Ekonomická fakulta
Faculty of Economics
Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Diplomová práce

Užitková funkce v rozhodovacím procesu

Vypracoval: Bc. Patrik Marýška
Vedoucí práce: doc. RNDr. Jana Klicnarová, Ph.D

České Budějovice 2023

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Patrik MARÝŠKA**
Osobní číslo:	**E20394**
Studijní program:	**N0613A140025 Aplikovaná informatika**
Studijní obor:	**Softwarové inženýrství**
Téma práce:	**Užitková funkce v rozhodovacím procesu**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Student se seznámí se základy teorie rozhodování a zaměří se na využití užitkové funkce v rozhodovacím procesu. Speciální důraz bude věnovat metodám konstrukce užitkové funkce. Poté navrhne uživatelsky vstřícný proces, který umožní rozhodovateli využít jeho vlastní užitkovou funkci v rozhodovacím procesu.

Metodický postup:

1. Student se seznámí se základními principy a metodami z teorie rozhodování, speciálně se zaměří na teorii užitkové funkce a její využití v rozhodovacím procesu.
2. Student se seznámí s různými metodami konstrukce užitkové funkce a navrhne aplikaci, která umožní rozhodovateli zkonstruovat si užitkovou funkci a následně ji využít v rozhodovacím procesu.
3. Závěrem student uvede modelový příklad, kde zhodnotí možná rozhodnutí v závislosti na zvoleném postupu – s využitím užitkové funkce či bez jejího využití, popř. s využitím alternativní konstrukce užitkové funkce.

Rozsah pracovní zprávy:	**50 – 60 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná**

Seznam doporučené literatury:

1. Anděl, J. (2007). *Statistické metody.* Matfyzpress.
2. Fotr, J. (2006). *Manažerské rozhodování: Postupy, metody a nástroje.* Ekopress.
3. Balakrishnan, N. R., Render, B., Stair, R., & Munson, C. (2017). Managerial decision modeling. In *Managerial Decision Modeling.* De Gruyter.
4. Peterson, M. (2017). *An introduction to decision theory.* Cambridge University Press.
5. Další studijní literatura dle zaměření práce.

Vedoucí diplomové práce: **doc. RNDr. Jana Klicnarová, Ph.D.**
Katedra aplikované matematiky a informatiky

Datum zadání diplomové práce: 6. září 2021
Termín odevzdání diplomové práce: 15. dubna 2022

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

L.S.

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 8. září 2021

Prohlašuji, že svou diplomovou práci "Užitková funkce v rozhodovacím procesu" jsem vypracoval/a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to – v nezkrácené podobě/v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích, dne 05.04.2023

Podpis: _____

Poděkování

Děkuji vedoucí diplomové práce, doc. RNDr. Janě Klicnarové, Ph.D, za cenná doporučení, trpělivost a poskytnutí odborné literatury k tématu.

Obsah

1. Úvod

Rozhodování je velmi častou činností v našich životech. V organizacích patří mezi velmi důležité aktivity a má velký vliv na úspěch či neúspěch společnosti. Můžeme také nalézt rozhodování prováděná individuálními zákazníky. Může se jednat o rozhodování běžných problémů – výběr vhodného hotelu na dovolené, auta, produktu v supermarketu atd. Je tedy zřejmé, že rozhodování v našem běžném životě hraje velmi důležitou roli.

Existují mnohem složitější rozhodování, která jsou založená na naší intuici – například rozhodnutí, která jsou prováděná lidmi při volbě vhodných slov při konverzaci s jinými osobami. Tyto problémy jsou většinou řešeny pomocí intuice a nějakých částečných informací, které jsou k dispozici. Nicméně existují matematické modely a techniky, které nám mohou pomoci nalézt optimální řešení určitých problémů. Někdy může postačit naše intuice k řešení problému, ale ne vždy.

Teorie rozhodování zahrnuje užití matematických postupů v rámci procesu rozhodování. Do těchto situací navíc může zasahovat i náhoda nebo riziko, které mohou ovlivnit naše rozhodnutí. Existují různé modely a techniky, které mohou rozhodovateli pomoci s rozhodnutím.

Někdy je potřeba do hodnocení přidat subjektivní pohled rozhodovatele. Může nás zajímat nejenom finanční stránka, ale i riziko, situace rozhodovatele a další faktory. Zavedením užitkové funkce můžeme zavést subjektivní pohled do hodnocení.

Tato práce se snaží shrnout různé přístupy z teorie rozhodování a zaměřuje se na teorii užitkové funkce a její využití v rozhodovacím procesu. Klade si za cíl navrhnout software, který umožní rozhodovateli zkonstruovat si vlastní užitkovou funkci a následně ji využít v rozhodovacím procesu. Závěrem jsou uvedeny modelové příklady, kde jsou zhodnocena možná rozhodnutí v závislosti na zvoleném postupu – s využitím užitkové funkce či bez jejího využití.

Tato práce je rozdělena na teoretickou a praktickou část. První kapitola teoretické části se zabývá teorií rozhodování. Druhá kapitola je věnována teorii užitku a speciálně konstrukci užitkové funkce. V rámci praktické části je podrobně popsána vytvořená aplikace pro konstrukci užitkové funkce peněz. Jedná se o desktopovou aplikaci v programovacím jazyce Python a frameworku PyQt5. V další kapitole jsou řešeny vybrané modelové příklady.

2. Teorie rozhodování

Teorie rozhodování zahrnuje užití různých matematických postupů sloužících k určení optimální alternativy za předpokladu, že rozhodovatel vybírá mezi několika alternativami. Do těchto situací také vkročuje určité riziko a náhoda, které mohou mít vliv na rozhodování.

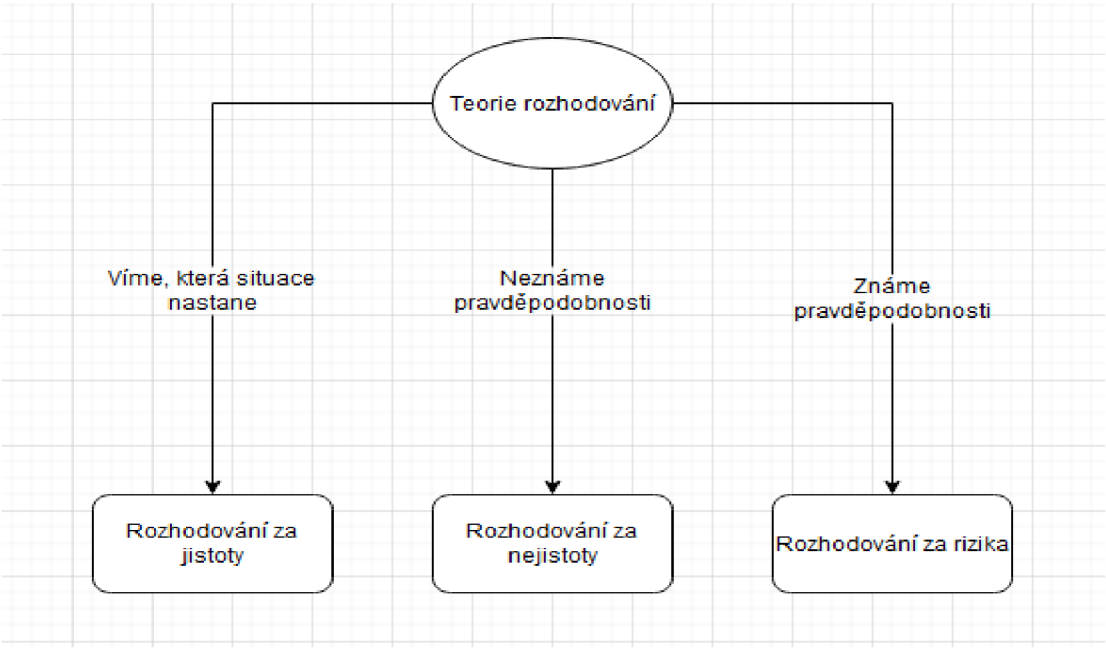
V nějakých případech může vybraná alternativa poskytnout dobré nebo dokonce skvělé výsledky nebo výsledek. Nicméně někdy může nastat neočekávaná událost a způsobit, že vybraná alternativa může poskytnout pouze průměrné nebo dokonce velmi špatné výsledky. (Anderson et al., 2013)

Je očekáváno, že rozhodnutí provádí racionální rozhodovatel, který se snaží maximalizovat svůj zisk. (Klicnarová, 2021)

Při volbě pravidla je nutné brát v úvahu, zda známe pravděpodobnosti, se kterými jednotlivé situace nastanou. Na základě toho můžeme jednotlivé přístupy dělit do tří základních skupin: (Balakrishnan et al., 2017)

- s jistotou víme, která situace nastane – jedná se o **rozhodování za jistoty**,
- nejsou známy pravděpodobnosti, s jakými jednotlivé situace mohou nastat – **rozhodování za nejistoty**,
- víme pravděpodobnosti, s jakými jednotlivé situace mohou nastat – **rozhodování za rizika**.

Obrázek 1: Možné dělení



Zdroj: vlastní

2.1 Definice problému

Při řešení problémů z teorie rozhodování musíme nejprve určit **kritéria rozhodování, varianty**. Zajímá nás, co je cílem rozhodování. Může se jednat o maximalizaci zisku, minimalizace nákladů nebo maximalizace užitku.

Dále musíme vybrat nějakou alternativu ze souboru alternativ mezi kterými se rozhodujeme. Jedná se o všechny možnosti našeho rozhodování. Tento soubor obsahuje všechny zvažované **alternativy**. (Hillier & Lieberman, 2010)

Výsledek bude ovlivněn náhodnými faktory, které nemohou být rozhodovatelem ovlivněny. Tyto náhodné faktory rozhodují, jaká situace nastane v čase, kdy vybraná alternativa bude provedena. Tyto všechny možné stavy systému, které mohou nastat, jsou nazývány **situacemi**. (Hillier & Lieberman, 2010)

Rozhodovatel má přehled o zisku pro každou kombinaci alternativy a situace. Zisk je většinou reprezentován finančním ziskem, nicméně i jiné formy mohou být použity. Tabulka ukazující kombinace všech alternativ a situací se nazývá **rozhodovací matice**. Jedná se o matici, která je tvořena z dopadů zvolení jednotlivých alternativ v případě, že nastaly jednotlivé situace. (Anderson et al, 2013).

Řádky matice obsahují všechny naše alternativy a sloupce matice obsahují všechny možné situace, které mohou nastat – viz tabulka číslo 1. Na jednotlivých pozicích matice R_{ij} se nacházejí důsledky volby A_i a nastalé situace S_j. Předpokládáme, že je *n* situací a *m* alternativ.

Tabulka 1: Rozhodovací matice

	S_1	S_2	…	S_n
A_1	R_{11}	R_{12}	…	R_{1n}
A_2	R_{21}	R_{22}	…	R_{2n}
…	…	…	…	…
A_m	R_{m1}	R_{m2}	…	R_{mn}

Zdroj: vlastní

Dle Hilliera a Liebermana (2010) lze tento proces tedy rozdělit do několika částí:

1. rozhodovatel zvolí alternativy,
2. příroda zvolí situace,
3. každá kombinace alternativy a situace dá výplatu, která je uvedena jako jeden ze záznamů v rozhodovací matici,
4. rozhodovací matice by měla být použita k nalezení optimální alternativy pro

rozhodovatele dle vhodného rozhodovacího kritéria.

2.2 Rozhodování za jistoty

Dle Tahy (1997) rozhodování za jistoty znamená, že jsme si jisti, jaká situace nastane. Jedná se o model, kdy pravděpodobnost jedné situace je 1.0 a ostatní situace mají nulovou pravděpodobnost. Řešení je vybrat alternativu, která nám dává nejlepší výsledek. Můžeme tedy rozhodovací matici zredukovat na matici obsahující jeden sloupec a následně zvolíme maximální hodnotu v případě maximalizace a minimální v případě minimalizace.

2.3 Rozhodování za nejistoty

Rozhodování za nejistoty je přístup k provedení rozhodnutí, která nevyžadují znalost pravděpodobností situací. Tyto přístupy jsou vhodné v situacích, kdy rozhodovatel nevěří, že je schopný odhadnout správně pravděpodobnosti nebo je jednoduše nezná a nedokáže odhadnout. Rozdílné přístupy mohou vést k rozdílným výsledkům, proto musí rozhodovatel zvolit správný přístup na základě vlastního posouzení. (Anderson et al., 2013)

2.3.1 Optimistický přístup

Optimistický přístup vyhodnocuje každou alternativu z pohledu nejlepšího zisku, který může nastat. Alternativa, která je doporučená, dává nejlepší možnou výplatu. (Klicnarová, 2021)

Pro problémy, kde je vyžadován maximální zisk, optimistický přístup vede rozhodovatele k určení alternativy s největším ziskem. V rámci tohoto přístupu je snaha dosáhnout co největšího zisku.

Pro maximalizační problémy je tento přístup často nazýván jako **maximax** přístup. Pro minimalizační problémy je často využíván název **minimin**. (Anderson et al., 2013).

Při tomto principu rozhodovatel tedy najde nejlepší možný výnos pro každou alternativu a z těchto nalezených hodnot opět nejlepší. V případě kritéria výnosového typu hledá rozhodovatel maximum v řádcích a následně hledá maximum z vybraných hodnot. V případě minimalizace hledáme minima v řádcích a následně minima z vybraných hodnot.

Matematicky lze **maximax** zapsat:

$$max_{i\in\{1,2\ldots m\}}\ max_{j\in\{1,2\ldots n\}}\quad R_{ij}. \tag{1}$$

Matematicky lze **minimin** zapsat:

$$min_{i\in\{1,2\ldots m\}}\ min_{j\in\{1,2\ldots n\}}\quad R_{ij}. \tag{2}$$

Nicméně tento přístup absolutně nezohledňuje riziko a používá pouze nejlepší hodnoty.

2.3.2 Pesimistický přístup

Tento přístup vyhodnocuje každou alternativu z hlediska nejhoršího výnosu, který může nastat. Doporučená alternativa je tedy taková, která poskytuje nejlepší výplatu z nejhorších možných výplat. Pro problémy, kde je výstupním měřítkem zisk, pesimistický přístup doporučí rozhodovateli alternativu, která maximalizuje minimální možný zisk, kterého by bylo možné dosáhnout. Pro problémy minimalizační tento přístup identifikuje alternativu, která minimalizuje maximální možný náklad, který může nastat při zvolení dané alternativy. (Anderson et al., 2013)

Tento přístup se nazývá pesimistický, protože identifikuje nejhorší možné zisky a poté doporučí alternativu, která se vyhýbá možnosti extrémně špatných výplat. Tedy v rozhodovací matici, v případě maximalizace, rozhodovatel vybere v každém řádku nejmenší hodnotu a následně z vybraných hodnot se určí maximum.

Pro maximalizační problémy se tento přístup také jinak nazývá **maximin** přístup a pro minimalizační problémy **minimax**. (Anderson et al., 2013)

Matematicky lze tento přístup vyjádřit následujícím způsobem:

- **maximin**

$$max_{i\in\{1,2\ldots m\}}\ min_{j\in\{1,2\ldots n\}}\quad r_{ij,} \tag{3}$$

- **minimax**

$$min_{i\in\{1,2\ldots m\}}\ max_{j\in\{1,2\ldots n\}}\quad r_{ij}. \tag{4}$$

Tento postup je vhodný, když soupeříme proti racionálnímu protivníkovi. Nicméně dle Hillera & Liebermana, 2010 tento přístup není příliš často využíván ve hrách proti přírodě. Příroda není racionální protivník a rozhodovatel se nepotřebuje soustředit pouze na nejhorší alternativy. Proto je tento přístup vhodný pro velmi opatrné rozhodovatele.

2.3.3 Savageovo pravidlo

V rámci tohoto pravidla je počítána ztráta. Ztráta je rozdíl mezi ziskem spojeným s konkrétní alternativou a ziskem spojeným s rozhodnutím, které by zaručilo nejvíce žádoucí zisk za dané situace. Ztráta tedy znamená, kolik potenciálního zisku by se rozhodovatel zřekl zvolením konkrétní alternativy za dané situace. (Anderson et al., 2013)

Obecně ztrátu spočítáme následujícím způsobem: (Anderson et al., 2013)

$$R_{ij} = |V_j^* - V_{ij}|, \tag{5}$$

- R_{ij} je ztráta při alternativě a_i a situaci s_j,
- V_j*je zisk odpovídající nejlepšímu rozhodnutí za dané situace s_j,
- V_{ij} je zisk odpovídající alternativě a_i a situaci s_j.

Spočteme matici ztrát a prvky této matice určíme tak, že pro všechny existující situace určíme ztrátu, která by vznikla při zvolení jednotlivých alternativ ve srovnání s nejvýhodnější alternativou za určité situace. Následně aplikujeme na matici ztrát pesimistický přístup popsaný v kapitole 2.3.2. (Taha, 1997)

2.3.4 Laplaceovo pravidlo

Pokud nemáme k dispozici informace ohledně pravděpodobností jednotlivých situací, je rozumné předpokládat, že každá situace nastane se stejnou pravděpodobností. Lze tedy předpokládat, že pokud máme *n* situací, pravděpodobnost každé situace je rovna *1/n*. Tento přístup také navrhuje, že rozhodovatel počítá odpovídající zisk pro každou alternativu a zvolí alternativu s nejvyšší očekávanou hodnotou v případě maximalizace. (Pažek & Rozman, 2009)

2.3.5 Hurwitzovo pravidlo

Toto pravidlo se snaží najít střed mezi extrémy, které představují pesimistický a optimistický přístup. Na rozdíl od těchto dvou přístupů, kde se předpokládá absolutní pesimismus nebo optimismus, Hurwitzovo pravidlo kombinuje tyto dva přístupy tím, že přiřazuje konkrétní procenta optimismu a pesimismu. Tedy minimum a maximum každé alternativy by mělo být zprůměrováno za použití takzvaného indexu optimismu α. Tento index optimismu nabývá hodnoty v intervalu $\langle 0,1 \rangle$. (Pažek & Rozman, 2009)

Tento index optimismu zobrazuje přístup rozhodovatele k riziku. Hodně opatrný rozhodovatel zvolí $\alpha = 0$ a tím se toto pravidlo redukuje na pesimistický přístup. V opačném

případě, kdy rozhodovatel je velmi optimistický a zvolí α = 1, se toto pravidlo redukuje na optimistický přístup.

Postup aplikace tohoto pravidla je následující. Pro každou alternativu najdeme nejlepší výsledek a ten vynásobíme indexem α. K tomu přičteme nejhorší možný výsledek vynásobený hodnotou *(1–α)*. Následně porovnáme výsledky všech alternativ a zvolíme tu nejlepší. Postup lze shrnout následujícím způsobem. (Pažek & Rozman, 2009)

1. Zvolíme index α.
2. Pro každou alternativu A_i spočteme Hurwitzův průměr H následujícím způsobem:

- pro **pozitivní výnosy (zisky, příjem)**

$$H(A_i) = \alpha\,(\text{maximum z řádku}) + (1-\alpha)\,\text{minimum z řádku}, \quad (6)$$

- pro **negativní výnosy (náklady, ztráty)**

$$H(A_i) = \alpha\,(\text{minimum z řádku}) + (1-\alpha)\,\text{maximum z řádku}. \quad (7)$$

3. Zvolíme alternativu s nejlepší hodnotou H.

Nevýhoda tohoto pravidla je, že nebere v potaz jiné hodnoty, které jsou mezi maximem a minimem v dané alternativě. Důležité hodnoty jsou pouze maximum a minimum.

2.4 Rozhodování za rizika

Jedná se o rozhodování v situacích, ve kterých jsme dostali odhad pravděpodobností jednotlivých situací. Tento odhad může být založen na nějakých historických datech, expertní odhad atd. (Murthy, 2007)

V rámci rozhodování za rizika jsou k výpočtu využívány pravděpodobnosti jednotlivých situací a musí být dodržena následující pravidla:

$$pro\ všechna\ j = 1, 2 \ldots n\ platí\colon\ P(S_j) \geq 0, \quad (8)$$

$$\sum_{j=1}^{n} P(S_j) = P(S_1) + P(S_2) + \ldots + P(S_n) = 1. \quad (9)$$

Musí platit, že pravděpodobnosti jednotlivých situací musí být větší nebo rovna nule. Druhé pravidlo znázorňuje, že součet pravděpodobností všech situací musí být roven jedné.

2.4.1 Rozhodování za rizika bez experimentu

V rámci této kapitoly budou popsány přístupy k řešení problémů, u kterých již máme nějaké odhady pravděpodobností, ale již nebudeme dále provádět nějaké další experimenty (průzkumy atd.) k vylepšení našich odhadů pravděpodobností.

2.4.1.1 Princip maximální věrohodnosti

Tento přístup se soustředí pouze na nejvíce pravděpodobnou situaci. Identifikuje nejvíce pravděpodobnou situaci – tedy tu s nejvyšší pravděpodobností. Pro tuto situaci nalezneme alternativu s nejvyšším ziskem (nebo s nejmenšími náklady v případě minimalizace) a tuto zvolíme. (Hiller & Lieberman, 2010)

Hlavní myšlenka tohoto přístupu je, že nejvíce důležitá je situace, která je nejvíce pravděpodobná a zvolená alternativa je nejlepší pro danou důležitou situaci. Tento přístup nespoléhá na nejisté subjektivní odhady pravděpodobností situací – důležité je identifikovat nejvíce pravděpodobnou situaci.

Dle Hillera & Liebermana, 2010 nevýhoda tohoto přístupu je, že kompletně ignoruje ostatní relevantní informace – neberou se v potaz žádné jiné situace než ta nejvíce pravděpodobná. Pokud řešíme problém, kde je velké množství situací, pravděpodobnost nejvýznamnější situace může být nízká a celkový výsledek nemusí být zaručený.

2.4.1.2 Pravidlo očekávané střední hodnoty (Bayesovo rozhodovací pravidlo)

Tento přístup používá nejlepší dostupné odhady pravděpodobností jednotlivých situací a počítá střední hodnotu zisku pro každou dostupnou alternativu. (Murthy, 2007)

Očekávaná střední hodnotu (EV) alternativy A_i je definována následujícím způsobem:

$$EV(A_i) = \sum_{j=1}^{n} P(S_j) * V_{ij}, \tag{10}$$

kde $P(S_j)$ je pravděpodobnost, že nastane situace S_j a V_{ij} je zisk odpovídající alternativě A_i a situaci S_j.

Dle Hillera & Liebermana, 2010 je velkou výhodou tohoto přístup, že zahrnuje všechny dostupné informace – tedy všechny zisky a všechny nejlepší dostupné pravděpodobnosti. Nevýhodou tohoto přístupu je, že odhady pravděpodobností jsou do značné míry subjektivní. Neexistuje přesný způsob, jak predikovat budoucnost a pravděpodobnosti situací.

2.4.1.3 Pravidlo očekávané střední hodnoty a rozptylu

V rámci pravidla popsaného v kapitole 2.4.2 jsme se rozhodovali dle očekávaného zisku, nicméně důležitou roli v rámci rozhodování má i očekávané riziko. Míru rizika lze vyjádřit pomocí rozptylu nebo směrodatné odchylky. Rozhodujeme se tedy nejenom podle očekávané střední hodnoty (EV), ale i podle rozptylu nebo směrodatné odchylky.

$$var\ V(A_i) = \sum_{j=1}^{n} (r_{ij} - EV_i)^2 * p_j \quad (6)$$

- var V (A_i) je rozptyl výnosu i-té alternativy,
- zisk při volbě i-té alternativy a nastalé situace j,
- p_j je pravděpodobnost j-té situace.

Za méně rizikovou alternativu se považuje taková, která má menší rozptyl. Dle tohoto pravidla preferuje rozhodovatel alternativu, která nabývá lepších hodnot očekávané střední hodnoty a rozptylu, nebo která nabývá lepších hodnot z jednoho aspektu a z druhého stejně. (Klicnarová, 2021)

2.4.1.4 Pesimistický přístup

Za předpokladu, že máme k dispozici pravděpodobnosti jednotlivých situací, kterým ale nedůvěřujeme nebo nechceme riskovat, můžeme využít pesimistický přístup, který je popsán v kapitole 2.3.2.

2.4.1.5 Průměrný užitek

Toto pravidlo vychází z propočtu očekávané utility rozhodovacího kritéria u každé alternativy. Je zde brán v potaz specifický postoj rozhodovatele k riziku, který je vyjádřen pomocí užitkové funkce. (Richter, 2019)

Matematicky lze toto pravidlo vyjádřit následujícím způsobem:

$$E\ u(A_i) = \sum_{j=1}^{n} u(A_{(ij)}) * p_j, \quad (7)$$

- E u(A_i) je očekávaný užitek kritéria (např zisku),
- u($A_{(ij)}$) je užitek kritéria při j-té situaci a i-té alternativě
- p_j je pravděpodobnost j-té situace.

2.5 Rozhodovací stromy

Rozhodovací strom je nástroj z teorie grafů. Umožnuje vizuální zobrazení problému a je zejména užitečný, pokud musí být provedena sekvence rozhodnutí. Můžeme si graficky znázornit možnosti a důsledky rozhodnutí v jednotlivých etapách a díky tomu může být rozhodovací problém více přehledný. (Hiller & Lieberman, 2010)

Rozhodovací strom je strom z teorie grafů. Strom je souvislý orientovaný graf. Obecně platí, že grafy jsou tvořeny uzly a hranami.

V rozhodovacím uzlu jsou uzly jednotlivé možnosti volby a hrany jednotlivé alternativy nebo situace. Z každého uzlu vychází právě tolik hran, kolik alternativ má daná možnost volby nebo kolik situací může nastat. (Friebelová & Klicnarová, 2007)

V rozhodovacím stromě může být uzel rozhodovací nebo situační. Uzel rozhodovací je reprezentován čtvercem a indikuje, že rozhodnutí musí být provedeno v tomto bodě procesu. Jedná se tedy o uzel, ve kterém rozhodovatel volí další hranu. Zatímco situační uzel je reprezentován kružnicí, indikuje, že náhodná situace může nastat v tomto bodě a rozhodovatel na to nemá vliv. (Hillier & Lieberman, 2010)

Dle Hillera & Liebermanna, 2010 lze nakreslený rozhodovací strom vyhodnotit následujícím způsobem:

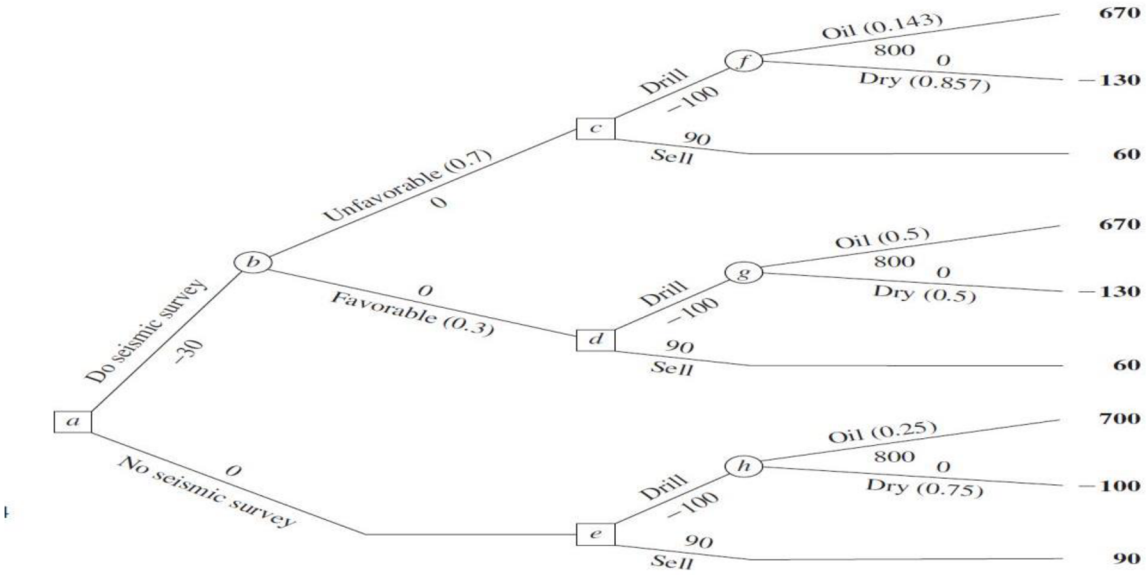
1. začneme od konce (zprava) rozhodovacího stromu a pohybujeme se vlevo po jednom sloupci. V každém sloupci, provedeme krok 2 nebo krok 3 v závislost na typu rozhodovacího uzlu ve sloupci,
2. pro každý situační uzel spočteme očekávaný zisk vynásobením očekávaným ziskem každé větve s pravděpodobností dané větve. Následně provedeme součet těchto produktů. Zaznamenáme očekávaný zisk pro každý rozhodovací uzel vedle uzlu a označíme toto množství jako očekávaný zisk pro hranu vedoucí do tohoto uzlu,
3. pro každý rozhodovací uzel porovnáme očekávané zisky jejich hran a zvolíme alternativu, jejíchž hrana má největší očekávaný zisk. V každém případě zaznamenáme volbu v rozhodovacím stromě vložením dvojitého přeškrtnutí pro zobrazení všech zamítnutých hran.

Procedura probíhá odzadu, nicméně po dokončení stromu tímto způsobem může rozhodovatel číst strom z levé strany do pravé a může tedy vidět skutečný vývoj události. Přeškrtnuté hrany značí nežádoucí cesty. Následováním otevřených cest z levé strany do pravé se získá optimální strategie.

Hlavní výhody rozhodovacích stromů jsou jejich univerzálnost, grafická názornost a možnost různých pokusů s pravděpodobnostmi jednotlivých situačních variant a s různými proměnnými, které mohou mít vliv na hodnoty rozhodovacího kritéria. Rozhodovací strom lze využít i pro znázornění jednoetapových rozhodovacích procesů. (Friebelová & Klicnarová, 2007)

Na obrázku číslo 2 lze vidět příklad rozhodovacího stromu, kde se firma rozhoduje o provedení seismického průzkumu.

Obrázek 2: Příklad rozhodovacího stromu



Zdroj: (Hillier & Lieberman, 2010)

3. Teorie užitku

V rámci kapitoly 2 jsme předpokládali, že očekávaný zisk v penězích je vhodným měřítkem. Nicméně ne vždy je tento předpoklad vhodný. Společnost nemusí být ochotná investovat velkou sumu peněz v například nějaký nový produkt, i když očekávaný zisk je značný, protože existuje riziko ztráty investice, které může způsobit zkrachování společnosti. (Hillier & Lieberman, 2010). Dále lidé kupují pojištění i když se jedná o nevýhodnou investici z pohledu očekávaného zisku.

Někdy je tedy potřeba do hodnocení zavést nějaký subjektivní pohled rozhodovatele. Mohou nás tedy zajímat různé faktory – finance, situace rozhodovatele a další. Zavedením užitkové funkce můžeme zavést subjektivní pohled do hodnocení. Zajímá nás, jaký užitek má daný rozhodovatel z kombinace volby alternativy a nastalé situace. Existuje tedy způsob, jak transformovat peněžní hodnoty na vhodnější měřítko, které bere v potaz preference rozhodovatele. Toto měřítko se nazývá užitková funkce peněz.

Dle Friebelové a Klicnarové (2007) lze tyto problémy popsat obecně následujícím způsobem: rozhodovatel volí z určitého množství různých variant, které se odlišují v jednom kritériu. Rozhodovatel stanoví přínos jednotlivých variant. Analytik na základě těchto ohodnocení určí, která varianta je pro rozhodovatele optimální.

3.1 Význam užitku

Užitek je měřítkem celkové hodnoty nebo relativní vhodnosti konkrétního výsledku. Bere v potaz postoj rozhodovatele k většímu množství faktorů jako například profit, ztráta a risk.

Vědci zjistili, že dokud peněžní hodnota výplat zůstává v rozsahu, které rozhodovatel považuje za rozumné, výběr alternativy s největší očekávanou peněžní hodnotou většinou vede k zvolení nejvíce preferovaného rozhodnutí. Nicméně pokud výplaty jsou extrémní, rozhodovatelé jsou často neuspokojeni s rozhodnutím, které jednoduše zaručuje největší očekávanou peněžní hodnotu. (Anderson et al., 2013)

3.2 Preference

Pokud řekneme, že rozhodovatel preferuje variantu A před B, myslíme tím, že rozhodovatel považuje variantu A více žádoucí nebo vhodnější než variantu B. Preference tedy porovnávají varianty.

Pro vyjádření preferencí se využívá několik různých značení: (Steele & Stefánsson, 2020)

- znak $\preceq$ reprezentuje slabou preferenční relaci. Tedy A $\preceq$ B znamená, že rozhodovatel považuje variantu B minimálně stejně vhodnou jako variantu A,
- znak $\prec$ značí ostrou preferenční relaci, který vyjadřuje, že rozhodovatel preferuje variantu a před variantou B,
- znak $\sim$ reprezentuje indiferenční relaci. Toto značení lze použít za předpokladu, že máme dvě alternativy, které jsou pro rozhodovatele indiferentní – tedy rozhodovatel nedává přednost žádné z těchto variant.

Matematicky se výše popsané vztahy nazývají relace. Dle Friebelové & Klicnarové (2007) relací na množině A x A rozumíme každou množinu uspořádaných dvojic prvků z této množiny.

Racionálně a dobře zadaná preferenční relace by měla splňovat následující vlastnosti. Předpokládáme, že S je množina variant. (Steele & Stefánsson, 2020)

$$Pro\ jakékoli\ A, B\ \in S\text{: buď A}\preceq\text{B nebo B}\preceq\text{A} \tag{13}$$

$$Pro\ A, B, C\ \in S\text{: } pokud\ \text{A} \preceq \text{B a B} \preceq \text{C},\ \text{potom A} \preceq \text{C} \tag{14}$$

Podmínka (13) zobrazuje podmínku pro úplnost relace. Relace je úplná za předpokladu, že rozhodovatel je schopen porovnat každé dva prvky množiny S.

Podmínka (14) popisuje podmínku pro tranzitivitu relace. Musí tedy platit, že pokud rozhodovatel slabě preferuje variantu B před variantou A a varianta C je slabě preferována před variantou B, potom varianta C je slabě preferována před variantou A.

Dle Friebelové & Klicnarové (2007) požadavek na tranzitivitu relace zajišťuje, že preference zadané rozhodovatelem jsou jednoznačně promyšlené. Například může dojít k porušení tranzitivity za předpokladu, že rozhodovatel má své preference zaměnitelné.

3.3 Užitková funkce za jistoty

Existují dva základní typy užitkové funkce – ordinální užitková funkce a více informativní kardinální užitková funkce.

3.3.1 Ordinální užitková funkce

Pokud množina variant S je konečná, tak jakékoli slabé uspořádání variant v množině S může být reprezentováno ordinální užitkovou funkcí. Platí tedy, že pokud *u* je užitková funkce s doménou *S*, tak říkáme, že funkce *u* reprezentuje preferenci $\preceq$ mezi variantami v

množině S. Matematicky to můžeme vyjádřit následujícím způsobem: (Steele & Stefánsson, 2020)

$$Pro\ každé\ A, B\ \in S : u(A) \leq \mathrm{u}(B) \Leftrightarrow \mathrm{A} \leq \mathrm{B}. \quad (15)$$

Máme tedy k dispozici množinu variant S a chceme pro tuto množinu sestrojit ordinální užitkovou funkci. Postup je, že musíme každé variantě v množině S přiřadit takové číslo, aby více preferovaná varianta měla vyšší hodnotu a méně preferovaná varianta nižší hodnotu.

Jediná informace, kterou máme k dispozici v ordinální užitkové funkci je, jak rozhodovatel, jehož preference jsou reprezentovány, seřadí varianty – tedy od nejméně vhodných až po nejvíce preferované.

Ordinální užitkové funkce nejsou vhodné pro matematické výpočty. Nezáleží zde vůbec na číselných hodnotách, ale pouze jen na pořadí. A z tohoto důvodu nelze s ordinální užitkovou funkcí počítat matematické operace jako průměry. (Steele & Stefánsson, 2020)

3.3.2 Kardinální užitková funkce

Při konstrukci kardinální užitkové funkce nás nezajímá pouze, jak rozhodovatel seřadil jednotlivé varianty, ale také jsou důležité rozdíly mezi jednotlivými variantami. (Steele & Stefánsson, 2020)

V tomto případě předpokládáme, že jsme schopni užitek měřit a nejenom seřadit preference. Pokud máme k dispozici varianty A, B, C jsme schopni na základě kardinální užitkové funkce zjistit, o kolik rozhodovatel preferuje například variantu C před variantou B a může to porovnat například s tím, jak moc je například varianta B preferována před A.

Například může jít o užitek z navštívení určitých míst – New York, Praha, Brno. Následně můžeme zjistit, které město navštěvuje rozhodovatel nejraději a jak moc ho preferuje před ostatními. Jak velký rozdíl je na druhé město v pořadí? Za předpokladu, že by byl velký rozdíl, tak víme, že rozhodovatel opravdu velmi rád navštěvuje město New York a ačkoli je například Praha druhá v pořadí, tak pro něj bude velký rozdíl, zda pojede do Prahy nebo New Yorku. Pokud by rozdíl byl marginální, nebude pro něj tak obrovský rozdíl, zda navštíví New York nebo Prahu (ačkoli stále nepatrně více preferuje New York).

Platí, že pokud máme dostatečné informace k sestrojení kardinální užitkové funkce, tak na základě těchto informací můžeme sestrojit i ordinální užitkovou funkci. Nicméně dle Friebelové & Klicnarové (2007) to opačně neplatí. Kardinální užitková funkce je vhodná pro matematické výpočty, protože poskytuje dostatečné množství informací.

3.3.3 Celkový a mezní užitek

Dle Šetka (2018) je celkový užitek takový užitek, který realizujeme ze spotřeby všech statků. Za předpokladu, že nás zajímá, jaký je užitek z konzumace například pěti kusů jablek dohromady, tak je to problém celkového užitku. Celkový užitek nenaznačuje vůbec nic o tom, jaký užitek má dotyčný člověk ze spotřeby každého kusu.

Zatímco mezní užitek (MU) vyjadřuje změnu celkového užitku při jednotkové změně vstupu. Mezní užitek je matematicky derivací celkového užitku tam, kde derivace existuje. (Friebelová & Klicnarová, 2007)

$$MU(x) = u'(x) \tag{16}$$

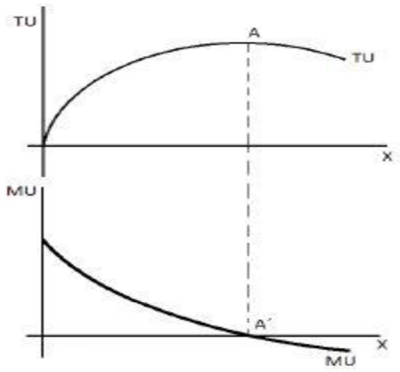
Tabulka 2: Příklad úplného a mezního užitku

X	TU	MU
0	0	
		10
1	10	
		7
2	17	
		5
3	22	

Zdroj: (Šetek, 2018)

Tabulka číslo 2 popisuje příklad úplného a mezního užitku. Předpokládejme, že X je spotřebované množství statku, TU je celkový užitek a MU je mezní užitek. Z tabulky číslo 2 je patrné, že mezní užitek klesá s rostoucím množství spotřebovávaného statku. Tomuto faktu se říká zákon klesající mezního užitku. Dle Šetka (2018) se tento zákon dá zdůvodnit tak, že pokud naše spotřeba určitého statku pořád roste, tak užitek z následující jednotky je vždy menší.

Obrázek 3: Vztah celkového a mezního užitku



Zdroj: (Šetek, 2018)

Na obrázku číslo 3 lze vidět vztah celkového a mezního užitku. Za předpokladu, že mezní užitek klesá, tak celkový užitek roste, ale s klesajícím rychlostí. Za předpokladu, že mezní užitek je roven nule, tak celkový užitek dosahuje maxima. Bod A je nazýván jako bod nasycení. Od bodu A´ je mezní užitek negativní a celkový užitek klesá. (Šetek, 2018)

3.4 Užitková funkce peněz

V rámci kapitoly 3.3 nás zajímala užitková funkce, která brala v potaz pouze množství dané komodity. Nicméně často platí, že s větší kvantitou komodity roste riziko s tím související. (Friebelová & Klicnarová, 2007).

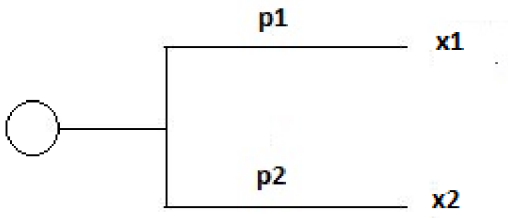
Stejně jako je konkávní obecná užitková funkce, to samé platí pro užitkovou funkci peněz většiny rozhodovatelů, respektive těch, kteří mají averzi k riziku. Tvar lze určit s využitím jistotních ekvivalentů. (Fotr, 2006)

3.4.1 Loterie, jistotní ekvivalent a riziková prémie

V rámci užitkové funkce za rizika je důležitý pojem loterie. Loterie je situace, kdy osoba obdrží odměnu r s pravděpodobností p. Loterie je často reprezentována jako strom, ve kterém každá větev obsahuje možný výsledek loterie a číslo na každé větvi reprezentuje pravděpodobnost, že daný výsledek nastane. (Winston, 1994)

Příklad takové loterie ilustruje obrázek číslo 4, kde osoba obdrží odměnu x_1 s pravděpodobností p_1 a odměnu x_2 s pravděpodobností p_2.

Obrázek 4: Loterie



Zdroj: vlastní

Dle Winstona (1994) je **jistotní ekvivalent** nějaké loterie L (dále značeno jako $\hat{x}$) takové číslo $\hat{x}$, kdy platí, že rozhodovatel je indiferentní mezi loterií L a výplatou $\hat{x}$. Rozhodovatel si tedy cení hodnoty rovné jistotnímu ekvivalentu stejně vysoko jako rizikové variantě. Dle Friebelové & Klicnarové (2007) lze jistotní ekvivalent definovat tak, že zvažujeme nějakou situaci, kde můžeme získat určitou kvantitu nějaké komodity – $x_1 \ldots x_k$ a každá kvantita komodity má danou pravděpodobnost $p_1 \ldots p_k$. Jistotním ekvivalentem k této situaci je takové množství komodity ($\hat{x}$), kdy musí platit, že užitek z tohoto množství komodity je ekvivalentní střední hodnotě užitku situace. Platí tedy následující vztah:

$$u(\hat{x}) = \sum_{i=1}^{n} p_i * u(x_i), \tag{8}$$

kde $\hat{x}$ je jistotní ekvivalent, $u(\hat{x})$ je užitek jistotního ekvivalentu, $u(x_i)$ je užitek důsledku velikosti x_i.

Například pokud budeme vlastnit lístek do tomboly, který nás stál 100 Kč. Na základě tohoto lístku do tomboly můžeme získat nějaký zisk. Minimální cena, za kterou jsme tento lístek do tomboly ochotni prodat případnému zájemci, je naším jistotním ekvivalentem. Je to tedy minimální cena, která je pro nás stejně výhodná jako hraní případné rizikové hry.

Rozdíl mezi očekávanou hodnotou loterie a garantovaným ziskem je nazýván **rizikovou prémií**, kterou je rozhodovatel ochoten zaplatit. Je možné ji spočítat dle následující vztahu: (Anderson et al., 2013)

$$RP = E(X) - \hat{x}, \tag{9}$$

kde RP je rizikovou prémií, EX je střední hodnota výnosu rizikového projektu a $\hat{x}$ je jistotní ekvivalent.

3.4.2 Postoj rozhodovatele k riziku

Postoj rozhodovatele k riziku patří mezi velmi důležité pojmy z teorie rozhodování za rizika a nejistoty. Popisuje chování rozhodovatele ve stavu, kdy rozhodovatel volí mezi dvěma variantami, které by měly být z hlediska potenciálu podobně výnosné, ale rozlišují se rizikem. (Fotr, 2006)

Rozhodovatel může mít averzi k riziku, sklon k riziku, nebo může být k riziku neutrální. Dle Friebelové & Klicnarové (2007) můžeme porovnat jistotní ekvivalent ($\hat{x}$) se střední hodnotou hry (E(X)).

U rozhodovatele s **averzí k riziku** lze konstatovat, že se spokojí se zaručenou sumou, která je menší než očekávaná střední hodnota hry. Lze také konstatovat, že rozhodovatel s averzí k riziku preferuje jednoznačně varianty, které jsou méně rizikové a takové, které mu jsou schopné zaručit dostatečné zisky s vysokou pravděpodobností. (Klicnarová, 2021) Rovnice (19) znázorňuje, že jistotní ekvivalent rozhodovatele s averzí k riziku je menší než očekávaná střední hodnota hry.

$$\hat{x} < E(X) \tag{19}$$

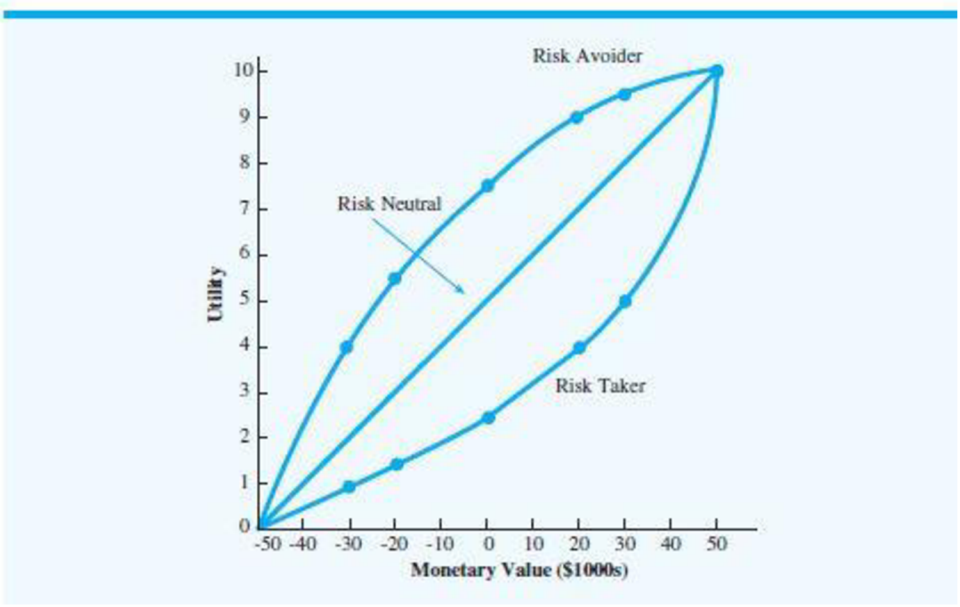
U **rozhodovatele se sklonem k riziku** lze tvrdit, že favorizuje hraní loterie před výdělkem, který je roven očekávané střední hodnoty hry, případně by musela být zaručená suma vyšší než očekává střední hodnota hry. (Klicnarová, 2021). Tento typ rozhodovatele zcela jistě preferuje varianty, které jsou velmi rizikové – tedy takové, které nabízí velmi vysoký zisk, ale tyto varianty jsou asociované s vyšší pravděpodobností ztrát. Rovnice (20) znázorňuje, že jistotní ekvivalent rozhodovatele se sklonem k riziku je větší než očekávaná střední hodnota hry.

$$\hat{x} > E(X) \tag{20}$$

U rozhodovatele, který **je neutrální k riziku** lze konstatovat, že mu nezáleží na tom, zda hraje loterii nebo dostane sumu, která odpovídá střední hodnotě hry. (Klicnarová, 2021) Dle Fotra (2006) u tohoto typu rozhodovatele je rovnováha mezi averzí k riziku a sklonu k riziku. Rovnice (21) znázorňuje, že jistotní ekvivalent rozhodovatele neutrálního k riziku je roven očekávané střední hodnotě hry.

$$\hat{x} = E(X) \tag{21}$$

Obrázek 5: Užitková funkce pro různé typy přístupu rozhodovatele k riziku



Zdroj: (Anderson et al., 2013)

Na obrázku číslo 4 lze vidět graf zobrazující různé užitkové funkce na základě různých přístupů rozhodovatele k riziku. Je evidentní, že užitková funkce pro rozhodovatele s averzí k riziku má tvar konkávní, pro rozhodovatele s averzí k riziku je konvexní a pro rozhodovatele, který je neutrální k riziku je lineární. Na ose X je zisk a na ose Y se nachází užitek.

Dle Fotra (2006) byly provedeny výzkumy chování rozhodovatelů za rizika, které prokazují, že mnohem více je rozhodovatelů s averzí k riziku. Nicméně se také prokázalo, že postoje rozhodovatele k riziku nemusí být stálý – záleží, zda jsou ve hře zisky nebo ztráty. Pokud jde o zisky, převládá averze k riziku, zatímco u ztrát je převažující sklon k riziku.

Na postoj rozhodovatele k riziku může působit hned několik dalších faktorů. Dle Fotra (2006) mezi nejdůležitější patří:

- osobní založení rozhodovatele,
- minulé zkušenosti,
- okolí, ve kterém je konán proces volby rizikových variant.

3.4.3 Konstrukce užitkové funkce peněz

Užitková funkce hraje velmi důležitou roli v rámci procesu rozhodování. Aby analytik mohl případně někomu poradit s rozhodnutím, potřebuje k tomu znát užitkovou funkci dotyčného. Křivka užitkové funkce může být různých tvarů – konkávní, konvexní a lineární. Záleží na postoji rozhodovatele k riziku.

Pokud rozhodovatel je neutrální k riziku, nemusíme konstruovat užitkovou funkci vůbec, neboť užitková funkce rozhodovatele, který je neutrální k riziku, je lineární. Pokud má rozhodovatel sklon k riziku, tak bude hledat variantu, která zaručuje nejvyšší očekávaný výnos s velkým rizikem.

Jak již bylo poukázáno v rámci kapitoly 3.4.2, největší zastoupení mají rozhodovatelé s averzí k riziku.

Dle Hilliera & Liebermana (2010) je také možné, aby existovala kombinace těchto popsaných typů chování. Například rozhodovatel může být neutrální k riziku s malým množstvím peněz, se středním množstvím peněz mít sklon k riziku a s velkým množstvím peněz může mít averzi k riziku. Navíc postoje rozhodovatele k riziku se mohou měnit v čase v závislosti na různých okolnostech.

Fakt, že rozdílní lidé mají rozdílné užitkové funkce peněz má důležitý důsledek pro rozhodování za rizika. Pokud užitková funkce inkorporovaná do rozhodování nějakého problému, užitková funkce musí být konstruována, aby odpovídala preferencím a hodnotám rozhodovatele. Rozhodoval navíc může být jeden nebo skupina lidí.

Pro konstrukci užitkové funkce peněz se využívá princip loterie a jistotního ekvivalentu, které byly popsané v kapitole 3.4.1. Existují dva základní typy otázek, které jsou rozhodovateli kladeny, pomocí nichž jsme schopni získat nějaké hodnoty užitkové funkce rozhodovatele v některých bodech.

V prvním kroku konstrukce užitkové funkce je nutné stanovit rozsah. Dle Hilliera & Liebermana (2010) je rozsah užitkové funkce irelevantní. Nicméně je doporučeno přiřadit situaci generující nejmenší hodnotu užitek rovný 0 – tedy u(minimum)=0. Situace, která generuje největší hodnotu přiřadíme užitek rovný 1 – tedy u(maximum)=1. Pokud víme, že rozhodovatel je neutrální k riziku, stačí pouze spojit dva nalezené body spojit a máme užitkovou funkci zkonstruovanou.

3.4.3.1 Postup při konstrukci užitkové funkce peněz

Pokud neznáme postoj rozhodovatele k riziku nebo pokud víme, že není neutrální k riziku, tak musíme získat nějaké hodnoty užitkové funkce rozhodovatele v některých bodech. K tomu využijeme sérii otázek, pomocí nichž tyto hodnoty můžeme získat. Jeden typ otázek funguje na principu nastavování různých hodnot pravděpodobností p_i nebo různou volbou výchozích bodů. Způsob konstrukce prvního typu otázky je následující.

1. Jsou vybrány dva náhodné body z množiny známých bodů.
2. Volíme libovolnou hodnotu p, která musí být v intervalu <0,1>. Proměnná p reprezentuje hodnotu pravděpodobnosti, která je použita pro tvorbu otázek a k výpočtu požadovaného nového bodu.
3. Na základě vybraných dvou bodů (dále značeno jako [Z_1, U_1] a [Z_2, U_2]) a zvolené hodnoty p je položena rozhodovateli otázka.
   3.1 Pro tento typ otázky je zapotřebí vytvořit dvě alternativy na základě výše zmíněných bodů a hodnoty p.
   A_1: S pravděpodobností p můžete získat zisk v podobě Z_1.
   A_2: S pravděpodobností $(1–p)$ můžete získat zisk v podobně Z_2.
   3.2 Rozhodovateli jsou představeny vytvořené alternativy a je položena otázka na jistotní ekvivalent rozhodovatele v situaci, kdy hypoteticky má na výběr mezi těmito dvěma alternativami A_1 a A_2. Tuto otázku je možné také přeformulovat tak, že rozhodovateli je položena otázka: Jakou cenu jste ochoten za tuto hru zaplatit?
4. Odpověď rozhodovatele (dále Z) je využita pro výpočet nového bodu. Jelikož je v tuto chvíli známá hodnota nového hledaného bodu na ose X, je nutné dopočítat odpovídající hodnotu na ose Y. Je nutné vypočítat odpovídající užitek ze získaného zisku Z, který byl získán na základě odpovědi rozhodovatele. Výpočet odpovídající hodnoty užitku pro zisk Z je popsán v rovnici (22).

$$u(Z) = p * u(Z_1) * (1 - p) * u(Z_2) \quad (22)$$

- $u(Z)$ je námi hledaný užitek k zisku Z,
- p je námi zvolená pravděpodobnost,
- Z_1, Z_2 jsou zisky námi zvolených dvou bodů.

Rozhodovatel svou odpovědí prohlašuje, že užitek z jeho odpovědi je stejný jako střední užitek z výše popsaného stavu, kdy s pravděpodobností p získá zisk Z_1 a s pravděpodobností $(1 - p)$ získá zisk Z_2.

V rámci druhého typu otázek postupujeme je struktura otázek jiná. K získání užitku dalších potencionálních zisků Z je postup následující. (Hillier & Lieberman, 2010)

1. Vybereme dva jakékoli body z množiny známých bodů a zvolíme zisk *Z*, ke kterému bude dopočítávána odpovídající hodnota užitku.
2. Na základě vybraných dvou bodů (dále značeno jako [Z_1, U_1] a [Z_2, U_2]) a zisku *Z* je položena rozhodovateli otázka.

2.1 Pro tento typ otázky je zapotřebí vytvořit dvě alternativy na základě výše zmíněných bodů a vybraného zisku *Z*.

A_1: Získejte zisk rovný Z_1 s pravděpodobností p a zisk rovný Z_2 s pravděpodobností $(1–p)$.

A_2: S jistotou získejte zisk *Z*.

2.2 Otázka pro rozhodovatele nyní je, jaká je jeho hodnota *p*, která ho dostává do situace, kdy je indiferentní mezi těmito dvěma alternativami. Výsledný užitek z částky *Z* je vypočítán na základě rovnice (22).

Nyní za předpokladu, že bylo nalezeno dostatečné množství bodů užitkové funkce, lze přikročit k dalšímu kroku a tím je proložení nalezených bodů užitkové funkce pomocí různých matematických metod.

Je známo, že rozhodovatelů s averzí k riziku velmi převládá, proto bude podrobně popsán postup k proložení užitkové funkce pro rozhodovatele s averzí k riziku.

Rovnice (23) znázorňuje, v jakém tvaru je užitková funkce peněz. (Klicnarová, 2021)

$$u = a * \exp\left(-\frac{x}{b}\right) + c \tag{23}$$

- x je zisk,
- a, b, c jsou parametry užitkové funkce. Platí, že parametr *b* musí být kladný, parametr *a* negativní.

Dle Klicnarové (2021) je cílem je nalézt takové hodnoty a, b, c, aby tato funkce nejlépe proložila naše nalezené body. Je zde řešen problém nelineární regrese a je používána Metoda nejmenších čtverců, která je založena na minimalizaci rovnice (24).

$$SSE = \sum_{i=1}^{N} (y_i - (a * \exp\left(-\frac{x_i}{b}\right) + c))^2 \tag{24}$$

- y_i je nalezená hodnota užitku ze zisku x_i,
- *N* je počet známých bodů, které jsme získali.

Tento problém lze řešit pomocí různých statistických softwarů. Například lze využít i Excel pomocí modulu Řešitel. Důležité je poznamenat, že úloha může mít vícero extrémů. Dle Klicnarové (2021) je nutné spustit software z většího množství výchozích bodů v případě modulu Řešitele v Excelu, jiný software to v sobě může mít zabudováno. Toho dosáhneme tak, že nastavíme různé startovací hodnoty pro proměnné a, b, c. Výsledky následně porovnáme podle výsledné funkce SSE a vybereme takovou hodnotu SSE (s korespondujícími hodnotami parametrů), která je minimální, nicméně nemáme zaručeno, že jsme nalezli optimální řešení.

3.4.4 Alternativní přístup ke konstrukci užitkové funkce

Dle Hilliera & Liebermana (2010) je velmi populární forma k odhadu užitkové funkce takzvaná exponenciální užitková funkce, která má následující tvar:

$$u(x) = R\left(1 - e^{\frac{-x}{R}}\right), \tag{25}$$

- R je tolerance k risku rozhodovatele,
- x je zisk.

Dle Hilliera & Liebermana (2010) je tato užitková funkce vhodná pro rozhodovatele s averzí k riziku. Velká averze k riziku koresponduje s malou hodnotou parametru R, zatímco velká hodnota R odpovídá malé averzi k riziku. Jinými slovy se jedná o parametr, který říká, jak moc rizika je rozhodovatel ochoten tolerovat. Dle Winstona (1994) je prokázáno, že bohatý rozhodovatel bude mít vysokou hodnotu R.

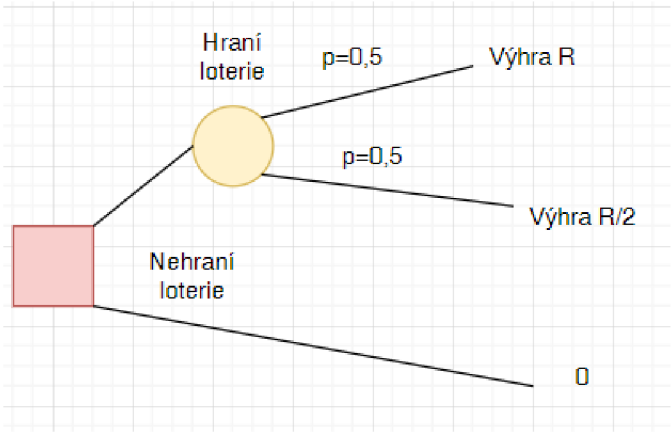
V situacích, kde důsledky potenciálních ztrát nejsou vážné pro rozhodovatele, lze předpokládat, že exponenciální užitková funkce může poskytnout rozumný odhad. Abychom odhadli vhodnou hodnotu parametru R, musí být rozhodovatel dotázán ohledně volby čísla R.

Rozhodovatele se dotážeme, aby zvolil číslo parametru R, které ho dostane do situace, že je indiferentní mezi následujícími dvěma alternativami: (Winston, 1994)

- A_1 je 50/50 loterie, kde by rozhodovatel získal R peněz s pravděpodobností 0,5 a ztratil R/2 peněz s pravděpodobností 0,5,
- A_2 je nezískal by nic, ale ani neztratil.

Celou tuto situaci lze také znázornit rozhodovacím stromem zobrazeným na obrázku číslo 6.

Obrázek 6: Rozhodovací strom pro určení R



Zdroj: vlastní

Pouze touto otázkou jsme schopni odhadnout parametr R daného rozhodovatele a tím celý proces konstrukce užitkové funkce je mnohem jednodušší a rychlejší než v rámci konstrukce užitkové funkce popsané v kapitole 3.4.3.

4. Metodika

Cílem praktické části je vytvoření aplikace, která umožní rozhodovateli zkonstruovat si vlastní užitkovou funkci peněz a následně ji využít v rozhodovacím procesu. V rámci demonstrace aplikace je záměrem zkonstruovat užitkové funkce dvou různých rozhodovatelů prostřednictvím navržené aplikace a vyřešit zvolené modelové příklady s využitím těchto zkonstruovaných užitkových funkcí a porovnat výsledná řešení s řešeními bez použití užitkové funkce.

Je předpokládáno, že užitek ze zisku rovného nula, je roven nule. Důvodem je, že aplikace je určená pouze pro konstrukci užitkové funkce peněz – nikoli užitkové funkce například teploty, kde všichni lidé rozhodně nemají nulový užitek při 0 °C. V rámci peněz je tento předpoklad smysluplný. Dále je aplikace určená pro nezáporné hodnoty zisků.

Prvním klíčovou volbou byla volba vhodného programovacího jazyka a grafického frameworku. Po důkladné analýze možných knihoven a jejich dostupných dokumentací byl zvolen programovací jazyk **Python**. Dále bylo nutné vybrat vhodné grafické rozhraní s podporou jazyka Python. Z možných variant byl nakonec vybrán grafický framework **PyQt5** z důvodu velmi intuitivní dokumentace a programu **Qt Designer**, který umožňuje snadno navrhovat grafická rozhraní. IDE pro vývoj této aplikace bylo použito **Visual Studio Code**.

Další klíčovou volbou byla volba vhodných matematických, statistických knihoven. Po důsledné analýze byla vybrána knihovna **scipy**, která je v aplikaci používána pro provedení nelineární Metody nejmenších čtverců. Společně s knihovnou **scipy** je také využívána knihovna **numpy**, která je užívána společně s knihovnou **scipy** pro výpočty. Tato knihovna je základní balíček pro vědecké výpočty v jazyce **Python**.

Velmi klíčovou volbou bylo zvolení knihovny umožňující vykreslování grafů, která zároveň je integrována do grafického frameworku PyQt5. Po zvážení všech alternativ byla zvolena knihovna **matplotlib**.

Pro popsání funkcionality aplikace byl zvolen jazyk UML a konkrétně use case a activity diagramy. Nástroj pro zakreslení diagramů byl zvolen **Microsoft Visio**.

Pro pomocné výpočty modelových příkladů byl použit nástroj **Microsoft Excel.**

5. Funkcionalita aplikace

V rámci této kapitoly bude podrobně popsána funkcionalita aplikace prostřednictvím jazyka UML. Konkrétně budou využity use case a activity diagramy.

5.1 Funkcionalita aplikace

Na obrázku číslo 7 lze vidět use case diagram, který popisuje funkcionalitu aplikace. Růžovou barvou je znázorněn aktor a tím je v tomto případě jen **Uživatel** a modře jsou znázorněny jednotlivé případy použití.

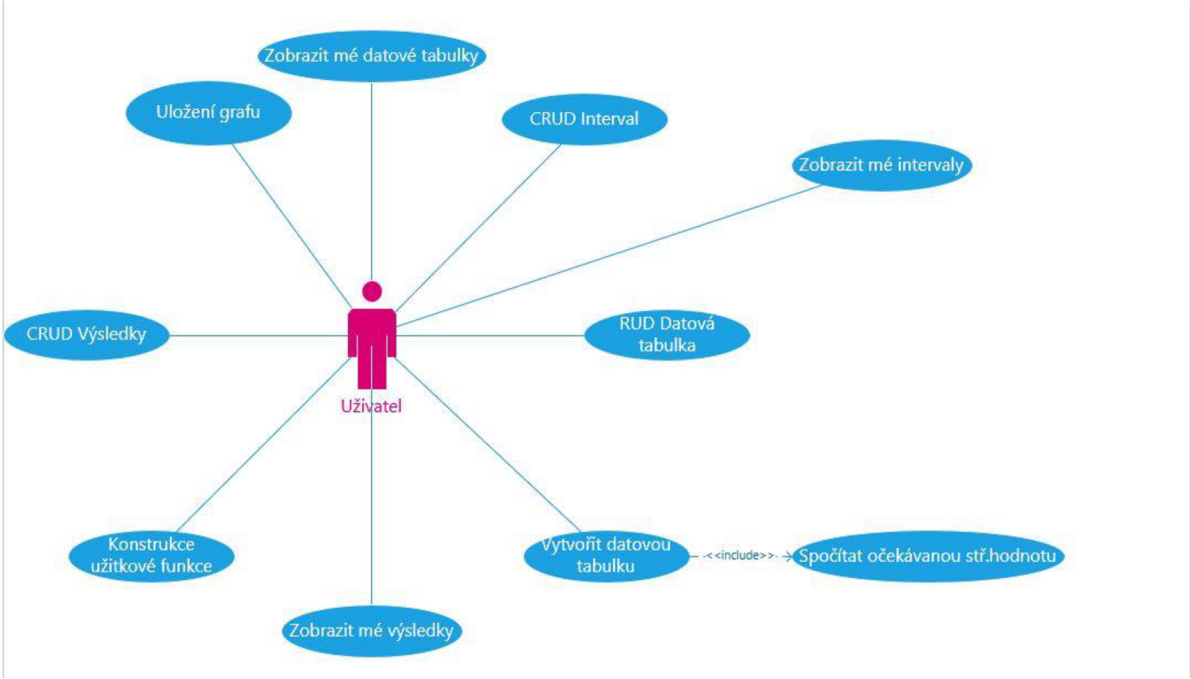
Uživatel může pro konstrukci užitkové funkce použít dva druhy dat – intervaly a datové tabulky. Může tyto dvě skupiny dat spravovat – mazat, číst, aktualizovat a vytvářet. Na základě intervalů nebo datových tabulek lze poté spustit proces konstrukce užitkové funkce peněz. Pro výpočty, které jsou nutné pro konstrukci užitkové funkce je nutné zadat minimálně jeden typ dat z výše zmíněných typů. Zároveň lze použít pro výpočet pouze jeden typ dat, nikoliv oba typy najednou. Lze tedy zkonstruovat užitkovou funkci na základě intervalu nebo tabulky, ale není možné použít pro konstrukci oba typy dat najednou.

Datové tabulky jsou tabulky, které obsahují dva sloupce – pravděpodobnost a odpovídající zisk. Datová tabulka může obsahovat libovolný počet řádků. U datových tabulek musí platit, že součet pravděpodobnosti ve sloupci „Pravděpodobnost“ musí být roven 1. Zároveň buňky ve sloupci „Pravděpodobnost“ mohou nabývat hodnot pouze v intervalu <0,1>. Buňky ve sloupci „Zisk“ mohou nabývat pouze kladných hodnot a nuly. Datové tabulky lze pojmenovat, měnit, mazat a číst z nich. Po vytvoření tabulky se počítá střední očekávaná hodnota dané datové tabulky.

Intervaly jsou druhým způsobem, jakým lze zadat vstupní data ke konstrukci užitkové funkce. Zde opět platí, že jsou akceptována pouze kladná čísla a nula. Intervaly lze opět

pojmenovat, změnit, mazat a číst.

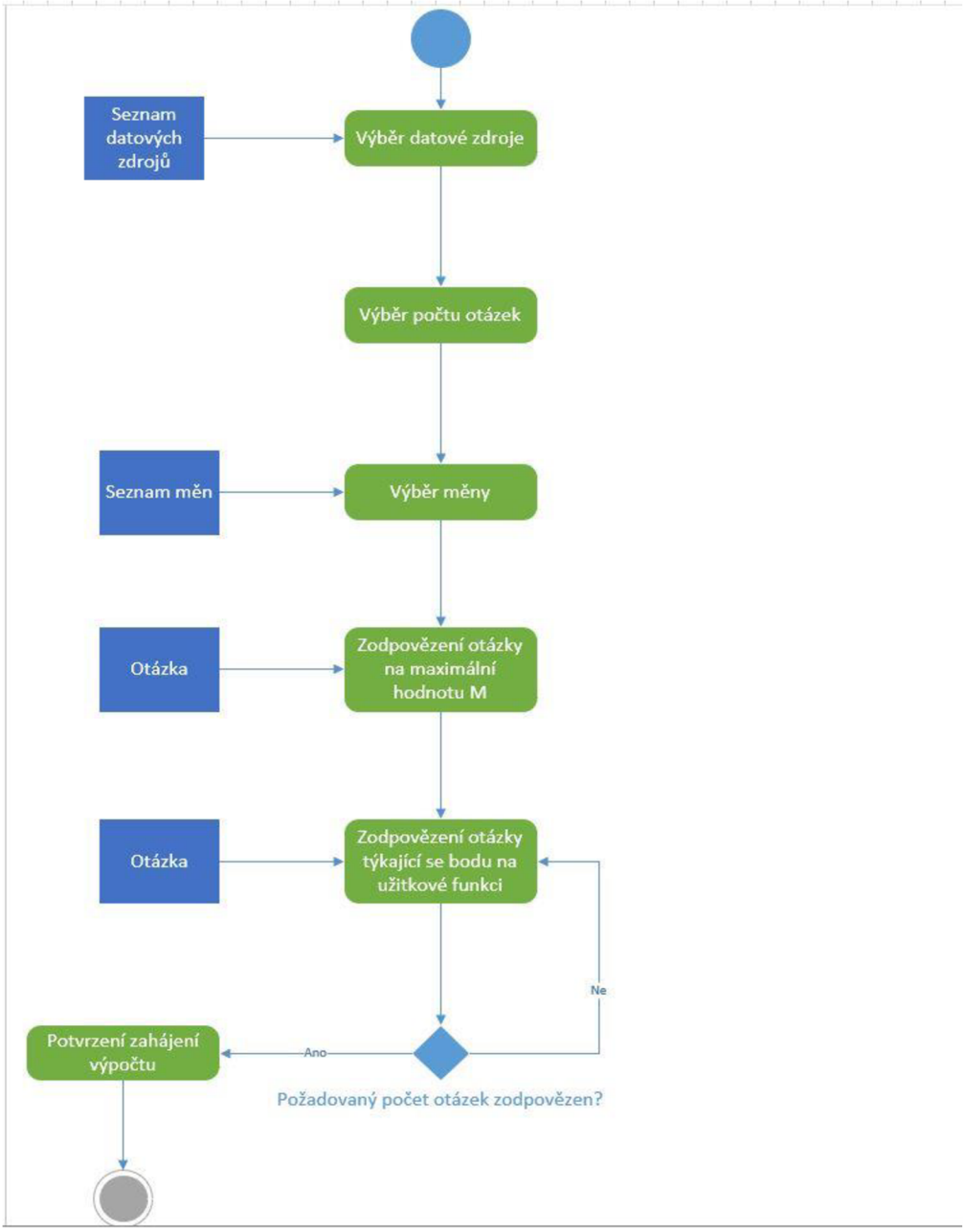
Obrázek 7: Use case diagram



Zdroj: vlastní

Nejdůležitější funkcionalitou aplikace je konstrukce užitkové funkce peněz. V tomto případě je nutné, aby uživatel měl již vytvořené nějaké datové tabulky nebo intervaly na základě kterých proběhne proces konstrukce užitkové funkce peněz. Vzhledem k tomu, že se jedná o aplikaci, jejíchž hlavní funkcionalitou je konstrukce užitkové funkce peněz, tak je stanoveno minimum na 0. Proces se skládá z volby parametrů konstrukce užitkové funkce – měna, počet otázek, výběr datového souboru a následně jsou otázky generovány uživateli. Po zodpovězení všech požadovaných otázek probíhá vyhledávání optimálních parametrů funkce, provedení nelineární regrese a zobrazení výsledků společně s grafem užitkové funkce peněz. Tato funkcionalita bude podrobně popsána v následujících kapitolách. Tento popsaný proces popisuje activity diagram na obrázku číslo 8.

Obrázek 8: Activity diagram – konstrukce užitkové funkce peněz



Zdroj: vlastní

Uživatel dále má možnost spravovat své výsledky. Může si je prohlížet, mazat, číst a může aktualizovat název. Zároveň uživatel má možnost uložit graf výsledné užitkové funkce ve formátu obrázku.

6. Vývoj aplikace

V rámci této kapitoly bude popsán vývoj aplikace. O běh hlavního vlákna se stará třída **MainView**, kterou spouští hlavní skript **main.py**, který se nachází nejvýše v hierarchii.

Aplikace obsahuje na levé straně menu, pomocí kterého lze volit položky. Kliknutí na položku v menu spouští korespondující akci. Jednotlivé položky menu mají přiřazené určité ID a po kliknutí na jakoukoli položku menu se dle nalezeného ID provede korespondující akce. Pro každé view jsou zachytávány různé eventy. Na základě zachycených eventů jsou volány konkrétní metody.

V horní části aplikace je možné najít menu s názvem „Pomoc“ a podmenu „Pomoc s používáním aplikace“, které obsahuje informace, jak aplikaci používat a jaké kroky jsou nutné udělat před samotnou konstrukcí užitkové funkce. Pomocníka lze kdykoliv vyvolat při stisknutí zkratky „CTRL+H“.

6.1 Struktura aplikace

Na obrázku číslo 10 lze vidět strukturu aplikace.

Obrázek 9: Struktura aplikace

Zdroj: vlastní

Aplikace je členěna do několika složek. Na absolutně nejvyšší hierarchické úrovni se nachází soubory **imgs.py** a **main.py**. **Imgs.py** je soubor, který umožňuje zobrazování obrázků v PyQt5 grafickém rozhraní. Soubor **main.py** je hlavním startovacím souborem a zajišťuje spouštění aplikace.

Dále je zde složka **model**, která obsahuje datové třídy. Tedy třídy, které slouží k uchovávání dat nebo pomáhají k zobrazování dat. V aplikaci jsou užívány následující datové třídy:

- **DataTable**, která je používána k ukládání datových tabulek, které uživatel vytvořil. Také je zde počítána očekávaná střední hodnota,
- soubor **enums.py** obsahuje všechny výčty, které jsou v aplikaci využívány. Používají se dva druhy výčtů – **ConstructionType** a **QuestionType**. **ConstructionType** rozeznává, zda konstrukce užitkové funkce bude na základě intervalu, nebo datové tabulky. **QuestionType** určuje, jaký bude typ vygenerované otázky. Podrobněji budou jednotlivé typy rozebrány v kapitolách řešící generování otázek a konstrukci užitkové funkce peněz,
- **Interval** je třída, která existuje k zaznamenání intervalu, který uživatel vytvořil. Obsahuje pouze proměnné pro uchovávání počáteční, koncové hodnoty a název,
- **OptimResult** je datová třída, která je používána k uchovávání nalezeného SSE společně s objektem, který obsahuje informace ohledně nalezeného výsledku. Tedy nalezené hodnoty parametrů funkce, počáteční hodnoty parametrů atd,
- **Question** je třída, jejíž účelem je ukládání informací ohledně vygenerované otázky. Také je zde počítán užitek po odpovědi uživatele na otázku. Tato třída bude vysvětlena v kapitole věnující se generování otázek,
- **QuestionGenerator** je třída, která je používána ke generování otázek pro uživatele. Existují různé typy otázek, které se náhodně generují. Bude podrobně vysvětlena v kapitole řešící generování otázek,
- **UtilityConstructor** je využívána k přenesení zvolených parametrů konstrukce užitkové funkce,
- **UtilityResult** je datová třída, která zajišťuje přenos výsledků regrese, SSE, nalezených bodů a dalších proměnných mezi jednotlivými okny aplikace,
- **CustomTableWidgetItem** je třída, která dědí od třídy *QTableWidgetItem*. Je používána pro vytváření buněk tabulky.

Dále je v hiearchii složka **views**, která obsahuje dva druhy souborů – grafické šablony a běžné python soubory. V této složce je další složka, která se nazývá **templates**. Složka **templates** obsahuje pouze soubory popisující grafické rozhraní. Jedná se soubory s koncovkou „ui“ vytvořené aplikací Qt Designer. Ve složce **views** jsou jednotlivé třídy, které se starají o nahrávání souborů ze složky **templates** a také se starají o veškerou logiku související s tímto grafickým rozhraním. Jedná se o následující třídy:

- třída **MainView** je zodpovědná za spouštění hlavního okna aplikace a funkcionalitu související s hlavním oknem aplikace,
- třída **QuestionUi** je třída, která se stará o logiku a načítání grafické šablony pro část aplikace, kde uživatel odpovídá na vygenerované otázky,
- třída **ConstructionDialog** je zodpovědná za část aplikace, která uživateli nabízí volbu parametrů pro konstrukci užitkové funkce peněz (měna, datová sada atd.) a předává informace dále,
- třída **InputDialog** je používána k vytváření generických oken s více vstupními poli. Používá se například pro tvorbu intervalů.

Dále je zde složka **utils**. Tato složka obsahuje soubory, které svou podstatou nenáleží ani do jedné ze zmíněných předchozích složek. Nachází se zde následující třídy:

- třída **ConstantManager**, která je užívána k uchovávání různých konstant a nastaveních. Obsahuje i nastavení pro regresi, barvy a další konfigurační parametry,
- třída **LeastSquareOptimalization** je hlavní třída, kde se skrývá funkcionalita týkající se všech klíčových výpočtů. Kromě samotné Metody nejmenších čtverců se zde nachází také metody, které zajišťují hledání vhodných počátečních hodnot parametrů pro nelineární regresi.

6.2 Datové objekty

V aplikaci se používají dva typy datových objektů – **Interval** a **DataTable**. V rámci této kapitoly budou vysvětleny nejpodstatnější části vývoje týkající se těchto datových objektů a grafického rozhraní týkající se těchto datových objektů.

6.2.1 Datové tabulky

Datové tabulky jsou užívány k uchovávání vytvořených datových tabulek uživatelem. Na obrázku číslo 10 lze vidět třídu **DataTable**, která obsahuje následující proměnné: jméno, 2D pole a očekávanou střední hodnotu datové tabulky. Metoda **countExpectedValue** je

používána ke kalkulaci očekávané střední hodnoty – viz obrázek číslo 10.

Obrázek 10: Třída DataTable

```
class DataTable:
    def __init__(self, name, darray):
        self.name = name
        self.darray = darray
        self.ev = self.countExpectedValue()

    def __str__(self):
        return self.name

    def countExpectedValue(self):
        ev = 0
        for array in self.darray:
            ev += array[0] * array[1]
        return ev
```

Zdroj: vlastní

Na obrázku číslo 12 lze vidět grafické rozhraní aplikace a konkrétně grafické rozhraní pro tvorbu nové tabulky. Jsou zde dva sloupce – sloupec pravděpodobnost a zisk. Sloupec pravděpodobnost může obsahovat pouze čísla v intervalu <0,1>. Zisk může obsahovat pouze nezáporná čísla. Za předpokladu, že tyto podmínky nejsou splněny, tak se daná chybná buňka v tabulce obarví na červeno a aplikace vypíše, že je chybný vstup. Těchto podmínek je zajištěno prostřednictvím regulárních výrazů. Na pravé straně jsou implementována další ovládání tabulky, která se mění na základě toho, zda je tabulka vytvářena, nebo editována.

Při tvorbě tabulky je nutné zvolit jméno tabulky a počáteční, koncovou řadu v tabulce, ze kterých se mají čerpat data pro uložení datové tabulky.

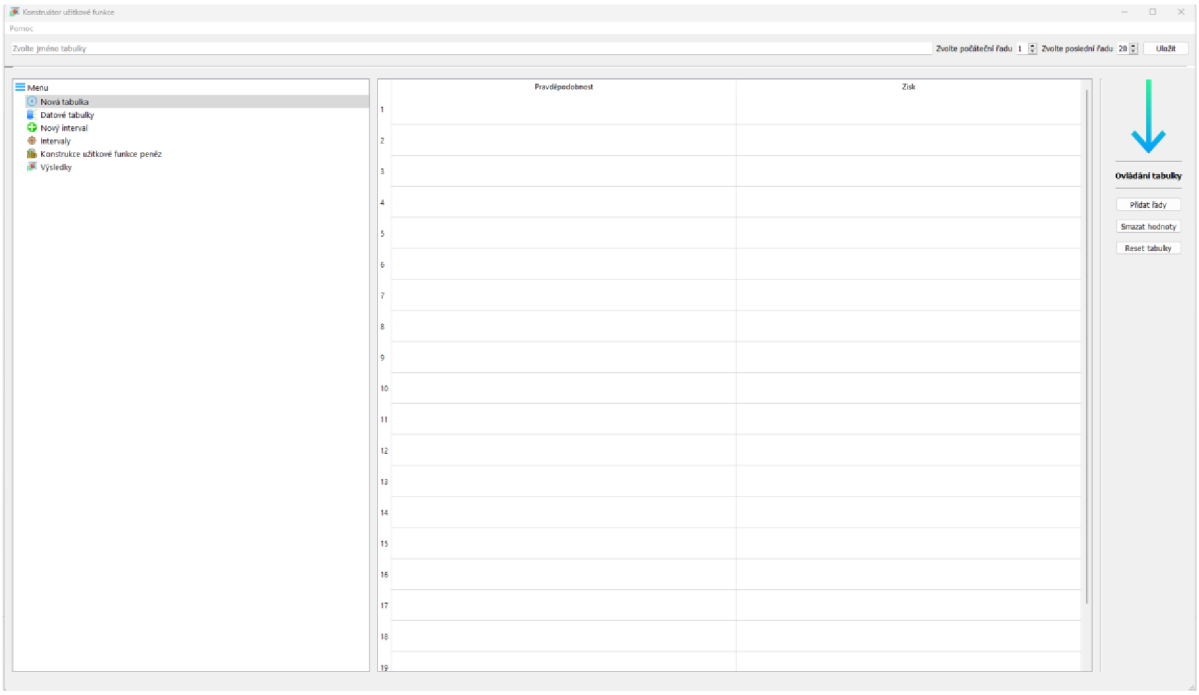
Hodnoty v tabulce je možné seřadit kliknutím na nadpisy sloupců. Je možné řadit vzestupně i sestupně. Je možné řadit dle pravděpodobnosti i zisku. To zajišťuje třída **CustomTableWidgetItem** a konkrétně metoda **__lt__,** díky které lze vzájemně porovnávat jednotlivé buňky tabulky. Na obrázku číslo 11 lze vidět tuto metodu.

Obrázek 11: CustomTableWidgetItem – řazení

```
class CustomTableWidgetItem(QtWidgets.QTableWidgetItem):
    def __lt__(self, other):
        try:
            return float(self.text()) < float(other.text())
        except ValueError:
            return super().__lt__(other)
```

Zdroj: vlastní

Obrázek 12: GUI – tvorba datové tabulky



Zdroj: vlastní

Kliknutím na položku v menu, která je potomkem uzlu Datové tabulky, je zachycena událost a je načtena již vytvořená tabulka. Tabulku je možné zde číst, měnit, mazat. Změnu jména je možné provést i dvojitým poklikem na položku v menu.

Na pravé straně je možné nalézt tlačítka k ovládání tabulky, intervalů atd. Kliknutím na daná tlačítka jsou provedeny následující akce při tvorbě nové tabulky:

- přidání řad,
- smazání všech hodnot v tabulce,
- reset tabulky, tedy navrácení tabulky do původní stavu.

Na obrázku číslo 13 je zobrazena část kódu, která je zodpovědná za editaci a tvorbu tabulky. Pokud uživatel odeslal datovou tabulku k uložení, tak je nejdříve volána metoda **onSaveBtnClicked**, kde je provedena kontrola vstupu a získání hodnot ze vstupu. Konkrétně se jedná o jméno datové tabulky, první a poslední číslo řady, ze kterých se má vytvořit nová tabulka. Následně je tabulka v grafickém rozhraní převedena na odpovídající 2D pole. Pokud nebyly nalezeny žádné chyby, tak je volána metoda **createNewTable**, která vygeneruje nové ID pro tuto tabulku a ta je následně uložena do hashovací mapy **tables**, kde klíčem je ID a odpovídající hodnotou je vytvořený objekt typu **DataTable**, který obsahuje jméno a 2D pole. V poslední fázi je vytvořen nový potomek pro uzel „Datové tabulky“ prostřednictvím metody

insertNewChildToParent.

Obrázek 13: Tvorba/editace datové tabulky

```
    def createNewTable(self, darray, name):
        id = str(uuid.uuid4())
        self.tables[id] = DataTable(name, darray)
        self.insertNewChildToParent(1, name, id, QtGui.QIcon("pck/img/cells.png"))
        self.resetState()
        self.treeMenu.topLevelItem(0).setSelected(True)
```

```
def onSaveBtnClicked(self):
    item = self.treeMenu.selectedItems()[0]
    startingRow = self.startingRow.text()
    lastRow = self.lastRow.text()
    if startingRow == lastRow:
        showdialog("Chyba řad", "Minimálně dvě řady jsou vyžadovány. Upravte prosím svůj vstup")
        return 0
    name = self.nameOfTableEdit.text()
    error = self.checkTableInput(startingRow, lastRow, name)
    if error is True: return 0

    darray = self.create2DArray(int(startingRow)-1, int(lastRow))
    if item.parent() is None:
        self.createNewTable(darray, name)
    else:
        self.updateTable(darray, name, item)
```

Zdroj: vlastní

Na obrázku číslo 14 je zobrazena metoda **deleteTable**, která je používána k mazání dané tabulky. Nejdříve je zjištěno pomocí události, která položka je vybrána v menu a následně je získáno její ID. Na základě tohoto ID je smazán záznam z hashovací tabulky **tables** a daný potomek je odstraněn z menu.

Obrázek 14: Mazání datové tabulky

```
def deleteTable(self):
    selectedMenuItems = self.treeMenu.selectedItems()
    selectedId = selectedMenuItems[0].data(0, QtCore.Qt.UserRole)
    del self.tables[selectedId]
    self.removeChild(selectedId, 1)
```

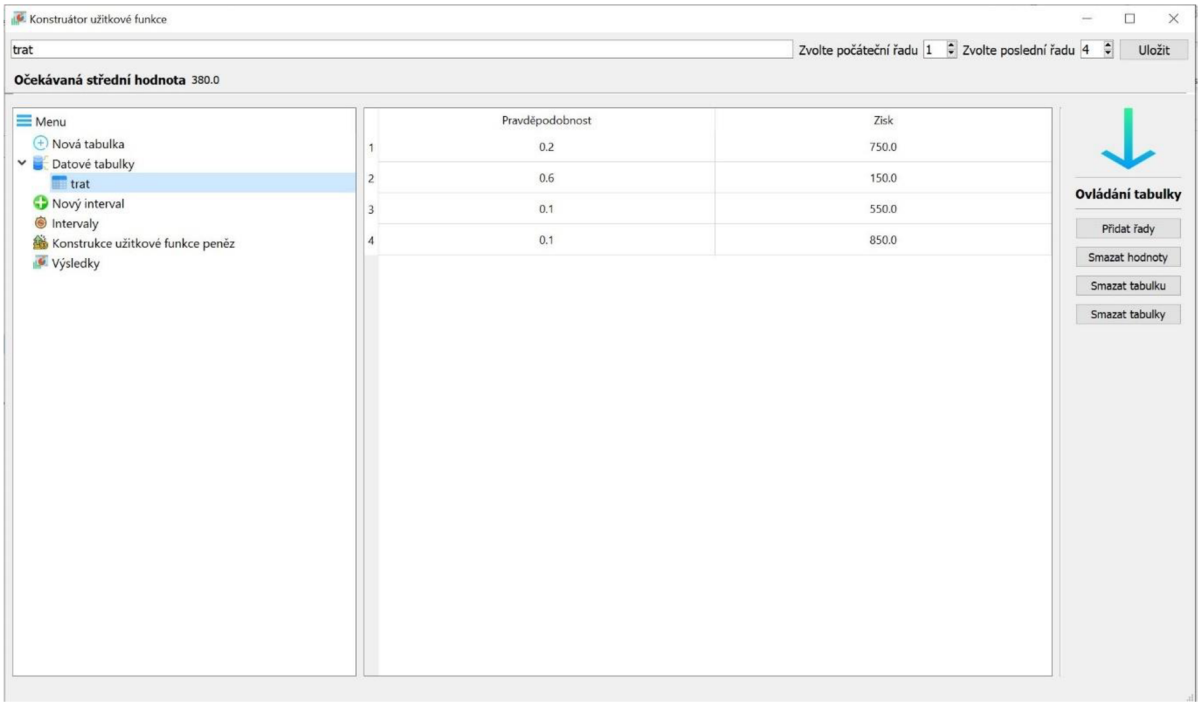
Zdroj: vlastní

Na obrázku číslo 15 je zobrazena podoba grafického rozhraní pro čtení tabulky. Na pravé straně přibyla tlačítka pro akce:

- tlačítko „Smazat tabulku“ umožňuje smazání vybrané datové tabulky,
- tlačítko „Smazat tabulky“ zajišťuje smazání všech momentálně vytvořených datových tabulek.

Nad levým menu je zobrazen výpočet očekávané střední hodnoty, který je počítán způsobem zobrazeným na obrázku číslo 10 prostřednictví metody **countExpectedValue**. Je zde proveden průchod 2 D polem, které reprezentuje vytvořenou datovou tabulku. Každý jeden průchod cyklem reprezentuje jeden řádek datové tabulky. V každém průchodu cyklu je vynásobena pravděpodobnost s korespondující hodnotou zisku a všechny takto získané hodnoty jsou sečteny.

Obrázek 15: GUI – editace a mazání datové tabulky



Zdroj: vlastní

6.2.2 Intervaly

Intervaly jsou druhým možným datovým objektem, který může být použit ke konstrukci užitkové funkce peněz. Třída **Interval** se skládá pouze z proměnných vyjadřující počáteční, koncovou hodnotu intervalu a jméno intervalu – viz obrázek číslo 16.

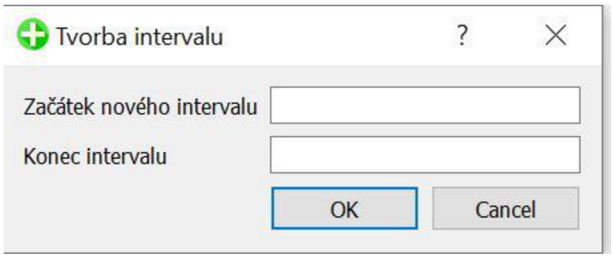
Obrázek 16: Třída Interval

```
class Interval:
    def __init__(self, start, end, name):
        self.startingValue = start
        self.endingValue = end
        self.name = name

    def __str__(self):
        return self.name
```

Zdroj: vlastní

Tvorba intervalu je zahájena kliknutím na položku menu „Nový interval“. Pro tvorbu intervalu je použito okno o dvou vstupních hodnotách, jak lze vidět na obrázku číslo 17. Vstup může obsahovat pouze nezáporná čísla. Kontrola vstupu je zajištěna regulárním výrazem. Je vynuceno, aby začátek intervalu byl roven nule.

Obrázek 17: GUI – nový interval



Zdroj: vlastní

Tvorba intervalu je poskytována metodou **createInterval**, která používá pro vygenerování grafického rozhraní pro tvorbu intervalu třídu **InputDialog**. Za předpokladu, že vstup je bez chyb, je vytvořeno nové ID pro tento interval, jméno je vygenerováno genericky následujícím způsobem: <první hodnota, druhá hodnota>. Dále je vytvořen objekt typu Interval obsahující počáteční, koncovou hodnotu a jméno, který je vložen do hashovací mapy **intervals**. Klíčem je ID intervalu, hodnotou potom je právě vytvořený objekt typu Interval. V poslední řadě je vytvořen nový potomek s právě vytvořeným ID. Rodič je v tomto případě položka v menu „Intervaly“.

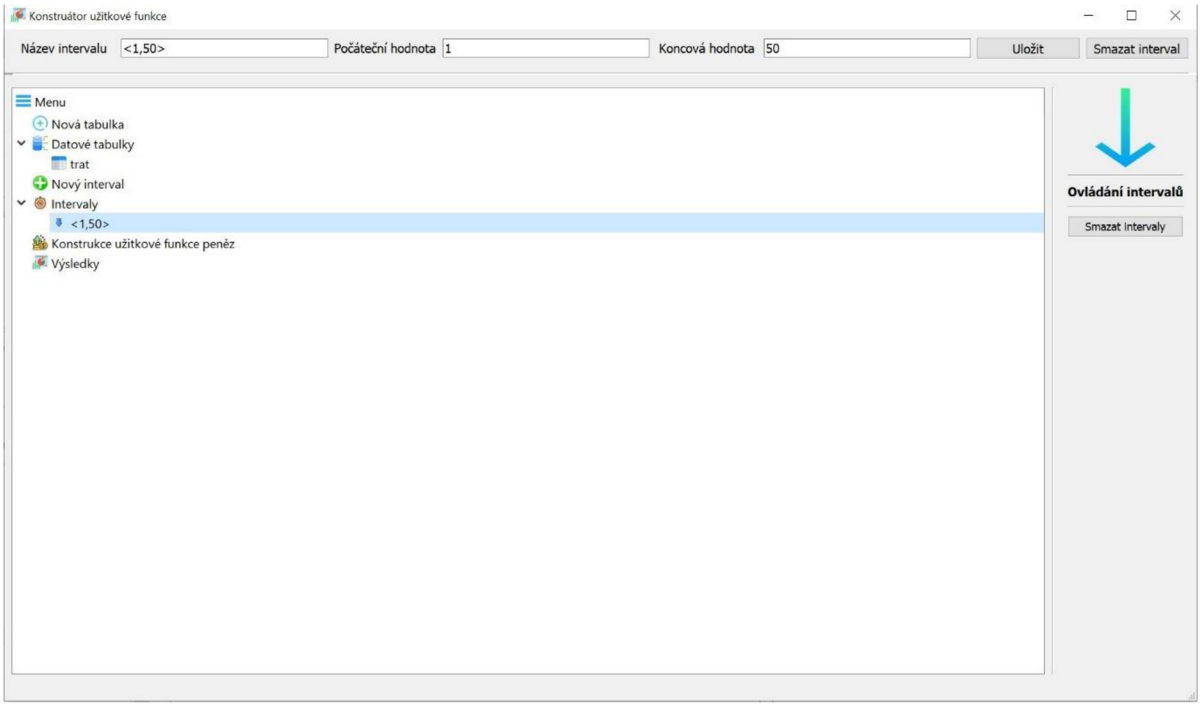
Obrázek 18: Vytvoření intervalu

```
def createInterval(self):
    dialog = InputDialog(labels=["Začátek nového intervalu","Konec intervalu"], title="Tvorba intervalu")
    values, error = self.setTwoInputDialog(dialog)
    if values is None:
        return 0
    while error is True:
        print(error)
        values, error = self.setTwoInputDialog(dialog)
        if values is None:
            return 0
    id = str(uuid.uuid4())
    name = "<" + values[0] + "," +  values[1] + ">"
    self.intervals[id] = Interval(values[0], values[1], name)
    self.insertNewChildToParent(3, name, id, QtGui.QIcon("pck/img/arrow-down.png"))
```

Zdroj: vlastní

Pro čtení, mazání intervalů je možné kliknout na potomky položky v menu „Intervaly“. Grafické rozhraní pro čtení, mazání, aktualizaci intervalů je zobrazeno na obrázku číslo 19. V této části aplikace lze již měnit název intervalu stejnými způsoby jako u datové tabulky. Na pravé straně je nyní tlačítko, které umožňuje smazat všechny vytvořené intervaly.

Obrázek 19: GUI – čteni/aktualizace/mazání intervalu



Zdroj: vlastní

Aktualizace a mazání intervalu je prováděna velmi obdobně jako u datových tabulek. Jedná se o načtení vstupu, kontrolu vstupu, načtení ID vybraného prvku a následnou aktualizací dat v hashovací mapě **intervals**. Mazání intervalu je taktéž prováděno podobně a to načtením ID a odstraněním položky v hashovací mapě **intervals** a v menu aplikace. Kód řešící aktualizaci intervalu je zobrazen na obrázku číslo 20.

Obrázek 20: Aktualizace intervalu

```
def updateInterval(self):
    name = self.nameOfInterval.text()
    startingValue = self.startValue.text()
    endingValue = self.endValue.text()

    if name == "" or len(name) > 50:
        showdialog("Chyba názvu", "Název nesmí být prázdný a může být maximálně dlouhý 50 znaků.")
        return 0

    if not re.match('(^[+]?([0-9]+)(\.[0-9]+)?)$', startingValue):
        showdialog("Chyba počáteční hodnoty", "Počáteční hodnota musí být kladné číslo.")
        return 0

    if not re.match('(^[+]?([0-9]+)(\.[0-9]+)?)$', endingValue):
        showdialog("Chyba koncové hodnoty", "Koncová hodnota musí být kladné číslo.")
        return 0

    item = self.treeMenu.selectedItems()[0]
    selectedId = item.data(0, QtCore.Qt.UserRole)
    self.intervals[selectedId] = Interval(startingValue, endingValue, name)
    item.setText(0, name)
    self.readInterval(selectedId)
```

Zdroj: vlastní

6.3 Konstrukce užitkové funkce peněz

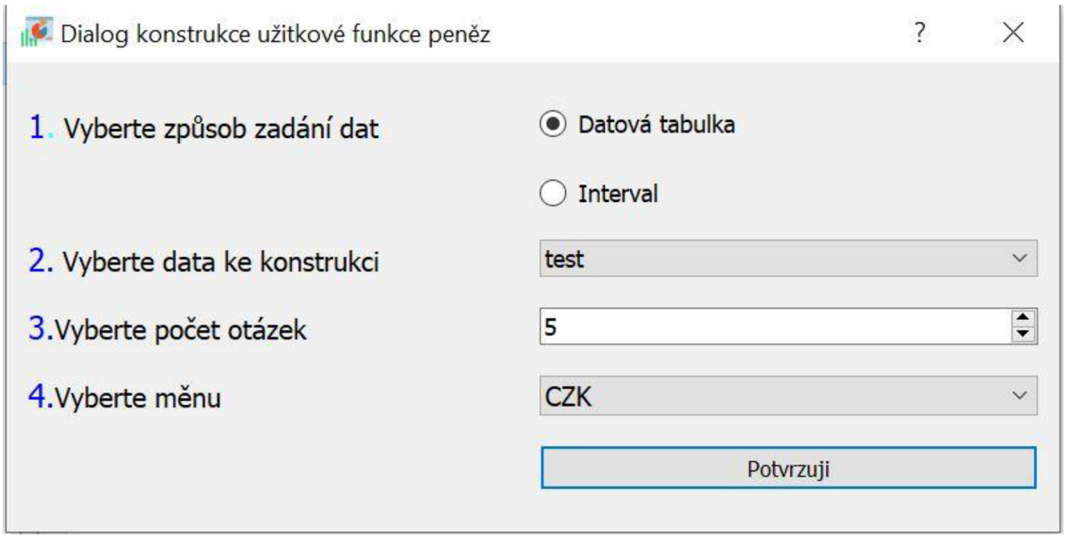
Vytvoření alespoň jednoho datového objektu je podmínkou k zahájení procesu konstrukce užitkové funkce peněz. Položka v menu s názvem „Konstrukce užitkové funkce“ je iniciátorem celého tohoto procesu

6.3.1 Výběr parametrů konstrukce užitkové funkce peněz

V první fázi jsou uživateli nabídnuty možné parametry pro konstrukci užitkové funkce peněz. Nabídka parametrů konstrukce užitkové funkce peněz je zobrazena na obrázku číslo 21. Uživatel může volit následující parametry:

- **způsob zadání dat** – jaký typ datového objektu bude použit pro konstrukci užitkové funkce peněz. Jsou zde na výběr dvě alternativy – Interval a Datová tabulka,
- **výběr dat ke konstrukci** – jakmile je zaškrtnut jeden ze způsobu dat, tak jsou načteny všechny intervaly, nebo datové tabulky do rozevíracího seznamu. Uživatel zde musí zvolit jednu variantu,
- **výběr počtu otázek** – minimální počet otázek, které budou uživatelovi položeny. Minimální počet otázek je 5. Vyšší počet vybraných otázek znamená lepší odhad,
- **výběr měny** – jsou zde na výběr různé měny v rozevíracím seznamu, ze kterého si uživatel může vybrat měnu, ve které budou otázky stylizovány. Možné varianty jsou: CZK, USD, EUR, Bezrozměrné.

Obrázek 21: Volba parametrů konstrukce užitkové funkce peněz



Zdroj: vlastní

Po kliknutí na položku v menu „Konstrukce užitkové funkce“ je spuštěno nové okno, které načítá a spravuje třída **ConstructionDialog.** Momentálně vytvořené datové tabulky a intervaly jsou předány z hlavní okna **MainView** do okna **ConstructionDialog** prostřednictvím konstruktoru.

Metoda **dialogueFinished** je volána v případě, že je provedeno potvrzení zvolených parametrů uživatelem. Zvolené parametry jsou nejdříve zkontrolovány a následně vraceny zpět ve formě objektu typu **UtilityConstructor** – viz obrázek číslo 22. Okno obsahující volbu parametrů pro konstrukci je ihned zavřeno po předání objektu typu **UtilityConstructor**. Pokud zadaná data jsou v pořádku vrácena do hlavního okna, tak je ihned otevřeno nové okno, kde jsou již generovány otázky uživateli. Tato část bude rozebrána v následující kapitole.

Obrázek 22: Dokončení nastavení parametrů

```
def dialogueFinished(self):
    btnOK, constructionType = self.isButtonChecked()
    selectedText = self.data.currentText()
    selectedObject = self.selectedDataByName(selectedText, constructionType)
    numOfQuestions = self.numOfQuestions.text()
    chosenCurrency = self.currencyBox.currentText()

    if selectedObject is None:
        showdialog("Žádný výběr", "Nebyla vybrána žádná data ke konstrukci.")
        return 0
    if btnOK == False:
        showdialog("Žádný výběr typu", "Nebyla zaškrnuta varianta konstukce.")
        return 0

    self.constructionParam = UtilityConstructor(constructionType, selectedObject, numOfQuestions, chosenCurrency)
    self.accept()
```

Zdroj: vlastní

Třída **UtilityConstructor** obsahuje následující proměnné: typ dat, vybraný datový objekt, počet otázek a zvolenou měnu. Tato třída je zobrazena na obrázku číslo 23.

Obrázek 23: Třída UtilityConstructor

```
class UtilityConstructor:
    def __init__(self, type, data, numOfQuestions, currency):
        self.type = type
        self.data = data
        self.numOfQuestions = numOfQuestions
        self.currency = currency
```

Zdroj vlastní

6.3.2 Generování otázek

V této kapitole bude popsáno, jakým způsobem je řešeno generování otázek aplikací uživateli. V první kapitole bude popsán obecný algoritmus generování otázek. V dalších kapitolách bude popsána konkrétní softwarová implementace.

6.3.2.1 Obecný algoritmus generování otázek

V rámci této kapitoly bude rozepsán obecný algoritmus, který je v aplikaci využíván pro generování otázek.

1. Generování uvítací zprávy.
2. Generování otázky k získání maximální hodnoty M – otázky typu M_BASED. Hodnota M je přidána do kolekce zisků.
3. Pokud kolekce zisků neobsahuje hodnotu 0, je přidána do kolekce zisků hodnota 0. Pokud obsahuje kolekce zisků již zisk 0, tento krok je přeskočen.
4. Seřazení kolekce zisků od nejmenšího po největší.
5. Nejnižší hodnotě zisku je přiřazen užitek 0, nejvyšší hodnotě zisku je přiřazen užitek 1.
6. Vytvoření tabulky z kolekce zisků a odpovídajících užitků.
7. Generování náhodných otázek. Otázky jsou generovány tak, aby bylo zaručeno rovnoměrné rozložení bodů do pěti intervalů.
   7.1 Pokud existují zisky bez přiřazených užitků, je preferenčně vygenerována náhodně otázka typu PROBABILITY_BASED. Jinak je vygenerována náhodně otázka typu VALUE_BASED.
   7.2 Pokud je otázka typu PROBABILITY_BASED, tak na základě zodpovězené pravděpodobnosti je spočítán užitek pro zvolený bod.
   7.3 Pokud je otázka typu VALUE_BASED, tak na základě zodpovězené hodnoty je vytvořen nový bod a dopočítán užitek pro daný bod.
8. Je provedena kontrola rovnoměrného rozložení bodů do intervalů v případě, že byla zadaná datová tabulka jako vstup pro konstrukci užitkové funkce peněz. Pokud není dosaženo rovnoměrného rozložení, tak jsou generovány další otázky typu VALUE_BASED k dosažení rovnoměrného rozložení.
9. Generování koncové zprávy.

V prvním kroku je generována uvítací zpráva, kde je uživatel obeznámen ohledně procesu generování otázek. Také je zde řečeno, že se předpokládá, že užitek ze zisku nula je roven nule. V poslední řadě je zdůrazněno, že nalezené řešení užitkové funkce peněz nemusí být optimální.

V druhém kroku je generována otázka, která je používána k získání maximální hodnoty M uživatele. Znalost toho bodu je klíčová pro odhad optimálních počátečních parametrů regrese. Získaná hodnota M je uložena do kolekce zisků.

M_BASED otázka je v následujícím formátu:

Uveďte nejnižší takovou hodnotu zisku, kdy nejste schopni rozlišit, zda jste dostali tuto hodnotu či dvojnásobnou.

V třetím kroku je přidána hodnota zisku 0 do kolekce, pokud ji ještě neobsahuje. Je to z důvodu, že je počítáno s tím, že užitek ze zisku nula je roven nule. Proto tuto hodnotu do kolekce zisků přidáváme, pokud tam ještě není.

Ve čtvrtém a pátém kroku je seřazena kolekce zisků od nejmenšího po největší a je přiřazen nejmenšímu zisku užitek nula a největšímu zisku užitek jedna.

Ve šestém kroku je vytvořena tabulka, která reprezentuje momentální kolekci zisků s odpovídajícími hodnotami užitků.

V sedmém kroku je definována podmínka, že jsou generovány v cyklu otázky, dokud jich již nebylo zodpovězeno takové množství, kolik uživatel navolil a zároveň musí být dosaženo rovnoměrného rozložení získaných bodů do pěti intervalů.

Nyní jsou generovány dva typy otázek – PROBABILITY_BASED a VALUE_BASED.

PROBABILITY_BASED otázka je v následujícím formátu:

Pokud byste mohl/a hrát takovou hru, kde první alternativa je, že s pravděpodobností p získáte zisk x_1 a zisk rovný x_2 s pravděpodobností (1–p). Druhá alternativa je, že s jistotou získáte zisk Z. Jaká hodnota pravděpodobnosti p Vás dělá indiferentní/m v této hře? Do jaké maximální hodnoty pravděpodobnosti byste byl/a ochoten hrát zmíněnou hru a od které byste preferoval/a přímou výplatu Z? Odpověď ve tvaru 0.NN.

- x_1 a x_2 jsou náhodně zvolené zisky z tabulky, které mají již přiřazenou hodnotu užitku,
- Z je náhodně zvolená hodnota zisku z tabulky, která nemá ještě přiřazenou hodnotu užitku.

VALUE_BASED otázka je v následujícím formátu:

Pokud byste mohl/a hrát takovou hru, že s pravděpodobností p získáte x_1 a s pravděpodobností (1-p) získáte x_2. Jakou cenu jste ochoten/na za tuto hru zaplatit? Jaká hodnota Vás dělá indiferentním v této situaci? Tedy částka, od které preferujete raději výplatu před hraním zmíněné hry.

- x_1 a x_2 jsou náhodně zvolené zisky, které mají již přiřazenou hodnotu užitku,
- p je náhodně generovaná pravděpodobnost zaokrouhlena na dvě desetinná místa.

V osmém kroku je prováděna kontrola rovnoměrného rozložení bodů do pěti intervalů. V případě, že uživatel zadal datovou tabulku jako vstup, tak je možné, že hodnoty nemusí být rovnoměrně rozložené v intervalech z důvodu preferenčního generování otázek cílených na zisky bez užitku v datové tabulce. V případě, že není dosaženo rovnoměrného rozložení bodů, tak jsou generovány dodatečné otázky typu VALUE_BASED, dokud není dosaženo rovnoměrného rozložení bodů do intervalů.

V posledních kroku je vygenerována koncovou zpráva informující uživatele o dokončení všech otázek.

6.3.2.2 Implementace generování otázek

Třída **QuestionsUi** je zodpovědná za grafické rozhraní pro generování otázek. Předání zvolených parametrů a dat pro konstrukci užitkové funkce peněz probíhá opět přes konstruktor. Proměnná **selectedData** zde ukládá objekt typu **UtilityConstructor**. Na základě této proměnné jsou generovány otázky. Počátek konstruktoru třídy **QuestionsUi** je zobrazen na obrázku číslo 24. Nachází se zde proměnná **counter**, která je používána k odpočítávání zbývajících otázek. Důležitou proměnnou je list **utilityIntervals**, který je tvořen z několik dalších interních listů. Každý samostatný list proměnné **utilityIntervals** zajišťuje uchovávání bodů v daném intervalu. Pomocí proměnné **NUMBER_OF_INTERVALS** je definováno celkové množství intervalů. Tato proměnná je nastavena na hodnotu pět.

Obrázek 24: Konstruktor třídy QuestionsUi

```
class QuestionsUi(QtWidgets.QDialog):
    def __init__(self, selectedData):
        super(QuestionsUi, self).__init__()
        uic.loadUi('pck/view/templates/questions_dialog.ui', self)
        self.setWindowTitle("Konstrukce užitkové funkce peněz.")
        self.setWindowIcon(QtGui.QIcon("pck/img/stadistics.png"))
        self.selectedData = selectedData
        self.initDialogue(selectedData)
        self.initLabels()
        self.questions = []
        self.M = 0
        self.numOfQuestions = int(self.selectedData.numOfQuestions)
        self.counter = 1
        self.utilityIntervals = []
        for i in range(ConstantManager.NUMBER_OF_INTERVALS):
            self.utilityIntervals.append([])
```

Zdroj: vlastní

6.3.2.2.1 Základní stavební bloky

Pro rozeznání typů otázek a typů konstrukce jsou využívány v aplikaci různé výčty. Na základě nich lze rozlišit, jakého typu je daná otázka nebo zpráva. Jsou používány celkem dva výčty – **ConstructionType** a **QuestionType**. Existuje celkem pět typů otázek a zpráv, které jsou generovány. K odlišení různých typů je používán výčet **QuestionType**, který

obsahuje právě těchto pět typů, které jsou v aplikaci využívané.

Obrázek 25: Soubor enums

```
class ConstructionType(Enum):
    INTERVAL_BASED=1
    DATATABLE_BASED=2

class QuestionType(Enum):
    VALUE_BASED=1
    PROBABILITY_BASED=2
    STARTING_MESSAGE=3
    FINAL_MESSAGE=4
    M_MESSAGE=5
```

Zdroj: vlastní

Velmi klíčovou třídou je třída **Question**, která je zobrazena na obrázku číslo 26. Tato třída je používána k uchovávání otázky, odpovědi a výpočtu užitku na základě vygenerované otázky. Význam klíčových proměnných této třídy je následující:

- **position** – uchovává pořadí otázky,
- **text** – vygenerovaný text pro otázku,
- **type** – typ otázky dle výčtu QuestionType,
- **p1** – náhodně vygenerovaná pravděpodobnost, nebo stanovena dle odpovědi uživatele v případě otázky typu PROBABILITY_BASED,
- **X** – proměnná pro uchovávání nové hodnoty zisku získané odpovědí uživatele v případě VALUE_BASED otázky, nebo náhodně vybraná hodnota zisku bez přiřazeného užitku v případě otázek typu PROBABILITY_BASED,
- **x1 a y1** – první náhodně vybraná dvojice zisku a korespondující užitku,
- **x2 a y2** – druhá náhodně vybraná dvojice zisku a korespondující užitku**,**
- **utility** – vypočítaný užitek pro danou otázku dle funkce **countUtility**.

Obrázek 26: Třída Question

```
class Question:
    def __init__(self, position, text, type):
        self.position = position
        self.text = text
        self.type = type
        self.p1 = None
        self.X = None
        self.x1 = None
        self.y1 = None
        self.x2 = None
        self.y2 = None
        self.utility = None

    def countUtility(self):
        reversedProb = 1-self.p1
        self.utility = round((self.p1 * self.y1) + (reversedProb * self.y2),5)
        return self.utility
```

Zdroj: vlastní

Další důležitou třídou je třída **QuestionGenerator**, která zahrnuje funkcionalitu ke generování otázek. Obsahuje různé metody, které na základě vstupních parametrů generují a vrací objekty typu **Question**. Text takto vygenerované otázky je vypsán uživateli v grafickém rozhraní aplikace. Nyní budou zmíněny metody této třídy, které jsou klíčové.

Na obrázku číslo 27 jsou zobrazeny metody pro generování maximální hodnoty M a pro generování otázky typu VALUE_BASED. Pro generování M metody se jedná pouze o nastavení textu a pořadí se stanovuje na -1, jelikož tento typ otázky není počítán mezi otázky sloužící k získávání bodů užitkové funkce peněz uživatele. Generování VALUE_BASED otázek je prováděno na základě metody s názvem **generateValueQuestion**, která přijímá následující vstupní parametry: x1, x2, p1, y1, y2. Tyto parametry jsou významově stejné jako proměnné týkající se třídy **Question**. Vstupní parametr p1 je náhodně vygenerovaná pravděpodobnost pro tuto metodu. Na základě těchto hodnot je vypisován text tohoto typu otázky. Dále je počítána reverzní pravděpodobnost. V poslední fázi je vytvářen objekt typu **Question**, který je nejdříve tvořen čítačem, textem otázky a je nastavený typ otázky na QuestionType.VALUE_BASED. Tomuto vytvořenému objektu typu **Question** jsou následně přiřazeny jeho instanční proměnné x1, x2, y1, y2, p1 dle odpovídajících vstupních parametrů metody x1, x2, y1, y2 a p1. Metoda vrací objekt typu **Question**. V aplikaci je používána i metoda **generateValueQuestionWithPositions**, která funguje téměř ekvivalentně jako metoda **generateValueQuestion**. Jediným rozdílem je, že metoda **generateValueQuestionWithPositions** přijímá jako vstupní parametry i pozici dat v tabulce a ukládá ji do objektu typu **Question**.

Obrázek 27: QuestionGenerator – generování M a VALUE_BASED otázky

```
def generateValueQuestion(self, x1, x2, p1, y1, y2):
    reverseP = round((1-p1),2)
    text=""
    if self.currency == "Bezrozměrné":
        text = text = "Pokud byste mohl/a hrát takovou hru, že s pravděpodobností " + str(p1)
        + " získáte " + str(x1) + " a s pravděpodobností "+ str(reverseP) + " získáte " + str(x2)
        ". Jakou cenu jste ochoten/na za tuto hru zaplatit? Jaká hodnota Vás dělá indiferentním v této situaci?"
        "Tedy částka, od které preferujete raději výplatu před hraním zmíněné hry."
    else:
        text = "Pokud byste mohl/a hrát takovou hru, že s pravděpodobností " + str(p1) + " získáte " + str(x1)
        + " " + self.currency + " a s pravděpodobností "+ str(reverseP) + " získáte "
        + str(x2) + " " + self.currency
        +  ". Jakou cenu jste ochoten/na za tuto hru zaplatit? Jaká hodnota Vás dělá indiferentním v této situaci?"
        "Tedy částka, od které preferujete raději výplatu před hraním zmíněné hry."

    question = Question(self.counter, text, QuestionType.VALUE_BASED)
    question.x1 = float(x1)
    question.x2 = float(x2)
    question.p1 = p1
    question.y1 = float(y1)
    question.y2 = float(y2)
    self.counter += 1
    return question
```

Zdroj: vlastní

Další důležitou metodou ve třídě ***QuestionGenerator*** je metoda **generateProbabilityQuestion**. Jedná se o generování otázky typu PROBABILITY_BASED vysvětlené v kapitole 6.3.2.1. Tato metoda je tvořena následujícími vstupními parametry: x1, x2, X, y1, y2. Vstupní parametry odpovídají významově proměnným ve třídě **Question**, které byly již vysvětleny dříve. Vstupní parametr **X** je náhodně vybraný zisk, který zatím nemá přiřazený žádný užitek. Na základě těchto vstupních parametrů je následně generován text otázky. Dále je tvořen objekt typu **Question**, který tentokrát je nastaven na typ QuestionType.PROBABILITY_BASED. Postup je zde velmi obdobný jako u metody **generateValueQuestion**. Rozdílem je, že není přímo v této metodě nastavována instanční proměnná p1 objektu typu **Question**, ale je nastavována instanční proměnná **X**. Metoda vrací objekt typu **Question**. Metodu zobrazující tento postup lze vidět na obrázku číslo 28.

Obrázek 28: Třída GenerateQuestion – metoda generateProbabilityQuestion

```
def generateProbabilityQuestion(self, x1, x2, X, y1, y2, row):
    text = ""
    if self.currency == "Bezrozměrné":
        text = "Pokud byste mohl/a hrát takovou hru, kde první alternativa je, že s pravděpodobností p"
        " získáte zisk " + str(x1) + " a zisk rovný " + str(x2) + " s pravděpodobností (1-p)"
        + ". Druhá alternativa je, že s jistotou získáte zisk " + str(X)
        + ". Jaká hodnota pravděpodobnosti p Vás dělá indiferentním v této hře?"
        "Do jaké maximální hodnoty pravděpodobnosti by jste byl/a ochoten/na hrát zmíněnou hru a od které byste preferoval/a přímou výplatu "
        + str(X)  + "? Vyplňte prosím hodnotu pravděpodobnosti ve tvaru 0.XX."
    else:
        text = "Pokud byste mohl/a hrát takovou hru, kde první alternativa je, že s pravděpodobností p" + " získáte zisk "
        + str(x1) + " " +  self.currency + " a zisk rovný " + str(x2) +" " + self.currency + " s pravděpodobností (1-p)"
        + ". Druhá alternativa je, že s jistotou získáte zisk "
        + str(X) + " " + self.currency
        + ". Jaká hodnota pravděpodobnosti p Vás dělá indiferentním v této hře?"
        "Do jaké maximální hodnoty pravděpodobnosti by jste byl/a ochoten/na hrát zmíněnou hru a od které byste preferoval/a přímou výplatu "
        + str(X) + " " + self.currency + "? Vyplňte prosím hodnotu pravděpodobnosti ve tvaru 0.XX."

    question = Question(row, text, QuestionType.PROBABILITY_BASED)
    question.y1 = float(y1)
    question.y2 = float(y2)
    question.X = float(X)
    question.y1 = float(y1)
    question.y2 = float(y2)
    self.counter += 1
```

Zdroj: vlastní

V této třídě se nadále nachází metody, jejichž úkolem je generování různých typů informačních zpráv – uvítací zpráva a koncová zpráva. K tomuto jsou použity dvě metody ve třídě **GenerateQuestion** a jejich názvy jsou následující: **generateStartingQuestion** a **generateFinalInformation**. Tyto metody vrací text, který informuje uživatele o průběhu dotazování, jakým způsobem bude jeho užitková funkce peněz zkonstruována a také je uživatel informován o tom, že nalezená užitková funkce peněz nemusí být optimální. Konkrétní text těchto otázek bude ukázán v rámci kapitoly, které popisuje grafické rozhraní aplikace pro generování otázek.

6.3.2.2.2 Implementace obecného algoritmu

V předchozí kapitole byly popsány základní stavební bloky, které jsou užívány k implementaci algoritmu popsaného v kapitole 6.3.2.1. Nyní bude popsán způsob, jakým je softwarově konkrétně implementován tento algoritmus. Je také zde užívaná proměnná **counter**, která je používána k odpočítávání otázek.

V rámci prvního kroku je zobrazena úvodní zpráva uživateli. Tato funkcionalita se nachází v metodě **initLabels**. Zde je generována počáteční zpráva vytvořenou instanční proměnnou **generator**, která je typu **QuestionGenerator**. Je zde volána metoda **generateStartingQuestion**, která je používána k vytváření počáteční zprávy. Tato zpráva je následně zobrazena uživateli. Tento popsaný postup lze vidět na obrázku číslo 29, který zobrazuje metodu **initLabels**. Druhý krok a s ním související generování otázky M začíná automaticky po potvrzení uvítací obrazovky uživatelem.

Obrázek 29: Generování počáteční zprávy

```
def initLabels(self):
    self.label_name.setText("Název datového souboru: " + self.selectedData.data.name)
    if self.selectedData.type == ConstructionType.DATATABLE_BASED:
        self.label_type.setText("Konstrukce na základě datové tabulky")
    else:
        self.label_type.setText("Konstrukce na základě zadaného intervalu")

    question = self.generator.generateStartingQuestion()
    self.answerBtn.hide()
    self.answerLine.hide()
    self.finish.hide()
    self.questionLabel.setText(question.text)
```

Zdroj: vlastní

V rámci druhého kroku je vygenerována a zobrazena otázka na maximální hodnotu M uživatele. Třída **QuestionUi** obsahuje instanční proměnnou **MQuestion**, která je typu boolean. Tato proměnná je používána k indikaci, zda bylo již na tento typ otázky zodpovězeno. Pokud ne, tak je vygenerována otázka na maximální hodnotu M za použití instanční proměnné **generator** a její metody **generateMQuestion**. Tento popsaný postup je zobrazen na obrázku číslo 34, kde je zobrazena část metody **generateQuestion**, která je zodpovědná za generování otázky na maximální hodnotu M.

Obrázek 30: Generování M otázky

```
if self.MQuestion:
    question = self.generator.generateMQuestion()
    self.questionLabel.setText(question.text)
    self.numq.setText("Úvodní otázka")
    self.remaining.hide()
    return 0
```

Zdroj: vlastní

Jakmile je odpověď uživatele na otázku potvrzena, tak je zachycena prostřednictví metody **confirmQuestion**, která je zobrazena na obrázku číslo 31. V rámci této metody jsou zachytávány všechny odpovědi na různé otázky a na základě typu otázky jsou prováděny akce. Pro zachycení odpovědi na M otázku je nejdříve přečtena hodnota ze vstupu. Následně platí, že pokud instanční proměnná **MQuestion** nabývá hodnoty **True**, tak je zpracovávána odpověď na M otázku. Je zde provedena standartní kontrola vstupu. V případě, že kontrola vstupu je bez chyb, tak je získaná odpověď uložena do instanční proměnné **M**. Tato proměnná je následně vložena do kolekce zisků. Důležité je zde také volání metody **generateQuestions**, která je používána ke generování nové otázky pro uživatele.

Obrázek 31: Zachycení a kontrola odpovědi na M otázku

```
def confirmQuestion(self):
    answer = self.answerLine.text()
    if self.MQuestion:
        if not re.match('(^[+]?([0-9]+)(\.[0-9]+)?)$', answer):
            showdialog("Chyba čísla", "Zadaný vstup není validní číslo. Prosím opravte hodnotu.")
            return 0
        else:
            self.M = int(answer)
            self.MQuestion=False
            self.answerLine.setText("")
            self.remaining.show()
            self.generateQuestions()
            return 0
```

Zdroj: vlastní

Třetí, čtvrtý, pátý, šestý krok jsou provedeny v rámci metody **initDialogue**, která je zobrazena na obrázku číslo 32. Zde je inicializována instanční proměnná **generator**, která je používaná ke generování otázek. Za předpokladu, že zadaná datová tabulka a kolekce zisků neobsahuje nulu, tak je přidána hodnota nula. Jakmile jsou data připravena, tak pro tvorbu tabulky na základě intervalu je použita metoda **createIntervalTable** a pomocí volání metody **sort** je možné pole obsahující hodnoty setřídit vzestupně. Pro tvorbu tabulky na základě datové tabulky je používána metoda **setTable**, kterou lze vidět na obrázku číslo 33. Metoda **sortItems**, která setřiďuje hodnoty dle zisku, je volána na konci. Jakmile jsou hodnoty seřazeny, tak nejnižšímu zisku je nastaven užitek nula a nejvyššímu zisku je nastaven užitek jedna. V metodě **createIntervalTable** je implementována podobná logika z hlediska toho, že nejmenší hodnotě je přiřazen užitek nula a nejvyšší hodnotě užitek jedna.

Obrázek 32: InitDialogue metoda

```
def initDialogue(self, selectedData):
    self.initSettingOfTable()
    self.generator = QuestionGenerator(self.selectedData.numOfQuestions, self.selectedData.currency)
    if selectedData.type == ConstructionType.INTERVAL_BASED:
        startingValue = float(self.selectedData.data.startingValue)
        endingValue = float(self.selectedData.data.endingValue)
        intervalArray = [startingValue, endingValue]
        item = 0
        if item not in intervalArray:
            intervalArray.append(item)
        intervalArray.sort()
        self.createIntervalTable(intervalArray)
    else:
        newData = np.array([0,0])
        updateData = np.vstack((self.selectedData.data.darray, newData))
        self.setTable(updateData)
```

Zdroj: vlastní

Obrázek 33: Metoda setTable

```
def setTable(self, array):
    row_count = (len(array))
    column_count = (len(array[0]))
    self.utility_table.setColumnCount(column_count)
    self.utility_table.setRowCount(row_count)

    for row in range(row_count):
        for column in range(0, column_count, 1):
            if column == column_count-1:
                item = CustomTableWidgetItem("")
                item.setTextAlignment(QtCore.Qt.AlignVCenter | QtCore.Qt.AlignHCenter)
                self.utility_table.setItem(row, column, item)
                continue
            item = CustomTableWidgetItem(str(array[row, column+1]))
            item.setTextAlignment(QtCore.Qt.AlignVCenter | QtCore.Qt.AlignHCenter)
            self.utility_table.setItem(row, column, item)

    self.utility_table.sortItems(0, QtCore.Qt.AscendingOrder)
    self.insertDataToColumn(0, column_count-1, 0)
    self.insertDataToColumn(row_count-1, column_count-1, 1)
```

Zdroj: vlastní

Všechny ostatní kroky algoritmu jsou implementovány v metodě **generateQuestions**. V rámci sedmého kroku jsou generovány náhodné otázky typu PROBABILITY_BASED a VALUE_BASED. Tato funkcionalita je implementována v rámci metody **generateQuestions**. Na obrázku číslo 34 lze vidět část metody řešící náhodné generování těchto dvou typů otázek. Nejdříve je zjišťován počet prázdných buněk v prvním sloupci, čímž je získán počet zisků bez přiřazených užitků. Tato hodnota je uložená v proměnné **emptyCols**. Je zde prováděno větvení na základě toho, zda proměnná **emptyCols** je nulová a na základě proměnné **isExtension**. Proměnná **isExtension** je nastavována na hodnotu **True** v případě, že jsou generovány dodatečné otázky. Pokud tedy v tabulce není žádný zisk bez přiřazeného

užitku a proměnná **isExtension** je nastavena na **True**, tak je prováděn kód v první větvi, která je využívána ke generování otázek typu VALUE_BASED. Metoda **generateValueBasedQuestionWithIntervals** je užívána ke generování náhodných hodnot pro otázku a náhodnému zvolení intervalu. Na základě těchto hodnot je vygenerována otázka prostřednictví proměnné **generator** a její metody **generateValueQuestion**. Metoda **generateValueBasedQuestionWithIntervals** a algoritmus pro generování VALUE_BASED otázek bude popsán u obrázků 35-37. Velmi obecný postup pro generování VALUE_BASED otázek v aplikaci je následující:

1. nalezení volných intervalů,
2. náhodné zvolení jednoho z volných intervalů,
3. nalezení vhodných kombinací spadajících do vybraného intervalu,
4. náhodné vybrání jedné z kombinací a navrácení hodnoty.

Pokud výše zmíněná podmínka není splněna, tedy existuje alespoň jeden zisk bez přiřazeného užitku a zároveň proměnná **isExtension** je nastavena na **False**, tak je vybrána druhá větev pro pokračování programu. Zde jsou generovány otázky typu PROBABILITY_BASED. Nejdříve je náhodně vybrán zisk v tabulce, který je bez přiřazené hodnoty užitku. Pro tento účel je generováno náhodné číslo, které je uloženo do proměnné **randomNumbX**. Pro tento typ otázky jsou náhodně vybírány dvě řady z tabulky, které již mají přiřazené hodnoty užitků. První náhodně vybraná řada musí obsahovat hodnotu zisku menší než náhodně zvolená hodnota zisku bez přiřazeného užitku (proměnná **itemX**) a druhá řada naopak musí obsahovat vyšší hodnotu zisku. K tomuto účelu je používána v kódu metoda **findFilledAndEmptyColumnsInTableColumn**, jejíchž účelem je zjištění počtu zisků s přiřazenými užitky v daném intervalu. Proměnná **filledColumnsBelow** obsahuje hodnotu, která odpovídá počtu řad, které mají již přiřazené užitky a které jsou zároveň menší než náhodně zvolená hodnota (proměnná **itemX**). Proměnná **filledColumnsUp** naopak obsahuje počet zisků s přiřazenými užitky, které jsou vyšší. Dále jsou generována dvě náhodná čísla, která jsou ukládána do proměnných **randomNumb1** a **randomNumb2**. Proměnná **randomNumb1** je používána k výběru jedné z možných řad tabulky, které obsahují hodnotu zisku nižší než hodnota uložená v proměnné **itemX** a proměnná **randomNumb2** je určena pro výběr jedné z řad tabulky, kde hodnota zisku je vyšší. Následně již probíhá nalezení konkrétních hodnot zisků a odpovídajících užitků v tabulce. Proměnné **item1** a **item2** ukládají konkrétní hodnoty zisků na náhodně zvolených pozicích a proměnné **y1** a **y2** obsahují korespondující hodnoty užitků. V poslední řadě je tvořen objekt typu **Question** a to pomocí

instanční proměnné **generator** a její metody **generateProbabilityQuestion**, kde vstupními parametry jsou náhodně vybrané hodnoty zisků a užitků. Na základě této metody je vygenerována otázka, která je následně zobrazena uživateli.

Obrázek 34: Část metody generateQuestion

```
emptyCols, filledUtilityColumns = self.iterateColumn(1)
if emptyCols == 0 or self.isExtension==True:
  question, positionOfInterval = self.generateValueBasedQuestionWithIntervals()
  self.positionOfCurrentInterval = positionOfInterval
  item1, item2 = self.findValuesAtCurrentPositions(question.r1, question.r2, 0)
  question = self.generator.generateValueQuestion(item1.text(), item2.text(), question.p1, question.y1, question.y2)
  self.currentRows = [item1.row(), item2.row()]
else:
   randomNumbX = self.generateRandomNumb(0, emptyCols-1)
   itemX, positionInTable = self.findValueAtPosition(randomNumbX)
   rowCount = self.utility_table.rowCount()
   emptyColumnsBelow, filledColumnsBelow = self.findFilledAndEmptyColumnsInTableColumn(1, 0, positionInTable)
   emptyColumnsUp, filledColumnsUp = self.findFilledAndEmptyColumnsInTableColumn(1, positionInTable, rowCount)
   randomNumb1 = self.generateRandomNumb(0, filledColumnsBelow-1)
   randomNumb2 = self.generateRandomNumb(0, filledColumnsUp-1)
   finalRandomNumber2 = randomNumb2+filledColumnsBelow
   item1, item2 = self.findValuesAtPositions(randomNumb1, finalRandomNumber2, 0)
   y1 = self.utility_table.item(item1.row(), item1.column()+1).text()
   y2 = self.utility_table.item(item2.row(), item2.column()+1).text()
   self.colorRow(itemX.row(), QtGui.QColor(255,255,0))
   question = self.generator.generateProbabilityQuestion(item1.text(), item2.text(), itemX.text(), y1, y2, itemX.row())
   self.currentRows = [item1.row(), item2.row()]
   self.currentRows.insert(2, question)
```

Zdroj: vlastní

Klíčovou metodou pro generování VALUE_BASED otázek je zde metoda **generateValueBasedQuestionWithInterrvals**, která je zobrazena na obrázku číslo 35.

Obrázek 35: Metoda generateValueBasedQuestionWithIntervals

```
def generateValueBasedQuestionWithIntervals(self):
    availableIntervals = self.findFreeIntervals()
    numberOfFreeIntervals = self.countNumberOfFreeIntervals(availableIntervals)
    randomNumber = self.generateRandomNumb(0, numberOfFreeIntervals-1)
    positionOfInterval = self.getPositionOfInterval(availableIntervals, randomNumber)
    part = 1/ConstantManager.NUMBER_OF_INTERVALS
    intervalStart = part*positionOfInterval
    intervalEnd = (intervalStart + part)-0.01
    utilities = self.getAllUtilities()
    combinations = self.findMostSuitableCombinations(utilities, intervalStart, intervalEnd)
    randomNumber = self.generateRandomNumb(0, len(combinations)-1)
    return combinations[randomNumber], positionOfInterval
```

Zdroj: vlastní

Na obrázku číslo 36 je zobrazena metoda **findFreeIntervals**. Je zde získáván list, který obsahuje na dané pozici intervalu hodnotu 1, pokud je daný interval volný, pokud není volný, tak na dané pozici je 0.

Obrázek 36: Metoda generateValueBasedQuestionWithIntervals

```
def findFreeIntervals(self):
    availablePositions = []
    numOfIntervals = len(self.utilityIntervals)
    if len(set(map(len, self.utilityIntervals))) == 1:
        for i in range(numOfIntervals):
            availablePositions.append(1)
    else:
        lengths = self.countLenghtOfAllLists(self.utilityIntervals)
        minLength = min(lengths)
        for i in range(numOfIntervals):
            if lengths[i]==minLength:
                availablePositions.append(1)
            else:
                availablePositions.append(0)
    return availablePositions
```

Zdroj: vlastní

Nyní bude dále pokračovat popis metody zobrazené na obrázku číslo 35. Výsledek volání výše popsané funkce **findFreeIntervals** je uložen do proměnné **availableIntervals**. Dále je počítán počet intervalů, do kterých může být generována otázka a tato hodnota je ukládána do proměnné **numberOfFreeIntervals**. Následně je generováno náhodné číslo prostřednictví metody **generateRandomNumb** v intervalu od 0 do hodnoty uložené v proměnné **numberOfFreeIntervals**-1. Dále je do proměnné **positionOfInterval** ukládána pozice náhodně zvoleného intervalu. Je zde náhodně vybírán jeden z volných intervalů. Následně je počítána velikost jednoho intervalu a hodnota je ukládána do proměnné **part**. Zde je velikost počítána jako *1/celkové množství intervalů (výchozí stav je 5).* V dalším kroku je počítán začátek intervalu, který je počítán jako *part * positionOfInterval*. Konec intervalu je stanoven jako: hodnota uložená v proměnné **part** krát hodnota uložená v proměnné **positionOfInterval**-0.01. Dále je zde tvořena pomocná proměnná **utilites**, která zajišťuje uchovávání všech momentálně získaných užitků v tabulce.

Dále je volána metoda **findMostSuitableCombinations**, která je zobrazena na obrázku číslo 37. Tato metoda je používána k nalezení vhodné kombinace pravděpodobnosti a užitků **y1** a **y2** takovým způsobem, aby bylo zaručeno, že výsledný bod je zařazen do vybraného intervalu, který byl náhodně zvolen v předchozích krocích. Jsou zde postupně zkoušeny různé kombinace všech užitků a pravděpodobností (od 0.05 do 1) s tím, že hodnota pravděpodobnosti je postupně zvyšována o 0.05. Jsou zkoušeny i reverzní hodnoty užitku. Pokud je zjištěno, že momentální kombinace je vyhovující chtěnému intervalu, tak je momentální kombinace vložena do listu **possibleCombinations**. Za předpokladu, že není nalezena ani jedna vhodná kombinace, tak je vygenerována výchozí náhodná otázka, u které je zaručeno, že bude umístěna do vhodného intervalu. U těchto výchozích otázek jsou vždy používány hodnoty užitku 0 a 1 (a korespondující hodnoty zisku). K tomuto se používá metoda **generateValueQuestionWithPosition** a ke generování pravděpodobnosti je užívána metoda **generateProbabilityFromInterval**. Z metody **findMostSuitableCombinations** je

vracen list **possibleCombinations**.

Na konci metody, která je zobrazena na obrázku číslo 35, dochází k vygenerování dalšího náhodného čísla, tentokrát v rozmezí od 0 do počtu nalezených kombinací. Z metody **generateValueBasedQuestionWithIntervals** je následně vracena jedna náhodně zvolená kombinace hodnot na základě hodnoty uložené v proměnné **randomNumber** a pozice intervalu. Na základě této kombinace hodnot je následně generována finální otázka uživateli.

Obrázek 37: Metoda findMostSuitableCombinations

```
probParameter = 0.05
length = (len(utilities)-1)
for i in range(len(utilities)):
    randomNumber = self.generateRandomNumb(0, length)
    randomNumber2 = self.generateRandomNumb(0, length)

    while randomNumber == randomNumber2:
        randomNumber2 = self.generateRandomNumb(0, length)

    for i in range(20):
        p1 = probParameter*i
        y1 = utilities[randomNumber]
        y2 = utilities[randomNumber2]
        testy1 = self.countTestUtility(y1, y2, p1)
        testy2 = self.countTestUtility(y2, y1, p1)

        if testy1 <= iend and testy1 >= istart:
            question = self.generator.generateValueQuestionWithPositions("-1", "-1", round(p1, 2), y1, y2, randomNumber, randomNumber2)
            possibleCombinations.append(question)

        if testy2 <= iend and testy2 >= istart:
            question = self.generator.generateValueQuestionWithPositions("-1", "-1", round(p1,2), y2, y1, randomNumber2, randomNumber)
            possibleCombinations.append(question)

if len(possibleCombinations)==0:
    # ADD DEFAULT VALUE
    if istart==0: question = self.generator.generateValueQuestionWithPositions("-1", "-1", self.generateProbabilityFromInterval(0.05,0.19), 0, 1, 0, len(utilities)-1)
    elif istart==0.2: question = self.generator.generateValueQuestionWithPositions("-1", "-1", self.generateProbabilityFromInterval(0.2,0.39), 0, 1, 0, len(utilities)-1)
    elif istart==0.4: question = self.generator.generateValueQuestionWithPositions("-1", "-1", self.generateProbabilityFromInterval(0.4,0.59), 0, 1, 0, len(utilities)-1)
    elif istart==0.6: question = self.generator.generateValueQuestionWithPositions("-1", "-1", self.generateProbabilityFromInterval(0.6,0.79), 0, 1, 0, len(utilities)-1)
    elif istart==0.8: question = self.generator.generateValueQuestionWithPositions("-1", "-1", self.generateProbabilityFromInterval(0.8,0.99), 0, 1, 0, len(utilities)-1)
    else: question = self.generator.generateValueQuestionWithPositions("-1", "-1", self.generateProbabilityFromInterval(0.05,0.19), 0, 1, 0, len(utilities)-1)
    possibleCombinations.append(question)

return possibleCombinations
```

Zdroj: vlastní

Na obrázku číslo 38 lze vidět část metody **confirmQuestion**, která je používána k zachytávání odpovědí uživatele na otázky, validaci dat, odbarvení řad a nastavováním dalších parametrů pro funkcionalitu algoritmu. Podstatné je, že na základě typu otázky se program větví na VALUE_BASED a PROBABILITY_BASED větve.

Pokud je otázka typu VALUE_BASED, tak jsou odbarveny jen dva náhodně vybrané řádky v předchozích krocích. Dále odpověď uživatele je převedena na datový typ float a je uložena do instanční proměnné X objektu **question**. Následně je počítán užitek pro zisk X pomocí metody **countUtility** na základě uložených hodnot v objektu **Question**. V poslední řadě je vložen nový řádek do tabulky, který odpovídá novému nalezenému bodu užitkové funkce peněz uživatele (proměnná X=zisk, vypočítaný užitek pomocí metody **countUtility**), který je ihned viditelný uživateli. Metoda **countUtility** je zobrazena na obrázku 26.

Pokud je otázka typu PROBABILITY_BASED, program pokračuje druhou větví. Zde je opět provedena kontrola vstupu a odbarvování řad. Hlavním rozdílem je, že je zde nastavována instanční proměnná **p1** objektu **question**, která je používána k uchovávání

odpovědi uživatele na vygenerovanou otázku. Následně je zde opět počítán užitek z uložených dat v objektu **question**. V posledním kroku této větve je volána metoda **updateRow**, která aktualizuje vybraný řádek. Je zde provedeno doplnění užitku v tabulce k náhodně zvolenému zisku **X**. Výpočet užitku je proveden metodou **countUtility**.

V obou větvích je prováděno vkládání vypočítané hodnoty užitku do odpovídajícího intervalu. Je vkládána hodnota spočítaného užitku do listu **utilityIntervals** na pozici konkrétního intervalu.

Pro oba typy otázek je na konci metody zvětšována proměnná **counter** o hodnotu jedna a je volána metoda **generateQuestions**, která pokračuje v generování dalších otázek.

Obrázek 38: Zachycení odpovědí na náhodně vygenerované otázky

```
if question.type == QuestionType.VALUE_BASED:
    if not re.match('(^[+]?([0-9]+)(\.[0-9]+)?)$', answer):
        showdialog("Chyba čísla", "Zadaný vstup musí být kladné číslo, opravte prosím svojí hodnotu.")
        return 0
    # insert new row
    self.colorRow(self.currentRows[0], QtGui.QColor(255,255,255))
    self.colorRow(self.currentRows[1], QtGui.QColor(255,255,255))
    question.X = float(answer)
    question.countUtility()
    self.insertNewRow(question)
    self.utilityIntervals[self.positionOfCurrentInterval].append(question.utility)
else:
    # update row
    try:
        value = float(answer)
    except:
        showdialog("Chyba pravděpodobnosti", "Zadaný vstup není ve tvaru tvar pravděpodobnosti. Prosím opravte hodnotu.")
        return 0
    if not 0 <= value <= 1:
        showdialog("Chyba pravděpodobnosti", "Zadaný vstup není v intervalu 0-1. Prosím opravte hodnotu.")
        return 0

    self.colorRow(self.currentRows[0], QtGui.QColor(255,255,255))
    self.colorRow(self.currentRows[1], QtGui.QColor(255,255,255))
    self.colorRow(self.currentRows[2].position, QtGui.QColor(255,255,255))
    question = self.questions[self.counter-1]
    question.p1 = float(answer)
    question.countUtility()
    self.updateRow(question)
    self.positionOfCurrentInterval = self.findInterval(question.utility)
    self.utilityIntervals[self.positionOfCurrentInterval].append(question.utility)
```

Zdroj: vlastní

Osmou a devátou část obecného algoritmu zajišťuje část kódu metody **generateQuestion**, která je zobrazena na obrázku číslo 39. Je zde kontrolováno, zda proměnná **counter** obsahuje hodnotu odpovídající vybranému množství otázek uživatelem. Pokud ano, tak je nejdříve zkontrolováno, zda je dosaženo rovnoměrného rozložení bodů do všech intervalů prostřednictví metody **checkEvenDistribution**. Tato metoda kontroluje, zda v každém intervalu je obsaženo stejné množství bodů. Pokud není, je rovnou spočítáno množství dodatečných otázek, které budou uživatelovi položeny. Pokud je dosaženo rovnoměrné rozložení, tak je zobrazena konečná obrazovka prostřednictví metody

showFinalScreen a je ukončeno dotazování.

Obrázek 39: Kontrola intervalů a ukončení

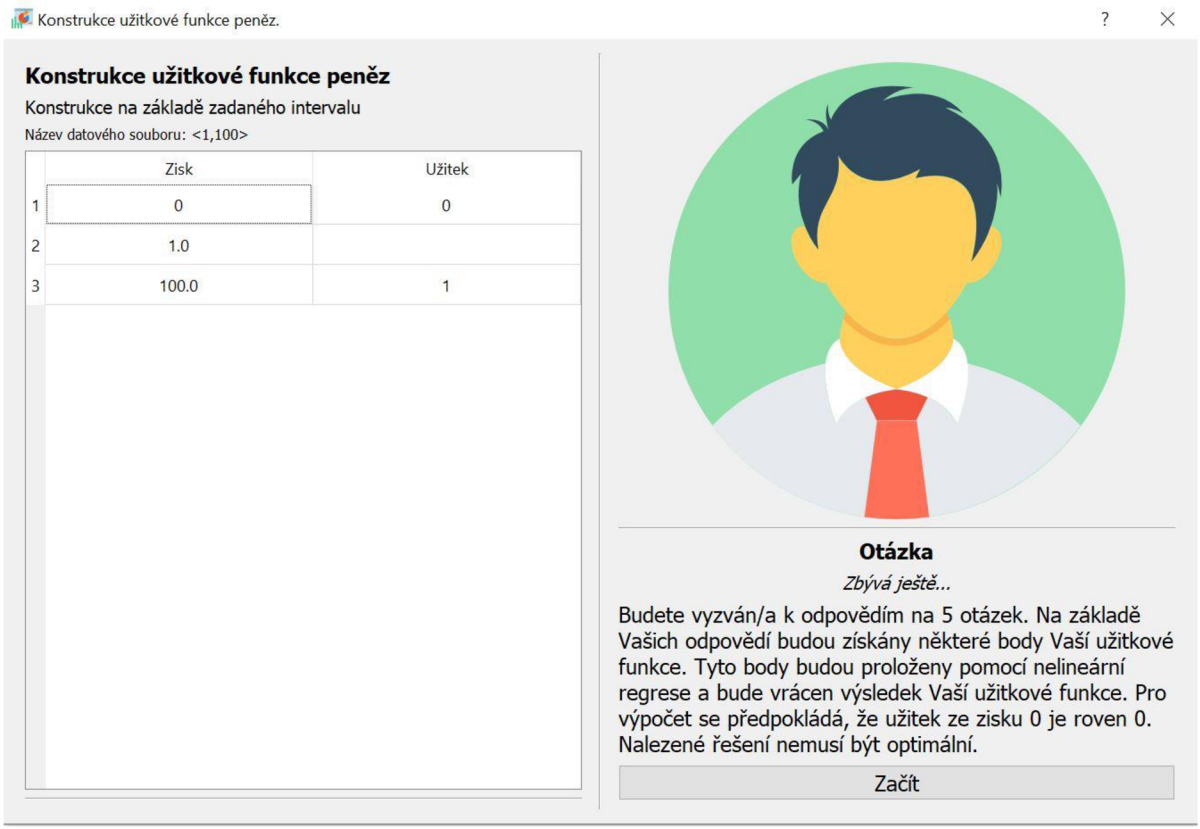
```
if self.counter ==  self.numOfQuestions+1:
    # zkontroluj rovnoměrnost
    isEvenDistr, additionalQuestions = self.checkEvenDistribution()

    if isEvenDistr:
        self.showFinalScreen()
        return 0
    else:
        self.isExtension=True
        self.numOfQuestions = self.numOfQuestions+additionalQuestions
        showdialog("Upozornění - další otázky", "K zajištění rovnoměrného získání bodů bude vygenerováno dodatečných " + str(additionalQuestions) + " hodnotových otázek.")
```

Zdroj: vlastní

6.3.2.2.3 Grafické rozhraní pro generování otázek

Na obrázku číslo 40 lze vidět úvodní grafické rozhraní pro generování otázek, kde je zobrazena úvodní zpráva na pravé straně a vytvořená tabulka na levé straně. Na pravé straně jsou generovány všechny typy zpráv a otázek. Tabulka na levé straně je aktualizována na základě odpovědích uživatele na otázky. Tabulka obsahuje dva sloupce – Zisk a Užitek. Tabulka je seřazena vzestupně dle zisku a nejnižší hodnotě zisku je přiřazen užitek 0. Nejvyšší hodnotě zisku užitek 1.

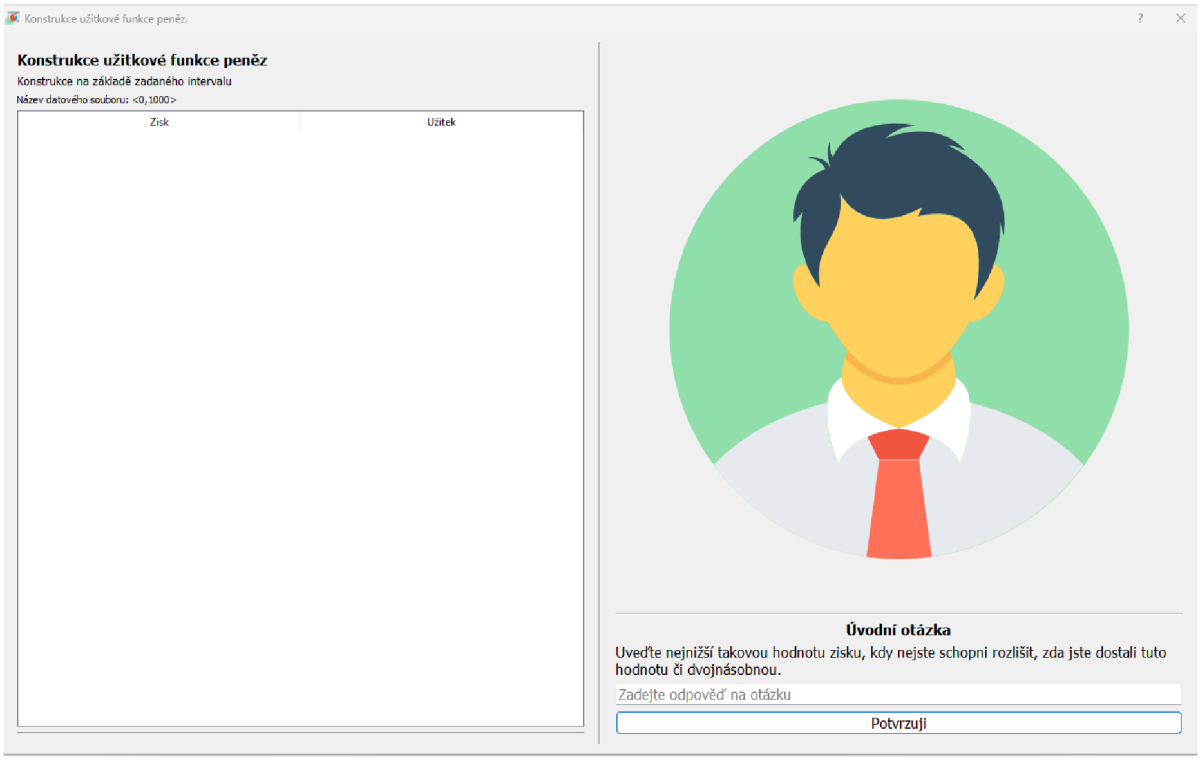
Obrázek 40: Úvodní grafické rozhraní pro generování otázek



Zdroj: vlastní

Na obrázku číslo 41 lze vidět grafické rozhraní pro otázku na maximální hodnotu M. Po kliknutí na tlačítko „Potvrzuji“ aplikace začne generovat náhodné otázky.

Obrázek 41: Grafické rozhraní pro M otázku

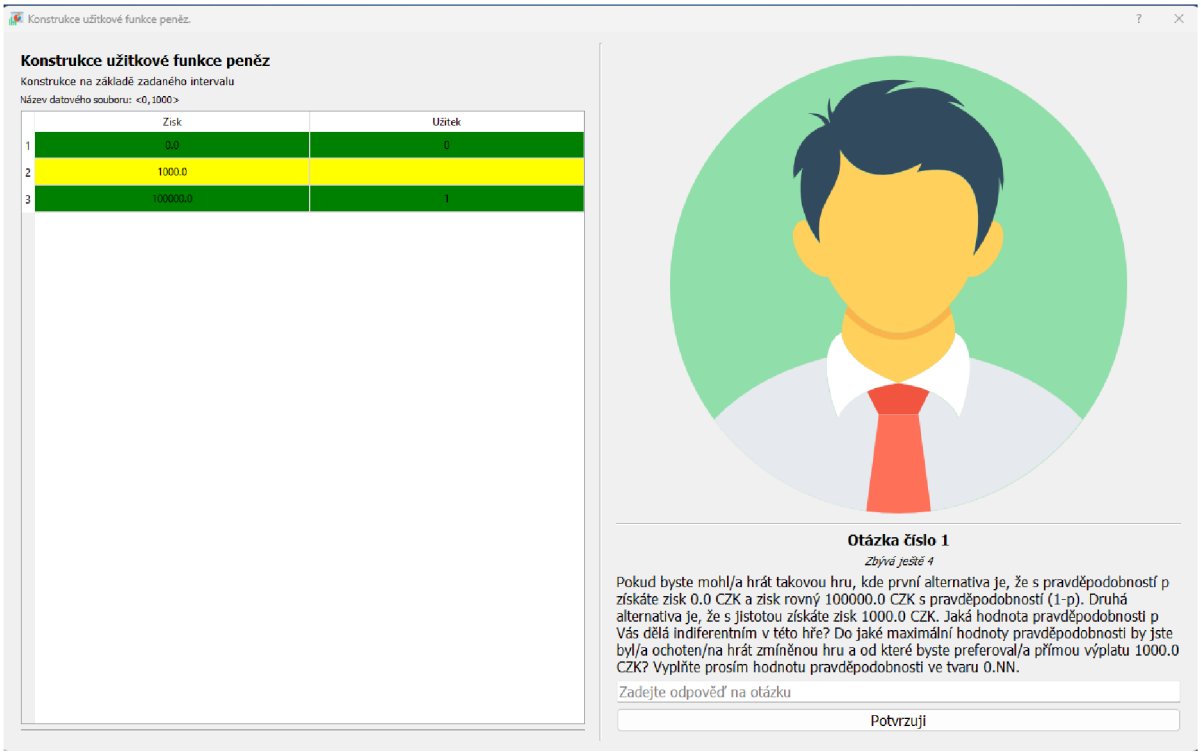


Zdroj: vlastní

Obrázek 42 již reprezentuje náhodně generovanou otázku typu PROBABILITY_BASED. Jsou vybrány náhodně dva zisky s přiřazenou hodnotou užitku. Řádky obsahující tyto náhodně vybrané zisky jsou označeny v tabulce zeleně. Žlutě je zobrazený řádek, který obsahuje náhodně vybraný zisk, který nemá přiřazenou hodnotu zisku. Užitek žlutě obarveného řádku je právě zjišťován zvolenou otázkou. Otázka na levé straně je ve formě, jak byla popsána v kapitole 6.3.2.1. U tohoto typu otázky se očekává, že uživatelova odpověď je ve formátu 0.NN. Na základě této odpovědi je proveden výpočet a tabulka na levé straně je aktualizována dopočítaným užitkem ve žlutém řádku.

Uživatel je průběžně informován, u jaké otázky se nachází a kolik otázek zbývá.

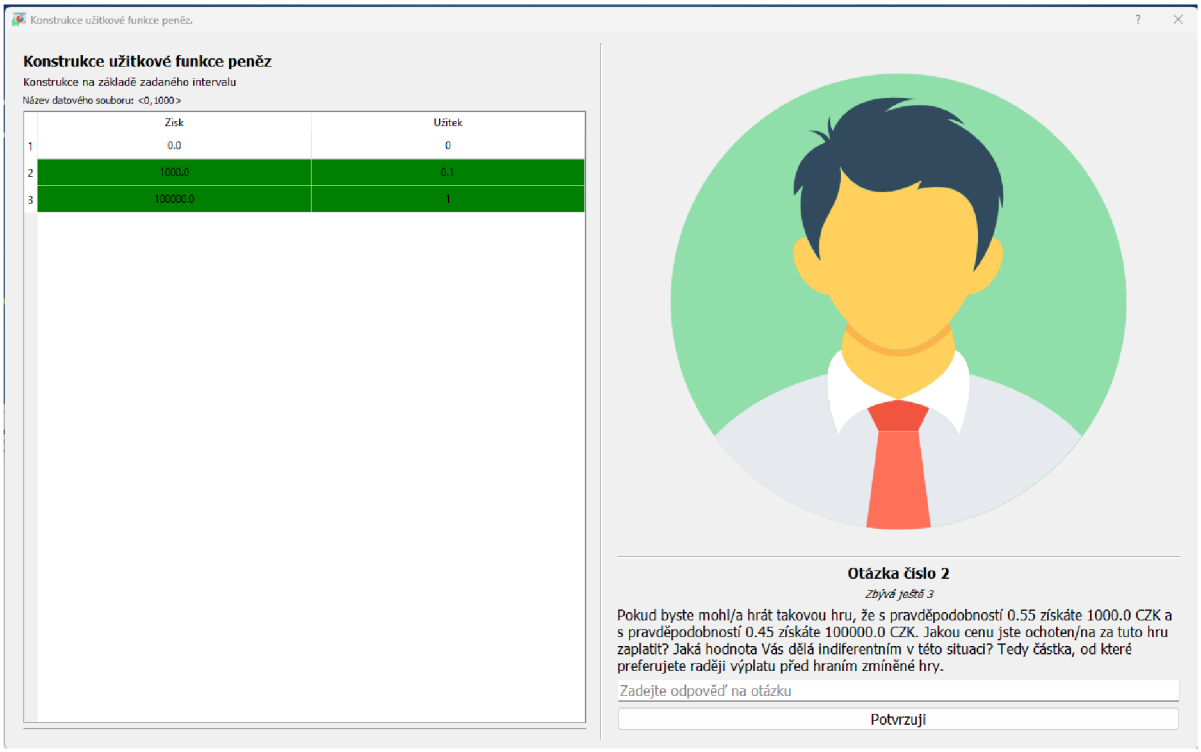
Obrázek 42: Grafické rozhraní – PROBABILITY_BASED otázka



Zdroj: vlastní

Na obrázku číslo 43 lze vidět grafické rozhraní pro náhodně vygenerovanou otázku typu VALUE_BASED. V rámci této otázky jsou zeleně obarveny řádky, které obsahují náhodně zvolené dva zisky. Na základě uživatelovi odpovědi je vytvořen nový seřazený řádek tabulky, který je složen z odpovědi uživatele na danou otázku (hodnota zisku) a s odpovídajícím vypočítaným užitkem.

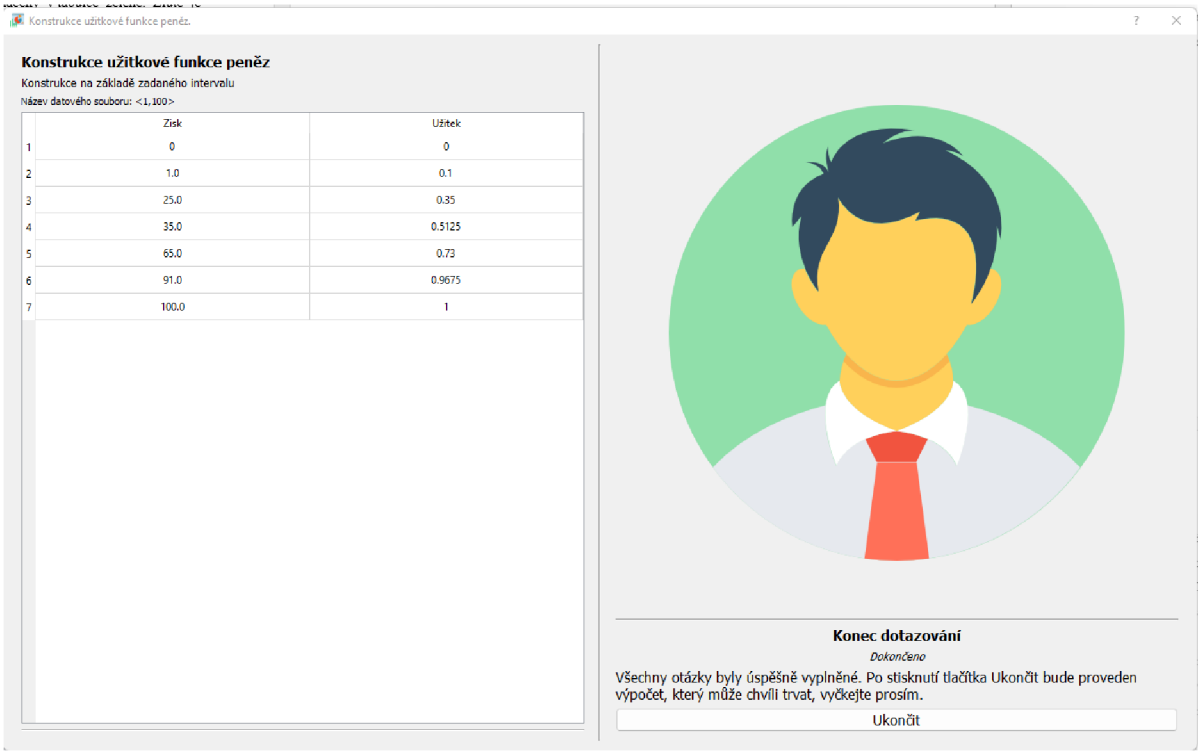
Obrázek 43: Grafické rozhraní pro VALUE_BASED otázku



Zdroj: vlastní

Po úspěšném zodpovězení vybraného množství otázek je dotazování ukončeno. Grafické rozhraní po koncovou zprávu a konečnou tabulku lze vidět na obrázku číslo 44. Na levé straně je aktualizovaná tabulka po zodpovězení všech náhodně vygenerovaných otázek. Plně zaplněné řádky v tabulce reprezentují body užitkové funkce peněz rozhodovatele. Na základě plně zaplněných řádku v této tabulce je následně prováděna optimalizace, která bude popsána v následující kapitole.

Obrázek 44: Grafické rozhraní pro konec dotazování



Zdroj: vlastní

6.3.2 Optimalizace pomocí Metody nejmenších čtverců

V rámci kapitoly číslo 6.3.1 jsou získávány různé body užitkové funkce peněz uživatele. Tyto body jsou získávány na základě odpovědí uživatele na náhodně generované otázky. Na základě odpovědích uživatele jsou dopočítávány různé body užitkové funkce peněz uživatele.

Jakmile jsou k dispozici tyto body, tak je nutné tyto nalezené body proložit vhodnou funkcí, která nejlépe prokládá získané body – tou je užitková funkce peněz zobrazená v rovnici číslo 23. Cílem je nalézt takové hodnoty parametrů a, b, c funkce zobrazené rovnicí (23), aby tato funkce nejlépe proložila naše nalezené body. Je zde řešen problém nelineární regrese a je používána Metoda nejmenších čtverců, která je založena na minimalizaci rovnice (24).

Pro výpočet nelineární Metody nejmenších čtverců je použita knihovna **scipy** společně s vědeckou matematickou knihovnou **numpy**.

6.3.2.1 Softwarová implementace

Třída **LeastSquaresOptimalization**, která se nachází ve složce **utils**, je zodpovědná za provádění optimalizace a za hledání ideálních počátečních parametrů regrese.

Na obrázku číslo 45 je zobrazena první část třídy **LeastSquaresOptimalization**. Tato třída přijímá v konstruktoru dva listy – list zisků a list korespondujících užitků. Na základě těchto vstupních dat je prováděna optimalizace.

Dále je zde definována metoda **function**, která definuje funkci, pomocí které je prováděno prokládání bodů. Funkce v této metodě je zcela odpovídající funkci definované v rovnici (23). Odpovídající je i definice parametrů a, b, c.

Metoda **residual** je plně odpovídající funkci zobrazené v rovnici (24). Význam jednotlivých proměnných je následující:

- **y** = list užitků, který je získán v krocích popsaných v kapitole 6.3.2,
- **self.function(params, x)** = volání metody **function** se vstupem odpovídající parametrům této funkce (**params**=a, b, c) a **x** je list odpovídajících zisků, který je získán v krocích popsaných v kapitole 6.3.2.

Obrázek 45: První část třídy LeastSquaresOptimalization

```
class LeastSquaresOptimalization:
    def __init__(self, profitList, utilityList):
        self.profitList = np.asarray(profitList)
        self.utilityList = np.asanyarray(utilityList)

    def function(self, params, x):
        a = params[0]
        b = params[1]
        c = params[2]

        return a * np.exp(-x/b) + c

    def residual(self, params, x, y):
        return (y - self.function(params, x))**2

    def countSSE(self, result):
        sum = np.sum(result.fun)
        return sum
```

Zdroj: vlastní

Na obrázku číslo 46 je zobrazena metoda **startOptimalization**, kde je prováděna samotná optimalizace. Vstupní parametry této funkce jsou:

- **a_start** = počáteční hodnota pro parametr a,
- **b_start** = počáteční hodnota pro parametr b,
- **c_start** = počáteční hodnota pro parametr c.

Dále je v této metodě využívána proměnná **x0**, která je zodpovědná za vytváření pole na základě vstupních počátečních hodnot parametrů.

Pomocí proměnné **bnds** jsou definovány dolní a horní meze pro jednotlivé parametry. Levé pole obsahuje dolní meze pro parametry a pravé pole obsahuje horní meze pro jednotlivé parametry. Pořadí parametrů je a, b, c. Je evidentní, že platí následující meze pro jednotlivé parametry:

- a: od mínus nekonečna do -0. 0000000000001,
- b: 0. 0000000000001 do nekonečna,
- c: od mínus nekonečna do nekonečna.

V dalším kroku je již volána funkce **least_squares**, která je nabízena knihovnou **scipy**. Tato metoda má následující vstupní parametry:

- **self.residual** = metoda **residual**, která byla popsána výše,
- **x0** = počáteční hodnoty parametrů a, b, c,
- **args** = list zisků a korespondujících užitků, které byly získány prostřednictví náhodně vygenerovaných otázek,
- **bounds** = definice dolních a horních mezích jednotlivých parametrů.

Obrázek 46: Metoda startOptimalization

```
def startOptimalization(self, a_start, b_start, c_start):
    x0 = np.array([a_start, b_start, c_start])
    bnds = ([-np.inf, 0.0000000000001, -np.inf], [-0.0000000000001, np.inf, np.inf])
    result = least_squares(self.residual, x0, args=(self.profitList, self.utilityList), bounds=bnds, xtol=1e-8, ftol=1e-8)
    SSE = self.countSSE(result)
    pointList = self.countPoints(result)
    return result, SSE, pointList
```

Zdroj: vlastní

Po zavolání funkce **least_squares** jsou výsledky optimalizace uložené v objektu **result**. Dále je dopočítávána hodnota SSE prostřednictví metody **countSSE**. Metoda **countPoints** je určena ke kalkulaci konkrétních hodnot užitků za pomoci optimalizovaných nalezených hodnot parametrů a, b, c. Finální výsledek optimalizace, nalezené SSE a výsledek metody **countPoints** jsou vraceny z metody. Tyto vracené hodnoty jsou nadále využívány v rámci vykreslování výsledků. Tato problematika bude vysvětlena v kapitole 6.4.

6.3.2.2 Volba vhodných počátečních parametrů

V rámci této kapitoly bude popsána problematika volby vhodných počátečních parametrů. V první kapitole bude uvedena logika, jak jsou parametry voleny a v druhé kapitole bude popsána softwarová implementace této problematiky.

6.3.2.1 Logika volby vhodných počátečních parametrů

Je nutné najít vhodné počáteční parametry a, b, c zobrazené v rovnici (23). Je předpokládáno, že zisk rovný nule odpovídá užitku nula. Díky tomuto předpokladu je možné dosadit do rovnice (23) zisk rovný nule a užitek rovný nule. Po úpravě této rovnice za

zmíněných podmínek platí následující vztah, který je využíván pro odhad **parametru c**:

$$c = -a, \tag{26}$$

kde **c** i **a** jsou parametry funkce zobrazené v rovnici (23).

Parametr a je nastavován v intervalu <1, 1.2>. Důvodem je, že užitek dosahuje maximálně hodnoty rovné jedna a lze předpokládat, že funkce může tuto hodnotu nepatrně překročit.

K výpočtu **parametru b** je nutné získat maximální hodnotu M uživatele pomocí otázky popsané v kapitole 6.3.2.1 (M_BASED otázka). Na základě získané maximální hodnoty M jsou počítány odhady parametrů pro horní a dolní hranici parametru b. Rovnice (27) zobrazuje způsob, jakým dochází k výpočtu spodní hranice parametru b. Rovnice (28) popisuje výpočet horní hranice parametru b.

$$mindolníhraniceb = \frac{M}{5} \tag{10}$$

$$maxhorníhraniceb = \frac{M}{3} \tag{11}$$

M je maximální hodnota získaná prostřednictví otázky na maximální hodnotu M. Tato hodnota ukazuje, kde je již užitková funkce celkem ustálená. Rozdíl mezi M a 2M je velmi malý.

Parametr b udává sklon dané užitkové funkce. Cílem je nastavit vhodné rozpětí, ve kterém je vyhledávána optimální startovací hodnota pro tento parametr. Pokud by žádné rozpětí nebylo stanoveno, musel by být procházen celý možný prostor. To není pro výpočet ideální. Je tedy potřebné nastavit rozpětí tak, aby nenastaly následující situace:

- užitková funkce se dostane velmi blízko k hranici definovanou parametrem c moc brzy. Je požadováno, aby se funkce dostala velmi blízko této hranici pro zvolenou maximální hodnotu M,
- užitková funkce poroste příliš pomalu.

V tomto případě lze konstatovat, že typ sklonu užitkové funkce je dán, protože parametry a, c užitkové funkce jsou téměř fixní. Je tedy zafixováno, že funkce prochází nulou, že funkce roste k hodnotě jedna a také typ sklonu užitkové funkce. Dále platí, že čím je menší parametr b, tím rychleji funkce roste k hodnotě jedna. Logika volby minimální a maximální hodnoty pro parametr b je založena na tom, že hledáme takové hodnoty, aby v bodě M nebyla hodnota užitkové funkce příliš brzy u jedné (výrazně menší hodnoty než M mají být pro

rozhodovatele rozlišitelné).

Na obrázku číslo 47 je zobrazena tabulka ukazující posuny mezi násobky. Sloupec **x** reprezentuje zisk, který je vyjádřen ve formě 1, 10, 100… 10^{17}. Sloupce pojmenované **fce(100)**, **fce(1000)**, **fce(10000)**, **fce(100000)** zobrazují výsledky užitkové funkce pro různé násobky parametru b**.** Odpovídající hodnota pro parametr b je definována hodnotou v závorce. Z tohoto obrázku je evidentní, že pro násobky parametru b platí, že mezi nimi je vždy posun o jedno místo dolů. Hodnoty ve sloupcích pojmenovaných **fce** se postupně posouvají vždy o jedno místo dolů při postupu z levé strany. Tento jev popsaný výše vychází z toho, že se jedná o exponenciální funkci. Vzhledem k tomu, že je definováno, že užitková funkce prochází nulou, tak je možné nastavovat minimální a maximální hranici pro parametr b dle této logiky.

Obrázek 47: Posun

x	fce(100)	fce(1000)	fce(10000)	fce(100000)
1	0,009950166	0,0009995	9,9995E-05	9,99995E-06
10	0,095162582	0,009950166	0,0009995	9,9995E-05
100	0,632120559	0,095162582	0,009950166	0,0009995
1000	0,9999546	0,632120559	0,095162582	0,009950166
10000	1	0,9999546	0,632120559	0,095162582
100000	1	1	0,9999546	0,632120559
1000000	1	1	1	0,9999546
10000000	1	1	1	1
100000000	1	1	1	1
1000000000	1	1	1	1
10000000000	1	1	1	1
1E+11	1	1	1	1
1E+12	1	1	1	1
1E+13	1	1	1	1
1E+14	1	1	1	1
1E+15	1	1	1	1
1E+16	1	1	1	1
1E+17	1	1	1	1

Zdroj: vlastní

Na obrázku číslo 48 je zobrazena rozšířená tabulka vycházející z tabulky zobrazené na obrázku číslo 47. V rámci obrázku číslo 48 je řešena volba minimální a maximální hodnoty pro parametr. Jsou hledány takové hodnoty, aby v bodě M nebyla hodnota užitkové funkce příliš brzy u jedné. Zvolená minimální hodnota je na obrázku zobrazena červeně, maximální hodnota zeleně.

Obrázek 48: Volba maximální a minimální hodnoty pro parametr b

x	fce(100)	fce(1000)	fce(10000)	fce(100000)
1	0,009950166	0,0009995	9,9995E-05	9,99995E-06
10	0,095162582	0,009950166	0,0009995	9,9995E-05
50	0,39346934	0,048770575	0,004987521	0,000499875
100	0,632120559	0,095162582	0,009950166	0,0009995
175	0,826226057	0,160542979	0,017347764	0,00174847
250	0,917915001	0,221199217	0,024690088	0,002496878
312,5	0,956063066	0,268384371	0,030766766	0,003120122
375	0,976482254	0,312710721	0,036805582	0,003742978
500	0,993262053	0,39346934	0,048770575	0,004987521
1000	0,9999546	0,632120559	0,095162582	0,009950166
5000	1	0,993262053	0,39346934	0,048770575
10000	1	0,9999546	0,632120559	0,095162582
15000	1	0,999999694	0,77686984	0,139292024
20000	1	0,999999998	0,864664717	0,181269247
30000	1	1	0,950212932	0,259181779
40000	1	1	0,981684361	0,329679954
50000	1	1	0,993262053	0,39346934
100000	1	1	0,9999546	0,632120559
1000000	1	1	1	0,9999546
10000000	1	1	1	1
100000000	1	1	1	1
1000000000	1	1	1	1
10000000000	1	1	1	1
1E+11	1	1	1	1
1E+12	1	1	1	1
1E+13	1	1	1	1
1E+14	1	1	1	1
1E+15	1	1	1	1
1E+16	1	1	1	1
1E+17	1	1	1	1

Zdroj: vlastní

6.3.2.2 Softwarová implementace volby vhodných počátečních parametrů

Kód, který je zodpovědný za hledání optimálních vhodných počátečních parametrů se nachází ve třídě **LeastSquareOptimalization.**

Metoda **estimateInitialParameters** je používána pro získání optimálních počátečních parametrů. Tato metoda je zobrazena na obrázku číslo 49. Tato metoda přijímá vstupní parametr M, který odpovídá maximální hodnotě, která je získávána na základě odpovědi uživatele na danou otázku. Dále jsou zde počítány hodnoty odpovídající maximální horní hranici a minimální dolní hranici parametru b způsobem popsaným rovnicí (27) a rovnicí (28). Je počítán nejlepší výsledek (tedy nejmenší SSE) pro maximální horní odhad parametru b a minimální dolní odhad parametru b prostřednictví metody **combineParameters**, která je zobrazena na obrázku číslo 50.

Obrázek 49: Metoda estimateInitialParameters

```
def estimateInitialParameters(self, M):
    bUpperBoundary = M/3
    bLowerBoundary = M/5
    bUpResult = self.combineParameters(bUpperBoundary)
    bDownResult = self.combineParameters(bLowerBoundary)
    optimumResult = self.estimateInitialParamsWithComparison(bUpResult, bDownResult, 30)
    return optimumResult
```

Zdroj: vlastní

Metoda **combineParameters** je používána k nalezení nejlepšího výsledku pro různé kombinace parametrů se zadaným vstupním parametrem b. Je zde volána cyklicky metoda **startOptimalization** se všemi možnými hodnotami parametru a, nastaveným odhadem pro parametr b (proměnná **value**) a parametrem c, který je zde nastavován jako $c=-a$. Tento výpočet proběhne pro všechny možné hodnoty v listu **A_POSSIBLE_VALUES_LIST**. Výsledky jednotlivých volání jsou ukládány do proměnné **tempArray**. Po ukončení cyklu je nalezen výsledek s minimálním SSE a ten je vracen z této metody.

Obrázek 50: Metoda combineParameters

```
def combineParameters(self, value):
    tempArray = []
    for a in ConstantManager.A_POSSIBLE_VALUES_LIST:
        c=-a
        bEstimate, foundSSE, list_u = self.startOptimalization(a, value, c)
        tempArray.append(OptimResult(bEstimate, foundSSE, value))
    result = min(tempArray)
    return min(tempArray)
```

Zdroj: vlastní

Nyní je pokračováno vysvětlováním metody zobrazené na obrázku číslo 49. Pomocí metody **combineParameters** jsou počítány nejlepší výsledky pro maximální a minimální odhady parametru b (proměnné **bUpResult**, **bDownResult**). V dalším kroku je volána metoda **estimateInitialParamsWithComparison**, která je zobrazena na obrázku číslo 51. Tato metoda je používána k nalezení optimální startovací hodnoty pro parametr b. Proces vyhledávání optimální počáteční hodnoty pro parametr b zahrnuje rekurzivní volání této metody, dokud není nalezeno již dobré řešení, nebo je dokončeno maximální počet opakování.

Vstupní parametry této metody jsou následující:

- **resultUpperBoundary** = horní hranice,
- **resultLowerBoundary** = dolní hranice,
- **levels** = počet opakování.

Algoritmus, implementovaný metodou **estimateInitialParamsWithComparison**, je následující.

1. Pokud nalezené parametry pro horní hranici parametru b jsou stejné jako pro dolní hranici parametru b, je vyhledávání ukončeno a je vracen výsledek. Výsledek je vracen i v případě, pokud je dosaženo maximálního počtu opakování (proměnná **levels**).
2. Pokud nalezené parametry v bodě 1 nejsou ekvivalentní, je vytvořena nová hodnota odhadu parametru b, která je reprezentována proměnnou **newNodeEstimate**. Tato proměnná je počítána jako součet horní a dolní hranice, které byly zadány jako vstupní parametry do této metody a tento součet je vydělen dvěma.
3. Je hledán nejlepší výsledek pro nový odhad parametru b (proměnná **newNodeEstimate**) pomocí metody **combineParameters** se vstupním parametrem odpovídající tomuto novému odhadu.
4. Dále jsou porovnávány SSE jednotlivých hranic.
   4.1 Pokud SSE pro horní hranici je nižší nebo rovno SSE pro dolní hranici, tak je volána tato stejná metoda **estimateInitialParamsWithComparison** rekurzivně, ale s následujícími parametry: novou horní hranicí je horní hranice vložená jako vstupní parametr do této metody (proměnná **resultUpperBoundary**), dolní hranicí je výsledek získaný pomocí nového odhadu parametru b (proměnná **newNodeResult**). Vzhledem k tomu, že je volána opakovaně stejná metoda rekurzivně, tak algoritmus se vrací opět ke kroku číslo 1 s jinými zadanými vstupními parametry.
   4.2 Pokud SSE pro dolní hranici je nižší než SSE pro horní hranici, tak je volána rekurzivně stejná metoda **estimateInitialParamsWithComparison** s tím, že vstupní parametry jsou následující: horní hranicí je výsledek získaný pomocí nového odhadu parametru b (proměnná **newNodeResult**) a novou dolní hranicí je dolní hranice vložená do této metody jako vstupní parametr (proměnná **resultLowerBoundary**). Stejně jako v případě kroku 4.1 i zde se algoritmus vrací

ke kroku číslo 1 s jinými zadanými vstupními parametry.

Obrázek 51: Metoda estimateInitialParamsWithComparison

```
def estimateInitialParamsWithComparison(self, resultUpperBoundary, resultLowerBoundary, levels):
    if levels==0:
        return min(resultLowerBoundary, resultUpperBoundary)

    # Pokud jsou nalezené parametry stejné u dolní i horní hranice tak končíme vyhledávání a vracíme nalezené parametry
    if round(resultUpperBoundary.result.x[0],2) == round(resultLowerBoundary.result.x[0], 2)
    and round(resultUpperBoundary.result.x[1],2) == round(resultLowerBoundary.result.x[1], 2)
    and round(resultUpperBoundary.result.x[2],2) == round(resultLowerBoundary.result.x[2],2):
        return resultUpperBoundary
    else:
        newNodeEstimate = (resultUpperBoundary.bEstimate+resultLowerBoundary.bEstimate)/2
        print(newNodeEstimate)
        newNodeResult = self.combineParameters(newNodeEstimate)
        if resultUpperBoundary.SSE <= resultLowerBoundary.SSE:
            return self.estimateInitialParamsWithComparison(resultUpperBoundary, newNodeResult, levels-1)
        else:
            return self.estimateInitialParamsWithComparison(newNodeResult, resultLowerBoundary, levels-1)
```

Zdroj: vlastní

6.4 Výsledky

V rámci této kapitoly bude popsána softwarová implementace výsledků v aplikaci, vykreslování grafů a související grafické rozhraní aplikace.

6.4.1 Softwarová implementace výsledků

Pro přenos výsledků a důležitých kolekcí dat z předchozího kroku se používá třída **UtilityResult**, která je zobrazena na obrázku číslo 52. Tato třída uchovává výsledek regrese (objekt typu **OptimumResult**), vstupní parametry konstrukce, nalezené nejnižší SSE, kolekci zisků a odpovídající kolekci užitků.

Obrázek 52: UtiltyResult

```
class UtilityResult:
    def __init__(self, result, SSE, selectedData, profitList, utilityList, resList):
        self.result = result
        self.selectedData = selectedData
        self.SSE = SSE
        self.profitList = profitList
        self.utilityList = utilityList
        self.resList = resList
```

Zdroj: vlastní

Objekt typu **UtilityResult** je vrácen do hlavní části aplikace, kde se výsledky ukládají opět dle vygenerovaného ID do hashovací mapy s názvem **results**. Následně je opět vytvořen nový potomek k položce v menu s názvem „Výsledky“. Tento postup lze vidět na obrázku číslo 53.

Obrázek 53: Tvorba výsledku

```
if questionsDialogue.exec_():
    id = uuid.uuid4()
    fResult = questionsDialogue.result
    self.results[id] = fResult
    self.insertNewChildToParent(5, fResult.selectedData.data.name, id, QtGui.QIcon("pck/img/profits.png"))
    self.showResultArea(True)
    self.showResult(id)
```

Zdroj: vlastní

Na předchozím obrázku je volána metoda **showResult**, která na základě ID výsledku zobrazí výsledek společně se zkonstruovaným grafem. Tuto metodu lze vidět na obrázku číslo 54. Nejdříve je zde inicializována tabulka zobrazující zisky s odpovídajícími užitky spočítané v rámci procesu konstrukce užitkové funkce peněz. Jelikož z předchozích kroků je k dispozici malé množství nalezených bodů užitkové funkce rozhodovatele, je nutné další dopočítat, aby vykreslovaný graf byl hladký. Je zde proto prováděn výpočet dalších bodů užitkové funkce na základě nalezených optimalizovaných parametrů. Na základě těchto parametrů a funkce je dopočítáno dalších 200 rovnoměrně rozložených bodů, které zajišťují hladkost vykreslovaného grafu. Je toho docíleno tak, že je nejdříve spočtená hodnota do proměnné **steep**, která je používána k zajištění rovnoměrného vygenerování dalších hodnot zisků metodou **arrange**. Tato proměnná udává vzdálenost mezi dvěma sousedními hodnotami. Tato proměnná je počítána tak, že je zjištěno maximum a minimum z kolekce zisků a tyto dvě hodnoty jsou od sebe odečteny. Výsledná tato hodnota je dělena hodnotou 200, jelikož cílem je získat 200 rovnoměrných bodů. Následně je pomocí metody **arrange**, která je nabízena knihovnou **numpy** vygenerováno 200 rovnoměrně vygenerovaných hodnot zisků v intervalu: <*minimum (z kolekce zisků), maximum (kolekce zisků)*>. Metoda **arrange** používá vstupní parametr **step**. Hodnota proměnné **steep** je vkládána do metody **arrange** jako vstupní parametr **step**. Metoda **arrange** s parametrem **step** jsou určeny ke generování rovnoměrně rozložených hodnot od minimálního zisku v kolekci do maximální zisku v kolekci s tím, že vstupní parametr **step** odpovídá vzdálenosti mezi dvěma sousedními hodnotami.

Obrázek 54: Metoda showResult

```
def showResult(self, id):
    self.tableControls.hide()
    self.initResultTable()

    result = self.results[id]
    self.fillResultTable(result)
    self.setLabelsForResult(result)

    x = np.asarray(result.profitList)
    y = np.asarray(result.utilityList)
    lsq = LeastSquaresOptimalization(result.profitList, result.utilityList)
    steep = (max(result.profitList)-min(result.profitList))/200
    x2 = np.arange(min(result.profitList), max(result.profitList), steep)
    resList = lsq.countAdditionalPoints(result.result, x2)

    self.ax.clear()
    self.ax.plot(x, y, "o", label="Počáteční body")
    self.ax.legend(loc='lower right')
    self.ax.plot(x2, resList, label='Výsledná užitková funkce peněz')
    self.ax.legend(loc='lower right')
    self.ax.set_xlabel("Zisk", fontweight='bold')
    self.ax.set_ylabel("Užitek", fontweight='bold')
```

Zdroj: vlastní

Po vygenerování těchto rovnoměrně rozložených hodnot zisků je volána metoda **countAdditionalPoints**, která je zobrazena na obrázku číslo 55. Tato metoda je užívána k výpočtu korespondujících užitků k rovnoměrně vygenerovaným ziskům. Je toho docíleno voláním metody **function** s optimalizovanými hodnotami parametrů funkce. Vstupní proměnná **result** je parametr, který obsahuje nalezené optimální parametry a, b, c. Parametr **array** je list obsahující rovnoměrně vygenerované hodnoty zisků.

Obrázek 55: Metoda countAdditionalPoints

```
def countAdditionalPoints(self, result, array):
    res = self.function(result.x, array)
    return res
```

Zdroj: vlastní

V metodě **showResult** je vykreslován graf užitkové funkce peněz uživatele. Postup je možné vidět na obrázku číslo 54. Pro vykreslování grafu je používána knihovna **matplotlib**. Nejdříve jsou vykreslovány počáteční body, které byly získány na základě náhodně generovaných otázek nebo body, kterým byly již přiřazeny odpovídající užitky před samotným dotazováním. Proměnné **x** a **y** jsou používány k uchovávání známých bodů – tedy kolekce zisků a odpovídající kolekce užitků. Na X osu jsou vloženy hodnoty odpovídající proměnné **x**. Na Y osu jsou vkládány hodnoty odpovídající proměnné **y**. Tyto počáteční body jsou vykresleny jako modré body na grafu. Druhý podgraf je vytvářen za pomocí rovnoměrně vygenerovaných bodů. Na X osu je vložena proměnná **x2**, která odpovídá rovnoměrně

vygenerovaným hodnotám zisků. Na Y osu je vložena proměnná **resList**, která obsahuje dopočítané hodnoty užitků na základě rovnoměrně vygenerovaných hodnot zisků. Dále jsou v rámci grafu nastavovány další popisky os, legenda a barvy.

Mazání výsledků je zajištěno podobným způsobem jako mazání a aktualizace datových tabulek a intervalů. Je využívána hashovací mapa **results**, ve které jsou uloženy všechny momentální výsledky. Na základě aktuální zobrazené položky je známo ID vybrané položky. Následně je nalezena položka v této hashovací mapě dle vybraného ID a smazána. Zároveň je smazána korespondující položka v menu. Tento popsaný postup je zobrazen na obrázku 56. Aktualizace výsledku je omezená jen na změnu názvu výsledku. Uživatel si může změnit název výsledku dvojitým kliknutím na danou položku a přepsat název. Tato změna je detekována v aplikaci jako signál. Po obdržení signálu změny je aktualizována položka v hashovací mapě **results** a zároveň je aktualizován název položky v menu.

Uložení obrázku vytvořené užitkové funkce peněz uživatele je implementováno prostřednictvím knihovny **matplotlib**.

Obrázek 56: Metoda deleteResult

```
def deleteResult(self):
    res = self.showdialogWithButtons("Smazání výsledku", "Opravdu chcete smazat tuto užitkovou funkci?")
    if res == 0:
        return 0
    item = self.treeMenu.selectedItems()[0]
    selectedId = item.data(0, QtCore.Qt.UserRole)
    parent = item.parent()
    parent.removeChild(item)
    del self.results[selectedId]
    self.resetState()
    self.tableControls.show()
    self.hideMenuForInterval(True)
    self.showResultArea(False)
```

Zdroj: vlastní

6.4.2 Grafické rozhraní výsledků

Na obrázku číslo 57 je možné vidět grafické rozhraní části aplikace zobrazující výsledky. Do menu na levé straně jsou přidáváni nový potomci do položky „Výsledky“. Grafické rozhraní zobrazující výsledek je načteno po kliknutí na potomka pod položkou „Výsledky“. Uprostřed grafického rozhraní je umístěna tabulka, která obsahuje tři sloupce – Zisk, Užitek a Užitková funkce. Hodnoty ve sloupcích „Zisk“ jsou získávány na základě zadaných dat nebo na základě náhodně vygenerovaných otázek typu VALUE_BASED. Hodnoty ve sloupci „Užitek“ jsou kalkulovány postupem popsaným v kapitole 6.3.2. Kalkulace užitku je prováděna na základě obou typů náhodně generovaných otázek. Vypočtený užitek prostřednictvím funkce zobrazené v rovnici (23) s nalezenými optimálními

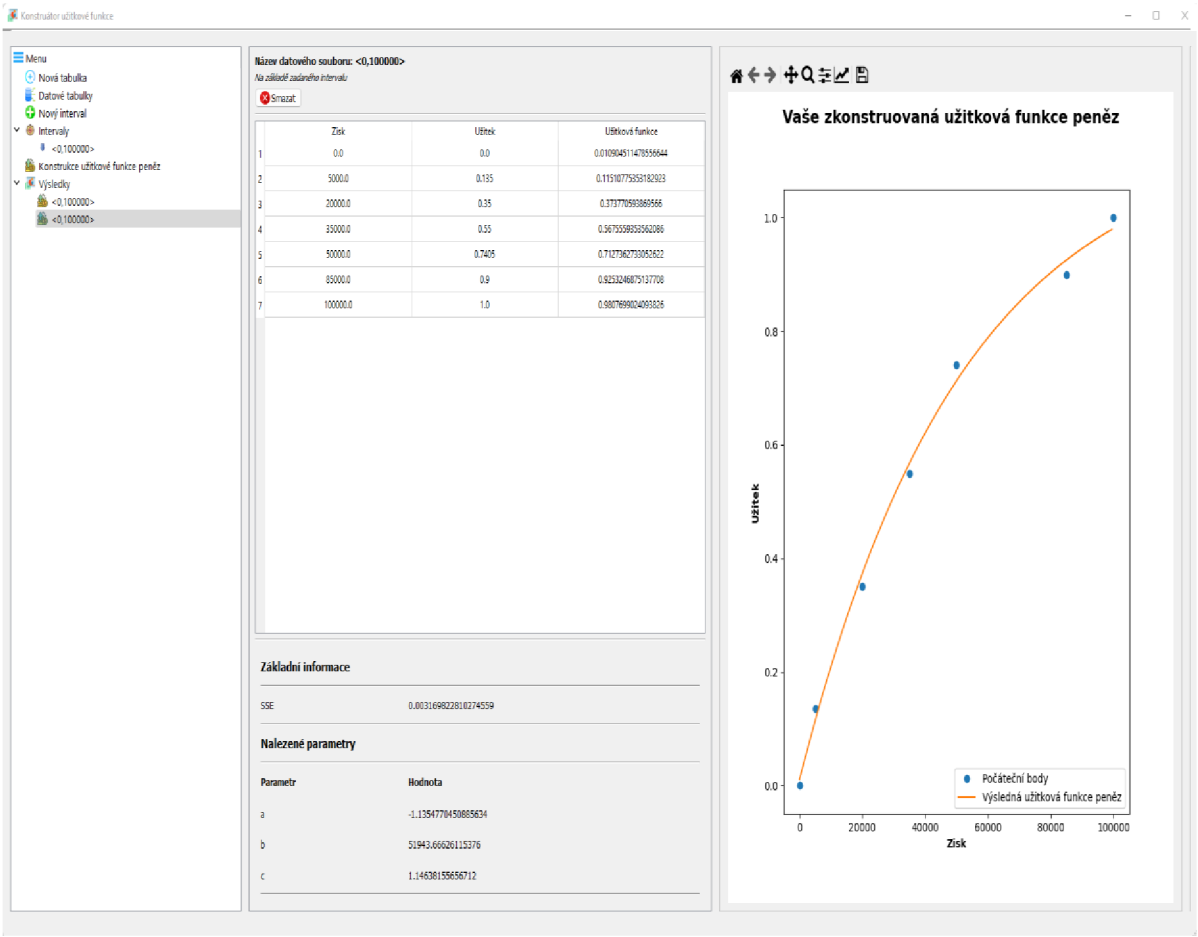
parametry a, b, c je zobrazen ve sloupci „Užitková funkce“. Uprostřed grafického rozhraní nad tabulkou je možné vidět název datového souboru, na základě kterého byla provedena konstrukce užitkové funkce peněz. Dále je zde uveden typ datového objektu a tlačítko „Smazat“, které je používáno ke smazání vybraného výsledku.

Pod výslednou tabulkou je uživatel informován ohledně nalezeného minimálního SSE a nalezených parametrů a, b, c.

Na pravé straně grafického rozhraní týkající se výsledků je zobrazena výsledná užitková funkce peněz uživatele. Na ose X je zisk, na ose Y je užitek. Modré tečky na grafu jsou body, které jsou uvedeny v tabulce ve sloupcích „Zisk“ a „Užitek“. Oranžová spojnice je kompletní nalezená užitková funkce peněz uživatele.

Je možné uložit graf užitkové funkce peněz uživatele prostřednictvím tlačítka, které je reprezentované černou disketou.

Obrázek 57: Grafické rozhraní výsledků



Zdroj: vlastní

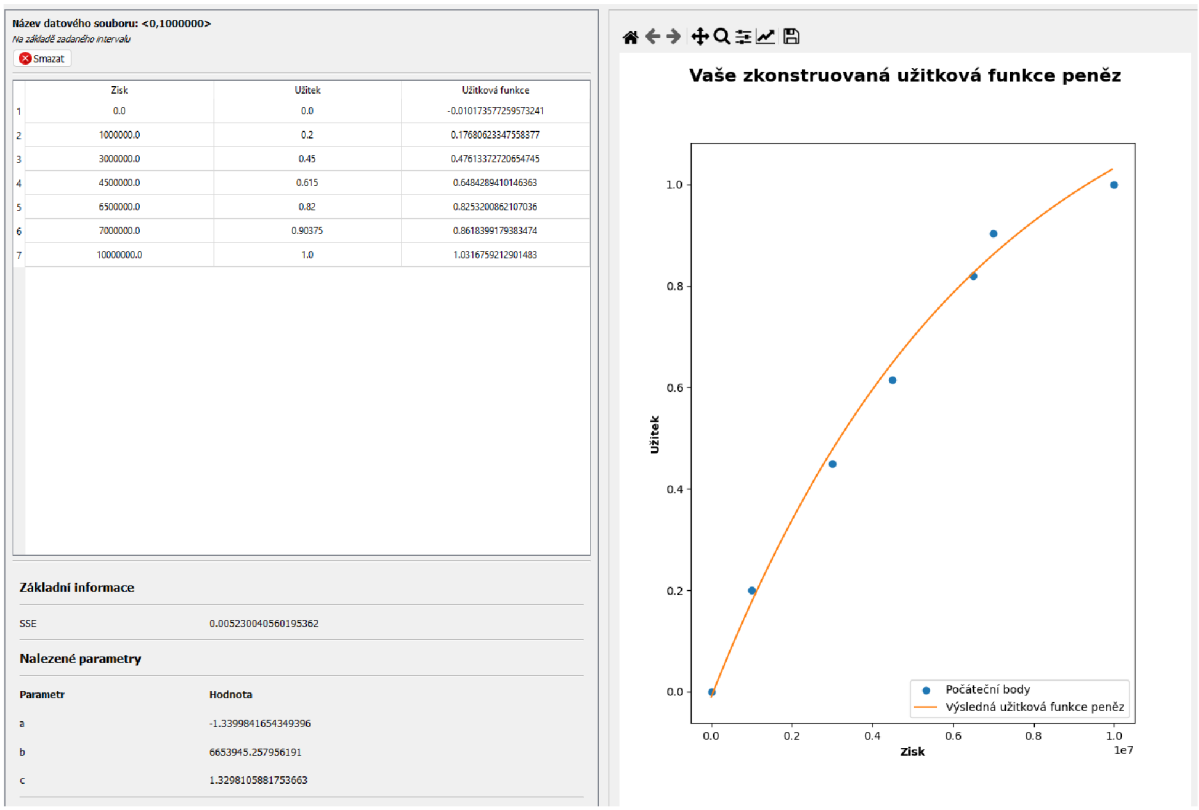
7. Modelové příklady

V rámci této kapitoly budou popsány tři příklady řešené prostřednictvím sestrojených užitkových funkcí peněz autorovo vytvořenou aplikací. Budou také uvedeny odlišné způsoby řešení modelových příkladů bez užitkové funkce, tedy pomocí pravidel z kapitoly teorie rozhodování za rizika.

Pro řešení modelových příkladů byly zkonstruovány dvě užitkové funkce dvou různých rozhodovatelů. Ke zkonstruování těchto užitkových funkcí byla využita autorem navržená aplikace. Vygenerované užitkové funkce dvou různých rozhodovatelů jsou zobrazeny na obrázcích číslo 58 a 59. Každý rozhodovatel odpověděl na pět náhodných otázek a jednu otázku ohledně maximální hodnoty M daného rozhodovatele.

První rozhodovatel zvolil maximální hodnotu M 10^7 a jeho užitková funkce je zobrazena na obrázku číslo 58. Byl použit pro konstrukci interval <0, 10^6>.

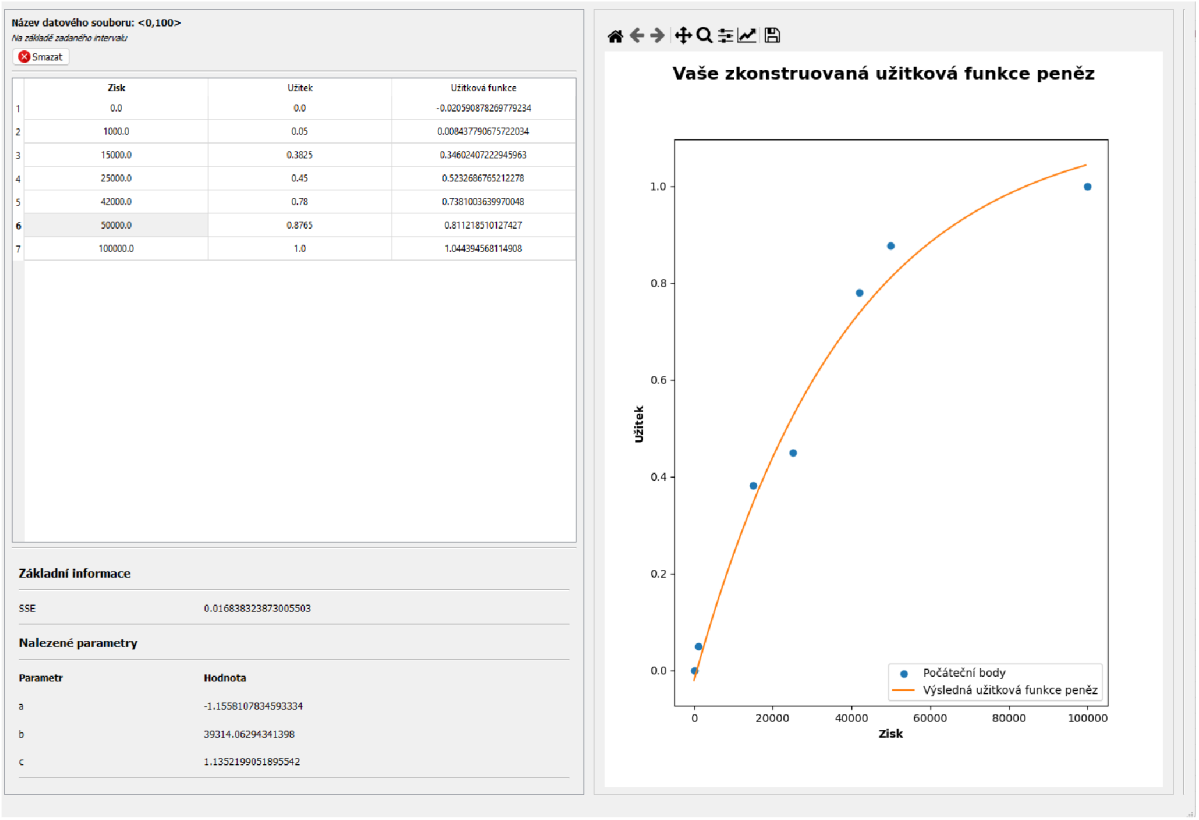
Obrázek 58: Užitková funkce prvního rozhodovatele



Zdroj: vlastní

Druhý rozhodovatel zvolil maximální hodnotu M=10^5 a jeho užitková funkce je zobrazena na obrázku číslo 59. Byl použit pro konstrukci interval <0, 10^3>.

Obrázek 59: Užitková funkce druhého rozhodovatele



Zdroj: vlastní

Na základě užitkových funkcí je možné konstatovat, že první rozhodovatel má menší averzi k riziku než rozhodovatel druhý. První rozhodovatel tedy může být člověk bohatší, s vyšším kapitálem, zatímco druhý rozhodovatel se nechce pouštět do většího rizika. Různé faktory mohou mít vliv na výslednou užitkovou funkci rozhodovatele – finanční problémy, rizika v podnikání, nějaké nové trendy ohrožující finanční situace rozhodovatele atd.

U všech příkladů je cílem spočítat jistotní ekvivalenty dvou různých rozhodovatelů na základě sestrojených užitkových funkcí peněz.

7.1 Petrohradský paradox

První modelový příklad byl vyřešen na základě vygenerovaných užitkových funkcí uvedených v kapitole 7 a poté byl řešen prostřednictví Bayesova pravidla pro výpočet očekávané střední hodnoty. Otázka je, kolik by vybraní rozhodovatelé byli ochotni zaplatit za takovou hru?

Petrohradský paradox je hra pro jednoho hráče, kde se hází mincí a to do takové doby, dokud nepadne hlava. Následně hra je ukončena. Výhra je matematicky definována jako 2^{k-1}, kde *k* je číslo odpovídající pořadí hodu, v němž padla hlava.

Tento příklad byl nejdříve řešen s nalezenými parametry užitkové funkce a, b, c prvního rozhodovatele. Průměrný užitek byl spočítán na základě rozkladu nekonečné sumy na *konečnou část + zbytek* a pomocí dolního a horního odhadu. Na základě tohoto průměrného užitku je počítán jistotní ekvivalent prostřednictví rovnice (29). Jistotní ekvivalent prvního rozhodovatele je 11,8878. Jinými slovy první rozhodovatel je ochoten zaplatit 11,8878 za tuto hru.

$$x = -b * \ln\left(\frac{u - c}{a}\right) \tag{29}$$

- x je jistotní ekvivalent,
- a, b, c parametry užitkové funkce,
- u průměrný užitek.

Výpočet jistotního ekvivalentu druhého rozhodovatele je proveden ekvivalentním způsobem. Jediným rozdílem je, že je nutné nastavit parametry a, b, c dle užitkové funkce druhého rozhodovatele. Jistotní ekvivalent tohoto rozhodovatele je 8,1872. Tedy nepatrně nižší než u prvního rozhodovatele.

Stejný příklad byl vypočítán prostřednictví Bayesova pravidla – tedy byla spočítaná očekávaná střední hodnota takovéto hry bez použití užitkové funkce. Výpočet očekávané střední hodnoty je zobrazen rovnicí (30).

$$E = \frac{1}{2} * 2 + \frac{1}{4} * 4 + \frac{1}{8} * 8 + \frac{1}{16} * 16 \ldots = 1 + 1 + 1 + 1 + \cdots = \infty \tag{30}$$

Očekávaní střední hodnota této hry je nekonečno. Střední hodnota výhry je nekonečno. Na základě toho lze říct, že i vstup by měl být roven nekonečnu. Petrohradský paradox je velmi extrémní, a proto jistotní ekvivalenty obou rozhodovatelů nejsou příliš rozdílné, ačkoli první rozhodovatel má menší averzi k riziku než druhý rozhodovatel. Dále lze konstatovat, že řešení s pomocí užitkové funkce poskytuje pro oba rozhodovatele výsledky, kolik za tuto hru jsou ochotni investovat, i přesto, že očekávaná střední hodnota této hry je rovna nekonečnu.

7.2 Powerball loterie

Jako druhý modelový příklad byla vybrána Powerball loterie. Powerball loterie má následující pravidla hry: (How to Play [Online]. Retrieved November 5, 2022, from https://www.flalottery.com/powerball-howToPlay)

- čísla jsou vybírána ze dvou separátních kontejnerů. První kontejner obsahuje 69 míčů (1 až 69). Druhý kontejner obsahuje 26 míčů,

- během výběru čísel je vybráno pět čísel z prvního kontejneru, jakmile je číslo vybráno, tak se nevrací zpět do kontejneru,
- následně je jedno Powerball číslo taháno z druhého kontejneru,
- pro výhru jackpotu je nutné, aby se všech šest čísel shodovalo s volbou čísel hráče.,
- některé částečně uhodnuté kombinace čísel získávají menší finanční odměnu.

Obrázek 60: Powerball – odměny a výherní kombinace

($1) DOUBLE PLAY® Odds and Prizes		
Match	**Odds**	**Prize**
5 of 5 + PB	1:292,201,338	$10,000,000**
5 of 5	1:11,688,053.52	$500,000
4 of 5 + PB	1:913,129.18	$50,000
4 of 5	1:36,525.17	$500
3 of 5 + PB	1:14,494.11	$500
3 of 5	1:579.76	$20
2 of 5 + PB	1:701.33	$20
1 of 5 + PB	1:91.98	$10
0 of 5 + PB	1:38.32	$7
Overall Odds	1:24.87	

Zdroj: How to Play [Online]. Retrieved November 5, 2022, from https://www.flalottery.com/powerball-howToPlay

S využitím zkonstruovaných užitkových funkcí rozhodovatelů byly spočteny jistotní ekvivalenty obou rozhodovatelů. Tedy částka, kolik jsou ochotni za takovou hru zaplatit.

Na obrázku číslo 61 je zobrazeno řešení tohoto příkladu pro prvního rozhodovatele v programu MS Excel. Průměrný užitek v tomto případě byl spočítán jako součin skalární sloupců Pravděpodobnost a Užitková funkce. Na základě tohoto výpočtu byl stanoven jistotní ekvivalent pomocí rovnice (29). Parametry užitkové funkce a, b, c byly nastaveny dle vygenerované užitkové funkce prvního rozhodovatele. Jistotní ekvivalent prvního rozhodovatele je 0,5045. Tato částka odpovídá částce, kolik je první rozhodovatel ochoten investovat do této hry.

Obrázek 61: Powerball loterie – první rozhodovatel

Shoda	Pravděpodobnost	Zisk	Užitková funkce	a	b	c
5 z 5 + Power Ball	3,4222978130237E-09	1000000	0,17686806638844700	-1,3399	6653945	1,3298
5 z 5	8,5557449132033E-08	500000	0,08689473496725460			
4 z 5 + Power Ball	1,0951355175446E-06	50000	-0,00006927159112502			
4 z 5	2,7378507871321E-05	500	-0,00999931915525343			
3 z 5 + Power Ball	0,00006899406651028	500	-0,00999931915525343			
3 z 5	0,00173611111111111	20	-0,01009597262095090			
2 z 5 + Power Ball	0,00142653352353780	20	-0,01009597262095090			
1 z 5 + Power Ball	0,01098901098901100	10	-0,01009798630896230			
0 z 5 + Power Ball	0,02631578947368420	7	-0,01009859041595580			
žádná výhra	9,5943499821301E-01	0	-0,01010000000000000			
		Průměrný užitek	-0,01009989841367970			
		Jistotní ekvivalent	0,504477826			
	Bayes	0,506497614				

Zdroj: vlastní

Příklad pro druhého rozhodovatele byl řešen ekvivalentním způsobem. Jediným rozdílem bylo jiné nastavení parametrů užitkové funkce dle vygenerované užitkové funkce druhého rozhodovatele. Jistotní ekvivalent druhého rozhodovatele pro Powerball loterii je 0,4397, což opět odpovídá hodnotě, kterou je ochotný tento rozhodovatel do hraní této hry investovat.

Obrázek 62: Powerball loterie – druhý rozhodovatel

Shoda	Pravděpodobnost	Zisk	Užitková funkce	a	b	c
5 z 5 + Power Ball	3,4222978130237E-09	1000000	1,13519999998962000	-1,1558	39314,06	1,1352
5 z 5	8,5557449132033E-08	500000	1,13519653674823000			
4 z 5 + Power Ball	1,0951355175446E-06	50000	0,81120162825436500			
4 z 5	2,7378507871321E-05	500	-0,00599350568725154			
3 z 5 + Power Ball	0,00006899406651028	500	-0,00599350568725154			
3 z 5	0,00173611111111111	20	-0,02001216655154940			
2 z 5 + Power Ball	0,00142653352353780	20	-0,02001216655154940			
1 z 5 + Power Ball	0,01098901098901100	10	-0,02030604589516560			
0 z 5 + Power Ball	0,02631578947368420	7	-0,02039422427582280			
žádná výhra	9,5943499821301E-01	0	-0,02060000000000000			
		Průměrný užitek	-0,02058707403271210			
		Jistotní ekvivalent	0,439673934			
	Bayes	0,506497614				

Zdroj: vlastní

Dále byla spočítaná i očekávaná střední hodnota této hry pomocí Bayesova pravidla. Výsledná očekávaná střední hodnota je 0,5065. V obou případech vychází opět jistotní ekvivalenty velmi obdobně u obou rozhodovatelů, neboť Powerball loterie je hra, kde s vysokou pravděpodobností není žádná výhra a také platí, že se zvyšující se možnou výhrou velmi rychle klesá pravděpodobnost výhry.

7.3 Loterie s oky kostky

Jako třetí modelový příklad byla zvolena hra, kde je házeno kostkou a je vypláceno 10^k, kde k je počet ok kostky. Zvolené pravděpodobnosti jednotlivých výher je možné vidět na obrázcích 63 a 64.

Na obrázku číslo 63 je zobrazen výpočet jistotního ekvivalentu prvního rozhodovatele. Jistotní ekvivalent prvního rozhodovatele je roven 3033,3304.

Obrázek 63: Loterie s oky kostky – první rozhodovatel

pravděpodobnost	počet ok	zisk	užitková fce		a	b	c
0,5800	1,0	10,0	-0,010097986308962300000000000000		-1,3399	6653945	1,3298
0,2100	2,0	100,0	-0,010079863225806100000000000000				
0,1530	3,0	1000,0	-0,009898645875722420000000000000				
0,0500	4,0	10000,0	-0,008087819848215720000000000000				
0,0050	5,0	100000,0	0,009886365010590480000000000000				
0,0020	6,0	1000000,0	0,176868066388447000000000000000				
1,0000							
Bayes	3179,8	průměrý užitek	-0,009489319190183890000000000000				
		jistotní ekvivalent	3033,333045853620000000000000000				

Zdroj: vlastní

Na obrázku číslo 64 je zobrazen výpočet jistotního ekvivalentu druhého rozhodovatele. Jistotní ekvivalent druhého rozhodovatele je roven 889,0469.

Obrázek 64: Loterie s oky kostky – druhý rozhodovatel

pravděpodobnost	počet ok	zisk	užitková fce		a	b	c
0,5800	1,0	10,0	-0,020306045895165600000000000000		-1,1558	39314,06	1,1352
0,2100	2,0	100,0	-0,017663820925803500000000000000				
0,1530	3,0	1000,0	0,008428398146081320000000000000				
0,0500	4,0	10000,0	0,238979821995151000000000000000				
0,0050	5,0	100000,0	1,044375510560800000000000000000				
0,0020	6,0	1000000,0	1,135199999989620000000000000000				
1,0							
Bayes	3179,8	průměrý užitek	0,005243904555276460000000000000				
		jistotní ekvivalent	889,0469022192580000000000000000				

Zdroj: vlastní

Očekávaná střední hodnota, která byla spočítána s použitím Bayesova pravidla, je rovna 3179,8. V rámci této loterie je mnohem více evidentní rozdíl mezi bohatším a chudším rozhodovatel. První rozhodovatel je ochoten investovat do této hry znatelně vyšší částku než rozhodovatel druhý.

8. Závěr

Cílem této diplomové práce bylo popsat teoretické poznatky z oblasti teorie rozhodování, teorie užitku a postupu konstrukce užitkové funkce za rizika. Na základě těchto poznatků bylo cílem navrhnout a naprogramovat aplikaci pro konstrukci užitkové funkce peněz pro rozhodovatele s averzí k riziku. Dalším cílem bylo využití této autorovy navržené aplikace k vyřešení různých modelových příkladů.

První část práce shrnuje poznatky ze studia odborné literatury, která je zaměřená na teorii rozhodování za jistoty, nejistoty, rizika a byly shrnuty i rozhodovací stromy. Dále byly shrnuty teoretické poznatky z teorie užitku, užitkové funkce za jistoty a užitkové funkce peněz. Další tři kapitoly se zabývají návrhem, popisem a implementací aplikace. Další kapitola je věnována modelovým příkladům, které jsou řešeny s využitím užitkových funkcí, které byly vygenerovány autorovo navrženou aplikací. Implementace aplikace byla úspěšná, neboť výsledná aplikace umožnuje rozhodovateli zkonstruovat si vlastní užitkovou funkci v zadaném intervalu, nebo na základě vložené datové tabulky. Byl podrobně rozepsán algoritmus generování náhodných otázek, který umožnuje generovat tři druhy otázek. Je umožněno rozhodovateli, aby si navolil, jaké množství otázek je ochoten odpovídat. Čím více zodpovězených otázek, tím přesnější výsledná užitková funkce je. Otázky jsou generovány náhodně, nicméně je vyžadováno rovnoměrné rozložení bodů v pěti intervalech. Jsou využívány dva typy náhodných otázek – rozhodovatel buďto odpovídá ve formě pravděpodobnosti, anebo ve formě zisku. Dále je řešen způsob hledání ideálních počátečních parametrů užitkové funkce do regrese. Za tímto účelem je zavedena otázka na maximální hodnotu M uživatele, pomocí které je zjišťováno, kdy rozhodovatel není schopen rozlišit, zda dostal tuto hodnotu či dvojnásobnou. Na základě této hodnoty je možné vymezit horní a dolní hranici pro parametr b užitkové funkce peněz při hledání počáteční ideální hodnoty. Pro hledání ideálních startovacích hodnot parametrů užitkové funkce je navržen algoritmus, který postupně prostupuje možný prostor hodnot. Je počítáno SSE, které je porovnáváno, dokud není nalezena shoda, nebo dokud není překročen maximální počet kroků. Jakmile jsou nalezené optimální startovací hodnoty parametrů, je proveden výpočet a rozhodovateli je zobrazena jeho užitková funkce peněz společně s nalezenými hodnotami parametrů užitkové funkce peněz a s příslušnými daty. Je také implementována možnost uložení výsledné užitkové funkce peněz ve formě obrázku. Rozhodovatel může následně na základě této funkce řešit nějaké konkrétní problémy – například může použít užitkovou funkci pro výpočet jistotního ekvivalentu, případně může využít hodnoty užitků v kombinaci s některými pravidly z teorie rozhodování.

Autorova navržená aplikace je následně užita pro konstrukci dvou užitkových funkcí dvou různých rozhodovatelů. Na základě těchto užitkových funkcí je počítán jistotní ekvivalent pro oba rozhodovatele pro všechny tři modelové příklady – pro Petrohradský paradox, Powerball loterii a Loterii s oky kostky. Pro porovnání jsou příklady řešeny také prostřednictví Bayesova pravidla z teorie rozhodování za rizika. V rámci Petrohradského paradoxu je ukázáno, že s využitím užitkové funkce lze vypočítat, kolik jsou zvolení rozhodovatelé ochotni investovat do této hry, přestože očekávaná střední hodnota tohoto příkladu je rovna nekonečnu. V rámci Powerball loterie jsou popsána pravidla a zisky s korespondujícími pravděpodobnostmi. Dále je v rámci tohoto modelového příkladu opět počítán jistotní ekvivalent pro oba rozhodovatele. Pro porovnání je opět vypočítaná očekávaná střední hodnota hry prostřednictví Bayesova pravidla. Jako třetí modelový příklad je uvedena Loterie s oky kostky, u které je více ukázán rozdíl mezi bohatším a chudším rozhodovatelem.

Tabulka 3: Porovnání výsledků modelových příkladů

Modelový příklad	$\hat{x}$ – 1. rozhodovatel	$\hat{x}$ - 2. rozhodovatel	Bayesovo pravidlo
Petrohradský paradox	11,8878	8,1872	Nekonečno
Powerball loterie	0,5045	0,4397	0,5065
Loterie s oky kostky	3033,3304	889,0469	3179,8

Zdroj: vlastní

V tabulce číslo 3 jsou uvedené zkompletované výsledky pro všechny řešené modelové příklady. V prvním sloupci tabulky se nacházejí řešené modelové příklady. Druhý a třetí sloupec obsahuje zjištěné jistotní ekvivalenty rozhodovatelů pro jednotlivé modelové příklady. V posledním sloupci jsou zobrazeny výsledky pro jednotlivé modelové příklady spočítané prostřednictvím Bayesova pravidla. Na základě získaných výsledků je možné konstatovat, že jistotní ekvivalenty obou rozhodovatelů jsou pro každý modelový příklad menší než očekávaná střední hodnota související hry. Dále je evidentní, že jistotní ekvivalenty obou rozhodovatelů nejsou příliš rozdílné pro první dva modelové příklady (Petrohradský paradox a Powerball loterie) a to i přesto, že první rozhodovatel je bohatší a s menší averzí k riziku než rozhodovatel druhý. Petrohradský paradox je velmi extrémní, a proto jistotní ekvivalenty obou rozhodovatelů nejsou příliš rozdílné. Nicméně je v tomto případě možné stanovit jistotní ekvivalenty obou rozhodovatelů i přesto, že očekává střední hodnota této hry je rovna nekonečnu. U Powerball loterie jsou nepříliš rozdílné výsledky jistotních ekvivalentů pro oba rozhodovatele dány tím, že se jedná o hru, kde s vysokou pravděpodobností není žádná

výhra a také platí, že se zvyšující se možnou výhrou velmi rychle klesá pravděpodobnost výhry. Třetí modelový příklad (Loterie s oky kostky) nejlépe demonstruje rozdíl mezi bohatým a chudším rozhodovatelem, neboť jistotní ekvivalenty obou rozhodovatelů jsou velmi rozdílné.

Závěrem lze říct, že aplikace byla úspěšně implementována a použita na modelové příklady. Autor se domnívá, že pomocí této aplikace lze řešit konkrétní rozhodovací problémy. Nicméně kvalita zkonstruované funkce závisí na množství získaných bodů při procesu generování otázek. Lze také konstatovat, že otázky jsou generovány rovnoměrně do pěti intervalů, nicméně pokud rozhodovatelem vytvořený interval, na kterém se generují otázky, je velmi velký, tak je sice dosaženo rovnoměrného rozložení do pěti dílčích intervalů, ale jednotlivé dílčí intervaly mohou být celkem velké. Další nevýhodou může být, že obecně generované otázky mohou být celkem složité a pro laika těžko zodpověditelné. Potencionální vylepšení aplikace by mohlo být dosaženo implementací konstrukce užitkové funkce prostřednictvím alternativní užitkové funkce, u které je vyžadována jen odpověď na jednu otázku a je tedy mnohem rychlejší, pokud rozhodovatel nechce podstupovat poměrně dlouhý proces, ve kterém je nutné odpovídat na větší množství otázek. Rozhodovatel by si pak mohl v rámci aplikace sám určit, jaký způsob konstrukce užitkové funkce chce použít. Další možné vylepšení by byla konstrukce užitkové funkce nejen peněz, ale i pro větší spektrum možností. Aplikace bude využita například k výuce předmětů, které se zabývají užitkovou funkcí na Ekonomické fakultě Jihočeské univerzity.

9. Summary

This diploma thesis deals with the process of creating an application with appropriate alghoritms and mechanisms for constructing utility function for money for a given decision maker. Related theoretical topics from the decision theory, utility theory and especially the process of construction of the utility function for money are explained. Application randomly generates questions in given intervals for the decision maker. After the process of the generating question is finished, alghorithm for finding optimal starting values is applied. This alghoritm gradually compares SSE in the calculated interval. Nonlinear regression is used to fit the data gathered in previous steps. The graph of the found utility function is drawn and found parameters of the utility function are displayed together with the corresponding data. The paper goes on to evaluate the functionality of the application by solving model examples (St.Petersburg paradox, Powerball lottery and Cube lottery) for two different decision makers. These decision makers used designed application for constructing their utility functions for money. These two utility functions are utilized for calculating certainty equivalents for both decision makers and the results are compared. Application is implemented in Python and PyQt5.

Keywords

Application software for construction of the utility function for money, Decision theory, Utility theory, Utility function, Construction of the utility function for money, Python, PyQt5

10. Seznam použité literatury

- Hillier, F. S., & Lieberman, G. J. (2010). Introduction to Operations Research (9th ed.). McGraw-Hill.

- Taha, Hamdy A. (1997). Operations Research, an Introduction. Prentice Hall.

- Winston, Wayne L. (1994). Operations Research, Applications and Algorithms Duxbury Press.

- Anderson, D. R., Sweeney, D. J., Williams, T. A., Camm, J. D., Cochran, J. J., Fry, M. J., & Ohlmann, J. W. (2013). Quantitative Methods For Business (12th ed.). Cengage Learning.

- Pažek, Karmen & Rozman, Črtomir. (2009). Decision making under conditions of uncertainty in agriculture: A case study of oil crops. Poljoprivreda (Osijek). 15.

- Murthy, P. R. (2007). Operations Research (2nd ed.). New Age International (P) Ltd., Publishers.

- Richter Jiří, Pravidla rozhodování za rizika a nejistoty [online prezentace]. 2019 [cit.2021-09-28]. Dostupné z:https://moodle.unob.cz/pluginfile.php/85160/mod_resource/content/3/MR%202019%20P8%20Rozhodov%C3%A1n%C3%AD%20za%20rizika%20a%20nejistoty.pdf

- Friebelová, J., & Klicnarová, J. (2007). Rozhodovací modely pro ekonomy. Jihočeská univerzita v Českých Budějovicích, Ekonomická fakulta.

- Fotr, J. (2006). Manažerské rozhodování: Postupy, metody a nástroje. Ekopress.

- Steele, K., & Stefánsson, H. O. (2020). *"Decision Theory": The Stanford Encyclopedia of Philosophy* [Online] (Winter ed., Vol. 2020). Stanford University, Stanford: {Metaphysics Research Lab, Stanford University}. Retrieved from https://plato.stanford.edu/archives/win2020/entries/decision-theory/

- Šetek, D. (2018). Mikroekonomie II. Vysoká škola ekonomie a managementu.

- Klicnarová, J. (2021). Rozhodovací modely. Jihočeská univerzita

- Balakrishnan, N. R., Render, B., Stair, R., & Munson, C. (2017). Managerial decision modeling. In Managerial Decision Modeling. De Gruyter.

- How to Play [Online]. Retrieved November 5, 2022, from https://www.flalottery.com/powerball-howToPlay

11. Seznam obrázků a tabulek

11.1 Seznam obrázků

11.2 Seznam tabulek

12. Přílohy

12.1 Obsah přiložených souborů

- **utilityconstructor.zip** obsahuje implementovanou aplikaci. Pro spuštění aplikace je nutné vyextrahovat na libovolné místo na disku a následně nalézt soubor main.exe, který aplikaci spouští. Aplikaci lze v tomto formátu spustit pouze na OS Windows.
- **modelovepriklady.xlsx** obsahuje řešené modelové příklady pro oba rozhodovatele v MS Excel.