Predviđanje uspjeha studenata u sustavu za automatsko ocjenjivanje programskog kôda Edgar

Bedeković, Vedran

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 398

PREDVIĐANJE USPJEHA STUDENATA U SUSTAVU ZA AUTOMATSKO OCJENJIVANJE PROGRAMSKOG KÔDA EDGAR

Vedran Bedeković

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 398

PREDVIĐANJE USPJEHA STUDENATA U SUSTAVU ZA AUTOMATSKO OCJENJIVANJE PROGRAMSKOG KÔDA EDGAR

Vedran Bedeković

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 398

Pristupnik:	**Vedran Bedeković (0036523256)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	prof. dr. sc. Igor Mekterović

Zadatak: **Predviđanje uspjeha studenata u sustavu za automatsko ocjenjivanje programskog kôda Edgar**

Opis zadatka:

Edgar je informacijski sustav za testiranje studenata i učenje razvijen na FER-u. Osnovna funkcionalnost mu je mogućnost strojnog ocjenjivanja programskog kôda u programskim jezicima SQL, C, Java, itd. Osim toga, Edgar podržava pitanja s više ponuđenih odgovora, te još nekoliko vrsta pitanja. Koristi se već niz godina te sadrži podatke o uspješnosti studenata na različitim predmetima koji se mogu iskoristiti kako bi se tehnikama strojnog učenja predviđao uspjeh studenata. Upoznati se s načinom rada Edgara te ustrojem testova i vrstama pitanja. Proučiti trenutne statističke izvještaje i vizualizacije testova. Proučiti različite ustroje predmeta koji koriste Edgar i obratiti pozornost na njihove posebnosti. Osmisliti općeniti parametarski (podesiv) model koji će nastavniku omogućiti definiciju kontinuiranog predviđanja uspjeha studenata. Model mora uvažiti različite ustroje predmeta te omogućiti da se predviđanja objavljuju u unaprijed zadanim intervalima ako su zadovoljeni zadani uvjeti (npr. nakon druge domaće zadaće, ako je student pisao obje domaće zadaće). Omogućiti odabir više različitih algoritama za predviđanje. Sustav izvesti kao samostojeći modul koji će kontinuirano (u podešenim dozvoljenim intervalima) računati opisane podatke. U tu namjenu, razmotriti korištenje programskih jezika R i Python te integraciju s postojećim podsustavom "CodeRunner" za sigurno obavljanje proizvoljnog kôda u Edgaru. Omogućiti pregled izračunatih podataka i uspješnost predviđanja uz odgovarajuće vizualizacije. Pratiti podatke o predviđanjima i uporabu sustava od strane studenata radi budućih analiza. Donijeti ocjenu ostvarenog pristupa.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

Uvod

Kako online platforme za učenje postaju sve više zastupljene i samim time sve veći broj akademskih predmeta koristi dijelom ili u potpunosti te platforme, otvara se mogućnost za provedbu različitih analiza u domeni studentovog obrazovanja. Za razliku od kolegija koji se predaju isključivo uživo, predmeti na tim platformama bilježe dragocjene podatke koji omogućavaju raznovrsne analize koje istraživači nazivaju dubinska analiza podataka u obrazovanju (engl. *Educational Data Mining*, EDM). Na Fakultetu elektrotehnike i računarstva se koristi jedna takva platforma, koja je razvijena na istom fakultetu. Ta online platforma je Edgar, koja je prvobitno bila zamišljena kao sustav za automatizirano ocjenjivanje programskog kôda, ali sve više kolegija počelo ju je koristiti u svrhu provedbe bliceva, ispita ili rokova. Pojedini kolegiji time se u potpunosti ocjenjuju kroz sustav Edgar i jedan od njih je ujedno i kolegij s možda najviše upisanih studenata u jednom semestru. Uvod u programiranje, skraćeno UPRO, provodi sve ispite i zadatke programiranja u sustavu Edgar, s oko 750 studenata po akademskoj godini. Samim time je teško nastavnicima prepoznati studente koji se muče s gradivom sve dok nije prekasno. Zato je potreban sustav koji može prepoznati problematične studente u ranim fazama kontinuirane nastave, kako bi nastavnici reagirali na vrijeme. U ovom radu se nudi i objašnjava programsko rješenje koje proširuje postojeću platformu kako bi se olakšalo nastavnicima pravovremeno prepoznati potencijalne probleme kod pojedinog studenta ili nad cijelom generacijom.

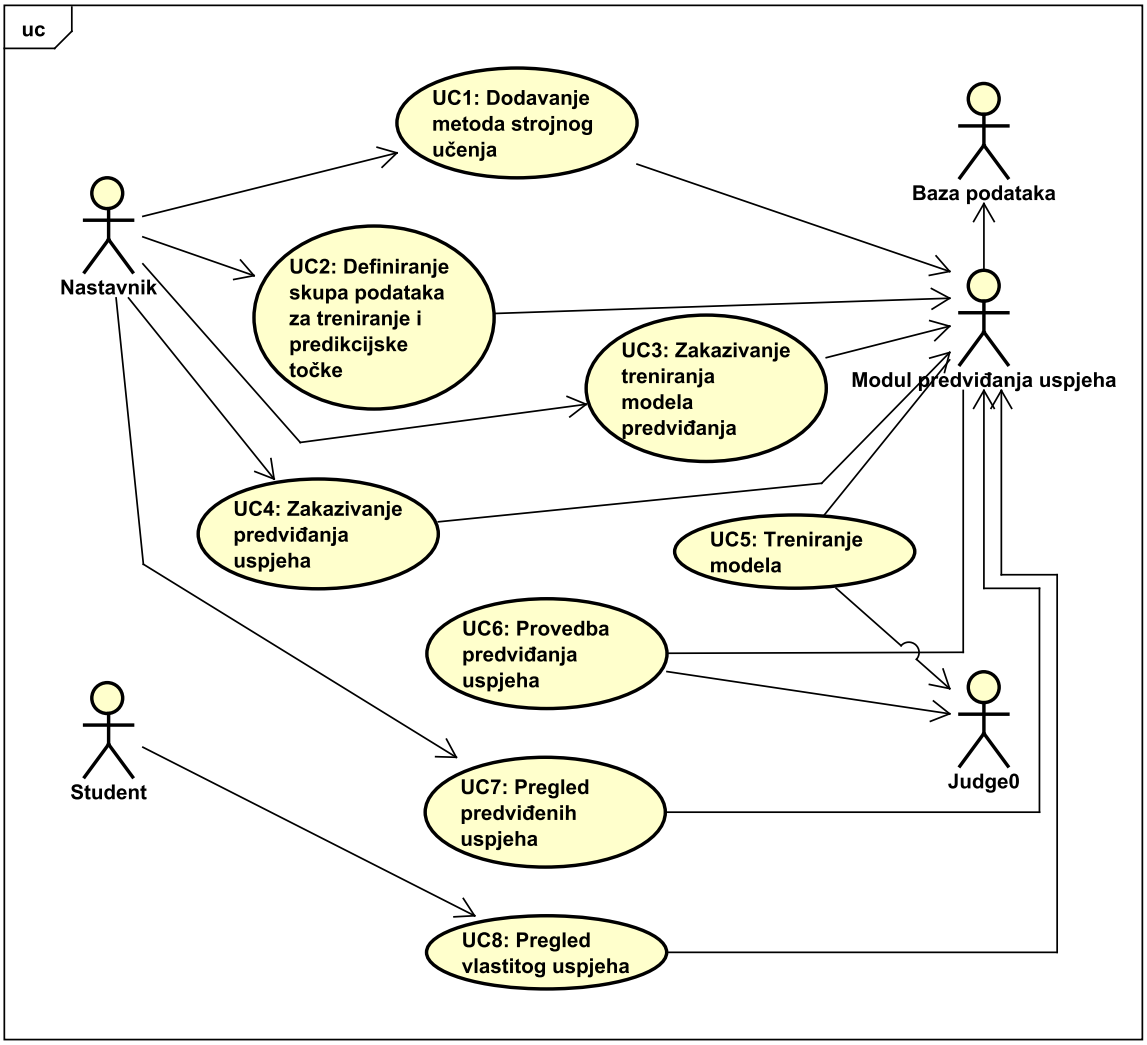
Rad uključuje analizu istraživanja na temu dubinske analize podataka u obrazovanju te se iz naučenog predstavlja sustav usporedbe različitih algoritama za odabir najboljeg modela predviđanja uspjeha za bilo koji kolegij. Prolazi se kroz iteracije tog sustava i daju se zaključci o korisnosti novo dodanih elemenata za porast točnosti predviđanja modela. Zatim se opisuje modul koji se gradi oko tog sustava kako bi se proširile mogućnosti sustava Edgar u domeni dubinske analize podataka. Kako bi nastavničko osoblje moglo upravljati nastalim sustavom, projektirano je korisničko sučelje koje omogućava podešavanje sustava i pregled predviđenih uspjeha. Rad završava primjerom kako se cjelokupni sustav koristi u praksi i kako svi dokumentirani dijelovi komuniciraju međusobno u cilju da predvide postignuća studenata prije završetka kontinuirane nastave.

1. Strojno učenje za predviđanje uspjeha

1.1. Zahtjevi predviđanja uspjeha

Edgar se kao informacijski sustav za testiranje studenata i učenje koristi duži niz godina. Time se neki kolegiji u potpunosti ocjenjuju preko Edgara ili samo poneke cjeline kolegija, ponajviše laboratorijske vježbe rješavanja problema u domeni programiranja. Naravno, dugogodišnjom uporabom zapisan je veliki broj postignuća različitih studenata na različitim kolegijima, ali ti zapisi ne prate isti format kroz različite kolegije, jer svaki se razlikuje u načinu testiranja i u kojoj mjeri koriste spomenuti sustav. Uz to se pojavljuje problem u domeni samog kolegija, gdje se svake akademske godine potencijalno promijeni broj testova[1] i time se remeti konstantnost formata podataka u analiza podataka, što zahtjeva korak generalizacije u njihovom formatiranju. Ciljni zahtjev sustava je dostaviti nastavnicima općeniti pregled uspjeha tekuće generacije studenata nad podacima iz prijašnjih akademskih godina. Samim time je omogućen pronalazak studenata na granici pada kolegija, što bi nastavnicima dalo vrijeme za pomoć istima prije kraja tekuće akademske godine. Općeniti pregled glavnih zahtjeva se može vidjeti na dijagramu (**Slika 1.1**).

[1] Test u sustavu Edgar predstavlja jedan element ocjenjivanja što može uključivati domaću zadaću, laboratorijsku vježbu, ispit i slično.



Slika 1.1 Dijagram obrazaca uporabe - Sustav predviđanja uspjeha

U svrhu dubinske analize podataka za predviđanje konačnog uspjeha studenta u kontinuiranoj nastavi za tekuću akademsku godinu, potreban je sustav koji može raditi paralelno s Edgarom i čitati zapise njegove baze podataka na podesivi način. Takav pristup omogućava nastavniku odabir relevantnih cjelina iz prijašnjih akademskih godina, koje najviše utječu na konačan uspjeh i kako se te prijašnje cjeline podudaraju s cjelinama iz tekuće akademske godine. Obradom tih podataka, sustav za predviđanje može trenirati i unakrsno usporediti modele različitih metoda strojnog učenja s različitim parametrima, što omogućava predviđanje hoće li pojedini student iz tekuće akademske godine pasti godinu i s kojom ocjenom okončati kontinuiranu nastavu. Te modele je potrebno spremiti kako bi se mogli ponovo koristiti za vrijeme predviđanja bez trošenja resursa za njihovo ponovo treniranje te se rezultati predviđanja i ostale konfiguracije spremaju u bazu podataka sustava

Edgar u zasebnoj shemi. Proširenjem Edgarovog aplikacijskog programskog sučelja omogućava se dohvat tih rezultata bez sustava predviđanja uspjeha. Time se upotpunjuje zahtjev da sustav predviđanja treba biti zaseban servis, koji ne mora biti u pogonu kako bi sustav Edgar i dalje obnašao svoju glavnu ulogu, a nastavnik i dalje ima uvid u već izvršena predviđanja neovisno o dostupnosti samog sustava. Dodatno, sustav za predviđanje mora odrediti u kojem okruženju će izvršavati programski kôd za treniranje i predviđanje. Dvije opcije su lokalno izvršavanje gdje je i sami servis postavljen ili pozivanjem na servis izvršavanja programskog kôda u izolaciji imena Judge0 [11], koji i sustav Edgar koristi u svrhu testiranja (potencijalno malicioznog) programskog kôda koji pišu studenti.

Za pregled predviđenih uspjeha, a i za intuitivno podešavanje modela predikcije, potrebno je korisničko sučelje u kojem nastavnik može:

- unijeti svoje metode strojnog učenja s parametrima po vlastitom nahođenju ili istraživanju
- odrediti broj predikcija tijekom tekuće akademske godine i kada će se te predikcije odrediti i prikazati nastavniku i samim studentima.
- definirati skup podataka za treniranje po prijašnjim akademskim godinama i prikladnim cjelinama kolegija
- odrediti datume i vrijeme u kojima će se odvijati treniranje modela radi očekivanog računarskog opterećenja za treniranje i usporedbu različitih modela predikcije
- odrediti datum i vrijeme u kojima će se odvijati predviđanje uspjeha iz prije treniranih modela
- vidjeti općenit i detaljan pregled predviđanja na kolegiju

Spomenuti pregled rezultata predikcije je moguć na dva načina. Generalni prikaz je nadzorna ploča s popratnim vizualizacijama koje daju općeniti pregled predviđanja za trenutnu akademsku godinu te detaljan prikaz gdje nastavnik vidi detaljna predviđanja o ishodu pada ili prolaza te ocjene za svakog studenta, s popratnim vjerojatnostima svakog ishoda u tabličnom formatu s poželjnim filtriranjem prikazanih studenata. Kako bi sučelje bilo funkcionalno potrebno je dodatno proširenje Edgarovog aplikacijskog korisničkog sučelja za posluživanje samog sučelja i za spajanje sučelja na zapisane podatke iz baze podataka te upravljanje sustavom za predviđanje preko Edgara.

Konačni zahtjev je prikazati samom studentu njegov predviđeni uspjeh s popratnim vjerojatnostima za pojedini ishod. Ovdje nema potrebe za robusnim sučeljem kao prethodno

spomenuto nastavničko sučelje, već je Edgarov sustav za prikaz statičkih web stranica s odabranim vizualizacijama dovoljan, jer student jedino može dohvatiti svoje podatke predviđanja, ako su mu dostupni, i nema ovlasti upravljati sustavom predviđanja uspjeha.

1.2. Povezani radovi

1.2.1. Radovi o dubinskoj analizi podataka u obrazovanju

Potrebno je istražiti koji parametri tj. osobine i okolina studenta najviše utječu na studentov uspjeh. Prolaskom više izvora mogu se vidjeti velike razlike u odabiru, a i samom opsegu uzetih parametara. Neki izgrađeni modeli osnivali su se samo na dostupnim podacima studenata koje je sveučilište već imalo pri ruci, dok su ostali istraživači odlučili prolaziti kroz veliku količinu značajka za koje su trebali anketirati same studente, a time i pravilno pročistiti podatke za potrebnu elektroničnu obradu i modeliranje. Naravno, samo pitanje koji čimbenici ponajbolje otkrivaju studentovo buduće postignuće izlazi van područja računalne znanosti te se ne može još točno odrediti kako najpreciznije predvidjeti uspjeh na nekom kolegiju. Zasad se mogu dobiti grube procjene koje se koriste kao dobra podloga u obrazovanju studenata.

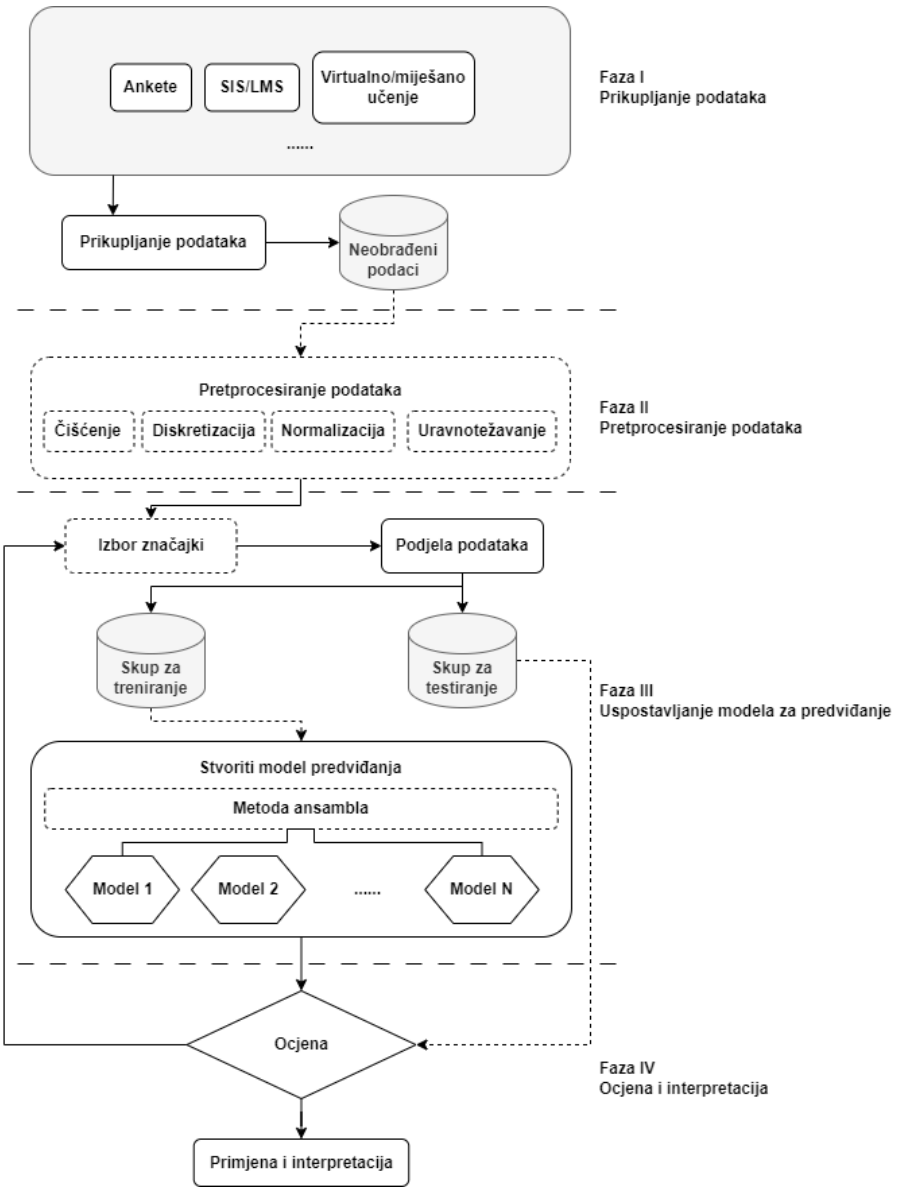
Analiza motivacije, stava i ponašanja studenta bitan je aspekt dizajniranja učinkovitih EDM modela. Poput znanstvenika koji proučavaju ponašanje svojih ispitanika kako bi ih bolje razumjeli, EDM znanstvenici mjere ponašanje studenta. Provedena su različita istraživanja u ovoj domeni kako bi se razumjela njihova motivacija prema postignućima u učenju i njihova epistemološka uvjerenja o stjecanju znanja. Međutim, još uvijek postoje mnoga područja koja treba istražiti za dobrobit obrazovnog sustava.

Jedno otkriće sugerira da studenti koji su slabi u određenom kolegiju ne žele sjediti u prvom redu na predavanjima [12]. Ovo opažanje je zanimljivo jer implicira da na studentovu motivaciju i stav prema učenju mogu utjecati čimbenici okoline kao što je položaj sjedenja. Nadalje, istraživanje je pokazalo da ako student sjedi što je moguće više u zadnjim redovima, uči sporije. Time možemo pridonijeti novu dimenziju u EDM modele, jer ti čimbenici mogu pokazati studentovu motivaciju za kolegij. Drugi način za analizu ponašanja studenta temelji se na tome koliko se puta student dnevno prijavljuje i pregledava stranice za e-učenje. Ti se podaci mogu koristiti za razumijevanje pojedinog angažmana i motivaciju

za učenje. Osim toga, koliko studenti cijene znanje i učenje može utjecati na njihov uspjeh [12]. Stoga je važno razumjeti da svako ima vlastiti proces učenja i to treba imati na umu pri izradi modela.

Dodatno, postoje različiti pristupi rješavanju nedostajućih vrijednosti atributa, uključujući odbacivanje n-torki, korištenje globalne konstante ili srednje vrijednosti atributa za popunjavanje vrijednosti koje nedostaju [13]. U kontekstu praćenja uspjeha studenta, ako student propusti ispit ili sličan događaj, podaci koji nedostaju mogu biti uklonjeni iz procesa modeliranja. Alternativno, može se tretirati kao da je student dobio nula bodova za taj ispit u procesu modeliranja ili se vrijednost koja nedostaje može zamijeniti prosječnom vrijednošću ispita. Izbor načina rukovanja vrijednostima koje nedostaju ovisi o specifičnoj situaciji i ciljevima analize.

Istraživački rad W. Xiaova i drugih istraživača [35] predočava 4 glavna koraka izrade modela predikcije. Ti koraci su: sakupljanje podataka, pretprocesiranje podataka, sastavljanje modela predikcije te na kraju provjera i interpretacija. Pojedinačni koraci su objašnjeni u nastavku, a prikazani su i na slici (**Slika 1.2**).



Slika 1.2 Postupak uspostavljanja modela predikcije (koraci prikazani isprekidanim okvirima nisu obavezni), izvor: [35]

Primarni cilj prikupljanja podataka je prikupljanje neobrađenih podataka iz nekoliko izvora za izradu modela predviđanja. Ovi izvori obuhvaćaju Studentske informacijske sustave (engl. *Student Information Systems*, SIS), Sustave za upravljanje učenjem (engl. *Learning Management System*, LMS) i različite vrste podataka proizvedene u digitalnim ili kombiniranim postavkama učenja kao što su Javno dostupni online tečajevi (engl. *Massive Open Online Course*, MOOC), obrazovne igre i drugi. Osim prikupljanja podataka iz digitalnih sustava ili okruženja, studentove podatke moguće je dobiti i anketiranjem. S obzirom da su većina klasifikacijskih algoritama strojnog učenja koji se koriste za razvoj

modela prognoze studentove uspješnosti nadzirani algoritmi učenja, prikupljeni podaci moraju biti klasificirani ručno ili automatski.

Obrada podataka uključuje poboljšanje kvalitete neobrađenih podataka prikupljenih u prethodnoj fazi rješavanjem nekoliko problema. U izvornim podacima mogu nedostajati podaci, mogu postojati pogreške, dupliciranja ili smetnje, što zahtijeva čišćenje podataka radi rješavanja nedosljednosti. Diskretizacija podataka koristi se za pretvaranje kontinuiranih vrijednosti, kao što su dob ili rezultat, u diskretne vrijednosti kako bi se ispunili zahtjevi mnogih algoritama strojnog učenja. Normalizacija skalira vrijednosti svake značajke na određeni raspon, kao što je [-1, 1] ili [0, 1], kako bi se izbjegao utjecaj različitih veličina između značajki na algoritam klasifikacije. Neuravnoteženi podaci, kao što su studentske ocjene, gdje se broj skupova koji pripadaju različitim kategorijama u neobrađenim podacima znatno razlikuju, mogu dovesti do niske točnosti (1) uvježbanog modela predviđanja. Stoga je balansiranje podataka potrebno kako bi podaci bili gotovo jednaki među različitim kategorijama, čime se poboljšava kvaliteta modela predviđanja.

$$točnost = \frac{broj\ točnih\ predikcija}{broj\ svih\ predikcija} \tag{1}$$

Kako bi se stvorio točan model predikcije uspješnosti studenta, istraživači moraju odabrati najznačajnije značajke iz prethodno obrađenih podataka kako bi ublažili utjecaj "prokletstva visoke dimenzionalnosti"[2]. Odabir značajki mogu ručno provoditi stručnjaci na temelju njihovog znanja ili automatski pomoću algoritama za odabir značajki. Proces dijeljenja podataka koristi se za podjelu podataka na podatke za obuku i podatke za testiranje, gdje se jedan dio koristi za obuku modela predviđanja, dok se drugi koristi za njegovu procjenu. Kada se koriste nenadzirane metode strojnog učenja, korak dijeljenja podataka nije potreban. Kako bi poboljšali točnost predviđanja, istraživači mogu koristiti ansamble metode za integraciju višestrukih modela predviđanja stvorenih pomoću različitih algoritama klasifikacije.

[2] Prokletstvo dimenzionalnosti odnosi se na različite fenomene koji nastaju prilikom analize i organiziranja podataka u visokodimenzionalnim prostorima koji se ne pojavljuju u niskodimenzionalnim okruženjima kao što je trodimenzionalni fizički prostor iz svakodnevnog života. Izraz je skovao Richard E. Bellman [7].

Za procjenu učinkovitosti modela razvijenog u prethodnoj fazi, ključno je provesti evaluaciju modela. Nalazi ove evaluacije mogu pomoći istraživačima da identificiraju načine poboljšanja odabira značajki i drugih taktika. Model predikcije koji zadovoljava standarde ocjenjivanja može se koristiti samostalno ili u kombinaciji s drugim sustavima, kao što su LMS i diplomski kvalifikacijski ispiti, kako bi se poboljšala razumljivost ishoda predviđanja.

Naravno, važno je održavati anonimnost studenata [27] pri anketiranju ili pri uzimanju pohranjenih podataka iz baze, ali potrebno je i pravilno usmjeriti pažnju na moguće pristranosti podataka, koje mogu uzrokovati pristranost i samog EDM modela. Q. Hu i H. Rangwala su istraživali kako napraviti ravnopravan EDM model u svom radu [16]. Kako bi riješili problem pristranosti u modelima predviđanja, proučavali su individualnu pravednost. Individualna pravednost zahtijeva da se sa sličnim pojedincima postupa na sličan način, bez obzira na osjetljive osobine koje imaju. Međutim, definiranje i mjerenje sličnosti na način koji jamči jednakost izazovan je problem koji zahtijeva donošenje strogih pretpostavki, ograničavajući primjenjivost individualne pravednosti u mnogim slučajevima. Njihovo istraživanje se usredotočuje na individualnu pravednost bez mjernih podataka, što znači da se studentova predikcija uspjeha ne bi trebala promijeniti ako se njegovi osjetljivi podaci promijene. Predloženi pristup temelji se na ideji da na kvalifikaciju pojedinca ne bi trebala utjecati njegova osjetljiva osobina. Stoga promjena studentovog osjetljivog atributa ne bi trebala promijeniti predviđanje klasifikatora, osiguravajući individualnu jednakost. Razvijanjem individualnog okvira pravednosti bez metrike, istraživači se mogu bolje posvetiti pristranošću u modelima predviđanja i osigurati da se prema svim studentima postupa pošteno, bez obzira na njihove osjetljive osobine. Ovaj pristup ima važne implikacije za obrazovanje, gdje su točna predviđanja uspješnosti studenta presudna za intervenciju nastavnika i poboljšanje ishoda učenja. Osiguravajući pravedna i nepristrana predviđanja sprječava se pad morala studenata kojima je dana kriva negativna predikcija na bazi njihove rase ili socijalnog statusa. Baza podataka sustava Edgar ne sadrži takve podatke o studentima i ne koriste se za treniranje modela u ovom radu.

1.2.2. Radovi o usporedbi modela predikcije akademskog uspjeha

Za učenje EDM modela mogu se koristiti podaci koji su već pohranjeni u sveučilišnoj bazi podataka, kao što je to napravio M. Yağcı u svom radu [36]. Uzeo je podatke o ostvarenim bodovima na međuispitu i završnom ispitu. Analizirajući ove podatke, istražio

je utjecaj studentovog uspjeha u sredini semestra na njegov uspjeh na kraju semestra. Iako takva analiza može dati dobra predviđanja, za točno modeliranje studenta i preciznija predviđanja potrebno je prikupiti više podataka. Usporedio je performanse između algoritama strojnog učenja kao što su algoritam nasumične šume (RF), algoritam najbližeg susjeda (NN), strojevi potpornih vektora (SVM), logistička regresija (LR), naivni Bayes (NB) i k-ti najbliži susjed (KNN) za predviđanje ocjena završnog ispita. Istraga se fokusirala na dvije teme, predviđanje akademskog uspjeha na temelju prethodnih postignuća i usporedbu pokazatelja uspješnosti algoritama strojnog učenja. Predloženi model postigao je točnost klasifikacije raspona postignutih bodova završnog ispita od 70-75%, što pokazuje da su ocjene međuispita važan prediktor ocjena završnih ispita. Na kraju se pokazalo da algoritmi RF, NN i SVM imaju najveću točnost klasifikacije, dok KNN ima najmanju. Rezultati govore da ovi algoritmi mogu dovoljno točno predvidjeti ocjene studentovog akademskog uspjeha. Nadalje, predviđanja su napravljena korištenjem samo tri kategoričke značajke: raspon postignutih bodova međuispita, pod koji zavod spada kolegij za koji se piše ispit i upisani fakultet. Potvrdio je da se algoritmi strojnog učenja mogu koristiti za predviđanje akademskog uspjeha, a nastavno osoblje može imati koristi od ovog istraživanja u ranom prepoznavanju studenta ispod ili iznadprosječne akademske motivacije.

U A. A. Saaovom radu [29] korištena su četiri algoritma stabla odluke na prikupljenim studentovim podacima, naime C4.5, ID3, CART i CHAID. Nadalje, deseterostruka unakrsna provjera valjanosti korištena je za provjeru i validaciju ishoda korištenih algoritama te računanje točnosti i preciznosti. Analiza jasno pokazuje da neki algoritmi nadmašuju druge. Preciznije, CART je pokazao najbolju točnost predviđanja ocjene od 40%, što je znatno više od očekivane točnosti zadanog modela. CHAID i C4.5 bili su sljedeći na redu s 34,07% odnosno 35,19% točnosti, dok je ID3 imao najnižu točnost od 33,33%. Većina navedenih algoritama suočavala se s poteškoćama u razlikovanju sličnosti između klasa, što je rezultiralo klasificiranjem više studenata najsličnijim klasama ispravnoj klasi, umjesto nje same. Na primjer, klasa "dobar" u CART matrici zabune je pokazala da je 38 objekata od 68 klasificirano kao "vrlo dobar", gornja najbliža klasa u smislu ocjena, dok je 18 objekata klasificirano kao "dovoljan", najbliža donja klasa u smislu ocjena. Ovo zapažanje sugerira da je diskretizacija atributa klase bila neadekvatna u hvatanju razlika u drugim značajkama ili su same značajke bile nejasne u hvatanju takvih razlika. Drugim riječima, razredi korišteni u ovom istraživanju nisu bili potpuno neovisni, budući da

"odličan" student može imati iste karakteristike kao i "vrlo dobar" student, što zbunjuje algoritam klasifikacije i značajno utječe na njegovu izvedbu i točnost.

U radu I. Mekterovića i Lj. Brkić [23], Automatizirani sustav za evaluaciju programskog kôda (engl. *Automated Programming Assessment System*, APAS) postavljen je za kolegij Baze podataka kako bi se smanjilo radno opterećenje nastavnika i pružila pravovremena povratna informacija studentima. Koristili su obrazac ocjenjivanja "malo i često" kako bi predvidjeli studentov uspjeh tijekom kolegija. Podaci za predviđanje uključivali su unutarnje varijable (prikupljene tijekom izvođenja kolegija) i vanjske varijable (poput stečenih ECTS bodova). Njihova analiza otkrila je da vanjske varijable, iako korisne, nisu značajno utjecale na točnost predviđanja. S 80% točnosti u predviđanju prolaznosti nakon sredine semestra, njihov je model omogućio proaktivan pristup pomoći rizičnim studentima. Također su identificirali vrijedne uvide za poboljšanje strukture zadataka i sadržaja kolegija. Koristeći tehnike dubinske analize podataka kao što su stabla odluke, nasumične šume, strojevi potpornih vektora i logistička regresija, istraživanje je pokazalo da su stabla odluke bila posebno učinkovita u predviđanju studentovog uspjeha, postižući točnost predviđanja postotka prolaza do 80% s internim varijablama. Ovo istraživanje pokazuje korisnost APAS-a u poboljšanju učinkovitosti podučavanja i uspjeha studenata u kolegijima programiranja.

1.3. Razmatrane tehnologije

Za provedbu algoritama strojnog učenja programski jezik Python 3 je najzastupljeniji, pogotovo u domeni dubokog učenja. Za razliku od njega, programski jezik R je optimiziran za rad u domeni analize podataka i statistike, ali se najzastupljenije metode strojnog učenja mogu i dalje koristiti pomoću vanjskih biblioteka. Isto vrijedi i za Python, ali njegove vanjske biblioteke su podržane većim doprinosima javnosti radi njegove popularnosti. Unatoč tome što u R-u postoje i dalje podržane biblioteke strojnog učenja, domena neuronskih mreža više nema modernu potporu kao i Python. Pošto sustav Edgar već koristi R za neke statističke izračune, ovaj rad se isto bazira na istom programskog jeziku, ali dodatno pomoću Pythona se upotpunjuje sposobnost usporedbe što većeg broja algoritama strojnog učenja, ponajviše neuronskih mreža.

Sustav predviđanja, radi dosljednosti, se piše u Node.js okruženju s Express.js bibliotekom, isto kao sustav Edgar. Taj odvojen sustav bi se spajao na istu PostgreSQL bazu podataka kao i Edgar, gdje je potrebno dodati nove tablice za praćenje treniranih modela i predviđanja uspjeha. Te tablice su uvrštene u pripadajuću shemu napravljenu za isticanja odvojenosti Edgara od sustava predviđanja uspjeha. Posluženi servis predviđanja ne mora znati za sustav Edgar te Edgar može raditi kada i sustav predviđanja nije u funkciji.

Naravno, API Edgar sustava mora biti proširen dodatnim krajnjim točkama kako bi poslužio i osposobio korisničko sučelje za upravljane modelima predikcije i pregledu rezultata. To korisničko sučelje je, radi kompleksnosti, izvedeno kao jednostranična web aplikacija pisana pomoću Angular radnog okvira, kao i ostale jednostranične aplikacije koje Edgar poslužuje.

2. Dubinska analiza podataka

Dubinska analiza podataka u obrazovanju interdisciplinarno je istraživačko područje koje primjenjuje teorije i metode iz strojnog učenja, statistike, dubinske analize podataka, obrazovne psihologije, kognitivne psihologije i drugih polja za analizu obrazovnih podataka. Jedno od najvažnijih pitanja u području EDM-a je predviđanje uspjeha studenata. To može pomoći u prepoznavanju rizičnih studenata i dati preporuke za personalizirano učenje. Osnovni proces predviđanja uspješnosti studenta korištenjem EDM-a uključuje prikupljanje povijesnih akademskih zapisa, njihovo označavanje razinom uspješnosti ili prosjekom i uspostavljanje modela predviđanja. Regresijska analiza i klasifikacija dvije su uobičajene metode koje se koriste u predviđanju uspjeha [28]. Regresijska analiza pronalazi odnos između zavisne varijable i jedne ili više nezavisnih varijabli. S druge strane, klasifikacija je postupak u kojem se pojedinačne stavke svrstavaju u skupine na temelju kvantitativnih informacija o jednoj ili više karakteristika svojstvenih značajki i na temelju skupa podataka za obuku prethodno označenih stavki. Koristeći ove metode istraživači mogu predvidjeti studentov akademski uspjeh ili što ranije identificirati one koji su izloženi riziku od akademskog neuspjeha. Predviđanje uspješnosti studenta također može pružiti osnovu za personalizirano učenje i podržati administraciju obrazovne institucije u donošenju odluka analizom čimbenika koji utječu na studente.

Računala se u obrazovanju koriste od sredine 20. stoljeća, uglavnom za učenje uz pomoć računala (engl. *Computer-Assisted Learning* CAL) ili obuku temeljenu na računalu (*eng. Computer-Based Training* CBT). Međutim, te su metode bile ograničene na niže razine Bloomove taksonomije [1], a CBT je izašao iz mode 1980-ih zbog nemogućnosti rukovanja vještinama razmišljanja na višoj razini. U kasnim 1980-ima mnogi su znanstvenici počeli eksperimentirati s računalima u obrazovanju za komunikaciju između učenika i nastavnika te među studentima. World Wide Web napravio je revoluciju u online nastavi i okruženjima za e-učenje, a prvi sustavi za upravljanje učenjem razvijeni su 1995. godine. Prva upotreba izraza "dubinska analiza podataka u obrazovanju" bila je 2005. godine na radionici o analizi obrazovnih podataka tijekom godišnje konferencije Udruge za unapređenje umjetne inteligencije (AAAI'05). Radionica je bila usmjerena na tehničke aspekte prikupljanja i analize podataka u informatičkom obrazovanju i obuci računalnih znanosti. Godine 2009. osnovani su časopis i međunarodna konferencija o EDM-u, što je dalo podosta pažnje za ljude u obrazovnim ustanovama. Danas EDM omogućava otkrivanje novih uvida na temelju

podataka o korištenju obrazovnih sustava za provjeru/evaluaciju i poboljšanje aspekata obrazovanja. EDM je također doveo do učinkovitijeg procesa učenja, budući da omogućava predviđanje uspješnosti učenika kroz regresijsku analizu i klasifikaciju na temelju kvantitativnih informacija u vezi s jednom ili više karakteristika svojstvenih značajki i na temelju skupa podataka za obuku [3].

2.1. Metode strojnog učenja

Strojno učenje je disciplina umjetne inteligencije fokusiran na razvoj statističkih algoritama koji uče i generaliziraju nad nekim skupom podataka, što im omogućava izvršavanje zadataka za koje su trenirani [17]. Iako se svo strojno učenje ne temelji na statistici, računalna statistika igra ključnu ulogu, jer se disciplina gradila na metodama iz domene matematičkog programiranja. Blisko povezano područje je dubinska analiza podataka, s naglaskom na istraživačku analizu podataka kroz učenje bez nadzora. Nadolazeća poglavlja opisuju odabrane metode strojnog učenja koje se obično koriste u dubinskoj analizi podataka.

2.1.1. Stabla odluke i algoritam nasumične šume

Stablo odluke je nadzirani algoritam strojnog učenja koji se koristi za klasifikaciju i regresiju podataka. Algoritam funkcionira kao hijerarhijsko stablo, gdje se svaki unutarnji čvor u stablu predstavlja atributom, a svaki list predstavlja klasifikacijski ili regresijski izlaz. Proces izgradnje stabla počinje s korijenom, gdje se podaci dijele na pod skupove temeljem vrijednosti atributa. Taj postupak se ponavlja sve dok se ne dostigne krajnji list, koji sadrži jedinstvenu klasifikacijsku ili regresijsku vrijednost [9]. Učenje stablom odluke je često korištena metoda u dubinskoj analizi podataka.

Prednosti stabla odluke su jednostavnost za interpretaciju i objašnjenje, jer se lako vizualiziraju te brzo obrađuje podatke i prihvaća nedostajuće vrijednosti. Također se može koristiti za rukovanje velikim skupovima podataka i može se kombinirati s drugim algoritmima.

Međutim, stablo odluke je osjetljivo na prenaučenost, što može dovesti do smanjene sposobnosti generalizacije na novim podacima [14]. Također, izgradnja optimalnog stabla odluke može biti zahtjevno i traži mnogo računalnih resursa.

Algoritam nasumične šume (engl. *random forest*, RF) je dobro poznati algoritam strojnog učenja koji spada u kategoriju nadziranog učenja. Svestran je i može se koristiti za zadatke klasifikacije i regresije u strojnom učenju. Temelj nasumične šume je koncept skupnog učenja, koji uključuje kombiniranje više klasifikatora za rješavanje složenih problema i poboljšanje performansi modela.

Kao što naziv sugerira, algoritam je klasifikator koji se sastoji od više stabala odluke treniranih na različitim podskupovima danog skupa podataka. Objedinjenjem predviđanja ovih stabala i korištenjem većinskog glasovanja, algoritam dolazi do konačnog rezultata. Ovaj pristup umanjuje oslanjanje na jedno stablo odluke i poboljšava točnost predviđanja.

Stabla odluke i regresijska stabla, koja služe kao temeljni modeli u nasumičnim šumama, osjetljiva su na prenaučenost. Kada dođe do prenaučenosti, model pokazuje visoku pristranost i donosi netočne odluke prilikom klasificiranja novih podataka. Kao rezultat toga, završava se modelom visoke varijance. S druge strane, stabla odluke imaju sposobnost uhvatiti složene odnose između atributa i ciljane klase, što je ključno za učinkovitost ansambla. Cilj algoritma je minimalizirati korelaciju između pojedinačnih stabala kako bi se učinkovito uhvatile varijacije podataka bez povećanja ukupne varijance grupe [8]. Nadalje, povećanje broja stabala u algoritmu povećava točnost i pomaže u sprječavanju problema prenaučenosti, gdje model može preblizu odgovarati podacima za obuku, ali se možda neće dobro generalizirati na nove podatke.

2.1.2. Logistička regresija

Logistička regresija je široko korišten algoritam strojnog učenja koji spada u tehniku nadziranog učenja. Posebno je dizajniran za predviđanje kategorički zavisnih varijabli na temelju danog skupa nezavisnih varijabli.

Za razliku od linearne regresije, koja se koristi za rješavanje problema regresije, logistička regresija se koristi za zadatke klasifikacije. Predviđa izlaz kao kategoričku ili diskretnu vrijednost, kao što je DA ili NE, 0 ili 1 te istina ili laž. Umjesto pružanja točnih vrijednosti 0 ili 1, logistička regresija proizvodi distribuirane vrijednosti koje padaju između 0 i 1 [22].

Logistička regresija značajan je algoritam strojnog učenja zbog svoje sposobnosti pružanja vjerojatnosti i klasificiranja novih podataka pomoću kontinuiranih i diskretnih skupova podataka. Široko se koristi u raznim područjima za rješavanje problema klasifikacije s kategoričkim ishodima [21].

Sigmoidna funkcija je matematička funkcija koja se koristi u logističkoj regresiji za preslikavanje predviđenih vrijednosti u vjerojatnosti. Pretvara bilo koju stvarnu vrijednost u vrijednost unutar raspona od 0 i 1, tvoreći krivulju u obliku slova "S". Ova krivulja je također poznata kao sigmoidna ili logistička funkcija [22].

U logističkoj regresiji, vrijednost praga koristi se za određivanje vjerojatnosti od 0 ili 1. Vrijednosti iznad praga obično se predviđaju kao 1, dok vrijednosti ispod praga imaju tendenciju predviđanja kao 0. Sigmoidna funkcija pomaže osigurati da predviđene vrijednosti logističke regresije spadaju u željeni raspon vjerojatnosti od 0 do 1.

2.1.3. K najbližih susjeda

K najbližih susjeda (KNN) jednostavan je algoritam nadziranog učenja koji se koristi u strojnom učenju. Za kategorizaciju novih podataka oslanja se na sličnost između novih i postojećih podataka.

KNN pohranjuje sve dostupne podatke i klasificira nove podatkovne točke na temelju njihove sličnosti s postojećim kategorijama. To omogućuje jednostavnu klasifikaciju novih podataka u odgovarajuće kategorije pomoću KNN-a. Nadalje, može se koristiti i za regresijske i za klasifikacijske zadatke, ali se obično koristi za klasifikaciju. To je neparametarski algoritam, što znači da ne donosi nikakve pretpostavke o temeljnoj distribuciji podataka.

Ujedno, to je tehnika klasifikacije koja lokalno aproksimira funkciju i odgađa izračun do evaluacije. Pritom normaliziranje ispitnih podataka može znatno poboljšati točnost kada značajke imaju različite jedinice ili skale [15].

Susjedi se biraju iz skupa objekata s poznatim vrijednostima klase ili svojstava, koji služe kao skup za obuku algoritma. Međutim, za razliku od mnogih drugih algoritama strojnog učenja, KNN ne zahtijeva eksplicitni korak obuke. Umjesto toga, pohranjuje podatke za učenje i odgađa izračunavanje do evaluacije funkcije. To čini KNN algoritmom „lijenog učenika“ [19] koji izvodi radnje na skupu podataka za vrijeme klasifikacije.

Standardna shema skaliranja u KNN je dodjeljivanje težine $1/d$ svakom susjedu, gdje je d udaljenost do susjeda. To znači da bliži susjedi imaju veće vrijednosti i više doprinose prosjeku nego oni udaljeniji. Ovo može biti korisno u slučajevima kada su neki susjedi relevantniji od drugih i može poboljšati točnost algoritma.

2.1.4. Stroj potpornih vektora

Stroj potpornih vektora (engl. *Support Vector Machine*, SVM) vrsta su modela nadziranog učenja koji se koristi za klasifikaciju i regresijsku analizu u strojnom učenju. SVM-ovi imaju pridružene algoritme učenja koji analiziraju podatke i grade model na temelju skupa podataka za treniranje. Ovaj se model zatim može koristiti za klasificiranje novih primjera u jednu od dvije kategorije, čineći SVM neprobabilističkim binarnim linearnim klasifikatorom, iako se metode probabilističke klasifikacije kao što je Plattovo skaliranje također mogu koristiti sa SVM-ovima.

Cilj SVM-a je maksimizirati širinu jaza između dviju kategorija u podacima o obuci preslikavanjem primjera na točke u prostoru. Ovo stvara jasnu razliku između kategorija, a novi primjeri mogu se predvidjeti da će pripadati kategoriji na temelju toga na kojoj su strani jaza projicirani. Uz linearnu klasifikaciju, SVM-ovi također mogu učinkovito izvesti nelinearnu klasifikaciju koristeći tehniku nazvanu "trik jezgre", koja implicitno preslikava ulaze u višedimenzionalne prostore značajki za složenije klasifikacije zadataka.

Nadalje, može se kategorizirati u dvije vrste: linearni SVM i nelinearni SVM. Linearni SVM koristi se za skupove podataka koji se mogu razdvojiti u dvije klase pomoću jedne ravne linije, a korišteni klasifikator naziva se Linearni SVM klasifikator. S druge strane, nelinearni SVM koristi se za skupove podataka koji se ne mogu klasificirati pomoću ravne linije, a korišteni klasifikator naziva se Nelinearni SVM klasifikator.

Detaljniji primjer kako SVM funkcionira može se vidjeti u [31].

2.1.5. Algoritam pojačavanja gradijenta

Pojačavanje gradijenta moćna je skupna tehnika strojnog učenja osmišljena za poboljšanje izvedbe samostalnih klasifikatora, kao što su stabla odluke, njihovim kombiniranjem u ansambl. Primarni cilj povećanja gradijenta je minimiziranje dane funkcije

gubitka iterativnim dodavanjem modela koji ispravljaju pogreške postojećeg kombiniranog modela. Proces počinje prilagođavanjem jednostavnog modela podacima za treniranje. U sljedećim iteracijama dodaju se novi modeli koji predviđaju reziduale, tj. pogreške, kombiniranog modela iz prethodne iteracije. To se postiže prilagođavanjem novih modela gradijentima funkcije gubitka s obzirom na predviđanja, otuda i naziv "pojačavanje gradijenta".

Algoritmi za pojačavanje gradijenta rade na stupnjevit način, pri čemu svaki stupanj uključuje prilagođavanje novog modela pseudorezidualima, koji su gradijenti funkcije gubitka s obzirom na predviđanja modela. Kombinirani model se zatim ažurira dodavanjem novog modela, skaliranog parametrom stope učenja. Ovaj se proces nastavlja sve dok se ne postigne određeni broj ponavljanja ili prestane rast performansi. Tehnike regularizacije, kao što je smanjenje stope učenja i korištenje nasumičnih podskupova podataka, često se koriste kako bi se spriječila prenaučenost i poboljšala generalizacija. Pojačavanje gradijenta prošireno je u razne napredne implementacije, uključujući XGBoost, LightGBM i CatBoost, a svaka uvodi optimizacije za brzinu i točnost [5].

2.1.6. Neuronske mreže

Neuronske mreže ključni su alati u strojnom učenju, pokreću napredne algoritme i aplikacije u područjima kao što su računalni vid, obrada prirodnog jezika i robotika. Sastoje se od međusobno povezanih neurona raspoređenih u slojeve: ulazni, skriveni i izlazni sloj. Svaki neuron obrađuje ulazne signale pomoću aktivacijske funkcije kako bi proizveo izlaznu vrijednost. Neuroni u različitim slojevima povezani su vezama, gdje varijable težine utječu na snagu signala između neurona. Proces obuke uključuje propagaciju unaprijed, pri čemu ulazni podaci prolaze kroz mrežu, i propagaciju unatrag, algoritam koji iterativno prilagođava težine i pristranosti kako bi minimizirao funkciju gubitka. Funkcija gubitaka mjeri razliku između predviđanja modela i stvarnih ishoda, usmjeravajući proces optimizacije, često korištenjem gradijentnog spuštanja, kako bi se poboljšala izvedba mreže [18].

Višeslojni perceptroni (engl. *multilayer perceptron*, MLP) su vrsta neuronske mreže s unaprijednim prijenosom podataka i među najpoznatijim su i najčešće korištenim neuronskim mrežama. Sastoje se od više slojeva neurona, uključujući ulazni sloj, jedan ili

više skrivenih slojeva i izlazni sloj. Signali u MLP-ovima prenose se u jednom smjeru, od ulaza do izlaza, bez formiranja petlji. Skriveni slojevi u MLP-ovima omogućuju mreži da nauči složene obrasce transformiranjem ulaznih podataka kroz višestruke faze izračuna. Svaki neuron u skrivenim slojevima primjenjuje nelinearnu aktivacijsku funkciju na zbroj svojih ulaza pomnožen popratnom težinom, što mreži omogućuje aproksimaciju složenih nelinearnih funkcija. Ovo svojstvo MLP-ova čini ih moćnim univerzalnim aproksimatorima, sposobnima za modeliranje širokog spektra funkcija. Uvođenje višestrukih slojeva povećava sposobnost mreže da formira zamršene granice odlučivanja, čineći MLP-ove prikladnim za rješavanje složenih zadataka kao što su klasifikacija i regresija [26].

2.1.7. Odabrane implementacije algoritama

Prije izlaganja korištenih biblioteka strojnog učenja, opisan je način kako se one integriraju u sustav. U kontekstu projektiranog sustava, metoda predstavlja skriptu koja sadržava funkcije treniranja i predviđanja pomoću jedne uvezene metode strojnog učenja s prikladnim parametrima te funkcije spremanja i učitavanja gotovih modela predikcije. Te funkcije serijalizacije modela se ručno definiraju, jer postoje razlike u formatima datoteka modela ovisno o implementaciji. Pregled kostura skripte metode može se vidjeti u isječku (**Kôd 2.1**). Na početku skripte se uvoze potrebne biblioteke te se popunjava funkcionalnost funkcije treniranja, koja kao unos prima kompletan skup podataka te skup podataka koji sadržava samo značajke i skup podataka koji sadržava samo ciljne značajke. Nije potrebno sve unose iskoristiti te najviše to ovisi o implementaciji algoritma strojnog učenja. Nadalje se definiraju dvije funkcije za klasificiranje skupa podataka ovisno o unesenom modelu. Jedna funkcija vraća pojedine šanse za svaku moguću klasifikaciju, dok druga daje točno definiranu klasu, koja je uglavnom ona s najvećom uspostavljenom šansom. Na kraju se popunjavaju metode spremanja i učitavanja. Glavna zadaća ovdje je definirati pripadajući format datoteke kojom bi se model za dotični algoritam trebao učitavati i spremati.

```r
#load needed libraries

train_model <- function(train_data, train_features, train_target) {
    #to be implemented
    return(model)
}

get_prob_predictions <- function(model, test_features) {
    #to be implemented
    return(prob_predictions)
}

get_predictions <- function(model, test_features) {
    #to be implemented
    return(predictions)
}

save_model <- function(model, file_path) {
    #to be implemented
    return(file_path_with_ext)
}

load_model <- function(file_path_with_ext) {
    #to be implemented
    return(model)
}
```

Kôd 2.1 Kostur funkcije skripte metode koje je potrebno popuniti

Tako se definira standardizirani format koji se koristi za vrijeme istraživanja i u konačnom servisu predviđanja uspjeha. Ujedno se postiže format koji glavna skripta unakrsne provjere modela može učitati i koristiti, a i omogućuje buduće dodavanje novih metoda s različitim parametrima. Sami kostur skripte metode nema značajne razlike ovisno koristi li se kao baza metode za treniranje modela pada ili prolaza ili modela ocjene, osim što skripte za treniranje metode za predviđanje ocjene imaju dodatan parametar za tablicu vrijednosti težine pojedine ocjene, čija se svrha objašnjava kasnije u radu.

Kako bi sustav usporedbe i odabira najboljeg modela funkcionirao, potrebno je definirati točne implementacije algoritama strojnog učenja za pojedinu metodu i njihove parametre. Sažeti prikaz implementacija se može vidjeti u tablici (**Tablica 2.1**) te je nadalje detaljnije definirano. Bitno je napomenuti da metode moraju imati jedinstvena imena u svojoj domeni predviđanja, ali mogu dijeliti ime u različitim domenama. Na primjer, postoje metode `knn` u metodama za model predviđanja pada ili prolaza i za model predviđanja

ocjena, ali ne mogu biti dvije metode imenovane `knn` u domeni predviđanja ocjena. Ako se radi o novoj skripti metode istog algoritma, ali s različitim parametrima, to je potrebno označiti u imenu same metode. Poput razlike između `svm-linear` i `svm-rad`. To je potrebno kako kasnije objašnjen sustav unakrsne usporedbe modela može pravilno izvršavati svoju ulogu.

Tablica 2.1 Pregled korištenih biblioteka i funkcija s pripadnim metodama

Biblioteka	Funkcija	Ime metode/a
e1071	gknn	knn
e1071	svm	svm-linear, svm-rad, svm-poly
u sklopu standardne biblioteke jezika R	glm	log-reg (za pad ili prolaz)
nnet	multinom	log-reg (za ocjene)
randomForest	randomForest	rf-500, rf-1000, rf-1500, rf-2000
xgboost	xgb.train	xgboost-d2, xgboost-d4, xgboost-d6, xgboost-d8, xgboost-d10
keras	fit	mlp, mlp-2-layers, mlp-3-layers, mlp-n-nodes
torch i brulee	brulee_mlp	mlp-torch-brulee, mlp-torch-brulee-2-layers, mlp-torch-brulee-3-layers, mlp-torch-brulee-n-nodes

Biblioteka `e1071` se koristi za implementaciju stroja potpornih vektora pomoću funkcije `svm`. Preko nje su definirana 3 različita SVM-a: linearni, radijalni i polinomni. U ovom radu su te metode imenovane `svm-linear`, `svm-rad` i `svm-poly`. Uz implementaciju SVM-a, biblioteka se koristi i za KNN algoritam uz funkciju `gknn`. Pripadna metoda `knn` definira varijablu k kao korijen duljine skupa podataka nad kojim se provodi treniranje modela.

Logistička regresija se koristi u dvije metode nazvane `log-reg` za svaku domenu predviđanja. Za predviđanje pada i prolaza koristi se funkcija `glm` iz standardne biblioteke u sklopu R jezika, kojoj je parametar obitelji regresije postavljena na binominalnu regresiju. A za predviđanje ocjene, koristi se biblioteka `nnet` te pripadajuća funkcija `multinom`.

Implementacija metoda koje koriste algoritam nasumične šume se koriste funkcijom `randomForest` iz istoimene biblioteke. Te metode su imenovane `rf-` te sufiksom koji predstavlja broj stabala odluke kojima dotična metoda raspolaže.

Isti princip imenovanja se koristi za metode `xgboost-` uz sufiks dubine sastavnih stabla odluke tog algoritma. Te metode koriste biblioteku `xgboost` i treniraju se funkcijom xgb.train.

Biblioteke neuronskih mreža `keras` i `torch` (uz `brulee`) se koriste u istu svrhu treniranja višeslojnih perceptrona. Keras u R-u radi kao API za pokretanje pripadnih biblioteka u Python 3 kôdu, a biblioteka Torch je inspirirana po PyTorch biblioteci za Python 3, ali koristi izvorne procese R programskog jezika. Imenovanje započinje bazom `mlp-` te oznakom radi li se o Torch biblioteci, a ako nema oznaku onda se radi o Keras biblioteci. Svaka metoda koja koristi Torch također koristi biblioteku Brulee, koja pruža jednostavnu sintaksu definiranja modela višeslojnog perceptrona te ta sintaksa prati R način pisanja i time izbjegava Python inspiriranu sintaksu koju koristi sami Torch. Sve metode višeslojnih perceptrona se međusobno razlikuju uglavnom o broju skrivenih slojeva koje koriste te svaki sloj sadrži broj neurona prema formuli (2), inspirirane po [20]. Jedino se razlikuju metode sa sufiksom `-n-nodes` gdje se nalazi samo jedan sloj s brojem neurona jednak broju značajki unesenog skupa podataka.

$$broj\ neurona = \frac{veličina\ skupa\ podataka\ za\ treniranje}{2 * broj\ značajki\ za\ treniranje + 1} \tag{2}$$

2.2. Sastavljanje skupa podataka

Za početak je potrebno prikupiti i formatirati tražene podatke prije provođenja metoda strojnog učenja nad podacima. Taj postupak će biti ugrađen u Edgar te će se kroz automatizirane metode i korisnički unos parametra formatirati konačni skup podataka s određenim atributima koje će se koristiti u kasnijoj analizi i treniranju modela.

2.2.1. SQL upiti

U EDM-u, korištenje SQL upita najvažnije je za izvlačenje vrijednih uvida iz složenih skupova podataka. Nadalje se pojašnjavaju glavni SQL upiti korišteni u istraživanju, ocrtavajući njihovu ulogu u predstavljanju dostupnih podataka nastavnicima i olakšavanje kasnije obrade podataka.

Prvi SQL upit (**Kôd 2.2**) daje informacije o određenom kolegiju kroz sve njegove registrirane akademske godine. Dohvaćaju se mjere o broju provedenih testova te izračunate medijane i srednje vrijednosti provedenih instanci po testu. Ove statističke mjere pružaju perspektivu o distribuciji i intenzitetu testiranja studenata kroz različite akademske godine. Time upit služi kao pomoć pri odabiru odgovarajućih godina kao skup za treniranje modela.

```
SELECT
    academic_year.id,
    academic_year.title,
    COUNT(DISTINCT test.id) AS total_tests,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY test_instance_count) AS
median_test_instance_count,
    ROUND(AVG(test_instance_count)::numeric, 3) AS mean_test_instance_count
FROM
    academic_year
JOIN
    test ON academic_year.id = test.id_academic_year AND test.id_course = 2017 --COURSE VARIABLE
LEFT JOIN (
    SELECT
        test.id AS test_id,
        COUNT(test_instance.id_test) AS test_instance_count
    FROM
        test
    LEFT JOIN
        test_instance ON test.id = test_instance.id_test
    GROUP BY
        test.id
) AS test_instance_counts ON test.id = test_instance_counts.test_id
GROUP BY
    academic_year.id,
    academic_year.title
ORDER BY
    academic_year.id DESC;
```

Kôd 2.2 Primjer SQL upita za dohvat generalnog pregleda aktivnosti kolegija kroz akademske godine

Drugi SQL upit (**Kôd 2.3**) ulazi u detaljniju analizu, vraćajući broj instanci testa za svaki test unutar određenog kolegija i akademske godine. Ova razina detalja omogućuje nastavnicima da izvrše prilagodbe u procesu odabira podataka, osiguravajući da su naknadne analize prilagođene specifičnom kontekstu interesa kolegija. Usredotočivši se na pojedinačne kolegije i akademske godine, ovaj upit pomaže u podešavanju skupa podataka kako bi se ispunili zahtjevi istraživanja.

```
SELECT test.*, COALESCE(COUNT(test_instance.id_test), 0) AS test_instance_count
    FROM test
    LEFT JOIN test_instance ON test.id = test_instance.id_test
    JOIN student ON test_instance.id_student = student.id
    WHERE test.id_course = 2017              -- COURSE VARIABLE
        AND test.id_academic_year = 2021    -- ACADEMIC YEAR VARIABLE
        AND student.id_app_user IS null
    GROUP BY test.id
    ORDER BY test.ts_available_from;
```

Kôd 2.3 Primjer SQL upita za dohvat pregleda svih ispita za određeni kolegij i akademsku godinu

Treći SQL upit (**Kôd 2.4**) igra središnju ulogu u glavnom procesu prikupljanja podataka. U njemu se navode kolegij, akademska godina te testovi značajni za sveukupnu ocjenu studenata. Upit ispisuje rezultate koje je postigao svaki student upisan na kolegij za navedenu godinu, poredane po studentu, a zatim po testu. Ono što je najvažnije, čak i ako učenik nije riješio određeni test, upit uključuje taj test s postignutim bodovima vrijednosti `null`. Ovo uključivanje osigurava ujednačenost u skupu podataka, pri čemu svaki student ima zabilježen jednaki broj testova. Ova ujednačenost je važna za ispravne analize u kasnijim fazama obrade podataka. Zapisivanjem vrijednosti `null` umjesto 0 zadržava se razlika između studenata koje su rješavali test te dobili 0 bodova i onih koji nisu nikad predali svoje rješenje. Primjer nastalog skupa podataka s 4 definirana ispita za neku akademsku godinu i kolegij, ako izdvojimo podatke samo 2 studenta, može se vidjeti u tablici (**Tablica 2.2**).

```
WITH AllPossibleCombinations AS (
  SELECT
    student.id AS id_student,
    test.id AS test_id,
    test.title,
    test.max_score
  FROM
    student
```

```
    LEFT JOIN student_course ON student.id = student_course.id_student
    CROSS JOIN test
  WHERE
    student_course.id_course = 2017             -- COURSE
    AND student_course.id_academic_year = 2021  -- ACADEMIC YEAR
    AND student.id_app_user IS NULL
    AND test.id_course = 2017                   -- COURSE
    AND test.id_academic_year = 2021            -- ACADEMIC YEAR
    AND test.id IN (13465, 13539, 13594, 13645) -- TEST ID ARRAY
)
, AllStudentsWithDummyTestInstances AS (
  SELECT
    apc.id_student,
    apc.test_id,
    apc.title,
    apc.max_score,
    NULL::int AS id_test_instance,
    NULL::int AS score,
    NULL::timestamp AS ts_submitted
  FROM
    AllPossibleCombinations apc
)
, RankedTestInstances AS (
  SELECT
    ast.id_student AS ranked_id_student,
    ast.test_id,
    ast.title,
    ast.max_score,
    test_instance.*,
    ROW_NUMBER() OVER (PARTITION BY ast.id_student, ast.test_id ORDER BY
test_instance.ts_submitted DESC) AS rnk
  FROM
    AllStudentsWithDummyTestInstances ast
    LEFT JOIN test_instance ON ast.id_student = test_instance.id_student
                            AND ast.test_id = test_instance.id_test
)
SELECT ranked_id_student AS id_student, test_id, title, score, max_score
  FROM RankedTestInstances
  WHERE rnk = 1
  ORDER BY ranked_id_student;
```

Kôd 2.4 Primjer SQL upita za dohvat rezultata testova po studentu za određeni kolegij i akademsku godinu

Tablica 2.2 Primjer skupa podataka, dohvaćenog pomoću 3. SQL upita

ID student	ID test	naslov	max bodovi	postignuti bodovi
3452	65433	1. dz	2	1.5
3452	65434	1. labos	10	10
3452	65435	2. labos	10	9
3452	65436	međuispit	25	20
5633	65433	1. dz	2	2
5633	65434	1. labos	10	5
5633	65435	2. labos	10	null
5633	65436	međuispit	25	12.5
…	…	…	…	…

2.2.2. Odabir i izgradnja značajki

Nakon prikupljanja pojedinih rezultata, potrebno je reducirati skup u konačan skup značajki po kojima se klasificiraju konačni ishodi kontinuirane nastave pojedinog studenta. Uz same značajke se nalaze i pomoćne varijable kojima se računaju ciljne značajke ili opisuju pojedini zapisi, poput kojega studenta zapis opisuje ili za koju akademsku godinu se bilježe značajke.

Prije što krene procesiranje podataka, potrebno je definirati grupacije testova. Grupacija predstavlja skup testova koji se u konačnom skupu podataka gledaju kao jedna značajka. Naravno test može postojati sam u svojoj grupaciji i to je poželjno radi granulacije informacija na kojima se model trenira, ali to neće biti uvijek moguće. Strukture pojedinog kolegija se mijenjaju kroz akademske godine te su neke cjeline dodijeljene jednom testu jedne godine, a druge godine podijeljene na dva testa. Grupacije time donose uniformnost zapisa iz različitih akademskih godina u konačnom skupu podataka. Same grupacije se definiraju po nastavnikovom nahođenju, ovisno o nastavnim cjelinama određenog kolegija.

Proces sastavljanja skupa podataka kreće stvaranjem entiteta koji opisuje trenutnog studenta s identifikacijskim brojem, kojoj akademskoj godini pripada te vrijednosti varijable za određivanje konačno postignutih bodova postavljene na nulu. Na zapis se još dodaju značajke i pomoćne varijable tih značajki ovisno o definiranim grupacijama testova. Nakon postavljanja zapisa u početno stanje, program prolazi po redcima rezultata SQL upita (**Kôd 2.4**) te dodaje bodove u skup konačno postignutih bodova i pripadnoj grupaciji kojoj trenutni test pripada. Test nije nužno pridružen određenoj grupaciji. To označava da test prelazi točku predviđanja za trenutni skup podataka, ali se i dalje bilježi kako bi se mogao odrediti konačni

rezultat kojeg je student postigao na kolegiju. Nakon što program dođe do novog studenta, trenutnom zapisu se računaju normalizirani bodovi za svaku grupaciju i konačno postignutih bodova, što je jednostavan izračun prijašnjeg zbroja s pripadajućom najvećom mogućom vrijednosti koju je student mogao postići u grupaciji ili cijelom kolegiju. Zapis se zatim dodaje skupu podataka za treniranje ili predviđanje te se proces ponavlja za novog studenta.

Nakon što su svi studenti obrađeni, računa se težina pojedine grupacije. Taj izračun se izvršava na kraju, jer je potrebno bodovanje svih studenta iz jedne generacije na kolegiju. Uzimaju se sve vrijednosti postignutih bodova jedne grupacije i gleda se distribucija tih rezultata. U suštini, ako rezultati prate normalnu distribuciju, onda bi to dobilo vrijednost 0.5. Ali, ako rezultati teže manjim bodovima to bi označavalo stroži test te bi konačna značajka poprimila vrijednost koja teži prema 1. U suprotnom slučaju, gdje je većina studenata dobila više bodova, značajka bi težila prema 0. Glavni cilj ove značajke je prikazati promjenu težine pojedine grupacije kroz godine te studentima koji su postigli više bodova na težim ispitima bi modeli davali veće ishode za prolaz i više ocjene.

Ciljne značajke se dodaju svakom zapisu ovisno o definiranom pragu bodovanja. Jedna od ciljnih značajki je vrijednost koja određuje je li student prošao kolegij ili pao. Ona poprima vrijednost 1 za prolaz i 0 za pad. Druga ciljna značajka je konačna ocjena, te ona može poprimiti vrijednosti 1, 2, 3, 4 ili 5. Mada su to brojevi, u R jeziku se taj stupac iz skupa podataka gleda kao kategorička varijabla s 5 mogućih vrijednosti. Te dvije značajke se koriste ovisno o tipu modela koji se trenira. Prilikom predviđanja, model neće imati uvid u te značajke, jer ih same predviđa, a one naravno nisu dostupne u skupu podataka koji predstavlja tekuću akademsku godinu.

Primjer izgleda konačnog skupa podataka se može vidjeti na tablici (**Tablica 2.3**), koja je nastala procesiranjem prijašnjeg primjera (**Tablica 2.2**). Bitno je primijetiti kako značajke opisuju jednu grupaciju. Svaka grupacija daje 3 značajke:

- `{grupacija}_score_achieved` – predstavlja postotak postignutih bodova od najviše mogućih za tu grupaciju
- `{grupacija}_of_overall_grade` – predstavlja koliki udio ocjene čini grupacija
- `{grupacija}_diffuculty` – predstavlja težinu postizanja bodova u toj grupaciji

Baza podataka sustava Edgar nema uvide u detaljne podatke studenata te najbolje što se može izvući su informacije o konačnim bodovima. Time se možda smanjuje kvaliteta predviđanja, ali je očuvana privatnost i izbjegavaju se eventualne pristranosti modela.

Tablica 2.3 Pregled formata procesiranih značajki i ciljnih značajki za dva studenta, tablica je invertirana radi preglednosti te izostavlja pomoćne varijable

id_student	3452	5633	…
dz1_score_achieved	0.75	1	…
dz1_of_overall_grade	0.02	0.02	…
dz1_difficulty	0.2	0.2	…
lab1_score_achieved	1	0.5	…
lab1_of_overall_grade	0.1	0.1	…
lab1_difficulty	0.3	0.3	…
lab2_score_achieved	0.9	0	…
lab2_of_overall_grade	0.1	0.1	…
lab2_difficulty	0.5	0.5	…
mi_score_achieved	0.8	0.5	…
mi_of_overall_grade	0.25	0.25	…
mi_difficulty	0.6	0.6	…
pass	1	0	…
grade	4	1	…

Definiranjem glavnih značajki i ciljnih značajki, sustav može slobodno integrirati nove značajke ako se prilikom istraživanja uspostavi potreba za njima. Radi konzistentnosti, te značajke se isto drže normaliziranih vrijednosti u rasponu od 0 do 1.

Nakon sastavljanja skupa podataka za treniranje ili predviđanje, on se sprema u trajnu memoriju kao CSV datoteka. Tako se osigurava prenosivost skupa podataka na udaljene servise izvršavanja programskog kôda i razdvaja proces prikupljanja podataka od procesa za treniranje modela ili njihovo predviđanje.

2.3. Treniranje i usporedbe modela

Prvo je potrebno unijeti podatke o kolegiju za koji se treniraju i uspoređuju modeli, poput identifikacijskog broja kolegija te liste naziva grupacija, koje čine sastavni dio ocjene kolegija. Kao što je definirano u prethodnom poglavlju, iz definicije grupa se izvodi lista značajki koje predstavljaju ostvarene normalizirane bodove, maksimalne bodove i težinu jedne cjeline (grupe) kolegija. Uz to je potrebno definirati točku predikcije nakon koje se treniraju modeli i uskladiti je s listom grupa kolegija, jer se neki elementi ocjenjivanja

kronološki nalaze nakon odabrane točke predikcije. Te elemente je potrebno izostaviti iz liste značajki, jer sve nakon odabrane točke ne može se koristiti kao unos u modele predviđanja uspjeha studenta. U primjeni ti podaci neće biti dostupni pa je potrebno trenirati modele bez tih podataka.

Vodeći se unesenim podacima, pronalazi se CSV datoteka i iz nje se učitavaju tražene značajke u paru s ciljnom značajkom (prolaz ili ocjena) pojedinog studenta. Nakon toga se redoslijed podataka permutira na slučajno odabrani redoslijed i dijeli u k podskupina, ovisno o zadanoj unakrsnoj provjeri. U ovom radu se koristila deseterostruka unakrsna provjera ($k = 10$).

Nakon što su podaci podijeljeni u k podskupova, započinje postupak unakrsne provjere. U svakoj iteraciji petlje, jedna podskupina podataka postaje validacijski skup podataka, dok se preostalih $k - 1$ podskupina koristi za treniranje različitih modela strojnog učenja. Više modela može biti izgrađeno pomoću iste metode, samo sa izmijenjenim parametrima. Tako se i uz različite metode uspoređuje utjecaj parametra pojedine metode na konačnu predikciju. Svaki izgrađeni model testira se na validacijskom skupu trenutne iteracije i bilježi se njegova točnost predviđanja. Primjer rezultata tog procesa može se vidjeti u tablici (**Tablica 2.4**).

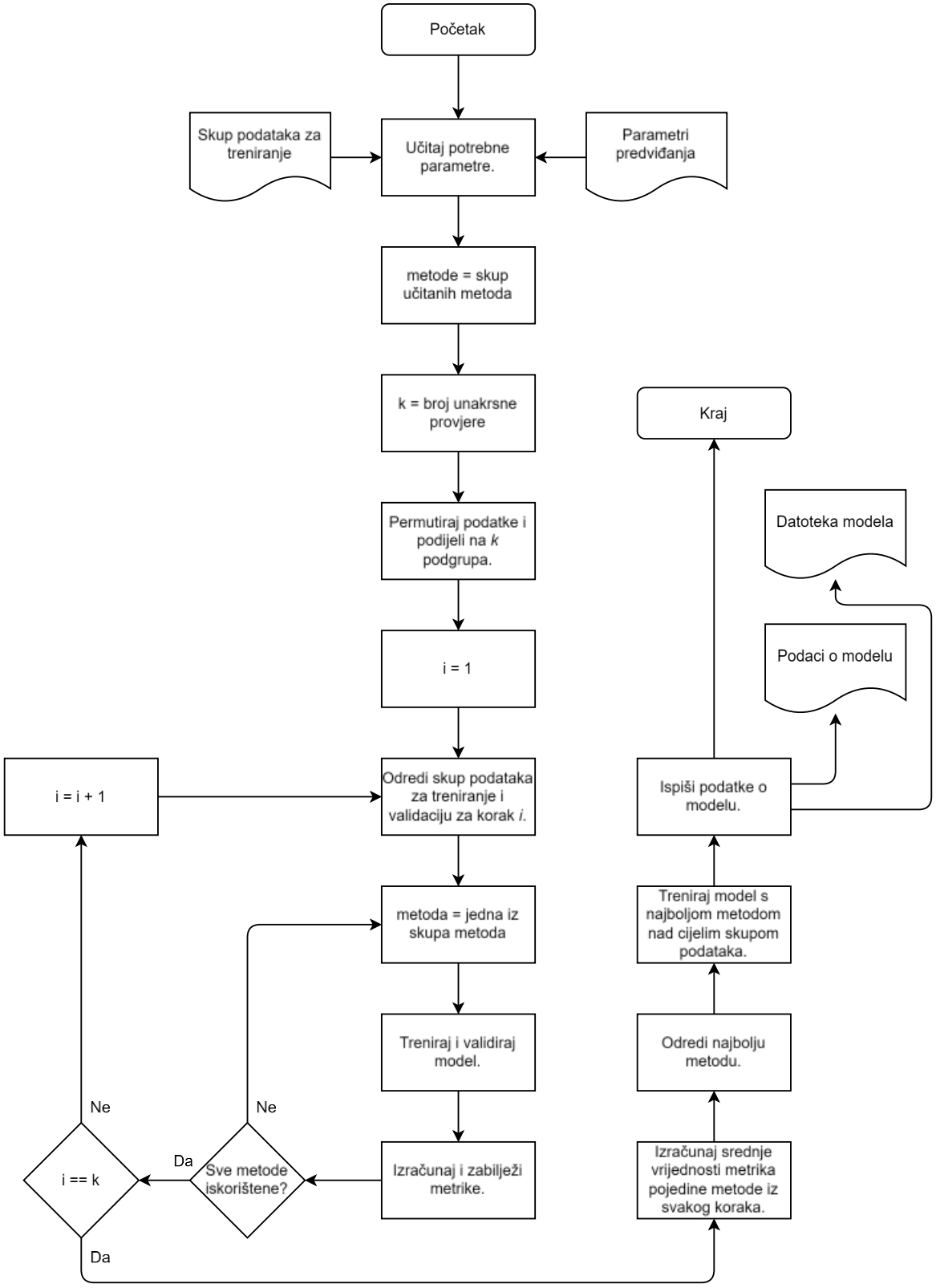
Provedbom unakrsne provjere, vrijednosti točnosti svakog modela se reduciraju u prosječnu vrijednost s popratnom standardnom devijacijom. Time se dobije uvid u najbolji model te koliko je njegova točnost bila stabilna kroz iteracije unakrsne provjere. Primjer usporedbe vrijednosti i odabira najboljeg modela može se vidjeti u tablici (**Tablica 2.4**).

Tablica 2.4 Usporedba točnosti predikcije modela po iteracijama unakrsne provjere te prosječne vrijednosti i standardne devijacije za predviđanje prolaza ili pada studenata za kolegij WEB2 s točkom predikcije neposredno prije završnog ispita. Najbolji model je rf-1000.

	knn	rf-1000	rf-2000	rf-500	svm-linear	svm-rad	xgboost-d10	xgboost-d6
1.	1.0000	1.0000	1.0000	1.0000	0.8333	1.0000	1.0000	1.0000
2.	0.8571	0.8571	0.9286	0.8571	0.8571	0.7857	0.8571	0.8571
3.	0.8333	1.0000	1.0000	1.0000	1.0000	0.9167	0.7500	0.7500
4.	0.7692	0.7692	0.7692	0.7692	0.8462	0.7692	0.9231	0.9231
5.	1.0000	1.0000	1.0000	1.0000	1.0000	0.9167	1.0000	1.0000
6.	0.6667	0.9167	0.8333	0.8333	0.6667	0.7500	0.9167	0.9167
7.	0.7692	0.7692	0.7692	0.7692	0.8462	0.7692	0.8462	0.8462
8.	0.9286	0.9286	0.9286	0.9286	0.9286	0.9286	0.9286	0.9286
9.	0.8462	0.8462	0.8462	0.8462	0.8462	0.8462	0.8462	0.8462
10.	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
μ	0.8670	0.9087	0.9075	0.9004	0.8824	0.8682	0.9068	0.9068
σ	0.1144	0.0939	0.0958	0.0967	0.1039	0.0966	0.0825	0.0825

Metoda s parametrima koja je izgradila najbolji model se koristi za treniranje novog modela koji sada koristi sve raspoložive podatke za treniranje. Tako građeni model se sprema u trajnu memoriju u formatu koji je definiran u samoj skripti metode, što je uglavnom RDS format. To je binarni format koji R koristi za spremanje objekata u trajnu memoriju. Za vrijeme tekuće akademske godine taj se model učitava iz memorije za primjenu predikcije uspješnosti studenata nakon točke predikcije za koju je bio treniran. Nadalje, ulazni podaci predikcije su isto formatirani kao i za vrijeme unakrsne provjere više modela, jedino ciljna značajka nije dostupna za provjeru točnosti predikcije odabranog modela.

Na slici dijagrama toka (**Slika 2.1**) se prikazuje sažeti pseudokôd sustava usporedbe modela i odabira najboljeg. Ujedno, skripte specijalizirane za treniranje modela pada ili prolaza i modela ocjena imaju isti pseudokôd, uz manje značajne razlike.



Slika 2.1 Dijagram toka - Sustav usporedbe modela

Radi bolje provedbe istraživanja, zadnja akademska godina s potpuno zabilježenom i ocijenjenom provedbom kolegija se gleda kao tekuća akademska godina u ovom diplomskom radu. To omogućuje provjeru predikcije izgrađenog modela odabranom metodom strojnog učenja za neki kolegij. Usporedba predviđenih vrijednosti i stvarnih vrijednosti se prikazuju matricom zabune (engl. *confusion matrix*), kao što se vidi na primjeru (**Tablica 2.5**). Njom se dobiva uvid u ponašanje modela, što se najbolje može vidjeti na modelu koji predviđa ocjene studenata. U tablici (**Tablica 2.6**) se prikazuje matrica zabune predviđenih i stvarnih konačnih ocjena, gdje se za predikciju koristio model treniran metodom linearnog SVM-a. Vrijednosti 0 su ostavljene prazne radi preglednosti. Iz matrice se može zaključiti da model ima problema u predviđanju ocjena 3 i 4, a petice nije ni jednom predvidio. Tu se naglašava problem u predviđanju slabije zastupljenih ocjena u skupu podataka za treniranje te je potrebno pronaći rješenje kako bi modeli mogli bolje generalizirati svoja predviđanja.

Tablica 2.5 Matrica zabune predviđenih i pravih ishoda pada ili prolaza studenata za kolegij WEB2 s točkom predikcije neposredno prije završnog ispita, 95% točno predviđenih ishoda

		Previđanje	
		Pad	Prolaz
Stvarni ishod	Pad	20	4
	Prolaz	3	110

Tablica 2.6 Matrica zabune predviđenih i konačnih ocjena za kolegij WEB2 s točkom predikcije neposredno prije završnog ispita, 49% točno predviđenih ishoda

		Predviđanje				
		1	2	3	4	5
Stvarni ishod	1	23	1			
	2	2	23	1		
	3		43	13		
	4			11	8	
	5				13	0

Prilikom testiranja sustava, koristile su se samo dvije točke predikcije za svaki kolegij, ali moguće je definirati i više točaka u primjeni. Radi jednostavnosti, na dalje će prva točka predikcije obilježavati modele trenirane s podacima neposredno nakon međuispita, a druga točka predikcija modele trenirane s podacima neposredno prije završnog ispita.

Utjecaj vremenskog trenutka predviđanja na rezultate samih predviđanja može se vidjeti usporedbom dva modela za kolegij WEB2. Jedan model vrši predikcije nakon prve točke predikcije, a drugi nakon druge točke predikcije. Koristile su se 3 metode strojnog učenja: KNN, RF (1000 stabla) i linearni SVM. Iz priložene tablice (**Tablica 2.7**) primjećuje se porast od otprilike 10% drugoj točki predikcije, što je u skladu s većom količinom informacija kojom raspolaže drugi model. Predviđanje pada ili prolaza isto bilježi rast točnosti, ali uz puno veću početnu vrijednost i dosta slabiji rast, kao što se vidi na tablici (**Tablica 2.8**). Zaključuje se da modeli koji imaju točku predikcije bliže završetku kontinuirane nastave imaju točnija predviđanja.

Tablica 2.7 Usporedba prosječne vrijednost i standardne devijacije točnosti modela između 1. i 2. točke predikcije, nakon unakrsne provjere za predviđanje ocjene za kolegij WEB2

	1. točka predikcije		2. točka predikcije	
model	μ	σ	μ	σ
KNN	0.5959	0.1069	0.7029	0.0471
RF	0.6237	0.0866	0.6818	0.0499
SVM	0.5787	0.0612	0.7773	0.0660

Tablica 2.8 Usporedba prosječne vrijednosti i standardne devijacije točnosti modela između 1. i 2. točke predikcije, nakon unakrsne provjere za predviđanje pada ili prolaza studenata za kolegij WEB2

	1. točka predikcije		2. točka predikcije	
model	μ	σ	μ	σ
KNN	0.8613	0.0759	0.8748	0.0942
RF	0.8683	0.0601	0.8832	0.1240
SVM	0.8915	0.0517	0.8987	0.1099

Sljedeća poglavlja istražuju načine poboljšanja predviđanja modela te kako se dodavanjem novih značajki ili novih koraka pretprocesiranja podataka utječe na rezultate unakrsne provjere za odabir najboljeg modela predikcije za traženi kolegij.

2.4. Utjecaj prijašnjeg zalaganja studenta

Sustav ne mora nužno raspolagati samo podacima trenutnog kolegija za treniranje modela predikcije. Pomoću sustava Edgara te njegove baze podataka, mogu se prikupiti podaci uspjeha na prethodno položenim kolegijima za upisane studente. Ali, ovdje postoji problem raznolikih struktura bodovanja te je potrebno tu raznolikost reducirati u normalizirani rang uspješnosti studenta. Nadalje, ovdje se istražuje koliko je ova metoda pretprocesiranja izvediva i donosi li pozitivne promjene u rezultate predviđanja.

Nastavnici slobodnom procjenom odabiru prikladne kolegije, koje je većina studenata upisana u kolegij za koji se trenira model predikcije polagala te time imaju neki temelj bodova za koji se može izračunati normalizirana značajka. Kao primjer se može dati kolegij Napredne baze podataka, kojemu prikladan kolegij prema kojemu se može ravnati procjena konačne ocjene studenta je kolegij vrlo srodne tematike Baze podataka. Baze podataka je obavezan kolegij za studente računarskog studija čime je osigurano da je većina studenata prilikom upisa kolegija Napredne baze podataka polagala prijašnji kolegij. Ovdje jedino odudaraju studenti koji su položili preddiplomski studij na nekom drugom fakultetu, ali oni čine zanemariv postotak upisanih studenata po kolegiju.

Značajka zalaganja studenta na već položenom kolegiju se računa prikupljanjem svih postignutih bodova po testu, koliko najviše bodova je moguće zaraditi na testu te koliki broj studenata je pristupio testu. Ta 3 elementa testa se množe te zbrajaju s ostalim umnošcima čime se dobije konačan rezultat za jednog studenta, vidljivo u formuli (3) gdje N predstavlja broj testova, a_{si} postignute bodove studenta na testu, m_i najviše moguć broj bodova na testu te t_i broj studenata koji su pristupili testu.

$$zalaganje_s = \sum_{i=1}^{N} (a_{si} * m_i * t_i) \tag{3}$$

Nakon što se taj rezultat odredi za svakog studenta u generaciji koja je polagala kolegij, rezultati se normaliziraju na vrijednosti između 0 i 1. Bitno je naznačiti da se ovo provodi za svakog studenta koji polaže kolegij za koji se pripremaju modeli predviđanja za tekuću akademsku godinu. Pošto se normalizacija računa na temelju generacije kada se odabrani

kolegij polagao, računa se i zalaganje studenata koji potencijalno nisu u skupu podataka za tekuću akademsku godinu. To se može činiti resursno zahtjevno za izračunati, ali studenti po generaciji nemaju veliko odstupanje u godinama polaganja pojedinog kolegija te se uglavnom računaju zalaganja od oko 3 generacije tog kolegija kako bi dobili značajku za tekuću akademsku godinu i kolegij kojeg su spomenuti studenti upisali.

Cilj izračuna je dati veću težinu ispitima koji su imali veći broj bodova i broj studenata koji su pristupili testu. Time se izbjegavaju testovi koji predstavljaju neku vrstu ankete i uglavnom nose 0 bodova i testovi koje je manji dio generacije rješavao, poput ispitnih rokova. Usporedba hipotetskog izračuna zalaganja prije normalizacije između dva studenta može se vidjeti na tablicama (**Tablica 2.9**) i (**Tablica 2.10**). Ako se uzme da je najbolji student na tom kolegiju i akademskoj godini imao zalaganje 160 000 te da je fiksno najgore moguće zalaganje nula, korištenjem *min-max* normalizacije student A bi dobio konačnu vrijednost značajke 0.9867 te student B 0.2391.

Tablica 2.9 Izračun nenormaliziranog zalaganja studenta A na prijašnjem kolegiju

Test	Izračun
1. labos	7 * 8 * 141
međuispit	25 * 40 * 150
anketa	0 * 0 * 137
	= 157 869

Tablica 2.10 Izračun nenormaliziranog zalaganja studenta B na prijašnjem kolegiju

Test	Izračun
1. labos	2 * 8 * 141
međuispit	6 * 40 * 150
anketa	0 * 0 * 137
	= 38 256

Nova značajka nekad pridonosi rastu točnosti, ali nekad kvari točnost. Na primjeru predviđanja prolaza studenta za kolegij Baze podataka, s postavljena dva relevantna kolegija koje su svi studenti polagali godinu prije: Uvod u programiranje i Objektno orijentirano programiranje, dobivaju se dvije dodatne značajke zalaganja za svakog studenta, ali je točnost predviđanja **opala** s 95% na 93%. Predviđanje ocjene je palo sa 62% na 56%. Razlog vjerojatno leži u korelaciji zalaganja prijašnjih kolegija s kolegijem za koji se rade predviđanja. Potrebno je uključiti provjeru korelacije tih dviju varijabli te ako se uspostavi korelacija ispod zadovoljavajuće razine, treba odbaciti značajke zalaganja za taj kolegij, jer daje nepotreban šum u model predviđanja. U ovom radu je zadana zadovoljavajuća korelacija vrijednosti jednake ili veće od 0.3.

Dodatno, prilikom računanja zalaganja pojedinog studenta pojavila se potreba preskočiti pojedine ispite u izračunu, ponajviše ispitne rokove koji unatoč manjem broju izlazaka studenata i dalje donose veliku količinu bodova i stvaraju šum u rezultatima koji ponajviše trebaju predstavljati zalaganje za vrijeme kontinuirane nastave. Kao rješenje se postavlja minimalni postotak studenata koji su morali pristupiti testu te ako je taj postotak za test manji od zadanog, on se neće stavljati u zbroj konačnog rezultata zalaganja. Za vrijeme istraživanja se tražilo minimalno 40% generacije da je pristupilo testu, ali u konačnom sustavu je ova vrijednost podesiva za pojedini kolegij.

S druge strane, uzevši u obzir uspjeh na prethodnim kolegijima kod predviđanja uspjeha na kolegiju Poslovna inteligencija (skraćeno PI), koji bilježi šezdesetak studenata po akademskoj godini, dobiva se porast preciznosti modela predviđanja prolaza s 85% na 90% te model predviđanja ocjena sa 73% na 78%. Kolegiji uzeti u obzir za PI su bili Uvod u programiranje (skraćeno UPRO) i Objektno orijentirano programiranje (skraćeno OOP) radi zadovoljavajuće korelacije. Korelacije između različitih kolegija se mogu vidjeti na tablici (**Tablica 2.11**).

Tablica 2.11 Tablica korelacije postignutih bodova na kolegiju (redovi) i zalaganja na položenom kolegiju (stupci)

	UPRO	OOP	BP
BP	0.485489	0.531785	-
NBP	0.326616	0.33518	0.26673
WEB 2	0.263331	0.221355	0.24208
PI	0.482102	0.447172	0.100963

2.5. Problem neuravnotežene distribucije ocjena

Prilikom treniranja različitih modela i pregleda rezultirajućih matrica zabune dolazi se do spoznaje da modeli za predviđanje konačne ocjene nakon kontinuirane nastave slabo predviđaju manje zastupljene ocjene, koje su uglavnom četvorke i petice. Na kolegiju PI se može najbolje vidjeti to opažanje na matrici zabune (**Tablica 2.12**). Skup podataka za treniranje modela ima 47 studenata te distribuciju ocjena prikazanu na tablici (**Tablica 2.13**). Primjećuje se da samo 20% ocjena sačinjavaju trojke, četvorke i petice. Konačni model za predviđanje ocjena je implicitno postao isti kao i model za predviđanje pada ili prolaza. Kako

modeli ne bi ignorirali više ocjene prilikom slabe zastupljenosti potrebno je priložiti korak pretprocesiranja za balansiranje skupa podataka za treniranje.

Tablica 2.12 Matrica zabune predviđenih i konačnih ocjena za kolegij PI s točkom predikcije neposredno nakon međuispita, 71% točno predviđenih ishoda

		Predviđanje				
		1	2	3	4	5
Stvarni ishod	1	33	7			
	2	4	23			
	3		7	0		
	4		4		0	
	5		1			0

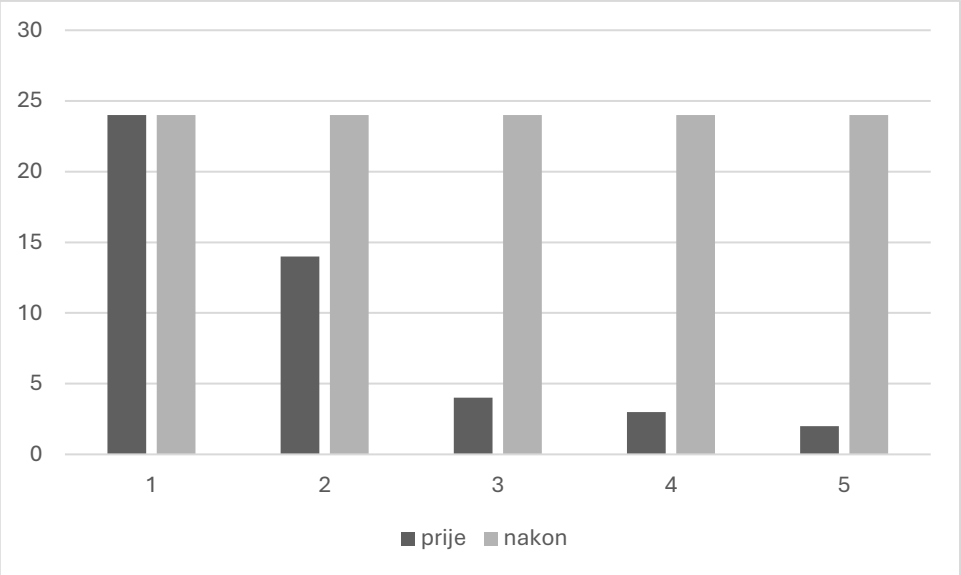
Tablica 2.13 Distribucija konačnih ocjena studenata nad skupom za treniranje modela predikcije kolegija PI

	1	2	3	4	5
#	24	14	4	3	2

2.5.1. Tehnika sintetskog povećanja broja manjinskih klasa

Prva tehnika kojom se pokušalo balansirati podatke za treniranje ocjena je tehnika sintetskog povećanja broja manjinskih klasa (engl. *Synthetic Minority Over-sampling TEchnique*, SMOTE). SMOTE je pristup osmišljen za rješavanje neravnoteže klasa u skupovima podataka stvaranjem sintetičkih primjera manjinske klase umjesto povećanja broja kopiranjem postojećih primjera. Ova tehnika funkcionira odabirom uzorka manjinske klase i uvođenjem sintetičkih primjera duž segmenata linije koji spajaju uzorak i njegove najbliže susjede manjinske klase. Broj najbližih susjeda (obično pet) i željena količina prekomjernog uzorkovanja određuju koliko će se sintetičkih primjera generirati. Primjećuje se da se u korijenu te tehnike nalazi KNN algoritam. Dodavanjem ovih sintetičkih primjera, SMOTE prisiljava regiju odlučivanja za manjinsku klasu da postane veća i općenitija, što pomaže u poboljšanju izvedbe klasifikatora smanjenjem pristranosti prema većinskoj klasi i povećanjem sposobnosti klasifikatora da prepozna instance manjinske klase [6].

Na slici (**Slika 2.2**) se može vidjeti prikaz distribucije ocjena kolegija PI prije i nakon SMOTE tehnike. Povećao se broj uzoraka svih ocjena, kako bi bile izjednačene s ocjenom koju ima najveći broj studenata. Novi uzorci nisu direktne kopije prvobitnih uzoraka, već se razlikuju od njih, a i dalje bi trebali sadržavati generalne čimbenike kako bi se klasificirali svojoj ocjeni.



Slika 2.2 Stupčasti dijagram distribucije studenata po ocjenama za skup podataka za treniranje za kolegij PI, prije i nakon SMOTE-a

Modeli trenirani na podacima koji su prošli SMOTE pretprocesiranje su predviđali manje zastupljene ocjene, ali se primjećuje pad generalizacije modela što je pridonijelo padu točnosti svih modela za 2% - 5%. Radi nezadovoljavajućih rezultata korištenja SMOTE-a, potrebno je pristupiti problemu neuravnoteženih klasa u podacima na drugačiji način.

2.5.2. Težina pojedinih klasa

Drugo rješenje ne manipulira veličinom skupa podataka, već određuje zastupljenost klasa u skupu podataka te pridjeljuje svakoj klasi određenu težinu ovisno o njezinoj zastupljenosti. Time predikcijski modeli neće ignorirati manje zastupljene klase u predviđanju i modeli neće biti trenirani na sintetiziranim primjerima. Razlika u težinama će pomoći klasifikatoru u fazi treniranja. Cilj je penalizirati kriva predviđanja nad manje zastupljenim klasama postavljajući veće težine nad njima, a manje težine nad više

zastupljenim [30]. Bitno je naglasiti da se time ne povećava informacija koju dobivamo iz skupa podataka, nego se upravlja pristranošću klasifikatora na određene klase [32].

U trenutnom istraživanju, ocjene koje studenti mogu konačno dobiti su klase te je cilj davati veće težine manje zastupljenim ocjenama. Točne težine se razlikuju po kolegiju i akademskoj godini. Po principu uravnotežavanja skupa podataka potrebno je više zastupljenim ocjenama davati manje težine, ali u ovom istraživanju to se neće primjenjivati. Smatra se da je i dalje potrebno održavati značajnu pažnju na granici pada i prolaza, a tome je srodna granica ocjena jedinice i dvojke. Kako bi se održala trenutna točnost predviđanja jedinice i dvojke, a u isto vrijeme povećala pristranost na više ocjene, nakon određivanja distribucije ocjena u skupu podataka te izračunom prikladnih težina, dodaje se uvjet da težina ne može biti vrijednost manja od 1. Konačna implementacija određivanja težina u sustavu se može vidjeti u isječku (**Kôd 2.5**).

```
calculate_class_weight_for_balance <- function(n_samples, n_classes, n_class_samples) {
    return(n_samples / (n_classes * n_class_samples))
}

n_samples <- nrow(data)
grade_distribution_table <- table(data$grade)

n_classes <- 5
n_1 <- grade_distribution_table[[1]]
n_2 <- grade_distribution_table[[2]]
n_3 <- grade_distribution_table[[3]]
n_4 <- grade_distribution_table[[4]]
n_5 <- grade_distribution_table[[5]]

class_weights <- c(
    "1" = max(1, calculate_class_weight_for_balance(n_samples, n_classes, n_1)),
    "2" = max(1, calculate_class_weight_for_balance(n_samples, n_classes, n_2)),
    "3" = max(1, calculate_class_weight_for_balance(n_samples, n_classes, n_3)),
    "4" = max(1, calculate_class_weight_for_balance(n_samples, n_classes, n_4)),
    "5" = max(1, calculate_class_weight_for_balance(n_samples, n_classes, n_5))
)
```

Kôd 2.5 Isječak R kôda za definiranje težine ocjena prije treniranja modela

Usporedbom novih (**Tablica 2.14**) i starih (**Tablica 2.12**) rezultata, može se primijetiti pad sveukupne točnosti modela za 1%, ali se održala stabilnost predviđanja između jedinice i

dvojke. Ujedno, model sada pokušava predviđati manje zastupljene ocjene, koje je prije svrstavao pod dvojke. Sličan trend se primijetio i na drugim kolegijima, ali nije bio toliko izražen kao na kolegiju PI. Unatoč blagom padu točnosti, ovaj proces balansiranja se koristi u konačnom sustavu, jer se paralelno koristi model za predviđanje pada i prolaza nad istim podacima te će on korigirati konačne ocjene na granici jedinice i dvojke, a model predviđanja ocjena će davati veći broj viših ocjena koje model pada i prolaza ne može odrediti sam.

Tablica 2.14 Matrica zabune predviđenih i konačnih ocjena za kolegij PI s pridruženim težinama svakoj ocjeni s točkom predikcije neposredno nakon međuispita, 70% točno predviđenih ishoda

		Predviđanje				
		1	2	3	4	5
Stvarni ishod	1	31	9			
	2	4	21		1	1
	3		4	1	1	1
	4		1	2	1	
	5					1

2.6. Metrike predvidivosti modela

Do sada kao glavna metrika usporedbe i testiranja modela se koristila točnost (engl. *accuracy*), ali ona ne mora nužno biti najbolji odabir za ovu domenu dubinske analize podataka. U nastavku se prikazuju 3 nove metrike koje također daju uvid u sposobnosti predikcije konačnog modela.

Točnost predikcije je jednostavan izračun dijeljenja točno predviđenih ishoda sa svim ishodima (1). Problem usporedbe modela s tom metrikom može se dati primjerom gdje podaci za treniranje sadrže 80% studenata kojima je zapisan prolaz na kolegiju. Moguće je da tim podacima krajnje odabran model uvijek predviđa prolaze, ali bi i dalje imao 80% točnost predikcije. Takvi banalni modeli nisu poželjni u konačnom sustavu te je potrebno odrediti bolju metriku usporedbe modela, pogotovo binarnog klasifikatora pada i prolaza.

Cohenov Kappa rezultat ili *kappa* koeficijent se razlikuje od točnosti upravo u tome što banalne modele predviđanja ocjenjuje nulom. *Kappa* koeficijent uspoređuje promatranu točnost sa očekivanom (4). Model s točnošću 80% se čini dosta manje impresivan ako je očekivana točnost 75% za razliku od modela s istom promatranom točnošću, ali s

očekivanom od 50% [25][34]. Ta metrika daje bolje međusobne usporedbe rezultata modela, jer svi modeli imaju istu očekivanu točnost zbog treniranja nad istim skupom podataka. U konačnom sustavu *kappa* se koristi kao glavna metrika usporedbe modela pada i prolaza radi odabira najboljih modela predikcije ne na bazi same točnosti, već i oprečnosti za pogađanje ishoda po samoj distribuciji podataka po kojima je isti model treniran.

$$kappa = \frac{promatrana\ točnost - očekivana\ točnost}{1 - očekivana\ točnost} \tag{4}$$

Balansirana točnost gleda točnost predviđanja pojedine klase. U svrhu usporedbe modela pada i prolaza neće služiti neku veću svrhu, ali ju je potrebno razmotriti u modelima predviđanja ocjena, jer balansirana točnost je bolja metrika usporedbe modela naspram same točnosti za nebalansirane podatke [24][33]. Formula balansirane točnosti u višeklasnim klasifikatorima (6) je srednja vrijednost vrijednosti *recall* (5) pojedine klase. Trenutna domena sadržava 5 klasa, tj. ocjena, te se usporedbom balansirane točnosti treniranih modela dobiva uvid u model koji najpreciznije predviđa konačnu ocjenu studenta na kolegiju. Konačni sustav gleda balansiranu točnost kao glavnu metriku za procjenu sposobnosti modela predviđanja ocjena te samog odabira najboljeg modela za kolegij.

$$recall_c = \frac{broj\ točnih\ predikcija\ klase\ c}{broj\ svih\ predikcija\ klase\ c} \tag{5}$$

$$balansirana\ točnost = \frac{\sum_{c=1}^{n} recall_c}{n} \tag{6}$$

Zadnja metrika je F_1 rezultat, harmonijska sredina vrijednosti *precision* i *recall* za binarne klasifikatore. Za višeklasne klasifikatore se u ovom radu koristi makro F_1 rezultat, što je srednja vrijednost F_1 rezultata pojedine klase. F_1 rezultat daje jedinstven uvid u vrijednosti preciznosti (engl. *precision*) i opoziva (engl. *recall*) konačnog modela. Time se traži najbolja ravnoteža između te dvije mjere. Preciznost predstavlja koliko relevantno predviđeni zapisi su zapravo relevantni, a opoziv koliko od svih relevantnih zapisa je predviđeno kao

relevantno. Usporedbom korištenja prijašnjih metrika za treniranje i konačnu evaulaciju modela, uočeno je da se korištenjem prijašnjih metrika i F_1 rezultata biraju podjednaki modeli, ali pošto su *kappa* i balansirana točnost više usklađene zahtjevima konačne predikcije, F_1 rezultat ostaje kao popratna metrika uz točnost.

Proces usporedbe modela nije potrebno znatno izmijeniti kako bi se umjesto točnosti koristile *kappa* i balansirana točnost. Umjesto samog spremanja točnosti za svaki korak unakrsne provjere, spremaju se i ostale 3 metrike. Zatim, nakon svakog koraka unakrsne provjere izračunavaju se srednje vrijednosti svake metrike te se model s najvećom glavnom metrikom sprema za kasnija predviđanja tijekom tekuće akademske godine. Izračuni metrika u R programskom jeziku provode se pomoću `confusionMatrix` metode iz `caret` biblioteke, vidljivo u isječcima za izračun metrika modela predviđanja prolaza i pada (**Kôd 2.6**) i modela predviđanja ocjena (**Kôd 2.7**).

```
calculate_pass_model_metrics <- function(predictions, test_target) {
    confusion_matrix <- caret::confusionMatrix(
        data = predictions,
        reference = test_target
    )

    accuracy <- confusion_matrix[["overall"]][["Accuracy"]]
    kappa <- confusion_matrix[["overall"]][["Kappa"]]
    balanced_accuracy <- confusion_matrix[["byClass"]][["Balanced Accuracy"]]
    f1_score <- confusion_matrix[["byClass"]][["F1"]]

    return(list(
        accuracy = accuracy,
        kappa = kappa,
        balanced_accuracy = balanced_accuracy,
        f1_score = f1_score
    ))
}
```

Kôd 2.6 Funkcija izračuna metrika za model predviđanja prolaza i pada

```
calculate_grade_model_metrics <- function(predictions, test_target) {
    confusion_matrix <- caret::confusionMatrix(
        data = predictions,
        reference = test_target
    )
```

```
    accuracy <- confusion_matrix[["overall"]][["Accuracy"]]

    kappa <- confusion_matrix[["overall"]][["Kappa"]]

    precision_vector <- confusion_matrix[["byClass"]][, "Pos Pred Value"]
    precision_vector[is.na(precision_vector)] <- 0
    precision <- mean(precision_vector)

    recall_vector <- confusion_matrix[["byClass"]][, "Sensitivity"]
    recall_vector[is.na(recall_vector)] <- 0
    balanced_accuracy <- mean(recall_vector)

    f1_scores <- 2 * (precision_vector * recall_vector) / (precision_vector + recall_vector)
    f1_scores[is.na(f1_scores)] <- 0

    macro_f1_score <- mean(f1_scores)

    return(list(
        accuracy = accuracy,
        kappa = kappa,
        balanced_accuracy = balanced_accuracy,
        f1_score = macro_f1_score
    ))
}
```

Kôd 2.7 Funkcija izračuna metrika za model predviđanja ocjena

2.7. Rezultati sustava usporedbe

Pomoću zaključaka iz prijašnjih poglavlja, prilažu se konačni rezultati sustava usporedbe i odabira najboljeg modela. Detalji veličine zapisa za treniranje i zapisa iz tekuće akademske godine po korištenom kolegiju su vidljivi u tablici (**Tablica 2.15**). Nadalje, u usporedbi su se koristile sve do sad implementirane metode prikazane u tablici (**Tablica 2.1**). Grupacije testova su bile što više specifičnije moguće za svaki kolegij te velika većina grupacija sadrži samo jedan test, uz poneke iznimke poput A i B grupe međuispita u kolegiju UPRO, podjela jedne cjeline na dva testa u kolegiju NBP i tako dalje. Isto se naglašava da se u ovim usporedbama nisu koristile značajke relevantnih kolegija te da bi se ponavljanjem mjerenja moglo dobiti drugačije rezultate radi permutacije podataka po slučajnoj varijabli ili same stohastičke naravi nekih algoritama strojnog učenja, poput neuronskih mreža. I naravno, za najbolji model pada ili prolaza se odabire onaj s najboljom vrijednosti *kappa*, a

model ocjene s najboljom uravnoteženom točnošću. Cilj toga je dobiti modele koji nemaju drastičnu razliku između predviđene i stvarne točnosti modela predikcije.

Tablica 2.15 Korišteni kolegiji u konačnoj inačici sustava usporedbe i njihove veličine pripadnih skupova podataka

kolegij	razina studija	broj zapisa u skupu za treniranje	broj zapisa u tekućoj akademskoj godini
BP	preddiplomski	535	567
NBP	diplomski	293	176
UPRO	preddiplomski	1523	765
WEB2	diplomski	128	137
PI	diplomski	47	79

Zabilježeni rezultati konačnog sustava se mogu vidjeti na tablicama najboljih metoda za svaki kolegij u domeni predviđanja prolaza (**Tablica 2.16**) i domeni predviđanja ocjene (**Tablica 2.17**). Primjećuje se raznolik odabir najboljih metoda za treniranje konačnih modela, čime se zaključuje da pojedini algoritmi strojnog učenja i njihove implementacije mogu bolje prepoznati značajke za predviđanje prolaza ili ocjena za različite kolegije te niti jedna metoda u potpunosti ne nadvladava ostale u tom aspektu. Mada MLP zastupa 11 od 20 odabranih modela, bitno je napomenuti da su ostali algoritmi u većini slučajeva izostavljeni, jer su bili slabiji za samo manje od 1% od konačno odabranog modela.

Tablica 2.16 Zabilježeni rezultati odabira najboljih modela predviđanja prolaza te testiranje tih modela na podacima iz tekuće akademske godine

	1. točka predikcije			2. točka predikcije		
kolegij	najbolja metoda	predviđena točnost odabrane najbolje metode	točnost predviđanja u tekućoj ak. godini	najbolja metoda	predviđena točnost odabrane najbolje metode	točnost predviđanja u tekućoj ak. godini
BP	xgboost-d6	0.9195	0.9400	rf-2000	0.9458	0.9541
NBP	mlp-2-layers	0.8940	0.8523	mlp-3-layers	0.9111	0.8977
UPRO	mlp	0.8852	0.7987	mlp-torch-brulee	0.9081	0.8980
WEB2	rf-500	0.8895	0.9343	mlp-torch-brulee	0.9615	0.9562
PI	svm-linear	0.9033	0.8608	rf-1000	0.9100	0.8987

Tablica 2.17 Zabilježeni rezultati odabira najboljih modela predviđanja ocjena te testiranje tih modela na podacima iz tekuće akademske godine

	1. točka predikcije			2. točka predikcije		
kolegij	najbolja metoda	predviđena točnost odabrane najbolje metode	točnost predviđanja u tekućoj ak. godini	najbolja metoda	predviđena točnost odabrane najbolje metode	točnost predviđanja u tekućoj ak. godini
BP	mlp-torch-brulee -n-nodes	0.7381	0.6667	mlp-torch-brulee	0.7718	0.6914
NBP	mlp-torch-brulee -n-nodes	0.5970	0.5909	mlp-torch-brulee -n-nodes	0.7365	0.7330
UPRO	mlp-torch-brulee -3-layers	0.5785	0.5242	mlp-torch-brulee -2-layers	0.6520	0.6026
WEB2	xgboost-d4	0.5405	0.3504	log-reg	0.7970	0.6642
PI	xgboost-d10	0.6783	0.7468	svm-linear	0.7167	0.7342

3. Arhitektura sustava predviđanja uspjeha

Glavni dio sustava se nalazi u samostalnom servisu predviđanja uspjeha studenata te nije potrebno konstantno održavati uslugu pokrenutom. Glavna zadaća usluge predviđanja uspjeha je prikupljati i procesirati potrebne podatke za izvedbu treniranja ili predviđanja, ovisno o korištenom kontekstu. Ujedno sadrži skripte programskog kôda za izvršavanje treniranja modela i predviđanja tim modelima. Te datoteke i podaci se unutar usluge pakiraju i formatiraju u format ovisno gdje će se konačno izvršavati zadani kôd. Ovisno o početnim postavkama servisa, kôd se može izvršavati u Judge0 servisu ili na računalu koje poslužuje sami servis. Na kraju sve bitne izmjene i predviđanja sprema u bazu podataka.

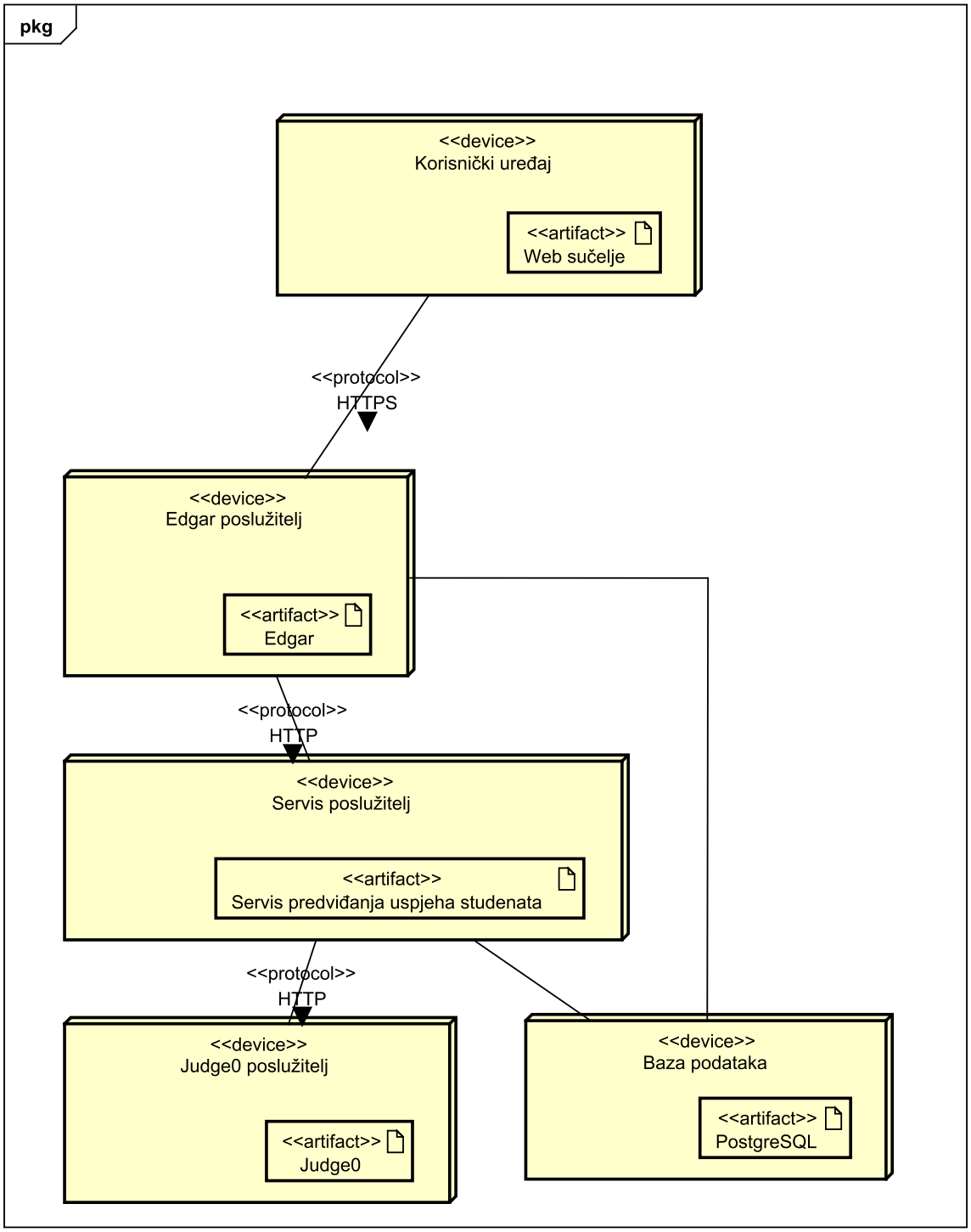
Sama baza podataka je ista koju koristi sustav Edgar te je definirana nova shema `student_forecast` s 4 nove tablice. Korištenje nove baze u svrhu predviđanja uspjeha nije praktično, jer su novim tablicama potrebni strani ključevi koji povezuju postavke i predviđanja s pripadnom akademskom godinom, kolegijem i studentom, što je sve definirano u postojećoj bazi. Tako se stvaranjem nove sheme izoliraju podaci od tablica iz javne sheme koje se češće koriste, a usluga i dalje barata potrebnim kontekstom za prikupljanje podataka i spremanje predviđanja u povezanu strukturu.

Edgar raspolaže novim API rutama za upravljanje novim servisom iz razloga što korisnicima nije namijenjeno baratati servisom direktno, jer sam servis nema implementiranu autorizaciju korisnika. On se oslanja implicitno na Edgarov sustav autentifikacije i autorizacije koje pojedina korisnička sjednica sadržava. Naravno, ne mora se preko svake API rute komunicirati sa servisom, već se preko nekih može pristupiti podacima iz baze podataka direktno. Zahtjevi koji se nalaze na tim rutama ne zahtijevaju procesiranje podataka ili izvršavanje kôda na samom servisu i potpuno su funkcionalne i kada servis ne radi, kao na primjer dohvat već izračunatih predviđanja koja su spremljena u bazi.

Jedna Edgarova API ruta poslužuje autoriziranog nastavnika korisničkim sučeljem, kojim može putem grafičkog sučelja konfigurirati predviđanja i pregledavati studentov predviđen uspjeh. Kolegij i akademska godina sučelja su određena varijablama korisničke sjednice koje se mogu birati na Edgarovoj početnoj stranici. Sučelje pruža način pregleda postojećih metoda i stvaranje novih, vođeno sučelje za postavljanje podataka za treniranje i definiranje točaka predikcije, planiranje vremena izvedbe treniranja modela i predviđanja tim modelima, pregled metrika točnosti tih modela uz generalni grafički pregled predviđene

uspješnosti generacije nakon odabrane točke predikcije te detaljan pregled pojedinačnog uspjeha s popratnim grafičkim elementima za svakog studenta. Druga ruta za pregled uspješnosti je izgrađena za studente, gdje mogu vidjeti ažurna predviđanja svog uspjeha. Ta rute ne poslužuje jednostraničnu web aplikaciju kao nastavničko sučelje, već se drži HTML stranica prilagođenih pojedinom korisniku na poslužitelju, sistemom koji Edgar koristi.

Fizički razmještaj svih tih sustava i njihove poveznice se mogu vidjeti na slici dijagrama razmještaja sustava predviđanja uspjeha studenata (**Slika 3.1**).



Slika 3.1 Dijagram razmještaja - Sustav predviđanja uspjeha studenata

3.1. Proširenje baze podataka

Nova shema Predviđanje uspjeha studenata sastavljena je od 4 nove tablice koje određuju podatke za treniranje modela i raspored točaka predikcije u kojima se pohranjuju postavljena predviđanja.

Tablica (**Tablica 3.1**) opisuje spremljene implementacije metoda strojnog učenja s popratnom datotekom programskog kôda, pisanog u R programskom jeziku. Metode se mogu gledati kao globalne vrijednosti u sustavu predviđanja, jer se povezuju na modele iz različitih akademskih godina i kolegija, što otežava samo brisanje tih metoda iz baze, jer veliki broj modela ovise o njima.

Tablica 3.1 Tablica registriranih metoda

Ime	Tip	Opis
id	INTEGER	Jedinstveni ID metode.
prediction_type	VARCHAR(10)	Vrsta predviđanja (prolaz ili ocjena).
name_method	VARCHAR(255)	Jedinstveno ime metode unutar vrste predviđanja.
description_method	TEXT	Opis metode.
file	BYTEA	Binarna datoteka kôda metode.

Tablica konfiguracije predviđanja (**Tablica 3.2**) se može smatrati korijenom sustava predviđanja za definiranu akademsku godinu i kolegij, jer su sve nadalje opisane tablice eksplicitno ili implicitno povezane na nju. U nju se također zapisuju bodovni pragovi ocjena kolegija za trenutnu akademsku godinu, radi različitih pragova između kolegija, a i akademskih godina. Na sličan način, zabilješkom maksimalnog broja bodova na kolegiju daje podesivost koju sustav zahtjeva prilikom mogućeg odstupanja vrijednosti od standardnih 100 bodova. Glavna stavka tablice je JSON konfiguracija koja definira grupiranje testova i raspored testova kroz odabrane akademske godine te broj točaka predikcije sa značajkama koje se koriste u klasifikaciji uspjeha. Ta JSON struktura ne sadrži testove koji su se smatrali nepotrebnima za treniranje modela predviđanja. Primjer jedne takve strukture se može vidjeti na primjeru kolegija UPRO u isječku (**Kôd 3.1**). Zadnja stavka tablice je statistika skupa podataka za treniranje, generiranog prema postavkama prošle JSON strukture. Ona zapisuje ukupan broj studenata te raspored studenata po padu ili prolazu te po ocjenama za svaku korištenu akademsku godinu. Glavna zadaća statistike je

dati nastavniku povijesnu raspodjelu uspjeha studenata na kolegiju kako bi mogao donositi pravilne zaključke o validnosti predviđenih uspjeha za tekuću akademsku godinu.

Tablica 3.2 Tablica konfiguracija predviđanja

Ime	Tip	Opis
id	INTEGER	Jedinstveni ID konfiguracije predviđanja.
id_course	INTEGER	Jedinstveni ID koji povezuje određeni kolegij.
id_academic_year	INTEGER	Jedinstveni ID koji povezuje određenu akademsku godinu.
grade_threshold_array	NUMERIC(6, 3)[]	Niz numeričkih vrijednosti koje određuju bodovne pragove.
max_score_cutoff	NUMERIC(6, 3)	Vrijednost maksimalnog broja bodova (najčešće jednak 100).
configuration_json	JSONB	Konfiguracija spremljena u JSON formatu. Definira koje akademske godine i testove koristiti u treniranju te broj točaka predikcije.
training_dataset_stats_json	JSONB	Statistika korištenog skupa podataka za treniranje modela u JSON formatu.


```
{
    "yearId": 2023,
    "courseId": 2004,
    "definedGroups": [
        "lab1abc", lab1c", "lab2abc", "lab2c",
        "lab3abc", "lab3c", "lab4abc", "lab4c", "mi",
        "lab5abc", "lab5c", "lab6abc", "lab6c", "lab7abc",
        "lab7c","lab8abc", "lab8c", "zi"
    ],
    "selectedTests": [
        {
            "id": 14034,
            "group": "mi",
            "is_in": true,
            "title": "Međuispit 2022/23 - prvi termin",
            "id_course": 2004,
            "max_score": 35,
            "id_academic_year": 2022,
            "test_instance_count": 392,
            "ignore_students_with_this_test": false
        },
        {
            "id": 14035,
            "group": "mi",
            "is_in": true,
            "title": "Međuispit 2022/23 - drugi termin",
```

```
            "id_course": 2004,
            "max_score": 35,
            "id_academic_year": 2022,
            "test_instance_count": 385,
            "ignore_students_with_this_test": false
        },
        {
            "id": 13966,
            "group": "lab2c",
            "is_in": true,
            "title": "2. laboratorijska vježba - C",
            "id_course": 2004,
            "max_score": 1.2,
            "id_academic_year": 2022,
            "test_instance_count": 758,
            "ignore_students_with_this_test": false
        },
        {
            "id": 13963,
            "group": "lab1abc",
            "is_in": true,
            "title": "1. laboratorijska vježba - ABC",
            "id_course": 2004,
            "max_score": 1,
            "id_academic_year": 2022,
            "test_instance_count": 760,
            "ignore_students_with_this_test": false
        },

        ...

        {
            "id": 13456,
            "group": "lab8abc",
            "is_in": true,
            "title": "8. laboratorijska vježba - ABC",
            "id_course": 2004,
            "max_score": 1.2,
            "id_academic_year": 2021,
            "test_instance_count": 495,
            "ignore_students_with_this_test": false
        },
        {
            "id": 13438,
            "group": "lab8c",
            "is_in": true,
            "title": "8. laboratorijska vježba - C",
            "id_course": 2004,
            "max_score": 1.6,
```

```
            "id_academic_year": 2021,
            "test_instance_count": 492,
            "ignore_students_with_this_test": false
        },
        {
            "id": 13630,
            "group": "zi",
            "is_in": true,
            "title": "Završni ispit 2021/22 - prvi termin",
            "id_course": 2004,
            "max_score": 45,
            "id_academic_year": 2021,
            "test_instance_count": 342,
            "ignore_students_with_this_test": false
        },
        {
            "id": 13631,
            "group": "zi",
            "is_in": true,
            "title": "Završni ispit 2021/22 - drugi termin",
            "id_course": 2004,
            "max_score": 45,
            "id_academic_year": 2021,
            "test_instance_count": 284,
            "ignore_students_with_this_test": false
        }
    ],
    "selectedYears": [2022, 2021],
    "maxScoreCutoff": 100,
    "gradeThresholds": [50, 62.5, 75, 87.5],
    "predictionPoints": {
        "numberOfPredictionPoints": 2,
        "allOrderedTestIdsInAcademicYearSplitByPredictionPoints": {
            "2021": [
                [
                    13421, 13431, 13432, 13450, 13451,
                    13433, 13434, 13452, 13525, 13526
                ],
                [
                    13453, 13435, 13454, 13436, 13455,
                    13437, 13456, 13438
                ],
                [
                    13630, 13631
                ]
            ],
            "2022": [
                [
                    13964, 13963, 13965, 13966, 13968,
```

```
                    13967, 13970, 13969, 14035, 14034
                ],
                [
                    13972, 13971, 13978, 13977, 13976,
                    13975, 13974, 13973
                ],
                [
                    14087, 14088
                ]
            ]
        }
    },
    "selectedGradeMethods": [18, 19, 20, 21, 24, 25, 26],
    "selectedPassFailMethods": [1, 2, 3, 4, 5, 6, 7, 10, 11, 12],
    "selectedPrimaryGroupFeaturesSuffixes": [
        "_score_achieved",
        "_of_overall_grade",
        "_difficulty"
    ],
    "selectedCourseCorrelationAdditionalFeaturesCourses": []
}
```


Kôd 3.1 Primjer JSON konfiguracije predviđanja za kolegij UPRO 2023./2024. gdje se određuju podaci iz prijašnje dvije akademske godine za treniranje modela predikcije uspjeha

Sljedeća tablica (**Tablica 3.3**) je usko povezana na prethodnu, jer broj točaka predikcije eksplicitno ovisi o konfiguraciji predikcije tekuće akademske godine. Sami zapis u tablici identificira točku predikcije po rednom broju i kojoj konfiguraciji pripada. Ostale varijable zapisa započinju praznim vrijednostima te se tijekom treniranja i donošenja predviđanja popunjavaju. Zakazivanje treniranja i predviđanja te popratne oznake njihove izvedbe služe sustavu kako bi i nakon ponovnog pokretanja ili ispadanja mogao i dalje izvršiti zadane zadaće izvršavanja kôda. Nakon izvršenog treniranja, zapis točke predikcije se povezuje s najboljom metodom koja je doprinijela najboljem modelu predviđanja prolaza za trenutnu točku predikcije i isto za najbolju metodu predviđanja ocjena. Uz njih su varijable koje bilježe ime tih modela kako su spremljene u datotečnom sustavu servisa predviđanja uspjeha i njihove metrike. Uz metrike učinka modela u izolaciji, bilježe se i metrike združenog predviđanja oba modela. Za kraj, kako bi zabilježeni modeli mogli obnašati predviđanje potrebno je povezati testove iz tekuće akademske godine grupaciji iz početne konfiguracije kako bi sustav mogao sastaviti skup podataka s istim značajkama kao što je imao i skup za

treniranje. To je JSON struktura koja sadrži samo informacije o testovima koji se nalaze prije same točke predikcije, što se može vidjeti u priloženom isječku (**Kôd 3.2**).

Tablica 3.3 Tablica točaka predikcije

Ime	Tip	Opis
id	INTEGER	Jedinstveni ID točke predikcije.
id_forecast_configuration	INTEGER	Jedinstveni ID koji povezuje određenu konfiguraciju predviđanja.
ordinal_prediction_point	INTEGER	Redni broj točke predikcije, definirane u konfiguraciji predviđanja.
training_schedule	TIMESTAMP	Zakazani termin treniranja modela.
training_executed	BOOL	Oznaka odrađenog treniranja modela.
training_output_log_filename_with_ext	VARCHAR(64)	Ime spremljene datoteke programskog ispisa tokom treniranja modela.
prediction_schedule	TIMESTAMP	Zakazani termin predviđanja.
prediction_executed	BOOL	Oznaka odrađenog predviđanja.
prediction_output_log_filename_with_ext	VARCHAR(64)	Ime spremljene datoteke programskog ispisa tokom predviđanja.
id_pass_prediction_method	INTEGER	Jedinstveni ID koji povezuje određenu metodu vrste predviđanja prolaza.
pass_model_filename_with_ext	VARCHAR(64)	Ime spremljene datoteke modela prolaza.
pass_model_accuracy	FLOAT	Zabilježena točnost modela prolaza.
pass_model_kappa	FLOAT	Zabilježena vrijednost *kappa* modela prolaza.
pass_model_balanced_accuracy	FLOAT	Zabilježena uravnotežena točnost modela prolaza.
pass_model_f1_score	FLOAT	Zabilježena F1 vrijednost modela prolaza.
id_grade_prediction_method	INTEGER	Jedinstveni ID koji povezuje određenu metodu vrste predviđanja ocjene.
grade_model_filename_with_ext	VARCHAR(64)	Ime spremljene datoteke modela ocjene.
grade_model_accuracy	FLOAT	Zabilježena točnost modela ocjene.
grade_model_kappa	FLOAT	Zabilježena vrijednost *kappa* modela ocjene.
grade_model_balanced_accuracy	FLOAT	Zabilježena uravnotežena točnost modela ocjene.
grade_model_f1_score	FLOAT	Zabilježena F1 vrijednost modela ocjene.

ensemble_accuracy	FLOAT	Zabilježena točnost zajedničkog predviđanja oba modela.
ensemble_kappa	FLOAT	Zabilježena vrijednost *kappa* zajedničkog predviđanja oba modela.
ensemble_balanced_accuracy	FLOAT	Zabilježena uravnotežena točnost zajedničkog predviđanja oba modela.
ensemble_f1_score	FLOAT	Zabilježena F1 vrijednost zajedničkog predviđanja oba modela.
prediction_point_configuration_json	JSONB	Konfiguracija točke predikcije u JSON formatu. Definira testove iz tekuće akademske godine koji se koriste za predviđanje.


```
[
    {
        "id": 14353,
        "group": "lab1abc",
        "is_in": true,
        "title": "1. laboratorijska vježba - ABC",
        "id_course": 2004,
        "max_score": 0.8,
        "id_academic_year": 2023,
        "test_instance_count": 739,
        "ignore_students_with_this_test": false
    },
    {
        "id": 14354,
        "group": "lab1c",
        "is_in": true,
        "title": "1. laboratorijska vježba - C",
        "id_course": 2004,
        "max_score": 0.7,
        "id_academic_year": 2023,
        "test_instance_count": 739,
        "ignore_students_with_this_test": false
    },

    ...

    {
        "id": 14446,
        "group": "mi",
        "is_in": true,
        "title": "Međuispit 2023/24 - prvi termin",
        "id_course": 2004,
```

```
        "max_score": 30,
        "id_academic_year": 2023,
        "test_instance_count": 440,
        "ignore_students_with_this_test": false
    },
    {
        "id": 14447,
        "group": "mi",
        "is_in": true,
        "title": "Međuispit 2023/24 - drugi termin",
        "id_course": 2004,
        "max_score": 30,
        "id_academic_year": 2023,
        "test_instance_count": 504,
        "ignore_students_with_this_test": false
    }
]
```


Kôd 3.2 Definirani testovi iz tekuće akademske godine kolegija UPRO za 1. točku predikcije

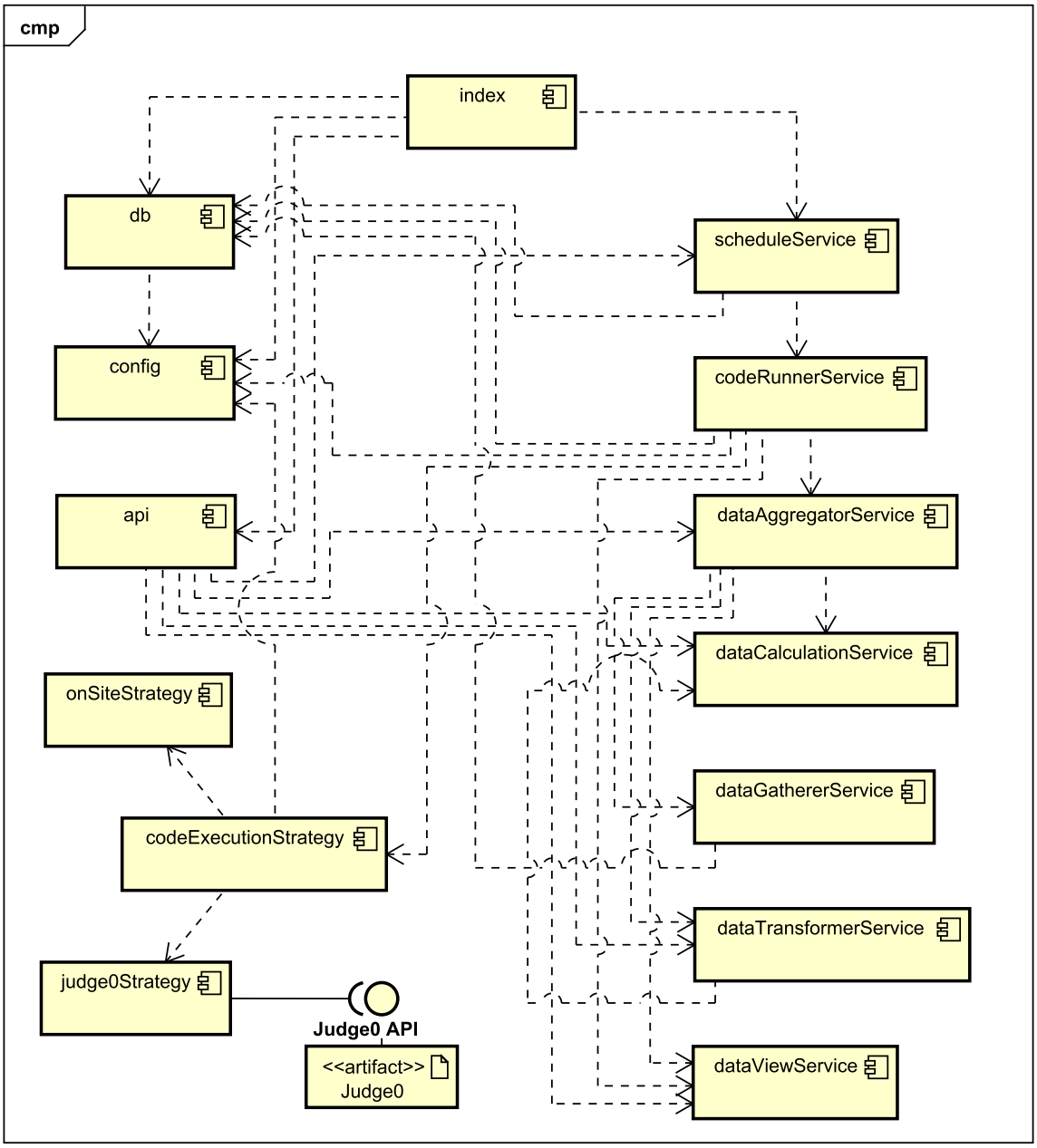
Posljednja tablica (**Tablica 3.4**) sadrži zapise o predviđenim uspjesima, povezane na pripadajuće studente. Samo predviđanje je također povezano na zapis točke predikcije iz koje je predviđanje doneseno. Uz same bilješke konačnog uspjeha, zapisuje se i bodovno stanje studenta za vrijeme donošenja predikcije te šanse pojedinih ishoda uspjeha, radi šireg uvida u studentov potencijal.

Tablica 3.4 Tablica predviđenih uspjeha studenata

Ime	Tip	Opis
id	INTEGER	Jedinstveni ID predviđenog uspjeha.
id_prediction_point	INTEGER	Jedinstveni ID koji povezuje određenu točku predikcije.
id_student	INTEGER	Jedinstveni ID koji povezuje određenog studenta.
at_the_time_of_forecasting_score	FLOAT	Postignuti bodovi za vrijeme predviđanja.
predicted_pass	INTEGER	Oznaka predviđenog pada ili prolaza.
predicted_grade	INTEGER	Oznaka predviđene ocjene.
fail_odds	FLOAT	Šansa pada kolegija.
pass_odds	FLOAT	Šansa prolaza kolegija.
grade_1_odds	FLOAT	Šansa jedinice kao konačne ocjene.
grade_2_odds	FLOAT	Šansa dvojke kao konačne ocjene.
grade_3_odds	FLOAT	Šansa trojke kao konačne ocjene.
grade_4_odds	FLOAT	Šansa četvorke kao konačne ocjene.
grade_5_odds	FLOAT	Šansa petice kao konačne ocjene.

3.2. Servis predviđanja uspjeha studenata

Srce sustava predviđanja uspjeha studenata je istoimeni servis. Razvijen kao samostalan modul, proširuje mogućnosti sustava Edgar s treniranjem modela predikcije uspjeha te davanja predviđenih uspjeha. Arhitektura samog servisa je inspirirana Edgarovom servisno orijentiranom arhitekturom. Cilj takvog pristupa je održati pristupačnost održavanja servisa, jer bi uvid u rad sustava bio sličan Edgarovom. Konačni modul sadrži 7 servisa, o kojima ovise glavne funkcije i API zahtjevi korisnika. Ujedno, bitniji dio sustava je i strategija izvršavanja kôda te se nude dvije strategije. Odabir koja strategija se koristi tijekom izvršavanja je definiran u konfiguracijskoj datoteci, koja služi istoj svrsi kao i konfiguracijska datoteka sustava Edgar. Pregledi zavisnosti pojedinih dijelova modula predviđanja mogu se vidjeti na dijagramu komponenti (**Slika 3.2**).



Slika 3.2 Dijagram komponenti - Servis predviđanja uspjeha studenata

Prvi servis, *Data Calculation Service*, obavlja složene izračune nad unesenim podacima. Jedna od glavnih funkcija servisa je izračunati značajku težine pojedine grupacije testova, a potom i njezine normalizacije. Druga funkcija se bavi izračunom vrijednosti zalaganja svakog studenta iz unesene generacije, formulom (3). Kao kraj funkcionalnosti koja se bavi zalaganjem, servis računa korelaciju izračunatih zalaganja studenata iz prije odabranih kolegija s postignutim uspjehom studenata iz tekuće akademske godine.

Sljedeći servis je usko vezan uz bazu podataka, jer je njegova glavna zadaća dohvat podataka koji se šalju ostalim servisima na procesiranje. To je zadaća komponente *Data Gatherer Service*, koja sadrži prije definiran SQL upit za dohvat postignutih bodova studenata na svakom testu iz definiranog kolegija i akademske godine (**Kôd 2.4**). Uz dohvat tih podataka, dohvaćaju se i studentova postignuća iz prijašnjih akademskih godina. Ti podaci se koriste za izračun značajke prijašnjeg zalaganja studenta na definiranim kolegijima.

Komponenta *Data Transformer Service* je servis zadužen za manipulaciju skupa podataka. U tu svrhu se definiraju funkcije dodavanja novih stupaca, čitanja vrijednosti pojedinih stupaca ili redova, izračun novih stupaca prema vrijednostima nekih drugih stupaca i slično. Najznačajnija funkcija komponente je procesiranje sirovih podataka koje se dobiju kao rezultat iz *Data Gatherer Service* komponente, što rezultira korijenskom skupu podataka za treniranje ili predviđanje s normaliziranim značajkama, ali bez ciljnih ili dodatnih značajki, jer to je uloga sljedeće komponente.

Data Aggregator Service sadržava funkcije koje akumuliraju različite podatke u konačan skup podataka, koji se nadalje sprema i obrađuje u sustavu predviđanja. Ovdje se spajaju dodatne značajke i ciljne značajke na skup podataka dobivenog iz *Data Transformer Service* komponente, što zaključuje konačan skup podataka za treniranje ili predviđanje ovisno o kontekstu u kojem se komponenta koristi. Ovdje je definirana i funkcija koja računa statistiku raspodjele ocjena te pada ili prolaza pojedinog studenta.

Za spremanje tog skupa podataka se koristi komponenta *Data View Service*. Omogućeno je spremanje u CSV i TXT formatu. CSV format je najčešće korišten, jer različita programska okruženja mogu čitati taj format, dok se TXT koristi više kao sredstvo za otklanjanje pogrešaka u sustavu. Zapisuje iste podatke kao CSV, ali ih formatira u jednostavnu tablicu koju ljudi mogu brže analizirati od CSV datoteke. Naravno, uz samo spremanje u memoriju omogućava i učitavanje skupa podataka iz CSV datoteka.

Prije spomenuti servisi se koriste u komponenti *Code Runner Service*. Ta komponenta prvo sakuplja sve potrebne statične datoteke R kôda, skripte metoda spremljene u bazi, skripte za izvršavanje kôda ovisno o operacijskom sustavu i skup podataka u jedan direktorij, koji se imenuje jedinstvenom oznakom, poput UUID, radi izolacije prikupljenih datoteka u slučaju istovremenog prikupljanja datoteka iz drugog zahtjeva. Ujedno se pridjeljuje i JSON datoteka, koja zapisuje sve podatke koji se smatraju unosom, koju R skripta čita prilikom izvođenja programa. Nakon izvedbe kôda za treniranje ili predviđanje, sustav čita rezultate izvršenog kôda iz JSON datoteka koje su se generirale pred kraj izvršavanja te sprema

podatke i metrike generiranih modela i logove u datotečni sustav i bazu podataka. Ako se izvršavalo predviđanje, sustav čita novostvorenu CSV datoteku koja bilježi predviđene uspjehe uz popratne statistike te se svi ti podaci šalju u bazu podataka. Sve prije postavljene i novo generirane datoteke se na kraju brišu kao i privremeni direktorij u kojem su se nalazile.

Komponenta *Code Execution Strategy* upravlja gdje se točno spomenuti kôd izvršava. Prva strategija koristi Judge0 servis za izvršavanje programskog kôda. Strategija prvo sažima sve potrebne datoteke te nastalu ZIP datoteku kodira u *Base64* format. Po specifikacijama API-ja [10], samo u tom formatu se mogu slati bilo kakve dodatne datoteke uz samu skriptu kôda. Prva potrebna datoteka je *bash* skripta koja mora biti naziva `run`. Pomoću nje Judeg0 zna koje okruženje i programski jezik korisnik želi koristiti. Uz to se specificira koja R skripta se izvršava. Moguće su dvije opcije: skripta za unakrsnu provjeru i treniranje ili skripta za predviđanje. R skripta koja započinje izvršavanje malo prije odabrane R skripte ima ulogu omotača koji modificira standardni ispis i lokaciju spremanja novostvorenih datoteka koje stvara glavna R skripta. Izvorna implementacija ove omotne skripte u svrhu proširenja Edgara kako bi integrirao strojno učenje pomoću Judge0 servisa se koristi u radu P. Benjaka [4]. S ovom omotnom skriptom (**Kôd 3.3**), R kôdom, koji je poslan na izvršavanje, upravlja se na sljedeći način: prvo se stvaraju direktoriji *logs* i *files* za pohranjivanje standardnog izlaza poslanog kôda i datoteka generiranih izvršavanjem kôda. U sljedećem koraku bilježe se informacije o trenutnim datotekama i direktorijima u radnom direktoriju. Zatim se izvršava poslana R skripta, čije se ime učitava iz popratne JSON datoteke. Nakon što se poslani kod izvrši, skripta provjerava koje su datoteke i direktoriji prisutni u trenutnom direktoriju. Uspoređujući to s popisom datoteka i direktorija prije izvođenja koda, identificira nove datoteke i direktorije stvorene tijekom izvođenja. Naredba `sink()` preusmjerava sav standardni izlaz u datoteku *files/files.txt*, koja je dio skupa datoteka generiranih tijekom izvođenja kôda. Upotreba naredbe `sink()` ključna je za sprječavanje sustava Judge0 da ovaj sadržaj prepozna kao standardni izlaz. Ovaj proces osigurava da su sve datoteke i ispisi stvoreni tijekom izvođenja koda izolirani. Datoteke kao što su modeli i JSON rezultati premještaju se u direktorij *files* jer sustav Judge0 ne dopušta kompresiju datoteka u trenutnom direktoriju, ali dopušta kompresiju poddirektorija. Te se datoteke zatim komprimiraju, a sadržaj dobivene komprimirane datoteke se ispisuje na standardni izlaz u *Base64* formatu, koji sustav Judge0 prepoznaje i uključuje u odgovor na poslani zahtjev u *stdout* polje.

```
library(jsonlite)
params <- fromJSON("parameters.json")
main_source_file_path <- params$main_source_file_path

dir.create("logs")
dir.create("files")
files_before <- list.files(".")
sink("logs/log.txt")
library("base64enc")
source(main_source_file_path)
files_after <- list.files(".")
files_new <- setdiff(files_after, files_before)
files_new_path <- paste("./files", files_new, sep = "/")
file.copy(files_new, files_new_path)
sink()
tar("source.tar.gz",
    c("logs", "files"),
    compression = "gzip"
)
cat(base64encode("source.tar.gz"))
```

Kôd 3.3 Izmijenjena omotna R skripta za pravilno formatiranje ispisa u Judge0 okruženju, izvor originalne inačice: [4]

Nakon što se pripreme sve potrebne datoteke, sustav šalje HTTP zahtjev Judge0 servisu, gdje se specificira najveća moguća memorija i vremensko ograničenje izvršavanja programa po Judge0 dokumentaciji. Nakon slanja zahtjeva za izvršavanje, sustav svakih 5 milisekundi provjerava stanje poslanog zahtjeva, sve dok izvršavanje ne završi i sustav dobije natrag rezultate treniranja ili predviđanja. Kako je Judge0 primarno napravljen za izvršavanje manjih programa, potrebno je paziti koliko resursa treniranje i usporedba više modela koristi. Tijekom istraživanja, kolegiji s manje od tisuću studenata u skupu podataka za treniranje su prolazili proces unakrsne provjere u Judge0 okruženju bez poteškoća. Problem se pojavljuje s kolegijem UPRO, za koji se u fazi treniranja koristilo 1523 studenata, gdje se u pola procesa zatražio prekid izvršavanja programa. Kako je baratanje većim skupom podataka u kontekstu strojnog učenja neizbježno, potrebna je druga strategija koja ima manja ograničenja nad dostupnim računalnim resursima.

Alternativna strategija izvršava kôd u okruženju u kojem je sami servis poslužen. Tim pristupom je faza treniranja kolegija UPRO prošla uspješno, za razliku kod treniranja modela u Judge0 sustavu. Radi dosljednosti, strategija sprema sve potrebne datoteke u privremeni

direktorij kao i Judge0 strategija i koristi omotnu R skriptu, ali u ovom slučaju nije potrebno komprimiranje i kodiranje tih datoteka.

Za izvršavanje R kôda s Python bibliotekama u obje strategije koristi se virtualno okruženje postavljeno pomoću programa *Conda*. To virtualno okruženje ima instalirane potrebne verzije R i Python programskog jezika uz popratne biblioteke kako bi sustav predikcije radio ispravno. Postavljanje tog okruženja se razlikuje od lokalnog sistema i Judge0 servisa te se može pronaći u repozitoriju izvornog koda, čija je poveznica u privitku uz ovaj rad.

Zadnji servis *Schedule Service* je komponenta koja obrađuje zahtjeve za zakazivanje treniranja i predviđanja modela te je zaslužna za pokretanje funkcija iz *Code Runner Service* komponente. Ujedno ažurira bazu podataka o tim terminima, kako bi prilikom ponovnog pokretanja modula komponenta mogla ponovo učitati zakazane termine. Nakon izvršavanja zakazanog zadatka, servis se pobrine o spremanju potrebnih podataka i rezultata u bazu podataka.

API komponenta sadrži krajnje točke kojima sustav Edgar pristupa kao posrednik korisničkih zahtjeva. Tako se i osigurava sigurnost modula predviđanja, jer sustav Edgar brine o sjednicama korisnika i upravlja nad njihovim ovlastima. Sama komponenta API izvršava svoje funkcije tako što se poziva na prije opisane servise. Definirane rute su:

- **POST /api/calculate-course-correlations**
    - Računa se korelacija definiranih relevantnih kolegija s kolegijem za koji se priprema skup podataka za treniranje modela predikcije, određeno iz predane konfiguracije predviđanja za tekuću akademsku godinu. Nakon izračuna, vraća se izmijenjena konfiguracija predviđanja s izbačenim relevantnim kolegijima koji nisu imali zadovoljavajuću korelaciju.
- **POST /api/calculate-student-success-training-data-stats**
    - Računa se i vraća statistika skupa podataka koji se generira iz predane konfiguracije predviđanja za tekuću akademsku godinu.
- **POST /api/schedule-training**
    - Zakazuje se vrijeme izvršavanja treniranja modela predikcije za definirani kolegij, akademsku godinu i redni broj točke predikcije.
- **POST /api/schedule-prediction**
    - Zakazuje se vrijeme predviđanja prije treniranog modela predikcije za definirani kolegij, akademsku godinu i redni broj točke predikcije.

3.3. Proširenje sustava Edgar

Sustav Edgar je proširen s jednom servis komponentom i jednom API komponentom. Komponenta *Student Success Forecast Service* se koristi za pozivanje na krajnje točke iz prošlog poglavlja. Uz njih, zadužena je za provjeru statusa modula predviđanja i neke manje pomoćne funkcije validacije i filtriranja podataka. Nadalje, komponenta API omogućuje pravilan rad korisničkog sučelja s rutama:

Studentovo sučelje

- **GET /student-forecast/prediction?prediction-point-ordinal=#**
    - Dohvaća se personalizirana HTML stranica predviđenog uspjeha za studenta. To se ostvaruje pomoću funkcije sustava Edgar, `appRender`. Ako je definirana točka predikcije, onda se uzimaju predviđanja iz te točke, a u slučaju nespecificirane točke predikcije daju se najnovija predviđanja.

Nastavnikovo sučelje

- **GET /student-forecast**
    - Dohvaća se nastavnikovo web sučelje za upravljanje sustavom predviđanja uspjeha.

Dostupnost servisa

- **GET /student-forecast/api/status**
    - Provjerava se dostupnost modula predviđanja uspjeha.

Informacije korisničke sjednice

- **GET /student-forecast/api/session-course-year**
    - Dohvaća se odabrani kolegij i akademska godina sjednice.

Kolegiji

- **GET /student-forecast/api/courses?search-term=''**
    - Dohvaća se skup kolegija čije se ime ili akronim parcijalno podudaraju s poslanim pojmom za pretraživanje.

- **GET /student-forecast/api/courses/enrollment?search-term=''**
    - Dohvaća se skup kolegija čije se ime ili akronim parcijalno podudaraju s poslanim pojmom za pretraživanje s pridruženim brojem upisanih studenata kroz zadnjih 5 akademskih godina.

Skripte metoda

- **GET /student-forecast/api/methods**
    - Dohvaćaju se informacije svih spremljenih implementacija metoda strojnog učenja, bez samih skripta tih metoda.
- **GET /student-forecast/api/methods/select-ids?ids=#,#,#...**
    - Dohvaćaju se informacije spremljenih implementacija metoda strojnog učenja definiranih u poslanoj listi identifikacijskih brojeva, bez samih skripta tih metoda.
- **POST /student-forecast/api/methods**
    - Sprema se nova metoda.
- **GET /student-forecast/api/methods/pass-fail**
    - Dohvaćaju se informacije spremljenih implementacija metoda strojnog učenja za predviđanje prolaza studenta, bez samih skripta tih metoda.
- **GET /student-forecast/api/methods/grade**
    - Dohvaćaju se informacije spremljenih implementacija metoda strojnog učenja za predviđanje ocjene studenta, bez samih skripta tih metoda.
- **GET /student-forecast/api/methods/:id**
    - Dohvaća se određena metoda i skripta te metode.

Pomoćne informacije za postavljanje konfiguracije predviđanja

- **GET /student-forecast/api/course-overview**
    - Dohvaća se pregled broja testova i srednja vrijednost koliko studenata je pristupilo pojedinom testu po akademskoj godini.
- **GET /student-forecast/api/tests-overview?id_year=#**
    - Dohvaćaju se informacije testova iz odabrane akademske godine uz broj studenata koji je pristupio testu.
- **GET /student-forecast/api/tests-overview/current-academic-year**
    - Dohvaćaju se informacije testova iz akademske godine definirane u korisničkoj sjednici uz broj studenata koji je pristupio testu.

Konfiguracija predviđanja uspjeha

- **GET /student-forecast/api/forecast-setup-json**
  - Dohvaća se konfiguracija predviđanja za kolegij i akademsku godinu definiranih u sjednici.
- **POST /student-forecast/api/forecast-setup-json**
  - Sprema se konfiguracija predviđanja za kolegij i akademsku godinu definirane u sjednici te se stvaraju novi zapisi u tablici točaka predikcije ovisno koliko je bilo definirano u predanoj konfiguraciji
- **GET /student-forecast/api/forecast-setup-json/is-configured**
  - Provjerava se postojanje konfiguracije predviđanja za definirani kolegij i akademsku godinu iz sjednice.
- **GET /student-forecast/api/forecast-setup-json/grade-thresholds**
  - Dohvaćaju se definirani bodovni pragovi ocjena konfiguracije predviđanja.

Točke predikcije

- **GET /student-forecast/api/prediction-points/overview**
  - Dohvaćaju se sve točke predikcije za trenutnu akademsku godinu i kolegij, s ciljem pregleda njihovih stanja treniranja modela i predviđanja.
- **PUT /student-forecast/api/prediction-points/:ordinal/training-schedule**
  - Ažurira se zakazani termin treniranja modela specificirane točke predikcije.
- **PUT /student-forecast/api/prediction-points/:ordinal/prediction-point-configuration-json**
  - Ažurira se JSON struktura definiranih testova pomoću kojih se izvršavaju predviđanja za tekuću akademsku godinu za specificiranu točku predikcije.
- **PUT /student-forecast/api/prediction-points/:ordinal/prediction-schedule**
  - Ažurira se zakazani termin predviđanja uspjeha studenata specificirane točke predikcije.

Predviđeni uspjesi

- **GET /student-forecast/api/student-prediction-data/:ordinal**
  - Dohvaćaju se svi predviđeni uspjesi i ishodi svih uspjeha pojedinog studenta nakon specificirane točke predikcije.

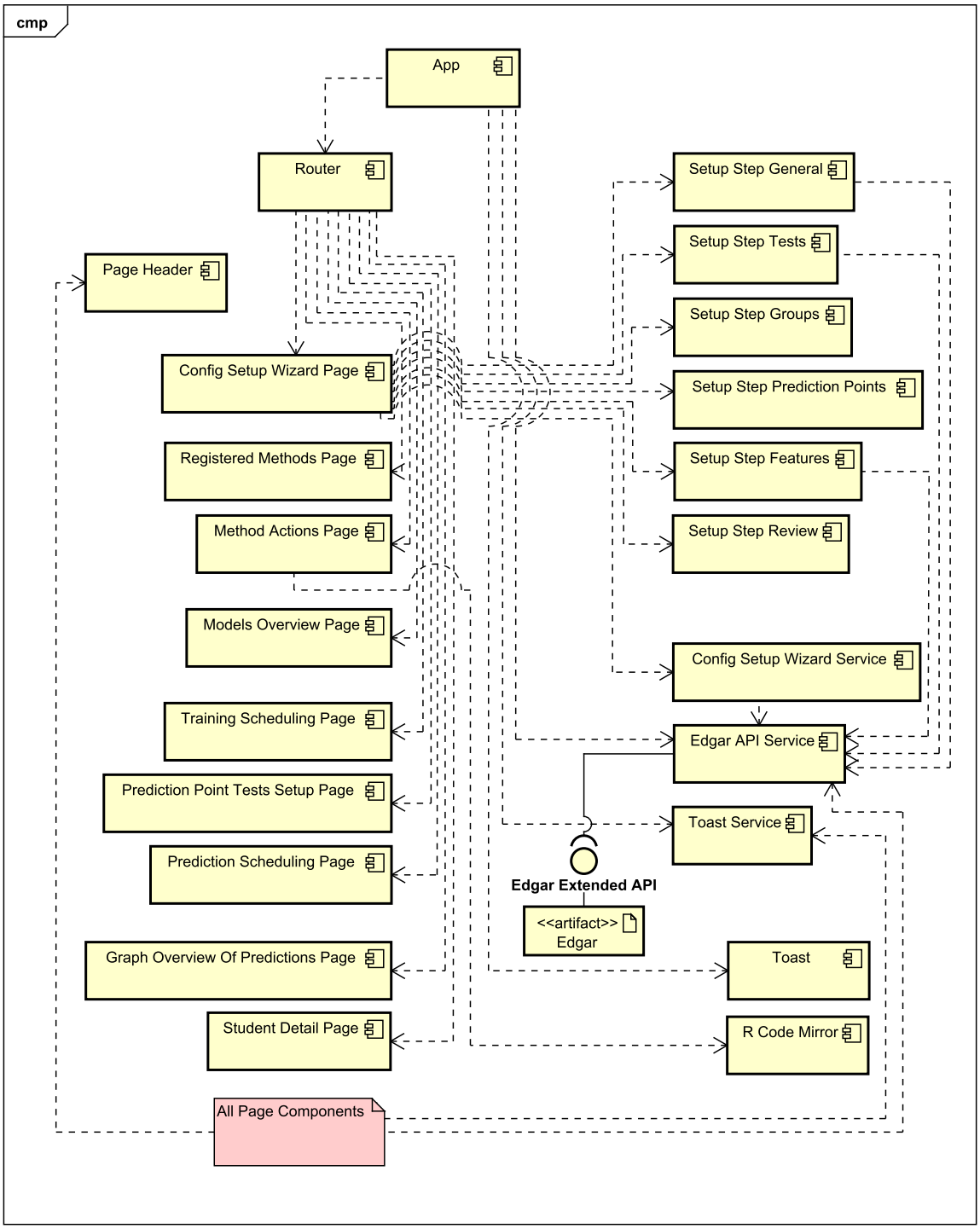
- **GET /student-forecast/api/student-prediction-data/:ordinal/overview**
    - Dohvaća se pregled svih predviđenih uspjeha i metrika korištenih modela nakon specificirane točke predikcije u svrhu grafičkog prikaz i analizu tih podataka.

Sve opisane rute zahtijevaju autentificiranu korisničku sjednicu te autoriziranu ulogu „nastavnik“. Jedino ruta **GET/student-forecast/prediction** je namijenjena studentima i time samo ona zahtjeva ulogu „student“.

3.4. Korisničko sučelje

3.4.1. Nastavnikovo sučelje

Edgar, kao sudionik cjelokupnog sustava predviđanja uspjeha studenata, sadrži zadaću posluživanja web sučelja za upravljanje tim sustavom. To sučelje je projektirano kao jednostranična web aplikacija, pisana pomoću Angular radnog okvira. Komponente aplikacije dijele se na stranice sučelja, samostalne ili pomoćne komponente i servise koji izvršavaju globalne funkcije dostupne svim komponentama aplikacije. Uz komponente, razvijene su i pomoćne funkcije te funkcije crtanja i upravljanja grafikonima. Pregled arhitekture web aplikacije se može vidjeti na slici dijagrama komponenti (**Slika 3.3**). Može se primijetiti da sve komponente stranica komuniciraju sa sustavom Edgar preko komponente *Edgar API Service*. Uz nju postoji globalna komponenta *Toast Service*, koja pri uspješnim izvršavanjima zahtjeva ili nastalih grešaka prikaže poruku o stanju poslanog zahtjeva te se makne nakon određenog vremena ili nakon što ju korisnik odbaci.



Slika 3.3 Dijagram komponenti - Nastavnikovo sučelje

Page Header komponentu koristi svaka komponenta stranice web aplikacije. Tako pojedina komponenta stranice ima mogućnost dodavanja vlastitog sadržaja u zaglavlje, ako je potrebno prikazati dodatan kontekst ili navigacijsku poveznicu. Inače, glavna funkcija zaglavlja je prikazati akademsku godinu i kolegij, izvučene iz trenutne korisničke sjednice,

te poveznicu povratka na početnu stranicu sustava Edgar. Uz tu poveznicu, komponenta može postaviti zaglavlje da sadrži poveznicu povratka na početnu stranicu samog web sučelja za predviđanje uspjeha, što čini svaka komponenta stranice osim početne.

Ta početna stranica je komponenta *Models Overview Page*. Njezina svrha je prikaz svih točaka predikcije definiranih u konfiguraciji predviđanja, ako ta konfiguracija postoji. Podaci tih točaka predikcije se mijenjaju kroz vrijeme, ovisno kako korisnik njima upravlja. Uglavnom sve ostale stranice služe za upravljanje pojedinom točkom predikcije, osim stranice vezane za metode i stranice izgradnje konfiguracije predviđanja. Time se svaka druga stranica sučelja može doseći iz početne stranice, ako stanje točke predikcije dozvoljava navigaciju na njih.

Komponenta *Registered Methods Page* je stranica web sučelja koja prikazuje sve spremljene informacije o implementacijama metoda strojnog učenja. Bitno je naglasiti da se u toj stranici ne može vidjeti kôd metoda, već korisnik mora navigirati na stranicu *Method Actions Page*, gdje može vidjeti sve o odabranoj metodi. Ta ista komponenta služi i za dodavanje novih metoda, ako je otvorena u prikladnom stanju. Kako bi se taj R kôd mogao pregledati i uređivati, koristi se komponenta *R Code Mirror*. Ona je specijalizirana za programski jezik R te kao bazu koristi zajedničku komponentu *Code Mirror* koja je definirana u Edgarovom okruženju.

Najopširnija komponenta je stranica *Config Setup Wizard Page*, jer služi kao pristupačan način konfiguriranja postavki predviđanja za određenu akademsku godinu i kolegij. Ona također upravlja sa 6 komponenti, koje predstavljaju pojedini korak konfiguracije predviđanja. Svaki korak je definiran svojim imenom, ulaznim te izlaznim podacima i instancom same komponente koraka. Tako se po potrebi može jednostavno napisati i dodati nova komponenta koraka, ako bude potrebe. Matična komponenta se brine o navigaciji između njih te upravlja procesiranim podacima, koje dobije iz pojedinog koraka. Spomenuti podaci se akumuliraju u komponenti *Config Setup Wizard Service*, čiji je glavni cilj spakirati podatke u konačnu JSON strukturu, koja se koristi dalje u sustavu.

Nadalje su redom dane prije spomenute komponente koraka konfiguracije predviđanja. Prvi korak je sadržan u *Setup Step General* komponenti, gdje se biraju akademske godine iz kojih će se vući podaci za treniranje modela, bodovni pragovi ocjena i koje točno metode će se koristiti za međusobnu usporedbu i odabir najboljeg modela. Drugi korak je komponenta *Setup Step Test*, koja omogućuje razvrstavanje testova iz prije odabranih akademskih godina u one koji su vezani za predviđanje uspjeha i one koji se neće koristiti, jer spadaju izvan

kontinuirane nastave ili služe za neku probnu svrhu. Sljedeći korak sadrži komponenta *Setup Step Groups*, u kojoj se stvaraju grupacije te se u njih razvrstavaju testovi određeni iz prošlog koraka. Nakon što je svaki test pridružen nekoj grupaciji i ni jedna grupacija nije ostala prazna, prelazi se na sljedeći korak uz komponentu *Setup Step Prediction Point*. U njoj se određuje broj točaka predikcije te se u svaku točku predikcije pridružuju testovi. Bitno je naglasiti da ako jedna točka predikcije u jednoj akademskoj godini sadrži test određene grupacije, ostale akademske godine moraju sadržavati test iste grupacije u toj točki. Sljedeći i predzadnji korak je komponenta *Setup Step Features*. U tom koraku se bira koje točno skupine značajki će biti korištene za treniranje modela od glavnih tri: broj postignutih bodova u grupaciji, udio bodova za konačnu ocjenu grupacije i težina postizanja bodova u grupaciji. Pritom je samo značajka postignutih bodova obavezna značajka. Ostale dvije mogu biti uključene ili ne moraju ovisno o nahođenju nastavnika. Na primjer, moguće ih je izostaviti ako se koristi samo jedna akademska godina za treniranje modela, jer tada oni postaju konstante te bi mogle davati nepotreban šum u modele predviđanja. Nije nužno velika greška i ostaviti ih uključene u toj situaciji, jer neke korištene metode strojnog učenja prepoznaju i zanemaruju značajke konstantne vrijednosti, poput logističke regresije. Uz odabir glavnih značajka, mogu se pretražiti svi kolegiji zapisani u sustavu Edgar te dodati kao dodatna značajka zalaganja studenta na tom kolegiju. Na kraju hoće li ta značajka ostati kao sastavni dio skupa podataka za treniranje odlučuje sustav po minimalnom kriteriju korelacije. Zadnji korak se nalazi u komponenti *Setup Step Review* te ona samo prikazuje sažet pregled prije odabranih postavki konfiguracije predviđanja. Uz to, ako se korisnik nalazi na zadnjem koraku, matična komponenta omogućuje slanje konfiguracije na poslužitelj.

Komponente stranica *Trainig Scheduling Page* i *Prediction Scheduling Page* imaju vrlo sličnu strukturu i funkciju zakazivanja termina izvođenja određenog zadatka za specificiranu točku predikcije, gdje prva služi za zakazivanje termina treniranja te druga za termin predviđanja. Bitno je napomenuti da se termin predviđanja ne može zakazati, ako korisnik prvo nije odredio testove iz tekuće akademske godine koji će se koristiti u predviđanju. To se obnaša u komponenti *Prediction Point Tests Setup Page*.

Nakon što su se izvršila predviđanja za točku predikcije, korisnik može pregledati te informacije u sveobuhvatnom pregledu generacije ili točno predviđanje za svakog studenta. Sveobuhvatni pregled se nalazi u komponenti *Graph Overview Of Predictions Page*, gdje se može vidjeti grafički izvještaj o točnosti pojedinog modela te raspored predviđenih uspjeha

i pregled prijašnjih uspjeha izvučenih iz podataka za treniranje modela. Ti grafikoni su napisani pomoću D3.js biblioteke i to su brzinomjeri, stupčasti grafovi, kružni grafovi i linijski grafovi. Izgledi tih vizualizacija se mogu vidjeti u kasnijem poglavlju. Nadalje, detaljniji prikaz daje komponenta *Student Detail Page*. Ona prikazuje tablični prikaz studenata s popratnim informacijama o postignutim bodovima za vrijeme davanja predviđanja te uz samo predviđanje i ishode pojedinih uspjeha. Radi različitog broja upisanih studenata po kolegiju, ta tablica se može filtrirati, pretraživati i sortirati prema ponuđenim opcijama.

3.4.2. Studentovo sučelje

Zadnja cjelina sustava predviđanja uspjeha studenata je opcija prikaza predviđenog uspjeha samom studentu. Ovdje student može vidjeti svoje ishode pada ili prolaza te ishode pojedine ocjene. Prikaz se upotpunjuje predviđenom zaključnom ocjenom koju sustav dobije kombinacijom ishoda pada ili prolaza i ishoda ocjene, a pseudokôd te odluke se može vidjeti u isječku (**Kôd 3.4**). Spomenuti ishodi su prikazani stupčastim grafom, koji se može vidjeti u 4. poglavlju.

```
IF predicted_pass == 0 THEN
    ensemble_grade = 1
ELSE IF predicted_pass == 1 AND predicted_grade == 1 THEN
    ensemble_grade = 2
ELSE
    ensemble_grade = predicted_grade
```

Kôd 3.4 Pseudokôd odabira zaključne ocjene

3.5. Korištene tehnologije

3.5.1. Programski jezici R i Python 3

Programski jezik R je specijaliziran za statističku analizu i vizualizaciju podataka. Kako bi se funkcionalnost proširila na domenu strojnog učenja koriste se vanjske biblioteke, koje implementiraju algoritme spomenute prije u radu. Nažalost, zahtijevani algoritam izgradnje neuronskih mreža više nije podržan u najmodernijem izdanju za R okruženje. S druge strane, programski jezik Python 3 je najpopularniji odabir u domeni strojnog učenja i ima podršku široke zajednice, pogotovo biblioteke za neuronske mreže kao Tenserflow i PyTorch [2]. Konačna implementacija sustava predviđanja uspjeha koristi uglavnom algoritme pisane za R programski jezik, osim neuronskih mreža za koje se koristi C++ biblioteka LibTorch, na kojoj je i PyTorch baziran, preko Torch biblioteke. Samo postavljanje tog okruženja predstavlja značajan izazov, jer zahtjeva konfiguraciju različitih biblioteka, koje nužno ne surađuju dobro s različitim verzijama programskog jezika i operacijskog sustava.

3.5.2. Conda

Kao rješenje nastalog problema koristi se upravitelj programskih paketa i okruženja Conda. Prvobitno nastao kao rješenje upravljanja raznovrsnim paketima u domeni znanosti podataka za Python programski jezik, sada pruža potporu za bilo koji jezik i radi na više operacijskih sustava. Pomoću programa Conda se definira virtualno okruženje u kojem se definiraju R i Python programski jezici s popratnim bibliotekama te se upravitelj paketa brine o instalaciji dodatnih ovisnosti poput DLL datoteka koje su potrebne za pravilan rad LibTorch biblioteke. Postavljeno okruženje se onda može pokrenuti na lokalnom računalu ili Judge0 servisu i na taj način se koristi isto okruženje za treniranje modela i predviđanje, neovisno koju strategiju izvršavanja kôda koristi modul predviđanja uspjeha studenata.

3.5.3. PostgreSQL

Korišteni sistem upravljanja bazama podataka je PostgreSQL. Kao glavna značajka tog sistema se navode transakcije s ACID svojstvima. Kako sustav Edgar već koristi

PostgreSQL za spremanje svojih podataka, modul predviđanja uspjeha samo proširuje postojeću bazu, definirajući vlastitu shemu koja je povezana na već postojeće zapise.

3.5.4. Node.js i Express.js

Node.js je izvršno okruženje za izvršavanje programskog koda pisanog u JavaScript programskom jeziku, izvan internetskog preglednika. Koristi arhitekturu događaja za asinkrono izvođenje procesa, što optimizira propusnost i skalabilnost u web aplikacijama. Te web aplikacije su uglavnom projektirane pomoću razvojnog okvira Express.js. Kako je Edgar razvijen na te dvije tehnologije, modul predviđanja uspjeha koristi iste okvire radi lakšeg razumijevanja oba sustava i njihove suradnje.

3.5.5. Node Schedule

Node Schedule služi za raspoređivanje poslova u Node.js okruženju. U implementaciji imitira raspoređivač poslova Cron iz UNIX operacijskog sustava. Mada je Cron napravljen za raspoređivanje periodičkih poslova, Node Schedule nudi i raspoređivanje poslova u točno definirani datum i vrijeme. Taj pristup najbolje paše u svrhe zakazivanja treniranja i predviđanja u modulu predviđanja uspjeha, jer ti poslovi nisu namijenjeni za periodičko izvođenje, već za jednokratno.

3.5.6. Docker

Projektirani modul predviđanja uspjeha ima poteškoće s pravilnim radom na različitim operacijskim sustavima i okruženjima poslužiteljskih računala. Obavljanje na različitim operacijskim sustavima bi zahtijevalo ručnu konfiguraciju tih okruženja i instalacije potrebnih tehnologija kako bi modul ispravno izvršavao svoje zadatke. Za rješenje tih problema koristi se Docker, kojemu je zadaća kontejnerizacija aplikacije u virtualni operacijski sustav, postavljen za ispravan rad poslužene aplikacije. Pomoću njega se na istom računalu može izvoditi više aplikacija u izolaciji te one ne ovise o operacijskom sustavu matičnog računala. Ujedno se mogu definirati koraci instalacije i postavke okruženja koje su potrebne za rad aplikacije, što omogućuje automatiziran način pokretanja modula

predviđanja uspjeha na različitim poslužiteljima, odnosno operacijskim sustavima. Docker se također koristi i u servisu Judge0.

3.5.7. Judge0

Judge0 je svestran, pouzdan i skalabilan sustav za izvršavanje kôda otvorenog pristupa izgrađen s modernim modularnim dizajnom, što ga čini primjenjivim na različitim računalima i operativnim sustavima. Omogućuje poboljšanu funkcionalnost i jednostavnost korištenja u usporedbi sa starijim platformama za izvršavanje udaljenog kôda, osobito od koristi sustavima koji procjenjuju i ocjenjuju korisnički kôd prema određenim testnim slučajevima. Judge0 je doživio široku primjenu na FER-u unutar sustava Edgar, gdje je bio ključan u automatizaciji ocjenjivanja testnih zadataka na nekoliko kolegija. Njegov jednostavan API uključuje dvije glavne krajnje točke: jednu za slanje koda u različitim programskim jezicima i drugu za praćenje statusa i rezultata izvršenja koda, kao što su izlaz, potrošnja memorije i vrijeme izvršenja [4].

Kako bi modul predviđanja uspjeha mogao raditi, bilo je potrebno promijeniti postavke sustava izvršavanja kôda, kako bi dozvolio unos najvećeg mogućeg vremenskog ograničenja i dostupne memorije, naspram standardnih postavka koje su predviđale izvađanje manje resursno zahtjevnih programa. Modul predviđanja u svom zahtjevu specificira izvršavanje programskog kôda identifikacijskog broja 89. Taj broj označava program koji je raspoređen kroz više izvršnih datoteka. API zahtjev s tim identifikacijskim kôdom traži da sve datoteke budu pakirane u kodiranu *base64* zip arhivu, uključeno s glavnim izvršnim kôdom te *shell* skriptom, imenovana `run` po specifikaciji API-ja, koja opisuje kako pokrenuti program. Modul tu skriptu koristi za pokretanje Conda okruženja te izvršavanje treniranja ili predviđanja u tom virtualnom okruženju. API zahtjev s parametrima koje koristi strategija izvršavanja kôda u Judge0 servisu se može vidjeti u isječku kôda (**Kôd 3.5**) te uz nju i pogled na upakiranu `run` skriptu (**Kôd 3.6**). Naravno, to zahtjeva instalirati programa Conda u Judge0 servis te postaviti prikladno virtualno okruženje. Unatoč tomu, tijekom istraživanja uspostavilo se da najveće moguće vrijednosti dostupne memorije nisu bile dovoljne za treniranje i usporedbu modela koji sadržavaju skup podataka od oko 1500 studenata.

```
const POLLING_TIME_MS = config.POLLING_TIME_MS || 5000;

async function postCodeSubmission(base64EncodedZip) {
    // Send POST request
    const postResponse = await axios.post(`${config.JUDGE0_API_URL}/submissions/`, {
        additional_files: base64EncodedZip,
        language_id: 89
    }, {
        params: {
            base64_encoded: false,
            wait: false,
            memory_limit: 1000000,
            cpu_time_limit: 43200,
            wall_time_limit: 86400
        }
    });

    const token = postResponse.data.token;

    let statusId = 1;
    let getResponse;

    // Poll the server until the status changes
    while (statusId === 1 || statusId === 2) {
        winston.info(`Polling Judge0 for submission ${token}...`);
        await new Promise(resolve => setTimeout(resolve, POLLING_TIME_MS)); // wait for 5 seconds

        getResponse = await
axios.get(`${config.JUDGE0_API_URL}/submissions/${token}?base64_encoded=false&fields=stdout,statu
s`);
        statusId = getResponse.data.status.id;
    }
    return getResponse;
}
```

Kôd 3.5 Zahtjev modula predviđanja uspjeha na Judge0 API

```
#!/bin/bash

# Activate the conda base environment
. /usr/local/miniconda/bin/activate

# Activate the conda student forecast environment
conda activate /usr/local/student-predictions-env

Rscript judge0-script.R
```

```
# Deactivate the conda environment
conda deactivate

# Deactivate the conda base environment
conda deactivate
```

Kôd 3.6 Skripta za pokretanje kôda sustava predviđanja u Judge0 servisu

3.5.8. Angular

Angular je okvir za razvoj aplikacija razvijen u Googleu. Dizajniran je za razvoj jednostraničnih web aplikacija koje mogu raditi na različitim uređajima. U Angularu, aplikacije su izgrađene od korijenskog modula, koji može uključivati više komponenti za grupiranje i prikaz različitih funkcionalnosti. Svaka komponenta sastoji se od razreda komponente, metapodataka i HTML predloška. Zatim još postoje servisi, koji se koriste za dijeljenje funkcionalnosti između komponenti. Nastavnikovo sučelje za upravljanje sustavom predviđanja uspjeha studenata je razvijeno u tom okruženju.

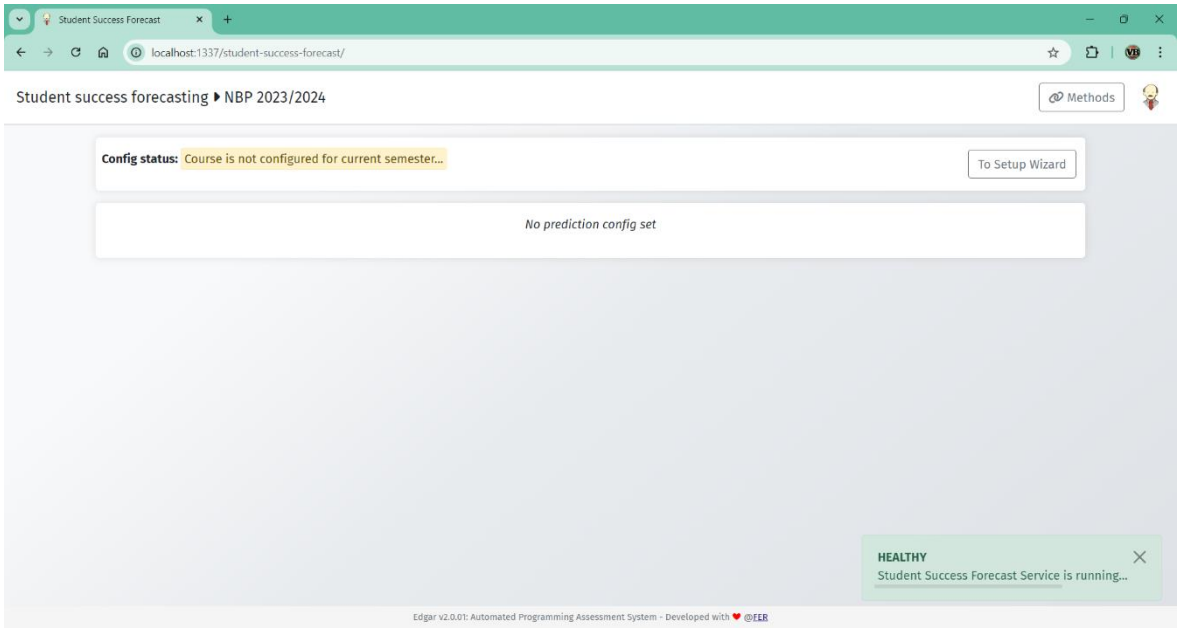
3.5.9. D3.js

Različite vizualizacije koje se mogu vidjeti u nastavnikovom i studentovom sučelju su napisane pomoću JavaScript biblioteke D3.js. Ime te biblioteke označava dokumente bazirane na podacima (engl. *Data Driven Documents*) te kako ime naznačuje, služi za izradu dinamičnih i interaktivnih vizualizacije na web preglednicima. Te vizualizacije u sustavu predviđanja pružaju prikladne grafikone za analizu predviđenih uspjeha nad cijelom generacijom ili pojedinog studenta i sadrže interaktivne elemente poput info-oblačića i isticanja jedne grupe elemenata grafikona nad drugima.

4. Sustav predviđanja u praksi

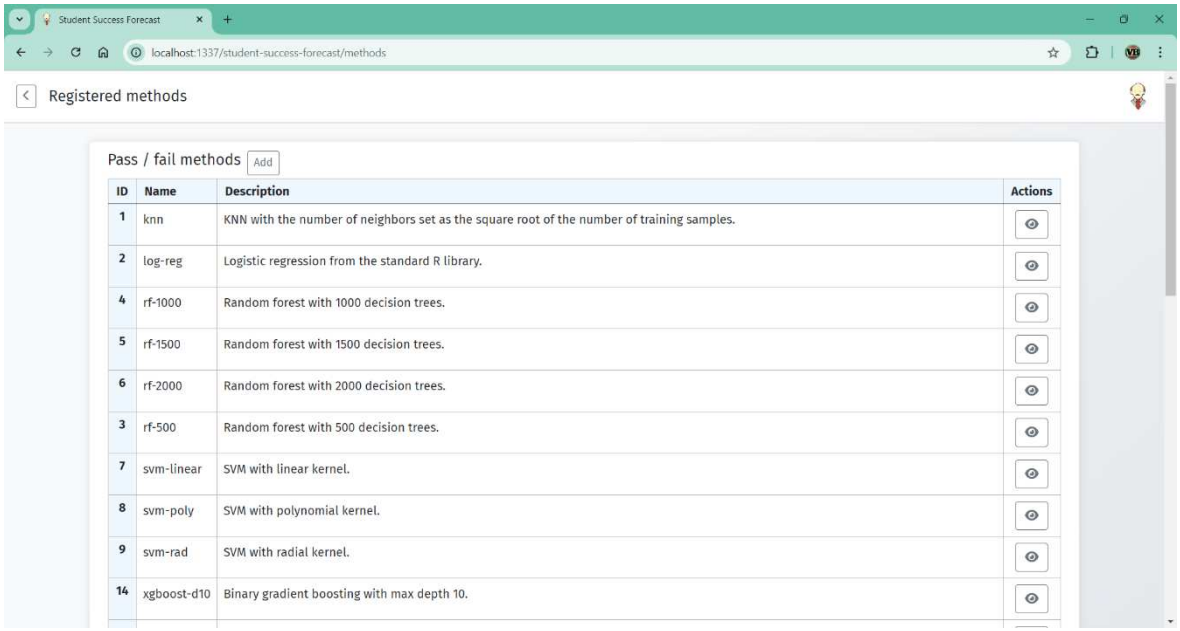
Kako korisnik upravlja sustavom predviđanja uspjeha demonstrira se na primjeru korištenja sustava u praksi. Predviđanja se vrše za kolegij Napredne baze podataka (skraćeno NBP) u akademskoj godini 2023./2024., gledana kao tekuća akademska godina.

Prvo što nastavnik vidi je početna stranica sučelja te odmah dobiva informacije o stanju modula predviđanja uspjeha studenata i je li konfiguracija predviđanja za trenutnu akademsku godinu i kolegij definirana, što u ovom slučaju nije. Navedeno je vidljivo na slici (**Slika 4.1**).

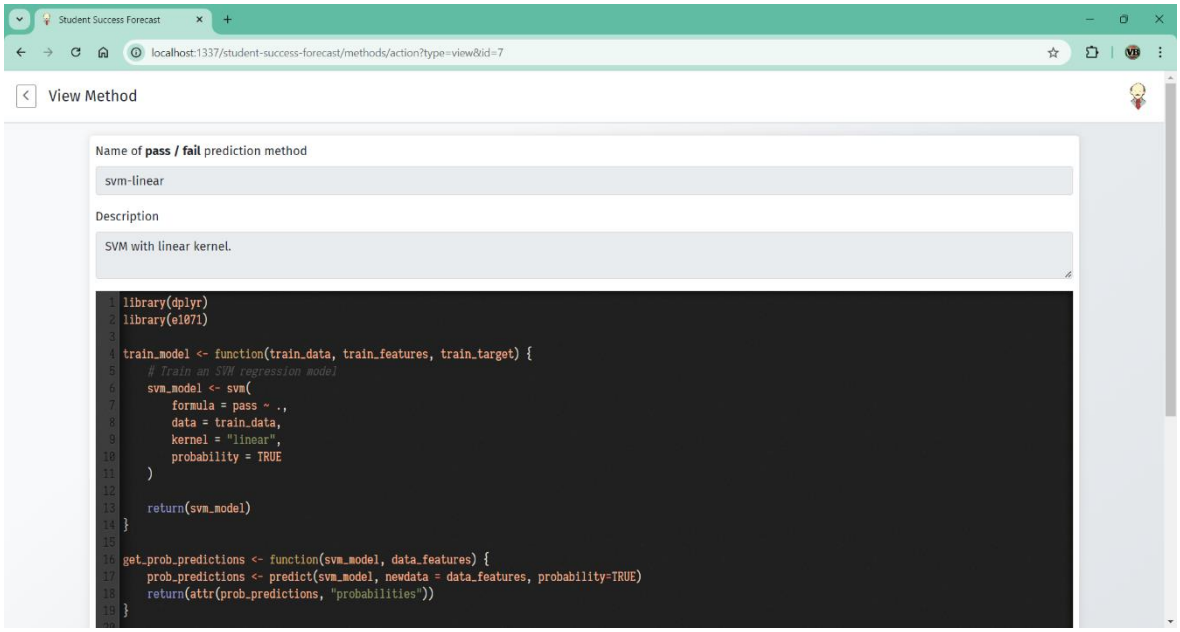


Slika 4.1 Snimka zaslona sučelja - Početna stranica bez postavljene konfiguracije

Prije postavljanja konfiguracije, prikazuje se prozor pregleda svih registriranih metoda u sustavu (**Slika 4.2**) te prozor detaljnijeg pregleda metode linearnog SVM-a (**Slika 4.3**). Tablica prikazuje imena i opise tih metoda, dok je za pregled kôda potrebno odrediti jednu metodu radi štednje resursa. Pregled tog kôda je omogućen putem *R Code Mirror* komponente. Bitno je primijetiti da uređivanje metoda nije dozvoljeno radi moguće ovisnosti tih metoda u drugim konfiguracijama predviđanja.



Slika 4.2 Snimka zaslona sučelja - Tablica spremljenih metoda

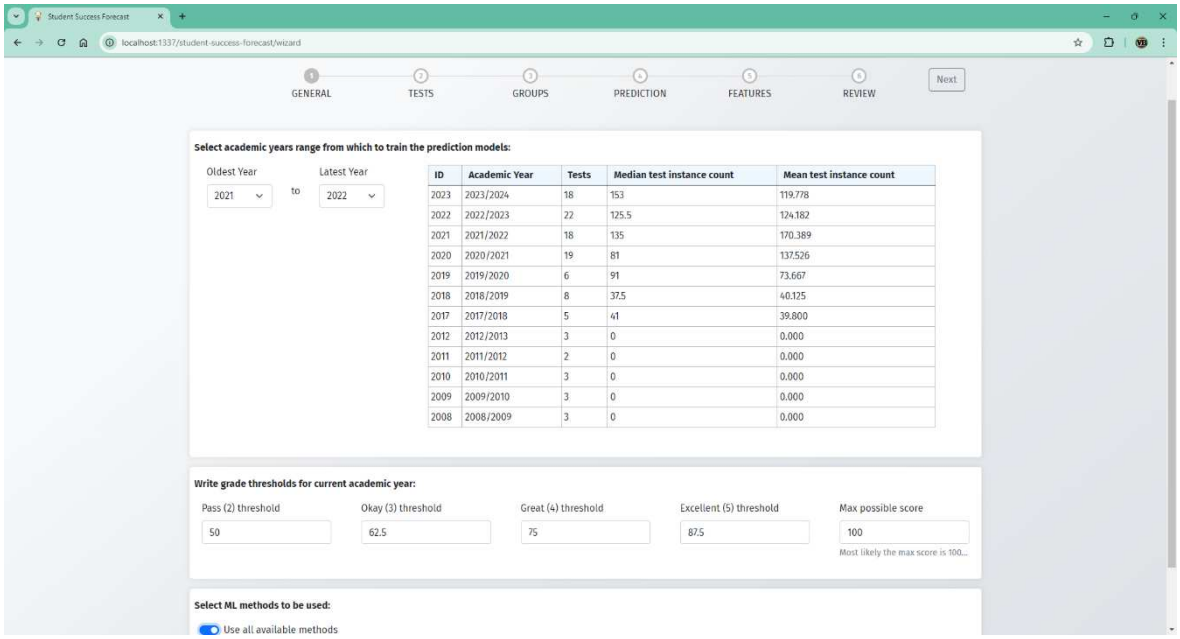


Slika 4.3 Snimka zaslona sučelja - Detalji metode *svm-linear*

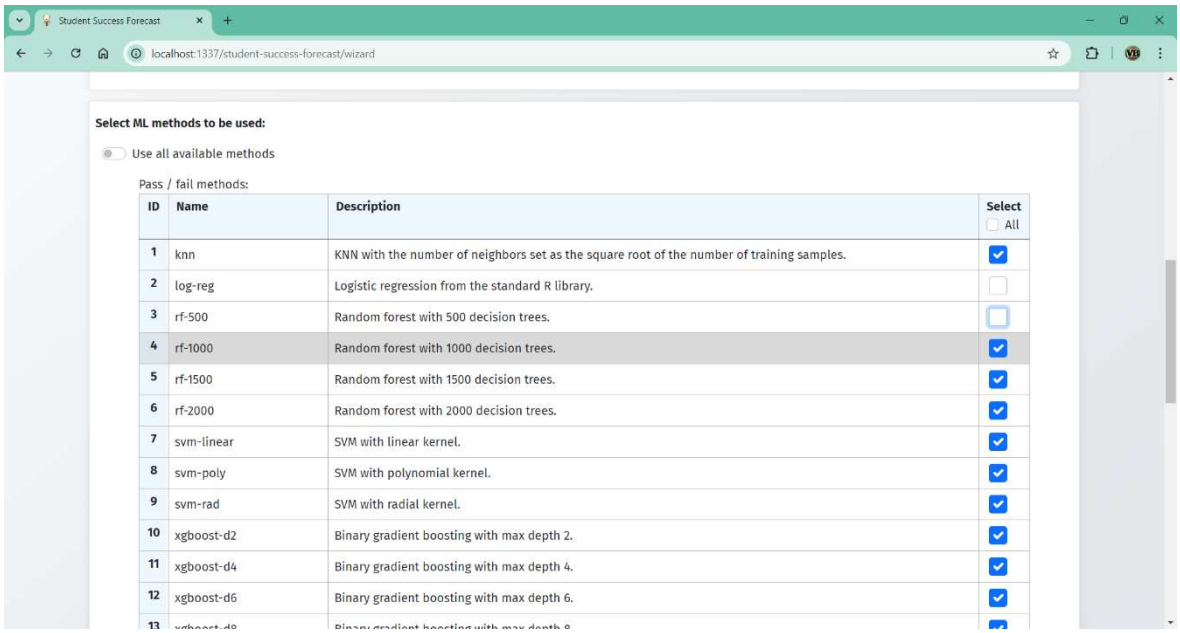
Isti prozor koji prikazuje detalje metode u stanju dodavanja ima otključana polja za unos podataka te korisnik može dodati novu metodu, ako sustav podržava biblioteke tog algoritma. Na primjer, pomoću toga korisnik dodaje novu metodu višeslojnog perceptrona implementiranog bibliotekama Torch i Brulee. Kako bi to napravio za oba tipa predikcije,

korisnik mora promijeniti tip predviđanja metode, što znači da će mu trebati dva posebna zahtjeva.

Nakon dodavanja metoda, konačno se prelazi na konfiguraciju sustava predviđanja za tekuću akademsku godinu. Prvi korak (**Slika 4.4**) daje pregled prošlih akademskih godina kolegija te se određuju podaci kojih akademskih godina će se koristiti za treniranje modela. U ovom slučaju se odabiru akademske godine 2021./2022. i 2022./2023. Dalje se upisuju bodovni pragovi ocjena i za kraj se odabiru koje točno metode će se koristiti u treniranju modela. Predodređeno je korištenje svih raspoloživih metoda i tako će ostati u ovom primjeru, ali korisnik se može odlučiti za precizniji odabir, što otvara tablicu svih metoda (**Slika 4.5**). Jedini uvjet je da svaka vrsta predviđanja (prolaza ili ocjene) sadrži barem jednu metodu.

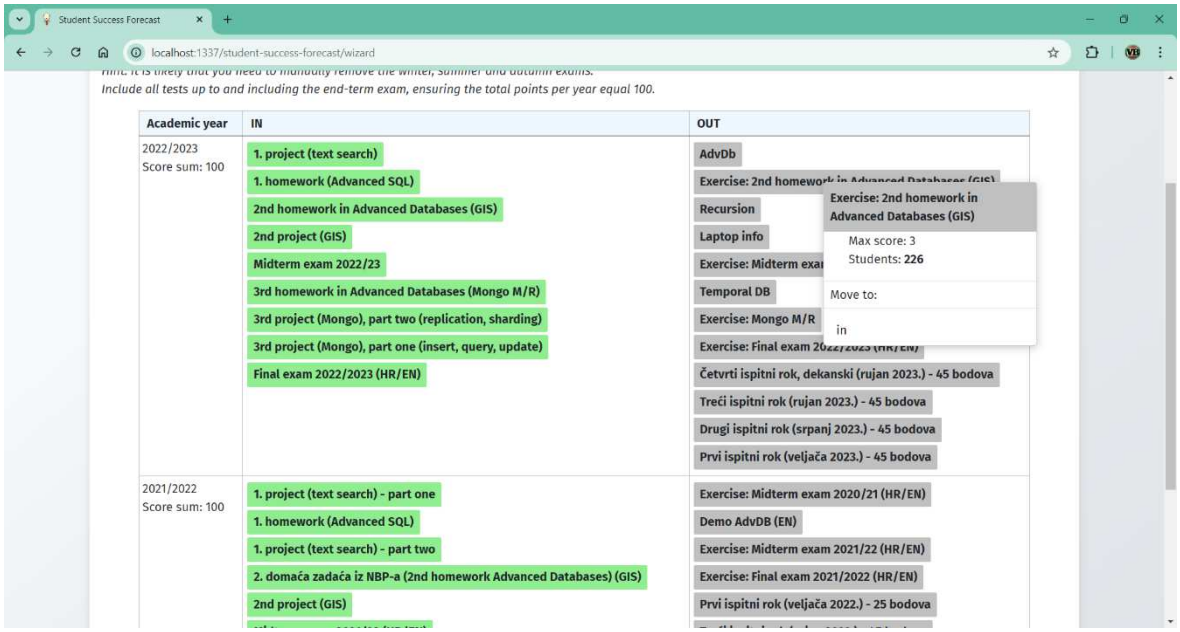


Slika 4.4 Snimka zaslona sučelja - 1. korak konfiguriranja predviđanja



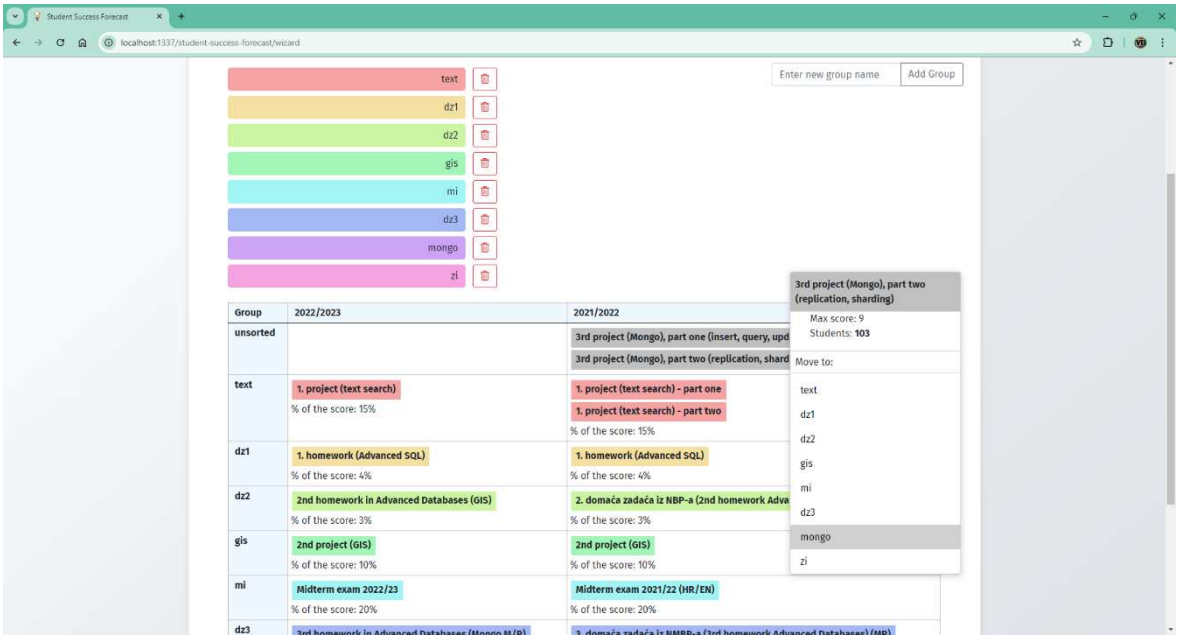
Slika 4.5 Snimka zaslona sučelja - Precizan odabir metoda

Sljedećim korakom (**Slika 4.6**) se definira koji testovi su bitni za davanje predikcija u kontekstu kontinuirane nastave. To raspoređivanje testova je moguće povlačenjem i ispuštanjem oznake testa u prikladna polja na sučelju ili desnim klikom na test, što otvara izbornik s opcijama gdje se može premjestiti. Isti izbornik popratno daje detalje o samom testu poput koliko bodova donosi i koliko studenata ga je pisalo.



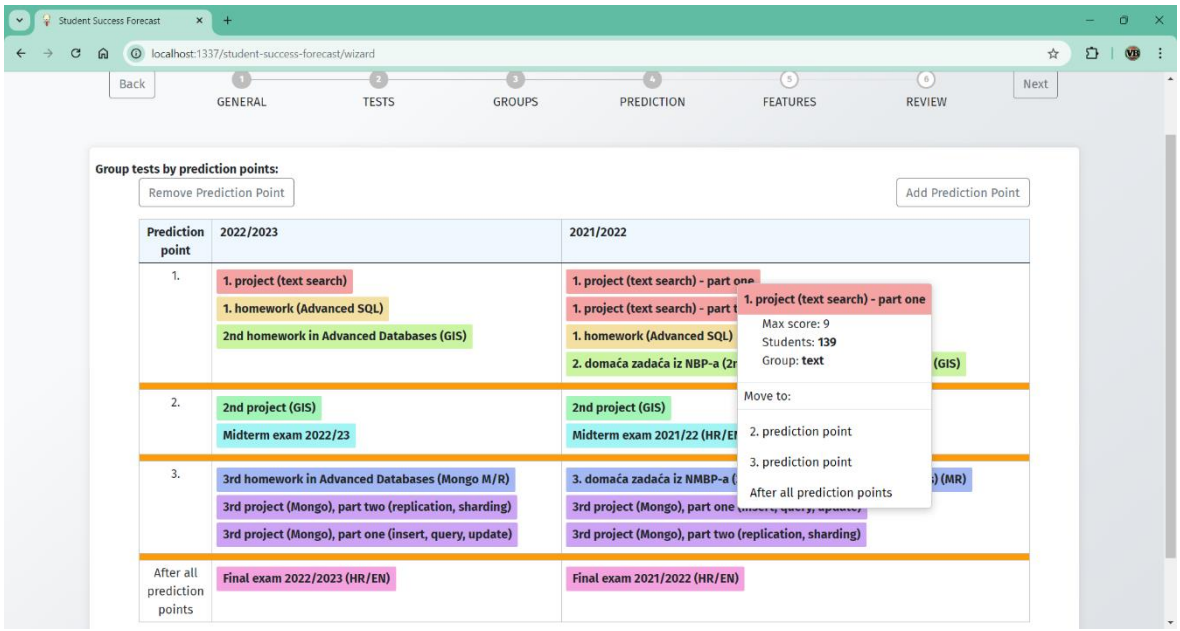
Slika 4.6 Snimka zaslona sučelja - 2. korak konfiguracije predviđanja

Treći korak (**Slika 4.7**) je stvaranje grupacija i grupiranje odabranih testova između njih. Grupacije je poželjno posložiti tako da jedan test pripada jednoj grupaciji u svakoj akademskoj godini, ali to nije nužno potrebno. Načini raspoređivanja su isti kao i za prošli korak te je bitno ne ostaviti negrupirane testove.



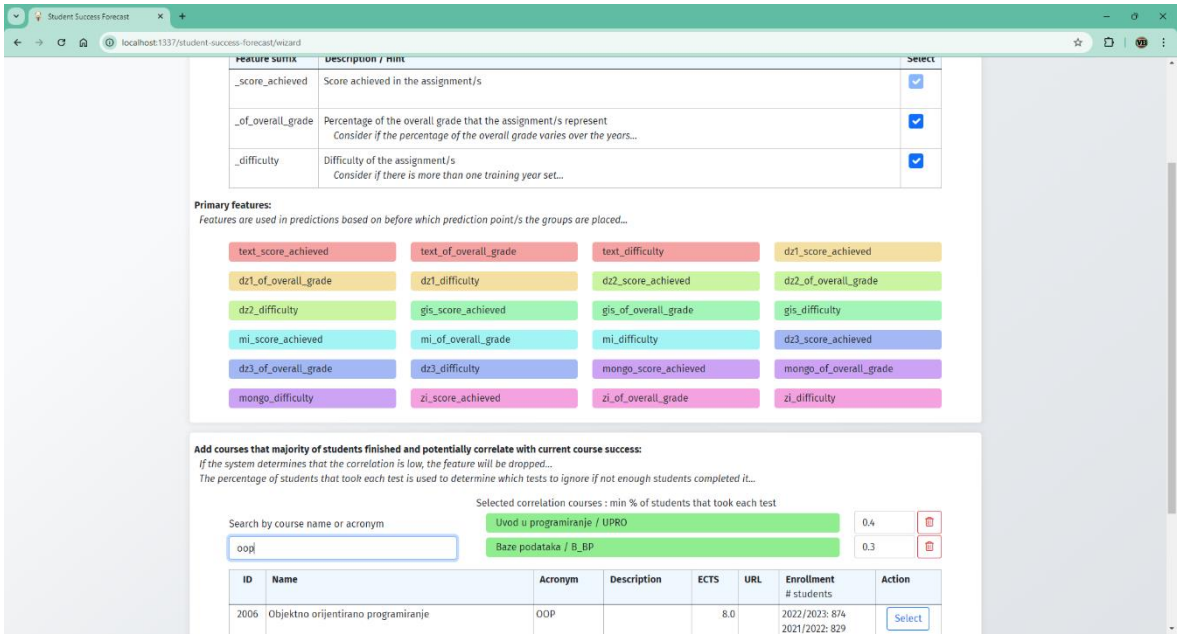
Slika 4.7 Snimka zaslona sučelja - 3. korak konfiguracije predviđanja

U četvrtom koraku (**Slika 4.8**) se određuje broj točaka predikcije i raspoređuju testovi. Ako je jedna grupacija smještena, na primjer u 1. točku predikcije u akademskoj godini 2021./2022., onda test ili testovi iste grupacije moraju biti smješteni u tu točku predikcije i za akademsku godinu 2022./2023. I naravno, raspoređivanje testova funkcionira isto kao i u prošlim koracima. Tipično, završni ispit tog kolegija ostaje nakon svih točaka predikcije i izuzet je u daljnjem postupku.



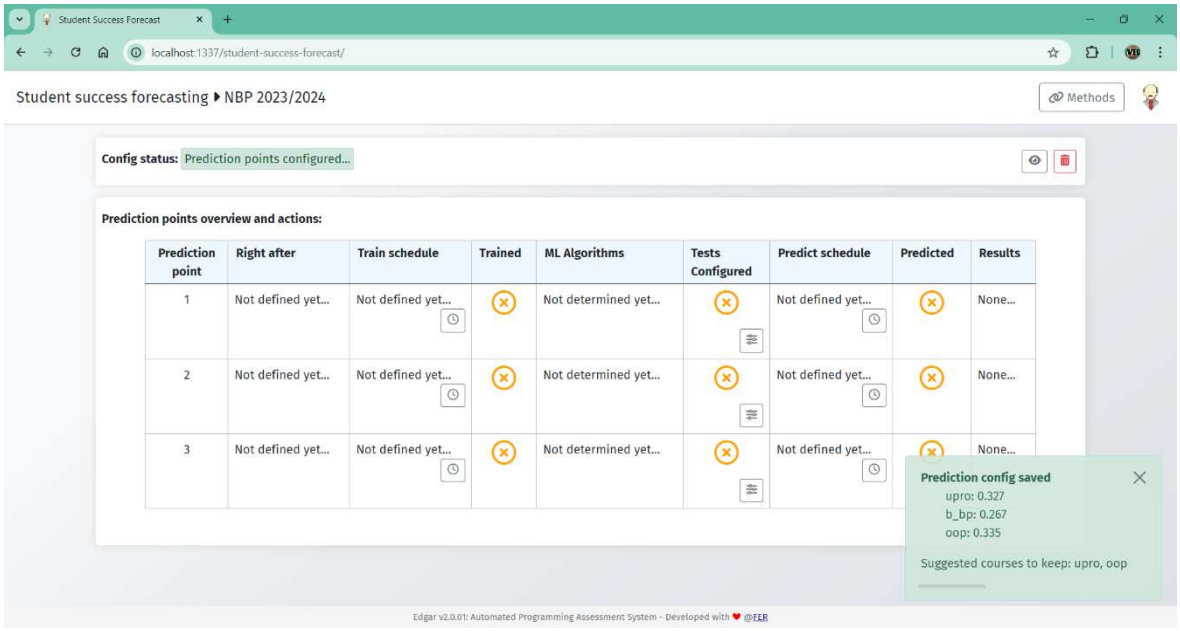
Slika 4.8 Snimka zaslona - 4. korak konfiguracije predviđanja

U predzadnjem koraku (**Slika 4.9**) se određuju korištene značajke i dodaju potencijalno relevantni kolegiji za predviđanje uspjeha trenutnog kolegija. Radi primjera se dodaju kolegiji UPRO, BP i OOP.



Slika 4.9 Snimka zaslona sučelja - 5. korak konfiguracije predviđanja

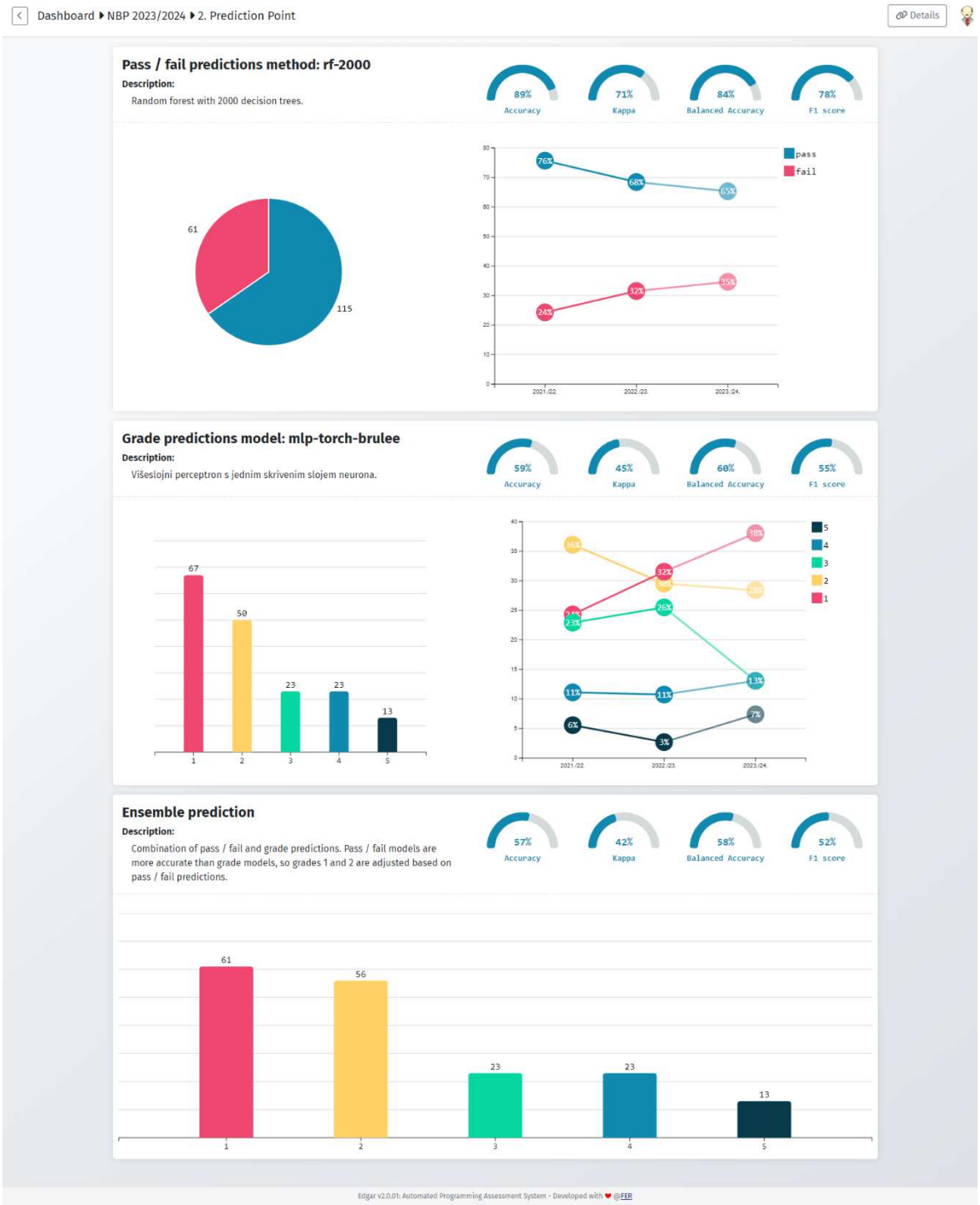
Zadnji korak je pregled svih prije odabranih postavki. Ako je korisnik zadovoljan odabirom, može poslati konfiguraciju sustavu te on vraća korisnika na početnu stranicu uz poruku koji relevantni kolegiji su ostali u značajkama skupa podataka za treniranje. U ovom slučaju kolegij BP nije zadovoljio kriterije korelacije jer je korelacijski koeficijent manji od 0.3 te je izbačen iz konfiguracije, dok su UPRO i OOP zadržani jer su im koeficijenti 0.32 i 0.35. Uz to korisnik može primijetiti tablicu točaka predikcije gdje može upravljati modelima. Trenutni primjer definira 3 točke predikcije, vidljive na slici (**Slika 4.10**).



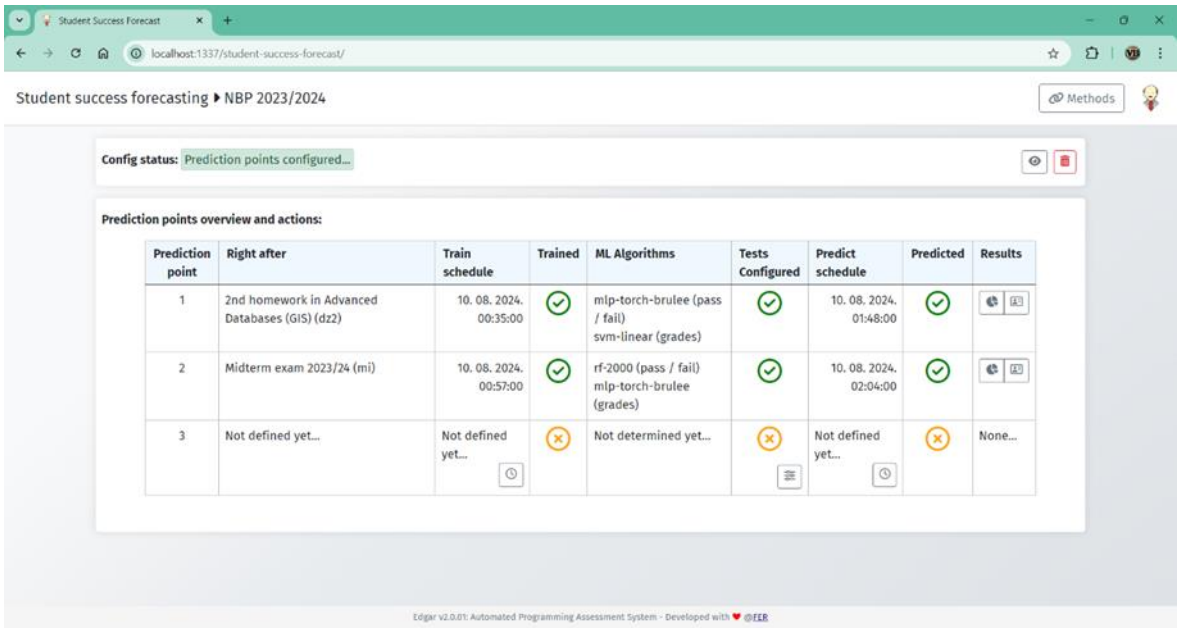
Slika 4.10 Snimka zaslona sučelja - Početna stranica s 3 točke predikcije

Sada nastavnik može zakazati vrijeme izvođenja treniranja modela te nakon toga konfigurirati ispite iz trenutne akademske godine kako bi se pravilno pretvorili u značajke za predviđanje, što se radi u pojednostavljenoj verziji 3. koraka s već definiranim grupacijama, ali sa istim opcijama raspoređivanja testova. Nadalje, može se zakazati vrijeme predviđanja, koje se izvršava ako su modeli predviđanja prolaza i ocjena generirani.

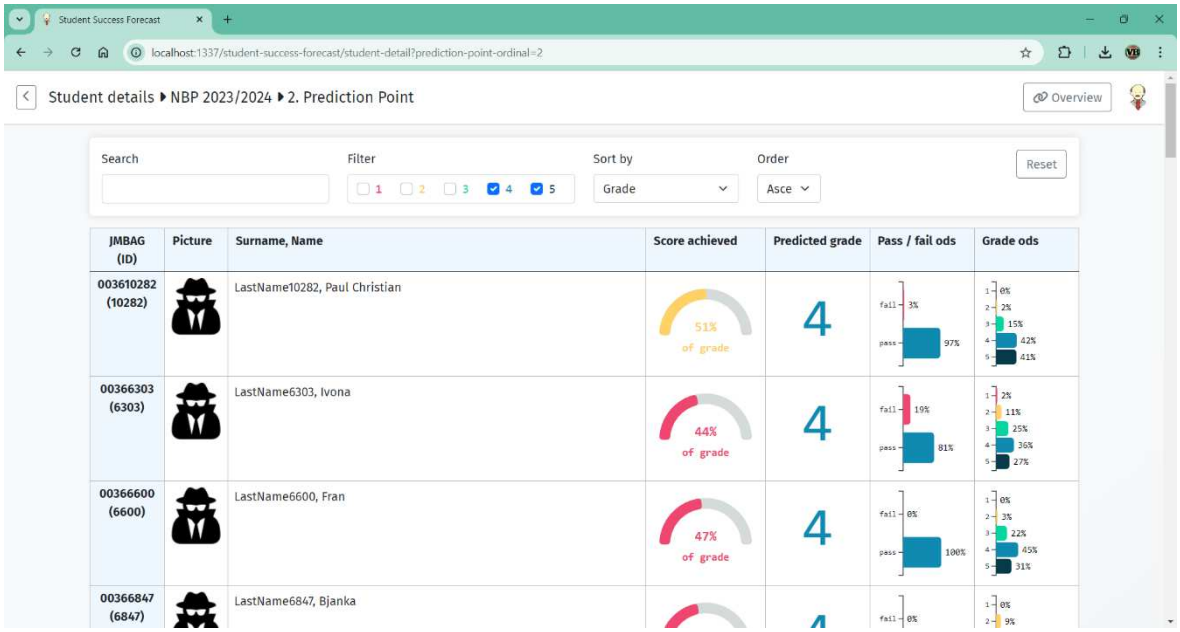
Nakon što su se obradila predviđanja za 2. točku predikcije (**Slika 4.12**), za nju se omogućuje navigacija na pregled predviđenih uspjeha studenata. Grafički pregled uspjeha generacije se može vidjeti na slici (**Slika 4.11**), a detaljan pregled po studentu na slici (**Slika 4.13**).



Slika 4.11 Snimka zaslona sučelja - Pregled uspjeha generacije nakon 2. točke predikcije

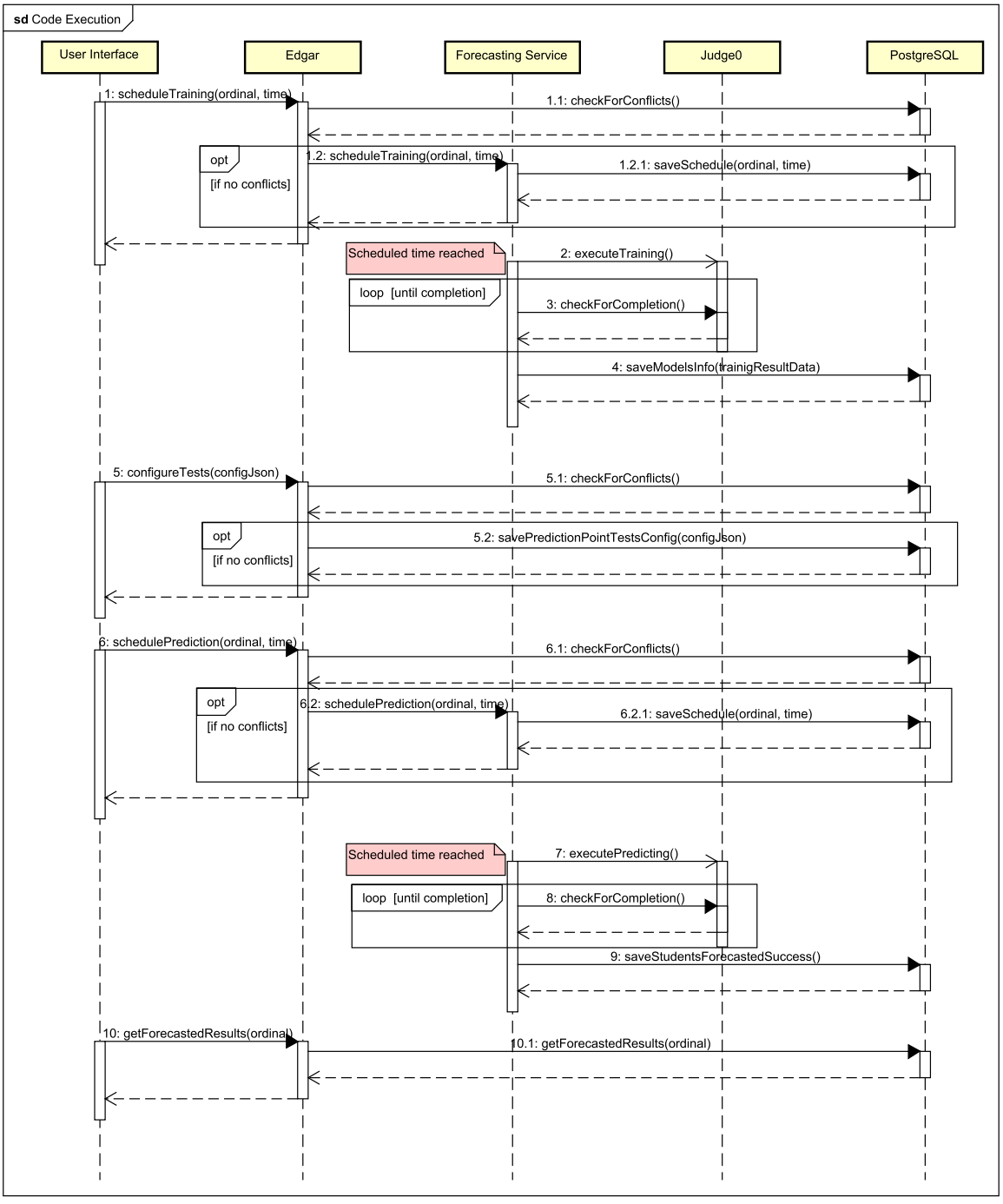


Slika 4.12 Snimka zaslona sučelja - 1. i 2. točka predikcije imaju trenirane modele i zabilježene predviđene uspjehe



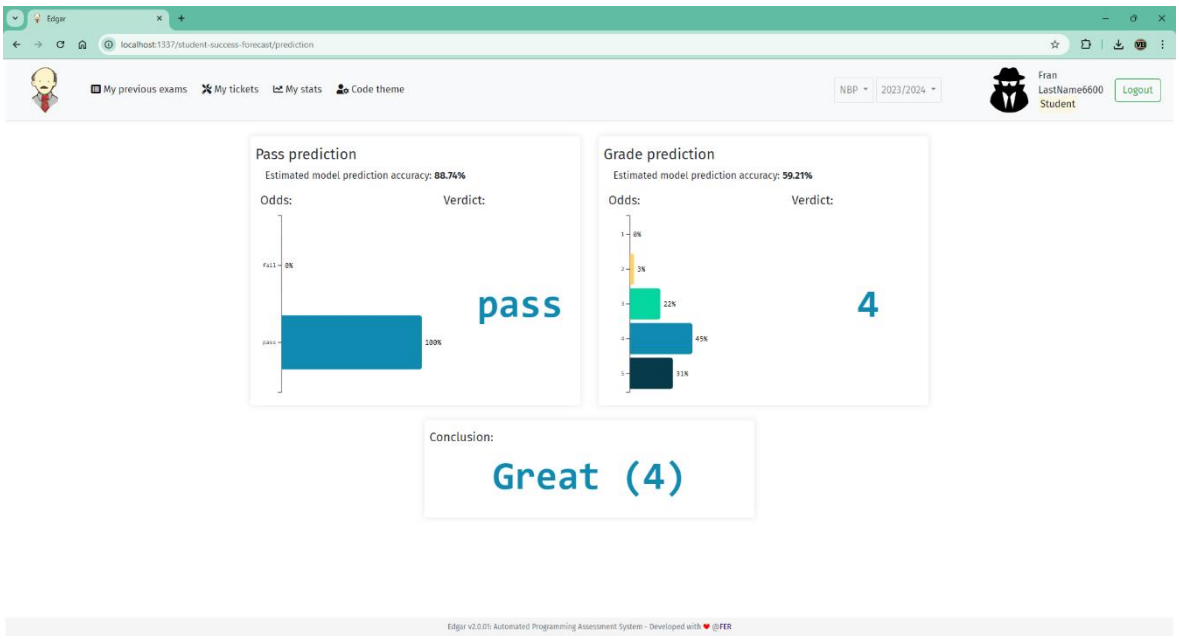
Slika 4.13 Snimka zaslona sučelja - Tablica predviđenih uspjeha po studentu nakon 2. točke predikcije

Općeniti tijek radnji sustava, te kako koji dio sustava upravlja i komunicira na korisnikove zahtjeve, može se vidjeti na dijagramu (**Slika 4.14**), gdje se prikazuju zahtjevi treniranja i predviđanja nad jednom točkom predikcije nastale iz definirane konfiguracije.



Slika 4.14 Sekvencijski dijagram - Zakazivanje poslova i izvršavanje kôda pomoću Judge0 sustava

Za kraj je prikazano kako student vidi vlastiti predviđeni uspjeh nakon 2. točke predikcije, vidljivo na slici zaslona (**Slika 4.15**). Njemu se predstavlja predviđanje pojedinog modela uz šanse mogućih drugih ishoda. Na dnu stranice se nalazi konačno zaključen uspjeh, koji nastaje spajanjem predviđanja oba modela u jedno. To se postiže tako da model predviđanja prolaza korigira predviđenu ocjenu modela predviđanja ocjena, ako se njihova predviđanja ne podudaraju, pošto je on precizniji.



Slika 4.15 Snimka zaslona - Studentova stranica za pregled uspjeha

Zaključak

U ovom radu istražena je uporaba dubinske analize podataka u obrazovanju u nastojanju da se još bolje iskoriste podatci o akademskom uspjehu studenata koji se prikupljaju ponajviše na online platformama za učenje, poput sustava Edgar. Pomoću tih radova se zaključuje da je korištenje podataka prijašnjeg uspjeha studenata dovoljno za izgradnju modela predviđanja uspjeha i nije nužno potrebno koristiti podatke koji izlaze van domene obrazovanja.

Stečeno znanje se koristilo u razvoju postupka unakrsne provjere različitih modela strojnog učenja s različitim hiperparametrima. Metode koje treniraju te modele su standardiziranog formata kako bi unakrsna provjera mogla pravilno usporediti i odabrati najbolji model za traženu svrhu. Korištenje sustava je davalo različite najbolje modele ovisno o kolegiju, što potvrđuje da različite metode mogu bolje raspoznati posebnosti između kolegija, odnosno da ne postoji „jedna najbolja metoda“. Nadalje, uvođenjem vrijednosti težina pojedinih ocjena ovisno o distribuciji i promjenom glavnih metrika za usporedbu predviđanja modela je popravilo generalizaciju tih modela, što je dalo bolje rezultate od prve iteracije algoritma unakrsne provjere.

Oko razvijenog postupka usporedbe je zatim izgrađen samostojeći modul predviđanja uspjeha, koji omogućuje Edgaru dubinsku analizu podataka. Potreba za lokalnom strategijom izvođenja kôda se pojavila nakon što se ispostavilo da Judge0 ne može podržati unakrsnu usporedbu modela treniranih na velikim skupovima podataka, točnije više od 1000 zapisa. Nadalje, kako bi korisnik mogao u potpunosti iskoristiti podesivosti modula predviđanja i pregled samih predviđenih uspjeha, projektirano je robusno web sučelje pomoću Angular radnog okvira. Sučelje omogućuje uvid u potrebne informacije za konfiguriranje točaka predikcije pojedinog kolegija u tekućoj akademskoj godini te omogućuje raspoređivanje poslova izvršavanja kôda.

Smjernice za daljnji razvoj su omogućiti korisniku ad hoc dodavanje novih biblioteka u modul predikcije, što bi doprinijelo većem broju algoritama strojnog učenja kojima bi sustav raspolagao. Ujedno se predlaže proširiti sustav da podržava korištenje dodatnih značajki koje korisnik sam pripremi i tako proširiti domenu informacija za predviđanje uspjeha studenata.

Literatura

[1] Armstrong, P. (2010). *Bloom's Taxonomy*, Vanderbilt University Center for Teaching, (2010). Poveznica: https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy/; pristupljeno 26. srpnja 2024.

[2] Awan, A. A. *Top 10 Data Science Tools To Use in 2024*, Datacamp (2023, studeni). Poveznica: https://www.datacamp.com/blog/top-data-science-tools; pristupljeno 9. kolovoza 2024.

[3] Bakhshinategh, B., Zaiane, O. R., ElAtia, S., Ipperciel, D. *Educational data mining applications and tasks: A survey of the last 10 years*, Education and Information Technologies, 23 (2018), str. 537-553.

[4] Benjak, P. *Prototip integracije sustava za strojno učenje u sustav Edgar*. Preddiplomski rad. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2022.

[5] Bentéjac, C., Csörgő, A., Martínez-Muñoz, G. *A comparative analysis of gradient boosting algorithms*, Artificial Intelligence Review, 54 (2021), str. 1937-1967.

[6] Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P. *SMOTE: synthetic minority over-sampling technique*. Journal of artificial intelligence research 16 (2002), str. 321-357.

[7] *Curse of dimensionality*, Wikipedia. Poveznica: https://en.wikipedia.org/wiki/Curse_of_dimensionality; pristupljeno 7. svibnja 2023.

[8] Dašić, D. *Postupci ansambla stabala odluke*. Završni rad. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2017.

[9] Decision Trees, Poveznica: https://scikit-learn.org/stable/modules/tree.html ; Pristupljeno: 20. travnja 2023.

[10] Došilović, H. Z. *Judge0 CE - API Docs*, Judge0 (2024, travanj). Poveznica: https://ce.judge0.com/; pristupljeno 20. lipnja 2024.

[11] Došilović, H. Z., Mekterović, I. *Robust and scalable online code execution system*, In 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO) (2020), str. 1627-1632. IEEE.

[12] Dutt, A., Ismail, M. A., Herawan, T. *A systematic review on educational data mining*. IEEE Access, 5 (2017), str. 15991-16005.

[13] Feng, G., Fan, M., Chen, Y. *Analysis and prediction of students' academic performance based on educational data mining*. IEEE Access, 10 (2022), str. 19558-19571.

[14] Gupta, P. *Decision Trees in Machine Learning*, (2017, svibanj). Poveznica: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052; pristupljeno 20. travnja 2023.

[15] Hastie, T., Tibshirani, R., Friedman, J. H., Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2, pp. 1-758). New York: springer

[16] Hu, Q., Rangwala, H. *Towards Fair Educational Data Mining: A Case Study on Detecting At-Risk Students*. International Educational Data Mining Society, (2020).

[17] IBM. *What is machine learning (ML)?*, Poveznica: https://www.ibm.com/topics/machine-learning; pristupljeno 20. travnja 2023.

[18] Jaiswal, S. *Multilayer Perceptrons in Machine Learning: A Comprehensive Guide*, Datacamp, (2024, ožujak). Poveznica: https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning; pristupljeno 28. ožujka 2024.

[19] *K-Nearest Neighbor(KNN) Algorithm for Machine Learning*, Javatpoint, (2021). Poveznica: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning; pristupljeno 21. travnja 2023.

[20] Lane, H. (https://stats.stackexchange.com/users/15974/hobs). *How to choose the number of hidden layers and nodes in a feedforward neural network?*, (2021, svibanj). Poveznica: https://stats.stackexchange.com/q/136542; pristupljeno 24. listopada 2023.

[21] *Logistic Regression in Machine Learning*, GeeksforGeeks, (2023, svibanj). Poveznica: https://www.geeksforgeeks.org/understanding-logistic-regression/; pristupljeno 21. travnja 2023.

[22] *Logistic Regression in Machine Learning*, Javatpoint, (2021). Poveznica: https://www.javatpoint.com/logistic-regression-in-machine-learning; pristupljeno 20. travnja 2023.

[23] Mekterović, I., Brkić, Lj. *Setting up Automated Programming Assessment System for Higher Education Database Course*. International Journal of Education and Learning Systems, 2 (2017).

[24] Olugbenga, M. *Balanced Accuracy: When Should You Use It?*, Neptune AI, (2023, kolovoz). Poveznica: https://neptune.ai/blog/balanced-accuracy; pristupljeno 1. svibnja 2024.

[25] Parsno, D. E. *Kappa Statistic¸* CSC 558 - Data Mining and Predictive Analytics, (2019, listopad). Poveznica: https://faculty.kutztown.edu/parson/fall2019/Fall2019Kappa.html; pristupljeno 1. svibnja 2024.

[26] Popescu, M. C., Balas, V. E., Perescu-Popescu, L., Mastorakis, N. *Multilayer perceptron and neural networks*, WSEAS Transactions on Circuits and Systems, 8,7 (2009). str. 579-588.

[27] Romero, C., Ventura, S. *Educational data mining and learning analytics: An updated survey*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10,3 (2020), e1355.

[28] Romero, C., Ventura, S. *Educational data mining: a review of the state of the art*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (applications and reviews), 40,6 (2010), str. 601-618.

[29] Saa, A. A. *Educational data mining & students' performance prediction*. International Journal of Advanced Computer Science and Applications, 7,5 (2016).

[30] Singh, K. *How to Improve Class Imbalance using Class Weights in Machine Learning?*, Analytics Vidhya, (2020, listopad). Poveznica: https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/; pristupljeno 15. travnja 2024.

[31] *Support Vector Machine Algorithm*, Javatpoint, (2021). Poveznica: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm; pristupljeno 20. travnja 2023.

[32] TensorFlow online dokumentacija. *Classification on imbalanced data*, TensorFlow, (2024, ožujak). Poveznica: https://www.tensorflow.org/tutorials/structured_data/imbalanced_data; pristupljeno 15. travnja 2024.

[33] Velez, D. R., White, B. C., Motsinger, A. A., Bush, W. S., Ritchie, M. D., Williams, S. M., Moore, J. H. *A balanced accuracy function for epistasis modeling in imbalanced datasets using multifactor dimensionality reduction*, Genetic Epidemiology: the Official Publication of the International Genetic Epidemiology Society, 31,4 (2007), str. 306-315.

[34] Viera, A. J., Garrett, J. M. *Understanding interobserver agreement: the kappa statistic*. Fam med, 37,5 (2005), str. 360-363.

[35] Xiao, W., Ji, P., Hu, J. *A survey on educational data mining methods used for predicting students' performance*. Engineering Reports, 4,5 (2022), e12482.

[36] Yağcı, M. *Educational data mining: prediction of students' academic performance using machine learning algorithms*. Smart Learning Environments, 9,1 (2022), str. 11.

Sažetak

Predviđanje uspjeha studenata u sustavu za automatsko ocjenjivanje programskog kôda Edgar

Rastom značajnosti online platformi za učenje omogućuje veliki broj upisanih studenata na pojedinim kolegijima, ali zauzvrat nastavnici ne mogu pravovremeno prepoznati studente koji imaju poteškoće s gradivom. Kako se sustav za automatizirano ocjenjivanje programskog kôda Edgar suočava s istim problemom, ovaj rad ulazi u istraživačke radove u području dubinske analize podataka u obrazovanju te se prema njima razvija sustav usporedbe različitih algoritama strojnog učenja za predviđanje uspjeha studenata. Pomoću njega se zaključuje da se najbolja metoda za predviđanje uspjeha razlikuje za pojedini kolegij. Oko tog sustava se gradi zaseban podesivi modul predviđanja uspjeha koji može koristiti bilo koji kolegij koji ujedno koristi Edgar. Modul raspolaže s Judge0 sustavom za izvršavanje treniranja modela, ali ako kolegij sadrži veliki broj upisanih studenata primoran je izvršavati svoj programski kôd lokalno. Konačno se daje pogled u web sučelje koje olakšava nastavnicima upravljanje projektiranim modulom te nudi vizualizacije za prepoznavanje i analizu kritičnih studenata.

Ključne riječi: dubinska analiza podataka u obrazovanju, strojno učenje, R programski jezik, procesiranje podataka, unakrsna provjera valjanosti, SQL, Angular, D3.js, vizualizacija podataka, Edgar, node.js, express.js, Judge0, Docker

Summary

Predicting student success in Edgar automated programming assessment system

The growing importance of online learning platforms allows for a lot off enrolled students in certain courses, but in turn, teachers cannot timely recognize students who have difficulties with the learning material. As the automated programming assessment system Edgar faces the same problem, this dissertation delves into research in the field of educational data mining and creates a system for cross validation of different machine learning models for predicting student success. This system concludes that the best method for predicting success varies from course to course. A standalone adjustable success prediction module is built around this system, which can be used by any course that also uses Edgar. The module uses the Judge0 system for executing the training of the model, but if the course contains a large number of enrolled students, it is forced to execute its code locally. Finally, a look at the web interface is provided, which facilitates teachers in managing the designed module and offers visualizations for identifying and analyzing at-risk students.

Keywords: educational data mining, machine learning, R programming language, data processing, cross validation, SQL, Angular, D3.js, data visualization, Edgar, node.js, express.js, Judge0, Docker

Skraćenice

APAS	*Automated Programming Assessment System*	Automatizirani sustav za evaluaciju programskog kôda
API	*Application Programming Interface*	Programsko sučelje aplikacije
BP		Baze podataka
EDM	*Educational Data Mining*	Dubinska analiza podatak u obrazovanju
KNN	*K-Nearest Neighbors*	K najbližih susjeda
LMS	*Learning Management System*	Sustav za upravljanje obrazovanjem
MLP	*Multilayer Perceptron*	Višeslojni perceptron
MOOC	*Massive Open Online Course*	Javno pristupan online tečaj
NBP		Napredne baze podatak
PI		Poslovna inteligencija
RF	*Random Forest*	Algoritam nasumične šume
SIS	*Students Information System*	Studentski informacijski sustav
SVM	*Support Vector Machine*	Stroj potpornih vektora
UPRO		Uvod u programiranje
WEB2		Napredni razvoj programske potpore za web

Privitak

Izvorni kod

Poveznica na repozitorij modula predviđanja uspjeha studenata: https://gitlab.com/vbedekovic16/student-success-forecast-service

Poveznica na granu Edgar repozitorija s proširenim funkcionalnostima za sustav predviđanja uspjeha studenata i s korisničkim sučeljem: https://gitlab.com/edgar-group/edgar/-/tree/dev-student-forecast

Detalji mjerenja iz poglavlja 2.7

Poveznica na dokument: https://gitlab.com/vbedekovic16/student-success-forecast-service/-/blob/main/dissertation/measurements/Kona%C4%8Dna%20mjerenja.pdf

R skripta za treniranje, usporedbu i odabir najboljih modela za predviđanje pada ili prolaza i ocjena

```
library(dplyr)
library(jsonlite)

cat("#################### Start of model training ####################\n")

# Load parameters from JSON file
params <- fromJSON("parameters.json")

# SCRIPT INPUTS from JSON
number_of_folds <- params$number_of_folds
course_id <- params$course_id
prediction_point <- params$prediction_point
groups <- params$groups
suffixes <- params$suffixes
additional_features <- params$additional_features
# Ensure params$additional_features is character type, even when empty
if (is.null(additional_features) || length(additional_features) == 0) {
  additional_features <- character(0)  # Set to an empty character vector
} else {
  # Ensure it's treated as a character vector in case it's not
  additional_features <- as.character(additional_features)
}

pass_method_folder_path <- params$pass_methods_folder
grade_method_folder_path <- params$grade_methods_folder
```

```r
data_filename <- params$data_filename

# Load data
data <- read.csv(data_filename)

cat("Start of pass model training\n")
selected_metric <- "kappa"

features <- c("pass", additional_features, as.vector(outer(groups, suffixes, paste0)))
cat("Features used for pass model training:\n")
print(features)

method_folder_path <- pass_method_folder_path
print(method_folder_path)

methods <- sapply(list.files(method_folder_path), tools::file_path_sans_ext)
methods <- as.vector(methods)
cat("Used machine learning methods and variants:\n")
print(methods)


select_features_data <- data[features]
select_features_data <- select_features_data %>% dplyr::mutate(across(-pass, as.double))
select_features_data$pass <- as.factor(select_features_data$pass)

number_of_samples <- nrow(select_features_data)
cat("Number of student samples:", number_of_samples, "\n")

pass_distribution_table <- table(select_features_data$pass)
cat("Pass distribution:\n")
print(pass_distribution_table)

calculate_pass_model_metrics <- function(predictions, test_target) {
    confusion_matrix <- caret::confusionMatrix(
        data = predictions,
        reference = test_target
    )

    accuracy <- confusion_matrix[["overall"]][["Accuracy"]]
    kappa <- confusion_matrix[["overall"]][["Kappa"]]
    balanced_accuracy <- confusion_matrix[["byClass"]][["Balanced Accuracy"]]
    f1_score <- confusion_matrix[["byClass"]][["F1"]]

    return(list(
        accuracy = accuracy,
        kappa = kappa,
        balanced_accuracy = balanced_accuracy,
        f1_score = f1_score
```

```
    ))
}
metric_names <- c("accuracy", "kappa", "balanced_accuracy", "f1_score")


# Shuffle the data
shuffled_data <- select_features_data[sample(1:nrow(select_features_data)), ]

folds <- caret::createFolds(shuffled_data$pass, k = number_of_folds)

# Init cross validation data frames for every metric
metrics_data_frame_list <- list()
for (metric in metric_names) {
    metrics_data_frame_list[[metric]] <- data.frame(matrix(ncol = length(methods), nrow = 0))
    colnames(metrics_data_frame_list[[metric]]) <- methods
}

for (i in 1:number_of_folds) {
    test_indices <- folds[[i]]
    test <- shuffled_data %>% dplyr::slice(test_indices)

    train_indices <- unlist(folds[-i])
    train <- shuffled_data %>% dplyr::slice(train_indices)

    cat("Fold iteration:", i, "/", number_of_folds, "\n")
    cat("Size of train sample:", nrow(train), "\n")
    cat("Size of test sample:", nrow(test), "\n")

    train_features <- train %>% dplyr::select(-pass)
    test_features <- test %>% dplyr::select(-pass)

    train_target <- train$pass
    test_target <- test$pass

    metrics_list <- lapply(methods, function(method) {
        # print(paste("pass-classification/methods/", method, ".R", sep = ""))
        # load algorithm with overidden train_model and validate_model functions
        source(paste(method_folder_path, "/", method, ".R", sep = ""))
        model <- train_model(train, train_features, train_target)

        predictions <- get_predictions(model, test_features)
        predictions <- as.factor(predictions)

        metrics <- calculate_pass_model_metrics(predictions, test_target)
        return(metrics)
    })

    for (metric in metric_names) {
        new_row <- as.data.frame(t(sapply(metrics_list, function(x) x[[metric]])))
```

```
        colnames(new_row) <- methods
        metrics_data_frame_list[[metric]] <- bind_rows(metrics_data_frame_list[[metric]], new_row)
    }
}

cross_validation_df <- metrics_data_frame_list[[selected_metric]]

best_method <- methods[0]
best_metric_mean <- 0
best_metric_sd <- 1
for (method in methods) {
    curr_mean <- mean(cross_validation_df[[method]])
    curr_sd <- sd(cross_validation_df[[method]])
    if (best_metric_mean < curr_mean) {
        best_method <- method
        best_metric_mean <- curr_mean
        best_metric_sd <- curr_sd
    }

    cat(
        "Method:", method, "\n",
        "\tMean:", curr_mean, "\n",
        "\tSD:", curr_sd, "\n",
        "\n"
    )
}

best_method_metrics <- sapply(metric_names, function(metric) {
    list(
        mean = mean(metrics_data_frame_list[[metric]][[best_method]])[1],
        sd = sd(metrics_data_frame_list[[metric]][[best_method]])[1]
    )
}, simplify = FALSE)

# Convert list to JSON
json <- toJSON(best_method_metrics, pretty = TRUE, auto_unbox = TRUE)

write(json, file = "pass-metrics.json")

cat(
    "Best pass model for", "COURSE", "is", best_method,
    "with a", selected_metric, "mean of", best_metric_mean,
    "and sd of", best_metric_sd,
    "\n"
)

# Create and save best model trained with all the data we have
all_data <- shuffled_data
all_data_features <- all_data %>% dplyr::select(-pass)
```

```r
all_data_target <- all_data$pass

source(paste(method_folder_path, "/", best_method, ".R", sep = ""))
model <- train_model(all_data, all_data_features, all_data_target)
print(best_method)
best_pass_method <- best_method
print(model)

pass_model_file <- save_model(model, "best_pass_model")

cat("################################################################\n")

cat("Start of grade model training\n")
selected_metric <- "balanced_accuracy"

features <- c("grade", additional_features, as.vector(outer(groups, suffixes, paste0)))

cat("Features used for grade model training:\n")
print(features)

method_folder_path <- grade_method_folder_path
print(method_folder_path)

methods <- sapply(list.files(method_folder_path), tools::file_path_sans_ext)
methods <- as.vector(methods)
cat("Used machine learning methods and variants:\n")
print(methods)

#data_filename <- paste(course_id, "-p", prediction_point, "-student-data.csv", sep = "")
#data <- read.csv(data_filename)


select_features_data <- data[features]
select_features_data <- select_features_data %>% dplyr::mutate(across(-grade, as.double))
select_features_data$grade <- as.factor(select_features_data$grade)

number_of_samples <- nrow(select_features_data)
cat("Number of student samples:", number_of_samples, "\n")

grade_distribution_table <- table(select_features_data$grade)
cat("Grade distribution:\n")
print(grade_distribution_table)

calculate_class_weight_for_balance <- function(n_samples, n_classes, n_class_samples) {
    return(n_samples / (n_classes * n_class_samples))
}

calculate_grade_model_metrics <- function(predictions, test_target) {
    confusion_matrix <- caret::confusionMatrix(
```

```
        data = predictions,
        reference = test_target
    )

    accuracy <- confusion_matrix[["overall"]][["Accuracy"]]

    kappa <- confusion_matrix[["overall"]][["Kappa"]]

    precision_vector <- confusion_matrix[["byClass"]][, "Pos Pred Value"]
    precision_vector[is.na(precision_vector)] <- 0
    precision <- mean(precision_vector)

    recall_vector <- confusion_matrix[["byClass"]][, "Sensitivity"]
    recall_vector[is.na(recall_vector)] <- 0
    balanced_accuracy <- mean(recall_vector)

    f1_scores <- 2 * (precision_vector * recall_vector) / (precision_vector + recall_vector)
    f1_scores[is.na(f1_scores)] <- 0

    macro_f1_score <- mean(f1_scores)

    return(list(
        accuracy = accuracy,
        kappa = kappa,
        balanced_accuracy = balanced_accuracy,
        f1_score = macro_f1_score
    ))
}
metric_names <- c("accuracy", "kappa", "balanced_accuracy", "f1_score")

# calculate class weights vector
class_weights <- c(
    "1" = max(1, calculate_class_weight_for_balance(number_of_samples, 5,
grade_distribution_table[[1]])),
    "2" = max(1, calculate_class_weight_for_balance(number_of_samples, 5,
grade_distribution_table[[2]])),
    "3" = max(1, calculate_class_weight_for_balance(number_of_samples, 5,
grade_distribution_table[[3]])),
    "4" = max(1, calculate_class_weight_for_balance(number_of_samples, 5,
grade_distribution_table[[4]])),
    "5" = max(1, calculate_class_weight_for_balance(number_of_samples, 5,
grade_distribution_table[[5]]))
)
cat("Class (grade) weights:\n")
print(class_weights)

# Shuffle the data
shuffled_data <- select_features_data[sample(1:nrow(select_features_data)), ]
```

```
folds <- caret::createFolds(shuffled_data$grade, k = number_of_folds)

# Init cross validation data frames for every metric
metrics_data_frame_list <- list()
for (metric in metric_names) {
    metrics_data_frame_list[[metric]] <- data.frame(matrix(ncol = length(methods), nrow = 0))
    colnames(metrics_data_frame_list[[metric]]) <- methods
}

for (i in 1:number_of_folds) {
    test_indices <- folds[[i]]
    test <- shuffled_data %>% dplyr::slice(test_indices)

    train_indices <- unlist(folds[-i])
    train <- shuffled_data %>% dplyr::slice(train_indices)

    cat("Fold iteration:", i, "/", number_of_folds, "\n")
    cat("Size of train sample:", nrow(train), "\n")
    cat("Size of test sample:", nrow(test), "\n")

    train_features <- train %>% dplyr::select(-grade)
    test_features <- test %>% dplyr::select(-grade)

    train_target <- train$grade
    test_target <- test$grade

    metrics_list <- lapply(methods, function(method) {
        # load algorithm with overidden train_model and validate_model functions
        source(paste(method_folder_path, "/", method, ".R", sep = ""))
        model <- train_model(train, train_features, train_target, class_weights)

        predictions <- get_predictions(model, test_features)
        predictions <- as.factor(predictions)

        metrics <- calculate_grade_model_metrics(predictions, test_target)
        return(metrics)
    })

    for (metric in metric_names) {
        new_row <- as.data.frame(t(sapply(metrics_list, function(x) x[[metric]])))
        colnames(new_row) <- methods
        metrics_data_frame_list[[metric]] <- bind_rows(metrics_data_frame_list[[metric]], new_row)
    }
}

cross_validation_df <- metrics_data_frame_list[[selected_metric]]

best_method <- methods[0]
best_metric_mean <- 0
```

```r
best_metric_sd <- 1
for (method in methods) {
    curr_mean <- mean(cross_validation_df[[method]])
    curr_sd <- sd(cross_validation_df[[method]])
    if (best_metric_mean < curr_mean) {
        best_method <- method
        best_metric_mean <- curr_mean
        best_metric_sd <- curr_sd
    }

    cat(
        "Method:", method, "\n",
        "\tMean:", curr_mean, "\n",
        "\tSD:", curr_sd, "\n",
        "\n"
    )
}

best_method_metrics <- sapply(metric_names, function(metric) {
    list(
        mean = mean(metrics_data_frame_list[[metric]][[best_method]])[1],
        sd = sd(metrics_data_frame_list[[metric]][[best_method]])[1]
    )
}, simplify = FALSE)

# Convert list to JSON
json <- toJSON(best_method_metrics, pretty = TRUE, auto_unbox = TRUE)

write(json, file = "grade-metrics.json")

cat(
    "Best grade model for", "COURSE", "is", best_method,
    "with a", selected_metric, "mean of", best_metric_mean,
    "and sd of", best_metric_sd,
    "\n"
)

# Create and save best model trained with all the data we have
all_data <- shuffled_data
all_data_features <- all_data %>% dplyr::select(-grade)
all_data_target <- all_data$grade

source(paste(method_folder_path, "/", best_method, ".R", sep = ""))
model <- train_model(all_data, all_data_features, all_data_target, class_weights)
print(best_method)
best_grade_method <- best_method
print(model)

grade_model_file <- save_model(model, "best_grade_model")
```

```r
saved_model_file_paths <- list(
    pass_method = best_pass_method,
    pass_model_file = pass_model_file,
    grade_method = best_grade_method,
    grade_model_file = grade_model_file
)
json <- toJSON(saved_model_file_paths, pretty = TRUE, auto_unbox = TRUE)
write(json, file = "saved-model-file-paths.json")

data_info <- list(
    students_used = number_of_samples,
    pass_distribution = as.list(pass_distribution_table),
    grade_distribution = as.list(grade_distribution_table),
    grade_class_weights = as.list(class_weights)
)
json <- toJSON(data_info, pretty = TRUE, auto_unbox = TRUE)
write(json, file = "data-info.json")

cat("##################################################################\n")

# Ensemble metrics calculation
cat("Ensemble predictions metrics calculation\n")

features <- c("pass", "grade", additional_features, as.vector(outer(groups, suffixes, paste0)))

select_features_data <- data[features]
select_features_data <- select_features_data %>% dplyr::mutate(across(-c(grade, pass), as.double))
select_features_data$grade <- as.factor(select_features_data$grade)
select_features_data$pass <- as.factor(select_features_data$pass)

# Shuffle the data
shuffled_data <- select_features_data[sample(1:nrow(select_features_data)), ]
folds <- caret::createFolds(shuffled_data$grade, k = number_of_folds)

ensemble_metrics_list <- list(
    accuracy = c(),
    kappa = c(),
    balanced_accuracy = c(),
    f1_score = c()
)

for (i in 1:number_of_folds) {
    test_indices <- folds[[i]]
    test <- shuffled_data %>% dplyr::slice(test_indices)

    train_indices <- unlist(folds[-i])
    train <- shuffled_data %>% dplyr::slice(train_indices)
```

```
    cat("Fold iteration:", i, "/", number_of_folds, "\n")
    cat("Size of train sample:", nrow(train), "\n")
    cat("Size of test sample:", nrow(test), "\n")

    train_features <- train %>% dplyr::select(-c(grade, pass))
    test_features <- test %>% dplyr::select(-c(grade, pass))

    train_target <- train %>% dplyr::select(c(grade, pass))
    test_target <- test %>% dplyr::select(c(grade, pass))

    source(paste(pass_method_folder_path, "/", best_pass_method, ".R", sep = ""))
    pass_model <- train_model(
        train %>% dplyr::select(-grade),
        train_features,
        train_target$pass
    )
    pass_predictions <- get_predictions(pass_model, test_features)
    pass_predictions <- as.factor(pass_predictions)

    source(paste(grade_method_folder_path, "/", best_grade_method, ".R", sep = ""))
    grade_model <- train_model(
        train %>% dplyr::select(-pass),
        train_features,
        train_target$grade,
        class_weights
    )
    grade_predictions <- get_predictions(grade_model, test_features)
    grade_predictions <- as.factor(grade_predictions)

    # Initialize the predictions vector with the grade predictions
    predictions <- grade_predictions

    # Adjust the predictions based on the pass predictions
    predictions <-
        # if student fails, set the grade to 1
        ifelse(pass_predictions == "0", "1",
        # if student passes and the grade is 1, set the grade to 2
        ifelse(pass_predictions == "1" & grade_predictions == "1", "2",
        # else keep grade as is
        grade_predictions))
    predictions <- as.factor(predictions)

    metrics <- calculate_grade_model_metrics(predictions, test_target$grade)

    ensemble_metrics_list$accuracy <- c(ensemble_metrics_list$accuracy, metrics$accuracy)
    ensemble_metrics_list$kappa <- c(ensemble_metrics_list$kappa, metrics$kappa)
    ensemble_metrics_list$balanced_accuracy <- c(ensemble_metrics_list$balanced_accuracy,
metrics$balanced_accuracy)
    ensemble_metrics_list$f1_score <- c(ensemble_metrics_list$f1_score, metrics$f1_score)
```

```
}

ensemble_metrics <- sapply(metric_names, function(metric) {
    list(
        mean = mean(ensemble_metrics_list[[metric]])[1],
        sd = sd(ensemble_metrics_list[[metric]])[1]
    )
}, simplify = FALSE)
json <- toJSON(ensemble_metrics, pretty = TRUE, auto_unbox = TRUE)
write(json, file = "ensemble-metrics.json")
print(ensemble_metrics)

cat("#################### End of model training ####################\n")
```

R skripta za predviđanje uspjeha pomoću treniranih modela

```
library(dplyr)
library(caret)
library(jsonlite)

odds_decimal_precision <- 4
results_file_name_csv <- "student-prediction-data.csv"
prediction_info_name_json <- "prediction-info.json"

cat("#################### Generating predictions ####################\n")

# Load parameters from JSON file
params <- fromJSON("parameters.json")

# SCRIPT INPUTS from JSON
number_of_folds <- params$number_of_folds
course_id <- params$course_id
prediction_point <- params$prediction_point
groups <- params$groups
suffixes <- params$suffixes
additional_features <- params$additional_features
# Ensure params$additional_features is character type, even when empty
if (is.null(additional_features) || length(additional_features) == 0) {
    additional_features <- character(0) # Set to an empty character vector
} else {
    # Ensure it's treated as a character vector in case it's not
    additional_features <- as.character(additional_features)
}

pass_method_folder_path <- params$pass_methods_folder
grade_method_folder_path <- params$grade_methods_folder
```

```r
pass_best_method <- params$pass_best_method
pass_model_path_with_ext <- params$pass_model_filename
grade_best_method <- params$grade_best_method
grade_model_path_with_ext <- params$grade_model_filename


data_filename <- params$data_filename

# Load data
data <- read.csv(data_filename)
prediction_results <- data.frame(
    id_student = data$id_student,
    at_the_time_score = data$score
)

features <- c(additional_features, as.vector(outer(groups, suffixes, paste0)))
cat("Features used for forecasting:\n")
print(features)

data_features <- data[features]

cat("Forecasting with pass model\n")
cat("Best pass method: ", pass_best_method, "\n")
cat("Pass model path: ", pass_model_path_with_ext, "\n")

source(paste(pass_method_folder_path, "/", pass_best_method, ".R", sep = ""))
model <- load_model(pass_model_path_with_ext)

predictions <- get_predictions(model, data_features)
predictions <- as.factor(predictions)
probs <- get_prob_predictions(model, data_features)

prediction_results$predicted_pass <- predictions
prediction_results$fail_odds <- round(probs[, "0"], odds_decimal_precision)
prediction_results$pass_odds <- round(probs[, "1"], odds_decimal_precision)


cat("Forecasting with grade model\n")
cat("Best grade method: ", grade_best_method, "\n")
cat("Grade model path: ", grade_model_path_with_ext, "\n")

source(paste(grade_method_folder_path, "/", grade_best_method, ".R", sep = ""))
model <- load_model(grade_model_path_with_ext)

predictions <- get_predictions(model, data_features)
predictions <- as.factor(predictions)
probs <- get_prob_predictions(model, data_features)
```

```
prediction_results$predicted_grade <- predictions
prediction_results$grade_1_odds <- round(probs[, "1"], odds_decimal_precision)
prediction_results$grade_2_odds <- round(probs[, "2"], odds_decimal_precision)
prediction_results$grade_3_odds <- round(probs[, "3"], odds_decimal_precision)
prediction_results$grade_4_odds <- round(probs[, "4"], odds_decimal_precision)
prediction_results$grade_5_odds <- round(probs[, "5"], odds_decimal_precision)

write.csv(prediction_results, results_file_name_csv, row.names = FALSE, quote = FALSE)

cat("Prediction stats:\n")
total_number_of_students <- nrow(data)
number_of_fails <- sum(prediction_results$predicted_pass == 0)
number_of_passes <- sum(prediction_results$predicted_pass == 1)
number_of_grade_1 <- sum(prediction_results$predicted_grade == 1)
number_of_grade_2 <- sum(prediction_results$predicted_grade == 2)
number_of_grade_3 <- sum(prediction_results$predicted_grade == 3)
number_of_grade_4 <- sum(prediction_results$predicted_grade == 4)
number_of_grade_5 <- sum(prediction_results$predicted_grade == 5)

cat("Total number of students: ", total_number_of_students, "\n")
cat("Number of fails: ", number_of_fails, "\n")
cat("Number of passes: ", number_of_passes, "\n")
cat("Number of grade 1: ", number_of_grade_1, "\n")
cat("Number of grade 2: ", number_of_grade_2, "\n")
cat("Number of grade 3: ", number_of_grade_3, "\n")
cat("Number of grade 4: ", number_of_grade_4, "\n")
cat("Number of grade 5: ", number_of_grade_5, "\n")

cat("Results saved in: ", results_file_name_csv, "\n")

prediction_info <- list(
    total_number_of_students = total_number_of_students,
    number_of_fails = number_of_fails,
    number_of_passes = number_of_passes,
    number_of_grade_1 = number_of_grade_1,
    number_of_grade_2 = number_of_grade_2,
    number_of_grade_3 = number_of_grade_3,
    number_of_grade_4 = number_of_grade_4,
    number_of_grade_5 = number_of_grade_5,
    results_file_name_csv = results_file_name_csv
)
json <- toJSON(prediction_info, pretty = TRUE, auto_unbox = TRUE)
write(json, file = prediction_info_name_json)

cat("#################### Predictions generated ####################\n")
```