UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET

Miljan Denić

Algoritmi stabla odlučivanja i primeri implementacije
Inteligentni sistemi
Studijski program: Računarstvo i informatika
Modul: Softversko inženjerstvo

Kandidat:

Miljan Denić, br. ind. 1517

Mentor:

prof. dr. Leonid Stoimenov

Niš, januar 2024. godina

UVOD

U današnjem svetu oblast mašinskog učenja je jedna od najbitnijih oblasti za razvoj i unapređenje autonomnih sistema koji mogu samostalno donositi odluke. Bitan pravac u okviru oblasti mašinskog učenja je upotreba algoritma stabla odlučivanja koji omogućavaju efikasnu implementaciju rešenja za donošenje odluka na osnovu dostupnih podataka.

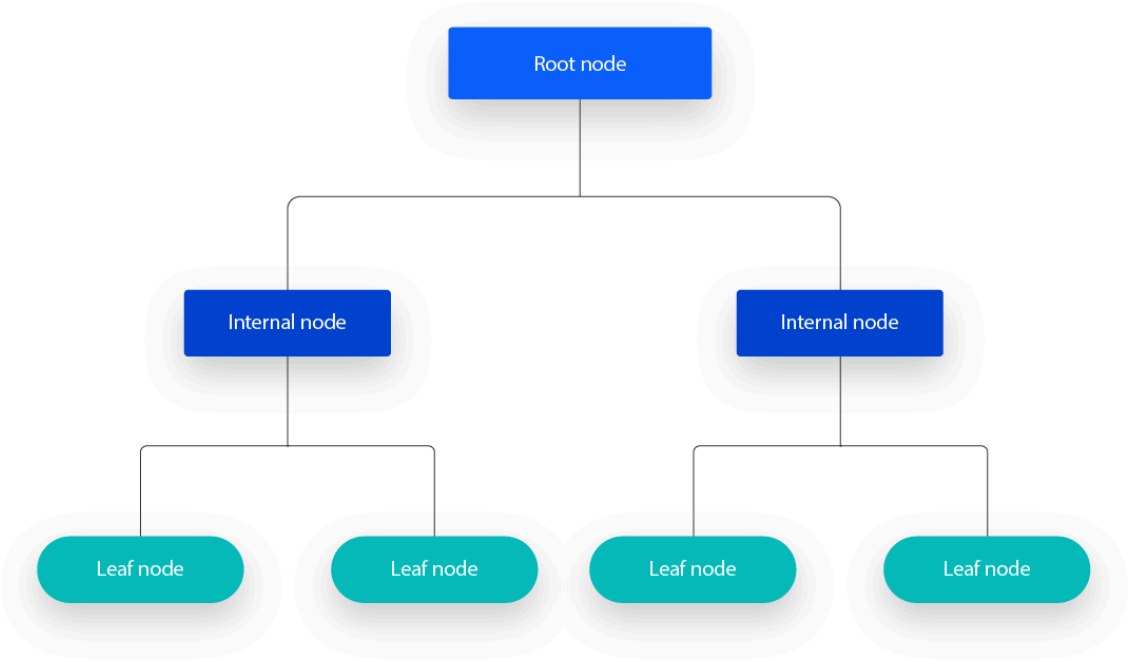
Stabla odlučivanja predstavljaju bitan koncept u mašinskom učenju, a u ovom radu će se posebno obraditi implementacija ID3 algoritma, jednog od prvih algoritma u oblasti kreiranja stabala odlučivanja. ID3 algoritam je zastupljen u klasifikacionim zadacima, gde se oslanja na iterativni proces razdvajanja podataka kako bi konstruisao stablo odluke.

Teorijski deo rada će pružiti detaljan uvid u osnovne principe rada stabala odlučivanja, sa fokusom na klasifikaciona i regresiona stabla. Dalje, rad će detaljnije istražiti ID3 algoritam, analizirajući teorijske elemente kao što su entropija i informaciona dobit, dok će se praktični deo fokusirati na konkretnu implementaciju ID3 algoritma.

Osim ID3 algoritma, kroz rad će takođe biti obrađeni i drugi algoritmi stabla odlučivanja, kao što su C4.5, CART i CHAID. Pored toga, biće analizirane karakteristike i primene svakog od ovih algoritama.

STABLA ODLUČIVANJA

Stablo odlučivanja je supervizovani algoritam bez parametara u mašinskom učenju koji se koristi za klasifikaciju i za regresione zadatke. Ovaj algoritam ima hijerarhijsku strukturu stabla koja se sastoji od korena, grana, unutrašnjih čvorova i listova.



Slika 1. Struktura stabla odlučivanja

Kao što se vidi na dijagramu iznad, stablo odlučivanja počinje od korena koji nema ulaznih grana. Izlazne grane iz korena zatim ulaze u unutrašnje čvorove, koji se još zovu i čvorovi odluke. Na osnovu dostupnih karakteristika, oba tipa čvorova vrše evaluacije kako bi formirali homogene podskupove, koji su označeni kao listovi ili terminalni čvorovi. Listovi predstavljaju sve moguće ishode unutar skupa podataka.

Ova vrsta strukture dijagrama olakšava razumevanje procesa donošenja odluka, omogućavajući različitim grupama unutar organizacije da bolje razumeju zašto je određena odluka doneta.

Tipovi stabla odlučivanja

Stabla odlučivanja mogu se kategorisati u dve glavne vrste na osnovu vrste rezulatata koje mogu da predvide: klasifikaciona stabla i regresiona stabla.

Klasifikaciona stabla

Ova vrsta stabala odlučivanja koristi se kada je cilj predviđanje kategoričke ili diskretne promenljive, odnosno pripadanje određenoj klasi ili kategoriji. Primena

klasifikacionih stabala može biti u različitim domenima, kao što su medicinska dijagnostika (npr. dijagnoza bolesti na osnovu simptoma), analiza potrošača (npr. predviđanje kupovnih navika), ili prepoznavanje oblika (npr. klasifikacija slika po kategorijama).

Ključne karakteristike klasifikacionih stabla su:

- Kategorički ishod: Ciljna promenljiva je kategorička (npr. da, ne ili spam, nije spam)
- Listovi: Terminalni čvorovi (listovi) stabla odgovaraju klasama. Svaki list predstavlja određenu klasu, pri čemu većinska klasa u listu predstavlja predviđenu klasu za nove podatke koji se nalaze u toj regiji.
- Kriterijum deljenja: Kriterijumi deljenja korišćeni za klasifikaciona stabla teže minimizaciji nečistoće ili povećanju homogenosti unutar svake regije. Česti kriterijumi nečistoće uključuju Gini nečistoću i entropiju.
- Primer upotrebe: Predviđanje da li je e-pošta spam ili nije na osnovu karakteristika kao što su naslov, pošiljalac i ključne reči.

Regresiona stabla

Ove vrste stabala odlučivanja koriste se kada je cilj predviđanje kontinualne vrednosti ili numeričke promenljive. Ovi modeli su korisni za predviđanje neprekidnih vrednosti, kao što su cena nekretnina na osnovu karakteristika, prognoza temperature na osnovu meteoroloških podataka, ili procena vremena potrebnog za završetak određenog zadatka.

Ključne karakteristike klasifikacionih stabla su:

- Kontinualni rezultat: Ciljna promenljiva je kontinualna numerička vrednost, kao što su temperatura, cena ili godine.
- List čvorovi: Terminalni čvorovi (listovi) stabla sadrže predviđene numeričke vrednosti. Predviđena vrednost za novi podatak koji se nalazi u određenom listu je prosečna (ili druga mera) vrednost ciljne promenljive u tom listu.
- Kriterijum deljenja: Kriterijumi deljenja u regresionim stablima teže minimizaciji varijanse ciljne promenljive unutar svake regije.
- Primer upotrebe: Predviđanje cene kuće na osnovu karakteristika kao što su površina, broj spavaćih soba i lokacija.

ID3 algoritam

ID3, skraćeno od Iterativni Dihotomizer 3, dobio je naziv zbog toga što algoritam iterativno deli karakteristike u dve ili više grupa u svakom koraku. ID3 je algoritam koji je osmislio Ross Quinlan kako bi generisao stablo odlučivanja iz skupa podataka i predstavlja jedan od najpopularnijih algoritama za konstrukciju stabala.

ID3 je osnovni algoritam za izgradnju stabla odlučivanja. Koristi top-down pretragu kroz sve moguće grane bez povratka. Ovaj algoritam koristi informacionu dobit i entropiju kako bi konstruisao stablo odlučivanja za klasifikaciju.

Informaciona dobit meri efikasnost određene osobine u klasifikaciji podataka, dok entropija meri nečistoću ili nesigurnost unutar skupa primera. ID3 traži osobine koje maksimizuju informacioni dobit ili minimizuju entropiju, omogućavajući donošenje odluka o najboljoj osobini za deljenje podataka na svakom čvoru.

ID3 je postao izuzetno popularan zbog svoje efikasnosti u konstrukciji stabala odlučivanja, posebno u situacijama gde je interpretacija i razumevanje osnovnog procesa donošenja odluka bitna.

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

Slika 2. Formula za računanje entropije

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

Slika 3. Formula za računanje informacione dobiti

Jedna od ključnih karakteristika ID3 algoritma je sklonost ka pretreniranju trening podacima. Radi sprečavanja pretreniranosti, preporučuje se korišćenje manjih stabala koja su manje sklona ovoj pojavi.

Takođe, ID3 algoritam obično generiše manja stabla, međutim, ne uvek najmanja moguća. Ovo može doprineti boljoj interpretaciji stabla ali takva stabla nisu uvek najoptimalnija u smislu minimalne strukture stabla.

Jedan od izazova prilikom korišćenja ID3 algoritma jeste rad sa kontinualnim podacima. Ako su vrednosti atributa kontinualne prirode, postoji mnogo mogućih mesta za deljenje podataka na osnovu tih atributa. Pronalaženje optimalne vrednosti za deljenje može biti vremenski zahtevno, što predstavlja jednu od poteškoća kod korišćenja algoritma u takvim slučajevima.

Kada je reč o prednostima, ID3 algoritam je ekonomičan u konstrukciji i brz u klasifikaciji nepoznatih podataka. Posebno se ističe u interpretaciji manjih stabala odlučivanja. Takođe, pokazuje otpornost na šum, naročito kada se primene metode koje sprečavaju pretreniranost. Osim toga, algoritam se može relativno dobro nositi sa redundantnim ili irelevantnim atributima, osim ako ti atributi nisu međusobno povezani.

Sa druge strane, ID3 algoritam ima i svoje mane. Veličina mogućih stabala odlučivanja je ogromana, pa top-down pristup često ne pronađe najbolje stablo. Takođe, algoritam ne uzima u obzir interakcije između atributa prilikom formiranja granica odluke, a svaka granica uključuje samo jedan atribut.

C4.5

C4.5 predstavlja proširenje ID3 algoritma koje je predstavio Ross Quinlan. Ovaj algoritam rešava određena ograničenja ID3 algoritma i radi kako za kategoričke, tako i za numeričke promenljive. C4.5 podržava koncepte poput odnosa dobitka (gain ration) i upravljanje nedostajućim vrednostima.

Ključne karakteristike C4.5:

- Odabir atributa: C4.5 koristi odnos dobiti, koji prilagođava informacionu dobit prema informacijama iz svakog atributa. Ovo rešava pristrasnost ka atributima sa mnogo vrednosti.

- Rukovanje numeričkim podacima: C4.5 može rukovati numeričkim karakteristikama tako što ih konvertuje u diskretne intervale.

- Orezivanje: C4.5 uključuje korak orezivanja kako bi smanjio pretreniranost uklanjanjem nepotrebnih grana.

- Složena stabla: C4.5 može kreirati složenija stabla u poređenju sa ID3 algoritmom, što dovodi do poboljšane tačnosti.

C4.5 algoritam rešava određena ograničenja ID3 algoritma uvodeći bolje metode odabira atributa, rukovanja numeričkim podacima i postupaka orezivanja radi smanjenja pretreniranosti. Ova unapređenja su omogućila širu primenu C4.5 algoritma u različitim oblastima, od analize podataka u poslovanju do medicinskih dijagnoza, pružajući preciznije modele odlučivanja i predviđanja.

CART

CART (Classification and Regression Trees) je algoritam stabala odlučivanja koji su predstavili Breiman i saradnici. Koristi se za klasifikaciju i za regresiju i generiše binarna stabla. Osnovni cilj CART algoritma je minimizacija nečistoće ili varijacija u svakom čvoru.

Ključne karakteristike CART algoritma:

- Odabir atributa: CART bira atribute na osnovu Gini nečistoće (za klasifikaciju) ili smanjenja varijanse (za regresiju).

- Binarno razdvajanje: CART generiše binarne podele, što dovodi do uravnoteženih stabala.

- Regresiona stabla: CART se široko koristi za regresione zadatke, gde predviđa kontinualne vrednosti.

- Orezivanje: CART uključuje korak orezivanja kako bi sprečio prenaučenost i poboljšao generalizaciju.

CART algoritam se ističe svojom sposobnošću rada kako za klasifikaciju tako i za regresiju, pružajući binarna stabla i prilagođavanje nečistoći ili varijansi u čvorovima. Korišćenjem Gini nečistoće ili varijanse kao kriterijuma odabira atributa, kao i kroz korake orezivanja radi kontrole prenaučenosti, CART postiže precizne modele odlučivanja i predviđanja.

CHAID

CHAID (Chi-square Automatic Interaction Detector) analiza je algoritam koji se koristi za otkrivanje veza između kategoričke ciljne promenljive i drugih kategoričkih prediktorskih promenljivih. Koristan je prilikom traženja obrazaca u skupovima podataka sa velikim brojem kategoričkih varijabli i predstavlja pogodan način sumiranja podataka jer se veze lako mogu vizualizovati.

Glavne karakteristike CHAID analize:

- Otkrivanje veza između kategoričkih varijabli: CHAID se fokusira na identifikaciju statistički značajnih veza između kategoričkih varijabli, posebno između ciljne promenljive i ostalih prediktorskih varijabli.

- Prilagođenost za kategoričke podatke: Algoritam je posebno prilagođen za rad sa kategoričkim podacima, što ga čini efikasnim za analizu skupova podataka koji sadrže više kategoričkih varijabli.

- Automatsko otkrivanje interakcija: CHAID automatski otkriva interakcije između različitih kategoričkih varijabli, pomažući u identifikaciji složenih veza i uzajamnih uticaja među njima.

- Vizualizacija veza: Jedna od prednosti CHAID analize je mogućnost lakoće vizualizacije rezultata. Odnosi i veze između kategoričkih varijabli mogu se jasno prikazati kroz dijagrame i grafikone.

CHAID analiza je korisna za istraživanje skupova podataka sa više kategoričkih varijabli, jer omogućava brzo otkrivanje veza i interakcija između tih varijabli. Vizualizacija rezultata olakšava interpretaciju rezultata i razumevanje kompleksnih odnosa među varijablama.

IMPLEMENTACIJA

Trening set

Za praktičan deo rada je napravljen trening set podataka koji određuje da li će student uspeti da upiše Elektronski faklutet ili ne. Atributi koji utiču na rezultat su broj godina kandidata, prethodni ostvareni uspeh u školovanju, da li je kandidat imao sve najviše ocene do sada, škola koju je pohađao i grad odakle dolazi.

elfak.csv > data

```
years,score,perfect score,school,city,enter elfak
18,excellent,yes,high school,Nis,yes
19,excellent,yes,high school,Nis,yes
19,excellent,no,medical,Leskovac,no
18,very good,no,high school,Vranje,yes
19,very good,no,tehnical,Nis,no
18,very good,no,other,Leskovac,yes
19,excellent,yes,high school,Nis,yes
19,excellent,no,tehnical,Nis,no
19,good,no,high school,Pirot,yes
20,very good,no,tehnical,Nis,yes
19,good,no,other,Nis,no
19,very good,no,high school,Nis,yes
18,excellent,yes,high school,Vranje,no
19,very good,no,tehnical,Leskovac,yes
```

Slika 4. Izgled trening seta

Preprocesiranje podataka

Proces preprocesiranja podataka bitan korak u analizi podataka, neophodan za obezbeđivanje tačnosti i bolje upotrebljivosti skupa podataka. Najčešće podaci koji su prikupljeni nisu idealni, u smislu da mogu postojati duplikati, određeni podaci mogu nedostajati, iste vrednosti mogu biti predstavljene na različite načine što može zbuniti algoritam koji koristimo i iz tog razloga je bitno odraditi dobru analizu podataka i eleminisati maksimalno moguće faktore koje mogu uticati na preciznost i validnost ulaznog seta podataka. Na konkretnom primeru korišćenog seta podataka su primenjene sledeće tehnike:

- Brisanje duplikata:

  Potrebno je identifikovati i ukloniti duplikate redova ukoliko postoje u skupu podataka. Ukoliko postoje mogu negativno uticati na analizu. Za uklanjanje duplikata je korišćena metoda *drop_duplicates* iz pandas bibliteke.

- Nedostajuće vrednosti:

  Nedostajuće vrednosti je potrebno obraditi kako bi se umanjio njihov potencijalni uticaj na analizu. Metoda *dropna* iz biblioteke pandas se koristi za uklanjanje redova koji sadrže nedostajuće vrednosti u bilo kojoj koloni. Pored ovog pristupa moguće je

koristiti metode koje će nedostajuće vrednosti popuniti zadatim vrednostima. (pr. *fillna*)

- Standardizacija tekstualnih podataka:

  Tekstualni podaci unutar određenih kolona standardizuju se radi unapređenja uniformnosti i olakšanja komparativnih analiza. Tehnika standardizacije u ovom slučaju se odnosila na smanjenje svih karaktera na mala slova.

- Konverzija kategoričkih promenljivih:

  Binarni kategorički atributi, poput 'perfect_score' i 'enter_elfak', transformišu se u binarni format kako bi se olakšale računarske operacije. Upotrebom funkcije map, kategoričke vrednosti 'yes' i 'not' mapiraju se na logičke vrednosti (True i False).

  Prilikom upotrebe algoritma za klasifikaciju iz sklearn bibiloteke koji ne mogu da rade sa kategoričkim vrednostima je iskorišćen *OneHotEncoder* način konverzije podataka u numeričke. One-hot enkodiranje u mašinskom učenju predstavlja konverziju kategoričnih informacija u format koji se može proslediti algoritmima mašinskog učenja radi poboljšanja tačnosti predikcije. One-hot enkodiranje je uobičajena metoda za rad sa kategoričnim podacima u mašinskom učenju.

Analiza i vizualizacija podataka

Radi lakše analize podataka korišćene su biblioteke koje omogućavaju lakši pregled podataka, kao i zavisnosti između njih.

Na slici 5 je dat primer iscrtavanja histograma za kolonu broj godina. Histogram je iscrtan korišćenjem naredbe:

preprocessed_data.hist('years', bins=range(18, 23, 1))

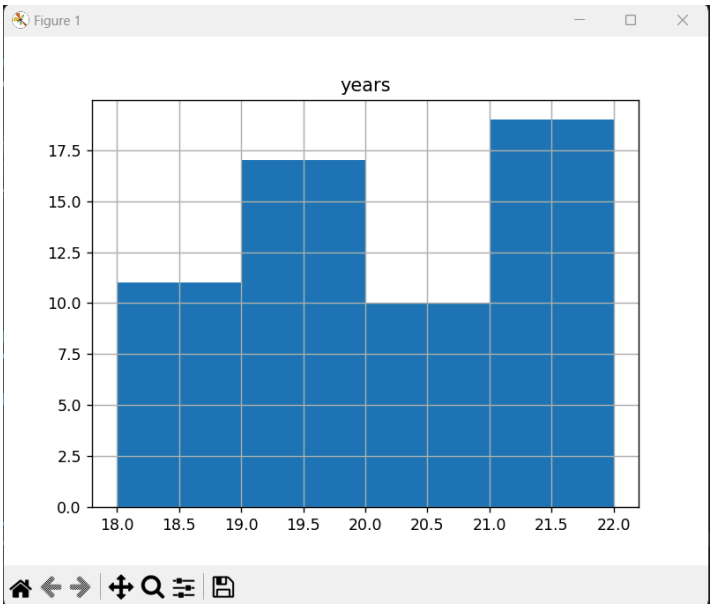
Ova naredba prikazuje histogram distribucije godina iz kolone 'years' u preprocesiranom datasetu 'preprocessed_data'. Histogram je grafički prikaz koji pokazuje koliko često se određene godine pojavljuju u skupu podataka. Na x-osi histograma su godine, dok je na y-osi broj pojavljivanja. Metoda pripada pandas bibloteci.

Na slici 6 je dat primer iscrtavanja heatmap-e za kolone uspeh i škola iz koje dolazi kandidat. Iscrtana je korišćenjem sledećih naredbi:

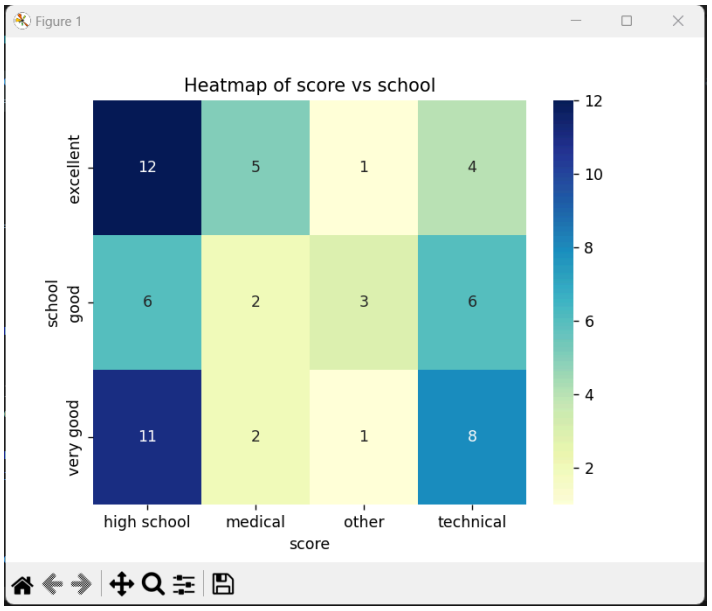
```
cross_tab = pd.crosstab(data['score'], data['school'])
sns.heatmap(cross_tab, annot=True, cmap='YlGnBu', fmt='d')
plt.xlabel('score')
plt.ylabel('school')
plt.title('Heatmap of score vs school')
```

Ovaj kod kreira unakrsnu tabelu pomoću funkcije *pd.crosstab()* iz pandas biblioteke. Unakrsna tabela prikazuje brojnost kombinacija vrednosti između dve kategoričke

promenljive, u ovom slučaju 'score' i 'school'. Zatim, koristi se seaborn biblioteka za iscrtavanje heatmap-e (toplotne mape) koja vizualizuje unakrsnu tabelu. Na x-osi heatmap-e su vrednosti promenljive 'score', na y-osi su vrednosti promenljive 'school', a svaka ćelija heatmap-e prikazuje brojnost odgovarajuće kombinacije vrednosti. Argument *annot=True* dodaje broj u svaku ćeliju kako bi se preciznije prikazala vrednost. Argument *cmap='YlGnBu'* definiše boju koja se koristi za prikaz vrednosti u heatmap-u. Formatiranje *fmt='d'* definiše da se vrednosti prikazuju kao celi brojevi. Na kraju, postavljaju se oznake x-osi, y-osi i naslov grafikona kako bi se dodatno objasnili prikazani podaci.



Slika 5. Primer iscrtavanja histograma za kolonu broj godina



Slika 6. Primer iscrtavanja hitmape

Implementacija ID3 algoritma

Pored trening seta implementiran je ID3 algoritam koji na osnovu trening seta pravi stablo odlučivanja i vrši klasifikaciju podataka kako bi se pokazalo kako se stablo odluke može primeniti na realnom primeru.

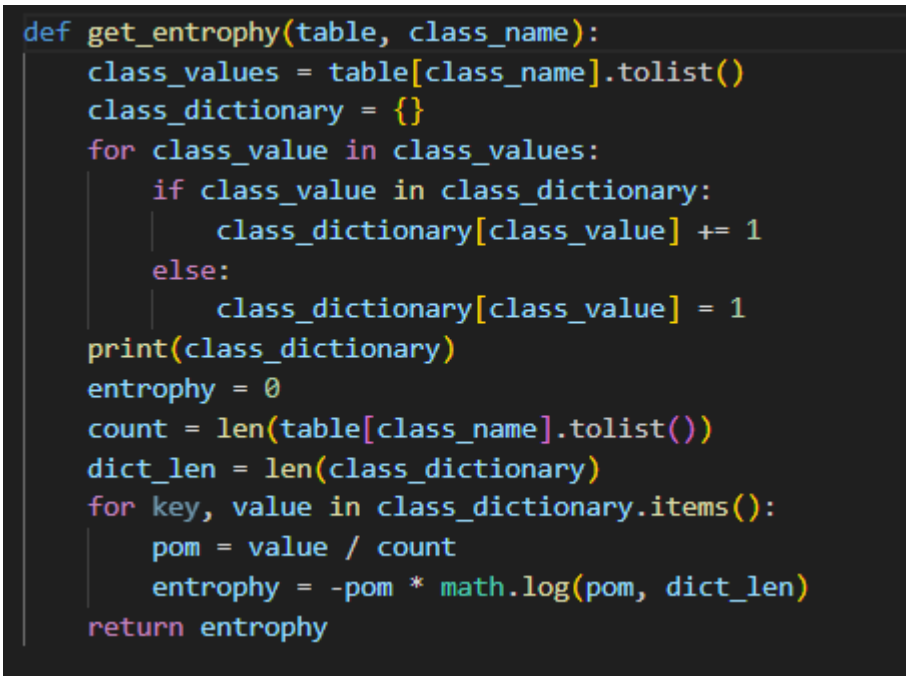
```
def id3_algorithm(table, class_name, node):
    print(table)
    if check_if_its_the_leaf(table, class_name):
        return TerminalNode(None, table[class_name].tolist()[0])
    else:
        highest_gain_attribute = get_highest_gain_attribute(table, class_name)
        print(highest_gain_attribute)
        all_attribute_values = get_all_attribute_values(table, highest_gain_attribute)
        node = Node(table, highest_gain_attribute)
        children_names = []
        children_names_dict = {}
        count = 0
        for i in all_attribute_values:
            t1 = table[table[highest_gain_attribute] == i]
            t2 = t1.drop(columns=highest_gain_attribute)
            child = id3_algorithm(t2, class_name, node)

            if child.name not in children_names:
                node.add_child_and_branch(child, i)
                children_names.append(child.name)
                children_names_dict[child.name] = count
                count += 1
            else:
                current_name = str(node.branches[children_names_dict[child.name]])
                current_name = current_name + " or " + str(i)
                node.branches[children_names_dict[child.name]] = current_name
        return node
```

Slika 7. Implementacija ID3 algoritma

Metoda id3_algorithm prvo proverava da li je trenutni skup podataka moguće direktno klasifikovati kao terminalni čvor. Ako jeste, vraća se odgovarajući terminalni čvor sa odgovarajućom vrednošću klase. U suprotnom, metoda nastavlja rad. Prvo identifikuje atribut sa najvećom informacionom dobiti. Zatim, kreira novi unutarnji čvor stabla sa imenom identifikovanog atributa.

Sledeći korak je iteracija kroz sve vrednosti ovog atributa. Za svaku vrednost, metoda rekurzivno poziva samu sebe na podskupu podataka koji odgovara toj vrednosti atributa. Rekurzivni pozivi generišu decu trenutnog čvora, odnosno podstablo koje se grana na osnovu vrednosti atributa.

Ako se tokom iteracije naiđe na vrednost atributa koja već postoji kao dete u trenutnom čvoru, tada se vrši spajanje grana kako bi se izbeglo dupliranje informacija u stablu. Na kraju, metoda vraća formirani čvor, tj. stablo odlučivanja.



```
def get_entrophy(table, class_name):
    class_values = table[class_name].tolist()
    class_dictionary = {}
    for class_value in class_values:
        if class_value in class_dictionary:
            class_dictionary[class_value] += 1
        else:
            class_dictionary[class_value] = 1
    print(class_dictionary)
    entrophy = 0
    count = len(table[class_name].tolist())
    dict_len = len(class_dictionary)
    for key, value in class_dictionary.items():
        pom = value / count
        entrophy = -pom * math.log(pom, dict_len)
    return entrophy
```

Slika 8. Implementacija metode za računanje entropije

Metoda sa imenom get_entropy, ima zadatak da izračuna entropiju, mernu vrednost nečistoće u okviru skupa podataka. Metoda ima dva parametra: table i class_name. table predstavlja tabelu podataka, dok class_name predstavlja naziv klase koju algoritam pokušava da predvidi.

Metoda prvo pribavlja sve vrednosti klase iz datog skupa podataka. Zatim, kreira dictionary u kom se broji pojavljivanje svake jedinstvene vrednosti klase. Nakon toga, metoda prolazi kroz dictionary i izračunava entropiju koristeći formulu za izračunavanje entropije.

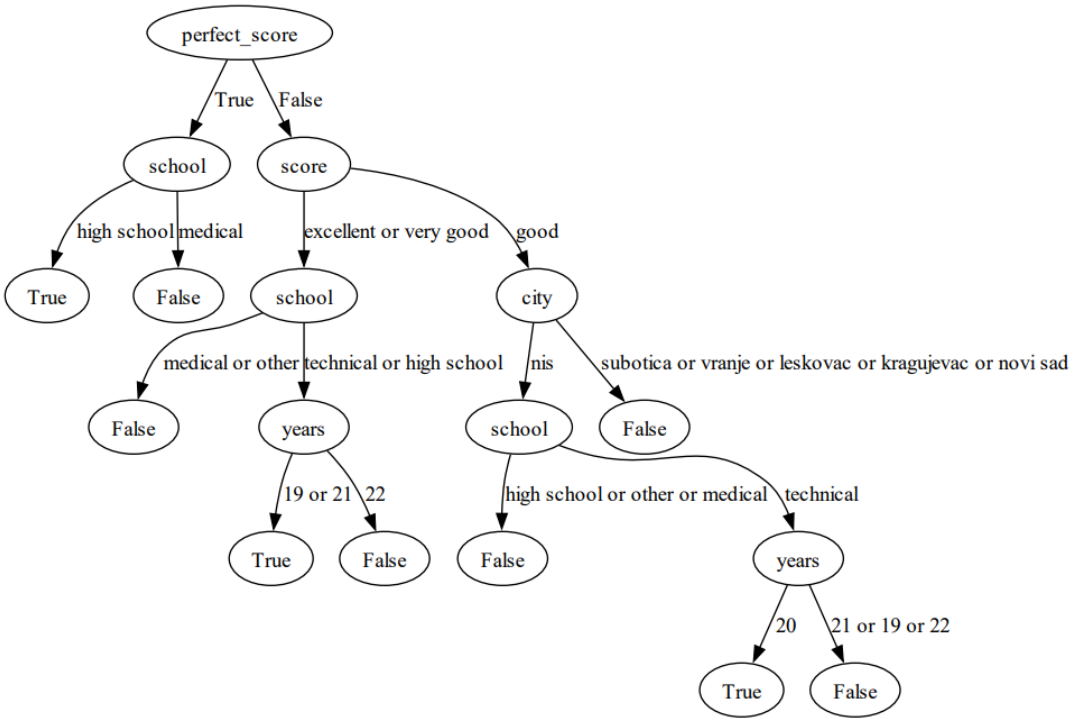
Metoda, nazvana get_gain_for_attribute, ima zadatak da izračuna informacionu dobit za određeni atribut u okviru stabla odlučivanja. Metoda ima tri parametra: table, class_name, i column. table i class_name imaju istu ulogu kao u predhodnoj metodi dok parametar column predstavlja naziv atributa za koji se računa informaciona dobit.

Metoda prvo vrši inicijalizaciju prazanog rečnika column_class_dict koji će čuvati informacije o raspodeli klasa za svaku vrednost atributa. Zatim, prolazi kroz redove tabele i popunjava rečnik podacima. Za svaki red u tabeli, metoda prati broj pojavljivanja svake vrednosti atributa (column_value). Ako vrednost već postoji u rečniku, povećava se brojač, a ako ne postoji, inicijalizuje se nova struktura podataka za tu vrednost atributa. Takođe, prati se broj pojavljivanja svake klase za svaku vrednost atributa. Nakon popunjavanja ovog rečnika, metoda izračunava informacionu dobit po formuli. Nakon izračunavanja, metoda vraća izračunatu informacionu dobit.

```python
def get_gain_for_attribute(table, class_name, column):
    column_class_dict = {}
    global data
    count = 0
    for ind in table.index:
        class_division_dict = init_class_dictionary(data, class_name)
        column_value = table[column][ind]
        if column_value in column_class_dict:
            column_class_dict[column_value]["count"] += 1

        else:
            column_class_dict[column_value] = {}
            column_class_dict[column_value]["count"] = 1
            column_class_dict[column_value]["classes"] = class_division_dict
        class_value = table[class_name][ind]
        if class_value in column_class_dict[column_value]["classes"]:
            column_class_dict[column_value]["classes"][class_value] += 1
        else:
            column_class_dict[column_value]["classes"][class_value] = 1
        count += 1
    print(column_class_dict)
    gain = 1
    for attribute_value in column_class_dict:
        local_count = 0
        # print(attribute_value)
        H = 0
        for class_value in column_class_dict[attribute_value]["classes"]:
            local_count += 1
            pom = int(column_class_dict[attribute_value]["classes"][class_value]) / int(
                column_class_dict[attribute_value]["count"]
            )
            # print()
            if pom != 0:
                H += -pom * math.log(pom, 2)
                H = round(H, 2)
                # print(H)
        gain -= local_count / count * H
    # print(gain)
    return gain
```

Slika 9. Implementacija metode za računanje informacione dobiti

Nakon pokretanja algoritma sa trening setom koji je prikazan dobija se stablo odluke sa slike 10.



Slika 10. Izgled dobijenog stabla odluke

Korišćenje scikit-learn bibiloteke

Scikit-learn (Sklearn) predstavlja najkorisniju i najrobustniju biblioteku za mašinsko učenje u Python programskom jeziku. Pruža širok izbor efikasnih alata za mašinsko učenje i statističko modeliranje, uključujući klasifikaciju, regresiju, klasterizaciju i smanjenje dimenzionalnosti putem već implementiranih metoda. Ova biblioteka se zasniva na NumPy, SciPy i Matplotlib bibilotekama.

Biblioteka je primenjena za demonstraciju CART algoritma i algoritma slučajne šume.

U oba slučaja se vrši učitavanje seta podataka preko *pandas* biblioteke i vrši se već opisana procedura preprocesinga podataka. Metoda *train_test_split* deli set podataka na nasumični podskup za trening i test.

Kod CART algoritma se se kreira instanca *DecissionThreeClassifier* klase čija je implementacija u pozadini zapravo CART algoritam, a zatim se koristi funkcija *fit()* kako bi se kreiralo stablo odlučivanja na trening skupu podataka (*X_train_encoded*) i odgovarajućim ciljnim vrednostima (*y_train*). Nakon obučavanja, algoritam se koristi za predviđanje ciljnih vrednosti za testni skup podataka (*X_test_encoded*) pozivanjem funkcije *predict()*. Na kraju, funkcija *accuracy_score()* se koristi za izračunavanje tačnosti predikcije tako što se uporede predviđene vrednosti (*predicted_dataset*) sa stvarnim ciljnim vrednostima testnog skupa podataka (*y_test*). Rezultat se ispisuje kao tačnost modela.

```
# CART
data = pd.read_csv("elfak.csv")
preprocessed_data = preprocess_dataset(data)

y = data[prediction_class]
X = data.drop(prediction_class, axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

encoder = OneHotEncoder(handle_unknown='error')
X_train_encoded = encoder.fit_transform(X_train,)
X_test_encoded = encoder.transform(X_test)
      You, 2 days ago • extend solution with CART and random forest alg…
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train_encoded, y_train)
predicted_dataset = decision_tree.predict(X_test_encoded)
print('CART: ', accuracy_score(predicted_dataset, y_test))
```

Slika 11. Primena CART algoritma

U drugom primeru je demonstriran rad algoritma slučajne šume. Algoritam slučajne šume kreira više stabla odluke koja se spajaju radi bolje preciznosti. Metod slučajnih šuma se zasniva na prostoj agregaciji stabala odlučivanja. Šuma se sastoji od m stabala treniranih na različitim podskupovima skupa za obučavanje. Jedno stablo se obučava tako što se izabere podskup skupa za obučavanje odredene velčine, pri čemu je moguće koristiti i samo podskup ukupnog skupa atributa. Stabla se obučavaju na različitim podskupovima kako bi njihove greške bile što slabije korelisane, što ostavlja prostor za popravku agregacijom.

Broj stabala se može posmatrati kao parametar sa svojstvom da visoke vrednosti obično daju bolje rezultate. Naravno, po cenu računskog vremena. Slučajna šuma je jedan od najprimenjenijih algoritama mašinskog učenja. Njegovo obučavanje je relativno efikasno, a preciznost predvidanja obično medu najboljim za vektorski predstavljene podatke.

```
# RANDOM FOREST
data = pd.read_csv("elfak.csv")
preprocessed_data = preprocess_dataset(data)

y = data[prediction_class]
X = data.drop(prediction_class, axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
random_forest_classifier = RandomForestClassifier(n_estimators=100)

encoder = OneHotEncoder(handle_unknown='error')
X_train_encoded = encoder.fit_transform(X_train,)
X_test_encoded = encoder.transform(X_test)

random_forest_classifier.fit(X_train_encoded, y_train)
predicted_dataset_rf = random_forest_classifier.predict(X_test_encoded)
print('Random forest: ', accuracy_score(predicted_dataset_rf, y_test))
```

Slika 12. Slučajna šuma

Rezultat

Nakon izvršenja program štampa brojeve koji označavaju tačnost svakog od algoritma u predviđanju rezultata. Očekivano se vidi da ID3 algoritam daje najslabije rezultate (80%), nakon njega je uspešniji CART algoritam sa 84% i najefikasniji je algoritam slučajne šume koji daje tačnost predviđanja od 92%.

```
ID3:  0.8
CART:  0.8461538461538461
Random forest:  0.9230769230769231
```

Slika 13. Rezultati izvršenja programa

ZAKLJUČAK

Kroz ovaj rad koji se bavi algoritmima stabla odlučivanja opisane su najvažnije karakteristike ovih algoritma i način na koji oni funkcionišu. Pored detaljnije analize ID3 algoritma i njegove praktične implementacije pažnja je posvećena i ostalim algoritmima stabla odlučivanja kako bi se stekla bolja slika ovih algoritma i načina na koji oni funkcionišu. Takođe, cilj je bio i da se prikaže koje su mane i prednosti svakog od ovih algoritma kao i koje su oblasti primene svakog od njih.

Kroz teorijski deo rada analizirani su osnovni principi ID3 algoritma, uključujući pojmove kao što su entropija i informaciona dobit kao ključni pojmovi. Kroz ovu analizu, cilj je bio razumeti kako ID3 algoritam efikasno razdvaja podatke i gradi stablo odlučivanja, čime omogućava klasifikaciju podataka.

Praktični deo rada prikazao je konkretnu implementaciju ID3 algoritma, koristeći podatke iz realnog sveta. Ovaj deo rada je detaljnije prikazao tehnički deo implementacije samog algoritma u programskom jeziku Python.

Analizom srodnih algoritama poput C4.5, CART i CHAID obrađeni su i drugi algoritmi stabla odlučivanja koji se mogu primeniti za klasifikaciju ili regresiju. Razumevanje prednosti i ograničenja ovih algoritama je vrlo važno prilikom odabira algoritma za rešavanje konkretnih problema.

Kao ideja za nastavak istraživnja na ovu temu može se odraditi praktična implementacija ostalih opisanih algoritama i napraviti poređenje u rezultatima koji su dobijeni.

LITERATURA

1. *Medium,*
https://medium.com/@ashirbadpradhan8115/decision-tree-id3-algorithm-machine-learning-4120d8ba013b

2. *Medium,*
https://medium.com/@satyarepala/a-deep-dive-into-decision-tree-algorithms-classification-regression-and-beyond-d2cf67d6f814

3. *IBM,* https://www.ibm.com/topics/decision-trees

4. *Medium,*
https://medium.com/@anirudhsreekumar98/decision-tree-id3-algorithm-f74875fa507d

5. *Analytics Vidhya*
https://www.analyticsvidhya.com/blog/2021/05/implement-of-decision-tree-using-chaid/

6. *FON*
https://ai.fon.bg.ac.rs/wp-content/uploads/2015/04/Klasifikacija-Stabla-odlucivanja-2015.pdf