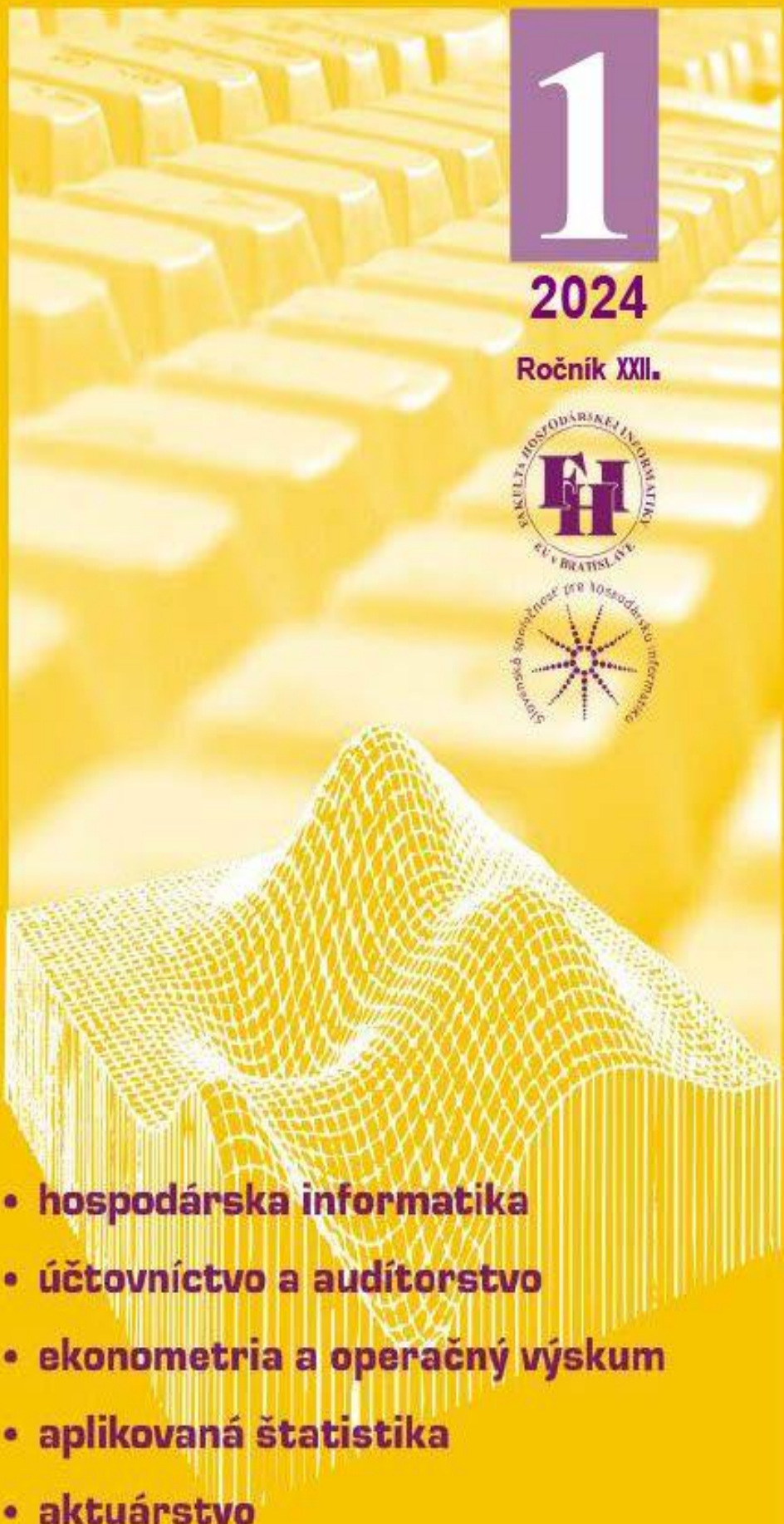
EKONOMIKA INFORMATIKA

vedecký časopis FHI EU v Bratislave a SSHI



1
2024
Ročník XXII.

- hospodárska informatika
- účtovníctvo a audítorstvo
- ekonometria a operačný výskum
- aplikovaná štatistika
- aktuárstvo

Vydavateľ
Fakulta hospodárskej informatiky Ekonomickej univerzity v Bratislave
a Slovenská spoločnosť pre hospodársku informatiku

IČO vydavateľa 00 399 957

Redaktorka: Eva Čerteková

Adresa redakcie: Fakulta hospodárskej informatiky, Ekonomická univerzita v Bratislave
Dolnozemská cesta 1, 852 35 Bratislava
tel.: 02/6729 5723, e-mail: eva.certekova@euba.sk

Dátum vydania periodickej tlače: júl 2024

ISSN 1339-987X (online)
ISSN 1336-3514 (online vydanie)

OBSAH 1/2024

VEDECKÉ STATE A DISKUSIE

Regresia pomocou metódy podporných vektorov: Nástroj pre presné a robustné predikcie

Support Vector Regression: A tool for accurate and robust predictions

Andrej Bednařík[1]

Abstrakt
Tento príspevok je zameraný na Support Vector Regression (SVR), pokročilú metodológiu strojového učenia pre riešenie regresných problémov v rôznych aplikáciách. SVR vychádza z algoritmov Support Vector Machine, využíva podporné vektory na modelovanie prediktívnych funkcií, ktoré minimalizujú chyby predikcie v rámci preddefinovaného prahu. Tento robustný mechanizmus umožňuje vysokú presnosť aj pri komplexných a šumivých dátových súboroch. Príspevok rieši princípy, metódy a aplikácie SVR, pričom zdôrazňuje prispôsobivosť na nelineárne problémy prostredníctvom kernelových metód a využitie.

Kľúčové slová
Support Vector Regression (SVR), Kernel, Python, Predikcia

Abstract
This document explores Support Vector Regression (SVR), an advanced machine learning methodology for solving regression problems across various applications. Originating from Support Vector Machine algorithms, SVR utilizes support vectors to model predictive functions that minimize prediction errors within a predefined threshold. This robust mechanism allows for high accuracy even with complex and noisy data sets. The paper discusses the principles, methodologies, and applications of SVR, emphasizing its adaptability to nonlinear problems through kernel methods and its applications.

Key words
Support Vector Regression (SVR), Kernel, Python, Prediction

JEL classification
C61, C89

1 Úvod

Support Vector Regression (SVR) predstavuje sofistikovanú metodiku v rámci strojového učenia, navrhnutú na efektívne riešenie regresných problémov v širokom spektre aplikácií. Tento prístup vychádza z algoritmu Support Vector Machine (SVM), ktorý bol pôvodne vyvinutý pre účely klasifikácie. Princíp SVM spočíva v identifikácii a optimalizácii rozhodovacích hraníc medzi jednotlivými klasifikačnými triedami, pričom SVR tento koncept rozširuje do domény regresnej analýzy (Vapnik, 1995). V kontexte SVR sa využíva koncept podporných vektorov na modelovanie prediktívnej funkcie, ktorá sa snaží minimalizovať chyby predikcie tak, že odchýlky medzi predpovedanými a skutočnými hodnotami sú držané v rámci vopred definovaného prahu známeho ako ε (epsilon). Tento mechanizmus umožňuje SVR zachovať vysokú presnosť predikcie aj v prípade komplexných a „šumivých“ dátových súborov, čo je neoceniteľné v mnohých praktických aplikáciách. Hyperrovina v modeli SVR je

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

reprezentovaná pomocou vybraných trénovacích bodov, známych ako podporné vektory. Tieto body sú kritické pre model, pretože určujú hranice predikčného intervalu a sú priamo zapojené do výpočtu konečného rozhodovacieho modelu (Smola & Schölkopf, 2004). SVR má svoje korene v 90. rokoch 20. storočia, keď boli položené základy teórie strojového učenia v práci Vapnika a jeho kolegov. Tieto algoritmy sú postavené na pevnom základe teórie štatistického učenia a konceptu štrukturálneho rizika, ktoré cieľavedome minimalizuje pravdepodobnosť chyby na nevidených dátach a zároveň redukuje riziko preučenia modelu (Vapnik, 1995). Dnes sa SVR aplikuje v širokom spektre odvetví, od predpovedí na finančných trhoch až po optimalizáciu energetických systémov a vývoj pokročilých zdravotníckych diagnostických nástrojov. Výhody SVR, ako sú robustnosť, presnosť a schopnosť efektívne spracovávať veľké objemy dát, sú zásadné pre tieto aplikácie.

2 Princípy a metódy SVR

Základný princíp SVR je že pracuje na princípe štruktúrneho minimalizovania rizika, ktorý je základom teórie strojového učenia Vapnika (Vapnik, 1995). Tento prístup sa snaží nájsť rovnováhu medzi zložitosťou modelu a mierou, do akej sa model prispôsobuje trénovacím dátam, aby sa minimalizovala chyba na nevidených dátach.

Loss funkcia a ε-insensitivity: SVR zavádza koncept ε-insenzitívnej loss funkcie, ktorá ignoruje chyby v predikcii, ktoré sú menšie ako ε. Tento prístup umožňuje určitú mieru odchýlok bez toho, aby boli penalizované, čo pomáha predchádzať preučeniu modelu (Smola & Schölkopf, 2004).

Optimalizačný problém: Cieľom SVR je nájsť funkciu, ktorá najlepšie oddelí všetky dáta plus a mínus ε od skutočnej cieľovej hodnoty y. To sa dosahuje riešením optimalizačného problému, kde sa minimalizuje norma váhového vektora w a zároveň sa trestajú odchýlky, ktoré sú väčšie ako ε (Bishop, 2006).

Kernelová metóda: Podobne ako SVM, aj SVR môže využívať kernelovú metódu. Kernelová metóda využíva matematické funkcie, nazývané kernelové funkcie, na transformáciu pôvodného vstupného priestoru do nového, typicky vyššieho dimenzionálneho priestoru, čo umožňuje efektívne riešenie nelineárnych regresných problémov. kde sú vzťahy medzi dátovými bodmi jednoduchšie alebo dokonca lineárne separovateľné. Tento prístup umožňuje SVM a SVR efektívne pracovať s komplexnými alebo nelineárnymi vzťahmi bez explicitného zvyšovania dimenzií vstupných dát, čo by bolo výpočtovo veľmi náročné. Bežné kernely zahŕňajú lineárne, polynomiálne, radiálne bázové funkcie (RBF) a sigmoidálne kernely (Hofmann, Schölkopf, & Smola, 2008).

2.1 ε-insensitivity loss function a optimalizačný problém

V kontexte Support Vector Regression (SVR) je kľúčovou súčasťou modelu tzv. ε-insenzitívna loss funkcia, ktorá hrá dôležitú úlohu v znižovaní preučenia a zvyšovaní schopnosti generalizácie modelu (Smola & Schölkopfom, 2004).
Vapnikova ε-insenzitívna loss funkcia, označovaná ako L_ε, je definovaná nasledovne:

$$L_\varepsilon\big(y, f(x)\big) = \max(0, |y - f(x)| - \varepsilon), \tag{1}$$

kde y je skutočná hodnota cieľovej premennej, $f(x)$ je predikovaná hodnota modelom, ε(epsilon) je nenulová hranica, ktorá definuje prah, do ktorého sú chyby ignorované.

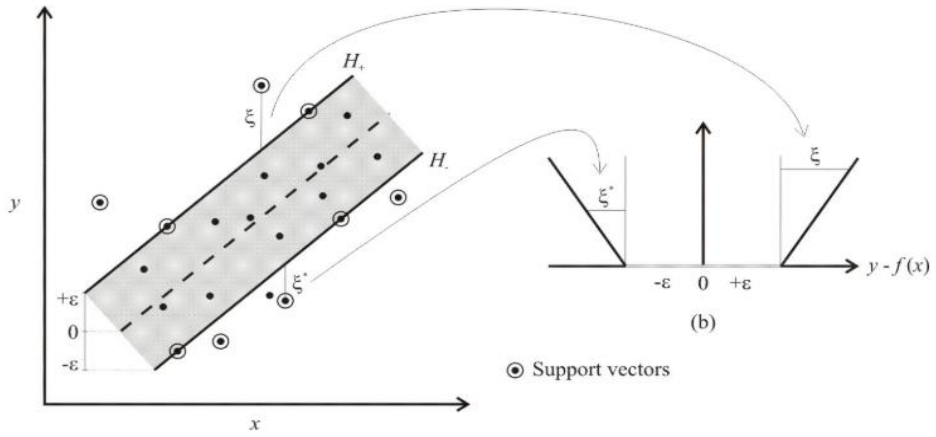
Funkcia L_ε vracia hodnotu 0 pre všetky chyby predikcie $|y - f(x)|$, ktoré sú menšie alebo rovné ε. Toto znamená, že chyby, ktoré sú v rámci prahu ε, nepripisujú žiadnu stratu, a teda model nie je penalizovaný za tieto malé odchýlky. Ak je odchýlka väčšia ako ε, potom je strata počítaná ako rozdiel medzi absolútnou hodnotou chyby a ε. Hlavnou výhodou tohto

prístupu je jeho schopnosť minimalizovať vplyv šumu a náhodných fluktuácií v trénovacích dátach, čo zvyšuje robustnosť a spoľahlivosť modelu. Funkcia umožňuje modelu SVR presne predpovedať dôležité vzory v dátach bez nadmernej citlivosti na malé odchýlky. Teda Vapnikova lineárna ε-insenzitívna stratová funkcia definuje "trúbka" s polomerom ε okolo cieľových hodnôt y. Potom platí:

$$|y - f(x)| - \varepsilon = \xi, \text{ pre dátové body "nad" trubkou.} \quad (2)$$

$$|y - f(x)| - \varepsilon = \xi^*, \text{ pre dátové body "pod" trubkou.} \quad (3)$$

Obr. 1: Znázornenie slack premenných



Zdroj: (Lins et al., 2010)

2.2 Kernel

Kernel, známy tiež ako jadro, je fundamentálna súčasť mnohých metód strojového učenia, najmä v kontexte Support Vector Machines (SVM) a príbuzných techník, ako je Support Vector Regression (SVR). Kernelové funkcie transformujú pôvodné vstupné dáta do vyššieho dimenzionálneho priestoru, umožňujúc efektívne riešenie nelineárnych problémov bez nutnosti explicitne zvyšovať dimenzionalitu dát. Tento proces je známy ako "kernelový trik" a je zásadný pre manipuláciu s komplexnými vzťahmi medzi dátami v nelineárnych priestoroch (Shawe-Taylor & Cristianini, 2004).

Typy kernelov a ich aplikácie:

Lineárny kernel s kernelovou fukniou: $K(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$.

- Lineárny kernel je najjednoduchší a počíta skalárny súčin dvoch vektorov. Je účinný, keď sú dáta lineárne separovateľné a nevyžadujú transformáciu do vyššieho dimenzionálneho priestoru.

Polynomiálny kernel s kernelovou funkciou: $\mathrm{K}(\mathbf{x}, \mathbf{y}) = (\gamma \cdot \langle \mathbf{x}, \mathbf{y} \rangle + r)^d$.

- Polynomiálny kernel umožňuje SVM modelovať rozhodovacie hranice v tvare polynómov určitého stupňa d. Jeho flexibilita pri modelovaní nelineárnych vzťahov je významnou výhodou v aplikáciách, kde lineárne modely zlyhávajú (Chang & Lin, 2011).

RBF kernel s funkciou: $K(\mathbf{x}, \mathbf{y}) = exp(-\gamma \parallel \mathbf{x} - \mathbf{y} \parallel^2)$.

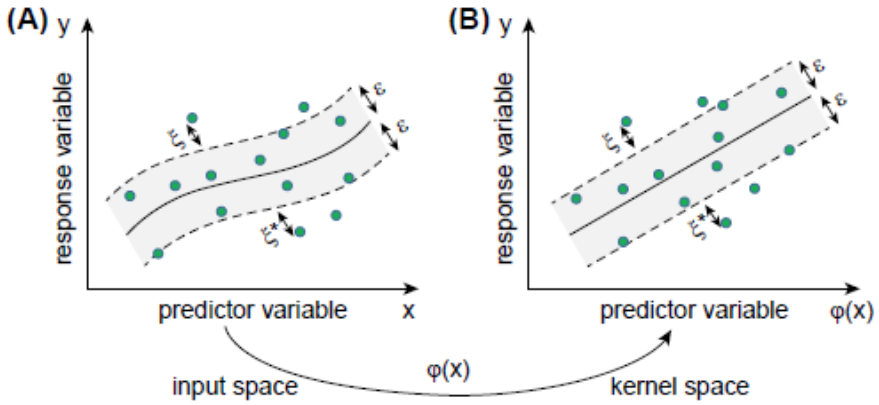
- RBF kernel, často nazývaný aj Gaussovský kernel, je veľmi obľúbený vďaka svojej schopnosti efektívne mapovať prvky do nekonečne dimenzionálneho priestoru, čo je ideálne pre veľmi zložité vzťahy medzi dátovými bodmi.

Sigmoidálny kernel s funkciou: $K(\mathbf{x}, \mathbf{y}) = tanh(\gamma \cdot \langle \mathbf{x}, \mathbf{y} \rangle + r)$.

- Sigmoidálny kernel, inšpirovaný neurónovými aktiváciami, môže poskytnúť výstupy, ktoré pripomínajú aktivačné funkcie používané v neurónových sieťach, čo je užitočné pre určité typy klasifikačných problémov.

Duálna formulácia tohto optimalizačného problému zjednodušuje výpočty a umožňuje využitie kernelových funkcií na zvládnutie nelineárnych vzťahov v dátach. Duálny problém využíva Lagrangeove multiplikátory na preformulovanie optimalizačného problému, ktorý sa následne rieši pomocou kvadratického programovania. Voľba správneho kernelu závisí od povahy dát a špecifických požiadaviek problému. Správne nastavenie kernelu môže dramaticky zvýšiť výkonnosť modelu, zatiaľ čo nesprávna voľba môže viesť k nedostatočnému učeniu alebo preučeniu (Hofmann, Schölkopf, & Smola, 2008).

Obr. 2: Transformácia priestoru



Zdroj: (Zhang & O'Donnell, 2020)

Na obrázku 2 je grafické znázornenie nelineárneho ε-SVR. Funkcia mapovania φ sa používa na transformáciu dát z vstupného priestoru (A), kde nie je možné lineárne oddeliť dáta, do vyššie-dimenzionálneho kernelového priestoru (B), kde môžu byť dáta oddelené lineárnou hyperrovinou.

3 Lineárny ε – SVR model

Cieľom ε-SVR (ε-Support Vector Regression) je odhadnúť funkciu s obmedzením, že odhad každého vstupného dátového bodu má najviac ε odchýlku od svojej skutočnej hodnoty odpovede, vytvorením ε-insenzitívnej trubice symetricky okolo odhadnutej funkcie. Matematická formulácia lineárneho ε-SVR môže byť vyjadrená nasledovne. Predpokladajme, že máme súbor trénovacích dát $\{(x_{11}..... x_{1k}, y_1), ..., (x_{n1} ..... x_{nk}, y_n)\}$, kde x_{i1},...., x_{ik}, sú hodnoty vstupných dát a y_i je cieľový výstup, $i=1,...,n$. Prípad lineárnej regresnej funkcie f má tvar:

$$y = f(x) = \langle \boldsymbol{\omega}, \mathbf{x} \rangle + b, \tag{4}$$

kde $\langle \boldsymbol{\omega}, \mathbf{x} \rangle$ označuje skalárny súčin vektora vstupných dát (vysvetľujúce premenné) $\mathbf{x}$ a vektor váh $\boldsymbol{\omega}$ a b je konštanta ktorá nie je pevne daná, ale je skôr parameter, ktorý sa optimalizuje počas trénovania modelu. Optimalizácia b sa vykonáva spolu s optimalizáciou vektora váh ω s cieľom minimalizovať chybu predikcie. V ε-SVR sa aproximácia funkcie f vykonáva nájdením ε-insenzitívnej trubice, ktorá je čo najplochejšia, čo je formálne označované ako plochosť.

Viacúčelový optimalizačný problém SVR možno zapísať nasledovne

$$min\left\{\frac{1}{2}||\boldsymbol{\omega}||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)\right\} \tag{5}$$

Pričom platia nasledovné obmedzenia:

$$\begin{gathered} y_i - \langle \boldsymbol{\omega}, \mathbf{x_i} \rangle - b \leq \varepsilon + \xi_i \\ \langle \boldsymbol{\omega}, \mathbf{x_i} \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0, \end{gathered} \tag{6}$$

kde $\mathbf{x_i}$ reprezentuje vstupné prvky, kde $\boldsymbol{\omega}$ obsahuje váhy priradené ku každému vstupnému prvku (feature) vo vstupnom vektore $\mathbf{x_i}$,tieto váhy určujú, aký vplyv bude mať každý vstupný prvok na výstupnú hodnotu modelu a C je regulačný parameter, ktorý kontroluje rovnováhu medzi hladkosťou modelu a veľkosťou odchýlok. $||\boldsymbol{\omega}||^2$označuje kvadrát normy váhového vektora $\boldsymbol{\omega}$ (Zhang & O'Donnell, 2020).

Optimalizácia v SVR zahŕňa minimalizáciu kombinácie normy váhového vektora a sumy ε-insenzitívnych strát cez všetky trénovacie dáta. Základný matematický model SVR sa snaží nájsť funkciu, ktorá minimalizuje chybu predikcie v rámci prahu ε a zároveň udržiava hladkosť modelu. Termín "hladkosť modelu" je kľúčový pre schopnosť modelu generalizovať, čo znižuje riziko preučenia. Hladký model efektívne predpovedá výsledky na nevidených dátach vďaka svojej jednoduchosti a odolnosti voči šumu v trénovacích dátach.
Definícia hladkosti modelu zahŕňa:

- Redukcia zložitosti: Hladký model má jednoduchšiu štruktúru, čo znamená menej parametrov alebo nižší stupeň polynómu. Toto pomáha zabezpečiť, že model nie je nadmerne prispôsobený na špecifické vzory alebo šum v trénovacích dátach.
- Generalizácia: Hladký model je menej citlivý na malé variácie v dátach, čo znižuje pravdepodobnosť, že model zachytí náhodné vzory, ktoré nie sú reprezentatívne pre všeobecné dáta.

Hladkosť modelu sa dosahuje pomocou

- Normy váhového vektora($||\boldsymbol{\omega}||^2$): Minimalizácia normy váhového vektora $\boldsymbol{\omega}$, ktorý predstavuje koeficienty regresného modelu, pomáha udržiavať model jednoduchý a hladký. Nižšie hodnoty normy znamenajú menej zložitý model, ktorý je všeobecne lepší pre generalizáciu.
- Penalizáciou odchýlok(ξ_i, ξ_i^*): V SVR sú zavedené takzvané slack premenné ξ_i, ξ_i^*, ktoré umožňujú určitú flexibilitu v prekročení prahu ε. Penalizácia týchto premenných zabezpečuje, že model neignoruje veľké odchýlky, čo pomáha vyvážiť medzi prísnosťou a prílišnou flexibilitou.

Táto rovnováha medzi udržaním modelu jednoduchým a zároveň dostatočne flexibilným na presné modelovanie dát, je kľúčom k úspešnému strojovému učeniu a predstavuje dôležitý aspekt pri návrhu a implementácii regresných modelov ako je SVR (Smola & Schölkopf, 2004). Slack premenné sú nevyhnutné pre správne fungovanie modelu, najmä pri riešení dát s prítomným šumom alebo výstupnými odchýlkami. Tieto premenné, známe ako ξ_i, ξ_i^*, umožňujú modelu tolerovať chyby predikcie, ktoré presahujú určený prah ε ,bez toho, aby boli príliš penalizované. Táto vlastnosť zaisťuje, že model je odolný voči preučeniu a zároveň zachováva jeho schopnosť generalizácie na nevidené dáta.

- $\boldsymbol{\xi_i}$ meria veľkosť odchýlky, keď predpovedaná hodnota presiahne skutočnú hodnotu o viac ako ε.
- $\boldsymbol{\xi_i^*}$ meria veľkosť odchýlky, keď skutočná hodnota presiahne predpovedanú o viac ako ε.

Tieto premenné sú penalizované v rámci optimalizačného problému SVR, čo zabezpečuje, že model dokáže spracovať dáta s inherentnými odchýlkami bez straty prediktívnej schopnosti. Flexibilita, ktorú slack premenné poskytujú, je kritická pre aplikácie v reálnom svete, kde dáta často obsahujú šum alebo sú neúplné (Zhu & Hastie, 2005).

4 Kernelový SVR model

Vyššie uvedená časť popisuje lineárny model ε-SVR, ktorý pracuje so vstupnými dátami v ich priestoroch znakov a predpokladá, že funkcia *f(x)* je lineárna funkcia. Aby sme umožnili ε-SVR spracovávať nelineárne dáta, môžeme zaviesť kernelovú funkciu, ktorá transformuje pôvodné vstupné dáta do vyššieho dimenzionálneho priestoru, nazývaného kernelový priestor. Používanie kernelov je jedným z najbežnejších prístupov v SVM (pre regresiu a klasifikáciu), pretože nie je potrebné riešiť vysoko-rozmernú separačnú hyperpovrch v vstupnom priestore, čo je oveľa komplikovanejšie v porovnaní s riešením lineárnej optimalizácie v kernelovom priestore.

Optimalizačný problém je často riešený v jeho duálnej forme, ktorá umožňuje využitie kernelových funkcií pre riešenie nelineárnych vzťahov v dátach. Duálna formulácia zahŕňa Lagrangeove multiplikátory. Štandardná metóda dualizácie využívajúca Lagrangeove multiplikátory je opísaná nasledovne:

$$\begin{aligned} \boldsymbol{L} = \frac{\mathbf{1}}{\mathbf{2}}||\boldsymbol{\omega}||^2 + \boldsymbol{C}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{\xi_i} + \boldsymbol{\xi_i^*}) - \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}\boldsymbol{\alpha_i}(\boldsymbol{\varepsilon} + \boldsymbol{\xi_i} - \boldsymbol{y_i} + \langle \boldsymbol{\omega}, \mathbf{x_i}\rangle + \boldsymbol{b}) \\ - \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}\boldsymbol{\alpha_i^*}(\boldsymbol{\varepsilon} + \boldsymbol{\xi_i^*} + \boldsymbol{y_i} - \langle \boldsymbol{\omega}, \mathbf{x_i}\rangle - \boldsymbol{b}) - \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{\eta_i}\boldsymbol{\xi_i} + \boldsymbol{\eta_i^*}\boldsymbol{\xi_i^*}) \end{aligned} \tag{7}$$

Duálne premenné v (7) musia spĺňať podmienky pozitívnosti, t.j. $\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geq 0$. Zo sedlového bodu vyplýva, že parciálne derivácie L vzhľadom na primárne premenné $(\boldsymbol{\omega}, b, \xi_i, \xi_i^*)$ musia zmiznúť pre optimálnosť.

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = 0 \tag{8}$$

$$\frac{\partial L}{\partial \omega} = \boldsymbol{\omega} - \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)\mathbf{x_i} = 0 \tag{9}$$

$$\frac{\partial L}{\partial \xi_i^*} = C - \alpha_i^{(*)} - \eta_i^{(*)} = 0 \tag{10}$$

Substituovaním (8), (9) a (10) do (7) vznikne duálny optimalizačný problém.

$$max\left\{-\frac{1}{2}\sum_{i,j=0}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle \mathbf{x_i}, \mathbf{x_j}\rangle - \varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n}y_i(\alpha_i - \alpha_i^*)\right\} \tag{11}$$

Za podmienok

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$$

$$\alpha_i, \alpha_i^* \in [0, C]$$

Duálne premenné η_i, η_i^* prostredníctvom podmienky (10) boli odstránené pre odvodenie (11). Rovnicu (9) možno prepísať takto:

$$\boldsymbol{\omega} = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\mathbf{x_i} \tag{12}$$

podľa vzťahu (12) potom

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\langle \mathbf{x_i}, \mathbf{x}\rangle + b \tag{13}$$

Výraz (12) je takzvaná expanzia podporných vektorov, t.j. $\boldsymbol{\omega}$ môže byť úplne popísané ako lineárna kombinácia trénovacích vzorov $\mathbf{x_i}$. Algoritmus SV (Support Vector) môže byť nelineárny jednoduchým mapovaním trénovacích vzorov $\mathbf{x_i}$ do viacrozmerného priestoru pomocou kernelu $\varphi: X \rightarrow \Im$ a následným použitím štandardného algoritmu SV regresie. Expanzia v (12) potom vyzerá nasledovne (Basak et al., 2007):

$$\boldsymbol{\omega} = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\varphi(\mathbf{x_i}) \tag{14}$$

podľa vzťahu (14) potom

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)k(\mathbf{x_i}, \mathbf{x}) + b \tag{15}$$

kde α_i, α_i^* sú Lagrangeove multiplikátory. Kernelová funkcia $k(\mathbf{x_i}, \mathbf{x})$bola definovaná ako lineárny skalárny súčin nelineárneho zobrazenia, t.j.

$$k(\mathbf{x_i}, \mathbf{x}) = \varphi(\mathbf{x_i})\varphi(\mathbf{x}) \tag{16}$$

5 Implementácia SVR na dataset pomocou programovacieho jazyka Python

Na začiatku každého projektu z oblasti strojového učenia je kľúčové mať jasne definovaný cieľ. Keď presne vieme, čo chceme dosiahnuť, môžeme pristúpiť k zberu a prvotnej analýze dát. Následne je nevyhnutné dáta očistiť, čo zahŕňa odstránenie chýb, úpravu formátu a rozdelenie dát na tréningové, validačné a testovacie súbory. S takto pripravenými dátami môžeme vyberať vhodné modely na ich spracovanie. Po vytvorení a otestovaní modelu

nasleduje fáza jeho optimalizácie a prípadného nasadenia do praxe. Je dôležité sa nezastaviť len pri prvej verzii modelu, ale pravidelne ho aktualizovať a prispôsobovať novým podmienkam a poznatkom. V konečnej fáze je model hodnotený na testovacej množine, čo poskytuje informácie o jeho reálnej efektivite a pripravenosti na implementáciu. Proces strojového učenia je však často iteratívny, a preto je nevyhnutné model priebežne monitorovať, aktualizovať a prispôsobovať ho s ohľadom na nové požiadavky alebo zistené nedostatky.

5.1 Dataset expenses

Dataset "expenses.csv" obsahuje údaje o výške poistného a zdravotných charakteristikách jednotlivcov. Tento dataset zahŕňa informácie ako vek, pohlavie, Body Mass Index (BMI), počet detí, fajčiarske zvyky a geografickú oblasť, kde osoba žije. Taktiež obsahuje údaje o výške poistného nákladu, ktorý jednotlivci platia(ročne), na základe ktorých je možné skúmať vzťahy medzi týmito faktormi a výškou poistného**.**

Obr. 3:Pohľad na premenné datasetu expenses

	age	sex	bmi	children	smoker	region	charges
0001	19	female	27.9	0	yes	southwest	16884.924
0002	18	male	33.77	1	no	southeast	1725.5523
0003	28	male	33	3	no	southeast	4449.462
0004	33	male	22.705	0	no	northwest	21984.47061
0005	32	male	28.88	0	no	northwest	3866.8552
0006	31	female	25.74	0	no	southeast	3756.6216
0007	46	female	33.44	1	no	southeast	8240.5896
0008	37	female	27.74	3	no	northwest	7281.5056

Zdroj: Vlastné spracovanie

Dataset teda obsahuje nasledujúce premenné:

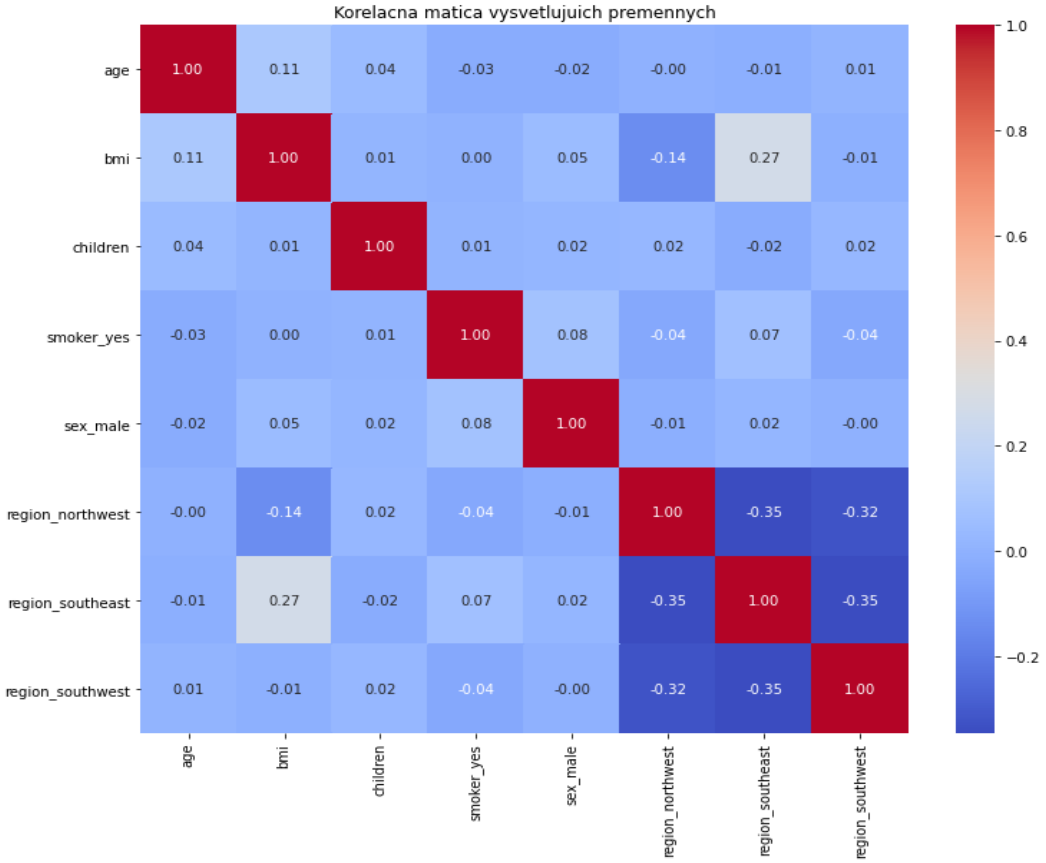
- **age (vek)**: Vek jednotlivca.
- **sex (pohlavie)**: Pohlavie jednotlivca (muž alebo žena).
- **bmi**: Index telesnej hmotnosti, ktorý je meradlom telesného tuku na základe výšky a váhy.
- **children (deti)**: Počet detí/závislých osôb, ktoré sú pokryté zdravotným poistením.
- **smoker (fajčiar)**: Udáva, či je jednotlivec fajčiar (áno alebo nie).
- **region (región)**: Región bydliska jednotlivca v Spojených štátoch (severovýchod, severozápad, juhovýchod, juhozápad).
- **charges (poplatky)**: Poplatky za zdravotné poistenie fakturované jednotlivcovi.

Zobrazené hodnoty v korelačnej matici na obrázku 4 sú hodnoty Pearsonovho korelačného koeficientu. Táto korelačná matica vysvetľujúcich premenných v datasete odhaľuje, že medzi väčšinou premenných sú veľmi slabé alebo žiadne korelácie, čo naznačuje nízku úroveň multikolinearity, čo je priaznivé pre modelovanie. Celkovo táto analýza naznačuje, že premenné sú vhodné na použitie v regresných modeloch, pretože nízka multikolinearita by nemala viesť k nestabilite modelu.

Blok kódu na obrázku 5 importuje potrebné knižnice a moduly, ktoré sa používajú na spracovanie dát, modelovanie a vizualizáciu v projekte.

Blok kódu na obrázku 6 načíta dataset zo súboru expenses.csv, vykoná one-hot encoding pre kategorizované premenné smoker, sex a region, rozdelí dáta na vstupné premenné `X` a cieľovú premennú `y`, a následne rozdelí dáta na tréningovú a testovaciu množinu v pomere 80:20. Tento pripravený dataset je potom pripravený na použitie v rôznych modeloch strojového učenia.

Obr. 4: Korelačná matica vysvetľujúcich premenných



Zdroj: Vlastné spracovanie

Obr. 5: Použité knižnice

```
import pandas as pd
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns
```

Zdroj: Vlastné spracovanie

Obr. 6: Práca s datasetom

```
data_path = 'C:\\Users\\PC\\Desktop\\expenses.csv'
hr_data = pd.read_csv(data_path)
hr_data = pd.get_dummies(hr_data, columns=['smoker', 'sex', 'region'], drop_first=True)
X = hr_data.drop('charges', axis=1)
y = hr_data['charges']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Zdroj: Vlastné spracovanie

Obr. 7: Nastavenie modelov a ich hyperparametrov

```
models = {
    'Linear Regression': LinearRegression(),
    'SVR (linear kernel)': make_pipeline(StandardScaler(), SVR(kernel='linear', C=15000, epsilon=800)),
    'SVR (poly kernel)': make_pipeline(StandardScaler(), SVR(kernel='poly', C=15000, epsilon=800, gamma='scale')),
    'SVR (rbf kernel)': make_pipeline(StandardScaler(), SVR(kernel='rbf', C=15000, epsilon=800, gamma='scale')),
    'SVR (sigmoid kernel)': make_pipeline(StandardScaler(), SVR(kernel='sigmoid', C=15000, epsilon=800, gamma='scale'))
}
```

Zdroj: Vlastné spracovanie

Blok kódu na obrázku 7 definuje slovník modelov s rôznymi typmi regresných modelov, vrátane lineárnej regresie a SVR s rôznymi kernelmi. Každý SVR model je obalený v `pipeline`, ktorá zahŕňa štandardizáciu vstupných dát pomocou `StandardScaler` a aplikáciu SVR s konkrétnymi hyperparametrami (pre dataset expenses sa po manuálnom testovaní hyperparametre nastavené na C=15000, ε=800, γ='scale' ukázali ako najefektívnejšie). Každý dataset má svoje vlastné charakteristiky, ako je štruktúra, množstvo šumu a komplexnosť vzťahov medzi premennými, čo ovplyvňuje optimálne nastavenie hyperparametrov.

Hyperparametre sú parametre, ktoré výrazne ovplyvňujú výkon a schopnosť modelu generalizovať. V kontexte Support Vector Regression (SVR) s rôznymi kernelmi sú hlavnými hyperparametrami C, epsilon (ε) a gamma (γ). Tu je vysvetlenie, ako tieto hyperparametre ovplyvňujú model:

Hyperparameter C (Regularizačný parameter):

- Funkcia**:** C určuje, ako veľmi chce model minimalizovať chyby. Vyvažuje medzi dvoma cieľmi: minimalizovať chyby na tréningovej množine a udržať model jednoduchý (vyhnúť sa overfittingu).
- Vplyv:
    - Vysoké hodnoty C**:** Model sa snaží minimalizovať chyby na tréningových dátach, čo môže viesť k overfittingu.
    - Nízke hodnoty C: Model je viac penalizovaný za chyby a môže mať tendenciu byť jednoduchší a generalizovať lepšie na nové dáta, ale môže trpieť underfittingom.

Hyperparameter epsilon (ε) v epsilon-insensitive loss funkcii:

- Funkcia: Epsilon určuje šírku pásma okolo predikovaných hodnôt, v ktorom sa chyby neberú do úvahy. Toto pásmo sa nazýva epsilon-tube.
- Vplyv:
    - Vysoké hodnoty epsilon**:** Väčšie pásmo, kde sa chyby ignorujú. Model môže byť menej presný, pretože viacej chýb sa nezohľadňuje.
    - Nízke hodnoty epsilon**:** Menšie pásmo, čo znamená, že model sa snaží presne predpovedať hodnoty s menšími chybami. Môže to viesť k lepšiemu prispôsobeniu sa tréningovým dátam, ale tiež k vyššiemu riziku overfittingu.

Hyperparameter gamma (γ) v RBF a Poly kerneloch:

- Funkcia: Gamma určuje, ako ďaleko dosahuje vplyv jednotlivých tréningových príkladov. Ovláda polomer rozhodovacieho regiónu.
- Vplyv:
    - Vysoké hodnoty gamma**:** Každý tréningový príklad má malý dosah, čo vedie k veľmi flexibilnému modelu, ktorý sa môže prispôsobiť šumu v dátach a viesť k overfittingu.

- Nízke hodnoty gamma: Každý tréningový príklad má väčší dosah, čo vedie k hladkému modelu, ktorý môže lepšie generalizovať, ale môže trpieť underfittingom, ak je gamma príliš nízke.

Obr. 8: Trénovanie modelov

```
results = {}
predictions = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    results[name] = (mse, r2)
    predictions[name] = y_pred
    print(f"{name} - MSE: {mse}, R^2: {r2}")
```

Zdroj: Vlastné spracovanie

Blok kódu na obrázku 8 trénuje rôzne modely, predikuje hodnoty na testovacej množine, vypočíta metriky výkonu (*MSE a* R^2) a ukladá výsledky pre každý model. Najprv sú inicializované prázdne slovníky `results` a `predictions`. Potom sa iteruje cez všetky modely definované v slovníku models. Pre každý model sa vykoná tréning na tréningovej množine `X_train` a `y_train` pomocou metódy fit. Následne model predikuje hodnoty na testovacej množine `X test` pomocou metódy predict. Vypočítajú sa metriky výkonu: *MSE* (Mean Squared Error), ktorý meria priemerný štvorec rozdielov medzi skutočnými a predikovanými hodnotami, a R^2 (R-squared), ktorý meria, aká časť variability závislej premennej je vysvetlená nezávislými premennými. Tieto hodnoty sú uložené v slovníku results pod názvom modelu. Predikované hodnoty `y pred` sú uložené v slovníku predictions pod názvom modelu. Nakoniec sa výsledky pre každý model vytlačia vo formáte, ktorý zobrazuje názov modelu, hodnoty MSE a R^2.

Obr. 9: Výsledky MSE a koeficientov determinacie modelov

```
Linear Regression - MSE: 18487911.523133304, R^2: 0.8733561613662507
SVR (linear kernel) - MSE: 28873875.77526806, R^2: 0.8022113822949408
SVR (poly kernel) - MSE: 4948200.620824575, R^2: 0.9661043855512286
SVR (rbf kernel) - MSE: 3018203.824105105, R^2: 0.9793250353029085
SVR (sigmoid kernel) - MSE: 4334971259.047177, R^2: -28.69493876719227
```

Zdroj: Vlastné spracovanie

MSE je priemer štvorcov rozdielov medzi skutočnými a predikovanými hodnotami. Nižšie hodnoty *MSE* naznačujú lepší model, pretože rozdiely medzi skutočnými a predikovanými hodnotami sú menšie. Vyššie hodnoty *MSE* naznačujú horší model, pretože rozdiely medzi skutočnými a predikovanými hodnotami sú väčšie.R^2 meria, aká časť variability závislej premennej (skutočné hodnoty) je vysvetlená nezávislými premennými (predikované hodnoty). Hodnoty R^2 sa pohybujú od 0 do 1.

- Hodnota blízka 1: Model veľmi dobre vysvetľuje variabilitu dát. Vysoké hodnoty R^2 naznačujú, že model je presný.

- Hodnota blízka 0: Model nevysvetľuje variabilitu dát. Nízke hodnoty R^2 naznačujú, že model nie je presný.
- Negatívna hodnota R^2: Model je horší ako jednoduchý priemer skutočných hodnôt. To sa stáva, keď je model veľmi nepresný.

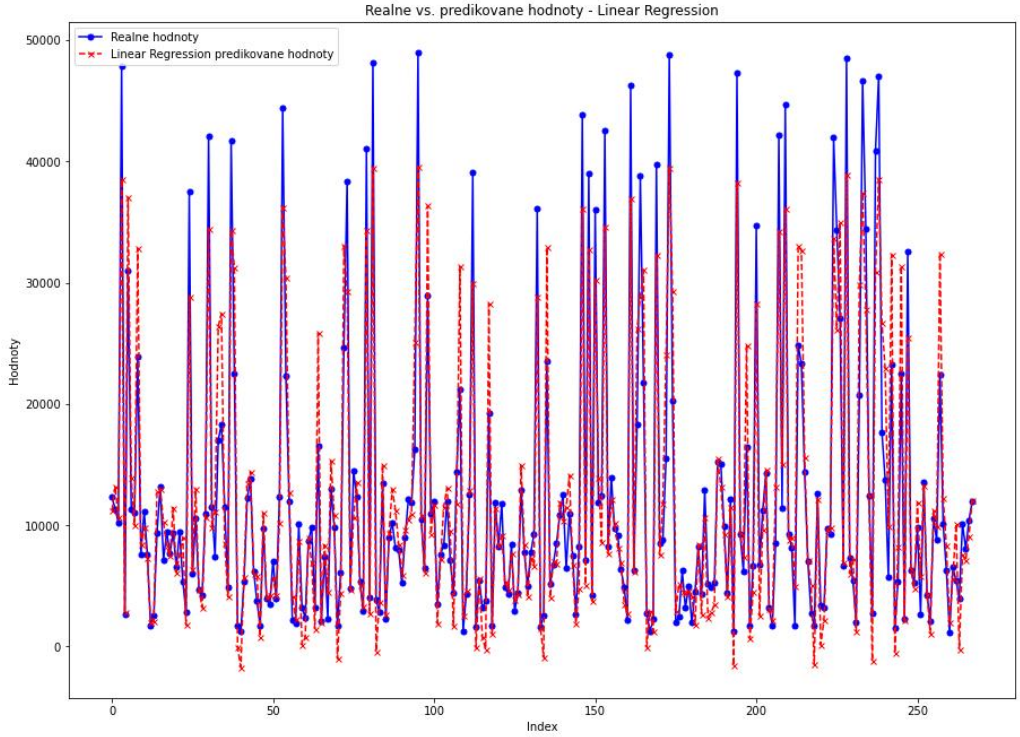
Na základe výsledkov zobrazených na obrázku 9, SVR s RBF kernelom dosahuje najlepší výkon, pretože má najnižšie MSE a najvyššie R^2, čo naznačuje, že najlepšie zachytáva zložité vzory v dátach. SVR s polynomiálnym kernelom tiež dosahuje dobrý výkon, ale nie taký dobrý ako RBF kernel. Lineárna regresia a SVR s lineárnym kernelom majú slušný výkon, ale sú horšie v porovnaní s RBF a poly kernelmi. SVR so sigmoid kernelom má veľmi zlý výkon a neodporúča sa pre tento dataset.

Obr. 10: Porovnanie skutočných a predikovaných hodnôt nákladov spolu s vysvetľujúcimi premennými pomocou modelu SVR (RBF kernel) pre náhodných 10 záznamov testovacej sady

age	bmi	children	smoker_yes	sex_male	region_northwest	region_southeast	region_southwest	Actual Charges	Predicted Charges (SVR rbf kernel)
23	23.180	2	False	False	True	False	False	3180.51010	3522.806766
53	22.610	3	True	False	False	False	False	24873.38490	22839.468760
38	40.150	0	False	False	False	True	False	5400.98050	5207.478675
26	32.900	2	True	True	False	False	True	36085.21900	31217.831135
31	25.900	3	True	True	False	False	True	19199.94400	19647.316551
31	25.800	2	False	False	False	False	True	4934.70500	4551.568099
24	32.700	0	True	True	False	False	True	34472.84100	32215.236296
37	30.875	3	False	True	True	False	False	6796.86325	6434.605838
46	35.530	0	True	False	False	False	False	42111.66470	42140.144084
26	35.420	0	False	True	False	True	False	2322.62180	2578.579903

Zdroj: Vlastné spracovanie

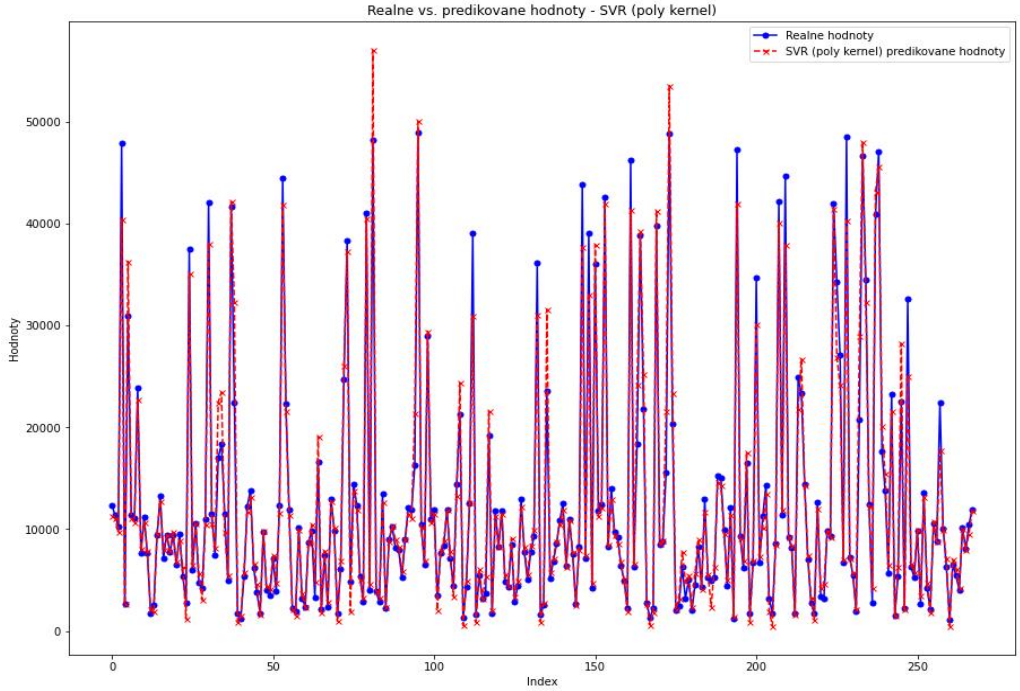
Obr. 11: Reálne vs. Predikované hodnoty(Lineárna regresia)



Zdroj: Vlastné spracovanie

Graf na obrázku 11 zobrazuje porovnanie reálnych a predikovaných hodnôt pomocou lineárnej regresie s koeficientom determinácie $R^2 = 0.873$.

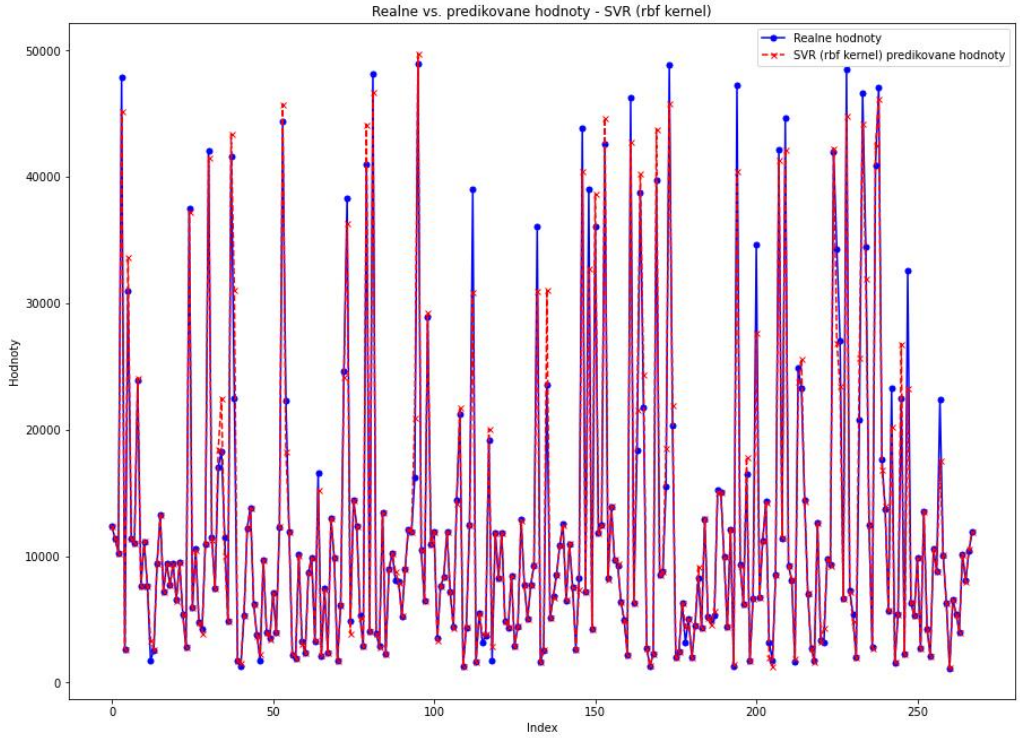
Obr. 12: Reálne vs. Predikované hodnoty(SVR s poly kernelom)



Zdroj: Vlastné spracovanie

Graf na obrázku 12 zobrazuje porovnanie reálnych a predikovaných hodnôt pomocou modelu SVR(kernel=poly) s koeficientom determinácie $R^2 = 0.966$.

Obr. 13: Reálne vs. Predikované hodnoty(SVR s RBF kernelom)



Zdroj: Vlastné spracovanie

Graf na obrázku 13 zobrazuje porovnanie reálnych a predikovaných hodnôt pomocou modelu SVR(kernel=RBF) s koeficientom determinácie $R^2 = 0.979$.

6 Záver

Podľa analýzy hodnôt MSE (Mean Squared Error) a koeficientu determinácie R^2 sa model Support Vector Regression (SVR) ukázal ako efektívnejší oproti lineárnej regresii pri odhade poistného. SVR modely s rôznymi kernelmi, najmä s RBF kernelom, poskytovali presné predikcie a efektívne vysvetľovali variabilitu v údajoch. Na druhej strane, lineárna regresia vykázala nižšiu presnosť a efektivitu v porovnaní s SVR modelmi. Výnimkou bol SVR so sigmoid kernelom, ktorý vykazoval najhoršie výsledky pravdepodobne z dôvodu, že na rozdiel od RBF kernelu, ktorý je veľmi efektívny pri zachytávaní nelineárnych vzťahov medzi dátovými bodmi, sigmoidálny kernel nemusí byť dostatočne flexibilný na zachytenie zložitých vzťahov v dátach.

Dôležité je pripomenúť, že účinnosť modelu SVR závisí od dôkladného nastavenia hyperparametrov, ako sú parametre C, epsilon a gamma, a vyžaduje presné škálovanie vstupných dát. Tieto parametre majú kritický vplyv na výkon modelu a vyžadujú starostlivú optimalizáciu. V prípade datasetu s poistnými nákladmi, SVR s RBF kernelom poskytol najpresnejšie predikcie s hodnotou R^2 0.9793, čo naznačuje, že bol schopný najlepšie vysvetliť variabilitu v údajoch. V konečnom dôsledku, aj keď je SVR časovo náročnejší na nastavenie a vyžaduje intenzívnejšie zdroje na optimalizáciu, jeho schopnosť poskytovať mimoriadne presné predikcie ho robí cenným vo vysoko špecializovaných aplikáciách, kde je kritická maximálna prediktívna presnosť a sú dostupné zdroje na jeho dôkladnú kalibráciu a údržbu.

Tento príspevok vznikol v rámci výskumného projektu VEGA 1/0431/22, *Implementácia inovatívnych prístupov modelovania rizík v procese ich riadenia v interných modeloch poisťovní v kontexte s požiadavkami direktívy Solvency II.*

Literatúra

1. Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.
2. Basak, D., Pal, S., & Patranabis, D. C. (2007). Support vector regression. *Neural Information Processing-Letters and Reviews*, *11*(10), 203-224.
3. Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3), Article 27. https://doi.org/10.1145/1961189.1961199
4. Hofmann, T., Schölkopf, B., & Smola, A. J. (2008). Kernel methods in machine learning. *The Annals of Statistics*, 36(3), 1171-1220. https://doi.org/10.1214/009053607000000677
5. Lins, I. D. et al, (2013). Sea Surface Temperature prediction via Support Vector Machines combined with Particle Swarm Optimization. *Expert Systems with Applications*, 40(5), 1766-1779.
6. Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge: Cambridge University Press.
7. Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3), 199-222. https://doi.org/10.1023/B:STCO.0000035301.49549.8
8. Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc. https://doi.org/10.1007/978-1-4757-2440-0
9. Zhang, F., & O'Donnell, L. J. (2020). Support vector regression. *In Machine Learning* (pp. 123–140). Elsevier. https://doi.org/10.1016/B978-0-12-815739-8.00007-9
10. Zhu, J., & Hastie, T. (2005). Kernel Logistic Regression and the Import Vector Machine. *Journal of Computational and Graphical Statistics*, 14(1), 185–205. https://doi.org/10.1198/106186005X25619

Aktuálne trendy v CRM aplikáciách

Current Trends in CRM Applications

Pavol Jurík[1]

Abstrakt
Budovanie vzťahu so zákazníkom a starostlivosť o neho by mala byť nepostrádateľnou zložkou fungovania každého úspešného podniku, ktorý sa chce presadiť oproti svojej konkurencii. Čím väčšia je konkurencia v danom odvetví, tým väčšmi táto zásada naberá na význame. Na podporu komunikácie medzi firmou a jej zákazníkmi, a tiež na podporu evidencie a vyhodnocovania dôležitých údajov o zákazníkoch slúžia aplikácie, ktoré sa označujú skratkou CRM (Customer Relationship Management, t. j. riadenie vzťahov so zákazníkmi). V tomto článku sa zameriame na stručnú charakteristiku toho, akým spôsobom sa dajú takéto aplikácie použiť pri podnikaní a identifikujeme aktuálne trendy v ich vývoji. Povedomie o týchto trendoch môže byť užitočné nielen pre softvérové firmy vyvíjajúce takéto aplikácie, aby sa so svojimi aplikáciami dokázali presadiť na trhu a neboli porazení v boji s konkurenciou, ale nepochybne aj pre podniky, ktoré takéto aplikácie využívajú na podporu svojich marketingových aktivít a budovania úspešných a dlhotrvajúcich vzťahov so svojimi zákazníkmi.

Kľúčové slová
CRM, riadenie vzťahov so zákazníkmi, marketing, segmentácia trhu, komunikácia so zákazníkmi

Abstract
Building a relationship with customers and taking care of them should be an indispensable component of the operation of any successful business that wants to stand out against its competition. The greater the competition in a given industry, the more important this principle becomes. Applications known as CRM (Customer Relationship Management) are used to support communication between the company and its customers, as well as to support the recording and evaluation of important customer data. In this article, we will focus on a brief description of how such applications can be used in business and identify current trends in their development. Awareness of these trends can be useful not only for software companies developing such applications so that they can establish themselves in the market with their applications and not be defeated in the fight with the competition, but undoubtedly also for businesses that use such applications to support their marketing activities and build successful and long-lasting relationships with their customers.

Key words
CRM, customer relationship management, marketing, market segmentation, customer communication

JEL classification
M30, M31, M37

[1] Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1, 852 35 Bratislava, pavol.jurik@euba.sk.

1 Úvod

CRM (Customer Relationship Management) je skratka pre aplikácie na podporu riadenia vzťahov so zákazníkmi. Prvotnou úlohou CRM je zjednotiť a centralizovať kontakty a komunikáciu so zákazníkmi. Ide teda o aplikácie zamerané na podporu podniku v úsilí predať svoje výrobky a služby. Podniky sa v súčasnosti usilujú o vytvorenie trvalých vzťahov so zákazníkmi a informačno-komunikačné technológie (IKT) im pri tom môžu výraznou mierou pomáhať.

Medzi formy elektronickej starostlivosti o zákazníkov môžeme zaradiť najmä:

- firemné webstránky, ktoré prehľadným spôsobom poskytujú relevantné informácie,
- zasielanie e-mailov a SMS reklamného charakteru (nesmú však zákazníka obťažovať),
- organizovanie webových diskusií (videokonferencií, webkonferencií) – ide o praktický komunikačný kanál, pomocou ktorého sa zákazníci môžu dozvedieť mnohé relevantné informácie o tej-ktorej firme a jej produkcii, pričom sa takejto akcie môžu zúčastniť z pohodlia domova, čo šetrí čas aj náklady nielen im, ale aj samotným organizátorom podujatia.
- diskusné fóra a živý čet,
- call centrá – ich využitie je dvojaké:
    - umožňujú zákazníkom vybavovať ich požiadavky alebo sťažnosti telefonicky, pričom ich štandardizovaným spôsobom obslúži práve neobsadený pracovník call centra,
    - umožňujú samotnej firme obvolávať existujúcich aj potenciálnych zákazníkov a oslovovať ich so svojou ponukou (na niektorých zákazníkov však toto môže pôsobiť rušivo).

Keďže mnohé podniky sa v súčasnosti usilujú o veľmi účinnú viac-kanálovú reklamnú kampaň zacielenú na rozličné skupiny zákazníkov, resp. trhové segmenty, môže byť pre ne prínosné zaobstarať si softvér na podporu evidencie, plánovania a uskutočňovania marketingových kampaní, ako aj na ich vyhodnocovanie a hlbšiu analýzu ich efektivity na základe získaných údajov. Výsledky takýchto analýz môže podnik využiť na kontinuálne zlepšovanie svojich marketingových stratégií, určených nielen na získanie nových zákazníkov, ale aj na udržanie existujúcich. Neefektívne vedená marketingová kampaň môže podnik totiž stáť nemalé peniaze a neprinášať pritom očakávaný účinok.

Aplikácie typu CRM môžu pre podnik poskytovať predovšetkým tri typy funkcionality:

- údajovú,
- komunikačnú,
- analytickú.

Údajová funkcionalita CRM spočíva v tom, že aplikácia podporuje vytvorenie databázy, v ktorej sú centralizované kontakty a všetky dôležité údaje o jednotlivých zákazníkoch. Do tejto databázy sa ukladajú aj údaje získané realizáciou marketingových kampaní, ktoré sa môžu analyzovať v rámci vyhodnocovania spätnej väzby od jednotlivých cieľových skupín zákazníkov, na ktoré boli tieto kampane smerované. Okrem toho sa každá interakcia (t. j. komunikácia) so zákazníkom pridáva do histórie kontaktov s ním. Pracovníci call centra sa potom na základe archivovanej histórie môžu pozrieť na to, aké obchody boli doposiaľ s týmto zákazníkom zrealizované, aké tovary, príp. služby si objednal, v akých množstvách, aké problémy (sťažnosti, požiadavky, reklamácie) sa v súvislosti s ním v minulosti riešili a pod. Pracovník call centra má vďaka tomu optimálny prehľad o dôležitých informáciách, ktoré sa týkajú jednotlivých zákazníkov, a tomu môže prispôsobovať spôsob svojej komunikácie s nimi. Aplikácia by mala umožňovať jednoduchú, intuitívnu a rýchlu prácu s údajmi v databáze, ako aj pridávanie nových údajov, či ich editáciu v prípade potreby. Vymazávanie údajov nie je

zvykom, pretože nikdy nevieme, kedy sa k nám zákazník znova vráti a žiadny zákazník nie je potenciálne celkom stratený.

Komunikačná funkcionalita CRM sa zameriava na evidenciu a podporu elektronickej komunikácie so zákazníkmi. Komunikácia medzi firmou a jej zákazníkmi vo všeobecnosti môže prebiehať aj fyzickou (tvárou v tvár), telefonickou či písomnou formou (pomocou klasickej pošty), ale aj pomocou televíznej reklamy a reklamy v časopisoch a novinách, pomocou bilboardov, letákov a pod. Avšak v prípade aplikácií CRM ide o podporu komunikácie elektronickou formou. Môže ísť o hovory realizované na základe internetovej telefónie (služba VoIP, Voice over Internet Protocol), komunikáciu formou videokonferencií, webkonferencií, diskusných fór, živého četu a pod.

Analytická funkcionalita CRM sa orientuje na analýzu získaných údajov o zákazníkoch, a to z rozličných uhlov pohľadu. Cieľom je najmä objektívne vyhodnocovanie dokončených, ale aj priebežné vyhodnocovanie práve prebiehajúcich marketingových kampaní a návrh možností na ich zlepšenie v budúcnosti. Pri analýze zákazníkov je vhodné uplatniť *princíp segmentácie trhu*, teda zákazníkov rozčleniť na viaceré kategórie podľa rozličných parametrov, a potom analyzovať každú kategóriu zvlášť. Ak sú zákazníkmi *jednotlivci*, sledovanými parametrami môžu byť (ManagementMania, 2015):

- *geografické parametre* – región, krajina, kontinent a pod.,
- *demografické parametre* – vek, pohlavie, etnikum, náboženstvo, rodinný stav a pod.,
- *socioekonomické parametre* – socioekonomický status, napr. vzdelanie, povolanie, príjem, spoločenské alebo pracovné postavenie a pod.,
- *psychologické parametre* – napr. životné záujmy, postoje, hodnoty a pod.,
- *parametre charakterizujúce nákupné správanie* – frekvencia a rozsah nákupov, lojalita k danej firme (t. j. či sa zákazník špecializuje na určitú značku, alebo nakupuje od rozličných firiem), postoj k riziku a pod.

Ak zákazníkmi sú *firmy*, a nie jednotlivci, sledovanými parametrami (ManagementMania, 2015):

- *geografické parametre* – región, krajina, kontinent a pod.,
- *parametre charakterizujúce organizáciu* – sektor, odvetvie alebo odbor, v ktorom daná organizácia pôsobí, veľkosť organizácie (malý, stredný či veľký podnik), kultúra organizácie (podnikové zvyky a hodnoty) a pod.,
- *parametre charakterizujúce prevádzku* – typ výroby, organizácia nákupu, naliehavosť dodávok (napr. zásobovanie typu Just-in-Time), kvalitatívne požiadavky na nakupované tovary alebo služby a pod.,
- *parametre charakterizujúce nákupné správanie* – nákupná politika organizácie (nákupná stratégia), kritériá nákupu a pod.

Cieľom systematického sledovania a vyhodnocovania rozličných skupín, resp. kategórií zákazníkov je snaha zistiť, ktoré skupiny zákazníkov oslovujume ako firma so svojimi produktmi viac a ktoré skupiny oslovujeme menej, resp. vôbec. Vďaka tomu môže daná firma popremýšľať, čo urobiť za účelom toho, aby sa jej produkty stala zaujímavými aj iné skupiny zákazníkov, príp. sa pokúsiť ešte viac zaujať tie skupiny, ktoré už teraz o produkty danej firmy javia určitú mieru záujmu.

Okrem vyhodnocovania úspešnosti a prínosu marketingových kampaní môžeme analyzovať napr. aj reakcie rozličných segmentov zákazníkov na zmeny v ponúkaných výrobkoch (zmena zloženia, zmena dizajnu výrobku, zmena obalu, zmena vo funkcionalite a pod.). Takýmito zmenami môžeme totiž získať nových zákazníkov, ale môžeme aj niektorých stratiť, pretože im nová verzia produktu so zmenenými parametrami už nevyhovuje a nespĺňa ich očakávania.

Súčasťou starostlivosti o zákazníka je aj poskytovanie servisu. *Servis* pritom môže byť:

- *predpredajný* – poradenstvo, predvedenie alebo porovnanie rozličných produktov, prednášky, semináre, možnosť vyskúšať si produkt a pod.,
- *predajný* – rýchly a bezproblémový predaj, aktívne riešenie aktuálnych potrieb zákazníka,
- *popredajný* – inštalácia produktu, jeho úpravy, záruka, vybavovanie reklamácií a sťažností, vrátenie tovaru, dočasná náhrada produktu a pod.

Optimálna CRM aplikácia by mala umožňovať evidenciu a podporovať realizáciu všetkých troch uvedených typov servisu. Dôsledná analýza zákazníka, jeho priorít a potrieb prostredníctvom CRM môže prispieť k tomu, aby prístup firmy k nemu bol čo najindividuálnejší (teda prispôsobený „na mieru“ konkrétnemu zákazníkovi) a aby sa z neho stal zákazník lojálny voči firemnej značke, resp. jej produktom či poskytovaným službám. Pre úplnosť však treba dodať, že ani aplikácie typu CRM nie sú samospasiteľné a ich prínos závisí aj od kreativity, schopností a pracovného nasadenia či nadšenia pracovníkov oddelenia marketingu a od toho, či vedia aplikáciu na podporu riadenia vzťahov so zákazníkmi dostatočne využiť. CRM aplikácie sú teda síce vhodným prostriedkom na zlepšovanie vzťahov so zákazníkom, no ich efektivita je do značnej miery podmienená ľudským faktorom.

2 Typy CRM aplikácií

V literatúre sa môžeme stretnúť s viacerými typmi CRM aplikácií podľa úžitku, ktorý prinášajú pre podnik. Ide najmä o tieto typy:

- *Strategické CRM* – jednou z odnoží riadenia vzťahov so zákazníkmi je považovať tento vzťah za celopodnikovú stratégiu, pri ktorej podnik kladie čo najväčší dôraz na zákazníka a jeho spokojnosť s cieľom zabezpčiť jeho loajalitu k danej značke. Analýza údajov o zákazníkov potom môže vrcholnému manažmentu podniku slúžiť ako podklad pri rozhodovaní sa o základných strategických cieľoch podniku na najbližšie roky, či stanovovanie dlhodobejších vízií (Wahlberg et al., 2009).
- *Operatívne CRM* – poskytuje podporu CRM procesom ako interakcia so zákazníkmi, ukladanie dát a kontaktov, prípadne dáta ohľadom histórie komunikácie podniku so zákazníkmi. Používaním operatívneho CRM podnik a jeho zamestnanci získavajú možnosť dohľadať a priamo získať tieto informácie prostredníctvom CRM aplikácie, čo umožňuje podniku reagovať vhodne a včas smerom k zákazníkom, ich požiadavkám a potrebám.
- *Analytické CRM* – tento typ CRM systému analyzuje dáta získané od zákazníkov a používa ich pre dizajnovanie a zriaďovanie marketingových kampaní, ktoré majú predpoklad na získanie nových klientov. Taktiež analytické CRM analyzuje tieto dáta pre účel získania informácií, podľa ktorých sú vrámci firmy rozhodované strategické ale aj operatívne rozhodnutia. Niektoré analytické CRM systémy poskytujú možnosť finančnej predpovede na základe zozbieraných dát ale ponúkajú aj modul, ktorý umožňuje analyzovať potencionálnu ziskovosť zákazníkov.
- *Campaign management CRM* (*CRM na riadenie marketingových kampaní*) – umožňuje organizácií cieliť marketingovú kampaň na základe kritérií podľa ktorých CRM aplikácia umožňuje filtráciu zákazníkov na špecifické skupiny. Tieto marketingové kampane a ich materiál môže byť klientom zasielaný prostredníctvom e-mailov (mailing), telefonicky, cez SMS alebo prostredníctvom iných platforiem. Campaign manažment CRM umožňuje aj následnú analýzu štatistík vytvorených kampaní a následné analyzovanie trendov (Wahlberg et al., 2009).
- *Kooperatívne CRM* – sústreďuje sa na komunikáciu so zákazníkom prostredníctvom rôznych komunikačných kanálov a na jej optimalizáciu. CRM umožňuje zdieľanie všetkých získaných dát ohľadom klientov naprieč celou spoločnosťou, čo má za dôsledok

dosiahnutie vyššej kvality interakcie so zákazníkmi. Tento typ CRM systému umožňuje rýchlu reakciu organizácie na požiadavky zákazníkov (Scheiner, 2024).

3 Aktuálne trendy v CRM aplikáciách

V oblasti CRM aplikácií je možné identifikovať určité trendy. Povedomie o týchto trendoch môže byť užitočné nielen pre softvérové firmy vyvíjajúce takéto aplikácie, aby sa so svojimi aplikáciami dokázali presadiť na trhu a neboli porazení v boji s konkurenciou, ale nepochybne aj pre podniky, ktoré takéto aplikácie využívajú na podporu svojich marketingových aktivít a budovania úspešných a dlhotrvajúcich vzťahov so svojimi zákazníkmi. Je totiž dobré vedieť o dostupných možnostiach a na základe toho si spomedzi ponúkaných aplikácií zvoliť tú, ktorá najviac vyhovuje potrebám a cieľom daného podniku. Medzi aktuálne trendy môžeme zaradiť najmä:

- *Mobilné CRM* – v súčasnej dobe môžeme konštatovať, že ľudská spoločnosť do veľkej miery „podľahla“ čaru mobilných telefónov a smartfónov (možno až príliš) a množstvo ľudí sa od týchto zariadení doslova nevie odtrhnúť. Tento trend sa zákonite musel prejaviť aj v oblasti CRM aplikácií. Dôvod je jednoduchý. Marketingoví pracovníci nachádzajúci sa v „teréne“ potrebujú okamžitý prístup ku všetkým dôležitým údajom o konkrétnom zákazníkovi a CRM systém, s ktorým sa nedá pracovať cez mobilný telefón, by bol pre nich značne nevýhodný. Okrem toho, aj samotní zákazníci očakávajú možnosť komunikovať s danou firmou prostredníctvom svojich mobilných zariadení. Mobilné CRM je teda výhodné pre samotnú firmu aj pre jej klientov a umožňuje obom stranám byť v lepšom vzájomnom spojení (Lendel & Kubina, 2010). Medzi CRM aplikácie, ktoré je možné používať aj na mobilných zariadeniach patria napr. HubSpot CRM, Creatio a Zoho CRM.
- *Implementácia umelej inteligencie a strojového učenia* pre inteligentnú automatizáciu procesov. Tieto technológie umožňujú automatizovať a personalizovať interakcie so zákazníkmi na úrovni, akú sme si predtým len ťažko dokázali predstaviť. Proaktívne odporúčania produktov, personalizované ponuky, predikcia správania zákazníkov a inteligentné analytické nástroje sú len niektoré z príkladov, ako podniky môžu využívať umelú inteligenciu na optimalizáciu svojich CRM procesov. Implementácia umelej inteligencie je aktuálne jedným z najpopulárnejších trendov nie len v oblasti CRM softvéru, je teda logické, že bude táto možnosť veľkou výhodou oproti konkurencii. Medzi moderné CRM nástroje využívajúce umelú inteligenciu patria napr. HubSpot CRM, Freshsales, Pipedrive, Zoho CRM, Zendesk Sell a iné (Vaughan, 2024).
- *Migrácia na cloudové riešenia* – toto je jedna z najpopulárnejších technologických inovácií, ktorá sa premietla aj do oblasti CRM. Vzhľadom na modularitu, škálovateľnosť a dostupnosť cloudových systémov začali tento trend prirodzene sledovať aj samotný vývojári aplikácií určených pre CRM (Wang, 2016). Cloudové riešenie znamená, že aplikácia nie je prevádzkovaná na interných serveroch podniku, ktorý ju používa, ale prevádzkuje ju externý poskytovateľ takejto služby za pravidelný poplatok (podľa času alebo aj rozsahu používania aplikácie). Táto aplikácia je potom pre používateľov dostupná cez webový prehliadač. Ide o cloud computingový model, ktorý sa označuje ako Software as a Service (SaaS, softvér ako služba). Výhody migrácie na cloudové riešenia sú:
    - *nižšie náklady na údržbu* – v prípade clodového riešenia spoločnosti odpadá povinnosť riešiť náklady na energie, správu daného servera, či iných služieb a úkonov spojených so zabezpeččovaním prevádzky servera,
    - *dostupnosť zálohy* – zálohovanie dát je zväčša vykonávané zo strany poskytovateľa daného cloudového úložiska. Väčšinou ide o viacnásobné

zálohovanie vykonané prostredníctvom viacerých serverov v rôznych svetových lokáciách z dôvodu väčšej bezpečnosti a nižšej šance na stratu dát organizácie.

Medzi nevýhody cloudového riešenia môžeme zaradiť najmä:

- *otáznu bezpečnosť uložených údajov* – keďže údaje nemáme uložené priamo v našom podniku, ale ich uchováva a spravuje externá firma, ktorá nám poskytuje cloudové služby, nemáme úplnú istotu, či sú tieto údaje naozaj v bezpečí, či nie sú poskytované tretím stranám (napr. konkurencii) alebo iným spôsobom zneužívané.
- Náklady na cloudovú licenciu – za možnosť používania cloudového riešenia CRM systému podnik platí poskytovateľovi tejto služby poplatok, ktorý je väčšinou pravidelný. Pravidelnosť záleží na dohodnutom fakturačnom období. Väčšinou ide o mesačnú alebo ročnú fakturáciu. Pri dlhodobom používaní takýchto služieb sa preto možno viac oplatí prevádzkovať CRM aplikáciu na vlastných serveroch (tzv. „in house“ riešenie).

- *Prehľadné spracovanie údajov* – moderný CRM systém (ako napr. Salesforce CRM) by mal umožňovať vytvárať komplexnejšie reporty, prognózy, grafy či infografiky priamo v CRM a prezentovať údaje racionálnym a koherentným spôsobom tak, aby tieto výstupy predstavovali vhodné podklady vykonávanie relevantných rozhodnutí príslušnými pracovníkmi podniku. Moderné CRM by mali umožňovať spracovávať rozsiahle kvantá údajov (tzv, big data), a to najmä kvôli schopnosti zabezpečiť nasledujúce prínosy pre podnik (Rolustech, 2017):
  1. *Prediktívne modelovanie* – potreby a preferencie dnešných digitálne vybavených zákazníkov sa neustále vyvíjajú. Big Data umožňujú firmám predpovedať, ako budú zákazníci reagovať v budúcnosti na základe ich minulého a súčasného nákupného správania, a podľa toho im prezentovať odporúčania. Čím precíznejšie a adekvátnejšie odporúčania daný softvér dokáže poskytovať, tým lepší prehľad získava podnik o tom, aké zmeny si podnik môže dovoliť vo svojich produktoch a ako na to s najväčšou pravdepodobnosťou budú reagovať jeho zákazníci. Schopnosť predpovedať trendy v správaní zákazníkov môže byť silný nástroj v boji s konkurenciou a poskytovať podniku určitú konkurenčnú výhodu.
  2. *Vynikajúce porozumenie zákazníkom* – každá značka má svoj príbeh, ktorý chce predať. Potreba poznať svoju zákaznícku základňu pre cielený marketing sa v silne konkurenčnom prostredí stáva dôležitejšou ako kedykoľvek predtým. Pomocou CRM a Big Data možno dôsledne zanalyzovať rozličné segmenty trhu, identifikovať ich aktuálne potreby, požiadavky a tiež postoj k danej značke, stanoviť vhodné cieľové skupiny zákazníkov a udržať si ich Vďaka integrácii veľdát s CRM môžu teraz spoločnosti jednoducho zistiť, ako ich produkt alebo značku vnímajú zákazníci online. Môžu použiť údaje na identifikáciu svojich slabých stránok a revíziu svojich marketingových a predajných stratégií.
  3. *Zvyšovanie výnosov* – Big Data poskytujú spoločnostiam presné metriky výkonnosti, ktoré im pomáhajú prijímať informovanejšie rozhodnutia. Predajné tímy môžu získavať potenciálnych zákazníkov a uzatvárať obchody rýchlejšie, pretože poznajú svoje cieľové publikum. Navyše, pracovníci zákazníckeho servisu majú pri obsluhe zákazníkov viac informácií.
- *Integrácia so sociálnymi sieťami* – sociálne siete taktiež ovplyvňujú trendy CRM. Povýšili totiž možnosť riadenia vzťahov so zákazníkmi na ešte vyššiu úroveň. Informácie získané z Facebooku, Twitteru či Instagramu umožňujú ešte lepšie pochopiť správanie zákazníkov. S rastúcim významom sociálnych sietí bude rovnako rásť aj sociálne CRM. Pomocou CRM systémov ako je Vtiger sa dajú vytvárať správy pre všetky komunikačné kanály naraz (Google, Facebook, Twitter, Instagram, ...) a súčasne

ich aj odosielať na všetky kanály. Nie je teda potrebné vytvárať správu pre každý kanál zvlášť a zvlášť ju odosielať. Aplikácia tiež umožňuje zbieranie údajov o hodnotení jednotlivých produktov zo všetkých takýchto kanálov a tieto hodnotenia agregovať a súhrnne vyhodnocovať. Pracovníci marketingu vďaka tomu nemusia sledovať množstvo rôznych kanálov, pretože majú všetko zjednotené na jednom mieste (VTiger, 2024).

4 Záver

V článku sme zhodnotili možnosti uplatnenia CRM systémov v podnikaní a ich potenciálne prínosy. Objasnili sme dátovú, analytickú a komunikačnú funkcionalitu týchto aplikácií a ich rozličné typy so špecifickým zameraním. Medzi aktuálne trendy v oblast CRM systémov môžeme zaradiť najmä:

- Podporu mobilných zariadení
- Implementáciu umelej inteligencie a strojového učenia
- Prechod na cloudové riešenia
- Prehľadné spracovanie údajov a schopnosť spracovávať veľké kvantá dát
- Integráciu so sociálnymi sieťami

Záverom môžeme konštatovať, že CRM systémy môžu pre podniky predstavovať výraznú pomoc z hľadiska ich lepšieho pochopenia svojich zákazníkov, predpovedania trendov zákazníckeho správania sa a budovania dlhotrvajúcich vzťahov so zákazníkmi s cieľom udržať si ich tak, aby sa vždy radi vracali k danej značke.

Literatúra

1. ManagamentMania.com. (2015, July 24). *Segmentácia trhu*. https://managementmania.com/sk/segmentacia-trhu.
2. Lendel, V., & Kubina, M. (2010). New Trends in Customer Relationship Management and their Application in Slovak Enterprises. Trends Economics and Management, 4(6), 19-26.
3. Rolustech. (2017, March 16). *CRM & Big Data Analytics*. Rolustech.com. https://www.rolustech.com/blog/crm-big-data-analytics.
4. Scheiner, M. (2024, February 5). *17 CRM Statistics: Growth, Revenue & Adoption Trends in 2024*. CRM.org. https://crm.org/crmland/crm-statistics.
5. Vaughan, A. (2024, February 12). *The Best AI CRM for 2024*. TechnologyAdvice. https://technologyadvice.com/blog/information-technology/ai-crm/.
6. VTiger. (2024). *Získajte viac zo sociálnych médií za menej*. VTiger.com. https://www.vtiger.com/sk/features/social-module/.
7. Wahlberg, O., Strandberg, C., Sundberg, H., & Sandberg, K. W. (2009). Trends, topics and under-researched areas in CRM research: a literature review. International Journal of Public information systems, 3, 191-208.
8. Wang, L. (2016). The new trend and application of customer relationship management under big data background. Modern Economy, 7(8), 841-848.

Exekučná efektívnosť použitia v jazyku integrovaných dopytov v aplikácii vytvorenej v jazyku C#

Execution Efficiency of the Use of Language-Integrated Queries in a C# Application

Igor Košťál[1]

Abstrakt
V mnohých aplikáciách musí programátor implementovať rôzne vyhľadávacie a usporiadavacie algoritmy. Častokrát tieto algoritmy vykonávajú zložitejšie vyhľadávanie a usporiadavanie, napr. vyhľadávajú študentov v ich zozname podľa rozsahu bodov na ubytovanie, ktoré dosiahli a v týchto skupinách ich usporiadavajú podľa týchto bodov alebo podľa priezviska. Takéto vyhľadávanie a usporiadavanie je možné vykonať pomocou usporiadavacích a vyhľadávacích metód aplikácie, bez použitia v jazyku integrovaných dopytov, alebo pomocou týchto dopytov. Zaujímalo nás, ktorý z týchto spôsobov vyhľadávania a usporiadavania je exekučne efektívnejší a vhodnejší pre uvedené vyhľadávanie a usporiadavanie najmä vo väčších, napr. v 2000-položkových súboroch dát študentov, v aplikácii vytvorenej v jazyku C#. Výsledky experimentu, v ktorom skúmame exekučnú efektívnosť usporiadavania a skupinového vyhľadávania pomocou vyhľadávacích a usporiadavacích metód a zdrojového kódu aplikácie, bez použitia v jazyku integrovaných dopytov, a pomocou týchto dopytov, sú uvedené v článku.

Kľúčové slová
v jazyku integrovaný dopyt, skupinové vyhľadávanie dát, usporiadavanie dát, programovací jazyk C#

Abstract
In many applications, the programmer has to implement various searching and sorting algorithms. These algorithms often perform more complex searching and sorting, e.g. they search for students in their list according to the range of points for an accommodation that they have achieved. In these groups, these algorithms sort students according to these points or the last name. Such searching and sorting can be done using the sorting and searching methods of the application, without using language-integrated queries, or by using these queries. We were interested in, which of these ways of searching and sorting is more execution efficient and more suitable for mentioned searching and sorting, especially in bigger data sets, e.g. in 2000-item students datasets, in an application created in C#. The results of the experiment, in which we examine the execution efficiency of sorting and group search using searching and sorting methods and the source code of the application without using language-integrated queries, and by using these queries, are presented in the paper.

Key words
language-integrated query, group searching for data, sorting data, the C# programming language

JEL classification
C88

[1] Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1/b, 852 35 Bratislava, igor.kostal@euba.sk

1 Úvod

Ako sme uviedli vyššie, programátor musí v aplikáciách častokrát implementovať algoritmy, ktoré vykonávajú zložitejšie vyhľadávanie a usporiadavanie, napr. vyhľadávajú študentov v ich zozname podľa rozsahu bodov na ubytovanie, ktoré dosiahli, tzv. intervalové vyhľadávanie, a v týchto skupinách ich usporiadavajú podľa týchto bodov alebo podľa priezviska. Programátor má vždy záujem implementovať uvedené algoritmy čo najefektívnejšie, čiže exekučne alebo pamäťovo najefektívnejšie. Uvedené vyhľadávanie a usporiadavanie je možné vykonať pomocou usporiadavacích a vyhľadávacích metód a zdrojového kódu aplikácie, bez použitia v jazyku C# integrovaných dopytov, alebo pomocou týchto dopytov. Zaujímalo nás, ktorý z týchto spôsobov vyhľadávania a usporiadavania je exekučne efektívnejší a vhodnejší pre uvedené vyhľadávanie a usporiadavanie najmä vo väčších, napr. v 2000-položkových súboroch dát študentov, v aplikácii vytvorenej v jazyku C#. Predpokladáme, že vyhľadávanie študentov podľa intervalov ich bodov na ubytovanie pomocou v jazyku C# integrovaného skupinového dopytu je exekučne efektívnejšie ako vykonanie takéhoto vyhľadávania pomocou zdrojového kódu, ktorý nepoužíva v jazyku C# integrovaný skupinový dopyt. Za účelom overenia tejto hypotézy sme pomocou našej C# aplikácie, ktorá dokáže vyhľadávať študentov podľa intervalov ich bodov na ubytovanie pomocou v jazyku C# integrovaného skupinového dopytu a pomocou zdrojového kódu, ktorý nepoužíva v jazyku C# integrovaný skupinový dopyt a zároveň dokáže zmerať exekučné časy, vykonali experiment. Vyhodnotenie experimentu potvrdí alebo vyvráti túto našu hypotézu.

V nasledujúcich kapitolách sa krátko zaoberáme v jazyku C# integrovanými dopytmi, triednou architektúrou a kľúčovými časťami zdrojového kódu našej C# .NET aplikácie a vyššie spomenutým experimentom.

2 V jazyku C# integrovaný dopyt, dopytovacie výrazy

V jazyku integrovaný dopyt (anglicky: Language-Integrated Query (LINQ)) je názov alebo pomenovanie pre sadu technológií integrujúcich dopytovacie schopnosti priamo do C# jazyka. Dopyty sa zapisujú ako jednoduché reťazce bez uvádzania dátových typov. Dopytovacie výrazy sa zapisujú v deklaratívnej *dopytovacej syntaxe*. Pomocou dopytovacej syntaxe môžeme vykonať filtrovacie, usporiadavacie, spájacie, zoskupovacie a projektovacie operácie na dátových zdrojoch s minimálnym množstvom zdrojového kódu. Pomocou rovnakého dopytovacieho výrazu môžeme vykonať dopyt a transformáciu dát z nejakého dátového typu dátového zdroja na iný dátový typ. (Microsoft, 2023)

Vlastnosti dopytovacích výrazov (Microsoft, 2023):

- dopytovacie výrazy vykonávajú dopyty a transformujú dáta z nejakého pre LINQ známeho dátového zdroja na iný dátový typ. Napr. jeden dopyt môže získať dáta z SQL databázy a vyprodukovať XML dáta ako výstup.
- dopytovacie výrazy používajú veľa známych syntaktických konštrukcií jazyka C#, ktoré ich robia ľahšie čitateľnými.
- premenné v dopytovacích výrazoch sú všetky prísne typové.
- dopyt nie je vykonaný, pokiaľ neiterujeme cez dopytovú premennú, napr. vo *foreach* príkaze.
- kompilátor jazyka C# konvertuje dopytovacie výrazy na volania štandardných dopytovacích operátorových metód podľa pravidiel definovaných v špecifikácii jazyka C#. Dopyt môže byť vyjadrený pomocou dopytovacej syntaxe alebo môže byť vyjadrený pomocou metódovej syntaxe. V niektorých prípadoch je dopytovacia syntax čitateľnejšia, výstižnejšia a stručnejšia, v iných prípadoch je takouto

metódová syntax. Medzi týmito dvomi formami zápisu dopytu nie je žiaden sémantický alebo vykonávací rozdiel.

- niektoré dopytovacie operácie, napr. *Count* alebo *Max*, nemajú ekvivalentnú dopytovaciu klauzulu, preto musia byť vyjadrené vo forme volania metódy. Metódová syntax môže byť kombinovaná s dopytovacou syntaxou rôznym spôsobom.
- dopytovací výraz môže byť kompilovaný do výrazového stromu alebo do delegáta, závisí to od toho, na aký dátový typ bol dopyt aplikovaný. Dopyty na dátový typ *IEnumerable<T>* sú kompilované do delegátov, dopyty na dátové typy *IQueryable* a *IQueryable<T>* sú kompilované do výrazových stromov.

Všetky LINQ dopytové operácie pozostávajú z troch odlišných akcií (Microsoft, 2024a):

- získanie dátového zdroja,
- vytvorenie dopytu a
- vykonanie dopytu, napr. vo *foreach* príkaze cyklu.

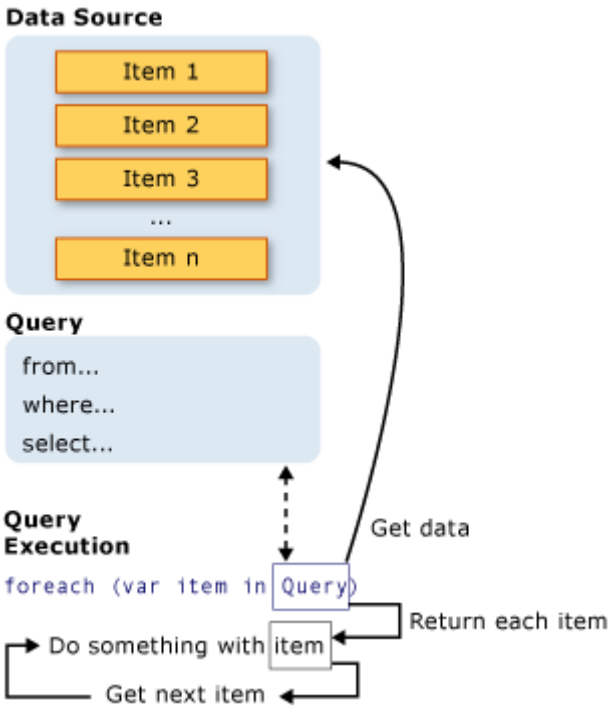
Príklad zdrojového kódu dopytovej operácie, ktorá získa dátový zdroj, zoznam študentov uložený v objekte *students* triedy *List*, vytvorí dopyt, v ktorom zoskupí a usporiada študentov podľa bodov na ubytovanie a vo *foreach* príkaze vykoná dopyt, zobrazí a zapíše takto získaných a usporiadaných študentov na konzolu a do diskového súboru, je na obr. 1.

Obr. 1: Zdrojový kód ukážkového dopytu

```
List<Student> students = new List<Student>();  // datovy zdroj, objekt 'students' triedy 'List'
List<Student> studentsSorted = new List<Student>();  // zoznam s buducimi usporiadanymi studentmi

var groupByPointsToAccomodationQuery =     // vytvorenie dopytu
        from student in students
        group student by student.Points_acc into newGroupAcc
        orderby newGroupAcc.Key descending
        select newGroupAcc;

foreach (var nameGroupAcc in groupByPointsToAccomodationQuery)  // vykonanie dopytu
{
        foreach (var item in nameGroupAcc)  {
            // vkladanie usporiadanych objektov s datami studentov podla bodov na ubytovanie
            // do noveho zoznamu, objektu 'studentsSorted' triedy 'List'
            studentsSorted.Add(item);

            string year_str = "";
            if (item.Aver_grade.ToString().Count() < 4)
                year_str = " " + item.Birthday.y.ToString();
            else
                year_str = item.Birthday.y.ToString();
                                                    . . .

            k++;
        }
}
```

Zdroj: Vlastné spracovanie

Obr. 2: Fungovanie LINQ dopytu



Zdroj: Microsoft, 2024a

Dopyt je sada inštrukcií, ktorá opisuje aké dáta chceme získať z daného dátového zdroja, alebo zdrojov, a aký tvar a organizáciu by tieto vrátené dáta mali mať (Microsoft, 2024b). Pre rôzne dátové zdroje sa používajú rozdielne natívne dopytovacie jazyky, napr. SQL sa používa pre relačné databázy, XQuery pre XML. Vývojári sa musia naučiť nový dopytovací jazyk pre každý typ dátového zdroja alebo dátového formátu. LINQ zjednodušuje túto situáciu ponúkajúc konzistentný C# jazykový model pre rôzne druhy dátových zdrojov a formátov. V LINQ dopyte vždy pracujeme so C# objektmi. Používame rovnaké základné kódové vzory pre dopytovanie a transformovanie dát v XML dokumentoch, SQL databázach, v .NET kolekciách a iných formátoch, pre ktoré je LINQ poskytovateľ prístupný. (Microsoft, 2024a)

Dopytovací výraz je dopyt vyjadrený v dopytovacej syntaxe. Kompilátorom jazyka C# je chápaný rovnako ako iný výraz a môže byť použitý v ľubovoľnom kontexte, v ktorom je možné použiť C# výraz. Dopytovací výraz sa skladá zo sady klauzúl napísaných v deklaratívnej syntaxe. Každá klauzula obsahuje jeden alebo viacero C# výrazov a tieto výrazy samotné môžu byť dopytovacími výrazmi alebo môžu obsahovať dopytovacie výrazy.

Dopytovací výraz musí začínať s klauzulou *from*, ktorá špecifikuje dátový zdroj a musí končiť klauzulou *select* alebo *group*, ktoré špecifikujú dátový typ vrátených elementov. Medzi prvou klauzulou *from* a poslednou klauzulou *select* alebo *group* môže dopytovací výraz obsahovať jednu alebo viacero voliteľných klauzúl *where*, *orderby*, *join*, *let* a dokonca aj ďalšie *from* klauzuly. Môžeme tiež použiť kľúčové slovo *into* pre získanie výsledkov z klauzúl *join* alebo *group*, ktoré slúžia ako zdroj pre viaceré dopytovacie klauzuly v rovnakom dopytovacom výraze. (Microsoft, 2024b)

Na obdržanom dátovom zdroji môžeme v dopytovacom výraze vykonať rôzne operácie (Microsoft, 2024c):

- filtrovanie dát pomocou klauzuly *where*,
- usporiadavanie dát pomocou klauzuly *orderby* a voliteľného kľúčového slova *descending*,

- zoskupovanie dát pomocou klauzuly *group* a voliteľného kľúčového slova *into*,
- spájanie dát pomocou klauzuly *join*,
- projektovanie (premietanie) dát pomocou klauzuly *select*.

Väčšinu z uvedených operácií a klauzúl sme použili v dopytovacích výrazoch v zdrojovom kóde našej C# .NET aplikácie.

3 C# .NET aplikácia hľadajúca študentov v ich zozname podľa rozsahu bodov na ubytovanie bez použitia a s použitím v jazyku integrovaných dopytov

Naša konzolová C# .NET aplikácia bola vytvorená v programovacom jazyku C# vo vývojom prostredí Microsoft Visual Studio Enterprise 2019. Aplikácia dokáže podľa požiadavky používateľa načítať dáta študentov z vybraného diskového súboru, tvoriaceho dátový vstup aplikácie, do zoznamu objektov. S týmto zoznamom študentov, s objektom knižničnej triedy *List*, pracujú metódy aplikácie. Pomocou svojho zdrojového kódu a svojich metód aplikácia dokáže:

- zobraziť neusporiadaný zoznam všetkých študentov na konzolu,
- vykonať základné usporiadanie študentov podľa používateľom vybraného kritéria, čiže podľa priezvisk alebo podľa bodov na ubytovanie študentov,
- ďalej aplikácia dokáže pomocou v jazyku integrovaného skupinového dopytu vyhľadať študentov podľa intervalov ich bodov na ubytovanie,
- pomocou svojho zdrojového kódu, bez použitia v jazyku integrovaného skupinového dopytu, vyhľadať študentov podľa intervalov ich bodov na ubytovanie,
- zmerať exekučné časy všetkých vyhľadávaní a ich výsledky spolu s exekučnými časmi jednotlivých vyhľadávaní zobraziť na konzolu (obr. 11 a 12) a zároveň ich zapísať do logovacieho súboru *LogFile.txt*.

C# .NET aplikácia obsahuje 3 triedy s nasledovnými členmi (uvedené sú len dôležité členy tried):

- *Date* - obsahuje celočíselné členy *d*, *m*, *y*. Objekty tejto triedy, vytvorené ako vnorené v triede *Student*, slúžia na ukladanie dňa, mesiaca a roku narodenia študenta.
- *Student* - obsahuje členské metódy pre načítanie a zmenu inštančných premenných obsahujúcich meno, priezvisko, bydlisko, body na ubytovanie, príjem, vzdialenosť bydliska od univerzity, priemernú známku a číslo ISIC karty študenta a tiež vnorený objekt *Birthday* triedy *Date*, ktorý obsahuje deň, mesiac a rok narodenia študenta, ktorého dáta sú uložené v objekte triedy *Student*. Ďalej táto trieda obsahuje nasledujúce kľúčové členské metódy:
  - *QuickS_movingSort* - verejná statická rekurzívna členská metóda (obr. 4), ktorá usporiada objekty zoznamu *students* (objekt triedy *List*) obsahujúce dáta študentov pomocou implementovaného algoritmu Quick Sort. Táto metóda používa presúvacie usporiadavanie, tzn. počas usporiadavania presúva objekty zoznamu *students*.
  - *Compare_sts* - privátna statická členská metóda (obr. 3), ktorá porovnáva dvoch študentov *a* a *b* podľa kritéria *criterX*. Táto metóda dokáže porovnať týchto dvoch študentov podľa priezvisk (*criterX == 41*), bodov na ubytovanie (*criterX == 1*), podľa vzdialeností ich bydlísk od univerzity (*criterX == 2*), ich príjmov (*criterX == 3*), priemerných známok (*criterX == 4*), ich dátumov narodenia a podľa ich čísiel ISIC kariet. Metóda *Compare_sts* je volaná metódou *QuickS_movingSort*.

Obr. 3: Časť zdrojového kódu členskej metódy 'Compare_sts' triedy 'Student'

```
private static int Compare_sts(Student a, Student b, int criterX)  {
  int porovnanie;

  if (criterX == 41)  // studenti sa budu usporiadavat najskor podla PRIEZVISKA,
  { // najskor sa porovnaju podla priezviska, ak maju aj priezvisko rovnake,
    porovnanie = String.Compare(a.LastName.ToLower(), b.LastName.ToLower());
    if (porovnanie < 0)   return 1;  else if (porovnanie > 0)   return -1;
    // tak potom sa porovnaju podla mena, ak maju aj meno rovnake
    porovnanie = String.Compare(a.FirstName.ToLower(), b.FirstName.ToLower());
    if (porovnanie < 0)   return 1;  else if (porovnanie > 0)   return -1;

    // tak potom sa porovnaju podla bodov na ubytovanie od NAJVACSIEHO poctu
    // bodov po najmensi, ak maju body na ubytovanie rovnake,
    if (a.Points_acc > b.Points_acc) return 1;  if (a.Points_acc < b.Points_acc) return -1;

    // tak potom sa porovnaju podla ich priemernej znamky za doterajsie studium od NAJMENSEJ po
    // najvacsiu priemernu znamku, ak maju aj priemernu znamku rovnaku,
    if (a.Aver_grade < b.Aver_grade) return 1;  if (a.Aver_grade > b.Aver_grade) return -1;

    // tak potom podla prijmu studenta od NAJMENSIEHO prijmu po najvacsi, ak maju aj prijem rovnaky,
    if (a.Income < b.Income) return 1;
    if (a.Income > b.Income) return -1;

    // tak potom podla vzdialenosti ich bydliska od skoly od NAJVACSEJ vzdialenosti po najmensiu
    if (a.Distance_resid > b.Distance_resid) return 1;
    if (a.Distance_resid < b.Distance_resid) return -1;
  }
  if (criterX == 1)  {  // studenti sa budu usporiadavat najskor podla BODOV na UBYTOVANIE,
    // cize najskor sa porovnaju podla bodov na ubytovanie od NAJVACSIEHO poctu bodov po
    // najmensi, ak maju body na ubytovanie rovnake,
    if (a.Points_acc > b.Points_acc) return 1;
    if (a.Points_acc < b.Points_acc) return -1;

    // tak potom sa porovnaju podla priezviska, ak maju aj priezvisko rovnake,
    porovnanie = String.Compare(a.LastName.ToLower(), b.LastName.ToLower());
    if (porovnanie < 0)   return 1;  else if (porovnanie > 0)   return -1;

    // tak potom sa porovnaju podla mena, ak maju aj meno rovnake;
    porovnanie = String.Compare(a.FirstName.ToLower(), b.FirstName.ToLower());
    if (porovnanie < 0)   return 1;  else if (porovnanie > 0)   return -1;

    // tak potom podla ich priemernej znamky za doterajsie studium od NAJMENSEJ po najvacsiu
    // priemernu znamku, ak maju aj priemernu znamku rovnaku,
    if (a.Aver_grade < b.Aver_grade) return 1;
    if (a.Aver_grade > b.Aver_grade) return -1;

    // tak potom podla prijmu studenta od NAJMENSIEHO prijmu po najvacsi, ak maju aj prijem rovnaky,
    if (a.Income < b.Income) return 1;
    if (a.Income > b.Income) return -1;

    // tak potom podla vzdialenosti ich bydliska od skoly od NAJVACSEJ vzdialenosti po najmensiu
    if (a.Distance_resid > b.Distance_resid) return 1;
    if (a.Distance_resid < b.Distance_resid) return -1;
  }
          . . .
}
```

Zdroj: Vlastné spracovanie

Obr. 4: Zdrojový kód členskej metódy 'QuickS_movingSort' triedy 'Student'

```
public static void QuickS_movingSort(List<Student> data, int left, int right, int criterx)
{
  if (left < right)
  {
        int i = left, j = right;
        Student p = data[(left + right) / 2];
        do
        {
            while (Compare_sts(data[i], p, criterx) > 0) i++;
            while (Compare_sts(p, data[j], criterx) > 0) j--;

            if (i >= j)
              break;
            Student t = new Student();
            t = data[i];
            data[i] = data[j];
            data[j] = t;
        } while (true);
        QuickS_movingSort(data, left, j, criterx);
        QuickS_movingSort(data, j + 1, right, criterx);
  }
}
```

Zdroj: Vlastné spracovanie

- *GetPercentile* - verejná statická členská metóda (obr. 5), ktorá vráti vypočítaný percentil študenta *s* z jeho bodov na ubytovanie. Napr., ak má študent 51, 52, 53, ... , 59 bodov na ubytovanie, tak táto metóda vypočíta a vráti percentil 5. Túto metódu volajú v jazyku integrované skupinové dopyty v statickej metóde *Main* triedy *Program*.

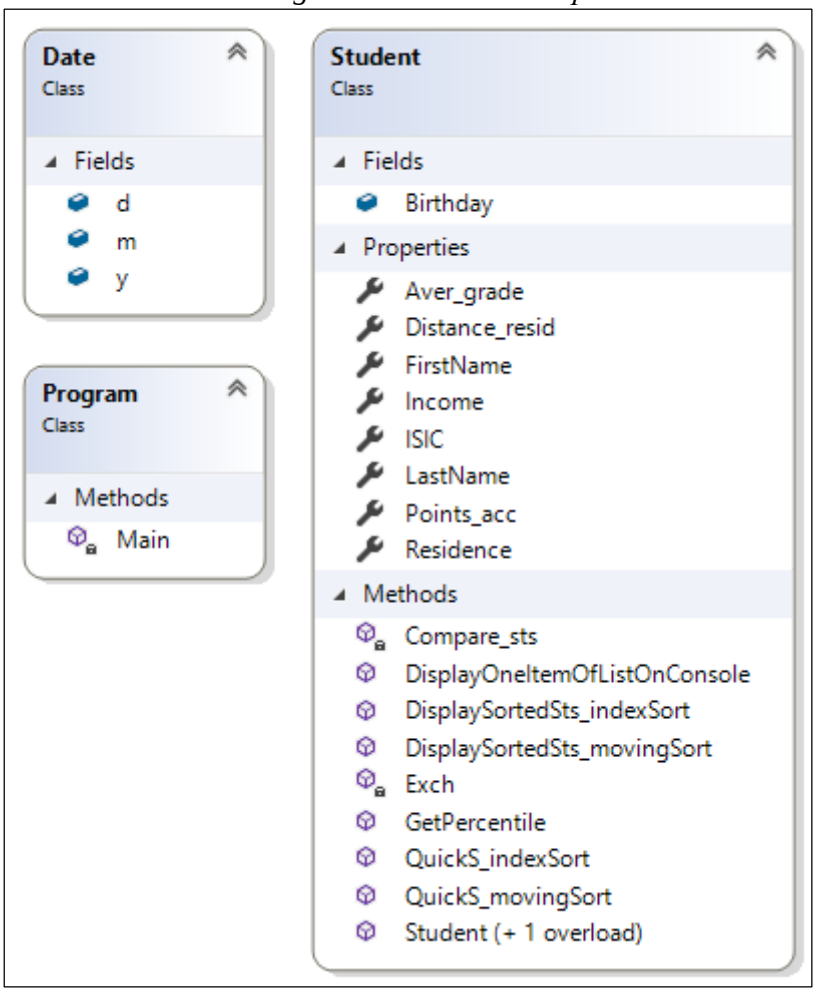
Obr. 5: Zdrojový kód členskej metódy 'GetPercentile' triedy 'Student'

```
public static int GetPercentile(Student s)
{
    return s.Points_acc > 0 ? (int)s.Points_acc / 10 : 0;
}
```

Zdroj: Vlastné spracovanie

Okrem týchto kľúčových metód trieda *Student* obsahuje dva konštruktory, jeden bezparametrický a druhý parametrický, verejnú statickú rekurzívnu členskú metódu *QuickS_indexSort*, ktorá usporiada objekty zoznamu *students* (objekt triedy *List*) obsahujúce dáta študentov pomocou implementovaného algoritmu Quick Sort indexovým usporiadavaním, bez ich premiestňovania v tomto zozname, privátnu statickú členskú metódu *Exch*, ktorá vymieňa len indexy v poli indexov bez premiestňovania objektov zoznamu *data* a ktorá je volaná členskou metódou *QuickS_indexSort*, ďalej verejnú statickú členskú metódu *DisplaySortedSts_ indexSort*, ktorá poskladá a vráti reťazec s dátami usporiadaných študentov členskou metódou *QuickS_indexSort*, verejnú statickú členskú metódu *DisplaySortedSts_movingSort*, ktorá poskladá, zobrazí na konzolu a vráti reťazec s dátami usporiadaných študentov členskou metódou *QuickS_movingSort* a verejnú statickú členskú metódu *DisplayOneItemOfListOnConsole*, ktorá poskladá, zobrazí na konzolu a vráti reťazec s dátami študenta uloženého v objekte *itemX* triedy *Student*.

Obr. 6: Diagram tried C# .NET aplikácie



Zdroj: Vlastné spracovanie

- *Program* - v jej statickej metóde *Main* sú vytvorené dva objekty *students* a *studentsSorted* triedy *List*. Po používateľovom výbere vstupného diskového súboru tvoriaceho dátový vstup aplikácie, je v zdrojovom kóde metódy *Main* vytvorený dátový prúd *sr*, ktorý je objektom triedy *StreamReader*, a do tohto dátového prúdu spojeného s vybratým vstupným diskovým súborom sú pomocou konštruktora triedy *StreamReader* načítané všetky dáta z pripojeného ASCII diskového súboru. V ďalšom zdrojovom kóde metódy *Main* sú po jednotlivých riadkoch čítané dáta z dátového prúdu *sr*, čo sú vždy dáta jedného študenta a tieto dáta sú zapísané do inštančných premenných objektu triedy *Student*, ktorý je následne pridaný na koniec zoznamu *students*. Ďalej metóda *Main* vypíše dáta študentov z neusporiadaného zoznamu *students* na konzolu, následne usporiada tento neusporiadaný zoznam študentov podľa ich priezvisk pomocou v jazyku integrovaného skupinového dopytu (obr. 7), usporiadaný zoznam zapíše do objektu *studentsSorted* triedy *List*, vypíše ho na konzolu (obr. 11 a 12) a zapíše ho do logovacieho súboru *LogFile.txt*.
  V ďalšom zdrojovom kóde metódy *Main* sú pomocou v jazyku integrovaného skupinového dopytu vyhľadaní študenti podľa intervalov ich bodov na ubytovanie (obr. 8), je zmeraný exekučný čas tohto vyhľadávania a následne sú títo vyhľadaní študenti vypísaní na konzolu (obr. 11 a 12) a do logovacieho súboru *LogFile.txt*.

Obr. 7: Usporiadanie študentov podľa ich priezvisk pomocou v jazyku integrovaného skupinového dopytu, ich zobrazenie na konzolu a ich zapísanie do premennej 'str_for_sw', ktorá bude zapísaná do logovacieho súboru 'LogFile.txt', v zdrojovom kóde metódy 'Main'

```
List<Student> students = new List<Student>();  // datovy zdroj, objekt 'students' triedy 'List'
List<Student> studentsSorted = new List<Student>();  // zoznam s buducimi usporiadanymi studentmi
                                                    . . .
var groupByLastNamesQuery =
from student in students
group student by student.LastName into newGroup
orderby newGroup.Key
select newGroup;
                                                    . . .
flagConsole = 0;
k = 1;
foreach (var nameGroup in groupByLastNamesQuery)
{
      foreach (var item in nameGroup)
      {   // vkladanie usporiadanych objektov s datami studentov podla 'Lastname' do zoznamu
          // 'studentsSorted'
          studentsSorted.Add(item);

          string year_str = "";
          if (item.Aver_grade.ToString().Count() < 4)
                year_str = " " + item.Birthday.y.ToString();
          else
                year_str = item.Birthday.y.ToString();
                                                    . . .
          if (k < 21)  // ak je poradove cislo studenta nacitaneho zo zoznamu studentov 'k < 21', tak ho
          {             // zobrazime na konzolu
            Console.WriteLine($"{rank_cons}  {item.ISIC} {item.LastName}{gapLastFirstNameCons}
                        {item.FirstName}\t({item.Points_acc}) " +
                        $"{item.Income} {item.Aver_grade} {year_str}-{month_str}-{day_str}\t  " +
                        $"{item.Residence}\t({item.Distance_resid})");
          }
          else if (flagConsole == 0)
          {
            Console.WriteLine($"\nThe list of students is too extensive, greater than 20 items." +
                        $"\nFull list of students is written into the 'LogFile.txt' file");
            flagConsole++;
          }

      str_for_sw += $"{rank}  {item.ISIC} {item.LastName}{gapLastFirstName_sw}{item.FirstName}" +
                  $"{gapFirstNamePoints_acc_sw}({item.Points_acc}) " +
                  $"{item.Income} {item.Aver_grade} {year_str}-{month_str}-{day_str}\t  " +
                  $"{item.Residence}\t({item.Distance_resid})" + Environment.NewLine;

          k++;
       }
}
```

Zdroj: Vlastné spracovanie

V ďalšom zdrojovom kóde metódy *Main* sú pomocou volania statickej členskej metódy *QuickS_movingSort* študenti uložení v objektoch triedy *Student*, ktoré sú uložené v zozname *students*, usporiadaní podľa ich priezvisk. Táto metóda používa presúvacie usporiadavanie, tzn. počas usporiadavania presúva objekty v zozname *students*. Následne sú títo usporiadaní študenti pomocou volania statickej členskej metódy *DisplaySortedSts_movingSort* vypísaní na konzolu (obr. 11 a 12). Ďalší zdrojový kód metódy *Main* vyhľadá študentov podľa intervalov ich bodov na

ubytovanie, zmeria exekučný čas tohto vyhľadávania a vyhľadaných študentov spolu so zmeraným exekučným časom vypíše na konzolu (obr. 9) a do logovacieho súboru *LogFile.txt*. Tento zdrojový kód nepoužíva v jazyku integrovaný skupinový dopyt.

Obr. 8: Vyhľadanie študentov podľa intervalov ich bodov na ubytovanie pomocou v jazyku integrovaného skupinového dopytu, ich zobrazenie na konzolu a ich zapísanie do premennej 'str_for_sw', ktorá bude zapísaná do logovacieho súboru 'LogFile.txt', v zdrojovom kóde metódy 'Main'

```
Stopwatch stopWatch = new Stopwatch();
stopWatch.Reset(); // zastavi meranie casu a resetuje uplynuty cas na 0
stopWatch.Start(); // spusti alebo obnovi meranie uplynuteho casu

var groupByPercentileQuery =
from student in studentsSorted
let percentile = Student.GetPercentile(student) // do skupiny vyberame studentov, ktori maju percentil
group new                    // napr. 5, napr. studenti s bodmi na ubytovanie 51, 53, 57 maju percentil 5
{
  student.FirstName,
  student.LastName,
  student.Points_acc,
  student.Income,
  student.Aver_grade,
  student.Birthday,
  student.Residence,
  student.Distance_resid
} by percentile into percentGroup
orderby percentGroup.Key
select percentGroup;

stopWatch.Stop(); // zastavi meranie uplynuteho casu v intervale
TimeSpan ts_IS = stopWatch.Elapsed;  // ziska uplynuty casa ako hodnotu 'TimeSpan'.
string strIntervalStopWatch_IS = ts_IS.ToString();
double msIntervalStopWatch_IS = ts_IS.TotalMilliseconds;
                                                    . . .
foreach (var studentGroup in groupByPercentileQuery) {
Console.WriteLine($"{studentGroup.Key * 10} - {studentGroup.Key * 10 + 9} [{studentGroup.Count()}]");
  str_for_sw += $"{studentGroup.Key * 10} - {studentGroup.Key * 10 + 9} [{studentGroup.Count()}]\n";
  flagConsole = 0;  k = 1;
  foreach (var item in studentGroup) {
      string year_str = "";
      if (item.Aver_grade.ToString().Count() < 4)  year_str = " " + item.Birthday.y.ToString();
      else  year_str = item.Birthday.y.ToString();
                                                    . . .
      else {   // ak je poradove cislo studenta nacitaneho zo zoznamu studentov 'k < 21', tak ho
        if (k < 21)  // zobrazime na konzolu
          Console.WriteLine($"\t{item.LastName}\t     {item.FirstName}\t({item.Points_acc})" +
                            $" {item.Income}  {item.Aver_grade}  {year_str}-{month_str}-{day_str}\t  " +
                            $"{item.Residence}\t({item.Distance_resid})");
        else if (flagConsole == 0)  {
          Console.WriteLine($"\nThe list of students is too extensive, greater than 20 items.\n" +
                            $"Full list of students is written into the 'LogFile.txt' file.\n");   flagConsole++;  }

        str_for_sw += $"\t{item.LastName}\t     {item.FirstName}\t({item.Points_acc})" +
                            $" {item.Income}  {item.Aver_grade}  {year_str}-{month_str}-{day_str}\t  " +
                            $"{item.Residence}\t({item.Distance_resid})\n";  }
      k++;  } }
```

Zdroj: Vlastné spracovanie

Obr. 9: Vyhľadanie študentov podľa intervalov ich bodov na ubytovanie bez použitia v jazyku integrovaného skupinového dopytu, ich zobrazenie na konzolu a ich zapísanie do premennej 'str_for_sw', ktorá bude zapísaná do logovacieho súboru 'LogFile.txt', v zdrojovom kóde metódy 'Main'

```
stopWatch.Reset(); // zastavi meranie casu a resetuje uplynuty cas na 0
stopWatch.Start(); // spusti alebo obnovi meranie uplynuteho casu

int flagConsWITHOUT_LINQ = 0;
Console.WriteLine($"{30} - {39}");
str_for_sw += $"\n{30} - {39}\n";
// hladanie studentov v skupine s bodmi na ubytovanie v rozsahu '30 - 39'
foreach (var item in students)  {
  if (item.Points_acc >= 30 && item.Points_acc < 40)  {
      str_for_sw += Student.DisplayOneItemOfListOnConsole(item, countInGroup + 1,
                                                          ref flagConsWITHOUT_LINQ);
      countInGroup++;
  }
}
if (countInGroup == 0)  {
  Console.WriteLine($"\tNONE students");
  str_for_sw += $"\tNONE students\n";
}
else  {
  Console.WriteLine($"\t[{countInGroup}]\n");
  str_for_sw += $"\t[{countInGroup}]\n";
}
                                                  . . .

flagConsWITHOUT_LINQ = 0;
countInGroup = 0;
Console.WriteLine($"{90} - {100}");
str_for_sw += $"{90} - {100}\n";
// hladanie studentov v skupine s bodmi na ubytovanie v rozsahu '90 - 100'
foreach (var item in students)  {
  if (item.Points_acc >= 90 && item.Points_acc <= 100)  {
      str_for_sw += Student.DisplayOneItemOfListOnConsole(item, countInGroup + 1,
                                                          ref flagConsWITHOUT_LINQ);
      countInGroup++;
  }
}
if (countInGroup == 0)  {
  Console.WriteLine($"\tNONE students");
  str_for_sw += $"\tNONE students\n";
}
else  {
  Console.WriteLine($"\t[{countInGroup}]\n");
  str_for_sw += $"\t[{countInGroup}]\n";
}

stopWatch.Stop(); // zastavi meranie uplynuteho casu v intervale
                  // ziska uplynuty casa ako hodnotu 'TimeSpan'
TimeSpan ts_IS1 = stopWatch.Elapsed;
string strIntervalStopWatch_IS1 = ts_IS1.ToString();
double msIntervalStopWatch_IS1 = ts_IS1.TotalMilliseconds;
```

Zdroj: Vlastné spracovanie

Nasledujúci zdrojový kód metódy *Main* vykoná podobnú činnosť, s tým rozdielom, že študentov z neusporiadaného zoznamu *students* **usporiada podľa ich bodov**

na ubytovanie pomocou v jazyku integrovaného skupinového dopytu (obr. 1), usporiadaný zoznam zapíše do objektu *studentsSorted* triedy *List* a vypíše ho na konzolu (obr. 11 a 12). Ostatná časť zdrojového kódu vykoná rovnakú činnosť, ako je uvedené v dvoch odstavcoch vyššie, s tým rozdielom, že študenti budú v ďalšom usporiadanom zozname, ktorý je usporiadaný pomocou volania statickej členskej metódy *QuickS_movingSort*, a v dvoch vyhľadávaniach podľa intervalov ich bodov na ubytovanie, prvé je vykonané pomocou v jazyku integrovaného skupinového dopytu (obr. 10), druhé je vykonané bez takéhoto dopytu (zdrojový kód pre takéto vyhľadávanie je zhodný so zdrojovým kódom na obr. 9), vždy usporiadaní podľa ich bodov na ubytovanie (obr. 11 a 12).

Výsledky všetkých vyhľadávaní spolu s exekučnými časmi jednotlivých vyhľadávaní metóda *Main* zobrazí na konzolu (obr. 11 a 12) a zároveň ich zapíše do logovacieho súboru *LogFile.txt*.

Obr. 10: Vyhľadanie študentov podľa intervalov ich bodov na ubytovanie pomocou v jazyku integrovaného skupinového dopytu, ich zobrazenie na konzolu a ich zapísanie do premennej 'str_for_sw', ktorá bude zapísaná do logovacieho súboru 'LogFile.txt', v zdrojovom kóde metódy 'Main'

```
Stopwatch stopWatch = new Stopwatch();
stopWatch.Reset(); // zastavi meranie casu a resetuje uplynuty cas na 0
stopWatch.Start(); // spusti alebo obnovi meranie uplynuteho casu

var groupByPercentilePointsToAccomQuery =
from student in studentsSorted
let percentile = Student.GetPercentile(student)  // do skupiny vyberame studentov, ktori maju percentil
group new  {                  // napr. 5, napr. studenti s bodmi na ubytovanie 51, 53, 57 maju percentil 5
  student.FirstName,
  student.LastName,
  student.Points_acc,
  student.Income,
  student.Aver_grade,
  student.Birthday,
  student.Residence,
  student.Distance_resid
} by percentile into percentGroupAcc
orderby percentGroupAcc.Key
select percentGroupAcc;
// zastavi meranie uplynuteho casu v intervale
stopWatch.Stop();                  // ziska uplynuty casa ako hodnotu 'TimeSpan'
TimeSpan ts_IS = stopWatch.Elapsed;  string strIntervalStopWatch_IS = ts_IS.ToString();
double msIntervalStopWatch_IS = ts_IS.TotalMilliseconds;
                                                    . . .
foreach (var studentGroupAcc in groupByPercentilePointsToAccomQuery)  {
  Console.WriteLine($"{studentGroupAcc.Key * 10} - {studentGroupAcc.Key * 10 + 9}
                        [{studentGroupAcc.Count()}]");
  str_for_sw += $"{studentGroupAcc.Key * 10} - {studentGroupAcc.Key * 10 + 9}
                    [{studentGroupAcc.Count()}]\n";

flagConsole = 0;  k = 1;
  foreach (var item in studentGroupAcc)  {
      string year_str = "";
      if (item.Aver_grade.ToString().Count() < 4)  year_str = " " + item.Birthday.y.ToString();
      else  year_str = item.Birthday.y.ToString();
                                                . . . } }
```

Zdroj: Vlastné spracovanie

Obr. 11: Príklad vstupu (tučným písmom) a výstupu C# .NET aplikácie, ktorá vyhľadala študentov podľa intervalov ich bodov na ubytovanie (v nich ich usporiadala podľa priezviska) pomocou v jazyku integrovaného skupinového dopytu a bez jeho použitia vo vstupnom dátovom súbore 'students10c2en.txt' s 10 študentmi

```
Choose input file: students11c2sk.txt (insert: 1), students10c2en.txt (2), students100c2en.txt (3),
students200c2en.txt (4), students300c2en.txt (5), students400c2en.txt (6), students500c2en.txt (7),
students600c2en.txt (8), students700c2en.txt (9), students800c2en.txt (10), students900c2en.txt (11),
students1000c2en.txt (12), students1100c2en.txt (13), students1200c2en.txt (14),
students1300c2en.txt (15), students1400c2en.txt (16), students1500c2en.txt (17),
students1600c2en.txt (18), students1700c2en.txt (19), students1800c2en.txt (20),
students1900c2en.txt (21), students2000c2en.txt (22)
2
                The student's data loaded from the 'students10c2en.txt' file.

    The number of the students: 10
---- Unsorted students:

 1  35988187656170751 Gagg        Hersch    (35) 483 2,27 1999-05-21 Stamford     (169)
 2  35352737044886535 Dougill     Percival  (53) 237 2,77 2000-02-10 Joliet       (383)
 3  35150528961405232 Cashen      Shayla    (57) 566 2,74 2002-03-22 Jamaica      (164)
 4  35978397704389789 Mannin      Cchaddie  (66) 340 2,05 2001-04-17 Jacksonville (32)
 5  32630895575785021 Mancer      Willey    (61) 160 2,92 2001-04-06 Augusta      (783)
 6  32322011686544786 Gerardot    Edithe    (57) 383 1,2  2001-01-22 Topeka       (703)
 7  33931482295090410 L'Anglois   Jeana     (51) 421 2,8  2000-08-06 Boynton-Beach (289)
 8  35350197848168233 Danilovitch Roxine    (72) 627 4,66 2003-03-01 Chicago      (326)
 9  33928014847747866 Gabbidon    Guilbert  (73) 381 2,05 1999-07-23 Bloomington  (572)
10  35201185262631838 Coopper     Roslyn    (62) 570 4,68 2003-11-24 Grand-Rapids (177)


Choose sorting criterion of students:
sorting according to:
- their last names (insert: 1),
- points for an accommodation (insert: 2),
1
                The students will be sorted according to their 'LAST NAMES'.

---- The students sorted according to LAST NAMES by a LINQ query:

 1  35150528961405232 Cashen      Shayla    (57) 566 2,74 2002-03-22 Jamaica      (164)
 2  35201185262631838 Coopper     Roslyn    (62) 570 4,68 2003-11-24 Grand-Rapids (177)
 3  35350197848168233 Danilovitch Roxine    (72) 627 4,66 2003-03-01 Chicago      (326)
 4  35352737044886535 Dougill     Percival  (53) 237 2,77 2000-02-10 Joliet       (383)
 5  33928014847747866 Gabbidon    Guilbert  (73) 381 2,05 1999-07-23 Bloomington  (572)
 6  35988187656170751 Gagg        Hersch    (35) 483 2,27 1999-05-21 Stamford     (169)
 7  32322011686544786 Gerardot    Edithe    (57) 383 1,2  2001-01-22 Topeka       (703)
 8  33931482295090410 L'Anglois   Jeana     (51) 421 2,8  2000-08-06 Boynton-Beach (289)
 9  32630895575785021 Mancer      Willey    (61) 160 2,92 2001-04-06 Augusta      (783)
10  35978397704389789 Mannin      Cchaddie  (66) 340 2,05 2001-04-17 Jacksonville (32)

---- The students sorted according to RANGES of POINTS for an ACCOMODATION
---- by a LINQ group query
---- (students in groups are sorted according to last names, then according to first names)
---- Execution time: 00:00:00.0007345, 0.7345 ms:

30 - 39 [1]
      Gagg        Hersch      (35) 483  2,27  1999-05-21      Stamford        (169)
```

Zdroj: Vlastné spracovanie

Obr. 11: (Pokračovanie)

```
50 - 59 [4]
      Cashen      Shayla      (57) 566  2,74  2002-03-22      Jamaica         (164)
      Dougill     Percival    (53) 237  2,77  2000-02-10      Joliet          (383)
      Gerardot    Edithe      (57) 383  1,2   2001-01-22      Topeka          (703)
      L'Anglois   Jeana       (51) 421  2,8   2000-08-06      Boynton-Beach   (289)
60 - 69 [3]
      Coopper     Roslyn      (62) 570  4,68  2003-11-24      Grand-Rapids    (177)
      Mancer      Willey      (61) 160  2,92  2001-04-06      Augusta         (783)
      Mannin      Cchaddie    (66) 340  2,05  2001-04-17      Jacksonville    (32)
70 - 79 [2]
      Danilovitch Roxine      (72) 627  4,66  2003-03-01      Chicago         (326)
      Gabbidon    Guilbert    (73) 381  2,05  1999-07-23      Bloomington     (572)


---- The students sorted according to LAST NAMES (then according to first names)
     by the 'QuickS_movingSort' static method (moving sort in Quick Sort):

 1  35150528961405232 Cashen      Shayla      (57) 566 2,74 2002-03-22 Jamaica       (164)
 2  35201185262631838 Coopper     Roslyn      (62) 570 4,68 2003-11-24 Grand-Rapids  (177)
 3  35350197848168233 Danilovitch Roxine      (72) 627 4,66 2003-03-01 Chicago       (326)
 4  35352737044886535 Dougill     Percival    (53) 237 2,77 2000-02-10 Joliet        (383)
 5  33928014847747866 Gabbidon    Guilbert    (73) 381 2,05 1999-07-23 Bloomington   (572)
 6  35988187656170751 Gagg        Hersch      (35) 483 2,27 1999-05-21 Stamford      (169)
 7  32322011686544786 Gerardot    Edithe      (57) 383 1,2  2001-01-22 Topeka        (703)
 8  33931482295090410 L'Anglois   Jeana       (51) 421 2,8  2000-08-06 Boynton-Beach (289)
 9  32630895575785021 Mancer      Willey      (61) 160 2,92 2001-04-06 Augusta       (783)
10  35978397704389789 Mannin      Cchaddie    (66) 340 2,05 2001-04-17 Jacksonville  (32)


---- The students sorted according to RANGES of POINTS for an ACCOMODATION
---- WITHOUT using a LINQ query
---- (students in groups are sorted according to last names, then according to first names):

30 - 39
      Gagg        Hersch      (35) 483  2,27  1999-05-21      Stamford        (169)
      [1]
40 - 49
      NONE students
50 - 59
      Cashen      Shayla      (57) 566  2,74  2002-03-22      Jamaica         (164)
      Dougill     Percival    (53) 237  2,77  2000-02-10      Joliet          (383)
      Gerardot    Edithe      (57) 383  1,2   2001-01-22      Topeka          (703)
      L'Anglois   Jeana       (51) 421  2,8   2000-08-06      Boynton-Beach   (289)
      [4]
60 - 69
      Coopper     Roslyn      (62) 570  4,68  2003-11-24      Grand-Rapids    (177)
      Mancer      Willey      (61) 160  2,92  2001-04-06      Augusta         (783)
      Mannin      Cchaddie    (66) 340  2,05  2001-04-17      Jacksonville    (32)
      [3]
70 - 79
      Danilovitch Roxine      (72) 627  4,66  2003-03-01      Chicago         (326)
      Gabbidon    Guilbert    (73) 381  2,05  1999-07-23      Bloomington     (572)
      [2]
80 - 89
      NONE students
90 - 100
      NONE students
---- Execution time: 00:00:00.0135332, 13.5332 ms
```

Zdroj: Vlastné spracovanie

Obr. 12: Príklad vstupu (tučným písmom) a výstupu C# .NET aplikácie, ktorá vyhľadala študentov podľa intervalov ich bodov na ubytovanie (v nich ich usporiadala podľa ich bodov na ubytovanie) pomocou v jazyku integrovaného skupinového dopytu a bez jeho použitia vo vstupnom dátovom súbore 'students10c2en.txt' s 10 študentmi

```
Choose input file: students11c2sk.txt (insert: 1), students10c2en.txt (2), students100c2en.txt (3),
students200c2en.txt (4), students300c2en.txt (5), students400c2en.txt (6), students500c2en.txt (7),
students600c2en.txt (8), students700c2en.txt (9), students800c2en.txt (10), students900c2en.txt (11),
students1000c2en.txt (12), students1100c2en.txt (13), students1200c2en.txt (14),
students1300c2en.txt (15), students1400c2en.txt (16), students1500c2en.txt (17),
students1600c2en.txt (18), students1700c2en.txt (19), students1800c2en.txt (20),
students1900c2en.txt (21), students2000c2en.txt (22)
2
                The student's data loaded from the 'students10c2en.txt' file.

    The number of the students: 10
---- Unsorted students:

 1  35988187656170751 Gagg        Hersch    (35) 483 2,27 1999-05-21 Stamford     (169)
 2  35352737044886535 Dougill     Percival  (53) 237 2,77 2000-02-10 Joliet       (383)
 3  35150528961405232 Cashen      Shayla    (57) 566 2,74 2002-03-22 Jamaica      (164)
 4  35978397704389789 Mannin      Cchaddie  (66) 340 2,05 2001-04-17 Jacksonville (32)
 5  32630895575785021 Mancer      Willey    (61) 160 2,92 2001-04-06 Augusta      (783)
 6  32322011686544786 Gerardot    Edithe    (57) 383 1,2  2001-01-22 Topeka       (703)
 7  33931482295090410 L'Anglois   Jeana     (51) 421 2,8  2000-08-06 Boynton-Beach (289)
 8  35350197848168233 Danilovitch Roxine    (72) 627 4,66 2003-03-01 Chicago      (326)
 9  33928014847747866 Gabbidon    Guilbert  (73) 381 2,05 1999-07-23 Bloomington  (572)
10  35201185262631838 Coopper     Roslyn    (62) 570 4,68 2003-11-24 Grand-Rapids (177)


Choose sorting criterion of students:
sorting according to:
- their last names (insert: 1),
- points for an accommodation (insert: 2),
2
                The students will be sorted according to 'POINTS FOR an ACCOMMODATION'.

---- The students sorted according to POINTS for an ACCOMODATION by a LINQ query:

 1  33928014847747866 Gabbidon    Guilbert  (73) 381 2,05 1999-07-23 Bloomington  (572)
 2  35350197848168233 Danilovitch Roxine    (72) 627 4,66 2003-03-01 Chicago      (326)
 3  35978397704389789 Mannin      Cchaddie  (66) 340 2,05 2001-04-17 Jacksonville (32)
 4  35201185262631838 Coopper     Roslyn    (62) 570 4,68 2003-11-24 Grand-Rapids (177)
 5  32630895575785021 Mancer      Willey    (61) 160 2,92 2001-04-06 Augusta      (783)
 6  35150528961405232 Cashen      Shayla    (57) 566 2,74 2002-03-22 Jamaica      (164)
 7  32322011686544786 Gerardot    Edithe    (57) 383 1,2  2001-01-22 Topeka       (703)
 8  35352737044886535 Dougill     Percival  (53) 237 2,77 2000-02-10 Joliet       (383)
 9  33931482295090410 L'Anglois   Jeana     (51) 421 2,8  2000-08-06 Boynton-Beach (289)
10  35988187656170751 Gagg        Hersch    (35) 483 2,27 1999-05-21 Stamford     (169)


---- The students sorted according to RANGES of POINTS for ACCOMODATION
---- by a LINQ group query
---- (students in groups are sorted according to points for an accomodation, then according to last
      names):
---- Execution time: 00:00:00.0000998, 0.0998 ms

30 - 39 [1]
      Gagg        Hersch      (35) 483  2,27  1999-05-21      Stamford        (169)
```

Zdroj: Vlastné spracovanie

Obr. 12: (Pokračovanie)

```
50 - 59 [4]
      Cashen      Shayla      (57) 566  2,74  2002-03-22    Jamaica          (164)
      Gerardot    Edithe      (57) 383  1,2   2001-01-22    Topeka           (703)
      Dougill     Percival    (53) 237  2,77  2000-02-10    Joliet           (383)
      L'Anglois   Jeana       (51) 421  2,8   2000-08-06    Boynton-Beach    (289)
60 - 69 [3]
      Mannin      Cchaddie    (66) 340  2,05  2001-04-17    Jacksonville     (32)
      Coopper     Roslyn      (62) 570  4,68  2003-11-24    Grand-Rapids     (177)
      Mancer      Willey      (61) 160  2,92  2001-04-06    Augusta          (783)
70 - 79 [2]
      Gabbidon    Guilbert    (73) 381  2,05  1999-07-23    Bloomington      (572)
      Danilovitch Roxine      (72) 627  4,66  2003-03-01    Chicago          (326)


---- The students sorted according to POINTS for an ACCOMODATION (then according to last names)
    by the 'QuickS_movingSort' static method (moving sort in Quick Sort):

 1  33928014847747866 Gabbidon   Guilbert   (73) 381 2,05 1999-07-23  Bloomington    (572)
 2  35350197848168233 Danilovitch Roxine    (72) 627 4,66 2003-03-01  Chicago        (326)
 3  35978397704389789 Mannin     Cchaddie   (66) 340 2,05 2001-04-17  Jacksonville   (32)
 4  35201185262631838 Coopper    Roslyn     (62) 570 4,68 2003-11-24  Grand-Rapids   (177)
 5  32630895575785021 Mancer     Willey     (61) 160 2,92 2001-04-06  Augusta        (783)
 6  35150528961405232 Cashen     Shayla     (57) 566 2,74 2002-03-22  Jamaica        (164)
 7  32322011686544786 Gerardot   Edithe     (57) 383 1,2  2001-01-22  Topeka         (703)
 8  35352737044886535 Dougill    Percival   (53) 237 2,77 2000-02-10  Joliet         (383)
 9  33931482295090410 L'Anglois  Jeana      (51) 421 2,8  2000-08-06  Boynton-Beach  (289)
10  35988187656170751 Gagg       Hersch     (35) 483 2,27 1999-05-21  Stamford       (169)

---- The students sorted according to RANGES of POINTS for an ACCOMODATION
---- WITHOUT using a LINQ query
---- (students in groups are sorted according to points for an accomodation, then according to last
      names):

30 - 39
      Gagg        Hersch      (35) 483  2,27  1999-05-21    Stamford         (169)
      [1]
40 - 49
      NONE students
50 - 59
      Cashen      Shayla      (57) 566  2,74  2002-03-22    Jamaica          (164)
      Gerardot    Edithe      (57) 383  1,2   2001-01-22    Topeka           (703)
      Dougill     Percival    (53) 237  2,77  2000-02-10    Joliet           (383)
      L'Anglois   Jeana       (51) 421  2,8   2000-08-06    Boynton-Beach    (289)
      [4]
60 - 69
      Mannin      Cchaddie    (66) 340  2,05  2001-04-17    Jacksonville     (32)
      Coopper     Roslyn      (62) 570  4,68  2003-11-24    Grand-Rapids     (177)
      Mancer      Willey      (61) 160  2,92  2001-04-06    Augusta          (783)
      [3]
70 - 79
      Gabbidon    Guilbert    (73) 381  2,05  1999-07-23    Bloomington      (572)
      Danilovitch Roxine      (72) 627  4,66  2003-03-01    Chicago          (326)
      [2]
80 - 89
      NONE students
90 - 100
      NONE students
---- Execution time: 00:00:00.0113859, 11.3859 ms
```

Zdroj: Vlastné spracovanie

4 Experiment, jeho výsledky, ich krátka analýza

Účelom experimentu je potvrdiť alebo vyvrátiť našu hypotézu: predpokladáme, že vyhľadávanie študentov podľa intervalov ich bodov na ubytovanie pomocou v jazyku C# integrovaného skupinového dopytu je exekučne efektívnejšie ako vykonanie takéhoto vyhľadávania pomocou zdrojového kódu, ktorý nepoužíva v jazyku C# integrovaný skupinový dopyt.

Experiment sme vykonali pomocou našej C# .NET aplikácie, ktorá na vstupe načítala vstupné dáta postupne z 20 diskových súborov: *students100c2en.txt*, *students200c2en.txt*, *students300c2en.txt*, *students400c2en.txt*, *students500c2en.txt*, *students600c2en.txt*, *students700c2en.txt*, *students800c2en.txt*, *students900c2en.txt*, *students1000c2en.txt*, *students1100c2en.txt*, *students1200c2en.txt*, *students1300c2en.txt*, *students1400c2en.txt*, *students1500c2en.txt*, *students1600c2en.txt*, *students1700c2en.txt*, *students1800c2en.txt*, *students1900c2en.txt* a *students2000c2en.txt*. Tieto vstupné súbory, ako je zrejmé z ich názvov, obsahovali dáta 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900 a 2000 študentov.

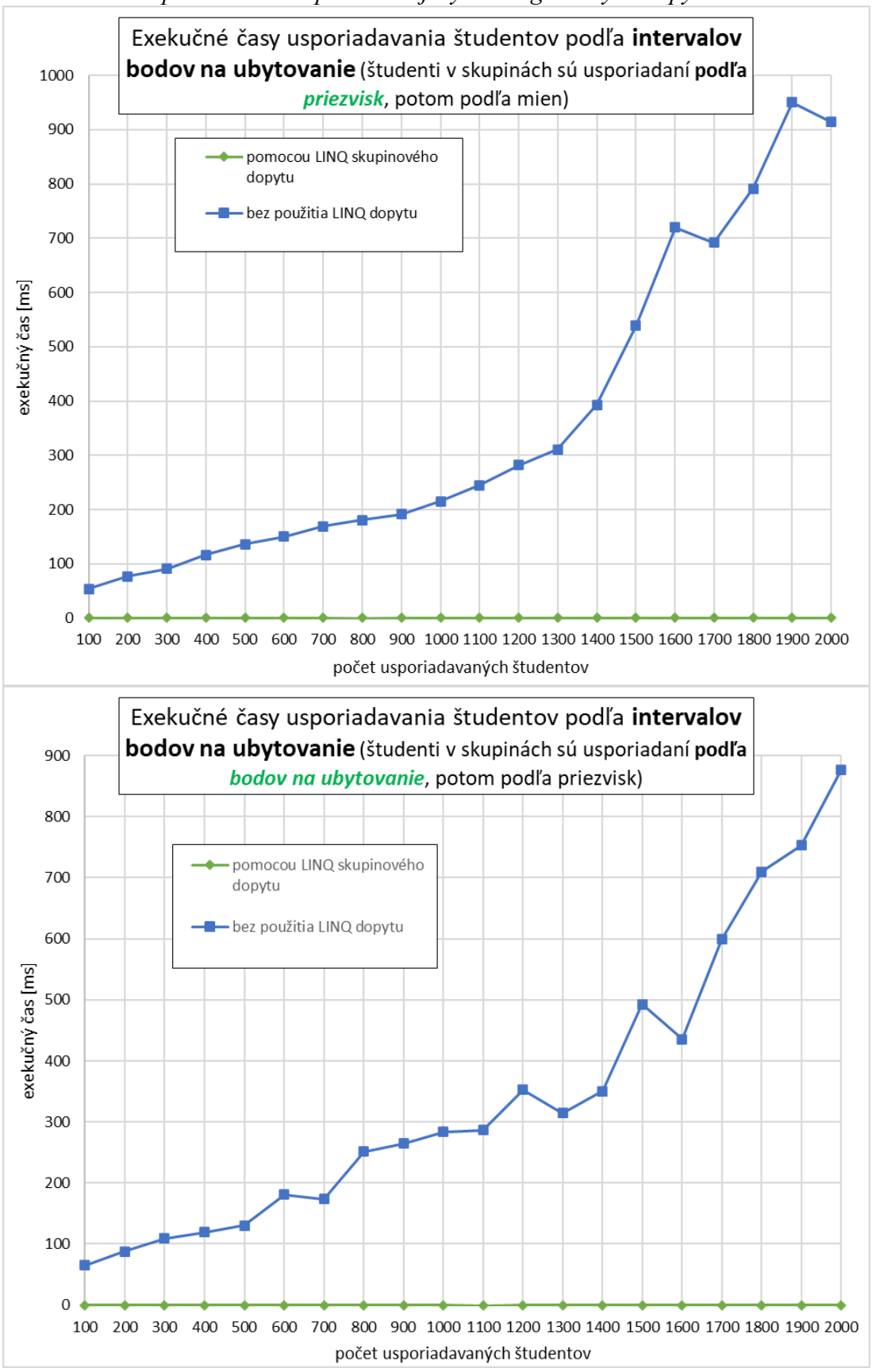
Po načítaní dát študentov z každého z uvedených vstupných diskových súborov do dátového prúdu spojeného s vybratým vstupným diskovým súborom aplikácia vykonala nasledujúce činnosti:

- zobrazila neusporiadaný zoznam všetkých študentov na konzolu,
- vykonala základné usporiadanie študentov podľa používateľom vybraného kritéria, čiže podľa priezvisk alebo podľa bodov na ubytovanie študentov,
- pomocou v jazyku integrovaného skupinového dopytu vyhľadala študentov podľa intervalov ich bodov na ubytovanie a zmerala exekučný čas tohto vyhľadávania. Študentov v jednotlivých skupinách usporiadala podľa ich priezvisk alebo podľa ich bodov na ubytovanie. Toto základné usporiadanie študentov v skupinách bolo vybraté pred vykonaním vyhľadávania používateľom.
- pomocou svojho zdrojového kódu, bez použitia v jazyku integrovaného skupinového dopytu, vyhľadala študentov podľa intervalov ich bodov na ubytovanie a zmerala exekučný čas tohto vyhľadávania. Študentov v jednotlivých skupinách usporiadala podľa ich priezvisk alebo podľa ich bodov na ubytovanie, čo bolo dané výberom používateľa pred vykonaním tohto vyhľadávania.
- výsledky všetkých vyhľadávaní spolu s exekučnými časmi jednotlivých vyhľadávaní zobrazila na konzolu (obr. 11 a 12) a zároveň ich zapísala do logovacieho súboru *LogFile.txt*.

Počas experimentu C# .NET aplikácia fungovala na počítači s nasledovnou základnou hardvérovou konfiguráciou: Intel Core i5-8250U Processor (6MB Cache, 1.60 GHz (Processor Base Frequency), 3.40 GHz (Max Turbo Frequency)), 4 GT/s (Bus Speed), 4 Cores, 8 Threads, RAM: 8 GB. Na tomto počítači bol nainštalovaný 64-bitový operačný systém Microsoft Windows 10 Home a Microsoft .NET Framework 4.

Exekučné časy všetkých uvedených vyhľadávaní sú uvedené v nasledujúcich grafoch. Exekučné časy všetkých vyhľadávaní študentov podľa intervalov ich bodov na ubytovanie pomocou v jazyku integrovaných skupinových dopytov, zobrazené v oboch grafoch, majú hodnoty menšie ako 1 ms.

Obr. 13: Exekučné časy usporiadavania študentov podľa intervalov bodov na ubytovanie s použitím a bez použitia v jazyku integrovaných dopytov



Zdroj: Vlastné spracovanie

Krátka analýza výsledkov experimentu

Z porovnania exekučných časov vyhľadávaní študentov podľa intervalov ich bodov na ubytovanie pomocou v jazyku integrovaného skupinového dopytu a pomocou zdrojového kódu, ktorý nepoužíva v jazyku integrovaný skupinový dopyt, je zrejmé, že vyhľadávanie študentov pomocou v jazyku integrovaného skupinového dopytu je exekučne násobne efektívnejšie ako realizácia takéhoto vyhľadávania bez použitia v jazyku integrovaného skupinového dopytu. Dokonca, pri vyhľadávaní študentov podľa intervalov ich bodov na ubytovanie pomocou v jazyku integrovaného skupinového dopytu v dátovej sade 2000 študentov je takéto vyhľadávanie 6223-krát rýchlejšie ako rovnaké vyhľadávanie v rovnakej dátovej sade, ale pomocou zdrojového kódu, ktorý nepoužíva v jazyku integrovaný skupinového dopyt. Takéto veľmi veľké zrýchlenie sme nepredpokladali. Uvedené skutočnosti potvrdzujú našu hypotézu: vyhľadávanie študentov podľa intervalov ich bodov na ubytovanie pomocou v jazyku integrovaného skupinového dopytu je exekučne efektívnejšie ako vykonanie takéhoto vyhľadávania pomocou zdrojového kódu, ktorý nepoužíva v jazyku integrovaný skupinový dopyt.

5 Záver

Z uvedených výsledkov experimentu je zrejmé, že vyhľadávanie študentov podľa intervalov ich bodov na ubytovanie pomocou v jazyku integrovaného skupinového dopytu je mnohonásobne, dokonca až 6223-násobne exekučne efektívnejšie ako vykonanie takéhoto vyhľadávania pomocou zdrojového kódu, ktorý nepoužíva v jazyku integrovaný skupinový dopyt. Vysokú exekučnú efektívnosť vyhľadávania študentov podľa intervalov ich bodov na ubytovanie pomocou v jazyku integrovaného skupinového dopytu zásadne neovplyvňuje základné usporiadanie týchto študentov pred samotným vyhľadávaním, tzn. táto vysoká exekučná efektívnosť takéhoto vyhľadávania študentov pomocou v jazyku integrovaného skupinového dopytu je nezávislá na prvotnom usporiadaní študentov.

Na základe výsledkov nášho experimentu môžeme jednoznačne odporučiť použitie v jazyku integrovaného skupinového dopytu pre skupinové vyhľadávanie študentov, napr. podľa intervalov ich bodov na ubytovanie.

Literatúra

1. Microsoft Corp. (2023). *Language Integrated Query (LINQ)*. https://learn.microsoft.com/sk-sk/dotnet/csharp/linq/.
2. Microsoft Corp. (2024a). *Introduction to LINQ Queries in C#*. https://learn.microsoft.com/sk-sk/dotnet/csharp/linq/get-started/introduction-to-linq-queries.
3. Microsoft Corp. (2024b). *Query expression basics*. https://learn.microsoft.com/sk-sk/dotnet/csharp/linq/get-started/query-expression-basics.
4. Microsoft Corp. (2024c). *Standard Query Operators Overview*. https://learn.microsoft.com/sk-sk/dotnet/csharp/linq/standard-query-operators/.

Stochastické programovanie a jeho využitie v ekonomike

Stochastic Programming and Its Use in Economics

Richard Martinus[1]

Abstrakt
Tento článok sa zaoberá problematikou stochastického programovania, ktoré je dôležitým nástrojom pre riešenie rozhodovacích problémov v prostrediach s neistotou a náhodnosťou. V práci bude poskytnutý všeobecný prehľad pravdepodobnostných rozdelení a metód používaných v stochastickom programovaní, s dôrazom na techniky ako L-Shaped a Dekompozícia bázy. Konkrétne bude vysvetlený prístup Monte Carlo, ktorý je široko využívaný na simuláciu náhodných procesov a odhadovanie hodnôt v stochastických problémoch. Tento prístup bude potom aplikovaný na riešenie príkladov, kde je demonštrovaná jeho efektívnosť a flexibilita. Okrem toho budú v práci predstavené príklady riešené aj inými metódami než Monte Carlo, pričom je použité normálne pravdepodobnostné rozdelenie na modelovanie neistoty a rizika v týchto príkladoch. Tento komplexný pohľad na problematiku stochastického programovania má za cieľ poskytnúť čitateľovi ucelené porozumenie rôznorodých metód a prístupov k riešeniu stochastických rozhodovacích problémov, pričom hlavným cieľom práce je ukážka aplikácií stochastického programovania v ekonomických príkladoch.

Kľúčové slová
Monte Carlo, Pravdepodobnosť, Python, Neistota

Abstract
This article deals with the issue of stochastic programming, which is an important tool for solving decision problems in environments with uncertainty and randomness. The thesis will provide a general overview of probabilistic distributions and methods used in stochastic programming, with an emphasis on techniques such as L-Shaped and Base Decomposition. Specifically, the Monte Carlo approach, which is widely used to simulate random processes and estimate values in stochastic problems, will be explained. This approach will then be applied to address examples where its effectiveness and flexibility are demonstrated. In addition, examples solved by methods other than Monte Carlo will be presented in the work, using the normal probabilistic distribution to model the uncertainty and risk in these examples. This comprehensive view of the issue of stochastic programming aims to provide the reader with a comprehensive understanding of common methods and approaches to solving stochastic decision-making problems, while the main goal of the work is to demonstrate applications of stochastic programming in economic examples.

Key words
Monte Carlo, Probability, Python, Uncertainty

JEL classification
C73, C8

[1] Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra operačného výskumu a ekonometrie, Dolnozemská cesta 1, 852 35 Bratislava, richard.misek@euba.sk.

1 Úvod

Stochastické programovanie je prístup na modelovanie optimalizačných problémov, ktoré zahŕňajú neistotu. Patria do skupiny matematického programovania a vieme pomocou nich modelovať úlohy na riešenie rozhodnutí, kde využívame simulácie na ich dosiahnutie. Tieto rozhodnutia sa zväčša týkajú problémov reálneho sveta, kde sa nachádzajú parametre, ktoré v momente rozhodnutia nie sú známe v čase. Príklady využitia v reálnom svete sú napríklad biológia, v ktorej je možné využiť stochastické programovanie na vytvorenie modelov resp. simulácií, ktoré zahŕňajú náhodnosť premenných ako sú genetické informácie alebo evolúcia spojená s dynamickosťou populácie. V informačných technológiách je možné využiť stochastické programovanie v oblasti kryptografie, umelej inteligencie alebo v strojovom učení. Ak by sme sa zamerali na obor financií, tu je možné náhodnosť modelovať pri cenách aktív alebo úrokových sadzbách. Simulácia znamená napodobňovať fungovanie reálneho systému pomocou počítačového modelu, pričom pod pojmom systém rozumieme určitú časť reálneho sveta, ktorú skúmame. Pri definovaní problému rozoznávame dva rôzne problémy (deterministický a stochastický), ktoré sa líšia v tom, ako sa správajú a ako sa s nimi pracuje. Aplikácie týchto problémov sú spísané v diele Comparison of deterministic and stochastic approaches to global optimization (Liberti & Kucherenko, 2005).

Deterministický problém je taký, kde môžeme presne vypočítať budúcu udalosť bez zapojenia náhodnosti. Ak je niečo deterministické, máme všetky potrebné údaje na predpovedanie (určenie) výsledku s istotou. Deterministické modely majú rôzne aplikácie vo vedeckom výskume a pri finančných trhoch. Aplikácie z praxe sú napríklad:

- Predpovedanie životnosti budov a ich komponentov je založené na regresnej analýze, ktorá môže popísať vzťah medzi degradačnými faktormi a stavom fasády (Silva, Brito & Gaspar 2016).
- Štúdie o vývoji klímy spočívajú vo vytváraní scenárov budúceho spoločenského vývoja, odhadovanie emisií skleníkových plynov. Tieto hodnoty sa vkladajú do matematických klimatických modelov, ktoré predpovedajú budúce otepľovanie a škody spôsobené týmto otepľovaním (Kung, Li & Lin, 2018).

Stochastický problém je taký, kde môžeme len odhadnúť pravdepodobnosť každého výsledku a rozhodnúť sa na základe tohto odhadu. Stochastické modely majú istú vrodenú náhodnosť, kde rovnaká sada parametrov a počiatočných podmienok povedie k súboru rôznych výstupov. Zatiaľ čo sa deterministické modely často používajú na predpovedanie budúcich investičných výnosov, stochastické modely sa používajú na riešenie neistôt vstupov. Príklady by mohli byť nasledovné:

- Modelovanie pohybu molekuly plynu - smer a rýchlosť pohybu molekuly plynu sú ovplyvnené náhodnými zrážkami s inými molekulami (Rudyak & Lezhnev, 2020).
- Modelovanie vo financiách - predpovedanie finančných trhov. Známe aplikácie sú simuláciou Monte Carlo, regresné modely a Markovove reťazce (Brooks, 2002).

Pri tvorbe simulačných modelov je často potrebné modelovať procesy ako náhodné premenné. Je to spôsobené tým, že vo väčšine reálnych prípadov (časové intervaly medzi príchodmi dvoch zákazníkov alebo čas obsluhy zákazníkov) procesy nie sú časovo konštantné. Modelovanie takejto udalosti je možné prostredníctvom vhodne zvoleného diskrétneho, alebo spojitého pravdepodobnostného rozdelenia. V modeloch obsluhy sa čas medzi príchodom zákazníkov a čas obsluhy môže modelovať ako konštanta, alebo ako náhodná premenná pomocou pravdepodobnostného rozdelenia. Na modelovanie času dodania sa najčastejšie používa gamma rozdelenie. V modeloch obnovy sa ako náhodná premenná môže modelovať čas zlyhania a najčastejšie sa tu používajú exponenciálne, gamma alebo Weibullovo rozdelenie. Ak sú k dispozícii obmedzené informácie (údaje) o procese, tak sa väčšinou používa rovnomerné rozdelenie, alebo trojuholníkové rozdelenie. Rovnomerné rozdelenie sa používa, ak vieme, že náhodná premenná sa nachádza v určitom intervale, ale nemáme žiadne ďalšie

informácie. Trojuholníkové rozdelenie sa používa v prípade, ak poznáme maximálnu, minimálnu a najpravdepodobnejšiu hodnotu (Prékopa, 1995).

2 Modely a metódy riešenia úloh stochastického programovania

V tejto kapitole sa budeme zaoberať modelmi a metódami riešenia stochastického programovania. Začiatok kapitoly bude sprevádzaný všeobecným popisom modelu lineárneho programovania, nasledovaný náhodnými procesmi a odhadom ich parametrov a základné typy spojitého rozloženia pravdepodobnosti.

2.1 Všeobecný popis modelu lineárneho programovania

Každý optimalizačný model je zložený z účelovej funkcie a obmedzujúcich podmienok, ktoré spolu tvoria dve hlavné časti. Účelová funkcia popisuje cieľ optimalizácie a obmedzujúce podmienky, zodpovedajú za náročnejšie dosiahnutie daných cieľov vďaka vytváraným obmedzeniam. Množina, ktorá spĺňa všetky podmienky sa nazýva množina prípustných riešení úlohy (X^P). Graficky sa dá množina X^P znázorniť ako *n*-dimenzionálna polyedrická množina, v ktorej *n* je počet premenných daného modelu. Pokiaľ existuje množina prípustných riešení, je následne vybrané riešenie, ktoré najlepšie vyhovuje zadanej účelovej funkcii. Toto riešenie je nazývané optimálne riešenie (x^{opt}). Optimálne riešenie úlohy lineárneho programovania (ďalej LP) sa vždy nachádza v jednom z vrcholov polyédru reprezentujúceho množinu X^P. Pokiaľ sa jedná o úlohu lineárneho programovania, potom sú všetky časti modelu popísané pomocou lineárnych rovníc a nerovníc. Všeobecný model LP je zachytený zápisom (1), kde n je počet premenných, m počet obmedzujúcich podmienok, c je vektor koeficientov účelovej funkcie rozmeru $n \times 1$, A je matika technických koeficientov rozmeru $m \times n$, a vektor pravých strán je označovaný ako b s rozmermi $m \times 1$. Platí, že $c, b, A \in$ R, kde R predstavuje reálne čísla.

$$z(x) = c^T x \rightarrow max \tag{1}$$

za podmienok:

$$Ax \leq b$$
$$x \in R_+^n$$

$$\max(c^T x \mid Ax \leq b, x \in R_+^n)$$

2.2 Náhodné procesy a odhad ich parametrov

Stanovenie alebo odhad rozdelení pravdepodobnosti je zásadným problémom, ktorému je nutné pri stochastickom modelovaní vždy čeliť. Vzhľadom k prítomnosti faktoru neistoty nie je možné stanoviť hodnoty premenných deterministicky, ale je možné tieto hodnoty aspoň odhadnúť spolu s pravdepodobnosťou, s ktorou nastanú.

Rozdelenie pravdepodobnosti náhodnej veličiny je pravidlo, ktoré určuje každej hodnote tejto veličiny, prípadne každému intervalu hodnôt pravdepodobnosť, s ktorou môže nastať. Jedná sa teda o zobrazenie $f : R \rightarrow < 0; 1 >$. Toto zobrazenie sa nazýva funkcia hustoty ak sa jedná o spojité náhodné veličiny alebo pravdepodobnostné funkcie, ak sa jedná o diskrétne náhodné veličiny. Primitívne funkcie k funkcii hustoty sa potom nazýva distribučná funkcie rozdelenia náhodnej veličiny $F(x)$ zobrazené v rovnici (2):

$$F(x) = \int_{-\infty}^{x} f\,(t)\,dt \tag{2}$$

Hodnota distribučnej funkcie v bode x udáva pravdepodobnosť, s ktorou náhodná premenná nenadobudne hodnôt vyšších než práve x.

2.3 Algoritmy pre optimalizáciu

Pre optimalizáciu dvoch alebo viacstupňových úloh existujú algoritmy ako je L-shaped metóda, alebo metóda dekompozície bázy.

Metóda L-shaped je založená na princípe rezania rozhodovacích stromov. Začína sa súborom ohraničujúcich podmienok, ktoré obmedzujú možné hodnoty rozhodovacích premenných. Potom je vykonaná optimalizácia pre prvý stupeň rozhodovania, pričom sa ignoruje stochastická povaha problému (predpokladá sa, že neexistujú žiadne neistoty). Na základe tohto výsledku sa generuje "suboptimálny" podproblém pre druhý stupeň. Podproblém pre druhý stupeň sa nazýva L-shaped podproblém a je to problém so zúženým priestorom možných riešení. Jeho cieľom je zlepšiť pôvodné riešenie získané pre prvý stupeň tým, že zohľadní stochastickú povahu. L-shaped podproblém sa rieši pomocou metódy rezania podľa rezov (cutting-plane method). Táto metóda postupne pridáva podmienky, známe ako rezy, ktoré odstránia neplatné časti priestoru riešení. Tieto rezy sa odvodzujú zo stochastických obmedzení modelu a aktualizujú sa postupne tak, aby sa získalo čo najlepšie riešenie. Proces rezania a aktualizácie rezov pokračuje, kým sa nedosiahne konvergencia a nie je možné nájsť lepšie riešenie alebo dosiahnuť požadovanú presnosť. Celkovo povedané, metóda L-shaped umožňuje efektívne riešenie dvojstupňových stochastických modelov tým, že kombinuje optimalizáciu na prvom stupni s postupným zlepšovaním riešenia pomocou rezy na druhom stupni, čím zohľadňuje stochastickú povahu problému (Pereira et al., 2020).

Druhou možnou metódou je princíp metódy dekompozície bázy, ktorý spočíva v rozložení pôvodného problému na menšie, menej zložité podproblémy. Tieto podproblémy sú následne riešené nezávisle a ich riešenia sú následne kombinované tak, aby poskytli riešenie pôvodného problému. Častá aplikácia v prípade, keď je možné rozdeliť model na dva alebo viac podproblémov, pričom každý z nich sa zaoberá jedným aspektom neistoty. Napríklad, v dvojstupňových stochastických modeloch by sa prvý stupeň mohol zaoberať rozhodnutiami priamočiaro, zatiaľ čo druhý stupeň by sa mohol zaoberať rozhodnutiami, ktoré sú ovplyvnené neistotou alebo stochastickými faktormi. Každý z týchto podproblémov je potom riešený samostatne. Po získaní riešení pre jednotlivé podproblémy sa tiež často vykonáva iteratívna aktualizácia, ktorá slúži na zlepšenie a konzistenciu riešení. Viac o tejto metóde je možné sa dozvedieť v dielach Decomposition Methods for Machine Learning with Small, Incomplete or Noisy Datasets (Caiafa et al., 2020) a Machine Learning-Based Epileptic Seizure Detection Methods Using Wavelet and EMD-Based Decomposition Techniques: A Review (Thangarajoo et al., 2021).

3 Modely a metódy riešenia stochastického programovania

Stochastické programovanie je oblasť matematickej optimalizácie, ktorá využíva znalosti teórie pravdepodobnosti a matematickej štatistiky pre zohľadnenie faktoru neistoty, ktorý je v modely prítomný (Dupáčová, 1986). Jedná sa o vedecký odbor, ktorý sa zaoberá náhodnými premennými (Prékopa, 1995). Pokiaľ by sme chceli formulovať všeobecnú formuláciu modelu stochastického programovania vyzerala by nasledovne:

$$f\ (x, c) \rightarrow min \tag{3}$$

za podmienok:

$$g_i(x, c) \leq 0, i = 1, \dots, n$$

kde x je rozhodovací vektor ($x \in R^n$) a c je náhodný vektor z pravdepodobnostného priestoru (Ω, β_Ω, P) a platí, že $c \in \Omega \subset R^n$.

Pokiaľ je modelovanie obmedzené iba na lineárne modely, v tomto prípade modely stochastického lineárneho programovania (SLP), sa dá všeobecne zapísať nasledovne:

$$z = c'^T x \rightarrow \min \tag{4}$$

za podmienok:

$$A'x \leq b'$$

Apostrof nad maticami koeficientov, vektor pravých strán a vektor koeficientov účelovej funkcie symbolizuje, že jeden alebo viac prvkov je zadaný pomocou stochastickej náhodnej premennej.

3.1 Klasifikácia modelov stochastického programovania

Pri stochastickom programovaní sa je možné stretnúť s viacerými kritériami, podľa ktorých sa dajú stochastické optimalizačné modely klasifikovať. Voľba konkrétneho modelu a postup jeho riešenia záleží na účelu a cieli modelovania a všeobecne na charaktere rozhodovacieho problému, ktorý má byť pomocou modelu riešený (Prékopa, 1995).

Prvým hľadiskom klasifikácie je čas, kedy je potrebné učiniť rozhodnutie. Prvou možnosťou je, že je možné počkať na realizáciu náhodného vektoru c a až potom na základe tejto informácie sa rozhodnúť. Tento model sa nazýva *wait-and-see*. Takýto prípad obsahuje náhodnú premennú a je obdobou deterministického rozhodovacieho problému (Prékopa, 1995). Druhá skupina úloh, v ktorý sa musí vykonať rozhodnutie x pred tým, ako je známa realizácia náhodného vektoru c sa nazývajú *here-and-now* úlohy. V ekonomickej praxi sú častejšie a jedná sa o výpočtovo zložitejšie problémy. Typickým príkladom z praxe je napr. problém voľby investičného portfólia kedy sú známe iba aktuálne nákupné alebo predajné ceny, ale tieto ceny sa budú ďalej v budúcnosti náhodne vyvíjať a meniť a výnos zo zvoleného portfólia preto nejde predom presne stanoviť. S tým, či sa rozhodnutie realizuje pred alebo po pozorovaní náhodnej premennej súvisí ďalšie delenie modelu SLP na statické a dynamické modely.

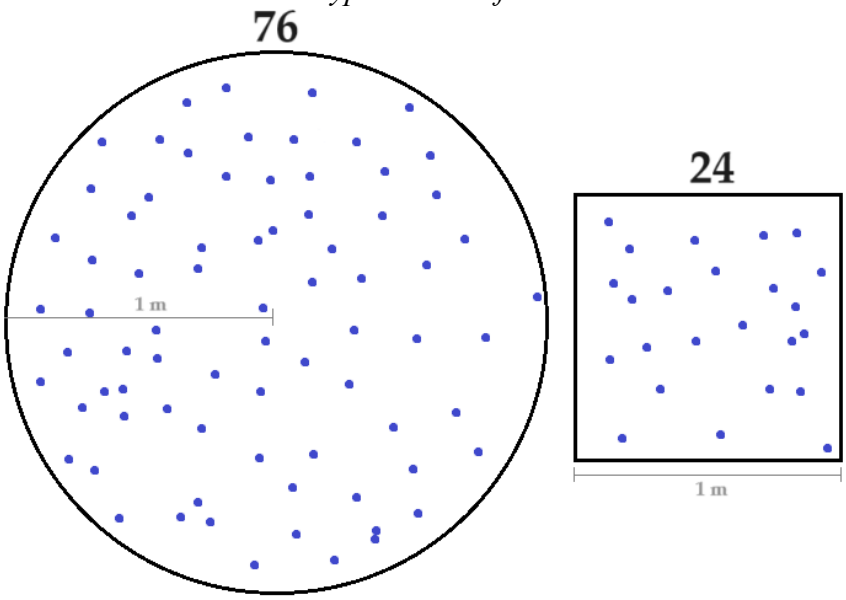
Statické modely sa používajú na predpovedanie statických výsledkov v danom momente, pričom sa najskôr realizuje rozhodnutie a potom až nasleduje pozorovanie (*here-and-now* úlohy). Statické modely sú jednostupňové (*single-stage*) Pri statických modeloch rozpoznávame najznámejšiu simuláciu Monte Carlo, ktorá predstavuje štatistický experiment umožňujúci riešiť úlohy deterministického alebo stochastického charakteru. Na experimentovanie je zostavená pravdepodobnostná úloha, ktorej výsledok riešenia má taktiež pravdepodobnostný charakter, čiže ide o štatistický odhad hodnôt jednotlivých veličín. Presnosť riešenia je závislá na počte opakovaných experimentov. Využitie simulácie je možné pri hľadaní hodnoty Ludolfovho čísla π. Na obrázku 1 si uveďme príklad výpočtu Ludolfovho čísla. Majme dve nádoby kruhovej a štvorcovej podstavy umiestnené vedľa seba spoločnej podložke (podstave). Nech je polomer kruhu a strana štvorca jeden meter. Pokiaľ by sme náhodne púšťali guľôčky zvrchu na podstavu, jednotlivé nádoby by sa pomaly plnili guľôčkami. Výsledok na výpočet Ludolfovho čísla spočíva v podiele počtu guľôčok v kruhu s počtom guľôčok v štvorci.

$$\pi \approx \frac{Počet\ guľôčok\ v\ kruhu}{Počet\ guľôčok\ v\ štvorci} \tag{5}$$

Čím vyšší počet iterácií (guľôčok) tým je vyššia presnosť experimentu a nastáva tu konvergencia k hodnote π. Pokiaľ by sme generovali 100 guľôčok, kde 76 by sa nachádzalo v kruhovej nádobe a 24 v štvorcovej nádobe výsledok by bol 3.167 (5). Ďalšie aplikácie statických modelov sú napríklad pri softvérovom modelovaní a dizajne, kde je modelovanie

využité na popis statickej štruktúry modelovaného systému alebo pri adaptívnom filtrovaní referenčného signálu.

Obr. 1: Výpočet Ludolfovho čísla



Zdroj: Vlastné spracovanie

Dynamické modely sa používajú na modelovanie systémov vyvíjajúcich sa v čase spojito, pričom stavy systému sa menia v diskrétnych časových okamihoch a aspoň jedno pozorovanie musí byť realizované pred vykonaním rozhodnutia (kombinácia *wait-and-see* úlohy s úlohou *here-and-now*). V praxi sa simulačné modelovanie uplatňuje na analýzu a optimalizáciu, napríklad výrobných procesov, organizáciu skladov, logistických procesov, plánovania výroby, finančného plánovania, riadenia vnútropodnikovej dopravy, riadenia a plánovania projektov. Simulácie teda použijeme ak skúmame vzťahy v rámci komplexných systémov. Pri skúmaní informačných, organizačných a environmentálnych zmien na vývoj systému. Pri zlepšovaní chápania systému, nakoľko získané informácie nám zo simulácie upresnia jeho správanie.

Stochastické modely sa teda dajú klasifikovať nasledovne:

Podľa počtu rozhodnutí a možnosti svoje rozhodnutia neskôr korigovať: jednostupňové a viacstupňové. Podľa počtu časových období, pre ktoré sa rozhodnutie prijíma: single-period a multi-period. Podľa toho, či sa pred rozhodovaním poznáme realizáciu náhodného vektoru: here-and-now a wait-and-see. Podľa linearity modelu: lineárne (SLP) a nelineárne.

Podľa typu náhodnej premennej: s diskrétnou náhodnou premennou a so spojitou náhodnou premennou. Postup pri riešení ľubovoľnej stochastickej optimalizačnej úlohy je nasledovný (Prékopa 1995):

1. Zostrojenie deterministického podkladového modelu. Ten slúži na ujasnenie si štruktúry cieľového modelu, prípadne pre získanie prvého odhadu optimálnej hodnoty účelovej funkcie.
2. Výber parametrov modelu, ktoré v modeli budú stochastické.
3. Stanovenie rozdelenia pravdepodobnosti jednotlivých stochastických premenných. Buď je možné využiť niektoré z „klasických“ rozdelení, u ktorých je známa rovnica distribučnej funkcie a funkcie hustoty a potom stačí iba odhadnúť parametre týchto rozdelení. Druhou možnosťou je stanoviť si vlastné empirické rozloženie pravdepodobnosti.
4. Stanovenie ďalších parametrov modelu.

5. Prevod stochastického modelu na jeho deterministický ekvivalent pomocou príslušných metód a algoritmov.
6. Optimalizácie deterministického modelu.

3.2 Využitie náhodnej zložky v stochastických modeloch

Prvý prípad ako môže náhodná zložka ovplyvňovať model je pri ovplyvnení koeficientov účelovej funkcie, pričom obmedzujúce podmienky sú stanovené deterministicky. Typickým príkladom takéhoto modelu je napr. úloha optimalizácie portfólia, kde sú stochastické výnosy z jednotlivých aktív a tiež riziko spojené s držaním týchto výnosov. Cieľom je maximalizovať úžitok majiteľa portfólia, ktorý rastie s celkovým výnosom z portfólia a ktorý v prípade risk-averse agenta klesá s rizikom portfólia viď Kall a Mayer (2011). Aby bolo možné realizovať optimalizáciu, je nutné zaviesť vhodné usporiadanie. Usporiadanie sa zavedie tým, že je účelová funkcia transformovaná do deterministickej podoby. Birge a Louveaux (1997), Kall a Mayer (2011) a Dupačová (1986) rozlišujú dve možnosti, ako ide zo stochastickej účelovej funkcie odstrániť náhodnú zložku. Prvou možnosťou je previesť účelovú funkciu na jej strednú (očakávanú) hodnotu podľa tzv. *E*-kritéria. Nevýhodou je, že v tomto prípade je využitá iba malá časť informácie o rozdelení pravdepodobnosti náhodnej premennej. Druhou možnosťou je do deterministického prepisu zahrnúť tiež riziko, ako tomu je napr. u modelov výberu portfólia. Riziko je možné modelovať rôznymi spôsobmi a pomocou rôznych ukazovateľov, vždy sa ale dá povedať, že pri odpore k riziku sa so zvyšujúcim sa ukazovateľom klesá celkový úžitok.

Ak je pri transformácii stochastickej účelovej funkcie do deterministickej formy zohľadnená stredná hodnota aj riziko vo forme napr. rozptylu, jedná sa o *mean-risk* modely.

Všeobecný zápis pomocou úžitkovej účelovej funkcie je nasledovný:

$$z(x) = E[f(x,c)] + \psi k \rightarrow max \tag{6}$$

kde $E[f(x,c)]$ je stredná hodnota pôvodnej funkcie, ψ je koeficient vyjadrujúci postoj k riziku (pre risk-averse agentov je hodnota ψ záporná, pre risk-lover agentov naopak kladná a pre agentov s neutrálnym vzťahom k riziku je ψ=0 hodnota účelovej funkcie je potom daná iba strednou hodnotou) a k je miera rizika.

Pri modeloch so stochastickou premennou v podmienkach vlastného obmedzenia, môžu byť náhodnou premennou nahradené koeficienty na ľavých stranách podmienok alebo konštanty na pravých stranách obmedzujúcich podmienok, prípadne sa môže jednať o kombináciu týchto prípadov. Model SLP so stochastickou premennou v podmienke vlastného obmedzenia má nasledovný všeobecný zápis:

$$z(x) = c^T x \rightarrow \min \tag{7}$$

za podmienok, že:

$$g_i\,(x,c) \leq 0, i = 1,2,\dots,n\,,$$

kde g_i je lineárna funkcia agregujúca ľavou aj pravou stranou obmedzujúcej podmienky a je ovplyvnená náhodnou zložkou c.

3.3 Viacstupňové stochastické modely

Hlavná myšlienka viacstupňových stochastických modelov bola predstavená pri klasifikácii modelov stochastického programovania.

Dvojstupňové stochastické programovanie sa vyznačuje rozhodnutiami, ktoré sú robené v dvoch fázach. Prvá fáza predstavuje rozhodnutia, ktoré sa musia urobiť predtým, ako sa stane nejaká náhodná udalosť (napríklad predaj tovaru pred tým, ako sa dozvieme o dopyte). Druhá fáza predstavuje rozhodnutia, ktoré sa robia po tom, ako sa táto náhodná udalosť stane (napríklad nákup dodatočného tovaru, keď je dopyt vyšší ako očakávaný). Príkladom môže byť farmárov problém, kde farmár musí rozhodnúť, koľko z každej plodiny vypestuje (prvý stupeň), a potom koľko z každej plodiny predá alebo kúpi (druhý stupeň). Všeobecný zápis dvojstupňovej úlohy je nasledovný:

$$z = c^T x + E(\min\{c'^T y \mid Ax + A'y \leq b\}) \rightarrow min, x \in X \quad (8)$$

kde $x \in R^n$ je rozhodovací vektor prvého stupňa úlohy (*here-and-now* úloha) a $y \in R^n$ je rozhodovací vektor druhého stupňa, v ktorom je realizácia náhodného vektoru známa. c^T a A je vektor koeficientov účelovej funkcie, resp. matice ľavých strán obmedzení pre prvý stupeň, ich ekvivalenty sa symbolom „ ´ “ sa viažu k druhému stupňu. Množina X obsahuje všetky také vektory x, pre ktoré má úloha $\min\{c'^T y \mid Ax + A'y \leq b\}$ prípustné riešenie pre každú možnú hodnotu náhodného vektoru c. Ten obsahuje údaje o náhodných premenných, ktoré sú odhalené pred druhým stupňom (buď sú stochastické všetky parametre c´, A, A´, b alebo iba niektoré z nich). V prvom stupni je optimalizovaná jednak deterministická časť účelovej funkcie $c^T(x)$ plus očakávané optimum účelovej funkcie v druhom stupni $\min\{c'^T y\}$. Druhý stupeň potom určuje rozhodnutie y, ktoré bude prijaté, ak sú známe hodnoty náhodného vektoru c, a ak bolo skôr rozhodnutie x prijaté. Iný pohľad na druhý stupeň môže byť ten, že je vnímaný ako akási „oprava“ pri porušení pôvodných obmedzujúcich podmienok $Ax \leq b$, ktoré bolo pôvodne zaťažené náhodou. Ľavé strany porušených podmienok sú potom kompenzované (viď $A'y$ v obmedzujúcej podmienke) a táto kompenzácie vedie k penalizácii účelovej funkcie ($c'^T y$). Pre samostatnú optimalizáciou dvoch a viacstupňových úloh existuje mnoho algoritmov, ako napríklad „L-shaped“ metóda alebo metóda dekompozície bázy.

Viacstupňové stochastické programovanie je rozšírením dvojstupňového stochastického programovania na viacero stupňov alebo časových období. Každé obdobie môže mať vlastné rozhodnutia a náhodné udalosti. Príkladom môže byť plánovanie výroby a skladovania v továrni, kde sa v každom období musí rozhodnúť, koľko výrobkov vyrobiť, koľko skladovať a koľko predávať, pričom dopyt je náhodný a môže sa líšiť v každom období.

S rekurentnými rozhodnutiami sa vyznačujú rozhodnutiami, ktoré sa musia robiť opakovanie v čase, pričom každé rozhodnutie ovplyvňuje stav systému pre budúce rozhodnutia. Príkladom môže byť riadenie zásob, kde sa musí rozhodnúť, koľko objednať v každom období, pričom množstvo objednaného tovaru ovplyvňuje množstvo tovaru na sklade v budúcich obdobiach.

S časovým horizontom sa vyznačujú rozhodnutiami, ktoré musia byť urobené v rámci istého časového horizontu. Časový horizont môže byť konečný (napríklad plánovanie výroby na nasledujúci rok) alebo nekonečný (napríklad plánovanie dôchodkového fondu s neistým dátumom smrti). Príkladom môže byť plánovanie investícií, kde sa musí rozhodnúť, koľko investovať do aktív v každom období, pričom výnosy z aktív sú náhodné a môžu sa líšiť v každom období.

4 Dvojstupňové stochastické programovanie s využitím programu MS Excel

V tomto príklade vytvárame simuláciu, kde predpokladáme existenciu apartmánového komplexu na Slovensku, kde je naplánovaná štvordňová konferencia a organizátori chcú čo najpresnejšie určiť, koľko hotelových izieb rezervovať, aby sa minimalizovali celkové náklady. Hostia budú ubytovaní v hoteli Slopartments, kde je maximálny počet hotelových apartmánov

je 400. Hotel Slopartments dal možnosť kvôli konferencii využiť zvýhodnenú cenu 145 € za noc ak sa apartmány rezervujú vopred. Za každý rezervovaný apartmán, ktorý nie je využitý musia organizátori platiť plnú sumu, čo činí 193 € za každý deň konferencie. Alternatívne, ak je viac uchádzačov o konferenciu ako rezervovaných apartmánov, organizátori musia zaplatiť 229 € za osobu na deň. Tieto náklady pokrývajú najbližší menej kvalitný hotel Spoor s cenou 161 € na deň plus transport z hotela na konferenciu v hodnote 68 €. Neistotou je, že organizátori nevedia koľko ľudí príde na konferenciu. Podľa organizátorov, je očakávaná návštevnosť vopred predpovedateľná a je znázornená v tabuľke 1.

Tab. 1: Predom známe informácie od usporiadateľov

Scenár (zajtrajšok)	**Pravdepodobnosť**	**Potrebné hotelové izby**
1	15%	250
2	25%	300
3	30%	350
4	20%	400
5	10%	450

Zdroj: Vlastné spracovanie

r = počet izieb, ktoré sa rezervujú dopredu v hoteli-1 (premenná z 1. stupňa)
o_i, u_i= nad-rezervácia, pod-rezervácia izieb podľa jednotlivých scenárov (i= 1, ..., 5) (2.stupeň)
Nad-rezervácia nastáva ak r je väčšie ako dopyt a pod-rezervácia ak r je menšie ako dopyt.
Formulácia matematického modelu je nasledovná:

$$\boldsymbol{min_z(u) = 145r + 0.15(48o_1 + 229u_1) + 0.25\,(48o_2 + 229u_2) + 0.30\,(48o_3 + 229u_3) + 0.20\,(48o_4 + 229u_4) + 0.10\,(48o_5 + 229u_5)} \quad (9)$$

kde 145 predstavuje cenu pred-rezervovaného apartmánu, desatinné hodnoty zodpovedajú pravdepodobnosti nastátia udalosti, 48 predstavuje dodatočné náklady ak nad-rezervujem, 229 ak pod-rezervujem. Účelová funkcia zobrazuje očakávané náklady. 3 predstavuje trojdňovú konferenciu.

Obmedzenia (premenná – prvý stupeň):

$$r \leq 400$$

Obmedzenia (premenné – druhý stupeň):

$$r - o_1 + u_1 = 250$$
$$r - o_2 + u_2 = 300$$
$$r - o_3 + u_3 = 350$$
$$r - o_4 + u_4 = 400$$
$$r - o_5 + u_5 = 450$$
$$r, o_i, u_i \geq 0$$

V programe MS Excel sme si zostrojili vyššie navrhnutý príklad. Žlté polia predstavujú polia, ktorých hodnota bola pôvodne nevyplnená. Oranžové pole predstavuje účelovú funkciu na výpočet očakávaných nákladov po druhom stupni modelu.

Obr. 2: Optimalizácia v MS Excel

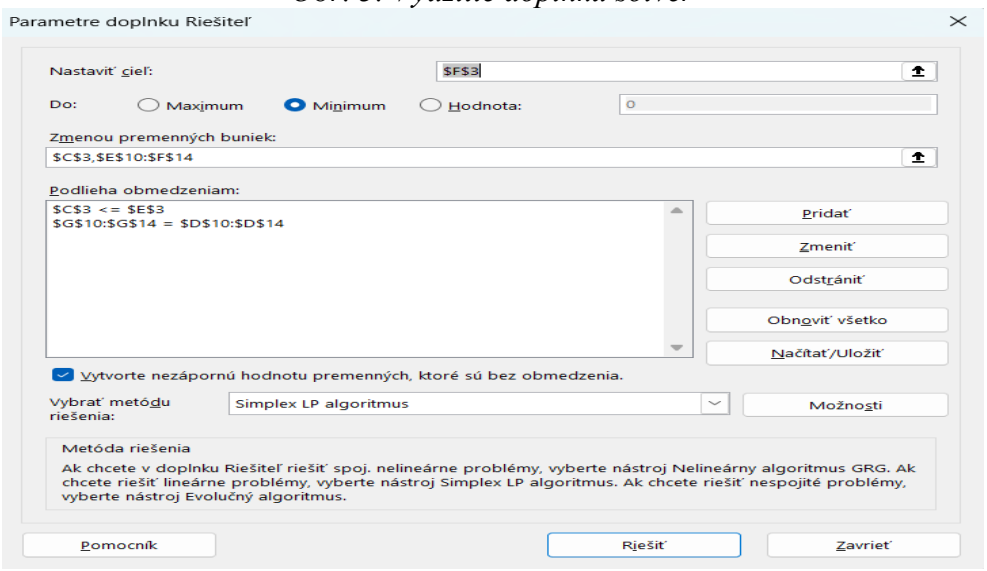
			Dostupných apartmánov	Očakávané náklady
Rezervovať pre dopyt v Slopartments	300	<=	400	221,240.00 €
Vopred rezervovanie Slopartments	145.00 €			
Plná cena apartmánu Slopartments	193.00 €			
Spoor hotel	161.00 €			
Cestovné zo Spoor	68.00 €			

Scenár	Pravdepodobnosť	Dopyt	Nepoužité izby (nad-rezervácia o)	Dodatočne požadované (pod-rezervácia u)	Rezervované - nadbytok + nedostatok
1	15%	250	50	0	250
2	25%	300	0	0	300
3	30%	350	0	50	350
4	20%	400	0	100	400
5	10%	450	0	150	450
		Náklady	48	229	

Zdroj: Vlastné spracovanie

Princíp výpočtu spočíva vo využití riešiteľa (doplnok solver), kde minimalizácia je vypočítaná v bunke F3 ako cieľová hodnota. Následne sú do modelu vnesené zmeny buniek žltej farby. Týmto interpretujeme nastavenie ideálneho rozloženie izieb. Na základe stanovených podmienok rezervovaných izieb, aby boli ich počet bol menší alebo rovný ako 400 dostupných apartmánov a aby sa rovnali počty návštevníkov (G stĺpec) a dopytu (D stĺpec). Po spustení riešiteľa lineárnym simplexovým algoritmom vidíme výsledok 300 izieb. Tento postup je zobrazený na obrázku 3.

Obr. 3: Využitie doplnku solver



Zdroj: Vlastné spracovanie

Tento výsledok vieme interpretovať nasledovne: neexistuje také riešenie, ktoré je možné učiniť dnes, aby bolo ideálne vo všetkých prípadoch. Tento dvojstupňový stochastický optimalizačný model udáva optimálnu hodnotu pre usporiadateľov takýchto typov konferencií očakávaný dlhodobý priemer minimalizujeme výdavky najviac ako je možné. Aktuálne je najlepšie pre usporiadateľov zarezervovať vopred 300 apartmánov. V 25 % prípadoch budeme v druhom scenári bez výdavkov. V ostatných prípadoch sme minimalizovali náklady tak, aby v každom možnom výsledku boli náklady minimálne. Na 15 % budeme mať rezervovaných 50 izieb naviac a v 60 % prípadoch sa bude musieť doobjednať hosťom ďalšie ubytovanie v hoteli Spoor. Zaujímavé si je všimnúť, že 30 % scenár s dopytom 350 nebol najlepším riešením tejto situácie. Očakávané náklady na štvordňovú konferenciu sa po minimalizácii dostávajú na hodnotu 221 240 €. Priebeh výpočtu príkladu je možné vidieť na obrázkoch 2 a 3.

5 Riešenie úlohy prístupom s normálnou distribúciou pravdepodobnosti s využitím Python

V tejto kapitole budú riešené úlohy v programovacom jazyku Python, kde bude využité pri zobrazení neistoty normálne pravdepodobnostné rozdelenie.

5.1 Úloha farmár

Pri tejto úlohe sme vychádzali zo skonštruovaných príkladov uvádzaných v dielach Linear programming with Python and Pulp (Parganiha, 2018) a Crop profit optimization for farmers (Romero & Smith, 2016). V tomto príklade uvažujeme o poľnohospodárovi, ktorý má 100 árov pôdy a môže pestovať pšenicu alebo kukuricu. Farmár sa chce rozhodnúť, koľko árov každej plodiny vysadí, aby maximalizoval očakávaný zisk. Zisk z každej plodiny je však neistý vzhľadom na faktory, ako sú počasie, škodcovia a trhové ceny. Túto neistotu modelujeme tak, že predpokladáme, že zisky na ár pšenice a kukurice sú náhodné premenné, ktoré sa riadia normálnym rozdelením. Stredná hodnota tohto rozdelenia predstavuje očakávaný zisk na ár a štandardná odchýlka predstavuje variabilitu alebo riziko. Problém poľnohospodára je potom formulovaný ako stochastický optimalizačný problém. Cieľom je maximalizovať očakávaný celkový zisk, ktorý je súčtom očakávaných ziskov z každej plodiny. Rozhodovacími premennými sú počet árov, ktoré sa majú osiať pre každú plodinu. Medzi obmedzenia problému patrí obmedzenie týkajúce sa pôdy, ktoré stanovuje, že celkový počet vysadených akrov nesmie prekročiť celkovú dostupnú pôdu, a obmedzenie nezápornosti, ktoré stanovuje, že počet akrov vysadených pre každú plodinu musí byť nezáporný. Na riešenie tohto problému používame prístup simulácie Monte Carlo. Táto metóda je štatistickým experimentom, kde presnosť výsledku stúpa s počtom iterácií a popisujeme ho v kapitole 3.1. To zahŕňa vytvorenie mnohých scenárov pre zisky na aker pre každú plodinu, riešenie optimalizačného problému pre každý scenár a následné spriemerovanie výsledkov. Riešenie nám poskytne priemerný počet akrov, ktoré treba osiať pre každú plodinu, čím sa maximalizuje očakávaný celkový zisk, pričom sa zohľadní neistota v ziskoch na aker pre pšenicu a kukuricu.

Matematická formulácia modelu:

$$E[Z] = E[c_1]x_1 + E[c_2]x_2$$

Obmedzenia:

$$x_1 + x_2 \leq 100$$
$$x_1, x_2 \geq 0$$

Nech x_1 a x_2 sú premenné reprezentujúce áre, na ktorých sa pestuje pšenica a kukurica,. c_1a c_2 predstavujú náhodné premenné reprezentujúce zisky na ár z pšenice a kukurice. Pri týchto premenných využívame normálne rozdelenie s priemerom μ_1 pre pšenicu a μ_2 pre kukuricu a štandardnou odchýlkou pre pšenicu σ_1 a kukuricu σ_2. $E[c_1]$ a $E[c_2]$ sú očakávané zisky v prepočte na ár pre pšenicu a kukuricu. Očakávaný zisk je priemerom normálneho rozdelenia, kde $E[c_1] = \mu_1$ a $E[c_2] = \mu_2$. Táto matematická formulácia cieli na nájdenie počtu árov na zasadenie každej z plodín a maximalizácie celkového zisku pomocou využitia neistoty v ziskoch na ár pre pšenicu a kukuricu.

Obr. 4: Optimalizácia výsevu plodín

```python
# Importujeme potrebné knižnice
import numpy as np
from scipy.optimize import linprog

# Nastavíme priemerný zisk a štandardnú odchýlku na akejkoľvek ploche pre pšenicu a kukuricu
mu = np.array([50, 30])  # Priemerný zisk
sigma = np.array([10, 5])  # Štandardné odchýlky

# Nastavíme počet scenárov, ktoré chceme generovať
num_scenarios = 1000

# Nastavíme seed pre generátor náhodných čísel pre reprodukovateľnosť
np.random.seed(0)

# Generujeme scenáre pre zisk na akejkoľvek ploche pomocou normálneho rozdelenia
profits = np.random.normal(mu, sigma, (num_scenarios, 2))

# Inicializujeme polia na ukladanie výsledkov
x1_values = np.zeros(num_scenarios)  # Akre pšenice pre každý scenár
x2_values = np.zeros(num_scenarios)  # Akre kukurice pre každý scenár

# Cyklus pre každý scenár
for i in range(num_scenarios):
    # Nastavíme koeficienty cieľovej funkcie pre aktuálny scenár
    # Poznámka: Používame záporné hodnoty, pretože linprog robí minimalizáciu
    c = -profits[i]

    # Nastavíme koeficienty pre nerovnosti
    A = [[1, 1]]  # Koeficienty pre obmedzenie pôdy

    # Nastavíme pravú stranu nerovností
    b = [100]  # Celková dostupná pôda

    # Nastavíme hranice pre rozhodovacie premenné
    x0_bounds = (0, None)  # Akre pšenice musia byť >= 0
    x1_bounds = (0, None)  # Akre kukurice musia byť >= 0

    # Riešime lineárny programovací problém pre aktuálny scenár
    res = linprog(c, A_ub=A, b_ub=b, bounds=[x0_bounds, x1_bounds], method='highs')

    # Uložíme riešenie pre aktuálny scenár
    x1_values[i] = res.x[0]
    x2_values[i] = res.x[1]

# Vypočítame priemerné hodnoty zo všetkých scenárov
x1_avg = x1_values.mean()
x2_avg = x2_values.mean()

# Vytlačíme priemerné hodnoty
print('Priemerný počet akrov pšenice na výsadbu:', x1_avg)
print('Priemerný počet akrov kukurice na výsadbu:', x2_avg)
```

```
Priemerný počet akrov pšenice na výsadbu: 96.3
Priemerný počet akrov kukurice na výsadbu: 3.7
```

Zdroj: Vlastné spracovanie

5.2 Úloha alokácie portfólia

Téma investícií je široko využívaná za pomoci stochastických optimalizačných prístupov. V táto téma bola skúmaná vo viacerých dielach, napríklad aj v diele A hybrid combinatorial approach to a two-stage stochastic portfolio optimization model with uncertain asset prices (Cui, Bai & Ding, 2020). Dvojstupňové stochastické modely boli popísané v kapitole 3.3. Existujú viaceré prístupy na optimalizáciu a odhad ideálneho portfólia. Alternatívna metóda na hľadanie váh portfólia môže predstavovať minimalizačná funkcia, ktorú bude v našom príklade v kóde predstavovať knižničná funkcia basinhopping. Optimalizačný kód vyberá optimálne váhy portfólia a spolu s nimi vypíše aj očakávaný výnos portfólia a riziko resp. štandardnú odchýlku portfólia. Výpočet úlohy je znázornený na obrázku 6. V tomto príklade budeme pracovať s troma aktívami, kde očakávané výnosy sú 8 %, 12 %, 10 % ročne. Štandardné odchýlky pohybu cien troch spoločností sú 10 %, 15 % a 12 %. V tejto úlohe sa budeme snažiť o optimalizáciu váh jednotlivých cenných papierov za pomoci využitia minimalizácie rizika portfólia.

Po inicializácii očakávaných výnosov a štandardných odchýlok sme vytvorili korelačnú maticu. Pri minimalizácii rizika je nutné mať vytvorenú funkciu na výpočet rizika čo zabezpečuje funkcia portfolio_risk, pričom táto funkcia dostáva ako vstupné argumenty váhy a kovariančnú maticu. Simulácia minimalizácie portfólia je zabezpečovaná pomocou funkcie basinhopping. Táto metóda kombinuje lokálnu optimalizáciu s náhodnými krokmi v priestore váh portfólia. Nachádza sa tu viacero argumentov, ktorými sa dá dodefinovať napríklad dodatočné argumenty, ktoré budú použité pri iteráciách minimalizačného algoritmu. Ďalej sú

tu definované obmedzenia, ktoré určujú aké hodnoty môžu mať optimalizačné parametre (váhy). Počet váh musí byť rovný jednej. Výsledok váh je zaznačený do premennej optimal_weights a vypísaný s očakávaným výnosom a rizikom portfólia. Tento kód minimalizuje riziko pre tri aktíva. Obdobný postup je možné aplikovať aj na reálne aktíva.

Obr. 5: Optimalizácia portfólia pomocou očakávaných výnosov a štandardnej odchýlky

```python
import numpy as np
from scipy.optimize import minimize
from scipy.optimize import basinhopping

# Predpokladané výnosy a štandardné odchýlky jednotlivých aktív
expected_returns = [0.08, 0.12, 0.10]  # Predpokladané očakávané výnosy
std_deviations = [0.10, 0.15, 0.12]     # Štandardné odchýlky výnosov

# Korelačná matica medzi výnosmi aktív (v tomto príklade uvádzame úplne nezávislé aktíva)
correlation_matrix = np.eye(3)          # Jednotková matica

# Funkcia na výpočet rizika portfólia
def portfolio_risk(weights, cov_matrix):
    return np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))

# Funkcia na minimalizáciu rizika portfólia
def min_risk(weights):
    return portfolio_risk(weights, np.diag(std_deviations))

# Funkcia na generovanie náhodných váh portfólia
def generate_random_weights():
    #Generuje náhodné váhy pre aktíva v portfóliu s použitím Dirichletovho rozdelenia.
    return np.round(np.random.dirichlet(np.ones(len(expected_returns)), size=1)[0], 4)

# Konštraint pre sumu váh
constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})

# Simulované minimalizácie rizika portfólia

#metóda basinhopping na minimalizáciu rizika portfólia - kombinuje lokálnu optimalizáciu s náhodnými krokmi
#v priestore váh portfólia, čo zahŕňa stochastický prvok

#minimizer_kwargs={"constraints": constraints} definuje dodatočné argumenty,ktoré budú použité pri volaní interného
#minimizačného algoritmu počas simulácie, constraints je dodatočný parameter - zoznam obmedzení, ktoré určujú, aké hodnoty
#môžu mať optimalizačné parametre (v tomto prípade váhy aktív v portfóliu).
#definujeme, že súčet váh musí byť rovný 1, čo zaručuje, že celková alokácia v portfóliu je 100%.

result = basinhopping(min_risk, x0=generate_random_weights(), minimizer_kwargs={"constraints": constraints})

# Optimalne váhy portfólia
optimal_weights = np.round(result.x, 4)

# Výpočet očakávaného výnosu portfólia
expected_return = np.round(np.dot(optimal_weights, expected_returns), 4)

# Výpočet rizika portfólia (štandardnej odchýlky) zaokrúhlené na 4 desatinné miesta
# Diagonálna matica zo vstupnej matice - np.diag vráti vektor obsahujúci prvky na diagonále tejto matice.
risk = np.round(portfolio_risk(optimal_weights, np.diag(std_deviations)), 4)

print("Optimalne váhy portfólia:", optimal_weights)
print("Očakávaný výnos portfólia:", expected_return)
print("Riziko portfólia (štandardná odchýlka):", risk)

```

```
Optimalne váhy portfólia: [0.4    0.2667 0.3333]
Očakávaný výnos portfólia: 0.0973
Riziko portfólia (štandardná odchýlka): 0.2
```

Zdroj: Vlastné spracovanie

6 Záver

V závere práce sme sa zamerali na štúdium stochastického programovania a jeho aplikáciu v rôznych oblastiach rozhodovania s neistotou. Práca bola zameraná na vysvetlenie základných pravdepodobnostných rozdelení a metód stochastického programovania, s dôrazom na využitie normálneho pravdepodobnostného rozdelenia. V rámci práce sme popísali význam metód ako L-Shaped a Dekompozícia bázy a aplikáciu metódy L-shaped v riešení komplexných rozhodovacích problémov. Okrem toho sme použili prístup Monte Carlo na simuláciu náhodných procesov a odhadovanie hodnôt v stochastických problémoch, čo mi umožnilo analyzovať a riešiť problémy s neistotou. Dôležitou súčasťou práce bolo tiež ukázať praktické príklady, kde boli využité normálne pravdepodobnostné rozdelenie na modelovanie neistoty a rizika. Tieto príklady umožnili ilustrovať a aplikovať teoretické poznatky na reálne situácie, čo posilnilo pochopenie problematiky stochastického programovania. Širokú použiteľnosť stochastického prístupu pri ekonomických aplikáciách vidíme v mnoho dielach ako sú napríklad A stochastic programming approach to multicriteria portfolio optimization (Şakar & Köksalan, 2013) alebo An interactive approach to stochastic programming-based portfolio optimization (Köksalan & Şakar, 2016). Stochastický prístup je vhodné použiť v prípade, kde hodnoty nie sú isté (v prípade neistoty). Toto je ideálne pre prácu s finančnými trhmi, kde cena nie je vopred známa a je tak možné vytvoriť predpovede potenciálnej ceny aktív. Celkovo považujem prácu za dôležitý príspevok k pochopeniu a aplikácii stochastického programovania

v praxi. Verím, že obsiahnuté poznatky a príklady poskytnú čitateľom užitočné nástroje na riešenie rozhodovacích problémov v prostrediach s neistotou a náhodnosťou.

Príspevok bol spracovaný v rámci riešenia grantovej úlohy VEGA – 1/0120/23 „*Environmentálne modely ako nástroj ekologicko-ekonomických rozhodnutí*“.

Literatúra

1. Birge, J.R., & Louveaux, F. (1997). Introduction to Stochastic Programming. Springer-Verlag, New York.
2. Brooks, S. (2002). Markov chain Monte Carlo method and its application. *The Statistician, 47*(1), 69–100.
3. Caiafa, C. F., Solé-Casals, J., Marti-Puig, P., Zhe, S., & Tanaka, T. (2020). Decomposition methods for machine learning with small, incomplete or noisy datasets. *Applied Sciences*, *10*(23), 8481.
4. Cui, T., Bai, R., Ding, S., Parkes, A. J., Qu, R., He, F., & Li, J. (2020). A hybrid combinatorial approach to a two-stage stochastic portfolio optimization model with uncertain asset prices. *Soft Computing*, *24*, 2809-2831.
5. Dupačová, J. (1986). Stability in stochastic programming with recourse. Contaminated distributions. *Stochastic Programming 84 Part I*, 133-144.
6. Kall, P., & Mayer, J. Stochastic linear programming. Models, theory, and computation, 2011. *Springer US, Boston*, *10*, 978-1.
7. Köksalan, M., & Şakar, C. T. (2016). An interactive approach to stochastic programming-based portfolio optimization. *Annals of Operations Research*, *245*, 47-66.
8. Kung, C. C., Li, H., & Lin, R. (2018). Bioenergy strategies under climate change: A stochastic programming approach. *Journal of cleaner production*, *188*, 290-303.
9. Liberti, L., & Kucherenko, S. (2005) Comparison of deterministic and stochastic approaches to global optimization. *International Transactions in Operational Research*, *12*, 263-285.
10. Parganiha, K. (2018). Linear programming with Python and PULP. *International Journal of Industrial Engineering Research and Development*, *9*(3), 1–8.
11. Pereira, G. C., Yoshida, M. I., LeBoulluec, P., Lu, W.-T., Alves, A. P., & Avila, A. F. (2020). Application of artificial intelligence models for predicting time-dependent spring-back effect: The L-shape case study. *Composites Science and Technology*, *199*, 108251.
12. Prékopa, A. (1995). *Stochastic Programming. Optimizations and Programming*. Springer Science & Business Media.
13. Romero, J., & Smith, K. (2016, April). Crop profit optimization for farmers. In *2016 IEEE Systems and Information Engineering Design Symposium (SIEDS)* (pp. 289-291). IEEE.
14. Rudyak, V. Ya. & Lezhnev, E.V. (2020). Stochastic molecular modeling the transport coefficients of rarefied gas and gas nanosuspensions. *Nanosystems: physics, chemistry, mathematics*, *11*(3), 285-293.
15. Silva, A., De Brito, J., & Gaspar, P. L. (2016). *Methodologies for service life prediction of buildings: with a focus on façade claddings*. Springer (pp.67-162).
16. Şakar, C., & Köksalan, M. (2013). A stochastic programming approach to multicriteria portfolio optimization. *Journal of Global Optimization*, 57, 299-314.
17. Thangarajoo, R. G., Reaz, M. B. I., Srivastava, G., Haque, F., Ali, S. H. M., Bakar, A. A. A., & Bhuiyan, M. A. S. (2021). Machine learning-based epileptic seizure detection methods using wavelet and EMD-based decomposition techniques: A review. *Sensors*, *21*(24), 8485.

Analýza mikroexpresií pre detekciu deep fake videí

Microexpression analysis for deep fake video detection

Peter Procházka[1]

Abstrakt
Technológia deep fake, využívajúca pokročilé neurónové siete, predstavuje významnú výzvu v oblasti informačnej bezpečnosti a autenticity digitálnych médií. Tento článok skúma možnosť využitia mikroexpresií, krátkych a nevedomých zmien vo výraze tváre, na detekciu deep fake videí. Naša metodológia zahŕňa detekciu mikroexpresií v reálnych videách pomocou konvolučných neurónových sietí a následnú analýzu rozdielov medzi reálnymi a syntetickými videami. Výsledky ukazujú, že deep fake videá majú výrazné ťažkosti s reprodukciou mikroexpresií, čo vedie k nižšej frekvencii a kratšiemu trvaniu týchto výrazov. Na základe týchto zistení vyvíjame algoritmus, ktorý by efektívne identifikoval deep fake videá s čo možno najvyššou presnosťou. Tento výskum ponúka nový prístup k detekcii syntetických videí a podčiarkuje význam mikroexpresií pri zabezpečení autenticity digitálnych médií.

Kľúčové slová
mikroexpresie, deep fake videá, detekcia, neurónové siete

Abstract
Deep fake technology, utilizing advanced neural networks, presents a significant challenge in the realm of information security and digital media authenticity. This paper explores the potential of using microexpressions, brief and unconscious changes in facial expressions, to detect deep fake videos. Our methodology involves detecting microexpressions in real videos using convolutional neural networks and subsequently analyzing the differences between real and synthetic videos. The results indicate that deep fake videos have significant difficulties reproducing microexpressions, leading to lower frequency and shorter duration of these expressions. Based on these findings, we are developing an algorithm that would effectively identify deep fake videos with the highest possible accuracy. This research offers a novel approach to detecting synthetic videos and underscores the importance of microexpressions in ensuring the authenticity of digital media.

Key words
microexpressions, deep fake videos, detection, neural networks

JEL classification
C63, O33

1 Úvod

V posledných rokoch sa technológia deep fake stala jednou z najdiskutovanejších tém v oblasti informačnej bezpečnosti a digitálnej etiky. Deep fake videá, ktoré sú vytvárané pomocou pokročilých algoritmov hlbokého učenia, sú schopné generovať realistické falošné videá, ktoré môžu byť použité na rôzne účely - od zábavy až po vážnejšie hrozby, ako sú politická manipulácia, podvody a podobne.

[1] Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1, 852 35 Bratislava, peter.prochazka@euba.sk

Jedným z hlavných problémov spojených s deep fake videami je ich potenciál na šírenie dezinformácií. Vzhľadom na to, že tieto videá môžu byť veľmi realistické, môže byť pre divákov ťažké rozoznať, čo je pravdivé a čo je falošné. To môže viesť k šíreniu falošných správ a manipulácii verejnej mienky. Navyše, deep fake videá môžu byť použité na vydieranie, poškodzovanie reputácie alebo na vytváranie falošných dôkazov v právnych prípadoch.

Ďalším aspektom, ktorý zvyšuje naliehavosť riešenia problému deep fake videí, je ich dostupnosť. V súčasnosti existuje množstvo nástrojov a softvérov, ktoré umožňujú aj laikom vytvárať deep fake videá s relatívne vysokou kvalitou. To znamená, že riziko zneužitia tejto technológie je vysoké a môže mať ďalekosiahle následky pre jednotlivcov aj spoločnosť.

Práve tieto potenciálne zneužitia deep fake technológie vytvárajú naliehavú potrebu vyvinúť účinné metódy na detekciu a overovanie autenticity multimediálnych materiálov.

Významným faktorom v hodnotení autentickosti videí môžu byť mikroexpresie. Mikroexpresie, ako krátke a nevedomé prejavy emócií, poskytujú jedinečný pohľad na skutočný emocionálny stav človeka. Tieto jemné zmeny vo výraze tváre sú výsledkom rýchlych svalových pohybov a trvajú len niekoľko milisekúnd. Vďaka svojej krátkej trvanlivosti a jemným detailom sú mikroexpresie ťažko reprodukovateľné aj pre najpokročilejšie deep fake technológie. Analýza týchto mikroexpresií preto predstavuje sľubnú metódu na odhalenie syntetických videí, ktoré môžu inak pôsobiť veľmi realisticky.

Cieľom tohto článku je podrobne preskúmať možnosti využitia mikroexpresií na detekciu deep fake videí. V prvej časti článku sa zameriame na teoretické pozadie deep fake technológií a mikroexpresií. Následne budeme diskutovať o cieľoch nášho výskumu a metodológii, ktorú sme použili na detekciu mikroexpresií a analýzu ich rozdielov medzi reálnymi a syntetickými videami. V experimentálnej časti článku predstavíme výsledky našich testov a vyhodnotíme výkonnosť vyvinutého detekčného algoritmu. Nakoniec budeme diskutovať o interpretácii výsledkov, porovnaní s existujúcimi metódami, obmedzeniach našej štúdie a potenciálnych budúcich smerovaniach výskumu.

2 Súčasný stav a teoretické pozadie

Existuje niekoľko nástrojov a softvérových platforiem, ktoré umožňujú aj laikom jednoducho generovať deep fake videá. Tieto nástroje sú často dostupné online a ponúkajú rôzne úrovne funkcií a prispôsobenia. Medzi najznámejšie patria:

1. FaceApp: Táto aplikácia je široko používaná na úpravu fotografií a videí. Používa neurónové siete na transformáciu tvárí, čo zahŕňa zmeny veku, pohlavia, účesu a ďalších atribútov. Hoci je primárne určená na zábavné účely, technológia použitá v aplikácii je veľmi pokročilá.
2. Reface: Je ďalšia populárna aplikácia, ktorá umožňuje používateľom nahradiť tváre v krátkych videách a GIF-och. Aplikácia využíva technológiu syntézy obrazu na vytváranie veľmi realistických deep fake videí v reálnom čase.
3. DeepFaceLab: Tento nástroj je jedným z najpokročilejších a najrozšírenejších v komunite tvorcov deep fake videí. Umožňuje detailné úpravy a je široko využívaný na akademické aj neakademické účely. Používateľom umožňuje vytvárať a trénovať vlastné modely na syntézu tvárí.
4. Faceswap: Je open-source projekt, ktorý je podobný DeepFaceLab. Tento nástroj je prístupný pre používateľov všetkých úrovní znalostí a poskytuje detailné návody a podporu pre vytváranie deep fake videí. Umožňuje výmenu tvárí vo videách s vysokou presnosťou.
5. Zao: Je čínska aplikácia, ktorá sa stala veľmi populárnou vďaka svojej schopnosti generovať realistické deep fake videá v priebehu niekoľkých sekúnd. Používatelia môžu nahrávať svoje fotografie a aplikácia automaticky vytvorí video s ich tvárou na mieste známych hercov alebo postáv.

6. MyHeritage Deep Nostalgia: Táto služba je špeciálne navrhnutá na oživenie starých rodinných fotografií. Používa technológiu na animáciu tvárí na fotografiách, čo vytvára dojem, že osoby na fotografiách sa pohybujú a usmievajú. Hoci je primárne určená na genealogické účely, technológia za ňou je veľmi pokročilá a môže byť zneužitá na tvorbu deep fake videí.
7. Avatarify: Umožňuje používateľom používať živé deep fake technológie na videokonferencie. Tento nástroj môže byť použitý na transformáciu tvárí v reálnom čase, čo umožňuje používateľom napríklad predstierať, že sú niekým iným počas videohovorov.
8. Deep Art Effects: Tento nástroj umožňuje používateľom aplikovať umelecké štýly na fotografie a videá, pričom využíva technológie hlbokého učenia. Hoci je primárne zameraný na kreatívne aplikácie, môže byť použitý aj na generovanie deep fake videí s vysokou mierou realizmu.
9. AvengeThem: Táto webová aplikácia umožňuje používateľom nahrávať svoje fotografie a vytvárať videá, kde sa ich tvár objavuje na postavách z populárnych filmov, najmä z Marvel Cinematic Universe. Hoci je aplikácia vytvorená na zábavné účely, používa technológie na výmenu tvárí, ktoré môžu byť zneužité na vytváranie deep fake videí.
10. Synthesia: Je nástroj, ktorý umožňuje tvorbu videí, kde avatar alebo osoba na videu hovorí text, ktorý používateľ zadá. Táto technológia je veľmi užitočná na tvorbu marketingových a edukačných videí, ale môže byť zneužitá na vytváranie falošných výpovedí alebo dezinformačných videí.
11. JigSaw: Je experimentálny nástroj od spoločnosti Google, ktorý umožňuje vytvárať deep fake videá na účely výskumu a vývoja detekčných metód. Hoci je jeho dostupnosť obmedzená, poskytuje výkonné nástroje na generovanie syntetických médií.
12. FakeApp: Bola jednou z prvých aplikácií na tvorbu deep fake videí, ktorá sa stala populárnou. Umožňuje používateľom vytvárať deep fake videá pomocou priateľského používateľského rozhrania. Aplikácia vyžaduje základné technické znalosti, ale poskytuje vysokú kvalitu výstupov.
13. Deep Video Portraits: Tento nástroj umožňuje upravovať výrazy tváre a pohyby hlavy v existujúcich videách. Používa technológie hlbokého učenia na vytváranie realistických animácií tváre, čo môže byť využité na tvorbu deep fake videí.
14. Deepfakes web β: Je to online platforma, ktorá umožňuje používateľom vytvárať deep fake videá bez potreby sťahovania softvéru. Táto platforma je navrhnutá tak, aby bola prístupná pre širokú verejnosť, a poskytuje relatívne jednoduché rozhranie na tvorbu falošných videí.
15. DeepFaceKit: Tento nástroj je navrhnutý pre pokročilých používateľov a výskumníkov, ktorí chcú vytvárať a študovať deep fake videá. Je to open-source projekt, ktorý ponúka množstvo funkcií na detailné úpravy a tréning modelov.
16. SimSwap: Je ďalší pokročilý nástroj, ktorý umožňuje používateľom vymieňať tváre vo videách s vysokou presnosťou. Tento nástroj je využívaný najmä v akademickom výskume a pri vývoji nových technológií na detekciu deep fake videí.

Použitie týchto nástrojov a softvérových platforiem predstavuje významné riziko pre šírenie dezinformácií a potrebu vyvinúť efektívne detekčné metódy na overovanie autenticity digitálnych médií. Je nevyhnutné, aby sa odborníci na informačnú bezpečnosť, vývojári a vedci zamerali na vytváranie a zdokonaľovanie technológií, ktoré budú schopné identifikovať a zabrániť šíreniu deep fake videí.

Deep fake technológie využívajú generatívne protivnícke siete (Generative Adversarial Networks, GANs) na vytváranie realistických falošných videí (Nguyen et al., 2019). Tieto siete pozostávajú z dvoch komponentov: generátora a diskriminátora (Goodfellow et al., 2014). Generátor vytvára syntetické obrazy, zatiaľ čo diskriminátor sa snaží odlíšiť tieto syntetické obrazy od skutočných. Tieto dve siete sa navzájom trénujú v protivníckom nastavení, čím generátor postupne zlepšuje kvalitu svojich výstupov.

Generatívne protivnícke siete sa ukázali byť mimoriadne efektívne pri vytváraní realistických obrazov a videí, pretože generátor sa neustále učí vytvárať obrazy, ktoré sú čoraz ťažšie odlíšiteľné od skutočných. Táto technológia bola pôvodne vyvinutá pre legitímne aplikácie, ako je tvorba realistických herných postáv a zlepšovanie kvality obrazov, avšak rýchlo sa rozšírila aj do menej etických oblastí.

S pokrokom v deep fake technológiách sa stali tieto videá čoraz sofistikovanejšími a ťažšie detekovateľnými. Napriek tomu však existujú určité obmedzenia. Napríklad, deep fake algoritmy často zápasia s reprodukciou jemných detailov, ako sú prirodzené pohyby očí, rýchle zmeny vo výraze tváre (mikroexpresie) a synchronizácia zvuku a obrazu. Tieto obmedzenia môžu byť využité na detekciu falošných videí.

Mikroexpresie: definícia a význam

Ako sme už uviedli, mikroexpresie sú krátke, nevedomé zmeny vo výraze tváre, ktoré trvajú len niekoľko milisekúnd. Sú výsledkom rýchlych a nevedomých pohybov svalov tváre, ktoré odrážajú skutočné emocionálne stavy človeka. Mikroexpresie boli prvýkrát detailne opísané psychológom Paulom Ekmanom a jeho kolegami, ktorí vyvinuli systém Facial Action Coding System (FACS) na analýzu a kódovanie týchto výrazov (Ekman & Friesen, 1978).

Mikroexpresie sú dôležité, pretože sú považované za autentické prejavy skutočných emócií. Zatiaľ čo makroexpresie, ktoré trvajú dlhšie, môžu byť ľahko kontrolované a manipulované, mikroexpresie sú príliš rýchle na vedomé ovládanie a preto poskytujú presnejší obraz o emocionálnom stave človeka. Tieto jemné zmeny vo výraze tváre môžu byť kľúčové pre detekciu nepravdivých výpovedí a identifikáciu skutočných emocionálnych reakcií.

V kontexte deep fake videí sú mikroexpresie zaujímavé tým, že ich reprodukcia je pre algoritmy hlbokého učenia veľmi náročná. Reálne mikroexpresie sú výsledkom zložitých fyziologických procesov a jemných svalových pohybov, ktoré sú ťažko simulovateľné. Preto môže analýza mikroexpresií poskytnúť účinný nástroj na odhalenie deep fake videí. Aj podľa (Ekman, 2009) deep fake videá majú výrazné ťažkosti s reprodukciou mikroexpresií.

Prehľad súčasných metód detekcie deep fake videí

Súčasné metódy detekcie deep fake videí sa zameriavajú na rôzne aspekty syntetických médií. Korshunov a Marcel (2018) diskutujú o rastúcej hrozbe technológie deep fake pre systémy rozpoznávania tváre a navrhujú rôzne detekčné techniky. Nguyen et al. (2019) poskytujú komplexný prehľad o tvorbe a detekcii deep fake pomocou metodológií hlbokého učenia. Tariq et al. (2018) sa zameriavajú na detekciu falošných obrazov tvárí vytvorených strojom aj človekom v reálnom prostredí, pričom predstavujú inovatívne prístupy na zvýšenie presnosti detekcie. Uvedené tímy autorov sa venujú napríklad analýze vizuálnych a zvukových artefaktov (Korshunov & Marcel, 2018), detekcii nesynchronizácie medzi obrazom a zvukom (Tariq et al., 2018), a využitiu strojového učenia na identifikáciu syntetických vzorov (Nguyen et al., 2019).

Niektoré z najbežnejších techník zahŕňajú:

1. Analýzu vizuálnych artefaktov:
   - Deep fake videá často obsahujú vizuálne chyby, ako sú nepresnosti v rozlíšení, deformácie tváre a neadekvátne osvetlenie. Tieto artefakty môžu byť identifikované algoritmami na analýzu obrazu.
   - Techniky na detekciu neprirodzených pohybov očí a tváre, môžu odhaliť nekonzistentné pohyby.
2. Detekciu anomálií vo zvuku:
   - Zvukové stopy v deep fake videách môžu vykazovať neautentické frekvenčné a amplitúdové charakteristiky. Analýza zvuku teda môže pomôcť identifikovať tieto nezrovnalosti.
   - Algoritmy tiež môžu analyzovať časovú synchronizáciu medzi obrazom a zvukom a zistiť nesúlad.
3. Využitie strojového učenia:
   - Hlboké neurónové siete môžu byť trénované na veľkých datasetoch obsahujúcich reálne a deep fake videá. Tieto modely sa učia rozlišovať medzi reálnymi a syntetickými videami na základe jemných rozdielov.
   - Transfer learning, kde sa modely trénované na súvisiacich úlohách adaptujú na detekciu deep fake videí, môže zvýšiť efektivitu a presnosť detekcie.
4. Kombinované prístupy:
   - Niektoré pokročilé prístupy kombinujú viacero techník na zvýšenie presnosti detekcie. Napríklad, kombinácia vizuálnej a zvukovej analýzy môže poskytnúť robustnejšie výsledky.

Každá z týchto metód má svoje výhody a nevýhody. Vizuálna analýza môže byť veľmi efektívna, ale je citlivá na vizuálne efekty a úpravy. Tieto môžu byť zámerne pridané do videa, aby zamaskovali stopy manipulácie. Napríklad, rôzne filtre, efekty a korekcie farieb môžu skryť nedokonalosti alebo artefakty, ktoré by inak mohli odhaliť, že video bolo falošné. Analýza zvuku je užitočná, ale môže byť obmedzená kvalitou zvukovej stopy. Metódy strojového učenia sú veľmi silné, ale vyžadujú veľké množstvo dát na tréning. Kombinované prístupy môžu ponúknuť najlepšie výsledky, ale sú tiež najkomplexnejšie a najnáročnejšie na implementáciu.

3 Ciele výskumu

Cieľom prvého kroku výskumu je vyvinúť spoľahlivé metódy na detekciu mikroexpresií v reálnych videách. Tieto metódy musia byť schopné zachytiť rýchle a jemné zmeny vo výraze tváre, ktoré trvajú len niekoľko milisekúnd. K tomu budeme potrebovať pokročilé techniky analýzy obrazu a strojového učenia, ktoré dokážu identifikovať a klasifikovať mikroexpresie na základe ich charakteristických znakov.

Pre tento účel plánujeme využiť konvolučné neurónové siete (CNN), ktoré sa ukázali byť mimoriadne efektívne pri analýze obrazových dát. Tieto siete budú trénované na rozsiahlych datasetoch obsahujúcich reálne videá s označenými mikroexpresiami. Cieľom je vytvoriť model, ktorý dokáže presne detekovať mikroexpresie v rôznych podmienkach a situáciách.

Po úspešnej identifikácii mikroexpresií v reálnych videách budeme skúmať rozdiely medzi mikroexpresiami v reálnych a deep fake videách. Tento krok zahŕňa štatistickú analýzu frekvencie, trvania a typu mikroexpresií v oboch kategóriách videí. Našim cieľom je identifikovať charakteristické znaky, ktoré sú prítomné v reálnych videách, ale chýbajú alebo sú nesprávne reprodukované v deep fake videách.

Na túto analýzu budeme používať rôzne štatistické metódy a techniky vizualizácie dát, aby sme mohli presne kvantifikovať a interpretovať zistené rozdiely. Predpokladáme, že deep

fake videá budú vykazovať nižšiu frekvenciu a kratšie trvanie mikroexpresií, ako aj vyššiu mieru nepresností a artefaktov v týchto výrazoch.

Na základe zistených rozdielov vyvinieme algoritmus, ktorý bude schopný efektívne detekovať deep fake videá pomocou analýzy mikroexpresií. Tento algoritmus bude kombinovať techniky strojového učenia a štatistickej analýzy na identifikáciu syntetických videí.

Algoritmus bude trénovaný na datasetoch obsahujúcich reálne a deep fake videá s označenými mikroexpresiami. Jeho výkon bude testovaný na nezávislej testovacej sade, aby sa overila jeho presnosť, precíznosť, a schopnosť detekovať deep fake videá v rôznych podmienkach. Cieľom je vytvoriť robustný a spoľahlivý nástroj na detekciu falošných videí, ktorý bude schopný pracovať v reálnom čase a poskytovať vysoko presné výsledky.

4 Metodológia

Prvým krokom v našej metodológii je zostavenie rozsiahleho datasetu obsahujúceho reálne a deep fake videá. Tento dataset musí byť dostatočne diverzifikovaný, aby zahŕňal rôzne osoby, emocionálne stavy a situácie. Videá budú získané z verejne dostupných zdrojov, ako sú video platformy, sociálne siete a databázy pre výskumné účely.

Dataset bude rozdelený na tréningovú, validačnú a testovaciu sadu. Tréningová sada bude použitá na trénovanie modelov, validačná sada na ladenie hyperparametrov a testovacia sada na vyhodnotenie výkonnosti finálneho modelu. Aby sme zabezpečili vysokú kvalitu a relevantnosť dát, každé video bude manuálne prekontrolované a označené expertmi na mikroexpresie.

Na detekciu mikroexpresií použijeme konvolučné neurónové siete (CNN), ktoré sú schopné analyzovať obrazové dáta a identifikovať jemné detaily vo výraze tváre. CNN budú trénované na označených datasetoch obsahujúcich mikroexpresie, pričom sa budú učiť rozpoznávať charakteristické vzory týchto jemných zmien.

Architektúra CNN bude optimalizovaná na detekciu rýchlych a krátkodobých zmien vo výraze tváre. Použijeme rôzne techniky, ako sú augmentácia dát a regulácia, aby sme zvýšili robustnosť a generalizáciu modelu. Výstupy modelu budú mikroexpresie klasifikované podľa ich typu a trvania, ktoré budú následne použité na ďalšiu analýzu.

Po detekcii mikroexpresií budeme analyzovať rozdiely medzi reálnymi a deep fake videami. Tento krok zahŕňa štatistickú analýzu frekvencie, trvania a typu mikroexpresií v oboch kategóriách videí. Použijeme rôzne štatistické metódy, ako sú t-testy, ANOVA a regresné analýzy, aby sme identifikovali signifikantné rozdiely.

Výsledky tejto analýzy budú vizualizované pomocou grafov a diagramov, ktoré nám umožnia lepšie pochopiť a interpretovať zistené rozdiely. Predpokladáme, že deep fake videá budú vykazovať nižšiu frekvenciu a kratšie trvanie mikroexpresií, ako aj vyššiu mieru nepresností a artefaktov.

Na základe zistených rozdielov vyvinieme algoritmus, ktorý bude schopný efektívne detekovať deep fake videá. Tento algoritmus bude kombinovať techniky strojového učenia a štatistickej analýzy na identifikáciu syntetických videí.

Algoritmus bude trénovaný na tréningových dátach a jeho výkon bude testovaný na testovacej sade. Budeme používať rôzne metriky, ako sú presnosť, precíznosť, recall a F1-score, aby sme vyhodnotili jeho výkonnosť. Naším cieľom je vytvoriť robustný a spoľahlivý nástroj na detekciu falošných videí, ktorý bude schopný pracovať v reálnom čase a poskytovať vysoko presné výsledky.

5 Experimenty a výsledky

Naše experimentálne nastavenie zahŕňa použitie výkonného hardvéru a softvéru na spracovanie a analýzu veľkých množstiev videodát. Používame výkonné grafické karty (GPU)

na urýchlenie tréningu konvolučných neurónových sietí a ďalšie výpočtovo náročné úlohy. Softvérové prostredie zahŕňa populárne nástroje a knižnice pre strojové učenie, ako sú TensorFlow a PyTorch.

V prvom kroku predspracúvame videodáta, ktoré zahŕňajú segmentáciu videí na jednotlivé snímky, normalizáciu obrazu a označenie mikroexpresií. Tieto predspracované dáta budú následne použité na tréning a testovanie našich modelov.

Po trénovaní konvolučných neurónových sietí na detekciu mikroexpresií predpokladáme dosiahnutie vysokej presnosti detekcie. Naša metóda by teda mala byť schopná spoľahlivo identifikovať mikroexpresie v reálnych videách s dostatočne vysokou presnosťou. Model by mal byť schopný klasifikovať rôzne typy mikroexpresií a analyzovať ich trvanie a frekvenciu.

Predpokladáme, že štatistická analýza rozdielov medzi reálnymi a deep fake videami ukáže signifikantné rozdiely vo frekvencii a trvaní mikroexpresií. Deep fake videá by mali vykazovať nižšiu frekvenciu mikroexpresií a kratšie trvanie týchto výrazov. Okrem toho by mali vykazovať vyššiu mieru nepresností a artefaktov v deep fake videách, čo by potvrdilo naše predpoklady o ťažkostiach reprodukcie mikroexpresií pomocou deep fake technológií.

Vyvinutý detekčný algoritmus testovaný na nezávislej testovacej sade by mal dosiahnuť vysokú presnosť a spoľahlivosť, pričom by mal vykazovať nízku mieru falošných pozitív a negatív.

6 Diskusia

Výsledky nášho doterajšieho výskumu naznačujú, že mikroexpresie môžu byť spoľahlivým indikátorom autenticity videí. Zistili sme, že deep fake videá majú výrazné ťažkosti s reprodukciou týchto jemných a rýchlych výrazov tváre, čo vedie k ich zníženej frekvencii a kratšiemu trvaniu. Tieto zistenia podporujú teóriu, že analýza mikroexpresií môže odhaliť syntetické videá, ktoré by inak mohli oklamať bežné detekčné metódy.

Naša metóda analýzy mikroexpresií poskytuje niekoľko výhod oproti existujúcim metódam detekcie deep fake videí. Vizuálna a zvuková analýza môže byť účinná, ale často zápasí s vysokou mierou falošných pozitív a negatív. Strojové učenie je veľmi silné, ale vyžaduje veľké množstvo dát na tréning. Naša metóda kombinuje výhody týchto prístupov a zameriava sa na jemné detaily, ktoré sú ťažko simulovateľné, čím poskytuje robustnejší a presnejší nástroj na detekciu.

Napriek doterajším pozitívnym výsledkom naša štúdia má niekoľko obmedzení. Naša analýza je závislá na kvalite a diverzite datasetu. Ak by dataset nebol dostatočne reprezentatívny, mohlo by to ovplyvniť výsledky. Ďalej, náš algoritmus bude trénovaný a testovaný na určitých typoch videí a situácií, takže jeho výkon v iných kontextoch môže byť odlišný. Budúci výskum by mal zahŕňať širšie spektrum videí a emocionálnych stavov, aby sa zabezpečila robustnosť a generalizácia modelu.

7 Budúce smerovanie a odporúčania

Naša štúdia poskytne základ pre ďalší výskum, ale existuje niekoľko oblastí, kde by bolo možné metódy zlepšiť. Napríklad, pokročilejšie techniky strojového učenia a hlbokého učenia by mohli zvýšiť presnosť a spoľahlivosť detekcie. Použitie hybridných modelov, ktoré kombinujú viaceré detekčné techniky, by mohlo tiež zvýšiť robustnosť systému.

Okrem detekcie deep fake videí by naša metóda analýzy mikroexpresií mohla nájsť uplatnenie aj v iných oblastiach. V psychológii a behaviorálnej vede by analýza mikroexpresií mohla pomôcť lepšie pochopiť emocionálne reakcie ľudí. V oblasti bezpečnosti a forenznej analýzy by mohla byť použitá na identifikáciu falošných výpovedí a odhalenie podvodov.

Výskum v oblasti detekcie deep fake videí si vyžaduje spoluprácu odborníkov z rôznych disciplín. Psychológovia, odborníci na strojové učenie, počítačoví vedci a forenzní analytici by mali spolupracovať na vývoji a zdokonaľovaní metód detekcie. Interdisciplinárny prístup môže

potom priniesť nové a inovatívne riešenia, ktoré budú schopné čeliť tejto rýchlo sa vyvíjajúcej hrozbe.

8 Záver

Tento článok skúma možnosti využitia mikroexpresií na detekciu deep fake videí. Na základe našich experimentov sme zistili, že mikroexpresie sú spoľahlivým indikátorom autenticity videí, pretože deep fake technológie majú výrazné ťažkosti s ich reprodukciou.

Výskum analýzy mikroexpresií predstavuje nový a sľubný prístup k detekcii deep fake videí. Naše výsledky naznačujú, že táto metóda môže poskytnúť účinný nástroj na boj proti syntetickým videám a prispieť k ochrane verejnosti pred dezinformáciami a manipuláciou. Budúci výskum by mal pokračovať v zdokonaľovaní týchto metód a ich aplikácií v rôznych oblastiach.

Literatúra

1. Ekman, P. (2009). *Telling lies: Clues to deceit in the marketplace, politics, and marriage*. W. W. Norton & Company.
2. Ekman, P., & Friesen, W. V. (1978). *Facial Action Coding System: A technique for the measurement of facial movement*. Consulting Psychologists Press.
3. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative adversarial nets. In Advances in Neural Information Processing Systems* (pp. 2672-2680).
4. Korshunov, P., & Marcel, S. (2018). *DeepFakes: A new threat to face recognition? Assessment and detection.* arXiv preprint arXiv:1812.08685. https://doi.org/10.48550/arXiv.1812.08685.
5. Nguyen, T. T., Nguyen, C. M., Nguyen, D. T., Nguyen, D. T., & Nahavandi, S. (2019). *Deep learning for deep fakes creation and detection: A survey.* arXiv preprint arXiv:1909.11573. https://doi.org/10.48550/arXiv.1909.11573.
6. Tariq, S., Lee, S., Kim, H., Shin, Y., & Woo, S. S. (2018*). Detecting both machine and human created fake face images in the wild.* In Proceedings of the 2nd International Workshop on Multimedia Privacy and Security (pp. 81-87). https://doi.org/10.1145/3267357.3267367.

Praktické predpoklady pre tvorbu a implementáciu platforiem dištančného vzdelávania

Practical Prerequisites for the Creation and Implementation of Distance Learning Platforms

Maxot E. Rakhmetov[1], Aigul K. Sadvakassova[1], Peter Schmidt[2]

Abstrakt
Rýchly vývoj informačných a komunikačných technológií si vyžaduje začlenenie flexibilných, webových vzdelávacích prostredí, ktoré sú dostupné bez geografických obmedzení. Táto štúdia skúma efektivitu LMS pri poskytovaní komplexných, adaptabilných a interaktívnych vzdelávacích skúseností vhodných pre inštruktorov aj študentov. Zdôrazňuje použitie štandardných a inovatívnych nástrojov v rámci LMS na zvýšenie produktivity výučby, splnenie vzdelávacích cieľov a podporu pútavého vzdelávacieho prostredia. Ďalej článok pojednáva o vývoji a implementácii prototypu systému riadenia vzdelávania založeného na štandardoch SCORM (Sharable Content Object Reference Model) na ilustráciu praktických aplikácií v dištančnom vzdelávaní. Empirické dôkazy z prieskumov uskutočnených v L.N. Eurázijská národná univerzita Gumilyov a univerzita Atyrau vykazujú významný trend smerom k prijatiu týchto platforiem, čo podčiarkuje ich rastúci význam pri plnení súčasných vzdelávacích požiadaviek a príprave študentov na profesionálnu konkurencieschopnosť v globálnom prostredí. Zistenia obhajujú neustále zlepšovanie online vzdelávacích platforiem, aby sa zabezpečilo, že budú spĺňať dynamické potreby študentov aj pedagógov v digitálnom veku.

Kľúčové slová
LMS, synchrónne učenie, asynchrónne učenie, vzdelávacie online platformy, technológie dištančného vzdelávania, virtuálne platformy

Abstract
The rapid evolution of information and communication technologies necessitates the incorporation of flexible, web-based educational environments that are accessible without geographical constraints. This study explores the effectiveness of LMS in providing comprehensive, adaptable, and interactive educational experiences, suitable for both instructors and students. It highlights the use of standard and innovative tools within LMS to enhance teaching productivity, meet educational objectives, and foster an engaging learning environment. Furthermore, the article discusses the development and implementation of a prototype learning management system based on the SCORM (Sharable Content Object Reference Model) standards to illustrate practical applications in distance education. Empirical evidence from surveys conducted at L.N. Gumilyov Eurasian National University and Atyrau University shows a significant trend towards the adoption of these platforms, underscoring their growing importance in meeting contemporary educational demands and preparing students for professional competitiveness in a global landscape. The findings advocate for continuous improvement of online educational platforms to ensure they meet the dynamic needs of both learners and educators in the digital age.

[1] L.N. Gumilyov Eurasian National University, Nur-Sultan, Kazakhstan, maksot.raxmetov.96@mail.ru

[2] University of Economics in Bratislava, Faculty of Economic Informatics, Department of Applied Informatics, Dolnozemská cesta 1, 852 35 Bratislava, peter.schmidt@euba.sk

Keywords
LMS, synchronous learning, Asynchronous Learning, Education-online platforms, distance learning technologies, virtual platforms

JEL classification
I21, I23

1 Introduction

Currently, one of the pressing issues in the training of future computer science teachers at universities is the necessity to integrate online educational platforms for distance learning. This requirement arises from the need to apply and implement modern technologies in the development of information and communication technologies, which is one of the primary tasks for computer science teachers. Learning Management Systems (LMS) enable students to transmit information both within and outside the classroom, allowing teachers and students to provide tailored instructions that are accessible anytime and anywhere without geographical constraints. Thanks to unique learning and design capabilities, LMS can be implemented for students at each educational level. Most LMS platforms come equipped with a standard set of tools designed to facilitate learning and discussion in an online environment. Some tools, such as discussion forums, are used for relationship-building and collaborative learning. Other tools, such as assessment devices and online evaluations, enhance teacher productivity and ensure that the courses meet educational objectives while also tracking student progress. Additionally, the optimal curriculum offers comprehensive solutions that address all aspects of online learning without the need for supplementary tools.

The Learning Management System (LMS) provides a platform for a web-based learning environment, enabling the management, provision, and monitoring of learning. LMS is often regarded as the foundation for any web-based learning program. An effective LMS should be fully suitable for web deployment without the need for additional client applications. It is also crucial that the LMS supports various sources from different manufacturers and is based on open industry standards for web deployment (Andyusev, 2020), (Tariq & Said, 2023).

We will then explore some effective methods of feedback with resources when creating a learning management system.

2 Methodology

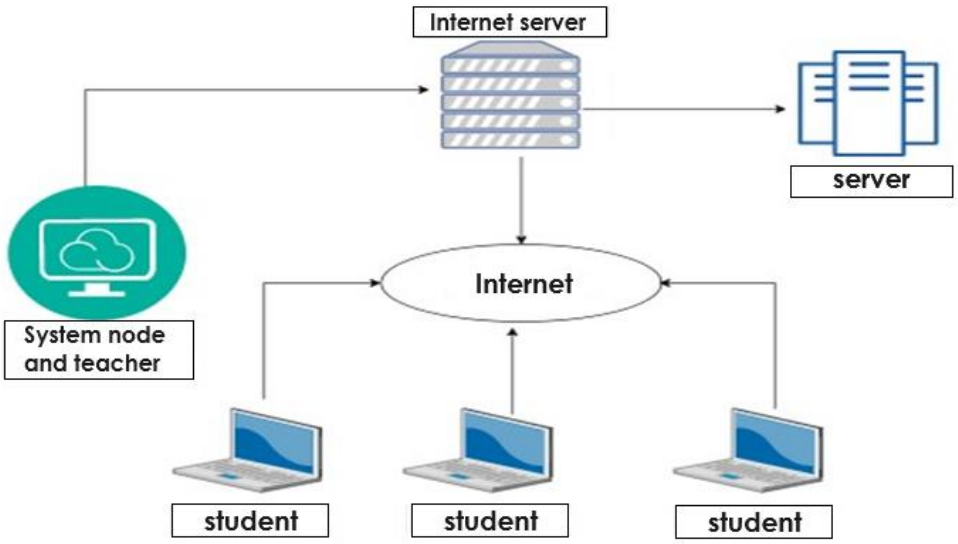
The most effective approach to organizing educational processes through a Learning Management System (LMS) is the use of distance learning platforms. The organizational structure of distance learning is illustrated in Figure 1.

Advantages of Using a Learning Management System for Education:

- Individual Training: Students can set the pace of learning according to their personal preferences and circumstances, allowing for a personalized learning experience.
- Freedom and Flexibility: Learners have the liberty to select from a variety of courses and can independently schedule the timing and duration of their lessons.
- Accessibility: Distance learning platforms enable educational access to any student at any university that supports these technologies, irrespective of geographical constraints. This universality helps to meet diverse educational needs effectively.
- Speed of Interaction: Rapid interaction between students and teachers is crucial and is seamlessly facilitated by LMS, enhancing the educational process.

- Productive Capacity: The use of cutting-edge information and telecommunication technologies in education maximizes the effectiveness of learning outcomes.
- Creativity: LMS provides a conducive environment for students to express themselves creatively within the learning process (Assaf Alfadly, 2013).

Fig. 1: Distance learning process



Source: Authors

Traditional methods of developing online learning platforms are generally costly, time-consuming, and require specific technical skills. For comprehensive online learning implementation, organizations employ Learning Content Management Systems (LCMS) to swiftly arrange, launch, and manage online course content.

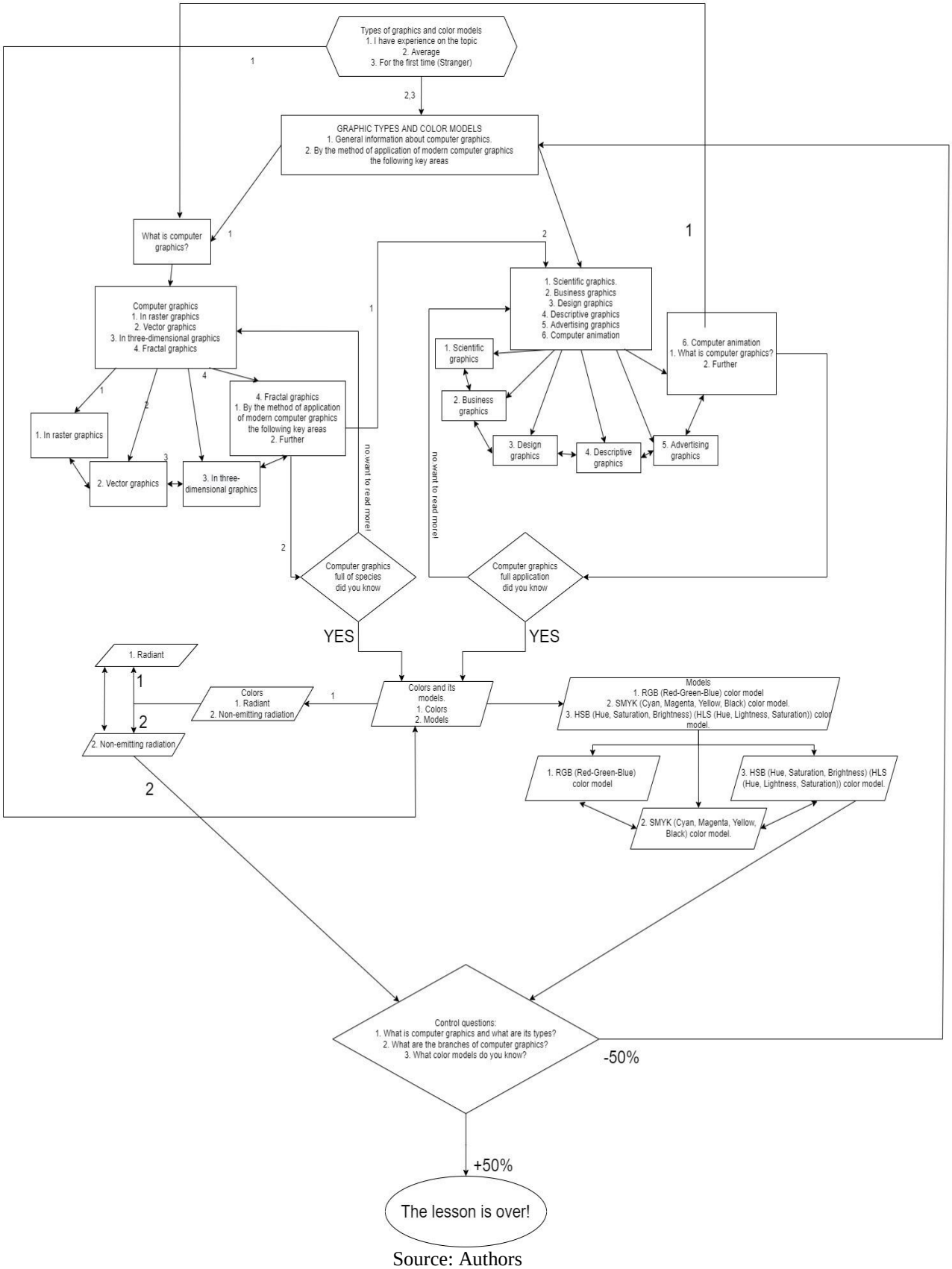
In 2000, the Advanced Distributed Learning (ADL) Initiative Group introduced the SCORM (Sharable Content Object Reference Model) standard for distance learning systems, which outlines requirements for organizing educational content and the entire educational platform system. SCORM, which is XML-based, includes specifications for content storage, execution environments, and search and navigation functionalities (Ruano et al., 2016).

SCORM is a set of recommended specifications and standards it consists of several sections:

- content storage model;
- current execution environment;
- search and navigation.

Consequently, a blueprint for an autonomous interactive learning management system based on SCORM standards was developed for distance learning applications. The schematic of this system is depicted in Figure 2.

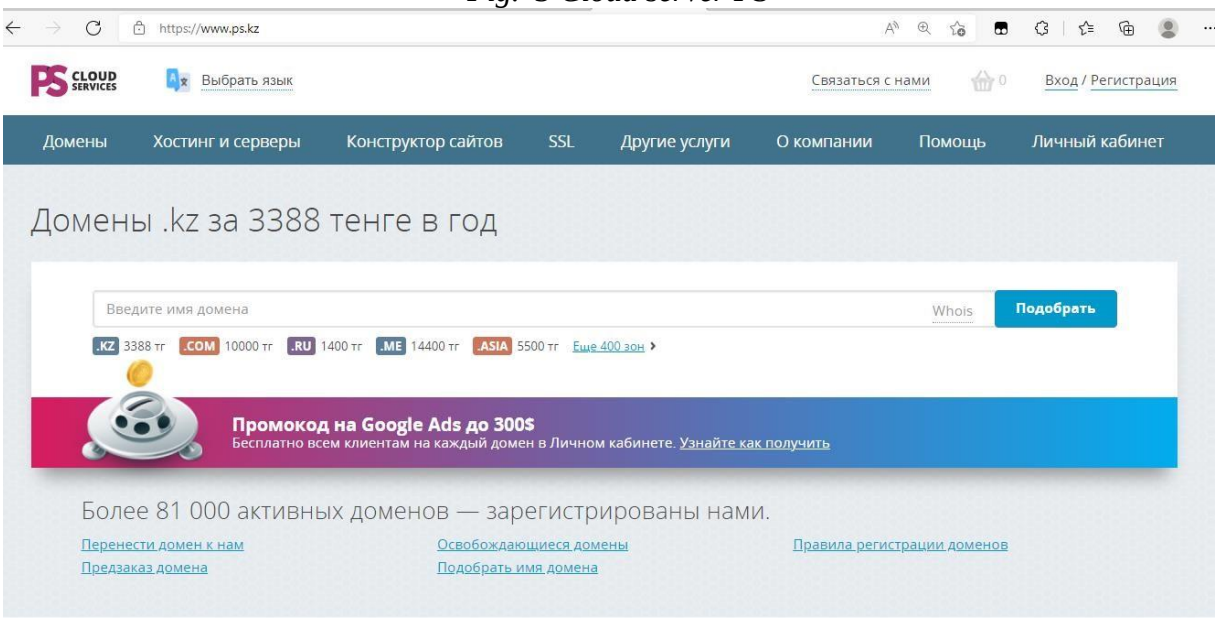
Fig. 2: Autonomous interactive learning management system



Source: Authors

Additionally, to facilitate the creation of a distance learning platform, hosting was secured from a cloud server at www.ps.kz.

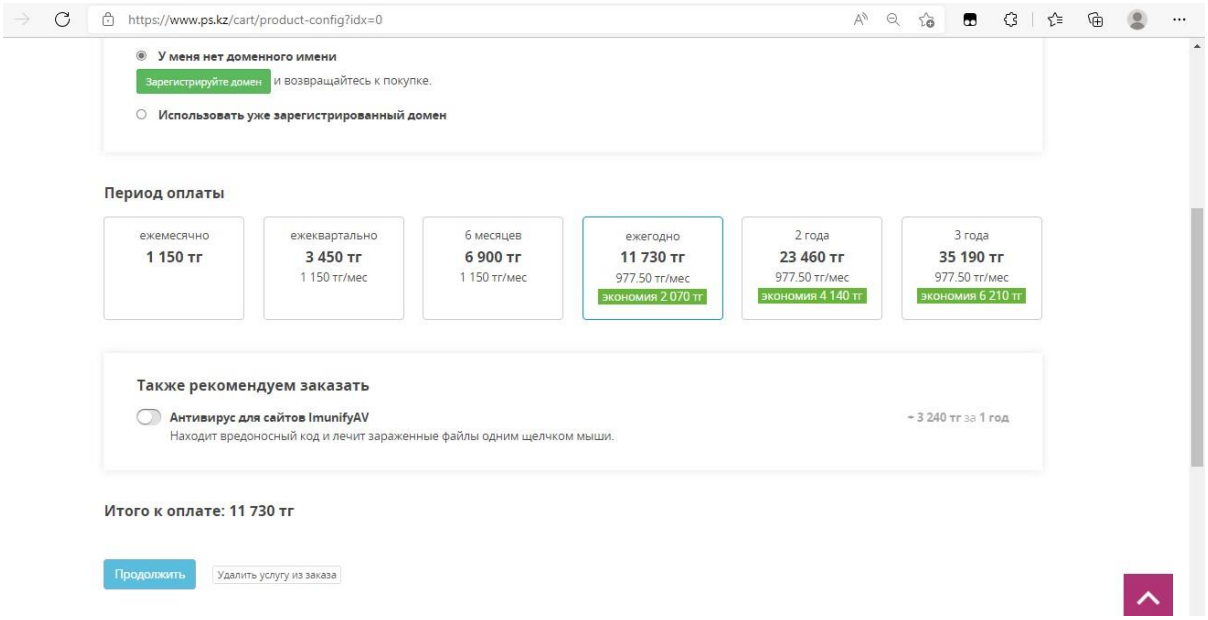
Fig. 3 Cloud server PS



Source: Authors

Registration on the cloud server can be accomplished by clicking the "support" button and entering the required data or using a Gmail account. Selections can also be made based on timing considerations since hosting incurs costs. The capabilities and pricing of the hosting are detailed in Figure 4.

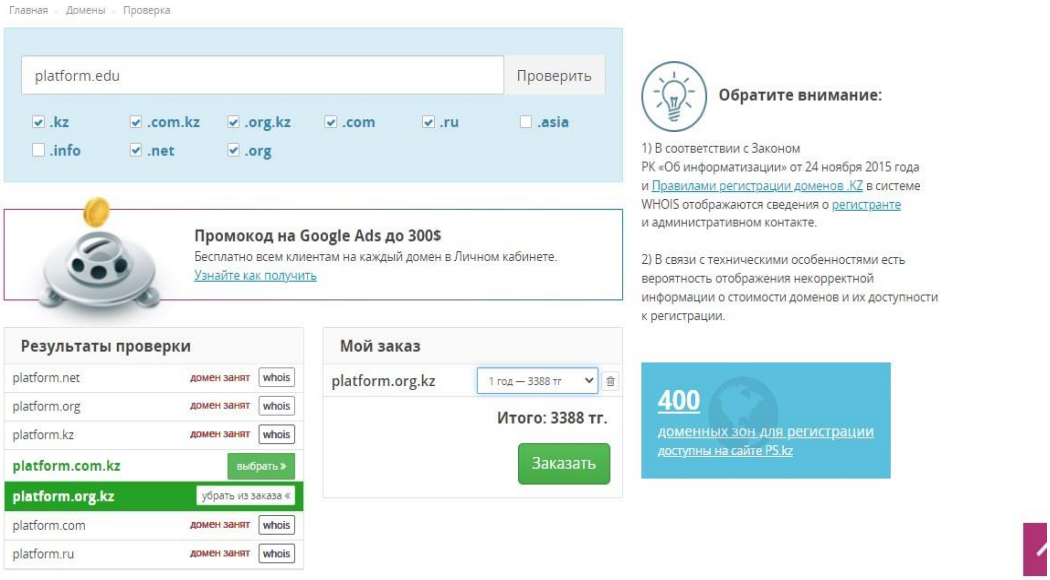
Fig. 4: Hosting capabilities



Source: Authors

For those without an existing domain, requests can be made by specifying the desired domain name. In this instance, the domain platform.org.kz was secured as shown in Figure 5.

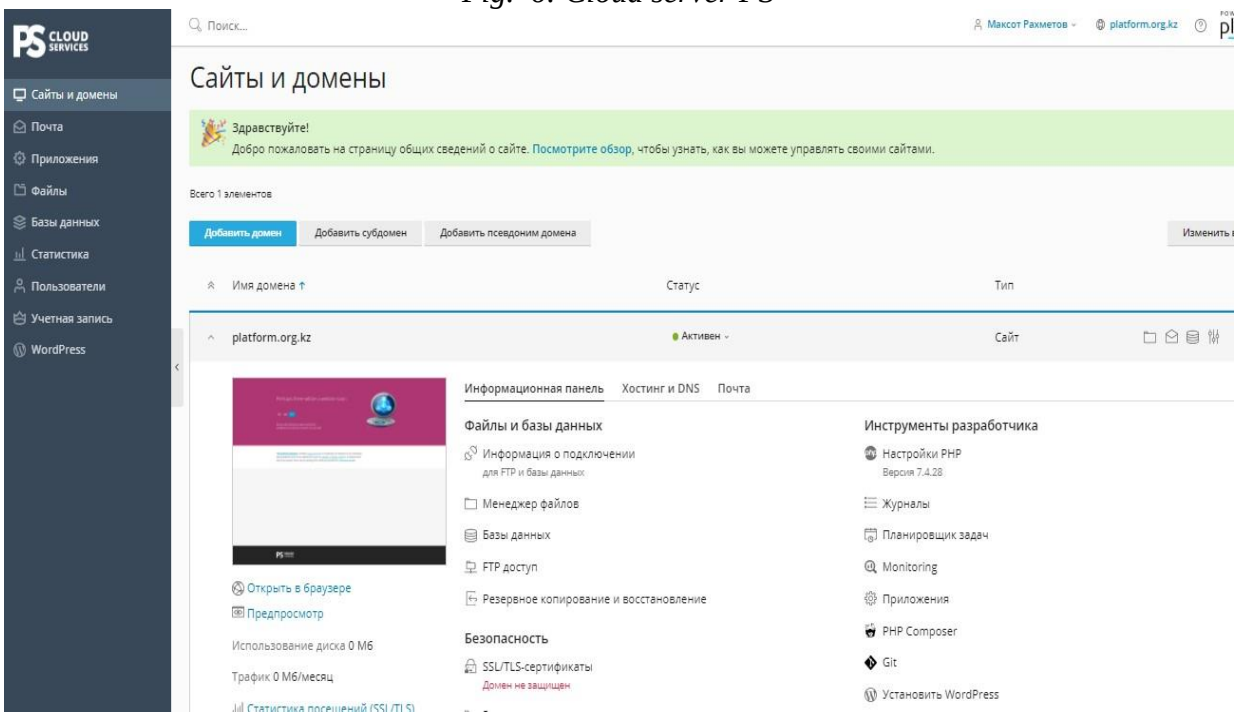
Fig. 5: Getting a domain



Source: Authors

The cloud server offers multiple functionalities for creating platforms and websites, including database creation, email setup, statistics management, user list customization, and WordPress plugin configurations. The operational interface of the PS cloud server is illustrated in Figure 6.

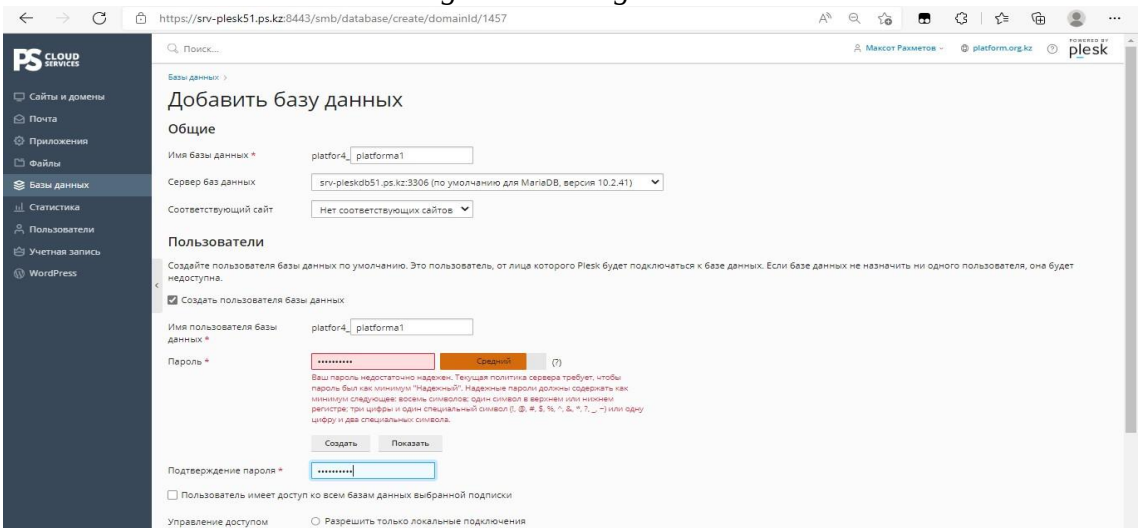
Fig. 6: Cloud server PS



Source: Authors

A database named 'platforma1' was established on the server, secured with a password, as depicted in Figure 7.

Fig. 7: Creating a database



Source: Authors

Following database setup, the latest WordPress 5.9 application model was installed on the cloud server to support the development of learning management system platforms. After logging into WordPress, the wp-admin page associated with our domain is accessible, allowing for future development of the learning management system using specific templates and programming languages tailored to meet technical specifications and diverse educational needs.

3 Distance Learning Platforms: The Effectiveness of Education

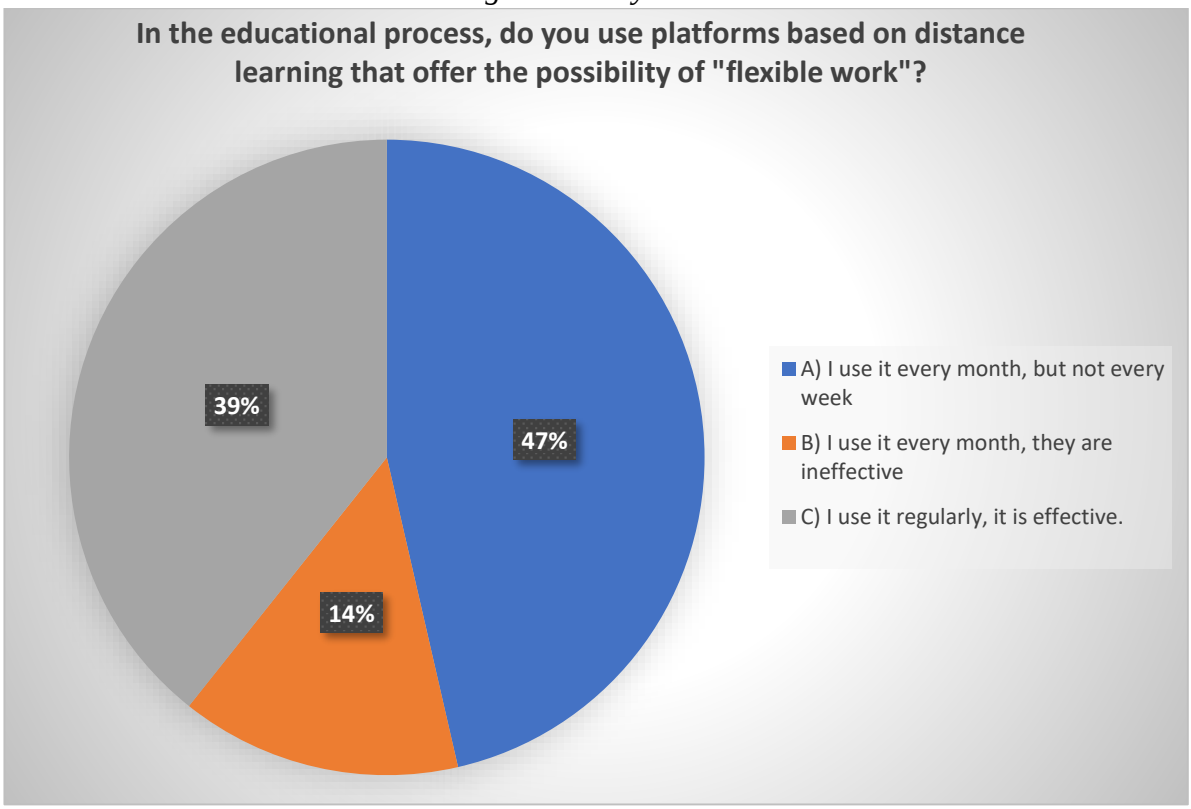
The Learning Management System (LMS) facilitates the provision of online classes for both educators and students within the training process. Engaging in education through online classes enhances the educational experience for instructors. The application of LMS supports the notion that online learning can operate on a non-profit basis (Najm et al., 2018). Internet user engagement increases as the LMS focuses on their achievements, providing flexibility in learning locations to accommodate other commitments such as employment or family. The online learning system allows instructors to independently manage the pace and scheduling of their educational activities (Holmes et al., 2018). Al-Fraihat (Al-Fraihat et al., 2019) pointed out that during the development of central and Constructivist online courses, the emphasis was on directing students selflessly toward their academic pursuits. Bradley emphasized the necessity of enhancing student autonomy and active participation in remote learning. Furthermore, Murcia and Vanga explored the potential of online education through discussion, modeling, and planning (Wu et al., 2010). Collectively, scholars concur that the advancement of the Internet and its tools such as email, chat, forums, and other technologies have enriched the interaction opportunities available to educators and students (Stenetorp et al., 2012).

4 Results and analysis

Educational content for training future computer science teachers has been developed on the newly created platform, alongside organized practical work. This initiative was executed during the educational processes at L.N. Gumilyov Eurasian National University. The utility and effectiveness of the educational platforms were assessed through experimental stages involving students from the departments of "Informatics" at both Gumilyov Eurasian National University and Atyrau University, under the educational programs "Informatics and Information and Communication Technologies in the Education System 6b01511" and "Informatics 7M01511". Based on the data gathered in 2018, it was predicted that the user base for distance education

platforms would expand rapidly. By the end of 2020, 41% of educational institutions globally were utilizing these platforms, as evidenced by survey outcomes. These findings are presented in Figures 8 and 9, which depict the strong inclination among students towards using distance learning platforms and indicate a growing interest in the development of autonomous platforms.

Fig. 8: Survey result



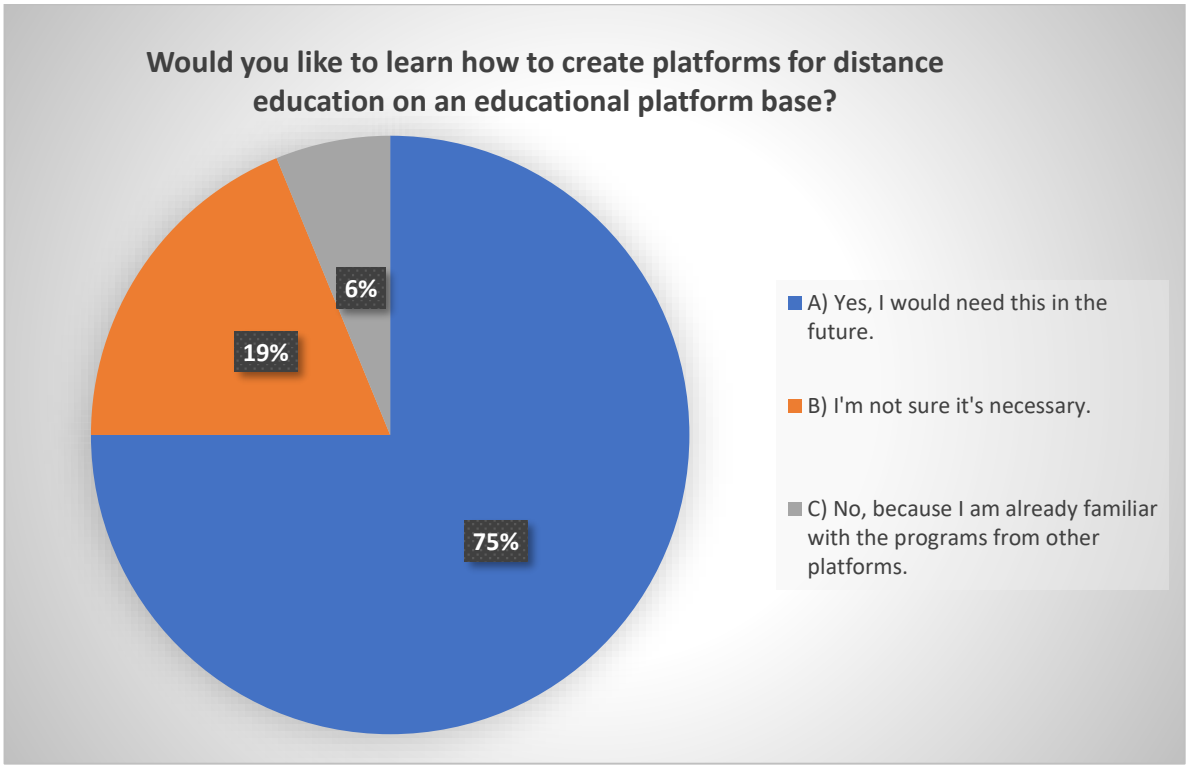
Source: Authors

From Figure 8, it is evident that 47% of respondents regularly use platforms based on distance learning that offer the possibility of "flexible work," demonstrating a high level of effectiveness and regular usage of these tools. An additional 39% indicate that they use such platforms monthly, though not weekly, suggesting less frequent but still consistent engagement. Only 14% of respondents find these platforms ineffective, even though they use them every month.

In Figure 9, a substantial 75% of respondents expressed interest in learning how to create platforms for distance education based on educational platforms, indicating a strong demand for developing these skills for future applications. Only 19% are unsure of the necessity, and 6% already have experience with similar programs from other platforms, they do not see a need for further education in this area.

Integrating these insights into the existing discussion, it becomes clear that distance learning platforms are not only widely used but are also seen as effective and beneficial for a significant portion of the educational community. This underscores the importance of further enhancing these platforms to meet the growing educational needs and preferences for flexible, accessible learning environments. Such data supports the ongoing development and refinement of educational technologies, aligning with the broader goals of increasing accessibility and personalizing the learning experience.

Fig. 9: Survey result



Source: Authors

5 Conclusion

In summation, the development of innovative Learning Management Systems and the integration of information technologies into distance learning processes are poised to significantly bolster student interest in academic disciplines, cultivate a scientific and creative outlook, enhance professional attributes, and produce market-competitive specialists. Consequently, the adoption of distance learning platforms within educational frameworks signifies a shift towards a new educational paradigm that leverages highly efficient technologies for societal advancement, integrates fully into the global educational landscape, and achieves international standards. Our ongoing research into distance education will continue to evaluate the efficacy of distance learning platforms and develop methodological guides that leverage these technologies in the training of future computer science teachers.

References

1. Andyusev, B. E. (2010). The case method as a tool for the formation of competencies. *Director of the School, No. 4.*, 61-69.
2. Al-Fraihat, D., Joy, M., Masa'deh, R., & Sinclair, J. (2019). E-learning systems measure. *PsycTESTS Dataset*. https://doi.org/10.1037/t74215-000.
3. Assaf Alfadly, A. (2013). The efficiency of the "Learning Management System (LMS)" in AOU, Kuwait, as a communication tool in an e-learning system. *International Journal of Educational Management*, *27*(2), 157-169. https://doi.org/10.1108/09513541311297577.
4. Holmes, K., & Prieto - Rodriguez, E. (2018). Student and staff perceptions of a learning management system for blended learning in teacher education. *Australian Journal of Teacher Education*, *43*(3), 21–34. https://doi.org/10.14221/ajte.2018v43n3.2.

5. Najm, E., Nasser, R., & Telatar, E. (2018). Content based status updates. *2018 IEEE International Symposium on Information Theory (ISIT)*. https://doi.org/10.1109/isit.2018.8437577.
6. Ruano, I., Cano, P., Gamez, J., & Gomez, J. (2016). Advanced LMS integration of SCORM Web Laboratories. *IEEE Access*, *4*, 6352-6363. https://doi.org/10.1109/access.2016.2587805.
7. Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., & Tsujii, J. I. (2012, April). BRAT: a web-based tool for NLP-assisted text annotation. In Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics (pp. 102-107).
8. Tariq, O., & Said, A. (2023). Moroccan students' acceptance and use of learning management systems (LMS) platforms to enhance writing skills: Moulay Ismail University as a case study. *World Journal of Advanced Research and Reviews*, *19*(3), 1047-1064. https://doi.org/10.30574/wjarr.2023.19.3.1916.
9. Wu, H., He, J., & Pei, Y. (2010). Scientific impact at the topic level: A case study in computational linguistics. *Journal of the American Society for Information Science and Technology*, *61*(11), 2274-2287. https://doi.org/10.1002/asi.21396.

Testovanie duševného zdravia s využitím Deep Learning

Mental health testing using Deep Learning

Peter Schmidt[1], Veronika Horniaková[2], Peter Procházka[3]

Abstrakt
Tento výskum sa zameriava na vývoj inovatívneho programu na detekciu stavu mentálneho zdravia, špecificky depresie, pomocou metód strojového učenia. Metodika spočíva v monitorovaní mimiky človeka prostredníctvom kamery, pričom reakcie na zobrazované obrázky poskytujú údaje o emočnom stave. Využitím technológií Deep Learningu, systém umožňuje naučiť algoritmus rozpoznávať emočné reakcie a identifikovať potenciálne odchýlky, ktoré môžu naznačovať psychické problémy ako depresiu. Táto technológia má potenciál pre včasnú diagnostiku a intervenciu, čo je kľúčové pre efektívnu liečbu depresie a prispôsobené terapeutické zásahy, čím by sa mohla zlepšiť kvalita života postihnutých jedincov.

Kľúčové slová
Detekcia duševného zdravia, psychiatria, psychológia, strojové učenie, hlboké učenie

Abstract
This research focuses on the development of an innovative program to detect a mental health condition, specifically depression, using machine learning methods. The methodology consists in monitoring a person's facial expressions through a camera, while reactions to displayed images provide data on the emotional state. Using Deep Learning technologies, the system allows the algorithm to be taught to recognize emotional reactions and identify potential deviations that may indicate psychological problems such as depression. This technology has the potential for early diagnosis and intervention, which is crucial for effective treatment of depression and tailored therapeutic interventions, which could improve the quality of life of affected individuals.

Keywords
Mental Health Detection, Psychiatric, Psychology, Machine Learning, Deep Learning

JEL classification
C38, I110

1 Úvod

Tento výskum sa zameriava na vývoj programu na detekciu stavu mentálneho zdravia, s dôrazom na depresiu, pomocou techník strojového učenia. Rozpoznávanie výrazu tváre na posúdenie emocionálnych stavov je široko skúmanou oblasťou v strojovom učení s mnohými

[1] Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1, 852 35 Bratislava, peter.schmidt@euba.sk.

[2] Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1, 852 35 Bratislava, veronika.horniakova@euba.sk.

[3] Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1, 852 35 Bratislava, peter.prochazka@euba.sk.

aplikáciami v zdravotníctve a interakcii človeka s počítačom (Zeng et al., 2018). Techniky hlbokého učenia preukázali významné zlepšenia v rozpoznávaní a kategorizácii výrazov tváre, ktoré možno využiť na hodnotenie duševného zdravia (Li & Deng, 2020). Naša metóda je založená na monitorovaní človeka prostredníctvom kamery počítača a pravidelnom automatickom snímkovaní jeho tváre. Testovaná osoba bude sledovať obrázky zobrazované na monitore. Na základe zaznamenanej mimiky tváre je možné vyhodnotiť emočnú reakciu. Keďže každý človek vyjadruje svoje emócie rôznou intenzitou, rozhodli sme sa využiť technológie Deep Learningu na naučenie algoritmu rozpoznávať reakcie, ktoré sa považujú za normálne. Všetky zachytené reakcie, ktoré sa odchýlia od normy, budú vyžadovať analýzu, či ide o dočasný nepriaznivý stav mentálneho zdravia spôsobený stresom alebo depresiou, alebo o charakterovú črtu indikujúcu určitú formu psychickej deviácie. Pri takýchto odchýlkach tradičné testy ako MMPI (Minnesota Multiphasic Personality Inventory) alebo Beckov inventár depresie nemusia byť dostatočne spoľahlivé, pretože sú navrhnuté tak, aby kvantitatívne merali určité aspekty duševného zdravia. Samotný dotazník zisťuje len základné údaje ako vek, pohlavie, zamestnanie a podobne. Užívatelia odpovedia na tieto základné otázky a program spustí samotný test. Na konci testu aplikácia vyhodnotí zistenia na základe toho v akom prostredí sa bude aplikácia používať. Strojové učenie má veľký potenciál pre predikciu duševných zdravotných stavov, vrátane depresie, a jeho aplikácia v zdravotníctve sa neustále rozširuje (Iyortsuun et al., 2023).

2 Formulácia problému

Depresia, známa aj ako veľká depresívna porucha, je duševné ochorenie, ktoré negatívne ovplyvňuje to, ako sa človek cíti, ako myslí a ako koná. Odhaduje sa, že približne 3,4 % (2-6 % vrátane chybového rozpätia) svetovej populácie trpí depresiou. Odhaduje sa, že jedna z troch žien a jeden z piatich mužov zažije vážnu depresiu počas svojho života (Dattani et al., 2023). Podľa Global Health Data Exchange trpí depresiou na celom svete 251-310 miliónov ľudí (WPR, 2024). Hoci je depresia bežná, našťastie je tiež liečiteľná. Depresia, ktorá je na celom svete uznávaná ako jedna z hlavných príčin invalidity, nielenže znižuje kvalitu života, ale tiež spôsobuje významné spoločenské náklady. Aj keď je možné depresiu spoľahlivo diagnostikovať a liečiť v primárnej zdravotnej starostlivosti, menej ako 25 % postihnutých má prístup k účinnej liečbe (WHO, 2023).

Heterogenita depresívnych porúch ďalej komplikuje ich zisťovanie a liečbu. Patria sem:

- Veľká depresívna porucha (MDD) - charakterizovaná ťažkými depresívnymi epizódami.
- Dystymická porucha (dystymia) - trvalá mierna depresia.
- Mierne depresia - menej vážne, ale stále výrazné depresívne príznaky.
- Psychotická depresia - depresia sprevádzaná bludmi alebo halucináciami.
- Popôrodná depresia - depresia, ktorá nastáva po pôrode.
- Sezónna afektívna porucha (SAD) - depresia spojená so sezónnymi zmenami.
- Bipolárna porucha - zahŕňa obdobia depresie a zvýšenej nálady.

Rôznorodé prejavy depresie si vyžadujú prístup pri hodnotení a zásahu šitý na mieru. Avšak chýbajúce univerzálne prístupné a prispôsobiteľné diagnostické nástroje predstavujú významnú prekážku efektívneho manažmentu. Tento výskum sa snaží riešiť tento nedostatok vývojom aplikácie riadeného hlbokým učením, schopného hodnotiť mentálne zdravie človeka ako aj rôzne stavy depresie prostredníctvom používateľsky príjemnej digitálnej platformy, čím zvyšuje prístupnosť a včasnosť podpory duševného zdravia. Včasná diagnostika umožnená strojovým učením a rýchla intervencia môže významne zlepšiť výsledky liečby depresie a iných duševných porúch (Shatte et al., 2019).

3 Príznaky a symptómy depresie

Dlhotrvajúce nešťastné, nekľudné alebo "prázdne" pocity, pocity smútku alebo cynizmu, pocity viny, bezcennosti alebo bezmocnosti, podráždenosť, nepokoj, strata záujmu o aktivity alebo koníčky, ktoré boli kedysi príjemné, vrátane sexu, únava a znížená energia. Ťažkosti so sústredením, zapamätaním si detailov a rozhodovaním. Nespavosť, skoré ranné bdenie alebo nadmerné spanie. Prejedanie sa alebo strata chuti do jedla. Myšlienky na samovraždu, pokusy o samovraždu. Silné bolesti alebo migrény, kŕče alebo tráviace problémy, ktoré neustupujú ani s liečbou.

Zdravotné problémy, ktoré často sprevádzajú depresiu:

- Panická porucha
- Sociálna fóbia
- Anxiozita
- Chronická bolesť
- Poruchy spánku
- Závislosť od alkoholu alebo drog
- Generalizovaná úzkostná porucha
- Obsedantno-kompulzívna porucha
- Posttraumatická stresová porucha (PTSD)
- Depresia môže tiež nastať s inými vážnymi lekárskymi ochoreniami ako sú srdcové choroby, mozgová príhoda, rakovina, HIV/AIDS, diabetes a Parkinsonova choroba.

Príčiny depresie

- *Genetické alebo dedičné* - Genetické faktory môžu zvýšiť predispozíciu k depresii, pričom niektoré depresívne poruchy môžu byť dedičné.
- *Biologické / Biochemické* - Nerovnováha neurotransmiterov v mozgu, ako sú serotonin a dopamín, môže prispievať k vzniku depresie.
- *Stravovacie* - Nedostatočná výživa, nedostatok určitých vitamínov a minerálov, môže mať vplyv na náladu a funkcie mozgu.
- *Environmentálne* - Externé environmentálne faktory ako dlhodobý stres, traumatické udalosti, alebo chronická izolácia môžu spúšťať depresívne epizódy.
- *Socio-kultúrne faktory / Situácie / Vzťahy / Osobnosť* - Sociálne a kultúrne faktory, ako sú rodinné vzťahy, pracovné prostredie alebo sociálna izolácia, môžu významne ovplyvňovať duševné zdravie jedinca.

Manažment

Antidepresíva primárne pôsobia na mozgové chemikálie nazývané neurotransmitery, najmä na serotonín a norepinefrín. Iné antidepresíva pôsobia na neurotransmiter dopamín. Pri liečbe depresie sa často využívajú rôzne nástroje a terapeutické prístupy. Okrem tradičných terapeutických metód sa v súčasnosti čoraz viac využívajú aj pokročilé technológie.

Medzi technológie, ktoré nachádzajú uplatnenie v podpore liečby depresie, patria rôzne aplikácie na spracovanie dát, ako aj niektoré programovacie jazyky ako Python alebo R, ktoré sú vhodné pre analýzu a spracovanie dát. Tieto technológie umožňujú realizáciu rôznych úloh, vrátane:

- Zber histórie: Programovacie nástroje umožňujú systematické zbieranie a analýzu údajov o histórii pacienta, čo pomáha klinickým pracovníkom pochopiť priebeh depresie a prispôsobiť plány liečby.

- Vyšetrenie duševného stavu: Aplikácie pomáhajú pri vykonávaní dôkladných vyšetrení duševného stavu, čo umožňuje zdravotníckym profesionálom hodnotiť kognitívne funkcie, emocionálnu pohodu a ďalšie dôležité faktory.
- Kritériá ICD-10: Technologické aplikácie sa používajú na zabezpečenie dodržiavania Medzinárodnej štatistickej klasifikácie chorôb a súvisiacich zdravotných problémov, 10. revízie (ICD-10) pre diagnostiku a kódovanie depresie a súvisiacich stavov. Táto klasifikácia je publikovaná Svetovou zdravotníckou organizáciou (WHO) a používa sa na diagnostiku, štatistické sledovanie a kódovanie chorôb a zdravotných stavov.

Antidepresíva

- Selektívne inhibítory spätného vychytávania serotonínu (SSRI) - Fluoxetín (Prozac), Sertralín (Zoloft), Escitalopram (Lexapro), Paroxetín (Paxil) a Citalopram (Celexa) patria medzi najčastejšie predpisované SSRI pri liečbe depresie.
- Tricyklické antidepresíva - Sú staršie druhy antidepresív. Majú silný účinok, ale dnes sa používajú menej často kvôli vážnejším možným vedľajším účinkom, napríklad Imipramín a Nortriptylín
- Inhibítory monoamín oxidázy (MAOI) - Sú najstaršou triedou antidepresívnych liekov. Môžu byť obzvlášť účinné pri atypickej depresii. Osoby užívajúce MAOI sa musia vyhýbať určitým potravinám a nápojom (vrátane syra a červeného vína), ktoré obsahujú látku zvanú tyramín.

Kognitívno-behaviorálna terapia (CBT)

Kognitívno-behaviorálna terapia pomáha ľuďom trpiacim depresiou prepracovať negatívne vzorce myslenia. Táto terapeutická metóda zameraná na riešenie problémov pomáha identifikovať a zmeniť deštruktívne alebo rušivé myšlienkové procesy, ktoré majú negatívny vplyv na správanie a emócie. CBT sa zameriava na aktuálne problémy a hľadá praktické spôsoby, ako ich klienti môžu prekonať, čo vedie k zlepšeniu ich každodennej funkčnosti a životnej spokojnosti.

Interpersonálna terapia (IPT)

- Pomoc s narušenými vzťahmi: Interpersonálna terapia (IPT) je psychoterapeutická metóda zameraná na riešenie medziľudských problémov, ktoré prispievajú k vzniku a udržiavaniu depresie. Cieľom IPT je pomôcť klientom rozpoznať vzory v ich vzťahoch, ktoré môžu negatívne ovplyvňovať ich emocionálny stav. Terapeut spolu s klientom pracuje na zlepšení komunikačných zručností, rozvíja schopnosť vyjadrovať emócie a podporuje pozitívne interakcie s ostatnými. Tento prístup môže viesť k zlepšeniu kvality existujúcich vzťahov a k vybudovaniu nových, podporných medziľudských väzieb.
- Elektrokonvulzívna terapia (ECT) pre odolné prípady: ECT je liečebná procedúra používaná pre pacientov trpiacich ťažkou depresiou, ktorá neodpovedá na štandardné liečebné metódy ako sú antidepresíva alebo psychoterapia. Počas procedúry sú u pacienta indukované kontrolované epileptické záchvaty použitím krátkeho elektrického prúdu prechádzajúceho mozgom. Táto metóda môže priniesť rýchle zlepšenie symptómov u pacientov s veľmi ťažkou, život ohrozujúcou depresiou alebo pri pacientoch, ktorí sú extrémne suicidálni. Hoci ECT je efektívna, je dôležité zvážiť potenciálne vedľajšie účinky, ako sú zmätenosť a strata krátkodobej pamäti, ktoré môžu po zákroku pretrvávať.

4 Transférové učenie

Transférové učenie je metódou, ktorá umožňuje efektívnejšie využitie existujúcich dát a získaných poznatkov. Táto metóda sa zameriava na aplikáciu vedomostí z jednej úlohy na novú, podobnú úlohu. V praxi to znamená, že neurónová sieť je najprv trénovaná na špecifickom súbore dát a jej dolné vrstvy, ktoré slúžia na extrakciu charakteristík, sú "zamrznuté". Následne sa tieto vrstvy používajú ako základ pre ďalšie špecializované trénovanie. Napríklad, model trénovaný na rozpoznávanie mimiky tváre mladých mužov môže byť adaptovaný na rozlišovanie medzi mimikami tváre detí alebo starších žien. Táto technika výrazne znižuje čas potrebný na trénovanie nových modelov, pretože využíva už existujúce výstupy z predchádzajúcich tréningov.

4.1 Využitie doménových znalostí v kontexte strojového učenia

Doménové znalosti predstavujú dôležitý nástroj pri vylepšovaní modelov strojového učenia, najmä v prípadoch, kedy sú dáta obmedzené. Tu je niekoľko kľúčových spôsobov, ako môžu byť doménové znalosti využité:

- *Vytvorenie expertných pravidiel*: Experti v danej oblasti môžu definovať pravidlá alebo heuristiky, ktoré sa integrujú do modelov. Tieto pravidlá môžu vychádzať z dlhoročných skúseností.
- *Výber charakteristík:* Doménové znalosti môžu pomôcť pri identifikácii najrelevantnejších charakteristík, čo zvyšuje efektivitu modelu.
- *Predspracovanie dát:* Odborníci môžu odporučiť konkrétne metódy pre čistenie a spracovanie dát, aby boli v súlade so štandardmi danej oblasti.
- *Interpretácia výsledkov:* Doménové znalosti sú neoceniteľné pri interpretácii výsledkov a identifikácii oblastí pre zlepšenie.
- *Validácia modelu:* Experti môžu pomôcť overiť, či sú výsledky modelu v súlade s očakávaniami a identifikovať potrebné úpravy.
- *Vytvorenie špecifických modelov*: Niekedy je vhodné vytvoriť modely špecificky pre danú doménu, pričom sa zohľadnia jej unikátne výzvy a charakteristiky.

Využitie doménových znalostí zvyšuje úspešnosť a spoľahlivosť strojového učenia tým, že poskytuje dôležitý kontext, ktorý nemôže byť získaný len z dát. Integrácia týchto znalostí do procesu modelovania môže výrazne zvýšiť kvalitu a spoľahlivosť výsledných modelov.

5 Vyhodnocovací model

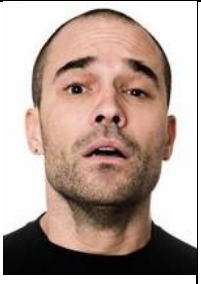
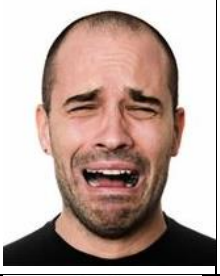
Už v deväťdesiatych rokoch minulého storočia boli realizované štúdie, kde boli analyzované emocionálne reakcie a rozpoznávacie schopnosti tváre u rôznych skupín osôb. Jedna zo štúdií zahŕňala 23 pacientov s akútnou schizofréniou, 21 pacientov s akútnou veľkou depresiou podľa výskumných diagnostických kritérií, a 15 zdravých kontrol. Emocionálna reaktivita a schopnosť rozpoznávania tváre boli hodnotené vo dvoch nastaveniach: klinickom a experimentálnom. Klinicky bol emocionálny výraz hodnotený na základe afektívneho sploštenia, zatiaľ čo v laboratórnych podmienkach boli skúmané mimovoľné aj dobrovoľné prejavy tváre pomocou video analýzy, ktorá poskytla dáta o intenzite a presnosti výrazov.

Výsledky ukázali, že obidve klinické skupiny - pacienti so schizofréniou aj s veľkou depresiou - preukázali špecifické vzorce dysfunkcií súvisiacich s afektom. Schizofrenici vykazovali výrazné deficity v spontánnom aj dobrovoľnom prejave tváre, nezávisle od typu a dávkovania použitých neuroleptík. U depresívnych pacientov bol zaznamenaný stabilný deficit v mimovoľnom prejave počas emocionálne vyvolávajúcich rozhovorov, pričom dobrovoľné

prejavy boli výrazne redukované počas postakútnej fázy. Tieto rozdiely môžu poukazovať na základné psychobiologické rozdiely medzi týmito skupinami, čo naznačuje odlišné neurobiologické mechanizmy ovplyvňujúce ich schopnosť emočného prejavu a rozpoznávania.

Na rozdiel od predchádzajúcich štúdií, ktoré sa spoliehali predovšetkým na odborné znalosti špecialistov a pracovali s relatívne malou vzorkou účastníkov, naše aktuálne zistenia poskytujú širší pohľad na problematiku. Aj keď naše diagnózy nie sú tak závažné ako v predchádzajúcich prípadoch, identifikácia a dokumentácia symptómov je komplikovaná ich sporadickou prítomnosťou a obtiažnou detekciou v ranných štádiách ochorenia.

Obr.1: Ukážka rôznych reakcií na zobrazený obrázok

Zobrazený obrázok	Zosnímaná mimika tváre	Očakávaná reakcia	Depresívna reakcia	Deviačná reakcia
				
				
				

Zdroj: (Flores, 2024)

Na Obr. 1 sú zobrazené rôzne reakcie testovanej osoby v závislosti od premietaného obrázka.

- V prvom riadku je vidno pár mačiatok, ktoré vo všeobecnosti vyvolávajú pozitívnu reakciu. Očakáva sa minimálne jemný úsmev, ale aj veľký úsmev s otvorenými ústami sa dá hodnotiť ako úplne normálna reakcia. Pri depresívnej reakcii vidíme stiahnuté kútiky, akoby sa testovaná osoba hneď rozplakala. S veľkou pravdepodobnosťou pôjde o prehnanú reakciu príčinou čoho je zlá psychická pohoda testovanej osoby. Typickým príznakom deviačnej reakcie môže byť "poker face", keby obrázok nevyvolá žiadnu emotívnu reakciu (Gaebel & Wölwer,1992).
- V druhom riadku je obrázok na ktorom ľudia berú obeť teroristických útokov. Očakávaná reakcia je skôr smutný výraz, alebo prekvapivý výraz tváre. Vidíme že zosnímaná mimika hovorí že testovaná osoba neprejavuje žiadnu empatiu a usmieva sa ako osoba s

deviačnou reakciou. Pri depresívnej reakcii už testovaná osoba nedokáže udržať slzy a reaguje veľmi emotívne.

- Na treťom riadku by sme očakávali spokojnú mimiku, s privretými očami, akoby sa už človek na tej pláži videl. Oproti tomu zosnímaná mimika tváre je skôr veľmi smutná, ktorú môže vyvolať napr. veľmi zlá spomienka. Vidíme, že pri depresívnej reakcii je snaha o úsmev, ale stiahnuté obočie prezrádza že obrázok vyvoláva menej pozitívnu emočnú reakciu než by sme u zdravého človeka očakávali. Pri deviačnej reakcii nevidíme žiadny emočný prejav.

Ak by sme chceli v hrubých rysoch vyhodnotiť obr.1, mohli by sme povedať že osoba na prvom riadku je osoba mentálne zdravá a jeho reakcia a empatia bola primeraná a zodpovedá očakávanej reakcii.

Na druhom riadku je osoba s deviačnou reakciou, nakoľko fotografia u normálnej populácie vyvolávajúca smútok, u tejto osoby vyvolala úsmev, čiže miera empatie je u neho veľmi nízka.

Na treťom riadku je typický prípad depresívnej reakcie, kde obrázok ktorý vo všeobecnosti vyvoláva pozitívne pocity u testovanej osoby vyvoláva smutnú reakciu.

6 Realizácia systému

Realizácia systému zahŕňa viacero dôležitých krokov, pričom hlavným cieľom je efektívne využitie transferového učenia. Predtrénované modely upravíme tak, aby odrážali špecifiká nového datasetu, čo umožní lepšie výsledky aj pri menšom počte trénovacích dát.

1. Vytvorenie aplikácie:
   a. Obsah aplikácie: Aplikácia bude obsahovať bohatú knižnicu obrázkov určených na vyvolanie emocionálnych reakcií používateľov.
   b. Zaznamenávanie reakcií: Pri zobrazení každého nového obrázku aplikácia zaznamená mimiku testovanej osoby pomocou integrovanej kamery a uloží dáta do databázy pre ďalšie spracovanie.
2. Vytvorenie trénovacej sady:
   a. Generovanie dát: Trénovacia sada sa bude generovať automaticky získavaním dát o reakciách užívateľov podľa vyššie uvedeného bodu 1a.
   b. Anotácia dát: Fotografie z trénovacej sady budú ručne anotované na základe prejavených emocionálnych reakcií.
3. Vytvorenie testovacej sady: Testovacia sada bude vytvorená podobným spôsobom ako trénovacia sada a použije sa na overenie účinnosti modelu.
4. Učenie modelu: Model bude učený s použitím učenia zo supervízorom, čo umožní modelu učiť sa z anotovaných dát a zlepšiť jeho schopnosť predikcie na základe emocionálnej reakcie.
5. Vypracovanie hodnotiaceho algoritmu: Hodnotiaci algoritmus bude vyvinutý na posúdenie účinnosti modelu v reálnych scenároch.
6. Testovanie systému na reálnej vzorke: Po vývoji systému a učení modelu bude systém otestovaný na reálnej vzorke používateľov na overenie jeho funkčnosti a presnosti.

7 Záver

Tento výskum predstavuje významný pokrok vo využití strojového učenia pre detekciu stavov mentálneho zdravia, špecificky depresie. Metóda, založená na monitorovaní človeka cez počítačovú kameru a analýze mimiky pri vizuálnych stimuloch, predstavuje inovatívny prístup k hodnoteniu emočných reakcií. Hlavným prínosom je možnosť identifikácie abnormálnych

emocionálnych stavov, ktoré by mohli naznačovať mentálne problémy ako depresiu alebo iné psychické odchýlky. Prístup založený na Deep Learningu umožňuje adaptáciu modelu na individuálne rozdiely v emočnom prejave, čo zvyšuje presnosť diagnostiky.

Vývoj aplikácie, ktorá je schopná učiť sa z rozmanitosti emocionálnych prejavov a reakcií testovaných osôb, poskytuje dôležitú platformu pre skorú diagnostiku a intervenčné stratégie. Tento systém má potenciál znížiť bariéry v prístupe k psychologickému hodnoteniu a podporiť včasnú intervenciu, čo je kľúčové pri liečbe depresie a iných súvisiacich porúch.

Celkovo možno povedať, že zavedenie takéhoto systému by mohlo výrazne prispieť k zlepšeniu duševného zdravia na individuálnej aj spoločenskej úrovni, keďže by poskytovalo spoľahlivé a rýchle vyhodnotenie mentálneho stavu bez nutnosti fyzickej návštevy zdravotníckeho zariadenia. Predpokladá sa, že výsledky tohto výskumu môžu byť aplikované v rôznych prostrediach, čím sa zvyšuje ich univerzálnosť a dostupnosť podpory pre ľudí trpiacich duševnými ochoreniami.

Literatúra

1. Dattani, S., Rodés-Guirao, L., Ritchie, H., & Roser, M. (2023, December 28). *Mental health*. Our World in Data. https://ourworldindata.org/mental-health#research-writing.
2. Flores, A. (2024, February 19). What is morphopsychology. Face & Profile. http://faceandprofile.com/things-to-know-about-morphopsychology/.
3. Gaebel, W., & Wölwer, W. (1992). Facial expression and emotional face recognition in schizophrenia and depression. *European Archives of Psychiatry and Clinical Neuroscience*, *242*(1), 46–52. https://doi.org/10.1007/bf02190342
4. Iyortsuun, N. K., Kim, S., Jhon, M., & Yang, H. (2023). A review of machine learning and deep learning approaches on mental health diagnosis. Healthcare, 11(3), 285. https://doi.org/10.3390/healthcare11030285.
5. Li, S., & Deng, W. (2020). Deep facial expression recognition: A survey. IEEE Transactions on Affective Computing. https://doi.org/10.1109/TAFFC.2020.2981446.
6. Shatte, A. B. R., Hutchinson, D. M., & Teague, S. J. (2019). Machine learning in mental health: A scoping review of methods and applications. Psychological Medicine, 49(9), 1426-1448. https://doi.org/10.1017/S0033291719000151.
7. WHO. (2023, March 31). *Depressive disorder (depression)*. World Health Organization. https://www.who.int/news-room/fact-sheets/detail/depression.
8. WPR. (2024). *Depression Rates by Country 2024*. Depression rates by country 2024. https://worldpopulationreview.com/country-rankings/depression-rates-by-country.
9. Zeng, Z., Pantic, M., Roisman, G. I., & Huang, T. S. (2018). A survey of affect recognition methods: Audio, visual, and spontaneous expressions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(1), 39-58. https://doi.org/10.1109/TPAMI.2008.52.

Porovnanie exekučnej efektívnosti programovacích jazykov Python a Scala používaných v aplikáciách pre dátovú vedu

Comparison of the code execution efficiency of programming languages Python and Scala used in data science applications

Pavol Sojka[1]

Abstrakt
Náš článok sa zameriava na exekučnú efektívnosť programovacích jazykov Python a Scala z hľadiska časovej náročnosti. Oba jazyky sa používajú v dátovej vede. Dátová veda sa v súčasnosti rozrástla vo veľkom meradle a tento článok by mal priniesť lepší prehľad o tom, ktorý z vybraných jazykov má lepší výkon. Pripravili sme experiment a scenár bežne používaný v praxi v oboch jazykoch, ktorý zahŕňa prípravu dát, ich spracovanie a výpočet a následne interpretáciu vrátených výsledkov.

Kľúčové slová
Python, Scala, množiny údajov, spracovanie údajov

Abstract
Our paper aims on using programming languages like Python and Scala to compare their efficiency in the meaning of the code execution times. Both languages are being used in data science. Data science has grown at large scale nowadays and this paper should bring a better insight into which of the chosen languages have better performance. We prepared an experiment and scenario commonly used in practice in both languages which comprises data preparation, data processing and calculation and consequently interpretation of returned results.

Key words
Python, Scala, datasets, data processing

JEL classification
C8

1 Úvod

V súčasnosti je veľký dopyt po dátových analytikoch a programátoroch, pretože dátová veda a priemysel v posledných rokoch výrazne narástli. S týmto trendom sa objavujú otázky, ako napríklad, ktorý programovací jazyk by sme mali použiť na splnenie požiadaviek na spracovanie dát. Mnohé odpovede možno nájsť na internete alebo v literatúre. Môžeme zvážiť tieto odporúčania na základe rozšírenosti jazyka, krivky učenia, dostupnosti zdrojov poznatkov a podobne. V našom článku sme si vybrali jedno kritérium na porovnanie a toto kritérium je založené striktne na meraní efektívnosti na základe hardvérových parametrov s prednastaveným softvérovým scenárom.

[1] Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1/b, 852 35 Bratislava, pavol.sojka@euba.sk

2 Príprava prostredia

Ako primárny operačný systém, na ktorom budú vykonávané experimenty, sme si vybrali Linux Ubuntu 22.04. Naše prostredie bolo umiestnené na platforme Google Cloud na virtuálnom stroji s hardvérovými parametrami začínajúcimi na ôsmich CPU (centrálna procesorová jednotka) a šestnástich gigabajtoch RAM (operačná pamäť). Počet CPU sa postupne menil a merali sme dopad na efektívnosť vykonávania kódu z hľadiska času. Môže sa zdať, že výsledky sú ľahko predvídateľné, ale naším cieľom nebolo len dokázať, že zníženie počtu CPU má negatívny vplyv na výkon, ale tiež presne merať hodnoty vo zvolenom prostredí a skúmať, či tento vplyv je rovnaký alebo podobný u oboch programovacích jazykov.

Po úspešnom nasadení operačného systému sme nainštalovali jazyk Python a jazyk Scala z repozitárov operačného systému. Spolu s Pythonom sme nainštalovali aj knižnicu Pandas. Táto knižnica je navrhnutá na efektívnu prácu s dátovými sadami uloženými vo formátoch CSV, XLSX a ďalších. Na úspešnú inštaláciu kompilátora Scala je potrebné nainštalovať Java framework a vývojové nástroje.

Naším cieľom bolo otestovať rýchlosť nášho naprogramovaného kódu v týchto krokoch:

- Otvoriť súbor CSV
- Načítať stĺpce dátovej sady do pamäte
- Vypočítať súčet celého stĺpca „Tržby“ a „Zisk“
- Vytlačiť názov súboru a výsledok na obrazovku
- Vytlačiť výsledný čas vykonávania a ukončiť aplikáciu

Rozhodli sme sa generovať syntetické dátové sady vo formáte CSV (hodnoty oddelené čiarkami). Na tento účel sme použili aplikáciu naprogramovanú v jazyku Java pôvodne pre iné experimenty vykonané v minulosti. Vygenerovali sme 500 testovacích súborov s údajmi o 20 riadkoch a 15 stĺpcoch. Náš program môže generovať dátové sady so stovkami tisíc riadkov, ale pre naše experimenty by malo byť 500 súborov x 20 riadkov dostatočných. Vygenerované súbory boli skopírované do adresárov Python a Scala.

3 Výsledky

Každé meranie sme vykonali trikrát za sebou v oboch jazykoch a zaznamenané časy sme uložili do tabuľky. Po troch cykloch sme vypli server a postupne menili počet používaných CPU z ôsmich na dve. Pozorovali sme malé zmeny v časoch vykonávania kódu, ale vyskytli sa len menšie rozdiely, až pri dvoch CPU bola zmena času významná. Ako vidno z tabuliek nižšie, môžeme povedať, že program v Scale bol efektívnejší ako v jazyku Python, pretože Scala je kompilovaná do bajtkódu spusteného na Java Virtual Machine, ktorý je vždy rýchlejší ako interpretovaný jazyk – Python. Oba programy bežali v konzolovom okne Linux bash. Prvý meraný cyklus v oboch jazykoch bol vždy pomalší ako ďalšie dva. Tento efekt možno vysvetliť prednačítaním systémových knižníc do pamäti po prvom behu.

Tab. 1: Časy vykonávania (8 CPU)

	CPU: 8, Pamäť: 16 GB		
	Python (s)	**Scala (s)**	**Rozdiel (python/scala)**
1. meranie	1,50	0,51	2,92
2. meranie	1,08	0,22	4,84
3. meranie	1,08	0,24	4,45
priemer	**1,22**	**0,33**	3,74

Zdroj: vlastné spracovanie

Tab. 2: Časy vykonávania (6 CPU)

	CPU: 6, Pamäť: 16 GB		
	Python (s)	**Scala (s)**	**Rozdiel**
1. meranie	1,44	0,56	2,55
2. meranie	1,07	0,25	4,22
3. meranie	1,09	0,26	4,14
priemer	**1,20**	**0,36**	3,33

Zdroj: vlastné spracovanie

Tab. 3: Časy vykonávania (4 CPU)

	CPU: 4, Pamäť: 16 GB		
	Python (s)	**Scala (s)**	**Rozdiel**
1. meranie	1,45	0,60	2,42
2. meranie	1,07	0,31	3,49
3. meranie	1,08	0,32	3,37
priemer	**1,20**	**0,41**	2,94

Zdroj: vlastné spracovanie

Tab. 4: Časy vykonávania (2 CPU)

	CPU: 2, Pamäť: 16 GB		
	Python (s)	**Scala (s)**	**Rozdiel**
1. meranie	1,76	0,84	2,08
2. meranie	1,22	0,52	2,35
3. meranie	1,25	0,49	2,57
priemer	**1,41**	**0,62**	2,28

Zdroj: vlastné spracovanie

Ako môžeme vidieť z údajov v tabuľkách, jazyk Scala je vždy efektívnejší ako jazyk Python. Efektívnosť oboch jazykov mierne klesá vždy, keď sa zníži počet jadier (CPU). Tabuľky 1–3 ukazujú, že efektívnosť Pythonu je podobná a rýchlo klesá v poslednom scenári iba s dvoma CPU. Efektívnosť jazyka Scala významne klesá v každom experimente. Ako bolo spomenuté vyššie, prvé meranie je pomalšie ako ďalšie dve a tento efekt ovplyvňujú interné procesy riadené operačným systémom. Rozdiely v jazykoch Python a Scala sú na prvý pohľad zrejmé. V porovnaní s Pythonom a knižnicou Pandas bude Scala vždy efektívnejšia, pretože Scala je staticky typovaný programovací jazyk a kompilátor pozná každú premennú a výraz pri behu (Jancauskas, 2016). Na druhej strane Python je dynamicky typovaný programovací jazyk a preto sa typ premennej odvodzuje na základe jej použitia (Kohli, 2021). Ďalší rozdiel je, že Scala je kompilovaná do bajtkódu, ktorý sa následne vykonáva na Java Virtual Machine (Jancauskas, 2016). Python s jeho najbežnejším použitím je interpretovaný vždy zo zdrojového kódu, čo je jasne viditeľné z časov vykonávania (nižší čas spracovania = vyššia efektívnosť aplikácie pri ostatných nezmenených podmienkach). Na optimalizáciu procesov nášho konceptu by sme mali zahrnúť špecifické algoritmy vhodné pre daný typ úlohy, hlbší vhľad do konkrétneho programovacieho jazyka, výber vhodnejších programovacích jazykov pre strojové učenie alebo výber výkonnejších knižníc jazyka Python použitých pri strojovom učení (Esposito & Esposito, 2020). Náš článok má za cieľ základné predstavenie často používaných

jazykov strojového učenia, ktoré zahŕňajú jazyky ako Scala, Python a R (Farth, 2018). Často používaným nástrojom v praxi je tiež rámec Spark (Mehrotra & Grade, 2019), ktorý využíva efektívnosť Scala/Java a knižnice Spark používaných na manipuláciu s obrovským množstvom dát, ako aj platformy Hadoop (Nordeen, 2020) používané na udržiavanie takýchto veľkých dátových sád. Nasledujú časti kódu, ktoré sme použili v našom testovacom prostredí.

Obr. 1: Ukážka kódu v jazyku Python

```
for f in csv_files:
        df = pd.read_csv
 (f, sep=';')
        print('Location:', f)
        print('File Name:', f.split("\\")[-1])
                print(f)
        print('Total profit:', df['profit'].sum())
        print('Total sales:', df['sale'].sum())
        print("------------------------------")
```

Zdroj: vlastné spracovanie

Obr. 2: Ukážka kódu v jazyku Scala

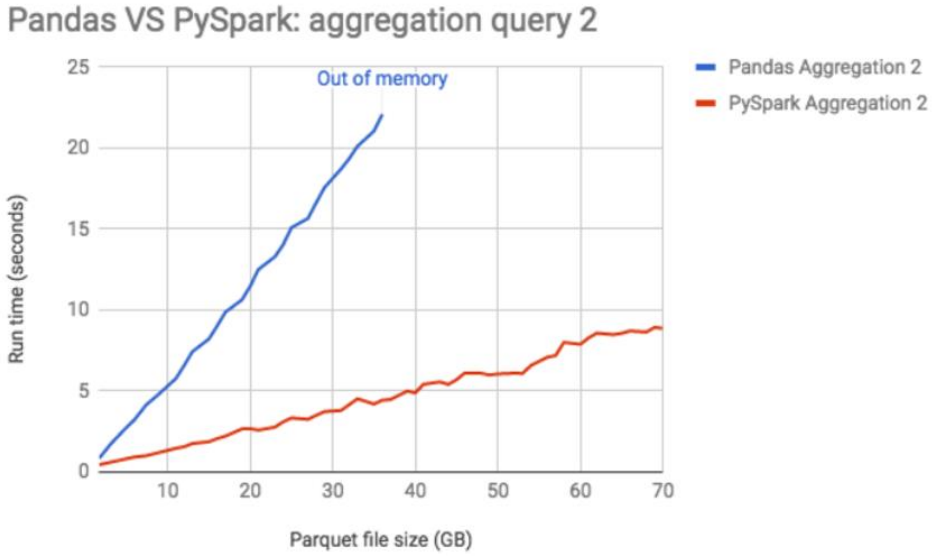
```
def compute(filename:String) {
   val bufferedSource = io.Source.fromFile(filename)
   var start = 1
   var value, value2 : Double = 0.0
   var sum : Double = 0
   var sum2 : Double = 0
   for (line <- bufferedSource.getLines) {
      val cols = line.split(";").map(_.trim)
        if (start!=1) { value = cols(4).toDouble
                        sum += value
                        value2 = cols(5).toDouble
                        sum2 += value2
                         }
        start = 0
   }
   bufferedSource.close
   println(filename)
   println("Total profit:"+sum)
   println("Total sale:"+sum2)
   println("----------------------")
 }
```

Zdroj: vlastné spracovanie

4 Diskusia

Náš prístup by mohol byť optimalizovaný pre lepšiu efektívnosť mnohými nástrojmi a knižnicami ako Hadoop, Spark a špecializovanými rámcami používanými oboma jazykmi. Ďalšie otázky, aký jazyk použiť, môžu vzniknúť, pretože v posledných rokoch dochádza k veľkému rozvoju mnohých ďalších jazykov (napr. Mojo, Julia a iné). Python môže byť tiež používaný spolu s rámcom (framework) Apache Spark. V tomto rámci je Python optimalizovaný pre špecializované knižnice, ktoré sú oveľa efektívnejšie ako samotný Python (Aven, 2018). Ako ukazuje graf nižšie, doba behu niekoľkých dopytov v PySpark (jazyk Python bežiaci spolu s Apache Spark) v porovnaní s rovnakými dopytmi v knižnici Pandas. Je zrejmé, že (Py)Spark naozaj umožňuje pracovať s väčšími dátami efektívnejším spôsobom (Element 61, 2020). Graf ukazuje (tak ako aj ako náš článok) celkovú efektívnosť funkcie „sum“ v knižnici Python Pandas oproti PySpark.

Obr. 3: Agregačný dopyt Pandas vs PySpark



Zdroj: Element (Element 61, 2020)

5 Záver

V našom článku sme sa zamerali na porovnanie dvoch jazykov aplikovaných na jednoduchú výpočtovú úlohu, len aby sme ukázali efektívnosť použitých jazykov. Na základe našich výsledkov sme demonštrovali, ktorý z vybraných jazykov je lepší z hľadiska výkonu. Existujú ďalšie aspekty, ktoré by sa mali zvážiť, a to je rozšírenosť konkrétneho jazyka a krivka učenia. Pri zohľadnení týchto možností je víťazom jazyk Python, kvôli jeho rozšírenosti a známosti. Jazyk Scala je pre nováčikov trochu komplikovaný na rýchle pochopenie a preto nie je veľmi známy mimo komunity vývojárov Scala. Veľkou výhodou pre nováčikov je predchádzajúca prax s jazykom Java. Tieto jazyky nie sú rovnaké, ale môžu zdieľať knižnice, sú silne typované, objektovo orientované (každá hodnota je objekt) a používajú Java Virtual Machine ako spoločné prostredie na vykonávanie bajtkódu. Na druhej strane Python je celosvetovo uznávaný jazyk s veľkou základňou používateľov a rozsiahlou poznatkovou bázou na riešenie problémov. Učenie sa tohto jazyka nie je pre nováčikov také zložité ako Scala. Z našich kódových fragmentov možno pozorovať, že Python je ľahko čitateľný a jednoduchý. Tieto výhody a nevýhody môžu byť podporené alebo potlačené kombinovaním nástrojov ako Apache Spark, ktorý implementuje podporu pre Scala aj Python. Špeciálne pripravené knižnice by mohli zvýšiť efektívnosť Pythonu, ale stále nebude tak výkonný ako kompilované jazyky.

Na záver pridávame obrázok najpoužívanejších jazykov v dátovej vede do konca roka 2020, veríme, že podobné trendy budú pokračovať aj v ďalších rokoch.

Obr. 4: Programovacie jazyky používané v dátovej vede

PROGRAMMING LANGUAGES FOR DATA SCIENCE

Platform	2019 % share	2018 % share	% change
Python	65.8%	65.6%	0.2%
R Language	46.6%	48.5%	-4.0%
SQL Language	32.8%	39.6%	-17.2%
Java	12.4%	15.1%	-17.7%
Unix shell/awk	7.9%	9.2%	-13.4%
C/C++	7.1%	6.8%	3.7%
Other programming and data languages	6.8%	6.9%	-17.1%
Scala	3.5%	5.9%	-41.0%
Julia	1.7%	0.7%	150.4%
Perl	1.3%	1.0%	25.2%
Lisp	0.4%	0.3%	46.1%
Javascript	6.8%	na	na

Zdroj: Jelvix (Naumenko, 2020)

Literatúra

1. Aven, J. (2018). *Data Analytics with Spark Using Python*. Pearson Education.
2. Element 61. (2020). How to use Python and Pandas while embracing the power of Spark. Element 61. https://www.element61.be/en/resource/how-use-python-and-pandas-while-embracing-power-spark.
3. Esposito, D., & Esposito, F. (2020). *Introducing Machine Learning*. Pearson Education.
4. Farth, T. (2018). Machine Learning Guide for Beginners with R/Python/Scala. CreateSpace Independent Publishing Platform.
5. Jancauskas, V. (2016). *Scientific Computing with Scala*. Packt Publishing.
6. Kohli, M. (2021). *Basic Core Python Programming*. BPB Publications.
7. Mehrotra, S., & Grade, A. (2019). *Apache Spark Quick Start Guide*. Packt Publishing.
8. Naumenko, V. (2020). Top Data Science Programming Languages. Jelvix. https://jelvix.com/blog/top-data-science-programming-languages
9. Nordeen, A. (2020). *Learn Hadoop in 24 Hours.* Guru99.

EXTERNÍ RECENZENTI

Igor Bandurič
Peter Červenka
Martina Kuncová
Tomáš Páleník
Ján Pittner
Branislav Skladan
Romana Šipoldová

POKYNY PRE AUTOROV

Rozsah:

- vedecké state a diskusie 10 až 15 strán. Základnou požiadavkou je originalita príspevku a komplexnosť jeho spracovania. Prijímame príspevky v slovenskom, českom a anglickom jazyku (uprednostňujú sa príspevky v anglickom jazyku);
- informácie maximálne 2 strany;
- recenzie maximálne 2 strany.

Forma:

Použite textový editor MS WORD, verzia 2 000 a vyššia. Šablóna pre písanie článkov je na webovej stránke:

https://fhi.euba.sk/veda-a-vyskum/vedecke-casopisy/ekonomika-a-informatika/o-casopise

a v elektronickom systéme na stránke:

http://ei.fhi.sk/index.php/EAI

Príspevky predkladajú autori elektronicky vo formáte .doc/.docx do systému na stránke http://ei.fhi.sk/index.php/EAI. Príspevky sú recenzované. Redakčná rada zabezpečí interné a externé posúdenie textu príspevku. Autor príspevku je povinný zapracovať pripomienky z posudkov najneskôr do 2 týždňov od doručenia e-mailov so žiadosťou o vykonanie oponentských posudkov v elektronickom systéme časopisu a zaslať príspevok so zapracovanými pripomienkami vo formáte .doc/.docx prostredníctvom elektronického systému časopisu *Ekonomika a informatika*. Konečné rozhodnutie o publikovaní príspevku urobí redakčná rada časopisu. Autor pred zverejnením príslušného čísla časopisu *Ekonomika a informatika* odsúhlasí formátovanie elektronickej verzie článku. Fakulta hospodárskej informatiky si vyhradzuje právo zverejniť príspevky schválené redakčnou radou v elektronickej forme časopisu *Ekonomika a informatika*.

Autorské honoráre sa neplatia. Predložením príspevku do elektronického systému vedeckého časopisu *Ekonomika a informatika* dáva autor príspevku vydavateľovi právo, aby bezplatne publikoval text príspevku v časopise *Ekonomika a informatika* v elektronickej forme vo formáte .pdf.

EKONOMIKA A INFORMATIKA

Vedecký časopis Fakulty hospodárskej informatiky Ekonomickej univerzity v Bratislave a občianskeho združenia Slovenská spoločnosť pre hospodársku informatiku.

Poslaním vedeckého časopisu je publikovať teoretické a aplikačné poznatky získané v ekonomickom výskume a hospodárskej praxi z oblastí hospodárskej informatiky, účtovníctva a audítorstva, ekonometrie a operačného výskumu, aplikovanej štatistiky a aktuárstva, s akcentom na aktuálne otázky harmonizácie, integrácie a kompatibility s európskou a svetovou metodológiou a praxou.
Uverejňuje vedecké state a diskusie, recenzie a informácie o dizertačných a habilitačných prácach, inauguračných prednáškach a vedeckých podujatiach v slovenskom, českom alebo anglickom jazyku, ktoré sú výsledkom vedeckovýskumnej činnosti autorov, vedeckých aktivít doktorandov, medzinárodnej výskumnej a pedagogickej spolupráce a ich aplikácie v ekonomickej praxi.

ECONOMICS AND INFORMATICS

A scientific journal of the Faculty of Economic Informatics of University of Economics in Bratislava and the Slovak Economic Informatics Association.

Mission of the scientific journal is to publish theoretical and application knowledge acquired in economic research and practice in the areas of economic informatics, accounting and auditing, applied statistics, actuarial science, econometrics and operations research, with emphasis on the current issues of harmonization, integration and compatibility with the European and global methodology and practice.
The journal publishes scientific articles and paper discussions, reviews and information on doctoral and habilitation theses, inauguration lectures and scientific events in Slovak, Czech or English language, which are results of scientific and research activity of authors, scientific activities of doctoral students, international research and educational cooperation and their application in the economic practice.

EKONOMIKA A INFORMATIKA

Vydáva: Fakulta hospodárskej informatiky Ekonomickej univerzity v Bratislave a Slovenská spoločnosť pre hospodársku informatiku
Vychádza: 2x ročne