ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA ELEKTROTECHNICKÁ

KATEDRA ELEKTROENERGETIKY

DIPLOMOVÁ PRÁCE

Parametrické modelování spotřeby elektrické energie pomocí prediktivních metod

Bc. Vojtěch Müller

2021

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: 2020/2021

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Vojtěch MÜLLER**
Osobní číslo:	**E19N0035P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Elektroenergetika**
Téma práce:	**Parametrické modelování spotřeby elektrické energie pomocí prediktivních metod**
Zadávající katedra:	**Katedra elektroenergetiky**

Zásady pro vypracování

1. Popište zvolenou metodu predikce spotřeby elektrické energie.
2. Na poskytnutých datech uveďte rozdíly při aplikaci metody v různých místech elektrizační soustavy.
3. Navrhněte systém pro online monitoring a řízení čtvrthodinového maxima zvolenou metodou.
4. Zhodnoťte aplikaci tématu s ohledem na přítomnost elektromobility a bateriových úložišť v okolní síti.

Rozsah diplomové práce: **40 – 60 stran**
Rozsah grafických prací: **podle doporučení vedoucího**
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. BISHOP, C. M. Pattern Recognition and Machine Learning. Cambrige, 2006
2. BAZALOVÁ, K. Krátkodobé predikce spotřeby elektrické energie s využitím neuronových sítí. Ostrava, 2003. Disertační práce. Vysoká škola báňská – Technická univerzita Ostrava

Vedoucí diplomové práce: **Ing. Václav Mužík, Ph.D.**
Regionální inovační centrum elektrotechniky

Datum zadání diplomové práce: **9. října 2020**
Termín odevzdání diplomové práce: **27. května 2021**

L.S.

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

Doc. Ing. Karel Noháč, Ph.D.
vedoucí katedry

V Plzni dne 9. října 2020

Abstrakt

Předkládaná diplomová práce se zabývá parametrickým modelováním spotřeby elektrické energie za pomoci prediktivních metod. V první části práce je zpracován průzkum současných trendů v oblasti prediktivních metod. Z možností, které průzkum poskytl, byly vybrány dvě metody, které byly dále důkladně zpracovány. Následně je rozebrána aplikovatelnost programového vybavení a jeho nástrojů na poskytnutých datech. Výchozím programem je výpočetní prostředí Matlab a dále je také využito prostředí Scikit-learn, které je vyvinuto v Pythonu. Poskytnutá naměřená data jsou použita na připravené metody a jsou diskutovány výsledky z různých míst elektrizační soustavy. V další části práce je navržen systém pro online monitoring a s ním spojené řízení čtvrthodinového maxima. V závěru práce je zhodnocena aplikace tématu s ohledem na přítomnost elektromobility a bateriových uložišť v okolní síti a dále je provedena patřičná diskuse všech dosažených výsledků.

Klíčová slova

Elektrická energie, predikce spotřeby, umělá neuronová síť, metoda zpětného šíření (backpropagation), rozhodovací stromy, tréninkové vzory, odebíraný výkon, maximální zatížení, položpičkové zatížení, diagram zatížení, distribuční transformátor, velkoodběratel, elektromobilita, bateriové uložiště.

Abstract

This diploma thesis deals with parametric load modeling using predictive methods. The first part of the thesis is a research of current trends in predictive methods. From the options provided by the research, two methods were selected and further elaborated. Subsequently, the applicability of software and its tools to the provided data is discussed. The default program is the computing environment Matlab and the Scikit-learn environment is also used, which is developed in Python. The provided measured data were used for the prepared methods and the results from various places of the electrical system are discussed. In the next part of the work, a system for online monitoring and associated control of the quarter-hour maximum is proposed. At the end of the work, the application of the topic is evaluated with regard to the presence of electromobility and battery storage in the surrounding network and a proper discussion of all achieved results is made.

Key words

Electrical energy, consumption prediction, artificial neural network, backpropagation method, decision trees, training patterns, maximum load, half-peak load, power consumption, load diagram, distribution transformer, large customer, electromobility, battery storage.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

.....

podpis

V Plzni dne 27.5.2021 Vojtěch Müller

Poděkování

Tímto bych rád poděkoval vedoucímu diplomové práce Ing. Václavu Mužíkovi, Ph.D. za cenné profesionální rady, připomínky a metodické vedení práce. Rovněž bych chtěl poděkovat za jeho nasazení a ochotu řešit problematiku při častých konzultacích a taktéž za poskytnutí všech naměřených dat potřebných pro řešení této práce.

Obsah

Seznam symbolů a zkratek

w ..... váha synaptického spojení mezi neurony

w ..... vektor vah synaptického spojení mezi neurony

x ..... vstup neuronu

x ..... vstupní vektor sítě

y ..... výstup neuronu

y ..... výstupní vektor sítě

Y ..... formální neuron

$f(y_{in})$ ..... přenosová funkce

n ..... počet neuronů

z ..... neuron skryté vrstvy

z ..... vektor skryté vrstvy

E(w) ..... chybová funkce

BP ..... backpropagation

LM-BP ..... Levenberg-Marquardt backpropagation

DRL ..... deep reinforcement learning

A3C..... Asynchronous Advantage Actor-Critic

DDPG ..... Deep Deterministic Policic Gradient

RDPG ..... Recurrent Deterministic Policic Gradient

W ..... elektrická energie (MWh)

P ..... činný výkon (MW)

Pmin..... minimální výkon (MW)

Pmax ..... maximální výkon (MW)

Pstř..... střední výkon (MW)

S ..... zdánlivý výkon (MVA)

t ..... čas (s)

τ..... doba využití maxima (h)

vvn ..... velmi vysoké napětí

vn ..... vysoké napětí

nn ..... nízké napětí

Úvod

„Vědění je moc“, řekl kdysi filozof Francis Bacon, obzvlášť když znáte výsledek události, která ještě nenastala, nebo alespoň máte věci promyšlené o tah dopředu. Budoucnost bohužel předvídat nelze, nebo alespoň vědecká obec se k tomuto tvrzení přiklání, co ale možné je, je zkusit odhadnout nadcházející události s co nejvyšší pravděpodobností a tím získat nadhled v oblasti, která vás zajímá, a z dlouhodobého hlediska potom i získat poměrně přesné údaje o předvídaných jevech.

Odhadováním, respektive predikováním, se zabývá i tato práce. Přesněji půjde o predikci spotřeby elektrické energie parametrickým namodelováním různých odběrů v různých místech elektrizační soustavy. Jedním ze zkoumaných objektů bude transformátor v rozvodně vvn/vn a potom soukromý odběr na napěťové hladině vn/nn. Práce je prováděna na poskytnutých datech od vedoucího této diplomové práce Ing. Václava Mužíka, Ph.D.

Součástí této práce je také menší průzkum v oblasti současných trendů v prediktivních metodách, jehož cílem je seznámit čtenáře s aktualitami v této oblasti, protože tato problematika je v dnešní době velice populární a její vývoj jde neustále dopředu. Je těžké zůstat v dobrém obraze o současné situaci a mít aktuální informace, a z tohoto důvodu je tato kapitola součástí diplomové práce.

Výchozím softwarem pro vytváření prediktivních modelů je nejnovější verze výpočetního prostředí Matlab, jehož licenci poskytla Západočeská univerzita. Toto prostředí poskytuje mnoho mocných nástrojů a nezaostává ani na poli prediktivních metod, kde je například využito umělých neuronových sítí. Sekundárním nástrojem je zvolen výběr z palety nástrojů Scikit-learn.

Dalším cílem této práce je určení rozdílů při aplikaci zvolené metody v různých místech elektrizační soustavy, tedy určit, jak se mohou lišit vstupní data pro různý druh odběru. Jaká data jsou nezbytná pro různé metody a jejich dobré natrénování a jak se výsledky budou lišit s ohledem na různé umístnění v elektrizační soustavě nebo různém charakteru sledované zátěže.

Práce si také dává za cíl navržení systému pro online monitoring a řízení čtvrthodinového maxima. V posledních kapitolách bude zhodnocena aplikace celého tématu s ohledem na přítomnost elektromobility a bateriových uložišť v okolní síti, neboť silná bateriová uložiště nám mohou pomoci při řízení elektrizační soustavy, kdežto narůstající elektromobilita celkovou stabilitu sítě může narušovat.

1 Průzkum současných trendů v oblasti prediktivních metod

Tento průzkum si klade za úkol čtenáře seznámit se současnými trendy v oblasti prediktivních metod, neboť v současné době se tato problematika stala velmi populární a prediktivní modely lze napasovat takřka na cokoliv, například na mnoho ekonomických modelů, jako je růst či pokles akcií, tak také pro různé odhady spotřeby všech zdrojů energie. Přitom za největší výhodu těchto prediktivních metod se dá považovat jejich univerzálnost a možnost využití stejného teoretického modelu na různé problematiky, samozřejmě s patřičným respektováním jejich struktury a užitím správných vstupních dat.

1.1 Regresní analýza

První metodou, která by mohla být potenciálně vhodná, je regresní analýza. Tato technika byla tradičně nejoblíbenější při modelování předpovídání spotřeby energie. Vícenásobný regresní model s více než jednou vypovídací proměnnou lze zapsat jako:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon \qquad (1)$$

kde y je výstupní proměnná, $ß_i$, i = 0, 1, 2,…p jsou regresní parametry a ε náhodný chybový člen. V modelu vícenásobné regrese se pro účely odhadu obecně využívá metoda nejmenších čtverců. Po získání regresních koeficientů lze použít predikční rovnici k predikci hodnoty spojitého výstupu (cíle) jako lineární funkce jednoho nebo více nezávislých vstupů. Popularitu regresních modelů lze přičíst interpretovatelnosti parametrů modelu a snadnému použití. Hlavní koncepční omezení všech regresních technik však spočívá v tom, že lze určit pouze vztah, avšak nikdy není možné si být jistý základním kauzálním mechanismem. [1]

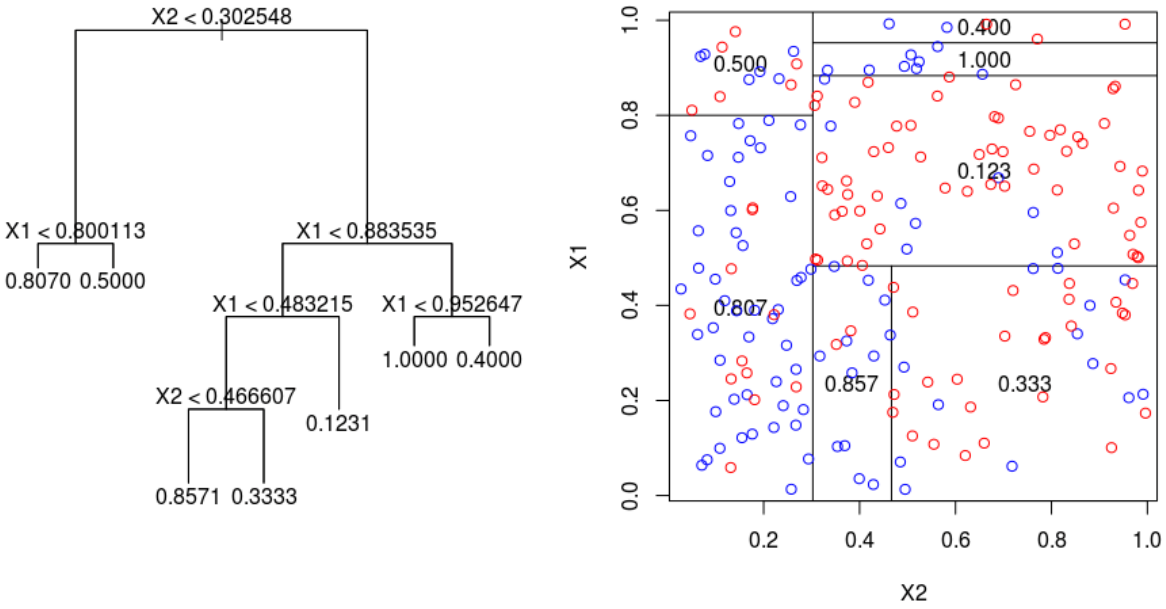
1.2 Rozhodovací strom

V modelování rozhodovacího stromu (decision tree) představuje empirický strom segmentaci dat, která je vytvořena použitím řady jednoduchých pravidel. Tyto modely generují sadu pravidel, která lze použít pro predikci prostřednictvím opakovaného procesu rozdělení. Mezi nejběžnější stromové metody patří automatická detekce interakcí chi-squared a klasifikační a regresní stromy.

Hlavní výhodou rozhodovacího stromu oproti jiným technikám modelování je, že vytváří model, který může představovat interpretovatelná pravidla nebo logické příkazy. Důležitým rysem této metody je vypovídací schopnost, která existuje u stromů produkujících osově paralelní rozhodovací plochy. Kromě toho lze klasifikaci provést

bez složitých výpočtů a tuto techniku lze použít pro spojité i kategorické proměnné. Výsledky modelu rozhodovacího stromu dále poskytují jasné informace o důležitosti významných faktorů pro predikci nebo klasifikaci.

Vyvozování výsledků rozhodovacího stromu však obecně nefunguje tak dobře jako například neuronové sítě pro nelineární data a tato metoda je též citlivá na rušivá data. Obecně je tato technika vhodnější pro předpovídání kategorických výsledků, a pokud nejsou k dispozici viditelné trendy a sekvenční vzory, rozhodovací stromy jsou méně vhodné pro aplikaci na data časových řad. [1]



Obrázek 1 Příklad rozhodovacího stromu [1]

1.3 Neuronové sítě

Neuronové sítě si v současnosti vytvořily své místo mezi nástroji pro analýzu dat v různých oblastech. Modely neuronových sítí byly původně vyvinuty vědci, kteří se snažili napodobit neurofyziologii lidského mozku. Jedná se o analytické techniky modelované v závislosti na předpokládaných procesech učení v kognitivním systému a neurologických funkcích mozku. Neuronové sítě jsou schopné předvídat nová pozorování (na konkrétních proměnných) z jiných pozorování (na stejných nebo jiných proměnných) po provedení procesu tzv. učení.

Neuronové sítě fungují dobře v aplikacích, kde je funkční forma nelineární. Jsou zvláště užitečné pro problematiku predikce, kde nejsou známy matematické vzorce a předchozí znalosti o vztazích mezi vstupy a výstupy.

Nevýhodou při použití neuronové sítě pro regresní analýzu je, že neposkytuje p-hodnoty pro testování významnosti odhadů parametrů. Kromě toho je nutný krok předběžného výběru funkcí před učením. Umělé neuronové sítě se skrytými vrstvami jsou lepší jako klasifikátory problémů zahrnujících nelineární rozhodovací hyperplochy, ale jejich interpretace je mnohem těžší. [1]

1.3.1 Neuronová síť zpětného šíření

Neuronová síť zpětného šíření (backpropagation neural network) je obvykle založena na vícevrstvé dopředné neuronové síti s chybovým algoritmem backpropagation. V současné době je nejpoužívanějším algoritmem učení neuronových sítí algoritmus backpropagation a téměř 80 % aplikací neuronových sítí je založeno na algoritmu backpropagation. Přenosová funkce neuronů v neuronové síti backpropagation je obvykle diferencovatelná funkce sigmoidního typu, která může získat jakékoli nelineární mapování mezi vstupem a výstupem. Tato metoda byla úspěšně použita v oblastech, jako je zpracování signálu, počítačové sítě, řízení procesů, rozpoznávání hlasu, rozpoznávání vzorů a komprese dat. Nejprve se pro distribuované úložiště informací používá vzor. Neuronová síť ukládá informace ve formě stavů jednotlivých procesorů a spojení mezi nimi. Zpráva není uložena na jednom místě, ale místo toho je distribuována po síti podle obsahu. Místo v síti se nepoužívá pouze k ukládání externí zprávy, protože ukládá také část informačního obsahu. Po zpracování více informací v celé síti jsou zprávy uloženy všude v síti v režimu distribuovaného úložiště. Zadruhé jsou kombinovány úlohy síťového úložiště backpropagation a výpočty procesů. Ukládání informací je tedy založeno na distribuci propojení mezi neurony, kde je paralelní distribuční vzor ve velkém měřítku lepší než sériové diskrétní zpracování symbolů v moderních digitálních počítačích. Zatřetí neuronové sítě backpropagation vykazují samoučení a adaptabilitu, kde váhy na každé vrstvě v síti backpropagation jsou plastické. Neuronové sítě mohou být vyškolené pomocí učení, aby určily váhy v síti, čímž vykazují vysokou schopnost přizpůsobit se prostředí a rovněž schopnost samoučení. A nakonec neuronové sítě backpropagation jsou vysoce robustní a odolné vůči chybám. Díky těmto výhodám lze síť backpropagation použít pro predikci spotřeby elektrické energie, kdy se systém spotřeby energie dá považovat za vysoce nelineární časovou řadu. [7]

1.3.2 Levenberg-Marquardtova neuronová síť zpětného šíření

Rozšířenou a vylepšenou variantou standardní backpropagation metody je Levenberg-Marquardtův algoritmus. Jedná se o neuronovou síť, která je využívána k nelineárním cvičením algoritmů a v rámci které je kombinována metoda gradientního sestupu a Kvazi-Newtonova metoda, aby byla zajištěna lokální vysoká rychlost konvergence a udržoval se lepší celkový výkon. Kvazi-Newtonova metoda tedy obecně slouží k vyhledávání nul či lokálních maxim a minim funkcí.

Základní myšlenkou Levenberg-Marquardtova algoritmu je, že každá iterace podél jednoho negativního směru gradientu není delší, než je nutné a umožňuje vyhledat chybu ve směru zhoršování. Váhy sítě lze navíc optimalizovat pomocí adaptivního přizpůsobení mezi metodou nejstrmějšího gradientního klesání a metodou Gauss-Newton, která umožňuje efektivní konvergenci sítě, což výrazně zlepšuje rychlost konvergence a zobecnitelnost dat.

Levenberg-Marquardtova neuronová síť zpětného šíření tedy na rozdíl od klasické neuronové sítě zpětného šíření řeší problém s pomalou konvergencí dat. Také řeší fluktuaci a oscilaci během tréninkového procesu, kdy byl model snadno zachycen místními minimy a bylo obtížné určit strukturu sítě. [7]

1.4 Generický algoritmus založený na numerické shodě momentů

Numerická shoda momentů (numerical moment matching) je metoda představující velkou populaci metod s podstatně menší velikostí vzorku při zachování statistických momentů (např. průměr, rozptyl, šikmost, atd.). Velikost vzorku techniky numerické shody momentů je řádově menší než u jiných metod náhodného vzorkování, jako jsou například metoda Monte Carlo nebo metoda směrodatné odchylky (mean and sigma).

Počáteční verze techniky numerické shody momentů byla založena na vícerozměrném Newton-Raphsonově schématu, které někdy způsobuje numerickou divergenci a závislost počáteční hodnoty. K překonání těchto omezení byl generický algoritmus spojen s Newton-Raphsonovým schématem v kontextu numerické shody momentů, aby nevykazoval žádná omezení pro nepravidelné distribuce, velké velikosti nebo mnoho proměnných technických dat. Tedy technika numerické shody momentů je metoda odhadu bodů, která je použitelná pro data ve velkém měřítku generováním malé sady reprezentativních vzorků základního souboru.

Využití této metody u predikce spotřeby bylo doposud jen vzácné. Nicméně u studie, z které tato kapitola čerpala, bylo dosaženo uspokojivých výsledků. Studie

se týkala predikce spotřeby na velkém vzorku rodinných domů v oblasti Cedar Falls v Iowě a byla vytvářena roční a měsíční predikce spotřeby elektrické energie. [8]

1.5 Deep reinforcement learning

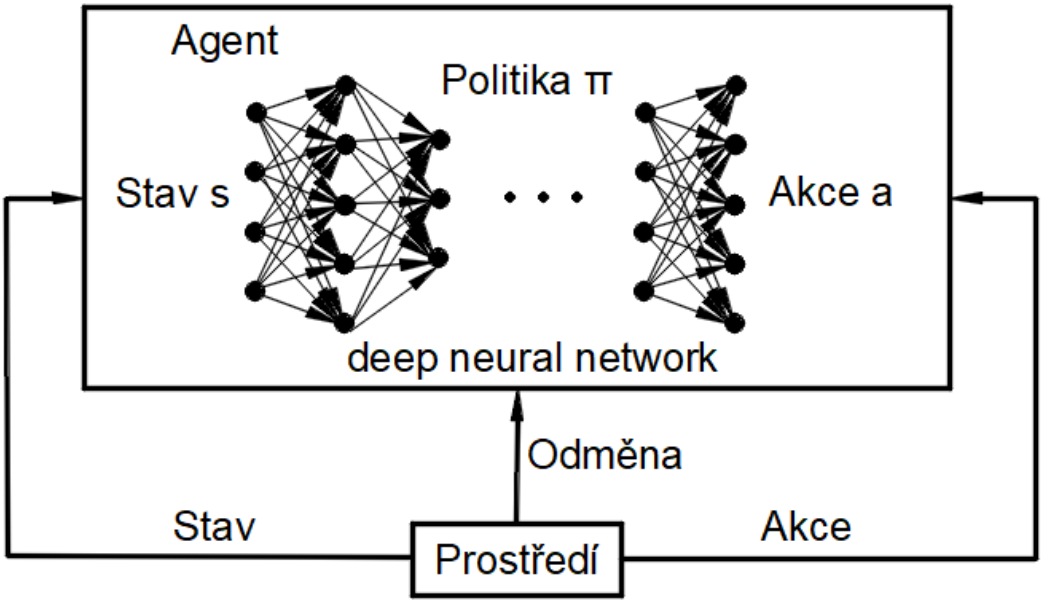
Reinforcement learning (RL) je podskupina strojového učení, která studuje, jak umělý agent provádí optimální akci na základě stavu pozorovaného prostředí pomocí odměn a trestů. Téměř všechny problémy RL lze popsat jako problémy s rozhodováním. V RL existuje pět významných konceptů a to stav (označeno jako s), akce (označeno jako a), odměna (označeno jako r), politika (označeno jako π), stejně jako hodnotová funkce (označena jako V (s)) nebo funkce akční hodnoty (označená jako Q (s, a)). Zde jsou zásady definující funkci chování agenta, v časovém kroku t agent nejdříve sleduje stav prostředí s t, provede akci a t (podle zásady π) a obdrží okamžitou odměnu r t. Okamžitou odměnou je skalární signál zpětné vazby, který může naznačovat, jak dobře si agent vede a jak daleko je od optimální politiky (označované jako π^*). Konečným cílem RL je najít optimální politiku. V algoritmu RL se pro predikci budoucí odměny používá buď hodnotová funkce, nebo funkce akční hodnoty. Hodnotová funkce označuje očekávaný celkový diskont na odměnu počínaje stavem s:

$$V(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} s_t = s \right] \quad (2)$$

kde γ je diskontní faktor. Funkce akční hodnoty označuje očekávanou celkovou diskontní odměnu počínaje stavem s a provedením akce a:

$$Q(s,a) = E \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} s_t = s, a_t = a \right] \quad (3)$$

V současné době převládající algoritmy deep reinforcement learning (DRL) integrují vnímací schopnost deep learningu a rozhodovací schopnost reinforcement learningu, čímž realizují přímou kontrolu nad komplikovanými problémy kontroly s vícedimenzionálním akčním prostorem. Algoritmy DRL nasazují nelineární aproximátory, jako jsou neuronové sítě, k odhadu hodnotové funkce (nebo funkce akční hodnoty) a aktuální politiky. A3C, DDPG a RDPG jsou tři nejčastěji používané techniky deep reinforcement learningu, které přinesly dobrý výkon v mnoha úlohách nepřetržitého řízení. [9]



Obrázek 2 Deep reinforcement learning

1.5.1 Asynchronous Advantage Actor-Critic (A3C)

V roce 2016 byla navržena Asynchronous Advantage Actor-Critic (A3C), což je rámec pro deep reinforcement založený na hercích a kriticích. A3C nasazuje současně několik agentů paralelně k výpočtu přechodu, každý s vlastním prostředím. Tito agenti provádějí asynchronní sestup gradientu za účelem optimalizace parametrů stejného globálního agenta a každý lokální agent pravidelně kopíruje parametry globálního agenta jako své vlastní parametry. Proto je A3C efektivní a lehká ve srovnání s jinými technikami DRL. [9]

1.5.2 Deep Deterministic Policy Gradient (DDPG)

Pro zlepšení trénovatelnosti existujících algoritmů DRL založených na hercích a kriticích byla navržena metoda DDPG (Deep Deterministic Policic Gradient) a existují v ní tři vylepšení. Zaprvé DDPG využívá zážitkovou vyrovnávací paměť jako paměťové zařízení k ukládání všech historických přechodů a učí se v mini-dávkách náhodně vzorkovaných z vyrovnávací paměti zkušeností, spíše než učení online. Zadruhé jsou vytvořeny samostatné cílové sítě pro herce i kritiky, aby vylepšily odhad funkce akční hodnoty. V důsledku toho existují v DDPG čtyři neuronové sítě. Zatřetí parametry dvou cílových sítí již nejsou přímo kopírovány z původních sítí, ale jsou aktualizovány tím, že se

nechají pomalu sledovat původní sítě. Tato „soft“ strategie aktualizace může omezit cílové sítě před rychlými změnami a stabilizovat tréninkový proces. [9]

1.5.3 Recurrent Deterministic Policy Gradient (RDPG)

V konvenční metodě DDPG je vícevrstvý perceptron, který se skládá z vícevrstvých plně propojených sítí, nasazen jak pro síť Actor, tak pro síť Critic. Jedním z omezení je odhad funkce akční hodnoty, který by nebyl dostatečně přesný při řešení složitého problému s řízením, kvůli nekompetentnosti vícevrstvého perceptronu v zachycení složité nelinearity. Ke snížení tohoto problému byla navržena ještě jedna metoda a to RDPG. Jediné vylepšení RDPG ve srovnání s DDPG spočívá v tom, že RDPG používá Long Short-Term Memory, což je vylepšená rekurentní neuronová síť k reprezentaci kritické funkce a odhadu funkce akční hodnoty. Tato metoda má tři důležité „brány“ (tj. vstupní brána, výstupní brána a brána pro zapomnění), které jsou navrženy pro řízení toku informací uvnitř každého paměťového bloku. Použitím Long Short-Term Memory jako kritika může RDPG přinést přesnější odhad funkce akční hodnoty než metoda DDPG, protože síť Critic pro odhad Q w (s, a) je schopna agregovat pozorování v průběhu času. Z tohoto důvodu lze poskytnout přesnější chybu pro lepší aktualizaci herce sítě. [9]

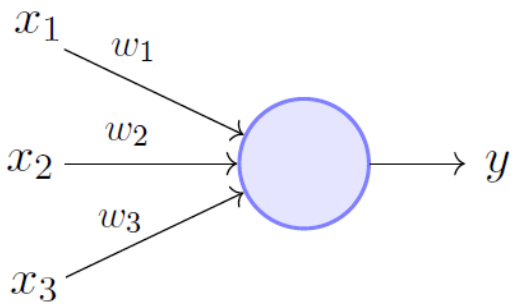
2 Levnberg-Marquardtova neuronová síť zpětného šíření

Jako první metoda, která v této práci bude aplikována na poskytnutých datech, je Levnberg-Marquardtova neuronová síť zpětného šíření. Jednoznačnou výhodou této metody je její rychlost a poměrně dobré a spolehlivé výsledky. Vyšší rychlost Levnberg-Marquardtova algoritmu je dána větším požadavkem na paměť zařízení, na kterém trénování sítě probíhá. Dříve, než ale bude popsán postup použití metody, by bylo dobré nejprve obecně přiblížit, co umělé neuronové sítě jsou a jak fungují.

2.1 Umělá neuronová síť

Původ umělých neuronových sítí je v biologické oblasti a jedná se o snahu napodobit proces učení, který probíhá v lidském mozku. Nejde tedy o napodobení lidského mozku jako takového, ale je to spíš jiný přístup k řešení složitých problémů, na které klasické metody výpočetní techniky nestačily. Základním stavebním kamenem umělých neuronových sítí je umělý neuron, podobně jako v lidském mozku, kdy se celá síť skládá z mnoha takových vzájemně propojených neuronů. Běžným využitím umělých neuronových sítí je predikce, klasifikace, aproximace nebo rozpoznávání obrazů a mnohé další možnosti.

Základním matematickým modelem umělé neuronové sítě je formální neuron. Jako značení formálního neuronu, nebo zkráceně jen neuronu, se používá Y. Neuron má n reálných vstupů, modelující dendrity a určující vstupní vektor $\mathbf{x} = (x_1,...,x_n)$. Pro vysvětlení, dendrity spolu s axony tvoří vstupní a výstupní přenosové kanály pro neuron. Vstupy do neuronu jsou potom ohodnoceny reálnými synaptickými váhami tvořícími vektor $\mathbf{w} = (w_1,\ldots,w_n)$. Ve shodě s biologickým podtextem mohou být synaptické váhy i záporné. Nejjednodušší neuronová síť obsahující pouze jeden neuron se potom nazývá perceptron. [2]



Obrázek 3 Model formálního neuronu [2]

Vnitřní potenciál neuronu Y představuje váženou sumu vstupních hodnot y_{in} definovanou následovně:

$$y_{in} = \sum_{i=1}^{n} w_i x_i \quad (4)$$

Dosazení hodnoty b, viz obrázek 3, indukuje výstupní stav y neuronu Y, který modeluje elektrický impuls axonu pro vnitřní hodnotu potenciálu y_{in}. Aktivační přenosová funkce při dosažení potenciálu b nabývá nelineárního nárůstu výstupní hodnoty $y = f(y_{in})$ a potom nejjednodušším typem přenosové funkce je ostrá nelinearita, jejíž tvar pro neuron Y má následující tvar:

$$f(y_{in}) = \begin{cases} 1 \; pro \; y_{in} \geq 0 \\ 0 \; pro \; y_{in} < 0 \end{cases} \quad (5)$$

Jak již tedy bylo nastíněno, neuronová síť se skládá z jednotlivých formálních neuronů, které jsou vzájemně spojeny tak, že výstup jednoho neuronu je vstupem do dalšího nebo dalších neuronů. Způsob vzájemného propojení neuronů potom určuje topologii sítě. Většina sítí má neurony organizovány ve vrstvách, které jsou mezi sebou vzájemně propojeny tak, že je vázán každý neuron s každým dalším. Všechny sítě obsahují vstupní a výstupní vrstvu a mezi nimi mohou být i vrstvy skryté. Existuje i případ totožných jednovrstvých sítí. Topologicky potom můžeme sítě rozdělit na přímé, mezi které patří dopředné a obousměrné, a dále na sítě rekurentní. Zpracování informace a její šíření sítí je dáno změnou stavů neuronů ležících na cestě mezi vstupní a výstupní vrstvou. Stav neuronové sítě určují stavy všech neuronů v síti a její konfiguraci představují synaptické váhy všech spojů. [2][4]

2.1.1 Dynamiky umělých neuronových sítí

Základní vlastností neuronové sítě je její vývoj v čase. Tento vývoj je dán změnou stavu neuronů a adaptací vah. S ohledem na tuto časovou změnu je dále možné rozdělit sítě do tří dynamik, ve kterých by nastávaly následující režimy práce sítě:

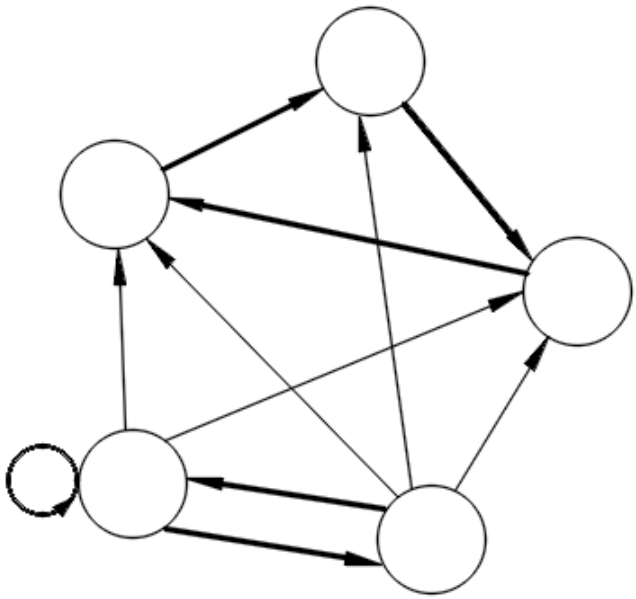
- Organizační (změna topologie)
- Aktivní (změna stavu)
- Adaptivní (změna konfigurace)

Výše zmíněné dynamiky neuronové sítě jsou zadány matematickou rovnicí a jejich počátečním stavem neboli pravidly, která určují vývoj topologie, stavu, či konfigurace sítě v čase. Využitím jednotlivých dynamik pak obdržíme různé modely neuronových sítí, vhodných pro řešení různých úloh. [2]

2.1.1.1 Organizační dynamika

Charakteristické pro organizační dynamiku je připouštění možných změn v topologii sítě. Změna v topologii se uplatňuje v rámci adaptivního režimu a to tak, že v případě potřeby je možné síť rozšiřovat o další neurony a spoje. Je však převážně předpokládána pevná architektura neuronové sítě, tedy neproměnlivá v čase. Potom lze rozlišit dva druhy architektury, a to cyklická, neboli rekurentní, a acyklická, také zvaná jako dopředná síť.

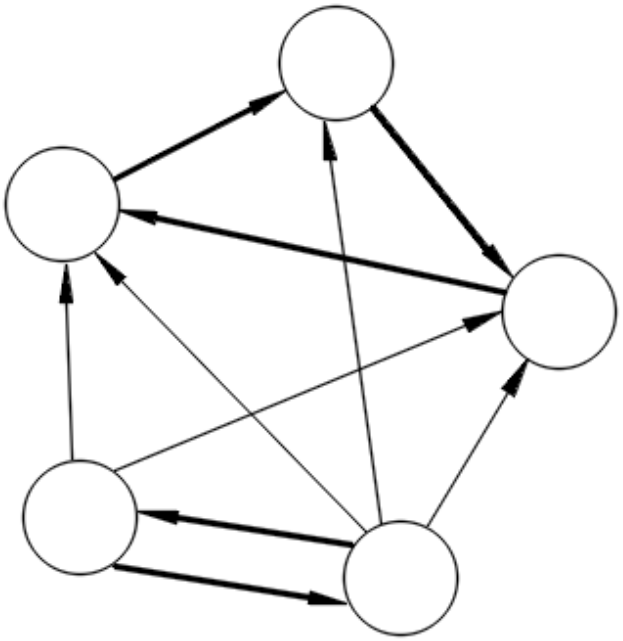
U cyklické (rekurentní) topologie, jak už název napovídá, se jedná o skupinu neuronů, která je v sítí spojena v kruhu a tvoří takzvaný cyklus. Tedy vstupem jednotlivých neuronů je výstup neuronu předchozího, kde se informace předávají tak dlouho, než se výstup posledního neuronu stane opět vstupem neuronu prvního. Jako příklad může posloužit zpětná vazba neuronu, kde jeho výstup je zároveň i vstupem. Nejčastější variantou potom bývá, že výstup libovolného neuronu je vstupem každého neuronu. Pro představu jsou na obrázku níže ilustrovány možné cykly, které přicházejí v úvahu. [2]



Obrázek 4 Cyklická architektura organizační dynamiky [2]

Opakem k sítím cyklickým jsou potom sítě acyklické (dopředné), v nichž všechny cesty vedou pouze jedním směrem. Tato síť se potom organizuje do jednotlivých neuronových vrstev, které jsou uspořádány vedle sebe a platí, že spoje mezi neurony vedou pouze z nižších vrstev směrem k vyšším, nicméně je ale také možné některé vrstvy přeskočit. Konkrétním příkladem takovéto sítě by byla vícevrstvá neuronová síť, kterou se

v této práci budeme dále ještě zabývat. Obecný příklad dopředné sítě je na obrázku níže, kde je znázorněna nejdelší možná cesta. [2]



Obrázek 5 Acyklická architektura organizační dynamiky [2]

2.1.1.2 Aktivní dynamika

Aktivní dynamikou je specifikován počáteční stav sítě a změny stavů v závislosti na čase při pevné topologii a konfiguraci. Na začátku jsou na vstup sítě nastaveny stavy vstupních neuronů a ostatní neurony zůstávají v počátečním stavu. Vstupní prostor neuronové sítě je tvořen všemi možnými vstupy neboli stavy sítě a po vyvolání stavu sítě probíhá vlastní výpočet. Předpokládá se spojitý vývoj stavů neuronové sítě a při závislosti na čase by se tento vývoj vyjádřil diferenciální rovnicí. Nicméně pro zjednodušení se většinou uvažuje čas diskrétní, kdy na začátku se síť nachází ve stavu 0 a potom se krokově mění. V každém kroku je potom vybrán jeden neuron, který mění svůj stav s ohledem na své vstupy. Výstupy se aktualizují na vstupy sousedních neuronů. Obvykle předpokládáme takovou aktivní dynamiku, že výstup sítě je po nějakém čase konstantní a neuronová síť tak realizuje nějakou funkci. [2]

Aktivní dynamika též určuje funkci jednoho neuronu ve tvaru matematického vzorce, který je potom pro všechny nevstupní neurony v síti stejný a zavádí se pojem aktivační, respektive **přenosové funkce**:

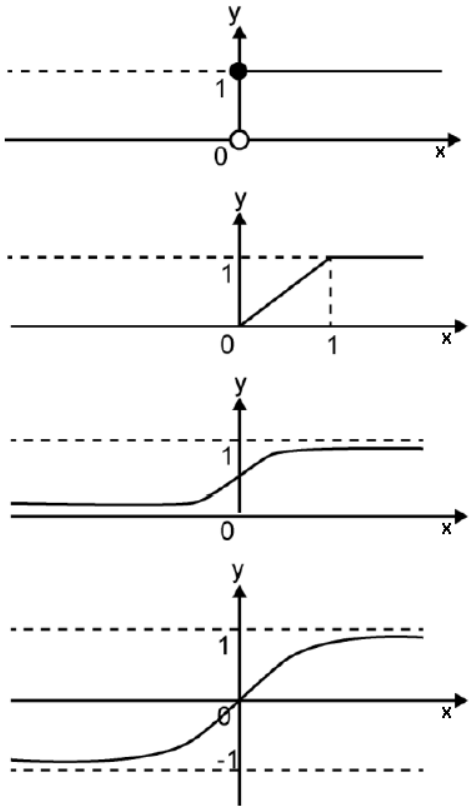
$$f(x) = \begin{cases} 1 \; když \; x \geq 1 \\ 0 \; když \; x < 0 \end{cases} \quad \textbf{(6)}$$

$$f(x) = \begin{cases} 1 \; pro \; x \geq 1 \\ x \; pro \; 0 \leq x \geq 1 \\ 0 \; x < 0 \end{cases} \quad \textbf{(7)}$$

$$f(x) = \frac{1}{1 + e^{-x}} \quad \textbf{(8)}$$

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \quad \textbf{(9)}$$

Rovnice 6 popisuje ostrou nelinearitu (skoková funkce), dále rovnice 7 popisuje po částech lineární funkci, rovnice 8 je standardní (logická) sigmoida a rovnice 9 je hyperbolický tangens. Ve stejném pořadí jsou jednotlivé **aktivační (přenosové) funkce** znázorněné i na obrázku níže. [2] [5]



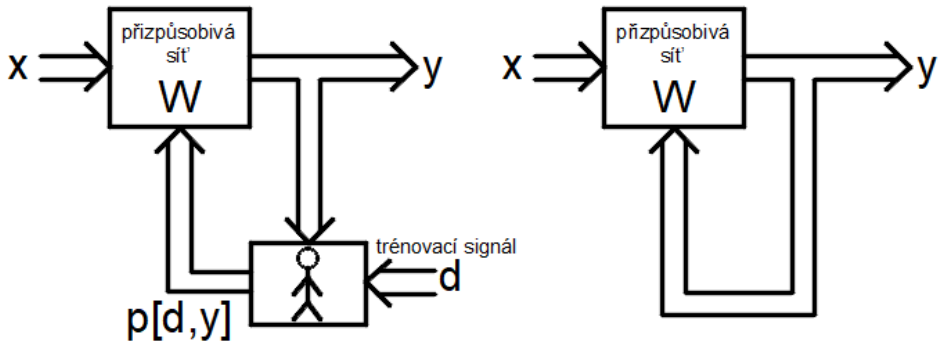
Obrázek 6 Příklady aktivačních (přenosových) funkcí [6]

2.1.1.3 Adaptivní dynamika

Adaptivní dynamika souvisí se změnami v konfiguraci sítě a specifikuje, jakým způsobem se mění váhové hodnoty na spojeních mezi jednotlivými neurony v závislosti na čase. V tomto režimu se tedy váhy všech spojů v síti nastaví na počáteční konfiguraci a následně probíhá vlastní adaptace. Dále je to podobně jako v aktivní dynamice, kdy můžeme uvažovat spojitý model, ale běžně se využívá diskrétního času adaptace. Z aktivního režimu víme, že funkce sítě v aktivním režimu závisí na konfiguraci. Cílem tedy je nalézt takovou konfiguraci sítě, která by v aktivním režimu realizovala předepsanou funkci. Takže na rozdíl od aktivního režimu sítě, který se využívá k vlastnímu výpočtu funkce sítě pro daný vstup, tak adaptivní režim slouží k učení této funkce. Požadované funkce se obvykle dosahuje zadáním tréninkové množiny, ze které dochází k adaptaci sítě a organizování tréninkových vzorů a v tomto bodě se adaptivní dynamika rozděluje ještě dle přístupu v učení na buďto **učení s učitelem** nebo **učení bez učitele**. [2]

Při **učení s učitelem** předpokládáme, že v každém okamžiku, kdy je vstup aplikován, učitel poskytuje systému požadovanou zpětnou vazbu. To je znázorněno na obrázku 7. Vzdálenost p[d,y] mezi skutečnou a požadovanou odezvou slouží jako míra chyby a slouží k externí korekci parametrů sítě. Jelikož předpokládáme nastavitelné váhy, může učitel uplatnit schéma odměny a trestu, aby přizpůsobil váhovou matici sítě W. Například při učení klasifikace vstupních vzorců nebo situací, kdy známe odpovědi, je možné chybu použít k úpravě vah tak, aby se snižovala. Pro tento druh učení je vyžadována tréninková sada, jež má nadefinovány vstupní a výstupní vzorce. [6]

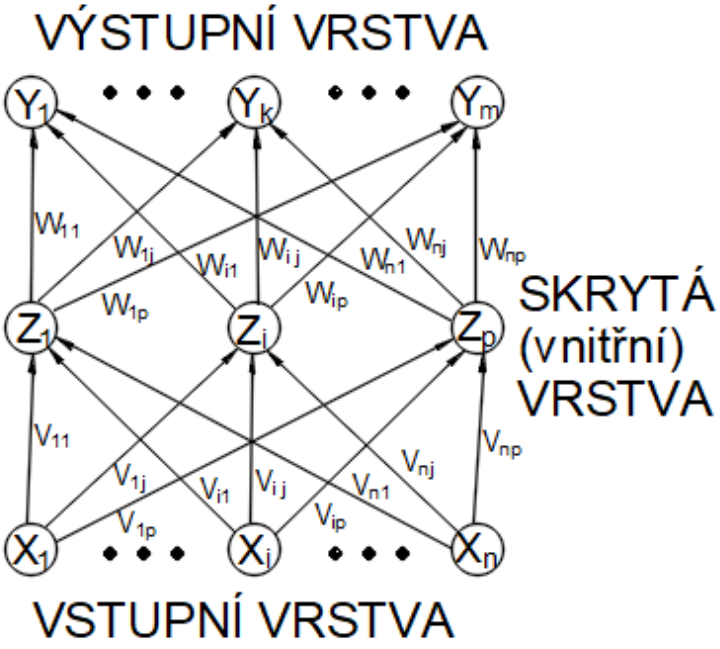
Učení bez učitele, nazývané také jako nekontrolované učení, je druh učení, při kterém jsou poskytnuta trénovací data, ale už nedochází k ovlivňování korekce chyb. Tedy váhy sítě se nastaví tak, aby výstup byl konzistentní. To znamená, že síť poskytuje stejnou odezvu na budící signál při stejných nebo podobných vektorech vstupu. [6]



Obrázek 7 Schéma způsobu učení (a) s učitelem, (b) bez učitele [6]

2.1.2 Vícevrstvé neuronové sítě

Vícevrstvé neuronové sítě se skládají ze vstupní a výstupní vrstvy a mezi těmito vrstvami jsou ještě takzvané vrstvy skryté. Jedná se o nejrozšířenější typ neuronové sítě. Přenosová funkce pro tento typ sítě je sigmoidní. Jak by taková síť vypadala, přibližuje obrázek níže, kde neurony skryté vrstvy jsou označeny $\mathbf{Z_i}$, i = 1,…,p, vstupní vrstva $\mathbf{X_i}$, i = 1,…,p a výstupní vrstva $\mathbf{Y_k}$, k = 1,…,k. Minimální množství skrytých vrstev je jedna, ale může jich být více. U vícevrstvé neuronové sítě platí, že každý neuron nižší vrstvy je spojen se všemi neurony z vrstvy vyšší, což je patrné z obrázku níže. [2]



Obrázek 8 Vícevrstvá neuronová síť [2]

Vlastnosti sítě:

- Síť je tvořena z jednotlivých neuronů
- Počet vstupních neuronů je dán počtem vstupů matematického modelu
- Počet výstupních neuronů je ovlivněn kódováním výstupu
- Počet skrytých neuronů je volen s ohledem na složitost úlohy
- Standardně je síť učena za pomoci učení s učitelem
- Přenosová funkce je většinou sigmoidální
- Učícím algoritmem je nejčastěji algoritmus Backpropagation Error
- Poměrně dlouhá doba učení [5]

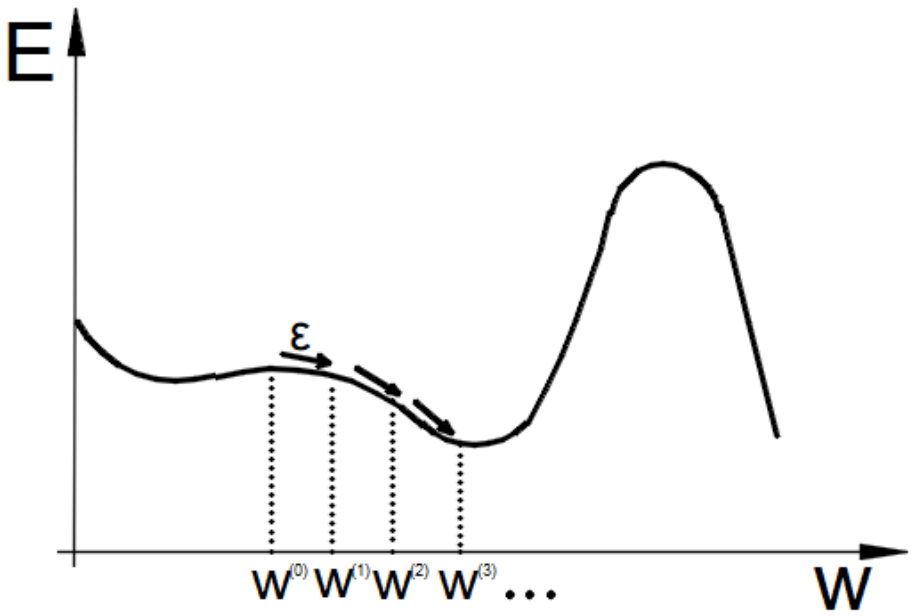
2.2 Metoda zpětného šíření (backpropagation)

Jedná se o metodu zpětného šíření chyb, takzvaná backpropagation a je to adaptační algoritmus, který se využívá až v 80 % všech aplikací neuronových sítí. Algoritmus je rozdělen do tří fází a to dopředné (feedfoward) šíření vstupního signálu tréninkového vzoru, zpětné šíření chyby a aktualizace váhových hodnot na spojeních. V rámci dopředného šíření je vyslán vstupní signál (x_i) na každý neuron ve vstupní vrstvě $\mathbf{X_i}$, i = 1,…,p a je zprostředkován přenos ke všem neuronům vnitřní vrstvy Z_1,…,Zp. Neurony ve vnitřní vrstvě počítají svou aktivaci (y_k), jenž odpovídá jeho skutečným výstupům k-tého neuronu po předložení vstupního vzoru. Tak získáme odezvu neuronové sítě na vstupní podnět daný excitací neuronů vstupní vrstvy, to je stejné jako šíření signálů v biologickém systému. Důležitý je proces stanovení synaptických vah, který souvisí s adaptací neuronové sítě. Metoda adaptace spočívá v opačném síření informace směrem od vrstev vyšších k vrstvám nižším. Během adaptace neuronové sítě metodou backpropagation jsou srovnávány vypočítané aktivace s definovanými výstupními hodnotami pro každý neuron ve výstupní vrstvě a pro každý tréninkový vzor. Díky tomuto srovnání je definována chyba neuronové sítě, která se může dále šířit od aktuálního neuronu do všech předchozích vrstev. [2]

Další důležitou schopností metody je generalizace neboli zobecnění nad naučeným materiálem. To je schopnost odvodit i jevy, které nebyly předmětem učení, ale díky zkušenosti s předchozími tréninkovými daty je síť schopna vyprodukovat smysluplný výsledek, toho je dosaženo zafixováním vzorků z trénovacích dat do neuronové sítě ve formě synaptických vah. [2]

Důležitým parametrem pro adaptaci a generalizaci je vhodné zvolení počtu skrytých

vrstev, které by mělo odpovídat složitosti řešeného problému. To znamená správné zvolení tréninkových vzorů, jejich vstupů a výstupů a správnou strukturu vztahů, které slouží k popisu problému. Tedy malá síť není vhodná pro řešení složitých problémů, protože by se taková síť při učení mohla zastavit na nějakém lokálním minimu. Rozšiřování sítě zvyšuje šanci nalezení minima globálního, samozřejmě s narůstající robustností sítě stoupá i výpočetní náročnost a učení sítě pak může zabrat mnoho času. Na druhou stranu je třeba dávat pozor, aby nedošlo k takzvanému přeučení sítě (overfitting), u kterého dochází ke špatné generalizaci a to tím, že jsou až příliš zobecněny tréninkové vzory včetně jejich nepřesností a chyb. Proto je tedy potřeba hledat optimální velikost sítě, abychom se mohli vyhnout oběma zmíněným problémům. Minimalizace chyby se řeší gradientní metodou, v rámci které je nutná diferencovatelnost chybové funkce E(w). Hledáme takovou konfiguraci, pro kterou je chybová funkce minimální. Začíná se s náhodnou konfigurací w(0) a sestrojíme v tomto bodě ke grafu chybové funkce tečný vektor grad(w(0)) a posuneme se ve směru tohoto vektoru dolů o ε. Získáme tak novou konfiguraci w(1) = w(0) + dw(1). Tímto se zmenší chybová funkce a celý proces se opakuje tak dlouho, než se limitně dosáhne do lokálního minima chybové funkce. [3]



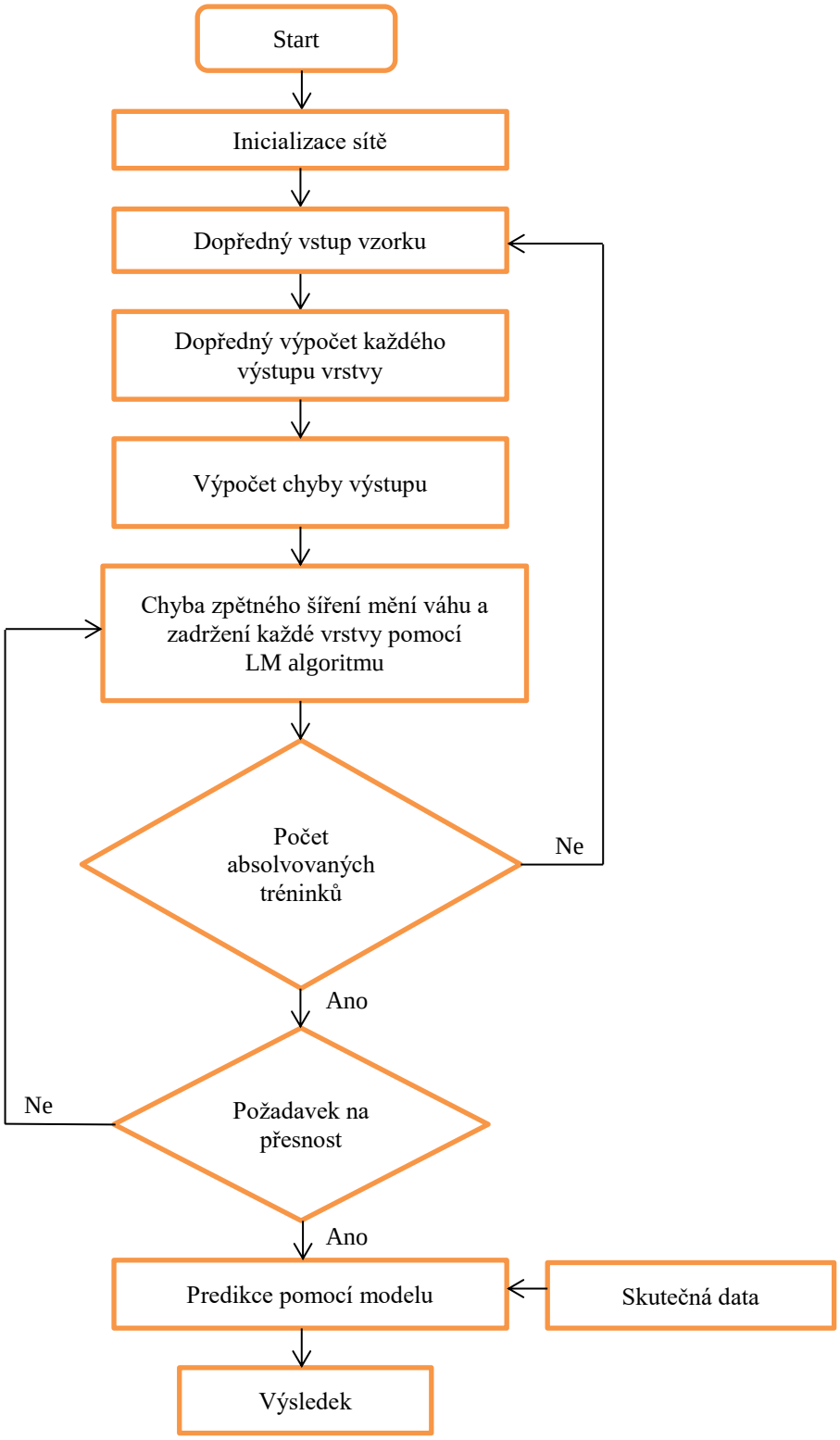
Obrázek 9 Gradientní metoda [3]

2.3 Levenberg-Marquardtův algoritmus

Jak už bylo zmíněno v první kapitole této práce, rozšířenou a vylepšenou variantou standardní backpropagation metody je Levenberg-Marquardtův algoritmus. Jedná se o neuronovou síť, která je využívána k nelineárním cvičení algoritmů a v rámci které je kombinována metoda gradientního sestupu a Kvazi-Newtonova metoda, aby byla zajištěna lokální vysoká rychlost konvergence a udržoval se lepší celkový výkon. Kvazi-Newtonova metoda tedy obecně slouží k vyhledávání nul či lokálních maxim a minim funkcí. [7]

Základní myšlenkou Levenberg-Marquardtova algoritmu je, že každá iterace podél jednoho negativního směru gradientu není delší, než je nutné, a umožňuje vyhledat chybu ve směru zhoršování. Váhy sítě lze navíc optimalizovat pomocí adaptivního přizpůsobení mezi metodou nejstrmějšího gradientního klesání a metodou Gauss-Newton, která umožňuje efektivní konvergenci sítě, což výrazně zlepšuje rychlost konvergence a zobecnitelnost dat. [7]

Levenberg-Marquardtova neuronová síť zpětného šíření tedy na rozdíl od klasické neuronové sítě zpětného šíření řeší problém s pomalou konvergencí dat. Také řeší fluktuaci a oscilaci během tréninkového procesu, kdy byl model snadno zachycen místními minimy a bylo obtížné určit strukturu sítě. Tréninkový proces Levenberg-Marquardtovy neuronové sítě zpětného šíření dále již jen zkráceně LM-BP (Levenberg-Marquardt backpropagation) neuronové sítě je znázorněn na schématu na následující straně. [7]

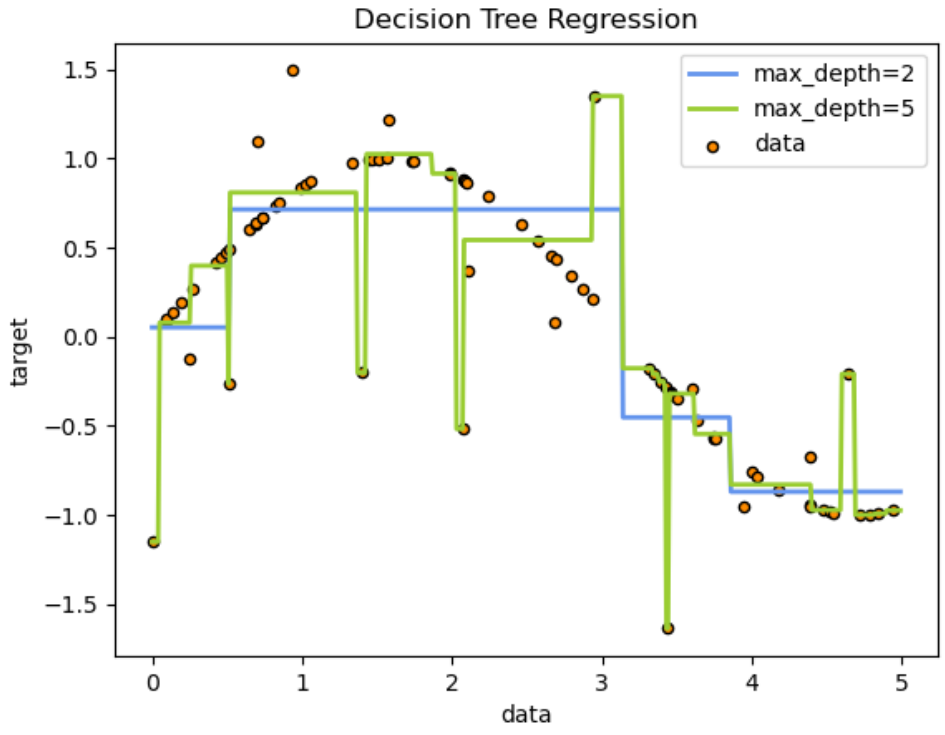

Start
Inicializace sítě
Dopředný vstup vzorku
Dopředný výpočet každého výstupu vrstvy
Výpočet chyby výstupu
Chyba zpětného šíření mění váhu a zadržení každé vrstvy pomocí LM algoritmu
Počet absolvovaných tréninků
Ne
Ano
Požadavek na přesnost
Ne
Ano
Predikce pomocí modelu
Skutečná data
Výsledek

3 Metody rozhodovacích stromů

Jako další metoda pro vytváření predikčního modelu se nabízí výpočetní metoda rozhodovacích stromů (decision tree). Tato metoda je vhodná díky dostupnosti softwaru, relativní jednoduchost ovládání a rozumných požadavků na výpočetní výkon zařízení, na kterém jsou modely vytvářeny. V této práci byly vybrány dva konkrétní algoritmy z palety nástrojů knihovny pro strojové učení Scikit-learn a to Gradient Tree Boosting method a AdaBoost method. Jedná se o vylepšené algoritmy klasické metody decision tree. Dále bude nastíněna základní teorie k těmto metodám.

3.1 Rozhodovací stromy

Rozhodovací stromy jsou neparametrickou metodou strojového učení typu učení s učitelem a využívají se ke klasifikaci a regresi. Cílem je vytvoření modelu, který předpovídá hodnotu cílové proměnné učením jednoduchých pravidel rozhodování odvozených z datových funkcí. Strom lze považovat za konstantní aproximaci po částech. Například v níže uvedeném příkladu se rozhodovací stromy učí z dat aproximovat sinusovou křivku pomocí sady rozhodovacích pravidel if-then-else. Čím hlubší je strom, tím složitější jsou pravidla rozhodování a tím je model lepší. [10]



Obrázek 10 Aproximace sinusové křivky [10]

Výhody rozhodovacích stromů:

Mezi výhody rozhodovacích stromů patří, že jsou snadno pochopitelné a interpretovatelné. Stromy lze také vizualizovat. Metoda vyžaduje malou přípravu dat, oproti jiným metodám, které často vyžadují normalizaci dat a je u nich potřeba vytvořit fiktivní proměnné a odstranit prázdné hodnoty. Predikce dat je logaritmická k počtu datových bodů použitých k trénování stromu. Dále je tu schopnost zvládnout problémy s více výstupy.

Jako výhodu lze také považovat, že rozhodovací stromy používají model bílé skříňky. Pokud je daná situace v modelu pozorovatelná, lze vysvětlení podmínky snadno vysvětlit booleovskou logikou. Naproti tomu v modelu černé skříňky (např. v umělé neuronové síti) může být výsledky obtížnější interpretovat.

Také je tu možnost ověřit model pomocí statistických testů. To umožňuje zohlednit spolehlivost modelu. Funguje dobře, i když jeho předpoklady jsou trochu porušeny skutečným modelem, ze kterého byla data generována. [10]

Nevýhody rozhodovacích stromů:

Při učení rozhodovacího stromu může docházet k příliš složité struktuře stromů, které data dobře nezobecňují. Tomu se říká overfitting. Aby se tomuto problému zabránilo, jsou nezbytné mechanismy, jako je prořezávání, nastavení minimálního počtu vzorků požadovaných v uzlu listu nebo nastavení maximální hloubky stromu.

Rozhodovací stromy mohou být nestabilní, protože malé odchylky v datech mohou mít za následek generování zcela jiného stromu. Předpovědi rozhodovacích stromů nejsou ani plynulé, ani spojité, ale po částech konstantní aproximace, jak je vidět na výše uvedeném obrázku 10. [10]

3.1.1 AdaBoost

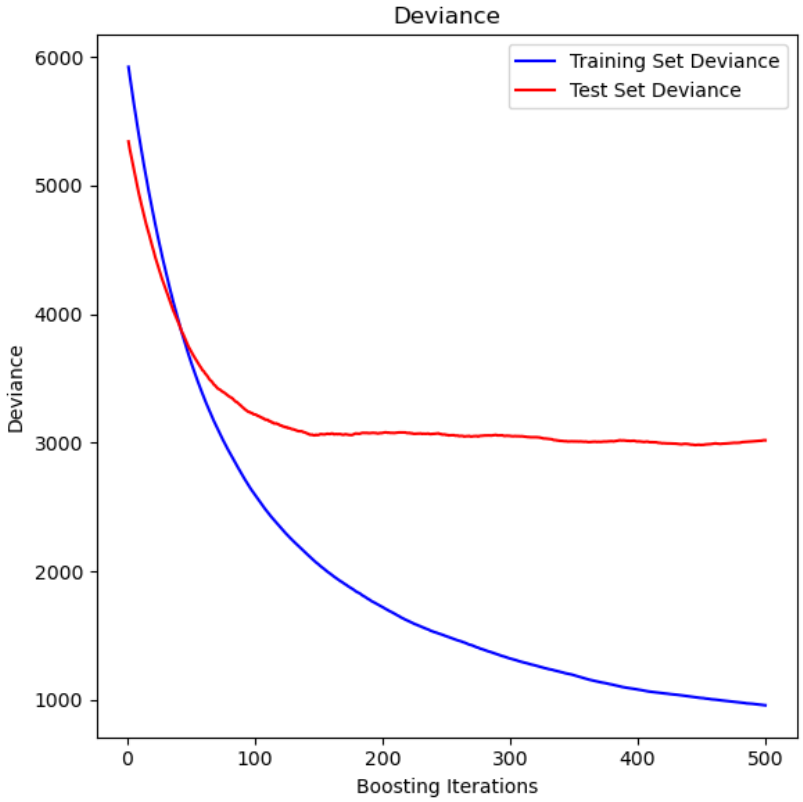
Základním principem AdaBoostu je přizpůsobit posloupnost slabých studentů (tj. modelů, které jsou jen o něco lepší než náhodné hádání, například malé rozhodovací stromy) na opakovaně upravené verze dat. Předpovědi ze všech z nich se poté spojí prostřednictvím hlasování vážené většiny (nebo součtu), aby se dosáhlo konečné predikce. Úpravy dat při každé takzvané boostovací iteraci spočívají v aplikaci vah $\omega_1, \omega_2, \ldots, \omega_N$ ke každému z tréninkových vzorků. Zpočátku jsou tyto váhy nastaveny na $\omega_1 = 1/N$, takže první krok jednoduše trénuje slabého studenta na původních datech. U každé po sobě jdoucí iteraci se vzorové váhy individuálně upravují a učící algoritmus se znovu použije

na vážená data. V daném kroku se u těch tréninkových příkladů, které byly nesprávně předpovězeny zesíleným modelem vyvolaným v předchozím kroku, zvýšila jejich váha, zatímco u těch, které byly předpovídány správně, se váhy snížily. Jak iterace pokračují, příklady, které je obtížné předvídat, získávají stále větší vliv. Každý následující slabý žák je tak nucen soustředit se na příklady, které minulé v pořadí chyběly. [10]

3.1.2 Gradient Tree Boosting

Gradient Tree Boosting nebo Gradient Boosted Decision Trees je zevšeobecnění posilování libovolné diferencovatelné ztrátové funkce. Tato metoda je přesná a její efektivní běžný postup lze použít pro regresní i klasifikační problémy v různých oblastech, v rámci této práce nás potom především zajímají možnosti řešení regresních problémů.

Obrázek níže ukazuje výsledky aplikace GradientBoostingRegressor se ztrátou nejmenších čtverců a 500 základních studentů na příkladovou datovou sadu. Graf ukazuje trénování a test error při každé iteraci. Chyba trénování se při každé iteraci ukládá do proměnné train_score_attribute našeho gradientního modelu. Testovací chybu při každé iteraci lze získat pomocí metody staged_predict, která vrací generátor, který poskytuje předpovědi v každé fázi. Výsledky jako tyto mohou být použity pro stanovení optimálního počtu stromů (tj. N_estimators) pro včasné zastavování trénování. [10]



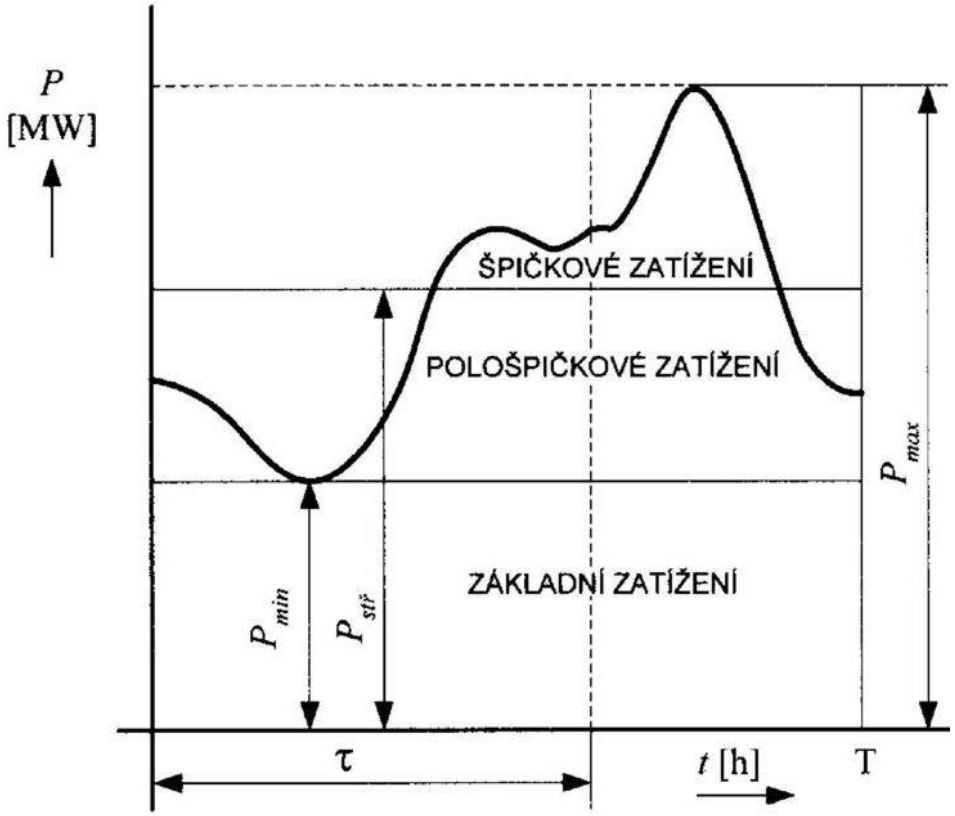
Obrázek 11 příklad Gradient tree boosting [10]

4 Diagram zatížení a jeho pokrývání

Cílem této práce je predikování spotřeby elektrické energie a s tím souvisí takzvaný diagram zatížení, který popisuje průběh odebíraného činného elektrického výkonu na čase. Obsah vytyčený takovýmto grafem potom reprezentuje celkovou spotřebovanou elektrickou energii. Běžné dělení diagramů zatížení potom podléhá většinou různým časovým úsekům, po které odebíraný výkon sledujeme, obvyklé jsou potom denní, týdenní, měsíční a roční diagramy zatížení. Diagram zatížení soustavy udává následující vztah:

$$P = f(t) \quad \textbf{(10)}$$

Celá tato problematika je způsobena omezenými možnostmi skladování elektrické energie a to zapříčiňuje, že výroba elektrické energie musí nutně reagovat na její spotřebu. V praxi se pak využívá statisticky získaných údajů z provozu, ze kterých se tyto diagramy stanovují. Průběh zatížení se samozřejmě bude měnit pro různá období a různé dny. Bude záležet na tom, zda se jedná například o pracovní dny, volné dny, o letní či zimní období atd. [11]



Obrázek 12 Denní diagram zatížení [11]

V diagramu zatížení, který charakterizuje obrázek 11, je možné vyčlenit tři typické oblasti. První oblastí je takzvané základní zatížení. Toto zatížení se v průběhu uvažovaného období nemění, jeho spodní hranicí je nulové zatížení a horní hranicí minimální zatížení. Po celou tuto dobu jsou v provozu zdroje, které jsou obtížně regulovatelné, a pracují proto v nepřetržitém provozu. Jedná se o elektrárny jaderné, tepelné a vodní průtočné. Tyto zdroje samozřejmě dle potřeby mohou pracovat pouze na snížený výkon a nemusí tak docházet k jejich úplnému odstavení. V druhé oblasti diagramu se nachází pološpičkové zatížení. To je nad základním zatížením a shora je omezeno velikostí středního výkonu $P_{stř}$. V tomto období je potřeba zvýšit výkon již pracujících elektráren a případně zapojit akumulační vodní elektrárny. Poslední oblastí je potom špičkové zatížení, které pokrývá maximální odběr. V tento čas startují vodní přečerpávací elektrárny a tepelné parní elektrárny se spalovacími turbínami. Tyto elektrárny jsou většinou dálkově řízeny a jejich provoz většinou bývá plně automatizován, a tedy dokáží plného výkonu dosáhnout velmi rychle v řádu několika minut. [11]

V ideálním případě bychom chtěli, aby byl odběrový diagram vyrovnaný, toho však bohužel nelze při měnící se zátěži dosáhnout. Na odběrovém diagramu se proto formují dva extrémy a to maximum (špička) a minimum. Špička je maximální hodnota požadovaného výkonu P_{max}, určuje velikost výkonu, který musí být v energetické síti k dispozici pro pokrytí maximálního odběru. Čím je špička strmější, tím je potřebná doba najetí zdrojů kratší, tedy strmost špičky definuje druh energetických zdrojů pro pokrývání špičkového zatížení. Oproti tomu zase stojí minimum, které představuje minimální odebíraný výkon v soustavě. Jeho velikost se liší zejména podle ročního období. Z pohledu energetiky je minimum stav, který značně komplikuje provoz celé sítě. Odlehčováním sítě, kupříkladu vlivem Ferrantiho jevu, vznikají v síti přepětí, a proto je nutné odstavovat některé elektrárny. Z toho důvodu se stanovují jak ekonomická opatření, která mají odběratele motivovat k odběru v době, kdy je to potřeba, tak se vytvářejí i technická opatření jako jsou operativní využívání HDO (Hromadné Dálkové Ovládání). Výsledkem těchto opatření by potom mělo být vyrovnávání odběrového diagramu. K vyrovnávání diagramu velkou měrou přispívají přečerpávací vodní elektrárny, které při přebytku energie v soustavě spotřebovávají energii na čerpání vody ze spodní nádrže do horní a v době špičkového zatížení energii vyrábějí a dodávají do sítě. [11]

Celková spotřebovaná energie lze vyjádřit následujícím vztahem:

$$W = \int_0^T P dt \quad \textbf{(11)}$$

Dalším běžně zaváděným termínem je doba využití maxima značená τ, je to doba, za kterou bychom při maximálním výkonu odebrali stejné množství elektrické energie jako při skutečném výkonu za celé sledované období. Platí tedy, že čím více se τ blíží skutečnému uvažovanému období (např. den), tím je odběrový diagram vyrovnanější. Z doby využití maxima může vycházet další vztah pro spotřebovávanou energii a to:

$$W = P_{max} \cdot \tau \quad \textbf{(12)}$$

Celkovou spotřebovanou energii ještě lze vyjádřit přes střední výkon $P_{stř}$. Jedná se o stálý výkon, při kterém odebereme stejné množství elektrické energie za uvažované období jako při skutečném časově proměnném výkonu. Vzájemný vztah by vypadal následovně:

$$W = P_{stř} \cdot T \quad \textbf{(13)}$$

Také se zavádí pojem vzájemného vztahu mezi středním a maximálním výkonem zvaným jako zatěžovatel z:

$$z = \frac{P_{stř}}{P_{max}} \quad \textbf{(14)}$$

Čím více se zatěžovatel blíží jedné, tím je diagram zatížení vyrovnanější. [11]

4.1 Problémy při pokrýváni diagramu zatížení

Z hlediska pokrývání diagramu zatížení ne všechny zdroje elektrické energie, které v soustavě provozujeme, jsou stabilní, a tak vnášejí do celé problematiky prvek nejistoty. Jedná se především o některé zdroje z řad obnovitelných zdrojů, jmenovitě pak fotovoltaické elektrárny a větrné elektrárny. Problémem těchto elektráren je jejich nekonzistentnost a neschopnost dodávat stabilní výkon do elektrizační soustavy, protože jejich provoz je závislý na klimatických podmínkách a ty jsou v čase proměnlivé. To má za následek nemožnost uplatnění zmíněných zdrojů v základním pásmu diagramu zatížení. Jako negativní důsledek této nekonzistentnosti lze potom chápat potřebu disponovat jinými zdroji elektrické energie o srovnatelném výkonu, které dokáží kompenzovat jejich proměnlivost dodávaného výkonu. Samozřejmě elektrárny, které takto kompenzují, jsou potom elektrárny, které jsou v současnosti často chápány jako neekologické, neboť jsou založené například na bázi spalování pevných hmot.

Řešením, které má obecně dobrý vliv na pokrývání diagramu zatížení a jeho

eventuální vyrovnávání, je přítomnost bateriových uložišť. Historicky akumulaci energie obstarávaly přečerpávací vodní elektrárny, které jistě své zastoupení v pokrývání špiček diagramů zatížení mít stále budou. Dnes jsou na vzestupu také bateriová uložiště. Hlavním důvodem pro možnost otevření tohoto nového trendu je snižující se cena bateriových uložišť a také jejích větší flexibilita. Intenzivní rozvoj akumulace je jedním z klíčových prvků pro intenzivnější využití solárních a větrných elektráren a to proto, aby bylo možné dodávat elektřinu spotřebitelům i v době, kdy tyto zdroje nevyrábí. Jedním z nejdůležitějších parametrů akumulačních zařízení je jejich celková účinnost, tedy poměr odebraného a později dodávaného množství elektrické energie. Bateriová uložiště, na rozdíl od přečerpávacích vodních elektráren, mají jednoznačně výhodu v jejich možnosti snadného škálování výstavby vzhledem k narůstajícím potřebám. Účinnost uložišť se pohybuje kolem 82 %, což je jen mírně nad účinností přečerpávacích vodních elektráren. [12]

Dalším velkým hráčem v otázce pokrývání diagramu zatížení, výkyvech v něm a otázky, zda vůbec bude dostatek elektrické energie, je rozvíjející se elektromobilita. Tento trend narůstá exponenciální rychlostí a může v budoucnu způsobit značné problémy v přenosové i distribuční soustavě. Nicméně elektromobilita jako taková představuje v současné době spojovací článek mezi třemi klíčovými trendy. Prvním je moderní energetika, která by byla založena na decentralizované výrobě, chytrých sítích a využívání obnovitelných zdrojů a akumulace energie. Dalším trendem je ekologizace a zefektivnění dopravy a třetím je aktivita chytrých měst v oblasti zlepšování životního prostředí a kvality ovzduší a s tím spojeného managementu dopravy. [13]

5 Aplikace neuronových sítí na predikci spotřeby

V následující kapitole bude popsáno, jakým způsobem byla zpracována poskytnutá naměřená data spotřeby z různých míst elektrizační soustavy a jakých bylo následně dosaženo výsledků. Na poskytnutá data byly aplikovány umělé neuronové sítě, které napodobují strukturu a funkci lidského mozku prováděním nelineárního zpracování informací a paralelních výpočtů ve velkém měřítku s distribuovanou úložnou kapacitou. Při práci s různými typy informací se neuronové sítě samy učí a neustále upravují své systémy, aby se přizpůsobily proměnlivému prostředí. Také se dokáží přizpůsobit mnoha nepravidelným a nepřesným pravidlům, které v elektrizační síti často nastávají. Dalšími nezpochybnitelnými výhodami je jejich schopnost zapamatovat si informace, samoučení či uvažování o znalostech a optimalizace informací.

Nejdůležitější charakteristikou neuronových sítí ve srovnání s ostatními algoritmy je jejich samoadaptibilita, a proto by neuronové sítě mohly hrát významnou roli pro predikci spotřeby elektrické energie. Pokud máme pečlivě nasbírané informace o spotřebě elektrické energie pro jednotlivé objekty, lze pomocí neuronové sítě vypočítat spotřebu elektrické energie v následujících časových úsecích jako je následující den, týden, měsíc. Čím širší naměřený časový interval spotřeby máte, tím dál do budoucnosti můžete hledět a tím přesnější predikci dostanete. V této práci jsou navržené prediktivní modely založeny na Leveberg-Marquardtově algoritmu neuronových sítí zpětného šíření.

5.1 Motivace

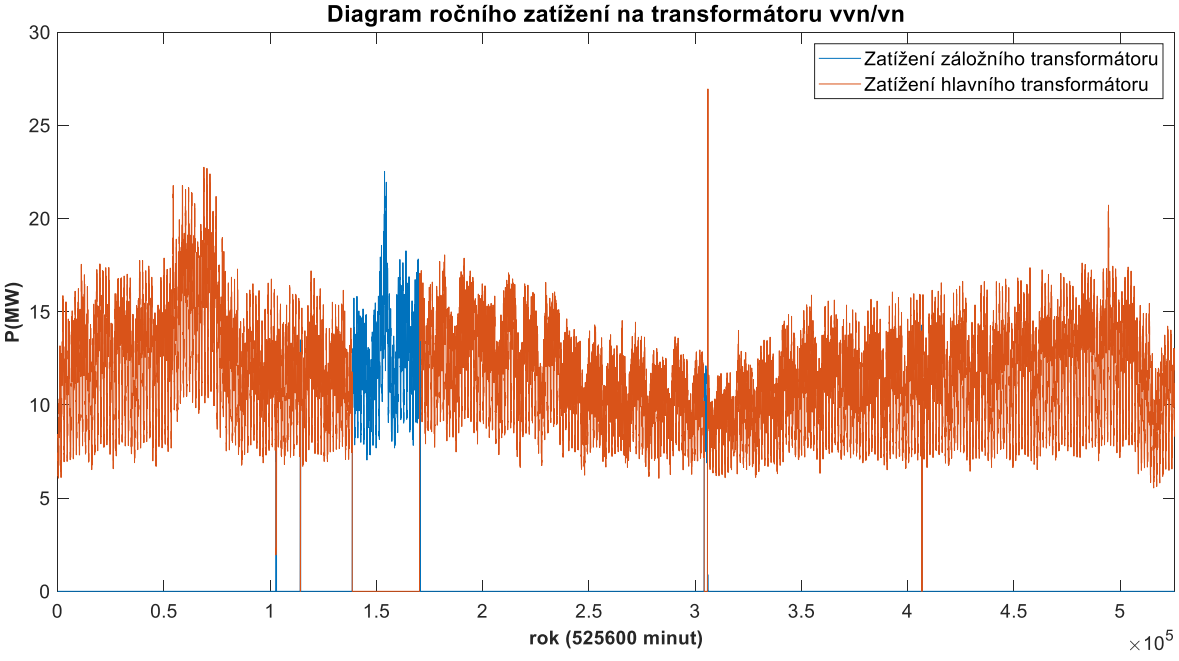
V důsledku populačního růstu a ekonomického rozvoje se za poslední tři desetiletí postupně zvyšuje spotřeba elektrické energie. Rostoucí využívání energie potom vede k problémům jako je zhoršování životního prostředí. S rostoucími nároky na komfort také vzrůstá spotřeba elektrické energie v domácnostech. Například novým lokálním trendem je rapidní nárůst poptávky po klimatizačních zařízeních a ty jsou v otázce spotřeby energie velkým hráčem. Vzhledem k rostoucímu vyčerpání neobnovitelných zdrojů energie, jako je ropa a uhlí, se elektřina stává stále důležitější otázkou každodenního lidského života a výroby. Svět se v současné době snaží transformovat na alternativní zdroje, jako je jaderná energie, vodní energie a větrná či solární energie. Nicméně celková spotřeba se každým rokem zvyšuje a v příštích 20 letech bude spotřeba elektřiny dále růst, pokud se situace překotně nezmění. Spotřebovávaná elektřina se používá k osvětlení, topení, chlazení a dalším aplikacím, z nichž většina je užitečná a nezbytná. Zlepšení účinnosti

predikcí spotřeby elektrické energie by mohlo pomoci zlepšit správu napájení, provoz sítě a související inspekční či opravné práce. Mimo jiné by to mohlo přinést i značné ekonomické výhody. Proto by predikce spotřeby elektrické energie měla být zásadní metodou pro rozvoj moderní energetiky. [7]

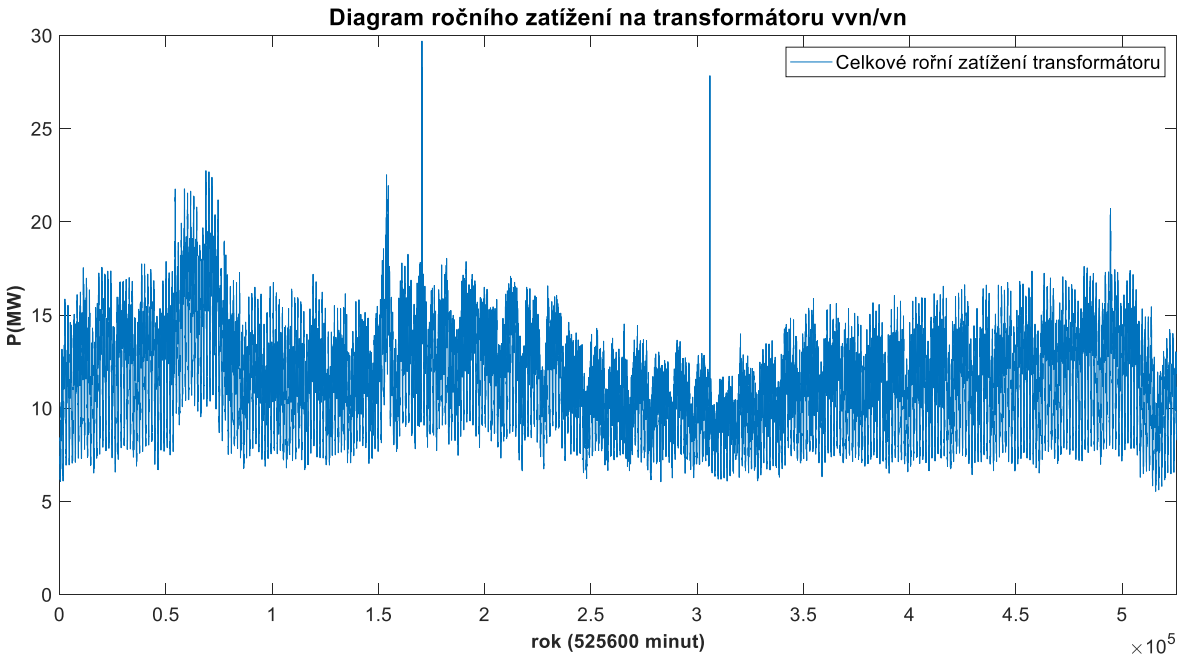
5.2 Predikce spotřeby na distribučním transformátoru

Prvním zpracovávaným objektem pro predikce je distribuční transformátor vvn/vn nacházející se v městě Plzeň. Jedná se o transformátor s převodem 110/22 kV a se zdánlivým výkonem 40 MVA. Tento transformátor napájí rezidentní zástavbu.

Zvolené prostředí pro zpracování dat bylo výpočetní programovatelné prostředí Matlab. Přesněji verze Matlab R2020a zaštítěná akademickou licencí. Pro vytváření predikčního modelu je nejprve nutné mít k dispozici naměřená data z reálného provozu. Měřil se odebíraný činný výkon a měření probíhalo po dobu jednoho roku s krokem jedné minuty, tedy každá minuta v roce má zaznamenaný aktuální odbíraný výkon. Sledovaným rokem byl celý rok 2014. Dále měření probíhalo na hlavním i záložním transformátoru, takže vstupní data jsou v maticovém tvaru 2x525600. Data ale bylo nutné upravit tak, aby vznikl pouze jeden vektor hodnot odběru. Jelikož transformátory byly zapojeny paralelně, je možné jejich výkon sčítat, a dále bylo nutné z dat pomocí jednoduchého skriptu odstranit nulové hodnoty a nahradit je jejich předchozími nenulovými hodnotami, protože s ohledem na trénování neuronové sítě nulové hodnoty v modelu nedávají smysl. Pro představu níže přikládám vzorová data před úpravou a po úpravě.



Obrázek 13 Naměřená data před úpravou



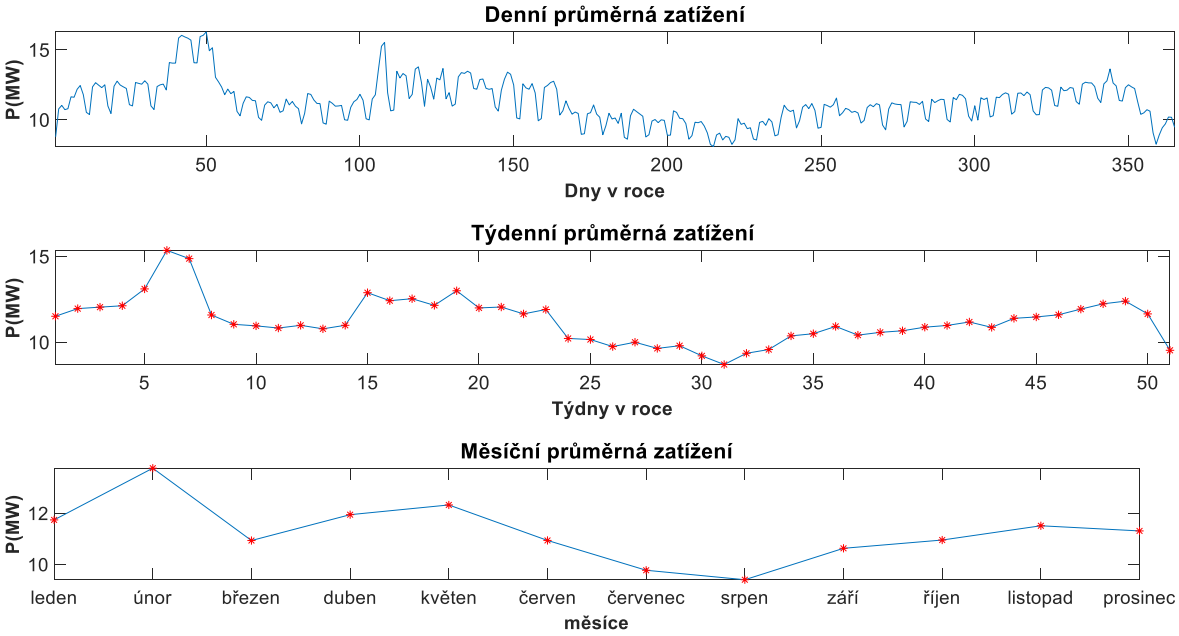
Obrázek 14 Naměřená data po úpravě

Takto upravená data viz obr. 14 již lze použít k trénování neuronové sítě a poslouží jako takzvaný target pro učení neuronové sítě, tedy jako vstupní vektor, podle kterého se síť bude učit. Nástrojem pro vytvoření neuronové sítě bude Neural Net Fitting (toolbox programu Matlab).

Z naměřených dat je patné, že během roku došlo k několika impulsům, či změnám v zatížení, které nejsou zcela přirozené. Za prvé jsou v grafu vidět dva významné impulzy na ose x v oblasti 1,6 a 3,1, které nastaly na dobu pár minut a pak zase odezněly, je otázkou zda takovýto náhlý nárůst má z hlediska provozu transformátoru nějaký význam, ale spíš je pravděpodobnější varianta, že se jedná pouze o chyby v měření, protože tento úsek v grafu zabíral v řádu vždy jen pár minut. Každopádně neuronová síť umí takováto data ignorovat, protože by predikci pouze zarušovaly, neboť se jedná o nahodilý jev. Za druhé došlo ke značnému nárůstu odebíraného výkonu v rozmezí 0,5 až 0,75 na x-ové ose, což z hlediska času odpovídá měsíci únoru. Z hlediska zatížení však není důvodu, proč by mělo k takovému nárůstu dojít, obzvlášť když tento nárůst má skokový charakter jak při jeho nástupu, tak při jeho následném poklesu na konci tohoto období. Takový jev si lze vysvětlit tím, že distributor v tuto dobu odklonil část výkonu z jiného úseku a pak jej následně zase odpojil. Nicméně takováto data jsou potom pro učení neuronové sítě chápána jako rušivá a negativně tak ovlivňují predikci. K podobným změnám v zatížení během roku docházelo ještě několikrát, nicméně už se jednalo o kratší časové intervaly.

Průběh spotřeby lze potom lépe vidět na grafech níže, kde jsou zachyceny vývoje

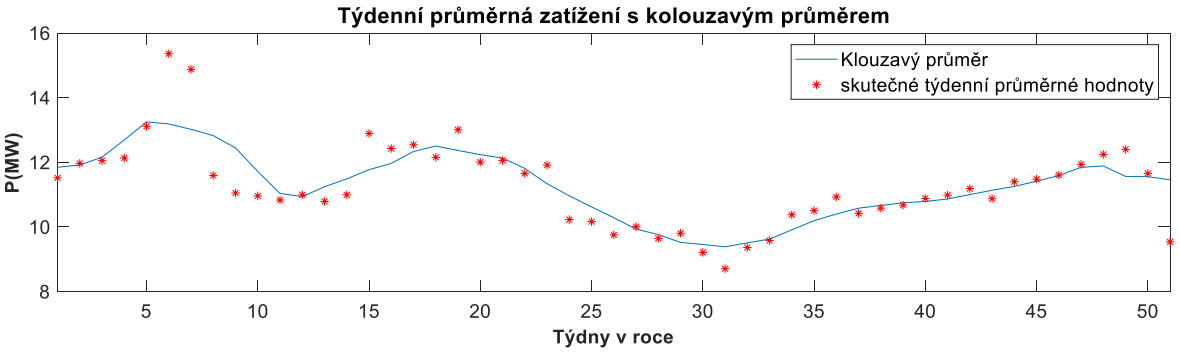
průměrných denních hodnot zatížení, průměrných týdenních hodnot zatížení a také měsíčních. Tyto grafy by měly sloužit, především pro lepší orientaci v naměřených datech.



Obrázek 15 průměrné hodnoty zatížení

Průměrné hodnoty jsou v grafech znázorněny červenými tečkami, které jsou proložené modrou křivkou. U grafu denních průměru byly červené body odstraněny, neboť by se s nimi graf stal nepřehledným.

Řešením, jak data upravit a vyhladit tak nepřirozené změny v zatížení, by potom mohla být aplikace klouzavého průměru. Z grafu níže je vidět, že takovýto nástroj by dobře vyhladil výkyv, který nastal v měsíci únor, bohužel by ale vyrušil v datech i přirozená minima a maxima ke kterým během roku dochází, jako je například pokles spotřeby koncem roku, či letní minimum.



Obrázek 16 klouzavý průměr

Nicméně toto je ukázáno jako možná cesta jak postupovat, ale jedná se o drastický zásah do naměřených dat, a proto je od toho v této práci upuštěno a data byla zpracována jen s drobnými úpravami, jak bylo na začátku kapitoly nastíněno. Také by tímto způsobem šlo upravit jen vybrané části roku, ale to by dávalo smysl, pokud by naměřená data na transformátoru byla za období dvou let a více a vytvářela by se predikce celého následujícího roku. Poskytnutá data jsou ale jen za období jednoho roku, proto v práci budou zvoleny predikční přístupy tak, aby se problematické části roku vyhnuly.

5.2.1 Vytváření vstupní matice

Dalším nutným parametrem pro vytvoření neuronové sítě je ještě input (vstupní matice). Je to matice, která určuje parametry pro trénování sítě, tedy jedné hodnotě z targetu přiřazuje několik dalších hodnot, které určují, proč je daná hodnota targetu právě takováto a umožňuje to neuronové síti hledat případné závislosti, podle kterých pak namodeluje nezávislý výstup. To, jak vypadá vstupní matice, je dáno charakterem zkoumaného problému. V našem případě se jedná o odběr z transformátoru vvn/vn, tedy parametry, které by mohly mít vliv na odběr, jsou například časové cykly, okolní teplota, vlhkost, či rychlost větru, popřípadě by se daly najít i další závislosti, které mají vliv na spotřebu elektrické energie. V případě této práce bude zvolen přístup na základě časových cyklů, který je založen na principu opakujících se jednotlivých časových úseků. Také budou v další variantě k modelu přidány i údaje o počasí, jakožto potenciálně zlepšující parametry pro predikci.

Odebírané zatížení je jakýsi vzor, který se při stálých podmínkách opakuje stále dokola. Liší se, je-li zatížení v létě či zimě, v pracovní den či o víkendu, nebo ve dne či v noci. Takto vytvořenou predikci lze považovat za stochasticky proces, protože je stanovena pouze na základě stejného okamžiku např. v minulem měsíci či roce. Vylepšujícím parametrem by potom mohly být zmíněné údaje o počasí, které by zajistily lepší rozhodování neuronové sítě o nadcházejícím vývoji spotřeby.

To, jak byla vstupní matice vytvářena, je ukázáno ve skriptu v příloze A. Matice má vytvořeno 16 řádků, které určují jednotlivé časové cykly v celém roce. První čtyři řádky reprezentují, o jaké roční období se jedná. 5. řádek je měsíc v roce, 6. – 12. řádek určuje den v týdnu, 13. řádek je hodina ve dni, 14. řádek minuta v jedné hodině, 15. řádek určuje, zda se jedná o víkend či nikoliv a 16. řádek rozhoduje o tom, zda se jedná o státní svátek. Délka dne je potom ohraničena hodnotou 1440 buněk, neboť tolik minut má jeden den. Příklad, jak vstupní matice vypadá, je zobrazen níže. Všechny parametry jsou

bez ohledu na velikost vztaženy k jedné, tedy nachází se v intervalu <0,1>. Tabulka níže tedy zatím reprezentuje pouze matici založenou na měnících se časových cyklech. Údaje o teplotě budou implementovány v práci později.

Tabulka 1 Vstupní matice (input)

Časový úsek:	Hodnota parametru										
jaro	1	1	1	1	1	1	1	1	1	1	1
léto	0	0	0	0	0	0	0	0	0	0	0
podzim	0	0	0	0	0	0	0	0	0	0	0
zima	0	0	0	0	0	0	0	0	0	0	0
měsíc	0,0833	0,0833	0,0833	0,08333	0,08333	0,08333	0,08333	0,08333	0,08333	0,08333	0,08333
pondělí	0	0	0	0	0	0	0	0	0	0	0
úterý	0	0	0	0	0	0	0	0	0	0	0
středa	1	1	1	1	1	0	0	0	0	0	0
čtvrtek	0	0	0	0	0	1	1	1	1	1	1
pátek	0	0	0	0	0	0	0	0	0	0	0
sobota	0	0	0	0	0	0	0	0	0	0	0
neděle	0	0	0	0	0	0	0	0	0	0	0
hodina ve dni	0,9583	0,9583	0,9583	0,95833	0,95833	0	0	0	0	0	0
minuta v hodině	0,9166	0,93333	0,95000	0,96667	0,98333	0	0,01667	0,03333	0,05000	0,06667	0,08333
víkend	0	0	0	0	0	0	0	0	0	0	0
státní svátek	1	1	1	1	1	0	0	0	0	0	0

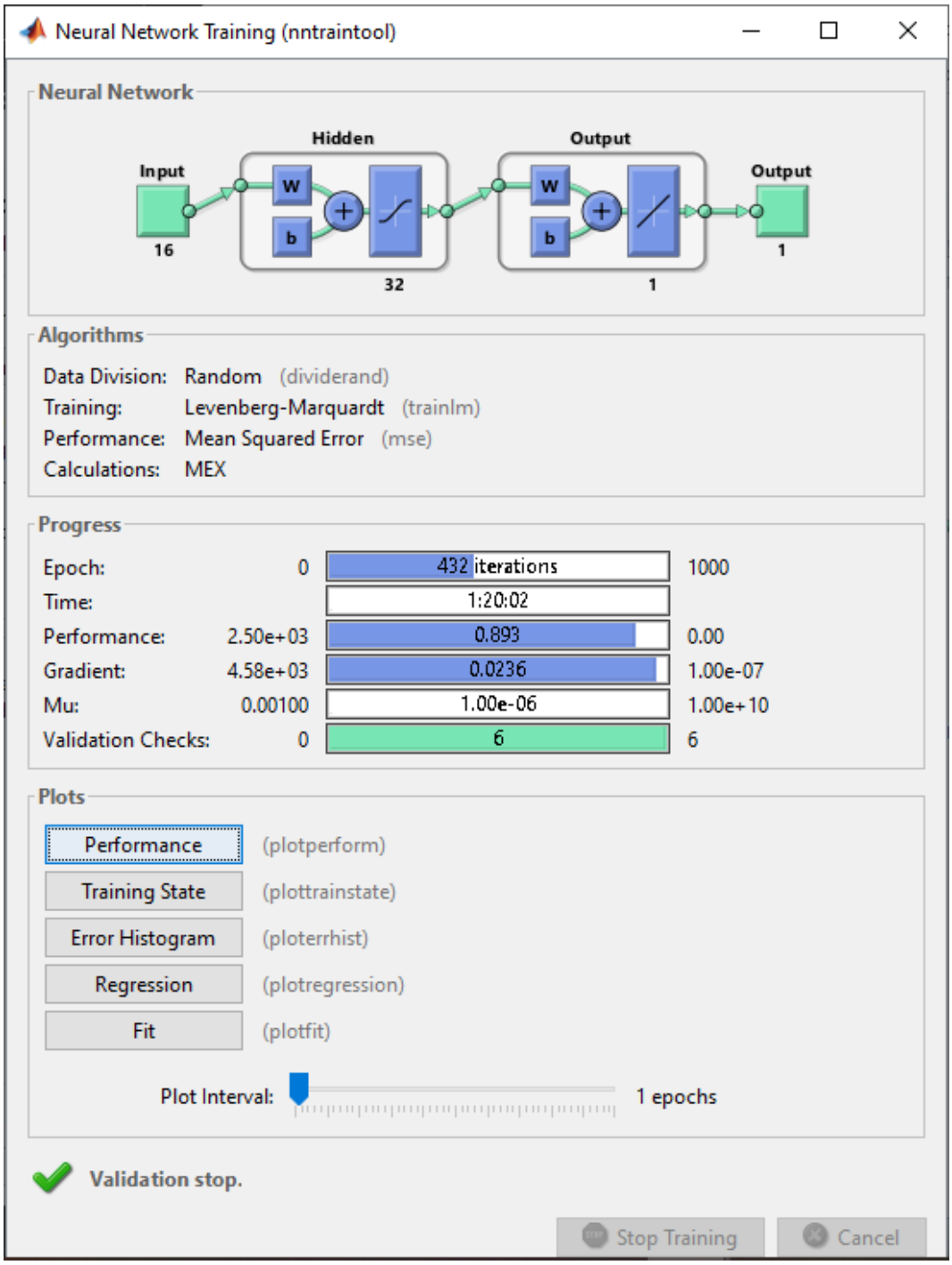
5.2.2 Trénování neuronové sítě

Po přípravě všech dat je již možné přejít ke grafickému prostředí nástroje Neural Net Fitting a natrénovat neuronovou síť. Do prostředí byly vloženy matice input (vstupní matice) a target a dále byl vybrán počet skrytých vrstev neuronové sítě. Zvolil jsem dvojnásobek velikosti vstupní matice tedy 32 skrytých neuronů, tím pádem byla síť robustní, avšak trénování probíhalo o to déle. Zvolený výpočetní algoritmus byl Levenberg-Marquardtův, jež byl popsán v předchozích kapitolách. Struktura sítě a její trénování je znázorněno na obrázku 17. Vygenerovaný skript programem Matlab k této síti je potom v příloze B.

U volby skrytých neuronů je třeba dávat pozor, aby jejich množství nebylo příliš velké, v této práci byl volen přístup maximálního počtu skrytých neuronů tak, aby nepřekročil dvojnásobek počtu parametrů vstupní matice, nicméně některá literatura uvádí, že by tento počet měl být ještě o něco menší. Problém, který by při příliš velkém množství skrytých neuronů mohl nastat, je takzvané přeučení sítě (overfitting), u kterého dochází

ke špatné generalizaci a to tím, že jsou až příliš zobecněny tréninkové vzory včetně jejich nepřesností a chyb. Tato problematika, ale už byla popsána v teoretické části práce.

První trénování sítě proběhlo na datech z celého roku měření na transformátoru a vstupní matice byla ve formátu časových cyklů znázorněných v tabulce 1.



Obrázek 17 Struktura neuronové sítě a její trénování

5.2.3 Zhodnocení prvních výsledků

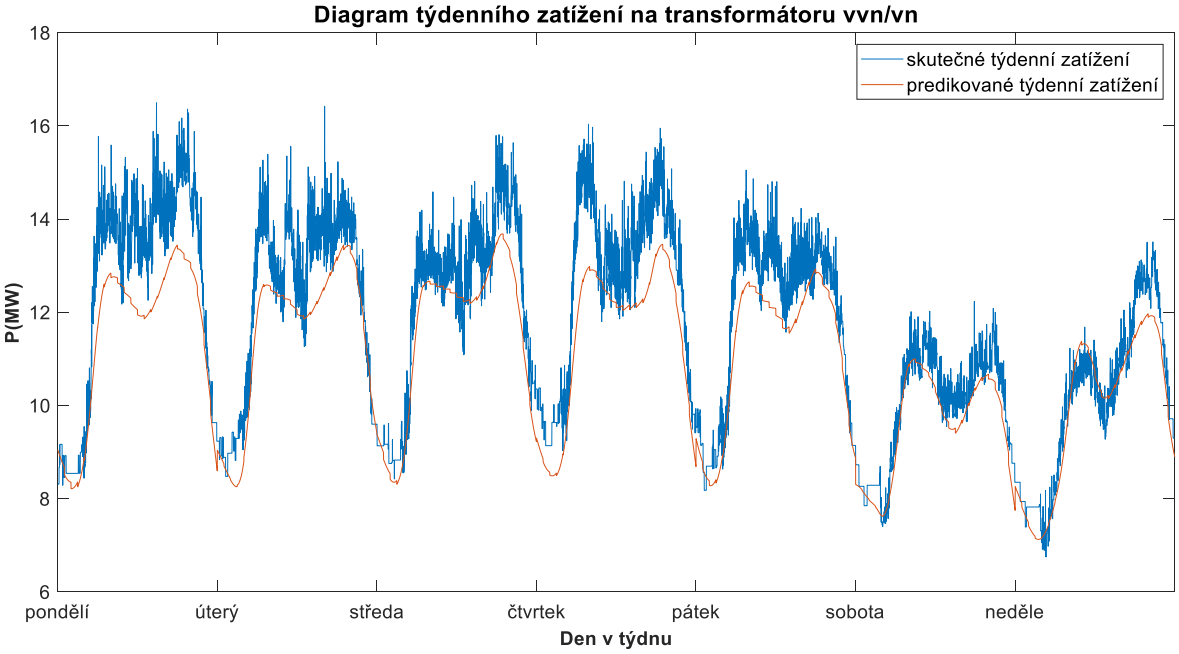
Natrénovaná síť nám vytvořila funkci, do které když dosadíme, dostaneme požadované výsledky. Obecný tvar této funkce je následující:

$$output = sim(net, input) \quad \textbf{(15)}$$

kde output reprezentuje vytvářenou predikci, kterou získáme dosazením do simulační funkce neuronové sítě, která má dva parametry a to net, což je matlabovská proměnná, v které je uložená natrénovaná síť, a input, do něhož se dosadí vstupní matice s parametry z požadovaného období.

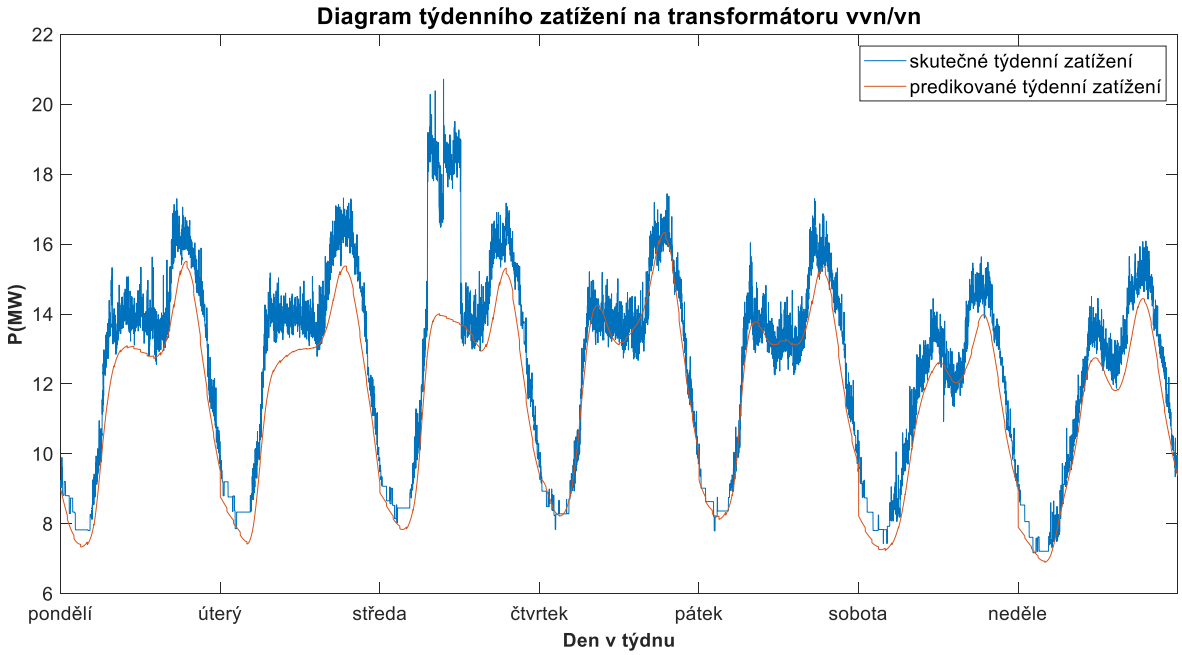
Přístup, který jsem zvolil, je takový, že po natrénování sítě na období jednoho roku hledám predikci ve tvaru, jak by vypadal například obecně jeden týden v červnu a porovnal jsem ho se skutečnými naměřenými hodnotami z tohoto období.

Pro představu na grafu níže je oranžovou křivkou predikována spotřeba pro první týden v červnu a modrou křivkou je potom doplněno skutečné zatížení v tom týdnu. Z grafu je vidět, že trend spotřeby byl zachycen poměrně přesně, nicméně v úseku od pondělí do pátku je predikce oproti skutečnosti o něco podhodnocena a na podchycení tohoto rozdílu by bylo třeba mít v modelu zakomponovány například další zpřesňující fyzikální parametry, jako již byly třeba zmiňované údaje o počasí. Víkend byl v tomto grafu zachycen poměrně přesně, k výraznější odchylce došlo potom až u nedělního maxima.



Obrázek 18 Predikce jednoho týdne v červnu

Dalším názorným příkladem potom může ještě být predikce jednoho týdnu v prosinci, průběh grafu můžete vidět na obrázku 19 a z grafu je patrné, že se neuronové síti podařilo zachytit jiný trend ve spotřebě v tomto období a špičková a pološpičková zatížení mají jiný charakter než v předchozím případě.



Obrázek 19 predikce jednoho týdne v prosinci

Zajímavostí v tomto grafu je odskok ve skutečných naměřených hodnotách, k němuž došlo v týdenním diagramu zatížení ve středu. Tuto anomálii lze vysvětlit tím, že se jedná o data z distribučního transformátoru a tedy správce tohoto transformátoru na něm dle potřeby může dělat změny a v rámci řízení elektrizační soustavy a může přes tento transformátor například odklonit nějaký výkon, který je běžně pokrýván z jiného transformátoru. Možnými příčinami potom může být řešení nějakého poruchového stavu v elektrizační soustavě či jiné manipulační operace.

Nicméně výsledek predikce neuronové sítě v bodě této anomálie je v pořádku a z hlediska predikce takovýto odskok nedává smysl. Obecnou výhodou neuronových sítí je, že dokážou takováto rušivá data úspěšně ignorovat, i když by se z nich například dále učily, ale samozřejmě jen za předpokladu, že budou mít k dispozici stále dost jiných vzoru na učení se.

Poslední poznámkou k výše získaným grafům je skutečnost, že naměřená data znázorněná modrou křivkou jsou poměrně rozkmitaná a oscilují vždy v určitém výkonovém pásu. Je to vcelku běžný jev, protože se jedná o transformátor napájející rezidentní zástavbu, jež čítá mnoho domácností a s nimi spojenými spotřebiči, které spínají

či odpínají nahodile, a tak potom vzniká rozkmit i v naměřených datech. Oproti tomu je však křivka predikce poměrně hladká, což je opět správné řešení, neboť by nebylo možné správně predikovat jednotlivé výkyvy, a kdyby tomu bylo naopak, zanášela by se do predikce zbytečná chyba.

5.2.4 Určování chyb

Výsledky zachycené v grafech mohou být zajímavé, nicméně o celkové přesnosti predikce už nedokážou vypovědět tolik, a proto je třeba vyjádřit přesnost predikce ještě i číselně. Pro určení přesnosti predikce slouží relativní chyba (odchylka), která bude dále označována jako error, a je možné ji definovat následovně:

$$error = \frac{|skutečné\ zatížení - predikované\ zatížení|}{skutečné\ zatížení} \cdot 100 \qquad \textbf{(16)}$$

Výsledkem je průměrná hodnota všech dílčích chyb spočtených výše uvedeným vzorcem a výsledná hodnota je v procentech. Při takovémto výpočtu chyby je ale dále třeba vzít v potaz to, že naměřená data a predikční křivka mají jiný charakter a tedy i pokud by predikční křivka procházela středem naměřených dat, tak kvůli jejich rozkmitu nebude relativní chyba nikdy nulová.

Pokud aplikujeme tento výpočet na data z grafů na obrázcích 18 a 19, tak pro týdenní zatížení v měsíci červnu dostaneme relativní chybu 6,95 % a jeden týden v prosinci chybu 7,19 %. U odchylky pro predikci jednoho týdne v prosinci je dále třeba brát v úvahu odskok v naměřených datech, jež nastal ve středu a celkový výpočet chyby tak zhoršuje, jinak by se dalo předpokládat, že by výsledek byl lepší.

Tabulka níže potom reprezentuje chyby pro 12 měsíčních vzorů takto namodelovaných predikcí. Pro upřesnění se tedy jedná o model, jak by vypadal typický týden v každém měsíci, který je pak srovnán se skutečnými naměřenými daty z jednoho z týdnů, který v daném měsíci nastal.

Tabulka 2 vyhodnocení chyb pro týdenní diagramy

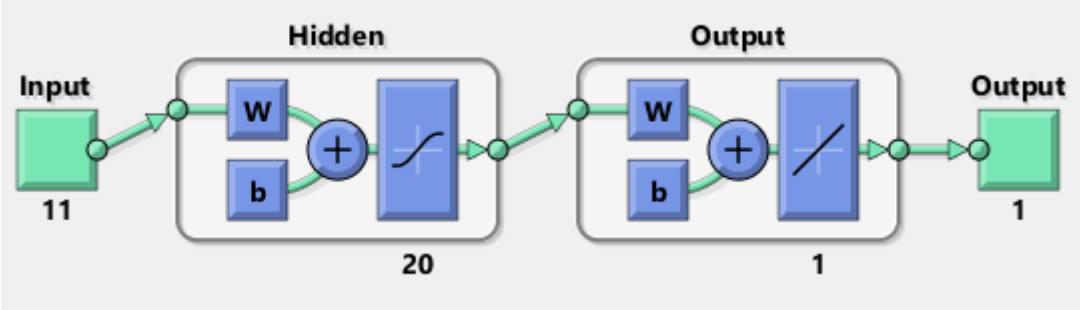
Relativní chyby predikce jednoho týdne v měsíci												
měsíc	leden	únor	březen	duben	květen	červen	červenec	srpen	září	říjen	listopad	prosinec
error (%)	8,97	10,66	3,76	8,05	5,72	6,95	4,77	4,67	4,35	3,84	3,64	7,19

5.2.5 Modelování čtvrtletních predikcí

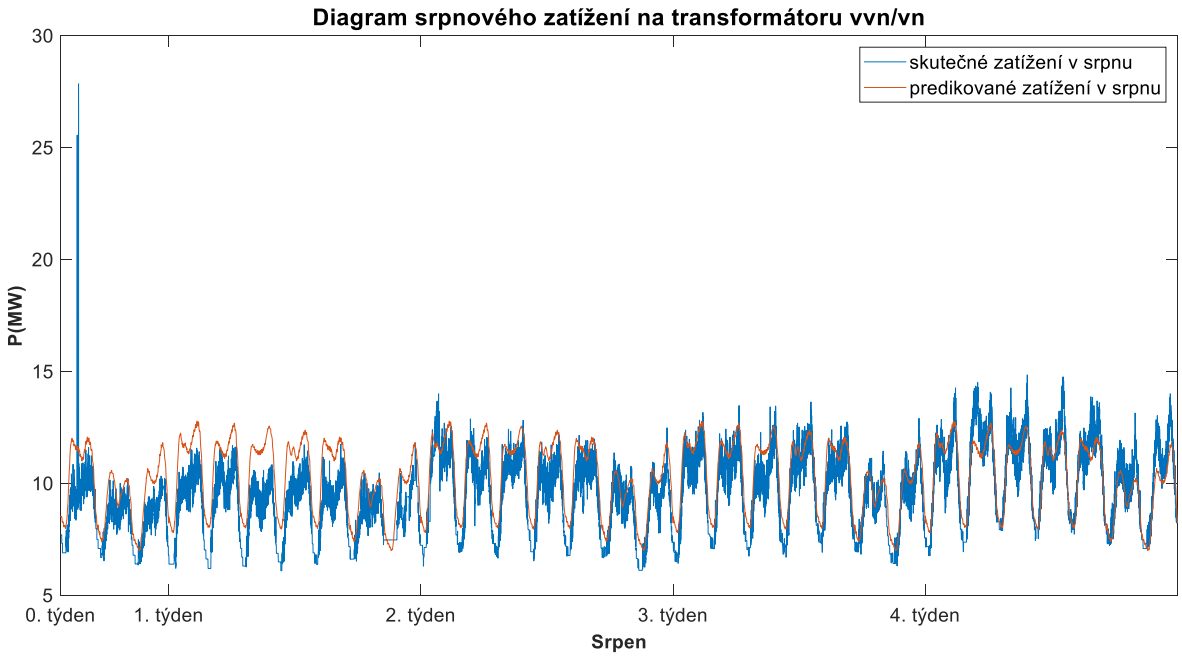
Výsledky dosažené v předchozí kapitole se mohou zdát poměrně dobré, ale je třeba mít na paměti, že se situace modelovaly na datech, na kterých se zároveň i neuronová síť učila a je tedy pravděpodobné, že bude dosaženo dobrých výsledků. Ideální přístupem by bylo po natrénování sítě na jednom roku sledovat rok další. Bohužel v tomto případě byly k dispozici jen data jednoho roku, tak nebylo možné takovýto přístup zvolit. Z tohoto důvodu vzniká i tako kapitola, ve které bude zvolen přístup takový, aby neuronová síť predikovala období, které ještě sama nezná a takovým optimálním přístupem by mohlo být sledování jednotlivých čtvrtletí.

Myšlenka je založena na tom, že by trénování neuronové sítě probíhalo pouze na dvou měsících jednoho ročního období, například léta, a následně by se udělala predikce třetího letního měsíce. Tedy učení by proběhlo na měsících červen a červenec. Následně by se zhotovila parametrická vstupní matice pro měsíc srpen, jež by se dosadila do funkce natrénované neuronové sítě a vytvořila se predikce měsíce srpna, která by se následně porovnala se skutečnými naměřenými hodnotami z tohoto období. Nevýhodou potom je, že během roku může mít každý měsíc svá specifika a tímto přístupem to není možné podchytit, což může způsobovat vyšší chybu metody. Také tréninková data jsou v tomto případě podstatně skromnější, než když probíhá trénování na celém roce. Výhodou je pak alespoň to, že trénování neuronové sítě probíhá podstatně rychleji v řádu minut a ne déle než hodinu jako v předchozím případě.

Oproti předchozímu případu byla vstupní data upravena tak, že byly vyňaty údaje o ročním období a pořadí měsíce v roce. V případě vzorové tabulky 1 tedy došlo k odstranění prvních pěti řádků, je to proto, že tyto údaje v takto postavené predikci nemají význam a v modelu by byly buď redundantní, nebo v horším případě rušivá. Na vstupu neuronové sítě tedy bylo 11 parametrů a počet skrytých neuronů byl nastaven na 20. Příklad jak takováto predikce dopadne je na obrázku 21.



Obrázek 20 struktura neuronové sítě pro čtvrtletní predikci



Obrázek 21 predikce měsíce srpen

V grafu jsou opět vidět i rušivá data, která se objevila na začátku grafu v intervalu 0. týdne. Označením 0. týdne je myšleno, že měsíce málokdy začínají pondělím a tedy se na jejich začátku objevují necelé týdny, obdobně to tak je i na konci měsíců. V tomto případě ale měsíc končí nedělí, takže je poslední týden úplný. Tedy k vyhodnocení, jak si predikce vedla po jednotlivých týdnech, jsou k dispozici čtyři týdny.

Zmíněná rušivá data mohou být chybou měření, protože je jinak otázka, co by mohlo způsobit nárůst odebíraného výkonu o více jak dvojnásobek, pouze na dobu pár minut a pak opět odeznět. Nicméně i kdyby data měla své fyzikální opodstatnění, tak z pohledu predikce opět nedávají smysl, protože se v ročních naměřených datech jedná o singularitu a tedy by takový jev byl nepredikovatelný.

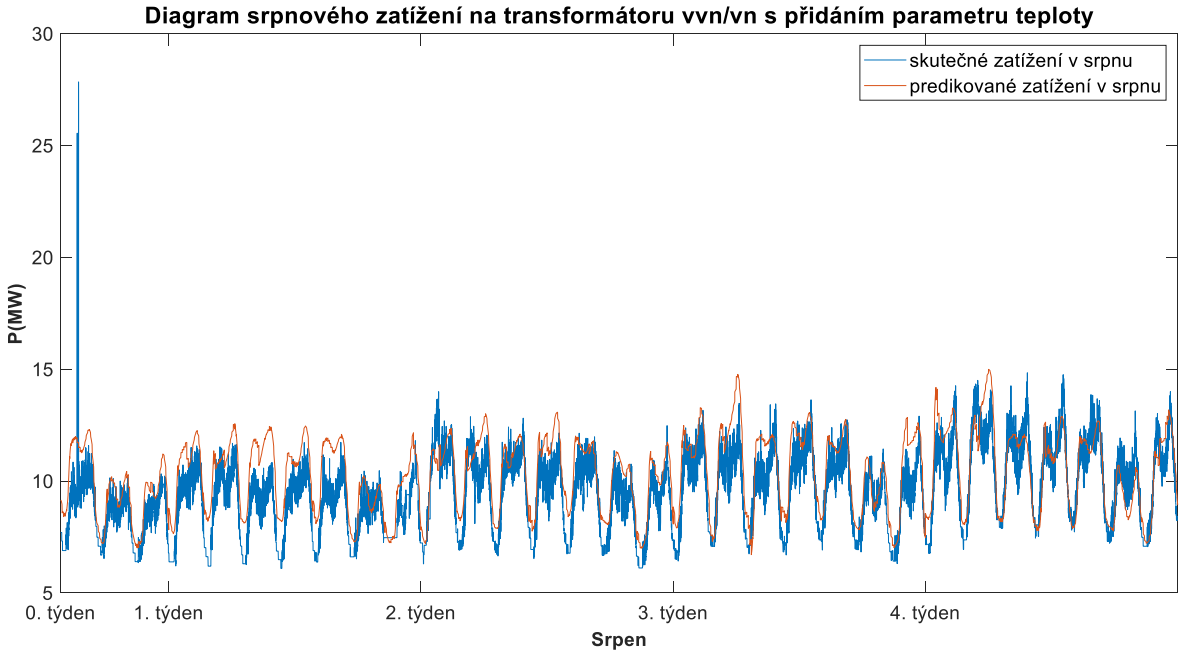
Vyhodnocení predikce je potom následující:

- Relativní chyba celého měsíce srpen: error = 10,79 %
- Relativní chyba 1. týdnu: error = 18,85 %
- Relativní chyba 2. týdnu: error = 10,65 %
- Relativní chyba 3. týdnu: error = 8,45 %
- Relativní chyba 4. týdnu: error = 5,66 %

Predikce pro 3. a 4. týden vyšla poměrně dobře a takové údaje by se pro reálné použití do provozu daly použít. Bohužel ale například predikce 1. týdne je velice špatná a chyba téměř 19 % je příliš velká a vrhá tak stín na celou zhotovenou predikci a snižuje tak její možnost být využita v reálném provozu.

5.2.5.1 Rozšíření modelu o fyzikální parametry

Možností, jak celou predikci vylepšit, může být zohlednění některých fyzikálních parametrů v predikčním modelu. Fyzikální parametry, které mohou mít vliv na vývoj spotřeby elektrické energie, by mohly být například údaje o počasí. Tedy v následující variantě se k datům vstupní matice dále přidaly údaje o teplotě. Data o počasí byla také poskytnuta vedoucím této práce a byla ve formátu hodinových naměřených úseků. S ohledem na vstupní matici pro trénování sítě bylo potřeba tyto data roztáhnout, aby odpovídala minutovému kroku původní matice. Tedy údaje se ve vstupní matici mění vždy po 60 hodnotách. Nová predikce měla na vstupu parametrů 12 a skrytých neuronů stále 20, výsledek je zobrazen níže.



Obrázek 22 predikce měsíce srpen s přidaným parametrem teploty

Z grafu na obrázku 22 je vidět, že vlivem parametru teploty došlo ke změně vymodelované predikce spotřeby, než jak tomu bylo u predikce na obrázku 21. Značně se změnilo rozložení špičkových a pološpičkových zatížení, kde ale občas docházelo až k přehnaným vrcholům. Špatně jsou například podchycena i minima v 1. týdnu, nicméně to byl problém i u předchozí predikce, tedy tento problém se nepodařilo vyřešit ani novým parametrem.

Vyhodnocení predikce s přidaným parametrem teploty (a srovnání bez teploty):

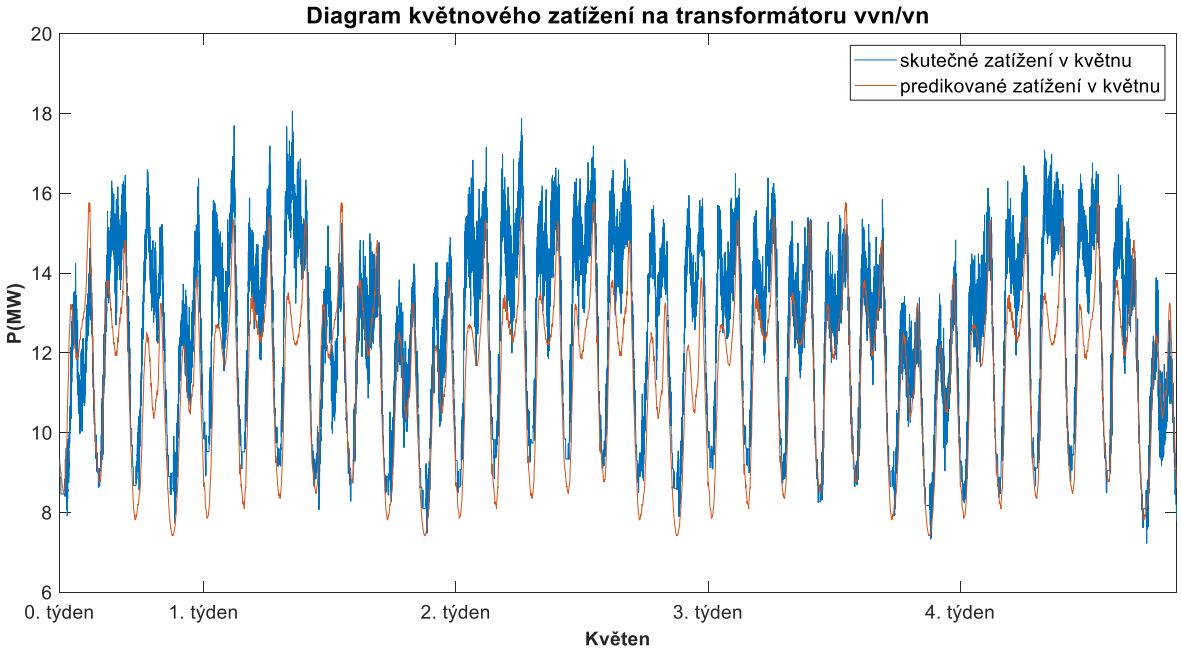
- Relativní chyba celého měsíce srpen: **error = 11,70 %** (10,79 %)
- Relativní chyba 1. týdnu: **error = 17,04 %** (18,85 %)
- Relativní chyba 2. týdnu: **error = 11,14 %** (10,65 %)
- Relativní chyba 3. týdnu: **error = 11,58 %** (8,45 %)
- Relativní chyba 4. týdnu: **error = 7,62 %** (5,66 %)

Z výsledků je tedy patrné, že se jedná o kategoricky horší predikci. Při snaze o vylepšení predikce byly do modelu postupně přidávány i parametry jako vlhkost, rychlost větru a jeho směr, oblačnost, či sluneční svit. Žádný z těchto dalších parametrů však predikci nezlepšoval ani v jejich vzájemných variacích, spíše docházelo ke zhoršování. Další snahou o lepší výsledek byla ještě úprava dat do jiného tvaru. Tedy údaje o teplotě byly rozděleny do tří nových parametrů, a to maximální a minimální denní teplota a průměrná denní teplota. Jednalo se o parametry, které byly unikátní pro každý den a měnily se vždy po 1440 hodnotách, které odpovídají počtu minut ve dni. Nicméně ani tento přístup nedokázal výsledek zlepšit. Otázkou tedy zůstává, proč tomu tak je, když přídavné fyzikální parametry by měly predikci spotřeby spíše vylepšovat. Odpovědí by mohl být fakt, že trénování dat na dvou měsících je pro implementaci přídavných fyzikálních parametrů příliš krátkou dobou a neuronová síť tak nemá k dispozici dostatek vzorů, aby dokázala podchytit případné závislosti. Mohlo by dojít k pochybám, jestli údaje o počasí mají vliv na spotřebu elektrické energie, ale fakt, že přídavné fyzikální parametry mohou mít pozitivní vliv na predikci spotřeby, bude potvrzen v dalších částech této práce.

5.2.5.2 Další výsledky čtvrtletních predikcí

S ohledem na výsledky a diskusi provedenou výše, nedává smysl, pro zbylá čtvrtletí vytvářet modely s přidanými fyzikálními parametry, a tedy postačí pouze model založený na časových cyklech. Také bude vypuštěno čtvrtletí pro zimní měsíce, a to z důvodu, že konec roku má dosti specifickou spotřebu a měsíc únor měl spotřebu zase abnormálně vysokou, jak bylo řešeno na začátku kapitoly 5.2 u naměřených dat.

Jaro:

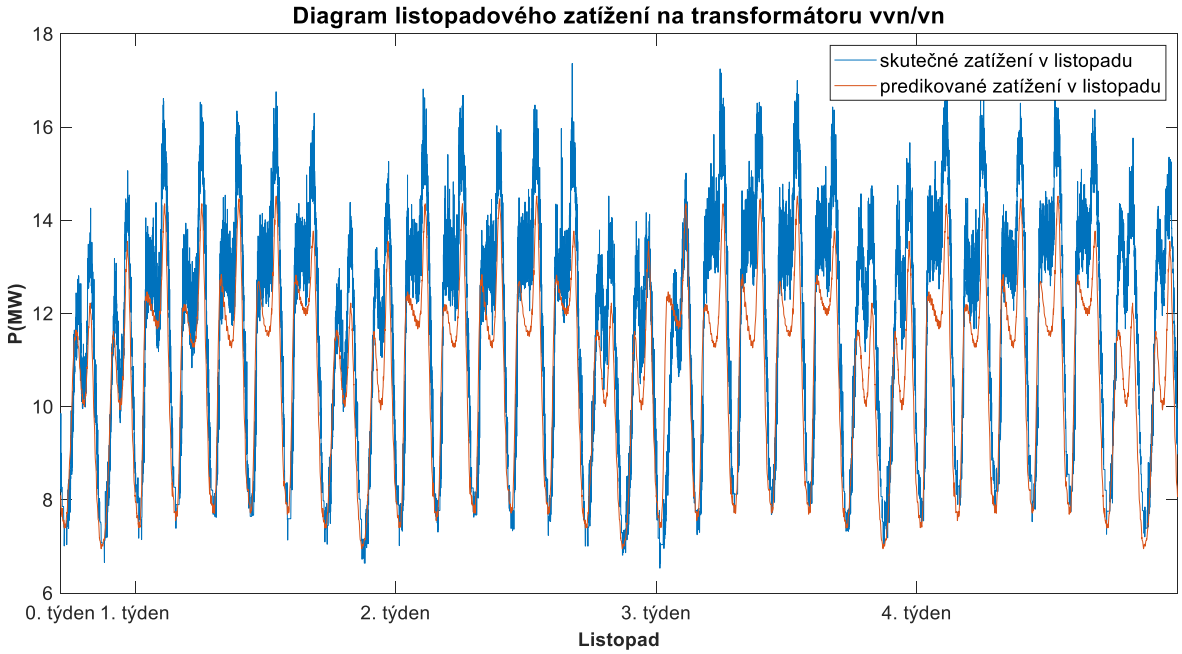


Obrázek 23 predikce měsíce květen

Vyhodnocení predikce:

- Relativní chyba celého měsíce května: error = 9,28 %
- Relativní chyba 1. týdnu: error = 8,62 %
- Relativní chyba 2. týdnu: error = 12,13 %
- Relativní chyba 3. týdnu: error = 6,26 %

Podzim:



Obrázek 24 predikce měsíce listopad

Vyhodnocení predikce:

- Relativní chyba celého měsíce listopad: error = 9,82 %
- Relativní chyba 1. týdnu: error = 8,34 %
- Relativní chyba 2. týdnu: error = 8,92 %
- Relativní chyba 3. týdnu: error = 11,02 %
- Relativní chyba 4. týdnu: error = 11,51 %

5.3 Predikce spotřeby velkoodběratele

Druhým zpracovávaným objektem pro predikci je velkoodběratel připojený na hladině vn/nn. Jedná se o zdravotní zařízení, které s ohledem na citlivost dat nemůže být blíže specifikováno. Převody transformátů připojených do tohoto objektu jsou 22/0,4 kV a celkový transformační výkon oblasti je 3520 kVA.

Poskytnutá data v tomto případě nereprezentovala aktuální výkon odebíraný v jednotlivých minutách za rok, ale byla ve formátu takzvaných čtvrthodinových maxim. Čtvrthodinové maximum reprezentuje celkovou odebíranou energii za časový interval 15 minut a hodnota výkonu s ní související potom odpovídá průměrné hodnotě výkonu, která během sledovaných 15 minut nastala.

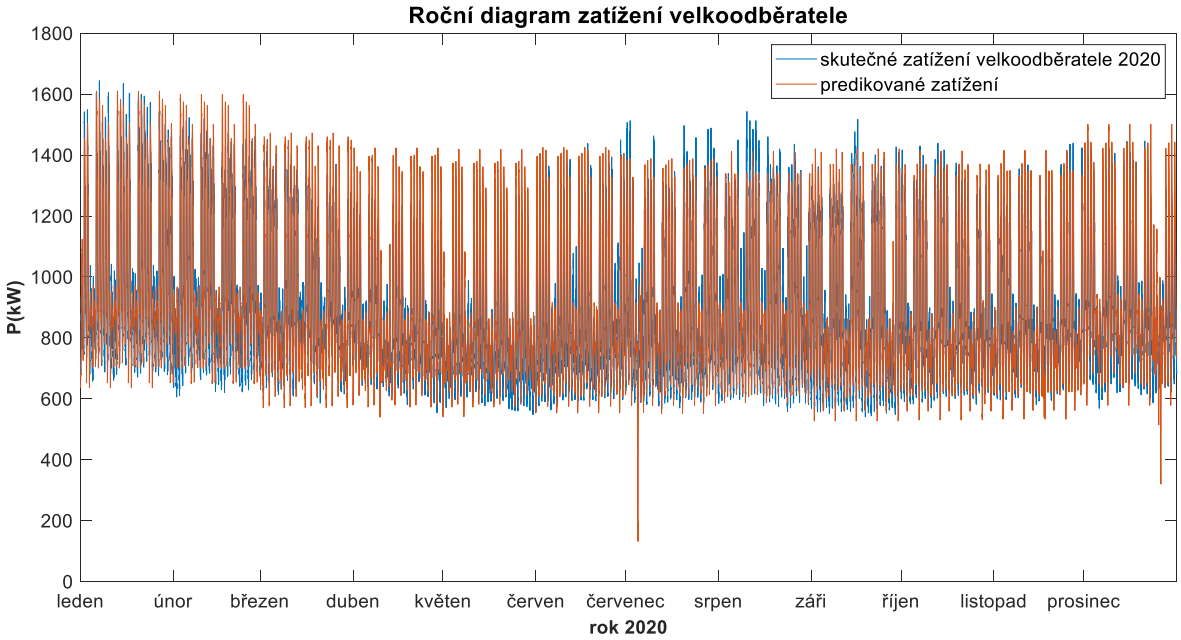
Zpracovávaná data jsou ve formátu hodnoty průměrného výkonu v patnáctiminutových úsecích v kW. Oproti hodnotám na distribučním transformátoru jsou zde k dispozici naměřená data za tři roky a to roky 2018, 2019 a 2020. Díky čtvrthodinovým úsekům je potom i celkový objem dat menší, tedy je vždy 35040 naměřených hodnot za rok. Díky charakteru dat je nebylo nutné nijak upravovat, jak tomu bylo například u distribučního transformátoru, a data bylo možné rovnou zpracovávat.

Vstupní matice časových parametrů zůstala ve svém charakteru stejná, jen byly přizpůsobeny rozsahy tak, aby odpovídaly čtvrthodinovým maximům. Tedy jeden den již nečítá 1440 prvků (minuty ve dni) ale pouze 96 (čtvrthodiny ve dni). Díky menším rozsahům dat potom i trénování neuronových sítí probíhalo podstatně rychleji. Také bylo třeba časové parametry přeorganizovat tak, aby odpovídaly jiným rokům, pro které byla predikce připravována. První přístup byl opět volen jen na základě časových parametrů a následně se k datům přidaly parametry o počasí jako teplota, vlhkost, rychlost větru a jeho směr, jež byly naměřené v dané lokalitě.

5.3.1 Výsledky dvouletého trénování

Díky většímu množství naměřených dat, přesněji tří rokům měření, bylo možné vytvořit takový model, který by se mohl učit na dvou letech dat a následně vytvořit predikci roku třetího. První výsledek je pouze pro časové parametry, kde se vstupní matice skládá z 16 parametrů, skrytých neuronů bylo zvoleno 25 a trénování sítě trvalo pouze něco přes deset minut, což je oproti učení na distribučním transformátoru velice rychlé.

Časové parametry:

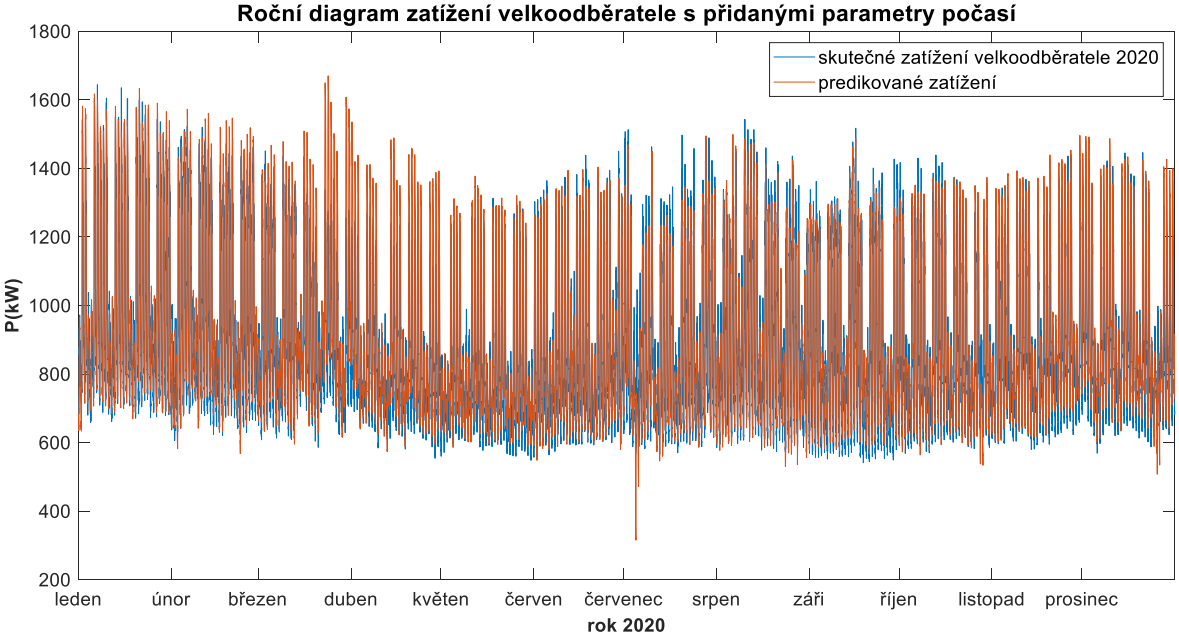


Obrázek 25 Predikce roku 2020 pro velkoodběratele

Vyhodnocení predikce:

- Relativní chyba celého roku 2020: error = 8,84 %
- Relativní chyba v lednu: error = 6,26 %
- Relativní chyba v únoru: error = 9,98 %
- Relativní chyba v březnu: error = 9,07 %
- Relativní chyba v dubnu: error = 12,63 %
- Relativní chyba v květnu: error = 11,76 %
- Relativní chyba v červnu: error = 8,58 %
- Relativní chyba v červenci: error = 7,59 %
- Relativní chyba v srpnu: error = 7,19 %
- Relativní chyba v září: error = 8,07 %
- Relativní chyba v říjnu: error = 7,17 %
- Relativní chyba v listopadu: error = 8,00 %
- Relativní chyba v prosinci: error = 11,33 %

Časové parametry plus parametry o počasí:



Obrázek 26 Predikce roku 2020 pro velkoodběratele s přidanými parametry počasí

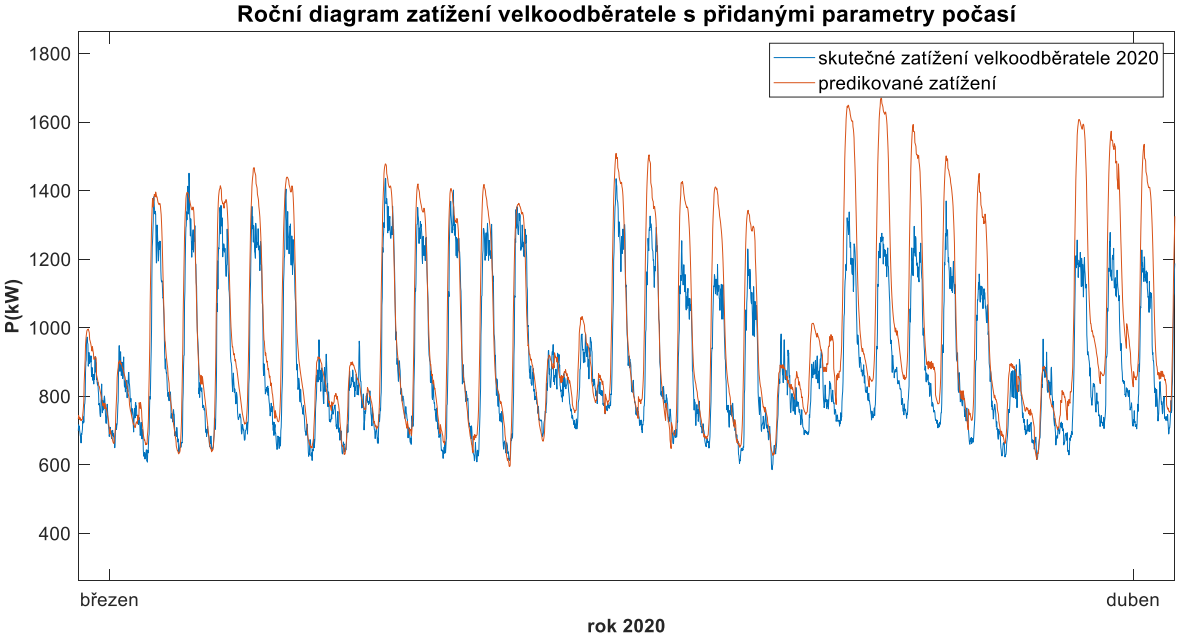
Vyhodnocení predikce (a výsledky predikce bez počasí):

- Relativní chyba celého roku 2020: **error = 7,39 %** (8,84 %)
- Relativní chyba v lednu: **error = 5,51 %** (6,26 %)
- Relativní chyba v únoru: **error = 6,88 %** (9,98 %)
- Relativní chyba v březnu: **error = 11,84 %** (9,07 %)
- Relativní chyba v dubnu: **error = 12,32 %** (12,63 %)
- Relativní chyba v květnu: **error = 7,51 %** (11,76 %)
- Relativní chyba v červnu: **error = 4,87 %** (8,58 %)
- Relativní chyba v červenci: **error = 5,05 %** (7,59 %)
- Relativní chyba v srpnu: **error = 5,22 %** (7,19 %)
- Relativní chyba v září: **error = 5,41 %** (8,07 %)
- Relativní chyba v říjnu: **error = 6,09 %** (7,17 %)
- Relativní chyba v listopadu: **error = 9,43 %** (8,00 %)
- Relativní chyba v prosinci: **error = 9,67 %** (11,33 %)

Z výsledků je patrné, že v tomto případě po implementaci údajů o počasí došlo ke zlepšení výsledku. Na ročním vyhodnocení chyby to činí rozdíl 1,45 % ve prospěch neuronové sítě se zahrnutými údaji o počasí. Prokázalo se také při vytváření modelů, že každý ze čtyř zmiňovaných údajů o počasí měl příznivý vliv na celkový vliv predikce.

Pravděpodobným důvodem, proč v tomto případě měly přídavné fyzikální parametry pozitivní vliv na predikci, bude zřejmě to, že oproti tréninku na čtvrtletních predikcích, jak tomu bylo u distribučního transformátoru, zde bylo k dispozici dostatek tréninkových vzorů, aby neuronová síť mohla podchytit případné závislosti. Také se dá předpokládat, že vliv počasí u menšího objektu bude mít větší vliv na spotřebu elektrické energie než je tomu třeba u distribučního transformátoru, který pokrývá spotřebu různorodou. Menší objekt tedy bude na základě počasí více reagovat se spínáním topení v případě, že je chladno, nebo spínáním klimatizací v případě teplého počasí.

Grafy výše nejsou až tolik přehledné, proto bude dobré dát náhled na to, jak predikce vypadala v určitém detailu. Vhodným okamžikem, na který se zaměřit, by mohl být okamžik, kdy se predikce rázem zhoršila. Z výsledků výše je vidět, že predikce v měsících leden a únor byla velice dobrá a v březnu došlo ke značnému zhoršení. Následující graf patří k predikci se zahrnutými parametry o počasí.



Obrázek 27 Detail ročního diagramu na měsíc březen

Z detailu grafu ročního zatížení je vidět, že predikce probíhala velice dobře až do půlky března. Od půlky března potom následuje velká odchylka naměřených dat a predikce. S ohledem na zhotovený prediktivní model by předpokládaný vývoj spotřeby měl být správný, avšak naměřená data tomu neodpovídají. Vysvětlení by ale mohlo být snadné, protože rok 2020 se vyznačuje začátkem celosvětové pandemie, která zasáhla téměř do všech odvětví lidské činnosti, tak se to projevilo i na charakteru spotřeby elektrické energie v různých oblastech. S ohledem na typ zařízení, které tento

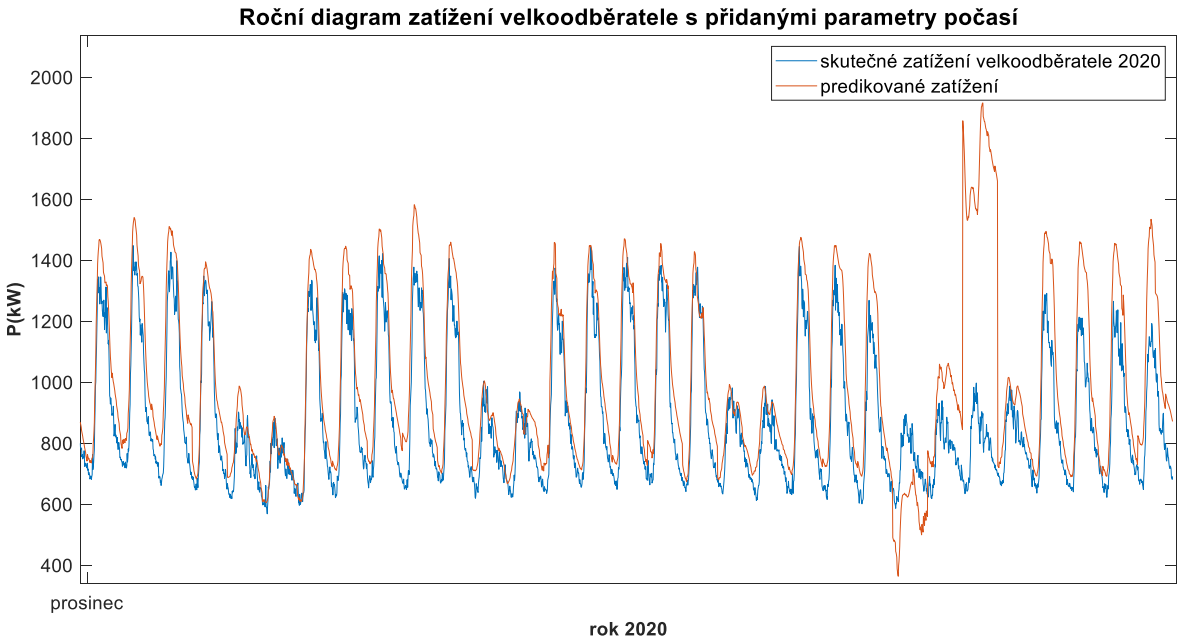
velkoodběratel představuje, je logické, že s celostátním lockdownem přišel i razantní pokles spotřeby alespoň na nějakou dobu. Z výsledů dalších měsíců už je potom patrné, že došlo k určité stabilizaci. Dá se tedy předpokládat, že pokud by v roce 2020 nenastala celosvětová pandemie, výsledek celoroční predikce by mohl být o něco lepší. Nicméně tento fenomén alespoň dává možnost hledat nová řešení, jak přistupovat k predikci založená na pravidelné aktualizaci naměřených dat, která budou ukázána v kapitole této práce týkající se systému pro online monitoring.

5.3.2 Výsledky jednoletého trénování

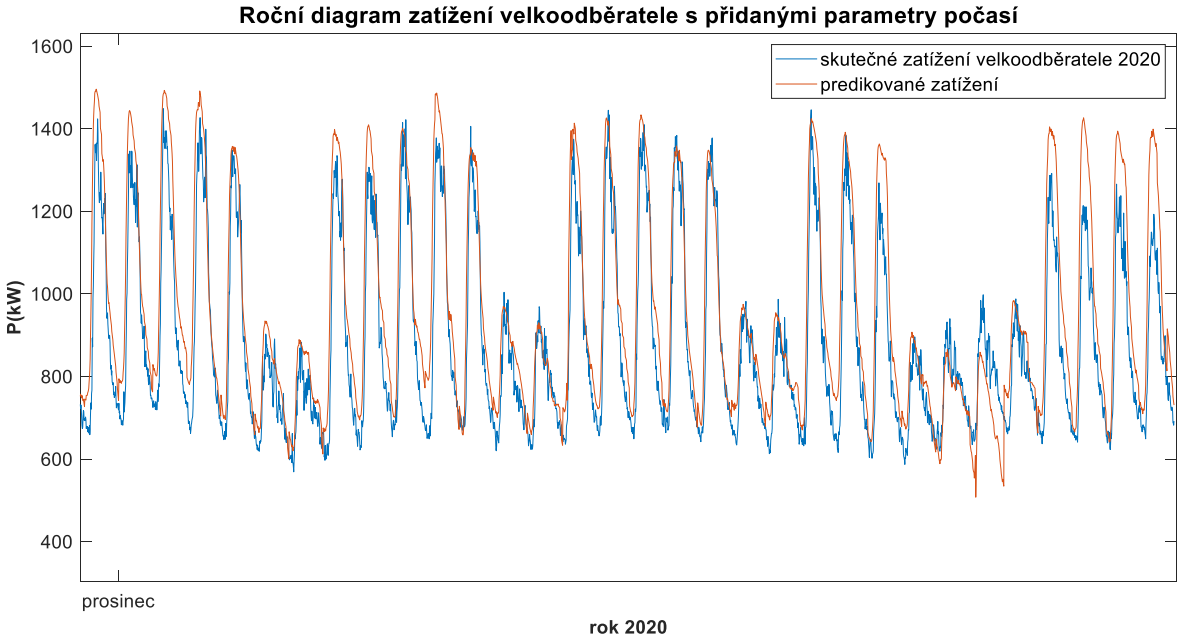
V rámci této kapitoly by ještě bylo vhodné ukázat, jaký smysl v predikci má velikost naměřených dat. Základním předpokladem je, že čím více naměřených dat máte, tím lepší prediktivní model v umělé neuronové síti dokážete zhotovit. Pokud chcete zhotovit celoroční predikci, je třeba mít na paměti, že během roku nastávají okamžiky, které jsou v daném roce specifické, jsou to například období jako Vánoce a Velikonoce. Aby bylo možné vytvořit predikci Vánoc, je zapotřebí mít údaje alespoň z jednoho předchozího roku, protože jinak vývoj spotřeby o Vánocích nelze odpredikovat, protože se jedná o ojedinělé období v roce. Dále je ale třeba mít na paměti, že svátky se v rámci jednotlivých let odehrávají v jiných dnech, takže pokud jsou k dispozici data, kde je například Štědrý den v pátek a další rok je v sobotu, může to být pro umělou neuronovou síť matoucí a predikce Štědrého dne by dopadla špatně. Tedy takovým optimálním množstvím naměřených let by mohlo být sedm a až potom by umělá neuronová síť měla dostatek vzorů pro určení přesné predikce i specifických dnů v roce. Tolik naměřených dat je ale málokdy k dispozici, proto se modely neuronových sítí snažíme dělat tak, abychom pokud možno i s méně daty dokázali poměrně přesnou predikci vytvořit.

Na tomto případě by tedy mělo být ukázáno, že menší množství vstupních dat negativně ovlivní výsledek predikce. Zvolen je model se zakomponovanými parametry o počasí, jakožto lepší model z předchozích dvou.

Relativní chyba jednoroční predikce, která činí 7,67 %, není o tolik větší jako chyba dvouleté predikce, která byla 7,39 %. Níže je však vidět detail v roční predikci na měsíc prosinec (obrázek 28), kde výsledek predikce vánočních svátků (v třetí čtvrtině grafu) je nanejvýš neuspokojivý. Pro srovnání je ještě na obrázku 29 detail stejného období, ale z tréninku provedeného na dvou letech. To k opodstatnění, proč má význam disponovat větším počtem vstupních dat pro trénování neuronových sítí.



Obrázek 28 Detail ročního diagramu na měsíci prosinec (1 rok tréninku)



Obrázek 29 Detail ročního diagramu na měsíci prosinec (2 roky tréninku)

6 Aplikace rozhodovacích stromů na predikci spotřeby

Dalším zvolenou metodu pro vytváření predikcí se staly rozhodovací stromy (decision trees). Oproti neuronovým sítím se jedná o podstatně omezenější a jednoduší metodu, neboť se v ní nedá vytvořit koncept univerzálních parametrů, na základě kterých by potom probíhala adaptace výsledků na různé podněty. Jedná se o metodu, která se pouze učí a reaguje na naměřená data. Je tedy jasné, že v tomto případě už nebude možné vytvářet tak komplexní modely, které by dokázaly odpredikovat celé roky, ale bude třeba hledat přístupy, pro které by tento model mohl dávat dobrý smysl.

Hlavním cílem v této práci bude zaměření se na vytváření krátkodobých predikcí za využití regresních algoritmů AdaBoost a Gradient Boosted Decision Trees. Zvoleným nástrojem pro vytváření prediktivních modelů se stalo vývojové prostředí pro strojové učení s názvem Scikit-learn. Scikit-learn je navržen pod programovacím jazykem Python a prostředí, ve kterém byly modely vytvářeny, bylo interaktivní prostředí Jupyter Notebook za využití programového rozcestníku Anaconda Navigator.

Pro obě zvolené metody Adaboost i Gradient Boosted Decision Trees byl zvolen přístup vytváření krátkodobých predikcí. Tedy hlavní myšlenka byla založená na tom, že změny v zatížení během roku jsou pomalu se měnící a tedy rozdíl mezi za sebou jdoucími týdny nebude příliš velký. Volit kratší časové úseky by opět nedávalo smysl, protože není možné predikovat například zatížení v úterý ze zatížení v pondělí, protože každý den v týdnu má s ohledem na zatížení svá specifika. Modelovat celé měsíce by byl zase příliš dlouhý úsek, protože v horizontu měsíců k viditelným změnám v zatížení již dochází. Proto by vhodným časovým úsekem pro vytváření krátkodobých predikcí mohly být právě časové úseky v rámci týdnů.

6.1 Predikce spotřeby na distribučním transformátoru

Prvním zkoumaným objektem opět bude distribuční transformátor, u kterého jsme si informace o datech přiblížili v kapitole 5.2. Úprava dat tedy zůstala obdobná, jen následně došlo k vyexportování dat z Matlabu a jejich načtení v prostředí Jupyter Notebook. U zvolených metod bylo využito nabídky vzorových skriptů nástroje Scikit-learn, které byly následně upravovány pro potřeby této práce.

6.1.1 Predikce na transformátoru pomocí Adaboost

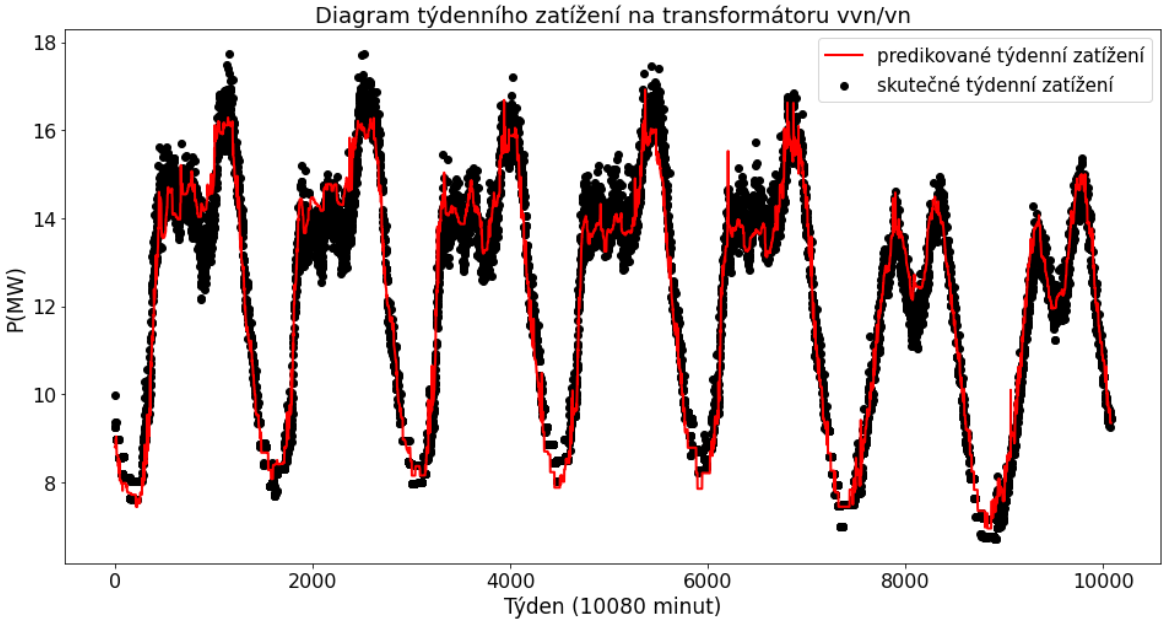
Po nahrání dat v prostředí Jupyter Notebook bylo možné už data rovnou zpracovávat, protože jejich předpříprava už proběhla v Matlabu. Pro vytváření modelu pro rozhodovací

stromy s regresním algoritmem AdaBoost bylo třeba si nadefinovat dvě hlavní proměnné. Jednu, která je naplněna naměřenými hodnotami výkonů, a druhou, jež by představovala čas, ve kterém byl výkon měřen. Tyto dvě proměnné je pak už možné dle potřeby dále upravovat.

Vstupními daty pro vytvoření predikce se potom stal úsek dvou týdnů, jež představovaly tréninková data, a následně se vytvořila predikce týdne následujícího. Hodnoty predikce a skutečné hodnoty byly vzájemně porovnány a spočetla se relativní chyba. Programovým cyklem bylo zajištěno, aby došlo k posunu dat a vytvořila se tak zase n+1 predikce následujícího týdne. Cyklus tedy běžel ve 49 smyčkách, protože rok 2014 měl 51 celých týdnů a první dva týdny sloužily jako základ pro vytvoření první predikce.

U samotné metody bylo ještě třeba nastavit dva vstupní parametry. Prvním je n_estimators, který určuje šíři rozhodovacího stromu. S ohledem na množství vstupních dat nebylo třeba tento parametr nastavovat příliš vysoký a byla zachována implicitní hodnota n_estimators = 300. Druhým parametrem této metody je max_depth, který představuje hloubku prováděné regrese. Tento parametr je třeba ladit tak, aby nedošlo k příliš složité struktuře rozhodovacího stromu a tak nedošlo k přeučení (ovefitting). V opačném případě by zase nebyly podchyceny základní mechanismy ve sledovaném ději. Názorný příklad byl v teoretické kapitole 3 na obrázku 10.

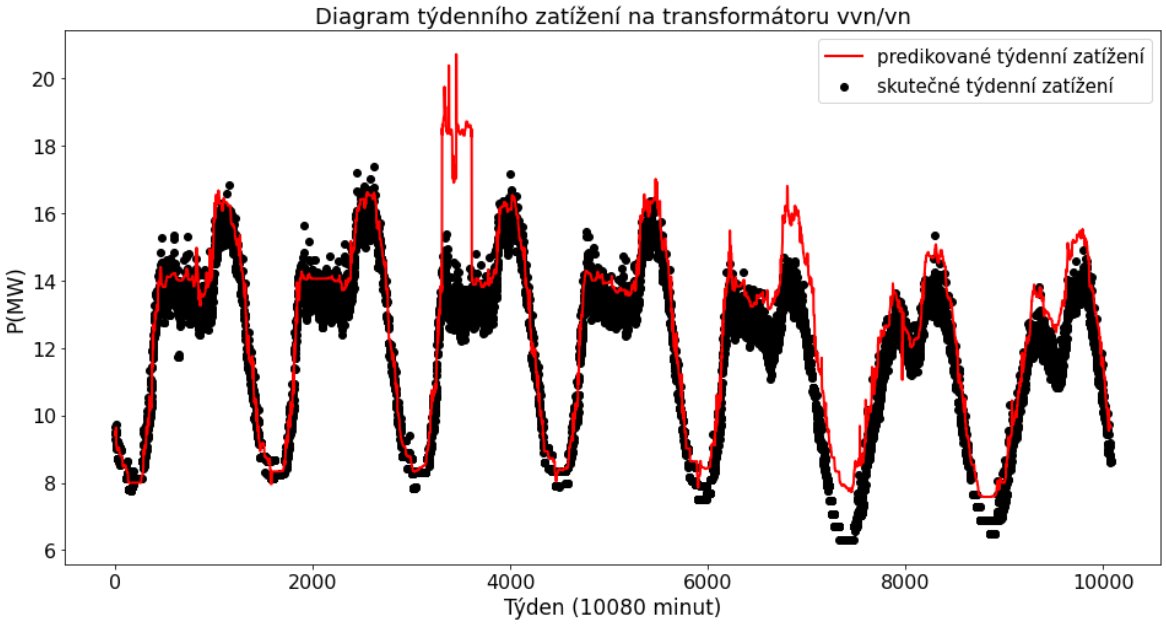
Ukázky, jak vypadaly skripty pro načtení dat a samotnou AdaBoost metodu, jsou v přílohách C a D. Níže jsou ukázány příklady výsledků získaných touto metodou.



Obrázek 30 predikce týdenního zatížení pomocí AdaBoost

Na obrázku 30 je znázorněna predikce jednoho týdne ze začátku roku přesněji se jedná o čtvrtý týden v roce. Relativní chyba byla vyhodnocována obdobně jako u neuronových sítí a v tomto případě bylo dosaženo relativní chyby 3,50 %. Takto by se mohlo jednat o velice přesnou predikci, protože v tomto případě docházelo právě k pomalým změnám mezi jednotlivými týdny. Nicméně jak již bylo diskutováno v kapitole 5.1, tak naměřená data pro distribuční transformátor byla poněkud skokově proměnlivá a například s vysokým nárůstem odebíraného výkonu v únoru se rázem relativní chyba této metody dostávala až nad 20 % a tedy se takováto predikce stávala bezcennou.

Metody rozhodovacích stromů jsou též velice citlivé na rušivá data, která se pak promítají do predikce, ačkoliv fyzikálně nedávají smysl. Takový příklad potom může být na obrázku 31.



Obrázek 31 predikce týdenního zatížení pomocí AdaBoost s rušivými daty

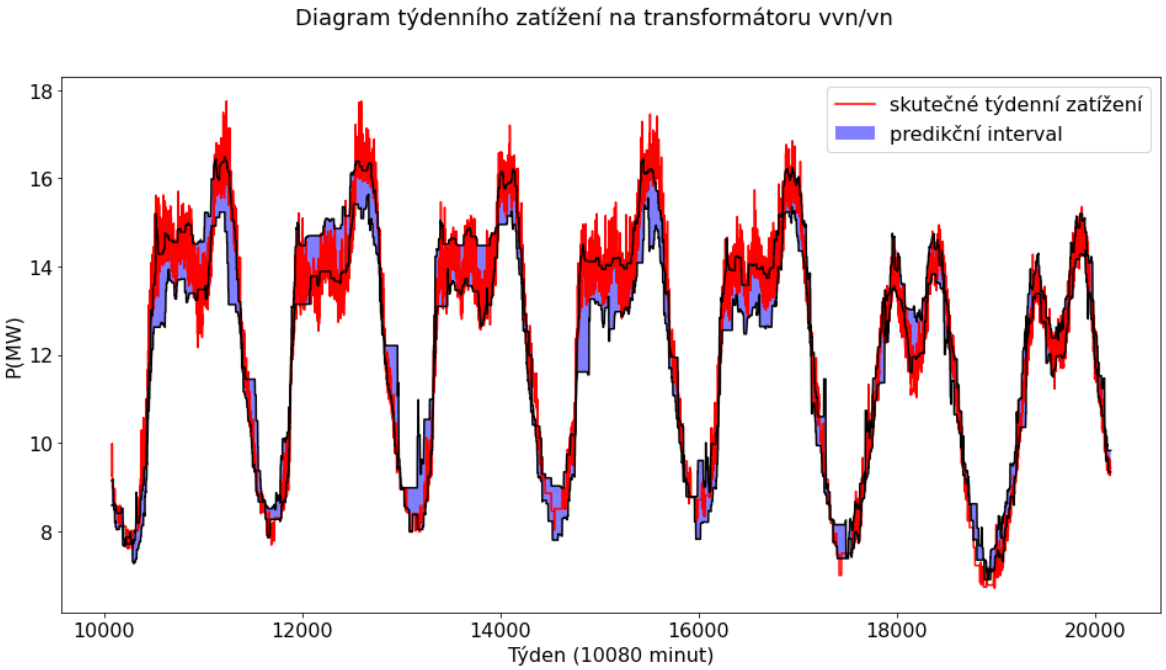
Jedná se o podobný případ, který nastal v měsíci prosinec, jak tomu bylo u týdenních predikcí pro neuronové sítě, jen je celý děj posunut o týden dopředu. Tedy rozhodovací stromy se učily na datech, v kterých byla i rušivá výchylka, a následně vytvořily predikci. Výsledek je přesně opačný, než jak tomu bylo u neuronových sítí, kdy odhad predikce neuronové sítě dokáže takovéto výchylky v datech ignorovat, ale u rozhodovacích stromů došlo vlivem rušivých dat k predikci, jež nedává smysl.

V sumarizaci se jedná o metodu, která může dosahovat dobrých výsledků, ale bohužel není aplikovatelná na data celého roku a je také velice citlivá na rušivá data. I tak jsou ale období roku, kdy by bylo možné tuto predikci nasadit, protože zase její jednoznačnou

výhodou je poměrná jednoduchost a snadná příprava vstupních dat. Dále také menší náročnost na výpočetní techniku a její rychlost.

6.1.2 Predikce na transformátoru pomocí Gradient Boosted Decision Trees

Následují metoda Gradient Boosted Decision Trees je v principu velice podobná předchozí. Proto i přístupy k predikci byly zvoleny stejné. Taktéž tedy i výsledky jsou, co do své kvality, obdobné a rovněž i problémy zde nastávají stejné. V čem se ale tato metoda odlišuje, je tvar získaného výsledku. Nejedná se o vytváření predikční křivky, ale o predikční interval, který by dával informaci o tom, v jakém rozmezí očekávat aktuální spotřebu elektrické energie. Metoda byla oproti AdaBoost náročnější na ladění, protože obsahovala více vnitřních parametrů, které ovlivňovaly tvar výsledku. Příklad upraveného skriptu z knihovny metod Scikit-learn je ukázán v příloze E a ukázka zdárného výsledku predikce je zobrazena níže.



Obrázek 32 predikce týdenního zatížení pomocí Gradient Boosted Decison Trees

V tomto případě nebylo tak snadné určit relativní odchylku, protože některá data se nacházejí pod predikčním intervalem a některá nad ním. Proto se počítala relativní odchylka hodnot spodní a horní meze. Spodní odchylka pro tento výsledek vyšla 2,61 % a horní odchylka 2,37 %. Jedná se o výsledky pěkné, nicméně stejně jako v předchozím případě platí, že nebylo možné tuto predikci aplikovat na data celého roku, protože při velkých změnách v systému nastávaly příliš velké odchylky.

6.2 Predikce spotřeby velkoodběratele

Druhým zkoumaným objektem je areál velkoodběratele. Hlavním specifikem tohoto odběru jsou naměřená data ve formátu čtvrthodinových maxim, která podstatně snižují množství vstupních dat a díky tomu urychlují celý proces výpočtu predikcí. Další výhodou tohoto odběru je větší konzistentnost v naměřených datech a menší množství dat rušivých. Díky tomu bylo možné dosáhnout i lepších výsledků při aplikaci stejného přístupu k predikci než tomu bylo u distribučního transformátoru.

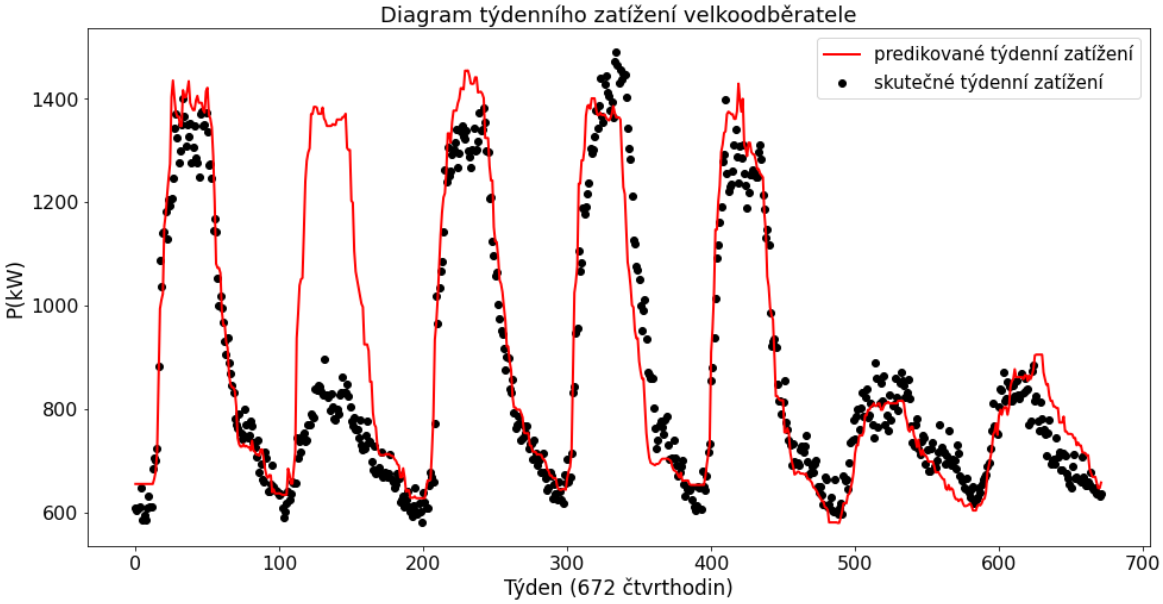
6.2.1 Predikce spotřeby velkoodběratele pomocí Adaboost

Základem pro predikci byly znovu zvoleny první dva týdny roku a postupně se v programovém cyklu postupovalo celým rokem a vytvářely se predikce týdnů následujících. Tabulka níže reprezentuje, jak predikce po jednotlivých týdnech dopadla. Predikce byla prováděna na datech z roku 2018.

Tabulka 3 relativní odchylka po týdnech pro velkoodběratele pro AdaBoost

Relativní chyba po týdnech										
týden	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**	**10.**	**11.**	**12.**
error (%)	5,37	4,05	6,93	6,85	4,90	6,09	6,84	18,89	12,33	13,10
týden	**13.**	**14.**	**15.**	**16.**	**17.**	**18.**	**19.**	**20.**	**21.**	**22.**
error (%)	20,58	15,24	7,59	4,22	5,51	9,66	5,03	7,33	5,41	6,02
týden	**23.**	**24.**	**25.**	**26.**	**27.**	**28.**	**29.**	**30.**	**31.**	**32.**
error (%)	4,62	8,49	8,68	8,97	10,44	8,67	5,13	6,32	7,82	8,54
týden	**33.**	**34.**	**35.**	**36.**	**37.**	**38.**	**39.**	**40.**	**41.**	**42.**
error (%)	10,18	9,35	12,39	5,27	4,82	5,03	10,01	6,64	4,82	5,18
týden	**43.**	**44.**	**45.**	**46.**	**47.**	**48.**	**49.**	**50.**	**51.**	
error (%)	5,12	5,15	3,56	6,93	9,33	8,47	7,46	6,96	10,39	

Z tabulky je patrné, že predikce mnohdy dopadly velice dobře. Velké odchylky, které vznikaly, byly většinou způsobeny změnami v zatížení, které jsou v běžném ohledu na spotřebu přirozené, avšak tento zvolený přístup k predikci je nedokáže podchytit. Jedná se tedy především o státní svátky, které jsou specifické výrazně nižší spotřebou, především pokud se nacházejí v době pracovních dnů. Pro vylepšení predikce by tedy bylo potřeba hledat možnosti, jak takovéto dny během vytváření predikce ošetřit, protože současně zvolený přístup na to není vhodný. Příklad, jak predikce týdenní spotřeby vypadá, pokud je v týdnu státní svátek, je znázorněná na obrázku 33.



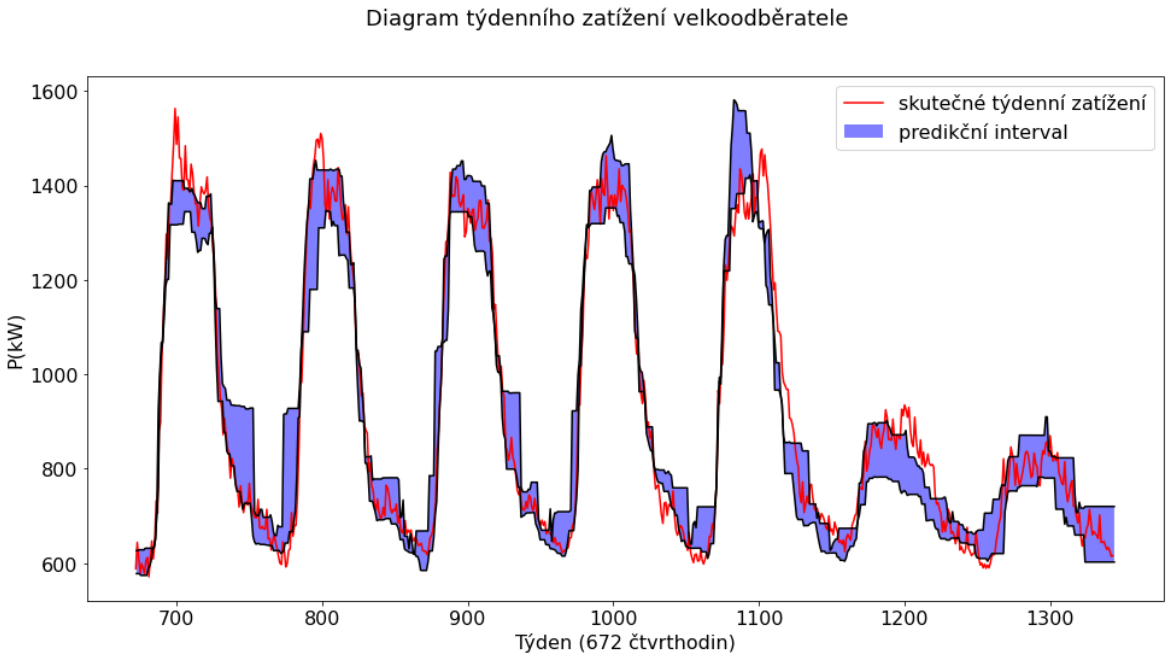
Obrázek 33 státní svátek v predikci týdenního zatížení pomocí AdaBoost

6.2.2 Predikce spotřeby velkoodběratele pomocí Gradient Boosted Decision Trees

Podobně jako u předchozí metody dopadly i výsledky pro metodu Gradient Boosted Desision Trees, jen forma výsledků se liší. V tabulce níže jsou zaznamenány relativní odchylky od horní a dolní meze po jednotlivých týdnech a na obrázku 34 je graf ukazující výsledek predikce v jednom zvoleném týdnu.

Tabulka 4 relativní odchylka po týdnech pro velkoodběratele pro Gradient Boosted Decision Trees

Relativní chyba po týdnech										
týden	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**	**10.**	**11.**	**12.**
horní odchylka (%)	5,59	2,69	4,35	6,76	3,26	5,99	6,18	1,63	12,94	11,59
dolní odchylka (%)	3,01	3,55	7,77	3,71	5,60	3,45	4,32	15,56	10,42	8,96
týden	**13.**	**14.**	**15.**	**16.**	**17.**	**18.**	**19.**	**20.**	**21.**	**22.**
horní odchylka (%)	5,27	15,34	10,76	4,42	4,59	5,69	3,45	12,15	5,87	6,02
dolní odchylka (%)	17,89	13,58	7,39	4,04	5,71	13,19	4,92	3,87	4,00	5,44
týden	**23.**	**24.**	**25.**	**26.**	**27.**	**28.**	**29.**	**30.**	**31.**	**32.**
horní odchylka (%)	4,90	3,98	9,01	4,85	6,91	11,03	4,92	6,43	7,38	4,76
dolní odchylka (%)	3,61	7,77	6,78	8,46	16,38	4,16	4,18	4,20	5,21	8,88
týden	**33.**	**34.**	**35.**	**36.**	**37.**	**38.**	**39.**	**40.**	**41.**	**42.**
horní odchylka (%)	6,93	5,80	5,11	5,55	4,73	4,07	6,35	8,83	3,32	5,43
dolní odchylka (%)	10,02	11,49	11,11	3,88	3,67	4,73	14,78	4,69	5,56	3,11
týden	**43.**	**44.**	**45.**	**46.**	**47.**	**48.**	**49.**	**50.**	**51.**	
horní odchylka (%)	5,05	3,25	3,09	7,20	7,46	7,08	2,37	6,36	4,38	
dolní odchylka (%)	5,17	5,69	3,80	3,61	8,33	6,20	7,55	4,35	11,20	



Obrázek 34 predikce týdenního zatížení promocí Gradient Boosted Decision Trees

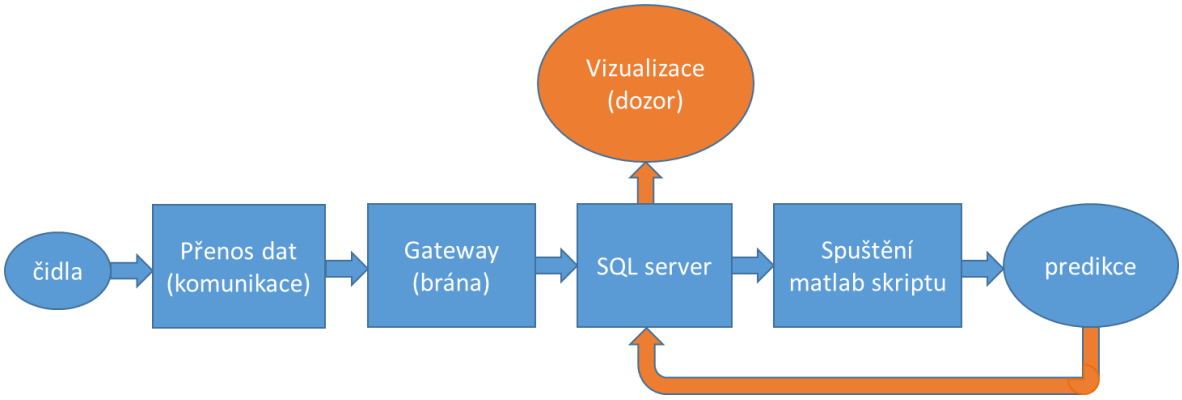
7 Návrh systému pro online monitoring

Dalším bodem této práce je zamyšlení se nad systémem, který by dokázal poskytovat informace o predikci spotřeby v reálném čase. Doposud zvolené metody pracovaly jen se statickými daty, která už byla někdy v minulosti získána, a na jejich základě by se typově modelovaly predikce spotřeby. V systému pro online monitoring bude myšlenka založena na tom, že se data budou neustále aktualizovat a tomu se bude přizpůsobovat i predikce spotřeby. Tento přístup by tedy měl simulovat situaci, kdy ve sledovaném objektu probíhá neustálé měření spotřeby a na základě již získaných dat bude probíhat i predikce. Predikovat se bude jen krátký nadcházející časový úsek, který se následně porovná i se skutečnými naměřenými hodnotami. Každá nová naměřená hodnota potom bude přidána k základnímu balíku dat, ze kterého se predikce vytváří, a dojde k přepočítání prediktivního modelu a vytvoří se opět predikce okamžiku následujícího. Díky tomuto přístupu bude docházet k neustále aktualizaci jak naměřených dat, tak i prediktivního modelu, který by potom mohl dokázat podchytit i neočekávané změny v systému a přizpůsobovat se jim.

Vhodným objektem, který sledovat, by mohl být objekt velkoodběratele, který byl zpracováván i v předchozích kapitolách a mohla by se tak řídit čtvrthodinová maxima. Mít informaci o tom, jak velké bude množství odebírané energie, může být pro velkoodběratele užitečná informace, protože velkoodběratelé většinou mívají s distributory elektrické energie smlouvy o tom, kolik energie můžou ve čtvrthodinových úsecích během dne spotřebovat, i proto se tomu říká čtvrthodinové maximum. Pokud hodnotu čtvrthodinového maxima překročí, doplácejí na to finančně. Díky dobré predikci by však mohli velkoodběratelé vhodně řídit spotřebu svého areálu a zbytečnému doplácení se tak vyhnout.

Nástrojem pro predikci se opět stanou umělé neuronové sítě tak, jak byly použity i v předchozích kapitolách. Jejich vlastnosti se ukázaly být jednoznačně lepší, než možnosti, které nabízely metody rozhodovacích stromů.

Schématický návrh systému:



Obrázek 35 Návrh systému pro online monitoring

Návrh na schématu je pouze teoretický, k jeho realizaci v rámci této práce nedošlo. Systém by fungoval tak, že by nejprve na měřicích přístrojích došlo k měření aktuální spotřeby. Informace o spotřebě by následně byla poslána přes nějaké komunikační prostředí, kupříkladu LoRa. Pro zajištění bezpečnosti by data vstupovala do SQL databáze přes bránu, aby došlo k oddělení databáze a přenosu. Následně by data z SQL serveru mohla být importována do prostředí Matlabu, kde by se spustil výpočetní skript a následně by se vytvořila predikce. Informace o predikci by byla přesměrována zpátky do SQL serveru, nad nímž by mohl být stanoven dozor, který by informace o predikci vyhodnotil.

7.1 Monitorování následujících čtvrthodin

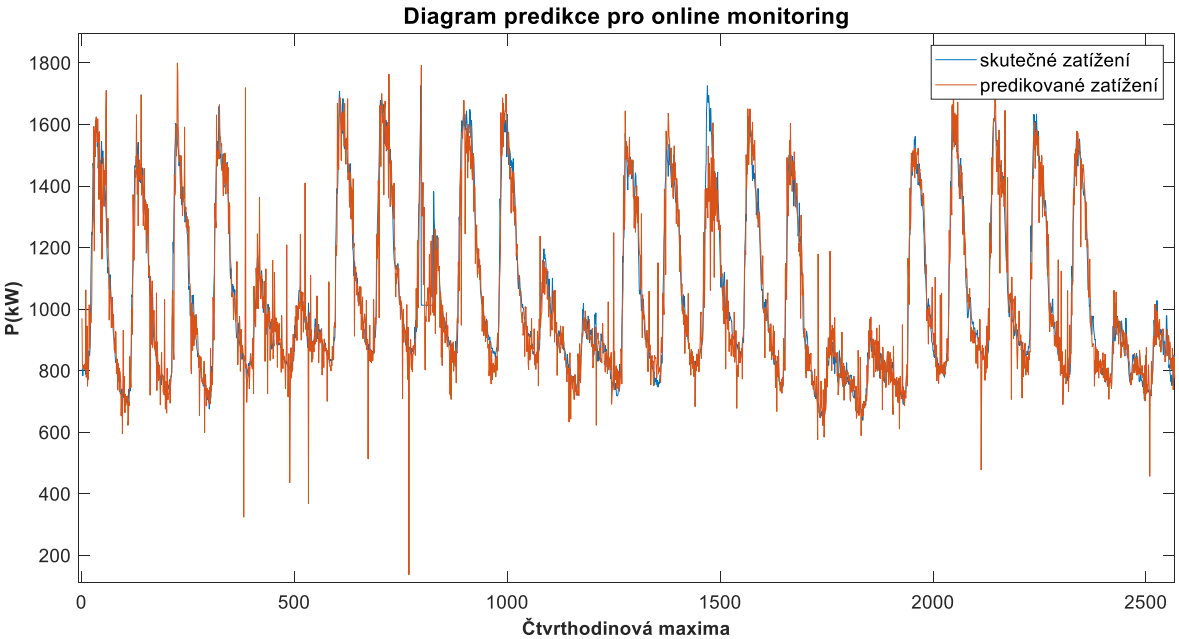
Prvním příkladem, jak by takový systém pro online monitoring mohl probíhat, je predikování následující jedné čtvrthodiny. Simulovaná situace bude vypadat následovně, objekt je k monitorování připojen krátce, a tedy není k dispozici velká sada dat a samotná predikce započne po dvoutýdenním měření. Neuronová síť se tedy naučí na dvou týdnech a odpredikuje následující čtvrthodinu a zapíše se predikovaná hodnota. Následně se predikovaná hodnota porovná s hodnotou, která skutečně nastala a spočte se relativní odchylka, která se také uloží. Potom, co tento proces proběhne, následuje další krok a to ten, že se nová naměřená hodnota přibalí do balíku dat prvních dvou týdnů a znovu dojde k přepočítání neuronové sítě, která následně odpredikuje v pořadí druhou čtvrthodinu. Proces se takto opakuje stále dokola. Výsledkem je to, že sada dat, na které se neuronová síť učí, je stále větší, což je ale nezbytné, aby predikce zůstávala aktuální. Taktéž to bude mít pozitivní vliv na vývoj relativní odchylky.

Celá situace je namodelována pouze v prostředí Matlab, kde výpočet probíhá v programovém cyklu a výsledky jsou postupně ukládány do proměnných k následnému

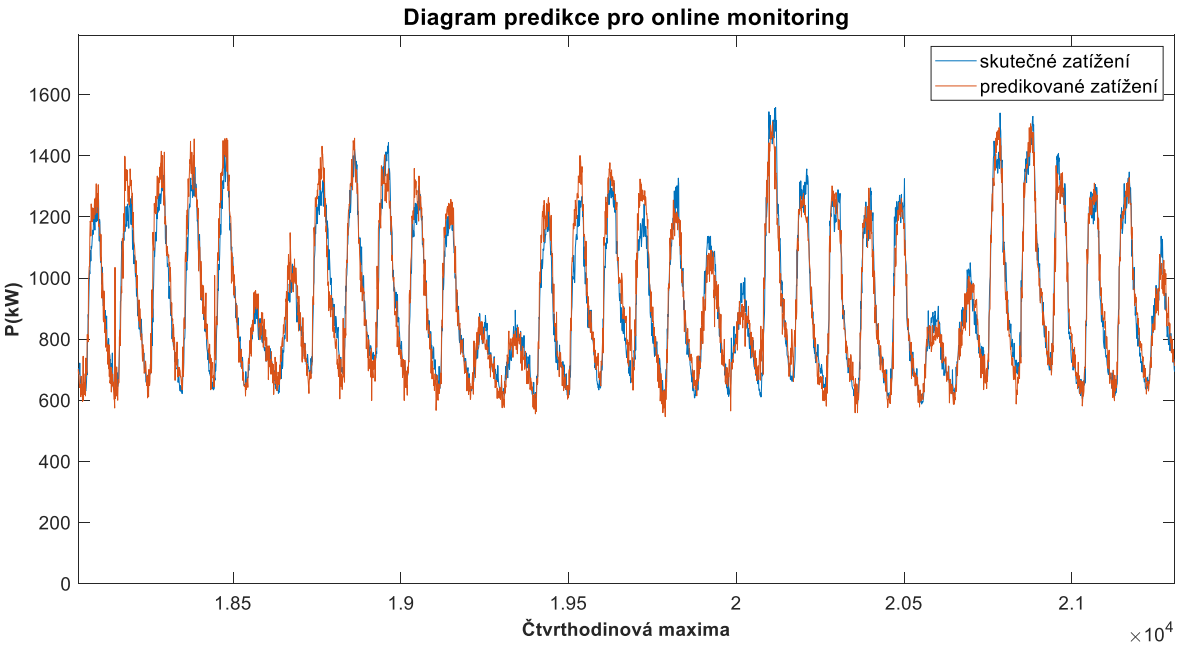
vyhodnocení. Jako vstupní data posloužily naměřené údaje z roku 2019. Z hlediska výpočetní techniky byl tento výpočet velice náročný, protože docházelo k neustálému přepočítávání neuronové sítě a s narůstajícím počtem dat se výpočet neustále prodlužoval. Původním cílem bylo dopočítat se až do konce roku, ale jelikož výpočet běžel už v řádu dní, tak byl zastaven zhruba ve dvou třetinách roku s tím, že vzorek dat bude dostačující. Přesněji to znamená zastavení po 21306 přepočtech běhu cyklu programu.

Také bylo třeba upravit vstupní matici, jelikož vstupní sada dat tvořila pouze dva týdny. Ze vstupní matice musela být odstraněna roční období a údaj o měsíci v roce. Je to z toho důvodu, že by dlouho neexistovaly v neuronové síti vzory pro tyto parametry a v momentě, kdy by nastaly, tak by došlo k zarušení celého výpočtu. Ve vstupní matici byly zahrnuty i údaje o počasí.

Ze zaznamenaných dat z výpočtu je možné sestavit podobné grafy jako v kapitole 5.3 o predikci spotřeby velkoodběratele jen s tím rozdílem, že nebyla vytvořena predikce celého dlouhého období, ale graf je složen z dílčích predikcí jednotlivých čtvrthodin. Pro lepší přehlednost budou uvedeny dva detaily. Na prvním bude začátek sledovaného období, na kterém budou patrné velké odchylky v predikci, způsobené především malým množstvím vstupních dat a na druhém obrázku bude konec sledovaného období, kde je patrné, že predikce už pobíhala velice přesně.



Obrázek 36 Začátek sledovaného období online monitoringu



Obrázek 37 Konec sledovaného období online monitoringu

Pro celkové zhodnocení online monitoringu bylo ještě zapotřebí spočítat relativní odchylku celého sledovaného období, která vyšla pouhých 5,51 %. Jedná se tedy o doposud nejlepší výsledek predikce a je třeba brát v potaz, že je v této odchylce započten i začátek sledovaného období, kdy docházelo ještě ke značným výkyvům kvůli malému množství tréninkových vzorů. Takto navržený systém pro online monitoring by byl bezesporu vhodný pro predikci spotřeby elektrické energie, protože je schopen vytvářet nejen přesné výpočty, ale zároveň může začít fungovat již po krátké době poté, co v objektu začne probíhat měření spotřeby.

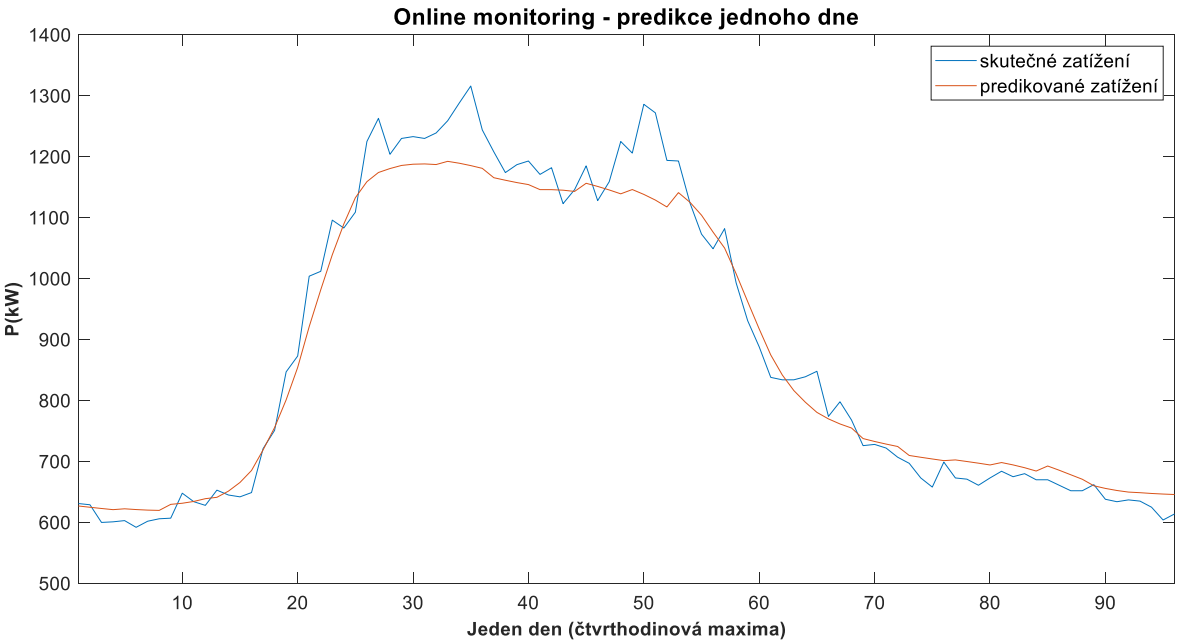
7.2 Monitorování 24 hodinových úseků

Dalším možným příkladem, jak přistupovat k systému pro online monitoring je nepredikovat každé čtvrthodinové maximum, ale vytvořit predikci celého následujícího dne a po uplynutí toho dne, znovu systém přepočítat a vytvořit predikci dne druhého. Podstatně by se tím snížilo množství prováděných výpočtů a to na 365 přepočítání místo předchozích 35040, pokud by výpočet běžel na datech celého roku.

Volba vstupních dat se oproti předchozímu příkladu lišila. Byla zvolena situace, kdy už je k dispozici rok naměřených dat a online monitoring se spouští až v roce následujícím. Tedy na konci sledovaného období bude neuronová síť určovat výsledek z takřka dvou let dat. Díky většímu množství vstupních dat mohly zůstat ve vstupní matici pro trénování neuronové sítě všechny modelované parametry, protože v tomto případě už neuronová síť měla k dispozici vzory pro jednotlivé měsíce či roční období, neboť měla k dispozici rok

stará data. Taktéž jsou mezi údaji i parametry o počasí. S ohledem na systém pro online monitoring by i tyto parametry měly dávat smysl, protože denní předpověď počasí bývá poměrně spolehlivá, a tak se muže nacházet i v predikčním modelu.

Vstupními daty se stal rok měření 2019 a postupná predikce probíhala na roce 2020. Z predikčního hlediska se jedná o podobnou situaci jako v kapitole 5.3.1, kde se odpredikoval celý rok 2020 z předchozích dvou let měření. Proto může být zajímavé porovnat relativní odchylku případu, kdy predikce probíhá na staré neuronové síti, a případu, kdy dochází k jejímu neustálému aktualizování.



Obrázek 38 Příklad jednodenní predikce systému pro online monitoring

Na obrázku výše je příklad jednodenní predikce ze sledovaného období. Zhodnocení celé predikce pro online monitoring je takové, že bylo dosaženo relativní chyby 6,33 % na celém roce 2020. Na straně druhé, pokud neuronová síť predikovala pouze na starých datech, byl výsledek pro rok 2020 7,39 %. Z těchto výsledků vyplývá, že pracovat s aktuálními daty se určitě vyplatí, protože došlo k celkovému zlepšení relativní odchylky o 1,06 %.

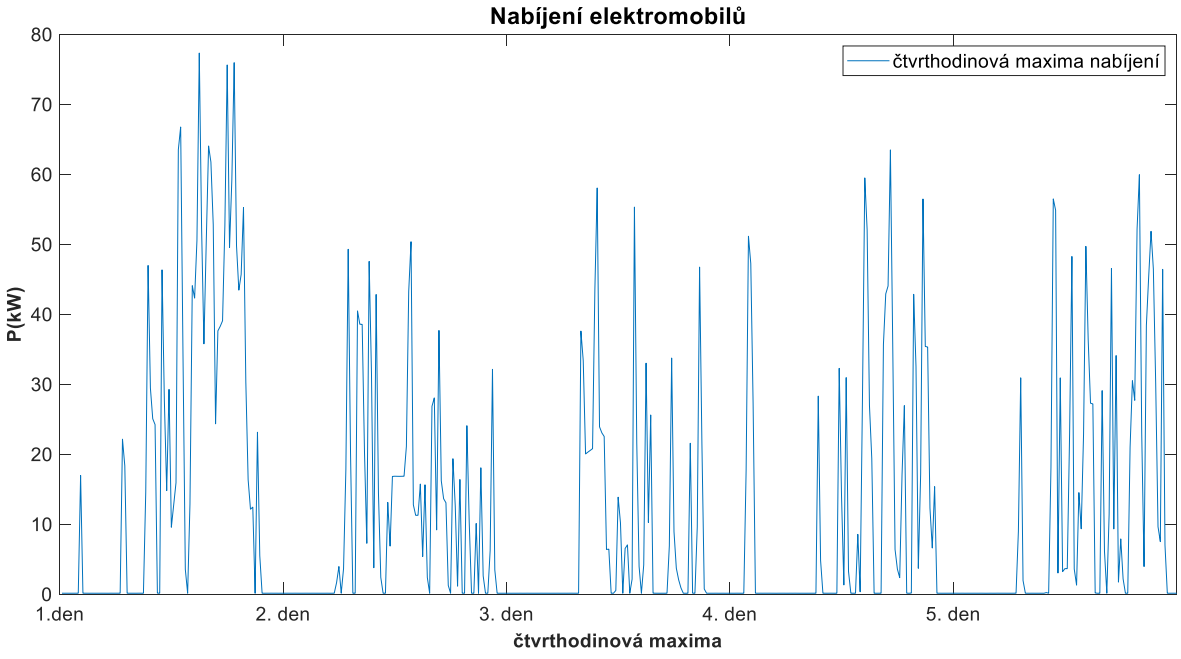
Dalším zajímavým pohledem by mohlo být srovnání i pro online monitoring, který se aktualizuje každou čtvrthodinu. Bohužel v této práci byl tento výpočet vypracován pro rok 2019 a ani nebyl dokončen výpočet na celém roce s ohledem na náročnost výpočtu pro výpočetní techniku. Nicméně u prvního modelu online monitoringu bylo dosaženo ještě menší relativní chyby a mohlo by být zajímavé se ubírat tímto směrem a hledat druh výpočtu, který obecně dosahuje nejlepších výsledků. V této práci však na to již nezůstal

prostor s ohledem na časovou náročnost výpočtů, a protože způsoby, jak přistupovat k predikci a hledání nových přístupů by mohl být nekončící.

8 Vliv elektromobility na predikci spotřeby

Dalším tématem k zamyšlení je, jakým způsobem nabývající elektromobilita může zasáhnout do predikce spotřeby elektrické energie. S elektromobily je spojena i jejich nutnost nabíjení se a právě při nabíjení dochází k zásahům do diagramu zatížení. Charakteristické pro nabíjení elektromobilů je, že nárůst odebírané energie je velice strmý a stejně potom i její pokles.

K této práci byla poskytnuta také reprezentativní data z nabíjení elektromobilů. Jednalo se celkem o 71 dní, kdy probíhalo nabíjení elektromobilů a měřen byl jejich odebíraný výkon. Data jsou z období konce července a začátku října roku 2019. Jak by odběr při nabíjení vypadal, je ukázáno na grafu níže, kde je záznam z pěti dnů nabíjení. Data z nabíjení byla původně ve formátu aktuálního výkonu zaznamenaného s minutovým krokem, jelikož ale data budou použita na data velkoodběratele, bylo třeba je předělat do formátu čtvrthodinových maxim.



Obrázek 39 Nabíjení elektromobilů

Otázkou dále zůstává, na jaký prediktivní model data z nabíjení superponovat. Pokud by data byla přidána do predikovaného období s tím, že učení sítě předtím probíhalo bez dat o nabíjení, je jasné, že by chyba predikce vzrostla prakticky o tolik, kolik energie nabíječky elektromobilů ze sítě odebíraly.

Vhodný přístup by tedy byl, pokud by do něj data z nabíjení zasahovala rovnou. Takovým případem může být systém pro online monitoring, ve kterém se hodnoty aktualizují po každé uplynulé čtvrthodině. Sledovaným obdobím se stal úsek oněch zmíněných 71 dní mezi koncem června a začátkem října roku 2019. Nejprve se vzala data bez elektromobility a vyhodnotila se relativní chyba. Následně se k naměřeným datům přičetla data z nabíjení elektromobilů a online monitoring se na sledovaném období spustil znovu. Výsledek dopadl tak, že výpočet online monitoringu na sledovaných 71 dní dosáhl relativní odchylky 5,41 %. Pro zahrnutou elektromobilitu výsledek dopadl s relativní odchylkou 5,51 %. Z výsledku plyne, že pokud dáte neuronové síti možnost se aktualizovat i za předpokladu, že se v systému nově objeví elektromobilita, výsledek predikce to příliš nezhorší. Nutnou poznámkou by ještě mělo být, že maximální zatížení se pro velkoodběratele ve sledovaném období pohybovalo kolem 1600 kW a elektromobilita přispívala maximálně 100 kW. Zásah do spotřeby to tedy určitě byl, ale nikoliv nějak zvlášť kritický. Každopádně schopnost neuronové sítě adaptovat se je v tomto ohledu stále velice zajímavá. Dále pro dovysvětlení, systém online monitoringu započal výpočet ve dni, kdy byla poprvé zapojena elektromobilita, tedy fungoval podle toho, jak byl navržen v kapitole 7.1 a na počátku výpočtu už disponoval daty od začátku roku 2019 až do dne, kdy byla do systému zapojena elektromobilita, a pak se dále obvyklým způsobem aktualizoval.

9 Přítomnost bateriových uložišť v okolní síti

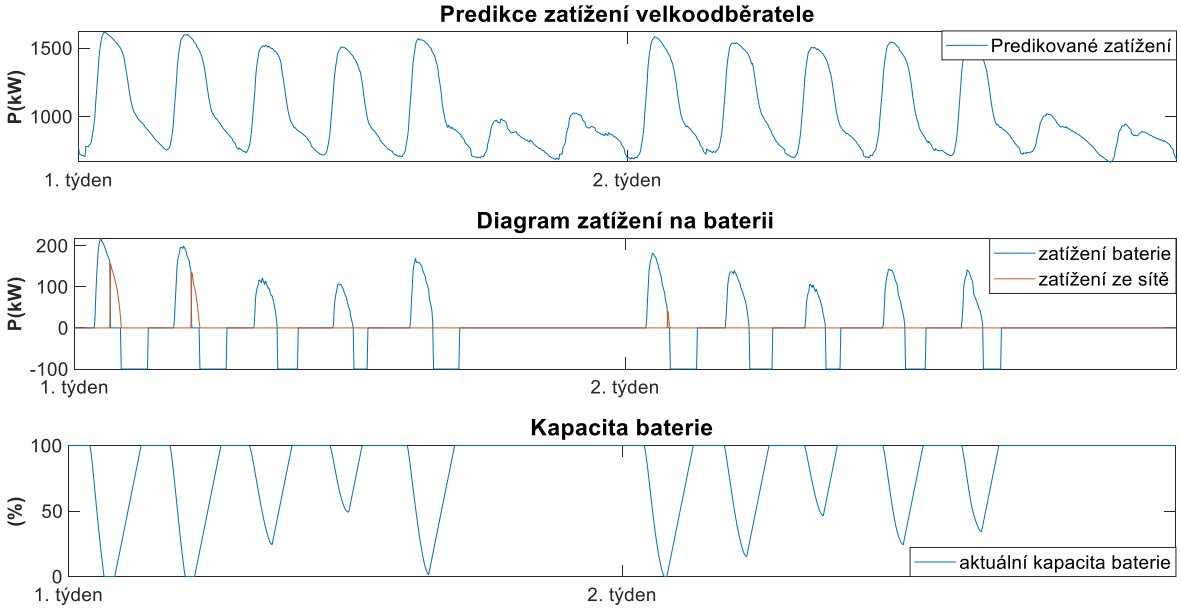
Posledním zamyšlením k tématu bude, jaký vliv, respektive jaké řešení, by se s ohledem na predikci dalo najít za využití bateriových uložišť. Důvodem, proč vůbec bateriová uložiště připojovat, může být již zmíněná problematika toho, že pokud velkoodběratel překročí smluvně dané hladiny pro čtvrthodinová maxima, tak potom doplácí extra poplatky za energii, kterou čerpal navíc. Díky přítomnosti bateriových uložišť by tak nemusel elektrickou energii navíc čerpat ze sítě, ale pokryl by jí z baterií. Analogicky by tento případ seděl na situaci, kdy je při pokrývání diagramu zatížení pro pokrývání špičkového zatížení využíváno například přečerpávacích vodních elektráren. V našem případě by se jednalo o pokrývání špičkového zatížení pomocí bateriových uložišť. Tak i tak se jedná o obecně problematickou oblast v diagramu zatížení.

Pro modelovou situaci se opět využilo dat velkoodběratele a pro určitou jednoduchost bylo využito již zhotovené predikce spotřeby pro rok 2020, která byla zhotovena na základě dvou předchozích let. Pracovalo se tedy s predikční křivkou pro rok 2020 a v rámci systému, který by byl obohacen o bateriová uložiště, bylo potřeba vytvořit systém podmínek, který by přítomnost baterií reprezentoval.

Systém byl navržen pouze typově, neboť v reálu by musel reagovat na určitou syntézu požadavků jak provozovatele, tak poskytovatele, a rovněž by musel odpovídat technickým možnostem a parametrům. Nastavenou horní mezí pro maximální odebíraný výkon byla zvolena hladina 1400 kW a kapacita baterie byla nastavena na 800 kWh. Předpokládané nabíjení baterie bylo 100 kW.

Ukázka, jak by systém fungoval, je znázorněna na grafech níže, kde první graf představuje predikované zatížení. Následující graf ukazuje modrou křivkou zatížení na baterii pro případ, že se celkové zatížení objektu pohybuje nad 1400 kW. Ve stejném grafu je oranžovou křivkou zaznamenána i situace, kdy se baterie vybije a je třeba začít opět odebírat výkon ze sítě, který je ovšem nad stanovený limit a bylo by třeba za něj doplácet. Hlavním smyslem modelu je tedy ukázat, kdy by během sledovaného období docházelo k situacím, že předpokládaná odebíraná energie by byla tak velká, že by bateriová uložiště nestačila a bylo tak třeba odebíranou energii ještě extra doplácet. Poslední graf zaznamenává aktuální procentuální kapacitu baterií.

K situacím během roku, kdy by kapacita baterie na pokrývání špiček nestačila, docházelo jen zřídka. Pro ukázku ale byla zvolena taková část roku, na která by bylo dobře vidět všechny situace, které mohou nastat.



Obrázek 40 Systém s bateriovými uložišti

Záporná hodnota výkonu v prostředním grafu znamená dobíjení baterií v momentě, kdy se jim uvolnil výkon. V modelu nebylo uvažováno provozování baterie s bezpečnostními mezemi pro prodloužení životnosti baterie mezi 20 – 90 % její kapacity. V tomto vzorovém případě byla využita pro regulaci celá kapacita baterie.

Závěr

Hlavním cílem práce bylo předpovídání, respektive predikování, spotřeby elektrické energie. Jelikož se jedná o určování budoucnosti, tj. jevu, který ještě nenastal, výsledek může být vždy nejistý. Nicméně i informace, která není zcela jistá, může poskytnout dobrou představu o nadcházejících událostech. Proto si i tato práce dala za cíl sestavit modely, které by dokázaly na základě různých získaných údajů předpovědět, jak bude spotřeba elektrické energie v následujících časových úsecích vypadat. Důvodem, proč mít informace o spotřebě dopředu, bezpochyby může být lepší možnost řízení jak elektrizační soustavy, tak spotřeby, což může vést k celkově ekonomičtějšímu provozu nebo také k většímu šetření životního prostředí, díky lepšímu rozvržení zdrojů elektrické energie.

Prvním krokem práce bylo vytvoření menšího průzkumu toho, jaké metody se dnes využívají pro predikování spotřeby, aby následně bylo možné z těchto metod vybrat takové, které by se mohly uplatnit v předložené diplomové práci. Kromě kvalitativních parametrů metod bylo také třeba vybrat takové, které by byly dostupné jak z hlediska programové dostupnosti, náročnosti na výpočetní techniku, tak i jejich realizačních možností. Vybrány byly nakonec dvě metody, a to umělé neuronové sítě a metoda rozhodovacích stromů. Majoritní pro tuto práci se potom staly umělé neuronové sítě, neboť z vědeckých článků, ze kterých práce čerpala, vyplynulo, že tato metoda vychází jednoznačně nejlépe, jak už z hlediska dosažených výsledků, tak i svou univerzálností.

Před samotnou aplikací zvolených metod k nim byla ještě rozebrána patřičná teorie a také byly nastíněny elektroenergetické pojmy nezbytné pro tuto práci. Prvním zpracovaným modelem se staly umělé neuronové sítě, u kterých bylo využito Levenberg-Marquardtova algoritmu neuronové sítě typu backpropagation. Druhým v pořadí bylo vytvoření modelu rozhodovacích stromů, u kterých se využilo regresních algoritmů AdaBoost a Gradient Boosted Decison Trees. Metody byly aplikovány na data z distribučního transformátoru napájejícího část města Plzeň, kde bylo k dispozici roční měření činného výkonu s minutovým krokem. Druhým vzorkem dat byl potom velkoodběratel, představující zdravotní zařízení, u něhož byly k dispozici tři roky naměřených údajů ve formátu takzvaných čtvrthodinových maxim.

U neuronových sítí byly vytvářeny především dlouhodobější predikce, kde bylo dosahováno výsledků s relativní odchylkou od skutečnosti často do 10 %. V různých detailech částí predikce někdy mohla chyba odskočit i na vyšší hodnoty, ale zároveň i na hodnoty nižší. Celkově však byly výsledky poměrně konzistentní. Zajímavým

výsledkem je predikce celého roku 2020 pro velkoodběratele, kde bylo dosaženo celkové relativní chyby 8,84 % pro model založený čistě na časových parametrech a chyby 7,39 % pro stejný model, ale obohacený o parametry počasí. Diskutovány byly i různé přístupy k predikci a dále také bylo řešeno, jaké zvolené parametry jsou kdy vhodné, či jaké je optimální množství tréninkových vzorů.

U rozhodovacích stromů byl volen přístup krátkodobých predikcí. Získané výsledky byly obecně méně stabilní oproti umělým neuronovým sítím a kvalita výsledků byla úzce spjata s kvalitou naměřených dat. Například pro distribuční transformátor tato metoda nebyla příliš vhodná, protože v naměřených hodnotách na distribučním transformátoru docházelo k častým výkyvům či změnám, které mohly být způsobeny operacemi provozovatele distribuční soustavy. Kvůli tomu pak docházelo k rozkolísání predikcí, jež vedlo k jejich nepoužitelnosti. Pro velkoodběratele tato metoda byla vhodnější, protože spotřeba velkoodběratele se během roku měnila více spojitě, a tak nedocházelo k takovému zarušení výsledků.

Další částí práce se potom stal ještě systém pro online monitoring, u kterého byla založena myšlenka na ještě sofistikovanějším výpočtu predikcí, který by mohl probíhat v reálném čase, a tak poskytovat informace odběrateli o tom, jakou má aktuálně očekávat spotřebu. Systém se opět opíral o umělé neuronové sítě. Zpracovaným objektem byl v tomto případě pouze velkoodběratel a při postupném odpredikování roku 2020 bylo dosaženo celkové relativní chyby 6,33 %, což je lepší než předchozí nejlepší dosažený výsledek, jenž byl 7,39 % při predikci na statických datech.

V závěru práce byl ještě zohledněn vliv potenciální přítomnosti elektromobility, která by mohla predikce negativně ovlivňovat. V rámci této kapitoly se využilo již zhotoveného systému pro online monitoring a sledovalo se období 71 dnů, kde byly k datům velkoodběratele přidány hodnoty z nabíjení elektromobilů. Sledovaný úsek se pak vyhodnotil za předcházejícího stavu bez elektromobility a nově s přidanou elektromobilitou a bylo zjištěno, jak negativně by byla predikce spotřeby přítomností elektromobilů zasažena. Nicméně bylo dosaženo velice uspokojivého výsledku, kdy po zapojení elektromobility došlo ke zhoršení predikce pouze o 0,10 %. Posledním návrhem se potom stal systém obohacený o bateriová uložiště, který by čerpal z informací o predikci, aby se dal odběr spotřebitele řídit co nejlépe a bylo dosaženo jeho nejekonomičtějšího provozu. Princip spočíval v pokrývání denních špičkových zatížení ve dnech s vysokou spotřebou elektrické energie tak, aby nemuselo docházet k doplatkům velkoodběratele za příliš velké čerpání elektrické energie.

Seznam obrázků

Seznam literatury a informačních zdrojů

[1] TSO, Geoffrey K.F. a Kelvin K.W. YAU. Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks. Elsevier Ltd. [online]. 2006, , 8 [cit. 2020-12-01]. Dostupné z: https://www.journals.elsevier.com/energy

[2] VOLNÁ, Eva. EVOLUČNÍ ALGORITMY A NEURONOVÉ SÍTĚ. 1. Ostrava: Ostravská univerzita v Ostravě, 2012.

[3] BISHOP, Christoprher. Pattern Recognition and Machine Learning. Cambridge: Springer Science, 2006. ISBN 0-387-31073-8.

[4] HORKÝ, L. a K. BŘINDA. Neuronové sítě. Fakulta jaderná a fyzikálně inženýrská [online]. Břehová 7, 115 19 Praha 1 [cit. 2020-12-01].

[5] Neuronové sítě. Mendel University in Brno [online]. [cit. 2020-12-01]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=21471

[6] ZURADA, Jacek M. Introduction to artificial neural systems. St. Paul: West, c1992. ISBN 0-314-93391-3.

[7] YE, Zhaoyang a Moon Keun KIMB. Predicting electricity consumption in a building using an optimized backpropagation and Levenberg–Marquardt back-propagation neural network: Case study of a shopping mall in China. ElsevierLtd. [online]. 2018, , 8 [cit. 2020-12-01]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2210670718300842?via%3Dihub

[8] CETIN, Kristen, In HO CHO a Elham JAHANI. Building and Environment: City-scale single family residential building energy consumption prediction using genetic algorithm-based Numerical Moment Matching technique. Elsevier Ltd. [online]. 2020, 13.1. 2020, , 10 [cit. 2020-12-15]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0360132320300251?via%3Dihub

[9] ZEHAN, Zehan TAN, Chengliang XU, Huanxin CHEN a Zhengfei LI. Energy & Buildings: Study on deep reinforcement learning techniques for building energy consumption forecasting. Elsevier Ltd. [online]. 2020, , 14 [cit. 2020-12-15]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0378778819324740?via%3Dihub

[10] Scikit-learn: Machine learning in Python [online]. 2007 [cit. 2021-04-16]. Dostupné z: https://scikit-learn.org/stable/

[11] Elektroenergetika: výroba elektrické energie. Inovace VOV [online]. ČVUT v Praze, Fakulta elektrotechnická [cit. 2021-04-18]. Dostupné z: https://www.vovcr.cz/odz/tech/284/page02.html

[12] Účinnost velkých bateriových úložišť a přečerpávacích elektráren v USA je srovnatelná. OENERGETICE.cz [online]. 2021, 17. 2. [cit. 2021-04-20]. Dostupné z: https://oenergetice.cz/akumulace-energie/ucinnost-velkych-bateriovych-ulozist-precerpavacich-elektraren-usa-srovnatelna

[13] SVĚT ENERGIE: VZDĚLÁVACÍ PORTÁL ČEZ [online]. 2020 [cit. 2021-04-20]. Dostupné z: https://www.svetenergie.cz/cz/energetika-zblizka/energetika-mest-a-domacnosti-smart-city/mestska-chytra-energetika/elektromobilita/vyklad

[14] MathWorks [online]. 1994-2021 The MathWorks [cit. 2021-5-4]. Dostupné z: https://www.mathworks.com/

Přílohy

Příloha A – Matlab skript – Příprava vstupních dat

Příloha B – Matlab skript – Neuronová síť

Příloha C – Jupyter Notebook skript – Načítání dat

Příloha D – Jupyter Notebook skript – příklad AdaBoost

Příloha E – Jupyter Notebook skript – příklad Gradient Boosted Decison Trees

Příloha F – Matlab skript – Systém pro online monitoring

Příloha A – Matlab skript – Příprava vstupních dat

```
% Úprava dat a vytvoření vstupní matice
clc
clear all

load('Plzen_sever.mat')

O1=Ostrov2_PT(1:end,1);
O2=Ostrov2_PT(1:end,2);

O1T=O1';
O2T=O2';
target=O1T+O2T;

for(k=1:525600)

    if O(1,k)<=0.002
        O(1,k)=O(1,k-1);
    end
end

%input matrix

%hodiny

n=0;
for k=1:60:1440

for r=k:k+60

    po(8,r)=n;
    ut(8,r)=n;
    st(8,r)=n;
    ct(8,r)=n;
    pa(8,r)=n;
    so(8,r)=n;
    ne(8,r)=n;


end

n=n+1/24;
end
clear k

%minuty

for k=1:60:1440
    n=0;

for r=k:k+60

    po(9,r)=n;
    ut(9,r)=n;
    st(9,r)=n;
    ct(9,r)=n;
```

```
    pa(9,r)=n;
    so(9,r)=n;
    ne(9,r)=n;
   n=n+1/60;


end

end
clear k

po(1,1:1440)=1;
ut(2,1:1440)=1;
st(3,1:1440)=1;
ct(4,1:1440)=1;
pa(5,1:1440)=1;
so(6,1:1440)=1;
ne(7,1:1440)=1;

po=po(:,1:1440);
ut=ut(:,1:1440);
st=st(:,1:1440);
ct=ct(:,1:1440);
pa=pa(:,1:1440);
so=so(:,1:1440);
ne=ne(:,1:1440);

po(10,1:1440)=0;
ut(10,1:1440)=0;
st(10,1:1440)=0;
ct(10,1:1440)=0;
pa(10,1:1440)=0;
so(10,1:1440)=1;
ne(10,1:1440)=1;

tyden=[po ut st ct pa so ne];

leden_2014=[st ct pa so ne tyden tyden tyden po ut st ct pa];
unor_2014=[so ne tyden tyden tyden po ut st ct pa];
brezen_2014=[so ne tyden tyden tyden tyden po];
duben_2014=[ut st ct pa so ne tyden tyden tyden po ut st];
kveten_2014=[ct pa so ne tyden tyden tyden po ut st ct pa so];
cerven_2014=[ne tyden tyden tyden tyden po];
cervenec_2014=[ut st ct pa so ne tyden tyden tyden po ut st ct];
srpen_2014=[pa so ne tyden tyden tyden tyden];
zari_2014=[tyden tyden tyden tyden po ut];
rijen_2014=[st ct pa so ne tyden tyden tyden po ut st ct pa];
listopad_2014=[so ne tyden tyden tyden tyden];
prosinec_2014=[tyden tyden tyden tyden po ut st];


%rocni obdobi

leden_2014(6:15,:)=leden_2014;
unor_2014(6:15,:)=unor_2014;
brezen_2014(6:15,:)=brezen_2014;
duben_2014(6:15,:)=duben_2014;
kveten_2014(6:15,:)=kveten_2014;
cerven_2014(6:15,:)=cerven_2014;
```

```
cervenec_2014(6:15,:)=cervenec_2014;
srpen_2014(6:15,:)=srpen_2014;
zari_2014(6:15,:)=zari_2014;
rijen_2014(6:15,:)=rijen_2014;
listopad_2014(6:15,:)=listopad_2014;
prosinec_2014(6:15,:)=prosinec_2014;

leden_2014(1:5,:)=0;
unor_2014(1:5,:)=0;
brezen_2014(1:5,:)=0;
duben_2014(1:5,:)=0;
kveten_2014(1:5,:)=0;
cerven_2014(1:5,:)=0;
cervenec_2014(1:5,:)=0;
srpen_2014(1:5,:)=0;
zari_2014(1:5,:)=0;
rijen_2014(1:5,:)=0;
listopad_2014(1:5,:)=0;
prosinec_2014(1:5,:)=0;

leden_2014(1,:)=1;
unor_2014(1,:)=1;
brezen_2014(2,:)=1;
duben_2014(2,:)=1;
kveten_2014(2,:)=1;
cerven_2014(3,:)=1;
cervenec_2014(3,:)=1;
srpen_2014(3,:)=1;
zari_2014(4,:)=1;
rijen_2014(4,:)=1;
listopad_2014(4,:)=1;
prosinec_2014(1,:)=1;

%mesic

leden_2014(5,:)=1/12;
unor_2014(5,:)=2/12;
brezen_2014(5,:)=3/12;
duben_2014(5,:)=4/12;
kveten_2014(5,:)=5/12;
cerven_2014(5,:)=6/12;
cervenec_2014(5,:)=7/12;
srpen_2014(5,:)=8/12;
zari_2014(5,:)=9/12;
rijen_2014(5,:)=10/12;
listopad_2014(5,:)=11/12;
prosinec_2014(5,:)=12/12;

%svatky

        leden_2014(16,1440*1-1439:1*1440)=1;
        duben_2014(16,1440*18-1439:18*1440)=1;
        duben_2014(16,1440*21-1439:21*1440)=1;
        kveten_2014(16,1440*1-1439:1*1440)=1;
        kveten_2014(16,1440*8-1439:8*1440)=1;
        cervenec_2014(16,1440*5-1439:5*1440)=1;
        cervenec_2014(16,1440*6-1439:6*1440)=1;
        zari_2014(16,1440*28-1439:28*1440)=1;
        rijen_2014(16,1440*28-1439:28*1440)=1;
        listopad_2014(16,1440*17-1439:17*1440)=1;
```

```
        prosinec_2014(16,1440*24-1439:24*1440)=1;
        prosinec_2014(16,1440*25-1439:25*1440)=1;
        prosinec_2014(16,1440*26-1439:26*1440)=1;

        unor_2014(16,:)=0;
        brezen_2014(16,:)=0;
        cerven_2014(16,:)=0;
        srpen_2014(16,:)=0;


%rok

rok_2014=[leden_2014 unor_2014 brezen_2014 duben_2014 kveten_2014
cerven_2014 cervenec_2014 srpen_2014 zari_2014 rijen_2014 listopad_2014
prosinec_2014];
```

Příloha B – Matlab skript – Neuronová síť

```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created 08-Nov-2020 22:44:59
%
% This script assumes these variables are defined:
%
%   rok_2014 - input data.
%   target - target data.

x = rok_2014;
t = target;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 32;
net = fitnet(hiddenLayerSize,trainFcn);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};

% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivision
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean Squared Error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
    'plotregression', 'plotfit'};

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% Recalculate Training, Validation and Test Performance
trainTargets = t .* tr.trainMask{1};
valTargets = t .* tr.valMask{1};
testTargets = t .* tr.testMask{1};
```

```
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)

% Deployment
% Change the (false) values to (true) to enable the following code
blocks.
% See the help for each generation function for more information.
if (false)
    % Generate MATLAB function for neural network for application
    % deployment in MATLAB scripts or with MATLAB Compiler and Builder
    % tools, or simply to examine the calculations your trained neural
    % network performs.
    genFunction(net,'myNeuralNetworkFunction');
    y = myNeuralNetworkFunction(x);
end
if (false)
    % Generate a matrix-only MATLAB function for neural network code
    % generation with MATLAB Coder tools.
    genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
    y = myNeuralNetworkFunction(x);
end
if (false)
    % Generate a Simulink diagram for simulation or deployment with.
    % Simulink Coder tools.
    gensim(net);
end
```

Příloha C – Jupyter Notebook skript – načítání dat

```
#importing required packages
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.io
import numpy
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import svm
from sklearn.neural_network import MLPClassifier
#from sklearn.lenar_model import SGDClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
%matplotlib inline

#loading dataset
load = scipy.io.loadmat('c:\ProgramData\Anaconda3\Lib\site-
packages\scipy\io\matlab\scikit_input_rok.mat')
rok = load['rok']

rok=rok[0,:]

t=514080 #sem zadej casovy interval
time=numpy.arange(t)

from matplotlib import pyplot

%matplotlib inline

pyplot.plot(time, rok)
```

Příloha D – Jupyter Notebook skript – příklad AdaBoost

```
error_matrix=[]
i=0
t=20160 #sem zadej casovy interval
time=numpy.arange(t)

for i in range(49):

    print(__doc__)

    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.tree import DecisionTreeRegressor
    from sklearn.ensemble import AdaBoostRegressor

    rng = np.random.RandomState(1)
    X=time [:, np.newaxis]
    y=dva_tydny

    # Fit regression model

    regr_2 = AdaBoostRegressor(DecisionTreeRegressor(max_depth=10),
                           n_estimators=300, random_state=rng)

    regr_2.fit(X, y)

    # Predict
    y_2 = regr_2.predict(X)

    predikce=y_2[10080:20160]
    tyden=numpy.arange(10080)[:, np.newaxis]

    # Set tick font size


    # Plot the results
    plt.figure(figsize=(16, 8))
    # Set general font size
    plt.rcParams['font.size'] = '16'

    plt.scatter(tyden, target, c="k", label="skutečné týdenní zatížení")
    plt.plot(tyden, predikce, c="r", label="predikované týdenní
zatížení", linewidth=2)
    plt.xlabel("Týden (10080 minut)", fontsize=17)
    plt.ylabel("P(MW)", fontsize=17)
    plt.title("Diagram týdenního zatížení na transformátoru vvn/vn",
fontsize=18)
    plt.legend(prop={"size":15})
    plt.show()

    e=0
    error=0

    for e in range(10080):

        var=abs(((target[e]-predikce[e])/target[e])*100)
        error=error+var

    error=error/10080
    error_matrix.append(error)
    print(error)
```

Příloha E – Jupyter Notebook skript – příklad Gradient Boosted Decision Trees

```
upper_matrix=[]
lower_matrix=[]

i=0
t=20160 #sem zadej casovy interval
time=numpy.arange(t)

for i in range(49):

    dva_tydny=rok[i*7*1440:i*7*1440+14*1440]
    target=rok[(i+2)*7*1440:(i+2)*7*1440+7*1440]
    import numpy as np
    import matplotlib.pyplot as plt

    from sklearn.ensemble import GradientBoostingRegressor

    np.random.seed(1)

    X = np.linspace(0., 20160., t, dtype = int, endpoint=False)[:,
np.newaxis]
    X = X.astype(np.float32)

    # Observations
    y = dva_tydny

    # Mesh the input space for evaluations of the real function, the
prediction and
    # its MSE
    xx = np.atleast_2d(np.linspace(0, t, t)).T
    xx = xx.astype(np.float32)

    alpha = 0.92 #0.92 dobry

    h=9
    clf = GradientBoostingRegressor(loss='quantile', alpha=alpha,
                                    n_estimators=350, max_depth=h,
                                    learning_rate=.1,
min_samples_leaf=25,
                                    min_samples_split=24)

    clf.fit(X, y)

    # Make the prediction on the meshed x-axis
    y_upper = clf.predict(xx)

    clf.set_params(alpha=1.0 - alpha)
    clf.fit(X, y)

    # Make the prediction on the meshed x-axis
    y_lower = clf.predict(xx)

    clf.set_params(loss='ls')
    clf.fit(X, y)

    # Make the prediction on the meshed x-axis
    y_pred = clf.predict(xx)


    X=X[10080:20160]
    y=target
```

```
    xx=xx[10080:20160]
    y_pred=y_pred[10080:20160]
    y_upper=y_upper[10080:20160]
    y_lower=y_lower[10080:20160]

    # Plot the function and the prediction
    fig = plt.figure(figsize=(16, 8))
    plt.rcParams['font.size'] = '16'
    #plt.plot(xx, f(xx), 'g:', label=r'$f(x) = x\,\sin(x)$')
    plt.plot(X, y, 'r',  markersize=10, label=u'skutečné týdenní
zatížení')
    #plt.plot(xx, y_pred, 'r-', label=u'Prediction')
    plt.plot(xx, y_upper, 'k-')
    plt.plot(xx, y_lower, 'k-')
    plt.fill(np.concatenate([xx, xx[::-1]]),
             np.concatenate([y_upper, y_lower[::-1]]),
             alpha=.5, fc='b', ec='None', label='predikční interval')
    plt.xlabel('Týden (10080 minut)')
    plt.ylabel('P(MW)')
    #plt.ylim(3, 20)
    plt.legend(loc='upper right')
    #fig.suptitle('Hloubka regrese %i' %h , fontsize=16)
    fig.suptitle('Diagram týdenního zatížení na transformátoru vvn/vn',
fontsize=18)
    plt.show()

    k=0
    chyba=0
    for k in range(10080):
        if y[k]<y_upper[k] and y[k]>y_lower[k]:
            pass
        else:
            chyba=chyba+1

    chyba=100*chyba/10080
    print(chyba)


    e=0
    error_upper=0
    error_lower=0
    var1=0
    var2=0

    for e in range(10080):
        if y[e]>y_upper[e]:
            upper=abs(((y[e]-y_upper[e])/y[e])*100)
            error_upper=error_upper+upper
            var1=var1+1
        elif y[e]<y_lower[e]:
            lower=abs(((y[e]-y_lower[e])/y[e])*100)
            error_lower=error_lower+lower
            var2=var2+1

    error_upper=error_upper/var1
    print(error_upper)

    error_lower=error_lower/var2
    print(error_lower)
    upper_matrix.append(error_upper)
    lower_matrix.append(error_lower)
```

Příloha F – Matlab skript – Systém pro online monitoring

```
dataload_velkoodberatel
rok_2019(1:5,:)=[];
rok_2019(9,:)=[];

%33688 cely rok
for(k=1345:33688)

input_monit=rok_2019(:,1:k);
target_monit=KNTB_2019(:,1:k);

%zahajeni vypoctu casu pri tisicovce
if mod(k,1000) == 0 || k == 1 || k == 2
    tic
end

% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created 14-Apr-2021 10:22:20
%
% This script assumes these variables are defined:
%
%   input - input data.
%   target - target data.

x = input_monit;
t = target_monit;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 20;
net = fitnet(hiddenLayerSize,trainFcn);

% vypnuti GUI nntools
net.trainParam.showWindow = 0;

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};

% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivision
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean Squared Error
```

```
% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
    'plotregression', 'plotfit'};

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y);

% Recalculate Training, Validation and Test Performance
trainTargets = t .* tr.trainMask{1};
valTargets = t .* tr.valMask{1};
testTargets = t .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,y);
valPerformance = perform(net,valTargets,y);
testPerformance = perform(net,testTargets,y);

% View the Network
%view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)

% Deployment
% Change the (false) values to (true) to enable the following code
blocks.
% See the help for each generation function for more information.
if (false)
    % Generate MATLAB function for neural network for application
    % deployment in MATLAB scripts or with MATLAB Compiler and Builder
    % tools, or simply to examine the calculations your trained neural
    % network performs.
    genFunction(net,'myNeuralNetworkFunction');
    y = myNeuralNetworkFunction(x);
end
if (false)
    % Generate a matrix-only MATLAB function for neural network code
    % generation with MATLAB Coder tools.
    genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
    y = myNeuralNetworkFunction(x);
end
if (false)
    % Generate a Simulink diagram for simulation or deployment with.
    % Simulink Coder tools.
    gensim(net);
end

predikce=rok_2019(:,k+1:k+8);
output=sim(net,predikce);
error=abs((KNTB_2019(:,k+1:k+8)-output))/KNTB_2019(:,k+1:k+8)*100;
```

```
k

skutecnost_matrix(k-1344,:)=KNTB_2019(:,k+1:k+8);
output_matrix(k-1344,:)=output;
error_avarage(k-1344,:)=error;
error_matrix(k-1344,:)=abs((KNTB_2019(:,k+1:k+8)-
output))./KNTB_2019(:,k+1:k+8)*100;

%vypocet casu jedne operace
if mod(k,1000) == 0 || k == 1 || k == 2
    elapsed_time(k,:) = toc;
end

end
```