LOKITIEDON ANALYSOINTIMENETELMÄT

Poikkeavuuksien havaitseminen

LAB-AMMATTIKORKEAKOULU
Insinööri (ylempi AMK)
Digitaaliset ratkaisut
Kevät 2020
Jarno Kurlin

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Kurlin, Jarno	Opinnäytetyö, YAMK	Kevät 2020
	Sivumäärä 64 sivua,4 liitesivua	

Työn nimi

LOKITIEDON ANALYSOINTIMENETELMÄT
Poikkeavuuksien havaitseminen

Tutkinto

Digitaaliset ratkaisut, Insinööri (ylempi AMK)

Tiivistelmä

Tässä opinnäytetyössä tutustutaan big dataan, sen analysointimenetelmiin sekä ongelmiin, joita datan kerääminen aiheuttaa. Kun datan rakenne ja ominaisuudet ymmärretään, voidaan perehtyä erilaisiin analysointimenetelmiin ja siihen, kuinka nämä analysointimenetelmät toimivat.

Opinnäytetyön tavoitteena on tutkia, kuinka suuresta lokitietomassasta voidaan löytää poikkeavuuksia. Lisäksi tavoitteena on selvittää, kuinka poikkeavuuksia etsivä algoritmi saadaan löytämään poikkeavuudet yksittäisen käyttäjän toiminnassa kirjautumistietoja analysoimalla. Poikkeavuuksien havaitseminen ei vielä itsessään paranna yrityksen tietoturvaa, joten algoritmin kehityksen ohessa kolmantena tavoitteena on pohtia, kuinka yrityksen tietoturvaa voidaan parantaa käyttäjien kirjautumistietojen tarkastelun avulla.

Käytännön osuudessa perehdytään käytettävissä olevan lokidatan rakenteeseen sekä ominaisuuksiin. Teoriaosuudessa läpi käydyistä algoritmeista testataan neljää erilaista datan luokittelualgoritmia, joiden avulla pyritään löytämään datasta sellaisia poikkeavuuksia, joita tutkimalla yrityksen tietoturvaa voitaisiin parantaa.

Työn lopputuloksena toteutettiin R-ohjelmointikielen avulla algoritmi, joka kykenee ennustamaan käyttäjän kirjautumisen onnistumisen viikonpäivän, päivän tietyn tunnin ja kirjautumisen lähdemaan perusteella. Ennusteen kokonaistarkkuudeksi saatiin 95,4 %, ja sen avulla pystytään parantamaan yrityksen tietoturvaa esimerkiksi vaatimalla vahvempaa kirjautumiskäytäntöä sellaisilta henkilöiltä, joiden kirjautumiset eivät ole ennusteen mukaisia.

Asiasanat

Big data, datan analysointi, tilastolliset menetelmät, koneoppiminen, R-ohjelmointi

Abstract

Author(s)	Type of publication	Published
Kurlin, Jarno	Master's thesis	Spring 2020
	Number of pages 64 pages, 4 pages of appendices	

Title of publication

Log data analyzing methods
Detecting anomalies

Name of Degree

Master of Engineering, Digital Solutions

Abstract

This Master's thesis deals with Big Data, data analyzing methods and issues which might rise, when data is stored in massive amounts. Individual data analyzing methods and practices can be studied only after the data's structure and features are known.

The objective of the thesis was to study how anomalies can be detected from a large log data mass. Another objective was to investigate how the algorithm can detect anomalies which concern only an individual user. The anomaly detection algorithm does not improve enterprise security on its own, which leads to the third objective, how enterprise security can be improved by studying user login information.

The structure and features of the available log data were examined in the applied study section. In this section, four pre-studied data classification algorithms were used to find anomalies the study of which could help improve enterprise security.

The outcome from this design science study was an algorithm which was developed using the R language. The algorithm predicts the success of a user's login attempt based on the day of the week, the hour of the day and the country of origin. The total accuracy of the algorithm was 95.4%. Enterprise security can be improved with this algorithm by requiring stronger authentication policies from the users that do not correspond with the prediction.

Keywords

Big Data, data analyzing, statistical methods, machine learning, R programming

SISÄLLYS

1 JOHDANTO 1
1.1 Tutkimusmenetelmä 1
1.2 Työn rakenne 3
2 BIG DATA 4
2.1 Big datan määritelmiä 4
2.2 Big datan hyödyntämisteknologioita 5
2.2.1 NoSQL-tietokannat 5
2.2.2 Hadoop 6
2.2.3 Elasticsearch 11
2.3 Big data ja yksityisyydensuoja 12
3 DATAN ANALYSOINTIMENETELMÄT 14
3.1 Tilastolliset menetelmät 14
3.1.1 Keskiluvut - moodi, mediaani ja keskiarvo 14
3.1.2 Fraktiilit 14
3.1.3 Keskihajonta 15
3.1.4 Normaalijakauma 15
3.1.5 Hypoteesien testaus tilastollisessa päättelyssä 16
3.2 Koneoppiminen 18
3.2.1 Ohjattu oppiminen 20
3.2.2 Ohjaamaton oppiminen 20
3.3 Koneoppimisen algoritmit 22
3.3.1 K-nearest neighbour 22
3.3.2 Support Vector Machine – tukivektorikone 23
3.3.3 Naive Bayes 24
3.3.4 Decision Tree – päätöspuu 26
3.3.5 Random Forest – satunnainen metsä 28
3.3.6 K-Means-algoritmi 29
3.3.7 Hierarkkinen klusterointi (Hierarchical clustering) 31
4 CASE: POIKKEUKSIEN HAVAITSEMINEN KIRJAUTUMISDATASTA 35
4.1 Graylog-lokitietojärjestelmän asennus 35
4.2 Lokitiedon kerääminen Office365-palvelusta Graylog-järjestelmään 35
4.3 Lokitietojen analysointi Graylog-järjestelmässä 37
4.4 Lokitietojen haku Graylog-järjestelmästä RStudioon 38

4.5 Lokitietojen analysointi RStudiossa....42
4.5.1 Lokitietoihin tutustuminen....42
4.5.2 Lokitietojen analysointi hierarkkisen klusteroinnin avulla....47
4.5.3 Lokitietojen analysointi normaalijakauman avulla....48
4.5.4 Lokitietojen analysointi hypoteesien avulla....51
4.5.5 Lokitietojen analysointi päätöspuiden avulla....53
5 YHTEENVETO....61
LÄHTEET....63
LIITTEET....65

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on tutustua big dataan, sen hyödyntämistapoihin ja siihen, minkälaisia ongelmia datan kerääminen aiheuttaa yksityisyyden suojan kannalta. Kun ymmärrämme datan rakenteen ja ominaisuudet, voimme perehtyä erilaisiin analysointimenetelmiin ja siihen, kuinka niiden avulla voidaan löytää poikkeavuudet suuresta lokitietomassasta.

Työn tavoitteena on tutkia erilaisia malleja poikkeavuuksien havaitsemiseen, ja pohtia, kuinka näiden havaintojen avulla voidaan parantaa yrityksen tietoturvaa. Tutkimuskysymykset ovat:

- Millä tavoin suuresta lokitietomassasta voidaan löytää poikkeavuuksia?

Jotta tähän kysymykseen voidaan vastata, tulee ensin selvittää, minkälaisesta big datasta on kyse. Tämän vuoksi tulee perehtyä big dataan, sen ominaisuuksiin ja sen analysointitapoihin syvällisemmin.

- Kuinka poikkeavuuksia etsivä algoritmi saadaan huomioimaan poikkeavuudet yksittäisen käyttäjän kirjautumistietojen perusteella?

Järjestelmään kirjautuminen esimerkiksi keskellä arkipäivää ei todennäköisesti näy poikkeuksena suuresta datamassasta, mikäli analysointialgoritmi ei huomioi yksittäisen käyttäjän normaalia toimintatapaa. Tätä varten tulee kehittää algoritmi, joka mukautuu käyttäjäkohtaiseksi.

- Kuinka lokitietojen analyysin avulla voidaan parantaa yrityksen tietoturvaa tarkastelemassa käyttäjien kirjautumistietoja?

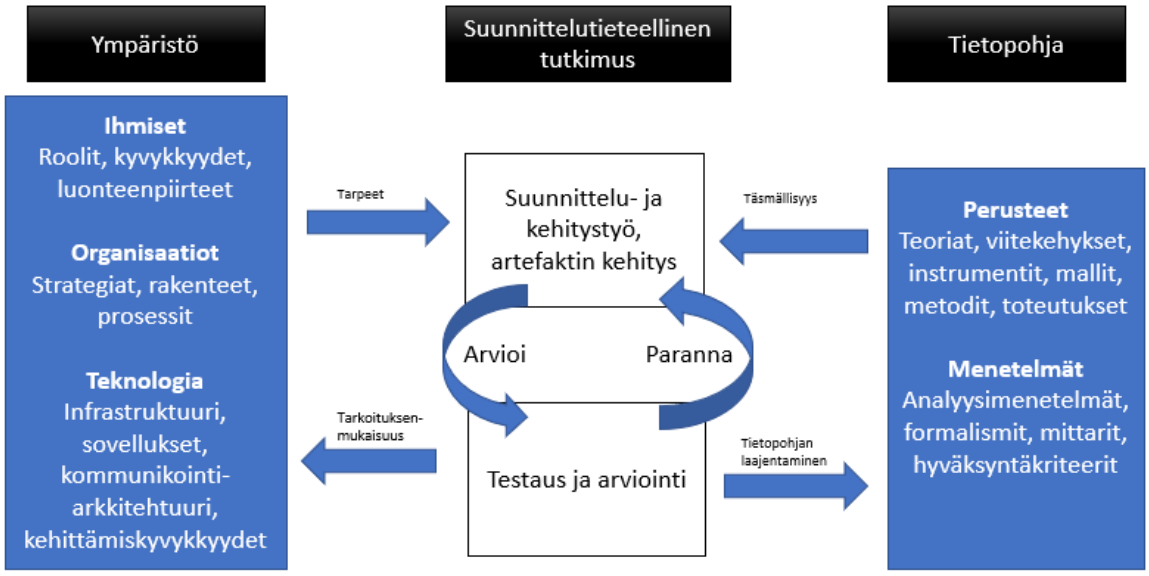
Poikkeavuuksien havaitseminen itsessään ei vielä itsessään paranna tietoturvaa. Vastataksemme tähän viimeiseen tutkimuskysymykseen tulee meidän ensin vastata kahteen ensimmäiseen tutkimuskysymykseen, ja tämän jälkeen tehdä erilaisia toimenpiteitä, joiden avulla tietoturvaa saadaan parannettua asteittain.

1.1 Tutkimusmenetelmä

Tämän opinnäytetyön tutkimusmenetelmäksi valittiin suunnittelutieteellinen tutkimus. Järvinen & Järvisen mukaan suunnittelutieteellisestä tutkimuksesta voidaan käyttää nimitystä

soveltava tutkimus - soveltavan tutkimuksen ajatuksena on hyödyntää perustutkimuksen tuloksia innovaatioiden toteutuksessa. Suunnittelutieteen tarkoituksena on lisäksi tuottaa sellaista tietoa, jota eri alojen ammattilaiset voivat käyttää hyväkseen suunnitteluongelmien ratkaisua etsiessä. (Järvinen & Järvinen 2004, 103.)

Suunnittelutieteellinen tutkimuksen viitekehys tietojärjestelmien kehityksessä muodostuu ympäristöstä, kehitettävästä artefaktista sekä tietopohjasta (kuvio 1). Ympäristö koostuu ihmisistä, organisaatioista, sekä teknologioista. Ympäristöstä kumpuavat tarpeet tai ongelmat, joita suunnittelutieteellisen tutkimuksen avulla aletaan ratkaisemaan. Tutkimus tulee sitoa ympäristöstä tulevaan tarpeeseen arvioimalla sen tarkoituksenmukaisuutta ja toimivuutta erilaisten kenttäkokeiden, simulaatioiden ja kokeilujen avulla. Suunnittelutieteellisen tutkimuksen tulos on aina itse artefakti tai prosessi, jonka avulla ratkaistaan tämä ympäristöstä tuleva tarve tai ongelma. Artefakti on siis aina jokin hyödyke tai työkalu, IT-järjestelmiä kehitettäessä se on usein ohjelmisto. Artefaktia kehitettäessä käytetään tietopohjassa olevia perusteita, teorioita ja osaamista, ja toisaalta kehityksen mukana tullutta tietoa lisätään tietopohjaan aina tutkimuksen edetessä. (Hevner 2004, 79-80.)



KUVIO 1. Informaatiosysteemien tutkimuksen viitekehys (Järvinen & Järvinen 2004, 103)

Iteraatiot ovat tärkeässä osassa artefaktin kehitystä. Suunnittelu- ja kehityssyklin valmistuttua tulee sen hetkistä tulosta testata ja arvioida. Testauksen ja arvioinnin perusteella voidaan tietopohjaan viedä uutta tietoa, jonka avulla artefaktia parannetaan. Lisäksi tulee

testata artefaktin sopivuutta ympäristöön, ja tarvittaessa tuoda ympäristöstä mukaan uusia tarpeita. (Hevner 2004, 78-80.)

1.2 Työn rakenne

Työ on jaettu kolmeen osioon, joista kaksi ensimmäistä ovat teoriaa läpi käyviä, ja kolmas käytännön case-esimerkki. Ensimmäiseksi tutustutaan yleisesti big data -käsitteeseen ja käydään lävitse erilaisia big datan hyödyntämisteknologioita. Big data osuudessa käsitellään lisäksi datan tallennuksen yksityisyyden suojaa sekä tutustutaan lyhyesti siihen, miten EU:n yleinen tietosuoja-asetus (GDPR) vaikuttaa datan keräämiseen.

Kokonaisuuden toisessa osiossa tutustutaan erilaisiin datan analysointimenetelmiin aloittaen matemaattisista tilastomenetelmistä. Tämän jälkeen käsitellään koneoppimista ja yleisimpiä koneoppimisen algoritmeja aina päätöspuusta hierarkkiseen klusterointiin. Jokaisen algoritmin toiminta käydään osiossa läpi esimerkkien kanssa niin, että lukija pystyy käsittämään toiminnan käytännön tasolla. Tämän osion tarkoituksena on vastata ensimmäiseen tutkimuskysymykseen siitä, millä tavoin suuresta lokitietomassasta voidaan löytää poikkeavuuksia.

Kolmannessa osiossa osuudessa tutustutaan case-esimerkkiin lokitietojen analysoinnista Graylog-lokijärjestelmään kerättyjen kirjautumistietojen perusteella. Järjestelmään kerätään käyttäjien kirjautumistietoja Offce365-järjestelmästä, ja tämän datan avulla pyritään löytämään poikkeavuuksia toisessa osuudessa läpi käytyjen analysointimenetelmien avulla. Analysoinnissa käytetään hyväksi RStudio-ohjelmistoa ja tutustutaan samalla siihen, kuinka Graylog-järjestelmän dataa voidaan viedä analysoitavaksi, ja mitä analysointitoimintoja järjestelmä itsessään mahdollistaa. Kolmannen osuuden tavoitteena on selvittää, kuinka poikkeavuuksia etsivä algoritmi saadaan huomioimaan yksittäisen käyttäjän poikkeukset kirjautumistiedoissa, ja kuinka näiden kirjautumislokitietojen analyysin avulla voidaan parantaa yrityksen tietoturvaa.

2 BIG DATA

2.1 Big datan määritelmiä

> *Data on perusyksikkö ja sitä voi olla monessa eri muodossa. Data on numeroita, tekstiä, kuvia, lukuja tai vaikkapa videoita. Kun dataan liitetään jokin merkitys, siitä saadaan informaatiota. Kun informaatiota tulkitaan, saadaan tietoa. Edelleen, kun tieto sidotaan ympäröivään kokonaisuuteen, saadaan tietämystä, jolla on merkitystä. (Kananen & Puolitaival 2019, 71.)*

Big dataksi kutsutaan sellaista tietoa, jota ei voida analysoida perinteisin datankäsittelytyökaluin. Tämä tieto on usein raakamuodossa tai vain puoliksi jäsenneltyä, eikä ole aivan selvää, kannattaako tällaista dataa edes säilyttää. IBM määrittelee big datan piirteiksi määrän, vaihtelevuuden ja nopeuden. Määrät ovat erittäin isoja, datan laatu vaihtelee jäsennellystä jäsentelemättömään ja dataa kertyy erittäin nopeasti. (Zikopoulos et al. 2012, 3-9.)

Big dataksi kutsutaan sellaista tietoa, joka vaatii uudenlaista data-arkkitehtuuria ja uusia analysointitekniikoita, joiden avulla päästään käsiksi uusiin liiketoiminnan arvoa kasvattaviin lähteisiin. Suurin osa big datasta on jäsentelemätöntä tai osittain jäsenneltyä, jonka vuoksi hajautetut laskentaympäristöt ovat suositelluin tapa käsitellä tällaista dataa. (EMC Education Services 2015, 2-3.)

Big dataksi kutsutaan sellaista toimintaa, jonka digitaalisen jäljen analysoinnilla voidaan tulla viisaammiksi. Itse datalla ei vielä itsessään ole arvoa, vaan arvo muodostuu siitä, mitä datalla voidaan tehdä. Suurin arvo muodostuu kyvykkyydestä käsitellä jäsentelemättömän dataa ja tehdä viisaita päätöksiä sen pohjalta. Marrin mukaan big dataa pitäisi kutsua älykkääksi dataksi, ja itse termi big data tulee katoamaan. (Marr 2015, 9-10.)

Yhteisenä tekijänä näissä kaikissa tulkinnoissa on se, että itse data on useimmiten puolijäsenneltyä tai täysin jäsentelemätöntä. Tällaisen datan analysointiin tarvitaan uudenlaisia välineitä ja paljon laskentatehoa. Yhteisenä tekijänä on myös se, että big datan suurin anti on kasvattaa liiketoiminnan arvoa.

Big dataa kertyy esimerkiksi kauppojen kassajärjestelmistä, hakukoneista, sosiaalisesta mediasta ja henkilöautojen suorituskykyä mittaavista järjestelmistä. Kertyneen datan avulla esimerkiksi kauppa voi tarjota grillaustarvikkeita ostaneelle asiakkaalle automaattisesti alennuskuponkia grillaustarvikkeisiin, kun sää on aurinkoinen ja asiakas sattuu olevaan kaupan lähistöllä. Eri teollisuusalat keräävät dataa ratkaistakseen erilaisia ongelmia. Kaupat yrittävät usein oppia tuntemaan asiakkaansa lisätäkseen myyntiä, kun taas

teollisuus yrittää sujuvoittaa toimintaansa vähentääkseen hävikkiä. Tätä varten jokaisen tuotantoketjuun osallistuvan ihmisen ja koneen tehokkuutta tulee pystyä mittaamaan. (Marr 2015, 11-13.)

2.2 Big datan hyödyntämisteknologioita

2.2.1 NoSQL-tietokannat

NoSQL-tietokannat (Not only SQL) ovat laajasti käytetty teknologia sen helpon skaalautuvuuden ja yksinkertaisen rakenteen vuoksi (Achari 2015, 4). Acharin mukaan kolme vuosikymmentä käytössä olleet relaatiotietokannat eivät skaalaudu riittävän kustannustehokkaasti, eivätkä ne NoSQL-kantoihin verrattuna tarjoa perusominaisuuksissaan vikasietoisuutta tai helppoa skaalautuvuutta. Achari jakaa erilaiset NoSQL-tietokannat tiedon organisointitavan perusteella avain-arvokantoihin, sarakekantoihin, dokumenttikantoihin ja graafikantoihin.

Avain-arvo-tyyppisissä tietokannoissa data on tallennettu aina tietyn avaimen arvoksi (Achari 2015, 5). Avain voi olla Acharin mukaan joko itse keksitty, synteettinen tai automaattisesti generoitu. Achari listaa mahdollisiksi arvoiksi esimerkiksi XML, JSON tai BLOB-tyyppisen datan. Suosituimpia avain-arvo-tietokantoja ovat Redis, Amazon DynamoDB, Microsoft Azure Cosmos DB sekä Memcached (DB-engines 2020).

Saraketyyppisissä tietokannoissa data on tallennettu relaatiotietokannoista poiketen sarakkeittain, jolloin kirjoitus ja lukuoperaatiot tehdään aina sarakkeittain rivien sijaan (Achari 2015, 5). Saraketyyppisten tietokantojen etuina ovat Acharin mukaan tiedon pakkausmahdollisuudet sekä erittäin korkea suorituskyky sen vuoksi, ettei kaikkea rivin tietoa tarvitse lukea keskusmuistiin. Achari listaa saraketyyppisten tietokantojen eduksi myös korkean skaalautuvuuden. Suosituimpia saraketyyppisiä tietokantoja ovat Cassandra, HBase sekä Microsoft Azure Cosmos DB (DB-engines 2020).

Dokumenttityyppiset tietokannat on suunniteltu dokumenttityyppisen tiedon tallennukseen, joka tarkoittaa sitä, että itse arvo on strukturoitua tai binäärimuodoista dataa kuten XML, YAML, JSON, BSON, PDF, XLSX tai DOCX (Achari, 2015, 5). Achari kertoo dokumenttityyppisten tietokantojen suurimpana etuna olevan se, että talletetun datan arvosta voidaan etsiä tietoa tehokkaasti. Suosituimpia dokumenttityyppisiä tietokantoja ovat MongoDB, Amazon DynamoDB ja Couchbase (DB-engines 2020).

Graafityyppiset tietokannat on suunniteltu kuvailemaan datan suhteita toisiinsa (Achari, 2015, 5). Achari kertoo relaatiotietokantojen vaativan monimutkaisia yhdistelyoperaatioita, jotka on graafitietokannoissa toteutettu tehokkaasti monimutkaisten algoritmien avulla.

Acharin mukaan näiden algoritmien avulla voidaan tehdä ennustamista, käyttäjäseurantaa sekä lyhyimpien reittien etsintää. Suosituimpia graafityyppisiä tietokantoja ovat Neo4j, Microsoft Cosmos DB sekä ArangoDB (DB-engines 2020).

2.2.2 Hadoop

Avoimeen lähdekoodiin pohjautuva Hadoop on laajimmin käytetty big datan prosessointia varten kehitetty ohjelmistoalusta (Achari 2015, 13-15). Achari kertoo Hadoopin olevan erittäin vikasietoinen järjestelmä sen vuoksi, että datan tallennussijaintien määrä on vapaasti konfiguroitavissa. Itse perusjärjestelmä koostuu Acharin mukaan hajautetusta tiedostojärjestelmästä nimeltään HDFS, sekä datan prosessointia rinnakkain suorittavasta MapReducesta (kuva 1). Achari listaa Hadoopin eduiksi myös kustannustehokkuuden sekä suoriutumisen monimutkaisesta analytiikasta. Järjestelmä toimii perinteisillä tietokoneilla, eikä se vaadi yksittäiseltä klusterin koneelta suuria tehoja prosessoinnin jakautuessa useille tietokoneille.

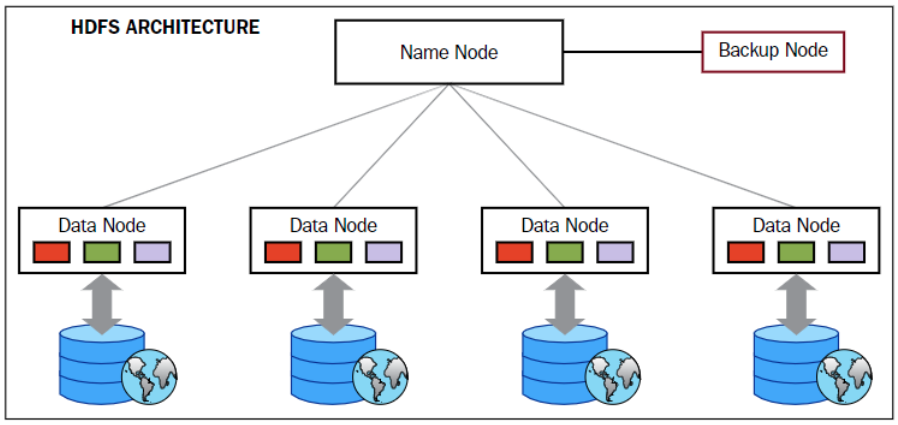
KUVA 1. Hadoop-järjestelmän pääkomponentit (Achari 2015, 15)

Hadoopin perusjärjestelmän ympärille on kehitetty useita ohjelmistoja, jotka helpottavat kokonaisuuden hallintaa (Achari 2015, 16-17). Acharin kertoo Apache Software Foundationin kehittävän Hadoopin perustoimintojen lisäksi myös isoa määrää muita komponentteja, joita on listattuna taulukkoon 1. Acharin mukaan nämä komponentit helpottavat Hapoodin kanssa toimimista ja parantavat sen hallintaominaisuuksia.

TAULUKKO 1. Apache Software Foundationin kehittämiä Hadoopin komponentteja (Achari 2015, 16)

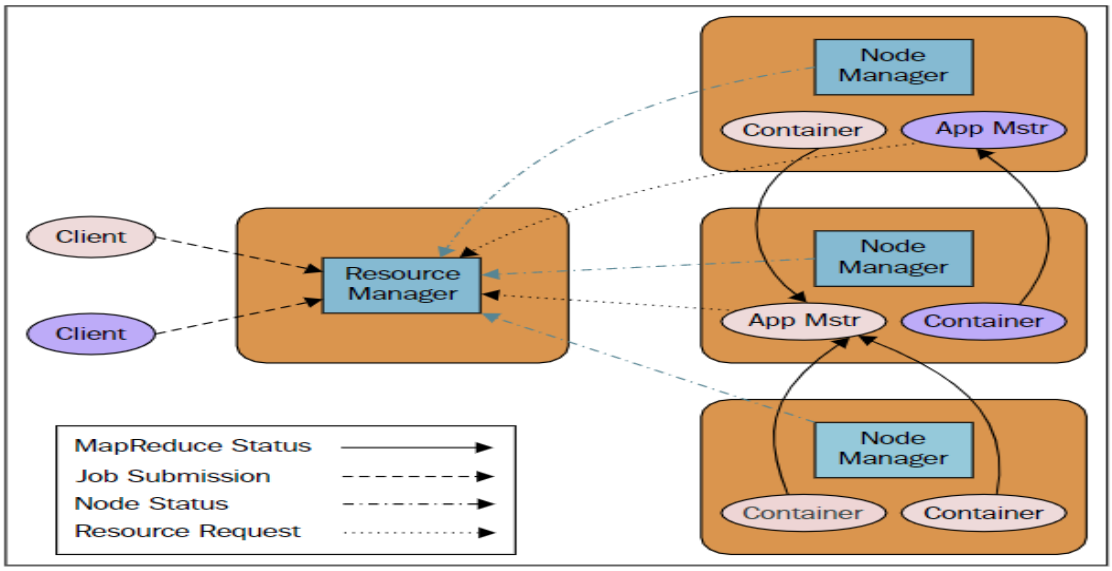
Layer	Utility/Tool name
Distributed filesystem	Apache HDFS
Distributed programming	Apache MapReduce
	Apache Hive
	Apache Pig
	Apache Spark
NoSQL databases	Apache HBase
Data ingestion	Apache Flume
	Apache Sqoop
	Apache Storm
Service programming	Apache Zookeeper
Scheduling	Apache Oozie
Machine learning	Apache Mahout
System deployment	Apache Ambari

HDFS-tiedostojärjestelmän tärkeimpiä ominaisuuksia ovat erittäin skaalautuvuus ja vikasietoisuus sekä itse toimintamalli, jossa laskentaprosessi tuodaan datan luo sen sijaan, että dataa liikuteltaisiin edestakaisin datasolmujen välillä (Achari 2015, 34-37). Acharin mukaan HDFS arkkitehtuuri koostuu koko järjestelmän toimintaa ohjaavasta NameNode-prosessista sekä yhdestä tai useammasta DataNode-prosessista, jotka sisältävät itse datan (kuva 2). Achari kertoo NameNoden huolehtivan myös tiedostojärjestelmän luku- ja kirjoitusoperaatioista, joka tekee siitä erittäin kriittisen. NameNode-prosessista voidaan tehdä vikasietoisempi kahdentamalla se erilliselle palvelimelle Secondary NameNode- tai BackupNode-prosessien avulla. Acharin mukaan HDFS-järjestelmässä DataNode-solmut raportoivat tilansa säännöllisin väliajoin NameNodelle. Mikäli NameNode ei vastaanota DataNoden tilaa, se merkitään kuolleeksi ja data replikoidaan tarpeen mukaan toiselle DataNodelle.



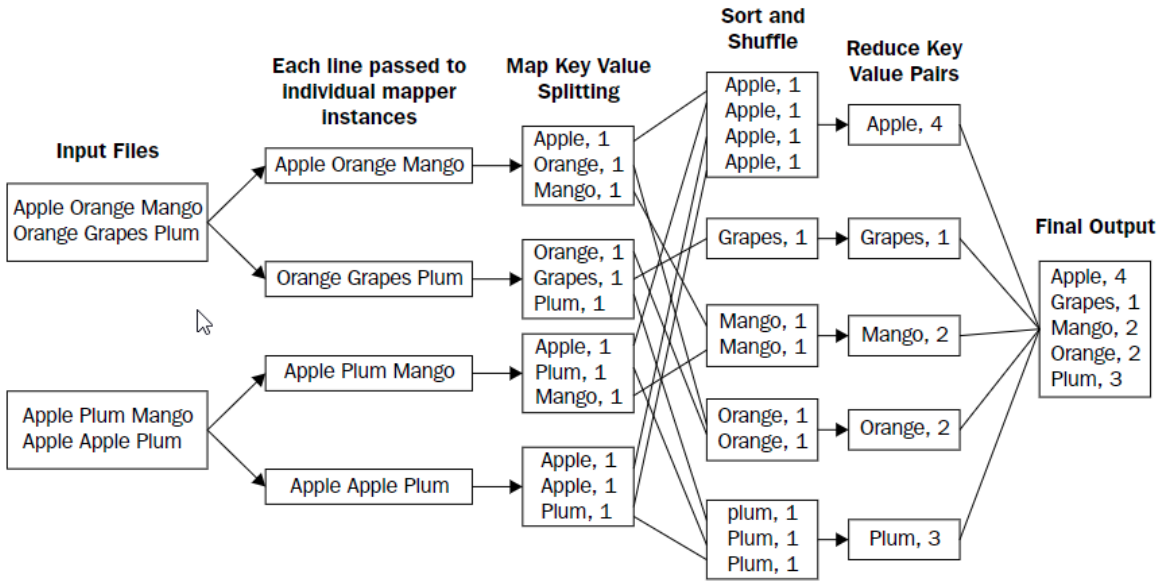
KUVA 2. HDFS-tiedostojärjestelmän arkkitehtuuri (Achari 2015, 35)

YARN (Yet Another Resource Negotiator) on huolehtinut Hadoopin 2.0 versiosta lähtien klusterin resurssien hallinnasta vikasietoisella tavalla (Achari 2015, 61-64). Acharin mukaan YARN koostuu ResourceManager-, NodeManager- ja ApplicationMaster-komponenteista (kuva 3). Achari kertoo ResourceManagerin olevan pääprosessi, joka vastaanottaa työt asiakkailta, ja huolehtii resurssien allokoinnista klusterin jäsenille. NodeManager prosessi huolehtii Acharin mukaan solmun säilöistä ja autentikoinnista, sekä valvoo klusterin jokaisen solmun tilaa raportoiden edelleen tietonsa ResourceManagerille. Achari jatkaa ApplicationMaster-komponentin valvovan suoritettavaa sovellusta raportoiden resurssien käytöstä NodeManagerille.



KUVA 3. YARN-arkkitehtuuri (Achari 2015, 63)

MapReducen avulla Hadoop ympäristössä voidaan prosessoida dataa rinnakkain usealla klusterin solmulla yhtäaikaisesti (Achari 2015, 46-51). Achari kertoo MapReducen koostuvan vaiheista Mapper, Shuffle and sort sekä Reducer. Acharin mukaan Mapper vaiheessa datalle suoritetaan käyttäjän kirjoittama Mapper-funktio, jonka lopputuloksena on avain-arvoparien joukko. Seuraavassa Shuffle and sort-vaiheessa avain-arvoparien joukot lajitellaan avaimen mukaiseen järjestykseen. Lopuksi Achari kertoo Producer-vaiheen summaavan saman avaimen lukumäärät yhteen ja tallentavan lopputuloksen HDFS-tiedostojärjestelmään. Kuvassa 4 kuvataan MapReducen toimintaa esimerkillä, jossa lasketaan, kuinka monta kertaa sanat esiintyvät aineistossa.



KUVA 4. MapReducen toimintaperiaate (Achari 2015, 48)

Apache Hiven avulla voidaan luoda MapReduce-funktioita käyttäjäystävällisemmällä tavalla kuin perinteisesti ohjelmoimalla (Achari 2015, 67-68). Achari kirjoittaa, että Hiven avulla Hadoopiin saadaan luotua datavarasto, johon voidaan suorittaa SQL-tyyppisiä HiveQL-kyselyitä (Achari 2015, 83-84). Acharin mukaan HiveSQ-kyselyn muuntaminen MapReduce-funktioksi tapahtuu tarkastelemalla ensin kyselyn oikeellisuus ja vertaamalla siinä olevia kenttiä Metastoressa kuvattuun tietokantarakenteeseen. Mikäli kaikki on kunnossa, Achari kerto kyselyn siirtyvän Query compilerille joka luo kyselyn pohjalta map- ja reduce-funktiot. Lopuksi Execution engine suorittaa nämä funktiot Hadoopissa.

HBase on Hadoopia hyväksi käyttävä hajautettu ja hyvin skaalautuva relaatioriippumaton datavarasto. Projektin tavoitteena on tarjota alusta tietokannoille, jotka saattavat sisältää

miljardeja rivejä ja miljoonia sarakkeita. HBase pohjautuu Googlen BigTable-tallennusjärjestelmään ja se mahdollistaa reaaliaikaiset luku- ja kirjoitustapahtumat Hadoopissa sijaitsevaan dataan. (Hbase 2020.)

HBase on yksinkertaistetusti avain-arvo-tyyppinen tietokanta, jonka jokainen riviavain on uniikki (deRoos et al. 2014, 182). Toisaalta deRoos et al. kirjoittaa HBasen olevan myös saraketyyppinen tietokanta, koska data ryhmitellään ja tallennetaan aina sarakeperheisiin, jotka tallennetaan yhteen loogiseen sarakkeeseen. deRoos et al. mukaan HBasessa datan versiointi tapahtuu automaattisesti ja versiotunnisteeksi tallentuu aina sen hetkinen Unix-aikaleima. Taulukossa 2 esitetään esimerkki HBase-tietokannan sarakeperheistä.

TAULUKKO 2. Esimerkki HBase datamallista (deRoos et al. 2014, 183)

Table 12-2 Logical View of Customer Contact Information in HBase

Row Key	*Column Family: {Column Qualifier:Version:Value}*
00001	CustomerName: {'FN': 1383859182496:'John', 'LN': 1383859182858:'Smith', 'MN': 1383859183001:'Timothy', 'MN': 1383859182915:'T'} ContactInfo: {'EA': 1383859183030:'John.Smith@xyz.com', 'SA': 1383859183073:'1 Hadoop Lane, NY 11111'}
00002	CustomerName: {'FN': 1383859183103:'Jane', 'LN': 1383859183163:'Doe', ContactInfo: { 'SA': 1383859185577:'7 HBase Ave, CA 22222'}

Sarakeperheet (Column family) tulee määritellä HBase-tietokannalle taulun luonnin yhteydessä (deRoos et al. 2014, 184). deRoos et al. mukaan sarakeperheiden tarkoitus on kasvattaa suorituskykyä ryhmittelemällä sarakeperheet samaan levyjärjestelmään, jolloin samaan asiaan liittyvä data kannattaa ryhmitellä samaan sarakeperheeseen. deRoos et al. suosittelee yhteen tauluun talletettavien dataperheiden maksimimääräksi enintään kolmea.

Sarakemääreiden (Column qualifiers) tarkoituksena on antaa datalle selväkielinen tunniste (deRoos et al. 2014, 184). deRoos et al. kirjoittaa HBase järjestelmän generoivan

tunnisteen automaattisesti, mikäli sitä ei anneta tietoa syötettäessä. Saraketunnisteiden lukumäärää tai tunnisteen merkkimäärää ei ole heidän mukaansa rajoitettu, mutta pitkät saraketunnisteet saattavat käyttää paljon tallennustilaa.

HBasessa avain-arvoparista puhuttaessa itse avain saattaa olla pelkästään riviavain, jolloin dataa noudettaessa palautetaan kaikki sarakeperheet kaikkine sarakemääreineen. Toisaalta avaimeksi voidaan määritellä riviavaimen lisäksi myös sarakeperhe, sarakemääre tai jopa sarakemääreen versionumero, jolloin dataa voidaan hakea hyvin yksityiskohtaisesti. Tällöin kuormitetaan järjestelmän hakukonetta enemmän, mutta toisaalta dataa liikutetaan vähemmän. Mikäli yksi riviavain sisältää tuhansia sarakkeita, saattaa tarkempi rajaus hyödyttää lopputulosta huomattavasti. (deRoos et al. 2014, 185.)

2.2.3 Elasticsearch

Elasticsearch on Apache Lucenen päälle rakennettu hajautettu hakukone, jota käyttävät mm. NASA, Wikipedia ja Github (Dixit 2016, 1-2). Apache Lucene on Dixitin mukaan erittäin nopea hakukirjasto, mutta se ei itsessään skaalaudu yhtä isäntäkonetta laajemmaksi ratkaisuksi, eikä tarjoa käyttöön kuin Java-pohjaisen API:n. Dixit kirjoittaa Elasticsearchin tärkeimmiksi ominaisuuksiksi REST-pohjaisen JSON API:n, hajautetun ja vikasietoisen arkkitehtuurin, dynaamisen tietokantarakenteen sekä yksinkertaisen kyselykielen.

Koska Elasticsearch tarjoaa erittäin hyvät hakuominaisuudet, käytetään sitä myös laajasti NoSQL-datavarastona (Dixit 2016, 5-6). Dixitin mukaan Elasticsearchin komponentteja voidaan verrata osittain perinteisen relaatiotietokannan komponentteihin, vaikka se ei itsessään mahdollista monimutkaisia datan suhteita relaatiotietokantojen malliin. Elasticsearch tukee kuitenkin SQL-tyylisiä taulujen yhdistämisiä sisäkkäisten ja vanhempi-lapsi -suhteiden avulla (Dixit 2016, 124).

Elasticsearchin hakukoneen nopeus perustuu käänteiseen indeksointiin, jonka avulla hakukone tietää nopeasti, mistä dokumentista hakutulos löytyy (Dixit 2016, 23). Käänteinen indeksointi jakaa Dixitin mukaan esimerkiksi tallennetun tekstin eri sanoihin, ja tallentaa tämän jälkeen jokaisen sanan sijainnin indeksitietoihinsa. Dixit kirjoittaa hakukoneen etsivän hakusanaa indeksin avaimista, johon saattaisi tallentua samoja sanoja esimerkiksi pienillä ja isoilla kirjaimilla kirjoitettaessa. Tätä varten Dixitin mukaan tarvitaan dokumenttianalyysiä, joka muodostaa järkeviä avaimia indeksiä varten. Erilaisia valmiita analysaattoreita on Dixitin mukaan useita, ja käyttäjä voi halutessaan luoda myös oman räätälöidyn analysaattorinsa. Taulukossa 3 esitetään esimerkkilauseen avulla erilaisten standardianalysaattoreiden toiminta käänteisen indeksin avaimia luotaessa.

TAULUKKO 3. Erilaisten standardianalysaattorien toiminta (Dixit 2016, 27)

i HATE when spiders sit on the wall and act like they pay 1000$ rent

Standard Analyzers	Analyzed Text
standard	i 1 hate 2 when 3 spiders 4 sit 5 on 6 the 7 wall 8 and 9 act 10 like 11 they 12 pay 13 1000 14 rent 15
simple	i 1 hate 2 when 3 spiders 4 sit 5 on 6 the 7 wall 8 and 9 act 10 like 11 they 12 pay 13 rent 14
whitespace	i 1 HATE 2 when 3 spiders 4 sit 5 on 6 the 7 wall 8 and 9 act 10 like 11 they 12 pay 13 1000$ 14 rent 15
stop	i 1 hate 2 when 3 spiders 4 sit 5 wall 8 act 10 like 11 pay 13 rent 14
keyword	i HATE when spiders sit on the wall and act like they pay 1000$ rent 1

2.3 Big data ja yksityisyydensuoja

Dataliikenteen solmukohdissa toimivat palveluntarjoajat voivat datan analysoinnilla luoda käyttäjäprofiilin jokaisesta henkilöstä sen perusteella, kuinka ja millä päätelaitteella he toimivat palveluissa (Salo 2014, 54). Salon mukaan profiilin ollessa riittävän laaja, se voidaan yhdistää yksilön identifioivaan tietoon ja tällöin myös reaalimaailman identiteettiin. Esimerkiksi vähittäiskaupan tapauksessa palveluntarjoaja saattaa tuntea asiakkaan paljon paremmin kuin asiakas itse haluaisi. Salo kertoo toisena esimerkkinä sosiaalisen median palvelut, joiden tarkoituksena on profiloida käyttäjiä ja tarjota näitä tietoja myös lisäpalveluja tuottaville kolmansille osapuolille.

Aiemmin historiassa palveluiden käyttäjäehdot ovat olleet pitkiä, eivätkä kuluttajat ole saanut selvyyttä siihen, mihin yritykset niille luovutettuja tietoja käyttävät (Salo 2014, 54-55). Salon mukaan useimmissa tapauksissa käyttäjä on hyväksynyt käyttöehdot lukematta niitä, jolloin palvelun käytöstä on syntynyt juridinen sopimus. Lisähaasteita aiheuttavat myös dataliiketoimintaa säätelevien lakien tulkinnanvaraisuus ja se, että palvelut ovat usein globaaleja.

Nykyisin tietosuoja-asetuksen mukaan yrityksiltä vaaditaan selkokielinen sopimus, jossa kuvataan keskeisimmät asiat henkilötietojen käsittelystä. Tietosuojavaltuutetun toimiston mukaan tiedot on annettava tiiviisti esitetyssä, helposti ymmärrettävässä ja helposti saatavilla olevassa muodossa. Henkilölle on kerrottava selkeästi mitä tietoja hänestä kerätään,

mitä tarkoitusta varten ne kerätään, millä tavoin tietoja käsitellään ja millaisia oikeuksia hänellä on. Lisäksi henkilötietojen keräämiselle täytyy olla olemassa käsittelyperuste. (Tietosuoja 2020.)

EU:n yleinen tietosuoja-asetus GDPR määrittää, että henkilöllä on oikeus saada pyydettäessä tietoja henkilötietojensa käsittelystä, saada pääsy tietoihinsa ja oikaista tietoja. Lisäksi henkilöllä on oikeus poistaa tiedot ja näin tulla unohdetuksi. Tietosuoja-asetuksen mukaan henkilöllä on myös oikeus rajoittaa henkilötietojen käsittelyä siten, että tietoja saadaan käsitellä esimerkiksi vain henkilön suostumuksella. Tiedot tulee myös saada siirtää toiseen rekisteriin, mikäli se on teknisesti mahdollista. Muita oikeuksia ovat oikeus vastustaa tietojen käsittelyä, sekä oikeus olla joutumatta automaattisen päätöksenteon kohteeksi. (Tietosuoja 2020.)

Oikeuksien tekninen toteuttaminen on erittäin haasteellista (Salo 2014, 55-56). Salo kirjoittaa asetuksen saaneen vastustusta esimerkiksi hakukoneyritysten suunnasta, mutta selkeä ajatus on siirtää valta yrityksiltä kuluttajille. Salon mukaan kuluttajan oikeuksien seuraamisen vastuu jää kuitenkin aina kuluttajalle itselleen, ja hyväksyttävyyden rajat ovat usein veteen piirrettyjä viivoja.

3 DATAN ANALYSOINTIMENETELMÄT

3.1 Tilastolliset menetelmät

Tilastolliset menetelmät kuuluvat nykyaikaisen tutkimuksen tärkeimpiin menetelmiin (Nummenmaa et al. 2014, 3-4). Nummenmaan et al. mukaan tilastollisten menetelmien avulla voidaan kuvailla tutkimusaineistoa, arvioida riskejä sekä luoda ennusteita päätöksenteon tueksi. He arvioivat epävarmuuksien käsittelyn ja arvioinnin yhdeksi tilastollisten menetelmien peruskivistä.

3.1.1 Keskiluvut - moodi, mediaani ja keskiarvo

Mikäli havaintoaineistossa on useita erilaatuisia muuttujia, on moodi ainut keskiluku, jolla aineiston jakauman keskikohtaa voidaan kuvata. Aineistoa luokiteltaessa on moodi se luokka, jonka alkioiden lukumäärä on suurin. Ilmoitettaessa keskiluku järjestetyn asteikon ominaisuudelle, käytetään kuvaamisessa usein mediaania. Mediaani saadaan jakamalla järjestetyn havaintoasteikon luvut kahteen osaan niin, että molempiin osiin tulee yhtä paljon alkioita. Mikäli asteikossa on pariton määrä lukuja, tulee mediaaniksi keskelle jäänyt alkio. Jos taas alkioita on parillinen määrä ja alkioiden arvot ovat lähellä toisiaan esitetään mediaanina vain toinen luvuista. Alkioiden ollessa kaukana toisistaan esitetään mediaanina joko molemmat, tai jätetään mediaani kokonaan esittämättä. (Nummenmaa et al. 2014, 71-72.)

Aritmeettista keskiarvoa käytetään kuvaamaan havaintoarvojen suuruutta siinä tapauksessa, että mitattava ominaisuus jaettaisiin yhtä suuriin osiin. (Nummenmaa et al. 2014, 75-76). Nummenmaa et al. mukaan kaikki havaintoarvot lasketaan yhteen ja jaetaan sitten havaintojen lukumäärällä. He kirjoittavat keskiarvon kuvaavan huonosti havaintojoukkoa, joka sisältää selvästi joukosta poikkeavia arvoja. Näissä tapauksissa Nummenmaa kehottaa käyttämään kuvaavana keskilukuna mediaania.

3.1.2 Fraktiilit

Mikäli ominaisuutta voidaan mitata järjestysasteikolla, voidaan siitä esittää lisätietoja fraktiilien ja mediaanin avulla (Nummenmaa et al. 2014, 77-78). Fraktiilien ideana on Nummenmaan et al. mukaan aineiston jakaminen neljään eri prosentuaaliseen osioon, joista käytetään nimityksiä alakvartiili (25 %), mediaani (50 %) sekä yläkvartiili (75 %). Nummenmaa et al. kertoo usein käytettävän myös desiilejä, joiden avulla aineisto jaetaan vielä pienimpään (10 %) ja suurimpaan (90 %) osaan. Fraktiilien tavoitteena on rakentaa kuvaaja, josta voidaan paikallistaa nopeasti erilaiset fraktiilit.

3.1.3 Keskihajonta

Aineiston havaintoarvojen keskimääräistä hajontaa voidaan mitata keskihajonnan avulla. Otoksen keskihajontaa laskettaessa lasketaan ensin aineiston keskiarvo, ja tämän lasketaan jokaiselle havaintoarvolle etäisyys keskiarvosta yhtälön 1 mukaisesti. Jotta keskiarvoa pienemmistä etäisyyksiä ei muodostuisi negatiivia lukuja, korotetaan kaikki etäisyydet toiseen potenssiin. Lopuksi jaetaan kaikkien etäisyyksien summa havaintoarvojen lukumäärällä, josta on vähennetty yksi ja kumotaan neliöinnin vaikutus ottamalla lopputuloksesta neliöjuuri. Mikäli halutaan kuvata otoksen sijaan koko aineiston keskihajontaa, tulee jakajana käyttää aineiston havaintoarvojen lukumäärää n. Tilanteessa, jossa aineiston havaintoarvojen lukumäärä on yli 30, ei tällä jakajan muutoksella ole käytännön merkitystä. (Nummenmaa et al. 2014, 82-83.)

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \qquad [1]$$

missä x_i = havaintoarvo

$\bar{x}$ = aineiston keskiarvo

n = havaintoarvojen lukumäärä

3.1.4 Normaalijakauma

Normaalijakauma kuvaa tilastollisesti tyypillisen aineiston havaintoarvojen jakautumista odotusarvon suhteen (Nummenmaa et al. 2014, 145). Nummenmaa et al. kirjoittaa normaalijakaman olevan jakaumaperhe, joka sisältää useita erimuotoisia jakaumia, jotka kuitenkin pohjautuvat samanlaisiin yleisiin ominaisuuksiin. Normaalijakauma on Nummenmaan et al. mukaan tärkeä tilastotieteen jakauma siksi, että useat toisistaan riippumattomat tekijät jakautuvat lähes aina normaalijakauman mukaisesti. Tällaisia ominaisuuksia ovat esimerkiksi luonnon biologiset ominaisuudet sekä havaintovirheet missä tahansa mittauskokeessa. Muuttuja noudattaa normaalijakaumaa parametrein μ ja σ mikäli sen tiheysfunktio on muodoltaan yhtälön 2 mukainen:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \quad [2]$$

missä μ = odotusarvo

σ = keskihajonta

e = Neperin luku (~2,71828)

μ ja σ määrittävät normaalijakauman sijainnin sekä muodon x- ja y-akselin suhteen. Mikäli σ:n arvo on suuri, leviää jakauma laajemmalle, ja kuvaajasta tulee litistyneempi. (Nummenmaa et al. 2014, 146.)

Normaalijakauman todennäköisyydet voidaan määrittää tiheysfunktion avulla. Tiheysfunktio integrointi on työlästä käsin laskettaessa, jonka vuoksi se suoritetaan yleensä tietokoneella. Mikäli todennäköisyyksiä halutaan laskea käsin, käytetään yleensä hyväksi standardoitua normaalijakaumaa. Standardoidun normaalijakauman kertymäfunktioiden arvot on taulukoitu, ne se noudattaa aina tiheysfunktion parametreja μ = 0, σ = 1. Normaalijakauman kuvaajan mittayksikkönä on keskihajonta ja sitä noudattavan aineiston havainnoista 68,2 % on alle yhden keskihajonnan päässä odotusarvosta. (Nummenmaa et al. 2014, 147-148.)

3.1.5 Hypoteesien testaus tilastollisessa päättelyssä

Tilastolliseen päättelyyn kuuluu hypoteesien testaus, eli olettamusten todenperäisyyden varmentaminen (Nummenmaa et al. 2014, 175). Nummenmaan et al. mukaan hypoteesien testauksessa asetetaan ensin nollahypoteesi sekä vaihtoehtoinen hypoteesi. Nollahypoteesina pidetään olettamusta siitä, miten asian luullaan olevan, kun taas vaihtoehtoinen hypoteesi on päinvastainen olettamus. Seuraavaksi he kehottavat ottamaan perusjoukosta yhden tai useamman otoksen ja tekemään tilanteeseen soveltuvan tilastollisen testin. Tilastollisen testin valmistuttua Nummenmaan et al. mukaan tulkitaan testin tulos ja päätetään, jääkö nollahypoteesi voimaan, vai hylätäänkö se ja käytetään vaihtoehtoista hypoteesia.

Tilastolliseen testaukseen liittyy aina epävarmuus siitä, onko testin tulos oikein. Mikäli testin tulkinnan jälkeen nollahypoteesi hylätään, mutta se todellisuudessa pitäisikin hyväksyä, puhutaan hylkäämisvirheestä. Hylkäämisvirheen todennäköisyyttä kutsutaan merkitsevyystasoksi, jota merkitään symbolilla α. Tilasto-ohjelmissa hylkäämisvirheen todennäköisyyttä kuvataan p-arvolla, joka kertoo, onko tulos tilastollisesti erittäin merkitsevä, merkitsevä vai melkein merkitsevä. Taulukossa 4 listataan yleisesti käytettyjen

hylkäämisvirheiden todennäköisyydet. Haluttu merkitsevyystaso tulee määrittää aina ennen testin aloittamista. (Nummenmaa et al. 2014, 176-177.)

TAULUKKO 4. Hylkäämisvirheen todennäköisyydet selitteineen

Merkitsevyystaso	Selite
α ≤ 0,001	Tilastollisesti erittäin merkitsevä (riski 0,1 %)
0,001 < α ≤ 0,01	Tilastollisesti merkitsevä (riski 1 %)
0,01 < α ≤ 0,05	Tilastollisesti melkein merkitsevä (riski 5 %)

Yksi tilastollisen testauksen testeistä on nimeltään keskiarvotesti (Nummenmaa et al. 2014, 181). Nummenmaan et al. mukaan keskiarvotestissä otetaan perusjoukosta yli 30 alkion otos, josta lasketaan keskiarvo sekä keskihajonta. Seuraavaksi he ohjeistavat laskemaan testimuuttujan Z-arvon yhtälön 3 mukaisesti.

$$Z = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}} \qquad [3]$$

missä $\bar{x}$ = keskiarvo

μ_0 = odotusarvo

s = keskihajonta

n = otoslukumäärä

Kaksisuuntaisessa testissä nollahypoteesi hylätään, mikäli:

- Merkitsevyystasona 5 % ja Z < 1,96 tai Z > 1,96
- Merkitsevyystasona 1 % Z < 2,58 tai Z > 2,58
- Merkitsevyystasona 0,1 % ja Z < -3,30 tai Z > 3,30

Yksisuuntaisessa testissä nollahypoteesi hylätään, mikäli:

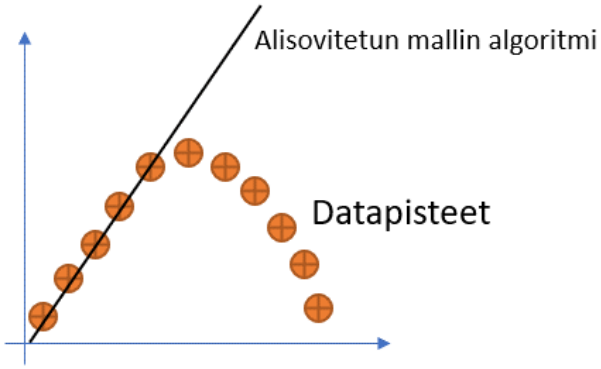
- Merkitsevyystasona 5 % ja Z < -1,64
- Merkitsevyystasona 1 % ja Z < -2,33
- Merkitsevyystasona 0,1 % ja Z < -3,09

3.2 Koneoppiminen

Datan analysoinnissa voidaan käyttää hyväksi myös tekoälyä eli tässä tapauksessa koneoppimista. Kone voi oppia ohjatusti, ohjaamattomasti tai vahvistusoppimisen kautta. Ohjattu oppiminen tarkoittaa usein neuroverkkoja sekä koneoppimisen menetelmiä, kun taas ohjaamattomassa oppimisessa käytetään ainoastaan koneoppimisen menetelmiä. Ohjastusta oppimisesta käytetään myös termiä syväoppiminen. Vahvistusoppiminen tarkoittaa nimensä mukaisesti sitä, että kone oppii yrityksen ja erehdyksen kautta sen jälkeen, kun sille on mallinnettu sopiva toimintaympäristö. Vahvistusoppiminen on näistä kolmesta oppimismuodosta ainoa, joka ei tarvitse oppiakseen isoa määrää dataa. (Kananen & Puolitaival 2019, 43-44.)

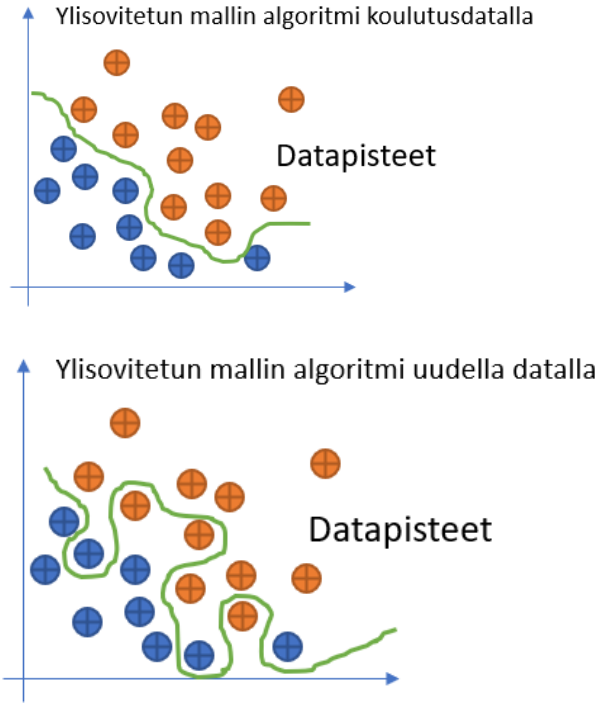
Koneälymallin kouluttaminen aloitetaan datan keräämisestä, jonka jälkeen kerätty data valmistellaan muokkaamalla sitä niin, että saadaan aikaiseksi eheä tietokanta. Mikäli valmistelu on tehty heikosti ovat lopputuloksetkin heikkoja. Valmisteltu data jaetaan kolmeen osaan, joista ensimmäistä, eli koulutusdatasettiä käytetään nimensä mukaisesti mallin kouluttamiseen. Testaus- ja validointidatasettien avulla testataan, että algoritmi toimii myös sellaisella datalla, jota se ei ole vielä koskaan nähnyt. Kun dataa on alle 100 000 näytettä, on suuntaan antavan koulutusdatan osuus hyvä olla 60 %, validointidatan 20 % ja testausdatan 20 %. Mikäli näytteitä on yli miljoona kappaletta, riittää validointidatan osuudeksi 1 % ja testausdatan osuudeksi 1 %, jolloin suurin osa datasta on itse koulutusdataa. (Kananen & Puolitaival 2019, 46-47.)

Koneoppimismallin sovittaminen tarkoittaa yksinkertaisimmillaan sitä, miten hyvin löydetään matemaattinen funktio, joka kuvaa aineistoa parhaiten. Sovittaminen on tasapainoilua mallin yksinkertaisuuden ja monimutkaisuuden välillä. Kun malli on liian yksinkertainen, kutsutaan sitä alisovitetuksi. Tällöin malli toimii suuntaa antavasti, mutta se ei pysty kuvailemaan tutkittavaa asiaa riittävällä tasolla. Liian yksinkertaisella mallilla sanotaan tällöin olevan liian iso bias, eli vinouma. Kuviossa 2 esitetään esimerkki alisovitetun mallin toiminnasta datapisteiden avulla. Alisovittamisen korjaaminen tapahtuu lisäämällä uusia ominaisuuksia malliin. (Kananen & Puolitaival 2019, 100-101.)



KUVIO 2. Alisovitettu koneoppimismalli

Ylisovitetussa koneoppimismallissa malli saattaa toimia hyvin mallin koulutusvaiheessa, mutta täysin uudella datalla malli muuttuu epätarkaksi. Mikäli ominaisuuksia on liikaa, muodostuu jokaisesta testitapauksesta oma yksittäinen tapauksensa. Tällöin tekoälymalli ei pysty yleistämään lainkaan, jolloin se ei toimi yleisellä tasolla. Ylisovittaminen voidaan korjata vähentämällä mallissa olevien ominaisuuksien määrää. Kuviossa 3 esitetään ylisovitetun mallin toiminta koulutusdatan ja täysin uuden datan avulla. (Kananen & Puolitaival 2019, 101.)



KUVIO 3. Mallin ylisovituksen vaikutus algoritmin toimintaan

Yleisesti sovituksessa kannattaa miettiä heti aluksi, mitkä mallin ominaisuuksista ovat oleellisia, ja mitkä eivät. Koneälymallin sovitus voidaan tehdä lähtemällä liikkeelle yhdestä ominaisuudesta, ja lisätä niitä, kunnes malli on riittävän tarkka. Vaihtoehtoisesti malliin voidaan valita heti aluksi kaikki ominaisuudet, ja vähentää niitä, kunnes mallin tarkkuus on sopivalla tasolla. (Kananen & Puolitaival 2019, 103.)

Koneoppimismallin ominaisuudet voidaan valita toimialan tuntevan ihmisasiantuntijan toimesta, tai käyttäen hyväksi matemaattisia tapoja kuten pääkomponenttianalyysiä (principal component analysis). Lisäksi kannattaa harkita näiden yhdistelmiä. Ominaisuuksien hyvyyksiä kuvaamaan voidaan käyttää erilaisia mittareita kuten gini-index, likehood-ratio tai odds-ratio. (Kananen & Puolitaival 2019, 96.)

Koneoppimismalli voidaan ottaa käyttöön, kun koulutusmallin tarkkuus on saatu riittävän tarkaksi. Tämä edellyttää, että sen toimivuus on varmistettu validointi- ja testausdatan avulla. Koulutusmallia on hyvä hienosäätää sen käyttökokemusten perusteella käyttöönoton jälkeenkin, jotta sen tarkkuus saadaan optimaaliselle tasolle. (Kananen & Puolitaival 2019, 48.)

Tärkein tekoälyn suorituskykyyn vaikuttava tekijä on datan määrä. Mitä suurempi määrä dataa, sitä parempi tarkkuus. Koska datasta löytyy aina kohinaa, eli satunnaissignaaleja, ei lopputulos voi olla koskaan täysin tarkka. On hyvä muistaa, että mikäli esimerkiksi kuvantunnistuksessa ei edes ihminen pysty tulkitsemaan kuvaa, ei sitä myöskään kone pysty tekemään. (Kananen & Puolitaival 2019, 64.)

3.2.1 Ohjattu oppiminen

Ohjatussa oppimisessa koneelle syötetään suuri määrä data-vastauspareja, joiden avulla kone opetetaan toimimaan halutusti. Datan ominaisuuksien, eli attribuuttien perusteella luodaan säännöstö, jonka avulla koneen halutaan päätyvän annettuun vastaukseen. Kun koneelle on koulutettu riittävästi näitä data-vastauspareja, osaa se tehdä ennusteita datassa olevien ominaisuuksien perusteella täysin uudelle datalle. Koska algoritmi opetetaan data-vastausparien avulla, on koulutusdatan laadulla suuri merkitys itse koneen ennusteiden tarkkuuteen. (Kananen & Puolitaival 2019, 48-49; Mueller & Massaron 2016, 168.)

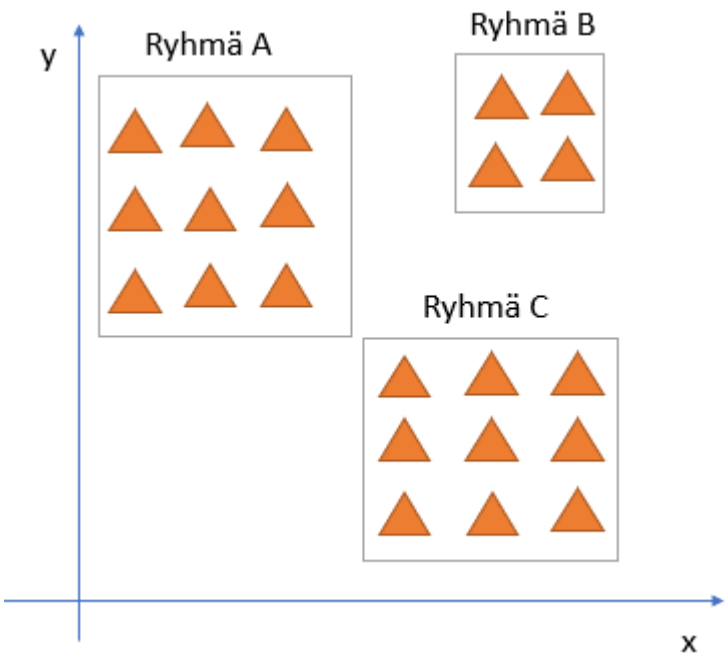
3.2.2 Ohjaamaton oppiminen

Ohjaamattoman oppimisen ideana on se, ettei koneelle syötetä valmiita attribuutteja, joiden avulla vastaus voidaan päätellä. Sen sijaan kone jäsentelee datan itsenäisesti ja luo attribuutit ja vastaukset itse. Tähän päästään vertailemalla dataa ja etsimällä säännönmukaisuuksia sekä eroavaisuuksia siitä. Algoritmia voidaan säätää joko jättämällä pois tai

korostamalla joitain löydettyjä ominaisuuksia – tällaisia säätöparametreja kutsutaan hyperparametreiksi. Koska ohjaamaton oppiminen etsii itse vertailtavat attribuutit, saattaa se löytää datasta myös asioita, joita ihminen ei pysty suoraan havaitsemaan. (Kananen & Puolitaival 2019, 51-53; Mueller & Massaron 2016, 169.)

Ohjaamaton oppiminen on yleinen tapa löytää erilaisia poikkeavuuksia datasta. Tällöin ohjaamatonta oppimista hyödynnetään luokittelemaan datasta asioita, ja voidaan havaita muun muassa väärinkäytöksiä, virheellisiä kirjauksia ja muita normaalista poikkeavia asioita. Yritykset voivat hyödyntää ohjattua oppimista myös esimerkiksi asiakkaiden ryhmittelyyn ja erilaisten ranking-algoritmien luontiin. (Kananen & Puolitaival 2019, 54.)

Yksi ohjaamattoman oppimisen tavoista on klusterointi, jonka perusideana on datan jakaminen klustereihin, eli ryhmiin (kuvio 4). Koska koneelle ei kerrota ennalta vertailtavia attribuutteja, toiminta perustuu siihen, että se etsii datasta ryhmiä, jotka sisältävät hyvin samankaltaisia alkioita. Toisaalta eri ryhmien välinen samankaltaisuus pyritään pitämään pienenä. Klusterointi poikkeaa luokittelusta, ennustamisesta ja kaavojen etsinnästä, mutta toisaalta ryhmiä perustaessaan se luo uutta dataa otsikoimalla ryhmiä. Koska klusterointi luo uusia ryhmiä, joiden otsikot se luokittelee, viitataan käsitteellä myös ohjaamattomaan luokitteluun. Vaikka klusterointi luo uutta dataa nimetessään ryhmiä, ei se missään vaiheessa osaa nimetä ryhmän nimeä siten, että se liittyisi itse ryhmän alkioiden sisältöön. (Kananen & Puolitaival 2019, 54-55.)



KUVIO 4. Esimerkki klusteroinnista

3.3 Koneoppimisen algoritmit

Koneoppiminen pohjautuu viiteen erilaiseen tapaan käsitellä dataa. Toisin sanoen kone pystyy vastaamaan viiteen erilaiseen kysymykseen, jotka ovat:

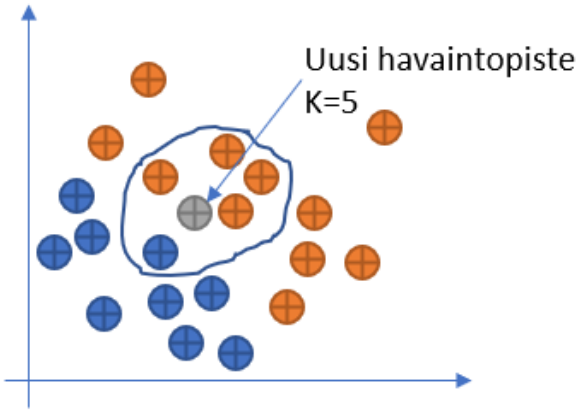
- Mitä tapahtuu arvolle x, jos arvo y muuttuu?
- Kuuluuko uusi havainto luokkaan a vai b?
- Kuuluuko uusi havainto ryhmään a, b vai c?
- Missä järjestyksessä asiat ovat?
- Miten jokin asia voidaan tuottaa tietokoneen avulla?

Näitä kysymyksiä kuvataan myös termeillä regressioalgoritmit, luokittelualgoritmit, ryhmittelyalgoritmit, sijoitusalgoritmit sekä generaatioalgoritmit. Koneoppimisen yhteydessä algoritmilla tarkoitetaan menetelmää, jolla saavutetaan lopputulos tietokoneohjelman avulla. (Kananen & Puolitaival 2019, 112-113.)

3.3.1 K-nearest neighbour

K-nearest neighbour eli K lähintä naapuria algoritmi kuuluu luokittelualgoritmien ryhmään, ja sen avulla voidaan esimerkiksi vastata kysymykseen, kuuluuko havainto ryhmään a, b vai c (Kananen & Puolitaival 2019, 114; Dangeti 2017, 187). Heidän mukaansa algoritmi päättelee uuden havainnon luokan seuraavin askelin (kuvio 5):

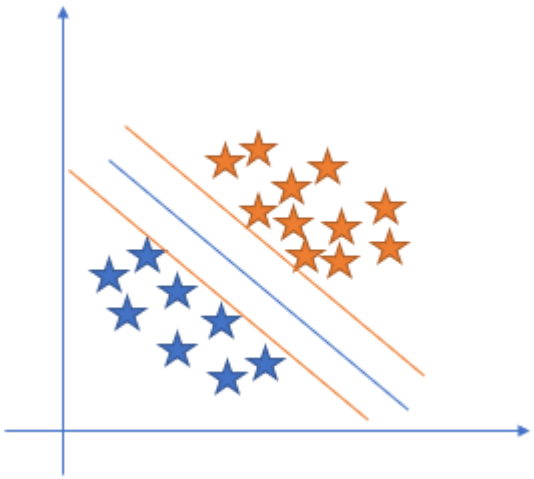
1. Valitaan uutta havaintoa lähinnä olevien datapisteiden lukumäärä, eli K-arvo.
2. Lasketaan uuden pisteen etäisyys lähimpiin naapureihin
3. Valitaan K-arvon mukainen määrä lähimpiä naapureita ja tutkitaan minkä luokan datapisteitä näissä on eniten.
4. Määritetään uuden havaintopisteen ryhmä sen mukaan, minkä ryhmän datapisteitä oli eniten kohdassa 3.



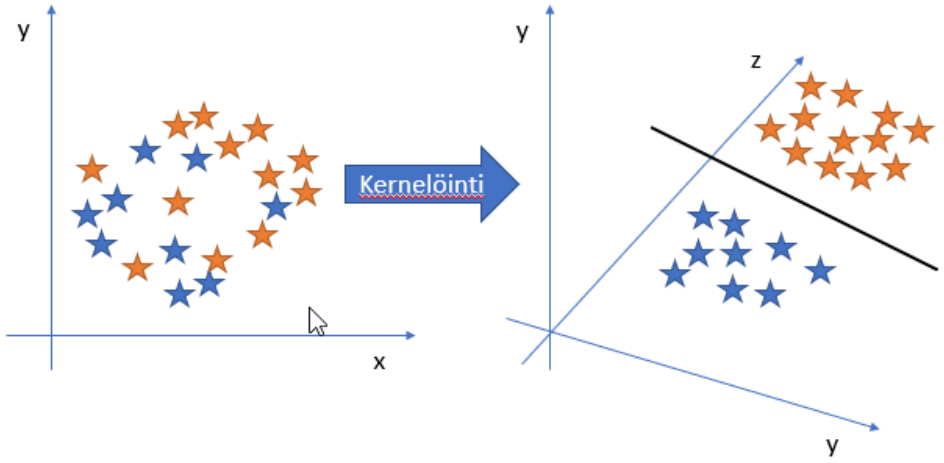
KUVIO 5. K-lähintä naapuria algoritmin toimintaperiaate (Dangeti 2017, 187)

3.3.2 Support Vector Machine – tukivektorikone

Tukivektorikone on menetelmä, jota voidaan käyttää luokittelu- ja regressio-ongelmien ratkaisemiseen. Luokitteluongelmien ratkaisussa algoritmilla pyritään luokittelemaan datapisteet lineaarisen suoran avulla, joka asetellaan siten, että sen etäisyys molempien ryhmien datapisteisiin on mahdollisimman suuri (kuvio 6). Mikäli dataa ei voi erottaa lineaarisella suoralla, voidaan datasetin dimensioiden määrää kasvattaa kernelifunktion avulla. Kuvissa 6 esitetään kernelöinnin suorittaminen hajautuneelle datajoukolle ja nähdään, kuinka uuden dimension avulla lineaarinen suora voidaan sovittaa datapisteiden välille. (Boyle 2011, 2; Kananen & Puolitaival 2019, 122; Dangeti 2017, 220.)



KUVIO 6. Tukivektorikoneen perusidea (Kananen & Puolitaival 2019)



KUVIO 7. Dimension lisääminen ja lineaarisen suoran sovittaminen datapisteiden välille (Dangeti 2017, 221)

3.3.3 Naive Bayes

Naive Bayes-algoritmi on yleisin koneoppimisen luokittelutavoista varsinkin, jos kyseessä on tekstinluokittelutehtävä. Bayesin teoreeman avulla voidaan laskea tapahtuman A todennäköisyys sillä ehdolla, että tapahtuma B havaitaan. Naive Bayes-algoritmi perustuu Bayesin teoreemaan, mutta nimensä mukaisesti olettaa nai'ivisti, että jokainen datasetin ominaisuus on yhtä tärkeä ja riippumaton muista. Tämä ei useinkaan ole todellisessa elämässä totta, mutta Naive Bayes-algoritmi suoriutuu tästä huolimatta luokittelutehtävistä useimmiten riittävän hyvin. Esimerkiksi sähköpostin roskapostisuodatuksessa ei ole merkitystä sillä, toimiiko luokittelu 51 % vai 99 % tarkkuudella, lopputulos voidaan olettaa samaksi molemmilla todennäköisyyksillä. Naive Bayes on nopea, yksinkertainen ja tehokas algoritmi, joka suoriutuu hyvin myös silloin, kun dataa on vähän tai siinä on paljon kohinaa. Algoritmin heikkoutena voidaan pitää sen olettamaa, että kaikki ominaisuudet ovat samanarvoisia. Naive Bayes-algoritmi ei myöskään suoriudu hyvin ominaisuuksista, jotka sisältävät numeraalista dataa. (Lantz 2015, 96-98; Theodoridis 2015, 288.)

Esimerkiksi roskapostin suodatuksessa voidaan saapunut sähköposti luokitella roskapostiksi, mikäli se sisältää, tai ei sisällä ennalta määriteltyjä sanoja. Aluksi Naive Bayesiin perustuva roskapostin suodatin tulee kouluttaa sähköpostiviestien avulla, jotta varmistetaan, että luokittelu toimii riittävällä todennäköisyydellä. Koska Naive Bayes-algoritmi olettaa kaikkien ominaisuuksien olevan toisistaan riippumattomia, voidaan todennäköisyydet laskea toisistaan riippumattomien tapahtumien tulona yhtälön 4 mukaisesti.

$$P(A \cap B) = P(A) * P(B) \quad [4]$$

missä P(A) = A-tapahtuman todennäköisyys

P(B) = B-tapahtuman todennäköisyys

Tästä johdettuna kahden eri sanan perusteella tehty roskapostin ja puhtaan postin luokittelu voidaan esittää yhtälöllä 5.

$$P(spam|W_1 \cap \neg W_2 \cap) = P(W_1|spam)P(\neg W_2|spam)P(spam)$$

$$P(ham|W_1 \cap \neg W_2 \cap) = P(W_1|ham)P(\neg W_2|ham)P(ham) \quad [5]$$

missä $P(W_1|spam) =$ ensimmäisen sanan esiintymisen todennäköisyys

$P(\neg W_2|spam)$ = toisen sanan puuttumisen todennäköisyys

$P(spam)$ = roskapostien suhde kaikkiin viesteihin, jotka sisältävät sanan W_1 mutta eivät sanaa W_2

Toisessa yhtälössä kuvataan sama puhtaille sähköposteille. Roskapostin todennäköisyys saadaan selville yhtälön 6 avulla jakamalla todennäköisyys molempien kaavojen summalla, eli sillä että sähköposti luokitellaan joko roskapostia, tai puhdas sähköpostiksi. (Lantz 2015, 98-99.)

$$\frac{P(W_1|spam)P(\neg W_2|spam)\ P(spam)}{P(W_1|spam)P(\neg W_2|spam)P(spam)+P(W_1|ham)P(\neg W_2|ham)P(ham)} \quad [6]$$

missä

$P(W_1|spam) =$ ensimmäisen sanan esiintymisen todennäköisyys roskapostissa

$P(\neg W_2|spam)$ = toisen sanan puuttumisen todennäköisyys roskapostissa

$P(spam)$ = *roskapostien suhde kaikkiin viesteihin, jotka sisältävät sanan W_1*

$P(W_1|ham) =$ ensimmäisen sanan esiintymisen todennäköisyys puhtaassa postissa

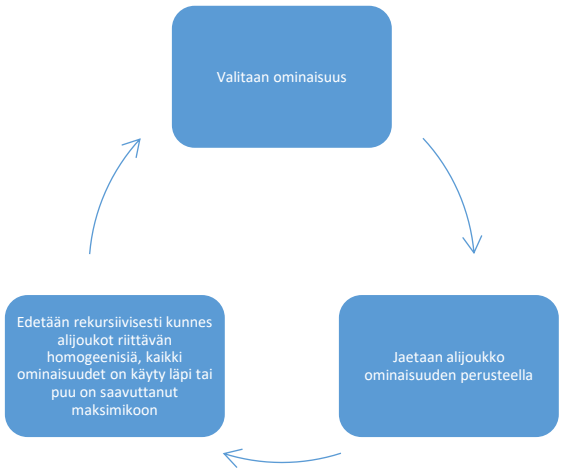
$P(\neg W_2|ham)$ = toisen sanan puuttumisen todennäköisyys puhtaassa postissa

$P(ham)$ = *puhtaiden postien suhde kaikkiin viesteihin, jotka sisältävät sanan W_1*

3.3.4 Decision Tree – päätöspuu

Päätöspuu on puumaista rakennetta hyödyntävä luokittelumenetelmä, jonka avulla voidaan mallintaa ominaisuuksien ja mahdollisten lopputulosten suhteita. Luokittelumenetelmä on saanut nimensä haarautuvista päätöksistä, jotka muistuttavat puun rakennetta laajoista juurista aina vain pienempiin oksiin. Liukukaaviomaisten päätöspuualgoritmien etuna on se, että mallin valmistuttua lopputuloksesta saadaan aikaiseksi myös ihmisystävällinen puurakenne, jonka avulla algoritmin toimintaa voidaan tutkia läpinäkyvästi. Tällä tavoin voidaan helposti tutkia, miksi algoritmi päätyi valitsemaansa lopputulokseen. Päätöspuumenetelmää käytetään laajasti koneoppimisen osana koska sen avulla voidaan mallintaa helpohkosti monen tyyppistä dataa. Päätöspuumenetelmä suoriutuu huonosti dataseteistä, jotka sisältävät suuren määrän numeraalisia tai eri tasoilla sijaitsevia ominaisuuksia - tällöin päätöspuusta muodostuu usein liian kompleksinen. (Lantz 2015, 126-127; Theodoridis 2015 300-301.)

Päätöspuun rakentaminen aloitetaan datan jakamisella alijoukkoihin. Samaa prosessia toistetaan rekursiivisesti uusille alijoukoille valiten aina paras ominaisuusvaihtoehto, kunnes alijoukot ovat riittävän yhdenmukaisia, datasetin kaikki ominaisuudet on käyty läpi tai puu on saavuttanut ennalta määritellyn maksimikoon. Tämä prosessi tunnetaan yleisesti myös nimellä hajota ja hallitse (divide and conquer) (kuvio 8). Periaatteessa jakamista voidaan suorittaa niin kauan, kunnes kaikki ominaisuusyhdistelmät on käyty lävitse. Tämä ei ole yleisesti suositeltua koska tällöin päädytään mallin ylisovittamiseen. (Lantz 2015, 127-130.)



KUVIO 8. Päätöspuun hajota ja hallitse menetelmä

Yksi tunnetuimmista ja parhaiten suoriutuvimmista päätöspuualgoritmeista on nimeltään C5.0. C5.0-algoritmi suoriutuu mallinnuksista lähes yhtä hyvin kuin neuroverkot ja tukivektorikoneet, mutta sen toiminta on huomattavasti helposti ymmärtää. Algoritmin on erittäin yleiskäyttöinen luokittelutehtävissä ja se suoriutuu hyvin myös, mikäli ominaisuudet ovat numeerisia tai nominaalisia. Algoritmi osaa myös käsitellä puuttuvan datan ja ohittaa automaattisesti merkitsemättömät ominaisuudet. Huonoina puolina voidaan pitää herkkyyttä mallin sovittamisen vaikeutta, suurien päätöspuiden vaikeaa tulkittavuutta ja sitä, että päätöspuumallit suosivat usein hajottamisessa sellaisia ominaisuuksia, joilla on useita eri tasoja. (Lantz 2015, 131-132.)

C5.0-päätöspuualgoritmi valitsee hajotettavan ominaisuuden alijoukon puhtauden perusteella käyttäen hyväksi entropia nimistä konseptia. Korkean entropian osajoukoissa tieto on monipuolista, kun taas päätöspuualgoritmien tavoitteena on saada aikaan mahdollisimman homogeeninen joukko, jolloin sen entropia on mahdollisimman pieni. Mikäli osajoukon luokkia on vain kaksi, vaihtelee entropia välillä nollasta yhteen. Mikäli luokkia on enemmän, vaihteluväliksi muodostuu 0 - $\log_2(n)$ jossa n on luokkien määrä. Datajoukon entropia voidaan laskea yhtälön 7 mukaisesti. (Lantz 2015, 133.)

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2(p_i) \qquad [7]$$

missä p_i = osajoukon osuus kaikista datajoukon alkioista

Esimerkiksi jos datajoukko koostuu punaisista, joita on 60 % ja valkoisista, joita on 40 % saadaan datajoukon entropiaksi yhtälön 8 mukaisesti.

$$-0.6 * log2(0{,}6) - 0{,}4 * log2(0{,}4) = 0{,}9709506 \qquad [8]$$

Entropian avulla C5.0-algoritmi laskee ominaisuuden jaon seurauksena syntyvän homogeenisyyden muutoksen. Tätä mittayksikköä kutsutaan informaatiolisäksi (information gain). Informaatiolisä lasketaan jakoa edeltävän joukon ja jaosta muodostuneiden alijoukkojen entropioiden erotuksena yhtälön 9 mukaisesti. Tässä tulee kuitenkin huomata, että alijoukon jakautuessa useampaan luokkaan, tulee Entropia(S_2) laskea yhtälön 10 mukaisesti alijoukkojen summana, jossa jokainen luokka painotetaan siihen kohdistuneen datamäärän tulona. (Lantz 2015, 133-134.)

$$Informaatiolisä(F) = Entropia(S_1) - Entropia(S_2) \quad [9]$$

missä S_1 = jakoa edeltävä joukko

S_2 = jaon jälkeinen joukko

$$Entropia(S_2) = \sum_{i=1}^{n} \omega_i \, Entropia(P_i) \quad [10]$$

missä ω_i = luokkaan sijoitettujen tapausten lukumäärä

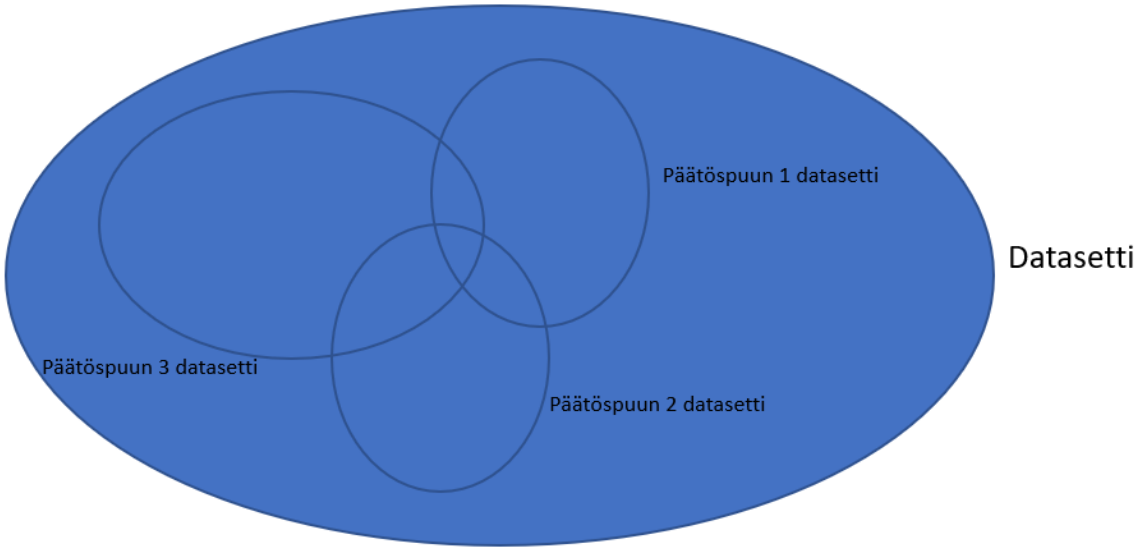
P_i = kyseisen luokan entropia

Jotta päätöspuun mallista ei tule ylisovitettu, tulee sen kokoa pienentää prosessilla, jota kutsutaan karsimiseksi. Karminen voidaan suorittaa joko ennakkoon päättämällä oksien enimmäismäärä tai jälkikäteen antamalla puun kasvaa tarkoituksella liian isoksi. Etukäteen suoritettava karsiminen saattaa aiheuttaa tilanteen, jossa tärkeä kaava jää huomaamatta koska puun ei anneta kasvaa riittävän isoksi. Tämän vuoksi jälkikäteen suoritettu karsiminen on tehokkaampi tapa, jota myös C5.0-algoritmi käyttää pääasiallisesti hyväkseen. Kun päätöspuu on tarkoituksella ylisovitettu, voidaan vähemmän tärkeät oksat poistaa mallista. Osassa tapauksista kokonainen oksa voidaan nostaa mallissa ylemmäs tai korvata se yksinkertaisemmalla päätöksellä, jolloin puhutaan alipuun kasvattamisesta ja korvaamisesta. (Lantz 2015, 135-136.)

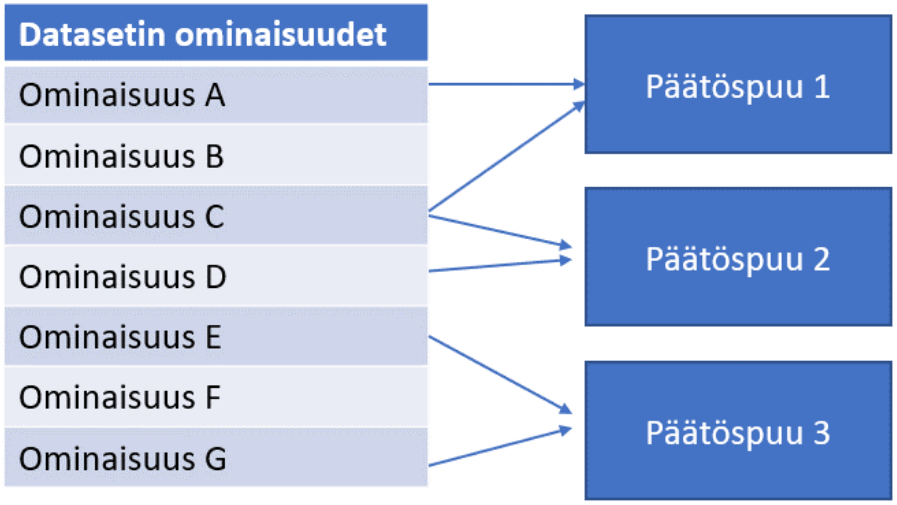
3.3.5 Random Forest – satunnainen metsä

Random forest algoritmi on perinteisen päätöspuun laajennus, joka pienentää virheen mahdollisuutta silloin, kun datasetissä on paljon ominaisuuksia. Algoritmi luo useita päätöspuita, joihin se valitsee datasetistä satunnaisesti eri ominaisuuksia ja alkioita. Kuviossa 9 esitetään datasettien valinta eri päätöspuille. Lopullinen ennuste saadaan yksittäisten päätöspuiden ennusteiden keskiarvona. (Gollapudi 2016, 180-181.)

Kun puuhun valitaan satunnaisesti alkioita, ja tarkasteltavat ominaisuudetkin valitaan satunnaisesti, kutsutaan puita satunnaispuiksi (Gollapudi 2016, 181). Gollapudin kirjoittaa Random forest algoritmin toiminnan perustuvan siihen, että ennusteen tarkkuutta saadaan lisättyä jokaisen satunnaispuun käyttäessä enimmäismäärän dataa, ja toisekseen datasetin virheet jakautuvat useisiin eri paikkoihin. Gollapudin mukaan samasta aineistosta luotujen satunnaispuiden määrä voi olla satoja. Kuviosta 10 löytyy esimerkki, jossa seitsemästä eri ominaisuudesta valitaan satunnaisesti ominaisuuksia kolmen eri päätöspuun luontiin.



KUVIO 9. Random forest päätöspuiden datasettien valinta (Gollapudi 2016, 180-181)

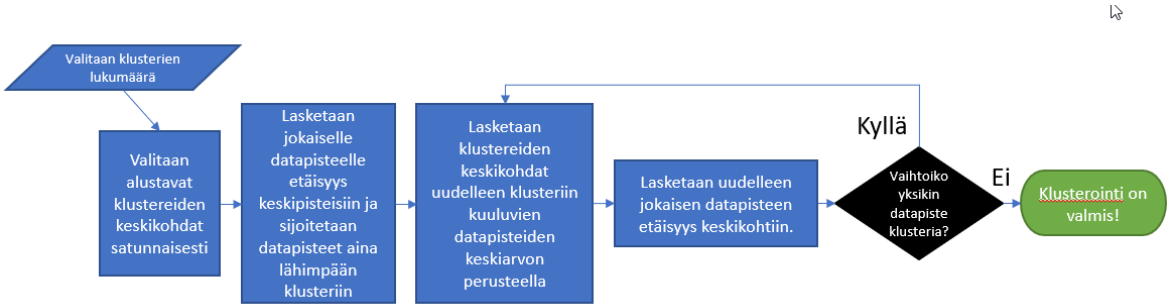


KUVIO 10. Random forest päätöspuiden ominaisuuksien satunnaisvalinta (Gollapudi 2016, 180-181)

3.3.6 K-Means-algoritmi

K-means algoritmi on yksi suosituimmista klusterointitekniikoista. K-means algoritmin ohelle on syntynyt useita kehittyneempiä klusterointitekniikoita, mutta se on silti erittäin käytetty tekniikka, koska sen yksinkertaiset toimintatavat voidaan kuvata ilman tilastollisia

käsitteitä. K-means on erittäin adaptoituva tekniikka ja se suoriutuu riittävän hyvin myös oikean elämän käyttötapauksista. K-means algoritmin heikkoutena voidaan pitää tarvetta arvata etukäteen muodostuvien klusterien lukumäärä, jonka perusteella algoritmi alkaa etsitään paikallisesti optimoituja ratkaisuja. Lisäksi klustereiden aloituspisteiden satunnaisasettelu saattaa aiheuttaa tilanteita, joissa K-means ei pysty löytämään optimaalisimpia ryhmiä. (Lantz 2015, 289.)



KUVIO 11. K-means algoritmin toimintaperiaate (Lantz 2015, 289)

K-means-algoritmi aloittaa klusteroinnin (kuvio 11) luomalla annetun lukumäärän mukaisen määrän klustereita valitsemalla datasetistä satunnaisesti annetun k:n mukaisen määrän datapisteitä. Näistä datapisteistä muodostuvat klusterien keskikohdat. Seuraavaksi algoritmi laskee jokaiselle datapisteelle etäisyyden kaikkiin keskipisteisiin Euklidisen etäisyyden yhtälön (11) avulla ja sijoittaa datapisteen lähimpänä olevaan klusteriin. Kun kaikki datapisteet on sijoitettu klustereihin, lasketaan klusterin keskipiste sen datapisteiden keskiarvon perusteella, ja datapisteiden etäisyydet näihin uusiin keskipisteisiin. Mikäli yhdenkin datapisteen klusteri vaihtuu tämän laskennan seurauksena, toistaa algoritmi tätä prosessia niin kauan, kunnes kaikki datapisteet pysyvät samoissa klustereissa. (Lantz 2015, 289-292.)

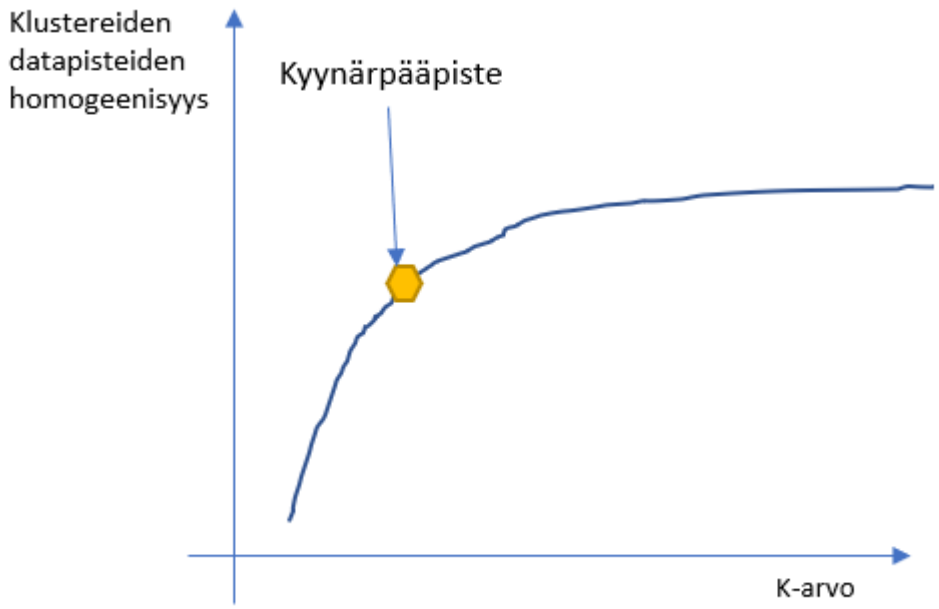
$$etäisyys(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad [11]$$

missä x_i = datapisteen X-koordinaatti

y_i = datapisteen Y-koordinaatti

K-means algoritmin lopputulos riippuu suuresti klusterien lukumäärän valinnasta, eli K-arvosta. K-arvon asettaminen liian suureksi ylisovittaa mallin herkästi, joten sen

määrittämiseen kannattaa käyttää aikaa. Ideaalitapauksessa lukumäärä osataan valita tutkimuskysymyksen perusteella, mutta se voidaan myös laskea kyynärpäämetodin (elbow method) avulla (kuvio 12). Kyynärpäämetodissa tutkitaan klustereiden datapisteiden homogeenisyyttä suhteessa K-arvoon ja valitaan sellainen K-arvo, jonka jälkeen homogeenisyys ei kasva merkittävästi. Metodin ongelmana on se, että käytännössä eri K-arvoja testattaessa dataa iteroidaan useita kertoja ja tämä saattaa viedä aikaa merkittävästi. (Lantz 2015, 294-295.)

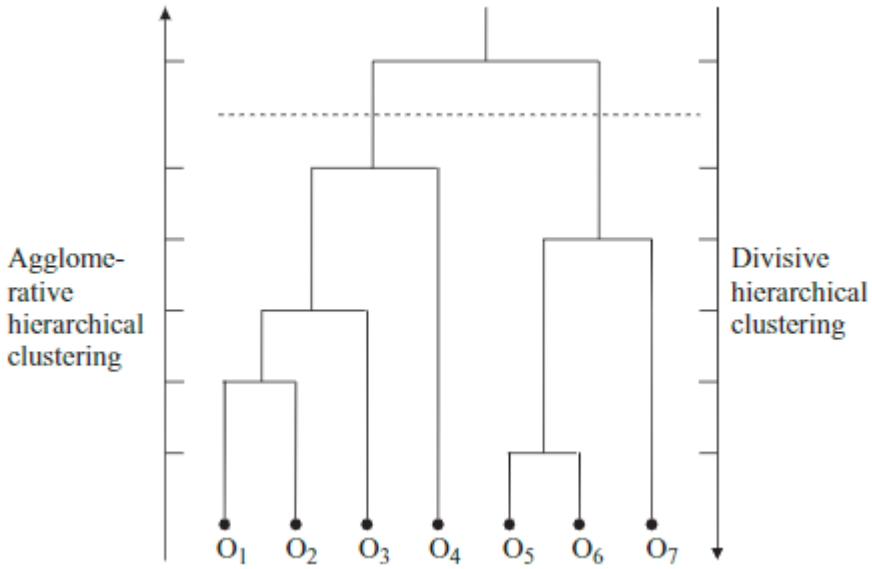


KUVIO 12. K-arvon hakeminen kyynärpääpisteen avulla (Lantz 2015, 294-295)

3.3.7 Hierarkkinen klusterointi (Hierarchical clustering)

Hierarkkisessa klusteroinnissa data ryhmitellään hierarkkisiin sisäkkäisiin joukkoihin joko niin, että ensimmäinen muodostettu klusteri sisältää kaikki alkiot, tai niin että ensimmäinen klusterointi muodostaa jokaisesta alkiosta oman klusterinsa. Näistä kahdesta eri klusterointitavasta käytetään nimityksiä divisiivinen ja agglomeratiivinen hierarkkinen klusterointi. Molemmat klusterointitavat käyttävät klustereiden muodostamisessa hyväkseen linkkifunktioita, ja niiden pohjalta syntynyt lopputulos esitetään usein binääripuuna tai dendrogrammina. Kuvassa 5 esitetään hierarkkisesti klusteroitu data, jonka lopputuloksena esitetään dendrogrammi katkaistuna sopivalle tasolle. Divisiivinen jakotapa on laskentatehollisesti vaativampi kuin agglomeratiivinen, jonka vuoksi sitä käytetään

harvemmin kuin agglomeratiivista tapaa. Yleisesti hierarkkisen klusteroinnin heikkoutena voidaan pitää sen suurta laskentatehovaatimusta. (Xu & Wunch 2009, 31-32.)

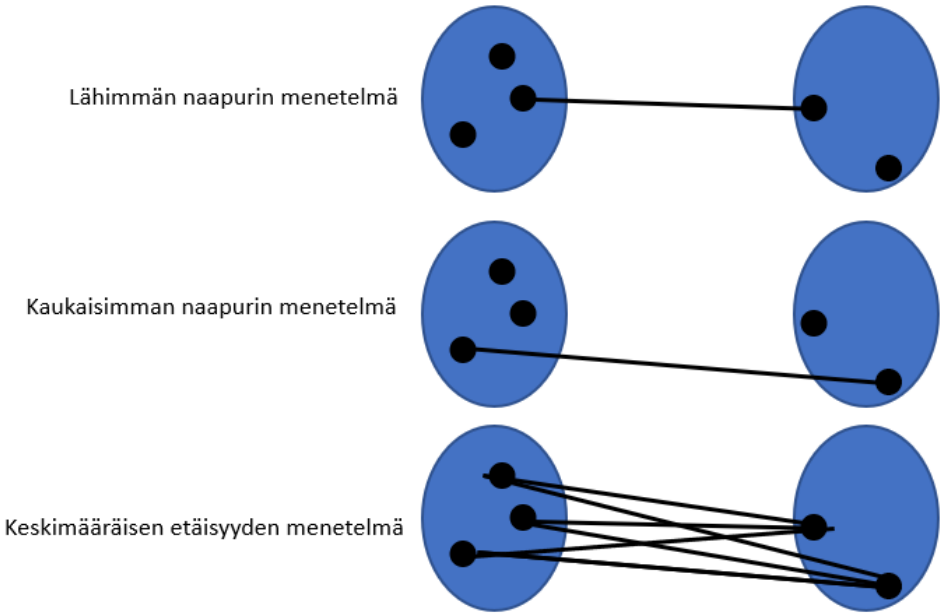


KUVA 5. Agglomeratiivisen ja divisiivisen hierarkkisen klusteroinnin ero (Xu & Wunch 2009, 31-32)

Agglomeratiivinen hierarkkinen klusterointi aloitetaan luomalla jokaisesta alkiosta oman klusterinsa. Tämän jälkeen yhdistetään klustereita toisiinsa valitun linkkifunktion avulla niin kauan, kunnes jäljellä on ainoastaan yksi klusteri. Mahdollisia linkkifunktioita on useita erilaisia, ja ne soveltuvat erityyppisille aineistoille. Lopputulokseen päästään, kun dendrogrammista leikataan valitun tason alapuolella olevat merkityksettömät klusterit pois. (Xu & Wunch 2009, 33.)

Linkkifunktiot jaetaan graafimetodeihin ja geometrisiin metodeihin. Graafimetodit sisältävät lähimmän naapurin (single linkage), kaukaisimman naapurin (complete linkage) ja keskimääräisen etäisyyden (average linkage) menetelmät (kuvio 13). Lähimmän naapurin menetelmässä yhdistetään klusterit toisiinsa kahden eri klusterin lähimpänä olevien alkioiden avulla. Menetelmä toimii hyvin, mikäli yhdistettävät klusterit sijaitsevat kaukana toisistaan. Mikäli aineisto sisältää paljon kohinaa, saattaa lähimmän naapurin menetelmä yhdistää toisiinsa kaksi klusteria, joiden ominaisuudet ovat kaukana toisistaan. Kohinan aiheuttama yhdistäminen muokkaa klustereista usein pitkänomaisia. Kaukaisimman naapurin menetelmä sen sijaan yhdistää kaksi klusteria toisiinsa kahden eri klusterin kauimpina olevien alkioiden avulla. Tämän vuoksi kaukaisimman naapurin menetelmä on tehokas löytämään

pieniä ja tiiviitä klustereita. Keskimääräisen etäisyyden menetelmässä kahden klusterin välinen etäisyys määritellään niiden kaikkien alkioiden etäisyyksien keskiarvona. (Xu & Wunch 2009, 34)



KUVIO 13. Graafimetodit klusterin etäisyyksien laskentaan (Xu & Wunch 2009, 34.)

Geometrisiä linkkifunktioita ovat painotettu keskimääräisen etäisyyden menetelmä (weighted average linkage), keskiarvostetun etäisyyden menetelmä (centroid linkage), mediaanisen etäisyyden menetelmä (median linkage) ja Wardin menetelmä. Merkittävien ero graafisten ja geometristen funktioiden välillä on se, että graafisissa metodeissa laskenta ottaa huomioon kaikki alkiot, kun taas geometriset metodit laskevat klustereille keskikohdat, joiden avulla laskenta suoritetaan. (Xu & Wunch 2009, 35-37.)

Painotetun keskimääräisen etäisyyden menetelmä toimii lähes samalla tavalla kuin keskimääräisen etäisyyden menetelmä, mutta se ottaa laskennassa huomioon myös klusterin alkioiden lukumäärän painottavana tekijänä. Keskiarvostetun etäisyyden menetelmässä lasketaan jokaiselle klusterille keskiarvopiste, jonka avulla saadaan laskettua etäisyys muihin klustereihin. Keskiarvostetun etäisyyden menetelmä vastaa Euklidisen etäisyyden laskentakaavaa. Mediaanisen etäisyyden metodia käytetään, kun kahden yhdistettävän klusterin alkioiden määrä on sama. Laskenta tapahtuu lähes vastaavasti kuin keskiarvostetun etäisyyden menetelmässä, mutta molemmille klustereille annetaan laskennassa sama painoarvo. Wardin menetelmä tunnetaan myös pienimmän varianssin

menetelmänä. Menetelmän ideana on yhdistää klusterit siten, että yhdistämisen seurauksena klusterien sisäinen varianssi kasvaa mahdollisimman vähän. Menetelmässä lasketaan kahden yhdistetyn klusterin varianssi, ja valitaan lopulliseksi muodostuneeksi klusteriksi se klusteri, jonka varianssi on pienin. (Xu & Wunch 2009, 35-37.)

Divisiivisessa hierarkkisessa klusteroinnissa hierarkian muodostaminen aloitetaan klusterista, joka sisältää kaikki datasetin alkiot. Koska alussa algoritmin täytyy muodostaa kaksi klusteria, on erilaisia jakamisvaihtoehtoja 2^{N-1}-1 kappaletta N:n ollessa alkioiden lukumäärä. Divisiivinen hierarkkinen klusterointi käyttää erittäin paljon laskentakapasiteettia, jonka vuoksi sitä käytetään harvemmin kuin agglomeratiivista tapaa. (Xu & Wunch 2009, 37-38.)

Eräs divisiivisen hierarkkisen klusteroinnin algoritmeista on nimeltään DIANA (Divisive ANAlysis). Aluksi DIANA-algoritmi valitsee jaettavaksi aina läpimitaltaan suurimman klusterin. Klusterointi aloitetaan kopioimalla jaettava klusteri C_i uudeksi klusteriksi C_l ja luomalla uusi tyhjä klusteri C_j sen rinnalle. Tämän jälkeen iteroidaan kaikki jaettavan klusterin alkiot lävitse ja valitaan siirrettäväksi tyhjään klusteriin C_j alkioista se, jonka etäisyys muihin alkioihin on suurin. Tämän jälkeen iteroidaan jälleen kaikki jaettavan klusterin alkiot läpi, ja lasketaan alkion keskimääräisten etäisyyksien erotus alkuperäiseen klusteriin C_i ja uuteen klusteriin C_j. Siirrettäväksi uuteen klusteriin C_j valitaan se alkio, jonka etäisyys on suurin. Mikäli kaikkien iteroitavien alkioiden etäisyys on nolla tai pienempi, siirrytään jakamaan seuraavaa klusteria niin kauan kunnes kaikki alkiot ovat omia klustereitaan. Tämän jälkeen katkaistaan dendrogrammi halutulta tasolta ja muodostetaan lopulliset klusterit. (Xu & Wunch 2009, 37-39.)

4 CASE: POIKKEUKSIEN HAVAITSEMINEN KIRJAUTUMISDATASTA

4.1 Graylog-lokitietojärjestelmän asennus

Graylog-järjestelmä koostuu MongoDB tietokannasta, Elasticsearch-hakukoneesta sekä itse Graylog-ohjelmistosta, jonka avulla sekä tietojen tallennus että analysointi tehdään. Graylog-järjestelmää varten asennettiin käyttöjärjestelmä Ubuntu Server 18.04 fyysiselle HP Proliant-palvelimelle, jolla oli riittävästi vapaata tallennustilaa lokitietoja varten. Itse järjestelmän asennus oli hyvin suoraviivainen, eikä vaatinut asennusohjeista poikkeamista.

4.2 Lokitiedon kerääminen Office365-palvelusta Graylog-järjestelmään

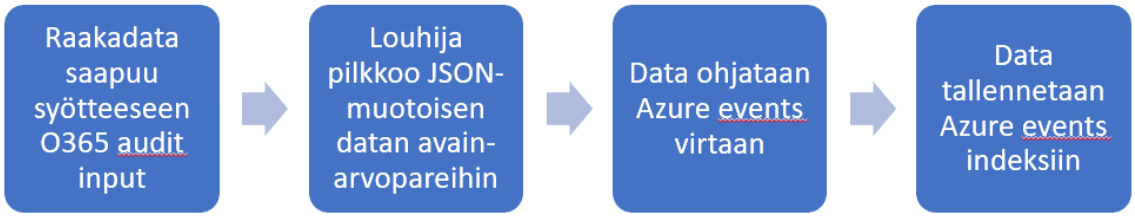
Graylog-järjestelmään voidaan syöttää lokidataa lähes minkälaisesta järjestelmästä tahansa. Yleisesti käytössä oleviin järjestelmiin löytyy valmiita lisäosia, jotka helpottavat lokitiedon jäsentelyä, mutta järjestelmään voidaan myös luoda louhijoita (extractor), joiden avulla voidaan purkaa lokitieto omiin kenttiinsä haun helpottamiseksi. Tässä työssä tutkittiin poikkeuksia käyttäen datana käyttäjien Office365 kirjautumistietoja. Kuvakaappauksista päädyttiin sumentamaan henkilötietoja sisältävät kohdat.

Microsoft tarjoaa kolmannen osapuolen sovelluksia varten REST-tyyppisen Office 365 Management Activity-rajapinnan, jonka avulla lokitiedot saadaan noudettua Graylog-järjestelmään. Rajapinta sisältää Azure Active Directory, Exchange, Sharepoint, General ja DPL tyyppisiä tapahtuma- ja toimintolokeja, ja näistä jokainen tulee tilata Microsoftilta rajapinnan avulla. Näistä General sisältää tapahtumat kaikista muista palveluista kuten Teams, Power BI ja Forms. Rajapinnan kautta saadaan noudettua samat tiedot kuin Office 365 Security & Compliance hallintapaneelin Audit log search -toiminnosta sekä Azure Active Directory:n Audit logs -toiminnosta. Tässä opinnäytetyössä keskityttiin Azure Active Directoryn kirjautumistietoihin, jotka kertovat varsin monipuolisesti työasemille sekä sähköpostiin kirjautumisesta.

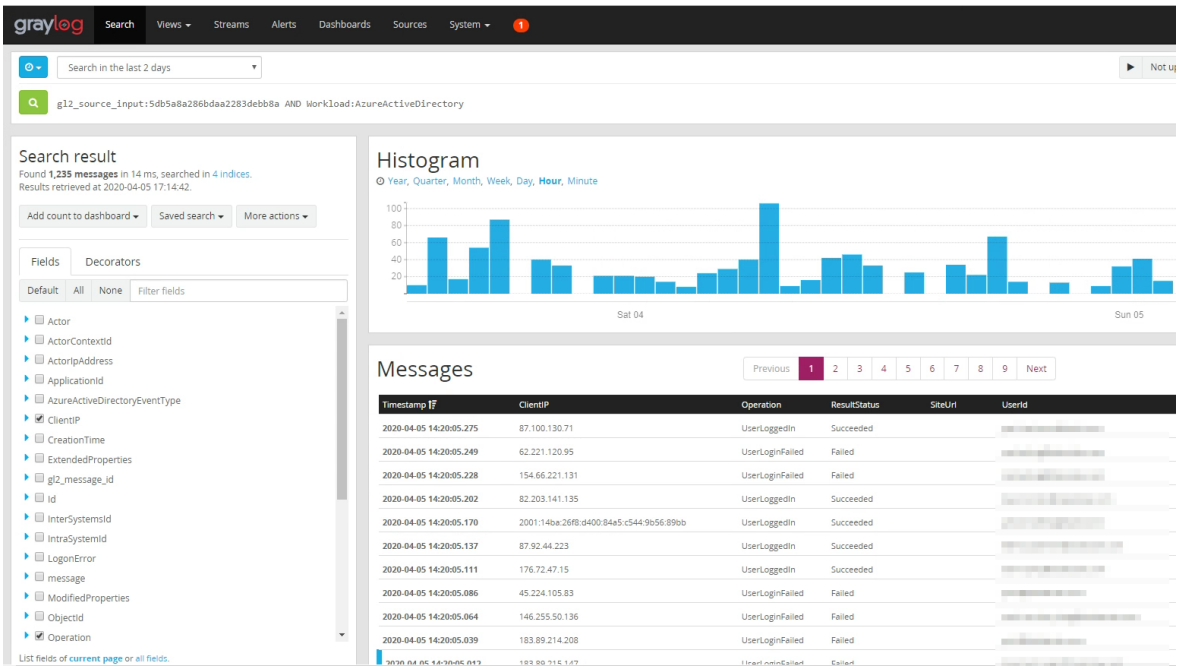
Tässä työssä käytettiin hyväksi Office365 API audit log collector nimistä Python-skriptiä. Muita avoimen lähdekoodin vaihtoehtoja olisi ollut esimerkiksi ohjelma nimeltä O365beat, mutta sen ollessa vielä hyvin kehitysasteella päätettiin edetä ensimmäisenä mainitun kanssa. Skriptin koodi käytiin aluksi läpi, jotta voitiin varmistua siitä, että se todella tekee vain ne asiat, joita sen oletettiin tekevän. Kun skripti oli todettu turvalliseksi, tehtiin tarvittavat muutokset Office365:een ja tilattiin auditointilokit noudettavaksi. Lokitapahtumien noutoskripti ajastettiin suoritettavaksi Graylog-palvelimella 30 minuutin välein, jolloin se noutaa aina kaikki noutamattomat tapahtumaryppäät edellisen vuorokauden ajalta. Noudon

jälkeen tiedot lähetettiin Graylog-järjestelmän käsiteltäväksi, jolloin ne saatiin tallennettua analysointia varten.

Graylog-järjestelmässä luotiin tietojen vastaanottoa varten uusi syöte (input), joka asetettiin kuuntelemaan TCP-porttia 6000. Kun uusi lokidata saapuu järjestelmään, ohjataan se louhijalle, joka käsittelee JSON-tyyppisen datan ja purkaa sen avain-arvoparelksi. Data ohjataan tämän jälkeen tietovirtaan (stream) syötteessä staattiseksi määritellyn lähteen perusteella. Datan kulku Graylog-järjestelmässä esitetään kuviossa 14. Kuvassa 6 esitetään Graylog-järjestelmän hakusivun yleisnäkymä.



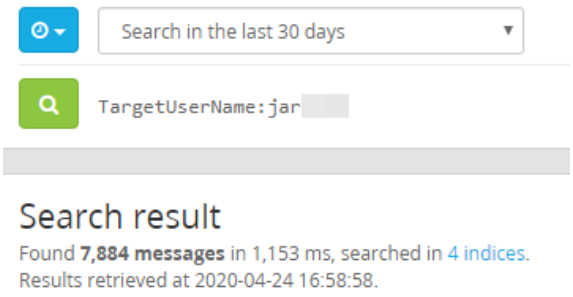
KUVIO 14. Datan kulku Graylog-järjestelmässä



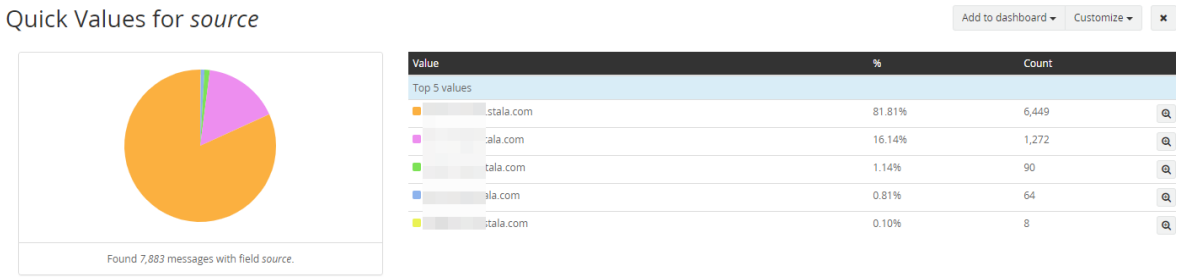
KUVA 6. Azure Active Directoryn kirjautumistapahtumia Graylog-järjestelmässä

4.3 Lokitietojen analysointi Graylog-järjestelmässä

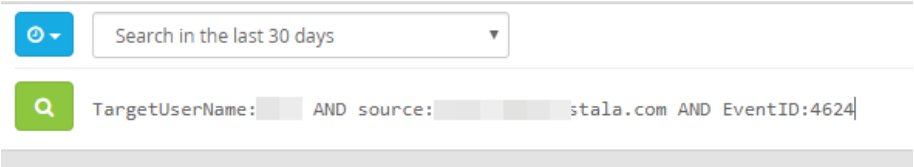
Lokitietoja voidaan analysoida Graylog-järjestelmässä käyttäen hakutyökalua. Järjestelmän hakutyökalu on parhaimmillaan ad hoc-kyselyissä, joissa etsitään esimerkiksi tietyn käyttäjän kirjautumistietoja, mutta valmista kyselyä ei vielä ole olemassa, tai ei tiedetä mitä lokimerkintää pitäisi etsiä. Kysely voidaankin rakentaa rajaustekijä kerrallaan siten, että aloitetaan esimerkiksi käyttäjätunnuksesta. Testattaessa järjestelmää kyselyn rakentaminen aloitettiin valitsemalla käyttäjätunnus ja aikarajaukseksi viimeiset 30 päivää (kuva 7). Seuraavaksi haluttiin tietää mitä järjestelmiä kyseinen käyttäjätunnus on käyttänyt. Tämä onnistui helposti valitsemalla kenttälistalta source-kentän Quick values-toiminto (kuva 8), joka esitti viestin lukumäärät lähdejärjestelmittäin kuvan 8 mukaisesti. Jokaisen rivin perässä olevasta suurennuslasin kuvakkeesta pystyi lisäämään kyseisen arvon kyselyn rajaukseksi, jolloin hakukysely muodostui automaattisesti vain valintoja tehden (kuva 9). Kyselyn olisi voinut syöttää myös itse, mutta tällöin tulee tuntea Elasticsearch -hakukoneen syntaksi.



KUVA 7. Graylog kyselyn rakentaminen



KUVA 8. Quick values -toiminnon käyttäminen



KUVA 9. Hakukyselyn muodostuminen valintojen perusteella

Graylog-järjestelmässä ei itsessään pystynyt suorittamaan kovinkaan syvällistä tietojen analysointia, vaan se toiminnot rajoittuivat hakujen suorittamiseen ja hakutulosten summailuun. Järjestelmään pystyi kuitenkin lisäämään erilaisia hälytyksiä, joiden avulla pystyi saamaan tiedon esimerkiksi siitä, mikäli jollakin käyttäjätunnuksella yritettiin kirjautua sisään virheellisesti useita kertoja peräkkäin.

4.4 Lokitietojen haku Graylog-järjestelmästä RStudioon

R-ohjelmointikielelle on tehty Elastic-niminen kirjasto, jonka avulla R:ssä voidaan kysellä dataa suoraan Graylogin-järjestelmän käyttämältä Elasticsearch-hakukoneelta R-skriptin yhteydessä. Lyhimmillään koodi koostuu yhdistämisfunktiosta connect ja hakufunktiosta Search, jotka on esitetty alla olevassa kuvassa 10. Esimerkissä allAzureEvents-muuttujaan noudetaan kymmenen ensimmäistä hakutulosta. Mikäli hakutuloksia halutaan enemmän, voidaan Search-funktiolle antaa parametrina arvo size, jossa määritellään haluttu hakutulosmäärä. Elasticsearch hakukoneessa on kuitenkin resurssien suojelemiseksi rajattu maksimi hakutulosten määrä oletuksena 10000 kappaleeseen, eli parametri ei itsessään mahdollista noutamaan suurta määrää dataa.

```
# Load Elastic library
library(elastic)

# Connect to Graylog elastic engine
connection<-connect(host = "localhost", port = 9200)

# Search from index azure_0 where workload = AzureActiveDirectory
allAzureEvents <-  Search(conn = connection, index = "azure_0", q = "Workload:AzureActiveDirectory")
```

KUVA 10. Esimerkki Elastic-hakukoneen käytöstä R:ssä.

Mikäli Elasticsearch-hakukoneesta halutaan enemmän dataa RStudioon, tulee käyttää hyväksi scroll-nimistä funktiota. Scroll-funktion avulla voidaan Elasticsearchista hakea useita hakutulossivuja yhden maksimissaan 10000 tuloksen sivun sijaan. Scroll-funktion käyttämistä varten tulee Search-funktiossa kertoa parametrilla time_scroll se aika, jonka sisällä

seuraava hakutulossivu noudetaan. Mikäli aika umpeutuu, sulkeutuu kyseisen haun ikkuna ja resurssit vapautetaan muuhun käyttöön. Kuvassa 11 kuvataan kaikkien hakutulosten noutaminen rows-nimiseen muuttujaan. Elasticsearch suosittelee järjestykseksi suurta datamäärää noudettaessa ”_doc”-asetusta, jolloin hakutulokset noudetaan hakuindeksin mukaisessa järjestyksessä.

```
# Search from index azure_0 where workload = AzureActiveDirectory.
result <-  Search(conn = connection, index = "azure_0",body = '{
  "query": {
    "bool": {
      "must": [
        {
          "query_string": {
            "query": "Workload:AzureActiveDirectory"
          }
        }
      ]
    }
  },
  "sort": ["_doc"]
}', size = 10000, time_scroll="1m")

# save first hits to rows array
rows = result$hits$hits

# loop scroll and save hits rows array until there's no hits left
hits <- 1
while(hits != 0){
  result <- scroll(connection, result$`_scroll_id`, time_scroll="1m")
  hits <- length(result$hits$hits)
  if(hits > 0)
    rows <- c(rows, result$hits$hits)
}
```

KUVA 11. Kaikkien hakutulosten noutaminen RStudiossa rows-muuttujaan

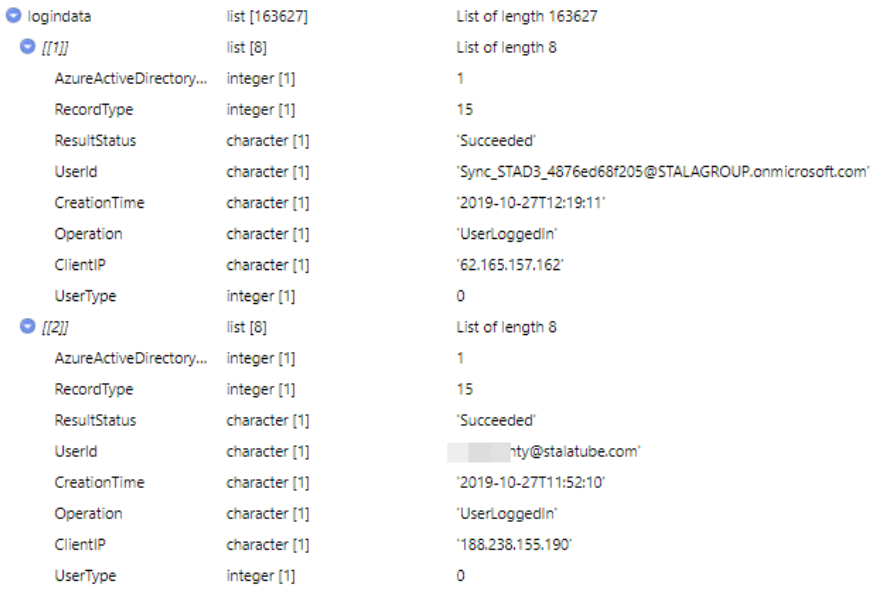
Alla olevasta kuvasta 12 nähdään, että kyseisen haun tuloksia muodostui 163627 kappaletta, ja ne löytyvät rows-muuttujasta. Ensimmäistä tietuetta tarkastelemalla selviää, että varsinainen kirjautumisdata löytyy source-nimisen listan alta, ja se sisältää erittäin paljon erilaisia ID-tunnuksia. Hakua muokattiin lisäämällä lista noudettavista sarakkeista, jolloin ID-tunnukset jätettiin pois. Lisäksi logindata-muuttujaan haettiin suoraan source-listan sisältö (kuva 13). Näillä toimenpiteillä datan määrä käsiteltävä datamäärä putosi 1,5Gb:sta noin 260Mb:iin. Tämän toimenpiteen jälkeen logindata-muuttujan datasisältö oli yksinkertaisessa muodossa valmiina datan analyysiä varten (kuva 14).

rows	list [163627]	List of length 163627
[[1]]	list [6]	List of length 6
_index	character [1]	'azure_0'
_type	character [1]	'message'
_id	character [1]	'1cb138d1-f8e0-11e9-a2e5-68b599726dbe'
_score	NULL	Pairlist of length 0
_source	list [31]	List of length 31
AzureActiveDirectoryEventType	integer [1]	1
gl2_remote_ip	character [1]	'127.0.0.1'
gl2_remote_port	integer [1]	57084
UserKey	character [1]	'10037FFE9445FD40@STALAGROUP.onmicrosoft.com'
ActorIpAddress	character [1]	'62.165.157.162'
source	character [1]	'Azure'
Operation	character [1]	'UserLoggedIn'
OrganizationId	character [1]	'97cad2c9-63d2-4afa-ac51-35cf86089869'
gl2_source_input	character [1]	'5db5a8a286bdaa2283debb8a'
ExtendedProperties	character [1]	'{Name=UserAuthenticationMethod, Value=1}, {Name=RequestType, Value=WindowsAuthe ...
IntraSystemId	character [1]	'2c031cd5-a847-43d8-854f-951518411500'
Target	character [1]	'{ID=00000002-0000-0000-c000-000000000000, Type=0}'
RecordType	integer [1]	15
Version	integer [1]	1
Actor	character [1]	'{ID=3f60403d-d46d-4a16-a488-9c483644b886, Type=0}, {ID=Sync_STAD3_4876ed68f205@ ...
gl2_source_node	character [1]	'7154174c-e2ad-4f6f-b537-74779d689fe4'
ActorContextId	character [1]	'97cad2c9-63d2-4afa-ac51-35cf86089869'
timestamp	character [1]	'2019-10-27 17:35:04.925'
ResultStatus	character [1]	'Succeeded'
ObjectId	character [1]	'00000002-0000-0000-c000-000000000000'
streams	list [1]	List of length 1
gl2_message_id	character [1]	'01DR74J96YEA2AJQFKKZVMBHRS'
message	character [1]	'{"CreationTime": "2019-10-27T12:19:11", "Id": "5900b05f-a280-4b8c-8fca-34245ed5 ...
ClientIP	character [1]	'62.165.157.162'
Workload	character [1]	'AzureActiveDirectory'
UserId	character [1]	'Sync_STAD3_4876ed68f205@STALAGROUP.onmicrosoft.com'
TargetContextId	character [1]	'97cad2c9-63d2-4afa-ac51-35cf86089869'
CreationTime	character [1]	'2019-10-27T12:19:11'
Id	character [1]	'5900b05f-a280-4b8c-8fca-34245ed580d9'
InterSystemsId	character [1]	'224fd491-1de7-4ee9-8351-a44615a3c111'
UserType	integer [1]	0
sort	list [1]	List of length 1

KUVA 12. Alkuperäinen data yksittäisestä lokitietomerkinnästä

```
# Search from index azure_0 where workload = AzureActiveDirectory. Limit timerange from the start of the year
result <-  Search(conn = connection, index = "azure_0",body = '{
  "query": {
    "bool": {
      "must": [
        {
          "query_string": {
            "query": "Workload:AzureActiveDirectory"
          }
        }
      ]
    }
  },
  "sort": ["_doc"],
  "_source": ["AzureActiveDirectoryEventType", "CreationTime","ClientIP","Operation","UserId","RecordType","UserType","ResultStatus"]
}', size = 10000, time_scroll="1m")


# loop scroll and save result hits rows array until there's no hits left
hits <- 1
i <- 1
logindata<-c()

while(hits != 0){
  hits <- length(result$hits$hits)

  # if there's data, loop all hits and add values from _source list to logindata variable
  if(hits > 0){
    for(x in result$hits$hits){
      logindata[[i]] <- x$`_source`
      i <- i + 1
    }
  }

  # proceed to next search page
  result <- scroll(connection, result$`_scroll_id`, time_scroll="1m")
}
```

KUVA 13. Tarkemmin rajattu kysely



logindata	list [163627]	List of length 163627
[[1]]	list [8]	List of length 8
AzureActiveDirectory...	integer [1]	1
RecordType	integer [1]	15
ResultStatus	character [1]	'Succeeded'
UserId	character [1]	'Sync_STAD3_4876ed68f205@STALAGROUP.onmicrosoft.com'
CreationTime	character [1]	'2019-10-27T12:19:11'
Operation	character [1]	'UserLoggedIn'
ClientIP	character [1]	'62.165.157.162'
UserType	integer [1]	0
[[2]]	list [8]	List of length 8
AzureActiveDirectory...	integer [1]	1
RecordType	integer [1]	15
ResultStatus	character [1]	'Succeeded'
UserId	character [1]	[redacted]nty@stalatube.com'
CreationTime	character [1]	'2019-10-27T11:52:10'
Operation	character [1]	'UserLoggedIn'
ClientIP	character [1]	'188.238.155.190'
UserType	integer [1]	0

KUVA 14. Sisäänkirjautumistiedot jäsenneltynä yksinkertaisempaan listamuotoon

Tässä vaiheessa kuitenkin huomattiin, että listamuotoisen datan käsittely R:llä ei ole järkevää, vaan data pitäisi saada taulukkomuotoiseen datatable-muotoon, jonka avulla haluttuisin sarakkeisiin voitaisiin viitata suoraan sen sijaan, että koko lista käsiteltäisiin esimerkiksi silmukassa. Lista yritettiin muuntaa datatableksi setDT-funkfion avulla, mutta se ei onnistunut, koska osassa listan elementeistä ei ollut mukana ClientIP-tietoa. Lopulta huomattiin, että R:n data.table-kirjastosta löytyvien rbindlist-, lapply- ja rbind-funktioiden avulla listamuotoinen data pystyttiin kääntämään suoraan taulukkomuotoiseksi ilman ylimääräisiä silmukoita. Kuvassa 15 kuvataan komentojen käyttö tähän tarkoitukseen. Lopuksi kirjautumisdata (kuva 16) tallennettiin fwrite-funktion avulla CSV-tiedostoon, jotta saisimme pidettyä datan muuttumattomana analyysia varten.

```
## Get only the data we're interested in. Get all the results and finally save the data to office365-logindata.csv file for analyze

# Search from index azure_0 where workload = AzureActiveDirectory.
result <-  Search(conn = connection, index = "azure_0",body = '{
  "query": {
    "bool": {
      "must": [
        {
          "query_string": {
            "query": "Workload:AzureActiveDirectory"
          }
        }
      ]
    }
  },
  "sort": ["_doc"],
  "_source": ["AzureActiveDirectoryEventType", "CreationTime","ClientIP","Operation","UserId","RecordType","UserType","ResultStatus"]
}', size = 10000, time_scroll="1m")


# loop scroll and save result hits rows array until there's no hits left
hits <- 1
logindata<-data.table()

while(hits != 0){
  hits <- length(result$hits$hits)

  # if there's data available, process it
  if(hits > 0){
    # "Pivot" the data using rbindlist,
    # select only "_source" list using lapply
    # because datatable needs to have same same number of columns, fill with NA if some data is unavailable
    dt <- rbindlist(lapply(result$hits$hits, "[[", "_source"), fill = TRUE, use.names = TRUE)

    # append data to logindata
    logindata<-rbind(logindata, dt)
  }

  # proceed to next search page
  result <- scroll(connection, result$`_scroll_id`, time_scroll="1m")
}

# save results to CSV so we can start to analyze data
fwrite(logindata, "office365-logindata.csv")
```

KUVA 15. Listamuotoisen datan kääntäminen taulukkomuotoiseksi dataksi

	AzureActiveDirectoryEventType	RecordType	ResultStatus	UserId	CreationTime	Operation	ClientIP	UserType
1	1	15	Succeeded	[redacted]GROUP.onmicrosoft.com	2019-10-27T12:19:11	UserLoggedIn	62.165.157.162	0
2	1	15	Succeeded	[redacted]	2019-10-27T11:52:10	UserLoggedIn	188.238.155.190	0
3	1	15	Succeeded	[redacted]	2019-10-27T12:11:13	UserLoggedIn	176.72.103.172	0
4	1	15	Failed	[redacted]	2019-10-27T11:23:38	UserLoginFailed	187.188.187.133	0
5	1	15	Failed	[redacted]	2019-10-27T12:04:26	UserLoginFailed	124.112.45.222	0
6	1	15	Succeeded	[redacted]	2019-10-27T12:03:44	UserLoggedIn	2409:8954:26d8:1082:9020:a773:80cc:57be	0

KUVA 16. Otos taulukkomuotoon käännetystä logindata-muuttujasta

4.5 Lokitietojen analysointi RStudiossa

4.5.1 Lokitietoihin tutustuminen

Kirjautumisdatan analysointi aloitettiin lataamalla Graylogista-järjestelmästä haettu data CSV-tiedostosta RStudioon. Aluksi huomattiin, että data sisälsi kirjautumistapahtumien lisäksi myös esimerkiksi käyttöoikeuksien päivitystapahtumia. Kirjautumistapahtumista rajattiin tarkasteltavaksi vain ne, joiden toiminto oli UserLoginFailed tai UserLoggedIn. Tämän jälkeen tapahtumat ryhmiteltiin käyttäjittäin (kuva 17), ja tutkittiin eniten lokitietoja tehneet käyttäjät. Heti ensimmäisenä paljastui, että paikallisen Active Directoryn ja Azure AD:n synkronointikäyttäjä tuotti eniten tapahtumia. Näistä tapahtumista ei kuitenkaan oltu juurikaan kiinnostuneita, joten ne rajattiin pois analyysista.

```
> logindata <- fread("office365-logindata.csv")
> # get only failed and successfull login events
> logins<-logindata[Operation == "UserLoginFailed" | Operation == "UserLoggedIn"]
>
> # Group by user id
> summarydata<-summarise( group_by(logins, UserId), count = n())
> topUsers<-summarydata[summarydata$count > 1000,] %>% arrange(desc(count))
> topUsers
# A tibble: 29 x 2
   UserId                                              count
   <chr>                                               <int>
 1 Sync_STAD3_4876ed68f205@STALAGROUP.onmicrosoft.com 12109
 2 [redacted]ennoinen@stalatube.com                    5524
 3 [redacted]@stala.com                                5213
 4 [redacted]ola@stala.com                             4359
 5 [redacted]tinen@stala.com                           4134
 6 [redacted]i@stala.com                               3241
 7 [redacted]in@stala.com                              2762
 8 [redacted]nen@stala.com                             2604
 9 [redacted]rajarvi@stalatube.com                     2451
10 [redacted]stala.com                                 2145
# ... with 19 more rows
```

KUVA 17. Eniten kirjautumistapahtumia tehneet käyttäjät

Seuraavaksi ryhmiteltiin kirjautumiset onnistuneiden ja epäonnistuneiden tapahtumien mukaan, ja tutkittiin kirjautumistapahtumien keskiarvoja ja mediaaneja. Koko aineistossa epäonnistuneita kirjautumistapahtumia oli yhteensä 80847 ja onnistuneita 65449. Epäonnistuneita kirjautumistapahtumia oli yhteensä 619 eri käyttäjältä mediaanin ollessa 10 ja keskiarvon 130,6. Tästä voitiin päätellä suuren osan epäonnistuneista kirjautumistapahtumista tulevan samoilta käyttäjiltä, joka näkyykin hyvin tarkasteltaessa viittä eniten virheellisiä kirjautua tehnyttä käyttäjää, joiden yhteenlaskettu epäonnistuneiden kirjausten määrä oli lähes 18 % kaikista epäonnistuneista kirjautumisista. Kuvasta 18 selviää, että erityisesti kahden ensimmäisen käyttäjän kirjautumiset erottuvat muusta aineistosta, joka saattaa merkitä joko viallista päätelaitetta tai yritystä murtautua heidän käyttäjätileillensä.

```
> # Get counts of successful and failed events
> operationSummary<-summarise( group_by(logins, Operation), count = n())
> operationSummary
# A tibble: 2 x 2
  Operation       count
  <chr>           <int>
1 UserLoggedIn    65449
2 UserLoginFailed 80847
>
> # Group by user id and operation
> summarydata<-summarise( group_by(logins, UserId, Operation), count = n())
>
> # Show 5 top failed users and summary of all
> failedUserLoginsPerUser<-summarydata[summarydata$Operation == "UserLoginFailed",] %>% arrange(desc(count))
> head(failedUserLoginsPerUser, 5)
# A tibble: 5 x 3
# Groups:   UserId [5]
  UserId                          Operation       count
  <chr>                           <chr>           <int>
1 [redacted]@stala.com            UserLoginFailed  5213
2 [redacted]ennoinen@stalatube.com UserLoginFailed  4876
3 [redacted]o@stala.com           UserLoginFailed  1659
4 [redacted]in@stala.com          UserLoginFailed  1470
5 [redacted]atube.com             UserLoginFailed  1309
> sum(head(failedUserLoginsPerUser$count, 5))
[1] 14527
> summary(summarydata[summarydata$Operation == "UserLoginFailed", "count"])
     count
 Min.   :   1.0
 1st Qu.:   3.0
 Median :  10.0
 Mean   : 130.6
 3rd Qu.: 255.0
 Max.   :5213.0
```

KUVA 18. Kirjautumisyritysten summat ja epäonnistuneiden kirjautumisten yhteenveto

Kuvassa 19 esitetään komennot ja tulokset, joiden avulla onnistuneita tapahtumia tutkittiin. Onnistuneiden sisäänkirjautumisten käyttäjäkohtainen mediaani oli 157 ja keskiarvo 316,2. Onnistuneissa sisäänkirjautumisissa keskiluvut ovat huomattavasti korkeampia kuin epäonnistuneissa kirjautumisissa, mutta havaittavissa on samankaltainen ilmiö, jossa viisi eniten sisäänkirjautumisia tehnyttä käyttäjää muodostaa lähes 24 % kaikista onnistuneista kirjautumisista. Kyseiset käyttäjät eivät olleet samoja kuin epäonnistuneissa kirjautumisissa, joten työssä päädyttiin tutkimaan tarkemmin näiden kymmenen käyttäjän sisäänkirjautumistietoja.

```
> # Show 5 top successful users and summary of all
> successUserLoginsPerUser<-summarydata[summarydata$Operation == "UserLoggedIn",] %>% arrange(desc(count))
> head(successUserLoginsPerUser, 5)
# A tibble: 5 x 3
# Groups:   UserId [5]
  UserId                          Operation    count
  <chr>                           <chr>        <int>
1 [redacted]@stala.com            UserLoggedIn  4310
2 [redacted]en@stala.com          UserLoggedIn  3720
3 [redacted]tala.com              UserLoggedIn  2846
4 [redacted]arvi@stalatube.com    UserLoggedIn  2430
5 [redacted]@stala.com            UserLoggedIn  2282
> sum(head(successUserLoginsPerUser$count, 5))
[1] 15588
> summary(summarydata[summarydata$Operation == "UserLoggedIn", "count"])
     count
 Min.   :   1.0
 1st Qu.:   8.0
 Median : 157.0
 Mean   : 316.2
 3rd Qu.: 436.5
 Max.   :4310.0
```

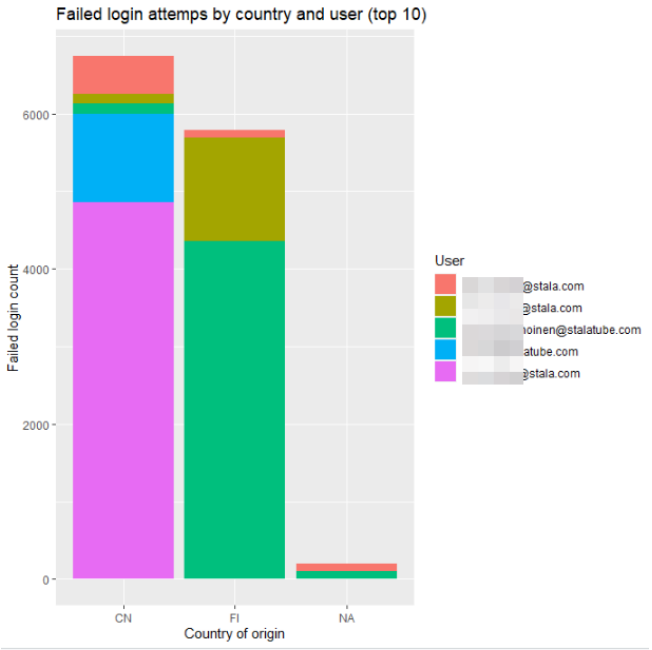
KUVA 19. Onnistuneiden sisäänkirjautumisten tutkimista

Rgeolocate-kirjaston avulla isolle osalle IP-osoitteista saatiin määriteltyä lähdemaat, joiden avulla pyrittiin tutkimaan, mistäpäin maailmaa kirjautumiset on suoritettu. Haun helpottamiseksi maakoodi lisättiin kuvan 20 mukaisesti uutena sarakkeena logins-taulukkoon merge-funktion avulla. Maxmind-funktion avulla olisi voinut myös paikallistaa jopa lähdekaupungin, mutta se olisi vaatinut Geolokaatiotietojen ostamista erikseen.

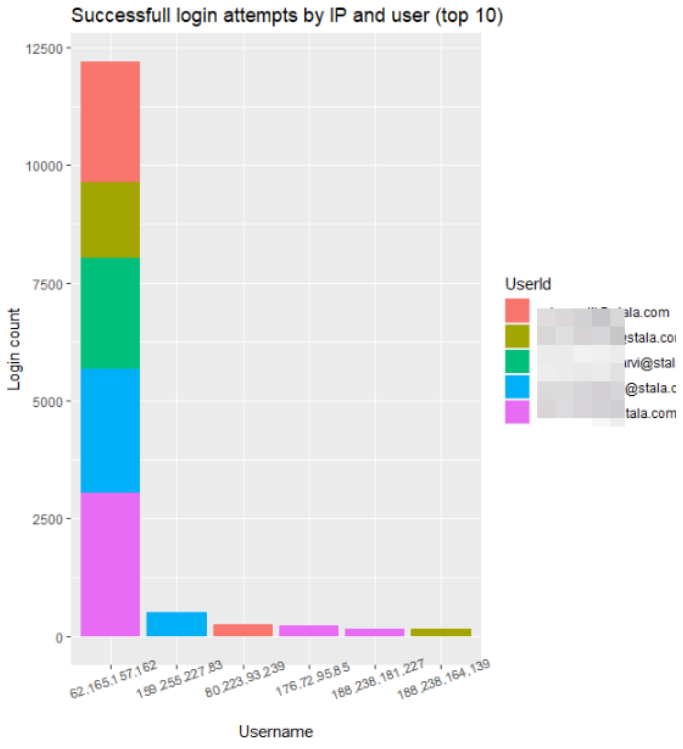
```
# Use geoip to find out country codes for IP addresss
geoipfile <- system.file("extdata","GeoLite2-Country.mmdb", package = "rgeolocate")
ipaddressesSummary<-summarise( group_by(logins, ClientIP), IPCountry = "")
ipcountrytable <- data.table(ClientIP = ipaddressesSummary$ClientIP, maxmind(ipaddressesSummary$ClientIP, geoipfile, "country_code"))

# Merge country codes to logins data
logins<-merge(logins, ipcountrytable, by="ClientIP")
```

KUVA 20. Maakoodin lisääminen logins-taulukkoon RStudiossa

Tutkittaessa epäonnistuneita kirjautumisyrityksiä, piirrettiin RStudiolla pylväskuvaaja (kuva 21), jolle annetusta datasta selvisi, että kaksi eniten merkintöjä aiheuttanutta merkintää oli yrityksen omista IP-osoitteista, ne olivat aiheutuneet todennäköisesti huonosti toimivista päätelaitteista. Myös kiinalaisista IP-osoitteista oli yritetty kirjautua poikkeuksellisen paljon. Koska kyseiset henkilöt eivät ole vierailleet Kiinassa, näiden pääteltiin olevan murtautumisyrityksiä käyttäjien sähköposteihin. Onnistuneiden kirjautumisten tapauksessa luotiin kuvan 22 mukainen pylväskuvaaja, jonka avulla saatiin selville, että suurimman lukumäärän kirjautumisia tehneet käyttäjät tekivät ne yrityksen omista IP-osoitteista. Lähdemaana oli suurimmaksi osaksi Suomi, joten tässä tapauksessa maarajauksesta päätettiin siirtyä IP-osoiterajaukseen. Kuvaajasta ei selviä, miksi juuri nämä käyttäjät ovat tehneet onnistuneita kirjautumisia huomattavasti enemmän kuin käyttäjät keskimäärin.



KUVA 21. Epäonnistuneiden kirjautumisyritysten lukumäärät lähdemaan ja käyttäjän mukaan



KUVA 22. Lukumäärältään suurimmat kirjautumismäärät käyttäjittäin ja IP-osoitteittain

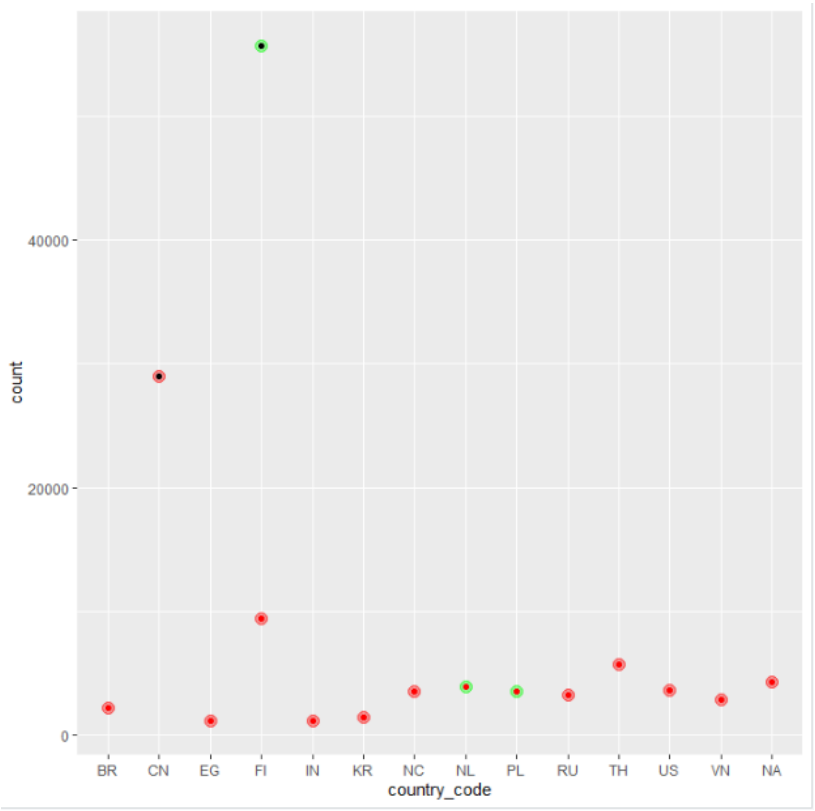
4.5.2 Lokitietojen analysointi hierarkkisen klusteroinnin avulla

Seuraavaksi RStudiossa kokeiltiin datan hierarkista klusterointia selvittääksemme löytyykö tällä tavoin datasta sellaista tietoa, jota ei edellä kuvatuilla selvitystavoilla löytynyt. Hierarkisessa klusteroinnissa pyrittiin löytämään datasta sellaisia ominaisuuksia, joiden avulla olisi voitu päätellä onnistuuko vai epäonnistuuko kirjautuminen. Vaikutti siltä, ettei millään ominaisuuksilla muodostunut järkeviä klustereita, jonka jälkeen kokeiltiin rajata datasta pois tunnetut IP-osoitteet pois. Tämä ei kuitenkaan tuntunut vaikuttavat juurikaan lopputulokseen. Lisäksi ongelmaksi tuntui muodostuvan myös R:n muistinkäyttöongelma, jonka vuoksi data piti ensin ryhmitellä käsin, jotta käsiteltäviä rivejä saatiin vähennettyä.

Kuvissa 23 ja 24 on kuvattu klusteroinnissa käytetyt komennossa ja klusteroinnin kuvaaja. Kuvaajasta voitiin päätellä, että kun molemmissa klustereissa on sekä onnistuneita kirjautumisia, että epäonnistuneita kirjautumisia, ei hierarkisella klusteroinnilla löydetä ominaisuuksia, joiden avulla voitaisiin päätellä, onnistuuko vai epäonnistuuko kirjautuminen. Kuvaajan perusteella voitiin kuitenkin todeta, että todennäköisesti Suomesta tullut kirjautuminen onnistuu, kun taas Kiinasta tuleva epäonnistuu, vaikkakin klusterointi on ryhmittänyt ne samaan ryhmään. Kuviosta nähdään myös, että Alankomaista ja Puolasta tulee niin vähän onnistuneita kirjautumisia, että ne on ryhmitelty epäonnistuneiden kanssa samaan ryhmään. Lopputuloksena todettiin, ettei hierarkisella klusteroinnilla pystytä havaitsemaan poikkeavia havaintoja yksittäisen käyttäjän kirjautumistiedoista, mutta sen avulla pystyttiin huomaamaan suuri määrä epäonnistuneita kirjautumisyrityksiä lähdemaan perusteella. Tämän tiedon myötä yrityksen tietoturvaa voitaisiin parantaa estämällä kyseisen käyttäjän kirjautuminen Kiinasta kokonaan.

```
# group by ResultStatus and country_code
loginsByCountryCode<-head(summarise( group_by(logins, ResultStatus, country_code), count = n()) %>% arrange(desc(count)), 15)
clusters <- hclust(dist(loginsByCountryCode))

clusterCut <- cutree(clusters, 2)
table(clusterCut,loginsByCountryCode$ResultStatus)

ggplot(loginsByCountryCode, aes(country_code, count, color = loginsByCountryCode$ResultStatus)) +
  geom_point(alpha = 0.4, size = 3.5) +
  geom_point(colour = clusterCut) +
  theme(legend.position = "none") +
  scale_color_manual(values = c('red', 'green'))
```

KUVA 23. Hierarkkisen klusteroinnin ja pistegraafin muodostaminen RStudiossa



KUVA 24. Hierarkkisen klusteroinnin pistegraafi

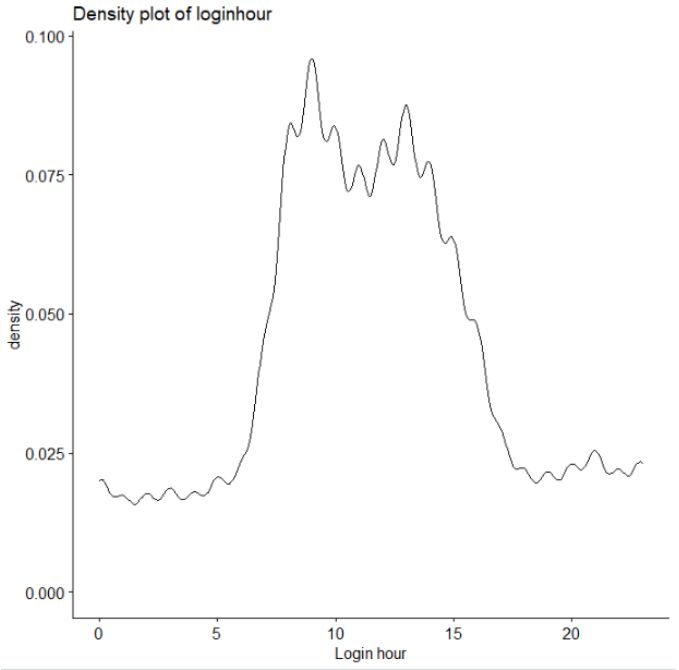
4.5.3 Lokitietojen analysointi normaalijakauman avulla

Seuraavaksi tutkittiin, noudattaako kirjautumisdata normaalijakaumaa kirjautumisten kellonaikojen suhteen. Kellonaikoja tutkimalla pyrittiin löytämään poikkeavuuksia sisäänkirjautumistiedoista sillä olettamalla, että normaalisti ihmiset kirjautuvat palveluihin aamuneljän ja iltakymmenen välillä. Analysointi aloitettiin valitsemalla datasta vain kentät käyttäjätiedosta, toiminnosta sekä tapahtuman ajankohdasta (kuva 25). Tämän jälkeen ajankohta muutettiin merkkijonosta lubridate-kirjastossa olevan ymd_hms-funktion avulla helpommin käsiteltävään date-muotoon ja tapahtuman tunti lisättiin omaksi sarakkeeksi. Tämän lisäksi aikavyöhyke vaihdettiin UTC-muotoisilta ajoilta Suomen ajaksi, jotta tuloksista saadaan todenmukaisia.

```
# get only successful login events, drop AD sync user out from results. Select only userid and timestamp
logins<-logindata[(Operation == "UserLoggedIn" | Operation == "UserLoginFailed") & UserId != "Sync_STAD3_4876ed68f205@STALAGROUP.onmicrosoft.com", 4:6]

# convert CreationTime field to lubridate
logins$CreationDatetime <- with_tz( ymd_hms(logins$CreationTime, tz = "UTC"), tzone = "Europe/Helsinki")
logins <- logins[, !"CreationTime"]

# get hour to separate field
logins$LoginHour <- hour(logins$CreationDatetime)
```

KUVA 25. Päivämäärän ja kellonajan muuntaminen lubridate-muotoon

Datalle suoritettiin kaksi testiä, joiden avulla arvioitiin noudattaako aineisto normaalijakaumaa. Ensin avulla luotiin ggdensity-funktion avulla tiheyskuvaaja (kuva 26), jota tarkastelemalla aineisto vaikuttaisi noudattavan normaalijakaumaa, joskin keskellä näkyy pieni kuoppa. Tämän jälkeen suoritettiin vielä Shapiro-Wilk testi, jonka tuloksena p-arvoksi saatiin 0,18 (kuva 27). Testin ohjeistuksen mukaan p-arvon ollessa yli 0,05 on aineisto normaalisti jakautunut (de Vries & Meys 2015, 304). Tässä yhteydessä kuitenkin huomattiin, että otoskoko tuntui vaikuttavan testin tuloksiin huomattavasti, mutta koska kuvaajakin tuntui noudattavan kellokäyrän muotoa, päädyttiin siihen, että aineisto on normaalisti jakautunut.

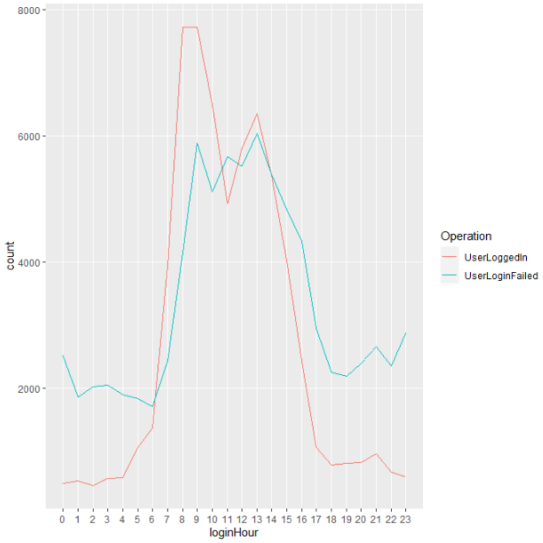


KUVA 26. ggdensity-funktion luoma kuvaaja

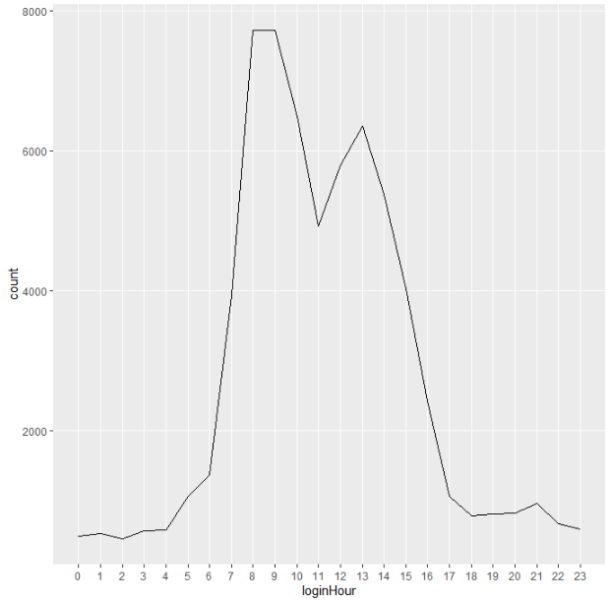
```
> #shapiro test
> set.seed(1234)
> sampleRows <- sample(1:nrow(logins), 100)
> shapiro.test(logins[sampleRows,]$LoginHour)

        Shapiro-Wilk normality test

data:  logins[sampleRows, ]$LoginHour
W = 0.9817, p-value = 0.18
```

KUVA 27. Shapiro-Wilk testin suorittaminen otokselle

Tarkastelemalla kaikkia tapahtumia tunneittain kuvasta 28 selvisi, että varsinkin sisäänkirjautumiset vaikuttaisivat muodoltaan noudattavan normaalijakaumaa. Tarkasteltaessa kuvaa 29 nähdään selvästi, kuinka suurin kirjautumispiikki osuu aamulla kello kahdeksan ja yhdeksän väliin, joka on yleisin toimistotöiden aloitusaika. Kirjautumisten määrässä näkyy kello 11 kohdalla kuoppa, jonka tulkittiin johtuvan ruokatunnin alkamisesta. Sisäänkirjautumisten määrä alkaa kasvaa aamulla kello neljältä, sen niiden määrä laskee olennaisesti kello 17 jälkeen.



KUVA 28. Kaikki kirjautumistapahtumat summattuna tunneittain



KUVA 29. Kaikkien käyttäjien sisäänkirjautumiset summattuna tunneittain

Seuraavaksi laskettiin normaalijakauman mukainen todennäköisyys sille, että käyttäjä kirjautuu kello neljän ja 22 välille. Kuvasta 30 voidaan todeta, että todennäköisyys että yksittäinen kirjautuminen osuu tälle aikavälille on ~95,6 %. Tilastollisesti p-arvo on tässä tapauksessa melkein merkitsevä. Koska yöaikaan ei ole juurikaan kirjautumisia, voitaisiin yrityksen tietoturvaa parantaa esimerkiksi pakottomalla tällöin käyttäjät käyttämään kaksivaiheista tunnistautumista.

```
> # calculate mean
> loginsMean<-mean(successLogins$LoginHour)
> loginsMean
[1] 11.23122
>
> # calculate standard deviation
> loginsSd<-sd(successLogins$LoginHour)
> loginsSd
[1] 4.120225
>
> # propability that user logins 22 or before
> a<-pnorm(22, mean = loginsMean, sd = loginsSd)
>
> # propability that user logins 4 or before?
> b<-pnorm(4, mean = loginsMean, sd = loginsSd)
>
> # propability that user logins between 4 and 22
> a-b
[1] 0.9558958
>
```

KUVA 30. Todennäköisyyksien laskenta eri kellonajoille.

4.5.4 Lokitietojen analysointi hypoteesien avulla

Normaalijakauman avulla suoritetusta testistä saatiin selville, että onnistuneet kirjautumiset tapahtumat tilastollisesti tarkasteltuna yli 95 prosenttisesti kello neljän ja 22 välillä. Kuvassa 31 tutkittiin T-testin avulla, onko kirjautumistunnilla merkitystä siinä, onnistuuko vai epäonnistuuko kirjautuminen (de Vries & Meys 2015, 306). T-testin nollahypoteesiksi H0 asetettiin olettama, että kirjautumistunnilla ei ole merkitystä kirjautumisen onnistumiselle. Vaihtoehtoinen hypoteesi H1 oli, ettei kirjautumistunnilla on merkitystä. T-testin P-arvoksi saatiin $2{,}2*10^{-16}$, eli nollahypoteesi hylättiin ja testin tuloksena oli, että kirjautumistunnilla on merkitystä kirjautumisen onnistumiselle. T-testin tulos tukee edellisessä kohdassa tutkittua mahdollisuutta tiukentaa tietoturvaa tiettyinä kellonaikoina.

```
> # test if login hour matters when comparing success and fail attempts
> successLogins = logins[Operation == "UserLoggedIn"]
> failedLogins = logins[Operation == "UserLoginFailed"]
>
> t.test(successLogins$LoginHour, failedLogins$LoginHour, paired = FALSE, alternative="two.sided", var.equal=FALSE)

        Welch Two Sample t-test

data:  successLogins$LoginHour and failedLogins$LoginHour
t = -29.561, df = 143530, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.8252814 -0.7226475
sample estimates:
mean of x mean of y
 11.23122  12.00518
```

KUVA 31. T-testin suorittaminen kirjautumisdatalle

Tämän jälkeen haluttiin tutkia, onko kirjautumismaalla tilastollisesti merkitystä kirjautumisen onnistumiseen (kuva 32). Nollahypoteesina H0 tässä tapauksessa oli ”kirjautumismaalla ei ole merkitystä kirjautumisen onnistumiseen”, kun taas vaihtoehtoinen hypoteesi H1 oli ”kirjautumismaalla on merkitystä kirjautumisen onnistumiseen”. Jotta t-testi pystyttiin suorittamaan, määritettiin aineistoon sarake foreignLogin, joka sai arvon 0 tai 1 sen mukaan, oliko kirjautumisyritys tapahtunut ulkomailta vai ei. P-arvon ($< 2{,}2 * 10^{-16}$) ollessa erittäin pieni, nollahypoteesi hylättiin ja vaihtoehtoinen hypoteesi astui voimaan. Testin tuloksena siis oli, että kirjautumismaalla on merkitystä kirjautumisen onnistumiseen.

```
> # test if origin country matters?
>
> # if country_code = fi then foreignLogin = 0, else 1
> logins$foreignLogin[logins$country_code == "FI"] <- 0
> logins$foreignLogin[logins$country_code != "FI"] <- 1
>
> successLogins = logins[Operation == "UserLoggedIn"]
> failedLogins = logins[Operation == "UserLoginFailed"]
>
> t.test(successLogins$foreignLogin, failedLogins$foreignLogin, paired = FALSE, alternative="two.sided", var.equal=FALSE)

        Welch Two Sample t-test

data:  successLogins$foreignLogin and failedLogins$foreignLogin
t = -399.7, df = 134538, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.733253 -0.726097
sample estimates:
mean of x mean of y
0.1471984 0.8768734
```

KUVA 32. Kirjautumismaan merkitys kirjautumisen onnistumiseen

Aiemmin lokidataan tutustuttaessa huomattiin, että kolme päätelaitteesta tulee erittäin paljon virheellisiä kirjautumisia. Nämä rajattiin vielä aineistosta pois, ja suoritettiin edellä tehty t-testi uudelleen. Kuvasta 33 selviää, että lopputulos ei sinällään muutu, mutta epäonnistuneiden kirjautumisten keskiarvo ulkomaalaisuuden suhteen nousee korkeaksi (0,95), jonka perusteella voidaan vahvistaa edellä huomattua tosiasiaa siitä, että suurin osa epäonnistuneista kirjautumisista tulee ulkomaisista IP-osoitteista. Näiden analyysien

perusteella yrityksen tietoturvan valvonnassa kannattaisi panostaa nimenomaan ulkomaisista IP-osoitteista saapuneisiin kirjautumisyrityksiin. Esimerkiksi vain paikallisesti Suomessa toimivilta käyttäjiltä voitaisiin sallia kirjautuminen vain Suomesta.

```
> # block [redacted] results from finland, remove na results
> failedLogins = failedLogins[!(UserId == "[redacted]com" & country_code == "FI")]
> failedLogins = failedLogins[!(UserId == "[redacted]@stalatube.com" & country_code == "FI")]
> failedLogins = failedLogins[foreignLogin != "NA"]
> successLogins = successLogins[foreignLogin != "NA"]
> t.test(successLogins$foreignLogin, failedLogins$foreignLogin, paired = FALSE, alternative="two.sided", var.equal=FALSE)

        Welch Two Sample t-test

data:  successLogins$foreignLogin and failedLogins$foreignLogin
t = -493.65, df = 108561, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.8032560 -0.7969027
sample estimates:
mean of x mean of y
0.1471984 0.9472778
```

KUVA 33. t-testi rajatummalla datalla

4.5.5 Lokitietojen analysointi päätöspuiden avulla

Viimeisenä analysointitapana testattiin perinteistä päätöspuuluokittelutapaa sekä siitä kehitettyä Random forest -luokittelijaa. Heti aluksi päädyttiin rajaamaan datasta pois kaksi eniten virheellisiä kirjautumisia tehnyttä käyttäjää, jotta tuloksista saataisiin johdonmukaisempia. Tämän jälkeen koulutusdata ja testidata jaettiin omiin muuttujiinsa kuvan 34 mukaisesti. Päätöspuumallin koulutus aloitettiin lataamalla tree-kirjasto ja suorittamalla kuvan 35 mukainen tree-funktio. Suoritus kuitenkin päättyi heti virheisiin, joista paljastui, ettei factor-muotoisessa sarakkeessa saisi olla kuin 32 eri arvoa. Tämä virhe aiheutui country_code -kentästä, jossa oli erilaisia maatunnuksia 96 kappaletta.

```
# get training and test data sets
inTrain <- createDataPartition(y=logins$Operation, p=0.7, list=FALSE)
train <- logins[inTrain,]
test <- logins[-inTrain,]
```

KUVA 34. Koulutus ja testidatan muodostaminen päätöspuumallin luontia varten

```
> tree.model <- tree(Operation ~ ., data=train)
Error in tree(Operation ~ ., data = train) :
  factor predictors must have at most 32 levels
In addition: Warning message:
In tree(Operation ~ ., data = train) : NAs introduced by coercion
```

KUVA 35. Päätöspuumallin kouluttaminen

Vaihtoehtoina olisi ollut luokitella maatunnukset esimerkiksi maanosan mukaisesti, mutta tässä työssä päädyttiin luokittelemaan maat vain sen mukaan, tuleeko kirjautumisyritys kotimaasta vai ulkomailta kuvan 36 mukaisesti. Toinen virheilmoitus aineistosta liittyvistä NA-arvoista johtui siitä, että mallin harjoittaminen kääntää automaattisesti character-tyyppiset kentät numeerisiksi, ja jollei se onnistu tässä, muodostuu kentän arvoksi NA. Nämä samat virheet saatiin toistettua useilla päätöspuita käyttävillä funktioilla, eli kyse ei ollut vain yhdestä virheellisestä kirjastosta.

```
# Decicion trees cannot handle factors over 32 categories, so let's change the country_code categories
logins$foreignLogin <- factor(ifelse(logins$country_code == "FI", "DOMESTIC", "FOREIGN"))
```

KUVA 36. Maakoodien luokittelu

Virheiden myötä aineistoa muokattiin niin, että siitä poistettiin ylimääräisenä pidettyjä kenttiä, ja lisättiin uutena kenttänä viikonpäivä kuvan 37 mukaisella komennolla. Kuvasta 38 nähdään, että lopullinen malli pyritään selvittämään viikonpäivän, kirjautumistunnin ja lähdemaan avulla. Käyttäjätieto pidettiin koulutusdatassa mukana selvyyden vuoksi. On tärkeää huomata, että luokittelun mahdollistamiseksi tekstikenttien tulee olla factor-tyyppisiä.

```
# get weekday to separate field
logins$weekday <- factor( format(as.Date(logins$CreationTime), "%a") )
```

KUVA 37. Viikonpäivän lisääminen aineistoon luontiajan avulla

	UserId	Operation	weekday	LoginHour	foreignLogin
1		UserLoginFailed	ke	9	FOREIGN
2		UserLoginFailed	ke	3	FOREIGN
3		UserLoginFailed	ke	3	FOREIGN
4		UserLoginFailed	ke	3	FOREIGN
5		UserLoginFailed	ke	3	FOREIGN

KUVA 38. Päätöspuumallin kouluttamiseen käytetyn datan rakenne

Päätöspuumallin koulutuksen jälkeen mallia koestettiin testausdatan avulla. Kuvasta 39 selviää, että mallin tarkkuus oli 90,2 %, joka itsessään ei vielä mahdollistaisi suoraan

kirjautumisen estämistä, mutta sen avulla voitaisiin parantaa yrityksen tietoturvaa esimerkiksi kohdentamalla käyttäjille erilaisia lisätunnistautumispyyntöjä. Mallin koulutusvaiheessa testattiin myös jättää pois viikonpäivä ja kellonaikatiedot, mutta tällä ei ollut vaikutusta lopputulokseen, eli vaikutti siltä, että suurin yksittäinen ominaisuus päätöspuun rakenteelle oli kirjautumisen lähdemaa. Päätöspuiden luontiin ja mallin visualisointiin löytyi useita eri vaihtoehtoja, joista lopulta päädyttiin käyttämään tree-funktiota. Train-funktiolle löytyi useampia visualisointityökaluja, mutta se ei esimerkiksi tukenut kaavamuodossa sarakkeiden pois jättämistä.

```
> require(tree)
> tree.model <- tree(Operation ~ .-UserId, data=train)
> tree.pred <- predict(tree.model, newdata=test, type = "class")
> test$pred <- tree.pred
>
> confusionMatrix(tree.pred, test$Operation)
Confusion Matrix and Statistics

                Reference
Prediction       UserLoggedIn UserLoginFailed
  UserLoggedIn          16432            1144
  UserLoginFailed        2825           20108

               Accuracy : 0.902
                 95% CI : (0.8991, 0.9049)
    No Information Rate : 0.5246
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8028

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8533
            Specificity : 0.9462
         Pos Pred Value : 0.9349
         Neg Pred Value : 0.8768
             Prevalence : 0.4754
         Detection Rate : 0.4056
   Detection Prevalence : 0.4339
      Balanced Accuracy : 0.8997

       'Positive' Class : UserLoggedIn
```

KUVA 39. Päätöspuumallin kouluttaminen ja ennusteen käyttö testidataan

Random forest-luokittelijan kouluttaminen tapahtui lähes samalla tavalla kuin perinteisen päätöspuun luokittelijan kouluttaminen (kuva 40). Funktiona käytettiin randomForest-funktiota, jolle annettiin samat lähdedatat kuin päätöspuun tapauksessa. Mallin tarkkuudeksi saatiin täsmälleen sama lukema kuin normaaleilla päätöspuuluokittelijoilla. Yksinkertaisen muutaman ominaisuuden datassa toiminnallisuus vaikuttaa olevan täysin sama molemmilla funktioilla. Tältä osin Random forest -luokittelija ei tuonut lisäarvoa lokidatan analyysiin. Testattaessa päätöspuualgoritmeja ei löydetty tapaa, jonka avulla yksittäinen käyttäjätieto olisi saatu mukaan luokitteluun. Suurimmaksi ongelmaksi muodostui se, että ominaisuuden olisi pitänyt olla factor-tyyppinen, mutta kun se muokattiin factor-tyyppiseksi, saatiin virheilmoitus, jonka mukaan muuttaja sisälsi liian monta tasoa.

```
> #
> # train model using random forest
> #
> rf.model <- randomForest(Operation ~ .-UserId, data = train)
> rf.pred <- predict(rf.model, test, type = "class")
> test$pred <- rf.pred
>
> confusionMatrix(rf.pred, test$Operation)
Confusion Matrix and Statistics

                Reference
Prediction       UserLoggedIn UserLoginFailed
  UserLoggedIn          16432            1144
  UserLoginFailed        2825           20108

               Accuracy : 0.902
                 95% CI : (0.8991, 0.9049)
    No Information Rate : 0.5246
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8028

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8533
            Specificity : 0.9462
         Pos Pred Value : 0.9349
         Neg Pred Value : 0.8768
             Prevalence : 0.4754
         Detection Rate : 0.4056
   Detection Prevalence : 0.4339
      Balanced Accuracy : 0.8997

       'Positive' Class : UserLoggedIn
```

KUVA 40. Random forest-luokittelijan mallin kouluttaminen

Koska käyttäjätietoa ei saatu mukaan päätöspuun luontiin, päätettiin kokeilla vielä päätöspuun luomista käyttäjäkohtaisesti. Ajatuksena oli tehdä silmukka, joka käy lävitse jokaisen käyttäjän kirjautumistapahtumat erikseen ja muodostaa ennusteen käyttäjäkohtaisesti. Tällä tavoin saataisiin vastattua toiseen tutkimuskysymykseen siitä, miten algoritmi saataisiin huomioimaan poikkeavuudet yksittäisen käyttäjän kirjautumistietojen perusteella.

Ensimmäiseksi ideaa testattiin muutaman käyttäjän kirjautumisten avulla, jotta voitiin todeta idean toimivan. Omalla käyttäjätunnuksella testattaessa vain kolme kirjautumista kaikista ennustettiin väärin, jonka myötä lähdettiin kehittämään algoritmia, joka käy kaikki käyttäjät lävitse. Käsiteltävä aineisto muokattiin kuvassa 41 näkyvän muotoiseksi, jonka jälkeen jokaiselle käyttäjälle koulutettiin oma päätöspuumalli ja tämän pohjalta suoritettiin ennustaminen. Aiemmasta poiketen aineistoon lisättiin myös ID-numero kuvan 42 mukaisesti, jonka pohjalta ennuste saatiin lopuksi liitettyä alkuperäiseen aineistoon validointia varten.

	ID	ClientIP	UserId	Operation	country_code	creationdatetime	weekday	loginhour	foreignlogin
1	1	1.10.167.117		UserLoginFailed	TH	2020-04-08 09:52:23	ke	9	FOREIGN
2	2	1.10.167.117		UserLoginFailed	TH	2020-04-08 09:52:23	ke	9	FOREIGN
3	3	1.161.102.82		UserLoginFailed	TW	2019-11-27 03:18:33	ke	3	FOREIGN
4	4	1.161.102.82		UserLoginFailed	TW	2019-11-27 03:18:33	ke	3	FOREIGN
5	5	1.161.102.82		UserLoginFailed	TW	2019-11-27 03:18:33	ke	3	FOREIGN
6	6	1.161.102.82		UserLoginFailed	TW	2019-11-27 03:18:33	ke	3	FOREIGN
7	7	1.161.102.82		UserLoginFailed	TW	2019-11-27 03:18:33	ke	3	FOREIGN

KUVA 41. Lokitiedot valmiina analysointia varten

```
# add rownumber as row ID to join the results in the end
logins <- rowid_to_column(logins, "ID")
```

KUVA 42. Rivinumeron lisääminen omaksi ID-sarakkeeksi

Käyttäjätietoja läpi käyvään silmukkaan muodostui lopulta kolme erilaista vaihtoehtoa, jonka mukaan ennuste tehtiin. Silmukan alussa valittiin käsiteltäväksi vain kyseisen käyttäjän kirjautumistapahtumat, jonka jälkeen testattiin, että datassa on riittävästi tapahtumia Random forest-algoritmin suorittamiseksi. Mikäli kaikki kuvan 43 ehdot täyttyivät, siirryttiin kouluttamaan luokittelumallia kuvan 44 mukaisesti viikonpäivän, kirjautumistunnin ja sen mukaan, tuliko kirjautumisyritys kotimaasta vai ulkomailta. Mallin tarkkuus testiaineiston perusteella päätettiin tallentaa testaccuracies-muuttujaan tarkempaa tarkastelua varten. Lopuksi käyttäjäkohtaisen ennustamismallin avulla suoritettiin ennustaminen kyseisen käyttäjän koko lokidatalle ja lopputulos talletettiin predictions-muuttujaan rivin ID-numeron sekä ennusteen tarkkuuden kanssa.

```
# loop every user, train the random forest model and predict the result with weekday, loginhour and origin (dc
users<-unique(logins$UserId)
for (user in users) {
  # get this users login events
  userLogins<-logins[(UserId == user)]

  # there has to be both events for randomForest to train the model. Also require 10 login events at minimum
  totalLogincountOk <- count(userLogins) > 10
  successLogincountOk <-count(userLogins[Operation == "UserLoggedIn"]) >= 2
  failLogincountOk <- count(userLogins[Operation == "UserLoginFailed"]) >= 2
```

KUVA 43. Silmukan alku ja käyttäjän kirjautumisdatan nouto

```
if(totalLogincountOk && successLogincountOk && failLogincountOk) {
  # get training and test data sets
  inTrain <- createDataPartition(y=userLogins$Operation, p=0.7, list=FALSE)
  train <- userLogins[inTrain,]
  test <- userLogins[-inTrain,]

  # train model
  rf.model <- randomForest(Operation ~ weekday+loginhour+foreignlogin, data = train)

  # predict using test set
  rf.pred <- predict(rf.model, test, type = "class")
  test$predicted <- rf.pred

  # save accuracy from confusionMatrix so we can review all random forest models testing accuracies and compare them to final prediction accuracy
  predictionAccuracy <- confusionMatrix(rf.pred, test$Operation)$overall["Accuracy"]
  testaccuracies <- rbind(testaccuracies, data.table(user = user, accuracy = predictionAccuracy))

  # predict all logins using this model
  rf.pred <- predict(rf.model, userLogins, type = "class")
  predictionAccuracy <- confusionMatrix(rf.pred, userLogins$Operation)$overall["Accuracy"]

  # store results to prediction data.table with ID
  userLogins$prediction <- rf.pred
  userLogins$accuracy <- predictionAccuracy
  predictions <- rbind(predictions, userLogins[, c("ID", "prediction", "accuracy")])
}
```

KUVA 44. Random forest-luokittelumallin kouluttaminen ja ennusteen luonti käyttäjän lokidatalle

Aineistoa tutkittaessa huomattiin, että epäonnistuneita kirjautumisyrityksiä tuli paljon myös sellaisiin käyttäjätileihin, joita ei ollut olemassa. Mikäli käyttäjän lokidata ei sisältänyt yhtään onnistunutta kirjautumistapahtumaa, ei tällöin luokittelija voinut toimia. Näissä tapauksissa kirjautumistapahtuman ennusteeksi määritettiin kuvan 45 mukaisesti UserLoginFailed, ja tarkkuudeksi 0. Kolmanteen vaihtoehtoon päädyttiin, mikäli aineisto sisälsi vain hyväksyttyjä kirjautumisia, tai kirjautumistapahtumien lukumäärä oli liian vähäinen.

```
else if(count(userLogins[Operation == "UserLoggedIn"]) == 0) {
  # if theres only UserLoginFailed events, "predict" that next events are also UserLoginFailed
  userLogins$prediction <- "UserLoginFailed"
  userLogins$accuracy <- 0
  predictions <- rbind(predictions, userLogins[, c("ID", "prediction", "accuracy")])
}
else {
  # if theres only UserLoggedIn events, "predict" that next events are also UserLoggedIn
  userLogins$prediction <- "UserLoggedIn"
  userLogins$accuracy <- 0
  predictions <- rbind(predictions, userLogins[, c("ID", "prediction", "accuracy")])
}
```

KUVA 45. Ennusteiden asettaminen, kun Random forest-luokittelijaa ei voitu käyttää

Lopuksi kaikki ennusteet yhdistettiin ID-numeron avulla alkuperäiseen aineistoon ja selvitettiin confusion matrixin avulla mallin tarkkuus (Towardsdatascience 2020). Kuvasta 46 nähdään, että algoritmin kokonaistarkkuus oli tässä tapauksessa 95,4 %, jota voidaan pitää hyvänä. Yhteensä virheellisesti luokiteltuja tapahtumia oli 7139 kappaletta. Seuraavassa suunnittelutieteellisen tutkimuksen syklissä olisi hyvä tutkia, aiheuttavatko luokittelijan ohi manuaalisesti luokitellut tapahtumat virhettä lopputulokseen ja pitäisikö ne

suodattaa kokonaan pois. Tällaisia manuaalisesti luokiteltuja tuli lopulta aineistoon 56902 kappaletta, joka on noin kolmannes kaikista testitapauksista. Tutkittaessa näitä tapauksia tarkemmin, selvisi, että näistä 16416 oli yrityksen sisäverkosta tulleita kirjautumisia, ja 1496 sellaisia, jotka oli luokiteltu virheellisesti.

Kuvasta 47 nähdään, että osalle käyttäjistä luokittelija on saanut tarkkuudeksi täydet 100 %. Kuvasta selviää hyvin, että ulkomailta tapahtuneet kirjautumisyritykset on luokiteltu tässä tapauksessa virheellisiksi, kun taas kotimaasta tulleet on luokiteltu onnistuneiksi. Yksittäisen käyttäjän mallin tarkkuus ei kerro, käytetäänkö luokittelussa hyväksi kaikkia kolmea ominaisuutta.

```
> # join prediction to logins table
> logins<-left_join(logins, predictions, by="ID")
>
>
> # check confusionMatrix for all results
> confusionMatrix(logins$prediction, logins$Operation)
Confusion Matrix and Statistics

                Reference
Prediction       UserLoggedIn UserLoginFailed
  UserLoggedIn          74396            4106
  UserLoginFailed        3033           72425

               Accuracy : 0.9536
                 95% CI : (0.9526, 0.9547)
    No Information Rate : 0.5029
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9073

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9608
            Specificity : 0.9463
         Pos Pred Value : 0.9477
         Neg Pred Value : 0.9598
             Prevalence : 0.5029
         Detection Rate : 0.4832
   Detection Prevalence : 0.5099
      Balanced Accuracy : 0.9536

       'Positive' Class : UserLoggedIn
```

KUVA 46. Ennusteiden liittäminen lokiaineiston sarakkeiksi ja algoritmin kokonaistarkkuuden arviointi

Filter

	ID	ClientIP	UserId	Operation	country_code	creationdatetime	weekday	loginhour	foreignlogin	prediction	accuracy
53	53	1.255.70.114	[redacted]stala.com	UserLoginFailed	KR	2020-03-15 02:33:36	su	2	FOREIGN	UserLoginFailed	1
577	577	103.192.76.112	[redacted]stala.com	UserLoginFailed	NP	2019-12-11 13:26:30	ke	13	FOREIGN	UserLoginFailed	1
2020	2020	109.240.191.60	[redacted]stala.com	UserLoggedIn	FI	2020-02-05 19:53:39	ke	19	DOMESTIC	UserLoggedIn	1
2021	2021	109.240.191.60	[redacted]stala.com	UserLoggedIn	FI	2020-02-05 19:53:39	ke	19	DOMESTIC	UserLoggedIn	1
2022	2022	109.240.191.60	[redacted]stala.com	UserLoggedIn	FI	2020-02-05 19:53:39	ke	19	DOMESTIC	UserLoggedIn	1
2023	2023	109.240.191.60	[redacted]stala.com	UserLoggedIn	FI	2020-02-05 19:53:39	ke	19	DOMESTIC	UserLoggedIn	1
2024	2024	109.240.191.60	[redacted]stala.com	UserLoggedIn	FI	2020-02-05 19:53:39	ke	19	DOMESTIC	UserLoggedIn	1
2049	2049	109.240.200.71	[redacted]stala.com	UserLoggedIn	FI	2020-03-10 14:06:47	ti	14	DOMESTIC	UserLoggedIn	1

KUVA 47. Ennusteen ja tarkkuuden esitys lokiaineistossa

Tarkasteltaessa alle 90 % tarkkuudella tehtyjä luokitteluja, huomattiin, että foreignlogin-ominaisuuden olisi hyvä ottaa huomioon ainakin käyttäjän kotimaa. Kuvasta 48 nähdään puolalaisen käyttäjän kirjautumistietoja, josta selviää, että luokittelija on luokitellut tapahtumat väärin, koska lähdemaa PL on tulkittu ulkomaalaiseksi osoitteeksi, vaikka kyseiselle käyttäjälle kyseessä on kotimaa.

36328	36328	183.89.215.71		UserLoginFailed	TH	2020-04-08 10:20:38	ke	10	FOREIGN	UserLoggedIn	0.879771
36461	36461	183.89.229.134		UserLoginFailed	TH	2020-04-07 17:55:03	ti	17	FOREIGN	UserLoggedIn	0.879771
36532	36532	183.89.237.105		UserLoginFailed	TH	2020-04-09 08:38:33	to	8	FOREIGN	UserLoggedIn	0.879771
37100	37100	183.89.238.3		UserLoginFailed	TH	2020-04-08 10:00:43	ke	10	FOREIGN	UserLoggedIn	0.879771
38073	38073	186.215.130.242		UserLoginFailed	BR	2020-04-08 09:40:09	ke	9	FOREIGN	UserLoggedIn	0.879771
38807	38807	188.146.100.202		UserLoggedIn	PL	2020-01-03 11:14:19	pe	11	FOREIGN	UserLoggedIn	0.879771
38808	38808	188.146.100.202		UserLoggedIn	PL	2020-01-04 18:58:09	la	18	FOREIGN	UserLoggedIn	0.879771
38809	38809	188.146.100.202		UserLoggedIn	PL	2020-01-04 18:58:09	la	18	FOREIGN	UserLoggedIn	0.879771
38810	38810	188.146.100.202		UserLoggedIn	PL	2020-01-07 00:47:21	ma	0	FOREIGN	UserLoggedIn	0.879771
38811	38811	188.146.100.202		UserLoggedIn	PL	2020-01-03 01:07:41	to	1	FOREIGN	UserLoggedIn	0.879771

KUVA 48. Puolalaisen käyttäjän kirjautumisten luokitteluvirhe

Kokonaisuutena algoritmi toimi hyvin, ja sen avulla löydettiin heti kirjautumistietoja, jotka muuten jäisivät huomaamatta. Algoritmin avulla pystytään löytämään poikkeuksia yksittäisen käyttäjän kirjautumistiedoista. Tämä vastaa suoraan toiseen tutkimuskysymykseen, ja sen osalta algoritmin toimintaa voidaan pitää onnistuneena.

5 YHTEENVETO

Tässä opinnäytetyössä tutustuttiin big dataan, siihen liittyviin teknologioihin ja analysointimenetelmiin sekä siihen, kuinka suuresta lokitietomassasta voidaan löytää poikkeavuuksia. Poikkeavuuksien etsimiseen on tarjolla useita erilaisia analysointimenetelmiä, ja käytettävä menetelmä tuleekin valita sen mukaisesti, millaista dataa ollaan käsittelemässä. Datan tutkimiseen ja koneälymallin sovittamiseen tulee käyttää riittävästi aikaa niin, että voidaan olla varmoja mallin soveltuvuudesta esimerkiksi datan luokitteluun.

Perinteiset matemaattiset tilastomenetelmät kuten keskiluvut, keskihajonta ja normaalijakauma toimivat hyvin dataa alustavasti tutkittaessa. Koneoppimisen algoritmien avulla voidaan kuitenkin datasta löytää sellaista tietoa, jota ei tilastomenetelmin pystytä havaitsemaan. Työn teoriaosuudessa perehdyttiin useisiin erilaisiin luokittelualgoritmeihin. Käytännön toteutuksessa päädyttiin testaamaan hierarkkista klusterointia sekä erilaisia päätöspuualgoritmeja, joista jälkimmäisen todettiin soveltuvan parhaiten saatavilla olevan datan analysointiin.

Käytännön toteutuksen tavoitteena oli luoda algoritmi, joka osaa huomioida poikkeavuudet yksittäisen käyttäjän kirjautumistietojen perusteella. Aluksi vaikutti siltä, ettei tavoitteeseen päästä R-kielen työkaluilla, mutta lopulta käytettäessä Random forest-luokittelijaa löydettiin keino, jonka avulla jokaiselle käyttäjälle saatiin koulutettua oma koneoppimismalli. Algoritmia kehitettäessä huomattiin, että Random forest-luokittelija vaatii toimiakseen tietyn vähimmäismäärän dataa, ja esimerkiksi vain muutamia kirjautumiskertoja tehneille käyttäjille ei tällä tavoin pystytty luomaan omaa koneoppimismallia. Näissä tapauksissa suoritettiin manuaalinen ennuste kirjautumistapahtumien määrän ja laadun perusteella. Tällä tavoin käyttäjäkirjautumisten onnistuminen pystyttiin kokonaisuutena ennustamaan testausdatasta 95,4 % tarkkuudella, jota voidaan pitää hyvänä tarkkuutena. Luokittelumalli luotiin kirjautumisen viikonpäivän, päivän tietyn tunnin ja lähdemaan perusteella.

Kehitetyn algoritmin avulla voidaan lokidatasta löytää ennustepoikkeamat, ja tällä tavoin parantaa yrityksen tietoturvaa esimerkiksi vaatimalla vahvempaa kirjautumiskäytäntöä sellaisilta henkilöiltä, joiden kirjautumiset eivät noudata ennustetta. Opinnäytetyön tuloksena saatu algoritmi tulee suorittaa lokidatalle manuaalisesti, eikä se vielä osaa huomioida esimerkiksi yksittäistä lähdemaata, vaan lähdemaat ryhmitellään kahteen ryhmään sen mukaan, tapahtuuko kirjautuminen kotimaasta vai ulkomailta.

Suunnittelutieteellisen tutkimusmenetelmän seuraavassa iterointisyklissä tullaan selvittämään, kuinka luokittelualgoritmi saataisiin huomioimaan kaikki lähdemaat erikseen, ja kuinka algoritmi saataisiin luomaan ennusteet automaattisesti esimerkiksi ajastetusti.

Seuraavien iterointisyklien aikana tullaan myös tutkimaan, voisiko algoritmi estää kirjautumisen automaattisesti, mikäli sen luoma ennuste on käyttäjäkohtaisesti riittävän tarkka. Mikäli kirjautuminen estetään automaattisesti, tulisi mallin tarkkuuden olla käyttäjäkohtaisesti erittäin lähellä 100 %. Mikäli ennustetta sen sijaan käytetään esimerkiksi kaksivaiheisen tunnistautumisen käynnistämiseen, riittää tarkkuudeksi jo saatu 95 %. Uutta dataa kertyy Graylog-järjestelmään jatkuvasti, jonka vuoksi tutkittava aineisto tullaan uudistamaan jokaisen tutkimussyklin aluksi.

R-ohjelmointikielen oppimisen suurimpana haasteena oli oikeanlaisen ja ajantasaisen tiedon löytäminen satojen erityyppisten lähteiden joukosta. Käytännön toteutuksessa kuitenkin huomattiin useaan otteeseen sen ollessa erittäin tehokas työkalu oikein käytettynä. Kokonaisuutena tutkimusta voidaan pitää onnistuneena, ja sen avulla opittua tietoa voidaan soveltaa kaikenlaisen big datan analysointiin aina geysirien purkautumistiedoista tehtaan tuotantolinjojen optimointiin.

LÄHTEET

Achari, S. 2015. Hadoop essentials. Birmingham: Packt Publishing

Dangeti, P. 2017. Statistics for Machine Learning. Birmingham: Packt Publishing.

DB-Engines. 2020. DB-Engines Ranking [viitattu 22.3.2020]. Saatavissa: https://db-engines.com/en/ranking

deRoos, D. & Zikopoulos, P. & Melnyk, R & Brown, B. & Coss, R. 2014. Hadoop for Dummies. New Jersey: John Wiley & Sons

de Vries, A. & Meys, J. 2015. R for Dummies. New Jersey: John Wiley & Sons

Dixit, B. 2016. Elasticsearch Essentials. Birmingham: Packt Publishing

Gollapudi, S. 2016. Practical machine learning. Birmingham: Packt Publishing

HBase 2020. Apache HBase [viitattu 29.3.2020]. Saatavissa: https://hbase.apache.org/

Kananen, H. & Puolitaival H. 2019. Tekoäly - bisneksen uudet työkalut. Helsinki: Alma Talent

Kroese, D. & Botev, Z. & Taimre, T. & Vaisman, R. 2020. Data science and machine learning. Florida: Taylor & Francis Group

Lantz, B. 2015. Machine learning with R – Second edition. Birmingham: Packt Publishing

Marr, B. 2015. Big Data. Hoboken Wiley

Mueller, J & Massaron, L. 2016. Machine learning for dummies. New Jersey: John Wiley & Sons

Nummenmaa, L. & Holopainen, M. & Pulkkinen, P. 2014. Tilastollisten menetelmien perusteet. Helsinki: Sanoma Pro Oy

Salo, I. 2014. Big Data ja pilvipalvelut. Jyväskylä: Docendo Oy

Theodoridis, S. 2015. Machine Learning, a Bayesian and optimization perspective. Elsevier Science & Technology

Tietosuoja. 2020. Tietosuojavaltuutetun toimisto [viitattu 20.3.2020]. Saatavissa: https://tietosuoja.fi

Towardsdatascience. 2020. Understanding Confusion Matrix [viitattu 9.5.2020]. Saatavissa:

https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Zikopoulos, P. & Eaton & deRoos. D. & Deutsch, T. & Lapis, G. 2011.

Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill Osborne Media

Xu, R. & Wunsch, D. 2009. Clustering. New Jersey: John Wiley & Sons, Inc.

LIITTEET

LIITE 1

Käyttäjäkohtaisen luokittelun tekeminen R:llä

```
#
# Creates own random forest for every userId
# Predicts the result with weekday, loginhour and ip address (domestic/foreign)
#


library(data.table)
library(lubridate)
library(dplyr)
library(rgeolocate)
require(randomForest)
require(tidyverse)


logindata <- fread("office365-logindata.csv")


set.seed(101)


# get only successful login events and select columns UserId, CreationTime, Op-
eration and ClientIP
logins<-logindata[(Operation == "UserLoggedIn" | Operation == "Us-
erLoginFailed"), 4:7]


# Use geoip to find out country codes for IP addresss
geoipfile <- system.file("extdata","GeoLite2-Country.mmdb", package = "rgeolo-
cate")
ipaddressesSummary<-summarise( group_by(logins, ClientIP), IPCountry = "")
ipcountrytable <- data.table(ClientIP = ipaddressesSummary$ClientIP, max-
mind(ipaddressesSummary$ClientIP, geoipfile, "country_code"))


# Merge country codes to logins data
logins<-merge(logins, ipcountrytable, by="ClientIP")


# remove NA country lines
logins = logins[country_code != "NA"]


# cleanup variables
rm(ipaddressesSummary, ipcountrytable)
```

```
# convert CreationTime field to lubridate
logins$creationdatetime <- with_tz( ymd_hms(logins$CreationTime, tz = "UTC"), tzone = "Europe/Helsinki")
logins <- logins[, !"CreationTime"]

# get weekday to separate field
logins$weekday <- factor( format(as.Date(logins$creationdatetime), "%a") )

# get hour to separate field and drop original
logins$loginhour <- hour(logins$creationdatetime)

# convert to factor
logins$Operation <- factor(logins$Operation)
logins$country_code <- factor(logins$country_code)

# Decicion trees cannot handle factors over 32 categories, so let's sort country codes to domestic and foreign
logins$foreignlogin <- factor(ifelse(logins$country_code == "FI", "DOMESTIC", "FOREIGN"))

# add rownumber as row ID to join the results in the end
logins <- rowid_to_column(logins, "ID")

testaccuracies = data.table(user = factor(), accuracy = numeric())
predictions = data.table(ID = numeric(), prediction = factor(), accuracy = numeric())

# loop every user, train the random forest model and predict the result with weekday, loginhour and origin (domestic/foreign)
users<-unique(logins$UserId)
for (user in users) {
  # get this users login events
  userLogins<-logins[(UserId == user)]

  # there has to be both events for randomForest to train the model. Also require 10 login events at minimum
  totalLogincountOk <- count(userLogins) > 10
  successLogincountOk <-count(userLogins[Operation == "UserLoggedIn"]) >= 2
  failLogincountOk <- count(userLogins[Operation == "UserLoginFailed"]) >= 2
```

```
  if(totalLogincountOk && successLogincountOk && failLogincountOk) {
    # get training and test data sets
    inTrain <- createDataPartition(y=userLogins$Operation, p=0.7, list=FALSE)
    train <- userLogins[inTrain,]
    test <- userLogins[-inTrain,]

    # train model
    rf.model <- randomForest(Operation ~ weekday+loginhour+foreignlogin, data =
train)

    # predict using test set
    rf.pred <- predict(rf.model, test, type = "class")
    test$predicted <- rf.pred

    # save accuracy from confusionMatrix so we can review all random forest
models testing accuracies and compare them to final prediction accuracy
    predictionAccuracy <- confusionMatrix(rf.pred, test$Operation)$overall["Ac-
curacy"]
    testaccuracies <- rbind(testaccuracies, data.table(user = user, accuracy =
predictionAccuracy))

    # predict all logins using this model
    rf.pred <- predict(rf.model, userLogins, type = "class")
    predictionAccuracy <- confusionMatrix(rf.pred, userLogins$Operation)$over-
all["Accuracy"]

    # store results to prediction data.table with ID
    userLogins$prediction <- rf.pred
    userLogins$accuracy <- predictionAccuracy
    predictions <- rbind(predictions, userLogins[, c("ID", "prediction", "accu-
racy")])
  }
  else if(count(userLogins[Operation == "UserLoggedIn"]) == 0) {
    # if theres only UserLoginFailed events, "predict" that next events are
also UserLoginFailed
    userLogins$prediction <- "UserLoginFailed"
    userLogins$accuracy <- 0
    predictions <- rbind(predictions, userLogins[, c("ID", "prediction", "accu-
racy")])
  }
  else {
```

```
    # if theres only UserLoggedIn events, "predict" that next events are also
UserLoggedIn
    userLogins$prediction <- "UserLoggedIn"
    userLogins$accuracy <- 0
    predictions <- rbind(predictions, userLogins[, c("ID", "prediction", "accu-
racy")])
  }
}

# join prediction to logins table
logins<-left_join(logins, predictions, by="ID")
logins<-data.table(logins)

# check confusionMatrix for all results
confusionMatrix(logins$prediction, logins$Operation)
```