VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

DETEKCE ANOMÁLIÍ V MNOŽSTVÍ GENEROVANÝCH ZÁZNAMŮ O INCIDENTECH

ANOMALY DETECTION IN GENERATED INCIDENT TICKET VOLUMES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TIMOTEJ ŠURINA**
AUTHOR

VEDOUCÍ PRÁCE **Mgr. ROMAN TRCHALÍK, Ph.D.**
SUPERVISOR

BRNO 2019

Brno University of Technology
Faculty of Information Technology

Department of Information Systems (DIFS) Academic year 2018/2019

Bachelor's Thesis Specification

Student: **Šurina Timotej**
Programme: Information Technology
Title: **Anomaly Detection in Generated Incident Ticket Volumes**
Category: Artificial Intelligence
Assignment:

1. Research existing methods used in time series anomaly detection problems, with a focus on STL decomposition, ARIMA, Exponential Smoothing, and LSTM Networks.
2. Analyse the provided dataset of incident tickets related to various servers and applications.
3. Discuss the applicability of the researched approaches to the provided dataset, and propose solutions to the anomaly detection problem.
4. Implement and test the proposed solutions. The result should be an algorithm which analyses the trend and attempts to detect anomalies in the trend, with a report generated (e.g. in a PDF or HTML format).
5. Discuss the results and propose possible improvements to your solutions.

Recommended literature:

- Christopher M. Bishop, Pattern Recognition and Machine Learning, ISBN: 0387310738, 9780387310732 Pages: 738, Year: 2006
- David J. C. MacKay, Information Theory, Inference and Learning Algorithms, ISBN: 0521642981, 9780521642989, Pages: 640, Year: 2003

Requirements for the first semester:

- Items 1 and 2

Detailed formal requirements can be found at http://www.fit.vutbr.cz/info/szz/

Supervisor: **Trchalík Roman, Mgr., Ph.D.**
Head of Department: Kolář Dušan, doc. Dr. Ing.
Beginning of work: November 1, 2018
Submission deadline: May 15, 2019
Approval date: May 9, 2019

Abstrakt

Táto bakalárska práca sa zaoberá problematikou detekcie anomálií v časových radoch. Predstavuje metódy STL decomposition, ARIMA, Exponential Smoothing a LSTM Networks. Cieľom je pomocou týchto metód vytvoriť algoritmus, ktorý dokáže analyzovať trend v množstve generovaných záznamov o incidentoch a detekovať anomálie z trendu. Riešenie bolo vytvorené na základe dátovej sady poskytnutej firmou AT&T Global Network Services Czech Republic s.r.o. a implementované v programovacom jazyku Python.

Abstract

This bachelor thesis deals with the issue of time series anomaly detection. It presents methods STL decomposition, ARIMA, Exponential Smoothing and LSTM Networks. The aim is to use these methods to create an algorithm that can analyze the trend in a volume of generated incident tickets and detect anomalies form the trend. The solution was created based on a dataset provided by firm AT&T Global Network Services Czech Republic s.r.o. and implemented in the Python programming language.

Kľúčové slová

Anomália, detekcia anomálií, časové rady, strojové učenie, štatistické metódy, STL decomposition, ARIMA, Exponential Smoothing, LSTM Networks, tiket

Keywords

Anomaly. anomaly detection, time series, machine learning, statistical methods, STL decomposition, ARIMA, Exponential Smoothing, LSTM Networks, ticket

Citácia

ŠURINA, Timotej. *Detekce anomálií v množství generovaných záznamů o incidentech.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Mgr. Roman Trchalík, Ph.D.

Detekce anomálií v množství generovaných záznamů o incidentech

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Mgr. Romana Trchalíka, Ph.D. Ďalšie informácie mi poskytol konzultant z firmy AT&T Global Network Services Czech Republic s.r.o. pán Bc. Vojtěch Krajňanský. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Timotej Šurina
15. mája 2019

Poďakovanie

Úprimne by som sa chcel poďakovať vedúcemu práce Mgr. Romanovi Trchalíkovi, Ph.D. a zároveň aj konzultantovi z firmy AT&T Global Network Services Czech Republic s.r.o. Bc. Vojtěchovi Krajňanskému za ich pomoc pri vypracovaní tejto bakalárskej práce.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

V dnešnej dobe sa pracuje so stále väčším množstvom dát, ktoré už sami nestíhame sledovať a spracovávať. Z tohto dôvodu sa stalo strojové učenie jednou z kľúčových techník pre riešenie problémov v mnohých oboroch. Je to technika, ktorá sa dokáže na základe vopred využitých dát naučiť vzor podľa, ktorého potom spracováva nové dáta. V prípade, že sa nové dáta od naučeného vzoru líšia jedná sa o anomáliu.

Cieľom tejto práce je vytvoriť riešenie, ktoré by dokázalo monitorovať trend anomálií v záznamoch o incidentoch a vďaka tomu prísť na to, kde vznikajú problémy a čo najskôr na ne upozorniť. Vďaka tomu by bolo možné problémy vyriešiť skôr ako sa začnú zhoršovať a tým uľahčiť celú prácu. Toto riešenie by taktiež malo byť schopné analyzovať záznamy za určené obdobie a podať o nich hlásenie, na základe ktorého by bolo možné určiť problémové oblasti. Tieto oblasti by vďaka tomu mohli byť vylepšené aby sa predošlo ďalším problémom.

Ako riešenie bol zvolený algoritmus, ktorý pomocou štatistických metód a metód strojového učenia využívaných na detekciu anomálií v časových radoch detekoval anomálie v množstve generovaných záznamov o incidentoch. Výsledný algoritmus by mal byť schopný analyzovať trend a detekovať v ňom anomálie a následne o nich vytvoriť správu vo formáte HTML alebo PDF. Tento algoritmus je tvorený pre firmu AT&T Global Network Services Czech Republic s.r.o..

V kapitole 2 sú predstavené základné princípy strojového učenia a detekcie anomálií. Kapitola 3 stručne popisuje problematiku detekcie anomálií v časových radoch a stacionaritu časových radov. Následne uvádza vybrané metódy pre detekciu anomálií vysvetľuje ich princíp a uvádza, niektoré možnosti ich implementácie. Nakoniec stručne predstavuje populárne knižnice využívané pre tvorbu algoritmov na detekciu anomálií. V kapitole 4 je analyzovaný firmou poskytnutý dátový set pre vypracovanie tejto práce. V kapitole 5 je popísaný priebeh tvorby návrhu a finálny návrh implementácie algoritmu pre detekciu anomálií. Kapitola 6 popisuje implementovaný algoritmus a jeho testovanie. Kapitola 7 predstavuje záver tejto bakalárskej práce a obsahuje stručné zhrnutie a zhodnotenie výsledkov.

Kapitola 2

Úvod do strojového učenia

Strojové učenie je oblasť umelej inteligencie zaoberajúca sa metódami a algoritmami, ktoré umožňujú programu učiť sa a reagovať na rôzne vstupné hodnoty na základe informácií, ktoré sa naučil. Strojové učenie využíva prvky štatistickej analýzy, hĺbkovej analýzy dát a matematickej štatistiky.

S nárastom množstva dát sa strojové učenie stalo kľúčovou technikou pre riešenie problémov v mnohých oblastiach ako napríklad:

- Rozpoznávanie hlasu
- Produkcia energie
- Výpočtová biológia – sekvencovanie DNA
- Spracovanie obrazu a počítačové videnie – rozpoznávanie tvárí a detekcia pohybu

Algoritmy strojového učenia je možné podľa Jasona Brownlee [2] rozdeliť do troch kategórií:

Učenie s učiteľom (supervised machine learning) je keď sú k dispozícii vstupné hodnoty X a výstupné hodnoty Y. Pomocou algoritmu sa potom učí funkcia mapovať vstupy na výstupy.

$$Y = f(X)$$

Cieľom je dosiahnuť mapovaciu funkciu, ktorá dokáže podľa nových vstupov predpovedať aké budú ich výstupy. Volá sa to učenie s učiteľom pretože algoritmus naučený na dátovej sade iteratívne robí predikcie, pri ktorých sú známe správne odpovede a upravuje sa, dokým nedosiahne prijateľnú úroveň.

Učenie bez učiteľa (unsupervised machine learning) je v prípade, že sú k dispozícii iba vstupné dáta bez ich výstupov. Cieľom učenia bez učiteľa je modelovať skrytú štruktúru alebo distribúciu dát, aby sme sa o nich viac dozvedeli.

Kombinácia učenia s učiteľom a bez učiteľa (semi-supervised machine learning) je ak je k dispozícii veľa vstupných dát, ale iba k niektorým z nich sú známe výstupy. Veľa problémov spadá do tejto kategórie, pretože označovanie dát je často drahé a časovo náročné.

Úlohy strojového učenia je možné rozdeliť na:

- Klasifikáciu – rozdelenie vstupných dát do rôznych tried, učenie s učiteľom
- Regresiu – odhadovanie číselnej hodnoty výstupu podľa vstupu, učenie s učiteľom
- Zhlukovanie – rozdelenie objektov do skupín s podobnými vlastnosťami, učenie bez učiteľa
- Detekciu anomálií – identifikácia vzorov odlišných od očakávania

2.1 Detekcia anomálií

Táto sekcia vychádza z [12, 9].

Anomaly detection je technika využívaná na identifikovanie nezvyčajných vzorov (napr. položky, pozorovania, udalosti), ktoré sa líšia od očakávaného vzorov danej skupiny. Tieto vzory sú nazývané outliery. Táto technika má veľa využití ako napríklad vyhľadávanie nádorov v MRI skenoch, detekcia chýb a podvodných platieb.

Outliery najčastejšie vznikajú:

- Chybným spracovaním údajov
- Chybným meraním
- Chybným zadávaním údajov
- Chybným odberom vzoriek
- Úmyselne tvorené – využívané na testovane detekčných metód
- Prirodzené – nie sú výsledkom chyby, sú to dáta, ktoré nepatria do žiadnej z existujúcich kategórií. Tieto outliery sú nazývané „novelties“.

2.1.1 Typy Anomálií

Anomálie môžu byť obecne kategorizované do týchto 3 druhov:

- Bodové anomálie: Jedna inštancia dát je anomáliou, ak je príliš ďaleko od ostatných inštancií.
- Kontextové anomálie (tiež nazývané ako podmienkové anomálie): Inštancia dát je anomáliou iba v špecifickom kontexte. Táto anomália je bežná v časovo závislých dátach – Je normálne predať 100 kusov plaviek denne počas leta, ale zvláštne v iných obdobiach.
- Kolektívne anomálie: Skupina súvisiacich inštancií dát je anomáliou v respektíve k celej sade dát. Individuálne inštancie dát tejto skupiny nemusia byť anomálie, ale ich spoločný výskyt je anomáliou.

Kapitola 3

Time Series Anomaly Detection

Time series anomaly detection alebo detekcia anomálií v časových radoch je hlavnou časťou tejto práce. Pri detekcii anomálií sa zvyčajne využívajú jednoduché ľuďmi vypočítané prahové hodnoty alebo priemerné a smerodajné odchýlky, pomocou ktorých sa určia výrazne odchyľujúce sa dáta. Pri časových radoch sú však takto jednoduché metódy nevhodné. Čo sú to vlastne časové rady?

Časové rady sú tvorené sekvenciou dobre definovaných dátových bodov, zozbieraných v konzistentných časových intervaloch počas určitého obdobia. Dáta zozbierané náhodne alebo nepravidelne netvoria časové rady. V časových radoch často vzniká veľké množstvo falošných anomálií, sú to spomínané kontextové anomálie. Tieto anomálie vznikajú vplyvom trendov a sezón [6].

Využívajú sa teda špeciálne algoritmy pre detekciu anomálií, ktoré je možné rozdeliť do dvoch skupín. Do prvej skupiny patria algoritmy, ktoré označia každý časový bod ako anomália/nie je anomália. V druhej skupine sú tie algoritmy, ktoré predpovedajú hodnoty pre časové body a potom získanú hodnotu porovnajú s predpovedanou a vyhodnotia, či je dostatočne odlišná, nato aby bola považovaná za anomáliu. Týmto spôsobom je možné zobraziť interval spoľahlivosti, pomocou ktorého je ľahšie zistiť prečo anomálie vznikajú a overiť to.

3.1 Stacionarita časových radov

Časové rady sú stacionárne ak ich vlastnosti (priemer μ a odchýlka σ) nie sú závislé na čase, v ktorom boli pozorované. Teda nie sú ovplyvnené trendom ani sezónnosťou, pretože to by ovplyvnilo hodnoty časových radov v rôznych časoch. Časové rady však môžu obsahovať zvyšky („biely šum“), pretože tie by mali byť rovnaké bez ohľadu na čase pozorovania [8].

Štatistické metódy modelovania predpokladajú, že časové rady sú stacionárne. Klasické metódy predpovedania sa zvyčajne snažia z nestacionárnych časových radov spraviť stacionárne. Existujú však aj metódy, ktoré pracujú s nestacionárnymi časovými radmi [6].

Stacionaritu je možné podľa Jasona Brownlee [1] overiť troma spôsobmi:

1. Vykresliť graf časového radu a vizuálne skontrolovať, či obsahuje výrazné trendy alebo sezóny.

2. Spraviť súhrnné štatistiky pre sezóny (zvyčajne ročné obdobia) alebo náhodné oddiely a skontrolovať, či vznikli výrazné rozdiely.

3. Použiť štatistické testy na skontrolovanie, či sú splnené alebo boli porušené podmienky stacionarity.

3.2 STL decomposition

STL (A Seasonal-Trend Decomposition Procedure Based on Loess) je filtrovacia procedúra pre rozloženie sezónnych časových radov na tri zložky, a to trendové sezónne a zvyškové. STL má jednoduché prevedenie, ktoré pozostáva zo sekvencie aplikovaní loess vyhladzovača, jednoduchosť umožňuje analyzovať vlastnosti postupu a zároveň umožňuje rýchly výpočet, a to aj pri veľmi dlhých časových radoch a veľkom množstve trendového a sezónneho vyhladzovania. Ďalšími charakteristickými vlastnosťami STL sú špecifikácia množstva sezónneho a trendového vyhladzovania, ktoré sa v takmer nepretržitom rozsahu pohybuje od veľmi malého množstva vyhladzovania až po veľmi veľké množstvo. Robustné odhady trendu a sezónnych komponentov, ktoré nie sú deformované odchylným správaním v údajoch, špecifikácia obdobia sezónnej zložky na akýkoľvek celočíselný násobok časového intervalu odberu vzoriek väčší ako jedna a schopnosť rozložiť časové rady s chýbajúcimi hodnotami [4].

Výhody STL oproti iným metódam [8]:

- Zvláda pracovať s akýmkoľvek typom sezónnosti nielen mesačnými a štvrťročnými údajmi.
- Sezónna zložka sa môže časom meniť a rýchlosť zmeny môže byť kontrolovaná užívateľom.
- Hladkosť trendového cyklu môže byť tiež ovládaná užívateľom.
- Používateľ môže špecifikovať robustný rozklad, takže príležitostné neobvyklé pozorovania neovplyvnia odhady trendovej a sezónnej zložky. Ovplyvní to ale zvyškovú zložku.

STL má však aj svoje nevýhody, napríklad sa nezaoberá kalendárnymi variáciami a poskytuje zariadenia len pre aditívny rozklad.

3.2.1 Ako to funguje

Táto podsekcia je založená na informáciách získaných z [7].

Cieľom STL je rozdeliť časové rady Y_v pre v = 1 až N do trendových T, sezónnych S a zvyškových R zložiek.

$$Y_v = T_v + S_v + R_v$$

Toto je možné dosiahnuť cez dva vnorené cykly. Vo vonkajšom cykle sa každému dátovému bodu priradia robustné váhy v závislosti od veľkosti zvyškovej zložky, tým sa zredukujú alebo kompletne eliminujú účinky outlierov. Vnútorný cyklus iteratívne aktualizuje trendové a sezónne zložky. To sa vykonáva odpočítaním súčasného odhadu trendu od neupravovaných dát časových radov. Časové rady sa potom rozdelia na cyklické podskupiny. V prípade denných dát s týždennou sezónou sa vytvorí sedem podskupín (pondelky, utorky, apod.). Tieto podskupiny sú vyhladené pomocou loess vyhladzovania a potom prechádzajú dolno-priepustným filtrom. Sezónne zložky sa získajú odčítaním výsledkov dolnopriepustného filtra od vyhladených cyklických skupín. Následne sa sezónne zložky odčítajú

od pôvodných dát. Výsledok je vyhladený pomocou loess vyhladzovania a tým sa získajú trendové zložky. Čo zvýšilo z pôvodných dát sú zvyškové zložky.

Kľúčom k STL prístupu je loess vyhladzovanie (LOcal regrESSion). Pre súbor meraní y_i a x_i poskytuje loess vyhladzovací odhad g(x) nie len pri hodnotách x_i (pri ktorých bol y meraný), ale pre y vo všetkých hodnotách x. Na výpočet g sa zvolí kladné celé číslo q. Voľba tejto hodnoty je kľúčovým rozhodnutím pre celý STL algoritmus. Väčší parameter q, poskytuje väčšie vyhladzovanie.

Pomocou odhadnutých bodov sa vyberú vhodne podmnožiny dát a tým sú pridelené váhy. Váhy sú počítané v troch krokoch. Najskôr sa určí vzdialenosť každého bodu k bodu odhadu. Potom sa tieto vzdialenosti vydelia najväčšou určenou vzdialenosťou. Nakoniec sa vypočítajú váhy vyhodnotením „tricube“ váhovej funkcie s využitím vydelených vzdialeností.

$$w(x) = \begin{cases} (1 - |x|^3)^3, & |x| < 1 \\ 0 & |x| \geq 1 \end{cases}$$

Po vypočítaní váh sa použije „weighted least square“, tiež známa ako vážená lineárna regresia, na podmnožinu dát. Na záver sa použije lokálny polynóm pre vypočítanie hodnoty regresívnej funkcie v bode odhadu.

Navyše je možné mať sadu robustných váh ρ_i pre každý dátový bod (x_i, y_i). Tieto hmotnosti umožňujú, aby sa niektoré dátové body považovali za silnejšie/ťažšie v regresii. Ak robustné váhy existujú, použite váhy $\rho_i v_i$.

3.2.2 Implementácia

Táto metóda je najčastejšie využívaná v jazyku R ako funkcia stl, alebo jej rozšírenie v balíčku stlplus. Tento balíček využíva rovnaký algoritmus ale pridáva možnosť práce s chýbajúcimi hodnotami a niekoľko funkcií pre vykresľovanie. Primárne využitie rozkladu je pre štúdium dát časových radov a skúmanie zmien v priebehu času, môže sa využiť aj pri predpovedaní, na to môžu byť využité rôzne predpovedacie metódy, napríklad exponential smoothing alebo ARIMA. Pri implementácií je možné využiť funkciu forecast() aplikovanú na stl objekt.

V jazyku Python je pre túto metódu možné využiť stldecompose, avšak toto je „naivná“ implementácia STL metódy. Je to variácia seasonal_decompose metódy z knižnice statsmodel v ktorej je výpočet trendovej zložky realizovaný pomocou loess namiesto konvolučnej metódy využívanej v seasonal_decompose. Stldecompose tiež rozširuje DecomposeResult taktiež z knižnice statsmodel pre umožnenie predpovedania na základe vypočítaného rozkladu.

Ako alternatívu je možné použiť rozhranie rpy2. Rpy2 beží na vstavanom R, ku ktorému poskytuje prístup z Pythonu cez vlastné C-API buď ako:

- Vysokoúrovňové rozhranie, vďaka čomu funkcie R fungujú rovnako ako Python funkcie a poskytuje bezproblémovú konverziu na dátové štruktúry z knižníc numpy a pandas
- Nízkoúrovňové rozhranie bližšie k C-API

3.3 ARIMA

Táto kapitola je založená na [11, 3].

ARIMA je akronym pre „AutoRegressive Integrated Moving Average“. Táto skratka zachytáva kľúčové aspekty samotného modelu:

- AR: Autoregresívny – Model, ktorý využíva vzťah závislosti medzi pozorovaním a určitým počtom oneskorených pozorovaní.
- I: Integrovaný – Použitie diferencovania na neupravené pozorovania (napríklad odčítanie pozorovania od pozorovania v predchádzajúcom časovom kroku) aby sa časové rady stali stacionárnymi.
- MA: Pohyblivý priemer – Model, ktorý používa závislosť medzi pozorovaním a zvyškovou chybou z modelu pohyblivého priemeru aplikovaného na oneskorené pozorovania.

Tieto komponenty sú v modeli explicitne špecifikované ako parametre, štandardný zápis je ARIMA(p, d, q). Parametre sú udávane celočíselnými hodnotami pre rýchle naznačenie špecifického ARIMA modelu, ktorý je použitý. Ako hodnotu parametru je možné použiť aj 0, tým sa naznačí, že tento komponent nebude použitý. Takto je možné konfigurovať ARIMA model aby spĺňal funkciu jednoduchších modelov ako napríklad ARMA, AR, I, MA.

- p – Udáva počet oneskorených pozorovaní zahrnutých do modelu, je tiež známy ako „lag order“.
- d – Značí koľko krát sa budú pozorovania diferencovať, je tiež známy ako stupeň diferencovania.
- q – Označuje veľkosť okna pohyblivého priemeru, je tiež známy ako „order of the moving average“.

3.3.1 ARIMA výpočet

Ako už bolo spomenuté ARIMA môže spĺňať aj funkcie jednoduchších modelov preto je jeho výpočet závislý od parametrov. Pre kompletný ARIMA model sa výpočet skladá z piatich častí:

1. y_t
2. AR filter – $AR(x) \quad y_t = a_1 * y_{t-1} + \cdots + a_x * y_{t-x}$
3. I filter – $I(x) \quad \Delta y = y_t - y_{t-1} - \cdots - y_{t-x}$
4. MA filter – $MA(x) \quad \epsilon_t = b_1 * \epsilon_{t-1} + \cdots + b_x * \epsilon_{t-x}$
5. ϵ_t – chyba bieleho šumu

$$ARIMA(1,0,0) \quad y_t = a_1 y_{t-1} + \epsilon_t$$

$$ARIMA(2,1,1) \quad \Delta y_t = a_1 \Delta y_{t-1} + a_2 \Delta y_{t-2} + b_1 \epsilon_{t-1}$$

3.3.2 Konfigurácia modelu

Najskôr je potrebné zabezpečiť stacionaritu modelu pomocou diferencovania. To je možné dosiahnuť zvolením vhodnej hodnoty parametru d. Stacionarita sa skontroluje pomocou Dickey-Fullerovho testu. Pre základný model AR(1):

$$y_t = py_{t-1} + u_t$$

Kde y_t je záujem, t je index času, p je koeficient a u_t je doba chyby. Ďalej je regresívny model možné zapísať ako:

$$\Delta y_t = (p-1)y_{t-1} + u_t = \delta y_{t-1} + u_t$$

Testovanie je vykonávané na zvyškových dátach, nie na originálnych dátach, preto nie je možné využiť štandardnú časovú distribúciu pre získanie kritických hodnôt. Využíva sa preto špecifická distribúcia známa ako Dickey-Fullerova tabuľka. Týmto testom sa zisťuje, či platí nulová hypotéza (H0), ktorá naznačuje že časový rad má „unit root“ (časovo závislú štruktúru), čo naznačuje že nie je stacionárna. Pre vyvrátenie H0 musí byť „p-value“ (pravdepodobnosť pravdivosti H0) <= 0.05 (5%). V prípade, že je táto hodnota väčšia ako 5% je potrebné časový rad diferencovať. Môže sa využívať aj Augmented Dickey-Fullerov test, je to rozšírená verzia DF testu pre modely tvorené väčšími a komplikovanejšími časovými radmi.

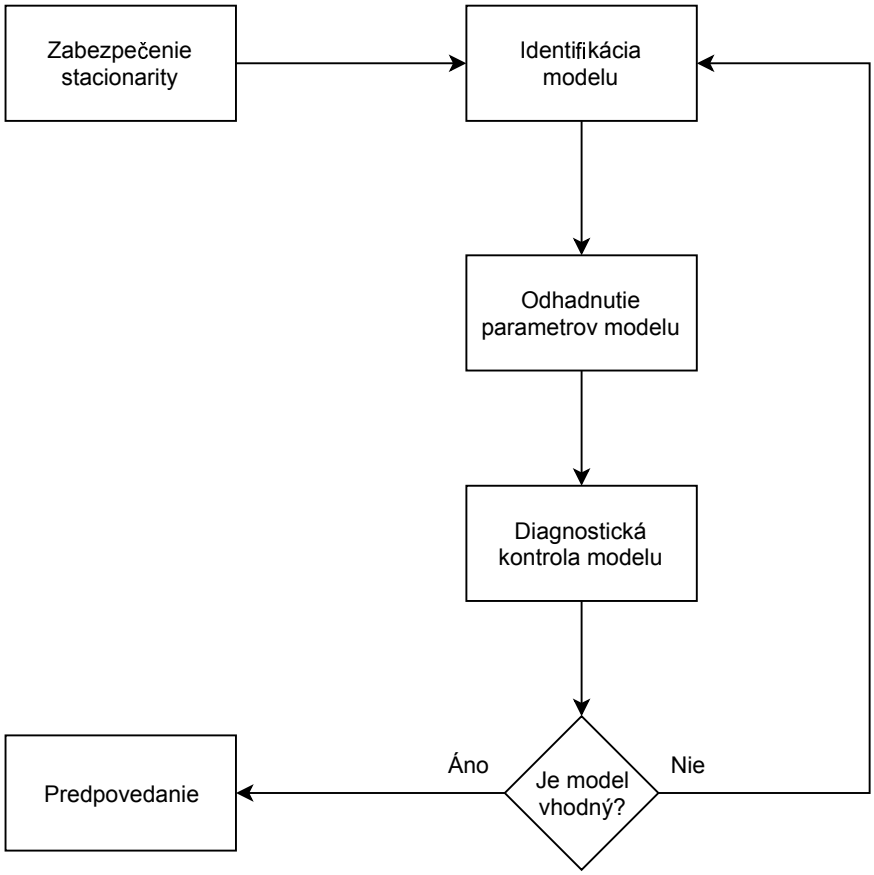
Po zabezpečení stacionarity sa využíva Box-Jenkinsov prístup. Tento prístup sa skladá z troch krokov:

1. Identifikácia modelu – Pomocou grafov a súhrnných štatistík sa identifikujú trendové, sezónne a autoregresívne prvky, aby sa získala predstava o množstve potrebného diferencovania a oneskorenia.

2. Odhadnutie parametrov modelu – Použiť vhodnú procedúru pre nájdenie koeficientov regresívneho modelu.

3. Diagnostická kontrola modelu – S využitím grafov a štatistických testov zvyškových chýb sa určí množstvo a typ temporálnej štruktúry, ktorú model nezachytil.

Tieto tri kroky sa opakujú, kým sa nedosiahne dostatočne vhodný model pre tréningovú alebo testovaciu dátovú sadu. Po získaní vhodného modelu, je tento model ďalej používaný pre predpovedanie.



Obr. 3.1: Konfigurácia ARIMA modelu

3.3.3 Implementácia

V jazyku Python môže byť ARIMA model vytvorený pomocou knižnice statsmodel v troch krokoch. Najskôr sa definuje model pomocou ARIMA(p, d, q). Potom sa použije funkcia fit() pre prípravu modelu na tréningových dátach. Nakoniec sa zavolá funkcia predict() alebo forecast() a špecifikuje sa index času, ktorý ma byť predpovedaný.

3.4 Exponential Smoothing – ES

Táto kapitola je založená na informáciách získaných z [14, 13].

ES je populárna alternatíva k metóde ARIMA. Zatiaľ čo ARIMA vykonáva predikciu na základe váženého lineárneho súčtu predošlých pozorovaní a oneskorení, ES využíva vážený súčet predošlých pozorovaní s exponenciálne klesajúcou váhou pre staršie pozorovania. Táto metóda vychádza z toho, že čím sú dáta staršie tým sú menej relevantné k novým predpovediam. ES sa využíva zvyčajne len pre krátkodobé predikcie, pretože tento spôsob môže byť pri dlhodobých predikciách dosť nespoľahlivý. Existujú 3 hlavné typy ES.

3.4.1 Single Exponential Smoothing — SES

SES je tiež známe ako Simple Exponential Smoothing. Táto metóda je vhodná pre prácu s dátami, ktoré neobsahujú žiadne alebo iba nevýrazné trendové a sezónne zložky. Základná rovnica pre SES:

$$s_t = \alpha y_t + (1 - \alpha) * s_{t-1}$$

Kde s značí vyhladenie (smoothing, občas je využívané l pre úroveň - level), t je čas a α je faktor vyhladenia pre úroveň s hodnotou $0 \leq \alpha \leq 1$. Čím menšia hodnota je zvolená, tým pomalšie je vyhladzovanie. Najvhodnejšia hodnota pre α je tá, z ktorej vznikne najmenší „mean squared error“ (MSE), to je priemerný na druhú umocnený rozdiel predikcie a prediktora.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (X_1 - Y_i)^2$$

Názov „exponential smoothing“ sa pripisuje používaniu funkcie „exponential window“ počas konvolúcie. Substitúciou definičnej rovnice za SES späť do seba dostávame:

$$s_t = \alpha * (y_t + (1 - \alpha) y_{t-1} + \cdots + (1 - \alpha)^{t-1} y_1) + (1 - \alpha)^t y_0$$

3.4.2 Double Exponential Smoothing -– DES

DES je tiež známe ako Holtova metóda, je to rozšírenie SES umožňujúce predikciu pre dáta s trendovou zložkou. Pôvodná rovnica bola rozšírená do tvaru:

$$s_t = \alpha y_t + (1 - \alpha) * (s_{t-1} + b_{t-1})$$

$$b_t = \beta * (s_t - s_{t-1}) + (1 - \beta) * b_{t-1}$$

Kde β je faktor vyhladenia pre trend, s hodnotou $0 \leq \beta \leq 1$. Ako aj pri α faktor β sa tiež volí tak, aby vznikol čo najmenší MSE.

Predikcie generované touto metódou zobrazujú konštantný stúpajúci alebo klesajúci trend. Podľa pozorovaní bolo dokázané, že čím dlhšia doba je predpovedaná tým viac sa hodnoty vzďaľujú od požadovaných. Na základe týchto pozorovaní Gardner a McKenzie vytvorili parameter ϕ s hodnotou $0 \leq \phi \leq 1$, ktorý stlmí trend do plochej čiary niekedy v budúcnosti. V praxi sa zvyčajne využívajú hodnoty 0.8 až 0.98, pretože menšie hodnoty vytvárajú príliš silné tlmenie a väčšie sú skoro na nerozpoznanie od netlmeného modelu. Rovnice s tlmením:

$$s_t = \alpha y_t + (1 - \alpha) * (s_{t-1} + \varphi b_{t-1})$$

$$b_t = \beta * (s_t - s_{t-1}) + (1 - \beta) * \varphi b_{t-1}$$

3.4.3 Triple Exponential Smoothing -– TES

TES vznikla spoluprácou Ch. Holta a jeho študenta P. Wintersa preto je známa ako Holt-Wintersova metóda. TES ďalej rozširuje DES o schopnosť práce s dátami obsahujúcimi sezónnu zložku. Rovnica je teda ďalej rozšírená takto:

$$s_t = \alpha \frac{y_t}{c_{t-L}} + (1 - \alpha) * (s_{t-1} + b_{t-1})$$

$$c_t = \gamma \frac{y_t}{s_t} + (1 - \gamma) c_{t-L}$$

L je dĺžka sezónneho cyklu a gama γ je faktor vyhladenia pre sezónnosť, taktiež s hodnotou $0 \leq \gamma \leq 1$. Ako aj u predošlých faktorov γ sa volí tiež tak, aby vznikol čo najmenší MSE. Táto metóda môže byť tiež rozšírená o parameter tlmenia.

3.4.4 Implementácia

V jazyku Python môže byť ES implementované pomocou tried z knižnice statsmodel. Ich implementácia je založená na implementácií týchto metód v balíčku „forecast“ z jazyka R.

Pre SES sa využíva trieda SimpleExpSmoothing, ktorej jediný parameter sú dáta časových radov.

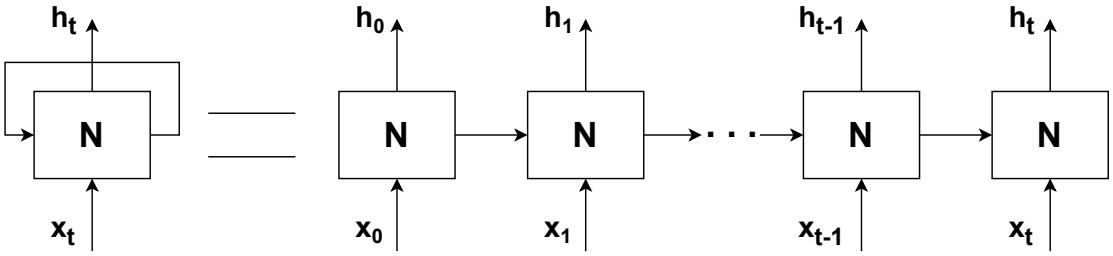
Pre DES je možné použiť triedu Holt s parametrom dáta a voliteľnými parametrami pre špecifikovanie typu trendu a stlmenia. Potom je ešte možné použiť triedu ExponentialSmoothing, ktorá je vhodná pre DES aj TES. Táto trieda má rovnaké parametre ako Holt, rozšírené o voliteľné parametre pre špecifikovanie typu sezónnosti a počtu krokov v sezóne (7 pre 7 dní v týždni pri dennom vzorkovaní, 12 pre 12 mesiacov v roku pri mesačnom vzorkovaní, atď.).

Po vytvorení inštancie triedy sa ako pri metóde ARIMA zavolajú funkcie fit() a predict() alebo forecast(). Pre DES a TES je možné vo funkcii fit() špecifikovať faktory vyhladenia a koeficient stlmenia.

3.5 LSTM Networks

Táto kapitola a jej podkapitoly sú založené na informáciách získaných z [10, 5].

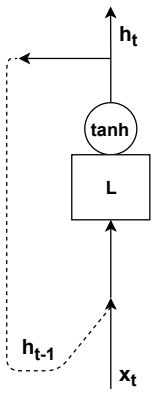
Long Short-Term Memory Networks alebo LSTM sú špeciálny druh Recurrent Neural Networks RNN. Niekedy je potrebné pre získanie správnych hodnôt potrebné vychádzať z vopred získaných informácií. Obyčajné neurónové siete nedokážu tieto informácie zachovať a preto boli vytvorené RNN. RNN sú site, ktoré využívajú cyklus, vďaka ktorému sa tieto informácie zachovávajú. Je možné ich tiež chápať ako veľa rovnakých sietí, ktoré idú za sebou a každá sieť predáva potrebné informácie svojmu následníkovi.



Obr. 3.2: RNN = opakujúce sa neurónové siete

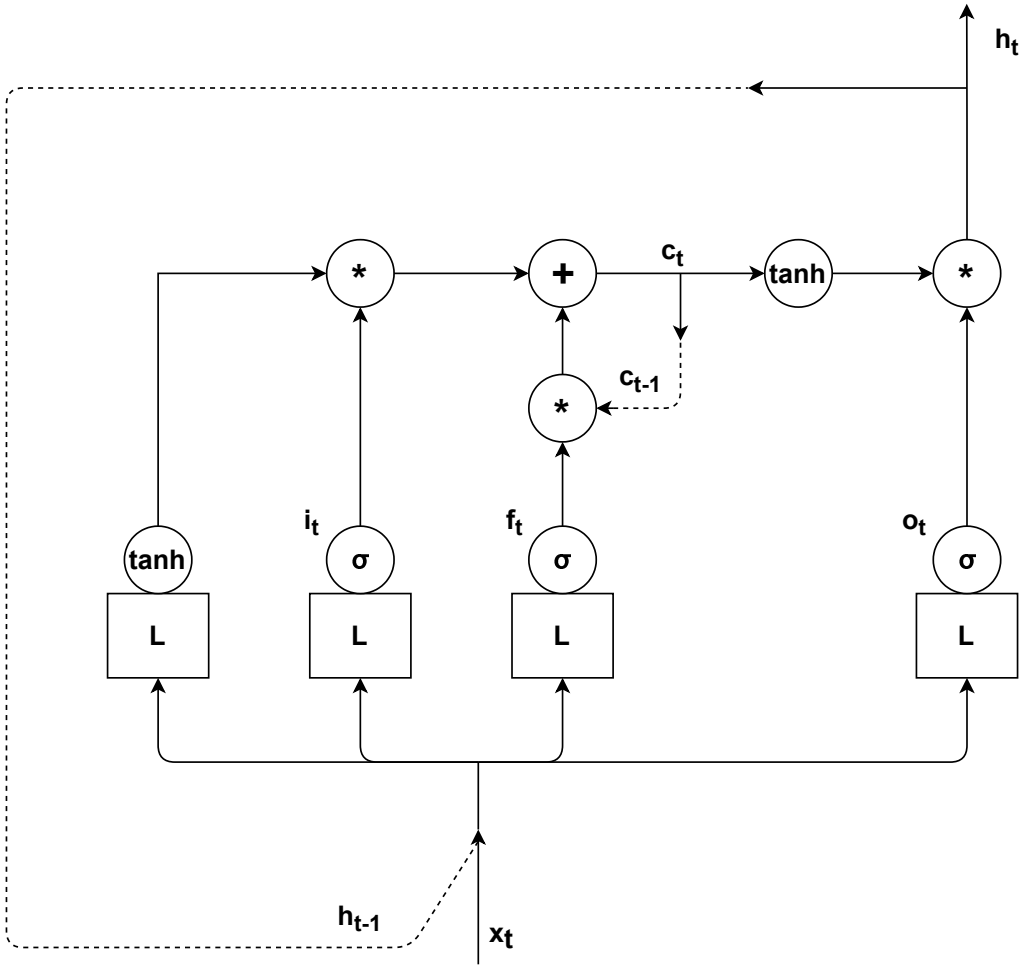
Štandardné RNN sa však dokážu naučiť pracovať iba s nedávnymi informáciami. Čím staršie informácie sú potrebné ako kontext pre predpovedanie novej informácie, tým ťažšie je pre RNN ich získať. Pre vyriešenie tohto problému boli špeciálne navrhnuté siete LSTM.

Všetky druhy RNN majú štruktúru zloženú z vrstiev neurónových sietí. Štruktúra štandardného RNN je veľmi jednoduchá skladá sa iba z jednej tanh vrstvy.



Obr. 3.3: Štruktúra štandardnej RNN

LSTM sa však skladá zo štyroch interaktívnych vrstiev a to jednej vrstvy tanh, ktorá je tiež nazývaná ako kandidátna vrstva, a troch sigmoid vrstiev. Okrem vrstiev obsahuje aj bodové operácie a to konkrétne vektorové násobenie a sčítanie a operáciu tanh.

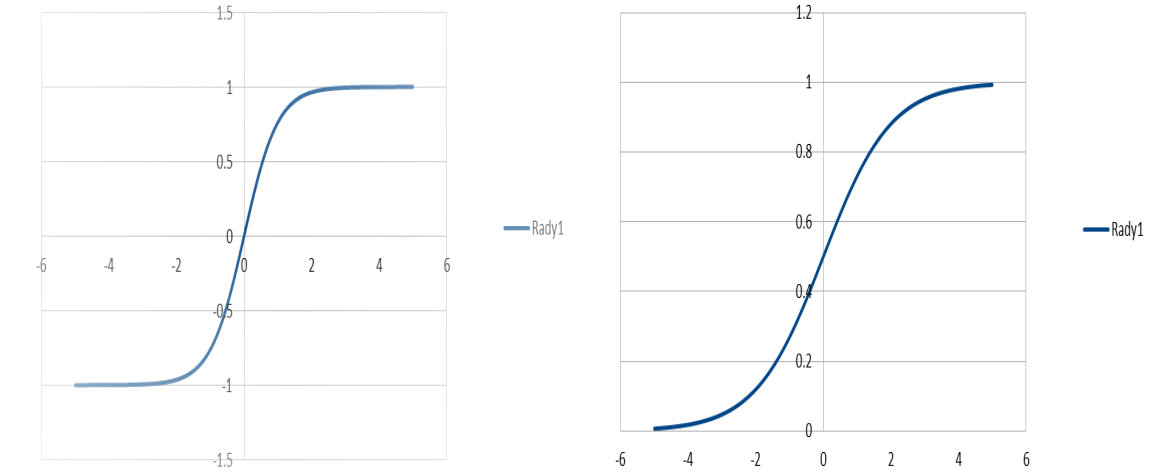


Obr. 3.4: Štruktúra LSTM

Na Obr. 3.4 je vidieť okrem vrstiev a operácií ešte označenia i_t, f_t, o_t a c_t. Hlavnou časťou LSTM sú štruktúry, ktoré sa nazývajú brány. Tieto štruktúry vznikajú kombináciou sigmoid vrstvy a vektorového násobenia. Brány sú tri a to i – vstupná brána, f – zabúdacia brána a o – výstupná brána. Nakoniec c_t je stav bunky, v ktorom sa nachádzajú všetky zozbierané možnosti.

3.5.1 Postup spracovania

Ako vstup sa využíva vstupný vektor x_t v spojení s predošlou predikciou h_{t-1}. Tento vstup potom prechádza cez jednotlivé vrstvy, čo sú vlastne samostatné neurónové siete. Tým, že sa x_t spája s h_{t-1} sa nám ale stále zväčšujú hodnoty a preto sa využívajú funkcie tanh a sigmoid, ktoré tieto hodnoty stlmia do rozsahu <-1, 1> pre tanh a <0, 1> pre sigmoid.



Obr. 3.5: tanh a sigmoid

Z kandidátnej vrstvy sa získajú možnosti predikcie, pomocou vstupnej brány sú ďalej filtrované, aby sa odstránili tie, ktoré nie sú relevantné. Relevantnosť je určená priepustnosťou brány, teda hodnotou získanou zo sigmoid funkcie. Ďalej zabúdacia brána určuje, ktoré z hodnôt získaných v predošlých krokoch sa zabudnú a ostatné sa pridajú k filtrovaným možnostiam. Pridávané hodnoty môžu byť aj opakom niektorých možností, ku ktorým sú pridávané a v tom prípade sa navzájom zrušia. Po pridaní sa všetky zozbierané možnosti uložia do pamäte, aby mohli byť použité pri ďalších krokoch. Keďže pri pridávaní mohli hodnoty znova presiahnuť rozsah <-1, 1> musia sa znova stlmiť funkciou tanh. Nakoniec sa pomocou výstupnej brány vyberie, ktoré možnosti pôjdu na výstup ako predikcie a zvyšok sa zablokuje.

3.5.2 Implementácia

Medzi najpopulárnejšie možnosti implementácie LSTM v jazyku Python patria knižnice keras a tensorflow.

3.6 Knižnice

Čo sú to vlastne za knižnice, ktoré sa používajú pre vopred spomínané metódy:

- **statsmodel** – poskytuje triedy a funkcie na odhadovanie rôznych štatistických modelov, vykonávanie štatistických testov a prieskum štatistických údajov.
- **keras** – vysokoúrovňové API pre neurónové siete, vytvorené so zameraním na umožnenie rýchleho experimentovania.
- **tensorflow** – knižnica pre vysoko výkonné výpočty, so silnou podporou strojového učenia a hĺbkového učenia. Tensorflow využíva zariadenia CPU aj GPU, ak má operácia v tensorflow implementáciu CPU aj GPU pri priradení operácie k zariadeniu má prednosť GPU.

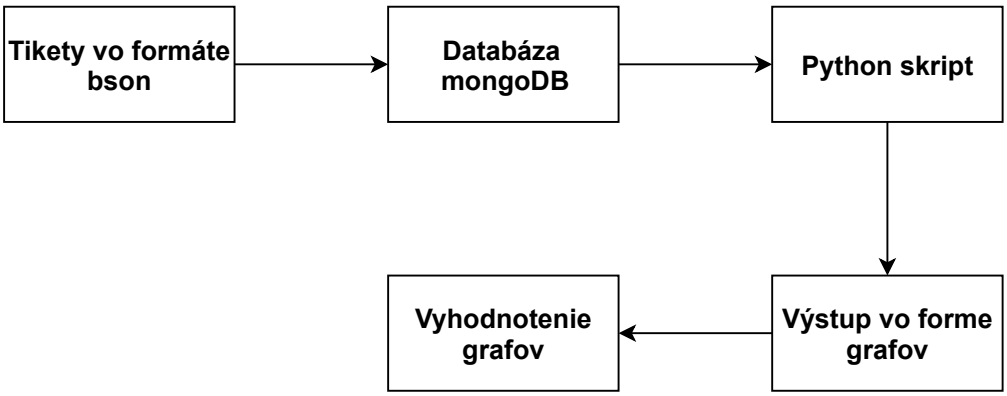
Okrem knižníc pre použitie samostatných metód sú samozrejme potrebné aj knižnice pre prácu s časovými radmi a matematickými funkciami:

- **pandas** – poskytuje vysoko výkonné a ľahko použiteľné dátové štruktúry a nástroje pre analýzu dát.
- **matplotlib** – knižnica pre vykresľovanie rôznych 2D grafov. Pre jednoduché vykresľovanie obsahuje modul pyplot, ktorý poskytuje rozhranie podobné MATLAB.
- **numpy** – základný balík pre vedecké výpočty. Je tiež možné ho využiť ako multidimenzionálny kontajner generických dát. Je možné definovať arbitrárne dátové typy a to umožňuje jednoduchú integráciu s veľkým množstvom rôznych databáz.
- **pymongo** – knižnica obsahujúca nástroje pre prácu s mongoDB. Práve v tejto databáze sa nachádzajú dáta s ktorými v tejto práci pracujem.
- **sklearn** – efektívny nástroj pre data mining a dátovú analýzu. Bol postavený na NumPy, SciPy a matplotlib. Známy pod názvom scikit-learn.
- **math** – poskytuje prístup k matematickým funkciám definovaným C štandardom.

Kapitola 4

Analýza Dátovej Sady

Pre vypracovanie programu na detekciu anomálií som dostal k dispozícií dátovú sadu, ktorá obsahuje 70 341 tiketov. Postup hlavnej časti analýzy je zobrazený na Obr. 4.1 Poskytnutú dátovú sadu s tiketmi vo formáte bson som importoval do databázy mongoDB, ktorá bola nastavená ako lokálne bežiaca služba. Tikety z tejto databázy boli ďalej spracované skriptom v programovacom jazyku Python. Konkrétne som využil verziu Python 3.7 spoločne s knižnicami pymongo, pandas, matplotlib a numpy. Výstupom tohto skriptu boli rôzne grafy, ktoré mi pomohli získať prehlad o rozložení tiketov do rôznych skupín. Vďaka tomuto rozloženiu som zistil, ktoré dáta sú najdôležitejšie pre detekciu anomálií.



Obr. 4.1: Postup analýzy

Pred získaním už spomínaných grafov bolo samozrejme potrebné zistiť význam konkrétnych častí tiketu. Na Obr. 4.2 je zobrazený tiket v stave "Closed", čo znamená, že bol spracovaný a vyriešený. Každý tiket obsahuje špecifický identifikátor "_id", stav tiketu "ticket_state", ktorý označuje, či bol tiket spracovaný alebo zrušený, typ tiketu, čo v prípade tejto dátovej sady je vždy automaticky generovaný a závažnosť problému "severityöznačenú hodnotou od 1 po 6, kde 6 značí najmenej závažný problém a 1 najzávažnejší problém. Ďalej tikety obsahujú informácie o tom akej služby sa týkajú änonymized_asset_idä bližšie informácie upresňujúce aká časť tejto služby bola ovplyvnená, konkrétny problém a jeho riešenie. Pre detekciu anomálií sú najdôležitejšie časti "trouble_reported_date", "ticket_end_date",

"functional_area", äctive_org", "managing_orgä örg_group". Z toho prvé dve časti určujú kedy boli tikety vygenerované a spracované. Zvyšné štyri časti spoločne delia tikety do rôznych kategórií a určujú skupinu, ktorá je za ne zodpovedná. Tiket tiež obsahuje pole "targetürčujúce čas, za ktorý by sa na tikete malo začať pracovať a čas, za ktorý by mal byť tiket vyriešený. Tieto časy sú uvádzané v minútach a informácie o ich dodržaní sú uvedené v častiach "time_to_pickup_metrics" a "time_to_restore_metrics". Na určenie dodržania v oboch prípadoch postačuje položka "label", ktorá môže nadobúdať tri rôzne hodnoty a to "metäk bol čas dodržaný. "missed slightlyäk bol čas prekročený o menej ako dvojnásobok a "missed by faräk bol čas prekročený o dvojnásobok a viac.


```
"_id" : 207470342,
"ticket_state" : "Closed",
"time_to_pickup_mins" : 2.07,
"key_item_affected" : "BHXOVO02",
"ticket_type" : "Auto Detect",
"service_restored_date" : ISODate("2015-11-10T20:22:32Z"),
"resolution_set_name" : "ServerNo Trouble - 374",
"ticket_cleared_date" : ISODate("2015-11-10T20:24:42Z"),
"resolution_action" : "Cleared",
"anonymized_asset_id" : 2084,
"severity" : 4,
"active_org" : "NC-GCSC-HMC",
"time_to_restore_metrics" : {
        "in_hours" : 0,
        "label" : "met",
        "target" : 7200,
        "factor" : 0.00103055555555555556
},
"ticket_end_date" : ISODate("2015-11-10T20:25:05Z"),
"managing_org" : "NC-GCSC",
"service_component" : "Software",
"org_group" : "SOLUTIONS",
"is_restricted" : false,
"time_to_restore_mins" : 7.42,
"root_cause" : "System Software",
"ticket_closed_date" : ISODate("2015-11-10T20:25:05Z"),
"time_to_repair_mins" : 7.63,
"functional_area" : "SOLUTIONS",
"trouble_reported_date" : ISODate("2015-10-06T08:54:25Z"),
"sub_root_cause" : "No Trouble Found",
"last_modified_date" : ISODate("2015-11-10T21:44:17Z"),
"resolution_item" : "No Repair Required",
"first_closed_date" : ISODate("2015-11-10T20:25:05Z"),
"target" : {
        "time_to_pickup" : 30,
        "time_to_restore" : 7200
},
"ticket_opened_date" : ISODate("2015-10-06T08:54:39Z"),
"time_to_pickup_metrics" : {
        "label" : "met",
        "accountable" : true,
        "target" : 30,
        "factor" : 0.06899999999999999
}
```


Obr. 4.2: Uzatvorený tiket

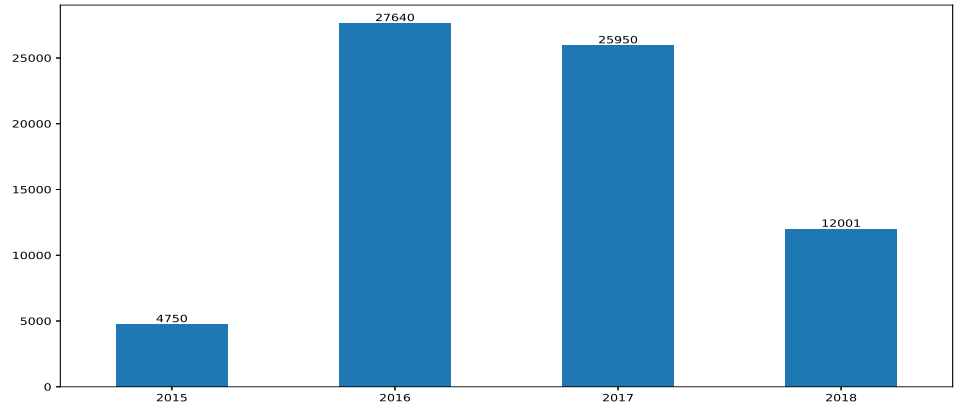
Na Obr. 4.3 je zobrazený tiket v stave "Cancel", teda zrušený tiket. Ako je z obrázka vidieť zrušený tiket sa líši od uzatvoreného iba v tom, že neobsahuje bližšie informácie o probléme uvedenej služby.

Z Obr. 4.4 je vidieť, že tikety boli zozbierané z rokov 2015 až 2018. Pri bližšom skúmaní dátumov som zistil, že tieto tikety tvoria časový rad, sú zozbierané denne podľa dátumu

```
"_id" : 202327826,
"last_modified_date" : ISODate("2015-12-01T15:01:27Z"),
"ticket_end_date" : ISODate("2015-12-01T15:01:27Z"),
"ticket_state" : "Cancel",
"ticket_type" : "Auto Detect",
"service_restored_date" : ISODate("2015-06-17T21:06:19Z"),
"anonymized_asset_id" : 4655,
"time_to_pickup_mins" : 2.1,
"severity" : 2,
"active_org" : "US-NTCM-IST3",
"time_to_restore_metrics" : {
        "in_hours" : 0,
        "label" : "met",
        "target" : 480,
        "factor" : 0.013895833333333333
},
"managing_org" : "US-NTCM",
"org_group" : "SOLUTIONS",
"is_restricted" : false,
"time_to_restore_mins" : 6.67,
"functional_area" : "SOLUTIONS",
"trouble_reported_date" : ISODate("2015-06-17T05:05:23Z"),
"target" : {
        "time_to_pickup" : 15,
        "time_to_restore" : 480
},
"ticket_opened_date" : ISODate("2015-06-17T05:05:26Z"),
"time_to_pickup_metrics" : {
        "label" : "met",
        "accountable" : true,
        "target" : 15,
        "factor" : 0.14
}
```

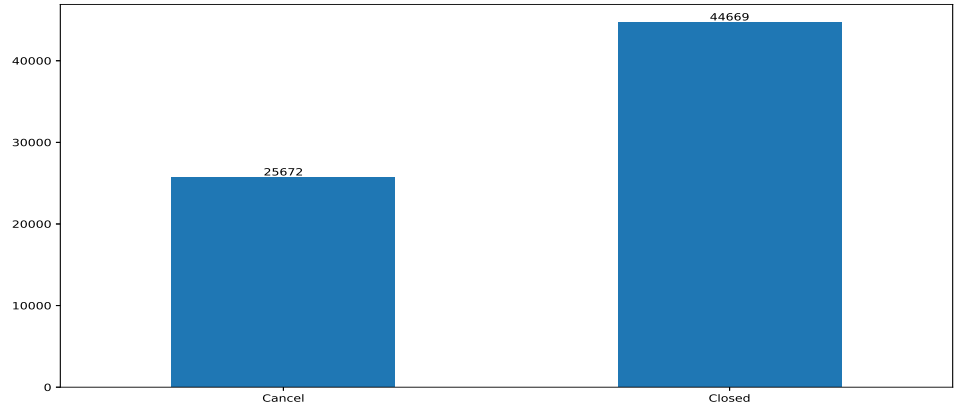

Obr. 4.3: Zrušený tiket

spracovania, ide teda o tikety spracované od 4. 11. 2015 do 22. 10. 2018. Z dátumu ich vytvorenia ide o tikety, ktoré vznikli v čase od 3. 11. 2015 do 22. 10. 2015, avšak je tu ešte 36 tiketov ktoré vznikli nepravidelne v počte 1-4 tikety denne od 15. 4. 2015 do 2. 11. 2015. Z toho vyplýva, že väčšina tiketov je spracovaná v deň vzniknutia alebo v nasledujúci deň, ale nájdu sa aj tikety, u ktorých to nie je možné. V prípade detekcie anomálií už pri vzniku tiketov by bolo potrebné z použitej dátovej sady vylúčiť týchto 36 tiketov a aj tikety z dňa 3. 11. 2015, pretože to je len malá časť tiketov, ktoré v tej dobe vznikli.



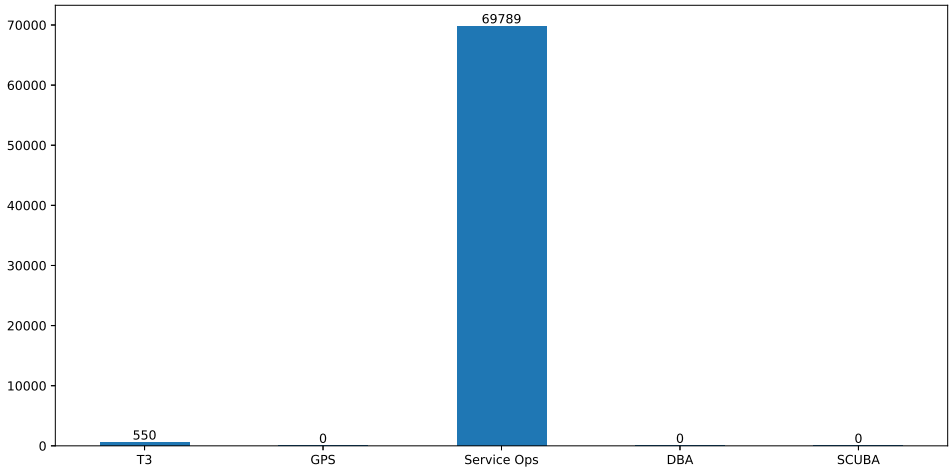
Obr. 4.4: Spracované tikety podľa rokov

Tikety sú podľa spôsobu spracovania rozdelené do dvoch skupín a to Closed a Cancel. Do skupiny Closed patria tikety, ktoré bolo treba skutočne riešiť a do skupiny Cancel patria tie, čo hlásia problém, ktorý už bol riešený v tiketoch zo skupiny Closed alebo chybné tikety. Ako už bolo povedané tikety boli zozbierané podľa dátumu spracovania. Obr. 4.5 potvrdzuje, že sa v dátovej sade nenachádzajú žiadne tikety, ktoré sú v stave čakajúcom na spracovanie.



Obr. 4.5: Tikety podľa stavu spracovania

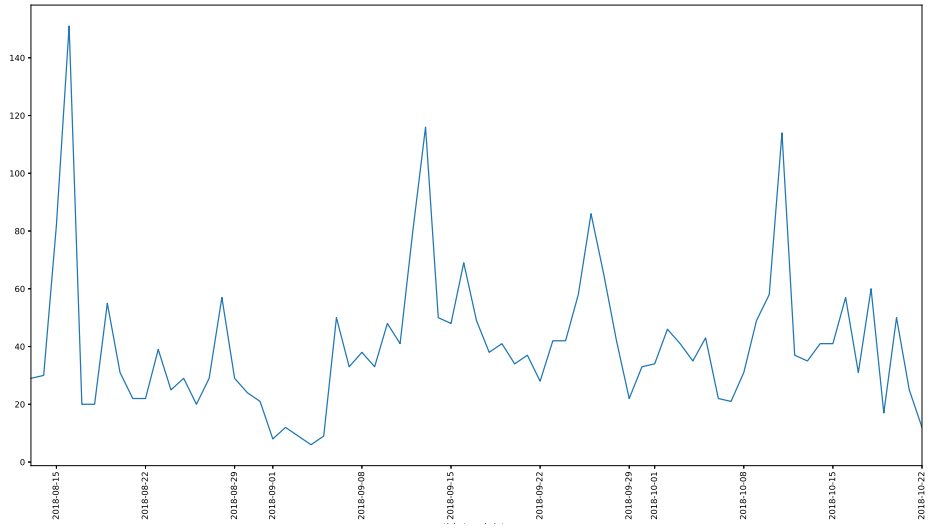
Ako je na Obr. 4.6 vidieť tikety sú rozdelené do piatich kategórií. Pri prepočte tiketov je však vidieť, že v dátovej sade sú dva tikety, ktoré nepatria do žiadnej kategórie. Konkrétne to sú tikety s _id = 214835439 a 251594957. V kategóriách GPS, DBA a SCUBA neboli riešené za posledné tri roky žiadne tikety a tým pádom každý tiket, ktorý by sa v nich objavil by bol anomáliou. V kategórii T3 síce bolo 550 tiketov ale vzhľadom nato, že to nie je ani 1 tiket denne je jasné že ich je príliš málo a sú nepravidelné a preto môžu byť všetky taktiež označené ako anomálie. Ďalej sa budem teda venovať iba tiketom z kategórie Service Ops.



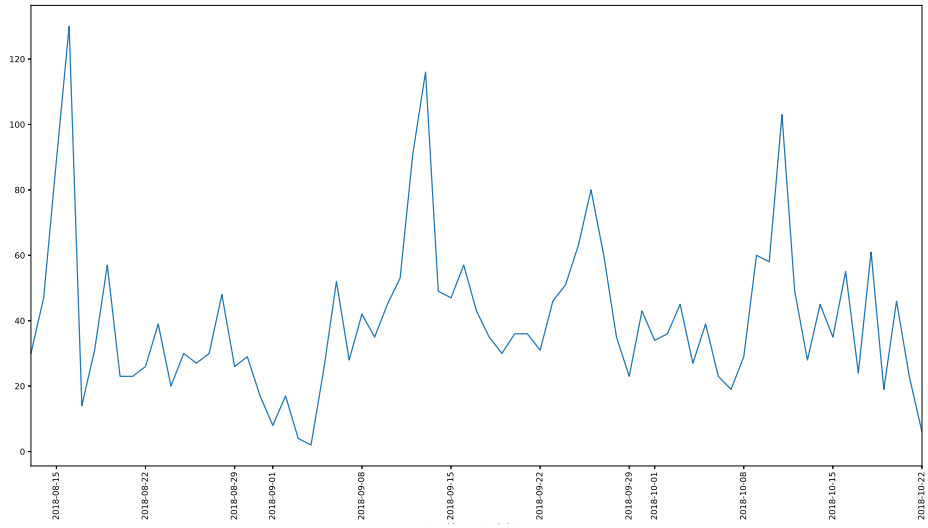
Obr. 4.6: Tikety rozdelené do kategórií

Na Obr. 4.7 a 4.8 sú zobrazené tikety z posledných desiatich týždňov v dátovej sade. Tieto grafy znázorňujú tikety podľa dátumu spracovania "ticket_end_dateä dátumu vygenerovania "trouble_reported_date". Na prvý pohľad je vidieť, že sú si grafy veľmi podobné, avšak v mnohých miestach sa líšia. Grafy majú približne rovnaký trend ale hodnoty sa skoro vo všetkých bodoch líšia. Už v prvom týždni sa vyskytla jasná anomália, pre oba grafy vznikla v ten istý dátum, no z pohľadu dátumu spracovania je vidieť 151 tiketov ale z pohľadu dátumu vygenerovania len 130 tiketov. Tento rozdiel vzniká najmä tým, že niektoré tikety nieje možne spracovať v deň ich vygenerovania. Tento rozdiel môže spôsobiť detekciu anomálií v iných dobách podľa toho, ktoré dáta sú skúmané.

Pri detekcii anomálií je dôležité prísť aj na dôvod vzniku anomálie. V tomto prípade je samozrejme najdôležitejšie zistiť odkiaľ prichádza najviac tiketov a akého problému sa týkajú. Anomália však môže vzniknúť aj na základe tiketu, ktorý vznikol aj v iný deň ale nepodarilo sa ho ešte vyriešiť. Preto si myslím, že je vhodné sledovať tikety zároveň podľa času kedy boli vygenerované aj spracované. Z anomálií, ktoré vzniknú v oboch prípadoch je možné získať mnoho užitočných informácií. Poskytnutá dátová sada však obsahuje len tikety, ktoré boli spracované a teda neviem presné údaje tiketov, ktoré boli ešte len vygenerované a čakajú na spracovanie. Z tohto dôvodu som usúdil, že je lepšie sa sústrediť na detekciu anomálií v spracovaných tiketoch.



Obr. 4.7: Tikety kategórie Service Ops spracované v posledných 10 týždňoch



Obr. 4.8: Tikety kategórie Service Ops, ktoré vznikli v posledných 10 týždňoch

Kapitola 5

Návrh

Nepodarilo sa mi nájsť existujúci spôsob, ktorým by som mohol určiť najvhodnejšiu metódu pre detekciu anomálií v počte tiketov a pri tomto rozhodnutí mi nepomohli ani moje znalosti vopred spomínaných metód. Rozhodol som sa teda pre tvorbu finálneho návrhu experimentovaním so všetkými metódami. Vytvoril som teda model každej metódy a ten som s rôznymi parametrami testoval kompletnej sade dát rozdelenej na tréningové a testovacie dáta. Keďže tréningové a testovacie dáta sa zvyčajne volia približne v pomere 2:1 ako testovacie dáta som zvolil tikety z roku 2018 a zvyšok bol použitý ako tréningové dáta.

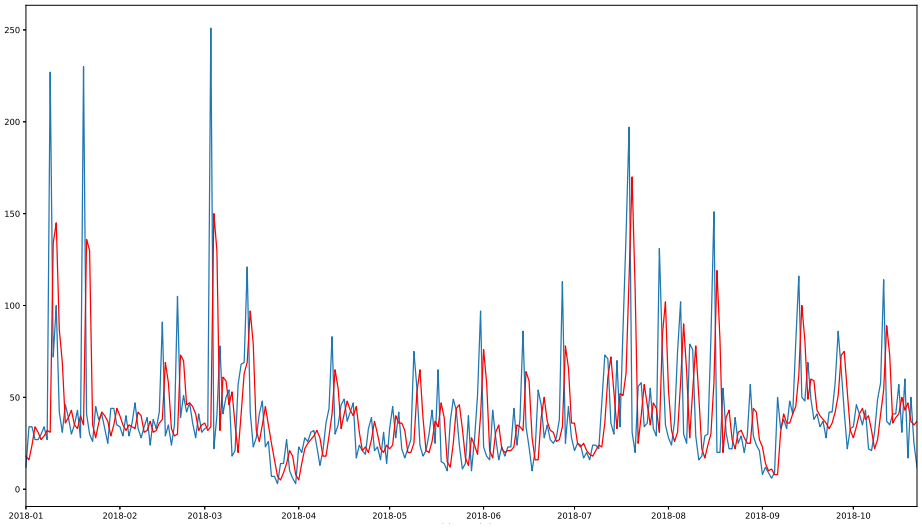
Ako boli uvedené v kapitole 3.2 metóda STL nieje určená na predpovedanie dát ale na ich analýzu, teda pre zistenie bližších informácií o dátach. V niektorých prípadoch je možné STL využiť na odhadnutie vhodných parametrov pre ostatné modely ale nie je možné určiť, či sú tieto hodnoty parametrov najlepšie a preto som túto metódu vynechal a parametre metód postupne testoval, kým som nedošiel k najlepším výsledkom.

5.1 Model - ARIMA

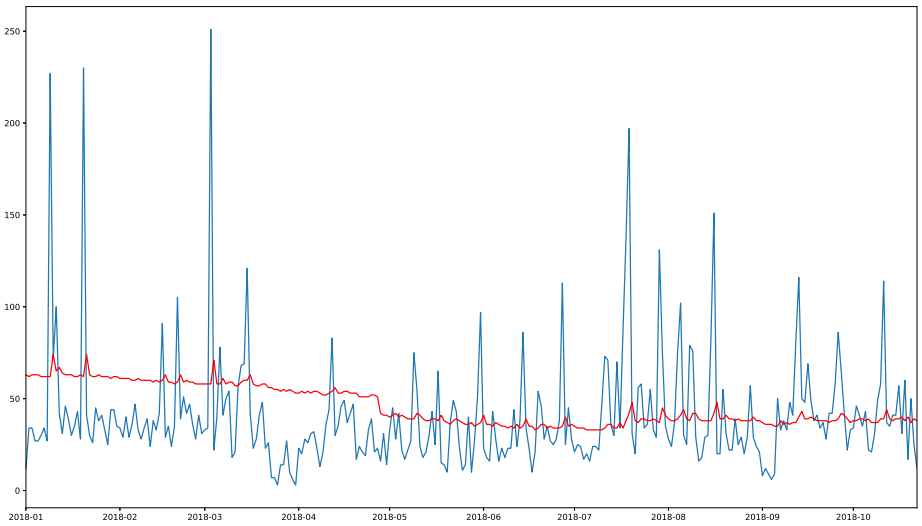
Ako bolo spomínané v kapitole 3.3 pre vytvorenie ARIMA modelu je potrebné určiť paramertry p, d a q. Ako prvý je potrebné určiť parameter d, ktorý udáva stupeň diferencovania. Pre vytvorenie modelu som využil triedu ARIMA z knižnice statsmodels, ktorá podporuje len tri možné hodnoty pre parameter d a to 0, 1, 2. Už z analýzy dát je možné určiť, že dáta je potrebné diferencovať a teda hodnotu 0 je hneď možné vylúčiť. Pri hodnote 2 som často získal zápornú predpoveď a teda ako parameter d som nakoniec zvolil hodnotu 1.

Na Obr. 5.1, kde je zobrazený výstup z modelu ARIMA(1,1,0) je vidieť, že model predpovedá hodnotu nasledujúceho dňa ako mierne znížený hodnotu z predošlého dňa a teda je nevhodný.

Z Obr. 5.2, ktorý zobrazuje výstup z modelu ARIMA(1,1,1) je na prvý pohľad jasné, že pohyblivý priemer a teda parameter $q > 0$ nie je pre tieto dáta vhodný.

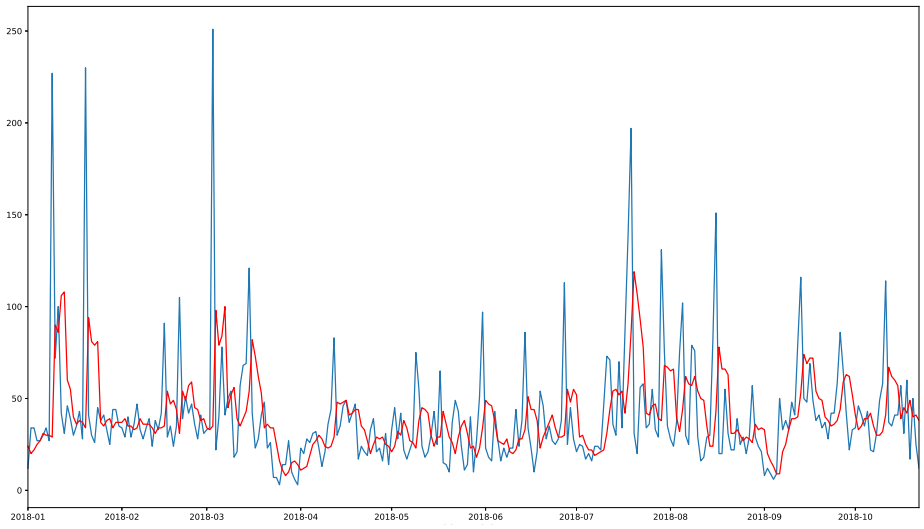


Obr. 5.1: Model ARIMA(1,1,0)

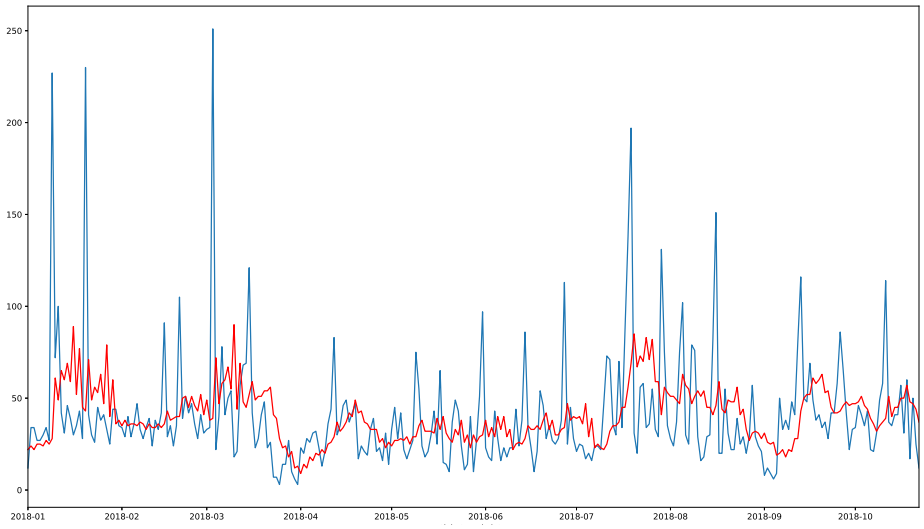


Obr. 5.2: Model ARIMA(1,1,1)

Keďže parametre d a q sú určené zostáva už len parameter p. Na nasledujúcich obrázkoch 5.3 a 5.4 sú zobrazené výsledky z modelov s parametrom p = 3 a 8. Z týchto grafov vyplýva že zvyšovaním parametra p sa výsledky postupne blížia vhodným hodnotám, ale stále sú nedostatočné. Pri ďalšom testovaní som usúdil, že pre dosiahnutie použiteľnej úrovne by bol potrebný parameter p s hodnotou väčšou ako 15, ale pri týchto hodnotách už je potrebný príliš veľký výpočtový výkon. Pri testovaní mi predpoveď na jeden deň zabrala niekoľko hodiny a preto som usúdil, že táto metóda nie je vhodná pre túto prácu.



Obr. 5.3: Model ARIMA(3,1,0)

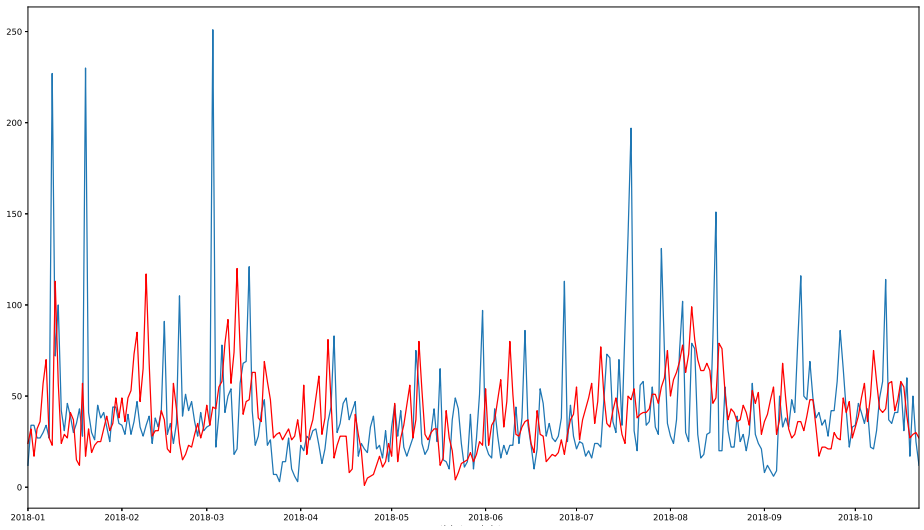


Obr. 5.4: Model ARIMA(8,1,0)

5.2 Model - Exponential Smoothing

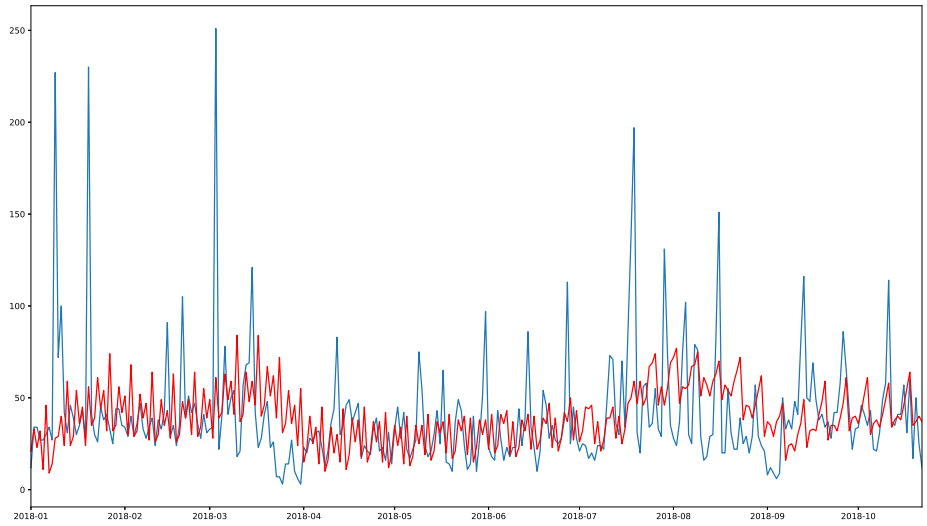
Pre model metódy Exponential Smoothing som využil typ Triple Exponential Smoothing, ktorý dokáže pracovať aj s trendovou a sezónnou zložkou. Implementoval som ju pomocou jej prevedenia v knižnici statsmodels a to triedou ExponentialSmoothing. Sezónnosť a trend som na základe doteraz získaných informácií určil ako aditívne a experimentovanie teda hlavne spočívalo v nájdení najvhodnejšieho počtu krokov v sezóne. Mimo to som vykonal aj experimenty s využitím stlmenia trendu a Box Cox transformácie ale ukázali sa ako nevhodné. Stlmenie by mohlo byť vhodné pri detekcii anomálií v rozsahu väčšom ako jeden rok ale myslím si, že je zbytočne robiť detekciu pre takto veľké rozsahy.

Na Obr. 5.5 je zobrazený priebeh s 30 krokmi teda jeden mesiac tvorí sezónu. Z tohto grafu je možné vidieť, že aj keď výsledok nie je najvhodnejší už v tejto podobe je podstatne lepší ako metóda ARIMA.



Obr. 5.5: Model Exponential Smoothing(season=30)

Po testovaní rôznych možností som prišiel k záveru, že najvhodnejší počet krokov je 7 teda týždenná sezóna. Na Obr. 5.6 je zobrazený výsledný graf s týždňovou sezónou. Z výsledkov je značné, že využitie menšej sezóny značne obmedzuje negatívny vplyv anomálií na nasledujúce predpovede a vďaka tomu sa zvyšuje presnosť. Pri študovaní výsledných hodnôt som taktiež zistil, že táto sezóna oveľa lepšie reaguje na vzrast a pokles tiketov počas týždňa. Vo väčšine prípadov počet tiketov počas týždňa totiž sleduje nasledujúci trend. Z pondelka na utorok počet tiketov klesne, potom začne do štvrtka rásť kde dosiahne vrcholu a do nedele postupne klesá až v pondelok zasa vzrastie.



Obr. 5.6: Model Exponential Smoothing(season=7)

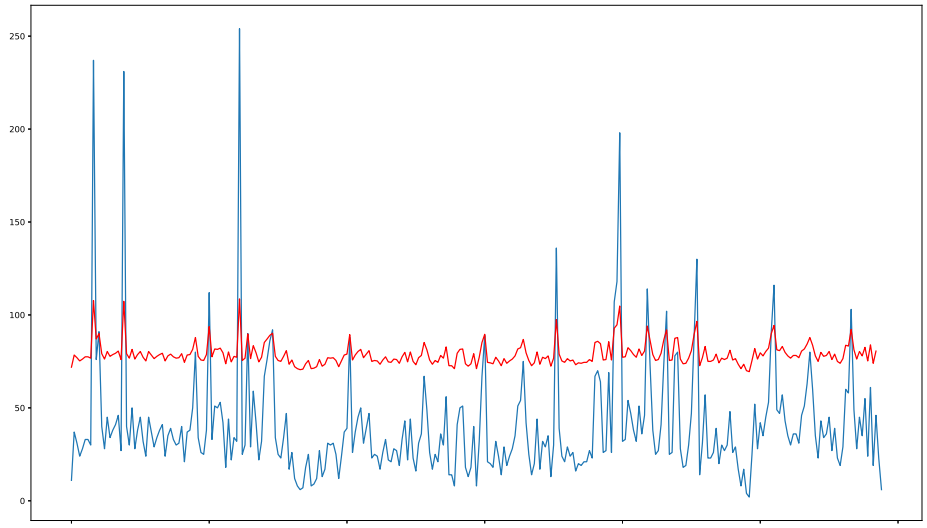
5.3 Model - LSTM network

Model LSTM bol vytvorený pomocou knižnice keras, ktorá je závislá od modulov numpy, scipy a tensorflow. Narozdiel od predošlých modelov, v ktorých je model tvorený implementáciou triedy samotnej metódy je v tomto prípade model tvorený pomocou Sequential, čo je implementácia modelu s lineárnym zväzkom vrstiev. V tomto prípade je model tvorený z dvoch vrstiev, z čoho prvá je LSTM a druhá je Dense. Dense predstavuje regulárnu husto spojenú vrstvu neurónovej siete.

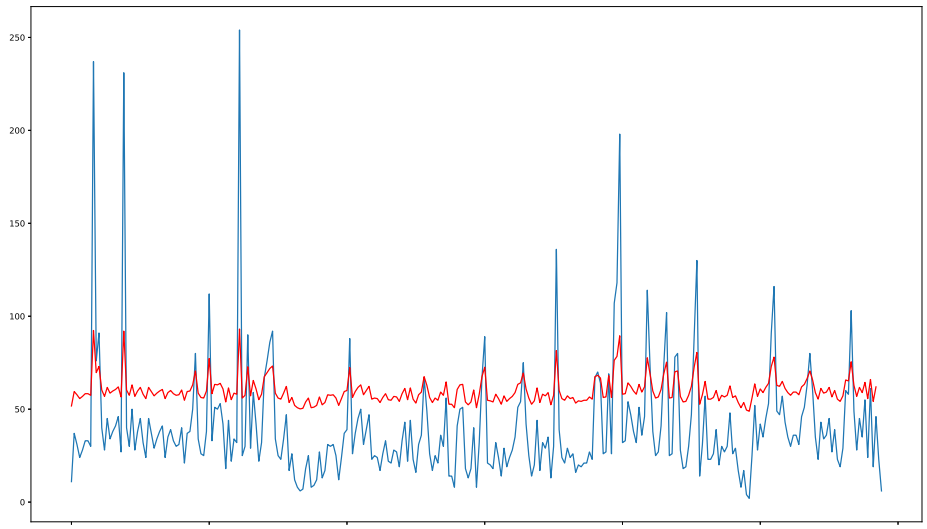
Pred vložením do modelu boli dáta transformovaná na rozsah <0, 1> pomocou triedy MinMaxScaler z knižnice sklearn. Na konci predpovedania boli hodnoty pomocou inverznej transformácie zmenené na reálne hodnoty.

Po rôznych testoch som zistil, že jediný parameter čo zlepšuje výsledok predpovedí je počet epoch. Jedna epocha prejde v momente keď celá dátová sada prejde vopred a späť cez celú neurónovú sieť. Pri nízkom počte epoch sú predpovedané hodnoty pre každý deň skoro rovnaké. Prvé vhodnejšie hodnoty sa objavili až pri využití 200 epoch ale ako je zobrazené na Obr. 5.7 predpovedané hodnoty sú podstatne vyššie ako reálne.

Pri ďalšom testovaní som zistil, že najlepšie hodnoty dokážem dosiahnuť pri využití 500 epoch ako je zobrazené na Obr. 5.8 ale na prvý pohľad je jasné, že výsledky sú stále nedostatočné. Ďalšie zvyšovanie epoch malo už len nepatrné účinky, ktoré v celku ničomu nepomohli. Po podrobnejšom študovaní som usúdil, že za nevhodné výsledky môže buď nedostatok dátových bodov alebo príliš veľký vplyv anomálií na celú dátovú sadu. Z toho som vyvodil záver, že aj táto metóda je nevhodná pre túto úlohu.



Obr. 5.7: Model LSTM(epochs=200)



Obr. 5.8: Model LSTM(epochs=500)

5.4 Výsledný návrh

Pre výsledný algoritmus som zvolil metódu Exponential Smoothing typu Triple Exponential Smoothing s týždennou sezónou. Optimálne riešenie by bolo aby algoritmus dokázal pracovať zároveň so spracovanými aj vygenerovanými tiketmi, poskytnutá dátová sada neobsahuje čisto vygenerované tikety a preto som sa rozhodol pracovať len so spracovanými tiketmi s možnosťou prepnutia na prácu s vygenerovanými tiketmi. Ďalej bude vytvorená podpora pre zvolenie databázy, na ktorú sa má pripojiť a to vo forme argumentov.

Ďalším testovaním bude určená hodnota pre vytvorenie rezervy od predpovedaných dát aby sa nedetekovalo každé prekročenie o jeden tiket. Taktiež je potrebné určiť najvhodnejšiu veľkosť pre tréningové dáta aby sa obmedzil počet prenášaných tiketov z databázy.

Výsledná správa o detekcii anomálie bude obsahovať len tie najdôležitejšie polia pre zistenie, kde anomália nastala aby bola správa čo najprehľadnejšia a vhodná pre spracované aj generované tikety.

Kapitola 6

Implementácia a testovanie

6.1 Použité technológie

Riešenie som implementoval v programovacom jazyku Python verzie Python 3.7. Ďalej som využil nasledovné knižnice:

- pymongo - pre prácu s databázou mongoDB.
- numpy - potrebna pre rôzne matematické operácie a sú na nej závislé moduly pandas, statsmodels, matplotlib, scipy a patsy.
- pandas - konštrukciu Dataframe pre prácu s tiketmi získanými z mongoDB.
- statsmodels - pre implementáciu metódy Exponential Smoothing, táto knižnica je závislá na moduloch numpy, scipy, pandas, patsy a matplotlib.
- matplotlib - pre vykreslenie grafov do výslednej správy o anomáliách.
- fpdf - pre vytvorenie výslednej správy vo formáte pdf.
- time a datetime - pre získanie aktuálneho času a prácu s dátumami.
- os - poskytuje funkcie závislé na operačnom systéme.
- scipy - knižnica pre vedecké a technické výpočty.
- patsy - knižnica pre popis štatistických modelov a tvorbu modelových matíc.

6.2 Funkčnosť

Samotné riešenie pozostáva z jedného súboru AnomalyDetection.py. Program môže bežať v dvoch rôznych stavoch v závislosti od toho, či bol zadaný argument -test. V oboch prípadoch sa najskôr určia dátumy pre vybranie tréningových a testovacích dát, v prípade neurčenia testovacích dát sa berie dátum spustenia programu, pomocou funkcie datetime.utcnow(), ako začiatok testovacích dát (aj keď v tomto prípade slúži len ako koniec tréningových dát). Následne prebehne pripojenie na databázu mongoDB a určenie vnútornej databázy a kolekcie. Ak pripojenie prebehlo úspešne a určená databáza a kolekcia skutočne existujú z databázy sa stiahnu tikety pre určené obdobie. Následne sa z nich vyberú len tie,

ktoré patria do kategórie SERVICE_OPS a tie sa podľa vopred určených dátumov rozdelia na tréningové a testovacie dáta.

Ak bol argument -test zadaný začne cyklus o dĺžke testovacích dát, ktorý pomocou ExponentialSmoothing modelu po 1 dni začne hľadať anomálie a na konci cyklu vytvorí jednu výstupnú správu, ktorá obsahuje informácie o všetkých anomáliách v určenej dobe a na koniec pridá graf znázorňujúci všetky predpovedané a skutočné hodnoty. Počas priebehu sa postupne vypisujú predpovedané a skutočné hodnoty v tvare „x: predicted y, expected z“, kde x je index začínajúci od 0, y je predpovedaná hodnota a z je hodnota testovacích dát.

V prípade že argument -test nebol zadaný program vojde do nekonečného cyklu, v ktorom vygeneruje predpoveď na aktuálny deň a začne kontrolovať, či vznikla anomálie. Ak anomálie nevznikla program sa odpojí od databázy mongoDB a na 1 hodinu sa uspí. Po hodine sa znova pripojí k databáze a stiahne si dáta pre aktuálny deň a opakuje kontrolu. Táto kontrola sa vykonáva pokým nenájde anomália alebo neskončí deň. V prípade, že sa anomália nájde vytvorí sa výstupná správa s presným časom kedy sa anomália našla a program sa odpojí od databázy a uspí sa do ďalšieho dňa. Keď sa zobudí znova sa pripojí stiahne dáta predošlého dňa pripojí ich k predošlým dátam aby mohol model presne vygenerovať dáta na ďalší deň a vytvorí kompletnú správu pre celý deň. Ak sa anomália nájde až na konci dňa vytvorí sa iba kompletná správa.

Bez ohľadu nato, či bol argument -test využitý sa v oboch prípadoch k predpovedanej hodnote pričíta 15 aby vznikla mierna rezerva pre vznik anomálie.

V programe využívam dve funkcie. Prvá je check_date(s), kde ako parameter s sa predáva argument po argumentoch -test a -train. Predaný string sa pokúsim previesť na dátum s pomocou funkcie datetime.strptime(), ktorá príma iba string v tvare dátumu rok-mesiac-deň. Ak bol string úspešne prevedený na dátum funkcia vráti string v tvare yyyy-mm-dd.

Druhá funkcia je get_report_value(df, d, g), kde df je dataframe, ktorý by mal obsahovať tikety kategórie SERVICE_OPS, d značí deň v ktorom nastala anomália, a g má predať flagGenerated, ktorý určuje, či sa pracuje s tiketmi podľa dátumu spracovania alebo dátumu vygenerovania. Vo funkcii sa vytvorí list, ktorý sa postupne napĺňa všetkými potrebnými informáciami pre hlásenie o anomálii, po získaní všetkých informácií je list funkciou vrátený.

6.3 Argumenty a chybové kódy

Program má k dispozícií 8 vstupných argumentov:

- -h - vypíše pomocnú správu s informáciami o spúšťacích parametroch.
- -test yyyy-mm-dd yyyy-mm-dd - tento argument určuje testovacie dáta v období od prvého dátumu po druhý.
- -train yyyy-mm-dd - nastaví začiatok tréningových dát na určený dátum, koniec tréningových dát bude vždy začiatok testovacích dát. Bez použitia tohto argumentu sa začiatok nastaví na jeden a pol roka pred začiatkom testovacích dát, k tejto hodnote som prišiel podľa testovania uvedeného v kapitole 6.5.
- -generated - prepne sledovanie tiketov z dátumov spracovania na dátumy vygenerovania.
- -database dbName - určuje meno databázy v mongoDB, z ktorej sa majú brať tikety, bez použitia tohto argumentu to bude meno tickets_dataset.

- -collection collName - určuje meno kolekcie vo vyššie spomenutej databáze, bez použitia tohto argumentu to bude meno tickets.
- -client uri - určuje uri pre pripojenie k mongoDB, bez použitia tohto argumentu to bude localhost.
- -pr filepath - pomocný parameter pre testovanie precision a recall, filepath určuje cestu k súboru obsahujúcemu dátumy, ktoré boli určené ako anomálie. Nie je spomenutý v pomocnom výpise.

Chybové kódy programu

- 10 - chybne zadané argumenty
- 15 - dátum zadaný v zlom tvare
- 20 - neexistujú tikety pre časť, alebo celý interval zadaných dátumov
- 30 - určená kolekcia neexistuje
- 35 - určená databáza neexistuje
- 40 - nepodarilo sa pripojiť k databáze mongoDB
- 90 - súbor Anomaly reports neexistuje a nepodarilo sa ho vytvoriť
- 100 - argument zadaný s argumentom -pr nieje súbor

6.4 Výstupná správa

Výstupná správa je tvorená vo formáte PDF a ukladá sa do adresára Anomaly reports, ktorý sa nachádza v adresári s programom. Názov správy sa tvorí podľa toho s akými argumentami bol program spustený. Názov správy vždy začína „Anomaly report“ ak bol použitý argument -generated tak sa pridá slovo generated inak nasleduje rovno dátum. Ako som už spomenul v prípade že nebol použitý argument -test sa môžu tvoriť dve rôzne správy. V prípade tej, čo vzniká pred skončením dňa sa použije dátum aj s časom získaným z datetime.utcnow() v tvare rok-mesiac-den hodiny h minúty m sekundy s a ak sa generuje až po skončení dňa tak použije iba dátum. Ak argument -test bol použitý tak sa ako dátum využíva dátum, ktorý bol zadaný spoločne s argumentom v tvare prvý dátum - druhý dátum. Ak bol použitý aj argument -train tak sa za tieto dva dátumy pridajú 3 - a dátum uvedený v s argumentom -train.

Samotná správa o každej anomálii obsahuje dátum vzniku anomálie a počty:

- všetkých tiketov alebo tiketov v dobe vygenerovania
- očakávaných tiketov, teda predpovedanú hodnotu
- tiketov podľa stavu - spracovaných a zrušených tiketov
- tiketov podľa toho ako bol splnený určený čas pre začatie pracovania na tikete a vypracovaní tiketu
- tiketov podľa závažnosti
- tiketov rozdelené do konkrétnych podskupín v kategórii SERVICE_OPS

Date:	2018-07-12	
All tickets:	71	
Expected tickets:	59	
State:	Closed 29	Cancel 42
Pickup time:	met 69	missed slightly 1
	missed by far 0	none 1
Restore time:	met 29	missed slightly 0
	missed by far 0	none 42
Severity:	1 - 4 2 - 0 3 - 5 4 - 52 5 - 1 6 - 9	
Functional_area:	GLOBAL_AM 64	SOLUTIONS 7
	NSDMNGSVCS 0	SHELL_ITI 0
Active_org (1):	NC-GCSC-HMC 7	
Org_group:	SOLUTIONS 4	PSTWEB 0
	SOL_RESTRICTED 3	VPR 0
	NIGEMS 0	
Active_org (2):	GL-SUPT-ISDK 64	US-NTCM-NMT 0
	EM-NTCM-NMT 0	

Obr. 6.1: Ukážka hlásenia o anomálii

Ako bolo už spomenuté v prípade správy o určenom časovom období sa nakoniec pridá graf. Správa by síce mohla byť aj detailnejšia, ale myslím si že je vhodnejšie zamerať sa na to najdôležitejšie aby bola správa prehľadnejšia a prípadne je možné dohľadať detaily zvlášť. Týmto spôsobom je možné správu využiť aj pre novo generované tikety bez zisťovania doplňujúcich informácií o ich presnom obsahu.

6.5 Testovanie

Implementácia bola testovaná na operačných systémoch Windows 8.1 a Ubuntu 18.04.1 LTS s použitím Python 3.7 aj Python 3.6.

Hlavná časť testovania spočívala v zistení optimálneho rozsahu tréningových dát pre dosiahnutie čo najpresnejšej detekcie algoritmov a zároveň ušetriť čas potrebný pre výpočty predpovedí a zároveň obmedziť počet tiketov prenášaných z databázy. Pre dosiahnutie tohto cieľa som prevádzal testy na rôznych časových obdobiach testovacích dát s viacerými dĺžkami tréningových dát. Testy boli realizované pomocou argumentu -pr. Výsledky testov nie sú úplne presné pretože som nedokázal s istotou určiť všetky dátumy anomálií správne a sú založené iba na mojich dedukciách z grafov zobrazených na Obr. 4.7 a 4.8. Okrem toho som ešte uplatnil informácie o sezónnosti a trende dát získaných počas tvorby návrhu.

Test prebieha ako aj iné spustenie programu s argumentom -test s tým, že detekované anomálie sa porovnávajú s označenými anomáliami, ktoré sú vypísané v určenom súbore ako dátumi a z tohto porovnania sa určia hodnoty pre True Positive TP, False Positive FP a True Negative TN. Tie sú potom použité pre výpočet **precision** a **recall**.

FP je taktiež nazývaný ako chyba typu I a je to vyvrátenie pravdivej nulovej hypotézy a k nej existuje ešte False Negative FN tiež známy ako chyba typu II a značí zlyhanie vyvrátenia falošnej nulovej hypotézy. Spolu tvoria značnú časť štatistického testovania. Jednotlivé časti sú v tomto prípade definované ako:

- True Positive - anomália bola detekovaná a skutočne existuje
- True Negative - anomália bola detekovaná ale neexistuje, býva teda tiež označovaná ako falošný poplach
- False Positive - anomália nebola detekovaná ale neexistuje
- False Negative - anomália nebola detekovaná a existuje

Precision označuje presnosť detekcie a je udávaná ako počet správne detekovaných anomálií vydelený počtom všetkých detekovaných anomálií.

$$precision = \frac{TP}{TP + FP}$$

Recall je niekedy nazývaný aj senzitivita a označuje pravdepodobnosť, že získaný výsledok bude relevantný. Je udávaná ako počet správne detekovaných anomálií vydelený jeho súčtom s počtom anomálií, ktoré sa detekovať nepodarilo.

$$recall = \frac{TP}{TP + FN}$$

Z týchto hodnôt je recall dôležitejší ale získať 100% recall je možné aj označením všetkých bodov ako anomálie a preto sa využíva spolu s precision.

Tabuľky 6.1 a 6.2 zobrazujú výsledky precision a recall testov pre štyri rôzne obdobia testovacích dát s tromi rôznymi rozsahmi tréningových dát. Ako je vidieť hodnoty sa pohybujú vo veľkých rozmedziach a väčší rozsah tréningových dát neznamená nutne lepšie výsledky. Pri nižších rozsahoch sú dáta však ľahko ovplyvniteľné nedávnymi anomáliami. Po veľkom množstve testov som určil tieto rozsahy ako najvhodnejšie. Testy boli vykonané ďalej na desiatich ďalších obdobiach. Zo záverečných výsledkov som usúdil, že rozsah 18 mesiacov je najstabilnejší a preto som ho zvolil ako základný rozsah pri neprítomnosti argumentu -train.

	18 mesiacov	12 mesiacov	9 mesiacov
2018-03-01 - 2018-05-31	64,7%	50%	45,8%
2018-05-01 - 2018-10-22	72,5%	62%	69,8%
2018-06-01 - 2018-10-22	87%	64.9%	77,8%
2018-01-01 - 2018-10-22	69,6%	64.3%	65,1%
Priemer	73.45%	60.3%	64.625%

Tabuľka 6.1: Precision

	18 mesiacov	12 mesiacov	9 mesiacov
2018-03-01 - 2018-05-31	84,6%	76,9%	84,6%
2018-05-01 - 2018-10-22	72,5%	77,5%	75%
2018-06-01 - 2018-10-22	64,5%	77,4%	67,7%
2018-01-01 - 2018-10-22	78%	72%	82%
Priemer	74.9%	75.9%	77.325%

Tabuľka 6.2: Recall

Zvyšok testov bol zameraný na správnu funkčnosť kontrol a to:

- správnosť zadania argumentov
- správnosť zadaného tvaru dátumu
- určenie existujúcej databázy
- určenie existujúcej databázy
- zlyhanie pripojenia k mongoDB
- neexistujúce tikety pre určené obdobie

Časť kódu, ktorá beží pri spustení bez parametru -test nebolo možné otestovať pretože využíva aktuálny čas získaný počas behu programu a detekovanie chyby z tiketov, ktoré postupne prichádzajú. Pre 100% test by bolo potrebné nasadiť program do živého behu na niekoľko dní. Túto časť som testoval nahradením aktuálneho času za konkrétne dátumy a priebežnú detekciu vymazaním a postupným nahrávaním tiketov späť do databázy.

Kapitola 7

Záver

Táto práca sa zaoberala štatistickými metódami a metódami strojového učenia využívaných pre detekciu anomálií v časových radoch za účelom vytvorenia algoritmu pre detekovanie anomálií v množstve generovaných záznamov o incidentoch. Algoritmus bol vytvorený pre firmu AT&T Global Network Services Czech Republic s.r.o. Experimenty boli vykonané s metódami ARIMA, Exponential Smoothing a neurónovou sieťou typu Long Short-Term Memory network. Výsledný algoritmus bol implementovaný pomocou metódy Exponentail Smoothing s týždennou sezónnosťou. Algoritmus bol vytváraný v jazyku Python verzie 3.7 a testovaný bol aj vo verzii Python 3.6. Testovanie prebehlo na operačných systémoch Windows 8.1 a Ubuntu 18.04.1 LTS. Počas testovania na poskytnutej dátovej sade bola dosiahnutá priemerná presnosť 73,45% a pravdepodobnosť detekcie správnej anomálie 74,9% pri využití základného rozsahu tréningových dát o velkosti 18 mesiacov. Dosiahnuť 100% presnosť je takmer nemožné a určenie anomálií sa môže meniť od potreby používateľa, podľa toho, či chce detekovať anomálie, čo najskôr aj za cenu viacerých falošných detekcií alebo radšej anomáliu detekovať neskôr ale vyhnúť sa tak menším falošným poplachom. Pri tvorbe som ako hranicu detekcie anomálií zvolil ako stav, v ktorom počet tiketov presiahol očakávaný počet o viac ako 15. Táto hodnota by mohla byť upravená v závislosti od potreby. Do budúcna pre snahu získať lepšie výsledky by bolo vhodné otestovať rôzne ďalšie metódy a využiť dátovú sadu s označenými anomáliami, prípadne viacero dátových sád pre presnejšie určenie najlepších parametrov.

Literatúra

[1] Brownlee, J.: *How to Check if Time Series Data is Stationary with Python*. 2016, [Online; 05.12.2018].
URL
https://machinelearningmastery.com/time-series-data-stationary-python/

[2] Brownlee, J.: *Supervised and Unsupervised Machine Learning Algorithms*. 2016, [Online; 20.10.2018].
URL https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

[3] Chen, J.: *How to Check if Time Series Data is Stationary with Python*. 2018, [Online; 07.12.2018].
URL https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp

[4] Cleveland, R. B.; Cleveland, W. S.; McRae, J. E.; aj.: *A Seassonal-Trend Decomposition Procedure Based on Loess. Journal of Official Statistics. Journal of Official Statistics*, ročník 6, č. 1, 1990, [Online; 23.11.2018].
URL https://www.wessa.net/download/stl.pdf

[5] Donges, N.: *Recurrent Neural Networks and LSTM*. 2018, [Online; 27.12.2018].
URL https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5

[6] Ericson, G.; Martens, J.; Petersen, T.; aj.: *Time Series Anomaly Detection*. 2018, [Online; 12.11.2018].
URL https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/time-series-anomaly-detection

[7] Gardner, D. R.: *STL Algorithm Explained: STL Part II*. 2017, [Online; 01.12.2018].
URL http://www.gardner.fyi/blog/STL-Part-II/

[8] Hyndman, R. J.; Athanasopoulos, G.: *Forecasting: Principles and Practice*. OTexts, 2018, ISBN 978-0987507112.

[9] Manoj, K.: *Types of Outliers*. 2012, [Online; 20.10.2018].
URL http://researchmining.blogspot.com/2012/10/types-of-outliers.html

[10] Olah, C.: *Understanding LSTM Networks*. 2015, [Online; 27.12.2018].
URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[11] Reddy, V.: *Data Analysis Course Time Series Analysis & Forecasting* . 2013, [Online; 01.12.2018].
URL https://www.slideshare.net/21_venkat/arima-26196965

[12] Santoyo, S.: *A Brief Overview of Outlier Detection Techniques*. 2017, [Online; 20.10.2018].
URL https://towardsdatascience.com/a-brief-overview-of-outlier-detection-techniques-1e0b2c19e561

[13] Stephanie: *Exponential Smoothing: Definition of Simple, Double and Triple*. 2018, [Online; 26.12.2018].
URL https://www.statisticshowto.datasciencecentral.com/exponential-smoothing/

[14] Zhang, A.: *How to Build Exponential Smoothing Models Using Python: Simple Exponential Smoothing, Holt, and Holt-Winters*. 2018, [Online; 20.12.2018].
URL https://medium.com/datadriveninvestor/how-to-build-exponential-smoothing-models-using-python-simple-exponential-smoothing-holt-and-da371189e1a1