

Fakultet organizacionih nauka  
Katedra za Softversko inženjerstvo i veštačku inteligenciju

# Projekat iz predmeta Alati i metode veštačke inteligencije i softverskog inženjerstva

Mentor:

Prof. dr Dragan Đurić

Student:

Viktorija Barišić 3804/23

Beograd, septembar 2024.

<b>1. Uvod</b>	<b>3</b>
<b>2. Opis Projekta</b>	<b>4</b>
Korišteni podaci	4
Glavne funkcionalnosti	5
<b>3. Arhitektura i tehnologije</b>	<b>6</b>
Arhitektura projekta	6
Korišćene tehnologije i alati	6
<b>4. Funkcionalnosti</b>	<b>8</b>
Učitavanje podataka i rad sa bazom	8
Analiza podataka	11
1. Priprema podataka	11
2. Statistička Analiza	12
3. Korelacija i analize	14
Vizualizacija podataka	16
1. Generisanje grafikona	16
Predikcija	20
1. Prediktivni model	20
2. Funkcionalnosti za predikciju	23
3. Analiza rezultata	24
<b>5. Testiranje - Midje biblioteka</b>	<b>27</b>
Testiranje Funkcionalnosti	27
1. Testiranje analize podataka	27
2. Testiranje prediktivnog modela	27
<b>6. Zaključak</b>	<b>29</b>
Preporuke za dalji rad	29
<b>7. Literatura</b>	<b>30</b>

# 1. Uvod

Ovaj dokument predstavlja dokumentaciju za Clojure projekat koji je kreiran za potrebe polaganja predmeta Alati i metode veštačke inteligencije i softverskog inženjerstva na Master studijama na Fakultetu organizacionih nauka, smjer Softversko inženjerstvo i veštačka inteligencija.

Projekat se bavi analizom *data seta* koji sadrži podatke o učenicima i njihovim dostignućima, kao i njihovim određenim karakteristikama koje potencijalno utiču na ocjenu.

Samim tim, glavni cilj ovog projekta je da se utvrdi koji to faktori (najviše) utiču na uspjeh, odnosno krajnju ocjenu kod učenika.

Ideja i inspiracija za projekat potiče od lične želje. Naime, kako neki učenici brže uče i savladaju gradivo, bez obzira na predmet, to me je navelo da se zapitam, koji to faktori mogu imati uticaj na akademski uspjeh. Da li predispozicije učenika koje on ne bira, kao što je, recimo, obrazovanje roditelja, utiče na njegove ocjene? Da li srednjoškolci koji volontiraju imaju bolje ocjene? Na kraju, da li učenici koji uče više zaista dobijaju bolje ocjene? Ovo su neka od pitanja na koja je sam tražila odgovore i pronašla ih kroz ovaj projekat, makar na nivou ovog uzorka učenika.

Projekat je razvijen koristeći Clojure programski jezik i koristi različite biblioteke za rad sa podacima, statistiku, vizualizaciju i prediktivnu analizu.

U ovom dokumentu će biti obuhvaćeni sledeći aspekti projekta:

- **Opis projekta:** Opis funkcionalnosti i svrhe, tj. rezultata projekta.
- **Arhitektura i tehnologije:** Detaljan pregled arhitekture projekta i korišćenih tehnologija.
- **Evaluacija rezultata i prijedlozi za dalji rad.**

## 2. Opis Projekta

Projekat se fokusira na analizu akademskih performansi učenika koristeći podatke kao što su starost, broj izostanaka, posjedovanje mentora, volontiranje, obrazovanje roditelja i vrijeme učenja. Koristi se za analizu uticaja ovih atributa na konačnu ocjenu učenika i za izgradnju prediktivnog modela koji može predviđati ocjene na osnovu unijetih atributa.

Napomena: Treba imati u vidu da su ocjene izražene u slovima jer se radi o data setu koji obuhvata učenike sa teritorije gdje se koristi ovakav način ocjenjivanja. Osim toga, bitno je da se naglasi da prilikom prevođenja GPA, tj. ocjene u atribut Grade Class, logika je bila takva da najniži broj za Grade Class je zapravo najveći GPA, tj. najviša ocjena. U nastavku je dato više detalja u odjeljku Akademski učinak.

### Korišteni podaci

Data set je preuzet sa sajta kaggle.com - <https://doi.org/10.34740/KAGGLE/DS/5195702>.

Ovaj skup podataka sadrži informacije o 2.392 učenika srednjih škola. Ukupno ima 15 atributa (uključujući i ID koji se ne razmatra), koji su objašnjeni u nastavku.

#### Demografski:

- **Starost**
- **Pol**
- **Etnička pripadnost:** 0: Bjelac; 1: Afroamerikanac; 2: Azijac; 3: Drugo
- **Edukacija roditelja:** 0: Bez obrazovanja; 1: Srednja škola; 2: Djelimično visoko obrazovanje; 3: Diplomski; 4: Više obrazovanje
- **Podrška roditelja:** 0: Nema; 1: Niska; 2: Umerena; 3: Visoka; 4: Veoma visoka
- **Vanškolske aktivnosti:** Učešće u vanškolskim aktivnostima, 0 označava Ne, a 1 označava Da.
- **Sport:** Učešće u sportskim aktivnostima, 0 označava Ne, a 1 označava Da.
- **Muzika:** Učešće u muzičkim aktivnostima, 0 označava Ne, a 1 označava Da.
- **Volontiranje:** Učešće u volontarizmu, 0 označava Ne, a 1 označava Da.

#### Navike u učenju

- **StudyTimeWeekly:** Vremenski period proveden u učenju na nedjeljnomy nivou, u satima, u rasponu od 0 do 20.
- **Absences:** Broj izostanaka tokom školske godine, u rasponu od 0 do 30.
- **Tutoring:** Status mentorstva, gde 0 označava "Ne" a 1 označava "Da".

#### Akademski učinak:

- **GPA:** Prosječna ocjena na skali od 0.0 do 4.0
- **Klasifikacija ocjena:**

Klasifikacija ocjena učenika na osnovu GPA:

- 0: 'A' ( $\text{GPA} \geq 3.5$ )
- 1: 'B' ( $3.0 \leq \text{GPA} < 3.5$ )
- 2: 'C' ( $2.5 \leq \text{GPA} < 3.0$ )
- 3: 'D' ( $2.0 \leq \text{GPA} < 2.5$ )
- 4: 'F' ( $\text{GPA} < 2.0$ )

Napomena: U projektu su prilikom predviđanja korišćeni sljedeći faktori: godine, odsustvo, volontiranje, posjedovanje mentora, obrazovanje roditelja i broj sati učenja na sedmičnom niovu.

## Glavne funkcionalnosti

1. **Učitavanje podatka iz data seta u bazu podataka**
2. **Analiza podataka:**
  - Analiza podataka iz SQLite baze podataka.
  - Statistička analiza varijacija i korelacija između različitih atributa i ocjena učenika.
3. **Vizualizacija:**
  - Generisanje grafikona i histograma koji prikazuju odnose između različitih atributa i uspjeha učenika.
  - Vizualizacija uticaja izostanaka na ocjene i analize sa više faktora.
4. **Predikcija:**
  - Kreiranje i obuka prediktivnog modela koristeći algoritam odlučivanja (J48) iz Weka biblioteke.
  - Predviđanje ocjena na osnovu unijetih atributa učenika.
5. **Interaktivno korišćenje:**
  - Omogućavanje korisnicima da unesu podatke o učenicima i dobiju predikciju ocjene na osnovu njihovih unosa.

# 3. Arhitektura i tehnologije

## Arhitektura projekta

Projekat je organizovan u nekoliko glavnih komponenata:

- **Glavni Modul (core):** Sadrži sve ključne funkcionalnosti projekta, uključujući analizu podataka, treniranje modela i generisanje vizualizacija.
- **Modul Repository:** Odgovoran za interakciju sa SQLite bazom podataka. Upravlja funkcijama za pristup podacima, kao što su preuzimanje podataka, dodavanje novih učenika i filtriranje učenika po određenim karakteristikama.
- **Baza podataka:** Sadrži tabelu učenici sa učitanim podacima iz data seta.
- **Testovi:** Sadrži Midje testove koji provjeravaju ispravnost određenih funkcija.

## Korišćene tehnologije i alati

Ovo poglavlje pruža osnovni uvid u tehnologiju, okruženje, kao i biblioteke koje su korištene za izradu projekta.

### Clojure

Clojure je programski jezik zasnovan na funkcionalnom programiranju, sa bogatim setom nepromenljivih, postojanih struktura podataka. Takođe je član porodice Lisp jezika, i koristi JVM za izvršavanje. [1]

Clojure je fokusiran na funkcionalno programiranje, što omogućava programerima da pišu kod koji je predvidljiviji i lakši za otklanjanje grešaka. Kroz minimizaciju promjenljivog stanja i promovisanjem korišćenja čistih funkcija, Clojure olakšava razumijevanje koda i obezbjeduje da se funkcije ponašaju dosljedno, bez obzira na to gdje ili kada su pozvane. [2]

Jedna od ključnih prednosti funkcionalnih jezika poput Clojure-a je **REPL** (Read-Eval-Print Loop). [3] Evo kako on funkcioniše:

1. **Eksperimentisanje sa kodom:** REPL omogućava brzo isprobavanje koda. Može se direktno unositi kod i odmah vidjeti rezultat, što omogućava efikasno testiranje i eksperimentisanje.
2. **Brz povratni ciklus:** Kada se unese kod, REPL ga odmah pročita, izvrši, prikaže rezultat, i zatim ponovo ponudi prompt za unos. Ovaj brzi ciklus povratnih informacija pomaže u razumijevanju i prilagođavanju koda u realnom vremenu.
3. **Interaktivni razvoj:** Koristeći REPL, može se interaktivno raditi s programom, što nije moguće u mnogim drugim jezicima. Ovo čini razvoj i otklanjanje grešaka mnogo jednostavnijim i intuitivnijim.

Ukratko, REPL omogućava brže i fleksibilnije učenje i razvoj u Clojure-u, što može značajno poboljšati razumijevanje koda i produktivnost programera. [3]

U ovom projektu, Clojure se koristi kao glavni jezik za implementaciju aplikacije, omogućavajući efikasno rukovanje podacima i kompleksnim analizama.

#### **Java i IntelliJ IDEA:**

Clojure zahtjeva Java Development Kit (JDK) za izvršavanje, a IntelliJ IDEA je jedno od najpopularnijih razvojnih okruženja koje se koristi za razvoj aplikacija u Clojure. IntelliJ pruža podršku za Clojure kroz dodatke, omogućavajući jednostavno upravljanje projektima i integraciju sa različitim tehnologijama. [4]

#### **SQLite**

SQLite je relaciona baza podataka koja se koristi za čuvanje podataka o učenicima. SQLite omogućava efikasno upravljanje podacima i jednostavno izvođenje SQL upita. [5]

#### **Biblioteke**

**Incanter:** Incanter je biblioteka za statistiku i vizualizaciju podataka u Clojure-u. Koristi se za kreiranje grafikona i histograma koji prikazuju analize podataka. [6]

**Weka:** Weka je alat za mašinsko učenje koji se koristi za obuku i primjenu prediktivnih modela. U ovom projektu, Weka se koristi za implementaciju algoritma odlučivanja (J48), koji pomaže u predviđanju ocjena na osnovu unijetih atributa učenika. Weka omogućava kreiranje i evaluaciju modela mašinskog učenja. [7]

**next.jdbc:** next.jdbc je biblioteka za rad sa bazama podataka u Clojure-u. Omogućava povezivanje sa SQLite bazom, izvođenje SQL upita i manipulaciju podacima. Ova biblioteka pruža jednostavan interfejs za rad sa bazama podataka i integraciju sa Clojure projektima. [8]

# 4. Funkcionalnosti

Ovo poglavlje opisuje ključne funkcionalnosti implementirane u projektu, uključujući i analizu podataka, vizualizaciju i predikciju.

## Učitavanje podataka i rad sa bazom

Naredne funkcije se nalaze u repository.clj. Samim tim, prvo je učitan *namespace* za glavni dio projekta i priključena biblioteka za rad sa bazom.

```
(ns clojure-projekat-master.repository
  (:require [next.jdbc :as jdbc]
            ))
```

Url je izdvojen u promljenjivu db-url.

```
(def db-url "identifier.sqlite")
```

Naredna funkcija omogućava učitavanje podataka iz baze.

```
(defn all-data []
  (let [ds (jdbc/get-datasource {:dbtype "sqlite" :dbname db-url})
        query "SELECT * FROM students"]
    (jdbc/execute! ds [query])))
)
```

Na ovaj način će podaci biti učitani u core.clj.

```
(def data (repository/all-data))
```

Filtriranje podataka, kao i grupisanje određenih vrijednosti atributa je omogućeno narednim funkcijama.

```
(defn get-Students-by-age-volunteering-tutoring [age]
  (let [ds (jdbc/get-datasource {:dbtype "sqlite" :dbname db-url})
        query "SELECT * FROM students WHERE Age = ? AND Volunteering = 1 AND Tutoring = 1
AND Gender = 1"]
    (jdbc/execute! ds [query age])))

(defn average-gpa-by-parental-education []
```

```
(let [ds (jdbc/get-datasource {:dbtype "sqlite" :dbname db-url})]

  query "SELECT ParentalEducation, AVG(GPA) AS AvgGPA FROM students GROUP BY ParentalEducation"]

  (jdbc/execute! ds [query]))
```

Naredna funkcija prikazuje sumirane vrijednosti ocjena koje su prisutne u datom data setu, što će biti relevantno i značajno za dalje rezultate u programu.

```
(defn count-students-by-grade-class []

(let [ds (jdbc/get-datasource {:dbtype "sqlite" :dbname db-url})]

  query "SELECT GradeClass, COUNT(*) AS Count FROM students GROUP BY GradeClass"

  results (jdbc/execute! ds [query]))

  (map (fn [{:keys [:Students/GradeClass :Count]}]

    {:GradeClass (grade-class-label GradeClass)

     :Count Count})

    results)))
```

Dakle, može se vidjeti da najveći dio data seta čine učenici sa lošim ocjenama - ukupno ih je 1211, dok učenika sa najvišom ocjenom ima svega 107.

```
(count-students-by-grade-class)
=>
{:GradeClass "Grade A", :Count 107}
{:GradeClass "Grade B", :Count 269}
{:GradeClass "Grade C", :Count 391}
{:GradeClass "Grade D", :Count 414}
{:GradeClass "Grade F", :Count 1211})
```

Narednom funkcijom se omogućava unos novog učenika u bazu podataka.

```
(defn add-student [db-url student-id age gender ethnicity parental-education
study-time-weekly absences tutoring parental-support extracurricular sports music
volunteering gpa grade-class]

(let [ds (jdbc/get-datasource {:dbtype "sqlite" :dbname db-url})]

  query "INSERT INTO students (StudentID, Age, Gender, Ethnicity, ParentalEducation,
StudyTimeWeekly, Absences, Tutoring, ParentalSupport, Extracurricular, Sports, Music,
Volunteering, GPA, GradeClass) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)" ]
```

```
(jdbc/execute! ds [query student-id age gender ethnicity parental-education  
study-time-weekly absences tutoring parental-support extracurricular sports music  
volunteering gpa grade-class]))
```

Napomena: ova funkcija nije pozvana u samom programu kako ne bi uticala na strukturu trenutnih podataka u data setu.

# Analiza podataka

## 1. Priprema podataka

Narednim funkcijama je provjereno da li postoje neke nedefinisane vrijednosti u data setu koje je potrebno mijenjati, kao i da li postoje duplirane vrijednosti.

```
; Checking if there are any missing values
(defn missing? [data]
  (some #(some nil? %) data))

; Result is nil => no missing values
(defn missing-values []
  (if (missing? data)
    "postoje nedostajuće vrijednosti"
    "nema nedostajućih vrijednosti"))

; Checking if there are any double rows
(defn find-duplicates [data]
  (let [unique-data (set data)]
    (if (= (count data) (count unique-data))
      nil
      (filter #(> (count (filter #{%} data)) 1) data)))

; Result is nil => no double rows
(defn duplicates []
  (if (find-duplicates data)
    "postoje duplikati"
    "nema duplikata"))
```

U nastavku se može vidjeti da je prilikom poziva funkcija utvrđeno da nema nedostajućih vrijednosti i data setu i da nema duplikata.

```
(missing-values)
=> "nema nedostajućih vrijednosti"
(duplicates)
=> "nema duplikata"
```

## 2. Statistička Analiza

Projekat omogućava detaljnu statističku analizu koristeći funkcionalnosti iz `incanter` biblioteke.

Da bi to bilo moguće, uključena je biblioteka u *Dependencies* u okviru project.clj, kao i u *namespace*-u core projekta.

```
[incanter "1.9.3"]
```

```
[incanter.core :as incanter]  
[incanter.stats :as stats]  
[incanter.charts :as charts]
```

Analiza uključuje:

**Uticaj različitih atributa na izlaznu varijablu *Grade Class*, tj. prosječnu ocjenu**

U nastavku su navedeni primjeri dvije takve funkcije.

Funkcija `average-grade-by-volunteering` grupiše podatke na osnovu toga da li učenici su volontirali ili ne i izračunava prosječne ocjene za obe grupe.

```
(defn average-grade-by-volunteering [data]  
  
(let [grouped-by-volunteering (group-by :Students/Volunteering data)  
  
      calculate-average (fn [rows]  
  
                         (let [total (reduce + (map #(get % :Students/GradeClass) rows))  
  
                             count (count rows)]  
  
                           (/ total count)))]  
  
  (into {} (map (fn [[volunteering rows]]  
  
                  [(if (= volunteering 1) "Volonteri:" "Oni koji ne volontiraju:")  
  
                   (calculate-average rows)]))  
  
                grouped-by-volunteering))))
```

Funkcija `average-grade-by-absences` grupiše podatke po opsegu izostanaka i izračunava prosječne ocjene za svaku grupu.

```
(defn average-grade-by-absences [data]
```

```

(let [grouped-by-absence-range (group-by (fn [row]
                                             (let [absences (get row :Students/Absences)]
                                               (cond
                                                 (<= absences 4) "0-4"
                                                 (<= absences 8) "5-8"
                                                 (<= absences 12) "9-12"
                                                 (<= absences 16) "13-16"
                                                 (<= absences 20) "17-20"
                                                 (<= absences 23) "21-23"
                                                 :else "24-29")))) data)

calculate-average (fn [rows]
                    (let [total (reduce + (map #(get % :Students/GradeClass) rows))
                          count (count rows)]
                      (/ total count)))

average-map (into {} (map (fn [[range rows]]
                            [range (calculate-average rows)])
                           grouped-by-absence-range))]

(let [key-order ["0-4" "5-8" "9-12" "13-16" "17-20" "21-23" "24-29"]]

(into {} (map (fn [k] [k (get average-map k)])
               key-order))))))

```

U nastavku su prikazani rezulatni poziva prethodno pomenutih funkcija i utvrđeno je da (neznačajno) veću ocjenu imaju oni koji volontiraju, dok učenici koji izostaju više, imaju manje ocjene.

```

(average-grade-by-volunteering data)
=> {"Oni koji ne volontiraju:" 2.976686507936508, "Volonteri:" 3.021276595744681}

(average-grade-by-absences data)
=>
{"0-4" 1.2791327913279134,
 "5-8" 2.1149425287356323,
 "9-12" 2.790378006872852,
 "13-16" 3.3527696793002915,
 "17-20" 3.7058823529411766,
 "21-23" 3.8916666666666666,
 "24-29" 3.8459869848156183}

```

### 3. Korelacija i analize

Projekat omogućava analizu korelacija između atributa kao što su `Absences`, `StudyTimeWeekly`, i `ParentalEducation` u odnosu na `GradeClass`. Analize se izvode koristeći funkcije kao što su `correlation-tutoring-gradeclass`.

U nastavku su dati primjeri nekoliko ovakvih funkcija.

```
(defn correlation-tutoring-gradeclass [data]
  (let [tutoring (map :Students/Tutoring data)
        grades (map :Students/GradeClass data)]
    (stats/correlation tutoring grades)))

(defn correlation-volunteering-gradeclass [data]
  (let [volunteering (map :Students/Volunteering data)
        grades (map :Students/GradeClass data)]
    (stats/correlation volunteering grades)))

(defn correlation-parental-education-gradeclass [data]
  (let [parentalEducation (map :Students/ParentalEducation data)
        grades (map :Students/GradeClass data)]
    (stats/correlation parentalEducation grades)))

(defn correlation-absences-gradeclass [data]
  (let [absences (map :Students/Absences data)
        grades (map :Students/GradeClass data)]
    (stats/correlation absences grades)))
```

U nastavku su prikazani rezultati poziva pomenutih funkcija i može se jasno vidjeti da najveći uticaj ima atribut izostanci. Takođe, zanimljivo zapaženje jeste da postojanje mentora ima negativnu korelaciju (iako malu), što znači da ukoliko neko posjeduje mentora, to će uticati na smanjivanje ocjene.

Napomena: Ne može se izvesti u potpunosti sa sigurnošću takav zaključak, bilo bi potrebno da se urade još neke analize kako bismo mogli sa sigurnošću tvrditi tako nešto.

```
74 (correlation-tutoring-gradeclass data)
=> -0.1116945788379372
75 (correlation-volunteering-gradeclass data)
=> 0.013156019636265172
76 (correlation-parental-education-gradeclass data)
=> 0.041031287770299886
77 (correlation-absences-gradeclass data)
=> 0.7286327104548662
80
```

# Vizualizacija podataka

## 1. Generisanje grafikona

Grafikoni i histogrami se generišu korišćenjem `incanter` biblioteke. Vizualizacije uključuju:

- **Veza između izostanaka i ocjena:** Histogram koji prikazuje distribuciju izostanaka i njihov uticaj na ocjene.
- **Analiza sa više faktora:** Grafički prikaz odnosa između `StudyTimeWeekly`, `Absences`, i `GradeClass`.

```
(defn plot-absences-vs-gradeclass [data]
  (let [absences (map :Students/Absences data)
        grades (map :Students/GradeClass data)
        grouped-data (group-by identity absences)
        binned-data (map (fn [[bin values]]
                           {:Absences bin
                            :AverageGrade (double (/ (apply + values)
                                                       (count values))))})
                      (into {} (map (fn [[k v]]
                                      [k (map (fn [d] (:Students/GradeClass d))
                                               (filter #(= (:Students/Absences %) k)
                                                       data))])
                                     grouped-data)))
        absences (map :Absences binned-data)
        avg-grades (map :AverageGrade binned-data)
        chart (charts/bar-chart absences avg-grades
                               :x-label "Absences"
                               :y-label "Average Grade Class"
                               :title "Average Grade Class by Absences")]
    (incanter/view chart)))
  (let [absences (map :Students/Absences data)
        study-time (map :Students/StudyTimeWeekly data)
```

```

grades (map :Students/GradeClass data)

colors (map (fn [grade]

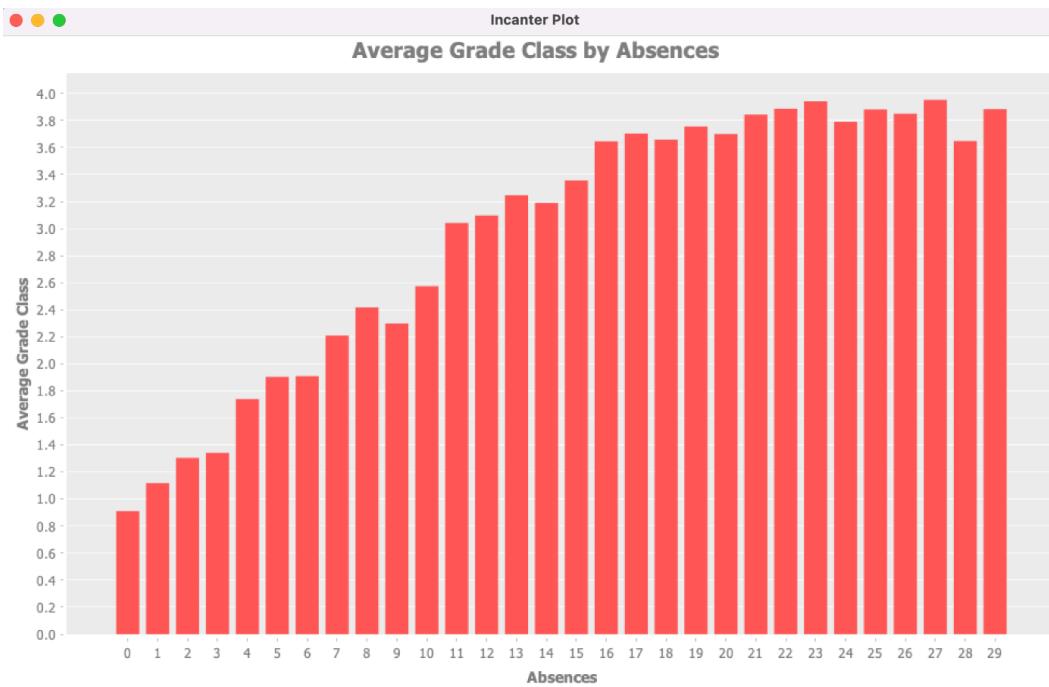
  (case grade
    0 "purple"
    1 "blue"
    2 "red"
    3 "green"
    4 "orange"
    "black"
  )))
grades)

chart (charts/scatter-plot absences study-time
  :x-label "Absences"
  :y-label "Study Time Weekly"
  :title "Absences vs Study Time Weekly"
  :group-by grades
  :color colors)
  )
  (incanter/view chart)))

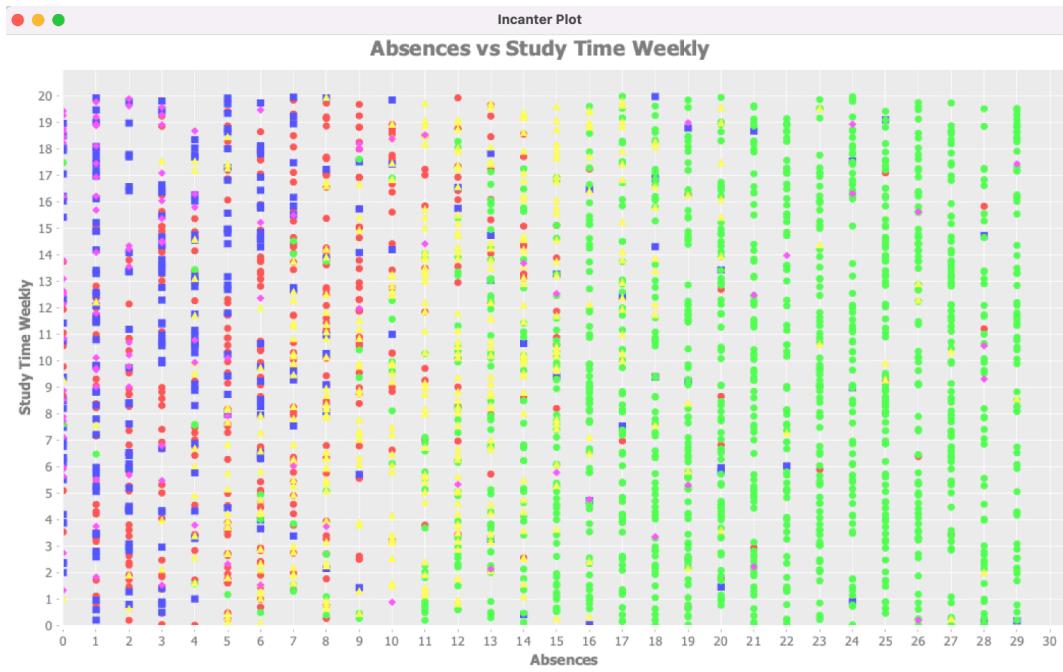
```

U nastavku su prikazani grafici koji se dobijaju pozivom funkcija i data su njihova objašnjenja.

Napomena: Ove funkcije nisu pozvane direktno u programu, samim tim grafici neće biti prikazani tokom izvršavanja programa jer se smatraju dodatnim načinom analize podataka, nepotrebnim za pregled korisnika u ovoj fazi projekta.



Na ovom grafiku je prikazana zavisnost izlazne varijable (y-osa), tj. ocjene učenika u odnosu na njegove izostanke (x-osa). Izabran je ovaj atribut za prikaz jer ima najveću korelaciju, tj. uticaj na sam ishod ocjene. Ono što se može zaključiti jeste da učenici koji su manje izostajali su imali bolje ocjene. Odnosno, kako je raste broj izostanaka, tako se smanjuje ocjena učenika.



Na ovom grafiku se prikazuje odnos između izostanaka (x-osa), sedmičnog učenja (y-osa) i ocjene učenika (različite boje i oblici na grafiku; ocjena A-ljubičasta, B-plava, C-crvena, D-narandžasta, F-narandžasta). Samim tim, može se vidjeti da je najveća koncentracija zelene boje, tj. najviše je loših ocjena i to kako se broj izostanaka povećava, tako i raste broj ovih ocjena, bez obzira na nivo učenja. Osim toga, u predjelu gdje su izostanci niski, bez obzira na nivo učenja, postoji veći broj boljih ocjena. Dok je, očekivano, u gornjem lijevom uglu, gdje su niski izostanci, a visok nivo učenja, najviše prisutne plava i ljubičasta koje odgovaraju boljim ocjenama.

# Predikcija

## 1. Prediktivni model

Projekat koristi Weka biblioteku za obuku prediktivnog modela koristeći algoritam odlučivanja (J48). Model predviđa ocene na osnovu atributa kao što su **Age**, **StudyTimeWeekly**, i **ParentalEducation**.

U nastavku su prikazane glavne funkcije za ovaj proces.

```
(defn create-instances [data]
  (let [grade-class-values ["A" "B" "C" "D" "F"]
        attributes (java.util.ArrayList. [(Attribute. "Age")
                                         (Attribute. "Absences")
                                         (Attribute. "Tutoring")
                                         (Attribute. "Volunteering")
                                         (Attribute. "ParentalEducation")
                                         (Attribute. "StudyTimeWeekly")
                                         (Attribute. "GradeClass" (java.util.ArrayList.
grade-class-values))])
        instances (Instances. "StudentPerformance" (java.util.ArrayList. attributes) (count
data)))
    (.setClassIndex instances 6)
    (doseq [row data]
      (let [instance (DenseInstance. (count attributes))]
        (.setDataset instance instances)
        (.setValue instance 0 (double (or (get row :Students/Age) 0)))
        (.setValue instance 1 (double (or (get row :Students/Absences) 0)))
        (.setValue instance 2 (double (or (get row :Students/Tutoring) 0)))
        (.setValue instance 3 (double (or (get row :Students/Volunteering) 0)))
        (.setValue instance 4 (double (or (get row :Students/ParentalEducation) 0)))
        (.setValue instance 5 (double (or (get row :Students/StudyTimeWeekly) 0))))
```

```

;; Handle GradeClass as a nominal value by setting it as a string directly

(let [grade-class (or (get row :Students/GradeClass) 0.0)

      nominal-value (get grade-class-map grade-class "A")]

  (.setValue instance (.attribute instances 6) nominal-value))

  (.add instances instance)))

instances)

;; Prediction using the model

(defn predict-instance [model instance-data dataset]

  (let [attributes-count 6

        instance (DenseInstance. attributes-count)]

    (.setDataset instance dataset)

    (.setValue instance 0 (double (get instance-data 0)))

    (.setValue instance 1 (double (get instance-data 1)))

    (.setValue instance 2 (double (get instance-data 2)))

    (.setValue instance 3 (double (get instance-data 3)))

    (.setValue instance 4 (double (get instance-data 4)))

    (.setValue instance 5 (double (get instance-data 5)))))

    ;; Predict using the model

    (.classifyInstance model instance)))

;; Creating instances for training

(def training-instances (create-instances training-data))

;; Training the model

(def model (J48.))

(.buildClassifier model training-instances)

```

Prethodno su podaci podijeljeni na training i test podatke, gdje je 80% data seta korišteno za treniranje, dok je preostalih 20% za testiranje.

```
(defn split-data [data ratio]
  (let [n (int (* ratio (count data)))]
    [(take n data) (drop n data)]))

;; Splitting data

(def data-split (split-data data 0.8))
(def training-data (first data-split))
(def test-data (second data-split))
```

## 2. Funkcionalnosti za predikciju

- **Funkcija `get-prediction`:** Omogućava korisnicima da unesu podatke pozivanjem funkcije `get-user-input` i zatim dobiju predikciju ocjene na osnovu unijetih atributa.

```
(defn get-user-input []

  (let [age (prompt-user "Unesite godine (Age 15-18):" (fn [x] (and (>= x 15) (<= x 18)))))

    absence (prompt-user "Unesite broj izostanaka (Absence 0-29):" (fn [x] (and (>= x 0)
(<= x 29)))))

    tutoring (prompt-user "Da li je učenik imao privatne časove? (0 za ne, 1 za da):"
(fn [x] (or (= x 0) (= x 1)))))

    volunteering (prompt-user "Da li je učenik volontirao? (0 za ne, 1 za da):" (fn [x]
(or (= x 0) (= x 1)))))

    parental-education (prompt-user "Unesite nivo obrazovanja roditelja
(ParentalEducation 0-4):" (fn [x] (and (>= x 0) (<= x 4)))))

    study-time-weekly (prompt-user "Unesite broj sati učenja nedeljno (StudyTimeWeekly
0-20):" (fn [x] (and (>= x 0) (<= x 20)))))

  [age absence tutoring volunteering parental-education study-time-weekly]))
```

```
(defn get-prediction []

  (let [user-input (get-user-input)

    prediction (predict-instance model user-input training-instances) ]

    (case (int prediction)

      0 "Grade A"

      1 "Grade B"

      2 "Grade C"

      3 "Grade D"

      4 "Grade F"

      "Unknown Grade")))
```

### 3. Analiza rezultata

U nastavku je dat primjer rezultata ukoliko se unesu dva učenika sa različitim atributima. Uzeta je u obzir prethodna analiza i zbog toga su unesene sljedeće vrijednosti. Naime, oba učenika imaju 15 godina, ali prvi ne izostaje, ne uzima privatne časove, volontira i uči 16 sati nedeljno, dok drugi učenik izostaje, uzima privatne časove i uči 2 sata nedeljno. Rezultati pokazuju da se za prvog učenika očekuje da će imati ocjenu B, dok će drugi učenik imati ocjenu F.

```
Unesite godine (Age 15-18):  
15  
Unesite broj izostanaka (Absence 0-29):  
2  
Da li je student imao privatne časove? (0 za ne, 1 za da):  
0  
Da li je student volontirao? (0 za ne, 1 za da):  
1  
Unesite nivo obrazovanja roditelja (ParentalEducation 0-4):  
2  
Unesite broj sati učenja nedeljno (StudyTimeWeekly 0-20):  
16  
Student ce imati ocenu: Grade B
```

```
.....  
Unesite godine (Age 15-18):  
15  
Unesite broj izostanaka (Absence 0-29):  
25  
Da li je student imao privatne časove? (0 za ne, 1 za da):  
1  
Da li je student volontirao? (0 za ne, 1 za da):  
0  
Unesite nivo obrazovanja roditelja (ParentalEducation 0-4):  
0  
Unesite broj sati učenja nedeljno (StudyTimeWeekly 0-20):  
2  
Student ce imati ocenu: Grade F
```

### Evaluacija modela

```
(defn evaluate-model [model test-instances]  
  
(let [evaluation (Evaluation. test-instances)]  
  
  (.evaluateModel evaluation model test-instances (make-array String 0))  
  
  ;; Ispis rezultata evaluacije  
  
  (println "Summary:")  
  
  (println (.toSummaryString evaluation))  
  
  (println "Confusion Matrix:")  
  
  (doseq [row (.confusionMatrix evaluation)]  
    (println (vec row)))
```

```

    (println "Class Details:")

    (println (.toClassDetailsString evaluation))

    ;; Vraćamo Evaluation objekat za dalje korišćenje ako je potrebno
    evaluation)

```

## Rezultati:

```

Summary:

Correctly Classified Instances      235          49.0605 %
Incorrectly Classified Instances   244          50.9395 %
Kappa statistic                   0.2819
Mean absolute error               0.2131
Root mean squared error          0.4062
Relative absolute error           71.2659 %
Root relative squared error     105.0927 %
Total Number of Instances        479

Confusion Matrix:
[3.0 10.0 11.0 7.0 17.0]
[5.0 15.0 13.0 13.0 27.0]
[2.0 14.0 28.0 14.0 23.0]
[0.0 7.0 20.0 29.0 30.0]
[1.0 5.0 11.0 14.0 160.0]
Class Details:
==== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC    ROC Area  PRC Area  Class
0.063    0.019    0.273    0.063    0.102    0.088    0.580    0.135    A
0.205    0.089    0.294    0.205    0.242    0.136    0.633    0.242    B
0.346    0.138    0.337    0.346    0.341    0.205    0.661    0.288    C
0.337    0.122    0.377    0.337    0.356    0.225    0.675    0.306    D
0.838    0.337    0.623    0.838    0.714    0.492    0.767    0.613    F
Weighted Avg.  0.491    0.195    0.445    0.491    0.454    0.301    0.693    0.399

```

Rezultati evaluacije pokazuju da model nije precizno klasifikovao većinu instanci u testnom skupu, s točnošću klasifikacije od 49.06%. To znači da je model tačno predvidio ishod za otprilike polovinu testiranih podataka, dok je za drugu polovinu pogrešno klasifikovao.

Kappa statistika (0.2819): Ova vrijednost ukazuje na slabu podudarnost između modela i stvarnih podataka. Kappa statistika mjeri koliko se klasifikacija modela razlikuje od slučajne klasifikacije. Vrijednosti blizu 1 ukazuju na dobru podudarnost, dok vrijednosti blizu 0 ukazuju na slučajnost.

Mean Absolute Error (MAE) i Root Mean Squared Error (RMSE): MAE (0.2131) i RMSE (0.4062) ukazuju na pogreške modela. MAE mjeri prosječnu apsolutnu razliku između stvarnih i predikovanih vrijednosti, dok RMSE naglašava veće pogreške, dajući im veću težinu. Niže vrijednosti su poželjne za obje metrike.

Confusion Matrix: Matrica konfuzije prikazuje točno i pogrešno klasifikovane primjere za svaku klasu. Na primjer, model je tačno klasifikovao samo 3 instance u klasu "A", dok je 17

instanci pogrešno klasificirao u klasu "F". Najveći broj tačnih klasifikacija model je postigao za klasu "F", s 160 tačno klasifikovanih instanci. Ovo se može povezati sa činjenicom da najveći dio ocjena u data setu zapravo čine F ocjene, što je spomenuto tokom prethodne analize.

**Detailed Accuracy By Class:** Ovaj dio pokazuje preciznost (Precision), odziv (Recall) i F-mjeru za svaku klasu. Najbolje performanse su u klasi "F" s F-mjerom od 0.714, što znači da model najbolje prepoznaje i predviđa ovu klasu, ali su performanse za ostale klase značajno slabije, posebno za klasu "A" s F-mjerom od samo 0.102. Ponovo, očekivan je bio uticaj toga što je u data setu mnogo mali broj učenika sa ocjenom A.

Bilo bi poželjno da se uradi analiza i predikcija sa nekim drugim modelima i zatim uporede dobijeni rezultati. Svakako, treba imati u vidu da se radi o većem broju atributa i da to isto ima uticaj na preciznost i tačnost predikcije.

# 5. Testiranje - Midje biblioteka

Ovo poglavlje opisuje metode i strategije testiranja korišćene za verifikaciju funkcionalnosti projekta. U te svrhe je korišćena Midje biblioteku.

Midje biiblioteka omogućava pisanje testova na izrazit, čitljiv način i integriše se s različitim testnim okruženjima kako bi osigurala robustan okvir za razvoj aplikacija. [9]

## Testiranje Funkcionalnosti

### 1. Testiranje analize podataka

Testiranje funkcionalnosti analize podataka uključuje verifikaciju tačnosti statističkih proračuna i korelacija:

- Prosječna ocjena po izostancima.
- Korelacija između atributa
- Postojanje nedostajućih vrijednosti.

```
(fact "Korelacija između atributa mentrostvo i ocjene"
  (correlation-tutoring-gradeclass [{:Students/Tutoring 2 :Students/GradeClass 3}
                                    {:Students/Tutoring 5 :Students/GradeClass 4}
                                    {:Students/Tutoring 1 :Students/GradeClass 2}])
  => (roughly 0.96 0.01))
=> true

(facts "Prosečne ocjene u zavisnosti od volontiranja"
  (fact "Volonteri"
    (average-grade-by-volunteering [{:Students/Volunteering 1 :Students/GradeClass 2}
                                    {:Students/Volunteering 1 :Students/GradeClass 3}])
    => {"Volonteri:" 5/2})

  (fact "Prosečna ocena za one koji ne volontiraju"
    (average-grade-by-volunteering [{:Students/Volunteering 0 :Students/GradeClass 4}
                                    {:Students/Volunteering 0 :Students/GradeClass 4}
                                    {:Students/Volunteering 0 :Students/GradeClass 3}])
    => {"Oni koji ne volontiraju:" 11/3}))
=> true
```

### 2. Testiranje prediktivnog modela

Testiranje prediktivnog modela obuhvata:

- **Tačnost predikcija:** Provjerava se tačnost modela upoređujući predikcije sa stvarnim vrijednostima.

```
(facts "Predikcija na osnovu unosa studenta"
  (fact "Predikcija ocjene A"
    (against-background
      (get-user-input) => {:Students/Hours 10 :Students/Absences 2}
      (predict-instance model {:Students/Hours 10 :Students/Absences 2} training-instances) => 0)
      (get-prediction) => "Grade A")

    (fact "Predikcija ocjene C"
      (against-background
        (get-user-input) => {:Students/Hours 5 :Students/Absences 5}
        (predict-instance model {:Students/Hours 5 :Students/Absences 5} training-instances) => 2)
      (get-prediction) => "Grade C")

    (fact "Predikcija ocjene F"
      (against-background
        (get-user-input) => {:Students/Hours 1 :Students/Absences 10}
        (predict-instance model {:Students/Hours 1 :Students/Absences 10} training-instances) => 4)
      (get-prediction) => "Grade F"))
=> true
```

## 6. Zaključak

Ovaj projekat pruža robustan okvir za analizu i vizualizaciju podataka o učenicima. Implementirane funkcionalnosti omogućavaju detaljnu analizu, predikciju i vizualizaciju podataka, što doprinosi boljem razumevanju faktora koji utiču na akademski uspeh učenika. Projekat je razvijen kao odgovor na ličnu želju za istraživanjem, ne samo tehničkih aspekata rada s podacima već i dubljih pitanja koja se odnose na obrazovanje i uspjeh učenika.

Glavni cilj projekta bio je da se istraži u kojoj meri određeni faktori, kao što su obrazovanje roditelja, volontiranje, i vreme provedeno u učenju, utiču na akademski uspjeh. Ovo pitanje je postavljeno jer su uočene razlike u rezultatima među učenicima koji se čine nesrazmernim u odnosu na trud koji su uložili.

Mada nije ostvarena velika preciznost i tačnost modela prilikom predviđanja, na osnovu dobijenih rezultata, moglo bi se zaključiti da faktori na koje učenici nemaju direktni uticaj, kao što je obrazovanje roditelja, nemaju značajan uticaj na njihove ocene, kao ni volontiranje, koje ipak zavisi od samog učenika. S druge strane, faktor kao što je vrijeme provedeno u učenju igra ulogu, ali ono što se pokazalo kao faktor koji najviše utiče jeste odsustvo sa nastave.

Iako data set zbog strukture učenika koji su analizirani i određenih razlika u obrazovnom sistemu u različitim državama, koji su neminovni, nije u potpunosti relevantan i primjenjiv na srednjoškolce u Srbiji, svakako pruža zanimljive uvide nakon vršenja analize i predikcije.

## Preporuke za dalji rad

- **Korištenje drugih modela:** Bilo bi dobro uraditi predviđanje sa drugim modelima, kao što je, recimo, random forest, kako bi se pokušao ostvariti veći nivo preciznosti.
- **Kreiranje korisničkog interfejsa:** Upotreba tehnologije kao što je React za kreiranje interaktivnog korisničkog interfejsa za korisnika.
- **Poboljšanje performansi:** Optimizacija koda i poboljšavanje performansi aplikacije.
- **Proširenje funkcionalnosti:** Razmotrite dodavanje novih funkcionalnosti kao što su napredne analize ili podrška za dodatne formate podataka.
- **Uključivanje AI:** Dodatna integracija vještacke inteligencije za unapređenje prediktivnih modela i analiza.

## 7. Literatura

1. Clojure. Retrieved from <https://clojure.org/>
2. Hickey, R. (2008). *The Clojure Programming Language*.
3. Higginbotham, D. (2015). *Clojure for the Brave and True: Learn the Ultimate Language and Become a Better Programmer*
4. IntelliJ IDEA. Retrieved from <https://www.jetbrains.com/idea/>
5. SQLite Documentation. Retrieved from <https://www.sqlite.org/>
6. Incanter Library. Retrieved from <https://github.com/incanter/incanter>
7. Weka Library. Retrieved from <https://github.com/CalebMacdonaldBlack/clj-weka>
8. Next-jdbc Library. Retrieved from <https://github.com/seancorfield/next-jdbc>
9. Midje Library. Retrieved from <https://github.com/marick/Midje>