jamk.fi

Reaaliaikainen data Microsoft Azure -pilvipalvelussa

Mikko Jokinen

Opinnäytetyö
Helmikuu 2017
Tekniikan ja liikenteen ala
Insinööri (AMK), tietotekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Jokinen, Mikko	Opinnäytetyö, AMK	Helmikuu 2017
	Sivumäärä 68	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Reaaliaikainen data Microsoft Azure -pilvipalvelussa

Tutkinto-ohjelma
Tietotekniikan tutkinto-ohjelma

Työn ohjaaja(t)
Mika Rantonen
Antti Häkkinen

Toimeksiantaja(t)
Inmics Oy
Perttu Kemiläinen

Tiivistelmä

Opinnäytetyön tavoitteena oli tutkia Microsoft Azure -pilvipalvelun reaaliaikaiseen Big Data -prosessointiin tarjoamien ratkaisujen toimintaperiaatteita ja ominaisuuksia. Tutkittavia palveluita olivat Event Hubs, Stream Analytics sekä Machine Learning. Työn toimeksiantajana toimi Inmics Oy. Työllä haluttiin antaa toimeksiantajalle kokonaiskuva tutkittavista palveluista sekä helpottaa niiden käyttöönoton mahdollisuuksien kartoittamista tulevaisuudessa.

Työ toteutettiin testaamalla palveluita kahdessa erilaisessa kokoonpanossa. Ensimmäisessä testauksessa testidataa syötettiin Event Hubs -palvelun kautta Stream Analytics -työlle, jolla dataan suoritettiin reaaliaikaisia kyselyitä. Toisessa testauksessa analysointi pyrittiin toteuttamaan Machine Learning -ratkaisulla, jota kutsuttiin Stream Analytics -työn kyselyssä. Testaus pystyttiin suorittamaan kokonaisuudessaan käyttäen palveluiden ilmaisia testijaksoja.

Event Hubs ja Stream Analytics -palveluiden ominaisuudet saatiin esitettyä kattavasti ja testikäytön perusteella todettiin, että yhdistelmä pystytään ottamaan käyttöön hyvin tehokkaasti ilman vuosien kokemusta, mutta dokumentaatioista löytyi vielä parantamisen varaa. Työssä myös havainnollistettiin kokonaisen Machine Learning -ratkaisun luominen ja käyttöönotto, jossa onnistumisen tärkeimmiksi määrittäviksi tekijöiksi nostettiin riittävän hyvä suunnittelu sekä perustaidot ohjelmoinnista. Yleisesti ottaen palveluiden todettiin olevan vielä hieman kehitysvaiheessa, mutta esimerkiksi dokumentaation huomattiin lisääntyvän ja kehittyvän jatkuvasti. Palveluiden arvioitiin myös kasvattavan asemaansa referenssikohteiden lisääntyessä tulevaisuudessa.

Avainsanat (asiasanat)
Microsoft, Azure, Big Data, Event Hubs, Stream Analytics, Machine Learning

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Jokinen, Mikko	Bachelor's thesis	February 2016
		Language of publication: Finnish
	Number of pages 68	Permission for web publication: x

Title of publication

Real-time data in Microsoft Azure

Degree programme

Information Technology

Supervisor(s)

Mika Rantonen

Antti Häkkinen

Assigned by

Inmics Oy

Perttu Kemiläinen

Abstract

The purpose of the thesis was research how real-time data can be utilized with Microsoft Azure's Big Data-oriented services, which were Event Hubs, Stream Analytics and Machine Learning. The bachelor's thesis was assigned by Inmics Oy with the goal to get a bigger picture of the operational principles of the services so that the possible future deployments could be even considered.

The services were tested in two segments and with two different set of services used. In the first segment, the test data was fed to the Stream Analytics through Event Hubs and then real-time queries were performed to the data. In the second segment, the whole analysis was carried out using the Machine Learning -model, which was called from a Stream Analytics query. All tests were made using a free trial period of each service.

As a result, the features of the Event Hubs and Stream Analytics services were represented extensively, and it was discovered that the combination of these services can be implemented highly efficiently without having years of experience. The creation and implementation of a whole Machine Learning -solution were also demonstrated successfully. Sufficient planning and the basic knowledge of coding were noticed to be the most important factors for succeeding in the process.

Broadly based on the user experience, it could be said that the services are still slightly under progress, and the major problem with all services was the occasional lack of proper documentation. However, the usability of the services was estimated to improve along the increase of the real life use cases in the future.

Keywords/tags (subjects)

Microsoft, Azure, Big Data, Event Hubs, Stream Analytics, Machine Learning

Miscellaneous

Sisältö

Kuviot

Taulukot

Lyhenteet

ASA	Azure Stream Analytics
AML	Azure Machine Learning
AMQP	Advanced Message Queuing Protocol
CSV	Comma Separated Values
GRS	Geographically Reduntant Storage
HTTPS	Hypertext Transfer Protocol Secure
IoT	Internet of Things
JSON	JavaScript Object Notation
LRS	Locally Reduntant Storage
RA-GRS	Read Access Geographically Reduntant Storage
SAS	Shared Access Signature
SQL	Structured Query Language

1 Opinnäytetyön lähtökohdat

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi Inmics Oy, joka on vuodesta 1989 asti toiminut suomalainen IT-palvelutalo. Yritys suunnittelee ja toimittaa kokonaisia IT-ratkaisuja niin pienille muutaman hengen yrityksille kuin myös suurille kansainvälisille yhtiöille. Inmics Oy myös valvoo IT-ympäristöjen toimintaa ja toimii tukipalveluna usealla eri tasolla aina loppukäyttäjälle asti. Tukea on saatavilla jopa ympäri vuorokauden ja vuoden jokaisena päivänä. Loppukäyttäjän kouluttaminen on myös osa tarjolla olevaa kokonaisuutta. (Yritys n.d.)

Yrityksen päätoimipiste löytyy Jyväskylästä. Pienempiä toimipisteitä sijaitsee Helsingissä, Lahdessa, Tampereella ja Kuopiossa (Yhteystiedot n.d.). Tällä hetkellä työntekijöitä on yli 135. Yritys on Suomen suurin yksityisomistuksessa oleva IT-palvelutalo, jonka omistaa Jukka Autere kahden poikansa, Timon ja Tomin, kanssa. (Perhe takaa kestävän kasvun n.d.) Viimeisimmän tilikauden liikevaihto oli yli 32 miljoonaa euroa (Liikevaihto ja henkilöstö n.d.).

1.2 Toimeksianto

Opinnäytetyön tavoitteena oli tutustua Microsoft Azure -pilvipalvelun reaaliaikaisen datan analysointiin ja prosessointiin kehitettyihin ratkaisuihin, joita ovat Azure Stream Analytics ja Machine Learning. Lisäksi tutustuttiin Azure Event Hubs -palveluun, jolla dataa voidaan syöttää reaaliaikaisesti Azure-pilvipalveluun.

Palveluiden testauksen tavoitteena oli saada niistä käyttökokemusta ja sitä kautta luoda käsitystä niiden soveltuvuudesta jatkokäyttöön. Testauksessa oli tarkoitus käyttää jotain sopivaa testidataa, jota voidaan helposti generoida Event Hubille. Muita vaatimuksia testidatalle tai käyttökohteelle ei asetettu, kunhan testaus vain toisi mahdollisimman monipuolisen käsityksen palveluiden käyttämisestä.

Työn teoriaosuudessa avataan big dataa ja sen eri osa-alueita sekä miten suurten datamassojen käsittelyä toteutetaan. Azuren tarjoamista työkaluista käydään läpi tärkeimpänä Streaming Analytics, Event Hubs sekä Machine Learning. Näitä työkaluja

käyttäen luotiin kaksi erilaista testimallia, joiden käyttötarkoitus ja luominen on kuvattu yksityiskohtaisesti.

1.3 Tutkimustavat

1.3.1 Tutkimusmenetelmän valinta

Ennen tutkimusmenetelmien valintaa on tehtävä tutkimuksen ongelmanasettelu. Siinä pyritään selventämään mahdollisimman hyvin, mitä tutkitaan ja miksi. Tutkimusmenetelmien valinta on helpompaa, jos ongelma on tarkasti kuvattu ja tutkija ymmärtää tutkittavan kohteen. (Hirsjärvi, Remes & Sajavaara 2007, 119.)

Tutkimusmenetelmien tarkoitus on selventää, millä menetelmillä löydetään parhaiten ratkaisu tutkimuksen ongelmanasettelussa määriteltyyn ongelmaan. Menetelmien valinta on oltava myös perusteltavissa. (Hirsjärvi ym. 2007, 120.)

1.3.2 Kvantitatiivinen tutkimus

Määrällistä eli kvantitatiivista tutkimusta käytetään yleisesti sosiaali- ja yhteiskuntatieteissä. Tämä tutkimusmenetelmä on kuitenkin lähtöisin luonnontieteellisiltä aloilta ja ajatuksesta, että kaikki tieto muodostuu loogisen päättelyn ja aistihavaintojen kautta. Määrällisessä tutkimuksessa on tärkeää, että kerätyt aineistot soveltuvat numeeriseen mittaamiseen ja että tulokset ovat tilastollisesti käsiteltävissä. Varsinaiset päätelmät tehdään sitten perustuen näihin tilastollisiin tuloksiin. (Hirsjärvi ym. 2007, 135.)

Kvantitatiivisen tutkimuksen ensimmäisiä vaiheita on jo saatavilla olevan tiedon tutkiminen ja aiempien tutkimusten tulosten tarkastelu sekä niiden huomioon ottaminen uutta tutkimusta tehdessä. Ongelmanasettelun jälkeen määritellään myös aihealueen keskeisimmät käsitteet. Lisäksi voidaan asettaa hypoteeseja, mikäli se sopii tutkimuksen tyyppiin ja tavoitteisiin. (Hirsjärvi ym. 2007, 136.)

1.3.3 Kvalitatiivinen tutkimus

Kvalitatiivisessa eli laadullisessa tutkimuksessa on tarkoituksena kuvata jotain todellisen elämän ilmiötä, ja myös aineisto kerätään todellisissa tilanteissa. Aineisto muodostetaan omien tai muiden koehenkilöiden havaintojen perusteella. Tutkimuksessa ei tehdä hypoteeseja, vaan yritetään löytää odottamattomia seikkoja ainoastaan tarkastelemalla kerättyä aineistoa. (Hirsjärvi ym. 2007, 157,160.)

1.3.4 Työssä käytettävät menetelmät

Kvantitatiivista ja kvalitatiivista tutkimusta voidaan käyttää toisiaan täydentävinä tutkimusmenetelminä. Näiden kahden menetelmän yhdistämiseksi on erilaisia tapoja, mutta yksi niistä on käyttää kvantitatiivista menetelmää ennen kvalitatiivista tutkimusvaihetta. (Hirsjärvi ym. 2007, 132-133.)

Tässä opinnäytetyössä käytettiin sekä laadullista että määrällistä tutkimusmenetelmää rinnakkain. Opinnäytetyön tietoperusta luotiin hyvin pitkälti noudattaen kvantitatiivisen tutkimuksen teoriaa ja käsitteitä koskevia piirteitä. Käytännön testausosuudesta tehdyt johtopäätökset perustettiin taas paljolti omien havaintojen ja empiiristen kokemusten varaan.

2 Big Data

2.1 Big Data käsitteenä

Big Data kuulostaa yksinkertaiselta termiltä, mutta syvemmin tarkasteltuna sen alta avautuu erilaisia näkökulmia. Yleensä Big Datan ajatellaan tarkoittavan vain suurta määrää dataa, mutta sen pääpiirteisiin kuuluvat myös, että data on vaihtelevan muotoista ja kasvaa kovalla vauhdilla. Käsitteellä viitataan myös suurten datamassojen käsittelyyn ja analysointiin kehitettyihin teknologioihin ja ratkaisuihin. (Salo 2013, 10.)

Tarkemmin ottaen Big Datalla viitataan sellaiseen tietoon, jota ei pystytä käsittelemään millään jo olemassa olevilla työkaluilla. Tämä taas on johtanut arvokkaan tiedon valumiseen hukkaan ainoastaan sen vuoksi, ettei sitä aikaisemmin ole pystytty

hyödyntämään. Esimerkiksi erilaiset yritykset säilyttävät suuria määriä arvokasta tietoa, jonka perusteella voitaisiin tehdä ennakoivia liiketoimintapäätöksiä, mikäli tietomassasta vain saataisiin puristettua oleellinen tieto ulos. (Eaton, Deutsch, deRoos, Lapis & Zikopoulos 2012, 3.)

Hyvin yleinen esimerkki tällaisesta tietotyypistä on lokidata. Lokeja on aina tuotettu ja kerätty erilaisista järjestelmistä, mutta niiden tutkiminen on tapahtunut lähinnä järjestelmien virhetilanteiden yhteydessä. Big Data -ratkaisujen avulla lokitiedoista voisi tunnistaa eri tapahtumien väliset riippuvuudet ja sen myötä ennakoida tulevaisuudessa vastaavanlaisia virhetilanteita. (Salo 2013, 17-18.)

Kuten aiemmin mainittiin, Big Datalla viitataan johonkin suurella vauhdilla kasvavaan erimuotoiseen datamassaan. Tähän perustuu myös ns. kolmen V-kirjaimen malli, joka tulee englanninkielisistä sanoista volume, variety ja velocity. V-kirjaimiin perustuva malli on hyvin yleisessä käytössä, kun puhutaan Big Datan määritelmästä. (Eaton, Deutsch, deRoos, Lapis & Zikopoulos 2012, 5.)

Kolmen V-kirjaimen käsite on hyvin yksinkertainen. *Volume* eli volyymi viittaa tiedon suureen määrään. *Velocity* eli vauhti taas kuvastaa tiedon määrän räjähdysmäistä kasvupyrähdystä ja sitä, kuinka sitä pitäisi pystyä myös hyödyntämään aina vain nopeammin. *Variety*-sanalla taas viitataan siihen, kuinka vaihtelevan muotoista data on. (Salo 2013, 21.)

Nämä kolme tärkeintä V-kirjainta ovat siis vakiinnuttaneet asemansa Big Datan yleisessä määritelmässä. Malli on kuitenkin saanut useita uusia epävirallisempia V-alkuisia sanoja rinnalleen, joihin törmää usein lukiessaan Big Data -keskeisiä julkaisuja. Esimerkiksi *veracity*- ja *validity*-sanojen lisääminen tulee siitä, voidaanko tietoa pitää luotettavana ja paikkaansa pitävänä. Value-sanalla taas tarkoitetaan datamassasta hyödynnetyn tiedon lopullista arvoa ja sitä, kuinka kustannustehokasta sen hankkiminen on. (Gupta, Khan & Uddin 2014.)

2.2 Strukturoitu ja strukturoimaton data

Kaikki olemassa oleva tieto voidaan jakaa luonteeltaan joko strukturoituun tai strukturoimattomaan dataan, jotka ovat toistensa vastakohtia. Näiden kahden käsitteen

välille voidaan sijoittaa myös semistrukturoitu data, joka ei ole siis täysin kumpaakaan edellä mainituista. Kaikesta tiedosta suurin osa on kuitenkin strukturoimatonta dataa. (Salo 2013, 25.)

Strukturoidulla tiedolla on aina jokin malli, joka määrittelee tiedon rakenteen ja tyypin sekä tietojen yhteydet toisiinsa. Esimerkiksi perinteiset relaatiotietokannat vastaavat tämän tyyppistä dataa. Strukturoimatonta tietoa taas ei voida järjestellä mitenkään rakenteellisesti. Tällaista dataa ovat esimerkiksi kuvat ja videot. Semistrukturoitua dataa on esimerkiksi video, jossa on liitettynä metatietoja. (Salo 2013, 25.)

2.3 Tiedostomuodot

2.3.1 CSV

Comma-Separated Values format (CSV) on tiedostomuoto, jossa jokaisen tapahtuman tiedot ovat eri riveillä ja yksittäisen tapahtuman sisällä olevat tietokentät ovat eritelty pilkulla. CSV-tiedostossa voi olla mukana myös erillinen otsikkorivi, joka sisältää otsikot tapahtumien jokaiselle tietokentälle. Jokaisella rivillä tulee olla sama määrä eri tietokenttiä ja rivin viimeisen tiedon jälkeen ei tule lainkaan pilkkua. (RFC 4180:2005, 1-2.)

2.3.2 JSON

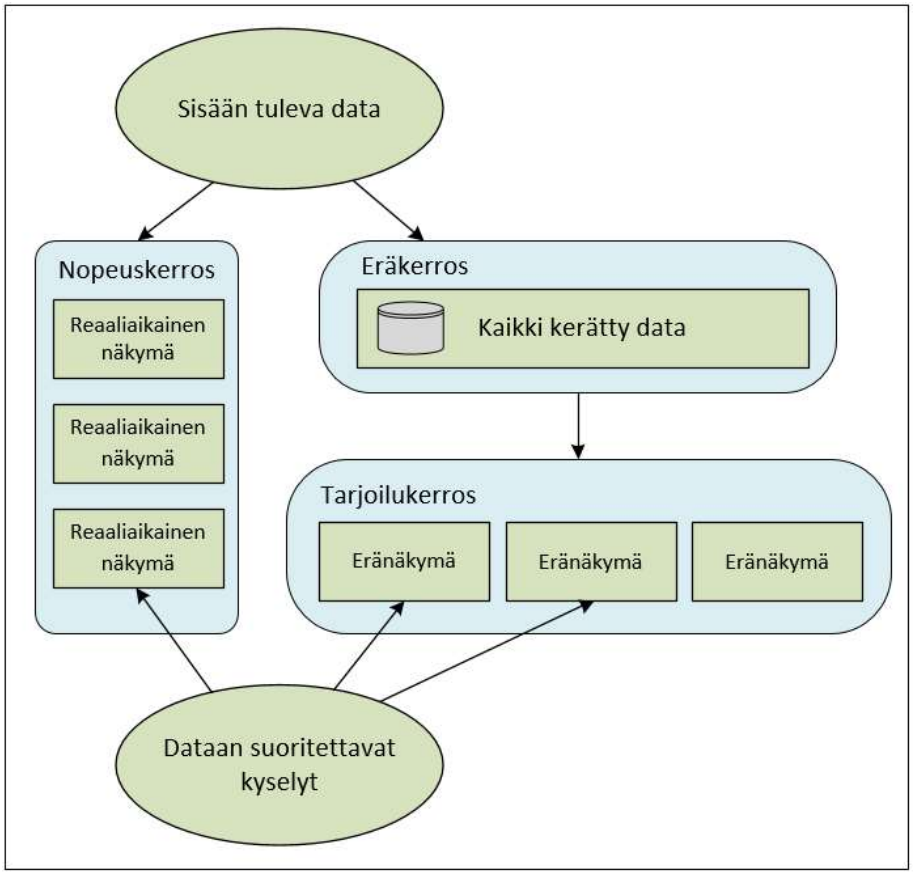
JavaScript Object Notation (JSON) on yksinkertaisempi tiedostomuoto, joka voi sisältää numeroita ja merkkijonoja, sekä boolean- ja null-arvoja. Lisäksi tiedostot voivat sisältää objekteja sekä taulukoita. Objekti yhdistää arvolle nimikentän, joka on aina merkkijono. Taulukko taas on tavallaan kuin lista useammasta eri arvosta. (RFC 7159, 3.)

3 Lambda-arkkitehtuuri

3.1 Yleistä

Lambda-arkkitehtuuri on mm. Apache Storm -projektin kehittäneen Nathan Marzin luoma toteutusmalli kokonaisilla Big Data -järjestelmille. Sen tavoitteena on, että datan käsittelystä saataisiin mahdollisimman skaalautuvaa ja vikasietoista ilman, että se tapahtuu käsittelynopeuden kustannuksella. Käytännössä tämä tapahtuu käyttämällä rinnakkain kahta eri käsittelytapaa. Näitä tapoja ovat datan käsittely isommissa erissä sekä reaaliaikainen datan käsittely. (Deshmane 2015, 6.)

Arkkitehtuuri jaetaan kolmeen eri kerrokseen (ks. kuvio 1). Kerroksille ei löydy virallisia suomenkielisiä nimityksiä, joten käytettäköön niistä seuraavia vapaita suomennoksia: eräkerros (eng. Batch Layer), tarjoilukerros (eng. Serving Layer) ja nopeuskerros (eng. Speed Layer). Näiden kerrosten tehtävä on siis tukea toisiaan ja muodostaa toimiva järjestelmä datan prosessoinnille tai ainakin toimia perusajatuksena sellaisen rakentamiselle. (Marz & Warren 2015, 14.)



Kuvio 1. Lambda-arkkitehtuurin rakenne (alkup. kuvio ks. Marz & Warren 2015, 19.)

3.2 Eräkerros

Eräkerros on Lambda-arkkitehtuurin toinen pääosista, jonne kaikki kerätty data varastoidaan yhdeksi tietomassaksi. Koko tietomassan käyminen läpi uudestaan jokaisella kyselyllä olisi hyvin raskasta, joten eräkerroksella tähän tietomassaan tehdään suuri määrä satunnaisia kyselyitä ja niistä muodostetaan sen hetkisiä eränäkymiä. Tavallaan iso datamäärä jaetaan siis pienempiin, ennalta järjesteltyihin osiin, joihin voidaan tehdä kyselyitä huomattavasti nopeammin. Vaikka datan käsittelyä saadaan tällä tavoin huomattavasti nopeammaksi, ei sen lukeminen edelleenkään ole reaaliaikaista. Eräkerroksella keskitytään enemmän datan syvempään analyysiin, joka luonnollisesti vie enemmän aikaa. (Marz & Warren 2015, 83-84.)

3.3 Tarjoilukerros

Tarjoilukerros toimii tavallaan eräkerroksen jatko-osana. Sen tehtävänä on ottaa eräkerroksen laskemat eränäkymät ja indeksoida ne. Lisäksi tarjoilukerros toimii rajapintana, jonka kautta eränäkymien muodostamaan dataan voidaan suorittaa reaaliaikaisia kyselyitä, vaikka tarjoilukerrokselle tullut data ei itsessään voikaan olla reaaliaikaista. (Marz & Warren 2015, 179.)

3.4 Nopeuskerros

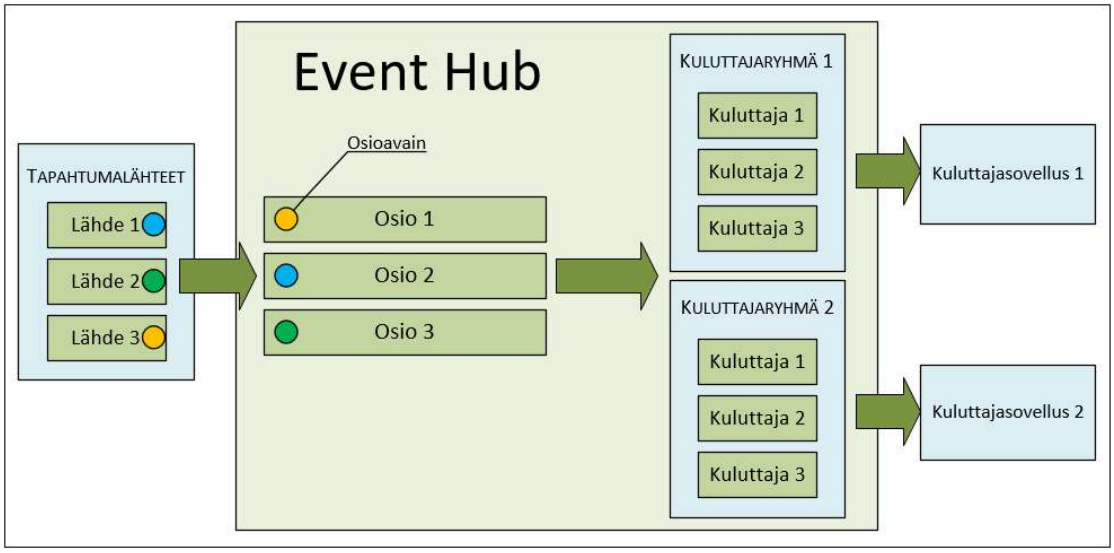
Nopeuskerroksen tarkoituksena on muodostaa reaaliaikaisia näkymiä dataan, jota eräkerros ei ole vielä ehtinyt käsittelemään loppuun asti. Toimintaperiaate eroaa eräkerroksesta siten, että nopeuskerros käsittelee kaiken kerättävän datan sijaan ainoastaan päivitetyn datan. Silloin kun eräkerroksen data on luovutettu tarjoilukerrokselle valmiiksi kyselyjä varten, voidaan nopeuskerroksella lopettaa saman datan käsittely ja vapauttaa resursseja uuden datan analysointiin. Nopeuskerroksella tapahtuva käsittely on kuitenkin huomattavasti alttiimpi virheille, kuin eräkerroksella tehty syvempi analysointi. Tämän vuoksi eräkerroksen dataa käytetään täydentämään reaaliaikaisen datan virheet. (Marz & Warren 2015, 20,208.)

4 Microsoft Azure Event Hubs

4.1 Yleistä

Event Hubs on palvelu, jonka kautta data voidaan kerätä ja syöttää Azuren muiden työkalujen hyödynnettäväksi. Se on optimoitu käsittelemään suuria määriä laitteilta ja sovelluksilta jatkuvasti virtaavaa dataa pienillä viiveillä. Se pystyy jopa miljoonien tapahtumien käsittelyyn sekunnissa. (Manheim 2016.)

Event Hubin toimintaperiaate perustuu pääasiassa osioihin sekä tapahtumien kuluttajaryhmiin ja kuluttajiin (ks. kuvio 2). Kaikkien Event Hubs -palveluun eri tapahtumalähteiltä saapuvien tapahtumien data tallennetaan ja järjestellään eri osioihin, joilta kuluttajaryhmissä olevat tapahtumien kuluttajat sitten lukevat sitä vuorollaan. (Manheim 2016.)



Kuvio 2. Event Hubsin toimintaperiaate (alkup. kuvio ks. Manheim 2016.)

4.2 Tapahtumalähteet

Tapahtumalähteellä tarkoitetaan virtuaalista pistettä, jonka kautta laite tai sovellus pystyy lähettämään dataa Event Hubsiin. Jokaiseen tapahtumalähteeseen suositellaan liitettävän vain yksi sovellus tai laite kerrallaan paremman hallittavuuden vuoksi. Lähettämiseen käytetään joko HTTPS (Hypertext transfer protocol secure) tai AMQP 1.0 (Advanced Message Queuing Protocol) -protokollaa. (Manheim 2016.)

Tapahtumalähteet ja Event Hubs käyttävät toistensa tunnistamiseen tokeneita jotka allekirjoitetaan käyttämällä 256-bittistä Shared Access Signature (SAS) -avainta. Lähettäjälle voidaan määrittää oikeudet erikseen lähettämiseen, vastaanottamiseen tai hallintaan käyttäen publisher policy -ominaisuutta. Uutta Event Hubsia luodessa luodaan automaattisesti myös uusi SAS-avain, jolla on oletuksena kaikki em. oikeudet. Avaimia voidaan kuitenkin luoda lisää tarpeen mukaan. (Manheim 2016.)

4.3 Osiot ja osioavaimet

Event Hubs käyttää datan tallentamiseen ja järjestelyyn osioita. Osioita ei täytetä tasaisesti, vaan jokaisen yksittäisen tapahtuman data sijoitetaan aina kokonaisuudessaan yhteen osioon ja se merkitään osioavaimella. Data säilytetään osiossa vain ennalta määritetyn ajan verran. Tämä aika määritetään uutta Event Hubia luodessa, ja sama aika pätee kaikkiin hubin osioihin. Jokaisen tapahtuman alkupiste merkitään

osiolle käyttämällä joko aikaleimaa tai erityistä tapahtuman alkupisteen merkitsevää arvoa. Tätä arvoa käyttämällä tapahtumadatan kuluttajat tietävät, missä kohtaa minkäkin tapahtuman data tarkalleen sijaitsee. (Manheim 2016.)

Event Hub pitää kirjaa eri tapahtumien lähettäjien käyttämistä osioista hyödyntämällä osioavaimia. Jokaiselle tapahtumien lähettäjän lähettämälle tapahtumalle määritellään oma osioavain, ja kaikki tällä osioavaimella vastaanotettu data löytyy aina samalta osiolta. Tällä tavalla tapahtuman lähettäjän ei tarvitse tietää Event Hubin muusta toiminnasta mitään muuta kuin oma osioavaimensa. (Manheim 2016.)

4.4 Kuluttajat ja kuluttajaryhmät

Jokainen tapahtumadataa kuluttava sovellus määritellään Event Hubin näkökulmasta omaan kuluttajaryhmäänsä, jonka kautta sovellukset pääsevät käyttämään osioille tallennettua dataa. Kuluttajaryhmä koostuu useista eri kuluttajista, jotka voidaan ajatella taas yksittäisinä prosesseina, joilla dataa käsitellään. Useampi eri kuluttaja pystyy toimimaan rinnakkain, mutta kuitenkin siten, että ainoastaan yksi kuluttaja voi käyttää yhtä osiota kerrallaan. (Manheim 2016.)

Osioiden käytöstä pidetään kirjaa käyttämällä tarkistuspisteitä. Näitä pisteitä käytetään ikään kuin merkitsemään, mitä kohtaa datassa mikäkin kuluttaja käsittelee. Näin muut kuluttajat tietävät, että osio on aktiivisena eikä sitä voida käyttää. Lisäksi yhteyden katketessa merkinnän tehnyt kuluttaja pystyy itse jatkamaan siitä mihin viimeksi jäi. Kuluttaja myös ilmoittaa, milloin on saanut käsittelyn valmiiksi, jolloin osio on taas vapaa käytettäväksi. (Manheim 2016.)

4.5 Hinnoittelu

Event Hubs -hinnoittelumalli koostuu Basic ja Standard -luokista. Hintaluokkien hintaerot ja ominaisuudet ovat esitetty taulukossa 1. Hinnat voivat vaihdella riippuen valitusta maantieteellisestä alueesta. Esitetyt hinnat ovat Pohjois-Euroopan alueen hintoja. Yhdeksi tapahtumaksi luokitellaan aina 64 KB dataa. Mikäli yksittäinen vastaanotettava viesti ylittää tuon koon, se määritellään useammaksi tapahtumaksi. Esimerkiksi 96 KB kokoinen viesti luokitellaan kahdeksi tapahtumaksi ja hinnoittelu tapahtuu sen mukaisesti. (Event Hubs Pricing n.d.)

Taulukko 1. Event Hubsin hinnoittelu (Event Hubs Pricing n.d.)

Hinnoitteluluokka	**BASIC**	**STANDARD**
Sisään syötettävien tapahtumien hinta	0,0236 € / milj. tapahtumaa	0,0236 € / milj. tapahtumaa
Throughput unitin hinta	0,0126 € / h	0,0253 € / h
Pyblishet policyt käytössä	Ei	Kyllä
Kuluttajaryhmien määrä	1 kpl	20 kpl
Yhteyksien määrä	100 kpl	1000 kpl
Tapahtumien säilytys	1 vrk	1 vrk
Arkistointi	Ei käytössä	0,0422 € / h

Kaikkien Event Hubien tehokkuutta voidaan lisätä throughput unit -yksikköjen määrällä. Yksi yksikkö sisältää sisään tulevaa liikennettä 1 MB/s verran tai vaihtoehtoisesti 1000 tapahtumaa sekunnissa. Ulosmenevää liikennettä voi lähteä 2 MB/s verran tai 2000 tapahtumaa sekunnissa. Liikennettä rajoitetaan automaattisesti, mikäli hankittu käyttökatto ylittyy. Sisään tulevan ja ulos lähtevän liikenteen käyttökatot ovat eritelty sen vuoksi, ettei tapahtumalähteiden lähettämisen rajoittaminen vaikuttaisi dataa kuluttaviin sovelluksiin ja päinvastoin. Yksikkö sisältää myös 84 GB tapahtumien säilytystilaa. Mikäli datan määrä ylittää ostettujen yksikköjen sisältämän tallennustilan, niin ylimenevä osuus laskutetaan Blob -tallennustilan hinnoittelun (ks. kappale 4.2) mukaisesti. Yksikköjä voidaan ostaa molemmissa hintaluokissa maksimissaan 20 kpl, mutta niitä pystyy hankkimaan lisää ottamalla yhteyttä Azuren tukeen. (Event Hubs Pricing n.d.)

Standard -luokassa pystyy ottamaan käyttöön myös tapahtumien arkistoinnin, jolla tapahtumia voidaan säilyttää myös pidemmän ajan kuin yhden vuorokauden. Arkistoinnin hinnoittelu määräytyy tunneittain ja käytettyjen yksikköjen määrän perusteella. Jos käytössä on esimerkiksi kolme yksikköä ja arkistointi on otettu käyttöön, niin se kustantaa 3 * 0,0126 € = 0,0378 € joka tunti. (Event Hubs Pricing n.d.)

Kaikille opinnäytetyössä mainittaville Azuren -palveluille luvataan vähintään 99,9% saatavuus. Saatavilla on myös erilaisilla ominaisuuksilla varustettuja tukivaihtoehtoja.

Halvin, yksittäisille käyttäjille suunnattu Developer -tuki maksaa 24,46 euroa kuukaudessa, jossa luvattu vasteaika on alle kahdeksan tuntia. Kallein tukisivustolla mainittu vaihtoehto on Professional direct, joka on tarkoitettu liiketoimintaan kriittisesti vaikuttaville palveluille. Vasteaika on alle tunnin ja se sisältää myös neuvontapalveluita. Hinnaksi on ilmoitettu 843,30 euroa kuukaudessa. Laajin tukivaihtoehto sisältää asiakkaalle räätälöidyn täyden tuen alle 15 minuutin vasteajalla. Palvelulle ei ole ilmoitettu kiinteää hintaa. (Azure support plans n.d.)

5 Microsoft Azure Blob Storage

Microsoft Azure tarjoaa erilaisia tallennustilaratkaisuja erilaisiin tarkoituksiin. Näistä yksi on Blob Storage, joka on tarkoitettu erityisesti strukturoimattoman datan säilömiseen. Tallennustilaan pääsee käsiksi käyttämällä sen tarjoamaa suoralinkkiä ilman minkäänlaista todennusta tai käyttämällä autentikointia samaan tapaan kuin Event Hubsissa (ks. kappale 4.1.2). (Macy 2016.)

Blob Storagen kustannukset määräytyvät valitun hinnoitteluluokan sekä varmuuskopiointitavan mukaan. Hinnoittelut ovat esitetty taulukossa 2. Hot -hintaluokka on tarkoitettu jatkuvassa käytössä olevalle tallennustilalle ja Cool -hintaluokka taas sellaiselle tallennustilalle, jonka dataa käytetään harvemmin. Hintaluokkaa pystyy vaihtamaan milloin tahansa. (Azure Storage Pricing n.d.)

Taulukko 2. Blob Storagen hinnoittelu (Azure Storage Pricing n.d.)

Varmuuskopiointi	**LRS**		**GRS**		**RA -GRS**	
Hintaluokka	**Cool**	**Hot**	**Cool**	**Hot**	**Cool**	**Hot**
100 TB / kk	0,0084 €	0,0202 €	0,0169 €	0,0405 €	0,0211 €	0,0514 €
+ 900 TB / kk	0,0084 €	0,0196 €	0,0169 €	0,039 €	0,0211 €	0,0497 €
+ 4000 TB / kk	0,0084 €	0,0188 €	0,0169 €	0,0376 €	0,0211 €	0,0478 €
Yli 5000 TB / kk	Hintatiedot kysymällä					

Varmuuskopiointitapoja on valittavissa kolmea erilaista: Locally Reduntant Storage (LRS) on nimensä mukaisesti paikallinen, yhden konesalin sisällä tehty varmuuskopio. Geographically Reduntant Storage (GRS) varmuuskopioi tiedot kahteen konesaliin. Read Access GRS (RA-GRS) on muuten sama kuin aikaisempi, mutta toisessa konesalissa olevia tietoja päästään lukemaan. (Azure Storage Pricing n.d.)

6 Microsoft Azure Stream Analytics

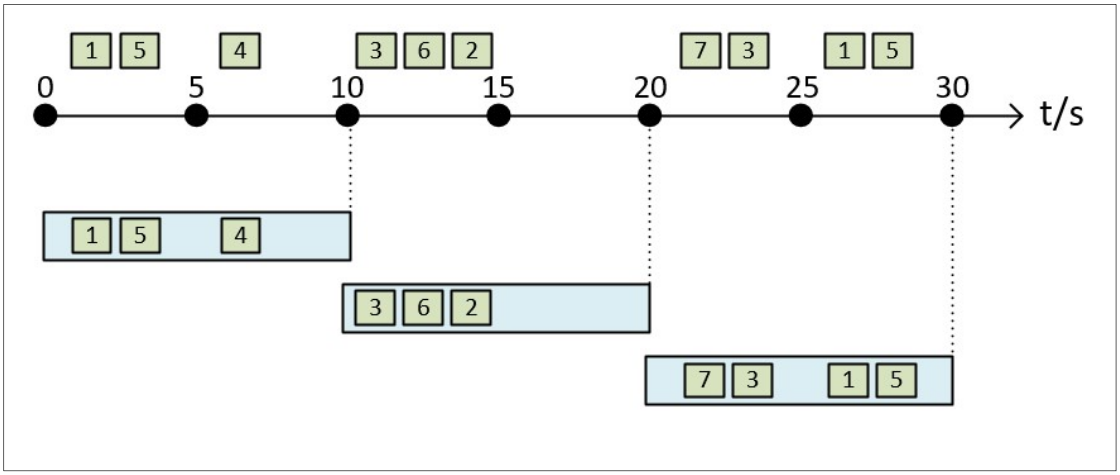
6.1 Yleistä

Azure -pilvipalvelusta löytyy kaksi erilaista nopeuskerroksen analysointityökalua. Toinen niistä on Azure Stream Analytics (ASA) ja toinen Azure Machine Learning (AML), joka käydään tarkemmin läpi kappaleessa 4.4. Stream Analyticsin ideana on yksinkertaisesti suorittaa sisään syötettävään datavirtaan ennalta määritettyjä kyselyitä ja antaa niille mahdollisimman reaaliaikaisia tuloksia. (Feddersen 2015, 21.)

ASA:lle voidaan syöttää reaaliaikaista datavirtaa suoraan Event Hubilta tai käyttää valmiiksi käytössä olevaa dataa Blob -tallennustilasta. Jos dataa syötetään Blob -tallennustilasta, täytyy tapahtumista löytyä aikaleima-kenttä, jotta dataan voidaan suorittaa aikaan perustuvia funktioita. Dataa voidaan syöttää myös molemmista em. lähteistä samanaikaisesti ja liittää tarvittaessa yhteen.

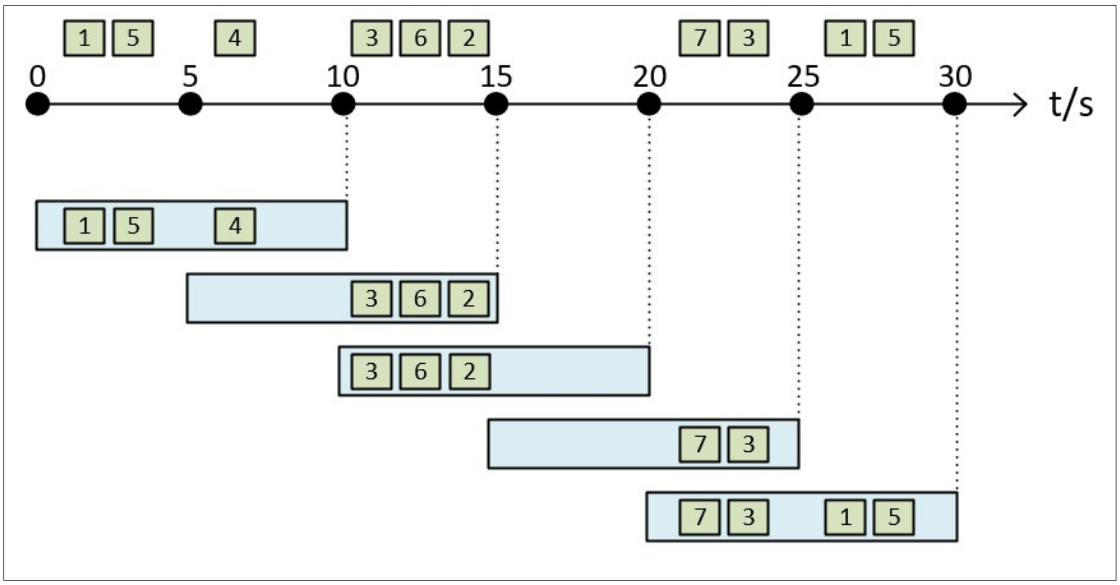
6.2 Kyselykieli

ASA:n kyselykielenä käytetään Structured Query Language (SQL) -tapaista kieltä, jolla voidaan esimerkiksi yhdistää kahden saapuvan datavirran tiedot käyttäen JOIN-lauseketta. Kyselykielestä löytyy myös lisättyjä ikkunointi-funktioita (Feddersen 2015, 21). Ikkunoinnilla (eng. Windowing) voidaan analysoida kaikkia sisään saapuvia tapahtumia tietyn aikaikkunan sisällä. Azuren kyselykielestä löytyy kolme erilaista ikkunointifunktiota (ks. kuviot 3-5). Kaikki kolme funktiota antavat tulokseksi yhden tapahtuman käyttäen Group By-lauseketta, joka summaa aikaikkunan sisällä olevien tapahtumien tiedot yhdeksi riviksi. (Stokes 2016.)



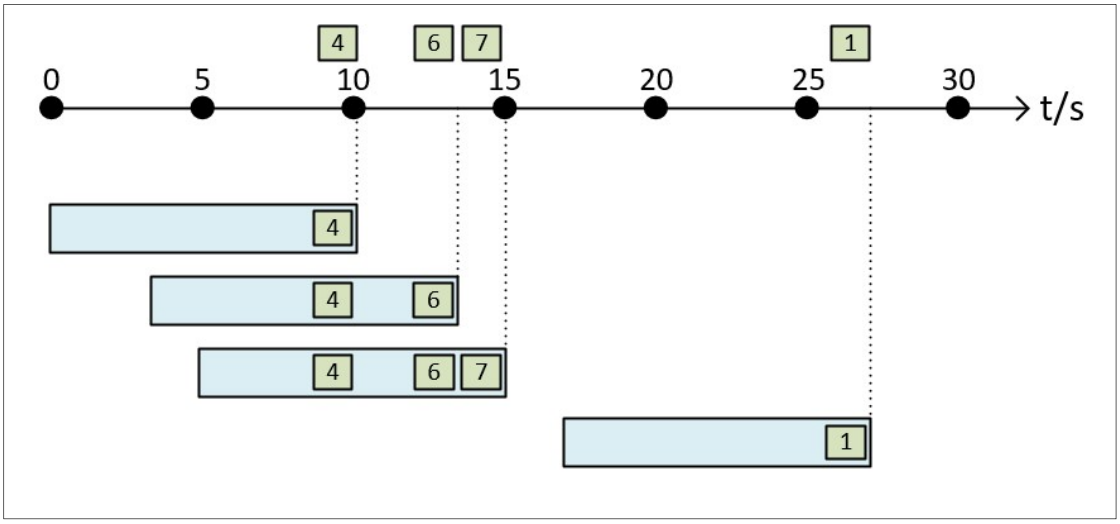
Kuvio 3. Tumbling Window (alkup. kuvio ks. Stokes 2016.)

Tumbling -ikkunalla voidaan jaotella tapahtumat toisistaan täysin erillisiin ikkunoihin ja yksittäistä ikkunaa toistetaan jatkuvasti. Kuvion 3 esimerkissä laskettaisiin kaikki tapahtumat aina 10 sekunnin välein. (Stokes 2016.)



Kuvio 4. Hopping Window (alkup. kuvio ks. Stokes 2016.)

Hopping -ikkunalla taas voidaan jaotella tapahtumia myös päällekkäisiin ikkunoihin, jolloin yksi tapahtuma voi kuulua useampaan eri ikkunaan. Kuvion 4 esimerkissä laskettaisiin taas kaikki 10 sekunnin aikavälillä olevat tapahtumat, mutta tulos päivitettäisiin viiden sekunnin välein. (Stokes 2016.)



Kuvio 5. Sliding Window (alkup. kuvio ks. Stokes 2016.)

Sliding -ikkunan erona edellisiin on, että se antaa tuloksia ainoastaan silloin, kun tapahtumia saapuu aikaikkunaan tai poistuu siitä. Tyhjiä ikkunoita ei siis pääse syntymään. Kuvion 5 esimerkissä laskettaisiin kaikki mahdolliset tapahtumat, jotka mahtuvat samaan 10 sekunnin aikaikkunaan missä kohtaa aikajanaa tahansa. (Stokes 2016.)

6.3 Hinnoittelu

Stream Analytics hinnoitellaan suoraan sen prosessoiman datamäärän, sekä siihen tarvittavan tehokkuuden mukaisesti. Stream Analytics -työn prosessoiman datan hinta on 0,0008 € / GB. Hintojen tarkastelussa on hyvä ottaa huomioon, että mikäli Stream Analytics -kysely käyttää samaa sisään tulevaa datavirtaa useamman kerran, tai sillä on useampi ulostulo, niin myös hinta moninkertaistuu. Hinnat voivat vaihdella riippuen valitusta maantieteellisestä alueesta. Tässä kappaleessa mainituissa hinnoissa käytetään Pohjois-Euroopan hintoja. (Stream Analytics Pricing n.d.)

Työn tehokkuutta voidaan lisätä ostamalla Streaming unit -yksikköjä. Yhdellä yksilöllä saadaan lisää muisti- ja prosessorikapasiteettia sekä 1 MB/s dataliikennettä. Yksikön käyttö maksaa 0,0261 euroa tunnissa ja lopullinen hinta lasketaan tunneittain sen mukaan, kuinka monta yksikköä on ollut enimmillään kerrallaan käytössä. Yksikköjä voidaan oletuksena ostaa maksimissaan 50 kappaletta, mutta raja voidaan ylittää ottamalla yhteyttä Azure -tukeen. (Stream Analytics Pricing n.d.)

7 Microsoft Azure Machine Learning

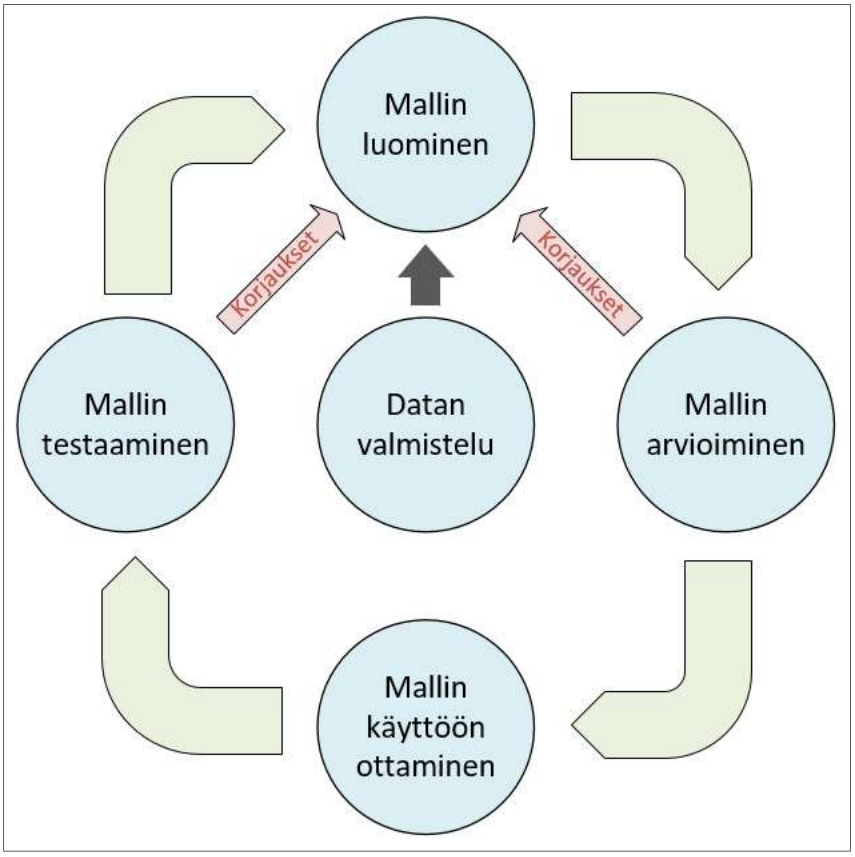
7.1 Yleistä

Machine Learning on tekniikka, jolla pyritään tekemään ennustuksia olemassa olevasta historiadatasta käyttäen erilaisilla algoritmeilla opetettuja analysointimalleja. Tarkemmin sanottuna historiadatasta pyritään puristamaan ulos kaikki oleelliset tiedot ja algoritmeja käyttäen tunnistamaan sekä oppimaan tietojen riippuvuudet toisistaan. Näitä tietoja käyttäen uudesta analysoitavasta datasta voidaan tehdä oletamuksia ja ennustuksia. Uutta dataa siis analysoidaan ja vertaillaan perustuen jo aikaisemmin analysoidusta historiadatasta opittuun tietoon. (Barnes 2015, 13-14.)

Azure Machine Learning on nimensä mukaisesti Azuren tarjoama Machine Learning -ratkaisu, jonka käyttöliittymänä toimii Azure ML Studio. Työkalun ideana on pystyä nopeasti luomaan erilaisia ML -kokeita, joilla ennustava analysointimalli opetetaan. Kokeen pystyy luomaan suoraan puhtaalta pöydältä tai soveltamalla työkaluun suoraan ladattavissa olevia kokeiden mallipohjia, joita on luotu erilaisiin käyttökohteisiin. (Barnes 2015, 25.)

7.2 Azure ML -mallin luominen

Machine Learning -oppiminen koostuu erilaisista vaiheista, jotka toistettaessa kehittävät jatkuvasti analysoinnin tarkkuutta ja tehokkuutta (ks. kuvio 6). Ensimmäisenä pitää tietenkin olla historiadataa, jolla rakennettava analysointimalli voidaan opettaa tunnistamaan oleelliset ominaisuudet uudesta, myöhemmin analysoitavasta datasta. Kun malli on luotu, sen tarkkuus testataan historiadatalla ja tuloksiin perustuen malliin tehdään muutoksia. Muutoksien jälkeen toimivuutta testataan uudestaan ja testauksen tuloksiin perustuen mallia pyritään taas kehittämään tarkemmaksi oppimalla edellisistä virheistä. Tätä sykliä toistetaan niin kauan, kunnes haluttu tarkkuus on saavutettu. (Barnes 2015, 26-27.)



Kuvio 6. Machine Learning - Oppimisen vaiheet (alkup. kuvio ks. Barnes 2015, 26.)

Uuden ML-mallin luominen lähtee käytännössä siis liikkeelle siitä, että jo olemassa oleva historiadata tuodaan malliin ja se valmistellaan käyttöä varten. Arvoja voidaan esimerkiksi muotoilla helpommin käsiteltävään muotoon, tai vaikka poistaa kokonaan analysoinnin kannalta tarpeettomia arvoja. (Barnes 2015, 51.)

Kun data on saatu siistittyä, data jaetaan opetusdataan ja testidataan. Jakamisen suhde voidaan määritellä itse, mutta kuitenkin siten, että suurin osa kuuluu opetusdataan. Ideana on, että malli opetetaan opetusdatalla ennustamaan jotain tiettyä lopputulosta tai tunnistamaan datan rakenteita käyttäen jotain kyseiseen käyttökohteeseen sopivaa ML-algoritmia. (Barnes 2015, 60.)

Mallin luomisen jälkeen sitä koeajetaan testidatalla, joka aikaisemmassa vaiheessa jaettiin käytössä olevasta historiadatasta. Testauksen tulosten tarkkuutta voidaan helposti mitata, koska testidata on myös historiadataa, jolloin kaikki lopputulokset ovat jo ennalta tiedossa. (Barnes 2015, 60,64.)

Opetetun mallin tarkkuuden arviointi tapahtuu siis peilaamalla testidataa mallin tuottamiin tuloksiin. Mikäli huomataan, että tarkkuus ei ole riittävä, niin mallia kehitetään esimerkiksi vaihtamalla käytettävää algoritmia tai siihen syötettäviä tietoja, jonka jälkeen testaus ja arviointi toistetaan. (Barnes 2015, 68.)

Kun mallille on saavutettu haluttu tarkkuus, se tallennetaan ja otetaan käyttöön julkaisemalla se web-palveluna. Lisäksi malli tallentuu ML Studioon valmiiksi malli-moduuliksi, jota voidaan käyttää tulevaisuudessa suoraan moduulivalikoimasta. (Barnes 2015, 71.)

7.3 Hinnoittelu

Machine Learning Studiosta on saatavilla ilmainen, lähinnä kuluttaja- ja testauskäyttöön tarkoitettu versio sekä maksullinen, tuotantokäyttöön suunnattu versio. Taulukossa 3 on esitetty näiden kahden versioiden ominaisuuksia. Maksullisen Standard -version hinta on 8,4246 euroa kuukaudessa ja lisäksi kokeen suorittamiseen kulutettu aika veloitetaan tunneittain. (Machine Learning Pricing n.d.)

Taulukko 3. Machine Learning hinnoittelu (Machine Learning Pricing n.d.)

Hintaluokka	**FREE**	**STANDARD**
Kiinteä hinta	Ilmainen	8,4246 € / tili / kk + 0,8433 / käyttötunti
Azure -tili vaadittu	Ei	Kyllä
Max. määrä moduuleita / koe	100	Rajoittamaton
Max. kokeen suorittamisen aika	1 tunti / koe	7 päivää / koe
Max. Tallennustila	10 GB	Rajoittamaton
Suoritusteho	Yksittäinen suorittava yksikkö	Useampia suorittavia yksikköjä
Web-palvelu käytössä	Ei	Kyllä
Palvelutasosopimus	Ei	Kyllä
Ulkopuolisen SQL-palvelimen käyttö	Ei	Kyllä

Ilmaisessa versiossa luotua ML-ratkaisua ei voida lainkaan julkaista web-palveluksi, joten sitä ei voida kutsua mistään ML Studion ulkopuolelta. Maksullisessa versiossa kyseinen ominaisuus on käytössä ja sen käyttäminen hinnoitellaan vielä erikseen taulukon 4 hintaluokkien mukaisesti. (Machine Learning Pricing n.d.)

Taulukko 4. Machine Learning web-palvelun hinnoittelu (Machine Learning Pricing n.d.)

Hintaluokka	**DEV/TEST**	**STANDARD S1**	**STANDARD S2**	**STANDARD S3**
Hinta / kk	0 €	84,33 €	843,30 €	8433 €
Max. tapahtumat (kpl)	1000	100 000	2 000 000	50 000 000
Ylimenevät tapahtumat	Ei saatavilla	0,4217 € / 1000 kpl	0,2108 € / 1000 kpl	0,0843 € / 1000 kpl
Max. käyttötunnit	2	25	500	12 500
Ylimenevät käyttötunnit	Ei saatavilla	1,6866 € / h	1,265 € / h	0,8433 € / h
Max. web-palveluiden määrä	2	10	100	500

7.4 Algoritmityypit

7.4.1 Yleistä

Machine Learning -algoritmit voidaan luokitella joko hallitun tai hallitsemattoman oppimisen algoritmiksi niiden käyttötavan mukaan. Hallitun oppimisen algoritmeja käytetään sellaisen historiadatan analysoimiseen, jonka lopputulokset tiedetään jo ennalta. Hallitsemattomassa oppimisessa datasta pyritään tunnistamaan eri tietojen riippuvuuksia ja tekemään sen perusteella johtopäätöksiä. (Barnes 2015, 28,33.)

Otetaan esimerkkinä tilanne, jossa yritettäisiin ennustaa, ylittääkö yksittäisten henkilöiden tulot vuodessa yli 50 000 € ja ennustus tehtäisiin perustuen henkilön muihin tietoihin, kuten ikään ja asemaan perustuen. Jos mallin luomiseen käytettävä historiadata olisi taulukon 5 mukaista, voitaisiin käyttää hallitun oppimisen algoritmeja.

Esimerkiksi kyseisen taulukon datasta voitaisiin tehdä suoraan olettamus, että 36-46-vuotias johtaja-asemassa olevan henkilön tulot ylittävät 50 000 € vuodessa. (Barnes 2015, 28,30.)

Taulukko 5. Esimerkki historiadatasta

Nimi	**Ikä**	**Asema**	**Tulot yli/alle 50 000 €**
Henkilö 1	46	Johtaja	Yli
Henkilö 2	24	Työntekijä	Alle
Henkilö 3	35	Johtaja	Yli
Henkilö 4	31	Työntekijä	Alle
Henkilö 5	40	Työntekijä	Alle

Hallitsemattomassa oppimisessa tavoitteet ovat lähtökohtaisesti erilaiset. Datasta pyritään yhden tietyn arvon tai lopputuloksen sijaan löytämään erilaisia riippuvuuksia ja rakenteita tietämättä mitään ennalta. Tällainen riippuvuus voisi olla esimerkiksi, että iso osa asiakkaista, jotka ostivat tuotteen X, ostivat myös tuotteen Y. (Barnes 2015, 33,34.)

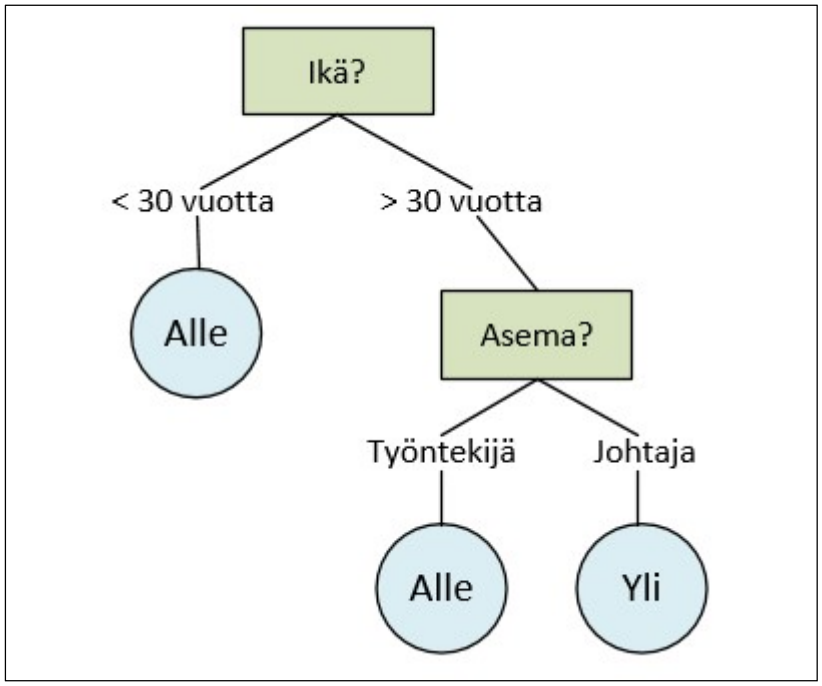
Azure ML Studiossa on valittavissa useita erilaisia hallittuun oppimiseen tarkoitettuja valmiita algoritmimoduuleita, jotka ovat luokiteltu vielä kolmeen eri luokkaan: Regression, Classification ja Anomaly Detection. Hallitsemattomaan oppimiseen tarkoitetut algoritmit löytyvät Clustering -luokittelun alta. (Barnes 2015, 27)

7.4.2 Classification

Classification, eli luokittelualgoritmit ennustavat ns. epäjatkuvia arvoja, eli diskreettejä, jotka voivat saada vain jonkun tietyn arvon. Tässä kohtaa voidaan palata aikaisemmin esitettyyn esimerkkiin, jossa taulukon 5 arvoista tehtiin päätelmä, että henkilön tulot voivat olla yli tai alle 50 000 € vuodessa. Esimerkissä on vain kaksi mahdollista lopputulosta: Yli tai Alle. Mahdollisia tuloksia voisi olla myös useampia, mutta kaikki vaihtoehdot pitää olla ennalta tiedossa. (Ericson 2016.)

Edellä mainitun esimerkin lopputuloksen saamiseksi voitaisiin käyttää yksinkertaista päätöspuu -menetelmää. Käytännössä menetelmä toimii siten, että data luokitellaan

eri tulosjoukkoihin (Yli/Alle) käyttäen yksinkertaisia kyllä/ei -tyyppisiä kysymyksiä perustuen käytettävän datan arvoihin. Kuviossa 7 on esitetty, miten kyseinen menetelmä toimisi taulukon 5 arvoilla. (Hackeling 2014, 97-98.)

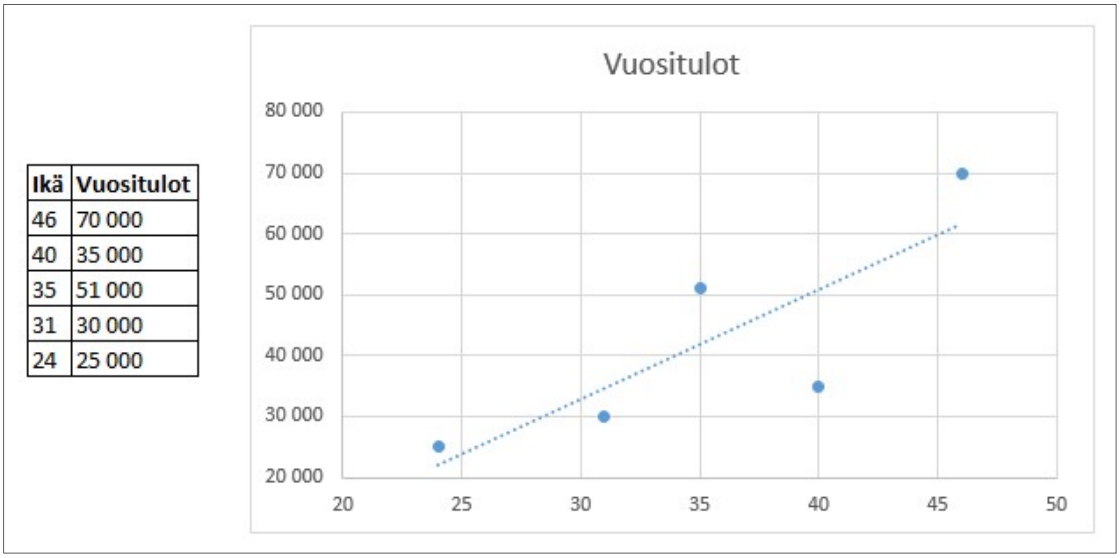


Kuvio 7. Decision Tree -esimerkki

7.4.3 Regression

Regressioalgoritmit toimivat hyvin samalla tavalla, mutta ne ennustavat jatkuvia arvoja, jotka voivat saada minkä tahansa arvon tietyltä väliltä. Verrattuna aikaisempaan esimerkkiin, regressioalgoritmeilla pyrittäisiin ennustamaan yksinkertaisen yli/alle tuloksen sijaan tarkkaa summaa henkilön vuosituloista. (Ericson 2016.)

Regressioanalyysiä voidaan myös havainnollistaa henkilöiden vuositulojen (ks. taulukko 5) ennustamisella. Tarkoituksena on löytää henkilöiden iän ja vuositulojen välinen riippuvuus ja ennustaa tulevia arvoja käyttäen historiadatan perusteella lasketun lineaarisen regressiosuoran antamia arvoja. Regressiosuora voidaan laskea kaavalla: $y = a + bx$, jossa a on vakiotermi ja b kaltevuuskerroin. Kuviossa 8 on esitetty valmiiksi laskettu suora vasemmalla näkyville arvoille ja pyritty havainnollistamaan regressioanalyysin toimintaa. Esimerkiksi 45-vuotiaan henkilön vuositulot voitaisiin ennustaa olevan suoran perusteella noin 60 000 € vuodessa ja 27-vuotiaan henkilön tulot olisivat taas noin 30 000 € vuodessa. (Hackeling 2014, 24.)



Kuvio 8. Lineaarinen regressio -esimerkki

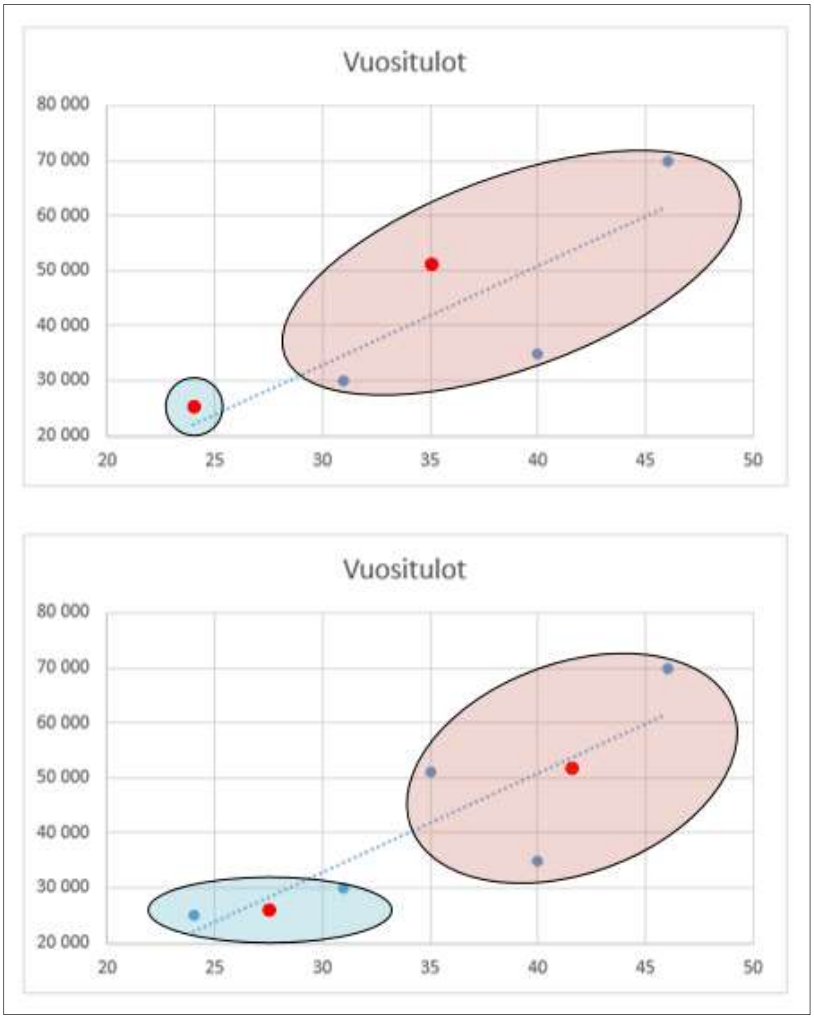
7.4.4 Anomaly Detection

Anomaly Detection -algoritmien tarkoituksena on nimensä mukaisesti tunnistaa poikkeavuuksia datasta. Tätä varten tarvitaan historiadataa, josta voidaan ensin tunnistaa, mikä on normaalia ja kyseiselle datalle ominaista. Esimerkkinä voitaisiin käyttää henkilöiden luottokorttien käyttödataa, josta ilmenisi, kuinka usein korttia käytetään, missä sijainnissa ja kuinka paljon kerrallaan. Lisäksi historiadataan voitaisiin liittää luokiteltua tietoa, millaisissa liikkeissä korttia on käytetty. Näitä arvoja käyttämällä voitaisiin tutkia, mikä on yksittäiselle henkilölle normaalia luottokortin käyttämistä ja esimerkiksi toisessa maassa tehty epätavallinen luottokorttiostos voisi nostattaa epäilyjä. (Ericson 2016.)

7.4.5 Clustering

Clustering -algoritmeilla datasta pyritään havaitsemaan rakenteita ja riippuvuuksia ilman mitään ennalta tiedettyjä lopputuloksia. Käytännössä se toimii muodostamalla ryhmiä sellaisista tiedoista, jotka sisältävät paljon samoja piirteitä. Otetaan esimerkkinä tilanne, jossa pankki arvioi henkilön luottokelpoisuutta lainan myöntämiselle. Perusteena arvioinnille voitaisiin käyttää esimerkiksi henkilöiden tulotasoa, ikää, ammattia ja anottavan lainansumman määrää. Näitä tietoja käyttämällä samantapaisia ominaisuuksia omaavat henkilöt voitaisiin ryhmitellä erilaisiin luottotason määritteleviin ryhmiin ja lopullinen päätös lainan myöntämiselle tehdä ryhmittelyn pohjalta. (Lazzeri 2016.)

Datan klusterointi, eli ryhmittely voidaan suorittaa käyttämällä esimerkiksi K-klusterialgoritmia. Algoritmille täytyy ensin määrittää arvo *K,* joka määrittää muodostettavien klusterien, eli ryhmien enimmäismäärän. Kuviossa 9 on käytetty taas taulukon 5 dataa ja siinä *K*:n arvoksi on määritetty *2.* Ensimmäisenä algoritmi valitsee tiedoista kaksi satunnaista keskipistettä, joiden mukaan kaikki muut tiedot määritellään ryhmiin riippuen kumman ryhmän keskipistettä lähempänä tieto ”sijaitsee”. Ryhmittelyn jälkeen keskipiste siirretään uuden muodostetun ryhmän keskelle, jonka jälkeen ryhmittely suoritetaan uudelleen. Samaa toistetaan niin kauan, kunnes ryhmien kokoonpanot eivät enää muutu. Kuvion 9 esimerkissä ryhmittely vaati vain kaksi ryhmittelykertaa. (Hackeling 2014, 118-121.)



Kuvio 9. Klusterointi-esimerkki

8 Työkalujen testaus

8.1 Testiosuuden kuvaus ja päämäärä

Testausosuudessa päämääränä oli tutkia Microsoft Azure Machine Learning, Stream Analytics ja Event Hubs -palveluita sekä havainnollistaa niiden toimintaperiaatteita ja käyttöönottoa käytännössä. Tämä päätettiin toteuttaa valitsemalla jokin kaikille palveluille sopiva käyttökohde, jolloin Machine Learning ja Stream Analytics -palveluita voitaisiin vertailla analysoimalla samaa reaaliaikaista testidataa.

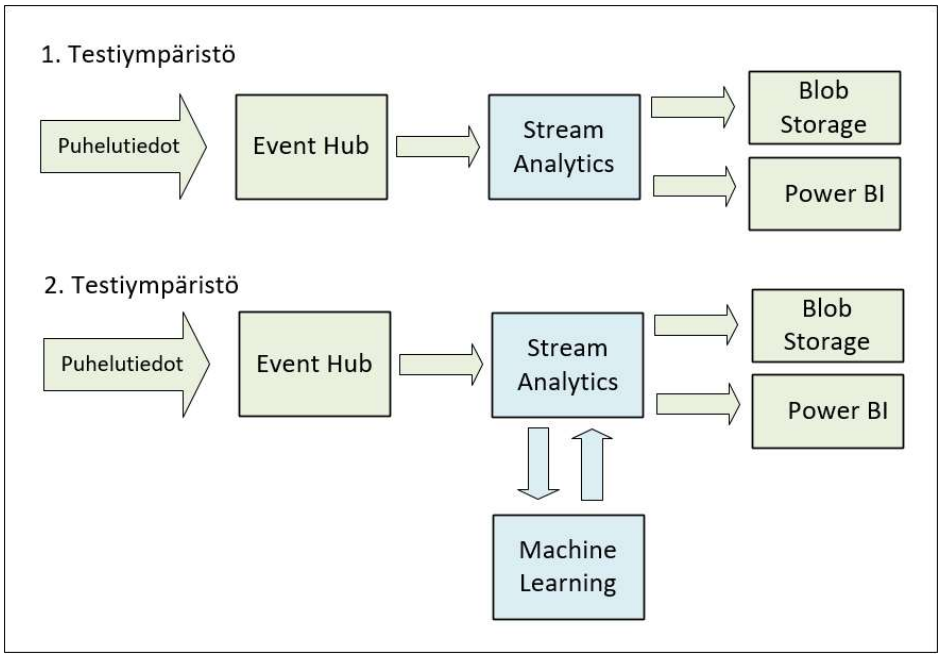
Mahdollisia käyttökohteita kartoitettaessa huomattiin, että ns. ”fraud detection”, eli petosten tai huijausten tunnistaminen erilaisissa tilanteissa osoittautui yhdeksi monipuolisimmista toteutuksista Big Data -ratkaisuilla. Yleisin esimerkki tästä käyttökohteesta lienee maksukorttipetosten tunnistaminen. Esimerkiksi PayPal -maksujenvälitysjärjestelmä käyttää Machine Learning -tekniikkaa epätavallisten maksutapahtumien tunnistamisessa tutkimalla käyttäjien maksukäyttäytymistä (Hamilton 2016).

Fraud detection -toteutus Machine Learning -ympäristössä perustuu siihen, että kaikki uudet tutkittavat tapahtumat analysoidaan peilaamalla niitä aikaisempiin huijaustapauksiin. Tämä tarkoittaa siis sitä, että käytössä pitää olla riittävä määrä historiadataa, jolla Machine Learning -malli voidaan opettaa. (Hamilton 2016.)

Käyttökohteeksi valittiin lopulta SIM-korttihuijauksien tunnistaminen. Testidatan generoimiseen löytyi valmis työkalu, jolla pystyttiin syöttämään puhelutietoja sisältävää dataa suoraan Event Hubille. Tarkoituksena oli tunnistaa datasta mahdolliset SIM-korttien väärinkäytöt. Käytettävällä datageneraattorilla pystyttiin suoraan säätelemään, kuinka suuri osa puhelutiedoista oli prosentuaalisesti huijauspuheluita.

Koska tarkoitus oli testata kahta erilaista työkalujen kokoonpanoa, luotiin myös kaksi erilaista testiasetelmaa. Molemmat testiasetelmat ovat esitetty kuviossa 10. Ensimmäisessä testiympäristössä mahdolliset huijaukset pyrittiin tunnistamaan sisään virtaavista puhelutiedoista reaaliaikaisesti. Ensin datavirta syötettiin Event Hubsin kautta Stream Analytics -palveluun, jossa huijaukset tunnistettiin käyttäen Stream Analyticsin omaa kyselykieltä. Kyselyn tulokset ohjattiin ennalta luotuun Blob -tallen-

nustilaan tarkastelua varten. Tulokset olisi voinut ohjata vaihtoehtoisesti myös esimerkiksi Power BI -työkalulle visualisoitavaksi, mutta tämän toteuttamisen ei nähty tuovan juurikaan lisäarvoa testaukselle. Tämä testaus luotiin Microsoftin Azure -dokumentaation esimerkkien pohjalta.



Kuvio 10. Testiympäristöjen rakenne

Toisessa, huomattavasti monimutkaisemmassa ympäristössä pyrittiin myös tunnistamaan huijauspuheluita, mutta tällä kertaa puhelutietojen analysoimiseen käytettiin Machine Learning -palvelua. Tämä tapahtui siten, että ensin luotiin historiadataa, jolla opetettiin uusi Machine Learning -analysointimalli, jonka jälkeen puhelutiedot voitiin syöttää Stream Analyticsiltä suoraan aikaisemmin luodun ML -ratkaisun analysoitavaksi. Lopuksi Machine Learning palautti Stream Analyticsille tiedon, kuinka todennäköisesti puhelutieto oli huijausyritys. Tällä testauksella päästiin siis testaamaan kaikkien kolmen palvelun toimintaa yhdessä.

ML -ratkaisu rakennettiin soveltaen Microsoftin Cortana Intelligence Gallerysta löytyviä Online Fraud Detection -mallipohjia, jotka sisältävät viisi eri vaihetta. Ensimmäiset kolme sisältävät historiadatan leimaamisen sekä valmistelun. Kaksi viimeistä taas keskittyy ML-mallin opettamiseen, arviointiin ja käyttöönottoon. ML-ratkaisun kutsuminen Stream Analyticsissä sekä lopullinen kysely luotiin Microsoftin Stream Analytics -dokumentaation pohjalta.

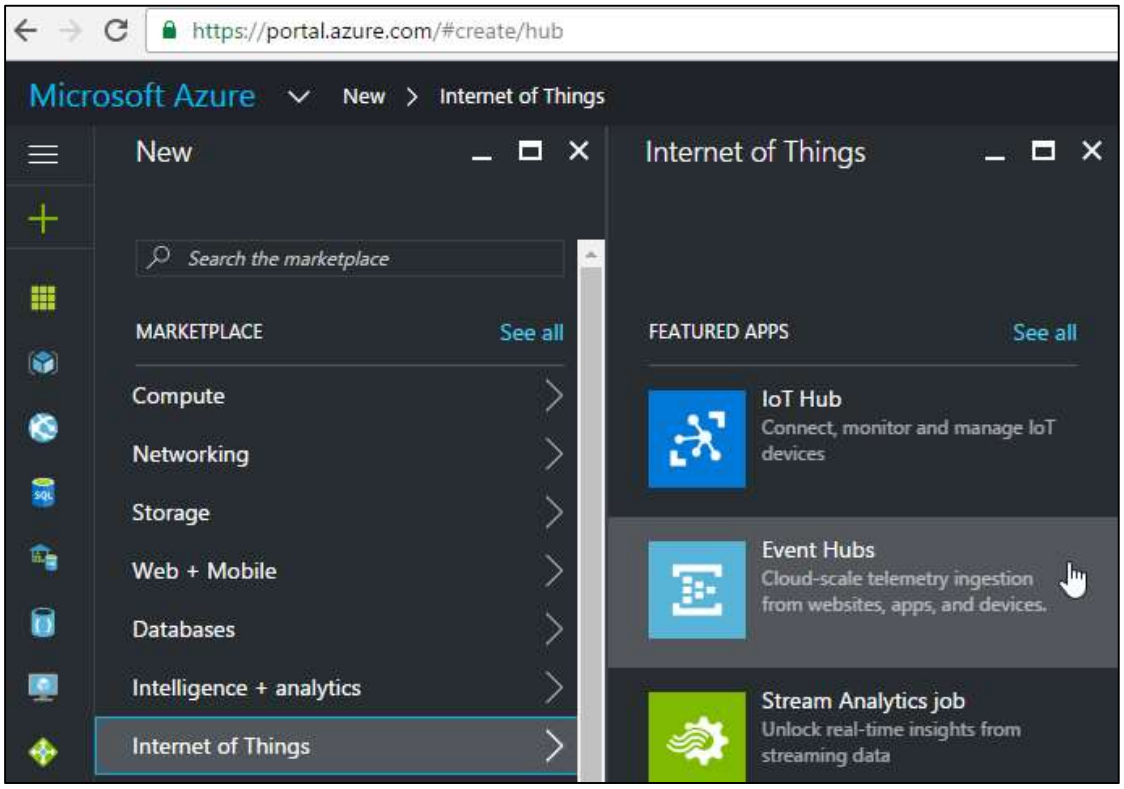
Erilaisia mallipohjia on luotu muihinkin yleisimpiin ML -käyttökohteisiin ja niitä voidaan helposti käyttää runkona omalle ML -ratkaisulle. Mallipohjat ovat vapaasti ladattavissa suoraan ML Studioon ja ne ovat täysin sovellettavissa omien käyttötarkoituksien mukaisesti. (Azure ML Team 2016.)

Näiden testiympäristöjen luominen on selostettu seuraavissa kappaleissa ja kaikki käytetyt kyselyt ja koodit löytyvät liitteinä tekstimuotoisena. Ennen testausta luotiin käyttäjätilit Azure-portaaliin osoitteessa https://portal.azure.com/ sekä Azure ML Studioon osoitteessa https://studio.azureml.net/. Näihin palveluihin oli mahdollista kirjautua omalla Microsoft-tilillä ja ottaa käyttöön kuukauden ilmaiset kokeilujaksot. Azure-portaalissa kuukauden ilmaisen lisenssin käyttökattona oli 200 euroa, mikä riitti testaukseen helposti datamäärien ollessa pieniä.

8.2 Testiasetelma 1.

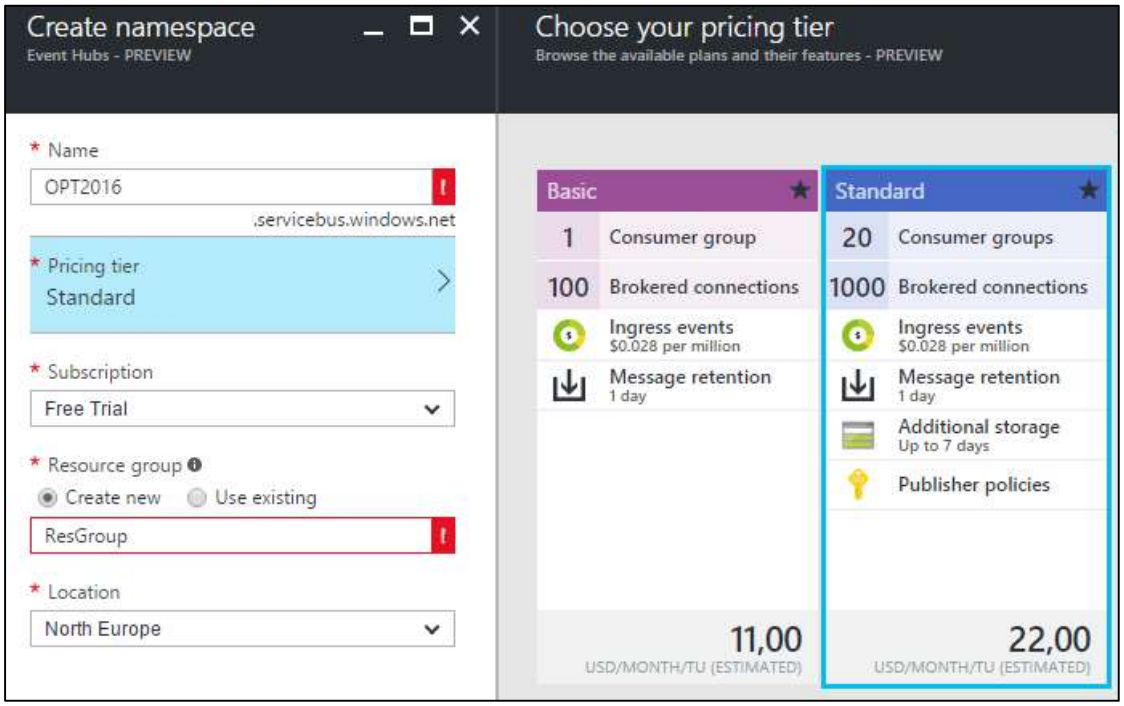
8.2.1 Event Hubsin käyttöönotto

Azure-portaalin päänäkymästä näkee suoraan kaikki käyttöönotetut palvelut. Uusi palvelu voidaan ottaa käyttöön vasemmasta sivupalkista painamalla ”+”-kuvaketta. Ensimmäisenä otettiin käyttöön Event Hubs -palvelu ”Internet Of Things”-kategorian alta (ks. kuvio 11). Toinen vaihtoehto olisi hakea haluttua työkalua suoraan hakupalkista. Päänäkymästä voidaan myös avata listaus kaikissa käytössä olevista työkaluista valitsemalla ”All resources”.



Kuvio 11. Palvelun käyttöönotto Azure-portaalissa

Ensimmäisenä luotiin siis Event Hubs -nimiavaruus, joka on tavallaan kuin juurihakemisto kaikille uusille Event Hub -töille. Event Hubs -palvelua käyttöön ottaessa tulee nimiavaruudelle ensimmäisenä asettaa vapaavalintainen nimi. Tähän laitettiin opinnäytetyötä kuvaavasti ”OPT2016”. Seuraavaksi valittiin hintaluokka, joista oli tätä kirjoitettaessa valittavissa kaksi eri mahdollisuutta (ks. kappale 4.1.5). Kuviossa 12 on vielä esitettynä nämä hintaluokat ja mitä ne sisältävät. Kalliimpi Standard -luokka mahdollistaa 20 eri kuluttajasovelluksen käyttämisen yhdelle Event Hubs -tilaukselle samanaikaisesti. Standard -luokka tuo myös publisher policyt käyttöön, minkä vuoksi se valittiin käytettäväksi hintaluokaksi.

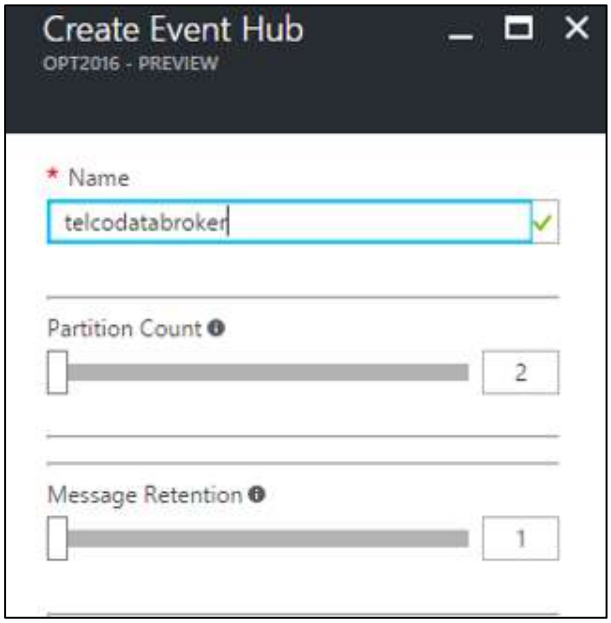


Kuvio 12. Event Hubs -nimiavaruuden luominen

Koska tämä oli ensimmäinen käyttöönotettava Azuren palvelu, luotiin samalla myös uusi ”Resource group”. Kaikki myöhemmin käyttöön otettavat palvelut liitettiin tähän samaan ryhmään. Lisäksi kaikki palvelut lisättiin saman alueen alle, jotta vältyttiin ylimääräisiltä alueiden välisiltä datansiirtokuluilta. Tässä tapauksessa valittiin luonnollisesti Pohjois-Eurooppa. Kun valinnat oli tehty, tarvitsi vain painaa sivun alalaidasta ”create”, ja hetken kuluttua ilmoituspalkkiin tuli teksti onnistuneesta lisäyksestä. Kuviossa 12 näkyvät punaiset huutomerkit johtuvat siitä, että kyseiset nimet ovat jo käytössä. Tämä johtuu siitä, että kuvankaappaus on otettu jälkikäteen.

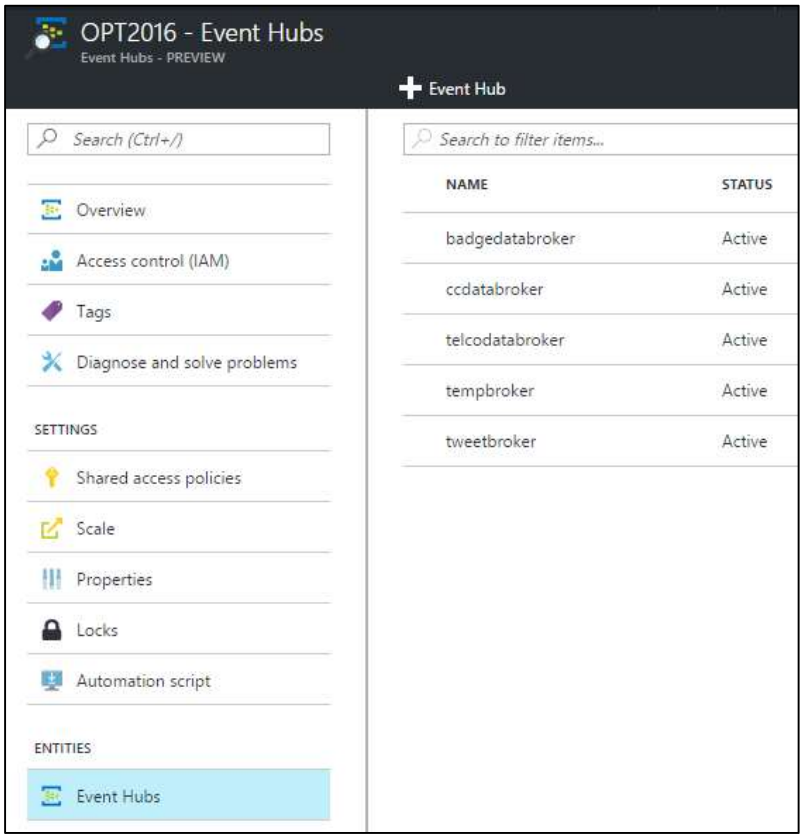
Seuraavaksi luotiin Event Hubs -nimiavaruuteen uusi Event Hub -työ, jolla vastaanotetaan testidataa sitä generoivalta ohjelmalta, sekä välitetään data Stream Analytics -palveluun analysointia varten. Kuviossa 13 on kuvankaappaus uuden Event Hubin luomisesta. Event Hubille asetettiin nimi ja osioiden määrä. Lisäksi määritettiin,

kuinka kauan tapahtumia säilytetään. Testauksessa ei nähty tarpeelliseksi määrittää näihin suurempia arvoja.



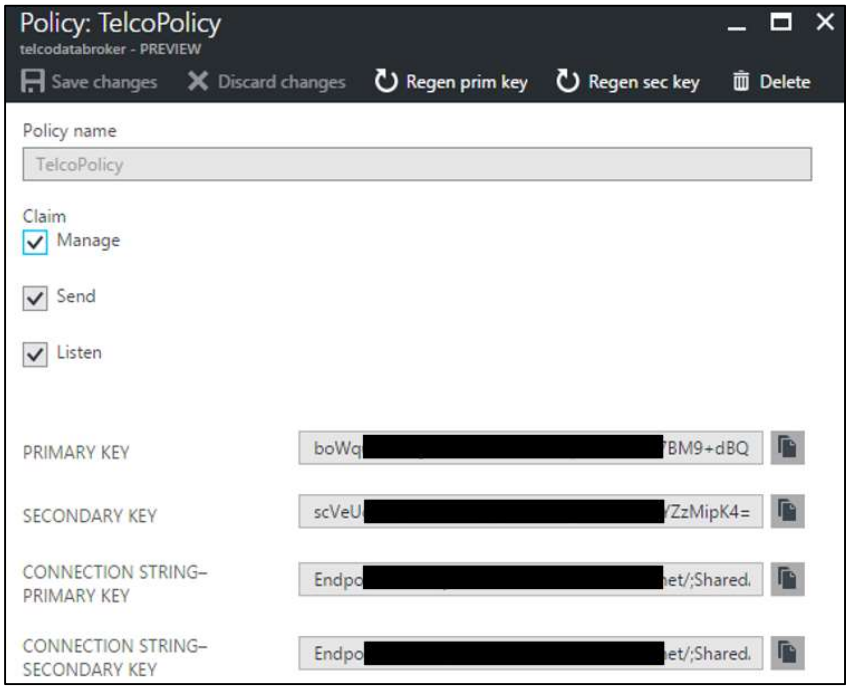
Kuvio 13. Event Hubin luominen.

Event Hubs -päänäkymän saa auki painamalla päävalikosta ”All resources” ja valitsemalla aikaisemmin lisätty OPT2016-nimiavaruus. Kaikki kyseisen nimiavaruuden alle luodut Event Hubit löytyvät vasemmasta palkista ”Event Hubs”-valikon kautta (ks. kuvio 14).



Kuvio 14. Event Hubs -näkymä

Uudelle Event Hubille luotiin myös Shared Acces Policy. Tämä tapahtuu klikkaamalla Event Hubs -näkymästä lisätyn Hubin nimeä ja painamalla auki valikosta Shared Access policies -osion. Event Hubille lisättiin ”TelcoPolicy”-niminen policy kuvion 15 mukaisesti, jonka yhteysavainta käytettiin testidatan lähettämisessä.



Kuvio 15. Event Hub Shared Access Policy

Mikäli Event Hubilla on useampia kuluttajasovelluksia, voidaan sille määrittää samasta näkymästä myös lisää kuluttajaryhmiä. Testausta varten luotiin ”telcogroup”-niminen kuluttajaryhmä Stream Analyticsiä varten.

8.2.2 Testidatan generointi

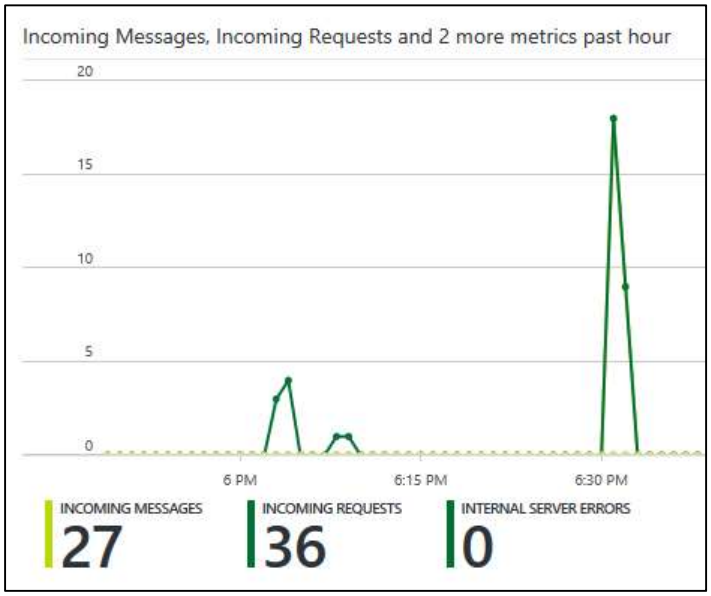
Testidataa generoitiin GitHubista löytyneellä ohjelmalla, jonka latauslinkki oli Microsoftin Event Hub -dokumentaatiossa. Ohjelmaa käytettiin omalta kotikoneelta ja siltä syötettiin puhelutietoja Azuren palveluille jatkokäsittelyä varten. Ohjelman käyttöönotto vaati ainoastaan, että ohjelman asetustiedostoon piti lisätä Event Hubille tehdyn Shared Access Policyn nimi, sekä yhteysavain. Suora linkki ohjelman lataussivulle: https://github.com/sslarson/TelcoDataGenerator

Ohjelmalla oli mahdollista generoida puhelutietoja JSON-muodossa suoraan Event Hubiin. Ohjelmaan piti määrittää, kuinka monta puhelua tunnissa generoidaan, kuinka suuri osa niistä oli prosentuaalisesti huijauspuheluita, sekä kuinka monta tuntia dataa generoitiin kerrallaan. Kuviossa 16 on esitetty esimerkki ohjelman käytöstä. Siinä generoidaan 1000 kpl puheluita kahden tunnin ajan ja puheluista 20 % olivat huijauksia.

```
C:\Users\Mikko\Desktop\telco\TelcoGenerator>telcodatagen.exe 1000 ,2 2
#Sets: 1,#FilesDump: 1,#CDRPerFile: 1000,%CallBack: 0,2, #DurationHours: 2
Time Increment Per Set: 2
20161120 183009
MO,d0,0,UK,012347664,466921200135361,678970412,466923200348594,20161120,183009,2,
MO,d0,2,China,012307142,466922702341485,567833902,466922702346260,20161120,183011
```

Kuvio 16. Testidatan generoiminen

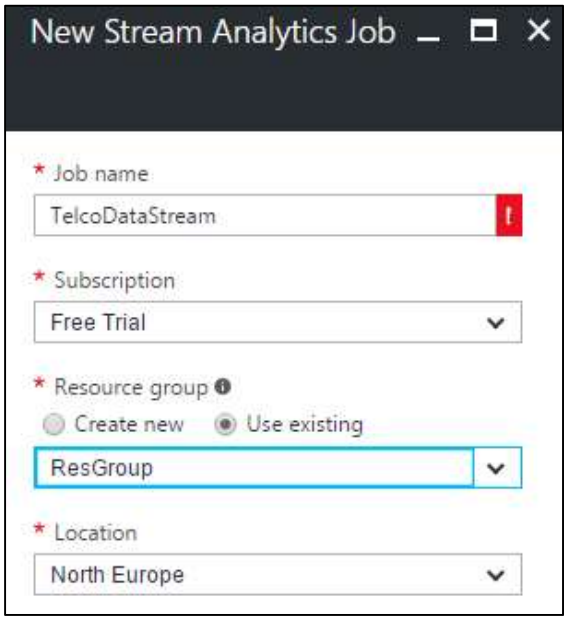
Azure-portaalissa Event Hubin tietosivulla pystyttiin tarkastelemaan Event Hubin lävitse kulkevaa liikennettä sinne automaattisesti piirtyneestä graafista. Kuviosta 17 voidaan huomata, että Event Hubiin oli alkanut virrata dataa noin klo. 18:30. Tämä vastasi myös kuviossa 16 näkyvää datan generoinnin aloittamisen aikaleimaa.



Kuvio 17. Event Hubin liikenteen kuvaaja

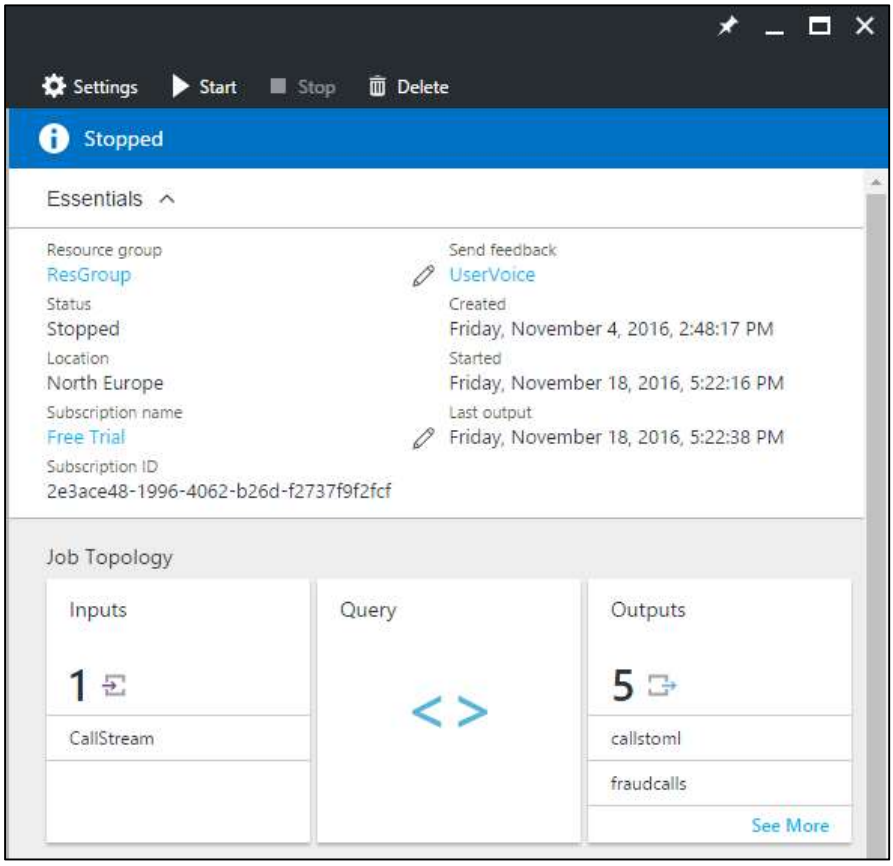
8.2.3 Stream Analyticsin käyttöönotto

Kun testidataa oli saatu onnistuneesti syötettyä Event Hubille, oli seuraavana vaiheena luoda dataa kuluttava Stream Analytics -työ. Tällä kertaa päävalikosta lisättiin siis ”New Streaming Analytics job”. Uuden työn asetukset määritettiin kuvion 18 mukaisesti. Tämä tapahtui siis samaan tapaan kuin uuden Event Hubs -nimiavaruuden luominen tapahtui kappaleessa 5.2.1.



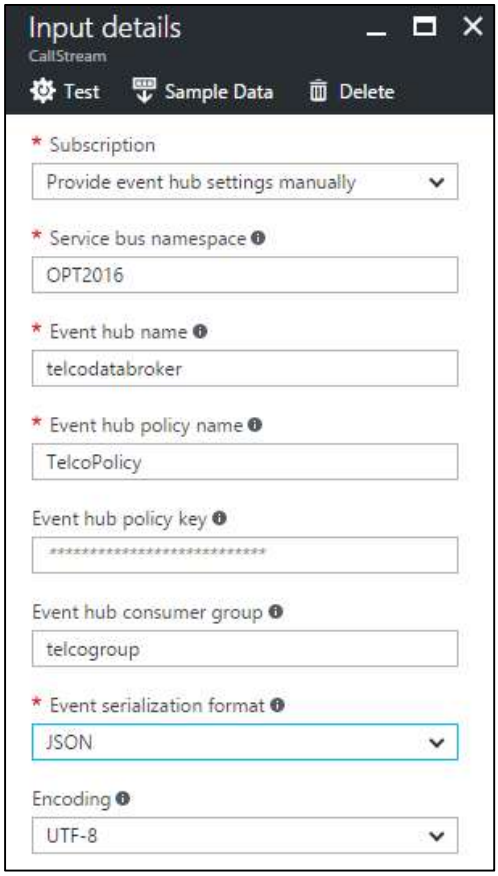
Kuvio 18. Uuden Stream Analytics -työn luominen

Kuviossa 19 on kuvankaappaus Stream Analytics -työn näkymästä. Siinä on listattu työn perustiedot sekä kuvattu työn topologia kolmella laatikolla, joista työlle voitiin määrittää sisään- ja ulostulo sekä määrittää haluttu kysely käyttäen ASA:n kysely-kieltä.



Kuvio 19. Stream Analytics -näkymä

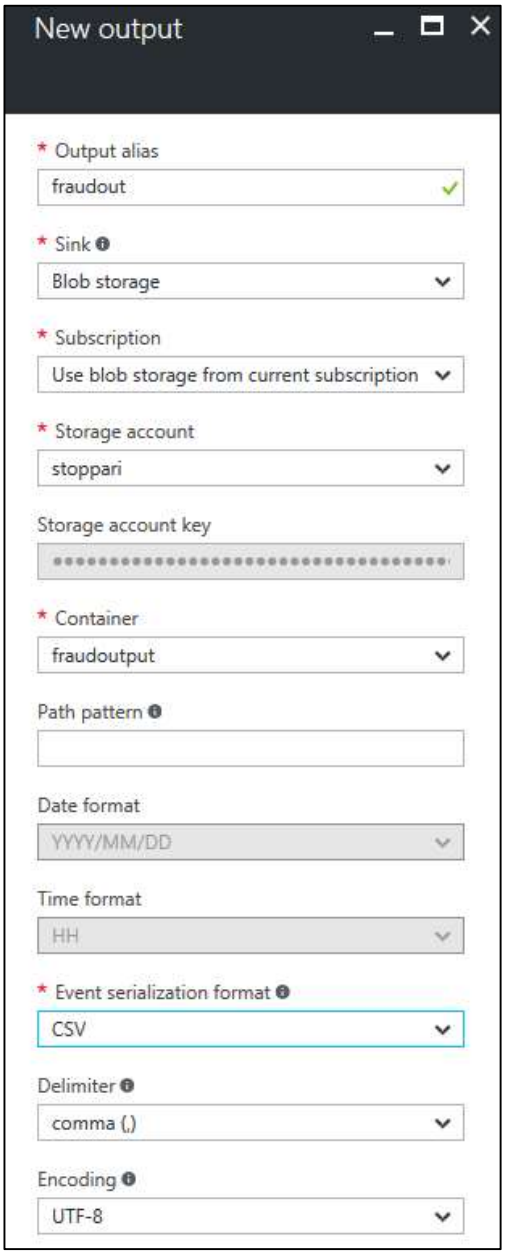
Ensimmäisenä työlle määriteltiin sisääntulo, painamalla ”Inputs”-laatikkoa. Uusi sisääntulo määritettiin kuvion 20 mukaisesti aikaisemmin luodun Event Hubin tiedoilla (ks. kappale 5.2.1). Lisäksi määritettiin, että sisään tuleva data on JSON-muotoista.



Kuvio 20. Stream Analytics -sisääntulo

Sisääntulon lisäksi työlle tarvitsi määritellä luonnollisesti myös ulostulo. Stream Analytics -palvelun prosessoima data piti varastoida jonnekin, jotta tuloksia voitiin tutkia. Tähän tarkoitukseen otettiin käyttöön Blob -tallennustila. Ensin Azure-portaalin kautta luotiin uusi Storage -tili, jonka jälkeen tilin alle lisättiin uusi Blob -tallennustila nimeltä ”fraudoutput”. Tallennustilan luominen ei vaatinut erityisempiä asetuksia.

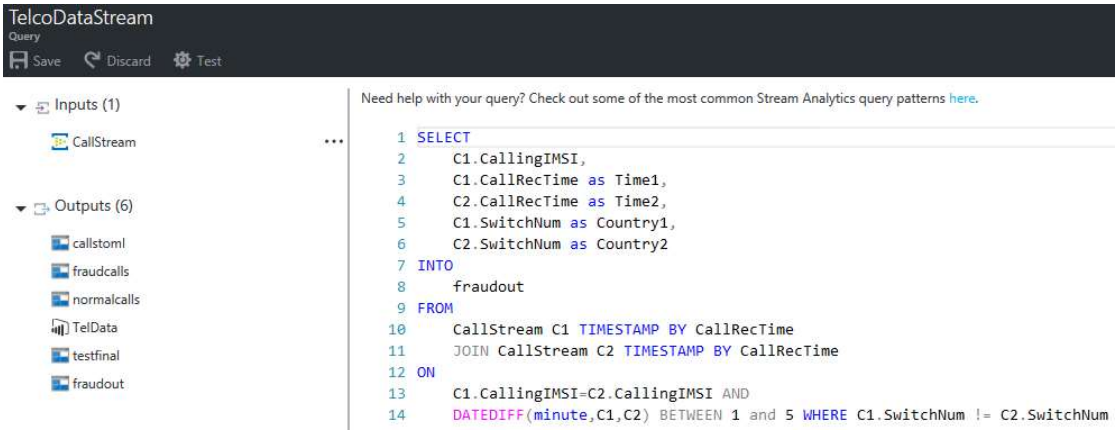
Ulostulo määritettiin klikkaamalla Stream Analytics -työn päänäkymässä ”outputs”-laatikkoa. Uusi output määriteltiin kuvion 21 mukaisesti. Output alias -kohtaan asetettavaa nimeä käytettiin myös dataan tehtävissä kyselyissä. Lisäksi ulostuleva data määritettiin CSV-muotoon, jotta sitä olisi helpompi tarkastella.



Kuvio 21. Stream Analytics -ulostulo

8.2.4 Kyselyn muodostaminen

Seuraavaksi muodostettiin kysely, jolla sisään tulevista puhelutiedoista pyrittiin poimimaan ainoastaan mahdolliset huijausyritykset. Kyselyn tarkoituksena oli valita kaikki puhelut, jotka tulevat saman SIM-kortin tunnisteella viiden minuutin sisällä, mutta soitot tapahtuvat eri maista. Kuviossa 22 on kuvattu tähän tarkoitukseen käytettävä kysely. Kyselyssä käytettiin hyväksi kolmea eri tietoa: ”CallingIMSI” oli puhelun aloittaneen laitteen SIM-kortin yksilöity tunniste, ”CallRecTime” oli soiton ajankohta ja ”SwitchNum” oli maa, josta puhelu oli lähtöisin.



Kuvio 22. Ensimmäinen Stream Analytics kysely.

Kysely toimi siten, että ensin se jakoi yhden sisään tulevan datavirran (”CallStream”) kahdeksi eri virraksi (”C1” & ”C2”) käyttäen JOIN-lauseketta ja vertaili niitä toisiinsa viiden minuutin aikavälillä. Kysely palautti tuloksena sellaisten puheluiden tiedot, jotka tulivat samalla SIM-kortin tunnisteella kahdesta eri maasta määritetyllä aikavälillä. Kuviossa 23 on esitettynä muutama rivi kyselyn tuloksista, joista voitiin huomata kyselyn toimineen odotetusti. Testauksessa käytetty kysely löytyy tekstimuotoisena liitteestä 1.

callingimsi,time1,time2,country1,country2
466920401237309,2016-11-20T18:39:17.0000000Z,2016-11-20T18:58:17.0000000Z,US,Germany
466922202546859,2016-11-20T18:39:18.0000000Z,2016-11-20T19:01:17.0000000Z,Australia,Germany
466922202546859,2016-11-20T18:39:17.0000000Z,2016-11-20T19:01:17.0000000Z,China,Germany
466921200135361,2016-11-20T18:39:18.0000000Z,2016-11-20T18:52:18.0000000Z,UK,China
466921200135361,2016-11-20T18:39:14.0000000Z,2016-11-20T18:52:18.0000000Z,UK,China
466923000886460,2016-11-20T18:39:17.0000000Z,2016-11-20T18:39:22.0000000Z,Australia,UK
466922202546859,2016-11-20T18:39:18.0000000Z,2016-11-20T18:42:18.0000000Z,Australia,US

Kuvio 23. Ensimmäisen kyselyn tulokset.

Koska kyselyn tulokset ohjattiin Blob -tallennustilaan CSV-muodossa, pystyttiin niitä tarkastelemaan Microsoft Office Excel -ohjelmalla. Blob -tallennustilaa pääsi helposti selaamaan Microsoft Azure Storage Explorer -työpöytäsovelluksella, joka löytyi Microsoftin sivuilta ilmaiseksi. Sovellukseen tarvitsi vain liittää oma Azure Storage -tili.

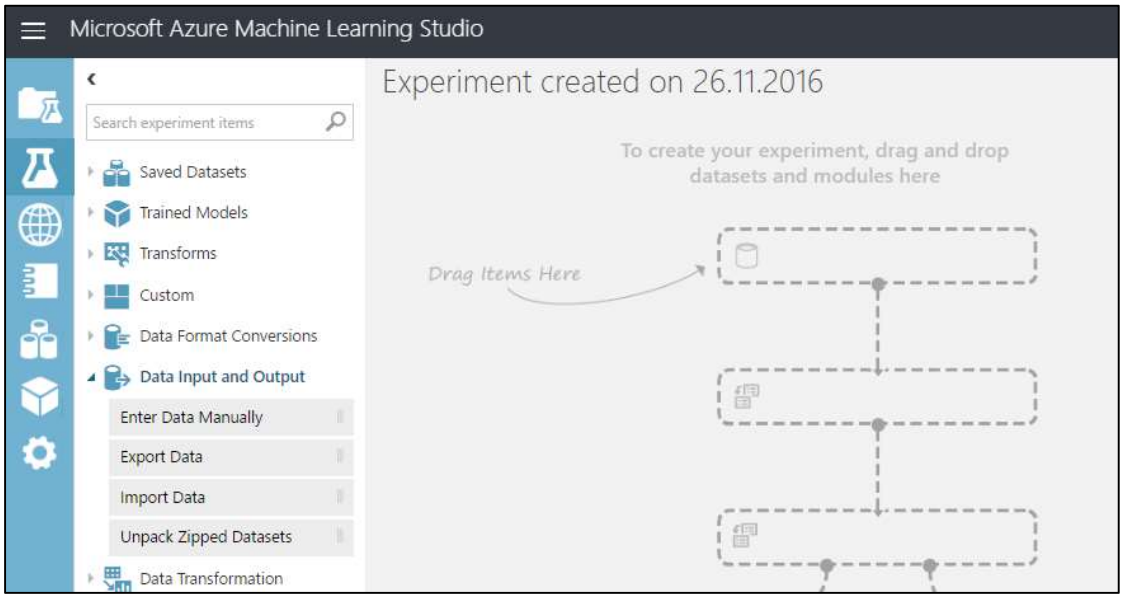
8.3 Testiasetelma 2.

8.3.1 Historiadatan valmistelu

Ensimmäisenä seuraavaa testausta varten tarvitsi luoda historiadataa, jolla Machine Learning -analysointimalli voitiin opettaa. Tämä toteutettiin siten, että ensin luotiin kaksi erillistä Blob -tallennustilaa. Toiseen generoitiin CSV-muodossa ainoastaan huijauspuhelutietoja ja toiseen vain normaaleja puhelutietoja. Tämän jälkeen jokaiselle puhelutiedolle asetettiin leima sen perusteella, onko se huijauspuhelu vai normaali puhelu. Huijaukset merkittiin numerolla ”1” ja normaalit puhelut numerolla ”0”. Lopuksi leimatut puhelutiedot yhdistettiin, jolloin saatiin yksi tiedosto, joka sisälsi molempia puheluita satunnaisessa järjestyksessä. Kuviossa 24 on esitetty kokonaisuudessaan ensimmäisessä vaiheessa käytetyn kokeen rakenne valmiina. Myöhempänä on selitetty kohta kohdalta jokaisen vaiheen lisääminen ja tarkoitus.

Kuvio 24. ML Experiment 1: Historiadatan leimaaminen.

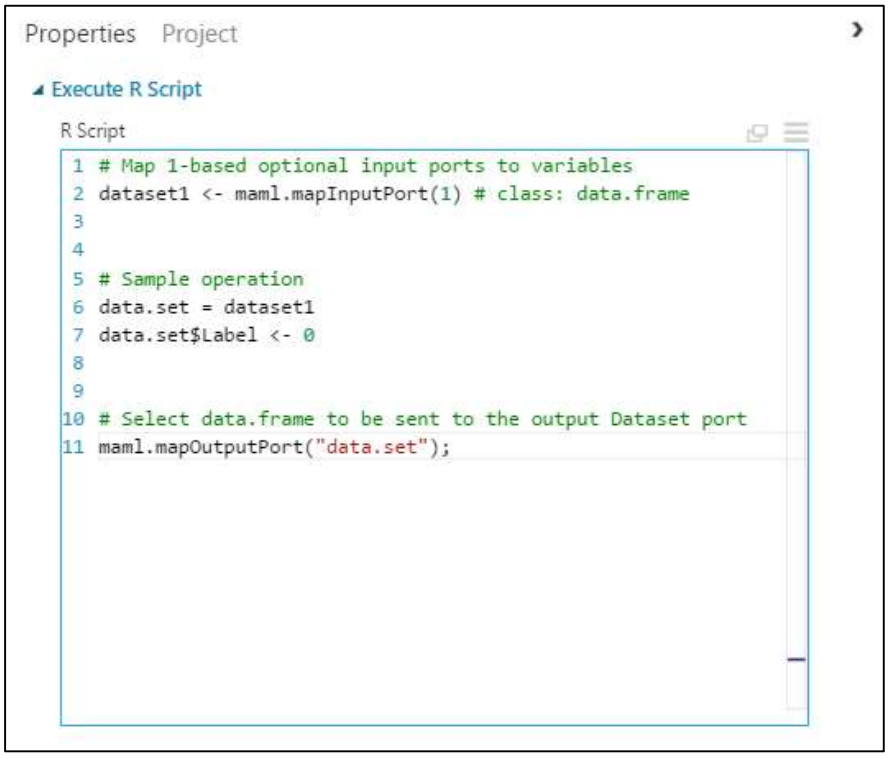
ML -ratkaisun rakentaminen lähti siis liikkeelle ensimmäisen yksittäisen ML -kokeen luomisesta. Kuviossa 25 on kuvakaappaus vielä tyhjänä olleen kokeen näkymästä. Kokeen rakentaminen aloitettiin yksinkertaisesti raahaamalla tyhjälle pohjalle vasemmasta sivupalkista haluttuja valmiita moduuleita.



Kuvio 25. ML Studio -näkymä.

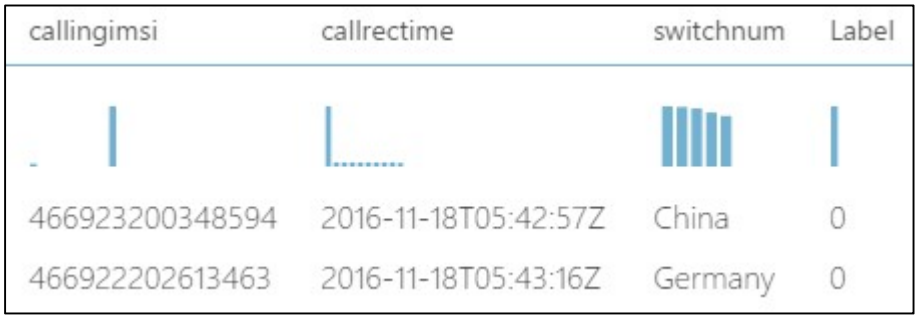
Aluksi molemmat puhelutietoja sisältävät Blob -tallennustilat piti saada linkitettyä kokeeseen. Tämä tapahtui lisäämällä kaksi uutta Import Data -moduulia. Toiseen linkitettiin tallennustila, joka sisälsi normaaleja puheluja ja toiseen taas huijauspuheluita sisältävä tallennustila.

Seuraavaksi puhelutietoihin lisättiin ”Label”-kenttä. Normaaleihin puheluihin kentän arvoksi määritettiin ”0” ja huijauspuheluihin ”1”. Tämä toteutettiin käyttämällä R-moduulia, johon voi syöttää vapaasti R-koodia klikkaamalla moduulia (ks. kuvio 26). Moduulissa käytetty koodi löytyy myös liitteestä 3 tekstimuotoisena.



Kuvio 26. R-moduulin käyttö

Kun moduuli oli lisätty, voitiin sitä testata painamalla ”Run”-painiketta, jolla voidaan suorittaa koko koe kerrallaan tai ainoastaan erikseen valittuja moduuleita. Tuloksia voitiin visualisoida suoraan ML Studiossa klikkaamalla halutun moduulin päältä hiiren oikealla näppäimellä ja valitsemalla ”Result dataset” -> ”Visualize”. Kuviossa 27 on esimerkki, miltä visualisoitu data näytti ML Studiossa leimaamisen jälkeen. Tulokset voidaan myös suoraan tallentaa uudeksi datasetiksi valitsemalla visualisoinnin sijaan ”Save as Dataset”.



Kuvio 27. Datan visualisointi.

Leimaamisen jälkeen puhelun aikatietoa, eli “CallRecTime” -kenttää haluttiin muuttaa helpommin käsiteltävään muotoon. Ensin aika ja päivämäärä eroteltiin eri kenttiin, jonka jälkeen kentistä poistettiin kaikki ylimääräiset merkit. Jäljelle jäi siis vain

numeerinen arvo. Tietojen erotteluun ja muotoiluun käytetyt R-koodit löytyvät liitteestä 6.

Muotoilun jälkeen alkuperäinen aikaleima, eli ”CallRecTime”-kenttä poistettiin kokonaan käyttämällä ”Select Columns in dataset” -moduulia. Sillä voidaan nimensä mukaisesti valita, mitkä kentät valitaan mukaan tuloksiin, tai jätetään pois tuloksista. Sarakkeet haluttiin vielä järjestellä käyttäen R-moduulia, jonka koodi löytyy liitteestä 7. Kun puhelutiedot olivat leimattu ja muotoiltu sopivaan muotoon, varmistettiin vielä, ettei tietoihin ole päässyt tuplarivejä. ”Remove Duplicate Rows”-moduulilla poistettiin mahdolliset ylimääräiset rivit.

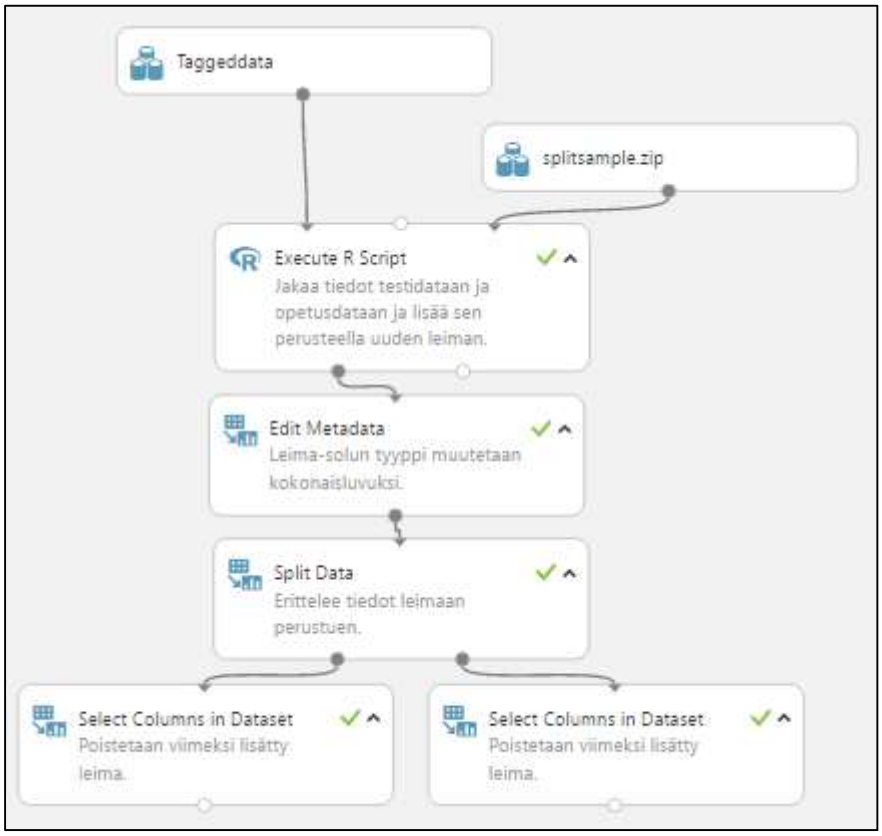
callingimsi	switchnum	Label	Date	Time
262021390056324	Australia	0	00010101	000000
262021390056324	Australia	0	00010101	000000

Kuvio 28. Puhelutiedot muotoilujen jälkeen

Lopuksi molemmat puhelutiedot yhtenäistettiin käyttäen ”Add Rows”-moduulia. Tulos tallennettiin yhdeksi uudeksi tiedostoksi nimeltä ”taggeddata” valitsemalla ”Save as dataset”. Tätä uutta datasettiä käytettiin sisääntulona seuraavassa kokeessa.

8.3.2 Datan jakaminen

Puhelutietojen leimaamisen ja sekoittamisen jälkeen data jaettiin taas kahteen osaan siten, että 70 % riveistä muodosti analysointimallin opettamiseen käytettävän datan ja loput 30 % mallin testaamiseen ja arviointiin käytettävän datan. Puhelutietoihin lisättiin uusi leima ”trainflag”, joka määritti, kumpaan osaan tieto kuuluu. Tietojen jakaminen tehtiin satunnaisesti, mutta kuitenkin siten, että kaikki samalla SIM-kortin tunnisteella tulleet tiedot sijaitsivat samassa osiossa. Tähän käytetyn ML-kokeen rakenne on esitetty kuviossa 29.



Kuvio 29. ML Experiment 2: Datan jakaminen

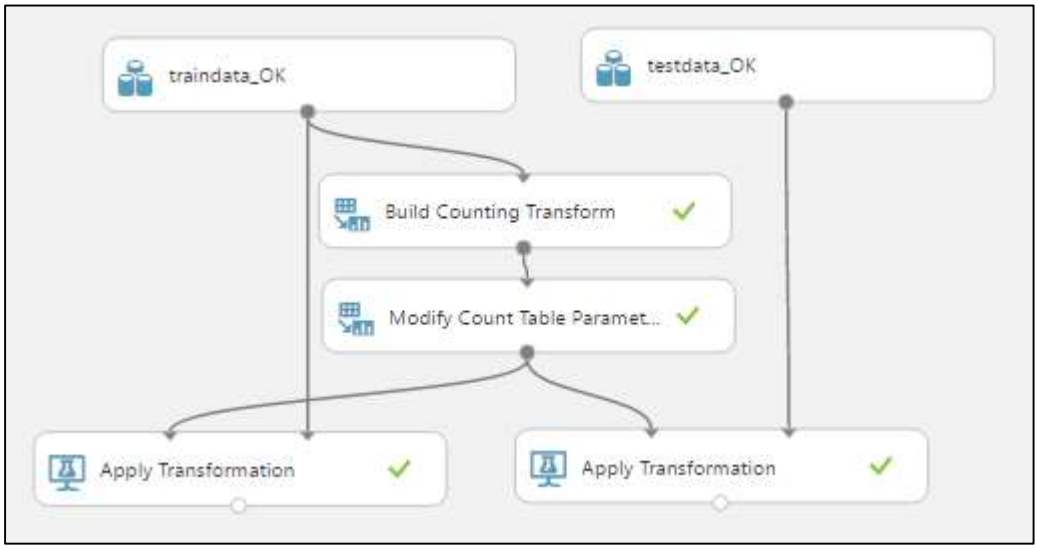
Kokeen ensimmäisessä vaiheessa puhelutiedot merkittiin kahteen eri ryhmään uudella leimalla. Tämä toteutettiin hyödyntämällä pakattuna tiedostona lisättyä R-koodia, joka pystyttiin liittämään suoraan R-moduuliin. Sille määritettiin ensin käytettävät parametrit kokeessa näkyvällä R-moduulilla. Moduulin koodi ja käytetyt parametrit löytyvät liitteestä 4 ja erillisenä lisätyn tiedoston koodi taas liittestä 5.

Uuden leiman sisältönä oli boolean-muuttuja (True/False), joka muutettiin numeeriseksi (1/0) ”Edit Metadata”-moduulissa. Seuraavaksi data jaettiin opetusdataan ja testidataan kyseisen leiman perusteella käyttäen ”Split Data”-moduulia. Jakamiseen käytetty leima voitiin nyt poistaa turhana ja tulokset tallennettiin taas uusiksi tiedostoiksi: Traindata & Testdata.

8.3.3 Arvojen muuntaminen

Tässä työvaiheessa datan arvot muutettiin lukumäärällisiksi arvoiksi käyttäen ”Build Counting Transform” -moduulia. Muunnoksen jälkeen alkuperäiset arvot poistettiin kokonaan, jolloin malli voitiin myöhemmässä vaiheessa opettaa käyttäen ainoastaan

näitä numeerisia arvoja. Kuviossa 30 on esitetty tämän vaiheen kokeen lopullinen rakenne. Tämän vaiheen suorittaminen ei ollut välttämätöntä, koska dataa oli vähän ja se oli hyvin yksinkertaista. Dataa olisi voitu käsitellä myös käyttäen alkuperäisiä arvoja, mutta datan muunnos haluttiin tehdä kokeilumielessä.



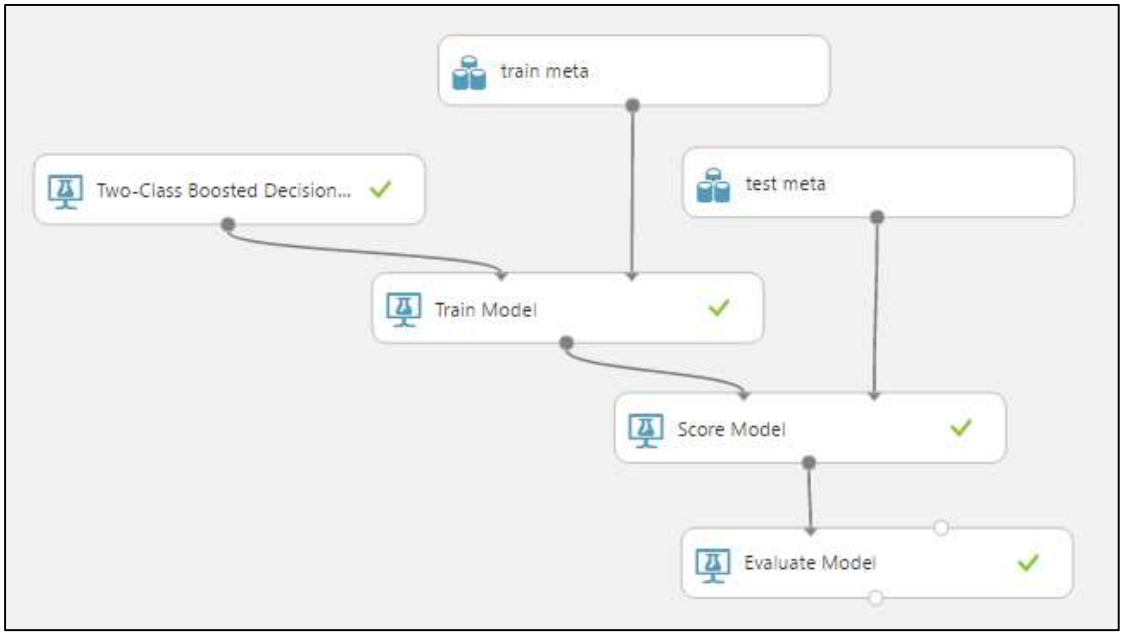
Kuvio 30. ML Experiment 3: Arvojen muuntaminen

Muunnos toimii siten, että se laskee, kuinka monta kertaa jokainen arvo esiintyy käytettävässä datassa, jolloin opetettava malli pystyy suoraan päättelemään, mitkä kentät sisältävät eniten oleellista tietoa. Jos datassa olisi esimerkiksi postinumero-kenttä ja siinä olisi yhteensä 40 000 eri postinumeroa, olisi huomattavasti helpompi laskea, kuinka monta huijausta tulee yhden postinumeron alueelta, kuin käsitellä kaikkia 40 000:tta numeroa erikseen. (Data Transformation / Learning with Counts 2016.)

8.3.4 Mallin luominen ja testaaminen

Kun historiadata oli jaettu testidataan sekä opetusdataan ja datan kaikki arvot olivat muunnettu helpommin käsiteltävään muotoon, oli aika valita käytettävä algoritmi ja luoda opetettu analysointimalli. Tässä vaiheessa käytetyn kokeen rakenne on esitetty kokonaisuudessaan kuviossa 31. Mallin opettaminen tapahtui käyttämällä ”Train Model” -moduulia. Siihen syötettiin aikaisemmin historiadatasta lohkottua opetusdataa ja liitettiin mallin opettamiseen valittu ”Two-Class Boosted Decision Tree” -algoritmimoduuli. Microsoftin dokumentaatioista löytyi hyvin ohjeistuksia algoritmimoduulin valitsemiseen.

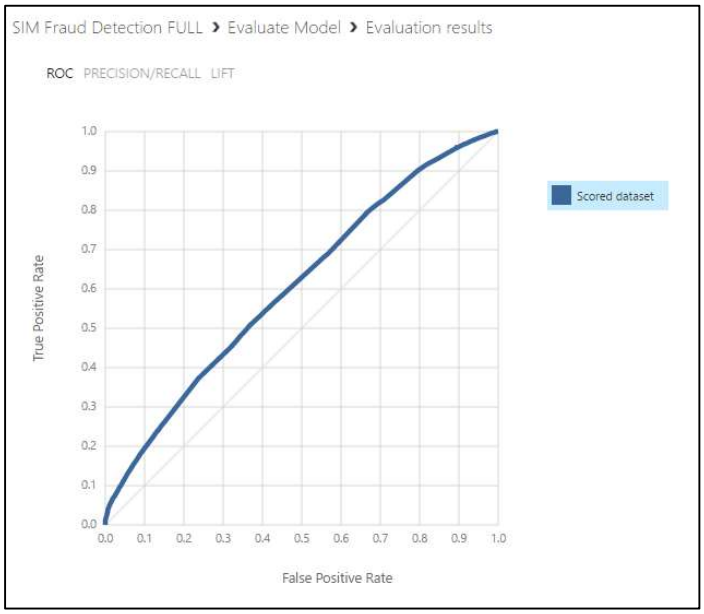
Valittu moduuli perustuu boosted decision tree -algoritmiin ja se löytyi hallittuun opettamiseen kuuluvien Classification -moduulien alta. Algoritmi soveltuu ennalta leimatulle datalle, jossa leiman arvoja voi olla vain kaksi. (Azure ML Team 2015)



Kuvio 31. ML Experiment 4: Mallin luominen ja arviointi

Mallin opettamisen jälkeen sitä voitiin koeajaa testidatalla käyttäen ”Score Model” -moduulia, johon laitettiin sisääntulona aikaisemmin lohkottu testidata ja siihen liitettiin opetettu analysointimalli. Testauksen tuloksia pystyttiin arvioimaan liittämällä moduulin ulostuloksi ”Evaluate Model” -moduuli.

Arviointi antoi tulokseksi visualisoidun Receiver Operating Charasteristics (ROC) -kuvaajan sekä muita numeerisia arvoja mallin tarkkuudesta. Käytännössä mitä enemmän käyrä sijoittuu vasempaan yläkulmaan, sitä tehokkaampi opetettu malli on. Kuviossa 32 on esitetty luodun mallin arvioinnin tulokset. Vaikka tarkkuus oli erittäin heikko johtuen käytettävän datan yksinkertaisuudesta ja vähäisestä määrästä, oli se silti yllättäen parempi kuin pelkkä puhdas arvaus. Mikäli käytettävässä datassa olisi ollut esimerkiksi arvo, joka määrittäisi soittajan kotiosoitteen, olisi tarkkuutta saatu nostettua jo huomattavasti korkeammalle.



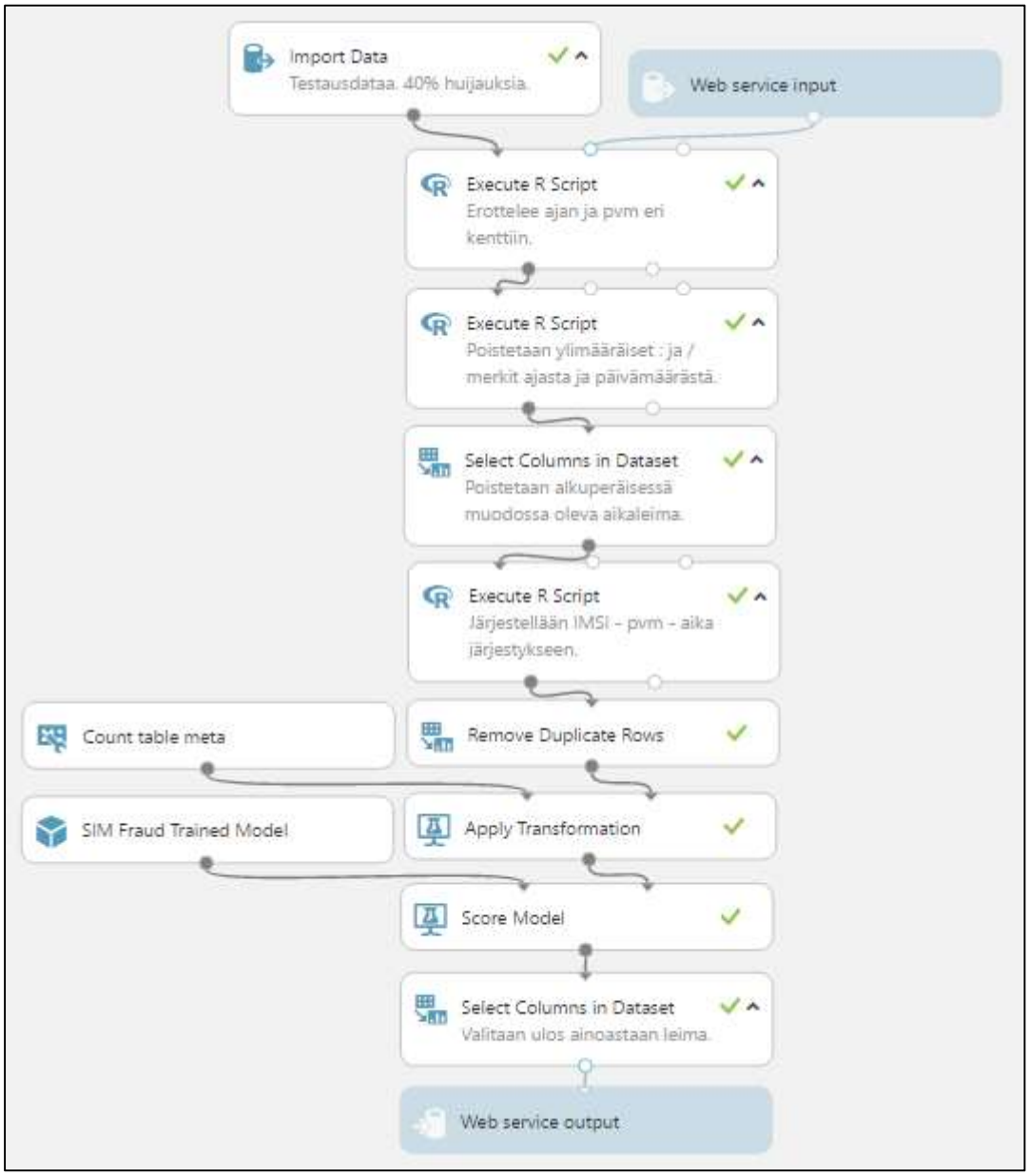
Kuvio 32. Mallin ROC -kuvaaja

Jos kyseessä olisi ollut oikea tuotantokäyttöön luotava ML -ratkaisu, niin tässä vaiheessa mallin luomisen prosessia olisi käyty useamman kerran läpi niin kauan, että tarkkuus olisi saatu mahdollisimman korkeaksi. Koska kyseessä oli kuitenkin työkalun ominaisuuksien testaus, tässä vaiheessa jatkettiin tallentamalla opetettu malli valmiiksi moduuliksi myöhempää käyttöä varten ML Studion moduulikirjastoon.

8.3.5 Web-palvelun luominen

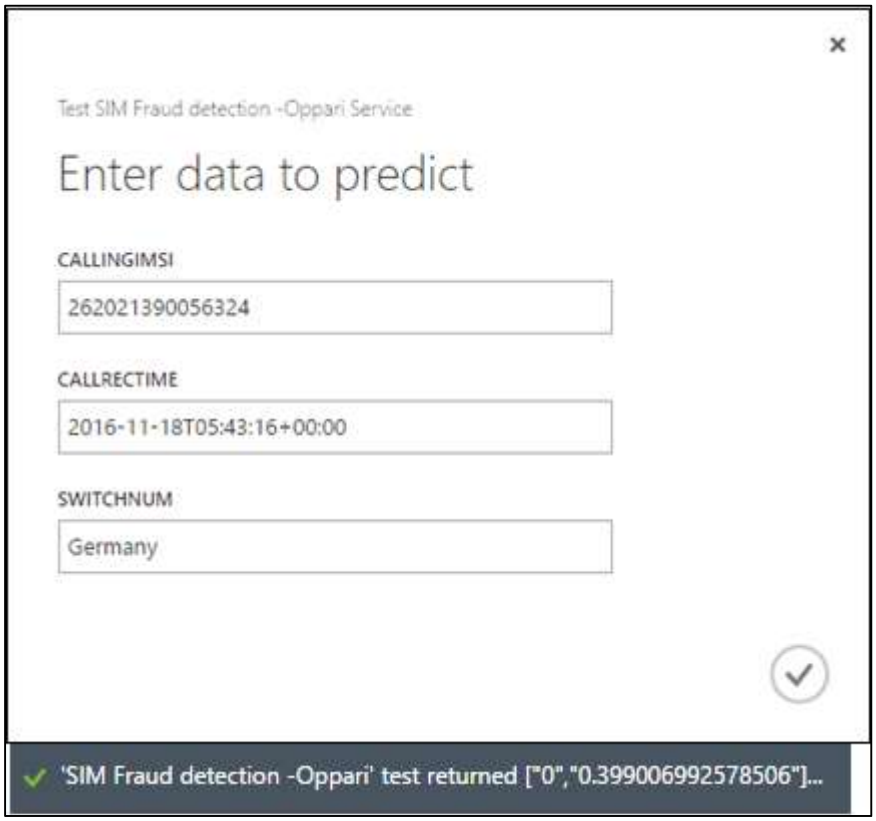
Seuraavana vaiheena oli muodostaa ML -ratkaisun lopullinen rakenne ja julkaista se web-palveluna, jota voidaan kutsua Stream Analyticsin kyselyssä. Kuten kuviosta 33 voidaan huomata, lopullinen rakenne oli oikeastaan vain tiivistetty yhdistelmä kaikista edellisistä vaiheista. Dataan tehtiin täsmälleen samat muutokset, kuin kappaleissa 5.3.1-5.3.3, mutta nyt sisään- ja ulostuloksi olikin määritetty web-palvelu. Sisään tulevia puhelutietoja analysoitiin ”Score Model” -moduulissa liittämällä siihen myös aikaisemmin opetettu analysointimalli.

Ennen lopullista julkaisua tehtiin vielä viimeinen testaus, että kaikki moduulit toimivat. Tätä varten generoitiin uutta puhelutietodataa 40 % huijausprosentilla. Tiedot tuotiin lopullisen ML-ratkaisun analysoitavaksi ”Import Data” -moduulilla ja katsottiin, että tulokset näyttivät järkeviltä. Ratkaisu tunnisti syötetyistä puhelutiedoista jopa 45 % huijauksiksi, mikä oli yllättävän hyvä tulos ottaen huomioon käytetyn mallin heikon tarkkuuden.



Kuvio 33. ML Experiment 5: Lopullinen ratkaisu

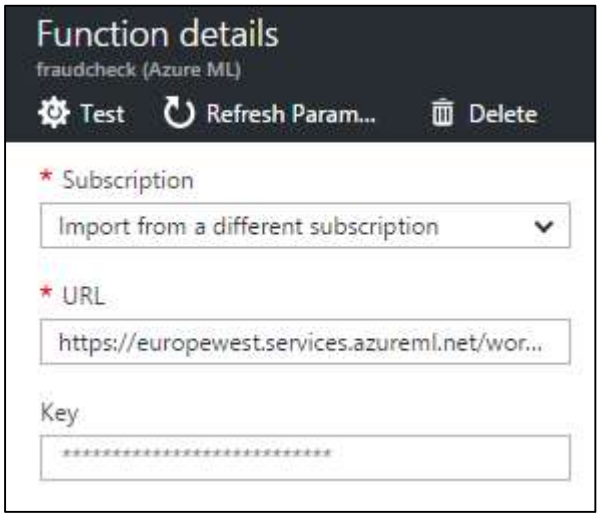
Kun ratkaisu oli julkaistu web-palveluna, voitiin sitä testata suoraan web-palvelu -välilehdeltä löytyvällä testausominaisuudella (ks. kuvio 34). Testaustyökaluun syötettiin vain esimerkkiarvot, jonka jälkeen se palautti sivun alareunaan tuloksena leiman sekä tarkan todennäköisyyden tulokselle.



Kuvio 34. Web-palvelun testaaminen

8.3.6 Stream Analytics -kyselyn muodostaminen

Seuraavaksi Stream Analytics määritettiin kutsumaan luotua ML -ratkaisua funktiona (ks. kuvio 35). Ensimmäisenä ML -funktio täytyi liittää Stream Analytics -työhön sen päänäkymästä löytyvältä ”Functions” -välilehdeltä. Liittämistä varten tarvittiin ML web-palvelun osoite sekä avain, jotka löytyivät suoraan web-palvelun päänäkymästä ML Studiosta ”Request/response” -linkin takaa.



Kuvio 35. ML -funktion lisääminen

Uutta funktiota lisättäessä Stream Analytics näytti suoraan esimerkkikyselyn, jolla kyseistä funktiota voitaisiin kutsua. Kuviossa 36 on esitetty lisätyn funktion esimerkkikysely. Funktion nimi oli ”fraudcheck” ja sille voitiin lähettää kolme eri ennalta määritettyä arvoa. Lopullinen työssä käytetty kysely löytyy liitteestä 2.

```
FUNCTION SIGNATURE
fraudcheck ( callingimsi BIGINT , callrectime DATETIME , switchnum NVARCHAR(MAX) ) RETURNS RECORD
```

Kuvio 36. ML -funktion toiminnallisuus

ML -ratkaisun analysoimat puhelutiedot ohjattiin uuteen Blob -tallennustilaan CSV-muodossa, jotta niitä pystyttiin tarkastelemaan suoraan Excel -ohjelmassa. Kuviossa 37 on otos kyselyn tuloksista ja kuten niistä voidaan huomata, puhelutiedoille muodostui leima oikein.

1	callingimsi,callrectime,switchnum,scored labels,scored probabilities
2	466923100098619,2016-11-20T11:35:47.0000000Z,Germany,1,0.50611013174057
3	466923101048691,2016-11-20T11:35:48.0000000Z,China,1,0.571562528610229
4	466923101048691,2016-11-20T11:35:48.0000000Z,Australia,1,0.552080035209656

Kuvio 37. Kyselyn tulokset

9 Pohdinta

Opinnäytetyön tavoitteena oli tutkia Microsoft Azure -pilvipalvelun reaaliaikaisen datan käsittelyyn ja analysointiin tarkoitettujen palveluiden toimintaperiaatteita yleisesti. Erityisesti työssä haluttiin testata Stream Analytics -ja Machine Learning -työkaluja johonkin mahdollisimman realistiseen käyttötarkoitukseen, jotta niiden erilaisista mahdollisuuksista saataisiin parempi käsitys.

Itselläni oli alun perin hyvin vähän tietoa tutkittavasta aiheesta, joten aluksi aikaa kului luonnollisesti perustietojen keräämiseen ja aiheen sisälle pääsemiseen. Big Data on tällä hetkellä kuuma aihe ja sen vuoksi siihen liittyvää ajantasaista tietoa oli onneksi helppoa löytää niin verkosta kuin myös painetusta kirjallisuudesta. Yleinen teoriaosuus Big Datasta onkin rakennettu hyvin pitkälti painetun kirjallisuuden pohjalta. Azuren työkaluihin liittyvä tieto löytyi kokonaan Microsoftin tarjoamista ilmaisesta Azure -dokumentaatioista ja e-kirjoista.

Työkalujen virallisissa dokumentaatioissa havaittiin paljon puutteita. Pelkästään Microsoftin omaa dokumentaatiota lukemalla palveluista sai todella yksinkertaisen ja helppokäyttöisen kuvan, mutta testausosuuden aikana ongelmakohtia alkoi ilmetä. Esimerkiksi palveluiden kehittyessä jatkuvasti ohjeistukset olivat jo osittain päässeet vanhenemaan, jonka vuoksi työkalujen ulkoasu oli usein hyvin paljon ohjeistuksista poikkeavaa. Ohjeistuksista tuntui myös monesti puuttuvan perustelut ja tarkemmat selitykset, miksi jokin asia tehdään niin kuin se on esitetty. Suurimmaksi ongelmaksi muodostui dokumentaatioiden puutteellisuus ongelmatilanteiden osalta ja koska verkosta ei vielä löydy paljoakaan muiden käyttäjien tekemiä testauksia palveluiden ollessa vielä uusia, oltiin suurimmaksi osaksi ainoastaan Microsoftin omien dokumentaatioiden varassa. Jos dokumentaatiosta ei löytynyt ratkaisua johonkin tiettyyn ongelmakohtaan, niin pientäkin ongelmaa saattoi ratkoa välillä pitkäänkin ja ainut tapa oli vain yksinkertaisesti kokeilla eri vaihtoehtoja ratkaisuksi. Tällainen ongelmakohta tuli esimerkiksi tiedostomuotojen kanssa, kun Stream Analyticsin käsittelemää JSON-muodossa olevaa dataa yritettiin viedä suoraan Machine Learning -työkalun analysoitavaksi. Missään ei tuntunut olevan selvää mainintaa, että miksi tämä ei olisi mahdol-

lista tai olisiko siihen jotain kiertotietä. Lopulta ongelma ratkaistiin tallentamalla dataa ensin Blob -tallennustilaan, jonne se vietiin Stream Analyticsiltä suoraan CSV-muodossa.

Työkalujen testausosuudella saatiin hyvä kuva työkalujen toimintaperiaatteista ja mielestäni opinnäytetyön tavoitteet saavutettiin myös siltä osin. Ensimmäinen testiasetelma havainnoillisti erittäin hyvin, mitä Stream Analyticsillä ja Event Hubsilla voidaan saada aikaiseksi. Vastaavanlainen kokoonpano voidaan ottaa käyttöön hyvinkin nopeasti ja helposti muihin saman tyyppisiin käyttökohteisiin. Kyselykielen käyttäminen onnistuu, mikäli SQL-kieli on vähänkään tuttua. Seuraavana kehitysaskeleena näiden kahden palvelun testaamiselle voisi olla niiden todellisen käsittelynopeuden tutkiminen. Työkalut eivät antaneet ainakaan suoraan minkäänlaista palautetta tai tilastoja suoriutumisistaan. Testauksen aikana analysoinnissa ei huomattu viiveitä, mutta olisi mielenkiintoista nähdä, alkaako datalähteiden ja datan määrän lisääntyessä ilmenemään mahdollisesti jotain ongelmia.

Machine Learning -maailmaan perehtyminen vei huomattavasti enemmän aikaa, koska alue oli hyvin laaja. Teoriaosuudessa pyrittiin tiivistämään aiheen perusteet ja testausosuudessa havainnollistamaan kokonaisen ML -ratkaisun käyttöönotto. Jo testauksen alkuvaiheessa oli epäilys, että valitulla testidatalla ei voitaisi luoda kovinkaan tarkkoja tuloksia antavaa ML -mallia datan ollessa hyvin yksinkertaista. Työkalujen toimintojen vertailun vuoksi testauksessa päätettiin kuitenkin jatkaa saman testidatan käyttämistä ja pyrittiin luomaan mahdollisimman realistinen ratkaisu huolimatta sen tuloksista juuri kyseisellä testidatalla. Tulokset kuitenkin vahvistivat epäilystä, että Machine Learning -tekniikkaa ei voitaisi käyttää, mikäli data ei ole tarpeeksi monipuolista. Yksinkertaisemmallakin datalla voidaan kuitenkin saada tuloksia, jos siihen on mahdollista liittää jokin toinen lisäarvoa tuova tietolähde. Testauksessa olisi esimerkiksi voinut käyttää jotain ulkoista tietokantaa, josta selviää soittajan henkilöllisyys perustuen soittajan tunnisteeseen. Tällaista tietoa ei kuitenkaan ollut saatavilla.

Azure Machine Learning -tutkimisen seuraavana luonnollisena askeleena olisi sen vertailu kilpailijoiden, kuten IBM:n vastaaviin palveluihin. Eri valmistajat hakevat jalansijaa tuotteilleen Big Data -markkinoilla ja vielä on mielestäni aikaista sanoa, kuka

tuon kilpailun tulee voittamaan. Lisäksi olisi hienoa nähdä tulevaisuudessa myös jokin esimerkki hallitsemattoman oppimisen clustering -algoritmin käyttämisestä, jota ei ehditty ajan puutteen vuoksi testaamaan.

Azure -portaalin käyttöliittymä vaati pientä totuttelua, mutta ottaen huomioon sen sisältämien eri palveluiden määrän, on sen rakenne kuitenkin pysynyt omasta mielestäni selkeänä. Portaalin käytössä huomattiin kuitenkin ajoittain hidastelua, mikä saattaa haitata palvelun aktiivisemmassa käytössä. ML Studion käytössä ei huomattu minkäänlaisia ongelmia. Käyttöliittymä oli selkeä ja Azure -portaalin kaltaista hidastelua ei ilmennyt. ML -kokeiden rakentaminen oli myös sujuvaa, kun moduulikirjasto oli yksinkertainen ja moduulit hyvin dokumentoituja. Mielestäni olisi kuitenkin käytön kannalta miellyttävämpää, jos ML Studio löytyisi myös Azure -portaalista.

ML -ratkaisun luomisessa käytettiin paljon R-moduuleita, jotka on luotu käyttäen R-ohjelmointikieltä. Kyseisestä kielestä ei ollut minkäänlaisia kokemuksia ennen testausta ja tekijän ohjelmointitaustan ollessa muutenkin heikko, oli täysin uuden kielen käyttäminen aluksi erittäin haastavaa. R-kieli osoittautuikin yllättävän helpoksi ja R-moduulit saatiin luotua etsimällä verkosta erilaisia erimerkkejä ja soveltamalla niitä. Mikäli R-kieli on ennestään tuttua, ei ML -ratkaisujen luomisessa kulu varmastikaan paljoakaan aikaa, jos työkalun perusidea on hallussa.

Lopuksi voitaisiin vielä todeta, että vaikka työkalut ovat erittäin tehokkaita, käytännöllisiä ja nopeasti käyttöönotettavissa, on niiden toiminnan ymmärtäminen kuitenkin vasta pieni osa kokonaisten Big Data -ratkaisujen rakentamisessa. Suurin ja haastavin osuus on keksiä jokin idea, millä kyseisiä työkaluja voidaan hyödyntää. Todelliset aarteet big datan hyödyntämisessä piilevät edelleen siis innovaatioissa, mutta nämä työkalut mahdollistavat niiden toteuttamisen ilman vuosien kokemusta. Haastavuutta tuo myös ratkottavan ongelman tyypin tunnistaminen. Ensin pitää pystyä tunnistamaan, onko ratkottava ongelma edes big data -ongelma, jonka jälkeen päästään vasta työkalujen valintaan. Nyrkkisääntönä Azure -maailmassa voidaan kuitenkin pitää, että yksinkertaisemmat reaaliaikaiset analyysit kannattaa tehdä käyttäen Stream Analyticsiä. Machine Learning taas tuo mukaan monimutkaisemmat analyysit ja mahdollisuuden ennustaa tulevaa, mutta tarkkojen tulosten saamiseksi historiadatan pitää olla riittävän monipuolista.

Lähteet

Azure ML Team. 2015. Machine Learning algorithm cheat sheet. Microsoft Docs. Viitattu 6.12.2016. https://docs.microsoft.com/fi-fi/azure/machine-learning/machine-learning-algorithm-cheat-sheet

Azure ML Team. 2016. Machine Learning Templates with Azure ML Studio. Cortana Intelligence Gallery. Viitattu 6.12.2016. https://gallery.cortanaintelligence.com/Collection/Machine-Learning-Templates-with-Azure-ML-Studio-1

Azure Storage Pricing. N.d. Microsoft Azure Storage -tuotesivu. Viitattu 11.12.2016. https://azure.microsoft.com/fi-fi/pricing/details/storage/blobs/

Azure support plans. N.d. Microsoft Azure tukivaihtoehtojen tuotesivu. Viitattu 11.12.2016. https://azure.microsoft.com/en-us/support/plans/?b=16.44

Barnes, J. 2015. Microsoft Azure Essentials: Azure Machine Learning. E-kirja. Microsoft Virtual Academy. Viitattu 22.10.2016. https://mva.microsoft.com/ebooks#9780735698178

Data Transformation / Learning with counts. 2016. . Microsoft Azure Machine Learning -dokumentaatio. Viitattu 17.12.2016. https://msdn.microsoft.com/en-us/library/azure/dn913056.aspx

Deshmane, S. 2015. Using the Lambda Architecture on a Big Data Platform to Improve Mobile Campaing Management. Viitattu 24.10.2016. http://www.talentica.com/pdf/Big-Data-Using-Lambda-Architecture.pdf

Eaton, C., Deutsch, T., deRoos, D., Lapis, G. & Zikopoulos, P. 2012. Understanding Big Data. Analytics for Enterprise Class Hadoop and Streaming Data. Viitattu 18.10.2016. http://public.dhe.ibm.com/common/ssi/ecm/im/en/iml14296usen/IML14296USEN.PDF

Event Hubs Pricing. N.d. Microsoft Azure Event Hubs -tuotesivu. Viitattu 10.12.2016. https://azure.microsoft.com/en-us/pricing/details/event-hubs/

Ericson, G. 2016. How to choose algorithms for Microsoft Azure Machine Learning. Microsoft Azure Machine Learning -dokumentaatio. Viitattu 17.12.2016. https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice

Feddersen, C. 2015. Real-Time Event Processing with Microsoft Azure Stream Analytics. Microsoft Docs. Viitattu 26.11.2016. https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-real-time-event-processing-reference-architecture

Gupta, N., Khan, A. & Uddin, M. 2014. Seven V's of Big Data. Understanding Big Data to Extract Value. Viitattu 19.10.2016. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6820689

Hamilton, D. 2016. The role of machine learning in real-time fraud detection. NCR blog. Viitattu 6.12.2016. https://www.ncr.com/company/blogs/financial/role-machine-learning-real-time-fraud-detection

Hackeling, G. 2014. Mastering Machine Learning with scikit-learn. E-kirja. PACKT Publishing. Viitattu 25.12.2016. https://www.packtpub.com/big-data-and-business-intelligence/mastering-machine-learning-scikit-learn

Hirsjärvi S., Remes P. & Sajavaara, P. 2007. Tutki ja kirjoita. 13. p. Keuruu: Otava

Lazzeri, F. 2016. Cluster Model. Microsoft Azure Machine Learning -dokumentaatio. Viitattu 17.12.2016. https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-r-csharp-cluster-model

Liikevaihto ja henkilöstö. N.d. Inmics Oy henkilöstön ja liikevaihdon kuvaus yrityksen kotisivulla. Viitattu 10.10.2016. http://www.inmics.fi/yritys/liikevaihto-ja-henkilosto/

Machine Learning Pricing. N.d. Microsoft Azure Machine Learning -tuotesivu. Viitattu 11.12.2016. https://azure.microsoft.com/en-us/pricing/details/machine-learning/

Manheim, S. 2016. Azure Event Hubs overview. Viitattu 30.10.2016. https://azure.microsoft.com/en-us/documentation/articles/event-hubs-overview/#

Manheim, S. 2016. Azure Event Hubs authentication and security model overview. Viitattu 10.12.2016. https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-authentication-and-security-model-overview

Marz, N. & Warren, J. 2015. Big Data. Principles and best practices of scalable real-time data-systems. New York: Manning Publications.

Perhe takaa kestävän kasvun. N.d. Inmics Oy esittely yrityksen kotisivuilla. Viitattu 10.10.2016. http://www.inmics.fi/yritys/perhe-takaa-kestavan-kasvun/

RFC 4180:2005. Common Format and MIME Type for Comma-separaterd Values (CSV) Files. Internet Engineering Task Force (IETF). Viitattu 26.11.2016. https://tools.ietf.org/html/rfc4180#page-2

RFC 7159:2014. The JavaScript Object Notation (JSON) Data Interchange format. Internet Engineering Task Force (IETF). Viitattu 26.11.2016. https://tools.ietf.org/html/rfc7159#page-4

Salo, I. 2013. Big Data. Tiedon vallankumous. Jyväskylä: Docendo.

Stokes, J. 2016. Introduction to Stream Analytics Window functions. Viitattu 26.11.2016. https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-window-functions

Stream Analytics Pricing. N.d. Microsoft Azure Stream Analytics -tuotetsivu. Viitattu 11.12.2016. https://azure.microsoft.com/fi-fi/pricing/details/stream-analytics/

Yhteystiedot. N.d. Inmics Oy toimipisteiden yhteystiedot yrityksen kotisivulla. Viitattu 10.10.2016. http://www.inmics.fi/yhteystiedot/

Yritys. N.d. Inmics Oy esittely yrityksen kotisivuilla. Viitattu 10.10.2016. www.inmics.fi/yritys/

Liitteet

Liite 1. Testiasetelma 1: Kysely

```
SELECT
        C1.CallingIMSI,
        C1.CallRecTime as Time1,
        C2.CallRecTime as Time2,
        C1.SwitchNum as Switch1,
        C2.SwitchNum as Switch2
INTO
        fraudout
FROM
                CallStream C1 TIMESTAMP BY CallRecTime
        JOIN    CallStream C2 TIMESTAMP BY CallRecTime
ON
        C1.CallingIMSI=C2.CallingIMSI
AND
        DATEDIFF(minute,C1,C2) BETWEEN 1 and 5
        WHERE CS1.SwitchNum != CS2.SwitchNum
```

Liite 2. Testiasetelma 2: Kyselyt

```
# Historiadatan keräämisessä käytetty yksinkertainen kysely.
SELECT
   CallingIMSI,
   SwitchNum,
   CallRecTime
INTO
   fraudcalls                    #Tämä tehtiin kahteen erilliseen tallennustilaan.
FROM
   CallStream


# Lopullinen kysely ML -kutsulla.
WITH subquery AS (
  SELECT
                CallingIMSI,
                CallRecTime,
                SwitchNum,
                fraudcheck(CallingIMSI, CallRecTime, SwitchNum)
                as result from CallStream
)

SELECT CallingIMSI, CallRecTime, SwitchNum, result.[Scored Labels]
Into testresults
From subquery
```

Liite 3. R-moduuli: Label-kentän lisääminen.

```
# Map 1-based optional input ports to variables
dataset1 <- maml.mapInputPort(1) # class: data.frame

# Sample operation
data.set = dataset1
data.set$Label <- 0

# Select data.frame to be sent to the output Dataset port
maml.mapOutputPort("data.set");
```

Liite 4. R-moduuli: Datan jakaminen

```
dataset1 <- maml.mapInputPort(1)          # Sisääntulon määrittäminen:

source("src/splitsample.R")


# splitsample.R -koodi (ks. liite 5) oli lisättynä ML Studioon pakattuna tiedostona.

# Se jakaa puhelutiedot kahteen eri ryhmään ”CallingIMSI” -kentän perusteella:
Kaikki samalla tunnisteella olevat tiedot pysyvät samassa ryhmässä.

#  key:   Määritellään avain, jonka perusteella jakaminen tapahtuu.

#  nfRate : Kuinka paljon normaaleja puhelutietoja otetaan.

#  frdRate: Kuinka paljon huijauspuhelutietoja otetaan.

# A `trainFlag` -leima lisätään.

#   trainFlag = 1 on opetusdataa

#   trainFlag = 0 on testidataa


key = 'CallingIMSI'

nfRate = 0.7

frdRate = 0.7


data.set = splitDataByKey(dataset1, key, nfRate, frdRate)

maml.mapOutputPort("data.set");          # Ulostulon määrittäminen
```

Liite 5. Splitsample.R

```
# Tämä koodi oli lisättynä pakattuna tiedostona ML Studioon.
splitDataByKey<-function(dataset, key, NFrate, Frate= NFrate)
{
  Fimsi = unique(dataset[dataset$Label > 0,][key])
  set.seed(23)
  Frand = runif(dim(Fimsi)[1])
  Fimsi$trainFlag = (Frand <= Frate)

  NFimsi = unique(dataset[dataset$Label == 0,][key])
  NFrand = runif(dim(NFimsi)[1])
  NFimsi$trainFlag = (NFrand <= NFrate)

  data.set = merge(dataset, rbind(Fimsi,NFimsi), by= key, all.x =TRUE, sort=FALSE)
  return (data.set)
}
```

Liite 6. R-moduuli: Ajan ja päivämäärän muotoilu

```
# YLIMÄÄRÄISTEN MERKKIEN POISTAMINEN
# Map 1-based optional input ports to variables
dataset1 <- maml.mapInputPort(1) # class: data.frame

data.set = dataset1
data.set$Time = gsub(":","",dataset1$Time)  #Poistaa erottimet ajasta.
data.set$Date = gsub("-","",dataset1$Date)   #Poistaa erottimet päivämäärästä.

# Select data.frame to be sent to the output Dataset port
maml.mapOutputPort("data.set");

# AJAN JA PVM EROTTELU
# Map 1-based optional input ports to variables
dataset1 <- maml.mapInputPort(1) # class: data.frame

# Jakaa merkkijonot uusiin erillisiin kenttiin välilyönnin kohdalta.
new <- do.call( rbind , strsplit( as.character( dataset1$callrectime ) , " " ) )
data.set = cbind( dataset1 , Date = new[,1] , Time = new[,2] )

# Select data.frame to be sent to the output Dataset port
maml.mapOutputPort("data.set");
```

Liite 7. R-moduuli: Sarakkeiden järjesteleminen

```
# Map 1-based optional input ports to variables
dataset1 <- maml.mapInputPort(1) # class: data.frame

# Sample operation / Järjestelee sarakkeet
data.set = dataset1[with(dataset1, order(callingimsi,Date,Time)),]

# Select data.frame to be sent to the output Dataset port
maml.mapOutputPort("data.set");
```