UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DIPLOMOVÁ PRÁCE

2024 Bc. David Kyncl

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Webová aplikace pro zpracování vědeckých dat

Diplomová práce

2024 Bc. David Kyncl

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. David Kyncl**
Osobní číslo:	**I22164**
Studijní program:	**N0613A140007 Informační technologie**
Téma práce:	**Webová aplikace pro zpracování vědeckých dat**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Cílem diplomové práce je návrh a implementace webové aplikace s uživatelským rozhraním pro zpracování vědeckých výsledků s cílem získat lepší přehled nad zpracovávanými daty. Aplikace bude poskytovat import dat ze souboru, popisné statistiky zpracovávaného datového souboru a tvorbu grafů. V teoretické části práce budou popsány vybrané statistické metody pro zpracování vědeckých dat a použité technologie. V praktické části student navrhne a implementuje samotnou webovou aplikaci. Backendová část aplikace bude využívat jazyk Python (Django) a vhodně zvolené knihovny. Frontendová část aplikace bude vytvořena pomocí vybraného component-based frameworku.

Rozsah pracovní zprávy: **cca 60 stran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

PECINOVSKÝ, Rudolf. Python: knihovny pro práci s daty pro verzi 3.11. Myslíme v.. Praha: Grada Publishing, 2023. ISBN 978-80-271-0659-2. LIM, Greg a Daniel CORREA. Django 4 for the Impatient. Packt Publishing, 2022. ISBN 978-1803245836. VAUGHAN, Daniel. Analytical skills for AI and data science: building skills for an AI-driven enterprise. Sebastopol, CA: O'Reilly Media, 2020. ISBN 1492060941.

Vedoucí diplomové práce: **Ing. Jan Merta, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **8. listopadu 2023**
Termín odevzdání diplomové práce: **17. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

prof. Ing. Antonín Kavička, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 30. listopadu 2023

Prohlašuji:

Práci s názvem Webová aplikace pro zpracování vědeckých dat
jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 17. 5. 2024

David Kyncl v.r.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu práce, panu Ing. Janu Mertovi, Ph.D., za jeho čas na konzultacích, cenné rady, které vylepšily výslednou aplikaci, a pomoc s řešením problémů, na které jsem při práci narazil.

ANOTACE

Cílem diplomové práce je návrh a implementace webové aplikace s uživatelským rozhraním pro zpracování vědeckých výsledků s cílem získat lepší přehled nad zpracovávanými daty. Aplikace bude poskytovat import dat ze souboru, popisné statistiky zpracovávaného datového souboru a tvorbu grafů. V teoretické části práce budou popsány vybrané statistické metody pro zpracování vědeckých dat a použité technologie. V praktické části student navrhne a implementuje samotnou webovou aplikaci. Backendová část aplikace bude využívat jazyk Python (Django) a vhodně zvolené knihovny. Frontendová část aplikace bude vytvořena pomocí vybraného component-based frameworku.

KLÍČOVÁ SLOVA

Datová analýza, data mining, statistická analýza, vědecká data

ANNOTATION

Goal of this diploma thesis is to design and implement web application with user interface for processing scientific data and gain insight into them. Application will offer importing data from local files, descriptive statistics of the processed data and creating graphs visualizing the data. Theoretical part of the thesis will focus on explaining chosen statistical methods for processing the data and used technologies. Practical part will be dedicated to the design and implementation of application itself. Application backend will be programmed in Python using Django framework and chosen libraries. Frontend will be made in student selected component-based framework.

KEYWORDS

Data analysis, data mining, statistical analysis, scientific data

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM KÓDŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

PDF	Portable Document Format
BE	Backend
FE	Frontend
DB	Database
JSON	JavaScript Object Notation
CSV	Comma-Separated Values
JWT	JSON Web Token
DAAP	Data Analysis and Processing Tool
MVC	Model View Controller
ORM	Object Relational Mapper
HTTP	Hypertext Transfer Protocol
DOM	Document Object Model

ÚVOD

Objem dat v internetu se zvyšuje každý den o stovky milionů terabajtů. Často je výhodné vyhledat informace, které pro nás mohou být důležité nebo zajímavé, ale jsou uvnitř metaforického „moře“ dat, a tak jejich zisk je svoje vlastní disciplína. Dříve byla datová analýza práce převážně pro vědce, kteří zpracovávají data a snaží se nalézt užitečné informace pro jejich výzkum. Dnes provádí datovou analýzu skoro každá větší společnost pro zisk informací, které mohou využít pro zlepšení produktu nebo nabízených služeb. Tento přesun od použití datové analýzy pro zisk informací navíc, k datové analýze pro zisk informací, které řídí celý chod společnosti, způsobil obrovský rozmach tohoto oboru po celém světě. S růstem objemu dat bylo také třeba vylepšovat metody, jak provádět datovou analýzu. V minulosti datová analýza obsahovala převážně statistické metody, v dnešní době ale zde najdeme nejen statistickou analýzu, ale i metody jako shlukování a od nedávna i aplikování umělé inteligence.

Cílem diplomové práce je vysvětlení technik jako je předzpracování dat k odstranění chybných záznamů, data mining neboli získání informací z velkého množství dat, datová analýza, která se zabývá zpracováním těchto informací a následně průzkumná, statistická, shluková analýza pro tvorbu grafů vysvětlující tyto informace. Druhým cílem je vytvořit aplikaci s webovým rozhraním, která nabízí možnost nahrát, zpracovat a vizualizovat vědecká data. K tomu jsou využity praktiky data miningu a datové analýzy.

Aplikace uplatňuje tyto praktiky a kombinuje je do jednotného prostředí, ve kterém lze provést očištění dat a datovou analýzu. Součástí práce je i vysvětlení vybraných konceptů spojených s oborem datové analýzy. Naleznete kapitoly vysvětlující jaké vědecké datasety lze nalézt a jak je zpracovat pro výzkum. Co obnáší zpracování, kategorizace a vizualizace takových dat. Jak nasbírat vlastní data a jak je předzpracovat pro následné použití. Jak provádět různé druhy analýz jako například průzkumnou, statistickou a shlukovou analýzu.

Součástí je i vysvětlení data miningu a co se za jeho pomoci snažíme získat. Zbylé kapitoly se zabývají prvky aplikace, jako jsou diagramy použití a diagramy architektury aplikace, nasazení aplikace v prostředí Docker, zvolené knihovny a frameworky, implementace individuálních modulů aplikace a také uživatelská příručka pro ovládání aplikace.

1 EXISTUJÍCÍ DATASETY A JEJICH VĚDECKÉ ZPRACOVÁNÍ

1.1 Přehled vědeckých datasetů

Následující datasety se často vyskytují ve statistických knihovnách a používají se pro modelování a analýzu, protože jsou to reálná data naměřená v různém množství a v různých letech. Některé splňují všechny předpoklady kvality, jiné pouze nějaké. Jednotlivé předpoklady kvality jsou zmíněné v následující kapitole.

- **„Iris flower“ (Kosatec)** [1]

  V roce 1936 Ronald Fisher nasbíral 50 vzorků ze tří různých typů kosatce, zaznamenal zde délku a šířku kalichu, délku okvětních listů a šířku každé rostliny. Dříve byl používán pro identifikaci nových druhů kosatců. V dnešní době je využíván primárně pro statistické testy a z tohoto důvodu je součástí statistických softwarů nebo knihoven jako je R nebo Python. Jedním z problémů tohoto datasetu je malý počet vzorků na dnešní standard.

- **„Boston housing“ (Ceny domů v Bostonu)** [2]

  V roce 1978 David Harrison a Daniel Rubinfeld nasbírali 506 záznamů o cenách domovů v Bostonu. Zaznamenali vlastnosti jako je například míra kriminality, sociální status lidí zde žijících, medián hodnoty budov a další. Opět je využíván ve statistických knihovnách pro R nebo Python. Používá se pro odhad ceny jiné budovy na základě jejích vlastností.

- **„Wine quality“ (Kvalita vína)** [3]

  Sestaven v roce 2009 autory P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis. Obsahuje chemické vlastnosti červeného a bílého vína jako jsou procenta alkoholu, kyselost, úroveň pH. Nejčastěji je využíván pro předpověď kvality vína na základě jeho vlastností. Opět je dostupný ve statistických knihovnách.

[1] https://www.kaggle.com/datasets/arshid/iris-flower-dataset/data

[2] https://www.kaggle.com/datasets/vikrishnan/boston-house-prices

[3] https://archive.ics.uci.edu/dataset/186/wine+quality

„Titanic passengers“ (Pasažéři Titaniku) [4]

Soupis informací jako je pohlaví, věk, sociální třída a jestli přežili tragédii Titaniku. Je sestaven převážně z dat archívů a seznamu pasažérů lodi. Využíván pro předpověď, zda mají jednotlivé vlastnosti vliv na přežití. Jako předchozí datasety je také dostupný ve statistických knihovnách

- **„World Happiness Report“ (Pocit šťastnosti ve světě)** [5]

  Kolekce dat o šťastnosti lidu z více jak 150 zemí, založena na mnoha faktorech, mezi které patří například očekávaný dožitý věk, sociální podpora, plat. Díky velkému objemu dat se jedná o ideální dataset pro provádění statistické analýzy a ověření hypotéz.

- **„World Bank Data“ (Data od World Bank)** [6]

  Přes 17000 záznamů z více jak 200 zemí, které popisují ekonomický, sociální a hospodářský stav. Podobně jako WHR disponuje velkým rozsahem dat a je tedy ideální na statistickou analýzu a předpověď růstu nebo poklesu těchto indikátorů.

- **„World Health Organization Statistics“ (Světová Zdravotní Organizace – Zdravotní statistiky)** [7]

  Statistiky sesbírané každý rok, ukazující indikátory zdraví a zdravotnictví po celém světě. Výhodou datasetu je jeho aktuálnost, díky novým údajům každý rok. Lze ho použít pro předpověď zdravotního stavu osob žijících v zemi nebo vývoj nemocí.

[4] https://www.openml.org/search?type=data&sort=runs&id=40945&status=active

[5] https://worldhappiness.report/data/

[6] https://datacatalog.worldbank.org/search?q=&sort=last_updated_date%20desc

[7] https://data.who.int

- **„United Nations Statistics Division“ (Statistický úřad Spojených národů)** [8]

  Kolekce všech statistických dat sbíraných Spojenými národy. Jsou zde informace od ekonomiky, přes technologický rozvoj až po zdraví. Splňuje prakticky veškeré předpoklady pro správný dataset, jsou totiž zaznamenávány i způsoby sběru se zdroji a referencemi.

- **„NASA Kepler Exoplanet Candidates“ (Kandidátní planety k životu objevené teleskopem Kepler od NASA)** [9]

  Seznam všech doposud objevených planet teleskopem Kepler, které mají podmínky podobné těm na zemi. Nacházejí se informace o velikosti planety, jak dlouho trvá její doba oběhu a další.

Populárním a veřejně dostupným zdrojem pro datasety je webová stránka www.kaggle.com.

Na stránce Kaggle je někdy složité najít kompletní vědecké datasety, které obsahují reálná data, naštěstí Kalifornská univerzita udržuje repozitář, známý hlavně jako „UCI Repository“, volně dostupných vědeckých datasetů na stránce https://archive.ics.uci.edu.

V době tvorby této práce je zde udržováno přes 660 datasetů, podobně jako Kaggle jsou zde datasety od fyzikálních měření, zdravotních dat, senzorů z výrobních linek, a dokonce i některé z dříve zmíněných jako je „Iris flower“ dataset. Krom datasetů, které univerzita udržuje, nabízí jiné zdroje jako jsou odkazy na statistické úřady, populační statistiky různých zemí a menší repozitáře jiných univerzit. Data mining a datová analýza je jedním z důvodů proč tento repozitář vznikl, umožňuje lépe testovat a porovnávat různé druhy algoritmů nad stejnými daty.

1.2 Zpracování dat pro výzkum

Pod pojmem zpracování dat si lze představit řadu kroků, které musíme podstoupit, než můžeme data označit jako zpracovaná a připravená k použití. Tento proces musí alespoň z části podstoupit každý vědec, datový inženýr či analytik, protože kvalita výsledného výstupu možných analýz přímo závisí na kvalitě vstupních zpracovaných dat. V této kapitole budou jednotlivé kroky zmíněny a objasněny a některé kroky budou dále rozvedeny v následujících kapitolách. Prvním krokem je vždy sběr dat, způsoby sběru a dostupnými zdroji se zabývá

[8] https://data.un.org

[9] https://exoplanetarchive.ipac.caltech.edu

kapitola 3.1. ale co je důležité zmínit zde je, jak vysoký důraz musíme klást na to, odkud data získáváme a jak vypadají, protože najdeme-li nebo sestavíme dataset obsahující pouze špatná data, žádný proces očištění z nich neudělá správná data. Následuje příprava dat také známá jako očištění. V tomto kroku se snažíme eliminovat co nejvíce chybných záznamů v našem datasetu a opravit poškozené nebo chybějící hodnoty. Ideálním výsledkem je dataset bez jakýchkoliv chyb, v praxi je většinou nemožné odstranit veškeré chyby, a proto se snažíme jich odstranit co nejvíce. Detaily, jaké chyby hledáme a jak je můžeme vyřešit, jsou vysvětleny v kapitole 3.4. Po očištění převádíme data do vhodného formátu pro budoucí manipulaci. Nejčastěji se setkáme s formátem CSV (Comma-separated values), který obsahuje hlavičkový řádek s kategoriemi a poté každý záznam na jednom řádku, hodnoty mohou být odděleny středníkem nebo desetinnou čárkou. Obsahují-li data komplexní struktury, další možností je formát JSON (JavaScript Object Notation), který je ve formátu *„jménoZáznamu“:“hodnota záznamu“*. JSON umožňuje přiřadit do záznamu složitější hodnoty za pomocí složených závorek, tímto vytvoříme takzvaný vnořený záznam. Oba formáty jsou dobře strojově čitelné, a to nám usnadní následující kroky. Když dokončíme předchozí kroky, můžeme se pustit do zpracování, zde nad daty provádíme analýzu (v práci se setkáme se shlukovou a statistickou analýzou, obě jsou dále rozebrány v kapitole 4. Výsledkem zpracování jsou získané informace, v podobě grafů, statistických hodnot nebo jiných poznatků, které se výzkumník snažil analýzou získat. Finálním krokem je zaevidovat výsledky analýzy a řádně uložit data pro budoucí použití. Analýzy mohou zabrat velkou část času a je tedy vhodné neztratit jejich výsledky a mít je uložené v takové podobě, se kterou se dá stále pracovat. [1]

1.3 Způsob převedení dat do grafů

Převedení výsledků analýz do grafů je klíčovým krokem chceme-li vysvětlit někdy až abstraktní data lidem, kteří se nemusí pohybovat v oblasti datové analýzy. Z grafů lze zjistit vzory a trendy, které nemusí být na první pohled viditelné v originálních datech. Tato kapitola se zaměřuje na to, jaký typ grafu zvolit na základě znalosti cílové skupiny, která graf obdrží, typu dat, nad kterými budujeme vizualizaci a jaké informace chceme předat. [2]

Než se pustíme do tvoření grafu je vždy důležité znát charakteristiky našich dat. Mezi ně patří jsou-li data kvantitativní nebo kvalitativní, jestli očekáváme diskrétní nebo spojité body, případně v jakých jednotkách jsou hodnoty uvedeny a další vlastnosti. Je také důležité položit si otázku co chceme grafem ukázat nebo dokázat, vědět co má reprezentovat nám pomůže zvolit

správný typ. Kdo graf uvidí a bude ho mít pochopit má velký vliv na naši volbu, výsledky zpracované do sloupcového nebo koláčového grafu pochopí převážná většina osob, ale například krabicový graf (boxplot) může být na první pohled matoucí. [2]

Co se týče jednotlivých typů grafů nejčastěji se setkáme s:

- **Spojový**

  vhodný pro spojité hodnoty, které na sebe navazují například napříč časem

- **Sloupcový**

  dobře popisuje rozdíly mezi kategoriemi dat nebo frekvenci záznamů mezi kategoriemi

- **Bodový**

  vykresluje nespojité body z jedné nebo více kategorií dat, ideální pro velké množství kategorií s jednou hodnotou

- **Histogram**

  vzhledově stejný jako sloupcový, ale reprezentuje změnu jedné hodnoty nejčastěji po dobu času

- **Koláčový**

  vhodný pro zobrazení procenta zastoupení kategorií z pohledu celého datasetu

Tvorba vhodných a vypovídajících grafů je disciplína sama o sobě, ale dobře vytvořené grafy jsou schopny vysvětlit komplexní datasety bez pomoci dodatečných vysvětlivek. [2]

1.4 Jazyky podporující zpracování dat a existující nástroje

Jazyk R je volně dostupné open-source prostředí a zároveň programovací jazyk vytvořený právě pro zpracování dat za cílem provést statistickou analýzu a vizualizovat výsledná data. Nabízí funkce jako je testování hypotéz, časové osy, regresní analýza a další, s možností rozšíření funkcí za pomoc knihoven. [3]

Python je podobně jako jazyk R programovací jazyk, který sice není zaměřen přímo na statistickou analýzu, ale nabízí podobně rozsáhlé knihovny. Python je také volbou pro tento projekt, takže zdůvodnění rozsáhlosti tohoto jazyka v oblasti statistiky a dostupné knihovny bude uvedeno ve vlastních kapitolách. [4]

SAS je zakoupitelný statistický software vyvíjen přímo pro zpracování velice rozsáhlých datasetů. Je často používán ve velkých společnostech nebo státních oddělení. Rozsáhlý objem nabízených funkcí dělá tento software méně přehledný pro nové uživatele v porovnání s jazykem R nebo Python. [5]

Minitab je také zakoupitelný software podobný SAS s více uživatelsky přívětivým rozhraním. Celý software je zaměřen na grafovou vizualizaci dat. Vhodný pro menší společnosti, které nepožadují vysokou komplexitu jako nabízí SAS. [6]

Stata jako předchozí Minitab se snaží nabídnout rozsáhlé statistické funkce v podání jednoduššího rozhraní. Stata se uchytila v ekonomickém sektoru, vědeckém výzkumu a v akademickém prostředí. Jako většina řešení je i Stata placená. [7]

MATLAB je celé prostředí pro výpočet komplexních funkcí, vykreslení grafické prostoru, simulace a datovou analýzu. Rozsáhlost MATLABu umožňuje jeho použití i pro statistickou analýzu, hlavně v případě, že jejím výsledkem budou komplexní grafy. [8]

JMP je další komerční řešení se snahou zlepšit vývoj rychlou a pochopitelnou analýzou dat. Nejčastěji se s JMP můžeme setkat ve společnostech cílených na analýzu a kvalitu kontroly jiných společností. [9]

Společnost IBM nabízí nástroj SPSS (Statistical Package for the Social Sciences), který obsahuje rozsáhlou nabídku statistických funkcí pro analýzu dat. Je cílen na odvětví sociálního, zdravotnického a trhového výzkumu. Nástroj je dostupný k vyzkoušení zdarma na třicet dní, dále je již placený. [10]

1.5 Existující aplikace na vědecké datasety

Ve světě existují aplikace, které plní stejnou nebo podobnou roli jako tvořená aplikace. O některých je vhodné se zmínit, protože jejich funkce byly inspirací, co přidat do vlastní aplikace.

Orange

Open-source software, který kombinuje funkce dataminingu, strojového učení a vizualizace dat. Celý program je ovládán za pomocí „drag-and-drop“ komponent, které kombinují operace datové analýzy. V porovnání s aplikací tvořenou v tomto projektu také nabízí předzpracování dat, statistickou analýzu a následnou vizualizaci. Podporuje i přímé Python skripty, které umožní import jiných knihoven pro datovou analýzu jako je například Scipy. [11]

KNIME

Další ze skupiny open-source programů zaměřený na datovou analýzu, tvoření reportů. Disponuje vlastní platformou pro vizualizaci analýz. Podobně jako program Orange jsou analýzy stavěny přes rozhraní, ale místo „drag-and-drop“ jsou zde propojovány takzvané „nodes“, které reprezentují jednotlivé operace. I zde je nabízeno propojení přímo s jazykem Python pro dodatečné knihovny. [12]

Tableau

Aplikace pro tvorbu interaktivní „dashboard“, kterou lze poté sdílet mezi uživateli bez potřeby dodatečného programování. Jako aplikace Orange i Tableau sdílí „drag-and-drop“ funkcionalitu pro propojení dat a operací. Lze namodelovat obyčejné grafy, ale i komplexní mapy kombinující data z řady zdrojů. [13]

2 ZPRACOVÁNÍ, KATEGORIZACE, VIZUALIZACE

2.1 Předpoklady zpracování vědeckých dat

V praxi lze zpracovávat skoro jakákoliv data, nejedná-li se ale o správně vytvořený dataset, výsledky budou mít nízkou vypovídací hodnotu. Abychom zpracováním dat dospěli k výsledkům v použitelné podobě, je zapotřebí dbát na to, že jsou dodrženy následující předpoklady. [14] [15]

- **Přesnost**
    - naměřené nebo jinak zjištěné hodnoty by měly být dostatečně přesné s ohledem na to, jak budou použity. Měříme-li například malé teplotní rozdíly, bude vyžadována přesnost na desetiny až tisíciny stupně.
- **Vypovídající hodnota**
    - data musí reprezentovat celou oblast, ze které byla získána, a ne pouze její část.
- **Dostatečnost**
    - dataset musí obsahovat dostatek dat, aby ho bylo možné zpracovat, analyzovat a získat očekávané výsledky. Pro většinu analýz je k dosažení dostatečné úplnosti zapotřebí přes deset tisíc záznamů.
- **Transparentnost**
    - pro ověření validity dat by k datasetu měl autor přiložit jakým způsobem byla data získána.
- **Etika**
    - pokud se jedná o data, která mohou obsahovat citlivé údaje (například populační data obsahující jména osob nebo jejich bydliště), je potřeba zajistit, že při zpracování bude dodržena ochrana osobních údajů.

Co se týče aplikace vyvíjené k této práci, přesnost bude určena na základě nahraného datasetu uživatelem. Vypovídající hodnota, dostatečnost a transparentnost bude závislá na uživateli používající aplikaci, nedodržení těchto předpokladů povede pouze k výsledkům, které nelze s jistotou označovat za validní. [14] [15]

2.2 Techniky kategorizace dat

Kategorizace dat je způsob rozdělení dat z datasetu do skupin na základě společných vlastností. Nejčastěji rozlišujeme dva druhy kategorizace, a to jsou manuální a automatizovaná. Za manuální považujeme zpracování datasetu člověkem, který ručně nebo s malou pomocí různých nástrojů, přiřazuje data do jednotlivých skupin. Tento přístup bývá často pomalý a spoléhá na zkušenosti osoby, která data třídí. Je-li dataset zpracováván zkušenou osobou, dá se touto metodou dosáhnout vysoké přesnosti rozdělení. V některých případech je zapotřebí uplatnit alespoň částečnou manuální kategorizaci i při použití automatizované kategorizace, protože dataset obsahuje nestrukturovaná data nebo data s vnitřními vazbami, které nejsou pro kategorizační algoritmy viditelné. Automatizovaná kategorizace využívá dostupných nástrojů a aplikací, které jsou buď schopny plně zpracovat data nebo asistovat a usnadnit manuální kategorizaci. Plně automatizovaná kategorizace bez použití umělé inteligence není příliš přesná a nemá možnost rozlišit malé rozdíly, které mohou tvořit svou vlastní kategorii. Nejčastěji je tedy aplikována kombinace manuální a automatizované kategorizace, kde nástroj nabízí možnosti, jak data zpracovat a člověk, který s ní pracuje, určuje dodatečné parametry. Takto je využita výhoda znalostí uživatele v oboru kategorizace s možností rychlejšího zpracování nabízeného aplikací. Tato práce se velkou měrou bude zabývat právě touto formou kategorizace, kde vyvíjená aplikace bude usnadňovat proces zpracování dat. [16]

2.3 Význam kategorizace pro analýzu

Většina veřejně dostupných datasetů již obsahuje kategorizaci, neboť data jsou například v tabulkovém formátu nebo CSV formátu. Nad nekategorizovanými daty je prakticky nemožné provést analýzu. Řádná kategorizace také zjednodušuje práci s daty, některé hodnoty se seskupí do stejného sloupce, nebo zjistíme, že je můžeme odstranit. Menší dimenze datasetu může také urychlit běh algoritmů při statistické analýze. Data lze vizualizovat i bez kategorizace, ale takové grafy mají minimální vypovídací hodnotu z důvodu chybějících označení os nebo bodů.

2.4 Teorie grafů a vizualizace

Při zpracování dat získaných z reálného života, ať již z průzkumu nebo naměřené v nějaké oblasti, velice často vznikají vazby mezi jednotlivými záznamy, a z nich se poté tvoří datové struktury, které jsme schopni za pomoci teorie grafů vizualizovat nebo i převést do matematické podoby. Vizualizace výsledného datasetu umožňuje snadněji pochopit jeho obsah i lidem, kteří

se nezabývali jeho postupným zpracováním. Avšak má přínos i v raných etapách zpracování datasetu, kde nám pomáhá odhalit duplicitní nebo chybné záznamy, které je stále potřeba odstranit. Uplatníme-li například síťovou reprezentaci na dataset, mohou se objevit vazby a tím pádem i nové kategorie, které nebyly prvotně nalezeny. [17]

2.5 Python pro zpracování dat

Pro zpracování dat je Python jasnou volbou ihned z řady důvodů. Často se do datové analýzy pouští lidé s limitovanou nebo dokonce žádnou znalostí programování, syntaxe jazyka Python je přívětivá, alespoň z pohledu někoho nového v programování. Spustit projekt lze také jednoduše přímo na zařízení přes volně zvolené vývojové prostředí, nebo existují služby jako například Google Collab, který umožňuje spouštět kód v cloud prostředí. Je nutné zmínit také rozsáhlost dostupných knihoven, které umožňují přidat funkcionalitu podle potřeby uživatele. Největší potíží analýzy jakéhokoliv datasetu je prvotní zpracování, které odstraní ty nejčastější chyby. Zde nám množství dostupných knihoven nabízí řadu možností, jak ke zpracování přistoupit. Jednou z nejznámějších je knihovna Pandas, které disponuje nástroji pro načtení datasetů v různých formátech, možnosti automatického očištění a výsledného převodu do finální podoby připravené pro vizualizaci a další použití. Dalším knihovnám, a těm které budou použity při vývoji aplikace, bude vyhrazena vlastní kapitola. Jazyk Python podporuje rozšiřovat výkon do šířky neboli rozdělit práci mezi více zařízení, toto je důležitou vlastností v momentu, kdy se velikost datasetu blíží řádů terabajtů a petabajtů. Tato přívětivost jazyka, dostupnost knihoven a dobrá podpora pro datovou analýzu jen dále rozšířila počet knihoven a ustanovila Python jako jednu z prvních voleb pro zpracování dat. Jednoduchost přizpůsobit aplikaci podle potřeby díky zmíněným výhodám, dělá Python primární volbou pro vývoj tohoto projektu. [4] [18] [19]

3 ZDROJE A PŘEDZPRACOVÁNÍ DATASETŮ

3.1 Metody sběru dat

Tvoříme-li vlastní dataset, je zapotřebí dbát na způsob sběru dat, a to z důvodu, že každá chyba, nepřesnost nebo chybějící záznam v tomto prvním kroku ovlivní složitost následujících kroků anebo dokonce sníží kvalitu výsledného datasetu. Sběr dat často rozdělujeme na primární a sekundární. Primární sběr je ten, kde data získáváme sami, například přes průzkumy, vlastní experimenty nebo sledování a měření. Výhodou vlastního sběru je možnost zaměřit se na přesně ten typ dat, které potřebujeme a ručíme za kvalitu nasbíraných dat. Způsob, jak data získat, závisí na typu dat, která se snažíme získat, ale nejčastěji se setkáme s již zmíněnými metodami v popisu primárního sběru. Průzkum může být prováděn například v podobě online nebo tištěného formuláře, jehož cílem je získat odpovědi. Tato metoda spoléhá na pravdivé odpovědi respondentů, bez tohoto předpokladu není možné považovat data za vypovídající. Vlastní pokusy mohou být prováděny v kontrolovaném prostředí, tímto zaručujeme vypovídající hodnotu a pokud bylo pečlivě měřeno, zaručujeme i přesnost. Bohužel vlastní pokusy pro zisk dat jsou často drahé, časově náročné a nelze je dělat ve velkém rozsahu. Relativně levným a dostupným způsobem měření je pasivně sledovat a zaznamenávat jevy, o kterých chceme získat data, zde je problémem velká časová náročnost, nemůžeme-li ovlivnit, jak často se jevy projevují. Sekundární sběr označuje data získaná již z existujících zdrojů, můžeme tímto doplnit primární sběr, ale je také možné vybudovat dataset pouze z těchto dat. Mezi nejčastější zdroje patří již existující datasety, které lze použít celé nebo vybrat jejich část a vytvořit subset, data veřejně dostupná od vládních institucí jako je například statistický úřad, veřejně dostupná nebo zakoupitelná data od společností, které je nasbíraly ve svých službách. Ať je dataset stavěn jednou z metod nebo kombinací metod, je důležité dbát na předpoklady vypsané v kapitole 2.1. [20]

3.2 Diskretizace dat

Pod pojmem diskretizace bereme techniky, které redukují počet hodnot pro danou spojitou proměnnou. Počet hodnot snižujeme dělením všech hodnot na intervaly. Každý interval je pak popsán vlastním „titulkem“, který nám reprezentuje reálné hodnoty. Podobně jako snížení počtu vlastností, metoda, kterou se zabývá následující kapitola, využívá diskretizaci pro zjednodušení a zmenšení originálního datasetu. Typy diskretizace rozlišujeme podle toho, jakým směrem prochází data, to je od shora dolů nebo od spodu nahoru. Aplikujeme-li metodu od shora dolů,

nejprve se najdou počáteční body, ve kterých originální dataset rozdělíme a vytvoří se prvotní intervaly, dělení intervalů se pak opakuje, než dosáhneme požadovaného počtu intervalů. Přístup od spodu nahoru naopak považuje všechny hodnoty za možné hranice intervalů, takže některé z nich spojí a tím nám vzniknou počáteční intervaly, na kterých rekurzivně tento přístup uplatňujeme a spojujeme interval s intervalem. Když zachováme hierarchii, která se buduje při tvoření intervalů, vytvoří se stromová struktura, kterou lze použít pro dodatečné popsání dat nebo převedení číselných intervalů na kategorické označení intervalů. Příkladem je diskretizace platů zaměstnanců, kde nadřazené intervaly před jejich rozdělením označíme jako *nízký, střední, vysoký.* Samozřejmě jako jiné algoritmy redukující nějakou z dimenzí datasetu, i zde ztrácíme detaily, někdy je tato ztráta přijatelná, pokud nám diskretizace zjednoduší následnou analýzu. [20]

3.3 Snížení počtu vlastností

Průběhem času, jak se zlepšuje výkon hardwaru, roste úložný prostor a vzniká více sofistikovaný software, přivádí to autory datasetů ke skladování více a více informací. Setkáme se s myšlením „čím více, tím lépe“, příkladem těchto nadbytečných údajů by byly například kroky nachozené po supermarketu zákazníkem s příručním skenerem. Dostáváme pak datasety, které vypadají rozsáhle, ale doopravdy je zde pouze hrst záznamů, kde každý má velké množství atributů. Skladovat veškeré informace eliminuje potřebu řešit otázku „Potřebujeme tyto informace?“, ale výsledně je to více na škodu než k užitku. Vrátíme-li se k příkladu supermarketu, když se budeme snažit nalézt spojitost mezi typem zákazníka a tím co nakupuje, zjistíme rychle, že nacházíme minimální podobnost, protože prohledáváme obrovské množství atributů, kde většina z nich nemá vliv na to, co hledáme. V lepším případě se pouze prodlouží výpočetní čas při zpracování, v horším případě budou ovlivněny výsledky. Pokud nesestavujeme vlastní dataset, nemáme možnost ovlivnit co v něm je, dostáváme se k technice snížení počtu vlastností. Princip snížení počtu vlastností je jednoduchý, přiřadíme každé vlastnosti hodnotu, jak hodně je důležitá pro to, co se snažíme zjistit a určíme, kolik nejvyšších vlastností nebo kolik procent ze všech vlastností ponecháme. Volba, jak vysokou hodnotu musí vlastnost mít, abychom jí považovali za důležitou nebo kolik vlastností ponechat, není jednoznačná, je potřeba nastavit tyto hranice na odhadnuté počáteční hodnoty a poté testovat, jak se výsledek mění, když hodnoty posouváme. Lze se i setkat s metodou „kolapsu“ vlastností, to znamená, že jsou si některé vlastnosti tak podobné, že je možné je spojit do jedné vlastnosti. [20] [21]

3.4 Čištění dat a ověření kvality

Po sestavení datasetu je dalším krokem jeho očištění, při kterém je zapotřebí odstranit špatné hodnoty zanesené lidskou chybou, chybným měřením nebo jiným vlivem. Při hledání chyb se zaměřujeme na pět hlavních typů chyb a to jsou:

Duplicitní záznamy

Jakýkoliv záznam, který reprezentuje stejné měření a byl chybně zapsán dvakrát, případně i vícekrát. Duplicitní záznamy hledáme pomocí podobnostních funkcí, to jsou funkce, které se snaží přiřadit hodnotám určité skóre, jak moc se sobě podobají. Můžeme provádět porovnání celého záznamu s jiným nebo pouze části záznamu, převážně se setkáme s částečným porovnáním, a to z důvodu, že pouze číselné hodnoty nebo kategorické hodnoty nelze s jistotou označit jako duplicitní, a proto porovnáváme textové hodnoty. [22] [23] [24]

Chybějící hodnoty

Máme-li záznamy (řádky v tabulkovém zobrazení), které mají více hodnot (sloupce v tabulkovém zobrazení), jedná se chybu, kde potřebná hodnota musí být vyplněna a není. K vyřešení tohoto problému se přistupuje dvěma způsoby, a to je odstranění celého záznamu nebo nahrazení nejčastější/průměrnou hodnotou. Odstranění záznamu je nejjednodušší, protože odstraníme každý záznam, ve kterém chybí jakákoliv hodnota. Jedná se o „bezpečnou“ metodu, a to z důvodu, že data nedoplňujeme odhadovanými hodnotami. Na druhou stranu, máme-li velké množství záznamů s chybějícími hodnotami nebo nízký počet záznamů, jejich odstranění snižuje spolehlivost výsledků. Z tohoto důvodu využíváme metodu odstranění pouze v případě, že počet chyb je nízký, jinak se nepoužívá.

Jak plyne z názvu nahrazení nejčastější/průměrnou hodnotou opravujeme chybějící hodnoty odhadem této hodnoty. Pro kategorické hodnoty se používá nejčastější hodnota a pro číselné hodnoty průměrná hodnota. Pro použití nejčastější hodnoty je důležité dbát na zastoupení jednotlivých hodnot v celém rozsahu dat. Když se nějaká hodnota vyskytuje v 70 % záznamů a zbylých 30 % obsahuje jiné hodnoty, můžeme s dobrou jistotou doplnit tuto hodnotu i do chybějících záznamů, aniž bychom z velké části ovlivnili spolehlivost výsledků. Jakmile se rozdělení hodnot blíží k vyváženým procentům, může doplnění nejčastější hodnotou negativně ovlivnit výsledky.

Při doplnění číselné hodnoty průměrem z celého datasetu záleží hlavně na tom, jak velký rozsah hodnot se zde nachází, a jestli dataset obsahuje extrémní hodnoty nebo datové anomálie. V obou

případech je potřeba sledovat, zda hodnota, kterou se snažíme doplnit, není ovlivněna jinými faktory. Příkladem této situace by bylo doplnění chybějící hodnoty v kategorii krevního tlaku pacientů, pokud data obsahují převážně mladé osoby, ale snažíme se odhadnout tuto hodnotu pro starší osobu, průměrná nebo nejčastější hodnota bude v tomto případě s velkou pravděpodobností chybná a povede k závažně nepřesným výsledkům. [23] [24]

Chybné datové typy

Při načtení datasetu jsou všechny hodnoty převedeny do datových typů reprezentující tuto hodnotu, byla-li hodnota špatně uložena do datasetu, může být interpretována jako jiný datový typ. [23] [24]

Chyby formátování

Nejčastější chyba v případě jiného kódování (například ASCII oproti UTF-8) souvisí se znaky, co nejsou v kódové sadě jsou převedeny na jiné, které nereprezentují správná data. Také se vyskytuje problém desetinné čárky a tečky. [23] [24]

Datové anomálie

Datové anomálie jsou takové hodnoty, které doopravdy nemohou být naměřeny nebo zjištěny, vznikají lidskou chybou při měření, vadou přístrojů, chybou při ukládání a v mnoha dalších situacích. [23] [24]

3.5 Normalizace a standardizace

Normalizace umožňuje posunout a změnit rozmezí, ve kterém existují data, nejčastěji normalizujeme na hodnoty mezi nulou a jedničkou. Příklad využití je pro dataset obsahující hodnoty v řádech setin a zároveň hodnoty v tisících až milionech. V takové situaci by jakékoliv porovnání hodnot mezi těmito kategoriemi a následné vykreslení do grafu vyprodukovalo jednostranný graf, který nepřináší žádné informace. Když data normalizujeme, ztratíme možnost používat přímé hodnoty, ale můžeme lépe provádět relativní porovnání kategorií. Standardizace transformuje data tak, aby je šlo porovnat s jinými daty, které mají jinou škálu. K tomu je potřeba spočítat střední hodnotu a směrodatnou odchylku, poté od každé hodnoty odečteme střední hodnotu a vydělíme směrodatnou odchylkou, tak získáváme standardizovanou hodnotu. [22]

4 SHLUKOVÁ, STATISTICKÁ A PRŮZKUMNÁ ANALÝZA

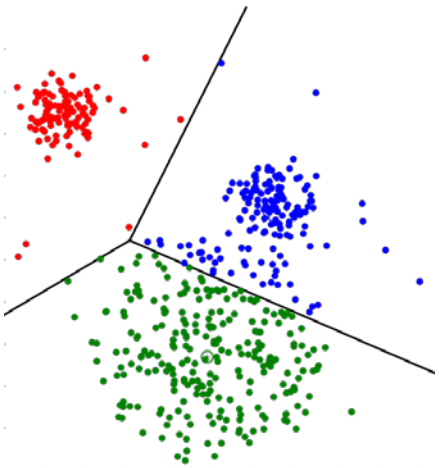
4.1 Shluková analýza a její algoritmy

Při shlukové analýze rozdělujeme celek na menší skupiny, do nich přidělujeme prvky z celku na základě jejich podobnosti, každá skupina potom tvoří jeden „shluk“ z anglického slova „cluster“ (dále v práci bude používáno slovo cluster). Podobnosti prvků využíváme v případě, že je nelze rozdělit do stejných skupin (nelze tedy provést jednoduchou kategorizaci, která byla vysvětlena v kapitole 2.2.), protože nesdílí jasně určující vlastnost. Příkladem této situace může být dataset vlastností pracovníků a cílem analýzy je seskupit je do několika druhů oddělení. Ne každý pracovník bude mít naprosto stejné vlastnosti, aby mohl být přiřazen do skupiny, musí se nejdříve nalézt podobnosti a na jejich základě vytvořit cluster. Analýza tedy označuje zpracování datasetu jedním nebo více algoritmy, které se snaží efektivně vytvořit co nejpřesnější clustery. Clustery jako takové můžeme ještě dělit na jejich modely, kde model reprezentuje, podle čeho jsou prvky seskupovány k sobě. Mezi známé modely patří:

- **spojový** – clustering dat založen na předpokladu, že data poblíž sebe v prostoru mají více společného než data daleko od sebe

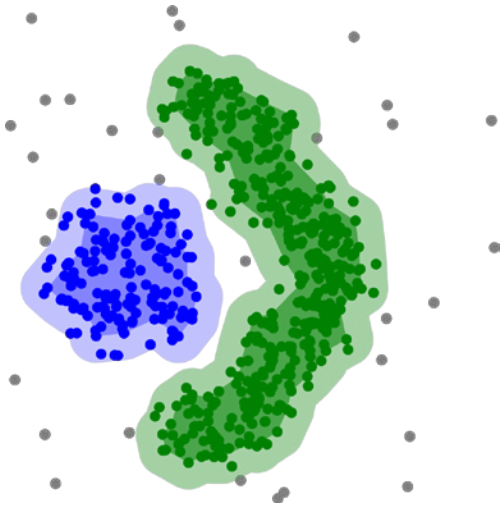
Obrázek 1: Příklad spojového shlukování [28]

- **středový (geometrický střed)** – podobnost dat je určena tím, jak blízko jsou data ke geometrickému středu clusteru



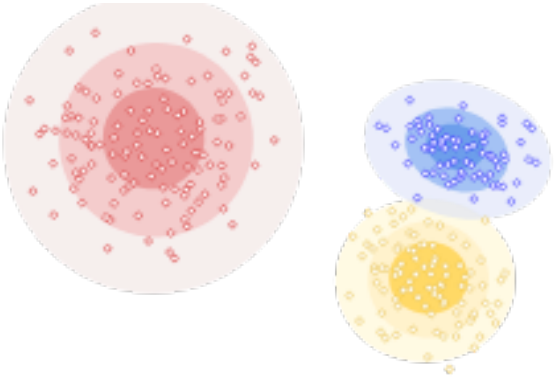
Obrázek 2: Příklad středového shlukování [28]

- **hustotový** – pokud je v části datasetu nashromážděno velké množství dat, stanou se tato data clusterem



Obrázek 3: Příklad hustotového shlukování [28]

- **statisticky rozložený** – clustery jsou tvořeny na základě pravděpodobnosti, že data patří do stejného pravděpodobnostního rozdělení



Obrázek 4: Příklad statisticky rozloženého shlukování [29]

Algoritmů existuje mnoho, a ne všechny odpovídají nějakému z výše zmíněných modelů, proto se budeme zabývat pouze pár algoritmy, se kterými se lze setkat nejčastěji.

Spojově orientované nebo také hierarchické algoritmy využívají předpokladu, že data blízko u sebe mají více společného a čím dále jsou, tím méně mají společného. Při použití spojově-orientovaných algoritmů je důležité mít velice dobře očištěný dataset a to hlavně od hodnot, které se blíží limitům měření nebo je i přesahují. Takovéto záznamy mohou způsobit, že se z nich stane celý nový cluster nebo propojí dva clustery do jednoho.

Shlukování na základě hustoty spojuje data do clusteru na místech, kde se dat nachází mnoho blízko u sebe. Hodnoty daleko od těchto bodů vysoké koncentrace se považují za okrajové pro cluster nebo ani nepatří do žádného clusteru. Takové hodnoty považujeme za neplatné, také nazývané jako „šum“. Aby algoritmy byly schopné rozeznat clustery, potřebují poznatelný pokles hustoty. Nejčastěji je hustota určena počtem objektů v dané oblasti.

Statistickým rozdělením-orientované algoritmy jsou založeny na pravděpodobnostním rozdělení. Setkáme se s nimi nejčastěji ve zpracování statistických hodnot, a to z důvodu, že takovýto typ hodnot velice často odpovídá nějakému pravděpodobnostnímu rozdělení. Oproti ostatním typům algoritmů se musí omezit doba běhu, hrozí jinak že komplexita modelu překročí hranici a začneme ztrácet výslednou vypovídající hodnotu. Nejvyšší úspěšnost mají tyto algoritmy nad náhodně generovanými datasety, ty totiž také odpovídají pravděpodobnostním rozdělením.

Středově orientované algoritmy zavádějí vektor, kolem kterého se tvoří cluster. Cílem je přiřadit hodnoty k nejbližšímu středu. Aplikováním středové orientace v algoritmech pro shlukovou analýzu dostáváme „k-means“ shlukování, kde *k* je počet clusterů. Nalezení nejbližšího středu je bohužel NP problém neboli nalezení přesného řešení je v polynomiálním čase nemožné, proto všechny algoritmy pouze odhadují řešení. Mnoho algoritmů vyžaduje, aby *k* bylo určeno ještě před spuštěním algoritmu, to pro valnou většinu datasetů nelze zjistit. [25] [26] [27]

4.1.1 K-means algoritmus

Někdy překládaný jako algoritmus *k-průměrů,* je nehierarchický algoritmus pro shlukovou analýzu. Jak již bylo zmíněno, předpokladem pro použití k-means (nebo jeho rozšířeních) vyžaduje znalost o počtu clusterů z důvodu, že přesný počet často neznáme, ale přesto chceme algoritmus použít, musíme jejich počet odhadnout. Při inicializaci algoritmu zvolíme *k* středů nebo častěji geometrických středů. Tato počáteční volba může ovlivnit výsledky, je tedy důležité spustit algoritmus několikrát s různými počátečními středy. V dalším kroku jsou všechny datové body přiřazeny ke středům a proběhne aktualizace těchto středů, aby více odpovídaly průměru datových bodů. Protože se jedná o iterativní algoritmus, opakujeme tento průběh do té doby, než se nám clustery stabilizují neboli jejich umístění se přestane poznatelně měnit. Z tohoto algoritmu se vyvinulo mnoho dalších, které se snaží zrychlit nalezení clusterů, vyřešit problematiku výběru *k* nebo zlepšit přesnost v případě, že data nelze jednoznačně spojit do clusterů. Mezi ně patří například [30]:

- **k-medians**
  - stejný přístup ke shlukování, ale místo průměru se používá medián.
- **k-means++**
  - algoritmus rozšiřující standardní k-means s cílem nalézt optimální počáteční hodnoty pro středy clusterů.
- **hierarchické k-means algoritmy**
  - rozdělují clustery a tvoří tak hierarchii a snaží se určit optimální hodnotu *k.*
- **fuzzy k-means**
  - rozdílný přístup k tomu kam jednotlivé hodnoty patří, místo aby každá hodnota byla pouze v jednom clusteru, při použití tohoto algoritmu mohou hodnoty patřit k více clusterům a u každého mají míru pravděpodobnosti, jestli k němu patří.

4.2 Průzkumná analýza

Při otevření datasetu často zjistíme, že i když nás jeho obsah zajímá, nevíme vlastně na jaké vlastnosti máme uplatnit statistickou nebo shlukovou analýzu. Dobrým prvním krokem je tedy provést průzkumnou analýzu, která má za cíl zobrazit co nejvíce informací, trendů, vztahů a vzorů obsažených v datasetu. Běžně se začíná shrnutím, jaké záznamy zde máme, jaký typ hodnot se v nich nachází, a poté uděláme vizualizaci nejčastěji v podobě sloupcových grafů. Příkladem informací, které získáme, je kolik číslic nebo desetinných míst mají hodnoty jednotlivých vlastností, kolik a jaké druhy kategorických hodnot zde jsou, jsou-li hodnoty spojité nebo diskrétní. Setkáme se také se situací, kde analyzujeme dataset hodnot, který jsme v minulosti ještě nezpracovávali, ale typ vlastností, které zde jsou, se silně podobá jinému již zpracovanému datasetu. Tvoříme si pak hypotézy o vztazích nebo vzorech, které v datech existují. Průzkumná analýza nám může tyto hypotézy potvrdit, případně vyvrátit, v obou situacích nám tento fakt usnadní práci, protože nebude zapotřebí provádět dodatečnou analýzu. [23]

Field	Sample Graph	Type	Min	Max	Mean	Std. Dev	Skewn...	Median	Mode	Unique	Valid
State		Set	--	--	--	--	--	--	WV	51	3333
Account Length		Range	1	243	101.065	39.822	0.097	101	105	--	3333
Area Code		Set	408	510	--	--	--	--	415	3	3333
Intl Plan		Flag	--	--	--	--	--	--	no	2	3333
VMail Plan		Flag	--	--	--	--	--	--	no	2	3333
VMail Message		Range	0	51	8.099	13.688	1.265	0	0	--	3333
Day Mins		Range	0.000	350.800	179.775	54.467	-0.029	179.400	154.000*	--	3333
Day Calls		Range	0	165	100.436	20.069	-0.112	101	102	--	3333
Day Charge		Range	0.000	59.640	30.562	9.259	-0.029	30.500	26.180*	--	3333

Obrázek 5: Ukázka průzkumné analýzy [23]

4.3 Statistická analýza

Pod pojmem statistická analýza si lze představit celý proces získání užitečných informací od deskriptivní statistiky a funkcí až po vykreslení hodnot do grafů.

Deskriptivní statistika je označení pro skupinu metod, které mají zobrazit hlavní vlastnosti datasetu. Nabízí nám stručný přehled, co za hodnoty se v datasetu nachází.

Rozlišujeme dva typy dat, na které můžeme uplatnit deskriptivní statistiku, těmi jsou kvantitativní a kvalitativní. Kvantitativní data mají číselnou hodnotu, vlastnosti deskriptivní statistiky nám zde zobrazí obecné rozdělení, hodnoty mimo rozsah a další. Oproti tomu kvalitativní mohou být v textové podobě (pravda/nepravda) nebo číselné hodnotě rozlišující stavy (1/0 pro pravda/nepravda) a při analýze zjistíme nejčastější hodnoty.

Měříme-li středovou tendenci, jsou to hodnoty, které reprezentují časté, typické nebo průměrné hodnoty. Získáváme tak jednu hodnotu z celého datasetu. Nejčastěji se zjišťuje aritmetický průměr, medián a modus.

Variabilita určuje míru rozptylu od středové tendence. Typické vlastnosti jsou rozsah, rozptyl, směrodatná odchylka a mezikvartilový rozsah. Rozsah měříme jako rozdíl mezi nejvyšší a nejnižší hodnotou. Rozptyl je střední hodnota kvadrátů odchylek od střední hodnoty. Jako směrodatná odchylka se bere odmocnina z rozptylu náhodné veličiny. Mezikvartilový rozsah je počítán stejně jako normální rozsah, ale pouze pro prostřední dva kvantily dolní a horní kvantil od mediánu. [24]

4.4 Regrese

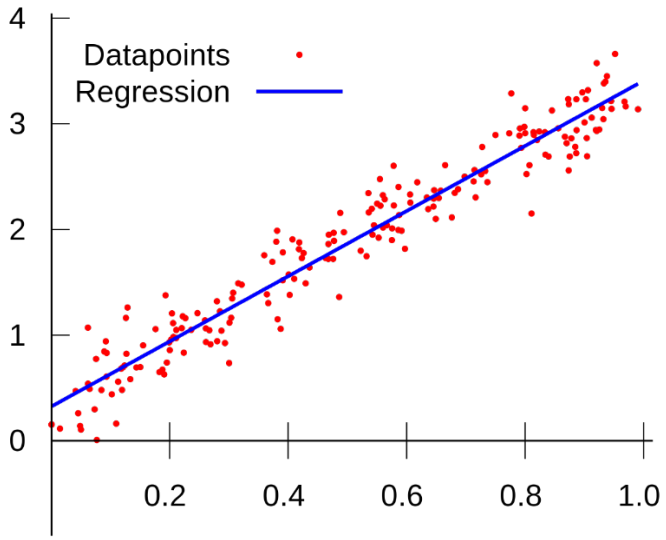
Druh statistické analýzy ke zjištění vztahu náhodné veličiny k jiným veličinám (neboli vztah závislé proměnné na nezávislé proměnné) a tím předpovědi jakých hodnot bude náhodná veličina nabývat. Cílem je namodelovat vztah mezi proměnnými tak, aby výsledná křivka odpovídala co nejvíce reálné křivce. Při tvorbě regresní křivky se odhadují koeficienty výpočtu. Koeficienty získáváme pomocí metod jako je metoda nejmenších čtverců. Regrese pomáhá nahlédnout do vztahů mezi proměnnými, vytvořit předpovědi, jak se hodnoty dané proměnné budou vyvíjet, testovat hypotézy kladené na data a identifikovat vlastnosti, které proměnnou ovlivňují. [24]

Mezi nejčastější typy regrese patří lineární, exponenciální a polynomická.

Lineární regrese je nejzákladnější typ regresní křivky, předpokládá že vztah mezi veličinami je lineární neboli výsledná křivka je přímka. Použití lineární regrese je vhodné na data, která rostou spojitě.

Exponenciální regrese, jak již vyplývá z názvu, je uplatňována v případě, že data rostou exponenciálně.

Pomocí polynomické regrese lze vytvořit křivku, která nepředpokládá lineární nebo exponenciální vztah mezi proměnnými. Za pomocí vyšších polynomických stupňů. Typické stupně, se kterými se lze setkat jsou dva, tři a čtyři.

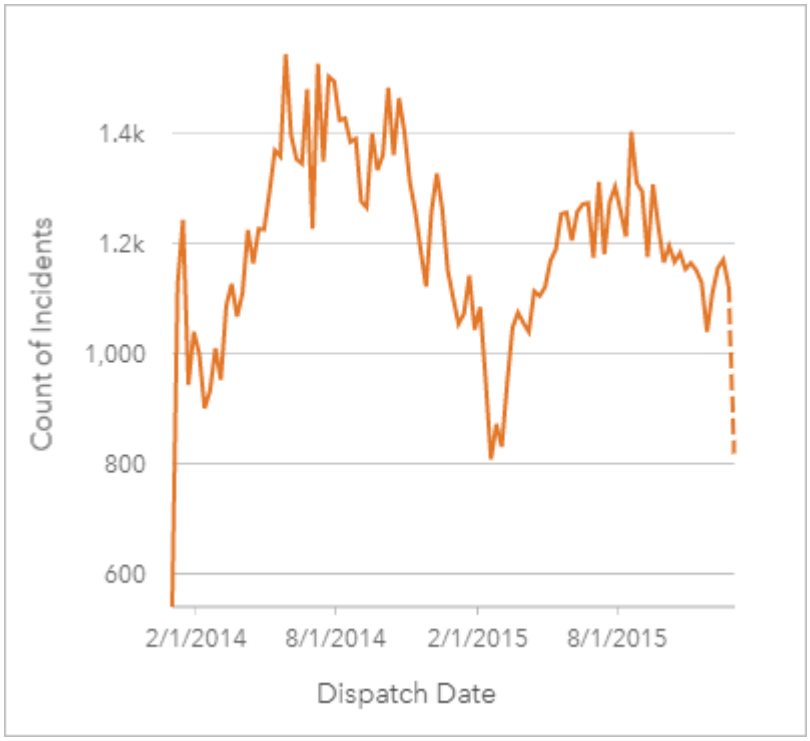


Obrázek 6: Ukázka lineární regrese [31]

4.5 Časová osa

Pod pojmem časová osa si lze představit vyobrazení hodnot, které nastaly v daný čas, nebo po sobě jdoucí události. Body vynesené do časové osy můžeme rozdělit na tři typy, spojité, ordinální a diskrétní. Spojité hodnoty navazují na sebe a jsou v pořadí v jakém vznikly nebo byly zaznamenány. Ordinální body pouze určují pořadí a rozmezí mezi nimi neznačí žádnou časovou jednotku. Hodnoty diskrétní určují pořadí i časovou jednotku, ale čas je ve velkých úsecích (jako jsou například roky nebo desetiletí) bez jakýchkoliv záznamů mezi úseky, které by z nich dělaly spojité hodnoty. Hodnoty vynesené v ose můžeme také odlišovat, jestli se jedná o přímý záznam nebo interval. Přímé záznamy reprezentují přesný časový moment,

například východ slunce v danou hodinu nebo 0 nebo den konání akce. Intervalové záznamy mají na ose dva body, mezera mezi nimi označuje časový úsek, jak dlouho jev trval. Frekvence bodů na ose je označována jako granularita, čím více bodů, tím vyšší granularita a naopak. Do časové osy lze kromě bodů zanést informace jako je křivka trendu, která umožňuje předpovídat, jak se bude časová osa dále vyvíjet. Pokud časová osa obsahuje záznamy z rozsáhlého období jako je třeba každý týden po celý rok, je možné rozdělit osu na sezónní části. Pro každou sezónu lze pak vytvořit trendovou křivku a zjistit trend pro každou sezónu. [32]



Obrázek 7: Ukázka časové osy [33]

4.5.1 Zvýšení vzorkování

Způsob zvýšení počtu bodů pomocí interpolace hodnot z existujících hodnot. Princip spočívá v použití jedné z interpolačních metod jako jsou například lineární, polynomická, „spline“ (to je speciální verze polynomické metody, která je rozdělena na intervaly také nazývané subdomény). Přesnost interpolovaných hodnot závisí na počáteční granularitě časové osy a jak velký úsek chceme převzorkovat. Časté využití je pro odhad hodnot mezi existujícími body nebo pro převedení na stejné vzorkování jako jiná časová osa, se kterou chceme převzorkovanou porovnat. [34]

4.5.2 Snížení vzorkování

Podobný koncept jako zvýšení vzorkování, avšak místo interpolace nových hodnot se jedná o spojení existujících hodnot do menšího počtu. Typicky se setkáme se snížením vzorkování pomocí střední hodnoty, mediánu, minimální nebo maximální hodnoty. Aplikace snížení vzorkování bývá v situaci, kdy modelovaná data mají příliš vysokou granularitu a analýza trvá dlouhou dobu nebo její výsledky jsou ovlivněny touto vysokou úrovní detailu. Lze také využít pro srovnání vzorkování s jinou časovou osou. [35]

4.6 Krabicové grafy

Jednou z možností, jak vizualizovat rozložení a popisnou statistiku datového souboru, je uplatnění box plotů neboli krabicových grafů. Graf se skládá z kvantilů, mediánu, tzv. vousů a případně odlehlých hodnot.

Vousy (vycházející z názvu „whiskers“) označují variabilitu dat mezi vrchním a spodním kvantilem. Odlehlé hodnoty jsou ty, které jsou mimo kvantily a může se jednat o anomální hodnoty nebo chyby měření.

Tento způsob zobrazení dat je výhodný, když data nenesou předpoklady normálního rozdělení. Z grafů lze tedy vyčíst střed dat, rozptyl hodnot, šikmost, kvantily (v rozmezí od prvního po třetí), rozsah mezi kvantily (označováno IQR) a již zmíněné odlehlé hodnoty. Jedním z problémů krabicových grafů je méně intuitivní vizualizace na pochopení oproti například rozložením distribuční funkce. [36]

5 DATA MINING

5.1 Koncept a co vlastně je data mining

Když se mluví o data miningu, jedná se o situaci, kde máme velké množství dat například z databáze, tato data nás zajímají pro výzkum, ale nevíme, v jakém stavu jsou a jestli z nich půjde vytvořit vhodný dataset. Dobrou otázkou je, co se považuje za velké množství dat, samozřejmě vždy to závisí na velikosti společnosti, která data vlastní, ale řádově se pohybujeme v jednotkách terabajtů až petabajtů. Podobně tedy jako když analyzujeme hotový dataset a hledáme tam podstatné informace, zde hledáme podstatná data, ze kterých později můžeme získat informace. Často se při data miningu setkáme s pojmem extrahování vlastností a není náhodou, že tento pojem existuje i v oblasti strojového učení, tato odvětví se často prolínají. Data mining jako technika existuje desítky let, ale dříve to nemělo tento název, pracovalo se s menším objemem dat a považovalo se to za formu analýzy. S rozmachem webu, který sám o sobě je obrovskou kolekcí dat, technologií a přístrojů používaných každý den, které sbírají všemožná data, je ve společnostech a výzkumných centrech data mining potřeba více než kdy jindy. Kromě rostoucího objemu dat, která sbíráme, se zájem o tuto oblast zvýšil díky několika faktorům. Počítačový výkon a kapacita disků se za poslední desetiletí mnohonásobně zvýšily, to umožňuje data skladovat dlouhodobě a zpracovat je v přiměřeném čase. S lepším hardwarem, který je schopný data pojmout, vzniká i software, který nabízí funkce, jak je zpracovat. [23] [37] [38]

Data mining má nejčastěji čtyři fáze. První je všeobecné očištění dat, stejně jako když očišťujeme dataset se snažíme odstranit chybné a chybějící hodnoty, opravit nekonzistentní formát a podobné. Do této fáze také zahrnujeme veškeré třídění, filtrování a kombinaci dat z různých zdrojů. Můžeme přidat i nová data vytvořená základními agregačními operacemi nad originálními daty. Opět platí, čím lépe očištěná data, tím lepší výsledky. Nad tímto celkem aplikujeme metody data miningu. Výsledky této fáze jsou analyzovány pro nalezení vzorů/trendů v datech. Finálním výstupem jsou nejčastěji grafické a tabulkové reprezentace těchto trendů. [23] [37] [38]

5.2 Typické cíle data miningu

Samozřejmě, že čeho má data mining dosáhnout záleží na osobě nebo skupině osob aplikující tuto metodu, ale setkáme se skupinou kategorií, které obecně popisují cíl. Patří mezi ně popis, odhad, asociace, předpověď, klasifikace a shlukování.

Mining s cílem popsat data se snaží pochopitelně vysvětlit co za vzory se nachází v datech. Jedná se o vzory, které se špatně vyjadřují statistickou analýzou, protože ne vždy spadají do statistického rozdělení. Je to také jeden z nejčastějších typů. Vhodným příkladem použití popisného data miningu jsou data o pracovní morálce v různých odděleních společnosti, která je v každém oddělení rozdílná. Hledáme vzory a trendy, které nám budou schopny popsat, co tyto rozdíly způsobuje, například individuální přístup každého vedoucího oddělení k zaměstnancům a podobně.

Jak již plyne z názvu, data mining pro odhad se snaží odhadnout číselnou hodnotu na základě znalosti této hodnoty z jiných záznamů a faktorů, které ovlivňují tuto hodnotu. Abychom byli schopni provést odhad, musíme mít větší množství záznamů, které tuto hodnotu již mají. Poté pro každý nový záznam můžeme hodnotu odhadnout. Faktory ovlivňující odhadovanou hodnotu jsou také typicky číselné, ale mohou být i slovní (kategorické) jako je „senior, dospělý, dítě“ a jiné. Jednoduchým příkladem je odhad trefených branek fotbalistou v zápase, kde faktory jsou proti komu hrají, kde se zápas hraje, jaký je brankář a podobné.

Při asociačním data miningu hledáme vlastnosti dat, které spolu souvisí. Asociaci určujeme jako procentuální pravděpodobnost nějakých vlastností spolu souvisejících a tu poté hodnotíme takzvanou hodnotou spolehlivosti. Pro příklad si představme 500 účastníků festivalu, kde 320 z nich si zařídilo pobyt přes noc poblíž akce, z těchto 320 si 70 sjedná na druhý den odvoz domů, máme tedy pravděpodobnost asociace 64 % (získáno výpočtem (320/500) * 100) se spolehlivostí 22 % (získáno výpočtem (70/320) * 100).
Předpověď se dá vyjádřit jako odhad delší doby do budoucnosti. Sdílí tedy stejné požadavky jako odhad, ale musíme více aproximovat hodnoty, které se postupem času mění, čím více dopředu předpovídáme, tím menší bývá přesnost. Jako předpověď lze považovat například vypočítání míry inflace rok napřed.

Klasifikace sdílí základ s odhadem, ale zde je potřeba slovní (kategorický) typ hodnoty, kterou chceme „odhadovat“ neboli v tomto případě klasifikovat. Platí stejná pravidla, že potřebujeme záznamy, které obsahují hodnotu a faktory, které jí ovlivňují. Uplatnit klasifikaci lze na rozdělení studentů do typů tříd, podle speciálních potřeb.

Shlukování bylo vysvětleno v kapitole 4.1. Shluková analýza a její algoritmy, ale pro rekapitulaci zde zmíníme princip shlukování a jak se liší od ostatních typů data miningu. Jedná se spojování záznamů do skupin podle jejich podobnosti. Shluk neboli „cluster“

je skupina podobných záznamů. Setkáme se se shlukováním v případě, že chceme rozdělit data na menší celky podle podobnosti a nad těmito celky provedeme jiný typ dataminingu. Jiné uplatnění této techniky je například při rozdělování kandidátů na pracovní pozici podle podobných vlastností v jejich životopisu. [39] [40] [41]

5.3 Uplatnění data miningu v praxi

Pro představu, jak často je data mining kolem nás, stojí za zmínku, v jakých odvětvích se používá a jak. [23] [37] [38] [39]

- **Obchodní sféra**
    - Zjišťování, jaký typ zákazníků nakupuje, jaké zboží a nabídnout jim cílené slevy na drahé produkty, které se snaží prodejce prodat.
- **Bankovnictví**
    - Sledování, který typ transakcí klient typicky provádí s cílem zjistit, jestli někdo cizí nezneužívá jeho účet.
    - Zjistit pravděpodobnost, že zájemce o půjčku bude schopen vše splatit ve stanoveném termínu na základě analýzy jeho příjmů v minulosti.
- **Televizní společnosti**
    - Zjistit jaká věková skupina sleduje dané vysílání a nabízet reklamy, které mají šanci tento typ publika zaujmout.
- **Zdravotnictví**
    - Předpovědět jaká je šance, že pacient má predispozice nakazit se nemocí.
    - Zjistit vliv genetické sekvence pacienta na pravděpodobnost genetické nemoci.
- **Sociální sítě**
    - Analýza chování uživatelů sociální sítě pro zlepšení algoritmu, který spojuje uživatele s ostatními s cílem zvýšit čas, který uživatel na sítí tráví.
- **Výrobní sféra**
    - Odhad životnosti výrobku na základě informací nasbíraných při jeho výrobě a znalosti těchto informací z předchozích výrobních procesů.

5.4 Často uplatněné metody při data miningu

Někdy je naším cílem provést data mining, abychom data set částečně upravili nebo připravili například pro statistickou analýzu. V této kapitole se zmíním o řadě metod z různých kroků data miningu.

Standardizované skóre (v data miningu nazýváno také jako Z-Score):

Stejně jako ve statistice nám standardizované skóre pomáhá nalézt hodnoty, které se vzdalují od průměru a může se jednat o okrajové hodnoty nebo datové anomálie. V případě, že naše data odpovídají normálnímu rozdělení, skóre jsme schopni jednoduše spočítat odečtením průměrné hodnoty od hodnoty, kterou se snažíme standardizovat a následně vydělit tento rozdíl směrodatnou odchylkou. Pokud je výsledné skóre mezi hodnotami -3 až 3 považujeme, že se nejedná o datovou anomálii. Je vhodné zmínit, že bychom neměli automaticky odstraňovat okrajové hodnoty, ale čím více se blížíme těmto hranicím, je potřeba se zamyslet, jestli jejich odstranění by vylepšilo naše výsledky. Protože standardizované skóre používá průměrnou hodnotu a směrodatnou odchylku, dvě vlastnosti, které jsou ovlivněny okrajovými hodnotami, je náchylné k ovlivnění, když v datasetu existuje okrajových hodnot mnoho. Z tohoto důvodu byla zavedena metoda mezikvartilového rozptylu, která je odolnější vůči těmto vlivům. Jak metoda funguje bylo vysvětleno v kapitole 4.3.

Desetinné škálování:

Tato metoda je formou normalizace, která byla dříve vysvětlena v kapitole 3.5. Jakoukoliv hodnotu v datasetu normalizujeme tím, že ji vydělíme $10^{\wedge d}$, kde d je maximální počet číslic pro tuto vlastnost.

Transformace pro normalizaci:

Některé statistické metody požadují nebo alespoň očekávají, že sledované hodnoty odpovídají normálnímu rozdělení. Takovéto symetrické rozdělení nazýváme zvonová křivka (bell curve). Bohužel velká část dat z reálného světa neodpovídá tomuto rozdělení, naopak je spíše zešikmená z pravé strany (right-skewed). S levostrannou šikmostí se lze setkat, ale zdaleka ne tak často. Prováděné transformace jsou druhá odmocnina a invertovaná druhá odmocnina. Ve většině případů se nám nepodaří data transformovat, aby perfektně odpovídala normálnímu rozdělení, ale i částečné přiblížení k rozdělení může zásadně zlepšit výsledky. Před použitím výsledků je velice důležité provést de-transformaci.

Speciální proměnné (flag variables):

Při analýze se můžeme setkat s problematikou použití kategorických hodnot pro funkce jako je například regrese, protože tato funkce očekává číselné hodnoty. V případě, že kategorická hodnota nabývá pouze dvou stavů, jednoduše převedeme jeden stav na hodnotu 0 a druhý stav na hodnotu 1. Komplikace nastane, když počet stavů je roven nebo více jak 3. Zde zavádíme speciální proměnné nebo podproměnné pro naší kategorickou hodnotu. Počet těchto proměnných se dá určit jako počet stavů, kterých originální proměnná nabývá, mínus jedna. K ukázce si lze představit dataset, který obsahuje vlastnost *stav* a ta nabývá hodnot *pozastavený, spuštěný, vypnutý*. Ustanovíme, že hodnoty 1 může nabýt pouze jedna z těchto proměnných, ostatní budou 0. Z toho plyne, proč nám stačí mít o jednu méně proměnnou, než je stavů. Hodnota 1 v jedné ze speciálních proměnných reprezentujících stav (*spuštěný* nebo *vypnutý*) jednoznačně uvádí, ve kterém stavu je, protože platí pravidlo pouze jedné proměnné s hodnotou 1. Pokud obě proměnné jsou 0, znamená to, že stav musí být *pozastavený*, protože není *spuštěný* ani *vypnutý*. Někoho by mohlo napadnout proč nepřevést stavy přímo do číselných hodnot, každý stav reprezentován určitým číslem. Pro předchozí příklad tedy *pozastavený = 1, spuštěný = 2, vypnutý = 3*. Tímto bychom pro algoritmus zpracovávající data zavedli předpoklady, které nechceme. Mezi ně patří, stav *vypnutý* s hodnotou 3 je roven součtu stavů *pozastavený* a *spuštěný*. Stavy jsou seřazeny v pořadí *pozastavený, spuštěný, vypnutý*. A mnoho dalších chybných předpokladů, které ovlivní výsledek.

Naopak od předchozí metody existují algoritmy, které upřednostňují kategorické hodnoty oproti číselným. Potřebujeme-li převést hodnoty, aplikujeme rozdělení do skupin, také nazývaných jako pásma. Uplatnění této metody bývá v případech, kde máme rozsáhlý rozsah hodnot, které se snažíme „zobecnit“, například převést cenu produktů pohybující se mezi 1 až 100 000 na kategorie *levný, adekvátní, drahý, velice drahý*. K tomu, jak rozdělit hodnoty můžeme přistoupit třemi způsoby. [39] [40]

- Stejná šířka pásma – jednoduché rozdělení číselných hodnot do *k* kategorií, kde každá kategorie má stejnou šířku. Pro ilustraci si představme hodnoty 0 až 1000, které rozdělíme do 4 kategorií, jednotlivé kategorie můžeme pak definovat jako (*h* označuje libovolnou hodnotu v datasetu):
    - Kategorie 1: $0 \leq h < 250$
    - Kategorie 2: $250 \leq h < 500$
    - Kategorie 3: $500 \leq h < 750$
    - Kategorie 4: $750 \leq h \leq 1000$

  Z definice lze vidět, že tento přístup je silně ovlivněn okrajovými hodnotami nebo datovými anomáliemi, které nám posunou pásmo.
- Stejná frekvence pásma – hodnoty rozdělíme do *k* kategorií, každé pásmo obsahuje stejný počet záznamů (+- jeden záznam v případě lichého počtu záznamů). Velikost pásma lze spočítat jako celkový počet záznamů vydělen počtem kategorií. Rozdělení není ovlivněno tím, které hodnoty se v datasetu nachází. Problémem je ale předpoklad, že každá kategorie má v datech stejné zastoupení, tomu tak v reálných datech nebývá.
- Pásmo tvořené shlukováním – použitím libovolného shlukovacího algoritmu (nejčastěji „k-means“) dostaneme clustery, které nám tvoří spolehlivé pásmo.

6 POŽADAVKY A DIAGRAMY POUŽITÍ

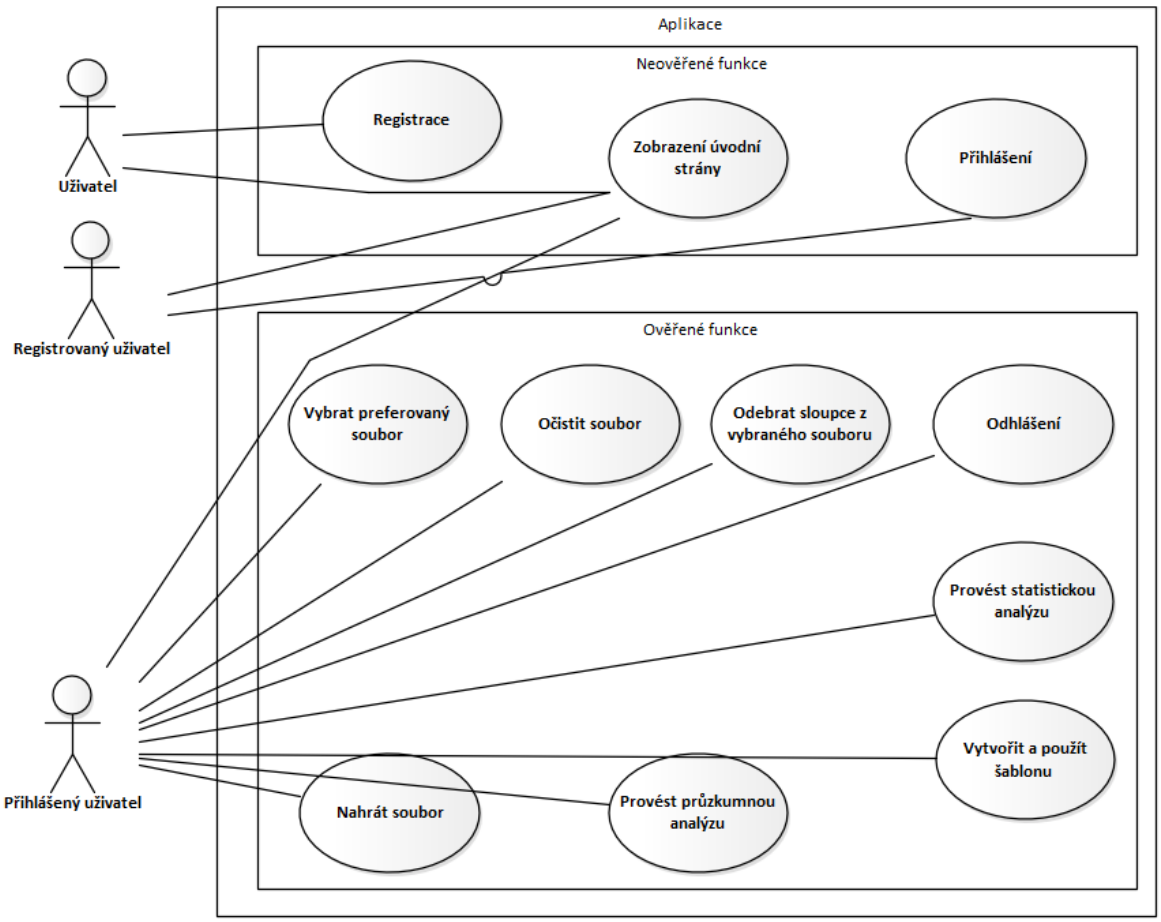
6.1 Funkční požadavky

- Aplikace umožní registraci uživatele
- Aplikace umožní přihlášení uživatele
- Aplikace umožní odhlášení uživatele
- Aplikace umožní import dat ve formátu CSV nebo JSON
- Aplikace umožní provést očištění dat
- Aplikace umožní vytvořit kopii dat, se kterou lze dále pracovat
- Aplikace umožní stáhnout uživatelské soubory, které jsou v aplikaci uloženy
- Aplikace umožní odebírat sloupce z nahraných souborů
- Aplikace umožní provést průzkumnou analýzu dat
- Aplikace umožní provést statistickou analýzu
- Aplikace umožní provést shlukovou analýzu
- Aplikace umožní vytvořit a aplikovat analýzu pomocí šablony

6.2 Nefunkční požadavky

- Aplikace bude schopna efektivně zpracovávat vědecká data a poskytovat rychlé odezvy na uživatelské požadavky.
- Aplikace bude zajišťovat ochranu dat a uživatelských účtů před neoprávněným přístupem.
- Aplikace má rozhraní, které je intuitivní, a snadno použitelné.
- Uživatelská příručka aplikace, popisuje funkcionalitu aplikace, procesy importu a analýzy dat.
- Aplikace by měla být použitelná na různých zařízeních, včetně mobilních telefonů.

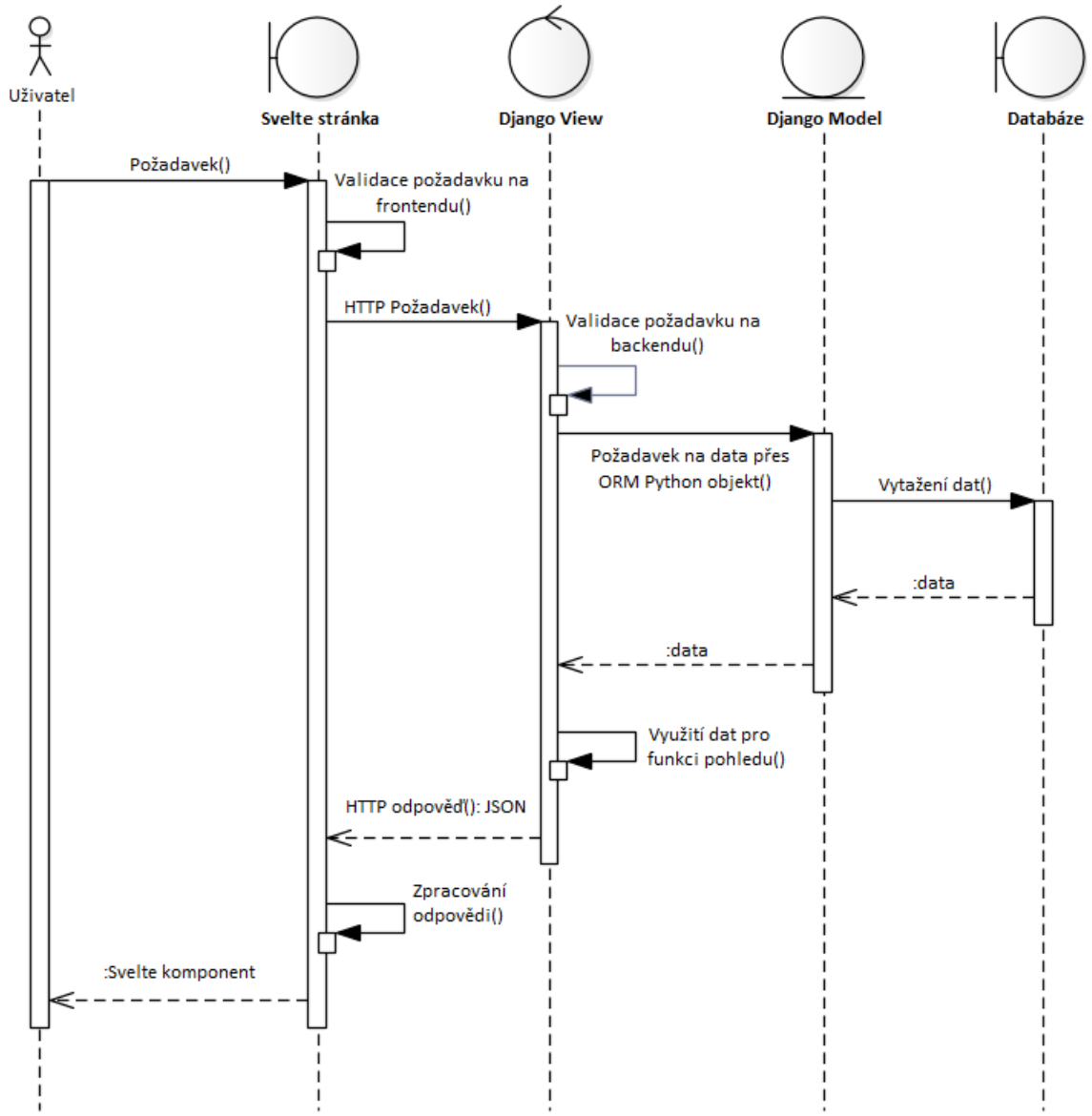
6.3 Diagram případů užití



Obrázek 8: Diagram případů užití pro uživatele [zdroj: vlastní]

7 DIAGRAM INTERAKCE APLIKACE

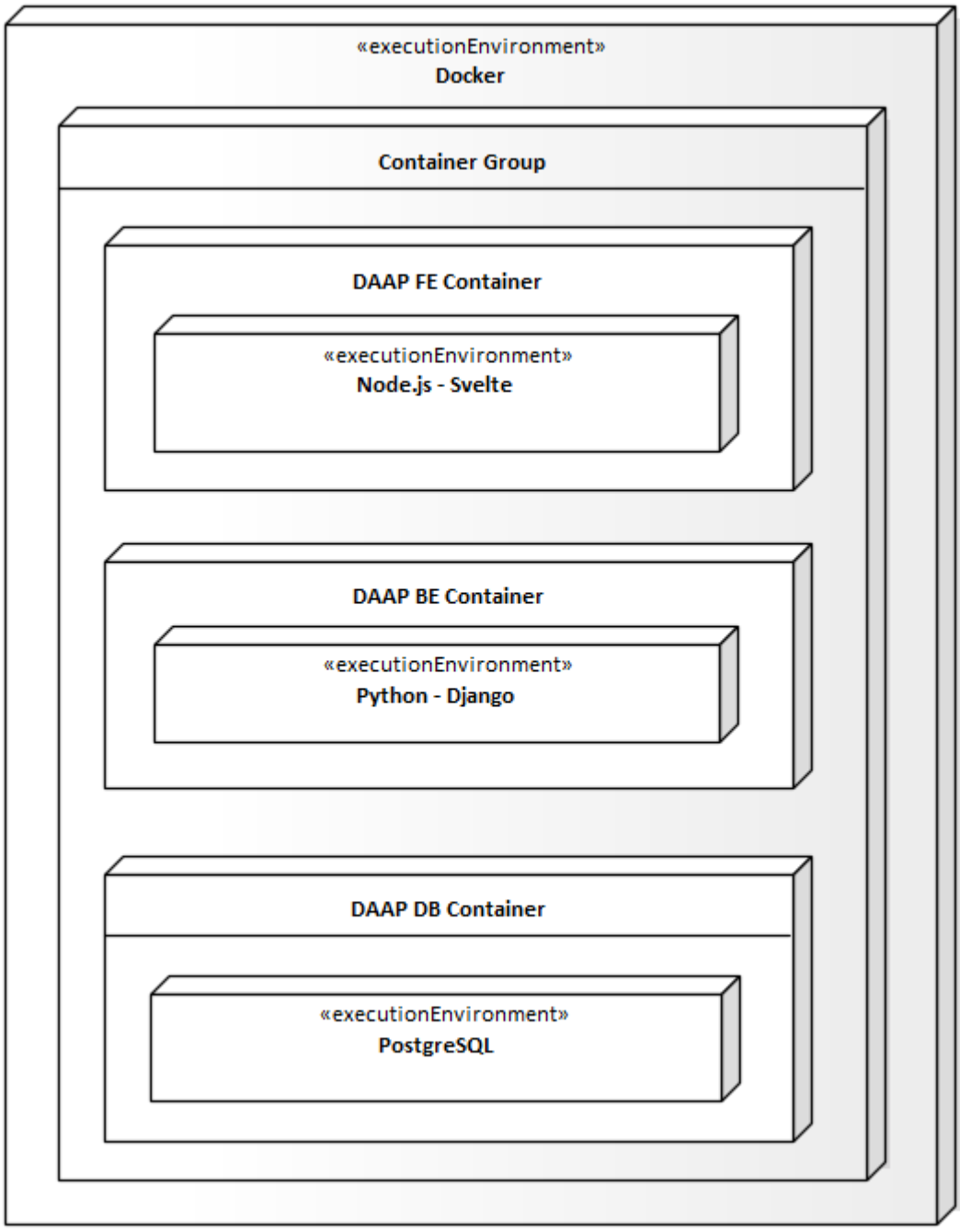
Diagram popisující proces komunikace mezi uživatelem, webovým rozhraním a aplikací.



Obrázek 9: Diagram komunikace jednotlivých částí aplikace [zdroj: vlastní]

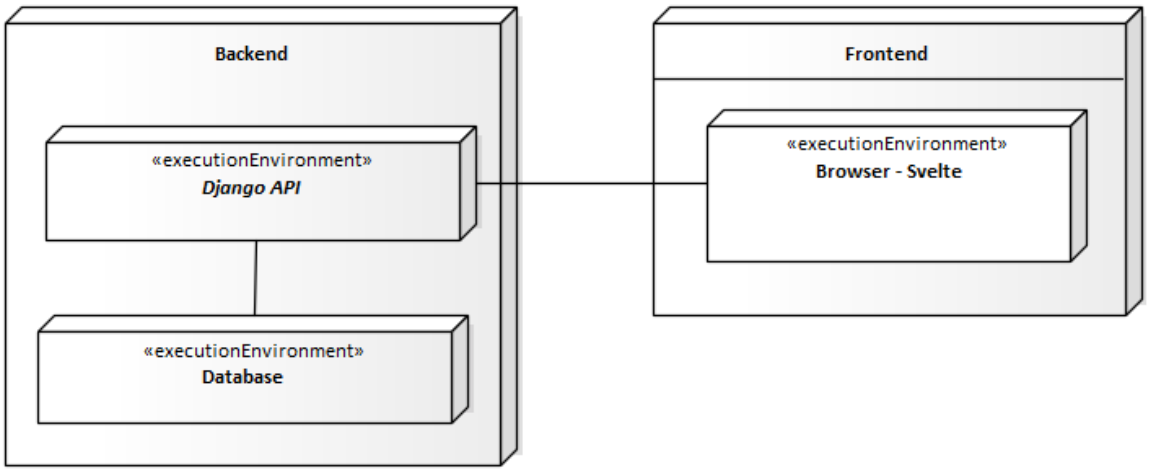
8 DIAGRAMY ARCHITEKTURY

8.1 Diagram architektury v prostředí Docker



Obrázek 10: Architektura projektu [zdroj: vlastní]

8.2 Diagram komunikace mezi komponenty



Obrázek 11: Komunikace mezi komponenty [zdroj: vlastní]

9 NASAZENÍ APLIKACE

Pro nasazení aplikace jsou nabízeny dvě možnosti. První je stáhnutí projektu[10] a spuštění ve volně zvoleném editoru, který podporuje Python virtuální prostředí a Node.js. Projekt již zahrnuje nainstalované balíčky a knihovny. Další možností je spuštění celého projektu (backend, frontend, databáze) jako multi-kontejner. Následující kódový blok popisuje obsah docker-compose souboru, kterým lze spustit projekt v prostředí Docker.

Kód 1: docker-compose soubor

```
name: daap-deployment

services:
  database-deployment:
    image: postgres
    ports:
      - 5432:5432
    environment:
      POSTGRES_DB: diplomka-db
      POSTGRES_USER: diplomka-db
      POSTGRES_PASSWORD: diplomka-db

  backend-deployment:
      image: ghcr.io/st61014/django-backend:latest
      command: bash -c "python djangoServer/manage.py migrate && python
djangoServer/manage.py runserver 0.0.0.0:5444"
      ports:
        - 5444:5444
      environment:
          PYTHONUNBUFFERED: 1
          DJANGO_SETTINGS_MODULE: djangoServer.settings
      depends_on:
        - database-deployment
  frontend-deployment:
    image: ghcr.io/st61014/svelte-frontend:latest
    ports:
      - 5173:80
    depends_on:
      - backend-deployment
```

[10] https://github.com/st61014/pyprocapp-backend-DP | https://github.com/st61014/pyprocapp-frontend-DP

10 NÁVRH A ARCHITEKTURA APLIKACE

10.1 Volba frameworků a knihoven

Proces výběru programovacího jazyka pro backend a následná volba jazyka Python byl objasněn v předchozí kapitole 2.5. Python pro zpracování dat. Programovat projekt v čistém Pythonu by znamenalo naprogramovat řadu podpůrných funkcí, než by bylo vůbec možné začít programovat funkce pro zpracování vědeckých dat. Z počátku bylo tedy jasné, že bude potřeba vybrat framework, který bude splňovat požadavky projektu. Při výběru byly zváženy například následující požadavky: RESTful (representational state transfer) API neboli dodržovat REST standard, správa uživatelů, jednoduchá implementace komunikace s libovolnou databází, dobré zabezpečení. Zároveň bylo hleděno na „nafouklost“ aplikace neboli, aby potřebná režie na běh aplikace nebyla několika řádově vyšší, než je potřeba.

10.1.1 Framework pro backend

Django framework je cílený na rapidní vývoj webových aplikací, se zaměřením minimalizovat potřebu opakovat kód (také označováno jako zvýšení opakované použitelnosti) a využití MVC architektury (Model-View-Controller, česky Model-Pohled-Ovladač).

Klíčovou vlastností je již zmíněná znovupoužitelnost komponent, to zde zaručují pohledy. Každý pohled si lze představit jako například třídu v jazyce Java, nebo případně jako ovladač (controller) ve frameworku Spring. Typicky se setkáme s navázáním URL na daný pohled, veškeré HTTP žádosti na tuto adresu jsou poté zpracovány v pohledu, parametry, které ovlivňují, jak pohled pracuje jsou součástí žádosti.

Django nabízí také možnost automatického převodu Python objektů na databázové objekty pomocí ORM (object-relational mapper, česky mapování objektů na relace v databázi), díky tomu lze pracovat s databází přes Python objekty a ORM se postará o správné navázání a překlad objektů pro zvolenou databázi. Django je silně zaměřený také na bezpečnost, předchází tak typickým bezpečnostním chybám jako je XSS (cross-site scripting), CSRF (cross-site request forgery), SQL injection a mnoho dalších. Snižuje tak objem práce potřebné na zabezpečení aplikace vývojářem. Za zmínku stojí i možnost generovat HTML obsah přímo přes nabízené šablony od Django, tato funkce není v projektu využita, protože tvorbu tohoto obsahu plní jiný frontend framework.

Framework také obsahuje implicitní správu uživatelů a správu jejich aktuálních přihlášení, která je session-based, ale s možností převodu na jiný typ jako například JWT (JSON Web Token), této možnosti je v projektu plně využito.

Jako další bonus je před-nastavení frameworku pro vývoj webových aplikací, Django samozřejmě nabízí velkou škálu konfigurace, ale výchozí hodnoty není často zapotřebí měnit.

Na závěr je vhodné zmínit škálovatelnost, framework disponuje funkcemi jako je caching, replikace databáze, vyvážení zátěže a další.

Jak plyne ze zaměření na vývoj webových aplikací, Django dodržuje REST standard, nabízí požadované funkce jako je správa uživatelů, komunikace s libovolnou databází a implicitní zabezpečení. Je tedy jasné, že Django bylo ideální volbou pro tento projekt. [42]

10.1.2 Framework pro frontend

Svelte je další z mnoha komponentových frameworků, ale s velkým rozdílem v přístupu k vykreslování webové stránky. Ve známých frameworcích jako je React, Angular nebo Vue je většina věcí zpracovávána při běhu, naopak Svelte přesouvá vše, co lze, do kompilace. Výsledkem je odlehčená a responzivní webová stránka. Dalším rozdílem je reaktivní model oproti typickému virtuálnímu DOMu (Document Object Model), což dále zvyšuje rychlost vykreslení webové stránky a obecně zlepšuje výkon.

Kódový syntax frameworku Svelte se snaží být co nejvíce přívětivý programátorovi, mezi významné vlastnosti patří skoro celková eliminace potřeby dodatečného kódu pro zajištění reaktivity proměnných, nabídka logických bloků, které zastupují JavaScript uvnitř HTML elementů. Klíčovým prvkem, který umožňuje zmíněné vlastnosti, je Svelte kompilátor, který převádí Svelte komponenty na JavaScript kód při sestavení. Tento kód poté manipuluje s DOM bez potřeby nadměrné režie, která je potřeba pro frameworky používající převod elementů za běhu nebo porovnání rozdílů DOMu. [43]

10.1.3 Knihovny pro backend

Pandas

Knihovna zaměřená na manipulaci s daty a jejich analýzu. Pro tento účel disponuje vlastními datovými strukturami jako je DataFrame a Series. DataFrame lze přirovnat k tabulce, ve které Pandas vede indexy pro interní manipulaci s daty. Series často obsahuje data jdoucí sekvenčně po sobě (například časové záznamy), a struktura je tak jednodimenzionální pole. V projektu

provádí Pandas převážnou většinu operací jako je očištění dat, agregace, filtrování a spojení do skupin. Výsledné DataFramy je možné používat přímo v knihovnách jako je například NumPy, Matplotlib, Bokeh, bez potřeby převádět na jiný datový typ. Za zmínku stojí nástroje této knihovny pro načítání souborů různých formátů přímo do DataFramu, které řeší časté chyby ve formátu jako je JSON. [44]

NumPy

Zkratka pro Numerical Python, jak vyplývá z jména jde o knihovnu, která je základem pro jakékoliv složitější matematické operace v Pythonu. Hlavním strukturou je zde ndarray, schopná pojmout a zpracovat multidimenzionální pole. V projektu jsou často využity transformační funkce (například normalizace dat), lineární algebra a matematické funkce (například veškerá popisná statistika pro Průzkumnou analýzu). Za zmínku stojí i rychlost, se kterou NumPy provádí operace, a to je díky implementaci knihovny v jazyce C. Nabízí tak možnosti jazyka Python s rychlostí kompilovaného kódu C. [45]

Matplotlib

Knihovna pro vizualizaci dat. Nabízí úpravu podoby výsledného grafu podle potřeby, od titulku, legendy po anotace bodů. Matplotlib také disponuje možností uložení vytvořených grafů v typických formátech jako je PNG, PDF, SVG. [46]

Seaborn

Další vizualizační knihovna, tentokrát založena na dříve zmíněné Matplotlib. Pomáhá vytvořit informativní statistické grafy za pomoci rozšířené nabídky grafů. Knihovna Matplotlib je schopna vytvořit například korelační graf (scatter plot), histogram, tepelnou mapu (heatmap), ale nenabízí nástroje přímo pro tyto typy, je na uživateli, aby si tento typ grafu poskládal dohromady. Seaborn eliminuje tento problém tím, že nabízí tyto grafy přímo. Přináší podporu pro možnosti jako je změna barevné palety, vykreslení dat do statistického rozdělení, použití kategorických dat a další. Seaborn tak v projektu doplňuje nedostatky Matplotlib. [47]

Bokeh

Již třetí knihovna zaměřená na vizualizaci dat za pomoci grafů, ale tentokrát cílená pro vizualizaci na webu a interaktivitu grafů. Podobně jako Seaborn nabízí Bokeh řadu grafů vhodné pro datovou analýzu. Výsledné grafy lze vyexportovat v řadě formátů jako předchozí knihovny, ale nejčastější je HTML, protože je pak možné využít interaktivity. Mezi nabízené

prvky interaktivity patří přibližování, posun, popis při najetí na prvek, stažení grafu a další. HTML formát je také možné vložit přímo do webové stránky. Veškeré interaktivní grafy nacházející se v projektu jsou tvořeny za pomoci této knihovny. [48]

SciPy

Knihovna postavena nad knihovnou NumPy, pro datovou analýzu a vědecké výpočty. Podobně jako NumPy hlavním využitím je aplikování lineární algebry, transformačních funkcí. Existují i funkce pro zpracování signálů, interpolaci, statistickou analýzu a další, ale ty nebyly v projektu využity. Uplatnění knihovny v projektu je spíše podpůrné pro jiné knihovny provádějící datovou analýzu. [49]

Sklearn

Zkratka pro plné jméno Scikit-learn, je z velké části knihovna pro strojové učení, ale nabízí i nástroje pro data mining a datovou analýzu. Knihovna je postavena na předešlých knihovnách v této kapitole, jako je NumPy, SciPy a Matplotlib, je tak integrace funkcí do nástrojů těchto knihoven velice jednoduchá. Hlavní využití v projektu je v modulu pro shlukovou analýzu, kde je potřeba uplatnit transformace pro provedení K-means shlukování. [50]

Pyclustering

Jak vyplývá z názvu, jedná se o knihovnu zaměřenou na shlukové algoritmy a datovou analýzu. Nabízí známé shlukové algoritmy jako je K-means, DBSCAN, OPTICS a hierarchické shlukování. Každý algoritmus lze nastavit podle potřeby, v projektu je to potřeba při změně výpočtu vzdálenosti bodů od středu shluku. Knihovna je také optimalizována na velké množství dat, se kterým se je možné setkat při shlukové analýze. [51]

SimpleJWT

Umožňuje přidat JSON Web token jako formu ověření uživatele ve webových aplikacích. Nabízí endpointy a uživatelské rozhraní pro generování, validaci, správu, blacklist tokenů. Knihovna nabízí integraci přímo pro frameworky Django a Flask, to umožňuje nahradit existující správu relací (v případě frameworku v projektu, Django nahrazuje session-based správu) za správu pomocí JWT. SimpleJWT je tak jednoduchým způsobem, jak implementovat JWT do projektu, bez potřeby dodatečného programování. [52]

Repair json

Malá knihovna rozšiřující možnosti opravy JSON souboru v případě jeho nevalidity. Dříve zmíněný Pandas má v nástrojích načítání JSON souboru zakomponované funkce pro opravu, ale ne vždy je schopný JSON soubor opravit. V takovém případě je využita tato knihovna, která bylo založena s cílem opravovat chyby v JSON formátech tvořených LLM (Large Language Model). Mezi časté chyby, které je schopna opravit patří chybějící uvozovky, ukončené složené závorky, slova vložená mimo složené závorky. V aplikaci je knihovny využita pro pokus opravit JSON soubory, když se nezdařilo jeho načtení za pomoci knihovny Pandas. [53]

10.1.4 Knihovny pro frontend

Flowbite for Svelte

Knihovna pro UI komponenty zaměřené přímo pro Svelte. Vychází z knihovny Flowbite. Rozdílem mezi Flowbite a Flowbite-Svelte je využití interaktivních elementů Svelte. Výchozí Flowbite je knihovna komponent tvořených HTML elementy s CSS třídami z knihovny Tailwind. Flowbite-Svelte rozšiřuje tyto komponenty o funkce nabízené frameworkem Svelte. [54]

Svelte routing

Klíčová knihovna pro vytvoření SPA (Single page application) aplikace. Za pomoci předem definovaných cest (route) lze místo přesměrování a znovu načtení stránky pouze nahradit komponenty jinými. Svelte routing podporuje i vnořené cesty a parametrizované cesty a funkci `navigate`, která umožňuje změnit obsah stránky bez potřeby odkazu nebo odeslání formuláře. [55]

Axios

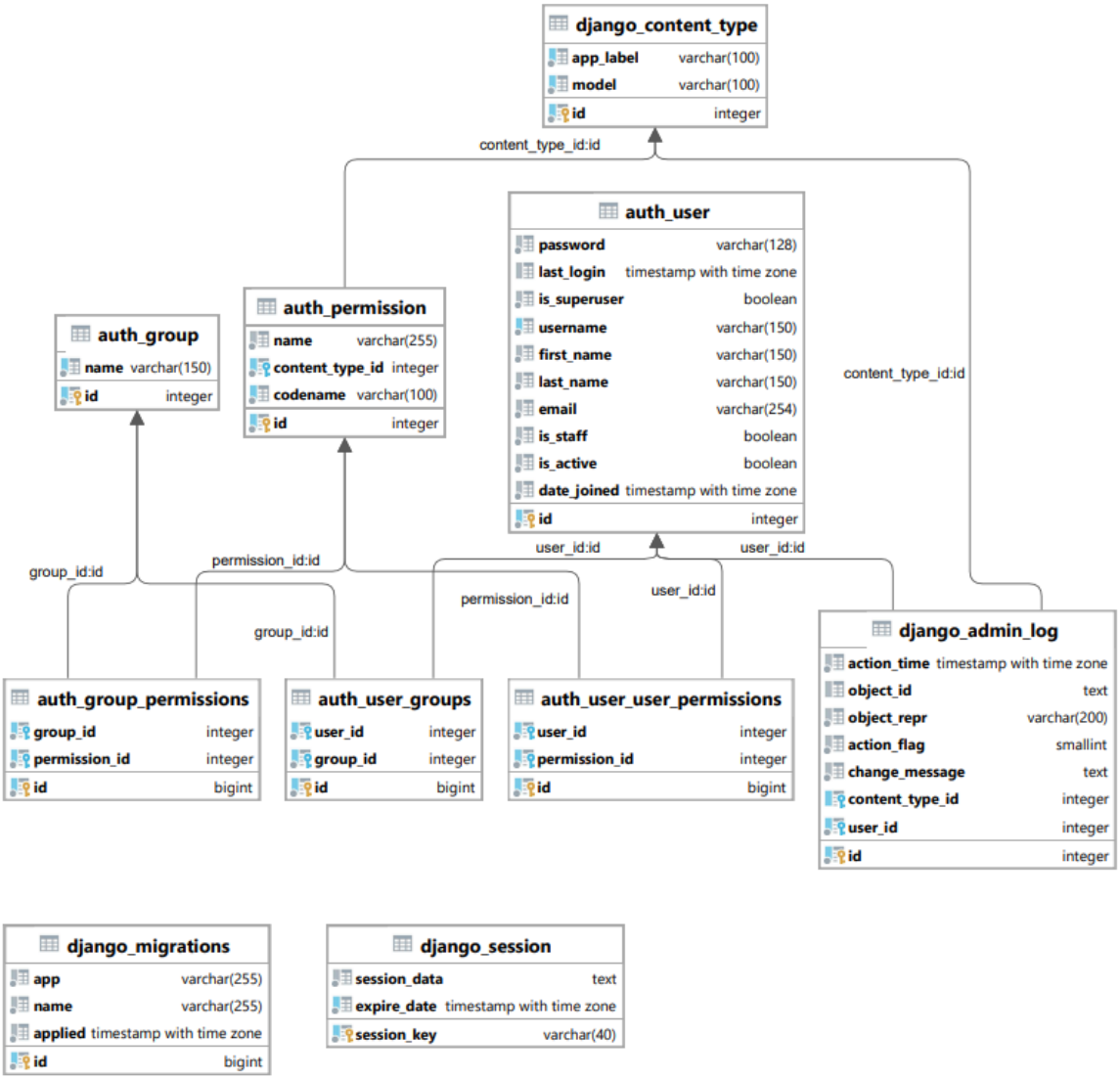
Populární JavaScript knihovna pro tvorbu HTTP žádostí z webových aplikací. Řeší správu odesílání žádostí a zároveň zpracování odpovědí. Mezi funkce nabízené knihovnou Axios patří například odesílání takzvaných promise-based žádostí a rušení aktivních žádostí. Knihovna také řeší časté problémy, se kterými se lze setkat při ruční tvorbě HTTP žádostí. [56]

10.2 Model aplikace

10.2.1 Základní databázový model pro Django

Vizualizace modelu databáze ihned po prvotním spuštění Django projektu a provedení migrací. [42] [57]

- **django_content_type**: Tato tabulka uchovává informace o všech modelových třídách v aplikaci. Každý model je reprezentován instancí ContentType.
- **auth_user**: Informace o uživatelích aplikace, od osobních údajů po informaci, jestli se jedná o správce nebo jestli je účet aktivní.
- **auth_group**: Spojová tabulka přiřazující skupinu uživatelů, ke skupině oprávnění.
- **auth_permission**: Seznam dostupných oprávnění k přiřazení uživatelům aplikace.
- **auth_group_permission**: Spojová tabulka pro propojení skupin uživatelů s jejich příslušnými oprávněními.
- **auth_user_groups**: Tato tabulka slouží k propojení uživatelů s jejich skupinami.
- **auth_user_user_permissions**: Tabulka reprezentující vazbu uživatelů na jednotlivá oprávnění.
- **django_admin_log**: Záznamy o akcích prováděných administrátory v administrátorském rozhraní.
- **django_migrations**: Informace o provedených migracích modelů a databáze.
- **django_session**: Tabulka pro uchování stavových informací o relaci uživatele, nejčastěji o stavu platnosti spojení. Po rozšíření aplikace o JWT je vyřazena z použití.



Obrázek 12: Výchozí databázový model aplikace [zdroj: vlastní]

10.2.2 JWT rozšíření

Vizualizace modelu databáze po přidání knihovny SimpleJWT a provedení migrací. [42] [57]

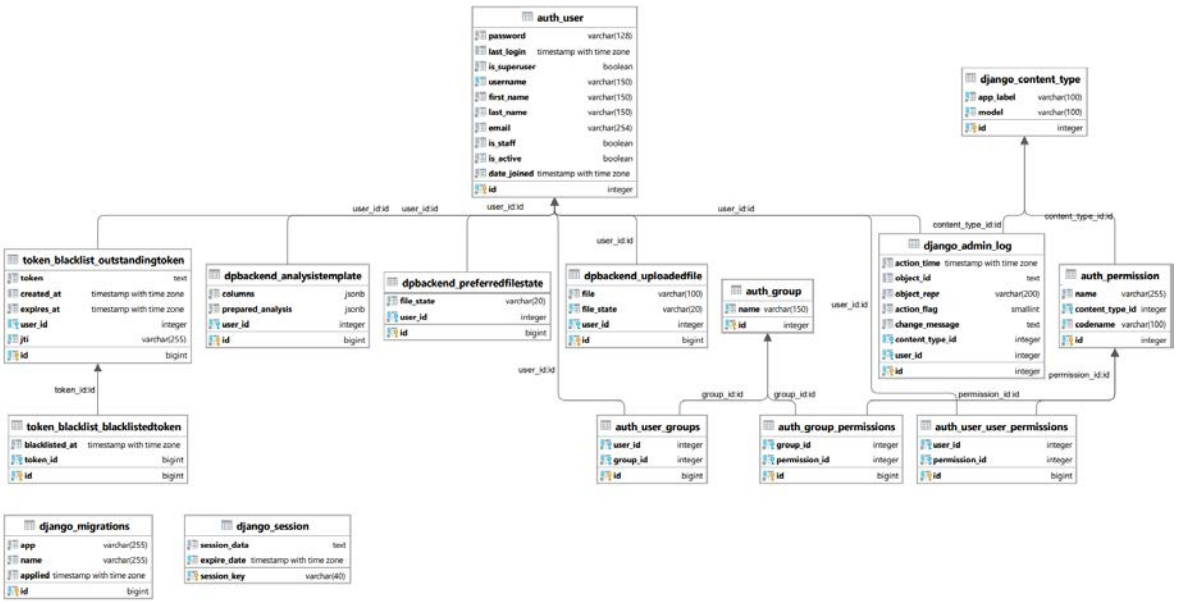
- **token_blacklist_outstandingtoken**: Tato tabulka uchovává informace o všech platných tokenech momentálně existujících v aplikaci. Uchovává nejen samotný JWT text, ale i informace, které jsou obsaženy v tokenu jako je datum tvorby a datum expirace, napojené na Django auth modul přes uživatelské ID.
- **token_blacklist_blacklistedtoken**: Tabulka pro evidenci tokenů, které byly zneplatněny, například když se uživatel odhlásí, je potřeba jeho tokeny zneplatnit, aby je někdo nemohl zneužít pro přihlášení v případě, že je získá.

Obrázek 13: Rozšířený databázový model aplikace o JWT [zdroj: vlastní]

10.2.3 Databázový model celé aplikace

V aplikaci existují vlastní modely AnalysisTemplate, PrefferedFileState, UploadedFile, ty jsou reprezentovány následujícími tabulkami. [42] [57]

- **dpbackend_analysistemplate**: Tato tabulka uchovává parametry uložené šablony uživatele a sloupce, které tato šablona vyžaduje.
- **dpbackend_prefferedfilestate**: Evidence stavu souboru, které uživatel preferuje pro využití při analýze.
- **dpbackend_uploadedfile**: Tabulka nesoucí informace o tom, které soubory má uživatel nahrané, pod jakým jménem jsou uloženy a o jaký stav souboru se jedná.



Obrázek 14: Finální databázový model aplikace [zdroj: vlastní]

10.3 Budoucí rozšíření

10.3.1 Vylepšení rozložení webu pro mobilní zařízení

Webové rozhraní je dostupné a použitelné na mobilních zařízeních nebo zařízeních s malým rozlišením obrazovky. Vhodným vylepšením v budoucnu by bylo přepracování způsobu, jakým se webové rozhraní zobrazuje na těchto zařízeních. Například menu pro výběr operací šablony překrývá seznam přidaných sloupců. Pracovat s aplikací na mobilním zařízením nebylo cílem projektu, ale nabízet takovou možnost by bylo ideální rozšíření, zejména v kombinaci s následujícím možným rozšířením.

10.3.2 Možnost nahrát dataset přes URL

Pro využití velké části funkcí aplikace je potřeba nahrát soubor, k tomu je zapotřebí mít soubor lokálně stažený. Problém nastane, když uživatel chce aplikaci využívat z jiného zařízení, ale nemá v ní data nahrána. Pokud jsou data veřejně dostupná, nebo je uživatel má nahrané v cloudovém úložišti, ale bez možnosti je osobně stáhnout z těchto zdrojů. Možným řešením tohoto problému by bylo umožnit nahrát do aplikace data přes URL odkaz na ně. Aplikace by poté data stáhla a uložila, aby byly následně dostupné pro zpracování. Tato funkce by i umožnila jednodušší práci s aplikací na mobilních zařízeních.

11 IMPLEMENTACE MODULŮ

11.1 Zpracování vstupních dat

Zpracování dat, také nazýváno očištění dat, je zahrnuto do očištění po nahrání souboru a jako dodatečná operace šablony, která se má aplikovat před tvorbou grafů. Všechny podmínkové proměnné jsou získány z HTTP žádosti za pomoci převodu stylem `proměnná = True if request.data.get("typ") == 'true' else False`. Při očištění je pracováno s kopií DataFrame z načtených dat. Odstranění odlehlých hodnot (v kódu *outliers*) prvně vyžaduje nalezení prvního a třetího kvartilu a vypočítání mezikvartilového rozsahu. Po výpočtu spodní a vrchní hranice lze vyhledat hodnoty, které jsou mimo tento rozsah přes negovanou funkci `between`. Po nalezení odlehlých hodnot jsou odstraněny z DataFramu. Odstranění duplicit je realizováno funkcí `drop_duplicates` s parametrem `inplace=True` pro jejich odstranění přímo v DataFramu bez potřeby tvořit kopii. Pokud je potřeba nahradit chybějící hodnoty je použita funkce *fillna* jejíž parametr definuje jaká hodnota má doplnit chybějící hodnoty. Při nahrazení nejčastější hodnotou je použit modus, pokud jich existuje více, je brán první. Když *fillna* vytvoří vyjímku typu `TypeError` nastala situace, kde prohledávaný sloupec nemá chybějící hodnoty, pak stačí výjimku zpracovat a pomocí klíčového slova `pass` pokračovat v průběhu zpracování. Pro nahrazení střední hodnotou je potřeba rozlišit číselné a textové sloupce, protože textové sloupce nemají střední hodnotu. Řešení je zachovat nahrazení modusem pro textové sloupce a použití střední hodnoty pouze pro číselné sloupce. Následuje oprava datových typů, tím se aplikace snaží nalézt chybné záznamy (například desetinná hodnota, kde místo tečky/čárky je omylem pomlčka), které způsobí že sloupec je rozpoznán jako špatný datový typ. Operace opravy datových typů je časově náročná, pokud dataset obsahuje velké množství záznamů. K výpočtům středních hodnot a nejčastějších hodnot jsou použity funkce knihovny Pandas, pro zpřístupnění výsledků na základě pozice (indexu) je funkce `iloc`. Poslední možností je plné zahození všech záznamů, které mají chybějící hodnotu v některém ze sloupců.

Kód 2: Očištění dat

```
if no_outliers:
    for col in processed_df.select_dtypes(include=['number']).columns:
        q1 = processed_df[col].quantile(0.25)
        q3 = processed_df[col].quantile(0.75)
        iqr = q3 - q1
        lower_bound = q1 - 1.5 * iqr
```

```
        upper_bound = q3 + 1.5 * iqr
        outliers = processed_df[~processed_df[col].between(lower_bound,
upper_bound)]
        processed_df = processed_df.drop(outliers.index)
if deduplicate:
    processed_df.drop_duplicates(inplace=True)
if replace:
    if replace_most_frequent:
        for column in processed_df.columns:
            try:
                processed_df[column] =
processed_df[column].fillna(processed_df[column].mode().iloc[0])
            except TypeError as e:
                pass
    if replace_average:
        for col in processed_df.columns:
            if pd.api.types.is_numeric_dtype(processed_df[col]):
                processed_df[col] = processed_df[col]
                                    .fillna(value=processed_df[col].mean())
            if pd.api.types.is_string_dtype(processed_df[col])
             | pd.api.types.is_object_dtype(processed_df[col]):
                try:
                    processed_df[col] = processed_df[col]
                   .fillna(processed_df[col].mode().iloc[0])
                except TypeError:
                    pass
if fix_datatypes:
    for col in processed_df.columns:
                type_counts = {}
                for value in processed_df[col]:
                    try:
                        value = int(value)
                    except ValueError:
                        try:
                            value = float(value)
                        except ValueError:
                            pass
                    data_type = type(value)
                    type_counts[data_type] = type_counts.get(data_type, 0) + 1

                majority_data_type = max(type_counts, key=type_counts.get)

                for index, value in processed_df[col].items():
                    try:
                        value = int(value)
                    except ValueError:
                        try:
                            value = float(value)
                        except ValueError:
                            pass
                    if not isinstance(value, majority_data_type):
                        try:
                            if type(value) is list:
                                value = 0
                            processed_df.at[index, col] =
```

```
                            majority_data_type(re.sub(r'[^0-9.]', '', value))
                    except ValueError:
                        pass
if drop_missing:
    processed_df.dropna(inplace=True)
```

11.2 Implementace individuálních modulů

Kapitola zaměřená na objasnění komplexnějších částí kódu aplikace.

11.2.1 Kód ve více nebo všech modulech

Některé sekce kódu se v projektu opakují (například zpracování HTTP žádostí) a budou proto vysvětleny zde, pro snížení opakování vysvětlení kódu.

Kód 3: Získání uživatelem preferovaného souboru

```
if use_preferred:
        preferred_file_state =
PreferredFileState.objects.filter(user=user).first()
        existing_file = UploadedFile.objects.filter(user=user,
file_state=preferred_file_state.file_state).first()
        if not existing_file:
            existing_file = UploadedFile.objects.filter(user=user,
file_state='raw').first()
            if not existing_file:
                return JsonResponse({'error': 'error.noValidFile'},
status=status.HTTP_400_BAD_REQUEST)
    else:
        existing_file = UploadedFile.objects.filter(user=user,
file_state='raw').first()
        if not existing_file:
            return JsonResponse({'error': 'error.noValidFile'},
status=status.HTTP_400_BAD_REQUEST)
```

Proměnná `user_preferred` je typu `boolean`, nejčastěji předávána jako parametr funkce pro generování grafu, nebo obsah těla HTTP žádosti.

Proměnná `user` vždy reprezentuje objekt uživatele z pohledu Django frameworku, je součástí pohledů, které vyžadují ověření uživatele před provedením funkce.

Pro přistoupení k databázovým objektům modelu `PreferredFileState` využíváme následující zřetězený dotaz `objects.filter(user=user).first()`, který vrátí první záznam v databázi pro daného uživatele `PreferredFileState` model vždy udržuje pouze jeden záznam pro každého uživatele a každý uživatel má jako výchozí hodnotu typ `"raw"`). V dalším kroku se

přistupuje k databázovým objektům modelu `UploadedFile` za pomocí stejného zřetězení dotazů jako pro model `PreferredFileState`, ale rozšířeného o parametr `file_state=preferred_file_state.file_state`, který získává objekt reprezentující soubor zvoleného stavu, uložený na serveru. V případě, že místo objektu se vrátí `None` je proveden pokus získat soubor stavu `"raw"`. Není-li informace o takovém souboru uložena v databázi, neexistuje na serveru a na HTTP žádost je odpovězeno stavem 400 BAD_REQUEST.

Když nebylo požadováno vyhledat user_preferred soubor, je automaticky hledán soubor stavu `"raw"` se stejnou odpovědí v případě, že neexistuje.

Kód 4: Oprava kódování a JSON souboru

```
try:
    with codecs.open(existing_file.file.path, 'r', encoding='utf-8',
errors='replace') as file:
                    incorrect_json = file.read()
                    file.close()

    df = pd.json_normalize(json_repair.repair_json(incorrect_json,
return_objects=True))
except:
    return JsonResponse({'error': 'error.unreadableJSON'},
status=status.HTTP_400_BAD_REQUEST)
```

JSON soubory mohou být neočekávaného kódování (bohužel ne vždy jsou UTF-8), pro tuto situaci nabízí Python modul `codecs`, který při otevření souboru převede soubor do požadovaného kódování s nastavením, co dělat s případnými chybami. V projektu je vždy soubor převeden na UTF-8 a nalezené chyby jsou nahrazeny (`encoding='utf-8', errors='replace'`), někdy tento přístup může vyústit v nesprávné znaky uvnitř dat, ale naopak zabrání kritické chybě aplikace z důvodu neznámého znaku. Po opravě kódování je ještě zapotřebí zkontrolovat případný chybný formát JSON souboru, který by Pandas nebyl schopen přečíst. Knihovna `json_repain` disponuje funkcí `repair_json`, která jako parametr bere JSON string a případně `return_objects` který specifikuje, jestli má funkce vrátit přímo JSON objekt nebo je serializovat na JSON string. Pokud i po opravách způsobí načtení JSON objektu přes Pandas `json_normalize` vyjímku, odpoví server na HTTP žádost stavem 400 BAD_REQUEST.

Kód 5: Tvorba interaktivního grafu

```
graph_name = f'{hash_of_params}_{plot_name}_plot.html'
        output_file(f'media/{user.username}/{folder}/{graph_name}')

source = ColumnDataSource(df) #zdroj dat nejčastěji přímo z DataFrame
p = figure(title='Titulek grafu', x_axis_label=x_column, y_axis_label=y_column)
p.scatter(x_column, y_column, source=source, size=3, alpha=1)
#případně p.line, p.vbar
p.resizable = True
p.sizing_mode = "scale_both"
hover = HoverTool(tooltips=[(f"{x_column}",f"@{x_column}"),
(f"{y_column}", f"@{y_column}")])
p.add_tools(hover)
p.grid.grid_line_alpha = 0.3
p.legend.location = "top_left"
save(p)
```

Podoba kódu pro generování interaktivní grafů se liší od modulu k modulu, ale obecný formát zůstává stejný. Pro unikátnost souboru je jeho jméno tvořeno kombinací otisku ze všech parametrů, které graf tvoří, typ grafu (regrese, časová osa apod.) a zakončeno „plot.html“. Aplikace ukládá všechny vygenerované grafy pod /media/uživatelské jméno/typ grafu/název grafu, příkladem cesty pro uložení grafu regresní analýzy by byl `/media/User1/regression/"hash"_regression_plot.html`.

Parametry jednotlivých typů grafů v knihovně Bokeh vyžadují definovat, který sloupec ze zdroje vybrat jako pro osu X a osu Y, případně volitelné parametry ovlivňující vzhled grafu (`size` pro velikost bodů, `alpha` pro průhlednost bodů). Bez upřesnění parametru `tools` Bokeh přidá všechny základní nástroje ke grafu (tj. posun, přiblížení, přiblížení kolečkem myši, stažení, obnovení pohledu, odkaz na dokumentaci, zobrazení popisu při najetí myši). Možnost `resizable` umožní grafu měnit velikost při změně velikosti okna, `sizing_mode` ovlivňuje, jak bude graf roztažen do velikosti okna, `"scale_both"` zachová poměr stran v obou směrech. Pro přidání popisu při najetí myši, je zapotřebí přiřadit hodnoty k bodům, nejjednodušší způsob je vložit odkaz na sloupec ve zdroji dat, ze kterého byl graf tvořen. Zavoláním funkce `save()` se graf uloží do souboru, který byl upřesněn ve funkci `output_file()`.

11.2.2 Přihlášení, registrace, odhlášení

Správa přihlášení uživatelů je řešena knihovnou SimpleJWT, co knihovna dělá bylo vysvětleno v kapitole 10.1.3. Knihovny pro backend.

Při registraci se odešle HTTP request obsahující uživatelské údaje, v pohledu RegisterView se data ověří přes model User, jestli už takové údaje existují, pomocí filtrů

Kód 6: Ověření existence uživatelského jména a emailu

```
User.objects.filter(username=username).exists()
User.objects.filter(email=email).exists()
```

Pokud uživatelské jméno i e-mail jsou unikátní, vytvoří se uživatel pomocí funkce modelu User `User.objects.create_user(username=username,email=email,password=password)` Existuje-li již uživatelské jméno nebo e-mail, případně pokud hesla nějak prošla validací na frontendu, ale zde nesedí, odpoví server stavem 400 BAD_REQUEST. Odhlášení opět využívá nástroje knihovny SimpleJWT, kde zároveň s access_token se odešle refresh_token, ze kterého se získá instance objektu Token a odebere se ze seznamu validních tokenů.

Kód 7: Vyřazení tokeny při odhlášení

```
token = RefreshToken(refresh_token)
token.blacklist()
```

11.2.3 Průzkumná analýza

V projektu pod názvem EDA (Exploratory Data Analysis) je možnost rychle nahlédnout do dat a získat přehled, jak data vypadají. Výsledkem je graf a popisná statistika pro každý zpracovatelný sloupec v souboru. Pokud nad sloupcem nejde analýzu provést, je vyřazen. Mezi nabízenou popisnou statistiku patří počet hodnot, počet unikátních hodnot, minimální a maximální hodnota, směrodatná odchylka, šikmost, medián, modus a střední hodnota. V úryvku kódu je vidět uložení informací pro číselný sloupec, s rozdíly pro textový sloupec uvedenými v komentářích.

Kód 8: Uložení výsledků analýzy

```
analysis_results[col] = {
    'graph': f'{graph_name}',
    'count': df[col].count(),
    'min': df[col].min(), #None
    'max': df[col].max(), #None
    'stdDev': ("{:.4f}".format(df[col].std())), #None
    'skewness': ("{:.4f}".format(df[col].skew())), #None
```

```
    'median': df[col].median(), #None
    'mean': ("{:.4f}".format(df[col].mean())), #None
    'unique': df[col].nunique(),
    'mode': df[col].mode().tolist()[0] if not df[col].empty else None
}
```

Pro zjištění těchto údajů nabízí všechny potřebné funkce knihovna Pandas. Valná většina popisné statistiky nelze určit pro textové pole bez transformací, které by ovlivnily vypovídající hodnotu. Z tohoto důvodu jsou nahrazeny hodnotou `None`, na webové stránce při zobrazení grafu jsou informace s hodnotou `None` skryty.

11.2.4 Regresní analýza

Modul sdílí generování grafu vysvětlené na začátku této kapitoly, s rozšířením pro různé druhy regresních křivek. Pro polynomickou regresní křivku jsou spočítány koeficienty pomocí funkce `polyfit` z knihovny NumPy, jejími parametry jsou X a Y sloupce obsahující data a hodnota udávající stupně. Povolené stupně jsou v rozmezí od 1 do 4, protože ve většině případů stupně vyšší jak 4 vedou k nežádoucím výsledkům. Funkce poly1d následné vytvoří hodnoty odpovídající požadované křivce a ta se vloží do grafu. Exponenciální regrese je spočítána jako lineární regrese, která je následně přepočítána na exponenciální pomocí NumPy funkce `exp` a vložena do grafu. Má-li se vykreslit regresní křivka a nebyla specifikována polynomická nebo exponenciální, je automaticky zvolena lineární, která opět používá funkci `linregress` a je poté vložena do grafu.

Kód 9: Výběr druhu regrese

```
if selected_type == "polynomial":
    coefficients = np.polyfit(df[x_column], df[y_column], poly_degrees)
    poly_function = np.poly1d(coefficients)
    p.line(x_values, poly_function(x_values), color='green',
           legend_label=f'Polynomial Regression (Degree {poly_degrees})')
elif selected_type == "exponential":
    slope_exp, intercept_exp, r_value_exp, p_value_exp, std_err_exp =
stats.linregress(df[x_column],
np.log(df[y_column]))
    p.line(x_values, np.exp(slope_exp * x_values + intercept_exp),
color='purple',
           legend_label='Exponential Regression')
else:
    slope, intercept, r_value, p_value, std_err = stats.linregress(df[x_column],
df[y_column])
    p.line(x_values, slope * x_values + intercept, color='red',
legend_label='Linear Regression')
```

11.2.5 Časová řada

Pro časovou řadu, kód sdílí způsob výběru sloupců a vykreslení do grafu s předchozími moduly, ale je vhodné objasnit metodu převzorkování. V obou případech převzorkování (zvýšení a snížení) je prvně zkontrolován parametr `scale_rate`, který je obsahem HTTP žádosti, tím je určeno na kolik jednotek se má vzorkování změnit. Typ vzorkování je uložen v proměnné `scale` a může být například hodnot typu *S* pro vteřiny, *M* pro minut, *H* pro hodiny a další, je podporováno až do jednotky roku. Když bylo určeno (hodnota nebyla 0) na kolik jednotek se má převzorkovat obsahem proměnné *rule* je složení počtu a typu (například 3H, 60M, 20D a další) Pro zvýšení vzorkování je jako první provedena funkce `infer_objects`, která se pokouší převést DataFrame sloupce typu object na přesnější typ. Toto je požadavek pro použití konečné funkce `interpolate`. Funkce `resample` aplikuje vytvořené pravidlo pro převzorkování. Zavoláním `interpolate(method='druh interpolace')` vyplníme nově vzniklé body lineární interpolací hodnot z originálních bodů. Při snížení vzorkování se originální body spojí a získají střední hodnotu ze všech bodů, které se spojily v dané oblasti. Protože časová osa vyžaduje číselné hodnoty na ose Y, můžeme uplatnit funkci `mean` pouze na číselné typy v DataFramu pomocí parametru `numeric_only`.

Kód 10: Aplikace změny vzorkování

```
if upsample:
    if scale_rate == 0:
        rule = scale
    else:
        rule = (str(scale_rate) + scale)
    df =
df.infer_objects(copy=False).resample(rule).interpolate(method='linear')
    chart_title += " (upsampled)"

if downsample:
    if scale_rate == 0:
        rule = scale
    else:
        rule = (str(scale_rate) + scale)
    df = df.resample(rule).mean(numeric_only=True)
    chart_title += " (downsampled)"
```

11.2.6 Box plot

Takzvané krabicové grafy vyžadují řadu výpočtů pro jednotlivé komponenty, které tvoří výsledný graf. Pro zlepšení přehlednosti byl následující kód rozdělen do sekcí, každá postupně vysvětlena.

V první sekci získáme unikátní hodnoty ze sloupce pro osu X (dále označováno jako sloupec X), následně spojíme DataFrame do skupin podle hodnot v sloupci X a spočítáme kvantily pro hodnoty ve sloupci pro osu Y (dále označováno jako sloupec Y). Funkce `unstack` změní vnitřní strukturu, definujeme nové sloupce pro každý kvantil a sloupec samotný a spojíme je zpět dohromady. Přes kvantily spočítáme vrchní a spodní hranici.

Druhá sekce převádí případné číselné sloupce obsahující unikátní hodnoty na strukturu Range1d, protože výchozí struktura pro unique nad číselnými sloupci je `ndarray` z knihovny Numpy, tato struktura není platná pro tvoření grafu. Po inicializaci grafu je struktura Range1d převedena do série, aby šlo hodnoty použít pro mapování barev.

Ve třetí sekci se přidají hlavní „boxy“ grafu v rozmezí kvantilů.

Poslední sekce přidává takzvané „whiskers“ (překládáno jako „vousy“), to jsou vertikální osy vycházející z boxů nahoru až po horní hranici a dolů až spodní hranici. Když není nastaven parametr filter_outliers jsou do grafu vloženy i odlehlé hodnoty reprezentovány částečně průhledným kruhem.

Kód 11: Výpočet potřebných údajů pro krabicový graf

```
# Sekce 1
x_column_uniques = df[x_column].unique()
try:
    qs = df.groupby(by=x_column)[y_column].quantile([0.25, 0.5, 0.75])
except:
    return JsonResponse({"error": "error.nonNumericQuantiles"},
                        status=status.HTTP_400_BAD_REQUEST)
qs = qs.unstack().reset_index()
qs.columns = [x_column, "q1", "q2", "q3"]
df = pd.merge(df, qs, on=x_column, how="left")

iqr = df.q3 - df.q1
df["upper"] = df.q3 + 1.5 * iqr
df["lower"] = df.q1 - 1.5 * iqr

source = ColumnDataSource(df)

# Sekce 2
if pd.api.types.is_numeric_dtype(x_column_uniques):
```

```
    x_column_uniques = Range1d(start=x_column_uniques.min(),
end=x_column_uniques.max())
p = figure(x_range=x_column_uniques, title=f'{x_column}+{y_column}',
y_axis_label=y_column)
p.resizable = True
p.sizing_mode = "scale_both"
if isinstance(x_column_uniques, bokeh.models.ranges.Range1d):
    x_column_uniques = [str(i) for i in range(int(x_column_uniques.start),
int(x_column_uniques.end) + 1)]


# Sekce 3
cmap = factor_cmap(x_column, "TolRainbow7", x_column_uniques)
p.vbar(x_column, 0.7, "q2", "q3", source=source, color=cmap, line_color="black")
p.vbar(x_column, 0.7, "q1", "q2", source=source, color=cmap, line_color="black")


# Sekce 4
whisker = Whisker(base=x_column, upper="upper", lower="lower", source=source)
whisker.upper_head.size = whisker.lower_head.size = 20
p.add_layout(whisker)


if not filter_outliers:
    outliers = df[~df[y_column].between(df.lower, df.upper)]
    p.scatter(x_column, y_column, source=outliers, size=6, color="black",
alpha=0.3)
```

11.2.7 Shluková analýza

Stejně jako předchozí kapitola, shlukování obsahuje velkou řadu výpočtů, a tak pro zlepšení přehlednosti byl následující kód rozdělen do sekcí, každá postupně vysvětlena.

První sekce se zabývá normalizací dat a volbou funkce pro výpočet vzdálenosti bodů od středů. Nabízené typy vzdáleností jsou uloženy v proměnné `distance_measures`, protože K-means z knihovny Pyclustering nepodporují kosinovskou vzdálenost je tato možnost nahrazena typem `USER_DEFINED`, který umožňuje použít vlastní funkci. Shlukování K-means vyžaduje počáteční středy, ty jsou dosazeny pomocí `kmeans_plusplus_initializer`, který na základě dat a počtu shluků vytvoří počáteční středy. Typ vzdálenosti je zvolen na základě jména, které je obsaženo v HTTP žádosti, pokud je zvolen typ USER_DEFINED je funkce dynamicky dosazena pomocí Python parametru `func=`.

Druhá sekce vytváří instanci K-means shlukování pomocí normalizovaných dat, počátečních středů a zvolené vzdálenostní funkce. Nad instancí je spuštěn algoritmus a následně jsou získány reálné středy a shluky.

Třetí sekce získává takzvané štítky, které jsou použity pro přiřazení bodů do shluků ve grafu.

Kód 12: Aplikace algoritmu K-means pro shlukovou analýzu

```
# Sekce 1
distance_measures = {
'euclidean': type_metric.EUCLIDEAN,
'squared_euclidean': type_metric.EUCLIDEAN_SQUARE,
'cosine': type_metric.USER_DEFINED,
'manhattan': type_metric.MANHATTAN,
'chebyshev': type_metric.CHEBYSHEV,
'canberra': type_metric.CANBERRA,
'chi_square': type_metric.CHI_SQUARE}

initial_centers = kmeans_plusplus_initializer(main, kclusters).initialize()

if not distance_calculation:
    distance_calculation = "euclidean"

selected_function = (distance_calculation + "_distance")
if distance_measures[distance_calculation] == type_metric.USER_DEFINED:
    selected_metric = distance_metric(distance_measures[distance_calculation],
func=locals()[selected_function])
else:
    selected_metric = distance_metric(distance_measures[distance_calculation])
# Sekce 2
kmeans_instance = pyclustering.cluster.kmeans.kmeans(main, initial_centers,
metric=selected_metric)

kmeans_instance.process()
centers = kmeans_instance.get_centers()
clusters = kmeans_instance.get_clusters()

# Sekce 3
cluster_encoding = kmeans_instance.get_cluster_encoding()
encoder = cluster_encoder(cluster_encoding, clusters, scaled_data)
labels = encoder.set_encoding(0).get_clusters()
```

11.2.8 Šablona

Šablona kombinuje všechny předchozí moduly vysvětleny v této kapitole, do jednoho. Rozdílem je místo vracení jména vytvořeného souboru obsahující graf, je vracen JSON objekt se všemi grafy, které šablona vytvořila. Protože chyba generování grafu nemůže zastavit běh celé šablony, je místo výsledného jména grafu uložen chybový stav, který je na webové stránce zobrazen jako varovné upozornění, že se generování grafu nezdařilo.

11.3 Způsob uložení zpracovaných dat

Po zpracování dat je potřeba je opět uložit, aby byly přístupné v zpracované podobě bez potřeby provádět operace znovu. Soubory CSV jsou uloženy v naprosto stejné podobě jako před zpracováním, protože každý řádek si lze představit jako index pro celý tento záznam, to znamená, že první řádek obsahuje celý první záznam, druhý řádek celý druhý záznam a tak dále. Ve validním souboru je počet separátorů vždy stejný. Pro soubory typu JSON je uložení částečně rozdílné, protože původní soubor může mít následující strukturu:

Kód 13: Vzor možné JSON struktury

```
{
    "country": "AD",
    "name": "Sant Julià de Lòria",
    "lat": "42.46372",
    "lng": "1.49129"
},
...
```

Jako JSON objekt je to naprosto v pořádku, ale při načtení do struktury DataFrame vznikne problém s požadavkem indexace. Možností je přidat index k celým objektům, ale to způsobí nadbytečnou duplikaci názvu atribut jako je `country`, `name`, a podobně. Výchozí řešení pro tuto situaci nabízí Pandas změnou struktury dokumentu na následující strukturu:

Kód 14: Struktura vzorového JSON souboru po zpracování knihovnou Pandas

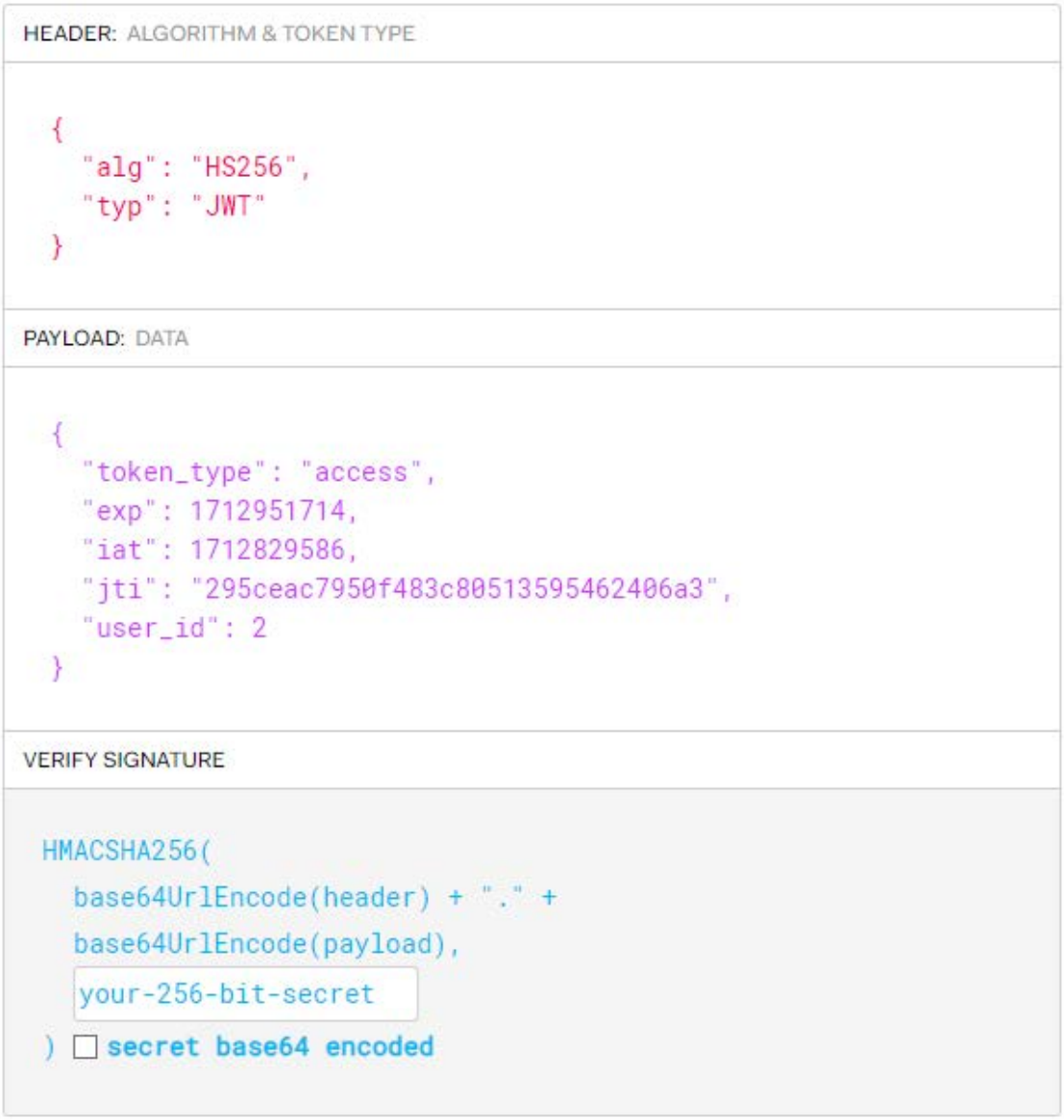
```
{
  "country": {
    "0": "AD",
    ...
    "n":"n záznam"
   },
  "name": {
    "0": "Sant Julià de Lòria",
    ...
    "n":"n záznam"
  },
  "lat": {
    "0": "42.46372",
    ...
    "n":"n záznam"
  },
  "lng": {
    "0": "1.49129",
    ...
    "n":"n záznam"
  }
}
```

Název atribut je pak uveden pouze jednou a lze data jednoduše indexovat. JSON soubory jsou uloženy v tomto formátu, jakmile je nad nimi provedena jakákoliv operace. Když je potřeba upravený soubor stáhnout, je index odstraněn a soubor vrácen do původní struktury.

11.4 JWT a oprávnění přístupu k souborům

11.4.1 JWT

Jak bylo nastíněno v kapitolách 9.1.3 a 9.2.2. aplikace využívá knihovnu SimpleJWT pro implementaci JSON Web Token pro celý framework Django. Aktivní a vyřazené tokeny jsou udržovány v databázi, na straně uživatele jsou v lokálním úložišti udržovány dva tokeny a to access_token a refresh_token. JWT jsou součástí skoro všech HTTP žádostí odcházejících ze strany frontendu na stranu backendu. Při přihlášení je uživatel autentizován a jsou mu vygenerovány token, které jsou uloženy a následně access_token je používán pro udělení oprávnění na uživatelské soubory a nabízené funkce aplikace. [52]



Obrázek 15: Dekódovaný JWT [zdroj: vlastní]

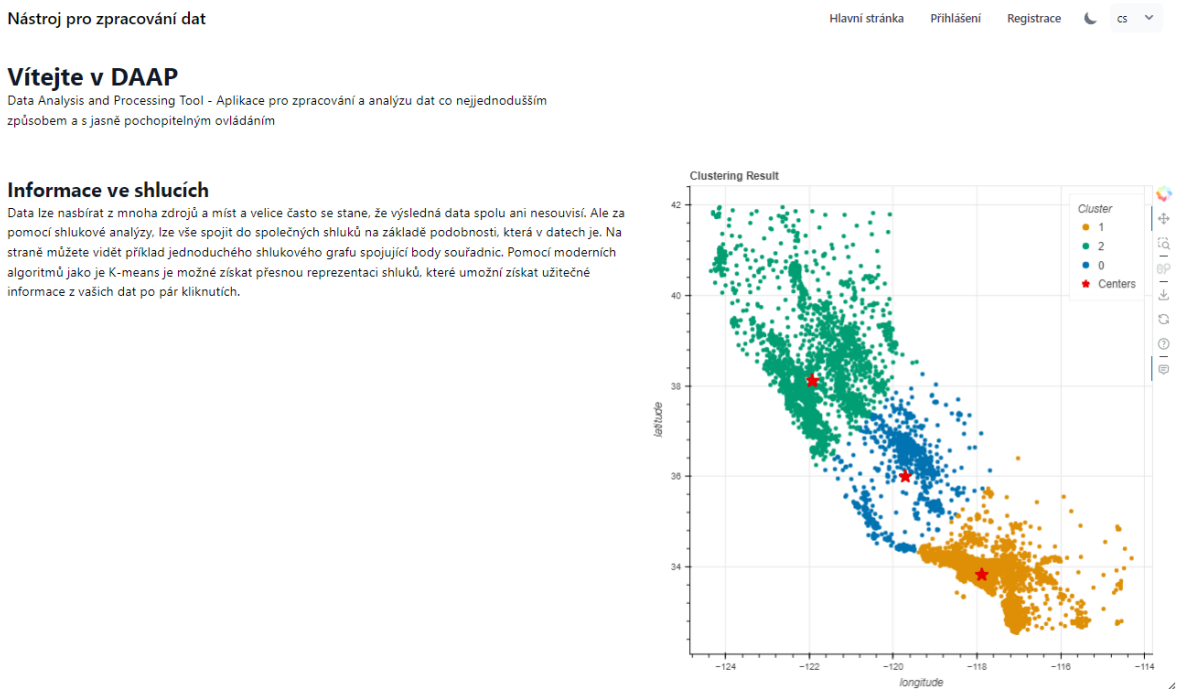
11.4.2 Přístup k /media a /upload složkám

I přesto, že aplikace nevytváří citlivá data je dobrou praktikou omezit přístup k souborům pouze pro uživatele, kterému patří. V případě aplikace je toto zajištěno několika způsoby. Pro nahrané a očištěné/zpracované soubory se při úspěšné operaci vloží záznam do databáze patřící uživateli, kterou tuto operaci provedl. Kdyby jiný uživatel znal uživatelské jméno jiného uživatele a znal i název souboru, který tento uživatel nahrál a podařilo se mu odeslat správnou HTTP žádost, nezíská data nebo soubor, protože pod jeho uživatelským ID není v databázi uložený odkaz na soubor. Toto je ale sekundární ochrana, primární je požadavek, aby veškeré HTTP žádosti přistupující k uživatelským datům byly opatřeny přístupovým JWT daného uživatele. Jak již bylo vysvětleno, tyto tokeny jsou integrovány přímo do frameworku Django, proto před provedením akce požadující nalezení uživatele v databází nebo získání jeho dat je uživatelské ID získáno z JWT. Získat tak přístup k cizím údajům by znamenalo získat jejich platný token. Pro generované grafy je zde ještě přidána funkcionalita vložení otisku parametrů grafu do názvu souboru, jak bylo vysvětleno v kapitole 11.2.1., není tak jednoduše zjistitelné, jak se vytvořený graf jmenuje. Stále platí ochrana pomocí JWT.

12 UŽIVATELSKÁ PŘÍRUČKA

12.1 První otevření webového rozhraní

Po úspěšné instalaci a nasazení aplikace, lze otevřít webovou adresu, která zobrazí úvodní stránku aplikace. Na úvodní stránce je vysvětleno k čemu aplikace je a jsou zde i některé grafy, které je aplikace schopna vytvořit. V horní části stránky je navigační menu, které v této situaci obsahuje pouze odkazy na registraci nebo přihlášení, přepínač světlého a tmavého zobrazení a výběr jazyka pro webovou stránku. Mód zobrazení a preferovaný jazyk se při výběru uloží do lokálního úložiště prohlížeče, a tato předvolba je dále respektována, když navštívíte stránku znovu. Pro zpřístupnění dalších funkcí aplikace je potřeba se registrovat a následně přihlásit.

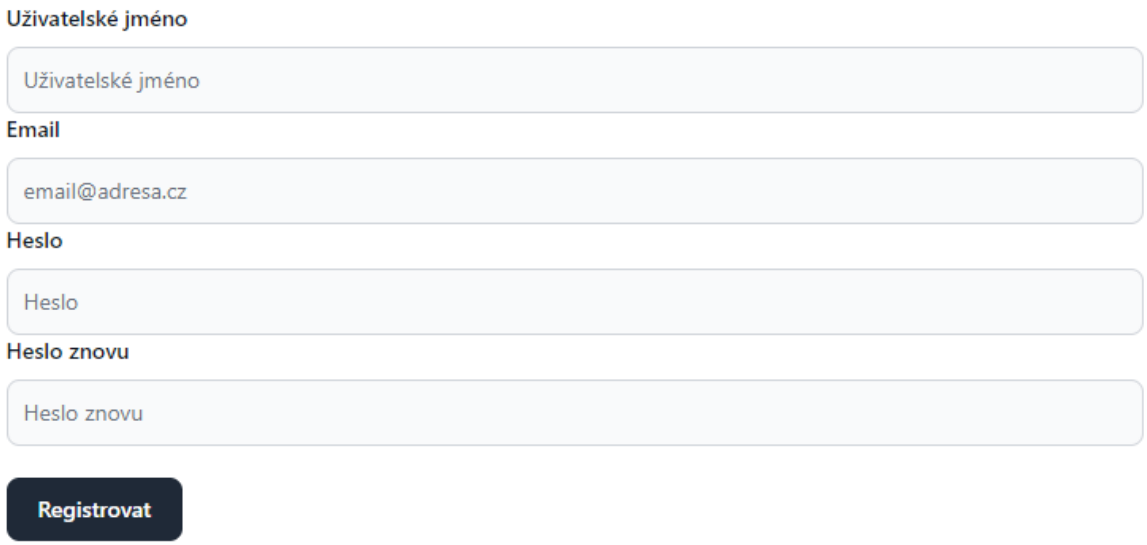


Obrázek 16: Úvodní stránka aplikace [zdroj: vlastní]

12.2 Registrace

Máte-li už účet (například údaje pro účet automaticky založený v aplikaci) přejděte k následující sekci Přihlášení

Stránka pro registraci obsahuje jednoduché formuláře, které musí uživatel vyplnit. Pro úspěšnou registraci je potřeba vyplnit všechny formuláře. Mezi ně patří uživatelské jméno, email, který musí mít validní formát, heslo a heslo znovu, hesla v těchto formulářích se musí shodovat. V případě, že uživatelské jméno nebo e-mail jsou již v aplikaci registrovány, zobrazí se uživateli upozornění s hláškou, které údaje je potřeba změnit. Na heslo jsou kladeny požadavky, které jsou uživateli zobrazeny pod formulářem pro heslo. Poslední částí je formulář pro zadání hesla znovu, heslo se musí shodovat s originálním heslem. Proběhne-li registrace úspěšně, opět se zobrazí upozornění, tentokrát informující o úspěchu.

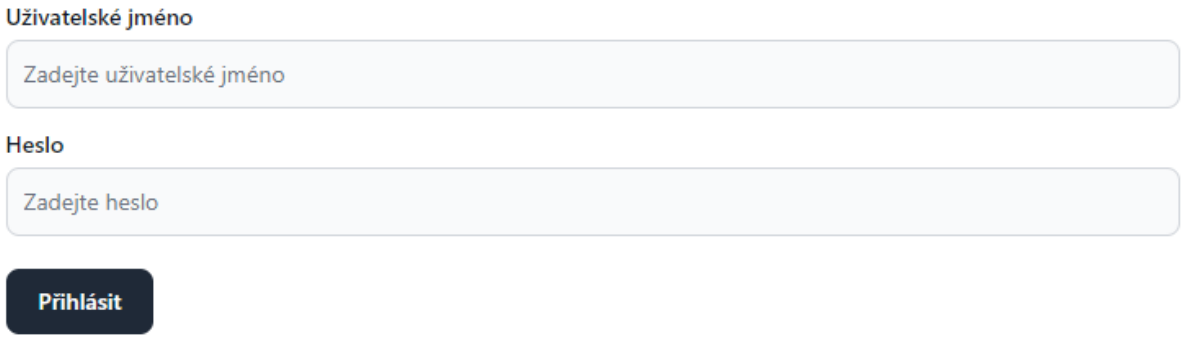


Obrázek 17: Formulář pro registraci [zdroj: vlastní]

12.3 Přihlášení

Po úspěšné registraci nebo s již existujícím účtem, lze přejít k přihlášení. Zde jsou pouze dva formuláře, jeden pro uživatelské jméno a jeden pro heslo. Stejně jako v registraci se v případě problému objeví uživateli upozornění, že přihlášení neproběhlo úspěšně, ale bez informace, který formulář je chybný, z důvodu bezpečnosti. V případě úspěšného přihlášení je uživatel přesměrován přímo na stránku pro provádění analýzy. Rozložení stránky se také částečně změní, v navigačním menu zmizí odkazy pro registraci a přihlášení, přibude odkaz na stránku

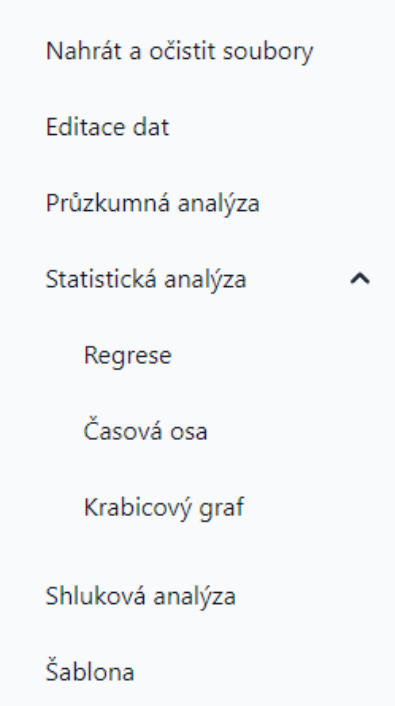
pro provádění analýz a na levé straně od přepínače pro světlé/tmavé zobrazení se objeví uživatelský avatar, který funguje jako „hamburgerové menu“, ve kterém lze najít tlačítko pro odhlášení. Zároveň jsou do lokálního úložiště prohlížeče uloženy dva JSON web tokeny, *access_token* a *refresh_token*. Ty jsou využívány k udržování platné relace spojení. Funkcionalita tokenů je objasněna v kapitole 11.4.1. JWT.



Obrázek 18: Formulář pro přihlášení [zdroj: vlastní]

12.4 Analýza

Zpřístupněna po úspěšném přihlášení, tato stránka je hlavním místem, kde lze využívat nabízené nástroje aplikace. Na levé straně se objeví nové navigační menu, které obsahuje odkazy pro zobrazení sekce Nahrání a očistění souborů, Editace dat, Statistická analýza, Shluková analýza a Šablona. Statistickou analýzu lze dále rozbalit pro zobrazení možností Regrese, Časové osy a Box plotů. Při přepínání mezi moduly se adresa nemění, pouze se vykreslí vyžádaný modul v prostoru na pravé straně od bočního navigačního menu.



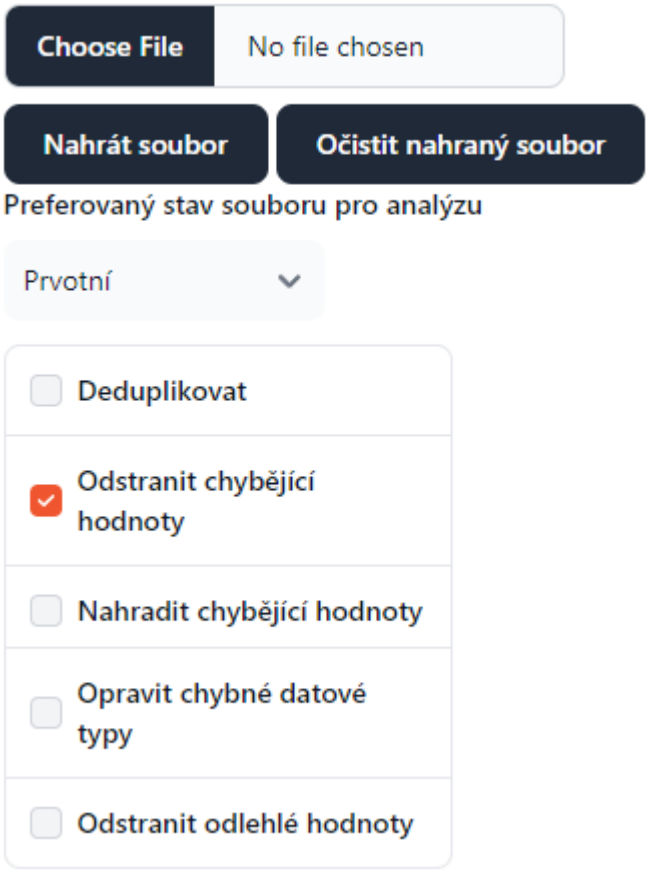
Obrázek 19: Boční menu pro přepínání mezi moduly [zdroj: vlastní]

12.5 Nahrávání a očištění souborů

Pro využití modulů analýz je zapotřebí nahrát (a případně očistit) soubor obsahující data. Hlavním prvek je zde formulář pro nahrání souboru, je možné nahrát pouze soubory typu CSV nebo JSON. Do formuláře lze vložit soubory kliknutím a výběrem v dialogovém okně, nebo přetáhnutím přímo do okna formuláře. Po vložení se objeví jméno souboru místo předešlého dočasného textu. Stačí pak jen kliknout tlačítko Nahrát soubor a soubor se odešle na server. V obou případech úspěšného a neúspěšného nahrání se objeví upozornění, informující o stavu a případně jaký problém nastal. Důležitým formulářem je výběr preferovaného typu souboru pro očištění a následné analýzy. Aplikace podporuje tři stavy souborů, a to jsou *prvotní*, *očištěný*, *zpracovaný*. Nahrané soubory jsou vždy typu *prvotní*, změnou preferovaného stavu lze uplatnit očištění na jiný typ souboru, který je na serveru. V případě, že vybraný stav souboru není na serveru, je brán *prvotní*, pokud i ten neexistuje je zobrazena chybová hláška. Nastavení stavu lze ponechat na *prvotní*, a je vhodné zvolit, které funkce pro očištění chcete uplatnit. Je nabízena deduplikace, nahrazení chybějících hodnot nejčastější hodnotou nebo průměrnou hodnotou, oprava chybných datových typů a odstranění extrémních hodnot. Není-li zvolena možnost nahrazení chybějících hodnot a ani možnost zachovat záznamy s chybějícími hodnotami, aplikace automaticky odstraní všechny záznamy, které mají chybějící hodnoty.

S vybranými možnostmi stačí kliknout na tlačítko *Očistit soubor* a aplikace provede očištění souboru v požadovaném stavu a vytvoří výstupní soubor ve stavu *očištěný*.

Je důležité poznamenat, že každý uživatel může mít pouze jeden soubor daného stavu. V případě nahrání nového souboru se odstraní i jeho očištěná verze, avšak zpracovaná verze zůstane. Nastavení preferovaného stavu také ovlivňuje, z jakého souboru se aplikace pokouší tvořit grafy. Je vhodné zmínit, že text formuláře pro nahrání souboru (to je tlačítko a pole, do kterého se vkládá soubor) se odvíjí od jazyka zvoleného v prohlížeči ne v aplikaci.

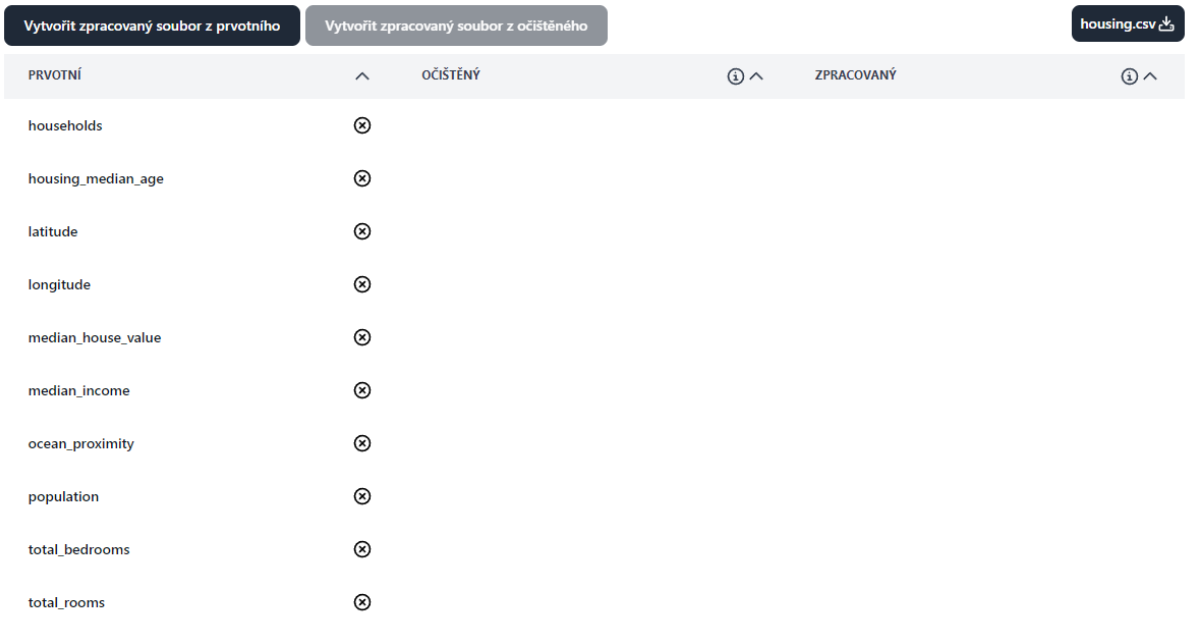


Obrázek 20: Menu pro nahrání a očištění souboru [zdroj: vlastní]

12.6 Editace dat

Další modulů nabízených v sekci Analýza, lze si zde zobrazit, jaké soubory má uživatel na serveru, a jaký je jejich obsah. Každý soubor je zde vyobrazen tabulkou, každý sloupec je jeden řádek. Názvy tabulek jsou stejné stavy souboru jako v předchozím modulu, je zde vidět i ikona šipky nahoru, která symbolizuje směr, jak jsou jména sloupců setříděná, výchozí nastavení je vzestupně. Kliknutím na hlavičku tabulky daného stavu lze otočit směr setříděný, změní se

i šipka reprezentující směr. U každého jména sloupce je i tlačítko pro odstranění sloupce ze souboru, toto upraví soubor bez nutnosti nahrát soubor znovu, ale změnu nelze vrátit. V případě, že uživatel nemá nějaký ze stavů souborů, je v hlavičce tabulky tohoto stavu informační ikona, která při najetí na ní myší zobrazí informaci, jak lze tento soubor vytvořit. Nad tabulkami se nachází dvě tlačítka, *Vytvořit zpracovaný soubor z prvotního souboru* a *Vytvořit zpracovaný soubor z očištěného souboru*. Zpracovaný soubor často slouží pro uložení verze zvoleného stavu, s možností odebrat některé sloupce a následnému použití i když se prvotní nebo očištěný soubor změní.

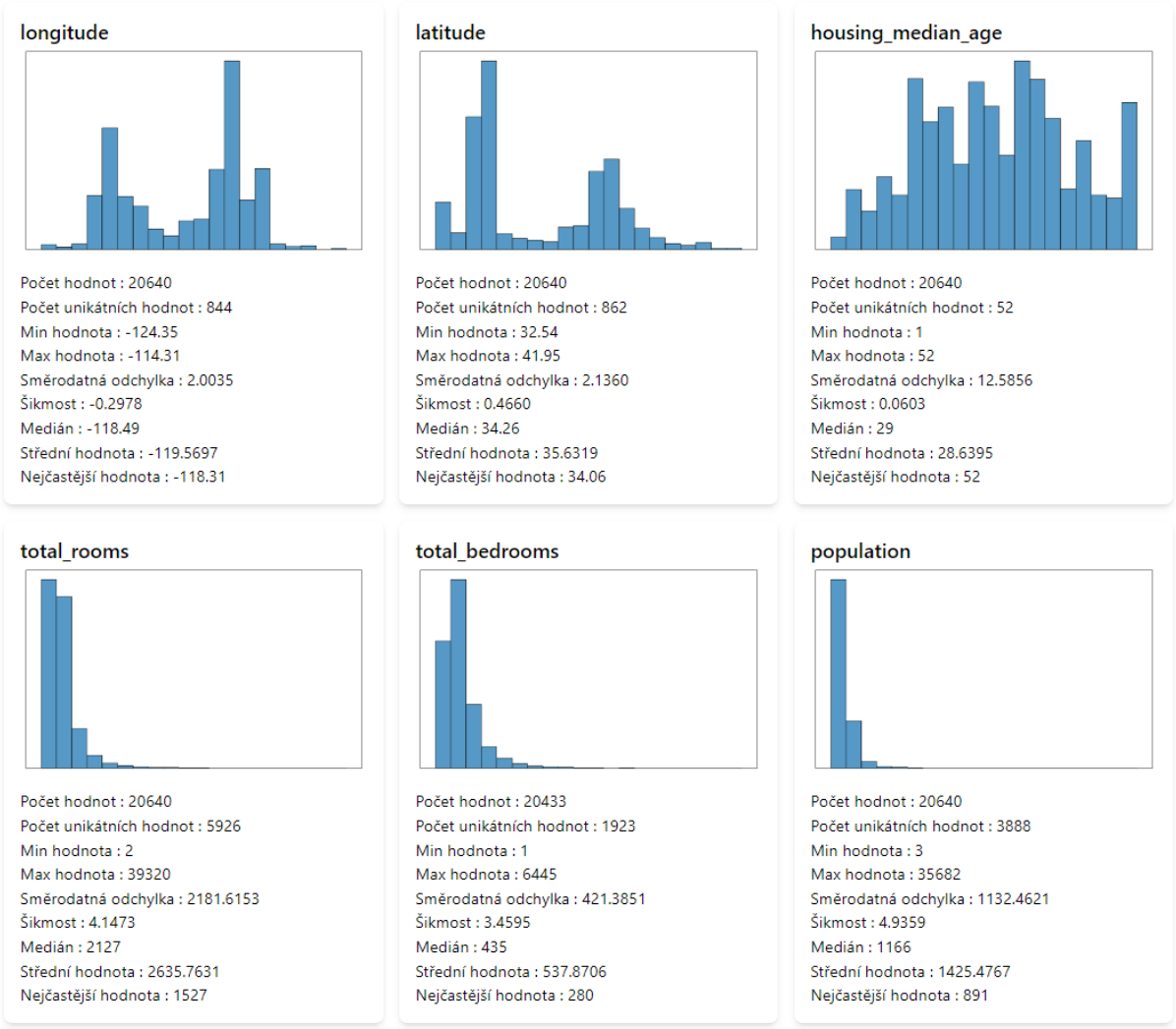


Obrázek 21: Rozložení editačního modulu [zdroj: vlastní]

12.7 Průzkumná analýza

Zde není zapotřebí vyplňovat žádné formuláře nebo nahrávat soubory, jedná se o modul, který má nastínit, jak vypadá rozložení dat v již nahraném souboru. Každý ze sloupců je zde zobrazen jako karta, její název je název sloupce, pod ním lze vidět graf znázorňující rozložení dat. Na pravé straně karty je soupis hodnot popisné statistiky, patří mezi ně min a max hodnota, střední hodnota, medián, směrodatná odchylka, modus a pak také informace jako je počet hodnot a počet unikátních hodnot. V případě, že některý ze sloupců nelze převést do grafu, je místo obsahu karty zobrazena varovná hláška. Grafy jsou tvořeny z preferovaného stavu souboru (viz. výběr v sekci 8.5 Nahrávání a očištění souborů), v případě že takovýto soubor neexistuje, je

brán stav *prvotní*. Pokud přejdete do této sekce bez nahrání souboru objeví se místo grafů upozornění, že nelze vytvořit grafy, protože neexistuje soubor.

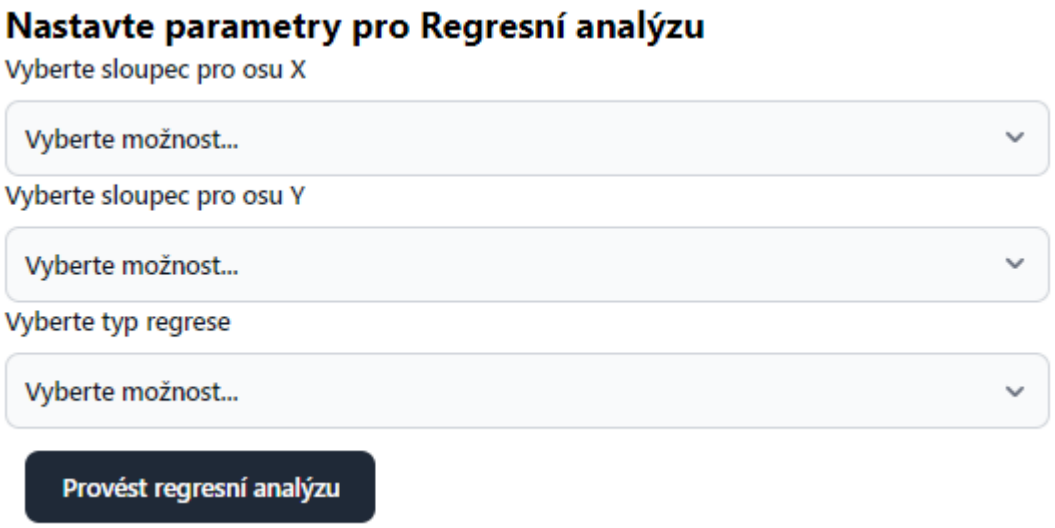


Obrázek 22: Příklad průzkumné analýzy [zdroj: vlastní]

12.8 Regrese

První modul z kategorie statistické analýzy schopný tvořit grafy obsahující regresní křivku. Požadované formuláře jsou sloupec pro osu X, sloupec pro osu Y a poté typ regrese. Podporované typy jsou lineární, exponenciální, polynomická. Nabízené sloupce pro výběr ve formuláři jsou brány z preferovaného stavu souboru (viz. výběr v sekci 8.5 Nahrávání a očištění souborů) a jsou zde pouze sloupce datového typu „číslo“. V situaci, kdy není zvolen typ regrese, je automaticky zvolena lineární. Pokud je zvolena polynomiální, objeví se zde dodatečný formulář pro výběr stupně. Platné hodnoty jsou pouze v rozmezí 1 až 4, jakákoliv jiná hodnota je zahozena a stupně jsou nastaveny na 2. Po kliknutí tlačítka Provést

regresní analýzu, jsou parametry odeslány a pod formuláři se objeví interaktivní graf. V případě chyby ve zpracování se objeví chybová hláška s informací, kde nastala chyba. Graf nabízí možnosti posunu v oblasti grafu, přiblížení na označenou sekci, přiblížení přes kolečko myši, reset pohledu, který vrátí náhled do původního stavu a možnost stáhnout graf, která v prohlížeči vytvoří upozornění s formulářem pro jméno souboru. Graf je stažen jako neinteraktivní PNG soubor v podobě, která byla nastavena před stažením neboli zachovává přiblížení a posun.

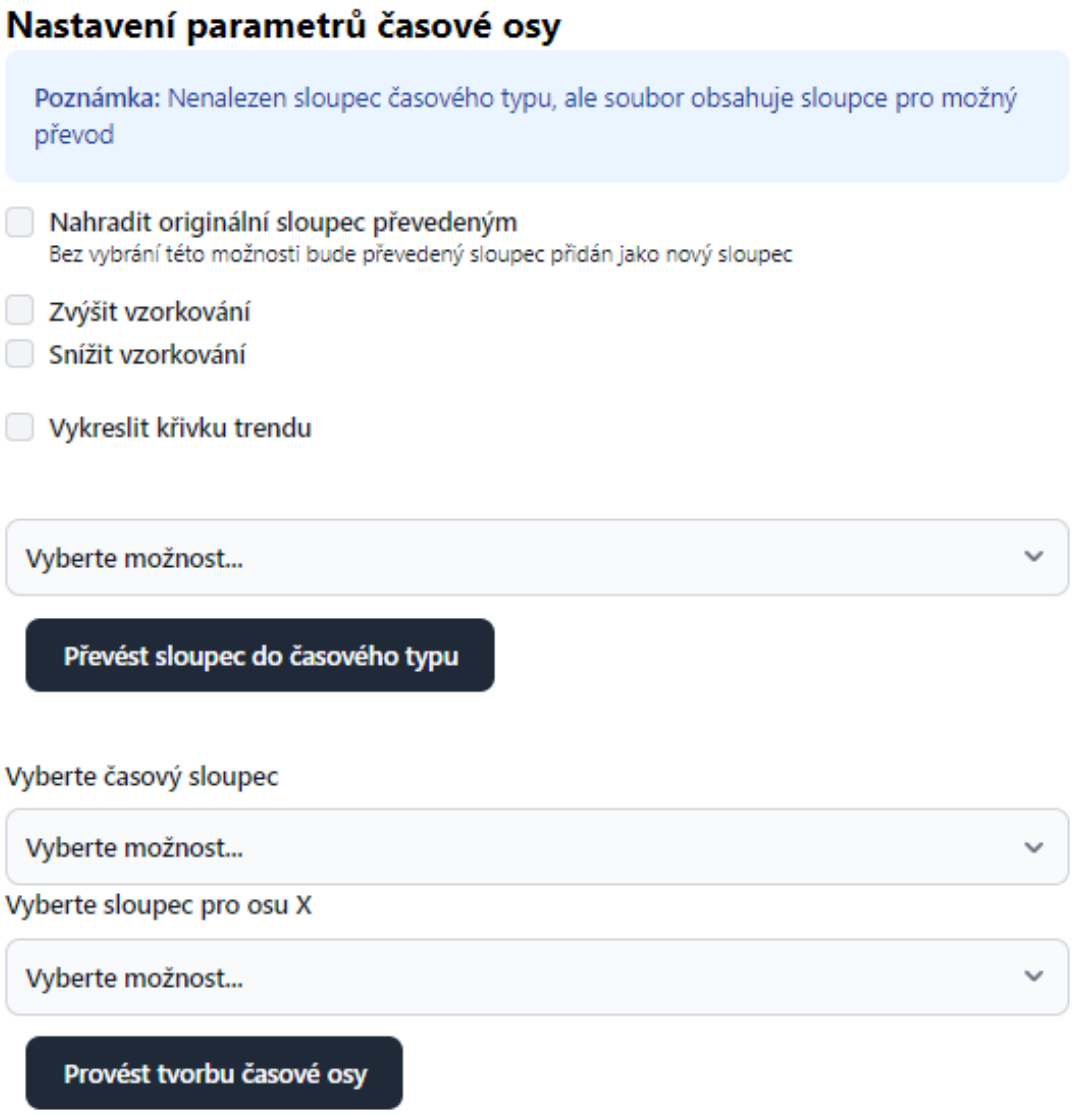


Obrázek 23: Nastavení regresní analýzy [zdroj: vlastní]

12.9 Časová osa

Druhý modul statistické analýzy nabízí vytvoření časové osy s možností změny vzorkování a vykreslení křivky trendu. Časová osa vyžaduje sloupec v souboru, který reprezentuje časovou hodnotu, například datum. Když je modul načten, jsou získány dostupné sloupce ze souboru, v případě, že žádný sloupec neodpovídá požadovanému formátu a neexistuje žádný sloupec s možností pokusit se o převod na datum, je místo formulářů zobrazeno varovné upozornění, že nad tímto souborem nelze provést analýzu časové rady. V situaci, kde nejsou žádné sloupce obsahující datum ve validním formátu, ale existují sloupce, které se lze pokusit převést, je zobrazen informační text vysvětlující, že se lze pokusit o převod. Výběrové menu pro převod nabízí všechny sloupce, které jsou ve formátu, který můžeme pokusit převést na datum. Je zde i možnost zvolit, jak má být zacházeno s převáděným sloupcem, zaškrtnutím *Nahradit originální sloupec* nahradí převedený sloupec výsledkem převodu, bez tohoto nastavení je do souboru přidán nový sloupec se sufixem. Po volbě sloupce pro převod a parametrů stisknutím tlačítka *Převést na časový sloupec* se aplikace pokusí provést převod,

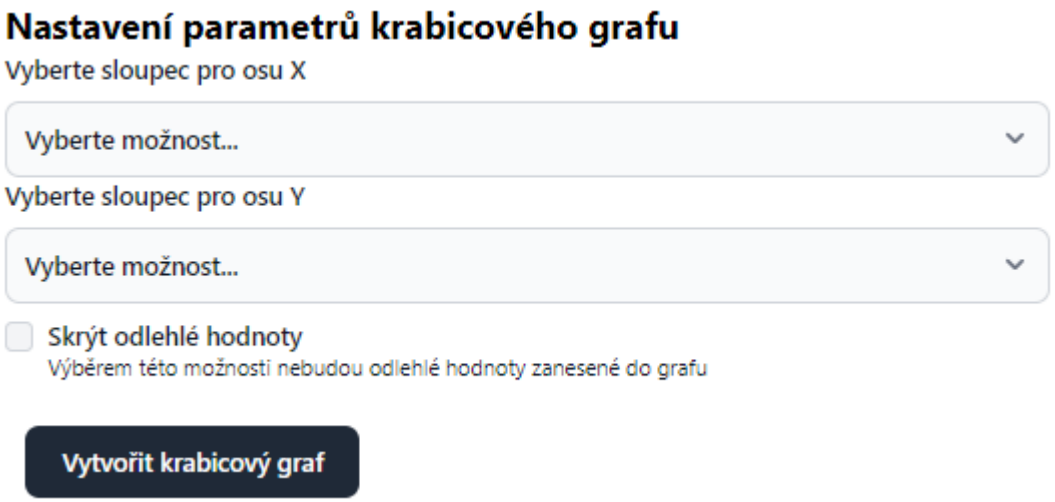
je-li úspěšný nově vytvořený sloupec (nebo nahrazený originální) se objeví ve *Vyberte časový sloupec*. Převod lze dále použít i na ostatní sloupce, které ještě nebyly převedeny. Následně stačí zvolit libovolný časový sloupec a sloupec reprezentující hodnoty pro body na časové ose a stisknout tlačítko *Provést časovou analýzu*. Pokud je analýza úspěšná, zobrazí se pod formuláři interaktivní graf. V případě chyby ve zpracování se objeví chybová hláška s informací, kde nastala chyba. Graf nabízí možnosti posunu v oblasti grafu, přiblížení na označenou sekci, přiblížení přes kolečko myši, reset pohledu, který vrátí náhled do původního stavu a možnost stáhnout graf, která v prohlížeči vytvoří upozornění s formulářem pro jméno souboru. Graf je stažen jako neinteraktivní PNG soubor, v podobě, která byla nastavena před stažením neboli zachovává přiblížení a posun.



Obrázek 24: Nastavení časové osy [zdroj: vlastní]

12.10 Box plot (krabicový graf)

Poslední modul statistické analýzy umožňuje vytvořit box plot, s možností skrýt odlehlé hodnoty. Jako v předchozích modulech jsou zde formuláře pro sloupec na ose X, sloupec na ose Y a již zmíněná možnost skrýt odlehlé hodnoty. Na volbu sloupců jsou kladeny minimální restrikce, prakticky pouze potřeba číselného sloupce na ose Y pro výpočet kvantilů. Nabízené sloupce jsou získány z preferovaného stavu souboru, případně z prvotního soubory, neexistuje-li preferovaný. Stejně jako ostatní moduly, otevření bez nahraného souboru zobrazí upozornění. Pro generování validních grafů je zapotřebí mít na ose X sloupec, který obsahuje malé množství unikátních hodnot, nejčastěji textových. Po volbě sloupců a možnosti, jestli skrýt odlehlé hodnoty, stisknutím tlačítka *Vytvořit box plot* se aplikace pokusí vytvořit interaktivní graf. V případě chyby ve zpracování se objeví chybová hláška s informací, kde nastala chyba. Graf nabízí možnosti posunu v oblasti grafu, přiblížení na označenou sekci, přiblížení přes kolečko myši, reset pohledu, který vrátí náhled do původního stavu a možnost stáhnout graf, která v prohlížeči vytvoří upozornění s formulářem pro jméno souboru. Graf je poté stažen jako neinteraktivní PNG soubor, v podobě, která byla nastavena před stažením neboli zachovává přiblížení a posun.

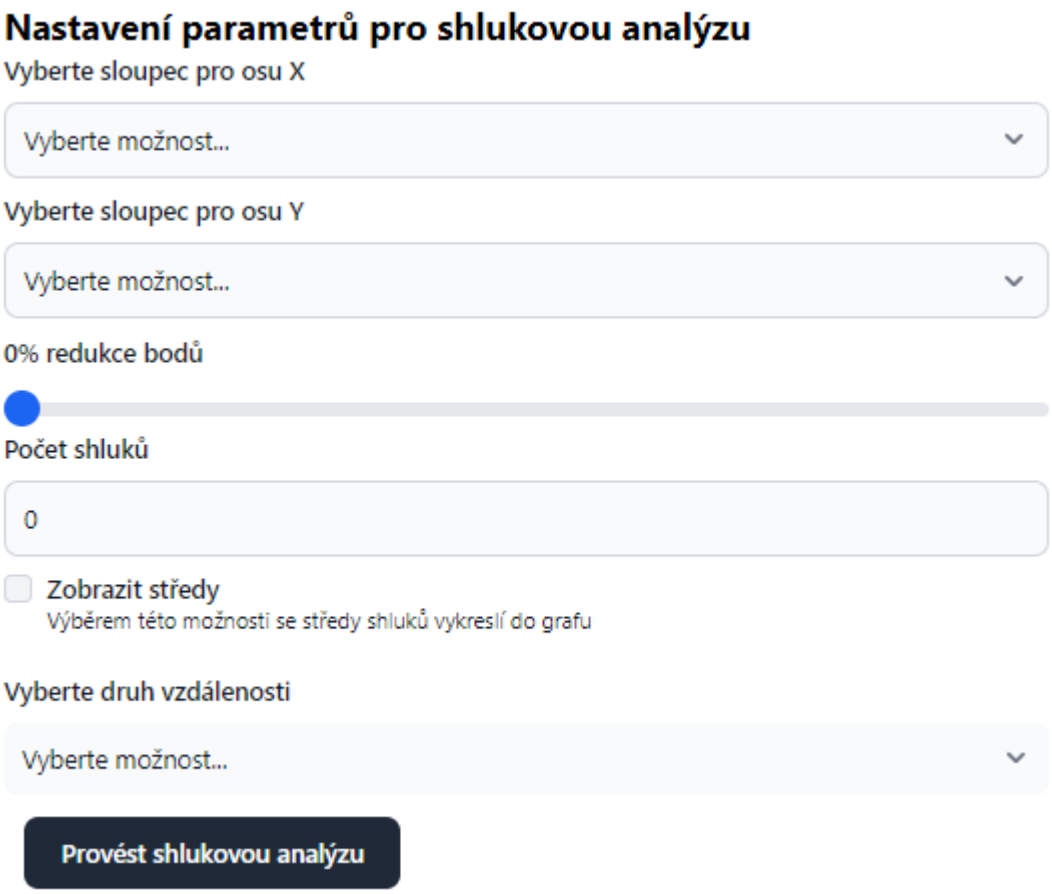


Obrázek 25: Nastavení krabicového grafu [zdroj: vlastní]

12.11 Shluková analýza

Jeden z nejrozsáhlejších modulů aplikace umožňující tvořit grafy shlukové analýzy s různými způsoby výpočtu vzdálenosti prvků ke středům. Podobně jako modul regrese jsou zde formuláře pro výběr sloupce pro osu X a sloupce pro osu Y, jsou zde dostupné pouze sloupce číselného

typu. Následně je zde posuvník nastavující, jak velkou redukci uplatnit na data, toto je potřeba v případě že počet záznamů v souboru je tak velký, že z bodů v grafu se stává jednolitý oddíl barvy a bez přiblížení není možné vidět individuální body. Povolená redukce je v hodnotách 0 až 70 procent. Pod tímto posuvníkem je volba počtu shluků, zde lze změnit počet shluků podle potřeby, například vygeneruje-li se graf kde hodnoty moc splývají do shluků, stačí zvýšit jejich počet. V opačném případě, kdy máme shluky, které v sobě mají sotva pár bodů, je vhodné snížit počet shluků. Zvolením možnosti *Zobrazit středy se do grafu* vloží i středy veškerých shluků. Poslední možností je volba funkce pro výpočet vzdálenosti prvku ke středu shluku. Nabízeny jsou funkce: Euklidovská vzdálenost, Euklidovská vzdálenost na druhou, kosinová vzdálenost, Manhattanská vzdálenost, Chebyshevova vzdálenost, Canberra vzdálenost a Chi vzdálenost na druhou. Tato volba ovlivňuje, jak budou prvky rozděleny do shluků, ale také může ovlivnit dobu běhu, než je vygenerován graf. Po nastavení požadovaných parametrů stačí stisknout tlačítko *Provést shlukovou analýzu* a počkat až bude graf vytvořen. V případě chyby ve zpracování se objeví chybová hláška s informací, kde nastala chyba. Graf nabízí možnosti posunu v oblasti grafu, přiblížení na označenou sekci, přiblížení přes kolečko myši, reset pohledu, který vrátí náhled do původního stavu a možnost *Stáhnout graf*, která v prohlížeči vytvoří upozornění s formulářem pro jméno souboru. Graf je stažen jako neinteraktivní PNG soubor, v podobě, která byla nastavena před stažením neboli zachovává přiblížení a posun.



Obrázek 26: Nastavení shlukové analýzy [zdroj: vlastní]

12.12 Šablona

Modul šablony kombinuje všechny předchozí moduly provádějící některou z analýz (tj. Regrese, Časová osa, Box plot, Shluková analýza). Šablona slouží pro konfiguraci analýz, které se mají uplatnit nad souborem s možností nahrát jiný soubor přímo v tomto modulu a ihned spustit přednastavenou analýzu. Při otevření se pokusí získat informace o existující šabloně, v případě že se jedná o první otevření a šablona neexistuje, je zde zobrazena informace podobně jako v modulu *Časové osy*, která informuje, že nebyla nalezena šablona, ale je možné ji vytvořit. Zobrazení je rozděleno na čtyři sekce, levá strana obsahuje rozbalovací menu, ve kterém lze najít aplikovatelné analýzy. Na pravé straně jsou ovládací prvky jako je formulář pro nahrání souboru, formulář pro vložení sloupce do dostupných sloupců. Je zde i seznam přidaných sloupců a sloupců, které jsou v momentálně nahraném souboru. Pro seznam přidaných sloupců je u každého záznamu tlačítko pro odebrání tohoto sloupce z přidaných sloupců. Pod seznamy jsou tlačítka *Uložení šablony* a *Načtení šablony*. Spodní část strany

obsahuje zbylé dvě části. Tlačítko *Uplatnit šablonu a vygenerovat grafy* slouží k aplikování šablony na soubor poté co je šablona připravena.

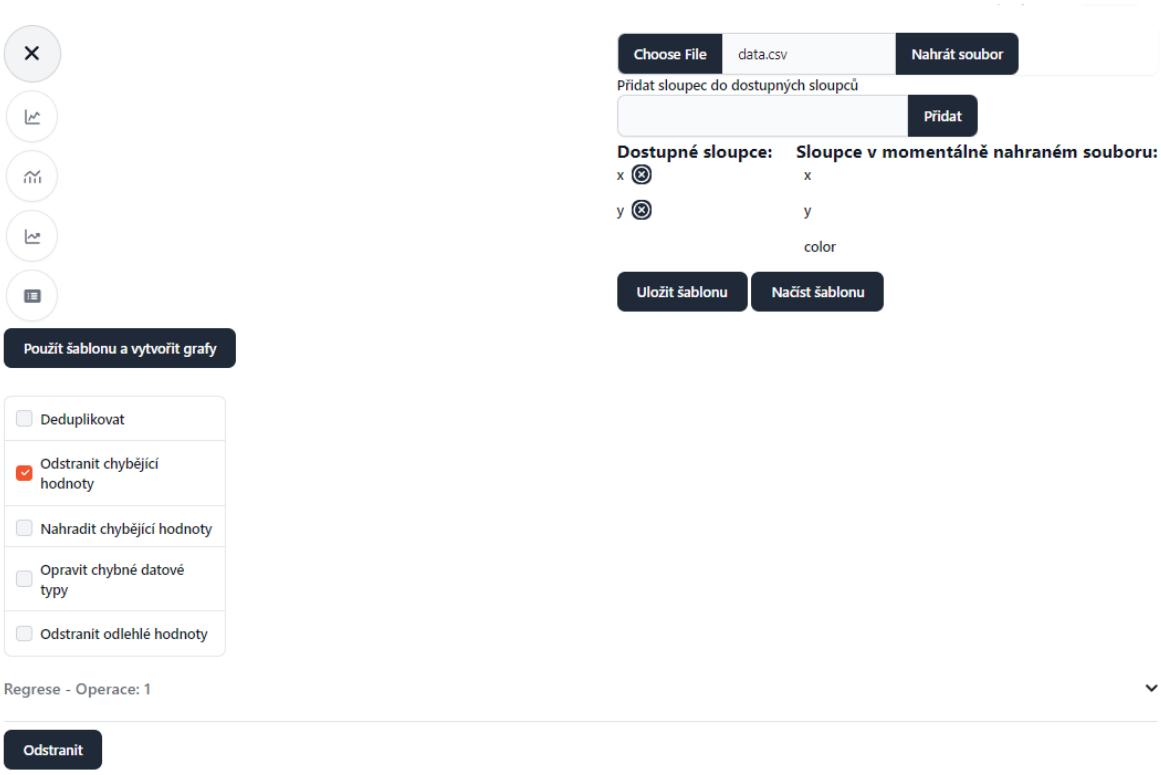
Je zde i seznam úprav, které lze uplatnit na soubor před tím, než jsou generovány grafy. Seznam sdílí funkce ze sekce *Nahrávání a očištění souborů*, to je deduplikace, nahrazení chybějících hodnot za pomoci nejčastější hodnoty nebo průměrné hodnoty, oprava chybných datových typů a odstranění odlehlých hodnot.

Zbytek spodní části obsahuje všechny přidané analýzy. Když se přidá analýza z rozbalovacího menu, je vložena do „akordeonového menu“, které obsahuje jednotlivé analýzy, jejich název reprezentuje typ operace a kolikátá v pořadí je, lze tyto položky rozkliknout pro zobrazení konfigurace. Jak možné parametry jednotlivých analýz ovlivňují výsledný graf si lze přečíst v předchozích podkapitolách této kapitoly (tj. kapitola 12.), v této podkapitole budou pouze zopakovány nabízené formuláře. Každý z formulářů požadující sloupec sdílí nabízené sloupce ze seznamu „dostupných sloupců“.

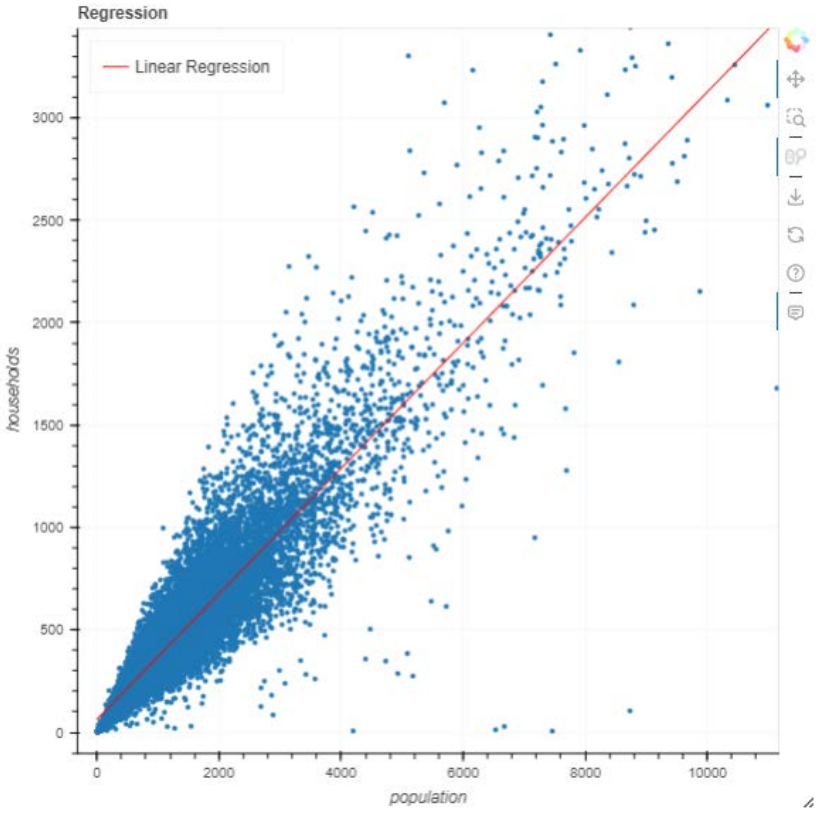
Regrese vyžaduje sloupec pro osu X a sloupec pro osu Y, volbu typu regrese a v případě, že se jedná o polynomickou, volba počtu stupňů.
Časová osa vyžaduje sloupec pro osu X (která v této analýze reprezentuje časovou stupnici), sloupec pro osu Y, volbu jestli se má změnit zvýšit nebo snížit vzorkování, pokud ano, tak podle jaké jednotky a o kolik jednotek, má-li se do grafu vložit trendová křivka. Shluková analýza požaduje sloupec pro osu X, sloupec pro osu Y, procentuální redukci datových bodů, počet shluků, možnost zobrazit středy a volbu funkce pro výpočet vzdálenosti. Boxplot požaduje sloupec pro osu X, sloupec pro osu Y a možnost skrýt odlehlé hodnoty. Obsah formulářů zůstává po celou dobu, co je modul otevřen.

V případě, že chcete modul opustit a zachovat nastavení analýz, je potřeba stisknout tlačítko *Uložit šablonu*. Pokud při uložení nastane chyba, objeví se upozornění. Uložení šablony uloží veškeré nastavení, od možností očištění souboru po jednotlivé analýzy. Každé následné otevření modulu šablony automaticky načte uloženou šablonu. V situaci, kde šablona existuje, ale aplikaci se jí nepodařilo načíst při otevření modulu, lze kliknout tlačítko Načíst šablonu. Toto tlačítko lze také použít v případě, že je požadováno zahodit provedené změny.

Při stisknutí tlačítka *Uplatnit šablonu a vygenerovat grafy* se pod spodním menu objeví načítací symbol, když je analýza dokončena objeví se zde interaktivní grafy. V případě že nastala chyba v tvorbě nějakého z grafů, je místo grafu zobrazeno upozornění s informací, co za chybu nastalo. Každý graf nabízí možnosti posunu v oblasti grafu, přiblížení na označenou sekci, přiblížení přes kolečko myši, reset pohledu, který vrátí náhled do původního stavu a možnost stáhnout graf, která v prohlížeči vytvoří upozornění s formulářem pro jméno souboru. Graf je stažen jako neinteraktivní PNG soubor, v podobě, která byla nastavena před stažením neboli zachovává přiblížení a posun.



Obrázek 27: Menu nastavení šablony [zdroj: vlastní]



Obrázek 28: Příklad grafu regrese vyprodukovaného šablonou [zdroj: vlastní]

13 MĚŘENÍ RYCHLOSTI APLIKACE

Pro vyhodnocení efektivity aplikace lze zapnout měření doby zpracování pro funkce pracující s daty přes nastavení MEASURE_FLAG v settings.py. Pro testování byly zvoleny datasety Museum Universe dostupný z Kaggle[11] a dataset Global Food Prices z Kaggle[12]. Dataset Museum Universe obsahuje okolo 33 tisíc záznamů a 44 sloupců, dataset Global Food Prices obsahuje 744 tisíc záznamů a 18 sloupců. Testovány byly moduly očištění dat, průzkumné analýzy a šablony. Pro všechny testy modulu očištění dat bylo aplikováno odstranění chybějících hodnot, pokud test neaplikoval nahrazení chybějících hodnot, nebude tento parametr uveden u aplikovaných operací. Všechny testy byly provedeny na počítači s procesorem Ryzen 5 2600 (doba generování grafu je závislá na výpočetním výkonu). Výsledky měření odpovídají očekávání a hodnoty zaokrouhleny na dvě desetinná místa jsou uvedeny v následující tabulce. Tyto informace lze nalézt na další stránce v Tabulka 1: Měření doby běhu pro různé operace očištění souboru, pro pochopení zkratek prováděných operací lze použít následující legendu:

Legenda: D – deduplikace, RM – odstranění chybějících hodnot, RF – nahrazení chybějících hodnot nejčastější hodnotou, RA – nahrazení chybějících hodnot průměrnou hodnotou, FD – oprava chybných datových typů, RO – odstranění odlehlých hodnot

[11] https://www.kaggle.com/datasets/mahdiehhajian/museum-universe-data-file

[12] https://www.kaggle.com/datasets/jboysen/global-food-prices

Tabulka 1: Měření doby běhu pro různé operace očištění souboru

Operace	Dataset Museum Universe (hodnoty uvedeny v sekundách)	Dataset Global Food Prices (hodnoty uvedeny v sekundách)
D	0,07	9,94
RF	0,17	6,24
RA	0,16	6,26
FD	0,06	43,3
RO	0,06	5,13
D+RF	0,18	9,29
D+RA	0,19	9,16
RF+FD	0,33	42,92
RA+FD	0,34	42,9
D+RF+FD	0,35	46,5
D+RA+FD	0,37	46,99
D+RF+FD+RO	0,36	48,69
D+RA+FD+RO	0,36	49,94

Se stejnými datasety byla spuštěna průzkumná analýza a šablona obsahující následující operace: Regrese, Krabicový graf, Shluková analýza. Výsledky lze nalézt v Tabulka 2: Měření doby běhu Průzkumné analýzy a Šablony

Pro dataset Museum Universe byly u regrese použity sloupce *ein* a *income* a druh regrese byl lineární. Pro krabicový graf sloupce *commonname* a *revenue*. Pro shlukovou analýzu sloupce *income* a *revenue* s nulovou redukcí bodů, tři středy a zapnutým zobrazením středů, jako vzdálenost byla použita Euklidovská.

Pro dataset Global Food Prices byly u regrese použity sloupce *mp_month* a *mp_price* a druh regrese byl lineární. Pro krabicový graf sloupce *mkt_name* a *mp_price*. Pro shlukovou analýzu sloupce *mp_price* a *mp_year* s nulovou redukcí bodů, tři středy a zapnutým zobrazením středů, jako vzdálenost byla použita Euklidovská.

Tabulka 2: Měření doby běhu Průzkumné analýzy a Šablony

Modul	Dataset Museum Universe (hodnoty uvedeny v sekundách)	Dataset Global Food Prices (hodnoty uvedeny v sekundách)
Průzkumná analýza	7,17	37,90
Průzkumná analýza po D+RF+RO	6,22	34,17
Šablona	0,05	52,65
Šablona po D+RF+RO	1,72	33,13

ZÁVĚR

Cílem práce bylo vytvoření aplikace pro nahrání, zpracování a vizualizaci dat. První kapitoly práce jsou zaměřeny na problematiku datové analýzy, vlastnosti dat a na možností získávání vědeckých datasetů. Dále jsou probírána témata vizualizace datasetů do podoby grafů, a jaké programovací jazyky nebo nástroje takové funkce nabízí. Další kapitoly jsou zaměřeny na předzpracování, kategorizaci a související uplatnění jazyka Python. Text vysvětluje i principy vlastní kolekce dat a jak redukovat a normalizovat nadměrný objem vlastností v datech.

Následně jsou vysvětleny jednotlivé funkce a metody, které jsou součástí aplikace Patří mezi ně například průzkumná analýza, shlukové algoritmy a změny vzorkování u časové osy. Po kapitolách vysvětlující sběr, předzpracování a uplatnitelných analýz se práce přesouvá k vysvětlení data miningu jako odvětví datové analýzy.

Práce poté pokračuje představením existujících funkčních a nefunkčních požadavků, diagramu použití a diagramu komunikace aplikace mezi jednotlivými vrstvami. V krátké sekci o nasazení aplikace je nastíněn postup, jak aplikaci spustit v prostředí Docker. Jak probíhal postupný vývoj modelu aplikace a jaká jsou možná budoucí rozšíření lze nalézt v kapitole Návrh a architektura aplikace.

Část pojednávající o volbě frameworků a knihoven, je rozdělena na část pro webové rozhraní (frontend) a kód aplikace provádějící analýzu (backend). V téže kapitole jsou uvedeny jednotlivé kroky, jak se vyvíjel databázový model aplikace.

V jedné z posledních částí práce jsou vysvětleny kódové sekce modulů a jak jsou zpracovaná data uložena a zabezpečena. V textu nechybí uživatelská příručka, která podává podrobné vysvětlení veškerých funkcionalit. Kromě výsledné aplikace vyvíjené společně s touto prací jsou zde i měření rychlosti aplikace při provádění různých nabízených funkcí.

POUŽITÁ LITERATURA

[1] B. BOURQUE, Linda a Virginia A. CLARK. *Processing Data: The Survey Example (Quantitative Applications in the Social Sciences #85)*. SAGE University: Sage Publications, 1992. ISBN 9780803947412.

[2] PATTERSON, Thomas F. a LEONARD, Jonathan G. *Turning spreadsheets into graphs: An information technology lesson in whole brain thinking*. Online. Journal of Computing in Higher Education. 2005, roč. 17, č. 1, s. 95-115. ISSN 1042-1726. Dostupné z: https://doi.org/10.1007/BF02960228. [cit. 2024-04-21].

[3] THE R FOUNDATION. What is R? THE R FOUNDATION. *The R Project* [online]. [cit. 2024-04-21]. Dostupné z: https://www.r-project.org/about.html

[4] PECINOVSKÝ, Rudolf. *Python: knihovny pro práci s daty pro verzi 3.11*. Myslíme v... Praha: Grada Publishing, 2023. ISBN 978-80-271-0659-2.

[5] SAS INSTITUTE INC. SAS: Analytics, Artificial Intelligence and Data Management. SAS INSTITUTE INC. *SAS Analytics Software & Solutions* [online]. c2024 [cit. 2024-04-21]. Dostupné z: https://www.sas.com/cs_cz/home.html

[6] ICG-CAPABILITY S.R.O. Minitab. ICG-CAPABILITY S.R.O. *Integrated Consulting Group* [online]. [cit. 2024-04-21]. Dostupné z: https://www.integratedconsulting.cz/software/minitab/?gad_source=1

[7] STATACORP LLC. Stata. STATACORP LLC. *Stata* [online]. c1996–2024 [cit. 2024-04-21]. Dostupné z: https://www.stata.com

[8] THE MATHWORKS, INC. MATLAB. THE MATHWORKS, INC. *MathWorks* [online]. c1996–2024 [cit. 2024-04-21]. Dostupné z: https://www.mathworks.com/products/matlab.html

[9] THE MATHWORKS, INC. JMP. JMP STATISTICAL DISCOVERY LLC. *JMP Statistical Discovery* [online]. c2024 [cit. 2024-04-21]. Dostupné z: https://www.jmp.com/en_us/home.html

[10] IBM. *SPSS*. Online. IBM. Dostupné z: https://www.ibm.com/spss. [cit. 2024-05-01].

[11] Orange data mining. UNIVERSITY OF LJUBLJANA. *Orange data mining* [online]. [cit. 2024-04-21]. Dostupné z: https://orangedatamining.com

[12] KNIME. KNIME. *Open for Innovation | KNIME* [online]. c2023 [cit. 2024-04-21]. Dostupné z: https://www.knime.com

[13] Tableau. SALESFORCE, INC. *Business Intelligence and Analytics Software | Tableau* [online]. c2024 [cit. 2024-04-21]. Dostupné z: https://www.tableau.com

[14] FAMILI, A.; SHEN, Wei-Min; WEBER, Richard a SIMOUDIS, Evangelos. Data Preprocessing and Intelligent Data Analysis. Online. *Intelligent Data Analysis*. 1997, roč. 1, č. 1, s. 3-23. ISSN 15714128. Dostupné z: https://doi.org/10.3233/IDA-1997-1102. [cit. 2024-04-21].

[15] LUENGO, Julián; GARCÍA-GIL, Diego; RAMÍREZ-GALLEGO, Sergio; GARCÍA, Salvador a HERRERA, Francisco. *Big Data Preprocessing*. Online. Cham: Springer International Publishing, 2020. ISBN 978-3-030-39104-1. Dostupné z: https://doi.org/10.1007/978-3-030-39105-8. [cit. 2024-04-21].

[16] AGGARWAL, Charu C. *Data classification: algorithms and applications*. Data mining and knowledge discovery series. Boca Raton, FL: CRC press, c2015. ISBN 1466586745.

[17] GOULD, Ronald. *Graph Theory*. Přetisk. Courier Corporation, 2012. ISBN 9780486498065.

[18] MIURA, Kota a SLADOJE, Nataša (ed.). *Bioimage Data Analysis Workflows – Advanced Components and Methods*. Online. Learning Materials in Biosciences. Cham: Springer International Publishing, 2022. ISBN 978-3-030-76393-0. Dostupné z: https://doi.org/10.1007/978-3-030-76394-7. [cit. 2024-04-21].

[19] NAVLANI, Avinash; FANDANGO, Armando a IDRIS, Ivan. *Python Data Analysis: Perform data collection, data processing, wrangling, visualization, and model building using Python*. 3rd ed. Packt Publishing, 2021. ISBN 9781789953459.

[20] AXINN, William G. a PEARCE, Lisa D. *Mixed Method Data Collection Strategies*. Online. Cambridge University Press, 2009. ISBN 9780521671712. Dostupné z: https://doi.org/10.1017/CBO9780511617898. [cit. 2024-04-21].

[21] SINGH, Tulika; GHOSH, Adarsh a KHANDELWAL, Niranjan. Dimensional Reduction and Feature Selection: Principal Component Analysis for Data Mining. Online. *Radiology*. 2017, roč. 285, č. 3, s. 1055-1056. ISSN 0033-8419. Dostupné z: https://doi.org/10.1148/radiol.2017171604. [cit. 2024-04-21].

[22] NAUMANN, Felix a HERSCHEL, Melanie. An Introduction to Duplicate Detection. Online. *Synthesis Lectures on Data Management*. 2010, roč. 2, č. 1, s. 1-87. ISSN 2153-5418. Dostupné z: https://doi.org/10.2200/S00262ED1V01Y201003DTM003. [cit. 2024-04-21].

[23] T. LAROSE, Daniel a Chantal D. LAROSE. *Discovering Knowledge in Data: An Introduction to Data Mining*. Druhá ed., Přetisk. John Wiley, 2014. ISBN 9780470908747.

[24] ANDERSON, T. W. a FINN, Jeremy D. *The New Statistical Analysis of Data*. Online. New York, NY: Springer New York, 1996. ISBN 978-1-4612-8466-6. Dostupné z: https://doi.org/10.1007/978-1-4612-4000-6. [cit. 2024-04-21].

[25] ROMESBURG, Charles. *Cluster Analysis for Researchers*. Ilustrovaná. Lulu.com, 2004. ISBN 9781411606173.

[26] S. EVERITT, Brian; LANDAU, Sabine; LEESE, Morven a STAHL, Daniel. *Cluster Analysis*. Pátá ed., Ilustrovaná. John Wiley, 2011. ISBN 9780470978443.

[27] ANALYTICS VIDHYA. Clustering | Different Methods, and Applications. KAUSHIK, Saurav. *Analytics Vidhya* [online]. 2024 [cit. 2024-04-21]. Dostupné z: https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

[28] WIKIPEDIA CONTRIBUTORS. Cluster analysis. *Wikipedia, The Free Encyclopedia* [online]. 2024 [cit. 2024-04-21]. Dostupné z: https://en.wikipedia.org/wiki/Cluster_analysis

[29] GOOGLE FOR DEVELOPERS. Clustering Algorithms. *Google for Developers* [online]. [cit. 2024-04-21]. Dostupné z: https://developers.google.com/machine-learning/clustering/clustering-algorithms

[30] LIKAS, Aristidis; VLASSIS, Nikos a J. VERBEEK, Jakob. The global k-means clustering algorithm. Online. *Pattern Recognition*. 2003, roč. 36, č. 2, s. 451-461. ISSN 00313203. Dostupné z: https://doi.org/10.1016/S0031-3203(02)00060-2. [cit. 2024-04-21].

[31] WIKIPEDIA CONTRIBUTORS. Lineární regrese. *Wikipedia, The Free Encyclopedia* [online]. 2024 [cit. 2024-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Lineární_regrese

[32] HRBÁČEK, David. *Zpracování časových údajů pro jejich vizualizaci* Online. Diplomová práce. Plzeň: Západočeská univerzita v Plzni, Fakulta aplikovaných věd. 2015. Dostupné z: https://theses.cz/id/w13muo/. [cit. 2024-04-16].

[33] ESRI. Create and use a time series graph. ESRI. *ArcGIS Insights* [online]. [cit. 2024-04-21]. Dostupné z: https://doc.arcgis.com/en/insights/latest/create/time-series.htm

[34] LEPOT, Mathieu; AUBIN, Jean-Baptiste a CLEMENS, François. Interpolation in Time Series: An Introductive Overview of Existing Methods, Their Performance Criteria and Uncertainty Assessment. Online. *Water*. 2017, roč. 9, č. 10. ISSN 2073-4441. Dostupné z: https://doi.org/10.3390/w9100796. [cit. 2024-04-21].

[35] STEINARSSON, Sveinn. *Downsampling Time Series for Visual Representation* [online]. University of Iceland, 2013 [cit. 2024-04-21]. Dostupné z: https://skemman.is/bitstream/1946/15343/3/SS_MSthesis.pdf. Disertace. University of Iceland. Vedoucí práce Jóhann Pétur Malmquist a Kristján Jónasson.

[36] MCGILL, Robert; TUKEY, John W. a LARSEN, Wayne A. Variations of Box Plots. Online. *The American Statistician*. 1978, roč. 32, č. 1, s. 12-16. ISSN 0003-1305. Dostupné z: https://doi.org/10.1080/00031305.1978.10479236. [cit. 2024-04-21].

[37] GUPTA, G.K. *INTRODUCTION TO DATA MINING WITH CASE STUDIES*. Třetí ed. PHI Learning Pvt., 2014. ISBN 9788120350021.

[38] YE, Nong, ed. *The Handbook of Data Mining*. CRC Press, 2003. ISBN 9781410607515.

[39] HAN, Jiawei; PEI, Jian a TONG, Hanghang. *Data mining: concepts and techniques*. Fourth edition. Cambridge: Morgan Kaufmann, [2023]. ISBN 978-0-12-811760-6.

[40] BRAMER, M. A. *Principles of data mining*. London: Springer, 2007. ISBN 1-84628-765-0.

[41] VAUGHAN, Daniel. *Analytical skills for AI and data science: building skills for an AI-driven enterprise.* Sebastopol, CA: O'Reilly Media, 2020. ISBN 1492060941.

[42] LIM, Greg a Daniel CORREA. *Django 4 for the Impatient*. Packt Publishing, 2022. ISBN 978-1803245836.

[43] Svelte.js for modern front-end development. *International Journal of Emerging Technologies and Innovative Research* [online]. 2020, 7(6), 206-211 [cit. 2024-04-21]. ISSN 2349-5162. Dostupné z: http://www.jetir.org/papers/JETIR2006034.pdf

[44] Pandas. PANDAS. *Pandas User Guide* [online]. c2024 [cit. 2024-04-21]. Dostupné z: https://pandas.pydata.org/docs/user_guide/index.html

[45] NumPy. NUMPY TEAM. *NumPy* [online]. c2024 [cit. 2024-04-21]. Dostupné z: https://numpy.org

[46] Matplotlib. MATPLOTLIB DEVELOPMENT TEAM. *Matplotlib* [online]. c2012–2024 [cit. 2024-04-21]. Dostupné z: https://matplotlib.org/stable/

[47] Seaborn. WASKOM, Michael. *Seaborn* [online]. c2012-2024 [cit. 2024-04-21]. Dostupné z: https://seaborn.pydata.org

[48] Bokeh. BOKEH CONTRIBUTORS. *Bokeh* [online]. c2024 [cit. 2024-04-21]. Dostupné z: https://bokeh.org

[49] SciPy. THE SCIPY STEERING COUNCIL. *SciPy* [online]. c2024 [cit. 2024-04-21]. Dostupné z: https://scipy.org

[50] Scikit-learn. KOMUNITA PROJEKTU SCIKIT. *Scikit-learn* [online]. c2024 [cit. 2024-04-21]. Dostupné z: https://scikit-learn.org/stable/

[51] PyClustering. NOVIKOV, Andrei. *PyClustering* [online]. c2024 [cit. 2024-04-21]. Dostupné z: https://pypi.org/project/pyclustering/

[52] Simple JWT. SANDERS, David. *Simple JWT* [online]. c2020 [cit. 2024-04-21]. Dostupné z: https://django-rest-framework-simplejwt.readthedocs.io/en/latest/

[53] BACCIANELLA, Stefano. JSON Repair. GITHUB, INC. *GitHub* [online]. c2024 [cit. 2024-04-21]. Dostupné z: https://github.com/mangiucugna/json_repair

[54] Flowbite Svelte. FLOWBITE™. *Flowbite* [online]. c2024 [cit. 2024-04-21]. Dostupné z: https://flowbite-svelte.com

[55] TORQUE, Krishna a tholle. Svelte-routing. NPM, INC. *Npm* [online]. 2024 [cit. 2024-04-21]. Dostupné z: https://www.npmjs.com/package/svelte-routing

[56] SARJEANT, John Jakob „Jake.“ Axios. *Axios* [online]. c2020 [cit. 2024-04-21]. Dostupné z: https://axios-http.com/docs/intro

[57] Django. DJANGO SOFTWARE FOUNDATION AND INDIVIDUAL CONTRIBUTORS. *Django* [online]. c2005-2024 [cit. 2024-04-21]. Dostupné z: https://docs.djangoproject.com/en/5.0/