Študentov osebni svetovalec

Tomaž Jerman

28. junij 2017

1 Uvod

Ste se že kdaj vprašali, pod kakšnimi pogoji športnik, glasbenik ali raziskovalec doseže najboljše rezultate. Za začetek potrebuje talent in veselje do svoje panoge. Pa vendar to še zdaleč ni dovolj. Obstajati mora močna želja za preseganje samega sebe in pripravljenost na trdo delo, vztrajnost ter še kaj. Ste že slišali, da je za vsako mojstrstvo potreben nek minimum vloženega dela? Za posamezno področje to lahko znaša tudi do 10.000 ur in več.

Pomembna je tudi sredina, okolje, ki mora biti naravnano spodbujevalno. Sledi vloga trenerja/mentorja, ki osebno nenehno spodbuja in usmerja. Postavljati mu mora vmesne izzive, na poti do končnega cilja.

Nekoč so bila najpomembnejša tekmovanja enkrat na leto, ali celo na štiri leta (olimpijada), danes pa je tekmovanj vse več. Enako je tudi v šolstvu. Namesto končnega preverjanja ob diplomi, so bili najprej organizirani letni izpiti, sedaj pa že semestralni. Sprotni študij smo poudarili z domačimi nalogami in s posebej kolokviji – najprej s štirimi letno, sedaj pa imamo lahko že 5 kolokvijev v enem semestru. Najbolj idealno bi bilo, če bi dobivali izzive dnevno. Enako je s primerjavo. Tudi ta bi lahko bila nenehna - tako kot recimo merjenje časa na vsakem treningu, izmerjena višina pri vsakem skoku itd. Seveda boksar ne more imeti borbe vsak dan.

Cilj naše spletne aplikacije eQuiz je, da bi študentom omogočila doseganje čim boljših rezultatov in doseganje čim višjih ciljev. Pri semestralnih predmetih VIS (UNI-FRI) in OVS (VSŠ-FRI) potekata pilotska projekta. Naslednji pilot se izvaja na MF, pri predmetu Interna medicina.

V ta namen želimo izbrati za vsako posamezno skupino približno 1000 preverjenih nalog. Res pa je tudi, da bomo skušali naloge nenehno izboljševati. Reševalci (študentje) jih bodo med reševanjem ocenjevali (zahtevnost, duhovitost/izvirnost, poučnost). Tudi aplikacija sama bo z ratingom ugotavljala tako znanje študentov, kot nivo/zahtevnost nalog).

Hkrati bi radi povečali količino povratne informacije, ki jo ob reševanju dobi študent. Ravno pomankanje povratne informacije predstavlja pri veliki količini študentov veliko oviro, katero jim bomo s skupnimi močmi, pomagali preiti.

Spletna aplikacija napoveduje končni izid na osnovi sprotnega dela med semestrom (kolokviji, reševanje nalog v eQuiz-u). Tako študent pridobi povratne informacije med opravljanjem sprotnih obveznosti in vaj.

Študentje so pri predmetih lahko ne motivirani, predvsem zaradi pomanjkanja povratnih informacij napredovanja njihovega znanja med reševanjem nalog. Spletna aplikacija eQuiz proskuša ta problem odpraviti tako, da podatke študentovega sprotnega dela analizira ter napoveduje izid njegovih ocenjevanj in končnega izpita. Prav tako študentu svetuje teme, ki si jih mora dodatno pogledati zaradi pomankanja znanja.

Po mojem mnenju je spletna aplikacija predvsem namenjena študentom, ki si želijo dodatne motivacije s povratnimi informacijami glede na njihovo sprotno delo.

2 Podatki

Podatke pridobimo iz spletne aplikacije eQuiz, kolokvijev in izpitov. Iz eQuiz aplikacije pridobimo seznam rešenih kvizov za vsakega študenta posebej, iz kolokvijev in izpitov pa dobimo rezultat preverjanja znanja, ki dodatno vpliva na končni izračun koeficientov znanja. Koeficiente znanja smo razdelili na 15 različnih tem, katere predmet pokrije s tekom semestra.

Količinsko so podatki o reševanju nalog precej obsežni, vendar je bilo veliko število podatkov potrebno izpustiti, ker niso bili relavantni pri naših analizah. Prav tako je v uporabljenih virih podatkov velika količina manjkajočih podatkov, kar negativno upliva na napovedovanje.

2.1 Omejitve podatkov

Pri zasnovi aplikacije smo morali upoštevati določene omejitve, ki se navezujejo na podatke o sprotnem delu študentov.

- Študent za posamezno temo reši relativno majhno število nalog, torej mora algoritem čim prej najti nek koeficient, ki se zdi najprimernejši. Prav tako določeni študentje sploh ne rešujejo kvizov oziroma na kvizih naključno iščejo prave rešitve, kar prav tako zelo negativno vpliva na delovanje algoritma.
- Ker se študentje učijo tudi iz drugih virov (ne le kvizov), moramo to tudi upoštevati. Potreben je nek smiselen izhodiščni koeficient na katerega se dodatno preračunava znanje.
- Kolokviji se tematsko ne navezujejo en na drugega, torej je težko uporabiti predhodnja ocenjevanja za napovedovanje novih ocenjevanj.
- Testi se po težavnosti med seboj razlikujejo, torej je potrebno najti neko utež, ki to ocenjevanje najbolje predstavlja.

2.2 Koeficient znanja

Koeficient znanja je rezultat računskih in statističnih algoritmov, ki na podlagi sprotnega dela študenta (reševanje kvizov, ocenjevanj) določa čim boljšo predstavo znanja študenta za podano temo.

Koeficienti znanja prav tako določajo osnovno izhodišče za napovedovalni algoritem.

2.3 Ocenjevanja

Seznam ocenjevanj pridobimo iz .csv datotek, katere s PHP skripto analiziramo ter vnesemo v podatkovno bazo za nadaljno uporabo v napovedovanju. Skripta za vnos ocen je zgrajena precej fleksibilno, zaradi potreb razvoja, prenosljivosti na druge predmete in popravke vnosov - če so bili po nesreči vneseni narobe jih lahko skripta sama posodobi.

3 Algoritem napovedovanja

Za napovedovanje smo se odločili, da zgradimo lasten algoritem. Problem s katerim smo se spoprijeli je bil precej specifičen, napovedovanje se nanaša na majhno število ocenjevanj (8 - 5 kolokvijev in 3-je izpitni roki). Poleg tega smo v algoritmih morali upoštevati veliko količino manjkajočih podatkov, kar je predstavljalo zelo veliko oviro.

- Najprej smo morali določiti začetne koeficiente znanja za vsako temo (izmed 15-ih), katere smo pridobili s spodaj opisanim postopkom iz rešenih kvizov (4.1).
- Nato smo morali preveriti, če lahko pridobimo novi koeficient znanja glede na že vnesene študentove ocene. Če ima študent že vneseno oceno, lahko to izkoristimo za popravek na koeficientu znanja.
- Nato smo morali določiti neko grobo napoved s pomočjo začetnih koeficientov znanja. Tako smo pridobili neko osnovno predstavo o študentovem znanju za dano ocenjevanje (4.2).
- Sledi analiza razvijanja krivulje ocenjevanja, s pomočjo katere lahko napovedi nadaljnih ocenjevanj dodatno popravimo. To storimo z pomočjo intervalnega trenda, katerega določimo z metodo drsečih sredin.
- Nato smo morali upoštevati težavost različnih ocenjevanj. Po testiranjih se je izkazalo, da je uporaba dinamičnih uteži naj primernejša. S pomočjo ostalih študentov, ki so na danem ocenjevanju že ocenjeni in se s koeficienti znanja prilegajo ciljnemu študentu, lahko pridobimo predstavo o težavnosti ocenjevanja (4.4).
- Nato smo morali v začetno napoved dodati, do sedaj, preračunane in utežene količine, katere pa na koncu še dodatno utežimo z utežjo ocenjevanja (4.5).

  Novi napovedi še dodatno dodamo uteženo povprečno napako, ki jo pridobimo na podlagi ostalih študentov iz izbranega vzorca (4.6).
- Na koncu moramo pridobljene napovedi le še posredovati API-ju, ki poskrbi za lep izris grafov študentu za napovedovanje in prikaz koeficientov znanja (4.7.2).
- Aplikacija na podlagi podatkov o ocenjevanjih in koeficientih znanja zna študentu prikazati kako z ocenjevanjem napreduje glede na celoten razred (4.8).

4 Izračuni in rezultati

4.1 Izračun začetnega koeficienta znanja

Za začetno osnovo koeficienta znanja smo se odločili za vrednost **0,5**. Predpostavimo, da študent hodi na predavanja, vaje in ve osnove določene snovi. Dodatno znanje pokaže s sprotnim delom in na ocenjevanjih. Podatke o reševanju sprotnih kvizov nato analiziramo po sledečem algoritmu:

- Za vsakega študenta zgradimo posebno podatkovno strukturo, katera hrani podatke o vseh rešenih nalogah, urejenih po vrsnem redu reševanja.

- Za vsakega študenta dodatno zgradimo podatkovno strukturo, ki hrani podatke o vektorjih, koeficientu znanja in utežeh za vsako temo posebej.
- Nato pošiljamo podatke reševanj skozi algoritem, ki:
  - preveri ali je bila naloga rešena pravilno. Če je bila rešena pravilno, se začetnemu koeficientu doda vrednost po enačbi:

    $$\texttt{k} = \min(1, \texttt{k} + (\texttt{upVekt} + (\texttt{streak} \times \texttt{raiseMult})))$$

    Novemu koeficientu se doda pozitivni vektor (`upVekt`*), ki je povečan z utežjo (*`raiseMult`*) in številom zaporednih pravilnih odgovorov (*`streak`*).* Koeficient spreminjamo na takšen način, ker nočemo, da manjše napake vplivajo preveč drastično na končni koeficient (vsak se kdaj zmoti). Na takšen način prav tako hitro pridemo do nekega koeficienta, ki študentovo znanje čim bolje predstavlja.
  - Spremeniti moramo tudi vektorja ter njuni uteži.

    $$\texttt{upVekt} = \min(0.05, \texttt{upVekt} \times (\texttt{streak} \times \texttt{raiseMult}))$$

    $$\texttt{downVekt} = \max(0.005, \frac{\texttt{downVekt}}{1 + \texttt{streak} \times \texttt{descendMult}})$$

    Imenovalcu moramo prišteti 1, ker je zmožek med 0 in 1, in ne želimo večati tega vektorja!

    $$\texttt{raiseMult} = \min(0.03, \texttt{raiseMult} \times 2)$$

    $$\texttt{descendMult} = \max(0.03, \frac{\texttt{descendMult}}{2})$$

    Množitelja in vektorja sta omejena, ker se lahko zgodi, da bi eden popolnoma izničil drugega, ker se precej hitro spreminjata.

    Pridobljene količine nato shranimo v posebno podatkovno strukturo za nadaljnjo uporabo.
  - Če naloga ni bila rešena pravilno, se zgodijo podobni preračuni, le v nasprotno smer.
- Po preračunu koeficientov za posamezne teme jim dodatno pripišemo lastnika s preslikavo med identifikacijo iz aplikacije eQuiz v vpisno številko. Takšne podatke nato pošljemo v podatkovno bazo za nadaljno uporabo.
- Takšen način pridobivanja koeficienta znanja je precej učinkovit in natančen, saj ocenjuje napredovanje študenta, prav tako pa ne kaznuje manjših napak, ki jih je študent morda naredil zaradi površnosti ali pa kaj podobnega.

4.2 Izračun začetne točke napovedi

Najprej je bilo potrebno določiti neko začetno točko napovedovanja. Ta korak se je, zaradi mankajočih podatkov, izkazal za precej problematičnega.

- Če je študent na dani točki napovedovanja že ocenjen, smo najprej morali dodatno preračunati koeficiente znanja za teme, ki so vključene v to ocenjevanje. Ocenjevanja približno pokažejo študentovo znanje za dene teme, prav tako pomagajo odpraviti problem manjkajočih podatkov pri študentih, ki ne opravljajo sprotnih obveznosti.

  Koeficientov ne nastavimo na dejansko oceno, ampak jih le dodatno popravimo, ker na podlagi ocene ne moremo določiti točnega koeficienta znanja.

  Novi koeficient pridobimo po spodnji enačbi:

  - **myScore** predstavlja procente točk pridobljenih na ocenjevanju,
  - **znanje** predstavlja koeficient znanja preračunan za dano temo,
  - **devisionAmount** predstavlja parameter oženja razpona koeficienta znanja. Pridobljen je bil s testiranjem različnih vrednosti, obdržala se je najboljša.

  Novi koeficient je preračunan tako, da trenutni koeficient znanja za posamezno temo zmanjšamo za ***devisionAmount*** *in ga prestavimo poleg pridobljene ocene. Tako v koeficient upoštevamo znanje, ki ga je pokazal na ocenjevanju in možnost napake na novi oceni znanja.*

  $$\texttt{nK} = \max(0, \texttt{myScore} - \frac{\texttt{znanje}}{\texttt{devisionAmount}})$$

  Če je študent ocenjen nad oceno znanja, oziroma

  $$\texttt{nK} = \min(1, \texttt{myScore} + \frac{\texttt{znanje}}{\texttt{devisionAmount}})$$

  če je študent ocenjen pod oceno znanja.

- začetno točko napovedi predstavlja povprečje koeficientov znanja med temami, ki so vključene v določeno ocenjevanje. *Znanje, ki naj bi ga študent pokazal na ocenjevanju je enak povprečnemu znanju vseh tem, ki so vključene v to ocenjevanje.*

- V trenutni verziji aplikacije se pri študentih, ki imajo pomankljive podatke (torej niso reševali nalog, se ne udeležujejo ocenjevanj ipd.) ne poskusimo pridobiti predvidenega koeficienta znanja, ker se je izkazalo kot manj učinkovito.

4.3 Izračun povprečne napake napovedovanja

Skozi razvoj napovedovalnega algoritma se preračunava utežena povprečna napaka med napovedjo in oceno, katera se upošteva v končni določitvi napovedi. V vsaki sledeči točki napovedovanja upoštevamo napake med dobljeno oceno in napovedano oceno prejšnih ocenjevanj, katero upoštevamo pri končni napovedi ocene.

- **predictionChanger** je preračunana povprečna utežena napaka med napovedovanjem in pridobljeno oceno,

- **predictionBuffer** vsebuje količino napake med napovedjo in pridobljeno oceno,

- **predictionBufferCounter** vsebuje količino vpoštevanih napak med napovedjo in ocenjevanjem,

- **predictionBufferCounterBuff** dodatno zmanjša pridobljeno povprečno napako. Količina je bila pridobljena s testiranjem različnih vrednosti, obdržala se je najboljša.

$$\texttt{predictionChanger} = \frac{\texttt{predictionBuffer}}{\texttt{predictionBufferCounter} \times \texttt{predictionBufferCounterBuff}}$$

Tako upoštevamo napake, ki se pojavljajo med napovedanimi ocenami in ocenami na ocenjevanjih. Te napake poskušamo z potekom algoritma izničiti.

Povprečna napaka je dodatno utežena, ker se ocenjevanja predvsem ne navezujejo tematsko, zato ne moremo upoštevati celotne napake za naslednja ocenjevanja.

4.4 Pridobivanje uteži ocenjevanja

V nadaljevanju je potrebno pridobiti utež za posamezno ocenjevanje, kjer se upošteva osnovna utež za posamezno ocenjevanje:

- kolokviji imajo začetno utež **0,8**
- izpiti imajo začetno utež **0,5**

Sledi pridobivanje uteži s pomočjo ostalih študentov. Iz podatkovne baze izluščimo tiste študente, ki se čim bližje prilagajajo ciljnemu študentu na podlagi koeficienta znanja znotraj določenega intervala. *Interval je implementiran, ker želimo uporabiti le študente, ki so po znanju podobni ciljnemu študentu za čim boljši rezultat.*

Nato po spodnjih enačbah približamo našo napoved pridobljeni uteženi uteži iz podatkovne baze.

Določeni fiksni uteži dodamo oz. odvzamemo nek delež pridobljene vrednosti iz podatkovne baze, na podlagi napovedi ciljnega študenta in povprečne ocene iz izbranega vzorca. Našo napved želimo le približati pridobljenemu povprečju, ker ne moremo z veliko verjetnostjo sklepati, da bo pridobil enako oceno. Zaradi tega je pridobljen podatek iz podatkovne baze dodatno utežen.

- **specialDragDown** je pridobljena dinamična utež za določeno ocenjevanje,
- **usedBase** vsebuje začetno statično utež, ki je podana vsakemu ocenjevanju posebej,
- **weight** je preračunano povprečje ocen izbranega vzorva iz podatkovne baze,
- **defaultFixedDbWeightUse** je konstantna vrednost, ki predstavlja količino pridobljenega povprečja iz podatkovne baze. Količina je bila pridobljena s testiranjem raznih vrednosti, obdržala se je najboljša.

$$\texttt{specialDragDown} = \texttt{usedBase} + \texttt{weight} \times \texttt{defaultFixedDbWeightUse}$$

$$\texttt{specialDragDown} = \texttt{usedBase} - \texttt{weight} \times \texttt{defaultFixedDbWeightUse}$$

Utež ocenjevanja računamo tako pozno, ker smo želeli uporabiti čim bolj preračunano napoved, kar se je izkazalo, da daje boljše rezultate, kot računanje uteži na samem začetku.

4.5 Združevanje pridobljenih podatkov

Sledi še predzadnja večja enačba, ki združi vse do sedaj preračunane količine, ki tvorijo napoved glede na uporabnikov razvoj znanja.

Novi napovedi dodamo prej preračunano grobo napoved, povprečno uteženo napako med napovedjo in oceno. Pridobljen rezultat dodatno utežimo z utežjo za dano ocenjevanje.

- **newPrediction** je napoved, ki jo pridobimo po preračunu enačbe,
- **prediction** je trenutna groba napoved,
- **predictionChanger** je preračunana utežena napaka med napovedjo in oceno,
- **specialDragdown** je preračunana utež za dano ocenjevanje.

$$\texttt{newPrediction} = \min(1, (\texttt{prediction} + \texttt{predictionChanger}) \times \texttt{specialDragDown})$$

Z zgornjo enačbo tako pridobimo precej natančno napoved. Sedaj jo še dodatno dopolnemo s pomočjo podatkov ostalih študentov.

4.6 Uporaba povprečne napake vzorca za trenutno ocenjevanje

V zadnjem pomembnejšem algoritmu še dokončno dodelamo našo napoved za določeno ocenjevanje. Iz podatkovne baze izberemo vzorec za določeno ocenjevanje, ter pridobimo povprečno napako med napovedjo in oceno. Pridobjeno napako ponovno utežimo iz istih razlogov kot prej. *Študentje se med seboj razlikujejo, zato ne moremo z vso verjetnostjo upoštevati napak drugih kot napako izbranega študenta.*

Dokončno napoved dobimo tako, da trenutni napovedi ***recalcPrediction*** *dodamo uteženo povprečno napako iz podatkovne baze iz izbranega vzorca. Vzorec ni naključen, ker smo želeli napraviti čim bolj konstantno napoved, kar z naključnim vzorcem ne bi bilo mogoče.*

- **recalcPrediction** je končna napoved,
- **recalcPrediction** je trenutna napoved,
- **dbAvg** je povprečna napaka iz podatkovne baze nad izbranim vzorcem,
- **avgAdderScale** je določena konstantna utež, ki določa koliko napake iz podatkovne baze naj se upošteva. Vrednost je bila pridobljena s testiranjem različnih vrednosti, obdržala se je najboljša.

$$\texttt{recalcPrediction} = \min(1, \texttt{recalcPrediction} + \texttt{dbAvg} \times \texttt{avgAdderScale})$$

4.7 Priprava končnega rezultata

Končna napoved je prikazana v grafu s pomočjo AMCharts API-ja, kateremu moramo priskrbeti json objekte za izris.

4.7.1 Estesko izpopolnjevanje izrisa

Pridobljene podatke podamo algoritmu, ki jih estetsko obdela. Upošteva razvoj grafa in s pomočjo dodanih pravil preslika napredovanje grafa v barvno kodo, katera označuje stanje razvoja študentovega znanja.

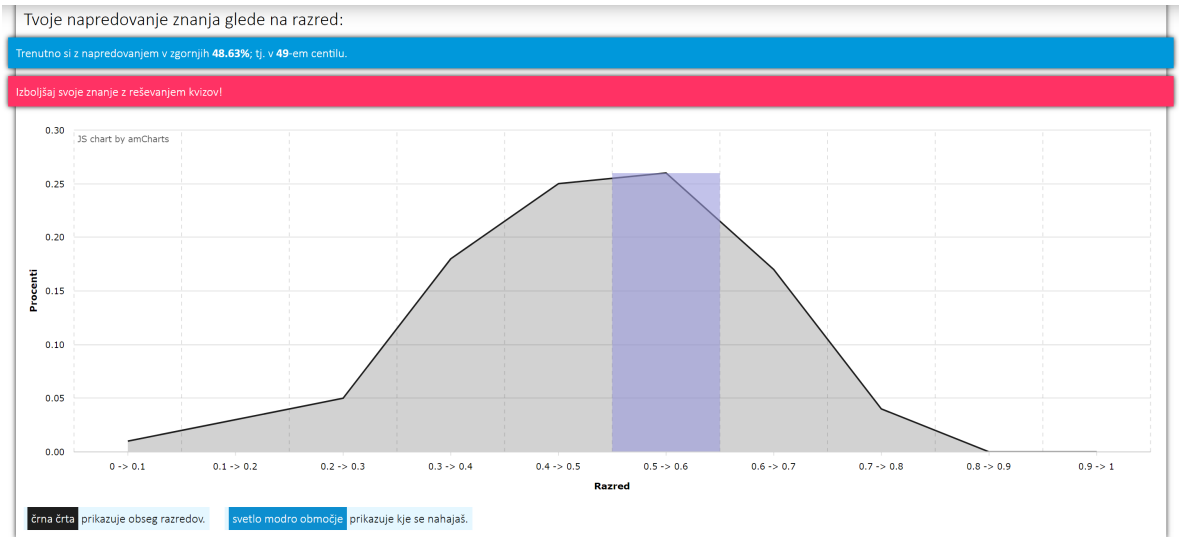
4.7.2 Priprava json objektov

Pridobljene podatke napovedovanj, ocenjevanj, koeficientov znanja podamo algoritmu, ki pripravi primerno json strukturo za AMCharts API. Nato pridobljene objekte dodatno obdelamo s skripto, ki naredi končne estetske popravke.

Na koncu vse skupaj podamo AMCharts API-ju, kateri podatke izriše uporabniku.

4.8 Dodatna motivacija študentu

Spletna aplikacija je prav tako zmožna ugotoviti študentov napredek znanja glede na opazovan razred. Ta podatek mu predstavi z informacijo v katerem centilu se nahaja, prav tako ta podatek grafično prikaže v kateri razpon znanja spada z grafom 1. Študentu aplikacija prav tako izpiše motivacijska sporočila glede na trenutni napredek znanja, torej v katerem centilu se študent nahaja. *Izkaže se da je znanje razreda porazdeljeno približno normalno.*



Slika 1: Primerjava študentovega napredovanja znanja glede na opazovani razred.

4.9 Napovedovanje možnosti udeležitve končnega izpita

Algoritem napovedovanja napoveduje tudi končni izid kolokvijev in verjetnost, da bo študent zbral dovolj točk za udeležitev izpita (120 točk).

Najprej se preveri, če študent že zadostuje podanemu pogoju za udeležitev končnega izpita, ki je **120** točk zbranih na kolokvijih.

Če študent še nima izpolnjenega pogoja za končni izpit:

- Se najprej preveri, če študent sploh še lahko pridobi dovolj točk za opravljeni pogoj. Ta podatek pridobimo tako, da pogledamo, če še sledi kakšno ocenjevanje. V primeru, da je bilo to zadnje ocenjevanje, se algoritem takoj zaključi in sporoči, da se izpita ne more udeležiti.

- Nato algoritem preveri povprečno število točk, ki jih študent ima ter ugotovi ali bo študent predvideno prejel dovolj dodatnih točk za opravljeni pogoj. V primeru da algoritem ugotovi, da študent ne bo zbral dovolj točk, mu to sporoči kot dodatna motivacija.

  Število dodatnih točk dobimo tako, da povprečno število točk pomnožimo z $\boldsymbol{n}$*, kjer je* $\boldsymbol{n}$ *število preostalih kolokvijev.*

- Na koncu algoritem še napove verjetnost takšnega izida. Takšno napoved je zelo teško napraviti in smo se odločili za sledeč postopek:

  – Preverimo nihanje pridobljenih ocen od povprečne ocene študenta in s pomočjo nihanja določimo zgornjo in spodnjo mejo ocen, ki jih je študent zmožen pridobiti. Nato z geometrijsko verjetnostjo pridobimo verjetnost, za takšen izid.

  Preračunamo geometrijsko verjetnost, ki predstavlja verjetnost, da smo končni izid napovedali pravilno.

    * **sumRange** predstavlja razpon nihanja znanja od povprečne ocene ocenjevanj,
    * **maxTestScore** predstavlja največ možnih točk na ocenjevanju,
    * **hitChanse** predstavlja verjetnost takšnega izida.

$$\texttt{procentsRange} = \frac{\texttt{sumRange}}{\texttt{maxTestScore}}$$

$$\texttt{hitChanse} = 1 - \texttt{procentsRange}$$

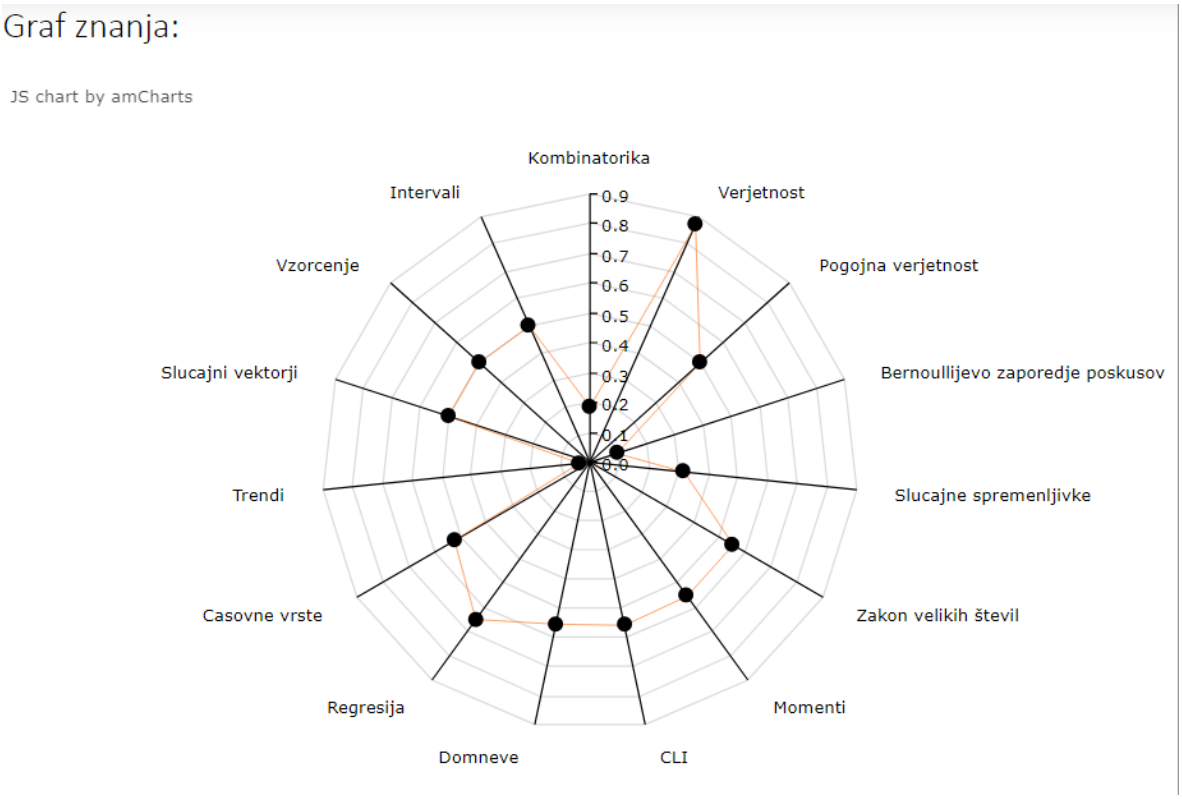
4.10 Prikaz študentovih koeficientov znanja

Študentu je na voljo tudi prikaz njegovih koeficientov znanja, ki jih aplikacija pridobi iz reševanja kvizov (4.1). Koeficienti so prikazani v radarskemu grafu, ki je na pogled hitro razumljiv in estetsko privlačen.

4.11 Prkaz izrisa grafa koeficientov znanja

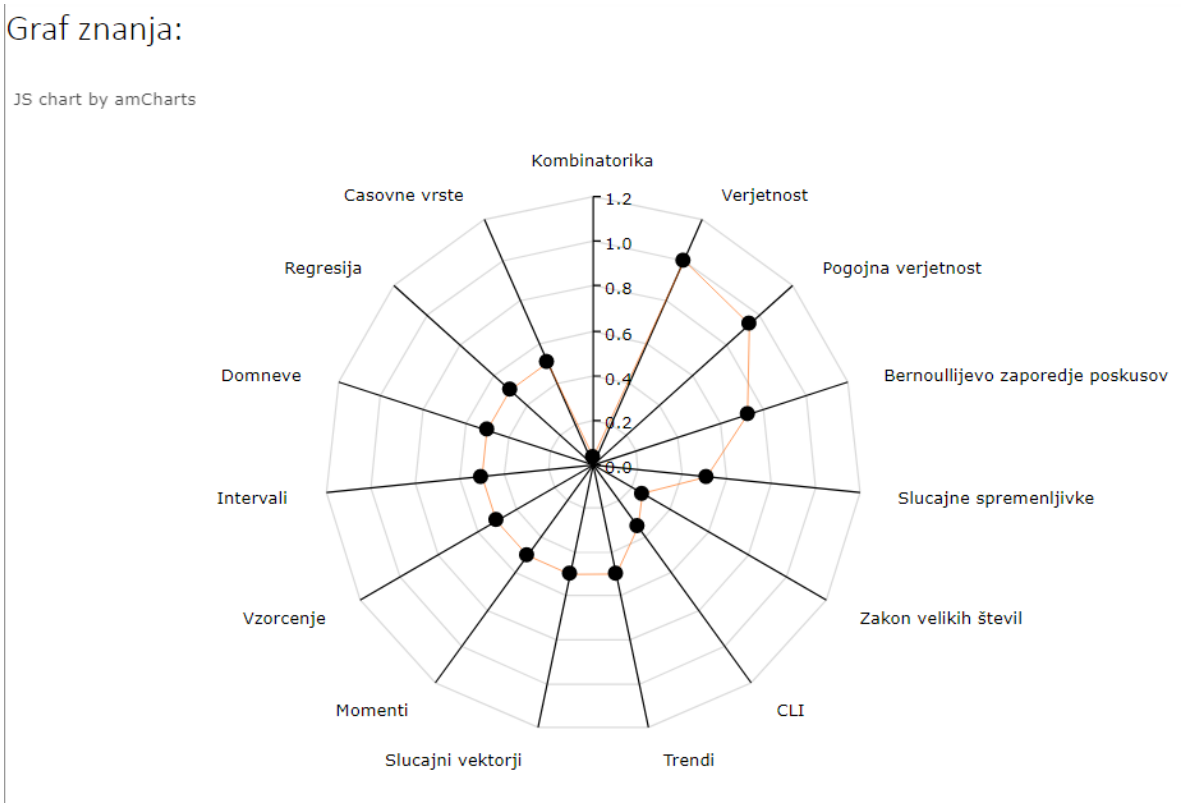
Sliki 2 in 3 prikazujeta izpis koeficientov znanja za izbrana študenta.

4.11.1 Primer izrisa 1



Slika 2: Prikaz koeficientov znanja za podanega študenta. Študent je rešil zelo malo nalog. zato nima preračunanih podatkov.

4.11.2 Primer izrisa 2



Slika 3: Prikaz koeficientov znanja za podanega študenta.

4.12 Prikaz delovanja algoritma napovedovanja

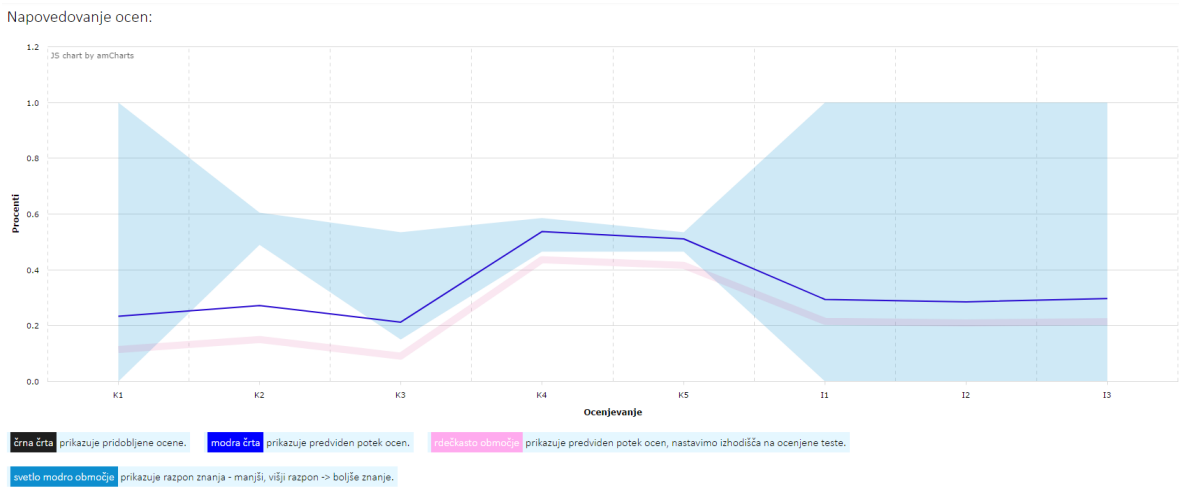
Spodnje skupine slik prikazujejo delovanje algoritma na izbranih študentih generacije 2015/16 oziroma 2016/17

Slike 4, 5, 6, 7, 8, 9, 10, 11 prikazujejo razvoj napovedovanja ocenjevanja za prvi primer.

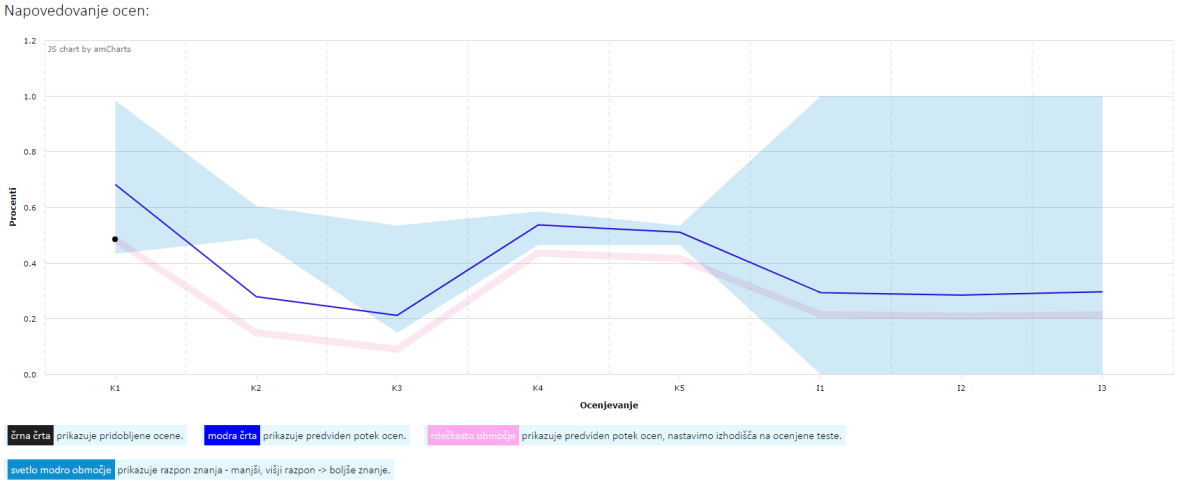
Slike 12, 13, 14, 15, 16, 17, 18, prikazujejo razvoj napovedovanja ocenjevanja za drugi primer.

Slike 19, 20, 21, 22, 23, 24, 25, prikazujejo razvoj napovedovanja ocenjevanja za tretji primer.

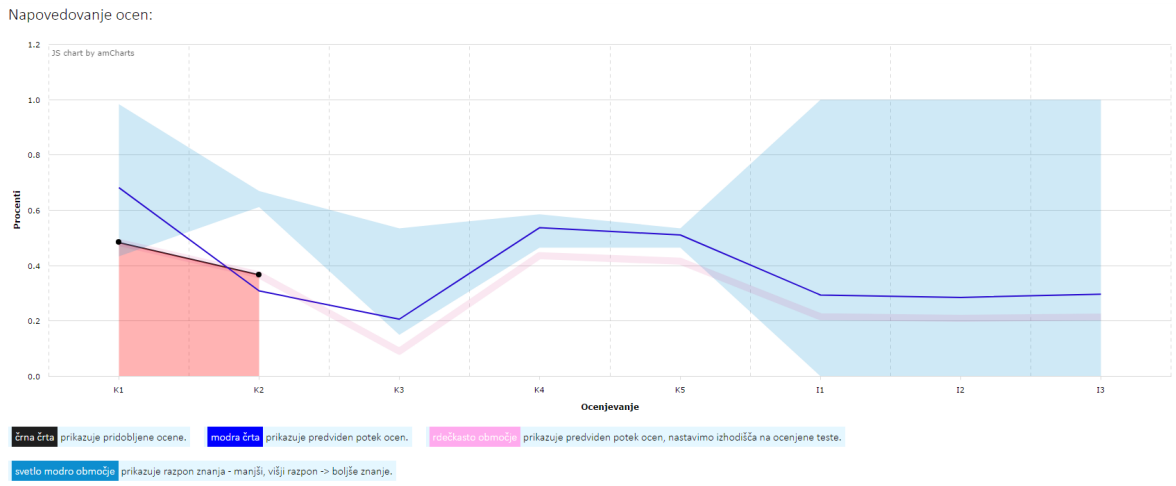
4.12.1 Primer 1 generacije 2015/16



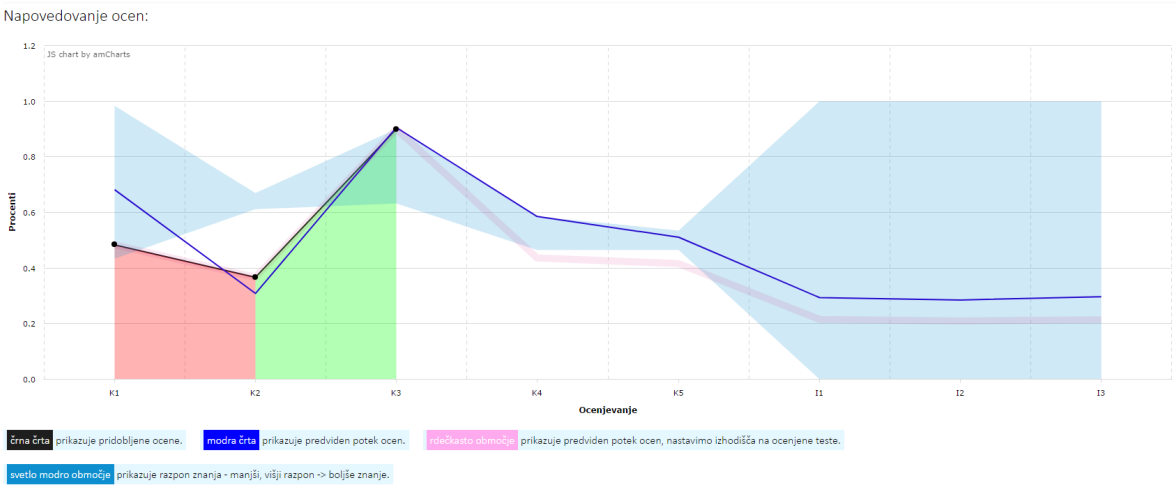
Slika 4: Začetna napoved ocenjevanja brez vnesenih ocenjevanj.



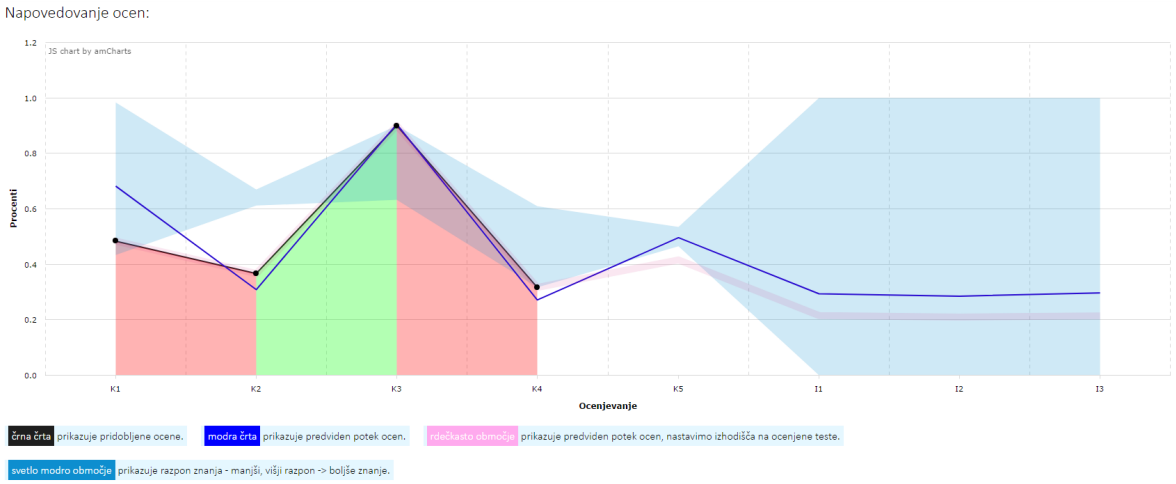
Slika 5: Začetna napoved ocenjevanja z enim vnesenim ocenjevanjem.



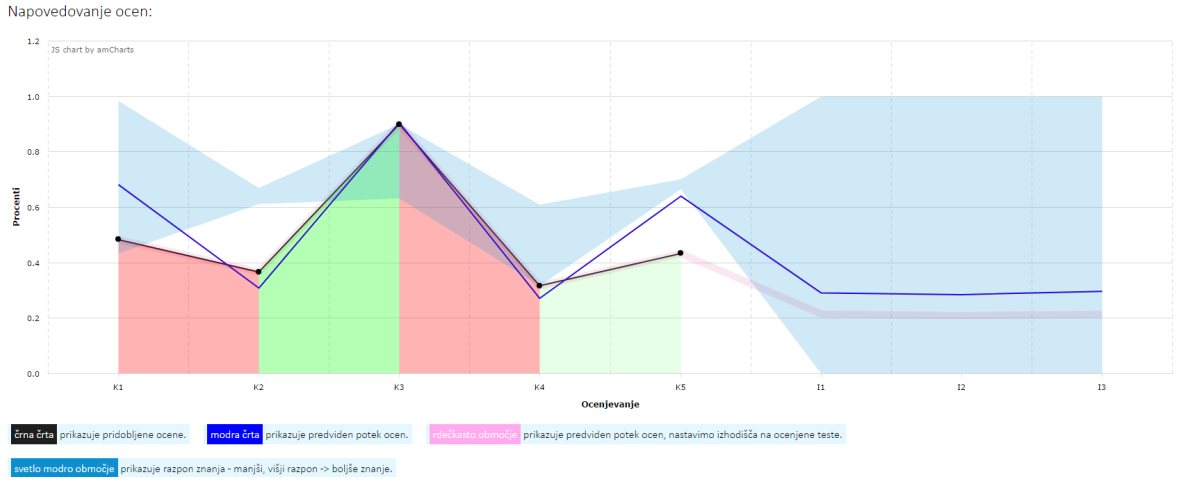
Slika 6: Začetna napoved ocenjevanja z dvema vnesenima ocenjevanjema.



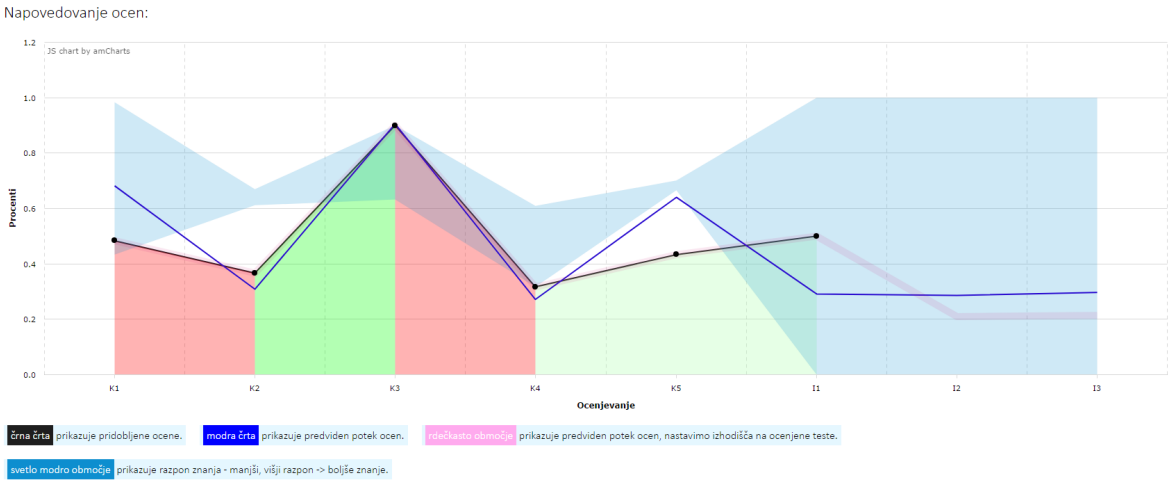
Slika 7: Začetna napoved ocenjevanja s tremi vnesenimi ocenjevanji.



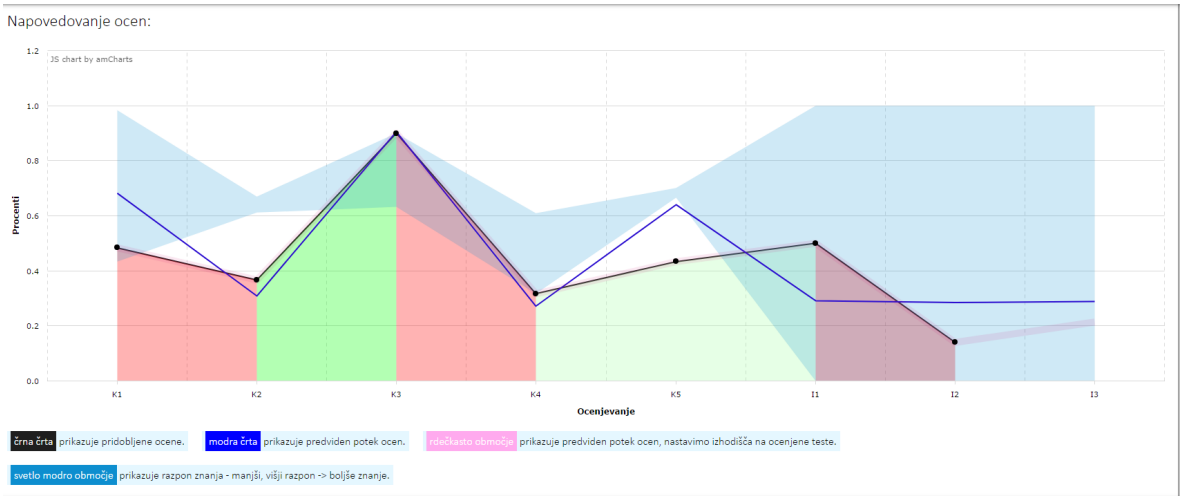
Slika 8: Začetna napoved ocenjevanja s štirimi vnesenimi ocenjevanji.



Slika 9: Začetna napoved ocenjevanja s petimi vnesenimi ocenjevanji.



Slika 10: Začetna napoved ocenjevanja s šestimi vnesenimi ocenjevanji.



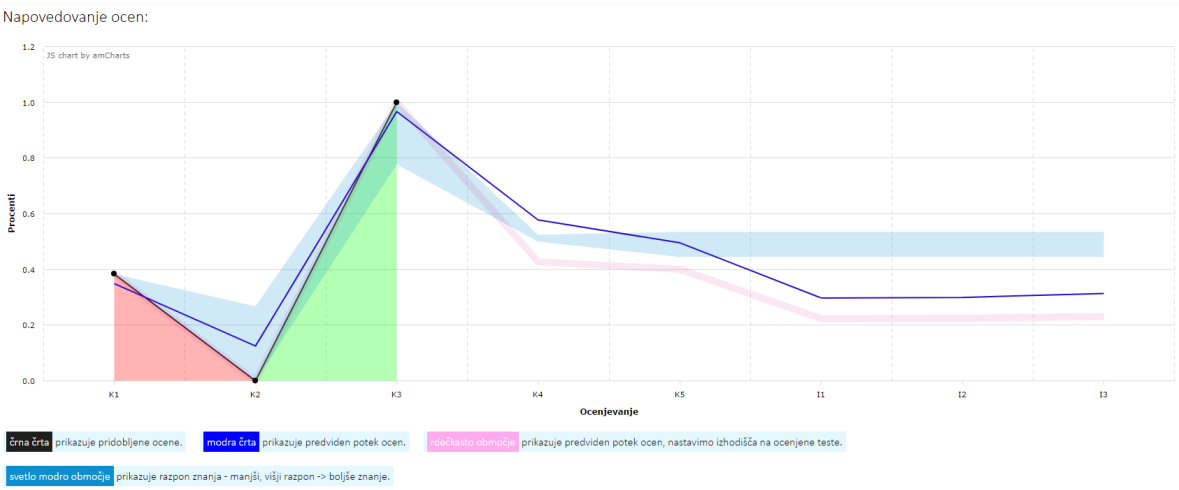
Slika 11: Začetna napoved ocenjevanja s sedmimi vnesenimi ocenjevanji.

4.12.2 Primer 2 generacije 2015/16

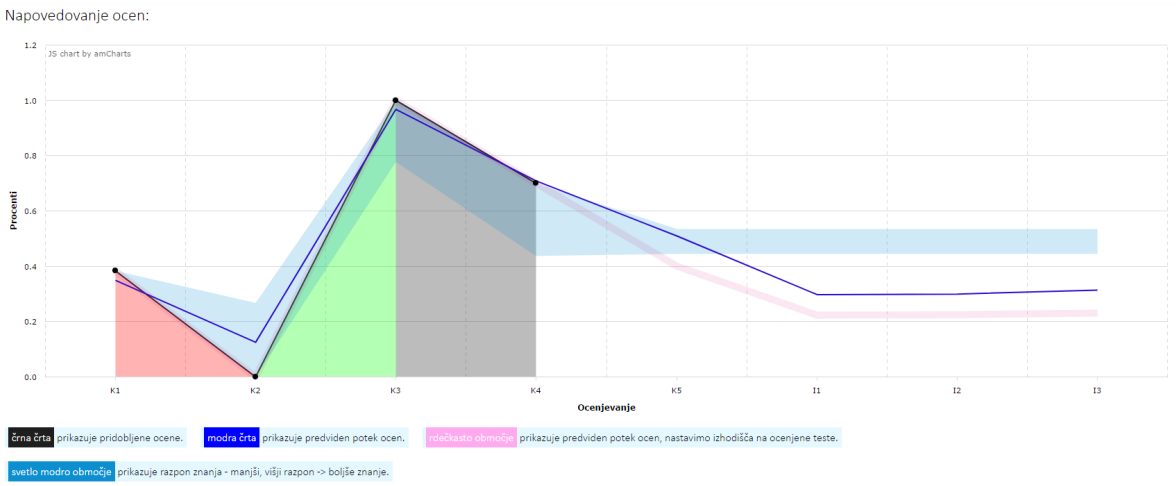
Slika 12: Začetna napoved ocenjevanja brez vnesenih ocenjevanj.

Slika 13: Začetna napoved ocenjevanja z dvema vnesenima ocenjevanjema.

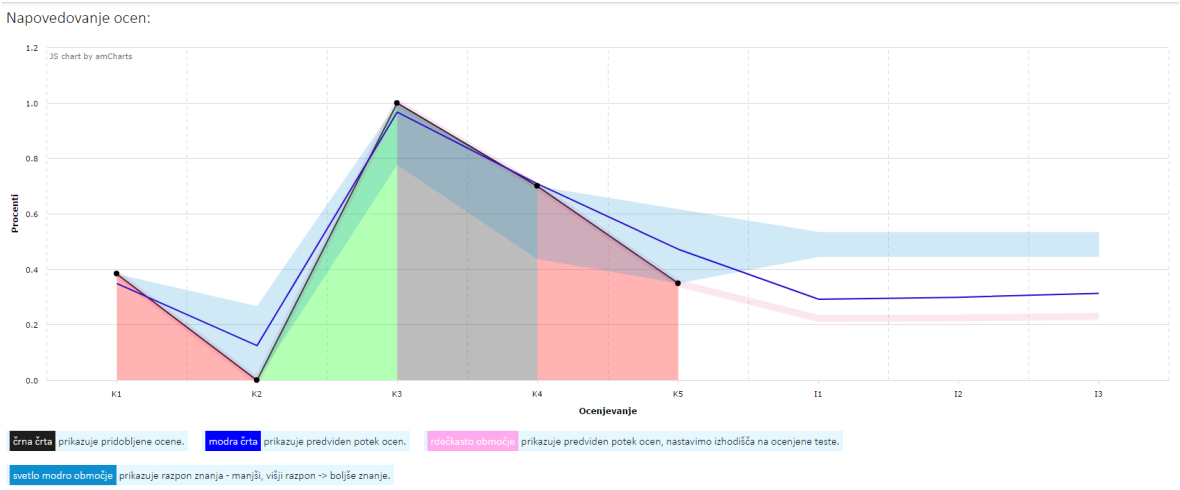
Slika 14: Začetna napoved ocenjevanja s tremi vnesenimi ocenjevanji.



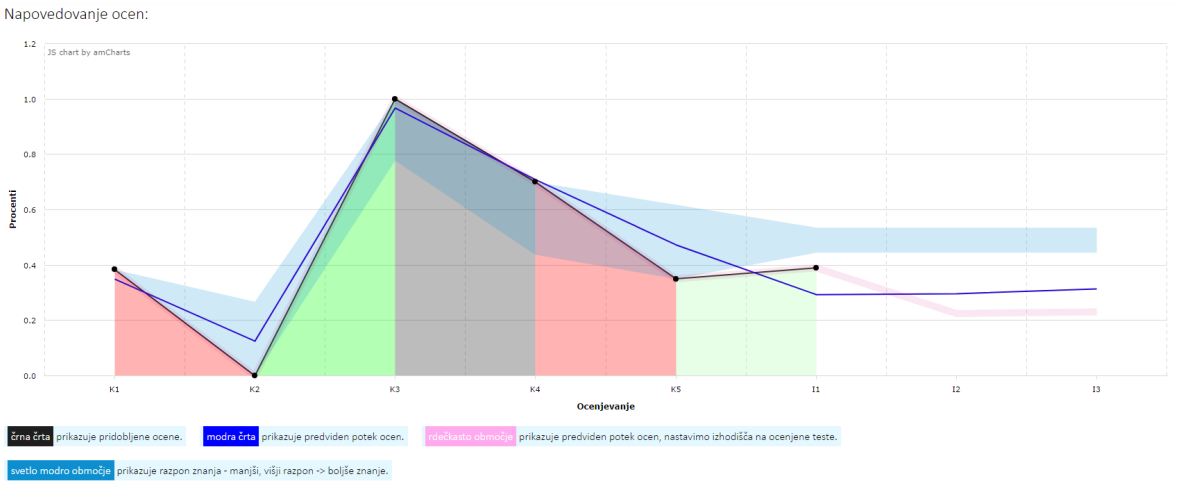
Slika 15: Začetna napoved ocenjevanja s štirimi vnesenimi ocenjevanji.



Slika 16: Začetna napoved ocenjevanja s petimi vnesenimi ocenjevanji.

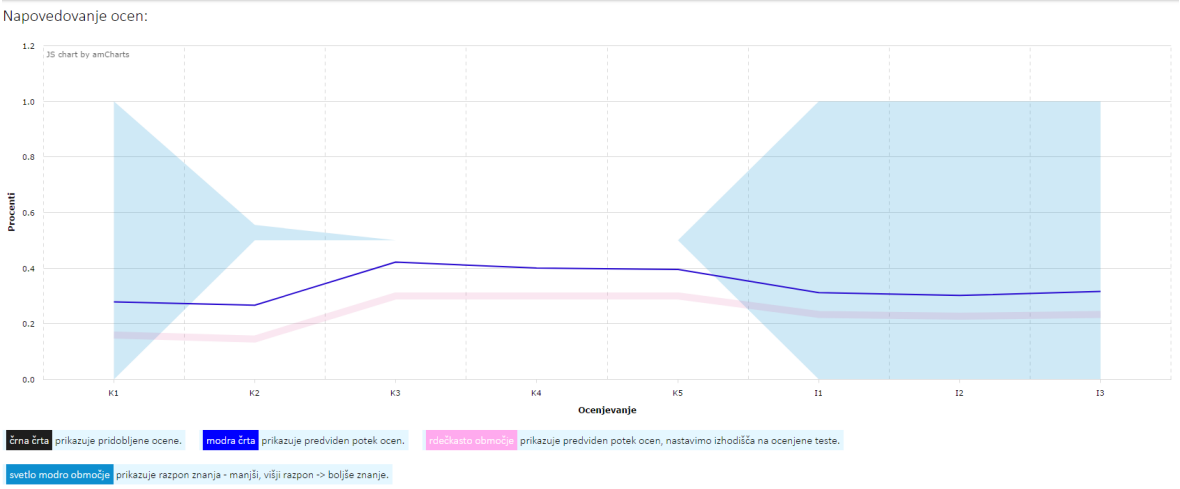


Slika 17: Začetna napoved ocenjevanja s šestimi vnesenimi ocenjevanji.

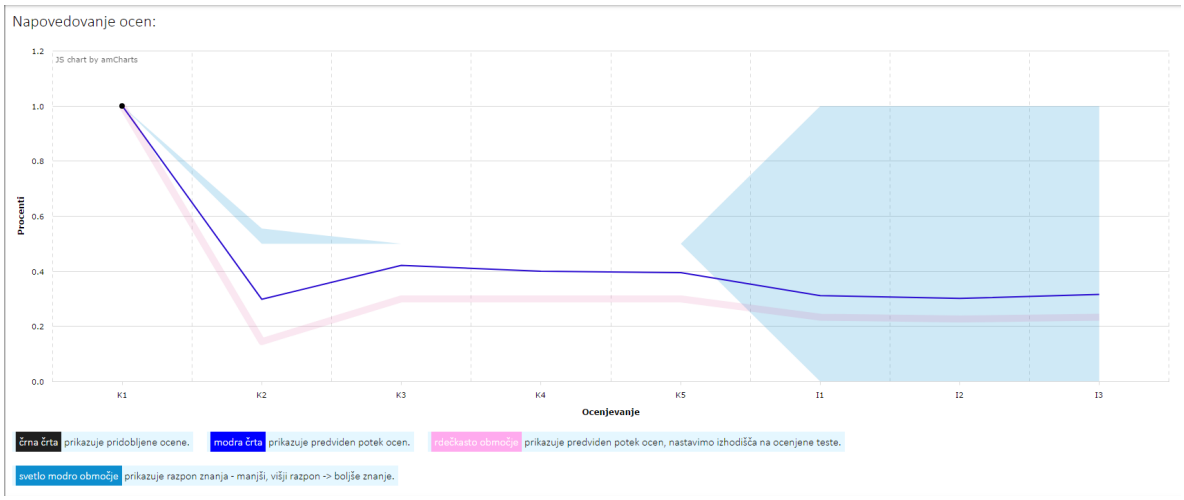


Slika 18: Začetna napoved ocenjevanja s sedmimi vnesenimi ocenjevanji.

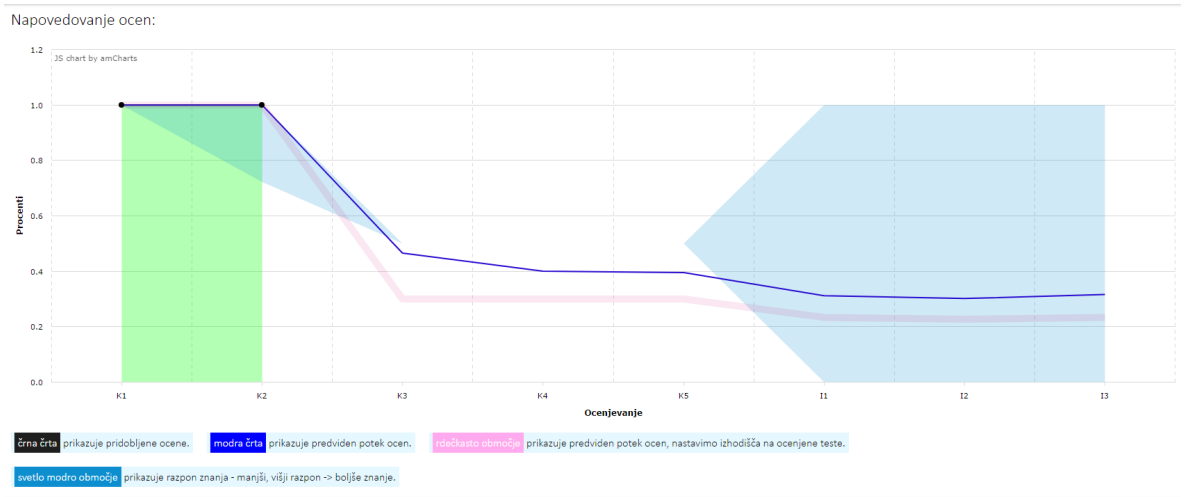
4.12.3 Primer 3 generacije 2016/17



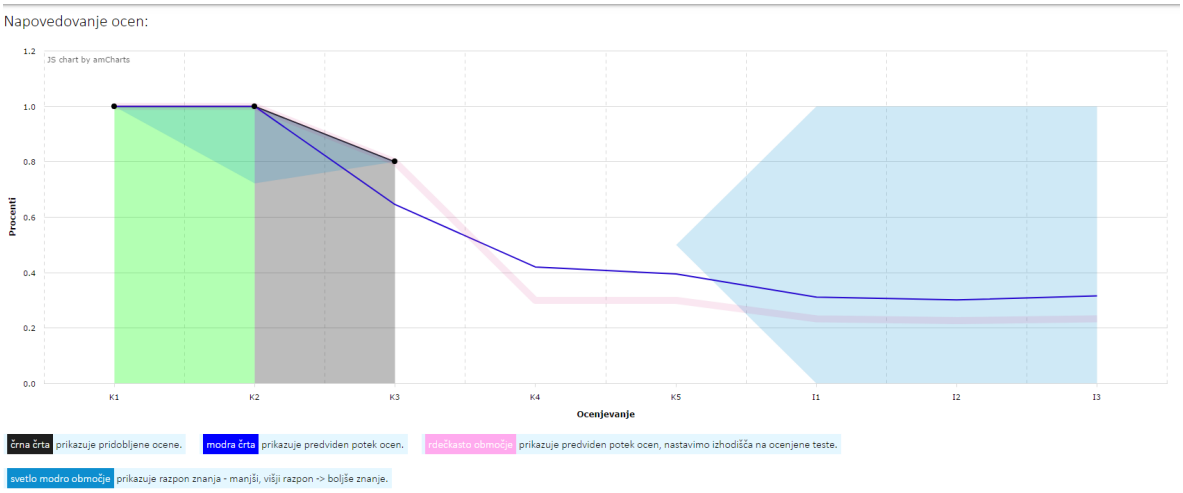
Slika 19: Začetna napoved ocenjevanja brez vnesenih ocenjevanj.



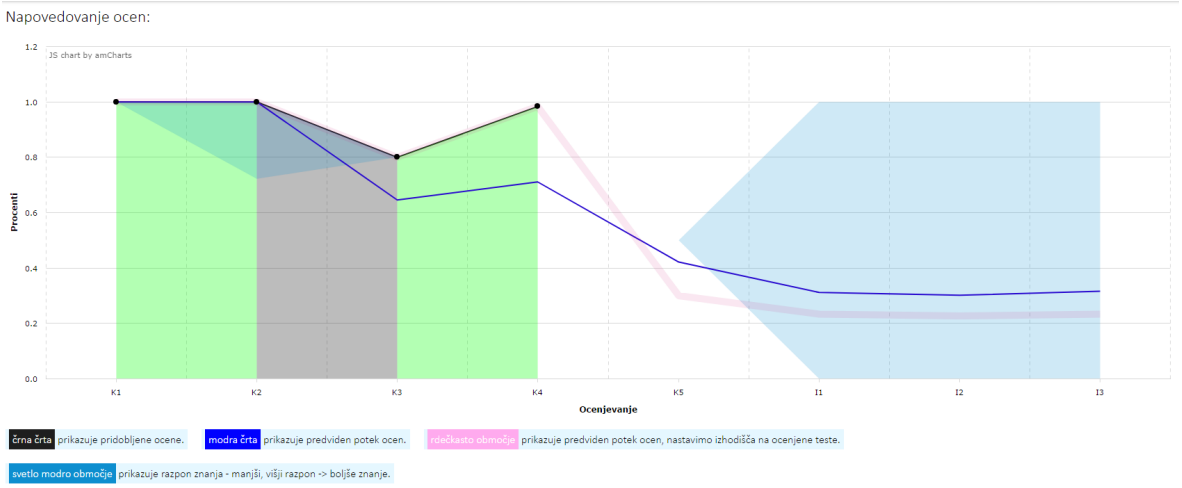
Slika 20: Začetna napoved ocenjevanja z dvema vnesenima ocenjevanjema.



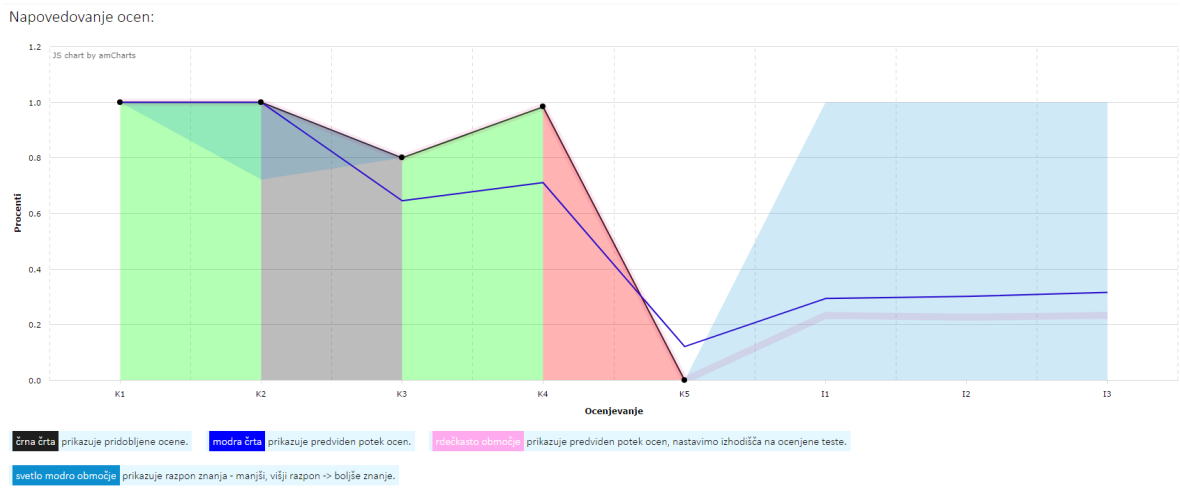
Slika 21: Začetna napoved ocenjevanja s tremi vnesenimi ocenjevanji.



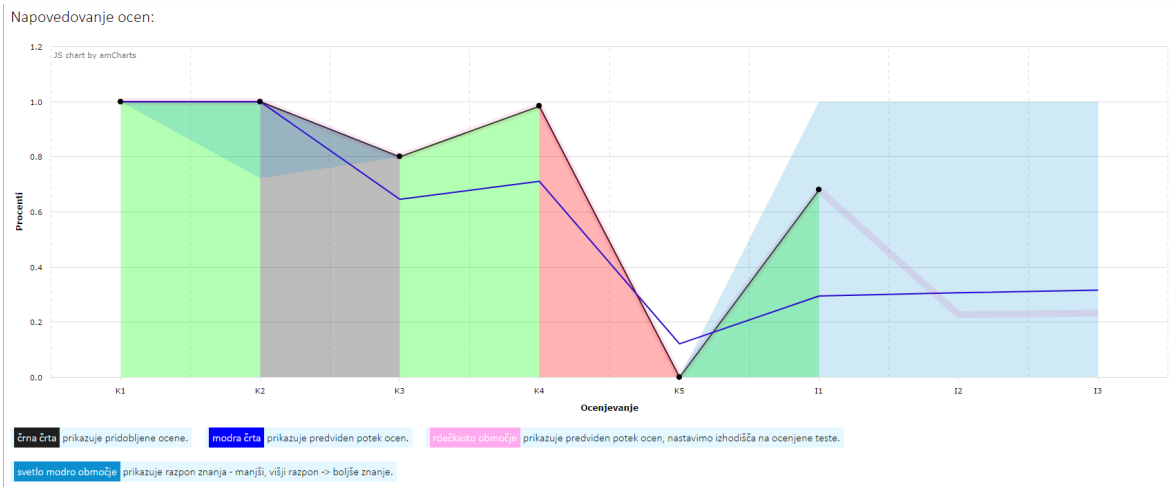
Slika 22: Začetna napoved ocenjevanja s štirimi vnesenimi ocenjevanji.



Slika 23: Začetna napoved ocenjevanja s petimi vnesenimi ocenjevanji.



Slika 24: Začetna napoved ocenjevanja s šestimi vnesenimi ocenjevanji.



Slika 25: Začetna napoved ocenjevanja s sedmimi vnesenimi ocenjevanji.

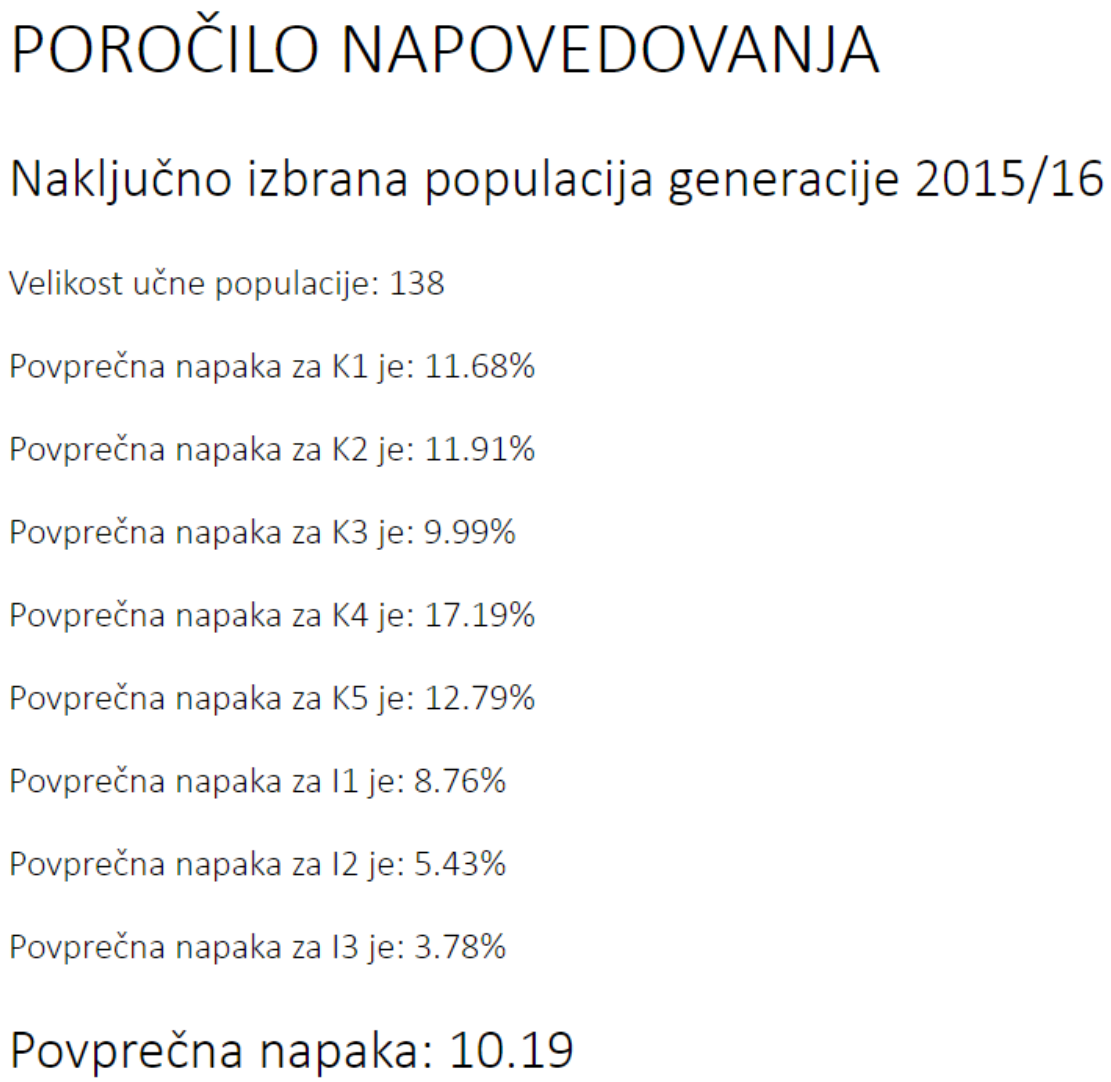
4.13 Prikaz poročila napovedovanja

Oznake ocenjevanj:

- **K#** kolokvij, številka #,
- **I#** izpitni rok, številka #.

Poročilo 1 (26) prikazuje končni rezultat napovedovanja ocenjevanj nad generacijo 2015/16 po vseh preračunih in končni obdelavi podatkov.

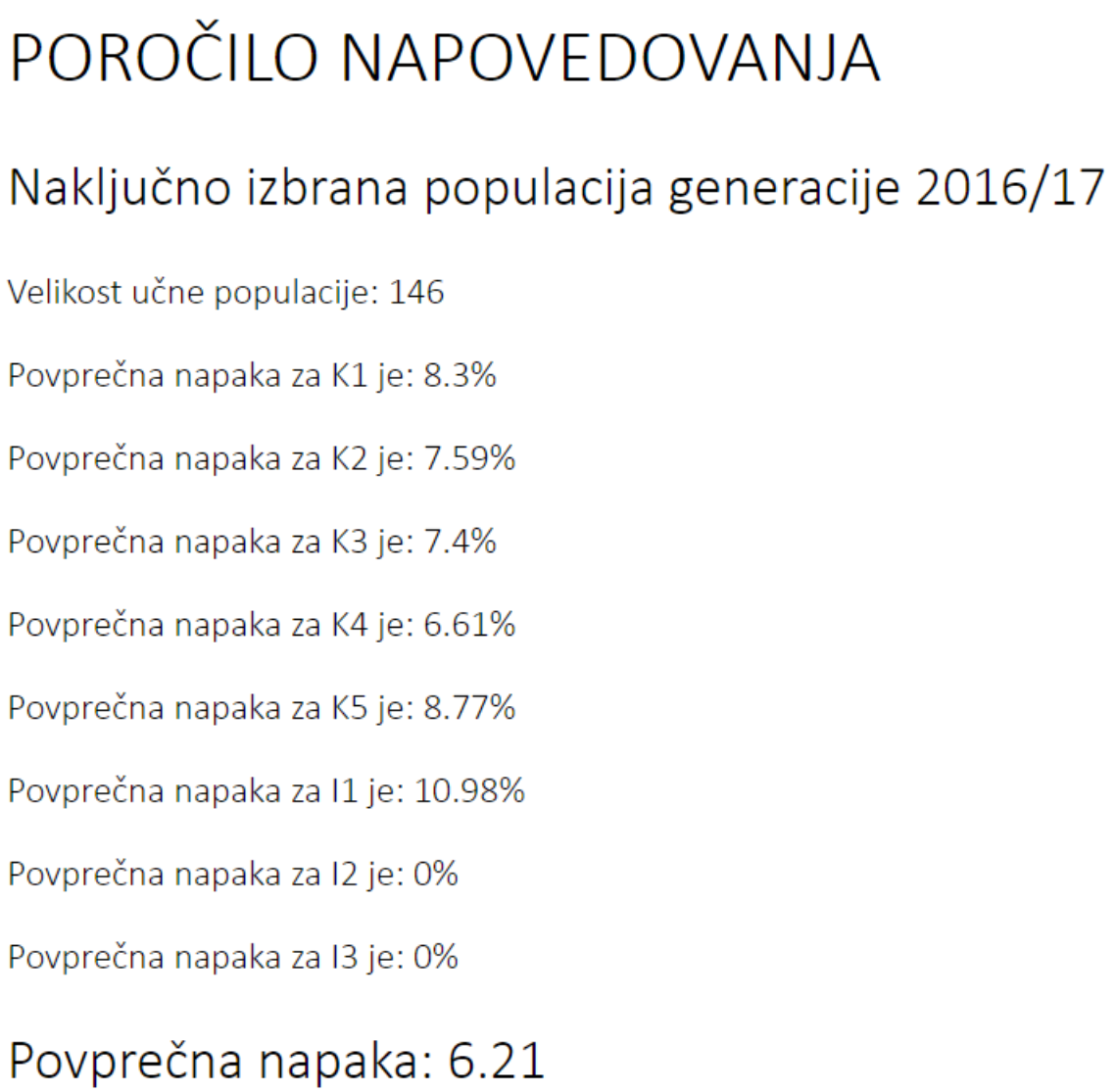
4.13.1 Poročilo 1



Slika 26: Končno poročilo napovedovanja ocen nad generacijo 2015/16.

Poročilo 2 (27) prikazuje končni rezultat napovedovanja ocenjevanj nad generacijo 2016/17 po vseh preračunih in končni obdelavi podatkov. Izpitna roka 2 in 3 imata pri napaki vredost 0%, ker podatka še nista na voljo.

4.13.2 Poročilo 2



Slika 27: Končno poročilo napovedovanja ocen nad generacijo 2016/17.

5 Zaključek

Rezultat napovedovanja se je izboljšal od napovedi nad generacijo 2015/16 na generacijo 2016/17, ampak ne drastično. Razlaga za takšen izid je pomankanje podatkov. Študentje niso zelo aktivni pri reševanju eQuiz nalog, ali pa le z ugibanjem skušajo odkriti pravi odgovor (izgleda, da jim bo potrebno v novi verziji to preprečiti - morda tako, da jih aplikacija opozori, da naj tega ne počnejo, ali z vsiljeno časovno zamudo pri preverjanju).

Posledica tega so nepopolni in zelo naključni rezultati pri preračunanju koeficientov znanja, ki skoraj izgledajo popolnoma naključni. Zaradi tega smo se poskušali čim manj zanašati na podatke ostalih študentov in zaradi tega so podatki ostalih študentov uteženi. Uteži so pa bile izbrane s poskušanjem raznih kombinacij. V kasnejših izdajah bodo koeficienti dinamično dodeljeni s strani sistema, ki se bo učil na podlagi rezultatov.

Aplikacija je torej namenjena študentom, ki so zainteresirani v povratne informacije svojega znanja ter želijo dodatno motivacijo na podlagi svojega sprotnega dela.

S tem projektom smo pridobil veliko novega znanja in izkušenj. Spoprijeli smo se s problemi zbiranja in filtriranja velikih količin podatkov, ter obdelavo podatkov nad izbranimi vzorci populacije. Prav tako je bilo potrebno razviti algoritme, ki upoštevajo omejitve podatkov in posledice pomankanja podatkov za čim natančnejši končni rezultat. Prav tako je bila odločitev, da smo razvili svoje algoritme pravilna, saj je ta problem zelo specifičen, napovedovanje je pa implementirano nad majhnim številom ocenjevanj za vsakega študenta posebej. Algoritmi so zasnovani na konceptih, ki smo jih spoznali na predavanjah, vendar so dodatno prilagojeni našim potrebam.

Osebno mislim, da nam je projekt zelo dobro uspel, saj so napovedi na podlagi trenutnega stanja informacij precej natančni tudi za študente, ki ne rešujejo eQuiz nalog.

Literatura

Aleksandar Jurišić - Verjetnost in statistika (`https://ucilnica.fri.uni-lj.si/pluginfile.php/3595/mod_page/content/34/vs1503.pdf`)