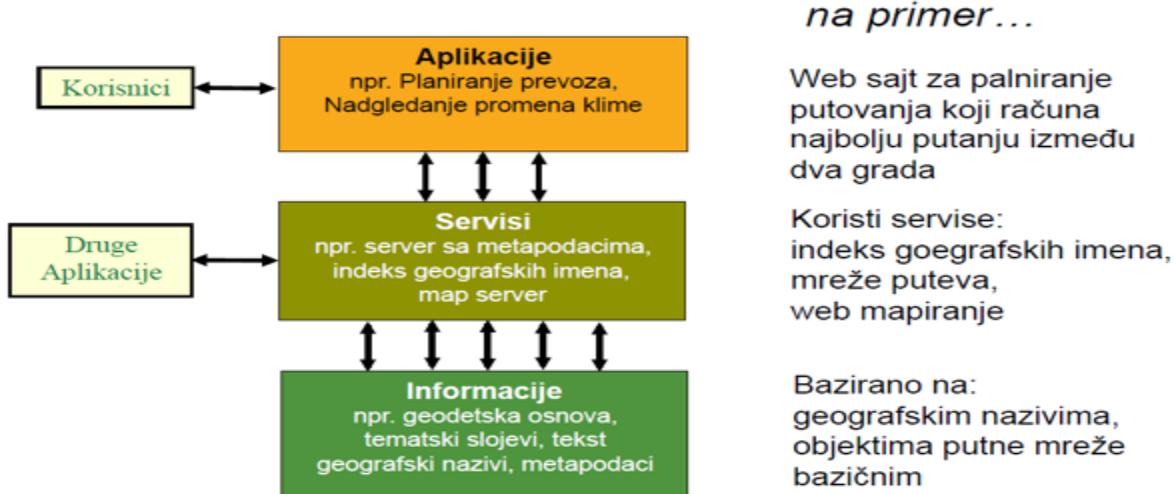
Tehnologije koje se koriste za realizaciju sloja podataka(Oracle,MS SQL server,PostgreSQL/Postgis)

**Sloj servisa** – implementira servise koji posreduju između sloja podataka i aplikativnog sloja, tako što na zahtev korisnika pristupaju sloju podataka, preuzimaju podatke, vrše obradu (u vidu renderovanja mape, filtriranja podataka, transformacije koordinata, itd.) i isporučuju rezultat krajnjem korisniku.

Tehnologije koje se koriste za realizaciju sloja servisa su:Apache, Spring, Jango, REST, WMS...

**Aplikativni sloj** – skup aplikacija koje pristupaju servisima. Ove aplikacije se mogu izvršavati u web braузeru (tzv. tanki klijent – eng. *thin client*) ili mogu biti desktop aplikacije (tzv. debeli klijent – eng. *thick client*) koje imaju podršku za standardne interfejse.

Tehnologije koje se koriste za realizaciju aplikativnog sloja su: Java,C#,Python...



## Šta su geoservisi i objasniti OGC WMS, WFS i WCS.

Geoservisi su softverske komponente ili aplikacije koje omogućavaju pristup, manipulaciju i razmenu geoprostornih podataka preko mreže. Geoservisi su RESTful servisi koji implementiraju OpenGIS standarde (imaju OGC interfejs). WMS, WFS i WCS predstavljaju najpoznatiji OGC standarde (OGC-međunarodni konzorcijum od više od 500 preduzeća, vladinih agencija, istraživačkih organizacija i univerziteta čiji je cilj da geoprostorne (lokacijske) informacije i servise učine FAIR – dostupnim, dostupnim, interoperabilnim i ponovo upotrebljivim).

OGC Web Map Service (**WMS**) – web mape

OGC Web Feature Service (**WFS**) – servis za preuzimanje vektorskih podataka u GML format (koji se može konvertovati u shp)

OGC Web Coverage Service (**WCS**) – servis za preuzimanje rasterskih podataka kao što je ortofoto

## **Navesti vrste geoprostornih servisa.**

Vrste geoprostornih servisa su: 2D servisi, 3D servisi, Geosenzorski servisi i Lokacijsko-bazirani servisi.

2D servisi su servisi za pristup, preuzimanje, vizualizaciju i obradu 2D kartografskog sadržaja

pr. Web mape – OGC Web Map Service

3D servisi omogućavaju pristup, preuzimanje, vizualizaciju i obradu 3D sadržaja.

pr. **WebGL** je verzija OpenGL-a, 3D engine. Omogućava korisniku da izvrši **3D manipulaciju** u web brauzerima.

Geosenzorski ervisi za preuzimanje podataka sa geosenzorskih mreža

**Sensor Web Enablement (SWE)** je skup standardnih enkodinga i web servisa koji omogućavaju:

- Pronalaženje senzora, procesa i opservacija
- Zadavanje zadataka senzorima ili modelima
- Pristup opservacijama i tokovima opservacija (observation streams)
- Publikovanje i prijava na servise za alarmiranje
- Robustni senzorski sistemi i opisi procesa

LBS(Lokacijsko-bazirani servisi) su informacioni servisi dostupni preko mobilnih uređaja putem mobilne mreže i koriste mogućnost korišćenja lokacije mobilnog uređaja. LBS su informacioni servisi za pružanje informacija koje su kreirane, sastavljene, odabrane ili filtrirane uzimajući u obzir trenutne lokacije korisnika ili lokacije drugih osoba ili mobilnih objekata.

## **Šta je *TypeScript?* *TypeScript vs JavaScript.***

**TypeScript** je nadskup JavaScripta. TypeScript je izrađen na JavaScriptu.

Kada se napiše TypeScript kod, kompajlira se u običan **JavaScript** kod upotrebom TypeScript kompajlera. Takav kod se može postaviti u bilo koje okruženje koje izvršava JavaScript. TypeScript fajl koristi **.ts** ekstenziju umesto **.js** ekstenzije koju koriste JavaScript fajlovi.



TypeScript koristi JavaScript sintaksu i dodaje dodatnu sintaksu za podršku tipovima. JavaScript program koji nema sintaksne greške je takođe TypeScript program.

To znači da su svi JavaScript programi i **TypeScript** programi, što je korisno kada se migrira postojeći kod na novu platformu.

Osnovni ciljevi TypeScripta su:

- Uvođenje opcionih tipova u JavaScript
- Implementirajte planiranih svojstava budućeg JavaScript-a, zvanog ECMAScript Next ili ES Next u trenutni JavaScript

## **TypeScript osnovni tipovi, primitivni i objektni.**

U TS, tip je zgodan način da se uputi na različite **atribute** i **funkcije** koje **vrednost** ima. Vrednost je bilo šta što se može dodeliti promenljivoj, npr. broj, string, niz, objekat ili funkcija.

**TypeScript kompajler** koristi tipove da analizira kod i pronađe bagove.

TypeScript tipovi su kategorizovani u:

- Primitivne tipove
- Objektne tipove

**Dve osnovne svrhe** tipova u TypeScriptu su:

- Tipove koristi TypeScript kompajler da analizira kod na greške
- Tipovi omogućavaju da se razume kakve vrednosti su pridružene promenljivama

## Primitivni tipovi:

Name	Description
string	represents text data
number	represents numeric values
boolean	has true and false values
null	has one value: null
undefined	has one value: undefined. It is a default value of an uninitialized variable
symbol	represents a unique constant value

## Objektni tipovi:

Objektni tipovi su funkcije, nizovi, klase...

Može se kreirati custom objektni tip.

## Anotacije tipova i zaključivanje tipova.

TypeScript koristi **anotacije** tipova da eksplisitno specificira tipove za identifikatore kao što su promenljive, funkcije, objekti, itd.

TypeScript koristi sintaksu: tip iza identifikatora kao anotacija tipa, gde tip može biti bilo koji validan tip. Jednom kada je identifikator anotiran tipom, može biti korišćen samo sa tim tipom, inače će kompajler javiti grešku.

```
let variableName: type;  
let variableName: type = value;  
const constantName: type = value;
```

Kada se tip eksplisitno ne anotira, TypeScript može da **zaključi** tip, npr. iz dodeljene vrednosti prilikom inicijalizacije.

U **praksi** treba uvek koristiti zaključivanje tipova što je više moguće.

Na primer, ako se promenljiva counter inicijalizuje brojem, kompajler će smatrati da je tip number:

```
function increment(counter: number) {  
    return counter++;  
}
```

```
function increment(counter: number) : number {  
    return counter++;  
}
```

**Naredbe za kontrolu toka izvršavanja u *TypeScript*-u.**

**If- else:**

```
if(condition) {  
    // if-statements  
} else {  
    // else statements;  
}
```

Ternarni operator:

```
const max = 100;  
let counter = 100;  
  
counter < max ? counter++ : counter = 1;  
  
console.log(counter);
```

Switch:

```
switch ( expression ) {  
    case value1:  
        // statement 1  
        break;  
    case value2:  
        // statement 2  
        break;  
    case valueN:  
        // statement N  
        break;  
    default:  
        //  
        break;  
}
```

For:

```
for(initialization; condition; expression) {  
    // statement  
}
```

While:

```
while(condition) {  
    // do something  
}
```

Do-while:

```
do {  
    // do something  
} while(condition);
```

## TypeScript funkcije.

Funkcije:

```
function name(parameter: type, parameter:type,...): returnType {  
    // do something  
}
```

Function tipovi:

TypeScript function tipovi omogućavaju da se definišu tipovi za funkcije. function tip ima dva dela: **parametri** i **return** tip.

```
(parameter: type, parameter:type,...) => type
```

### Opcioni parametri:

Sintaksa **parameter?: type** omogućava da parameter bude opcion.

```
function multiply(a: number, b: number, c?: number): number {  
  
    if (typeof c !== 'undefined') {  
        return a * b * c;  
    }  
    return a * b;  
}
```

### Default parametri:

Sintaksa **parameter:=defaultValue** se koristi da se inicijalizuje default vrednost za parametar. Default parametri su opcioni.

```
function name(parameter1=defaultValue1,...) {  
    // do something  
}
```

### Rest parametri:

Rest parametri omogućavaju da se beskonačan broj argumenata predstavi kao niz.

Rest parametri omogućavaju da funkcija primi 0 ili više argumenata određenog tipa.

```
function fn(...rest: type[]) {  
    // ...  
}
```

### Function overloading:

function overloading omogućava da se uspostavi veza između tipova parametara i rezultujućeg tipa funkcije. Primer, ako su parametri funkcije number, funkcija treba da vrati number.

```
function addNumbers(a: number, b: number): number {  
    return a + b;  
}  
  
function addStrings(a: string, b: string): string {  
    return a + b;  
}
```

## TypeScript klase, atributi, metode, vidljivost, konstruktor...

JavaScript nema koncept klase kao drugi programski jezici kao što su Java i C#.

**ES6** omogućava da se definiše klasa što je samo drugačiji sintaktni okvir za kreiranje konstruktor funkcije i prototipsko nasleđivanje :

```
class Person {
    ssn;
    firstName;
    lastName;

    constructor(ssn, firstName, lastName) {
        this.ssn = ssn;
        this.firstName = firstName;
        this.lastName = lastName;
    }

    getFullName() {
        return `${this.firstName} ${this.lastName}`;
    }
}
```

Modifikatori pristupa menjaju vidljivost atributa i metoda klase. TypeScript nudi 3 modifikatora pristupa:

- Private
- Protected
- Public

```
class Person {

    private ssn: string;
    private firstName: string;
    private lastName: string;

    // ...

}
```

TypeScript **getters/setters** se koriste da kontrolišu pristup atributima klase kroz get i set metode koje čitaju (get) i menjaju (set) vrednost atribut.

```

public set fullName(name: string) {
    let parts = name.split(' ');
    if (parts.length != 2) {
        throw new Error('Invalid name');
    }
    this.firstName = parts[0];
    this.lastName = parts[1];
}

```

### Readonly vs const:

	readonly	const
Use for	Class properties	Variables
Initialization	In the declaration or in the constructor of the same class	In the declaration

```

class Person {
    readonly birthDate: Date;

    constructor(birthDate: Date) {
        this.birthDate = birthDate;
    }
}

```

## Nasleđivanje, interfejsi, apstraktne klase, statički atributi i metode.

Ključna reč **extends** se koristi da omogući klasi da nasledi drugu klasu.

Metoda **super()** se koristi u konstruktoru klase naslednika da pozove konstruktor roditeljske klase.

```
class Employee extends Person {
    constructor(
        firstName: string,
        lastName: string,
        private jobTitle: string) {

        super(firstName, lastName);
    }

    describe(): string {
        return super.describe() + `I'm a ${this.jobTitle}.`;
    }
}
```

Za razliku od atributa i metoda instance, **statički atribut i metode** se dele između svih instanca date klase.

Da bi se atribut ili metoda deklarisali kao statički koriti se ključna reč **static**.

```
class Employee {
    private static headcount: number = 0;

    constructor(
        private firstName: string,
        private lastName: string,
        private jobTitle: string) {

        Employee.headcount++;
    }

    public static getHeadcount() {
        return Employee.headcount;
    }
}
```

**Apstraktne Klase** se ne mogu instancirati.

Da bi se klasa deklarisala kao apstraktna treba koristiti ključnu reč **abstract**.

```
abstract class Employee {
    constructor(private firstName: string, private lastName: string) {
    }
    abstract getSalary(): number
    get fullName(): string {
        return `${this.firstName} ${this.lastName}`;
    }
    compensationStatement(): string {
        return `${this.fullName} makes ${this.getSalary()} a month.`;
    }
}
```

TypeScript **interfejsi** definišu ugovore u kodu i pružaju eksplicitne nazive za proveru tipova.

Interfejsi se tipično koriste kao tipovi klasa koji prave ugovor između nepovezanih klasa.

TypeScript omogućava interfejsu da proširi klasu (**extends**).

Interfejs može naslediti **private** i **protected** atribute i metode, a ne samo **public** atribute i metode.

To znači da kada interfejs proširuje klasu **private** ili **protected** atributima i metodama, interfejs može da implementira samo ta klasa ili podklase te klase iz koje se interfejs proširuje.

```
interface Json {  
    toJSON(): string  
}
```

## Objasniti **type casting** u **TypeScript-u**.

**Type casting** omogućava konvertovanje promenljive iz jednog tipa u drugi.

Koristi se ključna reč **as** ili operator **<>** za type casting.

```
let a: typeA;  
  
let b = a as typeB;
```

```
let a: typeA;  
  
let b = <typeB>a;
```

## **TypeScript generici, generičke klase i interfejsi.**

TypeScript **generici** omogućavaju da se piše ponovo iskoristiva (reusable) i generalizovana forma funkcija, klasa, i interfejsa. Eliminišu type casting.

Omogućavaju implementaciju generičkih algoritama.

```
function getRandomElement<T>(items: T[]): T {
    let randomIndex = Math.floor(Math.random() * items.length);
    return items[randomIndex];
}
```

**Generičke klase** imaju listu generičkih tipova parametara u uglastim zagradama **<>** koji slede iza naziva klase.

TypeScript omogućava da se ima više generičkih tipova u listi tipova parametara.

```
class className<T>{
    ...
}
```

Kao i klase, **interfejsi** takođe mogu biti generički.

```
interface Pair<K, V> {
    key: K;
    value: V;
}
```

## TypeScript moduli, *import* i *export*.

Od **ES6**, JavaScript je počeo da podržava module kao izvorni deo jezika.

Modul se izvršava unutar sopstvenog opsega, a ne u globalnom opsegu. To znači da kada deklarišete promenljive, funkcije, klase, interfejse, itd., u modulu, one nisu vidljive izvan modula osim ako ih eksplisitno ne izvezete pomoću naredbe **export**.

S druge strane, ako želite da pristupite promenljivim, funkcijama, klasama itd. iz modula, morate da ih uvezete pomoću naredbe **import**.

Upotreba ključne reči **export**:

```
interface Validator {  
    isValid(s: string): boolean  
}  
  
export { Validator };
```

Da bi se učitao modul, koristi se import naredba.

```
import { Validator } from './Validator';  
  
class EmailValidator implements Validator {  
    isValid(s: string): boolean {  
        const emailRegex = /^[^@]+@[^\s@]+\.\w+$/;  
        return emailRegex.test(s);  
    }  
}  
  
export { EmailValidator };
```

## **NodeJS i TypeScript. Kojom ključnom reči se zadaju komande u NodeJS? Šta su NodeJS moduli?**

**Node JS** je višeplatformsko JavaSkript radno okruženje otvorenog koda za izvršavanje JavaSkript-a na serverskoj strani.

Node JS omogućava da se **JavaSkript** koristi za skripte na serverskoj strani koje omogućavaju da se sadržaj dinamičnih veb stranica generiše na serveru pre nego što se pošalje do veb pregledača korisnika.

U Node.js, ključna reč za zadavanje komandi je `require()`. Ova funkcija se koristi za uključivanje ili uvoz modula u Node.js aplikaciju.

```
const fs = require('fs');
```

Node.js moduli su delovi koda koji obavljaju određene funkcionalnosti i mogu biti nezavisni, ponovno upotrebljivi blokovi koda. **Moduli** pomažu u organizaciji i strukturiranju Node.js aplikacija tako da funkcionalnosti budu grupisane na logičan način.

Moduli se obično definišu u zasebnim fajlovima, gde svaki fajl predstavlja jedan modul. Možete koristiti `require()` funkciju da biste uključili module i pristupili njihovim izvoznim vrednostima, koje su obično funkcije, objekti ili vrednosti koje modul izlaže.

**Node.js moduli** pružaju dobru organizaciju koda, poboljšavaju ponovnu upotrebljivost i olakšavaju održavanje aplikacija.

## **Šta je Angular, a šta Angular CLI?**

Angular je otvorenokodni JavaScript okvir za izgradnju web aplikacija, dok je Angular CLI (Command Line Interface) alat koji olakšava rad sa Angular okvirom.

Angular CLI je alat koji se instalira zasebno i omogućava razvojnim programerima da brzo i efikasno stvaraju, izgrađuju i upravljaju Angular projektima. On pruža niz korisnih funkcionalnosti i komandi koje olakšavaju razvojni proces.

## Struktura Angular komponente.

Komponente u angularu kontrolisu deo ekrana koji se naziva pogled (view). U komponentama se definiše njena aplikativna logika – šta ona radi da podrži pogled (view) – unutar klase.

Komponenta se generiše komandom:

- ng generate component naziv\_komponente

U našem a4geo projektu za početak generišemo tri komponente:

- ng generate component geoportal
- ng generate component sifarnici
- ng generate component nepokretnosti

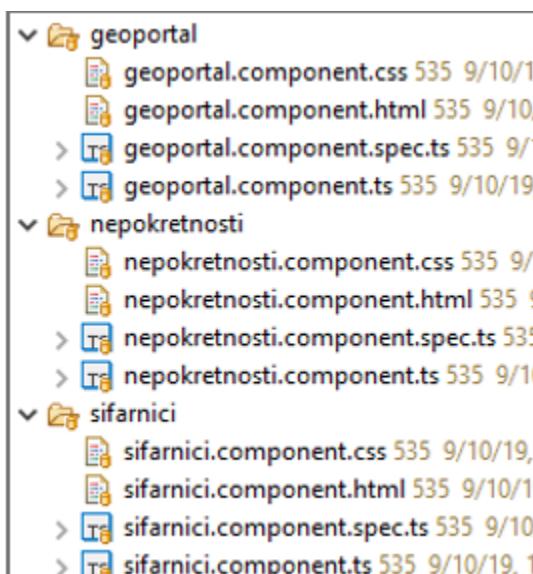
Ovim se generišu 4 fajla:

•HTML templejt

- Fajl u koji se smešta CSS komponente
- Fajl sa ekstenzijom ts u koji se smešta aplikativna logika komponente unutar klase napisane u TypeScriptu
- Fajl sa ekstenzijom spec.ts – testne jedinice

Dekorator identificuje klasu kao odmah ispod, i metapodatke

- selector – CSS Angularu da instancu ove nađe HTML templejtu
- templateUrl – odnosu na ove
- styleUrls - stil



@Component klasu komponentu specificira njene

selektor koji kaže kreira i insertuje komponente gde god odgovarajući tag u

adresa relativna u modul HTML templejta komponente

```
geoportal.component.ts
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-geoportal',
5   templateUrl: './geoportal.component.html',
6   styleUrls: ['./geoportal.component.css']
7 })
8 export class GeoportalComponent implements OnInit {
9
10   constructor() { }
```

## **Navesti komande u *Angular* okruženju za kreiranje projekta, komponente i servisa, pokretanje projekta, *building* projekta...**

**Angular CLI** je alat komandne linije za upravljanje ciklusom razvoja aplikacija u Angularu. Koristite ga za kreiranje početnog fajl sistema **Angular** projekta.

**Angular Material** sadrži komponente za razvoj grafičkog korisničkog interfejsa za Angular. Uobičajeni elementi su meni,dugme,tabela...

**Npm package manager** služi za preuzimanje sa weba i instaliranje npm paketa.

Kada se kreira projekat on inicijalno sadrži komponentu app koja obuhvata sve druge komponente koje će biti kreirane i predstavlja celu aplikaciju.

**Komponenta app:**

- app-routing.module.ts – definiše se navigacija između stranica
- app.component.css – definiše se globalni stil aplikacije
- app.component.html – html templejt
- app.component.ts – TypeScript kod
- app.module.ts – import biblioteka

Angular aplikacije su modularne. Angular ima svoj sistem koji se zove **NgModules**. Mogu da sadrže komponente, provajdere servisa ili bilo koji fajl sa kodom čiji opseg je definisan **NgModulom** u kom je sadžan.

Komponente u angularu kontrolišu deo ekrana koji se naziva pogled. U komponentama se definiše njena aplikativna logika – šta ona radi da podrži pogled unutar klase.

Pogled komponente se definiše sa pridruženim templejtom. **Templejt** je forma HTML-a koji kaže Angularu kako da renderuje komponentu.

**Servisi** u Angularu omogućavaju da se definiše kod kom se može pristupiti iz više različitih komponenti.

Uobičajena upotreba servisa je kada je potrebno komunicirati sa **backendom** da bi se poslali ili primili podaci.

## Za šta se koristi *app-routing-module*?

**AppRoutingModule** je deo Angular rutiranja i koristi se za definisanje ruta (putanja) unutar aplikacije.

Evo nekoliko glavnih funkcija AppRoutingModule-a:

- AppRoutingModule se koristi za definisanje svih ruta (putanja) koje aplikacija podržava.
- Kada korisnik navigira na određenu putanju, AppRoutingModule se koristi za određivanje koje komponente treba prikazati.
- AppRoutingModule omogućava dodavanje dodatne logike za upravljanje rutama. Na primer, možete dodati guardove (čuvare) koji određuju da li korisnik ima dozvolu da pristupi određenoj ruti ili interceptore koji izvršavaju određene akcije pre nego što se korisnik preusmeri na drugu rutu.
- Može uključivati redirekciju korisnika na drugu rutu, prikazivanje ili sakrivanje određenih komponenti i slično.

## **Data binding u Angular-u i vrste data binding-a.**

**Data-binding** u Angularu je automatska sinhronizacija izmedju modela i pogleda u templejtu. To je proces koji kreira vezu između UI aplikacije i podataka. Kada podaci promene svoju vrednost, UI elementi koji su povezani sa podacima se takođe menjaju. Vrste data-bindinga:

- jednosmeran(one-way)
- dvosmeran(two-way)

Kod **dvosmernog** data bindinga, angular okruženje ne prati samo promenu promenljivih u komponenti da bi tu promenu prikazao u grafičkom interfejsu. On takođe prati promene koje je napravio korisnik i u skladu sa tim ažurira promenljive u tih komponenti.

**One-way data binding** je koncept u Angularu koji omogućava prenos podataka sa komponente na šablon (template) u jednom smeru. To znači da kada se podaci u komponenti promene, automatski se reflektuju te promene u šablonu, ali promene koje se naprave u šablonu ne utiču na komponentu.

## Šta je *OpenLayers* i kako se prave mape i layer-i?

**OpenLayers** je biblioteka otvorenog koda koja omogućava dodavanje dinamičkih mapa na web stranicu. Može da prikaže rasterske mape, vektorske podatke i markere učitane iz bilo kog izvora.

Razvijen je sa ciljem da promoviše upotrebu geoprostornih informacija svih vrsta.

Za pravljenje mapa i slojeva koristimo sledeće korake:

- Instalacija OpenLayers biblioteke: Može se preuzeti putem npm-a (Node Package Manager) ili direktno uključiti preko u HTML fajl web aplikacije.
- Kreiranje HTML elementa za prikaz mape: Dodavanje `<div>` element u HTML fajl koji će prikazivati mapu.

```
<div id="map" style="width: 100%; height: 400px;"></div>
```

## Šta su *callback* funkcije?

**Callback** je funkcija koja se prosleđuje kao argument drugoj funkciji. Ova tehnika omogućava funkciji da pozove drugu funkciju. Callback funkcija može da se izvršava nakon što se druga funkcija završila.

Dobar primer je callback funkcija koja se izvršava unutar `.then()` bloka povezana na kraj Promise objekta, nakon što se on uspešno ili neuspešno izvrši.

Callback se često koristi za nastavak izvršavanja koda nakon završetka asinhronne operacije - ovo se zove **asinhroni callback**.

## Šta je *Promise* objekat?

**Promise** objekat predstavlja eventualni završetak asinhronne operacije i njenu rezultujuću vrednost.

Promise je **proxy** za vrednost koja nije nužno poznata kada se Promise kreira.

Omogućava da se povežu hendleri sa eventualnom uspešnom vrednošću ili razlogom neuspeha **asinhronne akcije**.

Ovo omogućava asinhronim metodama da vrate vrednosti isto kao **sinhrone metode**: ali umesto da odmah vrati konačnu vrednost, **asinhroni metod** vraća obećanje da će obezbediti vrednost u nekom trenutku u budućnosti.

**Promise stanja:**

- Pending - inicijalno stanje
- Fulfilled - operacija je uspešno završena
- Rejected - operacija nije uspela

## Šta je *FetchAPI* i gde se koristi?

**Fetch API** obezbeđuje JavaScript interfejs za pristup i manipulaciju delovima HTTP toka, kao što su zahtevi i odgovori.

`fetch()` vraća **Promise objekat**.

**Promise** ima **then()** metodu koja se izvršava nakon što je asinhroni poziv izvršen.

OpenLayers koristi **Fetch API**. Ovaj API omogućava programerima da komuniciraju sa serverom i razmenjuju podatke putem HTTP protokola.

Fetch API se koristi za slanje **AJAX** zahteva sa klijentske strane (browsera) kako bi se dobavili podaci sa servera, ili poslali podaci na server. Ovaj API je zamena za starije metode kao što su XMLHttpRequest i jQuery AJAX.

## **Objasniti *async-await*.**

To je način za upravljanje asinhronim operacijama koristeći jednostavniju i čitljiviju sintaksu.

Ključne reči **async** i **await** se koriste zajedno za definisanje asinhronih funkcija i čekanje na završetak asinhronih operacija unutar tih funkcija.

**Async** funkcija se sastoji od dve glavne ključne reči **async** i **await**.

Klučna reč **async** se koristi da učini funkciju asinhronom.

Klučna reč **await** će tražiti da izvršavanje koda sačeka dok se definisani zadatak ne izvrši.

Pošto `fetch()` vraća **Promise** objekat, može se pojednostaviti kod upotreboom **async/await** sintakse: **response=await fetch()**

## **RxJS Observable, razlika u odnosu na Promise.**

**Observable** je jedinstveni **Object** sličan **Promise-u** koji može da pomogne u upravljanju **asinhronim** kodom. RxJS uvodi **Observables**, novi **Push** sistem za JavaScript.

Funkcija je “**lenjo**” evaluirano izvršavanje koje sinhrono vraća jedinstvenu vrednost pri pozivanju.

**Promise** rukuje **jednim događajem** kada se asinhrona operacija završi ili vrati poruku o grešci.

**Observable** je kao **Stream** i omogućava da se prosledi 0 ili više događaja gde se callback poziva za svaki događaj.

Prednosti Observalbe u odnosu na promise:

- Može da sadrži više događaja
- Može se otkazati zahtev ako je prezahtevan i dovodi do zagušenja
- Za razliku od Promise koji počinje odmah, Observable počinje kada se uradi subscribe
- Observable pružaju operatore kao što su map,reduce,retry,replay...
- Lenjo izvršavanje omogućava da se napravi lanac operatora pre nego što se Observable izvrši kroz subscribe

## Šta je **CRUD?**

**Create/Read/Update/Delete** je forma koja omogućava **osnovnu manipulaciju podataka (kreiranje, čitanje, ažuriranje i brisanje)**.

**CRUD** forma obično sadrži odgovarajuća polja i kontrole koje omogućavaju korisniku da unese ili izmeni podatke, kao i dugmiće ili akcije za izvršavanje operacija.

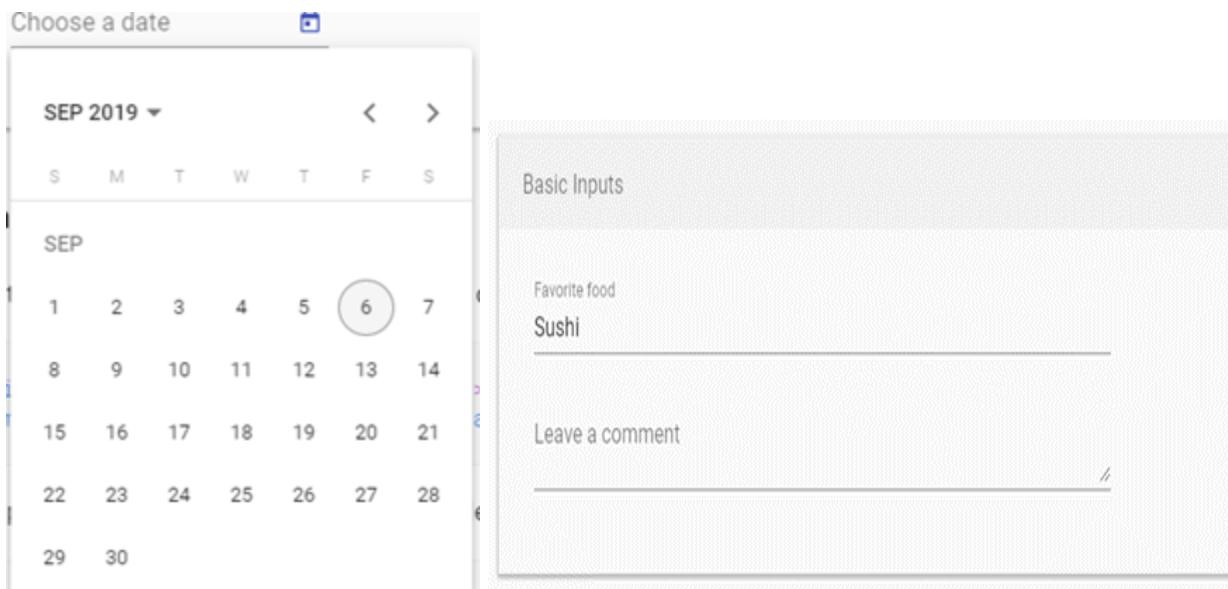
Nosioци права					
JMBG *	Ime *	Ime oca *	Prezime *	Adresa *	
JMBG	ime	ime oca	prezime	adresa	
Grad *	Država *	Tip nosioca prava *			
grad	država	tip nosioca prava			
<input type="button" value="Sačuvaj"/>					
Pretraga <input type="text"/>					
JMBG	Ime	Adresa	Grad	Država	Tip nosioca prava
1101914220011 ZLATKO IGOR POPOVIĆ	GAVRILA PRINCIPA BR. 18	NOVI SAD	SRBIJA		

## Šta je **Angular Material?**

**Angular Material** sadrži komponente za razvoj **grafičkog korisničkog interfejsa** za **Angular**. Sastoјi se od skupa komponenti koji implementiraju uobičajene obrasce za interakciju sa korisnikom prema **Material Design** specifikaciji.

- Uobičajeni elementi grafičkog korisničkog interfejsa kao što su **meni, dugme, tabela, forme za unos podataka...**

Korišćenje Angular Material-a može značajno ubrzati razvoj korisničkog interfejsa i poboljšati izgled i korisničko iskustvo vaše aplikacije. **Biblioteka** je dobro dokumentovana i podržana od strane *Angular* zajednice.



## Šta je **Spring Boot** okruženje i kako se kreiraju **rest** servisi u njemu?

**Spring** okruženje pruža sveobuhvatan model za programiranje i konfiguraciju savremenih poslovnih aplikacija zasnovanih na **Javi** - na bilo kojoj vrsti izvršne platforme. Ključni element **Spring-a** je infrastrukturna podrška na aplikativnom nivou - **Spring** se fokusira na strukturu poslovnih aplikacija tako da se timovi mogu fokusirati na poslovnu logiku na nivou aplikacije, bez nepotrebnom povezivanja sa specifičnim izvršnim okruženjima.

**Spring Boot** ide korak dalje, tako što omogućava lak razvoj **Spring** baziranih aplikacija koje se mogu *samo pokrenuti uz minimalno konfigurisanje*. **Spring** olakšava rad sa **REST** servisima koji šalju poruke u JSON formatu.

Kreiranje **REST** servisa u **Spring Boot**-u okruženju uključuje sledeće korake:

- **Kreiranje projekta** - Prvo je potrebno kreirati **Spring Boot** projekat. To možete učiniti koristeći neki alat kao što je **Spring Initializr** ili pomoću razvojnog okruženja poput **Eclipse**.
- **Definisanje REST kontrolera** - nakon kreiranja projekta, potrebno je definisati **REST kontrolere**. Kontroler je komponenta koja obrađuje **HTTP** zahteve i generiše **HTTP** odgovore.

## Šta je **Maven**?

**Maven** je alat za **upravljanje projektima i automatizaciju izgradnje softvera** koji se često koristi u Java razvoju. On omogućava jednostavno upravljanje zavisnostima, kompilaciju, pakovanje, testiranje i distribuciju projekta.

Nekoliko ključnih karakteristika i funkcionalnosti koje pruža Maven:

- **Struktura projekta** - Maven promoviše standardnu strukturu projekta koja olakšava organizaciju izvornog koda, resursa, konfiguracija i testova.
- **Podešavanje zavisnosti** - Maven olakšava upravljanje zavisnostima projekta.
- **Izgradnja projekta** - Maven olakšava izgradnju projekta.
- **Podešavanje konfiguracija** - Maven omogućava definisanje i upravljanje konfiguracijama projekta kroz *POM* datoteku.

Maven koristi **XML** sintaksu za definisanje konfiguracija projekta.

Maven je široko korišćen u Java razvoju i ima veliku zajednicu korisnika. Pruža standardizovani način upravljanja projektima.