VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

DOLOVÁNÍ Z DAT V JAZYCE PYTHON

DATA MINING WITH PYTHON

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JAKUB ŠENOVSKÝ**
AUTHOR

VEDOUCÍ PRÁCE — **Doc. Ing. JAROSLAV ZENDULKA, CSc.**
SUPERVISOR

BRNO 2017

Zadání diplomové práce/20060/2016/xsenov02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Šenovský Jakub, Bc.**
Obor: Bezpečnost informačních technologií
Téma: **Dolování z dat v jazyce Python**
Data Mining with Python
Kategorie: Data mining

Pokyny:

1. Seznamte se s problematikou dolování z dat.
2. Seznamte se dostupnými prostředky na podporu dolování z dat v jazyce Python.
3. Seznamte se rámcově s jazykem R, dostupnými balíčky na podporu dolování z dat a případovými studiemi použitými v knize Data Mining with R (viz literatura).
4. Po dohodě s vedoucím práce vyberte minimálně dvě vhodné případové studie, na nichž budete ilustrovat použití jazyka Python pro dolování z dat.
5. Naprogramujte úlohy případových studií a dolování realizujte.
6. Zhodnoťte způsob dolování v jazyce Python a rámcově rovněž porovnejte s použitím jazyka R pro tytéž účely.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p.
- Vettigli, G.: Practical Data Mining with Python. Dostupné na http://guidetodatamining.com/.
- Torgo, L.: Data Mining with R. Learning with Case Studies. Chapman & Hall/CRC Press. 2011. 289 p.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 4.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zendulka Jaroslav, doc. Ing., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem této práce bylo seznámení se s jednotlivými fázemi získávání znalostí z dat, s podporou programovacích jazyků Python a R v oblasti dolování dat a demonstrace jejich použití na dvou případových studiích. Následným krokem bylo porovnání těchto jazyků z hlediska dolování dat. Je zde popsaná fáze předzpracování dat a dolovací algoritmy pro klasifikaci, predikci a shlukování. Představeny zde byly významné knihovny pro jazyky Python a R. V první případové studii byla demonstrována práce s časovými řadami pomocí ARIMA modelu a neuronových sítí s ověřením přesnosti pomocí střední kvadratické chyby. V druhé případové studii byla popsaná klasifikace výsledků fotbalových zápasů pomocí k – nejbližších sousedů, Bayesova klasifikátoru, náhodného lesu a logické regrese. Přesnost klasifikace byla zobrazena pomocí skóre přesnosti a konfúzní matice. Práci uzavírá zhodnocení výsledků a návrhy pro budoucí vylepšení jednotlivých modelů.

Abstract

The main goal of this thesis was to get acquainted with the phases of data mining, with the support of the programming languages Python and R in the field of data mining and demonstration of their use in two case studies. The comparison of these languages in the field of data mining is also included. The data preprocessing phase and the mining algorithms for classification, prediction and clustering are described here. There are illustrated the most significant libraries for Python and R. In the first case study, work with time series was demonstrated using the ARIMA model and Neural Networks with precision verification using a Mean Square Error. In the second case study, the results of football matches are classificated using the K – Nearest Neighbors, Bayes Classifier, Random Forest and Logical Regression. The precision of the classification is displayed using Accuracy Score and Confusion Matrix. The work is concluded with the evaluation of the achived results and suggestions for the future improvement of the individual models.

Klíčová slova

Získávání znalostí, Python, R, předzpracování dat, shlukování, predikce, klasifikace, případové studie, ARIMA, porovnání Python a R, Bayes, knn.

Keywords

Data mining, Python, R, Data Preprocessing, Clustering, Prediction, Classification, Case studies, ARIMA, comparison Python and R, Bayes, knn.

Citace

ŠENOVSKÝ, Jakub. *Dolování z dat v jazyce Python.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Zendulka Jaroslav.

Dolování z dat v jazyce Python

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. Ing. Jaroslava Zendulky, Csc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Šenovský
22. května 2017

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu Doc. Ing. Jaroslavovi Zendulkovi, Csc za odborné vedení a rady, které mi při řešení práce poskytl. Dále bych chtěl poděkovat rodině za podporu při studiu.

Obsah

Kapitola 1

Úvod

Jak lidé, tak i společnosti, státy a nejrůznější organizace, generují v dnešní době velké množství dat. Tato data se nejčastěji vážou ke konkrétním subjektům, např. u lidí se jedná o data návštěvností webových stránek a sociálních sítí, historie nákupů konkrétních položek či zeměpisnou polohu jejich pohybu. U společností se jedná o hodnoty jejich prodeje, položky výroby, průběh zisků či ztrát. Státy a organizace poskytují data o počtu obyvatel v jednotlivých krajích, hladině řek a vodních ploch, počtu automobilů, jejich stáří a další nejrůznější statistiky. Jak je možné postřehnout, tato data se od sebe hodně liší. Avšak při jejich správné kombinaci se dají využít k získání netriviálních znalostí, které na první pohled nejsou zřejmé. Tyto znalosti se používají hlavně v komerční sféře ke zlepšení výsledků prodeje produktů a služeb, k lepšímu cílení marketingu nebo k predikci hodnot daných komodit na trhu. [10]

Mezi základní fáze získávání znalostí patří shromáždění dat, jejich předzpracování, transformace a nakonec samotné získávání znalostí. Avšak reálná data velmi často obsahují chyby v podobě chybějících, zašuměných či nesprávných hodnot.

Kapitola 2 popisuje předzpracování těchto dat tak, aby byla vhodná pro získávání znalostí. Jsou zde představeny techniky pro získání přehledu o hodnotách dat, metody pro odstranění chybných hodnot, jejich transformace a taktéž metoda pro výběr relevantních atributů dat zvaná redukce. Důležité je zmínit, že pod pojmem "data", resp. "datovým souborem", se rozumí data logicky uspořádaná do tabulky, jejíž sloupce představují hodnoty atributů a řádky jednotlivé záznamy.

Kapitola 3 představuje nejdůležitější třídy algoritmů pro dolování dat. První třídou je *klasifikace*, která klasifikuje kategorické hodnoty na základě ostatních numerických hodnot atributů. Jedná se např. o Bayesovský klasifikátor nebo techniku k – nejbližších sousedů. Pro predikování budoucích numerických hodnot jsou využívány algoritmy ze třídy *predikce*, konkrétně techniky regresní analýzy. Třída *Shlukové analýzy* je využívána pro slučování záznamů dat s podobnými vlastnostmi do různých skupin.

Kapitola 4 popisuje významné knihovny dvou nejvíce používaných programovacích jazyků pro dolování dat a to jazyků Python a R. Jedná se o knihovny Pandas, Scikit – learn, statsmodel, resp. v jazyce R o knihovny caret, forecast. Tyto knihovny byly výhradně použité při realizace případových studií.

Podrobný popis realizace těchto studií je možné nalézt v kapitole 5. Cílem případových studií bylo v souladu se zadáním ilustrovat použití prostředků obou jazyků a knihoven, ne nutně provedení rozsáhlých experimentů pro nalezení a vytvoření nejvhodnějších atributů a nastavení hodnot parametrů. První případová studie je z oblasti energetiky a představuje práci s časovými řadami. Pro predikci budoucích hodnot těchto časových řad je použit

ARIMA model a také neuronová síť. Druhá případová studie pracuje s fotbalovými statistikami ve snaze predikovat výsledný stav zápasu za pomocí různých technik, kterými je *náhodný les, Bayesovská klasifikace, K – nejbližšísch sousedů a logické regrese*. U každé případové studie jsou výsledné modely a jejich přesnost zhodnoceny podle dostupných metrik.

V poslední kapitole jsou porovnány dostupné prostředky jazyků Python a R podle jednotlivých fází dolování dat. Taktéž je zde popsána podpora paralelizace a možné propojení obou jazyků.

V závěru práce jsou zhodnoceny dosažené výsledky a taktéž nastíněny možnosti rozšíření práce v budoucnosti.

Kapitola 2

Předzpracování dat

Originální data, na která se budou aplikovat algoritmy pro získání znalostí, nemusí být ideální. Mohou být zašuměná, nekonzistentní, některé hodnoty můžou chybět nebo můžou být obsaženy vícekrát v různých datových typech. Pokud by se pracovalo s těmito daty, byly by výsledky dolování málo kvalitní. Tudíž je nutné provést předzpracování dat.

Předzpracování dat se skládá z několika typů úloh. První úlohou je *čištění dat*, kdy jsou odstraněna zašuměná a nekorektní data. Pokud jsou data získána z více zdrojů, je nutné tyto zdroje sloučit do datového skladiště. Tato úloha se nazývá *integrace dat*. Další úlohou je *transformace dat*, kde dochází k sjednocení tvarů atributů, např. datum. Pro lepší výsledky dolování je možné převést (normalizovat) číselná data do daného intervalu. Poslední úlohou je *redukce*, kdy jsou z velkého množství dat vybrány pouze relevantní data.

2.1 Sumarizující popisné charakteristiky dat

Aby bylo možné správně provést jednotlivé fáze předzpracování, je nutné mít celkový přehled o datech. Techniky sumarizujícího popisu slouží k určení jejich typických vlastností a mohou být použity pro určení zašumělých nebo odlehlých hodnot. Nezbytnou nutností je mít přehled o míře polohy ("středu") dat a jejich rozptylu. [5]

Míry polohy

Mezi nejznámější metody určení střední polohy patří *aritmetický průměr*. Vypočítá se jako suma všech hodnot vydělený jejich počtem. Pokud má každá hodnota přiřazenou váhu, je nutné tyto váhy brát v potaz. Pro tento případ je možné použít pro výpočet *vážený aritmetický průměr*.

Průměr není nejlepší způsob, jak zjistit "střed" dat, jelikož je ovlivněn odlehlými hodnotami – jak velmi malými, tak i velmi vysokými. Metoda, která tyto extrémní hodnoty ignoruje, se nazývá *ořezaný průměr*. V procentech je vyjádřeno, jak moc extrémních hodnot z minima a maxima bude ignorováno. Nevýhodou této metody je možnost ztráty důležitých informací.

Pro lepší zjištění střední hodnoty asymetrických dat je také možné použít *medián* nebo *modus*. Pokud je počet hodnot uspořádaného seznamu lichý, je medián střední hodnota, jinak je to průměr dvou středních hodnot. *Modus* je vybrán jako hodnota s nejčastějším výskytem. Pokud se jedná o jedinou hodnotu, jedná se o unimodální data, pokud o dvě, tak bimodální a pokud více, tak multimodálná. Pro unimodální mírně vychýlená data existuje následující vztah pro odhad mediánu $prumer - modus = 3 * (prumer - median)$. Jde

o holistickou míru, nelze tedy medián spočítat přesně nějakým distribuovaným výpočtem. V neposlední řadě může být použit střed rozsahu vypočítaný jako průměr největší a nejmenší hodnoty.

Míry atributů je možné rozdělit na tyto části:

1. **Distributivní míra** - mějme množinu dat, která je rozdělena na menší podmnožiny dat. U každé podmnožiny je vypočítána míra atributů. Výsledná míra pro celou množinu dat vznikne sloučením jednotlivých výsledků podmnožin.

2. **Algebraická míra** - míra, jež je výsledkem aplikování algebraické funkce na jednu nebo více distributivních měr.

3. **Holistická míra** - míra, jež musí být vypočítána na celém setu dat. V porovnání s distributivní mírou je na výpočet dražší. Patří sem například medián.

Míry variace

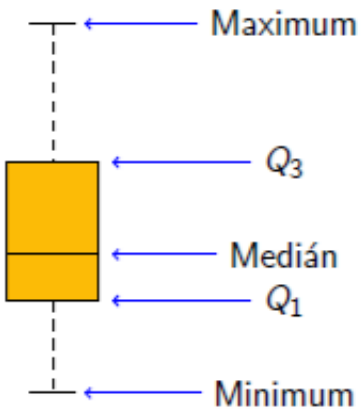
Míry variace vypovídají o rozptylu hodnot datového souboru. Patří sem *rozsah (maximální hodnota - minimální hodnota)*, *pěti číselná sumarizace založená na kvartilech*, *mezikvartilová vzdálenost* a *standardní odchylka.*

Kvartily, včetně mediánu, slouží k získání informací o rozptylu, středu a rozložení dat. Mějme taková data, která jsou vzestupně seřazená a rozdělená na čtyři intervaly o stejném počtu prvků. V takto uspořádaných datech lze nalézt tři hraniční kvartily $Q1$ až $Q3$. Kvartil $Q1$ je roven 25 percentilům, $Q2$ je medián, neboli 50 percentil a $Q3$ je 75 percentil. Potom výpočtem

$$IQR = Q3 - Q1$$

získáme interval obsahující prostřední polovinu dat nazývaný *Interkvartilní rozsah (IQR)*. Kvartily $Q1$, $Q2$ a IQR patří do skupiny holistických měr.

Samotný IQR nevypovídá nic o vychýlení dat, proto se kombinuje s kvartily $Q1$ a $Q3$, mediánem, minimem, maximem a nazývá se pěti číselná sumarizace. Existuje pravidlo, že pokud hodnoty nespadají do intervalu $1.5 * IQR$ nad $Q3$ nebo pod $Q1$, tak jsou označeny jako odlehlé. Pěti číselná sumarizace je často zobrazována krabicovým grafem. [9]



Obrázek 2.1: Ukázka krabicového grafu zobrazující nejdůležitější hodnoty[1]

[1] http://novak.blog.respekt.cz/k-cemu-je-nam-krabicovy-graf-karmicke-zkoumani/

Rozptyl a standardní odchylka

Rozptyl je možné definovat jako střední hodnotu druhých mocnin odchylek od střední hodnoty.

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

Směrodatná odchylka, σ, se následně vypočítá jako druhá odmocnina z rozptylu σ^2 a má tyto vlastnosti:

- ukazuje rozložení okolo průměru a měla by být použita pouze tehdy, pokud je průměr použitý jako míra středu
- pokud je $\sigma = 0$ znamená to, že zde není žádné rozložení a nastává tehdy, pokud zkoumané vlastnosti mají stejnou hodnotu

Rozptyl i směrodatná odchylka patří mezi algebraické míry. [1]

2.2 Čištění dat

Jak už bylo zmíněno výše, reálná data mohou být zarušená nebo nekompletní. Proces zvaný čištění dat se snaží s těmito problémy vypořádat aplikováním nejrůznějších metod. Cílem je vyprodukovat vhodná data pro dolování. Nyní budou popsány situace vyžadující čištění.

Chybějící hodnoty

S chybějícími hodnotami se lze vypořádat následujícími způsoby.

1. **Ignorovat záznam** - nejčastěji při klasifikaci a to tehdy, pokud chybí hodnota atributu určujícího třídu. Jsou efektivní pouze tehdy, když chybí hodnoty i v ostatních atributech.
2. **Manuální doplnění** - nevhodná u velkého množství dat.
3. **Nahrazení globální konstantou** - nahradíme všechny chybějící hodnoty konstantou, např. *null*. Pokud je mnoho chybějících hodnot nahrazeno konstantou, program se může mylně domnívat, že jde o společnou zajímavou hodnotu. I když je metoda jednoduchá, není moc spolehlivá.
4. **Použít průměr z ostatních hodnot atributů** - vypočítáme průměr z ostatních vyplněných hodnot a nahradíme jim chybějící hodnoty.
5. **Použít průměr hodnot atributu patřících do stejné třídy**
6. **Nejpravděpodobnější hodnota** - na základě ostatních hodnot v záznamu se najde za pomocí regrese, Bayesových formulí nebo rozhodovacího stromu, nejpravděpodobnější podobná hodnota a tou je vyplněna chybějící hodnota. Tato strategie je velmi populární.

Jelikož vyplňujeme chybějící data hodnotami nezávislými na hodnotách ostatních atributů, dochází zde k ovlivňování výsledků. U poslední metody se používají ostatní atributy k výpočtu chybějící hodnoty. Je pravděpodobné, že ostatní atributy jsou s touto hodnotou ve vztahu a tudíž poslední metoda může být označena jako nejlepší náhrada. [8]

Zašuměná data

Zašuměním dat rozumíme náhodné chyby nebo odchylky v naměřených datech. Zašuměné data můžeme eliminovat následujícími technikami:

1. **Rozdělení do košů (binning)** - tato technika vyhlazuje seřazená data na základě jejich sousedů a provádí tzv. lokální vyhlazování. Seřazené hodnoty jsou rozděleny do jednotlivých košů se stejnou frekvencí hodnot (např. každý koš obsahuje 5 hodnot). Následně je možné provést vyhlazování pomocí průměru koše (smoothing by bin mean) a znamená to, že každá hodnota v koši je nahrazena průměrnou hodnotou koše. Podobnou technikou je vyhlazování pomocí mediánu koše, kdy jsou hodnoty v koši nahrazeny mediánem. Vyhlazení pomocí hranic koše se provede tak, že se nalezne minimum a maximum koše a každou hodnotu nahradíme minimem/maximem podle toho, ke které hranici má blíže.[13]

2. **Regrese** - data můžou být nahrazená výsledkem nějaké funkce. U lineární regrese se hledá nejlepší shoda mezi dvěma atributy, kdy první atribut může být použit pro predikci druhého. Vícenásobná lineární regrese používá k vyhlazení více atributů.

3. **Shlukování** - extrémní hodnoty mohou být detekovány na základě shlukování. Stejné hodnoty jsou organizovány v shluku, kdežto extrémní hodnoty nepatří do žádného.

Mnoho metod pro vyhlazování může být použito i pro redukci dat, např. rozdělování do košů redukuje počet různých hodnot atributu.

Čistění dat jako proces

Čištění dat jako proces se skládá ze dvou částí. První částí je *detekce nesrovnalostí*, kdy nesrovnalosti mohu být způsobeny různými faktory jako jsou špatně navržené formuláře, lidskými chybami v záznamech, neaktuálními daty nebo úmyslným nevyplněním hodnot. Dalším zdrojem nesrovnalostí můžou být poruchy zaznamenávajícího zařízení nebo díky systémovým chybám. K chybě může dojít taktéž při integraci dat z různých zdrojů. Pro získání informací o vlastnostech dat a pro vyřešení nesrovnalostí jsou důležitá metadata, která obsahují datový typ, povolené hodnoty, jejich rozsah. Nekonzistence může také nastat u formátu atributu, např. datum. Dále zde může být kladen důraz na unikátnost dat, kdy každá hodnota musí být různá od další – unikátní pravidlo, žádná hodnota nesmí chybět – souvislé pravidlo, nebo může být prázdná – null pravidlo. Druhou částí je *odstranění nesrovnalostí*. Některé nesrovnalosti mohou být odstraněny ručně. Avšak odstranění chyb vyžaduje většinou transformaci dat. Existují komerční nástroje pro migraci dat, které dokážou provést jednoduchou transformaci.

Tyto procesy probíhají v iteracích, které jsou náchylné k chybám a zabírají určitý čas. Je nutné kontrolovat, zda v iteracích nebyly zaneseny nové nesrovnalosti. [5]

2.3 Integrace a transformace dat

Data mohou často pocházet z různých zdrojů jako jsou databáze, webové stránky či samotné soubory. Tato různá data je proto nutné sloučit či integrovat do jednotného celku, který je následně možný transformovat do stavu vhodného pro získávání znalostí. Nyní budou popsané jednotlivé části zvlášť.

Integrace dat

Jak bylo popsáno výše, při integraci se slučují data z více zdrojů. Mohou zde nastat následující problémy.

1. **Konflikty schémat a shoda objektů** - tento konflikt popisuje neshodu v pojmenování metadat jednoho zdroje s metadaty druhého zdroje. Např. sloupec customer_id se může jmenovat ve druhém zdroji cust_id.

2. **Konflikty a rozlišení hodnot** - hodnoty z různých zdrojů mohou nabývat různých rozsahů či jednotek. Např. cena může být uložená v odlišných měnách nebo váha v různých jednotkách. Taktéž je důležité dbát na strukturu dat, kdy např. sleva může být aplikovaná jednou na celou objednávku, po druhé na jednotlivé položky.

3. **Redundantní data** - pokud jde jeden atribut odvodit z jiného atributu, jedná se o redundanci. Redundance může být způsobena nekonzistentním pojmenováním atributů a jejich uložením do výsledného souboru dat.

Pro detekci redundance je možné použít korelační analýzu. Korelační analýza dvou atributů ukazuje, jak silně jeden atribut implikuje druhý. Pro numerická data lze vypočítat Pearsonův korelační koeficient r, kde x a y jsou jednotlivé hodnoty atributů, $\bar{x}$, resp. $\bar{y}$ je průměr hodnot atributu x, resp. y a n je jejich celkový počet, následovně:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Výsledná hodnota je v intervalu $-1 <= R_{A,B} <= 1$. Pokud je hodnota $R_{A,B}$ větší než 0, potom A a B jsou pozitivně korelované a hodnota A se zvětšuje s hodnotou B. Čím větší hodnota, tím silnější korelace a tím více je implikován druhý atribut. Proto je možné jeden atribut odstranit z důvodu redundance. Pokud je hodnota menší než 0, potom jsou atributy negativně v korelaci – hodnota atributu A se zvětšuje, kdežto hodnota atributu B se snižuje. Důležitým faktem je, že korelace neimplikuje kauzalitu. Mezi kategorickými diskrétními daty je možné nalézt korelační vztah pomocí χ^2 (chi-square) testu. [16]

Transformace dat

Při transformaci jsou data transformována do podoby vhodné pro dolování. Provádí se pomocí následujících operací.

- **Vyhlazení**
- **Agregace** - agregační operace jsou aplikovány na data.
- **Generalizace** - primitivní data jsou nahrazena vyšší abstrakcí, např. ulice je nahrazena městem nebo státem.
- **Normalizace** - data jsou mapována do specifického rozsahu např. $< -1; 1 >$.
- **Tvoření atributů** - nové atributy jsou tvořeny na základě již známých atributů

Normalizace zaručuje, že všechna číselná data budou v daném rozsahu. Je důležitá u algoritmů pro klasifikaci, pro metody založené na vzdálenostech, např. k – nejbližších sousedů,

a nebo pro shlukování (clustering). Pro normalizaci je možné použít jednu z následující metod. První metodou je *min – max* normalizace. Lineární transformace je aplikována nad originálními daty a to tak, že dochází ke změně původního rozsahu hodnot $min - max$ na nový rozsah $min_{new} - max_{new}$. Výhodou je udržení poměru mezi původními hodnotami dat. Další metodou je *z – skóre* normalizace. Hodnoty atributů jsou normalizovány na základě průměru a standardní odchylky. Je vhodná tehdy, když je rozsah atributu neznámý, popř. když obsahuje extrémy, které dominují min – max normalizaci. Další metodou je *normalizace podle desetinného škálování*, kdy dochází k pohybu desetinné čárky atributu. O kolik se hodnota posune záleží na maximální absolutní hodnotě a požadovaném rozsahu, typicky $< -1, 1 >$. [7]

2.4 Redukce dat

Při redukci se vybírají z velkého množství dat menší podmnožiny a to tak, aby výsledek při dolování z jednoho či z druhého byl skoro stejný. Je možné aplikovat následující techniky.

První technikou je *agregace datové kostky*, kdy dochází k aplikování agregačních operací při konstrukci datové kostky, např. získání sumy ročního prodeje.

Součástí dat jsou i nerelevantní nebo málo potřebná data. Při *výběru podmnožiny atributů* je snaha tato nepotřebná data ignorovat. Cílem je nalezení minima atributů, na kterých bude výsledek při dolování stejný jako na úplných datech. Jelikož pro n atributů je možné najít 2^n podmnožin, výpočet nejlepší podmnožiny může být neúnosně drahý. Proto se používají heuristické metody používající lokální optimalizace, které vždy vyberou nejlepší jen "lokálně", obecně ne globálně nejlepší, možnost. Metoda *postupného výběru vpřed* obsahuje na začátku prázdnou redukovanou množinu. Při iteracích jsou ze zbývajících atributů nalezeny vždy ty nejlepší a přidány do redukované množiny. Opakem je metoda *postupného výběru vzad*, kdy redukovaná množina obsahuje všechny atributy a v každé iteraci jsou odstraněny ty nejhorší. Kombinací postupného výběru vpřed a vzad jsou v každém kroku vybrány ty nejlepší a odstraněny ty nejhorší.

Při technice nazývané *redukce dimensionality* jsou data zakódována za účelem snížení velikosti. Mohou být ztrátové i bezeztrátové. Mezi populární metody patří vlnková transformace nebo analýza hlavních komponent (PCA).

Metodou zvanou *redukce počtu hodnot* jsou data nahrazena jinou datovou reprezentací. Používají se především parametrické modely a neparametrické metody jako je shlukování, vzorkování nebo histogram. Mezi parametrické techniky patří regrese a log – lineární modely, kdy při lineární regresi jsou data reprezentována přímkou. Log – lineární modely se používají pro aproximaci mnohorozměrných diskrétních distributivních pravděpodobností dat. Obě metody mohou být použity na řídká či zkreslená data. K populárním neparametrickým technikám patří histogram, který pro redukci atributů používá rozdělení do intervalů nebo podle stejné frekvence hodnot. Pro rozdělování hodnot do intervalů jsou použity metody stejné šířky či frekvence, dále metody v – optimální nebo maximální rozdíl dvojic hodnot, které jsou více praktické a přesnější. Při technice shlukování je vybrána ze shluku dat pouze jedna hodnota, která reprezentuje daný shluk. Výběr je proveden na základě maximální vzdálenosti mezi náhodnými dvěmi atributy nebo na základě těžiště shluku. Poslední neparametrickou metodou je vzorkování, kdy je z původních dat vybrán atribut reprezentující danou skupinu dat. Mezi techniky výběru vzorků dat patří metoda jednoduchého výběru atributu bez návratu nebo s návratem, dále výběr fyzických shluků hodnot nebo stratifikovaný vzorek. [5]

Kapitola 3

Metody pro získávání znalostí

Existuje mnoho druhů algoritmů, které lze použít na získání znalostí z dat. Pro použití konkrétního algoritmu je důležité, na jaké otázky by měl získat odpověď. Pokud chceme rozdělit data do skupin či najít hodnoty, které jsou netypické pro danou skupinu, je vhodné použít algoritmy pro *shlukování*. Pokud bychom chtěli odhadnout náhodnou veličinu v budoucnosti na základě již známých hodnot, je vhodné použít algoritmy ze třídy označených pro *regresi* nebo pro *klasifikaci*. Pro získání zajímavých, užitečných a neočekávaných vzorů v datech jako je asociace, podgrafy, trendy a další, je možné použít techniku zvanou *frekvenční vzory*. Na základě těchto vzorů je možné vytvořit *asociační pravidla*. Nyní bude představena každá skupina algoritmů a popsány její nejdůležitější fáze.

3.1 Klasifikace a predikce

Jak již bylo zmíněno výše, algoritmy z této skupiny slouží pro získání modelů popisujících důležitá data nebo pro predikci budoucích trendů. Jedná se o dvě hlavní metody pro předvídání hodnot. Konkrétně metody klasifikace jsou používány k předpovědi kategorických hodnot, např. zda je vhodné zákazníkovi poskytnout půjčku či nikoliv. Pod pojmem predikce si lze představit funkci pro předpovídání budoucí hodnoty v návaznosti na předchozích.

Klasifikace se nejčastěji skládá ze dvou kroků. První částí je krok *učení*, kde si klasifikační algoritmus vytváří klasifikátor na základě trénovací množiny. Neboli dochází k vytvoření mapovací funkce, která dokáže na základě jedné či více hodnot atributů dat odvodit výslednou hodnotu y daného atributu. Trénovací množinou jsou myšlena data, která jsou již přiřazena do správných kategorií. Nyní je důležité zmínit, že existuji dva typy učení. Jednak je to učení s učitelem, kdy jsou názvy kategorií algoritmu poskytnuty, jinak je to učení bez učitele.

Druhou částí je samotná *klasifikace* nových dat na základě již naučených hodnot. Výsledkem je kategorická hodnota, např. zákazníkovi je možné poskytnou půjčku.

Predikce je nejčastěji využívána pro číselné atributy a skládá se taktéž ze dvou kroků. Prvním krokem je *učení*, kdy je predikátoru předložena numerická posloupnost hodnot. Výsledkem je předpovězení následné numerické hodnoty posloupnosti.

Klasifikaci a predikci je možné porovnávat na základě několika kritérií. Prvním kritériem je přesnost klasifikátoru a udává, jak přesně jsou předpovězená budoucí data. Udává se v procentech a vyjadřuje počet testovacích záznamů, které byly klasifikátorem správně klasifikovány. Dalšími kritériemi je škálovatelnost, rychlost či odolnost proti zašuměným datům. Nyní budou představeny některé algoritmy pro klasifikaci a predikci.

Bayesovská klasifikace

Jedná se o statistický klasifikátor, který vypočítá pravděpodobnost, jak moc daná hodnota patří do dané kategorie či nikoliv. Je založená na Bayesovském teorému popsaném níže. Bayesovský klasifikátor je často označován jako naivní, jelikož předpokládá, že jednotlivé atributy nejsou ve vztahu s ostatními. Bayesovský klasifikátor je velmi přesný i rychlý při aplikování na velkých datech.

Bayesův teorém se zabývá podmíněnými pravděpodobnostmi. Předpokládejme existenci atributu A a dále hypotézy H takové, že atribut X patří do třídy T. Zápisem $P(A|H)$ je značena podmíněná pravděpodobnost atributu A patřící do třídy T za správného předpokladu hypotézy H. Vzorec pro výpočet Baesova teorému je následující:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

Tyto vzorce jsou využívány naivním Bayesovským klasifikátorem. Předpokládejme, že D značí množinu všech záznamů dat. Dále uvažujme třídy $C_1, C_2, \ldots, C_m$, kde m je celkový počet těchto tříd, do kterých budou záznamy z množiny D klasifikovány. Pak $P(T_i|X)$ udává podmíněnou pravděpodobnost, že libovolný záznam X padne do třídy C_i. Tedy je nutné najít třídu C_i s největší pravděpodobností pro daný záznam. Potom platí:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

Jelikož je P(X) pro všechny třídy konstantou, je nutné najít největší hodnotu výrazu $P(X|C_i)P(C_i)$. Hodnotu $P(C_i)$ je možné vypočítat jako $P(C_i) = |C_{i,D}|/|D|$, kde $|D|$ je počet hodnot obsažených v dané množině a $|C_{i,D}|$ je počet trénovacích záznamů třídy C_i v množině D. Hodnota $P(X|C_i)$ označuje pravděpodobnost, že libovolný prvek ze třídy C_i bude mít stejné hodnoty jako prvek X, vypočítáme pomocí tohoto vzorce

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i),$$

kde hodnota x_k odkazuje na atribut A_k vzorku X. Pokud je typ atributu A_k kategorický, je hodnota $P(X|C_i) = d_{ik}/d_i$, kde d_{ik} je počet vzorků množiny D zařazené do třídy C_i, jejichž atribut A_k má hodnotu x_k. Druhou možností je spojitá hodnota, která má velmi často Gaussovo normální rozložení

$$g(\chi, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(\chi-\mu)^2}{2\sigma^2}}$$

pro atribut A_k s hodnotou x_k, průměrem μ a směrodatnou odchylkou σ pro atributy A_k třídy C_i.[16]

K – nejbližších sousedů

Metoda k – nejbližších sousedů je založena na porovnávání podobnosti trénovacích a testovacích vzorků, kde samotný vzorek[1] je reprezentován n číselnými atributy se stejnou váhou. Tedy každý vzorek si lze představit jako bod v n – dimenzionálním prostoru. Při klasifikaci neznámého prvku je potom vybráno k nejbližších vzorků z trénovacích dat a neznámý prvek

[1] Vzorkem se rozumí záznam ze souboru dat.

je zařazen do třídy s nejvíce výskyty u k vybraných vzorů. Nejbližší vzdálenost mezi dvěma prvky $X = (x_1, x_2, \ldots, x_n)$ a $Y = (y_1, y_2, \ldots, y_n)$ je vypočítána pomocí tzv. Euklidovské vzdálenosti následovně $d(X, Y) = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 + \cdots + |x_n - y_n|^2}$.

Pokud se nejedná o číselné hodnoty, ale kategorické, jsou vzdálenosti získány různými způsoby. Nejjednodušší metodou je označit stejnou kategorickou hodnotu vzdáleností 0, jinak vzdáleností 1. [14]

Regresní analýza

Pro predikci spojité číselné hodnoty je taktéž možné použit regresní analýzu. Regresní analýza se používá pro modelování vztahů jedné či více nezávislých proměnných při předpovídání budoucí spojité číselné hodnoty.

Mezi nejznámější regresní metodu patří *lineární regrese*. Je vyjádřena lineární funkcí (přímkou), kdy na základě již známé hodnoty proměnné x a regresních koeficientů w_0 a w_1 je vypočítána výsledná hodnota y. Neboli $y = w_0 + w_1 x$. Regresní koeficienty se vypočítají metodou nejmenších čtverců, která minimalizuje chybovost mezi daty a lineární funkcí. Předpokládejme dataset D s atributy x a y. Metodou nejmenších čtverců vypočítáme koeficient w_1:

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2}$$

a koeficient w_0:

$$w_0 = \bar{y} - w_1 \bar{x},$$

kde $\bar{x}$, resp. $\bar{y}$ je průměr hodnot atributu x, resp. atributu y.

Pro predikci výsledné hodnoty za použití více jak jedné proměnné lze použít *multi – lineární regresi*, která rozšiřuje lineární regresi. Vzorec pro lineární regresi je upraven tak, že jsou použité hodnoty atributů následovně $y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$. Regresní koeficienty se taktéž vypočítají metodou nejmenších čtverců.

Na predikci atributů, které nevykazují lineární závislost, lze použít nelineární regresní analýzu jako je *polynomiální regresní analýzu*. Tuto metodu lze snadno transformovat na lineární analýzu a taktéž použít metodu nejmenších čtverců.

Mezi další metody založené na regresi patří *generalizované lineární modely* obsahující logické a Poissonovy regresní metody. Pro diskrétní multidimensionální rozložení pravděpodobnosti jsou často používány *log – lineární modely*. [15]

Rozhodovací strom

Jedná se o funkci, která na vstupu přijímá vektor atributů a vrací jednu z možných výstupních hodnot, tzv. rozhodnutí. Získání výsledků je dosaženo díky sekvenci testů. Každý vnitřní uzel reprezentuje podmínku aplikovatelnou na atribut, každá větev označuje výsledek rozhodnutí a v každém listu stromu je uložena výsledná klasifikační třída. Cesta od kořene až k výslednému listu stromu reprezentuje klasifikační pravidla, která byla aplikována. Některé implementace tohoto algoritmu mohou vytvářet pouze binární stromy, zatímco ostatní mohou produkovat i stromy nebinární.

Náhodný les (Random Forest)

Jedná se o soubor učících metod pro techniky klasifikace a regrese. Při operaci učení dochází k vytváření mnoha rozhodovacích stromů, které následně, nezávisle na sobě, určují výsledek

testování. Pokud se jedná o techniku klasifikace, je výsledkem náhodného lesa nejčastější hodnota tříd získaná jednotlivými stromy. Při technice regrese je výsledná hodnota získána zprůměrováním výsledků z jednotlivých stromů. Výhodou této techniky je, že řeší problém přeučení při vytváření a trénování rozhodovacích stromů. [3]

Přesnost a míra chybovosti klasifikátoru

Jednou z důležitých otázek je přesnost vytvořeného predikátoru nebo klasifikátoru. Důležitou podmínkou testování klasifikátoru je použití dat, která nebyla obsažena v datech při učení klasifikátoru. Proto jsou data náhodně rozdělena do skupin, kdy jedna skupina je použita pro testování (1/3) a zbytek pro učení (2/3). Přesnost je pak získána podle počtu správně zařazených testovacích hodnot do správné kategorie. Používá se tzv. konfúzní matice (confusion matrix), která se skládá z m řádků a m sloupců. V názvu řádků a sloupců jsou uvedeny kategorie, do kterých byly testovací hodnoty klasifikovány. Jednotlivé buňky obsahují počet hodnot, které reálně patří do dané třídy a byly klasifikátorem přiřazeny do stejné nebo různé třídy. Výpočtem tedy získáme procentní hodnotu určující přesnost klasifikátoru pro danou třídu. [4]

3.2 Shluková analýza

Shluková analýza se nejčastěji používá právě tehdy, pokud nejsou známy kategorie zpracovávaných dat. Cílem shlukování je rozdělit data do skupin ("shluků") na základě podobných atributů či vlastností. Dalším způsobem využití shlukování je detekce anomálii v datech, kdy mimo ležící data nepatří do žádné skupiny. Nejčastěji je shluková analýza založená na měření vzdáleností mezi jednotlivými vzorky, viz. metody níže. Krok učení je založený na pozorování, nikoliv na učení příklady. Shlukování je často využíváno v prozkoumávání nových možností trhu, rozpoznávání vzorů nebo analýze dat. Mezi nejčastější vlastnosti shlukovacích algoritmů patří následující:

- **Škálovatelnost** - shlukovací algoritmus by měl dobře pracovat jak s malým, tak i s velkým množství dat.
- **Schopnost pracovat s různými typy dat** - data mohou obsahovat číselné, binární, kategorické, ordinální nebo různé hodnoty dat. Shlukovací algoritmus by měl správně pracovat s různými typy.
- **Shluky s nahodilým tvarem** - shlukovací algoritmy založené na vzdálenostech vytvářejí shluky na základě podobné vzdálenosti. Avšak shluk může být různého tvaru s různými vzdálenostmi.
- **Minimální požadavky na vstup uživatele** - snaha minimalizovat interakci s uživatelem, např. na zadání počtu skupin.
- **Schopnost pracovat se zašuměnými daty** - data mohou mít nízkou kvalitu, chybějící nebo neznámé hodnoty.
- **Inkrementální shlukování a ignorace řazení dat** - algoritmus by měl umět začlenit nová data bez nutnosti vytvářet skupiny od začátku. Taktéž řazení vstupních dat by nemělo ovlivňovat jeho výpočet.

- **Podmíněné shlukování** - na základě daných podmínek by měly být vytvořeny shluky splňující tyto podmínky

Nyní budou představeny jednotlivé kategorie algoritmů pro shlukování a jejich nejznámější představitelé.[5]

Metody založené na rozdělování

Tyto metody rozdělí n vzorků dat do k skupin, kde $k \leq n$. Zároveň každá skupina musí obsahovat alespoň jeden vzorek a každý vzorek musí patřit pouze do jedné skupiny.

V první fázi se vybere tolik vzorků, kolik je zadaných jednotlivých skupin a ostatní vzorky jsou přiřazeny do skupin na základě jejich podobností. V dalších krocích dochází k reorganizaci vzorků až do té doby, dokud nejsou vzorky ve skupině maximálně podobné. Pro správné rozdělení vzorků do skupin se využívají různé heuristiky.

Nejznámější rozdělovací metodou je *metoda založená na centrálním bodu (k – means)*. U této metody je centrální bod jednotlivých skupin vypočítán jako průměrná hodnota atributů vzorků dané skupiny. Další vzorky jsou rozděleny do skupin na základě nejmenších vzdáleností od centrálních bodů skupin. Následně je vypočítán nový centrální bod pro každou skupinu. Přerozdělování končí, jakmile jsou dosaženy podmínky pro zastavení, např. změna v shlucích je menší než práh, maximální počet iterací, neměnný centrální bod nebo konvergenční funkce dosáhne určitého prahu. Nejčastěji se jedná o funkci čtvercové chyby

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2,$$

kde E je suma čtverců chyb pro všechny objekty v datech, p je daný objekt, m_i je centrální bod skupiny C_i. Tato metoda je snadno škálovatelná a dobře použitelná při zpracování velkých dat. Avšak může být aplikovaná pouze tehdy, pokud je definován centrální bod skupiny, což není dobře proveditelné u kategorických dat. Nevýhodou jsou také zašuměná a odlehlá data, která mohou negativně ovlivňovat výpočet centrálního bodu.

Další významnou metodou je *metoda založená na reprezentujícím objektu (k – medoids)*. U této metody se jako centrální body skupin vyberou konkrétní objekty z dat. Metoda funguje podobně jako k – means s tím rozdílem, že se nepočítá vždy nový fiktivní centrální bod, ale jako nový centrální bod se vybere konkrétní objekt. Snahou je vybrat objekt s polohou blízko středu skupiny. Prvním algoritmem, který tuto techniku využívá, byl algoritmus *PAM (Partitioning Around Medoids)*. Ve srovnání s k – means metodou je tato metoda méně ovlivnitelná odlehlými hodnotami, avšak její provedení je mnohem dražší. [12]

Hierarchické metody

Hierarchické metody vytvářejí stromovou strukturu shluků vzorků. Existují dva typy rozkladu a to *shlukující* a *rozdělující hierarchické modely. Shlukující hierarchická metoda* provádí shlukování zdola – nahoru. Nejprve je každý vzorek umístěn do samostatné skupiny. Následně dochází ke slučování těchto skupiny do větších, dokud netvoří jednotnou skupinu nebo nejsou splněny požadované podmínky. Patří sem např. algoritmus AGNES (AGglomerative NESting). Oproti tomu *rozdělující hierarchické modely* pracují ve směru shora – dolů, kdy shlukování začíná pouze s jednou skupinou. Následně je tato skupina rozdělena do menších skupin až do té doby, dokud není každý prvek v samotné skupině nebo nejsou dosaženy požadované podmínky. Reprezentantem této kategorie je algoritmus DIANA (DIvisive

ANAlysis). Jednotlivé skupiny jsou potom slučovány či rozdělovány na základě jejich vzdáleností. V potaz jsou brány níže uvedené metriky, kde $|p - p'|$ je vzdálenost mezi vzorky p a p', m_i je střed skupiny C_i a n_i je počet objektů skupiny C_i.

- **Minimální vzdálenost**: $d_{min}(C_i, C_j) = min_{p \in C_i, p' \in C_j} |p - p'|$
- **Maximální vzdálenost**: $d_{max}(C_i, C_j) = max_{p \in C_i, p' \in C_j} |p - p'|$
- **Střední vzdálenost**: $d_{mean}(C_i, C_j) = |m_i - m'_j|$
- **Průměrná vzdálenost**: $d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$

Metody založené na maximální či minimální vzdálenosti jsou velmi náchylné na zašumená nebo odlehlá data. Metody založené na střední a průměrné vzdálenosti těmto problémům předcházejí. U obou metod platí, že pokud se provede operace sloučení či rozdělení, nejde tato operace vzít zpět. Proto rozhodování při spojení či rozdělení skupin vyžaduje prověření velkého množství možností. Obvykle se hierarchické metody kombinují s dalšími shlukovacími metodami. [5]

Kapitola 4

Programovací jazyky pro dolování dat

Důležitým rozhodnutím datového analytika je zvolení vhodného programovacího jazyka, který bude použit pro získávání znalostí. Z technického pohledu je možno použít libovolný jazyk, avšak mezi nejpopulárnější jazyky patří jazyk R, Python, SAS, JAVA a SQL. [11] Zejména jazyky Python a R jsou využívány předními vědci a společnostmi již řadu let, a proto obsahují knihovny pro snadnou práci s daty a jejich vizualizaci. V následujících kapitolách budou tyto dva jazyky popsány a budou představeny jejich významné knihovny pro práci s daty.

Aby bylo možné snadno spouštět programy vytvořené v nejrůznějších jazycích (v současnosti je podporováno více jak 40 různých programovacích jazyků) v interaktivním grafickém prostředí v prohlížeči, je možné využít tzv. *Jupyter Notebook*[1]. Jedná se o webovou aplikaci určenou pro vytváření oddělených bloků kódů, které je možné spouštět samostatně, avšak ve výsledku tvořící celistvý program. Kromě bloků kódu může notebook obsahovat vizualizace nebo vysvětlující text ve formátu LaTeX. Nespornou výhodou této webové aplikace je možnost současného přístupu více uživatelů k jednomu notebooku, které je možné řídit pomocí autentizace a správy jejich oprávnění. Tato webová aplikace vznikla odloučením z projektu *IPython*, který je zde stále použit jako výpočetní jádro. Poslední dostupná verze knihovny nese označení 4.4.0 a je dostupná pod BSD licencí.

4.1 Python

Tento vysoko – úrovňový interpretovaný skriptovací jazyk se poprvé objevil v roce 1991 a byl navržen Guidem van Rossumem. Je populární díky své obrovské škále použitelnosti, jednoduchosti, přehledné syntaxi, čitelnosti a jeho multiplatformnosti. Oproti ostatním programovacím jazykům používá pro bloky kódu odsazení pomocí mezer místo složených závorek. Podporuje jak objektově orientované programování, tak i funkcionální či imperativní. Aktuální stabilní verze nese označení 3.5.3.

Samotný Python je velmi oblíbený pro manipulaci a transformaci dat, avšak pro analýzu a modelování dat nebylo do jádra jazyka integrováno mnoho prostředků. Tato mezera byla zaplněna vytvořením nepřeberného množství knihoven. Jelikož každá knihovna je zaměřená na rozdílné části dolování, nejčastěji se jednotlivé knihovny kombinují dohromady a společně

[1] http://jupyter.org/

tvoří výkonné, lehce použitelné prostředí pro získávání znalostí. Nyní zde budou stručně charakterizovány knihovny využité při řešení případových studiích.

NumPy

Jedná se o základní knihovnu[2] vytvořenou pro vědecké výpočty v jazyce Python. Obsahuje vysoce výkonné multidimenzionální pole s označením `ndarray`, ve kterém všechny položky musí mít stejný datový typ. Nad tímto polem jsou definované funkce pro přístup k jednotlivým dimenzím pole a jejich případné transformace. Dále disponuje rozhraním pro matematické, agregační a porovnávací operace. *NumPy* také obsahuje operace pro čtení a zápis dat do souboru. Poslední dostupná verze knihovny nese označení 1.12.0 a je dostupná pod BSD licencí.

Pandas

Významnou knihovnou pro manipulaci a analýzu velkého množství dat je knihovna *Pandas*[3] postavená na knihovně *NumPy*. Základním stavebním prvkem této knihovny je `Series` objekt, což je jednodimenzionální pole typu `ndarray` z knihovny *NumPy*. Pro lepší přehlednost může mít `Series` objekt přiřazeny názvy jednotlivých os. Na tomto základním prvku je postaven další významný prvek knihovny s názvem `DataFrame`. Jedná se o dvou dimenzionální tabulkovou strukturu, která může obsahovat jak numerická, tak i textová data. `DataFrame` nabízí rozhraní pro snadnou manipulaci s daty, např. jejich čtení, aktualizace, třídění, přidávání, odstraňování, a to buď podle indexů řádků, nebo podle názvu sloupců. Pokud data obsahují chybějící hodnoty, je možné je jednoduše odstranit nebo nahradit. Jelikož je knihovna postavena na knihovně *NumPy*, jsou v `DataFrame` dostupné všechny aritmetické i agregační operace z této knihovny. Důležitou vlastností `DataFrame` jsou sumarizující funkce, které přehledně zobrazí charakteristiku uložených dat. Knihovna *Pandas* umožňuje číst data z csv, xls souborů nebo databází, a také do nich zapisovat. Poslední dostupná verze knihovny nese označení 0.19.2 a je dostupná pod BSD-new licencí.

Statsmodels

Jedná se o Python modul, který je orientovaný na analýzu dat a na metody určené pro statistické testy. Obsahuje nejrůznější modely a funkce pro mnoho statistických výpočtů jako jsou např. ARIMA modely nebo Dickey–Fuller test. *Statsmodels*[4] je postavena nad knihovnami *NumPy* a *SciPy*[5], umožňuje taktéž pracovat s objekty knihovny *Pandas*. Pro zobrazení grafických výsledků je využívána knihovna *Matplotlib*. Poslední dostupná verze knihovny nese označení 0.8.0 a je dostupná pod BSD-new licencí.

Scikit-learn

Knihovna *Scikit-learn*[6] byla naprogramována v jazycích Python, C, C++ a slouží pro snadnou analýzu dat a z nich získávání netriviálních znalostí pomocí tříd strojového učení. Je založena na knihovnách *NumPy, SciPy, Pandas* a *Matplotlib*. Jsou zde implementovány

[2] http://www.numpy.org/
[3] http://pandas.pydata.org/
[4] http://www.statsmodels.org
[5] SciPy je knihovna pro matematické a věděcké výpočty.
[6] http://scikit-learn.org

algoritmy strojového učení jak s učitelem, tak i bez učitele. Především se jedná o třídy algoritmů pro klasifikaci, regresi nebo shlukování, tedy konkrétně o algoritmy Support Vector Machines, rozhodovací stromy, metoda nejbližších sousedů, Bayesova regrese a další. Taktéž disponuje rozhraním pro předzpracování dat, redukci počtu dimenzí a porovnání jednotlivých modelů na základě nejrůznějších kritérií. Knihovna klade důraz na její snadné použití, na optimalizovaný výkon modelů, přehlednou dokumentaci a konzistenci API. Poslední dostupná verze knihovny nese označení 0.18.1 a je dostupná pod BSD licencí.

Keras

Jedná se o vysoko-úrovňové API pro vytváření modelů neuronových sítí. *Keras*[7] je vytvořena v jazyce Python a jako výpočetní jádro využívá buď knihovnu *TensorFlow*[8] nebo *Theano*[9]. Vývojáři kladli důraz především na rychlou proveditelnost požadovaných experimentů, člověkem snadno čitelné API, modularitu a snadnou rozšiřitelnost. Po simulaci modelů je možné využít jak procesor, tak i grafickou výpočetní jednotku. Poslední dostupná verze knihovny nese označení 2.0.4 a je dostupná pod licencí MIT.

4.2 R

Jazyk R[10] byl vytvořen čistě pro statistické výpočty, analýzu a grafickou reprezentaci dat. Jedná se o open – source projekt, který byl navržen Rossem Ihakem a Robertem Gentlemanem a poprvé představen v roce 1993. R je dynamicky typovaný, multi – paradigmatický interpretovaný programovací jazyk a je možné ho spouštět na různých operačních systémech. Zdrojové kódy jádra jsou napsány v jazycích C, R nebo Fortran. Je distribuován pod licencí GNU a aktuální stabilní verze nese označení 3.3.2.

Jazyk R poskytuje prostředky pro různé statistické techniky jako je lineární a nelineární modelování, analýza časových záznamů, klasifikace, shlukování a další. Tyto techniky jsou integrované přímo v jádru jazyka a mohou být rozšířeny pomocí různých externích knihoven. Těchto knihoven existuje nepřeberné množství, avšak nyní budou představeny pouze knihovny použité při realizaci případových studiích.

Forecast

Tato knihovna[11] obsahuje metody a nástroje pro analýzu a predikci hodnot časových řad. Konkrétně obsahuje model typu ARIMA a významnou funkci `auto.arima`, která prohledává stavový prostor definovaný různým nastavením jejich parametrů a vrací nejpřesnější model. Dále zde najdeme metodu pro predikci s názvem `forecast`, metodu pro získání přesnosti predikce `accuracy` a metody pro zobrazení získaných výsledků, např. `plot.Arima`. Poslední verze knihovny nese označení 8.0 a je dostupná pod licencí GPL.

[7] https://keras.io/
[8] TensorFlow je využívána pro numerické výpočty pomocí "Data Flow Graph."
[9] Theano je určena pro vytváření a výpočet matematických výrazů.
[10] https://www.r-project.org/
[11] https://cran.r-project.org/web/packages/forecast/forecast.pdf

ggplot2

Pro grafické zobrazení získaných výsledků se využívá knihovna *ggplot2*[12], kterou je možné využít taktéž v jazyce Python. Knihovna *ggplot2* umožňuje vytvářet různé grafy pro numerická i kategorická data. Nespornou výhodou této knihovny je její úroveň abstrakce, kdy je možné snadno přidávat či odebírat jednotlivé komponenty grafu. Aktuální verze knihovny má označení 2.2.1 a je dostupná pod licencí GPLv2.

caret

Knihovna *caret*[13] byla vytvořena za účelem zefektivnit proces modelování dat. Obsahuje sady funkcí pro předzpracování dat, dále výběr nejlepších záznamů dat a vytvoření a porovnávání jednotlivých vytvořených modelů. Pro správné fungování knihovna *caret* vyžaduje instalaci mnoha dalších knihoven (celkově 27), které zapouzdřuje pod společné rozhraní. Díky tomu je možné snadno používat různé algoritmy z ostatních knihoven. Taktéž nabízí metody pro nastavení nejlepších parametrů modelů porovnáváním jejich všech možných kombinací. Tato knihovna bývá často přirovnávána ke knihovně *Scikit – learn* z jazyka Python. Aktuální verze knihovny má označení 6.0-76 a a je dostupná pod licencí GPLv2.

[12] http://www.statmethods.net/advgraphs/ggplot2.html
[13] http://topepo.github.io/caret/index.html

Kapitola 5

Případové studie

V této kapitole budou představeny dvě případové studie z oblasti získávání znalostí z dat. První případová studie je z oblasti energetiky, konkrétně predikce výroby elektřiny z jednotlivých zdrojů a její spotřeby. V druhé případové studii jsou zpracovány fotbalové statistiky. U každé případové studie jsou popsány algoritmy aplikované na dané data jak v jazyce Python, tak v jazyce R. V neposlední řadě budou získané výsledky porovnány.

Veškeré vytvořené programy jsou spouštěny na osobním počítači se čtyř – jádrovým procesorem Intel Core i5-4200M s frekvencí 2,5GHz, s 8GB operační paměti a integrovanou grafickou kartou Intel Haswell. Na tomto počítači je nainstalován 64 bitový operační systém Fedora verze 24. Dále použitý byl Python verze 3.5.3 a R verze 3.3.3. Konkrétní verze knihoven jsou popsány u každé případové studie zvlášť.

5.1 Predikce výroby/spotřeby elektřiny

Predikce výroby a spotřeby elektřiny je velice náročná, jelikož je ovlivněna mnoha neurčitými faktory. Elektřina je produkována z mnoha různých zdrojů jako jsou elektrárny jaderné, uhelné, větrné, solární nebo vodní. Mezi ovlivňující faktory patří jejich náhodné výpadky, dále u uhelných elektráren cena za uhlí a emisní povolenky v závislosti na produkci energie z přírodních zdrojů, kdy je cena za vyrobenou MWh větší než její výkupní cena a nevyplatí se ji vyrábět. S tímto konkrétně souvisí závislost na počasí, které je v posledních letech značně nepředvídatelné. Ideální model by měl vzít všechny tyto faktory v potaz a umět na ně správně zareagovat.

Tato případová studie představuje techniky pro práci s časovými řadami. K jejímu provedení jsou k dispozici čtyři datasety ve formě CSV souborů. První dva datasety obsahují hodnoty vyrobené a spotřebované elektřiny z jednotlivých zdrojů v České republice ve tvaru: *čas měření, ID zdroje, hodnota*, kde *ID zdroje* je následně převedeno na textový název podle třetího souboru s názvem *energy_names.csv*. Všechny tyto tři soubory byly získány od firmy zabývající se obchodováním s elektřinou na burze. Poslední dataset[1] s názvem *detailed-trading.csv* obsahuje informace o trhu s elektřinou během konkrétních hodin jednotlivých dní od roku 2010. Jsou zde sloupce jako je množství nabízené/požadované elektřiny v MWh, cena za 1MWh v různých zemích nebo kolik elektřiny bylo exportováno/importováno z/do Česka z/do ostatních zemí a další. Jelikož data v této formě nejsou vhodná pro aplikování dolovacích algoritmů, je nutné je předzpracovat. Všechny aplikované kroky jsou popsané v následujících kapitolách.

[1] https://www.okte.sk/en/short-term-market/published-information/annual-stm-report/

Jelikož nelze brát v úvahu všechny ovlivňující faktory, cílem této případové studie bude predikce hodnot nezávislých zdrojů výroby elektřiny a její spotřeby na následující den pomocí neuronových sítí v jazyce Python a pomocí statistického modelu ARIMA v jazycích Python a R.

Předzpracování dat

Jelikož jsou data rozdělená do čtyř souborů, je nutné provést jejich sloučení do jednoho výsledného souboru. K tomuto účelu byla použitá již dříve zmíněná knihovna *Pandas (verze 0.18)* jazyka Python. Nově vytvořený soubor je uložený ve složce *data-generated*[2] s názvem *data.csv*. Soubor obsahuje sloupce jednotlivých zdrojů energie, její spotřeby, ceny elektřiny v jednotlivých zemích a další. Indexem jednotlivých řádků dat je datum a konkrétní hodina, kdy byly hodnoty naměřeny.

Záznamy dat, které obsahovaly prázdné hodnoty, jsou nahrazeny hodnotou 0. Ostatní hodnoty jsou uloženy jako desetinné číslo a jsou zaokrouhleny na čtyři desetinná místa. Dalším krokem je vytvoření nových sloupců ze stávajících dat. Prvním, novým, sloupcem je celková výroba elektřiny v danou hodinu, kdy byly sečteny hodnoty ze všech zdrojů energie. Další vytvořený sloupec obsahuje rozdíl vyrobené a spotřebované energie. Tyto nově vytvořené sloupce mají pouze informativní charakter, pro predikce použité nejsou.

Jelikož jsou modely vytvářeny v jazycích Python i R, je výběr požadovaných sloupců, aplikování normalizace a dalších potřebných úkonů z fáze předzpracování ponecháno na konkrétní realizaci v daném jazyce.

Charakteristika dat

Před aplikováním dolovacích algoritmů je důležité vědět, jaké vlastnosti mají zpracované data. Nyní zde budou popsána data ze souboru vytvořeného v předchozím kroku.

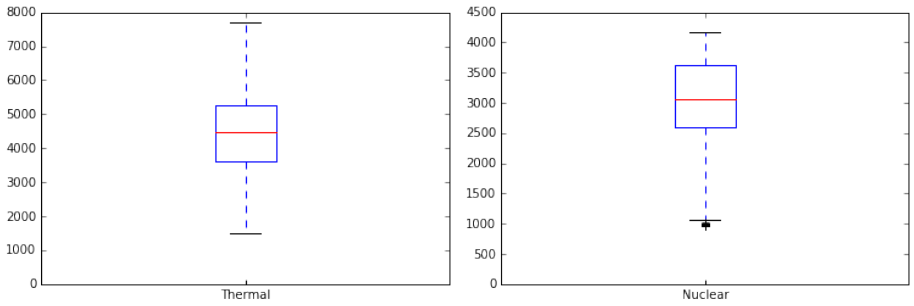
Výsledný soubor obsahuje hodnoty v hodinových intervalech od 23.4.2011 do 4.2.2017, tedy konkrétně 50 755 záznamů složených ze 34 sloupců. Pro samotnou predikci bude využito 8 sloupců hodnot jednotlivých zdrojů energie a jeden sloupec obsahují celkovou spotřebu energie na území České republiky. Charakteristiku jednotlivých sloupců je možné vidět v tabulce 5.1. Všechny hodnoty jsou udávané v MWh.

	Průměr	**Std**	**Min**	**25%**	**50%**	**75%**	**Max**
Thermal	4126.78	1783.12	0	3628.32	4476.20	5276.84	7708.33
Nuclear	2858.03	1212.08	0	2591.61	3067.03	3634.70	4174.18
Gas	294.78	242.22	0	223.53	260.25	312.03	1432.28
Hydro	146.63	124.76	0	62.10	98.75	214.57	730.53
Pump	107.90	173.40	-217.90	0	0	174.01	956.15
Biom	128.64	104.29	0	13.50	103.83	231.95	314.28
Wind	40.50	44.71	0	9.71	26.58	56.38	2655.15
Solar	206.28	360.75	0	0	1.05	264.68	1712.23
Consum.	2596.32	1821.11	0	1708.54	1 980.40	2322.01	10407.46

Tabulka 5.1: Charakteristika dat použitých pro predikci budoucích hodnot.

[2] Výslednou adresářovou strukturu projektu je možné nalézt v příloze B.

Krabicový graf rozložení hodnot pro termální a nukleární zdroj energie je možné vidět na obrázku 5.1.1. Graf s rozložením hodnot pro ostatní zdroje lze najít v příloze A.1.



Obrázek 5.1.1: Krabicový graf rozložení hodnot pro termální a jaderný zdroj energie.

Jak je možné vidět, největším zdrojem energie jsou jaderné a uhelné elektrárny, následované plynovými a vodními. Nepatrné množství výroby elektřiny tvoří přečerpávající elektrárny (Pump), biomasa (Biom), větrné a solární elektrárny. U přečerpávajících elektráren je důležité si všimnout záporné minimální hodnoty, která je způsobena spotřebou energie při přečerpávání vody do horní nádrže. Je tedy zřejmé, že pokud dojde k výpadku jaderných elektráren, musí být nahrazeny ostatními zdroji nebo energie musí být dovezena ze zahraničí. Proto se v následující kapitole budeme snažit predikovat hodnoty výroby a spotřeby elektřiny na následující den, tedy 24 hodnot.

Zpracování úlohy pomocí ARIMA modelů v jazyce Python

Jak bylo zmíněno výše, cílem této případové studie je predikce výroby a spotřeby elektřiny pro následující den za použití statistického modelu ARIMA.

Modely ARIMA (AutoRegressive Integrated Moving Average) se používají pro analýzu časových řad a predikci budoucích hodnot. Model ARIMA(p, d, q) se skládá ze tří částí a to: *autoregresní model (AR)*, *model klouzavých průměrů (MA)* a *diference časové řady (I)*. Výhodou tohoto modelu je, že umí pracovat jak se stacionárními, tak i nestacionárními časovými řadami. Stacionární časová řada je taková, která má hodnoty aritmetického průměru a rozptylu v celém průběhu řady stejné. U nestacionární řady se tyto hodnoty mění, což je známka např. trendu nebo sezonních vlivů.

Autoregresní model časové řady je založen na poznatku, že hodnoty v časové řadě jsou v relaci s předchozími hodnotami této řady. Autoregresní model AR(p) řádu p je možné definovat následovně:

$$y_t = k_1 y_{t-1} + k_2 y_{t-2} + \cdots + k_p y_{t-p} + \varepsilon_t,$$

kde $k_1, k_2, \ldots k_p$ jsou autoregresní koeficienty, ε_t je současná hodnota (bílý šum) a y_t je nově vypočítaná hodnota na základě předchozích hodnot. Řád modelu p se nejčastěji získává pomocí autokorelační funkce (ACF).

Model klouzavých průměrů se nejčastěji využívá k vyhlazení trendů časové řady. Model řádu q je vypočítán jako průměr q posledních hodnot a lze ho definovat jako:

$$y_t = \varepsilon_t + w_1 \varepsilon_{t-1} + w_2 \varepsilon_{t-2} + \cdots + w_q \varepsilon_{t-q},$$

kde $w_1, \ldots, w_q$ jsou parametry modelu a $\varepsilon_t, \varepsilon_{t-1}, \ldots, \varepsilon_{t-q}$ jsou hodnoty náhodného bílého šumu za posledních q kroků. Hodnotu q lze získat pomocí částečné autokorelační funkce (PACF).

Diferencí časové řady I(d) řádu d se rozumí rozdíl hodnot aktuální časové řady a téže řady posunuté o d kroků zpět. Cílem tohoto procesu je vytvořit takovou stacionární řadu, aby její rozdělení pravděpodobnosti bylo v čase neměnné.[6]

Protože se zde pracuje s více podobnými časovými řadami, byla vytvořena pro přehlednější práci generická třída s názvem `SourceMetadata`, která zapouzdřuje veškerou práci s konkrétní časovou řadou. Při vytváření objektu pro konkrétní časovou řadu se do konstruktoru předává `Dataframe` obsahující danou časovou řadu, její název, hodnotu alpha využitou pro výpočet confidence intervalu, počet hodnot, které chceme predikovat a parametry p, q pro ARIMA model.

Mezi veřejné rozhraní třídy patří metoda `init_data()`, která slouží pro inicializaci dat dané časové řady. Tato inicializace zahrnuje test na stacionaritu řady, ke kterému je využíván tzv. *Augmented Dickey-Fuller* test z knihovny *statsmodels* (verze 0.8). Testována je tzv. *nulová hypotéza*. Výsledkem testu je sada hodnot a lze je interpretovat tak, že pokud je hodnota s označením p menší než kritická hodnota s indexem 5%, lze s pravděpodobností 95% prohlásit časovou řadu za stacionární. Ostatní hodnoty v tomto případě nejsou důležité. Pokud časová řada zdroje nebo spotřeby energie není stacionární, je prováděna diference řady do té doby, dokud není stacionární. Celkový počet diferencí časové řady je následně využit v ARIMA modelu jako parametr d.

Metoda `make_prediction()` slouží pro vytvoření modelu ARIMA z knihovny *statsmodels* se zadanými parametry, jeho natrénování a následnému predikování výsledných dat. Pro natrénování modelu se používají všechna data z intervalu *<0; délka časové řady - počet predikovaných hodnot>*. Zbývající nepoužitá data se používají pro porovnání s predikovanými hodnotami. Jako metrika přesnosti modelu se používá střední kvadratická chyba (MSE) predikovaných hodnot od skutečných.

Poslední metoda s názvem `plot_result()` slouží k zobrazení výsledného grafu původních a predikovaných hodnot. Veškerý výše popsaný kód je možné nalézt v jupyter notebooku s názvem *Energy.ipynb*.

Jak bylo uvedeno výše, ARIMA model vyžaduje zadání parametrů p, d, q. Jelikož neexistuje žádná metoda v jazyku Python, která by tyto parametry automaticky vypočítala, bylo nutné pro každou řadu vytvářet modely s různými parametry a vybrat ten nejlepší. Jako rozhodující kritérium pro výběr nejlepšího modelu byl zvolen parametr AIC (Akaike Information Critera[3]) a střední kvadratická chyba. U obou parametrů platí, že čím nižší hodnota, tím lepší model. Výsledné získané hodnoty p, d, q pro jednotlivé časové řady je možné vidět v tabulce 5.2.

Zdroj	**p**	**d**	**q**	**Zdroj**	**p**	**d**	**q**
Nuclear	7	0	3	Pump	7	0	3
Biom	1	1	3	Solar	1	0	6
Consum.	7	0	6	Thermal	4	0	3
Gas	1	0	0	Wind	1	0	0
Hydro	1	0	0				

Tabulka 5.2: Získané parametry ARIMA modelů pro jednotlivé časové řady.

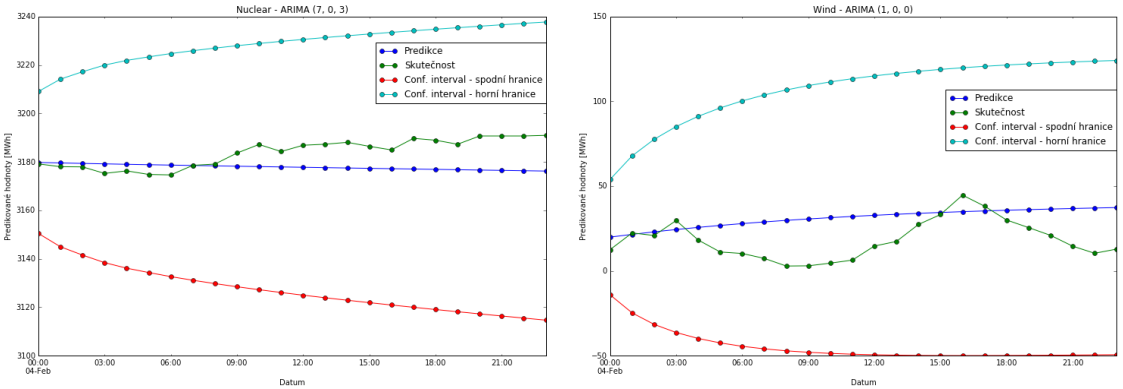
Získané střední kvadratické chyby predikce pro jednotlivé zdroje energie a její spotřebu je možné vidět v tabulce 5.3.

[3] http://www4.ncsu.edu/ shu3/Presentation/AIC.pdf

Zdroj	MSE [MWh]	Zdroj	MSE [MWh]
Nuclear	78.435	Pump	27450.595
Biom	249.093	Solar	28457.717
Consum.	133367.519	Thermal	45080.708
Gas	626.166	Wind	290.022
Hydro	1719.394		

Tabulka 5.3: Získané hodnoty střední kvadratické chyby predikce pro jednotlivé časové řady.

Výsledné predikované hodnoty výroby elektřiny na další den z jaderných a větrných elektráren je možné vidět na obrázku 5.1.2. Skutečné hodnoty jsou označené zelenou barvou, predikované barvou modrou a zbývající dvě křivky zobrazují confidence interval[4], do kterého by měly s danou pravděpodobností predikované hodnoty náležet. Tato pravděpodobnost je řízena již dříve zmíněnou hodnotou *alpha* – čím menší hodnota *alpha*, tím větší pravděpodobnost a tím větší rozsah intervalu. Jak je možné vidět na výsledném obrázku, dolní hranice confidence intervalu se v některých případech nachází v záporných hodnotách, i když je zřejmé, že záporná hodnota výroby elektřiny (kromě přečerpávacích elektráren) nikdy nemůže nastat. Je to způsobeno rozložením dat jednotlivých časových řad, které je velmi pozitivně nebo negativně vychýlené. Tudíž při predikci hodnot blízko 0 musí model reagovat na tyto výchylky jak směrem nahoru – což je správně, tak i směrem dolů. Bylo by možné omezit dolní interval nulovou hodnotou, avšak pro lepší ukázku to bylo ponecháno beze změny. Výsledná predikce pro další zdroje energií je možné nalézt v příloze A.2.



Obrázek 5.1.2: Výsledná predikce vyrobených hodnot pro jaderné a větrné elektrárny.

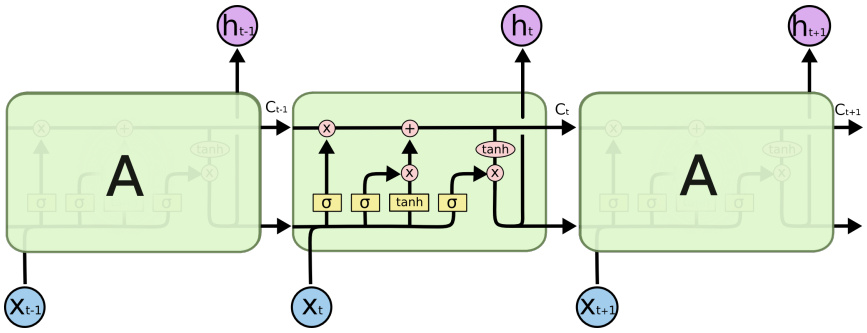
Zpracování úlohy pomocí Neuronových sítí v jazyce Python

Pro predikci budoucích hodnot časových řad lze taktéž využít neuronové sítě. Jak už název napovídá, ústředním prvkem neuronové sítě je jednoduchý procesor – neuron. Jednotlivé neurony jsou mezi sebou propojeny a komunikují pomocí signálů. Neuron může mít libovolný počet vstupů, avšak pouze jediný výstup. Neuronové sítě jsou inspirovány biologickými neuronovými sítěmi. Existuje mnoho typů neuronových sítí, např. vícevrstvé dopředné sítě, modulární sítě, sítě se zpětnou vazbou nebo fyzické neuronové sítě.

[4]Confidence interval je součástí výsledku predikce pomocí ARIMA modelu.

V této případové studii byly použity neuronové sítě se zpětnou vazbou (tzv. rekurentní), konkrétně síť zvaná Long Short Term Memory (LSTM). Jednou z výhod těchto sítí je, že si pamatují dlouhodobé závislosti mezi hodnotami a dokážou dynamicky reagovat na nové hodnoty za použití těchto zapamatovaných závislostí.

Jak je možné vidět na obrázku 5.1.3, LSTM sítě se skládají z opakujících se jednoduchých modulů neuronových sítí zřetězených za sebou. Klíčovou vlastností je možnost uchování si stavu v modulu, ke kterému dochází za pomocí horizontálního spojení označeného C_t. Tento stav je regulován několika tzv. bránami, v diagramu označenými žlutými obdélníky, a dalšími matematickými operacemi. První brána, tzv. "forget gate layer", nese označení σ a generuje číslo v intervalu $< 0; 1 >$. Toto číslo určuje, jak moc informací z předchozího výstupu h_{t-1} a aktuálního vstupu x_t bude ignorováno. Další dvě brány s označením σ a tanh slouží pro rozhodování, které informace budou uchovány ve stavu modulu. Poslední brána σ spolu s následnými matematickými operacemi určuje, jaké hodnoty budou poslány na výstup h_t.



Obrázek 5.1.3: Schéma konkrétního modulu opakujícího se v LSTM sítích.[5]

Cílem této části práce je opět predikce hodnot jednotlivých zdrojů elektřiny a její spotřeby na následující den. Pro predikování budou použita sloučená data vytvořená v předchozím kroku.

Stejně jako v předchozím případu byla vytvořena generická třída `SourceMetaData`, která zapouzdřuje práci s jednotlivými řadami. Při vytváření objektu je potřeba vložit do konstruktoru níže specifikované parametry. Hodnoty v hranatých závorkách určují výchozí hodnoty.

- **data_name** - určuje název časové řady
- **dataset** - data časové řady
- **look_back_forward** [1] - počet historických hodnot použitých pro predikci
- **layers** - jednotlivé mezi-vrstvy modelu, první a poslední vrstva je staticky definovaná
- **first_layer_ret_seq [False]** - zda bude vrácen poslední výsledek nebo celá sekvence
- **first_layer_neurons** [20] - počet neuronů první neuronové vrstvy
- **nb_epoch** [100] - počet period pro trénování modelu
- **batch_size** [240] - počet vzorků pro aktualizaci přechodu
- **verbose** [2] - slouží pro nastavení kontrolních výpisů při trénování modelu

[5] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

- **loss** [**"mean_squared_error"**] - název optimalizační funkce
- **optimizer** [**"adam"**] - název optimalizačního parametru

V konstruktoru třídy dochází k transformaci dat časové řady do požadovaného tvaru a také k normalizaci jejich hodnot do intervalu $< 0; 1 >$. Následně jsou data rozdělena na trénovací a testovací množiny. Jelikož se zde používá metoda učení s učitelem, výsledná data pro trénování a testování mají tvar *hodnota 1, hodnota 2, ..., hodnota X – výsl. hodnota*.

Dalším krokem je inicializace modelu, která je zajištěna metodou `init_model()`. Zde je vytvořen `Sequential` model z knihovny *Keras* (verze 1.2.2) a inicializován již dříve specifikovanými vrstvami a ostatními parametry. Posledním krokem této metody je jeho natrénování na trénovacích datech.

Další metoda třídy se nazývá `make_prediction()`, ve které dochází k predikci budoucích dat, jejich transformaci do původního rozsahu a vypočítání střední kvadratické chyby.

Poslední metoda `display_results()` slouží pro zobrazení grafu predikovaných hodnot pomocí knihovny *matplotlib*. Veškeré výše popsané funkce a celkový kód této části případové studie je možné vidět v jupyter notebooku s názvem *Neural Networks.ipynb*.

Jelikož je model závislý na mnoha parametrech a neexistuje žádný deterministický způsob, jak je jednoduše určit, bylo nutné tyto parametry získat za pomocí testování a porovnávání modelů. Jako rozhodující kritérium pro vybrání nejlepších parametrů byla zvolena výsledná střední kvadratická odchylka skutečných a predikovaných hodnot. Konkrétně se jednalo o vrstvy modelu, počet neuronů první vrstvy, počet period, zda je mezi první a druhou vrstvou zpětné propagování a také počet historických hodnot, které se berou v úvahu. Výsledné získané hodnoty pro jednotlivé časové řady je možné vidět v tabulce 5.4. Vrstvy jednotlivých modelů je možné videt v příloze A.3. Důležité je zmínit, že čím vyšší počet vrstev, počet neuronů, počet period a menší velikost dávky, tím delší čas je potřebný k natrénování modelu, což v tomto počtu řad může mít zásadní roli na rychlost predikce.

	Počet neuronů	**Period**	**Propagace**	**Vel. dávky**
Nuclear	30	80	True	192
Biom	40	90	True	96
Consum.	120	100	True	120
Gas	70	100	True	144
Hydro	20	100	False	288
Pump	20	40	True	48
Solar	30	80	True	264
Thermal	30	100	True	24
Wind	40	120	True	96

Tabulka 5.4: Získané parametry pro jednotlivé modely neuronových sítí.

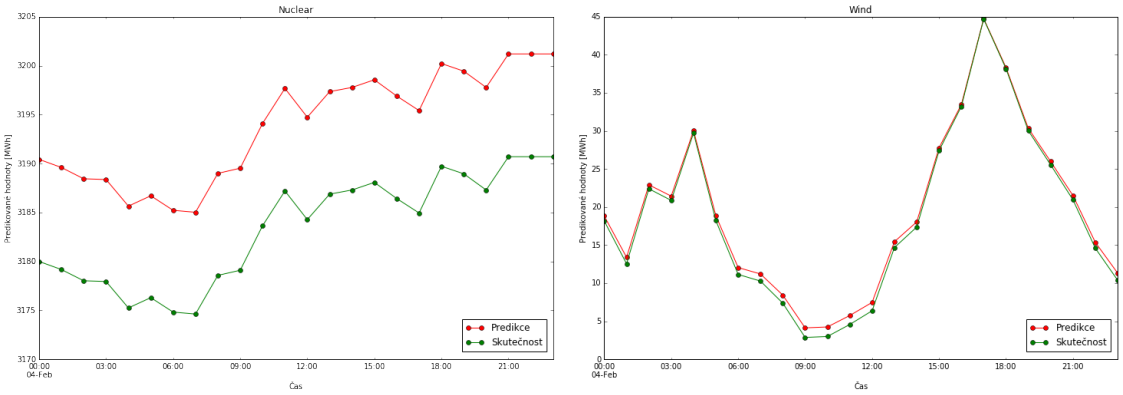
Získané střední kvadratické chyby predikce pro jednotlivé zdroje energie a její spotřebu pomocí neuronových sítí je možné vidět v tabulce 5.5.

Výsledné predikované hodnoty pro další den z jaderných a větrných elektráren je možné vidět na obrázku 5.1.4, kde skutečné hodnoty jsou označené zelenou barvou a predikované barvou červenou. Důležité je zmínit, že predikce probíhala vždycky pro jednu následující hodnotu, nikoliv pro 24 hodnot jako u ostatních případů. Bylo to z toho důvodu, že predikce

Zdroj	MSE [MWh]	Zdroj	MSE [MWh]
Nuclear	16.76	Pump	9235.84
Biom	68.82	Solar	5178.18
Consum.	6809.03	Thermal	9545.67
Gas	101.76	Wind	44.90
Hydro	998.60		

Tabulka 5.5: Získané hodnoty střední kvadratické chyby pro jednotlivé zdroje a její spotřebu.

více hodnot byla velmi nepřesná a tudíž nepoužitelná. Výsledné predikce pro další zdroje energií je možné nalézt v příloze A.4.



Obrázek 5.1.4: Výsledná predikce hodnot na následující den pro jaderné a větrné elektrárny.

Zpracování úlohy pomocí ARIMA modelů v jazyce R

Poslední částí této případové studie bylo vytvoření modelů pro predikci budoucích hodnot časových řad v jazyce R. V tomto případě byly taktéž použité modely typu ARIMA popsané výše. Jak již bylo zmíněno, tyto modely pracují s parametry p, d a q, které bylo nutné v jazyku Python získat ručním testováním. Oproti tomu v jazyku R existuje metoda `auto.arima` z knihovny *forecast* (verze 8.0), která tyto parametry získá automaticky. Parametry je možné specifikovat, zda je časová řada stacionární a taky horní hranici intervalu, ve kterém se hodnoty p, d a q budou hledat. Získané hodnoty lze vidět v tabulce 5.6.

Zdroj	p	d	q	Zdroj	p	d	q
Nuclear	0	0	5	Pump	2	0	1
Biom	0	0	5	Solar	3	0	3
Consum.	3	0	0	Thermal	0	0	5
Gas	1	0	2	Wind	2	0	2
Hydro	1	0	2				

Tabulka 5.6: Automaticky získané parametry modelů ARIMA pro jednotlivé časové řady.

Jelikož je R v jádru funkcionální jazyk, nebyla v tomto případě vytvořena třída zapouzdřující práci s jednotlivými časovými řadami, ale samostatné funkce. Hlavní funkce pro řízení predikce budoucích hodnot nese jméno `make_data_prediction()`. Mezi její parametry patří parametr `data`, který obsahuje danou časovou řadu. Parametr `predict_values`

určuje, kolik budoucích hodnot se bude predikovat a zda normalizovat data do intervalu $< 0; 1 >$ je specifikováno parametrem `use_normalization`. Pro určení výsledné velikosti confidence intervalu je využit parametr `alpha`.

Mezi další funkce patří `split_data()`, která slouží pro rozdělení dat na trénovací a testovací množiny. Jako parametr jsou předána data a počet hodnot, které se budou predikovat.

Funkce, která zapouzdřuje hledání výsledného modelu za pomocí `auto.arima` se nazývá `create_model()`. Parametrem této funkce jsou hranice intervalů pro p, d, q, ve kterých se bude výsledný model hledat. Konkrétně byla hranice intervalu nastavena pro všechny tři parametry na hodnotu 10.

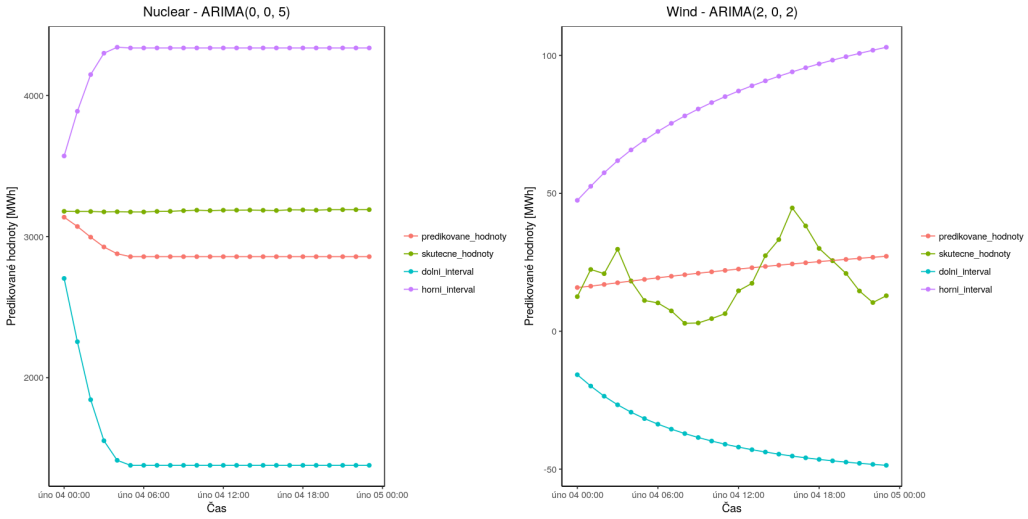
Grafické reprezentace výsledků predikce jsou zobrazeny ve funkci `plot_result()` s využitím knihovny *ggplot* (verze 2.2.1). Tyto funkce a celkový kód je možné najít v jupyter notebooku s názvem *R - Energy.ipynb*.

Získané střední kvadratické chyby predikce pro jednotlivé zdroje energie a její spotřebu je možné vidět v tabulce 5.7.

Zdroj	**MSE [MWh]**	**Zdroj**	**MSE [MWh]**
Nuclear	93268.12	Pump	30136.03
Biom	15100.33	Solar	27726.18
Consum.	34043.12	Thermal	2991806.76
Gas	532.1	Wind	130.28
Hydro	2461.3		

Tabulka 5.7: Získané hodnoty střední kvadratické chyby pro jednotlivé zdroje a její spotřebu.

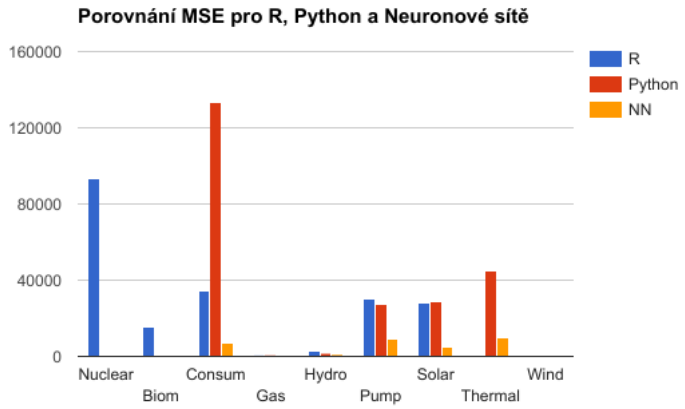
Výsledné predikované hodnoty pro následující den z jaderných a větrných elektráren je možné vidět na obrázku 5.1.5. Zbývající grafy je možné nalézt v příloze A.5. Na obrázku je také zobrazen confidence interval s pravděpodobností 95%. U některých časových řad zde nastává stejný případ se zápornou dolní hranicí confidence intervalu jako u jazyka Python. Logické vysvětlení této skutečnosti bylo popsáno na konci kapitoly 5.1.



Obrázek 5.1.5: Výsledná predikce hodnot na následující den pro jaderné a větrné elektrárny.

Zhodnocení výsledků

Protože u každé predikce byla vypočítána střední kvadratická odchylka (MSE) skutečných a predikovaných hodnot, je možné použít tuto míru pro srovnání přesnosti modelů. Jak je možné vidět v jednotlivých tabulkách obsahující tyto hodnoty nebo na obrázku 5.1.6, u některých časových řad se hodnoty liší v řádu stovek až tisíců MWh.



Obrázek 5.1.6: Porovnání naměřených MSE hodnot z jednotlivých modelů.[6]

U porovnání ARIMA modelů je nutné zdůraznit, že získané parametry p, d, q jsou u jednotlivých jazyků různé. Jedním z možných vysvětlení je nastavení dalších parametrů jako je *allowdrift, biasadj* a další u metody *auto.arima*. Tyto parametry byly ponechány ve výchozím nastavení. Dalším možným vysvětlením je, že *auto.arima* používá k získání nejlepšího modelu různé heuristické a optimalizační funkce, které výsledek ovlivňují. Avšak jak můžete vidět na obrázku 5.1.6 u řady s názvem Nuclear nebo Thermal, což jsou dva největší zdroje energie, je výsledek v jazyce R podstatně horší.

Pokud se zaměříme na výsledek z neuronových sítí, vidíme, že hodnoty predikce jsou oproti původním hodnotám mírně posunuté. Při aplikování detailnějšího testování nastavení modelů by mohly být získány ještě více relevantnější výsledky. Výsledky neuronových sítí jsou ze všech modelů nejlepší, avšak predikce probíhala pouze pro 1 hodnotu, nikoliv následujících 24 hodnot, a tedy kumulovaná chyba u dalších hodnot nebyla tak velká.

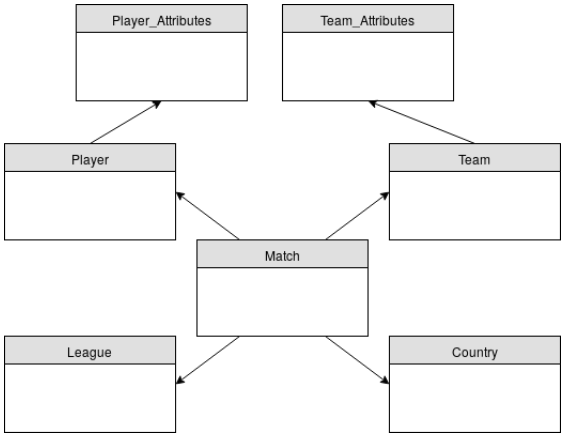
Tyto výsledky jsou použitelné v praxi pro krátkodobý přehled vývoje jednotlivých zdrojů a spotřeby energie, avšak při obchodování s elektřinou je důležitá dlouhodobější predikce. Pro dlouhodobější predikci by bylo nutné původní data zprůměrovat např. po týdnu, tudíž při predikci několika budoucích hodnot by docházelo k získání průměrných hodnot pro několik následujících týdnů. Jak je ale možné z grafů vidět, výroba nebo spotřeba může kolísat i během několika hodin, tudíž by tato predikce nebyla moc detailní. A jelikož je cena závislá na poměru vyrobené a spotřebované energie, mohly by tyto "hrubé" informace vést ke špatným rozhodnutím při uzavírání obchodů. Možným řešením by mohla být kombinace modelů používající pro natrénování a predikci různé časové intervaly mezi jednotlivými záznamy dat.

[6]Hodnota MSE u řady Thermal pro jazyk R byla z grafu vypuštěna, jelikož byla o několik řádů větší než všechny ostatní hodnoty a došlo by ještě k většímu zkreslení grafu.

5.2 Fotbalové statistiky

Druhá případová studie je z oblasti sportu, konkrétně představuje práci s fotbalovými statistikami. Existuje velké množství úloh a otázek, na které tato data mohou přinést odpověď z výsledků použití různých dolovacích algoritmů. Avšak konkrétním cílem této případové studie bude klasifikace výsledného stavu zápasu do tří možných tříd, kterými je výhra (win), remíza (draw) nebo prohra (lose).

Je zřejmé, že predikce výsledného stavu zápasu není nic jednoduchého, jelikož existuje mnoho faktorů, které mají na tento výsledek vliv. Mezi výrazně vypovídají faktory patří kvalita mužstva, styl hry (útočný, obranný), výsledky posledních zápasů a s tím související postavení v tabulce, domácí prostředí, vzájemná bilance s aktuálním soupeřem nebo soupeřovy poslední zápasy. Kvalitou mužstva se rozumí vlastnosti jednotlivých hráčů, jejich aktuální forma a další. Mezi neurčité faktory patří aktuální pohoda a zdravotní stav hráčů, počasí, zásahy trenéra do sestavy v průběhu zápasu a v neposlední řadě štěstí.



Obrázek 5.2.1: Schéma databáze obsahující data pro případovou studii.

K této případové studii jsou k dispozici data[7] uložená v 7 tabulkách databáze typu sqlite. Tabulky databáze jsou logicky uspořádané do schématu sněhové vločky, viz. obrázek 5.2.1, kde hlavní tabulka faktů nese název *match*. Jako rozšiřující dimenze jsou zde dostupné tabulky s názvem *country, league, player* a *team*. Jelikož má databáze schéma sněhové vločky, je tabulka *team*, resp. *player* spojena s rozšiřující tabulkou s názvem *team_attributes*, resp. *player_attributes*. Tyto rozšiřující tabulky obsahují nejrůznější statistiky o daném týmu, resp. hráči, za posledních několik let.

Jelikož v atributech záznamu zápasu nejsou obsaženy některé z výše zmíněných faktorů, např. výsledky posledních několika zápasů, bude nutné tyto atributy vytvořit. Tudíž bude kladen velký důraz na fázi předzpracování, konkrétně tvorba atributů, jejich standardizaci a popř. odstranění nepotřebných atributů technikou redukce. Samotná klasifikace bude pro lepší porovnání provedena za pomocí metody *k – nejbližších sousedů, Bayesovy klasifikace, logické regrese* a metody zvané *náhodný les*. Pokud výsledky klasifikace ověřené pomocí skóre přesnosti a konfúzní matice nebudou dostatečné, bude nutné zkoušet různé nastavení parametrů modelu, popř. ještě vytvořit více vypovídající atributy o výsledku zápasu ve fázi předzpracování.

[7] https://www.kaggle.com/hugomathien/soccer/downloads/database.sqlite.zip

Charakteristika dat

Konkrétně tato databáze obsahuje data o 11 evropských ligách jako je anglická, španělská, italská a další. Každá liga je tvořena několika týmy, které se mohou v průběhu sezón měnit. V celkovém součtu je v databázi uloženo 299 týmů s jejich charakteristickými atributy z rozmezí let 2010 až 2015. Tyto atributy jsou jak numerické (např. defencePressure), tak i kategorické (např. defencePressureClass). Celkově tedy tabulka *team_attributes* obsahuje 1 458 záznamů vlastností týmů. Stejný formát jako u týmů je použit u hráčů. Tabulka *player* obsahuje 11 060 záznamů hráčů se základními informacemi o hráči jako je jméno, váha a výška. Vlastností hráčů, jako je jejich průměrný výkon, preferovaná noha a další, z rozmezí let 2007 až 2016, jsou uloženy v tabulce *player_attributes* v celkovém počtu 183 978 záznamů. Poslední nejdůležitější tabulka s informacemi o zápasech má název *match* a obsahuje 25 979 záznamů. Každému zápasu odpovídá jeden záznam v tabulce a jsou v něm uloženy id hračů, kteří hráli daný zápas, liga a sezóna, do které zápas patří, konečný výsledek, události v zápase uložené ve formátu xml, kurzovní sázky z 10 předních sázkových kanceláří a další.

Avšak takto uložená data nejsou vhodná pro aplikování klasifikačních algoritmů, protože obsahují prázdné hodnoty a také nejsou ve tvaru, který by zahrnoval více ovlivňujících faktorů výsledku zápasu. Tudíž je nutné uložená data vhodně předzpracovat. Toto předzpracování je popsáné v následující kapitole. Jelikož jsou ligy na sobě nezávislé, bude konkretně popsáno předzpracování dat pouze z anglické ligy. Avšak tento postup je možné aplikovat i na ostatní ligy.

Předzpracování dat

Hlavním cílem této fáze dolování je ukázání, že pokud datový sklad obsahuje mnoho atributů, které mají menší vypovídající hodnotu o výsledném atributu (v tomto případě výsledný stav zápasu), je snadné je transformovat na atributy s vyšší vypovídající hodnotou.

Aby nebylo nutné udržovat podobný kód pro fázi předzpracování u obou jazyků, což by mohlo být náchylné na chyby a tudíž by mohlo docházet k aplikování klasifikace na různá data, byla fáze předzpracování provedena pouze v jazyce Python.

V prvé řadě je nutné načíst data z databáze do vhodné datové struktury. K tomuto účelu, podobně jako v předchozí případové studii, byla využita knihovna *Pandas*. Tato knihovna disponuje rozhraním pro načítání dat z databáze do `Dataframe` objektu za pomocí SQL dotazů. Tímto způsobem byly uloženy do třech různých `Dataframe` objektů záznamy o zápasech z anglické ligy, o týmech a hráčích i společně s jejich vlastnostmi.

Všechny zápasy, které mají alespoň jednu prázdnou hodnotu ve sloupcích s identifikátorem (cizí klíč) hráče ze základní sestavy, jsou ignorovány.

Následně dochází k přidání textového štítku podle výsledku zápasu (win, draw, lose), který bude jednotlivými klasifikátory predikován. Taktéž dochází k vytvoření 6 nových sloupců s názvem home_team_[win|draw|lose] obsahující hodnotu 1 v případě, že zápas skončil daným výsledkem z pohledu domácího týmu. Tím samým způsobem jsou vytvořeny nové sloupce z pohledu hostujícího týmu. Tyto sloupce jsou využité pro snadnější agregaci zápasů podle výsledku při následném zpracování.

V dalším kroku dochází k iterování skrze jednotlivé zápasy a konkrétní zápas je předán jako parametr funkci `process_match()` spolu s dalšími parametry. Tato funkce slouží jako řídící jednotka pro převod aktuálního formátu zápasu na nový tvar, který obsahuje kromě informací o týmech, datu, sezóně, štítku zápasu, následující atributy:

- **home_players_avg_rank** - průměrná kvalita hráčů domácího týmu.
- **diff_home_goals** - rozdíl vstřelených a obdržených gólů domácího týmu za posledních X zápasů, kde X je parametrem funkce.
- **home_team_home_goals_avg** - průměrný počet vstřelených gólů domácího týmu za posledních X domácích zápasů.
- **home_team_wins|home_team_draws|home_team_loses** - jedná se o tři samostatné atributy, které určují, kolik z posledních X zápasů domací tým vyhrál, resp. remizoval nebo prohrál, ať už v roli domácího, nebo hostujícího týmu.
- **home_team_wins_against** - tento atribut obsahuje počet vítězných vzájemných X zápasů domácího týmu proti hostujícímu týmu. Dále existují atributy(mění se pouze podřetězec názvu atributu na wins, draws nebo loses), které ukazují počet remizovaných či prohraných zápasů.
- **home_team_prob_win** - u každého zápasu jsou dostupné kurzy od různých sázkových kanceláří. Kurzy z jednotlivých sázkových kanceláří byly zprůměrovány a převedeny na pravděpodobnost. Tato pravděpodobnost určuje, jak moc sázkové kanceláře věří danému týmu, v tomto případě vítězství domácímu týmu. To samé bylo provedeno pro remízu a prohru, kde název nově vytvořených atributů se liší pouze v podřetězci win, draw nebo lose.

Všechny výše zmíněné atributy jsou brány z pozice domácího týmu, avšak ty samé atributy, mající v názvu podřetězec away místo home, byly vytvořeny pro hostující tým. Celkově bylo vytvořeno 29 atributů a 2963 záznamů pro anglickou ligu. Při testování bylo za X uvažováno posledních 5 zápasů. Takto transformovaná data jsou nyní připravena pro aplikování klasifikačních algoritmů, což je popsáno v následující kapitole.

Zpracování studie v jazyce Python

Jak už bylo řečeno, cílem této případové studie bude predikce výsledného stavu zápasu z pohledu domácího týmu za pomocí metody *k – nejbližších sousedů, Bayesovy klasifikace, logické regrese* a metody zvané *náhodný les*. Všechny tyto techniky byly detailněji představeny v kapitole 3.

K vytvoření modelů, výše popsaných technik, byla využita knihovna *Scikit-learn* (verze 0.18.1). Jelikož každý z modelů může mít různé nastavení parametrů a na první pohled není zřejmé, jaké parametry budou pro daný model nejlepší, je nutné brát tento úkol jako úlohu na prohledávání stavového prostoru s cílem nalezení nejlepších parametrů modelu. Výhodou knihovny *Scikit-learn* je, že obsahuje třídu `GridSearchCV`, která byla k tomuto účelu vytvořena. Tato třída umožňuje pracovat s tzv. `Pipeline`, které definují a provádí jednotlivé fáze dolování sekvenčně za sebou. Jedná se především o fáze předzpracování dat jako je např. normalizace hodnot nebo redukce počtu atributů. Nutnou podmínkou je, aby v posledním kroku `Pipeline` byl obsažen model pro klasifikaci. Dále je možné u metody `GridSearchCV` specifikovat množiny parametrů pro jednotlivé kroky `Pipeline`, které budou otestované při hledání nejlepšího modelu.

V této konkrétní úloze byly použity tři kroky `Pipeline`. První krok obsahoval objekt `StandardScaler`, který slouží ke standardizovaní hodnot jednotlivých atributů. Standardizací dat je myšleno odečtení průměru hodnot atributu od každé jeho hodnoty a taky jejich transformací tak, aby měly jednotkový rozptyl.

Druhý krok slouží k redukci atributů dat, ke kterému je využitá technika zvaná Analýza hlavních komponent (PCA). Tato technika hledá ve všech datech ty atributy, které mají největší rozptyl, a tudíž nejvíce ovlivňují výslednou predikci. Nevýhodou je, že dochází k zakódování atributů s největším rozptylem do tzv. komponent, ze kterých není možné zpátky získat názvy původních atributů. Proměnným parametrem je zde počet komponent, které budou použité pro predikci.

V posledním kroku je model, u kterého je možné specifikovat nejrůznější parametry. Tyto parametry jsou pro jednotlivé modely různé, jak je možné vidět v tabulce 5.8. Parametr `n_estimators` určuje počet stromů v *náhodném lese*, `min_samples_split` definuje minimální počet vzorků pro rozdělení vnitřního stromu uzlu, `max_features` určuje počet atributů testovaných ve vnitřním stromu uzlu. U klasifikátoru *KNN*, parametr `weight` určuje váhovou funkci použitou při predikci a `p` určuje metodu výpočtu vzdálenosti jednotlivých záznamů. U *Log. regrese* se jedná o parametr `solver` určují funkci pro hledání nejlepšího řešení. U Bayesova klasifikátoru se žádné další parametry nenastavují.

Klasifikátor	**Parametr**	**Hodnoty**
Náhod. les	*n_estimators*	10, 12, 16, 100, 200, 400
	min_samples_split	2, 4, 6
	max_features	sqrt, log2, auto
KNN	*n_neighbors*	6, 9, 50, 90, 130, 200, 400
	weights	uniform, distance
	p	1, 2, 3
Log. Regrese	*C*	1, 2, 10
	solver	lbfgs, sag, newton-cg

Tabulka 5.8: Atributy jednotlivých modelů a hodnoty, které byly testovány s cílem získání nejlepšího modelu.

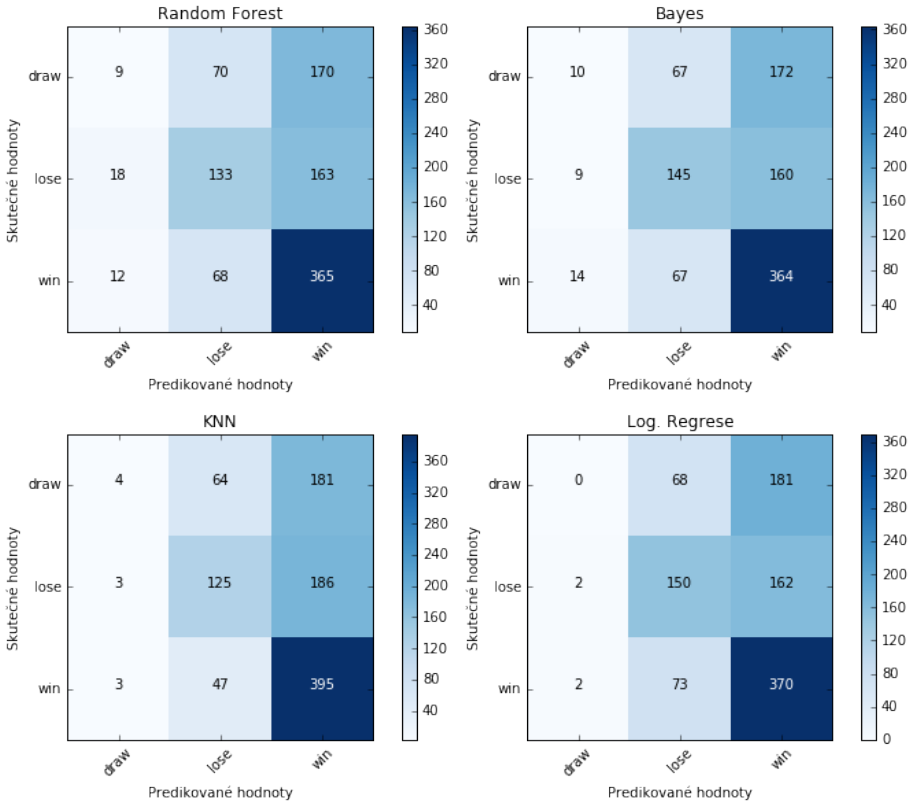
Při klasifikaci je taktéž možné u některých klasifikátorů zobrazit, s jakou pravděpodobností byl klasifikován daný záznam do které třídy. Avšak některé modely mohou poskytnout špatný odhad pravděpodobnosti[8]. Pro správné "zkalibrování" výsledné pravděpodobnosti modelů, kterou je pak následně možné použít jako úroveň důvěryhodnosti (confidence level), slouží třída `CalibratedClassifierCV`. Tato třída má jako parametr daný klasifikátor, název metody použité pro kalibraci a číselnou hodnotu určující počet "hromádek" při křížové validaci. Výsledné pravděpodobnosti testovacích hromádek jsou následně zprůměrovány a použité pro kalibraci klasifikátoru. Výhodou je, že tato kalibrace může být součástí dříve popsané techniky `Pipeline`.

Pro řízení klasifikace zde byla vytvořená funkce s názvem `make_classification()`, které je jako parametr `clf` předán klasifikátor, parametr `grid_params` obsahuje množiny hodnot využitých při různých nastavení parametrů klasifikátorů, parametr `data` obsahuje použitá data pro predikci a boolevská proměnná `use_calibration` určuje, zda bude použitá dříve zmíněná kalibrace.

[8] http://machinelearning.wustl.edu/mlpapers/paper_files/icml2005_Niculescu-MizilC05.pdf

Aby bylo možné otestovat přesnost klasifikátoru, byla dříve popsaná data rozdělena na dvě části: trénovací (2/3) a testovací (1/3). K tomuto účelu byla vytvořena funkce s názvem `split_data()`, která má jako parametr koeficient určující velikost trénovací množiny dat. Dále jsou to samotná data a název sloupce obsahují výsledné klasifikační třídy.

Pro určení přesnosti klasifikátoru bylo využito několik metrik. První metrika, popsaná v kapitole 3.1, určující přesnost klasifikátoru je konfúzní matice. V kódu programu je graficky zobrazena za pomocí metody `plot_confusion_matrix()`. Výsledky získané z testování jednotlivých klasifikátorů je možné vidět na obrázku 5.2.2.



Obrázek 5.2.2: Výsledné konfúzní matice pro testy jednotlivých klasifikátorů.

Jednou z dalších numerických metrik správnosti klasifikace je *skóre přesnosti* (accuracy score), které je získané pomocí knihovní funkce s názvem `accuracy_score()`. Vypočítá se jako $\frac{\text{počet správně klasifikovaných záznamů}}{\text{celkový počet záznamů}}$.

Další metrikou je *preciznost* (precision), která je vypočítaná jako $\frac{tp}{tp+fp}$, kde tp (true positive) označuje počet správně klasifikovaných hodnot v dané třídě a fp značí nesprávně klasifikované hodnoty v téže třídě. Čím je hodnota u této metriky bližší 1, tím je klasifikátor přesnější.

Následující metrikou je tzv. *úplnost* (recall), která vyjadřuje počet vybraných relevantních záznamů oproti celkovému počtu relevantních záznamů v souboru. Vypočítána je pomocí vzorce $\frac{tp}{tp+fn}$, kde fn (false negative) označuje počet nesprávně klasifikovaných záznamů.

Na základě těchto dvou metrik je možné vypočítat metriku $F1 = 2 * \frac{\text{preciznost} * \text{úplnost}}{\text{preciznost} + \text{úplnost}}$, která určuje výslednou přesnost testu. Hodnoty tp, fp, fn je možné získat výpočtem z hodnot konfúzní matice.

Jelikož jsou tyto vzorce definované pro binární klasifikátory, je u vícenásobné klasifikace nutné aplikovat techniku zvanou *mikro – průměr* (micro – average), resp. *makro – průměr* (macro – average), která ukazuje obecné výsledky modelu. Hlavním rozdílem těchto dvou technik je, že technika *makro – průměr* přiřazuje všem třídám klasifikace stejnou váhu, zatímco *mikro – průměr* přiřazuje stejnou váhu každému rozhodnutí uvnitř dané třídy. Tudíž např. výsledná hodnota *mikro – průměru* vzorce na výpočet *preciznosti* predikce je možné získat následovně:

$$preciznost_{mikro} = \frac{\sum_{i=1}^{n} tp_i}{\sum_{i=1}^{n} tp_i + \sum_{i=1}^{n} fp_i}$$

Hodnota n označuje celkový počet klasifikačních tříd. Podobným způsobem je možné získat výslednou hodnotu pro ostatní metriky. [2] Výpočet hodnoty *makro – průměru* se získá průměrem výsledných hodnot daných metrik. Veškeré výsledné hodnoty jsou získané za pomocí knihovní funkce `precision_recall_fscore_support()` a konkrétní výsledky testování je možné vidět v tabulce 5.9.

Metrika		**Bayes**	**KNN**	**Náhod. les**	**Log. regrese**
Skóre přesnosti		0.514881	0.519841	0.509921	0.515873
Preciznost	**mikro**	0.50099	0.470238	0.499008	0.519841
	makro	0.424370	0.414584	0.441723	0.459890
Úplnost	**mikro**	0.50099	0.470238	0.499008	0.519841
	makro	0.439517	0.421525	0.438496	0.447676
F1	**mikro**	0.50099	0.470238	0.499008	0.519841
	makro	0.410056	0.412443	0.422349	0.410967

Tabulka 5.9: Výsledné získané metriky pro jednotlivé klasifikátory v jazyce Python.

Veškeré výše popsané transformace dat, jednotlivé funkce a samotné testování modelů je možné nalézt v jupyter notebooku s názvem *Python-Fotbal.ipynb*.

Zpracování studie v jazyce R

Stejně tak, jako u předchozí případové studie, je tato studie vytvořena také v jazyce R. Pro klasifikaci jsou použité již dříve popsané techniky a to *k – nejbližších sousedů, Bayesova klasifikace, logická regrese* a *náhodný les*. Aby nebylo nutné data stejným způsobem znova předzpracovat, byla použitá data předzpracovaná v kapitole 5.2. Data byla taktéž rozdělena na trénovací (2/3) a testovací (1/3) množiny dat.

Aby bylo možné relevantně porovnat výsledky klasifikace z obou jazyků, bylo snahou co nejpodobnější nastavení parametrů jednotlivých modelů v závislosti na možnostech využitých knihoven v jazyce R.

Tak, jako při realizaci v jazyce Python, je nutné najít nejlepší parametry modelu způsobem prohledávání stavového prostoru. K tomuto účelu slouží funkce `train()` z knihovny *caret* (verze 6.0), která vrací nejlepší model ze zadaných parametrů aplikovaných na trénovací data.

Kromě parametrů obsahujících trénovací data a jejich výsledné klasifikační třídy, je zde nutné nastavit parametr `method`, který obsahuje název konkrétního klasifikátoru (`rf` – náhodný les, `knn` – k – nejbližších sousedů, `nb` – Naive Bayes, `multinom` – log. regrese).

Další parametr `preProcess` určuje, které techniky předzpracování dat budou použité. Konkrétně byly zvoleny hodnoty "center" – od každé hodnoty atributu je odečten jejich celkový průměr, "scale" – každá hodnota atributu je podělena jejich směrodatnou odchylkou. Těmito parametry byl zaručen stejný postup, který byl proveden objektem `StandardScaler` při realizaci v jazyce Python. Posledním hodnotou parametru `preProcess` je "pca", která udává metodu pro redukci dimenzionality.

Parametr `metric` určuje, jaká metrika bude použita pro výběr nejlepšího modelu. V tomto případě byla nastavena na "accuracy", tedy nejlepší skóre přesnosti klasifikace.

`tuneGrid` parametr slouží pro nastavení všech hodnot parametrů klasifikátoru, které budou otestovány. Nevýhodou je, že je omezené, které parametry[9] mohou být otestovány pro daný model. Ostatní parametry modelu mohou být nastaveny přímo ve funkci `train` jako dodatečné parametry.

Poslední specifikovaný parametr má název `trControl` a obsahuje objekt sloužící pro řízení chování této funkce při výběru nejlepšího modelu. V tomto objektu byla nastavena technika křížové validace (cross validation) s rozdělením do 10 hromádek. Pokud není zřejmé, jaké hodnoty zadat do parametru `tuneGrid`, je možné nastavit objektu `trControl` parametr `search="Random"` zaručující automatické nalezení nejlepších hodnot.

Klasifikátor	**Parametr**	**Hodnoty**
Náhod. les	*mtry*	1, 3, 6, sqrt(c), $\log_2(c)$, c
	*ntree**	400
KNN	*k*	6, 9, 50, 90, 130, 200, 400
Log. Regrese	*decay*	1, 2, 3
Bayes	*userkernel*	True, False

Tabulka 5.10: Testované parametry a hodnoty pro jednotlivé modely. (* znamená, že tento parametr byl nastaven přímo ve funkci train()).

Výhodou funkce `train` je, že vrací model společně s přehlednými informacemi o testování různých nastavení parametrů modelu. Testované parametry je možné vidět v tab. 5.10, kde parametr `mtry` má stejný význam jako parametr `max_features` v jazyce Python (hodnota c značí počet atributů), `k` označuje počet sousedů při klasifikaci technikou *k – nejbližších sousedů*, `decay` označuje regulaci pro zamezení přeučení a `userkernel` určuje využití hustoty pravděpodobnosti pro odhad výsledné třídy u Bayesova klasifikátoru.

Jakmile je získán nejlepší model, posledním krokem je predikce na neznámých datech. K tomu slouží knihovní funkce `predict()`, která má jako parametr získaný model, testovací data a typ, který určuje výsledek predikce. Typem výsledku predikce může být kategorická hodnota nebo hodnoty pravděpodobnosti pro jednotlivé třídy klasifikace.

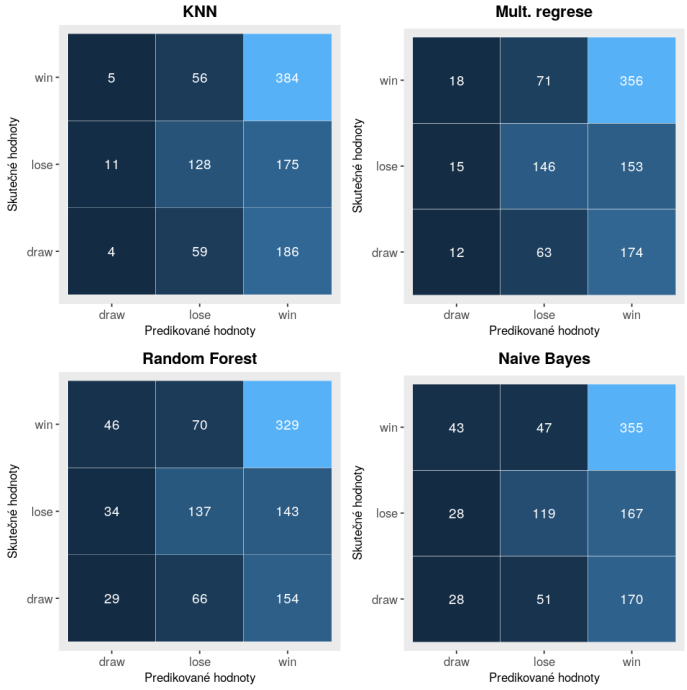
Pro určení přesnosti klasifikátoru jsou použité stejné metriky jako při realizaci v jazyce Python. Výsledná konfúzní matice je získaná díky funkci `confusionMatrix()` z knihovny `caret`. Ostatní metriky, zejména makro, resp. mikro průměr jsou vypočítané pomocí funkce `classificationMetrics()` z knihovny *performanceEstimation*. Výsledek obou funkcí je předán vytvořené funkci `plot_result()`, která vypíše výše popsané metriky, viz. tabulka

[9] https://www.jstatsoft.org/article/view/v028i05

Metrika		Bayes	KNN	Náhod. les	Log. regrese
Skóre přesnosti		0.498	0.5139	0.502	0.5089
Preciznost	**mikro**	0.4980	0.5119	0.4911	0.5099
	makro	0.4480	0.4140	0.4311	0.4014
Úplnost	**mikro**	0.4980	0.5119	0.4911	0.5099
	makro	0.4297	0.4288	0.4307	0.4377
F1	**mikro**	0.4980	0.5119	0.4911	0.5099
	makro	0.4111	0.3782	0.4144	0.4364

Tabulka 5.11: Výsledné metriky pro jednotlivé klasifikátory v jazyce R.

5.11, a vykreslí konfúzní matici za pomocí knihovny *ggplot2*. Konfúzní matice pro jednotlivé techniky je možné vidět na obrázku 5.2.3.



Obrázek 5.2.3: Výsledné konfúzní matice pro testy jednotlivých klasifikátorů.

Zhodnocení výsledků

Jako hlavní metrika pro porovnání výsledků klasifikátorů získaných provedením testů se nabízí skóre přesnosti. Výsledné skóre přesnosti klasifikace v číselné podobě je možné najít v jednotlivých kapitolách pro daný jazyk. Grafická reprezentace těchto výsledků je zobrazena na obrázku 5.2.4. Pro lepší porovnání výsledků jsou taktéž zobrazeny výsledky bez kalibrace klasifikátoru v jazyce Python. Pro jazyk R nebyla tato možnost dostupná.

Jak je možné vidět, u některých případů se výsledky pro různé klasifikátory a zvolený programovací jazyk značně liší, i když byla použita stejná data. To je způsobeno tím, že u modelů jazyka R bylo možné nastavit pouze omezené množství hledaných parametrů oproti těm, které byly nastaveny pro jazyk Python. Taktéž je možné z obrázku vyčíst, že ve

Obrázek 5.2.4: Porovnání skóre přesnosti pro jednotlivé programovací jazyky. Kalibrací označuje kalibrování klasifikátoru.

většině případů (kromě modelu pro log. regresi) byly nejlepší výsledky získány pro modely využívající kalibraci v jazyce Python. Logika této kalibrace byla popsána v kapitole při implementaci v jazyce Python.

Pokud porovnáme výsledky nejlepšího a nejhoršího modelu např. v jazyce Python bez použití kalibrace, lze vidět rozdíl přesnosti klasifikace 4,9% procent. U velkých souborů dat může tento procentní rozdíl znamenat nesprávnou klasifikaci pro několik stovek až tisíců záznamů.

Jelikož nebylo možné nastavit při prohledávání všechny stejné parametry modelů u obou jazyků, budou porovnány pouze stejně nastavené. Výsledná hodnota parametru u *náhodného lesu* je `max_features`="sqrt" (v tomto případě 4.39), resp. u jazyka R to je `mtry`=1. U metody *knn* byla výsledná hodnota parametru `n_neighbors` = 90, resp. u jazyka R hodnota `k` = 200. Pokud se podíváme na výsledné skóre přesnosti, lze vidět, že *knn* i *náhodný les* je v jazyce R přesnější. Jelikož jsou množiny hodnot parametrů stejné, je zřejmé, že celkový výsledek závisí na nastavení ostatních parametrů.

Pokud se zaměříme na hodnoty konfúzních matic u jednotlivých klasifikátorů v jazyce Python i R, vidíme, že mnoho zápasů se skutečnou třídou *draw* nebo *lose* bylo klasifikováno do třídy *win*. Tato skutečnost značí, že hlavní atributy určující výsledek klasifikace se značně překrývají, a proto dochází k těmto chybám. Možným řešením je vytvoření nových, více rozlišitelných atributů pro jednotlivé třídy klasifikace nebo vylepšit fázi předzpracování, např. odstranění takových zápasů týmů, které neodehrály před aktuálním zápasem alespoň další tři zápasy. Jedná se např. o situaci, kdy týmy postoupí do této ligy z nižší soutěže a nejsou k dispozici informace o tomto týmu z dřívějších let.

Kapitola 6

Porovnání jazyků Python a R pro dolování dat

Rozhodnout, zda pro analýzu a dolování dat je lepší jazyk Python nebo R, je velmi těžké. Oba dva jazyky nabízí prostředky pro splnění nejrůznějších úkolů z oblastí analýzy a dolování dat. Důležité je zmínit, že pro každý jazyk dále existuje velké množství externích knihoven rozšiřujících jejich vlastnosti a funkčnost.

Pro porovnání obou jazyků je nutné se zaměřit na všechny fáze dolování dat. Porovnání jednotlivých fází dolování z pohledu obou jazyků je popsáno v následujících kapitolách. V prvé řadě je však důležité se podívat u obou jazyků na podporu práce s tabulkovými daty.

6.1 Porovnání z hlediska práce s tabulkovými daty

Jak bylo řečeno výše, jazyk R byl primárně navržen pro statistické účely s důrazem na jednoduché rozhraní. Proto je v jádru tohoto jazyka zabudovaná struktura pro práci s tabulkovými daty s názvem `data.frame`. Tuto strukturu si je možné také představit jako pole vektorů o stejné délce. Jednotlivé sloupce a řádky `data.frame` mohou mít přiřazené názvy, což zlepšuje přehlednost dat. Z pohledu vstupně/výstupních operací disponuje jazyk R zabudovanými funkcemi, které umožňují načíst tabulkový soubor do struktury `data.frame` nebo tuto strukturu jednoduše zapsat do tabulkového souboru, jak je možné vidět v následující ukázce:

```
# nacteni dat ze souboru
data <- read.csv("data.csv")

# ulozeni dat do csv souboru
write.csv(data, file = "data.csv")
```

Zdrojový kód 6.1: Ukázka práce s tabulkovým souborem v jazyce R.

Jazyk Python, oproti jazyku R, nedisponuje žádnou zabudovanou strukturou vhodnou pro snadné manipulování tabulkových dat. Jelikož je to plnohodnotný programovací jazyk, není tak těžké si tuto strukturu naprogramovat. Z pohledu uživatele to však není vhodné, jelikož to zabere množství času, které může být lépe využité pro samotné dolování dat. V tomto případě je lepší použít vhodnou externí knihovnu, např. knihovna *Pandas* – typ `Dataframe`, který má tuto funkcionalitu již předdefinovanou. Ukázka načtení dat z csv

souboru do `Dataframe` objektu a následně jeho zapsání do téhož souboru, je zobrazeno v následující ukázce:

```
# nacteni knihovny
import pandas as pd

# nacteni dat z csv souboru
data = pd.read_csv('data.csv')

# ulozeni dat do csv souboru
data.to_csv('data.csv')
```

Zdrojový kód 6.2: Ukázka práce s tabulkovým souborem v jazyce Python

Na příkladu výše je důležité si všimnout významného rozdílu u těchto jazyků. Jak bylo řečeno výše, jazyk R je funkcionální jazyk, a proto je většina vlastností definovaná funkcemi. Tudíž struktura `data.frame` slouží pouze pro uchování dat a pro práci s nimi je nutné použít další funkce. Oproti tomu je jazyk Python více objektově orientovaný a u každého objektu je dostupné veřejné rozhraní definující práci s tímto objektem, např. u `Dataframe` objektu lze agregovat jeho hodnoty pomocí metody `data.sum()`.

6.2 Porovnání z hlediska charakteristiky dat

Předpokládejme, že tabulková data jsou načtená v `Dataframe` objektu `data` v jazyce Python a `data.frame` objektu, se stejným názvem, v jazyce R. Pokud chceme v jazyku R zobrazit zobrazit sumarizující informace o hodnotách jednotlivých sloupců, použijeme příkaz `summary(data)`. Oproti tomu je v jazyce Python zavolána metoda objektu ve tvaru `data.describe()`. Oba dva příkazy zobrazují stejný přehled dat.

Pro grafické zobrazení dat má jazyk R zabudovanou základní funkcionalitu pro tvorbu diagramů a grafů. Nutné je pouze znát funkci, která daný graf vytváří a zobrazí, např. pro sloupcový graf `barplot(data, main="Název",xlab="Index")`, pro histogram `hist(data)` a další. Pro pokročilejší vizualizaci je možné použít externí knihovny, které tuto základní funkcionalitu rozšiřují. Oproti tomu objekt `Dataframe` obsahuje metodu `plot()`, ve které je specifikován parametrem `kind`, např. hodnota `bar`, `hist`, typ vytvořeného grafu. Další vlastnosti grafů jsou řízeny pomocí dalších parametrů této funkce.

Další důležitou vlastností je aplikace agregačních funkcí na data. V jazyce R existují různé funkce, např. pro součet všech sloupců – `colSums()` nebo řádků – `rowSums()`, pro výpočet průměru – `colMeans()` a další. Jako parametr se těmto funkcím předají data, na které mají být aplikované. V jazyce Python, u `DataFrame` objektu, je každá agregační funkce implementována pouze jednou, např. `data.sum(axis=1)`, kde `axis` označuje osu (0 – řádky, 1 – sloupce), na kterou bude daná funkce aplikována, což je značně intuitivnější než používat různé funkce jako v R.

6.3 Porovnání z hlediska předzpracování dat

Důležitou vlastností předzpracování dat je vypořádání se s chybějícími hodnotami. Předpokládejme, že cílem je odstranit záznamy s chybějícími hodnotami. Jazyk R disponuje funkcí `complete.cases(data)`, která vrací logický vektor s hodnotou `True` pro řádky, které neobsahují chybějící hodnotu a `False` pro řádky s alespoň jednou chybějící hodnotou. Odstranění

řádků s chybějícími hodnotami je možné docílit příkazem `data[!complete.cases(data),]`. Pro kontrolu jednotlivých hodnot slouží funkce `is.na` a vrací `True` pro chybějící hodnotu, jinak `False`. V jazyce Python u *Pandas* `Dataframe` je možné odstranit všechny chybějící hodnoty příkazem `data.dropna(axis=0)`, kde `axis` opět značí osu, v tomto případě řádky, které budou odstraněny při výskytu alespoň jedné chybějící hodnoty. Požadovaný výsledek je dosažen stejně oběma způsoby.

Pro pokročilejší techniky (transformace, redukce) předzpracování dat je vhodné použít další externí knihovny. V jazyce R může být využita knihovna *Caret*, konkrétně metoda `preProcess()`. Tato metoda definuje sekvenčně aplikované techniky předzpracování pomocí parametru `method` a nastavení hodnot ostatních parametrů. V jazyce Python je vhodné použít knihovnu *Scikit-learn*, konkrétně třídy z jmenného prostoru `sklearn.preprocessing`. Oproti funkci `preProcess()` z jazyka R jsou zde jednotlivé úkony rozděleny do samostatných tříd. Pokud bychom je chtěli aplikovat sekvenčně, je nutné je zřetězit technikou zvanou `Pipeline` z téže knihovny. Nevýhodou je, že na posledním kroku `Pipeline` musí být vždy model, který provádí danou dolovací úlohu. To už se dostáváme do následující sekce s cílem porovnání modelů.

6.4 Porovnání z hlediska vytváření modelů

Jelikož byly případové studie zaměřené na práci s časovými řadami a klasifikaci, bude porovnáno vytváření a práce s těmito modely.

Pro práci s časovými řadami byly použité ARIMA modely, které jsou dostupné v externích knihovnách obou jazyků. V jazyce R se jedná konkrétně o dříve představenou knihovnu s názvem *forecast*. Knihovna disponuje jak samotným modelem ARIMA, tak i funkcí `auto.arima` pro nalezení nejvhodnějšího modelu na základě specifikovaných parametrů. Proměnné `p_max`, `d_max`, `q_max` určují maximální hodnoty parametrů. Tudíž k nalezení nejlepší modelu slouží pouze jediná funkce.

```
library(forecast)
model <- auto.arima(data, max.D=d_max, max.P=p_max, max.Q=q_max)
```

Zdrojový kód 6.3: Nalezení nejlepšího ARIMA modelu v jazyce R.

Knihovna, která obsahuje ARIMA model v jazyce Python, má název *Statsmodels*. Nevýhodou je, že nedisponuje žádnou funkcí pro automatické nalezení ARIMA modelu s nejlepšími parametry. Proto je nutné iterovat skrze parametry a vybrat ten nejlepší model, jak je možné vidět na následující ukázce kódu. Proměnné `p_max, d_max, q_max` určují maximální hodnoty parametrů modelu.

```
best_model = None    # promenna pro ulozeni nejlepsiho modelu

for d in range(d_max):
    for q in range(q_max):
        for p in range(p_max):

            # vytvoreni ARIMA modelu s parametry p, d, q
            # otestovani a porovnani modelu
            # pri lepsim vysledku nahrazeni modelu v best_model
```

Zdrojový kód 6.4: Nalezení nejlepšího ARIMA modelu v jazyce Python.

Pro klasifikační úlohy v jazyce R je možné využít knihovnu *caret*. Tato knihovna disponuje funkcí `train()`, která umožňuje najít nejlepší model ze zadaných množin parametrů. Jako parametry potřebuje název klasifikátoru, názvy technik předzpracování dat a ostatní potřebné parametry. Nevýhodou je, že lze specifikovat pouze některé množiny parametrů, které budou otestovány. Proto např. u techniky *náhodný les* je nutné zadat přesný počet stromů, nikoliv množinu čísel specifikují možný počty stromů. Avšak vytvoření celkového příkazu je snadné.

Jazyk Python, knihovna *Scikit-learn*, disponuje dříve popsanou třídou `GridSearchCV`. Ta v kombinaci s třídou `Pipeline` umožňuje sekvenčně definovat jednotlivé kroky předzpracování a následnou klasifikaci dat. Jednotlivé kroky `Pipeline` jsou tvořeny dvojicí (název kroku, konkrétní objekt). Testované množiny parametrů jsou pak uloženy v slovníkové struktuře tvaru `{"nazev kroku__název atributu objektu": [možné hodnoty]}`, která je předána atributu `param_grid` třídy `GridSearchCV`. Výsledkem této funkce je nejlepší model stejně jako u funkce `train()` v jazyce R. Jak je možné vidět, je definice příkazu složitější, avšak u každého objektu je možné specifikovat jeho všechny dostupné atributy.

6.5 Porovnání z hlediska ověření kvality modelů

Po získání nových hodnot je důležité zjistit, zda model predikuje výsledná data správně. V dříve popsaných případových studiích byly použité jako metriky střední kvadratická odchylka, skóre přesnosti a konfúzní matice.

Pro výpočet střední kvadratické odchylky v jazyce R slouží funkce `mse()` z knihovny *Metrics*. Konfúzní matice a skóre přesnosti je možné získat funkcí `confusionMatrix()` z knihovny *caret*.

Oproti tomu jsou všechny metriky klasifikátorů v jazyce Python dostupné opět v knihovně *Scikit-learn*, konkrétně ve jmenném prostoru `sklearn.metrics`.

6.6 Porovnání z hlediska podpory paralelizace výpočtů

V stále pokračujícím trendu generování více a více dat je čím dál tím větší nutností využití paralelizace při jejich zpracování a dolování. Důležité je zmínit, že rychlost zpracování vždy závisí na velikosti dat, dostupné paměti a výpočetním výkonu počítače, na kterém bude program spuštěn.

V jazyku R je pro paralelizaci výhodné použít knihovnu *parallel*. Tato knihovna umožňuje vytvořit tzv. cluster, což lze popsat jako kolekci "workerů", kteří budou paralelně zpracovávat dané operace.

V jazyce Python, v knihovně *Scikit-learn*, je implementováno uživatelsky přívětivé rozhraní pro podporu paralelizace. U většiny objektů je možné specifikovat parametrem `n_jobs` počet jader procesoru, které budou použity pro danou operaci. Pokud je zadaná hodnota -1, všechny jádra procesoru jsou využitá. Ukázku podpory paralelizace při vytvoření objektu pro aplikaci logické regrese je možné vidět na následujícím kódu.

Při realizaci případových studiích byla paralelizace použita pouze u modelů v jazyce Python.

```
# import knihovny
from sklearn.linear_model import LogisticRegression

#vytvoreni modelu s podporou paralelizace
model = LogisticRegression(n_jobs=-1)
```

Zdrojový kód 6.5: Použití paralelizace v knihovně Scikit-learn.

6.7 Porovnání z hlediska propojení jazyků

Jak je možné vidět v předchozích sekcích, ne všechny funkce a třídy jsou dostupné pro oba jazyky. Výhodou ovšem je, že tyto jazyky lze mezi sebou propojit a tím pádem mít přístup k požadovaným funkcím a třídám z toho druhého jazyku.

V jazyce Python je dostupná knihovna s názvem *rpy2*, která zpřístupňuje funkce jazyka R do prostředí Python. Objekty jazyka R jsou dostupné jako instance tříd implementovaných v jazyce Python s funkcemi jazyka R namapovanými na jejich rozhraní.

Pokud chceme v jazyce R používat jazyk Python, je možné využít knihovnu *rPython*. Při importování této knihovny je zpřístupněn objekt `python`, který slouží jako rozhraní pro volání funkcí jazyka Python.

Kapitola 7

Závěr

V první části práce jsem se zabýval problematikou dolování z dat. Na základě dostupné literatury jsem popsal jednotlivé fáze dolování. Jelikož se jedná o velmi obsáhlé téma, vybral jsem pouze části využité při realizaci případových studiích jako je čištění dat, jejich transformace, redukce dat a v nejdůležitější řadě techniky pro získávání znalostí.

V kapitole 4 jsem se zaměřil na popis programovacích jazyků Python, R a jejich dostupných prostředků pro dolování dat. Jak si lze všimnout, jazyk R disponuje velkým množstvím menších knihoven, na jejichž základech jsou vytvořeny další knihovny s rozšiřující funkcionalitou. Jedná se např. o knihovnu *caret* nebo *forecast*. Oproti tomu jazyk Python disponuje několika velkými knihovnami, které obsahují většinu prostředků pro dolování dat na jednom místě, např. knihovna *Scikit-learn*.

Použití těchto knihoven jsem demonstroval ve dvou případových studiích popsaných v kapitole 5. Cílem případových studií bylo v souladu se zadáním ilustrovat použití prostředků obou jazyků a knihoven, ne nutně provedení rozsáhlých experimentů pro nalezení a vytvoření nejvhodnějších atributů a nastavení hodnot parametrů. První případová studie byla z oblasti energetiky, konkrétně predikce hodnot výroby a spotřeby elektrické energie. Druhá případová studie se snaží předpovídat technikou klasifikování výsledný stav fotbalových zápasů. Při jejich realizaci byl kladen důraz na co největší přesnost modelů, proto bylo nutné data různě transformovat a hledat takové nastavení modelů, aby byly výsledky co nejpřesnější. Zhodnocení získaných výsledků jednotlivých případových studií je popsáné v kapitolách 5.1, resp. 5.2. Taktéž jsem uvedl možné rozšíření, které bych chtěl do budoucna u jednotlivých modelů případových studii aplikovat. U klasifikace fotbalových zápasů se jedná především o nalezení nebo vytvoření dalších atributů dat, na kterých je daný výsledek zápasu závislý. U predikce elektřiny se jedná o vytvoření dalších modelů pro predikci ceny elektřiny, které by byly závislé na predikovaných hodnotách těchto vytvořených modelů. Taktéž pro lepší porovnání výsledků práce s časovými řadami byla tato úloha zrealizovaná i za pomocí neuronové sítě.

V poslední kapitole 6 dochází k porovnání obou programovacích jazyků v souvislosti jejich použití v jednotlivých fázích dolování. Pro každou fázi dolování jsou zde představeny prostředky, které je možné použít na její provedení v konkrétním jazyce. Jelikož rok od roku roste velikost zpracovávaných dat, jsou zde popsány techniky využití paralelizace v jednotlivých programovacích jazycích. Taktéž je zde ukázána možnost propojení obou jazyků tak, aby bylo možné využít prostředky druhého jazyka.

Jednotlivé úkoly zadání jsem se snažil popsat co nejnázorněji tak, aby byly pochopitelné i pro člověka, který se touto problematikou nezabývá. Myslím si, že se mi to podařilo a tím jsem splnil zadání diplomové práce.

Literatura

[1] *Calculating standard deviation step by step.* [Online; navštíveno 16.12.2016].
URL https://www.khanacademy.org/math/probability/data-distributions-a1/summarizing-spread-distributions/a/calculating-standard-deviation-step-by-step

[2] Asch, V. V.: *Macro- and micro-averaged evaluation measure.* [Online; navštíveno 16.4.2017].
URL http://www.clips.uantwerpen.be/~vincent/pdf/microaverage.pdf

[3] Breiman, L.: *RANDOM FORESTS.* [Online; navštíveno 10.5.2017].
URL http://www.stat.berkeley.edu/~breiman/RandomForests

[4] Brownlee, J.: *Classification Accuracy is Not Enough: More Performance Measures You Can Use.* [Online; navštíveno 16.12.2016].
URL http://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/

[5] Han, J.; Pei, J.; Kamber, M.: *Data Mining, Southeast Asia Edition.* The Morgan Kaufmann Series in Data Management Systems, Elsevier Science, 2006, ISBN 9780080475585.

[6] Ing. Roman DANEL, P.: *Predikce časové řady pomocí autoregresního modelu.* [Online; navštíveno 16.2.2017].
URL http://homel.vsb.cz/~dan11/publikace/Danel_Autoregresni_model_predikce_casovych_rad.pdf

[7] Kantardzic, M.: *Data Mining: Concepts, Models, Methods, and Algorithms.* John Wiley & Sons, 2011, ISBN 9780470890455.

[8] Li, X.-B.: A Bayesian Approach for Estimating and Replacing Missing Categorical Data. *J. Data and Information Quality*, ročník 1, č. 1, Červen 2009: s. 3:1–3:11, ISSN 1936-1955, doi:10.1145/1515693.1515695.

[9] Lund, A.: *Measures of Spread.* [Online; navštíveno 16.12.2016].
URL https://statistics.laerd.com/statistical-guides/measures-of-spread-range-quartiles.php

[10] Marr, B.: *Big Data: 20 Mind-Boggling Facts Everyone Must Read.* [Online; navštíveno 16.12.2016].
URL http://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/

[11] Piatetsky, G.: *Four main languages for Analytics, Data Mining, Data Science.* [Online; navštíveno 16.12.2016].
URL http://www.kdnuggets.com/2014/08/four-main-languages-analytics-data-mining-data-science.html

[12] Sami Ayramo, T. K.: *Introduction to partitioning-based clustering methods with a robust example.* [Online; navštíveno 16.12.2016].
URL http://users.jyu.fi/~samiayr/pdf/introtoclustering_report.pdf

[13] Sayad, D. S.: *Binning.* [Online; navštíveno 16.12.2016].
URL http://www.saedsayad.com/binning.htm

[14] Sayad, D. S.: *K Nearest Neighbors - Classification.* [Online; navštíveno 16.12.2016].
URL http://www.saedsayad.com/k_nearest_neighbors.htm

[15] Sen, A.; Srivastava, M.: *Regression Analysis: Theory, Methods, and Applications.* Springer Texts in Statistics, Springer New York, 2012, ISBN 9781461244707.

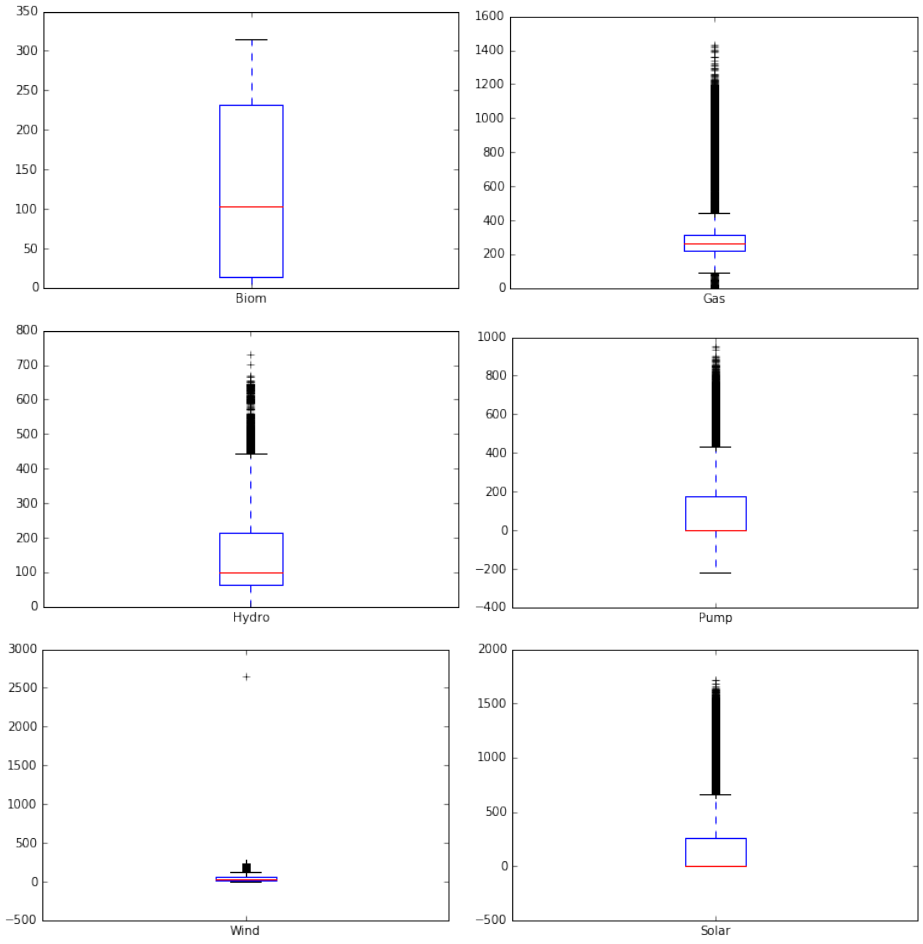
[16] Vettigli, G.: Practical Data Mining with Python. 2011, [Online; navštíveno 16.12.2016].
URL http://guidetodatamining.com/

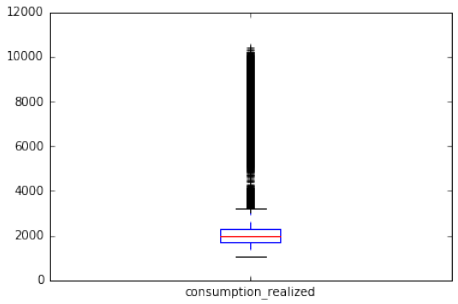
Přílohy

Příloha A

Predikce výroby/spotřeby elektřiny

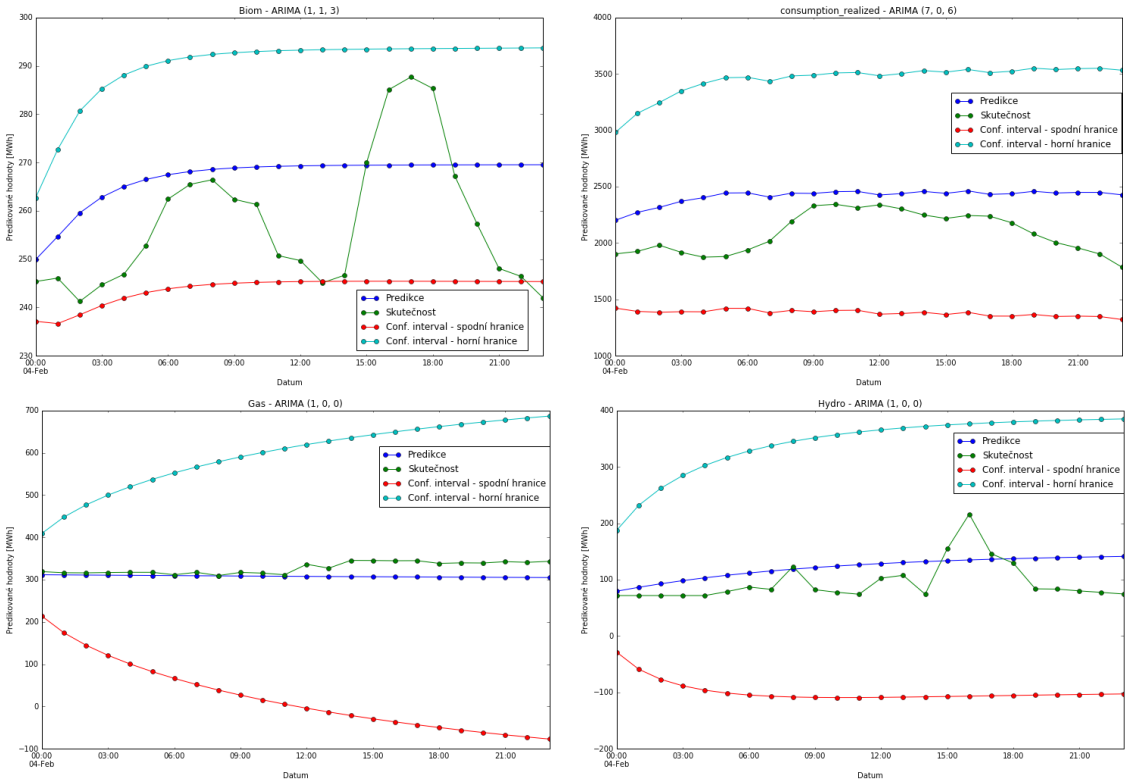
A.1 Boxplot grafy pro zdroje a spotřebu energie

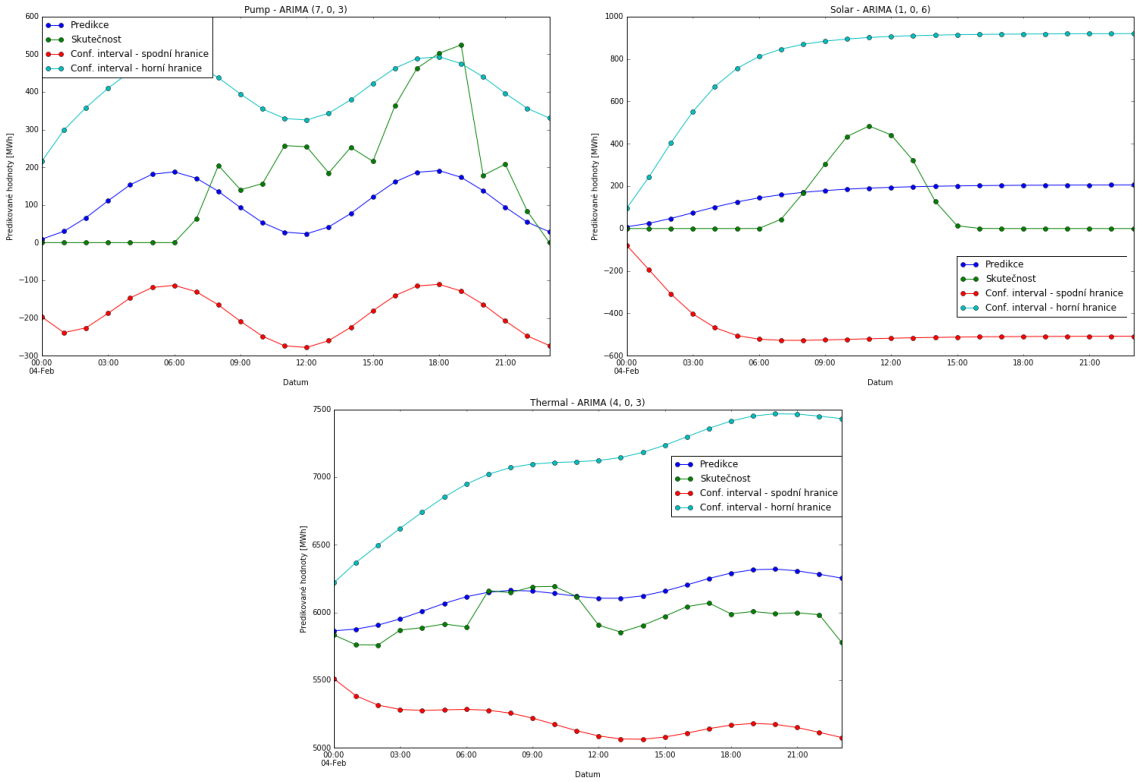




Obrázek A.1.1: Krabicové grafy zobrazující rozložení hodnot jednotlivých časových řad pro výrobu a spotřebu energie.

A.2 Predikované hodnoty pomocí ARIMA modelu v Python





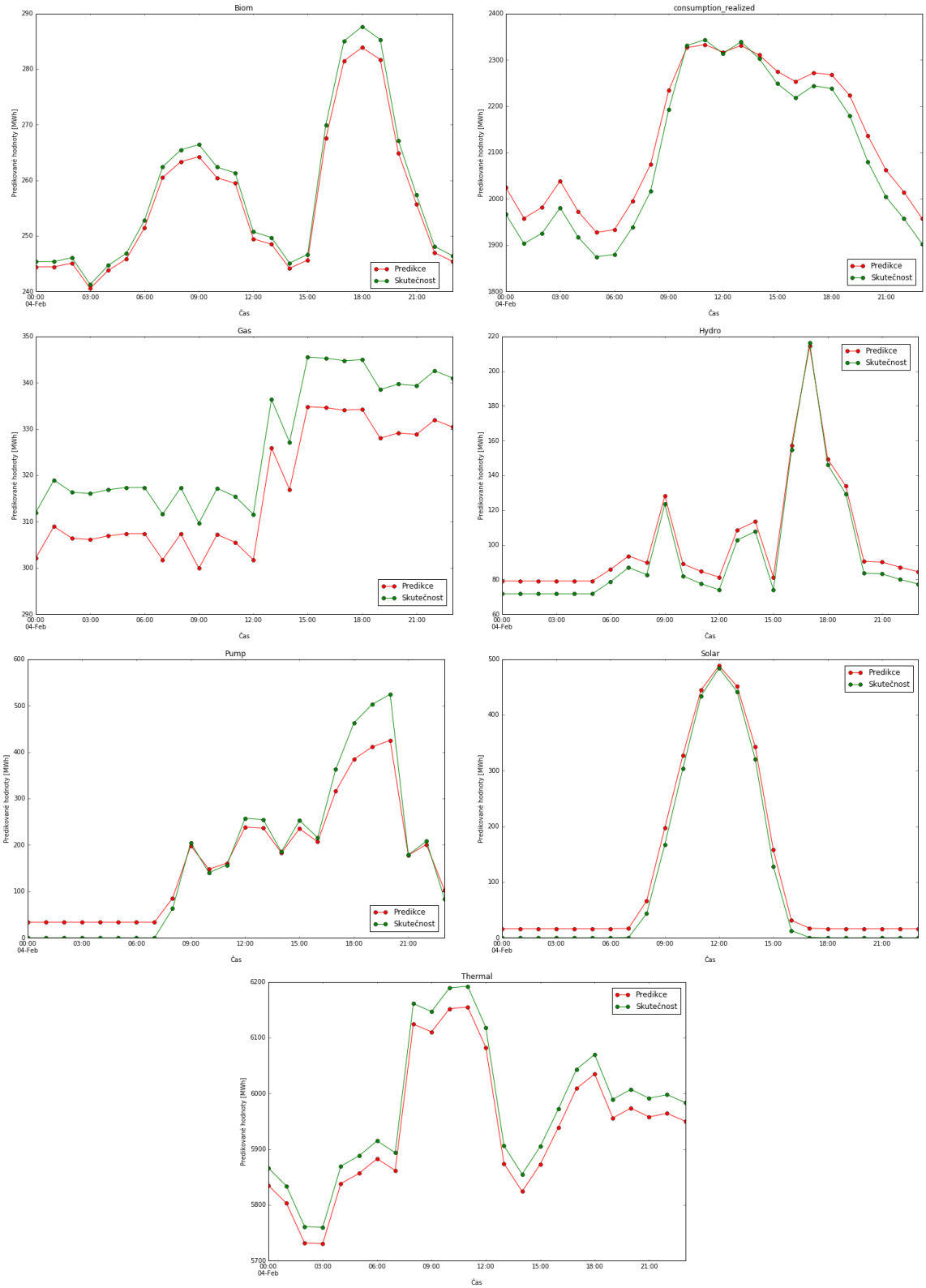
Obrázek A.2.1: Predikované hodnoty pro jednotlivé zdroje energie a pro její spotřebu.

A.3 Vrstvy modelu neuronových sítí

	Vrstvy
Nuclear	LSTM(20, return_sequences=False), Dense(20)
Biom	LSTM(50, return_sequences=False), Dense(50)
Consum	LSTM(80, return_sequences=False), Dense(150)
Gas	LSTM(80, return_sequences=False), Dense(100)
Hydro	-
Pump	LSTM(10, return_sequences=False), Dense(40)
Solar	LSTM(50, return_sequences=False), Dense(70)
Thermal	LSTM(60, return_sequences=False), Dense(50)
Wind	LSTM(70, return_sequences=False), Dense(60)

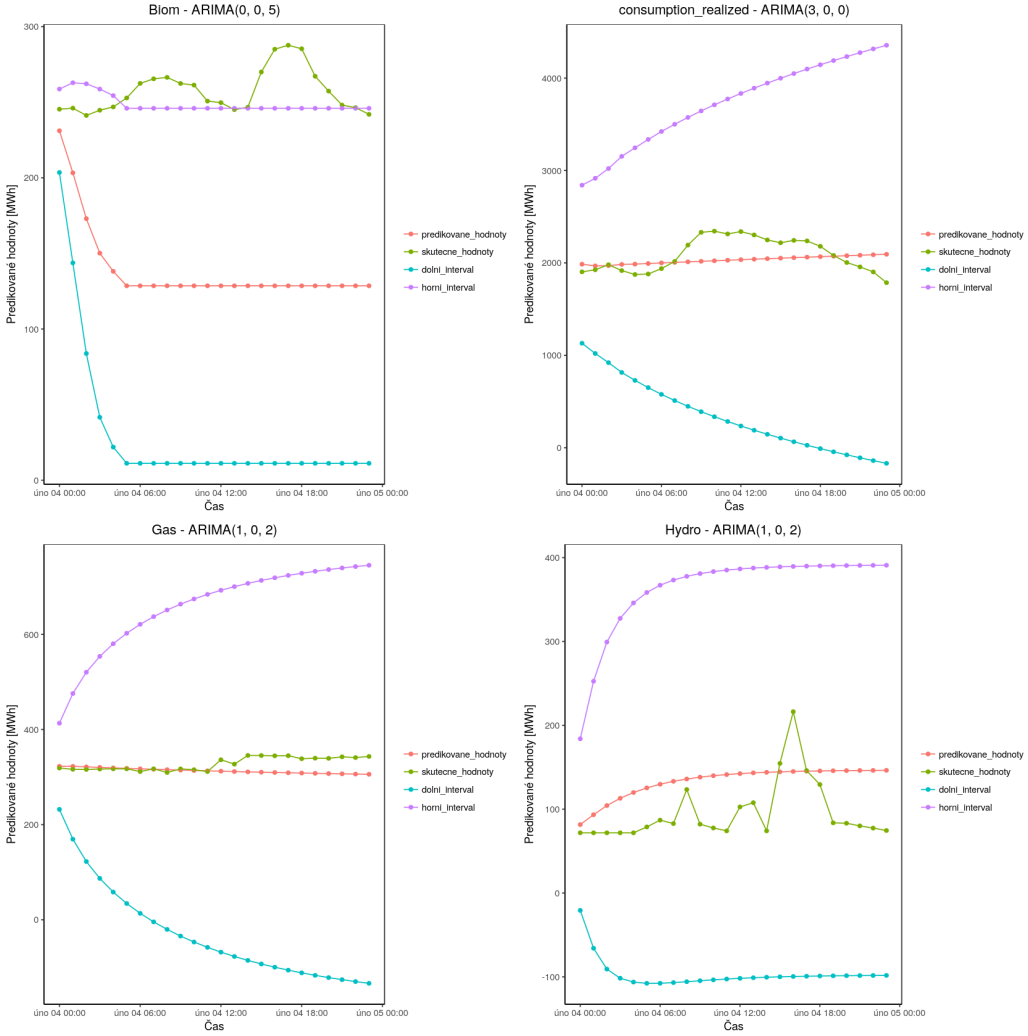
Tabulka A.1: Získané vhodné vrstvy neuronových sítí pro jednotlivé časové řady.

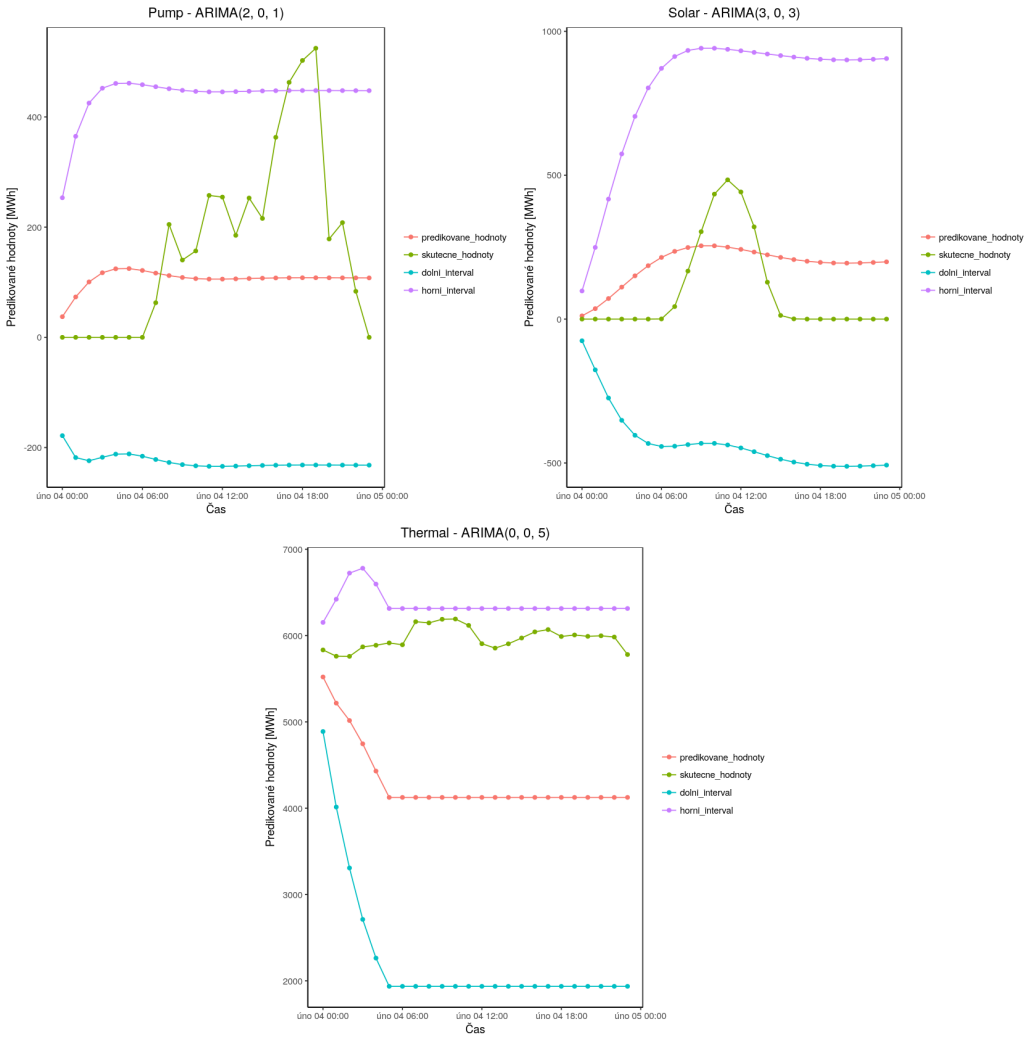
A.4 Predikované hodnoty pomocí neuronových sítí



Obrázek A.4.1: Predikované hodnoty výroby a spotřeby energie pomocí neuronových sítí.

A.5 Predikované hodnoty pomocí ARIMA modelu v R





Obrázek A.5.1: Výsledná predikce pro zbývající zdroje energie a její spotřebu v jazyce R.

Příloha B

Obsah CD

Součástí této práce je datový nosič s následujícím obsahem:

- **src** – adresář obsahující jupyter notebooky se všemi zdrojovými kódy
- **data** – adresář obsahující všechna zdrojová data pro případové studie
- **readme.txt** – soubor s informacemi o aplikace
- **thesis** – adresář obsahující zdrojové soubory textové části
- **thesis.pdf** – zpráva v elektronické podobě