Analiza velikog skupa podataka pomoću programskog jezika Python i razvojnog okružja Jupyter Notebook

Rosandić, Josip

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Josip Rosandić

ANALIZA VELIKOG SKUPA PODATAKA POMOĆU PROGRAMSKOG JEZIKA *PYTHON* I RAZVOJNOG OKRUŽJA *JUPYTER NOTEBOOK*

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Josip Rosandić

Matični broj: 45862/17–R

Studij: Informacijski sustavi

ANALIZA VELIKOG SKUPA PODATAKA POMOĆU PROGRAMSKOG JEZIKA *PYTHON* I RAZVOJNOG OKRUŽJA *JUPYTER NOTEBOOK*

ZAVRŠNI RAD

Mentor :

Izv. prof. dr. sc. Markus Schatten

Varaždin, rujan 2020.

Josip Rosandić

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Ovaj rad čitatelja uvodi u sferu analize i obrade podataka. Naglasak rada stavljen je na analizu otvorenog skupa podataka pomoću programskog jezika Python i modula Pandas, a sve unutar razvojnog okružja Jupyter Notebook. Nad otvorenim skupom podataka u valjanom datotečnom obliku čitljivom unutar korištenih modula provest će se analiza podataka što uključuje analizu količine podataka, tipova podataka, valjanosti podataka, strukture unutar koje su podaci pohranjeni. Nadalje, rad uključuje i prikaz optimiranja tipova podataka u svrhu postizanja boljih memorijskih performansi. Zatim slijedi semantička analiza podataka, procesiranje skupa te eksploatiranje nove vrijednosti iz postojećih resursa. Procesi obrade većinski su dio Python modula Pandas unutar kojega se nalaze metode za čitanje odgovarajuće datoteke skupa podataka, zatim formiranje podatkovnih okvira kao temeljne obradbene strukture ovoga modula te primjene atributa i metoda ovoga modula nad formiranim podatkovnim okvirima. Novo znanje korisno je vizualizirati korištenjem Python modula Matplotlib. Fokus rada trebao bi biti sami proces koji, promatramo li ga kao crnu kutiju, kao ulaze prima podatke, a kao izlaze vraća odgovarajuće semantički jasne, egzaktne i precizne izračune i vizualizacije.

Ključne riječi: podaci;analiza;python;Pandas;znanje;proces;rudarenje;znanje

Sadržaj

1. Uvod

Ovaj završni rad orijentiran je u najširem smislu ka obradi podataka. Obzirom na to da je obrada podataka vrlo širok pojam, u daljnjem tekstu će se elaborirati sama jezgra na koju je fokusiran rad. Egzaktnije govoreći, u ovome radu, tzv. *proof-of-concept* načinom prikazat će se obrada strukturiranog otvorenog skupa podataka u programskom jeziku Python i njegovim bibliotekama za manipulaciju skupovima podataka i Jupyter Notebooku - za svoju domenu snažnom razvojnom okružju. Sve navedeno alati su koji su trenutno aktualni u svijetu podatkovne znanosti, obrade i strukturiranja te analize skupova podataka. Ovi alati i metode vrlo su horizontalnog karaktera, što će reći da se ovim područjem ne bave isključivo ili u velikoj većini informatičari, kao što je to slučaj s primjerice administracijom računalnih mreža ili programskim inženjerstvom. Ovim područjem, ovim alatima i metodama koriste se i stručnjaci matematičari, ekonomisti, prirodoslovci (biolozi, fizičari) i pripadnici mnogih drugi znanstvenih područja unutar kojih su ovi alati i metode primijenjive. U svijetu kao domeni čija je važna komponenta računalo uopće kao koncept, nezamislivo je bilo što promatrati bez osvrta na podatkovnu komponentu. U svakoj interakciji s bilo kojom vrstom računala od jednostavnih mikrokontrolera do složenih poslužiteljskih hijerarhija i superračunala, podaci su ono što računalo pogoni. Oni su mu hrana bez koje *de facto* ne bi imao što procesirati. Naravno, mi podatke promatramo na nešto višim razinama i u nešto drugačijim oblicima no što ih vidi računalo, ali njihova uloga je u računarstvu fundamentalna. Slijedom toga dolazimo do složenijih podatkovnih struktura unutar kojih semantički strukturiramo podatke kako bismo ih lakše obrađivali i čitali. Tako se primjerice u ovome radu koriste otvoreni, strukturirani skupovi podataka. Otvoreni znači da ih je autor označio slobodnima za dijeljenje i korištenje u daljnjemu radu, strukturirani znači da možemo prepoznati određeni determinizam, odnosno strukturu unutar skupa kao što je primjerice tablica s recima i stupcima koji imaju svoju semantiku. Upravo ta fundamentalna i značajna uloga pruža motivaciju za izradu završnog rada na navedenu temu.

2. Metode, tehnike i alati rada

2.1. Uvod u tehnologije i alate

Postoji mnoštvo metoda i alata pomoću kojih se može vršiti obrada podataka u najširem smislu. U ovome radu fokus je na obradi otvorenog, strukturiranog skupa podataka. Slijedi pregled tehnologija i alata koji su korišteni za obradu podataka:

- Python 3 programski jezik
  - Pandas biblioteka
  - Matplotlib biblioteka
- Jupyter Notebook razvojno okružje

Skup alata korišten u ovome radu podignut je nad Anaconda distribucijom - programskim rješenjem koje objedinjuje alate za manipulaciju podacima, analizu i vizualizaciju podataka. Unutar Anaconda distribucije instalirani su i JupyterLab te JupyterNotebook. U daljnjem radu bit će prikazan proces pokretanja Jupyter Notebooka putem Anacondina tzv. conda terminala.

Anaconda distribucija sastoji se od dva glavna modula: [1]

- Miniconda
- Anaconda

Miniconda modul koji pruža Python interpreter i conda terminal za upravljanje paketima te alatima unutar distribucije na način sličan `apt` ili `yum` alatima unutar Linux terminala.

Anaconda modul kao takav sadrži sve što i prethodno spomenuti, no standardno nešto većega opsega.

Slika 1: Logotipi tehnologija korištenih u ovome radu (Izvori: `www.anaconda.com`; `www.jupyter.org`; `www.python.org`; `www.Pandas.pydata.org`; `www.matplotlib.org`)

2.1.1. Python

Programski jezik Python pripada skupini tzv. **skriptnih jezika** koji najčešće služe pisanju manjih dijelova programa - **skripti** koje služe automatizaciji izvođenja zadataka. Među skriptnim jezicima, Python je stasao na temeljima znanosti i podatkovne analitike. Python je jedan od najvažnijih programskih (skriptnih) jezika današnjice kada je u pitanju podatkovna znanost, strojno učenje te razvoj softwarea. [2, str. 2]

Posljednjih godina programski jezik Python raste u svojoj domeni te dobiva na snazi rješavajući kompleksne znanstvene izračune kao i analizu i vizualizaciju velikih skupova podataka. Takav rast na korisnosti i popularnosti proizlazi iz širokog Pythonova ekosustava koji sadrži pakete i biblioteke za rad u specifičnoj domeni. Neki od njih su **NumPy** - paket za manipulaciju homogenim skupovima podataka, **Pandas** - paket za manipulaciju heterogenim skupovima podataka, zatim **SciPy** - paket za znanstvene izračune, **Matplotlib** - paket za vizualizacije skupova podataka, **IPython** - za interaktivno izvršavanje koda i dijeljenje koda, **Scikit-Learn** za strojno učenje i mnogi drugi paketi. [1]

2.1.2. Jupyter Notebook (Project Jupyter)

Projekt Jupyter neprofitni je projekt temeljen na softwareu otvorenog koda koji je proizašao iz IPython projekta 2014. godine. Projekt Jupyter baziran je na podršci za interaktivnu obradu podataka u okviru podatkovnih znanosti i računarstva. Jupyter, kako navode autori, će uvijek biti u potpunosti otvorenog koda i besplatan za svakog korisnika te promoviran BSD licencom. Unutar projekta Jupyter možemo pronaći **JupyterLab**, **Jupyter Notebook**, **Jupyter Console** te **Qt Console**. [3]

Jupyter Console terminal je za pristup jezgrama koje koriste Jupyter protokol, a Jupyter Notebook grafičko je korisničko sučelje temeljeno na Jupyter Console. [3]

Jupyter Notebook razvojno je okružje otvorenog koda, odnosno web aplikacija koja pruža korisničko sučelje za napredan rad s Jupyterom. Unutar Jupyter Notebooka može se razvijati programski kod, dokumentirati te izvršavati. Ovo okružje sastoji se od dvije komponente: [3]

1. Web aplikacija
2. Notebook dokumenti

Web aplikacija odnosi se na cjelokupni alat smješten u web pregledniku, dok se notebook dokumenti odnose na sve prikaze programskog koda, dokumentacije, inputa, outputa i svega ostaloga unutar te web aplikacije. [3]

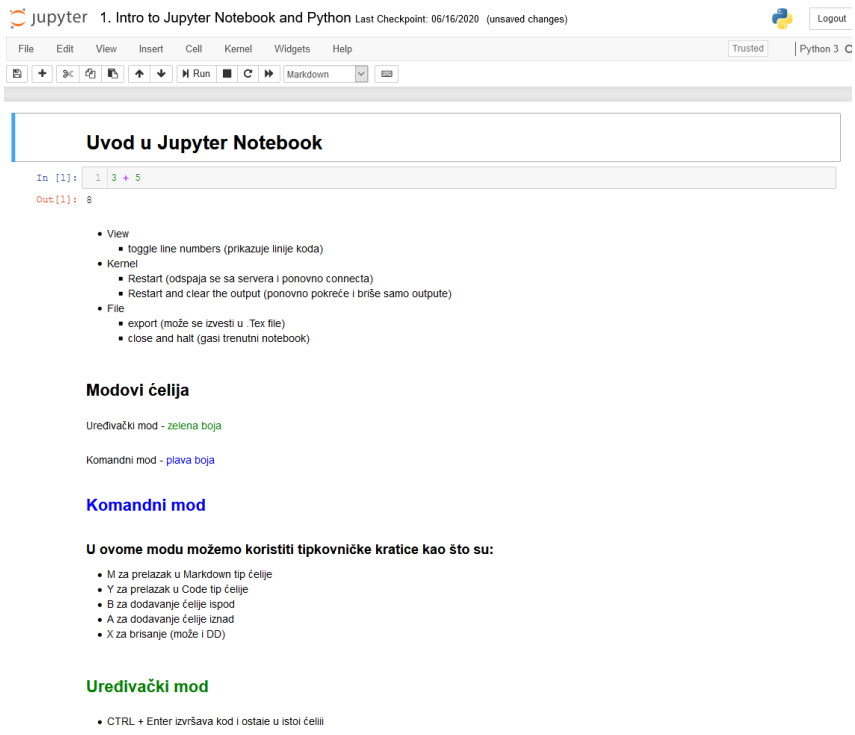
2.1.2.1. Jupyter Notebook okružje

Kako bismo imali pristup svim mogućnostima Jupyter Notebook okružja, a na kraju i rada s istim, u nastavku će biti prikazan proces instalacije, aktivacije i početnog rada u samom

Jupyter Notebook okružju.

1. je korak instalacija **Anaconda** distribucije preuzimanjem s web lokacije https://www.anaconda.com te instaliranjem na lokalno računalo
2. je korak pokretanje **Anaconda prompta** - komandne linije za rad s Anaconda distribucijom
3. unutar same komandne linije Anaconda distribucijom možemo upravljati sljedećim komandama:
   (a) `conda info –envs`
   (b) `conda update conda`
   (c) `conda create –name <nazivRadnogOkružja>`
   (d) `conda activate <nazivRadnogOkružja>`
   (e) `conda deactivate <nazivRadnogOkružja>`
   (f) `conda install Pandas jupyter bottleneck numexpr matplotlib`
   (g) `jupyter notebook`

   Nakon kreiranja radnog okružja te instaliranja i ažuriranja potrebnih biblioteka, Jupyter Notebook okružje pokrećemo unosom komande `jupyter notebook` pri čemu se Jupyter Notebook otvara unutar zadanog internetskog preglednika (slika 2) na localhostu s dediciranim portom na kojemu radi.

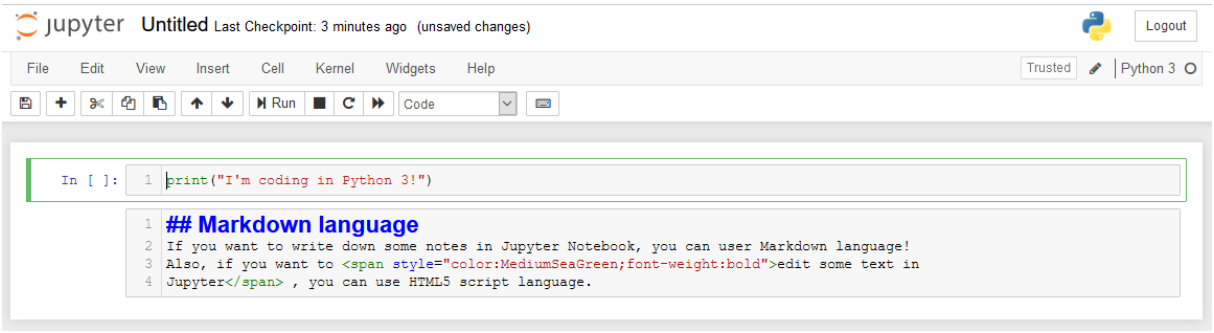


Slika 2: Snimka zaslona Jupyter Notebook sučelja (Izvor: vlastita izrada)

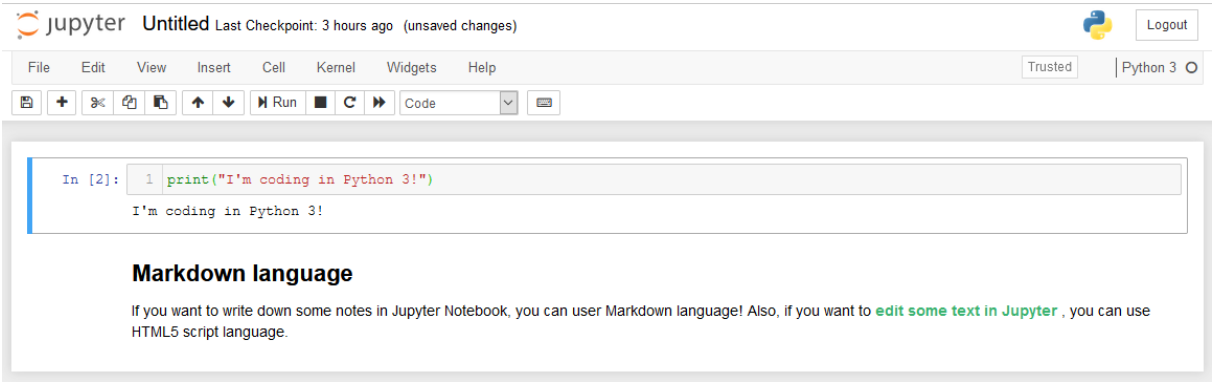
Samo Jupyter Notebook okružje načelno je web aplikacija koja se pokreće u internetskom pregledniku na lokalnom poslužitelju računala. Nakon pokretanja, pristupamo početnom sučelju na kojemu se prikaže korijenski direktorij trenutnog korisnika lokalnog računala, dakle C:\Users\jrosa. S ove lokacije možemo dalje kreirati direktorije i složiti cijelo projektno stablo.

Nakon što smo kreirali željene direktorije, kreiramo novi Notebook odabirom opcije **New** i zatim **Python 3** unutar sekcije **Notebook**. Kreirat će se novi Notebook čiji opis slijedi u nastavku.

Samo Jupyter Notebook okružje temeljeno je na dvije vrste **ćelija** unutar kojih korisnik unosi sve inpute koje daje na procesiranje Jupyteru. Prva vrsta ćelije služi za **izvršavanje kôda** i takvu vrstu ćelije vidimo na slici 3 kao ćeliju unutar koje se nalazi Python linija koda `print("I'm coding in Python 3!")`. Druga vrsta ćelije ćelija je za **kreiranje bilješki** (eng. notes), otkuda i samom okružju Nootebook u nazivu. Unutar potonje vrste ćelija mogu se kreirati tekstualne ili multimedijske bilješke pisanjem u Markdown skriptnom jeziku ili pak uređivanjem HTML/CSS koda. Dakle, ovaj alat koncipiran je kao razvojno okružje koje objedinjuje pisanje i izvršavanje programskog koda s jedne strane te napredno dokumentiranje istoga koda pomoću ranije navedenih tehnologija.



Slika 3: Snimka zaslona Jupyter Notebook sučelja prije izvršavanja koda (Izvor: vlastita izrada)



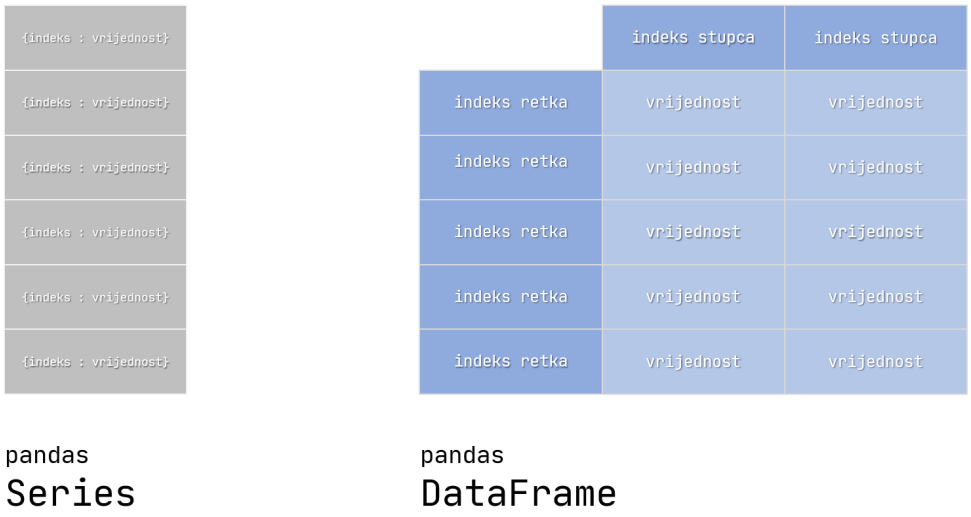
Slika 4: Snimka zaslona Jupyter Notebook sučelja nakon izvršavanja koda (Izvor: vlastita izrada)

Na slikama iznad vidimo primjere dva osnovna tipa ćelija te unos programskog koda i ispis nakon izvršavanja ćelije. Isto tako, na slici 3 vidimo zelenu crtu lijevo od ćelije dok na slici 4 vidimo plavu crtu lijevo od ćelije. Zelena crta označava da je ćelija trenutno **aktivna** dok plava

crta označava kako ćelije trenutno **nije aktivna**.

2.1.3. Pandas

Pandas je biblioteka koja pruža strukture podataka i metode kreirane kako bi rad sa strukturiranim ili tabularno posloženim podacima bio brži, lakši i pouzdaniji. [2] Od svoje inicijalne pojave 2010. godine, Pandas biblioteka omogućila je programskom jeziku Python da utvrdi svoju snagu i postane zaista ozbiljan alat za podatkovne analize. Temeljni entitet oko kojega se odvija posao u Pandas okruženju je `Series`, nazovimo je serija - jednodimenzionalna struktura s označenim recima koja ima jedan stupac te `DataFrame`, nazovimo ga podatkovnim okvirom - dvodimenzionalna tabularno orijentirana struktura podataka s označenim recima i stupcima.



Slika 5: Ilustracija osnovnih Pandas struktura podataka (Izvor: vlastita izrada)

U načelu, Pandas kombinira visoke performanse po uzoru na NumPy - biblioteku s kojom je Pandas inicijalno bila neraskidivo povezana te mogućnosti manipulacije podacima kakve pružaju relacijske baze podataka, odnosno upitni jezik SQL (Structured Query Language). Dakle, Pandas na temelju napredne mogućnosti indeksiranja pruža lakoću oblikovanja, dijeljenja, agregiranja, selektiranja i ostalih operacija nad podskupovima podataka.

Sam autor Pandas biblioteke dolazak na ovu ideju opisao je na sljedeći način: "*...Pandas sam počeo graditi početkom 2008. godine tijekom rada u kompaniji koja se bavi upravljanjem investicijama. Tada sam imao potrebu nekoliko željenih funkcionalnosti s kojima sam se svakodnevno susretao spojiti u jedinstven alat. Htio sam strukturu podataka s označenim osima dimenzija s automatskim poravnanjem podataka, zatim integrirana mogućnost vremenskih serija, vremenske i nevremenske serije u istoj podatkovnoj strukturi, aritmetičke operacije i operacije redukcije podataka, upravljanje nedostajućim podacima te relacijske operacije po uzoru na SQL upitni jezik.*" [2]

2.1.3.1. Series

Pandas **serija** jednodimenzionalan je objekt koji sadrži slijed vrijednosti i pripadnih indeksa. Na grafici lijevo prikazan je koncept Pandas serije koja se sastoji od elemenata - redaka koji sadrže vrijednosti i pripadajući indeks pridružen svakoj vrijednosti. Pandas serija mora sadržavati podatke istog tipa za razliku od python lista koje mogu sadržavati različite tipove - konzistencija tipova podataka.

Slika 6: Pojednostavljeni grafički prikaz Pandas serije (Izvor: vlastita izrada)

Konstruktor Pandas serije prikazan je u nastavku:

```
Series([data, index, dtype, name, copy, ...])
```
[4]

gdje se parametri tumače na sljedeći način:

- **data** - polja, iterabilne strukture, rječnici ili skalar
- **index** - polje vrijednosti
- **dtype** - str, numpy.dtype, ExtensionDtype ili opcionalno
- **name** - str ili opcionalno
- **copy** - bool (standardna vrijednost = `False`) [4]

Kreiranje Pandas serije u programskom kodu prikazano je u nastavku:

```
# Primjer kreiranja Pandas serije
podaci = array(['4','2','6','3'])
s = pd.Series(podaci)
s

# Ispis
# 0 4
# 1 2
# 2 6
# 3 3
```

Na prikazanom ispisu vidimo indekse s lijeve strane i vrijednosti s desne strane. Budući da nismo eksplicitno zadali indekse, Pandas dodjeljuje indekse kao cijele brojeve od 0 do N - 1 pri čemu je N duljina podatkovne serije, odnosno broj redaka u seriji, odnosno broj elemenata inicijalne strukture na temelju koje je serija kreirana.

Indekse možemo i eksplicitno zadati na sljedeći način:

```
serija = pd.Series([4, 2, 6, 3], index=['x', 'y', 'z', 'q'])
serija

# Ispis
# x 4
# y 2
# z 6
# q 3
dtype: int64
```

Pandas seriju možemo promatrati i kao **rječnik** obzirom na to da mapira indekse i vrijednosti. Slijedno tomu, python seriju možemo kreirati i tako da metodi `pd.Series()` jednostavno proslijedimo rječnik. Prikaz u programskom kodu slijedi u nastavku:

```
rjecnik = { "Zagreb" : 10000, "Varazdin" : 42000, "Slavonski Brod" : 35000 }
serijaIzRjecnika = pd.Series(rjecnik)
serijaIzRjecnika

# Ispis
# Zagreb            10000
# Varazdin          42000
# Slavonski Brod    35000
dtype: int64
```

Proslijedimo li rječnik Pandas metodi `.Series()`, ključevi iz rječnika u novoj seriji postat će indeksi, dok će vrijednosti iz rječnika postati vrijednosti serije.

Atributi i metode Pandas serije

Pandas seriji pridruženo je oko 20 atributa i 3 do 4 puta više metoda. Neke od najvažnijih slijede u nastavku:

Neki od **atributa** koji su primjenjivi na seriju su:

- `values` - vraća sve vrijednosti serije
- `index` - vraća raspon i prirodu indeksa serije
- `dtype` - vraća tip podatka vrijednosti serije
- `is_unique` - vraća true ako su sve vrijednosti serije jedinstvene
- `size` - broj elemenata serije (veličina serije)
- `name` - naziv serije

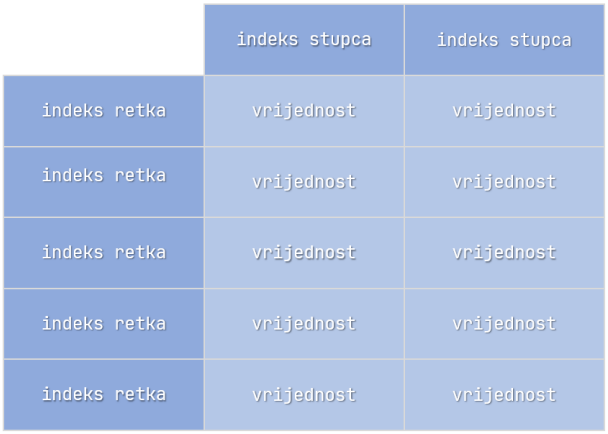
Neke od **metoda** koje su primjenjive na seriju su: [4]

- `sum` - vraća zbroj vrijednosti serije (u slučaju da su podaci numeričkog tipa)
- `product` - vraća umnožak vrijednosti serije (u slučaju da su podaci numeričkog tipa)
- `mean` - vraća aritmetički sredinu vrijednosti serije (u slučaju da su podaci numeričkog tipa)
- `sort_values` - sortira vrijednosti serije, standardno u rastućem poretku
- `sort_index` - sortira indekse serije, standardno u rastućem poretku
- `get` - dohvaća vrijednost na indeksu koji se proslijedi kao parametar
- `std` - vraća standardnu devijaciju
- `min` - vraća najmanji element serije
- `max` - vraća najveći element serije

- `median` - vraća medijalni element serije
- `mode` - vraća modalni element serije
- `describe` - vraća cjelokupni statistički izvještaj o podacima serije u obliku nove serije, a u koju su uključene metode median, std, min, max, ali i kvartili i slično

2.1.3.2. DataFrame

DataFrame predstavlja pravokutnu, tabularnu strukturu podataka, koja se sastoji od stupaca od kojih svaki stupac može imati vlastiti tip podatka (numerički, string, boolean...). Na grafici lijevo prikazan je koncept Pandas DataFramea. DataFrame se sastoji od indeksa stupca i indeksa retka. Slikovito govoreći, DataFrame je zapravo **rječnik serija u kojemu sve serije dijele isti indeks**.



Slika 7: Pojednostavljeni grafički prikaz Pandas data framea (Izvor: vlastita izrada)

DataFrame možemo kreirati na više načina. Neki od načina su uvoz CSV datoteke u kojoj je predefinirana struktura, zatim kreiranje DataFramea iz rječnika u kojemu su vrijednosti polja elemenata, a ključevi su indeksi stupaca (indeksi redaka automatski se dodjeljuju kao prirodni brojevi od 0 do N - 1). U nastavku slijedi popis načina kreiranja DataFramea:

- rječnik jednodimenzionalnih polja
- dvodimenzionalno polje
- serija
- dataframe

Konstruktor Pandas data framea prikazan je u nastavku:

`DataFrame([data, index, columns, dtype, copy])` [4]

gdje se parametri tumače na sljedeći način:

- **data** - n-dimenzionalno polje, iterabilne strukture, rječnik ili data frame
- **index** - polje vrijednosti indeksa redaka
- **columns** - polje vrijednosti indeksa stupaca
- **dtype** - str, numpy.dtype, ExtensionDtype ili opcionalno
- **name** - str ili opcionalno

- **copy** - bool (standardna vrijednost = `False`) [4]

Slijedi primjer kreiranja DataFramea u nastavku:

```
# Primjer
rjecnikVodostaj = {'grad': ['ZG', 'VZ', 'SB', 'VU', 'KA'], 'sat': [7, 7, 7,
 ↪  7, 7], 'vodostaj': [15, 20, 10, 18, 23]}
dataFrameVodostaj = pd.DataFrame(rjecnikVodostaj)
dataFrameVodostaj

#   Ispis
#   grad      sat     vodostaj
#   ZG        7       15
#   VZ        7       20
#   SB        7       10
#   VU        7       18
#   KA        7       23
```

Svaki stupac DataFramea je jedna serija koja ima indekse jednake indeksima redaka DataFramea. U slučaju da želimo iz DataFramea ekstrahirati samo jedan stupac, to možemo učiniti na sljedeće načine: `dataFrame['nazivStupca']` ili `dataFrame.nazivStupca`.

Atributi i metode Pandas DataFramea

Neki od **atributa** koji su primjenjivi na DataFrame su:

- `values` - vraća sve vrijednosti DataFramea
- `index` - vraća raspon i prirodu indeksa DataFramea
- `shape` - uređeni par koji predstavlja dimenzije DataFramea
- `columns` - vraća nazive (indekse) svih stupaca DataFramea
- `axes` - vraća raspon indeksa DataFramea
- `loc` - vraća podskup data framea prema najčešće ne-numeričkom indeksu stupca ili retka
- `iloc` - vraća podskup data framea prema numeričkom indeksu retka

Neke od **metoda** koje su primjenjive na DataFrame su:

- `info` - vraća osnovne informacije o data frameu kao što su broj stupaca, tipovi podataka po stupcima i veličina data framea
- `sum` - izvršimo li metodu `.sum()` nad data frameom, dobit ćemo seriju u kojoj su stupci data framea indexi, a nad podacima svakog stupca izvršena je metoda zbrajanja
- `head` - vraća prvih nekoliko elemenata data framea

- `tail` - vraća zadnjih nekoliko elemenata data framea
- `value_counts` - broji i vraća pojave vrijednosti u data frameu
- `dropna` - briše elemente unutar kojih se pojavljuje NULL vrijednost
- `fillna` - NULL vrijednosti mijenja zadanom vrijednošću
- `astype` - zamjena tipa podatka
- `sort_values` - sortira vrijednosti data framea
- `sort_index` - sortira indekse data framea
- `rank` - vrijednostima pridružuje vrijednost (rangira ih)

2.1.3.3. Pandas ulazni i izlazni tokovi

Pandas biblioteka prihvaća uvoz nekoliko različitih formata datoteka kao što su `.csv`, `.xlsx`, `.json`, `.html`, `.sql` i ostali navedeni u nastavku: [4]

1. Tekstualna datoteka kao spremište podataka (primjerice `.csv`)
2. Međuspremnik (eng. *clipboard*)
3. Excel
4. JSON
5. HTML
6. HDFStore (HDF5)
7. Feather (brza pohrana DF-a)
8. Apache Parquet
9. Apache ORC
10. SAS
11. SPSS
12. SQL
13. Google BigQuery
14. STATA

2.1.4. Primjena Pandas biblioteke unutar realnih problemskih domena

2.1.4.1. Upotreba Pandas biblioteke prilikom izrade fotografije crne rupe

Koristeći globalnu mrežu teleskopa Event Horizon, znanstvenici su konstruirali sliku crne rupe u centru M87 galaksije - supermasivne crne rupe u centru naše galakcije - Mliječne staze. Pandas biblioteka korištena je za normalizaciju podataka prikupljenih pomoću masivnih teleskopa. Naime, obzirom na to da je crna rupa toliko daleko da bi za njezino otkrivanje bio potreban teleskop veličine same planete Zemlja, znanstvenici su odlučili isti posao obaviti pomoću više manjih teleskopa. Snimljeni sadržaj svih teleskopa pohranjen je na tvrde diskove i poslan u laboratorij massachusettskog tehnološkog instituta u Sjedninjenim Američkim Državama gdje je sav materijal učitan u Pandas. Podaci su normalizirani, sinkronizirani u vremenu, uklonjene su podatkovne interferencije Zemljine atmosfere itd. Nakon normalizacije, podaci su poslani algoritmima za detekciju vrste slike koji su odradili konačan posao. [5]

Slika 8: Slika crne rupe kao rezultat projekta iz 2019. godine (Izvor: [6])

2.1.4.2. Upotreba Pandas biblioteke za predviđanje tržišnih kretnji unutar financijskog sektora

Mnoge financijske institucije, uz biblioteke za strojno učenje, koriste i Pandas biblioteku prilikom odlučivanja jesu li im nadolazeći podaci relevantni i korisni kao potpora donošenju odluka. Novi setovi podataka vrlo se često učitavaju u Pandas i normaliziraju kao takvi, a zatim se uspoređuju s povijesnim tržišnim podacima kako bi se pronašla možebitna korelacija s tržišnim vrijednostima. Ako se pokaže da su podaci korisni, isti se dalje prosljeđuju menadžerima koji su tada u mogućnosti raditi s kvalitetnim i sigurnim setovima podataka. Financijske institucije također koriste Pandas prilikom monitoringa (promatranja) svojih internih sustava. Naime, oni traže odstupanja ili neobične aktivnosti prilikom uobičajenih tržišnih aktivnosti te prema podacima korigiraju sustav. [5]

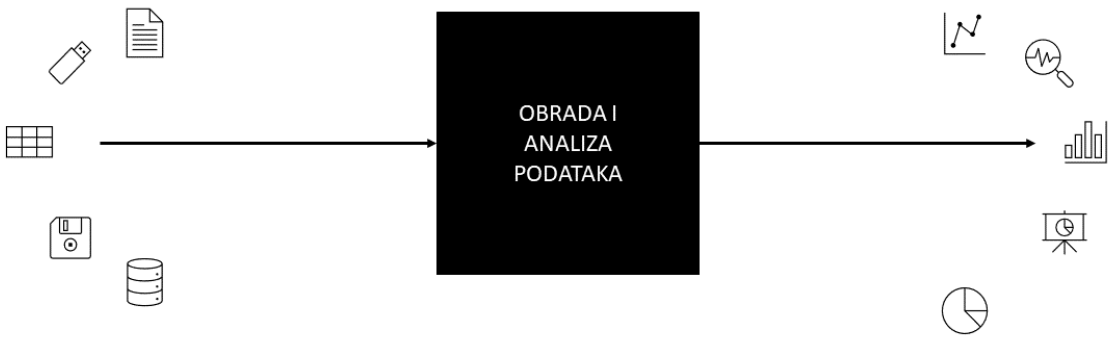
2.2. Metode i tehnike izrade rada

Cilj ovoga završnog rada je na jezgrovit način prikazati osnovne koncepte analize i obrade strukturiranih otvorenih podataka koristeći Python i njegove module Pandas te Matplotlib kao alate rada. Sami rad kreiran je na temeljima **proučavanja tehnika rada** u Pandas biblioteci, manipulacije podacima te **proučavanju** osnovnih Pandas komponenti i koncepata. **Istraživanjem literature** temeljene na Pythonu kao osnovi rada te Pythonivim modulima te razvoj programskog koda na temelju istražene literature osnovni je način kreiranja međurezultata i rezultata u ovome radu.

Nadalje, **istraživanje valjanih otvorenih skupova podataka** vrlo je zahtjevan i važan korak kao priprema za metodološki pristup obradi u Pandasu. Naime, kako bi rad u Pandas modulu bio moguć, vrlo je važno da sami skup podataka bude strukturiran i da njegovi tipovi podataka budu usklađeni i konzistentni. Također, nakon same **obrade i analize**, važna je i **semantička evaluacija rezultata istraživanja** kako bi se otkrili potencijalni propusti te kako bi se popravila kvaliteta samoga rada.

3. Teorija obrade i analize podataka

Analiza i obrada samih skupova podataka je **proces**. Vrlo je važno tu činjenicu imati na umu iz razloga što se shvaćanje procesa temelji na tvrdnji da se svaki proces po definiciji sastoji od točno definiranih aktivnosti koje ga čine. Promatramo li taj proces kao crnu kutiju što je prikazano na slici 9, dakle, promatramo li isključivo ulaze (eng. *inpute*, resurse) u proces i izlaze (eng. *outpute*, rezultate) iz procesa, bez zanimanja za prirodu i detalje aktivnosti, možemo primijetiti dvije osnovne komponente. Prva komponenta bili bi **resursi** - podaci te software i hardware pomoću kojih se isti obrađuju. Podatke općenito možemo promatrati kao strukturirane i nestrukturirane. Strukturirani su podaci najčešće pohranjeni u obliku uređenih podatkovnih struktura - tablica, , lista, polja i ostalih podatkovnih struktura te kao takvi imaju točno određenu strukturu s definiranim oznakama, brojnošću elemenata i ostalim relevantnim kvantifikatorima. Nestrukturirani podaci mogu biti vrlo kompleksni prilikom obrade obzirom na to da njihova semantička struktura nije poznata onome tko te podatke obrađuje. Često su to razni tekstualni sadržaji različitih tipova podataka, ali i multimedijski sadržaji čija klasifikacija ulazi u sferu strojnog i dubokog učenja. Kao izlaz (eng. *output*) procesa obrade i analize podataka najčešće proizlazi **dodana vrijednost**, informacija i u konačnici, ono najvažnije - **znanje** koje služi kao ključni resurs za napredak i razvoj.



Slika 9: Proces obrade i analize podataka kao crna kutija (Izvor: vlastita izrada)

No kako bismo podrobnije razumjeli problematiku, smjestimo neke od temeljnih pojmova vezanih uz znanje kao resurs u vlastiti kontekst. Prema [7], **znanje** je definirano kao skup činjenica, informacija i vještina stečenih izobrazbom ili iskustvom radi teorijskoga ili praktičnoga razumijevanja i rješavanja problema. Nadalje, prema [8], znanje se sastoji od istina i vjerovanja, pogleda i koncepata, prosudbi i očekivanja te metodologija i vještina. Također, [8] daje primjere

pogleda na znanje pa tako objašnjava situaciju u kojoj znanstvenik - istraživač uragana određuje vjerojatnost specifične putanje uragana na temelju analize x i y varijabli te prognoziranih putanja uragana dobivenih prethodnim softverskim obradama skupova podataka.

Upravo spomenute definicije pojma znanja pružaju podrobnije shvaćanje smisla i temelja ovoga procesa obrade (sirovih) podataka. Dakle, u tom procesu ulazni se podaci, strukturirani ili nestrukturirani, transformiraju u informacije ili znanje. To isto eksploatirano znanje dalje se koristi u upravljanju, odlučivanju, vođenju, poboljšanju i ostalim aktivnostima.

Na tragu spomenutoga, pojavljuje se pojam **Data-Driven Decision Making (DDDM)** koji podrazumijeva temeljenje poslovnih odluka na rezultatima analize podataka radije no oslanjanje na intuiciju ili čak iskustvo. [9, str. 5]. Primjerice, trgovac može plasirati oglase temeljene isključivo na dugogodišnjem iskustvu u svom polju ili pak svoje odluke može temeljiti na rezultatima analize podataka, no prednosti drugoga pristupa su višestruke. Naime, ekonomist Erik Brynjolfsson i njegovi kolege s Massachusetts tehnološkog insituta (MIT-a) proveli su studiju o utjecaju DDDM-a na produktivnost poslovnih sustava. Razvili su metriku koja mapira kompanije prema korištenju DDDM-a u poslovanju. Pokazalo se da su kompanije koje su više oslonjene na DDDM znatno uspješnijih rezultata. [9, str. 6]

3.1. Obrada podataka kao skup aktivnosti

Unatoč dobro razvijenim algoritmima za eksploataciju i obradu podataka, postoji samo mali broj temeljnih aktivnosti koje spomenuti algoritmi mapiraju, odnosno simuliraju. U poslovnoj domeni, prilikom analize podataka, važno je pronaći međuodnos varijabli i entiteta kojeg te varijable opisuju. Budući da smo prethodno elaborirali kako je eksploatacija i obrada podataka proces, ona se, dakle, može sastojati od samo nekoliko fundamentalnih aktivnosti. Prema [9, str. 20] to su:

- **Klasifikacija** - svrstavanje entiteta unutar jedne od grupa (klasa) koje su najčešće međusobno disjunktne.

  Npr. Ako je u pacijenta potvrđena pojavnost određenog virusa, on će pripadati klasi *zaraženi*, dok će pacijent kojemu ista nije potvrđena pripadati klasi *nezaraženi*. Pacijent ne može istodobno pripadati objema klasama. To je intuitivno jasno.

- **Regresija** - aproksimiranje numeričkih varijabli karakterističnih za pojedinu jedinku na temelju prethodnih vrijednosti.

  Npr. Svaki pacijent ima pridružen atribut, odnosno podatak o tomu koliko dugo je bio zaražen ili pak koje simptome je imao. Na temelju tih podataka prethodnih pacijenata, regresijom možemo doći do predviđene vrijednosti za nekog budućeg pacijenta.

- **Povezivanje prema sličnosti** - identifikacija entiteta čije vrijednosti atributa su slične.

  Npr. Povezivanje pacijenata koji imaju sličan skup simptoma.

- **Klasterizacija** - grupiranje entiteta, ali ne ciljano ili željeno.

  Npr. Pacijenti se prirodno dijele na one koji do liječnika idu pješice ili pak automobilom.

- **Skupna pojavnost** - pojavljuju li se određene pojave u paru češće no što je to očekivano

  Npr. Dolaze li teži oblici simptoma pacijenta u paru s većim brojem godina.

- **Profiliranje** - nastojanje da se okarakterizira ponašanje entiteta ili grupe entiteta.

  Npr. Jesu li zaraženi pacijenti češće među velikim grupama ljudi ili se ne kreću među ljudima.

- **Određivanje povezanosti** - određivanje određenih veza među entitetima ili grupama entiteta te njihovih važnosti i jačina.

  Npr. U slučaju u kojemu je pacijent kronični bolesnik određene vrste, moglo bi mu se preporučiti da prije posjeti liječnika koji će utvrditi je li zaražen ili nije.

- **Redukcija i optimiranje podataka** - reduciranje opsega skupa podataka s manjim gubicima.

  Npr. Smanjimo li broj varijabli koje nam nisu od primarne važnosti, možemo dobiti skup iz kojega možemo izvući vrijednost, ali koji je ujedno i manji.

- **Kauzalno modeliranje** - otkrivanje uzroka određene akcije.

  Npr. Određivanje je li pacijent zaista zaražen i ima simptome ili je liječniku došao zato jer je bio u prostoriji sa zaraženom osobom te mu se, upravo zbog toga, čini kako ima simptome . Dakle, koji je uzrok njegova dolaska.

[9, str. 20]

3.2. Proces obrade i analize podataka

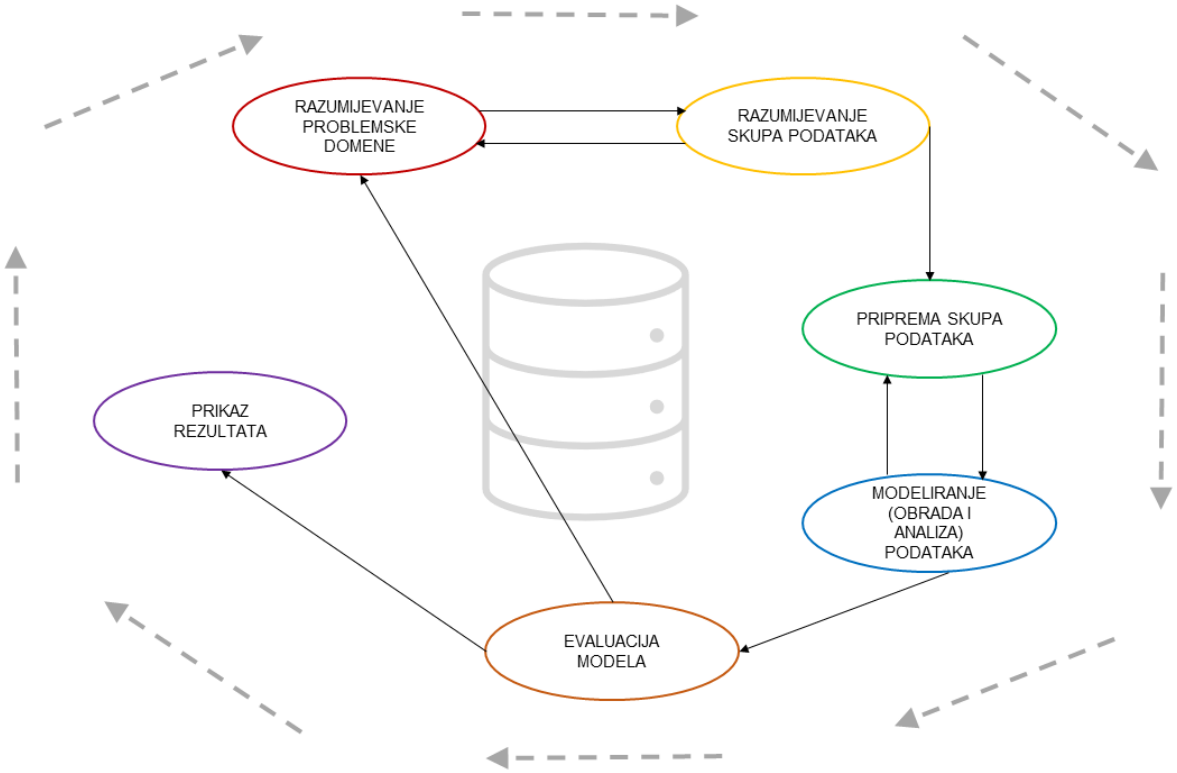
Kako bi proces obrade i analize podataka bio jasniji, preuzeta je i pripremljena grafika koja ga prikazuje. Ključno je za primijetiti da se proces obrade i analize sastoji od glavnih aktivnosti koje se na nižim razinama razlažu na manje aktivnosti, ali isto tako i da je proces obrade **cikličan**, odnosno **iterativan**. Dakle, prilikom obrade podataka, u prvoj iteraciji se možda neće dobiti zadovoljavajući rezultat visoke vrijednosti, no to ne mora nikako značiti neuspjeh. U sljedećim iteracijama moguće je popraviti rezultate prethodnih iteracija.

3.2.1. CRISP metodologija obrade i analize podataka

CRISP (**CR**oss **I**ndustry **S**tandard **P**rocess for Data Mining) procesni model rudarenja, obrade i analize podataka održava konzorcij vodećih podatkovnih eksploatora i korisnika u toj domeni kompanije DaimlerChryslerAG, SPSS (Statistical Package for the Social Sciences), NCR (National Cash Register), i OHRA osiguravajuća kuća. Također, projekt CRISP podupire i Europska komisija. [10] Sami CRISP iterativan je proces koji se u svom najširem smislu sastoji od 6 aktivnosti koje se mogu ponavljati do zadovoljavajuća rezultata. CRISP procesni model hijerarhijski je model u kojemu se svaka osnovna aktivnost dalje razlaže na manje aktivnosti,

dakle, specijalizirane zadatke ili pak samo instance procesa. Kao što grafika na slici 10 prikazuje, cijeli proces se temelji na podacima, odnosno vrti oko istih što je prikazano simbolom baze podataka u sredini ciklusa, a strelice simboliziraju cikličku prirodu ovoga procesa.

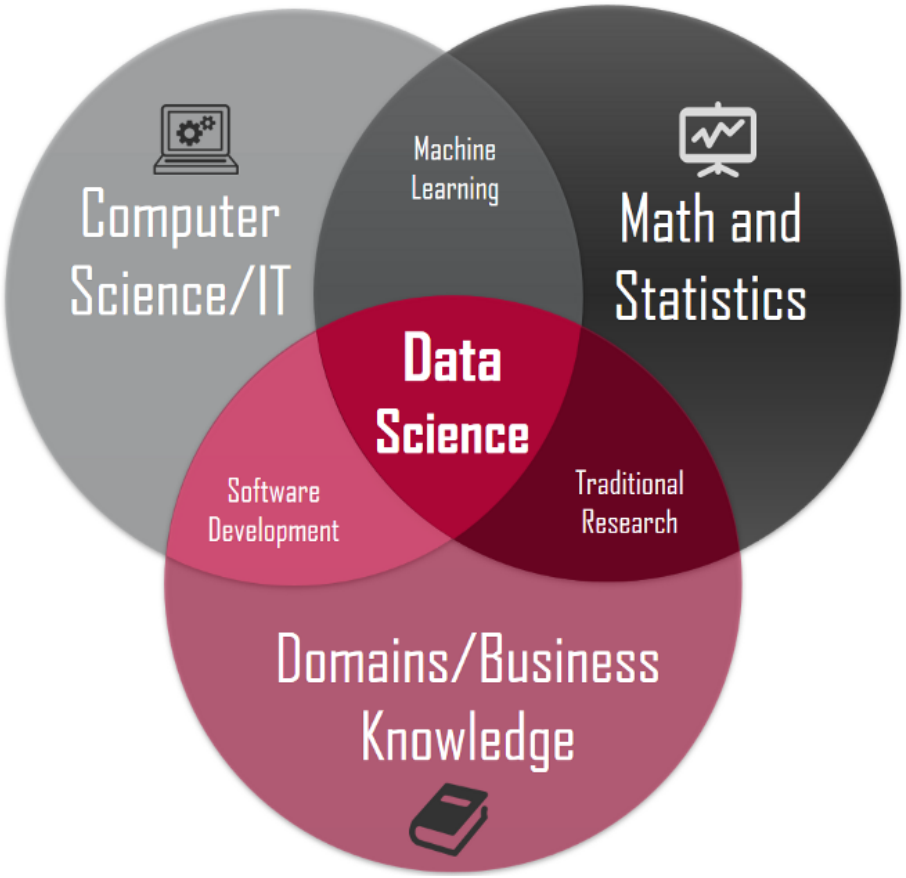
Važno je naglasiti kako je sami CRISP model potvrdila struka, odnosno kako je model dokazano efektivan. Isto tako treba znati da je CRISP model takav model koji će najbolje rezultate dati na projektima šireg opsega na kojima radi veliki broj ljudi. Kod projekata manjeg opsega, određena poboljšanja i dobici na efektivnosti neće biti toliko vidljivi, dok će na većim projektima ovaj model itekako dati svoje rezultate. Ovaj model odabran je u ovome radu zbog toga što pruža metodološki i znanstveno elaboriran pristup aktivnostima analize i obrade podataka te je kao takav metodološki primjenjiv kako na velike, korporativne projekte, tako i na manje projekte ili istraživačke aktivnosti.



Slika 10: CRISP proces obrade i analize podataka s osnovnim aktivnostima (Izvor: vlastita izrada, prema [9, str. 27])

U nastavku rada, obradit će se proces obrade i analize podataka pomoću metodologije CRISP procesnog modela. Sami model sastoji se od sljedećih aktivnosti:

- Razumijevanje problemske domene
- Razumijevanje skupa podataka (podatkovne domene)
- Priprema skupa podataka
- Modeliranje podataka
- Evaluacija modela
- Prikaz rezultata



Slika 11: Vennov dijagram s prikazom temeljnih koncepata unutar podatkovne znanosti (Izvor: [11])

3.2.2. Razumijevanje problemske domene

Iako se čini razumnim i logičnim, bitno je naglasiti kako je prvi korak u procesu obrade podataka razumijevanje problemske domene. Naime, vrlo je važno razumjeti što je srž i bit onoga što skupovi podataka opisuju. Što je to što ćemo mi obrađivati i iz čega će proizaći znanje. Naravno, kako bismo to isto znanje mogli i znali interpretirati kao rezultat obrade i analize podataka, vrlo je važno dobro poznavati problemsku domenu.

U ovoj fazi veliku ulogu igra poslovni analitičar čija uloga je formulirati poslovni problem kao problem kako bi ga uopće bilo moguće modelirati i opisati podacima. Najčešće je ovaj dio točka koja je vrlo bitna kako bi projekt bio što uspješniji. Uobičajeno je da se problem razvije i strukturira tako da se razloži na potprobleme, odnosno podzadatke koji su zasebni modeli npr. klasifikacije, regresije i slično. Jedan od najvažnijih koncepata unutar obrade podataka o kojemu je vrlo važno promišljati uključuje planiranje i razmišljanje o onomu na čemu će se raditi. Pitanja koja se u ovome slučaju postavljaju su *Što točno želimo dobiti kao rezultat? Na koji način bismo željeli doći do toga?* i slično. [9, str. 28]

3.2.3. Razumijevanje podatkovne domene

Ako na rezultat naše obrade i analize podataka gledamo kao na svojevrsni cilj, što taj rezultat i jest, onda su podaci materijal kojim ćemo taj rezultat izgraditi. Podaci su ono oko čega se i na čemu se i temelji cijela paradigma podatkovne znanosti. Oni su jezgra, materijal, sirovina, input koji ulazi u ogromni proces i iz kojega izlaze rezultati tog željenog procesa. Važno je znati prepoznati koliko su podaci snažni ili pak nisu snažni, bolje rečeno, koliko su korisni ili nisu korisni. Prije no što uopće započnemo obradu, moramo promisliti isto tako jesu li podaci koje imamo relevantni glede onoga što želimo dobiti kao rezultat. [9, str. 28]

Ovdje dolazimo i do prirode izvora podataka koje obrađujemo. Podatke, naravno, možemo prikupljati primjerice ručno. Takve situacije vrlo su rijetke, posebice ako se radi o vrlo velikim skupovima podataka koji prestavljaju input računalu. Podaci se najčešće prikupljaju automatiziranim načinima pomoću računala - primjerice pružanjem ankete korisnicima, automatizmom prikupljamo i sortiramo podatke koje nam korisnici pošalju. Podaci se uvelike prikupljaju putem interneta, putem raznih usluga, praćenjem korisnika i njihovih aktivnosti na mreži i sl. Podatke prikupljaju i neovisni uređaji - senzori, računala specijalizirana za neinteraktivno prikupljanje podataka, mjerne stanice, kamere itd.

Vrijednost pojedinih skupova podataka veća je no što bi se moglo zamisliti. Menadžerima u velikim kompanijama vrlo su bitni podaci koji nisu javno i lako dostupni, ali bi ulaganjem u njihovu eksploataciju vjerojatno mogli povećati dodanu vrijednost. U tom kontekstu, skupovi podataka mogu imati vrlo visoke cijene, odnosno mogu se prodavati ili kupovati. Tada se radi o važnim podacima koji bi bili primjesa ili pak primarni input nekom od procesa obrade podataka. Možemo slobodno zaključiti kako su podaci i investicija. [9, str. 29]

3.2.4. Priprema skupa podataka

Koliko god snažna bila neka od analitičkih tehnika obrade podataka, vrlo je vjerojatno da će ta tehnika imati određene zahtjeve prema podacima kao materijalu za obradu. Takvi zahtjevi su, primjerice oblik u kojemu se podaci nalaze, što dakle zahtjeva pretvorbu podataka iz jednog oblika u drugi. To implicira da je priprema podataka usko povezana uz razumijevanje podataka. Najčešći primjeri pripreme podataka su pretvaranje podataka u tablični oblik, uklanjanje nedorečenih ili nepostojećih (*NULL*) vrijednosti, pretvaranje podataka iz jednog tipa podatka u drugi tip podatka (primjerice radi optimizacije memorije).

3.2.5. Modeliranje podataka

Prije svega, definirajmo pojam modela. **Model** je pojednostavljena reprezentacija realnosti koja služi specifičnoj svrsi [9]. Model se temelji na pretpostavkama o tomu što je važno, a što ne glede specifičnog problema. Primjerice, zemljovid je model fizičkoga svijeta. Apstrahira veliki broj informacija za koje je kreator karte odlučio da su temeljne i najvažnije za taj dani model te na taj način pojednostavljuje shvaćanje o predmetu modeliranja.

Prilikom modeliranja podataka, u najširem smislu, tehnike modeliranja primjenjuju se nad podacima. Tehnike obrade način je, odnosno svojevrsna strategija kojom se iz početnih resursa dolazi do rezultata. Neke od tehnika su: [9, str. 35]

1. Statistička analiza
2. Upiti nad bazom podataka
3. Promatranje zakonitosti unutar skladišta podataka
4. Regresijska analiza
5. Strojno učenje i rudarenje podataka

Statistička analiza termin je koji se odnosi na izračunavanje specifičnih vrijednosti od interesa iz skupa podataka pa se tu radi o sumama, prosjecima, medijanima itd. Taj dio naziva se sumarnom statistikom koja je temelj mnogim teorijama i praksama unutar podatkovnih znanosti. No sa statistikom je potrebno vladati oprezno budući da su pokazatelji vrlo egzaktni i zahtijevaju znanje onoga tko se njima koristi. Najbolji primjer kao dokaz ovoj tvrdnji je korištenje aritmetičke sredine i medijana. Naime, ako računamo prosječnu plaću za neko područje, možemo reći kako je prosječna plaća vrlo visoka, ali veliki broj stanovnika ima vrlo niska primanja. Aritmetička sredina je prikladna srednja vrijednost za homogene statističke skupove s niskom razinom varijabilnosti. Na nju utječu sve jedinice niza, a izrazito ekstremne vrijednosti mogu značajno povećati ili smanjiti njenu vrijednost. U ovoj prilici potrebno je kao metriku upotrijebiti medijan obzirom na to da on u danom slučaju daje točnije podatke - govori nam kako polovica stanovnika zarađuje manje od X novčanih jedinica, dok druga polovica zarađuje više od X novčanih jedinica. [9, str. 36]

Upiti nad bazom podataka odnose se na, nazovimo "zahtjeve" za podskupom podataka koje šaljemo prema bazi podataka. Upiti nad bazom podataka formuliraju se točno određenim jezicima koji su specifični toj svrsi, a koje sama baza, odnosno sustav za upravljanje bazom podataka (SUBP) razumije i može izvršiti. Alati za izvršavanje ovakvih upita, odnosno SUBP najčešće razumiju strukturirani upitni jezik (Structured Query Language - SQL). Analitičar u SQL-u formulira upit koji bi prirodnim jezikom glasio primjerice: "Tko su korisnici s prezimenom Horvat koji imaju više od 20 godina?" te ga potom formulira u SQL jeziku na sljedeći način:

```
SELECT * FROM korisnik WHERE korisnik.prezime == "Horvat" AND korisnik.dob
 ↪  > 20;
```

[9, str. 37]

Promatranje zakonitosti unutar skladišta podataka tehnologija je koju koriste često velike kompanije želeći imati uvid i analitiku nad svim procesima te integracijom metrike više procesa istovremeno. Primjerice tvrtke mogu analizirati metrike odjela prodaje i proizvodnje, ali i financija te pronaći željene uzorke koji im mogu pomoći u usmjeravanju poslovanja. [9, str. 38]

Regresijska analiza odnosi se na pronalazak specifičnih uzoraka na postojećim skupovima podataka i pokušaj primjene na budućim skupovima. Točnije, regresijska analiza alat je za predikciju kretanja varijabli temeljem postojećih vrijednosti.

Strojno učenje i rudarenje podataka odnosi se na područje koje proučava metode za eksploataciju znanja i poboljšanje performansi inteligentnih agenata tijekom vremena. Ovo područje podrazumijeva čestu analizu okružja unutar kojega se agent nalazi te pripremu i izračun (predikciju) nepoznatnih vrijednosti iz varijabli okružja.

3.2.6. Evaluacija modela

Svrha same evaluacije je procijeniti valjanost i korektnost rezultata analize podataka. Naime, potrebno je egzaktno se uvjeriti kako su rezultati obrade ekstrahirani iz podataka zaista vrijedni i pokazuju očekivane zakonitosti. Obično je znatno lakše i bolje model testirati prvo u tzv. laboratorijskim uvjetima, a zatim ga, nakon svih provjera, označiti kao valjanog i dalje isporučivati u sljedeće faze. Drugo, ali jednakovrijedno prethodnom zahtjevu bilo bi evaluirati odnos dobivenih rezultata prema određenim realnim (poslovnim) ciljevima. Drugim riječima, rezultat kao takav bi trebao biti potpora odlučivanju bilo to donošenje odluka u poslovanju ili bilo kakvih drugih odluka. Model obrade nije sam sebi svrha. Sam rezultat koji je kvalitetan, vrlo je koristan i vrijedan resurs.

3.2.7. Prikaz rezultata

Engleski naziv ove faze je *deployment*, dakle, to je faza unutar koje se rezultati ne samo prikazuju nego i prosljeđuju na daljnje korištenje kao dovršeni output procesa obrade i analize podataka. Krajnji cilj rezultata je određena financijska korist ili korist druge prirode. Podatkovni model kao rezultat, odnosno kao individualna komponenta, mora, nakon obrade i analize, biti

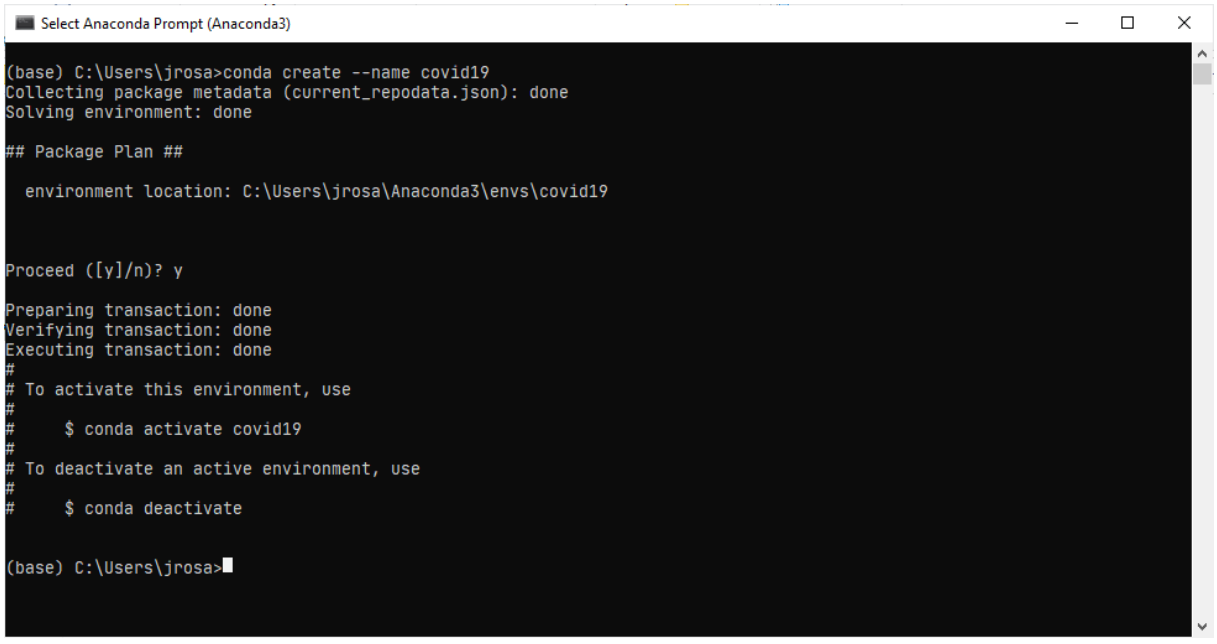
uklopljen natrag u određeni sustav unutar kojega bi donosio daljnju korist. Ponekad se taj model mora moći prilagoditi sustavu kao takvom iz nekog tehničkog ili netehničkog razloga. [9, str. 32]. Bez obzira na to je li i u kojoj mjeri korištenje modela u određene poslovne svrhe uredno proteklo, vrlo često se isti model može cirkularno vratiti natrag u prvu fazu - fazu razumijevanja poslovne logike. Sljedeća iteracija može producirati moguć bolji rezultat u odnosu na prethodni. Dani podaci su prethodno već analizirani, obrađeni, nad njima su se vršile analize i obrade te su skupovi kao takvi, u ovim fazama puno bolje razumijevani u odnosu na prethodne faze. Naravno, nije nužno cirkulirati aktivnostima prema CRISP metodici. Najbolji i najčešći način za indikaciju o vraćanju na početak nalazi se u fazi evaluacije. U toj fazi pokazuju se mane produciranog modela te se prema tome može djelovati. [9, str. 32].

4. Obrada otvorenog skupa podataka - praktični dio

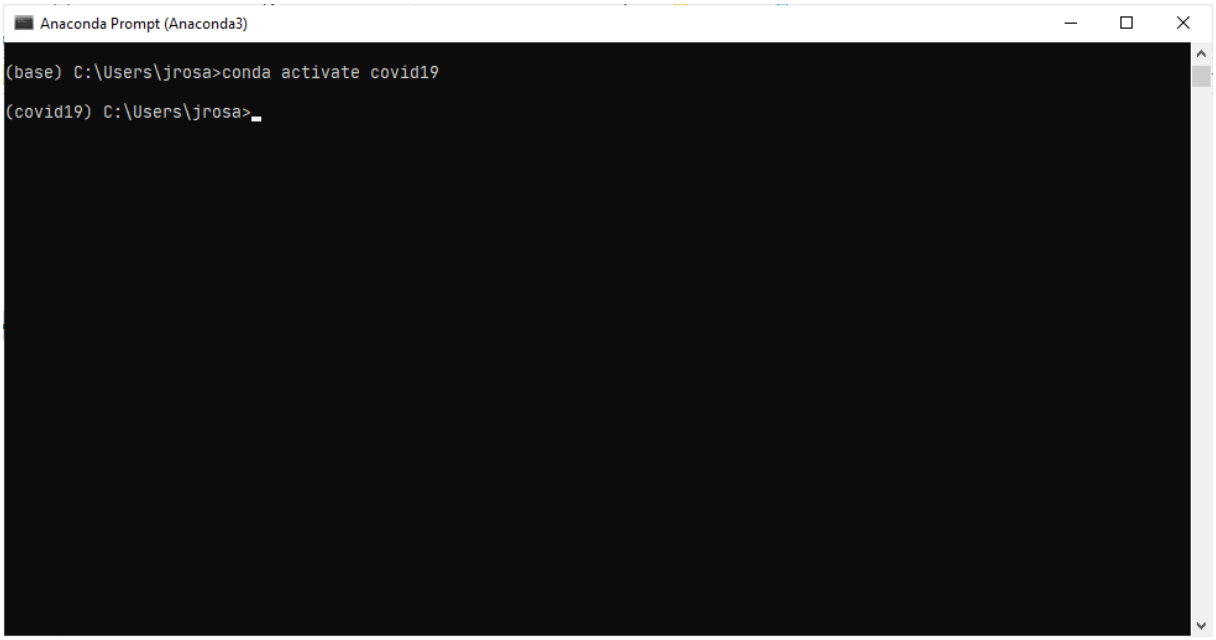
Nastavno na ranije razrađen teorijski opus, a što se odnosi na tehnologije obrade, alate te metode i metodologije, u ovom poglavlju će biti prikazan praktičan primjer analize otvorenog skupa podataka. No za početak će biti prikazan i teorijski analiziran te smješten u kontekst sam skup podataka. Nakon toga prikazat će se postavljanje okružja za analizu te obrada skupa podataka prolaskom svim fazama prema CRISP metodologiji. U prilogu na kraju ovoga rada priložen je izvadak cjelokupnog praktičnog rada izvezen iz Jupyter Notebook razvojnog okružja.

4.1. Postavljanje Jupyter Notebook okružja za rad

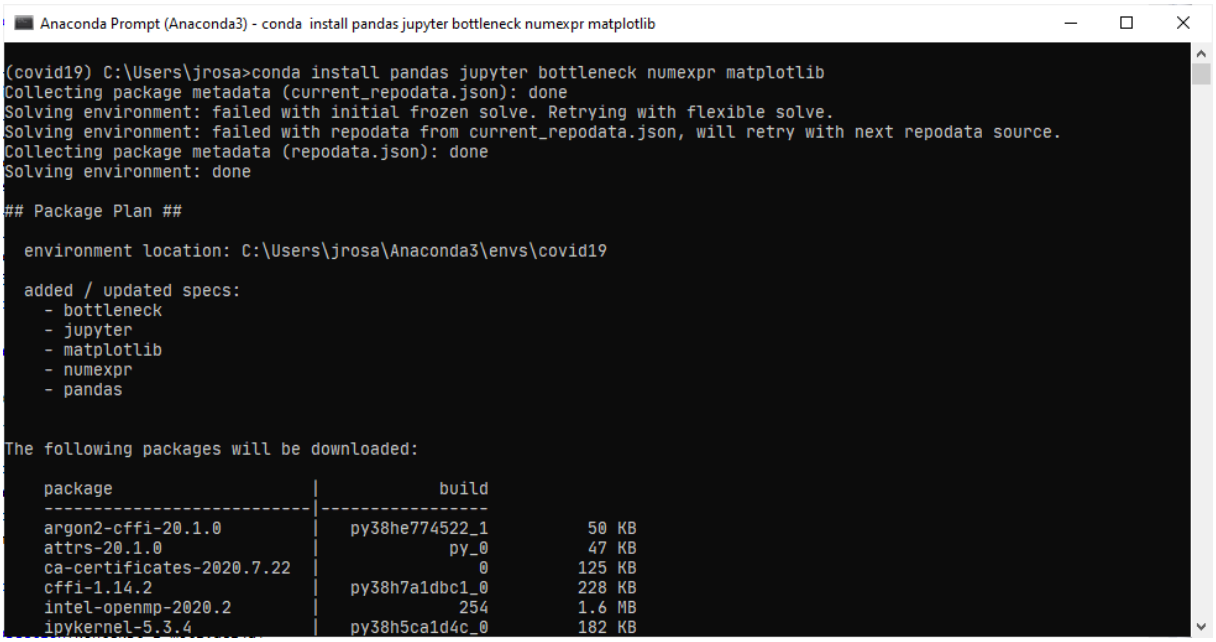
Nakon prethodne instalacije Anaconda distribucije, prvi korak je postavljanje okružja za rad na projektu što uključuje kreiranje virtualnog okružja unutar Anaconda distribucije. Virtualno okružje podrazumijeva direktorij koji sadrži specifične skupove conda biblioteka i paketa koje je korisnik instalirao za rad na projektu. [12] Primjerice, u ovome projektu, unutar conda virtualnog okružja instalirani su moduli prikazani na slici 14. Ukoliko želimo pokrenuti novi projekt za rad s nekim drugim modulima, to ćemo učiniti kreirajući novo conda virtualno okružje na sljedeći način:



Slika 12: Kreiranje novog virtualnog okružja unutar Anaconda komandne linije (Izvor: vlastita izrada)



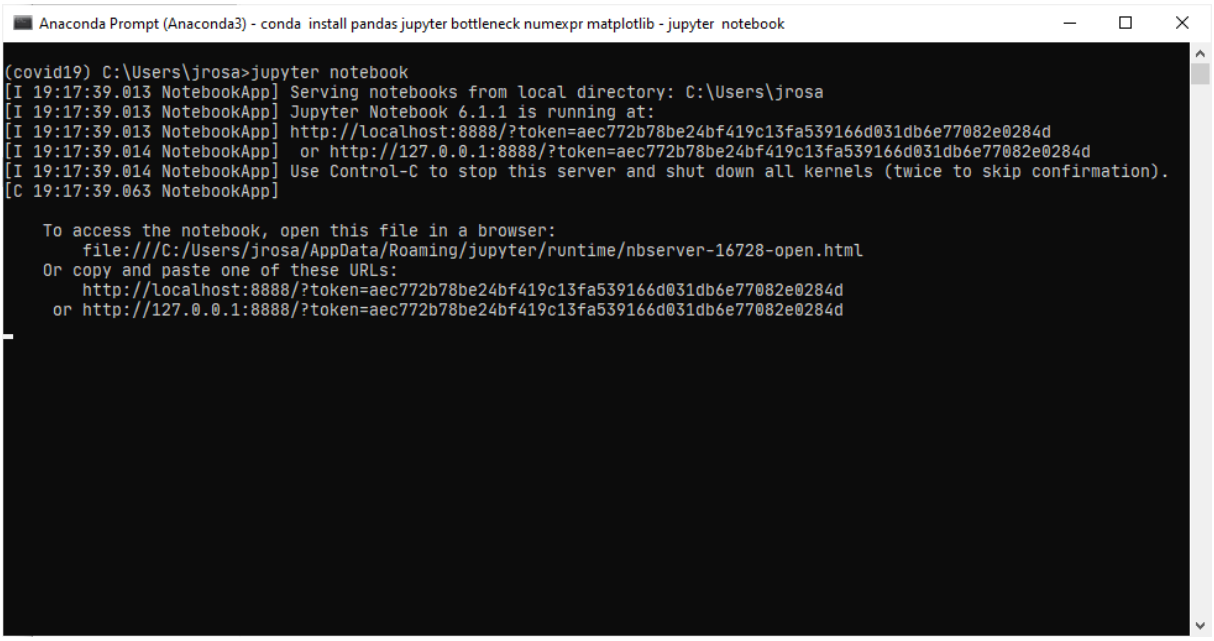
Slika 13: Aktivacija novokreiranog virtualnog okružja naziva `covid19` (Izvor: vlastita izrada)



Slika 14: Instalacija potrebnih biblioteka unutar virtualnog okružja (Izvor: vlastita izrada)

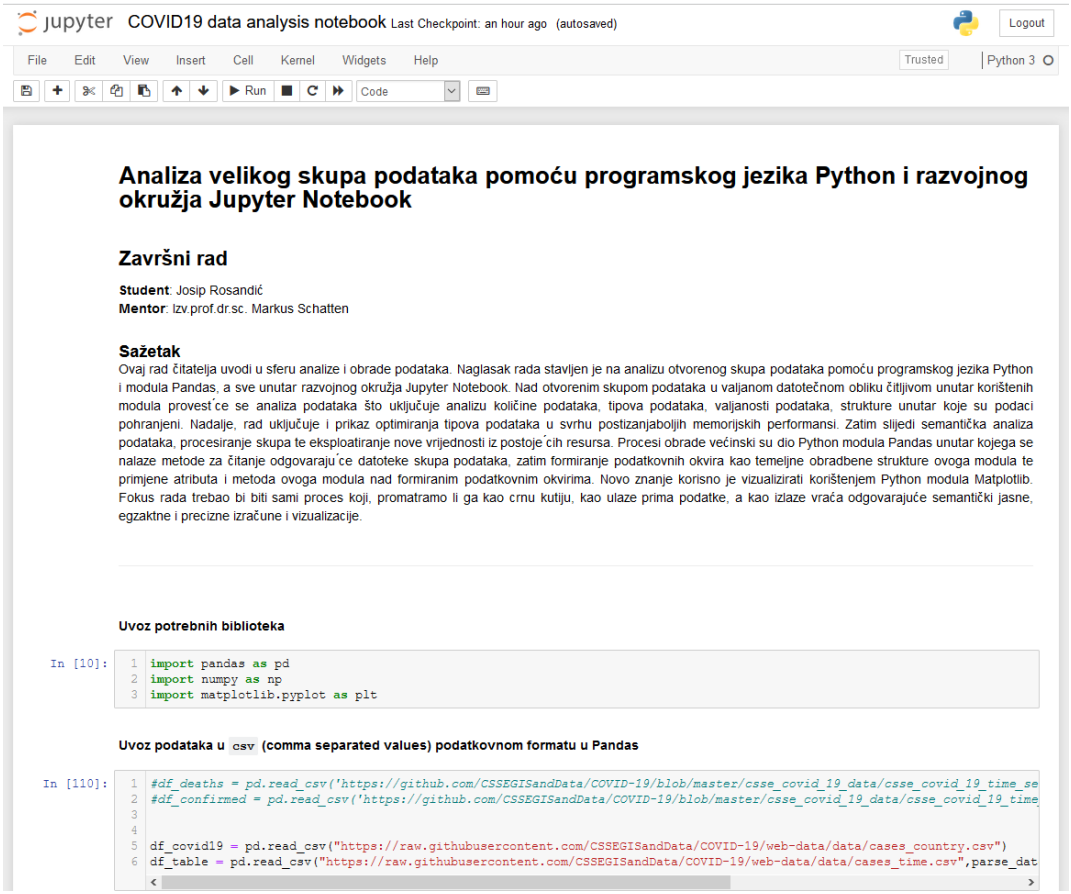
Instalirane su sljedeće biblioteke:

- `Pandas`
- `bottleneck`
- `numexpr`
- `matplotlib`



Slika 15: Pokretanje Jupyter Notebooka iz Anaconda komandne linije (Izvor: vlastita izrada)

Kao što vidimo na slici 15 iznad, komandom `jupyter notebook` pokrenuli smo lokalni server koji radi na dediciranom portu 8888. Dakle, notebook aplikaciji možemo pristupiti iz web preglednika na adresi `http://127.0.0.1:8888/?token=<token_sesije>`.



Slika 16: Jupyter Notebook korisničko sučelje unutar web preglednika (Izvor: vlastita izrada)

4.2. Skupovi podataka

U ovome dijelu rada bit će prikazana priroda skupova podataka korištenih u radu. Prvo će se skupovi podataka smjestiti u kontekst, zatim će se elaborirati izvor podataka te priroda skupa podataka.

4.2.1. Otvoreni podaci

Otvoreni podaci (eng. *open data*) su onakvi podaci ili skupovi podataka kojima svatko smije pristupiti, čitati ih i dalje dijeliti, ali uz pripadajuću licencu kojom su ti podaci licencirani. [13] Postoje tri osnovne komponente koje podatke čine otvorenima, a to su: [13]

1. **Ograničenja** - nad otvorenim podacima korisnik smije raditi što god želi, dakle, bez ograničenja
2. **Troškovi** - besplatni su za korištenje, no ponekad, zbog logističkih troškova, pristup podacima može se naplaćivati po simboličnim iznosima
3. **Ponovna iskoristivost** - korisnik može dalje dijeliti podatke

4.2.2. Kontekst i motivacija

Skupovi podataka koji su baza ovoga rada odnose se na statističke podatke o oboljelima od respiratorne bolesti **COVID** - **CO**rona**VI**rus **D**isease uzrokovane novim koronavirusom **SARS-cov-2**. Naime, ovaj virus i respiratnorna bolest koju isti uzrokuje, najaktualniji su koncepti o kojima stanovništvo planeta Zemlja promišlja u 2020. godini. Na tragu aktualnosti i gorućih svjetskih pitanja, ova tematika nametnula se kao nepresušan izvor podataka koji mogu poslužiti kao sirovina (*input*) za mnoge analitike, promatranja, istraživanja i sl. Globalna situacija glede ovoga pitanja svakodnevno se mijenja od samoga mjeseca siječnja na razini nekoliko sati, ponegdje i na razini kraćeg perioda. Mnoge svjetske organizacije sudjeluju u praćenju kretanja podatkovno utemeljenih pokazatelja po svim državama svijeta. Takav angažman svjetske zajednice učinio je pojavu generiranja podataka vezanih uz ovu temu vrlo izraženom pa slijedom toga mnoge svjetske institucije i organizacije prikupljaju i dijele otvorene podatke ili pak kao posrednik integriraju podatke iz više relevantnih izvora te na taj način skupove podataka vezane uz ovu temu čine široko dostupnima svjetskoj istraživačkoj zajednici.

4.2.3. Izvor podataka

Kao izvor skupa podataka, odabran je **COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University**, odnosno podatkovni repozitorij CSSE inženjersko-istraživačkog centra sveučilišta Johns Hopkins u Baltimoreu u saveznoj državi Maryland, Sjedinjene Američke Države (https://github.com/CSSEGISandData/COVID-19). Naime, Johns Hopkins sveučilište i CSSE istraživački centar jedna je od prvih institucija koja je početkom 2020. godine počela na ovaj način prikupljati statističke podatke o oboljelima od bolesti COVID19. Štoviše, Centar je na temelju prikupljenih podataka, kao web aplikaciju kreirao i kontrolnu ploču (eng. dashboard) koja prikazuje relevantne statističke pokazatelje o kojima se podaci najčešće i prikupljaju, a to su **ukupan broj osoba zaraženih koronavirusom**, **ukupan broj osoba umrlih od koronavirusa** (važno je napomenuti kako ovu stavku treba prihvatiti s posebnim oprezom budući da mortalitet uvelike ovisi o dobrom definiranju uzroka smrti osobe, što se pokazalo vrlo labilnim), **ukupan broj oporavljenih osoba** i sl.

Kao sekundarni izvor podataka, CSSE ne prikuplja podatke vlastoručno nego podatke povlači iz najrelevantnijih svjetskih izvora, potom ih integrira unutar jedinstvenog skupa podataka i pruža kao resurs. Centar podatke prikuplja izravno iz zaista velikog broja izvora, a neki od izvora su:

- **Svjetska zdravstvena organizacije (WHO)** https://www.who.int/
- **Europski centar za kontrolu i prevenciju bolesti (ECDC)** https://www.ecdc.europa.eu/en/geographical-distribution-2019-ncov-cases
- **Centar za kontrolu i prevenciju bolesti Sjedinjenih Američkih Država (CDC)** https://www.cdc.gov/coronavirus/2019-ncov/index.html

a neki od izvora su projekti ili inicijative nastali isključivo i ciljano radi bavljenja ovom problematikom, primjerice:

- **COVID tracking project** - volonterski projekt u SAD-u https://covidtracking.com/data

Izvora iz kojih CSSE integrira podatke ima zaista veliki broj, a ostali izvori poglavito su državne zdravstvene institucije diljem svijeta, kako razni zdravstveni odjeli tako i ministarstva zdravstva država svijeta.

Daljnju analizu strukture podataka skupa podataka radit ćemo unutar Jupyter okružja. Prvi korak bit će uvoz skupa podataka u Jupyter okružje. Podatke ćemo uvesti pomoću apsolutne putanje do `.csv` datoteke s resursima na GitHub lokaciji CSSE Centra:

1. https://github.com/CSSEGISandData/COVID-19/blob/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv
2. https://github.com/CSSEGISandData/COVID-19/blob/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv
3. https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-data/data/cases_country.csv
4. https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-data/data/cases_time.csv

4.3. Analiza i obrada skupa podataka 1

Napomena: Budući da se radi o jedinstvenoj problemskoj domeni unutar koje se nalaze sva četiri skupa podataka, a isto tako budući da je problemska domena prethodno opisana i analizirana u ranijim poglavljima, u ovom dijelu sama domena se neće opisivati, no vrlo je važno poznavati osnovne metapodatke o domeni i problemu o kojemu se podaci prikupljaju.

4.3.1. Priprema skupa podataka

Prvi skup podataka uvezen je u Pandas data frame iz `csv` datoteke u varijablu `df_latest_data` u koju je pohranjen. Ovaj skup podataka sadrži najnovije informacije praćene po indeksu od **188 država** i atributima kao što su `Last_Update`, `Confirmed`, `Deaths`, `Active` i ostalim atributima ovoga data framea.

	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	Incident_Rate	People_Tested	People_Hospitalized	Mortality
0	Afghanistan	2020-09-06 13:28:19	33.939110	67.709953	38398.0	1412.0	30537.0	6449.0	98.637648	NaN	NaN	3.67
1	Albania	2020-09-06 13:28:19	41.153300	20.168300	10102.0	312.0	5976.0	3814.0	351.032038	NaN	NaN	3.08
2	Algeria	2020-09-06 13:28:19	28.033900	1.659600	46071.0	1549.0	32481.0	12041.0	105.062495	NaN	NaN	3.36
3	Andorra	2020-09-06 13:28:19	42.506300	1.521800	1215.0	53.0	928.0	234.0	1572.510192	NaN	NaN	4.36
4	Angola	2020-09-06 13:28:19	-11.202700	17.873900	2935.0	117.0	1192.0	1626.0	8.930129	NaN	NaN	3.98
...	...	...	...	...	...	...	...	...	...	...	...	
183	West Bank and Gaza	2020-09-06 13:28:19	31.952200	35.233200	26127.0	181.0	16843.0	9103.0	512.151920	NaN	NaN	0.69
184	Western Sahara	2020-09-06 13:28:19	24.215500	-12.885800	10.0	1.0	8.0	1.0	1.674116	NaN	NaN	10.00
185	Yemen	2020-09-06 13:28:19	15.552727	48.516388	1983.0	572.0	1197.0	214.0	6.648569	NaN	NaN	28.84
186	Zambia	2020-09-06 13:28:19	-13.133897	27.849332	12709.0	292.0	11668.0	749.0	69.130931	NaN	NaN	2.29
187	Zimbabwe	2020-09-06 13:28:19	-19.015438	29.154857	6837.0	206.0	5345.0	1286.0	46.000360	NaN	NaN	3.01

Slika 17: Prikaz data framea kreiranog iz prvog skupa podataka (Izvor: vlastita izrada)

Koristeći Pandas data frame atribut `shape` dobit ćemo uređeni par **(188,14)** koji predstavlja broj redaka i stupaca data framea. Nadalje, koristeći metodu `info()` dobit ćemo ispis sa sljedećim informacijama:

```
# <class 'Pandas.core.frame.DataFrame'>
RangeIndex: 188 entries, 0 to 187
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Country_Region       188 non-null    object
 1   Last_Update          188 non-null    object
 2   Lat                  186 non-null    float64
 3   Long_                186 non-null    float64
 4   Confirmed            188 non-null    float64
 5   Deaths               188 non-null    float64
 6   Recovered            185 non-null    float64
 7   Active               188 non-null    float64
 8   Incident_Rate        186 non-null    float64
 9   People_Tested        0 non-null      float64
 10  People_Hospitalized  0 non-null      float64
 11  Mortality_Rate       188 non-null    float64
 12  UID                  188 non-null    int64
 13  ISO3                 186 non-null    object
dtypes: float64(10), int64(1), object(3)
memory usage: 20.7+ KB
```

Pomoću metode `info()` možemo analizirati koje sve stupce sadrži data frame te koje tipove podataka sadrži pojedini stupac. Isto tako, možeo vidjeti koliki memorijski kapacitet data frame zauzima. U ovom slučaju, radi se o **20.7+ KB**.

Vidimo kako su tipovi podataka ne bitno različiti. Dakle, prisutni su float64, int64 te object. Radi se o tomu da Pandas prilikom obrade `.csv` datoteke sve cjelobrojne numeričke podatke tipa integer (`int64`) pretvara u decimalni format tipa `float64`. Nadalje, `object` nije vrlo poželjan tip podatka i dobro je pretvoriti ga u specifičan tip podataka gdje god je to moguće.

Unutar Pandas biblioteke postoji metoda `infer_objects().dtypes` koja sve tipove podataka pretvara u one tipove koji su prikladni za pojedini skup.

Metoda `memory_usage()` vraća memorijsku veličinu svakog stupca u bajtovima

```
df_latest_data.memory_usage()/1000000 # dijeljenje s 1e6 kako bi rezultat
 ↪ bio u MB

# Ispis po stupcima
Index                0.000128
Country_Region       0.001504
Last_Update          0.001504
```

```
Lat                    0.001504
Long_                  0.001504
Confirmed              0.001504
Deaths                 0.001504
Recovered              0.001504
Active                 0.001504
Incident_Rate          0.001504
People_Tested          0.001504
People_Hospitalized    0.001504
Mortality_Rate         0.001504
UID                    0.001504
ISO3                   0.001504
dtype: float64
```

Pomoću metode `drop(columns=[])` izbacili smo irelevantne stupce iz data framea te smo tako značajno smanjili veličinu data framea.

```
df_latest_data.drop(columns=["Lat","Long_","Incident_Rate","People_Tested",
"People_Hospitalized","ISO3","UID","Mortality_Rate"],inplace=True)
```

Nadalje, provedena je pretvorba podataka u eksplicitne tipove:

```
# Parsiranje datuma
df_latest_data["Last_Update"] =
 ↪  pd.to_datetime(df_latest_data["Last_Update"])

# Pretvorba u int64
df_latest_data["Confirmed"] = df_latest_data["Confirmed"].astype("int64")
df_latest_data["Deaths"] = df_latest_data["Deaths"].astype("int64")
df_latest_data["Active"] = df_latest_data["Active"].astype("int64")
df_latest_data["Recovered"].fillna(0,inplace = True) # Zamjena nepoznatih
 ↪  vrijednosti eksplicitnim vrijednostima
df_latest_data["Recovered"] = df_latest_data["Recovered"].astype("int64")
```

Sada ispis metode `info()` izgleda ovako:

```
# <class 'Pandas.core.frame.DataFrame'>
Index: 188 entries, Afghanistan to Zimbabwe
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Last_Update  188 non-null    datetime64[ns]
 1   Confirmed    188 non-null    int64
 2   Deaths       188 non-null    int64
```

```
 3   Recovered    188 non-null    int64
 4   Active       188 non-null    int64
dtypes: datetime64[ns](1), int64(4)
memory usage: 8.8+ KB
```

4.3.2. Modeliranje podataka

Prikažimo najnovije podatke za **Republiku Hrvatsku:**

```
cro_newest = pd.DataFrame(df_latest_data.loc["Croatia"]).T # .T provodi
 ↪ transponiranje data framea
cro_newest
```

	Last_Update	Confirmed	Deaths	Recovered	Active	Mortality Rate (per 100)
Croatia	2020-09-06 13:28:19	11964	198	9008	2758	1.65

Slika 18: Ekstrakcija najnovijih podataka za RH u data frame (Izvor: vlastita izrada)

Pomoću metode `head()`, prikažimo prvih nekoliko redova skupa:

```
df_latest_data.head(10)
```

Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active	Mortality Rate (per 100)
Afghanistan	2020-09-06 13:28:19	38398	1412	30537	6449	3.68
Albania	2020-09-06 13:28:19	10102	312	5976	3814	3.09
Algeria	2020-09-06 13:28:19	46071	1549	32481	12041	3.36
Andorra	2020-09-06 13:28:19	1215	53	928	234	4.36
Angola	2020-09-06 13:28:19	2935	117	1192	1626	3.99
Antigua and Barbuda	2020-09-06 13:28:19	95	3	91	1	3.16
Argentina	2020-09-06 13:28:19	471806	9739	340381	121686	2.06
Armenia	2020-09-06 13:28:19	44783	897	40089	3797	2.00
Australia	2020-09-06 13:28:19	26279	753	22465	3061	2.87
Austria	2020-09-06 13:28:19	29271	736	25043	3492	2.51

Slika 19: Prikaz prvih 10 redova skupa pomoću metode head() (Izvor: vlastita izrada)

U nastavku ćemo nad skupom ćemo neke od statističkih metoda Pandas biblioteke:

- `sum()`
- `count()`

- median()
- min()
- max()

```
df_latest_data["Confirmed"].sum()
26918965

df_latest_data["Confirmed"].count()
188

df_latest_data["Confirmed"].median()
9077.0
```

Ispis države s najmanjim brojem potvrđenih slučajeva zaraze:

```
df_latest_data["Confirmed"].min()
df_latest_data[df_latest_data["Confirmed"] ==
 ↪ df_latest_data["Confirmed"].min()]
```

Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active
MS Zaandam	2020-09-06 13:28:19	9	2	0	7

Slika 20: Ispis države s najmanjim brojem potvrđenih slučajeva zaraze (Izvor: vlastita izrada)

Ispis države s najvećim brojem potvrđenih slučajeva zaraze:

```
df_latest_data["Confirmed"].max()
df_latest_data[df_latest_data["Confirmed"] ==
 ↪ df_latest_data["Confirmed"].max()]
```

Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active
US	2020-09-06 13:28:19	6246281	188554	2302187	3755539

Slika 21: Ispis države s najvećim brojem potvrđenih slučajeva zaraze (Izvor: vlastita izrada)

Ispis medijalne vrijednosti broja potvrđenih slučajeva po svim promatranim državama:

```
df_latest_data["Confirmed"].median()

9077.0
```

Pozovimo metodu `.describe()` koja će nam za svaki stupac prikazati osnovne metrike deskriptivne statistike:

```
df_latest_data.describe().astype("float").round(2)
```

	Confirmed	Deaths	Recovered	Active
count	188.00	188.00	188.00	188.00
mean	143185.98	4682.66	95622.53	42880.78
std	628318.95	18655.83	392275.11	284548.83
min	9.00	0.00	0.00	0.00
25%	1997.25	34.50	1119.75	299.25
50%	9077.00	184.00	5845.00	1779.00
75%	61045.25	1060.75	41161.50	9972.00
max	6246281.00	188554.00	3498999.00	3755539.00

Slika 22: Ispis osnovnih metrika deskriptivne statistike (Izvor: vlastita izrada)

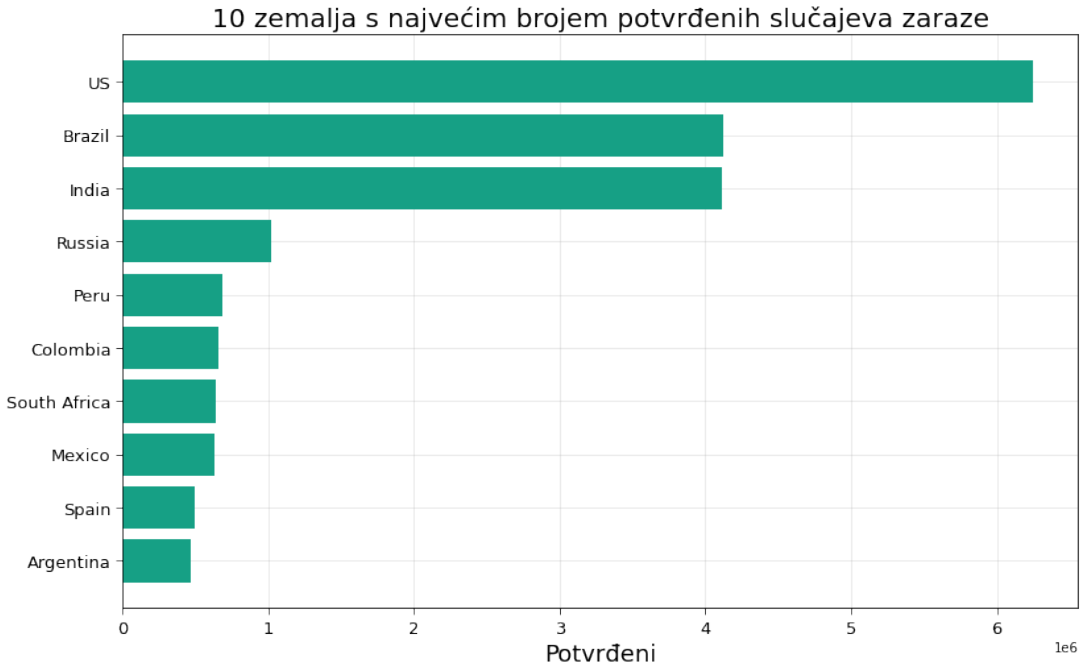
Prikaz još nekoliko rezultata obrade skupa 1 nalazi se u `.ipynb` rilogu na kraju rada.

4.3.3. Prikaz rezultata

Rezultate obrade prikazat ćemo grafički. U nastavku slijedi prikaz nekoliko grafičkih vizualizacija.

GRAF 1.1 - 10 zemalja s najvećim brojem potvrđenih slučajeva zaraze

```
f = plt.figure(figsize=(13,8))
f.add_subplot(1,1,1)

plt.axes(axisbelow=True) #grid osi ispod grafikona
plt.barh(df_latest_data.sort_values('Confirmed')["Confirmed"].index[-10:],df_latest_data
plt.tick_params(size=5,labelsize = 13) #veličina labela kontrolnih točaka
plt.xlabel("Potvrđeni",fontsize=18)
plt.title("10 zemalja s najvećim brojem potvrđenih slučajeva
 ↪  zaraze",fontsize=20)
plt.grid(alpha=0.3,which='both')
```

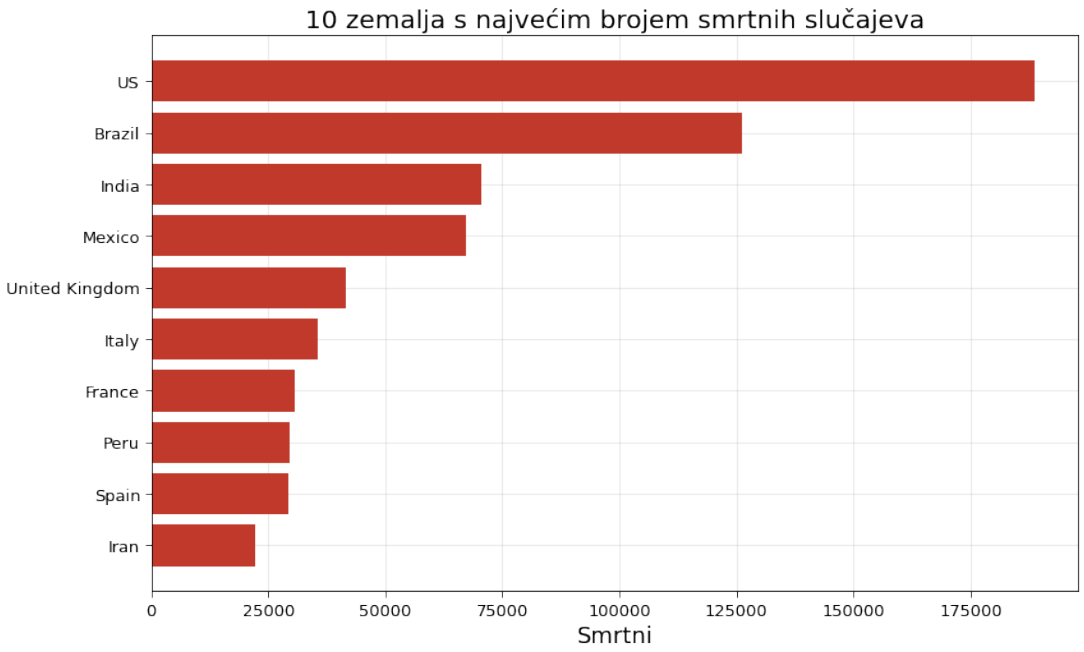


Slika 23: Grafička vizualizacija koja prikazuje 10 država s najvećim brojem potvrđenih slučajeva (Izvor: vlastita izrada)

Naime, koristeći biblioteku `matplotlib`, rezultat smo prikazali horizontalnim stupčastim grafikonom konstruiranim pomoću matplotlib metode `plt.barh()` s odgovarajućim parametrima. Kreirali smo tzv. *figure* kao područje unutar kojega prikazujemo grafikone. Zatim smo prikazali sami grafikon te mu pridružili odgovarajuće labele.

GRAF 1.2 - 10 zemalja s najvećim brojem smrtnih slučajeva

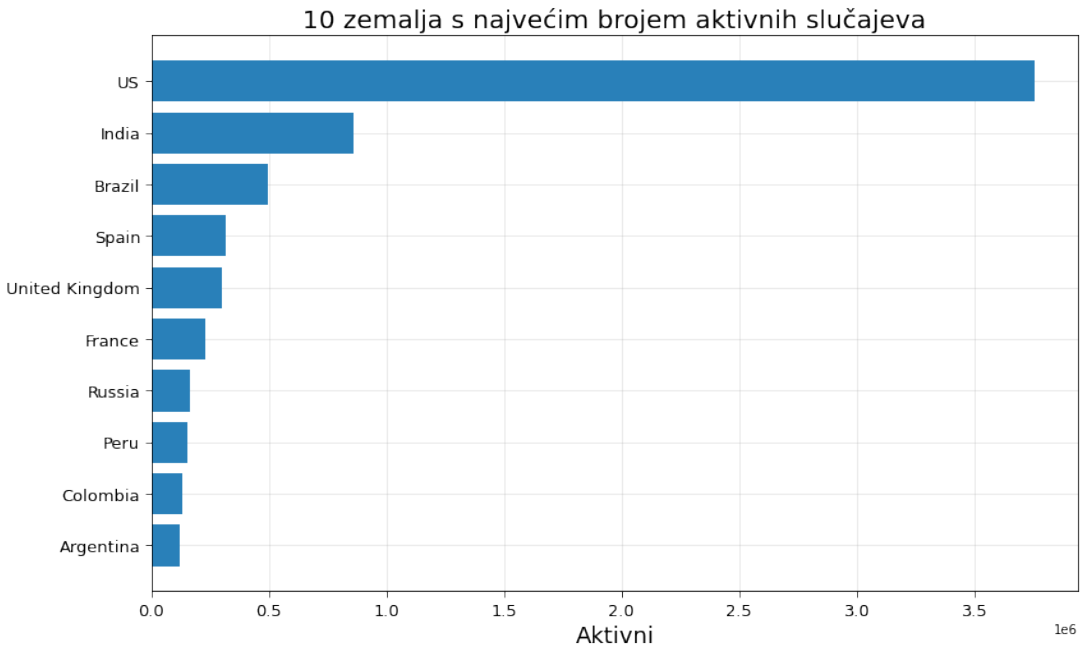
```
f = plt.figure(figsize=(13,8))
f.add_subplot(1,1,1)

plt.axes(axisbelow=True)
plt.barh(df_latest_data.sort_values('Deaths')["Deaths"].index[-10:],df_latest_data.sort_
plt.tick_params(size=5,labelsize = 13)
plt.xlabel("Smrtni",fontsize=18)
plt.title("10 zemalja s najvećim brojem smrtnih slučajeva",fontsize=20)
plt.grid(alpha=0.3,which='both')
```



Slika 24: Grafička vizualizacija koja prikazuje 10 država s najvećim brojem smrtnih slučajeva (Izvor: vlastita izrada)

GRAF 1.3 - 10 zemalja s najvećim brojem aktivnih slučajeva

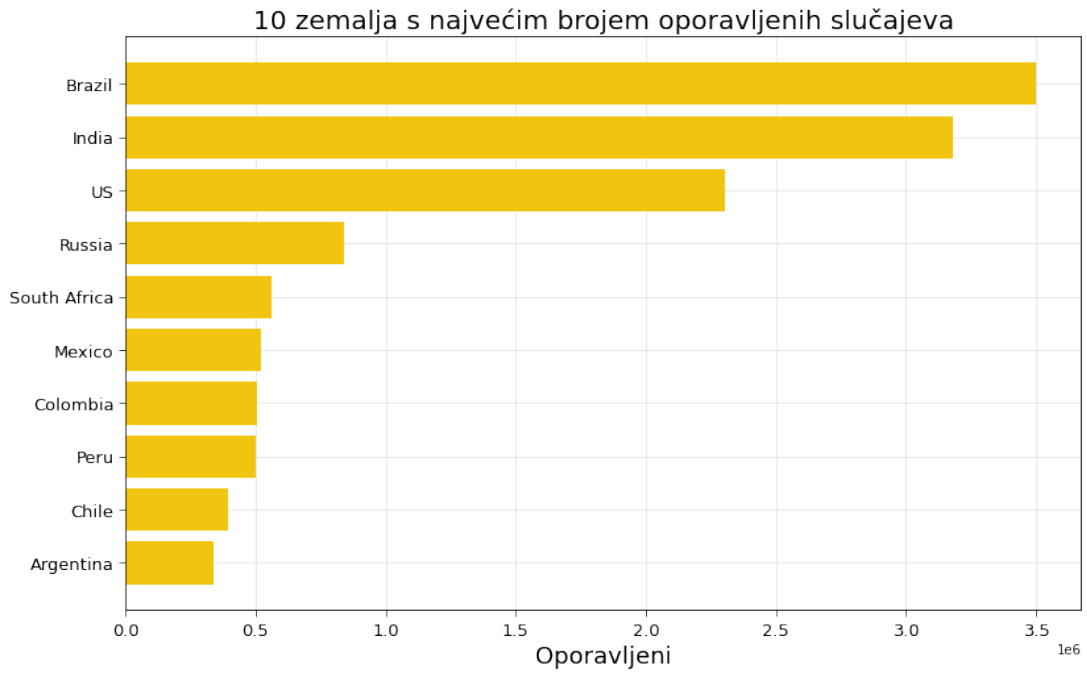
```
f = plt.figure(figsize=(13,8))
f.add_subplot(1,1,1)

plt.axes(axisbelow=True)
plt.barh(df_latest_data.sort_values('Active')["Active"].index[-10:],df_latest_data.sort_
plt.tick_params(size=5,labelsize = 13)
plt.xlabel("Aktivni",fontsize=18)
plt.title("10 zemalja s najvećim brojem aktivnih slučajeva",fontsize=20)
plt.grid(alpha=0.3,which='both')
```



Slika 25: Grafička vizualizacija koja prikazuje 10 država s najvećim brojem aktivnih slučajeva (Izvor: vlastita izrada)

GRAF 1.4 - 10 zemalja s najvećim brojem oporavljenih slučajeva

```
f = plt.figure(figsize=(13,8))
f.add_subplot(1,1,1)

plt.axes(axisbelow=True)
plt.barh(df_latest_data.sort_values('Recovered')["Recovered"].index[-10:],df_latest_data
plt.tick_params(size=5,labelsize = 13)
plt.xlabel("Oporavljeni",fontsize=18)
plt.title("10 zemalja s najvećim brojem oporavljenih
 ↪  slučajeva",fontsize=20)
plt.grid(alpha=0.3,which='both')
```



Slika 26: Grafička vizualizacija koja prikazuje 10 država s najvećim brojem oporavljenih slučajeva (Izvor: vlastita izrada)

Prikaz još nekoliko rezultata obrade skupa 1 nalazi se u `.ipynb` prilogu na kraju rada.

4.4. Analiza i obrada skupa podataka 2

Napomena: Budući da se radi o jedinstvenoj problemskoj domeni unutar koje se nalaze sva četiri skupa podataka, a isto tako budući da je problemska domena prethodno opisana i analizirana u ranijim poglavljima, u ovom dijelu sama domena se neće opisivati, no vrlo je važno poznavati osnovne metapodatke o domeni i problemu o kojemu se podaci prikupljaju.

4.4.1. Priprema skupa podataka

Drugi skup podataka uvezen je u Pandas data frame iz `csv` datoteke u varijablu `df_countries_dates` u koju je pohranjen. Ovaj skup sadrži podatke o vrijednostima atributa **po svim promatranim državama od početka mjerenja**, dakle od 22. siječnja. Radi se o atributima `Last_Update`, `Confirmed` i `Deaths` te ostalim atributima čije vrijednosti nećemo promatrati iz razloga jer ne postoje ili su nevažeće vrijednosti.

	Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active	Delta_Confirmed	Delta_Recovered	Incident_Rate	People_Tested	People_Hospitalize
0	Afghanistan	1/22/20	0	0	NaN	NaN	0.0	NaN	0.000000	NaN	Na
1	Afghanistan	1/23/20	0	0	NaN	NaN	0.0	NaN	0.000000	NaN	Na
2	Afghanistan	1/24/20	0	0	NaN	NaN	0.0	NaN	0.000000	NaN	Na
3	Afghanistan	1/25/20	0	0	NaN	NaN	0.0	NaN	0.000000	NaN	Na
4	Afghanistan	1/26/20	0	0	NaN	NaN	0.0	NaN	0.000000	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	
56083	US	9/1/20	3874	37	NaN	NaN	24.0	NaN	55926.086329	NaN	Na
56084	US	9/2/20	3911	41	NaN	NaN	37.0	NaN	56460.228093	NaN	Na
56085	US	9/3/20	3941	41	NaN	NaN	30.0	NaN	56893.316010	NaN	Na
56086	US	9/4/20	3990	42	NaN	NaN	49.0	NaN	57600.692941	NaN	Na
56087	US	9/5/20	3990	42	NaN	NaN	0.0	NaN	57600.692941	NaN	Na

Slika 27: Prikaz data framea kreiranog iz drugog skupa podataka (Izvor: vlastita izrada)

Koristeći Pandas data frame atribut `shape` dobit ćemo uređeni par **(56088,17)** koji predstavlja broj redaka i stupaca data framea. **Napomena:** ovaj data frame ima varijabilan broj redaka jer se svakodnevno nekoliko puta ažurira najnovijim podacima. Nadalje, pomoću metode `.info()` dobit ćemo ispis sa sljedećim informacijama:

```
#<class 'Pandas.core.frame.DataFrame'>
RangeIndex: 56088 entries, 0 to 56087
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Country_Region       56088 non-null  object
 1   Last_Update          56088 non-null  object
 2   Confirmed            56088 non-null  int64
 3   Deaths               56088 non-null  int64
 4   Recovered            0 non-null      float64
 5   Active               0 non-null      float64
 6   Delta_Confirmed      56005 non-null  float64
 7   Delta_Recovered      0 non-null      float64
 8   Incident_Rate        54948 non-null  float64
```

```
 9   People_Tested        0 non-null      float64
 10  People_Hospitalized  0 non-null      float64
 11  Province_State       13224 non-null  object
 12  FIPS                 13224 non-null  float64
 13  UID                  56088 non-null  int64
 14  iso3                 55632 non-null  object
 15  Report_Date_String   56088 non-null  object
 16  Delta_Deaths         55971 non-null  float64
dtypes: float64(9), int64(3), object(5)
memory usage: 7.3+ MB
```

Naime, vidimo kako je zauzetost memorije koju stvara ovaj data frame znatno veća u odnosu na prethodni iz razloga što ovaj data frame ima znatno veći broj zapisa. Tip podatka koji se najčešće pojavljuje je `float64`, a tip `object` poželjno je, ukoliko je to moguće, pretvoriti u eksplicitan tip podatka.

U nastavku je prikazano brisanje stupaca data framea i pretvaranje tipa podatka. Pogledamo li ispis nakon izbacivanja stupaca, možemo jasno vidjeti kako se zauzetost memorije koju stvara ovaj data frame znatno smanjila.

```
df_countries_dates.drop(columns=["Delta_Confirmed","Delta_Recovered","Incident_Rate",
"People_Tested","People_Hospitalized","iso3","UID","Province_State","FIPS",
"Report_Date_String","Delta_Deaths","Active","Recovered"],inplace=True)

df_countries_dates["Last_Update"] =
 ↪ pd.to_datetime(df_countries_dates["Last_Update"])

df_countries_dates.info()

#<class 'Pandas.core.frame.DataFrame'>
RangeIndex: 56088 entries, 0 to 56087
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Country_Region  56088 non-null  object
 1   Last_Update     56088 non-null  datetime64[ns]
 2   Confirmed       56088 non-null  int64
 3   Deaths          56088 non-null  int64
dtypes: datetime64[ns](1), int64(2), object(1)
memory usage: 1.7+ MB
```

4.4.2. Modeliranje podataka

U nastavku slijedi prikaz modeliranja podataka kao temelj za grafičke vizualizacije. Naime, vizualizacija koja će biti prikazana paralelno prikazuje grafikone kretanja broja potvrđenih slučajeva u državama regije. Za početak, potrebno je izvući podskupe, dakle data frameove koji sadrže podatke o državi za svaki dan od početka mjerenja. Prikaz slijedi u nastavku:

```
# Podaci o potvrđenim slučajevima u Hrvatskoj počevši s 22. siječnja
df_croatia = df_countries_dates[df_countries_dates["Country_Region"] ==
 ↪  "Croatia"]
df_croatia

# Podaci o potvrđenim slučajevima u Srbiji počevši s 22. siječnja
df_srb = df_countries_dates[df_countries_dates["Country_Region"] ==
 ↪  "Serbia"]
df_srb

# Podaci o potvrđenim slučajevima u Srbiji počevši s 22. siječnja
df_srb = df_countries_dates[df_countries_dates["Country_Region"] ==
 ↪  "Serbia"]
df_srb

# Podaci o potvrđenim slučajevima u Crnoj Gori počevši s 22. siječnja
df_mne = df_countries_dates[df_countries_dates["Country_Region"] ==
 ↪  "Montenegro"]
df_mne

# Podaci o potvrđenim slučajevima u Mađarskoj počevši s 22. siječnja
df_hun = df_countries_dates[df_countries_dates["Country_Region"] ==
 ↪  "Hungary"]
df_hun

# Podaci o potvrđenim slučajevima u Sloveniji počevši s 22. siječnja
df_slo = df_countries_dates[df_countries_dates["Country_Region"] ==
 ↪  "Slovenia"]
df_slo
```

Kako za Republiku Hrvatsku, tako su i metodama prikazanima u programskom kodu iznad ekstrahirani podaci za ostale države regije te su isti podaci prikazani na grafikonima **unutar figure 2.1.** Također, isti podaci korišteni su unutar figure 2.2. za grafičku vizualizaciju stope smrtnosti država regije. Naime, podatak o broju smrtnih slučajeva pomnožen sa sto, podijeljen je brojem potvrđenih slučajeva te je na taj način dobiven podatak o stopi smrtnosti **vizualiziran na figuri 2.2.**

	Country_Region	Last_Update	Confirmed	Deaths
9804	Croatia	2020-01-22	0	0
9805	Croatia	2020-01-23	0	0
9806	Croatia	2020-01-24	0	0
9807	Croatia	2020-01-25	0	0
9808	Croatia	2020-01-26	0	0
...	...	...	...	...
10027	Croatia	2020-09-01	10414	187
10028	Croatia	2020-09-02	10725	191
10029	Croatia	2020-09-03	11094	194
10030	Croatia	2020-09-04	11428	195
10031	Croatia	2020-09-05	11739	197

Slika 28: Data frame koji prikazuje podatke za Republiku Hrvatsku od početka mjerenja do najnovijih podataka (Izvor: vlastita izrada)

4.4.3. Prikaz rezultata

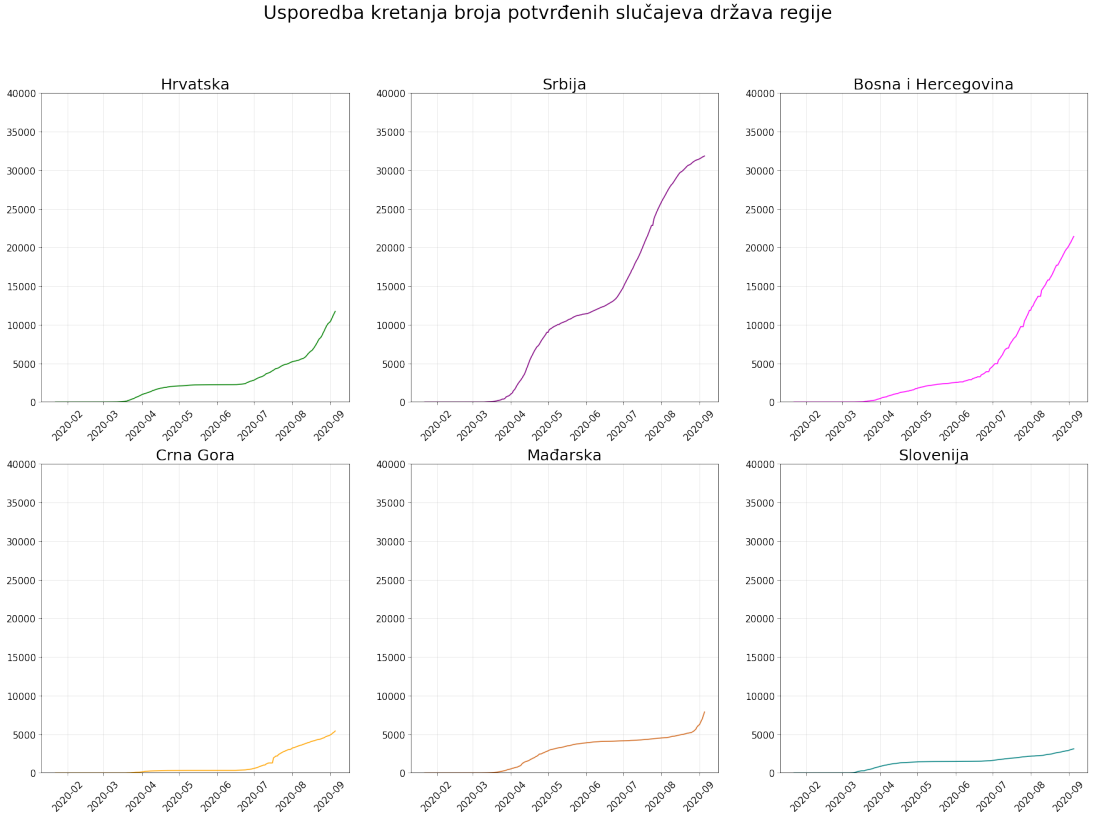
Rezultati gore opisanih ekstrakcija podataka prikazani su grafikonima u nastavku.

GRAF 2.1 - usporedba kretanja broja potvrđenih slučajeva država regije

```
# Kreiranje grafikona
fig = plt.figure(figsize=(30,20))

fig.suptitle('Usporedba kretanja broja potvrđenih slučajeva država regije',
             fontsize=30)

# Podgraf 1
ax1 = fig.add_subplot(231)
ax1.set_title('Hrvatska',fontsize=25)
ax1.set_ylim(0,40000)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
```

```
plt.grid(alpha=0.3,which='both')
ax1.plot(df_croatia['Last_Update'],
         df_croatia["Confirmed"],
         color='green')

# Podgraf 2
ax2 = fig.add_subplot(232)
ax2.set_title('Srbija',fontsize=25)
ax2.set_ylim(0,40000)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax2.plot(df_srb['Last_Update'],
         df_srb['Confirmed'],
         color='purple')

# Podgraf 3
ax3 = fig.add_subplot(233)
ax3.set_title('Bosna i Hercegovina',fontsize=25)
ax3.set_ylim(0,40000)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax3.plot(df_bih['Last_Update'],
         df_bih['Confirmed'],
         color='magenta')

# Podgraf 4
ax4 = fig.add_subplot(234)
ax4.set_title('Crna Gora',fontsize=25)
ax4.set_ylim(0,40000)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax4.plot(df_mne['Last_Update'],
         df_mne['Confirmed'],
         color='orange')

# Podgraf 5
ax5 = fig.add_subplot(235)
ax5.set_title('Mađarska',fontsize=25)
ax5.set_ylim(0,40000)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
```

```
63  ax5.plot(df_hun["Last_Update"],
64           df_hun["Confirmed"],
65           color='chocolate')
66
67  # Podgraf 6
68  ax6 = fig.add_subplot(236)
69  ax6.set_title('Slovenija',fontsize=25)
70  ax6.set_ylim(0,40000)
71
72  plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
73  plt.tick_params(size=5,labelsize = 15,axis='y')
74  plt.grid(alpha=0.3,which='both')
75  ax6.plot(df_slo["Last_Update"],
76           df_slo["Confirmed"],
77           color='teal')
78
79
80  plt.show()
```

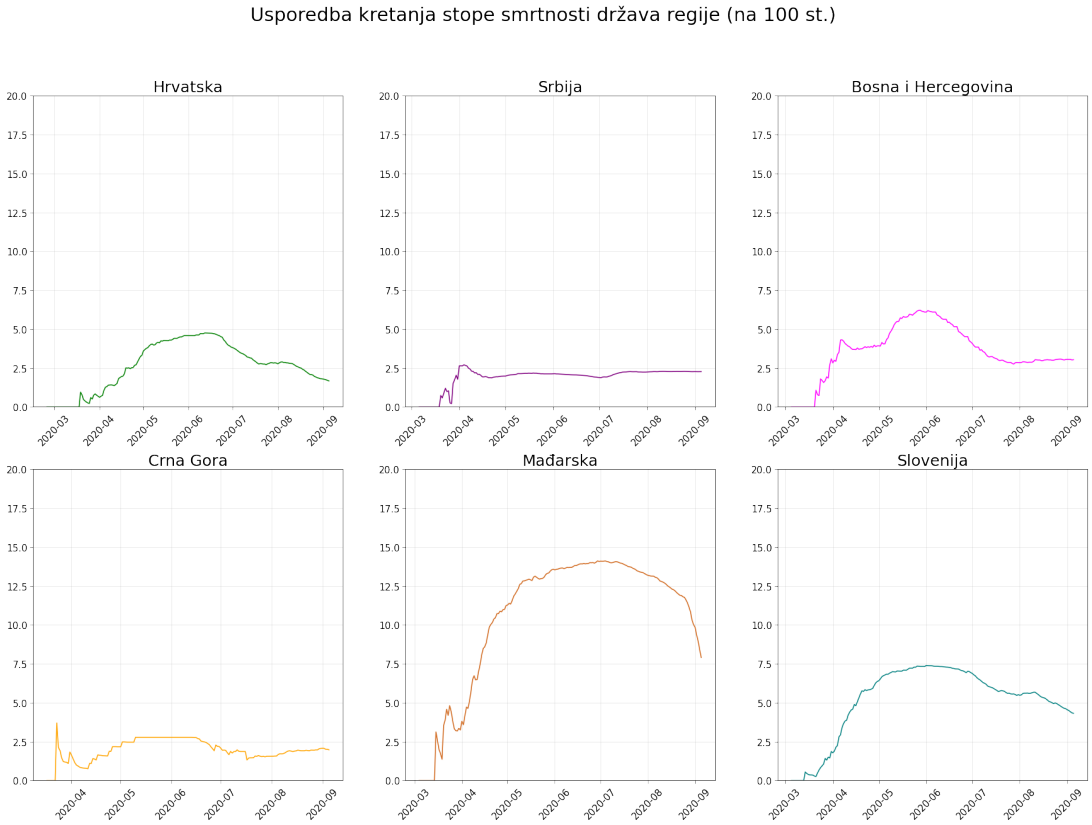


Slika 29: Figura s grafikonima koji usporedno prikazuju kretanje broja potvrđenih slučajeva država regije (Izvor: vlastita izrada)

GRAF 2.2 - usporedba kretanja stope smrtnosti država regije (na 100 st.)

```
# Kreiranje grafikona
fig = plt.figure(figsize=(30,20))

fig.suptitle('Usporedba kretanja stope smrtnosti država regije (na 100
 ↪  st.)',
             fontsize=30)

# Podgraf 1
ax1 = fig.add_subplot(231)
ax1.set_title('Hrvatska',fontsize=25)
ax1.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax1.plot(df_croatia['Last_Update'],
         100*df_croatia["Deaths"]/df_croatia["Confirmed"],
         color='green')

# Podgraf 2
ax2 = fig.add_subplot(232)
ax2.set_title('Srbija',fontsize=25)
ax2.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax2.plot(df_srb['Last_Update'],
         100*df_srb['Deaths']/df_srb["Confirmed"],
         color='purple')

# Podgraf 3
ax3 = fig.add_subplot(233)
ax3.set_title('Bosna i Hercegovina',fontsize=25)
ax3.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax3.plot(df_bih['Last_Update'],
         100*df_bih['Deaths']/df_bih["Confirmed"],
         color='magenta')

# Podgraf 4
ax4 = fig.add_subplot(234)
ax4.set_title('Crna Gora',fontsize=25)
```

```
ax4.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax4.plot(df_mne['Last_Update'],
         100*df_mne['Deaths']/df_mne["Confirmed"],
         color='orange')

# Podgraf 5
ax5 = fig.add_subplot(235)
ax5.set_title('Mađarska',fontsize=25)
ax5.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax5.plot(df_hun["Last_Update"],
         100*df_hun["Deaths"]/df_hun["Confirmed"],
         color='chocolate')

#Podgraf 6
ax6 = fig.add_subplot(236)
ax6.set_title('Slovenija',fontsize=25)
ax6.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax6.plot(df_slo["Last_Update"],
         100*df_slo["Deaths"]/df_slo["Confirmed"],
         color='teal')


plt.show()
```



Slika 30: Figura s grafikonima koji usporedno prikazuju stopu smrtnosti država regije (Izvor: vlastita izrada)

4.5. Analiza i obrada skupa podataka 3

Napomena: Budući da se radi o jedinstvenoj problemskoj domeni unutar koje se nalaze sva četiri skupa podataka, a isto tako budući da je problemska domena prethodno opisana i analizirana u ranijim poglavljima, u ovom dijelu sama domena se neće opisivati, no vrlo je važno poznavati osnovne metapodatke o domeni i problemu o kojemu se podaci prikupljaju.

4.5.1. Priprema skupa podataka

Treći skup podataka uvezen je u Pandas data frame iz `csv` datoteke u varijablu `df_confirmed` u koju je pohranjen. Ovaj skup sadrži podatke o potvrđenim slučajevima po svim državama te isto tako po svim zabilježenim danima **počevši od početka mjerenja**, dakle s 22. siječnja. Stupci prestavljaju dan u promatranom periodu pa prema tome sjecište stupca i retka daje točan podatak o broju slučajeva potvrđenih na točno taj dan.

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	8/27/20	8/28/20	8/29/20	8/30/20	8/31/20	9
0	NaN	Afghanistan	33.939110	67.709953	0	0	0	0	0	0	...	38129	38140	38143	38162	38165	3
1	NaN	Albania	41.153300	20.168300	0	0	0	0	0	0	...	9083	9195	9279	9380	9513	
2	NaN	Algeria	28.033900	1.659600	0	0	0	0	0	0	...	43016	43403	43781	44146	44494	4
3	NaN	Andorra	42.506300	1.521800	0	0	0	0	0	0	...	1098	1124	1124	1124	1176	
4	NaN	Angola	-11.202700	17.873900	0	0	0	0	0	0	...	2415	2471	2551	2624	2654	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
261	NaN	West Bank and Gaza	31.952200	35.233200	0	0	0	0	0	0	...	20677	21251	21668	22204	22729	2
262	NaN	Western Sahara	24.215500	-12.885800	0	0	0	0	0	0	...	10	10	10	10	10	
263	NaN	Yemen	15.552727	48.516388	0	0	0	0	0	0	...	1933	1943	1946	1953	1958	
264	NaN	Zambia	-13.133897	27.849332	0	0	0	0	0	0	...	11601	11779	11902	12025	12097	1
265	NaN	Zimbabwe	-19.015438	29.154857	0	0	0	0	0	0	...	6292	6388	6406	6412	6497	

Slika 31: Prikaz data framea kreiranog iz trećeg skupa podataka (Izvor: vlastita izrada)

Pomoću Pandas data frame atributa `shape` dobit ćemo uređeni par **(266,232)** gdje 266 predstavlja broj redaka, a 232 broj stupaca. Nadalje, pomoću metode `info()`, prikazat ćemo osnovne informacije o tipovima podataka i zauzetosti memorije.

```
df_confirmed.shape
#(266, 232)


df_confirmed.info()

#<class 'Pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Columns: 232 entries, State to 9/5/20
dtypes: float64(2), int64(228), object(2)
memory usage: 482.2+ KB
```

Naime, vidimo kako je ispis metode `info()` za ovaj data frame nešto drugačiji obzirom da se radi o velikom broju stupaca. Prikazani su zapisi te tipovi podataka i njihova brojnost.

U nastavku ćemo izbaciti nepotrebne stupce pomoću metode `drop()`, ali i primijeniti metodu `fillna()` koja će nepostojeće vrijednosti zamijeniti eksplicitno zadanim vrijednostima.

```
df_confirmed.drop(columns=["Lat","Long"],inplace=True)
df_confirmed["State"].fillna("Not available",inplace=True)
```

4.5.2. Modeliranje podataka

Izvršimo nekoliko jednostavnih operacija nad skupom podataka. Dakle, prva operacija vratit će data frame koji prikazuje **države bez dodijeljene vrijednosti atributa `Province_State`**. Druga operacija učinit će obrnuto, dakle prikazati **države s dodijeljenom vrijednošću atributa `Province_State`**. Treća operacija će **grupirati države** te **sumarno prikazati podatke po svakoj državi**. Primijetit ćemo kako ćemo u ovome slučaju dobiti skup koji podsjeća na skup

1. Dakle, skup će prikazivati vrijednosti za svaku državu po svakom od datuma, no ovdje se, dakako, radi samo o vrijednostima potvrđenih slučajeva zaraze.

```
df_confirmed[df_confirmed["State"] == "Not available"]

df_confirmed[df_confirmed["State"] != "Not available"]

df_conf_grouped = df_confirmed.groupby("Country").sum()
df_conf_grouped
```

	State	Country	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	...	8/27/20	8/28/20	8/29/20	8/30/20	8/31/20	9/1/20	9/2/20
0	Not available	Afghanistan	0	0	0	0	0	0	0	0	...	38129	38140	38143	38162	38165	38196	38243
1	Not available	Albania	0	0	0	0	0	0	0	0	...	9083	9195	9279	9380	9513	9606	9728
2	Not available	Algeria	0	0	0	0	0	0	0	0	...	43016	43403	43781	44146	44494	44833	45158
3	Not available	Andorra	0	0	0	0	0	0	0	0	...	1098	1124	1124	1124	1176	1184	1199
4	Not available	Angola	0	0	0	0	0	0	0	0	...	2415	2471	2551	2624	2654	2729	2777
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
261	Not available	West Bank and Gaza	0	0	0	0	0	0	0	0	...	20677	21251	21668	22204	22729	23281	23875

Slika 32: Prikaz data framea s državama koje nemaju dodijeljenu vrijednost prvoga atributa (Izvor: vlastita izrada)

Sljedeći dio opisat će dvije zanimljive funkcije Pandasa, a to su `loc[]` te `T`. Naime, atribut `loc[]` kao parametar prima indeks retka, a kao rezultat vraća cijeli redak. Operacija `T` transponira data frame tako da stupci postaju reci, a reci postaju stupci. Prikaz slijedi u nastavku:

```
cro_T = pd.DataFrame(df_conf_grouped.loc["Croatia"])
cro_T
```

Rezultat ispisa je sljedeći:

	Croatia
1/22/20	0
1/23/20	0
1/24/20	0
1/25/20	0
1/26/20	0
...	...
9/1/20	10414
9/2/20	10725
9/3/20	11094
9/4/20	11428
9/5/20	11739

Slika 33: Prikaz data framea kreiranog pomoću operacija `loc[]` i `T` (Izvor: vlastita izrada)

U sljedećem, nešto kompleksnijem primjeru, grupirat ćemo data frame po državama, zatim pomoću metode `dif()` za svako polje izračunati promjenu vrijednosti (poljeX - (poljeX-1)), sortirati vrijednosti te uzeti prvih deset sortiranih vrijednosti. Rezultat je **dnevni postotni**

prirast broja novopotvrđenih slučajeva u 5 odabranih država. Grafikon je prikazan u sljedećoj sekciji, a programski kod ekstrahiranja data framea u nastavku:

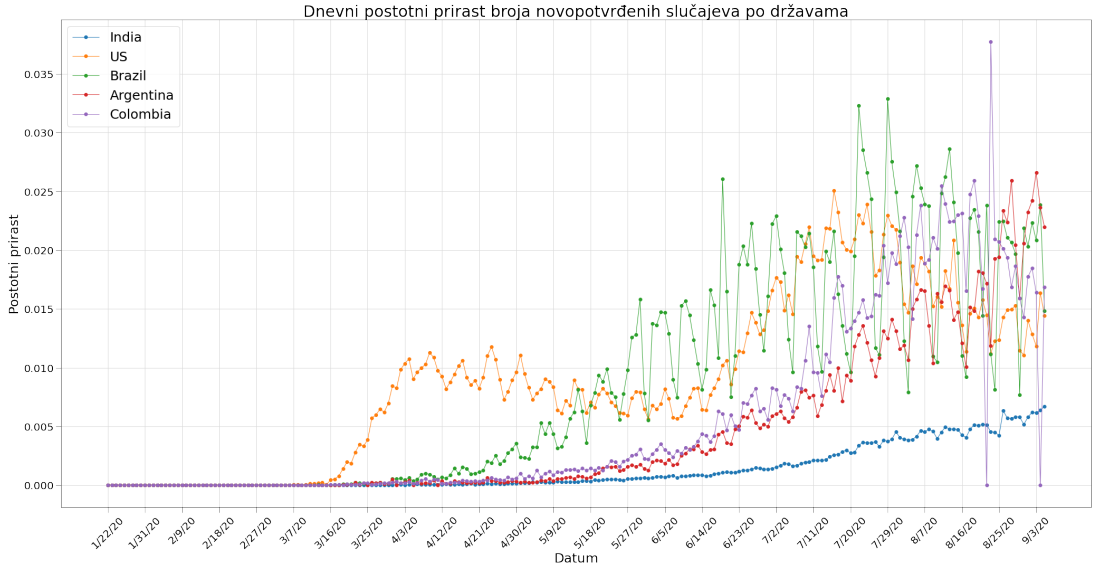
```
df_confirmed.groupby("Country").sum().diff(axis=1).sort_values(df_confirmed.columns[-1],
 ↪  =False).head(10).replace(np.nan,0)
```

4.5.3. Prikaz rezultata

GRAF 3.3 - dnevni prirast broja novooboljelih po nekoliko odabranih država

```
dfGrowth =
 ↪  df_confirmed.groupby('Country').sum().diff(axis='columns').sort_values(df_confirmed
 ↪  =False).head(5).replace(np.nan,0)
dts = dfGrowth.columns
pop = {
  "India": 1352642280,
  "US": 308401808,
  "Brazil": 210147125,
  "Argentina":45195774,
  "Spain":50372424
}

f = plt.figure(figsize=(35,17))
ax = f.add_subplot(111)

for i,country in enumerate(dfGrowth.index):            # dfGrowth.index su
 ↪  države
    t = dfGrowth.loc[temp.index == country].values[0]
    t = t[t>=0]

    date = np.arange(0,len(t[:]))
    plt.plot(date,(t/pop[country])*100,'-o',label = country,linewidth =1)

plt.xticks(list(np.arange(0,len(dts),int(len(dts)/25))),dts[:-1:int(len(dts)/25)])

plt.tick_params(which='both', width=1,labelsize=20,length=10)
plt.xticks(rotation=45)

ax.grid(lw = 1, ls = '-', c = "0.85", which = 'major')
ax.grid(lw = 1, ls = '-', c = "0.95", which = 'minor')

plt.title("Dnevni postotni prirast broja novopotvrđenih slučajeva po
 ↪  državama",fontsize=30)


plt.xlabel("Datum",fontsize =25)
```

```
33 plt.ylabel("Postotni prirast",fontsize =25)
34
35 plt.legend(fontsize=25)
```



Slika 34: Prikaz data framea kreiranog iz trećeg skupa podataka (Izvor: vlastita izrada)

Prikaz još nekoliko rezultata obrade skupa 1 nalazi se u `.ipynb` prilogu na kraju rada.

4.6. Analiza i obrada skupa podataka 4

Napomena: Budući da se radi o jedinstvenoj problemskoj domeni unutar koje se nalaze sva četiri skupa podataka, a isto tako budući da je problemska domena prethodno opisana i analizirana u ranijim poglavljima, u ovom dijelu sama domena se neće opisivati, no vrlo je važno poznavati osnovne metapodatke o domeni i problemu o kojemu se podaci prikupljaju.

4.6.1. Priprema skupa podataka

Četvrti skup podataka uvezen je u Pandas data frame iz `csv` datoteke u varijablu `df_deaths` u koju je pohranjen. Ovaj skup sadrži podatke o smrtnim slučajevima po svim državama te isto tako po svim zabilježenim danima **počevši od početka mjerenja**, dakle s 22. siječnja. Stupci prestavljaju dan u promatranom periodu pa prema tome sjecište stupca i retka daje točan podatak o broju slučajeva umrlih na točno taj dan.

Pomoću Pandas data frame atributa `shape` dobit ćemo uređeni par **(266,232)** gdje 266 predstavlja broj redaka, a 232 broj stupaca. Nadalje, pomoću metode `info()`, prikazat ćemo osnovne informacije o tipovima podataka i zauzetosti memorije.

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	8/27/20	8/28/20	8/29/20	8/30/20	8/31/20	9
0	NaN	Afghanistan	33.939110	67.709953	0	0	0	0	0	0	...	1401	1402	1402	1402	1402	
1	NaN	Albania	41.153300	20.168300	0	0	0	0	0	0	...	266	271	275	280	284	
2	NaN	Algeria	28.033900	1.659600	0	0	0	0	0	0	...	1475	1483	1491	1501	1510	
3	NaN	Andorra	42.506300	1.521800	0	0	0	0	0	0	...	53	53	53	53	53	
4	NaN	Angola	-11.202700	17.873900	0	0	0	0	0	0	...	105	106	107	107	108	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
261	NaN	West Bank and Gaza	31.952200	35.233200	0	0	0	0	0	0	...	141	145	147	152	152	
262	NaN	Western Sahara	24.215500	-12.885800	0	0	0	0	0	0	...	1	1	1	1	1	
263	NaN	Yemen	15.552727	48.516388	0	0	0	0	0	0	...	562	563	563	564	566	
264	NaN	Zambia	-13.133897	27.849332	0	0	0	0	0	0	...	282	283	284	287	288	
265	NaN	Zimbabwe	-19.015438	29.154857	0	0	0	0	0	0	...	189	195	196	196	202	

Slika 35: Prikaz data framea kreiranog iz četvrtog skupa podataka (Izvor: vlastita izrada)

```
df_confirmed.shape
#(266, 232)


df_confirmed.info()

#<class 'Pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Columns: 232 entries, State to 9/5/20
dtypes: float64(2), int64(228), object(2)
memory usage: 482.2+ KB
```

Zapisi ovog skupa i prethodnog su identični.

U nastavku će biti prikazano uklanjanje nepotrebnih stupaca te prilagodba tipova podataka data framea. Isto tako, pomoću metode `fillna()` popunit će se polja vrijednosti koje su nevažeće ili nepostojeće.

```
df_deaths.drop(columns=["Lat","Long"],inplace=True)

df_deaths["State"].fillna("Not available",inplace=True)

df_deaths.info()
```

4.6.2. Modeliranje podataka

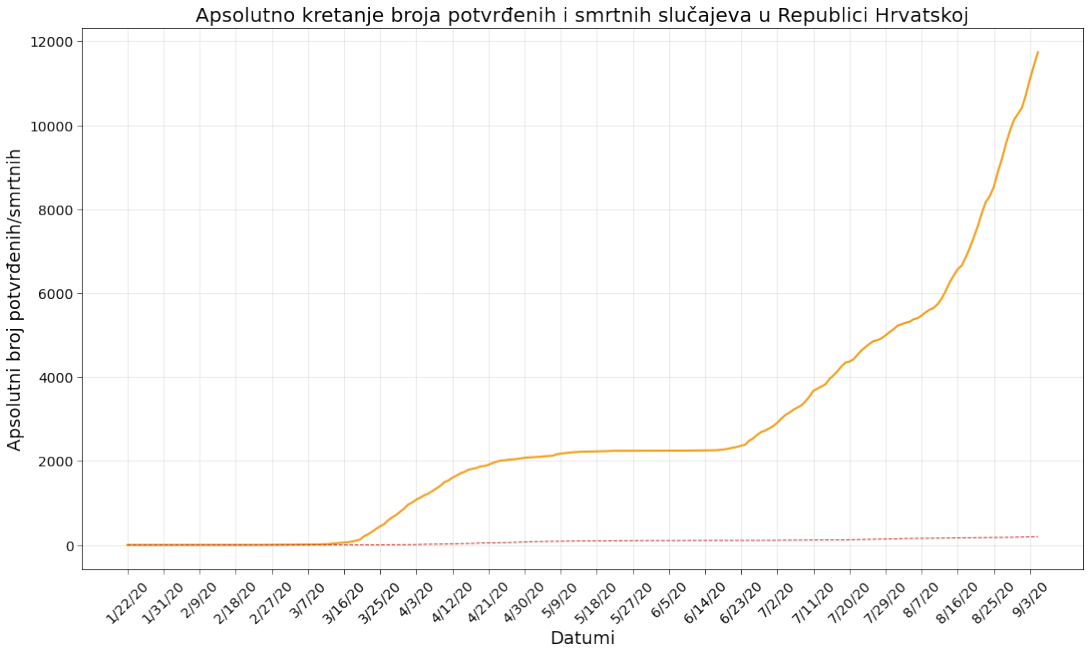
Pomoću operacije `loc[]` ekstrahirat ćemo podatke za Republiku Hrvatsku, ali ćemo ih istovremeno pretvoriti u data frame te će se tako automatski transponirati u data frame s jednim stupcem i onoliko redaka koliko je datuma zabilježeno.

	Croatia
1/22/20	0
1/23/20	0
1/24/20	0
1/25/20	0
1/26/20	0
...	...
9/1/20	187
9/2/20	191
9/3/20	194
9/4/20	195
9/5/20	197

Slika 36: Prikaz data framea kreiranog pomoću operacija `loc[]` i `pd.DataFrame()` (Izvor: vlastita izrada)

4.6.3. Prikaz rezultata

```
f = plt.figure(figsize=(18,10))
f.add_subplot(111)

plt.axes(axisbelow=True)
plt.plot(cro_T.index,cro_T, c="#f39c12", linewidth=2, linestyle='-')
plt.plot(cro_deaths_T.index,cro_deaths_T, c="#c0392b", linewidth=1,
 ↪  linestyle='--')
plt.tick_params(size=5, labelsize = 14)
plt.xticks(list(np.arange(0,len(cro_T.index),int(len(cro_T.index)/25))),rotation=45)

plt.xlabel("Datumi",fontsize=18)
plt.ylabel("Apsolutni broj potvrđenih/smrtnih",fontsize=18)
plt.title("Apsolutno kretanje broja potvrđenih i smrtnih slučajeva u
 ↪  Republici Hrvatskoj",fontsize=20)
plt.grid(alpha = 0.3,which='both')
plt.show()
```



Slika 37: Prikaz apsolutnog kretanja broja potvrđenih i smrtnih slučajeva u Republici Hrvatskoj (Izvor: vlastita izrada)

Prikaz još nekoliko rezultata obrade skupa 1 nalazi se u `.ipynb` prilogu na kraju rada.

5. Zaključak

Ovaj rad prikazao je najvažnije koncepte metodologije obrade podataka na stručno utemeljenom procesnom modelu, ali i na praktičnim primjerima pomoću programskog jezika Python a sve unutar Jupyter Notebook razvojne okoline. Ovim radom prikazana je metodologija i proces obrade podatka, od uvoza formatiranog i strukturiranog skupa podataka, čitanja tog istog skupa pomoću Pandas modula te manipulacije nad podacima skupa pomoću Pandas metoda. Pokazano je kako je vrlo važan metodološki pristup jer samo takvim pristupom postiže se modularnost faza obrade i analize što pospješuje detekciju potencijalnih pogreški ili incidenata prilikom obrade podataka. Dakako, produktivnost ove metodologije u svom punom sjaju pokazala bi se prilikom primjene unutar velikih korporacija gdje se svakodnevno obrađuju enormne količine podataka. U ovome radu, naglasak je na konceptu obrade kao procesu. Unutar svake od faza postoje točno definirani obrasci primjene metoda koji, prate li se dosljedno, mogu povećati i pospješiti produktivnost procesa obrade i analize. Na kraju, prikaz rezultata obrade proveden je putem grafičkih ili tabličnih vizualizacija pomoću modula Matplotlib. Iz prikazanih grafičkih vizualizacija može se jasno, pomoću temeljnog poznavanja statistike, pročitati ono što je cilj ovoga cjelokupnoga procesa, a to je znanje. Analizom i modeliranjem sirovih brojki, dobili smo modele koji mogu pomoći pri odlučivanju, donošenju pravodobnih i pravovaljanih odluka što je u svakom poslovnom sustavu vrlo bitno. Na primjeru korištenome u radu, dakle primjeru pandemije virusa, vidimo kako jasno možemo pomoću pravovaljanog praćenja podataka kreirati vrlo vrijedne resurse znanja koji će, ako su kreirani na korektan način, donositelje odluka usmjeriti u ispravnome smjeru.

Popis literature

[1] J. VanderPlas, *Python Data Science Handbook - Essential Tools for Working with Data*, K. Schanafelt, ur. United States of America: O'Reilly Media, Inc., 2017, 548 str., ISBN: 978-1-4919-1205-8.

[2] W. McKinney, *Python for Data Analysis*, 2nd. United States of America: O'Reilly Media, Inc., 541 str., ISBN: 978-1-4919-5766-0.

[3] (). Project Jupyter | Documentation, Project Jupyter, adresa: https://jupyter.org/documentation (pogledano 28. 7. 2020).

[4] (). pandas documentation — pandas 1.1.0 documentation, adresa: https://pandas.pydata.org/docs/ (pogledano 29. 7. 2020).

[5] H. Stepanek, *Thinking in Pandas*. Portland, Oregon, United States of America: Apress, a Springer Nature company, 2020, 190 str., ISBN: 978-1-4842-5839-2.

[6] K. Akiyama, „First m87 event horizon telescope results. i. the shadow of the supermassive black hole", *The Astrophysical Journal Letters*, str. 17, 2019.

[7] M. K. Leksikografski Zavod, *Znanje*, *Hrvatska enciklopedija*, Zagreb: Leksikografski zavod Miroslav Krleža, 2020. adresa: http://www.enciklopedija.hr/Natuknica.aspx?ID=67357 (pogledano 3. 8. 2020).

[8] M. Maleković, „Priroda znanja", prezentacija s predavanja, FOI, Varaždin, (pogledano 2. 8. 2020).

[9] F. Provost i T. Fawcett, *Data Science for Business*, 1st. United States of America: O'Reilly Media, Inc., 25. srpnja 2013, 409 str., ISBN: 978-1-4493-6132-7.

[10] R. Wirth i J. Hipp, *CRISP-DM: Towards a Standard Process Model for DataMining*. adresa: https://pdfs.semanticscholar.org/48b9/293cfd4297f855867ca278f7069abc6a9c24.pdf (pogledano 6. 8. 2020).

[11] M. Barber. (16. siječnja 2018). Data science concepts you need to know! part 1, Medium, adresa: https://towardsdatascience.com/introduction-to-statistics-e9d72d818745 (pogledano 6. 9. 2020).

[12] Anaconda. (). Conda environments — conda 4.8.4.post49+bf8ad50a documentation, Conda docs, adresa: https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/environments.html (pogledano 6. 9. 2020).

[13] E. D. Portal. (). What is open data?, European Data Portal, adresa: https://www.europeandataportal.eu/elearning/en/module1/#/id/co-01 (pogledano 6. 9. 2020).

Popis slika

1. Prilog 1

U nastavku se nalazi prilog `.ipynb` Jupyter Notebook datoteke izvezene izravno iz Jupyter Notebooka. Datoteka sadrži naslov i kratki opis problematike te po poglavljima podijeljenu obradu četiri zasebna skupa podataka.

Analiza velikog skupa podataka pomoću programskog jezika Python i razvojnog okružja Jupyter Notebook

Big Data Analysis in Python and Jupyter Notebook

Završni rad

Sveučilište u Zagrebu, Fakultet organizacije i informatike

Student: Josip Rosandić
Smjer studija: Informacijski sustavi
Mentor: Izv.prof.dr.sc. Markus Schatten

Sažetak

Ovaj rad čitatelja uvodi u sferu analize i obrade podataka. Naglasak rada stavljen je na analizu otvorenog skupa podataka pomoću programskog jezika Python i modula Pandas, a sve unutar razvojnog okružja Jupyter Notebook. Nad otvorenim skupom podataka u valjanom datotečnom obliku čitljivom unutar korištenih modula provest´ce se analiza podataka što uključuje analizu količine podataka, tipova podataka, valjanosti podataka, strukture unutar koje su podaci pohranjeni. Nadalje, rad uključuje i prikaz optimiranja tipova podataka u svrhu postizanjaboljih memorijskih performansi. Zatim slijedi semantička analiza podataka, procesiranje skupa te eksploatiranje nove vrijednosti iz postoje´cih resursa. Procesi obrade većinski su dio Python modula Pandas unutar kojega se nalaze metode za čitanje odgovaraju´ce datoteke skupa podataka, zatim formiranje podatkovnih okvira kao temeljne obradbene strukture ovoga modula te primjene atributa i metoda ovoga modula nad formiranim podatkovnim okvirima. Novo znanje korisno je vizualizirati korištenjem Python modula Matplotlib. Fokus rada trebao bi biti sami proces koji, promatramo li ga kao crnu kutiju, kao ulaze prima podatke, a kao izlaze vraća odgovarajuće semantički jasne, egzaktne i precizne izračune i vizualizacije.

Izvor podataka: COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University.
https://github.com/CSSEGISandData/COVID-19 (https://github.com/CSSEGISandData/COVID-19).

Uvoz potrebnih biblioteka

In [74]:
```
import pandas as pd #uvoz pandas biblioteke s aliasom "pd"
import numpy as np #uvoz numpy biblioteke s aliasom np
import matplotlib.pyplot as plt #uvoz matplotlib bilbioteke s aliasom plt
```

1. SKUP PODATAKA #1

Uvoz skupa podataka

In [75]:
```
# Skup podataka #1
df_latest_data = pd.read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-data/data/cases_country.csv")
```

1.1. Opis i analiza formata skupa #1

U varijablu `df_latest_data`, odnosno u pandas data frame uvezeni su podaci iz `.csv` datoteke `cases_country.csv`. Navedeni skup podataka sastoji se od **188 redova** te (inicijalno) **14 stupaca**. Svaki redak predstavlja jednu državu svijeta, a svaki stupac predstavlja atribute svojstvene problemskoj domeni. Tako od inicijalnih atributa, ovaj data frame sadrži:

- `Country_Region` - država ili regija o kojoj se vrijednosti atributa bilježe
- `Last_Update` - obzirom da je ovo skup podataka koji se svakodnevno ažurira unutar CSSE Centra u Baltimoreu, postoji atribut koji prestavlja timestamp kada su podaci posljednji put ažurirani.
- `Lat` - odnosi se na geografsku širinu lokacije
- `Long_` - odnosi se na geografsku dužinu lokacije
- `Confirmed` - broj potvrđenh slučaja zaraze
- `Deaths` - broj potvrđenih smrtnih slučaja uzrokovanih bolešću COVID19
- `Recovered` - broj slučajeva oporavka oboljelih
- `Active` - broj aktivnih slučajeva (dobije se tako da se od broja potvrđenih oduzme broj umrlih i oporavljenih)
- `Incident rate` - incidencija slučajeva na 100 000 osoba
- `People_Tested` - broj ljudi podvrgnutih testiranju na COVID19
- `People_Hospitalized` - broj ljudi hospitaliziranih uslijed zaraze
- `Mortality_Rate` - stopa smrtnosti od bolesti COVID19 ((brojUmrlih * 100)/brojPotvrdenih)
- `UID` - identifikator svakog retka
- `ISO3` - službeni identifikator države

In [76]:
```
# Pomoću varijable ispisujemo data frame skupa #1
df_latest_data
```

Out[76]:

	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	Incider
0	Afghanistan	2020-09-09 12:28:54	33.939110	67.709953	38544.0	1420.0	31048.0	6076.0	99.
1	Albania	2020-09-09 12:28:54	41.153300	20.168300	10553.0	321.0	6239.0	3993.0	366.
2	Algeria	2020-09-09 12:28:54	28.033900	1.659600	46938.0	1571.0	33183.0	12184.0	107.
3	Andorra	2020-09-09 12:28:54	42.506300	1.521800	1261.0	53.0	934.0	274.0	1632.
4	Angola	2020-09-09 12:28:54	-11.202700	17.873900	3033.0	124.0	1215.0	1694.0	9.
...	...	...	...	...	...	...	...	...	
183	West Bank and Gaza	2020-09-09 12:28:54	31.952200	35.233200	27919.0	192.0	18466.0	9261.0	547.
184	Western Sahara	2020-09-09 12:28:54	24.215500	-12.885800	10.0	1.0	8.0	1.0	1.
185	Yemen	2020-09-09 12:28:54	15.552727	48.516388	1994.0	576.0	1203.0	215.0	6.
186	Zambia	2020-09-09 12:28:54	-13.133897	27.849332	12952.0	297.0	11787.0	868.0	70.
187	Zimbabwe	2020-09-09 12:28:54	-19.015438	29.154857	7388.0	218.0	5477.0	1693.0	49.

188 rows × 14 columns

1.2. Tipovi podataka i potrošnja resursa

U nastavku vidimo prikaz informacija o data frameu. Kao što je ranije navedeno, radi se o 14 stupaca i 188 redaka što možemo vidjeti i pomoću atributa `shape` u nastavku:

In [77]:
```
df_latest_data.shape
```

Out[77]:
```
(188, 14)
```

Pogledajmo informacije o tipovima podataka prikazane pomoću metode `.info()`

In [78]:
```
df_latest_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 188 entries, 0 to 187
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Country_Region       188 non-null    object 
 1   Last_Update          188 non-null    object 
 2   Lat                  186 non-null    float64
 3   Long_                186 non-null    float64
 4   Confirmed            188 non-null    float64
 5   Deaths               188 non-null    float64
 6   Recovered            185 non-null    float64
 7   Active               188 non-null    float64
 8   Incident_Rate        186 non-null    float64
 9   People_Tested        0 non-null      float64
 10  People_Hospitalized  0 non-null      float64
 11  Mortality_Rate       188 non-null    float64
 12  UID                  188 non-null    int64  
 13  ISO3                 186 non-null    object 
dtypes: float64(10), int64(1), object(3)
memory usage: 20.7+ KB
```

Vidimo kako su tipovi podataka ne bitno različiti. Dakle, prisutni su float64, int64 te object. Radi se o tomu da pandas prilikom obrade `.csv` datoteke sve cjelobrojne numeričke podatke tipa integer (int64) pretvara u decimalni format tipa float64. Nadalje, object nije vrlo poželjan tip podatka i dobro je pretvoriti ga u specifičan tip podataka gdje god je to moguće.

Unutar pandas biblioteke postoji metoda `infer_objects().dtypes` koja sve tipove podataka pretvara u one tipove koji su prikladni za pojedini skup.

In [79]:
```
# Metoda memory_usage() vraća memorijsku veličinu svakog stupca u bajtovima --> ovdje smo dijelili
# s 1e6 kako bismo dobili megabajte
df_latest_data.memory_usage()/1000000
```

Out[79]:
```
Index                 0.000128
Country_Region        0.001504
Last_Update           0.001504
Lat                   0.001504
Long_                 0.001504
Confirmed             0.001504
Deaths                0.001504
Recovered             0.001504
Active                0.001504
Incident_Rate         0.001504
People_Tested         0.001504
People_Hospitalized   0.001504
Mortality_Rate        0.001504
UID                   0.001504
ISO3                  0.001504
dtype: float64
```

1.3. Redukcija i optimizacija skupa #1

Učinimo redukciju i optimizaciju skupa podataka. U nastavku ćemo izbaciti nepotrebne stupe te optimirati data frame na razini tipova podataka. Kao što smo ranije vidjeli, metodom info dolazimo do informacije koja govori kako data frame zauzima oko **20.7+ KB** memorijskog prostora.

In [80]:
```
df_latest_data.drop(columns=["Lat","Long_","Incident_Rate","People_Tested","People_Hospitalized","ISO3","UID","Mortality_Rate"],inplace=True)
```

In [81]:
```
df_latest_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 188 entries, 0 to 187
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Country_Region  188 non-null    object 
 1   Last_Update     188 non-null    object 
 2   Confirmed       188 non-null    float64
 3   Deaths          188 non-null    float64
 4   Recovered       185 non-null    float64
 5   Active          188 non-null    float64
dtypes: float64(4), object(2)
memory usage: 8.9+ KB
```

Naime, sada možemo vidjeti da smo eliminacijom podataka koji nam nisu nužni pri obradi za oko:

In [82]:
```
str(((20.7-8.9)/20.7)*100) + "%"
```

Out[82]:
```
'57.00483091787439%'
```

što sada iznosi **8.9+ KB**.

Nadalje, bilo bi dobro pozabaviti se tipovima podataka:

In [83]:
```
#df_latest_data = df_latest_data.rename(columns={"Country_Region": "Country"})
df_latest_data.set_index(keys="Country_Region", inplace=True)
```

In [84]:
```
# Parsiranje datuma
df_latest_data["Last_Update"] = pd.to_datetime(df_latest_data["Last_Update"])

# Pretvorba u int64
df_latest_data["Confirmed"] = df_latest_data["Confirmed"].astype("int64")
df_latest_data["Deaths"] = df_latest_data["Deaths"].astype("int64")
df_latest_data["Active"] = df_latest_data["Active"].astype("int64")
df_latest_data["Recovered"].fillna(0,inplace = True) # Zamjena nepoznatih vrijed
nosti eksplicitnim vrijednostima
df_latest_data["Recovered"] = df_latest_data["Recovered"].astype("int64")
```

In [85]:
```
df_latest_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 188 entries, Afghanistan to Zimbabwe
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype         
---  ------       --------------  -----         
 0   Last_Update  188 non-null    datetime64[ns]
 1   Confirmed    188 non-null    int64         
 2   Deaths       188 non-null    int64         
 3   Recovered    188 non-null    int64         
 4   Active       188 non-null    int64         
dtypes: datetime64[ns](1), int64(4)
memory usage: 8.8+ KB
```

Kao što možemo vidjeti, neznatno se smanjila veličina skupa, no ovakva promjene puno su značajnije na velikim skupovima podataka.

1.4. Modeliranje skupa podataka

Analiza i obrada

Najnoviji podaci za **Republiku Hrvatsku**:

In [86]:
```
cro_newest = pd.DataFrame(df_latest_data.loc["Croatia"]).T
cro_newest
```

Out[86]:

	Last_Update	Confirmed	Deaths	Recovered	Active
Croatia	2020-09-09 12:28:54	12626	206	9833	2587

Pomoću metode `.head()` prikažimo prvih nekoliko redova skupa:

In [87]:
```
df_latest_data.head(10)
```

Out[87]:

Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active
Afghanistan	2020-09-09 12:28:54	38544	1420	31048	6076
Albania	2020-09-09 12:28:54	10553	321	6239	3993
Algeria	2020-09-09 12:28:54	46938	1571	33183	12184
Andorra	2020-09-09 12:28:54	1261	53	934	274
Angola	2020-09-09 12:28:54	3033	124	1215	1694
Antigua and Barbuda	2020-09-09 12:28:54	95	3	91	1
Argentina	2020-09-09 12:28:54	500034	10405	366590	123039
Armenia	2020-09-09 12:28:54	45152	905	41023	3224
Australia	2020-09-09 12:28:54	26465	781	22861	2823
Austria	2020-09-09 12:28:54	30583	747	25764	4072

U nastavku ćemo nad skupom ćemo neke od statističkih metoda pandas biblioteke:

- `sum()`
- `count()`
- `median()`
- `min()`
- `max()`

In [88]:
```
df_latest_data["Confirmed"].sum()
```

Out[88]: 27610660

In [89]:
```
df_latest_data["Confirmed"].count()
```

Out[89]: 188

In [90]:
```
df_latest_data["Confirmed"].median()
```

Out[90]: 9649.0

In [91]:
```
# Država s najmanje potvrđenih slučajeva u ovom slučaju nije država nego kruzer MS Zaandam
df_latest_data["Confirmed"].min()
df_latest_data[df_latest_data["Confirmed"] == df_latest_data["Confirmed"].min()]
```

Out[91]:

Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active
MS Zaandam	2020-09-09 12:28:54	9	2	0	7

In [92]:
```
# Država s najviše potvrđenih slučajeva jesu Sjedinjene Američke Države
df_latest_data["Confirmed"].max()
df_latest_data[df_latest_data["Confirmed"] == df_latest_data["Confirmed"].max()]
```

Out[92]:

Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active
US	2020-09-09 12:28:54	6328154	189699	2359111	3779329

In [93]:
```
# Medijalna vrijednost broja potvrđenih slučajeva koja nam tumači kako polovica država ima manje od
# 9054 potvrđena slučaja, dok polovica država ima više potvrđenih slučajeva
df_latest_data["Confirmed"].median()
```

Out[93]: 9649.0

Pozovimo metodu `.describe()` koja će nam za svaki stupac prikazati osnovne metrike deskriptivne statistike:

In [94]:
```
df_latest_data.describe().astype("float").round(2)
```

Out[94]:

	Confirmed	Deaths	Recovered	Active
count	188.00	188.00	188.00	188.00
mean	146865.21	4778.18	98765.12	43323.56
std	643275.91	18860.64	407448.06	286727.09
min	9.00	0.00	0.00	0.00
25%	2046.50	35.75	1127.00	321.00
50%	9649.00	197.00	6074.00	1860.50
75%	61975.75	1072.00	42210.00	10735.75
max	6328154.00	189699.00	3572421.00	3779329.00

Promotrimo sada nekoliko **vršnih vrijednosti** po svakom od atributa:

In [95]:
```
# Prikaz 10 država s najmanjim brojem potvrđenih slučajeva
pd.DataFrame(df_latest_data.sort_values("Confirmed")["Confirmed"].head(10))
```

Out[95]:

Country_Region	Confirmed
MS Zaandam	9
Western Sahara	10
Holy See	12
Saint Kitts and Nevis	17
Dominica	22
Laos	22
Grenada	24
Saint Lucia	26
Timor-Leste	27
Fiji	32

In [96]:
```
# Prikaz 10 država s najvećim brojem potvrđenih slučajeva
pd.DataFrame(df_latest_data.sort_values("Confirmed", ascending=False)["Confirme
d"].head(10))
```

Out[96]:

Country_Region	Confirmed
US	6328154
India	4370128
Brazil	4162073
Russia	1037526
Peru	691575
Colombia	679181
Mexico	642860
South Africa	640441
Spain	534513
Argentina	500034

In [97]:
```
# Prikaz 10 država s najmanjim brojem smrtnih slučajeva
pd.DataFrame(df_latest_data.sort_values("Deaths")["Deaths"].head(10))
```

Out[97]:

Country_Region	Deaths
Timor-Leste	0
Saint Kitts and Nevis	0
Cambodia	0
Dominica	0
Saint Vincent and the Grenadines	0
Bhutan	0
Holy See	0
Laos	0
Seychelles	0
Saint Lucia	0

In [98]:
```
# Prikaz 10 država s najvećim brojem smrtnih slučajeva
pd.DataFrame(df_latest_data.sort_values("Deaths", ascending=False)["Deaths"].hea
d(10))
```

Out[98]:

Country_Region	Deaths
US	189699
Brazil	127464
India	73890
Mexico	68484
United Kingdom	41675
Italy	35563
France	30770
Peru	29976
Spain	29594
Iran	22669

Pristupimo sada podskupu država koje imaju **manje potvrđenih slučajeva od medijalne vrijednosti**: (GRAF 1.5.)

In [99]:
```
median_less = df_latest_data[df_latest_data["Confirmed"] < df_latest_data["Confi
rmed"].median()]["Confirmed"]
```

In [100]:
```
median_less = pd.DataFrame(median_less)
median_less["Confirmed"].sort_values()
```

Out[100]:
```
Country_Region
MS Zaandam                    9
Western Sahara               10
Holy See                     12
Saint Kitts and Nevis        17
Dominica                     22
                           ...
Gabon                      8608
Maldives                   8741
Tajikistan                 8860
Namibia                    8928
Malaysia                   9583
Name: Confirmed, Length: 94, dtype: int64
```

In [101]:
```
pd.DataFrame(median_less).describe()
```

Out[101]:

	Confirmed
count	94.000000
mean	2673.308511
std	2569.874471
min	9.000000
25%	473.250000
50%	2029.000000
75%	4394.250000
max	9583.000000

Sumarni prikaz vrijednosti svih relevantnih atributa u retku data framea:

In [102]:
```
df_summary = pd.DataFrame(pd.to_numeric(df_latest_data.sum()),dtype=np.float6
4).T
df_summary["Mortality Rate (per 100)"] =  (100*df_summary["Deaths"]/df_summary
["Confirmed"]).round(2)
df_summary.style.background_gradient(cmap='Reds',axis=1).format("{:.2f}").forma
t("{:.0f}",subset=["Confirmed","Deaths","Recovered","Active"])
```

Out[102]:

	Confirmed	Deaths	Recovered	Active	Mortality Rate (per 100)
0	27610660	898297	18567842	8144829	3.25

Prikaz vrijednosti atributa sortiranih po **broju potvrđenih slučajeva**.

In [103]:
```
df_latest_data["Mortality Rate (per 100)"] = np.round(100*df_latest_data["Death
s"]/df_latest_data["Confirmed"],2)
#df_latest_data.drop(columns="Last_Update",inplace=True)
df_latest_data.sort_values(["Confirmed"], ascending= False).style.background_gr
adient(cmap='Blues',subset=["Confirmed"])\
                        .background_gradient(cmap='Reds',subset=["Deaths"])\
                        .background_gradient(cmap='Greens',subset=["Recovere
d"])\
                        .background_gradient(cmap='Purples',subset=["Active"])\
                        .background_gradient(cmap='YlOrBr',subset=["Mortality R
ate (per 100)"])\
                        .format("{:.2f}")\
                        .format("{:.0f}",subset=["Confirmed","Deaths","Recovere
d","Active"])
```

Out[103]:

Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active	Mortality Rate (per 100)
US	.2f	6328154	189699	2359111	3779329	3.00
India	.2f	4370128	73890	3398844	897394	1.69
Brazil	.2f	4162073	127464	3572421	462188	3.06
Russia	.2f	1037526	18080	854069	165377	1.74
Peru	.2f	691575	29976	529751	131848	4.33
Colombia	.2f	679181	21813	541186	116234	3.21
Mexico	.2f	642860	68484	535141	39235	10.65
South Africa	.2f	640441	15086	567729	57626	2.36
Spain	.2f	534513	29594	150376	354543	5.54
Argentina	.2f	500034	10405	366590	123039	2.08
Chile	.2f	425541	11682	397730	16129	2.75
Iran	.2f	393425	22669	339111	31645	5.76
France	.2f	373718	30770	88876	254072	8.23
United Kingdom	.2f	354934	41675	1828	311431	11.74
Bangladesh	.2f	331078	4593	230804	95681	1.39
Saudi Arabia	.2f	322237	4137	298246	19854	1.28
Pakistan	.2f	299659	6359	286506	6794	2.12
Turkey	.2f	283270	6782	253245	23243	2.39
Italy	.2f	280153	35563	210801	33789	12.69
Iraq	.2f	269578	7657	206324	55597	2.84
Germany	.2f	255641	9342	228432	17852	3.65
Philippines	.2f	245143	3986	185543	55614	1.63
Indonesia	.2f	203342	8336	145200	49806	4.10
Ukraine	.2f	146511	3034	67092	76385	2.07
Israel	.2f	138719	1040	107600	30366	0.75
Canada	.2f	135757	9203	119420	7134	6.78
Bolivia	.2f	122308	7097	75098	40113	5.80
Qatar	.2f	120579	205	117497	2877	0.17
Ecuador	.2f	110757	10627	91242	8888	9.59
Kazakhstan	.2f	106498	1634	100042	4822	1.53
Egypt	.2f	100228	5560	79886	14782	5.55
Dominican Republic	.2f	100131	1889	73795	24447	1.89

Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active	Mortality Rate (per 100)
Panama	.2f	98407	2107	70751	25549	2.14
Romania	.2f	98304	4018	41002	53284	4.09
Kuwait	.2f	92082	552	82222	9308	0.60
China	.2f	90087	4733	84932	422	5.25
Belgium	.2f	89141	9912	18602	60627	11.12
Oman	.2f	87939	751	83115	4073	0.85
Sweden	.2f	85880	5842	0	80038	6.80
Netherlands	.2f	80932	6281	1669	72982	7.76
Guatemala	.2f	78721	2890	67462	8369	3.67
United Arab Emirates	.2f	75981	393	67359	8229	0.52
Morocco	.2f	75721	1427	57239	17055	1.88
Japan	.2f	73262	1411	62936	8914	1.93
Belarus	.2f	73208	721	71916	571	0.98
Poland	.2f	71947	2147	57135	12665	2.98
Honduras	.2f	65218	2034	14273	48911	3.12
Portugal	.2f	60895	1846	43146	15903	3.03
Ethiopia	.2f	60784	949	22677	37158	1.56
Singapore	.2f	57166	27	56461	678	0.05
Bahrain	.2f	56778	202	51574	5002	0.36
Venezuela	.2f	55563	444	44435	10684	0.80
Nigeria	.2f	55456	1067	43334	11055	1.92
Costa Rica	.2f	49897	531	19285	30081	1.06
Nepal	.2f	49219	312	33882	15025	0.63
Algeria	.2f	46938	1571	33183	12184	3.35
Switzerland	.2f	45306	2018	37700	5588	4.45
Ghana	.2f	45188	283	44042	863	0.63
Armenia	.2f	45152	905	41023	3224	2.00
Kyrgyzstan	.2f	44613	1061	40336	3216	2.38
Uzbekistan	.2f	44557	362	41898	2297	0.81
Moldova	.2f	40556	1087	28578	10891	2.68
Afghanistan	.2f	38544	1420	31048	6076	3.68
Azerbaijan	.2f	37557	552	34965	2040	1.47
Kenya	.2f	35356	599	21483	13274	1.69
Serbia	.2f	31994	727	0	31267	2.27
Austria	.2f	30583	747	25764	4072	2.44
Ireland	.2f	30080	1778	23364	4938	5.91
Czechia	.2f	29877	441	20164	9272	1.48
West Bank and Gaza	.2f	27919	192	18466	9261	0.69
El Salvador	.2f	26602	770	16786	9046	2.89
Australia	.2f	26465	781	22861	2823	2.95
Paraguay	.2f	24214	463	11920	11831	1.91
Bosnia and Herzegovina	.2f	21961	669	15172	6120	3.05

Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active	Mortality Rate (per 100)
Korea, South	.2f	21588	344	17023	4221	1.59
Lebanon	.2f	21324	207	6722	14395	0.97
Cameroon	.2f	19848	415	18448	985	2.09
Libya	.2f	19583	314	2247	17022	1.60
Denmark	.2f	18784	628	16330	1826	3.34
Cote d'Ivoire	.2f	18778	119	17688	971	0.63
Bulgaria	.2f	17313	692	12297	4324	4.00
Madagascar	.2f	15520	206	14243	1071	1.33
North Macedonia	.2f	15226	631	12700	1895	4.14
Senegal	.2f	14102	293	10176	3633	2.08
Sudan	.2f	13437	833	6730	5874	6.20
Zambia	.2f	12952	297	11787	868	2.29
Kosovo	.2f	12683	488	8788	3407	3.85
Croatia	.2f	12626	206	9833	2587	1.63
Greece	.2f	11832	290	1347	10195	2.45
Norway	.2f	11623	264	9348	2011	2.27
Albania	.2f	10553	321	6239	3993	3.04
Congo (Kinshasa)	.2f	10292	260	9501	531	2.53
Guinea	.2f	9848	63	9009	776	0.64
Hungary	.2f	9715	628	3984	5103	6.46
Malaysia	.2f	9583	128	9143	312	1.34
Namibia	.2f	8928	91	3981	4856	1.02
Tajikistan	.2f	8860	70	7650	1140	0.79
Maldives	.2f	8741	29	6157	2555	0.33
Gabon	.2f	8608	53	7533	1022	0.62
Finland	.2f	8430	337	7350	743	4.00
Haiti	.2f	8376	214	5991	2171	2.55
Zimbabwe	.2f	7388	218	5477	1693	2.95
Mauritania	.2f	7165	160	6681	324	2.23
Luxembourg	.2f	6974	124	6266	584	1.78
Montenegro	.2f	5875	112	4312	1451	1.91
Malawi	.2f	5630	176	3630	1824	3.13
Tunisia	.2f	5417	96	1862	3459	1.77
Djibouti	.2f	5388	61	5327	0	1.13
Equatorial Guinea	.2f	4985	83	4454	448	1.66
Eswatini	.2f	4904	96	4059	749	1.96
Congo (Brazzaville)	.2f	4891	114	3887	890	2.33
Slovakia	.2f	4888	37	2947	1904	0.76
Nicaragua	.2f	4818	144	2913	1761	2.99
Central African Republic	.2f	4735	62	1825	2848	1.31
Mozambique	.2f	4647	28	2715	1904	0.60
Rwanda	.2f	4439	20	2307	2112	0.45

Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active	Mortality Rate (per 100)
Suriname	.2f	4419	91	3595	733	2.06
Cabo Verde	.2f	4400	42	3851	507	0.95
Cuba	.2f	4377	104	3700	573	2.38
Uganda	.2f	4101	46	1876	2179	1.12
Thailand	.2f	3447	58	3286	103	1.68
Somalia	.2f	3371	97	2738	536	2.88
Jamaica	.2f	3323	36	992	2295	1.08
Slovenia	.2f	3312	135	2587	590	4.08
Gambia	.2f	3293	99	1460	1734	3.01
Syria	.2f	3289	140	760	2389	4.26
Lithuania	.2f	3163	86	2008	1069	2.72
Sri Lanka	.2f	3140	12	2946	182	0.38
Angola	.2f	3033	124	1215	1694	4.09
Mali	.2f	2882	127	2258	497	4.41
Bahamas	.2f	2657	63	1088	1506	2.37
Estonia	.2f	2585	64	2213	308	2.48
Jordan	.2f	2581	19	1885	677	0.74
South Sudan	.2f	2552	49	1290	1213	1.92
Trinidad and Tobago	.2f	2391	39	743	1609	1.63
Guinea-Bissau	.2f	2245	38	1127	1080	1.69
Benin	.2f	2213	40	1793	380	1.81
Malta	.2f	2162	14	1760	388	0.65
Iceland	.2f	2153	10	2067	76	0.46
Botswana	.2f	2126	9	493	1624	0.42
Sierra Leone	.2f	2064	72	1613	379	3.49
Yemen	.2f	1994	576	1203	215	28.89
Burma	.2f	1807	12	460	1335	0.66
New Zealand	.2f	1788	24	1639	125	1.34
Georgia	.2f	1773	19	1325	429	1.07
Uruguay	.2f	1712	45	1476	191	2.63
Guyana	.2f	1613	48	1030	535	2.98
Togo	.2f	1513	34	1127	352	2.25
Cyprus	.2f	1511	22	1237	252	1.46
Burkina Faso	.2f	1466	56	1112	298	3.82
Latvia	.2f	1443	35	1234	174	2.43
Belize	.2f	1361	16	321	1024	1.18
Liberia	.2f	1311	82	1194	35	6.25
Andorra	.2f	1261	53	934	274	4.20
Niger	.2f	1178	69	1099	10	5.86
Lesotho	.2f	1148	31	528	589	2.70
Vietnam	.2f	1059	35	890	134	3.31
Chad	.2f	1045	79	927	39	7.56

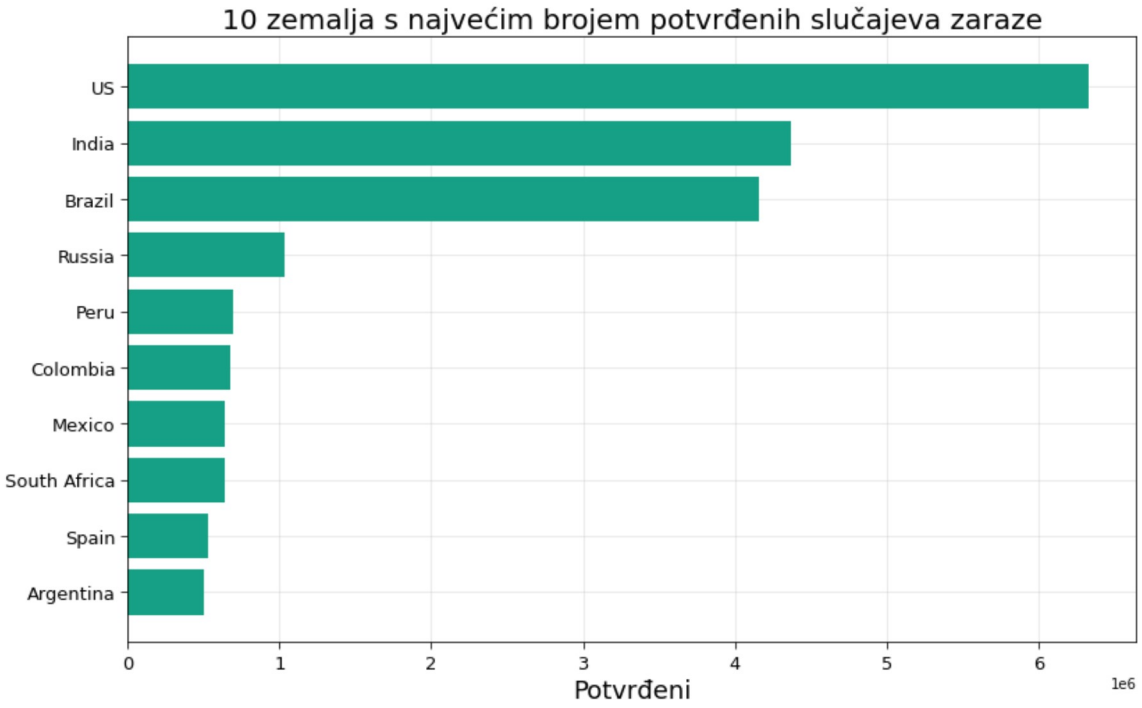
Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active	Mortality Rate (per 100)
Sao Tome and Principe	.2f	898	15	859	24	1.67
San Marino	.2f	716	42	660	14	5.87
Diamond Princess	.2f	712	13	651	48	1.83
Tanzania	.2f	509	21	183	305	4.13
Papua New Guinea	.2f	503	5	232	266	0.99
Taiwan*	.2f	495	7	475	13	1.41
Burundi	.2f	466	1	374	91	0.21
Comoros	.2f	456	7	415	34	1.54
Mauritius	.2f	361	10	335	16	2.77

1.4.1. Grafičke vizualizacije

GRAF 1.1.

Grafička vizualizacija koja prikazuje 10 država s najvećim brojem potvrđenih slučajeva:

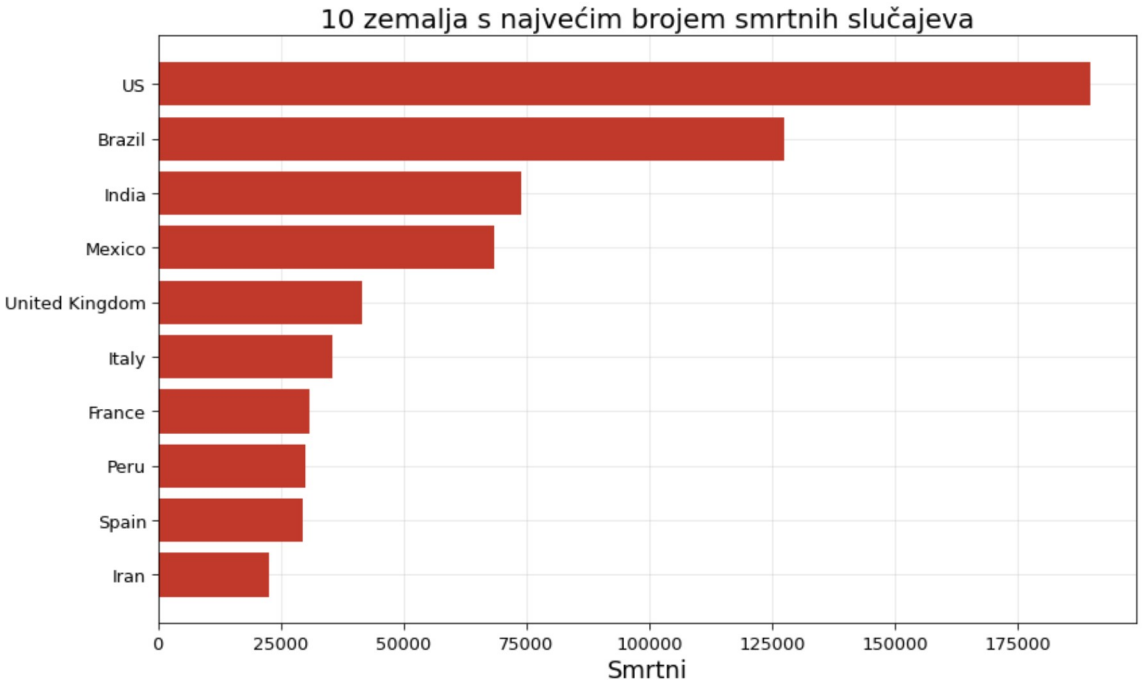
In [104]:
```
f = plt.figure(figsize=(13,8))
f.add_subplot(1,1,1)

plt.axes(axisbelow=True) #grid osi ispod grafikona
plt.barh(df_latest_data.sort_values('Confirmed')["Confirmed"].index[-10:],df_la
test_data.sort_values('Confirmed')["Confirmed"].values[-10:],color="#16a085")
plt.tick_params(size=5,labelsize = 13) #veličina labela kontrolnih točaka
plt.xlabel("Potvrđeni",fontsize=18)
plt.title("10 zemalja s najvećim brojem potvrđenih slučajeva zaraze",fontsize=2
0)
plt.grid(alpha=0.3,which='both')
```



GRAF 1.2

Grafička vizualizacija koja prikazuje 10 država s najvećim brojem smrtnih slučajeva:

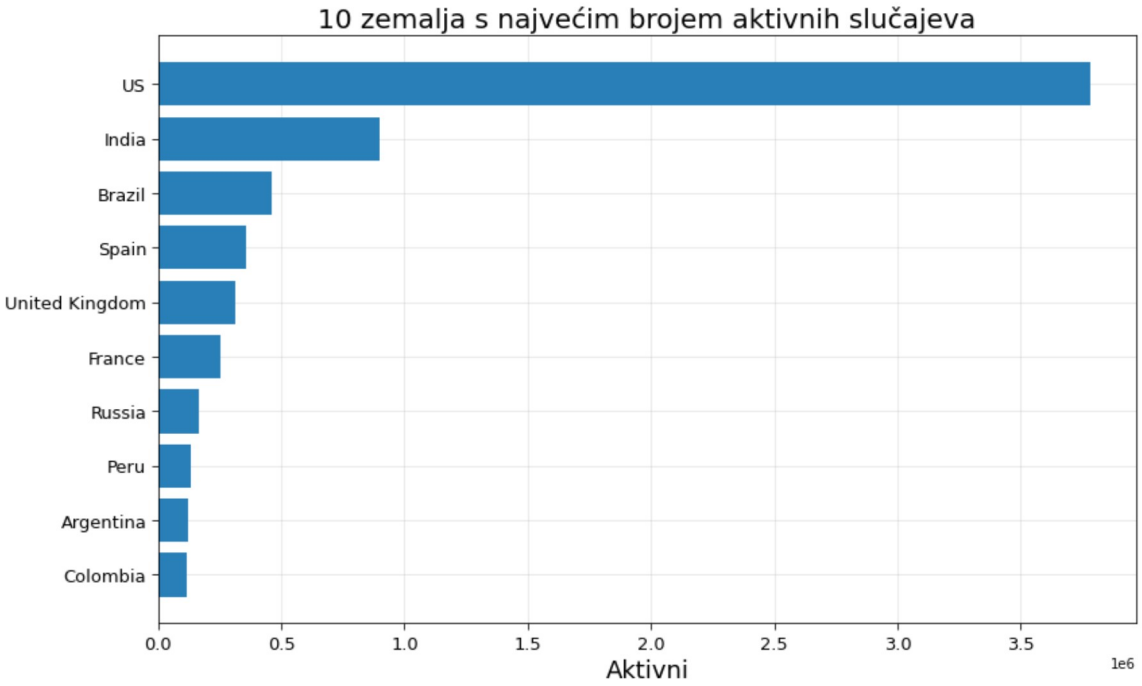
In [105]:
```
f = plt.figure(figsize=(13,8))
f.add_subplot(1,1,1)

plt.axes(axisbelow=True)
plt.barh(df_latest_data.sort_values('Deaths')["Deaths"].index[-10:],df_latest_d
ata.sort_values('Deaths')["Deaths"].values[-10:],color="#c0392b")
plt.tick_params(size=5,labelsize = 13)
plt.xlabel("Smrtni",fontsize=18)
plt.title("10 zemalja s najvećim brojem smrtnih slučajeva",fontsize=20)
plt.grid(alpha=0.3,which='both')
```



GRAF 1.3.

Grafička vizualizacija koja prikazuje 10 država s najvećim brojem aktivnih slučajeva:

In [106]:
```
f = plt.figure(figsize=(13,8))
f.add_subplot(1,1,1)

plt.axes(axisbelow=True)
plt.barh(df_latest_data.sort_values('Active')["Active"].index[-10:],df_latest_d
ata.sort_values('Active')["Active"].values[-10:],color="#2980b9")
plt.tick_params(size=5,labelsize = 13)
plt.xlabel("Aktivni",fontsize=18)
plt.title("10 zemalja s najvećim brojem aktivnih slučajeva",fontsize=20)
plt.grid(alpha=0.3,which='both')
```



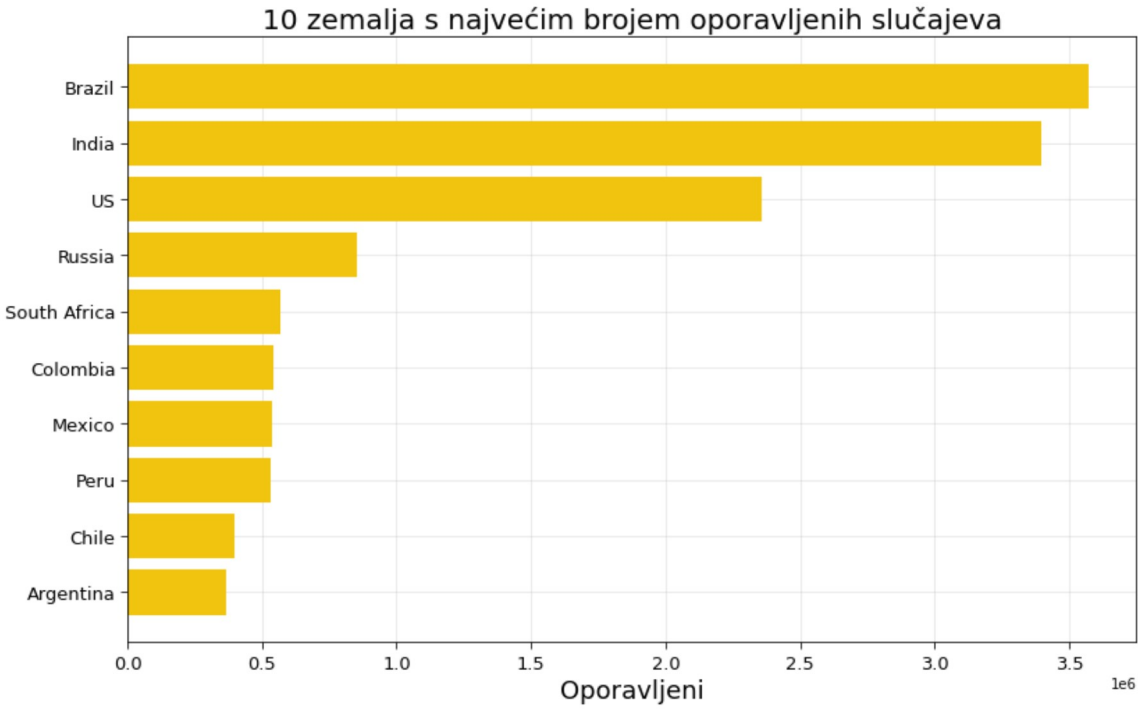
GRAF 1.4

Grafička vizualizacija koja prikazuje top 10 država s najvećim brojem oporavljenih slučajeva:

In [107]:
```
df_latest_data.sort_values('Recovered')["Recovered"].index[-10:]
```

Out[107]:
```
Index(['Argentina', 'Chile', 'Peru', 'Mexico', 'Colombia', 'South Africa',
       'Russia', 'US', 'India', 'Brazil'],
      dtype='object', name='Country_Region')
```

In [108]:
```
f = plt.figure(figsize=(13,8))
f.add_subplot(1,1,1)

plt.axes(axisbelow=True)
plt.barh(df_latest_data.sort_values('Recovered')["Recovered"].index[-10:],df_la
test_data.sort_values('Recovered')["Recovered"].values[-10:],color="#f1c40f")
plt.tick_params(size=5,labelsize = 13)
plt.xlabel("Oporavljeni",fontsize=18)
plt.title("10 zemalja s najvećim brojem oporavljenih slučajeva",fontsize=20)
plt.grid(alpha=0.3,which='both')
```



GRAF 1.5

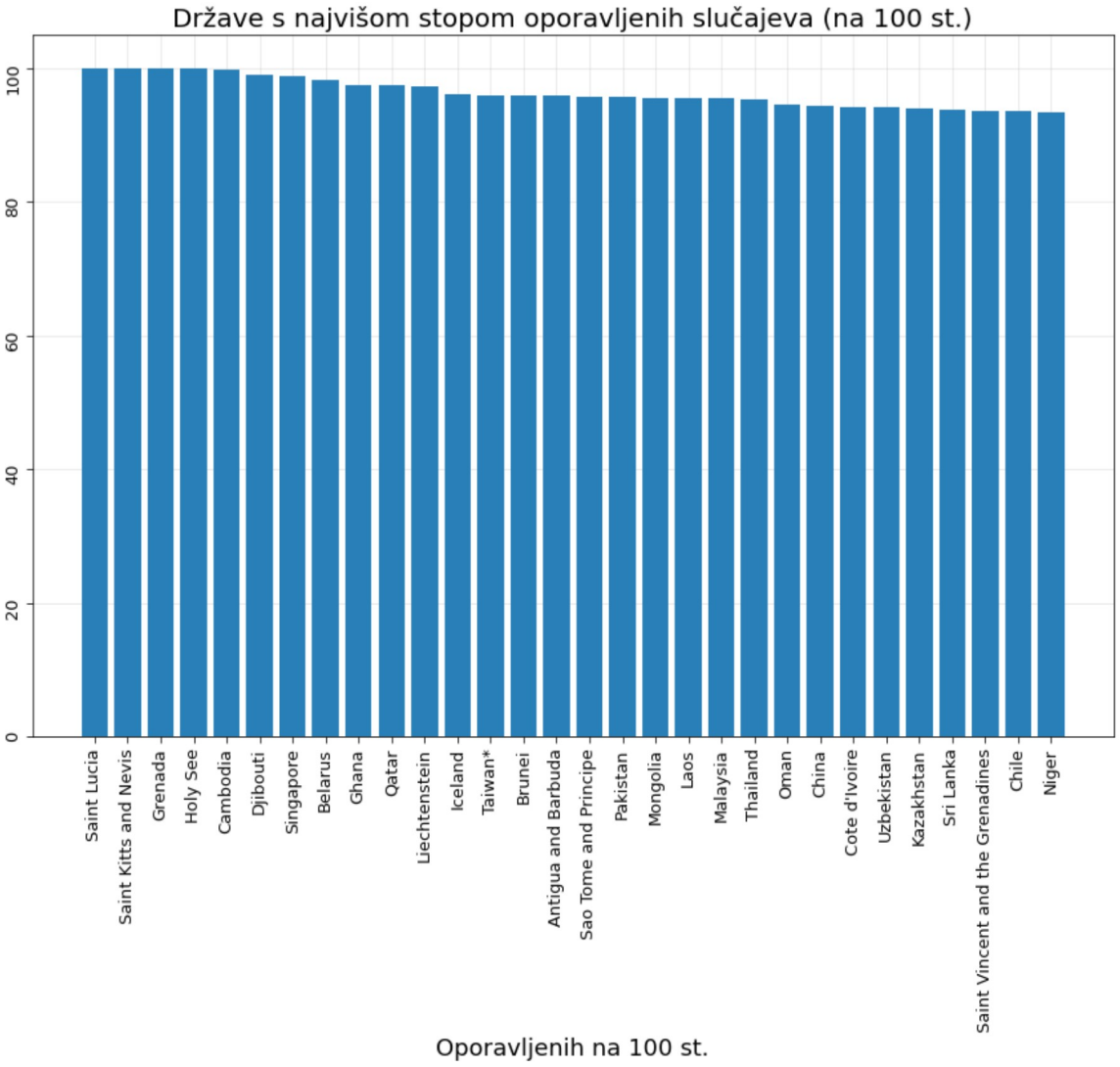
Grafička vizualizacija koja prikazuje 30 država s najvišom stopom oporavljenih slučajeva:

In [109]:
```
recov_rate = (100*df_latest_data["Recovered"]/df_latest_data["Confirmed"]).sort
_values(ascending=False).head(30)
pd.DataFrame(recov_rate).style.background_gradient(cmap='Blues')
```

Out[109]:

Country_Region	0
Saint Lucia	100.000000
Saint Kitts and Nevis	100.000000
Grenada	100.000000
Holy See	100.000000
Cambodia	99.635036
Djibouti	98.867854
Singapore	98.766749
Belarus	98.235166
Ghana	97.463928
Qatar	97.443999
Liechtenstein	97.222222
Iceland	96.005574
Taiwan*	95.959596
Brunei	95.862069
Antigua and Barbuda	95.789474
Sao Tome and Principe	95.657016
Pakistan	95.610677
Mongolia	95.483871
Laos	95.454545
Malaysia	95.408536
Thailand	95.329272
Oman	94.514379
China	94.277754
Cote d'Ivoire	94.195335
Uzbekistan	94.032363
Kazakhstan	93.937914
Sri Lanka	93.821656
Saint Vincent and the Grenadines	93.548387
Chile	93.464555
Niger	93.293718

In [110]:
```
f = plt.figure(figsize=(15,10))
f.add_subplot(1,1,1)

plt.axes(axisbelow=True)
plt.bar(recov_rate.index,recov_rate.values,color="#2980b9")
plt.tick_params(size=5,labelsize = 13, rotation=90)
plt.xlabel("Oporavljenih na 100 st.",fontsize=18)
plt.title("Države s najvišom stopom oporavljenih slučajeva (na 100 st.)",fontsi
ze=20)
plt.grid(alpha=0.3,which='both')
```



2. SKUP PODATAKA #2

Uvoz skupa podataka

In [111]:
```
# Skup podataka #2
df_countries_dates = pd.read_csv("https://raw.githubusercontent.com/CSSEGISandD
ata/COVID-19/web-data/data/cases_time.csv",low_memory=False)
```

2.1. Opis i analiza formata skupa #1

U varijablu `df_countries_dates` , odnosno u pandas data frame uvezeni su podaci iz `.csv` datoteke `cases_time.csv` . Navedeni skup podataka sastoji se od **55596+ redova** što je varijabilno budući da se broj povećava svakim danom nakon ažuriranja podataka te (inicijalno) **17 stupaca**. Data frame sadrži podatke o vrijednostima najrelevantnijih atributa po svakoj promatranoj državi od početka mjerenja, dakle 22. siječnja 2020. godine pa nadalje. Od inicijalnih atributa, ovaj data frame sadrži:

- `Country_Region` - država ili regija o kojoj se vrijednosti atributa bilježe
- `Last_Update` - obzirom da je ovo skup podataka koji se svakodnevno ažurira unutar CSSE Centra u Baltimoreu, postoji atribut koji prestavlja timestamp kada su podaci posljednji put ažurirani.
- `Confirmed` - broj potvrđenh slučaja zaraze
- `Deaths` - broj potvrđenih smrtnih slučaja uzrokovanih bolešću COVID19
- `Recovered` - broj slučajeva oporavka oboljelih
- `Active` - broj aktivnih slučajeva (dobije se tako da se od broja potvrđenih oduzme broj umrlih i oporavljenih)
- `Incident rate` - incidencija slučajeva na 100 000 osoba
- `People_Tested` - broj ljudi podvrgnutih testiranju na COVID19
- `People_Hospitalized` - broj ljudi hospitaliziranih uslijed zaraze
- `UID` - identifikator svakog retka
- `ISO3` - službeni identifikator države

i ostalih stupaca koji nisu relevantni u daljnjoj obradi.

In [112]:
```
df_countries_dates
```

Out[112]:

	Country_Region	Last_Update	Confirmed	Deaths	Recovered	Active	Delta_Confirmed	Delta_Recov
0	Afghanistan	1/22/20	0	0	NaN	NaN	0.0	
1	Afghanistan	1/23/20	0	0	NaN	NaN	0.0	
2	Afghanistan	1/24/20	0	0	NaN	NaN	0.0	
3	Afghanistan	1/25/20	0	0	NaN	NaN	0.0	
4	Afghanistan	1/26/20	0	0	NaN	NaN	0.0	
...	...	...	...	...	...	...	...	
56821	US	9/4/20	3990	42	NaN	NaN	49.0	
56822	US	9/5/20	3990	42	NaN	NaN	0.0	
56823	US	9/6/20	4032	42	NaN	NaN	42.0	
56824	US	9/7/20	4032	42	NaN	NaN	0.0	
56825	US	9/8/20	4105	42	NaN	NaN	73.0	

56826 rows × 17 columns

2.2. Tipovi podataka i potrošnja resursa

U nastavku vidimo prikaz informacija o data frameu. Kao što je ranije navedeno, radi se o trenutno 55350 redaka i 17 stupaca što možemo vidjeti i pomoću atributa `shape` u nastavku:

In [113]:
```
df_countries_dates.shape
```

Out[113]:
```
(56826, 17)
```

In [114]:
```
df_countries_dates.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56826 entries, 0 to 56825
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Country_Region       56826 non-null  object 
 1   Last_Update          56826 non-null  object 
 2   Confirmed            56826 non-null  int64  
 3   Deaths               56826 non-null  int64  
 4   Recovered            0 non-null      float64
 5   Active               0 non-null      float64
 6   Delta_Confirmed      56746 non-null  float64
 7   Delta_Recovered      0 non-null      float64
 8   Incident_Rate        55671 non-null  float64
 9   People_Tested        0 non-null      float64
 10  People_Hospitalized  0 non-null      float64
 11  Province_State       13398 non-null  object 
 12  FIPS                 13398 non-null  float64
 13  UID                  56826 non-null  int64  
 14  iso3                 56364 non-null  object 
 15  Report_Date_String   56826 non-null  object 
 16  Delta_Deaths         56712 non-null  float64
dtypes: float64(9), int64(3), object(5)
memory usage: 7.4+ MB
```

Nadalje, možemo vidjeti kako je tip podatka pretežno decimalni tip `float64` , zatim `object` kao tekstualni tip u pandasu te `int64` . Sami tip `object` bilo bi dobro izbjegavati kad god je to moguće te podatke pretvoriti u originalni, egzaktan tip podatka kao što su primjerice int ili datetime. Data frame, odnosno podaci zauzimaju **7.2+ MB** što će u nastavku biti optimizirano za potrebe obrade.

In [115]:
```
# Metoda memory_usage() vraća memorijsku veličinu svakog stupca u bajtovima --> ovdje smo dijelili
# s 1e6 kako bismo dobili megabajte
df_countries_dates.memory_usage()/1000000
```

Out[115]:
```
Index                0.000128
Country_Region       0.454608
Last_Update          0.454608
Confirmed            0.454608
Deaths               0.454608
Recovered            0.454608
Active               0.454608
Delta_Confirmed      0.454608
Delta_Recovered      0.454608
Incident_Rate        0.454608
People_Tested        0.454608
People_Hospitalized  0.454608
Province_State       0.454608
FIPS                 0.454608
UID                  0.454608
iso3                 0.454608
Report_Date_String   0.454608
Delta_Deaths         0.454608
dtype: float64
```

2.3. Redukcija i optimizacija skupa #2

Učinimo redukciju i optimizaciju skupa podataka. U nastavku ćemo izbaciti nepotrebne stupe te optimirati data frame na razini tipova podataka. Kao što smo ranije vidjeli, metodom info dolazimo do informacije koja govori kako data frame zauzima oko **20.7+ KB** memorijskog prostora.

In [116]:
```
df_countries_dates.drop(columns=["Delta_Confirmed","Delta_Recovered","Incident_
Rate","People_Tested","People_Hospitalized","iso3","UID","Province_State","FIP
S","Report_Date_String","Delta_Deaths","Active","Recovered"],inplace=True)
```

In [117]:
```
df_countries_dates["Last_Update"] = pd.to_datetime(df_countries_dates["Last_Upd
ate"])
```

In [118]:
```
df_countries_dates.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56826 entries, 0 to 56825
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   Country_Region  56826 non-null  object        
 1   Last_Update     56826 non-null  datetime64[ns]
 2   Confirmed       56826 non-null  int64         
 3   Deaths          56826 non-null  int64         
dtypes: datetime64[ns](1), int64(2), object(1)
memory usage: 1.7+ MB
```

Naime, samim izbacivanjem nekoliko stupaca iz ovoga data frame, zauzeće memorije znatno se smanjilo na **1.7+ MB**.

2.4. Modeliranje skupa podataka

Analiza i obrada

GRAF 2.1.

Usporedba kretanja broja potvrđenih slučajeva država regije

In [119]:
```
# Podaci o potvrđenim slučajevima u Hrvatskoj počevši s 22. siječnja
df_croatia = df_countries_dates[df_countries_dates["Country_Region"] == "Croati
a"]
df_croatia
```

Out[119]:

	Country_Region	Last_Update	Confirmed	Deaths
9933	Croatia	2020-01-22	0	0
9934	Croatia	2020-01-23	0	0
9935	Croatia	2020-01-24	0	0
9936	Croatia	2020-01-25	0	0
9937	Croatia	2020-01-26	0	0
...	...	...	...	...
10159	Croatia	2020-09-04	11428	195
10160	Croatia	2020-09-05	11739	197
10161	Croatia	2020-09-06	11964	198
10162	Croatia	2020-09-07	12081	201
10163	Croatia	2020-09-08	12285	203

231 rows × 4 columns

In [120]:
```
# Podaci o potvrđenim slučajevima u Srbiji počevši s 22. siječnja
df_srb = df_countries_dates[df_countries_dates["Country_Region"] == "Serbia"]
df_srb
```

Out[120]:

	Country_Region	Last_Update	Confirmed	Deaths
34419	Serbia	2020-01-22	0	0
34420	Serbia	2020-01-23	0	0
34421	Serbia	2020-01-24	0	0
34422	Serbia	2020-01-25	0	0
34423	Serbia	2020-01-26	0	0
...	...	...	...	...
34645	Serbia	2020-09-04	31772	721
34646	Serbia	2020-09-05	31849	723
34647	Serbia	2020-09-06	31905	724
34648	Serbia	2020-09-07	31941	725
34649	Serbia	2020-09-08	31994	727

231 rows × 4 columns

In [121]:
```
# Podaci o potvrđenim slučajevima u Bosni i Hercegovini počevši s 22. siječnja
df_bih = df_countries_dates[df_countries_dates["Country_Region"] == "Bosnia and Herzegovina"]
df_bih
```

Out[121]:

	Country_Region	Last_Update	Confirmed	Deaths
4851	Bosnia and Herzegovina	2020-01-22	0	0
4852	Bosnia and Herzegovina	2020-01-23	0	0
4853	Bosnia and Herzegovina	2020-01-24	0	0
4854	Bosnia and Herzegovina	2020-01-25	0	0
4855	Bosnia and Herzegovina	2020-01-26	0	0
...	...	...	...	...
5077	Bosnia and Herzegovina	2020-09-04	21142	639
5078	Bosnia and Herzegovina	2020-09-05	21439	651
5079	Bosnia and Herzegovina	2020-09-06	21560	655
5080	Bosnia and Herzegovina	2020-09-07	21660	664
5081	Bosnia and Herzegovina	2020-09-08	21961	669

231 rows × 4 columns

In [122]:
```
# Podaci o potvrđenim slučajevima u Crnoj Gori počevši s 22. siječnja
df_mne = df_countries_dates[df_countries_dates["Country_Region"] == "Montenegro"]
df_mne
```

Out[122]:

	Country_Region	Last_Update	Confirmed	Deaths
27027	Montenegro	2020-01-22	0	0
27028	Montenegro	2020-01-23	0	0
27029	Montenegro	2020-01-24	0	0
27030	Montenegro	2020-01-25	0	0
27031	Montenegro	2020-01-26	0	0
...	...	...	...	...
27253	Montenegro	2020-09-04	5275	106
27254	Montenegro	2020-09-05	5422	107
27255	Montenegro	2020-09-06	5553	108
27256	Montenegro	2020-09-07	5659	109
27257	Montenegro	2020-09-08	5875	112

231 rows × 4 columns

In [123]:
```
# Podaci o potvrđenim slučajevima u Mađarskoj počevši s 22. siječnja
df_hun = df_countries_dates[df_countries_dates["Country_Region"] == "Hungary"]
df_hun
```

Out[123]:

	Country_Region	Last_Update	Confirmed	Deaths
17787	Hungary	2020-01-22	0	0
17788	Hungary	2020-01-23	0	0
17789	Hungary	2020-01-24	0	0
17790	Hungary	2020-01-25	0	0
17791	Hungary	2020-01-26	0	0
...	...	...	...	...
18013	Hungary	2020-09-04	7382	621
18014	Hungary	2020-09-05	7892	624
18015	Hungary	2020-09-06	8387	624
18016	Hungary	2020-09-07	8963	625
18017	Hungary	2020-09-08	9304	626

231 rows × 4 columns

In [124]:
```
# Podaci o potvrđenim slučajevima u Sloveniji počevši s 22. siječnja
df_slo = df_countries_dates[df_countries_dates["Country_Region"] == "Slovenia"]
df_slo
```

Out[124]:

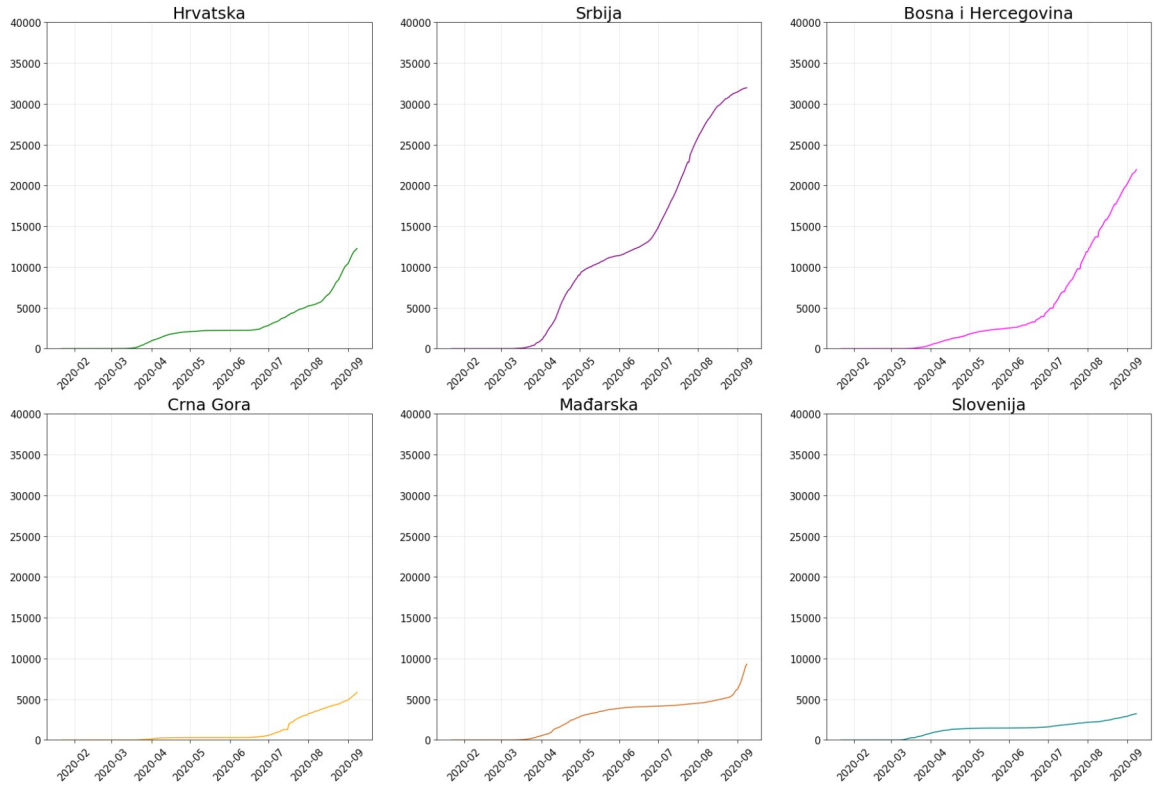
	Country_Region	Last_Update	Confirmed	Deaths
35574	Slovenia	2020-01-22	0	0
35575	Slovenia	2020-01-23	0	0
35576	Slovenia	2020-01-24	0	0
35577	Slovenia	2020-01-25	0	0
35578	Slovenia	2020-01-26	0	0
...	...	...	...	...
35800	Slovenia	2020-09-04	3079	134
35801	Slovenia	2020-09-05	3122	135
35802	Slovenia	2020-09-06	3165	135
35803	Slovenia	2020-09-07	3190	135
35804	Slovenia	2020-09-08	3232	135

231 rows × 4 columns

In [125]:
```
# Kreiranje grafikona
fig = plt.figure(figsize=(30,20))

fig.suptitle('Usporedba kretanja broja potvrđenih slučajeva država regije',
             fontsize=30)

# Podgraf 1
ax1 = fig.add_subplot(231)
ax1.set_title('Hrvatska',fontsize=25)
ax1.set_ylim(0,40000)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax1.plot(df_croatia['Last_Update'],
         df_croatia["Confirmed"],
         color='green')

# Podgraf 2
ax2 = fig.add_subplot(232)
ax2.set_title('Srbija',fontsize=25)
ax2.set_ylim(0,40000)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax2.plot(df_srb['Last_Update'],
         df_srb['Confirmed'],
         color='purple')

# Podgraf 3
ax3 = fig.add_subplot(233)
ax3.set_title('Bosna i Hercegovina',fontsize=25)
ax3.set_ylim(0,40000)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax3.plot(df_bih['Last_Update'],
         df_bih['Confirmed'],
         color='magenta')

# Podgraf 4
ax4 = fig.add_subplot(234)
ax4.set_title('Crna Gora',fontsize=25)
ax4.set_ylim(0,40000)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax4.plot(df_mne['Last_Update'],
         df_mne['Confirmed'],
         color='orange')

# Podgraf 5
ax5 = fig.add_subplot(235)
ax5.set_title('Mađarska',fontsize=25)
ax5.set_ylim(0,40000)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax5.plot(df_hun["Last_Update"],
         df_hun["Confirmed"],
         color='chocolate')

# Podgraf 6
ax6 = fig.add_subplot(236)
```

```
ax6.set_title('Slovenija',fontsize=25)
ax6.set_ylim(0,40000)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax6.plot(df_slo["Last_Update"],
         df_slo["Confirmed"],
         color='teal')


plt.show()
```

Usporedba kretanja broja potvrđenih slučajeva država regije



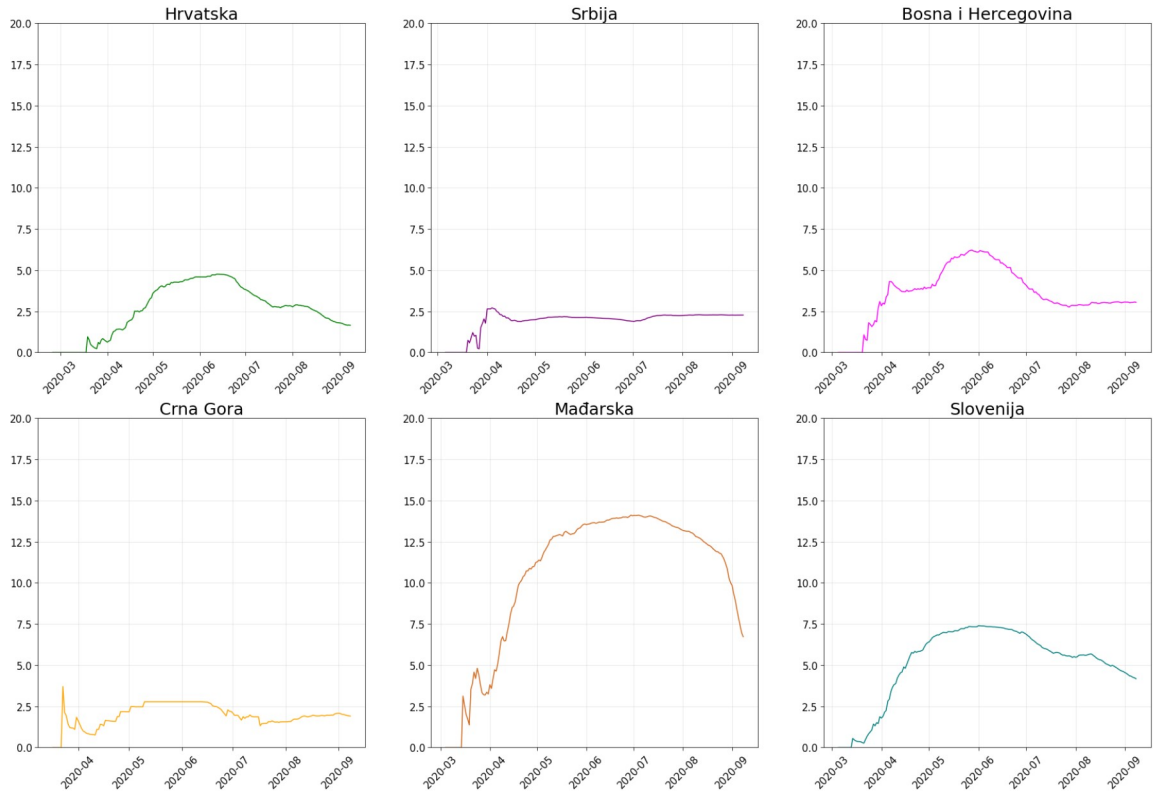
GRAF 2.2.

Usporedba kretanja stope smrtnosti država regije (na 100 potvrđenih slučajeva)

In [126]:
```
#100*df_latest_data["Deaths"]/df_latest_data["Confirmed"]
# Kreiranje grafikona
fig = plt.figure(figsize=(30,20))

fig.suptitle('Usporedba kretanja stope smrtnosti država regije (na 100 st.)',
             fontsize=30)

# Podgraf 1
ax1 = fig.add_subplot(231)
ax1.set_title('Hrvatska',fontsize=25)
ax1.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax1.plot(df_croatia['Last_Update'],
         100*df_croatia["Deaths"]/df_croatia["Confirmed"],
         color='green')

# Podgraf 2
ax2 = fig.add_subplot(232)
ax2.set_title('Srbija',fontsize=25)
ax2.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax2.plot(df_srb['Last_Update'],
         100*df_srb['Deaths']/df_srb["Confirmed"],
         color='purple')

# Podgraf 3
ax3 = fig.add_subplot(233)
ax3.set_title('Bosna i Hercegovina',fontsize=25)
ax3.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax3.plot(df_bih['Last_Update'],
         100*df_bih['Deaths']/df_bih["Confirmed"],
         color='magenta')

# Podgraf 4
ax4 = fig.add_subplot(234)
ax4.set_title('Crna Gora',fontsize=25)
ax4.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax4.plot(df_mne['Last_Update'],
         100*df_mne['Deaths']/df_mne["Confirmed"],
         color='orange')

# Podgraf 5
ax5 = fig.add_subplot(235)
ax5.set_title('Mađarska',fontsize=25)
ax5.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax5.plot(df_hun["Last_Update"],
         100*df_hun["Deaths"]/df_hun["Confirmed"],
         color='chocolate')

#Podgraf 6
```

```
ax6 = fig.add_subplot(236)
ax6.set_title('Slovenija',fontsize=25)
ax6.set_ylim(0,20)

plt.tick_params(size=5,labelsize = 15,rotation=45,axis='x')
plt.tick_params(size=5,labelsize = 15,axis='y')
plt.grid(alpha=0.3,which='both')
ax6.plot(df_slo["Last_Update"],
         100*df_slo["Deaths"]/df_slo["Confirmed"],
         color='teal')


plt.show()
```

Usporedba kretanja stope smrtnosti država regije (na 100 st.)



3. SKUP PODATAKA #3

Podaci o potvrđenim slučajevima

Uvoz skupa podataka

In [127]:
```
df_confirmed = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv')
```

3.1. Opis i analiza formata skupa #3

U varijablu df_confirmed, odnosno u pandas data frame uvezeni su podaci iz `.csv` datoteke `time_series_covid19_confirmed_global.csv` . Navedeni skup podataka sastoji se od **266** redova te (inicijalno) **230** stupaca.

Data frame sadrži podatke o vrijednostima atributa koji kvantificira potvrđene slučajeve zaraze po svakoj promatranoj državi od početka mjerenja, dakle **22. siječnja 2020.** godine pa nadalje. Dakle, retci data framea sačinjeni su od promatranih država, a stupci od datuma počevši od **22. siječnja .2020**.

In [128]:
```
df_confirmed
```

Out[128]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27
0	NaN	Afghanistan	33.939110	67.709953	0	0	0	0	0	
1	NaN	Albania	41.153300	20.168300	0	0	0	0	0	
2	NaN	Algeria	28.033900	1.659600	0	0	0	0	0	
3	NaN	Andorra	42.506300	1.521800	0	0	0	0	0	
4	NaN	Angola	-11.202700	17.873900	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
261	NaN	West Bank and Gaza	31.952200	35.233200	0	0	0	0	0	
262	NaN	Western Sahara	24.215500	-12.885800	0	0	0	0	0	
263	NaN	Yemen	15.552727	48.516388	0	0	0	0	0	
264	NaN	Zambia	-13.133897	27.849332	0	0	0	0	0	
265	NaN	Zimbabwe	-19.015438	29.154857	0	0	0	0	0	

266 rows × 235 columns

In [129]:
```
# Preimenovanje stupaca
df_confirmed = df_confirmed.rename(columns={"Province/State":"State","Country/R
egion": "Country"})
df_confirmed.head(1)
```

Out[129]:

	State	Country	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	8/30/20	8
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	...	38162	

1 rows × 235 columns

3.2. Tipovi podataka i potrošnja resursa

U nastavku vidimo prikaz informacija o data frameu. Kao što je ranije navedeno, radi se o trenutno 266 redaka i 229 stupaca što možemo vidjeti i pomoću atributa `shape` u nastavku:

In [130]:
```
df_confirmed.shape
```

Out[130]:
```
(266, 235)
```

In [131]:
```
df_confirmed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Columns: 235 entries, State to 9/8/20
dtypes: float64(2), int64(231), object(2)
memory usage: 488.5+ KB
```

Kao što vidimo, data frame sadrži **266 zapisa**, odnosno redova. Isto tako sadrži i **229 stupaca**. Cijeli data frame zauzima **478+ KB** memorije u računalu.

In [132]:
```
# Metoda memory_usage() vraća memorijsku veličinu svakog stupca u bajtovima --> ovdje smo dijelili
# s 1e6 kako bismo dobili megabajte
df_confirmed.memory_usage()/1000000
```

Out[132]:
```
Index      0.000128
State      0.002128
Country    0.002128
Lat        0.002128
Long       0.002128
             ...
9/4/20     0.002128
9/5/20     0.002128
9/6/20     0.002128
9/7/20     0.002128
9/8/20     0.002128
Length: 236, dtype: float64
```

3.3. Redukcija i optimizacija skupa #3

In [133]:
```
df_confirmed.drop(columns=["Lat","Long"],inplace=True)
```

In [134]:
```
df_confirmed["State"].fillna("Not available",inplace=True)
```

In [135]:
```
df_confirmed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Columns: 233 entries, State to 9/8/20
dtypes: int64(231), object(2)
memory usage: 484.3+ KB
```

In [136]:
```
df_confirmed.head(2)
```

Out[136]:

	State	Country	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	...	8/30/20	8/3
0	Not available	Afghanistan	0	0	0	0	0	0	0	0	...	38162	3
1	Not available	Albania	0	0	0	0	0	0	0	0	...	9380	

2 rows × 233 columns

3.4. Modeliranje skupa podataka

Analiza i obrada

Prikažimo samo države koje **nemaju** dodijeljenu vrijednost atributa `Province/State`

In [137]:
```
df_confirmed[df_confirmed["State"] == "Not available"]
```

Out[137]:

	State	Country	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	...	8/30/20
0	Not available	Afghanistan	0	0	0	0	0	0	0	0	...	38162
1	Not available	Albania	0	0	0	0	0	0	0	0	...	9380
2	Not available	Algeria	0	0	0	0	0	0	0	0	...	44146
3	Not available	Andorra	0	0	0	0	0	0	0	0	...	1124
4	Not available	Angola	0	0	0	0	0	0	0	0	...	2624
...	...	...	...	...	...	...	...	...	...	...	...	...
261	Not available	West Bank and Gaza	0	0	0	0	0	0	0	0	...	22204
262	Not available	Western Sahara	0	0	0	0	0	0	0	0	...	10
263	Not available	Yemen	0	0	0	0	0	0	0	0	...	1953
264	Not available	Zambia	0	0	0	0	0	0	0	0	...	12025
265	Not available	Zimbabwe	0	0	0	0	0	0	0	0	...	6412

185 rows × 233 columns

Prikažimo samo države koje **imaju** dodijeljenu vrijednost atributa `Province/State`

In [138]:
```
df_confirmed[df_confirmed["State"] != "Not available"]
```

Out[138]:

	State	Country	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	...	8/30/20
8	Australian Capital Territory	Australia	0	0	0	0	0	0	0	0	...	113
9	New South Wales	Australia	0	0	0	0	3	4	4	4	...	4050
10	Northern Territory	Australia	0	0	0	0	0	0	0	0	...	33
11	Queensland	Australia	0	0	0	0	0	0	0	1	...	1122
12	South Australia	Australia	0	0	0	0	0	0	0	0	...	463
...	...	...	...	...	...	...	...	...	...	...	...	...
251	Falkland Islands (Malvinas)	United Kingdom	0	0	0	0	0	0	0	0	...	13
252	Gibraltar	United Kingdom	0	0	0	0	0	0	0	0	...	285
253	Isle of Man	United Kingdom	0	0	0	0	0	0	0	0	...	336
254	Montserrat	United Kingdom	0	0	0	0	0	0	0	0	...	13
255	Turks and Caicos Islands	United Kingdom	0	0	0	0	0	0	0	0	...	505

81 rows × 233 columns

Grupirajmo države i prikažimo samo sumarne podatke po svakoj državi distinktivno:

In [139]:
```
df_conf_grouped = df_confirmed.groupby("Country").sum()
df_conf_grouped
```

Out[139]:

Country	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	...	8/30/2
Afghanistan	0	0	0	0	0	0	0	0	0	0	...	3816
Albania	0	0	0	0	0	0	0	0	0	0	...	938
Algeria	0	0	0	0	0	0	0	0	0	0	...	4414
Andorra	0	0	0	0	0	0	0	0	0	0	...	112
Angola	0	0	0	0	0	0	0	0	0	0	...	262
...	...	...	...	...	...	...	...	...	...	...	...	.
West Bank and Gaza	0	0	0	0	0	0	0	0	0	0	...	2220
Western Sahara	0	0	0	0	0	0	0	0	0	0	...	1
Yemen	0	0	0	0	0	0	0	0	0	0	...	195
Zambia	0	0	0	0	0	0	0	0	0	0	...	1202
Zimbabwe	0	0	0	0	0	0	0	0	0	0	...	641

188 rows × 231 columns

GRAF 3.1

Prikažimo rezultate broja potvrđenih slučajeva za **Republiku Hrvatsku** u periodu od 22. siječnja do najnovijih podataka:

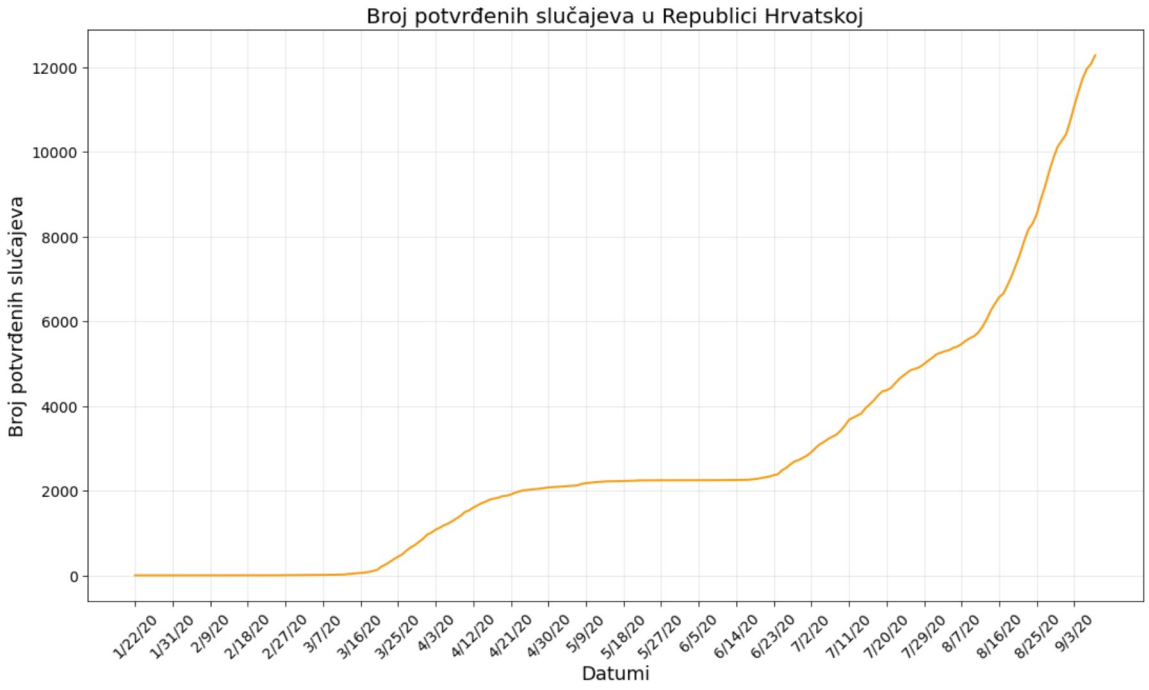
In [140]:
```
cro_T = pd.DataFrame(df_conf_grouped.loc["Croatia"])
cro_T
```

Out[140]:

	Croatia
1/22/20	0
1/23/20	0
1/24/20	0
1/25/20	0
1/26/20	0
...	...
9/4/20	11428
9/5/20	11739
9/6/20	11964
9/7/20	12081
9/8/20	12285

231 rows × 1 columns

In [141]:
```
f = plt.figure(figsize=(18,10))
f.add_subplot(111)

plt.axes(axisbelow=True)
plt.plot(cro_T.index,cro_T, c="#f39c12", linewidth=2, linestyle='-')
plt.tick_params(size=5, labelsize = 14)
plt.xticks(list(np.arange(0,len(cro_T.index),int(len(cro_T.index)/25))),rotatio
n=45)

plt.xlabel("Datumi",fontsize=18)
plt.ylabel("Broj potvrđenih slučajeva",fontsize=18)
plt.title("Broj potvrđenih slučajeva u Republici Hrvatskoj",fontsize=20)
plt.grid(alpha = 0.3,which='both')
plt.show()
```

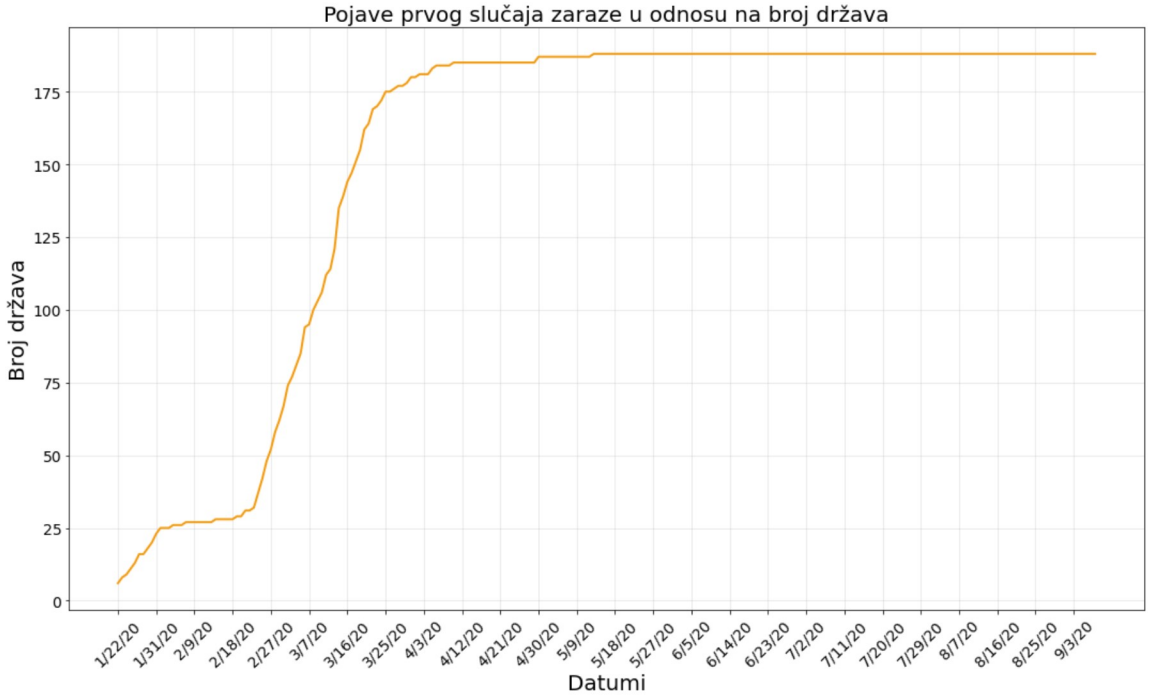


GRAF 3.2

Graf pojava prvog slučaja zaraze po državama:

In [142]:
```
affected_countries = df_confirmed.groupby("Country").sum().apply(lambda x: x[x
> 0].count(), axis =0)
dt = affected_countries.index
```

In [143]:
```
f = plt.figure(figsize=(18,10))
f.add_subplot(111)
plt.plot(dt, affected_countries, c="#f39c12",linewidth=2, linestyle='-')
plt.tick_params(labelsize = 14)
plt.xticks(list(np.arange(0,len(dt),int(len(dt)/25))), rotation=45)

#labels
plt.xlabel("Datumi",fontsize=20)
plt.ylabel("Broj država",fontsize=20)
plt.title("Pojave prvog slučaja zaraze u odnosu na broj država",fontsize=20)
plt.grid(alpha = 0.3)
```



GRAF 3.3

Dnevni prirast broja potvrđenih slučajeva po nekoliko izdvojenih država:

In [144]:
```
df_confirmed.groupby("Country").sum().diff(axis=1).sort_values(df_confirmed.col
umns[-1],ascending =False).head(10).replace(np.nan,0)
```

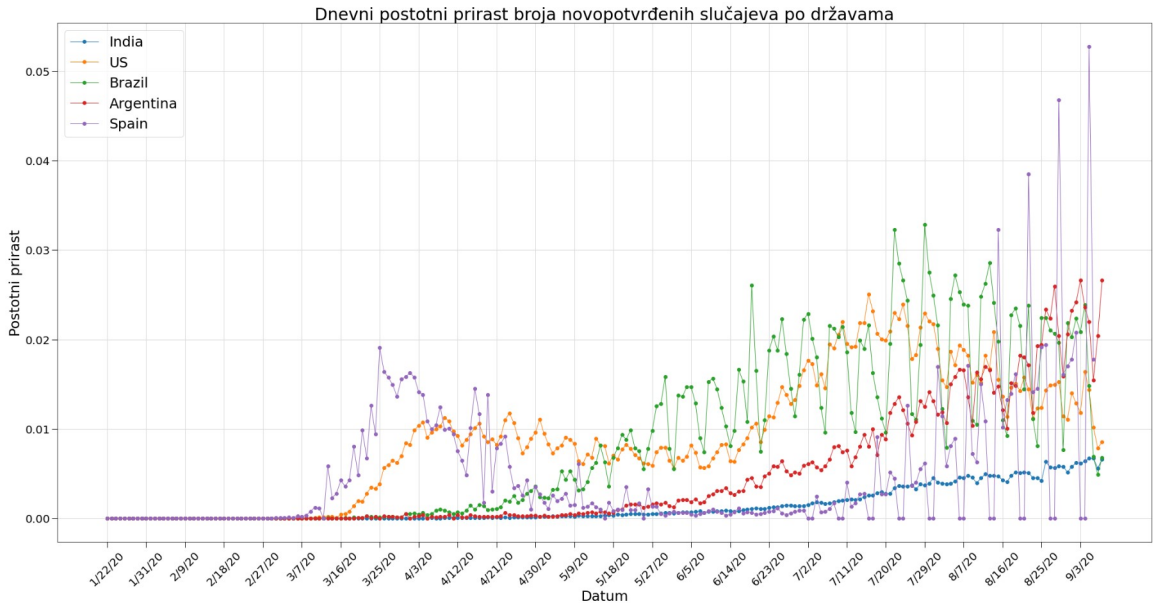
Out[144]:

Country	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	...	8/30/20
India	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	78512.0
US	0.0	0.0	1.0	0.0	3.0	0.0	0.0	0.0	0.0	2.0	...	35337.0
Brazil	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	16158.0
Argentina	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	7187.0
Spain	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
France	0.0	0.0	2.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	...	10866.0
Mexico	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	4129.0
Russia	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	...	4897.0
Iraq	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	3731.0
Israel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	555.0

10 rows × 231 columns

In [161]:
```
dfGrowth = df_confirmed.groupby('Country').sum().diff(axis='columns').sort_valu
es(df_confirmed.columns[-1],ascending =False).head(5).replace(np.nan,0)
dts = dfGrowth.columns
pop = {
  "India": 1352642280,
  "US": 308401808,
  "Brazil": 210147125,
  "Argentina":45195774,
  "Spain":50372424
}

f = plt.figure(figsize=(35,17))
ax = f.add_subplot(111)

for i,country in enumerate(dfGrowth.index):            # dfGrowth.index su drža
ve
    t = dfGrowth.loc[temp.index == country].values[0]
    t = t[t>=0]

    date = np.arange(0,len(t[:]))
    plt.plot(date,(t/pop[country])*100,'-o',label = country,linewidth =1)

plt.xticks(list(np.arange(0,len(dts),int(len(dts)/25))),dts[:-1:int(len(dts)/2
5)])

plt.tick_params(which='both', width=1,labelsize=20,length=10)
plt.xticks(rotation=45)

ax.grid(lw = 1, ls = '-', c = "0.85", which = 'major')
ax.grid(lw = 1, ls = '-', c = "0.95", which = 'minor')

plt.title("Dnevni postotni prirast broja novopotvrđenih slučajeva po državama",
fontsize=30)


plt.xlabel("Datum",fontsize =25)
plt.ylabel("Postotni prirast",fontsize =25)

plt.legend(fontsize=25)
```

Out[161]:
```
<matplotlib.legend.Legend at 0x14ad7ebdaf0>
```



4. Skup podataka #4

Podaci o smrtnim slučajevima

Uvoz skupa podataka

In [146]:
```
df_deaths = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv')
```

4.1. Opis i analiza formata skupa #4

U varijablu df_deaths, odnosno u pandas data frame uvezeni su podaci iz `.csv` datoteke `time_series_covid19_deaths_global.csv` . Navedeni skup podataka sastoji se od **266** redova te (inicijalno) **230** stupaca.

Data frame sadrži podatke o vrijednostima atributa koji kvantificira smrtne slučajeve po svakoj promatranoj državi od početka mjerenja, dakle **22. siječnja 2020.** godine pa nadalje. Dakle, retci data framea sačinjeni su od promatranih država, a stupci od datuma počevši od **22. siječnja .2020**.

In [147]:
```
df_deaths
```

Out[147]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27
0	NaN	Afghanistan	33.939110	67.709953	0	0	0	0	0	
1	NaN	Albania	41.153300	20.168300	0	0	0	0	0	
2	NaN	Algeria	28.033900	1.659600	0	0	0	0	0	
3	NaN	Andorra	42.506300	1.521800	0	0	0	0	0	
4	NaN	Angola	-11.202700	17.873900	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
261	NaN	West Bank and Gaza	31.952200	35.233200	0	0	0	0	0	
262	NaN	Western Sahara	24.215500	-12.885800	0	0	0	0	0	
263	NaN	Yemen	15.552727	48.516388	0	0	0	0	0	
264	NaN	Zambia	-13.133897	27.849332	0	0	0	0	0	
265	NaN	Zimbabwe	-19.015438	29.154857	0	0	0	0	0	

266 rows × 235 columns

In [148]:
```
df_deaths = df_deaths.rename(columns={"Province/State":"State","Country/Region": "Country"})
df_deaths.head(1)
```

Out[148]:

	State	Country	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	8/30/20	8
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	...	1402	

1 rows × 235 columns

4.2. Tipovi podataka i potrošnja resursa

U nastavku vidimo prikaz informacija o data frameu. Kao što je ranije navedeno, radi se o trenutno 266 redaka i 229 stupaca što možemo vidjeti i pomoću atributa `shape` u nastavku:

In [149]:
```
df_deaths.shape
```

Out[149]:
```
(266, 235)
```

In [150]:
```
df_deaths.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Columns: 235 entries, State to 9/8/20
dtypes: float64(2), int64(231), object(2)
memory usage: 488.5+ KB
```

Kao što vidimo, data frame sadrži **266 zapisa**, odnosno redova. Isto tako sadrži i **230 stupaca**. Cijeli data frame zauzima **478.1+ KB** memorije u računalu.

In [151]:
```
# Metoda memory_usage() vraća memorijsku veličinu svakog stupca u bajtovima --> ovdje smo dijelili
# s 1e6 kako bismo dobili megabajte
df_confirmed.memory_usage()/1000000
```

Out[151]:
```
Index      0.000128
State      0.002128
Country    0.002128
1/22/20    0.002128
1/23/20    0.002128
             ...
9/4/20     0.002128
9/5/20     0.002128
9/6/20     0.002128
9/7/20     0.002128
9/8/20     0.002128
Length: 234, dtype: float64
```

4.3. Redukcija i optimizacija skupa #4

In [152]:
```
df_deaths.drop(columns=["Lat","Long"],inplace=True)
```

In [153]:
```
df_deaths["State"].fillna("Not available",inplace=True)
```

In [154]:
```
df_deaths.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Columns: 233 entries, State to 9/8/20
dtypes: int64(231), object(2)
memory usage: 484.3+ KB
```

In [155]:
```
df_deaths.head(2)
```

Out[155]:

	State	Country	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	...	8/30/20	8/3
0	Not available	Afghanistan	0	0	0	0	0	0	0	0	...	1402	
1	Not available	Albania	0	0	0	0	0	0	0	0	...	280	

2 rows × 233 columns

4.4. Modeliranje skupa podataka

Analiza i obrada

GRAF 4.1.

Apsolutno kretanje broja potvrđenih i smrtnih slučajeva u Republici Hrvatskoj:

In [156]:
```
df_death_grouped = df_deaths.groupby("Country").sum()
df_death_grouped
```

Out[156]:

Country	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	...	8/30/2
Afghanistan	0	0	0	0	0	0	0	0	0	0	...	140
Albania	0	0	0	0	0	0	0	0	0	0	...	28
Algeria	0	0	0	0	0	0	0	0	0	0	...	150
Andorra	0	0	0	0	0	0	0	0	0	0	...	5
Angola	0	0	0	0	0	0	0	0	0	0	...	10
...	...	...	...	...	...	...	...	...	...	...	...	
West Bank and Gaza	0	0	0	0	0	0	0	0	0	0	...	15
Western Sahara	0	0	0	0	0	0	0	0	0	0	...	
Yemen	0	0	0	0	0	0	0	0	0	0	...	56
Zambia	0	0	0	0	0	0	0	0	0	0	...	28
Zimbabwe	0	0	0	0	0	0	0	0	0	0	...	19

188 rows × 231 columns

In [157]:
```
cro_deaths_T = pd.DataFrame(df_death_grouped.loc["Croatia"])
cro_deaths_T
```

Out[157]:

	Croatia
1/22/20	0
1/23/20	0
1/24/20	0
1/25/20	0
1/26/20	0
...	...
9/4/20	195
9/5/20	197
9/6/20	198
9/7/20	201
9/8/20	203

231 rows × 1 columns

In [158]:
```
f = plt.figure(figsize=(18,10))
f.add_subplot(111)

plt.axes(axisbelow=True)
plt.plot(cro_T.index,cro_T, c="#f39c12", linewidth=2, linestyle='-')
plt.plot(cro_deaths_T.index,cro_deaths_T, c="#c0392b", linewidth=1, linestyle=
'--')
plt.tick_params(size=5, labelsize = 14)
plt.xticks(list(np.arange(0,len(cro_T.index),int(len(cro_T.index)/25))),rotatio
n=45)

plt.xlabel("Datumi",fontsize=18)
plt.ylabel("Apsolutni broj potvrđenih/smrtnih",fontsize=18)
plt.title("Apsolutno kretanje broja potvrđenih i smrtnih slučajeva u Republici
Hrvatskoj",fontsize=20)
plt.grid(alpha = 0.3,which='both')
plt.show()
```