UNIVERZITET U BEOGRADU

FAKULTET ORGANIZACIONIH NAUKA

SEMINARSKI RAD

Analiza i klasifikacija recenzija

Profesor:

Dr Dragan Đurić

Student:

Anđela Milićević 2025/3826

Link ka github-u: https://github.com/andjelaiteh/fake-reviews-clojure

BEOGRAD, 2026.

Sadržaj

Popis slika

1.Uvod

Razvoj digitalnih tehnologija značajno je doprineo u razvoju onlajn sistema za elektronsku kupovinu. Pojava brojnih platformi za onlajn kupovinu kao što su Temu, AliExpress, Shein i ostali, omogućila je korisnicima jednostavan pristup širokom spektru proizvoda po pristupačnim cenama. Zato se danas sve veći broj ljudi opredeljuje za onlajn kupovinu.

Na ovakvim platformama korisnici mogu da pronadju različite ponude proizvoda, da uporede cene, karakteristike, da provere dostupnost, što sve spada u kriterijume za kupovinu. S obzirom na to da ne mogu fizički da vide i ocene proizvod, često kao za jedan od najbitnijih kriterijuma smatraju recenzije i ocene proizvoda i oni predstavljaju značajan izvor informacija pri kupovini.

Međutim, s obzirom na to da uglavnom ne postoji adekvatna kontrola istinitosti recenzija, danas postoji i veliki broj lažnih recenzija koje za cilj imaju manipulaciju kupovinom.

Cilj ovog rada je analiza recenzija sa fokusom na detekciju lažnih.

2.Korišćene tehnologije i alati

U okviru ovog rada korišćen je programski jezik Clojure. On pripada grupi funkcionalnih jezika i izvršava se na Java virtuelnoj mašini (engl. Java Virtual Machine-JVM). Njegov funkcionalan pristup je pogodan za analiziranje zbog svog jasnog i deklarativnog stila pisanja koda.

Kao razvojno okruženje korišćen je IntelliJ IDEA, a za upravljanje projektom i zavisnostima Leiningen. Leiningen je alat koji omogućava jednostavno organizovanje projekta, pokretanje REPL-a i učitavanje spoljnih biblioteka. Na taj način je značajno olakšan razvoj i testiranje aplikacije. Od spoljnih biblioteka korišćene su clojure.data.csv, clojure.java.io i midje za testiranje, koje će u narednim poglavljima biti opisane.

Takodje, korišćeno je mašinsko učenje koje je realizovano kroz implementaciju logističke regresije nakon čega je urađena evaluacija performansi.

3.Metologija rada

Prvi korak u realizaciji ovog projekta bio je pronalazak odgovarajućeg dataset-a. Kao izvor informacija korišćen je Kaggle, onlajn platforma za data-sciences i mašinsko učenje gde se nalazi veliki broj javno dostupnih dataset-ova pogodnih za analizu. U ovom radu korišćen je dataset ,,Fake Reviews Dataset ” koji je u csv formatu i može se videti na linku https://www.kaggle.com/datasets/thearijitdas/fake-reviews-dataset.

Nakon učitavanja dataseta u projekat, neophodno je bilo sredjivanje podataka kao i dodavanje novih kolona koje su potrebne za analizu.

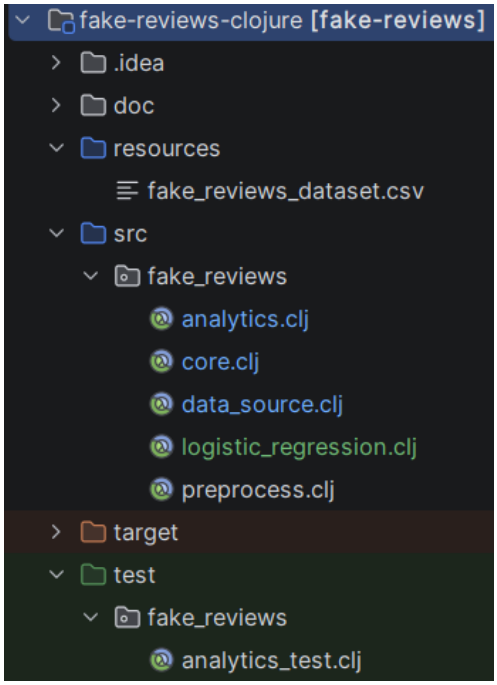
Analiza je sprovedena sa ciljem boljeg razumevanja podataka. Implementirane su različite funkcije za ocene, tekst i kategorije recenzija.

Nakon analize karakteristika recenzija, implementiran je model logističke regresije. Ovaj model je korišćen kako bi se izvršila klasifikacija na fake i real recenzije. Pre implementacije algoritma bilo je neophodno uraditi pripremu podataka i podelu dataseta na deo za trening i deo za testiranje.

Na samom kraju uradjena je evaluacija performansi modela i analiza rezultata. Za procenu uspešnosti korišćene su metrike tačnost (engl. Accuracy), preciznost (engl. Precision), odziv (engl. Recall) i F1 metrika.

4.Struktura projekta

Osnovna struktura projekta je podeljena na tri dela, kao što je prikazano na sledećoj slici, src, resources i test.



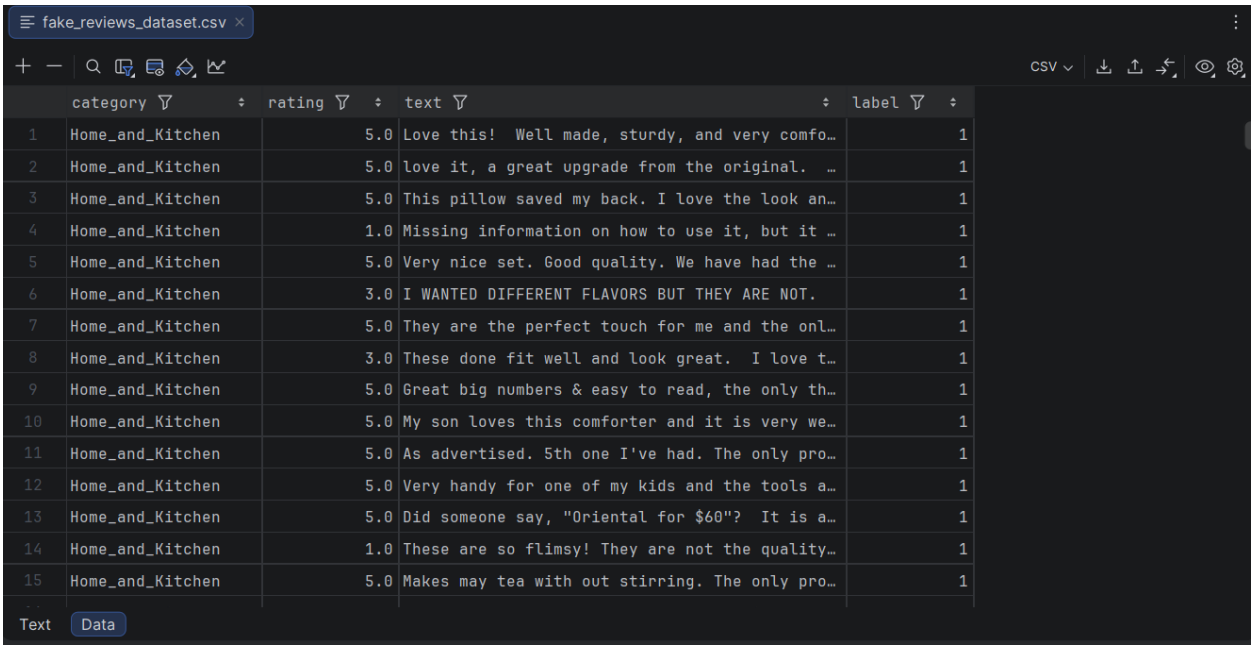
Slika 1 - Struktura projekta

Izvorni kod aplikacije smešten je u direktorijum src. U njemu se nalazi implementacija svih funkcionalnosti aplikacije. U okviru njega, kod je dodatno organizovan u više namespace-ova. Namespace predstavlja mehanizam za bolju organizaciju koda u logičke celine i sprečava konflikte između različitih delova programa.

U ovoj aplikaciji src sadrži sledeće namespace-ove, koji će detaljno biti opisani u narednim poglavljima:

1. data-source.clj – učitavanje csv fajla,
2. preprocess.clj – priprema podataka za analizu,
3. analytics.clj – analiza podataka,
4. logistic-regression – logistička regresija,
5. core.clj – pokretanje aplikacije.

Direktorijumu resources koristi se za čuvanje spoljnih resursa koji se koriste u projektu. U njemu se nalaze ulazni podaci u csv formatu, koji su prikazani na sledećoj slici.


fake_reviews_dataset.csv

	category	rating	text	label
1	Home_and_Kitchen	5.0	Love this! Well made, sturdy, and very comfo…	1
2	Home_and_Kitchen	5.0	love it, a great upgrade from the original. …	1
3	Home_and_Kitchen	5.0	This pillow saved my back. I love the look an…	1
4	Home_and_Kitchen	1.0	Missing information on how to use it, but it …	1
5	Home_and_Kitchen	5.0	Very nice set. Good quality. We have had the …	1
6	Home_and_Kitchen	3.0	I WANTED DIFFERENT FLAVORS BUT THEY ARE NOT.	1
7	Home_and_Kitchen	5.0	They are the perfect touch for me and the onl…	1
8	Home_and_Kitchen	3.0	These done fit well and look great. I love t…	1
9	Home_and_Kitchen	5.0	Great big numbers & easy to read, the only th…	1
10	Home_and_Kitchen	5.0	My son loves this comforter and it is very we…	1
11	Home_and_Kitchen	5.0	As advertised. 5th one I've had. The only pro…	1
12	Home_and_Kitchen	5.0	Very handy for one of my kids and the tools a…	1
13	Home_and_Kitchen	5.0	Did someone say, "Oriental for $60"? It is a…	1
14	Home_and_Kitchen	1.0	These are so flimsy! They are not the quality…	1
15	Home_and_Kitchen	5.0	Makes may tea with out stirring. The only pro…	1

Text Data

Slika 2- fake-reviews-dataset

Ovde vidimo da se dataset sastoji od četiri kolone, kategorija, ocena, tekst i labela koja predstavlja da li je recenzija fake ili real (1 ukoliko je fake, 0 ukoliko je real).

U okviru test direktorijuma sprovedeno je testiranje korišćenje midje okvira, koje će kasnije biti objašnjeno.

5.Implementacija

U ovom poglavlju biće prikazana kompletna implementacija sistema kroz objašenje svih gore navedenih namespace-ova i njihovih funkcija.

5.1 Namespace data-source

U ovom namespace-u vrši se učitavanje csv fajla.

```
1  (ns fake-reviews.data-source  1 usage  andjela
2    (:require [clojure.java.io :as io]
3              [clojure.data.csv :as csv]))
4
5
6  ;;ucitavanje csv fajla
7  (defn read-csv [csvname]  1 usage  andjela
8    (with-open [r (io/reader (io/resource csvname))]
9      (doall (csv/read-csv r))))
```

Slika 3 - data-source namespace

Na početku se nalazi naziv namespace-a (ns fake-review.data-source.clj).

U okviru :require navodimo sve spoljne biblioteke koje će biti korišćene u tom namespace-u. Ovde su to biblioteke clojure.java.io koja se koristi za rad sa fajlovima, čitanje, pisanje i biblioteka clojure.data.csv koja se koristi za učitavanje csv fajla.

Takodje možemo da koristimo :as kako bi u nastavku koda umesto punog naziva biblioteke mogli da koristimo skraćenice koje navedemo (io i csv).

U Clojure-u funkcije definišemo pomoći defn. Nakon defn pišemo ime funkcije a u uglastim zagradama ulazne argumente funkcije.

Funkcija read-csv kao argument prima naziv csv fajla.

io/resource u okviru resursa trazi fajl sa nazivom koji je prosledjen kao argument, a io/reader otvara fajl za citanje.

Pomoću csv/read-csv vrši se čitanje csv fajla i parsiranje redova u lazy sekvence redova. Lazy znači da se vrednosti ne računaju odma nego tek kada su potrebne. Kako se fajl ne bi zatvorio pre nego što se parsiraju svi redovi neophodno je koristi doall koji ih sve parsira odmah. Nako toga with-open zatvara fajl nakon što se završi blok za čitanje.

5.2 Namespace preprocess

Pošto smo nakon učitavanja csv fajla, dobili sekvence redova potrebno ih je parsirati u mape. Za to je napravljena funkcija transform-rows-to-maps, prikazana na sledećoj slici, koja kao argument prima sekvence redova koje pretvara u mape.

Mapa u Clojure-u predstavlja kolekciju ključ-vrednost gde svaki ključ identifikuje svoju vrednost. Ovde nazive kolona, category, text, rating i label pretvaramo u ključeve mape, a njihove vrednosti predstavljaju redove koje smo već učitali u sekvence redova.

```
(defn transform-rows-to-maps [rows]  1 usage  andjela
  (let [headers (map keyword (first rows))]
    (map #(zipmap headers %) (rest rows))))
```

Slika 4 - Funkcija transform-rows-to-maps

Let se koristi za definisanje lokalne promenljive [naziv vrednost] koja se može koristiti samo unutar let bloka. Ovde definišemo lokalnu promenljivu headers kojoj se dodeljuje vrednost prvog reda u prosledjenoj sekvenci redova odnosno nazivi kolona.

Kroz nazive kolona prolazimo pomoću map. Map funkcija se sastoji od funkcije i kolekcije.

(map funkcija kolekcija). Funkcija map prolazi kroz navedenu kolekciju i primenjuje navedenu funkciju nad svakim elementom kolekcije.

Nakon kreiranja mape za nazive ključeva ponovo pomoću map funkcije prolazimo kroz sve redove i pomoću zipmap spajamo ključeve sa vrednostima.

% označava trenutni red, a #() je skraćenica za anonimnu funkciju odnosno funkciju koja se definiše i koristi u okviru neke druge funkcije najčešće kao njen argument.

Na sledećoj slici prikazane su funkcije koje rade sredjivanje podataka.

```
;;parsiranje rating-a iz string u double
(defn parse-rating [s]  1 usage  andjela
  (try
    (Double/parseDouble (str/trim s))
    (catch Exception _ nil)))

;;racunanje duzine teksta recenzije
(defn text-length [s]  1 usage  andjela
  (if (string? s) (count s) 0))

;;sredjivanje kategorija
(defn prepare-category [category]  1 usage  andjela
  (str/trim
    (str/replace category "_" " ")))

;;dodavanje atributa rating-num, text-lenght i label-name(real ili fake
(defn prepare-reviews [row]  1 usage  andjela
  (assoc row
    :rating-num (parse-rating (:rating row))
    :text-len (text-length (:text row))
    :label-name (if (= (:label row) "1")
                  "fake"
                  "real")
    :category (prepare-category (:category row))))
```

Slika 5- Sredjivanje podataka

Funkcija parse-rating prima string s za koji prvo okloni razmake na početku i kraju stringa (str/trim), a zatim ga parsira u double (Double/parseDouble). Ukoliko pokušamo da parsiramo nešto što nije broj dolazi do greške i prekida programa. Zato koristimo try-catch blok kako bismo uhvatili grešku Exception. Ukoliko se prosledi nešto što nije string to postaje nil vrednost.

Funkcija text-len računa dužinu teksta recenzije tako što se prvo proveri u okviru if da li je ulazni argument string i ukoliko jeste pomoću funkcije count računa se njegova dužina a ukoliko nije dužina je 0.

Dakle, if prima tri argumenta uslov, izraz šta se izvršava ukoliko je uslov truthy i izraz šta se izvršava ako je uslov false ili nil.

Funkcija prepare-category pomoću str/replace zameni svaki _ znak sa razmakom i zatim ga trimuje kao što je već objašnjeno.

Na kraju ovog namespace-a je cilj da dobijemo mape redova koje će biti pogodne za dalje analize.

Završnu pripremu radimo u funkciji prepare-reviews. Funkcija assoc se koristi za izmenu ili dodavanje parova ključ vrednost u okviru mape. Korišćenjem prethodno napisanih funkcija dodajemo par ključ - vrednost rating-num - double vrednost ocene i text-len – dužina teksta recenzije. Takodje dodajemo ključ label-name koji će biti fake ukoliko label reda ima vrednost 1 ili 0 ukoliko ima vrednost 1 i menjamo vrednost kategorije odnosno njen naziv.

5.3 Namespace analitics

U ovom namespace-u realizovane su funkcije za analizu podataka.

Na sledećoj slici prikazana je funkcija avg koja se koristi za izračunavanje srednje vrednosti. Ovu funkciju je bilo neophodno prvo napisati jer će biti korišćena u okviru narednih funkcija.

```
(defn avg [data]  5 usages  andjela
  (let [data (remove nil? data)]
    (when (seq data)
      (/ (reduce + data) (double (count data))))))
```

Slika 6- Funkcija za računanje srednje vrednosti

Funkcija kao ulazni parametar prima kolekciju podataka iz koje prvo izbacuje sve nil vrednosti, a zatim pomoću when koristeći funkciju seq proveravamo da li kolekcija nije prazna. Ukoliko nije nastavljamo izvršavanje sledeće naredbe (/ (reduce + data) (double(count data)))). Ovde računamo prosek tako što pomoću reduce saberemo sve clanove kolekcije, pomoću count izbrojimo i na kraju podelimo. S obzirom na to da count data vraća integer potrebno je uraditi kastovanje u double kako bismo dobili kao rezultat double vrednost.

Kada ovu funkciju primenimo na sve ocene recenzija, dobijamo prosečnu ocenu svih recenzija, što je prikazano na narednoj slici. To radimo pomoću map funkcije koja prolazi kroz sve elemente prosledjene kolekcije podataka, uzima vrednost ključa rating-num i vraća ih kao sekvencu koja se prosleđuje avg funkciji.

```
(defn average-rating [data]  3 usages  andjela
  (avg (map :rating-num data)))
```

Slika 7- Funkcija za računanje prosečne ocene svih recenzija

Kako bismo izračunali prosečnu ocenu samo za fake ili real recenzije pišemo funkcije koje iz dataseta vraćaju samo sve fake ili sve real recenzije. One su prikazane na sledećoj slici.

```
;;vraca sve fake
(defn get-fake-reviews [data]  6 usages  andjela
  (filter #(= (:label-name %) "fake") data))

;;vraca sve real recenzije
(defn get-real-reviews [data]  6 usages  andjela
  (filter #(= (:label-name %) "real") data))
```

Slika 8- Funkcije koje vraćaju sve fake ili real recenzije

U okviru ovih funkcija koristimo funkciju filter koja prvo prima uslov a zatim kolekciju podataka nad kojom se primenjuje taj uslov. U okviru uslova definisana je anonimna funkcija koja za svaki podatak iz kolekcije uzima vrednost ključa label-name i proverava da li je fake ili real u zavisnosti od funkcije. Uslov se primenjuje na kolekciju podataka koja je prosledjena kao argument i kao rezultat funkcije vraća sve koji zadovoljavaju uslov.

Nakon toga, kada ovde funkcije prosledimo funkciji average-rating, kao što je prikazano na sledećoj slici, kao rezultat dobijamo prosečnu ocenu svih fake i svih real recenzija.

```
;;prosecna ocena real recenzija
(defn average-rating-real [data]  no usages  andjela
  (average-rating (get-real-reviews data)))

;;prosecna ocena fake recenzija
(defn average-rating-fake [data]  no usages  andjela
  (average-rating (get-fake-reviews data)))
```

Slika 9 - Funkcije za računanje prosečne ocene svih fake i svih real recenzija

Isto tako je uradjeno i za dužinu teksta recenzije, prikazano na narednoj slici.

```
;;prosecna duzina recenzija
(defn average-text-length [data]  3 usages  andjela
  (avg (map :text-len data)))

;prosecna duzina teksta fake recenzija
(defn average-text-length-fake [data]  no usages  andjela
  (average-text-length (get-fake-reviews data)))

;prosecna duzina teksta real recenzija
(defn average-text-length-real [data]  no usages  andjela
  (average-text-length (get-real-reviews data)))
```

Slika 10 - Funkcije ra računanje prosečne dužine teksta recenzija

Na sledećim slikama prikazane su funkcije za računanje broja recenzija po kategorijama.

```
;broj recenzija po kategoriji
(defn reviews-count-by-category [data]  3 usages  andjela
  (frequencies (map :category data)))

;broj fake recenzija po kategoriji
(defn fake-count-by-category [data]  1 usage  andjela
  (reviews-count-by-category (get-fake-reviews data)))

;broj real recenzija po kategoriji
(defn real-count-by-category [data]  1 usage  andjela
  (reviews-count-by-category (get-real-reviews data)))
```

Slika 11- Funkcije za računanje broja recenzija po kategorijama

Korišćena je već objašnjena map funkcija koja prolazi kroz mapu i uzima vrednosti za ključ kategorija, a zatim se pomoću funkcije frequencies računa broj pojavljivanja za svaku vrednost.

Na sledećoj slici prikazane su funkcije za izračunavanje tri kategorije sa najvećim brojem fake ili real recenzija korišćenjem prethodno implementiranih funkcija. Korisšćena je funkcije sort-by za sortiranje u opadajućem redosledu nakon čega funkcijom take uzimamo prve tri para ključ vrednost.

```
;;top 3 fake kategorije
(defn top-fake-categories [data]  8 usages  andjela
  (take 3 (sort-by val > (fake-count-by-category data))))

;;top 3 real kategorije
(defn top-real-categories [data]  no usages  andjela
  (take 3 (sort-by val > (real-count-by-category data))))
```

Slika 12 - Funkcije za ucitavanje top 3 kategorije po broju recenzija

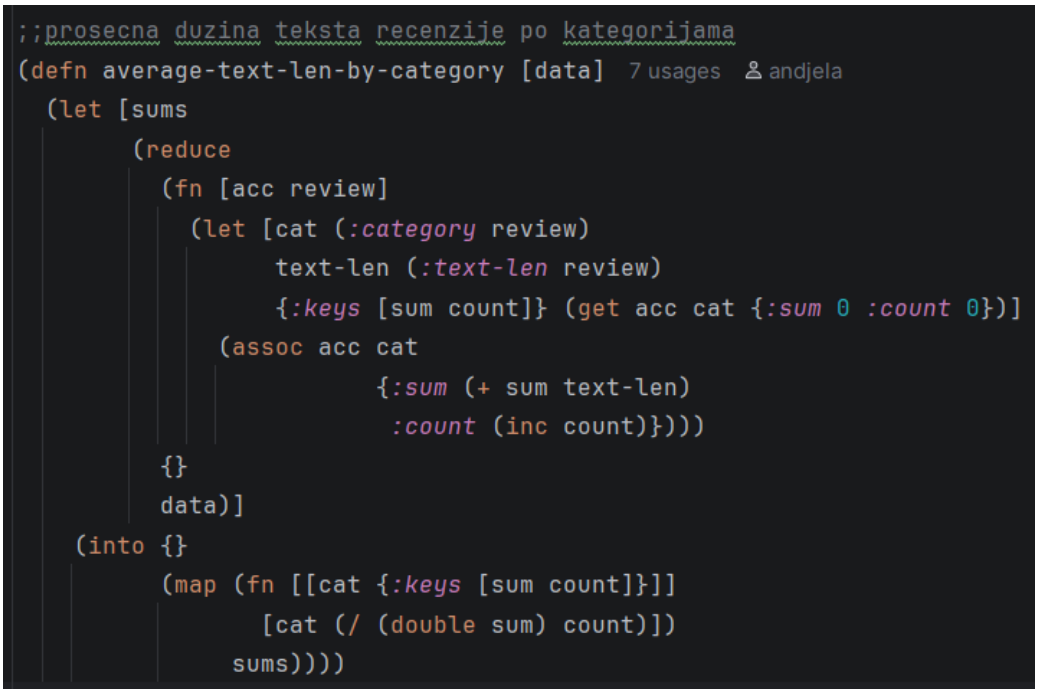
Još jedna implemetirana funkcija je funkcija za računanje prosečne ocene po kategoriji koja je prikazana na sledećoj slici.

```
;;prosecna ocena po kategorijama
(defn average-rating-by-category [data]  7 usages  andjela
  (let [sums
        (reduce
          (fn [acc review]
            (let [cat (:category review)
                  rating (:rating-num review)
                  {:keys [sum count]} (get acc cat {:sum 0 :count 0})]
              (assoc acc cat
                     {:sum (+ sum rating)
                      :count (inc count)})))
          {}
          data)]
    (into {}
          (map (fn [[cat {:keys [sum count]}]]
                 [cat (/ sum count)])
               sums))))
```

Slika 13 - Funkcija za računanje prosečne ocene po kategorijama

Prvo u okviru let definišemo funkciju sums koja koristeći reduce za svaku recenziju uzima kategoriju i ocenu, sabira ih i broji i vrednosti upisuje u akumulator acc. Nakon prolaska kroz mapu koja je prosledjena kao argument funkcije dobija se zbir i broj ocena svih recenzija po kategorijama(sums). Sledeći korak je da prolazak map funkcijom kroz dobijen rezultat, računanje srednje vrednosti ocene za svaku kategoriju i smeštanje u mapu korišćenjem into{}.

Na isti način implementirana je funkcija za računanje prosečne dužine teksta po kategorijama, prikazana na sledećoj slici.



```
;;prosecna duzina teksta recenzije po kategorijama
(defn average-text-len-by-category [data]  7 usages  andjela
  (let [sums
        (reduce
          (fn [acc review]
            (let [cat (:category review)
                  text-len (:text-len review)
                  {:keys [sum count]} (get acc cat {:sum 0 :count 0})]
              (assoc acc cat
                     {:sum (+ sum text-len)
                      :count (inc count)})))
          {}
          data)]
    (into {}
          (map (fn [[cat {:keys [sum count]}]]
                 [cat (/ (double sum) count)])
               sums))))
```

Slika 14 - Funkcija za računanje prosečne dužine teksta recenzije po kategorijama

Obe prethodno navedene funkcije korišćene su za računanje prosečne ocene i prosečne dužine teksta po kategorijama svih fake i svih real recenzija kao što je već objašnjeno.

5.4 Namespace logistic-regression

U ovom namespace-u implementiran je model logističke regresije. Logustčka regresija je jedna od metoda mašinskog učenja koja se koristi za rešavanje problema binarne klasifikacije i procenjuje verovatnoću pripadnosti određenoj klasi.

Model se zasniva na sigmoidnoj funkciji, koja transformiše linearni izraz u vrednosti između 0 i 1. Na osnovu dobijene verovatnoće i unapred definisanog praga, donosi se odluka o tome kojoj klasi posmatrana instanca pripada.

Cilj ovog modela je da predvidi da li je recenzija real ili fake na osnovu ocene i dužine teksta.

S obzirom da su većina funkcija koje su korišćene u ovom radu već objašnjene, u ovom poglavlju veći fokus će biti na sam algoritam logističke regresije.

Na početku je neophodno pripremiti podatke za model, podeliti dataset na deo za trening i deo za testiranje i napraviti vector ulaza. Funkcija za podelu dataseta prikazana je na sledećoj slici.

```
;;podela na trening i test deo
(defn split-60-40 [dataset]  1 usage  new *
  (let [shuffled (vec (shuffle dataset))
        idx (int (* 0.6 (count shuffled)))]
    {:train (subvec shuffled 0 idx)
     :test  (subvec shuffled idx)}))
```

Slika 15 - Funkcija za podelu dataseta

Funkcija shuffle nasumično „mesa" podatke dataseta, a zatim funkcija vec pretvara u vector. Idx prestavlja indeks koji koristimo za podelu koristeći subvec.

Kada je u pitanju priprema podataka, radi se normalizacija kako bi se izbegla različita razmera ulaznih podataka tako da se oni transformišu u opseg 0-1. U ovom projektu korišćena je min-max normalizacija koja tranformiše podatke prema sledećoj formuli:

$$x_{\text{norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

Bitno je napomenuti da se max i min vrednost računaju u okviru dataset-a za trening.

Funkcija za normalizaciju implementirana je kako je prikazano na sledećoj slici.

```
(defn normalize [x min-x max-x]  2 usages  new *
  (if (= min-x max-x)
    0.0
    (/ (- x min-x)
       (- max-x min-x))))
```

Slika 16 - Funkcija za normalizaciju

Kako bi se izbeglo deljenje nulom proveravamo da li su max i min jednaki.

U modelima mašinskog učenja svaka instanca iz skupa podataka predstavlja se vektorom ulaza. Za svaku korisničku recenziju formiran je vektor ulaza koji se sastoji od bias člana i normalizovanih vrednosti karakteristika, dužine teksta I ocene.

Na sledećoj slici prikazana je konačna priprema podataka za logistučku regresiju.

```
;;normalizacija i kreiranje vektora i formiranje vektora
(defn prepare-for-logistic [reviews min-text-len max-text-len min-rating max-rating]
  (mapv
    (fn [review]
      {:x [1.0
           (normalize (to-double (:text-len review)) min-text-len max-text-len)
           (normalize (to-double (:rating-num review)) min-rating max-rating)]
       :y (to-double (:label review))})
    reviews))

;;konačna priprema za logisticku regresiju
(defn prepare-split [dataset]  1 usage  new *
  (let [{:keys [train test]} (split-60-40 dataset)
        min-tl (min-text-len train)
        max-tl (max-text-len train)
        min-r (min-rating train)
        max-r (max-rating train)
        train-data (prepare-for-logistic train min-tl max-tl min-r max-r)
        test-data  (prepare-for-logistic test  min-tl max-tl min-r max-r)]
    {:train-data train-data
     :test-data  test-data}))
```

Slika 17 - Priprema podataka za logističku regresiju

U okviru logističke regresije, procena pripadnosti jedne instance određenoj klasi zasniva se na kombinaciji ulaznih karakteristika i odgovarajućih težina modela. Prvi korak je izračunavanje linearnog skora koji se dobija kao skalarni proizvod vektora težina i vektora ulaza, pri čemu svaka karakteristika utiče na rezultat u skladu sa svojom težinom.

Dobijeni linearni skor zatim se transformiše primenom sigmoidne funkcije. Sigmoidna funkcija preslikava realne vrednosti u opseg od 0 do 1, čime se omogućava interpretacija rezultata kao verovatnoće pripadnosti pozitivnoj klasi, prema sledećoj formuli:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

, gde je z skalarni proizvod vektora težina i ulaza.

Na osnovu vrednosti dobijene sigmoidnom funkcijom formira se skor modela, koji predstavlja procenjenu verovatnoću da instanca pripada određenoj klasi. Skor se kasnije poredi sa unapred definisanim pragom odlučivanja kako bi se donela konačna odluka o klasifikaciji instance.

Na sledećim slikama prikazana je implementacija.

```
;;racunanje skalarnog proizvoda
(defn dot-product [a b]  1 usage  new *
  (reduce + (map * a b)))

;;sigmoid funkcija
(defn sigmoid [z]  1 usage  new *
  (/ 1.0 (+ 1.0 (Math/exp (- z)))))

;;score
(defn score [w x]  2 usages  new *
  (sigmoid (dot-product w x)))
```

Slika 18 - Sigmoidna funkcija i score

Naredni korak je optimizacija težina gde se algoritam gradijentnog spusta. Cilj ovog koraka je da se postojeće vrednosti težina prilagode tako da se smanji greška između stvarne klase i predikcije modela. Model najpre izračuna predikciju prema formuli:

$$\hat{y} = \sigma(\mathbf{w} \cdot \mathbf{x})$$

zatim grešku:

$$\text{err} = \hat{y} - y$$

a potom se ažurira svaka težina prema formuli:

$$w_i := w_i - \alpha \cdot (\hat{y} - y) \cdot x_i$$

U kodu je to implementirano na sledeći način prikazan na slici.

```
(defn update-weights [w x y alpha]  1 usage  new *
  (let [p   (score w x)
        err (- p y)]
    (mapv (fn [wi xi] (- wi (* alpha err xi))) w x)))
```

Slika 19 - Funkcija za ažuriranje težina

Nakon ovoga, može da se počne treniranje modela. Trening modela se sastoji iz više epoha gde se u svakoj uzima ulazni vektor, izračunava predikcija I greška I ažurira vektor

težina. Na početku treniranja, vektor težina se inicijalizuje na nule koji se kroz epohe ažurira tako da kada se prođe kroz sve epohe dobijemo konačni vektor težina.

```
;;jedna epoha treniranja
(defn train-epoch [w train alpha]  1 usage  new *
  (reduce (fn [w {:keys [x y]}] (update-weights w x y alpha))
          w
          train))

;;trening
(defn train [train-data alpha epochs]  1 usage  new *
  (let [dim (count (:x (first train-data)))]
    (loop [w (vec (repeat dim 0.0))
           e 0]
      (if (>= e epochs)
        w
        (recur (train-epoch w train-data alpha)
               (inc e))))))
```

Slika 20 - Treniranje modela

Nakon treniranja, model se koristi za klasifikaciju novih instanci. Na osnovu izlaza sigmoid funkcije dobija se verovatnoća da instanca pripada pozitivnoj klasi.

Kako bi se donela odluka kojoj grupi instance pripada, koristi se prag odlučivanja threshold. Ukoliko je verovatnoća veća ili jednaka pragu, instanca se klasifikuje kao pozitivna klasa, a u suprotnom se klasifikuje kao negativna.

Najčešće korišćena vrednost praga je 0.5, ali se on može prilagođavati u zavisnosti od cilja modela. Prikazana na sledećoj slici.

```
(defn predict [w x threshold]  1 us
  (if (>= (score w x) threshold)
    1.0
    0.0))
```

Slika 21 - Funkcija za predikciju modela

Nako novog koraka prelazi se na testiranje modela i evaluaciju preformansi. Najpre se formira matrica konfuzije koja se sastoji od tp(true positive), fp(false positive), tn(true negative) i fn(false negative). Kao što je prikazano na sledećoj slici.

```
;;računanje matrice konfuzije
(defn confusion-matrix [preds]  1 usage  new *
  (reduce
    (fn [{:keys [tp fp tn fn] :as acc} {:keys [y-true y-pred]}]
      (cond
        (and (= y-true 1.0) (= y-pred 1.0)) (update acc :tp inc)
        (and (= y-true 0.0) (= y-pred 1.0)) (update acc :fp inc)
        (and (= y-true 0.0) (= y-pred 0.0)) (update acc :tn inc)
        (and (= y-true 1.0) (= y-pred 0.0)) (update acc :fn inc)
        :else acc))
    {:tp 0 :fp 0 :tn 0 :fn 0}
    preds))
```

Slika 22 - Računanje matrice konfuzije

Ovi podaci su nam potrebni kako bi izračunali tačnost, preciznost, odziv i F1 metrike.

```
(defn accuracy [{:keys [tp tn fp fn]}]  1 usage  new *
  (double (/ (+ tp tn) (+ tp tn fp fn))))

(defn precision [{:keys [tp fp]}]  2 usages  new *
  (if (zero? (+ tp fp)) 0.0
                        (double (/ tp (+ tp fp)))))

(defn recall [{:keys [tp fn]}]  2 usages  new *
  (if (zero? (+ tp fn)) 0.0
                        (double (/ tp (+ tp fn)))))

(defn f1 [cm]  1 usage  new *
  (let [p (precision cm)
        r (recall cm)]
    (if (zero? (+ p r))
      0.0
      (double (/ (* 2 p r) (+ p r))))))
```

Slika 23 - Računanje metrika

Na kraju ovog namespace-a dodata je funkcija koja objedinjuje sve prethodno objašnjene koja za cilj ima jednostavno pokretanje logističke regresije, prikazano na sledećoj slici.

```clojure
(defn run-logistic-regression  1 usage  new *
  [dataset alpha epochs threshold]
  (let [{:keys [train-data test-data]} (prepare-split dataset)
        w (train train-data alpha epochs)
        preds (mapv (fn [{:keys [x y]}]
                      {:y-true y
                       :y-pred (predict w x threshold)})
                    test-data)
        cm (confusion-matrix preds)]
    {:cm cm
     :accuracy  (accuracy cm)
     :precision (precision cm)
     :recall    (recall cm)
     :f1        (f1 cm)}))
```

Slika 24 - Funkcija logističke regresije

Rezultat logičke regresije je prikazan na sledećoj slici:

```
---------- LOGISTIC REGRESSION RESULTS -----------
Total samples: 40526
Training samples: 24315
Test samples: 16211
Learning rate (alpha): 0.001
Epochs: 10
Threshold: 0.5

Confusion matrix:
TP = 6452  FP = 5929
TN = 2171  FN = 1659

Accuracy:  0.5319227684905311
Precision: 0.5211210726112592
Recall:    0.7954629515472815
F1 score:  0.629709154792114
--------------------------------------------------
```

Slika 25 - Rezultat logističke regresije

Model logističke regresije pokazuje umerene performanse, sa tačnošću od oko 53% i visokom vrednošću odziva (≈0.80), što ukazuje da uspešno prepoznaje većinu pozitivnih instanci, ali uz veći broj lažno pozitivnih predikcija. Preciznost je srednjeg nivoa (≈0.52), dok F1-mera (≈0.63) ukazuje na solidan balans između preciznosti i odziva. Bitno je napomenuti da je ovo samo bazni model koji treba dodatno unaprediti.

5.5 Namespace core

Core namespace je zadužen za orkestraciju celokupnog procesa rada sistema. U ovom delu projekta povezuju se svi prethodno implementirani moduli, uključujući učitavanje i pripremu podataka, treniranje modela logističke regresije i evaluaciju rezultata. Zato se u njega uvoze svi ostali namespace-ovi.

U ovom namespace-u prvo definišemo reviews koja predstavlja sve recenzije i koju kasnije koristimo za pozivanje svih implementiranih funkcija u okviru main.

```
(ns fake-reviews.core  no usages  andjela *
  (:require [fake-reviews.data-source :as ds]
            [fake-reviews.preprocess :as prep]
            [fake-reviews.analytics :as an]
            [fake-reviews.logistic-regression :as lr]))

(def reviews  andjela
  (doall
    (map prep/prepare-reviews
         (prep/transform-rows-to-maps
           (ds/read-csv "fake_reviews_dataset.csv")))))

(defn -main []  andjela *
  (println "-----------------------------------------
  (println "Total reviews:" (count reviews))
```

Slika 26 - Namespace core

Kao konačan rezultat dobija se prikaz rezultata analize i logističke regresije, kao što je prikazano na sledećoj slici. Pokrenute su samo neke od funkcija iz analytics namespace-a radi lakšeg prikaza.

```
------------------------------------------- ANALYTICS -------------------------------------------------
Total reviews: 40526
Total real: 20232
Total fake: 20294
Average rating: 4.2554162759709815
Average rating of real: 4.253064452352708
Average rating of fake: 4.257760914556027
Average text length: 351.4265163105167
Average text length of real: 397.6539145907473
Average text length of fake: 305.34034690056177
Top 3 categories with most real reviews ([Kindle Store 2365] [Books 2187] [Pet Supplies 2127])
Top 3 categories with most fake reviews ([Kindle Store 2365] [Books 2192] [Pet Supplies 2127])
------------------------------------- LOGISTIC REGRESSION RESULTS -------------------------------------
Total samples: 40526
Training samples: 24315
Test samples: 16211
Learning rate (alpha): 0.001
Epochs: 10
Threshold: 0.5

Confusion matrix:
TP = 5583  FP = 4807
TN = 3212  FN = 2609

Accuracy:  0.5425328480661279
Precision: 0.5373435996150144
Recall:    0.6815185546875
F1 score:  0.6009041007426541
-------------------------------------------------------------------------------------------------------
```

Slika 27 – Rezultat

6.Testiranje

Za testiranje funkcionalnosti korišćena je Midje biblioteka koja omogućava jednostavno pisanje testova u formi činjenica. U ovom projektu urađeno je testitanje funkcija za analizu. Testovi se nalaze u posebnom namespace-u okviru test direktorijuma. (analytics-tests), gde se uvoze namespace gde se nalaze funkcije koje se testiraju i Midje. Kao što je prikazano na sledećoj slici test se sastoji od tri dela. Prvi deo je činjenica (fact) koju proveravamo. Drugi deo se odnosi na funkciju koju testiramo i treći je očekivana vrednost.

```
;testovi za avg
(fact "avg calculates the average of numbers"
      (avg [3 5]) => 4.0
      (avg [5 5 5]) => 5.0)

(fact "avg ignores nil values"  andjela
      (avg [3 nil 5]) => 4.0)
```

Slika 28 - Testiranje funkcije za računanje prosečne vrednosti

S obzirom na to da se ostali testovi odnose na recenzije potrebno je kreirati testne podatke. Kreirano je 13 različitih recenzija sa varijablama koje se koriste za testiranje, prikazano na sledećoj slici.

```
(def test-reviews  23 usages  andjela
  [;; Books: fake=2, real=1
   {:label-name "fake" :rating-num 5 :text-len 120 :category "Books"}
   {:label-name "fake" :rating-num 1 :text-len  80 :category "Books"}
   {:label-name "real" :rating-num 4 :text-len 200 :category "Books"}

   ; Electronics: fake=1, real=2
   {:label-name "fake" :rating-num 2 :text-len  60 :category "Electronics"}
   {:label-name "real" :rating-num 5 :text-len 150 :category "Electronics"}
   {:label-name "real" :rating-num 3 :text-len  90 :category "Electronics"}

   ;; Home: fake=3, real=0
   {:label-name "fake" :rating-num 4 :text-len 110 :category "Home"}
   {:label-name "fake" :rating-num 4 :text-len  70 :category "Home"}
   {:label-name "fake" :rating-num 2 :text-len  50 :category "Home"}

   ;; Toys: fake=0, real=2
   {:label-name "real" :rating-num 2 :text-len  40 :category "Toys"}
   {:label-name "real" :rating-num 4 :text-len 100 :category "Toys"}

   ;; Sports: fake=1, real=1
   {:label-name "fake" :rating-num 3 :text-len  55 :category "Sports"}
   {:label-name "real" :rating-num 1 :text-len  30 :category "Sports"}])
```

Slika 29 - Kreiranje podataka za testiranje

Na sledećim slikama prikazani su neki od sprovedenih testova.

Za funkcije koje vraćaju sve fake ili real recenzije testirano je da li vraćaju stvarni broj recenzija.

```
(fact "get-fake-reviews returns only fake reviews"
      (count (get-fake-reviews test-reviews)) => 7)

(fact "get-real-reviews returns only fake reviews"
      (count (get-real-reviews test-reviews)) => 6)
```

Slika 30 - Testovi za fake i real recenzije

Testovi za prosečne ocene I dužine teksta recenzije koriste roughty koje omogućava navadeno odstupanje (0.001) zbog rada sa double vrednostima.

```
(fact "computes average rating across all reviews (avg=3.0769)"
      (average-rating test-reviews)
      => (roughly 3.0769 0.001))

(fact "computes average review text length (avg=88.846)"  andjel
      (average-text-length test-reviews)
      => (roughly 88.846 0.001))
```

Slika 31 - Testovi za računanje pročnih ocena i dužina teksta recenzija

U testovima koji proveravaju top tri kategorije po broju fake i real recenzija prvo proveravamo da li nam vraća tačno tri kategorije, a zatim pošto neke kategorije imaju isti broj recenzija fake ili real po kategoriji i ne možemo da znamo kojim redom će se vratiti kao rezultat funkcije koristimo some kako bismo proverili da li se određena kategorija pojavljuje u rezultatu.

```
(fact "returns top 3 categories by number of fake reviews"  andjela
      (count (top-fake-categories test-reviews)) => 3
      (some #{["Home" 3]} (top-fake-categories test-reviews)) => truthy
      (some #{["Books" 2]} (top-fake-categories test-reviews)) => truthy
      (some #{["Electronics" 1] ["Sports" 1]} (top-fake-categories test-reviews)) => truthy)

(fact "top-real-categories returns the three categories with the most real reviews"  new *
      (count (top-real-categories test-reviews)) => 3
      (some #{["Electronics" 2] ["Toys" 2]} (top-real-categories test-reviews)) => truthy
      (some #{["Electronics" 2] ["Toys" 2]} (top-real-categories test-reviews)) => truthy
      (some #{["Books" 1] ["Sports" 1]} (top-real-categories test-reviews)) => truthy)
```

Slika 32 - Testovi za top 3 kategorije po broju recenzija

7. Zaključak

U ovom projektu razvijen je sistem za analizu i klasifikaciju recenzija primenom algoritma logističke klasifikacije. Projekat obuhvata kompletan process obrade podataka, od učitavanja i pripreme podataka za model do evaluacije performansi.

Dobijeni rezultati pokazuju da logistička regresija, kao relativno jednostavan model, može pružiti dobre rezultate, posebno kada je u pitanju odziv sistema. Iako je u ovom radu fokus bio da se prikaže algoritam logičke regresije, a ne sam model, analiza metrika pokazuje da dati model može još da se značajno unapredi.. Posebno značajan prostor za poboljšanje može biti uvođenje dodatnih varijabli koje se izvode iz samog teksta recenzija. Pored osnovnih numeričkih osobina, moguće je koristiti statističke karakteristike teksta, kao što su broj reči i rečenica, prosečna dužina reči, upotreba velikih slova i interpunkcije, kao i broj pozitivnih i negativnih reči u recenziji i njihov odnos. Takodje, performanse bi se mogle poboljšati optimizacijom parametara, stope učenja, broj epoha i prag odlučivanja, kao i obradom autlajera odnosno ekstremnih vrednosti instanci koje značajno odstupaju od ostalih. Dalje unapređenje može uključiti primenu složenijih algoritama mašinskog učenja poput neuronskih mreža.

Pored unapređenja samog modela, projekat pruža dobru osnovu za proširenje funkcionalnosti u vidu kompletne aplikacije. Sistem bi se mogao razviti u web aplikaciju sa korisničkim interfejsom koji omogućava unos recenzija i prikaz rezultata klasifikacije u realnom vremenu.

8.Literatura

- Higginbotham D. (2015), *Clojure for the brave and true,* dostupno na: https://www.kea.nu/files/textbooks/humblelearn2code/clojureforthebraveandtrue.pdf

-Clojure dokumentacija, dostupno na: https://clojure.org/

-GeeksforGeeks. (2025), *Logistic Regression in Machine Learning,* dostupno na: https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/