Prepoznavanje imenovanih entiteta u medijskim objavama s ciljem predviđanja rezultata parlamentarnih izbora u RH

Kraljić, Damjan

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Sveučilišni prijediplomski studij Informatika

Damjan Kraljić

Prepoznavanje imenovanih entiteta u medijskim objavama s ciljem predviđanja rezultata parlamentarnih izbora u RH

Završni rad

Mentor: prof. dr. sc. Ana Meštrović

Rijeka, kolovoz 2024.

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija
uniri

Rijeka, 3. lipnja 2024.

Zadatak za završni rad

Pristupnik/ica: Damjan Kraljić

Naziv završnog rada: Prepoznavanje imenovanih entiteta u medijskim objavama s ciljem predviđanja rezultata parlamentarnih izbora u RH

Naziv završnog rada na engleskom jeziku: Named entity recognition in media texts with the aim of predicting the results of the parliamentary elections in the Republic of Croatia

Sadržaj zadatka: Zadatak završnog rada je istražiti i implementirati algoritme za prepoznavanje imenovanih entiteta u tekstovima s online portala. Potom je potrebno razviti i analizirati postupak za predviđanje rezultata parlamentarnih izbora na temelju podataka iz medijskih objava.

Mentor/ica
Prof. dr. sc. Ana Meštrović

_____

Voditelj za završne radove
Izv. prof. dr. sc. Miran Pobar

_____

Zadatak preuzet: 3. lipnja 2024.

_____
(potpis pristupnika/ice)

Radmile Matejčić 2,
51000 Rijeka, Hrvatska

T: +385 (0)51/584-700
E: ured@inf.uniri.hr
www.inf.uniri.hr

OIB: 64218323816
IBAN: HR1524020061400006966

Sažetak

Tema ovog završnog rada je primjena tehnika strojnog učenja s ciljem predviđanja rezultata parlamentarnih izbora u Hrvatskoj 2024. godine. Za predikciju rezultata izbora korišteni su linearna regresija i algoritam slučajne šume (eng. Random Forest), podaci korišteni za treniranje modela te za kasnije predviđanje prikupljeni su putem web struganja internetskih hrvatskih medija. Implementacija modela za predikciju te prikupljanje članaka s web portala izvedeno je u programskom jeziku Python. Koristile su se biblioteke Selenium i Beautiful Soup za prikupljanje članaka te biblioteka *scikit-learn* za izradu modela linearne regresije i slučajne šume. Rezultati analize pokazuju da linearna regresija daje relativno točne rezultate s manjim pogreškama kad su u pitanju stranke sa srednjom medijskom pokrivenošću. Slučajna šuma iako je složeniji model od linearne regresije nije bio točan u predikcijama osobito kod većih stranaka.

Korištenje sentimenta i složenijih modela strojnog učenja omogućili bi preciznije prognoziranje izbora s toga preporučuje se korištenje istih za buduća istraživanja.

Ključne riječi: predviđanje rezultata izbora; web scraping; prepoznavanje imenovanih entiteta(NER); CLASSLA; linearna regresija; slučajna šuma;

SADRŽAJ

1. Uvod

U današnjem svijetu informacije i podaci postaju sve više ključni resurs. Sposobnost prikupljanja, analize interpretacije velikih količina podataka postaje presudna za donošenje odluka, u mnogobrojnim domenama, pa tako i u kontekstu politike. Zahvaljujući razvoju tehnologije pogotovo u području strojnog učenja, raste mogućnost korištenja naprednih alata za predviđanje događaja poput rezultata izbora. Ovaj rad istražuje mogućnost te točnost predikcije rezultata parlamentarnih izbora u Republici Hrvatskoj za 2024. godinu. Za treniranje modela korišteni su podaci iz poznatih hrvatskih članaka iz 2020. godine, te su primijenjeni na aktualne podatke iz 2024. godine.

U nastavku rada opisane su tehnike kako se prikupljaju podaci koristeći web scraping s bibliotekama Selenium i Beautiful Soupom u Python programskom jeziku, koji omogućuju automatizirano izvlačenje podataka s web portala. Kako bi se izvukle ključne riječi za istraživanje, poput imena i prezimena političara te imena stranaka i koalicija iz članaka koristila se metoda prepoznavanja imenovanih entiteta (engl. named entity recognition, NER) koja spada u područje analize teksta (engl. natural language processing, NLP).

Za predviđanje rezultata izbora korišteni su modeli strojnog učenja, konkretno linearna regresija koja je klasična metoda predikcije te slučajna šuma koja je metoda ansambla. Rezultati previđanja od ovih modela uspoređuju se kako bi se utvrdila preciznost istih što omogućuje donošenje zaključka koji model je prikladniji za predviđanje izbornih rezultata.

2. Opis tehnologija

U ovom poglavlju, opisane su tehnologije te metode korištene za izradu ovog rada, uključujući struganje web stranica, metode analize teksta, prepoznavanje entiteta te metode strojnog učenja kao linearne regresije i slučajne šume.

2.1. Uvod u web scraping i analizu teksta

Web scraping računalna je tehnika za izdvajanje podataka s web stranice pomoću softverskih rješenja. Simulira se ljudsko pregledavanje web stranica koristeći Hypertext Transfer Protocol (HTTP) ili putem preglednika, kao što su Google Chrome ili Mozilla Firefox. Web scraping je usko povezan s indeksiranjem stranica na webu. Upravo iz tog razloga Google i sve druge tražilice primjenjuju web scraping kako bi predstavili njihovim korisnicima ažurirane rezultate na pretrage njihovih korisnika. Web scraping se koristi i za razna istraživanja usko povezana s umjetnom inteligencijom i analize cijena kako bi se istražilo tržište i trendovi [1] [2] [3].

Web stranice postavljaju ograničenja kako bi spriječile web scraping, otkrivanjem i onemogućavanjem pregledavanja njihove stranice od strane botova ili blokiranjem prečestih zahtjeva s iste IP adrese. Kako bi se riješio ovaj problem koriste se sustavi koji se oslanjaju na tehnike poput DOM parsiranja i obrade prirodnog jezika kako bi se simuliralo ljudskog pregledavanja weba.

Za implementaciju mogu se koristiti različiti alati i biblioteke; Selenium i BeautifulSoup [4] su primjer biblioteka za programski jezik Python.

Postupci za web scrapanje su:

- identificiranje ciljane web stranice
- preuzimanje HTML strukture iste (dio ili cijele)
- izvlačenje željenih podataka iz HTML-a poput naslova, glavnog teksta, komentara, tagova
- te pohranjivanje tih podataka za daljnju obradu

Analiza teksta je proces obrade i analize nestrukturiranih tekstualnih podataka. Cilj iste je izvući korisne informacije iz tekstualnih podataka koji mogu biti u obliku članaka, objava na društvenim mrežama, komentara ili recenzija. Čišćenje podataka, tokenizacija (rastavljanje teksta na pojedinačne jedinice poput riječi), lematizacija (svođenje riječi na njihov osnovni oblik) te NER su koraci potrebni kako bi se izvršila analiza teksta [5].

2.2. Pregled tehnika za scraping podataka

HTML parsing, DOM parsing te XPath su tehnike scrapanja weba i sve dohvaćaju sadržaj s web stranica, obrađuju ga pomoću mehanizma struganja i generiraju jednu ili više podatkovnih datoteka s izdvojenim sadržajem.

HTML parsing uključuje korištenje JavaScripta za ciljanje linearne ili ugniježđene HTML stranice. To je moćna i brza metoda za izdvajanje elementa, atributa, teksta i veza (npr. ugniježđene veze ili adrese e-pošte), i povlačenje resursa. Selenium i BeautifulSoup koriste ovu vrstu parsiranja.

Objektni model dokumenta (DOM) definira strukturu, stil i sadržaj XML datoteke. Scraperi obično koriste DOM parser za dubinski pregled strukture web stranica. DOM parseri mogu se koristiti za pristup čvorovima koji sadrže informacije i struganje web stranice alatima poput XPatha. Za dinamički generirani sadržaj, strugači mogu ugraditi web-preglednike kao što su Firefox i Chrome za izdvajanje cijelih web-stranica ili njihovih dijelova. Selenium kada simulira preglednik zapravo manipulira s DOM-om; iako je cilj isti, izvući podatke iz HTML-a, pristup je drugačiji.

XPath je skraćenica za XML Path Language, koji je jezik upita za XML dokumente. XML dokumenti imaju strukture poput stabla, tako da strugači mogu koristiti XPath za navigaciju kroz njih odabirom čvorova prema različitim parametrima. Scraper može kombinirati raščlanjivanje DOM-a s XPathom kako bi izdvojio cijele web-stranice. BeautifulSoup omogućava korištenje XPath-a za pronalaženje elemenata u HTML-u [6].

2.2.1. Selenium

Selenium je automatizirani alat za testiranje otvorenog koda koji se koristi za testiranje web aplikacija u različitim preglednicima. Automatizira procese pisanih skripti, budući da skripte trebaju komunicirati s preglednikom kako bi se izvršili zadaci koji se ponavljaju poput klikanja, pomicanja, listanja i unosa podataka. Na službenoj web stranici Seleniuma opisano je kako on "Prvenstveno služi za automatizaciju web aplikacija u svrhe testiranja, ali svakako nije ograničeno samo na to." Stoga mogu se strugati podaci izgradnjom Python web scrapera koristeći sposobnosti Seleniuma za kontrolu preglednika, navigaciju DOM-om i korištenje alata za razvojne programere [7].

Prednosti korištenja Seleniuma su:

- jednostavan za korištenje jer je prvenstveno razvijen u JavaScriptu
- testovi se mogu kodirati u nekoliko programskih jezika kao što su Java, Python, Ruby, Perl i PHP
- je neovisan o platformi, što znači da se može implementirati na Windows, Linux i Macintosh operacijskim sustavima
- može se koristiti na različite preglednike kao što su Firefox, Chrome, Opera i Safari

2.2.2. Beautiful Soup

Beautiful Soup je Python biblioteka koja koristi unaprijed instalirani HTML/XML parser i pretvara web stranicu iz HTML-a ili XML-a u strukturu koja se sastoji od elemenata, atributa, klasa, tagova i vrijednosti. Beautiful Soup pruža jednostavne metode za navigaciju, pretraživanje i modificiranje stabala analize. Ova Python biblioteka ne dolazi s web

indeksiranjem, tj. ne dohvaća izravno web stranicu. Za pružanje HTML sadržaja oslanja se na vanjsku biblioteku. Obično se to postiže korištenjem Python ugrađene biblioteke *urllib* ili *requests*.

Metode *.find* i *.find_all* koriste za pretraživanje stabla, pružajući ulazne argumente. Ulazni argumenti su: naziv taga, nazivi atributa i drugi povezani argumenti. Ti se argumenti mogu predstaviti kao: niz, regularni izraz, popis ili čak funkcija.

Uobičajena upotreba objekta BeautifulSoup uključuje:

- Pretraživanje po CSS klasi
- Pretraživanje po adresi hiperveze
- Pretraživanje po ID-u elementa, tagu
- Pretraživanje po nazivu atributa.

Vrijednost atributa Beautiful Soup pojednostavljuje raščlanjivanje HTML-a i smanjuje složenost struganja weba.

Jednostavan je za korištenje, može analizirati nesavršen ili neispravan HTML kod te pruža korisne metode za pretraživanje i navigaciju stablima analize. Beautiful Soup ne obrađuje dinamički sadržaj učitan JavaScriptom. Međutim, može se koristiti u kombinaciji s alatima kao što je Selenium koji mogu renderirati dinamičke stranice prije prosljeđivanja HTML-a u Beautiful Soup [8].

2.3. Prepoznavanje imenovanih entiteta (NER)

Prepoznavanje imenovanih entiteta, dio je područja proučavanja koje se naziva ekstrakcija informacija (engl. Information Extraction, IE) . Bavi se identificiranjem i ispravnim klasificiranjem imenovanih entiteta prisutnih u tekstu kao što su ljudi, mjesta, organizacije, brojevi, vremenski izrazi itd. Kombiniranjem izdvojenih informacija računalni sustav može definirati temu teksta i pravilno ga razumjeti. Stoga pripada području obrade prirodnog jezika koji se može podijeliti u tri kategorije:

- Sintaksa - Razumijevanje strukture i pravila jezika
- Semantika - Izvođenje značenja riječi, teksta i diskursa i utvrđivanje njihovih odnosa
- Govor - Identificirati i prepoznati izgovorene riječi i pretvoriti ih u tekst

NER pomaže u semantičkom dijelu NLP-a, izdvajajući značenje riječi, identificirajući ih i locirajući ih na temelju njihovih odnosa. Kad bi se trebalo tražiti određene riječi u blogu od preko 10 000 članaka konvencionalnim načinom, sustav bi trebao pretraživati unutar svakog od njih, a budući da memorija je ograničena, ova operacija trebat će se ponoviti za svaki novi upit. Ovakav sustav, osim što nije baš skalabilan, pokazuje se i neučinkovitim. NER rješava ovaj problem analizom svakog članka samo jednom, izdvajanjem ključnih riječi i popunjavanjem popisa imenovanih entiteta koji se mogu koristiti u upitima za pretraživanje.

Postoji nekoliko metoda za implementaciju NER-a, od klasičnih metoda temeljenih na pravilima do naprednih modela dubokog učenja. U ovom istraživanju, uspoređeni su tri različita NER modela: Stanza, BERTić i CLASSLA.

Prepoznavanje nominativnih entiteta ima svoje propuste i postoji manji postotak pogreške, ljudima je intuitivno dovoljno jasna kategorija kojoj pojam pripada. Međutim, to nije slučaj s računalima: ona se susreću s problemima klasifikacije [9].

U ovom primjeru vidljivo kako NER ima svoja ograničenja, pošto kratica AP nije organizacija nego osoba (kratica od imena Andrej Plenković) te da samo s dodatnim treniranjem je moguće smanjiti pogreške istog.

Andrej Plenković (Osoba) *u Tuhlju dao podršku HDZ-ovoj koaliciji.*

Hrvatska je u EU u vrhu po korupciji u koju su upleteni AP (Organizacija) *i njegova svita.*

2.3.1. Pregled metoda za NER

Primarni cilj metode NER je označavanje entiteta u tekstualnim dokumentima i njihova klasifikacija. U tu se svrhu općenito koriste sljedeća tri pristupa. Međutim, može se kombinirati jedna ili više metoda.

Sustav temeljen na rječniku možda je najjednostavniji i najtemeljniji NER pristup. Koristit će se rječnik s mnogo riječi, sinonima i zbirke rječnika. Sustav će provjeriti je li određeni entitet prisutan u tekstu također dostupan u rječniku. Pomoću algoritma za podudaranje nizova entiteti se unakrsno provjeravaju. Nedostatak korištenja ovog pristupa je da postoji potreba za stalnim ažuriranjem skupa podataka vokabulara kako bi NER model učinkovito funkcionirao.

U sustavu temeljenom na pravilima informacije se izdvajaju na temelju skupa unaprijed postavljenih pravila. Koriste se dva glavna skupa pravila:

- Pravila temeljena na uzorku – pravilo temeljeno na uzorku slijedi morfološki obrazac ili niz riječi korištenih u dokumentu.
- Pravila temeljena na kontekstu – ovise o značenju ili kontekstu riječi u dokumentu.

U sustavima temeljeni na strojnom učenju, statističko modeliranje koristi se za otkrivanje entiteta. Ovaj pristup se oslanja na značajke koje opisuju tekstualni dokument. Prednost ove metode s usporedbom s tradicionalnim metodama je što može prepoznati različite varijacije u pisanju entiteta. Metode dubokog učenja za NER koriste neuronske mreže poput rekurentnih neuronskih mreža (RNN) i transformatora za razumijevanje dugoročnih ovisnosti o tekstu. Glavna prednost korištenja ovih metoda je njihova sposobnost da obrade veliku količinu podataka te da uče od istih. Nedostatak ove metode je što zahtjeva značajnu računalnu snagu za obuku i implementaciju.

Hibridne metode su kombinacija pristupa kao što su pristupi temeljeni na pravilima, statističko i strojno učenje za izdvajanje imenovanih entiteta. Cilj je kombinirati prednosti svake metode uz smanjenje njezinih slabosti. Najbolji aspekt korištenja hibridnih metoda je fleksibilnost koja se dobivaju kombiniranjem više tehnika pomoću kojih se mogu izdvojiti entiteti iz različitih izvora podataka.

2.3.2. Usporedba modela Stanza, BERTić, i CLASSLA

U ovom radu, analizirana su tri različita modela za zadatak NER: Stanza, BERTić te CLASSLA, kako bi se utvrdilo koji od njih najbolje prepoznaje entitete te koje različite funkcionalnosti ima svaki model.

Stanza je Python alat za analizu prirodnog jezika razvijen od strane Stanford NLP Grupe [10]. Ima alate koji se mogu koristiti u cjevovodu za pretvaranje niza teksta na ljudskom jeziku u popise rečenica i riječi, za proizvodnju morfoloških značajki i osnovnih oblika tih riječi, za pružanje analize ovisnosti o sintaktičkoj strukturi i za identificiranje imenovanih entiteta. Stanza podržava više od 70 jezika od kojih i hrvatski ali može biti manje precizan posebno u usporedbi s modelima koji su razvijeni za specifičnu jezičnu porodicu

BERTić je model koji se temelji na BERT arhitekturi, njegov transformativni model temeljen na principu transformera, koji je prethodno treniran na 8 milijardi tokena izvađenih s web stranica na hrvatskom, bosanskom, srpskom i crnogorskom jeziku [11]. Zbog toga rezultati BERTić modela često znaju biti točniji od drugih modela koji nisu ciljano trenirati za određeni jezik neko za veću skupinu istih.

CLASSLA je alat posebno prilagođen slavenskim jezicima [12]. Razvijen je kroz CLARIN infrastrukturu i uključuje metode koje su specifične za obrade slavenskih jezika, čineći ga posebno korisnim za hrvatski jezik. CLASSLA kombinira tradicionalne metode s modernim dubokim učenjem, što mu daje prednost u situacijama kada je potrebno raditi s tekstovima bogatim slavenskim morfološkim obilježjima.

2.4. Linearna regresija i slučajna šuma

Linearna regresija koristi linearni odnos između neovisne i jedne zavisne ili više zavisnih varijabli za predviđanje budućih događaja. Statistička je tehnika za prediktivnu analizu koja se koristi u znanosti o podacima i strojnom učenju.

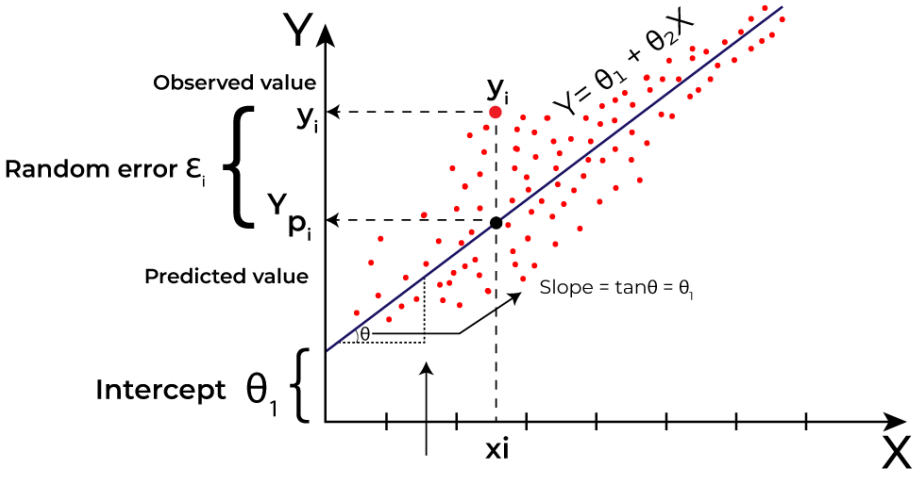
Linearna regresija je tehnika nadziranog učenja koja predviđa kontinuirane ili numeričke varijable poput prodaje, dobi, cijene proizvoda i broj mandata simulacijom matematičkog odnosa između varijabli.

Matematika iza ovog modela izražava se kroz jednadžbu pravca:

$$Y_i = \beta_0 + \beta_1 X_i + u_i$$

Gdje su:

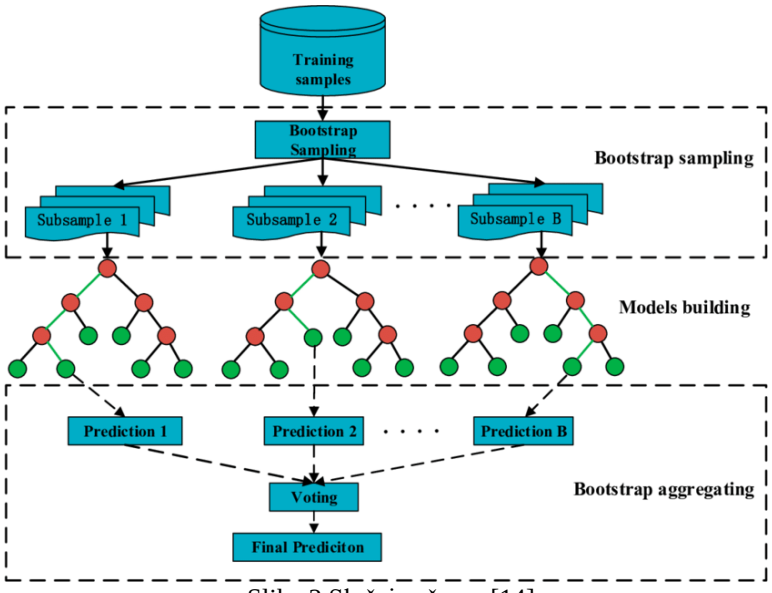
- ***Y***: zavisna varijabla koju predviđamo,
- ***X***: nezavisna varijabla koju koristimo za predviđanje,
- $\boldsymbol{\beta_0}$: presjek pravca s *Y* osi, poznat kao konstanta,
- $\boldsymbol{\beta_1}$: nagib pravca, odnosno koeficijent koji pokazuje koliko se *Y* mijenja za jedinicu promjene u *X*,
- $\mathbf{u_i}$: pogreška ili rezidual, koja predstavlja razliku između stvarne vrijednosti *Y* i predviđene vrijednosti *Y*.



Slika 1 Linearna regresija [13]

Linearna regresija je jednostavna za implementirati te kad postoji jasna linearna veza između varijabli jako je efikasna metoda no ista osjetljiva je na iznimke te mora se paziti na pretpostavke jer u suprotnome model može dati netočne rezultate [13].

Slučajna šuma je metoda strojnog učenja koja se koristi za klasifikaciju i regresiju u strojnom učenju. Ova metoda se može uvrstiti u kategoriju modela ansambla jer kombinira više modela kako bi se postigla bolja točnost traženih rezultata. Ime metode je usko povezano s načinom rada algoritma te broj stabala koji se koriste u algoritmu određuje njegovu točnost i sposobnost rješavanja problema [14].



Slika 2 Slučajna šuma [14]

Princip rada algoritma slučajne šume:

- Uzorkovanje s ponavljanjem (engl. bootstraping): ansambl slučajne šume počinje stvaranjem različitih podskupova podataka za obuku. Ti se podskupovi stvaraju pomoću uzorkovanja sa zamjenom iz izvornih podataka. Ovaj proces poznat je kao pakiranje u vrećice. Svaki podskup predstavlja jedinstveni set za obuku za jedno stablo.

- Izgradnja stabala odlučivanja: za svaki podskup podataka kreira se stablo odlučivanja. Svako se stablo obučava korištenjem samo podskupa podataka, ali može imati razlike u strukturi i podjeli čvorova zbog nasumičnog uzorkovanja.

- Prosjek: nakon što su sva stabla obučena, predviđanja sa svakog stabla se kombiniraju. U slučaju razvrstavanja uzima se u obzir klasa koju predviđa većina stabala (većina glasova). U slučaju regresije, predviđanja svih stabala su prosječna.

- Smanjenje prekomjernog prilagođavanja (engl. overfitting): upotreba različitih podskupova podataka i nasumičnost uvedena tijekom stvaranja stabla pomažu u smanjenju prekomjernog prilagođavanja, što je čest problem u pojedinačnim stablima odlučivanja.

3. Prikupljanje podataka

U ovom poglavlju bit će opisan proces prikupljanja potrebnih podataka potrebnih za treniranja modela za predikciju rezultata parlamentarnih izbora 2024. u Hrvatskoj. Prikupljanje podataka se može podijeliti u tri dijela: odabir izvora podataka, odabir tehnika za web scraping te priprema i čišćenje podataka za daljnju analizu.

3.1. Odabir izvora podataka

Prvi korak kako bi se skupili podaci bio je odabrati kvalitetne i raznolike izvore podataka. Odabrana su tri poznata hrvatska medijska portala: N1 Hrvatska, Index.hr te Večernji list. Osim same popularnosti gledalo se da izvori zastupaju različite političke perspektive kako se ne bi se unaprijed favorizirale određene političke stranke.

N1 Hrvatska je regionalni medijski portal koji je također regionalni partner CNN-a, vijesti su često usmjerene na politiku te aktualne političke događaje kao što su izbori. Smatra se neutralnim iako povremeno zna kritizirati konzervativne politike [15] [16].

Index.hr je među najčitanijim hrvatskim medijima, obuhvaća širok spektar tema kako u Hrvatskoj tako i u inozemstvu, duboko obrađuje političke afere te aktualne događaje poput parlamentarnih izbora. Njihovo izvještavanje često favorizira liberalne stavove i kritički se odnosi prema desnim političkim strankama [17] [18].

Večernji list je jedan od najstarijih i najčitanijih novina u Hrvatskoj. Nudi tradicionalniju i konzervativniju sliku političkih događaja. U svojim analizama i komentarima često favorizira vladajuću stranku (HDZ) iako ponekad su prisutne i kritike prema istoj [19] [20].

N1 Hrvatska i Večernji list imaju opciju pretplate prilikom čitanja premium članaka na njihovim web portalima te zbog toga njihovi tekstovi znaju biti kvalitetniji od članaka od Index.hr-a koji nema mogućnost pretplate na portalu. Svi podaci korišteni u ovom radu preuzeti su isključivo iz besplatno dostupnih verzija ovih izvora.

3.2. Web scraping članaka

Web scraping je proces automatskog prikupljanja podataka s web stranica. Koristile su se dvije tehnike za scrapanje članka s portala Index.hr, N1 Hrvatska te Večernji list; Selenium za prikupljanje linkova članaka povezana s izborima i politikom te Beautiful Soup za izvlačenje željenog sadržaja iz prethodno preuzetih članaka.

3.2.1. Prikupljanje linkova pomoću Seleniuma

Prvi postupak prikupljanja podataka koristeći web scraping je prikupljanje izvora (linkovi) članaka iz kojih se žele izvući podaci.

Za rješavanje ovog zadatka koristio se Selenium posebice jer omogućuje simulaciju korištenja preglednika i web stranica što je bilo potrebno za pronalazak željenih članaka. Svaki izvor je zahtijevao izradu vlastitog scrapera pošto svaki mediji ima različiti način prikaza elemenata te i imena klasa ili atributa istih.

Prvi korak potreban za korištenje Seleniuma u Pythonu je instalacija biblioteke pomoću komande:

```
pip install selenium
```

Potrebno je provjeriti prisutnost reklama koje mogu promijeniti položaj elemenata na stranici, što može dovesti do pogrešaka tokom scrapanja. To je bio slučaj Večernjeg lista, gdje su takve reklame bile prisutne, iz tog razloga je bilo potrebno koristiti adblocker.

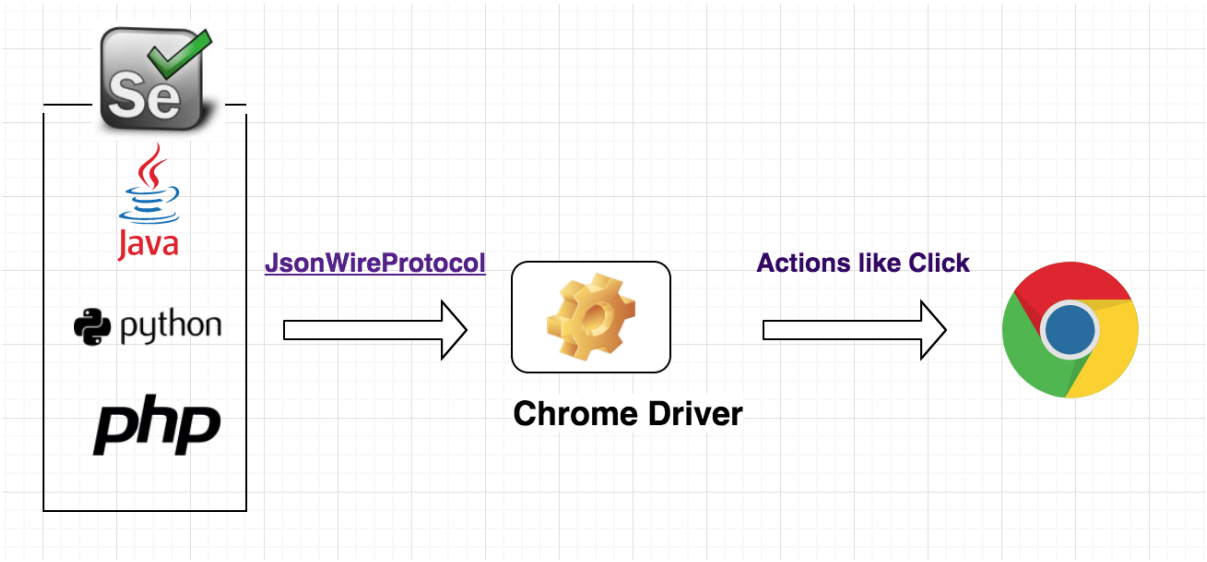
```
extension_path = '~/.config/google-
chrome/Default/Extensions/cjpalhdlnbpafiamejdnhcphjbkeiagm/1.55.0_0'

chrome_options = Options()

chrome_options.add_argument(f'--load-extension={extension_path}')
```

Primjer pokretanja pogonitelja za otvaranje web preglednika i dohvaćanje adrese željenog izvora:

```
option = webdriver.ChromeOptions()
driver = webdriver.Chrome(options = chrome_options)
driver.get('https://www.vecernji.hr/')
```



Slika 3 Selenium i korištenje web drivera [21]

Nakon učitavanje web stranice potrebno je prihvatiti kolačiće klikom na gumb te nakon što je ovaj postupak uspješno odrađen, slijedeći korak je kliknuti na pretragu te unijeti željene filtre u slučaju Večernjeg lista, ili unijeti ključne riječi poput: parlamentarni, izbori 2024, HDZ, SDP itd. za N1 i Index.hr.

Svaka stranica ima drugačiji pristup organizacije članaka, Večernji i Index.hr grupiraju članke po stranicama pa je potrebno simulirati klik na sljedeću stranicu sve do zadnje stranice. N1 ima drugačiji pristup te potrebno je listati sve dok se ne dođe do zadnjeg članka.

Primjer simulacije listanja do zadnjeg članka:

```
while True:
        body.send_keys(Keys.END)
        time.sleep(0.5)

        current_count = len(driver.find_elements(
            By.CSS_SELECTOR, '.search-results article'))

        if current_count == initial_count:
            break

        initial_count = current_count
        scrolls += 1
        article_elements = driver.find_elements(
            By.CSS_SELECTOR, '.uc-post-grid-item')
        article = article_elements[-1]
```

Zadnji korak obuhvaća dohvaćanje poveznica članaka te spremanje istih u JSON datoteku koja će se koristiti u idućem koraku [7].

3.2.2. Izvlačenje sadržaja članaka pomoću Beautiful Soupa

Nakon dohvaćanja linkova članaka koristeći Selenium, idući korak je izvlačenje sadržaja članaka koristeći Beautiful Soup pošto Selenium nema funkcije koje bi to izvršile. Beautiful Soup je Python biblioteka koja pretvara web stranicu iz HTML-a ili XML-a u strukturu raznih elemenata.

Kao što je bio slučaj za Selenium, svaki izvor je zahtijevao svoj scraper koristeći Beautiful Soup jer svaki element koji se trebao izvući iz HTML-a bio je organiziran na jedinstven način te s različito imenovanim klasama ili ID-ovima elemenata.

Kako bi se koristio Beautiful Soup u Pythonu potrebno je prvo instalirati biblioteku pomoću naredbe:

```
pip install beautifulsoup4
```

Zatim je potrebno dohvatiti sadržaj web stranice koristeći biblioteku *requests* koristeći HTTP GET zahtjev te samo ako je odgovor uspješan se nastavlja s parsiranje HTML koda dohvaćene web stranice.

```
response = requests.get(url)
    if response.status_code == 200:
        soup = BeautifulSoup(response.content, 'html.parser')
```

Biblioteka Beautiful Soup u Pythonu ima razne metode za dohvaćanje elemenata poput *find*, *find_all* te *select* metoda.

Find se koristi za pronalazak prvog elementa koji odgovora zadanim kriterijima. Moguće je dohvatiti samo jedan element pa se obično koristi na elemente koji su jedinstveni u kontekstu poput naslova ili specifičnog elementa.

Find_all pronalazi sve elemente koji odgovaraju zadanim kriterijima, primjer korištenja ove metode je pronalazak svih paragrafa <p> ili linkova <a> na stranici.

Select koristi CSS selektore kako bi pronašao elemente [8].

CSS selektori:

- Klasni
- ID
- Jednostavni
- Kontekstni
- Pseudoklase

Primjer koda gdje se dohvaćaju naslov, sadržaj i tagovi članaka u kojem su korištene opisane metode:

```
title = soup.find('h1', class_='vijesti-text-parsed title js-main-title')

content_paragraphs = []
paragraphs = soup.select('div.text.vijesti-link-underline p')

if paragraphs:
    paragraph = paragraphs.find_all('p')
    for p in paragraph:
        if p:
            content_paragraphs.append(p.text.strip())

tags = soup.findAll('a', class_="vijesti-bg-hover tag-item")
       tag_items = []
       if tags:
           for tag in tags:
               if tag:
                   tag_items.append(tag.text.strip())
```

Zadnji korak nakon što je dohvaćen sav sadržaj prikupljenih članaka, podaci se strukturiraju te spremaju u JSON datoteku. Primjer strukture JSON datoteke:


```
 {
        "title": "Plenković na Fejsu objavio zadnju poruku prije izbora: Bilo je propusta,
žalimo",
        "content_paragraphs": [
            "PREDSJEDNIK HDZ-a i tehnički predsjednik vlade Andrej Plenković  … vikao pred
HDZ-ovcima: Nek hrvatski narod kazni Milanovića"
        ],
        "tags": [
            "#HDZ",
            "#Andrej plenković",
            "#izbori 2024"
        ],
        "link": "https://www.index.hr/vijesti/clanak/plenkovic-vice-pred-hdzovcima-a-na-
fejsu-biracima-porucuje-bilo-je-propusta/2556740.aspx",
        "publish_date": "15.4.2024."
  },
```

3.3. Priprema podataka za analizu

Podaci prije analize trebaju proći proces normalizacije teksta, filtriranja i spajanja članka u jedan podatkovni okvir (*DataFrame*).

Prvi korak obuhvaća spajanje svih članaka iz tri različita izvora u jedan pandas *DataFrame* kako bi se pojednostavila analiza te manipulacija i procesi normalizacije i filtriranja podataka.

Proces normalizacije teksta pretvara dijakritičke znakove u slova te sva slova pretvara u mala kako bi se ujednačio oblik riječi i kako bi se izbjegle moguće razlike ili greške koje su mogle nastati tijekom pisanja samih članaka.

Proces filtriranja članaka uključuje brisanje svih onih članaka koji su greškom dohvaćeni i ne sadrže ni jednu ključnu riječ koja je povezana s hrvatskim političarima, strankama, izborima ili izbornim agencijama [22].

Tablica 1 Pregled prikupljenih članaka u 2020. godini

Izvor	Broj članaka	Period prikupljanja
Index.hr	879	1.1.2020. – 3.7.2020.
Večernji List	1472	1.1.2020. – 3.7.2020.
N1 Hrvatska	1054	1.1.2020. – 3.7.2020.

Tablica 2 Pregled prikupljenih članaka u 2024. godini

Izvor	Broj članaka	Period prikupljanja
Index.hr	2767	1.1.2024. – 15.4.2024.
Večernji List	2351	1.1.2024. – 15.4.2024.
N1 Hrvatska	2264	1.1.2024. – 15.4.2024.

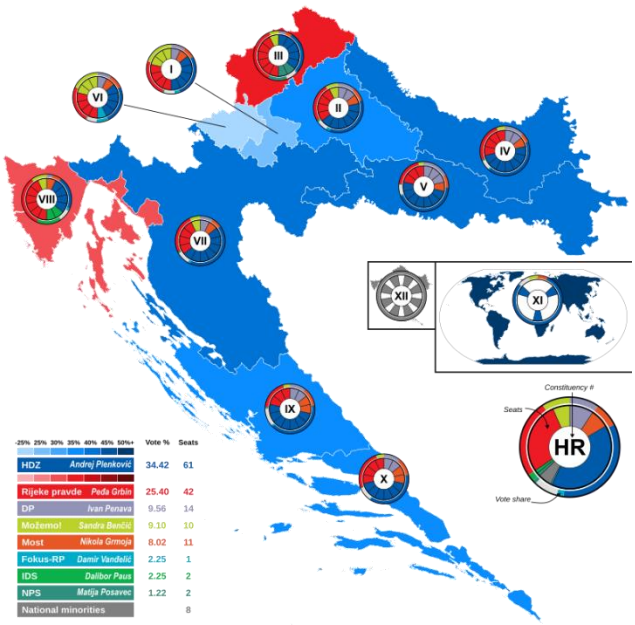
4. Analiza podataka

U ovom poglavlju obradit će se proces analize podataka koji ima cilj dobiti korisne informacije iz prikupljenih podataka. Korištene tehnike analize podataka su: vizualizacija podataka te prepoznavanje entiteta prisutni u preuzetim člancima. Analizom podataka pomoću ovih metoda omogućuje se lakše razumijevanje te interpretacija istih.

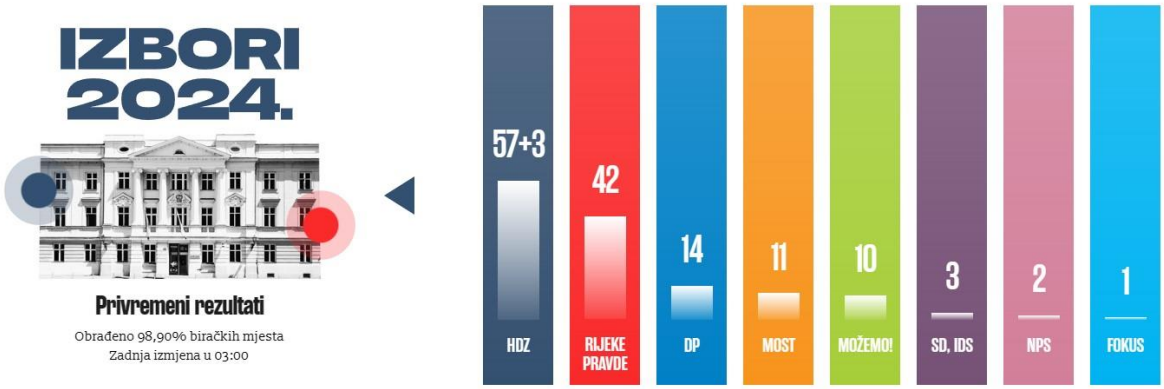
4.1. Vizualizacija podataka

Vizualizacija podataka predstavljanje je informacija sposobno pretvoriti podatke u vrijednost i učinkovito ih prenijeti.
Doista, svi podaci sadrže informacije, ali da se steklo znanje iz podataka, prvo ih morate izdvojiti. potom ih je potrebno vizualizirati i analizirati. Stoga je cilj vizualizacije podataka je stvoriti grafikone koji pružaju jasan i točan prikaz informacija, a zatim prijeći na analizu podataka te poboljšati donošenje odluka.

Ne postoji samo jedna mogućnost vizualizacije podataka, postoje različite grafičke konstrukcije te svakoj situaciji postoji tip koji je prikladniji od drugih. Najpoznatije vrste vizualizacije podataka uključuju grafikone poput stupčastih, linijskih te tortnih dijagrama, infografike, kartograma, histograma te gantograma.



Slika 4 Kartogram rezultata parlamentarnih izbora u RH u 2024. [23]



Slika 5 Histogram privremenih rezultata parlamentarnih izbora 2024. u RH [24]

U prikupljenim podacima, tagovi su prikupljeni iz svakog članka te spremljeni su u posebnu listu kako bi se izdvojili od ostatka teksta. Svaka lista tagova se zatim standardizira te spaja u jednu zajedničku u kojoj se uklanjaju nepotrebni znakovi poput „#“, točaka i razmaka. Slika 6. napravljena je pomoću funkcije *value_counts()* te biblioteke *matplotlib* u Pythonu. Funkcija *values_counts()* iz *pandas* biblioteke omogućuje brojanje učestalosti tagova u skupu podataka te *matplotlib* biblioteka omogućuje prikaz distribucije tagova u korištenim člancima.
Prilikom struganja svakog web portala prikupili su se datumi objave svakog preuzetog članka. Zbog boljeg razumijevanja, napravila se vizualizacija distribucije članaka po danima i tjednima u 2024. godini (slika 7. i 8.). Svaki portal koristi različitu vrstu formata datuma, pa bilo je potrebno standardizirati format datuma (DD.MM.YYYY.).

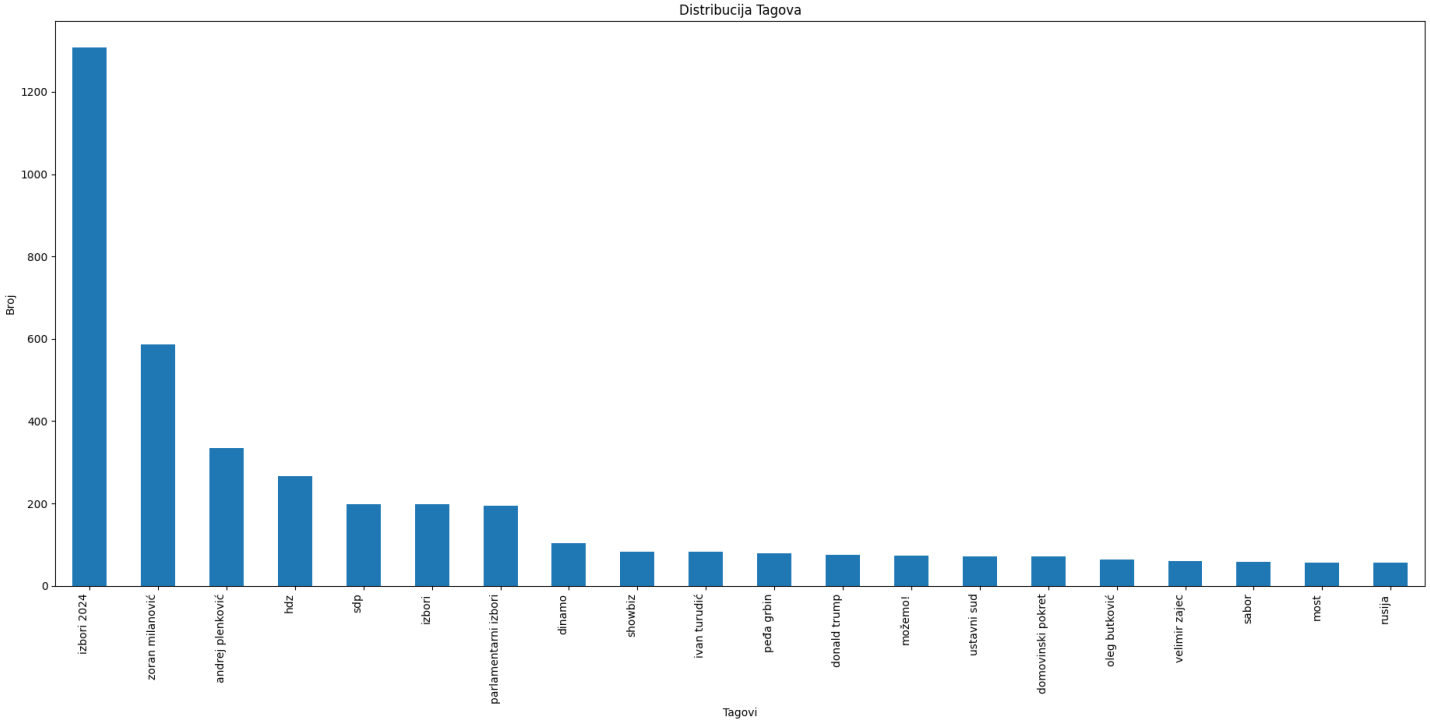
Što se tiče slike 7. datumi su grupirani po jedinstvenim vrijednostima. X-os predstavlja pojedinačne datume dok y-os prikazuje broj članaka objavljenih na taj datum. Slika je napravljena koristeći *matplotlib*. Na isti način se generirala slika 8., jedina razlika je postupak grupiranja podataka korištenih za vizualizaciju. Datumi su se grupirali po tjednima od ponedjeljka do nedjelje. X-os prikazuje raspon datuma za svaki tjedan te y-os prikazuje broj članaka koji su objavljeni u tom tjednu.

Korišteni kod za generiranje slike 8.:

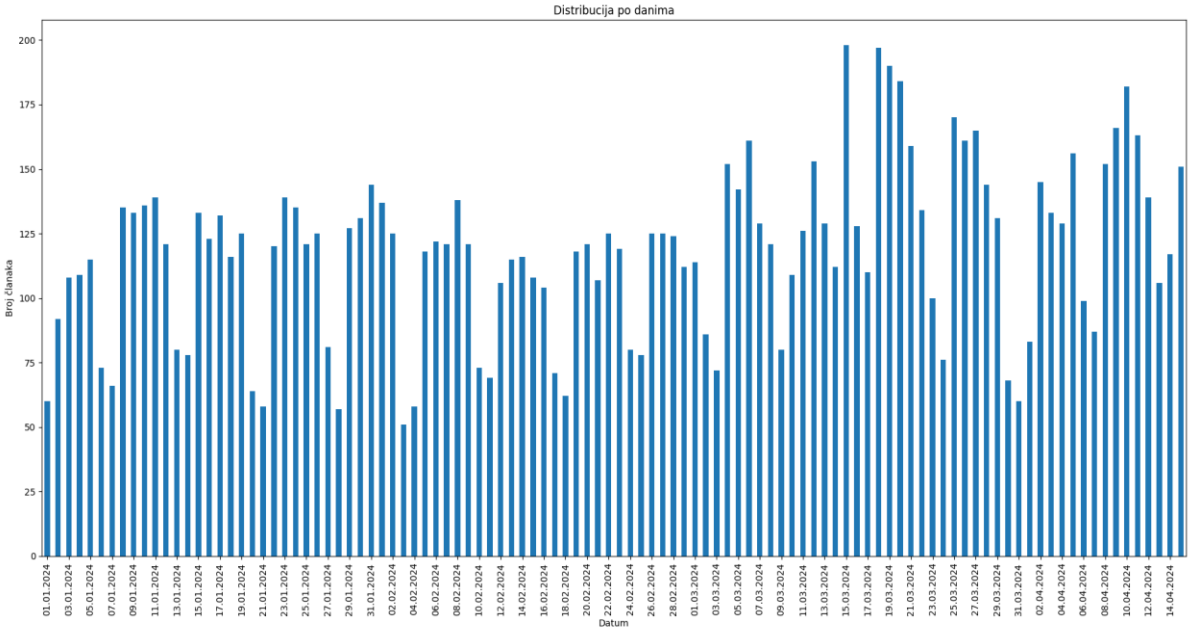
```
filtered_data['week'] = filtered_data['date'].dt.to_period('W').apply(lambda r:
r.start_time)
week_counts = filtered_data['week'].value_counts().sort_index()

plt.figure(figsize=(12, 8))
week_counts.plot(kind='bar')
tick_positions = range(len(week_counts))
tick_labels = [f"{week_counts.index[i].strftime('%d.%m.%Y')} -
{((week_counts.index[i] + pd.DateOffset(days=6)).strftime('%d.%m.%Y'))}" for i in
tick_positions]

plt.title('Distribucija po tjednima')
plt.xlabel('Tjedan')
plt.ylabel('Broj članaka')
plt.xticks(tick_positions, tick_labels, rotation=90)
plt.tight_layout()
plt.show()
```



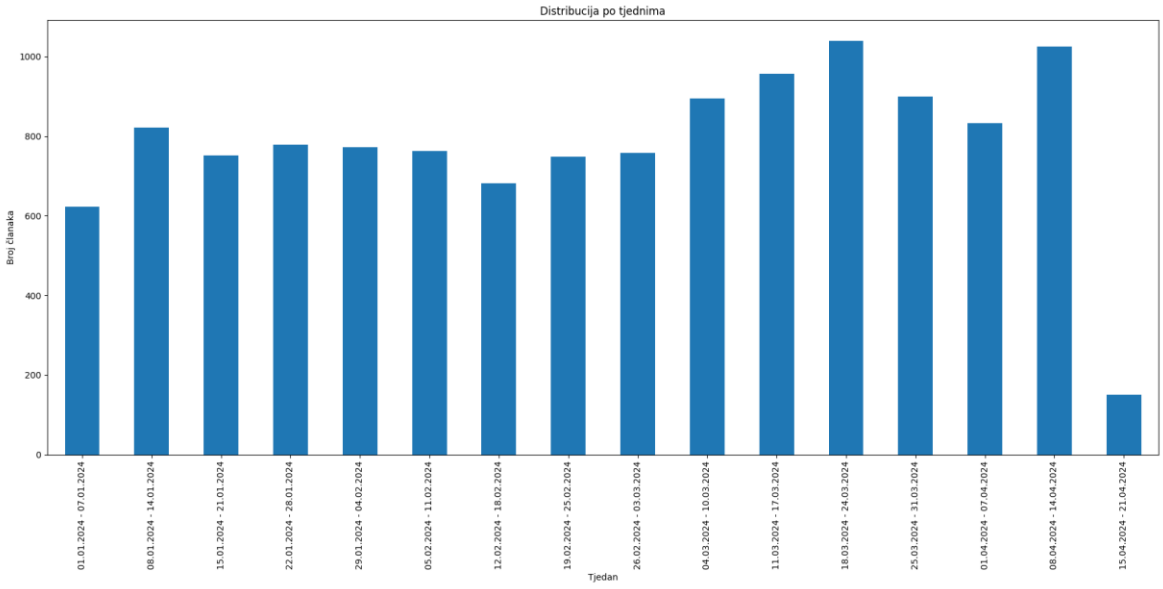
Slika 6 Distribucija najčešćih 20 tagova u scrapanim člancima



Slika 7 Distribucija članaka u 2024. po danima

Iz prikazanih histograma (slika 7 i 8) moguće je vidjeti kako subotom i nedjeljom je objavljeno manje članaka nego u ostalim danima te kako je 15.03.2024. bio dan s najviše objavljenih članaka iz obrađenih izvora. Dan 15.03.2024. je dan kad je trenutni predsjednik države Zoran Milanović objavio svoj ulazak u kampanju parlamentarnih izbora, iz tog razloga je u tom tjednu napisano najviše članaka koji imaju veze s politikom u razdoblju između 1.1 i 15.04.2024.

Zadnji tjedan ima najmanje napisanih članaka jer je izabrani dan izbora 17.04.2024. srijeda, 16.04.2024. je krenula predizborna šutnja s toga je 15.04.2024. (ponedjeljak) zadnji dan prikupljanja članaka.



Slika 8 Distribucija članaka u 2024. po tjednima

Važno je koristiti točnu vrstu vizualizacije kako bi se podaci mogli jednostavno interpretirati. Netočna vizualizacija podataka mogla bi uzrokovati netočnu interpretaciju onoga što se prikazuje i stoga dovesti do netočnih analiza i negativnih odluka.

4.2. Prepoznavanje imenovanih entiteta

Prepoznavanje entiteta ključna je komponenta u analizi tekstualnih podataka, posebno kada je riječ o izvlačenju informacija poput imena političara te naziva političkih stranaka, lokacija i drugih specifičnih entiteta iz velike količine tekstualnih podataka. U ovom dijelu rada opisan je rad tri različita alata za prepoznavanje entiteta, svaki od njih ima drugačije funkcionalnosti s toga je cilj je bio identificirati najprecizniji model koji može efikasno obraditi tekstove vezane za politiku, omogućujući točno prepoznavanje entiteta i njihovih morfoloških varijacija za hrvatski jezik.

U nastavku, detaljno je opisana implementacija svakog od ovih modela, uz naglasak na njihovu praktičnu primjenu u prepoznavanju entiteta u političkim tekstovima.

4.2.1. Implementacija Stanza, BERTić i CLASSLA modela

Za prepoznavanje entiteta u tekstualnim podacima vezanim za političare i političke stranke, korištena su tri različita modela: Stanza, BERTić, i CLASSLA. Koristila su se tri različita alata ako bi se utvrdila što veća preciznost u prepoznavanju entiteta u preuzetim člancima na hrvatskom jeziku.
Stanza je prvi alat koji se koristio za prepoznavanje entiteta. Utvrđeno je kako Stanza pogrešno prepoznaje te kategorizira specifične entitete. Pošto Stanza nije alat koji je izrađen

specifično za hrvatski jezik ili općenito za slavenske jezike poput BERTića i CLASSLA-e. Pozitivni aspekti ovo alata su prisutnost lematizacije, mana istog je da lematizacija nije prilagođena za hrvatski jezik što ukazuje na pogreške lematizacije imena političara ili imena stranaka. [10]

Za korištenje Stanze u Pythonu prvo je potrebno instalirati biblioteku s komandom:

```
pip install stanza
```

Te potrebno je dodati import u kod:

```
import stanza
```

Kako bi se provelo prepoznavanje entiteta na hrvatskom jeziku potrebno je preuzeti resurse za hrvatski:

```
stanza.download('hr')
```

Stvaranje pipelinea je bitno jer omogućuje organizirano i sekvencijalno provođenje različitih jezičnih zadataka poput lematizacije i tokenizacije nad podacima. Stanza.Pipeline je glavna funkcija u biblioteci jer omogućuje stvaranje pipelinea.

```
nlp_stanza = stanza.Pipeline(lang='hr', processors='tokenize,pos,lemma')
```

BERTić je prilagođeni BERT model optimiziran za prepoznavanje entiteta u tekstovima na jezicima jugoistočne Europe. Za razliku od Stanze pokazao se učinkovitijim u prepoznavanju entiteta političara i stranaka u preuzetim člancima. Mana BERTića je manjak procesora 'lemma' koji služi za lematizaciju riječi, koja je korisna kad se žele imati čisti podaci za buduću obradu. [25]

Da bi se mogao koristiti BERTić potrebno je prvo preuzeti biblioteku za BERT:

```
pip install transformers
```

Prvi korak koji je potreban za korištenje BERTić modela za NER je import prethodne instalirane biblioteke te modula *AutoTokenizer, AutoModelForTokenClassification te pipeline*. Korištenje ovih modela neophodno je kako bi se učitao tokenizer koji omogućuje pretvaranje teksta u niz tokena te kako bi se učitao unaprijed treniran model za klasifikaciju tokena (B-PER, I-PER, B-ORG itd.).

Primjer inicijalizacije tokenizera te učitavanje unaprijed treniranog modela:

```
tokenizer = AutoTokenizer.from_pretrained("classla/bcms-bertic-ner")
model = AutoModelForTokenClassification.from_pretrained("classla/bcms-bertic-
ner")Inicijalizacija pipelinea je slična kao sa Stanzom koja omogućuje spajanje
tokenizacije, modela i predikcija:


nlp = pipeline("ner", model=model, tokenizer=tokenizer)
```

Jedan od izazova s BERTićem bio je rad s tokenima koji su fragmentirani (označeni prefiksom "##".) To je zahtijevalo dodatnu obradu kako bi se sastavili cjeloviti entiteti iz fragmenata:

```
merged_entities = []
    for entity_list in ner_results:
        for entity in entity_list:
            if entity['word'].startswith("##"):
                if merged_entities:
                    merged_entities[-1]['word'] += entity['word'][2:]
            else:
                merged_entities.append(entity)
```

CLASSLA je zadnji obrađeni model te model koji se pokazao najpreciznijim za prepoznavanje entiteta u hrvatskom jeziku pogotovo što se tiče organizacija kao što su političke stranke. CLASSLA je također ponudila integriranu mogućnost lematizacije što nije bio slučaj kod BERTić modela. Lematizacija je omogućila uklanjanje morfoloških oblika riječi (poput padeža) što je bilo važno za normalizaciju imena i naziva, osobito u kontekstu hrvatskog jezika s bogatim morfološkim varijacijama; čime se dodatno pojednostavila daljnje korištenje entiteta u izradi predviđanja parlamentarnih izbora [26].

Instalacija CLASSLA biblioteke:

```
pip install classla
```

Preuzimanje modela za hrvatski jezik te inicijalizacija :

```
classla.download(lang='hr')
nlp = classla.Pipeline(lang='hr', logging_level='ERROR',
processors='tokenize,pos,lemma,ner')
```

Primjer spremanja imena i prezimena koja pripadaju istoj osobi:

```
  if len(parts) >= 9:
        if parts[9].startswith("NER=B-PER") # Beginning of a person entity
            word, lemma = parts[1], parts[2]
            if prev_entity.startswith("I-PER"):
                # Merge if the previous entity is also part of the current entity
                merged_names[-1] += ' ' + lemma
            else:
                merged_names.append(lemma)
        prev_entity = parts
```

Nakon provedenih usporedba između Stanze, CLASSLE i BERTića, CLASSLA model se pokazao najpreciznijim i najpouzdanijim za prepoznavanje entiteta u političkim tekstovima na hrvatskom jeziku. Njegova integrirana lematizacija omogućila je uklanjanje morfoloških varijacija i postizanje veće preciznosti u prepoznavanju relevantnih političkih entiteta.

5. Predikcija rezultata izbora

U ovom poglavlju obradit će se ključni dio ovoga rada: implementacija modela za predviđanje rezultata parlamentarnih izbora 2024. godine u Republici Hrvatskoj koristeći dobivene entitete političara te stranaka iz članaka web portala N1, Večernji List te Index.hr iz razdoblja 1.1.2020. – 3.7.2020. Linearna regresija te algoritam slučajne šume korišteni su kao modeli za predikciju rezultata.

5.1. Priprema podataka za modeliranje

Priprema podataka bitan je korak kako bi se osigurala kvaliteta i upotrebljivost podataka u modelu koji će se koristiti za predikciju.

Podaci koji su se koristili za treniranje modela obuhvaćaju razdoblje od 1.1.2020 do 3.7.2020. (dan prije predizborne šutnje) .Modeli su razvijeni na temelju podataka izvučenih iz članaka objavljenih na N1, Index.hr i Večernjem listu. 3405 članaka preuzeto je iz ovih izvora; unaprijed je postavljen filter (upit eng. query prilikom pretraživanja članaka) pri struganju podataka kako bi se preuzeli samo oni članci čiji sadržaj je povezan s izborima te hrvatskom politikom. Model se trenirao koristeći broj ponavljanja imena stranke, koalicije ili političara te iste stranke ili koalicije u člancima u 2020. godini. Imena stranka, koalicija te imena političara dobiveni su koristeći NER pomoću CLASSLA Python biblioteke.

Nakon što je izrađen NER nad člancima bilo je potrebno napraviti distribuciju entiteta te ih grupirati u jedan zajednički entitet. Npr. Milanović, Zoran Milanović te Milanović Zoran spadaju u istu grupu „Milanović“. Ovaj korak je napravljen pomoću Python biblioteke *Stemmer*.

```
for original, stemmed_pojam in zip(pojmovnici, stemmed_pojmovi):
    name, count = original['name'], original['count']
    stemmed_name = stemmed_pojam[0]
    added = False
    for group in groups:
        if stemmed_name in group or group in stemmed_name:
            groups[group].append({"name": name, "count": count})
            added = True
            break
    if not added:
        groups[stemmed_name].append({"name": name, "count": count})
```

Nakon što su pojmovi grupirani, pomoću dictionaryja sačuvane su samo grupe sa željenim vrijednostima (političari te stranke) dok su ostale vrijednosti uklonjene, ručno se provjerila točnost grupiranja entiteta pomoću *Stemmera* te u rijetkim slučajevima, gdje se uvidjela pogreška, pojam bi se uklonio [27].

Primjer grupacije riječi „plenković“:


```
"plenković": [
      {
          "name": "Plenković",
          "count": 4724
      },
      {
          "name": "Andrej Plenković",
          "count": 1868
      },
      {
          "name": "Andreja Plenković",
          "count": 22
      },
      {
          "name": "moći Plenković",
          "count": 21
      },
      {
          "name": "Plenković ””",
          "count": 13
      },
      {
          "name": "“ Plenković",
          "count": 12
      },
      {
          "name": "pred Plenković",
          "count": 11
      },
      {
          "name": "Vlada Plenković",
          "count": 10
      },
      {
          "name": "premijer Plenković",
          "count": 8
      },
      {
          "name": "koalicija Plenković",
          "count": 7
      },
     ],
```


Zadnji korak prije implementacije modela za predikciju rezultata izbora bio je spojiti i zbrojiti sve pojmove za svaku grupu kojoj pripadaju.

5.2. Implementacija modela za predikciju

Jednom kad su se podaci grupirali i očistili, implantirana su dva modela strojnog učenja: linearna regresija i slučajna šuma. Svaki model nudi različiti pristup predikciji što omogućuje dublje razmatranje rezultata te omogućuje međusobnu usporedbu.

Model linearne regresije implementiran je pomoću scikit-learn biblioteke, koja omogućuje jednostavno treniranje i evaluaciju modela, koja se instalira putem komande:

```
pip install scikit-learn
```

Prije same implementacije modela za predikciju potrebno je dodati stranke/koalicije koje nisu bile prisutne na izborima 2020. godine u suprotnom dogodila bi se greška.

Korištenjem podataka iz 2020. godine, trenira se model linearne regresije koji povezuje broj spominjanja stranke (medijska prisutnost) s njenim stvarnim izbornim rezultatom.:

```
model = LinearRegression()
model.fit(X_train, y_train)
```

Na temelju treniranog modela, predviđaju se rezultati za 2024. godinu. Predikcije se zaokružuju na cijele brojeve kako bi se uskladile s brojem mandata. Neke stranke, poput nacionalnih manjina, uvijek imaju isti broj mandata (8) te se za tu stranku osigurao fiksni broj te se nije vršila predikcija.

Koristeći greške modela s podacima iz 2020. korigiraju se rezultati za 2024. kako bi predviđanje bilo točnije.

```
df_2020['predicted_votes_2020'] = model.predict(df_2020[[count_column_2020]])
df_2020['error'] = df_2020[result_column] - df_2020['predicted_votes_2020']

average_error = df_2020['error'].mean()

df_2024['corrected_predicted_votes_2024'] = df_2024.apply(
    lambda row: row['predicted_votes_2024'] if row[name_column] in special_cases
else row['predicted_votes_2024'] + average_error,
    axis=1
)
```

Zadnji korak je normalizacija rezultata kako bi zbroj svih mandata bio 151, normalizacija se vršila koristeći skalar [28].

Slučajna šuma je složeniji model koji koristi više stabala odlučivanja za kreiranje ansambla predikcija. Svako stablo u ansamblu donosi svoju prognozu, rezultat se temelji na prosjeku ili većinskom glasu. Model slučajne šume također je implementiran je pomoću scikit-learn biblioteke.

U ovom koraku pripremaju se podaci iz 2020. godine za obuku modela. Iz podataka iz 2020. godine filtriraju se samo oni redovi koji sadrže stranke koje su prisutne i u podacima za 2024. godinu. Ovi redovi će poslužiti za obuku modela kako bi se predvidio broj mandata na temelju broja ponavljanja pojma stranke i njezinih političara.

```
df_train = df_2020[df_2020[name_column].isin(df_2024[name_column].values)]
X_train = df_train[[count_column_2020]]
y_train = df_train[result_column]
```

U ovom koraku vrši se inicijalizacija modela *RandomForestRegressor* i koristi se *GridSearchCV* kako bi se pronašao najbolji model.

```
rf = RandomForestRegressor(random_state=100)
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [1, 10],
    'min_samples_split': [2, 8],
    'min_samples_leaf': [1, 2]
}
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3, n_jobs=4,
verbose=2)
grid_search.fit(X_train, y_train)
best_rf = grid_search.best_estimator_
df_2024['predicted_votes_2024'] = best_rf.predict(X_2024)
```

- rf je inicijalizirani RandomForestRegressor model.
- param_grid definira različite parametre koji se koriste za GridSearchCV
- grid_search.fit(X_train, y_train) trenira model koristeći podatke iz 2020. godine.
- best_rf sadrži najbolji model prema rezultatima GridSearchCV.
- df_2024['predicted_votes_2024'] sadrži predikcije broja mandata za 2024. godinu koristeći najbolji model

Ovdje se prilagođavaju predikcije za specifične stranke koje imaju unaprijed definirane brojeve mandata. Nacionalne manjine imaju fiksnu vrijednost mandata. Također, ako ta stranka nije prisutna u df_2024, dodajemo ju s fiksnim brojem mandata [29].

```
for name, fixed_result in special_cases.items():
    if name in df_2024[name_column].values:
        df_2024.loc[df_2024[name_column] == name, 'predicted_votes_2024'] = fixed_result
    else:
        new_row = pd.DataFrame({
            name_column: [name],
            count_column_2020: [0],
            'predicted_votes_2024': [fixed_result],
            'is_new': [1]
        })
        df_2024 = pd.concat([df_2024, new_row], ignore_index=True)
```

5.3. Rezultati i usporedba točnosti treniranih modela

U ovom dijelu rada predstavljeni su rezultati predikcija osvojenih mandata u parlamentarnim izborima 2024. u Republici Hrvatskoj, koji su postignuti pomoću linearne regresije i slučajne šume. Analizirana je točnost oba modela na temelju podataka iz 2020. godine, te su rezultati uspoređeni kako bi se procijenila njihova učinkovitost.

Rezultati linearne regresije pokazali su da model može predvidjeti rezultate za stranke koje imaju jasnu linearnu vezu između medijske prisutnosti i izbornih rezultata. Međutim, kod stranaka s manjom ili nekonzistentnom medijskom prisutnošću, model je imao tendenciju podcjenjivanja ili precjenjivanja stvarnih rezultata, što je ukazivalo na ograničenja linearne regresije u složenijim scenarijima. Model se pokazao da je točnost predikcije proporcionalna

količini podataka te je ista bila slabija kad bi se koristili samo podaci samih stranaka te je preciznost rasla dodavanjem podataka predsjednika i ostalih političara stranke.

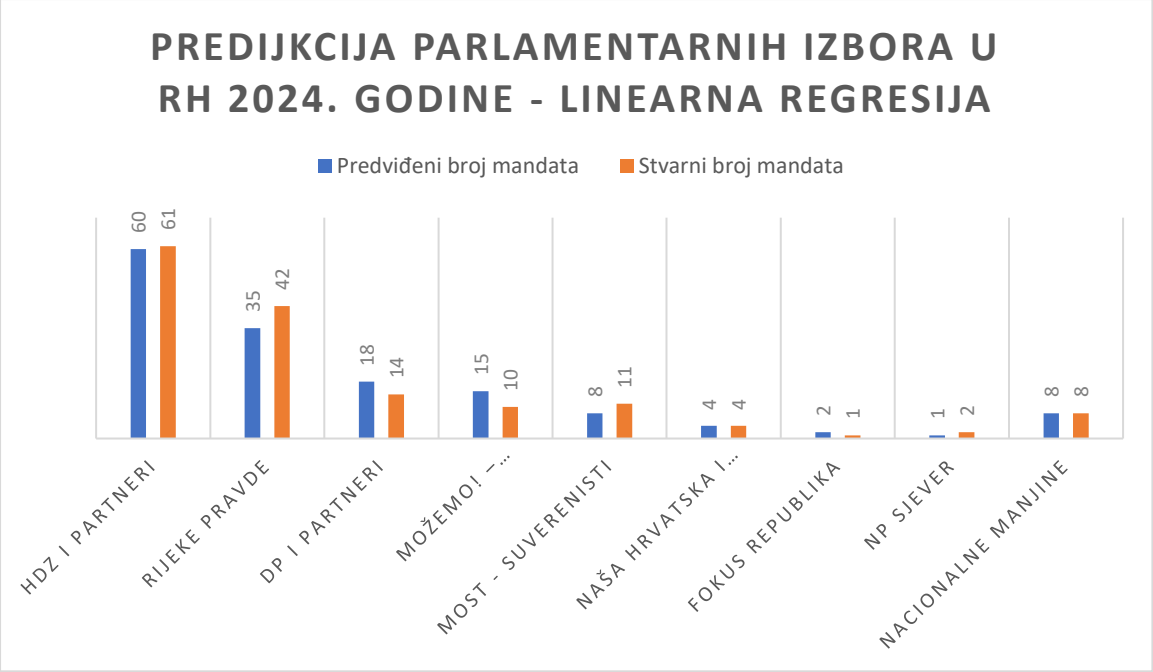
Tablični prikaz podataka predviđanja osvajanja mandata s linearnom regresijom:

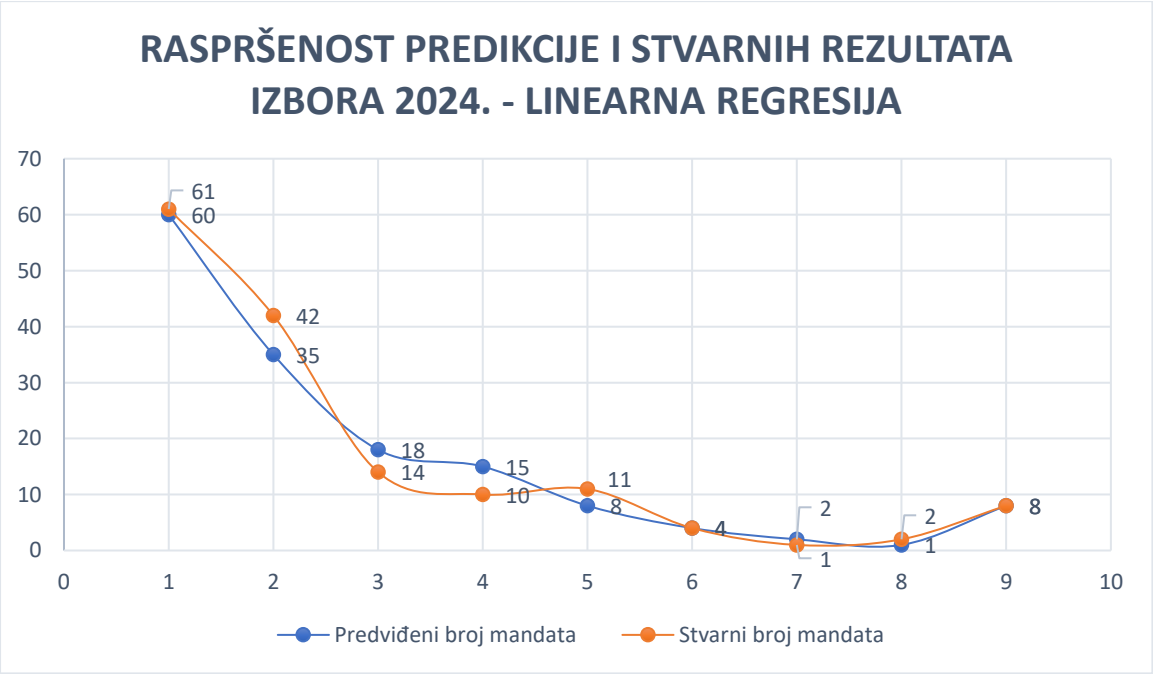
Tablica 3 Prikaz predikcije broja mandata koristeći linearnu regresiju

Ime stranke/kolacije	Broj ponavljanja u člancima	Predviđeni broj mandata	Stvarni broj mandata	Pogreška
HDZ i partneri	24921	60	61	-1
Rijeke pravde	14333	35	42	-7
DP i partneri	7118	18	14	+4
Možemo! – politička platforma	5797	15	10	+5
Most - Suverenisti	2991	8	11	-3
Naša Hrvatska i drugi	1112	4	4	0
Fokus republika	376	2	1	+1
NP Sjever	138	1	2	-1
Nacionalne manjine	-	8	8	0

Vidljivo je kako se najveći postotak pogreške linearne regresije nalazi među onim strankama koji se nalaze u sredini te je model točniji za one stranke koje se nalaze na ekstremima (HDZ, Naša Hrvatska, Fokus te Np Sjever).

Srednja kvadratna pogreška (eng. Mean square error, MSE) korištenog modela iznosi: 13.56





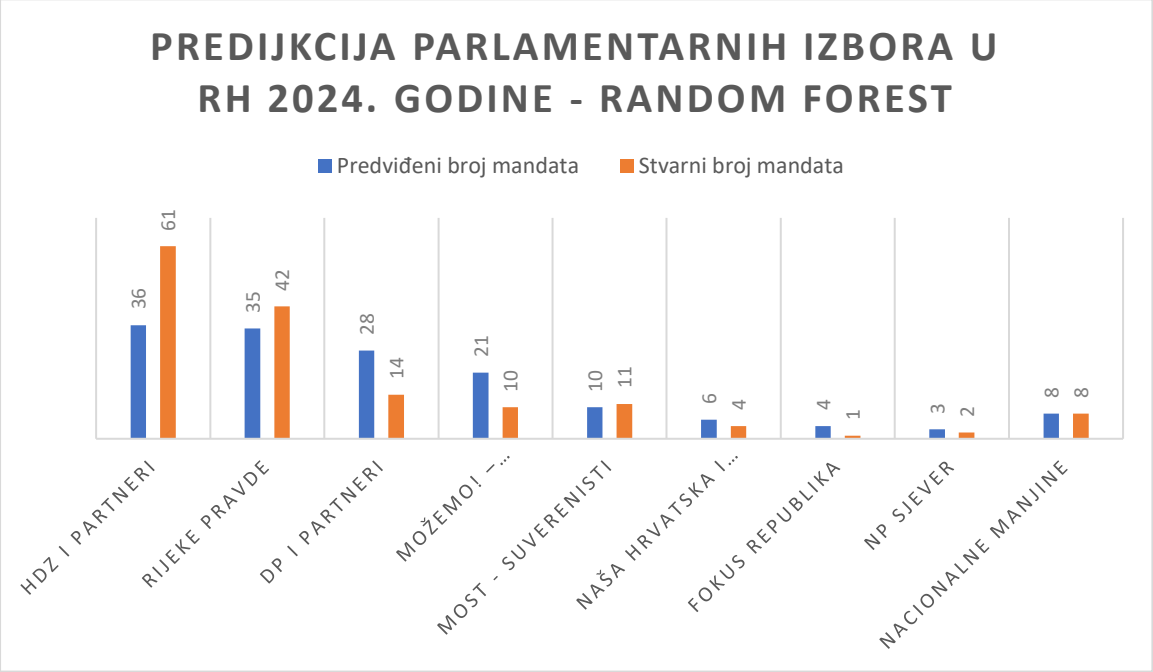
Model slučajne šume za razliku od onog iz linearne regresije slabije je predvidio osvajanje mandata svake stranke na izborima 2024. Korijen srednje kvadratne pogreške ovog modela je iznosio 10.57 (eng. Root Mean Square Error, RMSE). Model je koristio 100 i 200 stabala u šumi te je maksimalna dubina svakoga stabla bila 1 i 10, testirane vrijednosti broja uzoraka za podijelu čvora bili su 2 i 8 te broj uzoraka prisutan u listu je bio 1 i 2.

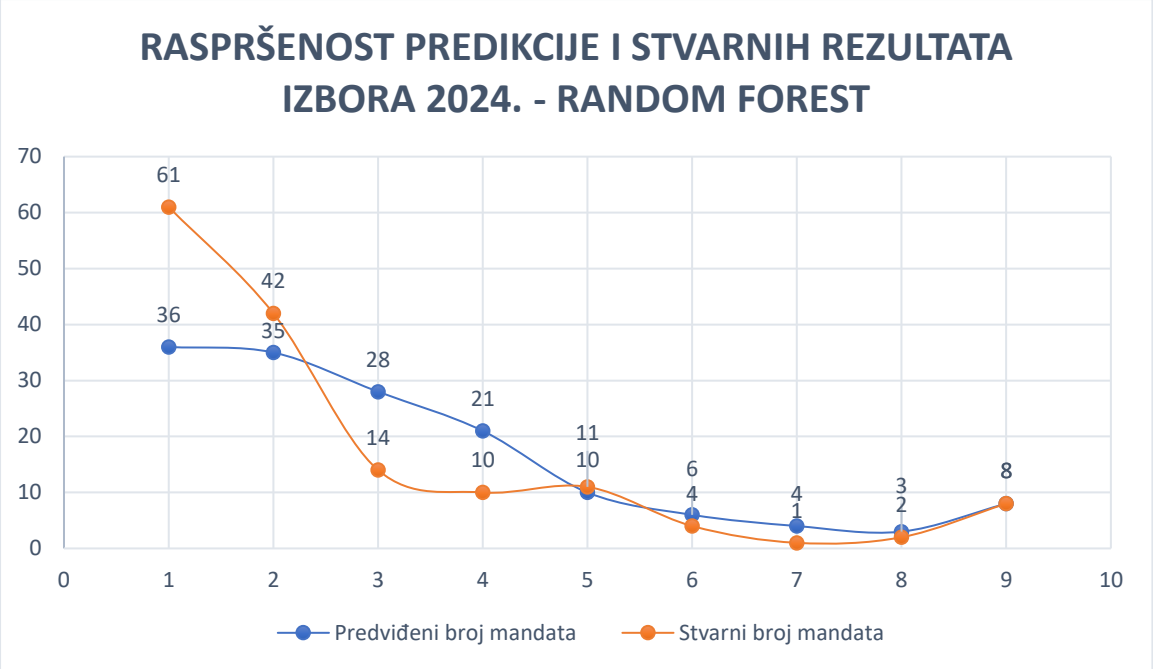
Tablični prikaz predikcije broja mandata s algoritmom slučajne šume :

Tablica 4 Prikaz predikcije broja mandata koristeći algoritam slučajne šume

Ime stranke/koalicije	Broj ponavljanja u člancima	Predviđeni broj mandata	Stvarni broj mandata	Pogreška
HDZ i partneri	24921	36	61	-25
Rijeke pravde	14333	35	42	-7
DP i partneri	7118	28	14	+14
Možemo! – politička platforma	5797	21	10	+11
Most - Suverenisti	2991	10	11	-1
Naša Hrvatska i drugi	1112	6	4	+2
Fokus republika	376	4	1	+3
NP Sjever	138	3	2	+3
Nacionalne manjine	-	8	8	0

Vidljivo je kako za razliku od linearne regresije koja je imala sveukupno manje vrijednosti pogreške, algoritam slučajne šume je dodijelio puno veći broj mandata srednjim te manjim strankama (osim Mosta) te je zakinuo za čak 25 mandata HDZ i njihove partnere.




RASPRŠENOST PREDIKCIJE I STVARNIH REZULTATA IZBORA 2024. - RANDOM FOREST
70
60
50
40
30
20
10
0
0
1
2
3
4
5
6
7
8
9
10
61
42
36
35
28
21
14
10
11
10
6
4
4
1
3
2
8
Predviđeni broj mandata
Stvarni broj mandata

6. Zaključak

Ovaj završni rad imao je kao primarni cilj procijeniti točnost jednostavnijih metoda strojnog učenja poput linearne regresije i slučajne šume u predviđanju rezultata parlamentarnih izbora u Hrvatskoj u 2024, koristeći podatke prikupljene primjenom postupaka web struganja internetskih portala.

Korišteni podaci obuhvatili su 3405 članaka iz perioda 1.1.2020 – 3.7.2020. za treniranje modela predikcije te 7382 članaka iz perioda 1.1.2024 – 15.4.2024. za predviđanje. Model je treniran tako da se iz članaka broje spominjanja imena političkih stranaka, koalicija i političara, koristeći prepoznavanje imenovanih entiteta uz pomoć CLASSLA Python biblioteke.

Analiza dobivenih rezultata je otkrila da linearna regresija daje pouzdana predviđanja za stranke s velikom ili relativno malom medijskom pokrivenošću. U slučaju srednje medijske pokrivenosti rezultati ovog modela su doveli do precjenjivanja ili podcjenjivanja odnosno na realne rezultate stranaka na izborima. Slučajna šuma s druge strane, dala je manje precizna predviđanja od linearne regresije, iako se pri odabiru modela mislilo da će rezultati biti točniji pošto se radi o sofisticiranijem modelu. Posebice model je podcijenio osvajanje broja mandata za HDZ i njihove partnere za 25 mandata manje.

Kako bi rezultati bili točniji za neka buduća istraživanja moglo bi biti od koristi korištenje složenijih modela dubokog učenja poput rekurentnih neuronskih mreža (RNN); osim toga uključivanje analize sentimenta u proces predviđanja dalo bi kvalitetnije podatke za treniranje modela. U ovom radu koristili su se sirovi podaci od kojih se nije znalo kakvo je spominjanje: pozitivno, negativno ili neutralno pa se samo predviđanje baziralo na broj spominjanja.

Literatura

[1] Nehal, »The Ultimate Guide to Web Scraping: Tools, Techniques, and Use Cases,« 26. Prosinac 2023.. [Mrežno]. Available: https://www.promptcloud.com/blog/the-ultimate-guide-to-web-scraping-tools-techniques-and-use-cases/. [Pokušaj pristupa 6 Rujan 2024.].

[2] V. M. A. &. B. S. Orešković, »In Central European Conference on Information and Intelligent Systems,« *Towards Computational Content Analysis of Crises-Related News in Electronic Media,* pp. 407-416, 2023.

[3] S. M.-I. S. M. M. &. M. A. Beliga, »In The Covid-19 pandemic as a challenge for media and communication studies,« *Natural language processing and statistic: the first six months of the COVID-19 infodemic in Croatia.,* pp. 78-92, 2022.

[4] »Beautiful Soup Documentation,« 2020.. [Mrežno]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. [Pokušaj pristupa 20 Studeni 2023.].

[5] T. Keary, »Technopedia,« 20 Lipanj 2024.. [Mrežno]. Available: https://www.techopedia.com/definition/5212/web-scraping.

[6] R. Nayak, »Medium,« 16 Veljača 2019.. [Mrežno]. Available: https://medium.com/ymedialabs-innovation/web-scraping-using-beautiful-soup-and-selenium-for-dynamic-page-2f8ad15efe25.

[7] »Selenium,« 1 Siječanj 2024.. [Mrežno]. Available: https://www.selenium.dev/.

[8] L. Richardson, »Python Package Index,« 17 Siječanj 2024.. [Mrežno]. Available: https://pypi.org/project/beautifulsoup4/.

[9] »IBM,« 8 Kolovoz 2024.. [Mrežno]. Available: https://www.ibm.com/topics/named-entity-recognition.

[10] »Stanza – A Python NLP Package for Many Human Languages,« 2020.. [Mrežno]. Available: https://stanfordnlp.github.io/stanza/ner_models.html.

[11] »EkonInfoChecker,« [Mrežno]. Available: https://ekoninfochecker.efri.uniri.hr/?p=735.

[12] S. r. i. z. j. v. i. tehnologije, »Clarin.si,« [Mrežno]. Available: https://www.clarin.si/info/k-centre/.

[13] m. gupta_omg, »GeeksForGeeks,« 7 Kolovoz 2024.. [Mrežno]. Available: https://www.geeksforgeeks.org/ml-linear-regression/.

[14] Sumbatilinda, »Medium,« 26 Ožujak 2024.. [Mrežno]. Available: https://medium.com/@sumbatilinda/random-forests-regression-by-example-1baa062506f5.

[15] »N1 Hrvatska,« 2024.. [Mrežno]. Available: https://n1info.hr/.

[16] T. P. Petra Kovačević, »FPZG,« 15 Studeni 2021.. [Mrežno]. Available: https://www.fpzg.unizg.hr/images/50022061/ms%20vol13%20br25_ukupni.pdf.

[17] »Index.hr,« 2024.. [Mrežno]. Available: https://www.index.hr/.

[18] M. B. F. Check, »Media Bias/ Fact Check,« 22 Lipanj 2024.. [Mrežno]. Available: https://mediabiasfactcheck.com/index-hr-croatia-bias/.

[19] »Večernji List,« 2024.. [Mrežno]. Available: https://www.vecernji.hr/.

[20] »Euro topics,« [Mrežno]. Available: https://www.eurotopics.net/en/148851/veernji-list.

[21] K. Edirisinghe, »Medium,« 30 Srpanj 2021.. [Mrežno]. Available: https://medium.com/@kaushalyaedirisinghe/how-to-solve-the-issue-when-chrome-browser-and-chrome-driver-have-different-versions-d98d612c6fa6.

[22] T. Valjavec, »Zemris FER,« 2023.. [Mrežno]. Available: https://www.zemris.fer.hr/~sgros/students/diploma_thesis/bakula_silvana_diplomski.pdf.

[23] »Wikipedia,« 2024.. [Mrežno]. Available: https://en.wikipedia.org/wiki/2024_Croatian_parliamentary_election#/media/File:2024_Croatian_parliamentary_election_map.svg.

[24] D. Jurmanović, »Portal Veterani,« 18 Travanj 2024.. [Mrežno]. Available: https://portal-veterani.info/izbori-2024-konacni-rezultati/.

[25] N. a. L. D. Ljubevšić, »Hugging Face,« Travanj 2021.. [Mrežno]. Available: https://huggingface.co/classla/bcms-bertic.

[26] L. T. a. N. Ljubešić, »Python Package Index,« 2023.. [Mrežno]. Available: https://pypi.org/project/classla/.

[27] R. Boulton, »Python Package Index,« 16 Siječanj 2023.. [Mrežno]. Available: https://pypi.org/project/PyStemmer/.

[28] »scikit-learn,« 2024.. [Mrežno]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

[29] »scikit-learn,« 2024.. [Mrežno]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

Popis tablica

Popis slika

Popis priloga

Korišteni kod te ostale datoteke: https://gitlab.com/Dami01/prediction-of-the-parliamentary-elections-in-the-republic-of-croatia