EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

UNIVERSITY OF ECONOMICS IN BRATISLAVA
FACULTY OF ECONOMIC INFORMATICS

ZBORNÍK

z XII. medzinárodnej vedeckej konferencie

„MLADÁ VEDA AIESA 2023“
„Participácia doktorandov a mladých vedeckých pracovníkov na budovaní spoločnosti založenej na vedomostiach“

organizovanej pod záštitou
dekana Fakulty hospodárskej informatiky
prof. Mgr. Erika Šoltésa, PhD.

10. november 2023
Bratislava

Zostavenie zborníka: doc. Ing. Michaela Chocholatá, PhD.
Rozsah: 3,26 AH
Počet strán: 67

Fakulta hospodárskej informatiky
Ekonomická univerzita v Bratislave
Dolnozemská cesta 1, 852 35 Bratislava

Vydavateľstvo EKONÓM, Bratislava 2023
ISBN: 978-80-225-5102-1

O B S A H

Nina Barčáková 4
Hodnotenie a odporúčanie výrobkov pomocou flexibilných funkcií zhody a agregačných funkcií

Andrej Bednařík 14
Klasifikačné rozhodovacie stromy a ich algoritmy

Daniel Dudek 31
Modelling of the Public Procurement State Process

Martina Horváthová 44
Využitie dynamického programovania pri zostavovaní produktu cestovného poistenia

Peter Knížat 52
Metóda najmenších štvorcov pre slabé inštrumentálne premenné

Richard Mišek 59
Modelovanie investičných portfólií jazykom Python – Monte Carlo prístup

Hodnotenie a odporúčanie výrobkov pomocou flexibilných funkcií zhody a agregačných funkcií
Evaluation and recommending products by flexibile conformance functions and aggregation functions

Nina Barčáková[1]

Abstrakt
Fuzzy odporúčacie systémy sa neustále rozvíjajú a zdokonaľujú, aby poskytli rôzne odporúčania na základe požiadaviek zákazníkov. Pre používateľov je najvhodnejšie, aby mohli požiadavky na výber preferovaných entít, na ich triedenie a hodnotenie vysvetliť výrazmi prirodzeného jazyka. Pri odporúčacích systémoch treba skúmať funkcie zhody a agregačné funkcie. Tieto úlohy vyžadujú výpočet intenzity zhôd požiadaviek s hodnotami atribútov a flexibilnú logickú agregáciu atomických požiadaviek. Fuzzy zhoda vo fuzzy množinách sa používa na určenie úrovne zhody medzi záznamami a želaním zákazníka. Agregačné funkcie sa používajú na spojenie hodnôt atomických požiadaviek, ktoré sú reprezentované, ako fuzzy množiny, v našom prípade agregovanie atomických zhôd. Týmto spôsobom sme získali zložitejšie informácie a výsledky, ktoré je síce náročnejšie vypočítať, ale nenúti zákazníka presne stanoviť požiadavky.

Kľúčové slová
Fuzzy logiky, fuzzy množiny, odporúčací systém, agregačné funkcie, funkcia zhody

Abstract
Fuzzy recommendation systems are constantly evolving and improving to provide various recommendations based on customer requirements. For users, it is the most convenient to explain their preferences for selecting the most suitable entities, sorting them, and evaluating using natural language expressions. In recommendation systems, it is necessary to examine conformance functions and aggregation functions. These tasks involve calculating the intensity of comformance of requirements with attributes values and a flexible aggregation of atomic requirements. Fuzzy conformance in fuzzy sets is used to determine the level of conformity between records and customer´s preferences. Aggregation functions are used to aggregate atomic requirements´ values, which are represented as fuzzy sets. In our case, this involves aggregating atomic conformances. In this way, we obtain a more complex information and results that may be a more computationally demanding, but on the other hand, do not require the customers to precisely specify their requirements.

Key words
Fuzzy logic, fuzzy sets, recommendation system, aggregation functions, conformance function

JEL classification
C69

[1] Ing. Nina Barčáková, Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1/b, 852 35 Bratislava, nina.barcakova@euba.sk.

1 Úvod

V minulosti sa predaj tovarov a služieb organizoval, len vďaka osobnému kontaktu, inak povedané face – to – face. Dnes, keď sa posunula doba sa predaj začal realizovať prostredníctvom internetu. Vďaka internetu sa ponúka vysoké množstvo tovarov i služieb.

Pod týmto si nepredstavujeme iba častejšie sa kupujúce výrobky napr. knihy alebo filmy. Nájdeme tam aj ponuky na menej často sa kupujúce výrobky alebo služby napr. nehnuteľnosti alebo dovolenky.

Pri takomto veľkom množstve tovarov a služieb nastáva u ľudí problém v nerozhodnosti, v prebytku (vysoký počet atribútov) a nedostatku (práve tie atribúty, ktoré potrebujeme nie sú dostupné) informácií. Ľudia vo všeobecnosti pri výbere určitého produktu zvažujú plusy a mínusy jednotlivých vlastností, analyzujú ich a následne vybrané podnety agregujú. Po agregácií svojich požiadaviek dostávajú pomyselný zoznam svojich produktov, od toho najviac preferovaného, po ten najmenej výhodný v rámci ich stanovených požiadaviek. Pri takomto spôsobe zadávania požiadaviek sa človek vyjadruje v prirodzenom jazyku, čo môže byť niekedy nejasné, napr. *nízka cena*, *stredná vzdialenosť* alebo teplota vzduchu *okolo* 25°C a pod.

Tým, že požiadavky máme vyjadrené výrazmi prirodzeného jazyka, a tým aj čiastočné splnenie atomických požiadaviek, výsledok nie je binárny: spĺňa alebo nespĺňa, ale zistíme intenzitu s akou sa prikláňame k jednému z týchto výsledkov.

Bližšie sa na to pozrieme v tomto článku, kde si najprv ukážeme, ako treba s takými požiadavkami zaobchádzať. Následne ukážeme názorný príklad, ako to celé funguje.

V prípade, že nemáme históriu nákupov, alebo niečo nenakupujeme, tak veľmi často, tak nastáva situácia, že nedisponujeme dostatočným množstvom údajov, či už o zákazníkoch, alebo zvolených destináciách a podobne, aby nám bežné odporúčacie systémy poskytli riešenie [3]. Práve preto sa musíme na situáciu pozrieť inak, a vybrať si iný spôsob jej riešenia. V našom prípade sme to obrátili na možnosť zhody a podobnosti medzi atribútmi.

Cieľom článku je preskúmať a aplikovať funkcie zhody (conformance) medzi hodnotami atribútov a želanými hodnotami tak, aby sa zachytili všetky relevantné dátové typy. Následne je potrebné agregovať tieto zhody. Tieto funkcie sa upravia podľa potreby a vytvorí sa nástroj na výpočet zhôd, a ich agregáciu.

2 Základ klasická logika a fuzzy logika pre túto prácu

V tejto kapitole zhrnieme koncepty, ktoré použijeme v práci. Fuzzy logika skúma pravdivostné hodnoty, ktoré sú vyjadrené výrazmi prirodzeného jazyka, inak povedané slovne. Napr. aká je pravdivostná hodnota, že skúmaný záznam má vlastnosť nízka cena [2].

Výhoda fuzzy logiky je jej schopnosť pracovať nie len s číselnými údajmi, ale dokáže matematicky spracovať aj údaje vyjadrené slovne. Takéto slovne vyjadrenie zahŕňa v sebe reč hovorovú, medzi ktorú patria slová ako málo, trochu, dosť, približne polovica, a podobne, ktoré práve fuzzy logika vie formalizovať matematickým aparátom a následne umožňuje počítaču vyhodnotiť [5].

Fuzzy množina predstavuje ostrú množinu rozšírenú o intenzitu príslušnosti. Neostrá množina teda neobsahuje iba hodnoty true a false, resp. 0 a 1, ale prvky v tejto množine sú odstupňované a nadobúdajú hodnotu z reálneho intervalu [0, 1], ktorú možno identifikovať ako intenzitu príslušnosti [4].

$$\mu_A(x): X \rightarrow [0,1] \tag{1}$$

Intenzita príslušnosti priraďuje množinám príslušnosť z intervalu [0, 1]. Toto má za následok, že vieme vďaka fuzzy množinám a logike ľahko vyjadriť aj neurčité informácie, ako

sú „dosť“, „málo“, „približne“, „veľa“ a podobne, a preto sa pohybujeme v matematickom modeli neurčitosti a nepresnosti [1].

3 Zhoda a podobnosť

V tejto kapitole preskúmame funkcie zhody a podobností, ktoré sú vhodné na výber najvhodnejších výrobkov alebo služieb v prostredí sémantickej neurčitostí a tiež keď nie sú k dispozícii objemné dáta o histórii nákupov [6].

Zoberme si tabuľku, v ktorej sa nachádzajú údaje. Záznamy, sa nachádzajú v riadkoch a ich atribúty sa nachádzajú v stĺpcoch. Pre vytvorenie odporúčacieho systému musíme nájsť zhodu medzi hodnotami atribútov a preferenciami zákazníkov.

Zhoda atribútov pre kategorické údaje sa určuje nasledovne [3]:

$$C\left(A\left[z_i, z_j\right]\right) = \min\left\{\min_{x\in d_i}\left\{\max_{y\in d_j}\{s\ (x,y)\}\right\}, \min_{x\in d_j}\left\{\max_{y\in d_i}\{s(x,y)\}\right\}\right\} \quad (2)$$

- d_i – zoznam hodnôt atribútu A pre z_i , kde z_j je záznam v databáze alebo jej štruktúry
- d_j – zoznam hodnôt atribútu A pre z_i , kde z_j je vektor požiadaviek kupujúceho
- s (x, y) – je relácia podobnosti pre hodnoty x a y

Zhoda atribútov pre numerické údaje sa určuje nasledovne [3]:

$$C\left(A\left[z_i, z_j\right]\right) = \min(\mu_{zi}(A), \mu_{zj}(A), s(z_i(A), z_j(A)) \quad (3)$$

- μ_{zi} (A) – stupeň príslušnosti hodnoty atribútu A do fuzzy množiny pre zi
- μ_{zj} (A) – stupeň príslušnosti hodnoty atribútu A do fuzzy množiny pre zj
- s (z_i (A), z_j (A)) – je relácia podobnosti medzi fuzzy množinami definovaná nad doménou atribútu A[1]

Relácia podobnosti nám umožňuje formalizovať podobnosť výrazov alebo fuzzy množín matematicky [1].

Relácia podobnosti je binárna operácia s: D*D → <0, 1>, ak platí pre každé x, y domény:

- reflexívna – P1 s(x, x) = 1
- symetrická – P2 s(x, y) = s(y, x)

K výpočtom treba nový parameter, ktorý zabezpečí spravodlivú penalizáciu hodnôt.

$$\Delta = \frac{|z_K - z_{atribút}|}{\alpha} \quad (4)$$

Kde z_k je preferovaná hodnota atribútu klienta, $z_{atribút}$ je prislúchajúca hodnota vybraného atribútu a α je vhodne vybraný koeficient, pre ktorý platí $\alpha > |A_{max} - A_{min}|$. Daný koeficient predstavuje intenzitu flexibility pre podobnosť [1].

Pre objasnenie rozprávame sa o situácii, kedy hodnoty, ktoré sú bližšie k požadovanej hodnote klienta dostanú nižšiu hodnotu parametra a preto dosiahnu práve vyššiu intenzitu príslušnosti. Naopak, ak pôjde o hodnoty, ktoré majú vyšší rozdiel s požadovanou hodnotou

klienta, tak ich trestanie bude naopak silnejšie a preto aj intenzita príslušnosti bude dosahovať nižšiu hodnotu. V ďalšej kapitole vysvetlíme ako tento prístup funguje.

4 Experimenty

V tejto časti najprv vysvetlíme dáta, ktoré sme vytvorili a následne ukážeme výpočet. Následne to zhrnieme v podkapitole diskusia.

4.1. Dáta

Odporúčací systém vytvoríme a následne budeme demonštrovať na produkte dovolenkové destinácie. Atribúty pre tento príklad – sú Štát, Cena, Vzdialenosť, Počet izieb, Hodnotenie. Odporúčací systém je možné neustále dopĺňať o ďalšie atribúty. V našom prípade sme sa rozhodli, že na znázornenie požadovaného výsledku nám postačia práve tieto vyššie uvedené. V tabuľke 1: Dovolenkové destinácie s prislúchajúcimi atribútmi, je zobrazený priestor, v ktorom sú naše destinácie.

Pre každý atribút je potrebné vytvoriť relácie podobnosti. Pri vytváraní relácií podobnosti pre ilustračný príklad sme dôraz kládli na bežné podobnosti, napr. pre krajiny. Pre iné dáta alebo požiadavky, hodnoty sú iné, ale vzorce a výpočet sa nemení [1].

Tab. 1: Dovolenkové destinácie s prislúchajúcimi atribútmi

ID	Štát	Cena (€)	Vzdialenosť od mora (m)	Počet izieb	Hodnotenie v bodoch
H1	Bulharsko	550	100	120	6,9
H2	Egypt	460	50	500	7,1
H3	Malta	600	60	350	5,6
H4	Španielsko	700	188	90	4,5
H5	Taliansko	580	120	80	9
H6	Keňa	950	0	140	8,6
H7	Dubaj	870	200	580	7,3
H8	Turecko	660	0	360	5,5
H9	Dominikánska republika	990	0	410	6,9
H10	Egypt	620	76	370	8,2
H11	Dubaj	740	150	450	2,8
H12	Bulharsko	485	200	95	6,4
H13	Dominikánska republika	870	190	400	9,2
H14	Bulharsko	580	85	210	8
H15	Egypt	550	0	250	4,6
H16	Malta	620	40	206	6,6
H17	Malta	593	30	350	8,2
H18	Španielsko	800	100	260	8,3
H19	Španielsko	600	200	300	9,1

H20	Taliansko	490	150	140	9,1
H21	Dominikánska republika	980	130	400	7,3
H22	Dominikánska republika	960	45	410	7,6
H23	Keňa	899	70	400	7,4
H24	Španielsko	760	100	230	8
H25	Taliansko	530	300	125	9
H26	Keňa	840	110	80	4,8
H27	Keňa	850	160	80	6,7
H28	Dubaj	792	195	270	6,9
H29	Turecko	489	210	260	8,8
H30	Španielsko	630	220	320	7,45
H31	Turecko	830	270	395	9,3

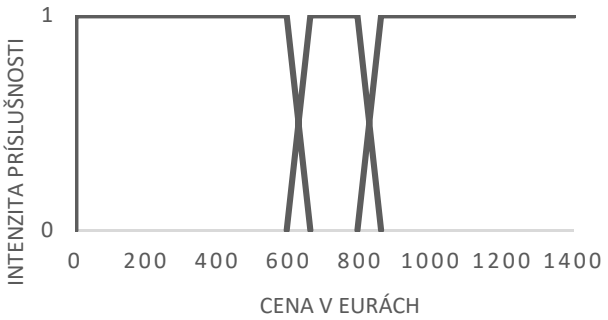
Zdroj: Vlastné spracovanie

Atribúty si popíšeme nižšie. Ich opis potrebujeme, aby sme vedeli čo znamenajú, aké informácie nesú a ako ich budeme využívať v našich výpočtoch. Obrázky 1 až 4 nám ukazujú relácie podobnosti pre všetky atribúty domény, ktoré budeme neskôr potrebovať pre výpočty. Popis atribútov:

Atribút ŠTÁT - určuje miesto dovolenkovej destinácie. V našom odporúčacom systéme sme vybrali deväť štátov, z ktorých si môže zákazník zvoliť, jemu najviac vyhovujúci.

Atribút CENA – stanovené peňažné prostriedky za dovolenkové destinácie v mene €. Zvolili, že daný atribút rozdelíme do 3 fuzzy množín nasledovne: nízka cena (NC), stredná cena (SC) a vysoká cena (VC).

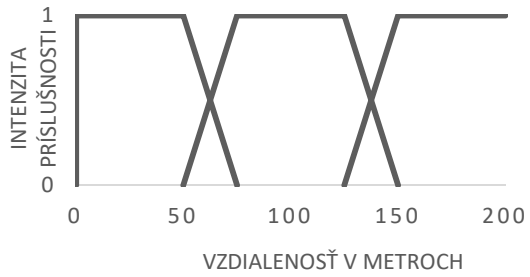
Obr.1: Lingvistická premenná Cena



Zdroj: Vlastné spracovanie

Atribút VZDIALENOSŤ – vzdialenosť od mora uvedená v metroch. Pri tomto atribúte sme opäť zvolili možnosť rozdeliť ho do 3 fuzzy množín: krátka vzdialenosť (KV), stredná vzdialenosť (SV) a dlhá vzdialenosť (DV).

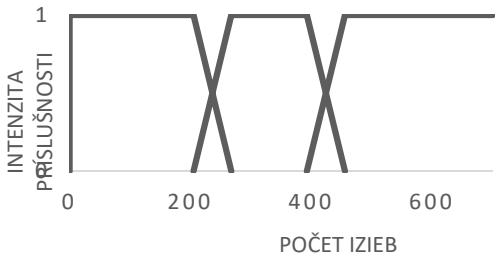
Obr. 2: Lingvistická premenná Vzdialenosť



Zdroj: Vlastné spracovanie

Atribút POČET IZIEB – definuje veľkosť hotela, či ide o malý, stredný, alebo naopak veľký hotel. Atribút je určený 3 množinami: malý počet izieb (MPI), stredný počet izieb (SPI), veľký počet izieb (VPI).

Obr. 3: Lingvistická premenná Počet izieb



Zdroj: Vlastné spracovanie

Atribút HODNOTENIE - ukazuje bodové ohodnotenie kvality danej dovolenkovej destinácie, maximálna hodnota dosahuje stupeň 10 bodov a minimálna hodnota je na bode 0. Hodnotenie je rozdelené do 3 fuzzy množín: nízke hodnotenie (NH), stredné hodnotenie (SH), vysoké hodnotenie (VH).

Obr. 4: Lingvistická premenná Hodnotenie v bodoch

Zdroj: Vlastné spracovanie

Tab. 2: Relácia podobnosti atribútu Štát

	BUL	TAL	ŠPA	MAL	TUR	EGY	DUB	DOM	KEŇA
BUL	1	0,9	0,8	0,7	0,50	0,40	0,30	0,20	0,10
TAL		1	0,85	0,75	0,60	0,40	0,35	0,25	0,10
ŠPA			1	0,60	0,55	0,50	0,40	0,30	0,20
MAL				1	0,70	0,60	0,80	0,85	0,30
TUR					1	0,80	0,75	0,70	0,40
EGY						1	0,70	0,65	0,60
DUB							1	0,90	0,65
DOM								1	0,70
KEŇA									1

Zdroj: Vlastné spracovanie

Tab. 3: Relácia podobnosti atribút Cena

SCENA	NC	SC	VC
NC	1	0,70	0,30
SC		1	0,60
VC			1

Zdroj: Vlastné spracovanie

Tab. 4: Relácia podobnosti atribút Vzdialenosť

SVZDIALENOSŤ	KV	SV	DV
KV	1	0,70	0,30
SV		1	0,60
DV			1

Zdroj: Vlastné spracovanie

Tab. 5: Relácia podobnosti atribút P. izieb

SP. IZIEB	MPI	SPI	VPI
MPI	1	0,70	0,30
SPI		1	0,60
VPI			1

Zdroj: Vlastné spracovanie

Tab. 6: Relácia podobnosti atribút Hodnotenie

SHODNOTENIE	NH	SH	VH
NH	1	0,70	0,30
SH		1	0,60
VH			1

Zdroj: Vlastné spracovanie

4.2. Výpočet

Pre ukázanie odporúčaní sme si zvolili dvoch klientov, ktorý si stanovili svoje požiadavky na dovolenkové destinácie. Klient 1 požaduje Cenu dovolenkovej destinácie na hodnote 800€, vzdialenosť od mora 85m a veľkosť hotela za pomoci počtu izieb v hoteli na úrovni 300. Klient 2 si ešte stanovil štát, dovolenku chce stráviť v Egypte a hodnotenie hotela na úrovni 7,7 bodov.

Bude potrebné vypočítať zhody pre dané atribúty a tie následne agregovať a tieto agregované skóre usporiadať. Na agregáciu sme zvolili v tomto prípade metódu aritmetického priemeru, kedy sme všetky skóre spočítali a vydelili počtom atribútov. Obrázok 6 zobrazuje výsledky zhôd atribútov a ich agregáciu. Pod obrázkom sú uvedené názorné výpočty pre prvý záznam na základe požiadaviek Klient 2.

Obr. 5: Odporúčanie dovolenkových destinácií Klient 1 a Klient 2

ID	ŠTÁT	CENA	VZDIALENOSŤ	POČET IZIEB	HODNOTENIE	VÝSLEDOK CENA	VÝSLEDOK VZDIALENOSŤ	VÝSLEDOK POČET IZIEB	VÝSLEDOK HODNOTENIE	VÝSLEDOK ŠTÁT	VÝSLEDOK KLIENT 1
H18	Španielsko	800	100	260	8,3	1	0,655	0,84			0,8317
H28	Dubaj	792	195	270	6,9	0,866	0,59	0,97			0,8087
H8	Turecko	660	0	360	5,5	0,8	0,645	0,94			0,795
H3	Malta	600	60	350	5,6	0,6	0,705	0,95			0,7517
H17	Malta	593	30	350	8,2	0,5965	0,675	0,95			0,7405
H14	Bulharsko	580	85	210	8	0,59	1	0,61			0,7333
H19	Španielsko	600	200	300	9,1	0,6	0,555	1			0,7183
H23	Keňa	899	70	400	7,4	0,5505	0,715	0,78			0,6818
H24	Španielsko	760	100	230	8	0,85	0,655	0,53			0,6783
H21	Dominikánska republika	980	130	400	7,3	0,51	0,655	0,78			0,6483
ID	**ŠTÁT**	**CENA**	**VZDIALENOSŤ**	**POČET IZIEB**	**HODNOTENIE**	**VÝSLEDOK CENA**	**VÝSLEDOK VZDIALENOSŤ**	**VÝSLEDOK POČET IZIEB**	**VÝSLEDOK HODNOTENIE**	**VÝSLEDOK ŠTÁT**	**VÝSLEDOK KLIENT 3**
H18	Španielsko	800	100	260	8,3	1	0,655	0,84	0,97	0,5	0,793
H10	Egypt	620	76	370	8,2	0,49	0,721	0,63	0,975	1	0,7632
H17	Malta	593	30	350	8,2	0,5965	0,675	0,95	0,975	0,6	0,7593
H8	Turecko	660	0	360	5,5	0,8	0,645	0,94	0,49	0,8	0,735
H28	Dubaj	792	195	270	6,9	0,866	0,59	0,97	0,48	0,7	0,7212
H14	Bulharsko	580	85	210	8	0,59	1	0,61	0,985	0,4	0,717
H19	Španielsko	600	200	300	9,1	0,6	0,555	1	0,93	0,5	0,717
H23	Keňa	899	70	400	7,4	0,5505	0,715	0,78	0,905	0,6	0,7101
H24	Španielsko	760	100	230	8	0,85	0,655	0,53	0,985	0,5	0,704
H13	Dominikánska republika	870	190	400	9,2	0,565	0,595	0,78	0,925	0,65	0,703

Zdroj: Vlastné spracovanie

$$C\left(\text{Štát}\,[z_K, z_{18}]\right)$$
$$= \min\{min\{max\{s\,(Egypt, \text{Španielsko})\}\}, min\{max\{s(Egypt, \text{Španielsko})\}\}\}$$
$$= \min\{min\{max\{0{,}5\}\}, min\{max\{0{,}5\}\}\} = \min\{0{,}5; 0{,}5\} = 0{,}5$$

$$C\,(Cena\ v\ eurách\,[z_K, z_{18}])$$
$$= \max(\mu_{z_K}(800), \mu_{z_{18}}(800), s(z_K(SC), z_{18}(SC)) - \frac{|800 - 800|}{2000}$$
$$= \max(0{,}87; 0{,}87; 1) = 1$$

$$C\,(Vzdialenosť\,[z_K, z_{18}])$$
$$= \min(\mu_{z_K}(85), \mu_{z_{18}}(100), s(z_K(KV), z_{18}(SV)) - \frac{|85 - 100|}{1000}$$
$$= \min(0{,}73; 0{,}67; 0{,}7) - 0{,}015 = 0{,}67 - 0{,}015 = 0{,}655$$

$$C\ (Počet\ izieb\ [z_K, z_{18}])$$
$$= \min(\mu_{z_K}(300), \mu_{z_{18}}(260), s(z_K(SPI), z_{18}(SPI)) - \frac{|300-260|}{1000}$$
$$= \min(1; 0{,}88; 1) - 0{,}04 = 0{,}88 - 0{,}04 = 0{,}84$$

$$C\ (Hodnotenie\ [z_K, z_{18}])$$
$$= \min(\mu_{z_K}(7{,}7), \mu_{z_{18}}(8{,}30), s(z_K(VH), z_{18}(VH)) - \frac{|7{,}7-8{,}3|}{20}$$
$$= \min(1; 1; 1) - 0{,}03 = 1 - 0{,}03 = 0{,}97$$

$$\sum_{i=1}^{5} C(A_i[z_k, z_{18}])$$
$$= \sum_{i=1}^{5} C\left(Štát\ [z_K, z_{18}]\right) + C\ (Cena\ v\ eurách\ [z_K, z_{18}])$$
$$+ C\ (Vzdialenosť\ [z_K, z_{18}]) + C\ (Počet\ izieb\ [z_K, z_{18}])$$
$$+ C\ (Hodnotenie\ [z_K, z_{18}]) = 0{,}5 + 1 + 0{,}655 + 0{,}84 + 0{,}97 = \frac{3{,}965}{5}$$
$$= 0{,}793$$

4.3. Diskusia

V článku sme sa venovali dvom situáciám ohľadne odporúčania dovolenkových destinácií na základe klientom stanovených požiadaviek v prirodzenom jazyku. Výsledky týchto požiadaviek sme museli neskôr agregovať. Pre agregáciu sme vyberali už vyššie spomínaný aritmetický priemer ako jednu z možností agregácie. Do budúcna by sme mohli voliť asymetrickú konjunkciu, disjunkciu alebo iné. Tieto agregácie neovplyvnia výpočet podobnosti a zhôd, ktoré boli hlavným cieľom práce. Prejdeme na objasnenie našich experimentov a výsledkov, ku ktorým sme sa dostali.

V prvej situácii si klient 1 uviedol svoje požiadavky na 3 atribúty a to Cena v hodnote 800€, Vzdialenosť hotela od mora 85 metrov a veľkosť hotela - počet izieb 300. Po výpočtoch sme prišli k zisteniu, že mu najviac vyhovovala dovolenková destinácia H18 s intenzitou príslušnosti 0,8317; ktorej cena je na hodnote 800 €, vzdialenosť hotela od mora 100 metrov a hotel disponuje s 260 izbami. Na druhej pozícii sa umiestnil H28 a na tretej pozícii H8.

Situácia druhá obsahovala, už 5 požiadaviek od klienta 2 na vybrané atribúty a teda klient 2 si pridal ešte požiadavku krajiny a to Egypt a hodnotenie hotela 7,7. Po pridaní požiadavky dosiahol usporiadanie dovolenkových destinácií nasledovné. Na prvom najviac vyhovujúcom mieste sa umiestnila dovolenková destinácia H18, ktorá napriek tomu, že nespĺňa požiadavku destinácie v Egypte dosiahla najvyššie hodnotenie a to 0,793. Na druhom mieste je H10 a tretie miesto patrí dovolenkovej destinácií H17.

5 Záver

Vytvorenie odporúčacieho systému založeného na využívaní fuzzy množín umožnilo efektívne hodnotenie v prípade nepresne vyjadrených informácií o kritériách alebo obmedzeniach, ktoré sú vyjadrené výrazmi prirodzeného jazyka, alebo podobné záznamy sa musia podobne hodnotiť.

Fuzzy zhoda vo fuzzy množinách sa používa na určenie úrovne zhody medzi elementmi. Táto zhoda môže byť založená na rôznych kritériách, ako napríklad min-max zhoda, vzdialenostná zhoda, alebo korelačná zhoda. Každá z týchto zhôd má svoje výhody a nevýhody v závislosti od aplikácie a konkrétnych požiadaviek.

Fuzzy agregačné funkcie sa používajú na spojenie (agregovanie) hodnôt rôznych atribútov, ktoré sú reprezentované, ako fuzzy množiny, v našom prípade agregovanie atomických zhôd. Týmto spôsobom sme získali komplexnejšie výsledky, ktoré je síce náročnejšie vypočítať, ale nenúti zákazníka presne stanoviť požiadavky.

Fuzzy odporúčacie systémy majú veľký potenciál na svoj neustály rozvoj. V pokračovaní výskumu preskúmame ďalšie možnosti napr. ako umožniť aby sa nezadávala hodnota funkcie podobnosti numerický, ale lingvistický: *veľmi podobné* a *trochu podobné* a ako ich formalizovať do trojuholníkových fuzzy čísiel a spracovať.

Príspevok bol spracovaný v rámci riešenia grantovej úlohy – COST Action CA19130, FinAI – Fintech and Artificial Intelligence in Finance.

Literatúra

1. Barčáková, N. (2023). Hodnotenie a odporúčanie výrobkov pomocou flexibilných funkcií zhody a agregačných funkcií. Diplomová práca. Fakulta hospodárskej informatiky, Ekonomická univerzita v Bratislave.
2. Fribourg, A. M. U. of, Meier, A., Fribourg, U. of, Fribourg, N. W. U. of, Werro, N., AG, M. A. S. F., Albrecht, M., AG, S. F., AG, M. S. S. F., Sarakinos, M., Norwegian University of Science & Technology, & Metrics, O. M. A. (2005, August 1). Using a fuzzy classification query language for Customer Relationship Management: Proceedings of the 31st International Conference on Very Large Data Bases. DL Hosted proceedings. https://dl.acm.org/doi/10.5555/1083592.1083717
3. Hudec, M. (2016). Fuzzy data in relational databases. Fuzziness in Information Systems, 139-176. https://doi.org/10.1007/978-3-319-42518-4_5
4. Navara, M., & Olšák, P. (2007). Základy Fuzzy množin. Nakladatelství ČVUT. https://cmp.felk.cvut.cz/~navara/Zaklady_fuzzy_mnozin/errata1ed.pdf
5. Novák, V. (1986, January 1). *fuzzy množiny a jejich aplikace - Vilém Novák (1986, SNTL - Nakladatelství technické literatury)*. Antikvariát Avion. https://www.antikavion.cz/kniha/fuzzy-mnoziny-a-jejich-aplikace-vilem-novak-1986
6. Vučetić, M., & Hudec, M. (2018). A fuzzy query engine for suggesting the products based on conformance and asymmetric conjunction. Expert Systems with Applications, 101, 143–158. https://doi.org/10.1016/j.eswa.2018.01.049

Klasifikačné rozhodovacie stromy a ich algoritmy
Classification decision trees and their algorithms

Andrej Bednařík[1]

Abstrakt
Rozhodovacie stromy sú základným nástrojom strojového učenia, ktorý sa často používa na klasifikačné a regresné úlohy. V súčasnej dobe masívneho množstva dát a vyspelých metód strojového učenia sa rozhodovacie stromy stali jedným z hlavných nástrojov analýzy dát a predikčného modelovania, takéto modely sa veľmi podobajú ľudskému uvažovaniu a sú ľahko pochopiteľné. Tento článok sa zaoberá mechanikou rozhodovacích stromov, s hlavným zameraním na klasifikačné kritériá a rozdelenie dát. Príspevok predstavuje známe algoritmy, ako sú ID3, C4.5 a CART, ktoré sú základom pre tvorbu klasifikačných rozhodovacích stromov. Záver je venovaný demonštrácii tvorby a vizualizácii rozhodovacích modelov prostredníctvom jazyka Python.

Kľúčové slová
Rozhodovacie stromy, ID3, C4.5, CART, Gini index, Entropia

Abstract
Decision trees are a fundamental tool of machine learning, often used for classification and regression tasks. In the current era of massive data sets and advanced machine learning methods, decision trees have become one of the primary tools for data analysis and predictive modeling. Such models closely resemble human reasoning and are easily understandable. This article delves into the mechanics of decision trees, with a main focus on classification criteria and data splitting. Topic introduces well-known algorithms such as ID3, C4.5, and CART, which form the basis for creation of classification decision trees. In the end, we demonstrate how to create and visualize a decision model using the Python language.

Key words
Decision trees, ID3, C4.5, CART, Gini index, Entropy

JEL classification
C61, C89

1 Úvod

Rozhodovacie stromy predstavujú jednu z fascinujúcich a najpraktickejších metód v oblasti dátovej analýzy. V ére, keď sa dáta stávajú novým "zlatom", je schopnosť efektívne a presne interpretovať tieto dáta najvyššou prioritou. A práve rozhodovacie stromy, s ich jedinečným spôsobom vizualizácie a analýzy, sa stávajú jedným z pilierov modernej dátovej vedy. V čase, keď technológia preniká do každého kúta nášho života, sa potreba rýchlych a spoľahlivých rozhodovacích nástrojov stáva kľúčovou. Rozhodovacie stromy sú považované za jednu z najefektívnejších metód strojového učenia, ktorá umožňuje efektívnu klasifikáciu a regresiu dát (Zhang et al., 2022). Táto metóda strojového učenia, ktorá kombinuje matematickú rigoróznosť s vizuálnou jednoduchosťou, umožňuje analytikom identifikovať kľúčové vzory a vzťahy v dátach.

[1] Ing. Andrej Bednařík, Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra matematiky a aktuárstva, Dolnozemská cesta 1, 852 35 Bratislava, andrej.bednarik@euba.sk.

V súčasných dátových analýzach sa rozhodovacie stromy využívajú na prediktívnu analýzu, klasifikáciu alebo výber čŕt. Pomocou prediktívnej analýzy predikujeme budúce udalosti alebo trendy na základe historických dát, v oblasti e-commerce môžu napríklad predpovedať, ktorý produkt si zákazník kúpi na základe jeho predošlého správania. V prípade klasifikácií sú rozhodovacie stromy široko využívané. Bežne aplikácie klasifikácie sú napríklad rozpoznávanie obrazov, detekcia podvodov alebo diagnostika chorôb. Výber čŕt nám umožňuje v dátových súboroch s veľkým počtom atribútov vybrať tie atribúty, ktoré sú skutočné dôležité. Rozhodovacie stromy môžu byť použité na identifikáciu a výber najdôležitejších čŕt alebo atribútov v dátovom súbore, čo môže byť užitočné pri redukcii dimenzií, alebo zlepšení efektívnosti modelovania (Chen & Guestrin, 2016).

Rozhodovacie stromy nachádzajú uplatnenie v rôznych oblastiach, od financií až po medicínu. V medicíne sú rozhodovacie stromy kľúčovým nástrojom pri identifikácii rizikových faktorov a predpovedaní výsledkov pacientov (Smith & Roberts, 2022). Vďaka ich schopnosti rýchlo a presne identifikovať kľúčové faktory, môžu lekári a výskumníci lepšie predpovedať výsledky liečby a zlepšiť kvalitu života pacientov. V 70. rokoch sa rozhodovacie stromy stali populárnejšími vďaka vývoju algoritmov ako ID3 od Rossa Quinlana. Tento algoritmus využíval koncept entropie na optimalizáciu klasifikačných rozhodovacích stromov. V 80. rokoch bol tento algoritmus vylepšený a viedol k vytvoreniu algoritmu C4.5, ktorý je aj dnes považovaný za jeden z kľúčových algoritmov v oblasti rozhodovacích stromov.

2 Základné princípy rozhodovacích stromov, algoritmy a metódy

Rozhodovací strom je hierarchický model, ktorý sa skladá z uzlov, vetiev a listov. Každý uzol reprezentuje test na jednom z atribútov, každá vetva reprezentuje výsledok testu a listy reprezentujú rozhodnutie alebo výsledok. Táto štruktúra umožňuje vizualizovať proces rozhodovania a je intuitívne zrozumiteľná. Vďaka svojej grafovej štruktúre je možné stromy ľahko vizualizovať a interpretovať. Sú schopné zachytiť nelineárne vzťahy medzi premennými bez potreby transformácie dát.

Rozhodovacie stromy predstavujú atraktívny nástroj v oblasti strojového učenia vďaka ich množstvu výhod. Jednou z najvýznamnejších výhod je ich interpretovateľnosť a schopnosť vizualizácie. Táto jedinečná vlastnosť umožňuje aj laikom jasne pochopiť a vizualizovať rozhodovacie procesy od koreňa stromu až po jeho listy. Toto jasné zobrazenie rozhodovacích procesov umožňuje rýchle pochopenie, ako boli rozhodnutia prijaté (Kelleher et al., 2015). Ďalšou významnou výhodou je ich flexibilita, keďže sú schopné spracovávať rôzne typy dát - od numerických po kategorické a dokonca aj dáta s chýbajúcimi hodnotami, a to všetko bez potreby predbežnej transformácie či normalizácie. Automatická selekcia atribútov je ďalším pozoruhodným aspektom, ktorý zefektívňuje proces modelovania. V procese vytvárania stromu sú dôležité atribúty vyberané automaticky, zatiaľ čo tie menej dôležité sú ignorované, čo eliminuje časovo náročné a chybové ručné vyberanie atribútov (Dietterich, 2000). Napokon, rozhodovacie stromy majú aj nízke nároky na predspracovanie dát, na rozdiel od mnohých iných metód (Kelleher et al., 2015).

Rozhodovacie stromy, hoci ponúkajú množstvo výhod, nezostávajú imúnne voči problémom a výzvam. Prvým kameňom úrazu je ich náchylnosť k pretrénovaniu, keďže majú sklon k tvorbe komplexných modelov, ktoré sa príliš prispôsobia trénovacím dátam a tým pádom riskujú nízky výkon na testovacích alebo nových dátach (Kelleher et al., 2015). Avšak, existuje riešenie v podobe orezávania stromu, ktoré spočíva v odstránení vetiev, ktoré nepridávajú signifikantnú prediktívnu hodnotu a tým zjednodušujú model a zlepšujú jeho generalizáciu. Nestabilita je ďalším problémom, pretože rozhodovacie stromy sú citlivé na malé

zmeny v trénovacích dátach, a zmena len niekoľkých príkladov môže viesť k radikálnej zmene štruktúry stromu (Dietterich, 2000). Na zmiernenie tejto variability môžeme využiť ensemble metódy, napríklad náhodné lesy, ktoré kombinujú výsledky mnohých stromov, trénovaných na rôznych podmnožinách dát. Lokálna optimalizácia je taktiež problémom, pretože pri vytváraní stromu sú vetvenia založené na lokálnych a nie globálnych, optimálnych riešeniach (Dietterich, 2000). Moderné varianty algoritmov sa však snažia tento problém riešiť integráciou globálnych aspektov do procesu vytvárania stromu. Napokon, aj keď rozhodovacie stromy sú pomerne flexibilné, môžu naraziť na prekážky pri modelovaní niektorých typov nelineárnych vzťahov alebo vzťahov, ktoré sú ťažko oddeliteľné v priestore atribútov (Kelleher et al., 2015). V takýchto prípadoch by mohlo byť efektívnejšie kombinovať rozhodovacie stromy s inými modelmi alebo sa obrátiť na alternatívne prístupy, ako sú neurónové siete.

ID3 (Iterative Dichotomiser 3): ID3 je jedným z prvých algoritmov na konštrukciu rozhodovacích stromov. Využíva entropiu a Information Gain(IG) na určenie najlepšieho rozdelenia v každom uzle a je optimalizovaný pre kategorické atribúty (Quinlan, 1986).ID3 je vhodný pre použitie na menšie dátové sady, kde je potrebné rýchlo vytvoriť model. Vzťah(1) je použitý v prípade výpočtu binárnej klasifikácie a vzťah (2) v prípade multiklasifikácie. V algoritme ID3 najvhodneší atribút pre rozdelenie je ten, ktorý má najvyššiu hodnotu IG. Algoritmus postupuje nasledovne:

1. Inicializácia: Načíta celý dataset ako koreňový uzol rozhodovacieho stromu.
2. Výpočet entropie pre celý dataset.
3. Výpočet entropie pre jednotlivé Atribúty datasetu.
4. Výpočet IG pre jednotlivé atribúty.
5. Výber atribútu: Vyberie sa atribút, ktorý najlepšie rozdeľuje dáta na základe najvyššej hodnoty IG.
6. Rekurzia: Pre každú hodnotu vybraného atribútu vytvorí vetvu zakončenú uzlom alebo listom; v nových uzloch sa opakujú kroky 2 až 5 pre každú novú podmnožinu dát v novom uzle.
7. Ukončenie: Algoritmus končí keď sú všetky atribúty použité na rozhodovanie alebo keď všetky príklady v podmnožine patria do rovnakej triedy.

$$Entropy(D) = -p_+ \, log_2(p_+) - p_- log_2(p_-) \tag{1}$$

D je dátová množina, p_+ je pravdepodobnosť pozitívnej triedy a p_- pravdepodobnosť negatívnej triedy v množine *D*.

$$Entropy(D) = -\sum_{i=1}^{n} p_i \, log_2(p_i) \tag{2}$$

D je datová množina, *n* je počet tried a p_i je pravdepodobnosť *i*-tej triedy v množine *D*.

$$IG(D, A) = Entropy(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} \, Entropy(D_v) \tag{3}$$

D je dátová množina, *A* je atribút, ktorý sa hodnotí, D_v je podmnožina *D*, ktorá obsahuje všetky príklady, kde atribút *A* ma hodnotu *v* (Mishra, 2020).

C4.5: Algoritmus C4.5, tiež vyvinutý Rossom Quinlanom ako rozšírenie predchádzajúceho algoritmu ID3, bol predstavený v roku 1993. Môže spracovávať kategorické aj spojité atribúty. Na rozdiel od ID3, C4.5 využíva pomer zisku namiesto čistého zisku informácií a obsahuje mechanizmy na spracovanie chýbajúcich údajov (Quinlan, 1993). Tento algoritmus používame pri väčších dátových sadách a keď je potrebné vylepšiť výkon algoritmu ID3. Má schopnosť pracovať s chýbajúcimi hodnotami a môže orezávať stromy po vytvorení, čím znižuje riziko pretrénovania. Fungovanie algoritmu C4.5 je takmer identické jediný rozdiel ako už bolo spomenuté je, že na výber atribútu do uzla využíva namiesto IG Gain ratio. Rovnako ako pri ID3 vyberá sa ten atribút, ktorý má najvyššiu hodnotu metriky v tomto prípade Gain Ratio.

$$IGR(D,A) = \frac{\mathrm{IG(D,A)}}{\mathrm{SplitInformation(D,A)}} \tag{4}$$

IG(D,A) je vzťah z predošlého agortimu ID3 a *SplitInformation(D,A)* vyzerá nasledovne.

$$SplitInformation(D,A) = -\sum_{v \in Values(A)} \frac{|D_v|}{|D|}\, log_2(\frac{|D_v|}{|D|}) \tag{5}$$

D je dátová množina, *A* je atribút, podľa ktorého rozdeľujeme množinu *D*. *Values(A)* predstavuje množinu rôznych hodnôt atribútu *A*. D_v je podmnožina z množiny *D*, ktorá obsahuje príklady s hodnotou *v* pre atribút *A*. $|D_v|$ je počet príkladov v podmnožine D_v. $|D|$ je celkový počet príkladov v množine *D* (Sharma et al., 2013).

CART (Classification and Regression Trees): Tento algoritmus, ktorý je využívaný na klasifikáciu aj regresiu, vytvára binárne stromy. Pre klasifikáciu je najčastejšie využívaná metrika zvaná Gini impurity (Breiman et al., 1986). Algoritmus CART je vhodný aj pre dátové sady s chýbajúcimi hodnotami a používa metriku Gini impurity na optimalizáciu rozhodovacích stromov pre klasifikáciu. Metriku Gini impurity vypočítame pomocou vzťahu (7). Rovnako ako v predchádzajúcich dvoch algoritmoch, ID3 a C4.5, postupuje algoritmus CART podobným spôsobom. Rozdiel je len v metrike, keď algoritmus nepočíta informačný zisk (IG) alebo pomer zisku (Gain Ratio), ale Gini index. Na základe metriky Gini index sa algoritmus rozhoduje, ktorý atribút je najlepší pre rozdelenie. Na rozdiel od IG a Gain Ratio, je vybraný ten atribút, ktorý má hodnotu Gini indexu najnižšiu.

$$Gini(D) = 1 - \sum_{i=1}^{n} p_i^2 \tag{6}$$

D je dátová množina, *n* je počet tried, p_i je pravdepodobnosť výskytu triedy *i* v množine *D*.

$$GiniIndex(A) = \sum_{v \in Values(A)} \frac{|D_v|}{|D|}\, Gini(D_v) \tag{7}$$

D je celková množina dát, D_v je podmnožina dát, kde atribút *A* ma hodnotu *v*. *Values(A)* sú všetky možné hodnoty atribútu *A* (Thakar & Tahsildar, 2022).

3 Pochopenie výpočtov vo vnútri algoritmov(ID3, C4.5, CART)

Máme dataset pozostávajúci z meteorologických záznamov rôznych dní, kde každý záznam obsahuje informácie o počasí (Outlook), teplote (Temp), vlhkosti (Humidity) a vetre (Wind). Na základe týchto atribútov sa snažíme predpovedať, či sa v daný deň bude hrať futbal (Play football?). Cieľom je vytvoriť model, ktorý by dokázal na základe vstupných meteorologických dát predpovedať, či sa futbal bude hrať alebo nie.

Obr. 1: Dataset meteorologické záznamy

Day	Outlook	Temp	Humidity	Wind	Play futball ?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Zdroj : Vlastné spracovanie

Výpočet pomocou algoritmu ID3:

$$D[9+,5-] \quad Entropy(D) = -\frac{9}{14}\, log_2(\frac{9}{14}) - \frac{5}{14} log_2(\frac{5}{14}) = 0.94$$

Entropy(D) je hodnota entropie pre celý dataset, kde 9+ znamená počet pozitívnych výskytov a 5- počet negatívnych výskytov triedy.

$$D_{Sunny}[2+,3-] \quad Entropy(D_{Sunny}) = -\frac{2}{5}\, log_2(\frac{2}{5}) - \frac{3}{5} log_2(\frac{3}{5}) = 0.971$$

$$D_{Overcast}[4+,0-] \quad Entropy(D_{Overcast}) = -\frac{4}{4}\, log_2(\frac{4}{4}) - \frac{0}{4} log_2(\frac{0}{4}) = 0$$

$$D_{Rain}[3+,2-] \quad Entropy(D_{Rain}) = -\frac{3}{5}\, log_2(\frac{3}{5}) - \frac{2}{5} log_2(\frac{2}{5}) = 0.971$$

Hodnoty entropie pre jednotlivé hodnoty atribútu Outlook (Sunny, Overcast, Rain).

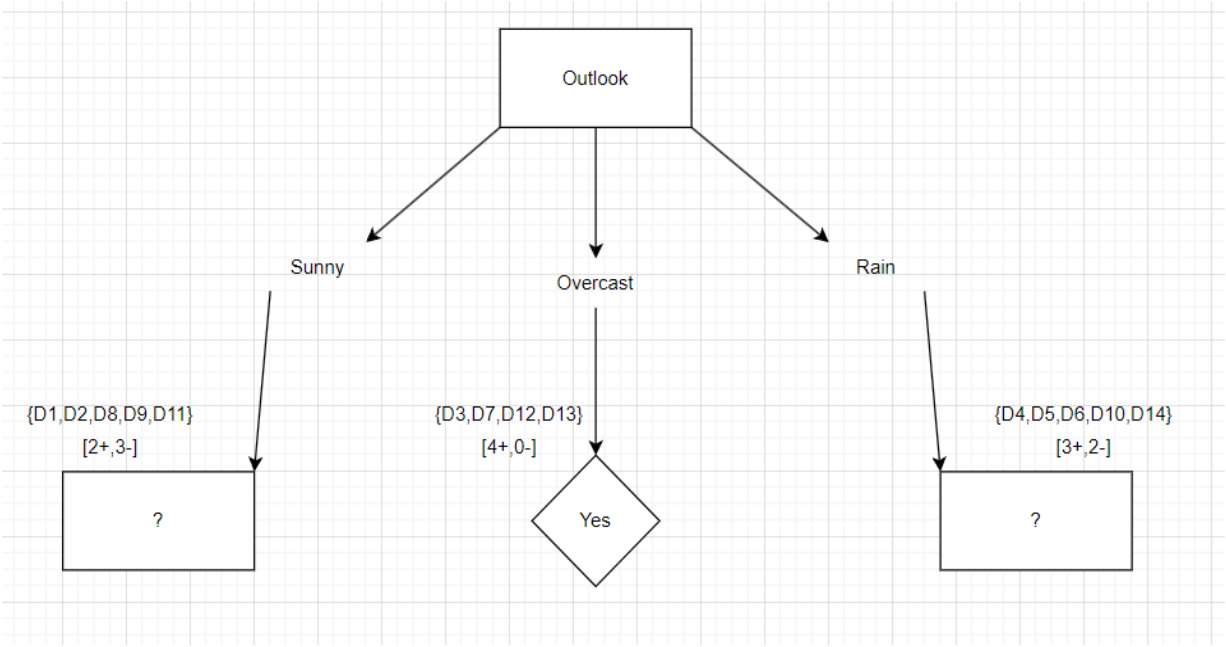
V prípade, že je počet negatívnych alebo pozitívnych výskytov jednohlasný, ako v prípade $D_{Overcast}$ [4+,0-], je entropia danej hodnoty atribútu rovná nule. V prípade, že počet negatívnych výskytov sa rovná počtu pozitívnych výskytov, je entropia danej hodnoty atribútu rovná 1. Tento postup výpočtu entropie opakujeme pre každý atribút a jeho hodnoty v datasete.

$$IG(D, Outlook) = 0.94 - \frac{5}{14}0.971 - \frac{4}{14}0 - \frac{5}{14}0.971 = 0.2464$$
$$IG(D, Temp) = 0.0289$$
$$IG(D, Humidity) = 0.1516$$
$$IG(D, Wind) = 0.0478$$

Vyššie uvedené vypočítané hodnoty predstavujú metriky IG pre jednotlivé atribúty datasetu *D*(Outlook, Temp, Humidity, Wind). Hodnotu information gain pre daný atribút získame použitím vzťahu (3). Po dopočítaní metrík IG analyzujeme hodnoty a vyberáme ten atribút, ktorého hodnota IG je najvyššia.

V algoritme ID3 boli vypočítané hodnoty IG pre atribúty datasetu: Outlook, Temp, Humidity a Wind. Na obrázku 2 môžeme pozorovať, že na základe týchto výsledkov bol ako atribút koreňového uzla zvolený „Outlook“. Tak ako bolo spomenuté vyššie, výber daného atribútu je založený na najvyššej hodnote IG. Strom sa následne z koreňového uzla delí na tri vetvy, ktoré zodpovedajú hodnotám atribútu „Outlook“ (Sunny, Overcast, Rain). Všimnime si, že jedna z týchto vetiev je zakončená listom a zvyšné dve vetvy sú zakončené uzlom. V prípade hodnoty Overcast je vetva zakončená listom, čo znamená, že z tohto uzla nevychádzajú žiadne ďalšie vetvy a nesie priamo konečné rozhodnutie, teda konečnú klasifikáciu triedy. Pokiaľ sú všetky dáta v uzle z jednej kategórie alebo triedy, nie je potrebné ďalšie rozdelenie, pretože tento uzol už dokonale klasifikuje dáta.

Obr. 2: Vetvenie stromu



Zdroj: Vlastné spracovanie

V zvyšných dvoch hodnotách atribútu „Outlook“, konkrétne Sunny a Rain, pozorujeme rozpor v zaradení. V týchto uzloch je výpočet najvhodnejšieho atribútu na ďalšie delenie analogický k výpočtu pri koreňovom uzle. Jediným rozdielom je, že pôvodnú entropiu celej dátovej sady, vypočítanú z *D*=[9+,5-], nahradí entropia príslušnej vetvy: pre Outlook(Sunny), t. j. [2+,3-], ktorý reprezentuje uzol vľavo, a pre Outlook(Rain), t. j. [3+,2-]. Polia tabuľky

zredukujeme len na tie riadky, v ktorých hodnota atribútu „Outlook“ zodpovedá hodnote danej vetvy. Stĺpec atribútu „Outlook“ sa z tabuľky odstráni.

Obr. 3: Parciálna tabuľka pre Outlook (Sunny)

Day	Temp	Humidity	Wind	Play futball ?
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

Zdroj : Vlastné spracovanie

Na obrázku 3 vidíme parciálnu tabuľku pre hodnotu Sunny atribútu Outlook, na základe ktorej vykonáme nasledovné výpočty.

$$D_{Sunny}[2+,3-] \quad Entropy(D_{Sunny}) = -\frac{2}{5}\, log_2(\frac{2}{5}) - \frac{3}{5} log_2(\frac{3}{5}) = 0.971$$

$D_{Hot}\ [0+,2-]$ $Entropy(D_{Hot}) = 0$
$D_{Mild}[1+,1-]$ $Entropy(D_{Mild}) = 1$
$D_{Cool}[1+,0-]$ $Entropy(D_{Cool}) = 0$

$$IG(D_{Sunny}, Temp) = 0.971 - \frac{2}{5}0 - \frac{2}{5}1 - \frac{1}{5}0 = 0.570$$

Podobne, ako bola vypočítaná entropia pre celý dataset, v tomto prípade sme vypočítali celkovú entropiu pre parciálny dataset D_{Sunny} a následne entropie pre jednotlivé hodnoty atribútu Temp. V nasledujúcom kroku sme vypočítali metriku IG pre hodnotu atribútu Temp v parciálnej tabuľke D_{Sunny}. Ďalším krokom bude dopočítanie *IG* aj pre zvyšné atribúty, Humidity a Wind, rovnakým spôsobom, akým bola dopočítaná hodnota *IG* pre atribút Temp.

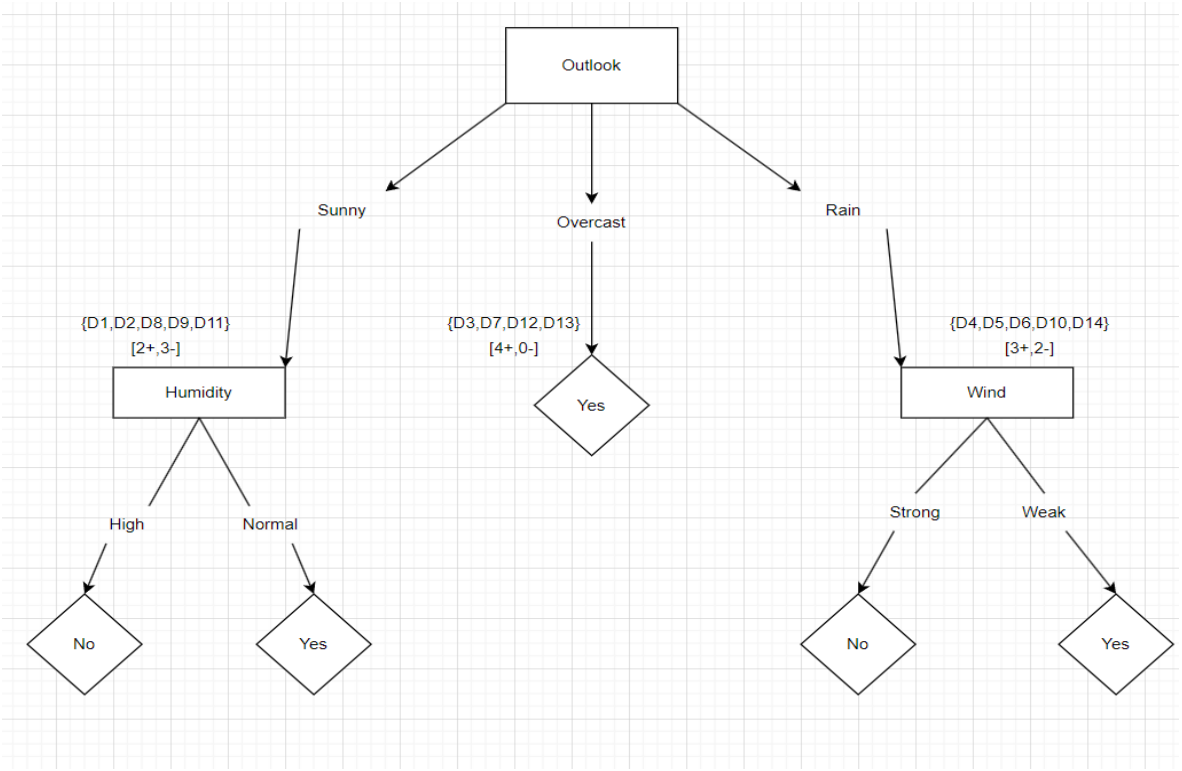
$$IG(D_{Sunny}, Humidity) = 0.97$$
$$IG(D_{Sunny}, Wind\ ) = 0.0192$$

Po vyčíslení hodnôt *IG* a ich analýze je najvhodnejším atribútom pre ďalšie delenie datasetu z vetvy Sunny atribút Humidity.

Na Obrázku 4 je znázornený kompletný rozhodovací strom pre úlohu hrania futbalu. Ako sme preukázali prostredníctvom výpočtu, v ľavom uzle je najlepšou voľbou pre rozdelenie atribút Humidity. Z tohto uzla vychádzajú dve vetvy: jedna s hodnotou High a druhá s hodnotou Normal, pričom obe vetvy sú zakončené listom. Tieto vetvy nám poskytujú jednoznačnú klasifikáciu, podobne ako to bolo v prípade hodnoty Overcast atribútu Outlook. Pre jednotlivé hodnoty (High, Normal) sa odpovede na otázku, či hrať futbal, neodlišovali a v každom prípade

bola dosiahnutá zhoda: presne dvakrát pre hodnotu Normal a trikrát pre hodnotu High. Na pravej strane stromu, po vypočítaní jednotlivých hodnôt IG, identifikujeme najlepší atribút pre ďalšie delenie uzla vetvy Rain: atribút Wind. Podobne ako v prípade atribútu Temp, aj atribút Wind vytvára dve vetvy (Strong, Weak), pričom obe tieto vetvy sú, rovnako ako v prípade Temp, zakončené listom. To znamená, že hodnoty atribútu Wind nám jednoznačne určujú, či sa má futbal hrať alebo nie.

Obr. 4: Rozhodovací strom príkladu hrať futbal ?



Zdroj : Vlastné spracovanie

C4.5: Výpočet pomocou algoritmu C4.5 je takmer identický s výpočtom algoritmu ID3, ktorý svoje rozhodovanie zakladá na informačnom zisku. Rozdiel je v tom, že C4.5 používa na rozhodovanie "Gain ratio". Tento pomer je získaný delením informačného zisku hodnotou "SplitInformation", ktorá je definovaná vzťahom (4), a ako atribút pre delenie je zvolený ten, ktorý má "Gain ratio" najvyšší.

$$IG(D, Outlook) = 0.2464$$

$$IGR(D, Outlook) = \frac{0.2464}{-\frac{5}{14}log_2\left(\frac{5}{14}\right) - \frac{4}{14}log_2\left(\frac{4}{14}\right) - \frac{5}{14}log_2\left(\frac{5}{14}\right)}$$

$IGR(D, Outlook) = 0.156$
$IGR(D, Temp) = 0.019$
$IGR(D, Humidity) = 0.152$
$IGR(D, Wind) = 0.049$

Hodnota *IGR* atribútu "Outlook" je najvyššia, tým pádom je najlepším atribútom pre delenie, atribút "Outlook". S každým ďalším vetvením stromu od koreňového uzla postupuje algoritmus C4.5, podobne ako algoritmus ID3. Dáta sú rozdelené v každom ďalšom uzle stromu podľa hodnoty *IGR*.

CART: Algoritmus CART pre klasifikáciu pracuje tak, že ako aj predošlé dva algoritmy, využíva rekurzívne rozdeľovanie dátovej množiny na základe hodnôt atribútov. Na identifikáciu optimálneho rozdelenia sa opiera o metriku nazývanú "Gini impurity" alebo "Gini Index", ktorá je definovaná vzťahom (7).

$$Gini(D_{Sunny}) = 1 - (\frac{2}{5})^2 - (\frac{3}{5})^2$$
$$Gini(D_{Sunny}) = 0.48$$
$$Gini(D_{Overcast}) = 0$$
$$Gini(D_{Rain}) = 0.48$$

$$GiniIndex(Outlook) = \frac{5}{14}0.48 + \frac{4}{14}0 + \frac{5}{14}0.48$$

$$GiniIndex(Outlook) = 0.329$$
$$GiniIndex(Temp) = 0.429$$
$$GiniIndex(Humidity) = 0.36$$
$$GiniIndex(Wind) = 0.375$$

Najnižší Gini index v našom príklade má atribút "Outlook". Výber atribútu na základe metriky Gini index sa odlišuje od predchádzajúcich metrík, ako sú Information Gain (IG) a Gain Ratio, v tom zmysle, že preferujeme atribút s najnižšou hodnotou Gini indexu, ktorý je následne považovaný za najlepší na rozdelenie dát. Atribút s najnižšou hodnotou Gini indexu je zvolený ako koreňový uzol na rozdelenie dát na prvej úrovni stromu. V tomto prípade bude vetvenie stromu vyzerať rovnako ako pri použití algoritmov ID3 alebo C4.5, čo je ilustrované na Obrázku 2.

4 Implementácia rozhodovacieho stromu na dataset pomocou programovacieho jazyka Python

Na začiatku každého projektu je dôležité mať jasný cieľ. Keď vieme, čo chceme dosiahnuť, môžeme začať zhromažďovať dáta a potom ich prekontrolovať. Dáta je potrebné očistiť teda upraviť, zbaviť sa nežiaducich chýb, zmeniť ich tvar a následne rozdeliť na rôzne časti: na učenie, kontrolu a testovanie modelu. S takto pripravenými dátami môžeme vybrať najlepší spôsob, ako vytvoriť model. Po jeho vytvorení a otestovaní môžeme model vylepšovať a nakoniec aj použiť. Dôležité je nezostať pri jednej verzii modelu. Keďže sa dáta a situácie menia, je dobré model prispôsobovať novým podmienkam.

Konečné hodnotenie modelu na testovacej množine určuje jeho pripravenosť na implementáciu v reálnom prostredí. Avšak, proces strojového učenia je často iteratívny, a preto je dôležité pravidelne monitorovať a aktualizovať model vzhľadom na nové požiadavky alebo zistené problémy.

Príklad použitý v tretej kapitole využijeme aj na demonštráciu vytvorenia modelu rozhodovacieho stromu pre klasifikáciu v jazyku Python. Zároveň poskytneme grafickú ilustráciu finálneho stromu, ktorý bol vytvorený pomocou algoritmu v jazyku Python.

Obr. 5: Náhľad datasetu play futball

	Outlook	Temp	Humidity	Wind	Play_futball
01	Sunny	Hot	High	Weak	No
02	Sunny	Hot	High	Strong	No
03	Overcast	Hot	High	Weak	Yes
04	Rain	Mild	High	Weak	Yes
05	Rain	Cool	Normal	Weak	Yes
06	Rain	Cool	Normal	Strong	No
07	Overcast	Cool	Normal	Strong	Yes
08	Sunny	Mild	High	Weak	No
09	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Zdroj : Vlastné spracovanie

Na obrázku 6 je zobrazený kód na vytvorenie klasifikačného modelu, pričom bola hlavne použitá knižnica Scikit-learn (Pedregosa et al., 2011), ktorá je jednou z najpopulárnejších knižníc v jazyku Python pre strojové učenie. Ponúka širokú škálu nástrojov pre štatistické modelovanie a klasifikáciu, vrátane regresie, klasifikácie, zhlukovania a redukcie dimenzionality. V kóde bola ako cieľová premenná zvolená "Play_futball"; kategorické premenné boli zmenené na numerické pomocou funkcie `LabelEncoder()`. Bez použitia funkcie LabelEncoder by algoritmus na vytvorenie stromu nemohol spracovať textové dáta, pretože nevie priamo pracovať s reťazcami a požaduje numerické hodnoty pre svoje výpočty. Enkódovanie hodnôt atribútov umožňuje presnú prácu s kategorickými premennými. Kritérium rozdelenia "criterion" sme nastavili na "entropy", čo znamená, že strom bude vytvorený pomocou algoritmu ID3. V tomto konkrétnom príklade neboli dáta rozdelené na trénovaciu a testovaciu množinu, keďže pracujeme s veľmi obmedzeným datasetom, ktorý slúži primárne na demonštráciu vytvárania modelu a vykresľovania rozhodovacieho stromu. Model bol natrénovaný na celom datasete s použitím metódy `clf.fit(X,y)`, kde *X* predstavuje vstupné vlastnosti a *y* je cieľová premenná. Po dokončení trénovania bola použitá metóda `clf.predict(X)` na predikciu cieľových hodnôt pre celý dataset, čím vznikla predikovaná množina hodnôt X_pred. Následne bola aplikovaná funkcia `accuracy_score(y, X_pred)` na vypočítanie presnosti modelu porovnaním predikovaných hodnôt `X_pred` s reálnymi hodnotami *y*.

Obr. 6: Vytvorenie klasifikačného modelu

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn import tree
import matplotlib.pyplot as plt
from sklearn.tree import export_text
from sklearn.metrics import accuracy_score
Play = pd.read_csv("futball_data.csv")
Le = LabelEncoder()
Play['Outlook'] = Le.fit_transform(Play['Outlook'])
Play['Temp'] = Le.fit_transform(Play['Temp'])
Play['Humidity'] = Le.fit_transform(Play['Humidity'])
Play['Wind'] = Le.fit_transform(Play['Wind'])
Play['Play_futball'] = Le.fit_transform(Play['Play_futball'])
y = Play['Play_futball']
X = Play.drop(['Play_futball'], axis=1)
clf = tree.DecisionTreeClassifier(criterion='entropy')
clf = clf.fit(X, y)
plt.figure(figsize=(12, 12))
tree.plot_tree(clf, feature_names=X.columns, class_names=['No', 'Yes'], filled=True)
plt.show()
X_pred = clf.predict(X)
accuracy = accuracy_score(y, X_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

Zdroj : Vlastné spracovanie

Obr. 7: Presnosť klasifikačného modelu

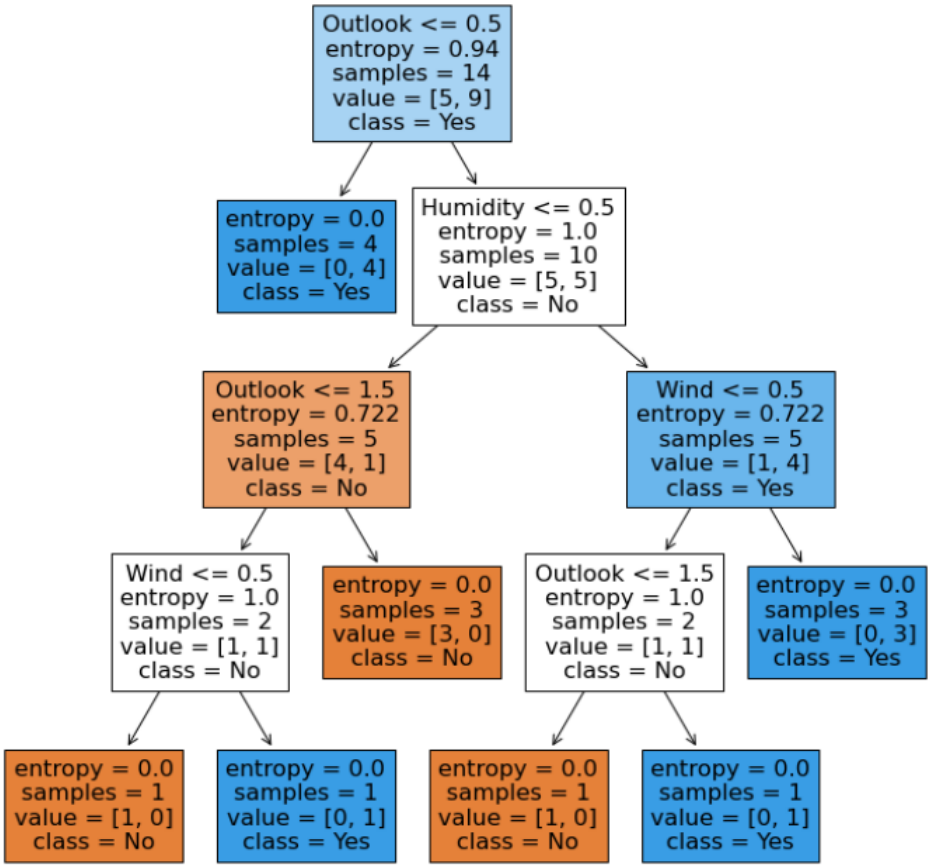
```
Accuracy: 100.00%
```

Zdroj : Vlastné spracovanie

Hodnota presnosti, zobrazená na obrázku 7, reprezentuje mieru presnosti klasifikačného modelu s akou model predpovedá hodnoty. Presnosť je základnou metrikou pre hodnotenie výkonu klasifikačných modelov a je definovaná ako pomer správne klasifikovaných príkladov ku všetkým príkladom. V našom prípade presnosť dosahuje hodnotu 100%, čo naznačuje, že model má vynikajúcu schopnosť predpovedať hodnoty závislej premennej Play_futball.

Na Obrázku 8 je zobrazený rozhodovací strom, ktorý bol vykreslený v jazyku Python pomocou knižnice matplotlib. Hneď na prvý pohľad si môžeme všimnúť rozdiel oproti stromu vytvorenému v časti 3. Ak sa zameriame na prvý uzol obrázka 8, kde je uvedené "Outlook <= 0.5", hodnota 0.5 reprezentuje prahovú hodnotu určenú na rozdelenie dát. Táto hodnota vznikla na základe transformácie kategorických dát na numerické. Prahová hodnota je stanovená tak, aby čo najviac optimalizovala čistotu v danom datasete. Cieľom algoritmu je nájsť hodnotu pre každý atribút, ktorá efektívne rozdelí dáta podľa cieľovej premennej. "Entropy" označuje hodnotu entropie, "samples" zastupuje počet príkladov, zatiaľ čo "value" predstavuje hodnoty tried: 5 pre "no" a 9 pre "yes". Finálne rozhodnutie je "class yes".

Obr. 8: Vykreslenie rozhodovacieho stromu pomocou jazyka Python(ID3)



Zdroj : Vlastné spracovanie

Farebné označenie jednotlivých uzlov a listov stromu sa odvíja od hodnoty entropie. Uzly s vyššou hodnotou entropie sú zafarbené menej intenzívne, a uzly s nízkou alebo nulovou hodnotou entropie sú zafarbené intenzívne. Konkrétne, na obrázkoch rozhodovacích stromov v tomto príspevku tmavo oranžové uzly predstavujú "čistú" alebo dominantnú triedu "Nie"(No), a tmavo modré uzly predstavujú triedu "Áno"(Yes). Čím je hodnota entropie vyššia, tým je farba svetlejšia. V prípade najvyššej neistoty, teda ak hodnota entropie dosahuje hodnotu 1, sú uzly zafarbené bielou farbou.

V tomto prípade je koreňovým uzlom opäť atribút Outlook, avšak jeho vetvenie sa líši. Hlavným rozdielom je, že knižnica Scikit-learn v jazyku Python implementuje binárny strom, ktorý pri vetvení rozdelí dáta na dve skupiny. Na rozdiel od stromu z Obrázku 4 môže tento strom opakovane využívať rovnaké atribúty, ale len ich nevyužité podmnožiny. Niekedy môže atribút poskytnúť optimálne rozdelenie dát viackrát, avšak v rôznych kontextoch alebo s rôznymi prahovými hodnotami. Atribút, môže prvýkrát rozdeliť celú dátovú množinu a neskôr znova, keď sa dáta ďalej rozdeľujú podľa iných atribútov. Opätovné využitie atribútu umožňuje stromu sa lepšie prispôsobiť kombináciám hodnôt atribútov, čím dosahuje presnejšie a relevantnejšie rozdelenia. Môžeme pozorovať, že strom na Obrázku 8 je zakončený listami, ktoré majú hodnotu entropie 0, a to znamená že miera neistoty v danom liste je nulová.

Z Obrázku 8 je jasné, že vetvenie je binárne, hoci atribút Outlook má tri hodnoty. Podobne ako pri strome na Obrázku 4, jeden list zodpovedá hodnote "Overcast" atribútu Outlook. Rozdiel však spočíva v tom, že ďalšia vetva zahŕňa aj zvyšné hodnoty atribútu (Sunny,

Rain), ktoré neumožňujú jednoznačné rozdelenie a teda neposkytujú presnú klasifikáciu. Algoritmus pre ďalšie vetvenie v nasledujúcom uzle pracuje s redukovanou tabuľkou.

Obr. 9: Redukovaná tabuľka

Day	Outlook	Temp	Humidity	Wind	Play_futball
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D14	Rain	Mild	High	Strong	No

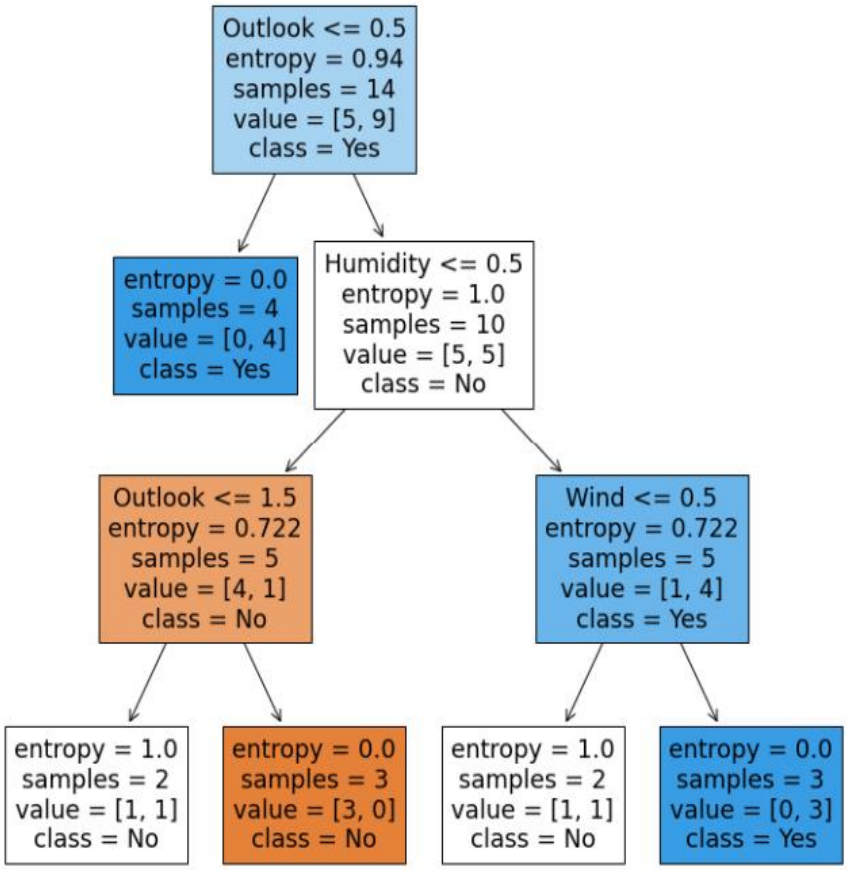
Zdroj : Vlastné spracovanie

Na obrázku 9 je zobrazené, ako algoritmus redukoval pôvodnú tabuľku. Oproti parciálnej tabuľke zobrazenej na obrázku 3, tabuľka použitá na výpočet ďalšieho vetvenia, v tomto prípade, stále obsahuje atribút, ktorý bol už zvolený do koreňového uzla. Algoritmus neodstránil celý stĺpec prislúchajúci atribútu Outlook, ale len tie riadky, ktoré na základe hodnôt v atribúte koreňového uzla jednoznačne určili triedu. Konkrétne boli odstránené riadky s hodnotou Overcast, keďže táto hodnota jednoznačne určila klasifikáciu. Týmto spôsobom algoritmus vytvoril redukovaný dataset pre ďalšie vetvenie. Tento prístup ilustruje, ako je vetvenie a redukcia atribútov spracovaná v jazyku Python a knižnici Scikit-learn pri vytváraní rozhodovacích stromov. Je to odlišný prístup oproti tomu, ktorý sme demonštrovali ručne, a poukazuje na to, ako môžu rôzne implementácie rozhodovacích stromov priniesť odlišné štruktúry a výsledky. Rozhodovací strom zobrazený na obrázku 10 bol vytvorený modifikáciou kódu z Obrázku 6. V riadku 23 bol kód upravený na:

```
clf=tree.DecisionTreeClassifier(criterion='entropy',max_depth=
3).
```

Pridali sme hyperparameter `max_depth` s hodnotou 3, čo znamená, že strom sa bude vetviť maximálne na tri úrovne. Úpravou kódu z obrázku 6 alebo zmenou hyperparametrov je možné vytvoriť a vykresliť rôzne varianty rozhodovacích stromov. Na obrázku 10 je zrejmé, že strom obsahuje uzly s hodnotami entropie nielen 0, ale aj 1. Hodnota entropie 1 predstavuje najvyššiu mieru neistoty v rozdelení dát. Tento strom nie je optimálny, čo dokazuje aj pokles presnosti z 100% v predošlom strome na obrázku 8 na 85%. Presnosť na trénovacích dátach nemusí byť jasným indikátorom kvality vytvoreného rozhodovacieho stromu, pretože model sa môže až príliš prispôsobiť trénovacím dátam. Ak je model pretrénovaný, tak môže na reálnych dátach dosahovať horšie výsledky než strom, ktorý dosahoval na trénovacích dátach nižšie hodnoty presnosti.

Obr. 10: Rozhodovací strom (ID3), max_depth=3



Zdroj : Vlastné spracovanie

Teraz otestujme presnosť stromu, ktorý sme vytvorili na Obrázku 4. Keďže Python umožňuje tvorbu iba binárnych stromov, musíme improvizovať. Na základe Obrázku 4 si vytvoríme pravidlovú funkciu. Kód je zobrazený na Obrázku 11. Po spustení kódu získame výpis: "Accuracy based on decision rules: 100.00%." To znamená, že náš vytvorený strom perfektne klasifikoval dáta.

Je dôležité poznamenať, že absencia rozdelenia dát na trénovaciu a testovaciu množinu môže viesť k nadmernému prispôsobeniu modelu, známemu ako overfitting. Tento jav môže negatívne ovplyvniť schopnosť modelu generalizovať na nových dátach. Avšak v kontexte týchto demonštračných príkladov bolo hlavným cieľom ilustrovať postup vytvárania a vykresľovania rozhodovacieho stromu pomocou jazyka Python. Pri modelovaní rozhodovacích stromov existuje riziko pretrénovania, čo znamená, že model sa príliš úzko prispôsobí trénovacím dátam na úkor svojej schopnosti generalizovať. Pretrénovanie sa často prejavuje prílišnou hĺbkou stromu a nadmerným počtom uzlov, ktoré môžu byť dôsledkom nedostatočného množstva dát alebo chýbajúcej regulácie. Neobmedzenie komplexnosti stromu môže viesť k jeho nadmernému rozrastaniu a pretrénovaniu. Orezávanie stromu, ktoré zahŕňa odstránenie niektorých koncových listov na zníženie komplexnosti je jednou z metód, ako predchádzať pretrénovaniu. Ďalšími opatreniami môžu byť nastavenia maximálnej hĺbky stromu alebo obmedzenie počtu príkladov v uzle potrebných na rozdelenie. Je teda dôležité si zapamätať, že cieľom je vytvoriť taký model, ktorý efektívne predpovedá výsledky na neznámych dátach, a nie len dosiahnuť vysokú presnosť na dátach, na ktorých bol natrénovaný.

Obr. 11: Zakódovanie stromu z obrázku 4

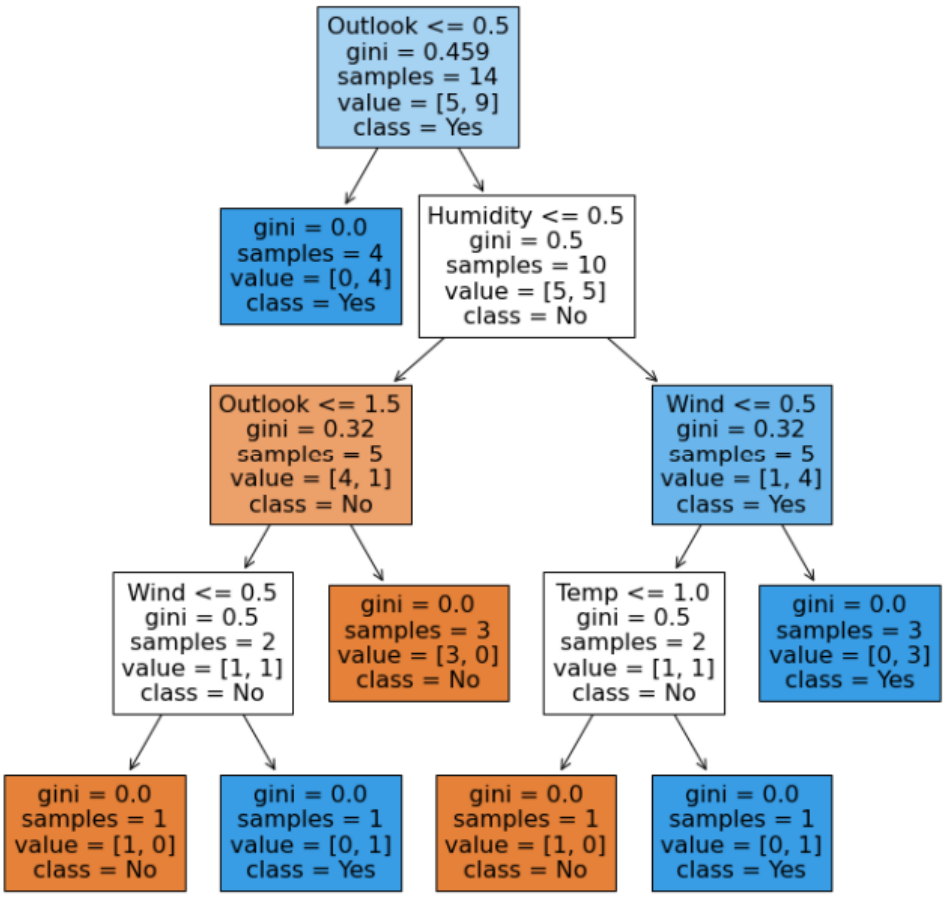
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
Play = pd.read_csv("futball_data.csv")
Le = LabelEncoder()
Play['Outlook'] = Le.fit_transform(Play['Outlook'])
Play['Temp'] = Le.fit_transform(Play['Temp'])
Play['Humidity'] = Le.fit_transform(Play['Humidity'])
Play['Wind'] = Le.fit_transform(Play['Wind'])
Play['Play_futball'] = Le.fit_transform(Play['Play_futball'])
print(Play)
y = Play['Play_futball']
X = Play.drop(['Play_futball'], axis=1)
def decision_based_on_rules(row):
    if row['Outlook'] == 2:  # Sunny
        if row['Humidity'] == 0:  # High
            return 0  # No
        else:
            return 1  # Yes
    elif row['Outlook'] == 0:  # Overcast
        return 1  # Yes
    else:  # Rain
        if row['Wind'] == 0:  # Strong
            return 0  # No
        else:
            return 1  # Yes
predictions = X.apply(decision_based_on_rules, axis=1)
accuracy = accuracy_score(y, predictions)
print(f"Accuracy based on decision rules: {accuracy * 100:.2f}%")
```

Zdroj : Vlastné spracovanie

Na obrázku 12 môžeme vidieť rozhodovací strom, ktorý bol vytvorený pomocou algoritmu CART. Ak chceme použiť algoritmus CART, musíme upraviť kód zobrazený na obrázku 6 a zmeniť hodnotu parametra "criterion" na "gini". Strom zobrazený na obrázku 12 je takmer identický so stromom na obrázku 8. Jediným rozdielom medzi týmito dvoma stromami je uzol na pravej strane: na tretej úrovni sa v strome z obrázku 12 uzol delí podľa atribútu "Temp", zatiaľ čo v strome na obrázku 8 sa tento uzol delil podľa atribútu "Outlook".

Na záver si predstavme situáciu, že namiesto kategorických hodnôt v atribúte by sme mali spojité(numerické) premenné, napríklad hodnoty atribútu Temp by boli v stupňoch. V takomto prípade algoritmus vykonáva takzvané binárne rozdelenie. Algoritmus hľadá najlepší možný spôsob, ako rozdeliť dáta. Dáta delí na dve skupiny z hľadiska cieľovej premennej. Algoritmus najprv zoradí hodnoty spojitej premennej, v našom prípade teploty, od najnižšej po najvyššiu. Následne skúma rôzne možné body rozdelenia medzi každou dvojicou susedných hodnôt. Každý takýto rozdeľovací bod reprezentuje potencionálnu prahovú hodnotu, pri ktorej by sa dáta rozdelili do dvoch skupín. Pre každý rozdeľovací bod algoritmus vypočíta miery nečistoty Gini impurity alebo entropiu, aby určil, aké "čisté" by boli skupiny po rozdelení. Algoritmus vyberá taký rozdeľovací bod, ktorý minimalizuje nečistotu, teda ktorý najlepšie oddeľuje dáta podľa cieľovej premennej.

Obr. 12: Vykreslenie rozhodovacieho stromu pomocou jazyka Python(CART)



Zdroj : Vlastné spracovanie

5 Záver

V tomto článku sme sa zoznámili s kľúčovými konceptami klasifikačných rozhodovacích stromov a ich aplikáciou v oblasti strojového učenia a dátovej analýzy. Uviedli sme si základné algoritmy spojené s rozhodovacími stromami, konkrétne ID3, C4.5 a CART. Predstavili sme si metriky, ktoré tieto algoritmy využívajú na evaluáciu a výber najlepších rozdelení dát. Pre ID3 sme sa venovali entropii a Information Gain, pre C4.5 Gain Ratio a pre CART Gini index. Zároveň sme aj prakticky demonštrovali vytvorenie a vizualizáciu rozhodovacieho stromu pomocou jazyka Python. Ukázali sme, ako môže byť s pomocou knižnice scikit-learn a matplotlib vytvorený, natrénovaný a vykreslený rozhodovací strom, a ako je možné vyhodnotiť jeho výkon pomocou metrík, ako je napríklad presnosť.

Celkovo nám článok poskytol komplexný pohľad na rozhodovacie stromy, ich teoretické základy, metriky a praktickú aplikáciu. Praktická demonštrácia a vizualizácia nám umožnili lepšie pochopiť vytváranie rozhodovacích stromov a ich silu a potenciál v oblasti dátovej analýzy a strojového učenia. Ich flexibilita a interpretovateľnosť ich robia ideálnou voľbou pre mnoho aplikácií. Avšak, ako vždy, kľúčovým krokom je pochopenie dát, s ktorými pracujeme, a nastavenie algoritmu tak, aby zodpovedal špecifikám daného problému.

Literatúra

1. Alpaydin, E. (2020). Introduction to Machine Learning. MIT Press.
2. Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1986). Classification and Regression Trees. Wadsworth.
3. Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.
4. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794).
5. Dietterich, T. G. (2000). Ensemble methods in machine learning. In Multiple classifier systems (pp. 1-15). Springer, Berlin, Heidelberg.
6. Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.
7. Kelleher, J. D., Mac Namee, B., & D'Arcy, A. (2015). Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies. MIT Press.
8. Mishra, S. (2020). Lecture 11.1. National Institute of Science Education and Research (NISER). Získané z https://niser.ac.in/~smishra/teach/cs460/2020/lectures/lec11_1/.
9. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.
10. Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1(1), 81-106.
11. Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.
12. Rokach, L., & Maimon, O. (2014). Data mining with decision trees: theory and applications. World Scientific.
13. Sharma, Seema & Agrawal, Jitendra & Sharma, Sanjeev. (2013). Classification Through Machine Learning Technique: C4. 5 Algorithm based on Various Entropies. International Journal of Computer Applications. 82. 28-32. 10.5120/14249-2444.
14. Smith, J., & Roberts, P. (2022). Decision Trees in Medical Diagnosis: A Comprehensive Study. *Research and Reports in Health Practice*, 5(2), 123-134.
15. Thakar, C., & Tahsildar, S. QuantInsti https://blog.quantinsti.com/gini-index/

Modelling of the Public Procurement State Process

Daniel Dudek[1]

Abstract
Process models are generally used to document a described object, as educational material, or as a simplified overview of a more complex object, enabling the decision maker to make a faster and more efficient decision. Such processes described by law or regulation are also found in the operation of the state. The primary goal of this paper is to create a flow chart of the life cycle process of creating a model law, regulation, or state process. The secondary goal is to demonstrate the application of the process from this flowchart to the modelling of a public procurement process conducted through a digital interface. The created models can serve not only as a guide for creating a state process model but also as a guide for realising public procurement through digital platforms.

Key words
Public Procurement, Business Process Model and Notation, Digital Government, Policy management, Process modelling

JEL classification
K00, O21, O31

1 Introduction

Business process modelling has been used in business practices for several decades. Today, modelling, built around business processes, brings companies a better overview across their structure, higher profitability, and greater stability. Another benefit of building models in enterprises is the ability to describe difficult tasks written in a simplified form into a more concise and easier-to-understand model.

Public procurement is defined as the rules and procedures for selecting a provider designed to ensure public funds' most cost-effective and efficient use. Public procurement is only launched by the public or state institutions, while private parties primarily provide the subject of public procurement. The procurement process results in the conclusion of a contract between the procurer and the successful provider, for example, the award of a contract or a concession. Considering various market factors, PP (Public Procurement) rules provide specific procedures to ensure that the needs of the public sector are met by private forces as rationally, transparently, and equitably as possible (Vargicová, 2021).

As such, digitisation is being developed and used in all sectors of human activity in which data, information and knowledge occur, in any form and for any purpose. Digitisation could be defined as the transformation, usually of less efficient processes, into more efficient ones, using information technology.

The paper aims to show the possibility and methods of conceptual modelling of laws, regulations, and state processes like the methodology used in the corporate sphere. The output of the paper is two process models.

In the first section, we discussed the theory regarding public procurement and the history of modelling of laws, regulations, and state processes. In the second section, we described the

[1] Ing. Daniel Dudek, University of Economics in Bratislava, Faculty of Economic Informatics, Department of Operations Research and Econometrics, Dolnozemská cesta 1, 851 04 Bratislava, daniel.dudek@euba.sk.

methodology of the work, the selection of the modelling method and the modelling tool. In the third section, we showed the process of modelling state matters throughout their life cycle, as well as the model of the procurement process carried out through the electronic platform.

Modelling of state processes into conceptual models has been done in some form since the existence of laws, at least in the form of pictograms. However, it is impossible to say precisely when state processes have been modelled using established rule modelling techniques since no public information is available. This is because such models are mainly used for internal purposes. At the same time, they are only for support, and it is not always guaranteed that the models are syntactically and semantically correct. According to the expert with whom the public procurement model was discussed and refined, the process models currently used in public procurement have all the details and facts of the law that the model describes.

Our vision is that every law in the state should have an operations manual composed of models. These models could make it easier for laypeople in the law to understand a complex law, and legislators would be better able to navigate through it. At the same time, more people would be able to participate in lawmaking because people would be able to file objections through a law-opposing web portal (for example, slovlex.sk or eur-lex.europa.eu). This would enable experts in scientific disciplines who have a better understanding of the issues, but not of the law, to make valid law objections, or ordinary folk who would at least have a broader overview of the laws that are being passed, and thus have a more critical mindset in making choices and perceiving the problems and operations of the state.

2 Literature review

A business process is an established and recurring activity consisting of several steps carried out in an enterprise to achieve one of its objectives (Jurík, 2018). A business process model is a graphical representation of a business process or workflow and its related sub-processes. A model can also be defined as a simplified view of reality. Process modelling produces comprehensive, quantitative diagrams and flow charts that contain critical insights into the operation of a given process, including the following information (IBM Cloud Education; 2021):" The events and activities that occur within the workflow; Who owns or initiates these events and activities; Decision points and the different paths that workflows can take based on their outcomes; Features involved in the process; Timing of the overall process and the individual steps of the process; Success and failure rates of the process."

The state process is like the business process, but the state process is carried out in a state environment and is carried out by actors that are specific units of government or citizens. It is a precise sequence of steps (activities) that build on each other and provide a predefined output, and these activities are performed by the actors in the state while trying to achieve the state's determined goals. All entities with either legislative, executive, or jurisdictional power can iterate the state process.

The key aspects of business process modelling can be classified into 4 parts (IBM Cloud Education, 2021):

1. Process models can also be created manually but are now beginning to be used more by data mining algorithms that use the data contained in event logs to create workflow models in their existing form. This way, processes can be better modelled and more automated as a generic best practice for many businesses.
2. Because process models are based on quantitative data, they offer a truly objective view of workflows as they exist in practice, including key data, metrics or events that would otherwise go unnoticed. For example, by creating a process model, a software company may discover that a process is incomplete because it takes too long and may be interrupted by an unexpected event (such as customer impatience).

3. Process models usually represent using one of two standardised business process graphical notation styles: the Business Process Modelling Notation (BPMN) or the Unified Modelling Language (UML). Within these notation systems, certain visual elements have generally accepted meanings when used in a process model.

4. Business process models should not be confused with process maps, another common type of business process diagram. Process maps are created manually based on employee reports, providing a higher-level view of workflows. Process models are in-depth analyses based on data that present a more objective view of workflows.

The first use of a conceptual model to describe the process can be traced back to prehistoric times, specifically to cave paintings describing hunting animals. Further back in antiquity, with the advent of writing, they could describe more complex concepts of processes and laws.

However, the first official use of a conceptual model to describe the elements of a process in diagrammatic form is the Adamiecki and Gantt charts, which were popularised at the turn of the 19th century and had their first mass use during the First World War (Marsh, 1974). In addition to depicting sequences of activities, these techniques also recorded the time aspect of processes. Hence, these diagrams were helpful when used in project management, where they are still used to this day. The main disadvantage of these techniques is that they fail to capture the decision logic, cycles, and alternative process routing. Thus, the full complexity of the process cannot be captured in such a diagram.

In 1921, the world's most famous process modelling technique, the flow chart, was created. The flow chart was designed by Lillian and Frank Gilbreth and introduced to the world at a presentation "Process Charts: First Steps in Finding the One Best Way to Do Work" to the American Society of Mechanical Engineers (ASME) (Gilbreth & Gilbreth, 1921). This flow chart differed somewhat from the current form, and it was not until 1947 that ASME established the form of the flow chart into the form that is still used today (ASME, 1947).

By the late 1980s, there were more than fifty separate modelling languages, each with syntax, structure, and notation. Many problems were associated with this bewildering variety of languages, and it was apparent that "such a situation" caused projects to be more costly and time-consuming, and projects had no guarantee of ever being completed (Erickson & Siau, 2013). In response to this very significant problem, the Object Management Group (OMG) was founded and, in 1997 created a single modelling standard, namely the UML (Thompson & Platt, 2015). Unified Modelling Language (UML) is vital in modelling relationships between business or state processes. This is because it visually expresses the behaviour and structure of a system, process, or procedure. UML helps point out possible application structure errors, system behaviour or sequence of processes (Microsoft Corp., 2023).

However, UML was mainly designed for software design, development, and implementation. For UML, modelling processes not directly related to software development was not a suitable task. In 2004, the Business Process Management Initiative (BPMI) created the first version of the BPMN (business process model and notation) diagram, which served as a more comprehensive alternative to flowcharts while being able to capture more specific situations. In 2005, BPMI and OMG merged, and since that year, BPMN has been implemented in the UML as an extension. The latest version of BPMN was released in 2014, and it can also fully interact with other modelling techniques from the UML (OMG, 2023).

As the management and functioning of the modern state begin to be centred around current technological trends, it is natural that practices established in the corporate environment could be used to properly deploy them, given that technological trends have been developed and implemented with the corporate trends for several decades. Many best practices have emerged from this time, which includes process modelling. Much effort is required to

implement such modern features, which is often lacking on the part of officials and legislators. For these cases, the European Union (EU) issues many directives for modernising different parts of the state. For example, in 2014, the European Union created directives that member states had to implement in their legislation within 2 years. These directives simplify and flexible' public procurement, with benefits for procurers and businesses such as simplifying procedures for procurers and preparing the ground for digitisation. The public procurement process assists procurers in the implementation of environmental policies, as well as policies governing social inclusion and innovation (European Commission, 2023). By Regulation (EU) 2019/1780, it has been established that from 25.10.2023, the TED form scheme will be replaced by a better, more modern, and more flexible alternative, namely the Eform. Eform is also structured according to XML, but it fulfils all the requirements of the Universal Business Language (UBL) standard (SIMAP group, 2023), which only encourages EU member states to be more digitalisation friendly.

The use of information technology in public procurement brings several advantages, such as considerable savings for all parties, simplified and shortened processes, reduced bureaucracy and administrative burden, increased transparency, more significant potential for innovation, new business opportunities by improving access for businesses, including small and medium-sized enterprises, to public procurement markets (European Commission, 2023).

In Slovakia, there are 2 platforms through which digital procurement can be carried out, namely the electronic marketplace and IS EVO (Information System of Electronic Public Procurement). The EU offers the TED (Tenders Electronic Daily) search system, where private parties, not only from the EU, can search for bids to carry out the subject of public procurement (Office of the Government of the Slovak Republic, 2023).

3 Methodology

We used a methodology commonly used in corporate practice to model the Slovak state process of PP carried out through the digital platform. First, we needed to find out what we wanted to model and what modelling technique we wanted to use to achieve the desired effect. From that point, we chose the modelling technique and the tool to model the process. In our case, we chose the BPMN modelling technique as it allowed us to break down and write the process in a sufficiently comprehensive way. Such a model can easily depict a global model, and we chose draw.io as the tool, which is freeware and provides all the necessary functions to model the process. The other parts of this methodology consist of the primary two parts, preparatory and diagram making. The preparatory part consists of (Process team, 2018): Planning the model creation, Identifying the process executors, and creating a list of all the activities/tasks that need to be modelled.

The part of the diagram creation consists of (Process team, 2018):

- Dividing the model into several imaginary parts. Definition of an initial node at the beginning of the whole model and placing it in the top left corner of the diagram as far as possible. The swim lane at the top of the diagram belongs to the actor that starts the process.
- Inserting the first activities. The question to ask is: When does the first implementer start the process? What are the first activities or tasks that need to be completed? This way, we gradually lay out the activities in all model parts.
- Chronological linking in the different parts of the activity model by using connector arrows and, if necessary, by placing connectors. The model will then begin to take shape. When we have connected this way all the activities in all the parts, we also connect the parts together.
- Integrating the symbols of decision-making, branching, and joining. There is also a possibility to make a few changes throughout the model so that all the decision and branching symbols make logical sense and best reflect all possible directions of process development.

- Insertion of additional symbols, such as document and data output. In a state process, this is how to record the necessary forms or data needed to further progress in the process or to record the output of data or forms.
- Adding annotations. When modelling a state process, it is crucial to declare a paragraph or law for each essential thing to which the activity, part of the process or the whole process is related. Annotations are helpful for this purpose.
- Checking that each activity makes it clear where to go next, that each branch is also unified, and that there is not a never-ending cycle somewhere. If all is well, the end nodes are added.

4 Results and Discussion

First, we planned the creation of a model, which consisted of defining what we would model and with what we would model it. We carried out this activity at the beginning of section three, deciding that we would create the model using the BPMN modelling technique in the draw.io web application. Once we had completed this phase, we needed to decide how we would model it. As an aid to this modelling task, we used, in addition to Act 343/2015 Coll., video tutorials from IS EVO, a chat-bot for quicker interpretation of the law, the TED platform and a two-hundred-page manual issued by the Public Procurement Office entitled Methodology for Contract Awarding (SIMAP group, 2023).

The most challenging part of the modelling, either in time or effort, was to analyse the process and the laws tied to it. We had to find out how many performers there were in the process. These are the procurer and the provider. The next step of the analysis was to understand the more significant parts of the process, as the process is entirely branching. So, we identified five major parts, plus the preparatory part, before the branching. We then constructed a verbally named "scenario" of the steps and how they are connected. Once all the analytical steps were done, we started modelling the first version of the PP process while strictly following the law. In this way, we gradually interconnected all the main parts in both swim lanes and outlined the process's beginning and primary end.

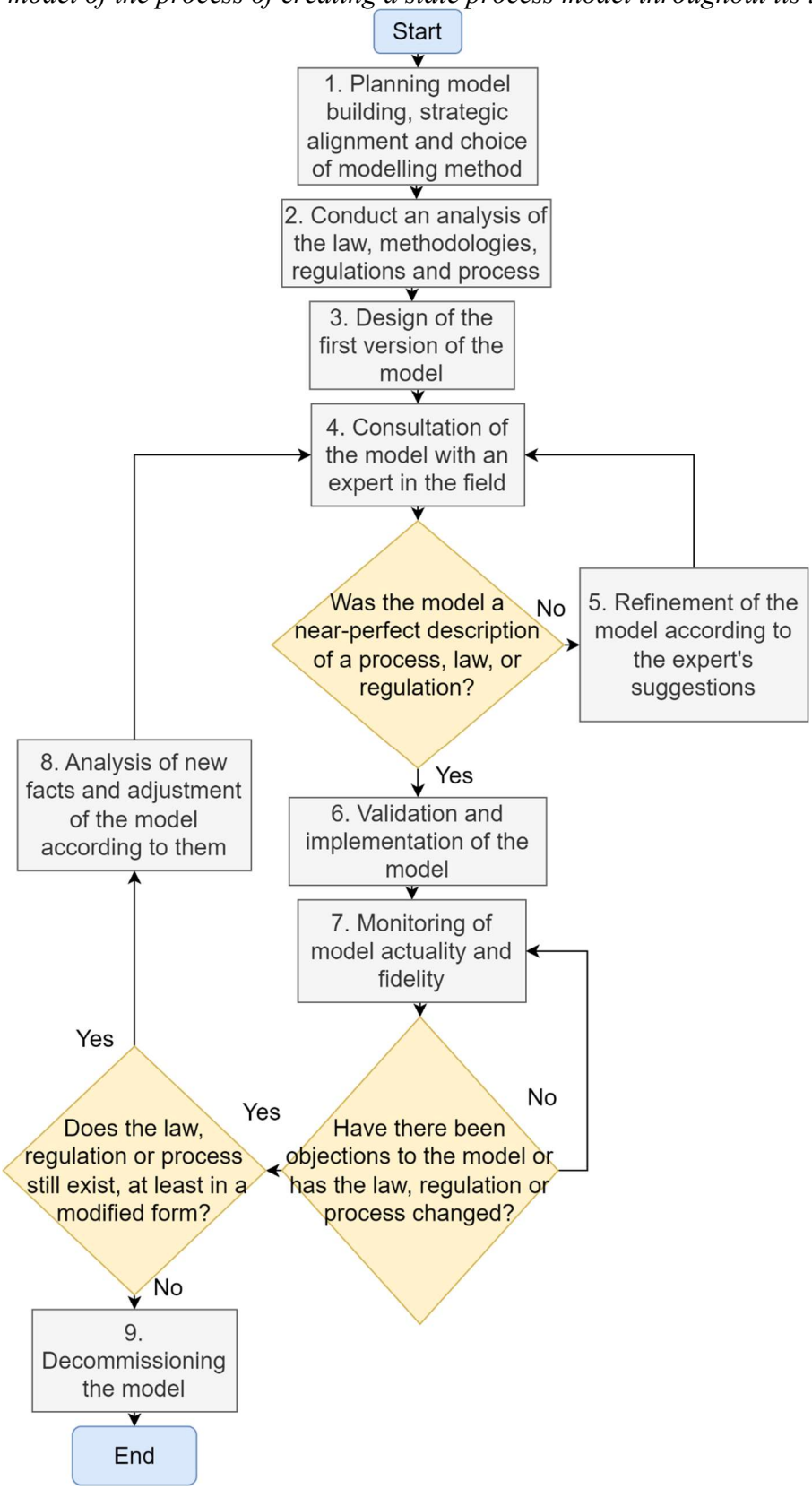
Once the modelling was completed, we refined the process with the help of a procurement expert. Our work on the model ended here as we obtained a refined model of the PP process. However, if the model was implemented and monitored, and if there were inconsistencies between the process and the law, such as a detected error or a change in the law or process, the life cycle would be completed by analysing and implementing the new features. The whole model of the state process model development procedure during its whole life cycle is shown in Fig. 1.

Fig. 2 shows the whole model of the state PP process. The green part of the model represents a small-scale contract, the red a low-value contract, the blue an under-limit contract, the purple an over-limit contract and the yellow a signing of the purchase contract. The model also colour-codes the platforms through which the procurement can be carried out. We also marked with red circles parts of processes, which we further depicted in this article from Fig. 3 to Fig. 7.

As we mentioned before, from Fig. 3 to Fig. 7, the process of conducting a low-value procurement is illustrated, both on the procurer's and provider's sides. Fig. 3 shows the initial analysis regarding the procurement, with the process branching off at the end according to the value limit of the procurement. Fig. 4 shows the procedure of the low-value sub-process on the procurer’s side, branching off internally according to the platform on which the procurement will be carried out, and there is a present end of the whole process, which is triggered only if the PP is carried on with low-value contract. Fig. 5 shows the low-value contract on the provider

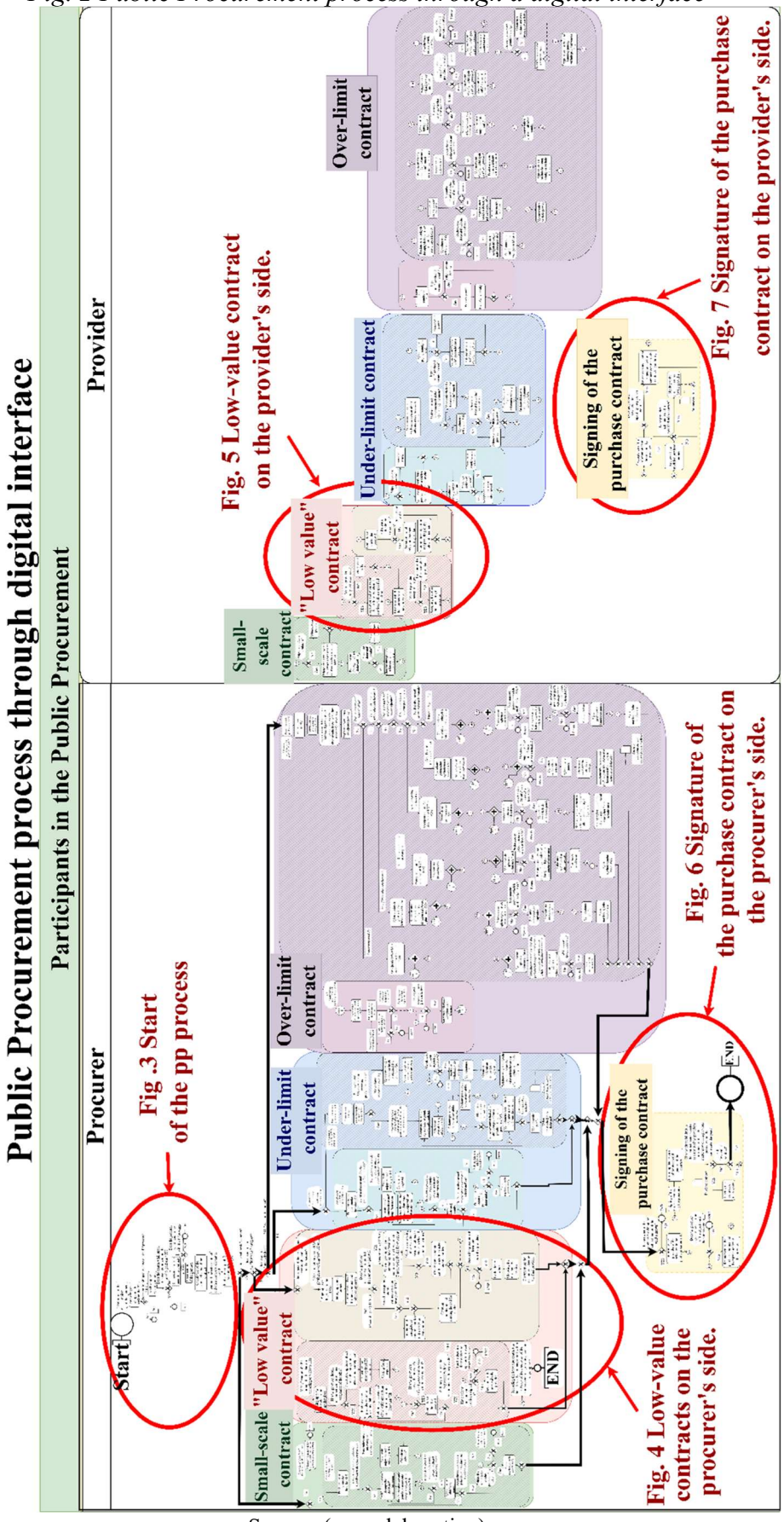
side, connected to the main process flow on the procurers' side with connectors, which are depicted with numbers in circles. Fig. 6 shows the signature of the contract on the procurer's side and Fig. 7 on the provider's side. For the low-value contract, the process ends up in Fig. 3.

Fig. 1 A model of the process of creating a state process model throughout its lifecycle



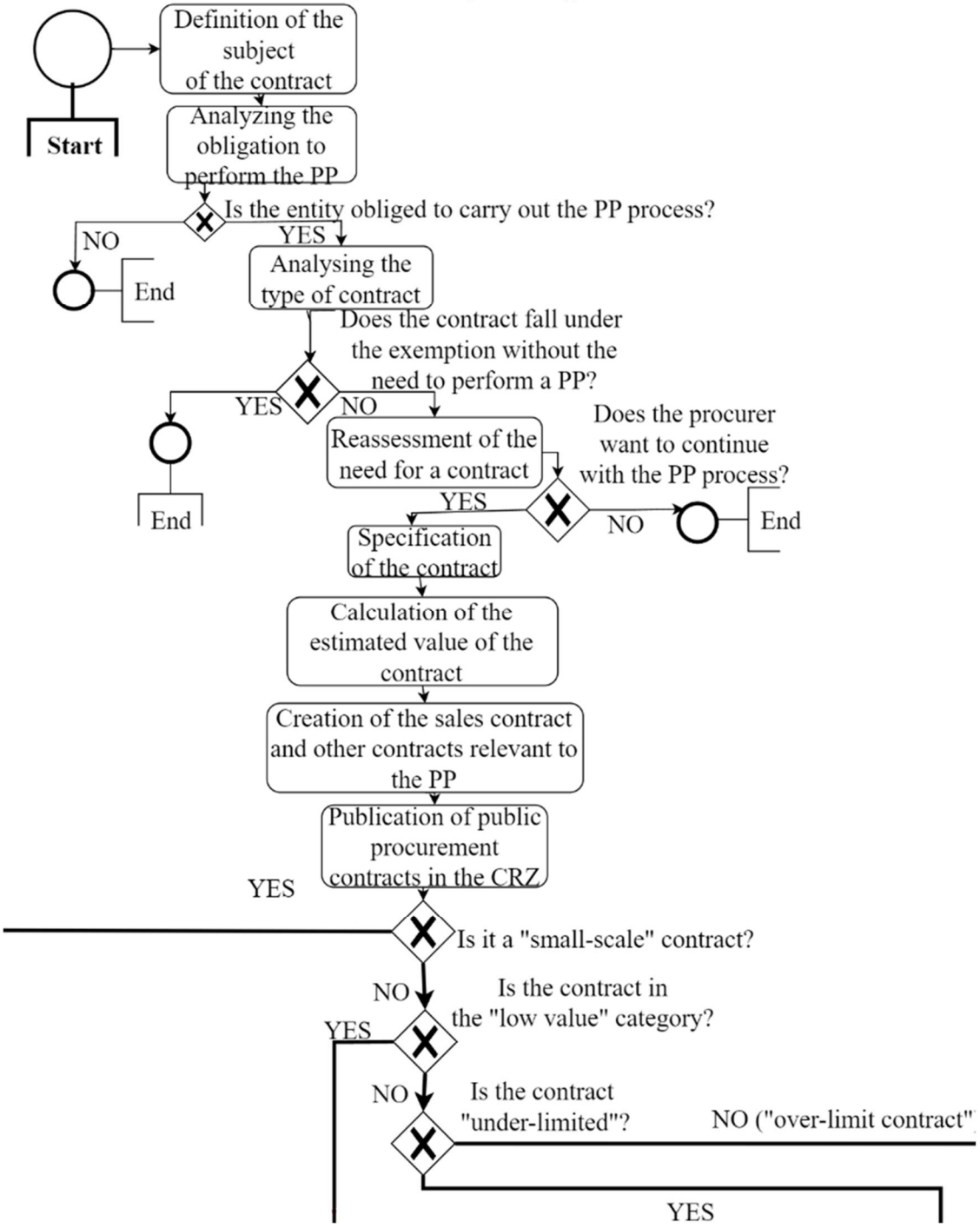
Source: (own elaboration)

Fig. 2 Public Procurement process through a digital interface



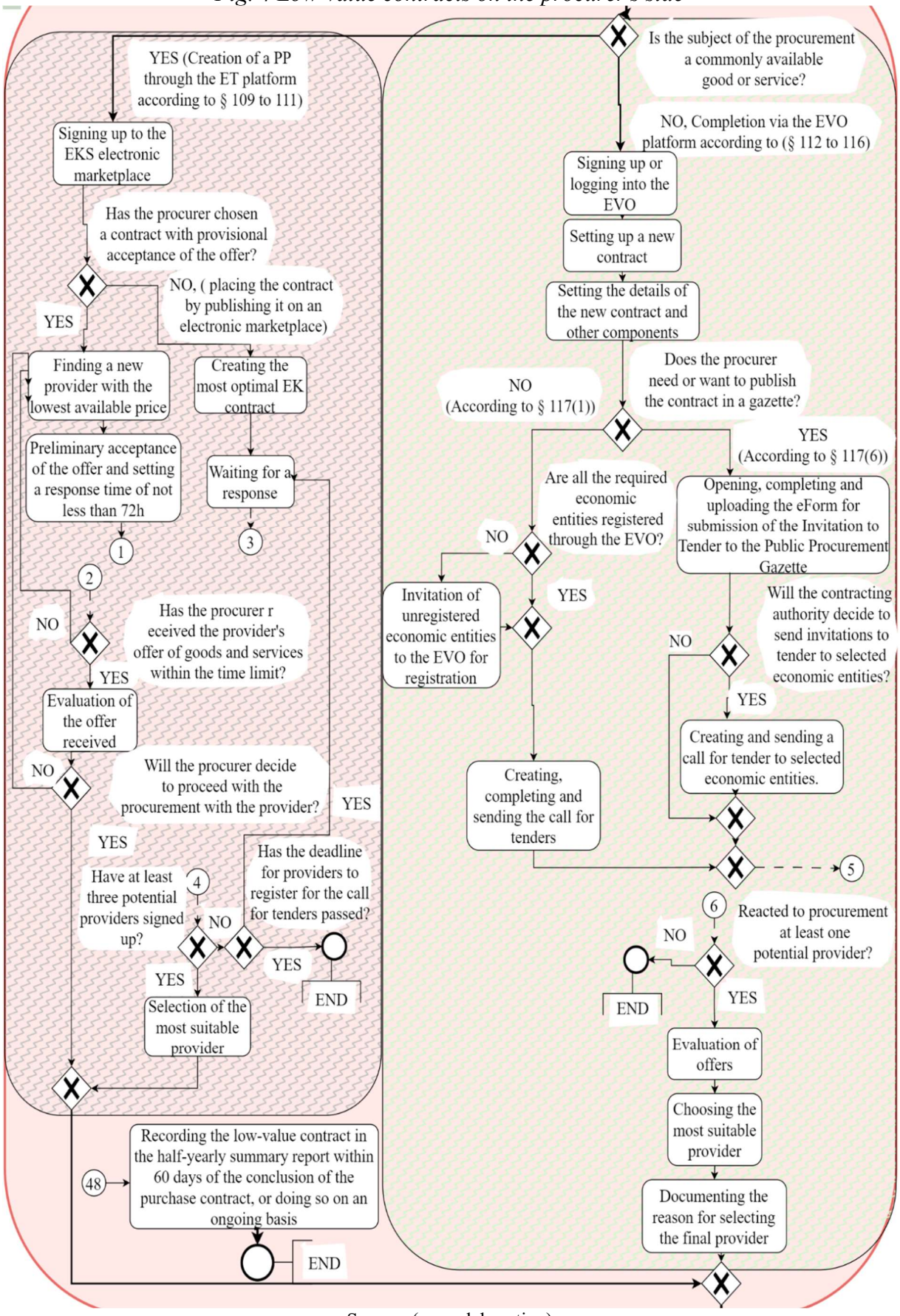
Source: (own elaboration)

Fig. 3 Start of the PP process



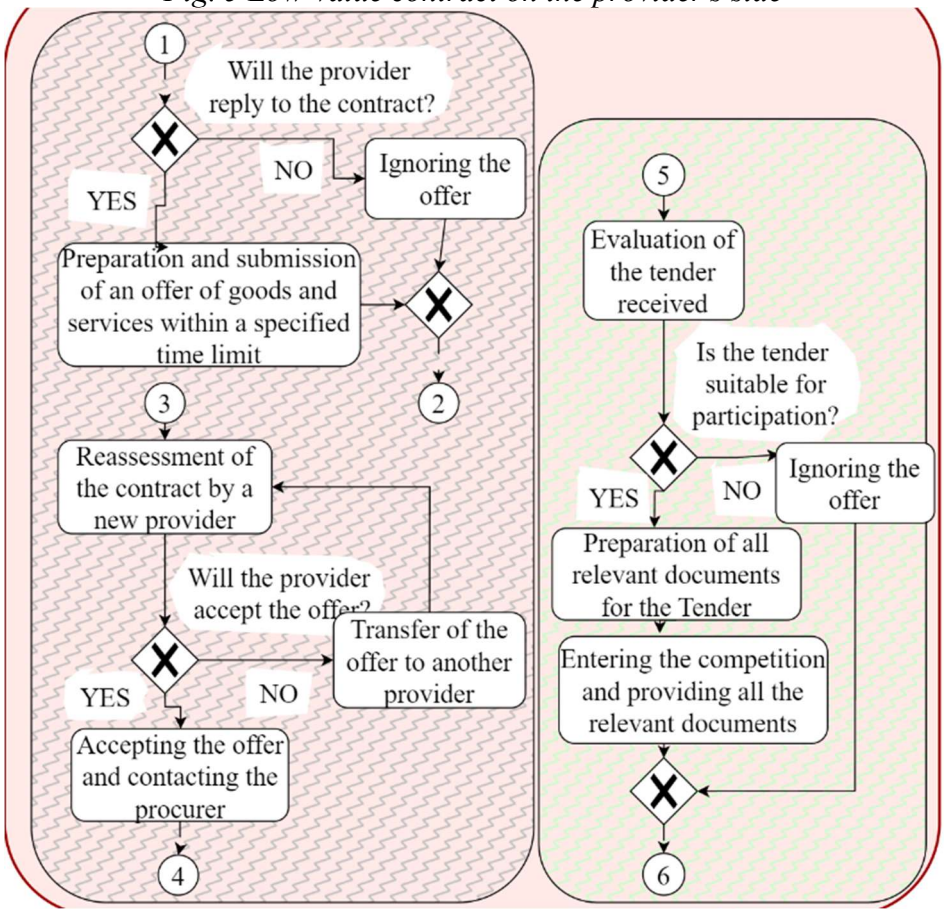
Source: (own elaboration)

Fig. 4 Low-value contracts on the procurer's side



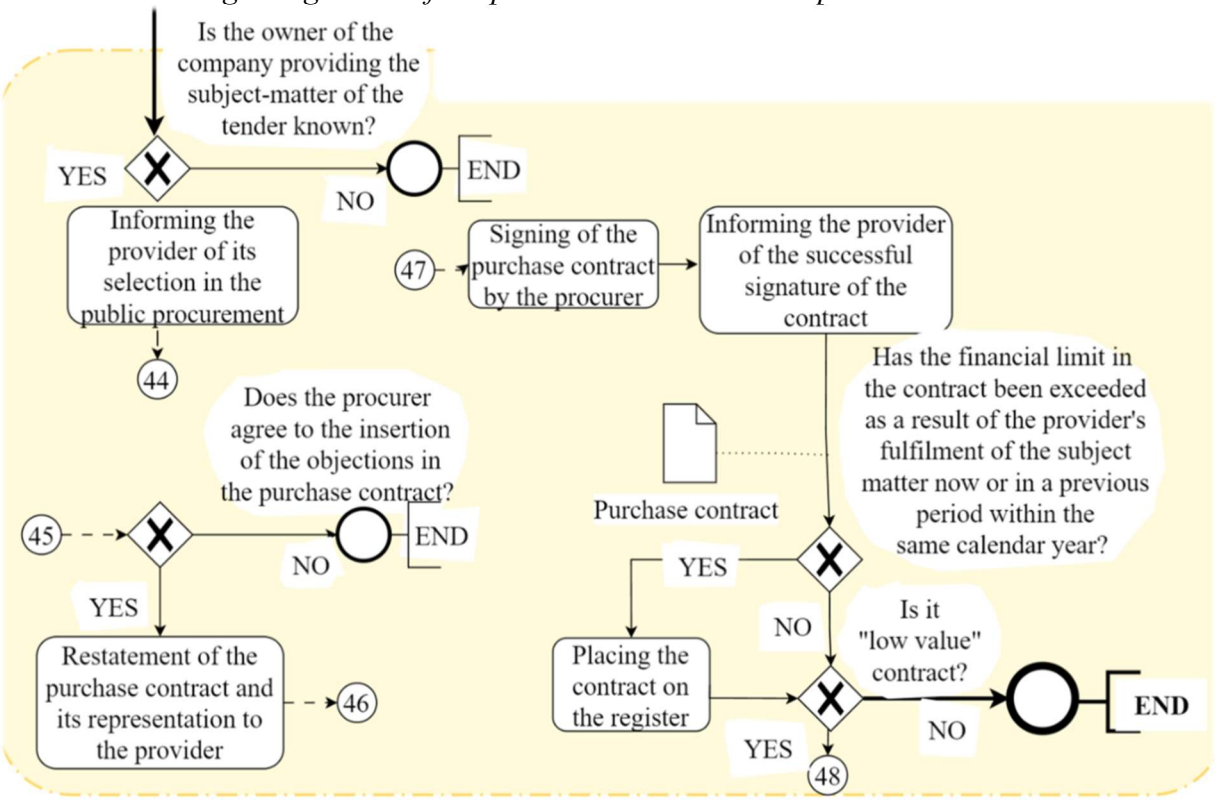
Source: (own elaboration)

Fig. 5 Low-value contract on the provider's side



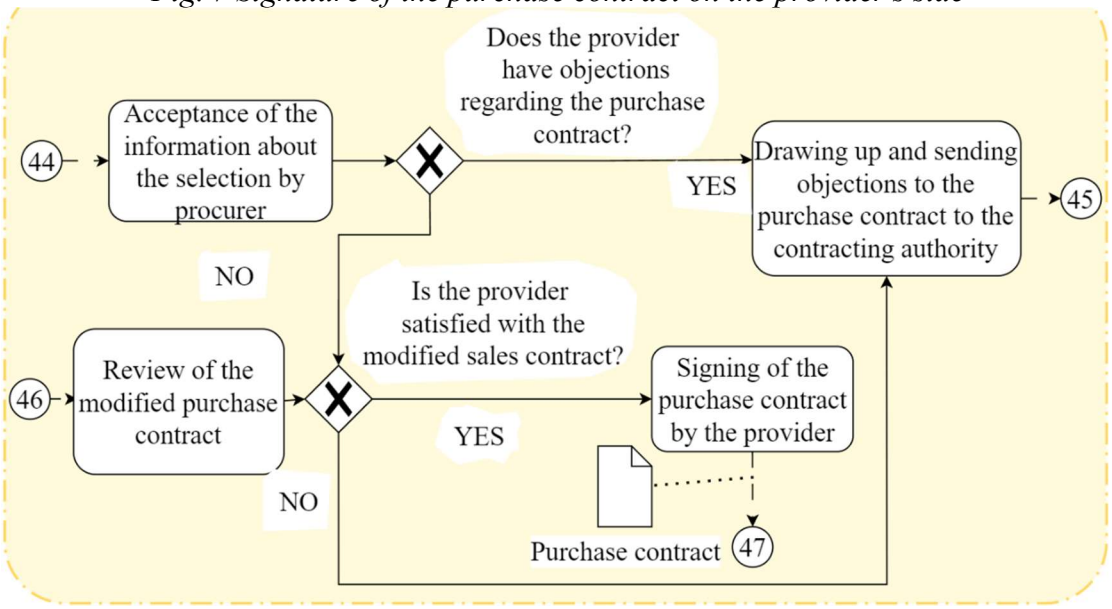
Source: (own elaboration)

Fig. 6 Signature of the purchase contract on the procurer's side



Source: (own elaboration)

Fig. 7 Signature of the purchase contract on the provider's side



Source: (own elaboration)

4.1 Discussion

In the specific case of modelling the PP process, two problems were encountered. The first visual problem with the process model is that the diagram can be, at first glance, chaotic to the average user, and thus, it can be difficult for them to navigate the diagram. We addressed this problem in several ways. The most visible solution applied in the model was to colour-code parts of the diagram according to which group of procurement processes are involved in the model and according to the platform through which the process will be carried out.

The second problem is a large amount of information flow between the main parts and between the process executors, which can make the diagram illegible. We solved this problem using connectors throughout the diagram, thus achieving higher clarity. We also placed these connectors at approximately similar heights in both swim lanes.

In general, two complications have also been found to arise when modelling any state process. The first widespread problem in modelling state processes is that the laws describing state processes do not have long persistence and need to be described as accurately and in as much detail as possible. Thus, constancy cannot be achieved in such a model, which needs to be modified every time the law changes. Part of the problem in the PP model has been that its laws often have value thresholds that are frequently changed, which are used to decide what style to use to continue the process. The solution was not to use decision-making based on the numerical value of the contract but to select the type of contract, as such recency and consistency should be achieved. Constantly changing models must also be actively modified, as noted in Fig. 1.

Another problem is that every step must reflect the law, and every step, sub-process and process must have a description defined in the law. A solution would be to modify BPMN for these needs, for example, having each step interleaved with a specific part of the law. This fact could also be implemented in the previously mentioned software to query information more efficiently from the state process model. In our model, we addressed this information part at least partially by defining the paragraph of the described law at the beginning of the branch and which section of the law would be followed next.

5 Conclusion

The first model, depicted in Fig. 1, our primary goal, was constructed and reviewed by combining best practices used in a business environment and knowledge gained while modelling established state processes. The model could be used as a guide on how to manage state process models through their whole lifecycle, which could be used not only through the digitalisation process of government but even in future after modernisation.

The second model, depicted from Fig. 2 to Fig. 7, related to the secondary goal, was made from dozens of minutes of video tutorials, hundreds of paragraphs in-laws, and hundreds of pages of manuals, and was refined by an expert in public procurement. The result is a model that can be printed on A2 or A1 paper. If the model was displayed in a digital version, allowing the user to interact with it and view only what they need, the diagram's content and the overall text could be reduced to a single A4 paper. From these facts, it is clear, and the expert confirms, that such a model can provide a better overview of the process and the steps. The process of PP was modelled manually to show the possibility of the model usage and a preview of the potential lifecycle. The model can be used to analyse modelled processes in practice, as a quick overview, and as educational material for private entities or students of the Public Procurement Act.

The sense of this work was to prepare for modern management of state processes, with the help of digitalisation, and thus the use of a unified information system, automation of state processes, the use of cloud services, the use of artificial intelligence or technologies based on IOT, or the use of quantum networks as a secure mediator of communication. Such technologies can simplify, clarify, speed up, and secure the fair functioning of the state, its processes, and laws.

Further research could be pursued by producing an IT (Information Technology) analysis on digital platforms and the procurement process, which could be used to optimise the digital environments already used for public procurement. At the same time, such IT analysis could later serve as a guide to unify all digital interfaces to different state processes into one centralised solution - a single massive state information system. The point of such an enormous state information system would be to increase the speed, efficiency, and reliability of state processes even more dramatically. Process models could be datamined and rendered with new IT tools, even automating the modelling process. This automation deserves deeper examination. Another possible research could also be in support of state project planning, where such a new adaptation of state process modelling would enable more accurate project planning, whether in the design of laws, projects, processes, or digital interfaces themselves for state purposes, as it would be easier to determine the expected periods for the execution of individual project tasks from the processes modelled in this way.

References

1. American Society of Mechanical Engineers. (1947 - 1952). *ASME standard: operation and flow process charts, 1947.* New York. https://catalog.hathitrust.org/Record/005735891
2. IBM Cloud Education. (2021, October 1). *What is business process modeling?* IBM Blog. https://www.ibm.com/cloud/blog/business-process-modeling
3. Erickson, J., & Siau, K. (1970, January 1). *Unified modeling language: The teen years and Growing Pains*. SpringerLink. https://link.springer.com/chapter/10.1007/978-3-642-39209-2_34

4. European Commision. (n.d.). *Legal rules and implementation*. Internal Market, Industry, Entrepreneurship and SMEs. https://single-market-economy.ec.europa.eu/single-market/public-procurement/legal-rules-and-implementation
5. Gilbreth, F. B., & Gilbreth, L. M. (1921). Process Charts. *The American Society of Mechanical Engineers*, 1–17. https://web.archive.org/web/20150509222833/https://engineering.purdue.edu/IE/GilbrethLibrary/gilbrethproject/processcharts.pdf
6. Jurík, P. (2018). *Informačné systémy v podnikovej praxi* (2.). Nové Zámky: Tlačiareň MERKUR, s. r. o. ISBN 978-80-970233-7-9.
7. Marsh, E. R. (1974). The harmonogram of Karol Adamiecki. *Academy of Management Proceedings*, *1974*(1), 32–32. https://doi.org/10.5465/ambpp.1974.17530521
8. Microsoft Corp. (n.d.). *Jednoduchá príručka tvorby diagramov a modelovania databáz UML*. Microsoft. https://www.microsoft.com/sk-sk/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling
9. OMG. (n.d.-b). *The history of BPMN*. BPMN. https://bpmn.gitbook.io/bpmn-guide/what-is-bpmn/the-history-of-bpmn
10. Process team. (2018, October 26). *UML tutorial: How to model any process or structure in your business: Process street: Checklist, Workflow and SOP software*. Process Street. https://www.process.st/uml-tutorial/
11. SIMAP group. (n.d.). *Standard forms for public procurement*. https://simap.ted.europa.eu/standard-forms-for-public-procurement
12. Weske, M. (2012). *Business Process Management*. Springer. ISBN 978-3-642-28615-5.
13. Thompson, N. & Platt, R. (2015). The Evolution of UML. https://www.researchgate.net/publication/309605753_The_Evolution_of_UML
14. Úrad vlády SR. (2023, July 10). Elektronická platforma. https://eplatforma.vlada.gov.sk/
15. Vargicová, P. (2021, April 13). Verejné obstarávanie v kocke - definícia, limity, fázy a postupy. https://www.podnikajte.sk/zakonne-povinnosti-podnikatela/verejne-obstaravanie-definicia-limity

Využitie dynamického programovania pri zostavovaní produktu cestovného poistenia
Using dynamic programming in making of travel insurance product

Martina Horváthová[1]

Abstrakt
Cieľom dynamickej optimalizácie je určiť súbor premenlivých časových profilov pre dynamické systémy, ktoré optimalizujú danú účelovú funkciu vzhľadom na špecifické obmedzenia. Jednou z metód využívanej pri riešení problému dynamickej optimalizácie je tzv. problém batohu. Takéto riešenie s použitím Bellmanovej rovnice poskytne výber, ktorý bude spĺňať stanovené obmedzenia, bude dosiahnutý koncový stav a zároveň získame optimálne riadenie pre zadané parametre. Ak by sme problém riešili pomocou tzv. metódy hrubej sily, spočívala by v prehľadávaní všetkých kombinácií daných položiek. Časová náročnosť je viazaná na počet všetkých riešení, ktorý môže byť pri rastúcej veľkosti daného problému naozaj veľký.

Kľúčové slová
Dynamické programovanie, optimalizačný problém, problém batohu 0-1, produkt cestovného poistenia

Abstract
The goal of dynamic optimization is to determine a set of time-varying profiles for dynamic systems that optimize a given objective function with respect to specific constraints. One of the methods used in solving the problem of dynamic optimization is the so-called knapsack problem. Such a solution using the Bellman equation will provide a choice that will satisfy the constraints, the end state will be reached, and at the same time we will obtain optimal control for the specified parameters. Otherwise solution would consist in searching all combinations of the given items. The time requirement is tied to the number of all solutions, which can be really large as the size of the given problem grows.

Key words
Dynamic programming, optimization problem, 0-1 knapsack problem, travel insurance product

JEL classification
C1

1 Úvod

Princíp *rozdeľuj a panuj* (divide-and-conquer) využívaný pri riešení mnohých problémov pracuje nasledovne: veľký problém rozdelíme na viaceré samostatné menšie podproblémy. Tieto podproblémy najskôr vyriešime a výsledky použijeme (skombinujeme) na vytvorenie riešenia celého veľkého problému. V dynamickom programovaní využijeme extrémny prístup k tomuto princípu: keď presne nevieme, ktoré podproblémy treba vyriešiť, jednoducho ich vyriešime všetky, výsledky si zapamätáme a opäť využijeme na doriešenie väčších problémov.

[1] Ing. Martina Horváthová, Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra matematiky a aktuárstva, Dolnozemská cesta 1, 852 35 Bratislava, martina.horvathova@euba.sk.

Pri aplikovaní dynamického programovanie musíme počítať s niektorými úskaliami. Po prvé, nie vždy je možné skombinovať riešenia menších problémov na vytvorenie riešenia väčšieho problému a po druhé, počet malých problémov, ktoré potrebujeme vyriešiť môže byť príliš veľký.

2 Problém batohu 0-1

Pre znázornenie typických procesov použitých pri riešení úloh práve dynamickým programovaním, použijeme jeden z najznámejších príkladov pre aplikáciu princípu optimality - **Problém batohu.** Ide o také riešenie aplikačného charakteru, kedy je potrebné vybrať z určitého súboru dopredu zadaný počet predmetov tak, aby bola celková úžitkovosť tohto výberu čo najvyššia – maximalizujeme zisk, výnos, a pod. Keďže vo väčšine historických publikácií sa predmety vkladali do batohu, nazývame takúto úlohu ako problém batohu.

Všeobecná formulácia problému:

- N druhov výrobku.
- Každý druh má určenú hmotnosť v_i, kde $i = 1, 2, \dots . N$.
- Každý druh má určenú výnosnosť c_i, kde $i = 1, 2, \dots N$.
- Zadaná je maximálna hmotnosť, ktorú batoh unesie.
- Úlohou je vybrať výrobky tak, aby bola ich výnosnosť čo najvyššia.

Matematická formulácia problému:

- x – hmotnostná kapacita batohu,
- x_i – počet naložených výrobkov i – teho druhu v batohu,
- Úlohou celočíselného programovania je nájsť maximum účelovej funkcie:

$$F(x_1, \dots, x_N) = c_1x_1 + c_2x_2 + \cdots + c_Nx_N = \sum_{i=1}^{N} c_ix_i \tag{1.1}$$

za podmienok

$$\begin{gathered} v_1x_1 + v_2x_2 + \cdots + v_Nx_N \leq x \\ x_i \geq 0,\ i = 1, 2, \dots . N \end{gathered} \tag{1.2}$$

Formulácia problému ako optimalizácia *N*-etapového rozhodovacieho procesu, pričom podľa vyššie spomenutého princípu dynamického programovania *rozdeľuj a panuj* rozdelíme proces do niekoľkých etáp. V každej etape bude rozhodnutie spočívať v určení množstva iba jedného výrobku. Začneme v prvej etape s rozhodnutím o počte *N*-tého výroku, pokračujeme v druhej s rozhodnutím o počte *N*-1 výrobku atď. Postupnosť $x_N, x_{N-1,\dots,}x_1$ vyhovuje vzťahom (1.2) zastupuje prípustnú stratégiu procesu $p^{(i+1)} = p^{(i)} - v_{N+1-i}x_{N+1-i}, i = 1, \dots, N;\ p^{(1)} = x$ stavovú transformáciu. Stavovou premennou je voľná váhová kapacita. Na základe princípu optimality dostaneme nasledovné rekurentné vzťahy:

$$f_N(x) = \max \quad [c_N x_N + f_{N-1}(x - v_N x_N] \qquad (1.3)$$
$$N \geq 2$$
$$x_N \in \left\{0,1, \dots, \left[\frac{x}{v_N}\right]\right\}$$

$f_1(x) = \left[\frac{x}{v_1}\right] c_1$, kde $\left[\frac{x}{v_i}\right]$ označuje najväčšie číslo nie väčšie ako $\frac{x}{v_i}$.

(Brázdilová – Šipošová, 2014)

Ako by sme problém riešili pomocou tzv. metódy hrubej sily, spočívala by v prehľadávaní všetkých kombinácií daných položiek, čo nám umožní vybrať si tú s maximálnym ziskom a hmotnosťou nepresahujúcou konštantu „*x*“. Časová náročnosť je viazaná na počet všetkých riešení, ktorý môže byť pri rastúcej veľkosti daného problému naozaj veľký. Tento prístup je preto využívaný ako dolná hranica hodnotenia výkonnosti iných algoritmov, pod ktorú by nemali klesnúť.

Ak by sme chceli vyskúšať všetky kombinácie, algoritmus by vyzeral takto:

pre *každú položku „i“*

vytvorte ***nový výber****, ktorý obsahuje položku „i“,* ***ak*** *celková hmotnosť nepresahuje kapacitu „x“,*

rekurzívne spracovať zvyšné položky,

vytvorte ***novú*** *množinu bez položky 'i' a rekurzívne spracujte zvyšné položky*

vrátiť *súbor z vyššie uvedených dvoch súborov s vyšším ziskom* (Bellman, 2003).

Postup riešenia pomocou dynamického programovania:

Keďže ide o problém s batohom 0–1, položku môžeme do nášho batohu zahrnúť alebo vylúčiť, avšak nemôžme zahrnúť iba jej *zlomok alebo ju zahrnúť viackrát*.

Krok č. 1:

Najprv vytvoríme 2-rozmerné pole (t. j. tabuľku) o rozmere *n* + 1riadkov a *w* + 1stĺpcov. Číslo riadku *i* predstavuje množinu všetkých položiek z riadkov *i*-1 . Predpokladajme teda, že záznam v riadku *i* , stĺpci *j* predstavuje maximálnu hodnotu, ktorú možno získať s položkami 1, 2, 3 … *i* , v batohu, do ktorého sa zmestí *j* jednotiek hmotnosti.

Krok č. 2:

Niektoré položky našej tabuľky môžeme začať vypĺňať okamžite - základné prípady, pre ktoré je riešenie triviálne. Napríklad v riadku 0, keď nemáme žiadne položky na výber, musí byť maximálna hodnota, ktorú možno uložiť v akomkoľvek batohu rovná nule. Podobne v stĺpci 0 pre batoh, ktorý môže obsahovať 0 jednotiek hmotnosti, maximálna hodnota, ktorá je v ňom možná uložiť sa rovná nule. (Za predpokladu, že neexistujú žiadne nehmotné, hodnotné predmety.)

Krok č. 3:

V nasledovnom kroku začneme vypĺňať polia tabuľky princípom dynamického programovania. Ako pri všetkých riešeniach dynamického programovania, v každom kroku využijeme naše riešenia predchádzajúcich čiastkových problémov. Pripomeňme, že v riadku *i* a stĺpci *j* riešime čiastkový problém pozostávajúci z položiek 1, 2, 3 … *i* s batohom s kapacitou *j*. V riešení tohto bodu máme 2 možnosti: môžeme položku *i* zahrnúť alebo nie. Pre rozhodnutie musíme porovnať maximálnu hodnotu, ktorú môžeme získať s a bez položky *i* . Maximálnu hodnotu, ktorú môžeme získať *bez* položky *i,* nájdeme v riadku *i*-1 ,

stĺpci j . Táto časť je ľahká. Zdôvodnenie je jednoduché: akákoľvek maximálna hodnota, ktorú môžeme získať s položkami 1, 2, 3 ... i musí byť samozrejme rovnaká maximálna hodnota, akú môžeme získať s položkami 1, 2, 3 ... i - 1 , *ak sa rozhodneme nezahrnúť položku* i . Na výpočet maximálnej hodnoty, ktorú môžeme získať *s* položkou *i*, najprv musíme porovnať hmotnosť položky *i* s nosnosťou batohu. Samozrejme, ak položka *i* váži viac, ako je batoh schopný uniesť, nemôžeme ju započítať, preto nemá zmysel vykonávať výpočet. V takom prípade je riešením tohto problému maximálna hodnota, ktorú môžeme získať bez položky *i* (t. j. hodnota v riadku vyššie, v rovnakom stĺpci). Predpokladajme však, že položka *i* váži menej ako celková hmotnostná kapacita batohu. Máme teda možnosť ju zahrnúť. Rozhodneme sa tak iba v prípade, ak potenciálne zvýši maximálnu dosiahnuteľnú hodnotu. Maximálna dosiahnuteľná hodnota zahrnutím položky *i* je teda rovná súčtu hodnoty samotnej položky *i* a maximálnej hodnoty, ktorú možno získať so zvyšnou kapacitou batohu. Keďže našou úlohou je dosiahnuť, čo najvyšší úžitok, je zrejmé, že chceme využiť kapacitu nášho batohu naplno a teda nenechať v batohu žiadnu voľnú hmotnostnú kapacitu. Preto v riadku *i* a stĺpci *j* (ktorý predstavuje maximálnu hodnotu, ktorú môžeme získať) **by sme vybrali buď maximálnu hodnotu, ktorú môžeme získať *bez* položky *i*** alebo **maximálnu hodnotu, ktorú môžeme získať *s* položkou *i*, *podľa toho, ktorá hodnota je väčšia.***

Krok č. 4:

Po naplnení tabuľky sa konečné riešenie nachádza v poslednom riadku v poslednom stĺpci, ktorý predstavuje maximálnu hodnotu, ktorú je možné získať so *všetkými zvolenými položkami* za predpokladu úplného vyčerpania *kapacity* batohu. (Bellman, 2003)

3 Aplikácia problému batohu na zostavenie produktu cestovného poistenia

V nasledovnej aplikačnej ukážke budeme zostavovať balíček cestovného poistenia s rôznymi pripoisteniami, ktorý bude spĺňať kritériá klienta. Jednotlivým zložkám poistenia budú pridelené číselné hodnoty 1-10, ktoré prezentujú veľkosť preferencie danej položky klientmi. V praxi by následná ukážka mohla byť využitá poisťovňou, ktorá by z informácii o už zakúpených modeláciách cestovných poistení vedela vytvoriť štatistiky preferencie jednotlivých zložiek tohto typu poistenia a následne by vedela vytvoriť balíček cestovného poistenia, ktorý by bol schopný uspokojiť potreby určitej skupiny klientov. Klient by si v tomto prípade nezakúpil základný balík, ku ktorému by pridával rôzne pripoistenia, pretože by bol v ponuke jeden či viaceré balíky, ktoré by boli v súlade s pomerom cena-rozsah-preferencie. Je vysoká pravdepodobnosť, že tieto balíky budú namierené na portfólio klientov lepšie, keďže budú vychádzať z historických údajov o kúpe cestovných poistení. Predpokladáme, že tvoríme balíček pre náročnejšieho klienta a preto zvolený finančný limit bude 10€/deň. Ďalej predpokladajme, že uvažujeme so skupinou klientov vo veku 18-59 rokov, poistenie je jednorazové a cieľová destinácia je v Európe. Pre danú ukážku sme použili dynamickú optimalizáciu pomocou princípu problému batohu 0-1. Pre nasledovnú kalkuláciu budeme používať softvér LINGO verzia 19.0. Tento softvér môžeme opísať, ako jednoduchý nástroj na rýchly a pomerne prehľadný výpočet rôznych optimalizačných problémov. Aplikačná ukážka zahŕňa 13 rozhodovacích premenných, ktoré si v skratke popíšeme:

- *Poistenie liečebných nákladov* – transporty, doprava sprevádzajúcej osoby, zásah horskej služby, návrat po skončení nariadenej preventívnej karantény v súvislosti s ochorením COVID-19...
- *Poistenie asistenčných služieb* – turistické a lekárske informácie, technická podpora v núdzi, tlmočenie a preklady, poistenie právnej asistencie, lifestyle asistencia.
- *Poistenie zodpovednosti* – zodpovednosť za škodu – zdravie, majetok následné finančné škody, spoluúčasť pri škodách.

- *Úrazové poistenie* – smrť úrazom, trvalé následky.
- *Poistenie batožiny* – strata osobných dokladov, elektronika, športové vybavenie, poistenie obchodného vybavenia.
- *Poistenie cestovania lietadlom* – poistenie omeškania batožiny, poistenie oneskorenia a zrušenia letu (o viac ako 6 hodín).
- *Poistenie domácich maznáčikov* – poistenie veterinárnej starostlivosti, doprava zvieraťa k veterinárovi, rozšírenie krytia z poistenia zodpovednosti pre škody spôsobené zvieraťom.
- *Rizikové športy* – rekreačné športy sú poistené už v základnom variante cestovného poistenia. Pripoistenie rizikových športov sa využíva pri vykonávaní profesionálnych športov či v prípade vykonávania bežných športov výkonnostne.
- *Poistenie práce a štúdia* – odporúča sa všetkým, ktorí idú do zahraničia za prácou alebo za štúdiom.
- *Drink povolený* – povolené užitie alkoholu, napr. na lyžiarskej zjazdovke až do 0,8 promile a následné krytie škôd.
- *Autoasistencia* – pripoistenie je vhodné pre všetkých, ktorí idú na dovolenku autom. Zaistí opravu na mieste alebo odtiahnutie do servisu, následnú opravu vozidla a v prípade vážnej poruchy prevoz auta a posádky späť domov.
- *Požičanie auta bez starostí* – požičanie auta, motorky alebo skútra. V prípade nehody vozidla pokryje spoluúčasť, ktorú bude chcieť požičovňa.
- *Chronické ochorenie* – rozširuje poistenie liečebných výdavkov aj na chronické ochorenia. Pokrýva úhradu nákladov spojených s poskytnutím akútnej a neodkladnej starostlivosti.

Tab. č. 1: Zoznam navrhovaných položiek

Č.	Položka (Variables)	Cena	Preferencia (1-10)
1	Poistenie liečebných nákladov	5	10
2	Poistenie asistenčných služieb	1	10
3	Poistenie zodpovednosti	3	8
4	Úrazové poistenie	2	9
5	Poistenie batožiny	3,5	8
6	Poistenie cestovania lietadlom	2	6
7	Poistenie domácich maznáčikov	1	2
8	Rizikové športy	3	5
9	Poistenie práce a štúdia	1	3
10	Drink povolený	2	8
11	Auto asistencia	1	2
12	Požičanie auta	1	4
13	Chronické ochorenie	3,5	3

Zdroj: Vlastné spracovanie

Preferencie klientov na zahrnutie danej zložky do balíčku poistenie v našom prípade vychádzajú zo vzorky 50 náhodných zakúpených cestovných poistení, ktoré spĺňajú vyššie uvedené podmienky.

V našom prípade rozoberieme dva optimalizačné problémy:

- Model A – maximalizácia počtu položiek a splnenie peňažného limitu,
- Model B – maximalizácia preferencií zákazníka a splnenie peňažného limitu.

Model A

Účelovú funkciu v Modeli A zapíšeme:

$$MaxZ = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13}$$

Účelová funkcia podlieha dvom obmedzeniam:

a) Maximálny peňažný limit je 10€/deň

$$5x_1 + 1x_2 + 3x_3 + 2x_4 + 3{,}5x_5 + 2x_6 + 1x_7 + 3x_8 + 1x_9 + 2x_{10} + 1x_{11} + 1x_{12} + 3{,}5x_{13} \leq 10$$

b) Garancia výberu prvých dvoch položiek – liečebné náklady a asistenčné služby

$$x_1 + x_2 = 2$$

Tab. č. 2: Výsledky Modelu A

Č.	Premenná	Položka	Preferencie	Cena
1	*x*1	Poistenie liečebných nákladov	10	5
2	*x*2	Poistenie asistenčných služieb	10	1
3	*x*7	Poistenie dom. maznáčikov	2	1
4	*x*9	Poistenie práce a štúdia	3	1
5	*x*11	Auto asistencia	2	1
6	*x*12	Požičanie auta	4	1
		TOTAL	**31**	**10 €**

Zdroj: Vlastné spracovanie

Optimálne riešenie je v tab. č. 2. Výsledok ukazuje, že do balíčku cestovného poistenia prešlo 6 z 13 položiek. Celková cena takéhoto balíčku je 10€/deň a preferencie zákazníkov boli uspokojené na iba na 39,74%. Sedem položiek sa do balíčku vôbec nedostalo.

Model B

Účelovú funkciu v Modeli B sme upravili, ale obmedzenia zostávajú nezmenené. Cieľovú funkciu je možné zapísať nasledovne:

$$MaxZ = 10x_1 + 10x_2 + 8x_3 + 9x_4 + 8x_5 + 6x_6 + 2x_7 + 5x_8 + 3x_9 + 8x_{10} + 2x_{11} + 4x_{12} + 3x_{13}$$

Tab. č. 3: Výsledky Modelu B

Č.	Premenná	Položka	Preferencie	Cena
1	*x*1	Poistenie liečebných nákladov	10	5
2	*x*2	Poistenie asistenčných služieb	10	1
3	*x*4	Úrazové poistenie	9	2
4	*x*10	Drink povolený	8	2
		TOTAL	**37**	**10€**

Zdroj: Vlastné spracovanie

Optimálne riešenie pri modeli B je zobrazené v tab. č. 3. Výsledok ukazuje, že do výsledného balíčku cestového poistenia prešli 4 z 13 položiek. Celková cena takéhoto balíčku je 10€/deň a preferencie zákazníkov boli uspokojené na 47,44%. Deväť položiek sa do balíčku vôbec nedostalo.

Tab. č. 4: Porovnanie výsledkov oboch modelov

Výsledky	Model A	Model B
Celková cena	10	10
Dodržanie cenového limitu (%)	100	100
Počet vybraných položiek	6	4
Uspokojenie preferencií (%)	39,74	47,44%

Zdroj: Vlastné spracovanie

4 Záver

Tabuľka č. 4 opisuje porovnanie medzi dvomi metódami výberu položiek. Praktický prístup k riešeniu daného optimalizačného problému je Model B, ktorý zahŕňa dve nevyhnutné položky a ostatné položky maximalizujú preferencie kupujúcich. Percento uspokojenia preferencií klientov je vyššie ako v modeli A. Čo sa týka cien, balíčky majú rovnakú cenu, avšak balíček podľa modelu B má o 2 položky menej. Tento balíček je však rozhodne lepšie zameraný na potreby klienta. Avšak náklady na pridanie ďalších položiek by už boli vysoké, keďže s danou cenou sa v balíčku nachádzajú iba 4 vybrané položky z 13 položiek.

Príspevok bol spracovaný v rámci riešenia grantovej úlohy VEGA1/0096/23: Vybrané metódy riadenia rizík pri implementácii parciálnych interných modelov pre stanovenie kapitálovej požiadavky pre solventnosť.

Literatúra

1. Bednarczuk, E. M., Miroforidis, J. & Pyzel, P. A. (2018). A multi-criteria approach to approximate solution of multiplechoice knapsack problem. Computational Optimization and Applications, 70(2018), 889–910. doi:org/ 10.1007/ s10589-018- 9988-z.
2. Bellman, R. (2003). Dynamic Programming. *Dover Publications INC.,* ISBN 978-04-864-2809-3.
3. Brázdilová, V. – Šipošová, A. (2014). Dynamické programovanie a multikriteriálna optimalizácia v návrhu konštrukčných prvkov. *Bratislava: STU v Bratislave SvF.*
4. Haddadh, A. K. – Yakhchali, S. H. – Jalili bal, Z. (2016). MCDM Techniques and Knapsack Approach for Project Selection Problem: A Case Study. International Journal of Humanities and Management Sciences, 4(4), 397-400.
5. Laščiak, A. a kol. (1985). Dynamické modely. *Bratislava: ALFA – vydavateľstvo tecnickej a ekonomickej literatúry.*
6. Mathew, M. (2023). 0/1 Knapsack Problem Fix using Dynamic Programming Example. Dostupné na: https://www.guru99.com/knapsack-problem-dynamic-programming.html.
7. Turhan, N. (2011). Deterministic and Stochastic Bellman's Optimality Principles on Isolated Time Domains and Their Applications in Finance. *Masters Theses & Specialist Projects. Paper 1045.* Dostupné na: http://digitalcommons.wku.edu/theses/1045.

8. Yang, Y. – Boland, N. – Savelsbergh, M. (2019). Multi-variable branching: A case study with 0-1 knapsack problems. Optimization. Dostupné na: https://www.researchgate.net/profile/ Yu-Yang-149/publication/340964301_Multi-Variable_Branching_A_Case_Study_with_0-1_Knapsack_Problems/ links/5ea7a95b45851553fab5ead7/MultiVariable-Branching-A-Case-Study-with-0-Knapsack-Problems.pdf.

Metóda najmenších štvorcov pre slabé inštrumentálne premenné
Method of least squares for weak instrumental variables

Peter Knížat[1]

Abstrakt
Štandardná metóda najmenších štvorcov je bežným nástrojom pri odhade parametrov endogénnych regresných premenných v ekonometrických modeloch. V prípade ak sú regresné premenné exogénne je ich potrebné nahradiť inštrumentálnymi premennými. V tomto prípade je štandardná metóda najmenších štvorcov modifikovaná ako dvojstupňová z dôvodu nenulovej kovariancie medzi náhodnými chybami redukovaného a štrukturálneho modelu. K nekonzistentnosti odhadovaných parametrov môže ďalej dochádzať ak sú inštrumentálne premenné korelované s náhodnými chybami štrukturálneho modelu. Modifikácia dvojstupňovej metódy najmenších štvorcov pomocou tzv. jackknife metódy túto koreláciu významne redukuje. Cieľom tohto príspevku je ukázať aplikáciu týchto troch metód odhadu parametrov na syntetických údajoch vygenerovaných Monte Carlo simuláciami.

Kľúčové slová
ekonometrický model, slabé inštrumentálne premenné, dvojstupňová metóda najmenších štvorcov, jackknife metóda

Abstract
The ordinary least squares method is a common tool in estimating parameters of endogenous regression variables in econometric models. If the regression variables are exogenous, they need to be replaced by instrumental variables. In this case, the ordinary least squares method is modified as a two-stage because of the non-zero covariance between random errors of the reduced and structural model. The inconsistency of estimated parameters can further occur if instrumental variables are correlated with random errors of the structural model. A modification of the two-stage method of least squares using the so-called jackknife method significantly reduces this correlation. The aim of this paper is to show the application of these three methods of estimating parameters on synthetic data generated by Monte Carlo simulations.

Key words
Econometric model, Weak instrumental variables, Two-stage least squares method, Jackknife method

JEL classification
C13, C36

1 Úvod

V tomto článku sa venujeme dvom modifikáciám štandardnej metóde najmenších štvorcov, ktoré sa využívajú pre odhad parametrov v ekonometrických modeloch. V klasickom ekonometrickom modeli, s endogénnymi nezávislými premennými je štandardná metóda najmenších štvorcov efektívna v dosiahnutí cieľa minimalizácie náhodných chýb modelu, vrátane dodržania všetkých predpokladov modelu, napr. nezávislosť a normálne rozdelenie náhodných chýb modelu s nulovou strednou hodnotou a konštantným rozptylom. Ak sú

[1] Peter Knížat, MSc, Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra operačného výskumu a ekonometrie, Dolnozemská cesta 1, 852 35 Bratislava, peter.knizat@euba.sk.

regresné premenné exogénne je ich potrebné nahradiť inštrumentálnymi premennými, ktoré sa zadefinujú v tzv. redukovanom modeli. V tomto prípade, pri odhade parametrov modelu je metóda najmenších štvorcov modifikovaná ako dvojstupňová z dôvodu nenulovej kovariancie medzi náhodnými chybami redukovaného a štrukturálneho modelu. Avšak ak sú inštrumentálne premenné korelované s náhodnými chybami štrukturálneho modelu môže dochádzať k vychýleniu odhadnutých parametrov, t. j. odhadované parametre sú nekonzistentné. Pre eliminovanie tejto korelácie bola navrhnutá modifikácia dvojstupňovej metódy najmenších štvorcov s použitím tzv. jackknife metódy.

Prvé použitie inštrumentálnych premenných v ekonometrických modeloch sa objavilo v práci z roku 1928 od Philipa G. Wrighta (Wooldridge, 2002). V súčasnosti je metóda inštrumentálnych premenných štandardná nástroj v ekonometrických modeloch. Definícia štrukturálneho a redukovaného modelu a detailné odvodenie dvojstupňovej metódy minimalizácie najmenších štvorcov je popísané v knihe od (Wooldridge, 2000).

V súčasnom výskume sa zistilo, že ak sú inštrumentálne premenné slabé dochádza k vychýleniu odhadovaných parametrov pri použití dvojstupňovej metódy najmenších štvorcov, viď napr. (Hahn a kol., 2002), (Hahn a kol., 2005). Presná definícia a problém, ktorý spôsobuje slabé inštrumentálne premenné nie je jednoznačne definovaná a bude detailnejšie predostretá v nasledujúcej časti tohto článku. Jedna z možných indikácií slabých inštrumentálnych premenných je, že odhadnutý redukovaný model má malý koeficient determinácie a hlavným dôvodom je korelácia medzi inštrumentálnymi premennými a náhodnými chybami štrukturálneho modelu (Hahn a kol., 2005). Tematika odhadu parametrov modifikovanou verziou dvojstupňovej metódy najmenších štvorcov pomocou tzv. jackknife metódy, ktorá dokáže značne eliminovať túto koreláciu, je obsiahle skúmaná vo vedeckých článkoch (Angrist a kol., 1995) a (Angrist a kol., 1999).

Cieľom tohto článku je predstaviť teoretický rámec pre dvojstupňový odhad najmenších štvorcov a jeho modifikovanú verziu pomocou jackknife metódy, ktorá eliminuje koreláciu medzi inštrumentálnymi premennými a náhodnými chybami štrukturálneho modelu. V prípadovej štúdii použijeme syntetické údaje, vygenerované Monte Carlo simuláciami, pre definovanie ekonometrického modelu a následne vyhodnotíme parametre modelu, ktoré sú odhadované pomocou horeuvedených metód. Aplikácia je v softvéri R a všetky kódy sú dostupné v tomto článku.

2 Teoretické východiská

V tejto časti predstavíme teoretický rámec pre ekonometrický lineárny model, definíciu metódy inštrumentálnej premennej a výpočet dvojstupňovým odhadom minimalizácie najmenších štvorcov modelu.

Štandardne zo štatistického pozorovania získame maticu reálnych čísel vo forme $\mathbf{X} = \{x_{ij}\}$, kde x_{ij} sú individuálne náhodné nezávislé premenné a $j = 1, \dots, p$ reprezentuje ich počet, a korešpondujúci vektor reálnych čísel $\mathbf{y} = y_i$, ktorý reprezentuje závislú premennú, obidve pozorované pre vzorky $i = 1, \dots, n$. Ekonometrický regresný model je možné definovať nasledovne (Wooldridge, 2000):

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{1}$$

kde $\boldsymbol{\varepsilon} = \varepsilon_i$, pre vzorky $i = 1, \dots, n$, je vektor náhodných chýb, pri ktorom predpokladáme, že má normálne rozdelenie so strednou hodnotou 0 a konštantným rozptylom σ^2. $\boldsymbol{\beta} = \beta_j$ sú neznáme parametre pre $j = 1, \dots, p$ nezávislé premenné vrátane prieniku β_0, pre ktorého odhad matica $\mathbf{X}$ obsahuje jednotkový prvý stĺpec. Parametre modelu $\boldsymbol{\beta}$ je možné odhadnúť štandardnou metódou minimalizácie najmenších štvorcov. V ekonometrickom modeli tiež

predpokladáme, že kovariancia medzi $\boldsymbol{\varepsilon}$ a $\mathbf{X}$ je nulová. Inak povedané, nezávislé premenné sú definované ako exogénne premenné. Použitím štandardnej metódy najmenších štvorcov pre odhad parametrov v rovnici (1) dostaneme nasledovné odhadnuté parametre (Wooldridge, 2000):

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{y}) \tag{2}$$

kde $\widehat{\boldsymbol{\beta}}$ je vektor odhadnutých parametrov pre nezávislé premenné $j = 1, \dots, p$, vrátane priesečníku $\hat{\beta}_0$, a T je označenie pre transponovanie matice. Ak nezávislé premenné $\mathbf{X}$ korelujú s náhodnými chybami modelu $\boldsymbol{\varepsilon}$, metóda výpočtu štandardným odhadom najmenších štvorcov vedie k nekonzistentným odhadovaným parametrom ekonometrického regresného modelu (Wooldridge, 2002). V tomto prípade sú nezávislé premenné $\mathbf{X}$ definované ako endogénne. Endogenita nezávislých premenných v regresnom modeli môže nastať ak:

- zmeny v závislej premennej menia hodnotu nezávislých premenných,
- sú vynechané premenné, ktoré ovplyvňujú závislú aj nezávislé premenné, prípadne
- nezávislé premenné podliehajú nenáhodnej chybe merania.

Metóda inštrumentálnych premenných v regresnom modeli umožňuje všeobecné riešenie problému endogénnych nezávislých premenných. Pre použitie inštrumentálnych premenných pre endogénne premenné $\mathbf{X}$ je potrebné nájsť maticu pozorovaných nezávislých premenných $\mathbf{Z} = \{z_{ij}\}$, kde z_{ij} sú individuálne náhodné nezávislé premenné $j = 1, \dots, p$ pre vzorky $i = 1, \dots, n$, ktoré nie sú súčasťou rovnice (1) a spĺňajú nasledovné kritériá:

- $\mathrm{cov}(\mathbf{Z}, \boldsymbol{\varepsilon}) = 0$, z čoho vyplýva, že $\mathbf{Z}$ je exogénna premenná v rovnici (1).
- Lineárne regresie pre premenné $j = 1, \dots, p$ $\mathbf{X}_{\mathrm{j}}$ ako závislých premenných a korešpondujúcich inštrumentálnych premenných $\mathbf{Z}_{\mathrm{j}}$, je možné definovať nasledovne, viď (Wooldridge, 1996) a (Wooldridge, 2002); vynecháme dolný index j pre jednoduchšiu čitateľnosť:

$$\mathbf{X} = \mathbf{Z}\boldsymbol{\theta} + \boldsymbol{\omega} \tag{3}$$

kde $\boldsymbol{\omega} = \omega_i$ je vektor náhodných chýb pre vzorky $i = 1, \dots, n$, pri ktorom predpokladáme, že má normálne rozdelenie s priemerom 0 a konštantnou rozptylom σ^2 a $\boldsymbol{\theta}$ sú neznáme parametre pre $j = 1, \dots, p$, vrátane prieniku θ_0, ktorý je zadefinovaný podobne ako v rovnici (1). Tiež predpokladáme, že korelácia medzi $\boldsymbol{\omega}$ a všetkými nezávislými premennými v rovniciach (3) je 0. Ďalším dôležitým predpokladom je, že koeficienty pre $\mathbf{Z}$ sú nenulové, $\boldsymbol{\theta} \neq 0$.

Rovnice (3) sú tiež nazývané ako redukovaná forma rovníc pre endogénne nezávislé premenné $\mathbf{X}$. Odhad parametrov $\boldsymbol{\beta}$, použitím dvojstupňovej metódy minimalizácie najmenších štvorcov, je nasledovný (Angrist a kol., 1995):

$$\widehat{\boldsymbol{\beta}} = ((\mathbf{Z}\boldsymbol{\theta})^{\mathrm{T}}\mathbf{X})^{-1}((\mathbf{Z}\boldsymbol{\theta})^{\mathrm{T}}\mathbf{y}) \tag{4}$$

kde odhad pre parametre $\boldsymbol{\theta}$ je:

$$\widehat{\boldsymbol{\theta}} = (\mathbf{Z}'\mathbf{Z})^{-1}(\mathbf{Z}'\mathbf{X}) \tag{5}$$

dosadením rovnice (5) do rovnice (4), dvojstupňový odhad najmenších štvorcov je odvodený nasledovne (Angrist a kol., 1995):

$$\widehat{\boldsymbol{\beta}}_{2SLS} = (\mathbf{X}^{\mathrm{T}}\mathbf{Z}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{Z}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{y}) \tag{6}$$

Výhoda dvojstupňovej metódy je, že eliminuje nekonzistentnosť štandardnej metódy najmenších štvorcov, ktorá nastáva ak kovariancia medzi náhodnými chybami $\boldsymbol{\varepsilon}$ a $\boldsymbol{\omega}$ je nenulová (Angrist a kol., 1995).

Ďalej sa budeme venovať modifikovanej dvojstupňovej metóde, ktorá sa využíva v prípade ak ku korelácii dochádza medzi inštrumentálnymi premennými a náhodnými chybami štrukturálneho modelu.

Ku skreslenie odhadu parametrov modelu použitím dvojstupňovej metódy najmenších štvorcov dochádza ak existuje korelácia medzi inštrumentálnymi premennými $\mathbf{Z}$ a náhodnými chybami $\boldsymbol{\varepsilon}$. V tomto prípade je potrebný alternatívny odhad pre parametre $\boldsymbol{\theta}$, ktorý eliminuje túto koreláciou. V článku (Angrist a kol., 1999) autori odporúčajú použitie modifikovanie dvojstupňovej metódy použitím jackknife metódy.

Odhadnuté parametre pre inštrumentálne premenné $\widehat{\boldsymbol{\theta}}$ môžu byť vyjadrené pre každú pozorovanú vzorku *i* nasledovne (Angrist a kol., 1999):

$$\mathbf{Z}_i\widehat{\boldsymbol{\theta}} = \mathbf{Z}_i(\mathbf{Z}'\mathbf{Z})^{-1}(\mathbf{Z}'\mathbf{X}) = \mathbf{Z}_i\boldsymbol{\theta} + \mathbf{Z}_i(\mathbf{Z}'\mathbf{Z})^{-1}(\mathbf{Z}'\boldsymbol{\omega}) \tag{7}$$

Kovariancia rovnice (5) s ε_i je odvodená ako, viď (Angrist a kol., 1999), $\mathbf{Z}_i(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}_i^T \mathrm{E}[\varepsilon_i\omega_i]$, kde $\mathrm{E}[\varepsilon_i\omega_i] = \sigma_{\varepsilon\omega}$, E je označenie pre očakávanú hodnotu, z čoho vyplýva, že je nenulová. Jackknife metóda eliminuje závislosť $\mathbf{Z}_i\widehat{\boldsymbol{\theta}}$ na endogénne regresné premenné $\mathbf{X}_i$ ak odhadneme inštrumentálne parametre nasledovne (Angrist a kol., 1999):

$$\widetilde{\boldsymbol{\theta}}_{-i} = (\mathbf{Z}^{\mathrm{T}}{}_{-i}\mathbf{Z}_{-i})^{-1}(\mathbf{Z}^{\mathrm{T}}{}_{-i}\mathbf{X}_{-i}) \tag{8}$$

kde $\mathbf{Z}_{-i}$ je $(n-1)\times p$ matica pozostávajúca zo všetkých riadkov $\mathbf{Z}$ okrem *i*-tého riadku a podobne pre $\mathbf{X}_{-i}$. Takže odhad pre inštrumentálne premenné je $\mathbf{Z}_i\widetilde{\boldsymbol{\theta}}_{-i} = \mathbf{Z}_i(\mathbf{Z}^{\mathrm{T}}{}_{-i}\boldsymbol{Z}_{-i})^{-1}(\mathbf{Z}^{\mathrm{T}}{}_{-i}\boldsymbol{X}_{-i})$, ktorého dosadením do rovnice (6) dostaneme tzv. jackknife dvojstupňový odhad najmenších štvorcov (Angrist a kol., 1999):

$$\widehat{\boldsymbol{\beta}}_{\mathrm{J2SLS}} = \left(\mathbf{Z}_i\widetilde{\boldsymbol{\theta}}_{-i}\mathbf{X}\right)^{-1}\left(\mathbf{Z}_i\widetilde{\boldsymbol{\theta}}_{-i}\mathbf{y}\right) \tag{9}$$

Na vyriešenie rovnice (9) je potrebný výpočet *n*-krát najmenších štvorcov pre každú pozorovanú vzorku $i = 1, \ldots, n$. Vo vzorkách s veľkým počtom pozorovaní tento výpočet môže byť príliš zdĺhavý. Náročnosť výpočtu je možné zredukovať pomocou metódy vplyvných pozorovaní (*influential observations*) (Cook, 1979):

$$\mathbf{Z}_i\widetilde{\boldsymbol{\theta}}_{-i} = \mathbf{Z}_i\frac{(\mathbf{Z}'\mathbf{Z})^{-1}}{1-\mathbf{Z}_i'(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}_i}(\mathbf{Z}'\mathbf{X} - \mathbf{Z}_i'\mathbf{X}_i) = \frac{\mathbf{Z}_i\widehat{\boldsymbol{\theta}}-h_i\mathbf{X}_i}{1-h_i} \tag{10}$$

kde $h_i = \mathbf{Z}_i'(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}_i$ pre pozorovania $i = 1, \ldots, n$, ktoré sa nazýva páka pozorovania (*observation leverage*), je súčasťou výpočtu vo viacerých softvérových balíčkoch.

Autori v článku (Angrist a kol., 1999) tiež navrhujú alternatívnu verziu jackknife metódy, kde modifikujú komponent $\mathbf{Z}^{\mathrm{T}}\mathbf{X}$. Tento článok tiež prezentuje detailné odvodenie vychýlenia pre obidve jackknife metódy.

3 Výsledky

V tejto časti ukážeme aplikáciu horeuvedených metód použitím synteticky simulovanými dátami. Cieľom je demonštrácia, že metódy sú pomerne ľahko aplikovateľné pomocou balíka Stein IV v softvéri R, ktorý bol vyvinutý autormi článku (Angrist a kol., 1995) a (Angrist a kol., 1999). Tento balík tiež obsahuje štandardnú metódu najmenších štvorcov, takže je možné porovnanie všetkých horeuvedených metód, t. j. porovnanie odhadovaných parametrov modelu a ich príslušných štandardných chýb. Detailné informácie funkcií použitých v balíku Stein IV sú poskytnuté autormi týchto článkov na webe, ktoré sú uvedené v použitej literatúre, referencie [9] a [10].

Pozorovania pre matice **X**, **Z** a vektor **y** je vygenerovaný kódom v softvéri R, ktorý je štandardný spôsob simulácie údajov pomocou Monte Carlo metódy , kde predpokladáme koreláciu medzi premennými **X** a **Z**[2]. R kód vygeneruje náhodné čísla pre n = 100 vzoriek a p = 5 nezávislých premenných v maticiach **X** a **Z**. Pozorovania pre závislú premennú **y** sú vygenerované tak aby korelovali s maticou pozorovaní **X**, takže odhadnuté parametre v ekonometrickom modeli sú štatisticky významné.

Následne použitím funkcií *ols.est(y,X,SE=TRUE)*, *tsls.est(y,X,Z,SE=TRUE)* a *jive.est(y,X,Z,SE=TRUE)* z balíka Stein IV odhadneme parametre modelu a ich príslušné štandardné chyby. Nasledovné tabuľky obsahujú odhadované parametre a štandardné chyby pre metódy najmenších štvorcov, dvojstupňovej a jackknife metódy.

Tab. 1: Ekonometrický model – odhadnuté parametre pre tri metódy

Odhadnuté parametre β ($\hat{\beta}_0 = 0$)			
	NŠ	2NŠ	Jackknife
$\hat{\beta}_1$	0.97	0.85	0.82
$\hat{\beta}_2$	0.99	0.82	0.76
$\hat{\beta}_3$	0.95	1.08	1.12
$\hat{\beta}_4$	1.03	1.19	1.22
$\hat{\beta}_5$	1.05	1.05	1.05

Zdroj: Výpočet autora

Z Tab. 2 môžeme konštatovať, že odhadnuté parametre sa odlišujú pre jednotlivé metódy odhadu modelu.

Tab. 3: Štandardné chyby odhadnutých parametrov – pre tri metódy

Štandardné chyby odhadnutých parametrov			
	NŠ	2NŠ	Jackknife
SE_1	0.082	0.139	0.344
SE_2	0.086	0.156	0.407
SE_3	0.067	0.104	0.288
SE_4	0.080	0.129	0.141
SE_5	0.072	0.094	0.128

Zdroj: Výpočet autora

Dôležitejší ukazovateľ je v Tab. 4, ktorý ukazuje štandardnú chybu pre jednotlivé odhadnuté parametre. Tu je zreteľné, že najväčšiu chybu majú parametre odhadnuté metódou

[2] Pri generovaní náhodných údajov sme využili balíček rTRNG v R, viď referencia [11] v použitej literatúre.

jackknife. Z tohto vyplýva, že metódy najmenších štvorcov a jej dvojstupňovej verzie môžu podhodnocovať štandardné chyby parametrov ak dochádza ku korelácii medzi premennými **X** a inštrumentálnymi premennými **Z**.

4 Záver

Cieľom tejto štúdie je ukážka teoretického rámca pre odhad parametrov ekonometrického modelu pomocou metódy najmenších štvorcov. Štandardná metóda najmenších štvorcov je modifikovaná za predpokladu, že sú regresné premenné exogénne, pri ktorom je nutné definovať redukovaný model pre inštrumentálne premenné. V tomto prípade dochádza ku korelácii náhodných chýb redukovaného a štrukturálneho modelu, ktorá je eliminovaná použitím dvojstupňovej metódy najmenších štvorcov. V ďalšom kroku sme ukázali jej modifikáciu pomocou jackknife metódy, ktorá slúži na elimináciu vychýlenia ak sa korelácia nachádza medzi inštrumentálnymi premennými a náhodnými chybami štrukturálneho modelu.

Pre prípadovú štúdiu sme použili náhodne vygenerované údaje pomocou metódy Monte Carlo. Následne sme ukázali aplikáciu horeuvedených metód odhadu parametrov ekonometrického modelu použitím balíka Stein IV v softvéri R. Na základe výsledkov môžeme skonštatovať, že štandardné chyby odhadnutých parametrov sú značne väčšie pre metódu jackknife, čo naznačuje, že použitím štandardnej metódy najmenších štvorcov a jej dvojstupňovej verzie môže dochádzať k významnému podhodnoteniu štandardných chýb odhadnutých parametrov modelu. Ďalší výskum by sa mal zamerať na empirickú štúdiu kde sú tieto metódy aplikované na skutočné ekonomické ukazovatele, keďže simulované údaje nie vždy reflektujú štruktúru rozdelenia skutočných ekonomických údajov.

Príspevok bol spracovaný v rámci riešenia grantovej úlohy VEGA 1/0047/23 *Význam priestorových spillover efektov v kontexte priority EÚ zelenšia a bezuhlíková Európa.*

Literatúra

1. Angrist, J.D., Imbens, G.W., Krueger, A. (1995). Jackknife instrumental variables estimation. *NBER Technical Working Paper No. 172*. doi: 10.3386/t0172.
2. Angrist, J.D., Imbens, G.W., Krueger, A. (1999). Jackknife instrumental variables estimation. *Journal of Applied Econometrics 14*(1), 57-67. http://www.jstor.org/stable/223249.
3. Cook, R.D. (1979). Influential Observations in Linear Regressions. *Journal of the American Statistical Association 74*, 169-174.
4. Hahn, J., Hausman, J. (2002). Weak Instruments: Diagnosis and Cures in Empirical Econometrics. *Unpublished draft*.
5. Hahn, J., Hausman, J. (2005). Estimation with Valid and Invalid Instruments. *Annales D'Économie et de Statistique*, No. 79/80.
6. Wooldridge, J. M. (1996). Estimating system of equations with different instruments for different equations. *Journal of Econometrics 74*, 387-405.
7. Wooldridge, J. M. (2000). Introductory econometrics: a modern approach. *Cincinnati*, OH: Sout-Western.
8. Wooldridge, J. M. (2002). Econometric analysis of cross section and panel data. *Massachusetts Institute of Technology*, Cambridge, Massachusetts.

9. R/steiniv.R documentation. Retrieved August 8, 2023 from https://rdrr.io/cran/SteinIV/src/R/steiniv.R.

10. R/steiniv.R documentation. Retrieved August 8, 2023 from https://rdrr.io/cran/SteinIV/man/jive.est.html.

11. rTRNG documentation. Retrieved August 8, 2023 from https://cran.r-project.org/web/packages/rTRNG/vignettes/mcMat.html.

Modelovanie investičných portfólií jazykom Python – Monte Carlo prístup
Modelling investment portfolios using Python – Monte Carlo approach

Richard Mišek[1]

Abstrakt
K tvorbe portfólia existuje množstvo analytických metód. Ako prvý definoval Markowitz modernú teóriu portfólia, ktorá sa zameriava na optimalizáciu skladby cenných papierov pomocou minimalizácie rizika, pričom využil ako mieru rizika rozptyl (štandardnú odchýlku) a maximalizáciu výnosov portfólia. Uvedený problém možno formulovať ako úlohu kvadratického programovania. V našej štúdii použijeme alternatívny spôsob riešenia pomocou náhodného priraďovania váh akciám a následného zisťovania efektívnosti riešení, čím získame efektívnu hranicu pre analyzovaný problém, ktorá je cieľom riešenia uvedenej úlohy. Je tak umožnený výber požadovaných alternatív na celej množine bez nutnosti stanovenia optimalizačnej úlohy. Spôsob, ktorým budeme realizovať zloženie portfólií je založený na metóde Monte Carlo v jazyku Python a realizovaný na indexe Dow Jones Industrial Average. Cieľom je priniesť väčšiu flexibilitu investorom pri procese výberu portfólia poskytnutím alternatívneho nástroja na rozhodovanie.

Kľúčové slová
Markowitz, Portfólio, Monte Carlo, Python

Abstract
There are several analytical methods for portfolio construction. Markowitz was the first to define modern portfolio theory, which focuses on optimizing the composition of securities by minimizing risk, using variance (standard deviation) as a measure of risk, and maximizing portfolio returns. The above problem can be formulated as a quadratic programming problem. In our study, we use an alternative solution method by randomly assigning weights to stocks and then finding the efficiency of the solutions, thus obtaining an efficient frontier for the problem under analysis, which is the objective of solving the above problem. This enables the selection of the desired alternatives on the entire set without the need to set up an optimization problem. The way we will implement the composition of the portfolios is based on the Monte Carlo method in Python and implemented on the Dow Jones Industrial Average index. The goal is to bring more flexibility to investors in the portfolio selection process by providing an alternative decision-making tool.

Key words
Markowitz, Portfolio, Monte Carlo, Python

JEL classification
G11, G17

[1] Ing. Richard Mišek, Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra operačného výskumu a ekonometrie, Dolnozemská cesta 1/b, 852 35 Bratislava 5, risomisek@gmail.com.

1 Úvod

V dnešnej dobe existuje mnoho investičných platforiem, poskytovateľov investičných produktov, správcovských spoločností, obchodníkov s cennými papiermi a rôznych skupín, ktoré nie vždy hája dobro pre klienta vo svete financií. Jedným z problémov, kvôli ktorému osoba neznalá finančných trhov podľahne kúpe nekvalitnej investície je nátlak s prísľubom istého výnosu alebo dôvera v spoločnosť, ktorá sa javí ako profesionálna. Za dôvodmi nadobudnutia nekvalitného portfólia stojí zväčša neznalosť klienta vo svete financií. Mnoho subjektov ukracuje klientov o zisk rôznymi nevhodnými poplatkami alebo nízkou vzdelanosťou predajcov finančných produktov.

Prínosom a zámerom tejto práce je vytvorenie optimálneho portfólia, podľa preferencie rizika investora, z vopred predvolených tridsiatich akcií indexu Dow Jones Industrial Average. Následné premietnutie tohto princípu na výber vlastného optimálneho portfólia, na základe stanovených požiadaviek, investorovi umožní vybrať si práve jeho požadovanú skladbu cenných papierov. V práci budeme porovnávať dva prístupy a to stanovenie optimalizačnej podmienky s výpočtom portfólia štandardným spôsobom, a následný alternatívny spôsob výpočtu s vyhotovením množiny portfólií, ktorá zobrazí dáždnikovú plochu s pomermi výnos a riziko, a vyobrazené budú aj váhy s názvami akcií pri jednotlivých portfóliách. V analýze budú využité denné uzatváracie ceny pre podkladové aktíva z indexu Dow Jones Industrial Average za obdobie piatich rokov od 01.10.2018 až po 1.10.2023. Týmto rozmedzím vieme zachytiť správanie trhu v období finančnej krízy, ale aj mimo nej. Spolu s aplikačno-programovým rozhraním (ďalej API) od Yahoo Finance, využívame aktuálne ceny a investori tak dokážu realizovať analýzu portfólia s najnovšími dátami. Mišek (2023) skúmal nastavenie vlastného výberu portfólia interaktívnou formou na základe náhodného priradenia váh a vytvorenia časti dáždnikovej plochy portfólií, pričom s 8 aktívami dokázal interpretovať veľkú časť všetkých dostupných portfólií. Pri výbere si investor môže zvoliť z vrchnej hranice krivky, ktorá predstavuje efektívnu hranicu portfólií. Na tejto hranici sa nachádzajú portfólia s najväčším výnosom pri požadovanom riziku.

2 Teória portfólia

Moderná teória portfólia bola vyvinutá v 50. rokoch 20. storočia Harrym Markowitzom. Markowitz zmenil pohľad na princíp tvorby portfólia, nakoľko sa pri optimalizácii nepozeral na portfólio ako na jednotlivé cenné papiere, ale bral portfólio ako celok, ktorý treba analyzovať. Müller (1988) v práci uvádza, že sa treba pozerať pri jednotlivých aktívach na ich očakávané výnosy spolu s rizikovosťou týchto aktív. Základom sa tak pri tvorbe portfólia stáva diverzifikácia a spoločný vývoj cien jednotlivých aktív. Princíp tohto prístupu spočíva v nájdení optimálnej skladby portfólia nami preddefinovaných akcií, na základe stanovenej optimalizačnej podmienky. Touto podmienkou je buď minimalizácia rizika pri stanovenom výnose alebo maximalizácia výnosu, pri stanovenej hladine rizika. Pri výpočte optimálneho portfólia sú využívané denné výnosy aktív a viaceré veličiny založené na matematických vzorcoch. Výpočet očakávanej výnosnosti portfólia sa vypočíta ako vážený priemer výnosností, ktoré očakávame pre jednotlivé aktíva.

$$E(r_p) = \sum_{i=1}^{n} \cdot w_i r_i = \mathbf{w^T} \cdot \mathbf{r}, \tag{1}$$

kde, $E(r_p)$ je očakávaná výnosnosť portfólia,

n predstavuje počet aktív

w_i je váha *i*-teho aktíva

r_i je výnos *i*-teho aktíva

$\mathbf{w^T}$ je transponovaný vektor váh akcií

$\mathbf{r}$ je vektor očakávaných výnosností akcií

Po výnose, ktorý bude predstavovať vertikálnu časť grafu, je horizontálnou časťou riziko, nazývané aj ako volatilita portfólia, ktoré vyjadrujeme pomocou strednej kvadratickej odchýlky alebo rozptylu.

$$\sigma_p^2 = \sum_{i=1}^{n} \cdot \sum_{j=1}^{n} \cdot w_i w_j \, cov(r_i, r_j) = \mathbf{w^T C w}$$

$$\sqrt{\sigma_p^2} = \sigma_p \tag{2}$$

kde, σ_p je štandardná kvadratická odchýlka

n predstavuje počet aktív

σ_p^2 je rozptyl výnosnosti portfólia

w_i, w_j sú váhy *i*-teho a *j*-teho aktíva

$cov(r_i, r_j)$ je prvok kovarianček matica výnosov aktíva *i*-teho a *j*-teho

C je kovariančná matica aktív

Kovariančná matica, predstavuje maticu *n* x *n*, z čoho vyplýva, že sa jedná o štvorcovú maticu. Na hlavnej diagonále sa nachádzajú rozptyly a ostatné miesta predstavujú kovariancie. Kovariancia hovorí o závislosti dvoch aktív medzi sebou. Rozoznávame tu kladnú a zápornú kovarianciu. Pri zápornej kovariancii vyjadruje rast prvého aktíva, pokles aktíva druhého. Pri kovariancii kladnej sa jedná o vzájomný rast oboch aktív v rovnakom smere. Aby sa dalo hovoriť o závislosti výnosností dvoch aktív, je nutné ku kovariancii pridať aj korelačný koeficient, ktorý nadobúda hodnoty z intervalu $< -1 ; 1 >$.

$$\rho_{ij} = \frac{cov\,(r_i\,,\ r_j)}{\sigma_i\,\sigma_j} \tag{3}$$

Pri skúmaní teórie portfólia sa vychádza z prípustnej a efektívnej množiny portfólií. Prípustná množina portfólií je tvorená všetkými portfóliami, ktoré je možné zostrojiť z danej množiny akcií. Efektívna množina dominuje množine prípustnej tým, že výnos je na tejto množine najvyšší možný pre stanovené riziko. Inak povedané väčší výnos sa pri danom riziku nedá dosiahnuť a platí to pre celú hranicu. Matematicky sa dá zapísať výnosnosť portfólia a jeho riziko vyjadrené štandardnou odchýlkou nasledovne.

$$r_p = \sum_{i=1}^{n} w_i \cdot r_i$$

$$\sigma_p = \sqrt{\sum_{i=1}^{n} \cdot \sum_{j=1}^{n} \cdot w_i \ \cdot \ w_j \ \cdot \ \sigma_{ij}} \tag{4}$$

kde, súčet váh je rovný jednej
r_p je výnosnosť portfólia
r_j je výnosnosť *j*-teho aktíva

n predstavuje počet aktív

σ_p^2 je rozptyl výnosnosti portfólia

w_i, w_j sú váhy *i*-teho a *j*-teho aktíva

σ_{ij} je vzájomná kovariancia medzi aktívom *i* a *j*

Všeobecná matematická formulácia problému optimalizácie portfólia ako úlohy kvadratického programovania je v tvare:

$$min\, \sigma_p^2\,(w) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} \cdot w_i \cdot w_j$$

$$\sum_{j=1}^{n} r_j \cdot w_j \geq r_p$$

$$\sum_{j=1}^{n} w_j = 1$$

$$w_i, w_j \geq 0,\ j = 1,2, \dots n \qquad (5)$$

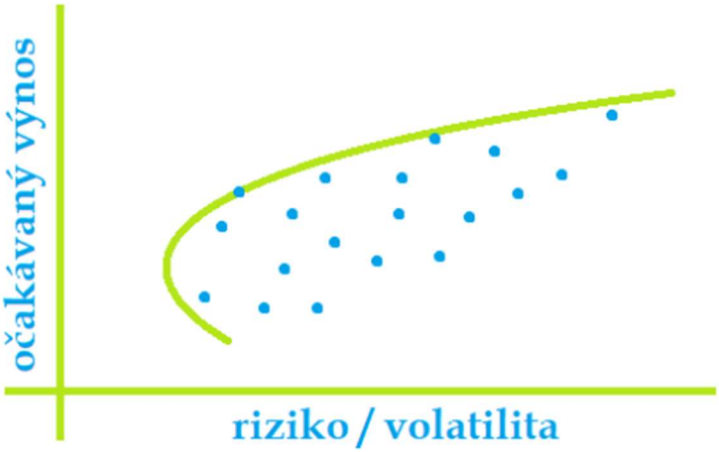
kde r_p predstavuje minimálnu mieru výnosnosti požadovanú investorom.

Výpočty je možné realizovať na základe dvoch typov výnosov: logaritmických a jednoduchých výnosov. Zvolená výnosnosť bola logaritmická, ktorej hodnoty sú na základe pozorovaní nižšie ako pri jednoduchej výnosnosti. Zároveň sa tieto hodnoty viac približujú ku skutočnej výnosnosti portfólia, nakoľko jednoduché výnosy bývajú často nadhodnotené voči realite. Naše skúmanie potvrdili aj viaceré príspevky, v ktorých sa tvrdí nasledovné.

Hudson (2014) uvádza teoretické dôkazy, ktorými sa zistilo, že priemer vypočítaný pomocou logaritmických výnosov je menší ako priemer vypočítaný pomocou jednoduchých výnosov o hodnotu súvisiacu s rozptylom množiny výnosov, pričom rozptyl je relatívne nemenný.

Miskolczi (2017) vo svojej analýze tvrdí, že rovnice uvedené v prípade akcií ukazujú, že logaritmický výnos má výhodu oproti jednoduchému výnosu, a to, že viacperiodový logaritmický výnos možno vypočítať ako súčet jednoperiodových logaritmických výnosov, zatiaľ čo viacperiodový jednoduchý výnos je súčinom jednoperiodových jednoduchých výnosov, čo môže viesť k výpočtovým problémom pre hodnoty blízke nule.

Obr 1: Efektívna hranica a prípustné portfóliá

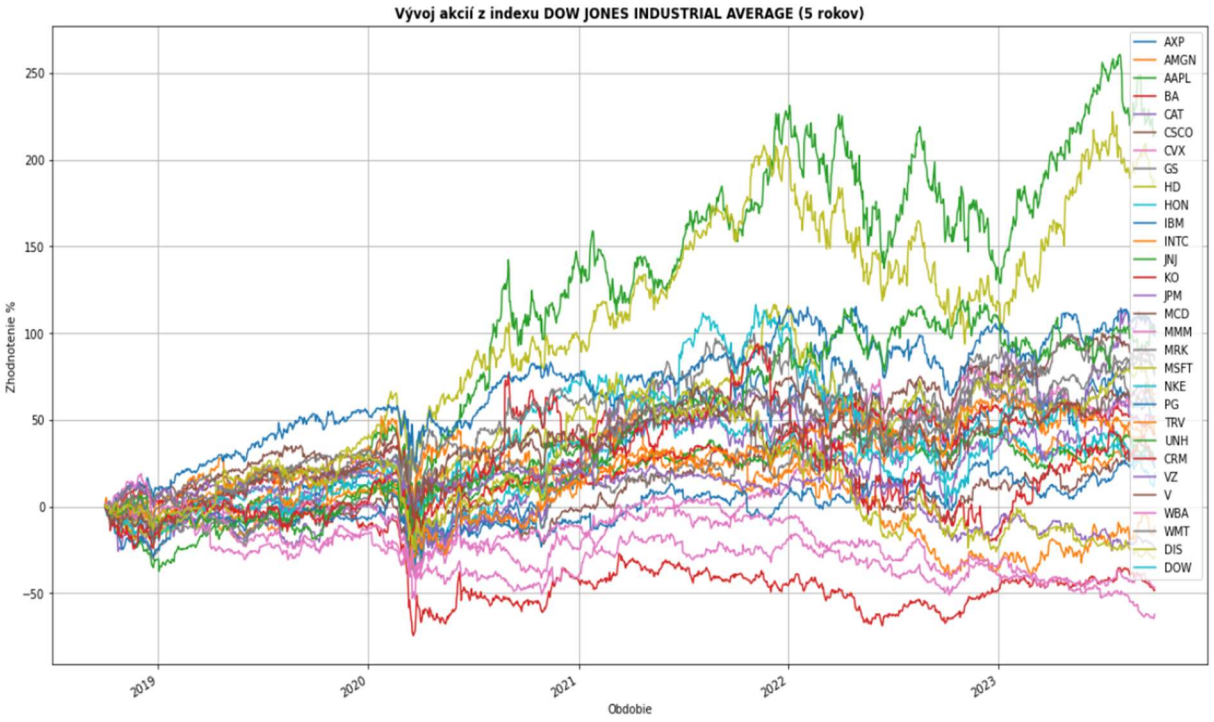


Zdroj: Vlastné spracovanie

3 Alternatívny spôsob riešenia

Alternatívny spôsob výpočtu portfólií pomocou Markowitzovej teórie je realizovaný v softvéri Jupyter Notebook, v ktorom sme využili programovací jazyk Python. Po naimportovaní knižníc, vrátane knižnice yfinance, ktorá nám slúži ako poskytovateľ dát v reálnom čase, sme načítali ceny všetkých 30 aktív za obdobie posledných piatich rokov. Po uložení cien do databázového rámcu vykresľujeme pohyb týchto denných cien v rozmedzí 1.10.2018 až 1.10.2023 a dostávame tak prvotnú vizuálnu informáciu o výnosoch jednotlivých podkladových aktív. Dáta o jednotlivých cenných papierov boli čerpané od Yahoo Finance (2023) a bola využitá voľne poskytovaná API yfinance.

Obr. 2: Denné výnosy DJ30



Zdroj: Vlastné spracovanie

Pomocou denných cien zostrojíme denné logaritmické výnosy akcií. Logaritmické výnosy využívame z dôvodu presnejšej reprezentácie výnosov, nakoľko anualizovaná výnosnosť bez logaritmickej zložky zvykne hodnoty výnosov navyšovať a vnáša tak do analýz nepresnosť. V prvom kroku sme na základe vypočítaných výnosov vypočítali kovariančnú maticu, kde sme pomocou maticového násobenia s plným počtom 30 aktív dospeli k výsledku, že portfólio má volatilitu 21,24%**.** Pri hľadaní výnosu portfólia sme vykonali maticový súčin pre rovnomerne rozložené váhy a logaritmické ročné výnosy podľa vzorca (2). Výnosnosť portfólia bola vypočítaná na 5,72%, ako je vidno na obrázku 3. Následne sa budeme pokúšať o prekonanie týchto parametrov a to spôsobom, aby pri 21,24% volatilite existoval vyšší očakávaný výnos ako pôvodných 5,72%.

Obr. 3: Výsledok rovnomerného zloženia portfólia

```
pfolio_vol = (np.dot(weights.T, np.dot(log_returns.cov() * tradingDaysIn5YearsYearly, weights))) ** 0.5
print ("Volatilita portfólia : "+str(round(pfolio_vol*100,2)) + '%')
print("Váhy : " + str(weights))

#Vykonnost pred markowitzom
log_returns_yearly = log_returns.mean() * tradingDaysIn5YearsYearly
pfolio = str(round(np.dot(log_returns_yearly, weights)*100, 2)) + ' %'
print("Vykonnost portfolia : " + pfolio)
```

```
Volatilita portfólia : 21.24%
Váhy : [0.03333333 0.03333333 0.03333333 0.03333333 0.03333333 0.03333333
 0.03333333 0.03333333 0.03333333 0.03333333 0.03333333 0.03333333
 0.03333333 0.03333333 0.03333333 0.03333333 0.03333333 0.03333333
 0.03333333 0.03333333 0.03333333 0.03333333 0.03333333 0.03333333
 0.03333333 0.03333333 0.03333333 0.03333333 0.03333333 0.03333333]
Vykonnost portfolia : 5.72 %
```

Zdroj: Vlastné spracovanie

Analýzu sme realizovali na dvoch dátových množinách. Prvá množina bola neupravená tridsať členná skupina akcií. Pri druhej množine sme stanovili podmienku ôsmych najlepších akcií na základe pomeru výnos a riziko, čoho výsledky sú zobrazené na obrázku 4. Pre obe platil nasledovný postup. Stanovili sme si modelovaný počet portfólií na 100 000, pričom sa jedná o náhodné priradenie váh akciám tak, aby ich súčet váh bol rovný jednej. Na vykreslenie bolo nutné v stotisíc iteráciách priradiť spolu s váhami aj vypočítané páry pre volatilitu a výnosnosť daného vytvoreného portfólia. Pri druhej množine aktív, nakoľko sa jednalo iba o osem členné zastúpenie, sme pozorovali o tretinu kratší čas tvorby portfólia. Nakoľko je zoskupenie akcií nižšie a mali sme k dispozícii 100 000 rôznych portfólií a vedeli sme s vysokou presnosťou simulovať a znázorniť takmer celú dáždnikovú plochu naplnenú týmito portfóliami. Čím vyššie sa na krivke dostávame, tým sme bližšie k efektívnej hranici. Na úplné zostavenie efektívnej hranice by bolo nutné zostrojiť túto krivku s mnohonásobne väčším počtom portfólií. Ak by sme túto analýzu vykonávali na výkonných počítačoch s vysokým výpočtovým výkonom, časom by sme sa k tejto hranici vedeli ešte viac priblížiť. Na nižšie priložených grafoch sa nachádzajú páry rizika, volatility a váh s názvami ich aktív. Tieto portfólia sa po prejdení kurzorom v prostredí Jupyter Notebook zobrazia a je teda možné sledovať presnú skladbu portfólií, bez nutnosti opätovne zadávať vstupné parametre. Čím vyššie sa na vertikálnej osi dostávame, tým viac stúpa výnos pri danom riziku. Naopak, pokiaľ by bol rozhodujúcim faktorom minimalizácia rizika pri danom výnose, sledovali by sme horizontálnu os a našli by sme také portfólio, ktoré sa pri danom výnose nachádza na efektívnej hranici s najmenším možným rizikom. Pri sledovaní výsledkov zobrazených na obrázku 6 znázorňujúcom hodnoty pre osem akcií vidíme, že zámer navýšiť očakávanú výnosnosť pri

stanovenej volatilite bol splnený. Pri totožnej volatilite sa tak jedná o navýšenie výnosnosti portfólia z pôvodných 5,72% na 17,15%, čo činí takmer trojnásobný očakávaný výnos. Pri investovaní by investor ako výhodu mohol vnímať aj náklady na tvorbu portfólia, nakoľko s nákupom akcie sa spája zväčša aj vstupný poplatok za obchod. Touto redukciou dospejeme teda nie iba k zvýšeniu očakávaného výnosu, ale aj zníženiu transakčných nákladov pri zaobstarávaní cenných papierov. Typ akým sme simulovali zostrojenie portfólií je Monte Carlo a jedná sa teda o náhodný výber váh, priraďovaný akciám tak, aby ich spoločný súčet bol vždy 100%.

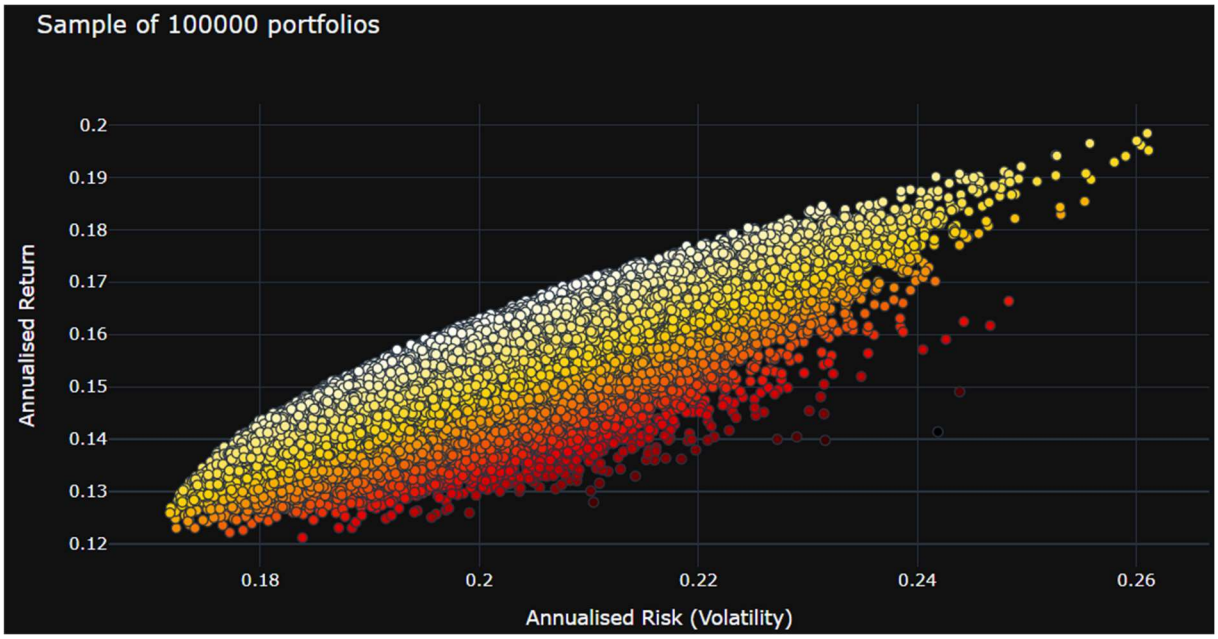
Obr. 4: Zoradených osem najlepších akcií na základe pomeru výnos / riziko

Ticker	Return	Volatility	Best-Stocks
AAPL	0.229802	0.333813	0.583567
MSFT	0.212251	0.313749	0.564946
PG	0.137295	0.216858	0.471712
WMT	0.122702	0.221976	0.395095
UNH	0.140811	0.301021	0.351508
MCD	0.115004	0.233381	0.342802
MRK	0.113034	0.233415	0.334313
CAT	0.141688	0.334946	0.318524
HD	0.099793	0.292793	0.221292
CVX	0.104945	0.359243	0.194702
V	0.091418	0.290205	0.194409
AMGN	0.082902	0.255811	0.187254
GS	0.09554	0.332237	0.18222
KO	0.069648	0.215558	0.160737
JPM	0.079461	0.321573	0.138262
TRV	0.070743	0.290876	0.122879
AXP	0.079174	0.375866	0.117527
JNJ	0.048623	0.2064	0.066005
DOW	0.060312	0.391514	0.064652
CSCO	0.049429	0.28983	0.049785
HON	0.04771	0.273822	0.046417
CRM	0.047678	0.395204	0.03208
IBM	0.040822	0.271871	0.021414
NKE	0.035033	0.332413	0.000099
INTC	-0.025591	0.390416	-0.155195
DIS	-0.068306	0.334939	-0.308433
BA	-0.13172	0.509685	-0.327105
VZ	-0.053104	0.202641	-0.43478
MMM	-0.125474	0.281013	-0.571055
WBA	-0.195006	0.337194	-0.682119

['AAPL', 'MSFT', 'PG', 'WMT', 'UNH', 'MCD', 'MRK', 'CAT']

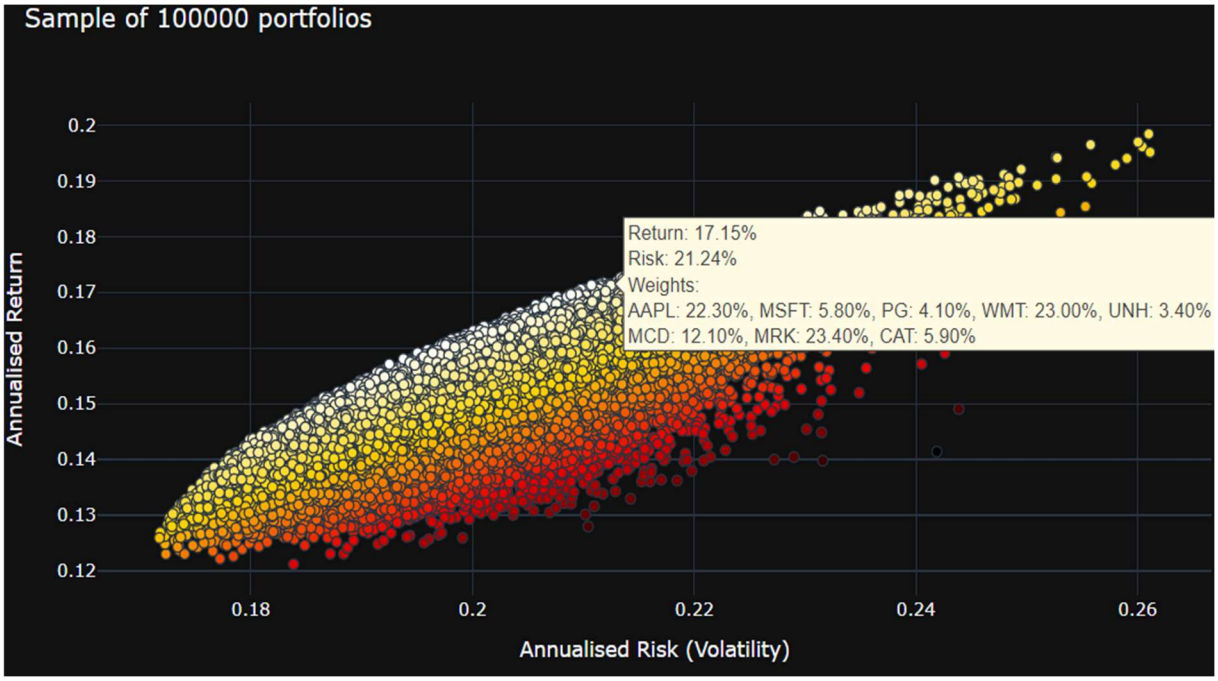
Zdroj: Vlastné spracovanie

Obr. 5: Množina portfólií pre osem akcií



Zdroj: Vlastné spracovanie

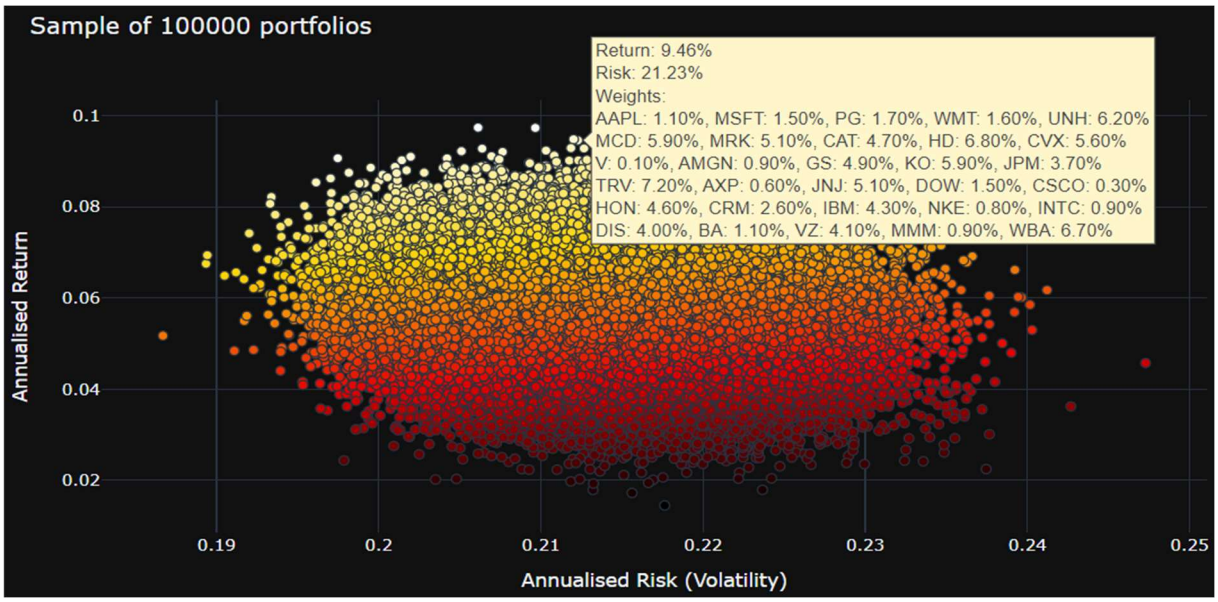
Obr. 6: Zobrazenie výsledku pre druhú množinu portfólií



Zdroj: Vlastné spracovanie

Pri prvej množine, ktorá obsahuje všetkých tridsať spoločností z indexu Dow Jones Industrial Average, nevieme vyobraziť množinu s efektívnou hranicou pri 100 000 portfóliách. Ak by sme simulovali túto problematiku na výkonnejších zariadeniach v hodnotách rovných miliónom portfólií, vedeli by sme sa priblížiť aj k efektívnej hranici. Aj napriek tomu, že momentálny výpočtový výkon zariadenia nepostačoval a zostrojenie už tak obšírnej množiny trvalo desiatky minút, vidíme zvýšenie o takmer 4% pri rovnakej volatilite. Môžeme tak pozorovať pozitívne navýšenie pri simulácii 100 000 portfólií o 30 akciách.

Obr. 7: Zobrazenie výsledku pre prvú množinu portfólií



Zdroj: Vlastné spracovanie

4 Záver

V tomto príspevku sme prezentovali zostrojenie portfólií metódou Monte Carlo, čo sa ukázalo ako ďalší funkčný prístup pri tvorbe portfólií. V porovnaní s použitím optimalizačného modelu, sa vieme priblížiť k efektívnej hranici a namodelovať väčšinu z dáždnikovej plochy obsahujúcej dosiahnuteľné portfóliá. Pri analýze boli využité denné uzatváracie ceny akcií z indexu Dow Jones Industrial Average od 1.10.2018 po 1.10.2023. V porovnaní s rovnomerne rozloženými váhami akcií portfólia, sme zaznamenali pozitívne výsledky zložení z tridsiatich akcií a výrazné zlepšenie zaznamenali portfólia s ôsmimi zložkami. Pri modelovaní boli nutné poznatky z modernej teórie portfólia, avšak nasledovná interpretácia výsledkov je jasná a zrozumiteľná aj pre osobu neznalú tejto problematiky. Simuláciou dostatočného množstva generovaných portfólií sa vieme priblížiť k efektívnej hranici a dosiahnuť tak hodnoty blízke stanoveným optimalizačnou úlohou. Nevýhoda tejto metódy spočíva v nutnosti disponovať vysokým výpočtovým výkonom, v prípade ak je výpočet realizovaný na väčšom počte cenných papierov. Markowitzova teória teda vychádza z historických údajov, jedná sa čisto o odhadované hodnoty a ich výsledky sú tak s najväčšou pravdepodobnosťou iba vôdzkami toho, ako by sa trh do budúcnosti mohol správať. Nemožno teda tvrdiť, že portfóliá zostrojené na základe modernej teórie portfólia reflektujú budúce správanie trhu.

Literatúra

1. Hudson, R. (2014). Calculating and comparing security returns is harder than you think: A comparison between logarithmic and simple returns. Retrieved October 1, 2023, from https://www.sciencedirect.com/science/article/abs/pii/S1057521914001380.
2. Miskolczi, P. (2017). Note on simple and logarithmic return. *Applied Studies in Agribusiness and Commerce*, *11*(1-2), 127–136. Retrieved October 1, 2023, from https://doi.org/10.19041/APSTRACT/2017/1-2/16.
3. Mišek, R. (2023). Optimalizácia portfólia s využitím jazyka Python. (Diplomová práca). Ekonomická Univerzita v Bratislave.
4. Müller, H. (1988). Modern Portfolio Theory: Some Main Results. ASTIN Bulletin: The Journal of the IAA, 18(2), 127-145. doi:10.2143/AST.18.2.2014947.
5. S, N. (2017, July 11). *List of top 4 portfolio theories: Theories: Portfolio management*. Essays, Research Papers and Articles on Business Management. Retrieved October 1, 2023, from https://www.businessmanagementideas.com/portfolio-management/theories-portfolio-management/list-of-top-4-portfolio-theories-theories-portfolio-management/15149.
6. Python Software Foundation. (2023). *Yfinance*. Retrieved October 1, 2023, from https://pypi.org/project/yfinance/0.2.22/.
7. Yahoo.Finance. (2023). Data on individual stocks. Retrieved October 1, 2023, from https://finance.yahoo.com.

ZBORNÍK

XII. medzinárodná vedecká konferencia
„Mladá veda AIESA 2023“
„Participácia doktorandov a mladých vedeckých pracovníkov na budovaní spoločnosti založenej na vedomostiach“

Vydalo: Vydavateľstvo EKONÓM
Dolnozemská cesta 1
852 35 Bratislava
Rozsah: 67 strán
AH: 3,26

ISBN: 978-80-225-5102-1

Fakulta hospodárskej informatiky
Ekonomická univerzita v Bratislave
Dolnozemská cesta 1/b, 852 35 Bratislava
tel.: +421 2 6729 5723, e-mail:veda.fhi@euba.sk

VYDAVATEĽSTVO EKONÓM, BRATISLAVA 2023
ISBN 978-80-225-5102-1