Opinnäytetyö (AMK)

Tietojenkäsittely

2019

Roni Ahti

KONEOPPIMISEN KÄYTTÖ ASIAKASPOISTUMA-ANALYYSISSA ASIAKASSUHTEEN PÄÄTTYMISEN ENNUSTAMISEEN

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely

2019 | 30 sivua, 11 liitesivua

Roni Ahti

KONEOPPIMISEN KÄYTTÖ ASIAKASPOISTUMA-ANALYYSISSA ASIAKASSUHTEEN PÄÄTTYMISEN ENNUSTAMISEEN

Yritykset usein haluavat ennustaa asiakaspoistumista. Asiakaspoistumisen ennustamisen avulla yritykset voivat säästää rahaa. Asiakaspoistumisen ennustamiseen voi käyttää koneoppimista. Opinnäytetyössä suunniteltiin ja luotiin prototyyppi ohjelmasta, joka ennustaa asiakaspoistumista koneoppimista käyttäen. Prototyypin toimivuutta mitattiin ohjelman tulostamien ennusteiden tarkkuudella.

Työn tavoitteena oli luoda prototyyppi koneoppimismallista, joka osaa Google Analytics -dataa hyödyntäen ennustaa, ketkä asiakkaat lopettavat palvelun käytön. Työhön kuului datan tuominen, datan vieminen, datan formatointi ja koneoppimismallin rakentaminen.

Työssä käytettiin työkaluina Python -ohjelmointikieltä ja siihen rakennettuja kirjastoja. Koneoppimismallin rakentamiseen käytettiin kirjastoja TensorFlow ja Keras. Työssä tarkasteltiin, onko mahdollista ennustaa asiakkaiden lähtemistä, käyttäen Google Analytics -dataa.

Lopputulokseksi saatiin toimiva prototyyppi, joka noin 70 prosentin varmuudella osaa ennustaa asiakaspoistumisen. Prototyypin käyttö on myös suhteellisen helppoa.

ASIASANAT:

Neuroverkko, koneoppiminen, asiakaspoistuminen, Google Analytics, TensorFlow, Keras, Python

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology

2019 | 30 pages, 11 pages in appendices

Roni Ahti

USING MACHINE LEARNING IN CHURN-ANALYSIS TO PREDICT TERMINATION OF CUSTOMER RELATIONSHIP

Businesses often want to predict customer churn. By predicting customer churn businesses can save money. It is possible to use machine learning to predict customer churn. The purpose of this thesis was to plan and create a prototype of a program that is able to predict customer churn with the help of machine learning. The functionality of the prototype would be measured by the accuracy of the outputted predictions.

The goal of the thesis was to create a prototype machine learning model that would be able to predict customer churn by using data from Google Analytics. The stages of the project included importing data, exporting data, formatting the data and building the machine learning model.

Tools used in the project are Python and many of its libraries. For building the machine learning model TensorFlow and Keras were used. The project was a proof of concept whether it is possible to predict churn using only data from Google Analytics.

The final result was a working prototype that is able to predict customer churn with a 70 percent accuracy. The use of the prototype is also relatively easy.

KEYWORDS:

Neural networks, machine learning, churn, Google Analytics, TensorFlow, Keras, Python

SISÄLTÖ

LIITTEET

KUVAT

TAULUKOT

KÄYTETYT LYHENTEET JA SANASTO

Asiakaspoistuminen	Kun palvelun käyttäjä lopettaa palvelun käytön.
Binäärinen	Kahteen jaollinen järjestelmä. Binäärisessä vertailussa vastaus on joko 1 tai 0, tosi tai epätosi.
Datakehys	Datan esittäminen taulukossa, jossa rivejä ja sarakkeita.
Google Analytics	Google tarjoama työkalu, jolla voi kerätä verkkosivuston käyttäjistä dataa.
Google Tag Manager	Google tarjoama työkalu, jolla voi injektoida Javascriptiä verkkosivustolle
Hajautusalgoritmi	Algoritmi, joka muuttaa syötteen ennalta määrätyn kokoiseksi tulosteeksi. Tulostetta ei voi muuttaa takaisin syötteeksi.
Hyperparametri	Koneoppimisessa käytettävä vakio, joka vaikutta mallin suorituskykyyn.
Javascript	Verkkoselaimen tulkitsema ohjelmointikieli, jolla lisätään verkkosivustoille ominaisuuksia.
JSON	Javascript object notation. datan tallentamisen tyyppi, joka emuloi Javascriptin objekteja.
Keras	Korkean abstraktion omaava koneoppimis rajapinta, joka on kirjoitettu Pythonilla.
Kirjasto	Ohjelmointikieleen rakennettuja apufunktioita, joita voi ladata oman ohjelman käyttöön.
Klassifikaatio	Koneoppimisalgoritmi, jonka avulla ennustetaan mihin luokkaan syöte kuuluu.
Koneoppiminen	Ohjelmatyyppi, joka muuttaa käyttäytymistään ilman ohjelmakoodin muuttamista.
Lineaarinen regressio	Koneoppimisalgoritmi, jonka avulla ennustetaan syötteen perusteella lineaarisia lukuja.
Neuroni	Koneoppimismallin sisällä oleva funktio, joka ottaa sisään dataa ja antaa sen perusteella tulosteen.
Neuroverkko	Koneoppimismalli, jossa data kulkee monen neuronikerroksen läpi.
Numpy	Tieteelliseen laskentaan käytettävä kirjasto.
Ohjelmointirajapinta	Määritelmä, jonka mukaan ohjelmaan voi lähettää pyyntöjä

Pandas	Helppokäyttöinen Python kirjasto, joka tarjoaa työkaluja data-analyysin ja -rakenteiden hallintaan.
Python	Korkeatasoinen ohjelmointikieli, jota käytetään usein data-analyysissä.
Tensorflow	Koneoppimiseen käytettäviä kirjastoja, työkaluja ja resursseja sisältävä ohjelmointi alusta.

1 JOHDANTO

Asiakaspoistumisen ennustamisen avulla yritykset voivat säästää paljon rahaa. Asiakaspoistumista ennustetaan monella eri tavalla. Nykyteknologian avulla asiakaspoistumista on mahdollista ennusta koneoppimista käyttäen.

Asiakaspoistumisen ennustaminen on tärkeää, koska uusien asiakkaiden hankkiminen on huomattavasti kalliimpaa, kuin vanhojen asiakkaiden säilyttäminen. Opinnäytetyössä rakennettu ohjelma tarjoaa yrityksille mahdollisuuden ennustaa asiakaspoistumista. Asiakaspoistumisen ennustamisen avulla yritykset voivat päätellä, ketä asiakkaita yrityksen tarvitsee aktivoida.

Koneoppimisen käyttö asiakaspoistuma-analyysissä on teknisesti haastavaa. Opinnäytetyössä ratkaistiin, miten koneoppimista käytetään teknisestä näkökulmasta.

Tässä opinnäytetyössä käsitellään asiakaspoistumisen ennustamista käyttäen Google Analytics -dataa ja koneoppimista. Rakennettuun prototyyppiin kuuluu datan tuominen ohjelmaan, datan formatointi, koneoppimismallin koulutus ja koneoppimismallin käyttäminen ennustamiseen.

Koneoppimiseen käytetään Python-ohjelmointikieltä ja Keras- sekä TensorFlow-kirjastoja. Google Analytics -datan tuomiseen käytetään Google Analytics Reporting -rajapintaa. Datan formatointiin käytettiin Pythonin kirjastoa nimeltä Pandas.

Opinnäytetyössä tuotetun prototyypin tarkoitus on luoda pohja tuotteelle, jota voi käyttää asiakaspoistumisen ennustamiseen. Toissijaisesti opinnäytetyössä tuodaan TensorFlow-osaamista toimeksiantajayritykseen.

2 KONEOPPIMINEN

Kone oppii, kun se muuttaa rakennettaan, ohjelmaansa tai dataansa tavalla, jolla sen odotettu suorituskyky paranee. Esimerkkinä puheentunnistusohjelma, joka tunnistaa puhetta paremmin sitä kuultuaan. Koneoppiminen on tärkeää, koska joitain ohjelmia on vaikea tai mahdoton ohjelmoida, ilman koneoppimismenetelmiä. (Nilsson. 2005, 1, 2.)

Koneoppiminen on rakentunut monen eri tieteenalan yhteistyön tuloksena. Eniten koneoppimisessa käytetään psykologiaa, tilasto- ja tietojenkäsittelytiedettä. (Nilsson. 2005, 3, 4.)

Koneoppimisen voi luokitella kolmeen kategoriaan, ohjattu oppiminen, ohjaamaton oppiminen ja vahvistusoppiminen. Eri koneoppimismetodit ovat hyviä ratkaisemaan erilaisia ongelmia. (Prateek. 2017, 30, 31, 385, 386.)

Ohjattu oppiminen tarkoittaa koneoppimismallia, jossa algoritmille annetaan opetusdataa, joka sisältää muuttujat ja millaisen tuloksen nämä muuttujat antavat. Kun algoritmi on datan avulla oppinut, miten muuttujat vaikuttavat tulokseen, se voi antaa ennusteita datasta, jolla on vain muuttujat ilman tulosta. (Prateek. 2017, 97, 98.)

Ohjaamattomassa oppimisessa mallille annetaan pelkästään muuttujat ilman tuloksia. Näin malli ei opi ennustamaan tuloksia muuttujiin liittyen, vaan enemmänkin, miten muuttujat ovat yhteydessä toisiinsa. Tällä metodilla voi esimerkiksi löytää datasta yhtäläisyyksiä, joita ihmiset eivät välttämättä huomaa. (Prateek. 2017, 97, 98.)

Valvottu oppiminen toimii eri tavoin kahdesta muusta kategoriasta. Tässä metodissa mallille ei anneta alussa minkäänlaista dataa. Mallin on tarkoitus tietynlaisessa ympäristössä tehdä erilaisia ennalta määrättyjä toimintoja. Toiminnot johtavat erilaisiin lopputuloksiin, jotka pisteytetään oppimisen yhteydessä. Malli ohjelmoidaan löytämään ketjutoimintoja, jotka tuovat parhaat mahdolliset pisteet. (Prateek. 2017, 385, 386.)

2.1 Tyypillisimmät koneoppimisen muodot

Koneoppimisen käyttö usein alkaa selvittämällä, mitä halutaan tietää. Kun ongelma on määritelty, valitaan siihen sopiva koneoppimismalli. Suurimman osan ongelmista voi ratkaista yleisimmillä koneoppimisen muodoilla. (Prateek. 2017, 54.)

2.1.1 Lineaarinen regressio

Lineaarinen regressio on yksi yleisimmistä ohjatussa oppimisessa käytetty malli. Malli vastaa kysymyksiin, joiden tulos on jatkuva luku. Kun tulos on jatkuva luku, mahdollisia vastauksia on äärettömästi. (Prateek. 2017, 54, 55.)

Lineaarisessa regressiossa oletetaan, että muuttujilla ja tuloksella on jonkinlainen yhteys toisiinsa. Joskus myös muuttujat korreloivat toisiensa kanssa, mutta lineaarisessa regressiossa. Perinteisessä data-analyysissa on usein tärkeä välttää muuttujia, jotka korreloivat keskenään. Tämän takia lineaarinen regressio on vahva työkalu. (Prateek. 2017, 54, 55.)

Lineaarisella regressiolla selvitetään, miten tulos muuttuu muuttamalla muuttujia. Lineaarisessa regressiossa muuttujien ja tuloksen suhde on lineaarinen. Kun muuttuja kasvaa, tulos kasvaa myös samassa suhteessa ylös tai alas päin. (Prateek. 2017, 54, 55.)

2.1.2 Klassifikaatio

Klassifikaatio kuuluu ohjattuun oppimiseen. Klassifikaatiossa tulosten määrä on ennalta määrätty luokka. Luokkien määrä voi olla mikä vain, mutta ei ikinä ääretön. Klassifikaatio toimii selvittämällä muuttujien ja tuloksen yhteyttä toisiinsa. Yleisin klassifikaatioon käytettävä malli on logistinen regressio. (Prateek. 2017, 31, 32, 37, 42.)

Logistinen regressio itsessään ei ole klassifikaatiotekniikka, mutta sen avulla usein tehdään klassifikaatiota. Logistisessa regressiossa tulos otetaan käyttämällä Sigmoid-käyrää, joka antaa todennäköisyyden tuloksen kuulumisesta tiettyyn luokkaan. Klassifikaatio harvoin ennustaa sadan prosentin varmuutta datapisteen kuulumisesta tiettyyn luokkaan. (Prateek. 2017, 31, 32, 37, 42.)

2.1.3 Klusterointi

Klusterointi on yksi suosituimmista ohjaamattomassa oppimisessa käytetty malli. Tekniikkaa käytetään datan analysoinnissa ja klustereiden löytämisessä. Klusterit ovat datapisteitä, jotka ovat samanlaisia keskenään. (Prateek. 2017, 98.)

Koska ohjaamattomassa oppimisessa ei ole oikeita tuloksia, klusterointi ei auta ennustamaan tietoja. Klusterointi auttaa ymmärtämään dataa ja löytämään siitä yhtäläisyyksiä, joita voisi muuten olla vaikea löytää. (Prateek. 2017, 99.)

2.2 Neuroverkot

Neuroverkot ovat koneoppimisen tyyli, joka mukailee ihmisten tapaa oppia uusia asioita. Neuroverkkoja voi käyttää kaikkien yleisien koneoppimismallien rakentamiseen. Neuroverkkojen etu on tarkemmat tulokset, mutta niiden kouluttaminen vaatii usein paljon laskentatehoa. (Prateek. 2017, 356, 357, 358.)

Ihmisten aivot toimivat hermosolujen avulla, jotka kerroksittain välittävät impulsseja eteenpäin. Keinotekoiset neuroverkot jäljittävät aivoja keinotekoisilla hermosoluilla eli neuroneilla. Jokainen keinotekoinen neuroni tekee päätöksen, lähettääkö tietoa eteenpäin vai ei. (Prateek. 2017, 356, 357, 358.)

3 ASIAKASPOISTUMA-ANALYYSI LIIKETOIMINNASSA

Koska uusien asiakkaiden saavuttaminen on kallista ja aikaa vievää, yrityksille on tärkeää asiakkaiden säilytys. Vaikka tavoitteena on aina asiakkaiden säilytys, osa asiakkaista tulee vaihtamaan palveluntarjoajaa. Asiakkaiden poistuminen yrityksen tarjoamasta palvelusta on erittäin kallista, varsinkin yrityksille, jotka eivät osaa ennustaa millaisissa tilanteissa asiakkaat lähtevät. (Kumar ym. 2012, 149.)

Kaikkein tehokkain tapa hallita asiakkaiden poistumista on selvittää syyt poistumiseen, pyrkiä ennustamaan poistumiset ja koittaa markkinoinnin avulla saada asiakkaita jäämään. Näin ollen yrityksillä on paljon kiinnostusta asiakaspoistuma-analyysille, jonka avulla selvitetään syitä ja ennustetaan asiakkaiden lähtemistä. (Kumar ym. 2012, 149.)

Asiakkaan aikomus lopettaa palvelun käyttö on binäärinen päätös, joten suosituin tapa analyysille on logistinen regressio. Mallissa voidaan hyödyntää tietoa asiakkaan käyttöhistoriasta ja selvittää sen avulla, kuinka todennäköisesti asiakas lopettaa yrityksen tarjoaman palvelun käytön. Ennustamiseen voi käyttää perinteisiä tilastotieteellistä tukimusta tai koneoppimista. (Kumar ym. 2012, 152.)

Analyysia varten tietoja valittaessa on tärkeää valita dataa, joka korreloi asiakkaiden poistumisen kanssa. Korrelaatioanalyysi on hyvä tehdä, ennen datan käyttöä koneoppimisessa ennustusten tekemiseen. (Kumar ym. 2012, 157, 158.)

4 GOOGLE ANALYTICS

Google Analytics on Googlen tarjoama työkalu verkkosivuston datan keräämiseen. Google Analytics toimii asettamalla pätkän JavaScript koodia verkkosivustolle, joka lähettää verkkosivuston dataa Googlen palvelimille. Dataa voi tarkastella Google Analyticsin raportointi näkymästä. (Medium 2017.)

Google Analytics kerää kahdenlaista dataa: käyttäjän hankintadataa ja käyttäjän käytöstapoihin liittyvää dataa. Käyttäjän hankintadata kertoo, mistä käyttäjät tulevat ja millaiseen väestöryhmään he kuuluvat. Käyttäjän käytöstapoihin liittyvä data kertoo, kuinka kauan käyttäjä on sivustolla ja mitä käyttäjä siellä tekee. (Medium 2017.)

4.1 Datan rakenne Google Analyticsissa

Jokainen Google Analyticsin raportti koostuu dimensioista ja metriikoista. Dimensiot ovat datan piirteitä. Esimerkki dimensiosta on kaupunki, jossa käyttäjä asuu. Metriikat ovat määrällisiä arvoja. Esimerkki metriikasta on tietyssä kaupungissa asuvien käyttäjien käyntikerrat sivustolla. (Google 2019a.)

Kaikkia metriikoita ja dimensioita ei voi yhdistää keskenään. Jokaisella dimensioilla ja metriikalla on oma tasonsa. Dimensiot ja metriikat voivat olla käyttäjä-, istunto- tai osumatasoisia. Näin ollen on järkevää yhdistää vain saman tason omaavia dimensioita ja metriikoita. (Google 2019a.)

Käyttäjätason datasta esimerkki dimensioita ovat käyttäjätyyppi, istuntojen määrä ja maantieteellinen sijainti, ja esimerkkimetriikoita ovat käyttäjät ja uudet käyttäjät. Istuntotason datasta esimerkkidimensioita ovat laskeutumissivu, liikenteen lähde / tulotapa, kampanjat, ja esimerkkimetriikoita ovat istunnot, välittömät poistumiset ja keskiverto istunnon pituus. Osumatason datasta esimerkkidimensioita ovat sivu, verkkoaseman tunnus ja tapahtumat, ja esimerkkimetriikoita ovat sivujen katselukerrat, sivulla vietetty aika ja tapahtumien määrät. (Bounteous 2016a.)

4.2 Google Analytics datan-vienti

Google Analytics sallii kaksi eri tapaa datan viemiseksi. Nämä vaihtoehdot ovat Google Analytics -raportointirajapinta ja Google Analytics 360 -asiakkaille BigQuery -linkitys. (Google 2019b.)

4.2.1 Google Analytics -raportointirajapinta

Google Analyticsista voi viedä dataa käyttämällä Googlen tarjoamaa raportointirajapintaa. Rajapinta on nimeltään Analytics Reporting AP v4. Rajapinnan avulla voi kysellä dataa Google Analyticsin palvelimilta ja tämän avulla siirtää dataa muihin applikaatioihin. (Google 2019b.)

Rajapinta toimii lähettämällä kyselyn, jossa pyydetään tiettyjä dimensioita tietyltä aikaväliltä, Google Analyticsin palvelimelle. Palvelin vastaa antamalla pyydetyn datan. Raportointirajapintaa voi käyttää palveluapplikaatiossa, asennetussa applikaatiossa tai nettiapplikaatiossa. (Google 2019b.)

4.2.2 Google Analytics BigQuery -linkitys

Google Analytics 360 sisältää ominaisuuden, jonka avulla datan vieminen Googlen BigQueryyn on helpompaa kuin raportointirajapinnan käyttö. BigQuery-viennissä ei tarvitse tehdä muuta kuin Google Analyticsin hallintapaneelista painaa linkitysnappia. Tämän jälkeen dataa kerääntyy automaattisesti BigQuery-palveluun, jossa sitä voi käyttää erilaisten applikaatioiden kanssa. (Google 2019c.)

4.3 Google Tag Manager

Google Tag Manager on käyttäjäystävällinen tuote, joka on tarkoitettu Javascript-pätkien lisäämiseen verkkosivustolle. Nämä Javascript-pätkät usein lisäävät markkinointitietoa sivulle tai lisäävät sivuston funktionaalisuutta. Google Analyticsin käyttö Google Tag Managerin avulla on helppoa. (Bounteous 2016b.)

Google Tag Manager tekee Javascriptsisällön lisäämisestä verkkosivustolle helppoa. Google Tag Managerin avulla sivuston lähdekoodia ei tarvitse muokata, kun sinne

haluaa lisätä Javascript-koodia. Google Tag Managerin käyttöliittymän avulla voi määritellä mitä koodia ajaa, missäkin tilanteissa. (Bounteous 2016b.)

Google Tag Manager koostuu kolmesta osasta: Tagien sisälle lisätään ajettava koodinpätkä. Triggereissä määritellään, missä tilanteissa koodi ajetaan. Muuttujan sisälle tallennetaan tietoa, jota tagit ja triggerit hyödyntävät. (Bounteous 2016b.)

Aiemmin sivustolle muutoksia tehdessä kaiken piti mennä sivuston kehittäjien kautta. Google Tag Managerin avulla kuka tahansa yrityksessä, jolla on tarvittavat oikeudet, voi tehdä muutoksia sivustolle. Tämä nopeuttaa huomattavasti sivustolle tehtävien muutosten nopeutta, sillä kaikkea ei tarvitse pyöräyttää kehittäjien kautta. (Bounteous 2016b.)

5 KÄYTÄNNÖN TOTEUTUS: KONEOPPIMINEN JA ASIAKASPOISTUMA-ANALYYSI

Tässä luvussa käsitellään, kuinka asiakaspoistuminen käytännössä ennustetaan. Työ alkoi pohdiskelusta, miten data pystytään tuomaan Google Analyticsiin. Kun datan tuominen oli määritelty, aloitettiin pohdinta, miten rakentaa ennustuksia tekevä neuroverkko.

Ensin suunnitelmana oli rakentaa takaisinkytkeytyvä neuroverkko, joka osaisi analysoida aikajanadataa. Tällainen koneoppimismalli osoittautui kuitenkin ongelmaan nähden ylimääräisen monimutkaiseksi.

Lopulta päädyttiin rakentamaan perinteinen tiheä neuroverkko. Tämän ratkaisun mahdollisti oivallus, että datan ei tarvitse olla aikajanallisessa muodossa. Ennustuksen pystyy tekemään opettamalla neuroverkolle, miten tietyn kuukauden data vaikuttaa siihen, ovatko käyttäjät seuraavassa kuussa lähteneet. Tällä tavalla neuroverkolla ei tarvitse olla ”ymmärrystä” asiasta.

5.1 Lähtötilanne

Projekti aloitettiin täysin tyhjältä pöydältä. Dataa ei kerääntynyt Google Analyticsiin oikeassa muodossa, eikä minkäänlaista koneoppimista ollut aloitettu. Tämän takia työn tekemiseen sai itse valita kaikki työkalut, ainoastaan datan oli tultava Google Analytcisista.

Projektin tavoite on tuoda yritykseen lisää koneoppimisosaamista ja, jos mahdollista kaupallistaa projektista syntyvä työkalu. Asiakaspoistuma-analyysi valittiin koneoppimisen kohteeksi, koska tämä tieto on universaalisti tärkeää kaikille yrityksille. Opinnäytetyössä selvitetään, onko asiakaspoistumisen ennustus mahdollista Google Analytics -datalla.

Ohjelmointikieleksi valittiin Python, koska Pythonille on kirjoitettu eniten data-analytiikkaan liittyviä kirjastoja. Python-ohjelmointikielen helppous auttaa keskittymään ohjelman suunnitteluun. Monimutkaisemmalla ohjelmointikielellä aikaa helposti kuluisi ohjelmaan liittymättömien ongelmien ratkaisemiseen, kuten tietokoneen muistin hallitsemiseen.

Datan lähteeksi valittiin Google Analytics, koska se on ilmainen teknologia. Google Analyticsiin on pääsy kaikilla yrityksillä. Opinnäytetyössä rakennetun ohjelman voi käytännössä ottaa käyttöön siis mikä tahansa yritys.

Koneoppimisteknologioiksi valittiin TensorFlow ja Keras. Teknologiat valittiin, koska ne ovat suosituimmat teknologiat neuroverkkojen luomiseen. Suosion ansiosta kirjastoja päivitetään usein ja ne saavat paljon uusi ominaisuuksia. Kyseiset teknologiat ovat myös yhteensopivia melkein kaikkien alustojen kanssa.

5.2 Datan tuominen Google Analyticsiin

Google Analytics ei automaattisesti kerää käyttäjäkohtaisesti dataa. Tätä varten oli luotava komentosarjat, jotka tuovat tiedon, kun käyttäjä kirjautuu tai rekisteröityy verkkosivustolle. Komentosarjat injektoitiin sivustolle Google Tag Managerin avulla.

Ensimmäinen komentosarja luo Javascript-olion, jota voi kutsua sivuston sisällä. Olion sisällä on 5 apufunktioita. Ensimmäinen funktio luo annetusta syötteestä SHA256 hajautuksen (Kuva 1).

```
sha256: function a(b) {
    function c(a, b) {
        return a >>> b | a << 32 - b
    }
    for (var d, e, f = Math.pow, g = f(2, 32), h = "length", i = "", j = [],
     k = 8 * b[h], l = a.h = a.h || [], m = a.k = a.k || [], n = m[h], o = {}, p = 2; 64 > n; p++)
        if (!o[p]) {
            for (d = 0; 313 > d; d += p) o[d] = p;
            l[n] = f(p, .5) * g | 0, m[n++] = f(p, 1 / 3) * g | 0
        } for (b += "\x80"; b[h] % 64 - 56;) b += "\x00";
    for (d = 0; d < b[h]; d++) {
        if (e = b.charCodeAt(d), e >> 8) return;
        j[d >> 2] |= e << (3 - d) % 4 * 8
    }
    for (j[j[h]] = k / g | 0, j[j[h]] = k, e = 0; e < j[h];) {
        var q = j.slice(e, e += 16),
            r = l;
        for (l = l.slice(0, 8), d = 0; 64 > d; d++) {
            var s = q[d - 15],
                t = q[d - 2],
                u = l[0],
                v = l[4],
                w = l[7] + (c(v, 6) ^ c(v, 11) ^ c(v, 25)) + (v & l[5] ^ ~v & l[6]) + m[d] + (q[d] = 16 > d ? q[d]
                  : q[d - 16] + (c(s, 7) ^ c(s, 18) ^ s >>> 3) + q[d - 7] + (c(t, 17) ^ c(t, 19) ^ t >>> 10) | 0),
                x = (c(u, 2) ^ c(u, 13) ^ c(u, 22)) + (u & l[1] ^ u & l[2] ^ l[1] & l[2]);
            l = [w + x | 0].concat(l), l[4] = l[4] + w | 0
        }
        for (d = 0; 8 > d; d++) l[d] = l[d] + r[d] | 0
    }
    for (d = 0; 8 > d; d++)
        for (e = 3; e + 1; e--) {
            var y = l[d] >> 8 * e & 255;
            i += (16 > y ? 0 : "") + y.toString(16)
        }
    return i
},
```

Kuva 1. SHA-256 hajautusalgoritmi.

Toinen ja kolmas funktio tallentaa käyttäjän sähköpostiosoitteen hajautettuna selaimen evästeeseen.

Kun käyttäjän hajautettu sähköpostiosoite on tallennettu evästeeseen, sen voi lähettää Google Analyticsiin käyttäjän tunnisteeksi. Google Analytics osaa yhdistää käyttäjän tunnisteen muuhun sivustolta lähtevään dataan. Täten saamme tietää, ketkä käyttäjistä tekivät sivustolla tiettyjä toimenpiteitä.

Datan tuomista ei muokattu muilla tavoilla, koska tavoitteena on ennustaa asiakaspoistumista Google Analyticsin oletusdatalla. Google Analytics mahdollistaa mukautetun datan keräämisen, mutta sitä ominaisuutta ei tässä työssä käytetty.

5.3 Datan tuominen Google Analyticsista

Google Analytics Reporting API mahdollistaa datan tuomisen Google Analyticsistä omaan sovellukseen. Rajapinta tarvitsee toimiakseen Google Cloud Consolesta haettavan tunnistustiedoston. Tiedosto tallennetaan JSON-muodossa ja tämän avulla rajapinta tunnistaa, kenen sovellus yrittää hakea dataa Google Analyticsistä.

JSON-tiedosto haettiin Google Cloud Consolesta ”API & Services” -palvelusta. Kyseisessä palvelussa voi luoda tunnisteen, jonka avulla Google sallii rajapintojensa käytön. Kun tunniste on luotu Google Cloud Console tarjoaa mahdollisuuden ladata JSON-tiedoston, jossa tunnisteen tiedot ovat Googlen rajapintojen ymmärrettävässä muodossa.

Google Analytics Reporting API tarjoaa Python-ohjelmointikielelle kirjaston, joka sisältää apufunktioita, joiden avulla rajapinnan käyttö on yksinkertaisempaa. Pythonilla kirjastojen lisääminen ohjelmaan on helppoa (Kuva 2).

```
import argparse
from apiclient.discovery import build
import httplib2
from oauth2client import client
from oauth2client import file
from oauth2client import tools
```

Kuva 2. Google Analytics Reporting API-kirjastojen tuonti Python-ohjelmaan.

Kun kirjastot on lisätty ohjelmaan, luodaan funktio (Kuva 3), joka luo instanssin analytics-oliosta. Analytics-oliota kuluttamalla, voidaan lähettää kyselyitä Google Analytics -palvelimille.

```
def initialize_analyticsreporting():
  """Initializes the analyticsreporting service object.

  Returns:
    analytics an authorized analyticsreporting service object.
  """
  # Parse command-line arguments.
  parser = argparse.ArgumentParser(
      formatter_class=argparse.RawDescriptionHelpFormatter,
      parents=[tools.argparser])
  flags = parser.parse_args([])

  # Set up a Flow object to be used if we need to authenticate.
  flow = client.flow_from_clientsecrets(
      CLIENT_SECRETS_PATH, scope=SCOPES,
      message=tools.message_if_missing(CLIENT_SECRETS_PATH))

  # Prepare credentials, and authorize HTTP object with them.
  # If the credentials don't exist or are invalid run through the native client
  # flow. The Storage object will ensure that if successful the good
  # credentials will get written back to a file.
  storage = file.Storage('analyticsreporting.dat')
  credentials = storage.get()
  if credentials is None or credentials.invalid:
    credentials = tools.run_flow(flow, storage, flags)
  http = credentials.authorize(http=httplib2.Http())

  # Build the service object.
  analytics = build('analytics', 'v4', http=http, discoveryServiceUrl=DISCOVERY_URI)

  return analytics
```

Kuva 3. Google Analytics Reporting API -olion luonnin apufunktio.

Toinen hyödyllinen apufunktio (Kuva 4) luo kyselyn oliolle ja palauttaa vastauksen.

```
def get_data(analytics, date_range, metrics, dimensions):
    return analytics.reports().batchGet(
      body={
        'reportRequests': [
        {
          'viewId': VIEW_ID,
          'dateRanges': date_range,
          'metrics': [{'expression': i} for i in metrics],
          'dimensions' : [{'name': j} for j in dimensions],
            'samplingLevel' : 'LARGE',
            'pageSize' : '10000'
        }]
      }
  ).execute()
```

Kuva 4. Google Analytics Reporting API -kyselyjen lähettämisen apufunktio.

Näiden kahden funktion avulla datan tuominen (Kuva 5) tapahtuu vain muutamalla rivillä koodia.

```
analytics = initialize_analyticsreporting()
feature_response = get_data(analytics, feature_date_range, feature_metrics, feature_dimensions)
label_response = get_data(analytics, label_date_range, label_metrics, label_dimensions)
features_df = convert_to_dataframe(feature_response)
labels_df = convert_to_dataframe(label_response)
```

Kuva 5. Datan tuominen apufunktioita käyttäen.

5.4 Datan formatointi oikeaan muotoon

Pythonilla on rakennettu datan käsittelyä varten kirjasto nimeltä Pandas. Tämän voi myös helposti tuoda (Kuva 6) omaan Python-ohjelmaan.

```
import pandas as pd
```

Kuva 6. Pandas-kirjaston tuominen Python-ohjelmaan.

Koska Google Analytics Reporting API palauttaa datan huonosti formatoidussa muodossa, rakennetaan vielä yksi apufunktio (Kuva 7). Tämä apufunktio ottaa parametrina Googlen rajapinnan vastauksen ja muuntaa sen datakehykseksi.

```
def convert_to_dataframe(response):

  for report in response.get('reports', []):
    columnHeader = report.get('columnHeader', {})
    dimensionHeaders = columnHeader.get('dimensions', [])
    metricHeaders = [i.get('name',{}) for i in columnHeader.get('metricHeader', {}).get('metricHeaderEntries', [])]
    finalRows = []


    for row in report.get('data', {}).get('rows', []):
      dimensions = row.get('dimensions', [])
      metrics = row.get('metrics', [])[0].get('values', {})
      rowObject = {}

      for header, dimension in zip(dimensionHeaders, dimensions):
        rowObject[header] = dimension


      for metricHeader, metric in zip(metricHeaders, metrics):
        rowObject[metricHeader] = metric

      finalRows.append(rowObject)


  dataFrameFormat = pd.DataFrame(finalRows)
  return dataFrameFormat
```

Kuva 7. Datakehyksen luomiseen käytettävä apufunktio.

Datan muuttaminen datakehykseksi mahdollistaa pandas-kirjaston apufunktioiden käytön.

Lopullinen datakokonaisuus koostuu kahdesta erillisestä datakehyksestä. Ensimmäinen datakehys (Taulukko 1) sisältää datan ominaisuudet.

ga:dimension1	ga:organicSearches	ga:productAddsToCart	ga:productRemovesFromCart	ga:sessions	ga:totalValue	ga:transactions
c8b0b00a0c8102319eef09f122d5947f73e4fb49...	0	0	0	1	0.00	0
bd0202a53453771770d89f684f2186fc250a1371...	0	147	1	4	642.26	4
eac9d074d93e8a364a7aeb6dffa6953ba006cab...	0	36	0	1	0.00	0
cbdae83e79843a02848d44bc9c16fe3de24e79a...	1	652	22	18	1512.12	12
bdd3ec55ac0233a7ea2396b1f7ffcd67be46d4be...	1	3	24	1	0.00	0

Taulukko 1. Datan ominaisuudet sisältävä datakehys.

Ominaisuuksien lisäksi kehyksessä on käyttäjän tunniste ja transaktioiden määrä. Transaktioiden määrää käytetään datan suodattamisessa. Jos käyttäjällä on nolla transaktiota, rivi poistetaan datasta. Käyttäjän tunnistetta käytetään hyväksi, kun datakehykset yhdistetään keskenään.

Ominaisuuksien datakehyksestä ensin poistetaan rivit (Kuva 8), joissa transaktioita on nolla.

```
features_df = features_df.apply(pd.to_numeric, errors = 'ignore')

remove_df = features_df
# Change transactions row to numeric
remove_df['ga:transactions'] = pd.to_numeric(remove_df['ga:transactions'])
# Remove rows with 0 transactions
remove_df = remove_df[remove_df['ga:transactions'] > 0]
features_df = remove_df
```

Kuva 8. Ominaisuuksien datakehyksen formatointi.

Kun datakehys on suodatettu, siitä voi poistaa transaktioiden määrän (Kuva 9) kertovan kolumnin.

```
features_df.drop('ga:transactions', axis=1)
```

Kuva 9. Kolumnin poistaminen ominaisuuksien datakehyksestä.

Toinen datakehys (Taulukko 2) sisältää tiedon, ovatko asiakkaat lopettaneet palvelun käytön seuraavana kuukautena.

ga:dimension1	ga:sessions	ga:transactions
00ae94f2f03a0ef2086e7a52034cec155e9d48d9e14ba7...	9	0
00bfbc503976e10ce0d6f30ee520f3548bf1db03029f84...	1	0
00da0d3116ccd4858a3e751f2c9f351ac674bb622ad9b6...	1	1
00e98ccbd88835d99301b735480ca0fd010952c4c858a7...	1	1
0174483f4b2f9c6be818e8b611028ab4542e78ef1fdc4d...	5	4

Taulukko 2. Datan tulokset sisältävä datakehys.

Jos transaktioita on nolla, voidaan olettaa käyttäjän lopettaneen palvelun käytön. Tätä datakehystä käytetään lopullisen datakokonaisuuden tuloksena.

Tulosten datakehyksestä tarvitsee muuttaa transaktioiden määrä binääriseksi (Kuva 10) tiedoksi. Eli jos transaktioita on enemmän kuin nolla, voidaan tieto merkitä luvulla yksi.

```
# Change transactions row to numeric
labels_df['ga:transactions'] = pd.to_numeric(labels_df['ga:transactions'])

# Change rows with more than 0 transactions to 1, for binary classification
labels_df.loc[labels_df['ga:transactions'] > 0, 'ga:transactions'] = 1
```

Kuva 10. Tulostedatan muuntaminen binäärimuotoon.

Lopuksi kehyksestä poistetaan sessioiden määrän tieto (Kuva 11).

```
# Drop sessions column, was only needed for getting rows with 0 transactions
labels_df = labels_df.drop('ga:sessions', axis=1)
```

Kuva 11. Kolumnin poistaminen tulosteiden datakehyksestä.

Formatoinnin jälkeen datakehykset voidaan yhdistää lopulliseksi datakokonaisuudeksi (Kuva 12).

```
dataset = features_df.merge(labels_df, how='left', on='ga:dimension1')
```

Kuva 12. Datakehysten yhdistäminen datakokonaisuudeksi.

Datakokonaisuudesta on vielä poistettava rivit, joissa on tyhjiä arvoja. Myös käyttäjien tunnisteita sisältävää kolumnia ei tarvita neuroverkolle syötettävässä datassa (Kuva 13).

```
# Remove nan columns
dataset = dataset.dropna()

# Remove user id
dataset = dataset.drop('ga:dimension1', axis=1)
```

Kuva 13. Datakokonaisuuden tarpeettomien tietojen poistaminen.

Vielä lopuksi on kannattavaa tasata rivien määriä (Kuva 14). Data usein sisältää enemmän rivejä, joissa asiakas ei ole lähtenyt palvelusta.

```
# Make data points more equal
if len(dataset.loc[dataset['churn'] == 0]) / len(dataset.loc[dataset['churn'] == 1]) < 1:
    dataset = dataset.drop(dataset.query('churn == 1').sample(frac=1-(len(dataset.loc[dataset['churn'] == 0]) / len(dataset.loc[dataset['churn'] == 1]))).index)
    print(len(dataset.loc[dataset['churn'] == 0]))
    print(len(dataset.loc[dataset['churn'] == 1]))
elif len(dataset.loc[dataset['churn'] == 1]) / len(dataset.loc[dataset['churn'] == 0]) < 1:
    dataset = dataset.drop(dataset.query('churn == 0').sample(frac=1-len(dataset.loc[dataset['churn'] == 1]) / len(dataset.loc[dataset['churn'] == 0])).index)
    print(len(dataset.loc[dataset['churn'] == 0]))
    print(len(dataset.loc[dataset['churn'] == 1]))
```

Kuva 14. Datakokonaisuuden rivien tasaus.

Epäsopusuhtaiset rivien määrät usein johtavat mallin ennustavan tuloksen, että kukaan käyttäjistä ei lähde palvelusta.

5.5 Koneoppimismallin rakentaminen

Neuroverkon rakentamiseen käytetään korkean abstraktion omaavaa Keras-kirjastoa. Keras-kirjastoa voi käyttää itsekseen omana kokonaisuutena tai sitä voi käyttää TensorFlow-kirjaston kanssa yhdessä. Tähän työhön valittiin TensorFlow-kirjaston ja Keras-kirjaston yhdistelmä. TensorFlow optimoi koneoppimisen nopeuden ja Keras tuo korkean ja helppokäyttöisen abstraktion. Molemmat kirjastot toimivat Pythonin kanssa (Kuva 15).

```
# Build model with keras (tf backend)
import tensorflow as tf
import numpy as np
from tensorflow.contrib.keras import backend as K
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.contrib.keras import models, layers, losses, optimizers, metrics
from sklearn.metrics import classification_report
```

Kuva 15. Koneoppimiskirjastojen tuonti Python-ohjelmaan.

Koneoppimiskirjastojen lisäksi Scikitlearn- ja Numpy-kirjastot tarjoavat hyödyllisiä apufunktioita datan tarkasteluun ja muokkaamiseen.

Korkean abstraktion omaavan Kerasin avulla itse mallin rakentaminen on melko suoraviivaista. Ensin määritellään hyperparametrit (Kuva 16). Hyperparametrien avulla määritellään koneoppimismallin käyttäytymistä.

```
# Constants
learning_rate = 0.003
# Substract 1 because user id in dimensions
input_dimensions = len(feature_metrics) + len(feature_dimensions) - 1
num_of_epochs = 1000
```

Kuva 16. Neuroverkon hyperparametrien määrittely.

Oppimisaste (learning rate) määrittelee, kuinka isoin askelin mallissa käytettävä optimointifunktio liikkuu. Suurella oppimisasteella koneoppimismalli oppii nopeasti, mutta se ei välttämättä ikinä saavuta hyviä tuloksia. Pienellä oppimisasteella koneoppimismalli oppii hitaasti, mutta malli antaa aina parhaan tuloksen, jonka malli pystyy antamaan. Oppimisasteen valitseminen ratkaisee, pystyykö koneoppimismalli ikinä löytämään vastauksia datasta.

Epookkien määrällä (number of epochs) valitaan, kuinka monta kertaa koneoppimismalli käy harjoitusdatan läpi. Epookkien määrä aina parantaa koneoppimismallin antamia ennusteita. Jos epookkeja on liikaa, koneoppimismalli ei ikinä saa oppimistapahtumaa valmiiksi. Epookkien määrän valitseminen riippuu siitä, kuinka paljon tehoa mallia pyörittävällä tietokoneella on.

Toiseksi määritellään neuroverkon syötteet ja niiden datatyypit (Kuva 17).

```
# Create variables
y_labels = dataset['churn']
X_features = dataset.drop('churn', axis=1)
num_cols = X_features.select_dtypes(['float64', 'int64']).columns
category_columns = X_features.select_dtypes(['object']).columns

try:
    X_features[category_columns] = X_features[category_columns].astype('category')
    X_features[category_columns] = X_features[category_columns].apply(lambda x: x.cat.codes)
except:
    print('No categorical columns')
```

Kuva 17. Neuroverkon syötteet ja datatyypit.

Koneoppimisen hyväksi huomattuihin tapoihin kuuluu datakokonaisuuden jakaminen kahteen osaan (Kuva 18). Ensimmäistä osaa käytetään mallin kouluttamiseen ja toista osaa hyödynnetään mallin toimivuuden testaamisessa. Neuroverkkoihin syötettävä numeraalinen data kannattaa myös skaalata siten, että kaikki datapisteet ovat yhden ja nollan välillä (Kuva 18).

```
# Create train test split
X_train, X_test, y_train, y_test = train_test_split(X_features, y_labels, test_size=0.33, random_state=42)

# Scale numerical features
scaler = MinMaxScaler()
X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols] = scaler.transform(X_test[num_cols])
scaled_x_train = np.array(X_train)
scaled_x_test = np.array(X_test)
X_test = np.array(X_test)
y_train = np.array(y_train)
y_test = np.array(y_test)
```

Kuva 18. Datakokonaisuuksien jakaminen ja skaalaus.

Skaalausta tehdään, jotta malli ottaisi kaikki ominaisuudet samassa mitoin huomioon. Jos ominaisuuksien skaalat ovat kovin eri kokoisia keskenään, neuroverkko helposti antaa ylimääräistä arvoa suuren koon omaaville ominaisuuksille. Molemmat apufunktiota ovat osa Scikitlearn-kirjastoa.

Itse mallin rakennus on abstraktion ansiosta vain muutama rivi koodia (Kuva 19).

```
# Create simple classification model
model = models.Sequential()
model.add(layers.Dense(units=15, input_dim=input_dimensions, activation='relu'))
model.add(layers.Dense(units=15, activation='relu'))
model.add(layers.Dense(units=15, activation='relu'))
model.add(layers.Dense(units=15, activation='relu'))
model.add(layers.Dense(units=1, activation='sigmoid'))

# Compile model
model.compile(optimizer=optimizers.Adam(lr=learning_rate),
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(scaled_x_train, y_train, epochs=num_of_epochs)
```

Kuva 19. Neuroverkon rakennus ja opetus.

Kerasin apufunktioiden avulla ensin määritellään neuroverkon tyypiksi peräkkäinen. Tämä tarkoittaa, että data liikkuu neuroverkossa yhden neuronikerroksen kerrallaan. Jokainen neuroverkon kerros on tiheä ja kaikissa on 15 neuronia. Poikkeuksena on viimeinen kerros, jossa on vain yksi neuroni. Viimeisen kerroksen tuloste on luku nollan ja yhden välillä. Tämä tuloste on ennustus siitä, mihin luokkaan syöte kuuluu.

Aktivaatiofunktiona käytetään neuronikerroksissa Rectified Linear Unit -funktiota. Kyseinen aktivaatiofunktio on uusimpia aktivaatiofunktioita, joita on kehitetty neuroverkkojen käyttöön. Se on erittäin tehokas aktivaatiofunktio.

Lopuksi neuroverkko koostetaan. Neuroverkko käyttää optimointifunktiona Adamfunktiota ja tappiofunktiona Binary Crossentropia. Koostamisen jälkeen neuroverkko oppii syötteen avulla (Kuva 19), mihin luokkiin eri asiakkaat kuuluvat, palvelusta poistuneet vai palveluun jääneet.

5.6 Koneoppimismallin ennusteiden raportointi

Kun neuroverkko on koostettu, sen pätevyys tulee testata. Testaus tapahtuu asettamalla neuroverkolle syötteeksi testaukseen luotu datakokonaisuus (Kuva 20). Neuroverkko pyrkii syötteen perusteella ennustamaan, mihin luokkaan asiakas kuuluu. Kun neuroverkko on saanut ennustukset tehtyä, tuloksia verrataan aiemmin luotuun datakokonaisuuteen, jossa ovat oikeat tulokset (Kuva 21).

```
# Test the model
predictions = model.predict_classes(scaled_x_test)
#print('X: {} \n Prediction: {}, Real: {}'.format(X_test, predictions, y_test))
for i in range(len(predictions)):
    print('X: {} \n Prediction: {}, Real: {}'.format(X_test[i], predictions[i], y_test[i]))
# 1 means not churned, 0 means churned
```

Kuva 20. Neuroverkon ennusteiden luonti ja niiden testaus.

```
X: [0.          0.01753702 0.00693241 0.00795756 0.02213042 0.        ]
 Prediction: [0], Real: 0.0
X: [0.125       0.03468433 0.03292894 0.03183024 0.21987331 0.17142857]
 Prediction: [1], Real: 1.0
X: [0.          0.23226812 0.08145581 0.13262599 0.24713951 0.2       ]
 Prediction: [1], Real: 1.0
X: [0.125       0.0872954  0.0034662  0.02917772 0.22215713 0.14285714]
 Prediction: [0], Real: 1.0
X: [0.          0.04949337 0.00866551 0.0397878  0.05007918 0.05714286]
 Prediction: [0], Real: 1.0
X: [0.          0.12042089 0.10051993 0.04244032 0.27171684 0.31428571]
 Prediction: [1], Real: 1.0
X: [0.          0.00623539 0.0017331  0.01061008 0.03314965 0.        ]
 Prediction: [0], Real: 0.0
X: [0.          0.07521434 0.14731369 0.04774536 0.04840636 0.02857143]
 Prediction: [0], Real: 1.0
```

Kuva 21. Neuroverkon ennusteiden vertaus oikeisiin tulosteisiin.

Testaamiseen voi myös käyttää scikitlearnin mukana tulevaa apufunktiota nimeltä ”classification_report” (Kuva 22). Tämä raportti palauttaa tiedon, kuinka hyvin malli suoriutui keskimäärin (Kuva 23).

```
print(classification_report(predictions,y_test))
```

Kuva 22. Klassifikaatio raportinluonti.

	precision	recall	f1-score	support
0	0.74	0.66	0.70	123
1	0.58	0.67	0.62	87
micro avg	0.66	0.66	0.66	210
macro avg	0.66	0.66	0.66	210
weighted avg	0.67	0.66	0.66	210

Kuva 23. Klassifikaatioraportti.

Mallia voidaan pyytää ennustamaan tuloksia uudesta datasta aiemmin käytetyllä ” predict_classes” -funktiolla.

6 JOHTOPÄÄTÖKSET JA POHDINTA

Opinnäytetyön tavoitteena oli tehdä soveltuvuusselvitys. asiakaspoistuma-analyysin tekemisestä koneoppimista ja Google Analytics -dataa hyödyntäen, sekä tuoda TensorFlow- ja koneoppimisosaamista yritykseen.

Neuroverkko pystyi ennustamaan asiakkaat, jotka tulevat poistumaan palvelusta 70 prosentin tarkkuudella. Tulos on erinomainen, koska arvailulla ennustusvarmuus olisi noin 50 prosenttia. Malli osaa siis ennustaa arvailua 20 prosenttiyksikköä paremmin. On uskottavaa, että mukautettua dataa käyttämällä neuroverkko pystyisi ennustamaan asiakkaiden palvelusta lähtemisen vielä tarkemmin. Tämä osoittaa, että neuroverkkojen käyttö ennustukseen on mahdollista.

Opinnäytetyötä aloittaessani koneoppiminen oli minulle jo tuttu käsite ja olin TensorFlow:n kanssa muutamia harjoituksia tehnyt. Työn aikana osaamiseni ja ymmärrykseni aiheesta kuitenkin kasvoi. Opinnäytetyön tehtyäni, koen olevani osaava koneoppimismallien rakentamisessa.

Tällä hetkellä neuroverkosta on rakennettu ainoastaan prototyyppi. Prototyyppi kuitenkin mahdollistaa mallin käyttämistä suuremmassa kaavassa.

Ohjelmaa voi automatisoida luomalla datan funktioita, jotka formatoivat dataa yleismaallisemmin. Tällä hetkellä dataa formatoidaan ilman yleisfunktioita. Automatisaatio Google Cloud -pilvipalveluissa noudattaisi arkkitehtuuria, jossa data tuodaan omana palveluna Google Cloudin Big Queryyn. Big Querysta data siirrettäisiin koneoppimismallia pyörittävän palvelun rajapintaosoitteeseen.

Tulevaisuudessa projektia voidaan vielä jatkaa automatisoimalla datan tuomisen koneoppimismallille ja tuomalla koneoppimismallin pilvipalveluun, jotta sen voi tuotteistaa.

LÄHTEET

Bounteous 2016a. Understanding Scope In Google Analytics Reporting. Viitattu 21.6.2019 https://www.bounteous.com/insights/2016/11/30/understanding-scope-google-analytics-reporting/

Bounteous 2016b. What Is Google Tag Manager (And How Does It Work With Google Analytics). Viitattu 10.9.2019 https://www.bounteous.com/insights/2016/02/15/what-google-tag-manager-and-how-does-it-work-google-analytics/

Google 2019a. Dimensions and metrics. Viitattu 21.6.2019 https://support.google.com/analytics/answer/1033861?hl=en

Google 2019b. Reporting API v4 Overview. Viitattu 4.7.2019 https://developers.google.com/analytics/devguides/reporting/core/v4/

Google 2019c. Se up BigQuery Export. Viitattu 4.7.2019 https://support.google.com/analytics/answer/3416092

Keras 2019. Keras. Viitattu 12.11.2019 https://keras.io/

Kumar, V. & Andrew P. 2012. Statistical Methods in Customer Relationship Management. 1. painos. John Wiley & Sons, Incorporated.

Medium 2017. What is Google Analytics and why is it important for my business. Viitattu 19.6.2019 https://medium.com/analytics-for-humans/what-is-google-analytics-and-why-is-it-important-to-my-business-8c083a9f81be

Nilsson, N. 2005. Introduction To Machine Learning. 1. painos. Stanford University.

Numpy 2019. Numpy. Viitattu 12.11.2019 https://numpy.org/

Pandas 2019. Pandas. Viitattu 12.11.2019 https://pandas.pydata.org/

Prateek, J. 2017. Artificial Intelligence With Python. 1. painos. Packt Publishing Ltd.

Tensorflow 2019. Tensorflow. Vittattu 12.11.2019 https://www.tensorflow.org/

Lähdekoodi

```
import argparse

from apiclient.discovery import build

import httplib2

from oauth2client import client

from oauth2client import file

from oauth2client import tools

import pandas as pd




def initialize_analyticsreporting():

  """Initializes the analyticsreporting service object.


  Returns:

    analytics an authorized analyticsreporting service object.

  """

  # Parse command-line arguments.

  parser = argparse.ArgumentParser(

      formatter_class=argparse.RawDescriptionHelpFormatter,

      parents=[tools.argparser])

  flags = parser.parse_args([])
```

```
  # Set up a Flow object to be used if we need to authenticate.
  flow = client.flow_from_clientsecrets(
      CLIENT_SECRETS_PATH, scope=SCOPES,
      message=tools.message_if_missing(CLIENT_SECRETS_PATH))

  # Prepare credentials, and authorize HTTP object with them.
  # If the credentials don't exist or are invalid run through the native client
  # flow. The Storage object will ensure that if successful the good
  # credentials will get written back to a file.
  storage = file.Storage('analyticsreporting.dat')
  credentials = storage.get()
  if credentials is None or credentials.invalid:
    credentials = tools.run_flow(flow, storage, flags)
  http = credentials.authorize(http=httplib2.Http())

  # Build the service object.
  analytics = build('analytics', 'v4', http=http, discoveryServiceUrl=DISCOVERY_URI)

  return analytics


def get_data(analytics, date_range, metrics, dimensions):
    return analytics.reports().batchGet(
```

```
      body={
        'reportRequests': [
        {
          'viewId': VIEW_ID,
          'dateRanges': date_range,
          'metrics': [{'expression': i} for i in metrics],
          'dimensions' : [{'name': j} for j in dimensions],
            'samplingLevel' : 'LARGE',
            'pageSize' : '10000'
        }]
      }
  ).execute()



def convert_to_dataframe(response):


  for report in response.get('reports', []):
    columnHeader = report.get('columnHeader', {})
    dimensionHeaders = columnHeader.get('dimensions', [])
    metricHeaders = [i.get('name',{}) for i in columnHeader.get('metricHeader', {}).get('metricHeaderEntries', [])]
    finalRows = []
```

```
    for row in report.get('data', {}).get('rows', []):
      dimensions = row.get('dimensions', [])
      metrics = row.get('metrics', [])[0].get('values', {})
      rowObject = {}

      for header, dimension in zip(dimensionHeaders, dimensions):
        rowObject[header] = dimension


      for metricHeader, metric in zip(metricHeaders, metrics):
        rowObject[metricHeader] = metric

      finalRows.append(rowObject)


  dataFrameFormat = pd.DataFrame(finalRows)
  return dataFrameFormat


SCOPES = ['https://www.googleapis.com/auth/analytics.readonly']
DISCOVERY_URI = ('https://analyticsreporting.googleapis.com/$discovery/rest')
CLIENT_SECRETS_PATH = 'client_secret_720354918110-tjd8qbmkqs44o2usiqnavjc89g7bfj17.apps.googleusercontent.com.json' # Path to client_secrets.json file.
```

```
VIEW_ID = '127826015'



feature_date_range = [{'startDate': '2018-05-01', 'endDate': '2019-06-01'}]

# Get nro of transactions from the next 30 days to determine churns

label_date_range = [{'startDate': '2019-06-02', 'endDate': '2019-07-01'}]


feature_metrics = ['ga:sessions', 'ga:productAddsToCart',
'ga:productRemovesFromCart',

                   'ga:transactions',

                   'ga:organicSearches',

                  'ga:totalValue']

# also take sessions to get data with 0 transactions

label_metrics = ['ga:sessions', 'ga:transactions']



feature_dimensions = ['ga:dimension1']

label_dimensions = ['ga:dimension1']



analytics = initialize_analyticsreporting()
```

```
feature_response = get_data(analytics, feature_date_range, feature_metrics, feature_dimensions)

label_response = get_data(analytics, label_date_range, label_metrics, label_dimensions)

features_df = convert_to_dataframe(feature_response)

labels_df = convert_to_dataframe(label_response)


features_df = features_df.apply(pd.to_numeric, errors = 'ignore')


remove_df = features_df

# Change transactions row to numeric

remove_df['ga:transactions'] = pd.to_numeric(remove_df['ga:transactions'])

# Remove rows with 0 transactions

remove_df = remove_df[remove_df['ga:transactions'] > 0]

features_df = remove_df


features_df.drop('ga:transactions', axis=1)


# Change transactions row to numeric

labels_df['ga:transactions'] = pd.to_numeric(labels_df['ga:transactions'])


# Change rows with more than 0 transactions to 1, for binary classification

labels_df.loc[labels_df['ga:transactions'] > 0, 'ga:transactions'] = 1
```

```
# Drop sessions column, was only needed for getting rows with 0 transactions
labels_df = labels_df.drop('ga:sessions', axis=1)



# Rename columns
labels_df.columns = ['ga:dimension1', 'churn']

dataset = features_df.merge(labels_df, how='left', on='ga:dimension1')

# Remove nan columns
dataset = dataset.dropna()

# Remove user id
dataset = dataset.drop('ga:dimension1', axis=1)



dataset.to_csv('data.csv', sep=',', encoding='utf-8')

# Make data points more equal
if len(dataset.loc[dataset['churn'] == 0]) / len(dataset.loc[dataset['churn'] == 1]) < 1:
    dataset = dataset.drop(dataset.query('churn == 1').sample(frac=1-(len(dataset.loc[dataset['churn'] == 0]) / len(dataset.loc[dataset['churn'] == 1]))).index)
    print(len(dataset.loc[dataset['churn'] == 0]))
    print(len(dataset.loc[dataset['churn'] == 1]))
```

```
elif len(dataset.loc[dataset['churn'] == 1]) / len(dataset.loc[dataset['churn'] == 0]) < 1:
    dataset = dataset.drop(dataset.query('churn == 0').sample(frac=1-len(dataset.loc[dataset['churn'] == 1]) / len(dataset.loc[dataset['churn'] == 0])).index)
    print(len(dataset.loc[dataset['churn'] == 0]))
    print(len(dataset.loc[dataset['churn'] == 1]))



# Build model with keras (tf backend)
import tensorflow as tf
import numpy as np
from tensorflow.contrib.keras import backend as K
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.contrib.keras import models, layers, losses, optimizers, metrics
from sklearn.metrics import classification_report



# Constants
learning_rate = 0.003
# Substract 1 because user id in dimensions
input_dimensions = len(feature_metrics) + len(feature_dimensions) - 1
num_of_epochs = 1000


# Create variables
```

```
y_labels = dataset['churn']

X_features = dataset.drop('churn', axis=1)

num_cols = X_features.select_dtypes(['float64', 'int64']).columns

category_columns = X_features.select_dtypes(['object']).columns


try:

    X_features[category_columns] = X_features[category_columns].astype('category')

    X_features[category_columns] = X_features[category_columns].apply(lambda x: x.cat.codes)

except:

    print('No categorical columns')


# Create train test split

X_train, X_test, y_train, y_test = train_test_split(X_features, y_labels, test_size=0.33, random_state=42)


# Scale numerical features

scaler = MinMaxScaler()

X_train[num_cols] = scaler.fit_transform(X_train[num_cols])

X_test[num_cols] = scaler.transform(X_test[num_cols])

scaled_x_train = np.array(X_train)

scaled_x_test = np.array(X_test)

X_test = np.array(X_test)

y_train = np.array(y_train)
```

```
y_test = np.array(y_test)


# Create classification model
model = models.Sequential()
model.add(layers.Dense(units=15, input_dim=input_dimensions, activation='relu'))
model.add(layers.Dense(units=15, activation='relu'))
model.add(layers.Dense(units=15, activation='relu'))
model.add(layers.Dense(units=15, activation='relu'))
model.add(layers.Dense(units=1, activation='sigmoid'))


# Compile model
model.compile(optimizer=optimizers.Adam(lr=learning_rate),
              loss='binary_crossentropy',
              metrics=['accuracy'])
# Train the model
model.fit(scaled_x_train, y_train, epochs=num_of_epochs)


# Test the model
predictions = model.predict_classes(scaled_x_test)
#print('X: {} \n Prediction: {}, Real: {}'.format(X_test, predictions, y_test))
for i in range(len(predictions)):
    print('X: {} \n Prediction: {}, Real: {}'.format(X_test[i], predictions[i], y_test[i]))
# 1 means not churned, 0 means churned
```

```
print(classification_report(predictions,y_test))
```