Primjena algoritama strojnog učenja za istraživanje i analizu čimbenika zadovoljstva korisnika smještaja

Čumlievski, Nola

Sveučilište u Rijeci – Odjel za informatiku

Informacijski i komunikacijski sustavi

Nola Čumlievski

Primjena algoritama strojnog učenja za istraživanje i analizu čimbenika zadovoljstva korisnika smještaja

Diplomski rad

Mentor: Prof. dr. sc. Maja Matetić

Rijeka, Srpanj 2021.

Sažetak

Ovaj rad se bavi istraživanjem i analizom podataka sa web stranice za rezervaciju smještaja „Booking“ izvedenom u programskom jeziku Python. Ideja rada jest analizirati odnos između pojedinih čimbenika smještaja i same recenzije korisnika određenog smještaja, kako bi utvrdili koji od čimbenika imaju najveći utjecaj na recenziju korisnika. Primjenom postupaka strojnog učenja izradit će se modeli strojnog učenja koji će na temelju prikupljenih podataka biti u mogućnosti predvidjeti recenziju (kategoriju) objekata koji još nisu ocjenjeni od strane samih korisnika. Cilj ovog rada jest teorijski opisati metode koje su korištene tijekom prikupljanja i analize podataka, prikazati i opisati programsko riješenje korišteno za dobivanje rezultata, omogućiti detaljan uvid u prikupljenje podatke i odnose između pojedinih istraživanih čimbenika te interpretirati dobivene rezultate.

Ključne riječi: Python, Strojno učenje, Web struganje, Podatkovna analitika, Booking

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 8. ožujak 2021.

Zadatak za diplomski rad

Pristupnik: **Nola Čumlievski**

Naziv diplomskog rada: **Primjena algoritama strojnog učenja za istraživanje i analizu čimbenika zadovoljstva korisnika smještaja**

Naziv diplomskog rada na eng. jeziku: **Application of machine learning algorithms for research and analysis of accommodation user satisfaction factors**

Sadržaj zadatka:

Dubinska analiza podataka, koja se temelji na algoritmima strojnog učenja, korisna je u otkrivanju novog znanja za unaprjeđenje procesa i poslovanja. Zadatak diplomskog rada je prikupiti, očistiti, integrirati, balansirati i analizirati podatke s područja korištenja smještaja, kako bi se utvrdilo koji faktori imaju najveći utjecaj na zadovoljstvo korisnika. Zadatak je izraditi modele strojnog učenja koji će na temelju prikupljenih podataka biti u mogućnosti predvidjeti kategoriju objekata koji još nisu ocjenjeni od strane samih korisnika, te otkrivanjem najznačajnijih prediktora dati smjernice za poboljšanje kvalitete usluge.

Mentor:
Prof. dr. sc. Maja Matetić

Voditeljica za diplomske radove:
Izv. prof. dr. sc. Ana Meštrović

Komentor:

Zadatak preuzet: 8. ožujak 2021.

Nola Čumlievski
(potpis pristupnika)

Sadržaj

1. Uvod

Tema ovog rada jest istraživanje podataka skinutih sa web stranice Booking i analiza prikupljenih podataka pojedinih objekata smještaja sa web stranice primjenom postupaka podatkovne analize izvedenim u programskom jeziku Python, kako bi se utvrdilo koji čimbenici (lokacija, cijena, dodatni sadržaji, pogodnosti i druge karakteristike pojedinih smještaja) imaju najveći utjecaj na zadovoljstvo korisnika, te izraditi model strojnog učenja koji će učinkovito predvidjeti kategoriju (recenziju) smještaja s obzirom na karakteristike samog smještaja. Svako poglavlje sadrži opis i rezultate određenog postupka primijenjenog tijekom navedene analize. Drugo poglavlje sadrži navod i opis svih koraka prilikom prikupljanja podataka (web struganje). Treće poglavlje fokusirano je na čišćenje samih podataka, gdje su opisane metode čišćenja navedenih podataka i konačan skup podataka koji će se koristiti za daljnju analizu. Četvrto poglavlje opisuje što je istraživačka analiza podataka (EDA), za što se koristi te sadrži rezultate primjene metoda istraživačke analize podataka nad podacima s Bookinga. Unutar petog i šestog poglavlja opisan je postupak prikupljanja i analize tekstova recenzija te su opisani algoritmi strojnog učenja primijenjeni u analizi skupa podataka sa dobivenim rezultatima.

2. Web struganje podataka

Web struganje (engl. *Web scraping*) predstavlja proces automatskog prikupljanja podataka s interneta pomoću automatiziranog programa koji pregledava određene web stranice te izdvaja informacije koje su potrebne korisniku. S obzirom da se danas većina podataka nalazi upravo na internetu, web struganje postalo je česta metoda prikupljanja podataka upravo zato jer omogućava spremanje prikupljenih podataka sa interneta u baze podataka [1].

Postupak web struganja izgleda tako da program za pretraživanje weba dohvaća HTML (engl. *HyperText Markup Language*) sa dobivene domene, parsira HTML stranice kako bi došao do željenih informacija te pohranjuje željene informacije po korisnikovoj želji [1]. Python raspolaže sa više modula koji se mogu koristiti za web struganje, te su u sklopu ovog rada korištene dvije biblioteke za struganje podataka, *BeautifulSoup* i *Selenium.*

Kako bi mogli koristiti *Selenium* biblioteku, potrebno je instalirati web driver za preglednik koji se planira koristiti za web struganje podataka. Web driver moguće je skinuti sa slijedeće stranice https://sites.google.com/a/chromium.org/chromedriver/downloads (za Google Chrome). Driver se bira prema verziji internet preglednika, koju je u Chrome pregledniku moguće provjeriti tako da odemo na izbornik koji se nalazi u gornjem desnom kutu preglednika (označen sa tri točke) i pod izbornikom „Pomoć“ odaberemo „O Google Chromu“ gdje možemo vidjeti koja je verzija internet preglednika instalirana na našem računalu.

Za web struganje podataka sa Bookinga odabrane su navedene dvije biblioteke iz razloga što je modul *BeautifulSoup* pogodniji za parsiranje statičkog sadržaja (HTML-a) no neke informacije ne bi uspio povući sa internet stranice (konkretno poveznice na slijedeću stranicu kod rezultata pretrage i poveznice na stranicu pojedinih hotela) te je *Selenium* iskorišten za dohvaćanje samog HTML-a stranica i ekstrakciju potrebnih poveznica, dok je *BeautifulSoup* korišten za parsiranje stranica objekata smještaja gdje je bilo potrebno dohvatiti većinom tekstualni sadržaj o samom objektu (ime, cijenu, lokaciju, recenziju, itd).

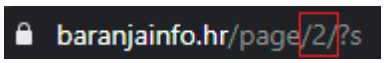
2.1. Plan web struganja

Prvotna ideja bila je filtracija objekata prema državi (Hrvatska) i web struganje svih rezultata dobivenih navedenom jednom pretragom. Nakon unosa navedenog filtera vidljiva je informacija da je dostupno preko 80 000 rezultata no prikazano ih je samo 1 000 (40 stranica pomnoženo s 25 rezultata po stranici) što znači da je na ovaj način moguće prikupiti informacije o samo 1 000 objekata.

Za daljnji rad odlučeno je da će se objekti pretraživati prema županijama države te se rezultati pohraniti na jedno mjesto te na takav način (s obzirom da Hrvatska ima 20 županija + grad Zagreb) možemo dobiti zasigurno više od 1000 objekata. Nadalje, potrebno je prilikom svakog pretraživanja navesti broj odraslih osoba i broj noćenja kako bi cijena smještaja bila vidljiva na stranici samog objekta (hotela, apartmana,...) te su kao filteri korišteni jedno noćenje za dvije odrasle osobe.

2.2. Paginacija rezultata

Paginacija (engl. *Pagination*) predstavlja niz web stranica koje su međusobno povezane i koje imaju sličan sadržaj. Kako bi mogli prikupiti rezultate sa svih mogućih stranica pretraživanja, program za struganje webom treba prijeći s prve stranice pretraživanja na drugu, zatim treću itd. Neke stranice imaju poveznice kojima se lako može pristupiti putem iteracije brojeva (slika 1).



Slika 1.. Poveznica stranice pretraživanja za portal baranjainfo

Na slici 1. prikazan je primjer uređene poveznice za news portal „baranjainfo“. S obzirom na to da se prelaskom na svaku sljedeću stranicu pretraživanja unutar navedene poveznice mijenja samo broj stranice (označen crvenim kvadratom na slici), moguće je jednostavnom iteracijom kroz listu s vrijednostima brojeva stranica (s brojem zadnje stranice pretraživanja kao maksimalnim brojem) proći kroz navedene poveznice i nije potreban pronalazak pojedine poveznice stranice. Međutim, kod Booking stranice, poveznica pretraživanja izgleda ovako:

„https://www.booking.com/searchresults.hr.html?label=gen173nr1FCAEoggI46AdIM1gEaGWIAQGYARC4ARfIAQzYAQHoAQH4AQuIAgGoAgO...“

od koje je navedeni tekst samo dio cijele poveznice te ne možemo jednostavnom iteracijom pomoću broja stranice doći do sljedeće stranice.

Nakon definicije web drivera:

```
driver = webdriver.Chrome(executable_path='C:/Users/38591/Desktop/chrom
edriver.exe')
```

gdje upisujemo putanju do web drivera (.exe datoteke koju smo prethodno skinuli) proučavamo web stranicu i njezinu paginaciju. Zanima nas poveznica koja se nalazi na strelici za sljedeću stranicu kako bi program za web struganje nastavio pristupati stranicama sve dok ne ponestane sljedeće stranice.

Desnim klikom na strelicu na web stranici i stiskom na „*inspect*“ možemo vidjeti HTML elementa.

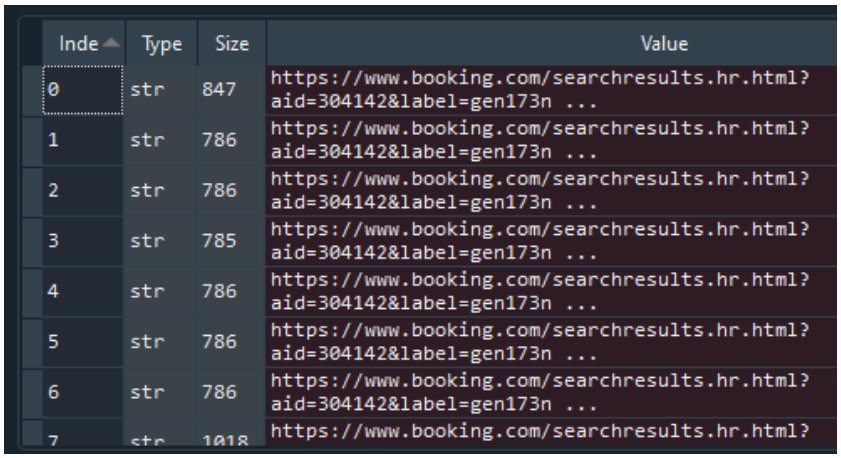
```
<li class="bui-pagination__item bui-pagination__next-arrow">
    <a href="/searchresults.hr.html?la..." data-page-
    next="" class="bui-pagination__link paging-
    next ga_sr_gotopage_2_4155"
    title="Sljedeća stranica">
```

Ovo je dio HTML-a koji sadrži poveznicu na sljedeću stranicu. Vidimo da se poveznica na sijedeću stranicu nalazi unutar „li“ elementa s klasom „bui_pagination__item“ ili „bui_pagination__next-arrow“. Kako, uz strelicu za sljedeću stranicu, brojevi stranica koji su prikazani na stranici također sadrže klasu „bui_pagination__item“ ali ne i drugu, potrebno je pretraživati poveznice pomoću druge navedene klase.

```
def dodavanje_stranica(url):
    while True:
        print(url)
        driver.get(url)
        next_page = driver.find_element_by_css_selector\
        ('li.bui-pagination__next-arrow>a').get_attribute('href')
        if next_page:
            url = next_page
            stranice.append(url)
        else:
            break
```

Zapisana je funkcija za skidanje poveznica sljedećih stranica. Ova funkcija ponovljena je za rezultate pretraživanja za svaku županiju posebno. Kod prve iteracije unesena je poveznica prve stranice rezultate pretraživanja koja je zatim proslijeđena prethodno definiranom driveru.

Prilikom izvođenja ovog dijela koda web driver otvara stranicu koja mu je proslijeđena i pretražuje u HTML-u element „li“ s navedenom klasom te uzima atribut „href“ elementa „a“ koji se nalazi unutar pronađenog elementa „li“ što je poveznica na sljedeću stranicu rezultata pretraživanja. Nadalje, sve dok postoji gumb za sljedeću stranicu, novi url koji se prosljeđuje funkciji postaje novo pronađena stranica. Kao rezultat imamo listu „stranice“ sa svim poveznicama na sljedeće stranice za sve županije (slika 2. prikazuje samo dio liste).

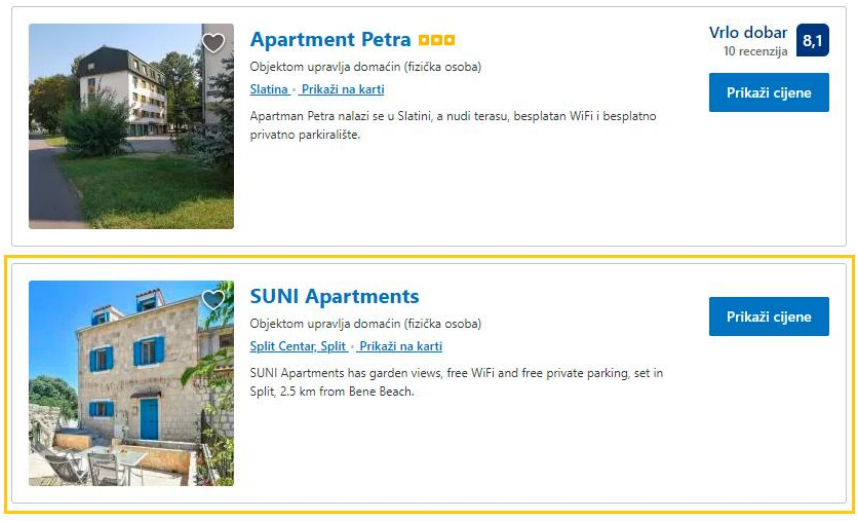


Inde	Type	Size	Value
0	str	847	https://www.booking.com/searchresults.hr.html?aid=304142&label=gen173n ...
1	str	786	https://www.booking.com/searchresults.hr.html?aid=304142&label=gen173n ...
2	str	786	https://www.booking.com/searchresults.hr.html?aid=304142&label=gen173n ...
3	str	785	https://www.booking.com/searchresults.hr.html?aid=304142&label=gen173n ...
4	str	786	https://www.booking.com/searchresults.hr.html?aid=304142&label=gen173n ...
5	str	786	https://www.booking.com/searchresults.hr.html?aid=304142&label=gen173n ...
6	str	786	https://www.booking.com/searchresults.hr.html?aid=304142&label=gen173n ...
7	str	1018	https://www.booking.com/searchresults.hr.html?

Slika 2.. Lista "stranice" sa svim poveznicama na slijedeće stranice

2.3. Struganje poveznica objekata

Nakon što je stvorena lista poveznica sa svim stranicama rezultata pretraživanja, potrebno je parsirati svaku pojedinu stranicu kako bi dohvatili poveznice na individualne stranice objekata (smještaja).



Slika 3. Element sa poveznicom na stranicu objekta

Kada kliknemo na „inspect“ dobijemo sljedeće:

```
<div class="sr_item  sr_item_new sr_item_default sr_property_block..
<div class="wl-entry-container">
      <a class="sr_item_photo_link sr_hotel_preview_track"
      href="/hotel/hr/suniapartment.hr.html?..." target="_blank" rel="n
      oopener" tabindex="-1" focusable="false" aria-hidden="true" data-
      et-click="">
```

Napomena: prilikom ispisivanja poveznica manualno su pridodane 3 točkice da se ne ispisuje cijela poveznica jer je preduga. Originalno, poveznica se sastoji od nekoliko redova nečistog teksta.

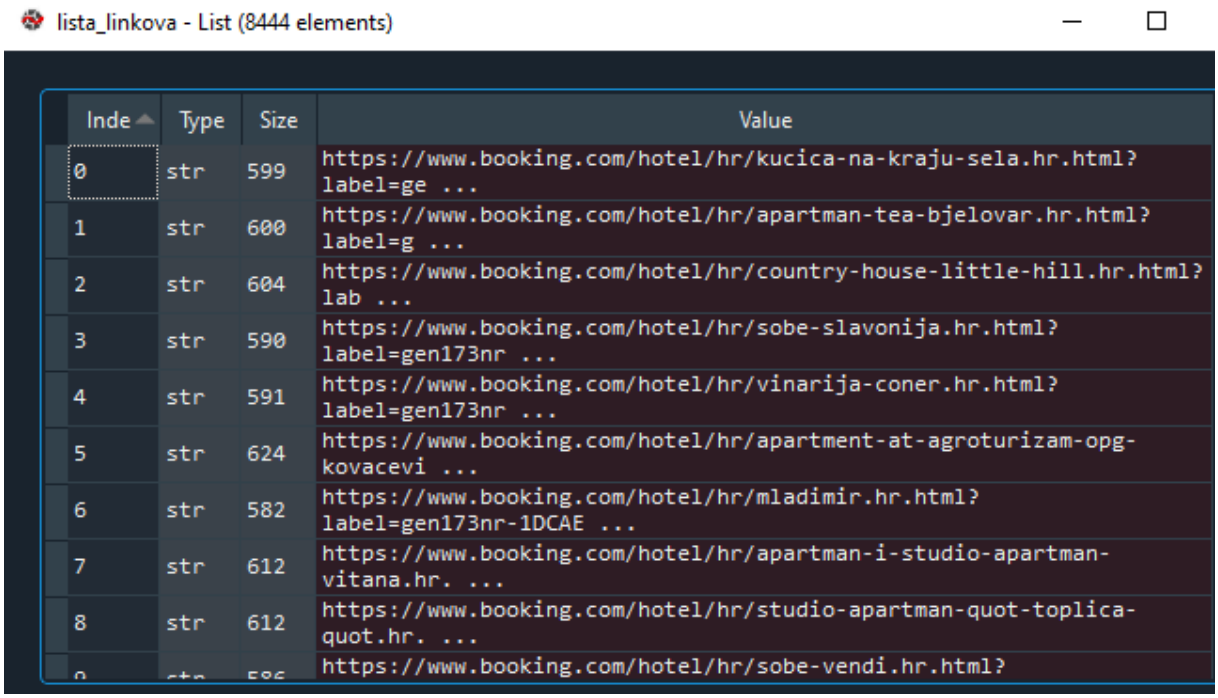
Nadalje, vidimo da se informacije o objektu nalaze unutar *„div“* elementa kod kojeg se može upotrijebiti nekoliko klasa za pretraživanje te je u ovom slučaju odabrana klasa *„sr_property_block“*. Unutar navedenog elementa nalazi se drugi *„div“* element klase *„sr_item_photo_link“* unutar kojeg se nalazi element *„a“* s poveznicom na stranicu objekta.

```
def parsiranje_stranica(url):
    driver.get(url)
    div = driver.find_elements_by_class_name("sr_property_block")
    for i in range(len(div)):
        item = div[i]
        a = item.find_element_by_tag_name("div").\
        find_element_by_tag_name( "a").get_property("href")
        lista_linkova.append(a)
```

Nakon što je poveznica proslijeđena web driveru, pronalazimo navedeni „div“ element koji sadrži potrebne podelemente. S obzirom na to da na stranici postoji više elemenata (točnije 25, jedan za svaki objekt na stranici) možemo pomoću petlje iterirati kroz navedeni objekt. Varijabli *item* pridružen je pojedinačni „div“ element u iteraciji te tražimo „div“ podelement koji sadrži element „a“ koji u konačnici sadrži poveznicu na objekt. Funkcija je pozvana nad svakom poveznicom unutar liste *stranice*.

```
for i in range(0, len(stranice)):
    print(i)
    parsiranje_stranica(stranice[i])
```

Pomoću navedene petlje izvršena je iteracija kroz svaku poveznicu rezultata pretraživanja te kao rezultat imamo listu *lista_linkova* koja sadrži 8444 poveznice, od kojih svaka predstavlja poveznicu na pojedinačni smještaj unutar Hrvatske.


lista_linkova - List (8444 elements)

Inde	Type	Size	Value
0	str	599	https://www.booking.com/hotel/hr/kucica-na-kraju-sela.hr.html?label=ge ...
1	str	600	https://www.booking.com/hotel/hr/apartman-tea-bjelovar.hr.html?label=g ...
2	str	604	https://www.booking.com/hotel/hr/country-house-little-hill.hr.html?lab ...
3	str	590	https://www.booking.com/hotel/hr/sobe-slavonija.hr.html?label=gen173nr ...
4	str	591	https://www.booking.com/hotel/hr/vinarija-coner.hr.html?label=gen173nr ...
5	str	624	https://www.booking.com/hotel/hr/apartment-at-agroturizam-opg-kovacevi ...
6	str	582	https://www.booking.com/hotel/hr/mladimir.hr.html?label=gen173nr-1DCAE ...
7	str	612	https://www.booking.com/hotel/hr/apartman-i-studio-apartman-vitana.hr. ...
8	str	612	https://www.booking.com/hotel/hr/studio-apartman-quot-toplica-quot.hr. ...
			https://www.booking.com/hotel/hr/sobe-vendi.hr.html?

Slika 4.. Lista sa poveznicama na smještaje

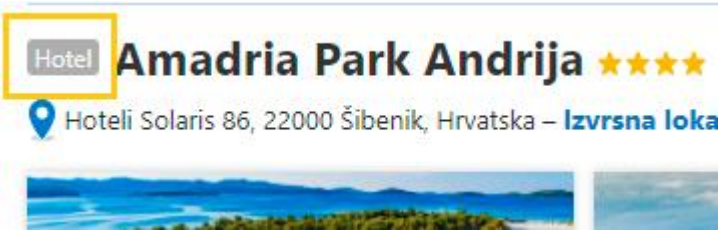
2.4. Struganje informacija o objektima smještaja

U ovom poglavlju opisani su postupci i rezultati web struganja pojedinih informacija o objektima smještaja. Za početak, informacije koje će se prikupljati jesu:

1) **Ime objekta** – ime objekta smještaja.

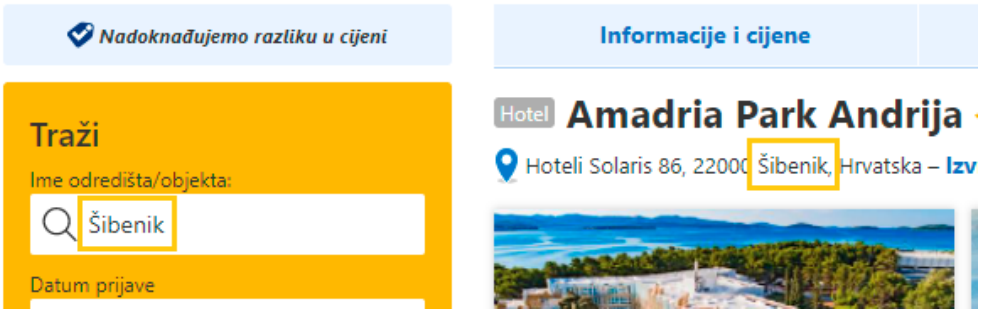
Slika 5. Ime objekta smještaja

2) **Tip objekta** – vrsta objekta u kojem se nudi smještaj (apartman, hotel, vila, ...)



Slika 6. Tip objekta

3) **Grad** –grad u kojem se nalazi smještaj. Kao što vidimo na slici 7. lokacija je zapisana na dva mjesta, u polju za pretraživanje i u sklopu adrese ispod imena smještaja. Kako je zapis adrese ispod naziva smještaja jedan tekst, potrebno je izvući grad iz cijelog zapisa, te je odlučeno da će se grad dohvaćati iz tražilice.



Slika 7. Lokacija smještaja

4) **Zvjezdice** – broj zvjezdica pridružen objektu smještaja.

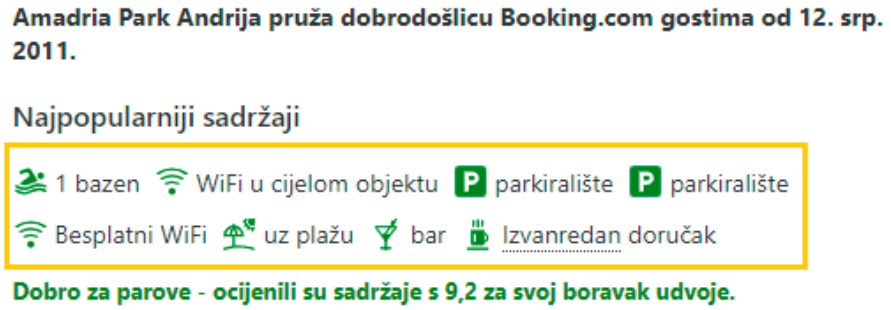
Slika 8. Broj zvjezdica objekta

5) **Ukupna ocjena, klasa objekta i broj recenzija** – ukupna ocjena hotela predstavlja konačnu recenziju objekta pridruženu od strane korisnika u obliku decimalnog broja (7.8, 9.4, ...). Klasa objekta predstavlja kategoriju ocjene koja je pridružena objektu u odnosu na

konačnu recenziju (dobar, izvanredan, sjajan,...). Broj recenzija predstavlja ukupan broj recenzija objekta.

Slika 9. Recenzija, kategorija i broj recenzija objekta

6) **Popularni sadržaji** – sadržaji dostupni u sklopu smještaja (doručak, bazen, besplatan internet, besplatno parkiranje, ...).

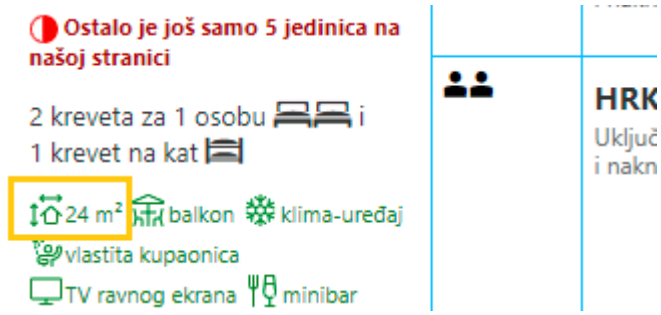


Slika 10. Popularni sadržaji objekta

7) **Cijena objekta** – cijena smještaja za jedno noćenje.

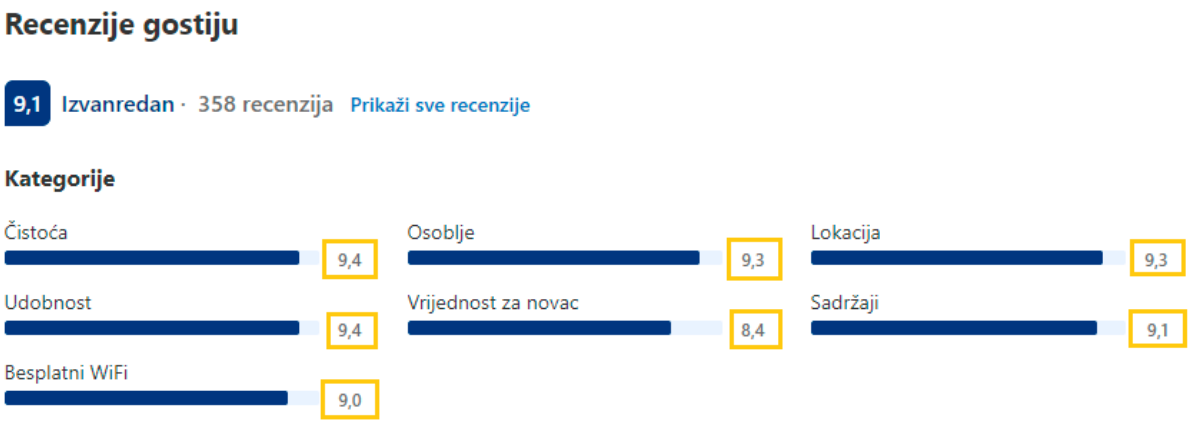
Slika 11. Cijena objekta za jedno noćenje

8) **Veličina hotela** – veličina objekta smještaja u m^2.



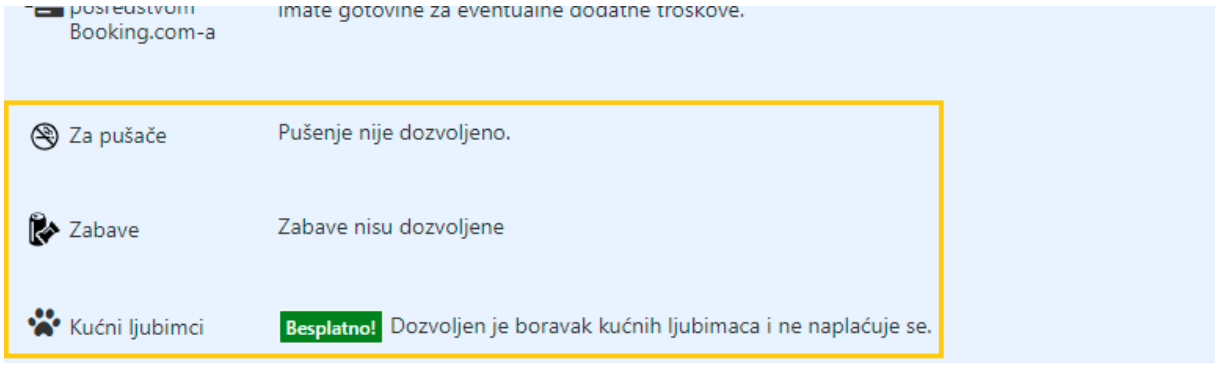
Slika 12. Veličina objekta

9) **Zasebne recenzije aspekata smještaja –** posebna recenzija za osoblje objekta, sadržaje objekta, čistoću objekta, udobnost objekta, vrijednost objekta, lokaciju objekta te Wifi objekta.



Slika 13. Zasebne recnezije objekta

9) **Dozvole objekta** – odnosi se na dozvole objekta smještaja vezano za pušenje, zabave, kućne ljubimce te kućni red i mir u objektu.



Slika 14. Informacije o dozvolama objekta

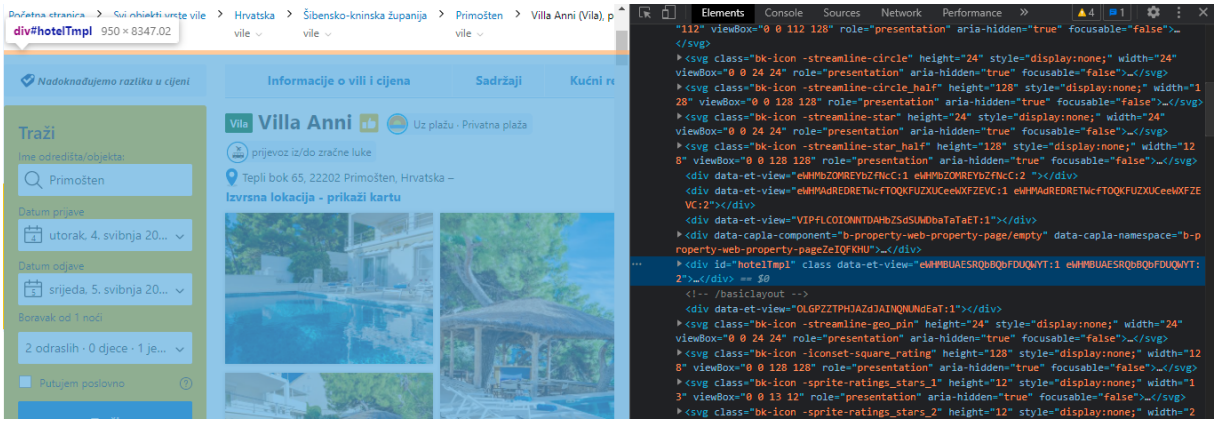
Prije prikupljanja samih podataka, potrebno je inicijalizirati pohranu za podatke (u obliku liste) od kojih će se kasnije izraditi skup podataka.

2.4.1. Funkcija za struganje informacija o objektima

Za web struganje sadržaja hotela napisana je podugačka funkcija koju ćemo u nastavku raščlaniti na dijelove te objasniti dio po dio.

```
def prikupljanje_podataka(url):
    driver.get(url)
    html = driver.page_source
    soup = BeautifulSoup(html,'lxml')
    objekt = soup.find('div', {"id" : "hotelTmpl"})
```

Za početak, driver prikuplja poveznicu pridodanu funkciji. Izvor web stranice (HTML) pridodajemo u varijablu *html* nad kojom pozivamo funkciju *BeautifulSoup.* Ovim postupkom dobili smo HTML stranice koji možemo parsirati za daljnje informacije. Varijabla *objekt* predstavlja *„div“* element stranice unutar kojeg se nalaze sve informacije koje želimo prikupiti o objektu.



Slika 15. "Div" element koji sadrži sve informacije o objektu

Kao što vidimo na slici, kada označimo *„div“* element unutar HTML-a, plavom bojom je prikazano koje dijelove web stranice označeni element obuhvaća. U ovom slučaju, *„div“* element sa *„id“* oznakom *„hotemTmpl“* sadrži sve blokove s potrebnim informacijama na dotičnoj web stranici. Iz tog razloga, navedeni *„div“* element pridružen je varijabli *objekt* koju ćemo nadalje koristiti za dohvaćanje željenih informacija o objektu.

1) **Ime objekta**

```
<a href="#availability" class="fn " id="hp_hotel_name_reviews" style="d
isplay:none"> Villa Anni </a>
```

Element „*a*“ klase „*fn*“ sadrži tekst koji nam treba za dohvaćanje imena objekta.

```
ime_hotela = objekt.find("a", class_="fn").text
ime.append(ime_hotela)
```

Prema tome, varijabli *ime_hotela* pridružujemo tekst unutar elementa „*a*“ klase „*fn*“ te dotično ime pridružujemo u prethodno izrađenu praznu listu *ime*. Sličan postupak provodi se za sve informacije koje prikupljamo, uz promjenu naziva elementa i klase.

2) **Tip objekta**

```
<div class="hp__hotel-title">...
  <span class="bui-badge bh-property-type bh-property-type--
  constructive-dark"
  data-exposure-name="property_tag_hp" style="margin-top: -
  3px;">Vila</span>
```

Element „*div*“ klase „*hp__hotel-title*“ sadrži „*span*“ element koji sadrži tekst u kojem se nalazi tip objekta.

```
tip_hotela = objekt.find("div", class_="hp__hotel-
title").find("span").text
tip.append(tip_hotela)
```

3) **Grad objekta**

```
<input type="search" class="c-autocomplete__input sb-searchbox__input"
placeholder="Primošten" value="Primošten"... aria-autocomplete="both">
```

Element „*input*“ klase „*c-autocomplete__input*“ sadrži oznaku imena *placeholder* unutar koje se nalazi naziv grada objekta. Navedene elemente možemo pretraživati po više atributa (identifikacijskom oznakom, klasom, ...) no radi pogodnosti i konzistentnosti, elemente ćemo pretraživati prema klasi ako je to moguće.

```
grad_hotela = objekt.find("input", class_="c-
autocomplete__input")["placeholder"]
grad.append(grad_hotela)
```

4) **Zvjezdice objekta**

```
<span class="bui-rating bui-rating--smaller" role="img" aria-
label='4 out of 5'>
```

Element „*span*“ klase „*bui-rating*“ sadrži oznaku *aria-label* gdje se nalazi informacija o broju zvjezdica npr. „3 od 5“.

```
try:
    zvjezdice_hotela = objekt.find("span", class_="bui-rating")
    ["aria-label"]
except:
    zvjezdice_hotela = "N/A"
zvjezdice.append(zvjezdice_hotela)
```

S obzirom na to da neki objekti nemaju pridružen broj zvjezdica, potrebno je koristiti try-except blok kako bi mogli preskočiti objekte bez navedene informacije i pridružiti im svojevoljnu vrijednost (u ovom slučaju string „N/A“). Bez navedenog bloka Python izbacuje AttributeError u kojem navodi da NoneType objekt nema atribut „find“ te se program prestane izvoditi čim naiđe na objekt bez određenog atributa (u ovom slučaju zvjezdica hotela, no isto se ponavlja i u svim varijablama vezanim za recenzije hotela jer nisu svi hoteli recenzirani).

5) **Ukupna ocjena objekta – recenzija objekta**

```
<div class="bui-review-score__badge" aria-
label="Ocijenjeno s 8,5 "> 8,5 </div>
```

Element „*div*“ klase „*bui-review-score__badge*“ sadrži recenziju objekta.

```
try:
    ocjena_ukupno_hotela = objekt.find("div",
    class_="bui-review-score__badge").text
except:
    ocjena_ukupno_hotela = "N/A"
ocjena_ukupno.append(ocjena_ukupno_hotela)
```

6) **Klasa objekta**

```
<div class="bui-review-score__title"> Vrlo dobar </div>
```

Element *„div“* klase *„bui-review-score__title“* sadrži klasu objekta u obliku teksta.

```
try:
    klasa_ocjene_hotela = objekt.find("div",
    class_="bui-review-score__title").text
except:
    klasa_ocjene_hotela = "N/A"
klasa_ocjene.append(klasa_ocjene_hotela)
```

7) **Broj recenzija**

```
<div class="bui-review-score__text"> 171 recenzija </div>
```

Element *„div“* klase *„bui-review-score__text“* sadrži broj recenzija objekta.

```
try:
    broj_recenzija_hotela = objekt.find("div",
    class_="bui-review-score__text").text
except:
    broj_recenzija_hotela = "N/A"
broj_recenzija.append(broj_recenzija_hotela)
```

8) **Popularni sadržaji**

```
<div class="hotel_description_wrapper_exp hp-description">
      <div class="important_facility" data-name-en="Swimming pool">
      <svg class="bk-icon -iconset-... focusable="false"></svg>1 bazen
      <div class="important_facility" data-name
      ... focusable="false"></svg>spa centar
      </div>
      ...
```

Element *„div“* klase *„important_facility“* koji se nalazi unutar elementa *„div“* klase *„hotel_descrription_wrapper_exp“* sadrži nazive popularnih sadržaja objekta.

```
div = objekt.find("div", {"class":"hotel_description_wrapper_exp"})
sadrzaji_hotela = div.find_all("div", {"class":"important_facility"})
```

```
lista_stringova = [svaki.text.replace("\n\n", "").\
                   replace("\n", "") for svaki in sadrzaji_hotela]
sadrzaji.append(lista_stringova)
```

Listi *lista_stringova* pridružuje se svaka tekstualna vrijednost za svaki prikupljeni *„div“* element te se ujedno znakovi za novi red (*„\n“*) zamjenjuju s praznim stringom s obzirom na to da bi Python, proučavanjem ispisa elemenata, pridodao nepotrebne dotične znakove. Ova operacija provodi se za svaki *„div“* element klase *„important_facility“* unutar iteratora navedenih elemenata.

9) **Cijena objekta**

```
<span class="prco-valign-middle-helper">
HRK 523</span>
```

Element *„span“* klase *„prco-valign-middle-helper“* sadrži cijenu objekta.

```
try:
    cijena_hotela = objekt.find("span", {"class":
    "prco-valign-middle-helper"}).text
except:
    cijena_hotela = "N/A"
cijena.append(cijena_hotela)
```

Kao što vidimo, neki objekti prilikom pokretanja programa nisu imali zapisanu cijenu te se privremeno cijena pohranjuje kao *„N/A“* te je naknadno (u idućem poglavlju) objašnjen razlog nedostatka navedenih vrijednosti.

10) **Veličina objekta**

```
<div class="hprt-facilities-facility" data-name-en="room size">
      <span class=""> <svg class="bk-icon -streamline-
      room_size" fill="#008009" height="16" width="16" viewBox="0 0 24 2
      4"... focusable="false">...</svg>17 m²</span>
```

Element *„div“* klase *„hprt-facilities-facility“* sadrži veličinu objekta u m^2. Na stranici objekta postoji više elemenata navedene klase te je potrebno informaciju pretražiti pomoću dodatne oznake *data-name-en* koja za veličinu objekta sadrži vrijednost *„room size“*.

```
try:
    velicina_hotela = objekt.find("div",attrs ={"class":
    "hprt-facilities-facility", "data-name-en":"room size"}).\
    find("span").text
except:
    velicina_hotela = "N/A"
velicina.append(velicina_hotela)
```

11) **Zasebne recenzije objekata smještaja**

```
<li class="v2_review-scores__subscore
hp_subscore_explanation_item_desktop">...
      <span class="c-score-bar__title">
      Čistoća</span><span class="c-score-bar__score">8,7</span>
```

Element *„li“* klase *„v2_review-scores__subscore“* sadrži *„span“* element klase *„c-score-bar__score“* unutar kojeg se u obliku teksta nalazi potrebna informacija (recenzija za dotičnu kategoriju).

```
try:
    recenzije_cistoca_hotela = objekt.find_all("li", {"class":
    "v2_review-scores__subscore"})[0].find("span", {"class":
    "c-score-bar__score"}).text
except:
    recenzije_cistoca_hotela = "N/A"
recenzije_cistoca.append(recenzije_cistoca_hotela)
```

Za svaku od prethodno navedenih 7 kategorija (WiFI, lokacija, čistoća, udobnost,...) se na isti način prikuplja navedena informacija, uz jedinu promjenu indeksa „li“ elementa. Tako npr. kad pogledamo sliku 13. vidimo da se čistoća nalazi prva po redu (pri tome indeks 0), zatim osoblje s indeksom 1, lokacija s indeksom 2, itd.

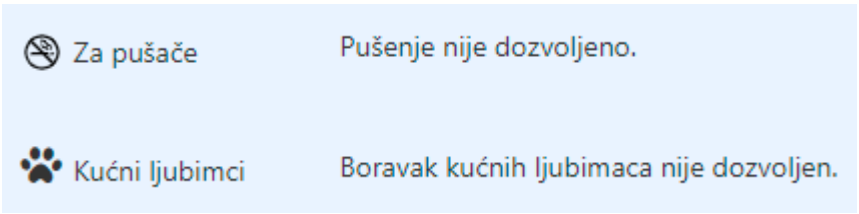
12) **Dozvole objekta**

```
<div class="description description--house-rule">...
</span>
<span>Kućni ljubimci</span>
<p>
Boravak kućnih ljubimaca nije dozvoljen.</p>
<div style="clear:both"></div>
```

Element *„div“* klase *„description--house-rule“* sadrži *„span“* element sa tekstom kao indikatorom o kojoj se dozvoli radi (kućni ljubimci, zabave ili pušenje) i *„p“* element sa samim tekstom koji se odnosi na dozvolu.

```
mjera_hotela = []
odobrenje = []
div1 = objekt.find_all("div", {"class":"description--house-rule"})
for div in div1:
    for span in div.find_all("span", {"class":None}):
        mjera_hotela.append(span.text)
time.sleep(2)
for div in div1:
    p = div.find("p", class_=None)
    odobrenje.append(p.text.replace("\n", ""))
res = dict(zip(mjera_hotela, odobrenje))
dozvole.append(res)
```

Pronalazimo prethodno spomenuti „*div*“ element koji sadrži sve potrebne podelemente. Nazivi dozvola i sam tekst dozvola se pridodaju posebnim listama. Konačan rezultat – res zamišljen je tako da se rezultat sprema u obliku rječnika iz navedene dvije liste tako da je svakom odgovarajućem elementu u listi *mjera_hotela* kao ključu, pridružena odgovarajuća vrijednost unutar liste *odobrenje* kao vrijednost navedenog ključa. Tako bi npr. za dozvole na slici 16:



Slika 16. Dozvole objekta (primjer)

odgovarajući rječnik izgledao ovako:

```
{"Za pušače": "Pušenje nije dozvoljeno.", "Kućni ljubimci": "Boravak ku
ćnih ljubimaca nije dozvoljen."}
```

Konačno, funkcija se poziva na sljedeći način:

```
for i in range(0, len(lista_linkova)):
    print(i)
    prikupljanje_podataka(lista_linkova[i])
```

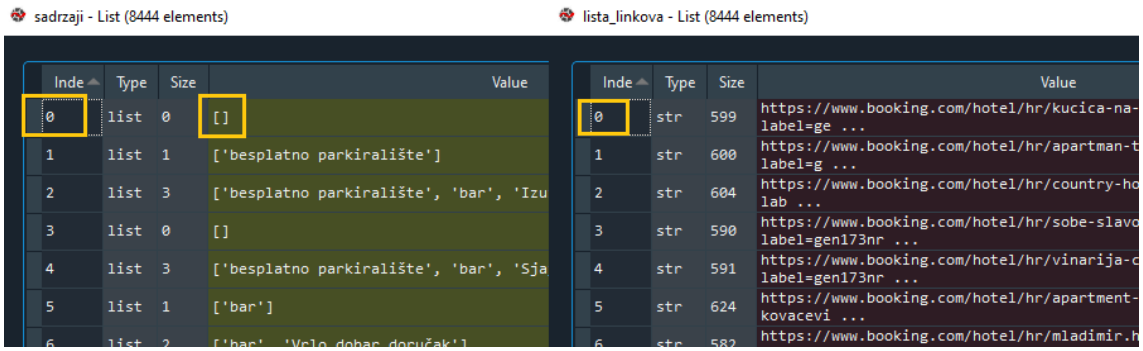
gdje iteriramo kroz svaku poveznicu unutar liste *lista_linkova.*

2.4.2. Istraživanje null vrijednosti unutar pojedinih listi

Unutar ovog poglavlja istraženi su razlozi zašto su cijene nekih hotela određene kao *„N/A“* kao i zašto su liste popularnih sadržaja nekih hotela prazne.

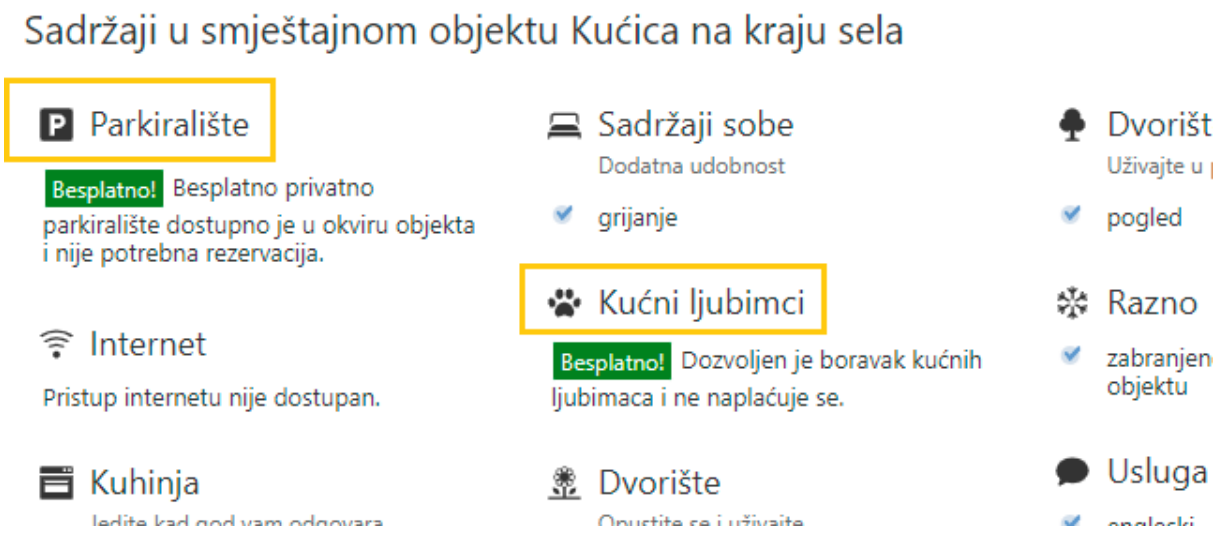
1) **Liste popularnih sadržaja hotela**

Primjer hotela s praznom listom sadržaja možemo proučiti tako da pogledamo listu *sadrzaji* te tražimo indeks elementa gdje je element prazna lista. Nakon što pronađemo dotični indeks, u listi *lista_linkova* pronađemo poveznicu s istim indeksom i otvorimo je u internet pregledniku (prikazano na slici 17).



Slika 17.. Istraživanje prazne liste sadržaja

Otvaranjem dotične poveznice u pregledniku primijećeno je da dotična poveznica nema sadržaje navedene na mjestu gdje smo ih pretraživali, već pri dnu stranice. Iako su objekti bez procesiranog odjeljka rijetki, svejedno želimo pronaći bar neke pogodnosti i popularne sadržaje objekata koji navedeni odjeljak nemaju.



Slika 18. Sadržaji hotela bez popularnih sadržaja

Pri kraju stranice objekta (prije popisa dozvola) svaki objekt ima zapisane uobičajene sadržaje koji su dostupni u objektu (da li ima kuhinju, televizor, dobar pogled,...) no uz to je primijećeno da se za svaki objekt navodi da li ima dostupan internet, besplatan parking te da li dozvoljava kućne ljubimce, što, ako je omogućeno, sadrži zeleni kvadrat s natpisom „Besplatno!“ ispod samog naziva sadržaja.

```
<div class="facilitiesChecklist">
<div class="facilitiesChecklistSection" data-section-id="16">
      <h5>Parkiralište </h5>
      <span class="bui-badge bui_fill_constructive">Besplatno!</span>
      Besplatno privatno parkiralište dostupno je u okviru objekta i nij
      e potrebna rezervacija.
</div>
```

Element *„div“* klase *„facilitiesChecklist“* sadrži *„div“* element klase *„facilitiesChecklistSection“* za svaki od elemenata vidljivih sa slike 18. (parkiralište, internet, kuhinja, sadržaj sobe,...). Elementi koji su dostupni u sklopu objekta imaju prethodno navedeni zeleni kvadrat s natpisom *„Besplatno!“* koji se nalazi unutar *„span“* elementa klase *„bui-badge“*.

```
null = []
for i, j in enumerate(sadrzaji):
    if len(j) == 0:
        print(i)
        null.append(i)
```

Stvorena je nova lista *null* koja sadrži sve indekse elemenata liste *sadrzaji* čija je dužina elementa jednaka nuli (na stranici objekta nisu definirani popularni sadržaji ispod opisa objekta).

```
for i in range(0, len(null)):
    print(i)
    driver.get(lista_linkova[null[i]])
    html = driver.page_source
    soup = BeautifulSoup(html,'lxml')
    objekt = soup.find('div', {"id" : "hotelTmpl"})
    popularnosti = objekt.find("div", class_="facilitiesChecklist")
```

Pomoću drivera pristupamo poveznici unutar liste *lista_linkova* s indeksom iz liste *null*.

```
popularnosti_hotela = popularnosti.find_all("div", {"class":"facilities
ChecklistSection"})
    lista_stringova = []
    for pop in popularnosti_hotela:
        if pop.find("span", class_="bui-badge") == None:
            continue
        else:
            naslovi = pop.find_all("h5")
            for ime in naslovi:
                lista_stringova.append(ime.text.replace("\n\n\n", "").\
                                       replace("\n", ""))
                print(lista_stringova)
    sadrzaji[null[i]] = lista_stringova
```

Zatim u varijablu *popularnosti_hotela* spremamo iterator koji sadrži sve spomenute „*div*“ elemente. Ukoliko element ne sadrži „*span*“ element klase „*bui-badge*“ petlja nastavlja dalje, no ako postoji, pridružujemo sve „*h5*“ elemente varijabli *ime* te za svaki „*h5*“ element pridružujemo tekst unutar elementa.

Prikupljene tekstove na kraju pridružujemo listi *sadrzaji* s pripadajućim indeksom (indeks gdje je lista sadržaja bila prazna).

2) **Objekti bez prikupljenje cijene objekta**

Pregledavajući jednu od poveznica gdje je cijena objekta označena sa *„N/A“* dobijemo sljedeći prikaz (slika 19).

Slika 19. Objekti bez cijene

Sa slike je vidljivo da, iz nekog razloga, određeni objekti ne mogu trenutno primati rezervacije na stranici Booking. Razlog tome može također biti to što je prošao određeni period između prikupljanja poveznice objekta i struganja sadržaja same stranice (web struganje je izvodilo po nekoliko dana).

```
cijena.count("N/A")
```

Prebrojavanjem elemenata unutar liste *ocjena* s vrijednosti *„N/A“* dobijemo 11, što znači da postoji 11 objekata unutar liste koji ne mogu trenutno primati rezervacije na stranici. Ovaj problem ćemo riješiti tako da, s obzirom na to da je broj navedenih objekata veoma malen kad ga usporedimo s ukupnim brojem objekata (8 444), izbacimo navedene objekte iz skupa podataka (gubimo samo 11 opservacija što nije velik gubitak).

2.4.3. Opis skupa podataka

Skup podataka stvoren je pomoću *pandas* funkcije *DataFrame* gdje u obliku rječnika navodimo naslov varijable (stupca) kao ključ te listu s određenim vrijednostima kao vrijednost navedenog ključa.

```
booking = pd.DataFrame({'Naziv objekta': ime,'Vrsta objekta': tip,'Loka
cija (grad)': grad, "Kategorizacija objekta": zvjezdice, "Recenzija obj
ekta": ocjena_ukupno, "Kategorija ocjene objekta" : klasa_ocjene, "Broj
 recenzija": broj_recenzija, "Veličina objekta": velicina, "Cijena obje
kta": cijena, "Recenzija osoblja": recenzije_osoblje, "Recenzija sadrža
ja": recenzije_sadrzaji, "Recenzija čistoće": recenzije_cistoca, "Recen
zija udobnosti": recenzije_udobnost, "Recenzija vrijednosti": recenzije
_vrijednost, "Recenzija lokacije": recenzije_lokacija, "Recenzija wifi"
```

```
: recenzije_wifi, "Dozvole objekta": dozvole, "Popularni sadržaji": sad
rzaji})
```

Naziv objekta	Vrsta objekta	Lokacija (grad)	Kategorizacija objekta
Kućica na kraju sela	Vikendica	Martinac Trojstveni	N/A
House of Tea	Apartman	Bjelovar	3 out of 5
Country House Little Hill	Ladanjska kuća	Bjelovar	3 out of 5
Rooms Slavonija	Privatni smještaj	Daruvar	4 out of 5
Bed&Breakfast Vinarija Coner	Smještaj s doručkom	Bjelovar	4 out of 5
Apartment at Agroturizam OPG K…	Apartmani	Daruvar	3 out of 5
Hotel Mladimir	Hotel	Daruvar	3 out of 5
Apartman i studio apartman Vit…	Apartmani	Garešnica	N/A
Studio apartman Toplica	Apartman	Daruvar	3 out of 5
Rooms Vendi	Privatni smještaj	Daruvar	4 out of 5

Slika 20. Stupci od 1 - 4 skupa podataka

Recenzija objekta	Kategorija ocjene objekta	Broj recenzija	Veličina objekta	Cijena objekta
N/A	N/A	N/A	80 m²	HRK 692
9,7	Izuzetan	26 recenzija	72 m²	HRK 335
9,5	Izuzetan	21 recenzija	35 m²	HRK 363
9,3	Izvanredan	26 recenzija	51 m²	HRK 348
9,3	Izvanredan	77 recenzija	25 m²	HRK 529
8,9	Sjajan	35 recenzija	28 m²	HRK 303
9,2	Izvanredan	64 recenzije	29 m²	HRK 537
N/A	N/A	N/A	38 m²	HRK 253
9,7	Izuzetan	33 recenzije	40 m²	HRK 291
9,0	Izvanredan	71 recenzija	18 m²	HRK 242

Slika 21. Stupci od 5 - 9 skupa podataka

Recenzija osoblja	Recenzija sadržaja	Recenzija čistoće	Recenzija udobnosti	Recenzija vrijednosti
N/A	N/A	N/A	N/A	N/A
10	9,8	10	9,8	9,8
9,9	9,9	9,6	9,6	9,4
9,5	9,0	9,2	8,9	9,1
9,6	9,8	8,9	9,6	9,0
9,5	8,9	8,4	8,8	9,2
9,6	9,6	9,3	9,3	9,3
N/A	N/A	N/A	N/A	N/A
9,9	9,7	9,7	9,6	9,5
9,3	8,7	9,2	9,1	8,9

Slika 22. Stupci od 10 - 14 skupa podataka

Recenzija lokacije	Recenzija wifi	Dozvole objekta	Popularni sadržaji
N/A	N/A	{'Za pušače': 'Pušenje nije dozv…	['besplatno parkiralište', 'Dozvoljeni ljubimci']
9,2	10	{'Za pušače': 'Pušenje nije dozv…	['besplatno parkiralište']
9,6	7,5	{'Kućni ljubimci': 'Besplatno!Do…	['besplatno parkiralište', 'bar', 'Izuzetan doručak']
9,6	6,7	{'Za pušače': 'Pušenje nije dozv…	['Besplatni WiFi']
9,2	7,5	{'Za pušače': 'Pušenje nije dozv…	['besplatno parkiralište', 'bar', 'Sjajan doručak']
9,0	2,5	{'Kućni ljubimci': 'Besplatno!Do…	['bar']
8,8	9,7	{'Kućni ljubimci': 'Boravak kućn…	['bar', 'Vrlo dobar doručak']
N/A	N/A	{'Za pušače': 'Pušenje nije dozv…	['Kućni ljubimci', 'Parkiralište', 'Internet']
9,8	10	{'Za pušače': 'Pušenje nije dozv…	['bar']
9,3	10	{'Za pušače': 'Pušenje nije dozv…	['Besplatni WiFi', 'besplatno parkiralište', 'Dozvoljeni ljubimci']

Slika 23. Stupci od 15 - 18 skupa podataka

3. Čišćenje podataka

U ovom poglavlju opisani su razlozi i postupci prilikom procesa čišćenja podataka, kao i konačan skup podataka dobiven primjenom navedenih postupaka.

Informacije o pojedinoj varijabli skupa podataka možemo dobiti na sljedeći način:

```
booking.info()
```

Ova metoda koristi se za ispis informacija o skupu podataka, uključujući tip (vrstu) podataka indeksa i samih stupaca (varijabli), informacije o broju vrijednosti unutar stupaca koje nisu *null,* broj redaka i stupaca te zauzeće memorije.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8444 entries, 0 to 8443
Data columns (total 18 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Naziv objekta              8444 non-null   object
 1   Vrsta objekta              8444 non-null   object
 2   Lokacija (grad)            8444 non-null   object
 3   Kategorizacija objekta     8444 non-null   object
 4   Recenzija objekta          8444 non-null   object
 5   Kategorija ocjene objekta  8444 non-null   object
 6   Broj recenzija             8444 non-null   object
 7   Veličina objekta           8444 non-null   object
 8   Cijena objekta             8444 non-null   object
 9   Recenzija osoblja          8444 non-null   object
 10  Recenzija sadržaja         8444 non-null   object
 11  Recenzija čistoće          8444 non-null   object
 12  Recenzija udobnosti        8444 non-null   object
 13  Recenzija vrijednosti      8444 non-null   object
 14  Recenzija lokacije         8444 non-null   object
 15  Recenzija wifi             8444 non-null   object
 16  Dozvole objekta            8444 non-null   object
 17  Popularni sadržaji         8444 non-null   object
dtypes: object(18)
memory usage: 1.2+ MB
```

S navedenog ispisa možemo vidjeti da skup podataka pripada klasi *DataFrame,* ima 8444 redaka i 18 stupaca od kojih su svi tipa *object* (stringovi - tekst) i svi navedeni sadrže 8444 vrijednosti od kojih niti jedna nije *null* (zato što su nepostojeće vrijednosti označene sa „*N/A*“ što u Pythonu ne predstavlja službeni zapis nepostojeće vrijednosti).

Ciljevi ovog koraka procesiranja podataka jesu:

1) formatiranje same vrijednosti podataka tako da se izbace dodatni razmaci s početka i kraja stringova kod onih varijabli gdje su isti prisutni te sa selekcijom podniza (engl. *substring*) izvučemo potrebne informacije iz teksta

2) pretvorba podataka u odgovarajuću vrstu podataka (ovisno o stupcu) kako bi prilikom analize podataka podaci bili u odgovarajućem formatu za lakšu manipulaciju podacima (brojevi kao decimalni brojevi ili cijeli brojevi – integeri, imena gradova i objekata kao tekst te određeni tekstovi kao kategorije – npr. varijabla *Vrsta objekta* može se pretvoriti u kategoriju s obzirom na to da imamo ograničen broj kategorija kojima određeni objekt pripada).

3) istraživanje vrijednosti koje nedostaju – promatramo stupac po stupac te gledamo koliko nepostojećih vrijednosti pojedini stupac ima te imputacija ili izbacivanje redova s navedenim vrijednostima.

Za početak, izbacujemo iz skupa podataka redove unutar skupa koji imaju cijenu jednaku vrijednosti „*N/A*“ jer ne postoji velik gubitak informacija ako izbacimo 11 redova unutar skupa podataka.

```
booking_proba = booking.copy()
booking_proba = booking_proba[booking_proba["Cijena objekta"] != "N/A"]
```

Izrađujemo kopiju originalnog skupa podataka pomoću funkcije *copy* jer se običnim pridruživanjem prilikom promjene vrijednosti jednog objekta mijenjaju vrijednosti originalnog objekta s obzirom na to da pridruživanjem ne dobijemo kopiju objekta već samo drugu varijablu koja referencira isti objekt unutar memorije. Svi koraci čišćenja provedeni su nad kopijom kako bi originalni skup podataka ostao netaknut ako se pri isprobavanju određenih metoda i funkcija dogodi neočekivana promjena nad podacima.

Nakon stvorene kopije skupa podataka, varijabli *booking_proba* pridružujemo vrijednosti unutar istog skupa podataka koji unutar varijable *Cijena objekta* sadrže vrijednosti različite od „*N/A*“ što su sve vrijednosti gdje je dohvaćena cijena objekta. Sada skup podataka sadrži 8434 opservacije.

1) **Naziv objekta**

Kod navedene varijable nije potrebno ništa mijenjati s obzirom da je prikupljen tekst već uređen te samo provjeravamo da li postoje *null* vrijednosti unutar varijable.

Ukupan broj *null* vrijednosti unutar varijable možemo provjeriti na sljedeći način:

```
booking_proba["Naziv objekta"].isnull().sum()
```

gdje prvo pozivamo metodu *isnull* koja vraća niz *boolean* vrijednosti (*True* ili *False*) gdje *True* označava vrijednosti koje su označene s *null* te tražimo na kraju sumu navedeneih vrijednosti.

```
booking_proba["Naziv objekta"].isnull().sum()
Out[14]: 0
```

Pozivanjem funkcije dobiveno je da varijabla s nazivima objekata ne sadrži niti jedan redak s *null* vrijednosti, što znači da su dohvaćena svi nazivi objekata s web stranice.

2) **Vrsta objekta**

Navedena varijabla je također uređena već pri dohvaćanju (sastoji se od čistog teksta bez suvišnih razmaka) te je potrebno samo provjeriti da li postoje *null* vrijednosti te pretvoriti vrijednosti u kategorije s obzirom na to da postoji ograničen raspon vrijednosti koje određeni objekt može poprimiti (jedan objekt može biti vila ili apartman ili vikendica, itd).

```
booking_proba["Vrsta objekta"].isnull().sum()
Out[17]: 0
```

Vidimo da varijabla *Vrsta objekta* kao i prethodna varijabla ne sadrži objekte bez informacije o vrsti objekta, drugim riječima, svi objekti su klasificirani pod određenom kategorijom.

```
booking_proba['Vrsta objekta'] = booking_proba['Vrsta objekta'].astype(
'category')
```

Za pretvorbu vrste podataka unutar varijable korištena je funkcija *astype* gdje kao parametar navodimo vrstu podataka koju želimo da vrijednosti unutar varijable poprime (to može biti cijeli broj – *int*, decimalni broj *float*,... no u ovom slučaju biramo *category* za kategoriju).

```
booking_proba['Vrsta objekta'].unique()
```

```
Out[16]:
['Vikendica', 'Apartman', 'Ladanjska kuća', 'Privatni smještaj', 'Smješ
taj s doručkom', ..., 'Motel', \
'Resort', 'Smještaj kod domaćina', 'Kamp s luksuznim šatorima', 'Smješt
aj na brodu'] Length: 23
Categories (23, object): ['Vikendica', 'Apartman', 'Ladanjska kuća', 'P
rivatni smještaj', ..., 'Resort', \
'Smještaj kod domaćina', 'Kamp s luksuznim šatorima', 'Smještaj na brod
u']
```

Pomoću funkcije *unique* možemo također provjeriti koliko jedinstvenih kategorija postoji unutar navedene varijable te iz samog ispisa možemo vidjeti da postoje 23 različite vrste smještaja među prikupljenim uzorcima. Neke kategorije su suvišne, s obzirom na to da predstavljaju množinu već postojeće kategorije, npr. unutar skupa imamo kategoriju apartmani i apartman, vikendice i vikendica, vile i vila, itd. te su kategorije koje sadrže množinski oblik inicijalne kategorije preimenovane u jedninu kako bi ujedno smanjili broj suvišnih kategorija. U konačnici, imamo 14 kategorija vrste smještaja.

3) **Lokacija (grad)**

Kao i prethodne dvije varijable, tekstualni sadržaj navedene ne sadrži suvišne razmake te nije potrebno dodatno uređivati sami tekst te će se samo provjeriti da li je svim objektima pridružen grad.

```
booking_proba["Lokacija (grad)"].isnull().sum()
Out[18]: 0
```

Navedena varijabla ne sadrži niti jedan redak bez odgovarajuće vrijednosti, što znači da je kod svakog objekta bilo moguće prikupiti naziv, vrstu objekta i lokaciju istog te iz tog razloga i nije bilo potrebno uvoditi *try-except* blok prilikom prikupljanja podataka s web stranice.

4) **Zvjezdice**

Navedena varijabla, za razliku od ostalih, ima prikupljen tekst u obliku „*3 out of 5*“ (vidljivo na slici 20) te nas zanima samo prvi broj unutar teksta (u ovom slučaju 3, koji označava točan broj zvjezdica te je ostali dio teksta suvišan).

```
booking_proba["Kategorizacija objekta"] = booking_proba["Kategorizacija
objekta"].str[0]
```

```
booking_proba["Kategorizacija objekta"] = booking_proba["Kategorizacija
objekta"].replace("N", np.nan)
```

Vrijednosti stupca se transformiraju tako da se preuzima samo prvi element teksta (točan broj zvjezdica), zatim indeksiramo string tako da uzimamo prvi član (pod indeksom 0). Zatim, s obzirom na to da postoje objekti za koje dotična varijabla iznosi *„N/A“* (što su objekti kojima nije pridružen broj zvjezdica) zamjenjujemo vrijednost *„N“* sa službenom *NaN* vrijednosti (*Not a Number*). Zamjenjujemo *„N“* umjesto *„N/A“* jer smo prethodno obavili indeksiranje teksta kojim je prikupljeno samo prvo slovo stringa, što je slovo N.

```
booking_proba["Kategorizacija objekta"] = pd.to_numeric(booking_proba[
"Kategorizacija objekta"], errors='coerce')
```

Zatim, s obzirom na to da su vrijednosti prikupljene indeksiranjem sami brojevi ili *NaN* vrijednosti, potrebno je isti stupac pretvoriti u broj. Korištena je funkcija *to_numeric* gdje navodimo varijablu koju želimo konvertirati u numerički tip podataka te je u ovom slučaju potrebno navesti vrijednost *„coerce“* uz parametar *errors* jer unutar varijable postoje *NaN* vrijednosti koje navodom specifičnog parametra ignoriramo (u suprotnom nije moguća pretvorba varijable u numeričku vrijednost jer nisu sve vrijednosti numeričke).

```
booking_proba["Kategorizacija objekta"].isnull().sum()
Out[19]: 1453
```

Nakon uređivanja, varijabla sadrži 1453 opservacije bez pridruženog broja zvjezdica.

5) **Recenzija objekta**

Format varijable vidljiv je na slici 22. te možemo vidjeti da imamo *„N/A“* vrijednosti te da su numeričke vrijednosti uvučene za jedno mjesto (unutar polja nalazi se jedan razmak te nakon razmaka sam broj). Navedeni problem možemo riješiti na više načina (od kojih je jedan primjena *strip* metode koja uklanja razmake na početku i kraju teksta) no u ovom slučaju riješeno je na sljedeći način:

```
booking_proba["Recenzija objekta"] = booking_proba["Recenzija objekta"]
.str[1:]
booking_proba["Recenzija objekta"] = booking_proba["Recenzija objekta"]
.replace("/A", np.nan)
```

```
booking_proba["Recenzija objekta"] = [str(x).replace(',', '.') for x in
booking_proba["Recenzija objekta"]]
booking_proba["Recenzija objekta"] = pd.to_numeric(booking_proba["Recen
zija objekta"], errors='coerce')
```

Prvo uzimamo podniz stringa koji uključuje znak na indeksu 1 (što je zapravo drugi znak unutar teksta, što je u ovom slučaju broj ili kosa crta - / kod objekata koji nemaju recenziju). Zatim vrijednosti *„/A“* zamjenjujemo s *NaN* te pomoću metode *replace* zamjenjujemo decimalni zarez s decimalnom točkom unutar teksta s obzirom na to da u prvotnom slučaju Python pri konverziji u numerički tip podataka sve vrijednosti pretvori u *null* vrijednosti. Nakon zamjene decimalnog zareza, izvršava se konverzija varijable u željeni tip podataka (decimalni broj) s ignoriranjem *NaN* vrijednosti te ispisom zbroja *NaN* vrijednosti vidimo da postoji 1384 objekta koji nemaju recenziju.

```
booking_proba["Recenzija objekta"].isna().sum()
Out[27]: 1384
```

6) **Kategorija ocjene objekta**

Kao i kod prethodne varijable, postoji suvišan razmak prije samog teksta.

```
booking_proba["Kategorija ocjene objekta"] = booking_proba["Kategorija
ocjene objekta"].str[1:]
booking_proba["Kategorija ocjene objekta"] = booking_proba["Kategorija
ocjene objekta"].replace("/A", np.nan)
booking_proba["Kategorija ocjene objekta"] = booking_proba["Kategorija
ocjene objekta"].astype('category')
```

Također je uzet podniz teksta počevši sa znakom pod indeksom 1 te su zamijenjene vrijednosti *„/A“* s odgovarajućim vrijednostima. Očišćeni tekstovi pretvoreni su u kategoriju te možemo iz ispisa vidjeti da postoji 6 različitih kategorija smještaja i 1384 *NaN* vrijednosti.

```
booking_proba["Kategorija ocjene objekta"].unique()
Out[24]:
[NaN, 'Izuzetan ', 'Izvanredan ', 'Sjajan ', 'Vrlo dobar ', 'Dobar ', '
Prosječna ocjena ']
Categories (6, object): ['Izuzetan ', 'Izvanredan ', 'Sjajan ', 'Vrlo d
obar ', 'Dobar ', 'Prosječna ocjena ']
booking_proba["Kategorija ocjene objekta"].isna().sum()
Out[25]: 1384
```

7) **Broj recenzija**

Broj recenzija, kao i prethodne dvije varijable sadrži suvišni razmak na početku teksta. Broj recenzija varira od objekta do objekta, te postoje objekti s jednoznamenkastom recenzijom do onih s četveroznamenkastom. S obzirom na razliku između brojeva, kako bi „uhvatili“ sve potrebne brojeve, treba nam podniz od znaka pod indeksom 1 do znaka s indeksom 5 (jer se na indeksu pet nalazi zadnja znamenka četveroznamenkastog broja recenzije).

```
booking_proba["Broj recenzija"] = booking_proba["Broj recenzija"].\
str[1:6]
booking_proba["Broj recenzija"] = booking_proba["Broj recenzija"].\
replace("/A", np.nan)
booking_proba["Broj recenzija"] = [str(x).replace('r', '') for x in \
booking_proba["Broj recenzija"]]
booking_proba["Broj recenzija"] = pd.to_numeric(booking_proba["Broj rec
enzija"], errors='coerce')
```

Za početak je preuzet potreban podniz (kao završni indeks navodi se 6 jer se znak na navedenom indeksu ne uključuje) te je vrijednost *„/A“* zamijenjena odgovarajućom vrijednosti. Zatim, kako tekstovi s na primjer jednoznamenkastom recenzijom sadrže viška slova (r, re ili rec ovisno o veličini broja) potrebno je ista slova zamijeniti s praznim stringom (treća linija koda). Kada smo istu liniju pokrenuli za sve moguće navedene kombinacije, varijabla se može konvertirati u numerički tip podataka.

```
booking_proba["Broj recenzija"].isna().sum()
Out[26]: 1384
```

Kao i kod recenzije i kategorije objekta, broj recenzija također sadrži 1384 opservacije s *null* vrijednosti što i logika nalaže s obzirom na to da hotel koji nema recenziju, neće naravno imati kategoriju (koja je osnovana na recenziji objekta) niti broj recenzija.

8) **Veličina objekta**

Veličina objekta je također u obliku teksta, s brojem i oznakom m^2 pokraj svakog broja bez iznimaka. Najveći broj unutar varijable je troznamenkasti broj te je potrebno izlučiti podniz od prvog znaka na indeksu 0 (nema suvišnih razmaka) do ne uključujućeg znaka na trećem indeksu.

```
booking_proba["Veličina objekta"] = booking_proba["Veličina objekta"].\
str[:3]
```

```
booking_proba["Veličina objekta"] = pd.to_numeric(booking_proba["Veliči
na objekta"], errors='coerce')
```

Izabran je podniz od ukupno 3 znaka te je varijabla pretvorena u numerički tip podataka.

```
booking_proba["Veličina objekta"].isna().sum()
Out[29]: 732
```

Ukupno 732 objekta nemaju pridruženu veličinu, te je odabran pristup imputacije prosjekom varijable kako bi popunili nepostojeće vrijednosti, a da distribucija podataka ostane nepromijenjena.

```
booking_proba["Veličina objekta"].fillna((booking_proba["Veličina objek
ta"].mean()), inplace=True)
```

NaN vrijednosti popunjavamo pomoću *fillna* metode gdje je korisnik slobodan navesti svojevoljnu vrijednost kojom želi popuniti navedene vrijednosti, te je u ovom slučaju odabran prosjek vrijednosti varijable *Veličina objekta*.

9) **Cijena objekta**

Cijena objekta ima oblik „*HRK x*“ (slika 21.) te broj možemo izlučiti kao podniz počevši od znaka pod indeksom 4 (prva znamenka broja) do kraja stringa.

```
booking_proba["Cijena objekta"] = booking_proba["Cijena objekta"].\
str[4:]
booking_proba["Cijena objekta"] = [x.replace('.', '') for x in \
booking_proba["Cijena objekta"]]
booking_proba["Cijena objekta"] = pd.to_numeric(booking_proba["Cijena o
bjekta"], errors='coerce')
```

Indeksiran je tekst unutar varijable i decimalna točka unutar stringa zamijenjena je praznim stringom jer se pretvorbom varijable u broj bez zamjene decimalne točke, broj kao što je 1.233 umjesto kao tisuću dvjesto trideset tri gledao kao decimalni broj 1.2. Nadalje, nakon zamjene (izbacivanja) decimalne točke, možemo varijablu pretvoriti u numerički tip podataka.

10) **Zasebne recenzije objekta**

Čišćenje varijabli recenzija osoblja, sadržaja, čistoće, udobnosti, vrijednosti, lokacije i WiFi-a opisano je zajedno s obzirom da su sve procesirane na isti način.

```
booking_proba["x"] = booking_proba["x"].replace("N/A", np.nan)
booking_proba["x"] = [str(x).replace(',', '.') for x in booking_proba["
x"]]
booking_proba["x"] = pd.to_numeric(booking_proba["x"], errors='coerce')
```

Varijabla x predstavlja sve navedene varijable. *„N/A“* vrijednosti zamijenjene su odgovarajućom vrijednosti, decimalni zarezi zamijenjeni su decimalnom točkom (kao i kod varijable *Recenzija objekta*) te su varijable pretvorene iz tipa *object* u numerički tip podataka.

```
booking_proba["Recenzija osoblja"].isna().sum(), booking_proba["Recenzi
ja sadržaja"].isna().sum(), booking_proba["Recenzija čistoće"].isna().s
um(), booking_proba["Recenzija udobnosti"].isna().sum(), booking_proba[
"Recenzija vrijednosti"].isna().sum(), booking_proba["Recenzija lokacij
e"].isna().sum(), booking_proba["Recenzija wifi"].isna().sum()

Out[31]: (1384, 1384, 1384, 1384, 1384, 1384, 1384)
```

Kao kod recenzije, kategorije i broja recenzija objekta, 1384 opservacija ima *null* vrijednosti i u navedenim varijablama jer su sve međusobno povezane.

11) **Dozvole objekta**

Dozvole objekta su u obliku rječnika:

```
{"x":x, "y":y, "z":z}
```

te je ideja ključeve rječnika izvući kao pojedinačne stupce (varijable) a vrijednosti ključa imputirati kao vrijednosti polja svake opservacije u obliku:

```
 |"x"|"y"|"z"
1| 1 | 1 | 0
2| 0 | 0 | 1
3| 1 | 0 | 1
```

gdje ćemo polja koja sadrže podniz „nije dozvoljeno“ pretvoriti u 0, a ostale u 1 (gdje se podrazumijeva ako npr. nije iskazano da pušenje nije dozvoljeno, da je dozvoljeno).

Stvaramo *pandas Series* objekt od navedene varijable

```
serija = booking_proba['Dozvole objekta'].apply(pd.Series)
```

i dobijemo slijedeće:

Za pušače	Kućni ljubimci	Zabave	Kućni red i mir
Pušenje nije dozvoljeno.	Besplatno!Dozvoljen je boravak kućnih ljubimaca i ne naplaćuje se.	nan	nan
Pušenje nije dozvoljeno.	Boravak kućnih ljubimaca nije dozvoljen.	Zabave nisu dozvoljene	Gosti ne smiju stvarati buku i 07:00.
nan	Besplatno!Dozvoljen je boravak kućnih ljubimaca i ne naplaćuje se.	nan	nan
Pušenje nije dozvoljeno.	Dozvoljen je boravak kućnih ljubim…	nan	nan
Pušenje nije dozvoljeno.	Besplatno!Na zahtjev je dozvoljen …	nan	nan
nan	Besplatno!Dozvoljen je boravak kućnih ljubimaca i ne naplaćuje se.	nan	nan
nan	Boravak kućnih ljubimaca nije dozvoljen.	nan	nan
Pušenje nije dozvoljeno.	Boravak kućnih ljubimaca nije dozvoljen.	Zabave nisu dozvoljene	nan

Slika 24. Pandas serija varijable Dozvole

Kod varijable *Za pušače* vidimo da, ako je pušenje zabranjeno u objektu, tako je i navedeno te *NaN* vrijednosti označavaju objekte koji nisu eksplicitno zabranili pušenje te se u ovom slučaju smatra da isto i dozvoljavaju. Isti je slučaj i kod varijable *Zabave,* gdje ako je navedeno da nisu dozvoljene, vrijednost je potrebno zamijeniti s 0 (*False*) dok su u suprotnom slučaju dozvoljene. Kod varijable *Kućni red i mir* korištena je ista logika uz zamjenu značenja, što znači da ako je navedena bilo koja vrijednost, istu zamjenjujemo s 1 što znači da postoji kućni red i mir u objektu koji je potrebno poštovati, u suprotnom slučaju isti ne postoji (*NaN* vrijednosti zamjenjuju se u ovom slučaju s 0). Varijabla *Kućni ljubimci* nema *NaN vrijednosti* te imamo nekoliko različitih iskaza od kojih samo jedan znači da kućni ljubimci nisu dozvoljeni te se zamjenjuju s 0, a to je iskaz „*Boravak kućnih ljubimaca nije dozvoljen.“*, te se ostale izjave odnose na dopušten boravak kućnih ljubimaca (plaćao se ili ne te bio potreban zahtjev ili ne).

```
serija["Za pušače"] = serija["Za pušače"].fillna(1)
```

```
serija["Za pušače"] = serija["Za pušače"].replace("Pušenje nije dozvolj
eno.", 0)


serija["Kućni ljubimci"] = serija["Kućni ljubimci"].replace("Boravak ku
ćnih ljubimaca nije dozvoljen.", 0)

serija["Kućni ljubimci"] = serija["Kućni ljubimci"].replace("x", 1)


serija["Zabave"] = serija["Zabave"].fillna(1)
serija["Zabave"] = serija["Zabave"].replace("Zabave nisu dozvoljene", 0
)

serija["Kućni red i mir"] = serija["Kućni red i mir"].fillna(0)
serija["Kućni red i mir"].replace(re.compile('.*stvarati.*'), 1, inplac
e=True)
```

Za pušače: *NaN* vrijednosti ispunjavamo s 1 (pušenje je dopušteno u objektu) te navedeni tekst zamjenjujemo s 0 (pušenje nije dopušteno u objektu).

Kućni ljubimci: iskaz „Boravak kućnih ljubimaca nije dozvoljen“ mijenjamo s 0 (nisu dopušteni kućni ljubimci) te ostale izjave (besplatan boravak kućnih ljubimaca bez zahtjeva, boravak je dopušten besplatno uz zahtjev, dopušten je uz zahtjev i plaćanje, dopušten je uz plaćanje) mijenjamo s 1 (na bilo koji način od navedenih, boravak kućnih ljubimaca je ipak dopušten).

Zabave: *NaN* vrijednosti mijenjamo s 1 (zabave su dopuštene) te navedeni tekst zamjenjuje s 0 (zabave nisu dozvoljene u objektu).

Kućni red i mir: *NaN* vrijednosti ispunjavamo s 0 (kućni red i mir ne postoji) te ostale izjave mijenjamo s 1 (kućni red i mir postoji unutar objekta i treba ga poštivati).

Za pušače	Kućni ljubimci	Zabave	Kućni red i mir
0	1	1	0
0	0	0	1
1	1	1	0
0	1	1	0
0	1	1	0
1	1	1	0
1	0	1	0
0	0	0	0
0	1	1	0
0	1	1	0

Slika 25. Uređena serija

Sada kada imamo uređenu seriju, možemo je spojiti sa originalnim skupom podataka.

```
booking_final = pd.concat([booking_proba, serija], axis = 1).drop('Dozv
ole objekta', axis = 1)
```

Spajanje je izvršeno pomoću *pandas* funkcije *concat* gdje navodimo skup podataka *booking_proba* i navedenu uređenu *pandas* seriju te pomoću funkcije *drop* izbacujemo varijablu *Dozvole objekta* s obzirom na to da nam ista više nije potrebna.

booking_final - DataFrame

Index	a lokacije	Recenzija wifi	Popularni sadržaji	Za pušače	Kućni ljubimci	Zabave	Kuć
0		nan	['besplatno …	0	1	1	0
1		10	['besplatno …	0	0	0	1
2		7.5	['besplatno …	1	1	1	0
3		6.7	['Besplatni WiFi']	0	1	1	0
4		7.5	['besplatno …	0	1	1	0
5		2.5	['bar']	1	1	1	0
6		9.7	['bar', 'Vrl…	1	0	1	0
7		nan	['Dozvoljeni…	0	0	0	0
8		10	['bar']	0	1	1	0

Slika 26. Spajanje serije sa skupom podataka

Na slici 26. vidimo da je prethodno uređena *pandas* serija uspješno spojena sa skupom podataka.

12) **Popularni sadržaji**

Popularni sadržaji su u obliku liste:

```
[x, y, z]
```

gdje je ideja sve postojeće elemente listi pretvoriti u stupce i kao vrijednost opservaciji pridodati 1 ako se element nalazi u listi opservacije i 0 ako opservacija unutar liste ne sadrži određeni element. Grafički, uzmemo li u obzir objekte 1, 2 i 3 s listama:

```
1 [x, y]
2 [x]
3 [x, y, z]
```

stupci i odgovarajuće vrijednosti bi trebali izgledati ovako:

```
 |"x"|"y"|"z"
1| 1 | 1 | 0
2| 1 | 0 | 0
3| 1 | 1 | 1
```

gdje vidimo da objekt 1 ima vrijednosti 1 pod stupcima „x“ i „y“ jer iste elemente sadrži u originalnoj listi dok element „z“ ne sadrži te je pripadajuća vrijednost za element „z“ 0. Objekt 2 sadrži samo element „x“, dok objekt 3 sadrži sva 3 elementa.
Za početak, objekti koji su inicijativno imali prazne liste pod varijablom *Popularni sadržaji* (obrađeno u poglavlju 2.4.2.), sadrže drugačije imenovane elemente za parking i WiFi te je potrebno prvo preimenovati elemente kako ne bi imali dva suvišna stupca koji se odnose na iste elemente.

```
booking_final["Popularni sadržaji"] = [[x.replace('Parkiralište','bespl
atno parkiralište') \
for x in i] for i in booking_final["Popularni sadržaji"]]
```

Potrebno je pristupiti elementima unutar liste, što je potrebno učiniti kroz obuhvaćanje liste (engl. *List comprehension*) gdje pomoću *for* petlje pristupamo svakom pojedinačnom elementu unutar svake liste navedene varijable i mijenjamo potrebne vrijednosti. Isti proces proveden je za string „*Internet“,* koji je preimenovan u „*besplatan wifi“*.

```
booking_zadnji = pd.concat([booking_final, pd.get_dummies(booking_final
['Popularni sadržaji'].apply(pd.Series))], axis=1)
```

Pomoću *pandas* funkcije *concat* spajamo zadnje uređeni skup podataka *booking_final* i *pandas Series* objekt napravljen, ovog puta od varijable *Popularni sadržaji* koji je pomoću *pandas* funkcije *get_dummies* pretvoren u binarne varijable (engl. *Dummy variables*).

Nakon spajanja, dobili smo 23 dodatne varijable: dostupan sadržaj za invalide, dostupan bazen, dostupan aparat za kavu/čaj, dostupan spa tretman, besplatan parking, dostupan jacuzzi, moguća posluga u sobu, prisutnost dizala, dostupnost posebnih soba za pušače, dostupna praonica rublja, dostupna oprema za roštilj, moguće grijanje, recepcija dostupna 24 h/dan, plaža u blizini objekta, dostupna terasa u objektu, dostupan prijevoz od/do zračne luke, dostupni restorani u blizini objekta, dostupne obiteljske sobe, besplatan WiFi, dostupan fitness centar, klima uređaj u objektu, vrt u sklopu objekta i dostupan doručak u sklopu smještaja. Broj objekata u kojima postoje dotični sadržaji možemo provjeriti na sljedeći način:

```
skup_podataka["varijabla"].value_counts()
```

S obzirom na to da su neke od varijabli dostupne u veoma malenom broju objekata (kao što su jacuzzi, spa, fitness centar, praonica rublja i slični) iste su izbačene iz skupa podataka jer ne sadrže relevantan broj opservacija s pozitivnim vrijednostima za navedene sadržaje te su na kraju, u sklopu skupa podataka, ostavljene sljedeće varijable: bazen, besplatni WiFi, doručak, bar, prijevoz od/do zračne luke, besplatan parking, plaža, obiteljske sobe i prostor za pušače. Kriterij za ostavljanje varijabli bio je da sadržaj sadrži barem 300 opservacija s pozitivnom vrijednosti (što znači da bar 300 objekata ima određeni sadržaj) u protivnom je uzorak objekata koji imaju određeni sadržaj premalen.

Index	Kućni red i mir	Bazen	Besplatni Wifi	Doručak	Bar	jevoz do zračne
0		0	0	0	0	0
1		0	0	0	0	0
2		0	0	1	1	0
3		0	1	0	0	0
4		0	0	1	1	0
5		0	0	0	1	0
6		0	0	1	1	0
7		0	1	0	0	0

Slika 27. Skup podataka sa spojenim popularnim sadržajima

Na slici 27. prikazan je dio skupa podataka gdje se može vidjeti da su varijable ispravno pridružene navedenom skupu.

13) **Pridodavanje novih varijabli – županija i regija**

Dosadašnja verzija skupa podataka trenutno sadrži samo jednu varijablu vezanu za samu lokaciju objekta – grad. Kada provjerimo broj objekata unutar svakog grada dobijemo:

```
booking["Lokacija (grad)"].value_counts()
Out[5]:
Zagreb         641
Dubrovnik      504
Zadar          406
Split          270
Pula           183
               ...
Vardarac         1
Gromača          1
Rakotule         1
Cestica          1
Name: Lokacija (grad), Length: 738, dtype: int64
```

Kao što vidimo, postoji 738 različitih gradova unutar skupa podataka, te ako pogledamo koliko njih sadrži po samo jedan objekt vidimo:

```
((booking["Lokacija (grad)"].value_counts()) == 1).sum()
Out[7]: 302
```

Od 738 gradova, njih 302 (što je skoro polovica ukupnog broja gradova) sadrži samo jedan objekt unutar skupa podataka te je potrebno objekte grupirati u veće cjeline kako bi, daljnjim postupkom analize, imali veće cjeline lokacije objekata koje sadrže značajan broj objekata na temelju kojih bi mogli analizirati karakteristike pojedine lokacije. U svrhu ove spoznaje, odlučeno je da će se objekti grupirati u cjeline županije, a zatim i u regije kako bi imali veće (regije) i manje (županije) grupe objekata gdje ćemo dobiti značajnije cjeline s određenim brojem objekata koje ćemo moći analizirati.

Pretraživanjem interneta, pronađena je Excel tablica koja sadrži županije hrvatske i njima pripadajuće gradove i općine (https://mpu.gov.hr/o-ministarstvu/ustrojstvo/uprava-za-politicki-sustav-i-opcu-upravu/lokalna-i-podrucna-regionalna-samouprava/popis-zupanija-gradova-i-opcina/22319). Tablica je pronađena na web stranici ministarstva pravosuđa i uprave i moguće ju je skinuti na računalo.

Nakon skidanja iste na računalo, moguće ju je učitati u Python pomoću *pandas* funkcije *read_excel.*

```
zupanije = pd.read_excel('C:/Users/38591/Downloads/zupanije i gradovi.xls')
```

Unutar spomenute funkcije potrebno je samo navesti putanju (direktorij skinute datoteke uz ime datoteke) tablice te je zatim ista učitana u radno okruženje Spyder. Nakon učitavanja, tablica izgleda ovako:

zupanije - DataFrame

Index	Zupanija	ip jedinic	Ime
0	I ZAGREBAČKA	Općina	Bedenica
1	I ZAGREBAČKA	Općina	Bistra
2	I ZAGREBAČKA	Općina	Brckovljani
3	I ZAGREBAČKA	Općina	Brdovec
4	I ZAGREBAČKA	Općina	Dubrava
5	I ZAGREBAČKA	Općina	Dubravica

Slika 28. Neuređena tablica sa županijama i gradovima države

Prije spajanja tablice sa skupom podataka, potrebno je obaviti nekoliko stvari:

1) urediti zapis same županije – želimo naziv županije bez rimskog broja napisan malim slovima.

```
zupanije["Zupanija"] = zupanije["Zupanija"].replace("XXI GRAD ZAGREB",
"Grad Zagreb")
```

Postupak prikazan za grad Zagreb proveden je za sve županije unutar tablice.

2) izbaciti stupac *Tip jedinice* s obzirom na to da nas ne zanima da li se radi o gradu ili općini.

```
zupanije.drop('Tip jedinice', axis = 1, inplace = True)
```

3) postaviti varijablu *Ime* kao indeks tablice s obzirom na to da ćemo spajanje izvršiti putem indeksa tablice (koji je ujedno i jedinstven).

```
zupanije = zupanije.set_index("Ime")
```

Nakon provođenja navedenih postupaka, tablica izgleda ovako:

Ime	Zupanija
Bedenica	Zagrebačka
Bistra	Zagrebačka
Brckovljani	Zagrebačka
Brdovec	Zagrebačka
Dubrava	Zagrebačka
Dubravica	Zagrebačka
Dugo Selo	Zagrebačka

Slika 29. Uređena tablica sa županijama

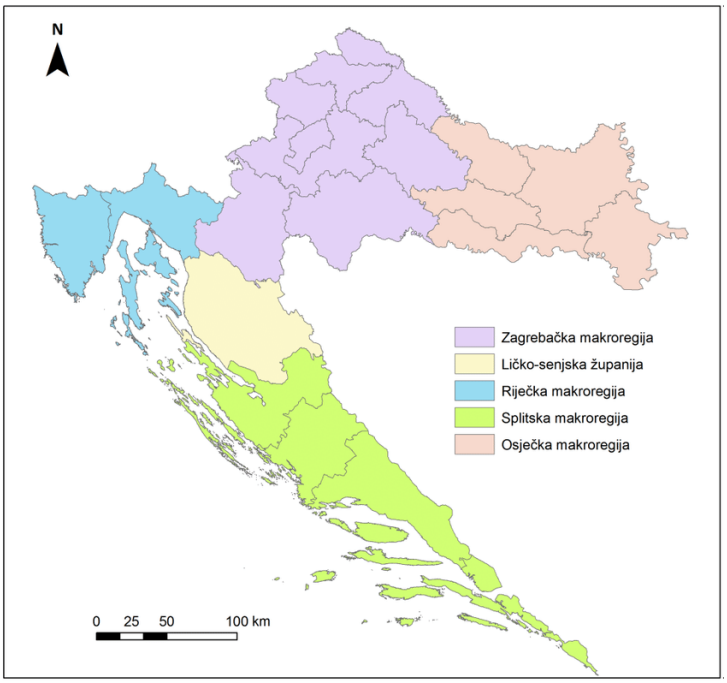
Nakon uređivanja tablice, možemo ju spojiti sa *booking* skupom podataka.

```
booking = booking_zadnji.merge(zupanije, how = "left", left_on =  "Loka
cija (grad)", right_on = zupanije.index)
```

Za spajanje skupova podataka može se koristiti *pandas* funkcija *merge*, koju pozivamo nad skupom podataka na koji želimo spojiti drugi, koji se navodi unutar funkcije, kao i tip spajanja (u ovom slučaju je *left* s obzirom na to da želimo zadržati sve opservacije i varijable lijevog skupa podataka, što je booking skup podataka) te i nad kojim varijablama se obavlja spajanje u oba skupa podataka (s obzirom na to da se ne radi u istoimenoj varijabli). Parametar *left_on* označava varijablu koja se koristi za spajanje skupova unutar skupa *booking*, a *right_on* označava varijablu koja se koristi za spajanje skupova unutar skupa podataka *zupanije*, što je indeks samog *DataFrame-a* kojeg čime imena gradova/općina.

Istraživanjem *null* vrijednosti unutar varijable *Zupanija* otkriveno je da postoji nekoliko gradova (točnije općina) kojima nije pridružena županija unutar tablice županija i gradova, no postoje u skupu podataka sa podacima s booking-a te su iste ispunjene ručno.

Kao predložak regija, županije su grupirane prema slici iz znanstvenog istraživanja [2].



Slika 30. Regije hrvatske, preuzeto sa: https://www.researchgate.net/profile/Neven-Tandaric/publication/259219848/figure/fig1/AS:651208582766592@1532271633198/figure-fig1.png

```
centralne = ["Međimurska", "Varaždinska", "Koprivničko križevačka", "Kr
apinsko zagorska", "Zagrebačka", \
"Grad Zagreb", "Bjelovarsko bilogorska", "Sisačko moslovačka", "Karlova
čka"],slavonija = ["Virovitičko podravska", "Požeško slavonska", "Brods
ko posavska", "Osječko baranjska", "Vukovarsko srijemska"]
planina = ["Primorsko goranska", "Ličko senjska"], istra = ["Istarska"]
```

Napravljene su četiri liste županija za četiri regije:

Centralna hrvatska: Međimurska županija, Varaždinska županija, Koprivničko-križevačka županija, Krapinsko-zagorska županija i Zagrebačka županija.

Slavonija: Virovitičko-podravska županija, Požeško-slavonska županija, Brodsko-posavska županija, Osječko-baranjska županija i Vukovarsko-srijemska županija.

Planinska hrvatska: Primorsko-goranska županija i Ličko-senjska županija.

Istra: Istarska županija.

Ostale županije pripadaju Dalmaciji.

```
def rezultat(df):
    if df["Zupanija"] in (centralne):
        return "Centralna"
    elif df["Zupanija"] in (slavonija):
        return "Slavonija"
    elif df["Zupanija"] in (planina):
        return "Planinska"
    elif df["Zupanija"] in (istra):
        return "Istra"
    else:
        return "Dalmacija"

booking["Regija"] = booking.apply(rezultat, axis = 1)
```

Napisana je funkcija koja kao parametar prima *Data Frame* i provjerava kojoj listi od prethodno izrađenih pripada vrijednost za pojedini redak pod varijablom *Zupanija,* te ako ne pripada niti jednoj od navedenih, vraća regiju *„Dalmacija“*. Zatim novoj varijabli *Regija* unutar booking skupa podataka pridružujemo vrijednosti vraćene od strane navedene funkcije za svaki redak unutar skupa podataka koristeći se funkcijom *apply* koja aplicira navedenu funkciju na skup podataka nad kojim je pozvana.

```
cols = ['Zupanija', 'Regija']
booking[cols] = booking[cols].astype('category')
```

Konačno, dvije uvedene varijable pretvaramo iz tipa podataka *object* u kategorije te

```
booking = booking.drop_duplicates()
```

izbacujemo sve duplikate (redove s istim vrijednostima u svim stupcima skupa podataka) kako bi na kraju dobili skup podataka sa 8 113 opservacija (objekata) i 31 stupcem.

Pozivanjem metode *info* provjeravamo vrste podataka unutar varijabli:

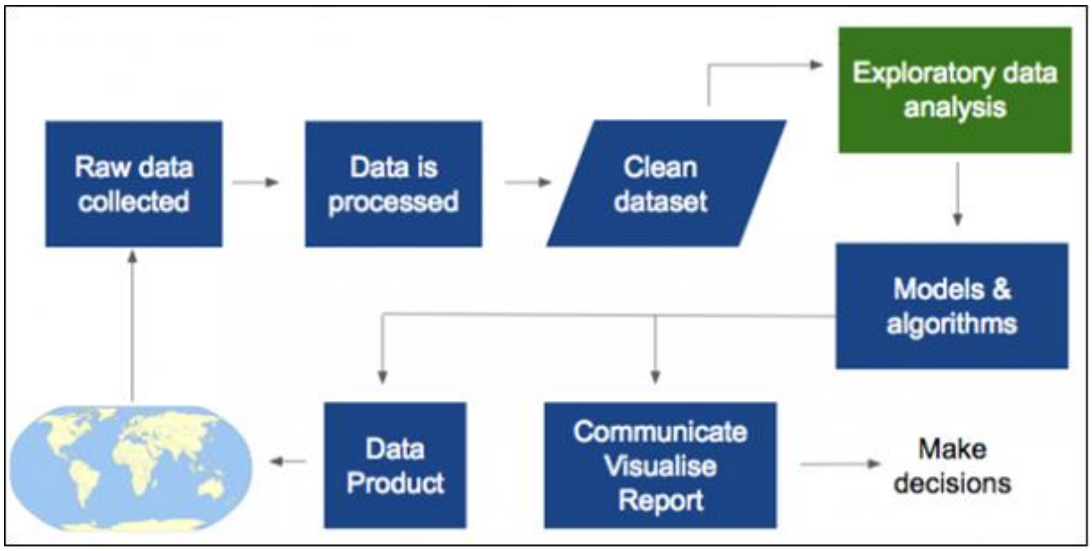
```
booking.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8113 entries
Data columns (total 31 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Naziv objekta              8113 non-null   object
 1   Vrsta objekta              8113 non-null   category
 2   Lokacija (grad)            8113 non-null   object
 3   Kategorizacija objekta     6660 non-null   float64
 4   Recenzija objekta          6729 non-null   float64
 5   Kategorija ocjene objekta  6729 non-null   category
 6   Broj recenzija             6729 non-null   float64
 7   Veličina objekta           8113 non-null   int32
 8   Cijena objekta             8113 non-null   int32
 9   Recenzija osoblja          6729 non-null   float64
 10  Recenzija sadržaja         6729 non-null   float64
 11  Recenzija čistoće          6729 non-null   float64
 12  Recenzija udobnosti        6729 non-null   float64
 13  Recenzija vrijednosti      6729 non-null   float64
 14  Recenzija lokacije         6729 non-null   float64
 15  Recenzija wifi             6729 non-null   float64
 16  Za pušače                  8113 non-null   int32
 17  Kućni ljubimci             8113 non-null   int32
 18  Zabave                     8113 non-null   int32
 19  Kućni red i mir            8113 non-null   int32
 20  Bazen                      8113 non-null   int32
 21  Besplatni Wifi             8113 non-null   int32
 22  Doručak                    8113 non-null   int32
 23  Bar                        8113 non-null   int32
 24  Prijevoz do zračne luke    8113 non-null   int32
 25  Parking                    8113 non-null   int32
 26  Plaža                      8113 non-null   int32
 27  Obiteljske sobe            8113 non-null   int32
 28  Prostor za pušače          8113 non-null   int32
 29  Zupanija                   8113 non-null   category
 30  Regija                     8113 non-null   category
dtypes: category(4), float64(10), int32(15), object(2)
```

Možemo vidjeti da su uspješno promijenjene vrste podataka te da imamo 4 *category* (kategorije) varijable (vrsta objekta, kategorija objekta, županija i regija), 10 *float* (decimalni

brojevi) varijable (zvjezdice objekta, recenzije objekta, broj recenzija te 7 varijabli vezanih uz posebne recenzije sadržaja smještaja), 15 *int* (cijeli brojevi) varijabli (što su sve binarne varijable proizvedene iz dozvola i popularnih sadržaja uz veličinu i cijenu objekta) te 2 *object* (string) varijable što su naziv objekta i lokacija (grad) objekta.

4. Istraživačka analiza podataka

Istraživačka analiza podataka (*engl. Exploratory data analysis, EDA*) predstavlja pristup analize skupa podataka koji sumira glavne karakteristike navedenih podataka, koristeći se statističkim i drugim vizualizacijskim metodama. Ova vrsta analize predstavlja prvi korak analize podataka određenog uzorka čiji su glavni razlozi otkrivanje pogrešaka u podacima, provjera pretpostavki koje imamo o podacima te otkrivanje veza između neovisnih varijabli i ciljne varijable. Podaci su uglavnom spremljeni u obliku tablice (koja potječe iz excel proračunske tablice ili baze podataka) gdje svaka tablica sadrži redove koji prikazuju podatke određene instance pregleda (u ovom slučaju to su objekti smještaja) i stupce kao identifikatore čimbenika (u ovom slučaju cijena smještaja, veličina, recenzija i druge) koje pripadaju neovisnim varijablama ili ciljnoj varijable (u ovom slučaju klasa objekta). Svaki stupac sadrži numeričke vrijednosti za određenu kvantitativnu varijablu ili razine za kategoričke varijable. Kako čovjek najčešće nije u mogućnosti odrediti važne karakteristike podataka iz samog pregleda tablica (posebno onih većih), tehnike istraživačke analize strukturirane su tako da omogućavaju korisniku pregled važnih karakteristika podataka pomoću statistike i vizualizacija, gdje je moguće istaknuti važne aspekte navedenih podataka [3]. Na slici 31. prikazan je, u koracima, proces analize podataka gdje vidimo da je prije izgradnje modela i primjene algoritama potrebno istražiti same podatke kako bi imali uvid u njihove karakteristike.



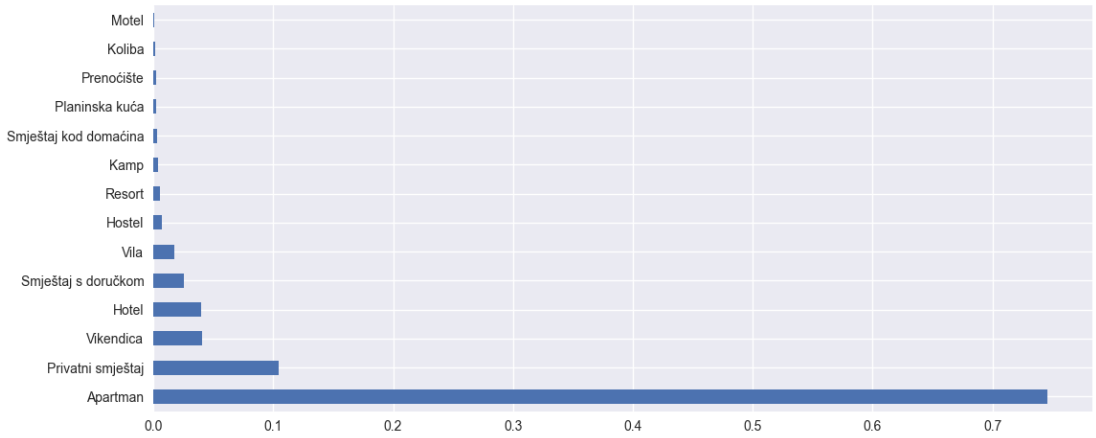
Slika 31. Proces analize podataka, preuzeto sa: https://dataingovernment.blog.gov.uk/wp-content/uploads/sites/46/2016/08/Data-Science-Process-5-620x309.png

Varijable skupa podataka analizirat ćemo redom kako se pojavljuju u skupu. Naziv objekta jedinstven je za svaki objekt te ne postoje informacije koje nas zanimaju posebno oko imena samog smještaja, Sljedeća varijabla je vrsta objekta, koja označava tip smještaja. Pomoću ove varijable možemo odgovoriti na nekoliko pitanja:

Vrsta objekta – koliko opservacija se nalazi unutar svake kategorije vrste smještaja?

```
booking["Vrsta objekta"].value_counts(normalize=True).plot.barh()
```

Value_counts je funkcija pomoću koje prebrojavamo koliko redova (opservacija, instanci) pripada određenoj kategoriji unutar varijable. Navođenjem parametra *normalize = True* navodimo da želimo postotak opservacija unutar kategorija umjesto samog broja opservacija. Nad navedenim izračunom možemo pozvati funkciju *plot.barh* kako bi vizualizirali navedene podatke.



Slika 32. Postotak opservacija unutar svake kateogije vrste objekta

Kao što je vidljivo sa slike 32., najveći postotak opservacija (74 %) pripada kategoriji apartmana, zatim privatnom smještaju (12 %), hotelima (5 %) itd.

Vrsta objekta – koliko iznosi prosječna cijena i recenzija za svaku kategoriju?

```
booking.groupby("Vrsta objekta").agg(Prosječna_cijena = ("Cijena objekt
a", "mean"), Prosjecna_recenzija = ("Recenzija objekta", "mean"))
```

Kako bi saznali odgovor na navedeno pitanje, potrebno je podatke grupirati prema vrsti kategorije i agregirati tako da navodimo varijable za koje nas zanima prosjek (u ovom slučaju to su varijable *Cijena objekta* i *Recenzija objekta*) i statističku funkciju koju želimo primijeniti nad navedenim varijablama (u ovom slučaju to je *mean* za prosjek, no može biti i minimum, maksimum, medijan,...).

Tablica 1. Prosječna cijena i recenzija prema vrsti objekta

Vrsta objekta	Prosječna_cijena	Prosjecna_recenzija
Apartman	311.305	8.775
Hostel	330.778	8.75333
Hotel	388.912	8.98076
Kamp	424.167	7.48333
Koliba	444.76	9.26
Motel	460.585	8.99588
Planinska kuća	533.723	9.21223
Prenoćište	570.6	9.35
Privatni smještaj	670.713	8.70823
Resort	705.067	9.24231
Smještaj kod domaćina	705.085	8.71333
Smještaj s doručkom	812.095	9.55882
Vikendica	964.924	9.35862
Vila	1878.65	9.27708

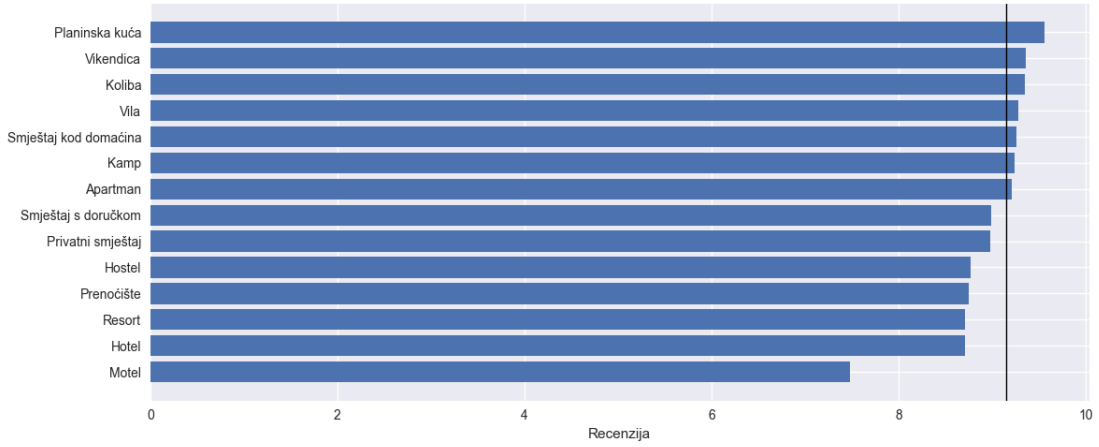
Na tablici 1. prikazana je sumirana statistika prema vrsti objekta. Varijable su sortirane uzlazno prema cijeni smještaja, te možemo vidjeti da apartmani, uz zatim hostele i hotele u prosjeku imaju najnižu cijenu smještaja, dok smještaj s doručkom, vikendica i zatim vila u prosjeku imaju najvišu cijenu smještaja. Ako gledamo recenzije, možemo vidjeti da najnižu recenziju u prosjeku imaju kampovi (7.5) dok je sljedeća najniža recenzija 8.7 koja pripada apartmanima i hostelima, dok najvišu prosječnu recenziju imaju kolibe, vikendice i smještaji s doručkom.

Vrsta objekta – koje vrste objekta imaju iznadprosječnu recenziju?

```
df = booking.groupby("Vrsta objekta")["Recenzija objekta"].mean()
.reset_index()
df.sort_values('Recenzija objekta',inplace=True)
plt.barh(df["Vrsta objekta"], df["Recenzija objekta"])
prosjek = booking["Recenzija objekta"].mean()
plt.axvline(x=prosjek,linewidth=1, color='k')
plt.xlabel("Recenzija")
```

Prvi korak je grupiranje skupa podataka u odnosu na vrstu objekta te izračun prosječne recenzije za pojedinu vrstu objekta. Nakon sortiranja varijabli silazno, vizualiziramo vrstu

objekta u odnosu na recenzije te izračunamo prosjek svih recenzija koji zatim pridodajemo izrađenom grafikonu.



Slika 33. Recenzije u odnosu na vrstu objekta

Na slici 33. prikazan je odnos vrste objekata i recenzije objekta. Crnom linijom povučen je generalni prosjek recenzije cijelog skupa podataka te možemo vidjeti da planinska kuća, vikendica, koliba, vila, smještaj kod domaćina, kamp i apartman sadrže prosječnu recenziju iznad prosjeka, dok ostale kategorije imaju prosječnu recenziju nižu od prosjeka.

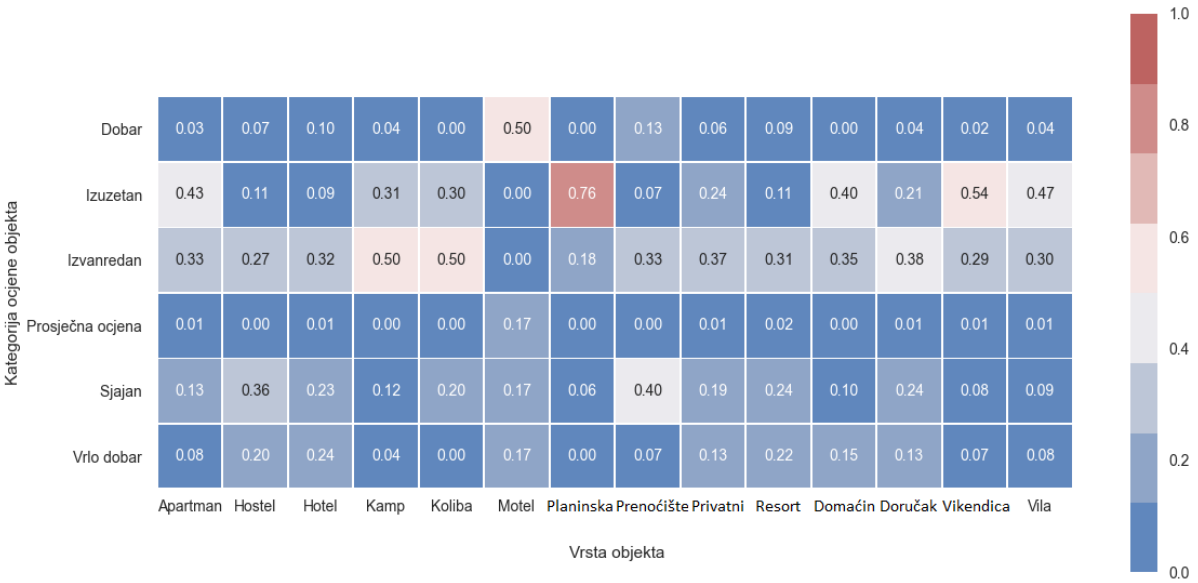
Vrsta objekta – koja je najčešća klasifikacija objekta unutar pojedine vrste objekta?

```
df = booking.groupby(["Vrsta objekta", "Kategorija ocjene objekta"]).si
ze().reset_index(name = "Broj")
df["Postotak"] = df['Broj'] / df.groupby('Vrsta objekta')['Broj'].trans
form('sum')
```

Kako bi odgovorili na ovo pitanje, stvarano novi skup podataka koji se sastoji od *booking* skupa podataka grupiranog prema vrsti objekta i klasi objekta te pomoću funkcije *size* dobijemo broj objekta svake klase unutar svake vrste hotela (npr. jedan red ovog skupa podataka izgledao bi ovako: *apartman, izuzetan, 104*). Postotak kao novu varijablu moguće je izračunati tako da broj opservacija s određenom klasom podijelimo s ukupnim brojem opservacija svake klase unutar određene vrste objekta.

```
df1 = df.pivot(columns='Vrsta objekta', index='Kategorija ocjene objekt
a', values="Postotak")
colormap = sns.color_palette("vlag", 8)
h = sns.heatmap(df1, annot=True, cmap=colormap, linewidth=.5, square =
True, linecolor='w', fmt='.2f', vmin=0, vmax=1, center=0.5)
h.set_xticklabels(h.get_xticklabels(), rotation=0)
```

Da bi napravili vizualizaciju u obliku matrice potrebno je pomoću funkcije *pivot* stvoriti pivot tablicu originalne tablice gdje će stupci predstavljati vrstu objekta, redovi klase objekta te sama vrijednost unutar polja (presjeka) predstavljat će izračunati prosjek. Za vizualizaciju koristi se *seaborn* funkcija *heatmap* koja kao obavezan parametar uzima pivot tablicu te neke opcionalne parametre kao što su širina linija matrice, minimalna, centralna i maksimalna vrijednost, broj decimala, itd. Funkcija *set_xticklabel* iskorištena je kako bi izravnali tekst duž x osi matrice (rotacija je jednaka nuli).



Slika 34. Postotak opservacija svake klase unutar pojedine vrste objekta

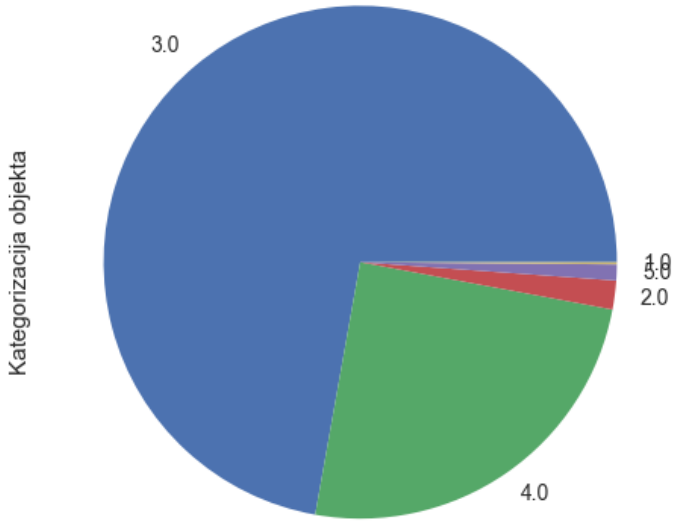
Pregledom slike 34. možemo primijetiti postotak svake klase unutar pojedine vrste objekta. Tako npr. možemo vidjeti da su apartmani najčešće klasificirani kao izuzetni, hosteli kao sjajni, hoteli kao izvanredni, kamp i koliba kao izvanredni itd. Klasa *Prosječna ocjena* sadrži najniže recenzije te nije najučestalija klasa niti jedne vrste objekta, sljedeća je *Dobar* što je najčešća ocjena motela. Najbolja klasa je *Izuzetan* što je najčešća klasa među apartmanima, planinskim kućama, smještaju kod domaćina, vikendicama te vilama.

Grad objekta je varijabla s mnogo klasa, točnije 590, te smo iz tog razloga pomoću gradova instance klasificirani u županije i gradove kako bi mogli adekvatno izvršili analizu na temelju regije te se varijabla s nazivom grada preskače. Sljedeća varijabla *Kategorizacija objekta* označava broj zvjezdica hotela i nju možemo analizirati.

Kategorizacija objekta – koliki je udjel svakog broja zvjezdica unutar skupa podataka?

```
booking["Kategorizacija objekta"].value_counts().plot.pie()
```

Pomoću iste funkcije pronalazimo broj vrijednosti za svaku kategoriju te pozivamo funkciju *plot.pie* za vizualizaciju podataka.



Slika 35. Udio objekata sa pojedinim brojem zvjezdica

Sa slike 35. možemo vidjeti da najveći broj objekata unutar skupa podataka ima pridružene 3 zvjezdice, zatim 4 zvjezdice, te tek neznatan broj opservacija ima pridruženo, 1,2 ili 5 zvjezdica.

Kategorizacija objekta – postoji li razlika u kategorizaciji prema vrsti smještaja?

```
booking.groupby("Vrsta objekta")["Kategorizacija objekta"].mean().round
(0)
```

```
Out[4]:
Vrsta objekta
Apartman                  3.0
Hostel                    3.0
Hotel                     4.0
Kamp                      4.0
Koliba                    NaN
Motel                     3.0
Planinska kuća            3.0
Prenoćište                4.0
Privatni smještaj         3.0
Resort                    4.0
Smještaj kod domaćina     3.0
Smještaj s doručkom       3.0
```

```
Vikendica                3.0
Vila                     4.0
```

Isti rezultat dobiven je korištenjem funkcije *median*. Prema rezultatima, možemo vidjeti da niti jedna koliba nema pridružene zvjezdice, te da većina vrsta objekata ima upravo i ocjenu koja je najčešća u skupu podataka (3) dok iznimke imaju višu od prosjeka (4) a to su hoteli, kampovi, prenoćišta, resorti i vile.

Kategorizacija objekta – postoji li razlika između klasa objekta?

Nakon utvrđivanja razlika između vrsta smještaja, zanima nas da li postoji razlika u kategorizaciji i kod samih klasa hotela, npr. da li najbolja klasa ima isti prosječan broj zvjezdica kao i ona najniža – *prosječna ocjena.*

```
booking.groupby("Kategorija ocjene objekta")["Kategorizacija objekta"].
median().round(0)
Out[7]:
Kategorija ocjene objekta
Dobar                3.0
Izuzetan             3.0
Izvanredan           3.0
Prosječna ocjena     3.0
Sjajan               3.0
Vrlo dobar           3.0
```

Kao i kod prethodne linije koda, isti rezultat dobiven je pomoću funkcije *median* i *mean* što je provjereno kako bi utvrdili da ne dolazimo do lažnih zaključaka. Iz rezultata možemo zaključiti da ne postoje veće razlike u kategorizaciji objekta na razini klasa tj. da su klase neovisne o kategorizaciji objekta (broju zvjezdica hotela). Isti test proveden je za županije:

```
Bjelovarsko bilogorska    3.0
Brodsko posavska          3.0
Dubrovačko neretvanska    3.0
Grad Zagreb               3.0
Istarska                  3.0
Karlovačka                3.0
Koprivničko križevačka    3.0
Krapinsko zagorska        3.0
Ličko senjska             3.0
Međimurska                3.0
Osječko baranjska         3.0
Požeško slavonska         3.0
```

```
Primorsko goranska        3.0
Sisačko moslovačka        3.0
Splitsko dalmatinska      3.0
Varaždinska               3.0
Virovitičko podravska     3.0
Vukovarsko srijemska      3.0
Zadarska                  3.0
Zagrebačka                3.0
Šibensko kninska          3.0
```

gdje vidimo da razlika ne postoji također niti na razini županije, odnosno lokacije.

Kategorizacija objekta – postoji li razlika između recenzija za objekte pojedine kategorizacije?

```
booking.groupby("Kategorizacija objekta")["Recenzija objekta"].mean().r
ound(2).sort_values()
```

Nakon što smo podatke grupirali prema boju zvjezdica objekta, izračunavamo prosjek recenzija za pojedinu kategoriju koji zaokružujemo na dvije decimale te sortiramo uzlazno prema vrijednosti prosjeka.

```
Kategorizacija objekta
1.0    8.66
2.0    8.78
3.0    9.11
5.0    9.27
4.0    9.32
```

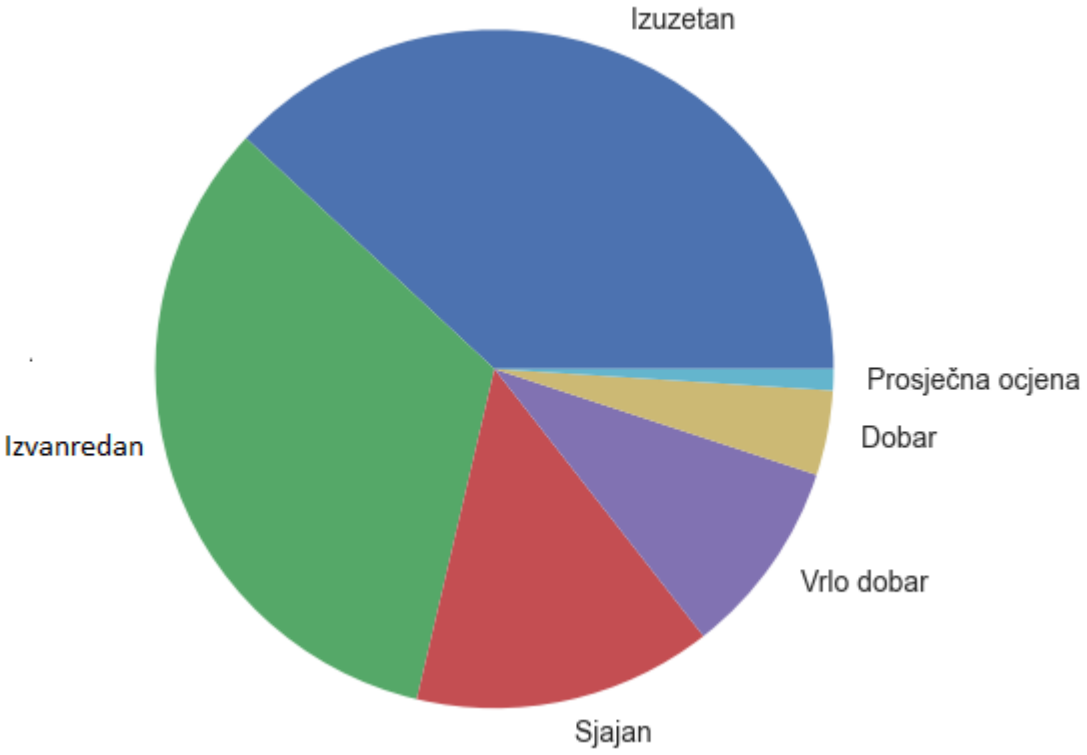
Iz rezultata vidimo da viša kategorizacija objekta i ima višu recenziju te možemo primijetiti pozitivnu korelaciju između navedene dvije varijable, uz izuzetak da objekti s kategorizacijom 4 imaju viši prosjek recenzija nego oni s kategorizacijom 5, ali tome uzrok može biti u znatno manjem broju objekata s kategorizacijom 5.

Recenzija objekta, kao i veličina, cijena i slične varijable su kontinuirane varijable i njihov odnos, kao što smo već kod nekih pitanja promatrali, promatramo u odnosu na druge varijable, ne posebno s obzirom da postoji u teoriji beskonačan niz vrijednosti unutar navedenog stupca i ne možemo promatrati karakteristike navedenih varijabli zasebno jer ne postoje definirane kategorije.

Klasa objekta – koliki je udjel svake klase unutar skupa podataka?

```
booking["Kategorija ocjene objekta"].value_counts().plot.pie()
```

Kao i inače, kako bi provjerili udio kategorija unutar skupa podataka pozivamo funkciju *value_counts* i vizualiziramo pomoću *plot* funkcije.



Slika 36. Udio klasa unutar skupa podataka

Sa slike 36. možemo vidjeti da najveći postotak objekata pripada klasama *Izvanredan* i *Izuzetan*, zatim klasi *Sjajan* i *Vrlo dobar* te vrlo mali dio objekata pripada klasama *Dobar* i *Prosječna ocjena.* Ovakva raspodjela klasa naziva se nebalansiranom, te će navedeni problem biti riješen prije upotrebe algoritama strojnog učenja kako bi model uspješno istrenirali, tj. kako bi za predviđanje klasa imao dostupan jednak broj opservacija svake klase.

Klasa objekta – koji je raspon recenzija unutar pojedine klase objekta?

```
df = booking.groupby("Kategorija ocjene objekta").agg(Min = ("Recenzija
 objekta", "min"), Max = ("Recenzija objekta", "max"))
```

Kako bi pronašli raspon unutar svake klase, potrebno je grupirati skup podataka prema klasi objekta te pronaći minimalni i maksimalnu recenziju za svaku klasu objekta.

Tablica 2. Raspon recenzija unutar klasa objekta

Kategorija ocjene objekta	Min	Max
Dobar	7	7.9
Izuzetan	9.5	10
Izvanredan	9	9.4
Prosječna ocjena	4.9	6.9
Sjajan	8.6	8.9
Vrlo dobar	8	8.5

Na tablici 2. možemo vidjeti da su kategorije, poredane redom od kategorija sa najmanjom recenzijom do kategorije s najvećim recenzijama, sljedeće:

1) **Prosječna ocjena –** klasa hotela sa najnižom recenzijom unutar raspona od 4.9 do 6.9.

2) Dobar – druga po redu klasa s recenzijom unutar raspona od 7 do 7.9.

3) Vrlo dobar – treća po redu klasa s recenzijom unutar raspona od 8 do 8.5.

4) Sjajan – četvrta po redu klasa s recenzijom unutar raspona od 8.6 do 8.9.

5) Izvanredan – peta po redu klasa s recenzijom unutar raspona od 9 do 9.4.

6) Izuzetan – klasa hotela s najvišim recenzijama unutar skupa od 9.5 do 10.

Klasa objekta – postoji li razlika u broju recenzija u odnosu na pojedine klase?

```
booking.groupby("Kategorija ocjene objekta")["Broj recenzija"].mean().r
ound(2).sort_values()
```

Nakon grupiranja skupa podataka prema klasi, gledamo prosječan broj recenzija pojedine klase.

```
Prosječna ocjena          60.93
Izuzetan                  68.61
Dobar                     95.36
Sjajan                   104.02
Izvanredan               104.12
Vrlo dobar               129.63
```

Prema rezultatima vidimo da objekti s klasom *Prosječna ocjena* imaju u prosjeku najmanje recenzija (~ 61) dok su objekti klase *Vrlo dobar* najčešće recenzirani (u prosjeku ~ 129 recenzija po objektu).

Klasa objekta – postoji li razlika u veličini objekta u odnosu na klase objekta?

```
booking.groupby("Kategorija ocjene objekta")["Veličina objekta"].mean()
.sort_values()
```

Kao i kod prethodnog pitanja, grupiramo podatke prema klasi te tražimo prosječnu veličinu objekta unutar pojedine klase.

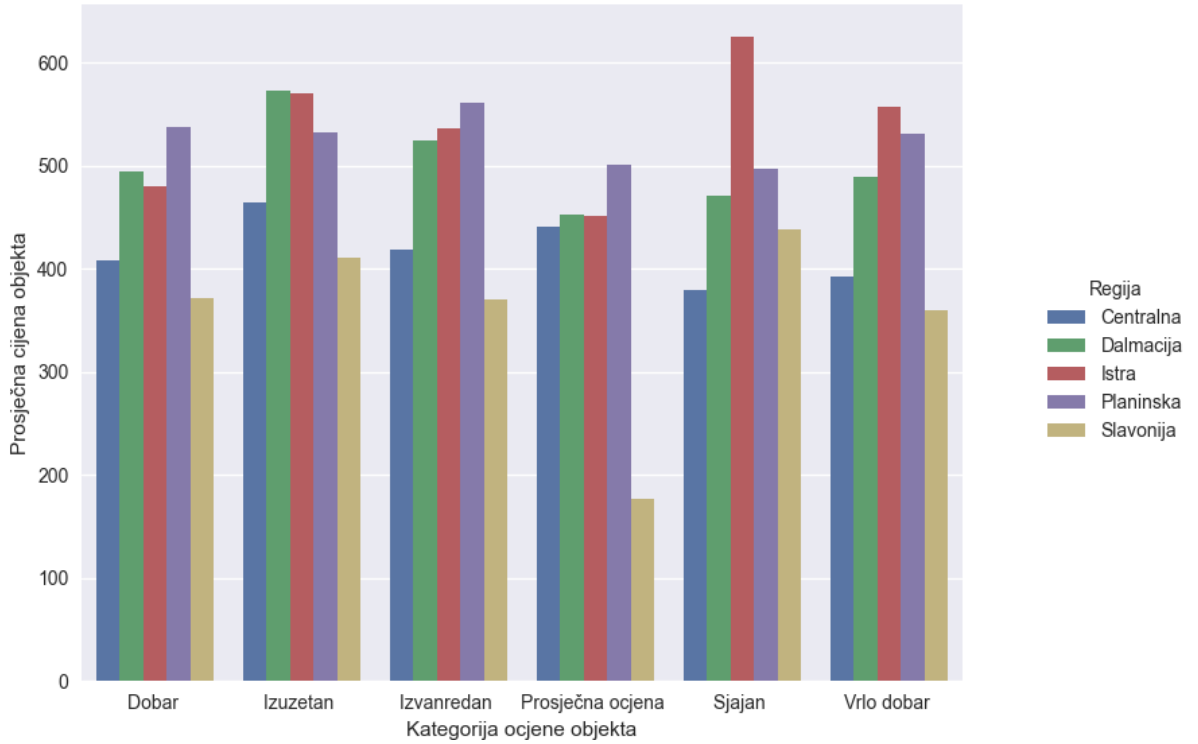
```
Vrlo dobar            36.899824
Prosječna ocjena      37.203704
Dobar                 37.402542
Sjajan                39.185438
Izvanredan            43.009560
Izuzetan              50.939644
```

Vidimo da u prosjeku klase s nižim recenzijama (prosječno ocijenjeni objekti, dobri i vrlo dobri objekti) sadrže manje objekte (~ 37 m^2), dok klase s višim recenzijama sadrže objekte malo veće kvadrature (39 – 51 m^2).

Klasa objekta – postoje li velike razlike u cijenama objekta svake klase pojedine regije?

```
df = booking.groupby(["Kategorija ocjene objekta", "Regija"])
["Cijena objekta"].mean().reset_index()
sns.catplot(x = "Kategorija ocjene objekta", y = "Cijena objekta",
kind = "bar", hue = "Regija", data = df)
plt.ylabel("Prosječna cijena objekta")
```

Skup podataka grupiramo prema klasi i regiji te računamo prosječnu cijenu objekata unutar svake kombinacije grupa (Dalmacija – sjajan objekt, Istra – prosječni objekti, Centralna – vrlo dobri objekti,...). Pozivanjem *seaborn* funkcije *catplot* možemo vizualizirati prosječnu cijenu svake klase unutar svake regije hrvatske.



Slika 37. Prosječna cijena objekata u odnosu na klase i regije

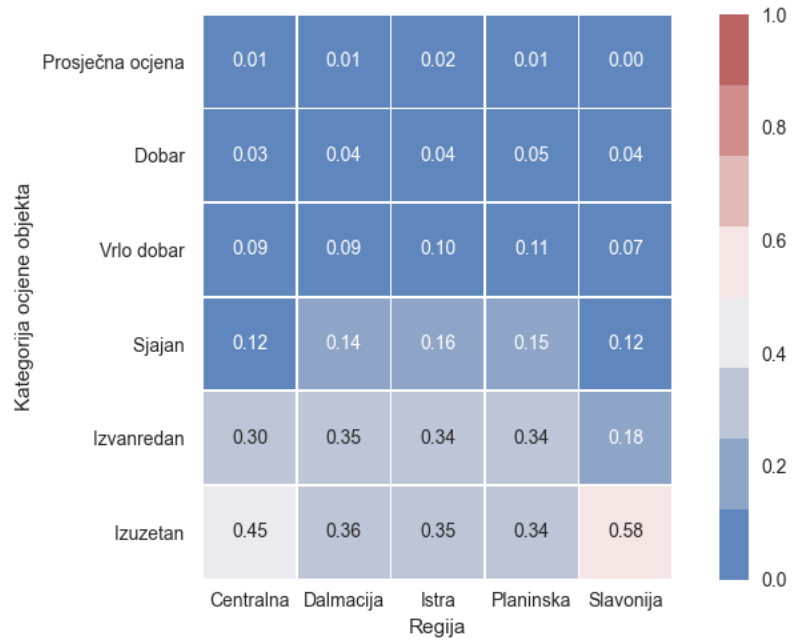
Na slici 37. nalazi se vizualizacija prosječne ocjene objekta u odnosu na klasu i regiju objekta. Promatranjem vizualizacije, možemo primijetiti da se najmanja odstupanja u prosječnoj cijeni prema klasi događa u Centralnoj hrvatskoj, gdje je prosječna cijena veoma slična kroz svaku klasu objekta. S druge strane, kod ostalih regija postoje manja i veća odstupanja tako da, pogledamo li na primjer Istru (crveni stupac) možemo vidjeti da prosječno ocijenjeni objekti u prosjeku i imaju nižu cijenu, dok sjajno ocijenjeni hoteli imaju znatno višu prosječnu cijenu u odnosu na drugačije ocijenjene objekte (u prosjeku preko 600 kuna noćenje). Isto tako, gledamo li Slavoniju, možemo primijetiti da je prosječna cijena jednog noćenja u objektu konzistentna kroz sve klase objekata (od 350 – 450 kn po noćenju) osim kod objekata koji su ocijenjeni kao prosječni kod kojih je prosječna cijena jednog noćenja ispod 200 kuna po noći.

Klasa objekta – postoje li veće razlike u udjelu klasa prema regijama?

Udio klasa prema regijama potrebno je analizirati kako bi utvrdili da unutar pojedinih regija ne postoje veća odstupanja u udjelu klase, tj. da je udio klasa sličan kod pojedinih regija kako ne bi imali situaciju da se npr. 90 % izuzetnih hotela nalazi u centralnoj hrvatskoj, a ostalih 10 % u drugim regijama i slično.

```
df = booking.groupby(["Regija", "Kategorija ocjene objekta"]).size().re
set_index(name = "Broj")
df["Postotak"] = df['Broj'] / df.groupby('Regija')['Broj'].transform('s
um')
df = df.pivot(columns='Regija', index='Kategorija ocjene objekta', valu
es="Postotak")
df.index.to_list()
df1 = df.reindex(['Prosječna ocjena ','Dobar ','Vrlo dobar ', "Sjajan "
, "Izvanredan ", "Izuzetan "])
h = sns.heatmap(df1, annot=True, cmap=colormap, linewidth=.5, square =
True, linecolor='w', fmt='.2f', vmin=0, vmax=1, center=0.5)
```

Grupiramo podatke prema regiji i klasi i računamo ukupan broj instanci unutar svake kombinacije grupa (centralna – izuzetan, centralna – sjajan,...). Izračunom postotka kao i kod prethodno izrađene matrice (za vrstu objekta) dobijemo udio svake klase unutar svake regije. Reindeksiramo skup podataka (kako bi klase bile poredane po visini recenzija) te stvaramo matricu pomoću već viđene funkcije *heatmap*.



Slika 38. Udio klasa prema regijama (matrica)

Na slici 38. vizualizirani su udjeli klasa za svaku regiju. Vidimo da svaka regija sadrži najviše instanci klase *Izuzetan* (udio kakav je i na razini cijelog skupa podataka), zatim klase *Izvanredan, Sjajan* te najmanji broj instanci klase *Dobar, Vrlo dobar i Prosječna ocjena* (kakav je udio bio i na razini cijelog skupa podataka). Iako su klase nebalansirane na razini

cijelog skupa podataka, možemo vidjeti da ne postoje veće razlike unutar regija, tj. da su postotci pojedinih klasa otprilike jednaki unutar svake regije.

Klasa objekta – postoje li veće razlike u distribuciji svake klase u odnosu na cijenu objekta?

```
fig, ((ax1, ax2, ax3), (ax4, ax5, ax6)) = plt.subplots(ncols=3,nrows=2)
```

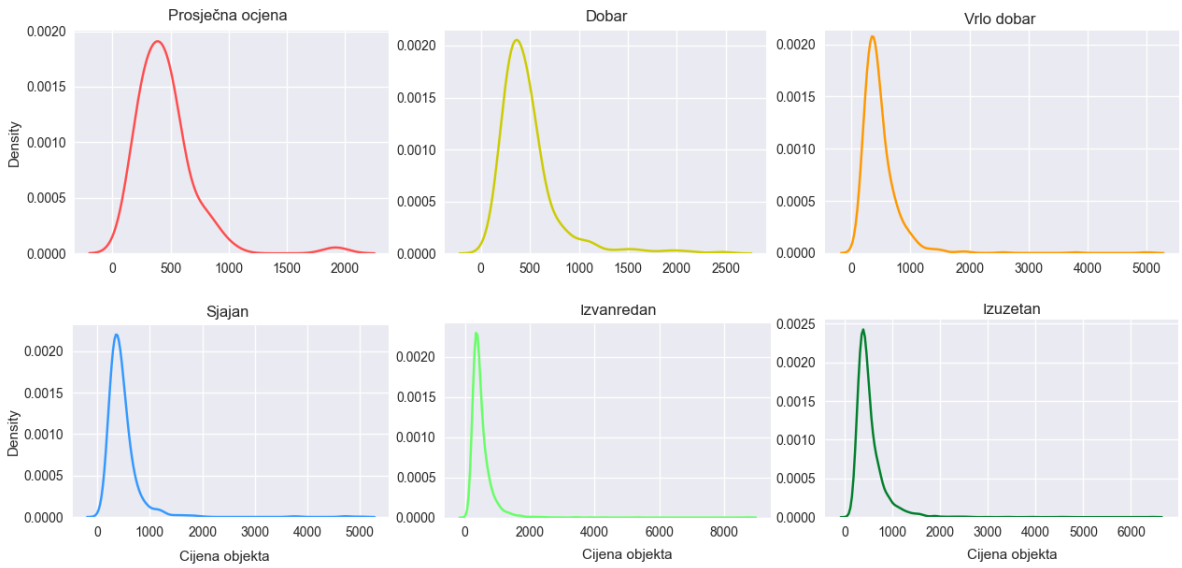
Prvo stvaramo figuru (canvas) vizualizacije koji je podjeljen na 6 djelova (po jedan za svaku kategoriju).

```
pros = booking[booking["Kategorija ocjene objekta"] == "Prosječna ocjen
a "]
dobar = booking[booking["Kategorija ocjene objekta"] == "Dobar "]...
```

Stvaramo *pandas Series* objekt za svaku klasu objekta i spremamo u odvojenu varijablu.

```
ax1 = sns.displot(pros, x="Cijena objekta", kind="kde", fill = False, c
olor = "#00802b")
plt.title("Prosječna ocjena")
```

Zatim pomoću funkcije *distplot* vizualiziramo pojedinu klasu unutar pojedinog *ax* elementa. Konačan rezultat izgleda ovako:



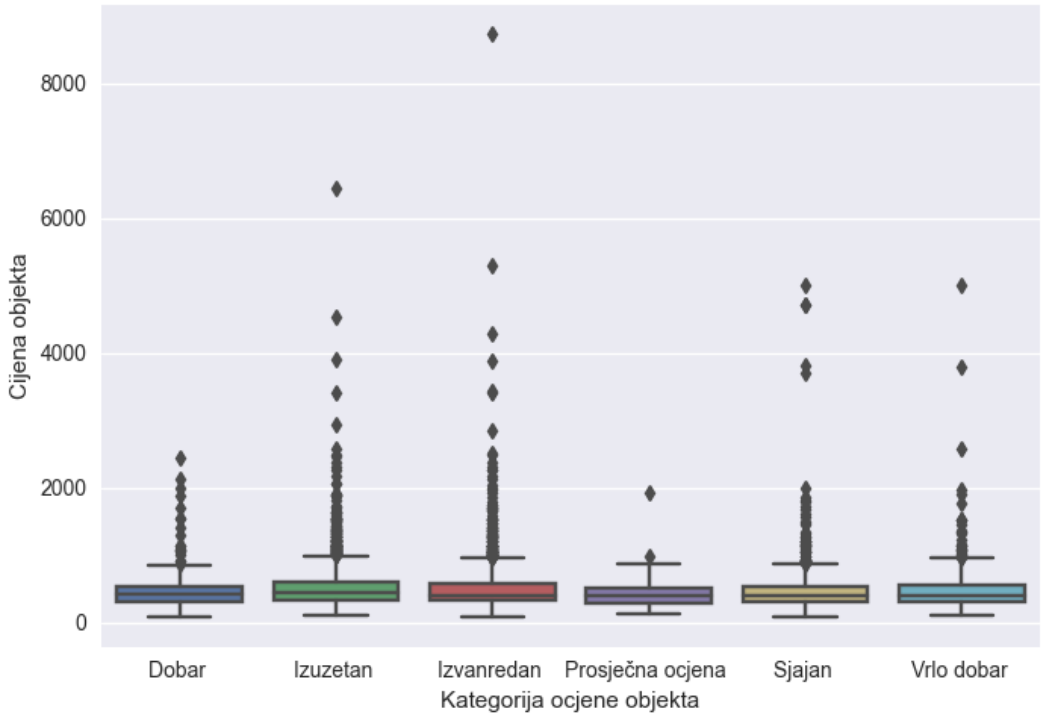
Slika 39. Distribucija klase objekta u odnosu na cijenu noćenja

Na slici 39. nalazi se prikaz distribucije pojedine klase u odnosu na cijenu objekta. Vidimo da je distribucija svake klase *right skewed* što znači da unutar svake klase postoji velik broj objekata s manjom cijenom te malen broj objekata s većom cijenom (outlieri). Vidimo da je cijena noćenja većine objekata unutar svake pojedine klase do 1000 kn po noćenju, te da outlieri (točke izvan distribucije većine) imaju cijenu po noćenju 1000 kn i veću.

Klasa objekta – istraživanje outliera

```
sns.boxplot(x="Kategorija ocjene objekta", y="Cijena objekta", data=boo
king)
```

Za istraživanje outliera kontinuirane varijable možemo koristiti *seaborn* funkciju *boxplot* gdje na os x postavljamo klase objekta, a na os y samu cijenu objekta.



Slika 40. Boxplot klase objekta u odnosu na cijenu noćenja

Na slici 40. prikazana je vizualizacija interkvartilnog raspona cijene objekata za svaku pojedinu klasu. Interkvartilni raspon jedna je od mjera korištena za mjerenje raspršenosti podataka, gdje kutija označava IQR (interkvartilni raspon) – podatkovne točne koje se nalaze između 25 % i 75 % kvartila. Linije izvan kutija (engl. whiskers) označene su podacima koje

se nalaze na dužini IQR-a pomnoženoj sa 1.5 (s gornje strane umnožak je pridodan trećem kvartilu, a s donje strane je oduzet od prvog kvartila). Podaci (točke) izvan toga smatraju se outlierima, što su instance koje ne dijele iste karakteristike kao većina točaka unutar distribucije klase. Iz priložene vizualizacije možemo vidjeti da većina klasa sadrži veći broj outliera (što se moglo primijetiti i po dužini „repa“ distribucije na prethodnoj vizualizaciji), izuzev klase objekata s prosječnom ocjenom koja sadrži najmanji broj outliera (2). Također, ova klasa sadrži i najmanji broj instanci u skupu podataka. Prema navedene dvije vizualizacije možemo zaključiti da se distribucija klasa u odnosu na cijene ne razlikuje u velikim mjerama te da klase dijele slične karakteristike podataka.

Klasa objekta – da li se klase razlikuju u odnosu na popularne sadržaje i dozvole smještaja?

Ovim pitanjem zanima nas da li postoje velike razlike u popularnim sadržajima objekata i klasi objekata, npr. izuzetni hoteli većinom dopuštaju pušenje i kućne ljubimce, prosječni hoteli nemaju besplatan parking u odnosu na druge i slična pitanja.

```
df = booking.groupby(["Kategorija ocjene objekta", "Prostor za pušače"]
).size().reset_index(name = "Broj")
df["Postotak"] = df['Broj'] / df.groupby('Kategorija ocjene objekta')['
Broj'].transform('sum')
```

Na ovaj način izračunat je postotak svih binarnih varijabli za svaku klasu, gdje gledamo što je u određenim klasama dopušteno, a što nije.

Tablica 3. Postotak dozvoljenih sadržaka prema klasi objekta

	Klasa					
Sadržaj	Prosječan	Dobar	Vrlo dobar	Sjajan	Izvanredan	Izuzetan
Pušenje	73 %	65 %	57 %	49 %	43 %	33 %
Kućni ljubimci	41 %	55 %	47 %	44 %	38 %	32 %
Zabave	68 %	69 %	62 %	59 %	49 %	38 %
Bar	19 %	25 %	23 %	21 %	12 %	7 %
Bazen	7 %	9 %	12 %	13 %	11 %	6 %
Doručak	5 %	16 %	18 %	17 %	10 %	3 %
Kućni red	82 %	81 %	69 %	66 %	59 %	57 %
Parking	88 %	84 %	85 %	84 %	86 %	90 %

Plaža	22 %	23 %	19 %	20 %	19 %	15 %
Wifi	75 %	84 %	86 %	86 %	87 %	88 %

Pušenje: Možemo primijetiti kako se smanjuje postotak objekata u kojima je pušenje dozvoljeno kako idemo prema boljim kategorijama (prosječno ocijenjeni objekti u 73 % slučajeva dopuštaju pušenje, dok se kod izuzetnih objekata dopušta samo kod 33 % objekata).

Kućni ljubimci: Ne postoji direktno smanjivanje broja objekata s dopuštenim kućnim ljubimcima ali možemo primijetiti kako je općenito navedeni postotak veći kod objekata niže recenzije (kod prosječnih, dobrih i vrlo dobrih objekata).

Zabave: Postotak objekata s dozvoljenim zabavama se, kao i kod pušenja, postepeno smanjuje kako gledamo prema boljim klasama objekata.

Bar: bar postoji u većem postotku kod objekata niže recenzije.

Bazen: bazeni u nešto većem postotku postoje kod objekata klasificiranih kao vrlo dobrih, sjajnih i izvanrednih, kod ostalih je postotak objekata koji imaju bazen manji od 10 %.

Doručak: kao kod bazena, uz navedene klase, objekti klase dobar kao i ostali sadrže nešto veći postotak objekata s dostupnim doručkom.

Kućni red: kod ove varijable gledamo koliki postotak objekata nema određen kućni red. Vidimo da se broj objekata koji nemaju definiran kućni red postepeno smanjuje kako gledamo prema boljim klasama (82 % objekata s prosječnom klasom nema definiran kućni red, dok tek 57 % objekata s izuzetnom klasom nema definiran kućni red).

Parking: Vidimo da je postotak objekata s dostupnim besplatnim parkiranjem veoma sličan kod svake klase (unutar svake klase imamo najmanje 80 % objekata s besplatnim parkingom).

Plaža: vidimo da je postotak objekata s dostupnom plažom veoma sličan kod svake klase (uz to da se unutar klase Izuzetan čak nalazi najmanje objekata s dostupnom plažom).

WiFI: iako je postotak objekata s besplatnim WiFI-em unutar svake klase visok, možemo primijetiti kako navedeni postotak raste kako gledamo prema boljoj klasi (objekti s prosječnom ocjenom u 75 % slučajeva imaju besplatan WiFI, dok kod objekata s izuzetnom ocjenom 88 % objekata ima besplatan WiFI).

Cijena – koliki je raspon vrijednosti za cijene noćenja?

```
booking["Cijena objekta"].describe()
```

Pozivanjem funkcije *describe* nad određenim stupcem skupa podataka možemo dobiti određene statističke podatke o navedenom stupcu.

Tablica 4. Statistički podaci o cijenama noćenja

Statistička mjera	**Rezultat**
Prosjek	506.94
Standardna devijacija	348.64
Minimum	114
25 %	333
50 %	419
75 %	575
Maksimum	8728

Na tablici 4. prikazani su statistički podaci za cijene noćenja smještaja. Vidimo da je ukupan prosjek cijena 507 kuna po noćenju, dok je standardna devijacija 349 kuna što znači da u prosjeku cijene za jedno noćenje odstupaju od prosjeka za 349 kuna po noćenju. Minimalna cijena jednog noćenja iznosi 114, dok je maksimalna cijena 8728 kuna po jednom noćenju. 25 % kvartil iznosi 333 kune što znači da 25 % objekata ima cijenu noćenja 333 kune ili manje dok je medijan 419 kuna po noćenju. Ovdje možemo vidjeti jednu specifičnu karakteristiku *right skewed* distribucije, a to je da je medijan (što je srednja vrijednost distribucije) manji od prosjeka, s obzirom na to da outlieri utječu na ukupan prosjek ali ne i na medijan. 75 % kvartil iznosi 575 kuna što znači da 75 % objekata ima cijenu noćenja od 575 kuna i manje dok tek ostalih 25 % objekata ima cijenu noćenja od 575 kuna do 8728 kuna.

Regija – postoje li razlike u recenziji lokacije u odnosu na regije?

```
booking.groupby("Regija").agg(Prosjek_lokacija = ("Recenzija lokacije",
"mean"), Prosjek_sadrzaj = ("Recenzija sadržaja", "mean"))
```

Grupiramo skup podataka prema regiji i agregiramo na način da tražimo prosječnu ocjenu recenzije lokacije i recenzije sadržaja.

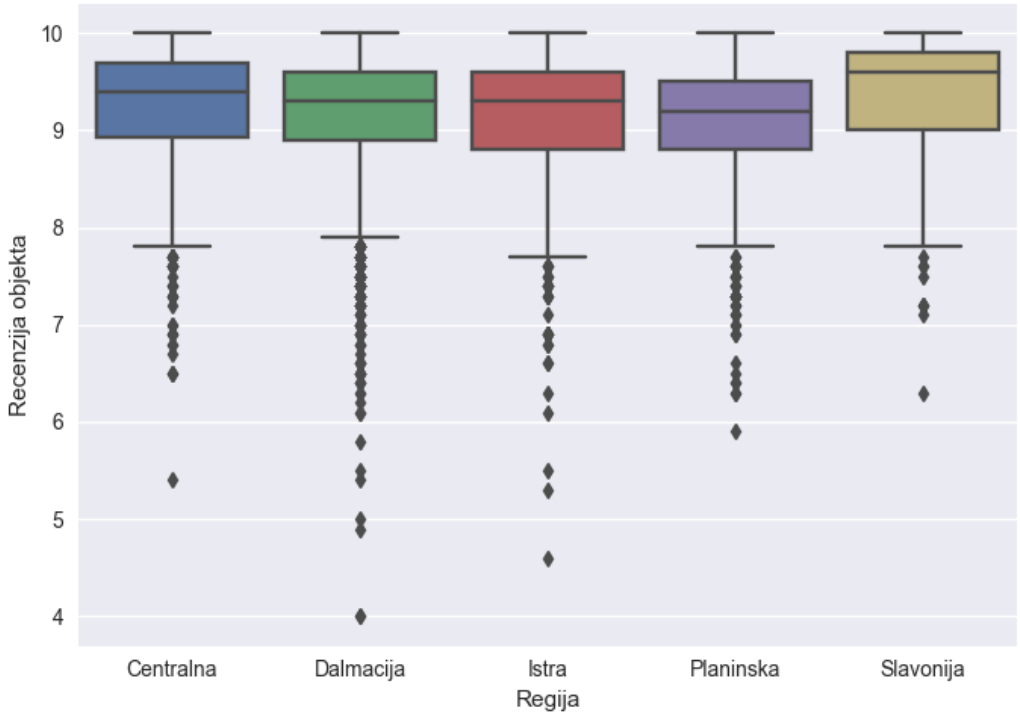
```
  Regija        Prosjek_lokacija Prosjek_sadrzaj

Centralna           9.157022         9.208714
Dalmacija           9.093644         9.061826
Istra               9.002236         9.057391
Planinska           9.005428         9.059203
Slavonija           9.190805         9.333716
```

Vidimo da se prosječne recenzije lokacije u odnosu na regije ne razlikuju u velikim mjerama (razlika između prosječnih recenzija je ~ 0.1 ili manje) dok je kod prosjeka recenzija sadržaja prosječna recenzija nešto viša kod Centralne Hrvatske i kod Slavonije.

Regija – koliko su raspršene recenzije u odnosu na regije?

```
sns.boxplot(x="Regija", y="Recenzija objekta", data=booking)
```

Izrađen je boxplot sa regijom hrvatske na x osi i recenzijom objekta na y osi.



Slika 41. Boxplot za regije naspram recenzija

Vidimo da u svakoj regiji postoji određen broj outliera, tj. recenzija koje ne pripadaju distribuciji većine. Vidimo da su medijani distribucija veoma slični kroz regije (uz malo višu srednju recenziju u Slavoniji). Koliko točno outliera ima, možemo vidjeti na sljedeći način:

```
q1 = booking['Recenzija objekta'].quantile(0.25)
q3 = booking['Recenzija objekta'].quantile(0.75)
iqr = q3 - q1
booking["Outlier"] = ((booking["Recenzija objekta"] < (q1 - 1.5 * iqr))
|(booking["Recenzija objekta"] > (q3 + 1.5 * iqr)))
```

U posebne varijable spremljeni su 25 % kvartil, kao i 75 % (koji čine IQR). Zatim stvaramo novi stupac *Outlier* tipa *boolean* koji se označava s *True* ako je recenzija objekta manja od razlike prvog kvartila i umnoška IQR-a sa 1.5 ili ako je recenzija objekta veća od zbroja trećeg kvartila i umnoška IQR-a i 1.5.

```
booking.groupby("Regija")["Outlier"].value_counts()
```

Na ovaj način možemo vidjeti koliko točno je vrijednosti unutar svake regije označeno s *True* ili *False*.

Tablica 5. Tablica outliera za regije

Regija	***True* outlier**
Centralna	37
Dalmacija	113
Istra	35
Planinska	46
Slavonija	10

Vidimo da, kako je i potvrđeno vizualizacijom, Slavonija ima najmanje outliera, zatim Istra, Centralna hrvatska i Planinska, te najviše outliera imamo u Dalmaciji.

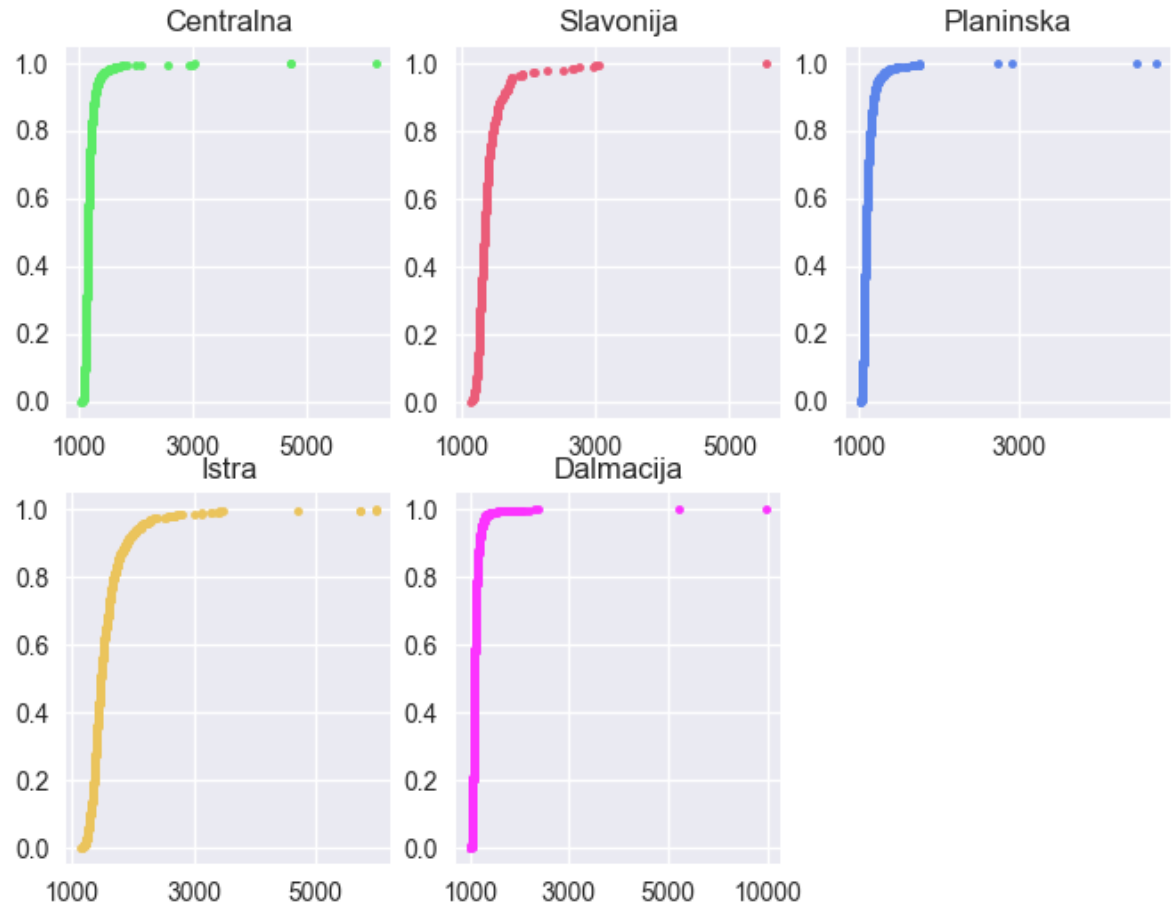
Regija – kako izgleda distribucija cijena prema regijama?

```
fig, ((ax1, ax2, ax3), (ax4, ax5, ax6)) = plt.subplots(ncols=3,nrows=2)
centr = booking[booking["Regija"] == "Centralna"]
slav = booking[booking["Regija"] == "Slavonija"]...
```

Stvaramo 6 *ax* objekata (po jednu za svaku regiju) unutar. Svaku regiju filtrirano iz originalnog skupa podataka i spremamo u odvojenu varijablu.

```
x = np.sort(centr['Cijena objekta'])
y = np.arange(1, len(x)+1) / len(x)
ax1.plot(x, y, marker='.', linestyle='none', color = "#5ceb67")
ax1.set_title("Centralna")
ax1.set_xticklabels(['500','1000','3000','5000','10000'])...
```

Ovaj dio koda predstavlja reprezentaciju ECDF funkcije u Pythonu. ECDF (Funkcija empirijske kumulativne distribucije) prikazuje udio rezultata koji je za svaki rezultat manji ili jednak. Na x osi nalaze se sortirane vrijednosti varijable zanimanja (u ovom slučaju cijene objekta određene regije) dok se na y osi nalaze vrijednosti od 1 do dužine od x polja uvećanog za 1 koji je zatim podijeljen s dužinom od x kako bi dobili postotak. Ovaj dio, kad i dio grafičkog prikaza x i y osi je izveden za svaku regiju posebno.



Slika 42. ECDF po regijama

Vizualizaciju čitamo tako da odaberemo postotak koji nas zanima i iz iscrtane točke povlačimo vertikalnu liniju na x os kako bi dobili cijenu objekta. Npr. ako gledamo Dalmaciju, uzmemo li postotak 0.6 tj. 60 % dobit ćemo otprilike 1000, što znači da 60 %

objekata unutar Dalmacije ima cijenu noćenja 1000 kuna ili manje. Kako su ECDF funkcije većinom izrazito uspravne, možemo zaključiti da su i po regijama, distribucije također *right skewed* jer vidimo da velik postotak objekata unutar regije ima cijenu noćenja 1000 kuna ili manju dok tek manji dio objekata ima cijenu višu od 1000 kuna po noćenju.

4.1. Testiranje statističke razlike između regija – ANOVA

ANOVA (engl. *One-Way Analysis of Variance*) koristi se za otkrivanje statističke razlike između aritmetičke sredine (prosjeka) tri ili više neovisne grupe podataka. ANOVA se provodi tako da se analiziraju razine varijacije unutar grupa kroz određeni broj uzoraka preuzet iz navedenih grupa. Pojednostavljeno, provjeravamo da li je aritmetička sredina numeričke varijable (u našem slučaju recenzije objekta) jednaka u svim grupama (u ovom slučaju regijama). ANOVA radi tako da mjeri dva izvora varijacije u podacima i uspoređuje njihove relativne veličine. Pod dva izvora varijacije, misli se na varijancu između grupa (razlika između aritmetičke sredine pojedine grupe i ukupne aritmetičke sredine svih grupa) i varijancu unutar grupa (razlika između pojedine vrijednosti unutar grupe i aritmetičke sredine navedene grupe) [4].

H_0 (*null* hipoteza): razlike vrijednosti aritmetičkih sredina grupa regija nisu statistički značajne.

H_a: razlike vrijednosti aritmetičkih sredina grupa regija statistički su značajne, postoji povezanost.

Postoji nekoliko vrsta ANOVA testova, no u ovom slučaju je odabrana *One-Way* ANOVA s obzirom na to da imamo jednu neovisnu varijablu (regije).

```
from scipy.stats import f_oneway
from statsmodels.stats.multicomp import pairwise_tukeyhsd

F, p = f_oneway(centr["Recenzija objekta"], slav["Recenzija objekta"],
plan["Recenzija objekta"], istr["Recenzija objekta"], dalm["Recenzija o
bjekta"])
```

Za izračun *F* i *p* vrijednosti korištena je *SciPy* funkcija *f_oneway* gdje navodimo od svakog filtriranog skupa podataka (prema regijama) stupac koji sadrži recenzije. Rezultat koji dobijemo je sljedeći:

```
F
Out[19]: 14.06561694759401

p<0.05
Out[20]: True
```

F – statistika predstavlja omjer varijacije između grupa i varijacije unutar grupa te visoka F vrijednost (niskom F vrijednosti se smatra F vrijednost oko 1) dokaz je protiv *null* hipoteze jer indicira da je veća razlika između grupa nego unutar pojedinih grupa. U našem slučaju F vrijednost je ~ 14 te provjeravamo i p vrijednost kako bi se uvjerili da možemo odbaciti *null* hipotezu.

P vrijednost (vrijednost vjerojatnosti) primarna je vrijednost koja se koristi za kvantificiranje statističke značajnosti rezultata testa hipoteze [5] te ukoliko je ista niska, smatra se da postoji dovoljno dokaza za odbacivanje *null* hipoteze. U ovom slučaju provjeravamo da li je p manji od 0.05 te vidimo da jest, točnije, p vrijednost iznosi

```
p
Out[27]: 1.9908132646358898e-11
```

što, uz visoku F vrijednost, dokazuje da postoji dovoljno dokaza za odbacivanje *null* hipoteze, tj. da su razlike između aritmetičkih sredina grupa regija statistički značajne.

Nadalje, jedna od mana ANOVA testa jest da ne prikazuje između kojih grupa postoje značajne statističke razlike, te u tu svrhu koristimo funkciju *pairwise_tukeyhsd* iz biblioteke *statsmodel.*

Tukey HSD (engl. *Tukey's Honest Significant Difference test*) je post-hoc test koji nam govori između kojih grupa je razlika statistički značajna. Može se provesti nakon testa kao što je ANOVA kada smo utvrdili da postoji značajna statistička razlika između aritmetičkih sredina grupa.

```
tukey = pairwise_tukeyhsd(endog=booking['Recenzija objekta'], groups=bo
oking['Regija'])
print(tukey)
```

U navedenu funkciju navodimo varijablu koja se analizira (u ovom slučaju recenzija objekta) te grupe koje se analiziraju (u ovom slučaju regije).

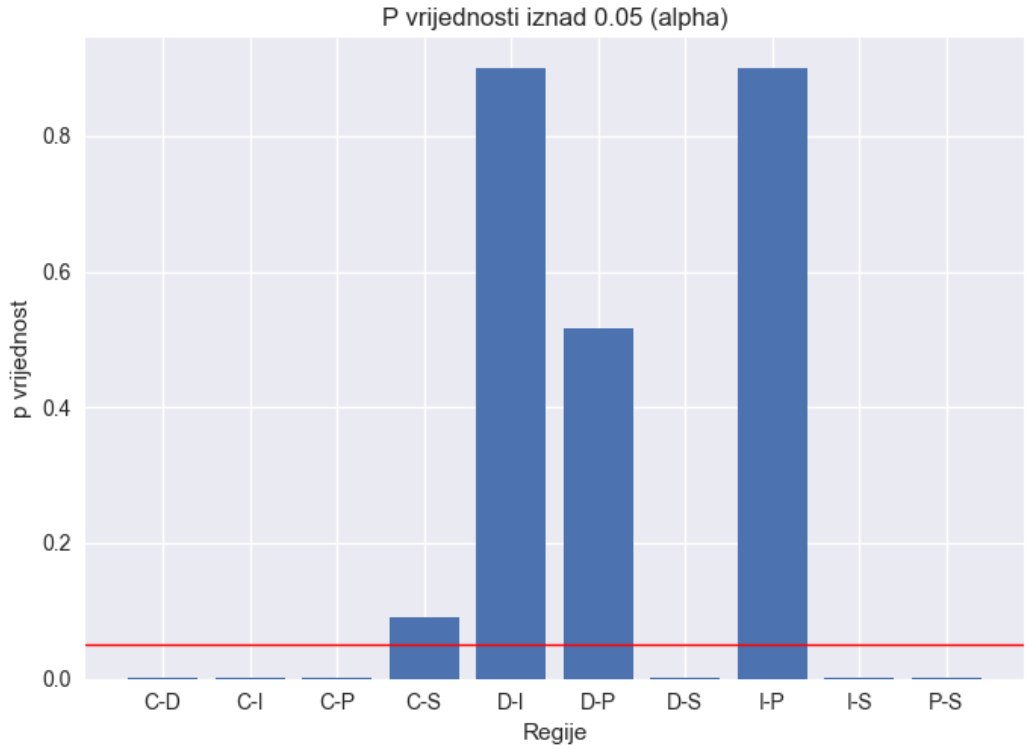
```
In [32]: print(tukey)
   Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
  group1    group2  meandiff p-adj   lower   upper  reject
-----------------------------------------------------
Centralna Dalmacija  -0.0975  0.001 -0.1554 -0.0396   True
Centralna     Istra  -0.1076 0.0011 -0.1835 -0.0316   True
Centralna Planinska  -0.1308  0.001 -0.1992 -0.0624   True
Centralna Slavonija   0.1041 0.0909 -0.0096  0.2177  False
Dalmacija     Istra  -0.0101    0.9 -0.0768  0.0566  False
Dalmacija Planinska  -0.0333 0.5168 -0.0912  0.0247  False
Dalmacija Slavonija   0.2016  0.001  0.0939  0.3093   True
    Istra Planinska  -0.0232    0.9 -0.0992  0.0528  False
    Istra Slavonija   0.2116  0.001  0.0933    0.33   True
Planinska Slavonija   0.2348  0.001  0.1212  0.3485   True
-----------------------------------------------------
```

Slika 43. Tukey HSD test

Na slici 43. prikazani su rezultati Tukey HSD testa gdje možemo vidjeti grupe koje se međusobno uspoređuju, razliku između aritmetičkih sredina navedenih grupa, p vrijednost te da li navedena razlika odbacuje ili prihvaća *null* hipotezu. Vidimo da značajna statistička razlika između aritmetičkih sredina postoji kod sljedećih grupa: između Centralne hrvatske i Dalmacije (p vrijednost 0.001), Centralne hrvatske i Istre (p vrijednost 0.0011), Dalmacije i Slavonije (p vrijednost 0.001), Istre i Slavonije (0.001) te Planinske hrvatske i Slavonije (0.001). Kada gledamo Centralnu hrvatsku, možemo primijetiti da je statistički različita od svih regija osim Slavonije (koja joj je ujedno i susjedna regija dotične), isto kao što su Dalmacija i Slavonija geografski poprilično udaljene. Isto tako, Istra i Slavonija kao i Planinska hrvatska i Slavonija geografski su udaljene jedne od druge i također statistički različite jedna od druge, dok npr. vidimo da Istra i Planinska hrvatska ili Dalmacija i Planinska hrvatska nisu značajno različite jedna od druge.

Navedene rezultate možemo i vizualizirati.

```
df = pd.DataFrame(data=m_comp._results_table.data[1:], columns=m_comp._
results_table.data[0])

plt.bar(df["Regije"], df["p-adj"])
plt.axhline(y=0.05,linewidth=1, color='r')
plt.xlabel("Regije")
plt.ylabel("p vrijednost")
plt.title("P vrijednosti iznad 0.05 (alpha)")
```

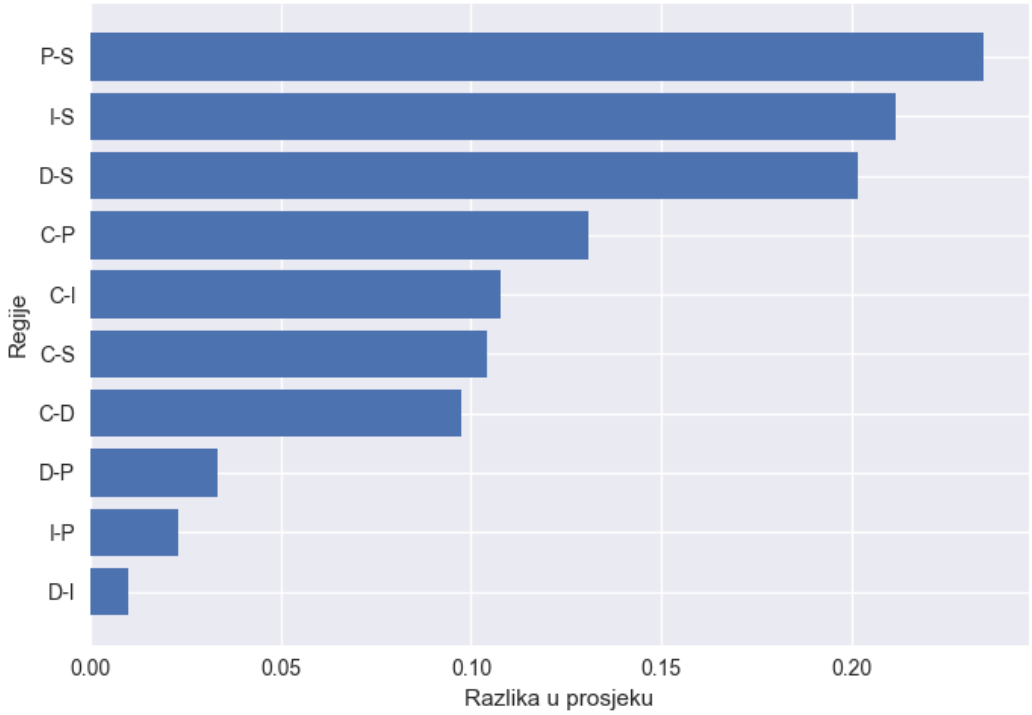


Slika 44. P vrijednosti iznad alphe

Iz navedene vizualizacije možemo vidjeti da Centralna hrvatska – Slavonija, Dalmacija – Istra, Dalmacija – Planinska hrvatska i Istra – Planinska hrvatska nisu statistički značajne, s obzirom na to da crvena linija na vizualizaciji predstavlja alpha vrijednost (odlučujuću vrijednost statističke značajnosti). Nazivi regija su skraćeni radi preglednije vizualizacije.

```
df["meandiff"] = df["meandiff"].abs()
df.sort_values('meandiff',inplace=True)
plt.barh(df["Regije"], df["meandiff"])
plt.xlabel("Razlika u prosjeku")
plt.ylabel("Regije")
```

Kod vizualizacije razlike aritmetičke sredine važno je samo napomenuti da gledamo apsolutnu vrijednost s obzirom na to da imamo negativne i pozitivne vrijednost, a nas zanima samo kolika je točno razlika.



Slika 45. Razlika u prosjeku između grupa

Možemo vidjeti da je najveća razlika u aritmetičkim sredinama grupa između Planinske hrvatske i Slavonije (0.23), zatim Istre i Slavonije (0.21) te Dalmacije i Slavonije (0.2), što su sve regije naspram Slavonije osim Centralne hrvatske te možemo zaključiti da se regija Slavonije (izuzev Centralne hrvatske) najviše razlikuje od drugih regija Hrvatske. Prema grafikonu, odmah sljedeća je Centralna hrvatska, prema čemu vidimo da se Slavonija i Centralna hrvatska (osim svog međusobnog odnosa) najviše razlikuju od ostalih regija.

4.2. Korelacija između varijabli

Korelacija je bivarijatna analiza koja mjeri jakost veze između dvije varijable. Ovisno o jačini veze, vrijednost korelacijskog koeficijenta nalazi se u rasponu od -1 do 1 gdje 1 predstavlja strogo pozitivnu korelaciju (rastom jedne varijable raste i druga), a -1 predstavlja strogo negativnu korelaciju (smanjivanje jedne varijable uzrokuje povećanje druge varijable). Kako se korelacijski koeficijent približava nuli (u slučaju negativne i pozitivne korelacije), veza između navedene dvije varijable je slabija [6].

```
booking["Vrsta objekta"] = booking["Vrsta objekta"].cat.codes
booking["Lokacija (grad)"] = booking["Lokacija (grad)"].astype('categor
y').cat.codes
booking["Kategorija ocjene objekta"].cat.categories = [1,5,4,0,3,2]
```

```
booking["Kategorija ocjene objekta"] = booking["Kategorija ocjene objek
ta"].astype('int')
booking["Zupanija"] = booking["Zupanija"].cat.codes
booking["Regija"] = booking["Regija"].cat.codes
```

Kako bi mogli kategoričke varijable iskoristiti u računanju korelacije, potrebno je iste pretvoriti u varijable numeričkog tipa. Kod svih varijabli kategoričkog tipa (vrsta objekta, grad, županija i regija) osim klase objekta možemo samo iskoristiti funkciju cat.codes i vrijednostima se pridružuju odgovarajući brojevi, no kod klase, s obzirom na to da imaju određeni redoslijed (prosječna ocjena je niža od dobre koja je niža od vrlo dobre, itd) potrebno je navesti redoslijed varijabli kako bi se ispravno iskodiralo. Kod 0 u ovom slučaju predstavlja prosječnu klasu, dok 5 predstavlja klasu izuzetan. Brojevi se u obliku liste navode (1, 5, 4, 0, 3, 2) prema abecednom redu klasa jer Python kategoričke varijable sortira prema abecednom redu (gleda prva slova riječi) te npr. klasa dobar je abecednim redom prva klasa te je istoj pridružen kod 1, zatim izuzetan i izvanredan kojima su redom pridružene klase 5 i 4, zatim prosječna ocjena s kodom 0, te sjajan i vrlo dobar s ocjenama 3 i 2. Nakon definiranja redoslijeda kategoričkih varijabli potrebno ih je naknadno pretvoriti u brojeve.

Vrsta objekta	Lokacija (grad)	Kategorizacija objekta	Recenzija objekta	Kategorija ocjene objekta	Broj recenzija	ič
0	25	3	9.7	5	26	72
13	25	3	9.5	5	21	35
8	54	4	9.3	4	26	51
11	25	4	9.3	4	77	25
0	54	3	8.9	3	35	28
2	54	3	9.2	4	64	29
0	54	3	9.7	5	33	40
8	54	4	9	4	71	18
8	54	3	9.9	5	214	19
11	25	4	8.7	3	30	30

Slika 46. Izgled kodiranog skupa podataka

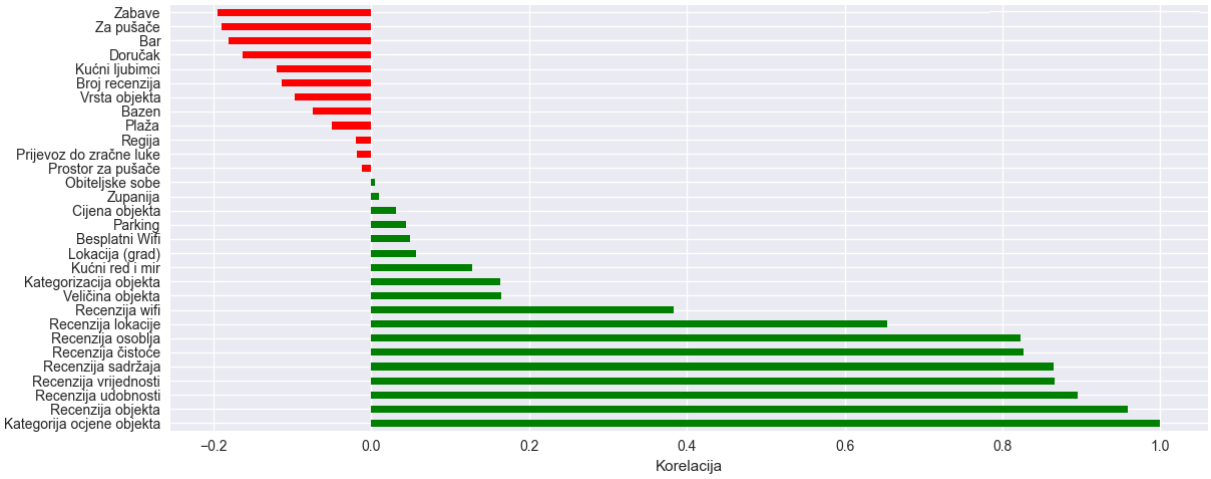
Na slici 46. prikazan je izgled skupa podataka nakon konverzije svih kategoričkih varijabli u brojeve.

```
df = booking_kodiran.corr()['Kategorija ocjene objekta'][:]
.reset_index()
df.sort_values('Kategorija ocjene objekta',inplace=True,
ascending = False)
```

Najprije stvaramo novi *Data Frame* koji sadrži korelacije svih varijabli u odnosu na kategoriju ocjene objekta (klasu).

```
df['Pozitivne'] = df['Kategorija ocjene objekta'] > 0
df.plot(x = "index", y = "Kategorija ocjene objekta", kind='barh',color
=df.positive.map({True: 'g', False: 'r'}))
plt.barh(df["index"], df["Kategorija ocjene objekta"])
plt.xlabel("Korelacija")
```

Stvaramo novi stupac *Pozitivne* koji sadrži *boolean* vrijednosti *True* ili *False* ovisno o tome da li je vrijednost pozitivna ili negativna. U funkciji *plot* naveden je pod parametar *color* mapa rječnika koji pridružuje boju varijabli ovisno o tome da li je red pozitivan ili negativan.



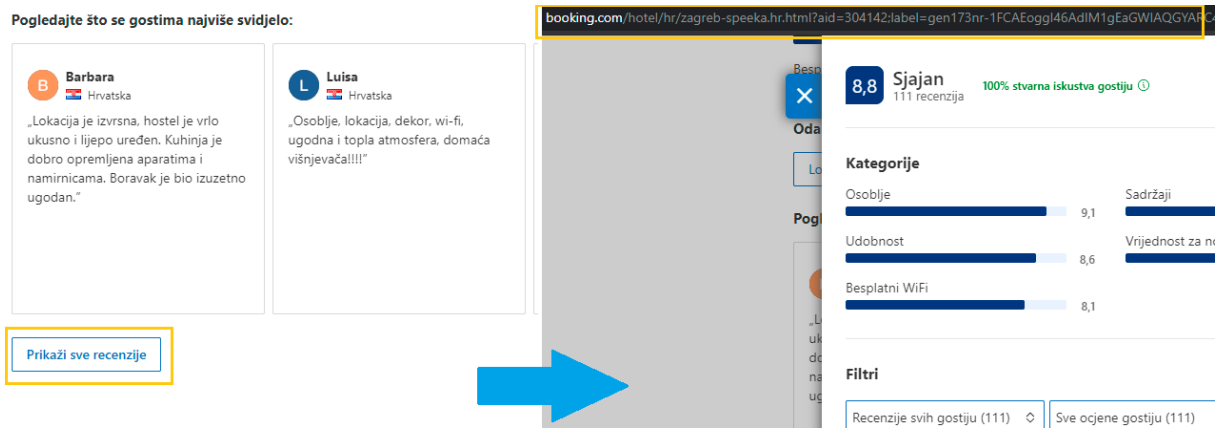
Slika 47. Korelacija varijabli

Vidimo da najveću korelaciju s klasom ima i sama recenzija što ima smisla s obzirom da se klasa pridružuje s obzirom na recenziju (prije izrade modela ova varijabla bit će izbačena iz skupa podataka). Nadalje, vidimo da su varijable s korelacijom većom od 0,6 redom sve recenzije vezane uz objekt (recenzija udobnosti, vrijednosti, WiFI, itd) gdje neka više neka manje utječe na samu klasu objekta. Dobivši ove rezultate, testirano je da li je sama recenzija rezultat prosjeka drugih recenzija i rezultati se ne preklapaju što znači da su recenzije pridružene objektu same za sebe. Kako navedene recenzije imaju najveću korelaciju sa samom klasom objekta, prikupit će se i analizirati sami tekstovi recenzija kako bi izvukli iz samih tekstova recenzija što to najviše utječe na ocjenu korisnika.

5. Analiza recenzija smještaja

Kako bi prikupili tekstove recenzija, potrebno je provesti tri koraka:

1) **Prikupljanje poveznice s prikazom recenzija:** trebamo pronaći poveznicu s popisom recenzija za svaki objekt. Program za struganje webom treba kliknuti na gumb na lijevoj strani slike (slika 48.) i dohvatiti poveznicu označenu na desnoj strani slike 48.



Slika 48. Poveznica sa recenzijama

```
<div class="hp-social_proof reviews-snippet-sidebar hp-social-proof-
review score">...
<button class="bui-button bui-button--secondary" type="button"...
<span class="bui-button__text">
Prikaži sve recenzije
</span>
</button>
</div>
```

Element *„button"* klase *„bui-button--secondary"* unutar *„div"* elementa klase *„hp-social-proof-review_score"* sadrži poveznicu na stranicu s recenzijama. Kako na cijeloj stranici postoji više gumbova navedene klase, potrebno je prvo pretražiti navedeni *„div"* element kako bi pristupili navedenom gumbu.

```
url_recenzije = []


def dodavanje_stranica_rec(url):
    print(url)
    driver.get(url)
try:
    element = driver.find_element_by_class_name("hp-social-\
    proof-review_score")
```

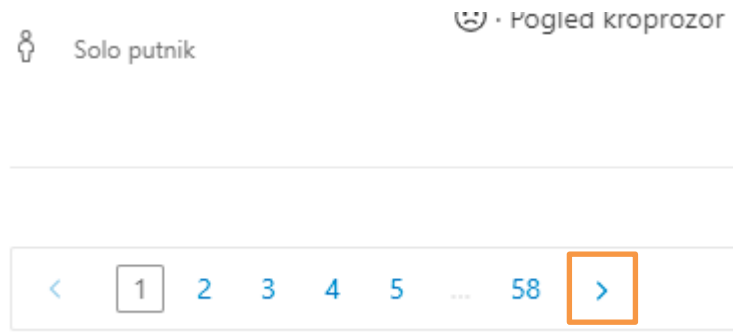
```
    element1 = element.find_element_by_class_name("bui-button\
    --secondary")
    time.sleep(2)
    element1.click()
    stranica = driver.current_url
    print(stranica)
    url_recenzije.append(stranica)
except:
    pass
```

Driveru se prosljeđuje url te, s obzirom na to da nemaju svi objekti recenzije, unutar *try – except* bloka driver pokušava pronaći potreban gumb te pomoću funkcije *click* pristupa istom. Pomoću metode *current_url* dohvaćamo poveznicu otvorene kartice (prozor) i spremamo u unaprijed definiranu listu. Ako ne postoji gumb unutar navedenog *div* elementa, navedeni objekt se preskače naredbom *pass*.

```
for i in range(0, len(lista_linkova)):
    print(i)
    dodavanje_stranica_rec(lista_linkova[i])
```

Nakon definiranja funkcije, iteriramo kroz svaku poveznicu iz postojeće liste poveznica.

2) **Paginacija poveznice s recenzijama:** svaka poveznica s recenzijama ima paginaciju s brojem stranica ovisno o broju recenzija.



Slika 49. Paginacija kroz prozor s recenzijama

Po jednoj stranici zapisano je 10 recenzija, te zasada u listi *url_recenzije* imamo samo poveznice na prvu stranicu pretraživanja recenzija.

```
<div class="bui-pagination__item bui-pagination__next-arrow">
      <a class="pagenext"
```

```
href="/reviewlist.hr.html?aid=304142;label=gen...offset=10;rows=10
" aria-label="Sljedeća stranica">
```

Element „*a*“ klase „*pagenext*“ koji se nalazi unutar „*div*“ elementa klase „*bui-pagination__next-arrow*“ sadrži poveznicu na sljedeću stranicu.

```
stranicerec_next =[]
def stranice_next(url):
    while True:
        print(url)
        stranicerec_next.append(url)
        driver.get(url)
        time.sleep(5)
        try:
            next_page = driver.find_element_by_class_name
            ('bui-pagination__next-arrow')
            .find_element_by_class_name("pagenext").get_property('href)
            url = next_page
            stranicerec_next.append(url)
        except:
            break
```

Sve dok postoji gumb za sljedeću stranicu, tražimo elemente unutar navedenih klasa te sljedeći url koji se pridružuje funkciji postaje poveznica koja je prikupljena prošlom iteracijom. Naredbom *break* naređujemo programu da zaustavi pretraživanje trenutne poveznice ako na istoj više ne postoji gumb za sljedeću stranicu.

Kako navedena lista ima prikupljeno preko 100 000 poveznica odlučeno je, zbog balansiranja klasa, izvući nasumičan broj indeksa svake klase jednak broju najmanje učestale klase, što je u ovom slučaju klasa *Prosječna ocjena* sa udiom objekata od 68.

```
booking["Kategorija ocjene objekta"].value_counts()
Out[39]:
Izuzetan              2563
Izvanredan            2236
Sjajan                 959
Vrlo dobar             466
Dobar                  272
Prosječna ocjena        68
```

```
prosjecna = np.where(booking["Kategorija ocjene objekta"] == "Prosječna
ocjena ")
```

Pomoću *numpy* funkcije *where* dohvaćamo indekse skupa podataka *booking* gdje je klasa jednaka prosječnoj ocjeni.

prosjecna - Tuple (1 element)

Inde	Type	Size	Value
0	Array of int64	(68,)	[5071 5072 5073 ... 5136 5137 5138]

Slika 50. Indeksi klase

Provođenjem metode dobijemo prikaz sa slike 50.

```
prosjecna = list(prosjecna[0])
```

Kako bi dobili listu elemenata, pristupamo prvom elementu navedene *n-torke.*

prosjecna - List (68 elements)

Inde	Type	Size	
0	int64	1	5071
1	int64	1	5072
2	int64	1	5073
3	int64	1	5074
4	int64	1	5075
5	int64	1	5076

Slika 51. Lista indeksa određene klase

Isti proces provodimo za sve klase objekata:

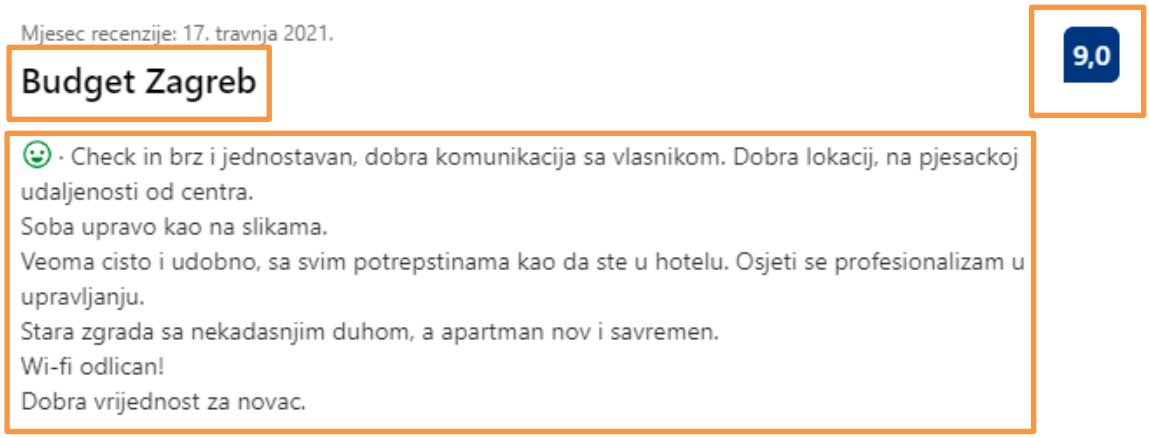
```
vrlodobar = np.where(booking["Kategorija ocjene objekta"] == "Vrlo doba
r ")
vrlodobar = list(vrlodobar[0])
vrlodobar = random.sample(vrlodobar,68)
```

no, s obzirom na to da druge klase imaju više od 68 objekata unutar grupe, potrebno je izvući nasumično 68 indeksa navedene liste kako bi za svaku klasu imali listu s jednakim brojem vrijednosti.

```
brojevi = prosjecna + dobar + vrlodobar + sjajan + izvanredan + izuzeta
n

for i in brojevi:
    print(i)
    stranice_next(url_recenzije[i])
```

Sada kada imamo posebne liste sa 68 elemenata za svaku klasu, spajamo ih u zajedničku listu brojevi kojom zatim iteriramo i prikupljamo poveznice na sljedeće stranice.

3) **Prikupljanje brojčane recenzije korisnika, naslova recenzije i samog teksta recenzije:** kada smo prikupili sve poveznice, potrebno je prikupiti sadržaj samih recenzija s navedenih stranica.



Slika 52. Sadržaj recenzije za prikupiti

Na slici 52. prikazan je sadržaj koji želimo prikupiti.

```
<li class="review_list_new_item_block" data-review-
url="4663f130f8aec7ac">...
      <h3 lang="hr" class=" c-review-block__title c-review__title--
      ltr  "> Izuzetan </h3>...
      <div class="bui-review-score c-score"> <div class="bui-review-
      score__badge" aria-label="Ocijenjeno s 10 "> 10 </div>...
      </span><span class="c-
      review__body" lang="hr">Nisam doruckovala, a krevet je ok</span>
      ...
</li>
```

Naslov se nalazi unutar *h3* elementa klase „*c-review-block__title*“. Cijena se nalazi unutar *div* elementa klase „*bui-review-score__badge*“. Tekst se nalazi unutar *span* elementa klase „*c-review__body*“.

```
ocjenarec = [], naslov = [], tekst = []
def skupljanje_recenzija(url):
    driver.get(url)
    html = driver.page_source
    soup = BeautifulSoup(html,'lxml')
    li = soup.find_all("li", {"class":"review_list_new_item_block"})
    time.sleep(2)
    for i in range(len(li)):
        item = li[i]
        try:
            ocjena = item.find("div", class_=
            "bui-review-score__badge").text
        except:
            ocjena = "N/A"
        ocjenarec.append(ocjena)
        try:
            nas = item.find("h3", class_="c-review-block__title").text
        except:
            nas = "N/A"
        naslov.append(nas)
        try:
            text = item.find("span", class_="c-review__body").text
        except:
            text = "N/A"
        tekst.append(text)
```

Driveru se prosljeđuje poveznica čiji je izvor stranice spremljen u *html* varijablu. Svaka recenzija nalazi se unutar „*li*“ elementa klase „*review_list_new_item_block*“ kojih ima 10 po stranici. Iteriramo kroz svaki *li* element te prikupljamo potrebne elemente – ocjenu, naslov i tekst recenzija pod već spomenutim klasama.

```
for i in range(0, len(stranicerec_next)):
    print(i)
    skupljanje_recenzija(stranicerec_next[i])
```

Iteriramo kroz poveznice unutar prethodno prikupljenih poveznica s recenzijama. Prilikom završetka iteracije, moguće je izraditi *Data Frame* iz navedene tri liste.

```
rec = pd.DataFrame({'Ocjena': ocjenarec,'Naslov': naslov,'Tekst': tekst
})
rec = rec[~rec.Ocjena.str.contains("N/A")]
```

Također, zbog izostanka ocjene iz recenzije, 15 recenzija je izbačeno iz skupa podataka.

Ocjena	Naslov	Tekst
10	Izuzetan	Izuzetno udoban krevet. Sobe su nove i mode…
10	Izuzetan	Osoblje izuzetno ljubazno i susretljivo, kr…
10	Izuzetan	Sve je bilo super.Čisto,udobno,mirno i jeftino,sve pohvale.Osoblje jako ljubazno.
10	Posjetila sam Bernardu vec 2 puta, vracam s…	Apsolutno sve je bilo savrseno, smjestaj, osoblje, ljubaznost, srdacnost, hrana...
9,0	Izvanredan	Preporučam!
10	Izuzetan	Čisto, uredno i mirno! Osoblje jako ljubazno!
10	Kratak odmor u malome i mirnome mjestu, odl…	Ljubazno i profesionalno osoblje, uvijek na…
9,0	Tople preporuke!	Odlična lokacija, smještaj ugodan!
10	Izuzetan	Super lokacija, čiste sobe!
9,0	odličan, ponovila bih!	sadržaji u sobi, nedostupnost nekih kanala, frižider

Slika 53. Izgled skupa podataka nakon stvaranja Data Frame-a

Na slici 53. prikazano je kako izgleda novo stvoren skup podataka.

5.1. Čišćenje recenzija

```
rec["Ocjena"] = rec["Ocjena"].str[1:]
rec["Ocjena"] = [str(x).replace(',', '.') for x in rec["Ocjena"]]
rec["Ocjena"] = pd.to_numeric(rec["Ocjena"], downcast='float')
```

Izvlačimo substring od prvog znaka pa na dalje (postoji praznina ispred broja), decimalni zarez zamjenjujemo decimalnom točkom i pretvaramo stupac u numerički tip podataka.

```
def klasa(df):
    if (df["Ocjena"] < 7)  :
        return "Prosječna ocjena"
    elif ((df["Ocjena"] >= 7) and (df["Ocjena"] < 8)):
        return "Dobar"
    elif ((df["Ocjena"] >= 8) and (df["Ocjena"] <= 8.5)):
        return "Vrlo dobar"
    elif ((df["Ocjena"] >= 8.6) and (df["Ocjena"] < 9)):
        return "Sjajan"
    elif ((df["Ocjena"] >= 9) and (df["Ocjena"] < 9.5)):
        return "Izvanredan"
    else:
        return "Izuzetan"

rec["Klasa"] = rec.apply(klasa, axis = 1)
rec["Klasa"] = rec["Klasa"].astype("category")
```

S obzirom na to da nemamo klasu kao klasu, već samo recenziju, definirana je funkcija za stvaranje stupca s nazivom klase. Rasponi si prikupljeni već iz prethodnog koraka analize, kada smo pregledali raspon recenzija unutar svake klase. Također, navedena varijabla pretvorena je u kategorički tip podataka. Unutar recenzija korisnici su koristili emotikone te je iste potrebno ukloniti iz teksta i naslova.

```
def makni_emoji(string):
    emoji_pattern = re.compile("["
                            u"\U0001F600-\U0001F64F"  # emotikoni
                            u"\U0001F300-\U0001F5FF"  # simboli
                             u"\U0001F680-
                             \U0001F6FF"  # simboli transporta
                            u"\U0001F1E0-\U0001F1FF"  # zastave (ios)
                            u"\U00002702-\U000027B0"
                            u"\U000024C2-\U0001F251"
                            "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', string)
rec["Tekst"] = rec['Tekst'].apply(makni_emoji)
```

Na internetu pronađeni su uzorci korišteni za prikaz emotikona. Navedena funkcija sve navedene uzorka zamjenjuje s praznim stringom te istu aplicirano na varijablu naslova i teksta skupa podataka.

```
rec["Tekst"] = rec["Tekst"].replace("Nema komentara u ovoj recenziji",
"")
```

Neke recenzije nemaju komentara već samo cijenu, te s obzirom na to da se kod takvih recenzija u tekstu prikazuje isti tekst – „*Nema komentara u ovoj recenziji*“, isti unutar teksta zamjenjujemo s praznim stringom.

```
import string
def micanje_specijalnih_znakova(string):
    clean = re.sub(r"[,.;@#?!&$]+\ *", " ", string)
    return clean
rec["Naslov"] = rec["Naslov"].apply(micanje_specijalnih_znakova)
rec["Tekst"] = rec["Tekst"].apply(micanje_specijalnih_znakova)
```

Također, iz teksta je potrebno ukloniti specijalne znakove te unutar definirane funkcije navedene znakove zamjenjujemo razmakom. Istu funkciju primjenjujemo na varijablu naslova i teksta recenzije.

```
rec['Ukupan tekst'] = rec['Naslov'].str.cat(rec['Tekst'],sep=" ")
rec.drop(["Naslov", "Tekst"], inplace = True, axis = 1)
```

Stvaramo novi stupac *Ukupan tekst* koji sadrži konkateniran sadržaj naslova i teksta recenzije te navedene dvije varijable možemo izbaciti iz skupa podataka.

```
from langdetect import detect
def jezik(text):
        return detect(text)

rec["Jezik"] = rec['Ukupan ekst'].apply(jezik)
rec["Jezik"].value_counts()
```

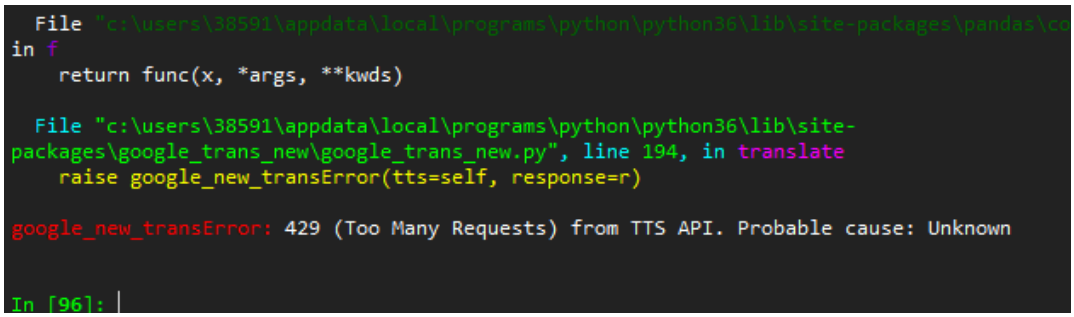
Kako u skupu podataka imamo različite jezike, koristimo *langdetect* funkciju *detect* pomoću koje unutar funkcije otkrivamo na kojem jeziku je pisana recenzija.

```
rec["Jezik"].value_counts()
Out[43]:
en    10390
de     4213
hr     3857
fr     1911
it     1568
es     1450
pl     1277
...
```

Kako postoji biblioteka za prijevod teksta, proveden je pokušaj prijevoda s npr. njemačkog jezika na engleski kako bi imali veći broj recenzija.

```
from google_trans_new import google_translator
rec_de = rec[rec["Jezik"] == "de"]
translator = google_translator()
rec_de['Ukupan tekst'] = rec_de['Ukupan tekst'].apply(translator.transl
ate, lang_src='de', lang_tgt='en')
```

No, prilikom izvođenja navedenog djela koda, nakon određenog vremena u konzoli iskoči upozorenje:



Slika 54. To many request

gdje vidimo da je poslano previše zahtjeva za dotičan API. Isti test proveden je za još dvije biblioteke, no za sve je broj zahtjeva bio prevelik te je odlučeno da ćemo odvojeno analizirati engleski i hrvatski tekst.

```
rec_eng = rec[rec["Jezik"] == "en"]
ec_hrv = rec[rec["Jezik"] == "hr"]
```

Podaci su filtrirani prema oznaci za jezik i stvoreni su odvojeni skupovi podataka za engleski i hrvatski jezik.

5.2. Analiza engleskih recenzija

U ovom poglavlju analizirani su tekstovi recenzija na engleskom jeziku.

Za početak, uklanjamo sve zaustavne riječi iz teksta (kao što su *the, and, ...*).

```
from nltk.corpus import stopwords
nltk.download('stopwords')
stop = stopwords.words('english')
```

Unutar *nltk* biblioteke skinut je modul koji sadrži zaustavne riječi za engleski jezik.

```
rec_eng["Ukupan tekst"] = rec_eng["Ukupan tekst"].str.lower()
rec_eng['Tekst'] = rec_eng['Ukupan tekst'].apply(lambda x: ' '.join([wo
rd for word in x.split() if word not in (stop)]))
```

Nakon što su sva slova unutar stupca s tekstom pretvorena u mala slova (engl. *lowercase*), pomoću razmaka spajamo sve riječi unutar navedenog stupca koje se ne nalaze u prethodno definiranoj varijabli zaustavnih riječi engleskog jezika.

```
from sklearn.feature_extraction import text
stop = list(text.ENGLISH_STOP_WORDS)
```

Također, lista zaustavnih riječi za engleski jezik postoji i unutar *sklearn* biblioteke, te je isti proces proveden i za navedenu listu zaustavnih riječi.

```
rec_eng.drop("Ukupan tekst", axis = 1, inplace = True)
```

S obzirom na to da imamo novo uređenu varijabli *Tekst* s tekstom bez zaustavih riječi možemo varijablu *Ukupan tekst* ukloniti iz skupa podataka.

Kako bi mogli koristiti *pos* tagiranje, potrebno je skinuti dva navedena modula:

```
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

kako bi mogli tokenizirati riječi i na temelju oznake (vrste) riječi izlučiti samo imenice iz ukupnog teksta.

Tag	Freq	Example	Comment
VBN	61	*burnt, gone*	verb: past participle
VB	51	*make, achieve*	verb: base form
VBD	36	*saw, looked*	verb: simple past tense
JJ	30	*ambiguous, acceptable*	adjective
VBZ	24	*sees, goes*	verb: third-person singular present
IN	18	*by, in*	preposition
AT	18	*a, this*	article

Slika 55. Primjer oznaka nltk biblioteke, preuzeto sa: http://nltk.sourceforge.net/doc/en/ch03.html

Svaka oznaka (vrsta riječi) sastoji se od minimalno jednog znaka (*tag* stupac) te vidimo iz primjera da oznake za glagole sve počinju slovom V (kao *Verb*) te za imenice znači da sve oznake započinju slovom N (kao *Noun*).

```
rec_eng["Tekst"] = rec_eng["Tekst"].apply(lambda x: " ".join([word for
(word, pos) in nltk.pos_tag(nltk.word_tokenize(x)) if pos[0] == 'N'])
rec_eng.drop("Tekst", axis = 1, inplace = True)
```

Stvaramo novi stupac koji se sastoji od svih riječi unutar stupca *Tekst* čija *pos* oznaka započinje s *N*.

Ocjena	Klasa	Jezik	Tekst
10.0	Izuzetan	en	quiet location cleanliness
10.0	Izuzetan	en	great nice modern room new sparkling clean …
10.0	Izuzetan	en	transit spent just night nice location easy…
9.0	Izvanredan	en	nice stay
10.0	Izuzetan	en	perfect ways location food prices great pen…
7.0	Dobar	en	good location value money good location peo…
10.0	Izuzetan	en	review place easily points best bed slept

Slika 56. Izgled uređenog skupa podataka

Na slici 56. prikazan je izgled novouređenog skupa podataka, koji se sastoji od same recenzije, klase recenzije, jezik teksta i sam tekst recenzije koji se sastoji uglavnom od imenica.

Daljnji cilj analize jest provesti analizu frekvencije riječi te za svaku klasu posebno vidjeti koja se kombinacija dvije riječi, zatim i samo jedne riječi, pojavljuju najčešće kod recenzije dotične klase.

```
grupirano = rec_eng.groupby("Klasa")
eng_grupirano = grupirano[["Tekst"]].agg(lambda column: " ".join(column
)).reset_index()
prosjecna = list(eng_grupirano[eng_grupirano["Klasa"] == "Prosječna ocj
ena"]["Tekst"])[0]
```

Skup podataka grupiramo prema klasi te stvaramo posebnu listu riječi za svaku pojedinu klasu.

dobar - List (4840 elements)

Inde	Type	Size	
0	str	8	location
1	str	5	value
2	str	5	money
3	str	8	location
4	str	6	people
5	str	7	highway
6	str	5	value
7	str	5	money
8	str	4	room

Slika 57. Izgled liste riječi za pojedinu klasu

Na slici 57. prikazano je kako izgleda prethodno napravljena lista riječi za pojedinu klasu.

```
prosjecna = re.sub("[^\w]", " ",  prosjecna).split()
```

Slijedeći korak je stvaranje stringa (teksta) koji sadrži riječi iz navedene liste odvojene razmakom.

```
location value money location people highway value money room bathroom bed breakfast hotel
staff location location breakfast staff breakfast service school hotel town self reception
comfy room comfy air rooms staff restaurant parking service dishes stay staff terrace room
night stay dog rent house experience remote breakfast dinner value location staff food
restaurant hotel danube river sights outdoor area views stuff hotell sourounding tidy
breakfast owner room staff food location nature restaurant dinner dinner room hotel city
staff polite helpful staff rooms value money location covid comfy location parking place
stay location facilities hotel restaurant staff location cool establishment belowaverage
customer service location rooms property lot staff downsides discount room price sleepover
wines dinner location door cellars people place stuff courtyard location backyard beds
groups families place service excellent youre car city centre stay improvement variety
breakfast town location room interesting keys place dinner portions roomy staff hotel
location pool animals weather views boat trips pool peacefulness property host clean
apartment shiny location proximity sea excursion staff terrace bar location facilities
room staff location interior location views location town area nice location location
views location price point hotel town dubrovnik terrace sea area views location closeness
town restaurant areas location quality service food facility hotel rate bit town minutes
summer day transfer stay town place pool frontage cocktails rooms location views hotel
reception staff place issues location room views upgrade room rooms dark wall road hotel
```

Slika 58. Izgled stringa riječi

Na slici 58. prikazano je kako izgleda string riječi kreiran iz liste riječi za pojedinu klasu.

```
from nltk.collocations import BigramCollocationFinder
```

```
finder = BigramCollocationFinder.from_words(prosjecna, window_size = 2)
ngram_prosjecna_eng = list(finder.ngram_fd.items())
ngram_prosjecna_eng.sort(key=lambda item: item[-1], reverse=True)
```

Stvaramo *finder* objekt unutar kojeg se nalaze bigrami riječi (sve kombinacije riječi od dvije riječi definirane kroz parametar *window_size*) iz kojeg je izrađena lista narednih bigrama koji su zatim sortirani od bigrama najčešće frekvencije (najčešćeg pojavljivanja) do bigrama najmanje frekvencije.

Inde	Type	Size	Value
0	tuple	2	(('location', 'location'), 30)
1	tuple	2	(('staff', 'location'), 24)
2	tuple	2	(('city', 'center'), 23)
3	tuple	2	(('location', 'room'), 19)
4	tuple	2	(('value', 'money'), 17)
5	tuple	2	(('location', 'town'), 17)

Slika 59. Izgled liste bigrama za pojedinu klasu

Na slici 59. prikazano je kako izgleda navedena lista frekvencija bigrama za klasu *dobar*. Isti postupak proveden je za sve klase recenzija. Daljnji postupak analize temelji se na analizi deset najčešćih bigrama pojedinih klasa.

Prosječna ocjena: analizom bigrama klase *Prosječna ocjena* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: pogled na more, pogled s lokacije, osoblje unutar smještaja, gradski centar te doručak unutar smještaja.

Dobar: analizom bigrama klase *Dobar* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: osoblje unutar smještaja, gradski centar, vrijednost smještaja za novac, osoblje unutar hotela i na samu lokaciju u kontekstu da se smještaj nalazi u gradu.

Vrlo dobar: analizom bigrama klase *Vrlo dobar* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: vrijednost smještaja za novac, gradski centar, lokaciju hotela, sobu unutar lokacije, lokaciju u odnosu na grad, osoblje smještaja i doručak smještaja.

Sjajan: analizom bigrama klase *Sjajan* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: osoblje unutar smještaja, autobusnu stanicu u blizini smještaja, gradski centar, lokaciju smještaja, osoblje smještaja, doručak unutar smještaja te vrijednost smještaja za novac.

Izvanredan: analizom bigrama klase *Izvanredan* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: osoblje unutar smještaja, gradski centar, vrijednost smještaja za novac, lokaciju apartmana, osoblje hotela te grad same lokacije.

Izuzetan: analizom bigrama klase *Izuzetan* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: lokaciju smještaja, osoblje smještaja te vrijednost smještaja za novac.

Analizom bigrama klasa vidimo da se najčešće pojavljuju iste kombinacije riječi (*location – staff, city-center, value-money,* itd.) te da je samo drugačiji redoslijed pojavljivanja određene lokacije. Npr. vidimo da se u kontekstu klase *Prosječna ocjena* ne pojavljuje bigram *value-money* dok se kod drugih klasa spominje ili da se bigram *staff-stay* spominje unutar svake klase ali da kod svake ima različit prioritet.

```
wordfreq = []
for w in prosjecna:
    wordfreq.append(prosjecna.count(w))
jedne_prosjecna = list(zip(prosjecna, wordfreq))
jedne_prosjecna = set(list(jedne_prosjecna))
jedne_prosjecna.sort(key=lambda item: item[-1], reverse = True)
```

Kod analize monograma (najčešćih riječi) stvaramo listu frekvencija riječi za svaku riječ unutar skupa riječi te gledamo unikatne kombinacije riječi i frekvencija koje zatim sortiramo silazno.

Prosječna ocjena: najčešće riječi za navedenu klasu su redom: lokacija (202 pojavljivanja), soba (137 pojavljivanja), pogled (96 pojavljivanja), hotel (87 pojavljivanja), osoblje (82 pojavljivanja), doručak (64 pojavljivanja), apartman (57 pojavljivanja), grad (40 pojavljivanja), smještaj (39 pojavljivanja) te more (37 pojavljivanja).

Dobar: najčešće riječi za navedenu klasu su redom: lokacija (330 pojavljivanja), soba (166 pojavljivanja), osoblje (153 pojavljivanja), hotel (111 pojavljivanja), apartman (104

pojavljivanja), pogled (98 pojavljivanja), grad (89 pojavljivanja), smještaj (83 pojavljivanja), doručak (81 pojavljivanje), okolno područje (69 pojavljivanja).

Vrlo dobar: najčešće riječi za navedenu klasu su redom: lokacija (329 pojavljivanja), soba (181 pojavljivanje), osoblje (149 pojavljivanja), hotel (127 pojavljivanja), apartman (105 pojavljivanja), doručak (104 pojavljivanja), grad (86 pojavljivanja), pogled (84 pojavljivanja), smještaj (76 pojavljivanja), domaćin (62 pojavljivanja).

Sjajan: najčešće riječi za navedenu klasu su redom: lokacija (236 pojavljivanja), osoblje (134 pojavljivanja), soba (131 pojavljivanje), apartman (92 pojavljivanja), doručak (71 pojavljivanja), hotel (68 pojavljivanja), pogled (64 pojavljivanja), domaćin (51 pojavljivanje), bazen (47 pojavljivanja), područje (44 pojavljivanja).

Izvanredan: najčešće riječi za navedenu klasu su redom: lokacija (604 pojavljivanja), osoblje (384 pojavljivanja), soba (355 pojavljivanja), apartman (209 pojavljivanja), hotel (256 pojavljivanja), doručak (238 pojavljivanja), pogled (212 pojavljivanja), grad (202 pojavljivanja), domaćin (180 pojavljivanja), područje (107 pojavljivanja).

Izuzetan: najčešće riječi za navedenu klasu su redom: lokacija (2251 pojavljivanje), apartman (1643 pojavljivanja), osoblje (1509 pojavljivanja), soba (1325 pojavljivanja), hotel (1105 pojavljivanja), domaćin (1006 pojavljivanja), doručak (867 pojavljivanja), grad (841 pojavljivanje), pogled (751 pojavljivanje), ljubaznost (391 pojavljivanje).

Vidimo da se tekstovi recenzija različitih klasa po najčešćim riječima ne razlikuju u velikoj mjeri. Uglavnom se kroz tekstove spominju slične riječi (kao što su osoblje, lokacija, pogled, grad i slično) uz par iznimaka (ljubaznost kod klase *Izvanredan,* bazen kod klase *Sjajan* itd.) no s neznatno drugačijim redoslijedom tj. prioritetom (kod klase *Prosječna ocjena* vidimo da su najvažniji lokacija, soba, pogled i doručak dok npr. kod klase izvanredan i izuzetan vidimo da je uz lokaciju i samu sobu smještaja također važno i ponašanje samog domaćina).

Jaccard indeks

Jaccard indeks jedna je od metoda za izračun sličnosti stringova bazirana na tokenima (riječima). Temelji se na pronalasku zajedničkih riječi te dijeljenju navedenog broja sa ukupnim brojem jedinstvenih riječi [7]. Formula za izračun Jaccard indeksa glasi:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

gdje brojnik predstavlja presjek skupova (broj zajedničkih riječi), a nazivnik uniju skupova (ukupan broj riječi korištenih u listama). Vrijednosti Jaccard indeksa nalaze se u rasponu od 0 do 1 te što je vrijednost bliža 1, sličnost dvaju tekstova je viša.

```
prosjecna = " ".join(list(set(prosjecna)))
```

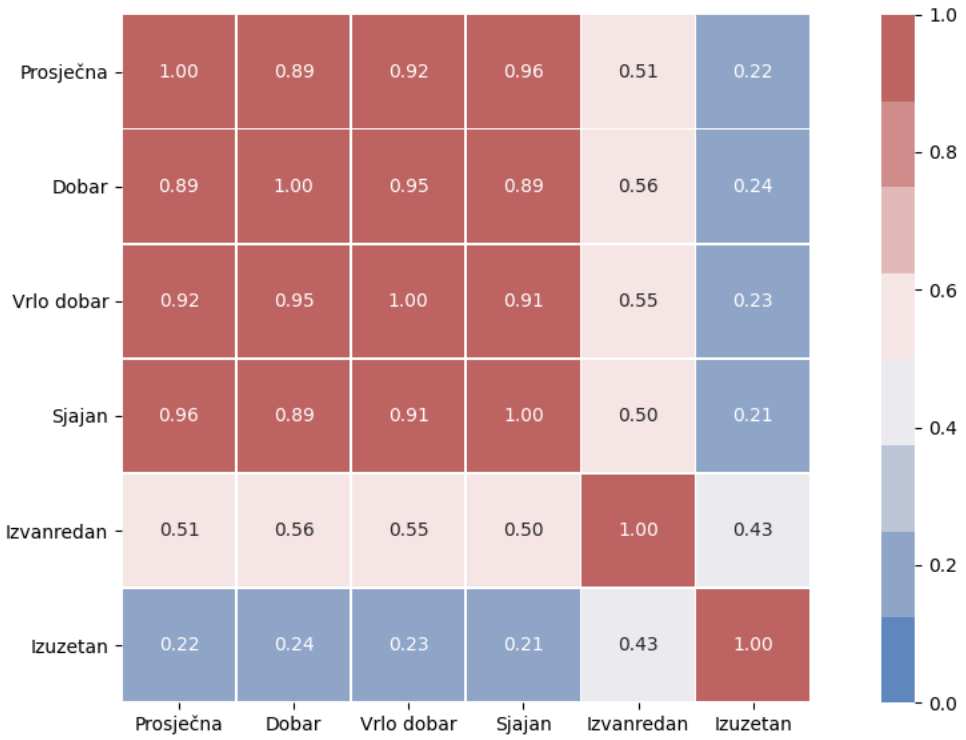
Za svaku klasu stvaramo string riječi iz prethodno kreiranih lista riječi.

```
import textdistance
from io import StringIO
data = [prosjecna, dobar, vrlodobar, sjajan, izvanredan, izuzetan]
dm = [[ textdistance.jaccard(a, b) for b in data] for a in data]
textdistance_table ='\n'.join(['' .join([f'{item:6.2f}' for item in row]
) for row in dm])
```

Nakon učitavanja potrebnih modula i biblioteka, stvaramo zajedničku listu stringova svih klasa te računamo Jaccard indeks za svaku riječ unutar svakog stringa unutar liste. Zatim stvaramo tablicu s izračunatim indeksima između svake pojedine klase objekta.

```
df = pd.read_csv(StringIO(re.sub(r'[-
+|]', '', textdistance_table)), sep='\s{2,}', engine='python', names =
["Prosječna", "Dobar", "Vrlo dobar", "Sjajan", "Izvanredan", "Izuzetan"
], header=None)
df = df.rename(index = { 0 : "Prosječna", 1: "Dobar", 2: "Vrlo dobar",
3: "Sjajan", 4: "Izvanredan", 5: "Izuzetan"})
h = sns.heatmap(df, annot=True, cmap=colormap, linewidth=.5, square = T
rue, linecolor='w', fmt='.2f', vmin=0, vmax=1, center=0.5)
h.set_xticklabels(h.get_xticklabels(), rotation=0)
```

Zatim stvaramo *Data Frame* iz dobivene tablice, preimenujemo indekse i stupce te vizualiziramo dobiveni *Data Frame* u obliku matrice.



Slika 60. Jaccard indeks matrica za engleske recenzije

Iz matrice na slici 60. možemo vidjeti da je sadržaj recenzija za klase *Prosječna ocjena, Dobar, Vrlo dobar i Sjajan* veoma sličan gdje je Jaccard indeks između svake klase najmanje 0.89 dok su klase *Izvanredan* i *Izuzetan* sadržajem recenzija međusobno i u odnosu na prethodne 4 klase veoma različite. Sličnost jedne s drugom je 0.43 dok se sadržaj recenzije klase *Izuzetan* od prve četiri klase podosta razlikuje (Jaccard indeks sa svakom je otprilike 0.2) dok je Jaccard indeks klase *Izvanredan* s ostalim klasama otprilike 0.5 što znači da je tekst recenzija klase *Izvanredan* i sličniji tekstovima prve četiri klase nego klasi *Izuzetan,* dok se klasa *Izuzetan* iznimno razlikuje od svih drugih klasa.

5.3. Analiza hrvatskih recenzija

U ovom poglavlju analizirani su tekstovi recenzija na hrvatskom jeziku. Proces analize hrvatskih tekstova nešto je drugačiji od engleskih tekstova s obzirom na to da je *nltk* biblioteka fokusirana na analizu engleskih tekstova i tekstova drugih raširenih jezika (kao što su španjolski i njemački) te će se za analizu hrvatskih tekstova trebati koristiti druga biblioteka *spacy_udpipe.*

Za početak, sav tekst konvertiramo u mala slova i uklanjamo zaustavne riječi iz tekstova pronađene u obliku liste na github repozitoriju https://github.com/explosion/spaCy/blob/master/spacy/lang/hr/stop_words.py uz zaustavne riječi unutar *text_hr* biblioteke.

```
from text_hr import get_all_std_words
lista = []
for word_base, l_key, cnt, _suff_id, wform_key, wform in get_all_std_wo
rds():
    lista.append(wform)
```

Lista iz spomenute poveznice spremljena je u varijablu *res* te stvaramo zajedničku listu zaustavnih riječi.

```
lista = res + zaustavne
pattern = r'\b(?:{})\b'.format('|'.join(lista))
rec_hrv['Ukupan tekst'] = rec_hrv["Ukupan tekst"].str.replace(pattern,
'')
```

Stvaramo regularni izraz zaustavnih riječi koje unutar tekstova recenzija zamjenjujemo praznim stringom.

```
lista = [sub.replace('š', 's') for sub in lista]
pattern = r'\b(?:{})\b'.format('|'.join(lista))
rec_hrv["Ukupan tekst"] = rec_hrv["Ukupan tekst"].apply(lambda x: ' '.j
oin([word for word in x.split()]))
```

Također, s obzirom na to da su recenzije pisane od strane korisnika (te se ne radi o formalnom tekstu kao što je npr. novinski članak) mijenjamo dijakritičke znakove (č, ć, š, đ, ž) u verzije istih bez crtica i kvačica (c, s, d i z). Zatim izmijenjene zaustavne riječi također uklanjamo iz teksta.

```
rec_hrv["Ukupan tekst"] = rec_hrv["Ukupan tekst"].apply(lambda x: ' '.j
oin([word for word in x.split() if not word.isdigit()]))
```

Zatim uklanjamo sve brojeve koji se nalaze unutar teksta.

Kako bi dohvatili samo imenice u tekstu, potreban nam je *custom spacy udpipe* model dostupan unutar biblioteke:

```
import spacy_udpipe
spacy_udpipe.download("hr")
```

```
nlp = spacy_udpipe.load("hr")
```

gdje navodimo jezik teksta za koji želimo skinuti model.

```
grupirano = rec_hrv.groupby("Klasa")
hrv_grupirano = grupirano[["Ukupan tekst"]].agg(lambda column: " ".join
(column)).reset_index()
```

Skup podataka grupiramo u odnosu na klase

```
prosjecna_hrv = hrv_grupirano[hrv_grupirano["Klasa"] == "Prosječna ocje
na"]["Ukupan tekst"][3]
```

i stvaramo posebne varijable za svaku pojedinu klasu objekta (kao i kod engleskog jezika).

```
doc = nlp(prosjecna_hrv)
imenice = []
for token in doc:
...     if token.pos_ == 'NOUN':
...         imenice.append(token.text)
```

Zatim, za svaku klasu posebno, stvaramo *nlp* dokument tokeniziranih riječi te ako je vrsta navedene riječi imenica, navedenu riječ pridodajemo prethodno izrađenoj listi *imenice*.

```
from nltk import BigramCollocationFinder

imenice = ' '.join([str(word) for word in imenice])
prosjecna_hrv = re.sub("[^\w]", " ",  imenice).split()
finder = BigramCollocationFinder.from_words(prosjecna_hrv, window_size
= 2)
ngram_prosjecna_hrv = list(finder.ngram_fd.items())
ngram_prosjecna_hrv.sort(key=lambda item: item[-1], reverse=True)
```

Kao i kod engleskog jezika, listu s imenicama pretvaramo u string riječi te stvaramo objekt *finder* koji sadrži kombinacije svake dvije riječi unutar teksta i njihove pripadajuće frekvencija kako bi pronašli 10 najčešćih kombinacija dvije riječi unutar recenzija.

Prosječna ocjena: analizom bigrama klase *Prosječna ocjena* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: lokaciju smještaja, osoblje smještaja, osoblje na recepciji, prozor sobe, blizinu smještaja centru, sobu i kupaonicu smještaja, doručak smještaja i parking smještaja.

Dobar: analizom bigrama klase *Dobar* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: blizinu centra grada smještaja, parking smještaja, osoblje smještaja, doručak, položaj objekta, lokaciju smještaja u odnosu na cijenu, odnos kvalitete i cijene smještaja.

Vrlo dobar: analizom bigrama klase *Vrlo dobar* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: doručak, osoblje smještaja, blizinu centra grada, vrijednost smještaja za novac, količinu vremena potrebnu da se od smještaja dođe do centra grada.

Sjajan: analizom bigrama klase *Sjajan* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: kvalitetu lokacije, terasu smještaja, sobu, doručak, uređenje sobe smještaja, blizinu centra grada, parking smještaja, osoblje smještaja i osoblje na recepciji.

Izvanredan: analizom bigrama klase *Izvanredan* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: blizinu centra grada, kvalitetu lokacije, osoblje smještaja, doručak, vrijednost smještaja za novac, mir i tišinu unutar smještaja te omjer cijene i kvalitete smještaja.

Izuzetan: analizom bigrama klase *Izuzetan* dolazimo do toga da se korisnici, u kontekstu ove klase, najčešće referiraju redom na: blizinu centra grada, mir i tišinu unutar smještaja, kvalitetu lokacije smještaja, vrijednost smještaja za novac, osoblje smještaja, doručak, parking smještaja te ljubaznost domaćina.

Vidimo da se u tekstu hrvatskih, kao i engleskih recenzija, kroz različite klase objekta također spominju slični izrazi ali u drugačijem redoslijedu (npr. kod klase *Prosječna ocjena* korisnici se najviše referiraju na samu lokaciju i osoblje smještaja dok se npr. kod klase *Izvanredan* i *Izuzetan* najviše referiraju na blizinu centra grada, vrijednost objekta za novac, mir i tišinu u kontekstu smještaja). Lokacija se gleda u kontekstu svih klasa, što je i logično no vidimo da, kako gledamo različite klase, kako negdje lokacija ima veći prioritet dok kod recenzija drugih klasa, prema sadržaju recenzija korisnika, veći prioritet ima samo osoblje, parking i mir i tišina unutar objekta gdje korisnici više gledaju udobnost same sobe i objekta smještaja nego lokaciju na kojoj se objekt nalazi - što, kad pogledamo korelacijski dijagram na slici 47. vidimo da je istina te da recenzija udobnosti od svih drugih recenzija najviše utječe na samu ocjenu objekta od strane korisnika.

```
wordfreq = []
for w in prosjecna_hrv:
    wordfreq.append(prosjecna_hrv.count(w))
jedne_prosjecna_hrv = list(zip(prosjecna_hrv, wordfreq))
jedne_prosjecna_hrv = list(set(jedne_prosjecna_hrv))
jedne_prosjecna_hrv.sort(key=lambda item: item[-1], reverse = True)
```

Kao i kod engleskog jezika, analiziramo i najčešće riječi koje se pojavljuju u tekstu pojedine klase.

Prosječna ocjena: najčešće riječi za navedenu klasu su redom: lokacija (55 pojavljivanja), osoblje (26 pojavljivanja), sobe (20 pojavljivanja), krevet (19 pojavljivanja), doručak (14 pojavljivanja), hotel (13 pojavljivanja), blizina (10 pojavljivanja), objekt (10 pojavljivanja), boravak (10 pojavljivanja), cijena (9 pojavljivanja).

Dobar: najčešće riječi za navedenu klasu su redom: lokacija (58 pojavljivanja), osoblje (35 pojavljivanja), doručak (21 pojavljivanje), parking (19 pojavljivanja), grad (15 pojavljivanja), soba (13 pojavljivanja), centar (12 pojavljivanja), hotel (12 pojavljivanja), pogled (10 pojavljivanja), krevet (9 pojavljivanja).

Vrlo dobar: najčešće riječi za navedenu klasu su redom: lokacija (67 pojavljivanja), doručak (39 pojavljivanja), osoblje (27 pojavljivanja), grad (19 pojavljivanja), soba (16 pojavljivanja), parking (13 pojavljivanja), centar (12 pojavljivanja), hotel (11 pojavljivanja), blizina (10 pojavljivanja), pogled (9 pojavljivanja).

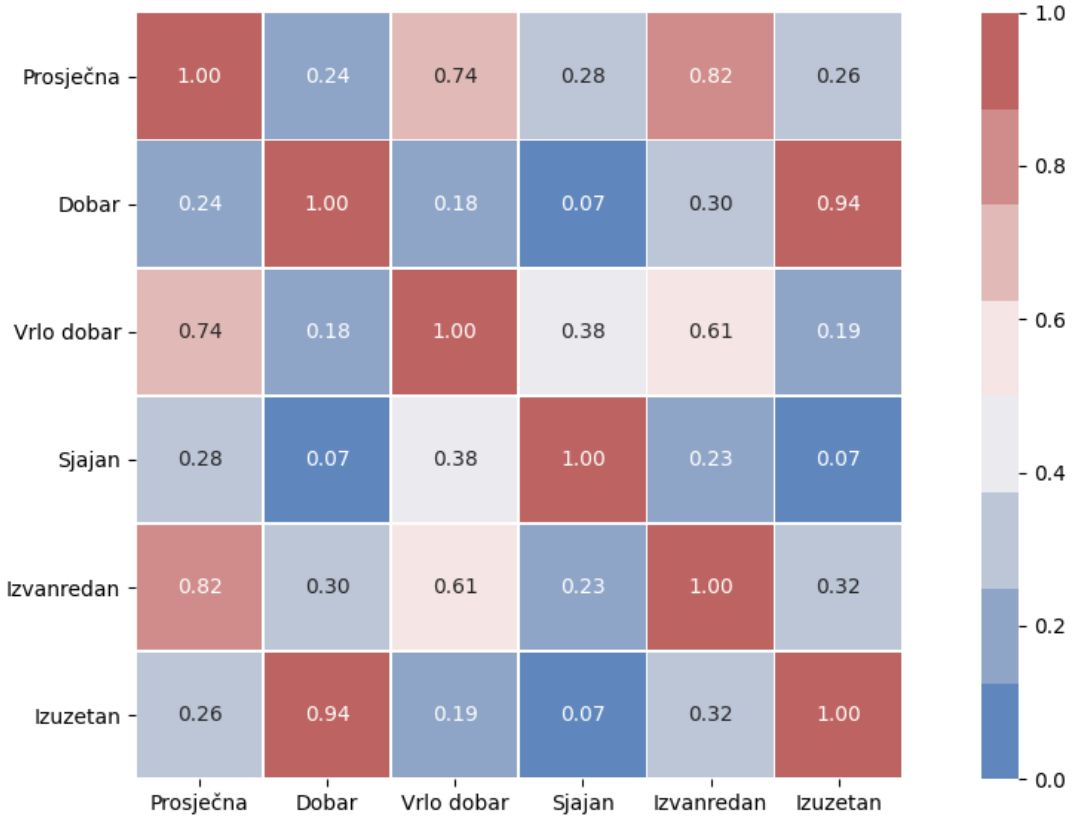
Sjajan: najčešće riječi za navedenu klasu su redom: lokacija (26 pojavljivanja), soba (11 pojavljivanja), osoblje (10 pojavljivanja), parking (9 pojavljivanja), doručak (6 pojavljivanja), objekt (5 pojavljivanja), grad (5 pojavljivanja), blizina (4 pojavljivanja), hrana (4 pojavljivanja), domaćin (3 pojavljivanja).

Izvanredan: lokacija (136 pojavljivanja), osoblje (76 pojavljivanja), doručak (57 pojavljivanja), grad (46 pojavljivanja), boravak (39 pojavljivanja), hotel (36 pojavljivanja), soba (35 pojavljivanja), domaćin (30 pojavljivanja), centar (27 pojavljivanja), blizina (26 pojavljivanja), krevet (23 pojavljivanja).

Izuzetan: lokacija (573 pojavljivanja), osoblje (345 pojavljivanja), apartman (317 pojavljivanja), doručak (230 pojavljivanja), boravak (173 pojavljivanja), domaćin (165

pojavljivanja), pohvale (146 pojavljivanja), soba (142 ponavljanja), grad (141 pojavljivanje), odmor (137 pojavljivanja).

Kao i kod engleskog jezika, vidimo da se prvih par klasa fokusira na samu lokaciju, sobu, osoblje i doručak smještaja, dok se kod zadnje tri klase počinju češće spominjati sam domaćin smještaja i kvaliteta odmora dobivena boravkom unutar smještaja.



Slika 61. Jaccard indeks hrvatskih recenzija po klasi

Na slici 61. nalazi se tablica Jaccard indeksa za tekstove hrvatskih recenzija u odnosu na pojedine klase. Matrica izgleda znatno drugačije nego kod engleskog jezika te vidimo da su tekstovi recenzija prosječno ocijenjenih objekata najsličniji tekstovima objekata vrlo dobre i izvanredne ocjene, tekstovi dobro ocijenjenih objekata su najsličniji tekstovima izuzetnih objekata, tekstovi vrlo dobro ocijenjenih objekata su najsličniji tekstovima prosječno ocijenjenih objekata i izvanrednim objektima, tekstovi sjajnih objekata imaju najmanju sličnost s tekstovima recenzija ostalih klasa no najsličniji su tekstovima vrlo dobro ocijenjenih objekata, tekstovi izvanrednih hotela najsličniji su tekstovima vrlo dobro ocijenjenih objekata i prosječno ocijenjenim objektima te su tekstovi izuzetnih objekata najsličniji dobro ocijenjenim objektima.

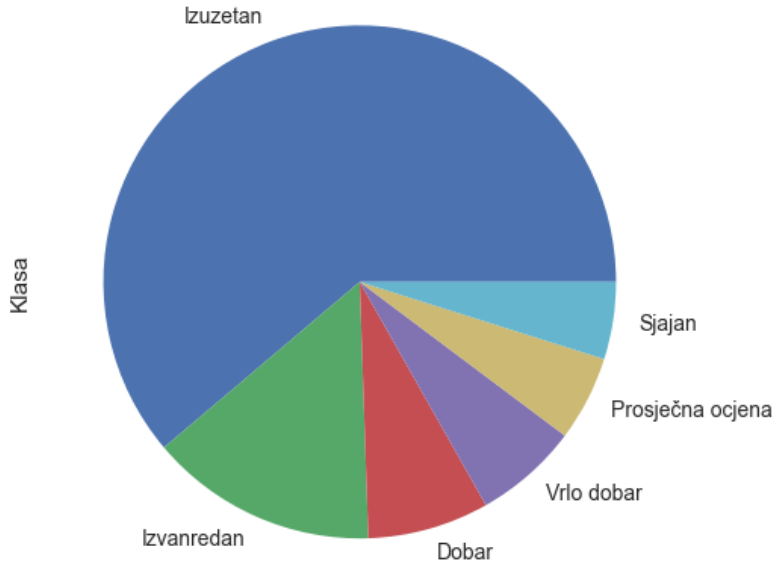
5.4. Izrada modela strojnog učenja na temelju tekstova recenzija

U ovom poglavlju opisani su koraci preprocesiranja podataka, algoritam korišten za klasifikaciju recenzija kao i sami rezultati dobiveni korištenjem navedenog algoritma.

Klasifikacija je provedena posebno za recenzije na engleskom i hrvatskom jeziku te su uspoređeni dobiveni rezultati za navedena dva skupa.

```
rec_eng["Klasa"].value_counts(normalize = True).plot.pie()
```

Za početak provjeravamo udio svake klase unutar skupa podataka engleskih recenzija.



Slika 62. Udio klasa engleskih recenzija

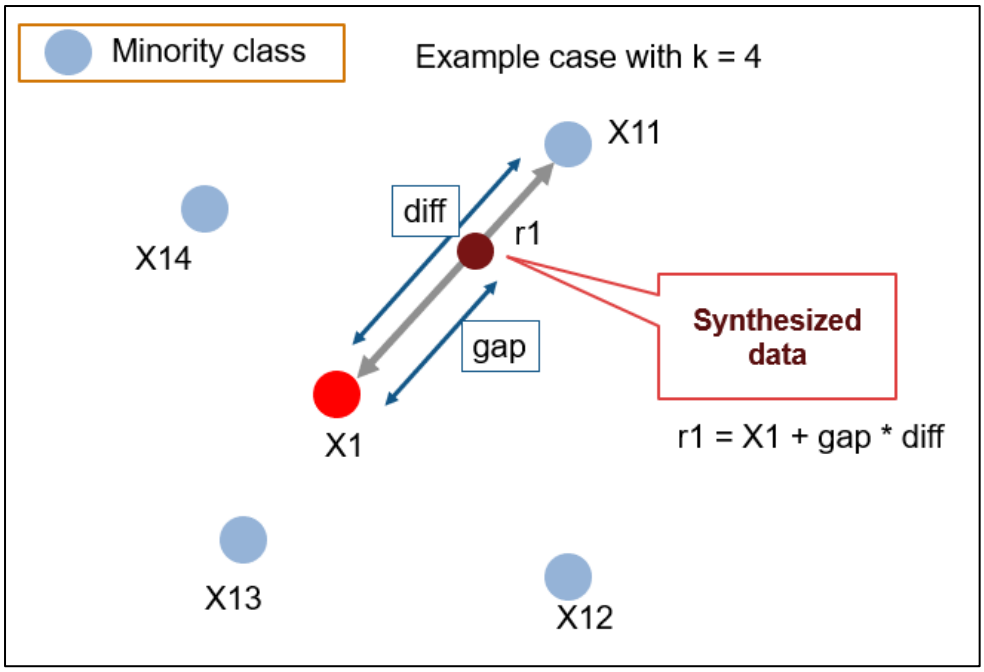
Na slici 62. prikazan je udio pojedine klase unutar skupa podataka s engleskim recenzijama. Možemo vidjeti da se većina recenzija (više od 50 %) odnosi na klasu *Izuzetan,* dok ostale klase sadrže znatno manji dio opservacija (recenzija). Navedeni problem možemo riješiti pomoću raznih metoda preuzorkovanja podataka, te je u ovom slučaju odabrana metoda SMOTE.

```
import nltk
nltk.download("wordnet")
tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()

def lematizacija(tekst):
    return [lemmatizer.lemmatize(rijec) for rijec in tokenizer.tokenize
(tekst)]
```

Prije daljnjeg procesiranja podataka, proveden je postupak lematizacije nad tekstovima recenzija, gdje riječi kao što su „stays“, „hotels“ i slične svodimo na lemu riječi – u ovim slučajevima „stay“ i „hotel“.

SMOTE

SMOTE (engl. *Synthetic Minority Over-sampling TEchnique*) je metoda preuzorkovanja podataka gdje se generiraju sintetički uzorci klasa s manjinskim udjelom uzoraka (u ovom slučaju, prilagođavamo broj opservacija ostalih klasa broju opservacija klase Izuzetan koja ima najviše uzoraka) . Manjinska klasa se prouzrokuje tako da se uzima svaki uzorak manjinske klase i stvaraju sintetički podaci u odnosu na k broj najbližih susjeda uzorka manjinske klase. Broj susjeda bira se u odnosu na postotak za koji je potrebno uvećati broj manjinske klase, npr. ako je potrebno broj instanci određene klase uvećati za 200 %, gledaju se dva susjeda navedenog uzorka i jedan uzorak je generiran u smjeru svakog susjeda. SMOTE metodu primjenjujemo zato što, u slučaju nebalansiranih podataka gdje imamo klasu koja sadrži većinu opservacija i klasu s manjinskim djelom opservacija (uzmimo u obzir slučaj 1:100 gdje 100 opservacija npr. predstavlja češću klasu kao što su generalni mailovi, a 1 predstavlja opservaciju manjinske klase kao što je u ovom slučaju spam mail), model je pristran većinskoj klasi te bi u ovom slučaju, ako predvidi sve klase kao većinsku klasu u skupu podataka, model imao točnost od 99 % nad skupom podataka za testiranje. Dobre performanse nad klasom koja sadrži većinu opservacija bila bi preferirana u odnosu na dobre performanse za obje klase. Ovakav slučaj nazivamo paradoksom točnosti, i potrebno je upotrijebiti jednu od metoda preuzorkovanja podataka kako bi model imao dobre performanse i točnost nad obje klase skupa podataka.



Slika 63. SMOTE - način rada, preuzeto sa: https://editor.analyticsvidhya.com/uploads/77417image1.png

Na slici 63. prikazan je način generiranja sintetičkih podataka: Uzima se razlika između značajnog vektora (uzorka) i njegovog najbližeg susjeda. Dobivena razlika množi se s nasumičnim brojem između 0 i 1 i pridodaje se uzorku koji trenutno promatramo. Ovaj postupak uzrokuje selekciju nasumične podatkovne točke u linijskom segmentu između navedena dva uzorka (originalnog uzorka i njegovog najbližeg susjeda). Konačno, svaka nasumična podatkovna točka jednaka je zbroju originalnog uzorka i umnoška razlike i nasumičnog broja [8].

```
from sklearn.feature_extraction.text import CountVectorizer
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report

x = rec_eng["Tekst"]
y = rec_eng["Klasa"]
```

Podaci su za početak podijeljeni u dvije varijable – *x* koja označava same tekstove recenzija pomoću koji će se provoditi klasifikacija i *y* koji sadrži klase po redovima i pomoću kojih će se model trenirati i na kraju predikcije modela usporediti sa stvarnim klasama skupa podataka (tekst recenzija uređen je tako da ne sadrži samo imenice).

```
x_1 = []
for recenica in x:
    x_1.append(recenica)
```

Tekstove recenzija pretvaramo u listu stringova gdje svaki string označava jednu recenziju.

Kako bi model mogao uspješno klasificirati recenzije, koristi se pristup *Bag of words,* koji predstavlja način ekstrakcije svojstva teksta u svrhu modeliranja.

*Slika 64. Pristup bag of words, preuzeto sa: https://miro.medium.com/max/451/1*eUedufAl7_sI_QWSEIstZg.png*

Na slici 64. vizualiziran je *bag of words* pristup. Recimo da imamo jednostavnu rečenicu, kao što je „Prekrasan hotel, ljubazno osoblje“, naš cilj je prezentirati navedenu rečenicu u obliku vektora tako da imamo 1 (pozitivno) u stupcima riječi koje rečenica sadrži, a 0 (negativno) pod stupcima riječi koje navedena rečenica ne sadrži. Navedeni vektor, uzimajući u obzir da gledamo nasumičnih 7 riječi, izgledao bi ovako:

```
|prekrasan| apartman | hotel | osoblje | lokacija | ljubazno | tišina|
|_________|__________|_______|_________|_________ |__________|_______|
|   1     |    0     |   1   |    1    |    0     |     1    |   0   |
```

Nadalje, oznake klasa (*y*) pretvaramo u integere (cijele brojeve) s definiranim redoslijedom.

```
cv=CountVectorizer(max_features=10000)
x_fin=cv.fit_transform(x_1).toarray()
sentiment_red = ['Prosječna ocjena', 'Dobar', 'Vrlo dobar', 'Sjajan', '
Izvanredan', 'Izuzetan']
y_fin = y.apply(lambda x: sentiment_red.index(x))
```

Konačno, *x_fin* je numpy polje dimenzija broja recenzija *x* broja riječi (nešto manje od 10 000), dok je *y_fin* jednodimenzionalni *pandas Series* objekt.

```
x_train, x_test, y_train, y_test = train_test_split(x_fin, y_fin, test_
size=0.3, random_state=1)
```

Stvaramo *x* i *y* za treniranje i testiranje modela pomoću navedene funkcije gdje navodimo prethodno definirane *x_fin* i *y_fin* te postotak vektora koji će se odvojiti za testiranje (u ovom slučaju 30 %). Parametar *random_state* navodimo radi mogućnosti repliciranja rezultata.

```
x_train.shape,y_train.shape
Out[13]: ((7252, 8981), (7252,))
```

Kada pogledamo dimenzije podataka za treniranje, možemo vidjeti da *x* ima 7252 reda (recenzija) i 8981 stupaca (od kojih svaki označava jedinstvenu riječ unutar recenzija) i da *y* ima također 7252 reda.

```
smote=SMOTE()
x_sm, y_sm = smote.fit_resample(x_train, y_train)
x_sm.shape, y_sm.shape

Out[16]: ((26412, 8981), (26412,))
```

Nakon upotrebe SMOTE metode i ponovnog pregleda dimenzija novih varijabli vidimo da *x* ima 26 412 redova, kao i *y*. Sada, kada pregledamo postotke klasa unutar *y* varijable dobijemo:

```
from collections import Counter
counter = Counter(y_sm)
for k,v in counter.items():
  per = v / len(y_sm) * 100
  print('Klasa=%d, Broj opservacija=%d (%.3f%%)' % (k, v, per))

Klasa=1, Broj opservacija=4402 (16.667%)
Klasa=5, Broj opservacija=4402 (16.667%)
Klasa=4, Broj opservacija=4402 (16.667%)
Klasa=0, Broj opservacija=4402 (16.667%)
Klasa=2, Broj opservacija=4402 (16.667%)
Klasa=3, Broj opservacija=4402 (16.667%)
```

gdje vidimo da svaka klasa ima jednak broj opservacija unutar skupa podataka (4 402) kao i udio klase unutar vektora *y* (16.667 %).

MULTINOMIAL NAIVE BAYES

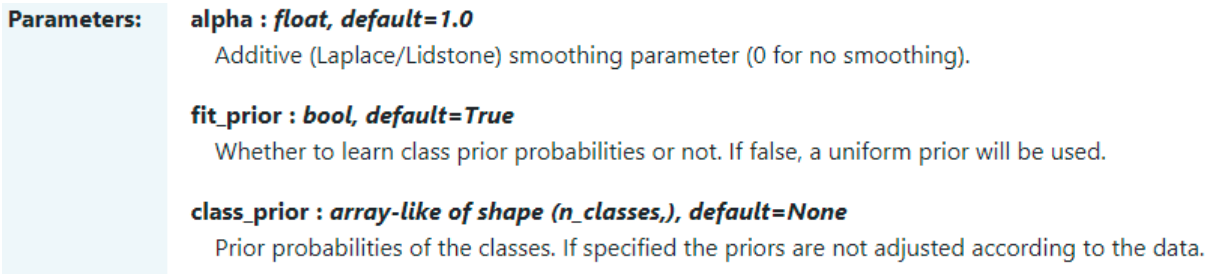
Multinomial Naive Bayes algoritam je za supervizirano strojno učenje (vrsta strojnog učenja u kojem model klasificira nove podatke na temelju podataka i oznaka podataka za treniranje) koji se najčešće koristi u području analize prirodnog jezika (engl. *Natural Language Processing*). Algoritam je baziran na Bayes teoremu, formuliranom od strane Thomasa Bayes-a, koji računa vjerojatnost određenog događaja koja je osnovana na prošlim uvjetima vezanim uz sam događaj. Teorem je baziran na formuli:

$$P(A|B) = P(A) * P(B|A)/P(B)$$

gdje računamo kolika je vjerojatnost klase A pod uvjetom prediktora B. P(A) označava prethodno definiranu vjerojatnost klase A, P(B|A) označava vjerojatnost pojavljivanja prediktora B u slučaju klase A, dok je P(B) prethodno definirana vjerojatnost prediktora B [9]. Pomoću navedene formule možemo izračunati vjerojatnost pojavljivanja određene oznake (riječi) unutar teksta određene klase. S obzirom na to da postoji i Naive Bayes algoritam, razlika je u tome što je multinomial Naive Bayes specijalizirana verzija Naive Bayes algoritma koji je prilagođen analizi tekstualnih dokumenata. Naive Bayes bi, pri analizi tekstualnog dokumenta, modelirao dokument kao prisustvo riječi ili manjak istih, dok multinomial Naive Bayes modelira frekvenciju riječi i prilagođava izračun dobivenom rezultatu.

```
model=MultinomialNB()
model.fit(x_sm, y_sm)
y_pred=model.predict(x_test)
cf=classification_report(y_test, y_pred)
print(cf)
```

Nakon definiranja modela (prema defaultnim vrijednostima) – slika 65:



Slika 65. Defaultni parametri za navedeni model, preuzeto sa: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

model treniramo nad podacima za treniranje te predikcije modela spremamo u posebnu varijablu. Kreiramo klasifikacijski izvještaj koji je prikazan na slici 65 i mjerimo točnost modela.

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
print(cf)
              precision    recall  f1-score   support

           0       0.69      0.53      0.60       170
           1       0.45      0.32      0.37       248
           2       0.61      0.46      0.52       197
           3       0.39      0.30      0.34       151
           4       0.53      0.49      0.51       434
           5       0.79      0.89      0.84      1917

    accuracy                           0.71      3117
   macro avg       0.58      0.50      0.53      3117
weighted avg       0.69      0.71      0.70      3117
```

```
accuracy_score(y_test, y_pred)
Out[126]: 0.7125441129290985
```

Dobiveni izvještaj prikazuje nekoliko stvari:

Preciznost modela (engl. *precision*) – omjer broja točno pogođenih instanci klase (*true positive*) i svih predviđenih rezultata navedene klase, bili oni netočno ili točno pozitivni. Vidimo da je preciznost najveća za klasu 5 (što je klasa *Izuzetan*) gdje je preciznost ~ 80 %, što znači da je od svih recenzija predviđenih kao recenzije izuzetnih hotela, model točno predvidio 80 %, što znači da je 20 % navedenih recenzija lažno pozitivno.

Opoziv (*engl. recall*) – omjer broja točno pogođenih instanci određene klase i ukupnog broja instanci određene klase. Ako pogledamo istu klasu (izuzetan) vidimo da navedeni omjer iznosi 90 % što znači da je od ukupnog broja instanci navedene klase njih 90 % pogođeno ispravno.

F_1 – F_1 mjera predstavlja omjer preciznosti i opoziva. Izračun se obavlja pomoću formule:

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Support – broj pojavljivanja određene klase unutar *y_test* varijable. Pojavljivanja klasa unutar navedene varijable su nebalansirana, jer je SMOTE metoda primijenjena samo na podacima za treniranje kako bi se model istrenirao na jednakom broju uzoraka svake klase.

Konačno, *accuracy_score* funkcija bavi se izračunom ukupne preciznosti modela, koja u ovom slučaju iznosi ~ 71 %.

K-struka unakrsna validacija

Unakrsna validacija predstavlja statističku metodu evaluacije i usporedbe algoritama strojnog učenja tako da se podaci dijele na dva dijela gdje se jedan segment podataka koristi za treniranje modela, a drugi za validaciju modela (testiranje).

Slika 66. Unakrsna validacija, preuzeto sa: https://es.mathworks.com/discovery/cross-validation/_jcr_content/mainParsys/image.adapt.480.medium.jpg/1611249630573.jpg

Kod k-struke unakrsne validacije, podaci se dijele na *k* jednake (ili približno jednake) dijelove (najčešće se koristi broj 10, gdje se podaci dijele na 10 dijelova), gdje se unutar svake iteracije (broj iteracija jednak je broju *k*) drugi segment koristi za validaciju, dok se ostali segmenti koriste za treniranje modela [10].

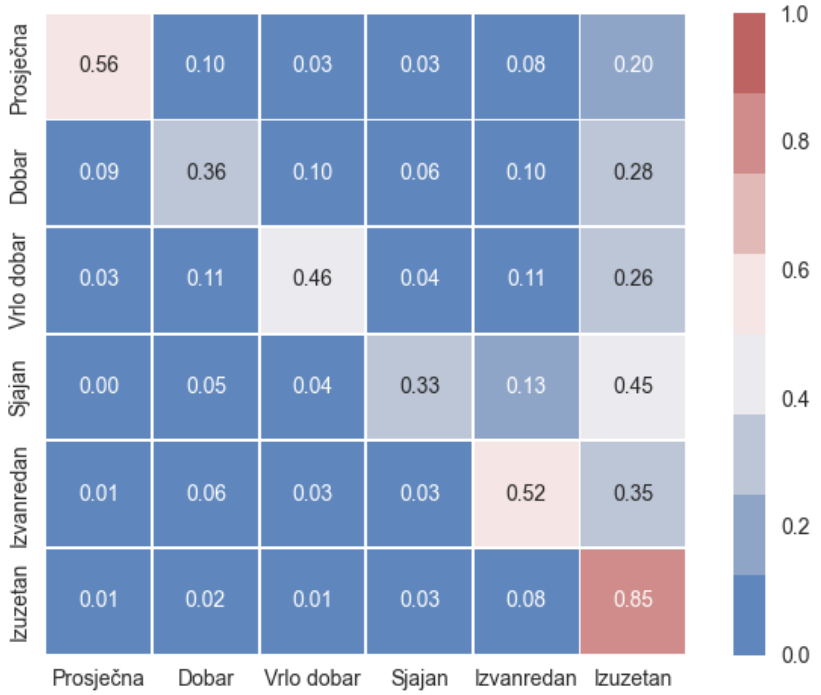
```
from sklearn.model_selection import GridSearchCV
parameters = {
    'alpha': [1, 0.7, 0.4, 0.2, 0.1, 0.09, 0.08, 0.07, 0.06, 0.03, 0.01
]
}

grid = GridSearchCV(model, param_grid=parameters, cv=10, refit=True)
grid.fit(x_sm, y_sm)
print(grid.best_score_)
print(grid.best_params_)
```

Promatramo li dokumentaciju, jedini parametar koji možemo promijeniti jest *alpha,* koji označava stopu učenja modela, tj. stopu promjene koeficijenata modela prilikom svakog ažuriranja. Moguće je *fit_prior* parametar postaviti na *False,* no to u ovom slučaju ne želimo.

Kada pogledamo koji *alpha* parametar pridonosi najboljem rezultatu modela dobijemo 1, što je defaultna vrijednost navedenog parametra te možemo navedeni model smatrati i konačnim za engleske recenzije.

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred, normalize = 'true')
cm = pd.DataFrame(cm)
cm = cm.rename(index = { 0 : "Prosječna", 1: "Dobar", 2: "Vrlo dobar",
3: "Sjajan", 4: "Izvanredan", 5: "Izuzetan"})
cm = cm.rename(columns = { 0 : "Prosječna", 1: "Dobar", 2: "Vrlo dobar"
, 3: "Sjajan", 4: "Izvanredan", 5: "Izuzetan"})
h = sns.heatmap(cm, annot=True, cmap=colormap, linewidth=.5, square = T
rue, linecolor='w', fmt='.2f', vmin=0, vmax=1, center=0.5)
h.set_yticklabels(labels=h.get_yticklabels(), va='center')
```

Nakon što je utvrđen konačni model, pomoću funkcije *confusion_matrix* možemo izraditi matricu konfuzije za navedeni model te istu vizualizirati pomoću *seaborn* biblioteke.

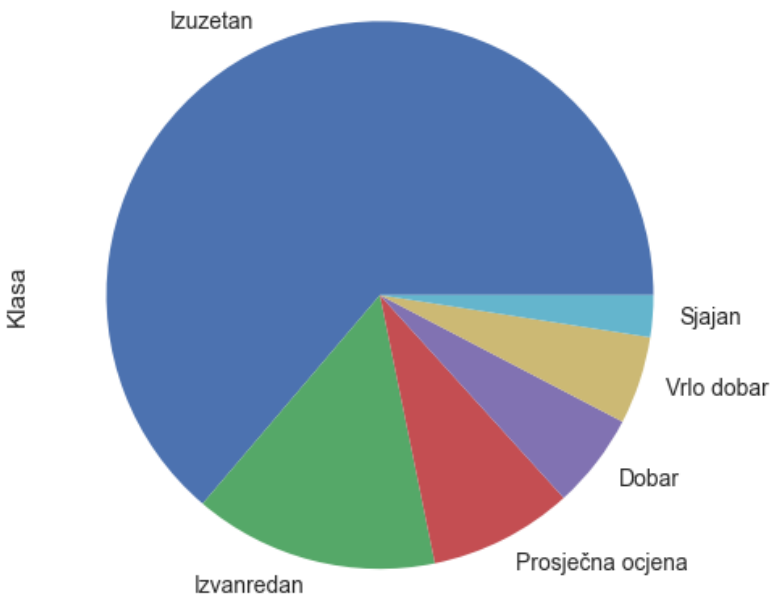


Slika 67. Matrica konfuzije - engleske recenzije

Na slici 67. nalazi se matrica konfuzije multinomial Naive Bayes modela za recenzije na engleskom jeziku. Na y osi nalaze se stvarne klase dok se na x osi nalaze klase predviđene od strane modela. Polje u presjeku predstavlja postotak presjeka određene dvije klase, te možemo vidjeti da, ako gledamo klasu *Izvanredan,* 1 % stvarnih klasa predviđeno je kao *Prosječna,* 6 % predviđeno je kao *Dobar,* 3 % predviđeno je kao *Vrlo Dobar,* 3 % predviđeno je kao *Sjajan* te je 35 % predviđeno kao *Izuzetan.* Model je točno predvidio 52 % instanci navedene klase. Ukratko, na dijagonali matrice nalaze se točno predviđeni postotci određene klase te možemo vidjeti da je model najbolje predvidio klasu *Izuzetan,* te da je najmanji postotak točno pogođenih opservacija predvidio unutar klase *Sjajan* (33 %).

Isti postupak provodi se za recenzije na hrvatskom jeziku kako bi se usporedili rezultati.

```
rec_hrv["Klasa"].value_counts(normalize = True).plot.pie()
```



Slika 68. Udio klasa unutar skupa podataka - hrvatske recenzije

Na slici 68. vidimo da su klase nebalansirane i kod skupa podataka s hrvatskim recenzijama te da je također potrebno upotrijebiti metodu SMOTE nad skupom podataka prije nego istreniramo model na navedenim podacima.

```
x = rec_hrv["Ukupan tekst"]
y = rec_hrv["Klasa"]
x_1 = []
for recenica in x:
    x_1.append(recenica)
cv=CountVectorizer(max_features=2000)
x_fin=cv.fit_transform(x_1).toarray()
y_fin = y.apply(lambda x: sentiment_red.index(x))
```

Definirane su odgovarajuće *x* i *y* varijable te je *x* varijabla vektorizirana pomoću primjene *Bag of words* pristupa i konvertirana u numpy polje. *Y* vrijednosti klase su kodirane u cijele brojeve koristeći se istim prethodno definiranim redoslijedom klasa.

```
x_train, x_test, y_train, y_test = train_test_split(x_fin, y_fin, test_
size=0.3, random_state = 1)
x_train.shape,y_train.shape
Out[157]: ((2699, 2000), (2699,))
smote=SMOTE()
x_sm, y_sm = smote.fit_resample(x_train, y_train)
x_sm.shape, y_sm.shape
Out[160]: ((10242, 2000), (10242,))
```

Zatim dijelimo podatke na podatke za treniranje i testiranje, provjeravamo dimenzije podataka, provodimo SMOTE metodu nad podacima i možemo vidjeti kako se broj opservacija uvećao za skoro 8 000 uzoraka.

```
counter = Counter(y_sm)
for k,v in counter.items():
  per = v / len(y_sm) * 100
  print('Klasa=%d, Broj opservacija=%d (%.3f%%)' % (k, v, per))
Klasa=4, Broj opservacija=1707 (16.667%)
Klasa=0, Broj opservacija=1707 (16.667%)
Klasa=5, Broj opservacija=1707 (16.667%)
Klasa=1, Broj opservacija=1707 (16.667%)
Klasa=3, Broj opservacija=1707 (16.667%)
Klasa=2, Broj opservacija=1707 (16.667%)
```

Također provjeravamo koliki je postotak svake klase unutar *y* podataka za treniranje i možemo vidjeti da su udjeli klasa jednaki (16.667 %).

```
model.fit(x_sm, y_sm)
y_pred=model.predict(x_test)
cf=classification_report(y_test, y_pred)
print(cf)
accuracy_score(y_test, y_pred)
```

Treniramo model na *x* i *y* podacima za treniranje te zatim koristimo isti model za predikciju vrijednosti opservacija unutar *x* skupa za testiranje. Kao i kod engleskih recenzija, stvaramo klasifikaciji izvještaj i mjerimo ukupnu preciznost modela.

```
              precision    recall  f1-score   support

           0       0.78      0.75      0.76        92
           1       0.27      0.33      0.30        55
           2       0.25      0.24      0.24        63
           3       0.81      0.76      0.79        34
           4       0.69      0.66      0.68       161
           5       0.90      0.91      0.91       753

    accuracy                           0.79      1158
   macro avg       0.62      0.61      0.61      1158
weighted avg       0.80      0.79      0.79      1158
```

```
accuracy_score(y_test, y_pred)
Out[166]: 0.7936096718480138
```

Možemo vidjeti da je model bolje predvidio klase recenzija na hrvatskom jeziku (s preciznošću od ~79 %).

```
parameters = {
    'alpha': [1, 0.95, 0.9, 0.85, 0.8, 0.75, 0.7]
}

grid = GridSearchCV(model, param_grid=parameters, cv=10, refit=True)
grid.fit(x_sm, y_sm)
print(grid.best_params_)
```

Pomoću *grid search*-a provjereno je nekoliko *alpha* vrijednosti kako bi provjerili hoće li uz navođenje drugog *alpha* parametra rezultati biti bolji.

```
parameters = {
    'alpha': [1, 0.95, 0.9, 0.85, 0.8, 0.75, 0.7]
}

grid = GridSearchCV(model, param_grid=parameters, cv=10, refit=True)
grid.fit(x_sm, y_sm)
print(grid.best_params_)

Out[196]: {'alpha': 0.95}
```

Iz priloženog rezultata možemo vidjeti da najbolji rezultat model ostvari sa *alpha* parametron od 0.95.

```
model=MultinomialNB(alpha = 0.95)
```

Ažuriranjem *alpha* parametra modela te potom treniranjem i spremanjem predikcija modela dobijemo sljedeći rezultat:

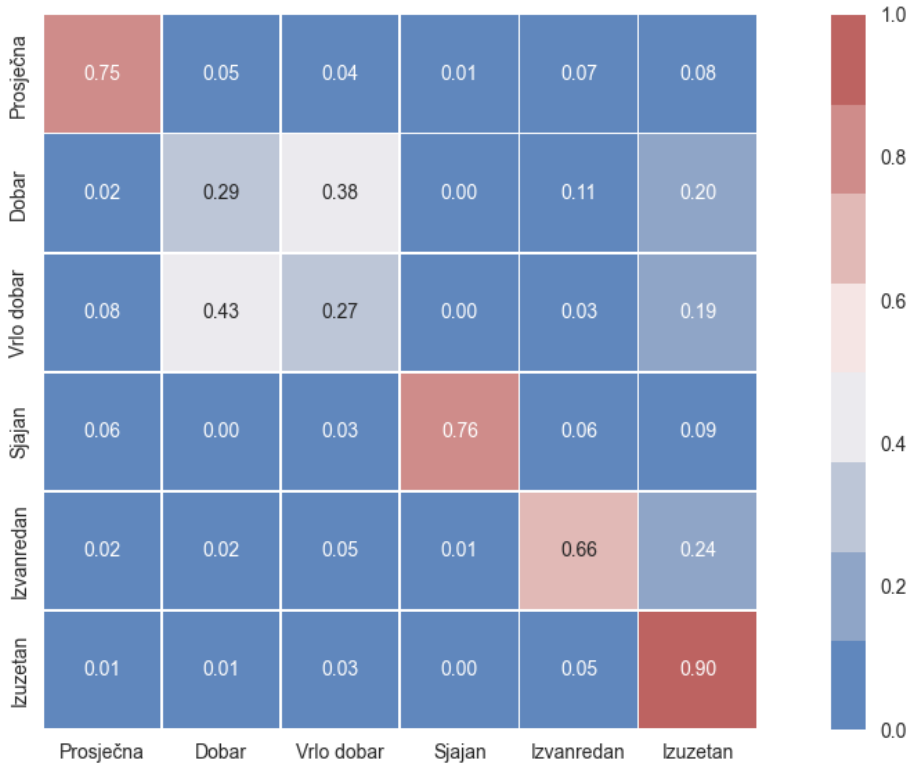
```
accuracy_score(y_test, y_pred)
Out[200]: 0.7944732297063903
```

te možemo vidjeti da se preciznost modela neznatno povećala (za 0.000861).

```
cm = confusion_matrix(y_test, y_pred, normalize = 'true')
cm = pd.DataFrame(cm)
cm = cm.rename(columns = { 0 : "Prosječna", 1: "Dobar", 2: "Vrlo dobar"
, 3: "Sjajan", 4: "Izvanredan", 5: "Izuzetan"})
```

```
h = sns.heatmap(cm, annot=True, cmap=colormap, linewidth=.5, square = T
rue, linecolor='w', fmt='.2f', vmin=0, vmax=1, center=0.5)
h.set_yticklabels(labels=h.get_yticklabels(), va='center')
```

Nakon provjere točnosti modela, pomoću matrice konfuzije vizualiziramo odnos stvarnih i predviđenih klasa.



Slika 69. Matrica konfuzije - hrvatske recenzije

Na slici 69. prikazana je matrica konfuzije multinomial Naive Bayes modela za recenzije na hrvatskom jeziku. Kada matricu usporedimo s matricom konfuzije recenzija na engleskom jeziku možemo vidjeti slijedeće:

- Preciznost predviđanja klase *Prosječna ocjena* veća je za ~ 20 % kod modela baziranog na recenzijama pisanim na hrvatskom jeziku (75 % u odnosu na prethodni model gdje je model točno predvidio 56 % opservacija)
- Preciznost predviđanja klase *Dobar* manja je za ~ 7 % kod modela baziranog na hrvatskom jeziku (29 % u odnosu na prethodni model gdje je model točno predvidio 36 % opservacija)

- Preciznost predviđanja klase *Vrlo dobar* manja je za ~ 19 % kod modela baziranog na recenzijama hrvatskog jezika (27 % u odnosu na prethodni model gdje je model točno predvidio 46 % opservacija)
- Preciznost predviđanja klase *Sjajan* veća je za 43 % kod modela baziranog na recenzijama hrvatskog jezika (76 % u odnosu na prethodni model gdje je model točno predvidio 33 % opservacija)
- Preciznost predviđanja klase *Izvanredan* veća je za ~ 14% kod modela baziranog na recenzijama hrvatskog jezika (66 % u odnosu na prethodni model gdje je model točno predvidio 52 % opservacija)
- Preciznost predviđanja klase *Izuzetan* veća je za ~ 5% kod modela baziranog na recenzijama hrvatskog jezika (90 % u odnosu na prethodni model gdje je model točno predvidio 85 % opservacija)

6. Primjena algoritama strojnog učenja za klasifikaciju objekata na temelju općih informacija o objektima

U ovom poglavlju opisane su metode procesiranja podataka prije modeliranja te sami rezultati dobiveni primjenom algoritama strojnog učenja nad podacima s bookinga.

Korelacije varijabli su izračunate u poglavlju 4.2. te za treniranje modela biramo varijable koje imaju korelaciju s ciljnom varijablom većom od 0.15.

```
df = df[abs(df["Kategorija ocjene objekta"]) > 0.15]
```

Kako pripadajuće korelacijske koeficijente imamo spremljene u skupu podataka *df,* filtriramo navedeni skup podataka tako da tražimo apsolutnu vrijednost koeficijenta korelacije veću od 0.15.

	index	Kategorija ocjene objekta	positive
4	Kategorija ocjene objekta	1.000000	True
3	Recenzija objekta	0.958823	True
11	Recenzija udobnosti	0.894639	True
12	Recenzija vrijednosti	0.865484	True
9	Recenzija sadržaja	0.864115	True
10	Recenzija čistoće	0.826946	True
8	Recenzija osoblja	0.822414	True
13	Recenzija lokacije	0.654055	True

```
14              Recenzija wifi                0.383175      True
6             Veličina objekta                0.164806      True
2       Kategorizacija objekta                0.163881      True
21                    Doručak                -0.162994     False
22                        Bar                -0.180763     False
15                  Za pušače                -0.189662     False
17                     Zabave                -0.195038     False
```

Kao rezultat dobijemo navedenih 15 varijabli te ćemo navedene varijable iskoristiti za treniranje modela.

```
varijable = list(df["index"])
```

U navedenu varijablu spremamo listu imena varijabli sa korelacijom većom od 0.15.

```
booking_v = booking.drop(booking.columns.difference(varijable), 1)
booking_v.drop("Recenzija objekta", 1, inplace = True)
```

Zatim stvaramo novi skup podataka iz kojeg izbacujemo sve varijable koje se ne nalaze unutar prethodno izrađene liste varijabli. Također, iz skupa podataka izbacujemo varijablu *Recenzija objekta* s obzirom na to da je klasa objekta proizašla iz same recenzije.

```
x = booking_v[booking_v.columns.difference(['Kategorija ocjene objekta'
])]
y = booking_v["Kategorija ocjene objekta"]
```

Potom stvaramo *x* i *y* varijable koje sadrže odvojeno podatke i same klase objekata. Varijabla *x* ima vrstu smještaja u tekstualnom obliku te ćemo istu najprije kodirati u cijele brojeve.

```
from sklearn import preprocessing
encoder = preprocessing.LabelEncoder()
x['Vrsta objekta'] = encoder.fit_transform(x['Vrsta objekta'])

sentiment_red = ['Prosječna ocjena ', 'Dobar ', 'Vrlo dobar ', 'Sjajan
', 'Izvanredan ', 'Izuzetan ']
y = y.apply(lambda x: sentiment_red.index(x))
```

Također, potrebno je kodirati i *y* nazive klasa, s obzirom na to da će biti potrebno primijeniti SMOTE metodu kako bi imali balansiran broj klasa (preduvjet za korištenje SMOTE metode jest da su svi podaci numeričkog tipa).

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3
, random_state = 1)

x_train.shape,y_train.shape
smote=SMOTE()
x_sm, y_sm = smote.fit_resample(x_train, y_train)
x_sm.shape, y_sm.shape
```

Dijelimo podatke na podatke za treniranje i testiranje te primjenjujemo SMOTE metodu nad dobivenim skupovima podataka za treniranje.

```
x_train.shape,y_train.shape
Out[28]: ((4293, 14), (4293,))

x_sm.shape, y_sm.shape
Out[29]: ((9654, 14), (9654,))
```

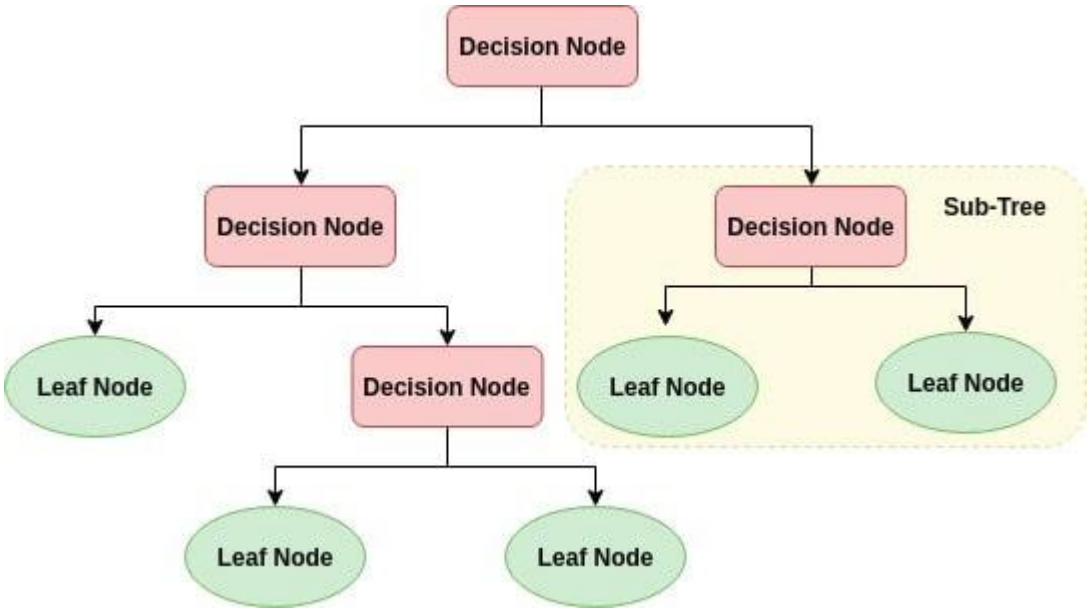
Možemo vidjeti da se broj opservacija povećao za ~ 5000 redova.

```
y_sm.value_counts()
Out[30]:
2    1609
3    1609
0    1609
4    1609
5    1609
1    1609
Name: Kategorija ocjene objekta, dtype: int64
```

Kada pogledamo udio kategorija u skupu za treniranje, možemo vidjeti da postoji točno 1609 opservacija svake klase unutar oznaka za treniranje.

6.1. Stablo odluke

Stablo odluke predstavlja algoritam nadziranog učenja koji se može koristiti za klasifikacijske, ali i za regresijske probleme. Kako samo ime govori, sadrži strukturu stabla gdje interni čvor predstavlja određeno svojstvo (ili atribut), grane predstavljaju pravilo odluke i svaki list predstavlja vrijednost izlazne varijable. Glavni čvor iz kojeg započinje grananje stabla naziva se korijenskim čvorom. Svaki skup unutar stabla dijeli se na dva ili više homogenih podskupova na temelju najznačajnijeg diferencijatora u ulaznim varijablama određenog čvora. Stablo odluke koristi rekurzivno particioniranje, metodu učenja u kojoj podjela (učenje) kreće od vrha stabla prema dolje (engl. *top-down*).



Slika 70. Struktura stabla odluke, preuzeto sa:
https://res.cloudinary.com/dyd911kmh/image/upload/f_auto,q_auto:best/v1545934190/1_r5ikdb.png

Na slici 70. prikazana je struktura stabla odluke. Algoritam radi na sljedeći princip:

1. Obavlja se selekcija najboljeg atributa koristeći ASM (engl. *Attribute Selection Measures*) za dijeljenje podataka – mjera za selekciju atributa predstavlja pravila za dijeljenje podataka kojom se pridružuje rank svakom atributu te se atribut s najboljim rezultatom izabire za dijeljenje skupa podataka. Najčešće korištene mjere za selekciju atributa jesu entropija, Gini indeks i omjer dobiti kojima se mjeri smanjivanje varijabilnosti distribucije ciljne varijable u granama ispod (*child nodes*) u usporedbi s granom na kojoj se radi podjela (*parent node*).

2. Odabran atribut postaje čvor (engl. *Decision node*) i dijeli skup podataka u manje podskupove.

3. Započinje izgradnja stabla rekurzivnimponavljanjem drugog koraka za svaki nastali čvor (engl. *child node*) dok sve n-torke podjele ne pripadaju istom atributu, ne postoji više atributa ili ako ne postoji više instanci (opservacija) [11].

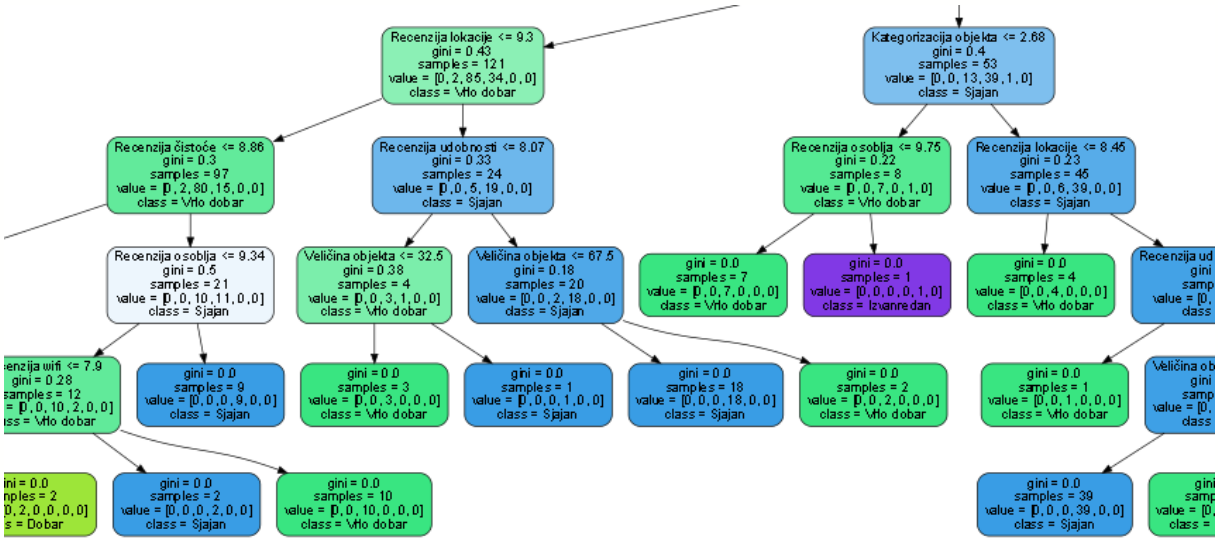
```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier()
tree = tree.fit(x_sm,y_sm)
y_pred = tree.predict(x_test)


accuracy_score(y_test, y_pred)
Out[37]: 0.7961956521739131
```

Točnost modela (bez podešavanja parametara) iznosi ~ 80%.

Pokušajem korištenja *GridSerach* objekta za istraživanje najboljih parametara bilo je neuspješno, s obizorm na to da je program radio preko 24 sata i nije završio, te je odlučeno ručno testirati neke od parametara kako bi optimizirali model. No prije toga, gledamo kako izgleda navedeno stablo i njegovu veličinu.

```
from sklearn.tree import export_graphviz
import pydotplus
from IPython.display import Image
import graphviz

dot_data = StringIO()
export_graphviz(tree, out_file=dot_data, feature_names = x.columns,
                class_names = ["Prosječna", "Dobar", "Vrlo dobar", "Sja
                jan", "Izvanredan", "Izuzetan"],
                rounded = True, proportion = False,
                precision = 2, filled = True)

graf = pydotplus.graph_from_dot_data(dot_data.getvalue())
graf.set_graphviz_executables( {'dot': r'C:\Program Files\Graphviz\bin\
dot.exe'})

graf.write_png('tree.png')
```

Za vizualizaciju stabla potrebna nam je *graphviz* biblioteka gdje navedeno stablo eksportamo u *dot* datoteku, te pomoću *pydotplus* biblioteke izvlačimo grafikon iz *dot* datoteke. Kako bi navedena operacija uspjela, potrebno je instalirati *graphviz* (dostupno je na poveznici: https://graphviz.org/download/) i u ovom slučaju, bilo je potrebno eksplicitno navođenje putanje do instalirane datoteke.



Slika 71. Dio stabla odluke

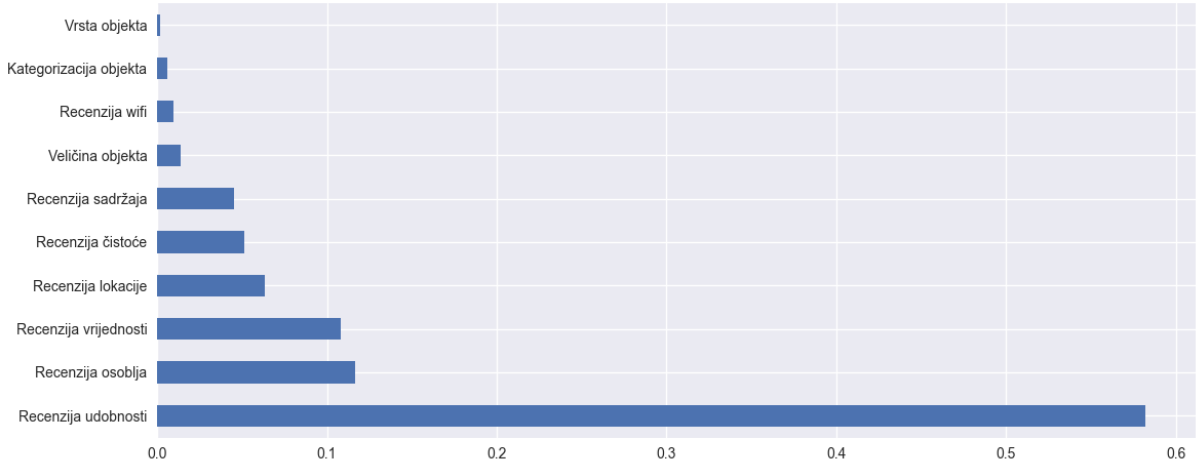
Na slici 71. prikazan je samo manji dio stabla odluke s obzirom na to da originalno stablo ima 655 čvorova i 328 listova i preveliko je da bi se jasno prikazalo na jednoj slici.

```
tree_obj = tree.tree_
tree_obj.node_count
tree_obj.n_leaves
```

Veličina stabla je provjerena tako da je klasifikator definiran kao objekt, nad kojim možemo zatim provjeriti koliko listova i čvorova ima.

```
feat_importances = pd.Series(tree.feature_importances_, index=x.columns
)
feat_importances.nlargest(10).plot(kind='barh')
```

Iz klasifikatora možemo također izlučiti mjere važnosti svake varijable.



Slika 72. Mjere važnosti varijabli

Na slici 72. prikazane su mjere važnosti 10 najvažnijih varijabli unutar modela. Možemo vidjeti da daleko najveću važnost ima varijabla recenzije udobnosti, te odmah nakon nje ostale recenzije korisnika (izuzev recenzije za WiFI). Odlučeno je isprobati izbaciti varijable manje važnosti te također promijeniti defaulte vrijednosti nekih parametara kako bi dobili veću točnost modela.

```
tree = DecisionTreeClassifier(random_state = 1, splitter = "best", max_
depth = 10, min_samples_split = 2)
tree = tree.fit(x_sm,y_sm)
y_pred = tree.predict(x_test)
```

Promijenjen je jedan parametar (ostali su defaulti) a to je *max_depth* koji označava maksimalnu dubinu stabla (default parametar je *None*). Također, pridodan je *random_state* parametar za potrebu replikacije rezultata.

```
accuracy_score(y_test, y_pred)
Out[52]: 0.8282608695652174
```

Točnost modela s izbacivanjem određenih varijabli i postavljanjem navedenih parametara iznosi ~ 83 % što je povećanje točnosti od skoro 4 % u odnosu na prošlu inačicu modela.

```
accuracy_score(y_test, y_pred)
Out[52]: 0.8233695652173914
```

Bez izbacivanja određenih varijabli i promjenom parametara dobijemo točnost od 82 % što znači da je, kada gledamo točnost modela, bolje izbaciti varijable manje važnosti ako želimo povećati preciznosti modela.

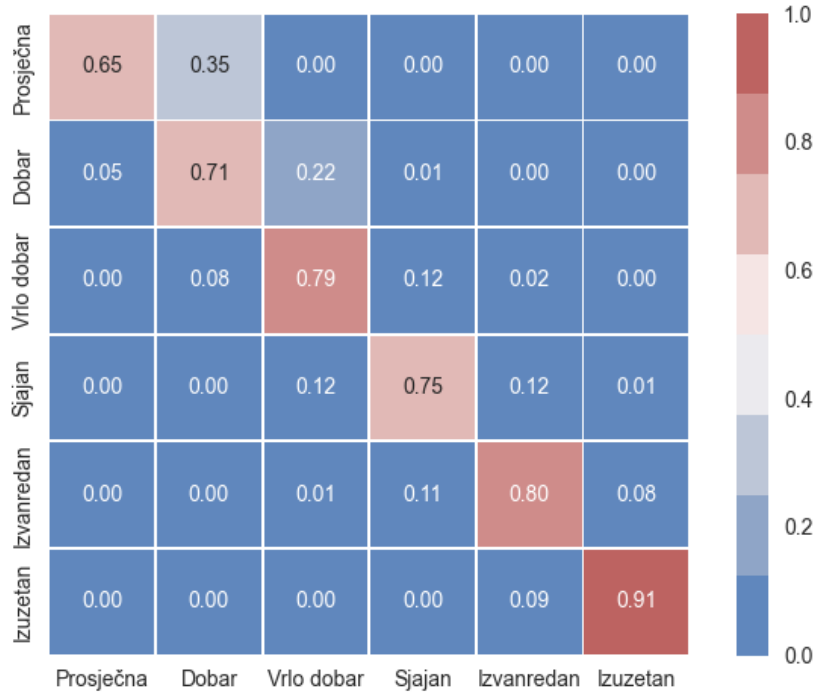
```
tree_obj.node_count
Out[53]: 1577

tree_obj.n_leaves
Out[54]: 789
```

Novo stablo ima 1 577 čvorova te 789 listova.

```
from sklearn.metrics import classification_report
cf=classification_report(y_test, y_pred)
print(cf)
              precision    recall  f1-score   support

           0       0.67      0.70      0.68        20
           1       0.70      0.75      0.72        73
           2       0.78      0.70      0.74       171
           3       0.65      0.71      0.68       245
           4       0.79      0.78      0.79       637
           5       0.89      0.88      0.88       694

    accuracy                           0.80      1840
   macro avg       0.75      0.75      0.75      1840
weighted avg       0.80      0.80      0.80      1840
```

```
cm = confusion_matrix(y_test, y_pred, normalize = 'true')
cm = pd.DataFrame(cm)
cm = cm.rename(index = { 0 : "Prosječna", 1: "Dobar", 2: "Vrlo dobar",
3: "Sjajan", 4: "Izvanredan", 5: "Izuzetan"})
cm = cm.rename(columns = { 0 : "Prosječna", 1: "Dobar", 2: "Vrlo dobar"
, 3: "Sjajan", 4: "Izvanredan", 5: "Izuzetan"})
h = sns.heatmap(cm, annot=True, cmap=colormap, linewidth=.5, square = T
rue, linecolor='w', fmt='.2f', vmin=0, vmax=1, center=0.5)
h.set_yticklabels(labels=h.get_yticklabels(), va='center')
```

Nakon tuniranja parametara modela, prikazan je klasifikacijski izvještaj i izrađena je matrica konfuzije kako bi mogli proučiti rezultate predikcije modela u odnosu na pojedine klase.

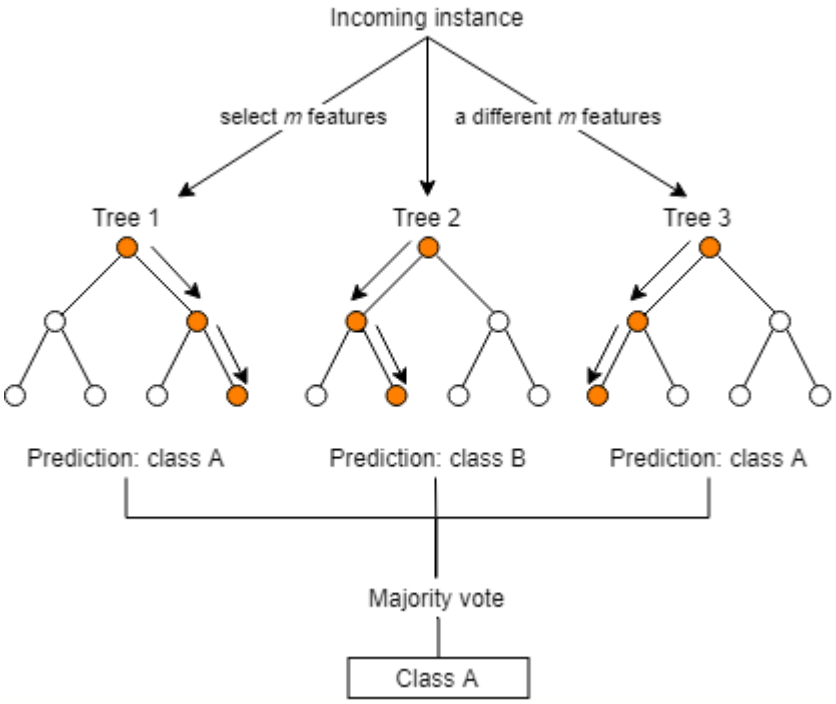


Slika 73. Matrica konfuzije - stablo odluke

Na slici 73. prikazana je matrica konfuzije za model stabla odluke. Možemo vidjeti točno predviđenih 75 % opservacija i više unutar klasa *Izuzetan, Izvanredan, Sjajan i Vrlo dobar.* Kod klase *Prosječna ocjena i Dobar* nešto je manji postotak točno predviđenih opservacija (65 % i 71 %) no možemo vidjeti da se model najviše kod navedenih klasa dvoumi s njihovim „susjedima“ (*Prosječna* klasa i klasa *Dob*ar te klasa *Dobar* i klasa *Vrlo Dobar*) te ne postoji slučaj gdje je model objekt sa stvarnom recenzijom od 5.6 predvidio kao 9.5 i obrnuto.

6.2. Slučajna šuma

Algoritam slučajne šume također je vrsta algoritma za nadzirano strojno učenje koji se može koristiti za klasifikacijske i regresijske probleme. Slučajna šuma predstavlja ansambl stabala odluke, što znači da se sastoji od većeg broja manjih stabala odluke (u ovom kontekstu zvanim procjeniteljima) od kojih svako donosi svoju predikciju. Model slučajne šume kombinira predikcije procjenitelja što dovodi do ukupno preciznije predikcije podataka.



Slika 74. Struktura slučajne šume, preuzeto sa: https://images.deepai.org/user-content/9196004107-thumb-1447.svg

Na slici 74. prikazana je struktura slučajne šume. Algoritam funkcionira tako da prosljeđuje određen broj varijabli (atributa) svakom od zasebnih stabala odluke (od kojih svako stablo prima različite varijable). Stabla odluke generiraju se metodom *Bootstrap* na istom skupu za treniranje. U grananju individualnog stabla koristi se samo jedan od prediktora stabla te se pri svakom grananju stabla uzima novi uzorak od m prediktora (obično je $m = sqrt\ (p)$ gdje je p ukupan broj prediktora). Početkom treniranja bira se n slučajnih podskupova iz skupa za učenje (jedan nasumični podskup za svako stablo odluke) prilikom čega se optimalna grananja za svako stablo temelje na odabranom slučajnom podskupu. Prilikom testiranja, svako stablo neovisno radi predikcije skupa za testiranje te se izvodi konačno predviđanje

šume – za svakog kandidata u testu, slučajna šuma koristi klasu s najviše glasova kao svoje glavno predviđanje [12].

```
from sklearn.ensemble import RandomForestClassifier
suma = RandomForestClassifier()
suma.fit(x_sm,y_sm)
y_pred=suma.predict(x_test)
```

Nakon učitavanja potrebne biblioteke, kreiramo model slučajne šume koji zatim treniramo nad skupovima za treniranje. Konačno, stvaramo varijablu s predikcijama kako bi mogli provjeriti točnost modela.

```
accuracy_score(y_test, y_pred)
```

```
Out[83]: 0.8782608695652174
```

Već u početku model slučajne šume ima znatno veću točnost od samog stabla odluke (bez tuniranja parametara oko 10 %). Kao i kod stabla odluke, pokretanjem navedenog dijela koda:

```
parameters = {
    'criterion': ["gini", "entropy"],
    'splitter' : ['best', 'random'],
    'max_depth': [10, 20, 50, 100, 200, 500, 1000, None],
    'min_samples_split': [2, 6, 8, 10, 15, 20],
    'ccp_alpha': [0.2, 0.4, 0.6, 0.8, 1]

}

grid = GridSearchCV(tree, param_grid=parameters, cv=10, refit=True)
grid.fit(x_sm, y_sm)
```

isti se provodio preko 30 sati. S obzirom na to da provjera svih ovih parametara traje predugo, ručno je testirano nekoliko parametara kako bi optimizirali model.

```
suma = RandomForestClassifier(random_state = 1, n_estimators = 150, cri
terion = "entropy", max_depth = 20, max_samples = None, max_features =
3, bootstrap = True, oob_score = True, min_samples_split = 2)
suma.fit(x_sm,y_sm)
y_pred=suma.predict(x_test)
```

Parametri koji su promijenjeni:

n_estimators – broj stabla odluke unutar slučajne šume (default je 100).

criterion – funkcija za mjeru kvalitete razdvajanja (grananja). Default je Gini.
max_depth – maksimalna dubina stabla odluke. Default vrijednost je *None.*
max_features – broj prediktora koji se uzimaju u obzir prilikom razdvajanja podataka. Default je *auto* (drugi korijen iz broja prediktora).
oob_score – da li koristiti *out-of-bag* uzorke za procjenu rezultata generalizacije. OOB su instance koje se nisu koristile za treniranje (učenje) modela. Default je *False.*

```
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import cross_val_predict, cross_val_score,
StratifiedShuffleSplit
kf = RepeatedStratifiedKFold(n_splits=5, n_repeats=10, random_state=1)
scores = cross_val_score(suma, x_sm, y_sm, scoring='accuracy', cv=kf, n
_jobs=-1)
```

Proveden je *StratifiedKFold* nad navedenim modelom slučajne šume. Izabrana je *Stratified* vrsta validacije jer ona, za razliku od obične *k* unakrsne validacije, osigurava da se skup za treniranje sastoji od jednakog postotka svake klase, npr. ako imamo 100 uzoraka od kojih je 80 klase 0 i 20 klase 1, skup za treniranje sastoji se od 64 uzorka klase 0 (80 % od klase 0) i od 16 uzoraka klase 1 (80 % od klase 1). Kod izračuna točnosti modela, navodimo model koji testiramo, *x* i *y* podaci za treniranje, metodu izračuna rezultata, generator unakrsne validacije (ili cijeli broj), *n_jobs* (-1 znači korištenje svih procesora).

```
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
Accuracy: 0.870 (0.014)
```

Gledamo prosjek točnosti svih rezultata i standardnu devijaciju od prosjeka te vidimo da je prosječna točnost modela sa unakrsnom validacijom 87 % sa standardnom devijacijom od 0.014 što znači da u prosjeku rezultati međusobno variraju za 0.014.

```
CV = StratifiedShuffleSplit(n_splits=10,test_size=0.2, random_state=1)
grid_search = GridSearchCV(suma,param_grid=dict(),verbose=3,scoring='ac
curacy', cv = CV).fit(x_sm, y_sm)
Fitting 10 folds for each of 1 candidates, totalling 10 fits
[CV 1/10] END ......................., score=0.948 total time=   4.5s
[CV 2/10] END ......................., score=0.944 total time=   3.5s
[CV 3/10] END ......................., score=0.943 total time=   3.5s
[CV 4/10] END ......................., score=0.947 total time=   3.6s
[CV 5/10] END ......................., score=0.943 total time=   3.6s
[CV 6/10] END ......................., score=0.940 total time=   3.5s
```

```
[CV 7/10] END ....................., score=0.942 total time=   3.5s
[CV 8/10] END ....................., score=0.945 total time=   3.5s
[CV 9/10] END ....................., score=0.949 total time=   3.5s
[CV 10/10] END ....................., score=0.945 total time=   3.5s
```

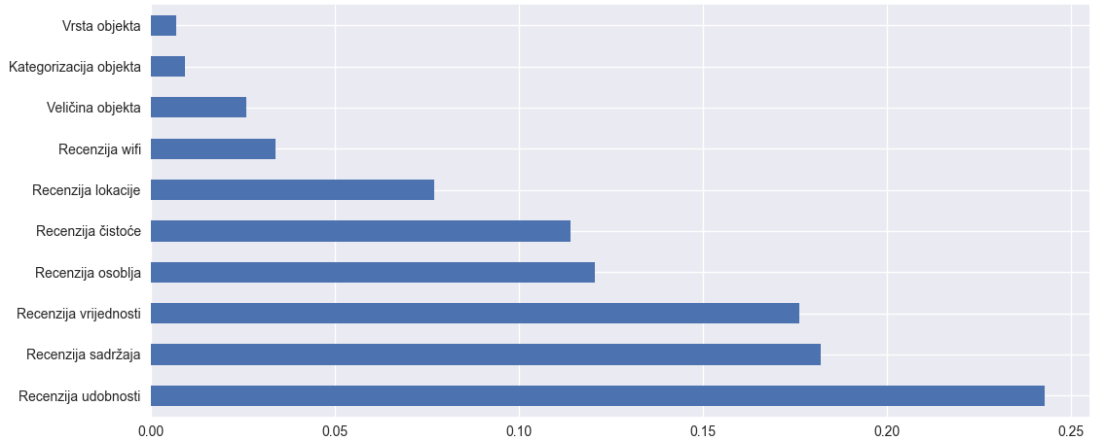
Provodimo *GridSearch* za model slučajne šume bez navođenja dodatnih parametara za testiranje u kojem navodimo metodu mjerenja preciznosti modela i generator unakrsne validacije (ili cijeli broj). Stvoreni grid treniramo nad podacima za treniranje.

```
print ('on train set')
on train set
scores = cross_val_score(grid_search.best_estimator_, x_train, y_train,
cv=3, scoring='accuracy')
print (scores.mean(),scores)
0.8725832750989984 [0.88609364 0.87840671 0.85324948]

print ('on test set')
on test set
scores = cross_val_score(grid_search.best_estimator_, x_test, y_test, c
v=3,scoring='accuracy')
print (scores.mean(),scores)
0.8750002214062663 [0.87459283 0.8776509  0.87275693]
```

Gledamo prosječnu točnost modela slučajne šume nad podacima za treniranje i testiranje. Vidimo da je prosječna točnost nad podacima za treniranje 87,25 %, dok je prosječna točnost modela nad podacima za testiranje 87,5 %.

```
feat_importances = pd.Series(suma.feature_importances_, index=x.columns
)
feat_importances.nlargest(10).plot(kind='barh')
```

Kao i kod stabla odluke, gledamo mjeru važnosti varijabli za navedeni model.

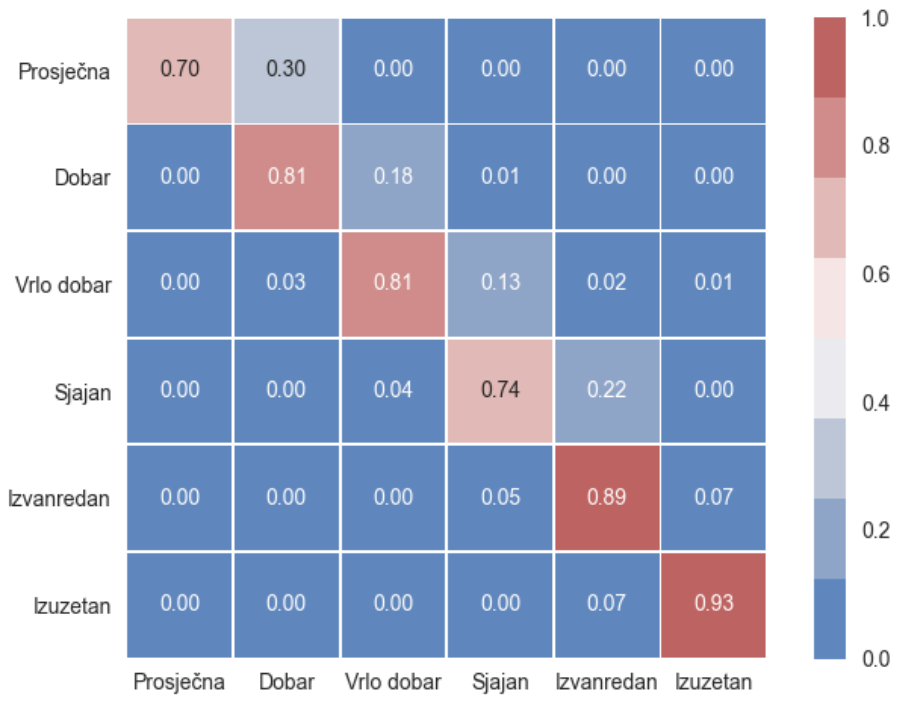


Slika 75. Mjere važnosti varijabli za model slučajne šume

Vidimo da su varijacije između mjere važnosti navedenih varijabli dosta manje nego kod modela stabla odluke gdje smo imali recenziju udobnosti s mjerom važnosti od 0.6 + dok su ostale varijable imale mjeru važnosti oko 0.1 i niže. Izbacivanjem dvije varijable s najmanjom mjerom važnosti dobijemo točnost modela od 87,28 %.

```
df1 = confusion_matrix(y_test, y_pred, normalize = 'true')
df1 = pd.DataFrame(df1)
df1 = df1.rename(index = { 0 : "Prosječna", 1: "Dobar", 2: "Vrlo dobar"
, 3: "Sjajan", 4: "Izvanredan", 5: "Izuzetan"})
df1 = df1.rename(columns = { 0 : "Prosječna", 1: "Dobar", 2: "Vrlo doba
r", 3: "Sjajan", 4: "Izvanredan", 5: "Izuzetan"})
h = sns.heatmap(df1, annot=True, cmap=colormap, linewidth=.5, square =
True, linecolor='w', fmt='.2f', vmin=0, vmax=1, center=0.5)
h.set_yticklabels(h.get_yticklabels(), rotation=0)
```

Nakon istraživanja mjera važnosti varijabli i tuniranja parametara modela, izrađujemo matricu konfuzije kako bi primijetili postotak točno pogođenih opservacija unutar svake klase.

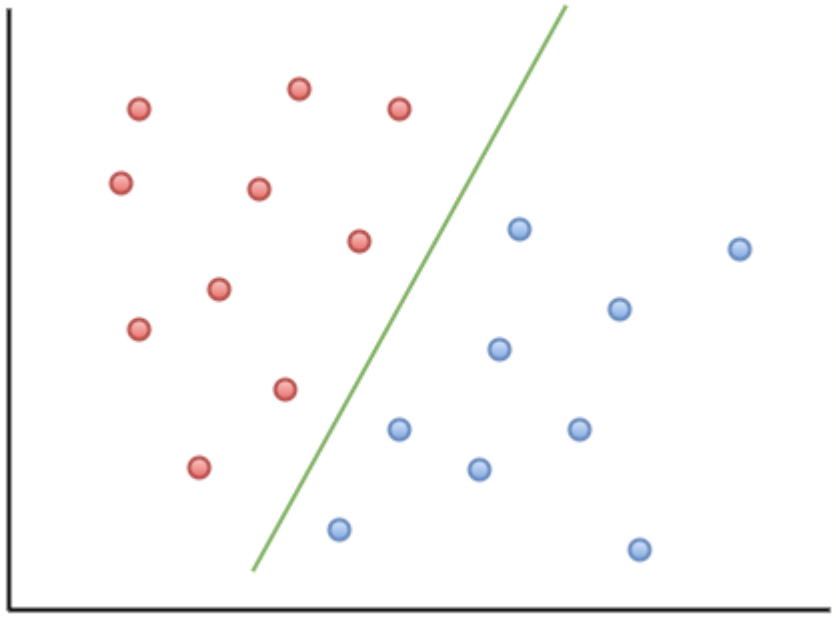


Slika 76, Matrica konfuzije slučajna šuma

Na slici 76. nalazi se matrica konfuzije za model slučajne šume. Kada usporedimo matricu konfuzije modela slučajne šume s matricom stabla odluke možemo vidjeti povećanje postotka točno pogođenih opservacija unutar svake klase osim klase *Sjajan* gdje je model slučajne šume pogodio 1 % manje opservacija nego model stabla odluke. No, gledajući širu sliku, možemo primijetiti velik porast u točno pogođenim opservacijama unutar klasa: *prosječna* – povećanje broja točno pogođenih opservacija za 5 %, *dobar* – povećanje broja točno pogođenih opservacija za 10 %, *vrlo dobar* – povećanje broja točno pogođenih opservacija za 3 %, *izvanredan* – povećanje broja točno pogođenih opservacija za 9 % te *izuzetan* – povećanje broja točno pogođenih opservacija za 2 %.

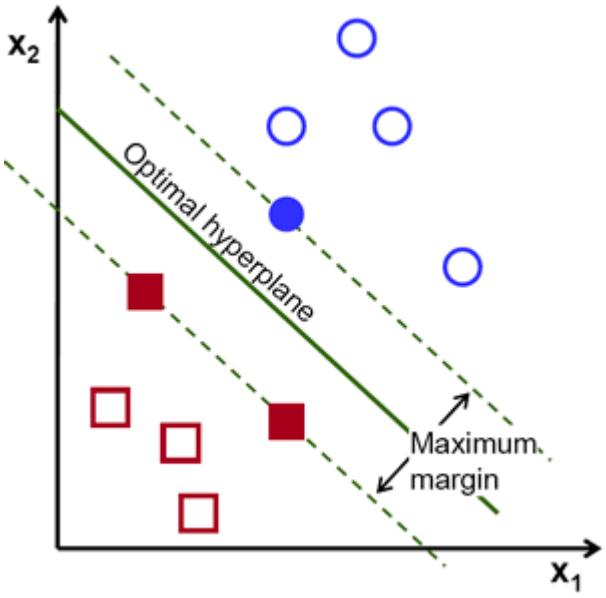
6.3. Metoda potpornih vektora

Metoda potpornih vektora (engl. *Support vector machines*) je algoritam nadziranog strojnog učenja koji analizira podatke i prepoznaje uzorke u istima. Osnovna ideja navedenog algoritma je rezanje (podjela) skupa podataka kroz najveći razmak (engl. *gap*) između kategorija. Kao i slučajne šume, navedeni algoritam može se koristiti za regresijske i klasifikacijske probleme.



Slika 77. Intuicija iza SVM algoritma, preuzeto sa: https://www.freecodecamp.org/news/content/images/2020/06/image-57.png

SVM algoritam koristi geometriju za predviđanja – mapira podatkovne točke i dijeli podatke tako da generira najveći mogući razmak između kategorija podataka. Novi podaci su klasificirani ovisno o tome kojoj strani linije pripadaju. Primjer možemo vidjeti na slici 77. gdje, ako je novi podatak smješten s lijeve strane zelene linije, tada se pretpostavlja da pripada crvenoj kategoriji (uzimajući u obzir da su crvena i plava klase kategorija). Linija podjele (zelena linija) u kontekstu ovog algoritma naziva se *hyperplane*.



Slika 78. Vizualizacija načina rada SVM algoritma, preuzeto sa: https://www.freecodecamp.org/news/content/images/2020/06/image-58.png

Optimalna *hyperplane* linija predstavlja liniju koja maksimizira veličinu margine između dvije najbliže točke iz svake kategorije (slika 78). Na slici također vidimo da tri točke dodiruju linije margine (iscrtane linije na slici) – dvije točke crvene kategorije i jedna točka plave kategorije. Takve točke, koje dodiruju linije margine, nazivaju se *support vectors* po kojima je algoritam dobio ime [13].

```
from sklearn import svm

SVM = svm.SVC()
SVM.fit(x_sm, y_sm)
y_pred = SVM.predict(x_test)
```

Kao i kod ostalih modela, stvaramo model upotrebom odgovarajuće funkcije (u ovom slučaju to je SVC), treniramo model nad podacima za treniranje i stvaramo predikcije podataka za testiranje.

```
accuracy_score(y_test, y_pred)
Out[14]: 0.7315217391304348
```

Točnost modela bez optimizacije parametara iznosi 73 %.

```
SVM = svm.SVC(kernel = "linear", random_state = 1, verbose = 3, decisio
n_function_shape = 'ovr', break_ties = True)
SVM.fit(x_sm, y_sm)
y_pred = SVM.predict(x_test)
```

Parametri koji su izmijenjeni:

kernel – vrsta kernela koji će se koristiti u algoritmu. Default je *rbf,* te od dostupnih (*poly, linear, sigmoid, rbf, precomputed*) najučinkovitijim se pokazao linearni kernel.

break_ties – ako je *true* i *decision_function* je postavljen na *ovr* (*one vs rest* funkcija), predikcijska funkcija će neriješene slučajeve odlučiti na temelju vrijednosti pouzdanosti (engl. *confidence values*) funkcije odlučivanja.

Također, ako funkciju odlučivanja postavimo na *ovo* (*one vs one*) bez *break_ties* parametra, dobijemo identičnu točnost modela koja iznosi:

```
accuracy_score(y_test, y_pred)
Out[25]: 0.8853260869565217
```

Optimizacijom parametara točnost modela se povećala za čak 15%.

```
CV = StratifiedShuffleSplit(n_splits=10,test_size=0.2, random_state=1)
grid_search = GridSearchCV(SVM,param_grid=dict(), verbose=3, scoring='a
ccuracy', cv = CV).fit(x_sm, y_sm)
Fitting 10 folds for each of 1 candidates, totalling 10 fits
[LibSVM][CV 1/10] END ................., score=0.910 total time=   3.5s
[LibSVM][CV 2/10] END ................., score=0.907 total time=   2.9s
[LibSVM][CV 3/10] END ................., score=0.916 total time=   3.4s
[LibSVM][CV 4/10] END ................., score=0.913 total time=   3.8s
[LibSVM][CV 5/10] END ................., score=0.915 total time=   3.2s
[LibSVM][CV 6/10] END ................., score=0.904 total time=   3.2s
[LibSVM][CV 7/10] END ................., score=0.912 total time=   3.2s
[LibSVM][CV 8/10] END ................., score=0.909 total time=   3.2s
[LibSVM][CV 9/10] END ................., score=0.921 total time=   2.5s
[LibSVM][CV 10/10] END ................., score=0.911 total time=   3.1s
[LibSVM]
```

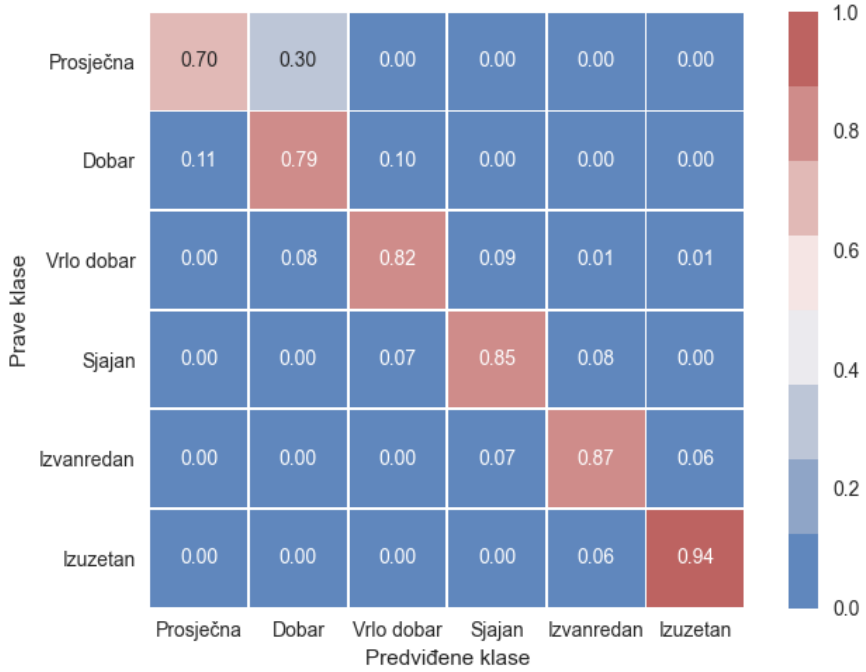
Provođenjem *GridSearchCV* funkcije možemo vidjeti da se točnost nad podacima za treniranje nalazi u rasponu od 90 – 92 %.

```
kf = RepeatedStratifiedKFold(n_splits=5, n_repeats=10, random_state=1)
scores = cross_val_score(SVM, x_test, y_test, scoring='accuracy', cv=kf
, n_jobs=-1)

print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
Accuracy: 0.897 (0.015)
```

Provodimo unakrsnu validaciju nad skupovima za testiranje i dobijemo da je prosječna točnost modela ~ 90 % uz standardnu devijaciju od 0.015 od prosječne točnosti modela.

```
df1 = confusion_matrix(y_test, y_pred, normalize = 'true')
df1 = pd.DataFrame(df1)
df1 = df1.rename(index = { 0 : "Prosječna", 1: "Dobar", 2: "Vrlo dobar"
, 3: "Sjajan", 4: "Izvanredan", 5: "Izuzetan"})
df1 = df1.rename(columns = { 0 : "Prosječna", 1: "Dobar", 2: "Vrlo doba
r", 3: "Sjajan", 4: "Izvanredan", 5: "Izuzetan"})
h = sns.heatmap(df1, annot=True, cmap=colormap, linewidth=.5, square =
True, linecolor='w', fmt='.2f', vmin=0, vmax=1, center=0.5)
h.set_yticklabels(h.get_yticklabels(), rotation=0)
h.set_xlabel("Predviđene klase")
h.set_ylabel("Prave klase")
```

Nakon testiranja modela, izrađena je matrica konfuzije kako bi proučili točno pogođene opservacije unutar svake klase.



Slika 79. Matrica konfuzije – SVM

Na slici 79. prikazana je matrica konfuzije za SVM model.

Prosječna – omjer pogođenih opservacija unutar klase jednak je onome modela slučajne šume, gdje je također točno predviđeno 70 % opservacija te je 30 % predviđeno kao *Dobar*.

Dobar – postotak točno pogođenih opservacija nešto je manji nego kod modela slučajne šume (79 % u odnosu na 81 %), te je kod SVM modela 10 % predviđeno kao *Vrlo dobar* i 11 % kao *Prosječna* dok je kod modela slučajne šume ostatak opservacija predviđen kao *Vrlo dobar* i 0 % kao *Prosječna*.

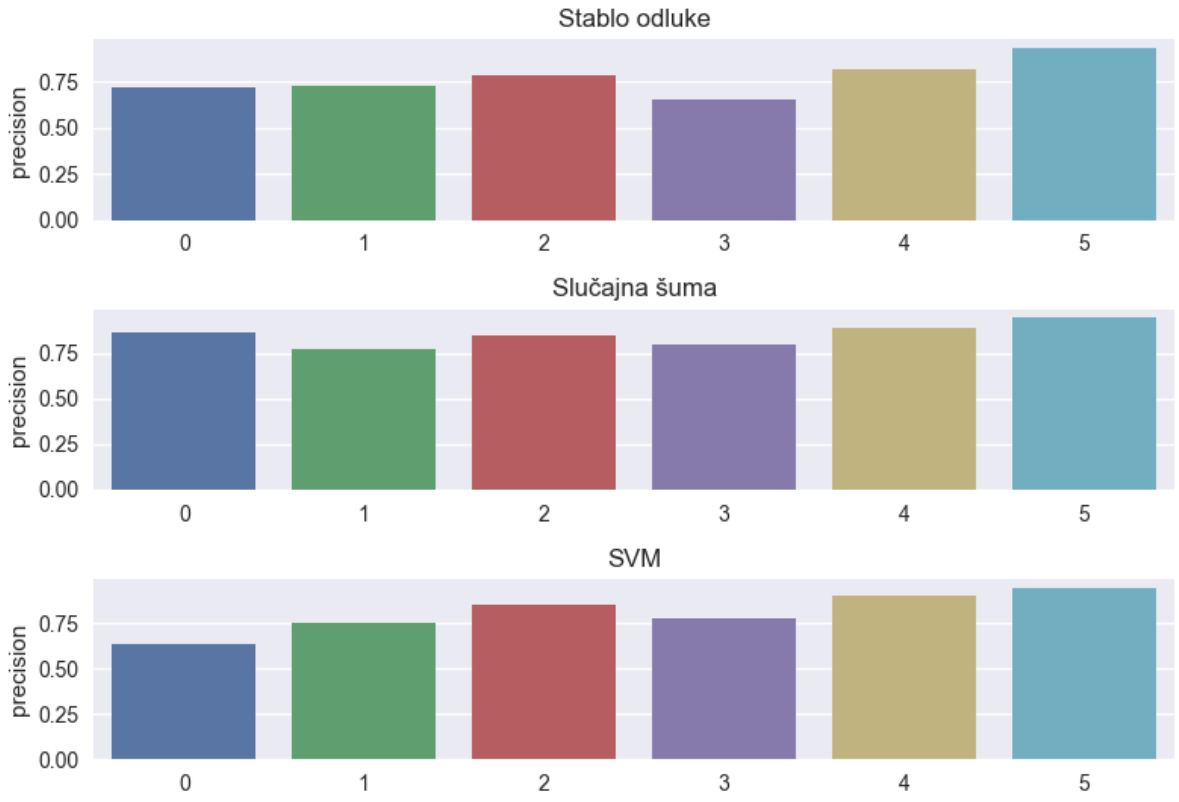
Vrlo dobar – postotak točno pogođenih opservacija kod SVM modela je za 1 % veći nego kod modela slučajne šume (82 % u odnosu na 81 %) te je kod SVM modela 8 % previđeno kao *Dobar* i 9 % kao *Sjajan* dok je kod modela slučajne šume većina ostatka opservacija (13 %) predviđeno kao *Sjajan* dok je tek mali dio (3 %) predviđeno kao *Dobar*.

Sjajan – postotak točno pogođenih opservacija kod SVM modela je za 11 % veći nego kod modela slučajne šume (85 % u odnosu na 74 %) te je kod SVM modela podjednak dio ostatka opservacija klasificirano kao *Vrlo dobar i Izvanredan* (7 % i 8 %) dok je kod modela slučajne šume 22 % preostalih opservacija klasificirano kao *Izvanredan* i 4 % kao *Vrlo dobar*.

Izvanredan – postotak točno pogođenih opservacija kod SVM modela je čak 2 % manji nego kod modela slučajne šume (87 % naspram 89 %) te je kod SVM modela 7 % opservacija predviđeno kao *Sjajan* i 6 % kao *Izuzetan* dok je kod modela slučajne šume 5 % predviđeno kao *Sajan* i 7 % kao *Izuzetan*.

Izuzetan – postotak točno pogođenih opservacija kod SVM modela je 1 % veći nego kod modela slučajne šume (94 % naspram 93 %) te je kod SVM modela ostalih 6 % predviđeno kao *Izvanredan* kao i 7 % opservacija kod modela slučajne šume.

6.4. Usporedba preciznosti i opoziva



Slika 80. Usporedba preciznosti modela

Na slici 80. nalazi se vizualizacija preciznosti za sva tri izrađena modela. Preciznost je, kao što je navedeno u jednom od prethodnih poglavlja, omjer broja opservacija stvarne klase i broja opservacija predviđenih kao navedena klasa, bili oni točno ili netočno predviđeni.

Klasa prosječna ocjena – model s najvećom preciznosti navedene klase je slučajna šuma, s preciznosti od 86 %.

Klasa dobar – model s najvećom preciznosti navedene klase je model slučajne šume, s preciznosti od 77 %.

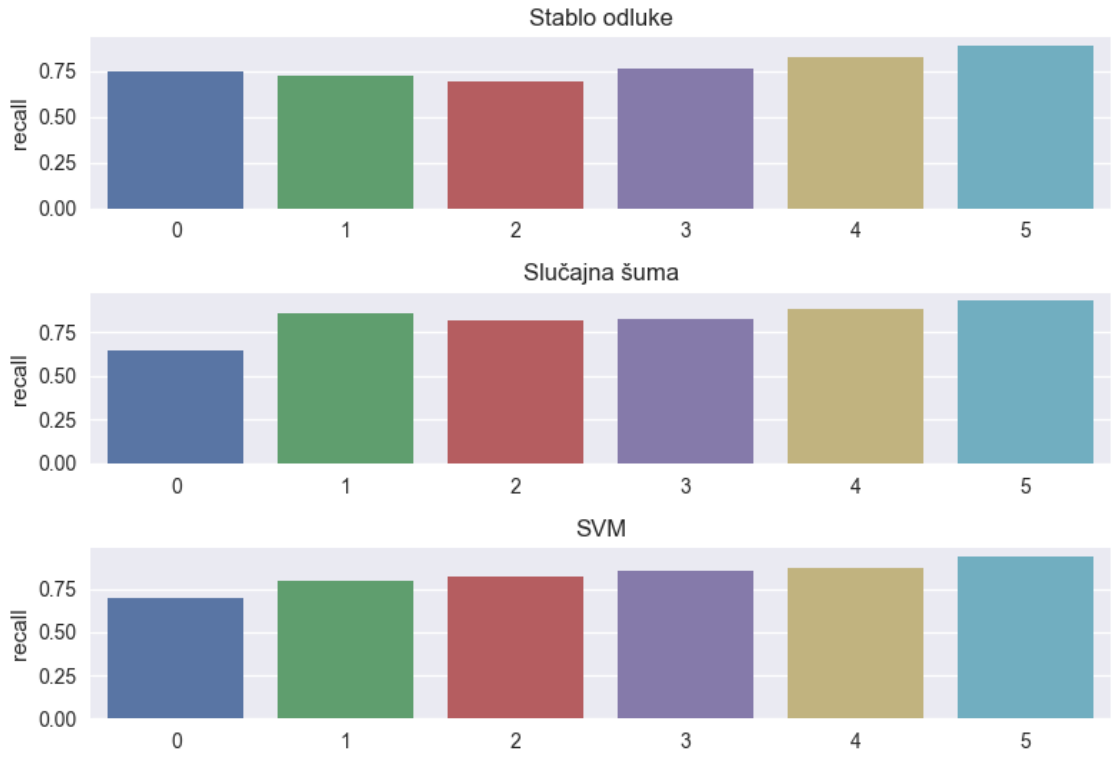
Klasa vrlo dobar – modeli s najvećom preciznosti navedene klase su model slučajne šume i SVM model, koji imaju identičnu preciznost za navedenu klasu od 85%.

Klasa sjajan – model s najvećom preciznosti navedene je model slučajne šume, s preciznosti od 79 %.

Klasa izvanredan – model s najvećom preciznosti navedene klase je SVM model, s preciznosti klase od 90 %.

Klasa izuzetan – model s najvećom preciznosti navedene klase je SVM model, s preciznosti klase od 94%.

Prema navedenim rezultatima, vidimo da je model slučajne šume više precizan kod klasa s manjom recenzijom, dok je klase s većom recenzijom preciznije predvidio SVM model.



Slika 81. Usporedba modela prema opozivu

Na slici 81. nalazi se vizualizacija opoziva za sva tri izrađena modela prema pojedinoj klasi. Opoziv je, kao što je navedeno u jednom od prethodnih poglavlja, omjer broja točno pogođenih instanci određene klase i ukupnog broja instanci određene klase.

Klasa prosječna ocjena - model s najvećim opozivom navedene klase je model stabla odluke, s opozivom klase od 75%.

Klasa dobar – model s najvećim opozivom navedene klase je model slučajne šume, s opozivom od 86 %.

Klasa vrlo dobar – modeli s najvećim opozivom su model slučajne šume i SVM, s identičnim opozivom od 82 %.

Klasa sjajan – model s najvećim opozivom navedene klase je model SVM, s opozivom od 85 %.

Klasa izvanredan – model s najvećim opozivom navedene klase je model slučajne šume, s opozivom od 88 %.

Klasa izuzetan – model s najvećim opozivom navedene klase je SVM model, s opozivom od 94 %.

Vidimo da, ako promatramo opoziv klase, model slučajne šume i SVM model imaju otprilike podjednak učinak, gdje u klasama gdje model slučajne šume ima manji opoziv, najveći ima SVM model i obrnuto. Za razliku od mjere preciznosti, vidimo također da postoji i klasa u kojoj model stabla odluke ima veći opoziv od modela slučajne šume i SVM modela, a to je klasa *Prosječna ocjena.*

ROC krivulja

ROC krivulja (engl. *Receiver Operating Characteristic*) je vizualizacija točno pozitivnih opservacija naspram lažno pozitivnih opservacija. ROC grafikon prikazuje točnost testa, što je linija bliže gornjem lijevom kutu grafikona, test je točniji. Točnost testa poznata je također pod nazivom „*područje ispod krivulje"* (engl. *area under the curve*) te što je veće područje ispod krivulje, model je precizniji prilikom predviđanja klasa. Savršeni test sastojao bi se od područja ispod ROC krivulje od 1. Dijagonalna linija grafikona označava savršenu slučajnost, tj. ako test prati dijagonalu grafikona, to bi značilo da model ima jednaku vjerojatnost pogađanja klase kao nasumično bacanje kovanice. Područje ispod dijagonale je 0.5, što znači da bi bezuspješan model za određenu klasu imao područje ispod krivulje od 0.5 [14].

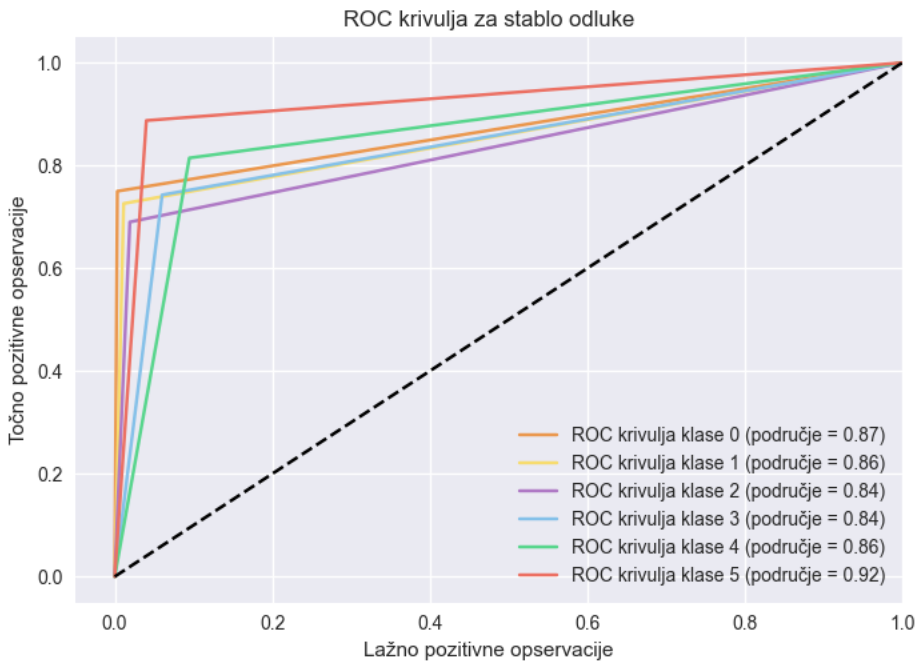
```
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
y_test = label_binarize(y_test, classes=[0, 1, 2, 3, 4, 5])
```

```
broj_klasa = y.shape[1] # 6
fpr = dict()
tpr = dict()
roc_auc = dict()
SVM.fit(x_sm, y_sm)
y_pred = SVM.predict(x_test)
y_pred = label_binarize(y_pred, classes=[0, 1, 2, 3, 4, 5])
```

Varijable sa stvarnim klasama i predikcijama je potrebno binarizirati kako bi mogli vizualizirati ROC krivulju (funkcija *label_binarize*).

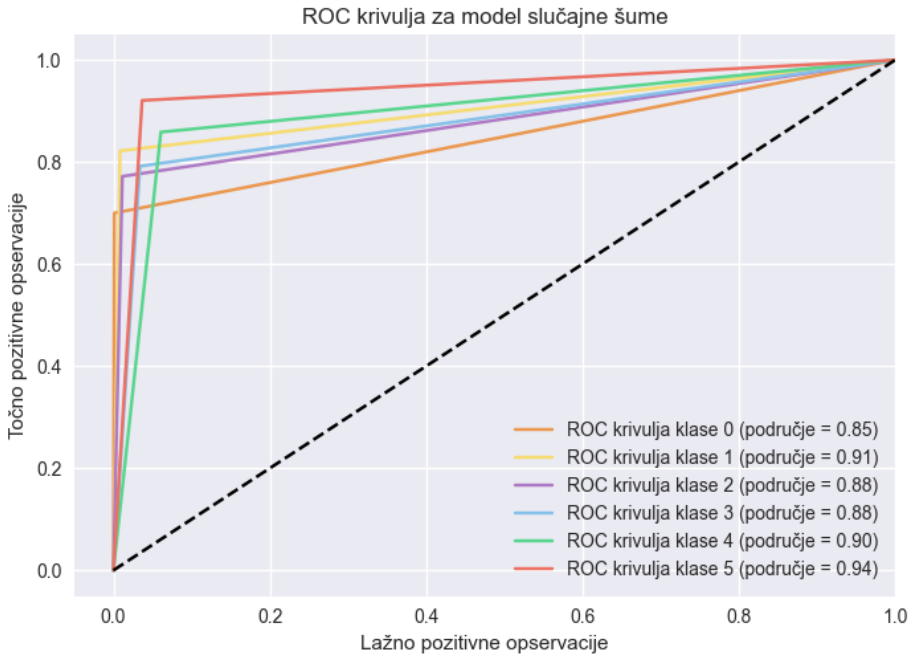
```
for i in range(broj_klasa):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

from itertools import cycle
boje = cycle(['#EC7063', '#EB984E', '#F7DC6F', '#AF7AC5', '#85C1E9', '#
58D68D'])
for i, boja in zip(range(broj_klasa), boje):
    plt.plot(fpr[i], tpr[i], color=boja,
             label='ROC krivulja klase {0} (područje = {1:0.2f})'
             ''.format(i, roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([-0.05, 1.0])
plt.xlabel('Lažno pozitivne opservacije')
plt.ylabel('Točno pozitivne opservacije')
plt.title('ROC krivulja za SVM model')
plt.legend(loc="lower right")
```

Za svaku pojedinu klasu, lažno pozitivna i točno pozitivna stopa, kao i područje ispod krivulje, spremljeni su u posebne rječnike te za svaku klasu drugom bojom označavamo na grafikonu liniju koja predstavlja područje ispod krivulje zasebnog modela za svaku posebnu klasu.



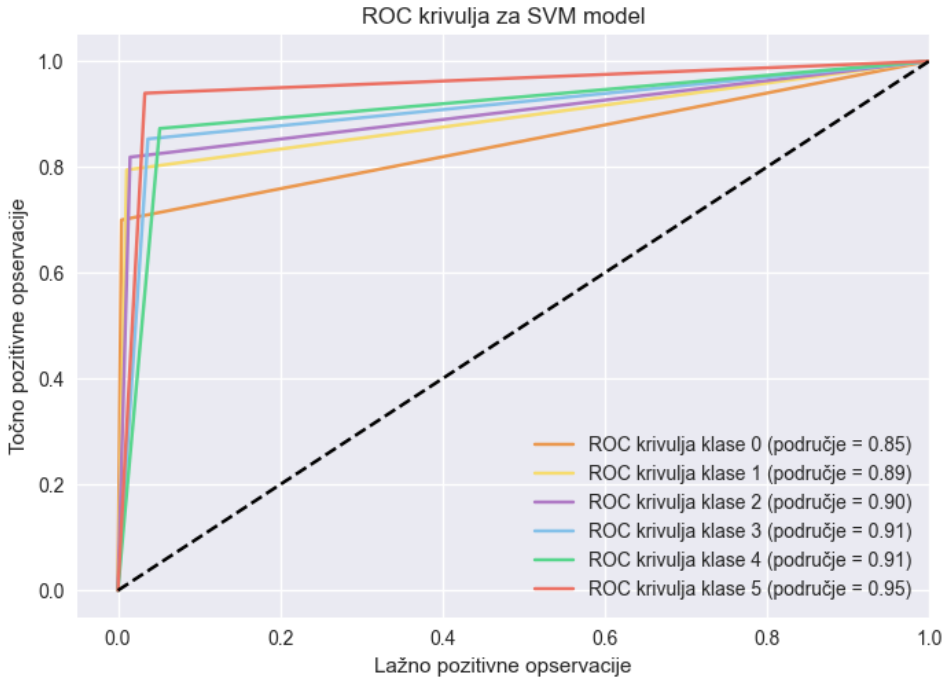
Slika 82.. ROC - Stablo odluke

Stablo odluke ima najveće područje ispod krivulje za klasu *Izuzetan* (0.92), zatim klase *Prosječna ocjena* (0.87), klase *Dobar* i *Izvanredan* (0.86) te najmanje područje ispod krivulje za klase *Vrlo dobar* i *Sjajan* (0.84).



Slika 83. ROC - Slučajne šume

Model slučajne šume ima najveće područje ispod krivulje za klasu *Izuzetan* (0.94), zatim klasu *Dobar* (0.91), klasu *Izvanredan* (0.90), klase *Vrlo dobar* i *Sjajan* (0.88) te najmanje područje ispod krivulje za klasu *Prosječna ocjena* (0.85).



Slika 84. ROC – SVM

SVM model ima, kao i ostali modeli, najveće područje ispod krivulje za klasu *Izuzetan* (0.95), zatim klase *Izuzetan i Sjajan* (0.91), klasu *Vrlo dobar* (0.90), klasu *Dobar* (0.89) te najmanje područje ispod krivulje za klasu *Prosječna ocjena* (0.85). Možemo primijetiti sličnost kod modela slučajne šume i SVM modela, koji veće područje ispod krivulje imaju za klase s većim recenzijama, model stabla odluke ima veće područje ispod krivulje od oba modela kada pogledamo klasu *Prosječna ocjena.*

Kada uzmemo prosjek rezultata svih klasa i podijelimo s brojem klasa modeli su, prema ukupnom području ispod krivulje, rangirani na sljedeći način:

1. SVM model – 0.901
2. Slučajne šume – 0.893
3. Stablo odluke – 0.865

7. Zaključak

Ovim radom prikazana je analiza podataka sa stranice booking kako bi zaključili koji čimbenici smještaja najviše utječu na zadovoljstvo korisnika smještaja. Analiza je provedena u nekoliko koraka: prikupljanje podataka s web stranice (web struganje podataka) gdje prikupljamo potrebne podatke s vanjskog izvora, čišćenje podataka gdje procesiramo i čistimo podatke kako bi bili pogodni za daljnju analizu, istraživačka analiza podataka gdje su podaci opisani pomoću statistike i raznih vizualizacija, analiza recenzija korisnika gdje je, s obzirom da je prethodno utvrđeno da na samu klasu smještaja najveći utjecaj imaju recenzije sporednih čimbenika kao što su udobnost, popularni sadržaji, vrijednost itd., analiziran tekst recenzija pisanih na engleskom i hrvatskom jeziku te je upotrijebljen Multinomial Naive Bayes algoritam za predviđanje klase smještaja s obzirom na recenziju korisnika te upotreba određenih algoritama strojnog učenja (stablo odluke, slučajna šuma, metoda potpornih vektora) za predviđanje klase smještaja s obzirom na razne karakteristike smještaja (da li objekt ima WiFI, lokacija – regija, razne recenzije, vrsta objekta, itd.). Rezultati analize pokazali su da na samu klasu smještaja najveći utjecaj imaju recenzije korisnika vezane uz udobnost smještaja, sadržaje, vrijednost za novac i slične te da generalni čimbenici, kao što su regija, županija, cijena, dopuštene zabave i slični nemaju direktnu povezanost sa samom klasom (i recenzijom) smještaja. Daljnjom analizom tekstova recenzija potvrđen je dobiveni rezultat, u objektima s višim recenzijama korisnici se više referiraju na čimbenike udobnosti smještaja i pogodnosti kao što su ljubazni domaćin, mir i tišina u objektu smještaja te vrijednost smještaja za novac, dok se generalno korisnici (u recenzijama smještaja svih recenzija) također referiraju na lokaciju smještaja u smislu da je blizu centra grada ili autobusne stanice, ljubaznost samog osoblja smještaja ili osoblja na recepciji te se također veoma često referiraju na sam doručak i sobu u sklopu smještaja. Nadalje, upotrebom navedenih algoritama strojnog učenja zaključeno je da najveći utjecaj na uspješnu predikciju modela imaju same recenzije korisnika smještaja te čimbenici kao što su veličina smještaja, vrsta smještaja i kategorizacija smještaja (broj zvjezdica). Od navedena tri modela, najveća preciznost (najviše pogođenih klasa opservacija), dobivena je pomoću modela potpornih vektora, koji je uspješno predvidio ~ 90 % opservacija. Jedno od ograničenja ovog rada jest to što booking prikazuje samo po 1 000 rezultata pretrage (s podjelom na županije imamo 8444 objekta no ukupno na bookingu za hrvatsku ima izlistano preko 70 000 objekata smještaja) te sama komputacijska snaga laptopa (Lenovo L430, 298 HDD, 8 GB RAM, Intel Core 2.50 Gz) koja se pokazala slabijom prilikom provođenja određenih zahtjeva (web struganje podataka je

trajalo predugo, gridsearch je prekinut nakon više od dana izvođenja,...) te bi idealno bilo ovakav projekt provesti preko cloud platforme koja posjeduje znatno veću komputacijsku snagu nego zasebni laptop. Također, API google prevoditelja nije bio u mogućnosti prevesti recenzije s drugih stranih jezika na engleski (njemačke, poljske, kineske) jer bi već na skupu podataka jednog jezika izbacivao upozorenje da se preko API-a šalje prevelik broj zahtjeva.

8. Literatura

[1] Mitchell, Ryan. *Web Scraping with Python.* O'Reilly Media, Inc., 2015.

[2] Tandarić, Neven, Tekić, Ivan. *Percepcija geografije u javnosti.* Zagreb: Prirodoslovno-Matematički Fakultet, 2012.

[3] Seltman, Howard J. *Experimental Design and Analysis,* 2018.

[4] „*What is ANOVA (Analysis Of Variance) and what can I use it for?".* Qualtrics.com. Dostupno na: https://www.qualtrics.com/uk/experience-management/research/anova/?rid=ip&prevsite=en&newsite=uk&geo=HR&geomatch=uk [08.06.2021].

[5] „*P-vrijednost - definicija, način korištenja i pogrešna tumačenja*". Pharoskc.com. Dostupno na: https://hr.pharoskc.com/1574-what-is-the-p-value [08.06.2021].

[6] Chen, P. Y., Popovich, P. M. *Correlation: Parametric and nonparametric measures.* Thousand Oaks, CA: Sage Publications, 2002.

[7] „*Jaccard Index / Similarity Coefficient*". Statisticshowto.com. Dostupno na: https://www.statisticshowto.com/jaccard-index/ [08.06.2021].

[8] Bowyer, Kevin W., Chawla, Nitesh V., Hall, Lawrence O., Kegelmeyer, W. Philip. *SMOTE: Synthetic Minority Over-sampling Technique.* Journal of Artificial Intelligence Research 16 (2002) 321–357.

[9] „*Multinomial Naive Bayes Explained: Function, Advantages & Disadvantages, Applications in 2021*". Upgrad.com, 2021. Dostupno na: https://www.upgrad.com/blog/multinomial-naive-bayes-explained/ [08.06.2021].

[10] Sanjay, M. „*Why and how to Cross Validate a Model?*“. Towardsdatascience.com, 2018. Dostupno na: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f [08.06.2021].

[11] Avinash, Navlani. „*Decision Tree Classification in Python*“. Datacamp.com, 2018. Dostupno na: https://www.datacamp.com/community/tutorials/decision-tree-classification-python [08.06.2021].

[12] Wood, Thomas. „*What is a Random Forest?*“. Deepai.org. Dostupno na: https://deepai.org/machine-learning-glossary-and-terms/random-forest [08.06.2021].

[13] McCullum, Nick. „*9 Key Machine Learning Algorithms Explained in Plain English*“. Freecodecamp.org, 2020. Dostupno na: https://www.freecodecamp.org/news/a-no-code-intro-to-the-9-most-important-machine-learning-algorithms-today/

[14] „*Receiver Operating Characteristic (ROC) Curve: Definition, Example*“. Statisticshowto.com. Dostupno na: https://www.statisticshowto.com/receiver-operating-characteristic-roc-curve/ [08.06.2021].