FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Analýza chování studentů v systému MARAST
Student:	Bc. Jan Nováček
Vedoucí:	Ing. Magda Friedjungová
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Data miningovými metodami analyzujte data popisující chování studentů v on-line kvízech v systému MARAST. Analýza pomůže k pochopení chování studentů a vylepšení výukových procesů. Cílem práce je implementovat komponentu systému MARAST analyzující data z budoucích kvízů.

1. Seznamte se s poskytnutým exportem dat ze systému MARAST.
2. Data předzpracujte a analyzujte pomocí vybraných data miningových metod (metody vizualizace, hledání vztahů, shluková analýza, aj.). Výsledky analýzy interpretujte a vyberte zajímavé metody vhodné pro implementaci.
3. Navrhněte Python modul, který ze systému MARAST automaticky získá data, zpracuje je, analyzuje a výsledek prezentuje oprávněným uživatelům.
4. Navržené řešení implementujte. Kód musí být open source, dobře strukturovaný, řádně otestovaný sadou jednotkových i integračních testů, vhodně okomentovaný a dokumentovaný (obojí v anglickém jazyce).
5. Proveďte uživatelské testování výsledné komponenty, vyhodnoťte jej, a zpětnou vazbu zapracujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 5. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Analýza chování studentů v systému MARAST

Bc. Jan Nováček

Katedra softwarového inženýrství
Vedoucí práce: Ing. Magda Friedjungová

9. května 2018

Poděkování

Má vděčnost patří vedoucí této práce, **Ing. Magdě Friedjungové**, za její neocenitelné rady a doporučení, trpělivost, pozitivní přístup, pohotové konzultace a dostupnost.

Dále bych rád poděkoval **Ing. Tomáši Kalvodovi, Ph.D.** za jeho rady a nadšení pro věc, zejména při řešení věcí souvisejících se systémem MARAST, analýze chování studentů a interpretaci výsledků.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. května 2018 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2018 Jan Nováček. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Nováček, Jan. *Analýza chování studentů v systému MARAST*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Práce se zabývá implementací komponenty výukového systému MARAST za účelem analýzy dat z kvízů. Práce obsahuje seznámení se s doménou a dodaným exportem dat, který je dále podroben manuální analýze. Během analýzy byly použity metody matematické statistiky a vytěžování znalostí z dat, včetně metod strojového učení bez učitele jako je shlukování a detekce anomálií. Další část obsahuje vývoj analytické komponenty. Vývoj se sestává z analýzy, návrhu a implementace samotné komponenty, která byla vytvořena jako webová aplikace. Výsledkem práce je funkční komponenta pro analýzu dat z kvízů systému MARAST, která je otestována, zdokumentována a nasazena.

Klíčová slova Educational data mining, předzpracování dat, strojové učení bez učitele, MARAST, analýza chování, Python, analytická komponenta.

Abstract

This work deals with implementation of a component of MARAST educational system in order to analyse data from quizes. The work includes introduction to the domain and to the provided data export that is further subjected to manual analysis. During analysis, methods of mathematical statistics and data mining were used, including methods of unsupervised machine learning

like clusternig and anomally detection. Next part of this work contains development of analytical component. Development consists of analysis, design and implementation of the component that was produced as web application. The result of this work is a functional component for data analysis of quizes from MARAST system, component is tested, documented and deployed.

Keywords Educational data mining, data preprocessing, unsupervised machine learning, MARAST, behavior analysis, Python, analytical component.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Informační systém MARAST (MAtematika RAdoSTně) slouží jako nástroj pro studenty a vyučující za účelem dosažení lepších výsledků vzdělávání na Fakultě informačních technologií Českého vysokého učení technického v Praze. Hlavní funkcionalitou systému je testovací portál, ve kterém vyučující vytvoří kvíz a k němu otázky. Student na dané otázky odpovídá a podle výsledků je ohodnocen. Systém je v provozu již několik let a za tu dobu se nahromadilo několik set tisíc odpovědí od studentů. Díky rozvoji datových věd se nabízí možnost prozkoumat tyto odpovědi a pokusit se v nich najít souvislosti či vzorce chování za účelem poskytnutí zpětné vazby vyučujícím (pro zlepšení poskytované výuky) či samotným studentům (pro zlepšení procesu učení). Cílem této práce je provést analýzu nashromážděných dat a vytvořit nástroj, který by byl schopen analýzu provádět automaticky, včetně prezentace výsledků.

Práce je rozdělena na dvě hlavní části. První část práce se zabývá teorií, rešerší a manuální analýzou dat. V první kapitole je seznámení s původcem dat, tj. systémem MARAST. Dále je část věnována rešerši podobných aplikací analýzy dat ve výukových zařízeních. Následuje popis procesu předzpracování a analýzy dat a zvolené metodiky. Posledním bodem je představení nástrojů použitých pro analýzu dat a implementaci modulu.

Druhá kapitola teoretické části práce se věnuje práci s poskytnutým exportem dat z MARASTu. To se sestává z popisu domény a dat, specifikace cílů analýzy, přípravy a předzpracování dat, jejich modelování a nakonec prezentací a vyhodnocením výsledků analýzy.

V druhé části práce se věnuji především praktické stránce. První kapitola je věnována realizaci modulu. Obsahuje popis vývoje modulu pro automatickou analýzu dat, tedy celý proces od analýzy až po nasazení a dokumentaci. Druhá kapitola v praktické části je pak věnována vyhodnocení, prezentaci výsledků a zapracování změn na základě zpětné vazby z testování.

V závěru se nachází shrnutí odvedené práce a sada doporučení pro další rozvoj.

Cíl práce

Jak již bylo řečeno v úvodu, hlavním cílem práce je provést analýzu dat ze systému MARAST a vytvořit nástroj, který je danou analýzu schopen provést automaticky. To se skládá z několika dílčích úkolů, které je potřeba k dosažení cíle splnit.

Prvním úkolem je seznámit se s daty ze systému MARAST. Je nutné zjistit, o jaká data se jedná, co obsahují, v jakém formátu byla dodána apod.

Na základě toho je možné zvolit vhodné nástroje a postupy ve zpracování a datové analýze. Pomocí zvolených metod a nástrojů jsou data převedena na informace, respektive znalosti, za účelem pochopení informací a souvislostí mezi nimi.

Třetím úkolem je získané znalosti správně interpretovat a využít. Jinak řečeno vybrat použitelné a smysluplné informace určené přímo pro vyučující (resp. studenty) a poskytnout jim onu zpětnou vazbu. Důležitou částí je také volba vhodné prezentace výsledků, ať už textově, nebo graficky.

Automatizace tohoto procesu je dalším krokem. Modul by měl být schopen provádět analýzu dat, interpretaci a prezentaci výsledků automaticky na vyžádání i z nových dat. To vše je také potřeba otestovat — z pohledu software, z pohledu uživatele i z pohledu správnosti analýzy nových dat.

Část I

Teoretická část

Kapitola 1

Rešerše

Kapitola je věnována popisu domény, technologiím a zdrojům.

1.1 MARAST

V sekci jsou bližší informace o zdroji dat určených k analýze, tedy systému MARAST.

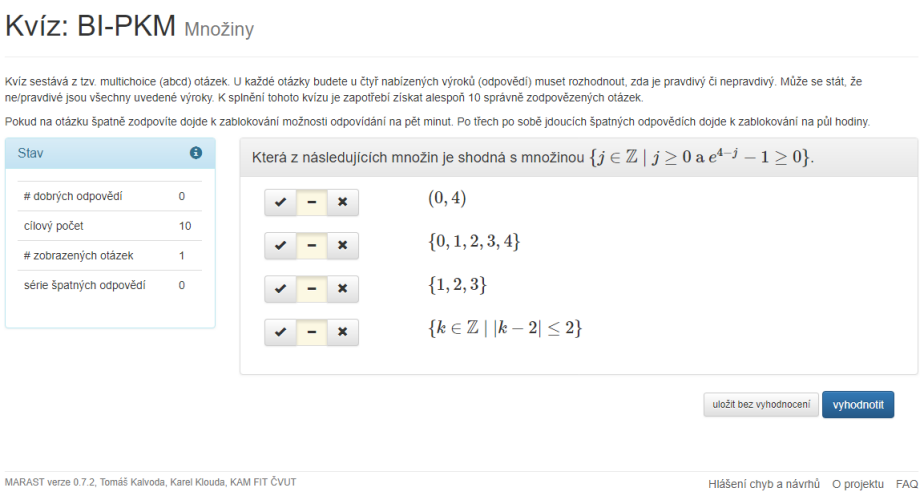
Jak již bylo řečeno, MARAST je nástroj pro podporu výuky. Vznikl na Katedře aplikované matematiky (KAM) FIT ČVUT a původně sloužil hlavně pro matematické předměty prvního ročníku bakalářského programu Informatika (Základy matematické analýzy, Přípravný kurz matematiky, Lineární algebra), nicméně se jeho působnost rozšířila i do dalších předmětů (např. bakalářský předmět Algoritmy a grafy, magisterský předmět Matematika pro informatiku).

Důvody vzniku tohoto systému byly prosté — zvýšit průchodnost předmětů, dát studentům do určité míry interaktivní pomůcku k učení a v neposlední řadě usnadnit práci vyučujícím pomocí automatizace opravování testů, čímž by se i snížily další náklady.

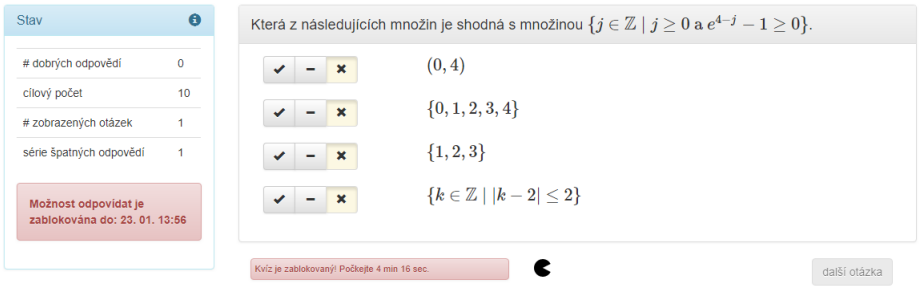
Uživatel se systémem interaguje přes webové rozhraní. Učitelé vytvářejí kvízy a kvízové otázky, studenti na ně odpovídají a jsou za to hodnoceni. Kvízy mají při vytváření několik parametrů — název kvízu, pro který předmět je kvíz určen, kolik otázek je potřeba zodpovědět správně apod. Otázky mají také různé parametry — znění otázky, správné odpovědi, počet bodů apod. Přesná specifikace pro nás důležitých parametrů, které byly poskytnuty v exportu dat, je v příslušné kapitole.

Otázky se studentům zobrazují na webové stránce Obr. 1.1, kde vidí zadání otázky společně s možnostmi odpovědí. Některé otázky mají místo seznamu možných odpovědí textové pole, do kterého uživatel zadá svou odpověď.

V případě správné odpovědi může student pokračovat v řešení kvízu, nebo kvíz dokončit v případě, že dosáhl cíle.



Obrázek 1.1: Odpovídání na otázku v systému MARAST



Obrázek 1.2: Zablokování odpovídání v systému MARAST

V případě špatné odpovědi platí totéž, ale u některých kvízů je nastaveno zablokování možnosti odpovídání Obr. 1.2. Toto zablokování znemožní studentovi přejít na další otázku po určitou dobu. Aktuálně jsou v systému využívána dvě zablokování. První zablokování funguje vždy po první špatné odpovědi a obvykle trvá kratší dobu, např. 5 minut. Druhé zablokování se spouští po sérii po sobě jdoucích špatných odpovědí (nepřerušená série špatných odpovědí, obvykle délky 3) a možnost odpovídat na otázky je zakázána na delší dobu (většinou po delší dobu než první zablokování, např. 30 minut). Nastavení délek dob obou zablokování může být pro každý kvíz různé, stejně tak délka série špatných odpovědí potřebných pro druhé zablokování.

1.2 O analýze dat

Analýza dat prožívá v poslední době velký rozmach. Ten byl ovlivněn rozvojem v mnoha oblastech počítačových věd, umělé inteligence, kybernetiky, Internetu atd. V těchto oblastech se za mnoho let nahromadilo velké množství dat.

Ovšem pouhé uložení dat nemá hodnotu, jen zabírají místo v úložištích. Co hodnotu má jsou znalosti, které je možné z dat vyčíst. Velké množství dat je ovšem nemožné zanalyzovat pouze lidskými silami. Místo toho je lze podrobit automatizované analýze prováděné počítačem s cílem vyčíst souvislosti z dostupných dat, lidským očím často neviditelné.

Přestože jsou nyní používané postupy a technologie známé už několik desítek let, je rozvoj právě v této době umožněn díky výkonnějších výpočetním strojům a efektivním implementacím algoritmů. Od sekvenčních algoritmů se přešlo k paralelním, které navíc mohou běžet distribuovaně na několika výpočetních uzlech, propojených sítěmi s dostatečnou kapacitou přenosu.

Jako příklad využití analýzy dat lze uvést doporučovací systémy v internetových obchodech nebo videoportálech. V případě obchodů může systém doporučit uživateli další zajímavé položky, které jsou určeny na základě historie prohlížení uživatele a například následném přidružení uživatele ke skupině uživatelů s podobnou historií. V případě videoportálu je filozofie podobná — doporučení dalších videí na základě chování uživatele.

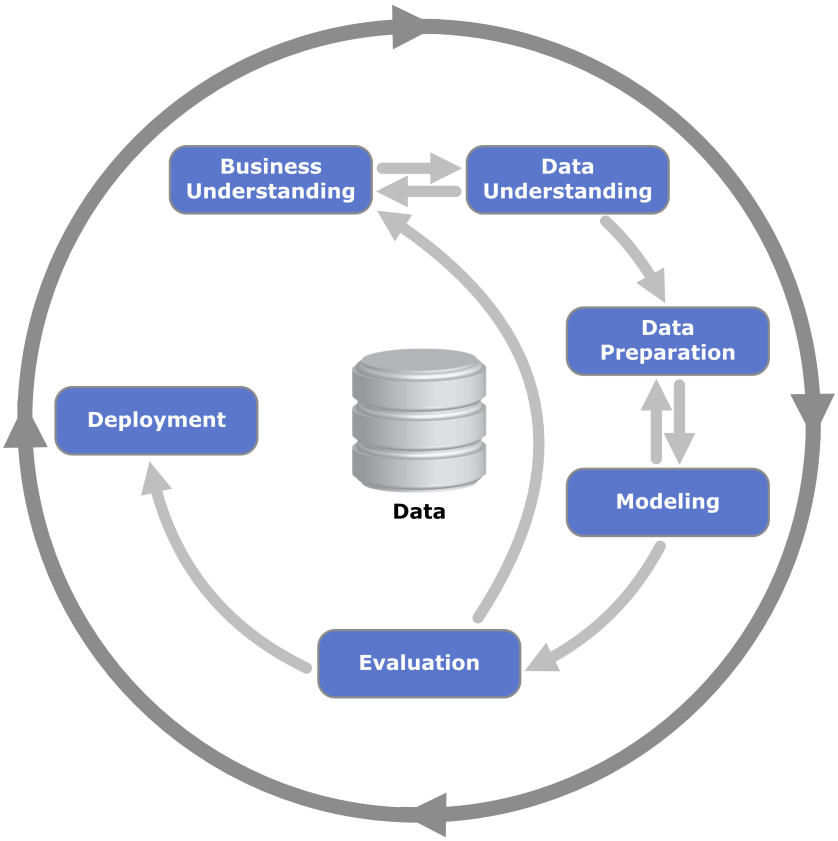
Jedno z odvětvích zabývající se analýzou dat se nazývá dolování dat (ang. *Data mining*). Zahrnuje v sobě matematickou statistiku, pomocí které se provádí zmíněné analýzy. Dolování dat pracuje s velkým množstvím dat, která jsou většinou uložena v databázích nebo datových skladech. Dále využívá algoritmy z odvětví tzv. strojového učení [2] (ang. *Machine learning*), které pomáhá hledat souvislosti v datech.

Proces dolování dat nám tedy dá nějaké informace obsažené v datech, ale ty je potřeba nutné dále pochopit a tedy získat z nich znalosti a zkušenosti, které dokážeme nějak využít. Proces získávání nových znalostí z dat je znám pod anglickým názvem *Knowledge Discovery* a jeho zkratkou KD. Dolování dat je pak jedním z kroků KD.

V souvislosti s tím se často uvádí zkratka KDD z anglického *Knowledge Discovery in Databases*, což značí proces KD aplikovaný na databáze. Zobecněný proces, kdy je proces KD aplikovaný na jakýkoliv zdroj dat (nejen databáze), je znám jako KDDM z anglického *Knowledge Discovery and Data Mining*.

1.2.1 Získávání znalostí

Proces získávání znalostí byl zmíněn již v předchozí kapitole. Zjednodušeně jej lze popsat jako získání užitečné informace (znalostí) z nějakého zdroje. Jedná se o proces s vstupy (např. data) a výstupy (informace, znalosti). Proces je to



Obrázek 1.3: Cyklus CRISP-DM [1]

vcelku komplexní, bylo vytvořeno několik postupů a návodů jak na akademické půdě, tak v průmyslu [3].

Pro naší analýzu byla zvolena metodika CRISP-DM (ang. *Crossindustry standard process for data mining*). Pro její úplně znění doporučuji [3, 4, 5, 1], zde je postup uveden pouze bodově, s popisem těch nejdůležitějších částí procesu. Cyklus je také znázorněn na Obr. 1.3.

1. Pochopení zadání a domény. Účelem kroku je pochopení cílů projektu a požadavků zákazníka. Dále je nutné pochopit doménu zákazníka, čím se zabývá a na co se zaměřit. Následně je nutné definovat cíle vytěžování znalostí z dat v technické řeči, tj. co přesně je nutné udělat.

2. Pochopení dat. Získání počátečních dat a jejich hrubý popis (co obsahují). Dále se provede prvotní průzkum dat s jednoduchou statistickou analýzou, což může mít vliv na původní cíle vytěžování znalostí. Po pochopení dat by se měla provést kontrola kvality dat, pokud je potřeba.

3. Příprava dat. S ohledem na definované cíle, kvalitu dat a technická omezení se vyberou vhodná data. Ty se následně vyčistí, resp. dojde ke zvýšení jejich kvality, s ohledem na použité metody analýzy, které mohou mít různé požadavky. Dále může dojít ke konstrukci pomocných atributů, vzájemné integraci dat z více zdrojů a formátování.

4. Modelování. Výběr modelu, který se aplikuje na data (např. algoritmy shlukování, neuronové sítě, rozhodovací stromy atd.). Dále se specifikují testovací procedury (chyba modelu, rozdělení dat na cvičná a testovací). Vytvoří se model na zvolených datech a výsledný model se posoudí, nutné přihlédnout k doméně.

5. Celkové vyhodnocení. V kroku dojde k vyhodnocení výsledného modelu a výsledků analýzy. Celý proces a všechny aktivity se zrevidují za účelem nalezení míst k zlepšení výsledků. Na konci se rozhodne, co dál — pokud výsledky vyhovují, model se může nasadit. Případně se proces upraví a provede znovu. Je nutné brát v potaz zdroje a další možnosti (finance, čas).

6. Nasazení a prezentace výsledků. Finální krok, kdy se model nasadí do ostrého provozu a výsledky analýzy lze prezentovat.

1.2.2 Dolování dat ve vzdělávání

Dolování dat lze využít i ve vzdělávacích institucích nebo na internetových vzdělávacích portálech. Při použití lze například přizpůsobit výuku (otázky, materiály atd.) studentovi na základě modelu získaného z jeho studijních výsledků. Dolování nad daty ve vzdělávání je pak ve zkratce označeno jako EDM z anglického *Educational Data Mining*.

Systematický přehled nejnovějších výzkumných projektů a aplikací EDM nabízí [6] V tomto článku autoři provedli průzkum literatury zabývající se EDM od roku 1983. Nabízí tak rozsáhlý přehled mnoha výzkumných článků a děl, které se zabývají např. shlukováním studentů na základě jejich výsledků v e-learningových kurzech, inteligentními výukovými systémy, nebo doporučováním předmětů na vysoké škole. V článku je kladen důraz na algoritmy shlukování, součástí je seznam použitých algoritmů a typ aplikace.

V České republice se EDM zabývají například na Masarykově univerzitě v Brně, kde se zaměřují na adaptivní učení. V jejich případě se jedná o přizpůsobování výukových materiálů a kvízů výkonům studenta, tedy aplikaci metod EDM, vytvoření modelu studenta a individuální zpětné vazby. Jejich cílovou skupinou jsou mladší studenti (základní a střední školy). Z výsledků jejich práce lze čerpat inspiraci k tomu, na které parametry chování studenta a metody DM se zaměřit. Například [7, 8] se věnuje zkoumání doby odpovídání na otázku a jejím vlivu na kvalitu modelu studenta. Na to navazuje [9], který se zabývá modelováním studenta pro systém výuky matematiky

pro děti. V tomto článku je pak vliv doby odpovídání na otázky při odhadu znalosti studenta ještě více zdůrazněn, dále jsou v článku popsány další možné ukazatele znalostí jako nezodpovězené otázky a špatně zodpovězené otázky. Pro případné budoucí aplikace a úpravy MARASTu nebo jiných výukových portálů lze zmínit několik dalších článků. [10] se věnuje analýze a hodnocení obtížnosti logických úloh. [11] je o způsobech detekce momentu, kdy se student danou látku skutečně naučil a ovládá ji. Z [12] usoudit, že dát studentům možnost upravit si obtížnost kvízů podle svého uvážení nepřináší velké zlepšení výkonnosti studentů. U některých dokonce dojde ke zhoršení, protože si berou schválně lehčí zadání.

Na Fakultě informačních technologií ČVUT se oblasti EDM věnovalo a stále věnuje hned několik studentů. V minulosti vzniklo i několik diplomových prací, např. Magda Friedjungová [13] každý rok predikuje studijní výsledky studentů 1. ročníku FIT po prvním semestru studia. Prediktivní model je sestaven na základě studentových výsledků v předmětech během semestru a na socio-demografických datech z přihlášky ke studiu. Eliška Hrubá v diplomové práci [14] prováděla analýzu studijních výsledků absolventů středních škol. Ondřej Nový [15] na základě studijních výsledků sestavil model, který studentům doporučuje vhodné volitelné předměty k zápisu v následujícím semestru. Předměty jsou doporučovány tak, aby byly tematicky studentovi blízké a zároveň byl zohledňován studentův výkon, aby doporučený předmět byl pro studenta průchozí. Stanislav Kuznetsov [16] se ve svojí dizertační práci věnuje generování ontologií s dovednostmi studentů, které studenti získali na základě absolvování předmětů, a doporučování vhodných zadání semestrálních projektů od firemních partnerů.

1.2.3 Nástroje vytěžování znalostí z dat

Pro práci s daty, jejich zpracováním a vytěžováním znalostí z nich lze najít relativně velké množství aplikací. Například lze využít tabulkové procesory (*Microsoft Excel* [17], *LibreOffice Calc* [18]) nebo specializované nástroje (SAS [19], RapidMiner [20]).

Ovšem dle zadání má být vytvořen modul v Pythonu [21], který bude danou analýzu dat provádět automaticky. Pro Python existuje mnoho balíčků se zaměřením na vizualizaci dat, datovou analýzu, statistické metody a matematické operace. Proto jsem se rozhodl udělat i samotnou analýzu dat s pomocí Python balíčků. To by mělo usnadnit automatizaci procesu, přechod k vývoji modulu, samotný vývoj a údržbu modulu.

Přímo pro vědu a výzkum existují ekosystémy a komunity kolem *SciPy* [22] a *PyData* [23]. Celý tento systém zahrnuje několik balíčků, které lze využít pro dosažení cíle této práce. Např. *pandas* [24] pro práci s daty a datovou analýzou, *scikitlearn* [25] pro algoritmy strojového učení. Všechny tyto balíčky mají širokou podporu komunity jak z pohledu vývoje, tak z pohledu podpory uživatelů.

Kapitola 2

Práce s daty

Jak již bylo avizováno v předchozí kapitole, pro analýzu dat byla zvolena metodika CRISP-DM. Jedná se o iterativní metodiku a popis postupu každé iterace by byl zdlouhavý a opakoval by se. V kapitole je popsána pouze poslední iterace metodiky s výsledkem analýzy a důležité části.

2.1 Pochopení domény

Popisu domény již bylo věnováno několik předchozích kapitol. Zde je uvedeno shrnutí toho nejdůležitějšího:

- Data pochází z kvízů, ty se skládají z převážně matematických (logických) úloh.
- Cílem kvízů je otestovat znalosti studentů z daných předmětů.
- Kvízy vyplňují studenti FIT ČVUT, většina jich pochází z bakalářského programu. V datech je však zahrnut i zlomek odpovědí od vyučujících.
- Odpovědi jsou přes webové stránky.
- V datech jsou obsaženy zejména kvízy k procvičování látky, tj. nebodované. Menší část kvízů již byla použita při zkouškách, zápočtech či jiných bodovaných testech.

2.2 Pochopení dat

Export dat byl poskytnut ve dvou souborech formátu *.csv*. V prvním souboru *quiz.csv* jsou informace o kvízech, v souboru *data.csv* pak informace o odpovědích. V obou souborech jsou tabulky, první řádek obsahuje hlavičku s popisem sloupce v angličtině.

Význam jednotlivých položek v souboru *quiz.csv* následuje:

quiz_id Jednoznačný identifikátor kvízu.

quiz_name Název kvízu.

quiz_type Typ kvízu, více informací dále v textu.

quiz_semester Název (označení) semestru, do kterého tento kvíz patří. Standardní značení na FIT, například B151 pro zimní semestr akademického roku 2015/2016, B162 pro letní semestr akademického roku 2016/2017 apod.

number_of_questions Počet otázek, které se najednou ukážou uživateli při vyplňování kvízu.

quiz_goal Počet otázek, které musí uživatel zodpovědět, aby dosáhl cíle kvízu (a jeho případného uzavření).

course_code Kód kurzu (předmětu), pro který je kvíz určen. Standardní značení předmětů na FIT, tj. například BI-ZMA pro bakalářský předmět Základy matematické analýzy, MI-MPI pro magisterský předmět Matematika pro informatiku apod.

open_at Časová značka, kdy byl kvíz otevřen. Časová zóna odpovídá České republice, tj. středoevropskému času (GMT+2, resp. GMT+1).

close_at Časová značka, kdy byl kvíz uzavřen a nemohl být dále odevzdán k vyhodnocení. Časová zóna odpovídá České republice, tj. středoevropskému času (GMT+2, resp. GMT+1).

first_penalty Číselná hodnota ve vteřinách. Určuje, jak dlouho nebude umožněno uživateli odpovídat po tom, co odpověděl špatně.

second_penalty Číselná hodnota ve vteřinách. Poté, co uživatel dosáhl série špatných odpovědí, jejíž délka je určena hodnotou *second_lock*, je uživateli zakázáno po dobu *second_penalty* odpovídat.

second_lock Číselná hodnota. Určuje délku série špatných odpovědí, po jejímž dosažení bude uživateli znemožněno odpovídat po dobu určenou hodnotou *second_penalty*.

available_problems Celkový počet otázek dostupných pro daný kvíz.

is_important Označení pro bodovaný kvíz.

is_tracked Příznak, které kvízy probíhaly během semestru. Jedná se o však o nově přidaný příznak a informace nemusí odpovídat realitě.

Datové typy v souboru *quiz.csv* a interpretace hodnot jsou v Tab. 2.1.
Sloupec *quiz_type* v souboru *quiz.csv* nabývá pouze následujících hodnot:

Tabulka 2.1: Tabulka s typy v souboru *quiz.csv*

Název sloupce dat	Datový typ	Poznámky
quiz_id	celé číslo	dáno vždy
quiz_name	řetězec znaků	dáno vždy
quiz_type	řetězec znaků	dáno vždy
quiz_semester	řetězec znaků	dáno vždy
number_of_questions	celé číslo	dáno vždy
quiz_goal	celé číslo	nepovinný (žádná hodnota)
course_code	řetězec znaků	dáno vždy
open_at	řetězec znaků	dáno vždy
close_at	řetězec znaků	nepovinný (prázdný řetězec)
first_penalty	celé číslo	dáno vždy
second_penalty	celé číslo	dáno vždy
second_lock	celé číslo	nepovinný (žádná hodnota)
available_problems	celé číslo	dáno vždy
is_important	pravd. hodnota	dáno vždy
is_tracked	pravd. hodnota	dáno vždy

- *progressive* – Uživatel odpovídá na otázky postupně. Otázky jsou náhodně voleny. Uživatel obvykle vidí (a řeší) pouze jednu jedinou otázku. Pokud je daný *quiz_goal* a uživatel dosáhne daného počtu správných odpovědí, má splněno, ale může v kvízu dále pokračovat.

- *prepared_test* – Uživatel odpovídá na předem připravenou sadu otázek, které se zobrazí najednou. Toto je typické pro bodované kvízy (zápočty, zkoušky).

- *random_test* – Uživatel odpovídá na otázky, které jsou náhodně voleny z nějaké sady otázek. Uživatel obvykle vidí více otázek najednou a řeší tak skupiny otázek.

Význam jednotlivých položek v souboru *data.csv*:

question_id Jednoznačný identifikátor tohoto záznamu, tj. dvojice otázka – odpověď.

user_id Jednoznačný identifikátor uživatele, který vytvořil tento záznam. Tedy ten, co odpověděl.

quiz_id Jednoznačný identifikátor kvízu, ke kterému tento záznam patří. Váže se na *quiz_id* z *quiz.csv*.

problem_id Jednoznačný identifikátor otázky (problému), kterou měl uživatel za úkol zodpovědět.

closed Značí, je-li otázka uzavřena (tj. zodpovězena, hodnota 1), nebo ne (hodnota 0).

kind Typ otázky. Informace dále v textu.

score Celkový počet možností v odpovědi, které byly uživatelem správně zvoleny (ať už správně zaškrtnuté, nebo správně nezaškrtnuté).

correct_options Počet možností v odpovědi, které mají být označeny, a byly správně označeny.

correct Značí, je-li otázka správně zodpovězena (hodnota 1), nebo ne (hodnota 0).

created_at Časová značka, kdy přesně byl tento záznam vytvořen. Prakticky značí moment, kdy se uživateli otevřela tato otázka k zodpovězení. Časová zóna odpovídá České republice, tj. středoevropskému času (GMT+2, resp. GMT+1).

answered_at Časová značka, kdy přesně byla otázka zodpovězena. Tj. moment, kdy se uživatel rozhodl vyhodnotit svou odpověď. Časová zóna odpovídá České republice, tj. středoevropskému času (GMT+2, resp. GMT+1).

duration Doba trvání odpovídání na otázku. Rozdíl *answered_at* a *created_at*.

to_deadline Doba mezi zodpovězením otázky (*answered_at*) a uzavřením příslušného kvízu (*close_at*) z *quiz.csv*.

Datové typy v souboru *data.csv* a interpretace hodnot jsou v Tab. 2.2.
Sloupec *kind* v souboru *data.csv* nabývá pouze následujících hodnot:

- *multichoice* – Uživatel má za úkol zvolit správné možnosti z nabízeného výběru. V datech jsou pouze otázky, u nichž jsou vždy celkem 4 možnosti k zatržení. Tedy, maximum ve sloupci *score* je 4 (vše správně), minimum 0 (vše špatně).

- *text_field* – Uživatel musí správně zadat hodnotu do textového políčka. U tohoto typu otázky nabývá sloupec *correct_options* vždy hodnoty 1. Sloupec *score* pak odpovídá hodnotou sloupci *correct*. Co přesně uživatel zadal není v datech poskytnuto.

K datům celkově je vhodné zdůraznit několik faktů, které nemusí být na první pohled zřejmé:

Tabulka 2.2: Tabulka s typy v souboru *data.csv*

Název sloupce dat	Datový typ	Poznámky
question_id	celé číslo	dáno vždy
quiz_id	celé číslo	dáno vždy
problem_id	celé číslo	dáno vždy
closed	celé číslo	dáno vždy
kind	řetězec znaků	dáno vždy
score	celé číslo	dáno vždy
correct_options	celé číslo	dáno vždy
correct	celé číslo	dáno vždy
created_at	řetězec znaků	dáno vždy
answered_at	řetězec znaků	není vždy (prázdný řetězec)
duration	desetinné číslo, řetězec znaků	dáno vždy, řetězec "na"
to_deadline	desetinné číslo, řetězec znaků	dáno vždy, řetězec "na"

- Sloupec *duration* může nabývat i záporných hodnot. Jedná se o rozdíl *answered_at* a *close_at*, přičemž *close_at* může být u některých uživatelů menší než *answered_at*. To je speciální případ, kdy řešitel nestihl odevzdat kvíz a byl za něj uzavřen vyučujícím po termínu.

- S předchozím bodem souvisí i sloupeček *to_deadline*. Ten také může nabývat záporných hodnot, ze stejného důvodu jako sloupec *duration*. Tj. hodnota *answered_at* může být větší než hodnota *close_at*.

- Sloupec *answered_at* nedává vždy informaci o tom, kdy ve skutečnosti uživatel zodpověděl danou otázku. Toto se vyskytuje u kvízů, ve kterých uživatel vidí a odpovídá na více otázek najednou a jsou odeslány k vyhodnocení společně. Celá tato skupina otázek pak má (téměř) stejnou hodnotu *answered_at*. V některých případech se hodnota doby odpovědi u otázek z jedné skupiny může lišit v řádu jednotek vteřin, to je způsobené pravděpodobně zpracováváním dat systémem.

- V datech není přímo dostupná informace, která otázka způsobila kolikáté zablokování. Tuto informaci je nutné dopočítávat.

- Uživatel může u kvízů typu *progressive* pokračovat v kvízu i po dosažení počtu správných odpovědí rovných *quiz_goal*.

2.3 Cíle analýzy

Cílem analýzy je prozkoumat poskynutá data, provést jejich analýzu a pokusit se z nich vyčíst nové poznatky.

Vyučující například zajímá, jak studenti ke kvízům přistupují. Z mnoha otázek, které se nabízejí, lze zmínit:

- Jak dlouho v daném kvízu trvá v průměru odpovědět na otázku?
- Kdy se obvykle odpovídá?
- Jaký vztah má odpovídání k blížícímu se konci kvízu?
- Jaký efekt má zablokování na další odpověď? Na její správnost, na její dobu odpovídání?
- Mají studenti tendenci opakovat své chyby? Tj. již správně zodpovězenou otázku odpoví poté špatně?

Pro dosažení cíle se provede základní a pokročilejší statistika, pomocné vizualizace, aplikace shlukování či detekování odlehlých hodnot.

2.4 Příprava a předzpracování dat

V sekci je popis procesu přípravy a zpracování dat za účelem zvýšení kvality dat.

2.4.1 Čištění dat

Nedílnou součástí je příprava dat. Účelem je data předzpracovat tak, abychom se zbavili chybných záznamů. Prvotní analýza dat byla provedena manuálně s použitím vizualizací, jednoduchých statistik kvízů apod. Na jejím základě se data čistila.

Dle poskytovatele dat se při databázové migraci v srpnu 2016 vyskytla chyba, kdy se hodnota sloupečku *answered_at* pro všechny tehdejší záznamy nastavila na 18.8.2016. Z těchto dat se sice dá vyčíst úspěšnost odpovědí, ale nelze určit doby trvání odpovídání. Po poradě s vedoucím a poskytovatelem dat bylo rozhodnuto, že se tyto záznamy nebudou brát v potaz a lze je smazat.

Co se semestrů týče, byly smazány semestry s kódem B000 a BTEST. Kvízy v těchto semestrech mají velmi málo řešitelů a odpovědí, i z názvů těchto semestrů a jím přiřazených kvízů lze usoudit, že jsou určené k testování funkcionality MARASTu, ne lidí.

Podobný problém je u některých kvízů přiřazených regulérním semestrům, nejen B000 a BTEST. Z názvů těchto kvízů je však patrné, že sloužily opět jen k testování funkcionality, nikoliv lidí.

Další aspekt vhodný k předzpracování je délka doby odpovědi na otázku. Velká skupina otázek je zodpovězena za relativně dlouhou dobu, řádově dny až týdny, v několika případech dokonce měsíce. Uživatel se k rozpracované otázce vrátil klidně po měsíci a otázku správně zodpověděl. Uživatel pravděpodobně neřešil otázku tak dlouho, zřejmě to jen otevřel ze zvědavosti a pak se k tomu po měsíci vrátil. Tyto odpovědi nepatří úplně do kontextu a navíc vysoká doba odpovídání narušuje další statistiky. Proto byly všechny odpovědi s dobou odpovědi vyšší než 2 týdny (14 dní) vymazány z dat.

Některé otázky jsou nezodpovězeny po mnoho týdnů. Tyto otázky byly ponechány v datech, neboť mohou nést informaci o obtížných otázkách. U této skupiny je obtížné říci, jestli je člověk jen otevřel a již se k otázce nevrátil a nevrátí, nebo otevřel a ještě se k ní vrátí, nebo otázku stále řeší.

Cílem analýzy jsou studenti a v datech je několik záznamů i od vyučujících, případě záznamy sloužící k testování. Tyto záznamy nelze bezpečně určit. Nicméně s ohledem na odhadovaný počet (stovky záznamů oproti statisícům od studentů) by měl být jejich vliv na výsledky minimální.

Cílem předchozích úprav bylo zbavit se všech dat, které byla poškozena, nesloužila k testování uživatelů nebo byla jinak kontextově mimo.

2.4.2 Prázdné hodnoty ve sloupcích

Při práci s daty je potřeba dávat pozor na prázdné hodnoty, známé také jako *null*, *NaN (Not-a-Number)*, *nil* apod. S těmito hodnotami mají algoritmy většinou problémy, neví, co s nimi mají dělat. Operace nad touto hodnotou může způsobit pád běhu algoritmu, nebo dokonce celého programu.

V *quiz.csv* se mohou prázdné hodnoty vyskytovat v následujících sloupcích: *quiz_goal*, *close_at*, *second_lock*. Pro praktické účely budou nedefinované hodnoty se sloupci *quiz_goal* nahrazeny 0, interpretovatelné způsobem, že kvíz nemá stanovený cíl. Hodnota 0 je přesně definována a detekovatelná.

Stejně tak nedefinované hodnoty ve sloupci *second_lock* budou nahrazeny nulou. Opět je 0 přesně definována a detekována, přičemž význam je stejný – kvíz nemá druhý zámek.

Hodnotu ve sloupci *close_at* nelze smysluplně nahradit a bude s tím potřeba počítat při manipulaci s daty.

V *data.csv* se prázdné hodnoty mohou vyskytovat ve sloupcích *answered_at*, *duration* a *to_deadline*. V tomto případě je nutné počítat s nedefinovanými hodnotami, nelze je smysluplně nahradit.

2.4.3 Pomocné atributy

Některé informace nejsou v datech poskytnuty explicitně a je nutné je dopočítat. V této části jsou pomocné atributy popsány i se způsobem jejich výpočtu.

Následuje seznam pomocných atributů pro odpovědi se stručným popisem jejich výpočtu:

incorrect Inverze sloupečku *correct*. Tedy pokud je *correct* roven 1, *incorrect* bude roven 0. Pokud je *correct* roven 0, *incorrect* bude 1.

incorrect_cumsum Kumulativní suma nad sloupečkem *incorrect*. Ke zjištění pořadí špatných odpovědí. Vyžaduje data seřazená podle *answered_at* vzestupně.

correct_cumsum Kumulativní suma nad sloupečkem *correct*. Ke zjištění pořadí správných odpovědí. Vyžaduje data seřazená podle *answered_at* vzestupně.

which_trial Kolikáty pokus o vyplnění kvízu a dosažení *quiz_goal*. Pouze pro kvízy se stanoveným *quiz_goal*. Výpočet na základě hodnoty *correct*, *correct_cumsum* a jejich porovnání vůči hodnotě *quiz_goal*.

which_lock Kolikáté zablokování daná odpověď způsobila. Inicializována pomocí hodnoty *incorrect_cumsum*, poté vynulována u správných odpovědí (nemohou způsobit zablokování). Poté se podle hodnoty *which_lock* (pořadí špatných odpovědí), *fist_lock* (kdy dojde k první blokaci) a *second_lock* (kdy dojde k druhé blokaci) vypočítá kolikátý zámek daná špatná odpověď způsobila. Vyžaduje data seřazená podle *answered_at* vzestupně.

previous_lock Kolikátý zámek způsobila předchozí odpověď. Výpočet probíhá posunutím sloupečku *which_lock* o jednu hodnotu napřed, tj. hodnota *previous_lock* odpovídá hodnotě *which_lock* předchozí odpovědi. První odpověď nemá definováno, neboť nemá předcházející odpověď, pro praktické účely je však tato hodnota nastavena na celočíselnou 0 (neexistující odpověď nezpůsobí blokaci). Vyžaduje data seřazená podle *answered_at* vzestupně.

pause_from_previous_answer Pauza mezi zodpovězením předchozí odpovědi a začátkem řešení nového příkladu. Výpočet je přes rozdíl sloupce *created_at* a *answered_at*, který je navíc posunutý o jednu hodnotu napřed. Tj. jedná se o rozdíl hodnoty *created_at* a hodnoty *answered_at* předchozí odpovědi. Vyžaduje data seřazená podle *answered_at* vzestupně. U kvízů, kde se zobrazuje více problémů najednou a odpovědi se tak posílají po skupinách, může být hodnota *pause_from_previous_answer* záporná. To je z důvodu stejného času *created_at* pro všechna řešení (otázky a odpovědi) z dané skupiny, protože byly společně vytvořeny. Totéž platí pro hodnoty *answered_at* z dané skupiny, které byly společně odeslány k vyhodnocení. Tento případ je řešen nahrazením záporných hodnot nulou, což odpovídá logice v dodaných datech.

pause_without_lock_time Pauza mezi zodpovězením předchozí odpovědi a začátkem řešení nového příkladu, ale bez času, kdy nebylo možné přejít

na další problém kvůli případnému zablokování kvůli špatné odpovědi. Výpočet je na základě hodnoty *pause_from_previous_answer* a hodnot souvisejících s blokováním, tj. *previous_lock*, *first_penalty*, *second_lock* a *second_penalty*.

2.5 Modelování dat

Vedle základních statistik se na data aplikují algoritmy patřící do skupiny shlukové analýzy. Ta patří mezi algoritmy strojového učení bez učitele (ang. *unsupervised learning*) [26], neboť pro vstupní data není znám výstup. Slouží k seskupování objektů dle jejich vlastností tak, aby si byly objekty v jedné skupině co možná nejpodobnější v porovnání s ostatními skupinami. Z hlediska domény se tak mohou například seskupit studenti dle jejich studijních výsledků, nebo zadané otázky podle doby odpovídání.

Algoritmy shlukování se v základu dělí na dvě skupiny. První skupina, tzv. nehierarchické shlukování, vytváří z dat systém disjunktních podmnožin. Druhou skupinu pak tvoří hierarchické shlukování, které vytváří z dat systém podmnožin, kde průnik dvou podmnožin je buď prázdná množina, nebo jedna z podmnožin. Tím se vytváří hierarchie shluků.

Ke shlukové analýze je možné využít několik algoritmů, jen v balíčku *scikit-learn* jich je k dispozici přes 10 [27]. Vhodné použití algoritmů závisí na datech a co od shlukování očekáváme. Výběr algoritmů je také proveden s přihlédnutím k rešerši a literatuře.

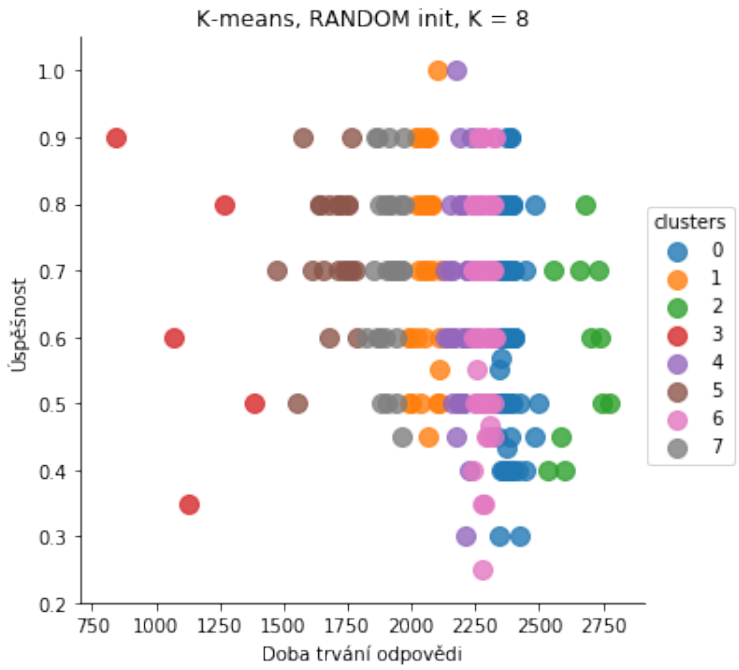
Nad daty se dále nabízí aplikovat algoritmy detekce anomálií a odlehlých hodnot (ang. *Anomally and Outlier detection*). Tímto způsobem lze najít data, která vybočují z běžných hodnot. Z hlediska domény se tak mohou najít například studenti s výjimečně dobrými nebo špatnými výsledky, velmi lehké nebo velmi těžké otázky.

Využití dalších odvětví strojového učení jako je klasifikace, regrese (odhady) nebo redukce dimenzionality nemá příliš smysl. Klasifikace vyžaduje popsaná data, ovšem naše data popsána nejsou. Regrese není cílem analýzy. Redukce dimenzionality by se teoreticky dala využít, ale naše data nejsou vysoce dimenzionální. Redukce dimenzionality se většinou aplikuje při desítkách či stovkách atributů, kdy významnost jednotlivých není zřejmá, a tak lze redukcí získat méně atributů s vyšším informačních přínosem.

2.5.1 Výběr dimenzí pro shlukování

Pro modelování dat je důležité vybrat relevantní dimenze (vlastnosti) dat. Jinak řečeno, nad kterými sloupečky v tabulce budou probíhat výpočty pro shlukování a detekci anomálií.

Není vhodné spouštět shlukování či detekci anomálií nad všemi dimenzemi. Taková volba by znamenala vysokou výpočetní náročnost, někdy označovanou



Obrázek 2.1: *k-means* shlukování nad uživateli, progresivní kvíz z BI-ZMA, nepřeškálovaná data

jako kombinatorická exploze. Některé sloupce ani nejsou vhodné pro modelování — identifikátory kvízů, uživatelů či otázek apod.

Podle rešerše (viz podsekce 1.2.2) jsou často sledovanými parametry doba odpovídání (sloupec *duration*) v kombinaci s průměrem správností odpovídání (průměr sloupce *correct* pro určitá data). V případě uživatelů tato kombinace parametrů reprezentuje jejich znalosti, např. v určitém kvízu, předmětu či celkově. V případě otázek pak může reprezentovat jejich obtížnost. Tyto dva parametry budou zkoumány.

Vybrané dimenze je obvyklé přeškálovat do stejného intervalu. Bez škálování by mohlo dojít k tomu, že při výpočtu vzdáleností bude mít jedna nebo více dimenzí převahu a rozdělení mezi shluky tak nebude vypočítáno rovnoměrně. K přeškálování lze využít několik možností. Běžný je *MinMaxScaler* [28] (přeškáluje data do intervalu mezi daným minimem a maximem) a v případě předpokladu normálního rozdělení dat lze použít *StandardScaler* [29]. Efekt nepřeškálovaných dat je vidět na Obr. 2.1, kdy má doba trvání odpovědí jasnou převahu nad úspěšností a shluky jsou tvořeny podle doby trvání odpovědi. Na Obr. 2.2 je vidět výsledek shlukování se stejným nastavením, ovšem s přeškálovanými daty.

Obrázek 2.2: *k-means* shlukování nad uživateli, progresivní kvíz z BI-ZMA, přeškálovaná data

2.5.2 Způsob vyhodnocování shlukování

Pro vyhodnocení kvality shlukování bude v prvé řadě sloužit vizualizace a manuální (lidské) posouzení. Interpretace výsledků člověkem je v našem případě velmi důležitá, chceme z dat získat nové použitelné znalosti. S ohledem na pozdější automatizaci analýzy přihlédneme i k jiným metodám hodnocení kvality shlukování, a to pomocí metrik. U výběru metrik jsme však omezeni tím, že pro vytvořené shluky neznáme správný výstup (tzv. *ground truth*).

Nabízí se použít skóre siluety [30] (ang. *Silhouette score*, *Silhouette Coefficient*). Silueta představuje poměr podobnosti a odlišnosti jednoho shluku od ostatních a výsledné skóre siluety je průměr hodnot skóre siluety všech shluků. Jinak řečeno, hodnota vyjadřuje, jak moc jsou si shluky blízko, případně jak moc se překrývají. Skóre siluety vychází v uzavřeném intervalu mezi -1 a 1. Hodnota -1 značí špatné shlukování (špatně přiřazené shluky), hodnota 0 překrývající se shluky a hodnota 1 kvalitní shlukování ve smyslu dobře oddělených shluků.

Další možností je využít skóre *Calinski-Harabasz* [31, 32] (také jako *Calinski-Harabasz index*, *CH index*), které opět nevyžaduje znalost správného výstupu. Toto skóre je definováno jako poměr rozptylu uvnitř shluků vůči rozptylu mezi shluky. Vyšší hodnota skóre značí lépe definované shluky, tj. vyšší skóre zna-

mená husté, dobře oddělené shluky. Z hodnoty siluety lze obdržet relativně více informací. Toto skóre slouží spíš k porovnávání výsledků několika modelů, kdy vyšší skóre znamená lepší (kvalitnější) shlukování. Poznámka pro ujasnění, v modulu *scikit-learn* je toto skóre pojmenováno jako *Calinski-Harabaz*, s chybějícím *s* ve jméně *Harabasz*.

2.5.3 Vybrané algoritmy shlukování

Podle přehledu aplikovaných algoritmů v EDM [6] je jednou z nejpoužívanějších metod shlukování *k-means* [27, 33, 34] z nehierarchických algoritmů, z hierarchických algoritmů pak aglomerativní shlukování (ang. *Agglomerative*) [27, 35]. Oba zmíněné algoritmy jsou již v balíčku *scikit-learn* a s ohledem na reference budou aplikovány.

Algoritmus *k-means* je podle [6] nejpoužívanějším algoritmem v EDM. Vstupem algoritmu *k-means* jsou data a hodnota k, která určuje výsledné množství shluků. Každý shluk má svůj centroid, který lze interpretovat jako reprezentant shluku. Při každé iteraci algoritmu se minimalizuje vzdálenost mezi daty a centroidy. Metrika vzdálenosti se většinou používá euklidovská [36, 37], je možné však použít i jiné. Počáteční umístění centroidů může zvolit uživatel, může se nechat na náhodě, nebo lze využít algoritmu *k-means++* [38], který volí centroidy s cílem optimalizace (zrychlení) běhu celého algoritmu.

S ohledem na relativně velké množství dat v dodaném exportu (kolem 400 000 záznamů po pročištění) lze uvažovat i variantu *k-means* zvanou *Mini-batch k-means* [39]. Tato varianta je přímo upravena pro velké množství dat, při každé iteraci výpočtu se pracuje s náhodně zvolenou podmnožinou dat, nikoliv se všemi jako v případě základního algoritmu. *Mini-batch k-means* má v porovnání se základním *k-means* nižší výpočetní nároky, nižší výpočetní čas a o něco méně přesné výsledky.

Aglomerativní shlukování je podle [6] nejpoužívanější hierarchický shlukovací algoritmus v EDM. Aglomerativní shlukování funguje na principu sjednocování množin (shluků) prvků do větších podle několika měřítek. Těmito měřítky jsou podobnosti, resp. vzdálenosti dvou prvků, prvku a shluku a dvou shluků. Pracuje s asociační maticí, která obsahuje tyto míry podobnosti, resp. vzdálenosti. Spojování shluků probíhá u nejpodobnějších z nich (těch s nejmenší vzdáleností).

Dle literatury se v EDM využívají i další varianty *k-means*, jako je *Fuzzy k-means*, *C-means*, *k-prototypes* aj. Tyto nejsou v balíčku *scikit-learn* dostupné a jejich využití je podle [6] v řádu jednotek. Tyto varianty jsou například méně striktní než *k-means* v přiřazování prvků do shluků, prvek má místo jasně daného shluku míru příslušnosti k několika (nebo všem) shlukům.

Modul *scikit-learn* nabízí i jiné shlukovací algoritmy [27], jako například *Mean Shift*, *Affinity Propagation*, *Birch* aj. Tyto algoritmy shlukování byly také otestovány na datech, ovšem jejich výsledky byly podobné nebo horší než u *k-means* a aglomerativního shlukování, žádné viditelné zlepšení nebo lépe

interpretovatelné výsledky nebyly nalezeny. Z tohoto důvodu, a s ohledem na literaturu, ve které se nejčastěji využívají algoritmy *k-means* a aglomerativní shlukování, bylo tyto dva algoritmy zvoleny pro aplikaci. Výsledky a ukázky běhů shlukovacích algoritmů jsou dostupné na přiloženém CD ve složce s *Jupyter notebooky*, ukázky výsledků jsou pak v sekci 2.6.8. Algoritmus *Mean Shift* má tendenci vytvářet velké množství shluků a je problém s interpretací výsledků. Výsledky algoritmů *Birch* a *Affinity Propagation* vrací výsledky podobné algoritmům *k-means* a aglomerativnímu shlukování.

Konečný výběr tedy padl na shlukování použitím *k-means* a aglomerativního shlukování. *k-means* je základní algoritmus v EDM a poskytuje relativně dobré výsledky, algomerativní shlukování lze díky mnoha parametrům uzpůsobovat a výsledné shluky lze dobře interpretovat. Z ostatních zmíněných měl dobré výsledky algoritmy *Birch* a *Affinity Propagation*, ostatní algoritmy poskytovaly horší výsledky.

Při aplikování shlukovacích algoritmů bylo dále zjištěno, že stávající vyčištění dat není pro aplikaci těchto algoritmů dostačující. Jak je vidět na Obr. 2.3, data obsahují odlehlé hodnoty (řádově jednotky) a pro ty jsou vytvářeny samostatné shluky. Algoritmus sice funguje správně, nicméně shlukování nad daty bez anomálií by mohlo přinést zajímavé informace a lepší možnost porovnání běžných dat. Tento problém byl vyřešen striktnějším pročištěním dat použitých pro shlukování a detekci anomálií.

2.5.4 Určování optimálního počtu shluků

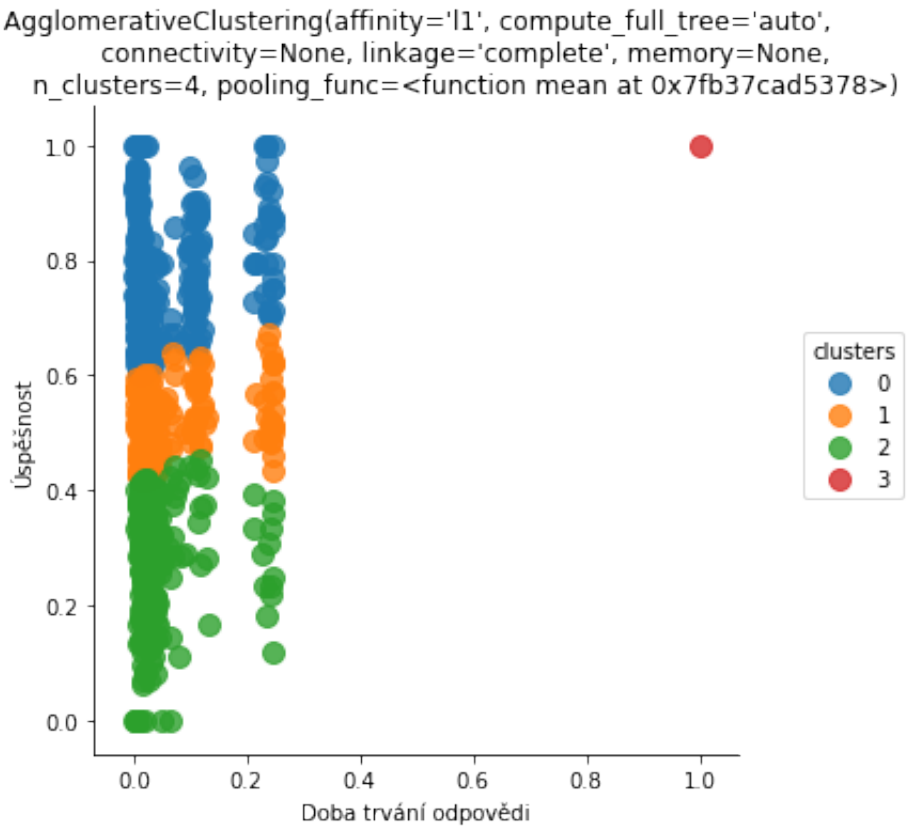
Algoritmy shlukování často vyžadují, aby uživatel nastavil počet shluků. Určení optimálního počtu shluků (obecně nazývané jako parametr *k*, případně *n_clusters*) je výpočetně náročné, jedná se o NP-těžký problém [40, 41]. V algoritmech shlukování existují výjimky, u kterých počet shluků vypočítává algoritmus nebo je určen jinak automaticky, jako např. *Mean Shift* [42].

Při určování počtu shluků je nutné brát v potaz interpretovatelnost výsledku. Příliš málo shluků obvykle nedá žádnou novou informaci, v příliš mnoha shlucích je obtížné se orientovat a rozdíly mezi shluky nemusí být příliš znatelné.

Máme určitou doménu a některé skupiny můžeme předpokládat. Budeme-li například dělit uživatele podle jejich průměrné úspěšnosti a mediánu doby odpovídání, můžeme předpokládat existenci skupin reprezentující studenty s následujícími charakteristikami:

- Studenti odpovídají správně a rychle. Tato skupina pravděpodobně látce rozumí.

- Studenti odpovídají správně a pomalu. Studenti látce rozumí, byť nad tím musí uvažovat delší dobu, resp. počítají dlouho.

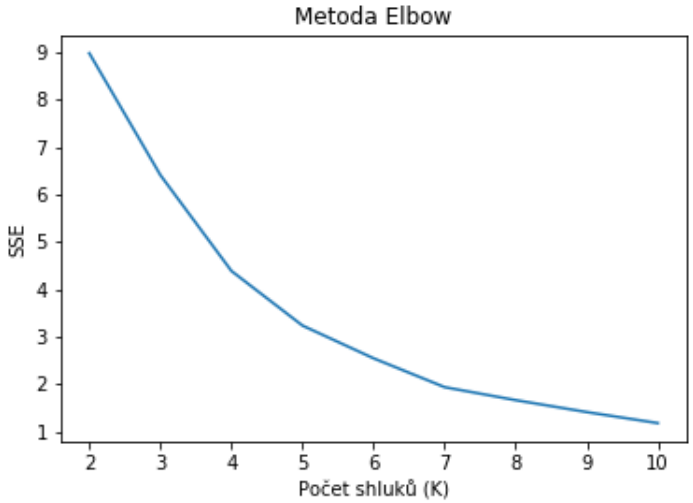


Obrázek 2.3: Aglomerativní shlukování nad otázkami, progresivní kvíz z BI-ZMA (Procvičování: Limita a spojitost funkce), výsledek ovlivněn anomálií (červený shluk)

- Studenti odpovídají špatně a rychle. Studenti, kteří mohou pouze hádat odpovědi, studenti s přehnaným sebevědomím atd.
- Studenti odpovídají špatně a pomalu. Tito se možná snaží nad problémem přemýšlet, přesto jim však látka nejde.

Tyto předpoklady lze dále rozvádět, například uvažovat jednu skupinu navíc, který by obsahovala studenty s celkově průměrnými výsledky, a kolem této skupiny by se vyskytovaly skupiny s výše uvedeným rozdělením.

Ke stanovení optimálního k u *k-means* byla zvolena tzv. *Elbow* metoda [43]. Jedná se o běh *k-means* s různými nastaveními k, pro každý model se stanoveným k se vypočítá součet druhých mocnin vzdáleností prvků od svých centroidů (*SSE*, nebo-li *Sum of Squared Errors*). To se vynese do grafu a dle tvaru křivky se stanoví optimální k. Optimální k je v bodě zlomu křivky,



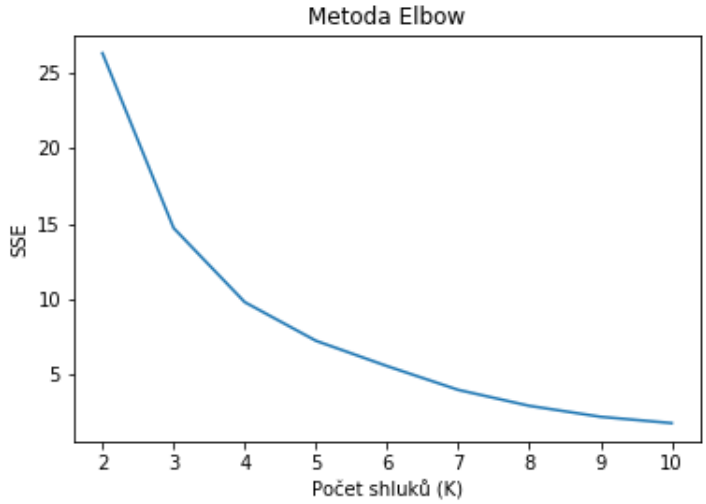
Obrázek 2.4: *Elbow* metoda pro algoritmus *k-means*, progresivní kvíz z BIZMA

tzn. vlevo od zlomu je křivka strmě klesající, vpravo od zlomu viditelně méně klesající.

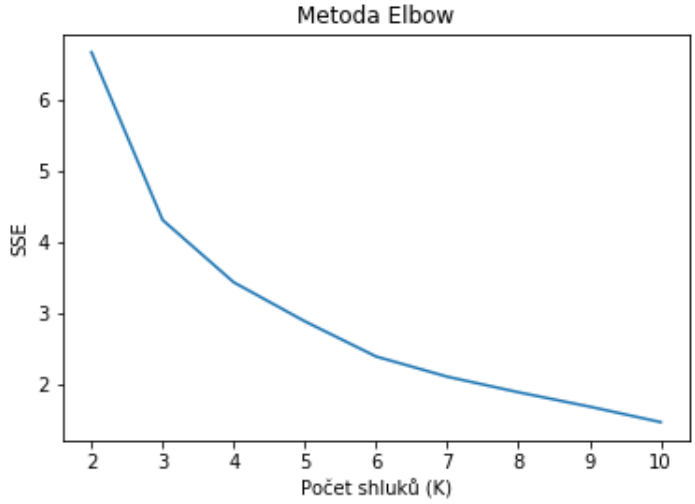
Obr. 2.4 je z dat z jednoho kvízu typu *progressive*, Obr. 2.5 pochází ze všech dat kvízů typu *progressive* a Obr. 2.6 je ze všech kvízů typu *prepared_test*. Z těchto obrázků lze usoudit, že optimální k je mezi 3 a 4, byť na Obr. 2.4 není optimální k zřetelné.

Pro aglomerativní shlukování vychází od pohledu nejlépe hodnota k mezi 4 a 7. Při tomto množství shluků je výsledek dobře interpretovatelný. Ukázky výsledků pro aglomerativní shlukování jsou na Obr. 2.7 a Obr. 2.8, další výsledky jsou dostupné na přiloženém CD. Z testování aglomerativního shlukování vyplývá, že metrika eukleidovská, l2 a manhattanská mají nejlépe interpretovatelné výsledky, ostatní metriky nedosahují tak dobrých výsledků.

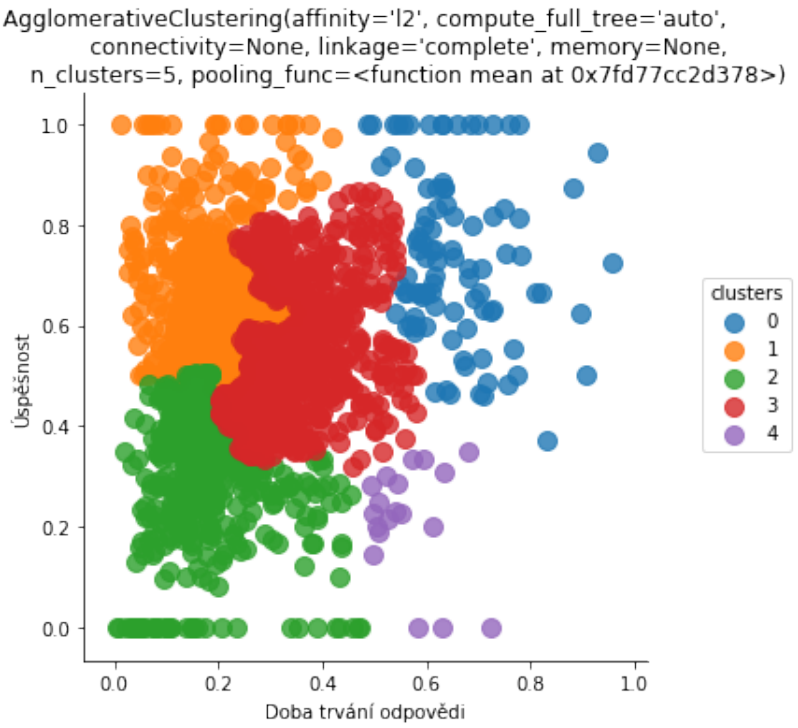
Použití dříve zmíněných metrik skóre siluety a skóre *Calinski-Harabasz* nám může pomoci maximálně v posouzení kvality pokrytí a nelze se spoléhat na to, že by vyšší hodnota jedné z metrik znamenala přehlednější výsledek. To lze demonstrovat na příkladu shlukování uživatelů (řešitelů) kvízu, kteří jsou charakterizováni úspěšností odpovědí a dobou odpovídání. Na Obr. 2.9 je vidět model s nejvyšší hodnotou skóre siluety, kdy bylo aglomerativní shlukování aplikováno na progresivní kvíz. Na stejných datech, ovšem s nejvyšším skóre *Calinski-Harabasz*, je model na Obr. 2.10. Mnohem lépe interpretovatelné modely jsou přitom vidět dříve v této podsekci, na Obr. 2.7 a Obr. 2.8.



Obrázek 2.5: *Elbow* metoda pro algoritmus *k-means*, všechny progresivní kvízy



Obrázek 2.6: *Elbow* metoda pro algoritmus *k-means*, všechny kvízy typu *prepared_test*



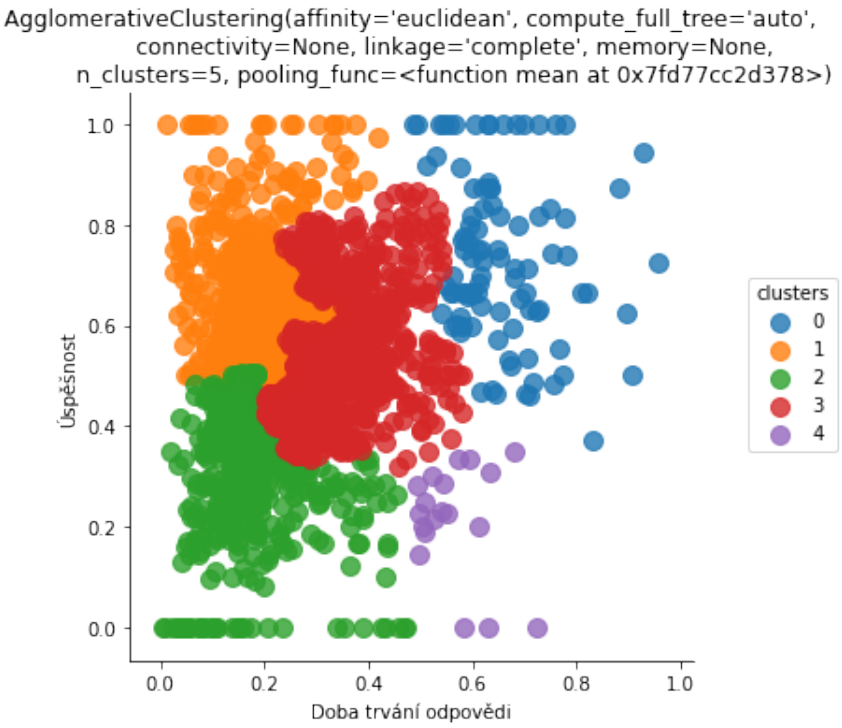
Obrázek 2.7: Aglomerativní shlukování nad uživateli, progresivní kvíz z BI-ZMA (Procvičování: Limita a spojitost funkce), metrika l2

2.5.5 Detekce anomálií a odhlehlých hodnot

Balíček *scikitlearn* nabízí několik způsobů hledání anomálií a odlehlých hodnot [44]. Algoritmy přistupují k hledání anomálií různě, společným parametrem je pro ně kontaminace, pomocí které lze určit relativní očekávané množství anomálií v datech.

Prvním algoritmem pro určování odlehlých hodnot a anomálií je *Elliptical Envelope* [45]. Algoritmus vychází z toho, že pro některá vstupní data lze předpokládat jejich rozdělení. *Elliptical Envelope* vychází z předpokladu, že data mají Gaussovské rozdělení. Vypočítá se střed elipsy, data se obalí elipsou (obálkou) a vše mimo obálku je anomálie.

Další technikou je *Isolation Forest* [46], ten se hodí zejména pro data s mnoha dimenzemi. Jeho fungování je založeno hodně na náhodném generátoru, s čímž souvisí i název techniky. Algoritmus vezme prvek, náhodně vybere jednu jeho dimenzi a náhodně zvolí dělící hranici mezi minimem a maximem dané dimenze, podle které umístí prvek do stromu. V případě náhodného dělení se pak anomálie vyznačují tím, že mají mnohem kratší cestu od kořene stromu.

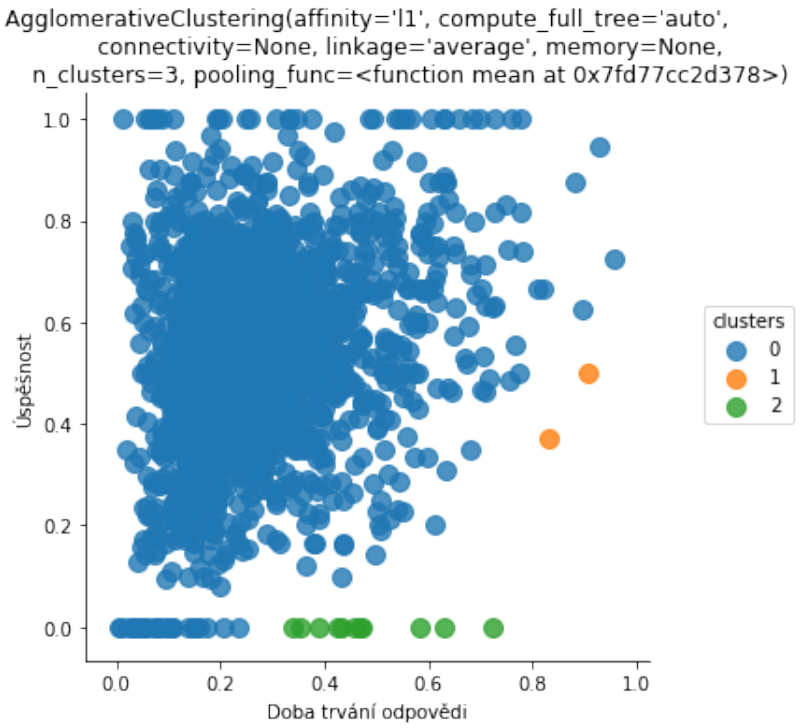


Obrázek 2.8: Aglomerativní shlukování nad uživateli, progresivní kvíz z BI-ZMA (Procvičování: Limita a spojitost funkce), eukleidovská metrika

Poslední technikou určenou k detekci anomálií je *Local Outlier Factor* (LOF) [47]. Je založen na lokální odchylce hustoty daného prvku vzhledem k sousedům prvku, kdy anomálie mají nižší hustotu (méně prvků v okolí) než běžné prvky. Jedná se o relativně komplexní algoritmus s mnoha parametry. Lze nastavit počet sousedů, de kterým se při výpočtu hustoty přihlíží. Dále algoritmus výpočtu nejbližších sousedů, kde lze zvolit stromy, metodu hrubou silou, případně automatické určení nejlepšího možného algoritmu. V případě volby metody využívající stromy lze zvolit velikost listu. Dále lze zvolit metriku výpočtu vzdálenost mezi dvěma prvky.

Pro úplnost lze zmínit i další, méně časté metody hledání anomálií. Lze za tímto účelem použít i algoritmy shlukování [48], ačkoliv to není obvyklé. Nad daty se provede shlukování s nastaveným vyšším počtem shluků, stanoví se hranice minimální velikosti jednoho shluku a všechny shluky, které mají méně prvků než daná hranice, jsou považovány za shluky odlehlých hodnot. Ve zkoumání jsou i algoritmy, které jsou přímo specializované na shlukování a detekování anomálií [49], přičemž algoritmus nebere při výpočtu shlukování nalezené odlehlé hodnoty v potaz. Tyto algoritmy nebudou zkoušeny.

Z výsledků testů detekce anomálií (dostupné v podsekci 2.6.9 a na přilo-



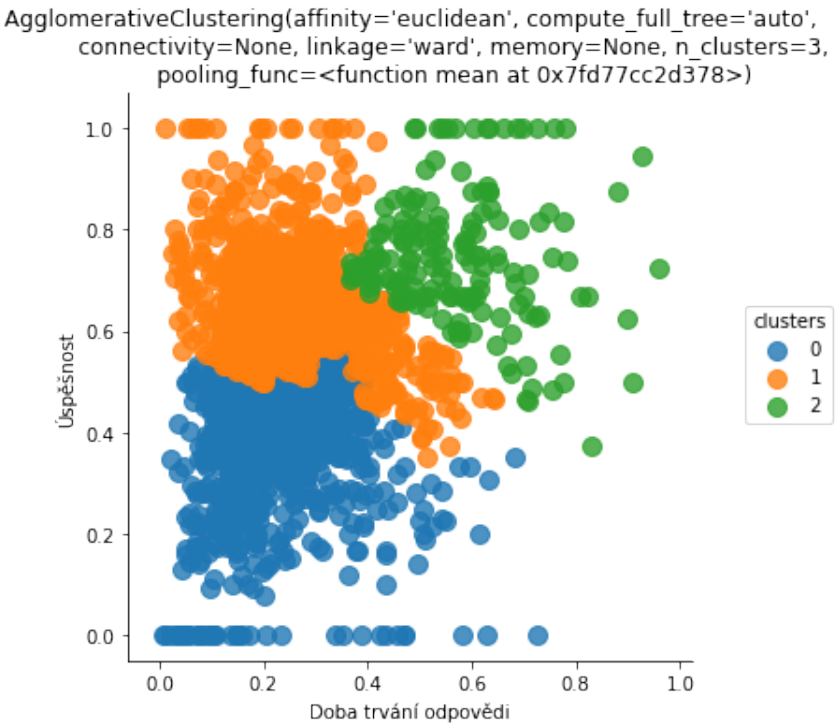
Obrázek 2.9: Aglomerativní shlukování nad uživateli, progresivní kvíz z BI-ZMA (Procvičování: Limita a spojitost funkce), nejvyšší skóre siluety

ženém CD) lze usoudit následující:

- Algoritmy detekují jako anomálie stejné prvky.
- U *Elliptical Envelope* je problém s nastavením velikosti elipsy.
- *Isolation Forest* je při hledání anomálií konzervativní, anomálie jsou většinou mimo hlavní shluky dat a jsou to skutečně okrajové hodnoty.
- *Local Outlier Factor* nalézá nejvíce anomálií, a to i uvnitř shluků dat, z lokálního hlediska.
- Kontaminace je důležitý parametr – nalezne přesně tolik anomálií, na jakou je nastaven. To vyžaduje mít předpoklad o počtu anomálií v datech před aplikací jejich detekce.

2.6 Celkové vyhodnocení analýzy

Sekce obsahuje výsledky manuální analýzy dat.



Obrázek 2.10: Aglomerativní shlukování nad uživateli, progresivní kvíz z BI-ZMA (Procvičování: Limita a spojitost funkce), nejvyšší skóre *Calinski-Harabasz*

Manuální analýza byla vypracována ve formátu tzv. *Jupyter notebook* (přípona souboru .ipynb) [50], podrobný popis použitých nástrojů je v další části práce v podsekci 3.3.7. S ohledem na rozsah analýzy jsou zde uvedeny pouze vybrané otázky a jejich odpovědi. Úplná analýza je dostupná na přiloženém CD.

Nasazením modelu se zabývá další kapitola.

2.6.1 Obecné informace

MARAST používá celkem 7 předmětů, jejichž seznam následuje:

- BI-AG1 (Algoritmy a grafy 1)
- BI-LIN (Lineární algebra)
- BI-PKM (Přípravný kurz matematiky)
- BI-PST (Pravděpodobnost a statistika)

Tabulka 2.3: Informace o předmětech využívajících MARAST

Název kurzu	Povinný	Doporučený semestr	Počet kreditů
BI-AG1	Ano	3	6
BI-LIN	Ano	2	7
BI-PKM	Ne	(není)	4
BI-PST	Ano	5	5
BI-ZDM	Ano	3	5
BI-ZMA	Ano	1	6
MI-MPI	Ano	1	7

Tabulka 2.4: Využití MARASTu v jednotlivých předmětech dle počtu kvízů

Kód předmětu	Počet kvízů
BI-PKM	27
BI-ZMA	24
BI-AG1	17
BI-LIN	14
BI-PST	4
BI-ZDM	3
MI-MPI	2

- BI-ZDM (Základy diskrétní matematiky)
- BI-ZMA (Základy matematické analýzy)
- MI-MPI (Matematika pro informatiku)

Už z názvů předmětů je zřejmé, že MARAST používají předměty založené na matematice a logickém uvažování.

Z Tab. 2.3 je vidět, že 6 ze 7 předmětů je povinných, výjimku tvoří kurz s názvem Přípravný kurz matematiky, který slouží k připomenutí středoškolské látky potřebné pro absolvování matematických předmětů na FIT. Dále, pouze jediný předmět (MI-MPI) není bakalářský.

Využití MARASTu v jednotlivých předmětech je vidět v Tab. 2.4. Z tabulky je vidět, že je hojně využívám v předmětech BI-PKM a BI-ZMA.

Dále je možné se podívat na kvízy, které mají nejvíce odpovědí v rámci předmětu, viz Tab. 2.5. Z tabulky je vidět, že s výjimkou BI-ZDM a BI-PKM jsou nejoblíbenější kvízy ty, které souvisí se zápočtem nebo zkouškou.

Kvízy v datech jsou přiřazeny do následujících semestrů:

- B151 (Letní semestr akademického roku 2015/2016), od 1.10.2015 do 21.2.2016. Zkouškové je od 11.1. do 20.2.
- B152 (Zimní semestr akademického reko 2015/2016), od 22.2.2016 do 30.9.2016. Zkouškové je od 23.5. do 2.7.

Tabulka 2.5: Kvízy s nejvyšším počtem odpovědí podle předmětu

Kód kurzu	Název kvízu	Celkem řešitelů	Celkem odpovědí
BI-AG1	Selftest AG1 - Příprava na zkoušku	287	34 520
BI-LIN	Zápočtový kvíz (část A)	513	18 867
BI-PKM	Matematická logika	586	13 517
BI-PST	2. zápočtový kvíz BIK-PST	16	365
BI-ZDM	Týdenní kvíz č. 1	382	4636
BI-ZMA	2. zápočtový kvíz	686	19 881
MI-MPI	zápočtový kvíz č. 1	224	4 129

Tabulka 2.6: Využití MARASTu v předmětech podle semestrů

Semestr	Kurzy
B151	BI-ZMA
B161	BI-PKM, BI-AG1, BI-ZMA, MI-MPI, BI-PST
B162	BI-LIN
B171	BI-ZDM, BI-PKM, BI-AG1

- B161 (Zimní semestr akademického roku 2016/2017), od 1.10.2016 do 19.2.2017. Zkouškové je od 9.1. do 17.2.
- B162 (Letní semestr akademického roku 2016/2017), od 20.2.2017 do 30.9.2017. Zkouškové je od 22.5. do 30.6.
- B171 (Zimní semestr akademického roku 2017/2018), od 1.10.2017 do 18.2.2018. Zkouškové je od 8.1. do 18.2.

Ze semestru B151 jsou v datech přítomny kvízy pro kurz Základy matematické analýzy. Tyto kvízy nemají stanovený čas uzavření kvízu a je možné je stále vyplnit. Pro ostatní kvízy čas otevření a uzavření odpovídá času vymezenému přiřazeným semestrem.

Využití MARASTu v jednotlivých předmětech a semestrech je vidět v Tab. 2.6. Z tabulky je vidět, že většina kvízů využívajících MARAST je v zimním semestru (kód semestru končící číslicí 1).

Co se týče průchodnosti studentů v daných předmětech (jinak řečeno úmrtnost), to lze vidět v Tab. 2.7.

Jak moc se MARAST využíval z hlediska celkového počtu odpovědí je pak vidět v Tab. 2.8. Z tabulky je zřejmé, že využití MARASTu na fakultě stoupá, uživatelů je víc a počet odpovědí také vzrůstá. Je potřeba vzít v potaz, že data jsou dostupná přibližně do poloviny semestru B171 (poslední záznam v datech

Tabulka 2.7: Průchodnost předmětů v daném semestru. BI-PKM a BI-ZMA má některé kvízy otevřené bez data uzavření. Data k semestru B171 nejsou kompletní, průchodnost je uvedena jen pro zajímavost. Data o průchodnosti pochází z Ankety ČVUT [51].

Semestr	Kurz	Průchodnost [%]
B151	BI-ZMA	40
B161	BI-AG1	59
	BI-PKM	(není)
	BI-PST	79
	BI-ZMA	42
	MI-MPI	65
B162	BI-LIN	43
B171	BI-AG1	59
	BI-PKM	(není)
	BI-ZDM	61
	BI-ZMA	37

Tabulka 2.8: Statistiky počtu odpovědí a kvízů podle semestru, seřazeno dle celkového počtu odpovědí

Semestr	Počet odpovědí	Počet uživatelů	Počet kvízů	Průměrný počet odpovědí v semestru na kvíz a uživatele
B161	195133	1417	40	3.442713
B171	110892	1115	14	7.103908
B162	68131	531	7	18.329567
B151	58	23	7	0.360248

je z 11.11.2017), statistiky za celý semestr B171 by jistě překonaly statistiky za semestr B161.

Celkový počet řešitelů je 2 577, ti se účastnili celkem 70 kvízů a řešili nebo ještě řeší celkem 391 107 problémů.

Data před pročištěním měla přibližně 524 000 odpovědí, předzpracování a odstranění extrémních hodnot tedy vzalo přes 25% dat. Chyba v databázové migraci ovlivnila data za první rok používání MARASTu, přičemž za další rok a čtvrt se nahromadilo téměř 400 000 odpovědí.

2.6.2 Základní statistiky pro kurzy - úspěšnosti, doby odpovídání aj.

Základní statistikou pro předměty je úspěšnost odpovědí. Tu lze najít v Tab. 2.9. Z tabulky je patrné, že nejvyšší úspěšnost má předmět BI-AG1 s 60,25%,

Tabulka 2.9: Úspěšnost odpovídání dle kurzu (seřazeno abecedně dle kódu kurzu)

Kód kurzu	Celkem odpovědí	Úspěšnost odpovědí [%]
BI-AG1	53692	60,25
BI-LIN	68131	57,20
BI-PKM	172877	54,01
BI-PST	703	43,39
BI-ZDM	7909	30,79
BI-ZMA	63159	39,60
MI-MPI	7727	47,55

Tabulka 2.10: Doby odpovídání (DO) podle kurzu. Časy jsou ve formátu hodina:minuta:sekunda.

Kód kurzu	Průměr DO	Medián DO	Nejkratší DO	Nejdelší DO
BI-AG1	07:50:18	00:15:47	00:00:03	13 dní 22:51:59
BI-LIN	02:12:18	00:01:27	00:00:03	13 dní 21:12:43
BI-PKM	03:14:35	00:02:51	00:00:02	13 dní 23:59:35
BI-PST	02:40:27	00:03:58	00:00:14	8 dní 16:01:22
BI-ZDM	04:55:43	00:03:33	00:00:04	10 dní 05:39:52
BI-ZMA	02:05:49	00:03:18	00:00:03	13 dní 22:14:22
MI-MPI	03:06:51	00:05:54	00:00:04	12 dní 06:09:12

hned za ním je BI-LIN s 57,20%. Na druhé straně je BI-ZDM s 30,79% a BI-ZMA s 39,60%.

Statistiky o době odpovídání jsou k dispozici v Tab. 2.10. Z této tabulky lze usoudit, že doba odpovídání neodpovídá době řešení. Těžko si lze představit, že by někdo trávil řešením jedné otázky téměř 2 týdny v kuse, je pravděpodobnější, že otázku otevřel a po delší době se k ní vrátil. Stejně tak lze vyvodit, že průměr doby odpovídání není nejvhodnější ukazatel doby odpovídání. Díky existenci extrémů je i průměr značně vysoký, na rozdíl od mediánu, který ukazuje reálnější hodnoty. Hodnoty mediánu by se již daly považovat i za doby řešení, byť nemusí dostatečně zohledňovat méně časté případy, trvající delší dobu.

Další zajímavou statistikou je rozdíl v odpovědích mezi semestrem a zkouškovým. Při rozdělení odpovědí podle času, kdy byla otázka zodpovězena, jsou výsledky úspěšnosti v Tab. 2.11 a doby odpovídání v Tab. 2.12.

Příznak *is_tracked* zatím nebylo doporučeno používat k rozlišení kvízů semestrálních a zkouškových, výsledky dostupné v Tab. 2.13 a Tab. 2.14 se od výsledků využívajících rozlišení přes datum o něco liší. Nicméně výsledky mohou pomoci i k lepšímu nastavování příznaku *is_tracked*.

Na Obr. 2.11 je znázorněno, kdy studenti obvykle odpovídají na otázky.

Tabulka 2.11: Úspěšnost odpovědí, rozdělení dle data odpovědi na semestr a zkouškové

Semestr	Zodpovězeno v	Celkem odpovědí	Úspěšnost odpovědí [%]
B151	semestr	42	26.19
B161	semestr	151605	47.70
	zkouškové	43528	64.74
B162	semestr	29823	45.50
	zkouškové	38308	66.31
B171	semestr	110892	51.07

Tabulka 2.12: Doba odpovídání (DO), rozdělení dle data odpovědi na semestr a zkouškové, čas ve formátu hodiny:minuty:sekundy

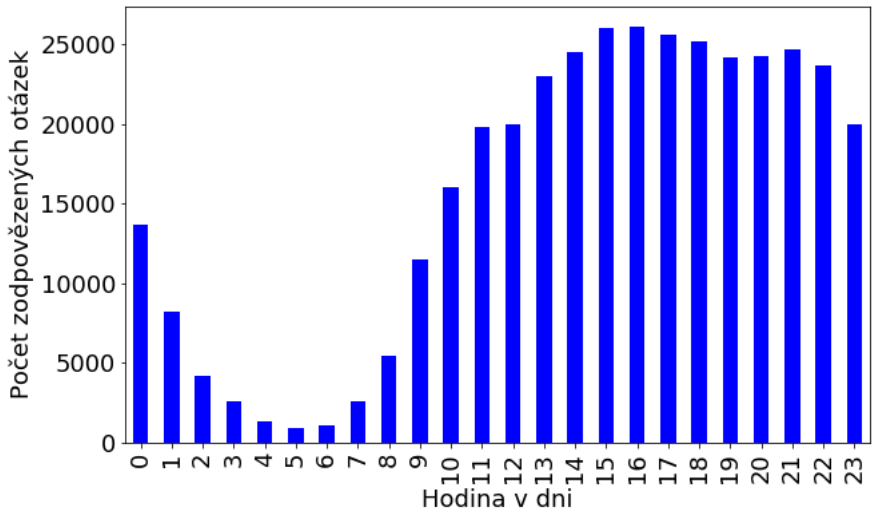
Semestr	Zodpovězeno v	Průměr DO	Medián DO
B151	semestr	06:05:16	00:00:40
B161	semestr	03:37:09	00:03:23
	zkouškové	05:58:55	00:15:17
B162	semestr	03:03:35	00:02:17
	zkouškové	01:32:22	00:00:57
B171	semestr	03:19:57	00:02:54

Tabulka 2.13: Úspěšnost odpovědí, rozdělení dle *is_tracked*

Semestr	Zodpovězeno v	Celkem odpovědí	Úspěšnost odpovědí [%]
B151	semestr	42	26.19
B161	semestr	175797	52.92
	zkouškové	19336	38.63
B162	semestr	40574	66.18
	zkouškové	27557	43.97
B171	semestr	110892	51.07

Tabulka 2.14: Doba odpovídání (DO), rozdělení dle *is_tracked*, čas ve formátu hodiny:minuty:sekundy

Semestr	Rozlišení dle is_tracked	Průměr DO	Medián DO
B151	semestr	06:05:16	00:00:40
B161	semestr	04:12:50	00:04:39
	zkouškové	03:31:52	00:03:35
B162	semestr	01:28:32	00:00:57
	zkouškové	03:16:44	00:02:25
B171	semestr	03:19:57	00:02:54



Obrázek 2.11: Celkové množství odpovědí podle hodiny ve dni

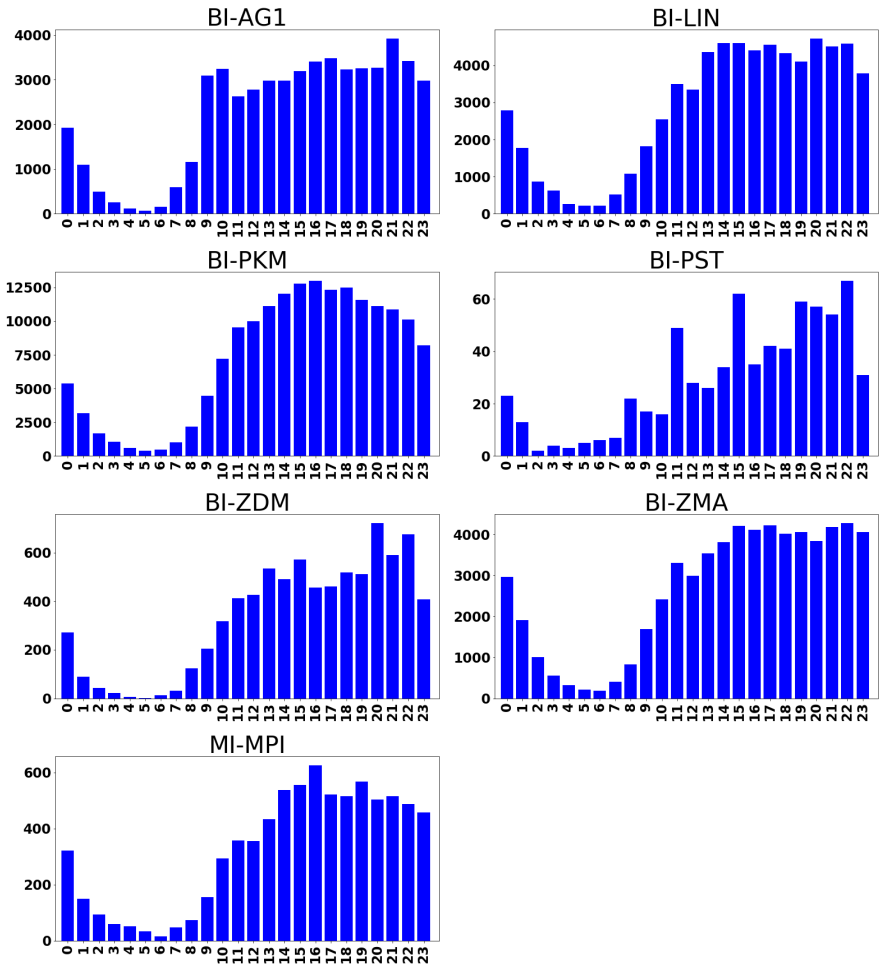
Z obrázku je patrné, že největší aktivita je mezi 14. a 23. hodinou, což je pravděpodobně hlavní čas, kdy se studenti učí. Nejmenší aktivita je pak kolem 5. hodiny ráno. Podobné rozdělení mají všechny předměty, jak je vidět na Obr. 2.12.

Podíváme-li se na uživatele s celkově nejnižší průměrnou dobou odpovědi, zjistíme, že první místa obsazují řešitelé s nepříliš vysokou úspěšností a s malým počtem otázek. Tito uživatelé se pravděpodobně jen chtěli podívat na pár příkladů a zkusit si MARAST. Proto byla statistika omezena na uživatele s alespoň 10 odpověďmi, kterých bylo 1 980. V Tab. 2.15 je zobrazeno 10 uživatelů s nejnižší průměrnou dobou odpovídání, kteří měli alespoň deset odpovědí.

Deset uživatelů s nejvyšší průměrnou dobou odpovídání je pak v Tab. 2.16. Z této tabulky je také vidět velký rozdíl mezi průměrem a mediánem doby odpovídání. Kvůli tomuto rozdílu je lepší držet se spíš mediánu, který je méně náchylný na extrémní hodnoty.

2.6.3 Statistiky pro kvízy

U jednotlivých kvízů se můžeme podívat na úspěšnost. V Tab. 2.17 a Tab. 2.18 je seznam kvízů s nejvyšší, resp. nejnižší úspěšností v každém předmětu.



Obrázek 2.12: Celkové množství odpovědí podle kurzu a hodiny ve dni. Vodorovná osa značí hodinu ve dni, svislá osa počet odpovědí

Tabulka 2.15: Deset nejrychleji odpovídajících uživatelů s více než 10 odpověďmi

ID uživatele	Odpovědi		Doba odpovídání [H:M:S]	
	Správně	Celkem	Průměr	Medián
143020	0	22	00:00:35	00:00:30
142625	0	18	00:00:54	00:00:21
140419	22	26	00:00:57	00:00:43
142905	6	12	00:01:15	00:01:07
143159	89	687	00:01:33	00:00:25
139872	19	101	00:01:36	00:00:32
140021	14	16	00:01:36	00:01:25
139490	18	24	00:01:52	00:01:33
142562	99	164	00:02:05	00:01:19
140820	20	42	00:02:08	00:01:26

Tabulka 2.16: Deset nejpomaleji odpovídajících uživatelů s více než 10 odpovědmi

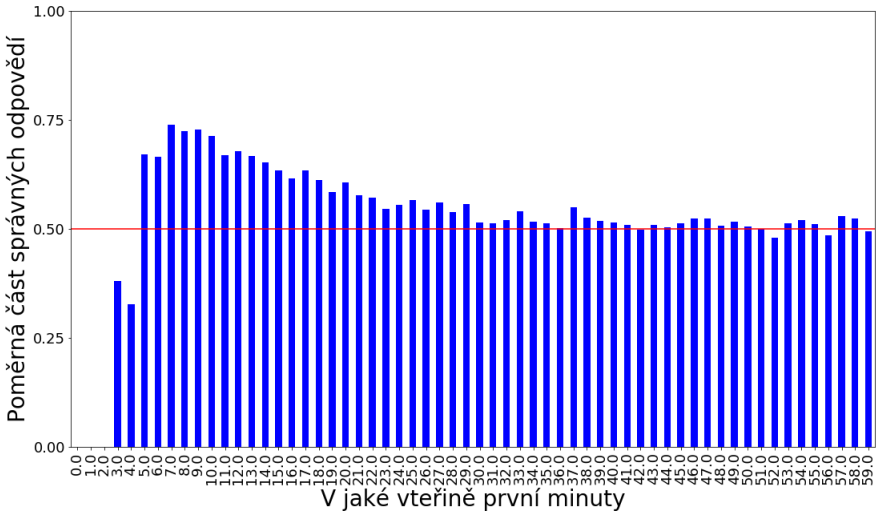
ID uživatele	Odpovědi		Doba odpovídání [H:M:S]	
	Správně	Celkem	Průměr	Medián
140418	54	94	1 den 12:42:14	00:07:47
139908	5	11	1 den 06:17:01	00:01:23
141082	35	64	1 den 05:30:12	00:25:55
143029	25	69	1 den 05:10:48	00:01:30
140302	38	60	1 den 03:51:20	00:36:52
140788	169	226	1 den 03:50:45	04:51:27
141112	8	11	1 den 02:52:42	00:31:49
142445	0	11	1 den 02:13:33	00:01:12
139609	118	180	1 den 00:58:25	00:24:48
139462	80	97	1 den 00:14:52	00:07:41

Tabulka 2.17: Kvízy s největší úspěšností

Kód kurzu	Název kvízu	Úspěšnost [%]
BI-ZMA	Časté chyby	75.42
BI-LIN	Trénovací kvíz 5: determinant, vlastní čísla, skalární součin	73.57
BI-AG1	Zkouška 17. 1. 2017	73.45
BI-PKM	Exponenciální funkce a logaritmus	73.36
MI-MPI	zápočtový kvíz č. 1	47.66
BI-PST	1. zápočtový kvíz BIK-PST	45.56
BI-ZDM	Týdenní kvíz č. 2	36.95

Tabulka 2.18: Kvízy s nejnižší úspěšností

Kód kurzu	Název kvízu	Úspěšnost [%]
BI-ZMA	Derivace funkce	0.00
BI-ZDM	Týdenní kvíz č. 1	27.28
BI-AG1	Semestral Quiz No. 1 - BIE AG1	36.06
BI-PKM	Matematická logika	40.77
BI-PST	2. zápočtový kvíz BIK-PST	41.36
BI-LIN	Zápočtový kvíz (část B)	43.68
MI-MPI	zápočtový kvíz č. 2	47.41

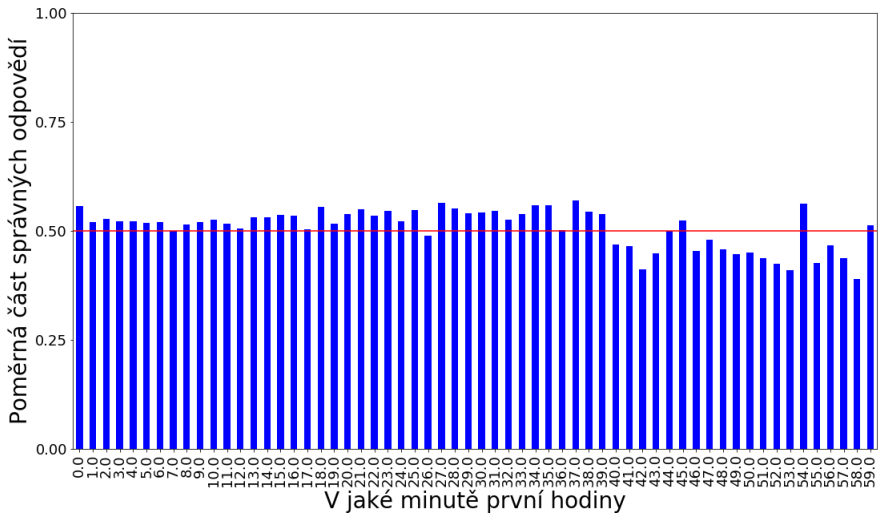


Obrázek 2.13: Úspěšnost odpovědí do první minuty

2.6.4 Vliv rychlosti odpovídání na úspěšnost, celá data

Z Obr. 2.13 lze usoudit, že s ohledem na nízkou úspěšnost a dobu trvání se do 5 vteřin spíš odhaduje řešení problémů. Mezi 5. a 7. vteřinou se pravděpodobně začínají vyskytovat první, kteří látce opravdu rozumí. Mezi 7. a 11. vteřinou pak lidé, kteří látku ovládají svědomitě (rychlá odpověď a vysoká úspěšnost). Od 11. vteřiny jsou pravděpodobně studenti, kteří potřebují větší množství času na rozmyšlenou. Přičemž se zdá, že čím déle je potřeba času, tím horší výsledek.

Na Obr. 2.14, Obr. 2.15, Obr. 2.16 a Obr. 2.17 je pak vidět, jak se úspěšnost odpovědí pohybuje, když je zodpovězeno do první hodiny, prvního dne, prvního týdne a pak dvou týdnů, což je maximum v pročištěných datech.



Obrázek 2.14: Úspěšnost odpovědí do první hodiny

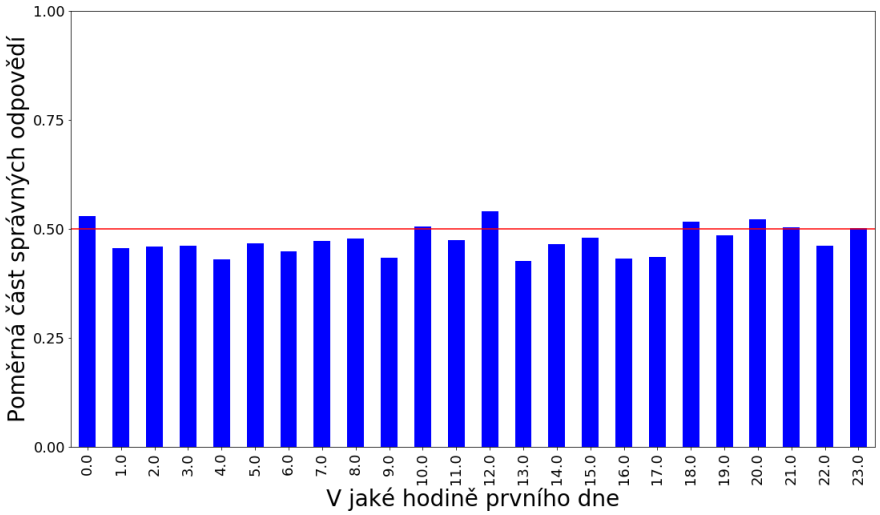
Za zajímavé lze považovat to, že se úspěšnost pro odpovědi trvající déle než 1 hodiny pohybuje v průměru kolem 50%. A to jak v rámci odpovědi do jednoho dne, i v rámci odpovědi do jednoho týdne a dokonce i odpovědi do 2 týdnů.

2.6.5 Práce studentů o zimních prázdninách

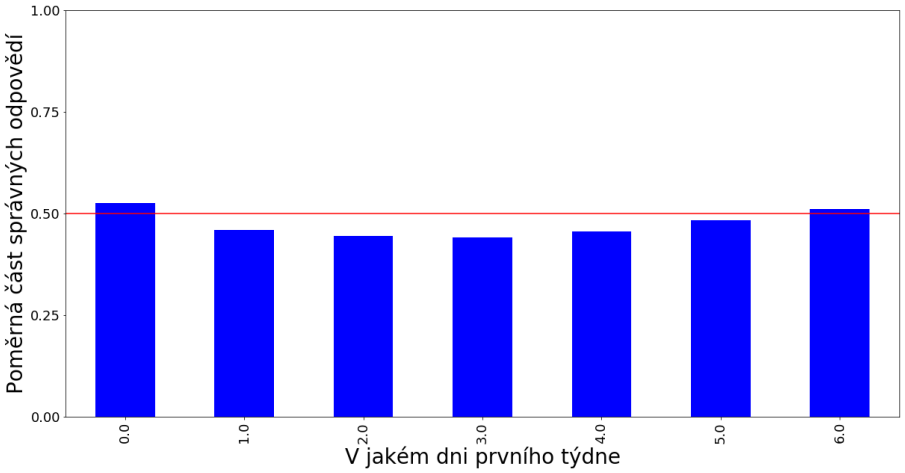
Zimní prázdniny jsou pro studenty časem dohánění učiva, úkolů, projektů a také příprava na poslední zápočtové testy, opravy zápočtových testů a na zkouškové období. Nabízí se tedy otázka, jak moc o zimních prázdninách studenti pracovali. Ze zimních prázdnin semestru B161 je relativně velké množství odpovědí, z ostatních prázdnin není příliš mnoho dat a hlavní je načasování zimních prázdnin těsně před zkouškové.

Kolik odpovědí bylo v každý den zimních prázdnin odesláno je vidět v Tab. 2.19. Jak je vidět, nejméně aktivní jsou hlavní dny prázdnin — Štědrý den, Silvestr a Nový rok.

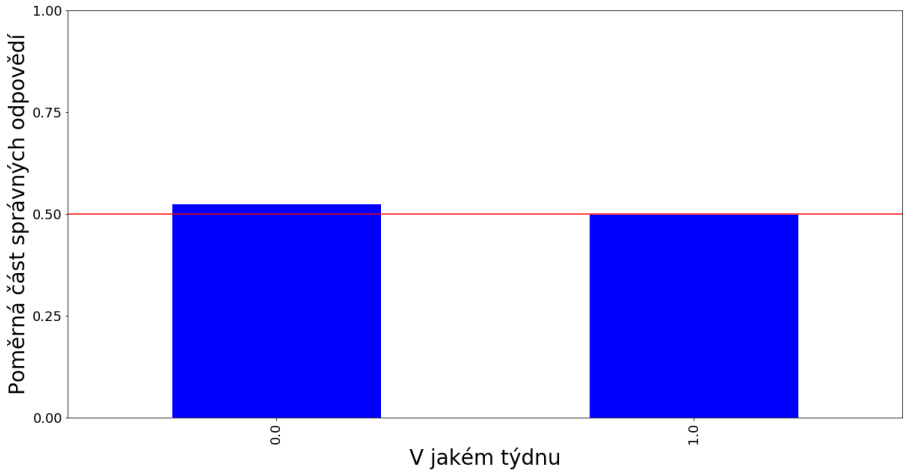
Na Štědrý den (24.12.2016) bylo otevřeno 12 kvízů, 127 uživatelů mělo otevřenou alespoň jednu otázku. Celkem bylo otevřeno (zobrazeno) 682 otázek. Tato hodnota je různá od hodnoty v tabulce, protože se jedná o celkový počet otázek, které byly v tento den buď zodpovězeny, nebo bylo jejich řešení započato, ale mohly být zodpovězeny v jiný den. Z nich bylo 228 zodpovězeno správně a 454 špatně, což dává úspěšnost odpovědí 33,43%. Co se týče doby



Obrázek 2.15: Úspěšnost odpovědí v prvním dni



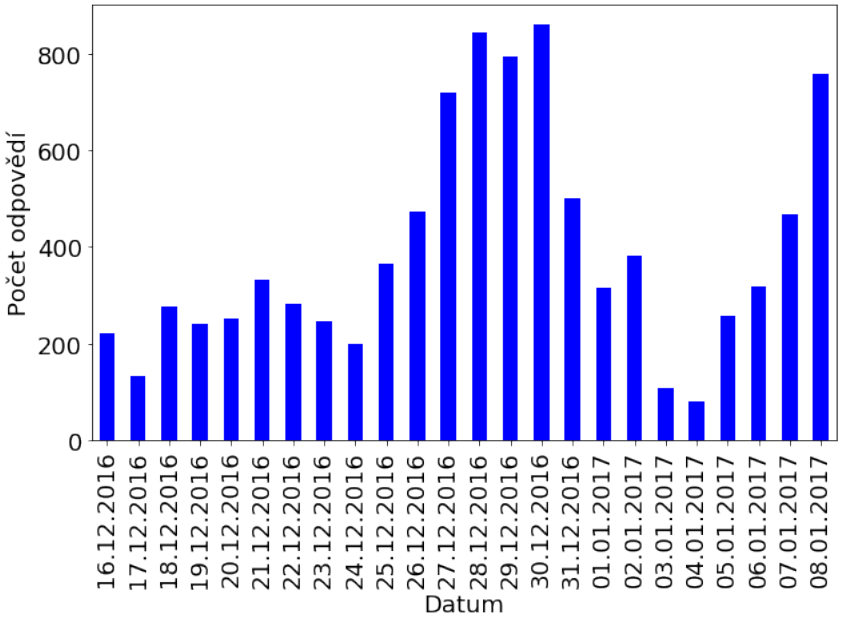
Obrázek 2.16: Úspěšnost odpovědí v prvním týdnu



Obrázek 2.17: Úspěšnost odpovědí během 2 týdnů

Tabulka 2.19: Počet odpovědí během zimních prázdnin semestru B161, po dnech

Den	Počet odpovědí v daném dni
23.12.2016	638
24.12.2016	610
25.12.2016	900
26.12.2016	1303
27.12.2016	1735
28.12.2016	2022
29.12.2016	1870
30.12.2016	2078
31.12.2016	1209
01.01.2017	495



Obrázek 2.18: Počet odpovědí ke konci vyučování v semestru B161

odpovídání, průměr je 5 hodin 40 minut a 47 sekund, kdežto medián je pouze 3 minuty 37 sekund.

Během Silvestra (31.12.2016) bylo aktivních 12 kvízů a 143 uživatelů mělo otevřenou alespoň jednu otázku. Celkem bylo otevřeno 1253 otázek (opět různá hodnota oproti tabulce), z nich bylo 506 zodpovězeno správně a 747 špatně, což dává úspěšnost odpovědí 40,38%. Průměrná doba odpovídání je 4 hodiny 27 minut a 46 sekund, medián je 3 minuty a 18 sekund.

Na Nový rok roku 2017 (1.1.2017) bylo stále 12 kvízů aktivních. Alespoň jednu otázku vidělo 81 uživatelů, ti viděli celkem 765 otázek, ale v porovnání s hodnotou v Tab. 2.19 jich v ten samý den moc nedořešili. Ze 765 otázek odpověděli 353 správně a 412 špatně, tedy úspěšnost 46,13%. Odpověď zabrala v průměru 7 hodin 59 minut a 36 sekund, medián je pak 6 minut 46 sekund.

Zajímavý je také obrázek z konce vyučování v zimním semestru, tedy před začátkem zkouškového. Z Obr. 2.18 je vidět, že před prázdninami se studenti příliš nepřipravovali, ale volno během prázdnin využili k učení. V posledním výukovém týdnu (od 2.1.) je zřejmý dvoudenní výpadek v přípravě ve dny 3.1 a 4.1., což mohlo být způsobeno výukou a posledními zápočtovými testy. Hned poté počet odpovědí prudce vzrostl, studenti se tedy nejspíš připravovali na zkoušky.

Tabulka 2.20: Počet blokací na předmět

Kód předmětu	Celkem odpovědí	Celkem správně	Celkem špatně (= počet blokací)
BI-PKM	172877	93367	79510
BI-ZMA	63175	25024	38151
BI-LIN	68131	38971	29160
BI-AG1	15872	7772	8100
BI-ZDM	7909	2435	5474
MI-MPI	7727	3674	4053
BI-PST	703	305	398

Tabulka 2.21: Kvíz s největším počtem blokací v daném předmětu

Kód předmětu	Název kvízu	Počet blokací
BI-ZMA	2. zápočtový kvíz	13513
BI-LIN	Zápočtový kvíz (část A)	10545
BI-PKM	Matematická logika	8006
BI-ZDM	Týdenní kvíz č. 1	3371
MI-MPI	zápočtový kvíz č. 1	2161
BI-AG1	Týdenní test č. 1	1726
BI-PST	2. zápočtový kvíz BIK-PST	214

2.6.6 Zablokování odpovídání

V datech je celkem 59 kvízů s nastaveným zablokováním uživatele po špatné odpovědi. Pro tyto kvízy je celkem 336 394 dostupných odpovědí. Kvízy mají nastavené buď jedno zablokování po každé špatné odpovědi, nebo obě zablokování (po jedné špatné odpovědi a po sérii špatných odpovědí). Průměrná úspěšnost této sady odpovědi je 50,99%, nelze se tedy divit, že celkem bylo 164 846 zablokování.

Z celkem 2070 řešitelů těchto kvízů jich bylo 2055 zablokováno alespoň jednou. 15 z nich nebylo zablokováno nikdy a celkem má těchto 15 řešitelů pouze 81 odpovědí, v porovnání s celkovým množstvím otázek (336 394) se jedná o zanedbatelné množství.

V Tab. 2.20 vidíme, že nejvíce blokací celkově je v přípravném předmětu BI-PKM.

Co se týče kvízů, ve kterých často dochází k zablokování, největší množství blokací v kvízu každého předmětu je vidět v Tab. 2.21. Z názvů kvízů lze usoudit, že nejvíc zablokování se vyskytuje v zápočtových kvízech.

Pro zajímavost se lze podívat na otázky (problémy), jejichž odpovědi způsobily nejvíce zablokování. Tato statistika je v Tab. 2.22 a údaje korespondují s tím, jak předměty využívají MARAST. Všech 10 otázek, které nejčastěji způsobily zablokování odpovídání, jsou z předmětu BI-PKM, který využívá

Tabulka 2.22: Deset otázek, které způsobily největší počet zablokování

ID otázky	Kód předmětu	Název kvízů	Počet zablokování
3153	BI-PKM	Matematická logika	1146
3387	BI-PKM	Kombinatorika	1120
3157	BI-PKM	Matematická logika	1112
3149	BI-PKM	Analytická geometrie	1102
3121	BI-PKM	Analytická geometrie	1097
3152	BI-PKM	Matematická logika	1008
3123	BI-PKM	Analytická geometrie	939
3174	BI-PKM	Matematická logika	899
3385	BI-PKM	Kombinatorika	894
4174	BI-PKM	Analytická geometrie	889

Tabulka 2.23: Průměrný počet zablokování v předmětu na kvíz a řešitele

Kód kurzu	Celkový počet odpovědí	Počet řešitelů	Počet kvízů	Průměrný počet zablokování na kvíz a řešitele
BI-PST	703	16	2	12,437500
MI-MPI	7727	224	2	9,046875
BI-LIN	68131	531	7	7,845036
BI-ZDM	7909	385	3	4,739394
BI-PKM	172877	1255	18	3,519699
BI-ZMA	63175	792	19	2,535287
BI-AG1	15872	703	8	1,440256

MARAST nejvíce ze všech předmětů.

Jiný pohled na zablokování nabízí Tab. 2.23. V ní je vypočtený průměrný počet zablokování na řešitele a kvíz v daném předmětu. Z toho je zřejmé, že nejvíce zablokování si užívají studenti předmětu BI-PST. BI-PKM, které má v absolutních číslech nejvíce odpovědí i zablokování, je pak až na 5. místě. Těží z velkého množství kvízů a řešitelů, relativní hodnota je pak výrazně nižší v porovnání s ostatními předměty.

Za prozkoumání stojí i vliv zablokování na následující odpověď. Vezmeme v potaz všechny odpovědi v kvízech, u kterých se najednou zobrazuje pouze jedna otázka a používají zablokování.

Podíváme-li se na pořadí zablokování bez uvažování doby, výsledky jsou následující. U takových kvízů mají odpovědi, kterým nepředcházelo zablokování, úspěšnost v průměru 58,58% a medián doby odpovídání 2 minuty a 28 sekund. Celkově pro odpovědi po jakémkoliv zablokování je úspěšnost 44,32% s mediánem doby odpovědi 2 minut a 46 vteřin. Po prvním zablokování je úspěšnost 46,90% s dobou odpovídání 2 minuty a 41 vteřin. Po druhém zablo-

kování je úspěšnost 41,57% a medián doby odpovídání 2 minuty a 42 sekund. Po třetím zablokování úspěšnost dosahuje 35,83% s mediánem odpovědí za 2 minuty 41 vteřin. Po čtvrtém je úspěšnost 46,07% s dobou odpovídání 1 minuta a 15 sekund. Po pátém 41,91% za 1 minutu 16 sekund.

Vezmeme-li však v potaz rozdílnou dobu prvního a druhého zámku, rozdíly jsou větší. Úspěšnost po první zámku je 45,44% s mediánem doby odpovědi 2 minuty a 49 vteřin. Po druhém (delším) zámku je to však pouze 38,93%, ovšem také za 2 minuty a 49 sekund.

Z výsledků se těžko usuzuje, jak studenti vnímají zablokování. Účelem zablokování je přimět studenta podívat se na špatně zodpovězenou otázku a látku se doučit. Rozdíl v úspěšnostech bez zámku, po prvním zámku (tj. zablokování odpovědi po relativně krátkou dobu) a po druhém zámku (delší doba čekání, např. 30 minut a více) je zřejmý.

2.6.7 Vliv počtu zobrazených otázek na úspěšnost

Další otázka, která se nabízí, je, jakým způsobem působí počet najednou zobrazených otázek na úspěšnost studentů? Statistika pro celá data je v Tab. 2.24, z ní vyplývá, že čím více otázek, tím vyšší úspěšnost řešení. Průměrná doba odpovědi na jednu otázku je pak kolem 3 minut.

Podíváme-li se blíže na kvízy podle počtu zobrazovaných otázek jednotlivě, zjistíme, že kvízy s pouze 1 otázkou jsou většinou kvízy typu *progressive* a slouží k učení, tj. studenti je používají k cvičení a trénování svých znalostí. V datech se vyskytuje pouze jediný kvíz s 9 zobrazovanými otázkami, nazvaný *Zkouška 10. 1. 2017* z předmětu BI-AG1 a typu *prepared_test*, celková statistika v Tab. 2.24 pro 9 otázek tak odpovídá právě tomuto jednomu kvízu. Co se týče kvízů s 10 zobrazovanými otázkami, v datech jich je celkem 8. Opět se jedná pouze o kvízy z předmětu BI-AG1, v jednom případě se jedná o *Selftest AG1 - Příprava na zkoušku*, v ostatních se jedná o zkouškové testy. Všechny kvízy jsou typu *prepared_test*.

Z předchozího lze vysvětlit, proč je úspěšnost vyšší u kvízů s více zobrazovanými otázkami. Kvízy s pouze jednou otázkou jsou většinou cvičné, kdežto kvízy s více zobrazovanými otázkami jsou přímo zkouškové nebo slouží k přípravě na zkoušku. Lze tedy předpokládat, že jsou studenti lépe připraveni na zkouškové testy a v datech je to skutečně vidět. Toto podporují i statistky v podsekci 2.6.2, kde v Tab. 2.11 a Tab. 2.12 je vidět, že odpovědi během zkouškového mají vyšší úspěšnost než odpovědi během semestru.

2.6.8 Shluková analýza

Ke shlukové analýze byly vybrány algoritmy *k-means* a aglomerativní shlukování. V textu jsou prezentovány jen vybrané výsledky.

Rozdělení studentů podle jejich úspěšnosti ve vybraném kvízu je vidět na Obr. 2.8, interpretaci výsledků lze pojmout stejně jako v podkapitole 2.5.4.

Tabulka 2.24: Vliv počtu najednou zobrazených otázek na úspěšnost řešení, statistika pro všechna data

Počet najednou zobrazených otázek	Úspěšnost	Celková doba řešení skupiny otázek	Průměrná doba řešení na jednu otázku
1	50,9962	00:02:38	00:02:38
9	62,4242	00:33:42	00:03:44
10	65,0147	00:20:06	00:02:00

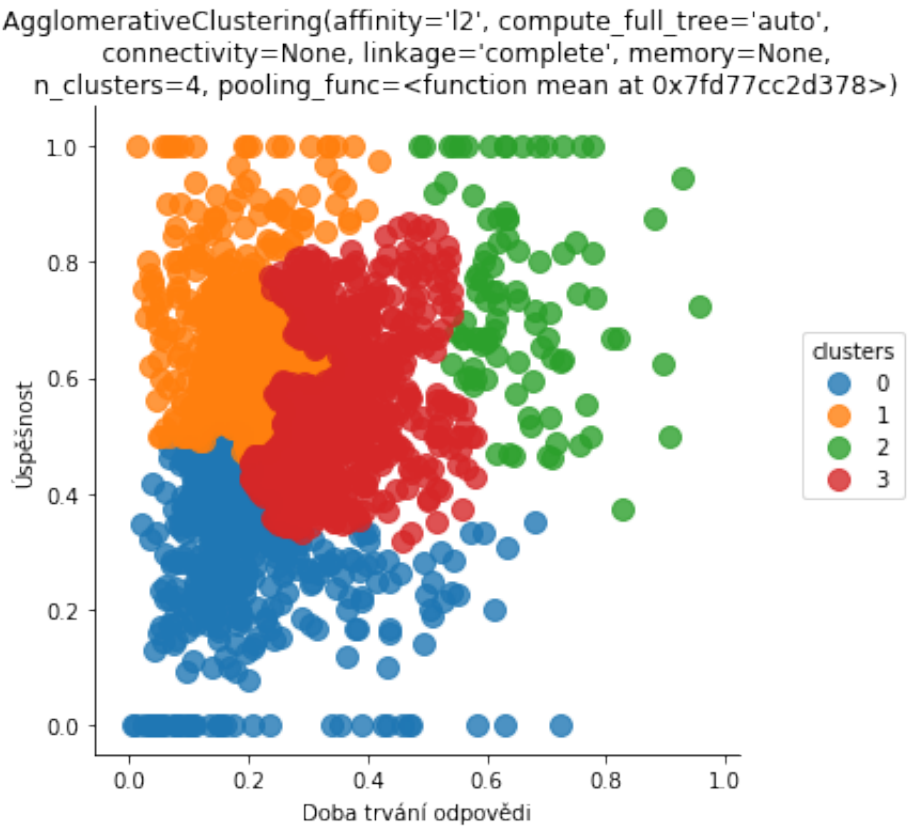
- Modrý shluk (č. 0) – 516 studentů. Odpovídají špatně, na rychlosti nezáleží. Tato skupina má se studiem problémy a bylo by potřeba se na ně zaměřit.
- Oranžový shluk (č. 1) – 613 studentů. Odpovídají rychle a správně. Látce rozumí.
- Zelený shluk (č. 2) – 96 studentů. Odpovídají správně, ale pomalu. Látce rozumí, ale nemají to ještě zažité.
- Červený shluk (č. 3) – 764 studentů, odpovídají relativně správně za průměrnou dobu.

Shlukování bylo provedeno také nad otázkami. Otázky jsou reprezentovány průměrnou úspěšností odpovědí a mediánem doby odpovědi. Shlukování pro jeden vybraný kvíz je vidět na Obr. 2.20, tento model má nejvyšší skóre siluety ze všech testovaných. Na obrázku je vidět celkem 5 shluků. Shluky 1 a 3 reprezentují otázky s dlouhou dobou odpovídání, přičemž shluk č. 3 (červený) otázky snazší, shluk 1 (oranžový) těžší. Na otázky ze shluku č. 1 by bylo vhodné se dále zaměřit, obsahuje otázky, které jsou řešeny dlouho a přesto špatně. Další dělení podle času je pak na shluk č. 2 (zelený), tj. otázky s průměrnou dobou odpovídání, a shluky 0 (modrý) a 4 (fialový). Dva poslední zmíněné shluky jsou otázky, které lze zodpovědět za krátký čas. Otázky v modrém shluku by se hodilo dále zkoumat, jejich úspěšnost je relativně nízká a v kombinaci s nízkou dobou odpovídání mohou být odpovědi odhadovány, otázky špatně položeny (špatně chápány) nebo mají studenti nějakou elementární neznalost, kterou je potřeba napravit.

Vybrané modely lze aplikovat i na další kvízy, k tomu bude možné použít výslednou komponentu.

2.6.9 Outliery

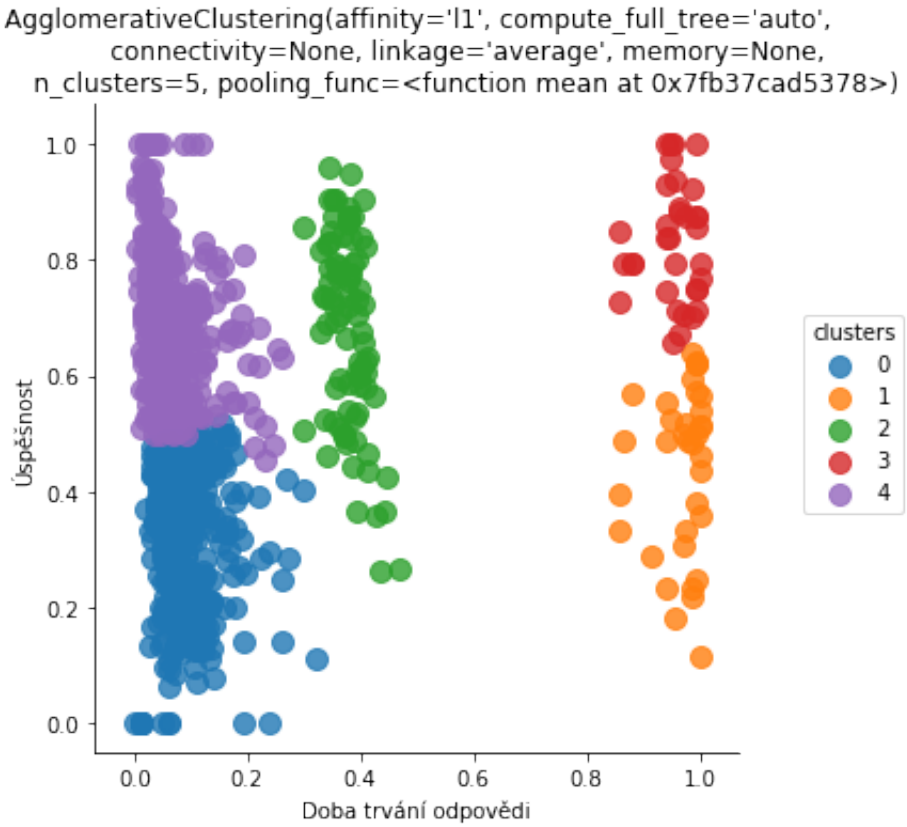
Pro detekci anomálií byl vybrán algoritmus *Local Outlier Factor*, který byl schopen hledat anomálie z lokálního hlediska, parametry algoritmu jsou stejné, jako základní nastavení. Dobrých výsledků dosahoval také *Isolation Forest*.



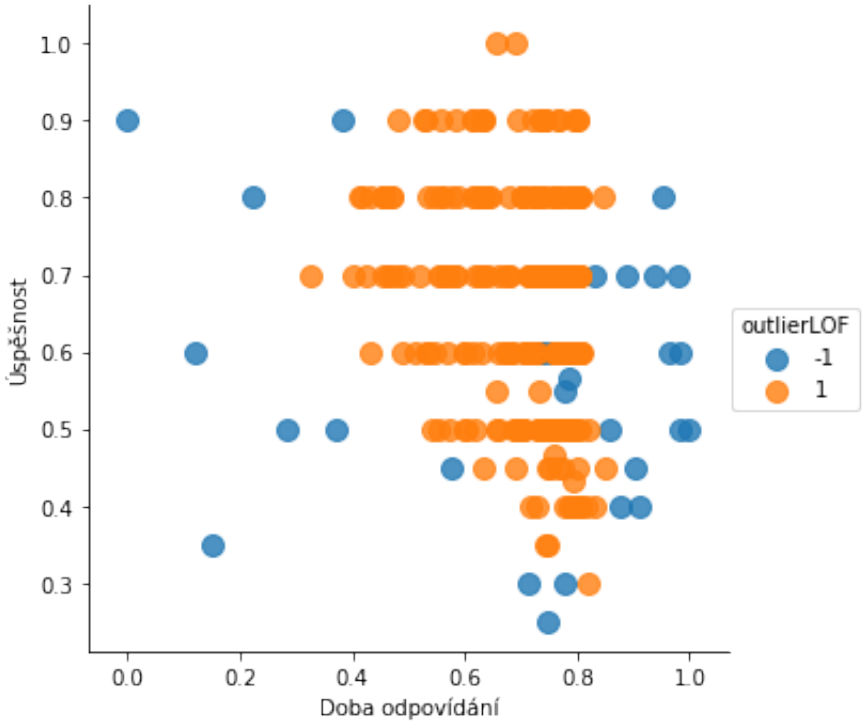
Obrázek 2.19: Aglomerativní shlukování nad studenty, progresivní kvíz z BI-ZMA (Procvičování: Limita a spojitost funkce), metrika *l2*

Na Obr. 2.21 a Obr. 2.22 jsou vidět výsledky hledání anomálií nad dvěma kvízy. Z obrázků jsou anomálie patrné a zdůvodnění je u většiny zřejmé – jsou na okraji lokálního shluku nebo celých dat.

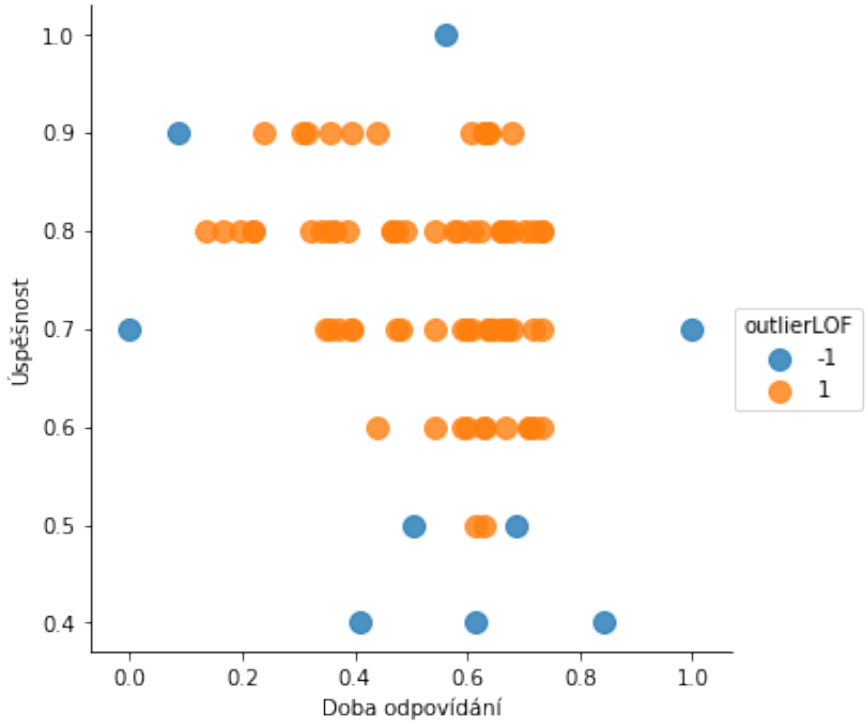
Detekce anomálií mezi otázkami je pak vidět na Obr. 2.23. Z obrázku je zřejmé použití algoritmu *Local Outlier Factor*, jelikož jsou nalezené anomálie i uvnitř shluků podle lokální hustoty dat. Na Obr. 2.24 je pak vidět aplikace algoritmu *Isolation Forest*, který v tomto případě našel lepší výsledky — anomálie jsou skutečně na okrajích a



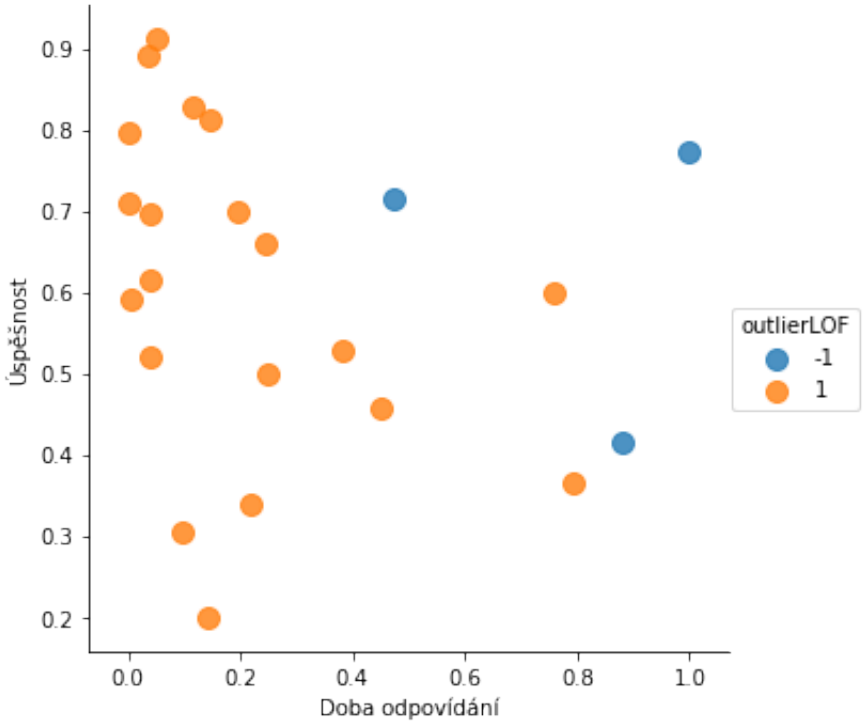
Obrázek 2.20: Aglomerativní shlukování nad otázkami, progresivní kvíz z BI-ZMA (Procvičování: Limita a spojitost funkce), metrika *l1*



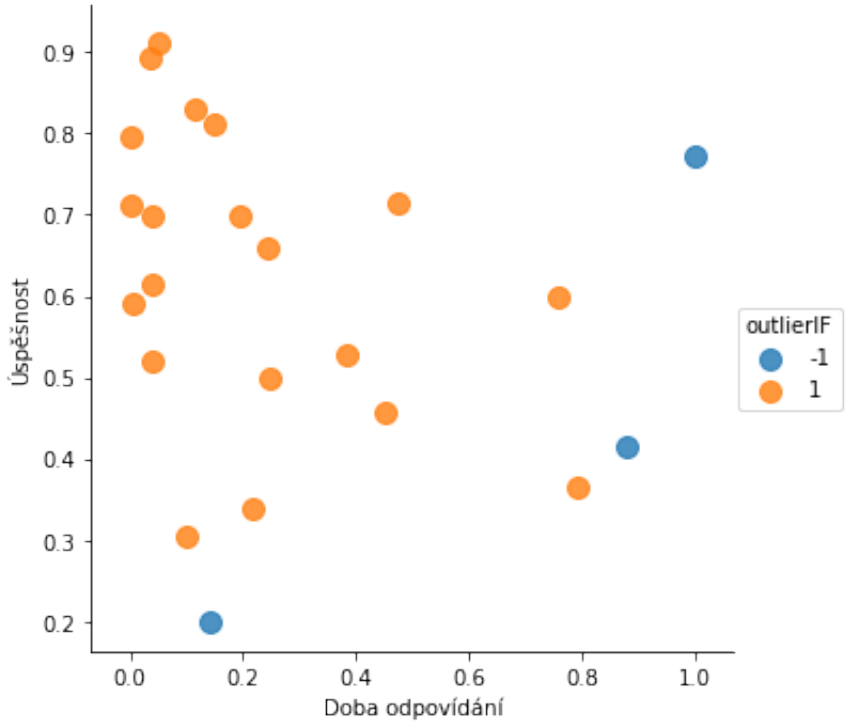
Obrázek 2.21: Detekce anomálií použitím *Local Outlier Factor* mezi uživateli, kontaminace 10%, progresivní kvíz z BI-ZMA (Procvičování: Limita a spojitost funkce)



Obrázek 2.22: Detekce anomálií použitím *Local Outlier Factor* mezi uživateli, kontaminace 10%, kvíz typu *prepared_test* z BI-AG1 (Zkouška 17. 1. 2017)



Obrázek 2.23: Detekce anomálií použitím *Local Outlier Factor* mezi problémy, kontaminace 10%, kvíz typu *progressive* z BI-ZMA (Procvičování: Limita a spojitost funkce)



Obrázek 2.24: Detekce anomálií použitím *Isolation Forest* mezi problémy, kontaminace 10%, kvíz typu *progressive* z BI-ZMA (Procvičování: Limita a spojitost funkce)

Část II

Praktická část

Kapitola 3

Realizace modulu

Tato kapitola je věnována vývoji výsledného modulu.

Při vývoji softwarových produktů je možné použít několik metodik, například vodopád či agilní metodiky. V tomto případě byl samotný vývoj modulu řízen spíše agilním způsobem, kdy postupně probíhaly konzultace a úpravy aplikace podle požadavků vývojáře MARASTu.

Struktura této kapitoly odpovídá jednotlivým částem vývoje produktu. Jednotlivé části jsou analýza, návrh, implementace, testování, nasazení a dokumentace a v každé jednotlivé sekci této kapitoly se věnuji jedné fázi. V první sekci je popsána analýza produktu, co uživatel od modulu chce a očekává, jaké jsou obecné požadavky na produkt, případy užití, procesy apod. V sekci návrh je analýza interpretována z technického pohledu, jsou vybrány vhodné technologie ke splnění požadavků, technické předpoklady vycházející z analýzy, návrh softwarových tříd, rozhraní, způsoby testování apod. Třetí sekce je věnována popisu implementace, tedy naprogramování modulu na základě návrhu. Po implementaci následuje popis testování, jaké technologie byly použity k otestování produktu, jaké chyby byly díky tomu nalezeny a jak s nimi bylo naloženo. Další sekce je věnována nasazení, co za technologie a infrastrukturu je potřeba. Popis je věnován jak cílové platformě, tak i demonstrativnímu nasazení aplikace u freehostingových služeb. Následuje sekce se stručným popisem dokumentace, co a jak bylo použito k dokumentaci, jak k ní přistupovat, používat, jak ji vygenerovat a kde ji najít.

3.1 Analýza modulu

V sekci je popsána analýza modulu pro automatickou analýzu dat ze systému MARAST. Analýza byla zpracována formou textu a UML [52, 53] diagramů. K vytvoření diagramů byl použit produkt *Enterprise Architect* [54], projekt s diagramy pro tento program je dostupný v elektronické příloze. Analýza a požadavky na modul byly konzultovány s administrátorem MARASTu.

3.1.1 Scénáře užití

V této podkapitole jsou popsány dva základní scénáře užití modulu. Scénáře mají sloužit k lepšímu pochopení domény a účelu modulu.

S cílem jasnějšího popisu jsou v procesech zahrnuti i potenciální uživatelé systému MARAST (garant předmětu, zkoušející apod.). Rozdělení rolí však výsledná aplikace nerozlišuje. Případné omezení přístupu k funkcionalitě se řeší v MARASTu, nikoliv ve výsledném modulu.

Základní scénář užití výsledné aplikace bude následující:

1. Koná se zkouška, zkouškové testy se píší na webových stránkách MARASTu.
2. Zkouška se dokončí.
3. Vyučujícího/zkoušejícího zajímá statistika daného kvízu, jak si studenti vedli, které otázky jim dělali problémy apod.
4. V MARASTu zkoušející u daného kvízu vyvolá analýzu dat kvízu.
5. MARAST odešle data na výslednou aplikaci. Ta provede automatickou analýzu přijatých dat.
6. Výsledná aplikace vygeneruje z provedené analýzy zprávu. Uživateli dá přístup k výsledku analýzy.
7. Zkoušející se může podívat na výslednou zprávu.

Druhý uvažovaný scénář je v případě, kdy se chce např. garant předmětu podívat na statistiku kvízů za celý semestr. Scénář pak může vypadat následovně:

1. Garant předmětu, který používá kvízy na MARASTu, označí jeden nebo skupinu kvízů, ke kterým se má udělat analýza. Zároveň označí ty kvízy, jejichž analýza se má provést společně, například globální statistiky ze všech kvízů daného předmětu.
2. Garant vyvolá analýzu označených kvízů.
3. MARAST odešle data na výslednou aplikaci. Ta provede automatickou analýzu přijatých dat podle zadání (např. jednotlivé statistiky kvízů, společné statistiky pro dané kvízy aj.).
4. Výsledná aplikace vygeneruje z provedené analýzy zprávu. Uživateli dá přístup k výsledku analýzy.
5. Garant tak bude mít možnost podívat se na statistiku kvízu.

3.1.2 Uživatelé

Přímá interakce běžného uživatele s výslednou aplikací bude pouze v případě, že uživatel bude chtít vidět výsledek analýzy.

Běžný uživatel tak bude prakticky patřit do jediné skupiny:

- Uživatel systému MARAST – Například garant předmětu, vyučující, cvičící, nebo zkoušející, který si bude chtít vygenerovat analýzu proběhnutého testu.

Toto omezení vychází i z jednoho požadavku na aplikaci ze zadání práce — výsledek analýzy lze prezentovat pouze oprávněným uživatelům, což je podmnožina uživatelů MARASTu.

3.1.3 Doménové modely

V podkapitole je popsán modul z obecného pohledu.

Hlavních entit v doméně není mnoho. Základní dělení a vztahy mezi entitami jsou vidět na Obr. 3.1. V systému bude entita reprezentující analýzu dat, a zastřešuje další dvě entity — výsledek analýzy a log z provedené analýzy.

Entita analýzy bude mít informace o tom, kdo požádal o provedení analýzy, kdy byla vytvořena, jestli byla provedena a identifikátor skupiny analyzovaných kvízů. Entita výsledku analýzy bude mít v sobě informace, statistiky, vizualizace apod., stručně řečeno vše, co je výsledkem analýzy a co by uživatel rád viděl. Entita logu pak obsahuje záznam z provedené analýzy, z ní bude možné vyčíst případné chyby, varování nebo výjimky vzešlé z celého procesu. O provedení analýzy bude žádat uživatel MARASTu, reprezentovaný stejnojmennou entitou.

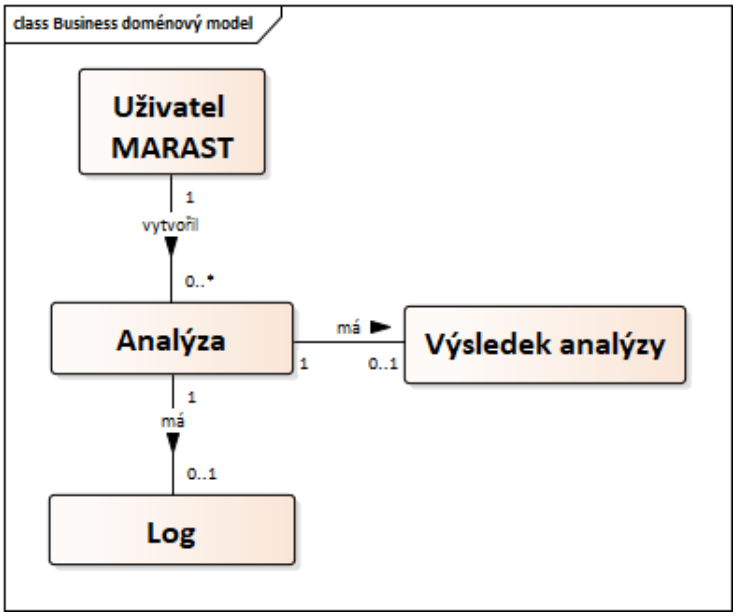
Detailnější doménový model je pak na Obr. 3.2, kde jsou vidět i vlastnosti jednotlivých entit.

3.1.4 Požadavky

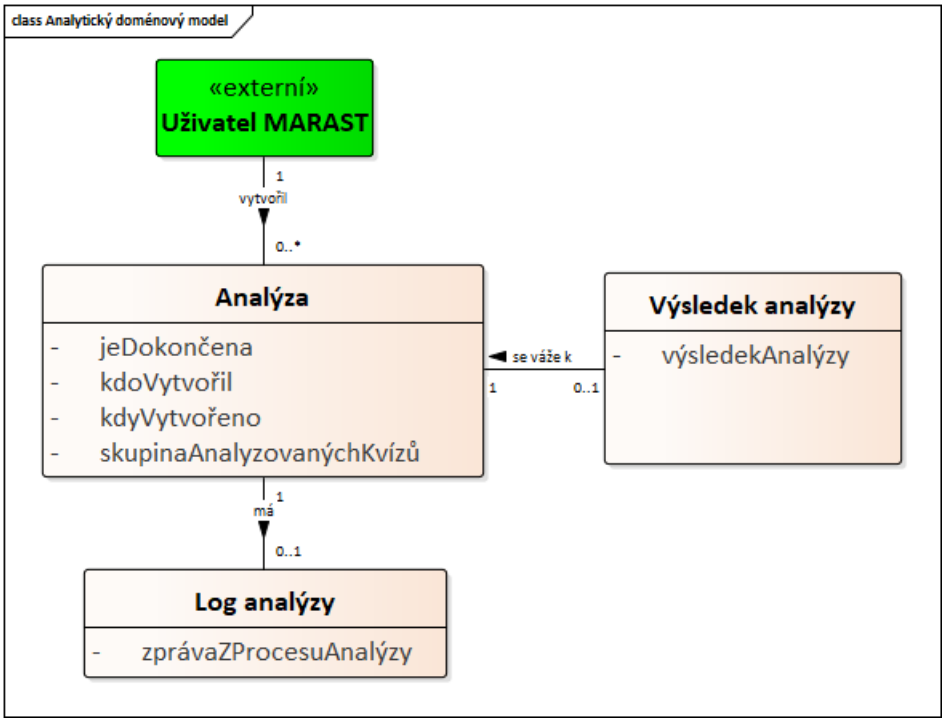
V podkapitole jsou specifikovány požadavky kladené na výslednou komponentu. Základní dělení požadavků je na funkční (co má komponenta umět) a nefunkční (technické parametry, co musí splňovat). Shrnutí požadavků je na Obr. 3.3. Bližší popis požadavků je dále v této podkapitole.

Funkční požadavky:

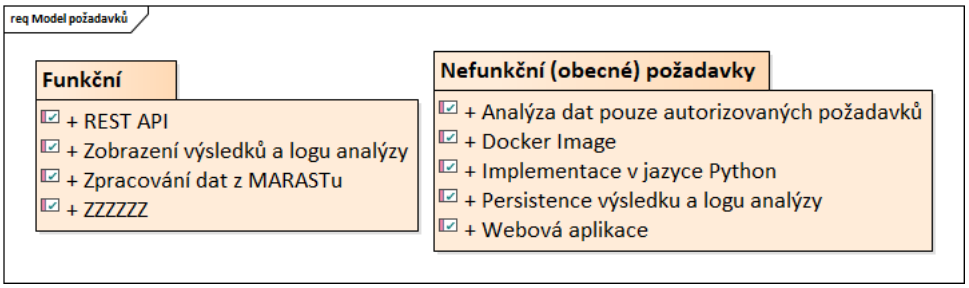
- REST API [55] — Modul umožní MARASTu komunikovat s aplikací prostřednictvím REST API. Toto rozhraní vytvořit či smazat analýzu a získávat informace o provedených analýzách. Komunikace bude probíhat na aplikační úrovni, tj. uživatel bude komunikovat s komponentou prostřednictvím MARASTu.



Obrázek 3.1: Business doménový model



Obrázek 3.2: Analytický doménový model



Obrázek 3.3: Model požadavků

- Zobrazení výsledků a logu analýzy — Modul umožní uživateli zobrazit výsledek analýzy a log. Výsledek i log budou dostupné přes odkaz na webové stránky s příslušným obsahem.

- Zpracování dat z MARASTu — Modul bude schopen přijmout data určená k analýze ze systému MARAST. Formát dat bude stejný jako formát exportu dat určeného k manuální analýze, jak byl popsán v sekci 2.2. Dále data předzpracuje, předzpracování bylo po konzultaci s administrátorem MARASTu ponecháno téměř stejné jako v sekci 2.4, změny jsou diskutovány v podkapitole 3.2.9 v návrhu. Po zpracování bude provedena analýza vybranými metodami, detailní specifikace je v další sekci, 3.2.9. Výsledek analýzy bude uložen ve formátu použitelném pro zobrazení webových stránek, návrh vzhledu výsledku analýzy je opět v části další sekce 3.2.6.

Nefunkční požadavky:

- Analýza dat pouze autorizovaných požadavků — Modul bude analyzovat data pouze z autorizovaných zdrojů. S ohledem na specifikaci oprávněných uživatelů a rolí v systému MARAST (mimo tento modul) se bude o přístup oprávněných uživatelů starat právě MARAST, nikoliv tento modul. Autorizace požadavků bude na úrovni aplikací a bude formou ověření původu dat (podpisem), přesný popis způsobu ověření je v sekci zabývající se návrhem, v podsekci 3.2.8.

- *Docker Image* [56] — Modul bude v tzv. *Docker Image*. Bližší informace v kapitole nasazení 3.5.

- Implementace v jazyce Python — Modul bude implementován za použití jazyka Python a s ním souvisejících modulů a technologií.

- Persistence výsledku a logu analýzy — Modul bude uchovávat výslednou zprávu a log v databázi.

- Webová aplikace — Modul bude možné kontaktovat pomocí webového rozhraní, to umožní veškerou komunikaci mezi aplikacemi, resp. mezi modulem a uživatelem.

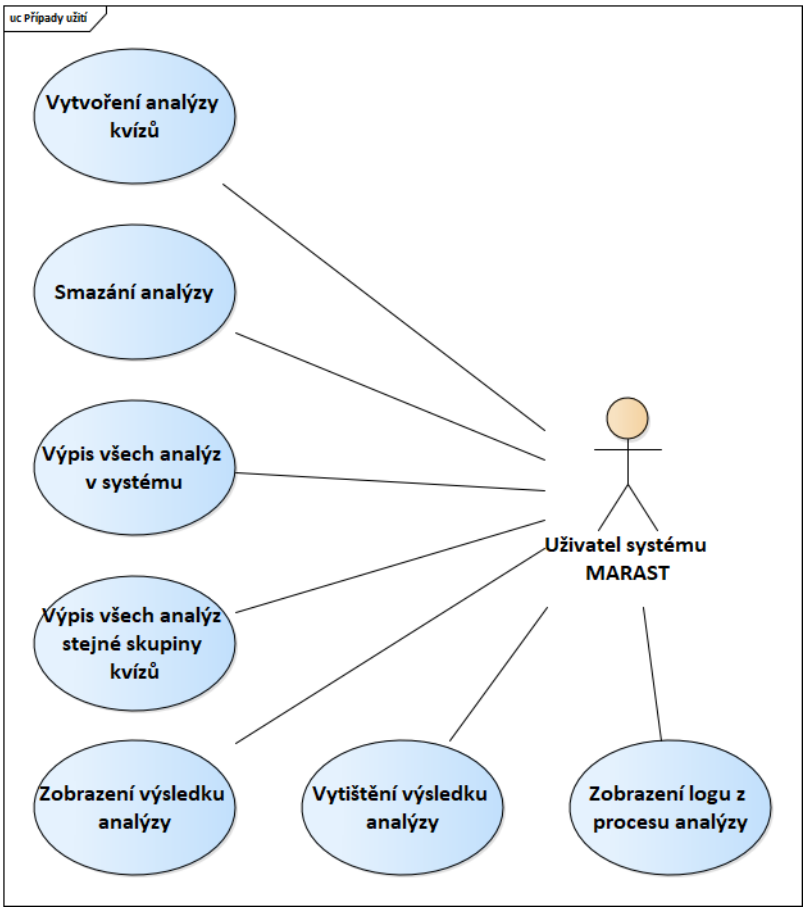
3.1.5 Případy užití

Případy užití jsou znázorněny na Obr. 3.4. S ohledem na jediného uživatele a jasný cíl užití komponenty jsou případy užití orientované hlavně na práci s analýzou, tj. její vytvoření, zobrazení a smazání. Následuje stručný popis případů užití:

- Vytvoření analýzy kvízů — Uživatel může chtít vytvořit analýzu kvízů.
- Smazání analýzy — Uživatel bude schopen smazat analýzu.
- Výpis všech analýz v systému — Pro administrativní účely, uživatel si může nechat vypsat všechny analýzy v systému a podle toho např. mazat starší výsledky.
- Výpis všech analýz stejné skupiny kvízů — Analýzy jsou vždy vytvořeny z nějaké skupiny kvízů, uživatel se tak může chtít podívat na výsledek analýzy pro dané kvízy z různých časových okamžiků a prohlédnout si tak vývoj statistik v čase.
- Zobrazení výsledků analýzy — Uživatel si zobrazí výsledek analýzy.
- Vytištění výsledků analýzy — Uživatel si vytiskne výsledek analýzy na papír/do souboru.
- Zobrazení logu z procesu analýzy — Uživatel nahlédne do logu např. v případě, že analýza doběhla s chybou. Může tak identifikovat chybu v datech (chybějící sloupec, chybné hodnoty, ...) a opravit ji.

3.1.6 Pokrytí případů užití funkčními požadavky

Pokrytí případů užití funkčními požadavky je zobrazeno na Obr. 3.5.



Obrázek 3.4: Model případů užití

Source \ Target	Smazání analýzy	Výpis všech analýz stejné skupiny kvízů	Výpis všech analýz v systému	Vytištění výsledku analýzy	Vytvoření analýzy kvízů	Zobrazení logu z procesu analýzy	Zobrazení výsledku analýzy
REST API	↑	↑	↑				
Zobrazení výsledků a logu analýzy				↑		↑	↑
Zpracování dat z MARASTu					↑		

Obrázek 3.5: Mapování realizace případů užití funkčními požadavky

3.2 Návrh, architektura, design

V sekci je popsán návrh výsledného balíčku a jeho rozhraní pro komunikaci s MARASTem. Návrh je tvořen s přihlédnutím k zadání práce, ve kterém je specifikováno, že modul má být napsán v jazyce Python.

3.2.1 Architektura

Architektura aplikace bude rozdělena do několika vrstev, každá vrstva se bude starat o určitou část funkcionality. Rozdělení bude následující:

1. Prezentační vrstva — Webové stránky, REST API.
2. Logická vrstva — Logika aplikace, předzpracování a analýza dat.
3. Datová vrstva — Persistence dat v databázi a další případné operace s daty.

Výsledný modul bude rozdělen na několik částí. Pro komunikaci s MARASTem bude sloužit webový server s REST API, kterým se budou do systému dostávat data. S ohledem na to, že zpracování dat může trvat i několik desítek minut (zjištěno během ruční analýzy dat), bude webová aplikace spouštět zpracování dat v jiném procesu. Po doběhnutí analýzy proces uloží výsledek a ukončí se, případně se stane nečinným do příchodu dalšího požadavku. Při přístupu k výsledku analýzy pak webová aplikace zkontroluje, jestli zpracování už doběhlo, a podle toho zobrazí uživateli výsledek analýzy, nebo výzvu k počkání.

Pro implementaci webové aplikace bude využit jeden z webových frameworků dostupných pro jazyk Python. To umožní rychlejší vývoj a do určité míry standardizovanou strukturu aplikace, další případný rozvoj tak bude přívětivější pro pokračovatele. Na výběr je z několika frameworků — *Flask* [57], *Django* [58] apod. Zvolen byl *Flask*, s ohledem na mé znalosti, jednoduchost použití a výbornou dokumentaci včetně tutoriálů a případné podpory od komunity. Struktura celé komponenty tak bude následovat příklad a doporučení v dokumentaci *Flasku* [59].

3.2.2 Prezentační vrstva

Prezentační vrstva bude rozdělena na dvě části.

První část uživateli poskytuje přístup pouze k výsledné zprávě a logu z analýzy. Na základě dodaného identifikátoru analýzy vrátí danou zprávu ve formátu webové stránky (HTML [60], CSS [61] a další), ve stejném formátu pak i log.

Druhou částí vrstvy je REST API. Toto rozhraní bude poskytovat přístup k informacím o analýzách, možnost jejich vytvoření a smazání. Rozhraní je

určené pro komunikaci s uživatelem prostřednictvím MARASTu, tj. prezentační vrstva MARASTu dá uživateli možnost volat metody rozhraní tohoto modulu.

3.2.3 Logická vrstva

Logická vrstva se stará o funkcionalitu aplikace. Její součástí je implementace chování REST API a logiky pro zobrazení výsledku a logu z analýzy uživateli. Dále obsahuje doménové třídy, logiku pro kontrolu, předzpracování a analýzu dat. Tato vrstva bude tvořena metodami a třídami v Pythonu a bude dodržovat strukturu běžných aplikací ve zvoleném frameworku. Pro spouštění procesu, který bude zpracovávat analýzy, bude opět využit jeden z dostupných balíčků. Psát tytu funkcionalitu přímo v systémových voláních není příliš perspektivní ani pohodlné.

3.2.4 Datová vrstva

Uložení dat je potřeba u přehledu analýz (které jsou v systému), výsledků analýz a logů. Pro usnadnění komunikace s databází bude snaha využít balíček, který umožňuje komunikovat s databází prostřednictvím rozhraní vyšší úrovně. To usnadní práci jednak tím, že nebude nutné psát přímo SQL [62] dotazy a také to poskytne do určité míry nezávislost na vybrané databázi. Pro persistenci dat bude zvolena *PostreSQL* [63], jelikož nabízí dobré rozhraní, je volně dostupná a má širokou podporu Python balíčků.

U způsobu ukládání výsledků analýzy je situace složitější, neboť velikost může být několik set KB až jednotky MB. Pro takové případy je v klasických SQL databázích (*PostgreSQL*, *MySQL*) vhodný datový typ *LargeBinary* pro uložení souboru s předem neznámou velikostí. S ohledem na to, že výsledky analýzy jsou ve formě textu a obrázků, je možné uvažovat i o neomezeném textovém datovém typu *text* či *varchar(max)*, opět za předpokladu uložení obrázků v kódování *base64*. V případě ukládání celých dokumentů by se dalo uvažovat i o tzv. *NoSQL* databázích orientovaných na dokumenty, jako je *Cassandra* [64], *Redis* [65] nebo *Riak* [66].

3.2.5 Definice rozhraní

V podkapitole je popis komunikačního rozhraní, tj. webové aplikace a REST API.

Pro REST API jsou definovány následující funkce a mapování:

- Požadavek pro vytvoření analýzy: `POST /analysis`
  - `user_id` Číslo, identifikátor uživatele, který žádá o vytvoření analýzy.

  - `quiz_definitions` Textový řetězec, informace o kvízech, formátem odpovídá souboru *quiz.csv*, viz 2.2.
  - `quiz_answers` Textový řetězec, informace o odpovědích, formátem odpovídá souboru *data.csv*, viz 2.2.
  - `quiz_analysis` Pole polí identifikátorů kvízů v datech, tj. celých čísel. Definuje skupiny kvízů, jejichž analýza se má provést. Tj. například hodnota `[[1,2],[1]]` znamená, že se z poskytnutých dat vyberou pouze data pro kvízy s identifikátory 1 a 2 a provedou se dvě analýzy. Jedna společná analýza pro kvízy s identifikátory 1 a 2, druhá analýza pro samotný kvíz s identifikátorem 1.

- Požadavek pro smazání analýzy: `DELETE /analysis/<analysis_id>`
  - `analysis_id` Celé číslo. Identifikátor analýzy, která má být smazána.
  - `analyses_group_id` Celé číslo. Identifikátor skupiny kvízů, ze kterých byla analýza vytvořena.

Funkce vracející webové stránky budou mít následující mapování:

- `GET /analysis/<analysis_id>` Vrátí webovou stránku s výsledkem analýzy.
- `GET /analysis_log/<analysis_id>` Vrátí webovou stránku s logem z analýzy.
- `GET /analyses_list/<analysis_id>` Vrátí webovou stránku se seznamem analýz stejných kvízů.

Rozhraní bude poskytovat odpovědi se statovými kódy podle standardu HTTP [67].

3.2.6 Reprezentace výsledků analýzy

Podkapitola obsahuje návrh formátu reprezentace výsledku analýzy uživatelům.

Jak již bylo zmíněno v předchozí kapitole, výsledek analýzy bude dostupný přes webové rozhraní. Nabízí se tedy použít technologie HTML5, CSS, případně JavaScript. Reprezentaci v HTML podporuje i výběr balíčku *Flask*, která má v sobě modul *Jinja2* [68], což je generátor HTML stránek z šablon. Toto řešení podporuje i případ, kdy by si chtěl uživatel výsledek vytisknout. Webové prohlížeče nabízí možnost vytisknout si stránku. V případě, že by bylo nutné výsledek analýzy pro potřeby tisku upravit, je možné použití speciálního CSS stylu pro tisk.

Výsledky analýzy uložené v proměnných Pythonu se předají do HTML šablon, ve kterých se definuje struktura výsledné reprezentace.

Jediným problémem je uložení obrázků a grafů, protože jich může být mnoho a zabírají relativně hodně paměti v porovnání s textem. Pro uložení obrázků se nabízí několik možností:

1. *imageData* a vložení dat obrázku do HTML ve formátu *base64* [69]. Výhodou je jediný soubor, nevýhodou pak jeho velikost. Důsledkem je také méně přehledný zdrojový HTML kód. To by se dalo považovat za negativum, nicméně zdrojový kód stránky není zamýšlen k prezentování, pouze jeho interpretace (finální podoba) v prohlížeči.

2. Využití databáze k uložení obrázků. Při každém dotazu by byly načteny obrázky, vloženy do šablony, vygenerován výsledek a odeslán v odpovědi. Potenciálně mnoho malých souborů a výsledek analýzy by se musel vždy znovu skládat (i s potenciálním využití *cachování*), tj. při každém požadavku vytáhnout obrázky z databáze, vyplnit HTML šablonu a vrátit stránku uživateli.

3. Úložiště obrázků v souborovém systému. Obrázky by mohly být konzistentně pojmenovány, ve složkách příslušných analýz (pojmenování např. dle identifikátoru analýzy). Při požadavku by se pouze načetli z dané složky souborového systému a v šablonách by k nim byl pouze odkaz. Problém však může být právě s uložením v souborovém systému na disku, kdy úložiště nemusí být úplně persistentní, viz kapitola 3.5.

Pro uložení obrázků byla zvolena metoda využívající *imageData*, její nevýhody se zdají být v porovnání s ostatními možnostmi minimální.

Obsah zprávy z analýzy bude následující:

- Hlavička stránky — Informace o analýzy, odkaz na log, odkaz na další analýzy stejné skupiny kvízů.
- Obecné informace o analyzovaných kvízech — název, typ kvízu, předmět, semestr atd. Jedná se hlavně o informace z definic kvízů.
- Obecné statistiky z kvízu — celkový počet odpovědí, otázek, uživatelů, zámků atd.
- Statistiky, vizualizace, shlukování a detekce anomálií v otázkách (problémech k řešení).
- Statistiky, vizualizace, shlukování a detekce anomálií v uživatelích.
- Identifikace otázek zodpovězených špatně i přesto, že je stejný uživatel již předtím zodpověděl správně.

Obsah zprávy bude dále měněn a přizpůsobován požadavkům, zejména po testování.

Co se týče logu z procesu analýzy, záznamy budou co možná nejjednodušší. Záznam v logu bude obsahovat čas, informaci o stavu a komentář o provedené akci/chybě/výjimce.

3.2.7 Zpracování požadavku a provedení analýzy

Jelikož mohou mít analyzovaná data několik desítek MB (stovky tisíc záznamů) či více a výpočet analýzy může trvat řádově desítky minut, je nutné zpracovat požadavek pro provedení analýzy asynchronně.

Prakticky to znamená, že webová aplikace obdrží požadavek s daty pro analýzu, do databáze uloží záznam o nové analýze a data ke zpracování pošle jinému procesu. Tento proces je běžně označovaný anglickým slovem *worker*. Proces provede analýzu, přidá do databáze její výsledek a log, následně aktualizuje záznam s tím, že je analýza hotová. Webová aplikace pak při požadavku o zobrazení analýzy zkontroluje příznak u vyžádané analýzy a buď vrátí výslednou zprávu, nebo uživatele požádá o počkání.

Webová aplikace i pracující proces budou programy v Pythonu. K předání dat mezi aplikacemi se využije zprostředkovatele zpráv (ang. *Message Broker*). S přihlédnutím k dostupným technologiím se nabízí jako nejlepší varianty *RabbitMQ* [70] nebo *Redis* [65]. Obě varianty lze využít zdarma, *RabbitMQ* je distibuována pod *Mozzila Public License* [71], *Redis* je *open source* s BSD licencí [72]. Obě varianty jsem shledal srovnatelné, nakonec byl vybrán *Redis* s přihlédnutím k možnostem nasazení.

Co se týče komunikace mezi webovou aplikací a procesem na pozadí, nabízí se modul *Celery* [73], případně *Redis Qeue* [74], který je přímo určený pro *Redis*. *Celery* je komplexní modul s širokou podporou zprostředkovatelů zpráv, rozhraní pro *RabbitMQ* je přímo součástí balíčku *Celery*, pro *Redis* je nutné doinstalovat příslušný modul s rozhraním pro komunikaci s databází. *Redis Qeue* je malý kompaktní modul, který je určen čistě pro *Redis*. Z těchto dvou variant byl vybrán modul *Celery*, jelikož nabízí obecnější použití a v případě nutnosti změny zprostředkovatele zpráv není potřeba přeimplementovat funkcionalitu aplikace.

3.2.8 Zabezpečení

Zabezpečení webového rozhraní bude řešeno pomocí podepisování požadavků posílaných na modul a šifrovaním komunikace pomocí HTTPS [75]. V případě, že bude chtít jiný systém využít služby výsledného modulu, hlavička požadavku bude muset mít definované pole s podpisem.

Zabezpečení požadavku na vytvoření analýzy bude následující:

1. Tělo požadavku je naplněno daty, tělo je ve formátu JSON.

2. Vytvoří se kód HMAC. Tj. za použití zkomprimovaného těla požadavku ve formátu JSON, tajného sdíleného klíče a vybrané hashovací funkce se vygeneruje výsledný kód.

3. Výsledný kód bude upraven do hexadecimálního tvaru, aby byl reprezentovatelný textovým řetězcem.

4. Výsledný kód se vloží do hlavičky HTTPS požadavku.

Server pak přijme požadavek, udělá stejný postup jako klient a v případě, že podpisy vyjdou stejně, je požadavek autorizován. Pokud vyjdou různě, požadavek je odmítnut.

Podobná metoda je například využívána serverem GitHub [76] při použití tzv. *webhooks* [77], je tedy otestována v provozu a lze ji považovat za ověřenou.

3.2.9 Automatizace analýzy dat

Analýzu dat je potřeba zautomatizovat tak, aby dávala použitelné a snadno interpretovatelné výsledky. Z manuální analýzy provedené v předchozí části práce budou vybrány takové statistiky a metody, které lze provést bez přítomnosti člověka, který by mohl interaktivně ladit parametry použitých algoritmů a modelů.

U základních statistik typu průměrná úspěšnost nebo medián doby trvání odpovědi není moc co řešit, jedná se o přímočarý výpočet. Problém nastává u algoritmů shlukování a detekce odlehlých hodnot a anomálií. Výsledky těchto algoritmů a jejich interpretovatelnost jsou silně závislé na datech. V případě nekvalitního pročištění dat může shlukování poskytnout nepříliš použitelné výsledky, jak bylo ukázáno v předchozí části práce v manuální analýze v podsekci 2.5.3. S ohledem na to byl proces čištění dat více zpřísněn a odlehlé hodnoty jsou mazány s mnohem přísnějšími kritérii.

K výběru vhodného modelu shlukování bude použito skóre siluety. Přestože se může zdát, že podle Obr. 2.9 a Obr. 2.10 v podkapitole 2.5.4 je použití skóre *Calinski-Harabasz* lepší i z hlediska interpretace výsledků než skóre siluety, bude skóre siluety upřednostněno při porovnávání kvalit modelů. Jak bylo diskutováno v podkapitole 2.5.2, hodnota skóre siluety poskytuje lépe interpretovatelné informace o shlucích. Při shlukové analýze budou vyzkoušeny algoritmy *k-means* a aglomerativní shlukování. Počet shluků bude nastaven na 4 až 7, podle výsledků *Elbow* metody v podkapitole 2.5.4 a výsledků v manuální analýze. Další parametry algoritmů se procyklují, všechny modely se spustí a vybere se nejlepší model podle skóre siluety. Toto řešení není vhodné pro vysoce dimenzionální data, což ale není náš případ.

Automatizovanou analýzu však nelze dělat bezchybně, vždy bude záležet na datech, která se uživatel rozhodne odeslat. Ukázkovým příkladem může být např. analýza 2 kvízů s různým počtem zobrazovaných otázek a výrazně odlišnou časovou náročností odpovědí. Společná analýza těchto kvízů z hlediska

doby odpovídání nedává moc smysl, ovšem takový případ lze automatizovaně a bezpečně zpracovat jen velmi obtížně. O těchto případech lze jen v omezené míře informovat uživatele, např. upozorněním na analýzu kvízů s různým počtem zobrazovaných odpovědí. Aplikace nebude bránit uživateli provést analýzu na takových datech, v tomto bodě je nutné předpokládat, že uživatel ví přesně, co chce.

3.3 Implementace

Předchozí kapitoly a sekce byly věnovány analýze a návrhu řešení. Nyní je na řadě implementace samotného modulu.

3.3.1 Konfigurace

Nastavení aplikace, konfigurace databáze, zprostředkovatele zpráv apod. je v souboru `config.py`. Ten obsahuje třídu se všemi konfiguracemi, třída se předá instanci aplikace (třída *Flask*) a komponenta je tak nakonfigurována k běhu. Konfigurace je psána tak, aby načítala jednotlivé proměnné z prostředí operačního systému a v případě neexistence těchto proměnných se nastavení základní hodnoty.

Z konfigurace také čerpá proces běžící na pozadí, který potřebuje své vlastní připojení k databázi.

3.3.2 Datová vrstva

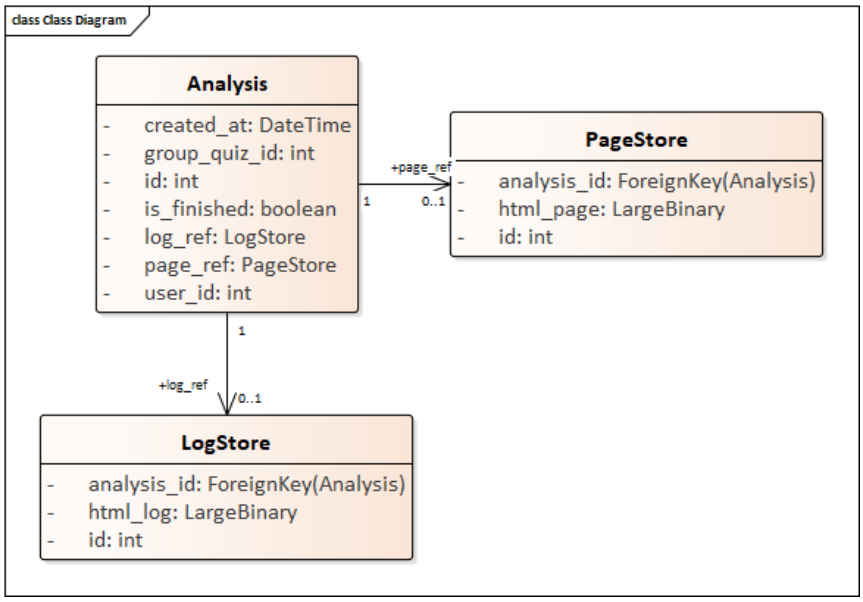
K persistenci dat jsou využity moduly *SQLAlchemy* [78] a rozšíření *Flask-SQLAlchemy*. Využití těchto modulů dovoluje definovat datovou vrstvu jako třídy v Pythonu a moduly se postarají o jejich správnou reprezentaci v SQL databázi.

Schéma databáze je tak automaticky generováno použitím funkcí modulu *SQLAlchemy*. Schéma je definováno v souboru `models.py` pomocí Python tříd, vizualizace je vidět na Obr. 3.6. Z takto definovaných tříd vygeneruje *SQLAlchemy* automaticky SQL se schématem databáze a případnými migracemi. Úpravy databáze jsou generovány do složky `migrations`.

3.3.3 Popis dat

Názvy sloupců s informacemi, které jsou očekávány v příchozích datech s kvízy jsou definovány v souboru `columns.py`. V případě přejmenování sloupce stačí z tomto souboru upravit hodnotu proměnné a aplikace se tomu po restartování přizpůsobí.

Očekávané položky v `POST` a `DELETE` požadavcích, očekávané sloupce v poskytnutých datech apod. jsou definovány v polích v souboru



Obrázek 3.6: Diagram tříd datové vrstvy

`data_requirements.py`, je možné tak přidávat další položky pro kontrolu do těchto polí.

3.3.4 Asynchronní zpracování dat

K vytvoření práce pro analýzu dat stačilo použít dekorátor z balíčku *Celery*, viz Obr. 3.7. Data určená procesu pak stačí vložit do metody jako parametry. *Celery* vyžaduje specifikovat adresu zprostředkovatele zpráv a databázové připojení, což je v konfiguraci popsané v předchozí podkapitole 3.3.1.

```
@celery.task
def background_task(analysis_id, inner_array,
                    quiz_definitions, quiz_answers):
    ...
```

Obrázek 3.7: Definice funkce pro proces běžící na pozadí pomocí dekorátoru *Celery*

3.3.5 Implementace rozhraní

Implementace rozhraní byla díky použití *Flasku* jednoduchá. Stačilo definovat Python metody a přidat k nim dekorátor se specifikací cesty a HTTP metody, jak je vidět na Obr. 3.8. Tento způsob stačil na definici celého rozhraní, tj. jak REST API, tak webových stránek.

```
@app.route('/analysis', methods=['POST'])
def create_analysis():
    ...
```

Obrázek 3.8: Mapování URL na metody pomocí dekorátoru

Zabezpečení požadavků bylo implementováno tak, jak je popsáno v podkapitole 3.2.8. Detailní implementace včetně příkladů je dostupná v dokumentaci a v testech, krok za krokem. Formát odpovědí u metod REST API má následující formát:

- V případě chyby obsahuje tělo zprávy položku `message` s informací o tom, co se stalo. Stavový kód odpovídá standardu.
- Metoda pro vytvoření analýzy (`POST /analysis`) při správném provedení vrací pole dvojic, kde každý prvek pole obsahuje informace o vytvořených analýzách. V prvku je dvojice hodnot `analysis_id` a `analysis_group_id`.
- Metoda pro smazání analýzy (`DELETE /analysis/<analysis_id>`) při bezchybném provedení vrací zprávu s kódem 200, v těle požadavku jsou pak dva řetězce, `status` a `message`, informující o tom, která analýza byla smazána.

Klient si musí adresu k jednotlivým analýzám sestavit sám. Zbylé metody vrací webové stránky se stavovými kódy podle standardu HTTP.

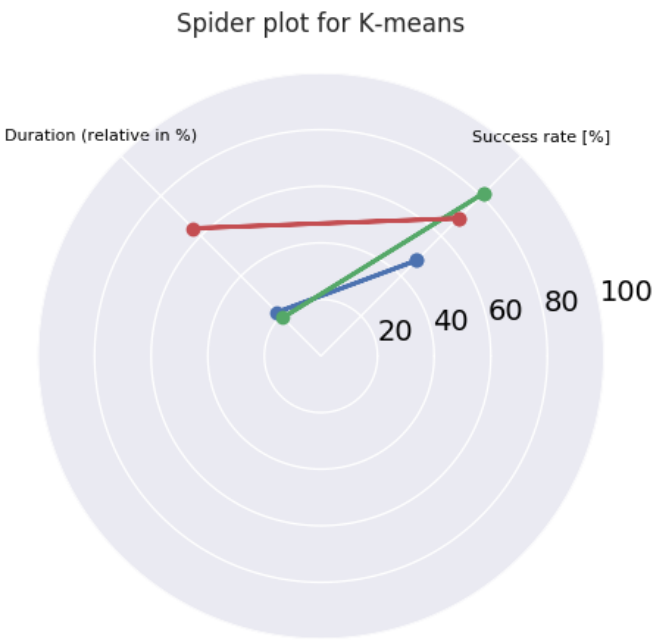
3.3.6 Prezentační vrstva

Pro generování HTML stránek je využit balíček *Jinja2*, který je součástí *Flasku*. Šablony HTML jsou uloženy ve složce `templates`. Šablony byl vytvořeny pro výslednou zprávu analýzy,

Zároveň byla nalezena a implementována další vizualizace výsledků shlukování pomocí tzv. *Spider plot* nebo *Radar plot* [79]. Přehledně zobrazuje reprezentanty jednotlivých shluků, výsledek je vidět na Obr. 3.9. Tento způsob vizualizace se hodí zejména pro data s vyšším počtem dimenzí.

Pro převod informací z analýzy do prezentační vrstvy byly vytvořeny následující třídy:

- `DataframeItem` — Reprezentuje *pandas.Dataframe*, převádí jej na HTML `table`.
- `ImageItem` — Reprezentuje obrázek/vizualizaci, reprezentována HTML tagem `figure` s vloženým `img`.
- `InformationItem` — Reprezentuje ostatní textové informace, v základu se jedná o HTML tag `p`.



Spider plot for K-means

Obrázek 3.9: Ukázka *Spider plot*, jednotlivé přímky reprezentují nalezené shluky. V případě více dimenzí se vytvoří mnohoúhelník.

Tři zmíněné třídy dědí od společného předka, `BaseItem`. Tyto třídy implementují metody `to_standard_html()`, která po zavolání vrátí HTML reprezentaci daného objektu.

3.3.7 Nástroje použité k analýze

Nástroje k analýze již byly stručně zmíněny v předchozí části práce 1.2.3. Nástroje byly zvoleny s přihlédnutím k zadání práce. Výsledný analytický modul má být v jazyce Python. S ohledem na to, že pro Python existuje mnoho knihoven a modulů pro analýzu dat, matematické výpočty, statistiku a strojové učení, byl Python a jeho technologie zvoleny jako nástroje k analýze dat.

Některé balíčky již byly zmíněny, pro shrnutí je uveden seznam všech pohromadě. Vedle standardních balíčků Pythonu jsou využity následující:

- *pandas* [24] hlavní modul pro analýzu dat.
- *matplotlib* [80] pro vizualizaci dat, grafů apod.

- *seaborn* [81] je rozšíření *matplotlib* pro vzhlednější a pohodlnější vytváření grafů.
- *scikit-learn* [25] pro algoritmy strojového učení, tj. shlukování, detekce anomálií a odlehlých hodnot, normalizace dat apod.
- *numpy* [82] pro optimalizované matematické výpočty. *pandas* i *scikit-learn* tento balíček využívají.

Během manuální analýzy byl navíc použit modul *Jupyter Notebook* [50].

Důvodem volby těchto modulů je také to, že nabízí relativně jednoduché rozhraní pro použití, širokou podporu komunity a neustálý vývoj.

Použití těchto modulů se neobešlo bez problémů. Modul *pandas* byl v průběhu vývoje modulu několikrát aktualizován a jeden z vedlejších efektů aktualizace byla nutnost přeimplementovat část analýzy dat.

Jeden z řady hůře pochopitelných problémů byl s tzv. *SettingWithCopyWarning*, na toto téma vznikly samotné články [83]. Přitom se jedná o relativně jednoduchý koncept, kde se rozlišuje mezi daty samotnými (tabulkou) a pouhým pohledem (odkazem) na data, stejně jako v databázích. Toto varování se objeví vždy při pokusu o úpravu pouhého pohledu, nikoliv dat samotných. Řešením tohoto varování je explicitní kopie pohledu.

Dalším problémem bylo použití seskupovací funkce *groupby* v kombinaci s výpočtem průměru sloupečku typu *timedelta64* [84] [85]. To se dalo snadno vyřešit použitím vlastní agregační funkce a nikoliv knihovní.

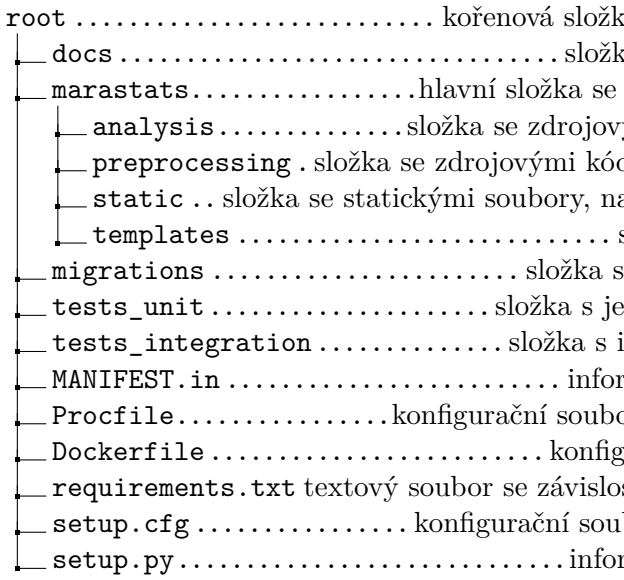
Další problém byl s datovými typy reprezentující čas. Stručně řečeno, používání datové typy pro čas z více modulů nefungovalo správně, ačkoliv měli být navzájem převoditelné. Řešením bylo explicitní převádění datových typů na jeden a toho se držet. Bližší informace jsou v příslušné podsekci.

3.3.7.1 Datové typy

S vybranými nástroji pro analýzu můžeme specifikovat datové typy, které budou jednotlivé sloupce mít. Přehled je v Tab. 3.1, jedná se o informace přímo z *pandas.Dataframe*. Sloupce typu *object* vyjadřují řetezce textu, *int64* celé číslo, *bool* pravdivostní hodnotu, *datetime64* časový okamžik a *timedelta64* délku časového intervalu.

K datovým typům reprezentující čas a časový interval je nutné poznamenat, že se s nimi v každém případě pracuje přes speciální datové typy *pandas.Timestamp* pro vyjádření časového okamžiku (datum) a *pandas.Timedelta* pro vyjádření doby trvání. Typy uvedené v Tab. 3.1 pochází z knihovny *numpy*, kterou *pandas* uvnitř používá, nicméně explicitně míchat dohromady typy z obou modulů a případně ještě z modulu *datetime* působilo při implementaci problémy s automatickým přetypováním.

Jedním (vážným) důvodem je různé chování a vyjadřování prázdných hodnot. Datové typy *pandas* tento problém řeší elegantně a byť bude kód o něco



Obrázek 3.10: Základní struktura komponenty

delší kvůli převodům, bude prost chyb a spoléhání se na automatické přetypování a bude zajištěn jeden způsob reprezentace dat.

3.3.8 Výraznější implementační problémy

Při implementaci (a již při manuální analýze) se vyskytly problémy s verzemi použitých Python balíčků. Zejména balíček *pandas* byl několikrát aktualizován, což způsobilo nefungující kód. Kód byl aktualizován pro nejnovější verzi, ovšem s ohledem na tuto zkušenost je další vývoj a nasazení modulu omezeno na určité verze balíčků. Tomuto omezení nahrává i fakt, že se na implementaci samotných balíčků podílí široká komunita vývojářů a změny a aktualizace jsou velmi často.

3.3.9 Struktura aplikace

Celková struktura aplikace je vidět na Obr. 3.10.

Jedná se o Python modul a jak již bylo zmíněno, struktura dodržuje vzor pro větší aplikace ve *Flasku*. Hlavní jádro aplikace je ve složce `marastats`. Dále jsou k dispozici soubory `setup.py` a `requirements.txt`, pomocí kterých lze komponenty nainstalovat celou, respektive nainstalovat pouze závislosti.

3.4 Testování

Sekce obsahuje popis testování výsledného modulu. Testování bylo rozděleno podle V-modelu [86], byly vytvořeny jednotkové testy pro kontrolu implemen-

Tabulka 3.1: Datové typy

Název sloupce	Datový typ
quiz_id	int64
quiz_name	object
quiz_type	object
quiz_semester	object
number_of_questions	int64
quiz_goal	int64
course_code	object
open_at	datetime64[ns, Europe/Prague]
close_at	datetime64[ns, Europe/Prague]
first_penalty	int64
second_penalty	int64
second_lock	int64
available_problems	int64
is_important	bool
is_tracked	bool
amount_of_users	int64
question_id	int64
user_id	int64
problem_id	int64
closed	int64
kind	object
score	int64
correct_options	int64
correct	int64
created_at	datetime64[ns, Europe/Prague]
answered_at	datetime64[ns, Europe/Prague]
duration	timedelta64[ns]
to_deadline	timedelta64[ns]
pause_from_previous_answer	timedelta64[ns]
previous_lock	int64
incorrect	int64
correct_cumsum	int64
incorrect_cumsum	int64
which_trial	int64
which_lock	int64
pause_without_lock_time	timedelta64[ns]

tace na nejnižší úrovni, integrační testy pro ověření správnosti komunikace komponenty s okolím. Tyto testy byly vytvořeny s ohledem na zadání a možnou automatizací. Dále bylo provedeno uživatelské testování, které sloužilo k ověření použitelnosti analýzy uživateli a otestování uživatelského rozhraní.

Pro implementaci a běh jednotkových a integračních testů byl použit testovací framework pytest [87]. Testy jsou umístěny ve složkách `tests_unit` a `tests_integration` v kořenové složce komponenty.

3.4.1 Jednotkové testy

Jednotkové testy se zaměřují na testování jednotlivých metod a tříd výsledné komponenty. Testují tak například jednotlivé funkce pro načtení a zpracování dat, detekci chybějících informací atd. Integrační testy v sobě zahrnují i složitější operace a volání rozhraní často obsahuje i komunikaci mnoha tříd a metod v rámci komponenty.

Testy se zaměřovali zejména na to, jestli se program chová správně:

- Odchytávání a zpracovávání výjimek od ostatních použitých modulů. Například shlukovací algoritmus nelze spustit nad prázdnými daty nebo jedním záznamem. Některé operace nejsou definovány nad prázdnou tabulkou a vyhodí výjimku.
- Detekování chybějících sloupců, položek, dat.
- Funkce pro vytvoření a ověření podpisu funguje správně.
- Funkce pro převod příchozích požadavků do objektů v Pythonu funguje správně.

Testy lze najít ve složce `tests_unit`. Lze je spustit příkazem `pytest ./tests_unit` v kořenové složce aplikace.

3.4.2 Integrační testování

Podkapitola se zabývá integračními testy, tj. jak dokáže komponenta komunikovat s okolím. Testování se tedy zaměřuje na REST API a webové stránky, které jsou komunikačním rozhraním komponenty. Testy byly vytvořeny pro základní funkcionalitu všech metod.

Integrační testy se zaměřují například na následující možné problémy:

- Funguje ověřování podpisu požadavku na vytvoření analýzy?
- Co když v požadavku na vytvoření analýzy bude chybět povinná položka?
- Dokáže si program poradit, když uživatel pošle prázdná data?

- Co se stane, když uživatel pošle data k analýze, ve kterých chybí očekávaný sloupec?
- Jsou správně nastavené stavové kódy pro HTTP komunikaci?

Například zpracování požadavku na vytvoření analýzy metodou `POST` na adresu `/analysis` bylo otestováno na to, jestli:

- Při chybějícím podpisu správně vrátí zprávu s kódem `HTTP 401 UNATHORIZED`, protože nelze ověřit původ dat.
- Při chybějícím poli v požadavku vrátí zprávu s kódem `HTTP 400 BAD REQUEST`.

Požadavek na zobrazení analýzy metodou `GET` na adresu `/analysis/<id požadavku>` bylo otestováno na to, jestli:

- Aplikace vrátí `HTTP 400 BAD REQUEST` při nečíselném identifikátoru analýzy.
- Aplikace vrátí `HTTP 202 ACCEPTED` při požadavku na zobrazení analýzy, která ještě nebyla dokončena.
- Aplikace vrátí `HTTP 404 NOT FOUND` při požadavku na zobrazení analýzy, která neexistuje.

Testy byly dělány v tomto smyslu, jsou dostupné ve složce `tests_integration`. Lze je spustit příkazem `pytest ./tests_integration` v kořenové složce aplikace.

Výsledkem těchto testů bylo nalezení několika chyb a překlepů v odpovědích na požadavky. V případě požadavku na zobrazení analýzy byla potřeba částečná změna logiky kvůli reakci na chybějící analýzu, kdy server chybně vracel `202 ACCEPTED` u analýzy, která neexistovala.

Integrační testy také daly podnět k rozšíření odpovědí na požadavky. V případě, že byl požadavek alespoň autorizován, obsahuje odpověď i přesnější popis chyby, která nastala při zpracovávání požadavku. Například pokud není komponenta schopná rozpoznat z `POST` požadavku na vytvoření analýzy identifikátor uživatele, který o ní žádal, nevrátí se pouze `HTTP 400 BAD REQUEST`, ale přidá se i informační zpráva v těle odpovědi (položka `message`), že typ této položky není pro komponentu zpracovatelný. To by mělo pomoci při integraci komponenty do dalších aplikací.

3.4.3 Uživatelské testování

Subsekce je věnována uživatelskému testování [88]. Účelem je vyzkoušet, jak se výsledek líbí potenciálním uživatelům.

Uživatelské testování má následující body:

1. Informace o testerovi — Kdo je testován, jaký má vztah k produktu, doméně.

2. Úvod — Účastníkovi testování (testerovi) se popíše situace, produkt, očekávání. Zároveň se požádá o to, aby nahlas popisoval, co právě dělá, kam kouká, o čem přemýšlí.

3. Testování podle scénářů — Účastník testování se pokusí splnit úkoly v testovacích scénářích. Zároveň se dělají poznámky, co tester dělá, na co se ptá, sepisují se jeho případné poznámky.

4. Dotazník — Nakonec jsou účastníkovi položeny doplňující otázky o produktu, jeho dojmech, názorech a nápadech.

Testování samotné začíná úvodní řečí, ve kterém se účastníkovi testu popíše situace a testovaný produkt. Dále je účastník požádán o to, aby při testování nahlas popisoval, co právě dělá, kam kouká, o čem přemýšlí. Dalším krokem je projití testovacích scénářů s účastníkem testování, tester má přitom plnit úkoly a odpovídat na otázky, přičemž se zaznamenává, co tester dělá, kam kouká, kliká apod. Nakonec se testerovi položí doplňující otázky ohledně produktu, o jeho názoru na něj, pocitech atd.

S ohledem na to, že modul ještě nebyl napojen na MARAST, je možné testovat pouze výsledek analýzy, tj. report. Testerům byl předkládám stejný scénář a otázky, výsledek analýzy byl však z kvízů z jimi vyučovaných předmětů a kvízy sami pomáhali vytvářet. Struktura výsledku analýzy tak byla stejná, obsah však byl pro testera zajímavější a motivující.

Následující podkapitola obsahuje popis scénáře a seznam otázek pro testery. Poté jsou samotné testy.

3.4.3.1 Testovací scénář

Jste garantem/vyučujícím předmětu, je konec semestru a chcete se podívat na to, jak si studenti vedli ve dvou kvízech s látkou, která dělala studentům problémy. Společný report kvízů máte právě před sebou.

1. **Jaké jsou názvy kvízů v reportu?**

2. **Kolik otázek bylo uživateli při řešení zobrazeno najednou?**

3. **Měl kvíz nastavené zablokování odpovídání za špatnou odpověď? Pokud ano, jaké to nastavení bylo?**

4. **Kolik uživatelů se zúčastnilo kvízů?**

5. **Jaká byla průměrná úspěšnost odpovědí?**

6. **Kolik bylo celkem odpovědí?**

7. **Dosáhl někdo druhého zablokování? Pokud ano, kolik bylo druhých zablokování celkem? Kolik bylo prvních?**
8. **Jaké je ID otázky s nejvyšší úspěšností? Jaká s nejnižší?**
9. **Jaký je největší počet zablokování způsobený jednou otázkou?**
10. **Podívejte se na graf s distribucí problémů podle úspěšnosti. Jak jej chápete?**
11. **Podívejte se na detailní informace o všech otázkách a seřaďte záznamy podle úspěšnosti.**
12. **Podívejte se na vizualizaci výsledků shlukování problémů. Který shluk představuje snadné a rychle zodpověditelné otázky?**
13. **Podívejte se na vizualizaci výsledků shlukování problémů pomocí tzv. *Radar/Spider plotu*. Jak jej interpretujete?**
14. **Podívejte se na výsledky hledání anomálií a odlehlých hodnot problémů. Kolik vidíte anomálií v grafu? Je vidět, že to jsou skutečně anomálie?**
15. **Podívejte se na vizualizaci výsledků shlukování uživatelů. Který shluk představuje uživatele, kteří mohou mít problém s učivem?**
16. **Podívejte se na vizualizaci výsledků shlukování uživatelů pomocí tzv. *Radar/Spider plotu*. Jak jej interpretujete?**
17. **Podívejte se na výsledky hledání anomálií a odlehlých hodnot uživatelů. Kolik vidíte anomálií v grafu? Je vidět, že to jsou skutečně anomálie?**
18. **Přejděte na log z procesu tvorby analýzy. Došlo během analýzy k nějakým problémům?**
19. **Běžte zpět na stránku s výsledkem analýzy, pokuste se najít seznam analýz ze stejných kvízů a přejděte na druhou nejnovější analýzy se stejnými kvízy.**

3.4.3.2 Otázky v dotazníku

Dotazník po konci testování podle scénáře vypadá následovně:

1. **Jak se orientujete ve zprávě?**
2. **Přijde Vám text ve zprávy dobře čitelný?**

3. **Přišla Vám nějaká informace zbytečná?**
4. **Které informace Vám přišly užitečné?**
5. **Která informace Vám chybí? Kterou byste rád doplnil?**
6. **Co říkáte na možnost interpretace výsledků shlukování?**
7. **Jak interpretujete výsledky detekce anomálií? Jsou užitečné?**
8. **Dokážete si představit využití výsledků analýzy ke zlepšení výuky?**

3.4.4 Uživatelský test č. 1

V podkapitole je popis prvního uživatelského testu.

Informace o účastníkovi:

- Účastník: TK
- Datum a čas: 4.5.2018, 14:30
- Místo: Kancelář vyučujícího.
- Role: Administrátor MARASTu, vyučující BI-ZMA, BI-PKM, BI-LIN.
- Předchozí zkušenost s komponentou: Výsledek reportu neviděl, pouze dílčí výsledky během manuální analýzy.
- Testovací data: Kvízy z předmětu BI-ZMA s ID 68 a 69. Analýza dostupná na `https://marastats.heroku.com/analysis/55`.
- Testovací prostředí: Účastník používal svůj počítač, širokoúhlá obrazovka, webový prohlížeč dle vlastního výběru (Google Chrome).

Testovací scénář:

1. **Jaké jsou názvy kvízů v reportu?**

   Názvy kvízů určil správně. Doporučil výraznější nadpisy u tabulek, větším nebo tučnějším písmem.

2. **Kolik otázek bylo uživateli při řešení zobrazeno najednou?**

   Počet určil správně pro každý kvíz. Název sloupečku v tabulce vyhovoval, nápad přidat *per problem*, aby bylo jasné, že je to při řešení jednoho problému.

3. **Měl kvíz nastavené zablokování odpovídání za špatnou odpověď? Pokud ano, jaké to nastavení bylo?**

   Již si zvykl na strukturu zprávy, tabulku s informacemi našel ihned. Opět poznamenal zvětšení nebo zvýraznění nadpisu tabulky.

4. **Kolik uživatelů se zúčastnilo kvízů?**

   Informaci neměl problém najít pro oba dva kvízy.

5. **Jaká byla průměrná úspěšnost odpovědí?**

   Našel bez problémů pro oba dva kvízy.

6. **Kolik bylo celkem odpovědí?**

   Našel bez problémů. Informace pouze za jednotlivé kvízy mu nevadila, u většího počtu kvízů by však uvítal součet.

7. **Dosáhl někdo druhého zablokování? Pokud ano, kolik bylo druhých zablokování celkem? Kolik bylo prvních?**

   Díval se na tabulku s obecnými statistikami, kde našel celkový počet zámků. Musel být upozorněn na tabulku s informacemi o samotných zámcích. Doporučil přejmenovat sloupec v tabulce s obecnými statistikami na *Total amount of locks*, aby bylo jasné, že je to celkový součet. Dále zvýraznit popisek tabulky se zámky.

8. **Jaké je ID otázky s nejvyšší úspěšností? Jaká s nejnižší?**

   V tabulkách s význačnými hodnota (nejvyšší úspěšnost, nejnižší úspěšnost atd.) se zorientoval rychle, na otázky odpověděl bez problémů. Nalezen překlep v názvu tabulky s nejnižší úspěšností.

9. **Jaký je největší počet zablokování způsobený jednou otázkou?**

   Informaci nalezl bez problémů, ocenil by informace ke všem otázkám.

10. **Podívejte se na graf s distribucí problémů podle úspěšnosti. Jak jej chápete?**

    Graf interpretoval tak, že výška obdélníků značí počet problémů s úspěšností danou na vodorovné ose, což je správně. Přišlo mu zvláštní, že aproximační křivka u 90% úspěšnosti klesá, přestože je obdélník vysoký. Do grafu by doplnil informaci o normalizované ose y (součet obsahu obdélníků je normalizován na 1). Dotázal se na význam čárek na vodorovné ose, což jsou jednotlivé otázky.

11. **Podívejte se na detailní informace o všech otázkách a seřaďte záznamy podle úspěšnosti.**

    Tabulku našel, seřazení zvládl. Tuto tabulku ocenil, již si ji vyžádal v jedné z předchozích otázek.

12. **Podívejte se na vizualizaci výsledků shlukování problémů. Který shluk představuje snadné a rychle zodpověditelné otázky?**

Výsledky shlukování neměl problém najít. Význam vizualizace pochopil rychle, měřítka os mu nedělala problém. Výsledné shluky otázek interpretoval dobře, pochopil vizualizaci. Doporučil doplnit do grafu mřížku, která by pomohla s určením přesnějších hodnot prvků v grafu. Navrhl interaktivní graf s tím, že by u prvků viděl jejich hodnoty a případně další popis. Informace o tom, že je model vyhodnocován dle skóre siluety, mu nic neřeklo, informace mu však nevadí a doporučil doplnit odkaz na vysvětlení.

13. Podívejte se na vizualizaci výsledků shlukování problémů pomocí tzv. *Radar/Spider plotu*. Jak jej interpretujete?

Využití tohoto typu grafu viděl poprvé, z počátku měl problém hlavně s interpretací os. K výsledku ale byl schopen dojít sám, pochopil interpretaci a ocenil jeho využití, zejména v případě, že e pro shlukování použije více než dvě dimenze. Ocenil by mít oba dva grafy, které vizualizují výsledek shlukování, vedle sebe.

14. Podívejte se na výsledky hledání anomálií a odlehlých hodnot problémů. Kolik vidíte anomálií v grafu? Je vidět, že to jsou skutečně anomálie?

Anomálií mezi problémy bylo příliš, z nalezených dokázal interpretovat pouze malou část problémů, které byly viditelně odděleny od zbytku. Dále mu jeden prvek přišel jako anomálie, nicméně algoritmus jej tak neurčil.

15. Podívejte se na vizualizaci výsledků shlukování uživatelů. Který shluk představuje uživatele, kteří mohou mít problém s učivem?

Výsledky shlukování neměl problém najít. Graf znal již z předchozích otázek, interpretoval jej snadno a rychle. Shluk s uživateli s možnými problémy našel rychle, zbytek shluků také interpretoval rychle a intuitivně.

16. Podívejte se na vizualizaci výsledků shlukování uživatelů pomocí tzv. *Radar/Spider plotu*. Jak jej interpretujete?

S interpretací grafu neměl problém, již měl znalost z problémů.

17. Podívejte se na výsledky hledání anomálií a odlehlých hodnot uživatelů. Kolik vidíte anomálií v grafu? Je vidět, že to jsou skutečně anomálie?

Otázky neměl problém zodpovědět, vizualizace mu přišla mnohem smysluplnější než v případě problémů. U většiny anomálií dokázal říct, proč tomu tak je, problémy byly pouze u anomálií blízko běžných prvků.

18. **Přejděte na log z procesu tvorby analýzy. Došlo během analýzy k nějakým problémům?**

    Logu si všiml již na začátku testování, našel jej rychle. Správně určil, že k chybě během analýzy nedošlo, přičemž pochopil, že by případné problémy byly uvedeny právě zde. Formát zprávy pochopil ihned, až poté uviděl text s jeho popisem.

19. **Běžte zpět na stránku s výsledkem analýzy, pokuste se najít seznam analýz ze stejných kvízů a přejděte na druhou nejnovější analýzy se stejnými kvízy.**

    Zpět šel přes tlačítko prohlížeče, odkazu si všiml již an začátku testování. Seznam analýz je nepřehledný, po přečtení popisku pochopil, jak je řazený. Datum by mělo být napsané přehledněji.

Dotazník obsahuje krátké odpovědi, protože mnoho poznámek uvedl sám účastník při testování scénáře. Odpovědi pro dotazník následují:

1. **Jak se orientujete ve zprávě?**

   Po prvotním seznámení se se strukturou dobře.

2. **Přijde Vám text ve zprávě dobře čitelný?**

   Nadpisy tabulek by to chtělo zvýraznit. U grafu s distribucí uživatelů dle úspěšnosti byl oříznutý kus popisu osy y.

3. **Přišla Vám nějaká informace zbytečná?**

   Vyloženě zbytečná nebo nepoužitelná ne.

4. **Které informace Vám přišly užitečné?**

   Vizualizace, informace o zámcích, zablokování, na otázku a uživatele.

5. **Která informace Vám chybí? Kterou byste rád doplnil?**

   Se shlukováním by se dalo hrát více, tj. zauvažovat nad dalšími možnými parametry, vhodnými pro shlukování. Navrhl využít parametr *to_deadline* a hodinu ve dni, kdy uživatel obvykle odpovídal.

6. **Co říkáte na možnost interpretace výsledků shlukování?**

   Pochopitelná a přehledná, byť *Radar plot* má využití hlavně při shlukování nad více dimenzemi.

7. **Jak interpretujete výsledky detekce anomálií? Jsou užitečné?**

   U problémů to nedávalo moc smysl, u uživatelů to bylo naopak dobré a pochopitelné.

8. **Dokážete si představit využití výsledků analýzy ke zlepšení výuky?**

   Jedná se o dobrý základ, určitě se v tom dá pokračovat. Využít lze informace ze shlukování, statistiky zablokování, doby odpovídání apod.

3.4.5 Uživatelský test č. 2

V podkapitole je popis druhého uživatelského testu.

Informace o účastníkovi:

- Účastník: KK
- Datum a čas: 7.5.2018, 13:30
- Místo: Kancelář vyučujícího.
- Role: Vyučující BI-LIN, MI-MPI.
- Předchozí zkušenost s komponentou: Žádná.
- Testovací data: Kvízy z předmětu MI-MPI s ID 44 a 58. Analýza dostupná na `https://marastats.heroku.com/analysis/70`.
- Testovací prostředí: Účastník používal svůj počítač, širokoúhlá obrazovka, webový prohlížeč dle vlastního výběru (Google Chrome).

Testovací scénář:

1. **Jaké jsou názvy kvízů v reportu?**

   Názvy kvízů určil správně.

2. **Kolik otázek bylo uživateli při řešení zobrazeno najednou?**

   Počet určil správně pro oba kvízy.

3. **Měl kvíz nastavené zablokování odpovídání za špatnou odpověď? Pokud ano, jaké to nastavení bylo?**

   Informaci našel a určil správně, jednotky ve vteřinách mu vyhovovaly, uvádět vyšší hodnoty v minutách by ocenil.

4. **Kolik uživatelů se zúčastnilo kvízů?**

   Informaci našel správně, podle názvu tabulky.

5. **Jaká byla průměrná úspěšnost odpovědí?**

   Informaci našel ihned, společně s počtem účastníků.

6. **Kolik bylo celkem odpovědí?**

   Našel bez problémů.

7. **Dosáhl někdo druhého zablokování? Pokud ano, kolik bylo druhých zablokování celkem? Kolik bylo prvních?**

   Nejprve si všiml celkového počtu zablokování v tabulce se shrnutím, poté našel tabulku s detailními informacemi.

8. **Jaké je ID otázky s nejvyšší úspěšností? Jaká s nejnižší?**

   Našel bez problémů.

9. **Jaký je největší počet zablokování způsobený jednou otázkou?**

   Našel bez problémů, orientoval se podle názvů tabulek.

10. **Podívejte se na graf s distribucí problémů podle úspěšnosti. Jak jej chápete?**

    Graf se mu příliš nezdál. Popisek u osy y matoucí, místo distribuce doporučil frekvenci/obsah/poměr. Počet sloupečků zvýšit. Aproximace křivkou nepřesná.

11. **Podívejte se na detailní informace o všech otázkách a seřaďte záznamy podle úspěšnosti.**

    Detaily našel až po navedení, odkaz obarvit nebo jinak zvýraznit. Doporučil dodělat odkazy na otázky na MARAST.

12. **Podívejte se na vizualizaci výsledků shlukování problémů. Který shluk představuje snadné a rychle zodpověditelné otázky?**

    Shlukování a vizualizaci pochopil, s interpretací shluků neměl problém. Doporučil shlukování nad více vlastnostmi (čas do uzávěrky, skóre). Informace o použitých algoritmech považoval za použitelné, parametry algoritmů a skóre siluety zná.

13. **Podívejte se na vizualizaci výsledků shlukování problémů pomocí tzv. *Radar/Spider plotu*. Jak jej interpretujete?**

    Graf již znal, neměl problém s interpretací.

14. **Podívejte se na výsledky hledání anomálií a odlehlých hodnot problémů. Kolik vidíte anomálií v grafu? Je vidět, že to jsou skutečně anomálie?**

    Výsledky mu přišly zajímavější než u shlukování.

15. **Podívejte se na vizualizaci výsledků shlukování uživatelů. Který shluk představuje uživatele, kteří mohou mít problém s učivem?**

    S interpretací nebyl problém.

16. **Podívejte se na vizualizaci výsledků shlukování uživatelů pomocí tzv. *Radar/Spider plotu*. Jak jej interpretujete?**

    Graf již znal, s interpretací neměl problém.

17. **Podívejte se na výsledky hledání anomálií a odlehlých hodnot uživatelů. Kolik vidíte anomálií v grafu? Je vidět, že to jsou skutečně anomálie?**

    Otázku zodpověděl správně.

18. **Přejděte na log z procesu tvorby analýzy. Došlo během analýzy k nějakým problémům?**

    Odkazu si všiml již na začátku kvízu. Správně zodpověděl otázku. Pochopil, že v případě problémů by se zde objevili informace z průběhu analýzy.

19. **Běžte zpět na stránku s výsledkem analýzy, pokuste se najít seznam analýz ze stejných kvízů a přejděte na druhou nejnovější analýzy se stejnými kvízy.**

    Odkazu si všiml již na začátku, nebyl problém najít. Doporučil lepší formát data, resp. přehlednější seznam.

Odpovědi pro dotazník následují:

1. **Jak se orientujete ve zprávě?**

   Nadpisy nevýrazné. Příliš široké zobrazení, byť responzivní. Omezení desetinných míst. Jednotky uvést přímo v buňce, ne v záhlaví sloupce. Nápad rozdělit jednotlivé sekce do záložek. Grafy a vizualizace by byly vhodné interaktivní.

2. **Přijde Vám text ve zprávě dobře čitelný?**

   Odkazy na detaily zvětšit.

3. **Přišla Vám nějaká informace zbytečná?**

   Vyloženě zbytečná ne.

4. **Které informace Vám přišly užitečné?**

   Graf s distribucí, statistiky, použité metody shlukování a detekce anomálií včetně jejich parametrů a hodnocení.

5. **Která informace Vám chybí? Kterou byste rád doplnil?**

   Shlukování nad více parametry, časové řady, vztah správnosti odpovědi k uzávěrce kvízu, vztah počtu odpovědí k uzávěrce kvízů.

6. **Co říkáte na možnost interpretace výsledků shlukování?**

   Interpretace pro 2 dimenze OK.

7. Jak interpretujete výsledky detekce anomálií? Jsou užitečné?

Interpretace je OK, chtělo by to více parametrů.

8. Dokážete si představit využití výsledků analýzy ke zlepšení výuky?

Ano, informace a statistiky o problémech a uživatelích mohou být užitečné.

3.4.6 Vyhodnocení uživatelského testování

Podkapitola je věnována shrnutí uživatelského testování.

Oba účastníci měli podobný názor — aplikace obsahuje základní informace, statistiky, shlukování a detekci anomálií a je na čem stavět. Možnosti dalších rozšíření a vylepšení jsou široké, při zapracovávání poznámek byly upřednostněny opravy a zpřehlednění aktuální zprávy. Přestože jsou některé nápady na analýzu a vizualizace zpracovány v manuální analýze, do výsledku nebyly zapracovány, protože automatizace analytické metody vyžaduje dostatek času na implementaci a ladění, zajištění robustnosti vůči datům apod. Některé návrhy navíc vyžadují hlubší analýzu a jiný pohled na doménu.

Přehled chyb je vidět v Tab. 3.2.

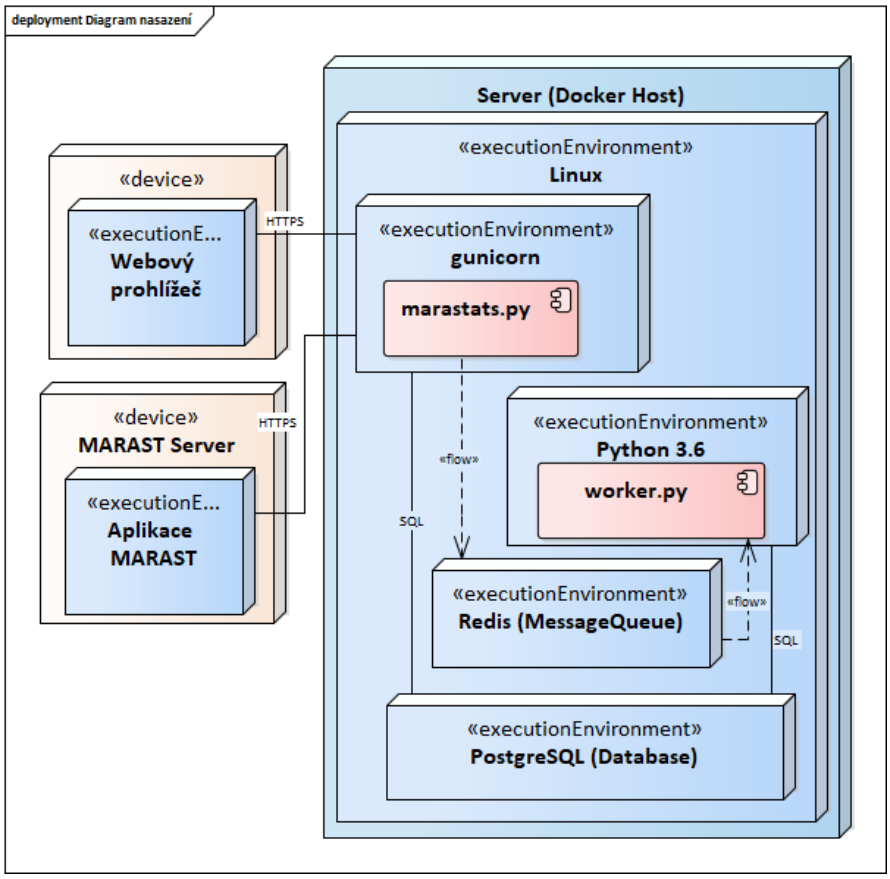
Tabulka 3.2: Návrhy a nalezené chyby při uživatelském testování, jejich závažnost (1 nejnižší, 5 nejvyšší) a stav

Problém	Závažnost	Stav
Výraznější nadpisy tabulek	5	Opraveno
Přidat grafům pomocnou mřížku	4	Opraveno
Popis souřadnice osy y u grafů s distribucemi	3	Opraveno
Přidat k názvu sloupečku *Number of questions in batch* nápis *per problem*	1	Ponecháno
Přidat celkový součet zámků ze všech kvízů	1	Ponecháno
Přidání slova *Total* ke sloupcům s celkovými statistikami (celkový počet odpovědí apod.)	2	Opraveno
Chyba v názvu tabulky s problémy s nejnižší úspěšností	2	Opraveno
Doplnit popis významu čárek na ose x v grafu distribuce	2	Opraveno
Interaktivní vizualizace	3	Ponecháno
Doplnit odkaz na vysvětlení skóre siluety	2	Opraveno
Umístění grafů s interpretací shlukování vedle sebe	4	Opraveno
Obtížná interpretace nalezených anomálií	3	Ponecháno
Změna formátu data v logu	1	Ponecháno
Aplikace shlukování a detekce anomálií nad jinými parametry	7	Ponecháno
Ve sloupci s časovými údaji upravit formát, vhodně převádět na minuty	1	Ponecháno
Ve sloupci s časovými údaji oříznout desetinná místa	1	Ponecháno
Zvýraznit odkazy na detailní informace	5	Opraveno
Omezit šířku zobrazení obsahu stránky	2	Opraveno
Vložit jednotky přímo do buněk tabulky	1	Ponecháno

3.5 Nasazení

Tato sekce se zabývá nasazením aplikace.

Diagram nasazení je vidět na Obr. 3.11. V případě nutnosti lze nasazení doplnit o *nginx* [89], jedná se o robustnější webový server. *nginx* by se staral o většinu výpočetně náročné práce související s HTTP komunikací, na kterou je určený. Nicméně *gunicorn* pro aktuální nasazení stačí.



Obrázek 3.11: Diagram nasazení komponenty

Dále v kapitole je popis nasazení na vybraných platformách.

3.5.1 Nasazení na cílovou platformu

Cílovou platformou je linuxový server, který bude využívat *Docker* [56].

Pro vytvoření *Docker Image* byl vytvořen skript, tzv. *Dockerfile*. Server je založen na distribuci linuxové distribuci *Alpine* [90].

Vedle aplikace je také nutné mít databázi *PostgreSQL* pro persistenci dat a *Redis* jako zprostředkovatele zpráv. Pro obě technologie existují jsou již vytvořeny *image* a lze je stáhnout z oficiální knihovny *Docker Hub* [91].

Výsledný *image* s aplikací má přibližně 1 GB, přestože je zvolená distribuce Linuxu minimalistická. Velikost narostla kvůli mnoha knihovnám, které je potřeba stáhnout. S ohledem na velikosti a případné aktualizace není žádný *image* přiložen na CD, pouze návody a odkazy, jak *image* sestavit a jak sehnat databáze. Výsledný soubor `Dockerfile` (dostupný na CD) se stará o sestavení *image*.

Ze způsobu nasazení použitím *Dockeru* pak vyplývají další možnost zabezpečení, lze omezit přístup ke komponentě pouze na vybrané zdroje.

3.5.2 Nasazení na *Heroku*

Pro demonstraci nasazení ukázkové aplikace bylo použito *Heroku* [92]. *Heroku* je cloudová platforma, poskytující tzv. *PaaS* (*Platform as a Service)* [93]. Služba byla vybrána s ohledem na její dobrou podporu Python technologií, cenu, přístupnost a jednoduchost nasazení aplikací. Je možné nasadit jednu aplikaci zdarma, s omezeným výkonem (1vCPU, 512MB RAM, 1 webová aplikace a 1 proces/aplikace na pozadí) a omezenou dostupností (aplikace usne po 30 minutách bez aktivity, přístup po omezenou dobu, omezený počet požadavků).

Zvláštností této platformy je použitý systém souborů. Používá tzv. *Ephemeral File System* [94, 95], což znamená, že aplikace běží na svých virtuálních serverech a ty mohou být kdykoliv zrestartovány a tím navráceny do původního (čistého) stavu. Přičemž vše, co se v tomto systému souborů vyskytuje, bude smazáno. V důsledku tak není možné využít disk k jakékoliv persistenci dat, např. pro *SQLite* databázi nebo pro vygenerované zprávy z analýz. Co se týče zdrojového kódu a dalších souborů aplikace, ty jsou uloženy v GIT repozitáři a při restartu se načtou. Pro úplnost — možnost ukládat data do tohoto *GIT* repozitáře sice je, ale není k tomu určen. V základním nastavení *push* do repozitáře navíc restartuje server a aplikaci.

Pro nasazení bylo potřeba založit účet, vytvořit profil aplikace a nastavit prostředí. Nastavení probíhá pomocí *Heroku CLI* [96], což v základu funguje jako rozšíření příkazové řádky v operačním systému. Způsob nasazení a sestavení aplikace se definuje v souboru *Procfile*, který lze vidět jako velmi zjednodušený *Dockerfile*. O některá nastavení proměnných prostředí (např. adresa a port databáze) se stará sama platforma. Služby jako databáze nebo fronta požadavků se přidávají pomocí přídavků (tzv. *Addons*) [97], náš modul využívá *PostgreSQL* jako databázi a *Redis* pro frontu požadavků. Výběr byl omezen na bezplatné přídavky, a i ty mají omezené možnosti, zejména co se týče výkonu. Pro demonstraci to však postačí.

Ukázková aplikace je nasazena na `https://marastats.heroku.com`, jak vypadá výstup analýzy (report) lze vidět např. na `https://marastats.heroku.com/analysis/55`.

3.6 Dokumentace

Sekce obsahuje popis dokumentace, jakým způsobem byla napsána, kde ji najít, jak vygenerovat atd.

Dokumentace aplikace včetně komentářů ve zdrojových kódech je psána v anglickém jazyce. V dokumentaci však nejsou podrobně vysvětleny kroky analýzy a jejich zdůvodnění tak, jako tomu je v teoretické části této práce.

Dokumentace aplikace obsahuje informace pro člověka, který by chtěl ji nasadit a používat, komentáře zase obsahují co program nebo metoda dělá. Detailní vysvětlení teorie je však pouze v tomto textu, teorie nebyla přeložena a vložena do samotné dokumentace modulu.

3.6.1 Dokumentace modulu

K vytvoření dokumentace byl použit balíček *Sphinx* [98, 99], který je použitý i např. balíčkem *Flask* [57]. *Sphinx* umožňuje generovat dokumentaci ze souborů ve formátu *reStructuredText* (zkráceně *reST*, *rst*) [100]. Z těch se generuje dokumentace ve formátu HTML, tj. ve formátu vhodném pro webové prohlížeče.

Zdrojové soubory dokumentace jsou dostupné v kořenové složce aplikace, ve složce `docs`, jak je vidět na Obr. 3.12. V případě potřeby ji lze vygenerovat pomocí příkazu `make html` ve složce `docs`, dokumentace se v základním nastavení vytvoří ve složce `_build/html`. Dokumentace je na přiloženém CD již v této složce vytvořená.

Dokumentace je rozdělena do několika souborů podle tématu, obsahuje popis aplikace, popis nasazení, strukturu aplikace, popis formátu přijímaných dat, licenci aj.

```
root ..... kořenová složka se všemi zdroji aplikace
├── docs ..... složka s dokumentací projektu
├── marastats ..... hlavní složka se zdrojovými kódy aplikace
└── .....
```

Obrázek 3.12: Umístění dokumentace modulu v adresářové struktuře

3.6.2 Dokumentace zdrojového kódu

Zdrojový kód je zdokumentován a okomentován. Dokumentace k funkcím, modulům a třídám byla napsána ve formátu Google komentářů [101], použití tohoto formátu umožňuje vygenerovat dokumentaci zdrojového kódu. Základní formát komentářů například u metod je následující:

1. Krátký popis, co metoda dělá.
2. Nepovinně dlouhý popis metody.
3. Seznam argumentů.
4. Seznam návratových hodnot.
5. Nepovinně seznam výjimek.

V případě komentářů u modulů nebo tříd je logika komentování podobná. Ukázka takového komentáře z práce je vidět na Obr. 3.13.

```
def check_dataframe_columns_for_null(dataframe, definition):
    """Check columns of dataframe for null values. Raises Exception if null
        value found in a column that should not have null values.

    Args:
        dataframe (pandas.Dataframe): Dataframe with columns to be checked.
        definition (list): List of column names that are expected in
            dataframe.
    Returns:
        None
    Raises:
        ValueError: If null values is present in column where it should not
            be.

    """
    for column in definition:
        if dataframe[column].isnull().sum() > 0:
            raise ValueError("There is NULL value in column called \"" + str(
                column) + "\".")
```

Obrázek 3.13: Ukázka komentáře metody v kódu ve formátu Google

Kapitola 4

Výsledky, zpětná vazba

V kapitole je shrnutí vývoje, funkcí aplikace, testování a jeho vyhodnocení.

4.1 Manuální analýza dat

Pro manuální analýzu byly vytvořeny *Jupyter notebooky*, jejichž počet narostl na 20. Obsahují načtení a předzpracování dat, informace a statistiky o dodaném exportu dat z MARASTu, vizualizace, využití několika algoritmů shlukování a detekce anomálií. Dále jsou z nich vybrány zajímavé informace a vloženy do teoretické části této práce. Pro komponentu jsou pak zvoleny metody, které jsou robustní vůči hodnotám v datech a poskytují smysluplné výsledky bez dozoru člověka během analýzy.

4.2 Výsledný modul

Výsledný modul komunikuje s okolím prostřednictvím webové aplikace, která poskytuje REST API pro vytvoření a smazání analýzy a výpis všech analýz v systému. Aplikace deleguje výpočetně náročné provedení analýzy na proces běžící v pozadí, se kterým komunikuje přes zprostředkovatele zpráv. Proces na pozadí se o tento úkol postará a výsledek uloží do databáze. Webová aplikace dále poskytuje přístup k výsledkům analýzy a logu jako na webové stránky.

4.2.1 Případy užití

Z pohledu případů užití je funkcionalita aplikace následující:

- Vytvoření analýzy kvízů — Možné prostřednictvím REST API.
- Smazání analýzy — Umožněno prostřednictvím REST API.
- Výpis všech analýz v systému — Funkci poskytuje REST API.

- Výpis všech analýz stejné skupiny kvízů — Zobrazení prostřednictvím webové stránky, přístup přímo přes adresu stránku nebo odkaz ve výsledku analýzy.
- Zobrazení výsledků analýzy — Uživatel si může zobrazit výsledek analýzy prostřednictvím webové stránky.
- Vytištění výsledků analýzy — Umožněno funkcí tisku ve webovém prohlížeči.
- Zobrazení logu z procesu analýzy — Uživatel může přejít na stránku s logem buď přímo přes adresu, nebo přes odkaz na stránce s výsledkem analýzy.

4.2.2 Vzhled výsledku analýzy

Analýza je vytvořena jako webová stránka. K vizualizaci tabulek je použita JavaScript knihovna *DataTables*, vzhled je vidět na Obr. 4.1. O další vizualizace se pak stará Python modul *seaborn*, příklad je na Obr. 4.2.

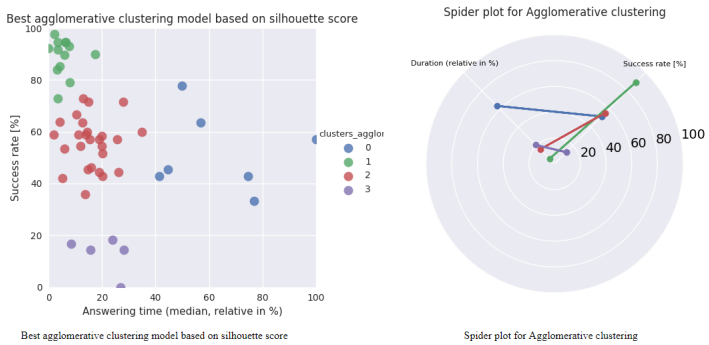
Show 10 entries Search:

General information about quizes

Quiz ID	Quiz name	Course code	Quiz semester	Open at	Close at
68	Procvičování: Integrace	BI-ZMA	B161	2017-01-13 18:40:00+01:00	NaT
69	Časté chyby	BI-ZMA	B161	2017-01-13 18:42:00+01:00	NaT

Showing 1 to 2 of 2 entries Previous 1 Next

Obrázek 4.1: Vzhled tabulky ve výsledku analýzy



Obrázek 4.2: Vizualizace výsledků analýzy

4.2.3 Technická stránka

Z architektury je rozdělení komponenty na webovou aplikaci a proces na pozadí výhodné hlavně díky neblokování webové aplikace náročnými výpočty.

Při testování na *Heroku* však byla zjištěna relativně vysoká náročnost na operační paměť, kdy dostupných 512MB RAM nebylo dostatečné pro běh více obsáhlejších analýz najednou a aplikace tak byla násilně ukončena kvůli překročení limitu. Příčinou paměťové náročnosti je kombinace následujících:

- Výsledky operací v *pandas* jsou často hluboké kopie zdrojových tabulek nebo jejich částí. Tento problém již byl částečně zmírněn prováděním operací nad zdrojovými daty (tzv. *inplace*), to ale není možné nastavit u všech operací. Další řešení by bylo v jemné (detailní) optimalizaci procesu analýzy, např. explicitním mazání tabulek v momentě, kdy už nejsou k analýze potřeba.

- Množství a velikost vizualizací. Obrázků se generuje relativně mnoho (přes 10) a zejména obrázek s vizualizací skóre pro každou otázku je paměťově náročný. Řešením může být generování méně obrázků nebo v nižší kvalitě. Také by bylo možné předělat způsob vizualizace, kdy by se klientovi posílala podstatně menší textová zdrojová data a obrázek by se tak tvořil až u klienta. To by ale vyžadovalo vytvoření klientské aplikace.

Na produkčním serveru bych proto doporučil mít více paměti RAM. Podle situace na *Heroku* bych doporučil mít vyhrazeno pro webovou aplikaci a jeden proces na pozadí alespoň 1GB RAM, 2GB RAM bych pak odhadoval na zcela dostatečné. Další škálování aplikace (více procesů na pozadí) nebo posílání vysokých objemů dat (statisíce záznamů) by si pak vyžádalo další zdroje.

4.3 Výsledky testování a zpětná vazba

Během testování bylo nalezeno několik chyb a nedostatků. Softwarové testy komponenty pomohli k ověření funkcionality, uživatelské pak k ověření využitelnosti.

Softwarové testy pomohly k identifikaci problémů v rozhraní a v několika případech neodchycené výjimky z použitého modulu (obvykle z metody modulu *pandas* nebo *scikit-learn*). Tyto chyby byly opraveny. Dále testy pomohly k ujištění, že základní funkcionalita programu je zajištěna.

Uživatelské testování mělo 2 účastníky, oba dva jsou uživatelé MARASTu a správci kvízů. Z jejich připomínek byla dána přednost opravám stávajícího před rozšiřováním. Jako uživatelé MARASTu měli mnoho nápadů na vylepšení a měli také shodný názor, že je na čem dále stavět.

Závěr

Cílem práce bylo seznámit se s daty ze systému MARAST, provést jejich analýzu a interpretovat její výsledky. Dále návrh a implementace komponenty ve formě Python modulu, který by byl schopen provést analýzu budoucích kvízů z MARASTu automaticky. I přes některé problémy lze zadání považovat za splněné a výsledky lze aplikovat dále. Komponenta i předzpracování dat mohou být použity k dalšímu rozvoji a výzkumu v této oblasti. Způsob, kterým bylo dosaženo splnění zadání, je po jednotlivých krocích uveden níže.

V první části práce jsem se zabýval studiem domény a metodami dolování dat ve vzdělávání. Doména je tvořena systémem MARAST a způsobem testování studentů. Rešerše se zabývala hlavně výzkumem v dané oblasti a aplikacemi podobného zaměření, tj. *educational data mining*. Z provedené rešerše jsem vycházel i v dalších částech práce. Například při výběru reprezentativních vlastností studenta z dodaných dat, kdy k reprezentaci studijních výkonů studentů byly použity jejich průměrné úspěšnosti odpovídání a doby odpovídání. Další použití poznatků z rešerše bylo při výběru vhodných algoritmů shlukování, nejčastěji používanými jsou *k-means* za nehierarchické algoritmy a aglomerativní shlukování za hierarchické algoritmy.

Další fází práce byl rozbor a analýza poskytnutého exportu dat ze systému MARAST. V exportu dat bylo celkem 129 kvízů, k nim bylo v prvním exportu přes 500 000 odpovědí. V druhém exportu odpovědí bylo celkem 524 131 záznamů a odpovědi byly navíc rozšířeny o několik ukazatelů. Tato data bylo nutné pochopit, správně předzpracovat, aplikovat analytické metody a výsledky poté interpretovat. V tomto kroku jsem se řídil metodikou CRISP-DM, která se používá při dolování dat a získávání znalostí.

Dalším krokem bylo navržení komponenty, která by byla schopná provádět analýzy automaticky. Krok byl rozdělen na menší části, sledující vývoj softwarového produktu. Nejprve byla vytvořena analýza požadované komponenty, tedy sesbírány požadavky a případy užití. Na základě analýzy jsem vypracoval návrh aplikace. S přihlédnutím k potenciálně výpočetně náročné analýze dat jsem se uchýlil k rozdělení aplikace na dvě části. První část fun-

guje jako webová aplikace, zpracovává příchozí požadavky a předává data k analýze druhé části aplikace. Druhá část pak tato data přijímá a provádí veškeré kroky analýzy, mj. předzpracování dat, náročné analytické výpočty a vygenerování výsledné zprávy. Součástí tohoto kroku byl také návrh vzhledu výsledné zprávy z analýzy a jeden z nejtěžších kroků vůbec — automatizace analýzy a nastavení modelů tak, aby byly schopny podávat rozumné výsledky bez lidského zásahu. To se na základě poznatků z manuální analýzy podařilo relativně dobře.

Krok implementace pak vycházel z větší části z návrhu, komponenta však musela být na několika místech upravena, ať už z důvodu aktualizace použitých modulů nebo změn požadavků.

Testování pak bylo provedeno jak softwarové pomocí jednotkových a integračních testů, tak uživatelské se správci a tvůrci kvízů v systému MARAST. Jednotkové a integrační testy se zaměřovaly zejména na pomocné funkce, správné ošetřování krajních případů a testování komunikačního rozhraní aplikace. Při uživatelském testování se objevovaly připomínky zejména k přehlednosti zprávy a velikosti textu. Uživatelé dále neviděli využití některých informací obsažených ve výsledku, jako je například způsob hodnocení kvality algoritmu shlukování, nicméně přítomnost této informace jim nevadila. Z hlediska interpretace výsledků bylo pro uživatele problematické interpretovat výsledek detekce anomálií v otázkách.

Komponentu se podařilo nasadit na cloudové platformě *Heroku*, kde i přes omezené výpočetní prostředky je schopna zpracovávat požadavky na analýzu dat. Pro budoucí nasazení byl připraven *Dockerfile* a jeho nasazení lokálně otestováno.

Hlavním přínosem práce je tak vytvoření komponenty, která je schopna analyzovat data z MARASTu. Tato analýza pak umožňuje vyučujícím lépe pochopit výsledky studentů na MARASTu, od poskytnutí základních statistik až po detekci problematických otázek, studentů s potenciálními studijními problémy apod. Dalším přínosem je popis a předzpracování dat z MARASTu, což může pomoci k rychlejšímu a snazšímu pochopení domény případným pokračovatelům. Z hlediska implementace byly dodržovány zvyky pro tvorbu větších aplikací ve *Flasku* a celou aplikaci lze distribuovat jako standardní balíček v Pythonu, rozšiřování aplikace by tak nemělo působit problémy.

Návrhy a doporučení

Práce se dotýkala mnoha témat a možnosti dalšího výzkumu a rozvoje jsou tak velmi široké.

Analýzu dat lze dále rozšiřovat a zkoumat. Lze vyzkoušet aplikaci jiných modelů a algoritmů, hlouběji analyzovat časové řady, pracovat s čerstvými daty apod. Rozšíření dat o znění otázek by mohlo přinést lepší zpětnou vazbu. Otázky by mohly být klasifikovány na základě jejich znění (možné použití

zpracování přirozeného jazyka, příp. analýza matematických zápisů) a v kombinaci s úspěšností odpovědí na dané otázky by se dalo určit, která látka je problematická.

Z pohledu architektury aplikace lze uvažovat o vylepšeních týkajících se dynamického obsahu výsledku analýzy ve smyslu interaktivních grafů a tabulek. Aktuální verze pracovala s tím, že výsledek analýzy bude pouze statická stránka, případně PDF. Nabízí se použití JavaScriptu nebo jiné technologie pro vytvoření interaktivních grafů, např. u grafů s výsledky shlukování by se u jednotlivých prvků mohly zobrazovat jejich vlastnosti při přejetím prvku myší.

Aktuální aplikace byla od počátku zamýšlena a navržena pouze pro jeden jediný výpočetní uzel, díky vybraným technologiím lze některé části jako je databáze nebo zprostředkovatel zpráv delegovat na jiný výpočetní uzel. Pokud by však stávající infrastruktura přestala dostačovat, lze se poohlédnout po distribuovaných řešeních. Takovým řešením je *Apache Spark* [102] a pro implementaci využít *pySpark* [103]. Větší část práce by jen stačilo přepsat pro rozhraní *pySpark*, kvůli podpoře distribuovaných dat a výpočtů. Zmíněné technologie jsou ostatně vyučovány v předmětu MI-DDM [104] na FIT ČVUT a pro zájemce o tyto technologie by se mohlo jednat o zajímavou (semestrální/bakalářskou) práci. Infrastruktura by pak samozřejmě musela být rozšířena o další výpočetní uzly.

Literatura

[1] Contributors of Wikipedia: *Cross Industry Standard Process for Data Mining*. [online]. [cit. 2018-3-30]. Dostupné z: https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining

[2] Domingos P.: *A Few Useful Things to Know about Machine Learning*. 2012. Dostupné z: https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

[3] Kurgan, L.; Musilek, P.: A survey of Knowledge Discovery and Data Mining process models. ročník 21, 03 2006: s. 1–24.

[4] Kordík P., Jiřina M.: *Lecture 1: Introduction, KDDM, CRISP-DM, DM software*. 2011, [online]. [cit. 2018-3-30]. Dostupné z: https://edux.fit.cvut.cz/oppa/MI-PDD/prednasky/l1-intro-crisp-viz.pdf

[5] Smart Vision Europe: *What is the CRISP-DM methodology?* [online]. [cit. 2018-3-30]. Dostupné z: https://www.sv-europe.com/crisp-dm-methodology/

[6] Dutt A., Ismail M. A., Herawan T.: *A Systematic Review on Educational Data Mining*. 2017. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7820050

[7] Papoušek J., et al.: *An Analysis of Response Times in Adaptive Practice of Geography Facts*. 2015. Dostupné z: https://www.fi.muni.cz/~xpelanek/publications/poster-response-times.pdf

[8] Nižnan J., Papoušek J., Pelánek R.: *Using Problem Solving Times and Expert Opinion to Detect Skills*. 2014. Dostupné z: https://www.fi.muni.cz/~xpelanek/publications/edm2014-expert-data.pdf

[9] Řihák J., Pelánek R.: *Choosing a Student Model for a Real World Application*. 2016. Dostupné z: https://www.fi.muni.cz/~xpelanek/publications/its-matmat.pdf

[10] Jarušek P., Pelánek R.: *Analýza obtížnosti logických úloh na základě modelů lidského chování*. 2010. Dostupné z: http://www.fi.muni.cz/~xpelanek/publications/kuz10.pdf

[11] Pelánek R., Řihák J.: *Experimental Analysis of Mastery Learning Criteria*. 2017. Dostupné z: https://www.fi.muni.cz/~xpelanek/publications/mastery-detection.pdf

[12] Papoušek J., Pelánek R.: *Should We Give Learners Control Over Item Difficulty?* 2017. Dostupné z: https://www.fi.muni.cz/~xpelanek/publications/difficulty-adjustment.pdf

[13] Friedjungová, M.: *Predikce studijních výsledků studentů bakalářského programu Informatika FIT ČVUT*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2016.

[14] Hrubá, E.: *Analýza výsledků absolventů středních škol na VŠ*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2014.

[15] Nový, O.: *Doporučovací systém pro výběr volitelných předmětů*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2017.

[16] Kuznetsov S., et al.: *Reducing cold start problems in educational recommender systems*. 2016. Dostupné z: http://ieeexplore.ieee.org/document/7727600

[17] Microsoft: *Microsoft Excel*. [online]. [cit. 2018-3-30]. Dostupné z: https://products.office.com/cs-cz/excel

[18] The Document Foundation: *LibreOffice Calc*. [online]. [cit. 2018-3-30]. Dostupné z: https://cs.libreoffice.org

[19] SAS Institute Inc.: *SAS*. [online]. [cit. 2018-3-30]. Dostupné z: https://www.sas.com/en_us/home.html

[20] RapidMiner, Inc.: *RapidMiner*. [online]. [cit. 2018-3-30]. Dostupné z: https://rapidminer.com

[21] Python Software Foundation: *Python*. [online]. [cit. 2017-12-2]. Dostupné z: https://www.python.org/

[22] SciPy developers: *PyData*. [online]. [cit. 2018-1-20]. Dostupné z: https://scipy.org

[23] PyData.org: *PyData*. [online]. [cit. 2018-1-21]. Dostupné z: https://pydata.org

[24] Pandas Core Team, et al.: *Pandas*. [online]. [cit. 2018-1-20]. Dostupné z: http://pandas.pydata.org

[25] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; aj.: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, ročník 12, 2011: s. 2825–2830.

[26] Maini V.: *Machine Learning for Humans, Part 3: Unsupervised Learning*. [online]. [cit. 2017-04-15]. Dostupné z: https://medium.com/machine-learning-for-humans/unsupervised-learning-f45587588294

[27] scikit-learn developers: *2.3. Clustering*. [online]. [cit. 2018-3-30]. Dostupné z: http://scikit-learn.org/stable/modules/clustering.html

[28] scikit-learn developers: *MinMaxScaler*. [online]. [cit. 2017-04-15]. Dostupné z: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[29] scikit-learn developers: *StandardScaler*. [online]. [cit. 2017-04-15]. Dostupné z: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[30] scikit-learn developers: *2.3.9.5. Silhouette Coefficient*. [online]. [cit. 2018-3-30]. Dostupné z: http://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient

[31] Caliński, T.; Harabasz, J.: A dendrite method for cluster analysis. *Communications in Statistics*, ročník 3, č. 1, 1974: s. 1–27, https://www.tandfonline.com/doi/pdf/10.1080/03610927408827101. Dostupné z: https://www.tandfonline.com/doi/abs/10.1080/03610927408827101

[32] scikit-learn developers: *2.3.9.6. Calinski-Harabaz Index*. [online]. [cit. 2018-3-30]. Dostupné z: http://scikit-learn.org/stable/modules/clustering.html#calinski-harabaz-index

[33] Institut biostatistiky a analýz Masarykovy univerzity: *Matematická biologie: e-learningová učebnice: Metoda k-průměrů*. [online]. [cit. 2018-3-30]. Dostupné z: http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--shlukova-analyza--shlukova-nehierarchicka-analyza--metoda-k-prumeru

[34] Bishop, C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006, ISBN 0387310738.

[35] Institut biostatistiky a analýz Masarykovy univerzity: *Matematická biologie: e-learningová učebnice: Hierarchické aglomerativní shlukování*. [online]. [cit. 2018-3-30]. Dostupné z: http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--shlukova-analyza--shlukova-hierarchicka-analyza--hierarchicke-shlukovani--hierarchicke-aglomerativni-shlukovani

[36] scikit-learn developers: *Euclidean distances*. [online]. [cit. 2017-04-15]. Dostupné z: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.euclidean_distances.html

[37] Hastie, T.; Tibshirani, R.; Friedman, J.: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics, Springer, 2009, ISBN 9780387848846. Dostupné z: https://books.google.cz/books?id=eBSgoAEACAAJ

[38] Arthur, D.; Vassilvitskii, S.: K-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, ISBN 978-0-898716-24-5, s. 1027–1035. Dostupné z: http://dl.acm.org/citation.cfm?id=1283383.1283494

[39] scikit-learn developers: *2.3.2.1. Mini Batch K-Means*. [online]. [cit. 2017-04-16]. Dostupné z: http://scikit-learn.org/stable/modules/clustering.html#mini-batch-k-means

[40] Mahajan, M.; Nimbhorkar, P.; Varadarajan, K.: The planar k-means problem is NP-hard. *Theoretical Computer Science*, ročník 442, 2012: s. 13 – 21, ISSN 0304-3975, doi:https://doi.org/10.1016/j.tcs.2010.05.034, special Issue on the Workshop on Algorithms and Computation (WALCOM 2009). Dostupné z: http://www.sciencedirect.com/science/article/pii/S0304397510003269

[41] Kovalyov, M. Y.; Pesch, E.: A generic approach to proving NP-hardness of partition type problems. *Discrete Applied Mathematics*, ročník 158, č. 17, 2010: s. 1908 – 1912, ISSN 0166-218X, doi:https://doi.org/10.1016/j.dam.2010.08.001. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0166218X10002647

[42] Comaniciu, D.; Meer, P.: Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 24, 2002: s. 603–619.

[43] Mubaris NK: *K-Means Clustering in Python*. [online]. [cit. 2017-04-15]. Dostupné z: `https://mubaris.com/2017/10/01/kmeans-clustering-in-python/`

[44] scikit-learn developers: *2.7. Novelty and Outlier Detection*. [online]. [cit. 2018-3-30]. Dostupné z: `http://scikit-learn.org/stable/modules/outlier_detection.html`

[45] scikit-learn developers: *Elliptic Envelope*. [online]. [cit. 2017-04-15]. Dostupné z: `http://scikit-learn.org/stable/modules/generated/sklearn.covariance.EllipticEnvelope.html`

[46] scikit-learn developers: *Anomaly detection with Local Outlier Factor (LOF)*. [online]. [cit. 2017-04-15]. Dostupné z: `http://scikit-learn.org/stable/auto_examples/ensemble/plot_isolation_forest.html`

[47] scikit-learn developers: *Anomaly detection with Local Outlier Factor (LOF)*. [online]. [cit. 2017-04-15]. Dostupné z: `http://scikit-learn.org/stable/auto_examples/neighbors/plot_lof.html`

[48] Andrade, A. T. C.; Montez, C.; Moraes, R.; aj.: Outlier detection using k-means clustering and lightweight methods for Wireless Sensor Networks. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Oct 2016, s. 4683–4688, doi:10.1109/IECON.2016.7794093.

[49] Gan, G.; Ng, M. K.-P.: k-means clustering with outlier removal. *Pattern Recognition Letters*, ročník 90, 2017: s. 8 – 14, ISSN 0167-8655, doi:https://doi.org/10.1016/j.patrec.2017.03.008. Dostupné z: `http://www.sciencedirect.com/science/article/pii/S0167865517300740`

[50] Project Jupyter, et al.: *Jupyter*. [online]. [cit. 2018-1-23]. Dostupné z: `http://jupyter.org`

[51] České vysoké učení technické v Praze, Fakulta informačních technologií: *Anketa FIT ČVUT*. [online]. [cit. 2017-04-15]. Dostupné z: `https://anketa.fit.cvut.cz`

[52] Object Management Group®, Inc.: *Unified Modeling Language*. [online]. [cit. 2018-3-26]. Dostupné z: `http://www.uml.org`

[53] Contributors of Wikipedia: *Unified Modeling Language*. [online]. [cit. 2018-3-26]. Dostupné z: https://cs.wikipedia.org/wiki/Unified_Modeling_Language

[54] Sparx Systems Pty Ltd.: *Enterprise Architect*. [online]. [cit. 2018-3-28]. Dostupné z: http://sparxsystems.com/products/ea/

[55] Malý, M.: *REST: architektura pro webové API*. [online]. 3.8.2009 [cit. 2018-4-5]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

[56] Docker Inc.: *Docker*. [online]. [cit. 2018-3-31]. Dostupné z: https://docs.docker.com

[57] Ronacher A.: *Flask*. [online]. [cit. 2018-3-31]. Dostupné z: http://flask.pocoo.org

[58] Django Software Foundation: *Django*. [online]. [cit. 2018-3-31]. Dostupné z: https://www.djangoproject.com

[59] Pallets Team: *Larger Applications*. [online]. [cit. 2018-3-31]. Dostupné z: http://flask.pocoo.org/docs/0.12/patterns/packages/

[60] W3Schools: *HTML - HyperText Markup Language*. [online]. [cit. 2018-4-5]. Dostupné z: https://www.w3schools.com/html/

[61] W3Schools: *CSS - Cascading Style Sheets*. [online]. [cit. 2018-4-5]. Dostupné z: https://www.w3schools.com/css/

[62] W3Schools: *SQL*. [online]. [cit. 2018-4-5]. Dostupné z: https://www.w3schools.com/sql/

[63] The PostgreSQL Global Development Group: *PostgreSQL: Documentation*. [online]. [cit. 2018-4-20]. Dostupné z: https://www.postgresql.org/docs/

[64] Apache Software Foundation: *Apache Cassandra*. [online]. [cit. 2018-4-5]. Dostupné z: http://cassandra.apache.org

[65] redislabs: *Redis*. [online]. [cit. 2018-4-15]. Dostupné z: https://redis.io

[66] Basho: *Riak*. [online]. [cit. 2018-4-5]. Dostupné z: http://basho.com/

[67] Contributors of Wikipedia: *List of HTTP status codes*. [online]. [cit. 2018-4-5]. Dostupné z: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

[68] Ronacher A.: *Welcome to Jinja2*. [online]. [cit. 2018-3-21]. Dostupné z: http://jinja.pocoo.org/docs/2.10/

[69] Whelan P.: *Embed Base64-Encoded Images Inline In HTML*. [online]. [cit. 2018-4-5]. Dostupné z: http://www.bigfastblog.com/embed-base64-encoded-images-inline-in-html

[70] Pivotal Software, Inc: *RabbitMQ*. [online]. [cit. 2018-4-15]. Dostupné z: https://www.rabbitmq.com

[71] GoPivotal, Inc.: *Mozilla Public License*. [online]. [cit. 2018-4-15]. Dostupné z: https://www.rabbitmq.com/mpl.html

[72] redislabs: *Redis license and trademark information*. [online]. [cit. 2018-4-15]. Dostupné z: https://redis.io/topics/license

[73] Solem A., et al.: *Celery: Distributed Task Queue*. [online]. [cit. 2018-4-10]. Dostupné z: http://www.celeryproject.org

[74] Driessen V.: *RQ (Redis Queue)*. [online]. [cit. 2018-4-10]. Dostupné z: http://python-rq.org

[75] Contributors of Wikipedia: *HTTPS*. [online]. [cit. 2018-4-5]. Dostupné z: https://cs.wikipedia.org/wiki/HTTPS

[76] GitHub Inc.: *GitHub*. [online]. [cit. 2018-4-5]. Dostupné z: https://github.com

[77] GitHub Inc.: *Securing your webhooks*. [online]. [cit. 2018-4-5]. Dostupné z: https://developer.github.com/webhooks/securing/

[78] SQLAlchemy authors and contributors: *SQLAlchemy*. [online]. [cit. 2018-3-31]. Dostupné z: https://www.sqlalchemy.org

[79] Contributors of Wikipedia: *Radar*. [online]. [cit. 2018-4-20]. Dostupné z: https://en.wikipedia.org/wiki/Radar_chart

[80] Matplotlib development team: *Matplotlib*. [online]. [cit. 2018-1-24]. Dostupné z: https://matplotlib.org

[81] Waskom, M.: *seaborn*. [online]. [cit. 2018-1-20]. Dostupné z: https://seaborn.pydata.org

[82] NumPy developers: *NumPy*. [online]. [cit. 2018-1-20]. Dostupné z: http://www.numpy.org

[83] Pryke, B.: *Understanding SettingwithCopyWarning in pandas*. [online]. [cit. 2018-1-23]. Dostupné z: https://www.dataquest.io/blog/settingwithcopywarning/

[84] Rothberg, A.: *Calling mean on groupby of timedelta does not work*. [online]. [cit. 2018-1-21]. Dostupné z: https://github.com/pandas-dev/pandas/issues/12440

[85] Harrison, H. S.: *groupby.mean, etc, doesn't recognize timedelta64*. [online]. [cit. 2018-1-21]. Dostupné z: https://github.com/pandas-dev/pandas/issues/5724

[86] Naveen: *What is V Model in software testing and what are advantages and disadvantages of V Model*. [online]. [cit. 2018-4-30]. Dostupné z: http://testingfreak.com/v-model-software-testing-advantages-disadvantages-v-model/

[87] pytest-dev team: *pytest: helps you write better programs*. [online]. [cit. 2018-4-24]. Dostupné z: https://docs.pytest.org/en/latest/

[88] AITOM Digital: *Uživatelské testování krok za krokem*. [online]. [cit. 2018-4-28]. Dostupné z: https://www.pojdmetestovat.cz/file/16

[89] Nginx Development Team: *nginx*. [online]. [cit. 2018-5-6]. Dostupné z: https://nginx.org/en/

[90] Alpine Linux Development Team: *Alpine Linux*. [online]. [cit. 2018-5-6]. Dostupné z: https://alpinelinux.org

[91] Docker Inc.: *Docker Hub*. [online]. [cit. 2018-6-6]. Dostupné z: https://hub.docker.com

[92] Salesforce.com: *Heroku: Cloud Application Platform*. [online]. [cit. 2018-3-31]. Dostupné z: https://www.heroku.com

[93] IBM: *What is cloud computing?* [online]. [cit. 2018-3-31]. Dostupné z: https://www.ibm.com/cloud/learn/what-is-cloud-computing

[94] Salesforce.com: *Dynos and the Dyno Manager*. [online]. [cit. 2018-3-31]. Dostupné z: https://devcenter.heroku.com/articles/dynos

[95] Grinberg M.: *The Flask Mega-Tutorial Part XVIII: Deployment on Heroku*. [online]. [cit. 2018-3-31]. Dostupné z: https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-xviii-deployment-on-heroku

[96] Salesforce.com: *Heroku CLI*. [online]. [cit. 2018-3-31]. Dostupné z: https://devcenter.heroku.com/articles/heroku-cli

[97] Salesforce.com: *Heroku Add-ons*. [online]. [cit. 2018-3-31]. Dostupné z: https://elements.heroku.com/addons

[98] Brandl G.; Sphinx team: *Sphinx Python Documentation Generator*. [online]. [cit. 2018-4-20]. Dostupné z: http://www.sphinx-doc.org/en/master/

[99] Hrončok M., et al.: *Dokumentace ve Sphinx*. [online]. [cit. 2018-4-20]. Dostupné z: `https://naucse.python.cz/2017/mipyt-zima/intro/docs/`

[100] Tibs; Goodger D.: *Quick reStructuredText*. [online]. [cit. 2018-4-20]. Dostupné z: `http://docutils.sourceforge.net/docs/user/rst/quickref.html`

[101] Ruana R.: *Example Google Style Python Docstrings*. [online]. [cit. 2018-4-20]. Dostupné z: `http://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html`

[102] The Apache Software Foundation: *Apache Spark^{TM}: Lightning-fast unified analytics engine*. [online]. [cit. 2018-4-24]. Dostupné z: `https://spark.apache.org`

[103] The Apache Software Foundation: *Apache Spark^{TM}: Welcome to Spark Python API Docs!* [online]. [cit. 2018-4-24]. Dostupné z: `https://spark.apache.org/docs/latest/api/python/`

[104] Borovička T., et al.: *Anotace předmětu Distribuovaný Data Mining*. [online]. [cit. 2018-4-24]. Dostupné z: `https://edux.fit.cvut.cz/courses/MI-DDM/annotation/start`

Příloha A

Seznam použitých zkratek

ang. anglicky

AG1 Algoritmy a grafy 1 (předmět na FIT ČVUT)

API Application Programming Interface

BI zkratka pro předmět na bakalářském stupni FIT ČVUT v českém jazyce

CD Compact Disc

CPU Central Processing Unit

CRISP-DM Cross-industry standard process for data mining

CSS Cascading Style Sheets

ČVUT České vysoké učení technické v Praze

DDM Distribuovaný Data Mining (předmět na FIT ČVUT)

DM Data Mining

EDM Educational Data Mining

FIT Fakulta informančních technologií

GB Gigabyte

HMAC Keyed-hash Message Authentication Code

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

LIN Lineární algebra (předmět na FIT ČVUT)

LOF Local Outlier Factor

KAM Katedra aplikované matematiky (FIT ČVUT)

KB Kilobyte

KD Knowledge Discovery

KDD Knowledge Discovery in Databases

KDDM Knowledge Discovery and Data Mining

MARAST MAtematika RAdoSTně

MB Megabyte

MI zkratka pro předmět na magisterském stupni FIT ČVUT v českém jazyce

MPI Matematika pro informatiku (předmět na FIT ČVUT)

MQ Message Queue

PDF Portable Document Format

PKM Přípravný kurz matematiky (předmět na FIT ČVUT)

PST Pravděpodobnost a statistika (předmět na FIT ČVUT)

RAM Random Access Memory

REST REpresentational State Transfer

RQ Redis Queue

SAS Statistical Analysis System

SQL Structured Query Language

SSE Sum of Squared Errors

UML Unified Modeling Language

URL Uniform Resource Locator

vCPU virtual Central Processing Unit

ZDM Základy diskrétní matematiky (předmět na FIT ČVUT)

ZMA Základy matematické analýzy (předmět na FIT ČVUT)

Příloha **B**

Obsah přiloženého CD

```
readme.txt ..... stručný popis obsahu CD
src ..... složka se zdrojovými kódy
    impl ..... zdrojové kódy implementace
    thesis ..... zdrojová forma práce ve formátu LaTeX
text ..... text práce
    DP_Nováček_Jan_2018.pdf ..... text práce ve formátu PDF
attachments ..... složka s přílohami
```