Univerzita Pardubice

Fakulta ekonomicko-správní

Predikce nemocnosti zaměstnanců Škoda Auto pomocí strojového učení

Bc. Dominik Novotný

Diplomová práce

2019

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Jičíně dne 24. 04. 2019

……..

Dominik Novotný

Poděkování:

Tímto bych rád poděkoval Bc. Jaroslavu Rysovi za ochotu při poskytnutí a přípravu dat pro experimentální část a doc. Ing. Petru Hájkovi Ph.D. za cenné rady a konzultace při vypracování této diplomové práce.

ANOTACE

Diplomová práce se zabývá popisem používaných metod strojového učení pro regresní úlohy a následnou aplikací těchto metod na úloze zaměřené na predikci nemocnosti zaměstnanců firmy ŠKODA AUTO, a.s. Použité metody predikce zahrnují zejména neuronové sítě, podpůrnou vektorovou regresi a XGBoost. Výsledky ukazují, že pomocí těchto metod lze predikovat nemocnost zaměstnanců na jednotlivých odděleních s nízkou chybou. Výsledky všech použitých metod jsou vyhodnoceny z hlediska chyby a vysvětleného rozptylu. Ukazuje se také, že navržený predikční model výkonností překonává současně používaný přístup a může tak značně přispět ke zvýšení efektivnosti řízení lidských zdrojů v oblasti personálního plánování.

KLÍČOVÁ SLOVA

Strojové učení, XGBoost, Python, Scikit-learn, Neuronové sítě, nemocnost zaměstnanců

TITLE

ŠKODA AUTO employee sickness prediction using machine learning

ANNOTATION

This thesis deals with description of commonly used machine learning methods for regression tasks followed by their application on a prediction task focused on ŠKODA AUTO, a.s. employee sickness prediction. The machine learning methods used for this task mainly include neural networks, support vector regression and XGBoost. The results show that it is possible to predict employee sickness in individual departments with a low error. The results of the used methods are compared in terms of error and variance explained. It is showed that the proposed prediction model outperforms the currently used approach and, thus, it is expected to improve the effectiveness of human resource management in personal planning.

KEYWORDS

Machine learning, XGBoost, Python, Scikit-learn, neural networks, employee sickness

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

CSV – comma separated values (hodnoty oddělené čárkami)

CV – cross validation (křížová validace)

DART – Dropouts meet Multiple Additive Regression Trees

L1 – regularizační metoda Lasso

L2 – regularizační metoda Ridge

MAE – mean absolute error (průměrná absolutní chyba)

MLP – multi layer perceptron (vícevrstvý perceptron)

R2 – vysvětlený rozptyl

RMSE – root means squared error (chyba průměrného čtverce)

RSS – residual sum of squares (reziduální součet čtverců)

SVM – support vector machine (podpůrný vektorový stroj)

SVR – support vector regression (podpůrná vektorová regrese)

ÚVOD

Schopnost predikovat absenci na pracovišti je pro výrobní firmy velmi užitečná zejména při plánování směn výrobních dělníků. Neplánovaná absence ve větším měřítku je schopná narušit provoz na výrobní lince a často se musí řešit využitím externích pracovníků, kteří pro firmu představují dodatečné náklady. Pokud by firma mohla mít nástroj, který by dokázal s dostatečnou přesností predikovat počet absentovaných zaměstnanců na dostatečně dlouhou dobu dopředu, mohla by se pak na takové krizové situace dopředu připravit. Tato práce se zaměřuje na otestování možnosti predikce absence zaměstnanců z důvodu nemocnosti ve firmě ŠKODA AUTO, a.s. v horizontu až tří měsíců dopředu.

K řešenému problému lze přistupovat buď jako ke klasifikačnímu problému, kdy třída 0 představuje přítomnost zaměstnance a třída 1 jeho absenci z důvodu nemoci. Predikovaná hodnota pak představuje pravděpodobnost, že zaměstnanec na daný den bude z důvodu nemoci chybět. Druhá možnost je agregování zaměstnanců do skupin, kde skupina představuje nějaký logický celek, například oddělení. V takovém případě se jedná o regresní úlohu, kdy predikovaná proměnná představuje ať už absolutní, či relativní počet nemocných pro daný den. Tato práce se zaměřuje na druhý typ přístupu k této úloze, což je dáno vysokým počtem zaměstnanců ve výrobních odděleních ve firmě ŠKODA AUTO, a.s. a charakterem výrobního odvětví, kde je snadné nahradit jednotlivce.

Predikce nemocnosti zaměstnanců je složitý problém a je ovlivněna řadou faktorů, např. obdobím roku, předchozí nemocností zaměstnanců, jejich finanční situací, atd. Běžně používané statistické regresní metody neumožňují modelování takto složitých a často také nelineárních vztahů mezi těmito faktory a budoucí nemocností. Proto jsou v této práci použity současné metody strojového učení. Cílem navrženého modelu je navrhnout model založený na strojovém učení k dosažení přesné predikce nemocnosti zaměstnanců.

Cílem práce tedy je charakterizovat vhodné metody strojového učení, shrnout současné přístupy k predikci nemocnosti zaměstnanců, charakterizovat a předzpracovat data o nemocnosti zaměstnanců ve firmě ŠKODA AUTO, a.s., navrhnout model predikce

nemocnosti zaměstnanců a použít jej na datech této firmy. Výsledky budou zhodnoceny z hlediska přesnosti predikce (chyby a vysvětleného rozptylu) a porovnány se současně používanou metodou predikce.

Práce je rozdělena na tři části. Nejprve jsou představeny teoretické základy, tj. přehled použitých metod strojového učení a současné přístupy používané k predikci nemocnosti zaměstnanců. Dále jsou charakterizována použitá data a jejich předzpracování. Nakonec je představen navržený predikční model a prezentovány výsledky dosažené pomocí tohoto modelu, včetně jejich zhodnocení v porovnání se současným stavem řešení a důležitosti použitých vstupních proměnných.

1 TEORETICKÉ ZÁKLADY

1.1 Strojové učení

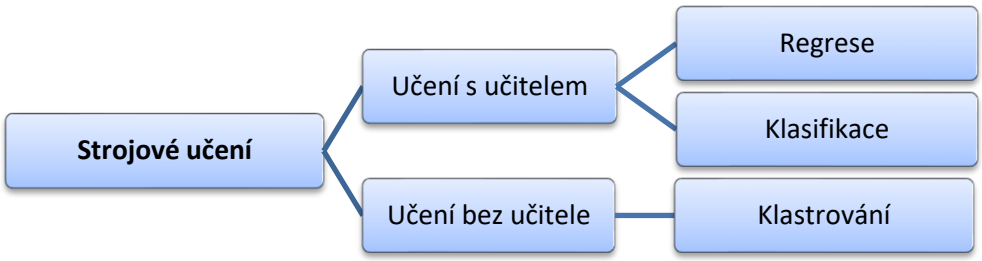
Strojové učení je technika analýzy dat, která učí počítače učit se ze zkušeností tak, jak to umí lidé a zvířata. Algoritmy strojového učení využívají výpočetní metody k tomu, aby získaly informace (naučily se je) přímo z dat, aniž by se spoléhaly na předem stanovenou rovnici jako model. Algoritmy adaptivně zlepšují svůj výkon, protože počet vzorků dostupných pro učení se zvyšuje.

S rostoucím počtem zpracovávaných dat (dnes označované jako „Big data“ – velká data) se strojové učení stalo klíčovou technikou pro řešení problémů v oblastech, jako jsou [1]:

- ✓ Počítačová finanční matematika, pro úvěrové hodnocení a algoritmické obchodování;
- ✓ Zpracování obrazu a počítačové vidění, rozpoznávání obličeje, detekce pohybu a detekce objektů;
- ✓ Výpočetní biologie, pro detekci nádoru, objev léků a řazení DNA;
- ✓ Výroba energie, prognózování cen a zatížení;
- ✓ Automobilový, letecký a výrobní průmysl, pro prediktivní údržbu;
- ✓ Zpracování přirozeného jazyka, pro aplikace rozpoznávání hlasu.

Strojové učení je jednou z nejrychleji se rozvíjejících oblastí počítačové vědy a to z dobrých důvodů: modely prediktivního strojového učení, které jsou naučeny na stále rostoucím počtu datových souborů, poskytují relevantní informace vědcům a podnikům na podporu jejich rozhodování, stejně jako umožňují vytvářet inteligentní aplikace pro běžné spotřebitele. Strukturovaná predikce se pak týká modelů strojového učení, které předpovídají relační informace, které mají strukturu a jsou složené z více vzájemně propojených částí. Tyto modely jsou například používány k předpovídání slov ve větách přirozeného jazyka nebo segmentování obrazu. Strukturované predikční modely jsou důležité v mnoha aplikačních doménách a byly použity s velkým úspěchem v biologii, počítačovém vidění a zpracování přirozeného jazyka [2].

Strojové učení využívá dva typy technik: **učení s učitelem**, který učí model na známých vstupních a výstupních datech, takže může předvídat budoucí výstupy a **učení bez učitele**, které nachází skryté vzory nebo vnitřní struktury ve vstupních datech. Hlavní technikou učení bez učitele je klastrování. Základními úlohami učení s učitelem je regrese a klasifikace – viz následující obrázek 1 [1]. Následující text se pak zaměřuje především na vybrané regresní metody, tedy strojového učení s učitelem, neboť navržený model využívá historická data o nemocnosti zaměstnanců (predikovaný výstup je tedy k dispozici) a výstupní proměnná je číselnou hodnotou počtu nemocných zaměstnanců na oddělení.



Obrázek 1: Techniky strojového učení [1]

1.2 Regresní metody strojového učení

Strojové učení při využití techniky učení s učitelem je učení modelu, který je charakterizován vstupní proměnnou *x*, výstupní proměnnou *y* a algoritmem mapujícím vstup na výstup. To znamená, že $y = f(x)$. Základním cílem učení s učitelem je aproximovat mapovací funkci $y = f(x)$ tak dobře, že když se objeví nová vstupní data *x*, pak lze předpovědět odpovídající výstupní proměnnou *y*.

Tento přístup se nazývá učením s učitelem, protože proces učení na známé hodnotě výstupní proměnné lze považovat za učitele, který dohlíží na celý proces učení. Učící algoritmus iterativně provádí předpovědi na trénovacích datech a je korigován „učitelem“ a učení se zastaví, když algoritmus dosáhne přijatelné úrovně výkonu (nebo požadované přesnosti).

Učení bez učitele na rozdíl od učení s učitelem je metoda, při které se využívají pouze vstupní data *x* a neexistuje tam žádná odpovídající výstupní proměnná. Cílem učení bez učitele je modelovat distribuci v datech, aby se uživatel dozvěděl více o struktuře dat.

Je tomu tak, protože neexistuje správná odpověď a učitel tedy neexistuje (na rozdíl od učení s učitelem). Algoritmy jsou ponechány, aby objevily a prezentovaly strukturu dat [3].

Pokud je cílem použít nějakou metodu strojového učení, tak existuje více možností, mezi kterými je nutné zvolit tu nejvhodnější. Zároveň každý přístup má určité výhody a nevýhody a není možné říci, že je některý nejlepší a některý nejhorší pro všechny typy úloh. Je vždy nutné zvolit podle dané situace tu nejvhodnější metodu pro daný případ podle na základě vlastností (výhod a nevýhod) metod strojového učení [4].

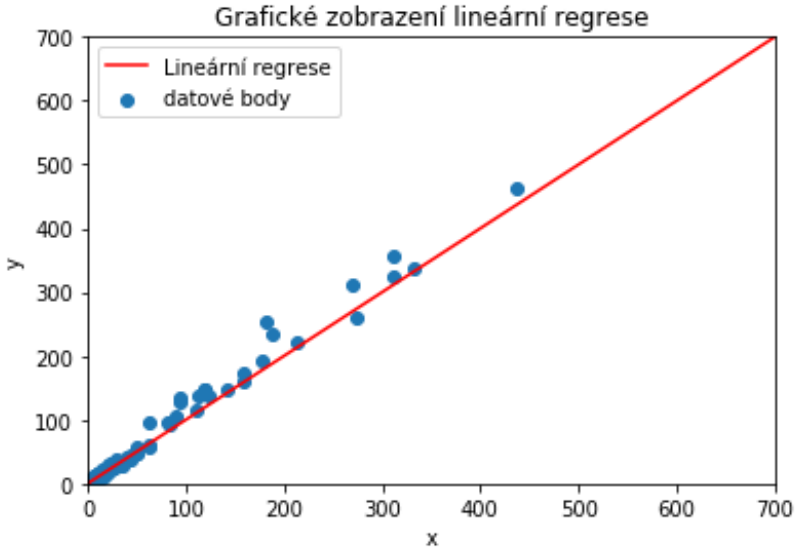
1.2.1 Regresní metody

Jednoduchá lineární regrese

Jednoduchá lineární regrese je jednou z nejběžnějších typů regresních metod. Při jejím použití se předpovídá cílová proměnná y na základě vstupní proměnné (prediktoru) x. Mezi cílovou proměnnou a prediktorem by měl existovat lineární vztah, který je nazýván lineární regresí.

Pokud se uvažuje např. předvídání mzdy zaměstnanců na základě jejich věku, lze snadno zjistit, že existuje korelace mezi věkem zaměstnance a jeho mzdou. Hypotéza jednoduché lineární regrese je: $y = a + bx$, kde y reprezentuje mzdu a x reprezentuje věk zaměstnance. Aby bylo možné predikovat mzdu zaměstnance, je pak nutné znát hodnoty a a b – koeficienty regresního modelu.

Při tvorbě regresního modelu jsou základem uvedené dva koeficienty a a b, které jsou zjištěny z trénovacích dat. Cílem strojového učení (učení s učitelem) je najít hodnoty těchto parametrů, které nejlépe vyhovují trénovacím datům tak, aby byla minimalizována nákladová funkce. Nákladová funkce reprezentuje chybu modelu. Během učení je cílem minimalizovat chybu mezi skutečnými a předpokládanými hodnotami a tím minimalizovat nákladovou funkci. Na obrázku 2 jsou modré body trénovací data a červená čára je regresní model naučený na trénovacích datech.



Obrázek 2: Grafické zobrazení příkladu jednoduché lineární regrese [vlastní]

Vícenásobná lineární regrese

Vícenásobná lineární regrese je takový model, do kterého je zapojena lineární kombinace více vstupních proměnných $x_1, \ldots, x_D$ podle vzorce:

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D \,, \tag{1}$$

kde $x = (x_1, \ldots, x_D)^T$. Tento zápis je známý jako vícenásobná lineární regrese. Klíčovou vlastností tohoto modelu je, že se jedná o lineární funkci parametrů w_0, ..., w_D. Tento model je však také lineární funkcí vstupních proměnných x_j, $j = 1, 2, \ldots, D$, což znamená významná omezení tohoto modelu. Proto se rozšiřuje třída modelů o lineární kombinace fixních nelineárních funkcí vstupních proměnných ve tvaru [5]:

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\boldsymbol{x}) \tag{2}$$

kde $\phi_j(\boldsymbol{x})$ jsou známy jako základní funkce. Celkový počet parametrů v tomto modelu je M.

Parametr w_0 umožňuje libovolný pevný posun v datech a někdy se nazývá parametr zkreslení (nesmí být zaměňován za "zkreslení" ve statistickém smyslu). Často je vhodné definovat dodatečnou figurální "základní funkci" $\phi_0(\boldsymbol{x}) = 1$, což znamená, že model pak vypadá takto [5]:

$$y(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w^T \phi}(\boldsymbol{x}), \tag{3}$$

kde $w = (w_0, \ldots, w_{M-1})^T$ a $\phi = (\phi_0, ..., \phi_{M-1})^T$. V praxi při rozpoznávání vzorů se pak používá nějaká forma přezpracování nebo extrakce prvků z původních proměnných. Pokud původní proměnné obsahují vektor $\boldsymbol{x}$, pak mohou být vlastnosti vyjádřeny pomocí základních funkcí $\{\phi_j(\boldsymbol{x})\}$.

Použitím nelineárních základních funkcí je dovoleno, aby funkce $y(\boldsymbol{x}, \boldsymbol{w})$ byla nelineární funkcí vstupního vektoru $\boldsymbol{x}$. Funkce podle rovnice (2) se nazývají lineární modely, protože tato funkce je lineární podle $\boldsymbol{w}$. Právě tato linearita v parametrech výrazně zjednodušuje analýzu této třídy modelů [5].

U vícenásobné regrese je vhodné, aby proměnné zařazené do vícenásobného vztahu byly na sobě nezávislé, nebo slabě závislé. Lineární závislost mezi vysvětlujícími proměnnými je označována jako multikolinearita.

Multikolinearita způsobuje, že odhady regresních koeficientů pak nemusí být přesné, především pokud se jedná o tzv. „škodlivou multikolinearitu“. Je označována jako škodlivá z toho důvodu, že některé párové korelační koeficienty mezi vysvětlujícími proměnnými jsou větší než určitá hraniční hodnota. Zjistí-li se multikolinearita, může to znamenat, že není vhodné zařadit takto silně korelované vysvětlující proměnné společně do vícenásobného regresního vztahu. Pro zjištění, kterou z dvojice silně korelovaných regresorů z regresní rovnice vypustit, tak pro to existují speciální parametry. S tím souvisí postup volby vhodné podmnožiny proměnných, mezi které patří [6]:

- Metoda postupného přidávání vysvětlujících proměnných;
- Metoda postupného vyřazování proměnných;
- Metoda stupňovité regrese.

Postup je pak takový, že se nejdříve do vícenásobného vztahu zařadí ta vysvětlující proměnná x_j, která má největší párový korelační koeficient se závisle proměnnou y. Následně se zařadí proměnná x_k, která má vysoký párový korelační koeficient se závisle proměnnou y, ale současně není silně závislá s již zařazenou proměnnou x_j. V třetím kroku se při zařazení dalších proměnných do vícenásobného vztahu ověřuje, zda nejsou silně závislé se všemi již zařazenými proměnnými [6].

Metoda nejmenších čtverců

Model lineární regrese předpokládá, že regresní funkce je lineární ve svých vstupech x_1, ..., x_D. Lineární modely byly do značné míry vyvíjeny v předpočítačovém věku statistik, ale i v dnešní počítačové éře jsou stále dobré důvody k jejich studiu a používání. Jsou jednoduché a často poskytují odpovídající a interpretovatelný popis toho, jak vstupy ovlivňují výstup. Pro účely predikce mohou někdy překonat lepší nelineární modely, zejména v situacích s malým počtem tréninkových případů, nízkým poměrem signálu k šumu nebo s řídkými daty. Lineární metody mohou být aplikovány na transformaci vstupů, což značně rozšiřuje jejich rozsah.

V případě, že se uvažuje jako vstup vektor $\boldsymbol{x}^{\mathrm{T}} = (x_1, x_2, \ldots, x_D)$, a je cílem předpovědět reálné výstupy, tak lineární regresní model má podobu:

$$f(x) = \beta_0 + \sum_{i=1}^{D} x_j \beta_j. \qquad (4)$$

Tento lineární model předpokládá, že regresní funkce je lineární, nebo že je přiměřeně podobná lineární funkci. V tomto modelu jsou β_j neznámé parametry nebo koeficienty a proměnné x_j mohou pocházet z různých zdrojů, jako jsou např. [7]:

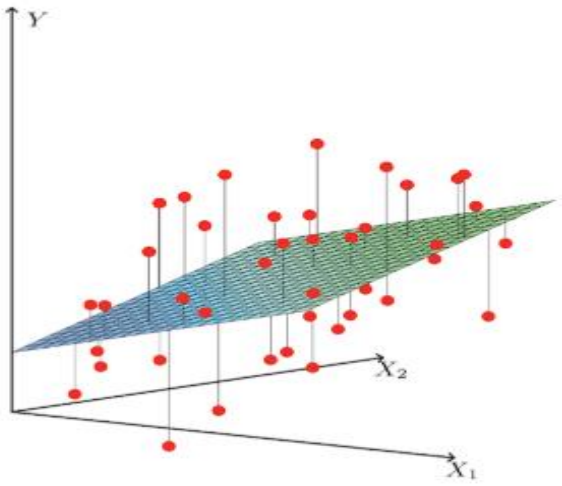
- kvantitativní vstupy;
- transformace kvantitativních vstupů, jako je logaritmus, mocnina nebo odmocnina;
- základní rozšíření, jako je $x_2 = x_1^2$, $x_3 = x_1^3$, což vede k polynomiální reprezentaci;
- interakce mezi proměnnými, například $x_3 = x_1 \times x_2$.

Bez ohledu na zdroj x_j je model v parametrech lineární.

Typicky se uvažuje sada trénovacích dat (x_1, y_1) ... (x_N, y_N), ze kterých lze odhadnout parametry β_j. Každý $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$ je vektor měření vlastností pro i-tý případ. Nejpopulárnější metoda odhadu je metoda nejmenších čtverců, ve kterých se použijí koeficienty $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_D)^T$ pro minimalizaci zbytkového součtu čtverců (RSS = Residual Sum of Squares) [7]:

$$\begin{aligned} \text{RSS}(\beta) &= \sum_{i=1}^{N}(y_i - f(x_i))^2 \\ &= \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{D} x_{ij}\beta_j))^2 \end{aligned} \qquad (5)$$

Ze statistického hlediska je toto kritérium přijatelné, pokud trénovací pozorování (x_i, y_i) představují nezávislé náhodné hodnoty z množiny dat. I když x_i nejsou vybrány náhodně, tak kritérium je stále platné, pokud jsou y_i podmíněně nezávislé vzhledem ke vstupům x_i. Obrázek 3 znázorňuje geometrii tvarování nejmenších čtverců v IR^{D+1}-rozměrném prostoru obsazeném dvojicemi (X, Y).



Obrázek 3: Grafické zobrazení použití metody nejmenších čtverců [7]

Na obrázku je vidět, že tato metoda neudává žádné předpoklady o platnosti modelu, ale jednoduše najde nejlepší lineární přizpůsobení vyhodnocovaným datům. Realizace metody nejmenších čtverců je realizována intuitivně uspokojivě bez ohledu na to, jak vznikají data; kritérium měří průměrný nedostatek způsobilosti.

Tento proces se zapisuje tak, že se označí pomocí X matice $N\times(D+1)$ každý řádek vstupního vektoru, a podobně se označí pomocí Y N-rozměrný vektor výstupů v trénovací množině dat. Pak je možné zapsat zbytkový součet čtverců jako [7]:

$$\text{RSS}(\beta) = (Y - X\beta)^T(Y - X\beta). \qquad (6)$$

Jedná se o kvadratickou funkci v parametrech D + 1. Pokud se rozliší tvar s ohledem na β, tak lze získat tvar [7]:

$$\frac{\delta RSS}{\delta \beta} = (Y - X\beta)^T (Y - X\beta)$$

$$\frac{\delta^2 RSS}{\delta \beta \delta \beta^T} = 2X^T X \tag{7}$$

Za předpokladu, že X má plný sloupec, a tedy $X^{\mathrm{T}}X$ je jednoznačné, tak se uvažuje první derivace rovna nule:

$$X^{\mathrm{T}}(Y - X\beta) = 0, \tag{8}$$

aby bylo získáno unikátní řešení:

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{9}$$

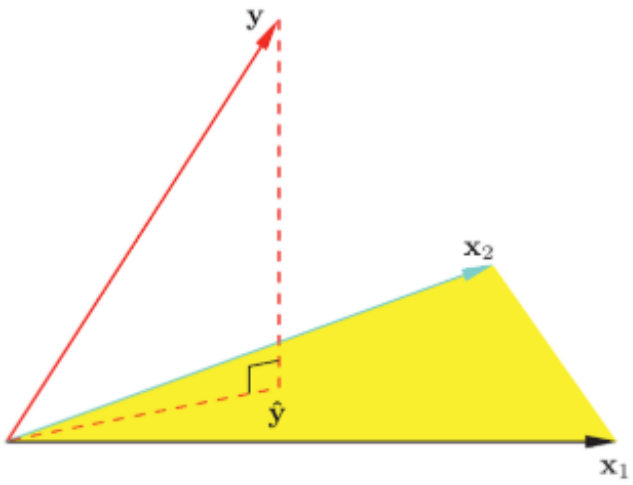
Predikované hodnoty vstupu vektoru $\boldsymbol{x}_0$ jsou pak dány funkcí:

$$\hat{f}(\boldsymbol{x}_0) = (1 : \boldsymbol{x}_0)^T \hat{\beta}, \tag{10}$$

správná hodící se data pak jsou:

$$\hat{y} = X\hat{\beta} = X(X^T X)^{-1} X^T Y, \tag{11}$$

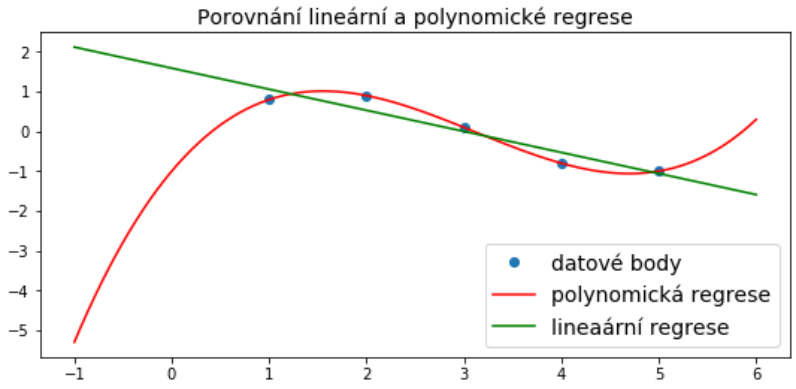
kde $\hat{y}_i = \hat{f}(\boldsymbol{x}_i)$. Na následujícím obrázku 4 je pak znázorněna N-rozměrná geometrie regresní metody nejmenších čtverců se dvěma prediktory. Výsledný vektor $\boldsymbol{y}$ je ortogonálně promítnut do prostoru vymezeného vstupními vektory $\boldsymbol{x}_1$ a $\boldsymbol{x}_2$. Projekce $\boldsymbol{y}$ pak představuje vektor předpovědí nejmenších čtverců [7].



Obrázek 4: Zobrazení projekce vektoru *y* pomocí metody nejmenších čtverců [7]

1.2.2 Polynomická regrese

Lineární regrese vyžaduje, aby vztah mezi závislou proměnnou a nezávislou proměnnou byl lineární. Ale jaký model použít, pokud rozložení dat je složitější, jako na následujícím obrázku? Právě to je možné řešit pomocí polynomické regrese, která lépe zachytí takováto data [8].

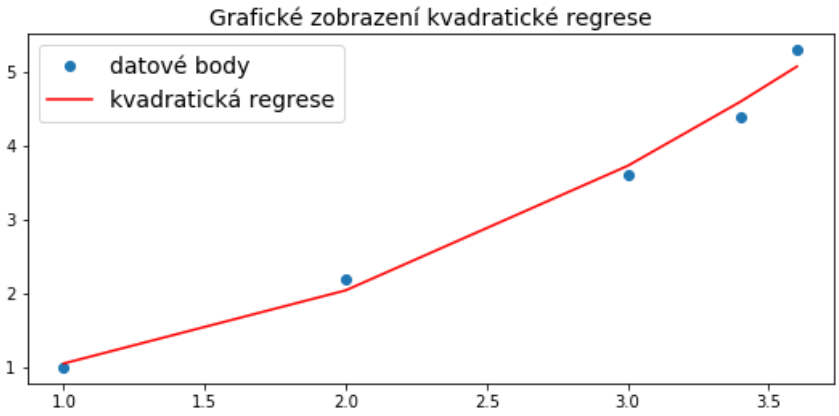


Obrázek 5: Příklad souboru dat, kde není vhodná lineární regrese [vlastní]

Vidíme, že přímka nemůže zachytit vzory v datech. Toto je typický příklad nevhodného použití lineární regrese. Aby bylo možné trend zachytit pomocí křivky, tak je potřeba zvýšit složitost modelu, použít vyšší řád rovnice [8].

V polynomické regresi se transformují původní proměnné do polynomických proměnných daného stupně a pak se aplikuje lineární regrese. V případě polynomu druhého stupně je pak křivka grafického vyjádření kvadratická – viz obrázek 6. Základní rovnice polynomické regrese pak vypadá takto:

$$Y = a + bX + cX^2. \tag{12}$$



Obrázek 6: Grafické zobrazení polynomické regrese [vlastní]

Polynomická regrese se používá pro takové vztahy, kde je pravděpodobně vztah mezi proměnnými kvadratický. Předpokladem v obvyklé vícenásobné lineární regresní analýze je, že všechny nezávislé proměnné jsou nezávislé. V modelu polynomické regrese tento předpoklad není splněn. Polynomická regrese se používá především v těchto případech definování nebo popisu nelineárního jevu, jako je např. [8]:

- Rychlost růstu;
- Průběh epidemií nemocí;
- Distribuce uhlíkových izotopů v jezerních sedimentech.

Základním cílem regresní analýzy je modelovat očekávanou hodnotu závislé proměnné *y* z hlediska hodnoty nezávislé proměnné *x*.

Výhody použití polynomické regrese [9]

- ✓ Široká škála funkcí, které může tato metoda zahrnout.
- ✓ Polynomická křivka v podstatě vyhovuje širokému rozsahu zakřivení.
- ✓ Polynomická křivka poskytuje nejlepší aproximaci vztahu mezi závislou a nezávislou proměnnou.

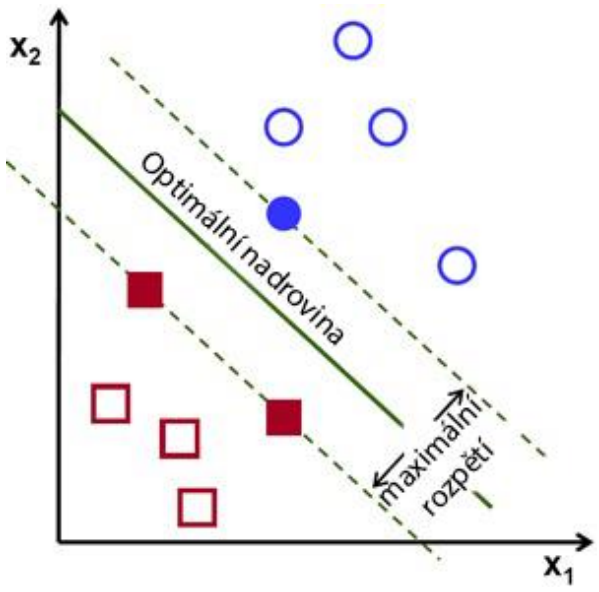
Nevýhody použití polynomické regrese

- x Velmi citlivá metoda na hodnoty umístěné mimo trend.
- x Polynomické funkce vyššího řádu jsou těžko interpretovatelné a způsobují přeučení [9].
- x Přítomnost jednoho nebo dvou hodnot mimo trend v datech může vážně ovlivnit výsledky nelineární analýzy.
- x Je k dispozici mnohem méně možných metod pro kontrolu hodnot ležících mimo trend, než je k dispozici u lineární regrese [10].

1.2.3 Regrese pomocí podpůrných vektorových strojů

Analýza podpůrných vektorových strojů (SVM) je nástrojem strojového učení pro klasifikaci a regresi, který poprvé identifikoval Vladimír Vapnik a jeho kolegové v roce 1992. Cílem této metody je najít funkci f*(x)*, která se odchyluje od *y* hodnotou ne větší než ε pro každý trénovací bod *x* [11].

Klasifikace pomocí podpůrných vektorových strojů je algoritmus, který slouží pro nalezení optimální dělící nadroviny, která má maximální rozpětí v *N*-dimenzionálním prostoru, který zřetelně klasifikuje datové body. Tato metoda slouží pro nalezení lineární rozhodovací plochy v případě nelineárního mapování vstupních vektorů do vícerozměrného prostoru příznaků (proměnných). Datové body, které spadají na obě strany zobrazené optimální nadroviny mohou být přiřazeny různým třídám. Rozměr optimální nadroviny také závisí na počtu funkcí. Pokud je počet vstupních funkcí 2, pak je optimální nadrovina definována pouze jednou funkcí (viz obrázek 7). Pokud je počet vstupních funkcí 3 a více, pak se z optimální nadrovina stane dvourozměrná rovina [12].



Obrázek 7: Zobrazení klasifikace pomocí podpůrných vektorových strojů [12]

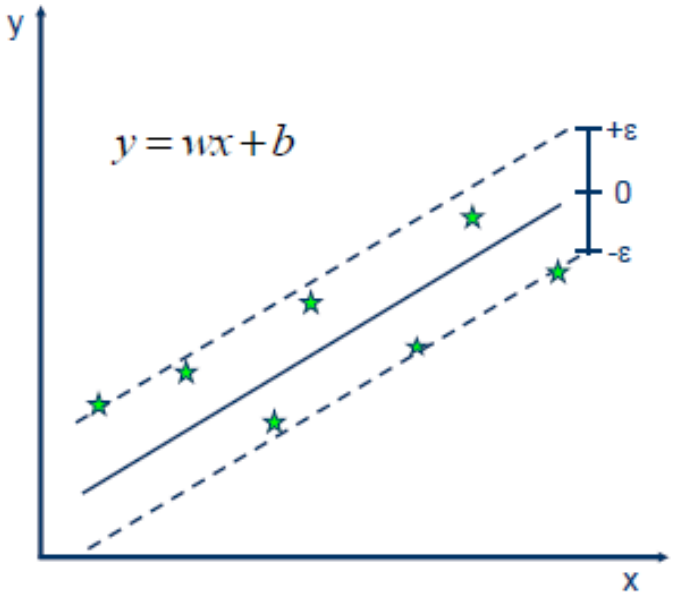
Regrese pomocí podpůrných vektorových strojů (SVR – Support Vector Regression) je regresní metoda, která zachovává všechny hlavní vlastnosti, které charakterizují algoritmus SVM. Použitím regrese pomocí podpůrných vektorových strojů je v aproximaci SVM stanovena hranice tolerance (epsilon), která je při řešení daného problému vyžadována. Hlavní myšlenka je však vždy stejná: minimalizovat chybu

a definovat optimální funkci nadroviny (hyperplane), která maximalizuje hranice (maximální rozmezí), přičemž je určitá chyba tolerována.

Lineární SVR

Primární tvar pro podpůrnou vektorovou regresi

Pokud je k dispozici sada trénovacích dat, kde x je mnohorozměrný soubor N pozorování s pozorovanými hodnotami odezvy y, tak je pak cílem najít lineární funkci $y = wx + b$ (obrázek 8) s tím, že je nutné se ujistit, že je tato funkce co nejrovnoměrnější s minimální normovanou hodnotou $\|w\|$. Takto charakterizovaný cíl je označován jako konvexní optimalizační problém [13].



Obrázek 8: Grafické zobrazení maximálního rozmezí při využití podpůrné vektorové regrese [14]

Pokud všechny uvažované hodnoty spadají do daného maximálního rozmezí, tak má rovnice pro optimalizaci takovýto tvar [14]:

$$min\frac{1}{2}\|w\|^2. \tag{13}$$

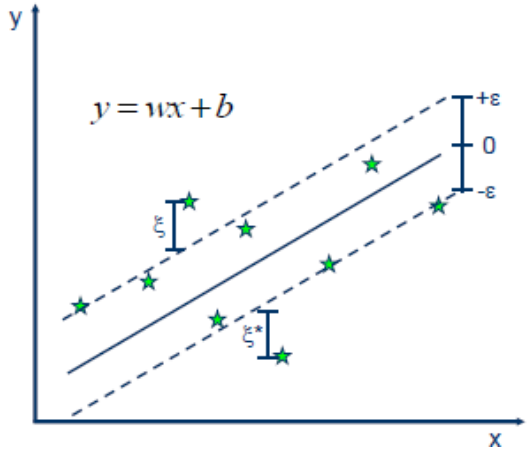
Omezení jsou pak vymezena následujícími rovnicemi:

$$y_i - wx_i - b \leq \varepsilon,$$

$$wx_i - b - y_i \leq \varepsilon,$$

$$\xi_i, \xi_i^* \geq 0. \tag{14}$$

Je však možné, že žádná taková funkce f(x) neexistuje pro splnění těchto omezení pro všechny body. V takovém případě je potřeba se vypořádat s nezařaditelnými hodnotami a zavést pro každý takový bod proměnné označení neudržitelnými omezeními, zavést pro každý bod proměnné ξ_i až ξ_i^*. Tento přístup je podobný konceptu „soft margin“ (měkkého rozpětí) v klasifikaci pomocí SVM, protože proměnné slack ξ_i umožňují existenci chyb v regresi až do hodnoty ξ a ξ^* (obrázek 9), přesto však splňují požadované podmínky.



Obrázek 9: Grafické zobrazení podpůrné vektorové regrese v případě tolerance chyby [14]

Pro popis funkce, která je vyjádřena hodnotami zobrazenými hvězdičkami na uvedeném obrázku 9 se použije hodnota vyjádřená následující funkcí, která by měla být co nejnižší (minimalizovaná) [14]:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*). \tag{15}$$

Tento tvar je někdy označován jako primární formule (z angl. primal formula). Hranice jsou pak vymezeny následujícími rovnicemi [14]:

$$y_i - wx_i - b \leq \varepsilon + \xi_i$$

$$wx_i - b - y_i \leq \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0 \tag{16}$$

Konstanta C je omezení - kladná číselná hodnota, která řídí velikost snížení vypovídací hodnoty uvažovaných pozorování, které leží mimo omezení epsilon (ε).

Tato hodnota pomáhá předcházet zahrnutí příliš vybočujících hodnot do modelu a určuje vztah mezi linearitou f(x) a úrovní, do jaké jsou tolerovány odchylky větší než ε [13].

Duální tvar pro podpůrnou vektorovou regresi

Výše popsaný problém optimalizace je výpočtově jednodušší řešit v jeho Lagrangeově duální formulaci. Řešení duálního problému poskytuje nižší hranici řešení primárního (minimalizačního) problému. Optimální hodnoty primárních a duálních problémů nemusí být stejné a tento rozdíl se nazývá „mezera duality“ (z angl. duality gap). Když je však problém konvexní a splňuje podmínku omezení, je hodnota optimálního řešení primárního problému dána řešení duálního problému daného rovnicí:

Vzorec duální formule se pak získá tak, že se vytvoří Lagrangova funkce z primární funkce zavedením nezáporných násobků α a α^* pro každé pozorování x. Z toho pak vyplyne vzorec duální tvar, kde se minimalizuje [13]:

$$L(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*)\left(\alpha_j - \alpha_j^*\right)x_i^{'}x_j + \varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) \quad (17)$$

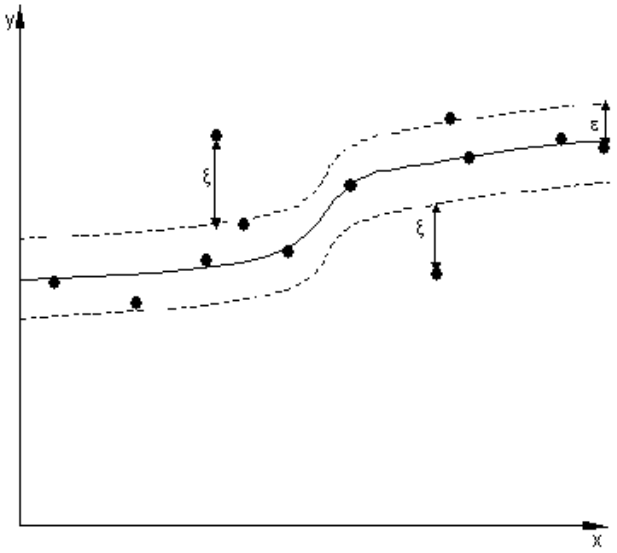
Maximální rozmezí je pak definováno pomocí funkce:

$$\sum_{n=1}^{N}(\alpha_n - \alpha_n^*) = 0, \text{ kde} \qquad \forall n: 0 \le \alpha_n \le C$$

$$\forall n: 0 \le \alpha_n^* \le C \quad (18)$$

Nelineární regrese

Na následujícím obrázku 10 je vidět podobný případ zobrazení pro nelineární podobu regresní úlohy [15]. Místo lineární funkce je v tomto případě použita polynomická funkce s cílem dosažení vyšší přesnosti proložení dat a přesnějšího tvaru maximálního rozmezí.



Obrázek 10: Nelineární podoba podpůrné vektorové regrese [15]

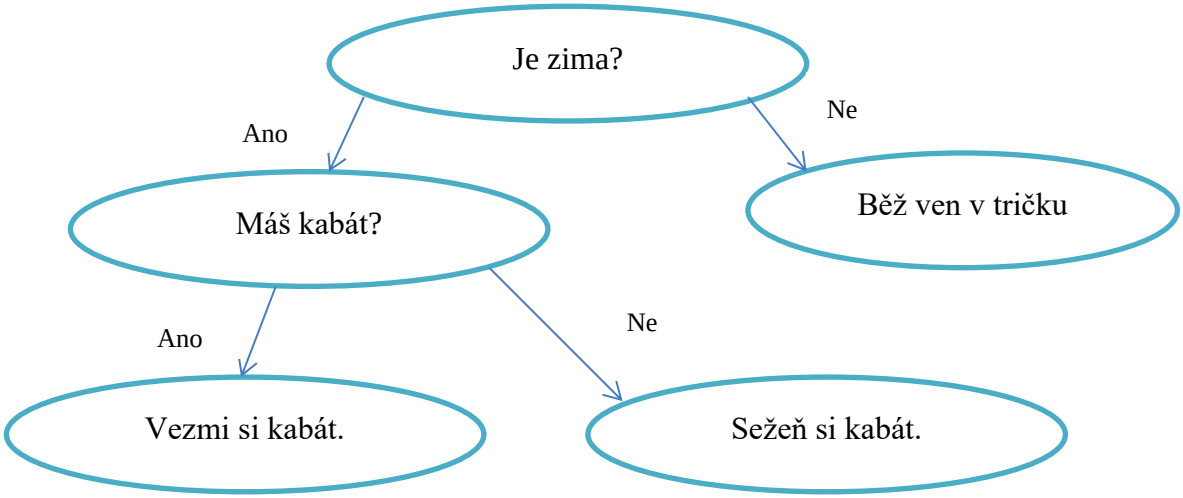
1.2.4 Regresní stromy

Algoritmus rozhodovacích stromů je generován na základě vstupních dat a je používán jak pro klasifikační, tak pro regresní řešení problémů. Je možné jako příklad uvést situaci, že chce jít tenista hrát ven tenis. V takové situaci řeší otázku, zda je v danou chvíli ideální jít ven na tenis, což závisí na různých faktorech, jako je čas, počasí, teplota atd. Tyto faktory jsou nazývány jako proměnné, které ovlivní rozhodnutí tenisty. Rozhodovací proces podle těchto faktorů je možné zapsat formou rozhodovacího stromu, kde se jednotlivé akce větví podle toho, jaká je odpověď na danou otázku např. ohledně počasí, času apod. [16].

Rozhodovací stromy se ve strojovém učení používají pro regresi s využitím stromové reprezentace k vyřešení problému, ve kterém každý uzel odpovídá označení hranice proměnné pro dělení.

Při používání regresních strojů se vychází z předpokladu, že trénovací data se považují za základní kořen rozhodovacího stromu. Kategorické proměnné jsou u regresních stromů preferovány. Pokud jsou hodnoty spojité, pak jsou před vytvořením modelu diskretizovány a rozděleny do kategorií [17]. Na základě hodnot proměnných jsou datové body rekurzivně distribuovány. Pro seřazení proměnných (vlastností) kořenu nebo vnitřních uzlů stromu se využívají statistické metody [16].

Jednoduchým příkladem rozhodovacího stromu může být např. situace zobrazená na následujícím obrázku 11, kdy rozhodovatel řeší, co bude dělat podle parametrů daných otázkami.



Obrázek 11: Příklad rozhodovacího stromu [vlastní]

Jako regresní stromy se označují takové rozhodovací stromy, kde jsou kategorie čísla a ne slovní kategorie [18]. Modely rozhodovacích stromů jsou vytvořeny pomocí 2 kroků: úvodní část a prořezávání. V úvodní části se věnuje čas promyšlení toho, jak se bude strom stavět, tj. nastavují se všechny hierarchické rozhodovací hranice na základě vstupních dat. Vzhledem k povaze rozhodovacích stromů mohou být náchylné k přílišnému rozvětvení. Proto pak nastává proces prořezávání, který slouží k odstranění zbytečné struktury z rozhodovacího stromu, což účinně snižuje složitost modelu a zajišťuje snadnější interpretaci.

Úvodní část (Indukce)

Vytváření rozhodovacích stromů na vysoké úrovni prochází čtyřmi hlavními kroky tvorby [16]:

1. Získání vstupní sady trénovacích dat, která by měla mít některé proměnné funkce a klasifikační nebo regresní výstup.
2. Určení „nejlepší funkce“ v datové sadě pro rozdělení dat.
3. Rozdělení dat do podmnožin, které obsahují možné hodnoty pro tuto nejlepší funkci. Toto rozdělení v podstatě definuje uzel ve stromě, tj. každý uzel je bod rozdělení na základě určitého parametru daného ze vstupních dat.
4. Rekurzivní generování nových stromových uzlů pomocí podmnožiny dat vytvořených z kroku 3. Pokračuje se ve větvení, dokud se nedosáhne bodu, kdy je optimalizována přesnost a zároveň minimalizován počet uzlů.

Pro regresní strom je možné použít jednoduchou čtvercovou chybu jako nákladovou funkci [16]:

$$E = \sum(Y - \hat{Y})^2, \tag{19}$$

kde Y je predikovaná hodnota a $\hat{Y}$ je předpokládaná hodnota. Součtem všech vzorků v datové sadě se zjistí celková chyba.

Pozitiva a negativa použití regresních stromů

Mezi hlavní výhody použití regresních stromů patří [19]:

- ✓ Jsou snadné na pochopení, interpretaci, vizualizaci.
- ✓ Rozhodovací stromy implicitně provádějí monitorování (screening) proměnných nebo výběr kritérií.
- ✓ Mohou zpracovat jak numerická, tak kategorická data. Zvládají problémy s více výstupy.
- ✓ Rozhodovací stromy vyžadují relativně nenáročnou přípravu dat.
- ✓ Nelineární vztahy mezi parametry neovlivňují vypovídací schopnost regresních stromů.

Mezi hlavní nevýhody použití regresních stromů patří [19]:

- x Učící mechanismy s rozhodovacími stromy mohou vytvářet příliš složité stromy, které data dobře nezobecňují.
- x Rozhodovací stromy mohou být nestabilní, protože malé odchylky v datech mohou mít za následek generování zcela jiného stromu.
- x Používané algoritmy nemohou zaručit vytvoření optimálního rozhodovacího stromu. To může být zmírněno učením více stromů, kde jsou datové vzorky vybírány náhodně a nahrazovány.
- x Učící mechanismy rozhodovacích stromů vytvářejí zkreslené stromy, pokud některé třídy dominují. Doporučuje se proto vyvažovat sadu dat před tím, než se vytváří samotný rozhodovací strom.

1.2.5 Regresní lesy

Rozhodovací lesy jsou souborovou technikou, která je schopná provádět jak regresní, tak klasifikační úkoly s použitím více rozhodovacích stromů technikou nazývanou Bootstrap Aggregation, běžně známou jako „bagging“. Bagging v metodě regresních stromů zahrnuje učení každého rozhodovacího stromu na jiném vzorku dat, kde se provádí vzorkování s náhradou. Základní myšlenkou je kombinovat více rozhodovacích stromů při určování konečného výstupu, nikoli spoléhání se na jednotlivé rozhodovací stromy [20].

Rozhodovací lesy v regresních úlohách jsou používány jako neparametrické modely, které provádějí sled jednoduchých testů pro každou instanci, procházejí datovou strukturou binárních stromů, dokud není dosaženo uzlu (rozhodnutí). Tento regresní model se skládá ze souboru rozhodovacích (regresních) stromů, kde každý strom v lese s regresním rozhodnutím má Gaussovo rozdělení pravděpodobnosti jako predikci. Agregace se provádí přes soubor stromů, aby se zjistilo nejbližší Gaussovo rozdělení ke kombinované distribuci pro všechny stromy v modelu [20].

Pokud se uvažuje koncept rozhodovacího stromu a aplikují se na něho principy propojení stromů, tak se využívá vzorkovací technika - tzv. „bootstrappingu“. Při této technice se využívá pouze asi 2/3 dat. Přibližně 1/3 dat (data „out-of-bag“) se v modelu nepoužívá a může být vhodně použita jako testovací sada [21].

Výhody regresních lesů

Rozhodovací stromy mají tyto výhody [20], [21]:

- ✓ Jsou efektivní a rychlé jak při výpočtu, tak při využití paměti během učení a predikce.
- ✓ Mohou představovat nelineární rozhodovací hranice.
- ✓ Provádí integrovaný výběr funkcí a klasifikaci a jsou funkční i v přítomnosti více vlastností.

- ✓ Jsou rychlejší pro učení než rozhodovací stromy, protože v tomto modelu se pracuje pouze na podmnožině funkcí, takže je možné snadno pracovat se stovkami funkcí.
- ✓ Odolnost vůči extrémním a nelineárním datům.
- ✓ Jedná se o model s nízkou odchylkou a mírným rozptylem.

Nevýhody regresních lesů

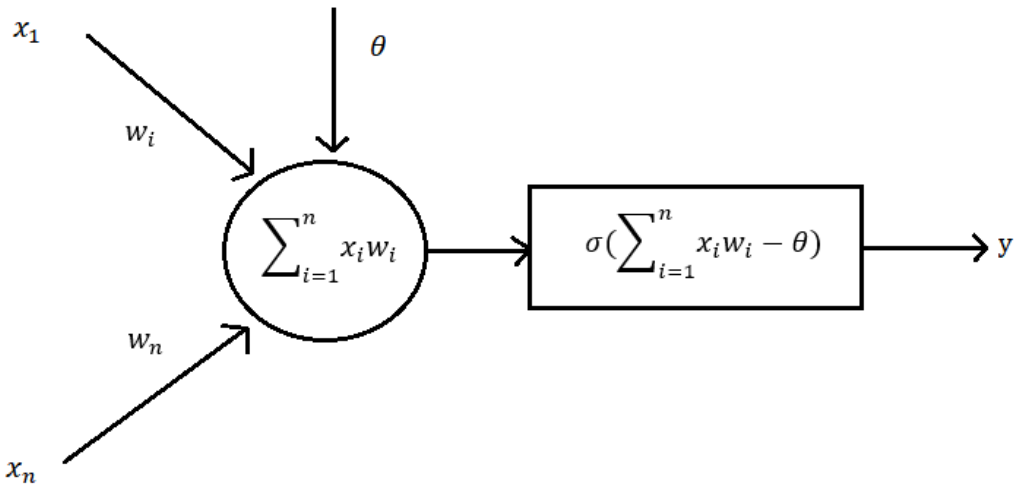
Mezi nevýhody regresních lesů patří [21]:

- x Náhodné modely lesů nejsou všechny interpretovatelné.
- x U příliš velkých datových souborů může velikost stromů zabrat velkou část paměti.
- x Tento model může inklinovat k zahrnování příliš velkého počtu dat, je vhodné nastavit parametry pro omezení.

1.2.6 Neuronové sítě

Ve strojovém učení se také používají přístupy k paralelním výpočtům, které mají za cíl modelovat chování nervové soustavy živočichů, především se jedná o modelování chování nervové soustavy živočichů, nejvíce pak lidského mozku. Lidský mozek obsahuje 10^{11} až 10^{14} neuronů, které jsou uloženy v šedé mozkové kůře, kde jsou synapse realizovány v rozsahu cca 10^4 na jeden neuron. S ohledem na takový rozměr je napodobení lidského mozku velmi obtížně realizovatelné. Přesto se však odborníci pokoušení alespoň o určitou simulaci některých funkcí myšlení, kterou nazývají neuronovými sítěmi. Jejich podstatou je modelování struktury a činnosti biologických neuronových sítí [22].

Fyziologický neuron je pak převeden do matematického modelu neuronu (viz obrázek 12).



Obrázek 12: Matematický model neuronu [vlastní]

Neurony vytvořené na základě tohoto modelu se pak propojují do neuronových sítí a hledají vhodné kombinace sítě, které jsou použitelné pro řešení úloh umělé inteligence [23]. Teoreticky je možné neuron popsat jako systém s mnoha vstupy a jediným výstupem, jak je vidět na uvedeném obrázku výše. Je tedy možné neuron považovat za matematický procesor, do něhož je zadáván n-rozměrný vektor vstupních signálů $x(t) \in R^n$ a ze kterého vystupuje jediný výstupní signál v podobě $y(t) \in R^1$. Vstupní vektor reprezentuje signály přicházející z n sousedních neuronů a činnost

takového matematického procesoru je možné popsat jako operaci zobrazení N ze vstupního prostoru R^n do výstupního prostoru R^1 [24]:

$$N: x(t) \in R^n \rightarrow y(t) \in R^1. \tag{20}$$

Umělé neuronové sítě jsou široce používaným nástrojem pro nelineární modelování, aproximaci funkcí, predikci, klasifikaci a asociaci. Síťová funkce f(x) je definována jako složení dalších funkcí g(x), které lze dále definovat jako složení dalších funkcí. To může být vhodně reprezentováno jako síťová struktura, se šipkami zobrazujícími závislosti mezi proměnnými. Široce používaným typem kompozice je nelineární vážený součet [25]:

$$f(x) = F(\sum_i w_i g_t(x)), \tag{21}$$

kde F se označuje jako aktivační funkce, kterou je nějaká předdefinovaná funkce, jako např. hyperbolická tangenta [25].

Hluboké učení

Hluboké učení (z angl. Deep Learning) je definováno jako metoda strojového učení, jejímž cílem je modelování dat s vysokou úrovní abstrakce. Hluboké učení spočívá v nastavení velmi složitých modelů, jejichž cílem je zajištění lepších výsledků při predikci.

Neuronové sítě patří k metodám hlubokého učení, které představují vysoce parametrické komplexní funkce, u nichž je cílem dosáhnout optimalizace – nalezení nejlepších parametrů, které by vyhovovaly vstupním datům. Zjednodušeně je také možné říci, že neuronové sítě jsou velmi složitou „evolucí“ lineární regrese, která má být schopna modelovat složité struktury dat.

Neuronové sítě je možné použít např. pokud se hledá regresní funkce f taková, že výstupy (y_1, y_2,…, y_m) jsou dobře aproximovány f(x_1, x_2,…, x_n), kde (x_1, x_2,…, x_n) jsou vstupy a cílem je najít funkci těchto vstupů, které dobře vysvětlují pozorované odpovídající výstupy. Myšlenkou modelování neuronových sítí je pak zapomenout na myšlenku nastavit lehce parametrizovanou funkci tvarovanou člověkem a upravenou strojem (prostřednictvím těchto několika parametrů, jako u lineární regrese), ale místo

toho nastavit vysoce parametrizovanou funkci, která bude velmi flexibilní, přestože nebude dávat pro člověka velký smysl, ale která bude funkční při fázi učení [26].

Využití neuronových sítí je pak velmi vhodné u složitějších případů, kde není vhodné použít jednoduchou regresi, což je zobrazeno např. na obrázku 13. Na obrázku jsou vidět datové body, které nelze spojit rostoucí přímkou, přestože intuitivně člověk vidí, že se jedná o dvě téměř horizontální funkce. Proto je v takovémto případě lepší pro lepší charakteristiku použít právě model neuronových sítí. V tomto případě se jedná o použití dvou rovnic – dvou funkcí pro popis analyzovaných dat [26]:

$$Y = m_1 \times x + b_1,$$

$$Y = m_2 \times x + b_2. \tag{22}$$

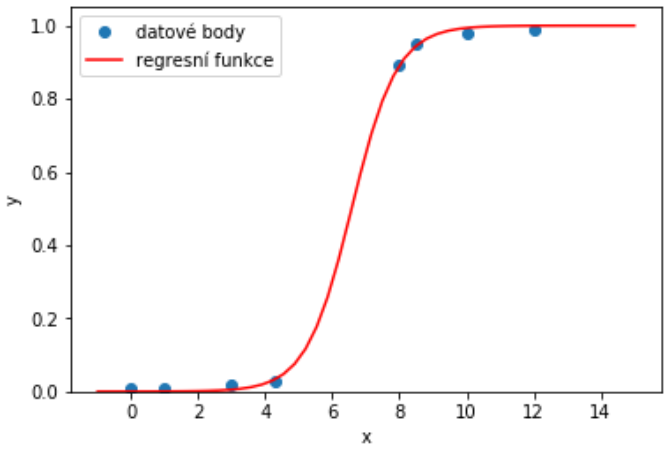
Pokud se výše uvedené rovnice spojí, tak lze dostat následující rovnici funkce:

$$Y = (m_1 + m_2) \times x + (b_1 + b_2). \tag{23}$$

Tato rovnice však opět definuje rovnou přímku, takže použití této rovnice nedává velký smysl. Cílem je tuto linku ohnout, čehož je možné dosáhnout pomocí sigmoidní funkce s hodnotami od 0 až 1, která má tvar:

$$Y = \frac{1}{1+e^{-x}}. \tag{24}$$

Sklon čáry řídí ostrost tvaru „S“ a parametr *b* určuje bod uprostřed. Takováto sigmoidní funkce ve tvaru „S“ pak velmi dobře charakterizuje výše uvedená data – viz obrázek 13.



Obrázek 13: Použití sigmoidální funkce [vlastní]

Postup při regresi pomocí neuronových sítí

Postup, jakým je možno realizovat regresi pomocí neuronových sítí, je k dispozici v rámci následujících kroků [27]:

1. Zpracování datové sady – vyřazení proměnných s chybějícími hodnotami, zakódování kategorických proměnných.
2. Vytvoření hluboké neuronové sítě – definování sekvenčního modelu, přidání skryté vrstvy, použití aktivační funkce pro skryté vrstvy.
3. Otestování modelu.
4. Vyzkoušení jiné techniky strojového učení.

V následujících krocích je možné dále více pracovat s datovou sadou, vyzkoušet jiné typy neuronových sítí a ladit zjištěné modely, aby byla chyba funkce minimální.

Přeučení

Jedním z hlavních aspektů učení modelu při strojového učení je vyhnout se tzv. „overfitting“, česky přeučení. To z toho důvodu, že model bude mít nízkou přesnost, pokud je přeučený. Děje se to proto, že se model snaží příliš silně zachytit šum (z angl. noise) v souboru dat v procesu učení. Jako šum (noise) jsou označovány datové body, které ve skutečnosti nepředstavují skutečné vlastnosti dat, ale náhodná data.

Koncept vyvážení zkreslení a rozptylu (balancing bias and variance) je užitečný pro pochopení přeučení. Jedním z dalších způsobů, jak se vyhnout přeučení, je použití křížové validace (cross validation), která pomáhá při odhalení chyb při testování a při rozhodování o tom, které parametry pro daný model fungují nejlépe.

Regularizace

Regularizace je forma regrese, která omezuje nebo reguluje odhady koeficientů na nulu, čímž zamezuje vytváření učení komplexnějšího modelu, aby se předešlo riziku přeučení. Jednoduchý vztah pro lineární regresi je zopakován níže, kde y představuje

naučený vztah a β představuje odhady koeficientů pro různé proměnné nebo prediktory (x):

$$y \approx \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_D x_D \,. \tag{25}$$

Postup regularizace pak zahrnuje funkci ztráty, známou jako zbytkový součet čtverců nebo RSS. Koeficienty jsou zvoleny tak, aby minimalizovaly tuto funkci ztráty [28]:

$$RSS = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{D} \beta_j x_{ij})^2 \,. \tag{26}$$

Tím se nastaví koeficienty na základě tréninovacích dat. Pokud je v trénovacích datech šum, pak odhadované koeficienty nebudou dobře zobecňovat budoucí data. Toto je situace, kdy dochází k regularizaci a zmenšuje nebo reguluje tyto naučené odhady směrem k nule [28].

Ridge regrese

Ridge regrese je takový způsob regrese, kde rovnice RSS je upravena přidáním množství smrštění. Pak jsou koeficienty odhadnuty minimalizací této funkce. Zde je λ parametr ladění, který rozhoduje, jak moc je cílem snížit vypovídací hodnotu daného modelu [28]:

$$\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{D} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{D} \beta_j^2 = RSS + \lambda \sum_{j=1}^{D} \beta_j^2 . \tag{27}$$

Zvýšení flexibility modelu je reprezentováno zvýšením jeho koeficientů, a pokud se má výše uvedená funkce minimalizovat, pak tyto koeficienty musí být malé. Tímto způsobem Ridge regresní technika zabraňuje příliš vysokým hodnotám koeficientů.

Když $\lambda = 0$, tak celý přidaný člen nemá žádný vliv a odhady vytvořené Ridge regresí budou rovny nejmenším čtvercům. Nicméně, jakmile $\lambda \to \infty$, dopad srážkového členu pak roste a odhady koeficientu Ridge regrese se budou blížit nule. Jak je vidět, volba dobré hodnoty λ je kritická [28].

Lasso regrese

Lasso regrese je další variantou regularizace, ve které je uvedená funkce minimalizována. Je jasné, že tato variace se liší od Ridge regrese pouze při penalizování vysokých koeficientů. Používá | β_j | (modul) místo čtverců β. Ve statistice je toto označováno jako norma L1 [28]:

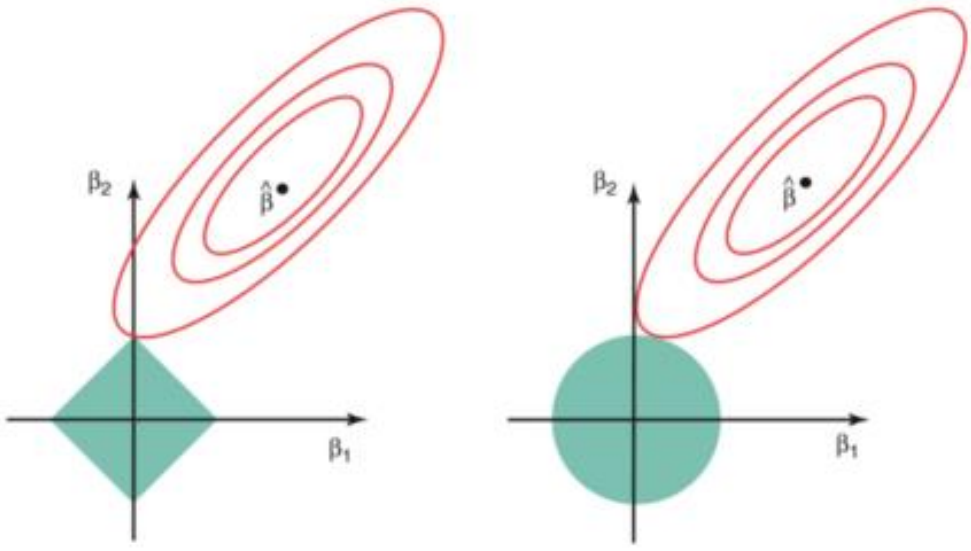
$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{D}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{D}|\beta_j| = RSS + \lambda \sum_{j=1}^{D}|\beta_j| \quad (28)$$

Lasso může být považováno za rovnici, kde součet modulů koeficientů je menší nebo roven s. Zde je s konstanta, která existuje pro každou hodnotu faktoru smrštění λ. Tyto rovnice jsou také označovány jako funkce omezení.

Podle výše uvedené formulace je Ridge regrese vyjádřena $\beta_1^2 + \beta_2^2 \leq s$. To znamená, že koeficienty regrese hřebene mají nejmenší RSS (ztráta funkce) pro všechny body, které leží v kruhu dané $\beta_1^2 + \beta_2^2 \leq s$. Podobně pro lasso se rovnice vyjádří jako $|\beta_1| + |\beta_2| \leq s$. To znamená, že koeficienty pro lasso mají nejmenší RSS (ztrátovou funkci) pro všechny body, které leží v kosočtverci daném funkcí:

$$|\beta_1| + |\beta_2| \leq s. \quad (27)$$

Obrázek 14 níže popisuje tyto rovnice [28].



Obrázek 14: Regularizace pomocí metody Ridge regrese a metody Lasso [28]

Elastické sítě

Regularizační metoda pomocí elastických sítí zahrnuje jak regularizační metodu Lasso (L1), tak Ridge regularizaci (L2). V regulaci L1 je přidán šum velikosti lambda | w^* |, zatímco v L2 regularizaci je šum velikosti lambda | w^* |, kde | w^* | je velikost vektoru optimálního parametru. Při regulaci se přidává lineární součet šumu dané funkce. Objektivní funkce pak tedy vypadá následovně [29]:

$$argmin_{\omega}(f(\omega^*) + \lambda_1|\omega^*| + \lambda_2|\omega^*|^2). \quad (30)$$

Regularizace L1 a L2 jsou speciální případy regularizace pomocí elastických sítí [29].

1.2.7 Metoda XGBoost

Systémy založené na strojovém učení jsou dnes hojně používány např. při ochraně počítačů před spamy, reklamami a škodlivými útoky. Systémy detekce anomálií pomáhají experimentálním fyzikům v událostech, které vedou k novým poznatkům. Tyto úspěšné aplikace pohánějí dva důležité faktory: využití efektivních (statistických) modelů, které zachycují složité závislosti na údajích a škálovatelné systémy učení, které se učí z velkých datových souborů, jako je právě metoda XGBoost.

Metodě XGBoost je založena na škálovatelném rozvětvování stromů a představuje vysoce efektivní a široce používanou škálovatelnou metodu strojového učení. Metoda XGBoost dnes poskytuje významné výpočty dle mnoha standardních klasifikačních srovnávacích kritérií. LambdaMART, varianta této techniky pro hodnocení, dosahuje nejmodernějšího způsobu výpočtu výsledku.

Kromě toho, že se tato technika používá jako samostatný prediktor, je také začleněna do reálných výpočtů pro predikci míry prokliků reklamy.

Metoda XGBoost byla dle článku z konference KDD 2016 v Kalifornii [30] testována na několika případech v rámci soutěže KDDCup v roce 2015, kdy došlo k testování v porovnání s dalšími používanými metodami strojového učení. Druhou nejpopulárnější metodou byly hluboké neuronové sítě. Vítězný tým však vždy využíval metodu XGBoost k učení modelu.

Tyto výsledky ukazují, že metoda XGBoost poskytuje nejnovější přístupy k výpočtům a také nejlepší výsledky v široké řadě problémů. Příklady problémů v těchto vítězných řešeních zahrnují [30]: predikci prodejního obchodu; klasifikace událostí fyziky vysokých energií; klasifikace webového textu; predikce chování zákazníků; detekce pohybu; predikce míry prokliků reklamy; klasifikace malwaru; kategorizace výrobků; predikce rizika apod.

Nejdůležitějším faktorem úspěchu metody XGBoost je její škálovatelnost ve všech scénářích. Škálovatelnost XGBoost je dána několika důležitými systémy a optimalizací algoritmů – nový algoritmus pro učení stromů pro zpracování řídkých dat; teoreticky oprávněná procedura váženého kvantilního náčrtu umožňuje manipulaci s váhami datových vzorků v přibližném stromovém učení. Paralelní a distribuované výpočty zajišťují rychlejší učení, což umožňuje rychlejší průzkum modelu. Ještě důležitější je, že XGBoost využívá mimo-procesorové výpočty a umožňuje zpracovat sto milionů příkladů najednou [30].

XGBoost je souborová metoda učení. Někdy nemusí stačit spoléhat na výsledky pouze jednoho modelu strojového učení. Souborové učení pak nabízí systematické řešení kombinující prediktivní sílu typů učení. Výsledkem je pak jediný model, který dává agregovaný výstup z několika modelů. Základní modely, které tvoří soubor, mohou být buď ze stejného algoritmu učení, nebo z různých algoritmů učení.

Metoda XGBoost se skládá ze dvou základních kroků – bagging a boosting. Jedná se o dva typy široce používaného souborového učení. Ačkoli tyto dvě techniky mohou být použity s několika statistickými modely, nejvíce převládající použití bylo s rozhodovacími stromy.

Bagging

Zatímco rozhodovací stromy jsou jedním z nejsnadněji interpretovatelných modelů, vykazují však velmi variabilní chování. Například pokud se použije jeden trénovací soubor dat, který se náhodně rozdělí na dvě části, tak při učení pomocí rozhodovacího stromu se získají dva modely s rozlišnými výsledky. Říká se, že rozhodovací stromy jsou v důsledku tohoto chování spojeny s vysokým rozptylem. Bagging (pytlování) nebo Boosting agregace pomáhá snížit rozptyl u každého učícího modelu. Několik rozhodovacích stromů, které jsou generovány paralelně, tvoří základní prediktory techniky pytlování. Data odebraná s náhradou se těmto prediktorům podávají za účelem učení. Konečnou predikcí je průměrný výstup ze všech základních prediktorů.

Boosting

Při Boostingu se stromy vytvářejí postupně tak, že každý následující strom má za cíl snížit chyby předchozího stromu. Každý strom se učí od svých předchůdců a aktualizuje zbytkové chyby. Základní učící modely při Boostingu jsou velmi slabé, jejich systematická chyba je vysoká a prediktivní síla je jen o něco lepší než náhodné hádání. Každý z těchto slabých modelů přispívá některými důležitými informacemi pro predikci, což umožňuje, aby technika Boosting vytvořila silný model učení efektivním spojením těchto slabých žáků. Konečný silný model snižuje jak systematickou chybu (bias), tak rozptyl [31].

Unikátní schopnosti metody XGBoost

XGBoost je populární implementace metody Boosting, která se dle [30] vyznačuje především následujícími vlastnostmi:

- Regularizace: XGBoost má možnost penalizovat složité modely prostřednictvím regularizace L1 a L2.
- Manipulace s řídkými daty: XGBoost obsahuje algoritmus pro rozpoznávání, který je založen na roztržitosti a který umožňuje zpracovávat různé typy vzorů v datech.

- Vážený kvantilový náčrtek: Většina existujících stromových algoritmů může najít dělené body, když mají datové body stejné váhy. Nejsou však vybaveny pro zpracování vážených dat. XGBoost má algoritmus distribuovaného váženého kvantilního náčrtu pro efektivní zpracování vážených dat.
- Struktura bloků pro paralelní učení: Pro rychlejší výpočet může XGBoost využít více jader na procesoru. To je možné díky konstrukci bloku v jeho návrhu systému. Data se třídí a ukládají do paměťových jednotek nazývaných bloky. Na rozdíl od jiných algoritmů to umožňuje opakované použití rozložení dat následnými iteracemi namísto opětovného výpočtu.
- Povědomí o vyrovnávací paměti: V XGBoost je pro získání statistik gradientu podle indexu řádků vyžadován nepřetržitý přístup do paměti. Proto byl XGBoost navržen tak, aby optimálně využíval hardware. To se provádí přidělením interních vyrovnávacích pamětí v každém vlákně, kde lze ukládat statistiky gradientu.
- Výpočtová technika mimo jádro: Tato funkce optimalizuje dostupné místo na disku a maximalizuje jeho využití při zpracování velkých souborů dat, které se nevejdou do paměti [30].

Vlivy nastavovaných parametrů na učení

Před tím, než se začne metoda XGBoost realizovat a spustí se výpočetní mechanismus, tak se nejdříve musí nastavit tři typy parametrů: obecné parametry, boosting parametry a úlohové parametry úlohy [32]:

- Obecné parametry se vztahují k tomu, který posilovač se použije pro zvýšení výkonu, běžně stromový nebo lineární model.
- Posilovací parametry závisí na tom, který posilovač byl zvolen.
- Úlohové parametry učení se rozhodují o scénáři učení. Úlohy regrese mohou například používat různé parametry s úlohami řazení.

Obecné parametry

booster

Parametr booster rozhoduje o tom, zda budou použity stromové, nebo lineární modely. K dispozici jsou parametry gbtree, gblinear nebo dart [32]. Gbtree a dart používají stromové modely, zatímco gblinear používá lineární funkce.

Parametry stromového posilovače

eta (learning_rate, rychlost učení)

Zmenšení velikosti kroku použité v aktualizaci zabraňuje přeučení. Po každém kroku boostingu je možné přímo získat váhy nových funkcí a eta zmenšuje váhu těchto prvků, aby proces boostingu byl konzervativnější. Rozsah tohoto parametru se udává v rozmezí [0,1].

gamma

Parametr gamma rozhoduje o minimální redukci ztráty nutné k vytvoření dalšího oddílu na uzlu stromu. Čím větší je gama parametr, tím bude algoritmus konzervativnější. Rozsah, ve kterém se hodnota parametru gamma pohybuje je: [0, ∞].

max_depth

Parametr max_dept určuje maximální hloubku stromu. Zvýšení této hodnoty způsobí, že model bude složitější a bude s větší pravděpodobností vznikat přeučení. Hodnota 0 je akceptována pouze ve ztrátové politice, když je parametr tree_method nastaven jako „hist“ a neoznačuje žádné omezení hloubky. Při využití hlubokých stromů metody XGBoost se velkou měrou spotřebovává paměť při učení. Rozsah parametru max_depth je [0, ∞].

min_child_weight

Tento parametr označuje minimální součet váhy případu (hessian) potřebné u dítěte. Pokud krok stromové oblasti vyústí v uzel se součtem váhy případu menší než je parametr min_child_weight, pak se rozvětvující proces zastaví. Při lineární regresi to jednoduše odpovídá minimálnímu počtu datových bodů, které musí být v každém uzlu. Čím větší je min_child_weight, tím bude algoritmus konzervativnější. Rozsah tohoto parametru se udává v rozmezí [0, ∞].

colsample_bylevel, colsample_bytree

Parametry colsample představují skupinu parametrů pro podvzorkování sloupců. Všechny parametry colsample_by * mají rozsah (0, 1), výchozí hodnotu 1 a specifikují zlomek sloupců, které mají být podvzorkovány.

- Colsample_bytree je poměr dílčích vzorků sloupců při konstrukci každého stromu. Vzorkování probíhá jednou pro každý konstruovaný strom.
- Colsample_bylevel je poměr dílčích vzorků sloupců pro každou úroveň. Vzorkování probíhá jednou pro každou novou úroveň hloubky dosaženou ve stromu. Sloupce jsou odebrány z množiny sloupců vybraných pro aktuální strom.

Učící úlohové parametry

objective

Parametr objective má několik různých podob, kde se specifikuje předmět učení. Mezi možné objekty učení mohou patřit následující prvky [32]:

- reg: squared error: regrese se čtvercovou chybou;
- reg: logistika: logistická regrese;
- binární: logistika: logická regrese pro binární klasifikaci, pravděpodobnost výstupu;
- binární: logit raw: logistická regrese pro binární klasifikaci, výstupní skóre před logistickou transformací.

eval_metric

Hodnotící metriky pro validační data jako výchozí metriky jsou přiřazeny podle cíle. Uživatel může přidat více hodnotících metrik. Možnosti parametru eval_metric, které může parametr řešit, jsou např. [32]:

- rmse: chyba průměrného čtverce;
- mae: střední absolutní chyba;
- logloss: negativní log-pravděpodobnost;

- error: Binární klasifikační chybovost. Pro předpovědi bude hodnocení s hodnotou predikce větší než 0,5 považováno za pozitivní instance a ostatní za negativní instance.

1.3 Současné přístupy k predikci nemocnosti zaměstnanců

Z veřejně dostupných internetových zdrojů lze dohledat například článek od C.R.L. Boota a spol. z časopisu Occupational Medicine [33], který popisuje studii analyzující záznamy o zaměstnancích letecké společnosti z let 2005 a 2008. Jednalo se o datový soubor se záznamy o 7652 zaměstnancích. V datovém soubory byly údaje k jednotlivým zaměstnancům jako vysoký věk, nedávné těhotenství, záznamy o dětech, vlastnictví parkovací karty, zhoršené pracovní podmínky a předcházející záznamy o nemocnosti. Cílem studie bylo ověřit dlouhodobé a časté absence z důvodu nemoci. Dlouhodobá absence byla ve studii stanovena na 42 dní.

Výsledný model byl založen na logistické regresi se skóre testu dobré shody $P = 0.627$ a vysvětlenou odchylkou 10.9 %.

Dlouhodobá i častá nemocnost byla nejlépe predikována za pomocí předcházející nemocnosti, což je ve shodě se studiemi [34, 35, 36]. Zmiňovaným nedostatkem datového souboru byla nedostupnost informace důvodu absence z důvodu nemocnosti, tedy nemožnost rozlišit mezi dlouhodobě nemocným z důvodu chronické nemocnosti, nebo z důvodu pracovního úrazu. Jako vlivný faktor byl dále identifikován vysoký věk těhotenství v průběhu zkoumaného časového období.

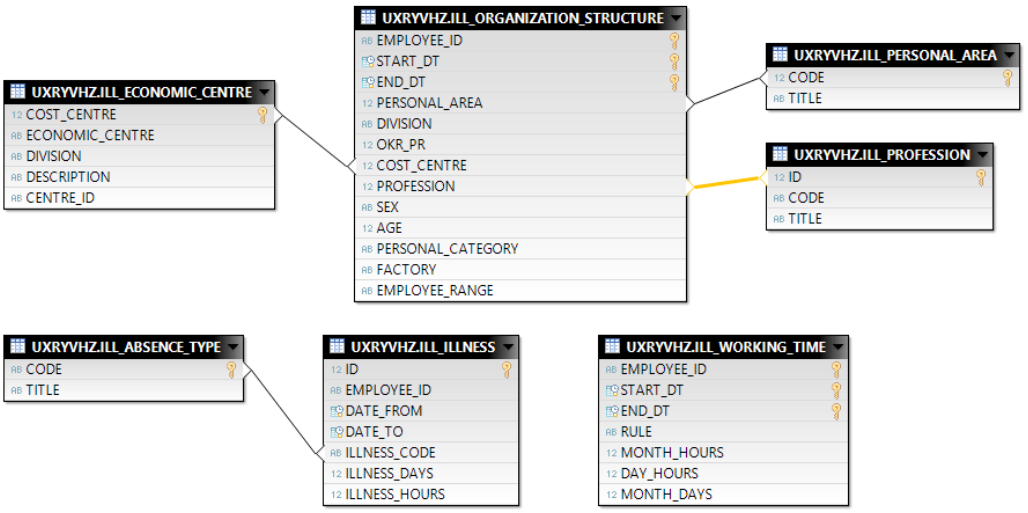
Závěrem je nutné podoktnout, že se nepodařilo nalézt žádnou podobnou studii, která by podobný problém řešila jako regresní úlohu celkové nemocnosti ve firmě. Například, stejně jako výše uvedená studie, se dle článku ve Skandinávském časopise „Scandinavian Journal of Work, Environment & Health“ [37] zabývá predikcí nemocnosti zaměstnanců veřejného sektoru pomocí Lasso regrese, avšak znovu se jedná pouze o klasifikační úlohu s cílem predikce dlouhodobé nemocnosti.

Současně používaný model predikce nemocnosti ve ŠKODA AUTO, a.s.

V současné době je ve firmě ŠKODA AUTO, a.s. v testovací fázi nasazený model predikce nemocnosti pro vybraná výrobní oddělení. Tento model funguje na principu agregované predikce pravděpodobností nemocnosti pro každého sledovaného zaměstnance zvlášť. Základem stávajícího modelu je logistická regrese, do které vstupují hodnoty z ARMA modelu, věk zaměstnance, pohlaví, případně informace o tom, jak dlouho je již nemocný. Pro každého zaměstnance je poté pomocí křivek hazardu a přežití počítána na každý následující den pravděpodobnost, zda se uzdraví, případně onemocní. Aktuálně nasazený model dosahuje hodnoty 0.81 vysvětleného rozptylu R2.

2 POPIS DATOVÉHO SOUBORU

Data byla k analýze poskytnuta od HR oddělení ŠKODA AUTO v prostředí SAP HANA, což je in-memory databázová platforma od firmy SAP. Pojem in-memory znamená, že je celá databáze umístěná pouze v operační paměti, a to umožňuje rychlý přístup k datům a veškerých operací s nimi. V poskytnutých tabulkách jsou údaje o nemocností pro každého zaměstnance ŠKODA AUTO, a.s. v období 1.1.2016 až 30.9.2018. Schéma poskytnutých dat vyobrazuje následující diagram (obrázek 15).



Obrázek 15: Databázové schéma s poskytnutými daty [vlastní]

Faktové tabulky:

ILL_ORGANIZATION_STRUCTURE – obsahuje informace o organizační struktuře. Každý zaměstnanec má v omezeném čase přiřazenu právě jednu pozici, útvar apod.

ILL_WORKING_TIME – fond pracovní doby. Každý zaměstnanec má ve sledovaném období (1.1.2016 – 30.09.2018) nejvýše jeden záznam o skutečné pracovní době – počet odpracovaných dnů v měsíci, počet pracovních hodin denně a počet odpracovaných hodin v měsíci.

ILL_ILLNESS – tabulka evidovaných nepřítomností. Každá nepřítomnost znamená jeden řádek s omezením v čase (start_dt – end_dt). Dále je zde vypočítán skutečný počet hodin a dnů nepřítomnosti.

Dimenze:

ILL_ECONOMIC_CENTRE – číselník nákladových středisek.

ILL_PERSONAL_AREA – číselník oblastí místa práce – Mladá Boleslav, Kvasiny, Vrchlabí.

ILL_PROFESSION – číselník profesí. Jedna profese dle kódu může mít rozdílné popisky, proto je použit syntetický primární klíč.

ILL_ABSENCE_TYPE – tabulka ILL_ILLNESS je omezena pouze na nemocnost, která má 11 konkrétních podtypů.

Data o nemocnosti nejsou k dispozici na denní bázi. HR má tedy všechna data za předchozí měsíc dostupná zpravidla pátý den v měsíci. Predikci na aktuální měsíc je tedy možné dělat vždy nejdříve pátý den v měsíci.

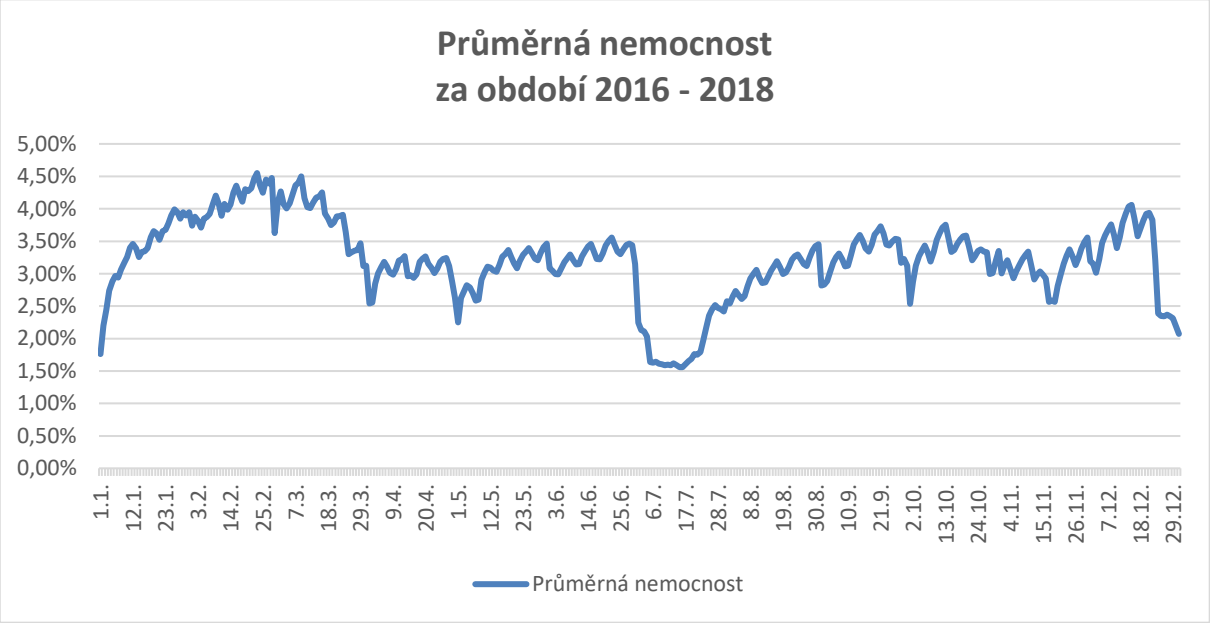
2.1 Popis proměnných

Pro predikci nemocnosti je v této práci použita následující množina proměnných:

1. DATE – datum [datetime]
2. M – hodnota měsíce, z kterého predikce vychází [integer]
3. Y – rok [integer]
4. DEPARTMENT – oddělení, v kterém zaměstnanec pracuje [string]
5. WEEKDAY – den v týdnu [integer]
6. WORKDAY – jedná se o pracovní den? [boolean]
7. EMPLOYEE_COUNT – počet zaměstnanců v oddělení [integer]
8. AGE – věk zaměstnance [integer]
9. SEX – pohlaví zaměstnance kde muž = 1 a žena = 0 [boolean]
10. DAY_HOURS – počet odpracovaných hodin v poslední den v měsíci [real]
11. EXECUTION – udává, zdali je na zaměstnance vystavena exekuce [boolean]
12. LOAN – má zaměstnanec půjčku od ŠKODA AUTO, a.s.? [boolean]

13. BLOOD_DONOR – udává, zdali měl zaměstnanec v poslední den v měsíci dovolenou [boolean]
14. VACATION – udává, zdali byl zaměstnanec v poslední den v měsíci darovat krev [boolean]
15. TEMP – teplota pro daný den [real], data pochází z ČHMÚ [38].
16. ILLNESS_DAY – uvádí, zdali byl zaměstnanec nemocný v poslední den v měsíci [boolean]

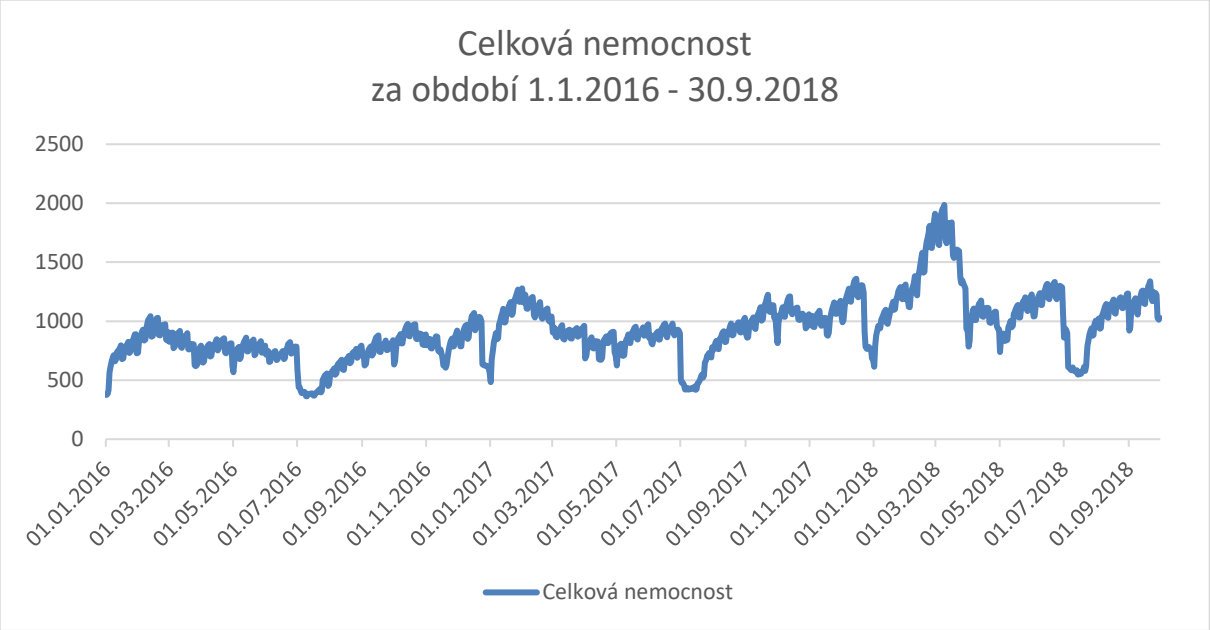
Průběh nemocností v čase se dá zobrazit jako průměrná nemocnost za celou firmu v celém dostupném období na obrázku 16.



Obrázek 16: Průměrná nemocnost za období 2016 – 2018 [vlastní]

Z grafu lze vidět, že nejnižší nemocnost je na začátku a konci roku. Zároveň je nejnižší nemocnost v letních měsících, kdy je období celozávodní dovolené. Naopak nejvyšší nemocnost bývá v únoru, březnu a listopadu a na začátku prosince. Údaje o dovolených jsou v datovém souboru k dispozici, proto vstupují do modelu jako vstupní proměnné, které ve formě průměrů dovolených na oddělení za poslední dostupný týden. Velkým nedostatkem datového souboru je však to, že nejsou dostupná data plánovaných dovolených (tedy budoucích hodnot), takže je do modelu takto není možné zavést.

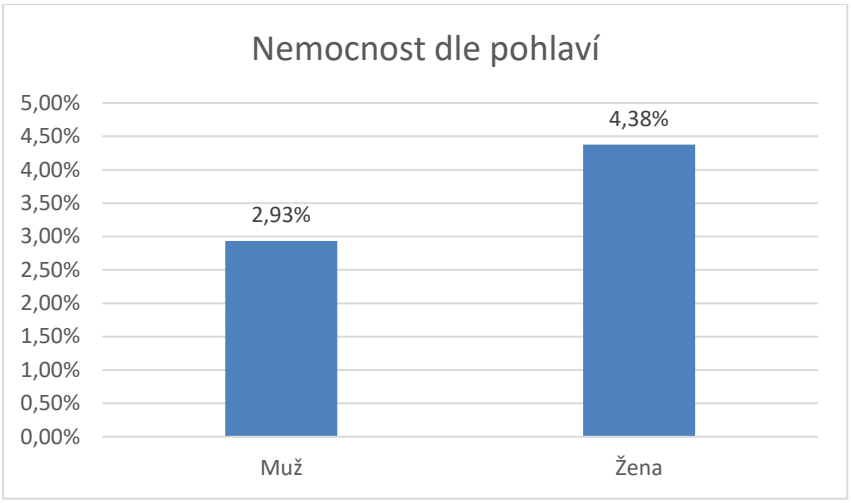
Vývoj nemocnosti lze také zobrazit v absolutní hodnotách, tedy součtu počtu nemocností po dnech. To je vidět na následujícím obrázku 17, který zobrazuje nemocnost v absolutních číslech pro celé období dostupné v datovém souboru.



Obrázek 17: Celková nemocnost za období 1.1.2016 - 30.9.2018 [vlastní]

Zde je vidět stejný trend jako v grafu průměrné nemocnosti. Z grafu je vidět, že absolutní počet nemocností stoupá v čase, to je jednoznačně způsobeno tím, že v čase rostl i počet zaměstnanců firmy. Zajímavý je však růst počtu nemocných lidí v březnu 2018. Na roku 2018 se bude přesnost modelu testovat. Bude tedy zajímavé sledovat, jak bude predikční model schopný tuto anomálii detekovat.

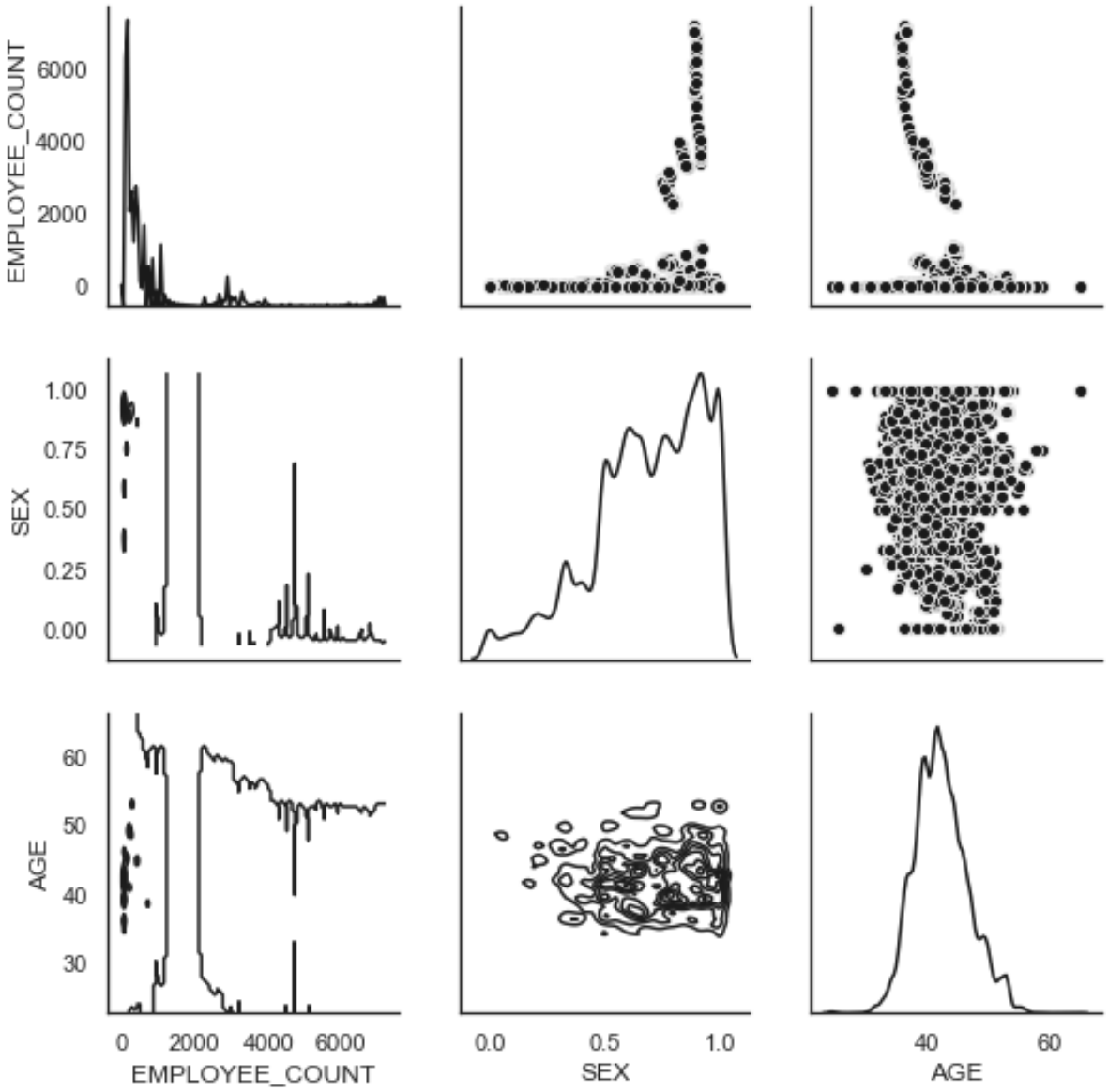
Průměrná nemocnost v datové sadě je 3,2165 %. Tomu odpovídá tvrzení, že zaměstnanec chybí přibližně jednou za 31 dní. Nemocnost dle pohlaví zobrazuje následující obrázek 18.



Obrázek 18: Nemocnost dle pohlaví [vlastní]

Ženy jsou tedy o cca 33 % náchylnější k absenci z důvodu nemoci. Pohlaví je tedy jedna z proměnných, která bude ve výsledném modelu zcela jistě vystupovat.

Nemocnost je také možné vyjádřit v závislosti na věku a pohlaví zaměstnanců. K tomu lze použít například párový graf za pomoci knihovny Seaborn [39] v prostředí Python (viz obrázek 19).



Obrázek 19: Vizualizace vybraných proměnných pomocí knihovny Seaborn [vlastní]

Tento typ grafu se v knihovně Seaborn nazývá PairPlot, a pomocí něho lze zobrazit vztahy mezi jednotlivými proměnnými v datovém souboru vícero způsoby najednou. Na hlavní diagonále jsou grafy rozložení pravděpodobností, které dávají přehled o relativním rozložení hodnot pro všechny proměnné. Nad hlavní diagonálou jsou takzvané korelační diagramy, které dokáží zobrazit hodnoty kartézských souřadnic pro každé dvě proměnné v datovém souboru. Vzhledem k tomu, že je k dispozici datových bodů velmi mnoho, projevuje se zde nedostatek tohoto typu grafu, a to ta, že se ztrácí informaci o hustotě zobrazovaných proměnných. Tento nedostatek ale řeší typ grafu umístěný pod hlavní diagonálou. Tomuto typu grafu se nazývá KdePlot.

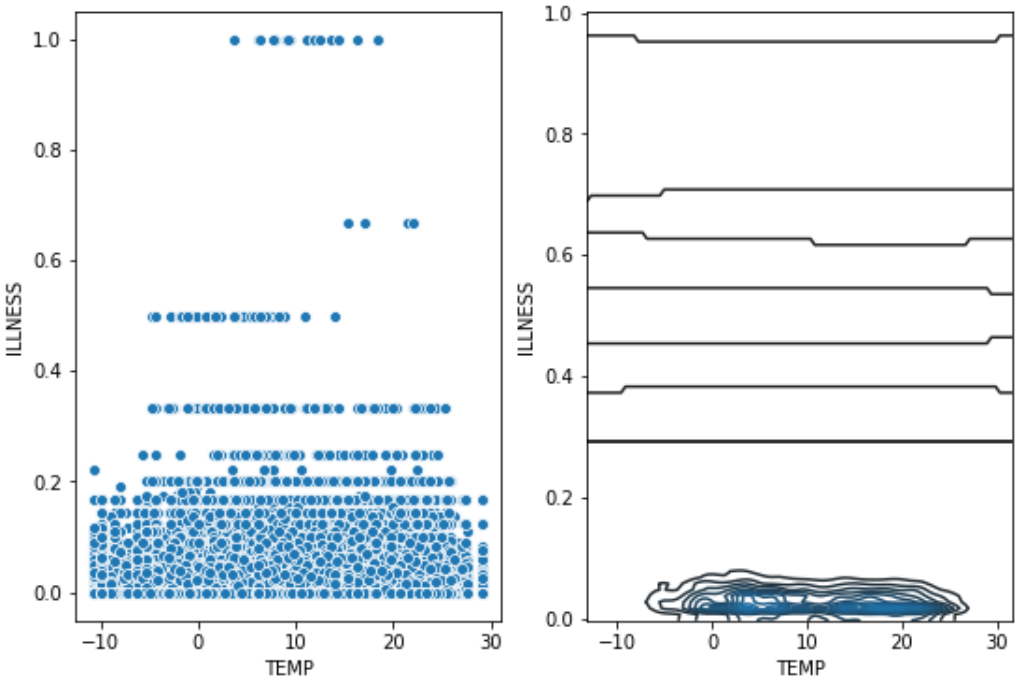
Data v uvedeném grafu vyobrazují proměnné již agregované na jednotlivá oddělení. Oddělení ve firmě jsou ve formátu XXX/YY, kde X představují písmena oddělení a Y číselné označení oddělení. Pro účely této práce byla agregace provedena tak, že se v potaz vzaly pouze oddělení na tři písmena. Výsledkem jsou tedy o něco větší oddělení než ve skutečnosti.

Z grafu lze mimo jiné jednoznačně vyčíst následující vlastnosti datového souboru:

- V datovém souboru převažují muži. (fakticky je v datovém souboru 74 % mužů a 26 % žen);
- Průměrný věk je něco přes 40 let (ve skutečnosti je to zhruba 41 let);
- Průměrné oddělení má do 300 zaměstnanců ve věku 40 až 50 let, kde převažují muži.

Jedna z proměnných byla dodána do datového souboru z externího zdroje. Tato proměnná zároveň figuruje ve výsledném modelu jako jedna ze vstupních proměnných. Jedná se o teplotní údaje za celé analyzované období, a to ve stupních Celsia, představující průměrné denní teploty pro ČR pro každý den. Teplotu v závislosti na nemocnosti pro stejný den zobrazuje následující KDE graf (obrázek 20).

Z grafu není vidět jednoznačná závislost, avšak již teď by se hodilo říci, že výsledný model teplotu jako vstupní proměnnou dokáže alespoň částečně využít.



Obrázek 20: Závislost nemocnosti na teplotě [vlastní]

2.2 Agregace dat

Data byla poskytnuta v systému SAP HANA. Data bylo nutné převést do jedné tabulky, která by sloužila jako vstupní data pro učení i testování algoritmů. Data bylo nutné nejdříve transformovat do jedné tabulky, a následně je převést do formy, která je zpracovatelná v programech použitých pro analýzu. Pro tyto účely byl použit formát CSV (Comma-separated values). Datová struktura formátu CSV spočívá v zapsání záznamů do řádků, kde jsou jednotlivé položky oddělené čárkou nebo středníkem. Pokud text obsahuje čárku nebo středník, je možné ho uzavřít do uvozovek. Pokud text obsahuje uvozovky, tak se v zápisu zdvojí [40].

Původní datový soubor obsahoval celkem 34 112 316 řádků. Tento datový soubor je tvořen kartézským součinem údajů o všech zaměstnancích pro každý den, kdy byli ve Škoda Auto v pracovním poměru. Celkem se jedná o záznamy 38 631 zaměstnanců.

Získaný datový soubor je vzhledem k predikované proměnné silně nevyvážený. To v našem případě znamená, že v datovém souboru významně převažuje zastoupení hodnot zdravých lidí. Takový stav odpovídá realitě, protože lidé bývají spíše zdraví, než nemocní. Pokud je cílem predikovat stav „nemocný“, který je v datovém souboru

zastoupen hodnotou „1“ v každém řádku, je vhodné pro fázi učení datový soubor upravit tím způsobem, že se pro trénovací podmnožinu dat vyberou řádky tak, aby zastoupení obou stavů bylo vyvážené [40]. Tomuto problému se ale lze vyhnout převodem klasifikačního problému na regresní. Úkolem je tedy vytvořit skupiny zaměstnanců, kteří mají nějaký společný znak, a proto je lze zařadit do skupin, pro které bude predikovaná veličina nabývat hodnot $(0; n) \epsilon N^0$, kde n představuje všechny zaměstnance v dané skupině. Nabízí se možnost agregovat zaměstnance pomocí kombinace společných znaků, které mají na predikovanou hodnotu nějaký vliv. K nalezení takové kombinace by mohla pomoci například shluková analýza. Pro účely firmy je však vhodné použít nějaký uchopitelný společný znak, a zde se přímo nabízí agregovat zaměstnance pomocí jednotlivých oddělení. V datovém souboru se nachází celkem 273 takových oddělení. Jednotlivá oddělení mívají jednotky až tisíce zaměstnanců. Dá se předpokládat, že v jednotlivých odděleních bude různá nemocnost, a proto je nutné, aby oddělení ve výsledném modelu vystupovalo nejen prostřednictvím počtu zaměstnanců, ale také jako kategorická proměnná, určující příslušenství k danému oddělení. Tato hypotéza bude dále v práci podrobně prozkoumána.

Data o nemocnosti nejsou k dispozici na denní bázi. HR má tedy všechna data za předchozí měsíc dostupná zpravidla pátý den v měsíci. Predikci na aktuální měsíc je tedy možné dělat vždy nejdříve pátý den v měsíci. Z tohoto důvodu do modelu budou vstupovat vždy údaje pouze za předchozí měsíc, a žádná data z měsíce, na který se provádí predikce. Požadavkem zadavatele bylo, že bude testována predikce na až tři měsíce dopředu. Pro zjednodušení výsledného skriptu zaokrouhlíme tento počet na devadesát dní. V případné implementaci modelu do produkčního prostředí není problém v kódu toto zaokrouhlení upravit na přesné požadované číslo. To znamená, že výsledkem této práce bude celkem devadesát modelů, kdy každý model bude predikovat hodnoty vždy na jeden další den dopředu. První model tedy počítá nemocnost na jeden den dopředu, druhý model počítá nemocnost na druhý den dopředu, a tak dále.

Protože vstupní data představují hodnoty vždy pro celé oddělení, bylo nutné z počátečního souboru získat pouze data, která má smysl do modelu agregovat. Do modelu tedy byly z počátku vybrány následující proměnné (tabulka 1).

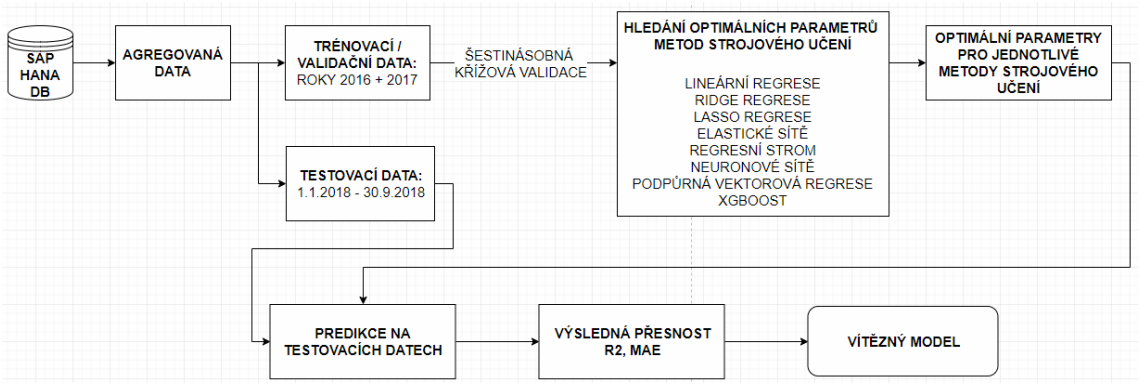
Tabulka 1: Vstupní proměnné

Proměnná	Popisné statistiky (regrese5)			
	Průměr	Minimum	Maximum	Sm.odch.
D	15,860	1,000	31,000	8,8648
M	6,581	1,000	12,000	3,4563
Y	2016,526	2016,000	2018,000	0,5160
WEEKDAY	4,015	1,000	7,000	1,9989
WORKDAY	0,685	0,000	1,000	0,4646
EMPLOYEE_COUNT	121,197	1,000	7047,000	496,9914
AGE	42,517	24,000	65,000	4,3265
SEX	0,693	0,000	1,000	0,2360
DAY_HOURS	6,540	3,000	10,500	2,1637
EXECUTION	0,003	0,000	0,200	0,0100
HANDICAP	0,003	0,000	0,250	0,0091
LOAN	0,162	0,000	1,000	0,1242
TEMP	9,975	-10,000	27,500	8,0425
BLOOD_DONOR_1M_PRECEDING	0,025	0,000	2,000	0,0834
BLOOD_DONOR_7D_PRECEDING	0,006	0,000	1,000	0,0407
VACATION_1M_PRECEDING	1,940	0,000	21,807	2,6005
VACATION_7D_PRECEDING	0,528	0,000	8,000	0,9885
30D_PRECEDING_AVG	0,329	0,000	13,000	0,6139
4W_PRECEDING_AVG	0,086	0,000	8,000	0,1975
3W_PRECEDING_AVG	0,087	0,000	8,000	0,1989
2W_PRECEDING_AVG	0,087	0,000	8,000	0,1980
PRECEDING_7	3,644	0,000	444,000	19,5211
PRECEDING_6	3,650	0,000	451,000	19,5621
PRECEDING_5	3,656	0,000	473,000	19,6091
PRECEDING_4	3,660	0,000	464,000	19,6364
PRECEDING_3	3,666	0,000	435,000	19,6733
PRECEDING_2	3,673	0,000	460,000	19,7197
PRECEDING_1	3,670	0,000	452,000	19,7296
ILLNESS_LASTDAY	3,678	0,000	465,000	19,7710

Proměnné AGE, SEX, EXECUTION, HANDICAP, LOAN. BLOOD_DONOR, VACATION a PRECEDING představují vždy průměrné hodnoty na oddělení. U pohlaví je to tak, že hodnota 1 představuje muže a hodnota 0 představuje ženu. Proměnné N_PRECEDING_AVG představují průměrné hodnoty nemocnosti pro dané oddělení za období N a proměnné PRECEDING_M představují absolutní hodnoty nemocnosti předcházející poslednímu dni. Tedy v případě, že ILLNESS_LASTDAY je hodnota pro 31.1.2018, PRECEDING_1 bude hodnota nemocnosti pro 30.1.2018.

3 MODELOVÁNÍ NEMOCNOSTI ZAMĚSTNANCŮ ŠKODA AUTO

Všechny výše uvedené metody strojového učení byly aplikovány v programovacím jazyku Python za pomocí dnes populární knihovny Scikit-learn. Tato knihovna je napsaná v programovacích jazycích Python, Cython, C a C++ [41]. Protože tato knihovna slouží pouze k aplikování metod strojového učení, bylo zapotřebí použít i jiné knihovny pro práci s maticemi ve fázi předzpracování dat a knihovnu XGBoost pro implementaci metody XGBoost.



Obrázek 21: Model predikce nemocnosti

Výše uvedený obrázek 21 zobrazuje použitou metodiku pro výběr vítězného modelu. Aplikování celého postupu v programovacím jazyku Python je popsáno v kapitolách 3.1 a 3.2.

3.1 Předzpracování dat

Načtení datového souboru do paměti

K načtení datového souboru do paměti slouží balíček Pandas. Pro další operace s datovým souborem je dále možné použít Pandas spolu s knihovnou NumPy, která umožňuje velmi efektivní práci s poli a maticemi.

```
import numpy as np
import pandas as pd
dataset = pd.read_csv('regrese5.csv')
```

První dva řádky importují balíčky pod uvedenými aliasy. Funkce read_csv() načte soubor ve formátu csv uvedený v parametru do paměti jako Pandas objekt pod jménem dataset.

Operace s proměnnými

Pokud je cílem přepočítat proměnné v datovém souboru, v tomto případě jde například o přepočítání proměnných, které představují počty nemocných zaměstnanců na oddělení na procentuální hodnoty, je možné použít operace s datovými objekty.

```
dataset.loc[:,'ILLNESS_FOLLOWING1'] =
(dataset.loc[:,'ILLNESS_FOLLOWING1']/dataset.loc[:,'EMPLOYEE_COUNT'])
```

Zde je takových proměnných celkem devadesát (odpovídající devadesáti dnům). Vzhledem k tomu, že takto je potřeba přepočítat všechny proměnné, je vhodné použít smyčku, například typu while.

```
#zadefinování počtu dní pro predikci
predict_length = 90
#převod proměnných na procentuální hodnoty
i=1
while i < predict_length+1:
      dataset.loc[:,'FOLLOWING_'+str(i)] = (dataset.loc[:,'FOLLOWING_'
      + str(i)]/dataset.loc[:,'EMPLOYEE_COUNT'])
      i += 1
```

Příkaz .loc umožnuje specifikovat pozice v datovém objektu, kterého se operace týká. Formát zadání se zapisuje jako [řádek, sloupec], kde symbol ":" může zastupovat všechny řádky. Řádky nebo sloupce je pak možné určovat buďto číslem představujícím index, nebo názvem řádku, respektive sloupce.

Převod datových objektů do NumPy polí

Protože knihovna Scikit-learn nepracuje s datovými objekty typu Pandas, je nutné pracovat s datovými poli typu NumPy. Převod na Numpy pole je velmi jednoduchý. Příkladem je rozdělení datového souboru na nezávislé a závislé proměnné, tedy proměnné *x*, respektive proměnné *y*. Toto se provede následovně:

```
#převod datového objektu dataset na NumPy pole X a y
X = dataset.iloc[:, 1:20].values
y = dataset.iloc[:,np.r_[3,21:111]].values
```

Indexy začínají v jazyku Python od nuly, tudíž první řádek kódu uvedeného výše vezme pro objekt dataset všechny řádky a sloupce s indexy 1 až 20, které v matici odpovídají druhému až dvacátému prvnímu sloupci, a uloží je jako pole typu NumPy. Parametr .values způsobí to, že do výsledného pole budou uložené pouze samotné hodnoty proměnných, nikoliv však názvy proměnných, tedy jména jednotlivých sloupců. Obdobně, v druhém řádku načítáme do proměnné *y* všechny hodnoty z objektu dataset, které odpovídají sloupcům s indexy 3 a 21 až 111. Takto si lze dovolit zjednodušení zápisu díky funkci NumPy_r, která slouží ke spojení více řezů (anglicky tzv. slice) dohromady.

Zpracování kategorických proměnných

V datovém souboru je několik kategorických proměnných. Protože knihovna Scikit-learn pracuje pouze s číselnými hodnotami, je nutné veškeré proměnné převést na čísla. V našem případě jde o proměnnou představující oddělení, která nabývá až tří písmen. Přesně pro tento účel obsahuje knihovna Scikit-learn funkci Label encoder.

```
#aplikace LabelEncoderu na první sloupec pole X
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
X[:, 0] = labelencoder_X.fit_transform(X[:, 0])
```

Tato funkce přiřadí každé distinktní kategorické nečíselné proměnné číselnou hodnotu. Metoda fit_transform() rovnou přiřadí příslušné hodnoty datům zadaným v parametru funkce. Toto se dá udělat i ve dvou krocích, kdy se použijí parametry fit() a transform() zvlášť. Pro opětovné získání původních hodnot lze použít metodu inverse_transform() [42].

Použitím této metody se získali v prvním sloupci datového souboru číselné hodnoty. Tímto krokem však zpracování kategorických hodnot nekončí, protože

algoritmus sám o sobě na základě číselných hodnot nedokáže rozlišit, zdali se jedná o kategorickou, nebo číselnou proměnnou, nabývající pouze kladných hodnot z oboru celých čísel. K řešení tohoto problému slouží funkce OneHotEncoder.

```
#úprava kategorických proměnných pomocí funkce OneHotEncoder
from sklearn.preprocessing import OneHotEncoder
onehotencoder = OneHotEncoder(categorical_features = (0,1,3))
X = onehotencoder.fit_transform(X).toarray()
```

Funkce OneHotEncoder převádí celá čísla do nových sloupců, které nabývají pouze hodnot 0 a 1. Toto je nutné pro práci s kategorickými proměnnými, protože jen tak je algoritmus dokáže správně využít.

Posledním krokem předzpracováním dat je úprava chybějících hodnot. V tomto případě původní datový soubor neobsahoval chybějící hodnoty, avšak vytvořením nových proměnných (hodnoty nemocnosti z předcházejících a následujících dnů) proměnné s chybějícími hodnotami vznikly, a to tam, kde datový soubor začíná, respektive končí. Kolik je v datovém souboru chybějících hodnot lze zjistit příkazem dataset.isna().sum(), kde dataset je název datového rámce, který se analyzuje.

K řešení tohoto problému je k dispozici v rámci knihovny Scikit-learn slouží funkce Imputer.

```
#úprava chybějících hodnot pomocí funkce Imputer
from sklearn.preprocessing import Imputer
imputer = Imputer(strategy = "most_frequent")
X = imputer.fit_transform(X)
y = imputer.fit_transform(y)
```

V případě práce s chybějícími hodnotami je na výběr z několika strategií, které se volí v parametru funkce.

- Mean – Nahradí chybějící hodnoty průměrem daného sloupce.
- Median – Nahradí chybějící hodnoty mediánem daného sloupce.
- Most_frequent – Nahradí chybějící hodnoty nejčastější hodnotou v daném sloupci.
- Constant – Nahradí chybějící hodnoty zvolenou hodnotou. Pro tuto volbu je povinný parametr fill_value, do kterého se vyplní požadovaná hodnota.

Vzhledem k povaze našich dat a také vzhledem k velikosti datového soboru si lze dovolit použít metodu most_frequent.

Vytvoření trénovacího a testovacího vzorku dat

Posledním krokem přípravy datového souboru je rozdělení na trénovací a testovací podmnožiny. Protože jsou k dispozici data v rozsahu 1.1.2016 až 30.9.2019, bude vhodné rozdělit datový soubor na roky 2016 a 2017, které budou představovat trénovací množinu dat, a testovat se bude na datech z roku 2018. K tomuto stačí použít standardní syntaxi pro operaci s maticemi.

```
#rozdělení datového souboru na trénovací a testovací podmnožiny
X_train = X[(X[:, 2] == 2016) | (X[:, 2] == 2017)]
X_test = X[X[:, 2] == 2018]
y_train = y[(y[:, 0] == 2016) | (y[:, 0] == 2017)]
y_test = y[y[:, 0] == 2018]
y_train = y_train[:, 1:predict_length+1]
y_test = y_test[:, 1:predict_length+1]
```

Pro metody, které mohou využít metody včasného zastavení trénovaní z důvodu prevence proti přeučení, je nutné z trénovacích dat vytvořit další podmnožinu, která se nazývá evaluační podmnožina, případně v některých prostředích validační podmnožina. K tomuto lze znovu využít nástroj train_test_split z balíčku Sklearn.

```
#rozdělení trénovací podmnožiny
from sklearn.model_selection import train_test_split
X_train, X_eval, y_train, y_eval = train_test_split(X_train, y_train,
test_size=0.1, random_state=7)
eval_set = [(X_eval, y_eval)]
```

Tato funkce vytvoří z dat dosazených v parametrech funkce novou podmnožinu podle poměru zadaného v parametru test_size. Parametr random_state zde figuruje jako fixní seed pro generování vždy stejných vzorků dat.

3.2 Nastavení metod strojového učení

Nastavení používané metody probíhá standardním volání funkce. V této ukázce je zvolena vítězná metoda XGBoost.

```
#definování metody strojového učení XGBoost
from xgboost import XGBRegressor
xgb_model = XGBRegressor(eval_metric = 'rmse', eval_set=eval_set)
```

Tímto se definují pouze základní fixní parametry, které se v průběhu hledání optimálních parametrů nebudou měnit. Jedná se o metriku pro ohodnocení modelu a definování evaluačního souboru dat.

Hledání optimálních parametrů zvolené metody

K časově nejnáročnější části trénování modelu patří hledání optimálních parametrů modelu. To se dá dělat dvěma způsoby. Buďto se zvolí pro každý parametr slovních hodnot, ze kterých program vyzkouší postupně všechny kombinace (grid search), nebo se zvolí rozsah hodnot pro každý parametr zvlášť, a program postupně vyzkouší dopředu zvolený počet náhodných hodnot z jednotlivých rozsahů (random search). Možné je tyto dva přístupy kombinovat, tedy po náhodném prohledání optimálních parametrů je možné selektivně parametry ladit. I když se může na první pohled zdát, že vyzkoušení všech kombinací ze slovníku najde lepší kombinaci parametrů, zpravidla tomu tak nebývá. Důkazem k tomuto může být například studie „Random Search for Hyper-Parameter Optimization“ z Univerzity v Montrealu, nebo článek „Comparing randomized search and grid search for hyperparameter estimation“ na oficiálních stránkách knihovny Scikit-learn [43] [44].

```
from sklearn.model_selection import RandomizedSearchCV
parameters = {
        'learning_rate': sp.stats.uniform(0.001, 0.1),
        'min_child_weight': [1, 2, 3, 4, 5, 6, 7],
        'colsample_bylevel' : sp.stats.uniform(0.2, 0.8),
        'subsample': sp.stats.uniform(0.2, 0.8),
        'colsample_bytree': sp.stats.uniform(0.2, 0.8),
        'max_depth': sp.stats.randint(3, 30),
        'gamma': sp.stats.uniform(0.0001, 0.05),
        'n_estimators': sp.stats.randint(200, 3000)
        }

#Náhodné hledání parametrů za použití šestinásobné křížové validace
xgb_random = RandomizedSearchCV(estimator = xgb_model,
param_distributions = parameters, n_iter = 5000, cv = 6, verbose=1,
random_state=0, return_train_score=True, scoring='explained_variance',
                                n_jobs = 6, pre_dispatch = '2 *
n_jobs')

#Spuštění učení metodou „fit“
xgb_random.fit(X_train, y_train)
```

K hledání optimálních parametrů náhodným prohledáváním lze použít funkci RandomizedSearchCV. Pro tuto funkci je nutné v parametrech zadat metoda strojového učení v parametru estimator, skórovací metriku v parametru scoring a slovník s hledanými parametry v parametru param_distribution. Dále je vhodné použít následující volitelné parametry:

- N_iter – volí počet iterací pro prohledávání

- Cv – Nastavuje strategii pro křížovou validaci. Defaultně nastavena pětinásobná křížová validace. Počet validačních podmnožin se však dá nastavit zadáním celého čísla.
- Verbose – Tento parametr zajistí výpis průběhu hledání parametrů, tedy počet provedených iterací, jednotlivé skóre a celkový čas běhu skriptu, pokud je zadán spolu s parametrem return_training_score.
- Random_state – fixní seed pro generování pseudonáhodných čísel, který zajistí vždy stejný výběr prvků pro podmnožiny při křížové validaci.
- N_jobs – Nastavuje počet vláken pro výpočet.
- Pre_dispatch – Nastavuje se jako ochrana proti přetečení paměťového bufferu. V případě, že by nebyl zadán, načetly by se před trénováním do paměti najednou všechny trénované modely. Takový stav může vést k zamrznutí systému z důvodu přeplnění paměti RAM.

Protože hledání optimálních parametrů probíhalo na počítači s procesorem s šesti fyzickými jádry, bylo vždy prováděno za použití šestinásobné křížové validace.

3.3 Proces učení modelu

Jak již bylo řečeno, výstupem této práce má být model, který bude schopný predikovat nemocnost až na 90 dní dopředu. K tomu se dá přistoupit například tak, že datum bude vstupovat do modelu mezi ostatními proměnnými jako kategorická proměnná složená ze tří prvků, které představují den, měsíc a rok, a model se bude snažit naučit souvislosti mezi těmito vstupními proměnnými a výstupní proměnnou. Tento přístup byl testován jako první, avšak vzhledem k tomu, že uvedené metody nebyly dostatečně schopné využít proměnnou představující den v měsíci, tento přístup byl později nahrazen jiným přístupem, který u jednotlivých metod představuje zlepšení v přesnosti predikce až o 5% (měřeno koeficientem R2).

Vzhledem k faktu, že nová data o nemocnosti jsou vždy dostupná pouze na měsíční bázi, bude se i predikce spouštět pouze jednou měsíčně, a to vždy na začátku nového měsíce. Do modelu tedy budou vždy vstupovat data aktuální v posledním dni předcházejícího měsíce. Po vytvoření proměnných, které zahrnují data nemocností

z předchozích dní (průměr za posledních třicet dní a průměr za poslední týden) a následném omezení trénovací množiny dat pouze na poslední dny v měsíci, vedlo k zpřesnění predikčního modelu, a to zejména u metody XGBoost. Touto úpravou vstupních dat se stala tato metoda nejpřesnější, s výsledky uvedenými v následující podkapitole.

Protože bylo nutné vytvořit celkem devadesát modelů pro každou z testovaných metod, nepřipadala v úvahu tvorba modelů v nástrojích neumožňujících nějakou možnost automatizace použitím skriptů. Právě kvůli této potřebě byla zvolena implementace v programovacím jazyku Python za použití knihovny Scikit-learn. Implementace v pythonu umožňuje automatickou tvorbu modelů na celé predikované období pro všechny testované metody, a to včetně uložení modelů ve formátu joblib do souborů na disku. Všechny použité skripty jsou na přiloženém CD.

3.4 Výsledky predikce

V tabulce 2 lze vidět shrnutí výsledků na testovacích datech z roku 2018 pro všechny použité metody strojového učení za celé devadesátidenní období. Jako hodnotící metriky byly vybrány vysvětlený rozptyl R2 a MAE.

Tabulka 2: Výsledky použitých metod při predikci na 90 dní

Regresní metoda	Vysvětlený rozptyl R2	MAE	Směrodatná odchylka R2 pro 90D
Lineární regrese	0.915	1.951	0.036
Elastické sítě	0.923	1.645	0.036
Ridge regrese	0.926	1.760	0.041
Lasso regrese	0.920	1.532	0.042
MLPRegressor	0.934	1.880	0.029
SVR	0.917	1.645	0.028
Regresní strom	0.907	1.495	0.044
V současnosti používaný model (ARIMA + Logistická regrese)	0.810		
XGBoost	0.941	1.150	0.029

Tabulka 3: Výsledky vysvětleného rozptylu R2 pro vybrané časové úseky predikce

Regresní metoda	1D	7D	30D	90D
Lineární regrese	0.977	0.955	0.929	0.915
Elastické sítě	0.944	0.948	0.914	0.923
Ridge regrese	0.971	0.956	0.919	0.926
Lasso regrese	0.953	0.948	0.907	0.920
MLPRegressor	0.951	0.950	0.940	0.934
SVR	0.966	0.956	0.927	0.917
Regresní strom	0.907	0.869	0.902	0.907
XGBoost	0.954	0.953	0.939	0.941

Tabulka 3 uvádí vysvětlené rozptyly pro sledovaná časová období pro všechny použité metody. Větší číslo znamená lepší výsledek. Zde lze vidět například fakt, že na kratší časový úsek dle vysvětleného rozptylu vychází o 0.01 R2 lépe než XGBoost regrese pomocí neuronových sítí.

Tabulka 4: Směrodatné odchylky vysvětleného rozptylu R2 pro různá predikovaná období

Regresní metoda	7D	30D	90D
Lineární regrese	0.019	0.017	0.036
Elastické sítě	0.018	0.021	0.036
Ridge regrese	0.020	0.021	0.041
Lasso regrese	0.016	0.024	0.042
MLPRegressor	0.006	0.018	0.029
SVR	0.017	0.017	0.028
Regresní strom	0.072	0.046	0.044
XGBoost	0.019	0.019	0.029

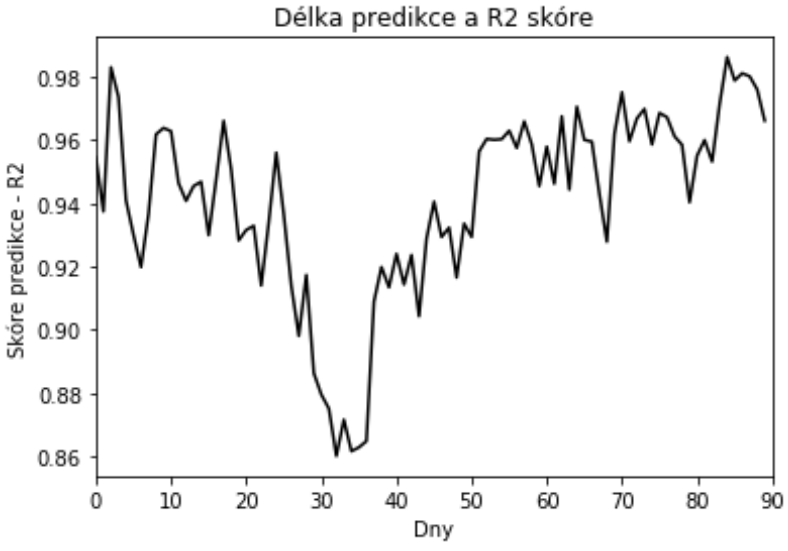
Tabulka 4 uvádí směrodatné odchylky vysvětleného rozptylu R2 pro sledovaná období. V této tabulka znamenají nižší hodnoty stabilnější predikce v čase. Z tohoto hlediska vychází nejlépe metoda SVR spolu s neuronovými sítěmi a metodou XGBoost.

Z výše uvedených tabulek je vidět, že nejlepších výsledků dosáhla metoda XGBoost, a to s následujícími parametry:

```
XGBRegressor(booster='gbtree',
             colsample_bylevel=0.83,
             colsample_bytree=0.821,
             gamma=0.00042975,
             learning_rate=0.084725,
             max_depth=14,
             min_child_weight=2,
             n_estimators=167,
             objective='reg:linear',
             subsample=0.814,
             max_delta_step = 2)
```

Pro tuto metodu bylo potřeba převést vstupní proměnné obsahující data o nemocnosti na procentuální hodnoty, protože jinak docházelo k podhodnocování počtu nemocných.

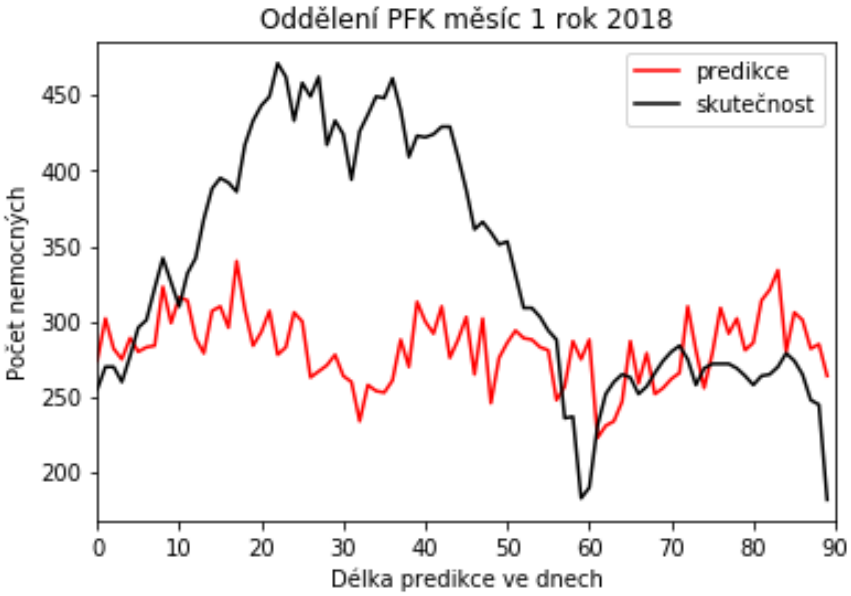
Výše uvedené výsledky jsou uváděné vždy za celý testovací soubor. Vzhledem k tomu, že v testovacím souboru se predikce provádí vždy za každé období, jednotlivé predikce se navzájem překrývají. To je způsobené tím, že se predikce provádí vždy na tři měsíce dopředu. Predikovaný měsíc číslo 2 pro predikci spouštěnou v lednu je únor, ale pro predikci spouštěnou v únoru, je únor měsíc číslo 1. Přesnost v čase zobrazuje následující graf (obrázek 21).



Obrázek 22: Vysvětlený rozptyl R2 v závislosti na predikovaném období pro vítěznou metodu [vlastní]

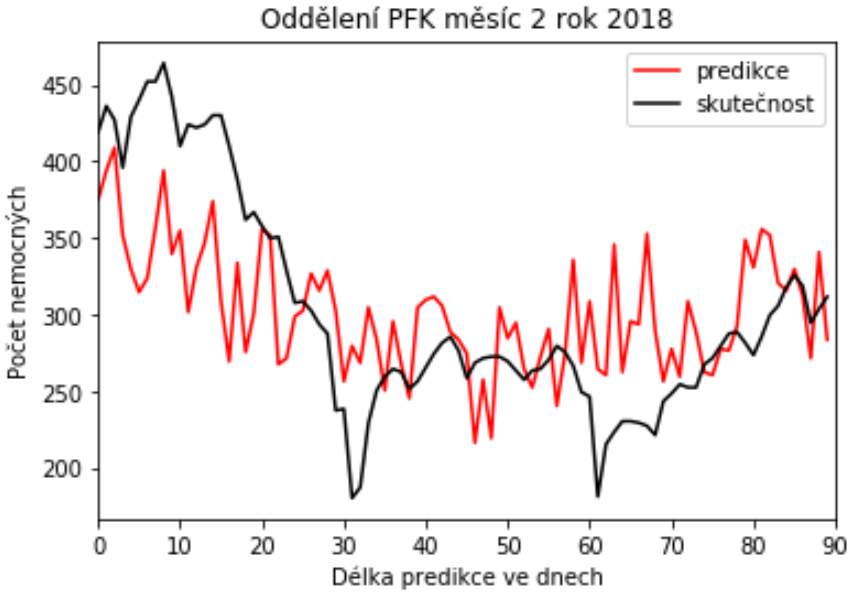
Z grafu je vidět, že navzdory očekávání, že přesnost predikce bude v čase klesat, nejproblémovější je právě predikce pro druhý měsíc. Tento fakt je způsobený právě

predikcí z měsíce ledna, kdy model nedokáže dostatečně odhadnout velký nárůst nemocnosti, která se stala v únoru v důsledku chřipkové epidemie. Tuto skutečnost může přiblížit následující obrázek, který se týká největšího oddělení ve firmě (PFK), které v lednu roku 2018 mělo zhruba sedm tisíc zaměstnanců.



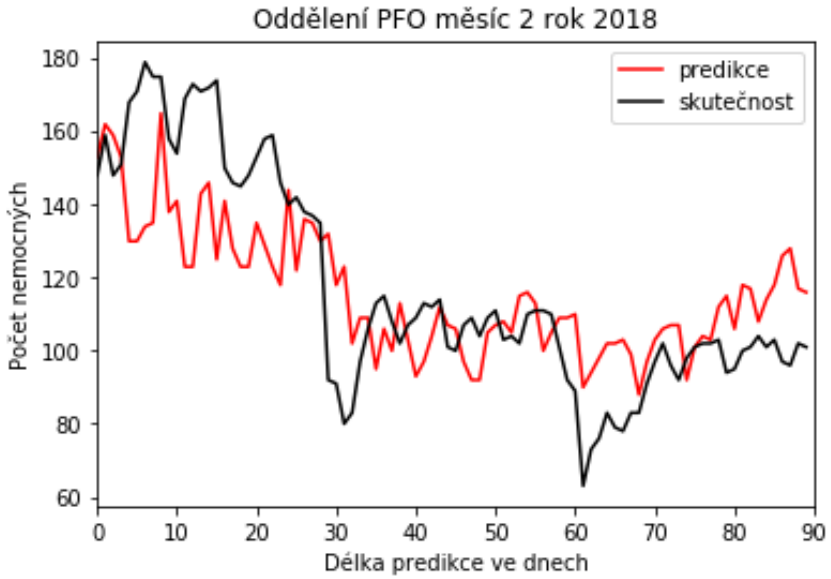
Obrázek 23: Predikce výstupu pro oddělení PFK – leden [vlastní]

Pokud se však zobrazí predikce z měsíce února pro stejné oddělení, je vidět, že se model dokáže za pomocí aktuálních hodnot přizpůsobit, a predikci tak zpřesnit.



Obrázek 24: Predikce výstupu pro oddělení PFK – únor [vlastní]

Tímto únorovým výkyvem, který nebyl v trénovacím vzorku dat obsažen, došlo ke zhoršení průměrných výsledků pro první měsíce. Výše uvedený příklad je spíše extrémním příkladem nepovedené predikce výsledného modelu. Porovnají-li se výše uvedené výsledky s jiným, o něco menším výrobním oddělením za stejný měsíc, je jasně vidět, že výsledný model dokáže dobře predikovat hodnoty nemocnosti, a je schopen vysvětlit většinu rozptylu v datech (obrázek 24).



Obrázek 25: Predikce výstupu pro oddělení PFO – únor [vlastní]

Knihovna XGBoost obsahuje funkci feature_importance, která vypočítá důležitost prediktorů modelu. Funkce počítá míru relevance (gain), který dané proměnné do modelu přináší. Tento výpočet důležitosti je popsán v [5]. Pro jeden rozhodovací strom se pro konkrétní proměnnou X_l vypočítá následovně:

$$I_l^2(T) = \sum_{t=1}^{J-1} \hat{\imath}_t^2 I(v(t) = l). \qquad (31)$$

Suma se počítá pro všechny interní uzly stromu (J-1). V každém uzlu t se jedna ze vstupních proměnných $X_{v(t)}$ použije k rozvětvení regionu spojeného s tímto uzlem na dva podregiony, a každému je přiřazena jedna konstanta jako odpovídající hodnota. Vybraná proměnná je ta, která dává maximální zlepšení $\widehat{\imath_t^2}$, tedy ve čtverci rizika chyby pro daný region. Čtverec relativní důležitosti proměnné X_l je suma všech vylepšení v interních uzlech stromu, kde byla vybrána jako rozdělující proměnná. Pro celý les se důležitost vypočítá jako průměr mezi všemi stromy.

Výsledky funkce predictor_importance pro vítěznou metodu XGBoost zobrazuje následující tabulka 3. Klíčovou proměnnou byla nemocnost v poslední den, následoval 7-denní průměr nemocností a počet zaměstnanců. Finanční situace zaměstnanců se naopak jeví jako nedůležitý determinant nemocnosti.

Tabulka 5: Relativní důležitost použitých proměnných

proměnná	**důležitost proměnné**
nemocnost v poslední den	0.3708
průměr nemocností (7 denní)	0.1539
počet zaměstnanců	0.1111
oddělení	0.0708
průměrný věk	0.0490
počet pracovních hodin denně	0.0416
průměrný počet dnů dovolených (7 dní)	0.0399
průměr nemocností (30 denní)	0.0321
průměrná denní teplota	0.0292
dárci krve (1 měsíc)	0.0267
měsíc	0.0247
rok	0.0160
podíl mužského pohlaví	0.0144
dárci krve (7 dní)	0.0136
podíl zaměstnanců s půjčkou	0.0053
podíl zaměstnanců v exekuci	0.0004
průměrný počet dnů dovolených (1 měsíc)	0.0004

ZÁVĚR

V teoretických základech práce byly představeny testované metody strojového učení pro účel predikce nemocnosti zaměstnanců firmy Škoda Auto, a.s. Následně byl popsán vstupní datový soubor a úpravy nutné k předzpracování dat. V další části byl prezentován způsob určení parametrů pro jednotlivé metody a technika pro učení jednotlivých modelů. Na konci byly prezentovány výsledky dosažené jednotlivými metodami. Nejlepšího výsledku bylo dosaženo za použití metody XGBoost se 167 estimátory, která dosahuje na testovacích datech přesností 94.1%. Rozhodnutí testovat model na datech z roku 2018 znamená znevýhodnění pro model, a to zejména proto, že na přelomu února a března roku 2018 lze vidět jednoznačně nejvyšší hodnoty nemocnosti, které v trénovacím vzorku dat nemají zastoupení. Vítězný model nedokázal tuto abnormalitu predikovat. Je však nutné dodat, že predikce s daty z února se již těmto hodnotám přizpůsobí, a predikce upřesní.

Největší váhu v modelu mají hodnoty nemocností z předchozích dní, což je stejný závěr, jako ve studii [33], která je popsaná v kapitole 1.3. Nejvyšší relativní váhu má hodnota nemocnosti z posledního dne, následovaná průměrnou nemocností na oddělení z posledního týdne.

Přílohu k této práci tvoří sada skriptů pro předzpracování dat a hledání optimálních parametrů pro jednotlivé metody a skript pro případnou implementaci vítězného modelu v prostředí umožňující spuštění kódu v Pythonu.

Na závěr by chtěl autor podotknout, že nasazením prediktivního modelu do produkce by práce na modelu neměla skončit. Nejen že je vhodné model pravidelně přeučit, ale do modelu je možné také přidat další proměnné. Proměnné, které by mohly vést k zpřesnění modelu, by mohly být například plánované dovolené, plánované směny nebo informace o zhoršených pracovních podmínkách. Jistě by také bylo dobré, pokud by se daly údaje o nemocnosti získávat častěji nežli jednou měsíčně.

CITOVANÁ LITERATURA

[1] SULTAN, K., H. ALI a Z. ZHANG. Big Data Perspective and Challenges in Next Generation Networks. *Future Internet*. 2018, 10(7), 10. DOI: 10.3390/fi10070056. ISSN 1999-5903. Dostupné také z: http://www.mdpi.com/1999-5903/10/7/56.

[2] NOWOZIN, S., P. V. GEHLER, J. JANCSARY a Ch. H. LAMPERT. Advanced Structured Prediction. London: Massachusetts Institute of Technology, 2014. ISBN 978-0-262-02837-0.

[3] ALPAYDIN, E. Introduction to machine learning. 2nd ed. Cambridge, Mass.: MIT Press, c2010. Adaptive computation and machine learning. ISBN 978-0-262-01243-0.

[4] WITTEN, I.H., E. FRANK a M. A. HALL. Data mining: practical machine Learning tools and techniques. 3rd ed. Amsterdam: Morgan Kaufmann, 2011. Morgan Kaufman series in data management systems. ISBN 978-0-12-374856-0.

[5] BISHOP, CH. M. *Pattern Recognition and Machine Learning*. Upravené šesté vydání. New York: Springer Science + Business Media, 2007. ISBN 978-0-387-31073-2.

[6] LAMPERT, Ch. H. a D. BLATNÁ. *Metody statistické analýzy*. Praha: Bankovní institut vysoká škola, 2009. ISBN 978-80-7265-143-6.

[7] HASTIE, T., R. TIBSHIRANI a J. FRIEDMAN. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second Edition. New York: Springer, 2017. ISBN 978-0-387-84857-0.

[8] MONTGOMERY, D. C., E. A. PECK a G. G. VINING. *Introduction to Linear Regression Analysis*. 5th ed. Hoboken, NJ: Wiley, 2012. ISBN 978-0-470-54281-1.

[9] RUMSEY, D. *Intermediate Statistics For Dummies*. Hoboken, NJ: Wiley Publishing, 2007. ISBN 978-0-470-04526-6.

[10] KUMAR, A. P*ython / Implementation of Polynomial Regression* [online]. [cit. 2019-03-22]. Dostupné z: https://www.geeksforgeeks.org/python-implementation-of-polynomial-regression/.

[11] VAPNIK, V. *The Nature of Statistical Learning Theory*. Springer: New York, 1995.

[12] *Introduction to Support Vector Machines*. OpenCV [online]. 2014 [cit. 2019-02-13]. Dostupné z: https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html.

[13] *Understanding Support Vector Machine Regression* [online]. 2019 [cit. 2019-03-24]. Dostupné z: https://www.mathworks.com/help/stats/understanding-support-vector-machine-regression.html.

[14] *Support Vector Machine - Regression (SVR)* [online]. [cit. 2019-03-24]. Dostupné z: https://www.saedsayad.com/support_vector_machine_reg.htm.

[15] *Support Vector Machine Regression* [online]. [cit. 2019-03-24]. Dostupné z: http://kernelsvm.tripod.com/.

[16] ROKACH, L. a O. MAIMON. *Data Mining with Decision Trees: Theory and Applications*. Second edition. Hackensack, New Jersey: World Scientific, 2015. ISBN 978-9814590075.

[17] GRĄBCZEWSKI, K. Meta-learning in Decision Tree Induction. New York: Springer, [2014]. *Studies in Computational Intelligence*, v. 498. ISBN 978-3-319-00959-9.

[18] SULLIWAN, W. *Machine learning Beginners Guide Algorithms: Supervised & Unsupervised learning, Decision Tree & Random Forest Introduction*. 1. CreateSpace Independent Publishing Platform, 2017. ISBN 978-1975632328.

[19] MAIMON, O. a L. ROKACH. D*ata Mining and Knowledge Discovery Handbook*. Second edition. New York: Springer, 2010. ISBN 978-0-398-09283-4.

[20] ERICSON, G. a et al. *Decision Forest Regression* [online]. 16. leden 2018 [cit. 2019-03-25]. Dostupné z: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/decision-forest-regression.

[21] WEST, P.W. *Tree and Forest Measurement*. 2nd edition. New York: Springer, c2990. ISBN 978-3-540-95965-6.

[22] OLEJ, V. a P. HÁJEK. *Úvod do umělé inteligence: Moderní přístupy*. Pardubice: Univerzita Pardubice, 2010.

[23] VESELÝ, A. *Metody umělé inteligence*. Praha: Česká zemědělská univerzita v Praze, 2012. ISBN 978-80-213-2295-0.

[24] HAMMER, M. *Metody umělé inteligence v diagnostice elektrických strojů*. Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-231-2.

[25] VAŘACHA, P. *Neural Network Synthesis*. Zlín: Tomas Bata University in Zlín, 2011.

[26] LOY, J. *Neural Network Projects with Python*. Birmingham: Packt Publishing, 2019. ISBN 978-1-78913-890-0.

[27] KOTU, V. *Data Science*. Waltham, MA: Elsevier, 2019. ISBN 978-0-12-814761-0.

[28] STAT 508 - *Applied Data Mining and Statistical Learning*. PennState Eberly College of Science [online]. Pennsylvania: The Pennsylvania State University, 2019 [cit. 2019-04-23]. Dostupné z: https://newonlinecourses.science.psu.edu/stat508/lesson/5/5.4.

[29] ElasticNet. *Scikit-learn* [online]. 2018 [cit. 2019-04-23]. Dostupné z: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html.

[30] CHEN, T. a C. GUESTRIN. *XGBoost: A scalable Tree Boosting System*. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785-794. ACM.

[31] NIELSEN, D. *Tree Boosting With XGBoost: Why does XGBoost Win „Every" Machine Learning Competition?* [online]. 2016 [cit. 2019-04-23].

[32] *XGBoost Parameters* [online]. [cit. 2019-03-28]. Dostupné z: https://xgboost.readthedocs.io/en/latest/parameter.html.

[33] BOOT, C.R.L. a A. DRONGELEN. Prediction of Long-Term and Frequent Sickness Absence using Company Data. *Occupational Medicine*. 2017, 3(67), 176-181.

[34] KOOPMANS, P.C., ROELEN, C.A. a J.W. GROOTHOFF. Risk of Future Sickness Absence in Frequent and Long-Term Absentees. *Occupational Medicine*. 2008, 58, 268-274.

[35] ROELEN, C.A. et al. The History of Registered Sickness Absence Predicts Future Sickness Absence. *Occupational Medicine*. 2011, 61, 96-101.

[36] DEKKERS-SÁNCHEZ, P.M. et al. Factors Associated with Long-Term Sick Leave in Sick-Listed Employees: A Systematic Review. *Occupational Environmental Medicine*. 2008, 65, 153-157.

[37] AIRAKSINEN, J., JOKELA, M. a M. VIRTANEN. Prediction of Long-Term Absence due to Sickness in Employees: Development and Validation of a Multifactorial Risk Score in two Cohort Studies. *Scandinavian Journal of Work, Environment and Health. Helsinki*. 2018, 44(3), 274-282.

[38] *Historická data - meteorologie a klimatologie*. Portál Českého hydrometeorologického ústavu [online].2018[cit. 2019-2-10]. Dostupné z: http://portal.chmi.cz/historicka-data/pocasi/denni-data.

[39] *Seaborn: statistical data visualization*. Seaborn [online]. Michael Waskom, 2018 [cit. 2019-04-21]. Dostupné z: https://seaborn.pydata.org/index.html.

[40] LINOFF, G.S. a M.J. BERRY. *Data Mining Techniques for Marketing, Sales, and Customer Relationship Management*. 3. vyd. Indianapolis, Wiley Pub. Inc., 2011. 847 s. ISBN 978-0-470-65093-6.

[41] *Scikit-learn GitHub*. GitHub [online]. 2019 [cit. 2019-04-21]. Dostupné z: https://github.com/scikit-learn/scikit-learn.

[42] *LabelEncoder*. Scikit-learn [online]. 2018 [cit. 2019-04-21]. Dostupné z: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

[43] *Random Search for Hyper-Parameter Optimization*. Journal of Machine Learning Research. Montréal, Canada, 2012, 2012(13), 282-305.

[44] *Comparing randomized search and grid search for hyperparameter estimation. Scikit-learn* [online]. 2018 [cit. 2019-04-21]. Dostupné z:https://scikit-learn.org /stable/auto_examples/model_selection/plot_randomized_search.html.