Automatizirano odlučivanje pomoću DMN-a: od stabla odluka do tablica odluka

Buljubašić, Lea

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

LEA BULJUBAŠIĆ

Automatizirano odlučivanje pomoću DMN-a: od stabla odluka do tablica odluka

ZAVRŠNI RAD

Pula, srpanj 2021.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

LEA BULJUBAŠIĆ

Automatizirano odlučivanje pomoću DMN-a: od stabla odluka do tablica odluka

ZAVRŠNI RAD

JMBAG: 0303007517
Studijski smjer: Informatika
Kolegij: Poslovni informacijski sustavi
Mentor: doc.dr.sc. Darko Etinger

Pula, srpanj 2021.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani_____, kandidat za prvostupnika _____ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

_____

Student

U Puli, _____

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja,_____dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj Završni rad pod nazivom

_____

_____

koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____

Potpis

_____

Sadržaj

Uvod

Poslovni ljudi na dnevnoj bazi se susreću s raznim problemima i sukladno tome moraju svakodnevno donositi razne poslovne odluke koje su većinom operativnog karaktera i koje znaju biti pogrešne zbog subjektivnih procjena. Fokusiranjem na poslovne odluke započinju uspješna poslovna pravila i analitički projekti.

Transparentan poslovni cilj je u analitičkim projektima ključan za uspjeh, a najveći uspjeh i najbolji rezultati zamijećeni su kod poslovnih odluka koje se mogu jednostavno povezati s svojim ciljevima.

Analizu poslovnih pravila definira jasnoća o zahtjevima za donošenje odluka, međutim pri toj analizi ishod je često velika količina podataka koju je teško nadzirati. Zbog toga, kada se analiza poslovnih pravila realizira, donošenje i razumijevanje poslovnih odluka puno je jednostavnije. Kako bi se takve poslovne odluke definirale i razumjele, poslovni analitičari vrše modeliranje odluka. Modeliranje odluka se pri kvalitetnoj analizi poslovanja pokazalo kao ključna metoda koja povezuje analitiku i poslovna pravila te pruža agilnost. Ono je također ključno pri upravljanju poslovnim odlukama jer osigurava detaljniju tehničku dokumentaciju koju kao predložak u nekoj poslovnoj organizaciji određeni timovi mogu koristiti i time ubrzava proces upravljanja. S ciljem da se pruži konstrukte potrebne za modeliranje odluka i da se poslovne odluke, od strane menadžera, mogu prikazati pomoću dijagrama koristi se DMN model i notacija odluke.

Strojno učenje u poslovanju

Strojno učenje i umjetna inteligencija dio su trenutno najaktivnijih i najuzbudljivijih područja računalne znanosti koji obuhvaćaju razne poslovne potrebe kao što su automatizacija, analiza podataka i donošenje odluka. Ti sustavi uspijevaju učiti iz svoje okoline i prilagođavati se novim prilikama i zadacima čime poboljšavaju svoju kvalitetu i učinkovitost. Smatra se da nam tehnologije strojnog učenja mogu uvelike pomoći jer nam štede vrijeme analize podataka, pružaju objektivan, cjelovit i točan rezultat čime nam omogućavaju puno veće znanje o podacima iz baze podataka.

U nastavku predstaviti ću nekoliko primjera u kojima razni autori pokušavaju riješiti poslovnu problematiku firmi uz pomoć strojnog učenja. Neki od problema s kojim se suočavaju su ekonomska problematika, procjena kreditnog rizika i problematika višeslojne ravnoteže.

Rezultati se kod mnogih modela umjetne inteligencije ne mogu objasniti, iz razloga što znaju biti nerazumljivi i neprozirni. To znači da ih se može naučiti da izvršavaju određene zadaće, međutim sam sustav nam neće moći svaki put pokazati zbog čega je rezultat baš takav. Razlog tome je što kod većine modela strojnog učenja visokih performansi ulazni podaci obrađuju na osnovu matematičkih apstrakcija i ne mogu prikazati procese ljudskog odlučivanja.
Za navedeni problem autori [1] predlažu četiri rješenja:

1. Obrazloženje vođeno modelom – u ovom pristupu autori [1] predlažu da se istrenira još jedan neproziran model, koji će uz obučavanje prvog neprozirnog modela objasniti potrebne procese. Za predviđanje ishoda koristi se neproziran model strojnog učenja pomoću kojeg se trenira drugi neproziran model. Oba modela se paralelno koriste. Prvi nudi točan rezultat, a drugi popratno objašnjenje.
   Ovo rješenje ima nekoliko nedostataka, a to su da niti jedan set nam ne može osigurati da ishodi, prvog i drugog modela, budu usklađeni; i dalje se ne razumije, s obzirom da su oba modela neprozirna, kako funkcioniraju u određenim slučajevima izvan rezultata; nerijetko se oznake objašnjenja moraju ručno

izvoditi, što otežava obuku modela; ponekad je teško naučiti set za učenje da obuhvati sve moguće scenarije s objašnjenjima.

2. Vrijednost perturbacije – kod ovog pristupa autori [1] predlažu da se na ulazu uskladi svaka značajka ulaza zasebno, kako bi se moglo saznati koliko i koja je razrada potrebna za drugačiji rezultat.

   Ovaj pristup koristi LIME za ometanje značajki ulaza kako bi se vidjelo koji će od njih utjecati na rezultat. Ne traži obuku drugog modela i može se upotrijebiti kod bilo kojeg modela strojnog učenja za predviđanje.

   Nedostaci ovog pristupa su ti da, s obzirom da je model neproziran, ne može se znati kako on funkcionira izvan rezultata određenih slučajeva; kako bi se ispitala njihova važnost, količina ulaznih podataka se treba enormno povećati i svaki se slučaj treba ponavljati zajedno sa smetnjama za svaku pojedinu značajku.

3. Korištenje ansambla – u ovom pristupu trebalo bi za ulazni skup podataka primjenjivati različite i prozirne modele strojnog učenja. Cilj ovog pristupa je dati rezultat i objašnjenje koje se koristi samo u slučaju kada su rezultati prozirnih i neprozirnih modela usklađeni. U slučaju da rezultat i objašnjenje nisu usklađeni, imamo dvije opcije, a to su:

   i. Rezultati i objašnjenje su usklađeni – tu je objašnjenje prikazuje na koji je način odluka donesena;
   ii. Rezultati i objašnjenje nisu usklađeni – u ovom slučaju od objašnjenja nemamo koristi, a i rezultat nam može biti nebitan.

   Kod ovog pristupa je cilj da se svi ulazni podaci obučavaju s istim podacima, ali pri tom se oni usredotočuju na različite karakteristike podataka, s ciljem da se poveća točnost modela. Ovaj sistem se koristi u slučajevima kad se od stabla odlučivanja žele stvoriti slučajne šume.

   S pretpostavkom da su članovi dovoljno raznoliki, pouzdanost ovog pristupa je veća od one njegovog najboljeg člana.

4. Predviđanje na temelju objašnjenja – u ovom pristupu trebalo bi upotrebljavati AI modele koji objašnjavaju svoj rezultat, jer bi tada temeljni aspekt u procesu predviđanja bila izgradnja obrazloženja.

   U ovom pristupu objašnjenja su nastala kao dio procesa predviđanja bez nadoknade.

Prednosti ovog pristupa su te da je točno definirano koji je slučaj doveo do rezultata; i vrijeme i točnost se većinom poklapaju s ostalim neuronskim mrežama iste vrste. Nedostaci ovog pristupa su te da ovaj način ne daje statistički pregled odlučivanja, već samo obrazloženje od slučaja do slučaja; i funkcionira samo sa jedinstvenom primjenom neuronskih mreže

Sustav utemeljen na znanju generira informacije čime su donesene odluke djelotvornije i objektivnije, što može uvelike pomoći poslovnjacima da donesu bolje odluke pri raznim poslovnim predviđanjima.
U radu [2] autori su na određeni ekonomski problem primijenili nekoliko modela strojnog učenja i predlažu XGBoost metodu koja je namijenjena poduzećima, i koja će na osnovu unesenih podataka pomoći pri donošenju odluka kod predviđanja učinkovitosti inovacija.
Modele koje su koristili pri utvrđivanju faktora učinkovitosti inovacija su: linerarna regresija, stablo odlučivanja, slučajne šume, neuronske mreže i XGBoost modeli.
Istraživanja autora [2] pokazuju kako je XGBoost prestigao ostala četiri modela sa učinkovitošću od 73,65% za R&D učinkovitost, 70,02% za komercijalnu učinkovitost i 70,09% za ukupnu učinkovitost, te je također bolji za tumačenje. Stoga smatraju da je XGBoost omogućava poslovnim ljudima najbolje predviđanje učinkovitosti inovacija te izgradnju inteligentnog sustava za podršku odlučivanju.
S obzirom da se trenutne metode strojnog učenja koriste u korporativnom upravljanju, većinom su orijentirane na proizvodni proces. Zbog toga autori [2] smatraju da njihova metoda XGBoost može uvelike pomoći tvrtkama u izgradnji inteligentnog sustava za podršku odlučivanju, te poslovnim ljudima da uz pomoć XGBoost metode donose poslovne odluke i povećaju svoju učinkovitost u konkurenciji s drugim tvrtkama.

Rad [3] se temelji na procjeni kreditnog rizika koristeći kombinaciju nadziranog i nenadziranog učenja, s time što tehnike nenadziranog učenja primjenjuju u dvije faze, a to su:

1. Izgradnja modela konsenzusa
2. Skupovi podataka klastera

A usporedbe izvedbe modela provode se u četiri različite kombinacije integracije:

a. Pojedinačni modeli

b. Pojedinačni modeli + model konsenzusa
c. Klasteriranje + individualni modeli
d. Klasteriranje + individualni modeli + model konsenzusa

Rezultati istraživanja ukazuju kako je za unapređenje interpretacije kreditnog bodovanja najdjelotvornija integracija u fazi klasteriranja ili fazi konsenzusa, premda varijanta u kojoj se kombiniraju obje faze pripomaže uspješnosti modela i na taj način se postižu najbolji rezultati. Autori ovog rada [3] predlažu efikasni algoritma klasteriranja bez nadzora pomoću kojeg bi se sinkronizirano odvajali podaci i nakon toga koristili se za model učenja pod nadzorom Sve to s ciljem da se istraži može li se na taj način unaprijediti model kreditnog bodovanja. Kako bi što više unaprijedili model kreditnog bodovanja, autori [3] su u radu fokusirali se na povezivanje nenadzornih metoda maksimalnih vrijednosti s nenadziranim klasifikatorima maksimalnih vrijednosti u različitim kombinacijama integracije (pojedinačni modeli; pojedinačni modeli + model konsenzusa; klasteriranje + individualni modeli; klasteriranje + individualni modeli + model konsenzusa). Većinom pri prikupljanju podataka jedan određeni dio podatak ne može biti prikupljen od svih klijenata i dio podataka time ostaje prazan, ta nepotpunost podataka često je ključni problem pri procjeni kreditnog rizika, što se u određenim situacijama može iskoristiti za optimizaciju podataka i na taj način poboljšati važne karakteristike modela.

Rad [4] se fokusira na problematiku višeslojne neravnoteže koja se nalazi u mnogim praktičnim područjima, i za koju se smatra je je značajnija od problema neravnoteže binarnih klasa.
Algoritmima strojnog učenja najveći problem su preopterećenje i skupovi podataka s iskrivljenom distribucijom klasa. Do takvih skupova podataka dolazi iz razloga što uobičajeni klasifikatori za cilj prvobitno imaju rješavanje problema s klasifikacijom ravnoteže, i njima svi uzorci imaju identičnu bitnost te time ignoriraju manje klase. Autori u radu [4] koristeći vrijednosti parametara regulacije i klase proporcije vrijednosti određuju klasične specifične parametre regulacije. Oni na osnovu rezultata predlažu unapređenje klasičnog kernel ekstremnog stroja za učenje (CSKLM) s ciljem rješavanja problema s više nejednakih klasa. U tom bi se slučaju koristio generalizirani klasični kernel ekstremni stroj za učenje (GCSKLEM), koji u kombinaciji s funkcijom gustoće normalne slučajne varijable jezgre ignorira problem neefikasnog skrivenog

čvora, a koji je povezan sa trenutnim verzijama klasičnog ekstremnog stroja za učenje (CS-ELM). Kao dodatnu prednost ove metode autori [4] također navode i manje financijske troškove u usporedbi s kernel ponderiranim strojem za ekstremno učenje (KWELM).

Automatizirano odlučivanje pomoću DMN-a

Kako bi unaprijedili svoje poslovanje i poboljšali analizu poslovanja poslovni ljudi moraju izraditi model zahtjeva za donošenje odluka pomoću DMN-a. Za grafički prikaz u DMN tablicama koriste se pravokutnici za prikaz odluka, pravokutnici sa izrezanim kutovima za modele poslovnog znanja i ovali koji predstavljaju unos podataka. DMN standard prikazuje uključena pravila u studiji slučaja, njegove prednosti su mnogobrojne, a ključne su te da se nudi fleksibilan i ponovljiv pristup u upravljanju i određivanju zahtjeva za donošenjem odluka. DMN za cilj također ima odlučivanje koje se rješava na dva načina, a to su modeli poslovnih procesa i model logike odluke. Modeli poslovnih procesa se koriste za definiranje određenih ciljeva unutar kojih se odlučuje. Opisuju ih poslovni analitičari, a definiranje se vrši pomoću grafikona koji se sastoji od skupa elemenata, pravila povezivanja te dijagrama zahtjeva koji služe kao bilješka. Model logike odluke koristi FEEL jezik i služi za definiranje specifične logike u donošenju pojedinačnih odluka. FEEL je „prijateljski“ jezik koji koristi IF/ELSE logiku, a njegova svrha je da definira i sastavlja tablicu odlučivanja, da radi jednostavne strukture podataka i izračune, te da osigurava notaciju za logiku odlučivanja.
DMN je prvenstveno namijenjen poslovnim korisnicima, on osigurava konkretne izraze i tablice odluka pri označavanju logike odluke. Tablica odluke je ključan oblik poslovnog znanja jer prikazuje logiku odluke koja nudi korisne informacije o tome kako prezentirati određene značajke u modelu odlučivanja.

Da bi se realizirao model odlučivanja potrebno je sve odluke i modele poslovnog znanja u cijelosti definirati pomoću logike odlučivanja. Korištenje logike nanovo unutar procesa odlučivanja definira usluga odlučivanja, na način da element za ponovnu upotrebu prezentira jednu ili više odluka iz modela odlučivanja. Tada se s potrebnim ulaznim podacima pozove usluga i za rezultat se dobiva izlaz izloženih odluka.
U takvim situacijama dolazi do manjeg problema, jer unutarnja struktura donošenja odluka nije pogodno definirana. Zbog tog problema autori [5] predlažu dijagram zahtjeva, koji će spajati model poslovnih procesa i model logike odluke. Na taj način bi se zadaci rasporedili tako, da bi se u modelu poslovnih procesa definirali zadaci unutar kojih je potrebno odlučivati, u modelu logike odluke bi se vrlo detaljno definirale odluke pomoću kojih bi se radila validacija, a u dijagramu zahtjeva bi se definirali međusobni

odnosi i zahtjevi za logiku odlučivanja te također i buduće odluke u datim zadacima. Autori [5] smatraju da dijagram zahtjeva i model logike odluke paralelno mogu nadopuniti model poslovnih procesa i time omogućiti kompletan model odlučivanja. Na taj način bi se detaljno prikazalo donošenje odluka u procesnim zadacima, jer bi se omogućila unakrsna validacija, automatsko izvršavanje odluka, dizajn i automatizacija procesa odozgo prema dolje.

Kako je prije navedeno, bitna implementacija DMN-a je definiranje logike odlučivanja koja se automatizira korištenjem usluga odlučivanja. Kao podloga za planiranje projekta provedbe može se koristiti struktura modela odlučivanja, te bi se time, i s ciljem objedinjavanja definicije i logike komponenti modela odlučivanja, uključilo specifične projektne zadatke. Pri donošenju odluka potrebno je, u određenim situacijama, odrediti i specifične algoritme koristeći logiku odlučivanja. Te je također potrebno unijeti i ulazne podatke koji ponekad mogu biti i rezultati drugih odluka. S obzirom da su ulazni i izlazni podaci u odlukama povezani, informacije se mogu primijeniti čak i ako su ponuđene samo djelomične informacije odluka. To omogućava da se u različitim poslovnim situacijama može koristiti kombinacija različitih područja poslovnog znanja i alternativnih verzija logike odlučivanja s ciljem modeliranja složene logike odlučivanja. U slučaju da se logika odluke promijeni, ne bi se trebalo mijenjati model procesa, jer promjena modela odluke ne utječe na normativni proces i samim time on ostaje očuvan. Prema autorima [6] model odluka ima četiri važna koraka koja treba ponavljati dokle god se odluke u potpunosti se preciziraju:

1. Korak - odrediti odluke – tu se definiraju potrebne odluke kao dio ukupnog procesa prikupljanja zahtjeva
2. Korak - opisati odluke – ovo je sama bit DMN-a. U ovom koraku se uz pomoć dijagrama dokumentiraju odluke, i opisuje se kako bi to moglo pozitivno utjecati na poslovanje. Dijagram sadrži jednu ili više odluka, te o svakoj odluci definirane ključne informacije
3. Korak - navesti zahtjeve za odluke – u ovom koraku se u dijagram zahtjeva spajaju informacije (ulazni podaci) i potrebna znanja. Dijagram može biti namijenjen samo jednoj odluci, ali može također poslužiti i za nekoliko odluka (to se primjenjuje samo u slučajevima kada su te odluke blisko povezane)

4. Korak - razgraditi i poboljšati – ovo je završna faza u kojoj je cilj s određenim brojem iteracija poboljšati model odluke dokle god on ne bude razvijen do željene razine. Tu se na osnovu grafičkog dijagrama, u slučajevima kada su potrebne, mogu dodati i dodatne odluke.

Kod predstavljanja znanja, informacije se mogu koristiti na više načina jer su generalno prikazane. Za predstavljanje jedinstvenog poslovnog znanja na osnovu kojeg se odlučuje mogu se koristiti modeli poslovnog znanja, međutim tada se DMN koristi kao instrument za službeno definiranje zahtjeva za upravljanjem znanjem. Model poslovnog znanja određuje se pomoću izraza vrijednosti koji definira kako se izraz opisuje iz skupa ulaza. U tom slučaju izraz, vrijednosti je inicijaliziran kao formulacija funkcije i može se pozvati iz atributa vrijednosti odluke. U modele poslovnog znanja unosi se logika odluke, a rezultat povezan s odlukom definira kako se pozivaju modeli poslovnog znanja te kako se povezuju rezultati s ciljem izračunavanja izlazne odluke. Prednost modela poslovnog znanja se najviše ističe u situacijama kada razna poslovna znanja znaju biti međusobno isprepletena, tada se uz pomoću modela poslovnog znanja mogu međusobno povezati. Model poslovnog znanja može također uključivati i logiku odlučivanja, što u svrhu modeliranja logičkih odluka u DMN omogućava uvoz postojećih standarada.

Zastupanje i upravljanje odlukama imaju vrlo slične ciljeve jer su oboje zaokupljeni idealnim prikazivanjem znanja i fokusirane su na postizanje pouzdanosti predstavljenog znanja. Da bi se modeliralo upravljanje donošenjem odluka definiraju se izvori znanja, koji se mogu zajedno povezati u jednu poslovnu cjelinu. Izvori znanja služe za pisanje dokumentacije, dok se modeli poslovnih znanja odnose na implementaciju. Pri modeliranju DMN-a svako poslovno znanje kojem je, ili će biti, potrebno održavanje trebalo bi modelirati kao model poslovnog znanja u DRD-u, a kao izvore znanja tad modelirati odgovorno osoblje. Iz razloga što DRD osigurava tehničku dokumentaciju potrebnu za održavanje znanja, a inicijalnu konfiguraciju poslovnog znanja kojeg treba održavati definira logika odluke.

Za svaku firmu je vrlo zahtjevno modeliranje poslovnih pravila, a donošenje potrebne poslovne odluke osiguravaju različiti oblici zaključivanja koji su osigurani sustavom baze znanja.

Autori [7] predlažu moderan pristup u donošenju poslovnih pravila i odluka koristeći DMN. Svojim pristupom jamče interaktivno donošenje odluka na osnovu automatski generiranog interaktivnog grafičkog sučelja iz specifikacije odluka. Koriste sustav baze znanja IDP u kojem je znanje prikazano pomoću FO() jezika.
FO() jezik se bazira na IDP sustavu gdje su definicije sklopovi pravila koji ne nameću smjer računanja i vode se po tijelima. Prijevod s DMN-a na FO () ponudio bi korištenje DMN-a za modeliranje poslovnih odluka u kojima bi znanje bilo jedinstveno u DMN-u, a IDP bi se mogao koristiti kao pričuvni element za izvršavanje traženih zaključaka. Trenutno je izravni prijevod s DMN-a n FO() u tijeku i još uvijek se izvodi ručno, međutim postoji kompjutersko prevođenje tablica odluka u RULEML.
U FO() jezik se mogu prevesti znanja predstavljena DMN modelom. I s obzirom na to, dokazi prisutni u IDP sustavu postaju raspoloživi za upotrebu s DMN modelima odlučivanja. Ta prednost omogućava poslovnim korisnicima iskorištavanje sposobnosti općenito primjenjivih zaključaka za donošenje oblikovanih odluka i predstavljanje potrebnih znanja na poslovno orijentiran način. U tom slučaju tablica i naziv prikazuju isto znanje. U radu autori [7] opisuju kako se kao zaključak za DMN može koristiti IDP sustav znanja, te uz pomoć raspoloživih zaključaka IDP sustava prezentiraju moguća proširenja trenutnih rješenja za upravljanje odlukama.
IDP sustav kombinacijom tehnika iz SAT-a i programiranja uključuje višestruke funkcionalnosti. Pomoću IDP sustava možemo izračunati definirane koncepte iz poznatog unosa ili pak, obzirom na poznatu vrijednost za zadane simbole, izračunati vrijednosti za nepoznati ulaz. Njihov sustav šireći direktno korisnikov unos smjesta pokazuje implikacije izbora. Za predstavljanje pravila odlučivanja autori [7] su u svom pristupu koristili bazu znanja, a kao prednost svog modela autori [7] predstavljaju dostupnost više oblika zaključivanja. Što znači da se postojeće obrazloženje raspoloživo u IDP-u, preobrazbom iz DMN-a u FO(), može koristiti i za DMN modele, tj. isti model DMN odluke može se koristiti za donošenje različitih odluka.

Pri planiranju procesa procjena složenosti metode od iznimnog je značaja, jer potvrđuje jednostavnost upotrebe i mogućnost učenja. Kako određene metode modeliranja mogu imati više svojih tehnika, u tim slučajevima je složenost metode ukupna tehnička složenost tehnika. DMN je metoda sa jednom tehnikom (modelom odlučivanja), a koja se sastoji od dva pod modela (dijagrama zahtjeva odluke DRD i tablice odluke).

Odvajanjem logike procesa i odluka, DMN osigurava stabilnost i slojevitost modela procesa te se time smanjuje i složenost procesa. Budući da je logika odlučivanja odvojena od logike procesa, kombinacija DMN-a i BPMN-a zajedno nudi svoje izvanredne kvalitete, zato što opisuju potrebna poslovna znanja i organizacijske odluke. Primjena DMN i BPMN modela zajedno ne bi trebala povećati njihovu zajedničku složenost modeliranja, jer kako se proces usredotočuje na njezinu bit složenost modela se smanjuje, te samim time DMN bi, po teoriji, trebao biti jednostavan za učenje i razumijevanje. Međutim kombinacija CMMN i DMN modela osigurava ugrađeni pristup modeliranju i omogućava prikazivanje formalnih i općenitih modela procesa čime pruža uklopljen prikaz temeljne poslovne logike. Autori rada [8] ocjenjivali su složenost DMN-a, te smatraju da uporabom modela CMMN, BPMN i DMN zajedno može se stvoriti sveobuhvatan pristup u strateškom planiranju poslovnih procesa. Korištenjem navedenih modela zajedno CMMN, BPMN i DMN može se prikazati temeljna poslovna logika, jer omogućava udruživanje proceduralnih i deklarativnih procesa.

Zaključak

S obzirom da donošenje odluka može biti neusklađeno i teško održivo, pri opisivanju odluka treba se koristiti metode modeliranja procesa. Opisivanje odluka unutar procesa može biti dugačko i kompleksno s puno detalja koji mogu zasmetati, i zbog toga se jasnim i povezanim modelom može pojednostaviti poslovne procese, olakšati promjene, ubrzati razvoj te izbjeći dodatne troškove.

Pri opisu projektnih ciljeva i zahtjeva razumijevanje istih od iznimne je važnosti, i u tu svrhu današnje organizacije koriste razne tehnike. Međutim do današnjeg dana nije još stvoren formalni pristup po kojem bi projektni ciljevi i zahtjevi bili opisani u ponovljivom i razumljivom formatu. U tu svrhu bitno je definirati prvo neophodne odluke, a tek nakon toga se fokusirati na detalje (npr. karakteristična poslovna pravila). Kako bi se obuhvatilo sve potrebne elemente pri modeliranju zahtjeva i poslovnih odluka, OMG zajednica je razvila DMN model odlučivanja i notacije. Prednosti modela zahtjeva zasnovanog na DMN standardu su mnogobrojne, a ključne su te da se nudi fleksibilan i ponovljiv pristup u upravljanju i određivanju zahtjeva za donošenjem odluka. DMN ne regulira kako se određene usluge trebaju implementirati, već omogućava definiranje funkcionalnosti usluga u odnosu na model odlučivanja. U tom slučaju, s obzirom da DRD ukazuje kako skup odluka ovisi o modelima poslovnog znanja i ulaznom podacima, usluga odlučivanja trebalo bi definirati u DRD-u. Još jedan od dodatnih razloga zbog kojeg bi usluga odlučivanja trebala biti definirana u DRD-u je također i to što DRD osigurava i tehničku dokumentaciju potrebnu za održavanje znanja, a inicijalnu konfiguraciju poslovnog znanja kojeg treba održavati definira logika odluke. Zbog toga bi pri modeliranju DMN-a trebalo svako poslovno znanje kojem je, ili će biti, potrebno održavanje biti modelirano kao model poslovnog znanja u DRD-u, a kao izvore znanja modelirati odgovorno osoblje.

Literatura

1. J. Purchase, „Better AI Transparency Using Decision Modeling“, 2018. Available: http://www.luxmagi.com/2018/08/better-ai-transparency-using-decision-modelling/

2. Yawen Li, Liu Yang, Bohan Yang, Ning Wang, Tian Wu, „Application of interpretable machine learning models for the intelligent decision“, 2018.

3. Bao Wang, Yue Kong, Yongtao Zhang, Dapeng Liu, Lianju Ning, „Integration of Unsupervised and Supervised Machine Learning Algorithms for Credit Risk Assessment“, 2019.

4. Bhagat Singh Raghuwanshi, Sanyam Shukla, „Generalized class-specific kernelized extreme learning machine for multiclass imbalanced learning“, 2018.

5. OMG, „Decision Model and Notation 1.2“, OMG Headquarters, 109 Highland Avenue, Needham, MA 02494,USA, June 2018.An Introduction to Decision Modeling with DMN

6. Decison Management Solution, “An Introduction to Decision Modeling with DMN”, 2016. Available: https://www.omg.org/news/whitepapers/An_Introduction_to_Decision_Modeling_with_DMN.pdf

7. I. Dasseville, L. Janssens, G. Janssens, J. Vanthienen, and M. Denecker, “Combining DMN and the knowledge base paradigm for flexible decision enactment”, CEUR Workshop Proc., vol. 1620, 2016.Better AI Transparency Using Decision Modeling

8. F. Hasić, J. De Smedt, and J. Vanthienen, “Towards assessing the theoretical complexity of the decision model and notation (DMN) research-in-progress”, CEUR Workshop Proc., vol. 1859, no. September 2015, pp. 64–71, 2017.Enabling dynamic decision making in business processes with DMN

9. K. Batoulis, A. Baumgraß, N. Herzberg, and M. Weske, “Enabling dynamic decision making in business processes with DMN”, Lect. Notes Bus. Inf. Process., vol. 256, pp. 418–431, 2016.

Automated decision-making with DMN: from decision trees to decision tables

D. Etinger*, S.D. Simić* and L. Buljubašić*
* Juraj Dobrila University of Pula/Faculty of Informatics, Pula, Croatia
{darko.etinger, ssimic, lea.buljubasic}@unipu.hr

***Abstract* - Recent advances in artificial intelligence, especially the subfield of machine learning, is commonly cited as one of the driving forces for digital transformation and innovative business models. Ongoing research is focusing on embedding solutions based on machine learning into business processes which are commonly modelled using the BPMN standard. The Object Management Group has recently adopted the Decision Model and Notation standard. By using the Decision Model and Notation (DMN) it is possible to replace multiple decision points embedded in business processes. The purpose of this research is to provide a method to derive DMN decision tables from the corresponding machine learning model generated by the decision tree classifier. The development is conducted using the Python machine learning library scikit-learn and Camunda Modeler. This approach facilitates and automates the process of converting machine learning models into DMN tables.**

Keywords* - *Business Process Management (BPM); Decision Model and Notation (DMN); decision trees

I. INTRODUCTION

Business process management (BPM) tools have been extensively used for digital transformation within companies: recourse orchestration, work routing and automation of routine manual tasks are just some examples of business/IT alignment that enable companies to work more efficiently. As business process management (BPM) systems become more advanced, there is a shift from pre-defined process models to intelligent systems that support and automate processes. Modern BPM systems are general-purpose application development platforms that allow integration of other technologies through e.g. RESTful API calls. These innovative and augmented processes usually include artificial intelligence applications, powered by machine learning algorithms. Technologies that are being integrated into modern BPM systems include business rules, predictive analytics, machine learning and robotic process automation (RPA), Internet of Things (IoT), process mining algorithms etc.

Advances in machine learning algorithms include image recognition systems, recommender systems and natural language processing automation tasks. As business processes are generally great candidates for automation, combining BPM systems and machine learning together can take automation to another level. By connecting machine learning applications to existing BPM tools and feeding process data to machine learning applications, companies could decrease the user tasks even more, while at the same time also deliver a better product to market.

The content of this paper is structured as follows. Section II presents selected relevant literature and establishes the theoretical framework for the development of the proposed method. Section III details the conceptual modelling process and the results of the development process. The method is presented and described in detail. Finally, section IV includes the summary of the results, guidelines for further actions, limitations of the study and concluding remarks.

II. LITERATURE REVIEW AND THEORETICAL FRAMEWORK

The DMN model is primarily designed for business people who are modeling business decisions. Its central construct is the decision table. Relevant objects in DMN tables are business knowledge models that relate to implementation and sources of knowledge that serve for documentation. DMN tables are based on the FEEL (Friendly Enough Expression Language) language, a standard DMN language. FEEL language has the following characteristics [1]: a) side-effect free; b) simple data model with numbers, dates, strings, lists, and contexts; c) simple syntax designed for a wide audience; d) three-valued logic (true, false, null) based on Structured Query Language (SQL) and Predictive Model Markup Language (PMML). The elements of the DMN decision table are [1]: Decision, Business Knowledge Model, Input Data, Knowledge Source and Decision Service.

The purpose of DMN is to provide constructs that are needed to model decisions, so that organizational decision-making can be readily depicted in diagrams, accurately defined by business analysts, and (optionally) automated [2]. When using a DMN table, it is desirable to understand clearly the business objective and specific business rules to avoid storing large amounts of data. The DMN decision table contains columns and rows that indicate the input/output logic and rules.

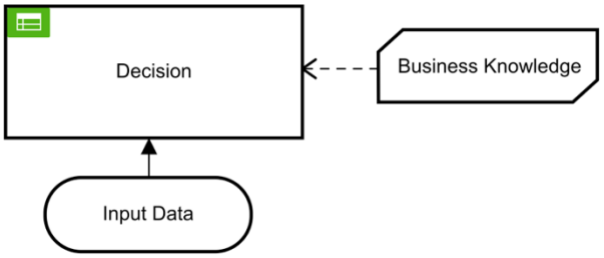

Figure 1. Basic elements of a decision model [2]

The basic elements and the structure of a decision model are shown in Figure 1. In this case, the decision is obtained using the logic of the business knowledge model that is derived from the input values.

Due to the increasing use of DMN decision tables in the business world, there is a growing need for analysis of these tables [2]. Rules for defining task analysis over this tables and a general approach for analysis and processing of decision tables based on geometric interpretation is proposed [3]. The authors provide a solution to simplify the DMN tables by merging the rules when they have the same expression and conclude that the variant with a single permutation is sufficient for practical application. The complexity of the Decision Model and Notation was assessed, and the results were compared with previous studies for other modelling notations, such as Unified Modeling Language (UML) activity diagrams, Business Process Model & Notation (BPMN), and Case Management Model & Notation (CMMN) [4]. The authors note that the complexity of the process models decreases as the process is reduced to its essence, i.e. without hard-coding the decision logic within the process. It is also suggested that employing DMN in organizational decision making might be specifically useful in a setting where business rules change frequently and where decisions have high risk for the operations [5].

The decision tree is a recursive graphical algorithm that uses the branching method to illustrate the possible out-come of a decision. It works on a Boolean algorithm principle divide and conquer. It divides data into smaller subcategories and if those unambiguous algorithms cease to work and we get the result. However, if subcategories are not uniquely determined, then the algorithm is the next subcategory of the other attributes as long as it does not get a sub-category that is uniquely determined. Using the decision tree, we get new data that we can use to build a DMN table.

Applications based on machine learning algorithms and other intelligent systems that integrate DMN decision tables with BPM systems can be developed in various ways. One approach is to integrate machine learning applications with microservice architecture, then use it as an external service in BPM systems through signal events (with BPMN represented as the triangles in enclosed circles). A dynamic decision-making framework for BPM is proposed as an extension for BPMS [6]. It aims at guiding the actions and decisions of the BPMS, thus, automatically proposing decisions that have a behavioral character. With this approach a Bayesian network is derived automatically from an event log and a decision model in DMN is based on historic and current process execution data. A user-friendly graphical user interface that allows for interactive decision enactment on the modelled decisions is proposed [7]. The authors proposed a system for translation of DMN to an extension of first-order logic.

The DMN decision modeling process structure shown in Figure 2 includes two activities addressed by the method proposed in this paper: formulate decision logic (usually a part of the business domain) and create decision services (usually part of the IT domain). The design-time approach taken in our case provides a clear advantage over a runtime approach when using decision trees, rules or linear models that can be produced transparently, represented statically by a decision service and made available to all stakeholders, both in the business and IT domain. Recent studies [8] show that such approach not only provides an explanation for any outcome, but also an understanding of how the decision is made in general, irrespective of any specific input data.

While this approach is not suitable for a broad range of applications, it is very convenient in the way that it takes into account existing DMN standards already included in many BPM systems (e.g. Camunda BPM, Goldman Sacks jDMN, Oracle Process Cloud, Red Hat Drools). This facilitates its implementation and further development and customization to specific needs and scenarios.

We argue that design-time approach may not be suitable in settings where Neural Networks, Support Vector Machines (SVM) or Bayesian networks are used. In such cases, a runtime approach would be preferred.

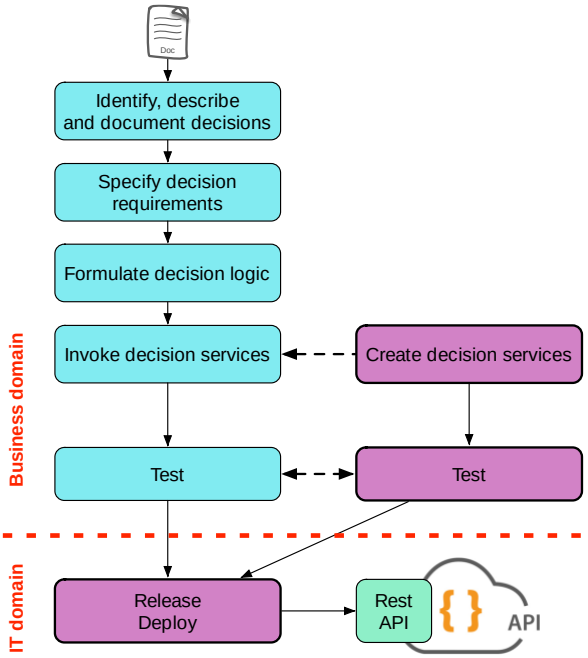

Figure 2. DMN decision modeling process

III. Research Method, Development Process and Results Analysis

The development process goal is to create a method that enables the conversion of the decision tree model generated by Python-based machine learning algorithm into a DMN based decision table XML artifact that can be imported into various BPM platforms and used within BPMN process models.

The Python programming language and Camunda modeler were the tools chosen for implementing our solution. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics while Camunda modeler is a desktop application (developed in JavaScript) for modeling BPMN and DMN diagrams which are stored in XML (Extensible Markup Language) format. Scikit-learn, a well-documented and widely popular open source machine learning library, and ElementTreeXML, a simple and efficient API for parsing and creating XML data, are the main libraries used in our script. Other Python libraries used in the development process include: Numpy and Pandas (used for reading csv files), deafultdict class from module collections, random and string (used for creating simple id generator), Graphviz (used for visualization of the decision tree classifier) and Progress (from which we used class Bar to create a simple progress bar).

The script we developed is divided into two classes, the base class (xmlDmn) consisting of functions needed for generating DMN tables and derived class (clfDmn) which contains the machine learning model.

A. *Machine learning model development*

A clfDmn object is created to which two mandatory parameters are passed: a csv file (our training data) and a dmn file (file that contains the table we want to edit).

The constructor is called and starts processing the csv file by creating and splitting the data frame into needed parts and after the data is ready, the machine learning algorithm (DecisionTreeClassifier) is started. The core function in this class is generateTableFromClf shown in Figure 3. This function first extracts information from the classifier: leaf Ids, decision path, thresholds, features, left and right children.

In the next step, a list of features (featuresForDiagram) is prepared, that will be used for creating the columns in the decision table and for the dictionary (inputOutput), because we have to ensure that the order of features are the same. Next follows the cleaning of the table (clearDecisionTable()), generating new table columns (generateTableColumns()) and the looping over leafs is started. From every leaf the class name and all samples in it (samplesInNode) is collected, but only the first one (sampleId) is used. For that sample, its decision path is tracked and looped over it. By doing that, all features, thresholds and threshold signs (<= or >) from all nodes are extracted, after which the filled dictionary is sent to generate the rules in the table (generateTableRows()). In the end of the function a new file is created, in which all changes made to existing table are written (writeTree()).

```
def generateTableFromClf(self):
    featureNames = [self.dfFeature[i] for i in self.clf.tree_.feature]
    leafIds = self.clf.apply(self.dfData)
    leftChildren = self.clf.tree_.children_left
    rightChildren = self.clf.tree_.children_right
    decPath = self.clf.decision_path(self.dfData)
    threshold = self.clf.tree_.threshold

    featuresForDiagram = list(self.dfFeature)
    featuresForDiagram.append(self.dfClassColumn)

    self.clearDecisionTable()
    self.generateTableColumns(featuresForDiagram)

    bar = IncrementalBar("Processing",suffix='%(percent)d%%', max = len(set(leafIds)))
    for i in set(leafIds):
        samplesInNode = decPath.getcol(i).copy()
        rows,cols = samplesInNode.nonzero()[0]
        sampleId = rows[0]
        nodeIndex = decPath.indices[decPath.indptr[sampleId]:decPath.indptr[sampleId+1]]
        className = self.clf.classes_[np.argmax(self.clf.tree_.value[i])]
        inputOutput = defaultdict(dict)
        for value in featuresForDiagram[:-1]:
                inputOutput[className][value] = {}
        for index, nodeId in enumerate(nodeIndex):
            nodeFeature = featureNames[nodeIndex[index-1]]
            nodeThreshold = threshold[nodeIndex[index-1]]
            if nodeId in set(leftChildren):
                inputOutput[className][nodeFeature]["<="] = nodeThreshold
            if nodeId in set(rightChildren):
                inputOutput[className][nodeFeature][">"] = nodeThreshold
        self.generateTableRows(inputOutput.items())
        bar.next()
    bar.finish()
    self.writeTree()
```

Figure 3. generateTableFromClf function from clfDmn class

```
def generateTableColumns(self,names):
    outputName = names[-1]
    inputNames = names[:-1]
    for name in inputNames:
        newInput = et.SubElement(self.decisionTableElement, "{%s}input"%(self.namespace), attrib= {"id": self.idGen("input_")})
        newInputExpression = et.SubElement(newInput, "{%s}inputExpression"%(self.namespace), attrib={"id": self.idGen("inputExpression_"),"typeRef":"double"})
        newText = et.SubElement(newInputExpression, "{%s}text"%(self.namespace))
        newText.text = name
    et.SubElement(self.decisionTableElement, "{%s}output"%(self.namespace), attrib={"id": self.idGen("output_"),"name": outputName,"typeRef":"string"})

def generateTableRows(self, mlDict):
    for k, v in mlDict:
        newRule = et.SubElement(self.decisionTableElement, "{%s}rule"%(self.namespace), attrib={"id":self.idGen("DecisionRule_")})
        for keyValues in v:
            tSignList = list(v[keyValues].keys())
            if len(v[keyValues]) == 0:
                self.createRuleCell(newRule)
            if len(v[keyValues]) == 1:
                self.createRuleCell(newRule).text = "{} {:.4f}".format(tSignList[0],v[keyValues][tSignList[0]])
            if len(v[keyValues]) == 2:
                if tSignList[0] == "<=":
                    if v[keyValues][tSignList[0]] > v[keyValues][tSignList[1]]:
                        self.createRuleCell(newRule).text = "]{:.4f}..{:.4f}]".format(v[keyValues][tSignList[1]],v[keyValues][tSignList[0]])
                    else:
                        self.createRuleCell(newRule)
                if tSignList[0] == ">":
                    if v[keyValues][tSignList[0]] < v[keyValues][tSignList[1]]:
                        self.createRuleCell(newRule).text = "]{:.4f}..{:.4f}]".format(v[keyValues][tSignList[0]],v[keyValues][tSignList[1]])
                    else:
                        self.createRuleCell(newRule)
        newOutEntry = et.SubElement(newRule, "{%s}outputEntry"%(self.namespace), attrib={"id":self.idGen("LiteralExpression_")})
        newOutText = et.SubElement(newOutEntry, "{%s}text"%(self.namespace))
        newOutText.text = str(k)
```

Figure 4. generateTableColumns and generateTableRows functions from xmlDmn class

B. *DMN tables implementation*

When clfDmn object is created xmlDmn constructor gets triggered and starts parsing the mandatory parameter (dmn file) using ElementTreeXML library. Two main functions of class xmlDmn are shown in Figure 4. Both functions are called from generateTableFromClf (see Figure 3). GenerateTableColumns loops over inputNames (sliced list of featuresForDiagram containing only input features) creating a new xml element for each element in the list and at the end the output element is created. At the moment, types from input and output elements are hardcoded. GenerateTableRows loops over dictionary which has specific structure {Class Name : { Feature : {Threshold Sign : Threshold }}}. For each Class Name (variable k in Figure 4) the function creates a new rule after which it loops over all features (variable keyValues in Figure 4) and collects threshold signs (<= or >) in list (tSignList). Depending on the number of signs in the list we decide to create an empty cell (number of signs = 0), comparison cell (number of signs = 1) or range cell (number of signs = 1). We complete our new business rule with the output element at the end, which contains the class name.

C. *Discussion*

As we can observe from Figure 5 that every sample in a leaf (node without children) took the same specific path from the root (main node) e.g. node 4 (preorder tree traversal, root node is 0) has 43 samples of class Iris-versicolor. We can conclude that every single sample in that node has "Petal width > 0.8, Petal length <= 4.75 and Petal width <= 1.65". This knowledge can be extracted from the decision tree and it can be used to create a DMN decision table as we have shown with our script.

Every rule (row) in Figure 5 represents one leaf of the decision tree. By observing Figure 4 we can notice that some cells are left empty which meaning that feature is irrelevant for classification. Furthermore we can notice ranges like "]0.8000..1.6500]" which means that feature "Petal width" needs to be in range 0.8 to 1.65, excluding 0.8 and including 1.65.

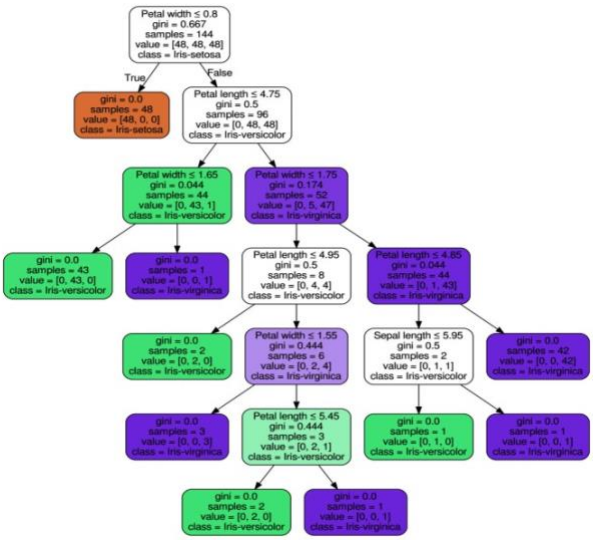

Figure 5. Decision tree visualized with Graphviz module

IV. CONCLUSION

As BPM systems collect and store detailed process information, paired with machine learning, process data can improve decision making by identifying patterns as the process advances through the workflow, thus automating user tasks that would take significant time (e.g. sales forecasting, demand planning).

The benefits of applying DMN to model business decisions can be summarized as a list that includes: (1) improved speed and faster project cycles, (2) increased participation of business stakeholders, (3) visualization through graphical logic that can reveal hidden relationships, (4) flexibility, transparency and auditability and (5) ease of deployment through REST calls. Benefits also include improved efficiency, both in terms of worker hours and end-to-end cycle time for a transaction, improved quality and compliance due to standardized processes and decisions. All of these contribute to improved customer satisfaction, increased job satisfaction by automating unskilled and semi-skilled tasks that are non-value-added overhead in most workers' workload.

First Decision

`Decision_13nychf`

U	Input +			Output +	
	Petal lenght	Sepal lenght	Petal width	Class	
	double	double	double	string	Annotation
1	-	-	<= 0.8000	"Iris-setosa"	-
2	<= 4.7500	-	]0.8000..1.6500]	"Iris-versicolor"	-
3	<= 4.7500	-	> 1.6500	"Iris-virginica"	-
4	]4.7500..4.9500]	-	]0.8000..1.7500]	"Iris-versicolor"	-
5	> 4.9500	-	]0.8000..1.7500]	"Iris-virginica"	-
6	]4.9500..5.4500]	-	]1.5500..1.7500]	"Iris-versicolor"	-
7	> 5.4500	-	]1.5500..1.7500]	"Iris-virginica"	-
8	]4.7500..4.8500]	<= 5.9500	> 1.7500	"Iris-versicolor"	-
9	]4.7500..4.8500]	> 5.9500	> 1.7500	"Iris-virginica"	-
10	> 4.8500	-	> 1.7500	"Iris-virginica"	-
+	-	-	-	-	-

Figure 6. Decision table created from extracted decision path of every leaf

In this paper we presented a method that creates DMN decision tables from a decision tree model. The proposed method automates the process of converting decision tree models (generated by the decision tree classifier) to DMN decision tables, which can be used alongside BPMN process models within various BPM systems.

The main limitation of the presented method is that relevant tasks have many attributes which can be successfully handled by machine learning algorithms but their presentation in the decision table may be a bottleneck. Future work will focus on extending the method to other machine learning algorithms.

REFERENCES

[1] OMG, „Decision Model and Notation 1.2“, OMG Headquarters, 109 Highland Avenue, Needham, MA 02494,USA, June 2018.

[2] K. Figl, J. Mendling, G. Tokdemir, and J. Vanthienen, “What we know and what we do not know about DMN”, *Int. J. Concept. Model.*, vol. 13, no. 2, p. 16, 2018.

[3] Decison Management Solution, “An Introduction to Decision Modeling with DMN”, 2016. Available: https://www.omg.org/news/whitepapers/An_Introduction_to_Decision_Modeling_with_DMN.pdf

[4] F. Hasić, J. De Smedt, and J. Vanthienen, “Towards assessing the theoretical complexity of the decision model and notation (DMN) research-in-progress”, *CEUR Workshop Proc.*, vol. 1859, no. September 2015, pp. 64–71, 2017.

[5] Diego Calvanesea, Marlon Dumasb, U¨ lari Laursonb, Fabrizio M. Maggib, Marco Montalia i Irene Teinemaab, „Semantics, Analysis and Simplification of DMN Decision Tables“, 2018.

[6] K. Batoulis, A. Baumgraß, N. Herzberg, and M. Weske, “Enabling dynamic decision making in business processes with DMN”, *Lect. Notes Bus. Inf. Process.*, vol. 256, pp. 418–431, 2016.

[7] I. Dasseville, L. Janssens, G. Janssens, J. Vanthienen, and M. Denecker, “Combining DMN and the knowledge base paradigm for flexible decision enactment”, *CEUR Workshop Proc.*, vol. 1620, 2016.

[8] J. Purchase, „Better AI Transparency Using Decision Modeling“, 2018. Available: http://www.luxmagi.com/2018/08/better-ai-transparency-using-decision-modelling/