



**FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE**

Bakalářská práce

Aplikace pro předpovídání transakcí na základě finančních akcí

Radek Puš

Katedra softwarového inženýrství

Vedoucí práce: Ing. Miroslav Balík, Ph.D.

14. ledna 2020

Poděkování

Děkuji společnosti Trask Solutions a.s., zejména pak Ing. Petru Hnízdilovi a Ing. Pavlu Svobodovi, za veškeré konzultace a datové podklady k práci. Dále děkuji své přítelkyni Mariyi Tsanko, za dodání doplňujících dat ze svých bankovních účtů. Také děkuji svým rodičům, za veškerou podporu při psaní práce

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 14. ledna 2020

.....

České vysoké učení technické v Praze
Fakulta informačních technologií

© 2020 Radek Puš. 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

Puš, Radek. *Aplikace pro předpovídání transakcí na základě finančních akcí*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020. Dostupný také z WWW: (<https://gitlab.fit.cvut.cz/pusradek/bakalarka>).

Abstrakt

V Bakalářské práci byl řešen problém předpovídání budoucích transakcí klientovi, na základě jeho uplynulé finanční historie. Cílem práce je pokusit se implementovat toto předvídání pomocí umělé inteligence spolu s vytvořením webového rozhraní. Toho bylo dosaženo pomocí Angularu, tvořící klientskou aplikaci, frameworkem .NET Core, který tuto aplikaci obsluhuje ve spojení s databází Microsoft SQL. Zabezpečení využívá JWT tokenu a o šifrování uživatelských hesel se stará RFC 2898/SHA512. Umělá inteligence byla řešena jako lineární regrese.

Umělá inteligence je implementována jako několik po sobě jdoucích vrstev, jejichž počet neuronů se postupně snižuje na jeden jediný, který dává informaci, zda se bude transakce opakovat, či nikoli. Aby se posléze dalo vyhodnotit, zda se transakce opravdu opakovala, byly porovnávány velikosti transakcí po jednotlivých týdnech, a ty, částkou si nejvíce podobné, byly označeny za opakující se.

Klíčová slova webový portál, předvídání finančních transakcí, analýza finančních transakcí, umělá inteligence, .NET Core, Entity Framework, Angular

Abstract

In the Bachelor thesis was solved the problem of forecasting future transactions to the client, based on his past financial history. The aim of this work is to try to implement this prediction by using artificial intelligence together with creating a web interface. This was achieved by using Angular, a client application, with the .NET Core framework, which runs the application in conjunction with the Microsoft SQL database. Security uses the JWT token and RFC 2898/SHA512 takes care of user password encryption. Artificial intelligence was solved as linear regression.

I have implemented artificial intelligence as several consecutive layers, the number of neurons gradually decreasing to one, which gives me information whether the transaction will repeat or not. In order to assess whether the transaction was actually recurring, I compared the size of the transactions on a week-by-week basis and described the amount most similar as recurring.

Keywords web portal, anticipating financial transactions, financial transaction analysis, artificial intelligence, .NET Core, Entity Framework, Angular

Obsah

Úvod	1
1 Cíl práce	3
2 Analýza	5
2.1 Analýza požadavků	5
2.2 Webové rozhraní	6
2.2.1 Single Page website	6
2.2.2 Multi Page website	6
2.3 Zabezpečení	6
2.4 Umělá inteligence	6
2.4.1 Obdobné frameworky pro umělou inteligenci	7
2.5 Obohacení dat	8
2.6 Transakce	8
2.7 Ukládání dat	9
2.7.1 Uživatel	9
2.7.2 Uživatelská data	9
2.7.3 Neuronová síť	9
3 Realizace	11
3.1 Webové rozhraní	11
3.2 Zabezpečení	11
3.3 Import CSV souborů	11
3.3.1 Import	12
3.3.2 Parsování	12
3.3.3 Ukládání	13
3.3.4 Obohacení	13
3.4 Umělá inteligence	14
3.4.1 Implementace základní části neuronové sítě	15
3.4.1.1 Předvídání	16

3.4.1.2	Učení	17
3.4.2	Export, import sítě	19
3.4.3	Testování funkčnosti neuronové sítě	19
3.4.3.1	Sudá a lichá čísla	19
3.4.3.2	XOR	20
3.4.3.3	Export, Import	20
3.4.4	Komparátor	20
3.4.4.1	Přiřazování do páru	22
3.4.4.2	Porovnávání vzdálenosti	22
3.4.5	Testování komparátoru	23
3.4.6	Datafeeder	23
3.4.7	Normalizace dat	24
3.5	Předvídání bez umělé inteligence	26
3.5.1	Předvídání na základě velikostí transakce	26
3.5.2	Předvídání na základě průměru	28
3.6	Databáze	28
3.6.1	Uživatel	28
3.6.2	Transakce	29
3.6.3	Konstantní symbol	30
3.6.4	Neuronová síť	30
4	Výsledné uživatelské rozhraní	33
4.1	Přihlašovací obrazovka	33
4.2	Registrační obrazovka	33
4.3	Hlavní obrazovka	34
5	Vyhodnocení kvality	37
6	Doplnit	39
6.1	scénáře testující použití	39
7	Závěrečné testování	41
7.1	Manuální testování	41
7.1.1	Změna dat	41
7.1.2	Autentifikace	42
7.1.3	Uživatel bez JavaScriptu	43
7.1.4	Responzivita	44
7.2	W3C validace	44
8	Možnosti rozšíření	47
8.1	Backend	47
8.2	Frontend	48
	Závěr	49

Literatura	51
A Seznam použitých zkratek	55
B Přiložené soubory	57
C Obsah přiloženého CD	59

Seznam obrázků

2.1	Moje zdravé finance	7
3.1	Struktura neuronové sítě	16
3.2	Průchod neuronem	17
3.3	Test sudých a lichých čísel	19
3.4	Test XOR operátoru s málo neurony	20
3.5	Test XOR operátoru s doplněnými neurony	20
3.6	Test exportu a importu sítě	21
3.7	Test komparátoru	24
3.8	Normalizace částky	27
3.9	Model databáze - uživatel	29
3.10	Model databáze - transakce	30
3.11	Model databáze - konstantní symbol	31
3.12	Model databáze - neuronová síť	32
4.1	Uživatelské rozhraní - přihlašovací obrazovka	33
4.2	Uživatelské rozhraní - registrační obrazovka	34
4.3	Uživatelské rozhraní - menu a účet	34
4.4	Uživatelské rozhraní - uživatelská historie	35
4.5	Uživatelské rozhraní - předpověď neuronovou sítí	35
4.6	Uživatelské rozhraní - předpověď průměrem a hodnotou	36
7.1	Jednotlivé stavy nahrávání souborů	42
7.2	Validace formuláře na změnu hesla	42
7.3	Validace přihlašovacích údajů	43
7.4	Validace polí na formuláři	43
7.5	Nepovolený JavaScript v prohlížeči	44
7.6	Responzivita – porovnání obrazovek	44
7.7	W3C – nepovolené atributy	45

Úvod

Ve své práci řeším problém zpracování finančních transakcí. Při výběru jsem využil možného zájmu bankovní instituce o odkoupení výsledného produktu. Ta by mohla využít mou práci k zobrazování pravděpodobných transakcí vlastním uživatelům a lépe tak porozumět chování a potřebám klientů samotných. Tím by mohla mít produkty lépe cílené na zákazníky. Mně osobně navíc vždy zajímal vývoj v oblasti umělé inteligence a chtěl jsem si proto své znalosti prohloubit.

Technologie, které jsem využil pro svou práci, jsem si vybral na základě svého současného zaměstnání. Skrze společnost Trask Solutions a.s. pracuji jako externista a software developer ve společnosti ČSOB Leasing a podílím se na vývoji aplikaci právě v jazyce C# a frameworku ASP.NET. Avšak framework jsem, pro účely této práce, použil novější (.NET Core ve spojení s Entity Frameworkem). Pro Frontend jsem zvolil Angular, protože jsem se chtěl také naučit něco nového. Zkušenosti nabyté při tvorbě bakalářské práce, bych rád využil ke zvýšení své kvalifikace v oboru. Navíc spolupracuji s týmem, který v tomto frameworku vyvíjí aplikace a v případě potřeby mi mohou poradit.

Práci jsem rozčlenil do čtyř kapitol. V první jsem analyzoval problém a zkoumal, jestli již byl řešen dříve. Ve třetí kapitole, realizaci, popisuji, jak jsem vše řešil. Ať už web, umělou inteligenci nebo databázi. V následující kapitole, testování zkoumám, jestli se aplikace chová korektně. V poslední kapitole jsem se pak zabýval, jakým způsobem by šlo aplikaci rozšířit.

Cíl práce

Cílem práce je vytvoření webové aplikace, která bude pro skupinu uživatelů poskytovat službu předvídání finančních transakcí na základě jimi nahraného výpisu z účtu.

Cílem teoretické části práce je nastudování Angular frameworku, TypeScriptu a Entity Frameworku Core / .NET Core. Dále je třeba zjistit možnosti využití umělé inteligence, která bude muset alespoň z části předpovědět chování uživatelů. Tato umělá inteligence bude předpovídat chování uživatelů na základě předem získaných (anonymizovaných) a obohacených dat jejich finanční historie.

Cílem praktické části je implementovat tuto webovou aplikaci, včetně zabezpečení, a pokusit se implementovat a integrovat umělou inteligenci, která bude vytěžovat údaje z uživatelem nahraných dat.

Analýza

Projekt je webová aplikace. Musí se proto skládat ze dvou částí: Frontendu a Backendu. Frontend je webovým rozhraním a Backend zpracovává údaje uživatele a dodává webovému rozhraní data.

2.1 Analýza požadavků

Aplikace musí splňovat jisté požadavky. Ty se dají rozdělit na požadavky funkční a nefunkční:

a) Funkční požadavky

- C#
- TypeScript
- Angular
- .NET Core
- JWT
- Microsoft SQL Databáze
- šifrování hesel

b) Nefunkční požadavky

- registrace a přihlašování uživatelů
- změna hesla
- smazání účtu
- nahrávání finanční historie ve formátu csv
- zobrazení historie plateb
- odhad výdajů neuronovou sítí
- odhad výdajů průměrováním výdajů
- odhad výdajů porovnáváním výdajů

2.2 Webové rozhraní

Pro webové rozhraní se nabízejí dvě možná řešení. Single Page Website a Multiple Page website. Každé má své pro a proti.

2.2.1 Single Page website

Jak už název napovídá, je to web, skládající se z jediné stránky. Stránka obvykle neobsahuje příliš mnoho informací a je založena zejména na dynamičnosti prvků. Velmi často jsou to doprovodné stránky nebo krátké informativní stránky, s jejichž pomocí se subjekty na webu prezentují.

2.2.2 Multi Page website

Tyto weby, vývojově starší, obsahují zpravidla hodně informací. Obsahují nějakou formu směřování - ať už složitější, dynamickou, či jednodušší, statickou. Uživatelé podávají poměrně velké množství informací. Jsou to zpravidla nejrůznější blogy, e-shopy, fóra. Jejich implementace je často snazší (zejména u statických webů), jsou schopny, často přehledněji, zprostředkovat uživateli více informací a jsou (i z historických důvodů) běžnější.

2.3 Zabezpečení

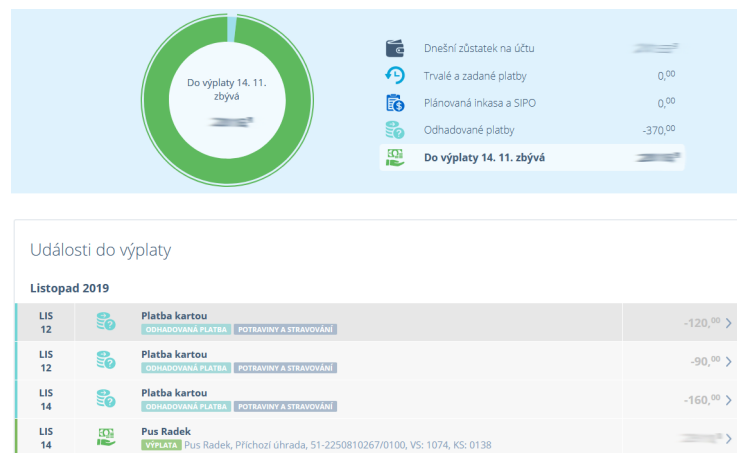
V aplikaci bude nutné zabezpečit HTTP API. To je možné pomocí Session anebo pomocí, novějšího, JWT tokenu. Tyto dva přístupy se v zásadě liší způsobem ukládání informace o autentifikaci uživatele. Session tuto informaci ukládá do databáze, kdežto JWT tokeny jsou uloženy u uživatele.

Ukládání u uživatele má tu výhodu, že se nemusí ukládat na serveru tolik dat do mezipaměti. Ukládání na straně serveru, v případě několika uživatelů, není nijak zásadní, avšak u většího množství to přináší podstatně vyšší nároky na server.[1] Navíc JWT tokeny jsou imunní vůči CSRF útoku[2].

Výhodou ukládání na server je zase fakt, že je mnohem snazší znevalidnit uživatelské přihlášení - stačí odebrat záznam z databáze. U JWT je nutné vyčkat na vypršení tokenu anebo vytvořit nějakou formu blacklistu (seznam neplatných tokenů).

2.4 Umělá inteligence

Na českém trhu je, co se týče předpovídání transakcí, velmi pravděpodobně zatím jediné finanční bankovníctví České Spořitelny, zvané "George" a její doplněk "Moje zdravé finance." [3] Aplikace je přístupná pouze klientům České spořitelny a zobrazuje pouze příjmy a výdaje. V části "Moje zdravé finance" lze pak vidět, jak dlouho mi vystačí současné úspory nebo například odhadnout



Obrázek 2.1: Moje zdravé finance - předpovídání transakcí České spořitelny

výši mého budoucího důchodu. Je ale schopna předpovědět některé mé budoucí transakce, tedy funkci, kterou mám za úkol implementovat. Aplikace byla spuštěna v průběhu května 2018[4],[5] (ačkoli přesné datum jsem nebyl schopen dohledat), tedy chvíli po schválení zadání mé Bakalářské práce. To tedy dokazuje, že o téma je v praxi zájem, ale bohužel také to, že v mezichase mého vypracovávání, byl problém zpracován jinde.

Na zahraničním trhu pak existují aplikace, které umožňují klientům sledovat např. jejich měsíční útratu a na základě výpočtu jim poskytnout informaci, jestli a jak moc, překročili průměrné výdaje v tom kterém měsíci.[6].

2.4.1 Obdobné frameworky pro umělou inteligenci

Existuje mnoho externích knihoven, které implementují neuronové sítě. Většina je však placená a nenabízí rozhraní pro C#, často se také omezují pouze na Python případně C++. Příkladem může být Opennn[7], PyTorch[8], Keras[9], Tensorflow[10] či Machine Learning od AWS (dceřiná společnost Amazonu)[11].

Umělá inteligence určená pro C# je vyvíjena hlavně společností Microsoft. Ta vyvinula dva frameworky CNTK a ML.NET.

CNTK je framework zaměřený na neuronové sítě[12]. Zpočátku obsahoval jen Python API, ale později byl rozšířen i o C# API. V současné době dochází ještě k tvorbě Java API.[13]

ML.NET je zaměřen na .NET vývojáře[14]. Zároveň se snaží být více multifunkční (obsahuje i algoritmy jako K-Means[15]). Také umožňuje základní generování kódu pro veřejnost (pomocí AutoML[16]), která se nezabývá detailní funkčností umělé inteligence.

2.5 Obohacení dat

Pro obohacení dat by bylo možné teoreticky využít veřejně dostupnou databázi MFČR - ARES (Administrativní registr ekonomických subjektů)[17]. Bohužel data v ní nejsou ukládána jednotným způsobem. Je sice zadefinována základní struktura dokumentu[18], avšak data v ní nejsou nijak strukturována. Živnosti jsou zadány naprosto nahodile a není využito žádného oficiálního klíče. Nedefinované hodnoty obsahují výrazy jako: "0", "null", "NULL", "prázdné", "nic", "není", "nedefinováno" apod. Jako nejlepší příklad záznamu v databázi by pak mohla dobře posloužit Komerční banka. Ta nemá dokonce definováno vůbec nic, až na popis, který obsahuje necelých 8000 znaků. Lze tedy usoudit, že využití tohoto zdroje je téměř nemožné a pro obohacení se nehodí.

Další možnost, která by se nabízela je využití bankovních symbolů. Variabilní a specifické symboly opět neobsahují žádné informace. Někdy do nich bývá zadáváno IČO, které by se dalo napojit na ARES, ale jak už je zmíněno výše, toto napojení nepřipadá v úvahu.

Zbývají tedy už jen Konstantní symboly. Ty se bohužel využívají pouze v rámci České Republiky, ale mají alespoň částečně definovanou strukturu. A ačkoli její použití již není povinné, ve výpisech z účtů, které byly pro tuto práci získány, se vyskytují hojně a mohou o transakcích něco říci.

2.6 Transakce

Od uživatele je vyžadován jistý vstup. Pro ten je nutné definovat určité minimální informace. Protože každá banka tvoří výpisy z účtu nepatrně jinak, došlo k omezení na tyto informace:

- 1) datum
- 2) částka
- 3) konstantní symbol

Z data transakce lze snadno zjistit, o jaký den se jednalo, i jestli k transakci došlo o víkend. Částka jednoduše označuje velikost transakce. Konstantní symbol pak pomáhá obohatit data o další vstupy, jak již bylo zmíněno v kapitole 2.5 Obohacení dat.

Transakční historie obvykle obsahuje ještě další údaje, ty se ale většinou v jednotlivých bankách liší a proto je nelze použít. Příkladem zde může být Typ transakce, který je označen v každé bance jinak.

2.7 Ukládání dat

Aplikace bude muset umožňovat ukládat uživatelské účty a zároveň k nim i transakční historii od uživatelů. Je tedy vhodné vytvořit nějakou formu databáze.

2.7.1 Uživatel

Za první část databáze by se dali považovat samotní uživatelé, kteří budou do aplikace přistupovat. Je nutné si pro ně pamatovat uživatelské jméno a heslo. Heslo samozřejmě musí být uloženo v šifrované podobě.

2.7.2 Uživatelská data

Druhá část, ukládání uživatelských dat, je náročnější. Je potřeba zvážit, jak je ukládat. Veškerá data totiž budou přístupná z csv souborů ("*Comma-Separated Values*" – čárkou oddělené hodnoty)[19]. Z toho důvodu se nabízí dvě možnosti:

- a) Ukládání ve formě NoSQL
- b) Parsování transakcí do podoby nějakého relačního modelu a uložení v něm.

Ukládáním ve formě NoSQL je myšleno ukládání dat, jak přijdou od uživatele s minimálními úpravami (as-is). Pro veškeré operace s uživatelskými daty by se využívala jedna "velká tabulka" (super-entita). Z transakcí samotných lze vyvodit nějakou jistou strukturu a v tom případě bude docházet k zbytečně velké redundanci dat.

Druhou možností je transakci rozdělit na několik částí (tabulek) a s těmi následně pracovat. Výhodou zde je práce s menšími daty i menší databází.

2.7.3 Neuronová síť

Neuronová síť je tvořena pomocí vrstev, neuronů a vah vstupů. Ukládání těchto dat přímo vybízí k uložení ve strukturované databázi, rozdělené přímo na tyto samostatné celky.

Realizace

3.1 Webové rozhraní

Pro webové rozhraní byla zvolena, Single Page website, doplněná o stránky s registrací a přihlášením. Není potřeba uživateli předávat velké množství informací. V zásadě jen zobrazovat grafy a předpoklady. Nemusí se tedy vytvářet rozsáhlý web a jen jednoduché směřování. V případě malého množství stránek se navíc snadněji pracuje s responzivitou, protože ji stačí vyladit na jednu, resp. tři stránky webu. Jakákoli práce s responzivitou pro každou další stránku pak zvětšuje prostor na chybu.

3.2 Zabezpečení

V případě zabezpečení komunikace mezi backendem a frontendem (API) bylo využito JWT tokenu. Kromě výhod, uvedených v analýze, je tento přístup k zabezpečení i novější (dle Wikipedie poprvé vydán 28.12.2010[20] - v průběhu psaní práce starý necelých osm let), a umožní vyzkoušet si novou technologii.

K zabezpečení směřování na straně klienta bylo využito tzv. guardů, jež jsou součástí Angular frameworku. A konečně k zabezpečení uživatelských hesel se využilo hashovacího algoritmu RFC 2898/SHA512. Pro každé heslo se navíc generuje jiná sůl a proto by nemělo být možné v databázi najít dvě stejná hesla se stejným hashem.

3.3 Import CSV souborů

Aby bylo možné činit nějaké předpovědi, je nutné od uživatele získat nějaká data. Toho bylo docíleno na základě nahrání vyexportované transakční historie z banky uživatele. Ta musí být zároveň vyexportována ve formátu CSV.

3.3.1 Import

Celý import začíná na webovém rozhraní. Zde uživatel vybere soubor a stiskne tlačítko "nahrát". To způsobí zahájení celého procesu parsování dokumentu a jeho ukládání do databáze.

Interně celý proces probíhá tak, že dojde k zavolání nějaké funkce, která převezme daný soubor a přes API pošle na server. Při tomto požadavku se pak automaticky (jako při každém volání API) zavolá tzv. "Interceptor". Ten vloží do hlavičky volání ještě údaj o JWT tokenu, aby server mohl rozpoznat, jestli je volání oprávněné či nikoli. JWT token navíc obsahuje údaje s informací, kterému uživateli tento požadavek náleží.

Po zpracování požadavku na frontendu, dojde ke zpracování požadavku na serveru. Ten ověří, zda-li je uživatel přihlášen a soubor je v pořádku a zároveň již nebyl nahrán soubor se stejným jménem pro daného uživatele. Pokud ano, pošle soubor dále ke zpracování. V opačném případě vrátí HTTP status BadRequest a dál nepokračuje.

3.3.2 Parsování

Pokud došlo k úspěšnému zpracování požadavku, dojde k parsování souboru. K tomuto procesu byla využita externí knihovna s názvem CsvHelper[21].

Tato knihovna potřebuje k úspěšnému zpracování dokumentu nadefinovanou třídu, jež obsahuje veškeré položky, které mají být v dokumentu vybrány. Přestože by pro extrakci stačilo vybrat pouze částku, datum a KS, bylo načteno podstatně více údajů pro případné rozšíření aplikace. Veškeré načítané údaje jsou proto následující:

- typ účtu
- číslo účtu (včetně kódu banky)
- datum zaúčtování
- částka
- měna
- zůstatek
- číslo protiúčtu (včetně kódu banky)
- konstantní, specifický, variabilní symbol
- typ operace
- ID transakce v bance
- poznámka

K výše zmíněnému výčtu je však nutno zdůraznit, že ne všechny položky jsou povinné. V práci došlo na omezení se na pouze dvě povinné položky a totiž datum zaúčtování a částka. V případě chybějícího čísla účtu dochází k jeho nahrazení ID uživatele s kódem banky "857368". Tato hodnota je ekvivalentní textu "UID" (User ID - uživatelské ID) v ASCII tabulce (kódová tabulka pro znaky anglické abecedy).

Při realizaci bylo myšleno i na různá pojmenování napříč jednotlivými bankami a k pokusu o vytvoření maximální univerzálnosti řešení. Proto se pro některé položky definovalo více názvů. Příkladem tak může být položka "označení operace", jak je pojmenována v ČSOB a "typ transakce", což je ekvivalent v České spořitelně.

Relativně překvapivým problémem je však parsování částky transakce. Při exportu dat totiž může být ve formátu "1 234.56" - s mezerou, oddělující řády. Také ale může obsahovat rozdílné oddělovače desetinných míst, př. "12,35". Problém mezery však lze snadno vyřešit povolením této skutečnosti při parsování pomocí metody, v .NET frameworku, System.Parse.

Problém desetinné čárky není o tolik těžší. Před parsováním stačí částku projít a nahradit znak ",", znakem ".". Tím dojde k unifikaci čísla.

3.3.3 Ukládání

Po parsování dokumentu dochází k ukládání pouze těch záznamů, které se podařilo úspěšně zpracovat. Všechny neúspěšné záznamy jsou přeskočeny. Tím dochází k založení záznamu o nahraném souboru. Ten obsahuje jen jméno souboru a identifikaci, uživatele, který ho nahrál. K tomu to záznamu se posléze přiřadí veškeré transakce, které byly tímto souborem nahrány. Děje se tak z důvodu, že by uživatel chtěl daný soubor smazat.

3.3.4 Obohacení

Protože data z ARES nelze smysluplně zpracovat, došlo k obohacení data na základě konstantních symbolů. Tento prvek sice nemusí být přítomen v transakci, pro operace, jež má na starosti banka (karetní operace, výběry z bankomatu,...), jsou však často přítomny.

Přidání závislostí na konstantních symbolech bylo založeno na číselníku MFČR. V práci pak bylo konkrétně využito jeho souhrn ze serveru Finance.cz[22]. Číselník byl zároveň rozdělen do následujících kategorií:

a) Druh

- označení druhu transakce
- platby za zboží a služby, mzdové náklady,...

b) Dle významu poslední číslice

3. REALIZACE

- označení druhu platby
- hotovost, dobropisy, přednostní platby,...

c) Spravované MFČR

- popisují vztahy ke státnímu rozpočtu a rozpočtům místních samospráv
- dotace, daně, cla, pokuty, penále,...

d) Rezervované symboly

- symboly rezervované pro mezibankovní styk
- platby kartou, šekem, platby na neexistující účet,...

Dále existuje ještě i sada zahraničních konstantních symbolů. Sada shrnuje různé druhy plateb do dvaceti různých kategorií. Vzhledem k faktu, že se pro práci nepodařilo zajistit žádné výpisy z účtu, jež by obsahovaly zahraniční platby, tato sada nebyla využita. Byla však definována pro případné pozdější rozšíření dat.

3.4 Umělá inteligence

Pro předpovídání budoucích transakcí je využita neuronová síť. Tedy síť, na způsob lidského mozku, která obsahuje množství neuronů. Celá tato síť pak funguje tak, že pokud dojde k aktivaci nějakého neuronu, tak ten postupně ovlivní jeden, či více dalších neuronů a ty zase další. Míru aktivace pak ovlivňují tzv. váhy pro každý z předchozích neuronů (vstupů). Způsob jejího fungování velmi názorně ukazuje např. Grant Sanderson ve svém videu "*But what is a Neural Network?*"[23]

Aby tedy neuronová síť mohla fungovat, potřebuje nějaká data, která by aktivovala vstupní neurony. Tato data musí být přizpůsobena tak, aby vypadala jako výstup jiných neuronů - musí být "normalizovaná".

Dále potřebuje nějaká data, za pomoci kterých by si mohla nadefinovat, na kolik budou jednotlivé neurony ovlivňovat ty ostatní. Jinými slovy data, podle kterých se bude síť učit předvídat. To jsou v této aplikaci veškeré transakce, jež se opakují.

Lze tedy pozorovat, že implementace předpovídání transakcí je poměrně rozsáhlá. Proto byla rozdělena do několika konzolových aplikací. Každou z těchto částí je tak možné nezávisle implementovat i testovat. V případě potřeby je možné takto i jeden modul reimplementovat a nedojde k ovlivnění dalších. Ve výsledku byla proto umělá inteligence rozdělena následovně:

1. Datafeeder

- část aplikace, která slouží k postupnému čtení dat z databáze a k předání dalším objektům (částí aplikace)
- tato část má na starost zároveň i normalizaci dat

2. Komparátor

- udává, které transakce se opakovaly a které nikoli

3. Neuronová síť

- na základě vstupních dat se pokusí předpovědět výsledek

3.4.1 Implementace základní části neuronové sítě

Samotná neuronová síť je stále velký celek, který bylo potřeba rozdělit na podobjekty. Bylo tak tedy učiněno dle logických celků - nejdříve se dělí na jednotlivé vrstvy. Vrstvy mohou být:

a) Vstupní vrstva

- prochází přes ni veškeré vstupy do neuronové sítě.
- svůj výstup předává vnitřním vrstvám
- je první a zároveň, v rámci sítě, unikátní vrstvou

b) Vnitřní/skrytá vrstva (vstupně-výstupní)

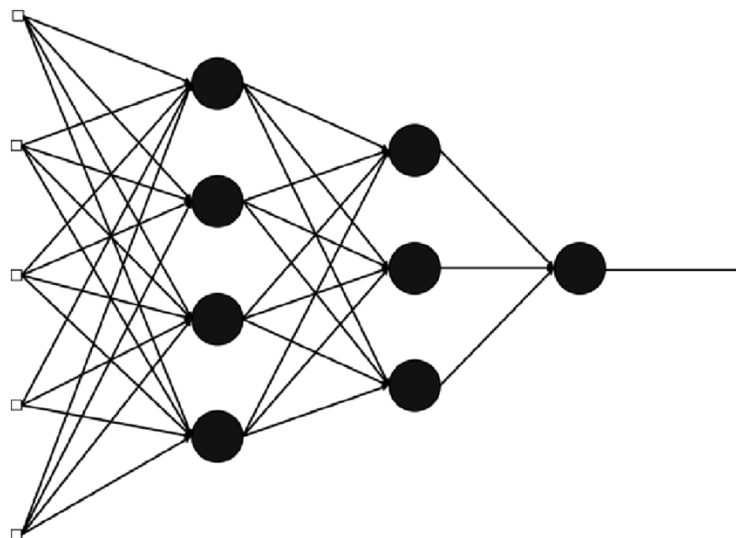
- přijímá vstup z předchozí vrstvy, zpracovává jej a předává dál
- vstup může být buď přijímán buď ze vstupní vrstvy, popř. z jiné vnitřní vrstvy.
- výstup je předán buď opět do další, vnitřní vrstvy, anebo konečné, výstupní.

c) Výstupní

- stejně jako vstupní vrstva je unikátní
- přijímá vstupy z vnitřní vrstvy a vyhodnocuje je

Každá z těchto vrstev následně obsahuje sadu neuronů a každý neuron obsahuje právě jednu sadu vah. Tato sada je složena z desetinných čísel, jejichž počet je ekvivalentní množství neuronů v předchozí vrstvě. Jedinou výjimkou je vrstva vstupní. Ta má neurony nahrazeny sadou vstupů (jeden vstup je ekvivalentem jednoho neuronu).

Tímto způsobem je vytvořen tzv. úplný k-partitní graf.



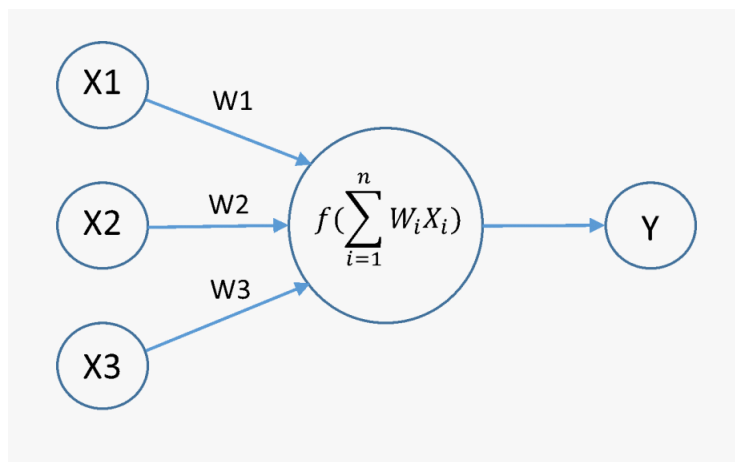
Obrázek 3.1: Struktura neuronové sítě[24]

3.4.1.1 Předvídání

Při předvídání výsledku se do počáteční (vstupní) vrstvy vloží vstupy. Tato vrstva je poslána do vrstvy vnitřní. V této vnitřní vrstvě se v každém neuronu všechny vstupy vynásobí s korespondujícími váhami a následně se sečtou. Výsledné číslo projde přes příslušnou formu aktivační funkce, která sníží velikost hodnoty na číslo v intervalu $<-1;1>$. Tato funkce bývá též nazývána "squashing function", z angl. zmáčknout/slisovat.

Vzhledem k náročnosti na pochopení výroku výše, je pro lepší ilustraci přiložen obrázek č. 3.2. Na obrázku jsou pomocí znaků $X_1 - X_3$ znázorněny jednotlivé vstupy neuronu (kompletní sada vstupů vrstvy). Znak $W_1 - W_3$ označují váhy, kterými se vstupy násobí. Váhy směřují do aktivační funkce, která udržuje hodnotu výstupu v daném intervalu. Písmeno Y označuje výstupní hodnotu neuronu.

Výstup jednotlivých neuronů vytváří sadu dalších vstupů pro další jednu vrstvu v pořadí. Tímto způsobem se postupně prochází veškeré vrstvy, až po vrstvu výstupní. Výstupní vrstva se liší od všech ostatních pouze tím, že obsahuje jeden neuron. Výstup toho neuronu lze uvažovat jako pravděpodobnost, na kolik je něco pravda či nikoli. V kontextu současné práce to znamená, jestli se daná transakce opakovala, či nikoli. Příkladem by se dalo uvést následující: Pokud získám ze sítě hodnotu blízkou -1 , transakce se pravděpodobně nebude opakovat. Analogicky pokud získám hodnotu blízkou 1 , transakce by se měla opakovat. Zároveň platí, že čím více je výsledná hodnota blíže 0 , tím menší je pravděpodobnost správné předpovědi.



Obrázek 3.2: Průchod neuronem[25]

Při bližším pohledu na fungování algoritmu si lze všimnout, že při násobení a sčítání vah platí následující vztah:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \cdot [w_1, w_2, w_3 \dots w_n] = S$$

Kde 'x' značí vstupy a 'w' váhy. 'S' je výsledek před vstupem do aktivační funkce.

Právě z toho to důvodu jsou v řešení uvažovány váhy i vstupy jako vektory, které se navzájem násobí. K násobení vektorů je využito externí knihovny Math.NET Numerics (dostupné z <https://numerics.mathdotnet.com/>). Aktivační funkce je pouhá hyperbolická funkce tangens z integrované knihovny Math.Tanh()

3.4.1.2 Učení

Aby síť měla nějakou vypovídací hodnotu, musí se nejdříve naučit, co má předpovídat. Prvotní předpovědi jsou čistě náhodné. Děje se tak proto, že při inicializaci jsou váhy v jednotlivých neuronech čistě náhodné. Je tedy nutné provést korekci vah a tím síť učit. Tomuto procesu se říká zpětná propagace ("backpropagation"). Nutnou podmínkou pro takové učení je pak samozřejmě, že je známý korektní výstup pro daný vstup.

Korekce probíhá tak, že síť nejdříve vyhodnotí nějaký vstup. Poté jí je řečeno, jaký byl očekávaný výsledek. Na základě rozdílu očekávaného výsledku

a skutečného výsledku se vyhodnotí chyba pro výstup. Ta se následně musí propagovat napříč celou sítí. Musí se ale také brát v úvahu, že ne každý neuron přispěl stejnou měrou k výsledné chybě. Je nutné najít přesnou chybu pro každý z neuronů. K tomuto výpočtu se dá využít vzorec:

$$\gamma = error \cdot (1 - output^2)$$

Příčemž *error* je chyba neuronu, *output* výstup neuronu po průchodu aktivací funkcí a γ by se dala pochopit jako chyba vstupu současného neuronu. Chybu jednotlivých vah lze najít vynásobením vstupů do neuronu (vektorem) a vypočítané, skalární, hodnoty γ , díky čemuž vznikne nový vektor. Tento vektor už po té stačí jen přičíst ke korespondující váze, čímž dojde k její opravě.

Je však stále třeba zjistit chybu každého celého neuronu ve vnitřní vrstvě, aby bylo možné využít výše zmíněný vzorec. Ve výstupní vrstvě je toho dosaženo, již zmíněným odečtením referenčního výsledku od předpokládaného. Pro vnitřní vrstvy tento způsob ale použít nelze. Zde toho je dosaženo tak, že se v každé vrstvě násobí vypočtená hodnota γ s korespondujícími váhami neuronu a vektorově se tyto hodnoty sečtou. Výsledný vektor se vždy předá předcházející vrstvě jako seznam chyb každého z neuronů. Celý proces předávání vektoru chyb funguje na obdobném principu jako předchozí vyhodnocování pravděpodobnosti. V tomto případě je postup ale opačný a vyhodnocuje se postupně od výstupní vrstvy ke vstupní.

Detailní odvození vzorce (ačkoli pro odlišnou implementaci neuronové sítě) přehledně vysvětluje uživatel, vystupující pod přezdívkou *The One* na svém videu o tvorbě neuronových stí.[26]

Jedním z problémům při učení neuronové sítě mohou být také vstupy samotné. Pokud se síť bude učit po samostatných vstupech a tyto vstupy budou rozděleny na samostatné celky, může se síť naučit nejdříve jeden a později přeučit na druhý.

Pokud se naopak využijí všechny vstupy najednou, může u velkých datasetů dojít ke značným problémům s mezipamětí. To může způsobit načítání vstupních dat samotných, ale mnohem pravděpodobněji alokaci prostředků ve vnitřních strukturách každého neuronu. Konkrétně jde o ukládání, vypočítaných korekcí jednotlivých vah, v generické kolekci *List*. Ta se při dosažení maxima svého objemu automaticky realokuje na násobek současného. Tím vzniknou několikanásobky velikosti původního setu dat. Navíc se zjistilo, že tento přístup není ani příliš efektivní z hlediska učení.[27] Pro učení je tedy využít přístup, který rozděluje data do několika, ne nutně stejně velkých, ale relativně malých celků ("minibatch"). Tento celek se nejdříve celý vyhodnotí a váhy jsou upraveny na základě průměru všech korekcí v něm.

Tento přístup uspoří paměť, je schopen se lépe učit a zároveň tím lze eliminovat problém, kdy chodí data jednoho druhu a posléze jiného (nemají

uniformní rozdělení). Síť by se totiž připravila jen na jeden druh dat. Což je zároveň problém, ke kterému došlo kvůli porovnávání transakcí (detailně rozebráno v sekci 3.4.4).

3.4.2 Export, import sítě

Aby bylo možné využít už jednou natrénovanou síť, je nutné si ji nějakým způsobem zapamatovat. Pro síť jsou důležité pouze dvě věci a totiž váhy neuronů společně s tvarem sítě. Tvarem sítě se rozumí množství vrstev a počet neuronů v nich obsažených.

Pro řešení uložení sítě si ji lze představit jako 3D objekt. Prvním rozměrem jsou vrstvy, druhým počet neuronů v každé vrstvě, a třetí jsou váhy v každém neuronu. Stejným způsobem se proto ukládají resp. načítají do/z databáze.

3.4.3 Testování funkčnosti neuronové sítě

Pro zjištění funkčnosti sítě musely být vytvořeny nějaké dva primitivní problémy, na základě kterých by se dalo ověřit, že opravdu funguje. Bylo tedy zvoleno předvídání sudých a lichých čísel, společně s testováním operátoru XOR pro dvě čísla.

Dále bylo třeba zjistit, zda při exportu sítě a následném importu nedojde k porušení struktury sítě a bude fungovat i nadále.

3.4.3.1 Sudá a lichá čísla

Prvním bylo rozhodování se, jestli je číslo mezi nulou a sedmi sudé nebo liché. Čísla 0-7 byla zvolena, protože je lze v binární soustavě zapsat pomocí tří cifer. Jediný rozdíl byl, že se číslo 0 zapisovalo jako -1. To je z důvodu, že nula není neutrální číslo pro násobení, a tím pádem by nedocházelo ke správnému vyhodnocování vah. Využití čísel -1 a 1 bylo doporučeno i na konferenci Microsoftu *Build 2013 - Developing Neural Networks using Visual Studio*[28]. Při testování se podařilo síti předpovědět, jestli je číslo sudé či liché na 100% (viz. obr. č. 3.3). Síť byla ve formě 3 - 3 - 3 - 1 s 1000 vstupy a minibatchem o velikosti 100. Vzhledem k jednoduchosti testu (výsledek závisí jen a pouze na jednom vstupu - ostatní jsou nepodstatné), změny na tvaru sítě neměly žádný větší vliv.

```
evaluate trained:
succes rate: 100% (p:1000 vs. f:0)
```

Obrázek 3.3: Test sudých a lichých čísel

3.4.3.2 XOR

Druhým problémem byl operátor XOR. Tento operátor označuje množinovou operaci která je pravdou pouze pokud jedno z čísel má nulovou, resp. zápornou (-1), hodnotu a druhé kladnou (1). Jak už bylo naznačeno test pro test byla nulová hodnota nahrazena zápornou ze stejných důvodů, jako v předchozím testu. První test probíhal na síti ve tvaru 2 - 3 - 3 - 1 s 1000 vstupy a mini-batchem o velikosti 100. Zde první číslo (2) značí vstupy a poslední (1) značí výstupy. Výsledek testování dosáhl úspěšnosti 67.1% (viz. obr. č. 3.4).

```
evaluate trained:  
succes rate: 67,1% (p:671 vs. f:329)
```

Obrázek 3.4: Test XOR operátoru s málo neurony

Bylo by možné tedy uvažovat, že téměř 3/4 vstupu byly odhadnuty správně, výsledek je proto lepší než čistý odhad a lze jej považovat za dostatečný. Problém ale je, že síti byly poskytnuty veškeré možné vstupy a tudíž musí předpovědět 100%.[29] Řešením tedy je třeba přidat větší množství neuronů a dat pro učení. Bylo proto zvoleno tvaru 2-3-4-2-1 s 10 000 vstupy a mini-batchem opět velikosti 100. Tenkrát bylo dosaženo lepšího výsledku (obr. č. 3.5).

```
evaluate trained:  
succes rate: 100% (p:10000 vs. f:0)
```

Obrázek 3.5: Test XOR operátoru s doplněnými neurony

Zároveň na zmíněném testu lze i vidět, že tvar a velikost sítě mají vliv na její výstup.

3.4.3.3 Export, Import

Export a import sítě lze ověřit snad jen jediným způsobem. A totiž exportováním již natrénované sítě a jejím následným naimportováním zpět. Pokud došlo ke korektnímu importu, síť bude nadále správně předvídat testovaná data. Bylo tedy implementováno simulované exportování a importování sítě a ověřeno, zda-li se předpovědi nezměnili. Výsledek opět dosáhl 100%, proto lze předpokládat, že funkčnost je bezchybná (viz obr. č. 3.6).

3.4.4 Komparátor

Aby neuronová síť mohla vyhodnocovat, jestli se daná transakce opakuje, je nutné z dat nejdříve určit, které transakce se konkrétně opakovaly. Tento

```

_ _ _ _ _
exporting:
done
evaluate trained:
succes rate: 100% (p:1000 vs. f:0)
Actualyes: 497, Actualno:503
predictionYes: 497, predictionNo:503
_ _ _ _ _
importing to new net:
done
test:
succes rate: 100% (p:1000 vs. f:0)
Actualyes: 488, Actualno:512
predictionYes: 488, predictionNo:512
_ _ _ _ _
done

```

Obrázek 3.6: Test exportu a importu sítě

úkol bohužel není jednoznačný. První myšlenka, která mnoho lidí napadne, je porovnávání výše transakce. Ne vždy platí, že pokud jsou dvě transakce stejně velké, automaticky se opakují. Lze ilustrovat na jednoduchém příkladu: Řekněme, že nějaký člověk, řekněme mu třeba Karel, si chodí ráno před prací kupovat snídani. Pokaždé si v obchodě koupí čerstvou bagetu. Problém ale nastává, pokud se bageta zlevní či zdraží, což je situace, která nastává relativně často. Bageta může být také někdy vyprodaná. Karel si stejně tak může každé ráno kupovat třeba pomeranče. To je položka, která nemá nikdy stejnou hodnotu a případné změny v ceně ještě zvyšují amplitudu. Navíc pomeranče mohou být ve špatném stavu, což ho bude nutit měnit potravinu, kterou kupuje. Tudíž není možné se rozhodovat podle identické ceny.

Druhá, související myšlenka, pak bývá: Pokud se nelze orientovat podle ceny, tak je možné se orientovat dle místa, potažmo čísla účtu. Tento případ opět není možný. Budeme-li dále používat Karla jako příklad, řekněme, že si kupuje ráno kávu. Pokud je venku pěkně, jde si ji koupit do stánku, kde se prodává káva levná a dobrá. Musí se ale posadit třeba na lavičku v parku. Pokud ale hezky není, jde si kávu koupit do kavárny, tedy úplně jiného podniku s úplně jiným číslem účtu. Stejně tak mohl zaspát a kávu si koupí až v práci, třebaže mu příliš nechutná. V tom případě si koupil kávu za úplně jinou cenu a ještě na úplně jiném místě s úplně jiným číslem účtu. Všechno to však lze považovat za opakující se transakci.

3.4.4.1 Přiřazování do páru

Jako možné řešení, které bylo implementováno, tedy je srovnávat transakce ze dvou různých časových úseků. Pro účely této práce byl zvolen rozsah jednoho týdne.

Jeden den je příliš malý rozsah, kdy se může stát, že dokonce nedošlo k žádné transakci. Pokud by se porovnával měsíc, transakcí je poměrně mnoho. Běžný člověk má navíc obvykle při opakující se činnosti zažité určité rituály. Nejrozšířenější opakující se činnost bývá chození do práce či školy přes týden. Opakující se rituál potom může být pití kávy po příchodu do práce. Takové činnosti se opakují právě v rámci týdnů a porovnávání by tedy mělo být také v rámci násobků týdnů.

V praxi se rozlišují i aktivity na sudé a liché týdny (cvičení ve škole, pravidelné schůzky v práci, jež se konají ob jeden týden). Ideální by tedy bylo porovnávání transakcí po dvou týdnech. Z důvodu nedostatečného objemu obdržených dat, byl nakonec zvolen rozsah jednoho týdne.

Pokud tedy budou uvažovány transakce ze dvou různých týdnů, měla by se opakující se transakce vyskytovat v každém týdnu právě jednou. V jednom z týdnů ale bývá z pravidla transakcí více. Proto by se dalo optimistickým přístupem (člověk po většinu času opakuje činnost) říci, že opakujících se transakcí je právě minimum z počtu transakcí v těchto dvou týdnech. Zároveň jsou v relaci 1:1, tj. každá opakující se transakce v prvním týdnu má právě jednu opakující se transakci v týdnu druhém. Zbylé transakce se neopakují.

Pro přiřazování do párů se transakce procházejí v cyklu a ke každé transakci se vybere ta, která je ji hodnotou nejbližší. V každém průchodu cyklu se pro každou transakci z menší množiny hledá transakce z množiny větší taková, že rozdíl v jejich částce je nejmenší (tzv. vzdálenost musí být nejkratší). Při hledání těchto dvojic může dojít k duplicitnímu propojení (dvě a více transakcí z jedné množiny, považují jednu transakci z druhé množiny za nejbližší). V tom případě se vezme dvě nejkratší vzdálenosti z tohoto vztahu. Celý cyklus se opakuje, dokud nejsou přiřazeny všechny transakce z menší množiny k množině větší.

3.4.4.2 Porovnávání vzdálenosti

S touto implementací však vyvstává jeden problém. Toto "optimistické" přiřazování transakcí ve skutečnosti říká, že týden s menší aktivitou je podmnožinou týdne s aktivitou větší. Jinými slovy - všechny platby jednoho z týdnů jsou beze zbytku obsaženy v druhém. To zcela jistě není správné. Opakující se transakce jsou spíše průnikem těchto dvou týdnů (ne všechny se opakují). Je tedy opět třeba se zamyslet, jakým způsobem toto vyřešit.

První možností by bylo omezit počet opakování v cyklu, kdy se párují transakce. Ať už říci, že se cyklus nesmí procházet např. více než třikrát anebo musí zůstat alespoň pět nerozhodnutých transakcí. Při využití těchto přístupů

ale není žádná kontrola nad samotnými transakcemi. Mohou zbýt transakce, které si jsou částkou blízké - to nastává třeba v případě, že dotyčný člověk jednoduše nenakupoval nic navíc a částky jsou si opravdu podmnožinou. Tak je možné, že mezi jednotlivými týdny nebyl finančně příliš aktivní a plateb je málo - všechny by se mohly vyfiltrovat jako neopakující se. To tedy nepoukazuje na dobré řešení.

Za mnohem korektnější přístup, by se dalo považovat porovnávání ještě podle relativní vzdálenosti jednotlivých částek. Tím je míněno, že hodnoty 10 Kč a 110 Kč se od sebe liší více, než například 5 000 Kč a 5 100 Kč. Obě hodnoty se v absolutním měřítku sice liší jen o sto korun, ale částky samotné se pohybují ve zcela jiných řádech. Avšak porovnávání řádů s hodnotou je poměrně abstraktní a jednoduché porovnání řádů s rozdílem částek nejde příliš jednoduše implementovat.

Za řešení lze považovat pohled z hlediska poměrů. Pokud se zůstane u předchozích částek, daly by se uvést do poměrů $110 : 10$ a $5\,100 : 5\,000$. Poměr pro první zmíněnou částku je roven 11, kdežto pro druhou 1.02. Zde už lze zřetelně vidět, jak moc se od sebe částky relativně liší.

Za povšimnutí však ještě stojí, že je třeba poměřovat vždy částku větší vůči menší, případně naopak. Je však nutné dodržovat pořadí velikostí. Po samotném přiřazení párů navíc není garantováno, která z částek je větší. Mohlo by tedy dojít k tomu, že by se porovnávala čísla normalizovaná do velikosti menší než jedna, s čísly většími než jedna. Takové porovnávání by dávalo zcela nesmyslné výsledky.

3.4.5 Testování komparátoru

Ačkoli komparátor je poměrně složitý, jeho výstup už tak složitý není. Vše lze porovnat pouhým okem z výpisu. Příklad vygenerovaného výpisu, s daty z poskytnuté transakční historie, je na obr. č. 3.7.

Lze zde pozorovat, jak se jednotlivé částky na sebe namapovaly (označeno šipkou) a jak jsou od sebe vzdáleny (v závorce). Transakce, které se nepodařilo přiřadit, jsou vypsány v části "Major". Ty jsou tedy označeny za neopakující se.

3.4.6 Datafeeder

Část, v práci pojmenovaná jako Datafeeder, vychází z anglických slov "*data*" a "*feeder*" – v doslovném překladu "data-krmítko". Tato funkcionality slouží pro načítání dat do neuronové sítě. Je také poměrně jednoduchá. Jako vstupní parametry dostane uživatel, jehož data má načítat, a počet "*epoch*", které má projít. Epochou se rozumí určité časové období. Z výše zmíněných důvodů byla pro účely této práce jedna epocha definována jako týden.

Po načtení vstupních parametrů, se vytvoří dotaz do databáze, který načte dvě epochy a pomocí komparátoru doplní označení o opakujících se transakcích

```
Create comparator:
Compare:
Write pairs:
-5 -> -17 (3,4)
-3 -> -1 (3)
-513,5 -> -193 (2,66062176165803)
-74,5 -> -47,38 (1,57239341494301)
-500 -> -357,6 (1,39821029082774)
-579 -> -782,46 (1,35139896373057)
-25 -> -20 (1,25)
-76,17 -> -87,5 (1,14874622554812)
-2276,27 -> -2029 (1,12186791522918)
-411,33 -> -451,85 (1,09850971239637)
-320,38 -> -295,39 (1,08460002031213)
-380 -> -362,42 (1,04850725677391)
Write undetermined:
Minor:
Major:
-5000
-4500
-2836,5
-1249
-189
-155,6
-150,25
-150
-144,5
Done
```

Obrázek 3.7: Test komparátoru

(viz. sekce 3.4.4 Komparátor). Dále data normalizuje a předá neuronové síti, jež tyto transakce postupně po jedné zpracuje.

3.4.7 Normalizace dat

Pro vyhodnocování neuronovou sítí je důležité mít data v rozmezí nějakého konstantního intervalu. Zde byl konkrétně zvolen ve velikosti $<-1,1>$. Interval je definován z toho důvodu, že pokud by došlo k jeho překročení některými ze vstupních dat, mohlo by to způsobit "přebíjení" ostatních vah v neuronu. Jinými slovy, výše aktivace neuronu by závisela na několika vstupních datech a ostatní by se ignorovala.

První položkou, která se normalizuje, je datum transakce. Datum sám o sobě není příliš vypovídající, proto by se tato akce dala považovat i za lehké

obohacení dat. Pro je normalizaci je totiž využito znalosti, že týden se dělí do sedmi dnů (pondělí – pátek). Z data transakce se tedy zjistí, v který den v týdnu transakce proběhla a zároveň se vytvoří sedm neuronů. Každý z nich bude mít hodnotu 1 nebo -1 (krajní body intervalu) podle toho, který den transakce proběhla.

Kupříkladu na vstupu přijde transakce, která proběhla v pondělí. Po normalizaci vznikne šest neuronů, s hodnotou -1, jež jsou obrazem všech dnů v týdnu mimo pondělí a jeden, zobrazující pondělí, jež má hodnotu 1.

Je třeba poznamenat, že hodnota 0 není zvolena záměrně. Uvnitř sítě se jednotlivé vstupy násobí s jednotlivými váhami. Nula není z hlediska násobení neutrální prvek a zároveň při násobení nulou vždy vznikne nula. Tím by došlo k znehodnocení vah a proto je místo nuly zvoleno -1.

Další položkou jsou Konstantní symboly. Konstantní symbol je, obdobně jako datum, číselně další neurčitou položkou. V sekci 2.5 Obohacení dat je však rozdělen do jednotlivých kategorií. Ty jsou definovány jako samostatné položky v několika tabulkách databáze. Z toho vychází i způsob implementace - jsou definovány jako výše zmíněný datum. Tedy pro každou položku je vytvořen neuron s hodnotou -1, kromě položek, pod který konstantní symbol spadá. Těm je přiřazena hodnota 1.

Poslední položka k normalizaci je výška transakce. Správnou hodnotu není lehké určit. Není totiž možné určit maximální velikost transakce. Úvaha pro normalizace byla tedy založena na základě výše škody, dle trestního práva. Z hlediska trestního práva jsou částky rozděleny následovně:

- 1) Škoda nikoli nepatrná – nejméně 5 000 Kč
- 2) Škoda nikoli malá – nejméně 25 000 Kč
- 3) Větší škoda – nejméně 50 000 Kč
- 4) Značná škoda – nejméně 500 000 Kč
- 5) Škoda velkého rozsahu – nejméně 5 000 000 Kč[30]

K rozdělení byla přidána ještě jedna částka 500 Kč. Ta vychází z limitu pro bezkontaktní platby kartou, kdy není třeba zadávat kód PIN.[31].

Tímto principem zároveň došlo k rozdělení přesně na šest druhů transakcí. Číslo šest je sudé a vzhledem k tomu, že je částky potřeba normalizovat do intervalu $<-1,1>$, přímo se vybízí rozdělit tyto kategorie na dvě části. Konkrétně 0 – 25 000 Kč v intervalu $<-1;0>$ a vše od 25 000 Kč výše v intervalu $(0;1>$

Částky jsou však stále vysoké. Aby bylo možné využít, bylo třeba vytvořit nějakou funkci, která by je rozprostřela do jednotlivých úsekových intervalů. K tomu se výborně hodí obdoba aktivační funkce (squashing function) ze sekce 3.4.1.1 Předvídání. Ta částku snižuje na velikost v intervalu $(0,1)$.

3. REALIZACE

Aby byly částky rozděleny do zmíněných šesti kategorií, stačí funkci rozdělit na šestiny, s aktivační funkcí přizpůsobenou jednotlivým částkám. Pro aktivační funkci bylo zvoleno tanh a částky roztaženy na celý interval pomocí "magických konstant" následovně:

1) 0 – 500 Kč

$$\frac{\tanh(\frac{x}{189})}{3} - 1$$

2) 500 – 5 000 Kč

$$\frac{\tanh(\frac{x-500}{1889})+1}{3} - 1$$

3) 5 000 – 25 000 Kč

$$\frac{\tanh(\frac{x-5000}{9446})+2}{3} - 1$$

4) 25 000 – 50 000 Kč

$$\frac{\tanh(\frac{x-25000}{18892})}{3}$$

5) 50 000 – 500 000 Kč

$$\frac{\tanh(\frac{x-50000}{188918})+1}{3}$$

6) 500 000 – 5 000 000+ Kč

$$\frac{\tanh(\frac{x-500000}{1889180})+2}{3}$$

Detailní rozložení lze též pozorovat na obr. č. 3.8 Normalizace částky

3.5 Předvídání bez umělé inteligence

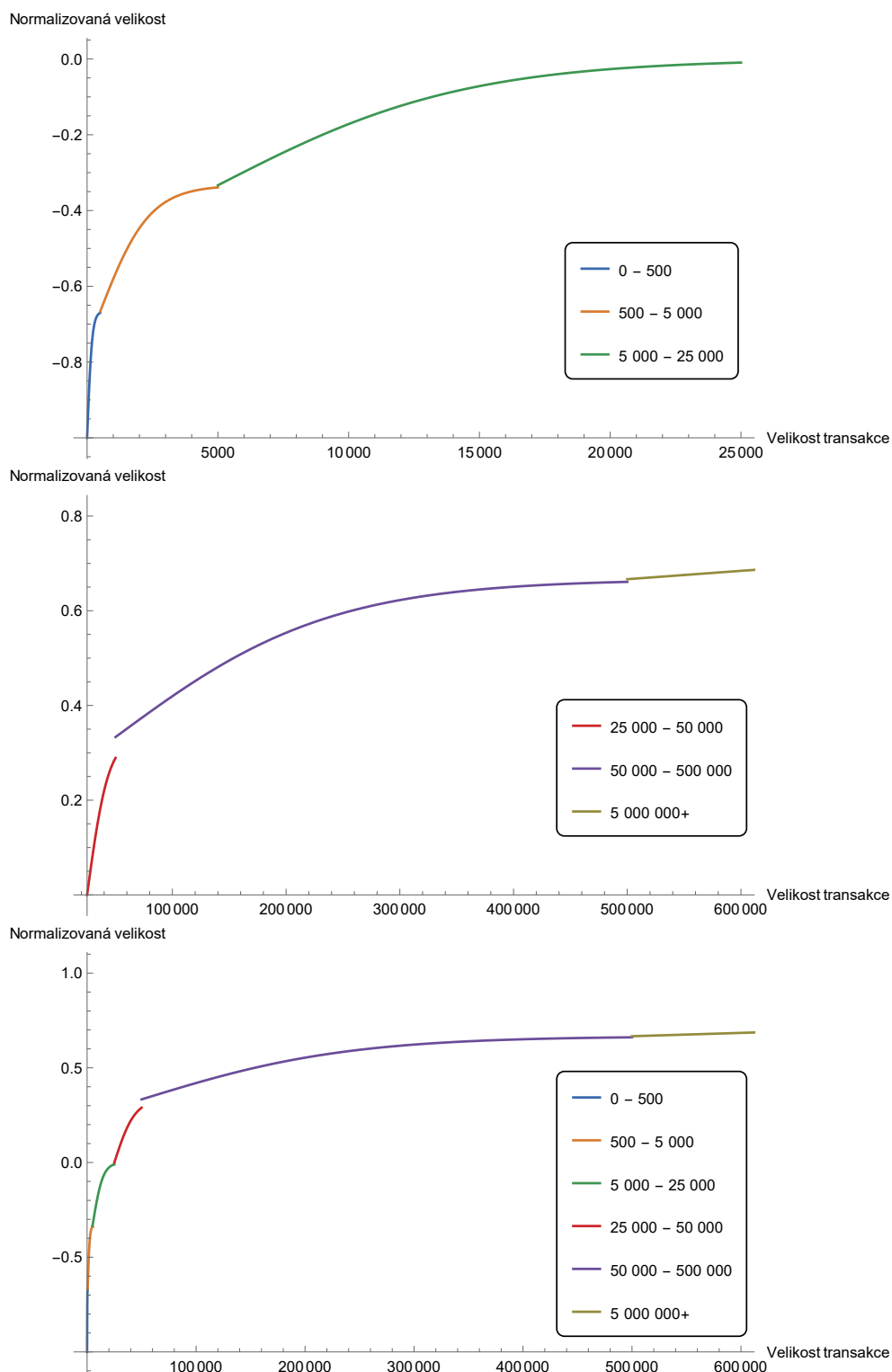
Jednotlivé předpovědi byly ještě rozšířeny o předpovědi bez strojového učení. Ty prochází data pokouší se z nich získat nějaké další závislosti.

3.5.1 Předvídání na základě velikostí transakce

Pro tuto předpověď bylo využito komparátoru ze sekce 3.4.4. Načtou se data za poslední týden a porovná se jejich částka s týdnem předchozím. Na základě dříve popsanych poměrů rozdílu v počtu transakcí se pak rozhodne, jestli došlo k opakování transakce, či nikoli.

Z transakcí určených jako opakující se, se určí kategorie na základě hodnot konstantních symbolů, jimiž byla data obohacena. Konkrétně se využívá jednotlivých tříd (třetí cifry v konstantním symbolu). Pokud tato třída není k dispozici nebo se konstantní symbol řadí mezi tzv. "rezervované symboly", kategorie se označí jako "nerozhodnutá."

3.5. Předvídání bez umělé inteligence



Obrázek 3.8: Normalizace částky

Další kategorií je pak rozdělení dle velikosti jako takové. Rozdělení se v tomto případě řídí velikostmi ze sekce 3.4.7 Normalizace. Názvy pak byly zvoleny následovně:

- 1) do 500 Kč = Drobné
- 2) 500 – 5 000 Kč = Malé
- 3) 5 000 – 25 000 Kč = Střední
- 4) 25 000 – 50 000 Kč = Velké
- 5) 50 000 – 500 000 Kč = Velmi velké
- 6) nad 500 000 Kč = Značně velké

3.5.2 Předvídání na základě průměru

Při předvídání na základě průměru byla data rozdělena do tří dimenzí. První je rok dimenzí je rok. Ten byl rozdělen do, po sobě jdoucích, čtyřtýdenních celků, reprezentujících měsíce. Každý týden byl následně rozdělen na jednotlivé dny (pondělí – neděle). Tedy každý rok má třináct čtyřtýdenních celků, rozdělených dále na dny v týdnu.

Průměrování probíhá po jednotlivých dnech mezi zmíněnými čtyřtýdenními celky. Např. hodnoty z pondělí z prvního týdne daného celku se zprůměrují s hodnotami jednotlivých pondělí pro veškeré první týdny a pro veškeré roky.

Tímto průměrováním se získá průměrná velikost a počet transakcí pro následující čtyři týdny. Uživateli je pak zobrazen pravděpodobný průběh útraty za tuto dobu a zároveň odhadová útrata na každý den v týdnu.

3.6 Databáze

Databáze by se dala rozdělit na čtyři samostatné celky. Každý celek je ale strukturovaný a propojený s ostatními. Níže jsou všechny popsány i s doprovodnými výřezy diagramu. Kompletní diagram databáze je však příliš rozsáhlý pro vložení do textu. Je proto přiložen mezi externími přílohami typu B jako obrázek s názvem *DB_diagram.png*.

3.6.1 Uživatel

První částí je ukládání dat o uživateli. Pomyslnou hlavní tabulkou je "User". Obsahuje uživatelská jména registrovaných uživatelů a heslo, které je zahasované a propojené se solí.

Pro každého uživatele si též v tabulce "Files" ukládá informace, jaké soubory nahrál, resp. jejich jméno a obsah. Je zde ale vystaveno omezení na ukládání souborů tak, aby bylo jejich jméno, v rámci uživatele, unikátní. Omezení

bylo vytvořeno z důvodu zabránění nahrávání duplicitních souborů, které je těžké odhalit. Uživatel sice může stále upravit jméno a soubor nahrát, tímto způsobem je ale upozorněn, že už systém data pravděpodobně obsahuje.

Další výhoda této tabulky tví v tom, že pokud si uživatel přeje smazat nahraný soubor, jsou známy transakce, které jsou k danému souboru navázány.



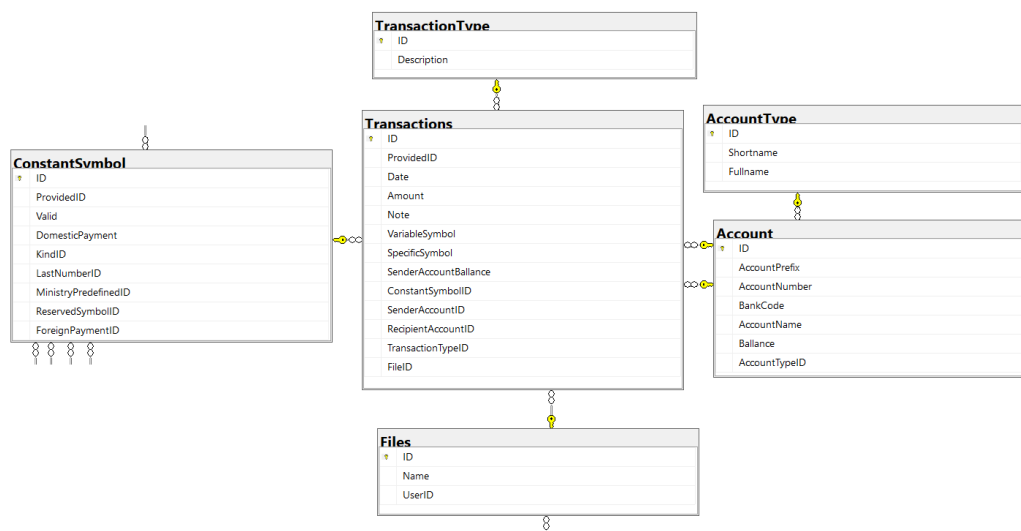
Obrázek 3.9: Model databáze - uživatel

3.6.2 Transakce

Další položka databáze jsou jednotlivé transakce. Ty představují pomyslné centrum databáze, protože jsou hlavní informací, od které se veškerá činnost odvíjí. Obsahuje načtená data transakce - datum transakce, částku, poznámku, symboly, účty příjemce i plátce a případný typ transakce.

Dále jsou zde napojeny odkazy na tabulky se záznamy, o jaký typ transakce se jednalo, i jaké byly účty příjemce a plátce.

3. REALIZACE



Obrázek 3.10: Model databáze - transakce

3.6.3 Konstantní symbol

Konstantní symboly v databázi jsou hlavně číselníkem. Slouží k obohacení transakcí a jaký mají význam již bylo zmíněno v sekci 3.3.4 Obohacení.

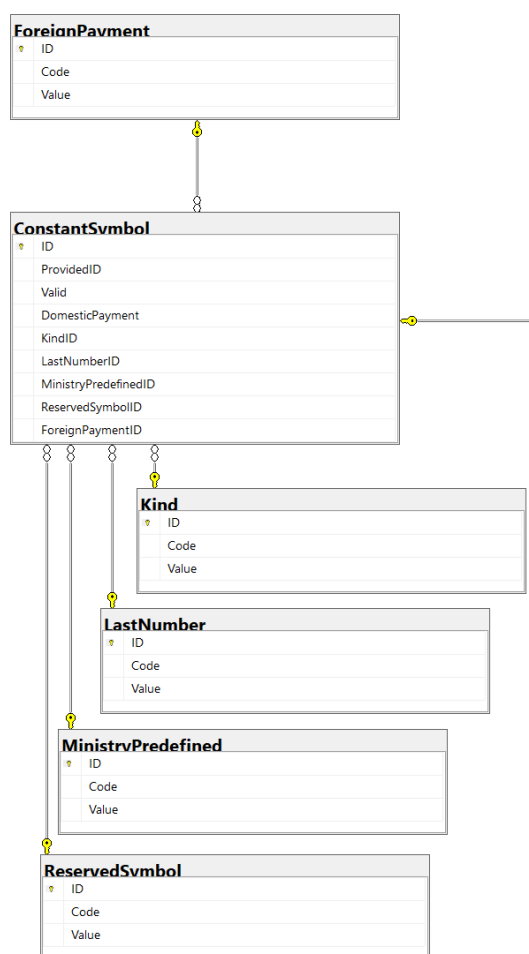
Tato část obsahuje tabulku "*ConstantSymbol*", což je entita, na kterou je odkazováno z "*Transactions*". Definuje, jestli případný konstantní symbol, který uživatel vložil je validní. Pokud ano, naváže na sebe jednotlivé číselníky, pro které je symbol definován. Těmito číselníky se rozumí "*Kind*" (druh transakce), "*LastNumber*" (význam poslední číslice), "*MinistryPredefined*" (Předdefinované třídy MFČR) a "*ReservedSymbol*" (Rezervované symboly).

Poslední tabulkou je "*ForeignPayment*", jež slouží k odlišení zahraničních transakcí.

3.6.4 Neuronová síť

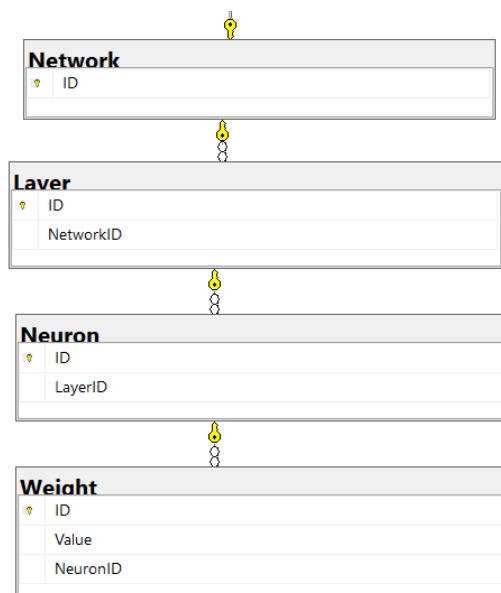
Poslední částí databáze je neuronová síť. Slouží k ukládání vypočítaných nastavení, aby natrénovanou sítí bylo možné později znovu použít. Systém ukládání funguje dle logiky zmíněné v sekci 3.4.2 Export, Import sítě.

V databázi je seznam sítí ("*Network*"). Každá se potom dělí na další podseznamy – vrstvy ("*Layer*"), a ty dále na neurony ("*Neuron*"). Jako poslední seznam je seznam vah ("*Weights*") ke korespondujícím neuronům.



Obrázek 3.11: Model databáze - konstantní symbol

3. REALIZACE



Obrázek 3.12: Model databáze - neuronová síť

Výsledné uživatelské rozhraní

Výsledné rozhraní uživatelské rozhraní má dohromady tři obrazovky. Přihlašovací, kterou uživatel vidí jako první při načtení aplikace. Druhou obrazovkou je registrační formulář, aby uživatel mohl aplikaci použít. Poslední obrazovka, tvořící hlavní část aplikace, je dostupná po přihlášení. Obsahuje grafy, které ukazují, jak se pravděpodobně uživatel bude chovat po finanční stránce v následujících dnech.

4.1 Přihlašovací obrazovka

Přihlašovací obrazovka obsahuje formulář pro zadání uživatelského jména a hesla. Dále obsahuje odkaz na stránku pro registraci, pro uživatele, kteří se do aplikace ještě nezaregistrovali.



Obrázek 4.1: Uživatelské rozhraní - přihlašovací obrazovka

4.2 Registrační obrazovka

Registrační obrazovka vychází z obrazovky přihlašovací. Obsahuje formulář pro vytvoření uživatelského jména, hesla a odkaz zpět na přihlašovací obrazovku, pro uživatele, jež klikli omylem na odkaz pro registraci.

4. VÝSLEDNÉ UŽIVATELSKÉ ROZHRAŇÍ

Předpovídač

Registrovat se

jméno
heslo
heslo znovu

odeslat

[Už jsem registrovaný](#)

Obrázek 4.2: Uživatelské rozhraní - registrační obrazovka

4.3 Hlavní obrazovka

Hlavní obrazovka je nejsložitější. V záhlaví obsahuje dropdown menu, jenž po rozkliknutí nabídne odhlášení, změnu hesla a zrušení účtu. V případě, že dojde k výběru odhlášení, uživatel je automaticky přesměrován na hlavní obrazovku. Pokud ale naopak vybere možnost pro změnu hesla nebo zrušení účtu, zobrazí se dialog pro potvrzení a zadání hesla.

Předpovídač

Menu

- Odhlásit se
- Změnit heslo
- Zrušit účet

Historie

Nahrát historii

Výbrat soubor

Útrata za poslední čtyři týdny

celková velikost transakcí

Smazat účet

Opravdu chcete smazat účet? Tato akce je nevratná!

Pro ověření zadejte prosím znovu heslo:

heslo

Storno Odeslat

Změnit heslo

staré heslo

nové heslo

nové heslo znovu

Storno Odeslat

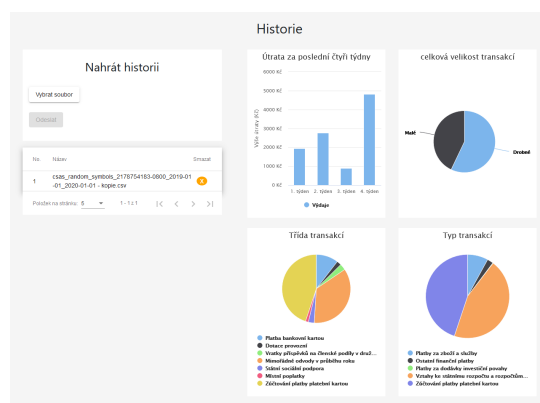
Třída transakcí

Výdaje

Obrázek 4.3: Uživatelské rozhraní - menu a účet

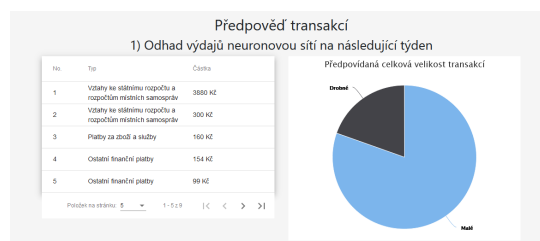
Pod záhlavím je obrazovka rozdělena na dalších pět částí. První z nich je část pro nahrání souboru obsahující výpis z účtu a pod ním je seznam všech úspěšně nahraných souborů. V tomto seznamu je zároveň tlačítko, které umožňuje uživateli libovolný soubor snadno odstranit.

Druhou částí jsou čtyři grafy, zobrazující, jak měl uživatel doposud výdaje. První z nich, sloupčový, ukazuje souhrnnou velikost transakcí za poslední čtyři týdny. Další tři jsou koláčové. První zobrazuje poměr velikostí transakcí (rozdělené na drobné, malé, střední, velké,...). Druhý i třetí graf pak využívají konstantní symboly k zobrazení, jakého druhu transakce pravděpodobně byly.



Obrázek 4.4: Uživatelské rozhraní - uživatelská historie

Zbylé části obrazovky tvoří samotné předpovědi. Nejdříve se zobrazuje předpověď základě umělé inteligence. Tu tvoří seznam pravděpodobných plateb na týden od nahrání historie, doplněný o typ plateb (odhadnutý z hodnot konstantních symbolů) a koláčový graf, který tyto transakce dává do poměru stejným způsobem, jako předešlý graf historie.

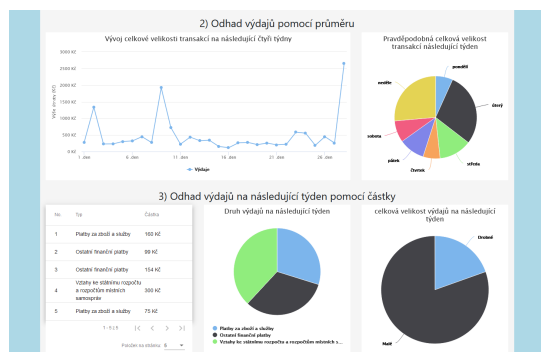


Obrázek 4.5: Uživatelské rozhraní - předpověď neuronovou sítí

Druhá předpověď, založená na průměru všech transakcí, zobrazuje čárový graf s odhadovanými výdaji na další čtyři týdny a koláčový graf pro předpovědi na jednotlivé dny v týdnu.

4. VÝSLEDNÉ UŽIVATELSKÉ ROZHRAŇÍ

Poslední předpověď, tvořící zároveň poslední sekci obrazovky, je předpovídání na základě porovnávání částek. Z této předpovědi vychází seznam pravděpodobných transakcí na další týden (strukturovaný stejně jako seznam pro předpovídání neuronovou sítí), koláčové grafy s druhem transakcí, vycházejících z konstantních symbolů a poměrnou velikostí transakcí.



Obrázek 4.6: Uživatelské rozhraní - předpověď průměrem a hodnotou

Vyhodnocení kvality

- vložit historii za určité období a porovnat ji s historií skutečnou za další období (oříznout dobu)

Doplnit

– vyscreenovat mathematicu ve FullHD místo 4K

6.1 scénáře testující použití

- dát cizím k otestování

Závěrečné testování

Proto, aby se zjistilo, jestli aplikace funguje správně, je nutné ji nejdříve řádně otestovat. K tomu bylo využito zejména manuálního testování.

7.1 Manuální testování

Při manuálním testování se kontrolovalo, jestli veškeré funkcionality pracují správně a jestli se změny projeví i v databázi. Výsledky na webu byly poté zachyceny a vloženy do práce.

7.1.1 Změna dat

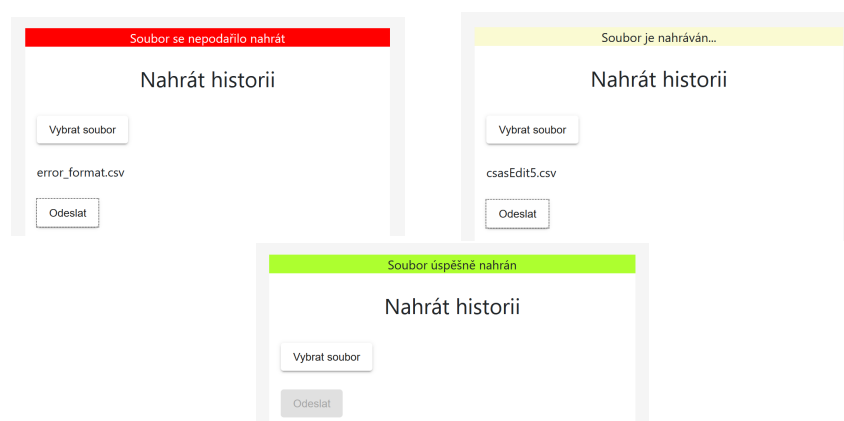
Když uživatel vybírá soubory s transakční historií, v základním nastavení jsou mu nabízeny soubory s příponou *.csv. Pokud vybere jiný anebo se pokusí koncovku souboru podvrhnout, aplikace soubor odmítne. Totéž se stane v případě, že se pokusí nahrát soubor se stejným názvem, který již dříve nahrál.

Pokud ale dojde k nahrávání platného souboru, je zobrazena hláška o nahrávání a po dokončení je uživatel informován, jestli byl soubor v pořádku nahrán, či nikoli.

Další možností, jak může uživatel změnit data, je změna hesla. Aby toho však dosáhnul, musí nejdříve vyplnit formulář, do kterého je třeba vložit jednou staré heslo, nové heslo a kontrolní heslo. Všechna pole se ve formuláři validují. Pokud se nové a kontrolní heslo neshodují, není dovoleno uživateli formulář odeslat. Obdobně pokud zadá nesprávně své staré heslo, změna mu není povolena.

Se změnou hesla do jisté míry souvisí i smazání celého uživatele. Aby mohl svůj účet smazat, musí se také nejdříve autentizovat svým starým heslem. Pokud jej zadá špatně, není dovoleno mu účet smazat. Pokud ale heslo zadá správně, je jeho účet smazán, spolu se všemi nahranými soubory a transakcemi, je odhlášen a přesměrován na přihlašovací obrazovku.

7. ZÁVĚREČNÉ TESTOVÁNÍ



Obrázek 7.1: Jednotlivé stavy nahrávání souborů



Obrázek 7.2: Validace formuláře na změnu hesla

7.1.2 Autentifikace

Aplikace má na starost i správu uživatelských účtů. V případě, že se uživatel pokusí přihlásit neplatným heslem, zobrazí se mu chybová hláška a systém ho nepustí dále. Obdobně, pokud se pokusí registrovat s uživatelským jménem, které již je použito, zobrazí se mu chyba, informujícího o této skutečnosti.

Při přihlašování i registraci jsou zároveň validovány formuláře samotné. Pokud se tedy uživatel vynechá jedno z povinných polí, zobrazí se mu k danému poli upozornění. Navíc, pokud formulář obsahuje chybu, uživatel nemá možnost jej odeslat (tlačítko je neaktivní).

The image shows two side-by-side screenshots of a web form. The left screenshot is titled 'Přihlásit se:' and shows a red error message 'Neplatné uživatelské jméno nebo heslo' at the top. Below the title, there are two input fields: the first contains 'userB' and the second contains ten black dots representing a password. Below the fields is a grey 'odeslat' button and a blue link 'Registrovat se'. The right screenshot is titled 'Registrovat se:' and shows a red error message 'Uživatelské jméno už existuje.' at the top. Below the title, there are three input fields: the first contains 'userB', the second contains ten black dots, and the third contains ten black dots. Below the fields is a blue-outlined 'odeslat' button and a blue link 'Už jsem registrovaný'.

Obrázek 7.3: Validace přihlašovacích údajů

The image shows two side-by-side screenshots of a web form. The left screenshot is titled 'Přihlásit se:' and shows two input fields. The first field is labeled 'jméno' and has a red error message 'Vyplňte uživatelské jméno' below it. The second field is labeled 'heslo' and has a red error message 'Vyplňte heslo' below it. Below the fields is a grey 'odeslat' button and a blue link 'Registrovat se'. The right screenshot is titled 'Registrovat se:' and shows three input fields. The first field is labeled 'jméno' and has a red error message 'Vyplňte uživatelské jméno' below it. The second field is labeled 'heslo' and has a red error message 'Vyplňte heslo' below it. The third field is labeled 'heslo znovu' and has a red error message 'Vyplňte kontrolní heslo' below it. Below the fields is a grey 'odeslat' button and a blue link 'Už jsem registrovaný'.

Obrázek 7.4: Validace polí na formuláři

Pokud se uživatel pokusí přejít na stránku, pro registrované uživatele, aplikace to zaznamená, a pokud není přihlášen, vrátí jej zpět na přihlašovací obrazovku. Pokud se ale uživatel neodhlásil a nevypršela mu doba trvání přihlášení, aplikace jej na zabezpečenou stránku pustí a zobrazí mu jeho data.

7.1.3 Uživatel bez JavaScriptu

Existuje i možnost, že na stránku přistoupí uživatel, který nemá zapnutý JavaScript. To je jazyk, ve kterém je napsán celý Angular framework a tím pádem bez ní aplikace nebude fungovat. Pokud však uživatel bez zapnutého JavaScriptu na stránku přistoupí, zobrazí se mu chybová hláška, že je třeba mít povolen JavaScript.

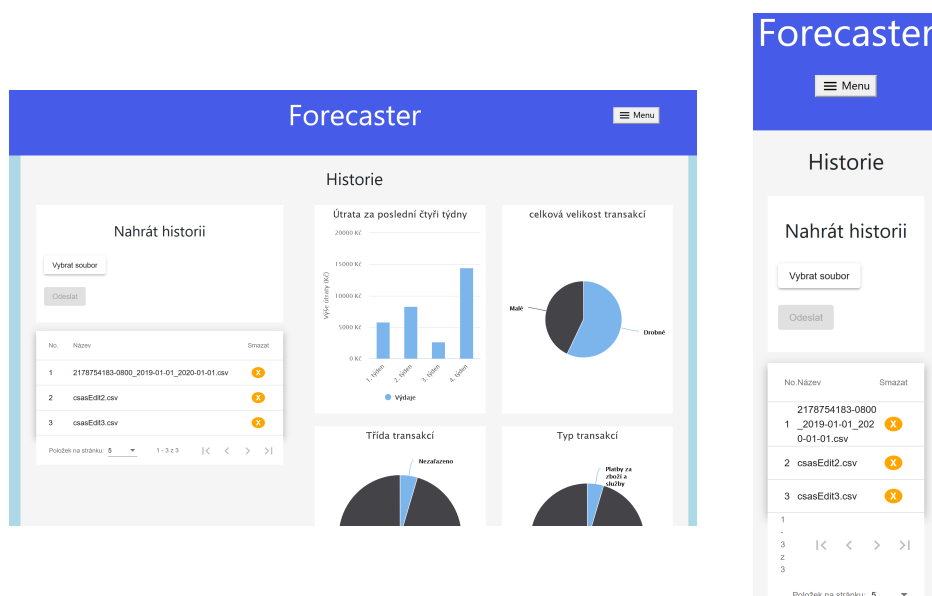
7. ZÁVĚREČNÉ TESTOVÁNÍ

Pro zobrazení stránky je třeba mít povolen JavaScript

Obrázek 7.5: Nepovolený JavaScript v prohlížeči

7.1.4 Responzivita

Jednou z podmínek práce je, aby byla aplikace responsivní. Tedy, že web bere ohled na velikost prohlížeče uživatele. Výsledek této operace se textově špatně demonstruje. Jako důkaz tedy alespoň poslouží několik obrázků.



Obrázek 7.6: Responzivita – porovnání obrazovek

7.2 W3C validace

Pro otestování validity kódu jsem chtěl aplikaci zkontrolovat pomocí W3C validátoru (dostupný z <https://validator.w3.org/>). Validátor bohužel nebere v úvahu syntaxe Angularu a generuje hlášení o nepovolených attributech. Z tohoto důvodu ho není bohužel možné pro test využít.

Error Attribute `wfd-id` not allowed on element `div` at this point.

From line 55, column 313; to line 55, column 379

```
/h1></div><div ngcontent-ng-cli-universal-cl="" class=" wrapper" wfd-id="1"><div _
```

Attributes for element `div`:

[Global attributes](#)

Error Attribute `ngcontent-ng-cli-universal-cl` not allowed on element `div` at this point.

From line 55, column 380; to line 55, column 447

```
fd-id="1"><div ngcontent-ng-cli-universal-cl="" class="container" wfd-id="2"><div _
```

Attributes for element `div`:

[Global attributes](#)

Error Attribute `wfd-id` not allowed on element `div` at this point.

From line 55, column 380; to line 55, column 447

```
fd-id="1"><div ngcontent-ng-cli-universal-cl="" class="container" wfd-id="2"><div _
```

Attributes for element `div`:

[Global attributes](#)

Obrázek 7.7: W3C – nepovolené atributy

Možnosti rozšíření

Možnosti rozšíření aplikace jsou velmi rozsáhlé. Může to zahrnovat zefektivnění výpočtů či doplnění webového rozhraní o další informace. Implementace všech těchto rozšíření ale značně objemem převyšuje rámec bakalářské práce.

8.1 Backend

Pro serverovou část je jako další rozšíření vhodné implementovat nějakou formu cachování (ukládání si již jednou vypočtených informací). Data zobrazovaná na webu se bez změny transakční historie nemění. Implementace je ale provedena tak, že pro každé zobrazení je musím znovu vypočítat. Tento způsob není velmi efektivní a bylo by vhodné je nějakým způsobem ukládat třeba do databáze a měnit je jen v případě, že dojde ke změně transakční historie.

Dále by bylo vhodné získat mnohem větší množství neanonymizovaných dat. Data, jež jsou v současnosti k dispozici nejsou bohužel příliš obsáhlá a limituje to schopnost umělé inteligence se učit.

S tím by ale souviselo i zamyšlení se nad vícevláknovým (paralelním) zpracováním učení umělé inteligence, jež by mohlo celý proces výrazně urychlit. Za zmínku by určitě stálo oddělení maticového násobení a jeho zpracování na grafických kartách. Tím by došlo ke snížení zátěže na procesoru.[32]

Optimalizaci by ale mohlo pomoci přepsání umělé inteligence třeba do C/C++ a použití C/C++ knihoven. Díky menší abstrakci v kódu by tímto teoreticky mohlo být dosaženo rychlejšího zpracování.

Také by bylo možné přidat automatické učení pro nové figuranty na základě úspěšnosti učení. To by mohlo být realizováno nějakým rozhodovacím stromem. Např. naučit se na uživateli, jehož předpověď je horší než 50% a této nové síti přiřadit veškeré uživatele, pro něž je předpověď této nově vycvičené sítě lepší.

Další možností by, zejména pro vývoj, bylo využít Dockeru a celou aplikaci spouštět v něm. Popřípadě sepsat UNIT testy, ať už pro lepší pochopení použití funkcí nebo testování jednotlivých částí jako takových.

8.2 Frontend

Webová část by mohla zcela jistě být rozšířena také o další funkcionality, ačkoli rozšíření funkcionalit na frontendu aplikace téměř vždy implikuje i rozšíření backendu.

Jednou z možností je například doporučování produktů na základě provedených transakcí. A ačkoli výstup by v tomto případě byl velmi jednoduchý (jeden textový box), implementace logiky za tím by byla velmi rozsáhlá. Bylo by třeba implementovat zcela novou rekomandační umělou inteligenci, která by byla navázaná na výsledky, už implementované, umělé inteligence na předpovídání transakcí.

Další funkcionalitou by mohlo být označování transakcí uživatelem. Mohly by to být třeba kategorie (drogerie, potraviny). Uživatel by též mohl označovat transakce, které považuje za opakující se a které nikoliv. Tím mohl vylepšit výsledky předpovídání budoucích transakcí.

Další možností by mohlo být propojení s emailovým účtem - notifikace po určitém počtu špatných přihlášení, přihlášení na novém zařízení nebo obnova hesla emailem.

Nakonec by stálo za zmínku vytvořit formulář, do kterého by uživatel doplnil o sobě nějaká data. Mohlo by to být pohlaví, věk, velikost města, ve které bydlí,... To by byl další způsob, jak zvýšit množství vstupů a vylepšit tak předpovědi umělé inteligence.

Závěr

Pro svou práci jsem si nastudoval syntaxi a použití jazyků, jmenovitě Typescriptu, Angularu a C# ve spojení z Entity frameworkem, na kterém jsem založil webovou aplikaci jako takovou. Od firmy Trask Solutions a.s. jsem si zajistil data a ty obohatil o druhy konstantních symbolů. Následně jsem na nich učil umělou inteligenci. Předpovídání transakcí umělou inteligenci jsem ještě doplnil o předvídání na základě průměrů a porovnávání částek.

V práci se mi podařilo umělou umělou inteligenci vytrénovat do té míry, že je do určité míry schopna předpovědět chování uživatelů. Pro lepší předpovědi by však bylo vhodné mít rozsáhlejší data.

Jako možná další zásadní rozšíření bych uvedl doporučování produktů na základě mnou implementované umělé inteligence. Na základě velikosti útraty, počtu transakcí za určité období a produktů ostatních uživatelů by mohla, např. pomocí kolaborativního filtrování, doporučovat uživatelům bankovní produkty.

Literatura

- [1] McKinnon, J.: JSON Web Tokens vs. Session Cookies: What's the Difference? *WP Rocket [online]*, 21.květen 2019, [cit. 2019-13-11]. Dostupné z: <https://wp-rocket.me/blog/difference-json-web-tokens-vs-session-cookies/>
- [2] Krishnan, S.: Auth Headers vs JWT vs Sessions: Choosing Right Auth Technique for APIs. *HashedIn Technologies Pvt. Ltd. [online]*, 18.leden 2018, [cit. 2019-13-11]. Dostupné z: <https://hashedin.com/blog/auth-headers-vs-jwt-vs-sessions-choosing-right-auth-technique-for-apis/>
- [3] Česká spořitelna a. s.: *George [online]*. [cit. 2019-23-04]. Dostupné z: <https://www.csas.cz/cs/internetove-bankovnictvi/george>
- [4] Fincentrum a. s.: *investujeme.cz [online]*. [cit. 2019-11-10]. Dostupné z: <https://www.investujeme.cz/clanky/george-prichazi-ceska-sporitelna-kompletne-meni-sve-internetove-bankovnictvi/>
- [5] Internet Info, s.r.o.: *Měšec.cz [online]*. [cit. 2019-11-10]. Dostupné z: <https://www.mesec.cz/clanky/george-startuje-naostro-sporitelna-spousti-nove-bankovnictvi-pro-1-5-milionu-lidi/>
- [6] A Medium Corporation: *Medium [online]*. [cit. 2019-11-10]. Dostupné z: <https://medium.com/product-soup/managing-money-and-technology-today-34be4d121c10>
- [7] Artnetics: *OpenNN*. [cit. 2019-13-12]. Dostupné z: <http://www.opennn.net/>
- [8] Paszke, A.; Gross, S.; Chintala, S.; aj.: *PyTorch*. [cit. 2019-13-12]. Dostupné z: <https://pytorch.org/>
- [9] Chollet, F.: *Keras*. [cit. 2019-13-12]. Dostupné z: <https://keras.io/>

- [10] Team, G. B.: *TensorFlow*. [cit. 2019-13-12]. Dostupné z: <https://www.tensorflow.org/>
- [11] Amazon Web Services, Inc.: *Amazon Machine Learning*. [cit. 2019-13-12]. Dostupné z: <https://aws.amazon.com/machine-learning/>
- [12] Microsoft Corporation: *ML.NET*. [cit. 2019-13-12]. Dostupné z: <https://docs.microsoft.com/en-us/cognitive-toolkit/>
- [13] Microsoft Corporation: *The Microsoft Cognitive Toolkit*. [cit. 2019-13-12]. Dostupné z: <https://docs.microsoft.com/en-us/cognitive-toolkit/>
- [14] Microsoft Corporation: *ML.NET*. [cit. 2019-13-12]. Dostupné z: <https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet>
- [15] Microsoft Corporation: *Tutorial: Categorize iris flowers using k-means clustering with ML.NET*. [cit. 2019-13-12]. Dostupné z: <https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/iris-clustering>
- [16] Microsoft Corporation: *Custom ML made easy with AutoML*. [cit. 2019-13-12]. Dostupné z: <https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet#automl>
- [17] Ministerstvo financí České republiky: *Seznam všech IČO - ARES (531MB) [online]*. [cit. 2019-13-11]. Dostupné z: https://wwwinfo.mfcr.cz/ares/ares_vreo_all.tar.gz
- [18] Ministerstvo financí České republiky: *ARES XML schéma [online]*. [cit. 2019-13-11]. Dostupné z: https://wwwinfo.mfcr.cz/ares/xml_doc/schemas/ares/ares_answer_vreo/v_1.0.0/ares_answer_vreo.xsd
- [19] Shafranovich, Y.: *Common Format and MIME Type for Comma-Separated Values (CSV) Files [online]*. SolidMatrix Technologies, Inc., říjen 2005, [cit. 2019-13-12]. Dostupné z: <https://tools.ietf.org/html/rfc4180>
- [20] Wikimedia Foundation, Inc.: *JSON Web Token [online]*. [cit. 2019-13-11]. Dostupné z: https://en.wikipedia.org/wiki/JSON_Web-Token
- [21] Close, J.: *CsvHelper*. [cit. 2019-15-12]. Dostupné z: <https://joshclose.github.io/CsvHelper/>
- [22] Finance.cz: Konstantní symboly: co který znamená a proč je už nepotřebujeme? *Finance.cz [online]*, 23. června 2018, [cit. 2019-15-12]. Dostupné z: <https://www.finance.cz/511323-konstantni-symboly-prehled/>

-
- [23] Sanderson, G.: But what is a Neural Network? [online], 5.říjen 2017, [cit. 2019-14-11]. Dostupné z: <https://youtu.be/aircAruvnKk>
- [24] clipartwiki.com: Artificial neural network. [online], [cit. 2019-14-11]. Dostupné z: https://www.clipartwiki.com/iclip/hxmwm_clipart-freeuse-download-network-clipart-artificial-neural-network/
- [25] clipartwiki.com: Neural network single neuron. [online], [cit. 2019-14-11]. Dostupné z: https://www.clipartwiki.com/iclip/hxmwm_clipart-freeuse-download-network-clipart-artificial-neural-network/
- [26] One, T.: Neural Network - Back-Propagation Tutorial In C#. [online], 26.srpen 2017, [cit. 2019-15-11]. Dostupné z: https://youtu.be/L_PByyJ9g-I
- [27] Dominic Masters, C. L.: Revisiting Small Batch Training for Deep Neural Networks. *arXiv.org [online]*, 20. dubna 2018, [cit. 2019-16-11]. Dostupné z: https://www.idnes.cz/technet/internet/youtube-software-google-umela-intelligence.A180514_114619_sw_internet_hege
- [28] McCaffrey, D. J.: 2013 Build Conference - Developing Neural Networks using Visual Studio. [online], 26.června 2013, [cit. 2019-16-11]. Dostupné z: https://youtu.be/-zT1Zi_ukSk
- [29] Ahire, J. B.: The XOR Problem in Neural Networks. *A Medium Corporation [online]*, 26. prosince 2017, [cit. 2019-16-11]. Dostupné z: <https://medium.com/@jayeshbahire/the-xor-problem-in-neural-networks-50006411840b>
- [30] Policie České republiky: *Majetkové trestné činy [online]*. [cit. 2019-19-12]. Dostupné z: <https://www.policie.cz/clanek/pomoc-obetem-tc-majetkove-trestne-ciny.aspx>
- [31] Československá obchodní banka, a. s.: *Nejčastější dotazy: Platební karty, platby na internetu a platební nálepky [online]*. [cit. 2019-19-12]. Dostupné z: <https://www.csob.cz/portal/documents/10710/25109/nejcastejsi-dotazy-platebni-karty.pdf>
- [32] Fatahalian, K.; Sugerman, J.; Hanrahan, P.: Understanding the Efficiency of GPU Algorithms for Matrix-Matrix Multiplication. *Graphics Hardware (2004) [online]*, 2004, [cit. 2020-02-01]. Dostupné z: <https://graphics.stanford.edu/papers/gpumatrixmult/>

Seznam použitých zkratek

JSON JavaScript Object Notation

JWT JSON Web Token

SQL Structured Query Language

NoSQL non SQL

CSRF Cross-site request forgery

ARES Administrativní registr ekonomických subjektů

XOR Exkluzivní disjunkce

CSV Comma-separated values

KS konstantní symbol

VS variabilní symbol

SS specifický symbol

API Application Programming Interface

HTTP Hypertext Transfer Protocol

ID identifikátor

MFČR Ministerstvo financí České Republiky

Přiložené soubory

KB_ARES.xml Výpis z ARES (Komerční banka)

DB_diagram.png Kompletní diagram datového modelu

Obsah přiloženého CD

readme.txt	stručný popis obsahu CD
src.....	zdrojové kódy implementace
publish.....	adresář se spustitelnou formou implementace
Recommender.bak.....	Databáze se základními daty
BP_Radek_Pus_2019.tex	zdrojová forma práce ve formátu L ^A T _E X
text	text práce
BP_Radek_Pus_2019.pdf.....	text práce ve formátu PDF
Přílohy.....	Přiložené soubory (Příloha B)