UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

DIGITALNA FORENZIKA

Razvoj alata za analizu i vizualizaciju metapodataka fajlova korišćenjem *Python* biblioteka za obradu i prikaz podataka

Seminarski rad

Studijski program: Računarstvo i informatika

Modul: Softversko inženjerstvo

Mentor:

Bratislav Predić

Student:

Petar Mančić

1. Definisanje problema 3
2. Opis problema 3
3. Način rešavanja problema 4
4. Korišćene tehnologije i alati 4
5.Detaljan opis vizualizacije i analize 8
5.1. Obrada metapodataka 8
5.2. Analiza po tipovima 10
5.3. Analiza fajlova po veličini 13
5.4. Kalendarska reprezentacija aktivnosti pristupa fajlovima 16
5.5. Identifikacija anomalija 20
6. Zaključak 25

1. Definisanje problema

U svakodnevnom radu sa računarom često se susrećemo sa velikim brojem fajlova i foldera koji su raspoređeni u različitim direktorijumima. Kada je potrebno dobiti pregled sadržaja nekog foldera – koliko fajlova postoji, koji su tipovi fajlova, koliku veličinu zauzimaju ili kada su poslednji put menjani – taj proces ručno postaje spor i nepraktičan.

Problem se dodatno komplikuje kada direktorijum sadrži veliki broj podfoldera, jer je tada gotovo nemoguće ručno proći kroz celu strukturu i prikupiti potrebne informacije na pouzdan način.

Zbog toga se javlja potreba za alatom koji može automatski da pročita sadržaj direktorijuma i pruži jasan i pregledan uvid u fajl sistem.

2. Opis problema

Konkretan problem koji se obrađuje u ovom radu odnosi se na **nepostojanje jednostavnog i fleksibilnog alata** koji omogućava automatsko iščitavanje sadržaja direktorijuma i prikupljanje osnovnih informacija o fajlovima.

Bez odgovarajućeg alata, korisnik je primoran da:

- ručno otvara foldere i podfoldere
- proverava svojstva svakog fajla pojedinačno
- vodi evidenciju o pronađenim podacima

Ovakav pristup ne samo da zahteva mnogo vremena, već povećava mogućnost grešaka, kao što su preskakanje fajlova ili pogrešno beleženje informacija. Takođe, ručna analiza postaje gotovo nemoguća kada se radi o direktorijumima koji sadrže stotine ili hiljade fajlova.

Zbog toga se javlja potreba za automatizovanim rešenjem koje može brzo i pouzdano da obradi ceo direktorijum, bez obzira na njegovu veličinu i dubinu strukture.

3. Način rešavanja problema

Kako bi se navedeni problem rešio, odlučeno je da se razvije skripta koja će automatizovati ceo proces analize fajl sistema. Osnovna ideja rešenja jeste da korisnik prosledi putanju do željenog direktorijuma, dok skripta samostalno obavlja sve ostale korake.

Rešenje je zasnovano na sledećem principu rada:

1. Skripta prima putanju do direktorijuma kao ulazni parametar
2. Proverava se da li prosleđena putanja postoji i da li je validna
3. Skripta rekurzivno prolazi kroz sve podfoldere
4. Za svaki pronađeni fajl prikupljaju se osnovni podaci
5. Prikupljeni podaci se čuvaju u strukturisanom obliku
6. Vizuelizacija prikupljenih podataka kro grafike

Na ovaj način korisnik dobija kompletan pregled sadržaja direktorijuma bez potrebe za ručnim pregledanjem fajlova. Automatizacija procesa značajno smanjuje vreme potrebno za analizu i povećava tačnost dobijenih rezultata.

4. Korišćene tehnologije i alati

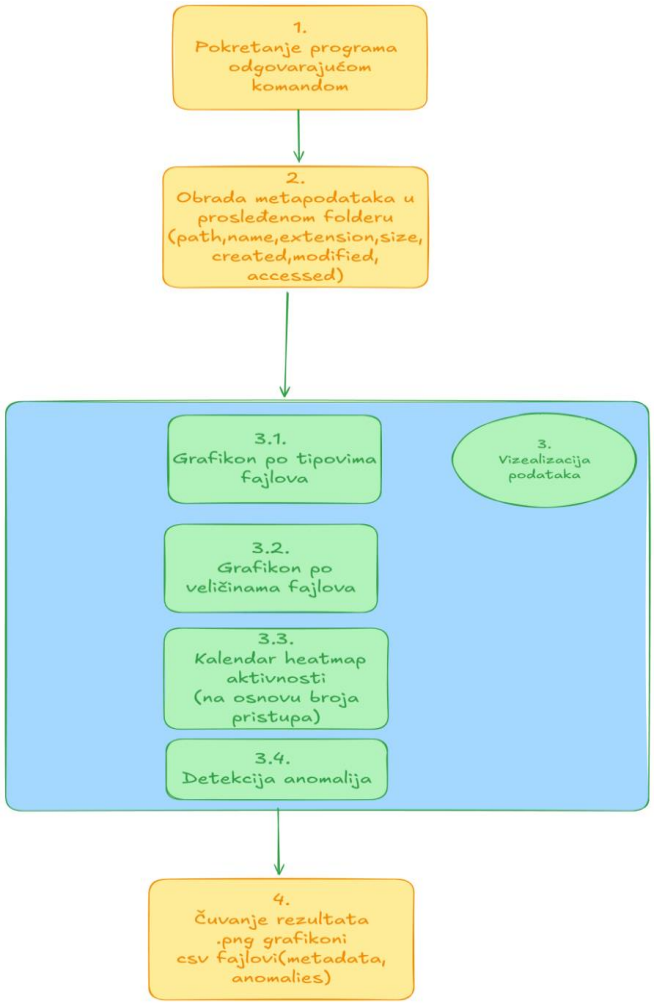
Za implementaciju rešenja izabran je programski jezik *Python*, pre svega zbog svoje jednostavnosti, čitljivosti i bogate standardne biblioteke. *Python* omogućava brz razvoj skripti koje su lake za razumevanje i održavanje, što ga čini pogodnim izborom za ovakav tip zadatka.

U okviru skripte korišćeni su sledeći alati i tehnologije:

- ***Python*** – osnovni programski jezik korišćen za razvoj rešenja
- **os modul** – omogućava rad sa fajl sistemom, uključujući obilazak direktorijuma i pristup metapodacima fajlova
- **argparse modul** – korišćen za prosleđivanje ulaznih parametara kroz komandnu liniju

- **CSV format** – korišćen za čuvanje rezultata analize u tabelarnom obliku
- **Matplotlib**- osnovna biblioteka za crtanje grafikova
- **Calplot**- za pravljenje kalendarskog heatmap-a
- **Pandas**- za učitavanje CSV fajlova, obradu kolona, grupisanje po datumima, izračunavanje broja pristupa.

Kombinacija navedenih alata omogućava da rešenje bude jednostavno za korišćenje, ali i dovoljno fleksibilno za buduća proširenja.



Slika 1 Flowchart programa

Na ovoj slici je prikazazan *flow* dijagrama aplikacije. *Flow* dijagram prikazuje **logički tok rada alata za analizu i vizualizaciju metapodataka fajlova**. Dijagram je podeljen u četiri osnovne faze:

1. Pokretanje programa

- Korisnik pokreće skriptu i prosleđuje putanju foldera koji želi da analizira.
- Program prihvata ovu putanju kao ulazni parametar, čime omogućava fleksibilnost analize različitih direktorijuma.

```
python imeFajla.py --path "putanja do odredjenog foldera" --out-
dir  "putanja do foldera gde zelimo da skladistimo rezultat"--
visualize --detect-iqr
```

Visualize je flag koji nam daje mogućnost da kreiramo grafike.

Detect-iqr je flag koji nam omogućava da kreiramo grafik baziran na anomalijama.

Primer poziva programa:

```
python test.py --path "C:\FAX\MASTER" --out-dir Petar --
visualize --detect-iqr
```

2. Obrada metapodataka

Nakon što smo pokrenuli program zadatak aplikacije je sledeci:

- Prodji kroz sve fajlove u prosledjenom folderu
- Prikupi sledeće metapodatke

Putanja fajla (***path***), ime (***name***), ekstenzija (***extension***), veličina (***size_bytes***), vreme kreiranja (***created***), modifikacije (***modified***) i poslednjeg pristupa (***accessed***).

- Izvrši skladištenje podataka koristeći biblioteku pandas i exportuj u CSV fajl-u za kasniju analizu

3. Vizualizacija i analiza

Ova faza je najzanimljivija i najznačajnija za celokupan rad zato što obuhvata nekoliko tipova analiza i grafičkih prikaza:

1. **Grafikon po tipovima fajlova (*file types*)**
    - Grupisanje fajlova po ekstenzijama i vizualizacija broja fajlova po tipu.
2. **Grafikon po veličinama fajlova (*file sizes*)**
    - Analiza distribucije veličina fajlova pomoću histograma ili boxplot-a.
3. ***Calendar heatmap* aktivnosti (po danima)**
    - Analiza kolone *accessed* za izračunavanje broja pristupa po danu.
    - Vizualizacija aktivnosti u obliku kalendarskog *heatmap-a*, sa opcijom interaktivnog prikaza gde *hover* pokazuje broj pristupa.
4. **Detekcija anomalija**
    - Identifikacija odredjenih dana u kojima se pristupalo fajlovima više u odnosu na neke druge dane

U narednim slajdovima ovaj korak (Vizuelizacija i analiza biće detaljno opisani).

5**. Čuvanje rezultata**

Svi generisani grafikoni se čuvaju u izlaznom folderu (*outputs*), npr.:

- *file_type_distribution.png*
- *file_size_overview.png*
- *calendar_accessed_calplot.png*
- Dodatno, CSV fajlovi sa metapodacima ili anomalijama se mogu sačuvati radi dalje obrade ili dokumentacije.

5.Detaljan opis vizualizacije i analize

U ovom delu bavićemo se detaljnije načinom na koji se generišu grafici.

5.1. Obrada metapodataka

Svakako krucijalni korak bez kojeg dalja izrada ne bi imala smisla jeste svakako obrada metapodata o fajlovima. Na *flowchart-u* je ovaj korak označen brojem 2.

```
def scan_directory(root_path, include_hidden=False):
    metadata = []
    for dirpath, dirnames, filenames in os.walk(root_path):
        if not include_hidden:
            dirnames[:] = [d for d in dirnames if not d.startswith('.')]
            filenames = [f for f in filenames if not f.startswith('.')]
        for fname in filenames:
            fpath = Path(dirpath)/fname
            try:
                st = fpath.stat()
                metadata.append({
                    'path': str(fpath),
                    'name': fname,
                    'extension': fpath.suffix.lower().lstrip('.') or 'no_ext',
                    'size_bytes': st.st_size,
                    'created': datetime.fromtimestamp(st.st_ctime),
                    'modified': datetime.fromtimestamp(st.st_mtime),
                    'accessed': datetime.fromtimestamp(st.st_atime)
                })
            except Exception as e:
                print(f'Warning: failed to read {fpath}: {e}')
    return pd.DataFrame(metadata)
```

Slika 2 Funckija koja je zadužena da prikupi metapotadke o fajlovima

Funkcija *scan_directory* je ključni deo alata za prikupljanje i analizu metapodataka fajlova u zadatom direktorijumu i njegovim podfolderima. Njena osnovna svrha je da rekurzivno prolazi kroz sve fajlove u prosleđenom folderu, prikuplja informacije o svakom fajlu i priprema podatke za dalje analize i vizualizaciju.

Ulazni parametri:

1. *root_path* (*string* ili *Path* objekat) – putanja do foldera koji će se analizirati.
2. *include_hidden* (*boolean*, podrazumevano *False*) – određuje da li će funkcija obraditi skrivene fajlove i foldere (koji počinju sa .).

Glavni koraci funkcije:

1. **Rekurzivno skeniranje foldera**
    - Funkcija koristi ***os.walk*()** da bi prošla kroz sve podfoldere i fajlove unutar *root_path*.
    - Ako je *include_hidden=False*, izuzimaju se svi fajlovi i direktorijumi koji počinju sa tačkom (skriveni fajlovi).
2. **Prikupljanje metapodataka za svaki fajl**
    - Za svaki fajl se prikupljaju sledeći atributi:
        - *path* – puna putanja fajla
        - *name* – ime fajla
        - *extension* – ekstenzija fajla (bez tačke; ako fajl nema ekstenziju, označava se kao *no_ext*)
        - *size_bytes* – veličina fajla u bajtovima
        - *created* – vreme kreiranja fajla
        - *modified* – vreme poslednje modifikacije
        - *accessed* – vreme poslednjeg pristupa fajlu
    - Za čitanje ovih informacija koristi se ***Path.stat*()** objekat i ***datetime.fromtimestamp*()** za konverziju vremenskih atributa u čitljiv format.

Ovako izgledaju podaci koji nakon obrade budu upisani u .csv fajl:

path	name	extension	size_bytes	created	modified	accessed
C:\POSAO VEGA\What do you know about API.docx	What do you know abo	docx	11912	2025-5-25 17:19	2025-5-25 17:19	2025-6-12 21:23
C:\POSAO VEGA\DependencyInjection\DependencyInjection.sln	DependencyInjection.s	sln	1171	2025-9-9 21:45	2025-9-9 21:45	2025-9-29 21:37
C:\POSAO VEGA\DependencyInjection\DependencyInjection\appsettings.Development.json	appsettings.Developm	json	127	2025-9-9 21:45	2025-9-9 21:45	2025-9-9 22:16
C:\POSAO VEGA\DependencyInjection\DependencyInjection\appsettings.json	appsettings.json	json	151	2025-9-9 21:45	2025-9-9 21:45	2025-9-9 22:16
C:\POSAO VEGA\DependencyInjection\DependencyInjection\DependencyInjection.csproj	DependencyInjection.c	csproj	333	2025-9-9 21:45	2025-9-9 21:45	2025-9-9 21:46
C:\POSAO VEGA\DependencyInjection\DependencyInjection\DependencyInjection.csproj.user	DependencyInjection.c	user	238	2025-9-9 21:45	2025-9-9 21:45	2025-9-9 21:46
C:\POSAO VEGA\DependencyInjection\DependencyInjection\DependencyInjection.http	DependencyInjection.h	http	151	2025-9-9 21:45	2025-9-9 21:45	2025-9-9 21:45
C:\POSAO VEGA\DependencyInjection\DependencyInjection\IRandomNumberService.cs	IRandomNumberServic	cs	132	2025-9-9 21:47	2025-9-9 21:55	2025-9-9 22:16
C:\POSAO VEGA\DependencyInjection\DependencyInjection\Program.cs	Program.cs	cs	575	2025-9-9 21:45	2025-9-9 22:16	2025-9-9 22:16

Slika 3 Izgleda podataka nakon obrade I upisivanja u .csv fajl

3**. Obrada grešaka**

- Funkcija hvata eventualne greške prilikom čitanja fajlova (npr. nedozvoljen pristup) i ispisuje upozorenje, ali ne prekida izvršavanje programa.

4**.Vraćanje rezultata**

- Prikupljeni metapodaci se skladište u *pandas.DataFrame*, što omogućava jednostavnu dalju obradu, analizu i vizualizaciju podataka.

5.2. Analiza po tipovima

Svrha ovog grafika jeste da nam omogući uvid u strukturu foldera do koji smo prosledili putanju i raspodelu prema tipovima.

Takođe daje nam uvid u identifikaciju dominirajućih tipova fajlova I potencijalno neubičajenih formata.

Funkcija za vizualizaciju distribucije tipova fajlova.

Ova funkcija služi za **analizu i grafički prikaz distribucije fajlova po tipu (ekstenziji)** na osnovu prikupljenih metapodataka.

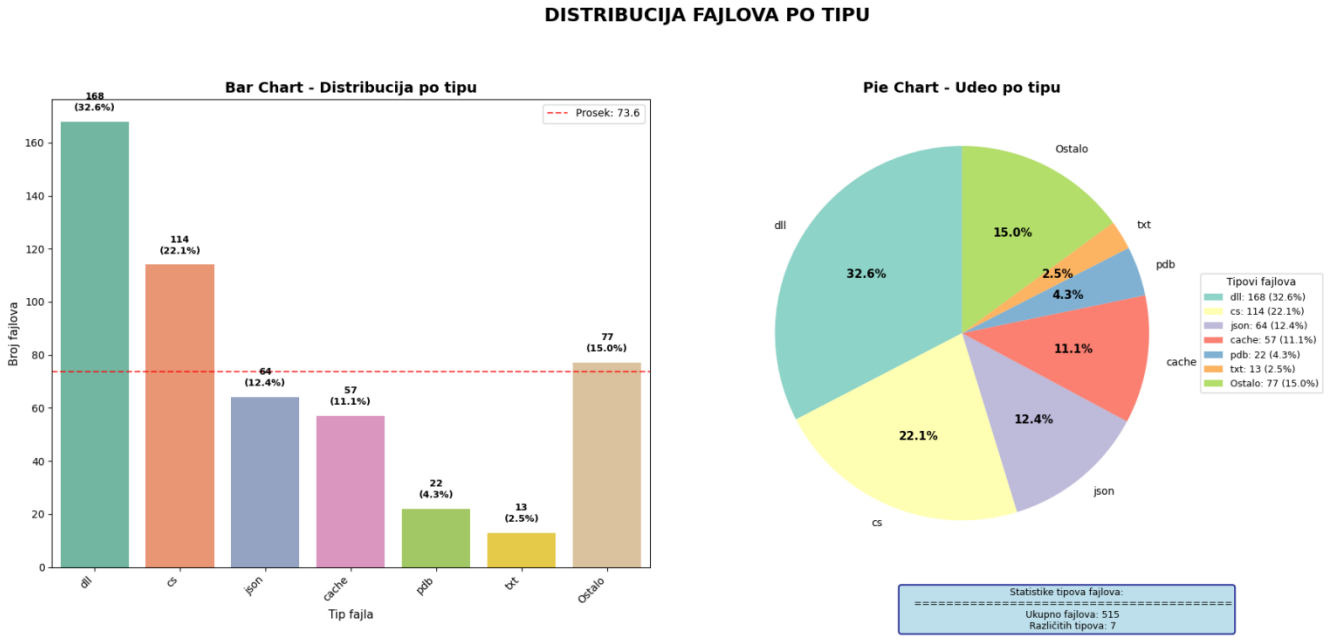
U prvom koraku se **broji koliko fajlova postoji za svaki tip ekstenzije**, a zatim se retko zastupljeni tipovi (manje od 2% ukupnog broja fajlova) **grupišu u kategoriju „Ostalo“**, kako bi vizualizacija bila preglednija.

Funkcija zatim kreira **dva komplementarna grafička prikaza**:

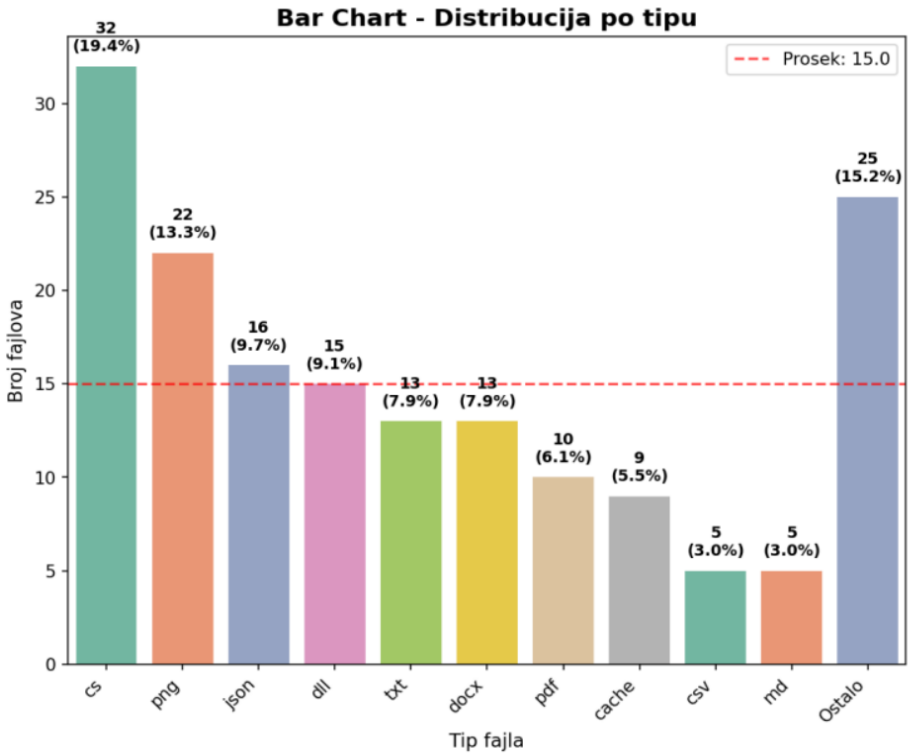
- **Stubičasti (*bar*) dijagram**, koji prikazuje apsolutni broj fajlova po tipu, uz dodatne oznake koje sadrže broj i procenat fajlova, kao i liniju koja označava prosečnu vrednost.
- **Pita (*pie*) dijagram**, koji prikazuje relativni udeo svakog tipa fajla u ukupnom skupu podataka.

Pored grafika, ispod kružnog dijagrama se prikazuju i **osnovne statističke informacije**, kao što su ukupan broj fajlova i broj različitih tipova fajlova.

Na kraju, generisani grafikoni se po potrebi **automatski čuvaju u izlazni direktorijum**, čime se omogućava njihova dalja upotreba u izveštajima i dokumentaciji.



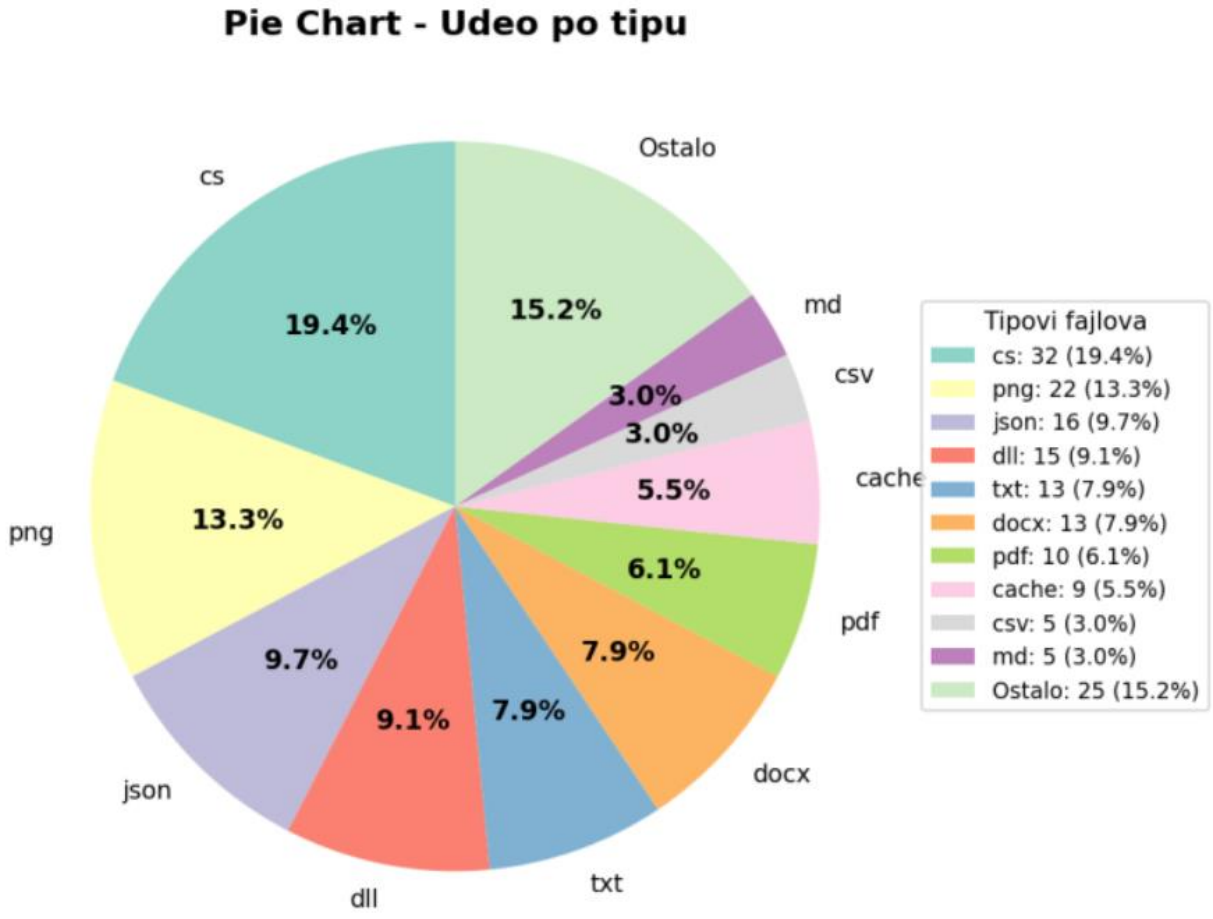
Slika 4 Distribuacija fajlova po tipu



Slika 5 Bar chart koji nam prikazuje distribuciju fajlova po tipu

Bar dijagram prikazuje raspodelu fajlova prema njihovom tipu u analiziranom direktorijumu. Na horizontalnoj osi nalaze se tipovi fajlova (ekstenzije), dok vertikalna osa predstavlja broj fajlova za svaki tip. Visina svakog stuba ukazuje na učestalost određenog tipa fajla, što omogućava brzo poređenje između različitih kategorija.

Na dijagramu su dodatno prikazane i procentualne vrednosti u odnosu na ukupan broj fajlova, čime se dobija jasniji uvid u relativni značaj pojedinih tipova. Horizontalna isprekidana linija označava prosečan broj fajlova po tipu, što olakšava identifikaciju tipova fajlova koji značajno odstupaju od proseka. Ovakav prikaz omogućava lako uočavanje dominantnih i ređe zastupljenih tipova fajlova u posmatranom skupu podataka.



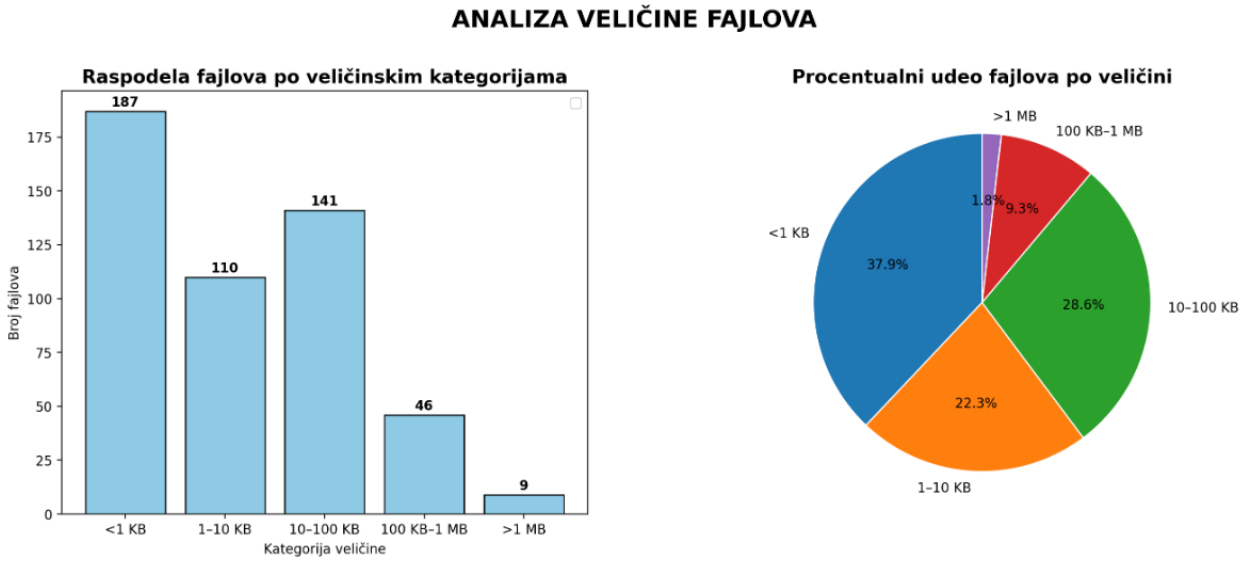
Slika 6 Distribuija fajlova po tipu korišćenjem pie chart-a

Kružni dijagram prikazuje procentualni udeo različitih tipova fajlova u ukupnom broju analiziranih fajlova. Svaki segment dijagrama predstavlja jedan tip fajla, dok veličina segmenta odgovara njegovom relativnom učešću u skupu podataka. Ovakav prikaz omogućava intuitivan pregled strukture fajlova i njihovog međusobnog odnosa.

Pie chart je posebno koristan za identifikaciju dominantnih tipova fajlova, kao i za sagledavanje ukupnog doprinosa manje zastupljenih kategorija koje su grupisane pod nazivom „Ostalo“. Time se dobija sažet i pregledan uvid u raspodelu fajlova po tipu, što olakšava interpretaciju rezultata u okviru analize fajl sistema.

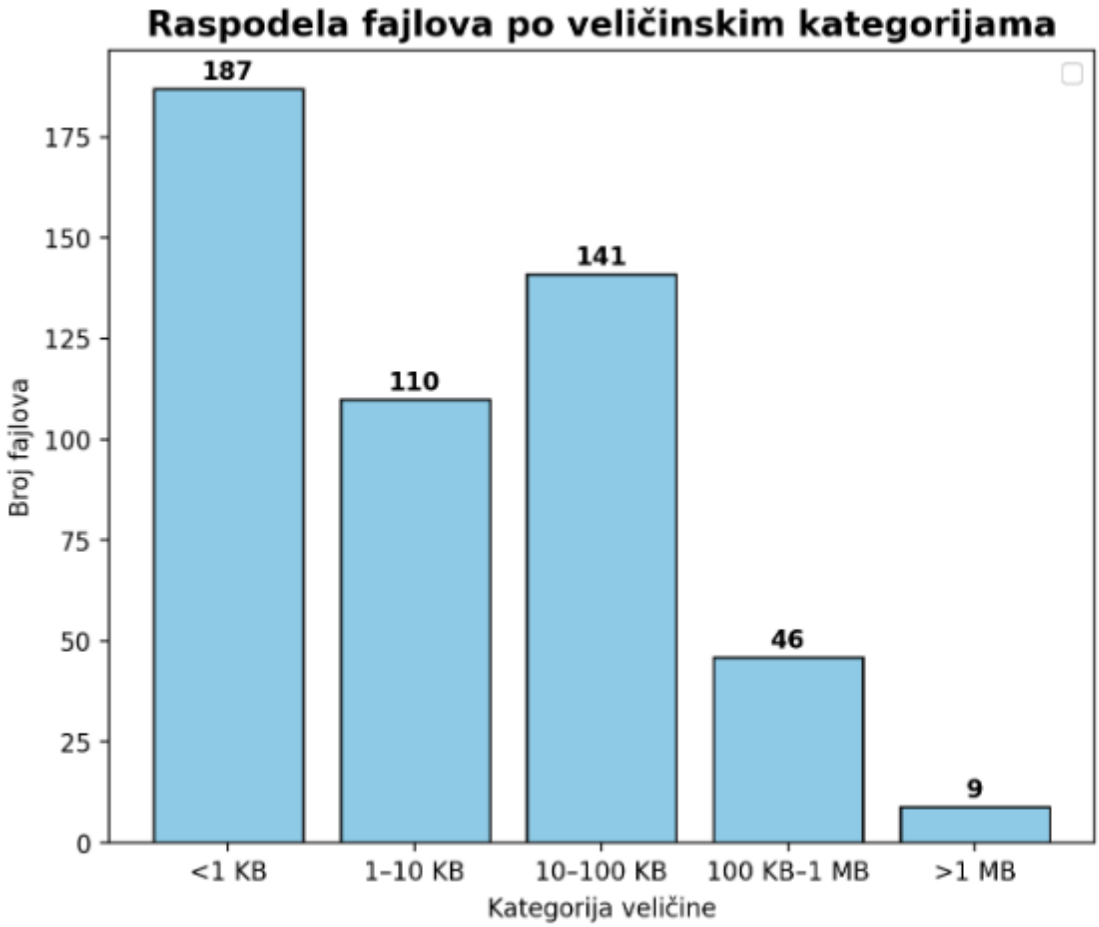
5.3. Analiza fajlova po veličini

Kao i u slučaju analize fajlova po različitim tipovima i kod analize fajlova po veličini kreirao sam 2 tipa grafika, bar chart I pie chart sa dodatnom tabelom koja služi da imamo neki generalni uvid u statistiku poput ukupan broj fajlova, najveci fajl, najmanji fajl itd.



Metod	Vrednost
Ukupno fajlova	515
Ukupna veličina (MB)	47.93
Prosečna veličina (KB)	95.29
Minimalna veličina (B)	0
Maksimalna veličina (MB)	6.20

Slika 7 Grafikoni koji predstavljaju analizu veličine fajlova



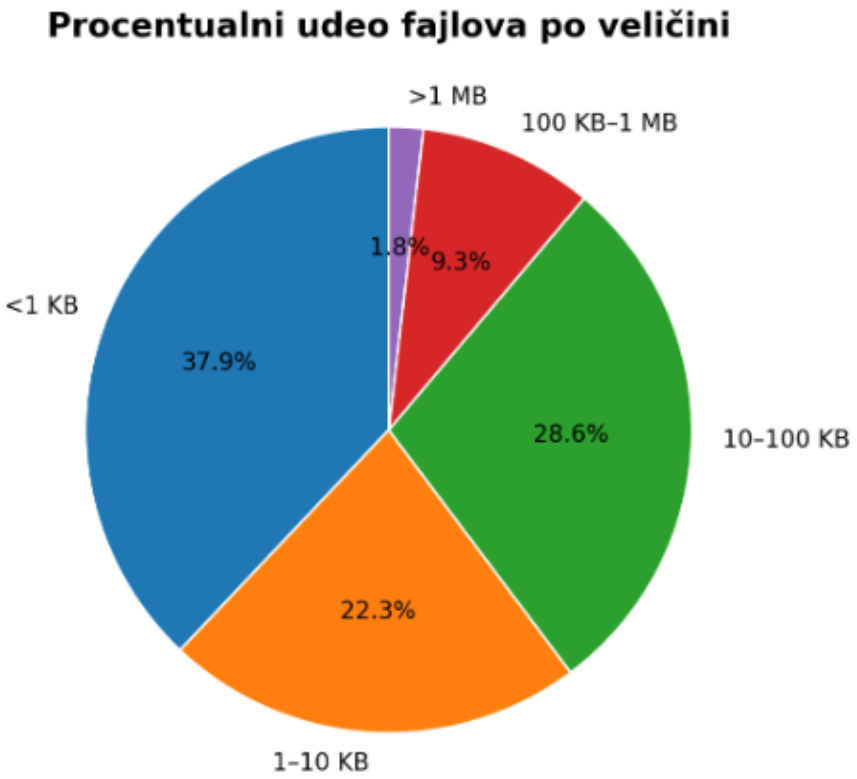
Slika 8 Raspodela fajlova po veličini fajlova-Bar chart

Grafikon prikazuje raspodelu fajlova po veličinskim kategorijama u analiziranom folderu.

Bar chart pokazuje koliko fajlova spada u svaku kategoriju veličine, od najmanjih fajlova (<1 KB) do najvećih (>1 MB).

Zajedno, ove vizualizacije omogućavaju da se brzo razume raspodela fajlova u folderu, identifikuju male i velike datoteke i uoče potencijalne anomalije u veličini fajlova.

Pie chart predstavlja **procenat fajlova u svakoj veličinskoj kategoriji**, što omogućava vizualno uočavanje dominantnih veličina fajlova. Svaki segment pita odgovara određenoj kategoriji, a njegova površina proporcionalna je broju fajlova u toj kategoriji. Ovakva vizualizacija je posebno korisna kada želimo brzo da sagledamo **koje veličine fajlova čine najveći deo foldera**, bez potrebe da brojke posebno analiziramo. *Pie chart* u kombinaciji sa bar *chart-om* daje sveobuhvatan uvid u strukturu fajlova po veličini, čineći podatke lakše razumljivim i preglednijim.



Metod	Vrednost
Ukupno fajlova	515
Ukupna veličina (MB)	47.93
Prosečna veličina (KB)	95.29
Minimalna veličina (B)	0
Maksimalna veličina (MB)	6.20

Slika 9 Pie chart i tabela statistika

Takođe prisutna je i tabela statistika koja nam pokazuje neke zanimljive podatke poput koliko ukupno fajlova se nalazi u analiziranom folderu, zatim kolika je ukupna veličina svih fajlova, min i max veličine a takodje i prosečna veličina fajlova.

5.4. Kalendarska reprezentacija aktivnosti pristupa fajlovima

Kalendarska reprezentacija aktivnosti pristupa fajlovima ima za zadatak da nam kalendarski prikaže koliko smo u toku godine pristupali fajlovima, na taj način možemo pratiti svoj progres u radu.

Za generisanje ovog grafika korišćena je biblioteka ***catplot.***

Catplot je funkcija iz *Python* biblioteke ***Seaborn***, namenjena za **vizualizaciju kategorijalnih podataka**. Ona predstavlja objedinjeni interfejs za više tipova grafika (kao što su *bar*, *box*, *violin*, *strip*, *swarm*), omogućavajući jednostavno poređenje numeričkih vrednosti između različitih kategorija.

Catplot automatski podržava razdvajanje podataka po redovima i kolonama pomoću parametara *row* i *col*, što olakšava analizu složenijih skupova podataka.

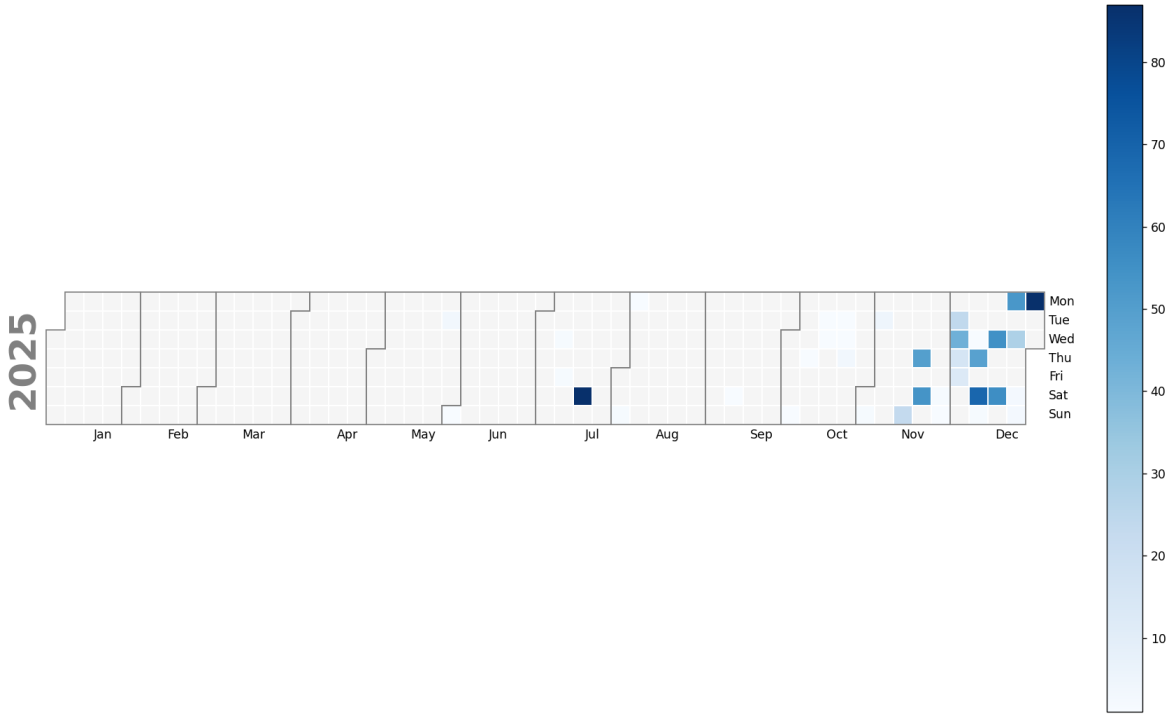
Ukratko, ***catplot*** se koristi kada je potrebno jasno i pregledno prikazati odnose i raspodele podataka unutar kategorija.

U ovom radu, funkcija ***catplot*** iz biblioteke *Seaborn* korišćena je za vizuelizaciju podataka koji su prethodno agregirani po danima. Kao ulaz za *catplot* nisu korišćeni sirovi podaci, već rezultati funkcije ***groupBydays***, čija je svrha da izvrši obradu i grupisanje podataka po datumu pristupa.

Funkcija *groupBydays* učitava metapodatke o fajlovima i vrši normalizaciju vremenskih oznaka na dnevni nivo, nakon čega grupiše podatke po danima. Za svaki dan izračunavaju se relevantne metrike, kao što su ukupan broj pristupa fajlovima, broj jedinstvenih fajlova i direktorijuma, kao i vremenski raspon aktivnosti.

Rezultujući skup podataka predstavlja strukturirani calendar u kojoj svaki red odgovara jednom danu i sadrži agregirane vrednosti. Ovakav format je direktno pogodan za upotrebu u funkciji ***catplot***, gde su datumi korišćeni kao kategorijalna osa, a broj pristupa fajlovima kao numerička vrednost. Na taj način, catplot omogućava jasan i pregledan prikaz dnevnih obrazaca aktivnosti, kao i poređenje intenziteta pristupa između različitih dana.

Primer takvog grafika je prikazan na sledećoj slici.

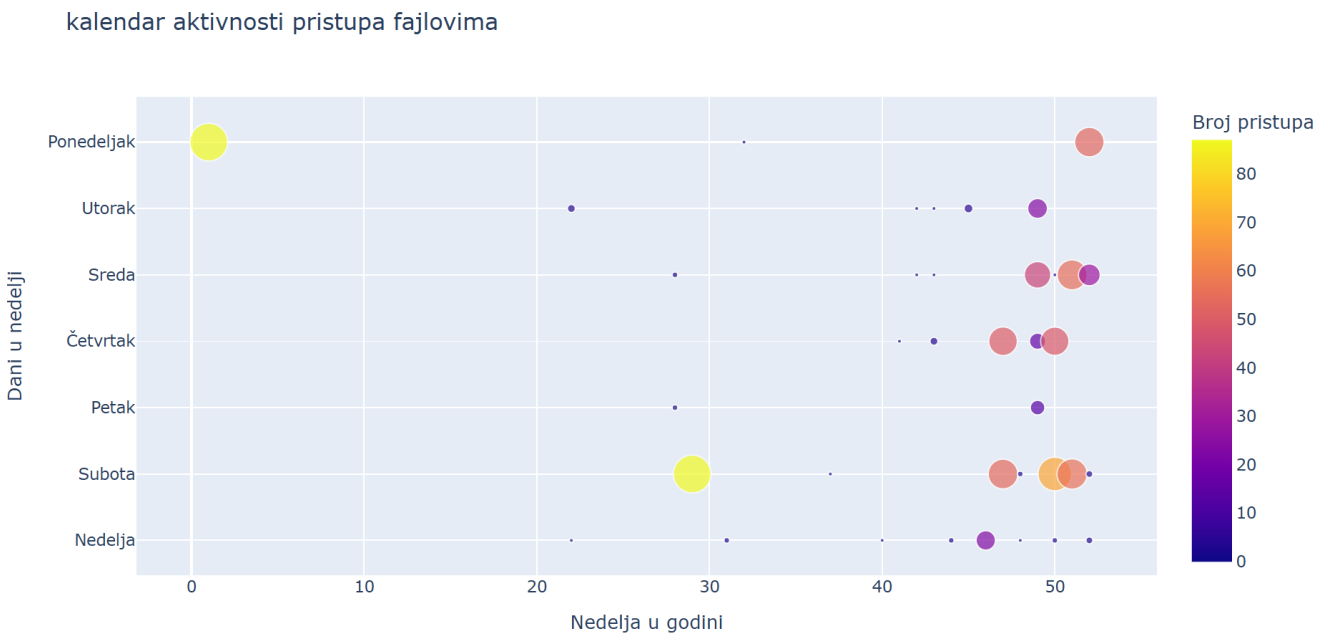


Slika 10 Kalendarski prikaz broja pristupa fajlovima po danu

Na prikazanoj slici može se videti učestalost pristupa fajlovima po danima. Sa desne strane nalazi se skala boja (*color bar*) koja predstavlja broj pristupa fajlovima, pri čemu se u ovom konkretnom primeru vrednosti kreću u opsegu od 0 do 80.

Prikazani grafikon je statičkog tipa, što znači da ne omogućava prikaz dodatnih informacija prilikom prelaska mišem preko pojedinačnih datuma. Iz tog razloga generisan je i dodatni, interaktivni grafikon koji se otvara u *web* pregledaču. Ovaj interaktivni prikaz omogućava korisniku da, prilikom hoverovanja preko određenog datuma, dobije detaljnije informacije, kao što su tačan datum i broj pristupa fajlovima.

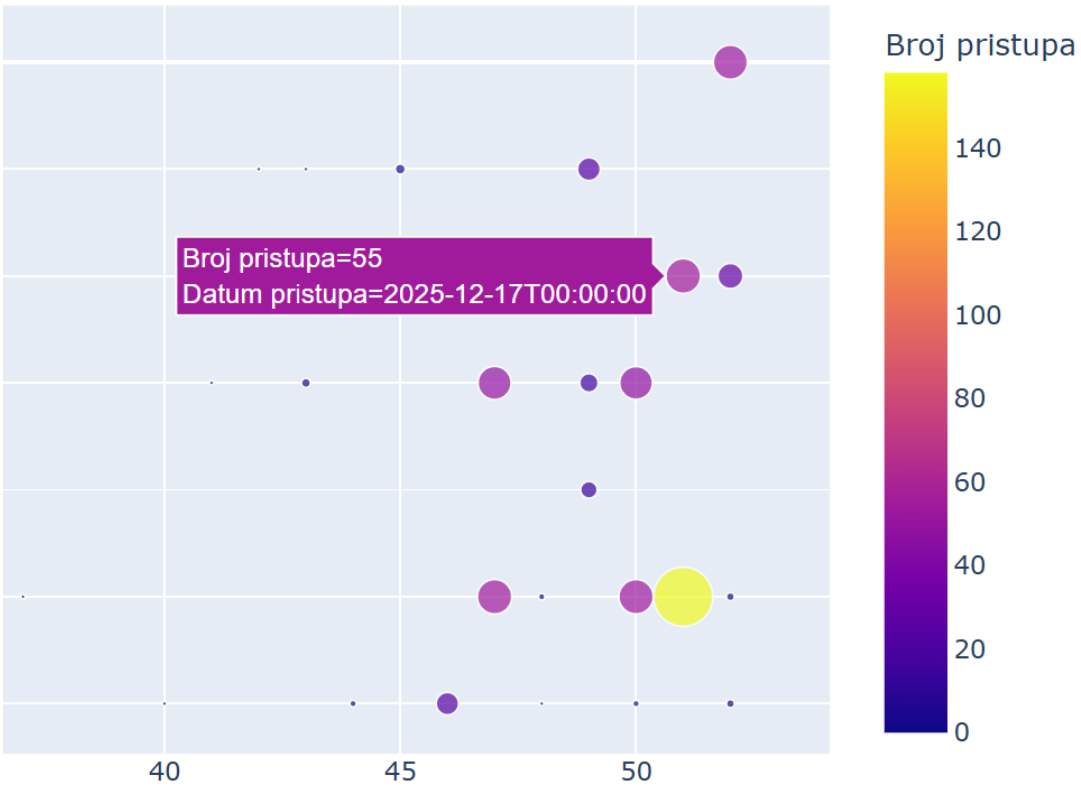
Kako bismo dobili detaljniji uvid u to koliko i kada je pristupano fajlovima, kreiran je dinamički grafikon.



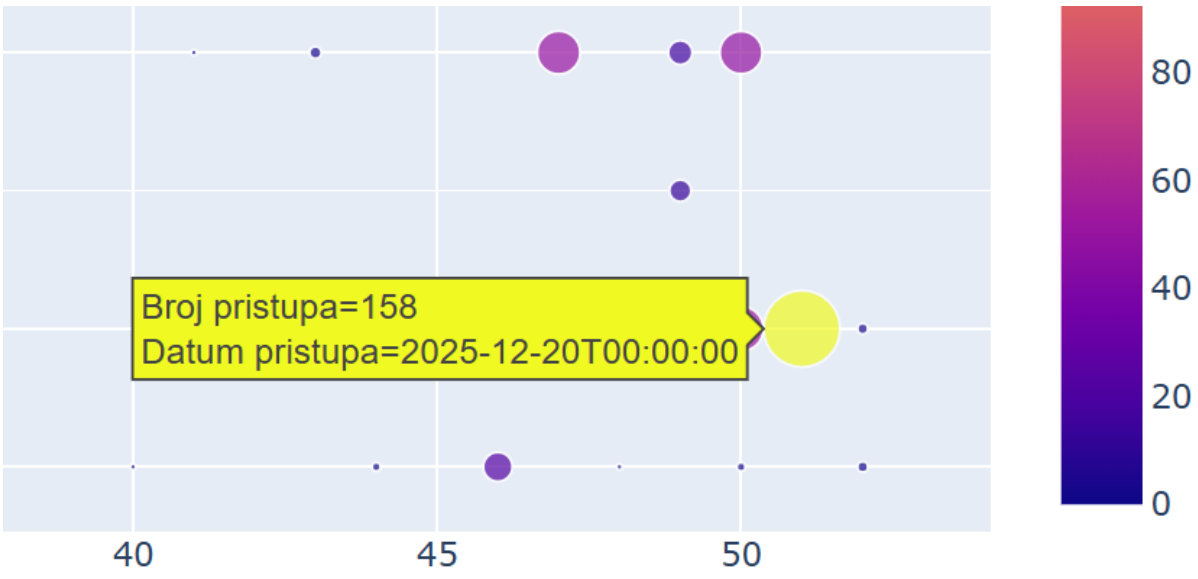
Slika 11 Kalendar aktivnosti pristupa fajlovima

Na slici su prikazani kružići koji označavaju datume kada je bilo pristupa fajlovima.

Prednost ovog grafikona u odnosu na prethodni je njegova „dinamičnost“ – kada pređemo mišem preko kružića, biće prikazan tačan broj pristupa fajlovima tog dana, što omogućava precizniju i interaktivnu analizu podataka.



Slika 12 Detaljne informacije o pristupima fajlovima odredjenog dana



Slika 13 Dan kada se najviše pristupalo fajlovima

5.5. Identifikacija anomalija

Kako bi se identifikovali neuobičajeni obrasci u pristupu fajlovima, primenjena je statistička metoda za detekciju anomalija zasnovana na kvartilima. Cilj ove analize je da se izdvoje kod kojih je broj pristupa značajno veći (ili manji) u odnosu na uobičajene vrednosti, što može ukazivati na potencijalno sumnjivo ponašanje, greške u sistemu ili specifične događaje.

Za detekciju anomalija korišćena je ***IQR (Interquartile Range)*** **metoda**, koja je robusna i ne zavisi od ekstremnih vrednosti u podacima.

Objašnjenje Q1, Q3 i IQR

- **Q1 (prvi kvartil)** predstavlja vrednost ispod koje se nalazi 25% svih podataka
- **Q3 (treći kvartil)** predstavlja vrednost ispod koje se nalazi 75% svih podataka
- **IQR (Interquartile Range)** je raspon između trećeg i prvog kvartila i računa se kao:

$$IQR=Q3-Q1$$

IQR opisuje „normalan“ raspon srednjih 50% podataka i koristi se za identifikaciju odstupanja.

Pravila za detekciju anomalija

Na osnovu izračunatog IQR-a definišu se granice prihvatljivih vrednosti:

- **Donja granica**:

  Q1−1.5*IQR

- **Gornja granica**:

  Q3+1.5*IQR

Sve vrednosti koje se nalaze **ispod donje** ili **iznad gornje granice** smatraju se anomalijama (*outlier-ima*).

Primena u analizi pristupa fajlovima

U kontekstu ove analize, Q1 i Q3 su izračunati na osnovu broja pristupa fajlovima po danu. Dani kod kojih je broj pristupa značajno veći od gornje granice identifikovani su kao potencijalne anomalije. Ovakvi skokovi mogu ukazivati na:

- neuobičajeno intenzivnu aktivnost korisnika,
- automatizovane procese ili skripte,
- greške u logovanju podataka,
- ili potencijalne bezbednosne incidente.

Na ovaj način dobijen je objektivan i statistički utemeljen mehanizam za identifikaciju sumnjivih obrazaca u podacima.

Da bismo što bolje razumeli na koji način se identifikuju anomalije objasnićemo na sledećem primeru.

Zamislimo da je broj pristupa po danu opisano sledećim nizom brojeva:

[12, 14, 15, 15, 16, 17, 18, 18, 19, 20, 21, 22, 23, 25, 60]

1. Prvi korak jeste sortiranje niza.
2. Određivanje **Q2 (medijane)**

   Q2 je **medijana** – vrednost koja deli skup na dve polovine.

- Ako je broj elemenata **neparan** → Q2 je srednji element
- Ako je broj elemenata **paran** → Q2 je prosek dva srednja elementa

U našem primeru ima **15 vrednosti (neparan broj)**.

3. Određivanje Q1, prvi kvartil

   Q1 je medijana **donje polovine podataka** (vrednosti manje od Q2).

   Donja polovina:

   [12, 14, 15, 15, 16, 17, 18]

   Kako ovaj niz ima 7 vrednosti, srednja vrednost je 15.

4. Određivanje Q3, treći kvartil

   Q3 je medijan **gornje polovine podataka** (vrednosti veće od Q2).

   Gornja polovina:

   [18, 19, 20, 21, 22, 23, 25, 60]

   Ovaj skup ima 8 elemenata → medijan je prosek srednje dve vrednosti:

   Q3 = (21 + 22) / 2 = 21.5

5. Izračuvanje IQR-a
   Prvo je potrebno izračunati interkvartilni raspon poznat kao IQR.
   IQR=Q3-Q1

   Na osnovu IQR-a potrebno je definisati gornju I donju granicu normalnih vrednosti

   Donja granica se izračunava po formuli:
   Q1-1.5*IQR

   Gornja granice se izračunava po formuli:
   Q3+1.5*IQR

Ove granice određuju do koje mere se podaci mogu „normalno“ raspršiti oko centralnog dela raspodele.

Pravilo za identifikaciju anomalija

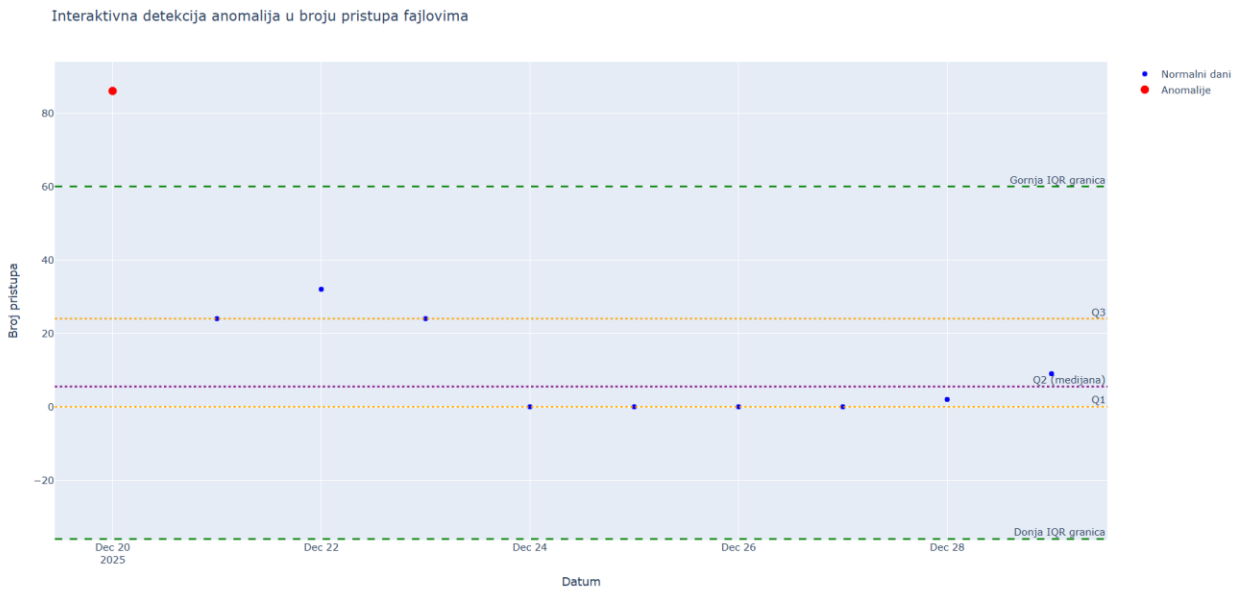
Svaka vrednost koja:

- je **manja od donje granice**, ili
- je **veća od gornje granice**

smatra se **anomalijom (*outlier-om*)**.

Formalno: $x < Q1 - 1.5 \times IQR \quad ili \quad x > Q3 + 1.5 \times IQR$

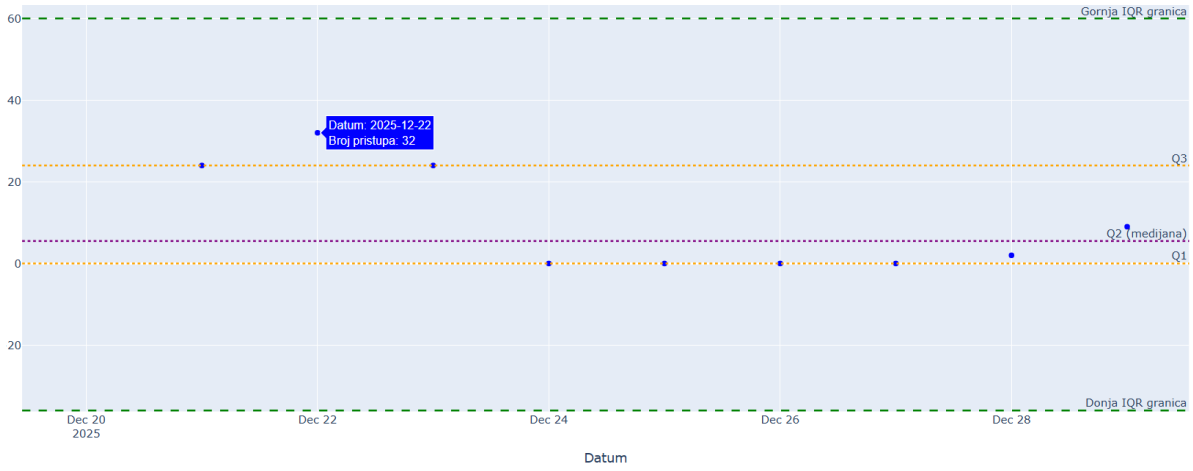
Sada ćemo videti kako to zapravo izgleda na praktičnom primeru.



Slika 14 Detekcija anomalija u broju pristupa fajlovima

Ova slika nam prikazuje sve ono gore navedeno. Jasno se vidi Q1, Q2 i Q3, takodje je prikazana gornja i donja granica.

U prevodu sve sto je iznad gornje IQR granice i sve sto je ispod donje IQR granice se identifikuje kao anomalija.

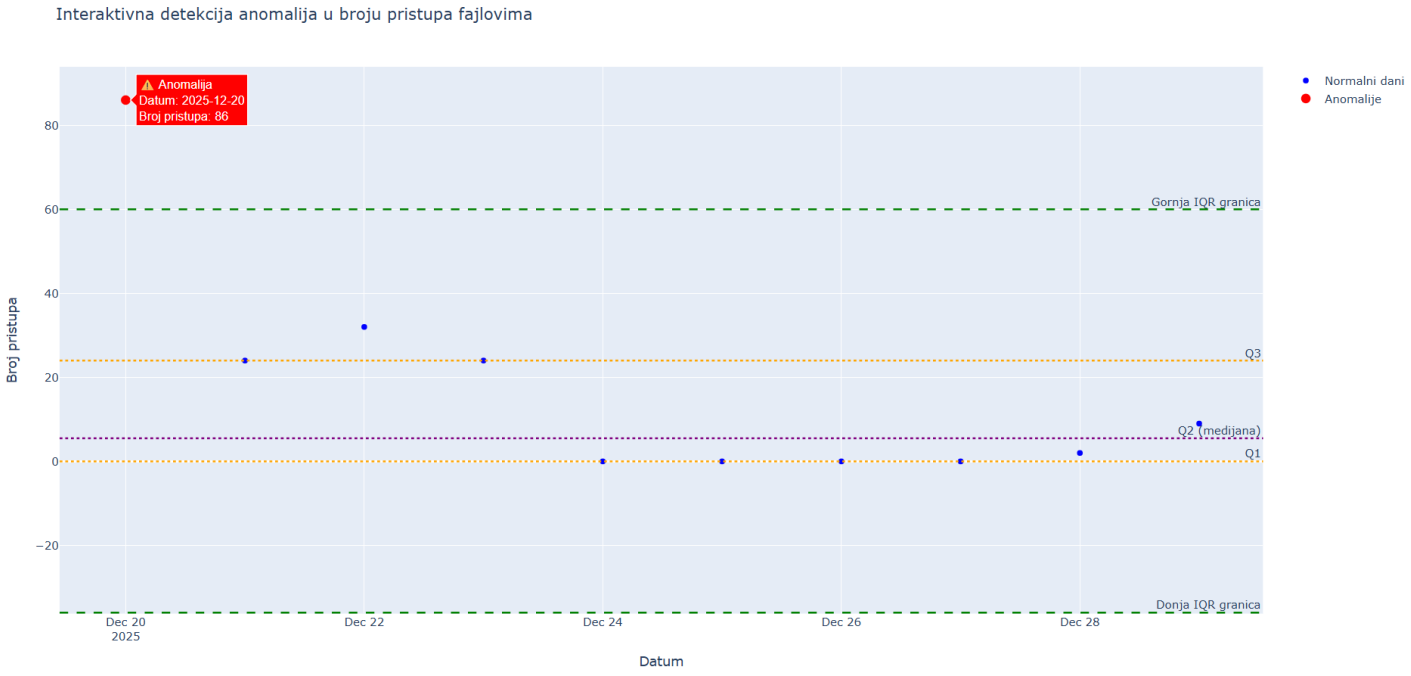


Slika 15 Primer jednog normalnog dana koji se ne identifikuje kao anomalija

Plavim tačkicama su označeni dani u kojima se nije desio “prestup” u broju kreiranja I pristupa fajlovima.

Sa druge strane postoji I crveni tačkice koje nam prikazuju dane koji su označeni kao anomalije.

Takođe dodata je mogućnost da se prilikom hoverovanja mišem preko plavih i crvenih tačkica dobiju informacije kojeg dana se to desilo i koliko pristupa.



Slika 16 Identifikovana anomalija

6. Zaključak

U ovom radu prikazana je implementacija alata za analizu i vizualizaciju metapodataka fajlova korišćenjem programskog jezika *Python* i relevantnih biblioteka za obradu podataka i grafičku prezentaciju, kao što su ***pandas***, ***matplotlib*** *i* ***plotly***. Alat omogućava prikupljanje, obradu i analizu različitih atributa fajlova, uključujući veličinu, tip, vreme kreiranja, modifikacije i poslednjeg pristupa.

Kroz praktične primere, demonstrirano je kako se prikupljeni podaci mogu prikazati na grafički pregledan način, uključujući raspodelu fajlova po tipu i veličini, kalendarsku vizualizaciju pristupa fajlovima i identifikaciju anomalija u podacima. Upotrebom vizualizacija, korisnici mogu brzo i efikasno da uoče obrasce i neuobičajene aktivnosti u fajl sistemu, što može biti od značaja u kontekstu digitalne forenzike, optimizacije skladištenja podataka i nadzora sistema.

Ovaj rad pokazuje da kombinacija tehnika obrade podataka i vizualizacije u *Python-u* predstavlja moćan alat za detaljnu analizu fajl sistema, pružajući kako kvantitativne, tako i vizuelne uvide u strukturu i ponašanje podataka. Dalji rad može uključivati proširenje funkcionalnosti alata, uključujući interaktivne *dashboard-e*, analizu većih datasetova i integraciju sa sistemima za nadzor i bezbednost podataka.