Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Računarstvo usluga i analiza podataka

SEMINARSKI RAD

„Predviđanje rezultata "kružić-križić“

Antonio Došlić

Ivan Klešić

Osijek, 2018.

Sadržaj

[1 Uvod 1](#_Toc517641997)

[1.1 Zadatak 1](#_Toc517641998)

[2 Opis problema 2](#_Toc517641999)

[2.1 Korišteni podaci 2](#_Toc517642000)

[2.1.1 Korišteni postupci početne analize podataka 3](#_Toc517642001)

[2.2 Korišteni postupci strojnog učenja 5](#_Toc517642002)

[3 Opis programskog rješenja 6](#_Toc517642003)

[3.1 Model strojnog učenja 6](#_Toc517642004)

[3.2 Način korištenja API-ja 10](#_Toc517642005)

[3.3 Klijentska aplikacija 10](#_Toc517642006)

[4 Zaključak 13](#_Toc517642007)

[5 Literatura 14](#_Toc517642008)

# Uvod

Križić-kružić je popularna igra namjenjena za dva igrača. Prvi igrač upisuje X u polje za igru, a drugi igrač O. Cilj je skupiti tri uzastopna simbola kako bi igrač pobjedio. Vrlo se lako može uočiti kada jedan od igrača pobjedi budući da se igra na polju s tri retka i tri stupca. Iako vro jednostavno za čovjeka, računalo ne uočava tako lako rezultat igre.

## Zadatak

Zadatak ovog projekta je bio napraviti program koji će predvidjeti rezultat igre križić-kružić na osnovu nekoliko atributa. Potrebno je preuzeti skup podataka, jedan dio iskoristiti za stvaranje klasifikacijskog modela, a drugi dio za treniranje i vrednovanje.

Nakon toga treba napraviti API koji će korisnicima ponuditi funkcionalnost našeg modela i izraditi klijentsku aplikaciju koja će koristit API.

# Opis problema

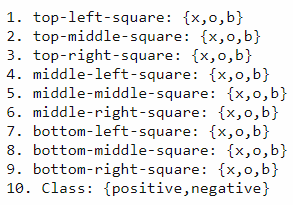
Problem koji rješavamo je taj da računalo naučimo kako pravilno protumačiti rezultat igre. Najbolji način za to je strojno učenje. Zbog specifičnosti podataka koje koristimo, predviđanje rezultata smo ograničili na igrača X, odnosno predviđamo je li igrač X pobjedio ili ne.



Slika 1. Izgled polja odigrane igre

## Korišteni podaci

Korišteni podaci preuzeti su s UCI repozitorija [[1]](#prvi). Podaci se sastoje od 958 instanci, devet atributa i klase. Atributi predstavljaju devet polja za igru.



Slika 2. Formalni opis podataka

Slika 2 prikazuje popis atributa i klase. U svrhu pojednostavljenja, atributi su poredani i opisno imenovani. Podaci atributa i klase nalaze se unutar vitičastih zagrada. Vidimo da podaci za atribute su x, o ili b. Ti podaci predstavljaju što se nalazi na određenom polju: simbol X, simbol O ili b (eng. blank-prazno polje). Podaci klase koja predstavlja rezultat su positive ili negative. Positive znači da je igrač X pobjedio, a negative da igrač X nije pobjedio. Vrijednosti klase smo zamjenili s brojevima 1 i 0 zbog pojednostavljivanja.

### Korišteni postupci početne analize podataka

Zbog specifičnosti podataka napravili smo početnu analizu podataka. Prvo smo napravili omjere klasa.



Slika 3. Omjer klasa skupa podataka

Slika 3 predstavlja omjer klasa. Od ukupno 958 instanci podataka njih 626 rezultira pozitivno (pobjeda igrača X), a ostalih 332 negativno (igrač X nije pobjedio).

Nakon toga smo metodama deskriptivne statistike podatke podijelili u devet razreda koje možemo vidjeti na slici 2. Svaki razred predstavlja određeno polje igre koje možemo iščitati iz imena razreda.

|  |  |  |
| --- | --- | --- |
| 1. top-left-square  11.jpg | 2. top-middle-square  22.jpg | 3. top-right-square  11.jpg |
| 4. middle-left-square  22.jpg | 5. middle-middle-square  55.jpg | 6. middle-right-square  22.jpg |
| 7. bottom-left-square  11.jpg | 8. bottom-middle-square  22.jpg | 9. bottom-right-square  11.jpg |

Tablica 1. Prikaz rezultata deskriptivne statistike

Tablica 1 prikazuje rezultate deskriptivne statistike po razredima. Svaka slika prikazuje učestalost pojavljivanja određenog simbola u tom razredu, te dodatno prikazuje koliko od tih slučajeva završava pozitivno, odnosno negativno (boje su nasljeđene sa slike 3). Iz dobivenih rezultata uočavamo da su razredi 1, 3, 7 i 9 isti, te razredi 2, 4, 6 i 8 također isti. Imajući to u vidu, možemo od devet razreda napraviti 3 različita razreda koja će predstavljati sve dosadašnje razrede.

|  |
| --- |
| 1. Razred X (1, 3, 7 i 9)  11.jpg |
| 2. Razred Y (2, 4, 6 i 8)  22.jpg |
| 3. Razred Z (5)  55.jpg |

Tablica 2. Prikaz novonastalih razreda

Tablica 2 prikazuje novnonastale razrede s njihovim novim imenima i rednim brojevima razreda koji čine novonastale razrede.

|  |  |  |  |
| --- | --- | --- | --- |
| Redni broj razreda | Naziv razreda | Frekvencija razreda fi | Relativna frekvencija razreda |
| 1. | X | 4 | 4/9 |
| 2. | Y | 4 | 4/9 |
| 3. | Z | 1 | 1/9 |

Tablica 3. Formalni prikaz novonastalih razreda

Tablica 3 prikazuje učestalost novonastalih razreda od ukupnog broja razreda.

Iako nam ova tablica jasnije dočarava odnos među podatcima, u njoj su se neke informacije izgubile. Na primjer, iz nje očitavamo da su u prvom razredu četiri podrazreda, ali ne znamo koja su to četiri podrazreda niti koji sve podaci pripadaju tom razredu. Gubitak je neznatniji što je uzorak veće duljine, a razredi uži. Za valjanost nekih statističkih zaključivanja često se traži da frekvencija svakog razreda bude barem 5, što ovo naše grupiranje ne zadovoljava.

## Korišteni postupci strojnog učenja

Strojno učenje je vrsta obrade podataka gdje se računalo može istrenirati da samo donosi zaključke i odluke bez da je to eksplicitno isprogramirano. Postoji nadzirano, djelomično nadzirano, nenadzirano i učenje s povratnom vezom. Postoje razredi učenja, a to su klasifikacija, regresija, rangiranje, grupiranje i smanjenje dimenzionalnosti.

U projektu je korišeno nadzirano učenje. To znači da su računalu dani ulazi i izlazi, te računalo na osnovu njih uči. Kao razred učenja korištena je klasifikacija. To je postupak dohvaćanja uzoraka i smještanje tog uzorka u odgovarajuću kategoriju, odnosno riječ je o dodjeli oznake klase novoj ulaznoj vrijednosti.

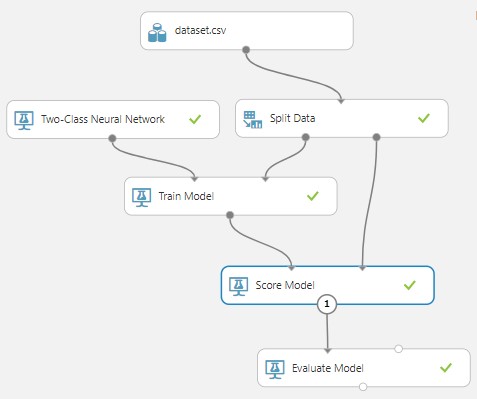
Metode koje obavljaju klasifikaciju nazivaju se klasifikatorima, a neki od najčešćih su algoritam najbližih susjeda (engl. k-nearest neighbours, kNN), umjetne neuronske mreže (engl. artificial neural networks, ANN) i stroj s vektorima podrške (engl. support vector maching, SVM). Svaki klasifikator najprije je potrebno trenirati, odnosno potrebno ga je naučiti kako razlikovati uzorke različitih kategorija (klasa). Formalnije, potrebno izgraditi klasifikacijski model na temelju kojeg će se donositi općenite odluke za buduće prikupljene podatke nepoznatih oznaka [[6]](#sesti).

# Opis programskog rješenja

Zadatak je bio napraviti program koji će na osnovu unesenih parametara kao izlaz vratiti predviđeni rezultat igre. Program je napravljen u C#, dok je za strojno učenje korišten Azure Machine Learning Studio. Korisnik unosi parametre u program. Program se spaja na Azure, te šalje podatke. Nakon što dobije odgovor u obliku stringa, korisniku se ispisuje rezultat.

## Model strojnog učenja

Model strojnog učenja vrlo je sličan onome što smo radili na laboratorijskim vježbama. Prvo smo učitali podatke u AMLS (Azure Machine Learning Studio), a nakon toga sastavili model. Odabrali smo 75% podataka za treniranje i 25% za evaluaciju.



Slika 4. Prikaz Azure ML modela

Slika 4 prikazuje korištenje Two-Class Neural Network klasifikatora, no to nije jedini klasifikator s kojim smo testirali naš model. Koristili smo sve ponuđene dvoklasne klasifikatore koje nam je AMLS ponudio i uspoređivali rezultate kako bi odredili koji klasifikator najbolje odgovara našem slučaju.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Naziv algoritma | True Positive | False Negative | False Positive | True Negative | Accuracy | Precision | Recall | F1 Score |
| Two-Class Averaged Perceptron | 155 | 2 | 4 | 78 | 0.975 | 0.975 | 0.987 | 0.981 |
| Two-Class Bayes Point Machine | 157 | 0 | 4 | 78 | 0.983 | 0.975 | 1 | 0.987 |
| Two-Class Boosted Decision Tree | 157 | 0 | 1 | 81 | 0.996 | 0.994 | 1 | 0.997 |
| Two-Class Decision Forest | 150 | 7 | 3 | 79 | 0.958 | 0.980 | 0.955 | 0.968 |
| Two-Class Decision Jungle | 147 | 10 | 4 | 78 | 0.941 | 0.974 | 0.936 | 0.955 |
| Two-Class Local-Deep SVM | 152 | 5 | 10 | 72 | 0.937 | 0.938 | 0.968 | 0.953 |
| Two-Class Logistic Regression | 155 | 2 | 7 | 75 | 0.962 | 0.957 | 0.987 | 0.972 |
| Two-Class Neural Network | 154 | 3 | 3 | 79 | 0.975 | 0.981 | 0.981 | 0.981 |
| Two-Class SVM | 141 | 16 | 18 | 64 | 0.858 | 0.887 | 0.898 | 0.892 |

Tablica 4. Prikaz rezultata svih korištenih klasifikatora

Tablica 4 prikazuje sve klasifikatore i rezultate dobivene testiranjem tih klasifikatora na našem modelu. Nakon uspoređivanja rezultata, klasifikator Two-Class Boosted Decision Tree je pokazao najbolje rezultate vidljive u tablici 4. Ovaj klasifikator je imao samo jednu krivu procjenu (False Negative + False Positive) od 239 instanci podataka koje je obradio tijekom testiranja. Vidljivo je da su svi ostali parametri vrlo visoki. Parametar "Accuracy" nam govori preciznost predviđanja rezultata, ali to nam ne znači puno jer nam klase nisu isbalansirane (slika 3). Parametar "Precision" nam govori koliko je točno pogođenih slučajeva da će igrač X pobjediti (True Positive) od ukupnog broja pobjeda igrača X (True Positive + False Positive). Parametar "Recall" nam govori omjer True Positive i (True Positive + False Negative). Zadnji parametar F1 Score nam sažima dva parametra (Precision i Recall), te je zbog toga jako dobar pokazatelj uspješnosti modela. Izračunamo ga formulom: F1 = 2 (precision x recall) / (precision + recall).

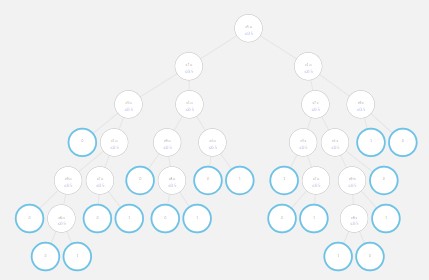
„Decision tree learning“ (učenje sa stablom odlučivanja) predstavlja konstruiranje stabla odlučivanja iz podataka za treniranje koji su označeni klasama. Stablo odlučivanja je struktura koja podsjeća na „flow-chart“ gdje svaki interni čvor označava test na nekom atrubutu, svaka grana označava ishod testa, a svaki terminalni čvor (list stabla) sadrži klasnu oznaku. Prvi čvor u stablu se naziva korijenski čvor.

**„**Boosted trees“ ili poboljšana stabla je skupina stabala čija se svaka nova instanca kreira tako da ispravlja pogreške u prethodnim instancama koje su krivo modelirane. To znači da bi algoritam trebao raditi bolje sa povećanjem broja konstruiranih stabala. Međutim, prevelik broj stabala može dovesti do problema koji nazivamo „overfitting“ gdje je model istreniran u toj mjeri da jako precizno predviđa ako se koriste podaci za trening, ali ne toliko precizno ako se koriste podaci za test. „Two-class boosted decision tree“ algoritam ovaj problem rješava na način da na kraju ne uzima u obzir samo rezultat finalnog stabla nego uzima rezultate iz svakog stabla te računa prosjek. Ovo u velikoj mjeri smanjuje efekt overfitting-a te i dalje nudi precizna predviđanja.

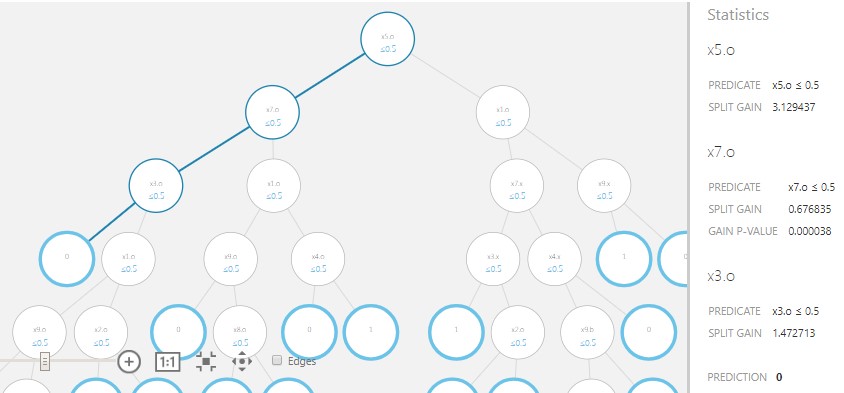
Parametar "Maximum Number of Leaves per Tree" ili maksimalan broj listova po stablu označava maksimalni broj razdvajanja čvorova (kreiranja „novog pravila“). Ranije nastali čvorovi u stablu su značajniji od čvorova koji nastaju kasnije. Povećanjem ovog paramentra se povećava šansa za overfitting.

Parametar "Minimum Number of Samples per Leaf Node" ili minimalni broj uzoraka po listu predstavlja razinu značaja koja je potrebna za razdvajanje čvorova. Povećavanje ovog parametra rezultira stabilnim predikcijama (više slučajeva je obuhvaćeno), a smanjivanje rezultira preciznijim predikcijama.

Parametar "Learning Rate" ili stopa učenja omogućava definiranje razlike između svakog drveta u nizu. Stopa učenja određuje koliko brzo trening konvergira prema optimalnom rješenju.



Slika 5. Prikaz stabla odlučivanja

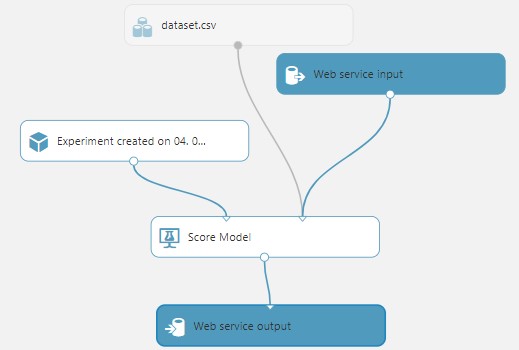


Slika 6. Prikaz stabla odlučivanja

Slika 6 prikazuje stablo odlučivanja. S desne strane slike vidimo atribute koji se promatraju u čvorovima stabla. U prvom čvoru promatrali smo x5, odnosno peto polje igre je li O ili ne. U drugom čvoru sedmo polje je li O, a u trećem čvoru treće polje je li O. Nakon ta tri čvora dobili smo predviđanje 0 što nam predstavlja klasu da igrač X nije pobjedio. Na slici su ta tri čvora povezana plavom bojom s rezultirajućim čvorom.

## Način korištenja API-ja

Nakon što je model istreniran i pruža zadovoljavajuće rezultate, potrebno ga je staviti na Azure Web Service.



Slika 7. Prikaz modela koji ćemo postavit na web servis

Time postupkom je napravljen API preko kojega se može putem Weba koristiti istrenirani model. Također je moguće preuzeti programski kod za klijentsku aplikaciju koji koristi taj API.

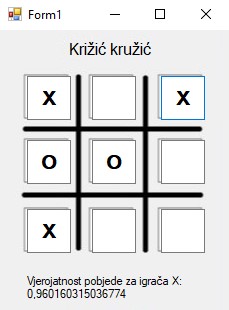
Uz taj programski kod dokumentirani su input i output. Input se sastoji od naziva stupaca i jednog ili više seta podataka. Kao output se vraćaju nazivi stupaca zajedno s tipovima podataka, unešenim vrijednostima i dva nova podatka, scored labels i scored probabilities. Scored labels nam pokazuje vrijednost koju je model predvidio za rezultat igre, a scored probabilites pokazuje vrijednost ovisno o rezultatu. Ako je igrač X pobjedio, scored probabilites će biti blizu vrijednosti 1, a u suprotnom blizu vrijednosti 0.

## Klijentska aplikacija

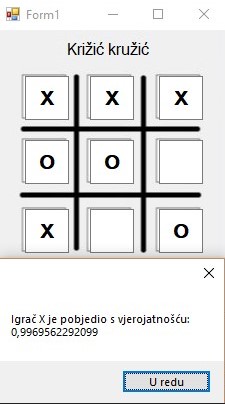
Kostur aplikacije je bio dan od strane Azure-a u C#, te smo se iz tog razloga odlučili napraviti korisničku aplikaciju u C# programskom jeziku. Aplikacija predstavlja korisničko sučelje preko kojeg korisnik komunicira sa Azure-om igrajući igru križić kružić.



Slika 8. Izgled klijentske aplikacije



Slika 9. Izgled klijentske aplikacije



Slika 10. Izgled klijentske aplikacije

Sučelje se sastoji od devet polja koja predstavljaju polja za igru križić kružić i prikaza rezultata ispod tih polja vidljivih na slikama 8, 9 i 10. Aplikacija je zamišljena da se nakon svakog unosa polja ispisuje vjerojatnost pobjede igrača X kao što slike 8 i 9 prikazuju. Slika 10 prikazuje prekid igre s ispisom vjerojatnosti. Prekid igre se događa ukoliko jedan od igrača pobjedi ili nema više praznih polja za unos.

# Zaključak

Prilikom izrade projektnog zadatka koristili smo C# programski jezik i Microsoft Azure ML Studio. Nakon umetanja podataka u Microsoft Azure i kreiranja ML modela, dobili smo kostur API-a kojeg smo oblikovali na svoj način tako da korisniku preko jednostavnog sučelja omogućimo komunikaciju s Azure-om. Komunikacija s Azure-om se uspostavlja nakon svakog unosa u polje za igru, a kao povratnu informaciju dobivamo predikciju pobjede igrača X.

# Literatura

Programsko rješenje je dostupno na linku: <https://github.com/adoslic/RUAP-projekt>

1. <http://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame>
2. <http://www.ijiet.org/papers/314-k010.pdf>
3. <https://www.kdnuggets.com/2017/09/neural-networks-tic-tac-toe-keras.html>
4. <https://datahub.io/machine-learning/tic-tac-toe-endgame>
5. <https://docs.microsoft.com/en-us/azure/machine-learning/studio/interpret-model-results>
6. <https://breaking-bi.blogspot.com/2016/11/azure-machine-learning-classification.html>
7. <https://en.wikipedia.org/wiki/Decision_tree_learning>
8. <https://www.c-sharpcorner.com/article/two-class-boosted-decision-tree/>
9. Laboratorijske vježbe