TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Analýza naučené neuronové sítě v jazyku R

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Michal Nguyen**
Vedoucí práce: Ing. Vratislav Žabka, Ph.D.

Liberec 2019

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Analysing Neural Network Performance After Training

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Michal Nguyen**
Supervisor: Ing. Vratislav Žabka, Ph.D.

Liberec 2019

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Analýza naučené neuronové sítě v jazyku R

Jméno a příjmení: **Michal Nguyen**
Osobní číslo: M15000046
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav mechatroniky a technické informatiky
Akademický rok: **2018/2019**

Zásady pro vypracování:

1. Seznamte se s programováním v jazyku R a s prací v Rstudiu. Naučte se vytvářet náhodné časové řady, načítat a zobrazovat reálné časové řady pomocí prostředků jazyka R. Naučte se používat programové balíky "neuralnet", "neuralNetTools" a "shiny".

2. S pomocí několika uměle vytvořených časových řad s různou úrovní šumu a několika typů závislostí (lineární, kvadratická, lomená) připravte výslednou časovou řadu. Naprogramujte rekonstrukci výsledné časové řady pomocí alespoň dvou typů lineárních modelů a neuronové sítě. S pomocí Garsenova a Oldenova algoritmu analyzujte naučenou neuronovou síť a porovnejte získané parametry s výstupy lineárních modelů.

3. Naprogramujte celý proces pro jednoduchou opakovatelnost s proměnnými vstupními hodnotami úrovně šumu pro jednotlivé vstupní řady, pro různé koeficienty závislostí vstupních dat, pro různou přesnost výpočtu neuronové sítě a pro různou strukturu neuronové sítě. S využitím výše zmíněných balíků vytvořte unikátní webovou aplikaci umožňující uživateli opakovat celý proces učení a vyhodnocování výsledků.

4. S připravenou aplikací analyzujte a porovnejte možnosti lineárních modelů a neuronové sítě na rekonstrukci výsledné časové řady v závislosti na různých úrovních vstupního šumu a struktury neuronové sítě.

5. Celý postup aplikujte na systém reálných dat z oblasti hydrologie a využijte získané znalosti při analýze časových řad srážkových úhrnů, velikosti průtoku, čerpání z vrtu a výšky hladiny podzemní vody ve vrtu.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30–40 stran
Forma zpracování práce:	tištěná/elektronická

Seznam odborné literatury:

[1] Eva Volná, Neuronové sítě 1. 2008 Ostravská univerzita v Ostravě.

[2] C.W. Dawson, and R.L. Wilby, Hydrological modelling using artificial neural networks. Progress in Physical Geography 25,1 (2001) pp. 80–108.

[3] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2017). shiny: Web Application Framework for R. R package version 1.0.4. https://CRAN.R-project.org.

Vedoucí práce:	Ing. Vratislav Žabka, Ph.D. Ústav mechatroniky a technické informatiky
Datum zadání práce:	10. října 2018
Předpokládaný termín odevzdání:	30. dubna 2019

L. S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci 10. října 2018

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že texty tištěné verze práce a elektronické verze práce vložené do IS STAG se shodují.

29. 4. 2019 Michal Nguyen

Poděkování

Rád bych tímto poděkoval vedoucímu této práce za profesionální přístup a cenné rady, které byli přínosem při vytváření této práce. Dále bych rád poděkoval Haně Nguyenové, která byla nápomocná při úpravě textu v souladu s gramatickými pravidly.

Abstrakt

Cílem této práce je vytvoření nástroje pro analýzu neuronové sítě, tímto nástrojem se stala webová aplikace, která byla vytvořena v rámci této práce. Tato analýza se zakládá na myšlence, že některé ze vstupních proměnných lze z výpočtů vynechat a tím zefektivnit výpočty neuronové sítě při rekonstrukci výsledné řady. Analýza využívá dvou algoritmů, které určují důležitost jednotlivých proměnných. Na těchto dvou algoritmech zkoumáme, jestli je možné jejich využití na reálných datech, která jsou zatížená šumem. V první části jsou popsány jednotlivé metody pro zjišťování důležitosti jednotlivých vstupních proměnných, které jsou použity v této práci. Druhá část popisuje webovou aplikaci, která využívá těchto metod a umožňuje uživateli měnit parametry, které ovlivňují výpočet neuronové sítě a následně vizualizuje jednotlivé výstupu z těchto metod. Závěrem této práce je ukázka implementace těchto metod na reálných datech z oblasti hydrologie a vyhodnocení získaných výsledků.

Klíčová slova:

neuronová síť, programovací jazyk R, Shiny, RStudio, Garsonův algoritmus, Oldenův algoritmus, lineární model, korelace, odchylka, analýza neuronové sítě

Abstract

The aim of this work is to create a tool for neural network analysis, the tool had become a web application that was created within this work. This analysis is based on the idea that some of the input variables can be omitted from the calculations, thus making the neural network calculations more effective in reconstructing the resulting series. The analysis uses two algorithms that determine the importance of each variable. On these two algorithms, we examine whether their use on real data that are marred by noise is possible. In the first part, each method for determining the importance of individual input variables that are used in this work are described. The second part describes a web application that uses these methods and allows the user to change parameters that affect neural network calculation and then visualize the individual outputs from these methods. The conclusion of this work is an example of the implementation of these methods on real data from the field of hydrology and the evaluation of obtained results.

Key words:

neural network, programming language R, Shiny, RStudio, Garson algorithm, Olden algorithm, linear model, correlation, deviation, analysis of neural network

Obsah

Úvod....15
1 Teoretická část....16
1.1 Tvorba webové aplikace s pomocí jazyka R....16
1.1.1 Jazyk R....16
1.1.2 RStudio....16
1.1.3 Package Shiny....17
1.2 Analýza přesnosti výpočtů....18
1.2.1 Odchylka....18
1.2.2 Korelace....18
1.3 Výpočetní modely....18
1.3.1 Lineární modely....18
1.3.2 Neuronová síť....19
1.3.3 Analýza neuronové sítě....21
1.3.4 Garsonův algoritmus....21
1.3.5 Oldenův algoritmus....21
2 Praktická část....22
2.1 Popis postupu....22
2.1.1 Práce s číselnými řadami....23
2.1.2 Grafické uživatelské prostředí....24
2.1.3 Serverová část....27
2.1.4 Analýza na serverové části....31
2.1.5 Webová aplikace....33
2.2 Práce s reálnými daty....36
2.3 Vyhodnocení naměřených dat....36
2.3.1 Umělá data – lineární závislost....37
2.3.2 Umělá data – polynomiální závislost....38
2.3.3 Umělá data – lomená závislost....39
2.3.4 Reálná data....40
Závěr....43
Citace....44

Seznam ilustrací

Obr 1: Metoda nejmenších čtverců.....19
Obr 2: Schéma neuronové sítě.....20
Obr 3: Názorná ukázka jednotlivých dat v datasetech.....23
Obr 4: interaktivní posuvník.....25
Obr 5: Zadávací číselné pole.....25
Obr 6: RadioButton.....25
Obr 7: Přepínací panely.....26
Obr 8: tableOutput.....27
Obr 9: dataTableOutput.....27
Obr 10: Graf neuronové sítě s jednou skrytou vrstvou a šesti neurony.....28
Obr 11: Funkce vytváření formulace lineárních modelů.....30
Obr 12: Graf neuronové sítě s jednou skrytou vrstvou a osmi neurony.....31
Obr 13: Odchylky lineárního modelu.....32
Obr 14: Ovládací panel webové aplikace.....33
Obr 15: Výsledné odchylky jednoduchého lineárního modelu.....34
Obr 16: Výsledné odchylky neuronové sítě.....34
Obr 17: Grafické znázornění výpočtů garsonova a oldenova algoritmu.....35
Obr 18: Tabulka koeficientů bez šumu (lineární závislost).....37
Obr 19: Tabulka koeficientů se šumem 0,4 (lineární závislost).....37
Obr 20: Tabulka koeficientů se šumem 1 (lineární závislost).....37
Obr 20: Tabulka koeficientů bez šumu (polynomiální závislost).....38
Obr 21: Tabulka koeficientů se šumem 0,4 (polynomiální závislost).....38
Obr 22: Tabulka koeficientů se šumem 1 (polynomiální závislost).....38
Obr 23: Tabulka koeficientů bez šumu (lomená závislost).....39
Obr 24: Tabulka koeficientů se šumem 0,4 (lomená závislost).....39
Obr 25: Tabulka koeficientů se šumem 1 (lomená závislost).....39
Obr 26: Graf reálných surových dat.....40
Obr 27: Graf reálných dat upravených klouzavým průměrem a posunutím.....40
Obr 28: Graf koeficientů - oldenův algoritmus.....41
Obr 29: Graf koeficientů - garsonův algoritmus.....41
Obr 30: Graf koeficientů - lineární model.....41
Obr 31: Graf koeficientů - garsonův algoritmus.....42
Obr 32: Graf koeficientů - oldenův algoritmus.....42
Obr 33: Graf koeficientů - lineární model.....42

Úvod

Hlavní motivací této práce je využití programovacího jazyka R ve vývojovém prostředí Rstudia pro statistickou analýzu neuronové sítě. Základním předpokladem pro vypracování této práce je zvládnutí vytváření náhodných časových řad, jejich načítání a následná vizualizace. Pro práci s neuronovými sítěmi a vytvoření webové aplikace bylo nutné se naučit využívat balíky „neuralnet“, „neuralNetTools“ a „shiny“.

Díky schopnosti vytváření časových řad jsme schopni vytvořit několik časových řad s různou úrovní šumu a připravit výslednou časovou řadu s různým typem závislosti (lineární, polynomiální, lomená), která slouží k naučení neuronové sítě a k výpočtu alespoň dvou lineárních modelů. Pomocí těchto matematických modelů je možné se pokusit o rekonstrukci výsledné řady. S pomocí Garsonova a Oldenova algoritmu (analýza neuronové sítě) je možné porovnávat jednotlivé lineární modely a neuronovou síť a porovnat parametry získané touto analýzou neuronové sítě s výstupy lineárních modelů.

Aby bylo možné celý tento postup jednoduše opakovat s různými vstupními hodnotami úrovně šumu pro jednotlivé vstupní řady, různými koeficienty závislostí vstupní dat, různou přesností neuronové sítě a strukturu neuronové sítě byla naprogramována webová aplikace, která slouží jako nástroj pro snadnější analýzu této problematiky. Celá tato webová aplikace byla vytvořená za pomoci výše zmíněných balíků. Tato aplikace umožňuje uživateli opakovat celý postup učení a vyhodnocování výsledků.

Pomocí této aplikace je možné analyzovat a porovnat možnosti lineárních modelů a neuronové sítě při rekonstrukci výsledné časové řady v závislosti na úrovni šumu a struktury neuronové sítě. Tuto analýzu je možné využít při rozhodování toho, jaký model je vhodnější využít pro daná vstupní data.

Stěžejním bodem této práce je aplikování této analýzy na reálných datech z oblasti hydrologie a porovnat jednotlivé výpočty analýz. Hlavními vstupními řadami jsou zde srážkový úhrn, velikost průtoku a čerpání z vrtu, kdy se snažíme zjistit jestli přímo ovlivňují hladinu podzemní vody ve vrtu.

Hlavním cílem této práce je vytvoření nástroje pro analýzu a porovnání lineárních modelů a neuronové sítě, který umožní určit při jak vysoké úrovni šumu jsou jednotlivé modely přesné, jaké vstupní řady ovlivňují výslednou řadu a který z těchto modelů je vhodnější pro využití na reálných datech.

1 Teoretická část

1.1 Tvorba webové aplikace s pomocí jazyka R

1.1.1 Jazyk R

Jedná se o programovací jazyk pro statistikou analýzu dat a následné grafické zobrazení. Jazyk R vychází z jazyka S, který je oproti jazyku R komerční. Díky tomuto faktu se více rozšířil mezi uživateli statistických jazyků. Jazyk R vychází z jazyka S, proto má lépe řešené objektově orientované programování než jiné statistické jazyky. Základem při práci s jazykem R je využívání balíčků, které díky lexikálním pravidlům snadno rozšiřují tento jazyk. Vzhledem k tomu, že se jedná o OpenSourcový jazyk, vznikla velká komunita uživatelů, kteří tento jazyk využívají a existuje přes 11 tisíc balíčků, které tento jazyk rozšiřují.

K největším výhodám, které tento jazyk nabízí je vizualizace dat, které lze využít při publikacích ve vědeckých kruzích. Zvládá generování statistických grafů, jak statických tak i dynamických. Tyto grafy mohou obsahovat i matematické symboly, kdy jiné jazyky toto nepodporují. Jazyk R je podobný jazyku Markdown a díky tomu je v něm možné vytváření dokumentace, podobně jako to zvládá Latex. Tento jazyk byl vydán v několika skriptovacích jazycích jako je Python, Perl, Ruby a C#.

Jedná se o interpretovaný jazyk, ke kterému se nejčastěji přistupuje přes příkazovou řádku nebo přes vývojové prostředí. Stejně jako Matlab, jazyk R podporuje maticové počty. Jeho datovými strukturami jsou vektory, matice, listy, datové matice, ale skaláry jsou zde prezentovány jako vektory o jednom prvku. Podporuje procedurální programování a pro některé funkce objektově orientované programování s generickými funkcemi. Mezi nejvyužívanější grafická vývojová prostředí jazyka R patří RKWard, RStudio, R Commander a rozšíření pro OpenOffice Calc R4Calc.

(R Documentation, 2017)

1.1.2 RStudio

Jedná se o vývojové prostředí jazyka R, které podporuje přímé provádění jednotlivých příkazů, jak už přes konzolový terminál nebo editor příkazů. Toto vývojové prostředí lze využívat podobně jako Octave, které je odlehčenou OpenSource alternativou k Matlabu podobně jako jazyk R k jazyku S. Výhodou oproti Matlabu je licence, kde základní vývojové prostředí je OpenSource pro nekomerční účely a celé je pod licencí AGPL v3.

Toto vývojové prostředí bylo vydáno 28. února 2011 Josephem J. Allairem, který je autorem programovacího jazyka ColdFusion. Prostředí je vyvíjeno v jazycích Java, C++ a JavaScript a je dostupné ve dvou verzích. První z nich je lokální, kdy program běží

na klientské straně a druhá je serverová, kdy se k prostředí přistupuje přes internetový prohlížeč.

(R Documentation, 2017)

1.1.3 Package Shiny

Jedná se o balíček jazyka R, rozšiřující jeho využití. Používá se k vytváření interaktivních webových aplikací. Jeho výhodou je, že stačí základní znalosti k vytvoření webové aplikace a není zapotřebí znalosti JavaScriptu pro tvoření interaktivních prvků aplikace.

Pro komunikaci mezi webovým prohlížečem a jazykem R se využívá balík hhtpuv. Ten poskytuje podporu na úrovni socketů a protokolů pro zpracování http požadavků přímo v prostředí R. Převážně se využívá jako základní prvek pro tvorbu dalších balíčků z důvodu snazší tvorby webových aplikací než přímo přes httpuv.

Pomocí knihovny Bootstrap3 (ve vyšších verzích) je vytvářen vizuální vzhled celé aplikace. Jedná se o knihovnu vytvořenou firmou Twitter. Tato knihovna je určena pro práci s webovými technologiemi JavaScript, CSS a HTML. Díky této knihovně lze jednoduše docílit responsního designu aplikace, který staví na principu „mobile first“, kdy se začíná vytvářet mobilní verze designu a z té přechází na verzi pro tablety a monitory s vyšším rozlišením a obrazovou úhlopříčkou.

Tento balík patří pod licenci GPLv3 a je zdarma pro nekomerční účely stejně jako je tomu u vývojového prostředí RStudio.

(Shiny from RStudio, 2017)

1.2 Analýza přesnosti výpočtů

Aby bylo možné porovnat správnost rekonstrukce výsledných řad lineárních modelů a neuronové sítě, využívá se odchylky a korelace, kde pomocí těchto dvou metod ověřujeme, jestli dopočítaná výsledná řada se příliš neliší od reálných výsledků.

1.2.1 Odchylka

Odchylka se využívá k určování podobnosti mezi typickými případy v souboru zkoumaných čísel. Pokud je hodnota odchylky malá, tak jsou si tyto zkoumané prvky navzájem podobné a naopak. V našem případě tuto odchylku počítáme jako sumu absolutních rozdílů mezi námi analyticky vypočítanými hodnotami a výslednými hodnotami vypočítanými jednotlivými modely. V této práci využíváme vlastní funkci napsanou v jazyce R.

(Chára, 2007)

1.2.2 Korelace

Korelace se používá k určení vztahu mezi procesy nebo veličinami. Míru korelace vyjadřuje korelační koeficient, který nabývá hodnot od -1 do +1. Pokud se korelační koeficient blíží k -1, jedná se o nepřímou závislost (antikorelaci). To znamená, že pokud se hodnoty v první skupině zvětšují, tím víc se zmenšují hodnoty v druhé. Naopak pokud se blíží k +1, tak se jedná o přímou závislost. Pokud se koeficient rovná 0, tak to značí to, že mezi hodnotami není zjištěna lineární závislost. To ale neznamená, že veličiny na sobě nezávisí, jen nelze jejich vztah vyjádřit lineární funkcí. V našem případě využíváme funkci „cor“ v Rstudiu.

(Chára, 2017)

1.3 Výpočetní modely

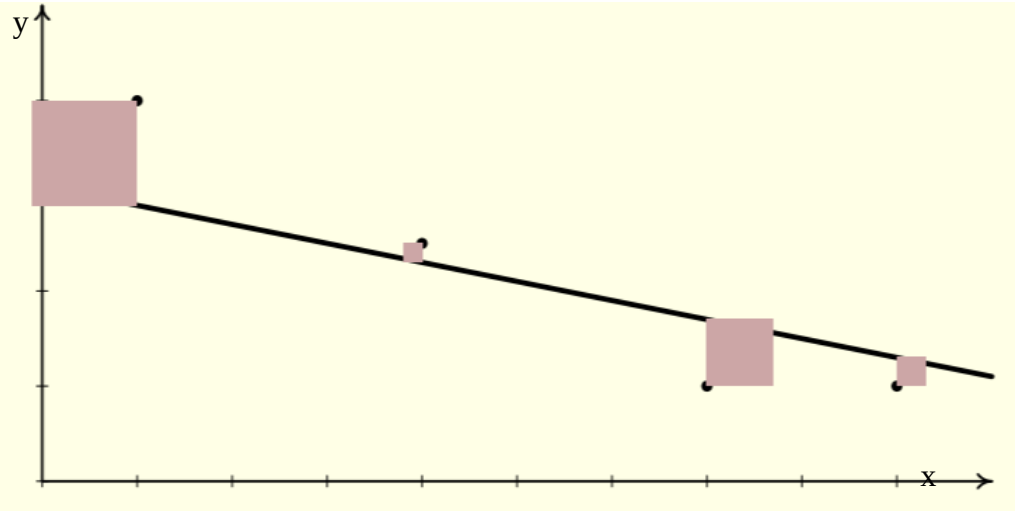
1.3.1 Lineární modely

Lineární modely jsou matematické modely, které se též dají nazvat lineární regresí. Jedná se o statistickou metodu, která umožňuje odhadnout hodnotu náhodné veličiny na základně její závislosti. Tato metoda patří mezi nejpoužívanější v oblasti entropické a aplikované vědy.

Základem lineární regrese je aproximace dat přímkou, která popisuje vztah mezi vstupními a výslednými hodnotami. Přičemž předpokládá, že tyto hodnoty jsou zatíženy náhodnými chybami.

(Řehák, Brom, 2007)

Základem lineární regrese je metoda nejmenších čtverců. Ta se snaží aproximovat hodnoty takovým způsobem, že součet všech čtverců je minimální.



Obr 1: Metoda nejmenších čtverců

V jazyce R se využívá funkce lm, která nám umožňuje zkoumání odchylky a kovariance daného souboru dat. Vstupními parametry jsou matematický předpis funkce, která určuje o jak složitou aproximaci dat se snažíme dosáhnout (lineární, polynomiální apod) a vstupní data ve vektorové či maticové reprezentaci. Díky této funkci můžeme zkoumat vztah vstupních a výstupních hodnot.

(Řehák, Brom, 2017)

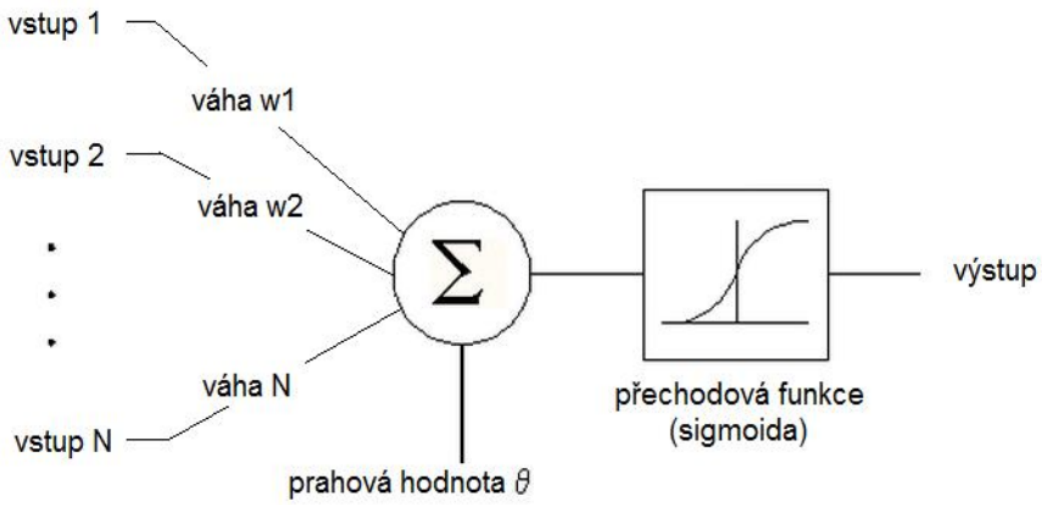
1.3.2 Neuronová síť

Jedná se o matematický model, který vychází z chování mozkových buněk. Hlavními oblastmi využití neuronových sítí jsou aproximace, klasifikace a predikce.

Základním prvkem jsou zde neurony (perceptrony), které jsou v síti spojeny vazbami, které se nazývají synapsí. Jedná se o vstupy do neuronu ze vstupní vrstvy (vnějšího okolí) nebo výstupy z neuronů z jiné vrstvy (u vícevrstvých sítí). Všechny tyto synapse jsou rozšířené o váhu těchto vstupů. Jednotlivé neurony obsahují prahovou hodnotu (potenciál neuronu), která určuje, jestli je neuron nabuzen. Pokud je potenciál neuronu dostatečně velký, neuron vyšle signál do aktivační (přechodové) funkce.

Potenciál neuronu:

$$P = W_1 * X_1 + W_2 * X_2 + ... + W_n * X_n$$



Obr 2: Schéma neuronové sítě

Aktivační funkce nám moduluje výstup z nabuzeného neuronu. Jedná se o funkci, která se chová jako rozhodovací element, který rozděluje výstup neuronů do dvou stavů (převážně do 0 a 1). Výběr této funkce závisí na úloze, kterou řešíme, zkušenostech s podobnými úlohami nebo je čistě experimentální, kdy se vybírá funkce lineární či nelineární.

(Shiffman, 2016)

Nejpoužívanější aktivační funkce:

- Jednotkový skok
- Funkce Signum
- Lineární funkce
- Hyperbolický tangent
- Sigmoidální funkce
- Saturační přenosová funkce

Druhy neuronových sítí:

- Vícevrsvé neuronové sítě
- Hopfieldovy sítě
- Samoorganizující se sítě
- Radialní báze
- ostatní modely

V této práci využíváme práci s vícevrstvými neuronovými sítěmi. Jedná se o nejrozšířenější a nejpoužívanější neuronové sítě. Neuronová sít je tvořena vstupní vrstvou, skrytými vrstvami a výstupní vrstvou. Tuto neuronová síť lze využít ve všech třech oblastech.

(Fritsch, Guenther, 2016)

1.3.3 Analýza neuronové sítě

1.3.4 Garsonův algoritmus

Tato funkce využívá Garsonův algoritmus. Tento algoritmus určuje relativní důležitost mezi vstupními hodnotami pro jednotlivé proměnné jako rozdělení vah. Důležitost každé proměnné je určena všemi vahami mezi vrstvami sítě. Všechny váhy každé vstupní proměnné, které projdou skrytou vrstvou k výsledné hodnotě, jsou zaznamenány, a toto se opakuje pro všechny ostatní neurony, dokud nejsou získány všechny váhy jednotlivých vstupních proměnných. Jednotlivé hodnoty jsou zaznamenány pro každou výstupní proměnnou a popisují vztah mezi neurony v modelu. Výsledky ukazují relativní důležitost jako absolutní hodnotu mezi nulou a jedničkou. Tento algoritmus lze použít jen na sítě s jednou skrytou vrstvou.

(NeuralNetTools: Visualization and Analysis Tools for Neural Networks, 2018)

1.3.5 Oldenův algoritmus

Funkce „olden“ je alternativní a flexibilnější způsob k vyhodnocení důležitosti jednotlivých proměnných. Tato funkce vypočítá důležitost jako součin vstupů a výstupů z jednotlivých neuronu a sum napříč všemi skrytými neurony. Výhoda tohoto postupu je, že si uchovává podíl všech propojení neuronů v celém rozsahu na rozdíl od Garsonova algoritmu, který využívá pouze absolutních hodnot. Oldenův algoritmus je schopný vyhodnotit neuronové sítě s více skrytými vrstvami. Důležitosti každé proměnné je souhrn sum součinů všech vah.

(NeuralNetTools: Visualization and Analysis Tools for Neural Networks, 2018)

2 Praktická část

2.1 Popis postupu

Hlavní částí této práce je analýza jednotlivých modelů a vlivu typu závislosti a zašumění dat na jejich správnosti výpočtu. Pro usnadnění zkoumání jsme vytvořili webovou aplikaci v prostředí RStudia s podporou jazyka R a balíčku Shiny. Pomocí této kombinace jsme schopni využívat výpočty z jazyka R a možnost vizualizace naměřených hodnot ve webovém prostředí.

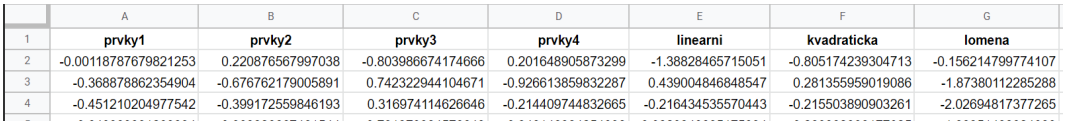
Premisou pro toto zkoumání je určení dvou faktů. Jedním z faktů je správnost určení jednotlivých vlastností v závislosti na zašuměných datech. Druhým faktem je, jak jednotlivé modely zvládají aproximovat a odhadnout vstupní a výstupní data, kdy tyto data jsou též zašuměná.

Výsledné výpočty těchto modelů podrobujeme analýzou. Analýza je rozdělena na tři typy. Prvním typem je určení, jak jsou modely odolné vůči míře zašumění, který je výpočtem součtem odchylek kontrolních výstupních hodnot od hodnot vypočítaných modely. Druhým typem je určení vztahu vstupních veličin pomocí korelace. V tomto případě se snažíme zjistit, jestli existuje vztah mezi vstupními hodnoty a vztahem vstupních veličin k výsledným. Třetím typem je určení důležitosti jednotlivých vstupních hodnot na základě garsonova a oldenova algoritmu.

2.1.1 Práce s číselnými řadami

Pro testování lineárních modelů a neuronové sítě bylo vytvořeno sedm datasetů uložených v souborech CSV, které načítáme do programu externě. Struktura každého datasetu je následující:

- 4 vstupní proměnné
- výsledná hodnota lineární funkce
- výsledná hodnota polynomiální funkce
- výsledná hodnota lomené funkce



	A	B	C	D	E	F	G
1	**prvky1**	**prvky2**	**prvky3**	**prvky4**	**linearni**	**kvadraticka**	**lomena**
2	-0.00118787679821253	0.220876567997038	-0.803986674174666	0.201648905873299	-1.38828465715051	-0.805174239304713	-0.156214799774107
3	-0.368878862354904	-0.676762179005891	0.742322944104671	-0.926613859832287	0.439004846848547	0.281355959019086	-1.87380112285288
4	-0.451210204977542	-0.399172559846193	0.316974114626646	-0.214409744832665	-0.216434535570443	-0.215503890903261	-2.02694817377265

Obr 3: Názorná ukázka jednotlivých dat v datasetech

Jednotlivé datasety počítaly výsledné hodnoty s různou mírou zašumění, kdy byl koeficient zašumění u prvního data setu nulový a u dalších se stupňoval. K zašumění byla použita funkce *runif* v jazyce R tak, že ke vstupním hodnotám byla přidána náhodná hodnota z intervalu od záporné hodnoty koeficientu zašumění až po kladnou hodnotu.

Tabulka s hodnotou šumu v daném datasetu							
Název	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6	Dataset7
Koeficient	0	0,5	0,8	1	1,5	3	5

Výhodou využití datasetů byla možnost zaznamenávat jednotlivé hodnoty různých výpočtů. Díky tomu bylo možné porovnávat výpočty lineárních modelů a neuronové sítě na stejných vstupních datech při různé konfiguraci parametrů těchto modelů.

Po zvolení ovládacích prvků webové aplikace bylo možné od využití datasetů upustit a tyto hodnoty se generovat při každém jednotlivém výpočtů těchto modelů. Tím, že jsou řady generovány pro každý výpočet zvlášť, je zaručené to, že analýzy jsou nezávislé na stejných vstupních hodnotách. Kdyby docházelo k nechtěné závislosti mezi jednotlivými hodnotami, toto řešení není náchylné na zkreslení v důsledku nechtěné korelace mezi těmito hodnotami.

2.1.2 Grafické uživatelské prostředí

Celý program a webová aplikace je tvořená ve vývojovém prostředí RStudia. Výsledný program je poté rozdělen na dvě části, kde první je logická část (serverová) a druhá je prezenční (userinterface). V jazyce R je to toto rozdělení řešeno pomocí komponent, které jsou přiřazeny proměnným *ui* a *server*. Při přeložení se proměnná *ui* stává webovou stránkou, která přistupuje k serverové části pomocí balíčku httpuv, která byla přeložena z proměnné *server.*

(Shiny from RStudio, 2017)

Pro tyto dvě proměnné lze externě definovat funkce, externí css styly a javascriptové funkce. Externí css styly se zde využívají přes globální proměnnou *appCss*, kdy se tato proměnná přidává k proměnné *ui* přes komponentu *inlineCSS,* která při přeložení programu přidá do hlavičky webové stránky tento css styl, který je přiřazen proměnné *appCss.*

Aby bylo možné využívat již definovaných javascriptových funkcí balíčku shiny, je nutné v proměnné *iu* zavolat komponentu *useShinyJS*, která zajišťuje, že se do webové aplikace naimportují tyto externě definované funkce a tím zajistí plnou funkčnost ovládacích prvky a jejich interaktivitu. Tuto komponentu je možné zavolat v jakékoliv části kódu, kde se definují ovládací prvky aplikace.

Celé definování grafického prostředí probíhá tak, že jednotlivé prvky jsou řadou vkládaných parametrů nadřazených prvků. Jedná se o pokročilou metodu javascriptové knihovny React, která je založená na principu, že vyšší komponentou je funkce, která má na vstupu komponentu a vrací jinou komponentu. Tomuto návrhovému vzoru se říká *Higher-Order Components*.

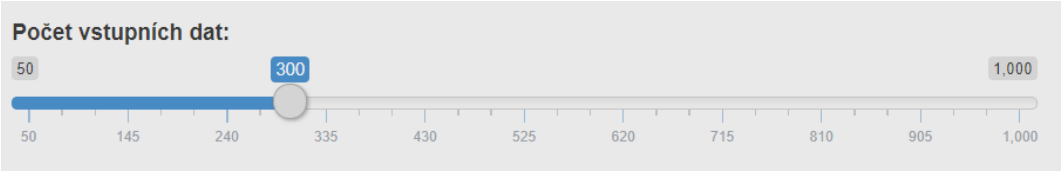
(Higher-Order Components, 2017)

V této aplikaci je hlavním prvkem *fluidPage*, což je obalovací prvek *div*, který má definovanou třídu css *content-fluid* v css stylech rozšíření bootstrap. Dalšími bloky, které jsou potomky hlavního prvku je *sidebarLayout* a *mainPanel*. Tyto dva prvky rozdělují obrazovku na dvě části, kdy se v *sidebarLayout* definují ovládací prvky aplikace a v *mainPanelu* pak prvky, které nám reprezentují naměřená data.

K ovládání aplikace využíváme tyto ovládací prvky:

- interaktivní posuvníky
- zadávací textové pole s možností přepínání mezi hodnotami
- radiobuttony
- tlačítka

Interaktivní posuvníky jsou prvky, které nám zobrazují předem definovaný interval, na kterém vybíráme hodnotu, kterou pak dále využíváme. Zde v jazyce R je lze definovat pomocí komponenty *sliderInput*, kde se jeho vlastnosti definují vstupními parametry. Parametry dané komplonenty jsou pojmenování, přes které lze k dané hodnotě přistupovat v serverové části, textový popis daného prvku, který se zobrazuje uživateli, poté se definuje minimální a maximální hodnota, počáteční hodnota a krok, po kterém se daný posuvník pohybuje. Níže je interaktivní posuvník, kde je definován interval od 50 do 1000 s krokem 1. Tento posuvník využíváme k určení počtu dat, která budou vygenerována.

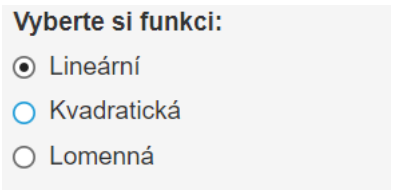


Obr 4: interaktivní posuvník

Dalším prvkem jsou zadávací pole, která se definují jako komponenta *numericInput*, kde parametry jsou pojmenování, popisek, minimální a maximální hodnota a krok, po kterém můžeme přidávat nebo ubírat z hodnoty. Na obrázku Zadávací číselné pole jsou vyobrazeny čtyři tato pole, kde intervalem jsou hodnoty od 0 do 5 s krokem 0,1. Tato pole nám slouží k určení koeficientu, jak jednotlivé vstupní parametry ovlivní výslednou hodnotu a poté je můžeme porovnat s výsledky jednotlivých algoritmů k určení důležitosti.

Obr 5: Zadávací číselné pole

Pro zvolení závislosti dat slouží prvek radiobutton, který je definován jako komponenta *radioButtons*, kde se definuje pojmenování, popis a jednolivé volby, které jsou ve vektoru, kde se definuje index (ten slouží i jako popisek) a hodnota, kterou při vybrání tento prvek bude mít. Na obrázku RadioButton je skupina radiobuttonu, které nám určují, s jakou závislostí dat chceme počítat.

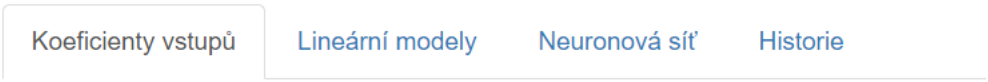


Obr 6: RadioButton

V druhé části webové aplikace, která se definuje do komponenty *mainPanel* definujeme prvky, které nám zobrazují dané výsledky výpočtů. Zde se využívá gridové členění, které umožňuje si prostor obalovacího prvku rozdělit na dílčí bloky, do kterých si definuje jeho potomky. Toto členění je definováno pomocí css stylů, které poskytuje spolupráce s css knihovnou bootstrap. Díky tomuto členění je zajištěná responsibilita celého vzhledu a je možné prvky jednoduše vkládat za sebe jak horizontálně, tak i vertikálně.

(Chang, 2017)

K lepší přehlednosti se využívají panely, které se chovají jako záložky. Kdy do jednotlivých panelů je možné definovat jiný obsah a přepínáním mezi nimi ho měnit. Zde se panel definuje komponentou *tabsetPanel*, kdy se jedná o obalovací prvek, do kterého se jako parametry vkládají jednotlivé panely komponentou *tabPanel*. Tímto je zajištěno zaskupinování těchto panelů a lze mezi nimi libovolně přepínat. U těchto panelů se definuje jejich popisek a další elementy, které jsou v tomto panelu obsahem. Například u tableOutput je jeho obsahem tabulka s přehledem vypočítaných důležitostí jednotlivými algoritmy a souhrnem s porovnáním těchto algoritmů mezi sebou.



Obr 7: Přepínací panely

Existuje několik možností, jak je možné v jazyce R vizualizovat data. V této aplikaci se využívá:

- dataTableOutput
- tableOutput
- plotOutput

Všechny tyto prvky jsou zde definovány jako komponenty, které jsou přiřazeny do jednotlivých obalovacích elementů. K jejich definování stačí pouze pojmenování, aby bylo možné k nim v serverové části přistupovat.

Prvek *dataTableOutput* je komponenta, která nám zobrazuje tabulku, ve které definujeme matici, která se má zobrazit. Oproti komponentě *tableOutput* zvládá i podmíněné formátování, která pomáhá z vizualizací důležitých dat. Díky této komponentě je možné vytvářet složité tabulky, které je možné stránkovat, filtrovat, vyhledávat v nich a řadit.

(Yihui Xie, 2016)

Jak je z dataTableOutput patrné, tak komponenta zvládá graficky znázornit, jak velký vliv mají jednotlivé vstupní proměnné na výsledné hodnoty. Díky tomu lze jednoduše a přehledně znázornit hodnoty, které jsou v naši práci důležité.

Jednoduchý lineární model

Vstupy	Koeficienty
vstup 1	1.00
vstup 2	1.00
vstup 3	4.00
vstup 4	0.10

Obr 8: tableOutput

Prehled koeficientů

	Garson	Olden	LM2	LM1
vstup 1	0.15	6.73	1	1
vstup 2	0.17	6.82	1	1
vstup 3	0.66	26.89	4	4
vstup 4	0.02	0.65	0.1	0.1

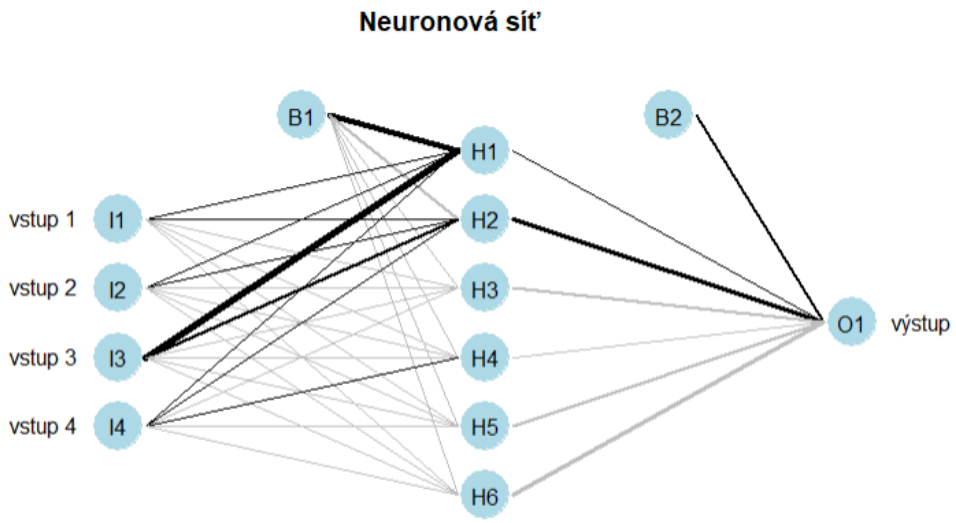
Obr 9: dataTableOutput

Za pomocí komponenty *plotOutput* je možné vytvářet a zobrazovat grafy. Funkcionalita této komponenty je určená na serverové části, kdy se podle pojmenování dané komponentě přiřadí funkce *renderPlot,* v jejímž těle je definováno, o jaký plot se bude jednat.

2.1.3 Serverová část

Serverová část programu je definována v komponentě *server*, které je přiřazená funkce se vstupními argumenty, což jsou vstupní a výstupní proměnné, které jsou definovány pojmenováními, která definovala jednotlivým komponentám grafického rozhraní. Díky těmto argumentům jsme schopni k jednotlivým prvkům přistupovat z kódu na serverové části.

Aplikace vytvořená v balíčku shiny je takzvaná *real-time* aplikace, což znamená, že v reálném čase reaguje na události, které se spouští s akcí uživatele nebo na základě časovače, který se obecně nastavuje na takt procesoru, a tím je událost vyvolávána s každým jeho taktem. Na tuto událost je nastavená i funkce, kterou máme přiřazenou v proměnné *server*.



Obr 10: Graf neuronové sítě s jednou skrytou vrstvou a šesti neurony

Aby bylo možné definovat proměnné, které nebudou měněny při každém spuštění funkce, je potřeba danou proměnnou definovat funkcí *reactiveValues*, kde si danou proměnou definujeme a deklarujeme na námi zvolenou hodnotu. Tento přístup nám umožňuje vytvořit z této proměnné globální, kterou je možné volat v celém programu.

Jak už bylo výše zmíněno, funkce která nám zajišťuje serverová část, je prováděná s každým taktem procesoru. Abychom tomuto zabránili, je možné využít funkci *observeEvent*, kde prvním parametrem je proměnná, která spouští danou funkci, kterou definujeme jako druhý parametr této funkce.

Pomocí vstupních argumentů funkce serveru, které nám předávají referenci na vstupní a výstupní prvky z grafického prostředí jsme schopni přistupovat k jejich hodnotám. To se provádí tak, že se daný argument chová jako vektor několik proměnných, ke kterým se lze přistupovat přes symbol „$“ a pojmenováním komponenty. Například *input$neurony*, kdy vstupním argumentem je proměnná *input* a pojmenování komponenty je *neurony*.

Ke komponentám, které nám zobrazují data se přistupuje stejným způsobem přes druhý vstupní argument serverové funkce. Ale aby bylo možné danou komponentu vykreslit, je nutné této komponentě definovat funkci, která danou komponentu vykreslí. Podle typu funkce určujeme, jak bude daná komponenta vypadat.

Typy funkcí pro vykreslení:

- renderTable (tabulka)
- renderDataTable (datová tabulka)
- renderPlot (graf)

Tyto funkce slouží k tomu, abychom byli schopni určit a definovat grafické zobrazení a jaká data se vykreslí.

Kromě tohoto ovládání komponent grafického rozhraní, je v této funkci zakomponována logika programu. Tato logika se skládá z generování číselných řad na základě vstupních hodnot jednotlivých komponent, poté z výpočtu jednotlivých matematických modelů (lineární modely, neuronová síť) a z následné analýzy výsledků těchto modelů.

Jak už bylo výše zmíněno , jsou pro každou kombinaci vstupů vygenerována nová data. Toto generování se provádí ve funkci *observeEvent*, když je v ovládacím panelu stisknuto tlačítko „Odeslat“. Jednotlivé vstupní proměnné jsou vygenerovány na základě zvolené délky pomocí funkce *runif*, která vytvoří vektor s délkou, která byla definováno komponentou posuvníku, s hodnotami v intervalu od -1 do 1. Poté se pro každý vektor vytvoří další vektor, který má stejnou délku a obsahuje zašumění pro daný vstup. Vektor zašumění se vytvoří pomocí námi definované funkce *createNoise*, kde vstupními parametry je délka a míra zašumění, kterou si též definujeme posuvníkem v ovládacím panelu.

Pro jednotlivé úlohy je vytvářen další vektor s hodnotami podle definovaného vztahu pro danou závislost. Typ úlohy je zde možné měnit podle zaškrtnutého radiobuttonu v ovládacím panelu. Zároveň tento vektor slouží jako kontrolní prvek, protože vstupem jsou nezašuměná data, a díky tomu je možné porovnávat správnost výsledku jednotlivých modelů.

Tyto úlohy jsou:

- lineární

  $$y=(x_1+s_1)*k_1+(x_2+s_2)*k_2+(x_3+s_3)*k_3+(x_4+s_4)*k_4$$

- polynomiální

  $$y=((x_1+s_1)*k_1)^4+((x_2+s_2)*k_2)^3+((x_3+s_3)*k_3)^2+((x_4+s_4)*k_4)$$

- lomená $y=\frac{(x_1+s_1)*k_1+(x_2+s_2)*k_2}{(x_3+s_3)*k_3+(x_4+s_4)*k_4}$

  x – vstupní proměnné

  k – koeficient důležitosti

  s – koeficient šumu

Aby bylo možné porovnávat důležitost jednotlivých vstupů, je ke každé vstupní proměnné přidáván koeficient důležitosti, který je volen v zadávacích polích ovládacího panelu. Tímto koeficientem je daná hodnota proměnné vynásobená. To nám zajišťuje kontrolní informaci o tom, jak daná proměnná ovlivňuje danou úlohu.

```
#Create simple formula
createForm1 <- function(mat,index){
  formula <- as.formula
  (
    mat[,index] ~ 0
    +mat[,1]
    +mat[,2]
    +mat[,3]
    +mat[,4]
  )
  return(formula)}
#Create multiple formula
createForm2 <- function(mat,index){
  formula <- as.formula
  (
    mat[,index] ~ 0
    +mat[,1]+mat[,2]+mat[,3]+mat[,4]
    +mat[,1]*mat[,2]+mat[,2]*mat[,3]+mat[,3]*mat[,3]+mat[,4]*mat[,3]
    +mat[,4]*mat[,2]+mat[,4]*mat[,3]+mat[,1]*mat[,2]*mat[,3]*mat[,4]
  )
  return(formula)}
```

Obr 11: Funkce vytváření formulace lineárních modelů

Dalším funkčním celkem je výpočet jednotlivých matematických modelů. Jedná se o dva lineární modely a neuronovou síť. Dva lineární modely jsou zde z důvodu jejich možnosti aproximovat data. Jeden je jednoduchý a druhý je úplný. Jednoduchým modelem je myšleno, že jeho možnost aproximace je čistě lineární a u úplného je aproximace nastavená na polynomickou. V jazyce R se k výpočtům těchto modelů využívá funkce *lm*, která má vstupními proměnnými formulaci typu aproximace, která je zde jako datový typ *as.formula* a data v maticovém zobrazení. Pro zjednodušení a zpřehlednění byla vytvořená funkce *createForm1* pro první model a *createForm2.* Vstupními proměnnými je matice se vstupními hodnotami úlohy a index s informací, ve kterém sloupci matice se nachází výstupní proměnná úlohy.

(Stefan Fritsch, Frauke Guenther, 2016)

Pro vytvoření neuronové sítě se v práci využívá balíček neuralnet. Ten obsahuje stejnojmennou funkci *neuralnet*, která vytváří neuronovou síť, která slouží ke klasifikaci a predikci dat. Vstupními parametry je formulace typu aproximace stejně jako u lineárních modelů, vstupní matice s hodnotami vstupních hodnot a výsledných hodnot, které slouží k učení dané sítě. Dalším parametrem je hodnota či vektor hodnot, který určuje kolik daná síť bude obsahovat skrytých vrstev a kolik neuronů bude obsahovat jednotlivá vrstva. Dalšími parametry je hodnota přípustné chyby, které poté co neuronová síť dosáhne, přestává se učit a proměnná *lifesign*, která pokud je nastavená na hodnotu „full“ vypisuje jednotlivé fáze učení do terminálu vývojového prostředí RStudia. Tato neuronová síť využívá algoritmu zpětné propagace, což je typ učení, kdy probíhá dopředný průchod sítí od vstupní vrstvy až po aktivační funkci, která je v tomto případě lineární (pokud ji parametrem funkce nezměníme). Poté probíhá

zpětný průchod sítí, kdy chyby, které vznikly dopředným průchodem, jsou propagovány zpět na vstupní vrstvu a celý proces se opakuje do doby, než je chyba rovna námi nastavenému parametru. Zjednodušeně to probíhá tak, že se neuronová síť snaží dosadit vhodné koeficienty do rovnice aproximované přímky a dál se s těmito koeficienty snaží dopočítávat výstupní hodnoty. Pokud dojde k aktivační funkci, kde se snaží tyto koeficienty dosadit do obrácené rovnice a zpětným průchodem zjistit, jestli se rovnice znovu rovná vstupním hodnotám.

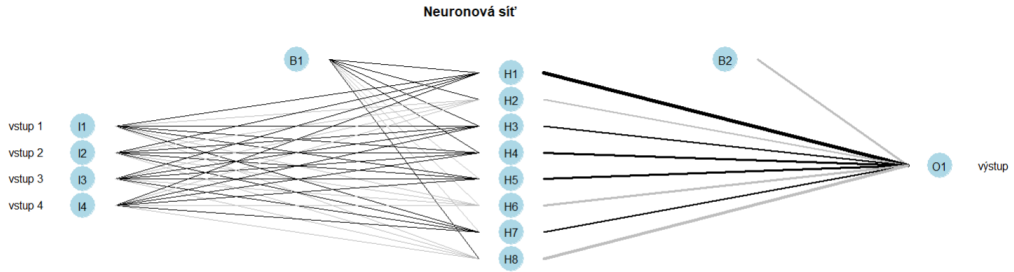
Poslední částí toho programu je analýza výše zmíněných modelů. Po fázi učení neuronové sítě se provádí predikce dat, která daná síť neměla k dispozici. Pomocí posuvníků je možné nastavit, kolik hodnot se celkem vygeneruje a kolik bude učících dat. Díky tomu jsme schopni neuronovou síť naučit na podmnožině vygenerovaných dat a poté zbytek nechat neuronovou síť dopočítat. V balíčku neuralnet k tomu slouží funkce *compute*, která má vstupními parametry natrénovanou neuronovou síť a proměnou, vektor nebo matici hodnot, které je jejím úkolem dopočítat.

(Beck, 2016)

2.1.4 Analýza na serverové části

Jako analytické nástroje využíváme garsonova algoritmu, oldenova algoritmu a odchylky. Kdy garsonův a oldenův algoritmus jsou funkce, které jsou implementovány v balíčku NeuralNetTools. Tyto metody slouží k určení důležitosti jednotlivých vstupních proměnných, které vstupují do neuronové sítě.

Těchto algoritmů využívá komponenta *plot.nnet*, která vytváří graf neuronové sítě, ve které nejdůležitější synapse mezi neurony zvýrazní na základě jejich důležitosti. Jak je z Graf neuronové sítě s jednou skrytou vrstvou a osmi neurony vidět, velký vliv na výpočet má první z neuronů skryté vrstvy a jeho kombinace vstupních proměnných na vstupu neuronu.



Obr 12: Graf neuronové sítě s jednou skrytou vrstvou a osmi neurony
(Wickham, Chang, 2016)

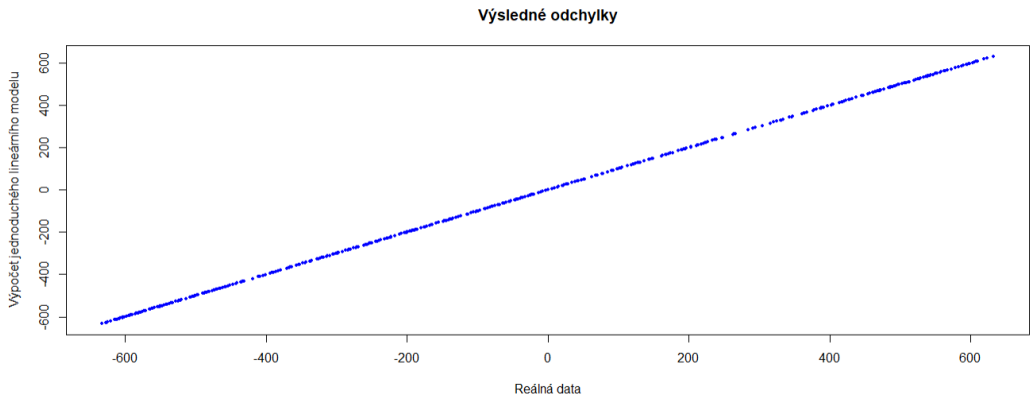
Aby bylo možné ověřit správnost těchto algoritmů, byly hodnoty vstupů vynásobeny koeficientem důležitosti (viz Práce s číselnými řadami). Tímto způsobem lze porovnat,

pro jak velká zašumění dat lze tyto algoritmy použit, a tím pádem zjistit, jestli je možné je využít na reálných datech.

Pokud se jedná o analýzu lineárních modelů, tak je možné použít funkci *coefficients*, která má vstupními proměnnými lineární model, který byl použit k aproximaci dat a vektor s velikostí, která odpovídá vstupním proměnným. Tato funkce nám vrací vektor hodnot, které odpovídají důležitostem jednotlivých proměnných.

(Beck, 2016)

Pro porovnání správnosti výpočtu jednotlivých modelů a zjištění, jak jsou dané modely odolné vůči šumu, využívá práce výpočtu odchylky. Zde se jedná o výpočet, kde jednotlivé hodnoty výpočtu a reálných dat od sebe odečítáme a tuto hodnotu (odchylku) znázorňujeme v grafu. Na grafu z Odchylky lineárního modelu je patrné, že pokud se hodnoty rovnají, tak vzniká rovná křivka, která danou oblast grafu protíná diagonálně. Čím blíž jsou data v grafu umístěna k této diagonální přímce, tím mezi hodnotou výpočtu a reálnou hodnotou nastává menší odchylka.



Obr 13: Odchylky lineárního modelu

Jako poslední analytickou metodu využíváme korelaci, kde v jazyce R tuto metodu zajišťuje funkce *cor*, kde vstupními argumenty jsou vektory hodnot, které jsou mezi sebou porovnávány a podle korelačního vztahu je vypočítán korelační koeficient, který určuje, jestli jsou výsledné hodnoty modelu a reálná data závislá. Pokud korelační koeficient vychází hodnotou v okolí nuly, pak jsou tyto hodnoty zcela odlišné a model nebyl schopen správně data spočítat. Pokud vychází korelační koeficient v okolí hodnoty jedna, pak jsou data závislá, a tím je výsledek modelů správný. Pokud vychází v okolí mínus jedné, nastává opačná závislost a hodnoty modelů jsou zkreslené. V této práci tato metoda slouží pouze informativně, protože se jedná pouze o jedno číselnou hodnotu a nebyl možný způsob, jak ji graficky znázornit ve webové aplikaci.

(Beck, 2016)

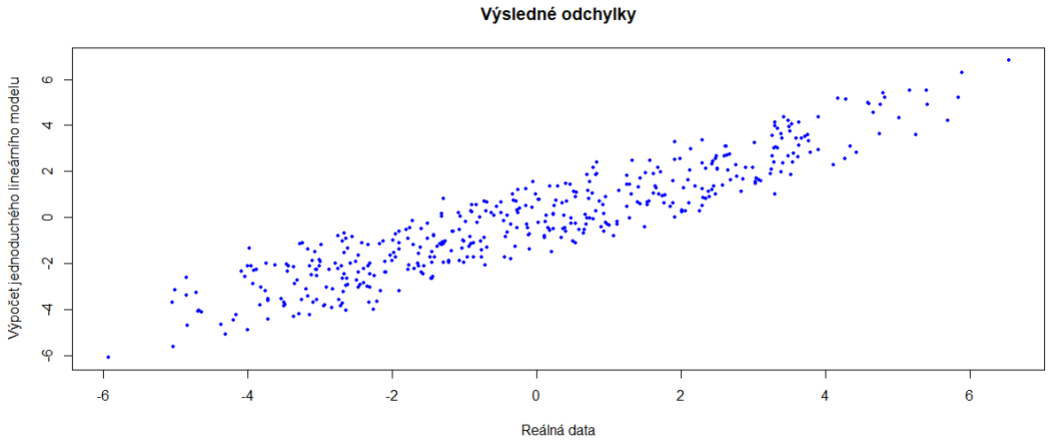
2.1.5 Webová aplikace

Webová aplikace je vytvořená pomocí balíčku Shiny, pomocí vlastních Css stylů a rozšiřujících balíčků pro vizuální vzhled komponent. Celá aplikace je rozdělena na dvě hlavní části. První je interaktivní panel, kde uživatel zadává jednotlivé parametry a druhá část obsahuje informace o naměřených hodnotách.

Ovládací panel se skládá z pěti posuvníků, čtyř zadávacích polí a třech radiobuttonech. Pomocí posuvníků si uživatel volí, kolik se vygeneruje záznamů datasetu, kolik z těchto dat bude mít k dispozici neuronová síť jako učících, jak velký koeficient zašumění bude použito na datech, kolik bude obsahovat neuronová síť neuronů na druhé skryté vrstvě a jak velkou povolí chybovost výpočtu neuronové sítě. Pomocí zadávacích polí si uživatel zadává jednotlivé koeficienty u vstupních hodnot, které určují důležitost jednotlivých hodnot a výběrem z radiobuttonů si zvolí, o jakou závislost dat se bude jednat.

Obr 14: Ovládací panel webové aplikace

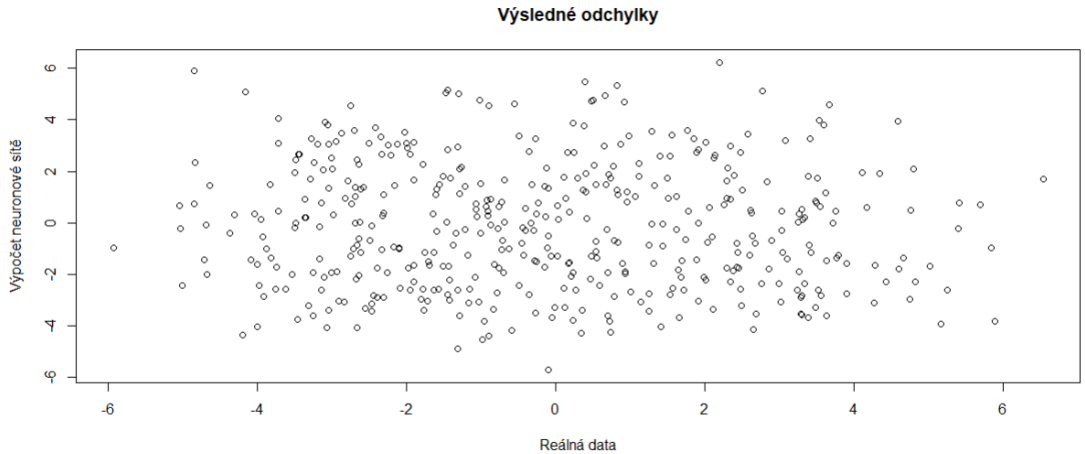
Druhá část se skládá ze čtyř listů. První z nich obsahuje informace o naměřených důležitostech jednotlivých metod. Prvním prvkem je graf, který obsahuje porovnání jednotlivých metod, jak byly schopné vypočítat jednotlivé důležitosti vstupních dat. Zbylé tři prvky jsou tabulky, které zobrazují výstupy jednotlivých metod.



Obr 15: Výsledné odchylky jednoduchého lineárního modelu

Druhý z listů obsahuje grafické znázornění odchylek vypočítaných hodnot jednotlivých lineárních modelů od reálných výsledků. Na Výsledné odchylky jednoduchého lineárního modelu je znázornění, jak byly hodnoty správně vypočítány lineárními modely a jak velkou odchylkou se liší od reálných dat. Jedná se poměr mezi reálnými a vypočítanými daty.

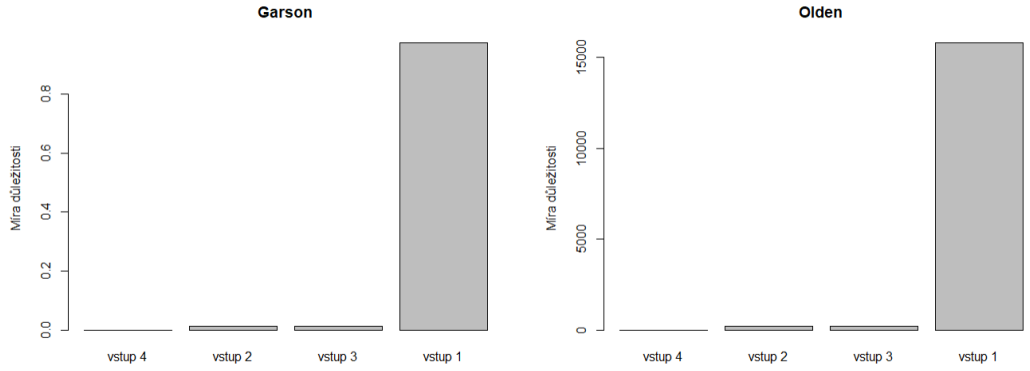
(Wickham, 2009)



Obr 16: Výsledné odchylky neuronové sítě

Třetí z listů obsahuje informace o neuronové síti. Tento list se skládá z dalších dvou panelů, kde v prvním z nich je grafické znázornění neuronové sítě komponentou *plot.net* a grafické znázornění odchylek vypočítaných hodnot neuronovou sítí od reálných výsledků. Ve druhém z panelů je znázornění výpočtů důležitostí Garsonovým a Oldenovým algoritmem. Grafy z Grafické znázornění výpočtů Garsonova a Oldenova

algoritmu jsou závislé na důležitosti jednotlivých vstupů. Čím je v grafu u vstupu vyšší hodnota, tím je tento vstup důležitější a výrazněji ovlivňuje danou úlohu.



Obr 17: Grafické znázornění výpočtů Garsonova a Oldenova algoritmu

Poněvadž jsou všechny grafické elementy naplněné daty až po první spuštěním výpočtu, bylo nutné vymyslet, jak bude vypadat prvotní stránka při spuštění aplikace. Tento problém byl vyřešen tím, že se při prvním spuštěním aplikace vygeneruje přes *mainpanel* modrý div, ve kterém jsou popsány výpočty a pokyny k využívání aplikace. Po prvním kliknutí na tlačítko, které spustí funkci výpočtů na sever, se tento div zneviditelní metodou *hide* a zobrazí se pomocí funkce *show* jiný div s informací, že se načítají data. Obě tyto metody mají vstupním parametrem pojmenování elementu, kterému se při spuštění této funkce přenastavují in-line css styly, kde se atribut *display* nastaví buď na *none,* nebo *block*.

Protože se některá data grafů načítají delší dobu, vzniká na stránce prázdné místo, které neobsahuje zprvu žádný element. Aby uživatel věděl, že se na serverové části provádí načítání dat grafu, využíváme balíček shinycssloaders, která nám pomocí komponenty *withSpinner* vytváří indikátor toho, že se graf načítá. Funguje to tak, že vstupem této komponenty je komponenta grafu či tabulky, kdy komponenta *withSpinner* pomocí vnitřních hodnot vstupní komponenty překresluje danou oblast indikátorem.

2.2 Práce s reálnými daty

Hlavním bodem této práce je pomocí jednotlivých analýz určit to, jak jednotlivé modely zvládají práci s reálnými daty. V této práci se využívají hydrologická data, která byla měřená od roku 1951 až do roku 2014. K dispozici jsou informace o průtoku řeky, úhrn srážek a čerpání vody.

Na těchto datech se snažíme zjistit, jak dané vstupní řady ovlivňují výslednou řadu, což je výška hladiny ve vrtu. Toto zjišťujeme pomocí lineárních modelů, garsonova a oldenova algoritmu.

Z důvodu neúplnosti měření jednotlivých vstupních řad, je nutné nejprve všechny toto řady upravit. Protože u jednotlivých řad se vyskytují období ve kterých nejsou dané hodnoty naměřeny, z toho důvodu je nutné jednotlivé řady vůči sobě posunout, aby bylo možné vytvoření předpisu pro dané lineární modely a neuronovou síť. Toto řešíme posunutím jednotlivých dat do hodnot s nejvyšší mírou korelace. Dalším z problémů těchto vstupních dat je ta, že jednotlivé hodnoty mohou být značně zkreslená náhodným šumem. Tento faktor ve velké míře ovlivňuje správnost rekonstrukce a analýzy daných modelů. Proto tyto vstupní proměnné upravujeme klouzavým průměrem.

Aby bylo možné zjistit, jak dané modely reagují na upravená a neupravená data, tak jsme jednotlivé analýzy a rekonstrukce prováděli na souboru surových data a upravených dat.

2.3 Vyhodnocení naměřených dat

Pro vyhodnocení bylo využito webové aplikace, kde bylo možné měnit jednotlivé parametry a opakovat jednotlivé výpočty.

V případě umělých dat je konfigurace pokusu nastavená tak, že se vygenerují čtyři vstupní řady s náhodnými hodnotami a výsledná řada těchto vstupních řad. Definujeme si délku vstupní řady na hodnotu 800 a z těchto vstupních dat bude mít neuronová síť přístup pouze k 500 hodnotám. Pro jednotlivé vstupní řady bude přidán koeficient důležitosti, díky kterému můžeme porovnat správnost výpočtů jednotlivých analýz. Pro tento příklad je použitá neuronová síť se šesti neurony ve skryté vrstvě.

Problém s analýzami neuronové sítě je ten, že v jistých případech dochází k přetrénování a tím pádem je možné, že neuronová síť nezvládá korektně rekonstruovat výslednou řadu.

2.3.1 Umělá data – lineární závislost

První ze zkoumaných závislostí je lineární, v které lze předpokládat, že jednotlivé výpočty lineární modelů budou přesnější než výpočty neuronové sítě a algoritmů, které vyhodnocují jednotlivé důležitosti.

Prehled koeficientů

	Garson	Olden	LM2	LM1
vstup 1	0.13	5.76	1	1
vstup 2	0.19	5.73	1	1
vstup 3	0.65	23.34	4	4
vstup 4	0.03	0.6	0.1	0.1

Obr 18: Tabulka koeficientů bez šumu (lineární závislost)

Prehled koeficientů

	Garson	Olden	LM2	LM1
vstup 1	0.26	5.4	1.05	1.04
vstup 2	0.16	4.83	1.05	1.05
vstup 3	0.5	24.04	3.61	3.61
vstup 4	0.09	0.59	0.25	0.24

Obr 19: Tabulka koeficientů se šumem 0,4 (lineární závislost)

Prehled koeficientů

	Garson	Olden	LM2	LM1
vstup 1	0.22	25.79	1.03	1.02
vstup 2	0.16	8.4	1.06	1.07
vstup 3	0.44	102.05	2.52	2.52
vstup 4	0.18	48.36	0.58	0.57

Obr 20: Tabulka koeficientů se šumem 1 (lineární závislost)

Z výše uvedených obrázků tabulek s jednotlivými výpočty na různých úrovních šumů lze pozorovat pokles správnosti výpočtů garsonova a oldenova algoritmu. Kdy u lineárních modelů klesá správnost nepatrně, kdežto u těchto algoritmů se hodnoty jednotlivých důležitostí rapidně mění. Přesto ale dokáží spočítat, která ze vstupních dat má největší podíl na rekonstrukci výsledné řady.

Lze usoudit, že u garsonova a oldenova algoritmu dochází k vysokým výchylkám z důvodu natrénování neuronové sítě, kdy se neuronová síť snaží vstupní data až moc dokonale popsat a dochází k přetrénování.

2.3.2 Umělá data – polynomiální závislost

Další ze zkoumaných závislostí je polynomiální. V tomto případu by měl být pozorovatelný pokles správnosti i u lineárních modelů.

Podle naměřených koeficientů lze zkoumat, že i v tomto případu jednotlivé metody do jisté míry zvládají vypočítat důležitosti správně. Až na garsonův algoritmus, který při úrovni šumu s hodnotou jedna do výpočtů zahrnuje i poslední vstupní řadu.

Prehled koeficientů

	Garson	Olden	LM2	LM1
vstup 1	0.12	4.24	1	1
vstup 2	0.11	4.15	1	1
vstup 3	0.72	65.68	16	16
vstup 4	0.05	0.4	0.1	0.1

Obr 20: Tabulka koeficientů bez šumu (polynomiální závislost)

Prehled koeficientů

	Garson	Olden	LM2	LM1
vstup 1	0.13	163.66	1.04	1.04
vstup 2	0.12	391.01	1.27	1.22
vstup 3	0.62	4143	13.32	13.39
vstup 4	0.13	162.62	0.07	0.1

Obr 21: Tabulka koeficientů se šumem 0,4 (polynomiální závislost)

Prehled koeficientů

	Garson	Olden	LM2	LM1
vstup 1	0.2	566.42	0.69	0.67
vstup 2	0.11	892.31	0.79	0.74
vstup 3	0.42	3271.4	7.64	7.67
vstup 4	0.27	291.74	0.15	0.14

Obr 22: Tabulka koeficientů se šumem 1 (polynomiální závislost)

2.3.3 Umělá data – lomená závislost

Tento příklad zkoumá lomenou závislost. V tomto případě víme, že podle zvoleného předpisu výstupní řady by měla mít čtvrtá vstupní řada nejvyšší důležitost při výpočtu výstupní řady.

Prehled koeficientů

	Garson	Olden	LM2	LM1
vstup 1	0.27	11626.98	-1.98	-1.87
vstup 2	0.18	7436.86	-0.93	-0.92
vstup 3	0.51	391278.8	-0.06	-0.06
vstup 4	0.04	5010.83	-0.87	-0.85

Obr 23: Tabulka koeficientů bez šumu (lomená závislost)

Prehled koeficientů

	Garson	Olden	LM2	LM1
vstup 1	0.2	945.83	-1.39	-1.35
vstup 2	0.07	1393.31	-1.96	-1.93
vstup 3	0.39	649.74	-0.52	-0.59
vstup 4	0.34	983.71	-0.36	-0.34

Obr 24: Tabulka koeficientů se šumem 0,4 (lomená závislost)

Prehled koeficientů

	Garson	Olden	LM2	LM1
vstup 1	0.11	682.91	-0.72	-0.67
vstup 2	0.14	1022.44	-1.49	-1.47
vstup 3	0.39	1184.11	-0.74	-0.71
vstup 4	0.36	3362.99	-0.29	-0.24

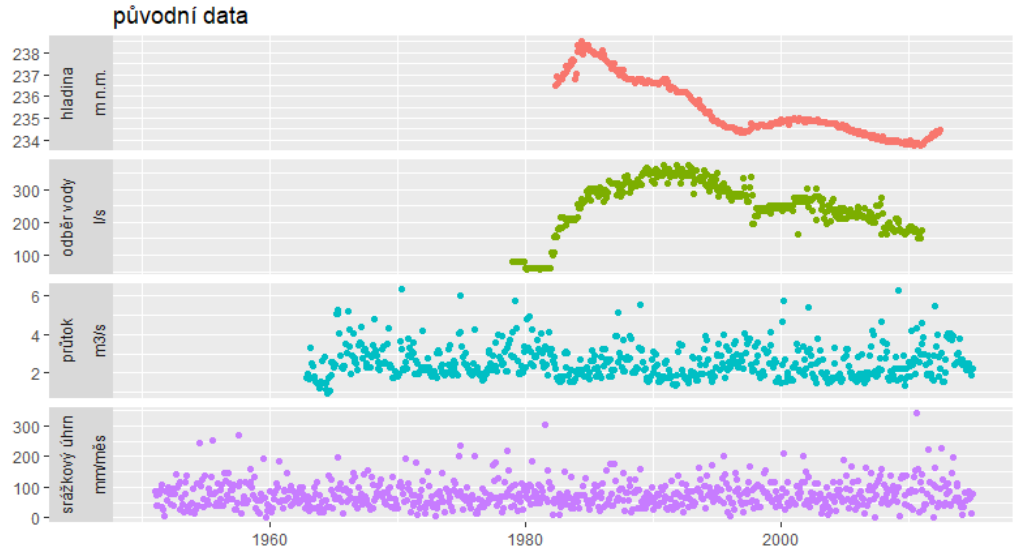
Obr 25: Tabulka koeficientů se šumem 1 (lomená závislost)

V tomto případě je z tabulek viditelné, že se vyhodnocení jednotlivých metod značně liší. U nejvyšší úrovně šumu nejlépe důležitosti vypočítal oldenův algoritmus.

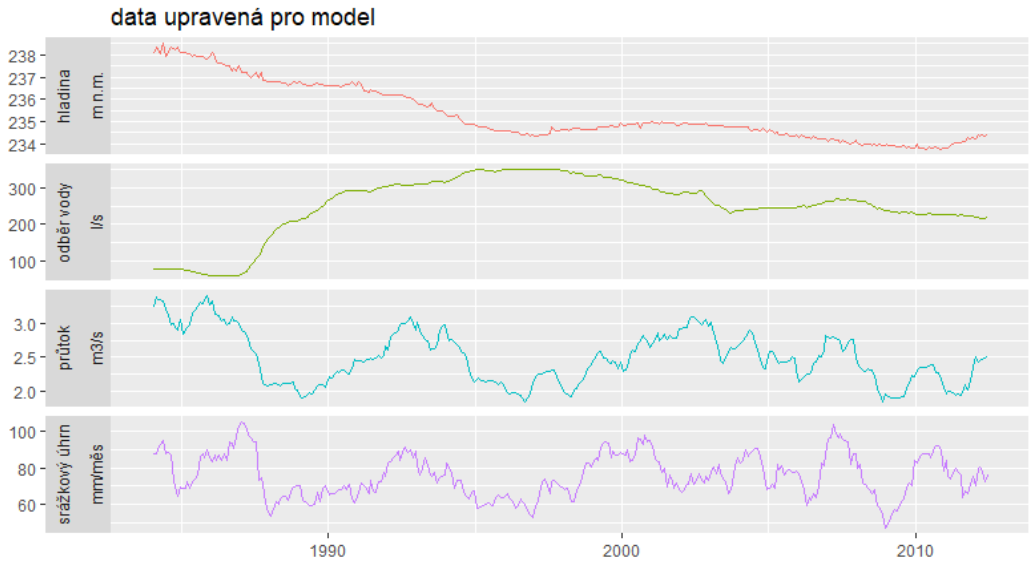
2.3.4 Reálná data

U reálných dat bylo zkoumáno, jak moc jsou dané vstupní řady důležité v odhadu výšky hladiny podzemní vody ve vrtu. K dispozici máme tři vstupní řady s informacemi o čerpání vody, informaci o srážkách a průtoku řeky. Na těchto řadách provádíme jednotlivé analýzy.

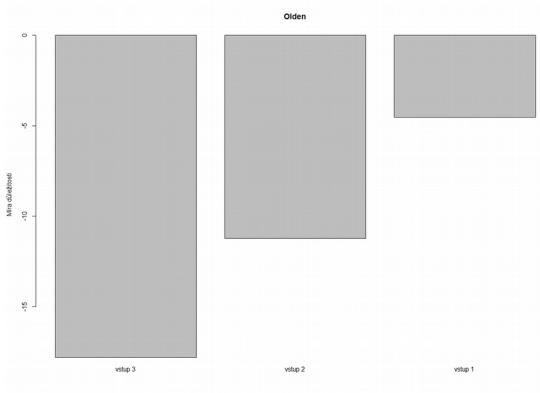
Celkem na reálných datech provádíme analýzu na surových datech a poté na upravených datech klouzavým průměrem a posunem do hodnot s nejvyšší míro korelace.



Obr 26: Graf reálných surových dat



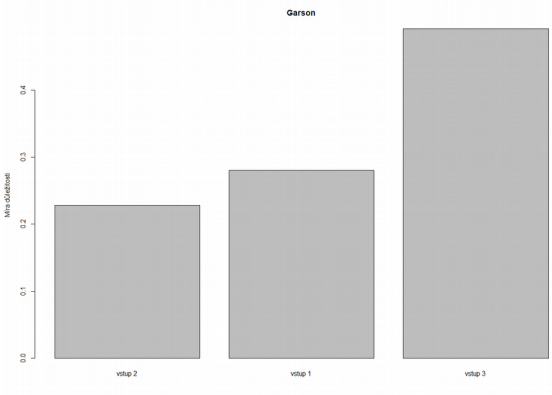
Obr 27: Graf reálných dat upravených klouzavým průměrem a posunutím



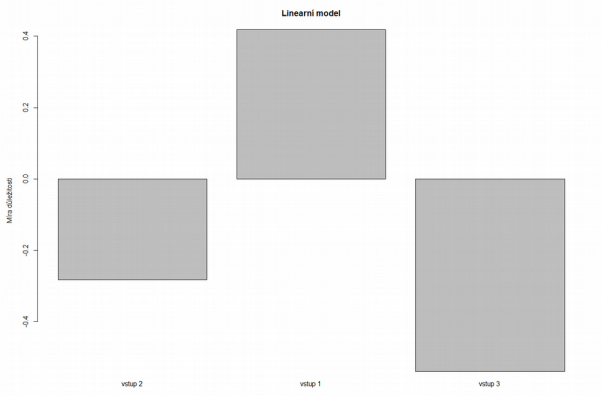
Obr 28: Graf koeficientů - Oldenův algoritmus

Na reálných datech byla provedená dvojí analýza. První z analýz byla prováděná na neupravených datech (viz Graf reálných surových dat). Z této analýzy jsou výstupy Graf koeficientů - Oldenův algoritmus, Graf koeficientů - Garsonův algoritmus a Graf koeficientů - lineární model, kde lze pozorovat jak jednotlivé metody zvládly vypočítat jednotlivé důležitosti. Zde je pod proměnnou vstup1 čerpání podzemní vody, pod proměnnou vstup2 průtok řeky a proměnnou vstup3 jsou hodnoty srážek.

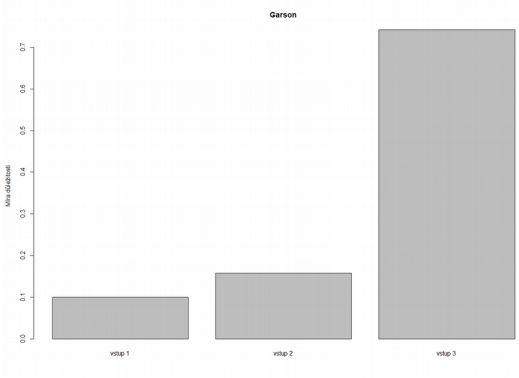
Z jednotlivých grafů lze určit, že nejvyšší míru důležitosti zde mají srážky. V tomto případě se výsledky liší u Garsonova algoritmu s lineárním modelem od Oldenova algoritmu.



Obr 29: Graf koeficientů - Garsonův algoritmus



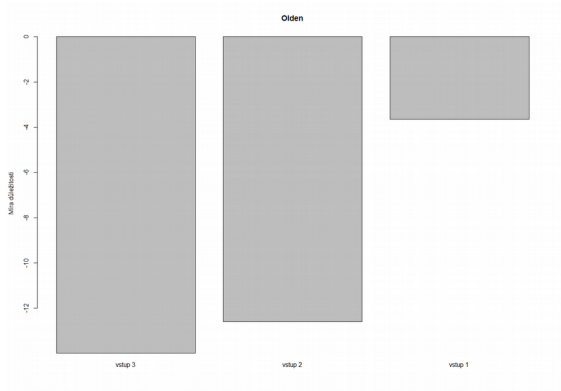
Obr 30: Graf koeficientů - lineární model



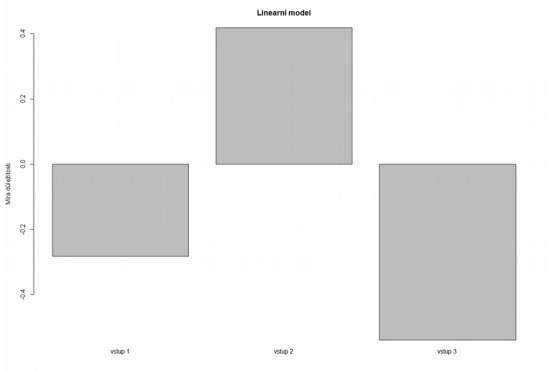
Obr 31: Graf koeficientů - Garsonův algoritmus

Druhá analýza byla prováděna na upravených datech (viz Graf reálných dat upravených klouzavým průměrem a posunutím), kde by mělo dojít k zlepšení jednotlivých výsledků.

V tomto případě je vidět rozdíl při využití surových a upravených reálných dat. Při vizualizaci jednotlivých výsledků metod se korektně zobrazily jednotlivé důležitosti.



Obr 32: Graf koeficientů - Oldenův algoritmus



Obr 33: Graf koeficientů - lineární model

Jedním z možných závěrů je ten, že oldenův algoritmus a lineární modely lépe zvládají vypočítat důležitosti jednotlivých vstupních řad při vyšší míře zašumění a zkreslení. Kdežto garsonův algoritmus je hodně ovlivněn šumem.

Závěr

Tématem této bakalářské práce bylo zkoumání důležitosti jednotlivých vstupních řad, které ovlivňují odhadování výšky hladiny podzemní vody ve vrtu. Jedná se o podrobnou analýzu, která dokáže určit, které ze vstupních řad lze z výpočtů odstranit pro zefektivnění a urychlení. Pro tuto práci byla vybrána data z oblasti hydrologie, kde se snažíme určitě, která ze vstupních řad ovlivňuje odhadování výšky hladiny ve vrtu.

Původním předpoklad byla konzolová aplikace, která pouze zobrazuje jednotlivé výsledky analýz a vytváří grafy. Jenže tato aplikace by byla složitá pro opakování jednotlivých výpočtů a možnost práce s jednotlivými daty. Z tohoto důvodu bylo rozhodnuto, že se pro práci využije balík Shiny jazyka R, díky kterému lze v tomto jazyce vytvářet webové aplikace.

Výsledkem práce je webová aplikace, která slouží k opakování jednotlivých pokusů a vizualizaci jednotlivých výpočtů. Celá tato aplikace slouží pro usnadnění zadávání jednotlivých parametrů pro vyhodnocování jednotlivých pokusů. Z této aplikace se stal nástroj pro analýzu neuronové sítě a lze ji využít k porovnání s lineárními modely.

Pomocí této aplikace bylo možné zkoumat na reálných datech, jak obstojí jednotlivé analýzy. Díky využití analýzy na uměle vytvořených datech, která je možné plně kontrolovat, je možné vyhodnotit, jestli jednotlivé algoritmy lze využít na reálných datech.

Stěžejním bodem této práce bylo nasazení jednotlivých analýz na reálných datech. Podle umělých dat bylo vyhodnoceno, že jednotlivé analýzy obstojí i u dat, která jsou ve vyšší míře zašuměná. Díky tomu bylo možné říci, že výstupy těchto analýz jsou do jisté míry možné uznat za korektní.

Po nasazení těchto analýz na reálných datech bylo zjištěno, že výšku hladiny ve vrtu nejvíce ovlivňují srážky a průtok řeky. Tím pádem údaj o čerpání vody je možné z výpočtů vyřadit a tím zefektivnit jednotlivé rekonstrukce výsledné řady jednotlivých lineárních modelů a neuronové sítě.

Pomocí umělých dat a možnosti zadávat jednotlivé parametry, které ovlivňují výpočty neuronové sítě, je možné využít tuto aplikace k analýze celé neuronové sítě. Díky tomu, že lze jednotlivé pokusy opakovat, lze tato aplikace v budoucnu rozšířit o analýzu toho, jak jednotlivé parametry ovlivňují výpočty neuronové sítě. V této práci je vypracováno ukládání jednotlivých parametrů, ale z časových důvodu nebyla tato analýza do práce začleněna.

Citace

Beck M (2016). _NeuralNetTools: Visualization and Analysis Tools for Neural Networks_. R package version 1.5.0. https://CRAN.R-project.org/package=NeuralNetTools

Hadley Wickham and Winston Chang (2016). devtools: Tools to Make Developing R Packages Easier. R package version 1.12.0. https://CRAN.R-project.org/package=devtools

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2009.

CHÁRA, Petr, Dana BLAHUNKOVÁ a Eva ŘÍDKA. Maturitní otázky – Matematika. Praha: Fragment, Albatros Media, 2007. ISBN 9788025315507.

ŘEHÁK, Jan a Ondřej BROM. SPSS – Praktická analýza dat. Praha: Computer Press, Albatros Media, 2017. ISBN 8025146294, 9788025146293.

SHIFFMAN, Daniel. Chapter 10. Neural Networks. Nature of code [online]. New York, 2016 [cit. 2019-02-28]. Dostupné z: https://natureofcode.com/book/chapter-10-neural-networks/

Stefan Fritsch and Frauke Guenther (2016). neuralnet: Training of Neural Networks. R package version 1.33. https://CRAN.R-project.org/package=neuralnet

Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2017). shiny: Web Application Framework for R. R package version 1.0.3. https://CRAN.R-project.org/package=shiny

Yihui Xie (2016). DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.2. https://CRAN.R-project.org/package=DT

R Documentation [online]. Northern Ave, Boston, MA: The R Foundation, 2017 [cit. 2019-02-28]. Dostupné z: https://www.r-project.org/other-docs.html

Shiny from RStudio [online]. Northern Ave, Boston, MA: RStudio, 2017 [cit. 2019-02-28]. Dostupné z: https://shiny.rstudio.com/deploy/

Neural Network Models in R [online]. DataCamp, 2019 [cit. 2019-03-28]. Dostupné z: https://www.datacamp.com/community/tutorials/neural-network-models-r

NeuralNetTools: Visualization and Analysis Tools for Neural Networks [online]. 2018 [cit. 2019-04-28]. Dostupné z: https://rdrr.io/cran/NeuralNetTools/

Higher-Order Components [online]. Facebook, 2017 [cit. 2019-03-28]. Dostupné z: https://reactjs.org/docs/higher-order-components.html