Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Vika Lampret

Spletna aplikacija za prijavo na objavljene zaposlitve

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Prijavljanje na prosta delovna mesta je za kandidate lahko problematično zato, ker se lahko prijavijo na delovno mesto, kjer glede na svoje karakteristike nimajo možnosti, da bi bili sprejeti. Zasnujte spletno aplikacijo, ki bo omogočala sprejem prijav in hkrati z uporabo klasifikacijskih metod kandidatu sproti podala evaluacijo, če ima glede na svoje karakteristike možnosti, da bo na dano delovno mesto sprejet.

Mentorju doc. dr. Roku Rupniku se lepo zahvaljujem za potrpežljivost, dostopnost in vso pomoč, ki mi jo je nudil tekom pisanja diplomske naloge.

Zahvaljujem se tudi bratu Leonu za vzpodbudo in podporo med študijem.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	Hyper Text Markup Language	jezik za označevanje nadbesedila
k-NN	k-nearest neighbors	k-najbližjih sosedov
SVM	support vector machine	metoda podpornih vektorjev
CSS	Cascading style sheets	kaskadne stilske podloge
SQL	Structured query language	strukturirani povpraševalni jezik
DQL	Data query language	jezik za rokovanje s podatki
DDL	Data description language	jezik za definiranje podatkov
DCL	Data control language	jezik za nadzor nad podatki
TCL	Transaction control language	jezik za nadzor nad transakcijami
DML	Data manipulation language	jezik za manipulacijo podatkov
ER model	Entity–relationship model	entitetno-relacijski model
EER model	Enhanced entity–relationship model	razširjen entitetno-relacijski model

Povzetek

Naslov: Spletna aplikacija za prijavo na objavljene zaposlitve

Avtor: Vika Lampret

V diplomski nalogi smo se posvetili problemu zaposlovanja, ko je število iskalcev zaposlitve za nekatere poklice zelo veliko. Posledično so zaposleni ob razpisu prostega delovnega mesta zelo obremenjeni in težko opravljajo svoje redno delo. V večini primerov ima delodajalec nek vzorec lastnosti, ki jih išče pri bodočem zaposlencu. Običajno so za delodajalca najbolj pomembne *starost*, *izobrazba* in predvsem *delovne izkušnje*. Zato smo se v okviru diplomske naloge odločili ustvariti spletno aplikacijo, ki bi delodajalcu omogočila zožitev izbora primernih iskalcev zaposlitve. Za zožitev izbora smo uporabili klasifikacijske metode, ki na podlagi atributov *starost*, *izobrazba* in *delovne izkušnje* določi, če je iskalec zaposlitve primeren za to delovno mesto.

Ključne besede: spletna aplikacija, statistična klasifikacija, zaposlovanje.

Abstract

Title: Web application to support applications for jobs

Author: Vika Lampret

In the graduation thesis, we will focus on the following problem of the employment procedure. For certain professions, there is a high number of job applicants. Consequently, when a job vacancy is advertised, employees are overburdened and it is difficult to perform their own regular work. In most cases, the employer has a sample of the desirable properties he is looking for in a future employee. Usually, the most important ones are age, education, and above all, work experience. Therefore, as the objective of the thesis, we decided to create a web application that allows the employer to narrow down the list of suitable applicants. For this purpose, we used classification methods, which, based on attributes *age*, *education* and *work experience*, determine if the applicant is suitable for the given job.

Keywords: web application, statistical classification, employment procedure.

Poglavje 1

Uvod

1.1 Motivacija za izdelavo aplikacije

Znano je, da za določene poklice zelo primanjkuje delovnih mest. Ob razpisu takega prostega delovnega mesta je pričakovati zelo veliko prijav. Veliko kandidatov lahko ne ustreza vsem pogojem, vendar se določeno podjetje kljub vsemu odloči za enega izmed njih. Primer: običajno so zelo iskane delovne izkušnje, vendar je primeren kandidat lahko tudi zelo priden študent, ki je vse izpite opravil pravočasno in v predvidenem roku diplomiral. Po drugi strani pa delodajalcem ni sprejemljiv delavec, ki je star 58 let in ima 40 let delovnih izkušenj, ker bi lahko bil tik pred upokojitvijo.

1.2 Cilji

V okviru izdelave diplomske naloge smo se ukvarjali s problemom prevelikega števila prijavljenih na razpisano delovno mesto. Odločili smo se ustvariti spletno aplikacijo, ki delodajalcem omogoči avtomatsko izločitev manj primernih kandidatov in jih s tem razbremeni branja vseh poslanih življenjepisov. Pri izdelavi smo uporabili klasifikacijske metode, s katerimi pred samo prijavo na delovno mesto preverimo, če je iskalec zaposlitve primeren. To je najlažje storiti z uvozom podatkov o prejšnjih prijavah in dejanskih rezultatih zapo-

slitve. Omenjena množica bi predstavljala učno množico, na podlagi katere bi nato klasifikacijska metoda odločila ali bi bil trenutni iskalec zaposlitve primeren.

1.3 Kratek pregled poteka izdelave in težav s katerimi smo se srečali

Najprej smo se lotili izbire primernega programskega jezika, ogrodij, knjižnic in razvojnega okolja. Želeli smo preizkusiti nove tehnologije, s katerimi se tekom študija nismo seznanili. Po končani izbiri tehnologij smo se lotili pregleda primernih klasifikacijskih metod.

Vse metode smo implementirali v program in jih tudi testirali na izmišljeni učni množici. Srečali smo se s problemom, kako vnesti v program in zgraditi učno množico, da bi se čimbolj prilegala zahtevam razpisanega delovnega mesta. Ker ima vsako delovno mesto svoje zahteve, smo to prepustili uporabniku aplikacije z vnosom Excel datoteke s podatki o preteklih prijavah in njihovih razultatih zaposlitve.

Tekom razvoja aplikacije smo se srečali tudi s problemom izbire klasifikacijske metode. Ker ima vsaka metoda svoje prednosti in slabosti ter nimamo vnaprej znane učne množice, je dokončna izbira metode težavna in ni nujno, da je optimalna rešitev. Zato smo implementirali tri metode in izbiro prepustili uporabniku aplikacije. Pogosta težava, ki bi se posledično lahko pojavila, je uporabnikovo nepoznavanje delovanja posameznih metod.

Dodali smo poglede (angl. view) za pregled prostih delovnih mest, vpis uporabnika, oddajo novega oglasa, urejanje obstoječega oglasa, vnos potrebnih podatkov za izvršitev klasifikacijske metode in prijavo na prosto delovno mesto. Bolj podrobno se bomo z razvojem aplikacije spoznali v poglavju Implementacija aplikacije.

1.4 Pregled poglavij

Podrobnejšega pregleda se bomo lotili v naslednjih poglavjih:

- Uporabljene tehnologije: opis uporabljenih programskih jezikov, ogrodij, knjižnic in razvojnega okolja.
- Pregled klasifikacijskih metod: značilnosti in delovanje uporabljenih metod.
- Implementacija aplikacije: opisali bomo vzpostavitve novega praznega Djangovega projekta in vzpostavitve nove baze, povezave z bazo, implementacijo klasifikacijskih metod, vnos in branje podatkov iz baze, itd.
- Zaključek: opiše glavne ugotovitve, težave, rešitve in rezultate s katerimi smo se srečali tekom razvoja aplikacije.

Poglavje 2

Uporabljene tehnologije pri izgradnji aplikacije

V tem poglavju bomo na kratko opisali uporabljene programske jezike, ogrodja, knjižnice in razvojna okolja, ki smo jih uporabili tekom razvoja aplikacije.

2.1 Python

Python je visokonivojski programski jezik. Ustvaril ga je Guido van Rossum leta 1991 [27]. Njegove značilnosti so:

- brezplačno dostopen,
- podpira module in pakete,
- uporaba zamikov, namesto zavitih oklepajev ali ključnih besed za označitev bloka kode,
- sintaksa, ki omogoča programiranje s krajšimi izrazi [21] [27].

2.2 HTML

HTML je jezik za izdelavo spletnih strani in spletnih aplikacij. Skupaj z jezikoma CSS in JavaScript sestavljajo temelje tehnologij za svetovni splet.

Uporablja se ga z zaporedjem ukazov, ki se nato znotraj brskalnika prevedejo in prikažejo v vizualni obliki. Ustvarjen je bil leta 1993. HTML elementi so določeni z oznakami, napisanimi s kotnimi oklepaji. Običajno oznake zapišemo v parih (npr. $< button >< /button >$), kjer prva oziroma začetna oznaka predstavlja začetek elementa, druga oziroma končna oznaka pa konec elementa. Znotraj teh dveh oznak so navedene lastnosti, klici itd. Poznamo tudi prazne elemente, ki so navedeni samo z eno oznako (npr. $< /br >$) [11] [12].

2.3 SQL

SQL (Structured Query Language) je standardni povpraševalni jezik, namenjen delu s podatkovnimi bazami [24].

SQL vsebuje več tipov ukazov:

- Jezik za poizvedbe o podatkih (angl. data query language ali DQL): namenjen izvrševanju poizvedb. Omenjeni stavki ne vplivajo na podatke znotraj baze.

- Jezik za definiranje podatkov (angl. data definition language ali DDL): namenjen ustvarjanju in oblikovanju baze, spreminjanju strukture in brisanju.

- Jezik za manipuliranje podatkov (angl. data manipulation language): namenjen dodajanju, posodabljanju in brisanju vrstic tabele.

- Jezik za nadzor nad podatki (angl. data control language ali DCL): namenjen opredeljevanju in nadzoru dostopa do podatkov.

- Jezik za nadzor nad transakcijami (angl. Transaction Control Language ali TCL): namenjen sprožanju in preklicu izvajanja transakcij [25].

2.4 Django

Django je brezplačno odprtokodno visokonivojsko ogrodje za razvoj spletnih aplikacij, napisano v jeziku Python. Je zbirka modulov, ki nam omogoča hitrejši in lažji razvoj [6] [7].

2.5 Knjižnica Scikit-learn

Scikit-learn je brezplačna, odprtokodna knjižnica strojnega učenja za programski jezik Python. Vsebuje več klasifikacijskih, regresijskih in clustering algoritmov. Deluje v sodelovanju s knjižnicama NumPy in SciPy [23].

2.6 Bootstrap

Bootstrap je odprtokodno HTML, JavaScript in CSS ogrodje, ki se ga uporablja za izdelavo uporabniškega vmesnika spletnih strani ali spletnih aplikacij. Podpira tudi HTML5 in CSS3. Ustvarili so ga razvijalci in oblikovalci iz podjetja Twitter leta 2010. Testirano in podprto je znotraj večine sodobnih brskalnikov: zadnja različica Safari, Google Chrome, Firefox in Internet Explorer 8+ [3] [4].

2.7 Visual studio

Visual studio je integrirano razvojno okolje podjetja Microsoft. Namenjeno je razvoju spletnih aplikacij, spletnih strani, programov namenjenih operacijskemu sistemu Windows ter aplikacij na osnovi .NET ogrodja [26].

2.8 MySQL

MySQL je odprtokodni sistem za upravljanje s podatkovnimi bazami. Za delo z bazami uporablja jezik SQL. Napisan je v C in C++ jeziku. MySQL

deluje po principu odjemalec-strežnik in deluje na več operacijskih sistemih (Linux, MacOS, Microsoft Windows, IRIX, FreeBSD, ...) [18] [19].

Poglavje 3

Pregled klasifikacijskih metod

Klasifikacija se ukvarja s problemom uvrstitve opazovanega objekta v eno izmed predpisanih množic, glede na lastnosti objekta. Uvrstitev se izvede s pomočjo učne množice, ki vsebuje znane objekte in njihove pripadnosti množicam.

Poznamo dva tipa učnih metod:

- leno učenje (angl. lazy learning), kjer sistem odlaša z učenjem in se klasifikacijski model generira šele po prejemu nove poizvedbe;
- željno učenje (angl. eager learning), kjer se na podlagi učne množice klasifikacijski model generira na samem začetku, pred poizvedbami [5].

V okviru izdelave diplomske naloge smo se srečali s težavo izgradnje učne množice, da bi model kar se da natančno napovedoval primernost zainteresirane osebe. Zamislili smo si, da podjetje ob objavi oglasa za zaposlitev naloži spisek vseh v preteklosti prijavljenih iskalcev zaposlitve ter rezultatov sprejetja novega delavca. Omenjen seznam oseb predstavlja učno množico.

Končni množici sta v našem primeru:

- primeren iskalec dela glede na zahteve podjetja,
- neprimeren iskalec dela glede na zahteve podjetja.

Na podlagi učne množice ob prijavi novega iskalca zaposlitve izvedemo eno izmed klasifikacijskih metod, ki določi ali je primeren za oglaševano delovno mesto. V nadaljevanju si bomo pogledali tri klasifikacijske metode.

3.1 Metoda odločitvenih dreves

Metoda odločitvenih dreves je napovedovalni model strojnega učenja. Sodi med željni tip klasifikacijskih algoritmov. Uporablja se za reševanje klasifikacijskih problemov (napovedovanje pripadosti razredu) in regresijskih problemov (napovedovanje vrednosti). Metoda napove vrednost odvisne spremenljivke (ki je v naši situaciji primernost za zaposlitev) novega vzorca, na podlagi množice večih predhodno znanih podatkov ter vrednosti njihovih atributov (ki so v naši situaciji starost, izobrazba in delovna doba). Atribut, ki ga napovedujemo, je odvisna spremenljivka, ki je odvisna od vrednosti vseh ostalih atributov (neodvisnih spremenljivk).

Metoda odločitvenih dreves je enostaven algoritem v primerjavi z ostalimi klasifikacijskimi algoritmi. Deluje tako, da zgradi odločitveno drevo na podlagi učne množice.

Ustvarjanje drevesa poteka po algoritmu [9], [8]:

1. Postavimo najbolj primeren atribut na vrh drevesa. Primernost atributa smo ocenili s pomočjo Gini indeksa (glej spodaj!).

2. Razdelimo učno množico na dve podmnožici (leva in desna veja vozlišča). To storimo tako, da določimo mejo za vrednost izbranega atributa, in v levo vejo damo elemente, ki imajo atribut manj od meje, v desno vejo pa elemente, ki imajo atribut več od meje.

3. Ponavljamo 1. in 2. korak, dokler ne pridemo do listov drevesa. Vozlišče je list, ko imajo vsi njegovi elementi enako vrednost odvisne spremenljivke (v našem primeru so torej v listu sami primerni ali sami neprimerni iskalci zaposlitve).

Ko želimo napovedati vrednost odvisnega atributa novega vzorca na podlagi neodvisnih atributov, se sprehodimo čez že generirano odločitveno drevo in primerjamo vrednosti neodvisnih atributov novega vzorca z mejami v vozliščih drevesa. Na podlagi sprehoda pridemo do lista drevesa, kjer dobimo tudi napovedano vrednost odvisnega atributa novega vzorca [22, str. 2].

Za izbiro primernega neodvisnega atributa za posamezno vozlišče je bil uporabljen dodatni kriterij. Pri izdelavi diplomske naloge smo uporabili gini indeks, katerega si bomo tudi podrobneje pogledali.

Gini indeks je namenjen merjenju, kako pogosto bo naključno izbrani element narobe identificiran. Atribut z nižjim gini indeksom se najprej uporabi pri gradnji odločitvenega drevesa. Izračuna se po formuli [16, str. 63, formula (5.5)]

$$\text{Gini} = 1 - \sum_i p_i^2,$$

kjer je p_i delež vseh objektov, ki pripadajo i-temu ciljnemu razredu. V našem primeru je $i \in \{0, 1\}$, saj ima odvisni atribut le 2 možni vrednosti.

Za potrebe diplomske naloge smo si izmislili učno množico vzorcev z znanimi odvisnimi in neodvisnimi atributi. Na omenjeni množici bomo zgradili odločitveno drevo in na njem tudi poiskali vrednost odvisnega atributa za nek nov vzorec.

V diplomski nalogi smo uporabili tri neodvisne atribute:

1. starost
2. izobrazba
   (a) osnovna šola - 2. stopnja
   (b) poklicna šola - 4. stopnja
   (c) srednja šola - 5. stopnja
   (d) višješolska izobrazba - 6. stopnja

(e) visokošolska izobrazba - 7. stopnja

(f) univerzitetna - 7. stopnja

(g) magisterij - 8. stopnja

(h) doktorat - 9. stopnja

3. delovna doba

Odvisna spremenljivka bo atribut Primernost zaposlitve, kjer bo vrednost 1 predstavljala primernega in vrednost 0 neprimernega kandidata.

Za začetek si izmislilmo učno množico, in na njej z uporabo kriterija gini indeks zgradimo odločitveno drevo. Tako si bomo podrobneje pogledali delovanje metode odločitvenih dreves.

Starost	Izobrazba	Delovna doba	Primernost zaposlitve delavca
18	2	0	0
23	6	0	1
30	4	0	0
26	8	2	1
26	4	7	1
30	9	3	1
45	2	5	0
45	2	25	1
52	6	15	0
55	6	35	0
60	9	37	0

Tabela 3.1: Primer učne množice za izgradnjo odločitvenega drevesa

Gini indeks za vse tri atribute:

- pri atributu *Starost* smo za mejo izbrali naključno vrednost 48.5, na podlagi katere vzorce razdelimo v dve množici.

Starost > 48.5	Primernost
52	0
55	0
60	0

Z upoštevanjem pogoja $Starost > 48.5$ dobimo tri vrstice. Vse tri imajo vrednost atributa $Primernost = 0$.

$$\text{Gini} = 1 - \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2 = 0$$

Starost ≤ 48.5	Primernost
18	0
23	1
30	0
26	1
26	1
30	1
45	0
45	1

Preostale vrstice, ki ne ustrezajo zgornjemu pogoju, zapišemo v svojo tabelo. Opazimo, da imajo 3 vrstice vrednost atributa Primernost enako 0, preostalih 5 pa ima vrednost 1.

$$\text{Gini} = 1 - \left(\frac{3}{8}\right)^2 - \left(\frac{5}{8}\right)^2 = 0.469$$

Izračunamo Gini indeks atributa Starost iz gini indeksov obeh podmnožic, s tem da upoštevamo uteži (velikost obeh množic).

$$\text{Gini}(Starost) = \frac{3}{11} \cdot 0 + \frac{8}{11} \cdot 0.469 = \underline{\underline{0.341}}$$

- Po enakem postopku izračunamo tudi vrednost Gini indeksa za atribut Izobrazba. Naključno izberemo mejo 5.

Izobrazba > 5	Primernost
6	0
6	0
9	0
6	1
8	1
9	1

$$\text{Gini} = 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0.5$$

Izobrazba ≤ 5	Primernost
2	0
4	0
4	1
2	0
2	1

$$\text{Gini} = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 0.48$$

Gini indeks za atribut Izobrazba:

$$\text{Gini}(\textit{Izobrazba}) = \frac{6}{11} \cdot 0.5 + \frac{5}{11} \cdot 0.48 = \underline{\underline{0.491}}$$

- Po enakem postopku izračunamo tudi vrednost Gini indeksa za atribut Delovna doba. Naključno izberemo mejo 6.

Delovna doba > 6	Primernost
7	1
25	1
15	0
35	0
37	0

$$\text{Gini} = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 0.48$$

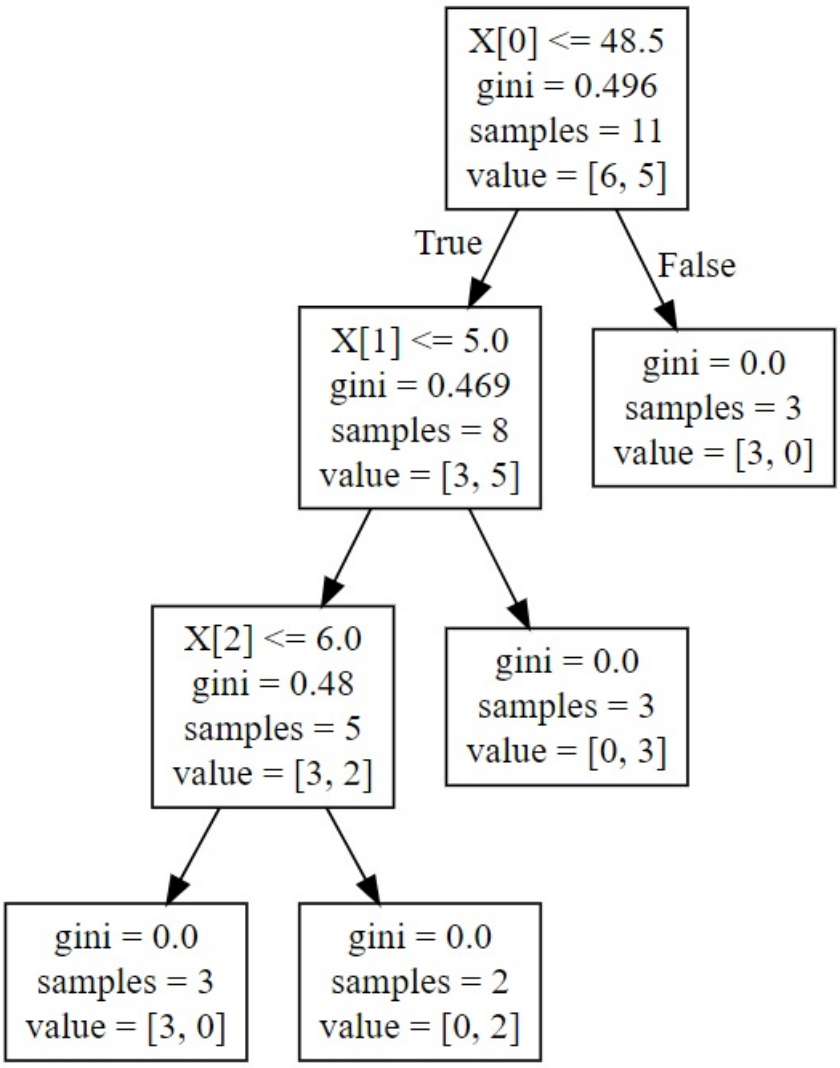
Delovna doba ≤ 6	Primernost
0	0
0	1
0	0
2	1
3	1
5	0

$$\text{Gini} = 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0.5$$

$$\text{Gini}(\textit{Delovna doba}) = \frac{5}{11} \cdot 0.48 + \frac{6}{11} \cdot 0.5 = \underline{\underline{0.491}}$$

Vidimo, da ima najmanjši Gini indeks atribut Starost, katerega postavimo na vrh odločitvenega drevesa. Gini indeks atributov Delovna doba in Izobrazba sta enaka, zato je vseeno, katerega prikažemo najprej.

Odločitveno drevo za zgornjo učno množico enajstih oseb smo zgradili in vizualizirali znotraj programa (tu so oznake $X[0]$ =Starost, $X[1]$ =Izobrazba, $X[2]$ =Delovna doba). Uporabili smo ukaz `DecisionTreeClassifier()` ter `fit()`. Za izvoz slike smo uporabili `export_graphviz()`, za napovedovanje pa `predict()`.



Slika 3.1: Odločitveno drevo za podano učno množico

Pogosta težava odločitvenih dreves je prevelika prilagoditev podatkom. Težavo se običajno rešuje z rezanjem:

- Naprej: gradnjo ustavimo preden pride do napake.
- Nazaj: najprej zgradimo drevo, nato odrežemo manj zanesljive dele.

3.2 Naivni Bayesov klasifikator

Naivni Bayesov klasifikator je klasifikacijska tehnika, zasnovana na Bayesovi formuli [13, str. 6, enačba 1.4]

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)},$$

kjer A in B predstavljata dogodka, P označuje verjetnost ter je $P(B) \neq 0$.

- $P(A|B)$ predstavlja pogojno verjetnost, da se zgodi dogodek A, pri predpostavki, da je B resničen.
- $P(B|A)$ predstavlja pogojno verjetnost, da se zgodi dogodek B, pri predpostavki, da je A resničen.
- $P(A)$ predstavlja verjetnost, da se zgodi dogodek A.
- $P(B)$ predstavlja verjetnost, da se zgodi dogodek B.

Naivni Bayesov klasifikator predvideva, da je vsak atribut *neodvisen* od preostalih atributov. Algoritem je preprosto zgraditi in je še posebej uporaben pri večjih množicah podatkov. Sodi med lene algoritme.

Pogledali si bomo uporabo Gaussovega naivnega Bayesovega klasifikatorja na učni množici iz prejšnjega podpoglavja o klasifikacijski metodi odločitvenih dreves:

Starost	Izobrazba	Delovna doba	Primernost zaposlitve delavca
18	2	0	0
23	6	0	1
30	4	0	0
26	8	2	1
26	4	7	1
30	9	3	1
45	2	5	0
45	2	25	1
52	6	15	0
55	6	35	0
60	9	37	0

Tabela 3.2: Primer učne množice za izgradnjo odločitvenega drevesa

Za vsakega od atributov smo elemente razdelili v dve množici glede na primernost zaposlitve in za posamezno množico izračunali povprečno vrednost ter varianco izbranega atributa.

Povprečna vrednost vzorca [13, str. 33] je podana z

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}.$$

Varianca vzorca [13, str. 33] je mera statistične razpršenosti spremenljivke. Podana je z

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1},$$

kjer x_i predstavlja posamezno vrednost atributa (zgornja tabela), $\overline{x}$ povprečno vrednost atributa in n velikost vzorčne množice.

Vpeljimo oznake za naše atribute: $a =$ starost, $b =$ izobrazba, $c =$ delovna doba, in $d =$ primernost za delovno mesto.

Primernost zaposlitve d	Povprečna starost $\mu_{a,d}$	Varianca starosti $\sigma^2_{a,d}$
0	43.33	262.27
1	30	76.5

Primernost zaposlitve d	Povprečna izobrazba $\mu_{b,d}$	Varianca izobrazbe $\sigma^2_{b,d}$
0	4.83	7.37
1	5.8	8.2

Primernost zaposlitve d	Povprečna delovna doba $\mu_{c,d}$	Varianca delovne dobe $\sigma^2_{c,d}$
0	15.33	286.7
1	7.4	103.3

Vzemimo za primer iskalca zaposlitve s starostjo $a = 30$ let, s stopnjo izobrazbe $b = 6$, z delovno dobo $c = 5$ let; radi bi pa iz učne množice napovedali, če je primeren za delovno mesto. V resnici nas torej zanima, katera od pogojnih verjetnosti

$$P(d{=}1|a{=}30, b{=}6, c{=}5) \quad \text{in} \quad P(d{=}0|a{=}30, b{=}6, c{=}5)$$

je večja. Po Bayesovem izreku za $x \in \{0, 1\}$ velja

$$P(d{=}x|a{=}30, b{=}6, c{=}5) = \frac{P(a{=}30, b{=}6, c{=}5|d{=}x)P(d{=}x)}{P(a{=}30, b{=}6, c{=}5)}.$$

Ker je imenovalec v obeh primerih enak, je dovolj primerjati števca. Če so a, b, c neodvisne, je števec enak

$$P(a{=}30|d{=}x)P(b{=}6|d{=}x)P(c{=}5|d{=}x)P(d{=}x).$$

V našem primeru iz tabele razberemo, da je $P(d{=}0) = \frac{6}{11}$ in $P(d{=}1) = \frac{5}{11}$. Če uporabljamo Gaussov tip [20] naivnega Bayesovega klasifikatorja, modeliramo porazdelitev vsakega atributa z normalno porazdelitvijo $N(\mu, \sigma)$ [17, poglavje 2.4, str. 3]. Tako dobimo

$$\begin{aligned}
P(a{=}30|d{=}0) &= \tfrac{1}{\sqrt{2\pi}\sigma_{a,0}} e^{-(\frac{30-\mu_{a,0}}{\sigma_{a,0}})^2/2} = 0.0176 \\
P(b{=}6|d{=}0) &= \tfrac{1}{\sqrt{2\pi}\sigma_{b,0}} e^{-(\frac{5-\mu_{b,0}}{\sigma_{b,0}})^2/2} = 0.1339 \\
P(c{=}5|d{=}0) &= \tfrac{1}{\sqrt{2\pi}\sigma_{c,0}} e^{-(\frac{5-\mu_{c,0}}{\sigma_{c,0}})^2/2} = 0.0196 \\
\text{števec} &= 0.0176 \cdot 0.1467 \cdot 0.0196 \cdot \tfrac{6}{11} = \underline{\underline{2.5 \cdot 10^{-5}}}
\end{aligned}$$

in podobno

$$P(a{=}30|d{=}1) = \frac{1}{\sqrt{2\pi}\sigma_{a,1}} e^{-(\frac{30-\mu_{a,1}}{\sigma_{a,1}})^2/2} = 0.0456$$
$$P(b{=}6|d{=}1) = \frac{1}{\sqrt{2\pi}\sigma_{b,1}} e^{-(\frac{5-\mu_{b,1}}{\sigma_{b,1}})^2/2} = 0.1389$$
$$P(c{=}5|d{=}1) = \frac{1}{\sqrt{2\pi}\sigma_{c,1}} e^{-(\frac{5-\mu_{c,1}}{\sigma_{c,1}})^2/2} = 0.0382$$
$$\text{števec} = 0.0456 \cdot 0.1339 \cdot 0.0382 \cdot \frac{5}{11} = \underline{\underline{1.09 \cdot 10^{-4}}}.$$

Ker je števec večji v primeru, ko je delavec primeren, napovemo, da bi bil novo vnešeni delavec s podanimi podatki primeren za omenjeno delovno mesto.

Izračune verjetnosti z uporabo Bayesove formule smo izvedli znotraj programa. Uporabili smo ukaz `GaussianNB()` ter `fit()`. Za napovedovanje smo uporabili `predict()`.

3.3 Klasifikacija z metodo najbližjega soseda (k-NN algoritem)

K-NN algoritem [15] je neparametrična metoda za klasifikacijo in regresijo. Sodi med lene algoritme. Rezultat je sestavljen iz k-najbližjih učnih primerov. Pri klasifikaciji je rezultat algoritma pripadnost razredu, medtem ko je pri regresiji dejanska vrednost.

K-NN algoritem [1] deluje po naslednjem vrstnem redu:

1. Vsak element učne množice predstavimo kot točko $q \in \mathbb{R}^n$. Dobimo novo poizvedbo (deloiskalec $p \in \mathbb{R}^n$).

2. Vnaprej izberemo k. Ta številka bo predstavljala število najbližjih sosedov točke p, s pomočjo katerih bomo določili odvisni atribut za p.

3. Po formuli Evklidske razdalje [1, str. 100, formula 3.2]

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + ... + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

izračunamo razdaljo med atributi nove poizvedbe p in atributi vseh primerov q v učni množici.

4. Iz učne množice izberemo k najbližjih sosedov točke p.

5. Preštejemo kolikokrat se posamezna vrednost odvisne spremenljivke pojavi pri izbranih sosedih. Največkrat pojavljena vrednost predstavlja dejansko napovedano vrednost odvisne spremenljivke novega vzorca p.

Da si bomo delovanje algoritma lahko pogledali v praksi [14], bomo sedaj izvedli k-NN algoritem na zgoraj definirani učni množici. Izberimo $k = 3$. Želeli bi izvedeti, če je deloiskalec v starosti 38 let, s 6. stopnjo izobrazbe in z 8 let delovne dobe primeren za delovno mesto znotraj podjetja, katerega pretekle prijave na delovne mesto so prikazane v spodnji tabeli.

	Starost	Izobrazba	Delovna doba	Primernost	Razdalja do $(38, 6, 8)$	vsebovan v k-NN?
1	18	2	0	0	21.91	NE
2	23	6	0	1	17	NE
3	30	4	0	0	(11.49)	DA
4	26	8	2	1	13.56	NE
5	26	4	7	1	12.21	NE
6	30	9	3	1	(9.9)	DA
7	45	2	5	0	(8.6)	DA
8	45	2	25	1	18.81	NE
9	52	6	15	0	15.65	NE
10	55	6	35	0	31.9	NE
11	60	9	37	0	36.5	NE

Tabela 3.3: Učna množica s pripadajočimi razdaljami

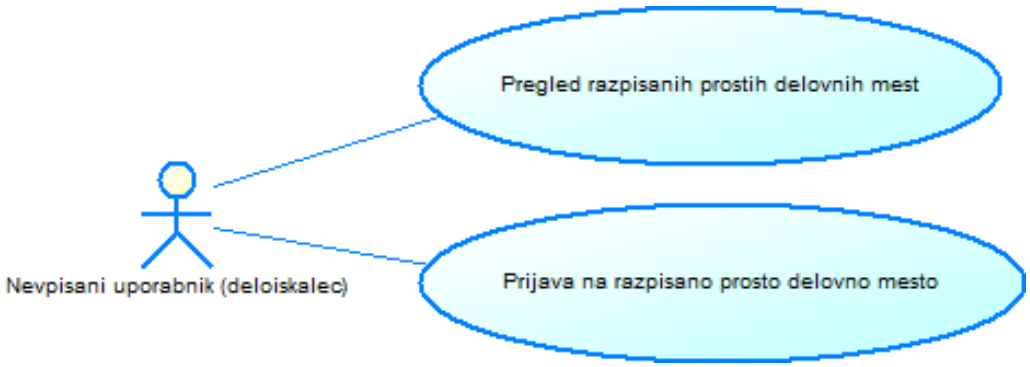
Trije najbližji sosedi se nahajajo v vrsticah 3, 6, 7 (obkrožene številke). Opazimo, da imata dva izmed sosedov Primernost$= 0$, eden pa Primernost$= 1$. Ker je v bližini našega deloiskalca $(38, 6, 8)$ več neprimernih kot primernih

sosedov, napovemo, da je tudi on neprimeren za oglaševano delovno mesto: $\text{Primernost}(38, 6, 8) = 0$.

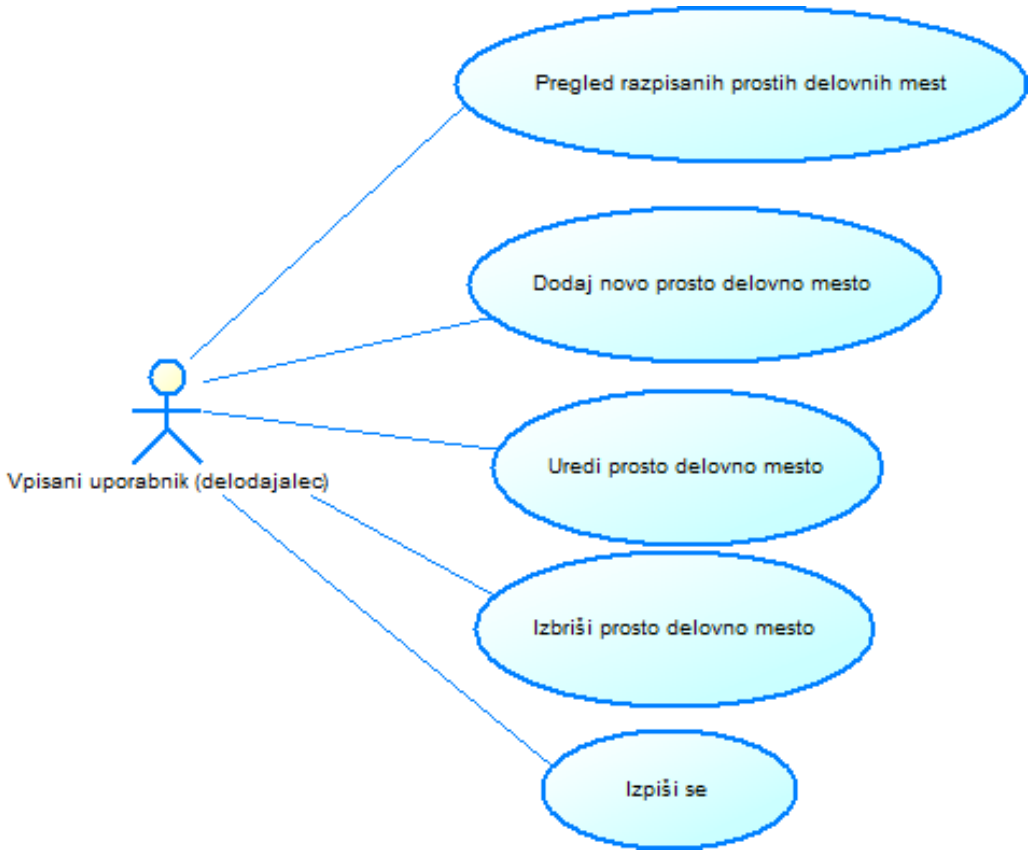
Poglavje 4

Implementacija aplikacije

Za lažje razumevanje delovanja aplikacije smo najprej zgradili diagrama primerov uporabe za prijavljenega ter neprijavljenega uporabnika.



Slika 4.1: Diagram primerov uporabe nevpisanega uporabnika



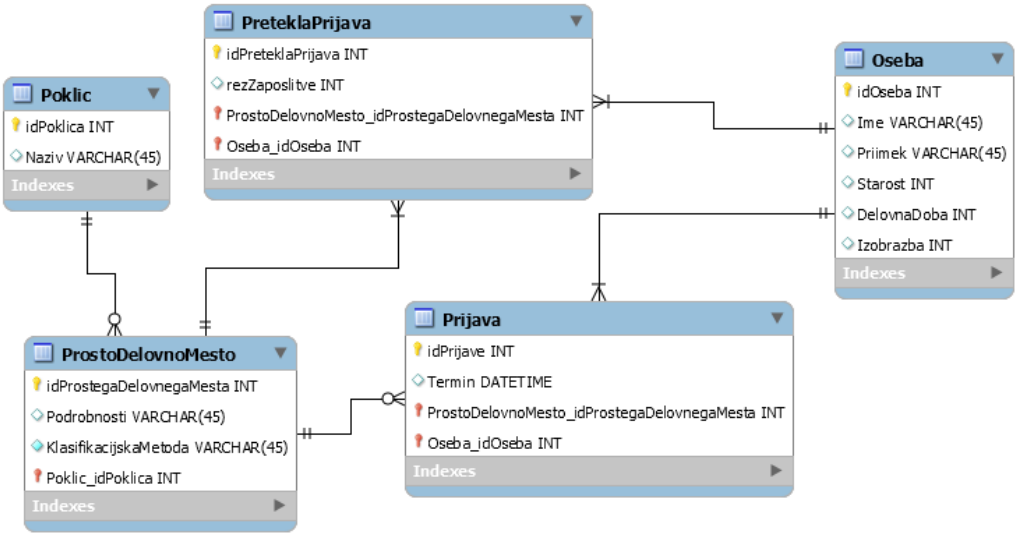
Slika 4.2: Diagram primerov uporabe vpisanega uporabnika

Pri razvoju aplikacije smo najprej ustvarili nov prazen Djangov spletni projekt znotraj Visual studio razvojnega okolja. Sledilo je načrtovanje izgradnje EER modela. ER model je natančna logična predstavitev podatkovnega modela, medtem ko je EER visokonivojski podatkovni model, ki vključuje razširitve prvotnega ER modela. Med temi razširitvami so podrazredi, nadrazredi, specializacija in generalizacija [10].

Sprva smo razmislili, kaj naj bi hranili v podatkovni bazi, kakšnega tipa naj bi bili posamezni podatki, kako bi poimenovali posamezne tabele ter kakšne naj bi bile relacije med njimi.

Glede na predhodno zastavljene cilje smo ugotovili, da bomo potrebovali tabele namenjene:

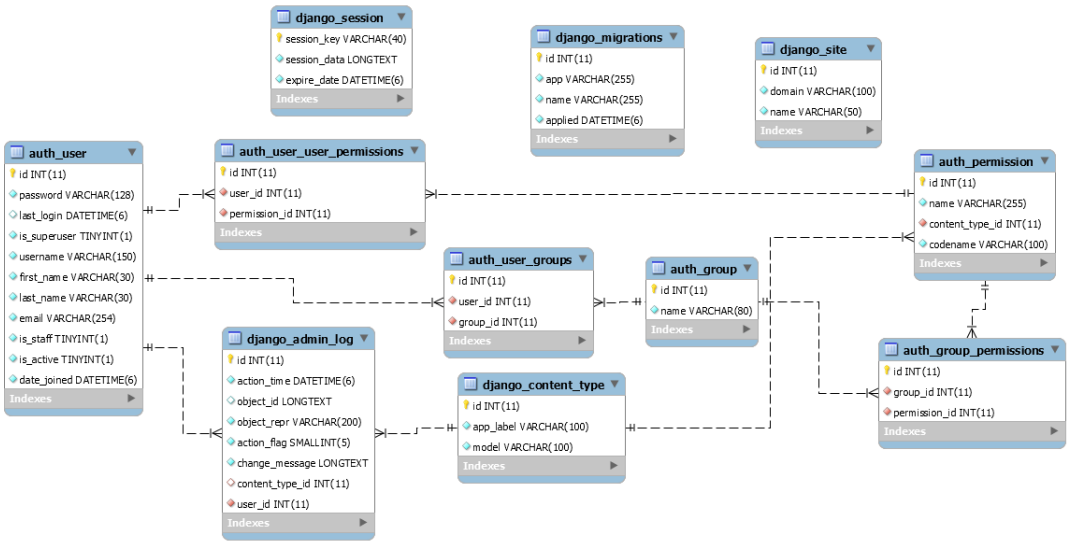
- hranjenju razpisanega prostega delovnega mesta,
- hranjenju posameznih možnih poklicev,
- hranjenju preteklih prijav na posamezno delovno mesto,
- hranjenju trenutnih prijav na razpisano prosto delovno mesto,
- hranjenju oseb, prijavljenih na razpisana prosta delovna mesto ali na pretekle prijave.



Slika 4.3: EER diagram za hranjenje zgoraj navedenih elementov

Sledilo je ustvarjanje nove podatkovne baze z imenom `db`. Ker ima Django že vgrajen sistem za avtentikacijo, je po dokončani vzpostavitvi le ta že vsebovala potrebne tabele. S pomočjo vnaprejšnjega inženiringa za EER diagram iz slike 4.3, smo v bazo `db` dodali še manjkajoče tabele.

S pomočjo vzvratnega inženiringa pa smo dobili še EER diagram tabel, namenjenih avtentikaciji.



Slika 4.4: EER diagram za potrebe avtentikacije

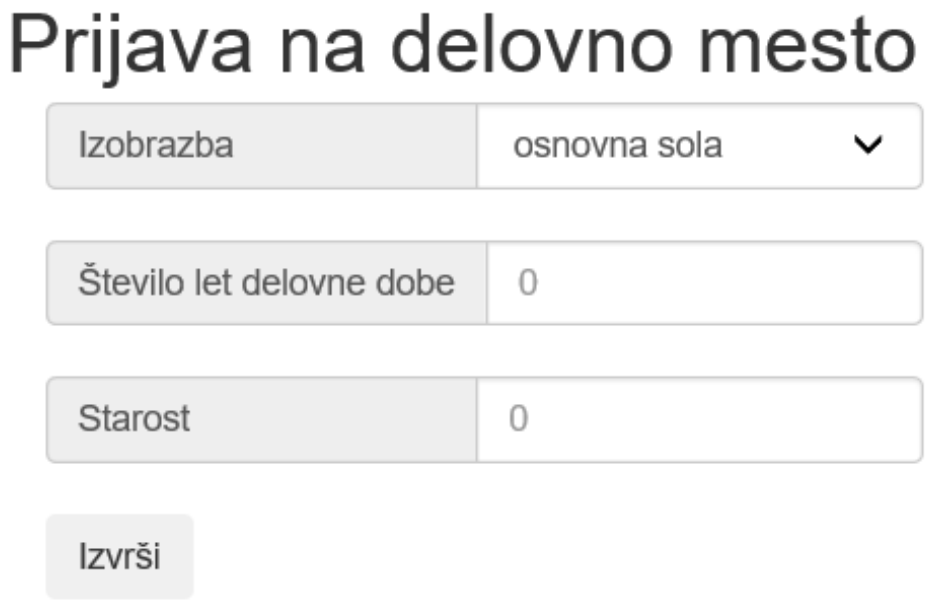
Djangov vgrajeni sistem skrbi tako za avtentikacijo kot tudi avtorizacijo. Prvi je zadolžen za prepoznavanje uporabnika, medtem ko je drugi zadolžen za prepoznavanje pravic posameznega uporabnika.

Djangov avtentikacijski sistem je sestavljen iz naslednjih elementov:

- uporabniki;
- dovoljenja: binarne zastavice, ki določajo ali lahko uporabnik izvrši določeno nalogo;
- skupine: omogoča kombiniranje posameznih uporabnikov z njihovimi pravicami in oznakami;

- nastavljiv sistem šifriranja gesel;
- orodja za vpis in omejevanje vsebine;
- vgrajen zaledni sistem.

Po uspešni vzpostavitvi baze smo se lotili ustvarjanja enostavnega pogleda (angl. view), namenjenega vnosu potrebnih atributov za izvršitev klasifikacijske metode. Ustvarili smo datoteko prviPogled.html in znotraj nje definirali vnosna polja za starost, izobrazbo in delovno dobo ter pripadajoče oznake (angl. label) za lažje razumevanje. Dodali smo tudi gumb `Izvrši`, ki ob pritisku izvede metodo `drugiPogled`.



Slika 4.5: Pogled, namenjen vnosu podatkov za izvršitev klasifikacijske metode

Znotraj metode `drugiPogled` se izvede ena izmed klasifikacijskih metod. Implementirali smo vse tri metode. Imeli smo težavo, kako ustvariti učno množico, ki bi ustrezala posameznemu delovnemu mestu. Za potrebe testiranja smo sprva ustvarili neko izmišljeno učno množico in z njo testirali delovanje posameznih metod ter prikaz strani.

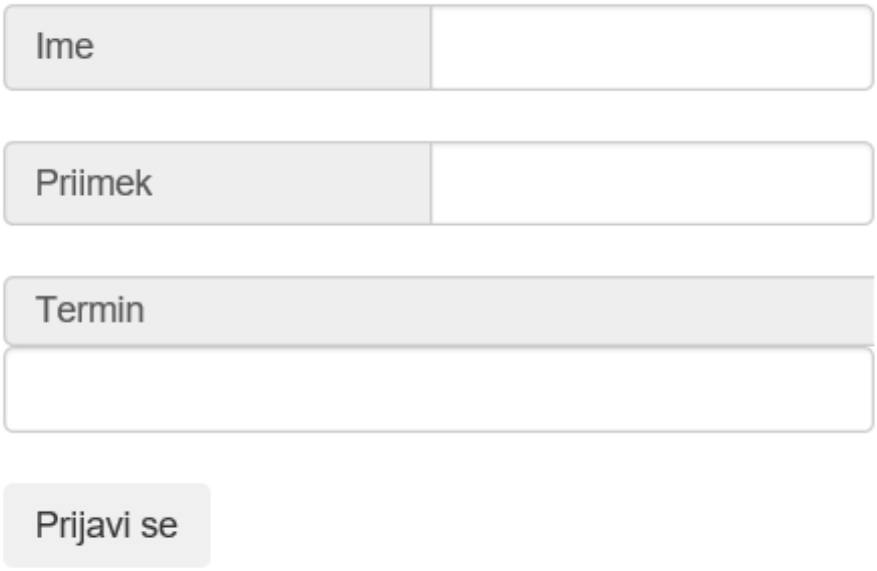
```
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier

#Preverimo za katero klasifikacijsko metodo gre
if klMetoda=="odlocitveno drevo":
   clf = tree.DecisionTreeClassifier()
   #1-neprimeren, 0-primeren
   clf = clf.fit(ustrMn, rez)
   predict=clf.predict([[starost, izobrazba, delDoba]])
elif klMetoda=="naivni Bayes":
    clf = GaussianNB()
    x=np.array(ustrMn).astype(np.float)
    y=np.array(rez).astype(np.float)
    clf.fit(x, y)
   predict=clf.predict([[starost, izobrazba, delDoba]])
elif klMetoda=="knn":
    neigh = KNeighborsClassifier(n_neighbors=3)
    neigh.fit(ustrMn, rez)
    predict=neigh.predict([[starost, izobrazba, delDoba]])
```

Na podlagi izvršene izbrane klasifikacijske metode in njenega rezultata se na drugi strani prikaže eden izmed zapisov:

- "Glede na vnešene podatke, žal, niste primerni za to delovno mesto."
- "Čestitamo! Primerni ste za omenjeno delovno mesto. Prosimo vnesite spodnje podatke."

V primeru, ko je bilo ugotovljeno, da je uporabnik primeren za določeno delovno mesto, se mu odpre tudi obrazec za vnos podatkov (ime, priimek in termin razgovora, ki bi mu ustrezal).



Slika 4.6: Forma, namenjena vnosu podatkov na razpisano prosto delovno mesto

Za potrebe vnosa termina smo uporabili razširjeno različico Bootstrapovega Datepicker-ja. DateTimePicker nam omogoča izbiro razpoložljivega datuma in ure. Po pritisku gumba `Prijavi se`, se s pomočjo Ajax-a izvede klic metode vnos.

```
<script type="text/javascript">
   $(document).on('submit', '#klic', function(e) {
      e.preventDefault();
        $.ajax({
            url: '/vnos/',
            type: 'GET',
            async: 'true',
            data: {
               ime: $("#im").val(),
                priimek: $("#priim").val(),
                dat: $('#dt').val(),
```

```
                starost: """+str(starost)+""",
                delDoba: """+str(delDoba)+""",
                izobrazba: """+str(izobrazba)+""",
                csrfmiddlewaretoken:
                $('input[name=csrfmiddlewaretoken]').val()
            },
            success: function(request) {
               alert("Zahvaljujemo se vam za prijavo. Se vidimo.");
                window.location = "/";
            },
            error: function(obj, status, err) {
               alert(err)
            }
        });
    });
</script>
```

Ajax je tehnologija na odjemalski strani, ki se uporablja za izdelavo asinhronih zahtev strežniški strani, kjer odzivi ne povzročijo osvežitve strani [2].

Slika 4.7: Delovanje tehnologije Ajax [2]

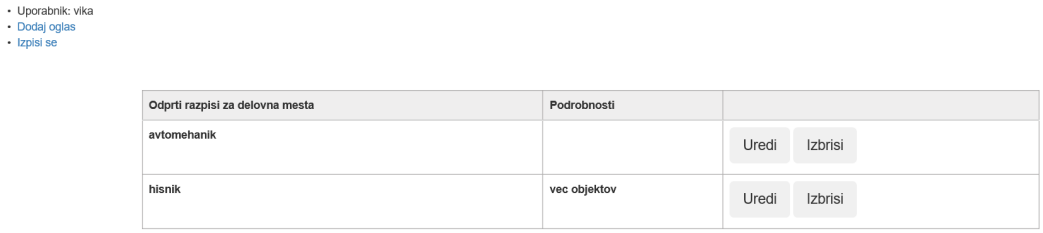
Ob pritisku gumba se v tabelah `Prijava` in `Oseba` shranijo podatki. Ker ima tabela `Prijava` tuj ključ, vezan na tabelo `ProstoDelovnoMesto`, smo za potrebe testiranja dodali neko izmišljeno prosto delovno mesto in vse nove prijave vezali nanj.

Po uspešni implementaciji prijave na delovno mesto, smo se odločili ustvariti vstopno stran, kjer se bo nahajala tabela vseh razpisanih prostih delovnih mest. Ustvarili smo datoteko odlocitev.html in jo določili za vstopno stran. Znotraj omenjene datoteke, smo ustvarili tabelo in jo napolnili s podatki, poslanimi iz Python kode preko ukaza `Template.render(context)`. Spremenljivka `context` je v našem primeru slovar, v katerega smo dodali s ključem `data` množico vseh prostih delovnih mest. Prikaz tabele smo izvedli s

kodo, ki se sprehodi čez množico prostih delovnih mest. V tabelo smo dodali tudi formo z gumbom za preusmeritev na stran, ki vsebuje vnosna polja za starost, izobrazbo in delovno dobo. V formo smo dodali tudi skrito vnosno polje, v katero smo shranili `id` trenutno izbranega prostega delovnega mesta.

```
{% for item in data %}
   <tr>
       <th><item>{{ item.poklic_idpoklica.naziv }}</item></th>
       <th><item>{{ item.podrobnosti }}</item></th>
       <th>
            <form id="preusmeritev" action="/prva/" method="GET">
               <input type="hidden" name="tag" value="{{
                   item.idprostegadelovnegamesta }}" />
               <input class="btn btn-secondary btn-lg btn-block"
                   type="submit" value="Prijavi se" />
            </form>
       </th>
    </tr>
{% endfor %}
```

Po uspešnem prikazu prostih delovni mest iz tabele smo začeli z implementacijo avtentikacije uporabnika in možnosti oddaje novega oglasa. Uporabili smo že vgrajen Djangov avtentikacijski sistem, ki ima tudi implementirani metodi `login` in `logout`. Ustvarili smo nov pogled (angl. view) za vnos uporabniškega imena in gesla. V primeru, da uporabniško ime in geslo nista pravilna, se izpiše napaka, sicer pa se preusmeri na prvotno stran, kjer se nahaja tabela razpisanih prostih delovnih mest. Na omenjeni strani smo vodili sejo ali je uporabnik vpisan. Vpisan uporabnik ima možnosti oddaje novega oglasa, urejanja in brisanja že obstoječih oglasov. Nevpisani uporabnik pa ima le možnost prijave na razpisano prosto delovno mesto.



Slika 4.8: Zaslonska maska vpisanega uporabnika

Slika 4.9: Zaslonska maska nevpisanega uporabnika

Urediti je bilo potrebno tudi vnos novega prostega delovnega mesta. Kot je razvidno iz zaslonskih mask iz Slik 4.8 in 4.9 ima to možnost le vpisan uporabnik. Ker smo imeli pri implementaciji klasifikacijskih metod težavo z ustvarjanjem učne množice smo razmišljali, da bi se za vsako novo delovno mesto posebej vneslo podatke o preteklih prijavah in njihovih rezultatih zaposlitve. To bi nato pri posamezni izvršitvi klasifikacijske metode uporabili kot učno množico. Zdelo se nam je najbolj smiselno, da bi uporabnik aplikacije vodil evidenco prijav in rezultatov znotraj Excel datoteke, ki bi jo program ob oddaji oglasa prebral ter s podatki napolnil tabeli `PreteklaPrijava` in `Oseba`.

```
mn=request.GET.get('file')
wb = open_workbook(mn)
for sheet in wb.sheets():
   stVrstic = sheet.nrows
    stStolpcev = sheet.ncols
    elementi = []
    stolpci = []
    for vrstica in range(1, stVrstic):
```

```
vrednosti = []
  for stolpec in range(stStolpcev):
   vrednost = (sheet.cell(vrstica,stolpec).value)
      try:
         vrednost = str(int(vrednost))
      except ValueError:
         pass
      finally:
          vrednosti.append(vrednost)
  oseba=Oseba(ime=vrednosti[0], priimek=vrednosti[1],
      starost=int(vrednosti[2]),
      delovnadoba=int(vrednosti[3]),
      izobrazba=int(vrednosti[4]))
  oseba.save()
  pretPrijave=Preteklaprijava(rezzaposlitve=int(vrednosti[5]),
      oseba_idoseba=oseba,
      prostodelovnomesto_idprostegadelovnegamesta=maxId)
  pretPrijave.save()
```

Prav tako smo znotraj oddaje novega oglasa omogočili izbiro klasifikacijske metode, ki naj se izvrši pri prijavi. Po kliku na gumb so se podatki iz Excel datoteke prenesli v tabeli `PreteklaPrijava` in `Oseba`, podatki iz vnosnih polj pa so se shranili v tabeli `ProstoDelovnoMesto`.

Poglavje 5

Zaključek

Pri razvoju aplikacije smo veliko časa porabili za spoznavanje novih tehnologij. Med študijem namreč nisem prišla v stik z uporabljenimi orodji, zato je bil to zame precejšen izziv. Želeli smo preizkusiti orodja, ki se dandanes uporabljajo v praksi, in z njimi ustvariti novo aplikacijo. Želela sem tudi dobiti izkušnje, ki bi bile lahko uporabne pri bodoči zaposlitvi. Sprva smo se lotili pregleda vseh sodobnih tehnologij. Ker sem se v okviru predmeta Podatkovno rudarjenje že seznanila s klasifikacijskimi metodami, implementiranimi v knjižnici Sci-kit, smo se odločili za programski jezik Python. Nadaljne izbire so bile naključne.

Izdelana aplikacija je uporabna v praksi znotraj podjetij, ki iščejo novega delavca iz prevelike množice kandidatov. Pri uporabi moramo paziti le na primerno izbiro klasifikacijske metode. Tekom razvoja smo namreč implementirali tri metode in izbiro le te prepustili posameznemu delodajalcu ob oddaji novega oglasa za prosto delovno mesto. Namreč ne zdi se nam primerno prepustiti izbiro podjetjem, saj le te mnogokrat nimajo dovolj znanja o klasifikacijskih metodah. Iz omenjenega razloga bi pri nadaljnem razvoju dodali še preverjanje točnosti napovedovanja posameznih metod in se na podlagi tega odločili za eno izmed njih.

Vsesplošno pa se nam zdi aplikacija zelo uporabna in bi podjetjem olajšala izbiro primernih kandidatov za razpisano delovno mesto. Vzorci novih zapo-

slitev se mnogokrat ponavljajo. Tudi delodajalci sledijo vzorcem, kaj iščejo pri bodočem delavcu. Zato se nam zdi primerno, da bi se te vzorce upoštevalo še pred dejanskimi razgovori. Veliko preveč časa sicer zaposleni porabijo za prebiranje življenjepisov in izpraševanje kandidatov.

Literatura

[1] D.A. Adeniyi, Z. Wei, and Y. Yongquan. Automated web usage data mining and recommendation system using k-nearest neighbor (knn) classification method. *Applied Computing and Informatics*, 12(1):90 – 108, 2016.

[2] Django and AJAX Form Submissions – Say 'Goodbye' to the Page Refresh. Dosegljivo: `https://realpython.com/blog/python/django-and-ajax-form-submissions/`, 2014. [Dostopano 9. 2. 2018].

[3] What is Bootstrap? Dosegljivo: `https://stackoverflow.com/questions/14546709/what-is-bootstrap`, 2017. [Dostopano 21. 12. 2017].

[4] About twitter-bootstrap. Dosegljivo: `https://stackoverflow.com/tags/twitter-bootstrap/info`, 2017. [Dostopano 21. 12. 2017].

[5] Statistical classification. Dosegljivo: `https://en.wikipedia.org/wiki/Statistical_classification`, 2017. [Dostopano 24. 11. 2017].

[6] Meet Django. Dosegljivo: `https://www.djangoproject.com/`, 2017. [Dostopano 18. 12. 2017].

[7] What is Django? Dosegljivo: `https://tutorial.djangogirls.org/en/django/`, 2017. [Dostopano 18. 12. 2017].

[8] CLASSIFICATION METHODS. Dosegljivo: `http://www.d.umn.edu/~padhy005/Chapter5.html`, 2017. [Dostopano 29. 11. 2017].

[9] HOW DECISION TREE ALGORITHM WORKS. Dosegljivo: https://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/, 2017. [Dostopano 4. 12. 2017].

[10] Entitetni-relacijski model. Dosegljivo: http://www.tutorialride.com/dbms/enhanced-entity-relationship-model-eer-model.htm, 2017. [Dostopano 8. 2. 2018].

[11] HTML. Dosegljivo: https://en.wikipedia.org/wiki/HTML, 2017. [Dostopano 21. 12. 2017].

[12] What is HTML? Dosegljivo: http://www.yourhtmlsource.com/starthere/whatishtml.html, 2017. [Dostopano 21. 12. 2017].

[13] Hans-Michael Kaltenbach. *A Concise Guide to Statistics*. SpringerBriefs in Statistics. Springer, Berlin, 2012.

[14] Numerical example of K Nearest Neighbor Algorithm. Dosegljivo: http://people.revoledu.com/kardi/tutorial/KNN/KNN_Numerical-example.html, 2017. [Dostopano 23. 11. 2017].

[15] k-nearest neighbors algorithm. Dosegljivo: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm, 2017. [Dostopano 23. 11. 2017].

[16] Oded Maimon Lior Rokach. *Data Mining With Decision Trees: Theory and Applications (2nd Edition)*. Series in Machine Perception and Artifical Intelligence. World Scientific Publishing Company, 2 edition, 2015.

[17] Vangelis Metsis and et al. Spam filtering with naive bayes – which naive bayes? In *THIRD CONFERENCE ON EMAIL AND ANTI-SPAM (CEAS)*, 2006.

[18] MySQL. Dosegljivo: https://sl.wikipedia.org/wiki/MySQL, 2017. [Dostopano 11. 2. 2018].

[19] MySQL. Dosegljivo: https://en.wikipedia.org/wiki/MySQL, 2018. [Dostopano 11. 2. 2018].

[20] Naive Bayes classifier. Dosegljivo: https://en.wikipedia.org/wiki/Naive_Bayes_classifier#Gaussian_naive_Bayes, 2017. [Dostopano 8. 2. 2018].

[21] What is Python? Dosegljivo: https://www.python.org/doc/essays/blurb/, 2017. [Dostopano 23. 11. 2017].

[22] Rajeev Rastogi and Kyuseok Shim. Public: A decision tree classifier that integrates building and pruning. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 404–415, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[23] scikit-learn. Dosegljivo: https://en.wikipedia.org/wiki/Scikit-learn, 2017. [Dostopano 18. 12. 2017].

[24] SQL. Dosegljivo: https://sl.wikipedia.org/wiki/SQL, 2017. [Dostopano 15. 2. 2018].

[25] MySQL/Language/Definitions: what are DDL, DML and DQL? Dosegljivo: https://en.wikibooks.org/wiki/MySQL/Language/Definitions:_what_are_DDL,_DML_and_DQL%3F, 2017. [Dostopano 15. 2. 2018].

[26] Visual Studio. Dosegljivo: https://sl.wikipedia.org/wiki/Visual_Studio, 2017. [Dostopano 18. 12. 2017].

[27] Python (programming language). Dosegljivo: https://en.wikipedia.org/wiki/Python_(programming_language), 2017. [Dostopano 23. 11. 2017].