VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

PŘÍPADOVÁ STUDIE NA DOLOVÁNÍ Z DAT V JAZYCE PYTHON

DATA MINING CASE STUDY IN PYTHON

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ANASTASIIA STOIKA**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. JAROSLAV ZENDULKA, CSc.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

Studentka: **Stoika Anastasiia**
Program: Informační technologie
Název: **Případová studie na dolování z dat v jazyce Python**
Data Mining Case Study in Python
Kategorie: Data mining

Zadání:

1. Seznamte se se základy získávání znalostí z dat (data mining).
2. Seznamte se dostupnými prostředky na podporu dolování z dat v jazyce Python.
3. Po dohodě s vedoucím zvolte některou z případových studií prezentovaných v knize Data Mining with R.
4. Podrobněji se seznamte s přístupy a algoritmy pro řešení zvolené úlohy a potřebnými knihovnami jazyka Python.
5. V jazyce Python vytvořte řešení úlohy demonstrující využití prostředků jazyka Python pro dolování z dat.
6. Na základě získaných zkušeností zhodnoťte vhodnost použití jazyka Python pro řešení podobných úloh.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1. (Kapitola 1 a další informace relevantní ke zvolené úloze a použitým datům).
- Layton, R.: Learning Data Mining with Python: Use Python to manipulate data and build predictive models. 2nd Edition. Packt Publishing, 2017, 358 p., ISBN 978-1787126787.
- Nielsen, F.A.: Data Mining with Python. 2015. 101 p. Dostupné na http://www.freetechbooks.com/data-mining-with-python-working-draft-t1159.html.
- Torgo, L.: Data Mining with R. Learning with Case Studies. Chapman & Hall/CRC Press. 2011. 289 p.
- Podpůrné materiály ke knize http://www.dcc.fc.up.pt/~ltorgo/DataMiningWithR/.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Zendulka Jaroslav, doc. Ing., CSc.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 5. března 2019

Abstrakt

Tato práce se zabývá základními koncepty a technikami procesu získávání znalostí z dat. Cílem práce je demonstrovat dostupné prostředky jazyka Python, které umožňují provádět jednotlivé kroky tohoto procesu. Práce je zaměřena především na metody a techniky detekce odlehlých pozorování, založené na shlukování a klasifikaci. Jedná se o řešení analytické úlohy, která se týká zdrojů dat s omezeným množstvím využitelné informace. Tato kontrolní činnost by měla sloužit k detekci podezřelých prodejních transakcí nějaké společnosti, které mohou znamenat pokusy o podvod jejích prodejci.

Abstract

This thesis focuses on basic concepts and techniques of the process known as knowledge discovery from data. The goal is to demonstrate available resources in Python, which enable to perform the steps of this process. The thesis addresses several methods and techniques focused on detection of unusual observations, based on clustering and classification. It discusses data mining task for data with the limited amount of inspection resources. This inspection activity should be used to detect unusual transactions of sales of some company that may indicate fraud attempts by some of its salespeople.

Klíčová slova

získavání znalostí z dat, datová analýza, detekce odlehlých hodnot, detekce podvodních transakcí, detekce anomalií, analýza odlehlých hodnot, učení bez učitele, učení s učitelem, kombinace učení s učitelem i bez, klasifikace, Bayesovská klasifikace, lokální faktor odlehlosti, předzpracování dat, čištění dat

Keywords

KDD, knowledge discovery in databases, data mining, data analysis, outlier detection, anomaly detection, outlier analysis, detecting fraudulent transactions, unsupervised learning, supervised learning, semi-supervised learning, classification, Naive Bayes, Local Outlier Factor, Isolation Forest, data preprocessing, data cleaning

Citace

STOIKA, Anastasiia. *Případová studie na dolování z dat v jazyce Python.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Jaroslav Zendulka, CSc.

Případová studie na dolování z dat v jazyce Python

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Jaroslava Zendulky, CSc. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Anastasiia Stoika
19. května 2019

Poděkování

Chtěla bych poděkovat svému vedoucímu panu docentovi Jaroslavu Zendulkovi za cenné rády poskytnuté během vypracování této práce. Velice si cením jeho pozitivní přístup, skutečný zájem a ochotu pomoct při psání odborného textu.

Obsah

Kapitola 1

Úvod

Co je informace? Za informaci považujeme interpretované kvantitativní vyjádření obsahu zprávy. Jednotka informace může být interpretována různými způsoby. Například jako binární hodnota může vyjadřovat dvě alternativy – 0 a 1. Pojem informace často souvisí s pojmem data, kterým označujeme jakýkoliv zaznamenaný informační údaj. Když jsou data zpracovávána, organizována, strukturována nebo prezentována v daném kontextu tak, aby byla užitečná, nazývají se *informace*. Informace jsou interpretovaná data. V této práci se budu bavit o datech a problematice získávání znalostí z dat.

Výsledkem rychlého rozvoje výkonných automatických nástrojů pro shromažďování a ukládání dat se pozoruje nesmírný narůst objemu dostupných dat. Současně s růstem objemu dat v elektronické podobě se začaly značně zvětšovat také kapacity uložišť. Postupně si lidé začali uvědomovat, že data, dříve využívaná pouze s cílem evidence něčeho, mohou mít potenciálně užitečnou hodnotu. Jedná se o data, která mají perzistentní podobu a jsou trvalé uchovávána v uložišti. Shromážděná data v tak velkém objemu mohou být potenciálním zdrojem užitečné informace, která není viditelná na první pohled. Taková data mohou být užitečná pro rozhodování na vyšší úrovni. Jedná se o rozhodování v různých oblastech jako ekonomika nebo výzkum. Vyhodnocení experimentů v určité oblasti může být užitečným podkladem pro provedení řády dalších experimentů. Shromážděné informace o uživatelem navštívených internetových obchodech může být využito s cílem personalizace reklamy. Následně se uživateli může zobrazovat opakovaně reklama prohlíženého zboží s cílem přimět ho k opakovanému prohlížení tohoto zboží a opakovanému uvažování o jeho koupi.

Jedním z cílů této bakalářské práce je seznámit se s základními pojmy, kroky a metodami v problematice získávání znalostí dat. Dále je cílem seznámit se s dostupnými prostředky v jazyce Python pro podporu dolování z dat a demonstrovat jejich využití pro řešení zvolené případové studie, prezentované v knize *Data Mining with R* [36]. Závěrečnou úlohou práce je vyhodnotit vhodnost použití jazyka Python pro řešení podobných úloh.

Kapitola 2 bude věnována základům problematiky získávání znalostí z dat. Zde budou vymezeny klíčové pojmy související s dolováním. V kapitole 3 budou uvedeny některé dostupné prostředky pro dolování z dat, které poskytuje jazyk Python. Kapitola 4 zahrnuje úvod do zvolené případové studie. Je zaměřená na přípravu teoretického základu, jehož pochopení je důležité pro řešení daného problému. Dále v kapitole 5 bude prezentováno řešení případové studie s využitím dostupných prostředků v jazyce Python. Kapitola 6 obsahuje vyhodnocení jazyka Python pro řešení úloh podobného charakteru. Poslední kapitola 7 je závěrečnou a shrnuje zhodnocení dosažených výsledků této práce.

Kapitola 2

Získávání znalostí z dat

V této kapitole budou vysvětleny důležité pojmy související se získáváním znalostí z databází, jejichž pochopení je základem pro řešení dané bakalářské práce. Hlavní pojem odpovídá názvu kapitoly a skládá se z řady kroků, které budou vysvětleny v sekci 2.1.

Důležité je uvést zdroje dat pro dolování, od kterých se odvíjí různé metody dolování. Součástí této kapitoly je také vymezení základních typů dolovacích úloh a typů znalostí, které se za jejich pomocí snažíme z dat získat. V závěru budou zmíněny způsoby pro vyhodnocení získaných znalostí.

2.1 Pojem získávání znalostí z dat a dolování dat

Dolování z dat (častěji se používá termín v angličtině *Data Mining*) lze definovat mnoha způsoby – je to disciplína, která se zabývá vývojem metod a algoritmů pro získávání znalostí z dat. *Získávání znalostí z dat* (v angličtině *Knowledge Discovery in Databases*, zkráceně *KDD*) charakterizuje proces extrakce zajímavých informací, neboli modelů dat, a užitečných vzorů. Použití statistických metod v dolování dat není zdaleka triviální a mají vysokou složitost výpočtu. Často může být vážným problémem přizpůsobit statistickou metodu na velký soubor dat nebo analyzování datových toků online. Jedním z příkladů je zpracování dotazů ve vyhledávači, nebo analyzování řeči a videopřenosu v reálném čase.

Ve skutečnosti samotný pojem *dolování z dat* definuje pouze jeden krok procesu získávání znalostí, přesto se často používá jako synonymum k tomuto procesu. Proces získávání znalostí z dat je sekvencí následujících kroků [33]:

1. *Čištění dat* – cílem je odstranění šumu v datech, vypořádání se s chybějícími a nekonzistentními daty.

2. *Integrace dat* - cílem je kombinace více zdrojů dat.

3. *Výběr dat* – cílem je výběr dat relevantních pro danou analytickou úlohu. Pokud zdrojem dat je relační databáze, výběr dat se tyká výběru relevantních sloupců jedné nebo několika tabulek.

4. *Transformace dat* – cílem je transformace a konsolidace dat do tvaru, vhodného pro dolování. Příkladem je konstrukce atributů, které usnadní nebo zkvalitní dolování.

5. *Dolování z dat* – podstata celého procesu, zahrnuje extrakci modelů dat a vzorů představujících znalosti.

6. *Hodnocení modelů a vzorů* – cílem je identifikovat skutečně zajímavé vzory, reprezentující znalosti.

7. *Prezentace znalostí* – zahrnuje použití technik vizualizace pro prezentaci získaných znalostí.

Jak jíž bylo zmíněno, dolování dat je proces objevování zajímavých vzorů a znalostí z velkého množství dat, který zahrnuje výše uvedené kroky. Kroky 1 až 4 spadají do problematiky předzpracování dat, kterému se budu věnovat v kapitole 4.2.

2.2 Druhy dat pro dolování

V této sekci zmíním některé zdroje dat, které používáme k získávání znalostí. V současné době existuje celá řada potenciálních zdrojů užitečných znalostí, které můžeme rozlišovat na „klasická“ a „netradiční“ data. Klasickými daty rozumíme perzistentně uložená data v nějakých uložištích. Nejčastějším zdrojem takových dat je *relační databáze*. Data v relační databázi představují kolekci tabulek, z nichž každé je přiřazen jedinečný název. Každá taková tabulka se skládá z řády atributů (neboli sloupců) a obvykle obsahuje velké množství záznamů (řádků). Modelování dat se provádí pomoci ER[1] diagramu. Typickým příkladem relační databáze je databáze klientů banky, která by mohla obsahovat tabulku *Klient* se sloupci *ID, Jméno, Adresa, Pobočka,* apod.

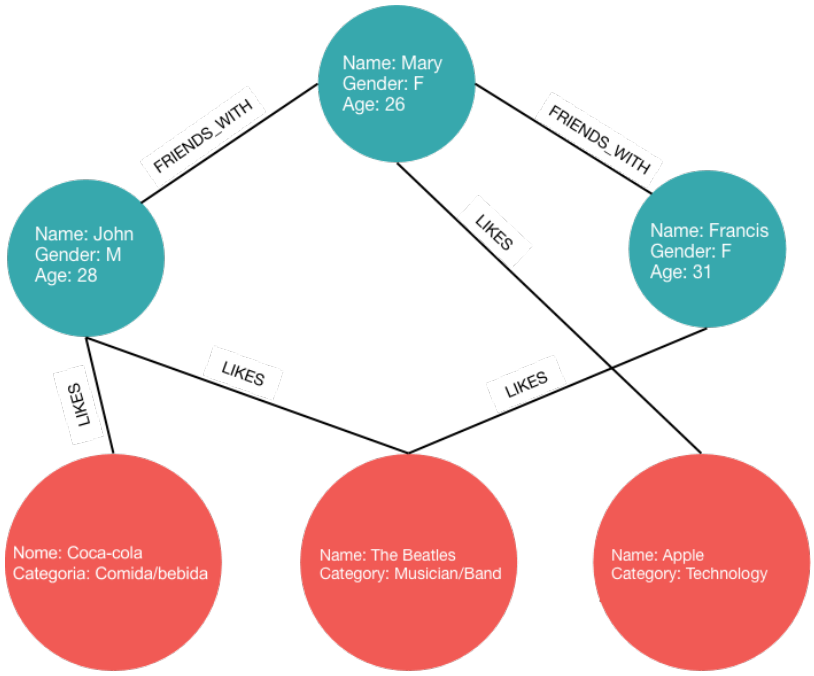
Dalším typem zdrojů dat, se kterým se lze setkat při řešení dolovacích úloh, je *datový sklad*. Datový sklad je uložištěm informací shromážděných z více zdrojů a obvykle umístěných na jednom místě. Datové sklady se vytvářejí prostřednictvím procesu čištění dat, integrace dat, transformace dat, načítání dat a periodického obnovování dat. Data jsou zde uložená tak, aby poskytovala informace z historického hlediska, například za období posledních 12 měsíců. Datový sklad je obvykle modelován ve tvaru vícerozměrné datové struktury, která se nazývá *datová kostka*, jejíž dimenze odpovídá jednomu, nebo více atributům. Datová krychle poskytuje vícerozměrné zobrazení dat a umožňuje rychle získat agregované výsledky. Příkladem může být firma, která má více poboček s vlastní databází obsahující detailni informace o prodejích zboží, a potřebuje shrnout výsledky prodejů za každé čtvrtletí posledního roku. V takovém případě datová kostka může mít tři dimenze – adresa pobočky, čtvrtletí s hodnotami Q1, Q2, Q3 a Q4 a druh zboží, vůči které se vypočítá objem prodeje. Objem prodeje v tomto případě bude agregovanou hodnotou.

V dnešní době intenzivně vzrůstá podíl „netradičních dat“ na celkovém objemu dat, která máme k dispozici. Když mluvíme o dolování v netradičních datech, rozumíme tím získávání znalostí z proudů dat, z časových řád, sekvencí, databáze grafů, zvuku, videa apod. Taková data vznikají dynamicky v nějakém zdroji a často potřebují zpracování během sběru dat, online. Sem patří například datové toky počítačových sítí, které lze využit pro detekci útoků na sít' na základě anomálií v toku zpráv. Anomálie mohou být objeveny provedením shlukové analýzy, dynamické konstrukce proudových modelů (angl. *Data stream model* [24]) nebo porovnáním frekventovaných vzorů, charakterizujících typické chování na síti. Dalšími příklady jsou proudy nákupů, proud bankovních transakcí, proudy dat z výstupu různých senzorů, jako je senzor pohybu, teploty atd. Protože velikost proudu je potenciálně neomezená, klasické relační databáze nejsou pro ukládání takových dat vhodná ani použitelná. Taková data není možné a taktéž není nutné z dlouhodobého hlediska ukládat.

[1] ER diagram, v angličtině entity-relationship diagram – diagram entit a vztahů

Data jsou uložena v dočasném uložišti a poté zpracována dolovacím algoritmem, přizpůsobeným k tomu, že data jsou dostupná jenom po nějakou omezenou dobu. Jinými slovy, doba běhu dolovacího algoritmu musí být předvídatelná a krátká. Po provedení analýzy a uplynutí určité doby se pamět'ová oblast, ve které se data uchovávaly, uvolní a celý proces se opakuje.

Zdroje dat, se kterými se lze také velmi často setkat při dolování, jsou nestrukturovaná nebo částečně strukturovaná data (např. texty, HTML stránky, XML). Patří sem také data, která reprezentují naopak bohatší strukturu – typickým příkladem jsou data ze sociálních sítí. Například skupina přátel určité osoby může být reprezentována jako *grafová databáze* (angl. *Graph Database*) s několika uzly propojené hranami, které tvoří vztah mezi danými objekty (přáteli) – viz Obrázek 2.1.



Obrázek 2.1: Příklad grafové databáze, reprezentující uživatele sociální sítě a jejich vzájemné vztahy. (Převzato z [5]).

2.3 Typy dolovacích úloh

Cílem této části je seznamit se s nekterými základními typy dolovacích úloh. Na základě dostupných dat a jejich druhu rozhodujeme, jakou dolovací úlohu zvolit pro získání požadovaného modelu dat. Jak jíž bylo zmíněno dříve, jde o jádro procesu získávání znalostí. Obecně typy dolovacích úloh lze rozdělit do dvou kategorií: *deskriptivní* a *prediktivní* [37]. Deskriptivní úlohy charakterizují vlastnosti dat. Prediktivní úlohy provádějí analýzu aktuálních dat s cílem předpovědi budoucího chování.

V terminologii strojového učení [33] algoritmy pro získavání znalostí podle způsobu učení můžeme rozdělit do následujících skupin:

- *Učení s učitelem* (angl. *Supervised Learning*) – metoda učení z tzv. trenovácí množiny, obsahující již známe třídy dat pro predikci třídy objektů, jejichž zařazení neznáme. Daná metoda je synonymem pro *klasifikaci a regresi*.
- *Učení bez učitele* (angl. *Unsupervised Learning*) – cílem je analýza datových objektů, jejíchž prířazení do tříd není známé. Cílem zde je nalezení shluků dat, které mají mezi sebou společné vlastnosti. Daná metoda je synonymem pro *shlukovou analýzu*.
- *Kombinace učení s učitelem a bez učitele* (angl. *Semi-supervised Learning*) - je typickou metodou v případě, když máme malé množství dat, jejichž zařazení známe, s velkým množstvím dat, jejichž přiřazení do tříd neznáme.

Podívejme se nyní na některé „klasické“ druhy dolovaných znalostí, mezi které patří asociační pravidla a frekventované vzory, klasifikace a regrese a také shluková analýza.

Frekventované vzory a asociační analýza

Do prvního klasického druhu dolovacích znalostí lze zařadit získávání informací o vztazích mezi hodnotami atributů, o asociacích a korelacích v datech. Sem patří úloha hledání frekventovaných vzorů, při které se detekují hodnoty s velkou společnou četností výskytu. Mohou to být uzly grafu v grafové databázi, které se vyskytují často společně, úseky sekvencí v sekvenční databází apod. Příkladem takové úlohy je analýza nákupního košíku s cílem zjistit, které položky zboží z nabízeného sortimentu se prodávají často společně. Na základě historie nákupů lze následně odvodit asociační pravidlo, které říká, že pokud si zákazník pořídí zboží X, s určitou spolehlivostí si také pořídí zboží Y. Taková analýza umožňuje managementu obchodu organizovat uspořádání zboží jiným způsobem s cílem podpořit „společné“ nákupy.

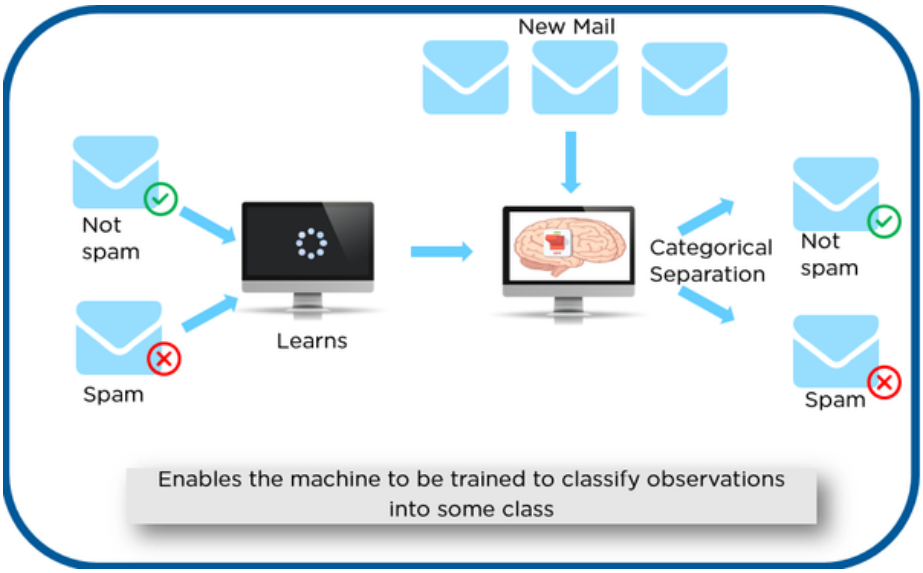
Frekvenční analýzu je možné také použít na luštění různých monoalfabetických nebo polyalfabetických substitučních šifer – druh šifry, při které dochází k záměně (substituci) nějaké množiny symbolů za jinou množinu symbolů. Pro dešifrování monoalfabetické šifry je potřeba určit četnost výskytu jednotlivých znaků v textu, když v případě polyalfabetické šifry se pracuje se skupinami znaků, například s dvojicemi písmen, bigramy.

Klasifikace a regrese

Druhým typem znalostí je klasifikace a regrese. Cílem klasifikační úlohy je určení hodnoty atributu na základě jeho vlastností. Hodnoty, kterých může nabývat atribut se označují jako třídy. Na základě historických znalostí algoritmus vytvoří takový model, který popisuje a rozlišuje třídy dat [37]. Model se použije pro predikci třídy objektu, jehož zařazení předem neznáme. Případně výsledkem může být míra pravdepodobnosti, s jakou objekt může být zařazen do konkretní třídy.

Speciálním případem je binární klasifikace. Je základním a nejjednodušším případem rozhodování. Binární klasifikace klasifikuje do dvou tříd. Jinými slovy se používá k predikci třídy binárního atributu, který může nabývat pouze dvou možných hodnot. Příkladem daného druhu klasifikace je klasifikace zákazníků banky z hlediska schopnosti splácet úvěr. Výsledkem analýzy je přiřazení zákazníků do tříd „ano“/„ne“, případně označení binární hodnotou 1 nebo 0. Kromě toho častým příkladem v reálním životě je klasifikace emailů, bud’ do množiny tříd, které reprezentují různé složky vytvořené uživatelem (například na základě odesílatele), nebo pro detekci spamů - viz Obrázek 2.2.

Regrese se liší od klasifikace pouze typem atributů, které předpovídáme. V tomto případě jde o hodnoty spojitých, neboli numerických atributů. Výsledek, který nás zajímá, může nabývat relativně velkého až nekonečného počtu různých hodnot. Příkladem může být predikce cen na burze na základě známého vývoje akcí v průběhu několika posledních dní. Nejčastější metodou regrese je regresní analýza.



Obrázek 2.2: Příklad klasifikace emailů. (Převzato z [27]).

Klasifikaci a regrese a modelům, které umožňují danou analýzu provádět, se budu podrobně věnovat v kapitole 4.3.2.

Shluková analýza

Na rozdíl od klasifikace a regrese, které provádí analýzu souboru dat s klasifikovanými do tříd atributy, shlukování se realizuje bez ohledu na třídy objektů, pokud takové existují [37]. Cílem shlukování je seskupit objekty do tříd, které mají co nejvíc společných rysů, a naopak minimalizovat podobnost objektů různých tříd.

V určitých případech ale zajímavé vzory mohou naopak reprezentovat hodnoty, které se významně liší od ostatních. Takové datové objekty nazýváme *odlehlé hodnoty* (v angličtině *outliers*) a jejich analýzu *analýzou odlehlých hodnot*.

Kapitola 3

Podpora jazyka Python pro dolování z dat

Jazyk Python je vysokoúrovňový skriptovací programovací jazyk, který nabízí instalační balíky pro většinu běžných platforem (Unix, MS Windows, macOS, Android). Tento jednoduchý na učení programovací jazyk umožňuje efektivně pracovat s vysokoúrovňovými datovými strukturami a podporuje různá programovací paradigmata, včetně objektově orientovaného programování[26].

Python je úspěšně používán v tisících reálných aplikací po celém světě, včetně mnoha důležitých systémů. Příkladem úspěšného softwaru založeného na jazyce Python je volně dostupný cloudový operační systém Openstack[15], který slouží pro vytváření a spravování soukromých a veřejných cloudů. Avšak hlavní úlohou této bakalářské práce je prozkoumat moduly jazyka Python pro oblast analýzy dat, které umožní odhalit užitečné netriviální informace a podpořit rozhodování.

Cílem této kapitoly je vymezení dostupných prostředků jazyka Python pro podporu dolování z dat v různých fázích celého procesu, který jíž byl popsán v kapitole 2.1.

3.1 Jupyter Notebook

Jupyter Notebook (dříve označovaný jako notebook IPython) je *open-source* webová aplikace, která poskytuje komfortní prostředí pro psaní kódu *online* (angl. *live code*) [25]. Jupyter umožňuje snadno vytvářet a sdílet dokumenty obsahující rovnice, provádět vizualizaci dat a vkládat volný text. Aplikaci lze použit také pro snadné čistění a transformaci dat, numerickou simulaci, statistické modelování, strojové učení a mnoho dalšího. Navíc Jupyter Notebook podporuje více jak 40 programovacích jazyků, včetně Pythonu, R, Julie a Scaly.

Po instalaci lze Jupyter Notebook spustit z příkazové řádky příkazem **jupyter notebook**, který spustí webový server na místním počítači. Následně pracujeme s aplikací v prohlížeči. Tento interaktivní násroj má podobu příkazových řádků.

3.2 Explorační analýza dat a předzpracování dat

V této sekci budou uvedeny knihovny, které se široce používají pro analýzu a manipulace s daty, zejména ve fázi předzpracování. Jsou to knihovny `Pandas`, `NumPy`, `Matplotlib` a `Plotly`. Dané knihovny poskytují prostředky pro rychlé a snadné načtení dat, vizualizaci dat a různé manipulace s cílem pochopit data a připravit je pro dolování.

Pandas

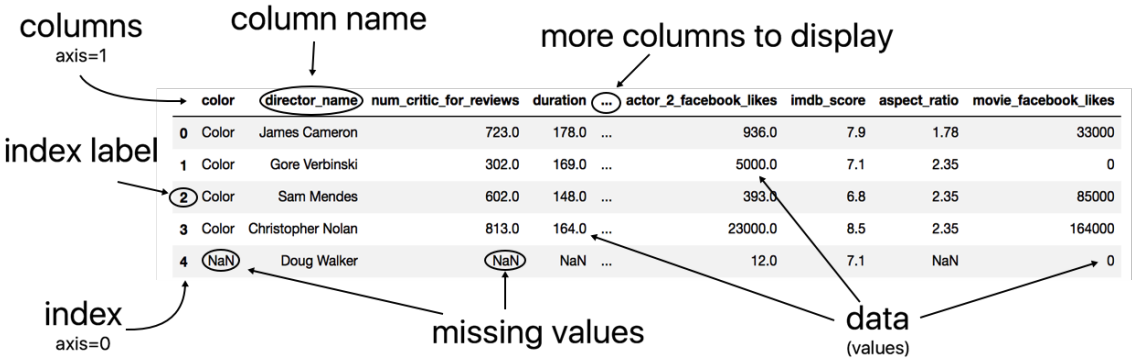
`Pandas` je významnou *open source* knihovnou Pythonu, obsahující metody pro rychlou a flexibilní práci se strukturovanými daty (dvourozměrnými tabulkovými strukturami, vícerozměrnými datovými strukturami) a časovými řadami dat.

Knihovna poskytuje široké možnosti pro snadnou manipulaci s daty, mezi které patří [18]:

- Ukládání/načítání dat do/ze souborů ve formátu CSV a textových souborů, SQL databází, Microsoft Excel tabulek a HDF5 formátu;
- Integrovaná manipulace s chybějícími daty, reprezentovanými jako *NaN* hodnoty;
- Agregace a transformace dat pomocí nástrojů, umožňujících provádět tzv. operace „split-apply-combine“[1];
- Slučování (angl. *merge*) a spojování (angl. *join*) dat;
- Hierarchické indexování pro efektivní práci s vícerozměrnými daty;

Hlavními datovými typy knihovny `Pandas` jsou `Series` (1-rozměrná datová struktura) a `DataFrame` (2-rozměrná datová struktura). Datový typ `DataFrame` se používá častěji, než datový typ `Series`, a lze ho přirovnat k datovému rámci, který vytváří metoda `data.frame`[2] jazyka R. Obrázek 3.1 ilustruje strukturu datového rámce v jazyce Python, který je prezentován formou klasické tabulky a skládá se ze tří různých komponent – indexů, sloupců a samotných dat.

Objekt typu `Series` je potom reprezentován jako jeden sloupec takové tabulky – pole obsahující posloupnost hodnot.



Obrázek 3.1: Struktura objektu typu `DataFrame`. (Převzato z [22]).

Nejčastějším způsobem naplnění objektu `DataFrame` je načtení dat ze souboru. Následně lze k datum přistupovat různými způsoby, například nahlédnout na prvních pár řádku pro získání první představy o datech. Knihovna poskytuje také dobrou podporu pro snadný přístup k datům (angl. *selection*) datového rámce pomocí metod `DataFrame.loc`[3],

[1] https://pandas.pydata.org/pandas-docs/stable/user_guide/groupby.html
[2] https://www.rdocumentation.org/packages/base/versions/3.6.0/topics/data.frame
[3] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.loc.html

`DataFrame.iloc`[4], `DataFrame.at`[5], `DataFrame.iat`[6]. Metody umožňují vybírat data na základě určitých podmínek, na základě názvů sloupců a řádků, nebo také pomocí celočíselného indexování.

Následující příklady kódu v jazyce Python slouží pro ukázku základních funkcí knihovny [1]. Prvním krokem je importování knihovny. Potom načtení dat ze souboru `data.csv` ve formátu CSV s vytvořením objektu `DataFrame` lze provést následovně:

```
In [1]: import pandas as pd
In [2]: df = pd.read_csv("data.csv")
```

Uvažujme, že načtená data reprezentují tabulku 3.1, obsahující informace o některých druzích ovocí, jejich barvě a naskladněném počtu. Jaké druhy ovoce jsou k dispozici? Výběr jednoho sloupce lze provést následujícím způsobem:

```
In [3]: df['Fruit']

Out[3]: 0      Banana
        1    Cherries
        2        Kiwi
        3       Lemon
```

Jsou k dispozici nějaké druhy ovoce, jejichž kusů je méně než 100? V tomto případě příkladem může být výběr řádků, splňujících požadovanou podmínku a sloupců, které chceme zobrazit ve výsledku:

```
In [4]: df.loc[df['Amount'] < 100, ('Fruit', 'Amount')]

Out[4]:     Fruit   Amount
        0   Banana  60
        2   Kiwi    50
```

Fruit	Color	Amount
Banana	Yellow	60
Cherries	Red	100
Kiwi	Green	50
Lemon	Yellow	120

Tabulka 3.1: Příklad tabulky načtené ze souboru do objektu DataFrame.

Využití dalších užitečných metod, které poskytuje daná knihovna, bude vysvětleno v kapitole 5, obsahující řešení případové studie.

NumPy

NumPy je základním balíčkem pro vědecké výpočty v Pythonu, poskytujícím vysoce výkonné objekty N-rozměrného pole a nástroje pro práci s nimi. Mimo jiné knihovna obsahuje nástroje pro integraci kódu v jazyce C/C++ a Fortran. Kromě zřejmého využití

[4] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.iloc.html
[5] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.at.html
[6] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.iat.html#pandas.DataFrame.iat

knihovny v oblasti vědy, *NumPy* může být také použita jako efektivní vícerozměrný kontejner generických dat. Zde mohou být definovány libovolné typy dat, což umožňuje *NumPy* bezproblémovou a rychlou integraci s řadou databází [13].

Knihovnu lze importovat následujícím způsobem:

```
In [1]: import numpy as np
```

Následně budou ukázány některé způsoby vytvoření objektu *ndarray*, reprezentující homogenní pole dat. Existuje několik obecných mechanizmů pro vytváření polí, některé z nich jsou [14]:

- Konverze z jiných datových struktur Pythonu (například seznamů, slovníků) za použití funkce *array()*.
- Použití speciálních funkcí knihovny (např. *random*).
- Vytváření polí načtením po jednotlivých bajtech z řetězců nebo místa v paměti.
- Vlastní vytvoření pole *NumPy Array* za použití vestavěných funkcí knihovny. Například funkce *zeros(shape)* vytvoří pole ve specifikovaném tvaru vyplněné nulami.

Ukázka vytvoření pole v Pythonu:

```
# create one-dimensional array with float numbers
In [2]: a = np.array([1.2, 3.5, 5.1])
In [3]: a.dtype # get type of elements
Out[3]: dtype('float64')

# create two-dimensional array
In [4]: b = np.array([(1.5,2,3), (4,5,6)])
In [5]: b
Out[5]: array([[1.5, 2. , 3. ],
               [4. , 5. , 6. ]])
```

S polem lze různě a snadně manipulovat, například provádět transformaci z 2-rozměrného pole do 3-rozměrného pole apod. (v angličtině říkáme *Shape Manipulation*) [19]. Počet dimenzí lze zjistit přístupem k objektu **ndarray.ndim** a velikost pole v každé dimenzi přístupem k objektu **ndarray.shape**. Příklad vytvoření jednorozměrného pole obsahujícího sekvenci čísel a jeho následné transformace do vícerozměrného pole:

```
In [6]: c = np.arange(15).reshape(3, 5)
Out[6]: array([[ 0,  1,  2,  3,  4],
               [ 5,  6,  7,  8,  9],
               [10, 11, 12, 13, 14]])
```

Knihovna *NumPy* tvoří základ pro knihovnu *SciPy* [28]. Jinými slovy knihovna *SciPy* je nadstavbou knihovny *NumPy*. *SciPy* je také významnou knihovnou pro matematiku, vědu a inženýrství a skládá se z mnoha modulů. Jsou to moduly určené na interpolaci, Fourierovu transformace, zpracování signálů, lineární algebru atd.

Matplotlib

Pravděpodobně nejzákladnější způsob konstruování grafu je vytváření grafu pomocí nástrojů knihovny *Matplotlib*. Daná knihovna poskytuje možnosti vytvářet 2D anebo také 3D

grafy za použití *mplot3d* nástroje. Součástí knihovny je modul *mathtext*[7], který podporuje vytváření matematických výrazů takovým způsobem, jak je to řešeno v LaTeX. Při vykreslování grafu se často používá *matplotlib.pyplot*[8] API. Jedná se o sbírku funkcí ve formě příkazů, díky nimž *Matplotlib* funguje jako MATLAB. Každá funkce umožňuje provést nějakou změnu obrázku, například vytvoří obrázek, vytvoří plochu pro vykreslování obrázku, přidává názvy os do grafu apod.

Matplotlib lze použit v Python skriptech, Python a IPython shellech, Jupyter Notebooku anebo také v jednom z uživatelských rozhraní [10]. Knihovna poskytuje kvalitní grafický výstup, včetně formátů PNG, PDF, SVG, EPS.

Plotly

Grafická knihovna `Plotly` se používá pro vizualizaci dat, umožňuje vytvářet interaktivní grafy [17]. Jsou to například krabicové grafy, histogramy, bodové grafy, řádkové grafy (angl. *line plots*), bublinové grafy, teplotní mapy (angl. *heatmaps*) a mnoho dalších. Uživatelé, kteří mají vytvořený učet na webových stránkách Plotly, mohou vykreslovat, interagovat a sdílet grafy online v prohlížeči, nebo také přes aplikaci nainstalovanou na lokální počítač.

3.3 Dolování z dat

V předchozí sekci byl uveden popis knihoven, které obsahují různé metody podstatné pro fázi předzpracování. V této sekci budou popsány nástroje velmi důležité pro dolování z dat dostupné v knihovně `scikit-learn`.

Scikit-learn

Scikit-learn je *open-source* balíčkem založeným na třech dříve zmíněných knihovnách – *NumPy*, *SciPy* a *Matplotlib*. Je určený pro podporu strojového učení v jazyce Python [21]. *Scikit-learn* nabízí celou řádu nástrojů určených pro datovou analýzu, včetně nástrojů pro předzpracování dat. Modul *sklearn.preprocessing* obsahuje metody pro škálování, centrování, normalizaci, tzv. binarizaci (angl. *binarization*) a imputaci. Pokud se jedná o klasifikaci, příkladem může být transformace kategorických hodnot tříd na hodnoty numerické:

```
In [1]: label_binarize(['yes', 'no', 'no', 'yes'], classes=['no', 'yes'])
Out[1]: array([[1],
               [0],
               [0],
               [1]])
```

Dále *scikit-learn* nabízí snadné a efektivní nástroje pro různé úlohy dolování z dat – klasifikaci a regresi, shlukování, detekci odlehlých hodnot na základě shlukování a některé metody pro kombinaci učení s učitelem a bez učitele. Velmi často nezkušený datový analytik neumí zvolit algoritmus pro řešení určité úlohy. V tomto případě na stránkách *příručky uživatele scikit-learn* lze najít příklady použití a porovnání algoritmů pro specifickou úlohu [21].

Mimo jiné baliček disponuje nastrojí pro vyhodnocování výsledných modelů a jejich porovnání. V případě klasifikace a regrese se jedná o porovnání přesnosti předpovědí. Pro

[7] https://matplotlib.org/gallery/text_labels_and_annotations/mathtext_examples.html
[8] https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot

dané účely baliček *metrics*[9] poskytuje řadu metrik pro vyhodnocení – *accuracy_score*, *precision_recall_curve*, *precision_score* a *recall_score* a další. Jako příklad lze uvést získání přesnosti klasifikace. Předpokládejme, že skutečné hodnoty třídy jsou uložené v proměnné y_true a předpovězené, které by jim měly odpovídat, v proměnné y_pred:

```
In [2]: import numpy as np
        from sklearn.metrics import accuracy_score
        y_true = [0, 0, 0, 0, 1, 0, 1, 1]
        y_pred = [0, 1, 1, 0, 0, 0, 0, 1]
        accuracy_score(y_true, y_pred)
Out[2]: 0.46
```

Více příkladů použití různých nástrojů balíčku *scikit-learn* budou uvedeny v kapitole 5 popisující řešení případové studie. Popis metrik pro vyhodnocení modelů bude uveden v kapitole 4.4.

[9] https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics

Kapitola 4

Případová studie

V této kapitole se zaměřím na teoretický základ, jehož znalost je pro řešení případové studie nezbytná. Začátek kapitoly je věnovaný popisu zvolené dolovací úlohy. V podkapitole 4.2 budou vymezeny základní techniky používané ve fázi předzpracování dat s cílem připravit data pro aplikaci dolovacího algoritmu. Následně v podkapitole 4.3 budou podrobněji rozvinuty dva typy dolovacích úloh, o kterých již bylo zmíněno v kapitole 2.3. Jedná se o detekci odlehlých hodnot a klasifikaci. Také budou uvedeny některé modely pro získávání znalostí, které jsou schopné tyto úlohy provádět. Jejich pochopení je klíčové pro řešení zvoleného problému. Poslední část 4.4 této kapitoly zahrnuje způsoby a metriky pro vyhodnocení úspěšnosti použitých modelů.

4.1 Úvod do případové studie

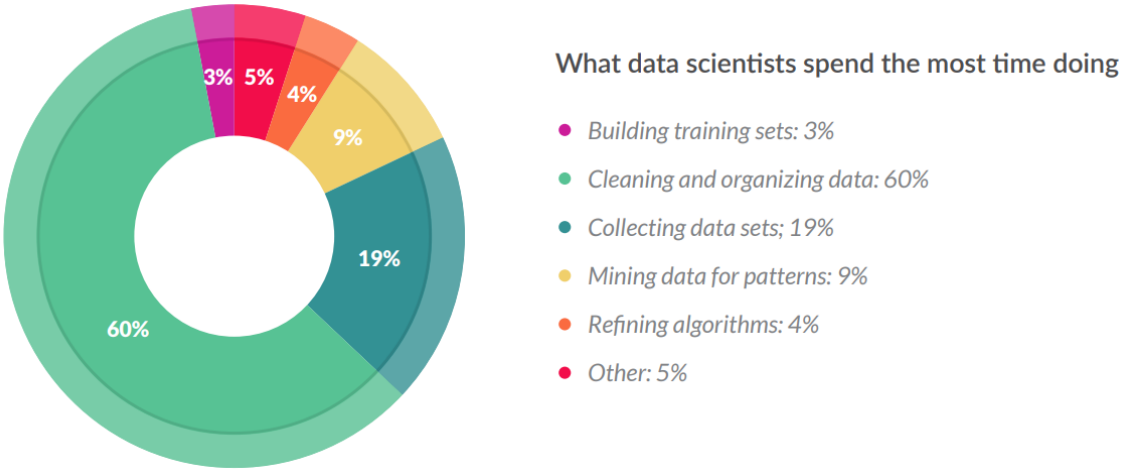
Kniha „*Data Mining with R*“ nabízí 4 případové studie [36], prezentujících řešení různých dolovacích úloh v jazyce R. Cílem této práce bylo zvolit jednou z nabízených případových studií, seznámit se s přístupy a algoritmy, které byly použity při řešení dané úlohy, a následně vytvořit řešení úlohy v jazyce Python. Řešení je prezentováno v kapitole 5 a je zaměřené na demonstraci dostupných prostředků v jazyce Python pro získávání znalostí z dat.

Po dohodě s vedoucím byla zvolená případová studie na téma „Detekce podvodných transakcí“. Transakcemi v daném případě rozumíme data, které pochází z údajů v databázi o jednotlivých prodejích určité sady produktů, prováděných zaměstnanci firmy X. Jak již plyne z názvu, podstata této úlohy spočívá v detekci takových dat, které se vymykají obecnému chování. Cílem je najít takové transakce, které mohou naznačovat pokusy o podvod některými prodejci dané firmy. Vzhledem k omezenému počtu informací o jednotlivých prodejích, nelze se 100% jistotou určit, jestli určitá transakce může být považována za podezřelou nebo ne. Proto výsledek analýzy bude mít formu hodnocení pravděpodobnosti typu transakce (podvodná/v pořádku). Následně by byl výsledek předán na kontrolu firmě.

Tento typ analytické úlohy, který se tyká vyšetření zdrojů dat s omezeným množstvím využitelné informace, se může vyskytovat v mnoha oblastech pří řešení reálných problémů. Například při kontrole daňových přiznání nebo vyšetření transakcí prováděných kreditními kartami. Při řešení daného typu analytické úlohy se lze setkat s takovými dolovacími úlohy jako *detekce odlehlých pozorování na základě shlukování*, *klasifikace*, anebo také s *metodami kombinovaného učení s učitelem a bez*. V této práci se budu zabývat prvními dvěma typy dolovacích úloh.

4.2 Předzpacování dat

Jak jíž bylo zmíněno dříve, v dnešní době lze pozorovat obrovský nárůst objemu dostupných dat, často pocházejících z několika různých datových zdrojů. Spolu s tím stoupá četnost výskytu chyb a nekonzistencí v datech. Taková data není možné bezprostředně podstoupit dolovacímu algoritmu k dolování. Jelikož se jedná o data nekvalitní, nelze očekávat, že výsledkem dolování budou kvalitní znalosti. Z celého procesu získávání znalostí fáze předzpracování představuje na úspěšném výsledku nezanedbatelný podíl. Různé zdroje uvádějí, že datoví analytici tráví 60-80% svého času připravováním dat pro aplikaci dolovacího algoritmu a jenom kolem 10-15% času na samotném dolování z dat. Zajímavé statistiky prezentuje skupina vědců CrowdFlower [31], které ukazují procento času tráveného datovými analytiky v jednotlivých fázích procesu získávání znalostí – viz Obrázek 4.1.



Obrázek 4.1: Čas trávený datovými analytiky v jednotlivých fázích procesu získávání znalostí. (Převzato z [31])

Existují tří základní problémy, resp. projevy nekvality dat [37], se kterými se zpravidla můžeme setkat při předzpracování:

- Data jsou neúplná
- Data jsou zašuměná
- Data jsou nekonzistentní

Neúplností dat rozumíme přítomnost chybějících hodnot v hodnotách atributů. Jednou z možných příčin vzniku takových hodnot může být jejích nepovinné zadání při sběru dat. Jiným důvodem může být porucha zařízení při sběru dat. Taktéž mohou nastat situace, když nejsou dostupné všechny potřebné atributy, které bychom pro dolování potřebovali. Typicky k tomu dochází, když v době návrhu databáze nebyl dostatečný důvod na vytvoření daného atributu.

Šumem v datech rozumíme výskyt náhodných chyb nebo odchylek hodnot atributů. Zdrojem odlehlých hodnot může být lidská chyba, špatné pochopení významu požadovaného údaje nebo porucha zařízení při sběru dat. Bohužel nejsme vždy schopni rozpoznat, kdy se jedná o chybu v datech. Jsou to situace, když chybna hodnota patří do tzv. běžného

rozsahu hodnot. Příkladem může být výskyt chyby při monitorování tělesné teploty pacientů. Předpokladejme, že rozsah hodnot se pohybuje od 35 do 42 stupňů. V případě, když nesprávně zadaná teplota pacienta bude patřit k danému rozsahu, bez příme interakce s pacientem chybu není možné identifikovat.

O *nekonzistenci dat* zpravidla mluvíme v případě, když dochází k integraci heterogenních informačních zdrojů. Použití dat z několika datových zdrojů může vést k vzniku redundance a způsobit nekonzistenci dat. Typickými dílčími problémy nekonzistence jsou rozpory v identifikátorech, kódování nebo formátech několika datových objektů, které vyjadřují stejný význam. Potom jedním z cílů úlohy předzpracování je zjistit, jestli dané atributy vyjadřují totéž.

Hlavním úkolem předzpracování je tedy odstranit výše zmíněné nedostatky a zajistit co nejkvalitnější data pro dolování. Mezi základní úlohy předzpracování patří *čištění dat*, *integrace dat*, *transformace* a *redukce*. Čištění dat se provádí s cílem odstranit chybějící data, šum v datech a nekonzistenci. Integraci dat provádíme s cílem integrovat data pocházející z různých zdrojů do jedné databáze. V případě více datových zdrojů úlohy čištění a integrace velmi úzce souvisí spolu. Po provedení integrace dat mohou vzniknout nové nekonzistence, proto celý proces „čistění integrace“ se může opakovat několikrát. Další typickou úlohou je *transformace* dat do tvaru vhodného pro řešení dané dolovací úlohy. Příkladem může být normalizace numerických hodnot atributu do intervalu <0,1>.

Později v této kapitole bude podrobněji vysvětlena úloha čistění dat, jelikož je nejvíc relevantní pro řešení dané práce. Dříve ale než k tomu přistoupím, zaměřím se na souhrnné charakteristiky popisující data.

4.2.1 Sumarizující popisné charakteristiky dat

Než přejdeme k samotnému kroku předzpracování dat, nejprve je potřeba získat celkovou představu o datech, která máme k dispozici. Pro dosažení maximální úspěšnosti v řešení úlohy je potřeba datům porozumět. Základní úlohou je porozumět sémantice dat a získat co nejvíc informací o daných atributech – jejich počet, jednotlivý význam, informace o hodnotách daného atributu apod. Techniky sumarizujícího popisu dat slouží právě pro identifikaci problémů popsaných výše – identifikaci nekompletních dat, nekonzistencí, odlehlých hodnot a dalších projevů nekvality dat.

Na začátku je důležité rozlišit základní typy atributů. Z hlediska vlastnosti oboru hodnot atributů rozlišujeme atributy:

- *kvantitativní* (také *numerické*),
- *binární*,
- *kategorické*,
- *ordinální*.

Kvantitativní atribut, neboli *numerický*, je takový atribut, jehož hodnoty reprezentují celá nebo reálná čísla udávající množství, počet. Obor numerických hodnot je uspořádaný a je potenciálně nekonečný nebo velký. Například počet virtuálních procesorů, kapacita paměti, teplota.

Binární atribut je atribut, reprezentující pouze dva stavy – 0 nebo 1, kde 0 typicky znamená, že atribut chybí a 1 znamená, že je přítomen. Hodnoty binárního atributu jsou označovány datovým typem `Boolean`, pokud nabývají dvou stavů – *true* a *false*.

Kategorický atribut je vyjádřený diskrétními hodnotami. Obor hodnot takových atributů je konečný (typicky poměrně malý) a není uspořádaný, příčemž hodnoty mohou, ale nemusí být reprezentována celými čísly. Například atribut *Barva* může nabývat několika hodnot – {„červená“, „žlutá“, „zelená“}. *Ordinální atribut* je obdobou kategorického atributu, jehož obor hodnot je uspořádaný. Například vzdělání – „základní“, „střední“, „vyšší“.

Dále budou vysvětleny základní statistické metody charakterizující data a jejich role v předzpracování dat.

Základní statistické míry pro charakterizaci dat

Základní statistické míry lze použít k identifikaci vlastností dat a rozpoznání hodnot, které se mohou považovat za šum v datech nebo odlehlé hodnoty. Takové míry můžeme rozdělit do tří skupin: míry polohy, míry variability a míry šikmosti a špičatosti. V této sekci se zaměřím pouze na dvě z nich – míry polohy a variability.

Míry polohy (známe také jako míry střední hodnoty, míry centrální tendence) charakterizují polohu „středu“ v souboru dat. Mezi známé míry polohy patří *aritmetický průměr*, *medián*, *modus*, nebo také *kvantily* [37].

Aritmetický průměr je nejznámějším odhadem střední hodnoty. Pro náhodný výběr (X) obsahující n hodnot se počítá jako součet všech hodnot vydělených jejich počtem n:

$$\bar{X} = \frac{1}{n}(x_1 + x_2 + \ldots + x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.1}$$

Medián se určuje prostřední hodnotou seřazeného datového souboru. V případě sudého počtu hodnot je medián aritmetickým průměrem dvou prostředních prvků. Například máme k dispozici soubor numerických hodnot {1, 5, 8, 12, 16, 21}. V tomto případě hodnota mediánu je 10.

Modus (angl. *mode*) je hodnota reprezentována s největší frekvencí výskytu v souboru. Pokud se jedná o jednou takovou hodnotu, hovoříme o *jednomodálním* souboru. V jinem případě, pokud takových hodnot je víc – soubor je *multimodální*. Modus je významný zejména při charakterizování polohy v souboru kategorických a ordinálních hodnot. Například pro soubor hodnot {1, 4, 4, 8, 3, 4, 16} modus je číslo 4.

Jako poslední míry polohy vysvětlím *kvantily*, které reprezentují hodnoty náhodné veličiny nacházející se na hranicích pravidelných intervalů. Kvantily mohou být různě označeny [8], například:

- *Percentil*, neboli 100-kvantil, dělí statistický soubor na setiny.
- *Decil*, neboli 10-kvantil, dělí statistický soubor na desetiny.
- *Kvartil*, neboli 4-kvantil, dělí statistický soubor na čtvrtiny. hraniční hodnota intervalu, ve kterém se nachází 25% hodnot se označuje jako kvartil Q1. Odpovídajícím způsobem se označuje kvartíl Q3 pro interval, ve kterém se nachází 75% hodnot.
- *Medián* je také kvantilem, který dělí statistický soubor na dvě stejně početné množiny.

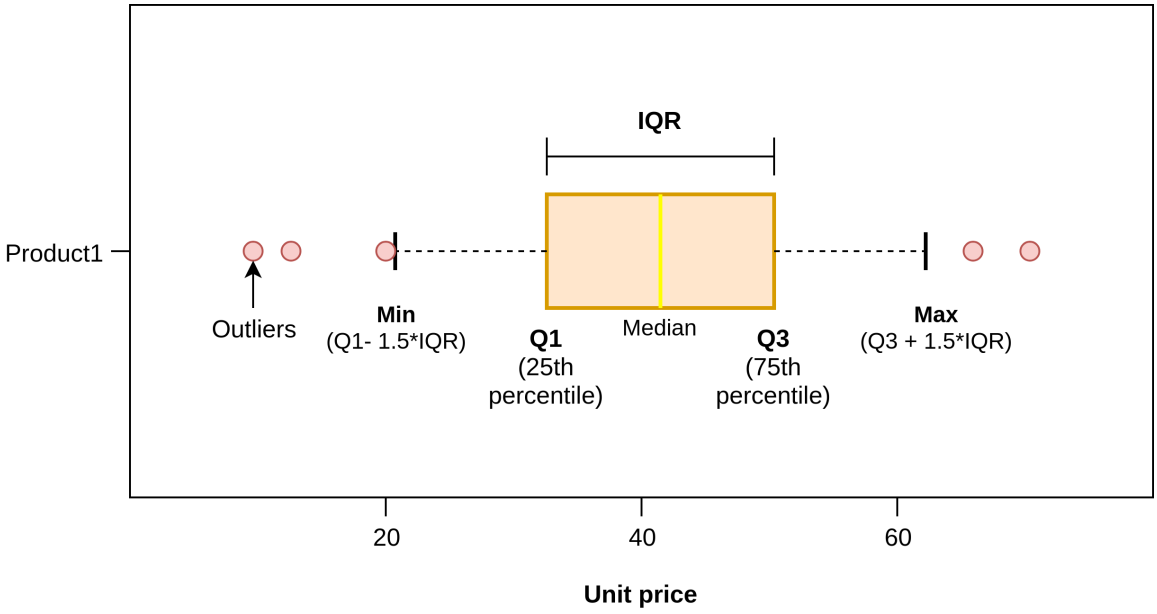
Kvantily poskytují užitečnou informaci o rozptylu hodnot, proto se ještě objeví při popisu měr variace.

Míry variace poskytují informace o rozptylu hodnot datového souboru. Takové míry zahrnují *rozsah hodnot*, *kvantily*, *kvartily*, *mezikvartilovou vzdálenost*, *rozptyl* a *směrodatnou*

odchylka. *Rozsah hodnot* je nejjednodušší mírou variace a definuje rozdíl maximální a minimální hodnoty souboru. *Mezikvartilová vzdálenost*, neboli IQR, je rozdílem mezi třetím Q3 a prvním Q1 kvartilem a vypovídá o intervalu, ve kterém se nachází 50% prostředních hodnot.

Po vysvětlení základních pojmů lze uvést často používané pri charakterizaci dat tzv. složené míry. Jednou z nich je *sumarizace pěti čísly* (min, Q1, medián, Q3, max). Zpravidla pětice hodnot je reprezentována graficky v podobě krabicového grafu (angl. *boxplot*) [32] – viz Obrázek 4.2.

Krabicový graf je jedním z nejpopulárnějších způsobů grafické reprezentace souhrnných charakteristik. Graf reprezentuje vztah mezi hodnotou kategorického atributu a hodnotou kvantitativního atributu. Horní a spodní hranice (podle stupnice kvantitativního atributu) střední části diagramu, tzv. „krabice“, označují v pořadí kvartily Q3 a Q1. Potom délka krabice znázorňuje *mezikvartilovou vzdálenost* IQR. *Medián* (také kvartil Q2) je označen horizontální čáry uvnitř krabice. Dvě čáry mimo krabici (v angličtině říkáme *whiskers*) rozšiřují graf o nejmenší a největší pozorování, tzv. *minimum* a *maximum*. Mimo jiné, krabicový graf lze využit pro identifikaci odlehlých pozorování. V tomto případě se jedná o tzv. *boxplot pravidlo*, které říká, že hodnoty ležící ve vzdálenosti větší než 1.5 x IQR pod Q1, resp. nad Q3, jsou potenciálně odlehlé hodnoty (na obrázku 4.2 jsou znázorněné jako červené body, které leží za hranicemi minimální a maximální hodnoty).



Obrázek 4.2: Příklad krabicového grafu, reprezentujícího rozložení jednotkových cen produktu *Product1*.

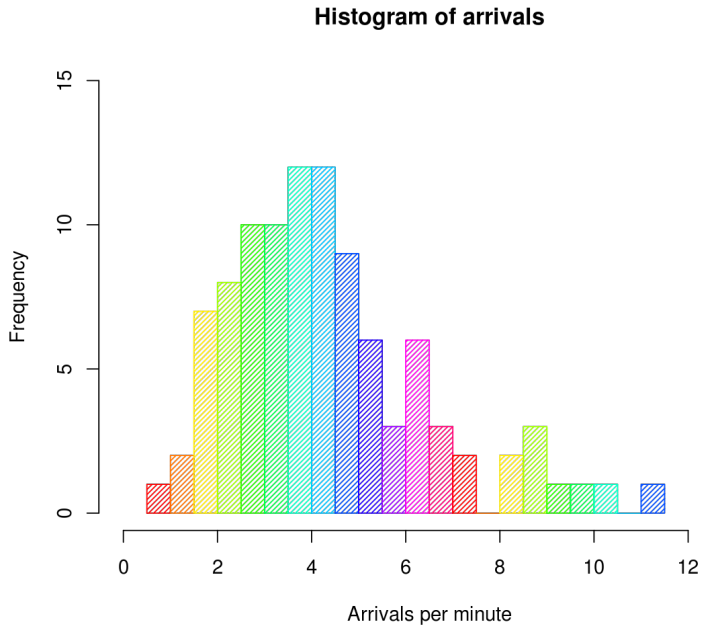
Rozptyl (angl. *variance*) je poměrně často užívanou mírou variability. Je klasickou charakteristikou náhodné veličiny. Jeho hodnota je definována střední kvadratickou odchylku od průměru. Jedná se o charakteristiku, která vyjadřuje variabilitu rozdělení souboru náhodných hodnot kolem její střední hodnoty. Hodnota rozptylu pro N pozorování pro numerický atribut X se vypočítá následovně, přičemž $\bar{x}$ je střední hodnotou veličiny X:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{4.2}$$

Z tohoto vzorce lze odvodit *směrodatnou odchylku* – jedná se o odmocninu z rozptylu náhodné veličiny X:

$$\sigma = \sqrt{\sigma^2(X)} \tag{4.3}$$

Kromě krabicového grafu, existuje mnoho dalších technik pro znázornění různých vlastností dat prostřednictvím grafické vizualizace. Takové techniky zahrnují kvantilové grafy, histogramy, bodové grafy, Q–Q grafy a mnoho jiných. Dané grafy jsou užitečné pro vizuální kontrolu dat ve fázi předzpracování. Například uvedený krabicový graf lze použit pro kontrolu odlehlých hodnot a následného odstranění šumu v datech. Nebo také široce používané pro vizualizaci dat jsou *histogramy*, neboli sloupcové grafy. Histogramy reprezentují distribuci hodnot numerických atributů. Výška sloupců vyjadřuje četnost veličiny v daném intervalu. Uvažujme, že máme k dispozici data z autobusového nádraží X, jejichž obsahem jsou informace o jednotlivých příjezdech [6]. Pomocí histogramu lze znázornit frekvenci, se kterou autobusy dorazí na nádraží za minutu – viz Obrázek 4.3.



Obrázek 4.3: Příklad histogramu reprezentujícího četnost příjezdů autobusů na autobusové nádraží X za minutu. (Převzato z [6].)

Znalost těchto výše uvedených základních statistických údajů o každém atributu může usnadnit přípravu dat pro dolovací úlohu. Jedná se o odstranění chybějících hodnot, vyhlazení dat, detekci odlehlých hodnot během čištění dat. Pochopení významu atributů a jejich hodnot může také napomoci při řešení nekonzistencí, které vznikly při integraci dat z různých zdrojů.

4.2.2 Čištění dat

V předchozích sekcích této kapitoly byly vymezeny základní problémy, se kterými se můžeme setkat při předzpracování dat a také způsobů charakterizace dat s cílem je identifikovat.

Velmi častým projevem nekvality dat je přítomnost chybějících hodnoty a šumu v datech. Cílem *čistění dat* je práve takové nesrovnalosti v datech detekovat a odstránit.

V této sekci se zaměřím především na odstranění chybějících hodnot. Součastí sumarizujících popisných charakteristik dat jsou údaje o počtu chybějících hodnot. Typicky tento údaj uvádí, v kolika řádcích tabulky chybí hodnota pro každý sloupec, reprezentující jednotlivý atribut. Neznamé hodnoty mohou negativně ovlivnit výsledek dolování. Navíc ne vždy je možné aplikovat dolovací algoritmus na data, ve kterých počet chybějících hodnot je přílíš velký nebo pokud chybějí velmi podstatná pro dolovací úlohu data. V tomto případě je potřeba provést odstranění neznamých hodnot. Existují některé techniky, jak se s takovými daty vyporadat. Nejčástější z nich jsou [36]:

1. *Smázání chybějících dat z datového souboru* (v anglickém jazyce říkáme této technice *Drop missing values*). Typicky jde o odstranění z databáze celého řádku, který obsahuje jednu nebo více neznamých hodnot. *Dropping* se doporučuje použit pouze v případě, když celkový počet neznamých hodnot je příblížně 0.01–0.5% dat.

2. *Náhrada chybějících hodnot manuálně nebo automaticky.* Vzhledem k tomu, že často krát data jsou velmi rozsáhla a obsahují stovky, tisice a víc neznamých hodnot, *manuální náhrada* s použitím potenciálních znalostí uživetele by měla velkou časovou náročnost. Navíc je možné, že uživatel takovými znalosti vůbec nedisponuje. Proto táto metoda je v praxi nepoužitelná. Opakem k tomu je *automatická náhrada*, která se v praxi používá často. Táto metoda nabízí několik variant nahrazení [37]:

   (a) Vyplnění neznamých hodnot *globální konstantou*.

   (b) Použití *míry centralní tendence* pro vyplnění hodnoty atributu. Například *medián*, nebo *aritmetický průměr* hodnot.

   (c) Vyplnění *nejpravděpodobnější hodnotou*. Jedná se o použití nějakého algoritmu strojového učení. Daný problém řeší úloha klasifikace a regrese s hledaným atributem jako cílem.

Další úlohou, kterou zahrnuje proces čištění dat, je tzv. *vyhlazení šumu* v datech. Zašumená data obsahují nesprávné nebo odlehlé hodnoty, které se obvykle odstraňují s cílem zlepšení kvality dat. Avšak v případě této práce odlehlé hodnoty mohou být naopak potenciálně užitečnými při odhalení podvodů. Proto nadál se nebudu zabývat vyhlazením šumu v datech, ale jeho použitím pro řešení zvolené případové studie. Například shlukování může být učinnou metodou pro nalezení odlehlých pozorování.

4.3 Typy dolovacich uloh

Po provedení základní analýzy dat a předzpracování, data jsou přípravena pro jádro procesu získávání znalostí – dolování z dat. Jak jíž bylo uvedeno, z pohledu strojového učení rozlišujeme tří různé přístupy pro dolování – učení s učitelem, učení bez učitele a kombinace učení s učitelem a bez učitele. Vzhledem ke zvolené případové studie, jedná se především o analýzu odlehlých hodnot. V svém řešení jsem rozhodla použit první dvě uvedené metody. V případě učení bez učitele se jedná o hlednání odlehlých hodnot na základě shlukování. V případě učení s učitelem jde o úlohu klasifikace. Obě dvě metody zahrnují celou řádu algoritmů pro získávání znalostí, avšak jenom některé z nich jsou vhodné pro řešení analytické ulohy, která se tyká detekce podezřeleho chování. V této části kapitoly se zaměřím na vysvětlení

právě takových algoritmů, které jsou v daném případe nevíc relevantní. Řešení projektu bude zaměřené na dolování s využitím dostupných nástrojů knihovny `scikit-learn`, která poskytuje prostředky pro analýzu odlehlých hodnot a klasifikaci.

4.3.1 Detekce odlehlých pozorování

Detekce odlehnych hodnot (v angličtině také často používáme pojem *Anomaly detection*) a shluková analýza jsou dvě úlohy dolování, které velice souvísí mezi sebou. Zatímco cílem shlukování je klasifikace datových objektů do tzv. shluků podle jejich podobnosti, cílem detekce odlehlých hodnot je zachytit ty výjímočné objekty, které svými vlastnostmi se od nalezených shluků výrazně líší. Obecně odlehlá pozorování se považují za šum v datech, který lze identifikovat některými statistickými metody (například výše zmíněný krabicový graf) a je nutné odstránit, jelikož nadále takové hodnoty mohou výsledky zkreslovat. Avšak odlehlé hodnoty mohou představovat také poteciálně užitečné znalosti, proto je vhodné je prozkoumat detailnější. Dále je uveden stručný přehled populárních technik strojového učení zaměřených na detekci anomálií.

Local Outlier Factor

Local Outlier Factor (LOF) je algoritmus, jehož koncept spočívá ve vyhledávání lokálních odlehlých objektů založených na hustotě [30]. Prvním kdo zavedl pojem *Local Outlier Factor* byl Markus Breunig. Uvádí, že je smyslúplné přidělit každému objektu skóre, určující jeho stupeň odlehlosti. Takové skóre nazýváme *lokální faktor odlehlosti* (LOF). Stupeň odlehlosti objektu závisí na tom, jak moc je objekt izolován vzhledém k okolí. Algoritmus srovnává lokální hustotu objektu, neboli jeho četnost výskytu ve sledovaném okolí, s lokálními hustotami jeho k nejblížších sousedů. Každému objektu z datového souboru je přiřazená hodnota, která udává hustotu okolí tohoto objektu. Skóre se určí na základě srovnání vypočtené lokální hustoty daného objektu s lokálními hustotami jeho sousedů. Objekty, které mají podstatně nížší hustotu, než jejích sousedi, považují se za odlehlé hodnoty.

Podlé LOF skóre pro daný počet sousedů k lze následujícím způsobem určit, jestli jednotlivé datové objekty lze považovat za odlehlé:

- $LOF(k) \sim 1$ znamená, že vzorek má podobnou hustotu, jako jeho sousedi
- $LOF(k) < 1$ uvádí, že vzorek má větší hustotu, než jeho sousedi. V tomto případě není odlehlým objektem (angl. *Inlier*)
- $LOF(k) < 1$ – datový objekt má nižší hustotu, než jeho sousedi. Je odlehlým objektem (angl. *Outlier*)

Isolation Forest

Isolation Forest je algoritmem, který detekuje anomálie izolováním datových vzorků, aniž by spoléhal na měření vzdálenosti mezi nimi nebo měření hustoty [34]. Pro dosažení tohoto cíle metoda využívá dvou vlastností anomálních hodnot:

- jedná se o malou množinu dat, tvořenou několika vzorky,
- hodnoty atributů vzorků se výrazně liší od běžných případů.

Jinými slovy anomálie jsou data, kterých je málo a jejich vlastnosti se výrazně liší od vlastností ostatních datových vzorků.

Algoritmus používá binární stromovou strukturu nazývanou *isolation tree* (*iTree*). Daná stromová struktura je založená na principu rozhodovacího stromu. Rozhodovací stromy jsou tvořeny statistickou metodou nazývanou *rekurzívní dělení* (angl. *Recursive partitioning*) [20]. Každý uzel stromové struktury je testem hodnoty určitého atributu [37]. Uzel je rozdělený do dvou uzlů reprezentujících třídy, do kterých datový objekt může být klasifikován. V tomto případě lze říct, že datové objekty budou „klasifikovány" do tříd *outlier*/*inlier*, neboli detekovány jako odlehlé objekty nebo přilehlé. Proces se nazývá *rekurzivní*, protože každý takový uzel ve stromové struktuře může být opakovaně rozdělován, neboli rozvětven, tak dlouho, dokud nebude dosaženo kritérií pro jeho zastavení. V případě klasifikační úlohy, proces rekurzivního dělení zastaví v okamžik, když datový objekt bude klasifikován do cílové třídy.

Algoritmus *Isolation Forest* používá náhodné rekurzivní rozdělení. Nejprve se náhodně vybere atribut. Následně se výbere hodnota náhodného rozdělení (angl. *random split value*), která patří do intervalu hodnot vybraného atributu. Protože rekurzivní dělení je v daném případě reprezentováno stromovou strukturou, počet rozdělení potřebných pro izolování vzorku je ekvivalentní délce cesty od kořenového uzlu ke koncovému uzlu. Kvůli náchylnosti k izolaci, anomálie jsou s větší pravděpodobností izolovány blíže ke kořenu stromu *iTree*. Zatímco normální datové vzorky jsou s větší pravděpodobnosti izolovány na hlouběji, blíž je ke koncovým uzlům stromu. V případě, že každý strom v tzv. lese vytváří kratší délky cest pro analyzovaný datový vzorek, je velmi pravděpodobné, že daný vzorek je odlehlou hodnotou. Výsledná délka cesty je průměrem délek cest takových náhodných stromů (lesu) [34].

4.3.2 Klasifikace

Klasifikace a regrese patří mezi prediktivními úlohy. V případě zvolené dolovací úlohy se jedná o predikci hodnot kategorických (třídy *fraud* a *ok* značící podvodné, resp. nepodvodné transakce), což je úlohou *klasifikace*. Vzhledem k tomu, že v případě klasifikace je k dispozici množina dat, u které známe do kterých tříd patří jednotlivé datové objekty, je táto úloha známá jako *učení s učitelem*, neboli *řízené učení*. Učení klasifikátora probíha „pod dohledem" na základě množiny dat, u které jsou cílové třídy známé [33]. V tomto spočívá zásadní rozdíl úlohy klasifikace od úlohy detekce odlehlých hodnot, u které hodnota třídy každého vzorku nemusí být předem známá. Proto taková úloha je známá jako *učení bez učitele*, pro „predikci" tříd nepotřebuje se učit na základě označených třídami dat.

Samotný proces klasifikace je složen z dvou podstatných fází – fáze učení a fáze testování. Avšak celý proces lze popsat sekvencí následujících kroků (viz Obrázek 4.4):

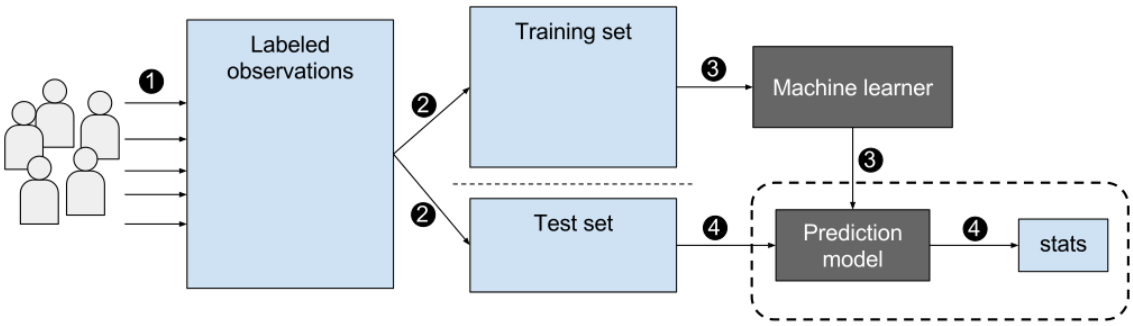
1. Výběr vzorků dat z databáze, u kterých je předem známa klasifikační třída.

2. Rozdělení množiny dat získané z kroku 1 na dvě podmnožiny – *trénovací* a *testovací*. Trénovací množina se použije ve fázi učení pro trénování klasifikátoru. Testovací množina bude sloužit pro otestování přesnosti klasifikátoru – viz sekce 4.4. Trochu později v této kapitole budou uvedeny způsoby, jakými lze rozdělit data na trénovací a testovací množinu.

3. Fáze trénování. Táto fáze zahrnuje učení klasifikátoru na trénovací množině. Jeho cílem je zjístit tzv. klasifikační pravidla, pomocí kterých klasifikátor nadále bude klasifikovat data – viz Obrázek 4.4, krok 3.

4. Fáze testování. „Natrenovaný“ klasifikátor pomocí klasifikačních pravidel určí do jakých tříd přiřadit datové objekty z testovací množiny – viz Obrázek 4.4, krok 4.

5. Vyhodnocení úspěšnosti modelu. Na základě znalostí pravdivých tříd datových objektů z testovací množiny, lze použit různé míry na vyhodnocení úspěšnosti klasifikátora. Podle získaných výsledku lze potom rozhodnout, zda je vhodné použit daný klasifikační model pro data, u kterých není známe, do kterých tříd patří [37]. Více informací o způsobu vyhodnocení modelů je uvedeno v sekci 4.4.



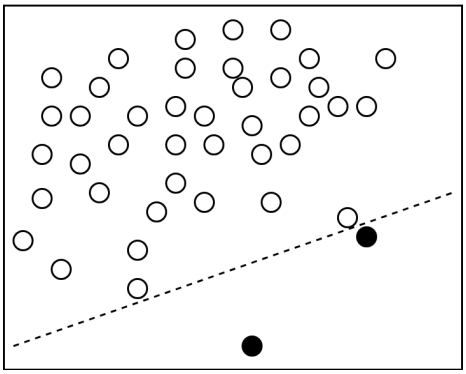
Obrázek 4.4: Proces klasifikace. (Převzato z [35].)

Binární klasifikace je základní a nejjednodušší variantou úlohy klasifikace [23]. Binární klasifikace klasifikuje pouze do dvou tříd, jejíž cílové atributy mohou nabývat jednu z dvou možných hodnot binárního atributu, například 0, 1. Dané třídy se také někdy nazývají *pozitivní třída* a *negativní třída*. Pozitivní třída je třída, kterou se snažíme predikovat. Například pokládáme otázku „*Je táto transakce podvodná?*“. Snažíme se předpovědět *podvodné* transakce, proto třídu, která značí takové transakce, označujeme jako *pozitivní* (např. třída *fraud*). Do této třídy patří vzorky, jejichž výslek analýzy odpovídá na danou otázku „ano“. Naopak *negativní třída* v tomto případě bude reprezentovat hodnoty, jejichž odpověď na otázku je „ne“ (např. třída *nonfraud* označující nepodvodné transakce). Binární klasifikaci pro tento příklad lze znázornit graficky – viz Obrázek 4.5. Černé body zastupují vzorky pozitivní třídy *fraud*, a naopak bíle body zastupují vzorky třídy *nonfraud*. Svislá čára představuje tzv. hraniční hodnotu (angl. *threshold*), podle které určujeme, jaké datové objekty lze považovat za podvodné [36]. Tento obrázek také dobře znázorňuje příklad, když četnosti jednotlivých tříd nejsou vyvážené – četnost výskytu vzorků třídy *fraud* je přibližně 4% oproti 96% vzorků, které patří do třídy *nonfraud*.

Rozdělení dat na trénovací a testovací množinu

Vrátím se ke kroku klasifikace, ve kterém před samotným učením klasifikátora je potřeba rozdělit datovou množinu se známými hodnotami tříd na dvě podmnožiny – trénovací a testovací. V této části vysvětlím některé významné metody pro rozdělení dat, mezi které patří:

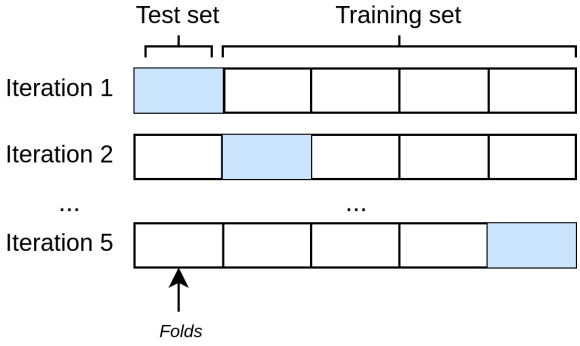
- Křížová validace *K-Folds*
- Křížová validace *Leave-One-Out*
- *Stratifikovaný KFold* validátor



Obrázek 4.5: Binární klasifikace.

- *Shuffle Split*, neboli náhodný permutační křížový validátor
- *Stratified Shuffle Split*, nebolí stratifikovaný náhodný validátor

Nejznámější a nejpoužívanější metodou je tzv. křížová validace *K-Folds* (angl. *K-Folds cross-validation*). Cílem validátoru je rozdělit datovou množinu na k stejně velkých podmnožin (v angličtině nazýváme *folds*), přičemž každá podmnožina je přibližně stejné velikosti [33] – viz Obrázek 4.6. Fáze trénování a testování probíhá v několika zadaných parametrem k iteracích, číže proces trenování-testování proběhne k krát. V každé iteraci jedna z podmnožin se použije jako testovací, zbývající se použijí k natrénování modelu. Na konci iterace se provede vyhodnocení přesnosti modelu. Výsledkem celého procesu je průměr hodnot, udávající přesnost klasifikátoru v každé iteraci. Víc informací o mírách hodnocení modelů je poskytnuto v sekci 4.4. *K-Folds* validátor se používá pro zvýšení přesnosti klasifikátoru. Bohužel v případě binární klasifikaci s velkou nevyvážeností tříd není garantováno, že v trénovací množině po rozdělení množiny dat budou zastoupeny hodnoty obou tříd. Kvůli malému množství označených dat jedné třídy může nastat situace, když trénovací množina nebude obsahovat žádné vzorky z dané třídy.



Obrázek 4.6: Křížová validace *K-Fold* s parametrem k=5.

Leave-One-Out je speciálním případem *K-Folds* křížové validace. V tomto případě parametr k se nastaví podle celkového množství datových vzorků, přičemž v každé iteraci jeden vzorek je „vynechán“ (*one left out*) pro testování [33]. Jelikož datová sada pro zvolenou případovou studií je příliš rozsáhlá, zvolit danou metodu také není vhodné. Víc o datové sadě pro zvolenou analytickou úlohu bude popsáno v kapitole 5.

Další tří metody rozdělení datové množiny poskytuje knihovna *scikit-learn* [21]. *Stratifikovaný K-Fold* křížový validátor (angl. *Stratified K-Fold*) je jinou verzí *K-Fold*, která vrací k stejně velkých stratifikovaných podmnožin, s výjimkou zachování četnosti vzorků pro každou třídu. *Shuffle Split* je validátorem náhodných permutací. Daný validátor potřebuje mít zadané dva parametry – počet iterací *n_splits*, během kterých se provede „zamíchaní" a náhodný výběr testovací množiny, a také parametr *test_size*, který udává poměr testovací množiny k trénovací. Například $test\_size = 0.2$ udává, že v každé iteraci po zamíchání datových objektů bude vybráno 20% vzorků pro trénovací množinu. Posledním validátorem, který bych chtěla uvést, je *stratifikovaný náhodný validátor* (angl. *Stratified Shuffle Split*). Tento validátor je kombinací stratifikovaného *K-Fold* validátoru a náhodného permutačního validátoru. To znamená, že rozdělení datové množiny na trénovací a testovací podmnožinu se provede náhodně a se zachováním četnosti jednotlivých tříd.

Naive Bayes

Jíž bylo zmíněno, že cílem zvolené analytické úlohy je detekovat podvodné transakce. Významnou vlastností dat, která jsou k dispozici, je velmi malá množina dat u kterých je stav vyšetření transakce známý (podvodná/v pořádku). Navíc známých podvodných transakcí je výrazně méně oproti známým transakcím, které se považují jako nepodvodné. Hodnota cílové třídy je výsledkem kontroly transakcí. Může nabývat jedné z dvou možných hodnot – {„fraud", „ok"}, viz kapitola 5. Cílem je naučit se konceptům, neboli pravidlům, podle kterých podvodné transakce se identifikují jako podvodné, a ostatní transakce jako v pořádku. Čelíme tak klasifikačnímu problému. V tomto případě se jedná o binární klasifikaci.

V případě řešení tohoto problému, využití klasifikačních nástrojů zahrnuje několik požadavků. Je podstatné, aby výsledek stavu transakce mel formu hodnocení pravděpodobnosti, že dána transakce patří k jedné ze dvou tříd [36]. Výsledkem některých klasifikačních algoritmů je pouze hodnota cílové třídy (angl. *label*), ke které je přiřazený datový objekt z testovací množiny dat. Pro daný problém pouze taková forma výsledku není dostatečná. Mnoho algoritmů v případě velké četnosti nevyvážených tříd provádí klasifikaci pouze na základě znalosti klasifikačních pravidel třídy s největší četností výskytu. Tím pádem případy z třídy s menší četností výskytu mohou být ignorovány a považovat se za šum. Proto potřebujeme klasifikaci, založenou na statistice, tzv. statistickou klasifikaci, která na určí třídu vzorku, pro kterou je největší pravděpodobnost, že daný prvek patří zde [37].

Vhodným pravděpodobnostním klasifikačním modelem je **Naive Bayes**. Je to tzv. naivní klasifikátor založeny na Bayesové teorii podmíněné pravděpodobnosti. Daný klasifikátor předpokládá, že vliv hodnoty atributu na cílovou třídu je nezávislý na hodnotách ostatních atributů. Tomu říkáme podmíněná nezávislost tříd (angl. *classconditional independence*) [33]. Takový přístup je zaveden s cílem zjednodušit náročnost výpočtů, a v tomto smyslu se považuje za „naivní". Bayesovská klasifikace je jednoduchá, avšak v praxi se prokázalo, že je velmi efektivní. Například táto klasifikace může být použitá pro detekci spamů v elektronické poštovné schránce na základě analýzy textových dat – viz Obrázek 2.2. Mimo jiné, pro trénování daného klasifikátoru je dostatečné mít relativně malou testovací množinu, což je výhodou v případě řešení zvoleného problému.

Bayesovou podmíněnou pravděpodobnost, je známá také jako *Bayesův vzorec*, lze formulovat následujícím způsobem:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}, \tag{4.4}$$

kde $P(A)$ je pravděpodobnost jevu A, $P(B)$ je pravděpodobnost jevu B, a $P(B|A)$ udává pravděpodobnost výskytu jevu A, pokud je známe, že nastal jev B. Říkáme, že pravděpodobnost A je podmíněna výskytem jevu B [2]. Analogicky $P(A|B)$ udává pravděpodobnost výskytu jevu B podmíněnou výskytem jevu A.

Použitím Bayesova vzorce klasifikátor vypočítá s jakou pravděpodobností vzorky $X_1, ..., X_p$ z testovací množiny patří do třídy c [36]:

$$P(c|X_1, ..., X_p) = \frac{P(X_1, ..., X_p|c)\, P(c)}{P(X_1, ..., X_p)} \tag{4.5}$$

Problém nevyvážené četnosti tříd

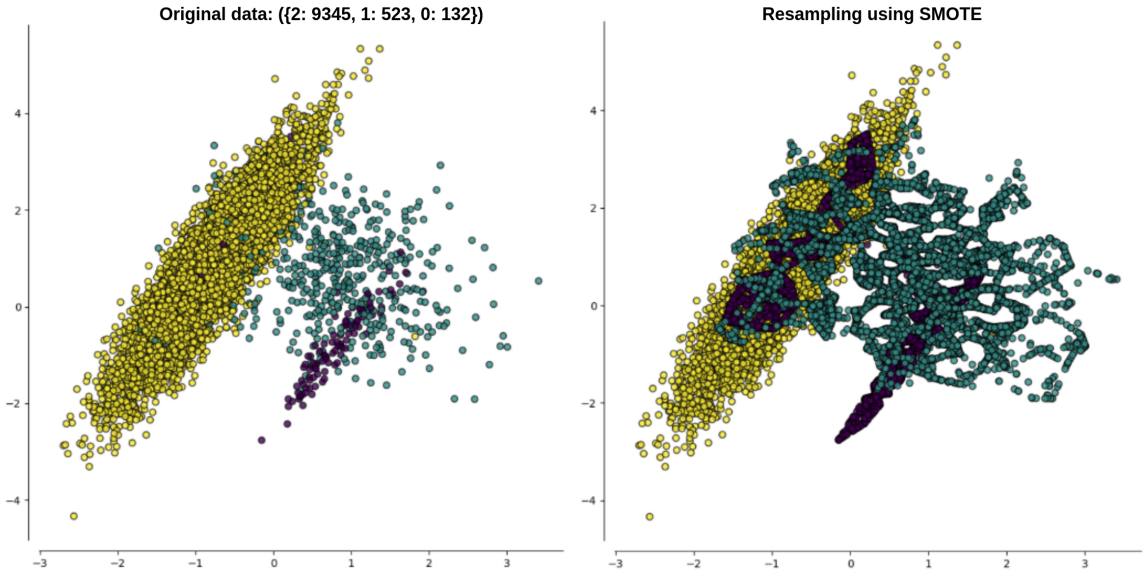
Výběr správního klasifikačního modelu bohužel nemusí zaručovat dosažení přesných výsledků při testování. To může být způsobeno již zmíněným problémem nevyvážené četnosti tříd. Existuje několik technik, které byly vyvinuty s cílem napomoct klasifikačním algoritmům překonat problémy, vyvolané třídní nerovnovahou. Obecně je to řešeno pomocí dvou metod [36]:

- *Over-sampling* – metoda spočívá v rozšíření množiny testovacích vzorků minoritní třídy, s požitím nějakého procesu pro replikaci vzorků.
- *Under-sampling* – metoda spočívá ve výběru malé množiny vzorků majoritní třídy, a nádsledně její přidání do množiny vzorků minoritní třídy.

Existuje mnoho variant těchto dvou obecných přístupů. Úspěšným příkladem je metoda **SMOTE** [36], která patří do metod s přístupem *over-sampling*. Hlavní myšlenkou této metody je uměle generování nových případů minoritní třídy s použitím nejbližších sousedů jednotlivých vzorků, což přivede k vyváženější datové množině. Příklad je uvedený na obrázku 4.7. Původní data jsou nevyvážena, četnost výskytu vzorků třídy *2* je 9345, třídy *1* – 523 a třídy *0* – 132. Zpráva na obrázku je výsledek použití metody SMOTE pro vygenerování nových vzorků dat minoritní třídy.

4.4 Výběr a vyhodnocení modelů

V předchozí podkapitole byly prezentovány jednotlivé metody, které se používají při detekci odlehlých pozorování a klasifikaci. Následuje další důležitý krok dolovacího procesu – vyhodnocení modelů. Tím rozumíme hodnocení úspěšnosti modelu v případě jeho aplikace na nové, neklasifikované do tříd data. Po natrénování klasifikátora, proběhne jeho testování na testovací množině dat. Následně na základě porovnání predikovaných tříd klasifikátorem s třídy známými z testovací množiny dat, lze určit podíl celkového počtu správných předpovědí. V tomto případě jde o metriku, které říkáme *skóre přesnosti* (častěji se používá pojem v angličtině *accuracy score*). Nicméně při práci s nevyváženým poměrem tříd není vhodné zvolit danou metriku při vyhodnocení modelu. Uvažujme, že máme k dispozici datový soubor, který se skládá z 1000 vzorků dat, neboli obsahuje celkově 1000 řádků, jejichž přiřazení k třídám je známé. Z nich jsou 970 datových vzorků zařazeny do třídy *NonFraud* a ostatních 30 vzorků jsou zařazeny do třídy *Fraud*. V tomto případě lze říct, že třída *Fraud* je vzácná, protože z celkového objemu k ní patří pouhých 3% dat. Dále předpokládáme, že na dané data byl aplikován klasifikační algoritmus a jeho skóre přesnosti dosahuje až 96%. Klasifikátor, který dosahuje přesnosti 96% s tak velkou nevyváženosti tříd nelze považovat za přesný.



Obrázek 4.7: Zleva je ukázáno původní rozložení datových objektů různých tříd, data mají nevyváženou četnost tříd. Zpráva je ukázka výsledku použití metody SMOTE pro vytvoření většího počtu vzorků minoritních tříd. (Převzato z [16]).

Standardní klasifikační modely, jako například klasifikace pomocí rozhodovacích stromů, mají tendenci předpovídat data pouze tzv. hlavní třídy (nebo také pozitivní třída). V tomto příkladě to je třída *NonFraud*, ke které patří 97% dat z celkové množiny. Ostatní 3% dat, které patří do tzv. „menšinové“ třídy (angl. *minor class*), mohou být považovány za šum [7]. Proto v této části práce se zaměřím na metriky, které budou relevantní v případě dat s nevyváženým poměrem tříd.

Matice záměn

Matice záměn (v angličtině *Confusion Matrix*) je reprezentována tabulkou velikosti *m krát m*, kde m ≥ 2 je počet předpovídaných tříd. Matice se často používá pro grafické znázornění úspěšnosti typicky klasifikačního modelu na testovací množině dat, pro které jsou známy skutečné hodnoty tříd. Matice záměn pro binární klasifikaci je složena z dvou řádků a dvou sloupců, kde sloupce tvoří předpovězené hodnoty tříd a řádky tvoří pravdivé hodnoty tříd – viz Tabulka 4.1.

		Predicted	
		Yes	No
Actual	Yes	TP	FN
	No	FP	TN

Tabulka 4.1: Matice záměn pro binární klasifikaci.

Matice záměn pro binární klasifikaci poskytuje čtyři různé výsledky, které jsou průníkem řádků (počet skutečných hodnot tříd) a sloupců (počet předpovězených hodnot tříd) matice [33]:

- *Pravdivě pozitivní* (angl. *True Positives*, TP) – reprezentuje počet datových objektů, které byly správně předpovězeny klasifikátorem jako pozitivní.
- *Falešně pozitivní* (angl. *False Positives*, FP) – reprezentuje počet datových objektů, které byly nesprávně označené klasifikátorem jako pozitivní, když jejich skutečné hodnoty tříd jsou negativní.
- *Pravdivě negativní* (angl. *True Negatives*, TN) – reprezentuje počet datových objektů, které byly správně předpovězeny klasifikátorem jako negativní.
- Falešně pozitivní (angl. *False Negatives*, FN) – reprezentuje počet datových objektů, které byly chybně označeny klasifikátorem jako negativní, když jejich skutečné hodnoty tříd jsou pozitivní.

Matice záměn je dobrým nástrojem pro analýzu úspěšnosti klasifikátora, resp. jak dobře může klasifikátor rozpoznávat data pocházející z různých tříd. *TP* a *NP* nám říkají, kdy predikce klasifikátorem byla správná, zatímco *FP* a *FN* nám říkají, kdy predikce klasifikátoru neuspěla. Celkový počet datových objektů, které byly podstoupeny klasifikační analýze, je $TP + FP + FN + TN$. Následně podle výsledku této matice lze odvodit některé užitečné míry pro vyhodnocení úspěšnosti klasifikace testovacích dat.

Míry pro vyhodnocení modelů

Nyní se podívejme na jednotlivé metriky pro vyhodnocení modelů. V případě nevyváženého poměru označených tříd za nejvíc vhodné se považují takové metriky, jako *přesnost* (angl. *Precision*), *senzitivita* (angl. *Recall*) a *F1 skóre*. Taktéž bude uvedená jíž zmíněna na začátku kapitoly populární míra *accuracy score*.

Skóre celkové přesnosti (*accuracy score*)

Skóre celkové přesnosti udává podíl správně klasifikovaných datových objektů ze všech datových objektů z testovací množiny dat:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.6}$$

Přesnost (*precision*)

Dána přesnost udává podíl správně klasifikovaných datových objektů pozitivní třídy ze všech predikcí pozitivní třídy:

$$Precision = \frac{TP}{TP + FP} \tag{4.7}$$

Senzitivita (*recall*)

Senzitivita je mírou, která udává podíl správně klasifikovaných datových objektů pozitivní třídy ze všech skutečně pozitivních případů:

$$Recall = \frac{TP}{TP + FN} \tag{4.8}$$

F1 skóre

Alternativním způsobem použití *přesnosti* a *senzitivity* je jejich kombinace do jedné míry. Tento přístup se nazývá *F1 skóre*. Míra *F1* je tzv. harmonickým průměrem obou metrik:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{4.9}$$

Kapitola 5

Řešení případové studie v jazyce Python

Obsahem této kapitoly je demonstrace řešení jednotlivých kroků procesu získávání znalostí z datv jazyce Python pro zvolenou případovou studií – *Detekce podvodných transakcí*. Před začátkem popisu procesu dolování budou stručně popsána data, která jsou k dispozici pro řešení této úlohy. Sekce 5.2 prezentuje dostupné prostředky v jazyce Python pro získání sumarizujících popisných charakteristik dat. Kapitola 5.3 navazuje na získané základní charakteristiky dat. Jejím cílem je ukázat přípavu dat pro aplikaci dolovacího algoritmu.

Poslední podkapitola 5.4 se věnuje demonstraci provedených experimentů s cílem detekovat podezřelé transakce. Budou ukázány různé nástroje knihovny *scikit-learn*, které umožní tuto analýzu provést.

5.1 Datová sada

Pro získávání znalostí z dat byla použita stejná datová sada, kterou použil autor případové studie ve svém řešení. Datový soubor ve formátu `Rdata` je volně dostupný na stránkách obsahujících podpůrné materiály ke knize *Data Mining with R* [4]. Jazyk R používá vlastní formát pro efektívní ukladání dat. Prvním krokem bylo převedení souboru z formátu `Rdata` do formátu `CSV`, vhodného pro analýzu v jazyce Python. Druhým krokem bylo nahrát data z souboru do objektu typu `DataFrame` (viz kapitola 3.2). Ukázka v jazyce Python:

```
df = pd.read_csv("sales.csv")
```

Data jsou reprezentována formou tabulky. Metoda `head()` knihovny `Pandas` umožnuje zobrazit prvních n řádků tabulky, kde n je nepovinným vstupním parametrem metody. Implicitní hodnota paramtru je *5*. Použitím dané metody jsem zjistila, že pravděpodobně při konverzi datového souboru do formátu `CSV` se vytvořil jeden sloupec navíc, obsahující číslování řádků tabulky. Tento sloupec je potřeba odstánít, což je prvním krokem přípravy dat. Tabulka obsahuje 5 atributů, jejíchž význam popíšu dále.

```
df.head(2)
>>>
    Unnamed: 0 ID  Prod    Quant   Val     Insp
0            1 v1  p1      182.0   1665.0  unkn
1            2 v2  p1     3072.0   8780.0  unkn
```

Popis atributů

- **ID** – jednoznačná identifikace prodavače, hodnoty jsou kategorické
- **Prod** – jednoznačná identifikace produktu, hodnoty jsou kategorické
- **Quant** – počet prodaných jednotek produktů, hodnoty jsou kvantitativní
- **Val** – celková tržba za prodej, hodnoty jsou kvantitativní
- **Insp** – výsledek analýzy na detekci podvodu může nabývat různých hodnot: `ok` – transakce kontrolována a považována, že je v pořádku; `fraud` – transakce je považována za podvodnou; `unkn` – transakce nebyla vyšetřena společností. Jedná se o hodnoty cílové třídy.

5.2 Sumarizující popisné charakteristiky dat

Knihovna `Pandas` (viz 3.2) jazyka Python poskytuje dvě metody pro získání základních statistických charakteristik dat – `describe()` a `value_counts()`. Metoda `describe()` umožňuje získat základní statistické charakteristiky o hodnotách kvantitativních atributů, daných mírami centrální tendence a rozptylem (viz *Sumarizující popisné charakteristiky dat* 4.2.1) bez ohledu na chybějící data.

Metoda `value_counts()` umožňuje získat základní informace o hodnotách kategorických atributů. Jazyk R poskytuje pro dané účely jednou funkci – `summary()`[1]. V tomto případě vstupním parametrem této funkce je datový soubor, který může obsahovat hodnoty kvantitativních a také kategorických atributů.

Výstupem metody `describe()` jsou základní statistické charakteristiky pro hodnoty numerických atributů *Quant* a *Val*. Příklad výpisu v Pythonu:

```
df.describe()
>>>
                Quant            Val
count     387304.000000  399964.000000
mean        8441.995585   14617.074312
std       918351.027958   69712.589097
min          100.000000    1005.000000
25%          107.000000    1345.000000
50%          168.000000    2675.000000
75%          738.000000    8680.000000
max    473883883.000000 4642955.000000
```

Výpis obsahuje navíc míru *count*, která udává počet záznamů s definovanými hodnotami jednotlivých atributů. Porovnáním této hodnoty u atributů *Quant* a *Val* lze odvodit, že atribut *Quant* obsahuje více neznamých hodnot, než atribut *Val*.

Pro hodnoty atributů kategorických, takových jak *ID*, *Prod* a *Insp*, lze zjistit četnost výskytu jednotlivých hodnot. Například:

```
df['ID'].value_counts().head()
df['Prod'].value_counts().head()
df['Insp'].value_counts().head()
```

[1] https://www.rdocumentation.org/packages/base/versions/3.6.0/topics/summary

```
>>>
v431     10159              p1125    3923
v54       6017              p3774    1824
v426      3902              p1437    1720
v1679     3016              p1917    1702
v1085     3001              p4089    1598
Name: ID, dtype: int64      Name: Prod, dtype: int64

unkn     385414
ok        14462
fraud      1270
Name: Insp, dtype: int64
```

Lepší přehled může dát relativní vyjádření (vztažené k celkovému počtu řádků) frekvence v procentech. Počet řádků (ekvivalentní počtu transakcí) je možné získat pomocí vlástnosti `.shape`:

```
df['Insp'].value_counts() / df.shape[0] * 100
# alternative
df['Insp'].value_counts(normalize = True)

>>>
unkn    96.078236
ok       3.605171
fraud    0.316593
Name: Insp, dtype: float64

# get rows count
df.shape[0]
>>> 401146
```

Předchozí výpis ukázal, že celkově jsou k dispozici 401146 řádků tabulky. Jedná se o rozsáhnou databází prodejních transakcí. Je známo, že z celkové množiny dat teměř 4% transakcí byly kontrolovány a považují se společností za platné (`ok`). Méně než 1% transakcí byly vyšetřeny a označeny jako podvodvodné (`fraud`). Zbyvající 96% transakcí nebyly kontrolovány, jejích stav není znám (`unkn`).

Počet prodejců a počet jednotlivých produktů, vyskytujících v prodejích, lze získat použitím metody `nunique()`. Metoda vratí počet unikatních prvků atributu:

```
df['ID'].nunique()
>>> 6016

df['Prod'].nunique()
>>> 4548
```

Počet prodejců dané společnosti je tedy 6016 a celkově se prodává 4548 různých produktů.

Určení chybějících hodnot jednotlivých atributů

Neznamé hodnoty objektu typu `DataFrame` jsou označovány knihovnou `Pandas` jako *NaN*. Je potřeba získat přehled o počtu takových hodnot jednotlivých atributů. Metoda `describe()`

neposkytuje informace o počtu chybějících hodnot jednotlivých atributů. Daný cíl lze splnit použitím metody `isna()` v kombinaci s metodou `sum()`. První metoda detekuje neznáme hodnoty, druhá metoda tyto výskyty spočítá:

```
df.isna().sum()
>>>
ID           0
Prod         0
Quant    13842
Val       1182
Insp         0
dtype: int64
```

Atributy `Quant` a `Val` obsahují značné množství neznamých hodnot. Problém by mohl nastát při výskytu neznámých hodnot obou atributů zároveň. Jelikož k dispozici je pouze omezené množství informací, nedostatek takových významných pro analýzu hodnot vylučuje možnost určit status transakce. Celkově takových transakcí, kterým chybějí hodnoty v obou sloupcích zároveň, je 888:

```
df.loc[df['Quant'].isna() & df['Val'].isna()].shape[0]
>>> 888
```

Přidání nového sloupce

Pro další analýzu dat bude užitečným přidat nový atribut, který bý udával jednotkové ceny produktů v jednotlivých prodejních transakcích. Ceny se vypočítají na základe následujícím způsobem:

```
df['Uprice'] = df['Val'] / df['Quant']
```

Přidání daného sloupce poskytuje nové možnosti pro zkoumání transakcí, například na základě distribucí cen produktu lze určit odlehlé hodnoty.

Určení nejdražších a nejlevnějších produktů

V daném kroku je popsán způsob, pomocí něhož lze zjistit informace o nejdražších a nejlevnějších produktech. Zaprvé je potřeba udělat seskupení jednotlivých produktů a aplikovat na každou skupinu agregační funkci s cílem získat souhrnné statistiky. Příklady agregačních funkcí jsou funkce max(), min(), mean(), first(), last() anebo také vlastní funkce.

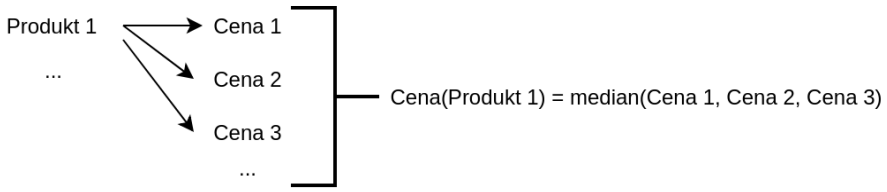
Použitím funkcí groupby() a median() lze najít typickou cenu pro jednotlivé produkty. Funkce mean() není úplně vhodná pro daný účel, protože se ceny za produkt v různých transakcích mohou velmi lišit, střední hodnota by mohla být zkreslena. Proto nejvhodnějším řešením je nalezení mediánu cen pro každou skupinu produktů (obrázek 5.1).

Chybějící hodnoty by měli byt igronorovány. V jazyce R se při použití agregační funkce nastaví parametr „na.rm[2]“ na hodnotu True, pomocí kterého se vyloučí chybějící hodnoty. V Pythonu skupiny neznamých hodnot jsou vyloučeny automaticky.

Řešení v Pythonu může mít různé formy zápisu, jeden z nich je:

```
tPrice = df.groupby(['Prod'])['Uprice'].median()
tPrice.head()
```

[2] **na.action**: funkce, která určuje, jaká operace se provede v případě, když data obsahují neznamé hodnoty



Obrázek 5.1: Určení typické ceny produktu.

```
>>>
Prod
p1      11.428571
p10     42.232846
```

Alternativním řešením je předat metodě **agg()** slovník[3] s mapováním názvů sloupců na operace, které se provedou v daném sloupci:

```
df.groupby(['Prod'], sort=False).agg({'Price': 'median'})
>>>
        Price
Prod
p1      11.428571
p2      10.877863
```

Výsledkem jsou objekty s víceúrovňovým indexováním. Pro seřazení vypočítaných cen je potřeba provést resetování indexu datového rámce do původního indexování pomocí metody **reset_index()** [4]. Obsahuje-li datový rámec MultiIndex[5], tato metoda také odstrání jeden, nebo více úrovní. Výsledkem použití metody je návrat do původního indexování datasetu typu RangeIndex[6] – výchozí typ indexu používaný objekty DataFrame a Series, pokud uživatel nenastaví žádný explicitní index.

Po aplikaci popsané metody lze získat 5 nejlevnějších produktů:

```
cheapestP = tPrice.reset_index().sort_values(['Uprice'],ascending=True)
cheapestP = cheapestP.reset_index(drop=True).head()
```

```
    Prod    Uprice
0   p560    0.016885
1   p559    0.018844
2   p4195   0.030259
3   p601    0.055223
4   p563    0.055764
```

Obdobným způsobem, pomocí nastavení parametru „ascending=False“ v metodě `sort_values()` (řázení od největšího čísla), získám 5 nejdražších produktů:

[3] angl. dictionary – datový typ v Pythonu; kolekce neuspořádaných, proměnlivých a indexovaných prvků. Slovníky se zapísuji se složenými závorkami a každá položka v slovníku se skládá z dvojice (klíč, hodnota).

[4] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.reset_index.html

[5] https://pandas.pydata.org/pandas-docs/stable/user_guide/advanced.html

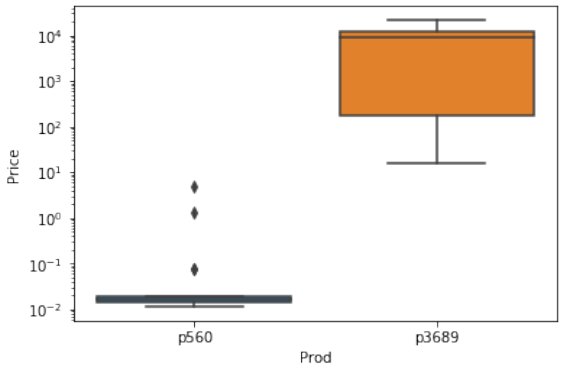
[6] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.RangeIndex.html

```
    Prod    Uprice
0   p3689   9204.195372
1   p2453   456.078431
2   p2452   329.313725
3   p2456   304.851485
4   p2459   283.811881
```

Pomocí krabicového grafu lze porovnat rozložení cen nejdražšího a nejlevnějšího produktu. Pro výkresení krabicového grafu byly použity prostředky knihovny `seaborn`. Táto knihovná poskytuje alternativní způsob vykreslení daného grafu, než například knihovna *Matplotlib*, který byl osobně příznán lepším.

```
top_prod = [topP.iloc[0,0], cheapestP.iloc[0,0]]
df_top_cheap = df[['Prod', 'Uprice']].loc[df['Prod'].isin(top_prod)]
sns.boxplot(x='Prod', y='Uprice', data=df_top_cheap).set_yscale('log')
```

Rozložení cen nejdražšího a nejlevnějšího produktu je zcela odlišné. Z tohoto důvodu osa Y je v grafu v logaritmickém měřítku („log scale“), jinak by mohlo dojít k nerozlišitelnosti hodnot nejlevnějšího produktu. Výsledek je vidět na obrázku 5.2:



Obrázek 5.2: Rozložení cen nejlevnějšího a nejdražšího produktu.

Analýza počtu prodaných produktů

Z hlediska analýzy počtu prodaných kusů jednotlých produktů, může být přínosne získat informace o nejméně/nejvíce prodaných vyrobcích. V tomto případě ale řešení spočívá v seskupení produktů a aplikaci agregační funkce `sum()` na hodnoty atributu `Quant`, označující počet prodaných kusů v dané transakci. Podle toho, jak se seřádí výsledek, lze získat identifikátory nejprodavanějších produktů, počet prodaných kusů jednotlivých výrobků a jejích cenu. Zajimavý výsledek se ukázal v případě produktů s nejmenším počtem prodaných kusů:

```
    Prod    Quant       Uprice
0   p2443   0.000000    nan
1 p2442   0.000000    nan
2 p1653   108.000000  10.833333
```

Počet prodaných kusů výrobků p2443 a p2442 se rovná 0, protože všechny jejich transakce obsahují neznámé hodnoty ve sloupci Quant, tím pádem není možné určit ceny jednotlivých produktů a vypočitát medián cen.

Úspěšnost prodejců

V této sekci pro zajímavost bude ukázana analýza úspěšnosti prodejců firmy. Úspěšné prodejce jsou takové lidi, které vydělali pro firmu nejvíce peněz. Lze uvést top 5 lidí, které mají pro firmu největší, resp. nejmenší přínos:

```
Most:                              Least:
    ID       Val                       ID       Val
0   v431     211489170.0          0    v3355    1050.0
1   v54      139322315.0          1    v6069    1080.0
2   v19      71983200.0           2    v5876    1115.0
3   v4520    64398195.0           3    v6058    1115.0
4   v955     63182215.0           4    v4515    1125.0
```

Celkový přínos 100 nejúspěšnějších lidí vyjádřený v procentech celkové tržby z prodeje lze vypočítat následujícím způsobem:

```
top_people['Val'][:100].sum()/df['Val'].sum() * 100
```

```
38.332773113245366
```

Naopak přínos 2000 nejméně úspěšných lidí:

```
group_people.reset_index().sort_values(['Val'],ascending=True)
```

```
1.988716285446494
```

Je zajímavé, že top 100 zaměstnanců generují až 38% z celkového příjmu společnosti. Oproti tomu, 2000 prodejců z 6016 generuje pouze necelých 2% příjmu. Firma by mohla použit týto údaje ke zlepšení efektivity práce, případně zvážit propuštění neefektivných zaměstnanců.

Základní přehled o počtu odlehlých hodnot

Cílem této části je vyzkoušet jednu z dostupných statistických metod pro identifikaci odlehlých hodnot, známou jako *pravidlo krabicového grafu* (viz 4.2.1). V daném případě se nebudou brát v úvahu dostupné informace o podvodných transakcích reprezentujících datové objekty, které vymykají obecnému chování. Cílem je vyzkoušet metodu založenou na krabicovém grafu a zjístit, jestli jsou v datech skutečně přítomné nějaké odlehlé hodnoty a jak vélké je množství takových objektů. Analýza bude provedena pro každnou skupinu cen příslušných produktů.

Autor případové studie používá pro identifikaci odlehlých hodnot atributu `Uprice` metodu `boxplot.stats`[7] dostupnou v jazyce R [36]. V jazyce Python existuje podobná metoda `boxplot_stats()`[8], která vrácí statistické údaje, použité k výskreslení krabicových grafů (viz 4.2.1). Avšak experiment ukázal podivné chování dané metody, která nevrácela korektní výsledky pro všechny produkty. Výpočet se testoval zvlášť na produktech `p1125` a `p538`. Ve výsledku se ukázalo, že u produktu p1125 nebyly detekovány žádné odlehlé hodnoty, navíc zbylé hodnoty vrácené metodou `boxplot_stats()` byly neznáme. Naproti tomu krabicový graf pro tento produkt odlehlé hodnoty detekoval. Produkt `p538` obsahoval korektní výsledek v obou případech.

[7] https://www.math.ucla.edu/~anderson/rw1001/library/base/html/boxplot.stats.html
[8] https://matplotlib.org/api/cbook_api.html#matplotlib.cbook.boxplot_stats

Jinou možností, jak získat statistické údaje, použité při vytvoření krabicového grafu, je použití metody `get_ydata()` po výkreslení grafu. Při velkém počtu produktů by bylo ale velmi obtižné a pomalé vykreslovat graf pro každý produkt a následně z něho získavat potřebné údaje. Z tohoto důvodu jsem se rozhodla naprogramovat vlastní agregační metodu založenou na pravidle krabicového grafu – `sum_outliers()`. Metoda zahrnuje výpočet IQR, minimální a maximální hodnoty, detekci odlehlých hodnot a jejich sumarizaci. Celkově bylo detekováno 29790 odlehlých objektů, které představují 7.42622% dat. Typicky daná metoda se používá s cílem detekovat odlehlá pozorování a odstranit je z datové množiny, jelikož takové hodnoty se považují za šum. V případě této úlohy odlehlá pozorování jsou naopak středem zájmu, proto odstráněny nebudou. Více problematikou detekce odlehlých hodnot se budu zabývat v podkapitole 5.4.

5.3 Čištění dat

Explorační analýza dat ukázala značné množství chybějících dat, které je potřeba z datové množiny odstranit s cílem zlepšit kvalitu dat před použitím dolovacího algoritmu. Táto podkapitola popisuje dvě techniky, které byly použity v řešení dané úlohy – smázání chybějících dat z datové množiny a vyplnění neznamých hodnot

Odstranění neznamých hodnot

Dřívejší analýza ukázala 888 transakcí, kterým chýbějí hodnoty v obou sloupcích `Quant` a `Val`. Je užitečným získat informace o poměru takových transakcí pro jednotlivého prodejce, resp. produkt. Procento transakcí obsahujících současně neznamé hodnoty v atributech Quant a Val ze všech transakcí daného produktu, resp. daného prodavače:

```
# The total number of transactions per product:
totP = df['Prod'].value_counts()
# The salespeople and products involved in the problematic transactions:
nas = df.loc[df['Quant'].isna() & df['Val'].isna()]
nas = nas[['ID', 'Prod']]
propP = 100 * nas['Prod'].value_counts()/totP

>>>
Products:                       Salespeople:
p2689   39.285714               v1237    13.793103
p2675   35.416667               v4254     9.523810
p4061   25.000000               v5248     8.333333
p2780   22.727273               v4038     8.333333
p4351   18.181818               v3666     6.666667
```

Hodnoty atributů `Quant` a `Val` jsou klíčovými pro výpočet ceny produktu a následnou analýzu stavu transakce. Transakce obsahující neznámé hodnoty u obou atributů nepředstavují žádnou užitečnost. Jediným možným řešením v daném případě je smazat všechny takové transakce. U některých produktů by to znamenalo odstranění více než 20% jejich transakcí. Například v případě produktu p2689.

Metoda `dropna()` knihovny `Pandas` umožní odstranění všech problematických transakcí. Vstupním parametrem `subset` se určí atributy, pro které budou vyhledany řádky obsahující neznamé hodnoty. Paremetrem `how` se v tomto případě určuje podmínka, při

které smazaní řádku z tabulky se provede pouze v případě, pokud budou nalezeny neznamé hodnoty u obou atributů zároveň. Příklad použití metody:

```
tClear = df.dropna(subset=['Quant', 'Val'], how='all')
```

Počet neznamých hodnot po provedené operaci se značně zredukoval, ale chybějících hodnot pořád zbývá hodně:

```
Quant    12954
Val        294
Price    13248
```

Analýza počtu prodaných produktů ukázala 100% neznamých hodnot atributů Quant u výrobků p2443 a p2442. Jelikož není možné vypočítat jejich typickou cenu, transakce daných produktů nejsou přínosné pro další výzkum a mohou být odstraněné:

```
tClear = tClear[tClear['Prod'] != 'p2442']
tClear = tClear[tClear['Prod'] != 'p2443']
```

Kdyby se při podobné analýze ukázalo, že některým prodavačům chybějí všechny informace o počtu prodaných produktů, smazaní takových transakcí by nebylo nutné. Chybějící hodnoty atributu Quant lze doplnit podle jiných zpráv obsahujících identický výrobek, z hodnoty tržby a jednotkové ceny. Lze provést také podobnou analýzu procentuálního zastoupení transakcí s chybějící hodnotou pro atribut `Val`. Provedeným výpočtem jsem zjistila, že nejvíc hodnot chybí produktu p1110, avšak takových transakcí je pouze 25%, proto chybějící hodnoty mohou být nahrazeny z hodnot atributů `Quant` a *Uprice*.

Procentuální zastoupení chybějících hodnot atributu `Val` zaměřené na prodavače také ukázalo přiměřené hodnoty. Nejvíce takových transakcí se objevilo u prodejců v5647 – 37.5%.

V této fázi jsou odstraněny všechny transakce, v jejichž případě nebylo možné použit rozumnou strategii pro nahrazení dat. Pro zbyvající transakce s chybějícími hodnotami lze použit metodu založenou na předpokladu, že transakce totožného produktu by měly mít přibližně stejnou cenu.

Náhrada chybějících hodnot

V předchozím kroku byly odstráněny neznamé hodnoty za použití redukce dat. Poslední fáze předzprácování spočívá v doplnění chybějících hodnot. Neznámé hodnoty atributů `Quant` a `Val` mohou být odstráněny automatickou náhradou určitým výpočtem (víc 4.2.2). Výpočet hodnot ve sloupci Quant zahrnuje dělení hodnoty atributu Val příslušného produktu jeho jedntkovou cenou. Chybějící hodnoty ve sloupcí Val se vypočítají vynásobením počtu prodaných kusů produktu, jeho mediánem cen, tzv. typickou cenou. Nalezení typické ceny bylo zmíněno dříve v sekci 5.2, popisující určení nejdražších a nejlevnějších produktů, výsledek je uložený do samostatného objektu `tPrice` typu `Series`. Náhrada chybějících hodnot atributu Quant by mohla vypadat následovně:

```
tClear['Quant'].fillna(value = tClear['Val']
                  / tPrice['Uprice'].loc[tPrice['Prod'] == tClear['Prod']])
```

Řešení používá metodu `pandas.DataFrame.fillna`[9] a ukázuje vyplnění neznamé hodnoty ve sloupci Quant hodnotou, která se vypočítá dříve popsaným způsobem, příčemž typická

[9] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.fillna.html

cena příslušného produktu se vybere z objektu *tPrice*. Jazyk Python bohužel neumožňuje porovnavání dvou objektu typu Series, jejichž struktura se liší, objekty *tPrice* a *tClear* by měli mít stejnou strukturu sloupců a jejich počet. Tím pádem v jazyce Python vyplnění hodnot takovým způsobem není možné provést. Moje řešení spočívá ve sloučení dvou daných objektů pomocí metody `merge`[10] a následným výpočtem chybějících hodnot. Jde o obdobu operace vnitřního spojení (angl. inner join), známého z SQL.

Výsledkem spojení objektů *tPrice* a *tClear* vznikne nový datový rámec `mFrames`, obsahující pouze ty záznamy, kde hodnoty atributu `Prod` prvního datového rámce se rovnají hodnotám atributu *Prod* druhého datového rámce. Nový datový rámec bude obsahovat všechny sloupce datového rámce `tClear` a navíc sloupec `meadianP`, označující mediány cen jednotlivých produktů. Následně lze vypočítat a doplnit chybějící hodnoty ve sloupcích `Quant` a `Val`. Všechny výpočty se provedou v rámci jednoho datového rámce. Výsledkem atributu `Quant` musí být celé číslo, proto je potřeba provést zaokrouhlení čísla nahoru. Řešení v jazyce Python může výpadat následovně:

```
mFrames = pd.merge(tClear, tPrice, on=['Prod'], how='inner')
mFrames['Quant'].loc[mFrames['Quant'].isna()] = mFrames['Val']/mFrames['medianP']
# Round UP Quant values and convert to int
mFrames.Quant = np.ceil(mFrames.Quant).astype(int)
mFrames['Val'].loc[mFrames['Val'].isna()] = mFrames['Quant'] * mFrames['medianP']
```

Posledním krokem je doplnění cen produktů za jeden kus v rámci každé transakce – hodnoty atributu `Val` se podělí hodnotami atributu `Quant`. Následně je potřeba odstránit z datového rámce všechny atributy, které nejsou relevantími pro analýzu dolováním z dat. Výsledkem bude datový ramec obsahující atributy `ID`, `Prod`, `Uprice`, `Insp`. Vyčištěná data mohou být uložena do souboru a připravena k dalšímu zpracování:

```
mFrames.to_csv("salesClean.csv", encoding='utf-8', index=False)
```

5.4 Dolování z dat

Po dokončení fáze předzpracování lze přistoupit k nejdůležitějšímu kroku celého procesu získávání znalostí – dolování z dat. V kapitole 4 byly prezentovány dva typy dolovacích úloh, které jsou klíčovými pro řešení této práce – *detekce odlehlých pozorování* a *klasifikace*. Pro každý typ úlohy byly vymezeny algoritmy relevantní pro řešení zvolené analytické úlohy. V případě detekce odlehlých hodnot se především jednalo o algoritmy *Local Outlier Factor* a *Isolation Forest*. V případě klasifikace se jednalo o pravděpodobnostní klasifikační model *Naive Bayes*. Cílem této části práce je popsat proces dolování, který zahrnuje aplikaci jednotlivých modelů pro získávání znalostí. Následně je cílem modely vyhodnotit a porovnat výsledky.

V sekci 5.4.1 bude popsán způsob, kterým budou výsledné modely vyhodnocovány a také porovnávány mezi sebou. Následně sekci 5.4.2 a 5.4.3 bude ukázán proces dolování v případě detekce odlehlých hodnot a klasifikace. Závěrem těchto sekcí je prezentace dosažených výsledků použitím vybraných modelů.

[10] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.merge.html

5.4.1 Určení způsobu vyhodnocení a porovnání modelů

Jelikož klasifikace a detekce odlehlých hodnot jsou dva různé typy dolovacích úloh, obvykle hodnocení jejich modelů probíhá také odlišnými způsoby. V případě této studie je potřeba ale specifikovat způsob vyhodnocení, který by umožnil porovnat dosažené výsledky získané různými modely těchto dvou typů dolovacích úloh.

Během provedení sumarizující charakterizace dat se ukázalo, že k dispozici je 4% dat, u kterých přiřazení do tříd `ok`/`fraud` je známo. V případě úlohy *učení bez učitele* se detekce odlehlých hodnot provádí bez úvahy na těchto znalostí. Nicméně znalost tříd v daném případě lze využit pro vyhodnocení úspěšnosti modelu. Jelikož výsledkem detekce anomálií je znalost, zda datový objekt je odlehlý nebo ne, úlohu lze přirovnat k binární klasifikaci, kde odlehlá hodnota zastupuje třídu `fraud` a naopak tzv. přilehlá hodnota zastupuje třídu `ok`. Neboli hodnoty výsledku detekce anomálních hodnot pro každý datový vzorek mohou být transformované na hodnoty odpovídajících tříd. Následně vyhodnocení takového modelu lze provést podobným způsobem, jak v případě klasifikační úlohy. Zde ale uvažujme, že testovací množina dat je množinou `všech` datových vzorků, jejichž přiřazení do tříd je známo. Jinými slovy, pro validaci modelu se použijí všechna data, u kterých je známa cílová třída. Potom na základě znalostí pravdivých tříd datových objektů z testovací množiny lze vyhodnotit úspěšnost „předpovědí“ modelu pro odpovídající množinu dat.

Hodnocení všech modelů tedy bude provedeno na základě metrik, které byly popsány v kapitole 4.4. Balíček `sklearn.metrics` knihovny `scikit-learn` poskytuje celou řadu nástrojů pro vyhodnocení modelů. Metoda `confusion_matrix()` umožňuje vypočítat matici záměn pro vyhodnocení přesnosti klasifikace. Navíc na stránkách knihovny lze dohledat příklad použití metody s grafickým vykreslením matice [3].

Metoda `classification_report()`[11] má výstup ve formátu textové zprávy, obsahující hlavní metriky klasifikace pro každou třídu – přesnost (v tomto případě se jedná o metriku *precision*, nikoliv *accuracy*), senzitivitu a F1 skóre. . Vstupními agrumenty metody jsou:

- Jednorozměrné pole obsahující pravdivé hodnoty cílových tříd. Uvažujme pole `y_true` datového typu `ndarray`.
- Jednorozměrné pole, obsahem kterého jsou předpovězené hodnoty cílových tříd. Uvažujme pole `y_pred` datového typu `ndarray`.
- Kvantitativní nebo kategorické hodnoty cílových tříd. V tomto případě kategorické hodnoty `ok`/`fraud`.

Příklad volání metody:

```
classes = ['fraud', 'ok']
classification_report(y_true, y_pred, target_names=classes)
```

Metriku *accuracy* lze získat použitím metody `accuracy_score()`:

```
accuracy_score(y_true, y_pred)
```

Nejdůležitější metrikou je metrika *recall*, která ukazuje schopnost modelu korektně předpovídat pozitivní třídu. V případě této úlohy – schopnost modelu správně predikovat podvodné transakce. To znamená, že cílem je ideálně dosáhnout 100% hodnoty senzitivity.

[11] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html#sklearn.metr

5.4.2 Detekce odlehlých pozorování

Pro experimentování v hledání odlehlých pozorování byly vybrány dvě metody strojového učení – *Local Outlir Factor* a *Isolation Forest* (viz 4.3.1).

Je potřeba určit, které atributy datové sady budou použity pro detekci anomálií. Kategorické hodnoty atributů *ID* a *Prod*, představující jednotlivé identifikátory zaměstnanců firmy, resp. identifikátory produktů, jsou pro daný typ úlohy irrelevantní. Vstupní množinou dat, která bude podstoupena analýze odlehlých hodnot, budou hodnoty numerického atributu *Uprice*. Jelikož zaměstnancemi firmy bylo uskutečněno více prodejů jednotlivých produktů, následně v dané databázi lze pozorovat více transakcí, ve kterých se vyskytuje identifikátor každého konkterního produktu. To znamená, že jsou k dispozici více jednotkových cen u každého produktu, cena se může lišit v každé transakci. Předpokládá se, že ceny by měly mít přibližně stejné rozložení.

Proces detekce odlehlých hodnot je znázorněn na obrázku 5.4.2) a bude zahrnovat následující kroky:

1. Příprava dat pro provedení detekce odlehlých hodnot. Je potřeba rozdělit datový soubor – rozdělit jednotkové ceny do skupin podle produktů (viz levá část obrázku5.4.2). Pro tento účel se použije metoda `groupby()` knihovny `Pandas`.

2. Každá skupina cen příslušných produktů bude podstoupena detekci odlehlých hodnot aplikací algoritmů *LOF* a *Isolation Forest*, která zahrnuje:

   (a) Inicializaci modelů. Model `IsolationForest` byl inicializován s následujícími parametry $contamination = 0.1$ a $random\_state = 42$. Model `LocalOutlierFactor` byl inicializován s parametry $contamination = 0.1$ a počtem sousedů $n_neighbors = 7$.

   (b) Aplikace metody fit_predict() příslušného algoritmu. Doslova metoda „*přizpůsobí model trénovací množině X a vrátí hodnoty cílového atributu (label)*“ [9]. Trénovací množinou v tomto případě rozumíme množinu numerických hodnot, reprezentující jednotkové ceny příslušné skupiny. Cílový atribut bude reprezentovat hodnotu *-1* pro odlehlé objekty, resp. hodnotu *1* pro normální datový objekt. Hodnoty {-1, 1} jsou určeny podle *LOF skóre* (také *faktor lokální odlehlosti*) a parametru `contamination`, udávající poměr odlehlých hodnot.

   (c) Obnovení datového rámce získanými výsledky do nově vytvořených sloupců *IF_Insp* a *LOF_Insp* pomocí metody `update()`[12].

3. Transformace hodnot {-1, 1} na odpovídající kategorické hodnoty tříd {`fraud`, `ok`}.

Příklad pseudokódu, prezentujícího výše popsané kroky procesu detekce odlehlých hodnot:

```
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor

# Outlier detection process
def detectOutliers (prodPrices):
    ...
    for name, algorithm, resultCol in anomaly_algorithms:
        y_pred = algorithm.fit_predict(X)
```

[12] https://www.programiz.com/python-programming/methods/dictionary/update

```
        ...
        # update original dataset with algorithm's predicted labels
        df.update(prodPrices)
...
# Define outlier detection methods to be compared
anomaly_algorithms = [
    ("Local Outlier Factor", LocalOutlierFactor(n_neighbors=neighbors,
                        contamination=outliers_fraction), "LOF_Insp"),
    ("Isolation Forest", IsolationForest(behaviour='new',
                                         contamination=outliers_fraction,
                                         random_state=42), "IF_Insp")]
# Get groups of products with their unit prices
gb = df.groupby('Prod')
# Iterate though products
for group in gb.groups:
    detectOutliers(gb.get_group(group))
```

Vyhodnocení modelů

Aplikací metod *Local Outlier Factor* a *Isolation Forest* podařilo se dosáhnout dobrých výsledků. V případě první metody, jejíž koncept spočívá ve vyhledávání lokálních odlehlých objektů založených na hustotě, model korektně detekoval odlehlé hodnoty v 70% případů. Celková přesnost modelu dosáhla 86%. V případě druhé metody, která je založená na principu izolování datových vzorků, detekce odlehlých pozorování byla ještě víc úspěšná – model detekoval správně v 82% případů. Celková přesnost modelu je 79%. Tabulka 5.1 ukazuje všechny metriky pro porovnání obou modelů pro třídu `fraud`.

Algorimus	Precision	Recall	F1 score	Accuracy
`LocalOutlierFactor`	0.34	0.70	0.46	0.86
`IsolationForest`	0.25	0.82	0.39	0.79

Tabulka 5.1: Výsledky detekce odlehlých hodnot pomocí algoritmů *Local Outlier Factor* a *Isolation Forest*.

5.4.3 Klasifikace

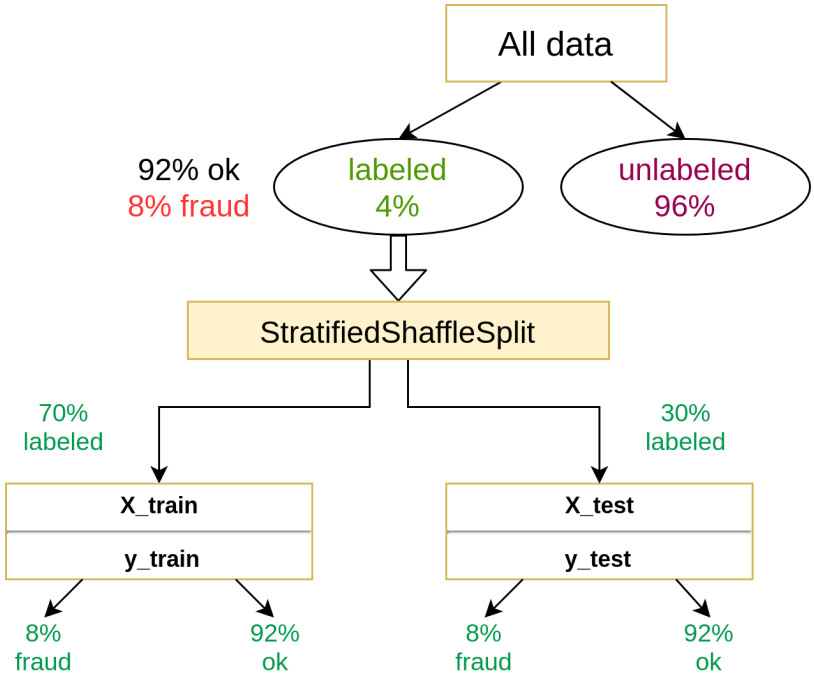
Nejvhodnějším klasifikačním modelem pro řešení této případové studii je požití probabilistického klasifikátoru Naive Bayes. Metody Naive Bayes patří k algoritmům strojového učení, známe jako učení s učitelem, které jsou založeny na aplikaci Bayesové věty s „naivním“ předpokladem podmíněné nezávislosti mezi každou dvojící atributů (více 4.3.2). Obecně metoda Naive Bayes je známá tím, že se primárně používá pro analýzu hodnot kategorických atributů. Knihovny scikit-learn poskytuje čtyři různé modely Bayesovského klasifikátoru [11]:

- `Gaussian Naive Bayes` – pro analýzu hodnot spojitých atributů;
- `Multinomial Naive Bayes` – pro diskrétní hodnoty atributů, které jsou mnohonásobně distribuovány, například počty výskytu slov;

- `Complement Naive Bayes` – rozšíření předchozí varianty;
- `Bernoulli Naive Bayes` – pro hodnoty s multivariačním binomickým rozdělením.

Bohužel ale prostředky knihovny neposkytují implementaci, kterou by bylo možné použít pro klasifikaci kategorických atributů. Nicméně kategorické hodnoty atributů *ID* a *Prod* lze transformovat na hodnoty diskrétní, a následně aplikovat klasifikační model `Gaussian Naive Bayes` pro klasifikaci hodnot všech třech atributů – *ID*, *Prod* a *Uprice*.

Proces klasifikace odpovídá již zmíněným dříve jednotlivým krokům (viz 4.3.2). S adaptací na řešení případové studie proces lze znázornit následujícím způsobem na obrázku 5.3. Skládá se z následujících kroků:



Obrázek 5.3: Proces klasifikace při řešení úlohy detekce podvodných transakcí.

1. Výběr vzorků dat z datového rámce, u kterých je předem známá klasifikační třída – {`fraud`, `ok`}. Takové data stanoví pouze 4% z celkového objemu dat.
2. Volba vhodného validátoru. Z obrázku 5.3 je vidět, že z celkové množiny dat označených cílovou třídou jsou 92% vzorků označených třídou `ok` a pouze 8% vzorků jsou zařazeny do třídy `fraud`. Jedná se o nevyváženou četnost tříd, proto je potřeba zvolit takovou metodu rozdělení datové množiny, u níž budou zachovány poměry četnosti vzorků jednotlivých tříd. Vhodným validátorem splňujícím tyto podmínky je validátor *Stratified Shuffle Split*. Parameter $test\_size = 0.3$ udává poměr testovací množiny jako 30%. Zbyvajících 70% dat se použije na trénování klasifikátoru. Výsledkem rozdělení datové množiny vzniknou čtyři proměnné datového typu *array*:
   - `X_train` – reprezentuje hodnoty atributů trénovací množiny
   - `y_train` – reprezentuje odpovídající pro dané atributy hodnoty tříd z trénovací množiny

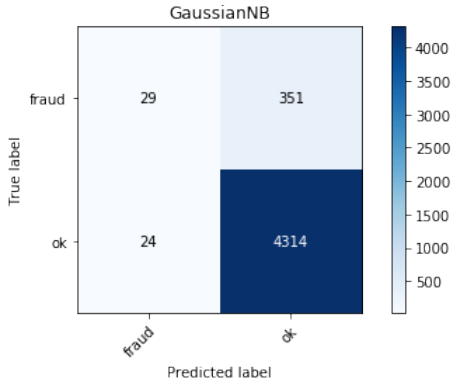
- `X_test` – obsahuje hodnoty atributů z testovací množiny
- `y_test` – obsahuje hodnoty tříd příslušných hodnot atributů z testovací množiny

3. Fáze trénování. Na začátku proběhne inicializace modelu `GaussianNB()`. Dále následuje samotná fáze trénování, která zahrnuje učení klasifikátoru na trénovací množině reprezentované proměnnými `X_train` a `y_train`. Pro tento účel se použije metoda `fit()`.

4. Fáze testování. V tomto kroku klasifikační model je natrénovaný a může být otestován na datech v testovací množině. Pomocí metody `predict()` model předpovědí třídy pro datové vzorky uložené v proměnné `X_test`. Následně porovnáním předpovězených hodnot tříd s hodnotami uloženými v proměnné `y_test` lze vyhodnotit přesnost použitého klasifikačního modelu.

Vyhodnocení modelu

Bohužel detekce odlehlých hodnot pomocí úlohy klasifikace nebyla vůbec úspěšná. Matice záměn model `GaussianNB` je znázorněna na obrázku 5.4. Je vidět, že pouze 29 vzorků bylo předpovězeno modelem správně z celkově 380 vzorků označených třídou `fraud`.



Obrázek 5.4: Matice záměn pro klasifikační model `Gaussian Naive Bayes`.

Detailnější znázornění výsledků je uvedeno v tabulce 5.2. Hodnota *recall* je v tomto případě pouze 0.08, což znamená, že pouze v 8% případech model klasifikoval vzorky do třídy `fraud` správně. Avšak metrika *accuracy* udává výsledek až 92%. Tento případ je právě důkazem toho, že *přesnost* není vhodnou metrikou pro hodnocení úspěšnosti modelu s nevyváženým poměrem tříd.

Precision	Recall	F1 score	Accuracy
0.55	0.08	0.13	0.92

Tabulka 5.2: Výsledky klasifikace pomocí klasifikačního modelu `GaussianNB`.

Vyvážení poměru tříd

S cílem pokusit se zlepšit výsledky klasifikace byla použita metoda SMOTE pro vyvážení poměru tříd. Metoda je založená na zvětšení počtu vzorků minoritní třídy (*over-sampling*)

(viz 4.7). Příklad inicializace modelu SMOTE a jeho použití pro vygenerování dodatečných vzorků:

```
# Over-sample observations with fraud class
sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())
```

Z matice záměn 5.4 lze odvodit, že testovací množina obsahuje celkově $29 + 351 = 380$ vzorků, které skutečně patří do třídy `fraud` a $29 + 4314 = 4343$ vzorků, které patří do třídy `ok`. Výsledkem použití metody SMOTE je stejný počet vzorků označených třídou `fraud` a třídou `ok`. Následně se zopakují kroky 3 a 4 z výše popsaného procesu klasifikace. Dále lze provést vyhodnocení modelu po aplikaci metody SMOTE – viz tabulka 5.3.

Precision	Recall	F1 score	Accuracy
0.47	0.12	0.19	0.92

Tabulka 5.3: Výsledky klasifikace pomocí klasifikačního modelu `GaussianNB` po vygenerování dodatečných vzorků třídy `fraud` metodou SMOTE.

Díky této metodě se podařilo dosáhnout lepšího výsledku. Hodnota míry *recall* je o 4% lepší oproti předchozímu výsledku. Nicméně hodnota *recall* je stále velmi nízkou. Aplikace tohoto modelu na nová data by nemohla zabezpečit věrohodnost výsledků. Experiment lze vyhodnotit jako neúspěšný.

Kapitola 6

Vhodnost použití jazyka Python pro řešení podobných úloh

Jazyk Python a jazyk R jsou momentálně nejpoužívanějšími jazyky v oblasti získávání znalostí z dat. Proč tomu tak je? Cílem této práce bylo seznámit se s prostředky pro dolování z dat, které poskytuje jazyk Python. Následně je aplikovat pro řešení zvolené analytické úlohy. Především se jednalo o využití dostupných metod pro analýzu dat, které poskytují různé knihovny jazyka Python.

Po dokončení procesu získávání znalostí z dat, prezentovaným v této práci, mohla bych shrnout důvody použití jazyka Python pro datovou analýzu do několika následujících bodů:

- Přenositelnost. Jazyk je nezávislý na platformě, na které běží [29].
- Intuitivní a lehce čitelný kód. Při pohledů na kód napsaný v jazyce Python většinou i pro nezkušeného programátora je snadné pochopit, v čem spočívá jeho funkce, co provádí.
- Obsahuje nesmírně velké množství vestavěných funkcí, díky čemu je možné efektivně zkrátit kód. Hlavně umožňuje zaměřit se primárně na datovou analýzu, a méně na implementaci požadovaných prostředků.
- `Pandas` pro předzpracování dat. Knihovna poskytuje hodně prostředků pro explorační analýzu s cílem pochopit data. Také samotné prostředky pro přípravu dat před aplikací dolovacího algoritmu.
- Knihovna `scikit-learn`. Knihovna vznikla v roce 2007 v rámci projektu `Google Summer of Code` a momentálně je nejpoužívanější knihovnou pro dolování z dat. Použité prostředky této knihovny demonstrované v práci jsou pouze malou ukázkou ze všech možných nástrojů.
- Ostatní významné knihovny pro podporu dolování z dat, popsané v kapitole 3 – NumPy, Plotly, apod.
- `Jupyter Notebook`. Použití interaktivního nástroje ve formě příkazových řádků usnadňuje provedení analýzy dat. Jednotlivé části kódu lze snadno upravovat bez nutnosti spouštět celý skript.

Druhou otázkou, na kterou zodpovím je: *Poskytuje jazyk Python dostatečné prostředky pro řešení takových úloh, jako je detekce odlehlých pozorování?* Odpověď je ano. Demon-

strované prostředky v této práci jsou tomu důkazem. Slouží ukázkou několika metod, kterými danou úlohu lze řešit. Avšak existují i další způsoby. Například kromě prostředků pro detekci odlehlých hodnot na základě neřízeného učení (*unsupervided learning*), knihovna `scikit-learn` také uvádí metodu, která se nazývá *Detekce novot* (angl. *Novelty detection*). Detekce novot je metodou *kombinovaného učení s učitelem i bez učitele*, kde učení se provádí jen na množině dat, která patří k majoritní třídě. Následně natrénovaný model se aplikuje na množinu dat, jejíž hodnoty tříd nejsou známě, anebo známe v kontextu odlehlých hodnot. V tomto případě se předpokládá, že anomálie mohou naopak tvořit shluky dat [12]. Knihovna `scikit-learn` poskytuje čtyří možné modely pro detekci novot s cílem detekovat anomálie. Jsou to rozšíření modelů `neighbors.LocalOutlierFactor`, `ensemble.IsolationForest`, také modely `covariance.EllipticEnvelope` a `svm.OneClassSVM`.

Během řešení úlohy případové studie jsem se setkala i s některými problémy, které bych v daném případě považovala na nevýhodu jazyka Python. Jedná se o klasifikaci hodnot kategorických atributů. V jazyce Python neexistuje zatím žádný model založený na naivním Bayesovém klasifikátoru pro daný typ atributů. O tomto svědčí nahlášení daného problému s identifikátorem *10856*[1]. Pro provedení klasifikace pro hodnoty kategorických atributů bylo potřeba provést transformaci na diskrétní hodnoty všech 4546 identifikátorů produktů a 6016 identifikátorů zaměstnanců firmy (uvažujme vyčištěná data). Naproti jazyk R poskytuje implementaci tohoto klasifikátoru pro oba dva typy atributů – kategorické a numerické nevyžadující dodatečnou přípravu dat před jeho aplikací.

[1] https://github.com/scikit-learn/scikit-learn/issues/10856

Kapitola 7

Závěr

Jedním z cílů této práce bylo pochopit základním principům a technikám získávání znalostí z dat. Dalším cílem bylo poskytnout řešení v jazyce Python pro zvolenou případovou studii, jejíž tématem bylo *Detekce podvodných transakcí*. Dána úloha je velmi zajímavá a je netriviální z hlediska toho, že jsou k dispozici data, která obsahují velmi malou množinu vzorků označených cílovou třídou. Takové případy stanoví pouze 4% dat z jejiich celkového objemu. Jedná se o transakce prodeje, které jíž byly vyšetřeny a výsledkem jejich kontroly je stav `fraud` nebo `ok`. Navíc poměr normálních transakcí a podezřelých je velice nevyvážený, podvodné transakce jsou vzácnými případy. Pro detekci možných podvodů byly využity algoritmy strojového učení, založené na učení bez učitele – Local Outlier Factor a Isolation Forest, a také metoda založená na učení s učitelem – Bayesovská klasifikace. V praxi se prokázalo, že úloha klasifikace není vhodná pro řešení takového typu problému. Zaprvé, protože trénovací množina může být nedostatečná pro dobré naučení klasifikátoru. Zadruhé, problém nevyvážené četnosti tříd může způsobit to, že klasifikátor se bude učit pouze na základě vzorků majoritní třídy (třída `ok`). Vzorky minoritní třídy v tomto případě mohou být považovány za šum v datech a být ignorovány. Dobrých výsledků bylo dosaženo pomocí metod založených na principu učení bez učitele. Použitím algoritmu `Isolation Forest` bylo dosaženo až 82% senzitivity. Jinými slovy, tento model dokázal úspěšně detekovat odlehlé hodnoty až v 82% případů. Když algoritmus `Local Outlier Factor` byl úspěšný v daném smyslu v 70% případů. Na základě znalostí výsledků, nejpřesnějším modelem je tedy `Isolation Forest`, který se může použit pro detekci podvodných transakcí na nových datech, jejichž stav vyšetření není známý.

Do budoucna bylo by zajímavé rozšířit práci o experimenty s modely kombinovaného učení s učitelem a bez učitele (*unsupervised learning*). Taktéž vyzkoušet některé statistické metody s cílem detekovat odlehlé hodnoty. V případě klasifikace by bylo zajímavé vyzkoušet a porovnat mezi sebou jiné metody pro vyvážení poměru tříd, například ADASYN. Nebo naopak metody založené na principu *under-sampling*. Co se tyká metod založených na učení bez učitele, bylo by možné také vyzkoušet jiné algoritmy založené na shlukování.

Práce mi dala možnost seznámit se s velkou rychle vyvíjecí oblastí získávání znalostí z dat. Navíc poskytla mi možnost zúčastnit se výzkumu, který se tyká detekci odlehlých hodnot při omezeném počtu informací, a také způsobů řešení tohoto problému. Hodně času bylo věnováno nastudováním různých materiálů potřebných pro úspěšné zvládnuti této úlohy jak z teoretické stránky, tak i z praktické. Znalosti získané během řešení této práce považují za pozitivní a cenní zkušenost.

Literatura

[1] *10 Minutes to pandas*. [Online; navštíveno 30.02.2019].
URL
https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

[2] *Bayesova věta*. [Online; navštíveno 06.05.2019].
URL https://cs.wikipedia.org/wiki/Bayesova_věta

[3] *Confusion Matrix*. [Online; navštíveno 04.04.2019].
URL https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py

[4] *Data Mining with R. Datasets*. [Online; navštíveno 02.10.2018].
URL http://www.dcc.fc.up.pt/~ltorgo/DataMiningWithR/

[5] *Graph Databases: Talking about your Data Relationships with Python*. [Online; navštíveno 02.10.2018].
URL https://medium.com/labcodes/graph-databases-talking-about-your-data-relationships-with-python-b438c689dc89

[6] *Histogram*. [Online; navštíveno 06.05.2019].
URL https://en.wikipedia.org/wiki/Histogram

[7] *How to handle Imbalanced Classification Problems in machine learning*. [Online; navštíveno 06.05.2019].
URL https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem

[8] *Kvantil*. [Online; navštíveno 06.05.2019].
URL https://cs.wikipedia.org/wiki/Kvantil

[9] *Local Outlier Factor*. [Online; navštíveno 30.02.2019].
URL https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html#sklearn.neighbors.LocalOutlierFactor

[10] *Matplotlib*. [Online; navštíveno 05.05.2019].
URL https://matplotlib.org/index.html

[11] *Naive Bayes*. [Online; navštíveno 30.02.2019].
URL https://scikit-learn.org/stable/modules/naive_bayes.html

[12] *Novelty and Outlier Detection*. [Online; navštíveno 10.5.2019].
URL https://scikit-learn.org/stable/modules/outlier_detection.html

[13] *NumPy*. [Online; navštíveno 05.05.2019].
URL https://www.numpy.org

[14] *Numpy Basics. Array creation*. [Online; navštíveno 05.05.2019].
URL https://www.numpy.org/devdocs/user/basics.creation.html

[15] *Openstack software*. [Online; navštíveno 05.05.2019].
URL https://www.openstack.org/software/

[16] *Over-sampling*. [Online; navštíveno 06.05.2019].
URL https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html#smote-adasyn

[17] *Plotly Python Open Source Graphing Library*. [Online; navštíveno 05.05.2019].
URL https://plot.ly/python/

[18] *Python Data Analysis Library*. [Online; navštíveno 30.02.2019].
URL https://pandas.pydata.org/

[19] *Quickstart tutorial*. [Online; navštíveno 05.05.2019].
URL https://www.numpy.org/devdocs/user/quickstart.html

[20] *Recursive partitioning*. [Online; navštíveno 06.05.2019].
URL https://en.wikipedia.org/wiki/Recursive_partitioning

[21] *Scikit-learn. Machine learning in Python*. [Online; navštíveno 30.02.2019].
URL https://scikit-learn.org

[22] *Selecting Subsets of Data in Pandas*. [Online; navštíveno 02.10.2018].
URL https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c

[23] *Statistical classification*. [Online; navštíveno 06.05.2019].
URL https://en.wikipedia.org/wiki/Statistical_classification

[24] *Streaming algorithm*. [Online; navštíveno 28.04.2019].
URL https://en.wikipedia.org/wiki/Streaming_algorithm#Data_stream_model

[25] *The Jupyter Notebook*. [Online; navštíveno 05.05.2019].
URL https://jupyter.org/index.html

[26] *The Python Tutorial*. [Online; navštíveno 30.02.2019].
URL https://docs.python.org/3/tutorial/

[27] *What is supervised learning*. [Online; navštíveno 02.10.2018].
URL https://www.quora.com/What-is-supervised-learning

[28] *What is the difference between NumPy and SciPy?* [Online; navštíveno 05.05.2019].
URL
https://www.quora.com/What-is-the-difference-between-NumPy-and-SciPy

[29] *Data Mining with Python*. 2017.

[30] Breunig, M. M.; Kriegel, H. P.; Ng, R. T.; aj.: *LOF: Identifying Density-Based Local Outliers*. ACM SIGMOD, Dalles, 2000.

[31] CrowdFlower: *Data Science Report 2016*. [Online; navštíveno 07.05.2019].
URL https://visit.figure-eight.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf

[32] Galarnyk, M.: *Understanding Boxplots*. [Online; navštíveno 06.05.2019].
URL https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

[33] Han, J.; Kamber, M.; Pei, J.: *Data Mining: Concepts and Techniques. Third Edition*. Morgan Kaufmann Publishers, 2012, ISBN 978-0-12-381479-1.

[34] Liu, F. T.; Ting, K. M.: *Isolation-based Anomaly Detection*. ACM Transactions on Knowledge Discovery from Data (TKDD).

[35] Salian, I.: *SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?* [Online; navštíveno 07.05.2019].
URL https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/

[36] Torgo, L.: *Data Mining with R. Learning with Case Studies*. Chapman & Hall/CRC Press, 2011, ISBN 978-1-4398-1018-7.

[37] Zendulka, J.; Bartík, V.; Lukáš, R.; aj.: *Získávání znalostí z databází, Studijní opora*. Jaroslav Zendulka a kol., Brno 2006.

Příloha A

Obsah přiloženého paměťového média

stoika_thesis.pdf – Text bakalářské práce

jupyter – Složka obsahující zdrojové kódy a datové doubory.

readme.txt – Návod na spuštění.

LICENSE – Licence 3-Clause BSD License.