VILNIAUS UNIVERSITETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ KATEDRA

Anomalijų aptikimas interneto prieigos stebėsenos sistemos įrašuose

Detection of anomalies in internet access monitoring system data

Bakalauro darbas

Atliko:	Jokūbas Rusakevičius	(parašas)
Darbo vadovas:	asist. dr. Vytautas Valaitis	(parašas)
Darbo recenzentas:	j. asist. Linas Petkevičius	(parašas)

Vilnius – 2019

Santrauka

TODO: Santrauka

Raktiniai žodžiai: raktinis žodis 1, raktinis žodis 2, raktinis žodis 3, raktinis žodis 4, raktinis žodis 5

Summary

TODO: summary

Keywords: keyword 1, keyword 2, keyword 3, keyword 4, keyword 5

TURINYS

Įvadas

Šis darbas yra Programų Sistemų studijų baigiamasis bakalauro darbas apie anomalijų aptikimą interneto prieigos stebėsenos sistemos įrašuose naudojant „MacroBase“ atviro kodo analitinį įrankį bei „SQL“ paketą.

Problematika

Informacijos rinkimas fizine forma, juos užrašant ant popieriaus lapo, pasak MIT Media Lab direktoriaus Joi Ito, dar visai neseniai buvo vienintelis būdas rinkti, klasifikuoti ir saugoti duomenis. Žmogus mintis ir idėjas užrašydamas ant popieriaus lapo – paversdavo žiniomis (angl. *knowledge*). Tačiau laikai keičiasi, ir dabartinė didžiųjų duomenų (angl. *big data*) situacija yra kitokia. Dėl milžiniškų surenkamų duomenų kiekio, nėra trivialu atskirti naudingus ir nenaudingus duomenis, todėl, natūraliai, negalima visų surenkamų duomenų automatiškai priskirti prie ir vadinti žiniomis. Surinkti duomenys nėra žinios, kol jų nepradedama nagrinėti, analizuoti ir filtruoti, ir tik pradėjus šį procesą galima pastebėti, kad gaunama informacija yra įdomi ir netgi svarbi [Smo14].

Duomenų, kuriais yra operuojama, juos yra renkama ir saugoma, kiekiai nuolatos didėja [EMC14]. Iš tiesų, dėl daugelio priežasčių net greitis, kuriuo šie duomenys yra generuojami eksponentiškai kyla [Mak17]. Jau 2015–2016 metais didžiosios socialinių tinklų kompanijos Twitter, Facebook ir LinkedIn pranešė fiksuojančios iki 12 milijonų įvykių (angl. *events*) per sekundę [Ast16; PFT+15; Woo15]. Šie duomenų kiekiai gerokai lenkia žmogaus gebėjimą juos apdoroti bei žymiai apsunkina darbą tiek analitikams, tiek analitiniams įrankiams. Net aukščiausios kvalifikacijos (angl. *best-of-class*) šių dienų taikomųjų programų operatoriai bei analitikai praneša panaudojantys anekdotiškai mažą jų surenkamų duomenų kiekį – mažiau nei 6% [BGR+17].

Renkant tokius didelius kiekius duomenų natūraliai, užfiksuojami įrašai, kurie yra nebūdingi įrašų rinkiniui bei išsiskiria. Duomenų vienetai nukrypę nuo kitų duomenų rinkinyje yra vadinami anomalijomis. Anomalijų aptikimas yra procesas, kurio metu yra aptinkama ir identifikuojama anomalija arba išskirtis (angl. *outlier*) duomenų rinkinyje [Tec19]. Šio darbo metu bus tiriamas anomalijų aptikimas remiantis belaidės interneto prieigos duomenų perdavimo spartos kontrolinių matavimų, atliktų 2014–2018 metais, rezultatais, kurie gauti naudojantis Lietuvos Respublikos ryšių reguliavimo tarnybos administruojama IPSS [rrtar].

Fiksuojami esminiai ar sisteminiai duomenys tobulu atveju turėtų būti peržiūrimi reguliariai ir pilnai, tam kad būtų galima užtikrinti tolygią sistemų veiklą, matavimų taisyklingumą ar renkamų duomenų korektiškumą ir laiku identifikuoti bei eliminuoti kylančias klaidas bei užkirsti

kelią kenkėjiškoms atakoms. Tačiau kaip jau minėta anksčiau, didėjantis duomenų kiekis gerokai lenkia žmogaus gebėjimą juos apdoroti. Duomenų kiekis nuolatos didėja, tačiau žmogaus dėmesys išlieką riboto dydžio. Iškyla iššūkis prioritetizuoti žmogaus dėmesį – akcentuoti, filtruoti, jungti, grupuoti ir kontekstualizuoti svarbiausius analizuojamų duomenų elgsenos ir būsenos pakitimus, taip padedant išskirti svarbiausius duomenų ypatumus ir rodyti žmogui tik apibendrintą, svarbia ir ribotą kiekį informacijos [BGR+17]. Visa nereikalinga informacija, kuri yra rodoma žmogui reikalauja ir eikvoja jo dėmesį [Sim71]. Nors žmonėms yra fiziškai neįmanoma peržiūrėti visų duomenų, mašinos ir/ar kompiuteriai gali tai atlikti, ar bent gebėtų, jei jų resursai būtų ekonomiškai paskirstyti. Analitikų teigimu, didelis kiekis svarbių duomenų yra praleidžiama dėl neefektyvaus ir/ar lėto mašinų darbo [BGR+17].

TODO: pabaigti

0.1. Darbo tikslai ir uždaviniai

Šio darbo **tikslas** – ištirti „MacroBase“ duomenų analizavimo ir anomalijų aptikimo įrankio „SQL“ paketą – „MacroBase SQL“ ir pritaikyti jo veikimą analizavimui ir anomalijų aptikimui IPSS belaidės interneto prieigos duomenų perdavimo spartos kontrolinių matavimų įrašuose.

Darbui iškelti **uždaviniai**:

1. Išanalizuoti „MacroBase SQL“ architektūrą ir veikimo principą.
2. Paruošti eksperimentinę aplinką ir IPSS įrašų rinkinį eksperimentui.
3. Atlikti eksperimentus su kontroliniais „MacroBase“ pateiktais duomenimis ir pritaikytais IPSS duomenimis
4. Palyginti eksperimentų rezultatus, keičiant pagrindinių parametrų (procentilio, palaikymo ir rizikos koeficiento) reikšmes.
5. Atlikti duomenų transformacijas (pridėti, atimti, filtruoti, keisti reikšmes) ir palyginti eksperimento rezultatus.
6. Pateikti rekomendacijas IPSS duomenų anomalijų aptikimui naudojantis „MacroBase“.

1. Duomenų analizės eksperimentas naudojant „MacroBase SQL“ modulį

1.1. Duomenų rinkinys

Šiame skyriuje aprašyti eksperimentui pritaikyti IPSS duomenų rinkiniai.

1.1.1. Eksperimentui pritaikyti duomenys

Eksperimentui buvo naudojami Lietuvos Respublikos ryšių reguliavimo tarnybos administruojamos IPSS atliekami belaidės interneto prieigos duomenų perdavimo spartos kontrolinių matavimų 2014–2018 metų rezultatai. Matavimai atlikti operatorių UAB „Bitė Lietuva“, AB „Telia Lietuva“, UAB „TELE2“ ir AB LRTC judriojo ryšio tinkluose visoje Lietuvos teritorijoje. Matavimų paskirtis yra stebėti ir įvertinti teikiamų interneto prieigos paslaugų kokybę, kaip to reikalauja Europos Sąjungos direktyvos, taip pat supažindinti visuomenę su tokių matavimų rezultatais. Įrašai pateikiami CSV formato failais. Iš viso gauti 3 failai skirti trims skirtingoms technologijoms: 3G, LTE ir WiMAX. Viename CSV faile priklausomai nuo failui priskirtos technologijos buvo apytiksliai 136000, 157000 ir 18000 įrašų eilučių, vidutiniškai 104000. Vieno failo dydis vidutiniškai 8,5MB.

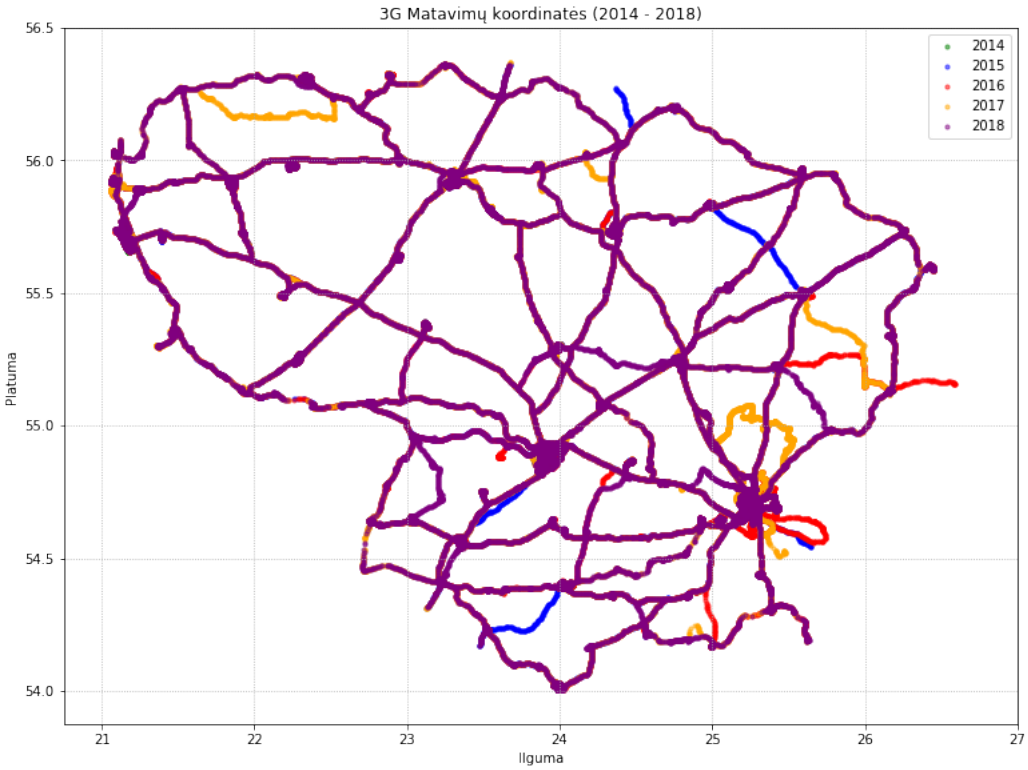
1.1.1.1. Duomenų aprašymas

Kiekvienas IPSS įrašų failas prasideda antraštės (angl. *header*) eilute, kuri susideda iš faile saugomų įrašų sąrašo laukų (angl. *fields*) pavadinimų. IPSS įrašų sarašų laukai ir jų paaiškinimas:

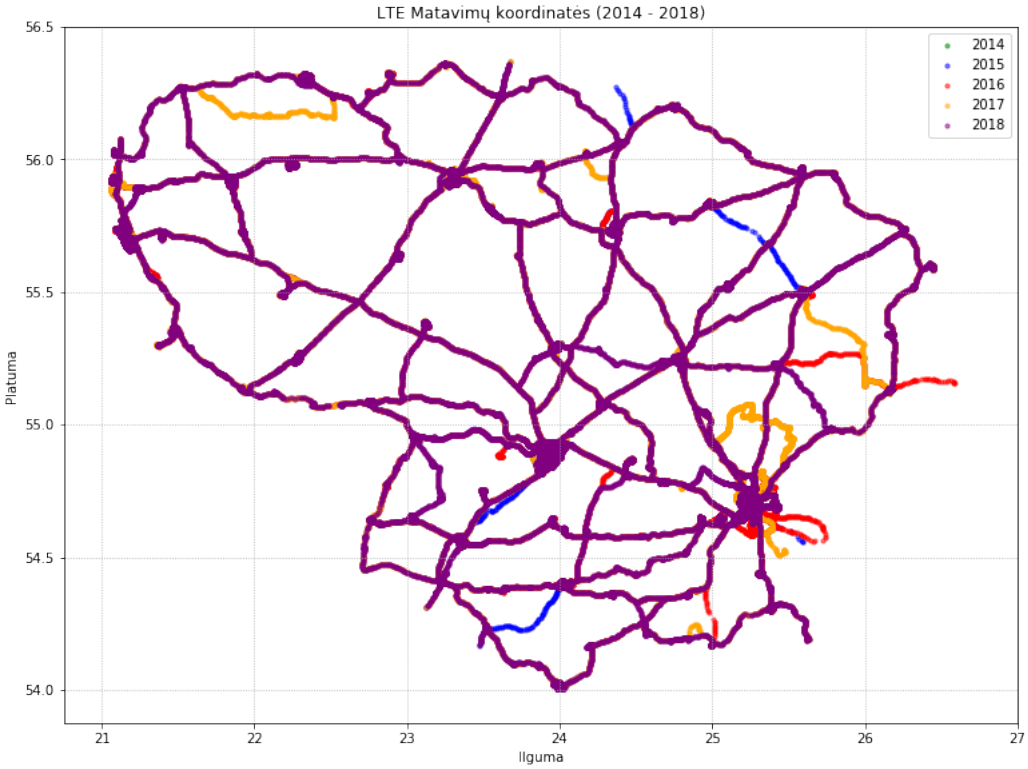
- **Data ir laikas** – įrašo fiksavimo data ir laikas, užrašoma „yyyy-MM-dd hh:mm:ss“ formatu.
- **Platuma** – matavimo įrašo fiksavimo koordinačių platuma užrašoma dešimtainiais laipsniais (angl. *decimal degrees*).
- **Ilguma** – matavimo įrašo fiksavimo koordinačių ilguma užrašoma dešimtainiais laipsniais (angl. *decimal degrees*).
- **Operatorius** – matavimo įrašo operatoriaus skaitinis kodas arba pavadinimas (vieno iš jau minėtų operatorių: „Bitė Lietuva“, „Telia Lietuva“, „TELE2“ ar LRTC).
- **Ryšio tinklas** – tinklo technologija (3G, LTE arba WiMAX).
- **Celės id** – korinio tinklo ląstelės identifikacinis numeris atvaizduotas skaitine šešioliktaine forma.

- **Bazinės stoties id** – bazinės stoties identifikacinis numeris atvaizduotas skaitine šešioliktaine forma, parametras pateikiamas tik LRTC WiMAX tinkle atliktuose matavimuose.
- **3G ryšio technologija** – matavimų metu nustatytas duomenų perdavimo technologijos porūšis (WCDMA, HSPA+, DC-HSPA+).
- **RSSI** – (angl. *Received Signal Strength Indicator*) signalo lygis, dBm (apklausiant modemą gaunamos vertės nuo -51 dBm iki -113 dBm. Mažiausia signalo vertė -113 dBm reiškia labai silpną signalą, kuriam esant praktiškai nebegalima naudotis duomenų perdavimo paslauga).
- **RSRP** – (angl. *Reference Signal Received Power*) pilotinio signalo galia, dBm (apklausiant modemą gaunamos vertės nuo -140 dBm iki -44 dBm. Mažiausia signalo vertė -140 dBm reiškia labai silpną signalą, kuriam esant praktiškai nebegalima naudotis duomenų perdavimo paslauga). Parametras pateikiamas tik LTE tinkle atliktuose matavimuose.
- **RSRQ** – (angl. *Reference Signal Received Quality*) pilotinio signalo kokybė, dB (apklausiant modemą gaunamos vertės nuo -19,5 dB iki -3 dB. Mažiausia vertė -19,5 dB). Parametras pateikiamas tik LTE tinkle atliktuose matavimuose.
- **SINR** – (angl. *Signal to Interference and Noise Ratio*) signalo ir trukdžių plius triukšmo santykis, dB (apklausiant modemą gaunamos vertės nuo -20 dB iki 30 dB. Mažiausia vertė -20 dB). Parametras pateikiamas tik LTE tinkle atliktuose matavimuose.
- **CINR** – (angl. *Carrier to Interference and Noise Ratio*) nešlio ir trukdžių plius triukšmo santykis, dB, parametras pateikiamas tik LRTC WiMAX tinkle atliktuose matavimuose.
- **Sparta kbit/s** – (angl. *Download speed*) išmatuota duomenų gavimo spartos vertė, kbps.

IPSS matavimai atliekami naudojant Ryšių reguliavimo tarnybos turimą įrangą, kuri yra sumontuota matavimams skirtame automobilyje. Matavimai atliekami automobiliui judant (angl. *drive test*) miestų gatvėmis, automagistralėmis arba rajoniniais keliais pagal pasirinktus maršrutus. Atliktų matavimų maršrutų bei koordinates galima pamatyti 1, 2 ir 3 paveikslėliuose.



1 pav. 3G Matavimai 2014–2018 metais



2 pav. LTE Matavimai 2014–2018 metais

3 pav. WiMAX Matavimai 2015–2017 metais

Aiškiau išskirtus įrašų koordinačių duomenis galima rasti Priede nr. 1.

1.1.1.2. Pradinių duomenų paruošimas

Eksperimentui atlikti buvo atrinkti laukai, kurie turi reikšmes visose įrašų sąrašo eilutėse bei taip pat buvo atmesti pasikartojančios informacijos laukai (ryšio tinklo ir ryšio technologijos laukai, kurie nurodo, koks tinklo tipas iš trijų tiriamųjų yra naudojamas faile). Taigi, 3G tinklui atrinkti šie pradiniai laukai: data, laikas, ilguma, platuma, operatorius, celės id, rssi, 3G ryšio technologija ir sparta; iš LTE duomenų atrinkti laukai: data, laikas, ilguma, platuma, operatorius, celės id, rssi, sparta; iš WiMAX duomenų atrinkti laukai: data, laikas, ilguma, platuma, operatorius, bazinės stoties id, rssi, sparta.

Eksperimentui atlikti buvo reikalingas CSV formato failai, kuriuose nebūtų lietuviškų raidžių ir būtų naudojami vieno žodžio arba vienos simbolių eilutės antraštės pavadinimai (pavyzdžiui, tarpai gali būti pakeičiami „_“ simboliu), todėl buvo realizuotas „Python“ skriptas pavadintas „RenameAndRemoveFields.py“, kuris nuskaito duotųjų duomenų CSV failus ir juos apdorojus, grąžina naujus CSV failus. Skriptas, kaip ir visi darbo metu kurti skriptai, kurtas ir leistas „Jupyter Notebook“ aplinkoje. Skripte galima pasirinkti kelią iki duotųjų CSV failų ir kelią iki naujai sukuriamų CSV failų. Taip pat galima nurodyti norimas antraštes ir norimus laukus. Šio skripto paskirtis yra pervadinti antraštes ir atmesti nereikalingus laukus ir atskirti datą nuo

laiko. Skriptas atlieka šiuos veiksmus 3 kartus kiekvienai ryšio technologijai (3G, LTE, WiMAX) (Priedas nr. 2):

1. Nuskaito CSV įrašų failą pateiktą nurodytu keliu *(18–23 ir 38 eilutės)*.
2. Sukuria naują CSV failą nurodytų keliu ir pavadinimu *(24 ir 39 eilutės)*.
3. Sukuria „csv.writer“ objektą skirtą duomenų rašymui į CSV failus *(25 eilutė)*.
4. Įrašo į naujai sukurtą CSV failą, naują antraštę, su naujais angliškais vieno žodžio (simbolių eilutės) pavadinimais *(26 eilutė)*.
5. Vykdo ciklą, kuris iš pradinio failo nuskaitytų duomenų kiekvieno įrašo atrenka reikalingus laukus, atskiria laiką nuo datos ir įrašo į CSV failą naujai sukurtą eilutę. Taip pat atlieką kiekvienos eilutės skaičiavimą *(28–34 eilutės)*.
6. Išvedamas galutinis eilučių skaičius *(35 eilutė)*.
7. Į konsolę išvedama operacijos užbaigimo žinutė *(40 eilutė)*.

1.1.1.3. Papildytų duomenų paruošimas: koordinačių suapvalinimas, aukščio virš jūros lygio pridėjimas

IPSS duomenyse yra pateikta daug informacijos, tačiau, dėl didelės jų įvairovės ir dėl realių skaičių laukuose, tik maža dalis jų yra potencialiai panaudojama aiškinimo procesui. Todėl buvo sukurtas antras skriptas pradinių duomenų papildymui (papunktis 1.1.1.2). Skripto paskirtis buvo pridėti papildomus koordinačių laukus, kuriuose būtų tos pačios koordinatės tačiau suapvalintos iki pasirinkto skaičiaus po kablelio. Skriptas turėjo ir antrą uždavinį – pasinaudojus tiksliomis, originaliomis koordinatėmis rasti aukštį virš jūros lygio ir pridėti šią bei suapvalintą reikšmes į joms skirtus naujus laukelius. Šis skriptas buvo pavadintas „AddAltitudeAndRoundedColumns.py“. Skriptas leidžia pasirinkti kelią iki pradinio CSV failo, kelią iki naujai sukuriamo CSV failo, taip pat skaitmenų po kablelio apvalinimo skaičių, bei aukščio virš jūros lygio apvalinimo sąlygas. Kadangi skriptas naudoja „Jawg.io“ API, jam reikia pateikti vartotojo prieigos žymę (angl. *access token*). Taigi naujas „Python“ skriptas nuskaito prieš tai sukurtus CSV failus, gauna aukščio virš jūros lygio reikšmes, suapvalina koordinačių bei aukščio reikšmes ir sukuria naujus CSV failus pasirinktu keliu ir pavadinimu. Skriptas atlieka šiuos veiksmus 3 kartus kiekvienai ryšio technologijai (3G, LTE, WiMAX) (Priedas nr. 3):

1. Nuskaito CSV įrašų failą pateiktą nurodytu keliu bei atskiria antraštę nuo įrašų *(9–13 ir 89 eilutės)*.
2. Iš antraštės surandami platumos ir ilgumos laukų indeksai *(14–17 ir 90 eilutės)*.
3. Pasinaudojus gautais indeksais, sukuriami visų ilgumos ir platumos reikšmių masyvai *(18–21 ir 91 eilutės)*.

4. Kadangi bus atliekamos HTTP užklausos ir URL užklausos ilgis yra ribots, yra vykdomas ciklas, kuris visus įrašus padalina į pasirinkto žingsnio dydžio dalis *(22–34 ir 92)*.
5. Vykdomas ciklas cikle, kuris visas ilgumos ir platumos reikšmes sujungia į „string“ tipo kintamuosius, kuriuose ilguma ir platuma yra atskirtos kableliu „,“ *(26–31 eilutės)*.
6. Kiekvienam žingsniui, iš platumos ir ilgumos bendrų kintamųjų sukuriamas vienas „string“ kintamasis, kur kiekvienas koordinačių poros kintamasis yra atskirtas vertikaliu brūkšniu „|“ *(33 eilutė)*.
7. Vykdomas ciklas, kuris kiekvienai žingsnio koordinačių kiekio „string“ kintamajam sukuria URL užklausą „Jawg.io“ API, kuri susideda iš koordinačių kintamojo, pagrindinio URL ir vartotojo prieigos žymės (angl. *access token*) *(35–40 ir 93 eilutės)*.
8. Kiekvienam žingsniui yra siunčiama HTTP užklausa, kuri grąžina JSON duomenų struktūrą su koordinatėmis ir jų aukščiais virš jūros lygio *(41–44 eilutės)*.
9. Iš JSON duomenų struktūros atskiriamos aukščio virš jūros lygio reikšmės ir jos yra patalpinamos į masyvą atitinkamu indeksu *(45 eilutė)*.
10. Kas 100 žingsnių yra atspausdinamas atliktų žingsnių kiekis (naudotas žingsnio dydis – 100 įrašų, todėl spausdinama, kas 10000 įrašų), tai atliekama, kad būtų galima stebėti procesą, ypač, kadangi tinkle atliekamos operacijos yra lėtesnės ir yra ribotas „Jawg.io“ API duodamų operacijų skaičius *(37 ir 47–49 eilutės)*.
11. Vykdomas ciklas, kuris kiekvienam nuskaitytam įrašui sukuria nauja įrašą su suapvalintomis iki pasirinkto skaičiaus po kablelio reikšmėmis bei aukščio virš jūros lygio ir suapvalinta iki dešimčių aukščio virš jūros lygio reikšmėmis. Šie įrašai sujungiami su atitinkamais įrašais ir sukuriamas naujas sąrašas su šiais įrašais *(51–65 ir 94 eilutės)*.
12. Sukuriami nauji laukų antraštės pavadinimai bei pertvarkomos naujai pridėtų laukų (suapvalintų ir aukščio reikšmių) pozicijos *(47–87 ir 95 eilutės)*.
13. Sukuriamas naujas CSV failas nurodytu keliu ir pavadinimu *(66–72 ir 96 eilutės)*.
14. Įrašoma nauja antraštė su papildytais pavadinimais *(69 eilutė)*.
15. Vykdomas ciklas, kuris kiekvieną naujo įrašų sąrašo elementą įveda į jam skirtą naujo CSV failo poziciją *(70–72 eilutės)*.
16. Išvedama operacijos užbaigimo žinutė *(97 eilutė)*.

1.1.1.4. Papildytų duomenų paruošimas: metų ir valandų išskyrimas

Ruošiant IPSS duomenis eksperimentui jau buvo panaudota didžioji dalis originaliai paruoštų duomenų laukų (papunkčiai 1.1.1.2 ir 1.1.1.3). Tačiau dėl didesnio potencialių aiškinamųjų laukų kiekio buvo sukurtas skriptas išskirti valandas iš laiko lauko (pvz. iš „09:15:39“ gaunama

„09“), kadangi laiko laukas buvo nenaudojamas ir metus iš datos lauko (pvz. iš „2015-09-13“ gaunama „2015“). „Python“ skriptas pavadintas „AddYearAndHour.py“. Naudojantis skriptu, galima nurodyti pradinių CSV failų kelius ir naujų CSV failų kelius. Skriptas atlieka šiuos veiksmus 3 kartus kiekvienai ryšio technologijai (3G, LTE, WiMAX) (Priedas nr. 4):

1. Nuskaito CSV įrašų failą pateiktą nurodytu keliu bei atskiria antraštę nuo įrašų *(6–10 ir 39 eilutės)*.
2. Iš antraštės surandami datos ir laiko laukų indeksai *(11–14 ir 40 eilutės)*.
3. Vykdomas ciklas, kuris kiekvienam nuskaitytam įrašui sukuria naują įrašą į kurį yra įterpiamos naujos metų ir valandų reikšmės į joms skirtas vietas. Šie įrašai sujungiami ir sukuriamas naujas sąrašas *(15–24 ir 41 eilutės)*.
4. Sukuriami nauji laukų antraštės pavadinimai su įterptomis naujomis metų ir valandų reikšmėmis, joms skirtose pozicijose *(25–31 ir 42 eilutės)*.
5. Sukuriamas naujas CSV failas nurodytu keliu ir pavadinimu *(32–37 ir 43 eilutės)*.
6. Įrašoma nauja antraštė su papildytais pavadinimais *(34 eilutė)*.
7. Vykdomas ciklas, kuris kiekvieną naujo įrašų sąrašo elementą įveda į naują CSV failą *(36–37 eilutės)*.
8. Išvedama operacijos užbaigimo žinutė *(44 eilutė)*.

1.1.1.5. Papildytų duomenų paruošimas: apvalinimo variacijos

Eksperimentui atlikti jau buvo sukurta pakankamai laukų (papunkčiai 1.1.1.2, 1.1.1.3 ir 1.1.1.4), tačiau patikrinti, kuri apvalinimo aukščio ir koordinačių kombinacija yra geriausia ir paaiškina daugiausiai anomalijų, reikia sugeneruoti daugiau duomenų rinkinių. Tam buvo parašytas dar vienas „Python“ skriptas. Šis skriptas leidžia

Skriptas atlieka šiuos veiksmus 3 kartus kiekvienai ryšio technologijai (3G, LTE, WiMAX) (Priedas nr. 5):

1.

Eksperimentui atlikti buvo nuskaityti ir apdoroti apie 300000 įrašų, vidutiniškai 100000 per failą. TODO: pridėti citatą

1.2. Eksperimento aplinkos paruošimas

Eksperimentas buvo atliekamas naudojant „Ubuntu (64-bit)“ operacinę sistemą įdiegtą virtualioje mašinoje „Oracle VM VirtualBox“. „MacroBase SQL“ įdiegtas ir paruoštas darbui naudojantis „MacroBase“ dokumentacija.

1.2.1. Virtuali mašina eksperimentui

„MacroBase“ pateikiami pavyzdžiai ir konfigūraciniai nurodymai yra pateikiami „Linux“ operacinėms sistemoms. Dėl šios priežasties, eksperimentui atlikti buvo pasirinkta atviro kodo nemokama operacinė sistema „Ubuntu“. Dėl paprastumo buvo nuspręsta naudoti virtualią mašiną operacinei sistemai įdiegti. Atviro kodo nemokama virtuali mašina „Oracle VM VirtualBox“ buvo pasirinkta šiai užduočiai. Galutinės eksperimentui naudotos sisteminės specifikacijos:

- Reali mašina, kurioje atliekamas eksperimentas – nešiojamasis kompiuteris „Lenovo Y50-70“.

  Realios mašinos Operacinė sistema **„Microsoft Windows 10 Pro“**, versija 10.0.17763.

  Realios mašinos procesorius: **i7-4720HQ**

  Realios mašinos operatyvioji atmintis: **8GB**

- **„Oracle VM VirtualBOX“** virtuali mašina, versija **6.0.8** r130520 (Qt5.6.2).

  Virtualios mašinos operacinės sistemos „Ubuntu“ 64 bitų versiją **„Ubuntu (64-bit)“**, versija **18.04 LTS**.

  Virtualiai mašinai suteiktas virtualus standusis diskas: **40GB**.

  Virtualiai mašinai skirta operatyvioji atmintis: **4GB**.

1.2.1.1. Virtualios mašinos paruošimas

Virtualios mašinos paruošimas eksperimentui:

1. Iš „Oracle VM VirtualBox“ internetinės svetainės atsisiųstas naujausias „Windows 10“ operacinei sistemai skirtas diegimo failas (`https://www.virtualbox.org/wiki/Downloads`).
2. Sekant sąrankos vedlio nurodymus įdiegta „Oracle VM VirtualBox“ virtuali mašina.
3. Iš „Ubuntu“ internetinės svetainės atsisiųstas naujausias „Ubuntu (64-bit)“ operacinės sistemos ISO failas (`https://www.ubuntu.com/download/desktop`).
4. Atidarius „Oracle VM VirtualBox“ programinę įrangą pradėtas naujos virtualios mašinos pridėjimo procesas spaudžiant mygtuką su tekstu „New“.
5. Sekant naujos virtualios mašinos pridėjimo langus pasirinktos šios parinktys:

   Operacinės sistemos tipas: „Linux“.

   Versija: „Ubuntu (64-bit)“.

   Operatyviosios atminties kiekis megabaitais: 4096MB.

   Virtualus standusis diskas: 40GB.
6. „Oracle VM VirtualBox“ pagrindiniame lange pasirinkus naujai sukurtą vitualią mašiną

spaustas mygtukas su tekstu „Start“.

7. Atidarytame virtualios mašinos lange pasirinktas anksčiau atsiųstas „Ubuntu (64-bit)“ ISO failas.
8. Pasirinkta „Install“ operacija ir sekant diegimo vedlį, įdiegta „Ubuntu (64-bit)“ operacinė sistema.

1.2.2. „MacroBase“ analitinio įrankio paruošimas

„MacroBase SQL“ diegiamas pagal „MacroBase“ dokumentacijoje nurodytus žingsnius. Tačiau prieš pradedant diegimo procesą reikia įdiegti įrankius bei papildomus paketus, tam kad „MacroBase“ taisyklingai veiktų bei būtų galima atlikti diegimo procesą. Eksperimentui naudotos programinės įrangos specifikacijos:

- „MacroBase“ versija **v1.0**.
- Java JRE versija **1.8.0_212**.
- Java JDK versija **1.8.0_212**.
- „Apache Maven“ versija **3.6.0**.

1.2.2.1. „MacroBase SQL“ diegimui reikalingi papildomi paketai

Sėkmingam „MacroBase SQL“ diegimo procesui naujai paruoštoje „švarioje“ „Ubuntu“ operacinėje sistemoje, reikalingi papildomi pagalbiniai paketai. Šiuos paketus įdiegti buvo naudojamos šios terminalo komandos:

1. Versijavimo kontrolės sistemos „Git“ diegimas:

   ```
   sudo apt install git
   ```

2. Java projektų valdymo ir diegimo priemonės „Apache Maven“ diegimas:

   ```
   sudo apt install maven
   ```

3. Java JRE diegimas:

   ```
   sudo apt install openjdk-8-jre
   ```

4. Java JDK diegimas:

   ```
   sudo apt install openjdk-8-jdk
   ```

1.2.2.2. „MacroBase SQL“ analitinio įrankio diegimas

TODO: CHANGE Sekant „MacroBase“ dokumentacijoje nurodytus žingsnius įdiegtas „MacroBase SQL“ įrankis:

1. Atidarytas „Ubuntu“ terminalas.
2. Klonuota projekto repozitorija:

   ```
   git clone https://github.com/stanford-futuredata/macrobase.git
   ```

3. Pereita į „MacroBase“ naujai sukurtą direktoriją:

   ```
   cd macrobase
   ```

4. Paleistas „MacroBase SQL“ kompiliavimo ir paruošimo BASH skriptas:

   ```
   ./build.sh sql
   ```

5. Patikrinta ar diegimas atliktas teisingai ir „MacroBase SQL“ įrankis veikia korektiškai pasinaudojus demonstraciniais duomenimis:

   ```
   bin/macrobase-sql -f sql/demo.sql
   ```

1.3. Eksperimento vykdymo eiga

Šiame poskyryje aprašoma eksperimento vykdymo eiga naudojant IPSS pateiktus 2014-2018 metų duomenų rinkinius naudojantis „MacroBase SQL“ analitinį įrankį.

1.3.1. Konfiguracija

„MacroBase SQL“ paleidimui buvo naudojama komandinė eilutė, kuriai paduodami SQL formato failai. Šiuose failuose nurodomi tokia informacija:

-

1.3.2. Eksperimento eiga

Šiame skyriuje aprašyta eksperimento eiga

Rezultatai ir išvados

Rezultatų ir išvadų dalyje išdėstomi pagrindiniai darbo rezultatai (kažkas išanalizuota, kažkas sukurta, kažkas įdiegta), toliau pateikiamos išvados (daromi nagrinėtų problemų sprendimo metodų palyginimai, siūlomos rekomendacijos, akcentuojamos naujovės). Rezultatai ir išvados pateikiami sunumeruotų (gali būti hierarchiniai) sąrašų pavidalu. Darbo rezultatai turi atitikti darbo tikslą.

Literatūra

[Ast16] Anthony Asta. Observability at twitter: technical overview, part i. 2016. URL: https://blog.twitter.com/engineering/en_us/a/2016/observability-at-twitter-technical-overview-part-i.html (tikrinta 2019-03-10).

[BGR+17] Peter Bailis, Edward Gan, Kexin Rong ir Sahaana Suri. Prioritizing attention in fast data: principles and promise. *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research*. Stanford Infolab, 2017.

[EMC14] Dell EMC. The digital universe of opportunities: rich data and the increasing value of the internet of things. 2014. URL: URL:%20http://www.emc.com/leadership/%20digital-universe/.

[Mak17] Aybeniz S. Aliyeva Makrufa Sh. Hajirahimova. About big datameasurement methodologies and indicators. *I.J.Modern Education and Computer Science*, p. 1–9, 2017-10. DOI: 10.5815/ijmecs.2017.10.01.

[PFT+15] Tuomas Pelkonen, Scott Franklin, Justin Teller, Paul Cavallaro, Qi Huang, Justin Meza ir Kaushik Veeraraghavan. Gorilla: a fast, scalable, in-memory time series database. *VLDB Endowment - Proceedings of the 41st International Conference on Very Large Data Bases*, p. 1816–1827, Kohala Coast, Hawaii. Facebook Inc., 2015. (Tikrinta 2019-03-10).

[rrtar] Lietuvos Respublikos ryšių reguliavimo tarnyba. Interneto prieigos stebėsenos sistema 2014-2018 metais. URL: http://matavimai.rrt.lt/about_archive.html (tikrinta 2019-05-20).

[Sim71] Herbert A. Simon. Designing organizations for an information-rich world. in computers, communications, and the public interest. Martin Greenberger, redaktorius, *Computers, Communication, and the Public Interest*, p. 37–72, 1971.

[Smo14] Sandy Smolan. The human face of big data. Trumpametražis dokumentinis filmas, 2014-04-10. URL: https://humanfaceofbigdatafilm.com/ (tikrinta 2019-05-15).

[Tec19] Techopedia. Anomaly detection. 2019. URL: https://www.techopedia.com/definition/30297/anomaly-detection (tikrinta 2019-03-07).

[Woo15] Alex Woodie. Kafka tops 1 trillion messages per day at linkedin. 2015. URL: https://www.datanami.com/2015/09/02/kafka-tops-1-trillion-messages-per-day-at-linkedin/ (tikrinta 2019-03-10).

Santrumpos

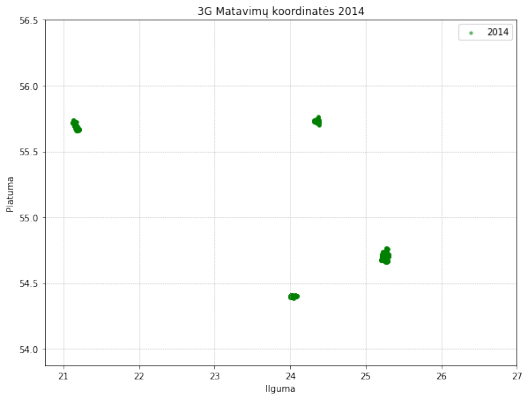
- LRTC
- CSV
- 3G
- LTE
- MAC adresas
- IPSS
- URL
- API
- HTTP
- JSON
- JRE
- JDK
- BASH
- SQL

Sąvokų apibrėžimai ir santrumpų sąrašas sudaromas tada, kai darbo tekste vartojami specialūs paaiškinimo reikalaujantys terminai ir rečiau sutinkamos santrumpos.

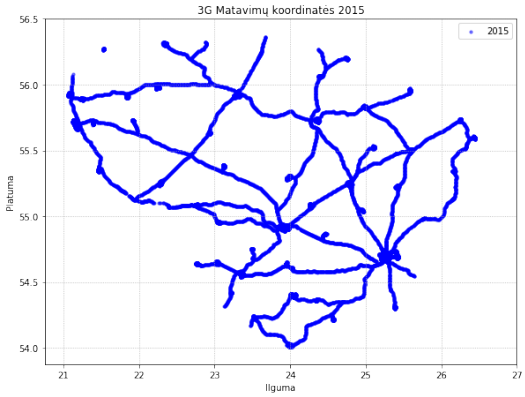
Priedas nr. 1

IPSS matavimų koordinatės 2014–2018 metais

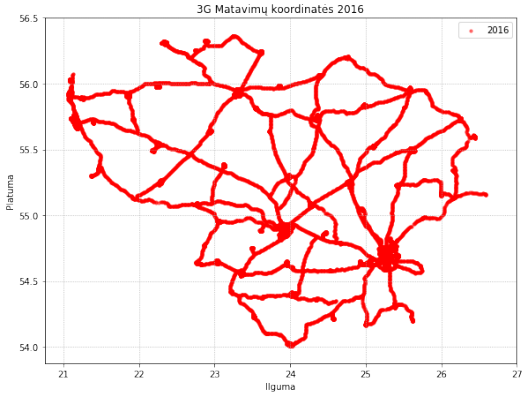
IPSS **3G** ryšio technologijos matavimai 2014–2018 metais. Viso 136003 įrašai.



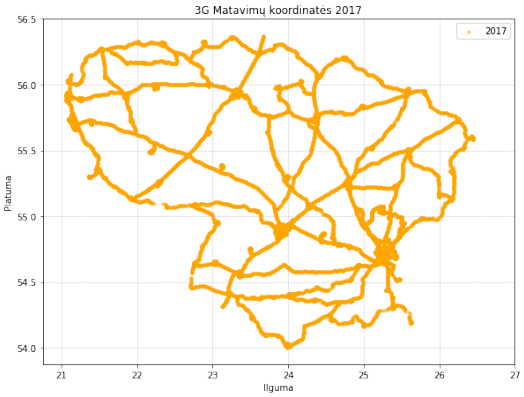
4 pav. 3G Matavimai 2014 metais. Viso – 2245 matavimai



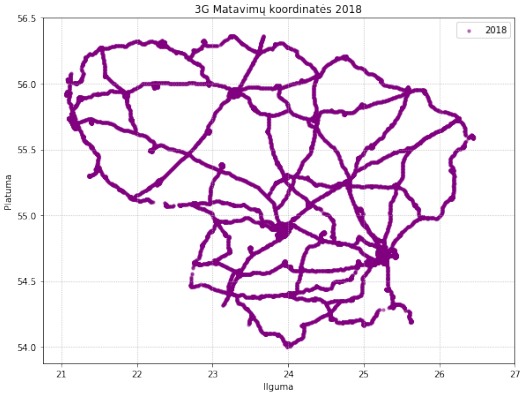
5 pav. 3G Matavimai 2015 metais. Viso – 19427 matavimai



6 pav. 3G Matavimai 2016 metais. Viso – 34769 matavimai

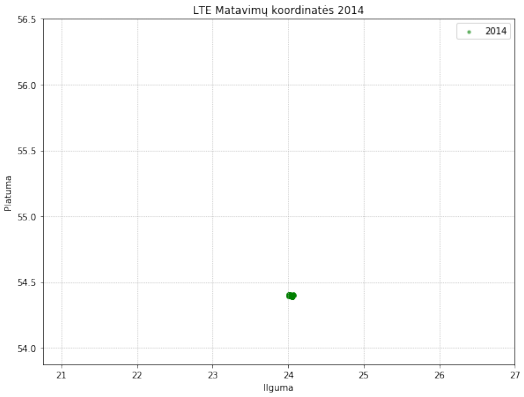


7 pav. 3G Matavimai 2017 metais. Viso – 38789 matavimai

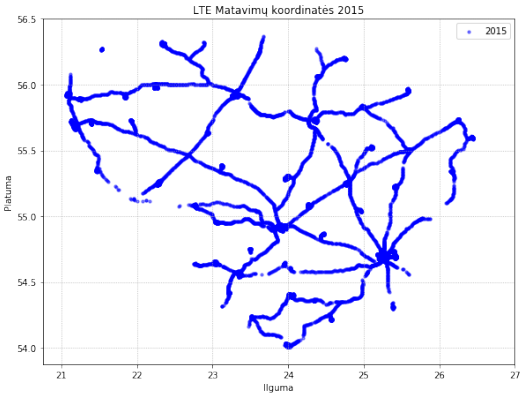


8 pav. 3G Matavimai 2018 metais. Viso – 40773 matavimai

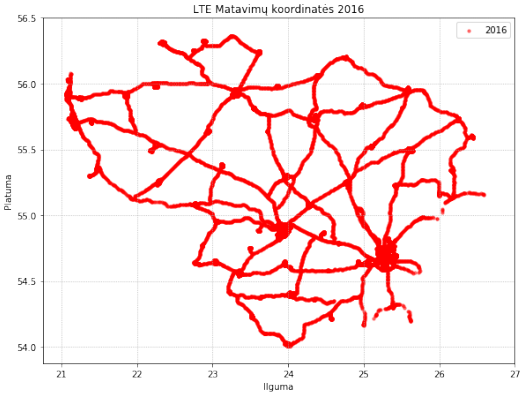
IPSS **LTE** ryšio technologijos matavimai 2014–2018 metais. Viso 157522 įrašai.



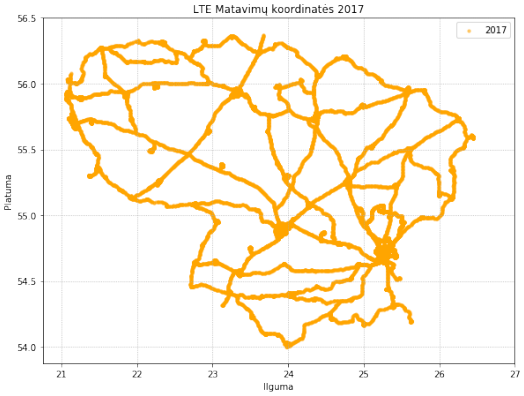
9 pav. LTE Matavimai 2014 metais. Viso – 232 matavimai



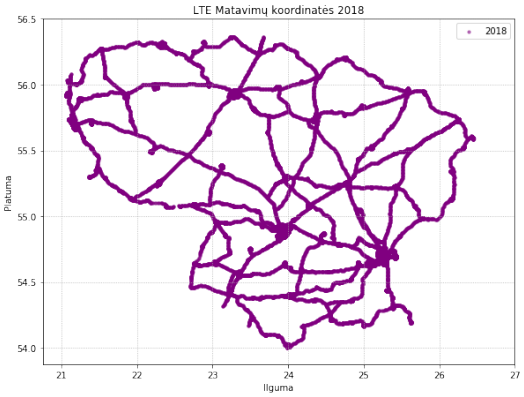
10 pav. LTE Matavimai 2015 metais. Viso – 13962 matavimai



11 pav. LTE Matavimai 2016 metais. Viso – 40738 matavimai

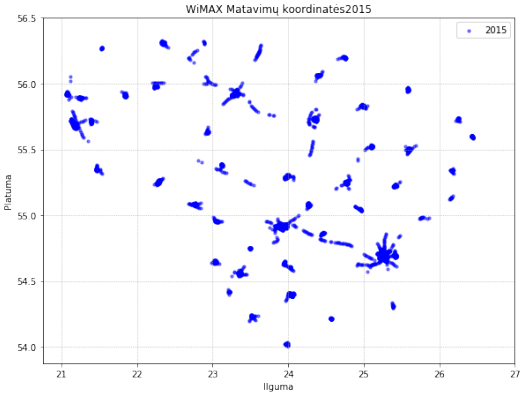


12 pav. LTE Matavimai 2017 metais. Viso – 49672 matavimai

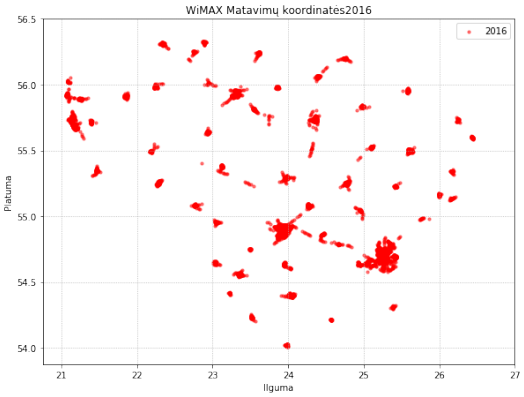


13 pav. LTE Matavimai 2018 metais. Viso – 52918 matavimai

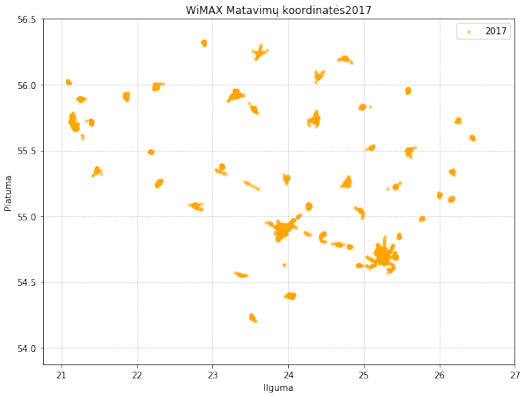
IPSS **WiMAX** ryšio technologijos matavimai 2014–2018 metais. Viso 18166 įrašai.



14 pav. WiMAX Matavimai 2015 metais. Viso – 4307 matavimai



15 pav. WiMAX Matavimai 2016 metais. Viso – 7653 matavimai



16 pav. WiMAX Matavimai 2017 metais. Viso – 6206 matavimai

Priedas nr. 2

Pradinių IPSS CSV failų apdorojimo „Python“ skriptas

„Python“ skriptas „RenameAndRemoveFields.py“ skirtas iš duotųjų IPSS duomenų CSV failų atrinkti nereikalingus laukus ir pervadinti lietuviškus antraštės pavadinimus į vieno žodžio (vienos simbolių eilutės) angliškus pavadinimus bei sugeneruoti naujus CSV failus:

```
#!/usr/bin/env python
# coding: utf-8
import csv
import numpy as np
import pandas as pd
file_3G_csv = 'Data/Raw/Matavimai-3G.csv'
file_LTE_csv = 'Data/Raw/Matavimai-LTE.csv'
file_WiMAX_csv = 'Data/Raw/Matavimai-WiMAX.csv'
new_file_3G_csv = 'Data/Raw/raw-3G.csv'
new_file_LTE_csv = 'Data/Raw/raw-LTE.csv'
new_file_WiMAX_csv = 'Data/Raw/raw-WiMAX.csv'
new_3G_labels = ['date', 'time', 'latitude', 'longitude', 'provider', 'cell_id
    ', 'rssi', 'technology', 'speed']
new_LTE_labels = ['date', 'time', 'latitude', 'longitude', 'provider', '
    cell_id', 'rssi', 'speed']
new_WiMAX_labels = ['date', 'time', 'latitude', 'longitude', 'base_station_id'
    , 'rssi', 'speed']
useful_3G_cols = [0, 1, 2, 3, 4, 5, 7, 8]
useful_LTE_cols = [0, 1, 2, 3, 4, 5, 10]
useful_WiMAX_cols = [0, 1, 2, 4, 5, 8]
def read_csv(file_name):
    read = pd.read_csv(file_name)
    x = np.array(read)
    return x
def create_new_file(file_name, x, useful_cols, new_labels):
    line_count = 0
    with open(file_name, mode='w') as csv_file:
        writer = csv.writer(csv_file, delimiter=',', quotechar='"', quoting=
    csv.QUOTE_MINIMAL)
        writer.writerow(new_labels)
        line_count = 0
        for row in x:
            useful_row = [row[i] for i in useful_cols]
            date, time = useful_row[0].split()
```

```
            useful_row[0] = date
            useful_row.insert(1, time)
            writer.writerow(useful_row)
            line_count += 1
    print(f'Processed {line_count} lines.')
    return line_count
def read_and_write(file_name, new_file_name, new_labels, useful_cols):
    x = read_csv(file_name)
    create_new_file(new_file_name, x, useful_cols, new_labels)
    print('Done')
read_and_write(file_3G_csv, new_file_3G_csv, new_3G_labels, useful_3G_cols)
read_and_write(file_LTE_csv, new_file_LTE_csv, new_LTE_labels, useful_LTE_cols
    )
read_and_write(file_WiMAX_csv, new_file_WiMAX_csv, new_WiMAX_labels,
    useful_WiMAX_cols)
```

Priedas nr. 3

Atrinktų IPSS CSV failų duomenų papildymo aukščio virš jūros lygio ir suapvalintų reikšmių laukais „Python“ skriptas

„Python“ skriptas skirtas jau atrinktiems CSV failams pridėti papildomus laukus: aukščio virš jūros lygio (gaunamas iš koordinačių), suapvalintų koordinačių reikšmių bei suapvalintos aukščio reikšmės. Šiems duomenims yra sugeneruojamas nauji CSV failai:

```
#!/usr/bin/env python
# coding: utf-8
import csv
import numpy as np
import pandas as pd
import urllib.parse as up
import urllib.request as ur
import json
def read_csv(file_name):
    read = pd.read_csv(file_name)
    X = np.array(read)
    labels = np.array(read.columns)
    return X, labels
def get_coord_indexes(columns):
    lat = np.where(columns == 'latitude')[0][0]
    lng = np.where(columns == 'longitude')[0][0]
    return lat, lng
def get_coords(x, lat_index, lng_index):
    lats = x[:,lat_index]
    lngs = x[:,lng_index]
    return lats, lngs
def create_formated_location_steps(lats, lngs, r_step):
    prev_r = 0
    l = len(lats)
    locations = []
    for r in range(r_step, l + r_step, r_step):
        locs = []
        for i in range(prev_r, r):
            if i >= l:
                break
            locs.append(str(lats[i]) + ',' +  str(lngs[i]))
        prev_r = r
```

```
33          locations.append("|".join(locs))
34      return locations
35  def get_elevations(locations, main_url, token):
36      elevations = np.array([])
37      a = 0
38      for loc in locations:
39          params = {'locations': loc, 'access-token' : token}
40          full_url = main_url + up.urlencode(params)
41          request = ur.urlopen(full_url)
42          contents = request.read()
43          request.close()
44          jdata = json.loads(contents)
45          el = np.array([j['elevation'] for j in jdata])
46          elevations = np.concatenate((elevations, el))
47          a += 1
48          if a % 100 == 0:
49              print(f'Processed altitude requests: {a}')
50      return elevations
51  def create_new_X(x, lat, lng, els, c_round_digits, alt_round_parts=1):
52      new_X = []
53      for i in range(len(x)):
54          item = x[i]
55          new_item = []
56          new_item.append(round(item[lat], c_round_digits))
57          new_item.append(round(item[lng], c_round_digits))
58          el = els[i]
59          new_item.append(el)
60          round_el = round(round(el / 10 * alt_round_parts) * 10 /
    alt_round_parts, 2)
61          new_item.append(round_el)
62          complete_item = np.append(np.array(item), np.array(new_item))
63          new_X.append(complete_item)
64      new_X = np.array(new_X)
65      return new_X
66  def write_csv(file_name, x, labels, positinos):
67      with open(file_name, mode='w') as new_csv_file:
68          writer = csv.writer(new_csv_file, delimiter=',', quotechar='"',
    quoting=csv.QUOTE_MINIMAL)
69          writer.writerow(labels)
70          for item in x:
```

```
            correct_pos_item = [item[p] for p in positions]
            writer.writerow(correct_pos_item)
def create_new_labels_and_positions(labels, lat, lng):
    positions = np.arange(len(labels)).tolist()
    labels_list = labels.tolist()
    last_i = len(labels)
    i = max(lat, lng)+1
    positions.insert(i, last_i+2)
    positions.insert(i+1, last_i)
    positions.insert(i+2, last_i+1)
    positions.insert(i+3, last_i+3)
    labels_list.insert(i, 'altitude')
    labels_list.insert(i+1, 'lat')
    labels_list.insert(i+2, 'lng')
    labels_list.insert(i+3, 'alt')
    labels_list = np.array(labels_list)
    return positions, labels_list
def create_new_csv_file(file_name, new_file_name, request_step, token,
    main_url, round_digits=2, round_altitude_parts=1):
    X, labels = read_csv(file_name)
    lat, lng = get_coord_indexes(labels)
    lats, lngs = get_coords(X, lat, lng)
    locations = create_formated_location_steps(lats, lngs, request_step)
    els = get_elevations(locations, main_url, token)
    new_X = create_new_X(X, lat, lng, els, round_digits, round_altitude_parts)
    positions, new_labels = create_new_labels_and_positions(labels, lat, lng)
    write_csv(new_file_name, new_X, new_labels, positions)
    print('Done')
file_3G_csv = 'Data/Raw/raw-3G.csv'
file_LTE_csv = 'Data/Raw/raw-LTE.csv'
file_WiMAX_csv = 'Data/Raw/raw-WiMAX.csv'
new_file_3G_csv = 'Data/Processed/altitude-11-3G.csv'
new_file_LTE_csv = 'Data/Processed/altitude-11-LTE.csv'
new_file_WiMAX_csv = 'Data/Processed/altitude-11-WiMAX.csv'
token = "nIXn7WWJTSeguG1aEuSioR2hM5PqAZUcRfGqCOV9RJvDwVJIYwDdDmwVDKvs5FYm"
main_url = "https://api.jawg.io/elevations?"
request_step = 100
round_digits = 1
round_altitude = 1
create_new_csv_file(file_3G_csv, new_file_3G_csv, request_step, token,
```

```
    main_url, round_digits, round_altitude)
110 create_new_csv_file(file_LTE_csv, new_file_LTE_csv, request_step, token,
    main_url, round_digits, round_altitude)
111 create_new_csv_file(file_WiMAX_csv, new_file_WiMAX_csv, request_step, token,
    main_url, round_digits, round_altitude)
```

Priedas nr. 4

Atrinktų IPSS CSV failų duomenų papildymo metų ir valandu laukais „Python“ skriptas

„Python“ skriptas skirtas jau atrinktiems CSV failams pridėti papildomus laukus: metų ir valandų (gaunamas iš datos ir laiko). Šiems duomenims yra sugeneruojamas nauji CSV failai:

```
#!/usr/bin/env python
# coding: utf-8
import csv
import numpy as np
import pandas as pd
def read_csv(file_name):
    read = pd.read_csv(file_name)
    X = np.array(read)
    labels = np.array(read.columns)
    return X, labels
def get_datetime_indexes(columns):
    dd = np.where(columns == 'date')[0][0]
    tt = np.where(columns == 'time')[0][0]
    return dd, tt
def create_new_x(x, dd, tt):
    i = max(dd, tt)
    new_x = []
    for row in x:
        new_row = row.tolist()
        new_row.insert(i+1, row[dd][:4])
        new_row.insert(i+2, row[tt][:2])
        new_x.append(new_row)
    new_x = np.array(new_x)
    return new_x
def create_new_l(l, dd, tt, yl, hl):
    i = max(dd, tt)
    new_l = l.tolist()
    new_l.insert(i+1, yl)
    new_l.insert(i+2, hl)
    new_l = np.array(new_l)
    return new_l
def write_csv(file_name, x, labels):
    with open(file_name, mode='w') as new_csv_file:
```

```
34          writer = csv.writer(new_csv_file, delimiter=',', quotechar='"',
    quoting=csv.QUOTE_MINIMAL)
35          writer.writerow(labels)
36          for row in x:
37              writer.writerow(row)
38  def your_hour_change_file(file_name, new_file_name, year_label, hour_label):
39      x, l = read_csv(file_name)
40      dd, tt = get_datetime_indexes(l)
41      new_x = create_new_x(x, dd, tt)
42      new_l = create_new_l(l, dd,tt, year_label, hour_label)
43      write_csv(new_file_name, new_x, new_l)
44      print('Done')
45  file_3G_csv = 'Data/Processed/altitude-3G.csv'
46  file_LTE_csv = 'Data/Processed/altitude-LTE.csv'
47  file_WiMAX_csv = 'Data/Processed/altitude-WiMAX.csv'
48  new_file_3G_csv = 'Data/Processed/yh-3G.csv'
49  new_file_LTE_csv = 'Data/Processed/yh-LTE.csv'
50  new_file_WiMAX_csv = 'Data/Processed/yh-WiMAX.csv'
51  year_label = 'year'
52  hour_label = 'hour'
53  your_hour_change_file(file_3G_csv, new_file_3G_csv, year_label, hour_label)
54  your_hour_change_file(file_LTE_csv, new_file_LTE_csv, year_label, hour_label)
55  your_hour_change_file(file_WiMAX_csv, new_file_WiMAX_csv, year_label,
    hour_label)
```

Priedas nr. 5

Atrinktų IPSS papildomų CSV failų generavimo „Python“ skriptas

„Python“ skriptas skirtas jau atrinktiems CSV failams sugeneruoti skirtingų koordinačių ir aukščio virš jūros lygio reikšmių apvalinimo duomenų rinkinius. Šiems duomenims yra sugeneruojamas nauji CSV failai:

```
#!/usr/bin/env python
# coding: utf-8
import csv
import numpy as np
import pandas as pd
def read_csv(file_name):
    read = pd.read_csv(file_name)
    X = np.array(read)
    labels = np.array(read.columns)
    return X, labels
def get_col_indexes(columns):
    latitude = np.where(columns == 'latitude')[0][0]
    longitude = np.where(columns == 'longitude')[0][0]
    altitude = np.where(columns == 'altitude')[0][0]
    lat = np.where(columns == 'lat')[0][0]
    lng = np.where(columns == 'lng')[0][0]
    alt = np.where(columns == 'alt')[0][0]
    return latitude, longitude, altitude, lat, lng, alt
def change_x(x, latitude, longitude, altitude, lat, lng, alt, rd, ra):
    for i in range(len(x)):
        item = x[i]
        item[lat] = round(item[latitude], rd)
        item[lng] = round(item[longitude], rd)
        item[alt] = round(round(item[altitude] / 10 * ra) * 10 / ra, 2)
def name_new_file(file_name, rd, ra):
    return file_name.format(rd, ra)
def write_csv(file_name, x, labels):
    with open(file_name, mode='w') as new_csv_file:
        writer = csv.writer(new_csv_file, delimiter=',', quotechar='"',
    quoting=csv.QUOTE_MINIMAL)
        writer.writerow(labels)
        for row in x:
            writer.writerow(row)
def perform_all_tests(file_name, new_file_name, rdt, rat):
    x, l = read_csv(file_name)
```

```
    latitude, longitude, altitude, lat, lng, alt = get_col_indexes(1)
    for rd in rdt:
        for ra in rat:
            full_name = name_new_file(new_file_name, rd, ra)
            change_x(x, latitude, longitude, altitude, lat, lng, alt, rd, ra)
            write_csv(full_name, x, 1)
            print(f'{rd}{ra} Done.')
        print('Cycle.')
    print('All done.')
file_3G_csv = 'Data/Processed/yh-3G.csv'
file_LTE_csv = 'Data/Processed/yh-LTE.csv'
file_WiMAX_csv = 'Data/Processed/yh-WiMAX.csv'
new_file_3G_csv = 'Data/Processed/3G/{}{}-3G.csv'
new_file_LTE_csv = 'Data/Processed/LTE/{}{}-LTE.csv'
new_file_WiMAX_csv = 'Data/Processed/WiMAX/{}{}-WiMAX.csv'
round_digit_tests = range(3+1)
round_altitude_tests = range(1, 5+1)
perform_all_tests(file_3G_csv, new_file_3G_csv, round_digit_tests,
    round_altitude_tests)
perform_all_tests(file_LTE_csv, new_file_LTE_csv, round_digit_tests,
    round_altitude_tests)
perform_all_tests(file_WiMAX_csv, new_file_WiMAX_csv, round_digit_tests,
    round_altitude_tests)
```

Priedas nr. 6

Eksperimentinio palyginimo rezultatai

1 lentelė. Lentelės pavyzdys

Algoritmas	$\bar{x}$	σ^2
Algoritmas A	1.6335	0.5584
Algoritmas B	1.7395	0.5647