

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA JADERNÁ A FYZIKÁLNĚ INŽENÝRSKÁ

Katedra softwarového inženýrství v ekonomii
Obor: Inženýrská informatika
Zaměření: Softwarové inženýrství v ekonomii



Modely zátěže výpočetních serverů

Computational requirements modelling

BAKALÁŘSKÁ PRÁCE

Vypracoval: Dmitriy Burdin
Vedoucí práce: Ing. Jan Doubek
Rok: 2015

Před svázáním místo téhle stránky

vložíte zadání práce

 s podpisem děkana a do pdf verze oskenované zadání.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v příloženém seznamu.

V Praze dne

.....
Dmitriy Burdin

Poděkování

Děkuji Ing. Janu Doubkovi za vedení mé bakalářské práce a za podnětné návrhy, které ji obohatily.

Dmitriy Burdin

Název práce:

Modely zátěže výpočetních serverů

Autor: Dmitriy Burdin

Obor: Inženýrská informatika

Druh práce: BAKALÁŘSKÁ PRÁCE

Vedoucí práce: Ing. Jan Doubek
CISCO Systems s.r.o.

Abstrakt: Cílem této práce je efektivně modelovat časovou, výpočetní a paměťovou náročnost distribuovaných úloh. Implementace modelů na základě časových řad výsledků minulých úloh. Výsledkem práce bude studie použitelnosti ekonometrických metod v prostředí velkých data center. Dále návrh algoritmů pro implementaci studovaných metod. Výstupní modely budou použity pro lepší plánování rozvržení výpočetních zdrojů.

Klíčová slova: Časová řada, ekonometrie, predikce, analýza, server

Title:

Computational requirements modelling

Author: Dmitriy Burdin

Abstract: The goal of this thesis is to effectively make a model of temporal, computational and memory requirements of distributed tasks. Model implementation based on time series results of past tasks. The result of the thesis will be a study of applicability of econometric methods in large scale data centres. In addition, design of algorithms for implementation of the studied methods. Output models will be used for better planning of computational sources arrangement.

Key words: Times series, econometrics, forecasting, analysis, server

Obsah

Úvod	7
1. Teoretická část	8
1.1. Časové řady	8
1.1.1. Úvod a charakteristika	8
1.1.2. Analýza časových řad	9
1.1.3. Klouzavé průměry	11
1.1.4. Predikce	13
1.2. Regresní analýza	15
1.2.1. Lineární regrese	15
1.2.2. Vícenásobná lineární regrese	18
2. Praktická část	21
2.1. Softwarová aplikace	21
2.1.1. Způsob realizace	21
2.1.2. Základní předpoklady a nástroje	21
2.2. Chybné pokusy	23
2.3. Implementace	25
2.3.1. Datová vrstva	25
2.3.2. Aplikační vrstva	26
2.4. Testování aplikace	28
2.4.1. Testovací data	28
2.4.2. Zpracování a analýza dat	28
2.4.3. Vizualizace	29
Závěr	31
Literatura	32
Přílohy	33
A. Uživatelská příručka	33
B. Obsah přiloženého DVD	34

Úvod

V dnešní době náš život je naplněn obrovským a rostoucím množstvím různé informace. Jednou z hlavních příčin je rostoucí počet lidí, firem a zařízení připojených k internetu. Kolem třetiny světové populace má dnes přístup k internetu. Velké množství informace, jinak řečeno - velká data, představuje velmi rychle rostoucí sféru. Pro maximálně efektivní ovládání tohoto toku informací existuje příslušné programové a technické vybavení neboli software a hardware. S výrazným zvýšením úrovně informační potřeby se začaly vyskytovat nové problémy a otázky. Jakým způsobem přenášet data rychleji a spolehliveji? Kde uchovávat informace? Jak poskytnout bezpečnější přístup k datům?

Důležitým prvkem řešení této problematiky je datacentrum. Jsou to specializované prostory pro umístění a zajištění stabilního provozu výpočetní a serverové techniky. Většinou se tato technika skládá z počítačových clusterů, což je několik spolupracujících počítačů propojených počítačovou sítí. Clustery se využívají pro výpočet komplikovaných početních úloh. Jeden z hlavních parametrů kvalitních datacenter je správné plánování distribuovaných úloh pro rychlé zpracování dat. Při stále se zvětšujícím množství dat se tento problém komplikuje. Proto je potřeba vědět v jakém pořadí zpracovávat určité úlohy a jak dlouho to bude trvat.

Cílem této bakalářské práce bylo navrhnout a vytvořit softwarovou aplikaci pro analytické zpracování dat, se kterými pracují výpočetní servery. Podstatou aplikace je predikce budoucího chování určité hodnoty na základě analýzy minulých výsledků. Určitou hodnotou pro zkoumání v této práci je prvek log souboru *Computation time*, který byl spolu se souborem poskytnut vedoucím bakalářské práce. Protože velká data obvykle obsahují dynamické systémy dat, které se mění s časem, zkoumaná data jsou většinou ve formě časových řad. Proto se tato práce v první (teoretické) části věnuje způsobům zpracování, analýzy a predikce časových řad. Potom jsou uvedeny principy lineární regrese. Závěrem první části práce je popis využití těchto znalostí v tvorbě softwarové aplikace.

Druhá (praktická) část bakalářské práce za prvé popisuje chybné postupy v realizaci postaveného cíle. Poté následuje implementace vhodné analytické metody a konstrukce aplikace. Na konci praktické části je demonstrováno využití aplikace na vzorových datech. Vlastní posouzení dosažených výsledků a jejich možná zlepšení jsou uvedeny v závěru bakalářské práce.

1. Teoretická část

1.1. Časové řady

1.1.1. Úvod a charakteristika

Jak již bylo zmíněno, časové řady jsou základním zkoumaným prvkem při analýze různých dynamických systémů obsahující chronologicky uspořádaná data. Časovou řadou rozumíme soubor pozorování, jednoznačně uspořádaných podle času v příslušném systému. Data ve formě časových řad pochází z různých odvětví vědy, například fyzikální vědy, biologie, společenské vědy, medicína a další.

Například ve společenských vědách, jsou časové řady užitečné při popsání počtu obyvatel, porodnosti nebo nemocnosti. V ekonomii je teorie časových řad jednou z nejdůležitějších metod při analýze ekonomických procesů. Časové řady mohou popisovat vývoj ukazatele například objem výroby, produktivitu práce, nezaměstnanost nebo spotřebu surovin. V technice mohou časové řady představovat průběh signálu, spolehlivost nebo intenzitu zatížení elektrického zařízení.

Pro lepší porozumění mechanismu nebo procesu, popsaného časovou řadou, existuje analýza časových řad skládající se z různých metod. Tyto metody pomáhají vytvořit vhodný model popisující chování pozorovaných hodnot. Znalost takového modelu umožňuje kontrolovat činnost a sledovat vývoj zkoumaného systému. Jako důsledek správně provedené analýzy se naskytá možnost predikovat budoucí chování systému.

Časové řady se skládají ze dvou prvků:

- časový úsek, během kterého byla provedena pozorování
- hodnoty příslušných ukazatelů časové řady

Podle časového úseku se časové řady člení do dvou typů. Jedním je okamžiková časová řada, jejíž pozorování jsou naměřena v jisté časové okamžiky. Příkladem může být řada udávající počet zaměstnanců ve firmě na začátku roku. Druhým typem je intervalová časová řada. Pozorování v intervalových časových řadách jsou závislé na délce časového intervalu sledování. Například porodnost ve státě za rok.

Časové úseky jsou obvykle stejnoměrně rozděleny, což znamená, že mezi jednotlivými pozorováními jsou stejné časové intervaly. V opačném případě jsou pozorování rozdělena různorodě a tím se analýza časových řad komplikuje.

Obvykle rozlišujeme dva základní modely časových řad:

- Deterministický - model, ve kterém časová řada nemá náhodné prvky a je generována známou matematickou funkcí. Důsledkem je srovnatelně jednoduchá analýza časové řady.
- Stochastický - model popisuje náhodný proces a časová řada obsahuje náhodný prvek. Většina běžně se vyskytujících časových řad v praxi mají stochastické modely.

Jedna z dalších důležitých charakteristik časových řad, vyplývající ze stochastického modelu, je jejich stacionarita, případně nestacionarita. Při stacionaritě se střední hodnota a rozptyl časové řady v čase nemění, což nelze říct o nestacionárních časových řadách, ve kterých se objevují změny ve střední hodnotě či rozptylu. Proto se říká, že nestacionární časové řady mají určitý trend vývoje. I když stacionarita je běžným předpokladem většiny metod analýzy časových řad, některé modely se omezují pouze na modely stacionárních procesů.

1.1.2. Analýza časových řad

Grafická analýza časových řad

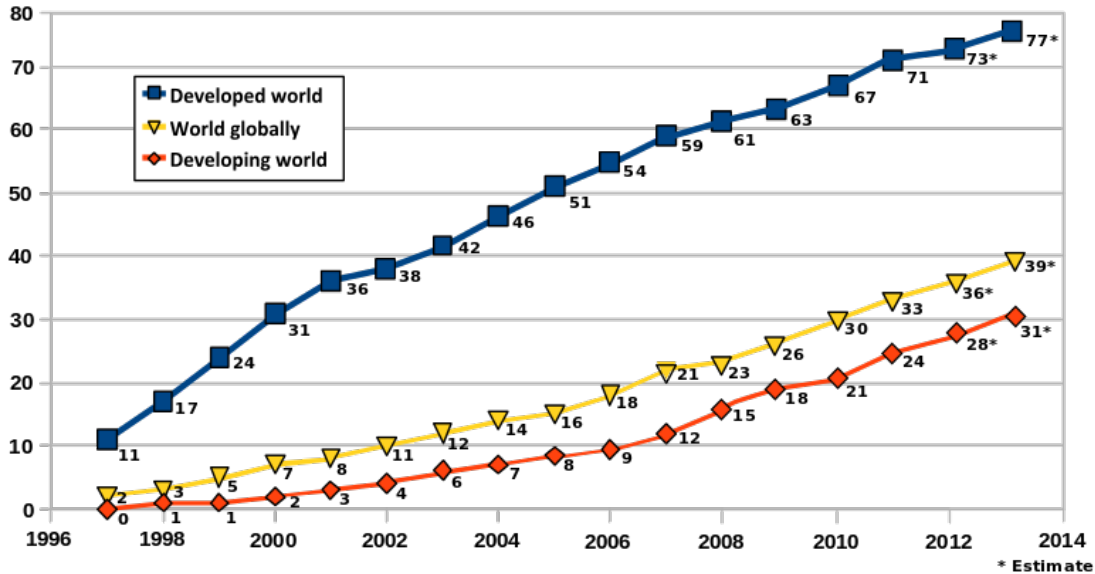
Graf časové řady je základním prostředkem její prezentace. Grafy často slouží pro určení přibližného typu časové řady a jejích složek. Proto se většinou graf vybírá tak, aby byl přehlednější a zobrazoval charakteristické vlastnosti příslušné časové řady. Pro takové účely existují určité typy grafů jako například spojnicové grafy jedné nebo více časových řad, krabičkové grafy, grafy sezónních hodnot a atd.

Základní informaci pro analýzu časových řad můžeme získat ze *spojnicových grafů*. Podstatou spojnicových grafů je zakreslení jednotlivých prvků časové řady do souřadných os. Ve výsledku je na ose horizontální časová proměnná časové řady a na ose vertikální hodnota ukazatele časové řady. Spojnicový graf může obsahovat i více časových řad. V takovém případě (např. dvě časové řady s různými měřítky) se dají současně použít dvě vertikální osy - levá a pravá.

Detailnější pohled na zkoumanou časovou řadu poskytuje *krabičkový graf*, který obsahuje podrobnější charakteristiku časové řady. Pomocí tohoto typu grafu je možné získat rozsáhlejší informace o hodnotě ukazatele v určitém časovém okamžiku. Hlavním prvkem tohoto grafu je krabička, která má dolní a horní hrany tvořené 25% a 75% kvantilem. V krabičkovém grafu jsou také zobrazeny hodnoty maxima a minima ležící na koncích svislé čary, medián a aritmetický průměr. [4]

Dekompozice časových řad

Abychom mohli s časovou řadou pracovat, předpovídat budoucí hodnoty, dělat odhady (intervalové či bodové) apod., potřebujeme mít časovou řadu, která je alespoň stacionární, respektive slabě stacionární. Většina zkoumaných časových řad je nestacionární povahy. Ale často se nám správným očištěním (dekompozicí) řady může podařit, získat stacionární řadu.



Obrázek 1.1.: Počet uživatelů Internetu na 100 obyvatel ve světě.
(Příklad spojnicového grafu se třemi časovými řadami)
(Zdroj: <<https://cs.wikipedia.org>>)

Většinou se předpokládá, že časová řada y_t pro $t = 1, 2, \dots, T$ je sestavena z trendové, cyklické, sezónní a nesystematické složky.

Trendová složka (T_t) je obecná dlouhodobá tendence vývoje zkoumaného jevu. Tato složka zachycuje stálé procesy působící během dlouhého období. Trend může mít rostoucí nebo klesající charakter a nebo může být časová řada bez trendu.

Cyklická složka (C_t) popisuje jednotlivé periody (cykly) v hodnotách ukazatele časové řady. Taková kolísání se okolo trendu objevují během dlouhého období, tj. značně delší než jeden časový úsek časové řady. V těchto cyklech se střídají fáze růstu a poklesu. Délka a amplituda cyklické složky mohou být nepravidelné a mít měnící se charakter.

Sezónní složka (S_t) vyjadřuje stále se opakující odchylku trendu. Perioda této složky je mnohem menší než u složky cyklické. Takové výkyvy se objevují ve stejných obdobích kvůli obecně známým jevům.

Čtvrtou složkou časové řady je *náhodná* neboli *nesystematická složka* (u_t). Náhodné procesy a chyby měření v datech jsou vyjádřeny pomocí této složky. Náhodná složka má vlastnosti procesu bílého šumu, a proto pro hodnoty u_t platí

- střední hodnota v čase t je nulová: $E(u_t) = 0$
- konstantní rozptyl: $D(u_t) = \sigma^2$
- vzájemně lineární nezávislost: $cov(u_t, u_{t-k}) = 0$
- normální rozdělení: $u_t \sim N(0, \sigma^2)$ ¹

¹Bílý šum může mít i jiný typ rozdělení. Vžak normální rozdělení je nejčastějším případem.

Existují dva typy dekompozice časových řad:

1. aditivní: hodnoty ukazatele časové řady jsou součtem hodnot jednotlivých složek

$$y_t = T_t + C_t + S_t + u_t \quad (1.1)$$

2. multiplikativní: hodnoty ukazatele časové řady jsou součinem hodnot jednotlivých složek

$$y_t = T_t \cdot C_t \cdot S_t \cdot u_t \quad (1.2)$$

Při aditivní dekompozici časové řady jsou jednotlivé složky uvažovány ve stejných jednotkách jako hodnoty ukazatele příslušné časové řady. Předpokladem použití aditivní dekompozice je konstatní variabilita hodnot časové řady v čase.

V případě multiplikativní dekompozice má trendová složka stejné jednotky jako příslušná časová řada, ale ostatní složky zůstávají vyjádřeny relativně. Při takové dekompozici se variabilita časové řady v čase obvykle mění. Existuje několik důvodů pro využití dekompozice a odstranění jednotlivých složek časových řad:

- a) charakter vývoje, chování zkoumané řady lze odhalit z analýzy příslušných složek
- b) sezónním očišťováním je možné odstranit výkyvy časové řady pro přehlednější zkoumání trendu
- c) a nebo se dá odstranit trendová složka časové řady pro detailnější analýzu sezónnosti
- d) dekompozice umožňuje provést předpověď jednotlivých složek, což vede k lepšímu výsledku v předpovídání samotné časové řady; předpovědi složek se v takovém případě sečtou nebo vynásobí v závislosti na typu dekompozice

1.1.3. Klouzavé průměry

Při analýze časové řady je velmi důležité zkoumat jestli má časová řada trend nebo nemá. Metoda klouzavých průměrů je jednou z adaptivních metod pro modelování trendové složky časové řady. Adaptivní přístup je založen na tom, že pomocí něho můžeme pracovat s trendovou složkou, která se mění v čase a kterou nelze popsat matematickou křivkou s neměnnými parametry. Však můžeme předpokládat, že v malých úsecích časové řady existuje možnost takové vyrovnaní matematickou křivkou použít. [2]

Základní myšlenkou metody klouzavých průměrů je určení hodnot vyhlazené časové řady pomocí vypočítaných průměrů hodnot původní časové řady. Počet hodnot, pro které vypočítáme průměr, určujeme pomocí tzv. *okna*.

Okno (délka klouzavé části) je určena počtem sezón v sezónních časových řadách. Pro ostatní časové řady musíme okno volit pro každý případ zvlášť. Obecně čím je okno větší (delší klouzavá část), tím je vyrovnaná časová řada hladší. [4]

Existují různé typy metody klouzavých průměrů. Nejčastějšími typy jsou *jednoduchý klouzavý průměr* (angl. Simple Moving Average - SMA), *lineárně vážený*

klouzavý průměr (angl. Linear Weighted Moving Average - LWMA) a *exponenciální klouzavý průměr* (angl. Exponential Moving Average). Tyto metody jsou podrobně porbrány v [4] a [6]. V této práci ale budou popsány pouze jejich využití pro předpovídání časových řad.

Jednoduchý klouzavý průměr

Jednoduchý neboli aritmetický klouzavý průměr je nejjednodušším a méně spolehlivějším případem, kdy pro vyhlazení nebo předpověď časové řady vypočítáme aritmetický průměr hodnot z určeného okna. Pro předpověď zkoumané hodnoty y_t v čase $t + 1$ můžeme použít výraz

$$\hat{y}_{t+1} = \frac{y_t + y_{t-1} + \dots + y_{t-n+1}}{n}, \quad (1.3)$$

kde n je velikost okna.

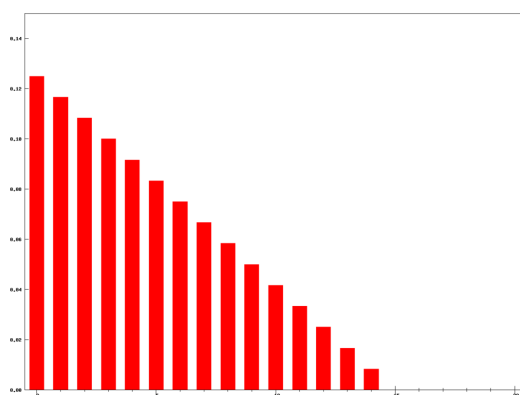
Lineárně vážený klouzavý průměr

Lineárně vážený klouzavý průměr se od jednoduchého liší tím, že jsou jeho hodnotam přiřazeny váhy, které jsou lineárně uspořádány podle velikosti. To znamená, že při výpočtu lineárně váženého klouzavého průměru předpokládáme, že poslední (nejnovější) hodnoty časové řady jsou významnější a mají větší vliv, než hodnoty předešlé. Predikovanou hodnotu časové řady y_t v čase $t+1$ můžeme vypočítat pomocí následujícího výrazu

$$\hat{y}_{t+1} = \frac{2}{n(n+1)} \sum_{i=0}^{n-1} (n-i)y_{t-i}, \quad (1.4)$$

kde n je velikost okna a zároveň musí platit, že součet všech váh je roven jedné

$$\frac{2}{n(n+1)} \sum_{i=0}^{n-1} (n-i) = 1 \quad (1.5)$$



Obrázek 1.2.: Graf ukazující váhy v případě velikosti okna $n = 15$ pro LWMA.
(Zdroj: <<https://cs.wikipedia.org>>)

Exponenciální klouzavý průměr

Při výpočtu exponenciálního klouzavého průměru se využívají váhy, jejichž hodnoty se zmenšují exponenciálně. Obecný výraz pro tento typ klouzavých průměrů je následující

$$\hat{y}_t = \alpha y_t + (1 - \alpha)\hat{y}_{t-1}, \quad (1.6)$$

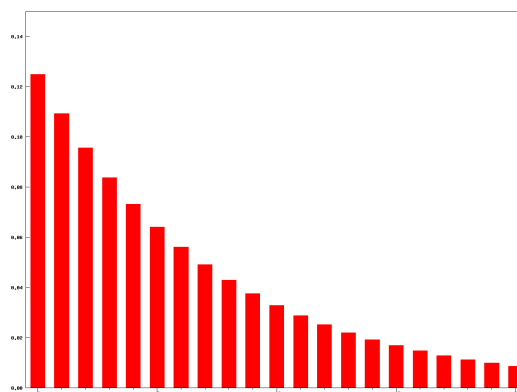
kde \hat{y}_{t-1} je hodnota exponenciálního klouzavého průměru v časový okamžik $t-1$, což znamená, že tento výraz je rekursivní. Číslo α je vyrovnávací konstanta, která je nejčastěji určena pomocí velikosti zkoumaného "okna" dat

$$\alpha = \frac{2}{n+1}, \quad (1.7)$$

kde n je velikost okna.

Na základě rekurentnosti může být výraz pro exponenciální průměr přepsán a pro predikovanou hodnotu tehdy platí

$$\hat{y}_{t+1} = \alpha \sum_{i=0}^{n-2} (1 - \alpha)^i y_{t-i} + (1 - \alpha)^{n-1} y_{t-n}, \quad (1.8)$$



Obrázek 1.3.: Graf ukazující váhy v případě velikosti okna $n = 15$ pro EMA.

(Zdroj: <<https://cs.wikipedia.org>>)

1.1.4. Predikce

Protože je tato práce zaměřena na predikci časových řad, je konstrukce předpovědí jednou z nejdůležitějších částí. Vhodná konstrukce předpovědí má velký význam pro dosažení lepších výsledků. V praktické části bude použit určitý model predikce, ale ještě předtím se v této kapitole zmíníme o některých jiných obecných aspektech, které jsou spojeny s předpovídáním časových řad.

Bodová předpověď je hodnota představující nejpřesnější odhad budoucí hodnoty časové řady v jistém okamžiku. Tato hodnota je vždy zatížena určitou chybou,

jejíž velikost záleží na vybraném modelu predikce. Pro úroveň přesnosti existuje pojem *předpovědní interval*, který je analogií intervalu spolehlivosti v matematické statistice. Předpovědní interval tak určuje horní a dolní mez, mezi nimiž s určitou pravděpodobností leží predikovaná budoucí hodnota.

Předpovědní metody se většinou rozdělují do dvou velkých skupin - *kvalitativní* a *kvantitativní*. Základem kvalitativních metod jsou názory expertů a odborníků v určité oblasti. Takové metody jsou použitelné v případě, kdy se zavádí nová technologie nebo zcela nový produkt a zatím neexistují žádná data. Dále jsou v práci prozkoumány a použity pouze kvantitativní předpovědní metody. Tyto metody jsou oproti kvalitativním založeny na analýze empirických dat. Analýza se provádí pomocí matematicko-statistického přístupu. Použití kvantitativních předpovědních metod předpokládá, že se s časem, kdy se provádí předpověď, charakter řady nemění. Tento fakt nemůžeme vynechávat a proto budeme zavádět drobná omezení v konstrukci předpovědní metody (např. rozdělení dat do intervalů s maximálně stejným vývojem). [2]

Chyba předpovědi

V případě, že existují předem známá data a předpovědní metoda se určuje na základě těchto dat, hrají chyby v předpovědích a jejich minimalizace důležitou roli. Chyby (viz též rezidua) jsou většinou definovány jako rozdíl mezi reálnou hodnotou y_t a predikovanou hodnotou \hat{y}_t , a zároveň jsou odhadem náhodné složky u_t :

$$e_t = y_t - \hat{y}_t \quad (1.9)$$

Existují určité míry pro kontrolu kvality provedených předpovědí. Často se můžeme setkat s následujícími typy:

Součet čtvercových chyb SSE (Sum of Squared Errors):

$$\sum_{t=1}^n (y_t - \hat{y}_t)^2 = \sum_{t=1}^n (e_t)^2 \quad (1.10)$$

Střední čtvercová chyba MSE (Mean Squared Error)

$$\sum_{t=1}^n \frac{(y_t - \hat{y}_t)^2}{n} = \sum_{t=1}^n \frac{(e_t)^2}{n} \quad (1.11)$$

Střední absolutní odchylka MAD (Mean Absolute Deviation)

$$\sum_{t=1}^n \frac{|y_t - \hat{y}_t|}{n} = \sum_{t=1}^n \frac{|e_t|}{n} \quad (1.12)$$

1.2. Regresní analýza

Regresní analýza je souhrnem statistických metod, používaných k matematickému popisu vztahu a závislosti jedné veličiny (závisle proměnná, vysvětlovaná proměnná neboli regresand) na jiné veličině (nezávisle proměnná, vysvětlující proměnná neboli regresor). Výsledkem regresní analýzy je informace o tom, jak se mění hodnota vysvětlované proměnné při změnách hodnoty vysvětlující proměnné. Tuto informaci pak můžeme využít pro predikci hodnoty vysvětlované proměnné pomocí regresní funkce.

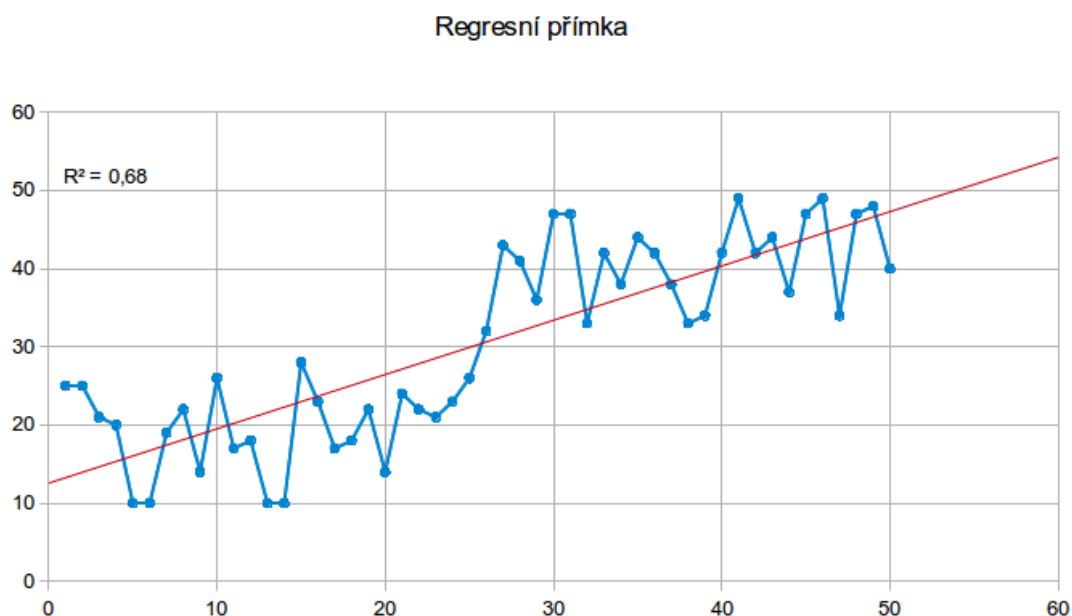
Regresní funkce se často používá pro odhad trendu časové řady, proto dále analýza regresní funkce a její parametrů bude považována za analýzu trendové funkce časové řady.

1.2.1. Lineární regrese

Pokud je regresní funkce lineární vzhledem k regresním koeficientům a vyjádřena vztahem

$$\hat{y} = \sum_{j=1}^k b_j f_j(x), \quad (1.13)$$

hovoříme o lineární regresní funkci, kde b_j jsou regresní koeficienty a $f_j(x)$ jsou předem známé funkce. Pomocí lineární regrese můžeme proložit soubor bodů v grafu přímkou či křivkou, v závislosti na typu vysvětlujících proměnných x .



Obrázek 1.4.: Příklad proložení bodů v grafu regresní přímkou.
(R^2 - koeficient determinace. Vlastní zdroj.)

V případě přímky (viz Obrázek 1.2) lineární regresní funkce má následující tvar

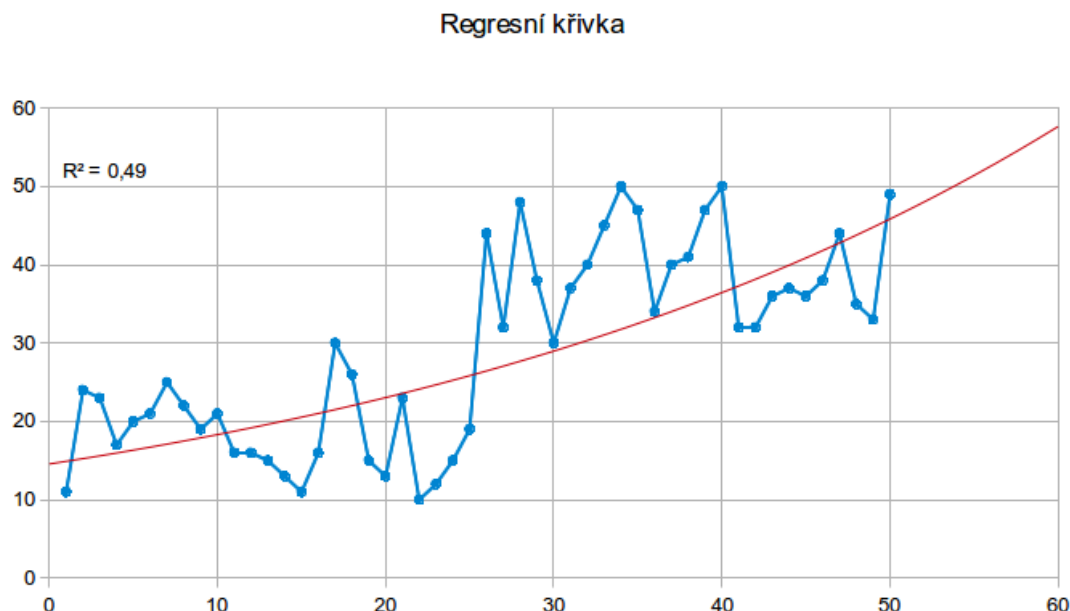
$$\hat{y} = b_0 + b_1x, \quad (1.14)$$

kde b_0 , b_1 jsou regresní koeficienty a x je jedinou vysvětlující proměnnou. Regresní funkce je odhadem příslušného regresního modelu

$$y = \beta_0 + \beta_1x + e, \quad (1.15)$$

kde e je chyba mezi reálnou hodnotou y a predikovanou hodnotou \hat{y} , pro kterou se předpokládá, že její střední hodnota je rovna nule, tj. $E(e) = 0$.

V komplikovanějším případě regresní funkce může být například mocninnou funkcí $\hat{y} = b_0 \cdot x^{b_1}$ nebo exponenciální $\hat{y} = b_0 \cdot b_1^x$. Tyto komplikace se řeší zlogaritmováním příslušné rovnice. Zlogaritmováním pak dostáváme obě rovnice ve tvaru podobném rovnici (1.8), který můžeme zkoumat jako regresní funkci ve tvaru přímky.



Obrázek 1.5.: Příklad proložení bodů v grafu regresní exponenciální křivkou. (R^2 - koeficient determinace. Vlastní zdroj.)

Dalším zvláštním příkladem lineární regresní funkce je polynomická funkce. Úkolem se stává určit nejvhodnější regresní koeficienty polynomu stupně k , který nejlépe aproximuje zadané body

$$\hat{y} = P_k(x) = p_0 + p_1x + \dots + p_kx^k, \quad (1.16)$$

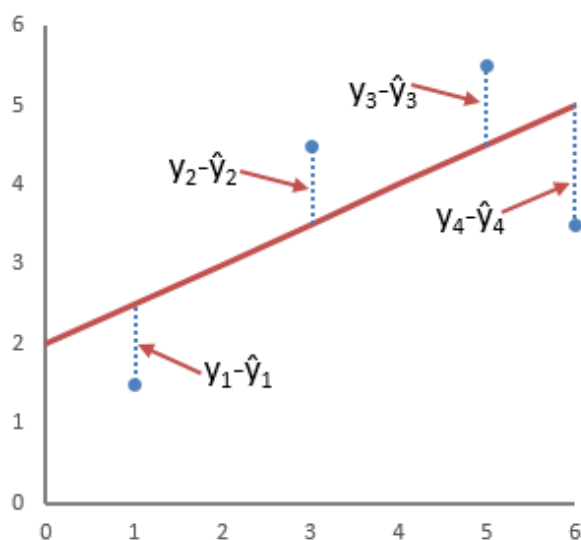
kde koeficienty polynomu p_0 , p_1 ... p_k jsou koeficienty regresní polynomické funkce.

Je ale pochopitelně, že přímka nebo křivka nemůže přesně procházet všemi datovými body v grafu. Zatímco jsou x-ové souřadnice přesné, y-ové souřadnice regresní funkce obvykle zatíženy chybou (1.3). Proto potřebujeme využít metodu, pomocí které dokážeme určit neoptimálnější hodnoty regresních koeficientů.

Metoda nejmenších čtverců

Pro jednoznačnost výběru regresní funkce se nejčastěji používá *metoda nejmenších čtverců* (OLS - Ordinary least squares), pro kterou platí, že součet čtverců odchylek je minimální

$$\sum_{t=1}^n (y_t - \hat{y}_t)^2 = \sum_{t=1}^n (e_t)^2 \rightarrow \min. \quad (1.17)$$



Obrázek 1.6.: Grafické znázornění odchylek. Zdroj: <<http://exceltip.ru/>>

Ukažeme využití MNČ na výpočtu regresních koeficientů regresní funkce ve tvaru přímky, tj. ve tvaru (1.8). Minimalizujeme součet čtverců odchylek

$$\sum_{t=1}^n (y_t - \hat{y}_t)^2 = \sum_{t=1}^n (y_t - b_0 - b_1 x)^2 \rightarrow \min. \quad (1.18)$$

Přirovnáme parciální derivace podle parametrů b_0 a b_1 k nule a po úpravách získáme dvě rovnice

$$\sum_{t=1}^n y_t = n b_0 + b_1 \sum_{t=1}^n x_t \quad (1.19)$$

$$\sum_{t=1}^n y_t x_t = b_0 \sum_{t=1}^n x_t + b_1 \sum_{t=1}^n (x_t)^2 \quad (1.20)$$

Řešením těchto rovnic a dosazením daných hodnot x_t a y_t , dostaneme hodnoty regresních koeficientů přímky

$$b_0 = \frac{\sum_{t=1}^n y_t \sum_{t=1}^n (x_t)^2 - \sum_{t=1}^n x_t \sum_{t=1}^n y_t x_t}{n \sum_{t=1}^n (x_t)^2 - (\sum_{t=1}^n x_t)^2} \quad (1.21)$$

$$b_1 = \frac{n \sum_{t=1}^n y_t x_t - \sum_{t=1}^n y_t \sum_{t=1}^n x_t}{n \sum_{t=1}^n (x_t)^2 - (\sum_{t=1}^n x_t)^2}, \quad (1.22)$$

kde n je počet naměřených hodnot neboli pozorování.

1.2.2. Vícenásobná lineární regrese

Sledování závislosti a vztahů mezi pouze dvěma proměnnými je v praxi velmi častým zjednodušením skutečnosti. Pro modely, které mají více než jednu nezávislou (vysvětlující) proměnnou, se využívá vícenásobná lineární regrese. Závislost vysvětlované proměnné v takovém případě je možno popsat regresní rovnicí

$$\hat{y}_t(x) = b_0 + b_1 x_{t1} + b_2 x_{t2} + \dots + b_k x_{tk}, \quad (1.23)$$

kde index t značí jednotlivá pozorování a k je počet regresorů (vysvětlujících proměnných). Daný model má jeden důležitý předpoklad, jehož podstatou je to, že libovolnou z nezávislých proměnných nemůžeme vyjádřit jako lineární kombinaci ostatních nezávislých proměnných. Jinak bychom nemohli jednoznačně určit regresní koeficienty, protože by mohlo existovat mnoho kombinací regresních koeficientů, které by vysvětlovaly veličinu y se stejnou přesností. Tento problém se nazývá *multikolinearita*, který se vyskytuje v důsledku korelace mezi vysvětlujícími proměnnými. V praxi se multikolinearita objevuje skoro v každém modelu.

Pro vícenásobnou regresní analýzu platí stejná metoda nejmenších čtverců, kterou taky můžeme popsat v maticovém tvaru

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1k} \\ 1 & x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}$$

neboli

$$\mathbf{y} = \mathbf{X}\mathbf{b} \quad (1.24)$$

Po stejných úpravách pomocí parciálních derivací získáváme rovnici pro odhad vektoru regresních parametrů

$$\mathbf{b} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (1.25)$$

Jak již bylo řečeno, multikolinearita se do jisté míry vyskytuje v různých regresních modelech. Multikolinearita obecně ukazuje na závislost mezi vysvětlujícími proměnnými. Příčiny vyskytu mohou být následující

- přeurčený regresní model
- pokud je matice vysvětlujících proměnných \mathbf{X} singulární
- špatná specifikace modelu

Důsledky multikolinearity

- velké standardní chyby
- silná citlivost regresních koeficientů i na malé změny v datech
- odhádnuté regresní parametry mohou být statisticky nevýznamné

V praxi můžeme multikolinearitu objevit z korelační matice, která je důležitým prvkem v korelační analýze. Míru závislosti obvykle popisujeme korelačním koeficientem. Nejčastěji se používá párový (Pearsanův) korelační koeficient, který vyjádřen následujícím vztahem

$$\mathbf{R} = \frac{\sum_{t=1}^n (\mathbf{x}_t - \bar{\mathbf{x}})(\mathbf{y}_t - \bar{\mathbf{y}})}{\sqrt{\sum_{t=1}^n (\mathbf{x}_t - \bar{\mathbf{x}})^2 \sum_{t=1}^n (\mathbf{y}_t - \bar{\mathbf{y}})^2}}, \quad (1.26)$$

kde jsou \bar{x} a \bar{y} příslušné střední hodnoty vysvětlující a vysvětlované proměnné.

Hodnota korelačního koeficientu \mathbf{R} je mezi -1 a +1. Orientačně pokud je $\mathbf{R} > 0.8$, lineární závislost se považuje za silnou.

Existují některé možnosti odstranění multikolinearity

- pro případ s přeurčeným regresním modelem můžeme vyřadit zbytečné vysvětlující proměnné, které určíme podle korelační matice
- zvětšení počtu pozorování
- transformace pozorování - první difference, podíl proměnných

Vhodnost modelu

Existuje hodně různých metod, pomocí kterých můžeme sestavit model časové řady. Tyto metody se dají taky kombinovat. Proto je vždy mnoho možností, jakým způsobem zkoumat časovou řadu. V závislosti na vybrané metodě, můžeme dosáhnout různých výsledků. Posouzení těchto výsledků je důležitým procesem při určení vhodnosti modelu pomocí různých analytických mír.

Rozptyl neboli *střední kvadratická odchylka* je ukazatelem variability náhodné veličiny okolo její střední hodnoty. Z reálných empirických hodnot a vypočtených hodnot regresní funkce můžeme určit tři typy rozptylů

- rozptyl empirických hodnot y

$$s_y^2 = \frac{1}{n} \sum_{t=1}^n (y_t - \bar{y})^2, \quad (1.27)$$

tj. celkový součet čtverců (CSČ).

- rozptyl vyrovnaných hodnot

$$s_{\hat{y}}^2 = \frac{1}{n} \sum_{t=1}^n (\hat{y}_t - \bar{y})^2, \quad (1.28)$$

tj. vysvětlený součet čtverců (VSČ).

- reziduální rozptyl

$$s_{y-\hat{y}}^2 = \frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2, \quad (1.29)$$

tj. nevysvětlený součet čtverců (NSČ).

Významným ukazatelem vhodnosti regresního modelu je koeficient determinace \mathbf{R}^2 , který určíme jako podíl vysvětleného součtu čtverců na celkovém součtu čtverců

$$\mathbf{R}^2 = \frac{\text{VSČ}}{\text{CSČ}} = 1 - \frac{\text{NSČ}}{\text{CSČ}} = 1 - \frac{e^T e}{y^T y - n\bar{y}^2} \quad (1.30)$$

a zároveň je druhou mocninou korelačního koeficientu \mathbf{R} .

Koeficient determinace je vhodný ke statistickému testování modelu jako celku a nabývá hodnot z intervalu $<0, 1>$, kde případ $\mathbf{R}^2=1$ ukazuje na silnou závislost a vhodnost použití lineární regrese. [3]

2. Praktická část

2.1. Softwarová aplikace

2.1.1. Způsob realizace

Po důkladném seznámení se s problematikou práce a s ní související teorií, byl stanoven způsob dosažení cílů práce, tj. způsob její praktické realizace. Tato realizace měla tři hlavní kroky:

1. Pomocí regresní a korelační analýzy zjistit určité parametry v datech, které nejméně nebo nejvíce ovlivňují závislé parametry, a jestli mezi sebou vůbec mají nějakou souvislost.
2. Potom na základě těchto údajů, s uvážením výsledků regresní a korelační analýzy získat regresní model, který může být použitelný k predikci budoucích hodnot zkoumaného parametru. V daném případě jde o daný parametr *Computation time* (z angl. doba trvání výpočtu), který vyvolává zatížení, jehož snížení je vlastně problematikou práce.
3. Posledním krokem je posouzení kvality výsledků.

Pro konstrukci aplikace je dán soubor dat skládající se z matice vysvětlujících promenných \mathbf{X} a vektoru vysvětlovaných promenných \mathbf{y} . Protože je vysvětlovaná proměnná závislá na více než jedné vysvětlující proměnné, potřebujeme využít vícenásobnou lineární regresi. Pomocí korelační analýzy nejdříve zjistíme vzájemnou závislost mezi vysvětlujícími proměnnými. Na základě dosažených výsledků dokážeme říct, které jsou nezávisle proměnné pro nás významné nebo zbytečné. Uvidíme to z korelační matice, která popisuje závislosti mezi všemi proměnnými v modelu.

Zároveň použijme metodu výběru nejlepší podmnožiny pro určení vysvětlujících parametrů s nejlepším koeficientem determinace. Metoda spočívá v tom, že z množiny všech vysvětlujících proměnných $\{x_1, x_2, \dots, x_k\}$ se vybírají kombinace (podmnožiny), pro které se vypočítá koeficient determinace. Podmnožina s nejlepším koeficientem determinace se pak bude použita pro odhad regresního modelu. Pomocí regresní analýzy pak dokážeme získat hodnoty regresních koeficientu, které mají větší vliv na chování závisle proměnné.

2.1.2. Základní předpoklady a nástroje

Za výsledek této práce může být také považována softwarová aplikace, která by měla být užitečnou pomůckou pro dosažení cíle práce. Podstatou aplikace je načíst a zanalizovat datový soubor regresní metodou, poté použít dosažené výsledky pro

získání regresního modelu a případně udělat predikci budoucích hodnot a vykreslit vývoj nejlépe predikovaných hodnot ve tvaru grafu časových řad.

Jako programovací nástroj jsem vybral objektově orientovaný programovací jazyk *Java*. Jedná se o populární a docela starý programovací jazyk jehož historie sahá do roku 1990, kdy ve firmě Sun Microsystems začali pracovat na jeho vytvoření. Na začátku to byl efektivní a jednoduchý jazyk určený pro spotřební elektroniku. S postupem času a s rostoucím užitím internetu se Java stala jedním z nejpoužívanějších programovacích jazyků ve světě. Hlavními důvody proč jsem zvolil právě tento jazyk jsou:

- výkonnost a jednoduchost syntaxe
- široká použitelnost
- přenositelnost a nezávislost na architektuře nebo na operačním systému

Pro snadnější programátorskou práci se hojně využívá *vývojové prostředí* neboli *IDE* (*IDE* - *Integrated Development Environment*). Je to soubor nástrojů pro přehlednější přístup ke zdrojovému kódu, ladění programů, hledání a opravu chyb a další užitečné věci pro programování. Vývojové prostředí se většinou skládá z textového editoru, kompilátoru a debuggeru.

S programovacím jazykem Java se používá několik různých IDE. Liší se podle nabízených vlastností a možností. Pro svou práci jsem zvolil vývojové prostředí *IntelliJ IDEA*.

Pro lepší koordinaci se svým vedoucím jsem během práce použil webový nástroj *GitHub*. Je to služba usnadňující spolupraci vývojářů, kteří používají verzovací nástroj Git. Využívá se pro sdílení a společnou práci na softwarových projektech. Tato služba poskytuje možnost sdílet a upravovat projekty jiných programátorů, což vede k lepším výsledkům a rychlejšímu ladění kódu.

Jako nástroj pro správu, řízení a automatizaci tvorby aplikace jsem použil *Apache*. I když je možné využít tento nástroj k psaní projektů v různých programovacích jazycích, podporován je převážně programovací jazyk Java. Maven se používá hlavně pro usnadnění práce při buildování programů a aplikací. Vývojové prostředí IntelliJ IDEA umožňuje pomocí nástroje Maven vytvořit programátorský projekt podle modelu *Project Object Model*. Tento model určuje ze začátku kostru zdrojového kódu, závislosti na externích knihovnách a proces spouštění potřebných testů a podobně. Základ samotného nástroje Maven jsou pluginy, které běží pouze na příkazové řádce.

K implementaci analytických metod bylo navrženo použít již existující knihovnu v jazyce Java, nebo si vytvořit vlastní.

2.2. Chybné pokusy

Při výběru programovacích nástrojů byla uvažována možnost využití cizích, svobodně distribuovaných knihoven nebo frameworků vhodných pro práci s daty a případně pro jejich analýzu. Předpokladem pro pomocné nástroje byla jejich kompatibilita s programovacím jazykem Java.

Knihovna je v programování soubor funkcí, procedur, datových typů a tříd, které slouží pro různé programy. Každá knihovna má své *API* (z angl. *application programming interface* - rozhraní pro programování aplikací). Pomocí kterého se určuje jakým způsobem jsou funkce z knihovny volány. Zatímco framework je softwarová struktura, která volá samotný kód uživatele. Framework může obsahovat vlastní procedury, podprogramy i knihovny. Framework nás ovšem může omezovat svoji pevnou strukturou a architekturou.

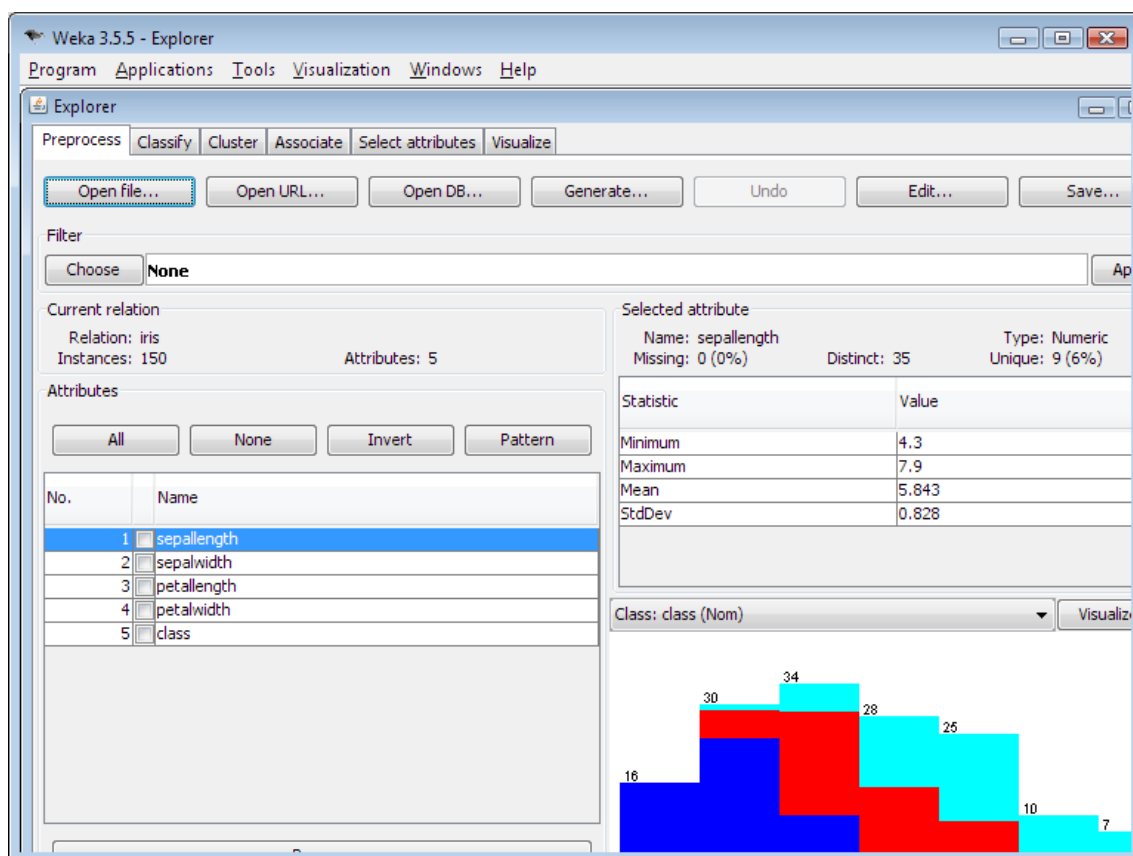
Během hledání pomocných nástrojů a vhodných knihoven jsem našel dva zdroje, u kterých jsem předpokládal že by mohly zlepšit průběh práce. Po hlubším prozkoumání se ukázalo že nejsou zas až tak vhodné pro náš projekt. Jednalo se především o velkou sbírku algoritmů pro práci s daty *Weka* a framework *Encog*.

Weka

Weka je populární balík programů, který slouží pro analýzu dat, modelování predikce a využívá se ve strojovém učení. Weka používá textové soubory s formátem ARFF (*Attribute Relationship File Format*), který obsahuje atributy a jejich data. Weka má i grafické uživatelské rozhraní pro přehledný přístup k využití všech možností programu, což je velkou výhodou. Zaujalo mě taky to, že pomocí tohoto balíku by šlo snadno vizualizovat výsledky analýzy. Na to má Weka vhodné nástroje a proto může vykreslit graf časové řady s veškerými informacemi o jejím vývoji.

Tato sbírka nástrojů pro analýzu dat je napsaná v jazyce Java a proto jsem se rozhodl ji zkusit zapojit do programu a využít její možnosti. Hned zkrájím jsem narazil na problém při konverzi CSV souboru do formátu ARFF, při kterém se vyskytovaly chyby v datech.

Po několika pokusech odstranit tyto chyby jsem zjistil, že se zdrojový kód stává komplikovanějším a využití tohoto způsobu už nedává moc smysl. Proto hlavní důvod proč jsem odmítl balík programů Weka, je nekompatibilita se strukturou kódu mého programu a ztěžování při konverzi a načítání datového souboru.



Obrázek 2.1.: Datový soubor otevřený v uživatelském rozhraní Weka
(Zdroj: <https://cs.wikipedia.org>)

Encog

Encog je framework pro strojové učení (z angl. *Machine Learning*), což je oblast zabývající se algoritmy, které dávají strojům schopnost "učit se". Je to podoblast umělé inteligence (z angl. *Artificial Intelligence*), která se používá pro studování inteligentního chování, adaptaci ve strojích, a také pro řízení, rozhodování a plánování procesů v různých systémech.

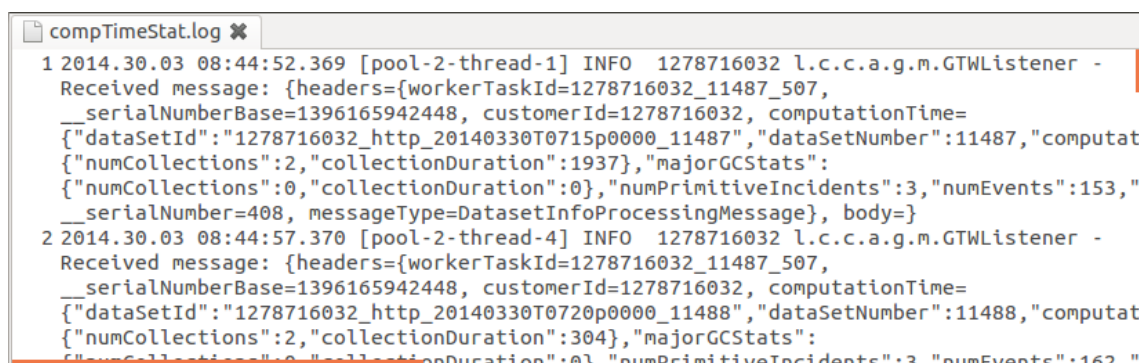
Tento framework zahrnuje pro problematiku této práce velmi užitečné techniky s regresními modely. Komplikace jeho využití byla v tom, že je vytvořen za použití neuronových sítí. Jak jsem později zjistil, ovládnutí těchto technologií pro mě bylo časově náročné a mimo oblast analytických metod, které jsem si na začátku stanovil. Kvůli tomu jsem se rozhodl zanechat tento nápad použití frameworku a hledat jiná řešení.

2.3. Implementace

2.3.1. Datová vrstva

Vzorová data se kterými pracují výpočetní servery, jsem dostal od svého vedoucího jako *log file* (nebo taky *žurnál*). Log je většinou textový soubor s příponou *.log*, který obsahuje záznam činností nějakého programu nebo procesu. Tento záznam je ve tvaru posloupnosti jednotlivých kroků příslušného běhu programu, tj. skládá se z informací o tom jak, kdy, čím nebo kým byla využívána konkrétní aplikace, jaké zatížení bylo vyvoláno nebo kolik času trvalo, aby příslušná aplikace proběhla. Logy se používají pro kontrolu procesů a zjištění přesné příčiny nějaké chyby, která nastala. Z jejich záznamů je možno určit co vedlo k chybě, jaký měla tato chyba vliv a jaké měl pak běžící proces důsledek.

Daný soubor se skládá z dostatečně velkého množství řádků, v každém z nich je zaznamenáváno několik parametrů. Tyto parametry obsahují vlastní hodnotu a/nebo jejich název.

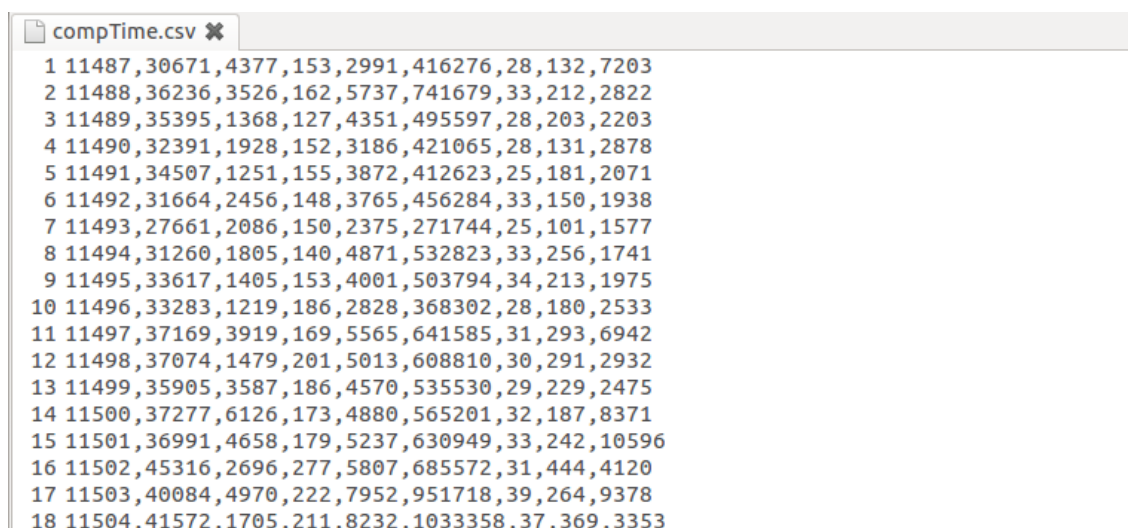


```
compTimeStat.log x
1 2014.30.03 08:44:52.369 [pool-2-thread-1] INFO  1278716032 l.c.c.a.g.m.GTWListener -
Received message: {headers={workerTaskId=1278716032_11487_507,
__serialNumberBase=1396165942448, customerId=1278716032, computationTime=
{"dataSetId":"1278716032_http_20140330T0715p0000_11487","dataSetNumber":11487,"computat
{"numCollections":2,"collectionDuration":1937},"majorGCStats":
{"numCollections":0,"collectionDuration":0},"numPrimitiveIncidents":3,"numEvents":153,"
__serialNumber=408, messageType=DatasetInfoProcessingMessage}, body=}
2 2014.30.03 08:44:57.370 [pool-2-thread-4] INFO  1278716032 l.c.c.a.g.m.GTWListener -
Received message: {headers={workerTaskId=1278716032_11487_507,
__serialNumberBase=1396165942448, customerId=1278716032, computationTime=
{"dataSetId":"1278716032_http_20140330T0720p0000_11488","dataSetNumber":11488,"computat
{"numCollections":2,"collectionDuration":304},"majorGCStats":
{"numCollections":0,"collectionDuration":0,"numPrimitiveIncidents":3,"numEvents":162,"
```

Obrázek 2.2.: Log soubor (fragment). Vlastní zdroj.

Pro pohodlnější práci se souborem a odstranění nepotřebných parametrů (např. čas, datum) jsem se rozhodl překonvertovat log soubor do souborového formátu *CSV* (z angl. *comma-separated values*, neboli hodnoty oddělené čárkami). Je to jeden z nejjednodušších formátů pro práci s tabulkovými daty. Považuji vybraný formát za vhodný, protože má podobnou strukturu jako log soubor a je často používán pro výměnu informací mezi různými systémy. Také je dost důležité, že se sloupce CSV souboru mohou tvářit jako časové řady, které se hojně využívají v regresní analýze.

Pro konvertování log souboru do formátu CSV jsem použil *shellový skript Bash*. Bash je jeden z unixových shellů, který interpretuje příkazový řádek. Shellové skripty jsou zápisem příkazů do souboru, které bychom jinak zadávali postupně na příkazovém řádku. Skript lze snadno spustit a vykonávat jím opakovanou činnost. Ty-picky se shellové skripty používají na unixových platformách, ale používají se též na Microsoft Windows.



Obrázek 2.3.: CSV soubor (fragment). Vlastní zdroj.

2.3.2. Aplikační vrstva

Projekt vytvořený pomocí spravovacího nástroje Maven obvykle má následující adresářovou strukturu

Adresář	Popis
Kořenový adresář	obsahuje pom.xml ostatní adresáře
src/main/java	obsahuje kompilovatelné .java soubory
src/main/resources	obsahuje další soubory, například konfigurační XML
src/test/java	obsahuje třídy testů
src/test/resources	obsahuje konfigurační soubory pro testy

V mém případě je pouze obsazená adresář src/main/java. I když testování je velmi dobrou praktikou mnoha programátorů, ja jsem se testy zdrojového kódu v dané práci nezabýval.

Celý program se skládá ze tří tříd:

1. `com.bp.prediction.oldClasses.ReadData.java`

Daná třída byla vytvořena pro načítání dat. Jak již bylo zmíněno, pro vytvoření programu byl použit datový soubor *Data.csv*, proto byl využit externí balíček na práci s CSV soubory pomocí

```
import au.com.bytecode.opencsv.CSVReader;
```

Následně se tento soubor stal maticí z vysvětlujících a vysvětlované proměnné.

2. `com.bp.prediction.oldClasses.NumberPrediction.java`

V této třídě se prováděly hlavní procedury a výpočty. Obsahuje tato třída několik různých metod. První z nich je metoda pro převedení souboru dat do matice v příslušné třídě:

```
public List<String []> createMatrix ()
```

Další metoda rozděljuje matici dat do matice vysvětlujících proměnných *xData* a do vektoru hodnot vysvětlované proměnné *yData*:

```
void fillMatrix (int [] xParameters)
```

Potom je tam metoda pro využití korelační analýzy. Výstupem je pole hodnot koeficientů determinace všech kombinací, vytvořených z podmnožin množiny vysvětlujících proměnných:

```
double [] getRSquaredStatistics (ArrayList<int []> list )
```

Pro vytvoření kombinací v předchozí metodě napsal jsem dvě metody využívající rekurze:

```
static void permutation (int pos , int maxUsed)  
ArrayList<int []> getCombinations ()
```

Poslední metodou v této třídě je metoda pro zjištění regresních koeficientů:

```
double [] regressionParameters (int [] bestParameters )
```

3. `com.bp.prediction.oldClasses.GraphDrawing.java`

Tato třída byla využita pro nakreslení vývoje zkoumané časové řady a odhadnuté regresní přímky.

2.4. Testování aplikace

2.4.1. Testovací data

Jako testovací data jsem měl soubor dat ve formátu CSV. Tento soubor dat popisující určitý proces se skládá z osmi nezávisle proměnných a jedné závisle proměnné:

x_1 – *dataSetNumber*
 x_2 – *numFlows*
 x_3 – *flowLoadingTime*
 x_4 – *numEvents*
 x_5 – *numFlowsInEvents*
 x_6 – *resultSize*
 x_7 – *numArtificialFileDownloadEvents*
 x_8 – *numFileDownloadFlows*

$x_9 = y$ – *computationTime*

2.4.2. Zpracování a analýza dat

Korelační matice

Jako první metoda analýzy, kterou program spouští je korelační matice všech proměnných. Pro přehled jsem vytvořil trojúhelníkovou korelační matici

$$\begin{bmatrix} x_1 & 1,000 & & & & & & & & \\ x_2 & 0,847 & 1,000 & & & & & & & \\ x_3 & 0,721 & 0,820 & 1,000 & & & & & & \\ x_4 & 0,832 & 0,958 & 0,777 & 1,000 & & & & & \\ x_5 & 0,802 & 0,981 & 0,827 & 0,920 & 1,000 & & & & \\ x_6 & 0,805 & 0,981 & 0,821 & 0,925 & 0,999 & 1,000 & & & \\ x_7 & 0,849 & 0,991 & 0,828 & 0,944 & 0,987 & 0,988 & 1,000 & & \\ x_8 & 0,811 & 0,955 & 0,807 & 0,909 & 0,954 & 0,955 & 0,958 & 1,000 & \\ y & 0,774 & 0,806 & 0,802 & 0,786 & 0,781 & 0,786 & 0,800 & 0,786 & 1,000 \end{bmatrix}$$

Z analýzy korelačních koeficientu matrice je vidět, že je v těchto datech multikolinearita. Vyloučením zbytečných vysvětlujících proměnných jsem dostal korelační matici z dvěma vysvětlujícími proměnnými x_3 a x_4 a jednou vysvětlovanou proměnnou y .

Následně jsem vypočítal koeficient determinace a příslušné regresní koeficienty pro tento zmenšený model pomocí příslušných tříd:

$$R^2 = 0.7101$$
$$b_0 = 3430.5582, b_1 = 0.5897, b_2 = 6.1441$$

Metoda výběru nejlepší podmnožiny

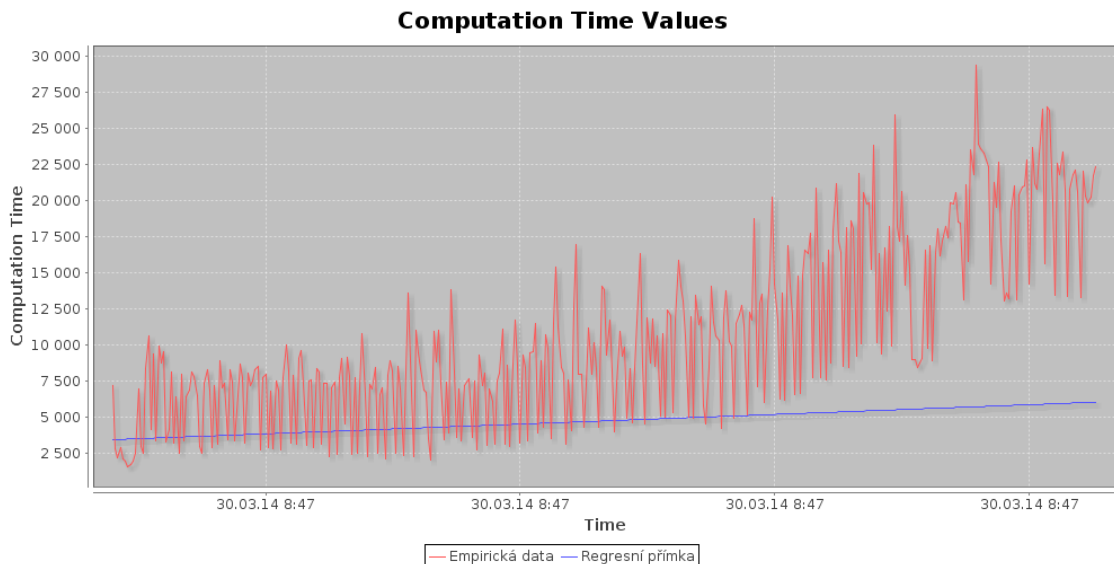
Výsledkem metody výběru nejlepší podmnožiny se stala samá množina všech vysvětlujících proměnných, tj. $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$. Jejím koeficientem determinace je hodnota $R^2 = 0,7616$. Však my tuto metodu musíme zamítnout kvůli silné multikolinearitě.

2.4.3. Vizualizace

Pro realizaci dosažených výsledků bychom měli nejdříve sestavit regresní odhadovou funkci a pak nakreslit tuto přímku. Přímka je v dané, případě lineárním odhadem trendové funkce zkoumané časové řady. Pro regresní funkci platí

$$\hat{y} = 3430.5582 + 0.5897x_3 + 6.1441x_4 \quad (2.1)$$

Pomocí třídy `com.bp.prediction.oldClasses.GraphDrawing.java` dostáváme graf popisující vývoj časové řady a její odhadnutý trend.



Obrázek 2.4.: Graf vývoje časové řady a trendu(Součástí výstupu programu)

Statistickou významnost modelu jako celku podle koeficientu determinace, lze testovat podle testovacího kritéria

$$F = \frac{R^2}{1 - R^2} \frac{n - (k + 1)}{k}, \quad (2.2)$$

které má Fisherovo rozdělení s k a $n - (k + 1)$ stupni volnosti $F[k, n - (k + 1)]$.

Nulová a alternativní hypotézy:

- H_0 : statistická nevýznamnost R^2
- H_A : statistická významnost R^2

Poduk platí $F > F^*$, odmítáme hypotézu H_0 a znamená to, že hodnota R^2 je statisticky významná a ne všechny regresní parametry jsou rovny nule.

Jestliže platí $F < F^*$, akceptujeme hypotézu H_A o statistické nevýznamnosti R^2 . To znamená, že vysvětlující proměnné neovlivňují významně vysvětlovanou proměnnou a všechny regresní parametry jsou nulové.

V tomto případě pro Fisherovo rozdělení platí $k = 2$, $n - (k + 1) = 384$. Pak $F = 195,28$ a $F^* = 470,34$. Odsud vyplývá, že $F < F^*$ a akceptujeme hypotézu H_0 , tj. koeficient determinace R^2 je statisticky nevýznamný.

Závěr

Základním cílem této bakalářské práce bylo navrhnout počítačovou aplikaci, která by pomocí regresní a korelační analýzy mohla zjistit závislost nebo nezávislost mezi hodnotami v určitých souborech dat. Po využití analytických metod a jejich implementování v programu, byly získány výsledky odhadu regresního modelu. Z grafu ukazujícího odhadnutý trend a z výsledku testování je evidentně, že odhadnutý model není dostatečně významný. Popis trendu časové řady matematickými křivkami je častým způsobem v analýze časových řad. Použil jsem dvě metody pro řešení vícenásobné lineární regrese.

První je metoda výběru podmnožiny s nejlepším koeficientem determinace. Ukázala tato metoda, že nejlepší podmnožina pro odhad regresního modelu je samá množina se všemi proměnnými. Z korelační matice ale vidíme, že jsou v tomto modelu silné závislosti mezi mnoha vysvětlujícími proměnnými. Proto jsem musel postoupit k druhé metodě.

Druhá metoda je založena na analýze korelační matice. Korelační matice může popisovat závislosti mezi úplně všemi proměnnými v modelu a může takovým způsobem objevit multikolinearitu, což je zbytečná závislost mezi vysvětlujícími proměnnými. Kvůli výskytu silné multikolinearity bylo potřeba vyřadit zbytečné nezávislé proměnné. Po takovém vyloučení zbylo jen dvě vysvětlující proměnné, jejichž koeficient determinace jsem následně analyzoval. Testování významnosti koeficientu determinace ukázalo, že i tyto dvě proměnné významně neovlivňují vysvětlovanou hodnotu.

Pro implementaci potřebných analytických metod byla navržena aplikace. I když architektura vytvořeného programu není výborná, běh programu je dostatečně rychlý. Program obsahuje třídy s pouze základními metodami pro analýzu regresních modelů. Je možné přidat hodně různých metod pro analýzu a testování regresních modelů. Tak by bylo užitečné v budoucnu přidat třídu pro analýzu nelineárních regresních modelů. Jako další možné zlepšení aplikace je přidání metod regresní analýzy popisující nelineární trend. Tyto metody jsou možným řešením nalezení regresního modelu pro uvedený v této práci soubor dat.

Z programovacího pohledu, kvalita zdrojového kódu se taky může zlepšit. Zatím byly použity jednoduché algoritmy a využity externí knihovny. Uživatelské rozhraní by bylo vhodným pro ty kteří se dobře nevyznají v programování. I když v programu není zatím uživatelské rozhraní, pro programátora je docela jednoduché se sebrat v konstrukci programu.

V závěru jsem popsal použité postupy pro dosažení staveného cíle a jejich výsledky. Taky jsem uvedl možná rozšíření programu.

Literatura

- [1] Fumio Hayashi. *Econometrics*. Princeton. Princeton University Press, 2000. ISBN 9780691010182.
- [2] Tomáš Cipra. *Analýza časových řad s aplikacemi v ekonomii*. Praha. SNTL - Nakladatelství technické literatury, 1986.
- [3] Petr Fiala. *Úvod do ekonometrie*. Praha. Nakladatelství ČVUT, 2008. ISBN 978-80-01-04004-1
- [4] Josef Arlt, Markéta Arltová, Eva Rublíková. *Analýza ekonomických časových řad s příklady*. Skripta VŠE Praha, 2002. ISBN 80-245-0307-7. V elektronické podobě k dispozici na <http://kstp.vse.cz/o-katedre/clenove-katedry/marketa-arltova/>

Internetové zdroje

- [5] Použitá Java dokumentace. URL: <http://commons.apache.org/proper/commons-math/apidocs/org/apache/commons/math3/stat/regression/>
- [6] Hana Řezanková, Luboš Marek, Michal Vrabec. *IASTAT - Interaktivní učebnice statistiky*. VŠE. URL: <http://iastat.vse.cz/casovky/> >

A. Uživatelská příručka

Výslednou aplikací se v této práci rozumí zdrojový kód, který slouží pro analýzu souboru dat. Pro jeho správné a spolehlivé použití uživatel musí mít nainstalované vývojové prostředí IntelliJ IDEA verze 13.1.6, protože zdrojový kód byl vytvořen pomocí spravovacího nástroje Maven, který přidává aplikaci určitou konstrukci.

Po otevření programu v IntelliJ IDEA by uživatel měl nejdříve importovat závislosti nástroje Maven. Tyto závislosti způsobují stažení a využití externích knihoven, potřebných pro příslušné metody. Jednotlivé závislosti se spravují v souboru *pom.xml*.

V třídě *com.bp.prediction.oldClasses.ReadData.java* se nastavuje jméno zkoumaného souboru dat, který se musí nacházet v adresáři */com.bp.prediction.oldClasses.NumberPrediction*. Při změně souboru dat se musí taky v třídě změnit předdefinovaná hodnota *xParams* určující počet nezávisle proměnných v modelu.

Pro analýzu souboru dat se musí spustit třída *com.bp.prediction.oldClasses.NumberPrediction* která obsahuje metodu *public static void main(String[] args)*.

Uživatel by pak měl dostat ve výstupu (předpokládá se využití testovacího souboru *Data.csv*):

- vypsána korelační matice
- obrázek s vývojem časové řady a odhadnutého trendu
- textový výstup ve tvaru:

Metoda výběru nejlepších podmnožin:

Nejlepší koeficient determinace = 0,7616

Vysvětlující proměnné:

$x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8$

Určení koeficientů podle korelační matice:

Nejlepší vysvětlující proměnné podle korelační matice: x_3, x_4

Regresní koeficienty nejlepších vysvětlujících proměnných:

$b_0 = 3430.5582520692515 b_1 = 0.5897176399280367 b_2 = 6.144125208268643$

B. Obsah přiloženého DVD

- bakalářská práce ve formátu PDF
- zdrojový kód aplikace