KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

Mika Salmi

KONEOPPIMISEN MAHDOLLISUUDET LEMONSOFT OY:SSA

Opinnäytetyö
Helmikuu 2020

Karelia AMMATTIKORKEAKOULU	**OPINNÄYTETYÖ** **Joulukuu 2019** **Tietojenkäsittelyn koulutusohjelma** Tikkarinne 9 80200 JOENSUU +358 13 260 600 (vaihde)

Tekijä(t)
Mika Salmi

Nimeke
Koneoppimisen mahdollisuudet Lemonsoft Oy:ssa

Toimeksiantaja
Lemonsoft Oy

Tiivistelmä

Opinnäytetyön aiheena oli koneoppiminen. Toimeksiantajana on Lemonsoft Oy, joka on ottamassa koneoppimista osaksi toiminnanohjausjärjestelmäänsä. Tavoitteena oli tutustua koneoppimisen historiaan ja tulevaisuudennäkymiin ja tutkia, miten toimeksiantaja voisi mahdollisesti hyödyntää koneoppimista ohjelmistossaan. Opinnäytetyössä tarkasteltiin myös Gartnerin hypekäyrää ja otettiin selvää, millaisina ilmiöinä koneoppiminen ja sen eri muodot näyttäytyvät käyrällä.

Opinnäytetyössä tutustuttiin myös erilaisiin koneoppimisen sovelluskehyksiin. Näitä ovat esimerkiksi Amazon Web Services sekä heidän tarjoamansa Amazon Sagemaker, Microsoft Azure Machine Learning Services ja ML.NET. Sovelluskehyksiä vertailtiin ohjelmistokehittäjän näkökulmasta ja arvioitiin, täyttääkö mikään niistä Lemonsoft Oy:n tarpeita.

Sovelluskehyksien vertailun tuloksena päädyttiin, että ML.NET on paras valinta yritykselle. Tämän vuoksi työssä keskityttiin tutkimaan Microsoftin kehittämää ML.NET-sovelluskehystä kahden eri demon avulla. Demot toteutettiin Visual Studiolla käyttämällä C#-ohjelmointikieltä. Testauksessa käytettiin apuna ML.NET-sivuston tarjoamaa materiaalia.

Kieli suomi	Sivuja 39

Asiasanat
Koneoppiminen, ML.NET, Visual Studio, Model Builder

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
December 2019
Business Information Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600 (switchboard)

Author (s)
Mika Salmi

Title
Possibilities of Machine Learning at Lemonsoft Ltd

Commissioned by
Lemonsoft Ltd

Abstract
The commissioner, Lemonsoft Ltd, is going to incorporate machine learning into their enterprise resource planning system, so in this thesis the goal was to explore machine learning and its possibilities. The goals of this study was to study the history of machine learning and ponder about its future, learn how the client could potentially utilize it in their software, and examine the Gartner hypegraph to see how Machine learning is generally viewed in the industry.

This study involved three different machine learning frameworks: Amazon Web Services and their Amazon Sagemaker, Microsoft Azure Machine Learning Services and ML.NET by Microsoft. The data was compared between these frameworks from a software developer's point of view and considered whether any of them fulfils the client's needs.
It was concluded that the most suitable machine learning framework for Lemonsoft Oy would be ML.NET so this thesis focuses especially on ML.NET by developing two different demos using Visual Studio and C# programming language. The project was carried out using materials provided by the ML.NET website.

Language
Finnish

Pages 39

Keywords
Machine Learning, ML.NET, Visual Studio, Model Builder

Sisältö

1 Johdanto

Tämän opinnäytetyön tarkoituksena on kertoa, mitä koneoppiminen on, käydä lyhyesti läpi koneoppimisen historiaa ja tulevaisuuden näkymiä, kertoa eri koneoppimisen viitekehyksistä ja vertailla niitä keskenään. Koneoppimisesta aiheena olisi voinut tehdä hyvinkin laajan kokonaisuuden, mutta päädyin rajaamaan koneoppimisen viitekehyksien vertailun kolmeen viitekehykseen, jotka ovat ML.NET, Amazon Sagemaker ja Azure Machine Learning Service.

Osana opinnäytetyötä teen myös kaksi demoa koneoppimisen käytöstä, joiden tulokset raportoin opinnäytetyöhön. Testit suoritan käyttämällä ML.NET-viitekehystä Visual Studion Community editionilla ja käyttämällä C#-ohjelmointikieltä. Demojen materiaali löytyy Microsoftin omalta ML.NET-sivustolta, jossa ohjelmoijat ja muut asiasta kiinnostuneet voivat testata koneoppimista. Opinnäytetyön lopussa pohdin demojen tuloksia ja millaiselta koneoppiminen vaikuttaa. Pohdin myös, olisiko siitä mitään hyötyä Lemonsoft Oy:n ohjelmistossa. Opinnäytetyön aihe tuli Lemonsoft Oy:n tarpeista saada tietoa koneoppimisesta, sillä Lemonsoft Oy on ottamassa käyttöönsä koneoppimista osana sen kehittämää toiminnanohjausjärjestelmää.

2 Mitä on koneoppiminen?

Koneoppiminen on tekoälyn osa-alue, jossa koneelle ei erikseen määritetä ohjeita, joiden mukaan sen pitäisi toimia erilaisissa tilanteissa, vaan koneen on tarkoituksena oppia itsenäisesti datasta. Tällä tavalla saadaan tehtyä huomattavasti kustannustehokkaampia ratkaisuja kuin silloin jos jokaista mahdollista tilannetta varten ohjelmoidaan oma ratkaisu. Yleisenä sääntönä voidaan pitää, että mitä enemmän laadukasta dataa ohjelma saa opeteltavaksi, sitä varmempi toimintavarmuus sillä on. (Expert System, 2020)

Koneoppiminen on yleistynyt yrityksien käytössä, ja siitä on tullut osa arkea hyvin suurelle osalle ihmisiä. Aina ei voida havaita, mitä kaikkea koneoppimisella on tehty, vaikka sitä käytetäänkin usein. Näitä käyttökohteita ovat esimerkiksi hakukoneet, ennakoiva tekstinsyöttö ja roskapostin suodatin. Myös Netflixin ja YouTuben suositukset ovat koneoppimisen kautta tulevia ehdotuksia. (Bean 2018.)

2.1 Miten koneet oppivat?

Tietokoneet voivat oppia usealla tavalla. Kirjassaan Neitteenmäki ja Siukonen kertovat tavoista, joilla tiekone oppii materiaalistaan seuraavanlaisesti.

> Tietokoneet voivat oppia kolmella tavalla: ohjaamattomasti, ohjatusti ja vahvistetusti. Ohjelmistojen avulla ne kykenevät havaitsemaan ympäristöä, tekemään johtopäätöksiä ja tallentamaan kaiken muistiin. Aineistoja tutkittuaan ne oppivat asioita, pieniä ja isoja, vaikkapa pelaamaan seurapelejä, seuraamaan ilmastonmuutosta, säätämään kiinteistöjen lämmitys-, ilmastointi- ja sähköjärjestelmiä. Koneita voidaan opettaa luonnonlakien, esimerkiksi eläinten käyttäytymisen perusteella. (Neittaanmäki & Siukonen 2019, 80.)

Ohjatussa oppimisessa koneelle annetaan tarpeeksi dataa, jossa on oikea vastaus. Ohjelma opettelee saadun datan perusteella ennustamaan oikeaa tulosta, jos lopputulosta ei tiedetä. Ohjelma voi ennustaa esimerkiksi, miten todennäköisesti asiakas maksaa laskun ajallaan. Ennuste saadaan, kun ohjelmalle annetaan asiakkaan tietoja kuten miten asiakas on aikaisemmin maksanut laskunsa. Tällöin ohjelma saa syötteen ja tuloksen, joita opettelemalla ohjelma osaa tehdä arvioita. Ohjelma saadaan toimimaan paremmin, nopeammin ja varmemmin, kun sille annetaan tarpeeksi määrällisesti ja laadukasta dataa käytettäväksi. (Brownlee 2016.)

”Ohjaamattomassa oppimisessa kone päättelee asioita datassa olevien säännönmukaisuuksien ja suhteiden pohjalta” (Merilehto 2018, 19). Ohjelmalle

annetaan dataa tutkittavaksi, mutta syötteen lisäksi ohjelmalle ei anneta valmista vastausta. Ohjelma opettelee saadun tiedon perusteella muodostamaan erilaisia ryhmiä ja sääntöjä. Algoritmi tutkiikin erilaisia kaavoja, jotka voivat liittyä, vaikka asiakkaan ostokäyttäytymisen tutkimiseen. Algoritmin avulla voidaan tutkia, miten usein asiakas, joka on ostanut tuotteen x, ostaa myös tuotteen. Tässäkin opetustavassa datan määrä ja laatu on tärkeä osa laadukkaan tuloksen saamiseksi. (Merilehto 2018.)

Vahvistetussa oppimisessa koneelle ei anneta valmiita vastauksia, vaan sille annetaan palautetta sen perusteella, miten hyvin kone on toiminut erilaisissa tilanteissa. Technopedian artikkelissa kuvataan vahvistetun oppimisen prosessia seuraavanlaisesti: vahvistetussa oppimisessa kone oppii olemalla vuorovaikutuksessa ympäristönsä kanssa, jolloin koneelle annetaan palkintoja oikeasta suorituksesta ja rangaistuksia väärästä suorituksesta Tällainen tilanne voi olla vaikkapa shakkipeli, jossa ohjelmaa opetetaan pääsemään tiettyyn pisteeseen mahdollisimman nopeasti ja tehokkaasti. Ohjelma saa palautetta, miten se suoriutui työstä ja muuttaa palautteen perusteella työskentelyään. (Technopedia 2019).

2.2 Koneoppimisen mallit

Jotta kone oppii, luodaan malli valitsemalla sopiva algoritmi, jonka perusteella se oppii (Merilehto 2018.) Koneoppimisessa on useita malleja, joista olen käynyt läpi viisi, jotka ovat ryhmittely, luokittelu, regressio, suosittelu ja poikkeamien etsiminen.

Ryhmittely (engl. clustering) on valvomatonta opetusta. Ryhmittelyssä annetaan koneelle dataa, jota ei ole luokiteltu. Kone analysoi datan ja tunnistaa siitä erilaisia ryhmiä, joiden mukaan se jaottelee saadun datan. Ryhmittelyä voidaan käyttää markkinointiin tunnistamaan erilaisia asiakasryhmiä ja heidän ominaisuuksiaan, joita voidaan hyödyntää markkinoinnissa. (Priya 2019.)

Luokittelu (engl. classification) on valvottua oppimista. Sidanan Mediumissa julkaistussa artikkelissa luokittelu selitetään vapaamuotoisesti kääntäen näin: luokittelu on valvottua oppimista, jossa kone oppii sille annetusta datasta. Syötetty data voi olla luokiteltuna kahteen tai useampaan kategoriaan. Tätä mallia voidaan käyttää, jos halutaan tunnistaa kuvista esineitä, eläimiä, ihmisiä tai sähköposti roskapostista sekä puheen ja kirjoittamisen tunnistamiseen. (Sidana 2017.)

Regressio (engl. regression) on valvotun oppimisen malli. Gupta kuvailee regressiota seuraavasti GeeksforGeeksin artikkelissaan: Regressio on valvotun oppimisen malli, jossa kone oppii sille annetusta datasta. Regressio malleja on useampiakin, joilla voidaan suorittaa monimutkaisiakin tehtäviä. (Gupta 2019.) Mediumin artikkelissaan Dave kuvailee tämän mallin sopivan erityisen hyvin hintojen ennustamiseen erilaisten muuttujien avulla. Esimerkkinä voidaan käyttää talon hinnan ennustamista sen ominaisuuksien mukaan, kuten sijainti, rakennusvuosi, koko ja niin edelleen. (Dave 2018.)

Suosittelu (engl. recommendation) on valvotun oppimisen malli. Artikkelissaan Mediumissa Kordík selittää suosittelusta seuravanlaisesti: suosittelussa koneelle annetaan dataa, josta se tutkii ja etsii yhtäläisyyksiä muun opetellun datan kanssa. Tätä voidaan soveltaa esimerkiksi nettikaupoissa, joissa voidaan suositella tuotetta, jonka joku muu on ostanut ostamasi tuotteen kanssa. Tätä käytetään myös paljon sellaisissa palveluissa kuin Netflix, YouTube ja Spotify. (Kordík 2018.)

Poikkeamien etsiminen (engl. anomaly detection) voi olla valvottua tai valvomatonta oppimista. Gupta kertoo artikkelissaan poikkeamien etsimisestä, joka voidaan tehdä joko valvottuna tai valvomattomana koneoppimisena. Koneelle annetaan dataa tutkittavaksi ja se muodostaa kaavoja datasta, joiden perusteella se löytää poikkeavat kohdat. Näitä voidaan käyttää esimerkiksi luottokorttipetosten tai kyberhyökkäyksen huomaamiseen. (Gupta 2019.)

Koneoppimisen avulla saadut ratkaisut ja tulokset ovat hyviä ja useammalla yrityksellä onkin jo koneoppiminen jollain tavalla käytössä. On kuitenkin muistettava, että kone antaa oppimansa tiedon avulla ehdotuksia, jotka eivät ole 100 % varmoja vaan loppujen lopuksi tietojen tarkastaminen jää aina käyttäjän vastuulle. (Kozyrkov 2018.)

3 Koneoppimisen historia

Koneoppiminen alkoi kehittymään harppauksin 1940-luvun jälkeen mutta koneoppimisen käyttämät matemaattiset säännöt oli keksitty jo huomattavasti aikaisemmi. Matemaattisista säännöistä koneoppimisessa on käytössä esimerkiksi Thomas Bayesin kehittämä Bayesin teoraama vuodelta 1812 ja Andrei Markovin kehittämä Markovin ketjut vuodelta 1913. (BBC 2019.)

Vuonna 1943 Warren McCulloch ja Walter Pitts kirjoittivat esseen neuroneista ja miten ne toimivat. He päättivät luoda mallin käyttämällä sähkökaaviota ja näin syntyi neuroverkko. Vuonna 1950 Alan Turing julkaisi artikkelin Computing machinery and intelligence, jossa hän kehitti kuuluisan Turingin testin. Testi on yksinkertainen ja siinä tietokoneen tarkoituksena on vakuuttaa ihminen, että se on ihminen eikä tietokone. Vuonna 1951 Marvin Minsky ja Dean Edmonds rakensivat ensimmäisen keinotekoisen neuroverkon. 1950- ja 1960-luvuilla oltiin todella innostuneita tekoälyä kohtaan, mutta kiinnostus alkoi lopahtamaan, koska suuria läpimurtoja ei juurikaan tullut. Tämän myötä rahoituksen määrä väheni tutkimuksilta. (BBC 2019.)

Vuonna 1982 kiinnostus neuroverkkoja kohtaan kasvoi uudelleen, kun John Hopfield ehdotti tietoverkon luomista, joka olisi kaksisuuntainen kuten neuronitkin oikeasti ovat. Tämän lisäksi Japanissa aloiteltiin tutkimaan kehittyneempiä neuroverkkoja, joka sai amerikkalaiset sijoittajat laittamaan lisää rahaa neuroverkkojen tutkimiseen. (BBC 2019.)

Vuonna 1986 kolme tutkijaa Stanfordin psykologian osastolta jatkokehittivät algoritmia, jonka kehittivät Widrow ja Hoff vuonna 1962. Tämä antoi mahdollisuuden käyttää useampia kerroksia neuroverkossa antaen koneelle mahdollisuuden oppia pitkällä aikavälillä. Vuonna 1997 Yleinen kiinnostus tekoälyä kohtaan kasvoi, kun IBM:n luoma tietokone nimeltään Deep Blue voitti shakin maailmanmestarin Garry Kasparovin ensimmäistä kertaa. Deep Blue tutki noin 6-20 siirtoa eteenpäin olevia mahdollisia siirtoja, jotka se oli oppinut arvioimalla tuhansia pelejä shakkia koittaen etsiä mahdollisuuksia voittoon. (BBC 2019.)

Vuonna 2006 yksi perustekniikoista, joita käytetään koneoppimisessa, on nimeltään backpropagation-algoritmi, jolla opetetaan neuroverkkoja. Ensimmäisen kerran kyseistä algoritmia esiteltiin 1960-luvulla, mutta siitä ei juurikaan kiinnostuttu, kunnes Geoff Hinton ja muut demonstroivat nykyaikaisia prosessoreita käyttäen sen tehokkuutta. Syväoppimisen verkot ovat nykyään osa koneoppimista. Vuonna 2017 Hinton, joka työskentelee nykyään Googlella, ilmaisi huolensa siitä, että backpropagation-algoritmi on tullut tiensä päähän koneoppimisessa ja sen tilalle pitäisi kehittää muita tapoja opettaa neuroverkkoja (BBC 2019).

4 Gartnerin hypekäyrä

Tutkimusyhtiö Gartnerin kehittämä hypekäyrä on trendiraportti, jolla tuodaan esille eri teknologioita, jotka voisivat mahdollisesti tulla kaupalliseen käyttöön. Gartnerin hypekäyrä näyttää, missä eri vaiheissa teknologiat ovat sekä kehityksessään että niiden käyttöönotossa. (Gartner 2020.)

Pernaa (2015) kertoo: ”Hypekäyrässä on viisi vaihetta, jotka ovat: käynnistys-, innovaatiovaihe, odotusten huippu, realismi tai pettymys iskee, ymmärrys kasvaa ja tuotanto alkaa”. Hypekäyrässä annetaan myös arvioita, miten pian teknologia pääsee viimeiseen vaiheeseen, jolloin teknologia otetaan enemmän käyttöön ja se alkaa tuottamaan. Nämä ennustukset menevät sykleittäin, jotka

ovat: alle 2 vuotta, 2-5 vuotta, 5-10 vuotta ja yli kymmenen vuotta. Gartnerin hypekäyrä on pitänyt hyvin paikkansa ja moni yritys seuraakin sitä tarkkaan etsien mahdollisia kohteita sijoituksille. (Pernaa 2015.)

Kotecki (2018) huomauttaa artikkelissaan Medium-julkaisussa, että vuoden 2018 hypekäyrässä koneoppiminen ei enää ollut mukana ensimmäistä kertaa sitten vuoden 2014 jälkeen mutta syväoppiminen on korvannut koneoppimisen. Syväoppiminen on koneoppimisen muoto, joten kokonaan koneoppiminen ei ole kadonnut hypekäyrältä. Koneoppiminen tai syväoppiminen on aina ollut vuoden 2015 jälkeen hypekäyrän huipulla merkittynä noin 2-5 vuoden päästä tuottavuuden alkamisesta. (Kotecki 2018.)

Vuoden 2019 Gartnerin hypekäyrällä on koneoppiminen ja syväoppiminen kadonnut. Kuitenkin näiden tilalle on käynnistysvaiheeseen tullut adaptive ML, joka vapaasti suomennettuna on mukautuva koneoppiminen. Kyseinen teknologia on merkitty 5-10 vuoden päähän, kunnes se saavuttaa tuottavuuden.
On siis kuitenkin selvää, että koneoppiminen on arvokas teknologia yrityksille. Vaikka koneoppiminen on hävinnyt syväoppimisen kanssa Gartnerin hypekäyrältä on kuitenkin jossain muodossa koneoppiminen käyrällä. Tämä taas kertoo siitä, miten isot odotukset koneoppimisesta ihmisillä on. Kuitenkin nykyisiä koneoppimisen malleja ja menetelmiä käytetään aktiivisesti jo, mikä viestii miten hyödyllisiä menetelmät ovat. (Kotecki 2018.)

5 Koneoppimisen soveltaminen yrityskäytössä

Koneoppiminen on oiva apuväline, mutta miten sitä voisi hyödyntää? Tässä luvussa on tarkoituksena tutkia keinoja, joita muut yritykset ovat ottaneet käyttöön ja miettiä miten Lemonsoft Oy hyötyisi näistä keinoista. On kuitenkin huomattava, että mikäli ongelmat ovat hyvin yksinkertaisia, on koneoppimisen käyttäminen turhaa. Koneoppiminen on tarkoitettu monimutkaisten ongelmien

ratkaisemiseen, joissa on useampia muuttujia, joiden kirjoittaminen käsin olisi monimutkaista ja aikaa vievää. (Expert System 2020.)

5.1 Yleisesti käytössä yrityksissä

Columbus (2019) kirjoittaa Forbesin kolumnissaan, että ennusteet ovat yksi yleisessä käytössä oleva ominaisuus koneoppimisessa. Mikäli yrityksellä on käytössä dataa, voidaan tehdä erilaisia ennusteita. Ennusteita, joiden avulla voidaan selvittää esimerkiksi, paljonko tuotteita on mennyt tiettyinä kuukausina kaupaksi, onko yksityishenkilö tai yritys vertailtavan datan perusteella halukas ostamaan uusia ominaisuuksia ohjelmiinsa. (Columbus 2019.)

Koneoppimisen avulla voidaan tutkia, onko jonkun tuotteen ostajalla tullut reklamaatiota tietystä tuotteesta. Kone voidaan opettaa ilmoittamaan, mikäli tietystä erästä tuotteita on tullut paljon reklamaatioita, jolloin ongelma saadaan nopeammin selville ja korjattua tilanne joko vetämällä tuote pois markkinoilta tai tekemällä niihin tarvittavat korjaukset. (Columbus 2019.)

Tukipyyntöjä voidaan myös tutkia kattavasti, ja sieltä voidaan poimia dataa. Teleoperaattori voi tutkia saatua tietoa, jossa tietyltä alueelta tulee paljon virheilmoituksia. Koneen voi opettaa huomaamaan yhtäläisyyksiä tukipyynnöissä kuten mikä yhteys on käytössä, mihin kellonaikaan, kartoittaa aluetta, josta tukipyyntöjä tulee. (Grossfeld 2019.)

5.2 Lemonsoftin toiminnanohjausjärjestelmään sopivia ominaisuuksia

Edellisessä luvussa mainittujen esimerkkien perusteella voi jo miettiä muutamia ideoita, mitä koneoppimisella voidaan saada aikaan. Myyntiä tutkimalla voidaan havaita, minä kuukausina yleisesti on ollut eniten myyntiä ja voidaan keskittää enemmän resursseja juuri noihin kausiin. (Columbus 2019.)

Tarjouksia tutkimalla voidaan ennakoida, miten tietyn toimialan edustaja, tietyllä liikevaihdolla on suhtautunut vastaaviin tarjouksiin. Tämän tiedon myötä pystytään miettimään, onko tarjous ollut tarpeeksi hyvä vai pitäisikö tarjousta muuttaa jollain tapaa myynnin varmistamiseksi. (Columbus 2019.)

Koneoppimisen avulla voidaan myös ennakoida, millä aikavälillä tiettyjen lisenssien käyttäjät ovat mahdollisesti alkaneet tarvitsemaan uusia lisenssejä entisten lisäksi. Ennakkoarvioinnin perusteella myyjät voivat tiedustella yrityksen tarvetta tietyille lisensseille. (Columbus 2019.)

Koneoppimista voi soveltaa myös laskujen maksuun, jolloin ohjelma ehdottaa maksun saajan nimen perusteella tiliä, mikäli kyseiselle maksun saajalle on tehty aikaisemmin maksuja. Käyttäjän on kuitenkin pitänyt syöttää maksunsaajan tiedot aikaisemmin ohjelmaan, jotta ohjelma osaa sitä ennakoida ja ehdottaa käyttäjälle. (Columbus 2019.)

Tukipyyntöjä tutkimalla koneoppimisella voisi huomata, jos tulee epätavallisia piikkejä tukipyynnöissä. Näitä tukipyyntöjä tutkimalla selviäisi mitä lisenssejä asiakkailla on ja onko jostakin tietystä asiasta tullut paljon tukipyyntöjä. Näin asiaan päästään nopeammin kiinni ja korjausta voidaan aloittaa hyvissä ajoin. Koneoppimista voisi myös soveltaa asiakaspalautteen seulomiseksi positiiviseen tai negatiiviseen palautteeseen automaattisesti. (Grossfeld 2019.)

6 Työkalut ja kehitysympäristö

Tässä luvussa tutustutaan ML.NET-koneoppimisen sovelluskehykseen ja Visual Studio -ohjelmointiympäristöön, joita käytän testatessa demoja, jotka Microsoft on laittanut ML.NET sivustolle testattavaksi kehittäjille.
Lemonsoft Oy käyttää monipuolisesti Microsoftin tarjoamia palveluita ja .NET-kehitysympäristöä, joten ML.NET oli luonteva valinta tutkinnan kohteeksi.

Tämän lisäksi ohjelmointikielet, joita yrityksessä käytetään ovat tuettu ML.NET sovelluskehyksessä, joten ohjelmoijille ei tule tarvetta opetella uutta kieltä. Ohjelmointikielenä käytän C#-kieltä, joka on itselleni tuttu opintojen kautta sekä Visual Studiosta itselläni on kokemusta jo useamman vuoden ajalta niin koulun kuin vapaa-ajan kautta. Käytössäni on Microsoftin Visual Studio Community edition 2017, joka on ilmainen käyttäjille. Tämän hetken uusin versio Visual Studiosta on versio 2019.

6.1 Visual Studio

Visual Studio on Microsoftin kehittämä ohjelmointiympäristö eli IDE (engl. integrated development environment). Se on helppo käyttää ja sillä voidaan luoda ohjelmia Androidille, iOS:lle, Windowsille, verkkoon ja pilveen. Kuten muissakin vastaavissa ohjelmointiympäristöissä Visual Studiossa on standardien mukaiset työkalut editoimiseen ja ongelmien ratkaisuun(debug). Tämän lisäksi Visual Studio tarjoaa muitakin ominaisuuksia, kuten graafisen suunnittelun työkalut, joita ei löydä kaikista ohjelmointiympäristöistä, jotka helpottavat ohjelmankehitystä. Muutamana esimerkkinä Visual Studion käyttäjillä on mahdollisuus ottaa käyttöönsä sellaiset ominaisuudet kuten Squiggles, Quick Actionit, Refactoring ja Intellisense. (Lee, Hogenson, Robertson & Warren 2019.)

Squigglet ovat aaltoilevia alleviivauksia, jotka kertovat virheestä tai mahdollisista ongelmista koodia kirjoittaessa. Tämä säästää aikaa, kun ohjelmoija näkee mahdollisen virheen heti ja voi korjata sen ennen ohjelman ajamista. Vietäessä hiiren kursorin virheen päälle ehdottaa Quick Action, miten virheen tai ongelman voisi korjata. (Lee, Hogenson, Robertson & Warren 2019.)

Refactoring auttaa nimeämään uudestaan useamman muuttujan ohjelmasta, muuttamaan metodin parametrien järjestystä ja paljon muuta. Tämän avulla voi siis helposti muuttaa koko ohjelmassa olevien muuttujien nimeä kerralla, käyttämättä ylimääräistä aikaa kaikkien muuttamiseen manuaalisesti. Mikäli

koodissa löytyy virhe voi refactoring myös ehdottaa korjausta virheelle automaattisesti. (Lee, Hogenson, Robertson & Warren 2019.)

Intellisense kertoo käyttäjälle tietoa koodista suoraan editorissa ja joissakin tapauksissa se voi myös kirjoittaa valmiiksi pieniä koodin muutoksia. Intellisensen tuki vaihtelee kielien välillä. Intellisensen avulla voi huomata editorissa viemällä hiiren muuttujan päälle omat kommentit muuttujasta tai metodista, jolloin aikaa säästyy, kun ei tarvitse etsiä tietoa komponenteista erikseen. (Lee, Hogenson, Robertson & Warren 2019.)

Visual Studio tukee .NET-ohjelmointikieliä ja tämän lisäksi se tukee Python-, JavaScript- ja C++-ohjelmointikieliä. Visual Studion käyttö onnistuu Windowsilla tai macOS-käyttöjärjestelmällä. Visual Studiosta on tarjolla kolme versiota: Visual Studio Community, Visual Studio Professional ja Visual Studio Enterprise. Näistä Visual Studio Community on ilmainen ja suoritan demojen testaamisen sillä. (Microsoft 2019.)

6.2 ML.NET

ML.NET-sivustolla kerrotaan, että ML.NET on avoimen lähdekoodin koneoppimisen sovelluskehys, jonka on kehittänyt Microsoft käytettäväksi .NET ohjelmistokehittäjien alustalle. ML.NET on alustariippumaton, ja se toimii macOS-, Linux- ja Windows-käyttöjärjestelmillä. (Microsoft 2019.) Microsoftin ML.NET Github-sivustolla kerrotaan, että ML.NET on lisensoitu MIT lisenssillä ja sitä voidaan käyttää ilmaiseksi kaupallisesti (Microsoft 2019). ML.NET 1.0 versio julkaistiin 4.5.2019 ja tällä hetkellä uusin versio on 1.3.1, joka on julkaistu 6.8.2019. Ennen 1.0 versiota oli tarjolla mahdollisuus testata ML.NET käyttöä. Ensimmäinen ennakkoversio ML.NET:stä tuli julkiseksi 7.5.2018. (Microsoft 2019.)

Morgan toteaa artikkelissaan Hacker Noon julkaisussa, että ML.NET koneoppimista voidaan laajentaa lisäämällä koneoppimisen kirjastoja ohjelman

käytettäväksi. Tällaisia koneoppimisen kirjastoja ovat esimerkiksi TensorFlow, Accord.NET, CNTK ja monet muut. (Morgan 2019). ML.NET sovelluskehys tukee .NET-kieliä eli kaikkia niitä voidaan käyttää koneoppimisen mallin luomiseen. Nämä ohjelmointikielet ovat C#, Visual Basic .NET ja F#. Näiden lisäksi Python on tuettu mutta se vaatii oman moduulin asentamisen. NimbusML moduulilla voi muuttaa algoritmit, jotka on kirjoitettu C++ tai C# -ohjelmointikielillä suoraan Python kielelle sopivaksi. (Bal 2019.)

ML.NET käyttää putkioppimista, joka yhdistää datan lataamisen, muuntamisen ja mallin opettamisen yhdeksi putkilinjaksi. Muutokset, jotka määritellään käyttäjän luomaan putkeen tulevat käyttöön opetukseen tarkoitetussa datassa ja syötteissä, joita käytetään arviointien tekoon opetetussa mallissa. (Alexander, Kershaw, Kulikov, Mohona, Schonning & Xu 2019.) ML.NET voi käyttää seuraavaa dataa opetukseen tarkoitetussa putkessaan: tekstiä (CSV/TSV), Apachen luomaa Parquet tietojen tallennusmuotoa, binaaria, IEnumerable<T> ja tiedostojoukkoja. Datan lataamisen jälkeen putki muuttaa datan haluttuun muotoon. ML.NET sisältää sisäänrakennettuja malleja, joilla muutetaan tieto haluttuun muotoon. ML.NET avulla voidaan muuttaa: tekstiä, datan skeemaa sekä tehdä puuttuvan datan arvojen käsittelyä, kategorisen muuttujan muuttamista koodiksi, normalisointia, valitsemaan toimivimman opetusominaisuuden ja NGram ominaisuudet. (Microsoft 2019.)

Tämän jälkeen pitää valita algoritmi, joka sopii parhaiten haluamaasi käyttötarkoitukseen. ML.NET tukee seuraavia algoritmeja: Linear, Boosted Trees, K-Means, SVM ja Averaged Perception. Algoritmin valinnan jälkeen mallia aletaan opettamaan. Opetus tapahtuu käyttämällä ohjelmassa Train metodia, joka palauttaa PredictionModel objektin, joka käyttää käyttäjän valitsemia datatyyppejä tehdessään arviointeja. (Microsoft 2019.)

Mallin saadessa opettelun loppuun voi mallia arvioida. ML.NET tarjoaa useita erilaisia arviointitapoja, joilla saa mallia arvioitua useiden erilaisten mittareiden avulla. Käyttäjän ollessa tyytyväinen luotuun malliin se voidaan tallentaa

binaaritiedostona, jonka voi integroida kaikkiin .NET sovelluksiin (Microsoft 2019).
Mikäli yrityksellä on tarkoitus käyttää tekoälyä helppoihin ja yksinkertaisiin ratkaisuihin on ML.NET liian raskas ratkaisu. ML.NET on valintana järkevä, jos yritys tarvitsee omia koneoppimisen malleja käytettäväksi yrityksen ohjelmistossa. Torre kirjoittaa blogissaan, miten ML.NET antaa myös mahdollisuuden kehittäjien luoda omat koneoppimisen mallit, omiin yrityskohtaisiin ongelmiin käyttäen yrityksen omaa dataa, jota se on kerännyt vuosien mittaan ML.NET-koneoppimisen mallit toimivat niin verkossa kuin ilmankin verkkoa, joten ne toimivat myös pilvipalveluissa. (Torre 2019.)

Microsoftilla on valmiina suunnitelmia, joilla on tarkoitus parantaa ML.NET:iä. Suunnitelmia on niin lyhyelle kuin pitkällekin aikavälille, joten ML.NET ei ole jäämässä pois tuetuista järjestelmistä. Pitkällä aikavälillä on tarkoitus parantaa putkirakenteen optimointia, lisätä uutta automaatiota tiedonkäsittelyyn, graafisen käyttöliittymän (GUI) parannuksia, uusien ohjelmointikielien tuen lisäämistä ja monia muita uudistuksia. (Gleb, Pagnoni, Shariq, Siddiqui, Tienhoang & Yamada 2019.)

6.3 Microsoft Azure Machine Learning service

Microsoftin työntekijät kertovat omalla sivustollaan, mikä Azure Machine Learning service on. Se on täysin pilvipalveluna toimiva koneoppimisen alusta. Azure Machine Learningin avulla voi opettaa, ottaa käyttöön ja ylläpitää koneoppimisen malleja. Azure Machine Learning tukee avoimen lähdekoodin teknologioita, joten siihen on helppo yhdistää Python paketteja kuten TensorFlow, Pytorch ja scikit-learn. Azure Machine Learning tarjoaa erilaisia työkaluja kehittäjien käytettäväksi kuten Azure notebooks, Jupyter notebooks ja Azure Machine Learning for Visual Studio Code joiden avulla datan muuttaminen haluttuun muotoon, mallin opettaminen ja mallin käyttöönotto on helppoa. Azure Machine Learning tukee ainoastaan Python ohjelmointikieltä. (Buck, Ericson, Petersen, Lee, Martens, Schonning, Martens, Urnun, Wasson & Wilson 2019.)

Azure Machine Learningia voi käyttää vähäisellä koodilla tai ilman koodia, kun käyttää interaktiivista visuaalista käyttöliittymää, jolla voi helposti ja nopeasti rakentaa, testata ja ottaa käyttöön malleja käyttämällä esirakennettuja koneoppimisen oppimisalgoritmeja (Buck ym. 2019).

Azure Machine Learning -sivustolta löytyy tiedot Azuren käytön hinnoittelusta. Azure Machine Learningia voi testata ilmaiseksi, mutta sen käyttö on maksullista. Maksu määräytyy sen mukaan, miten raskasta mallia ollaan tekemässä, paljonko RAM-muistia tarvitaan, millä käyttöjärjestelmällä tehdään opettaminen, maksetaanko kerran kuukaudessa, vuoden välein vai kolmen vuoden välein. (taulukko 1 ja taulukko 2) (Microsoft 2019.)

Taulukko 1. Azure Machine Learning B-series kuukausimaksullinen hinnasto. (Microsoft 2019)

Instance	VCPU	RAM	MLS Surcharge	Pay as you go	One year (% Savings)	Three Year (% Savings)
B1S	1	1 Gb	~€24.625/month	~€32.012/month	~€28.909/month (~10%)	~€27.414/month (~14%)
B2S	2	4 GB	~€49.249/month	~€78.798/month	~€66.536/month (~16%)	~€60.373/month (~23%)
B1LS	1	0.5 GB	~€24.625/month	~€28.319/month	~€26.804/month (~5%)	~€26.010/month (~8%)
B1MS	1	2 GB	~€24.625/month	~€39.399/month	~€33.268/month (~16%)	~€30.178/month (~23%)
B2MS	2	8 GB	~€49.249/month	~€108.348/month	~€83.822/month (~23%)	~€71.503/month (~34%)
B4MS	4	16 GB	~€98.498/month	~€216.695/month	~€167.649/month (~23%)	~€143.006/month (~34%)
B8MS	8	32 GB	~€196.995/month	~€433.389/month	~€335.298/month (~23%)	~€285.988/month (~34%)
B12MS	12	48 GB	~€295.493/month	~€650.084/month	~€503.015/month (~23%)	~€428.994/month (~34%)
B16MS	16	64 GB	~€393.990/month	~€866.778/month	~€670.663/month (~23%)	~€572.000/month (~34%)
B20MS	20	80 GB	~€492.488/month	~€1,083.472/month	~€838.312/month (~23%)	~€714.981/month (~34%)

Taulukko 2. Azure Machine Learning B-series tuntihinnat. (Microsoft 2019.)

Instance	VCPU	RAM	MLS Surcharge	Pay as you go	One year (% Savings)	Three Year (% Savings)
B1S	1	1 Gb	€0.011/hour	€0.034/hour	€0.040/hour (~10%)	€0.038/hour (~14%)
B2S	2	4 GB	€0.041/hour	€0.068/hour	€0.092/hour (~16%)	€0.083/hour (~23%)
B1LS	1	0.5 GB	€0.006/hour	€0.034/hour	€0.037/hour (~5%)	€0.036/hour (~8%)
B1MS	1	2 GB	€0.021/hour	€0.034/hour	€0.046/hour (~16%)	€0.042/hour (~23%)
B2MS	2	8 GB	€0.081/hour	€0.068/hour	€0.115/hour (~23%)	€0.098/hour (~34%)
B4MS	4	16 GB	€0.162/hour	€0.135/hour	€0.230/hour (~23%)	€0.196/hour (~34%)
B8MS	8	32 GB	€0.324/hour	€0.270/hour	€0.460/hour (~23%)	€0.392/hour (~34%)
B12MS	12	48 GB	€0.486/hour	€0.405/hour	€0.690/hour (~23%)	€0.588/hour (~34%)
B16MS	16	64 GB	€0.648/hour	€0.540/hour	€0.919/hour (~23%)	€0.784/hour (~34%)
B20MS	20	80 GB	€0.810/hour	€0.675/hour	€1.149/hour (~23%)	€0.980/hour (~34%)

6.4 Amazon Web Services

AWS (Amazon Web Services) on Amazonin tarjoama pilvipalveluna toimiva koneoppimisen järjestelmä. AWS on johtava pilvipalveluiden tarjoaja yhdessä Microsoftin ja Googlen kanssa. Tämä käy ilmi Gartnerin tutkimuksessa, joka on tehty heinäkuussa 2019. (Bala, Gill, Smith & Wright 2019.) Amazon Web Services sivustolta näkee, että heidän palveluitaan käyttävät sellaiset yritykset tai sivustot kuten Netflix, Razer, Rovio, Siemens ja Nasa (Amazon Web Services 2019).

Amazon Web Servicesin tarjoamia koneoppimisen palveluita käyttää puolestaan sellaiset yritykset kuin Wärtsilä, Kia, T-Mobile, Lenovo, NFL, Formula 1 ja monet muut. Yhteensä koneoppimisen palveluita käyttää yli 10 000 asiakasta. Amazon väittää sivustollaan kaikesta pilvipalveluissa käytetyistä syväoppimisesta tapahtuvan 81 prosenttisesti Amazon Web Servicen kautta. (Amazon Web Services 2019.)

Amazon Web Servicesin käytössä oleva koneoppimisen sovelluskehys on nimeltään Amazon SageMaker. Amazon SageMaker ilmoittaa sivustollaan tukevansa JavaScriptiä, Javaa, .NET, Node.js, PHP, Ruby ja Python ohjelmointikieliä (Amazon Web Services 2019).

Amazon Web Services hinnoittelee koneoppimisen käytön tuntihinnalla, jossa datan tutkiminen, mallin rakentaminen, mallin opettaminen sekä mallin käyttöönotto maksaa käytetyn ajan mukaan. Tämän lisäksi, mikäli käyttäjä haluaa rakentaa reaaliaikaisesti toimivan mallin, käytetystä datasta tulee myös maksu. Amazon Web Servicellä on Tukholmassa konesali, joka mahdollistaa nopean datan liikkumisen. Kuten Azuren palveluissa myös Amazonin Web Servicesin hintaan vaikuttaa miten uutta tietotekniikkaa halutaan käyttää ja hinnoissa voikin olla suuria eroja. Halvimmillaan hinta voi olla kuusi senttiä ja kalleimmillaan melkein 7 dollaria tunnilta (Amazon Web Services 2019) (taulukko 3).

Taulukko 3. Mallin rakentamisen hinta Amazon Web Services palvelussa. (Amazon Web Services 2019)

Standard Instances - Current Generation	Price per Hour
ml.t3.medium	$0.060
ml.t3.large	$0.121
ml.t3.xlarge	$0.242
ml.t3.2xlarge	$0.484
ml.m5.xlarge	$0.286
ml.m5.2xlarge	$0.572
ml.m5.4xlarge	$1.143
ml.m5.12xlarge	$3.428
ml.m5.24xlarge	$6.855

6.5 Koneoppimisen alustojen vertailua

AWS, Microsoft Azure Machine Learning Service ja ML.NET ovat kaikki erittäin vahvoja koneoppimisen sovelluskehyksiä ja hyviä valintoja käytettäväksi alustaksi. Näistä kolmesta tutkitusta sovelluskehyksestä AWS on käytetyin isojen yrityksien keskuudessa. Yhtiön oman ilmoituksen mukaan heidän pilvipalveluidensa avulla käytetään 81 % koko syväoppimisesta, mitä tapahtuu pilvipalveluissa (Amazon Web Services 2019). ML.NET on uusin tulokas joukosta ja sitä on julkisesti kehitetty viimeisen vuoden ajan.

AWS ja Azure tarjoavat pelkästään pilvipalveluiden varassa käytettävää koneoppimisen palveluita. ML.NETiä voidaan käyttää verkossa mutta myös ilman verkkoa. Koska ML.NET ei käytä pilvipalveluita, on mallin opettaminen, rakentaminen ja käyttöönotto tehtävä omilla koneilla, jolloin koneiden tehot vaikuttavat miten nopeasti palvelu saadaan käyttöön.

Hinnoittelussa on selkeitä eroja ja hyvin paljon vaihtelua. AWS tarjoaa ainoastaan tuntitaksan mukaan menevää hinnoittelua, kun taas Azuren käyttäjä voi valita useammasta vaihtoehdosta haluamansa. Azure tarjoaa niin tuntihinnan mukaista hinnoittelua kuin myös kuukausimaksullista palvelua. ML.NET on avoimen lähdekoodin tuote, joten sen käyttö on ilmaista.

Kolmesta tutkitusta sovelluskehyksestä ainoastaan Azure ei tue .NET ohjelmointikieliä. ML.NET ja Amazon SageMaker tukevat .NET ohjelmointikieliä ja näiden lisäksi muitakin kieliä, vaikkakin ML.NET ohjelmointikielien tuki on paljon suppeampi kuin Amazon SageMakerin.

Lemonsoft Oy:n tarpeisiin on haluttu koneoppimisen sovelluskehys, joka toimii .NET-kielillä, ja tämän lisäksi sen tulisi toimia myös ilman verkkoa. Näistä kolmesta tutkitusta sovelluskehyksestä ainoastaan ML.NET täyttää kaikki vaatimukset. Vaikka Amazonin tarjoama SageMaker on kattava kokonaisuus, ongelmaksi muodostuu pelkästään pilvipalveluna toimiva koneoppimisen malli. ML.NETin puolesta puhuu myös sen perustuminen avoimeen lähdekoodiin, jonka ansiosta ylimääräisiä kuluja ei tule yritykselle. Täytyy kuitenkin muistaa, että ML.NETillä kaikki koneoppimisen vaiheet täytyy tehdä yrityksen omilla laitteilla, joten laitteiden pitää olla riittävän tehokkaita ja eri vaiheille pitää antaa oma aikansa valmistua. Tämäkin on tosin riippuvainen siitä, miten isosta mallista on kyse.

7 ML.NET demot ja niiden rakentaminen

Tässä luvussa tutkin miten helppoa ML.NET demojen rakentaminen on. Microsoft tarjoaa omalla ML.NET sivustollaan testattavaksi demoja ohjeineen, jolla pääsee testaamaan miten ML.NET toimii ja mitä se tarvitsee toimiakseen.

Harjoituksessa rakennan mallin, opetan sen, arvioin sen toimintaa ja testaan mallin toimivuutta. Demossa käytetään Microsoftin tarjoamaa materiaalia, tässä tapauksessa Wikipedian arvosteluja, jotka on jaoteltu valmiiksi negatiiviseen ja positiiviseen palautteeseen. Arvo 1 tarkoittaa, että arvio on negatiivinen tai aggressiivinen ja arvo 0 tarkoittaa arvion olevan positiivinen tai kohtelias (kuva 1). Tiedostossa on 250 riviä tekstiä, joka riitti tällaisen pienen testin tekemiseen.

wikipedia-detox-250-line-data.tsv

Sentiment	SentimentText
1	==RUDE== Dude, you are rude upload that carl picture back, or else.
1	== OK! == IM GOING TO VANDALIZE WILD ONES WIKI THEN!!!
0	I hope this helps.

Kuva 1. Kuvakaappaus ensimmäisistä riveistä opetettavasta datasta (Kuva: Mika Salmi).

Tämän lisäksi testasin toista mallia, jossa halutaan ennustaa hintaa taksimatkalle. Opetettavana datana toimii yli 100 000 rivin csv-tiedosto, jossa on kuusi muuttujaa, jotka vaikuttavat taksikyydin hintaan. Nämä muuttujat ovat myyjän nimi, hinnoittelun koodi, matkustajien määrä, matkan kesto sekunteina, matkan pituus ja maksutapa.

7.1 Model Builder

Microsoftin tarjoamassa demossa käytetään ML.NETin Model Builder -nimistä visuaalista käyttöliittymälaajennusta, joka helpottaa työskentelyä ML.NET parissa. Aikaisempaa kokemusta koneoppimisesta ei tarvita käyttöliittymän käyttämiseksi. Käyttääkseen Model Builder laajennusta on käyttäjällä oltava Visual Studio 2017 versio 15.9.12 tai uudempi versio. (Kapsi & Murtaugh 2019.)

Model Builderilla on myös mahdollista tutkia automaattisesti erilaisia koneoppimisen algoritmeja ja asetuksia, jotka parhaiten sopivat rakennettavaan sovellukseen. Tämä onnistuu käyttämällä automatisoitua koneoppimista, AutoML, jolle on tuki Model Builderissa. Tämänhetkisessä versiossa käyttäjällä on mahdollisuus yhdistää suoraan csv-, tsv-tiedostoja ja SQL server tietokantoja. (Kapsi & Murtaugh 2019.)

AutoML tutkii useita eri malleja ja valitsee niistä parhaimman mahdollisen kyseiseen tehtävään, sekä näyttää käyttäjälle tutkitut mallit ja miten ne suoriutuivat. Mitä pidempi opetus asetetaan niin sitä useamman mallin ja asetuksen

AutoML ehtii tutkimaan. Microsoftin .NET tiimin työntekijät ovat testanneet eri aikoja ja tiedostojen kokoa ja saaneet seuraavanlaisia tuloksia. (taulukko 4) (Kapsi & Murtaugh 2019.)

Taulukko 4. Taulukossa näytetään miten opetettavan datan koko vaikuttaa aikaan, joka pitää käyttää mallin opettamiseen (Kapsi & Murtaugh).

Opetettavan datan koko	**Datan tyyppi**	**Keskiverto opetusaika**
0 – 10 Mb	Numeerinen ja teksti	10 sekuntia
10 – 100 Mb	Numeerinen ja teksti	10 minuuttia
100 – 500 Mb	Numeerinen ja teksti	30 minuuttia
500 Mb – 1 Gb	Numeerinen ja teksti	60 minuuttia
1 Gb+	Numeerinen ja teksti	3 tuntia +

Käytettävään opetusaikaan vaikuttaa:

- ennustamisen ominaisuuksien tai rivien määrä
- rivien tyyppi esim. tekstin ja numeerisen datan erot
- koneoppimisen työtehtävä.

Microsoftin .NET-tiimi on testannut AutoML:n avulla myös 1 teratavun kokoista dataa mutta korkealaatuisen mallin rakentaminen voi noin suurella tiedon määrällä viedä yli neljä päivää. (Kapsi & Murtaugh 2019).

On kuitenkin muistettava, ettei Model Builder ole vielä täysin vakaa versio vaan se on testivaiheessa vielä. Tämä voi aiheuttaa ongelmia joissakin tapauksissa eikä luettavien tiedostojen tuki ole vielä kovin laaja. Microsoft kehittää kuitenkin

Model builderia aktiivisesti ja siihen on tulossa lisää ominaisuuksia ja tuki lukea erilaisia tiedostoja. (Kapsi & Murtaugh 2019.)

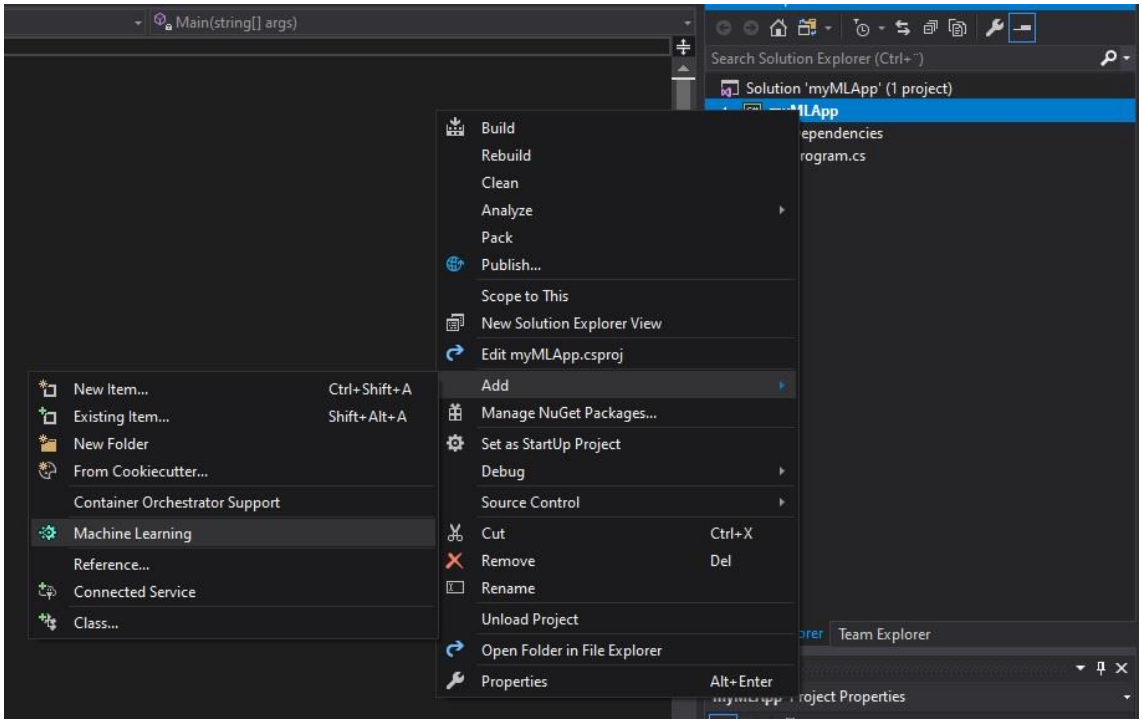
7.2 Demojen rakentaminen

Tein ohjelmat käyttämällä C#-ohjelmointikieltä ja Microsoft Visual Studio 2017 Community 15.9.14 versiolla. Olin ladannut jo ennakkoon Model Builder laajennuksen, joten varsinaisen ohjelman rakentaminen oli helppoa, kun esivalmistelut olivat suoritettu. Valitsin alun perin vain yhden demon mutta muutin mieltäni koska halusin saada täysin erilaisen demon myös, jotta voin tehdä vertailuita niiden välillä.

Ensimmäisessä demossa dataa ei ole läheskään niin paljon kuin toisessa demossa, joten pystyin vertailemaan miten paljon opetettavan datan määrä vaikuttaa tuloksiin. Ensimmäinen demo on myös paljon yksinkertaisempi kuin toinen, sillä toisessa demossa on paljon enemmän muuttujia, jotka vaikuttavat lopputulokseen ja ennusteeseen.

7.3 Demo 1: Jaottelu kahteen luokkaan

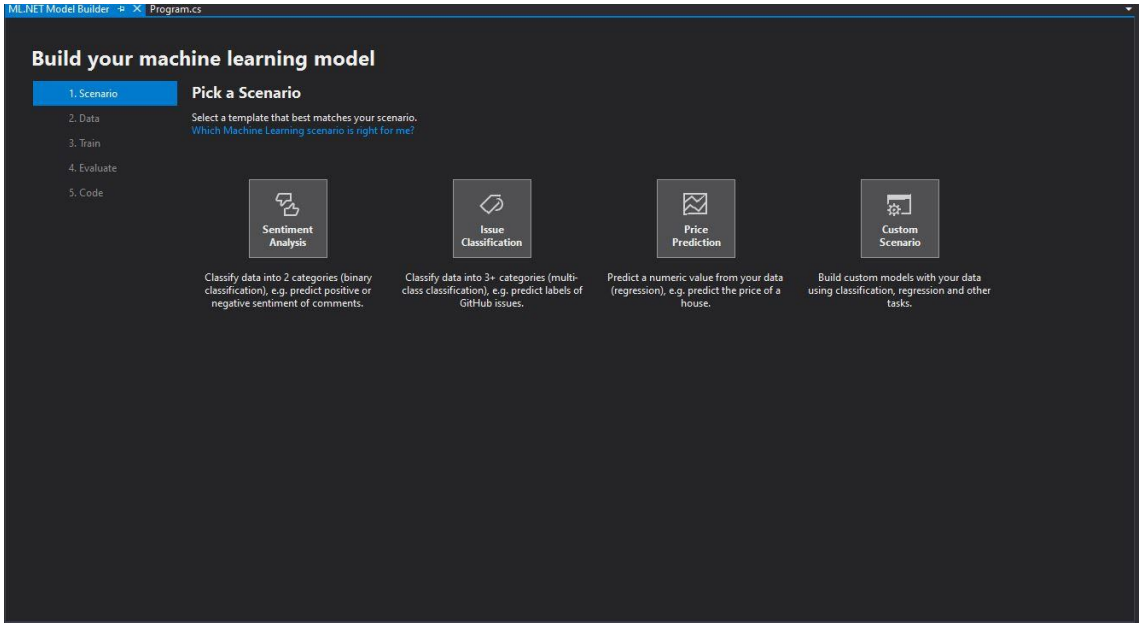
Ohjelmaa varten loin uuden projektin Visual Studiossa ja valitsin C# Console App(.NET Core) -mallin. Projektin nimeksi vaihdoin myMLApp ja loin kansion projektilleni. Koneoppimisen lisääminen projektiin oli mutkatonta Model Builderin avulla. Koneoppimisen lisääminen projektiin onnistuu klikkaamalla oikealla Solution Explorerissa projektia ja sen jälkeen valitsemalla Add -> Machine Learning (kuva 2).



Kuva 2. Kuvakaappaus koneoppimisen lisäämisestä projektiin (Kuva: Mika Salmi).

Seuraavaksi pääsin valitsemaan koneoppimisen mallin. ML.NET Model Builder tarjosi minulle neljää eri mallia, joista valitsin Sentiment Analysisin, joka on tarkoitettu datan erotteluun kahteen eri kategoriaan kuten positiiviseen ja negatiiviseen.

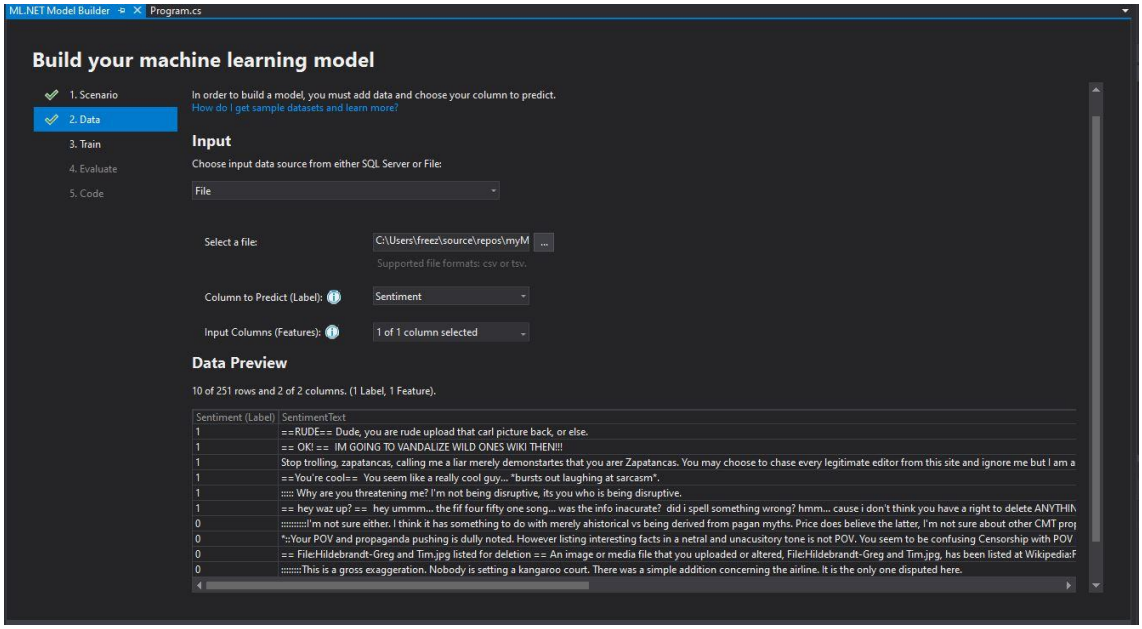
Issue Classificationilla voidaan data erotella yli kolmeen luokkaan kuten vaikka Githubissa tapahtuviin virheisiin. Price Prediction -mallilla voidaan ennustaa tietyn asian hintaa, kuten vaikka talon arvoa. Mikäli näistä ei mikään sovi valmiina voidaan valita Custom Scenario vaihtoehto, jolloin käyttäjä saa itse rakentaa sopivan mallin omiin tarkoituksiinsa (kuva 3).



Kuva 3. Kuvakaappaus koneoppimisen mallin valinnasta (Kuva: Mika Salmi).

Tämän vaiheen jälkeen valitaan data, jota käytetään mallin opetuksessa. Kuten aikaisemmin kirjoitin, käytin Wikipedian arvosteluita, jotka olivat jo jaoteltu negatiivisiin ja positiivisiin palautteisiin.

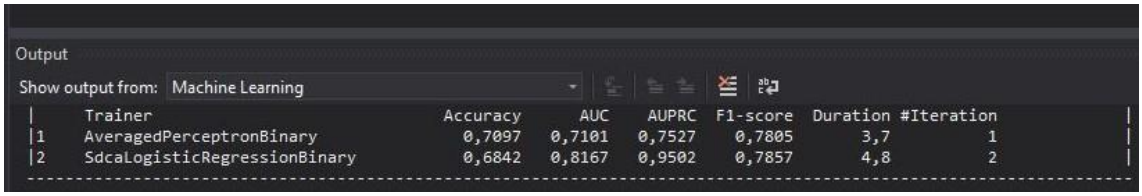
Add Data -valikossa voidaan lisätä dataa ohjelman opeteltavaksi joko paikallisesta tiedostosta tai SQL serverin tietokannasta (kuva 4). Tässä tapauksessa opeteltava tiedosto oli tallennettu minun koneelleni, joten Input-vetovalikosta valitsin File. Tämän jälkeen etsin kyseisen tiedoston koneeltani Select File kohtaan. Column to predict (label) kohtaan valitsin Sentiment, sillä tarkoituksena on valita, onko arvo 1 (negatiivinen palaute) vai 0 (positiivinen palaute). Input columns (features) -vetovalikkoon en koskenut, sillä opeteltava asia on kyseisten käyttäjien kirjoittamat palautteet. Input columns ovat muuttujia, jotka vaikuttavat ennusteeseen mutta koska tällä hetkellä niitä on vain yksi, ei malli ole kovinkaan monimutkainen.



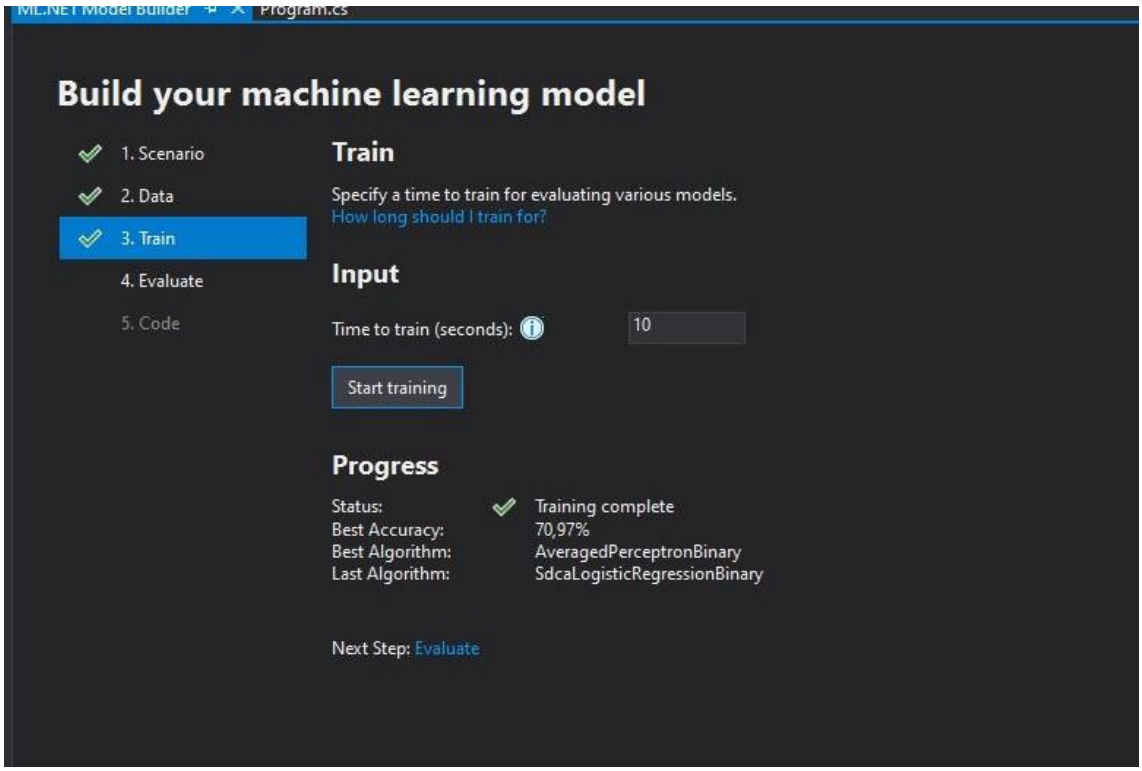
Kuva 4. Kuvakaappaus opeteltavan datan lisäämisestä (Kuva: Mika Salmi).

Seuraavana vuorossa oli mallin opettaminen. Model Builderin avulla minun ei itse tarvinnut valita erilaisia malleja tai algoritmeja, joilla opettaa malliani vaan se hoidettiin minun puolestani automaattisesti. Annoin ohjelman opetella annetusta datasta 10 sekunnin ajan, mutta mitä enemmän dataa on opeteltavana, sitä enemmän kannattaa aikaa antaa käytettäväksi mallille. Tämä takaa sen, että arviot ovat tarkempia. Mallin opetuksen jälkeen Model Builder antaa arvion mallin opetuksesta, näyttää mitä algoritmeja käytettiin, montako algoritmia kokeiltiin, ja tämän lisäksi se antaa vielä arvion, millaisella todennäköisyydellä arvio on oikein. Minun opetukseni kohdalla se oli 70,97 % oikein (kuva 5 ja kuva 6). Tämä siis tarkoittaa, että pyöristettynä palautteen arviointi on oikeassa 71 % todennäköisyydellä. Mikäli dataa ja aikaa olisi ollut enemmän käytössä olisi mallin arviot suuremmalla todennäköisyydellä osunut oikeaan. Ohjelma olisi myös ehtinyt käydä useamman algoritmin läpi mutta tällä kertaa se ehti vain tutkia kahta.

Testasin myös mallin opettamista pidemmällä aikavälillä, ja tulokset olivat sellaiset kuten arvelinkin niiden olevan. Mitä enemmän aikaa oli käytössä, sitä paremmat arviot olivat. Jo 90 sekunnin opettamisella sain nostettua arvion tarkkuuden 73 %:iin.

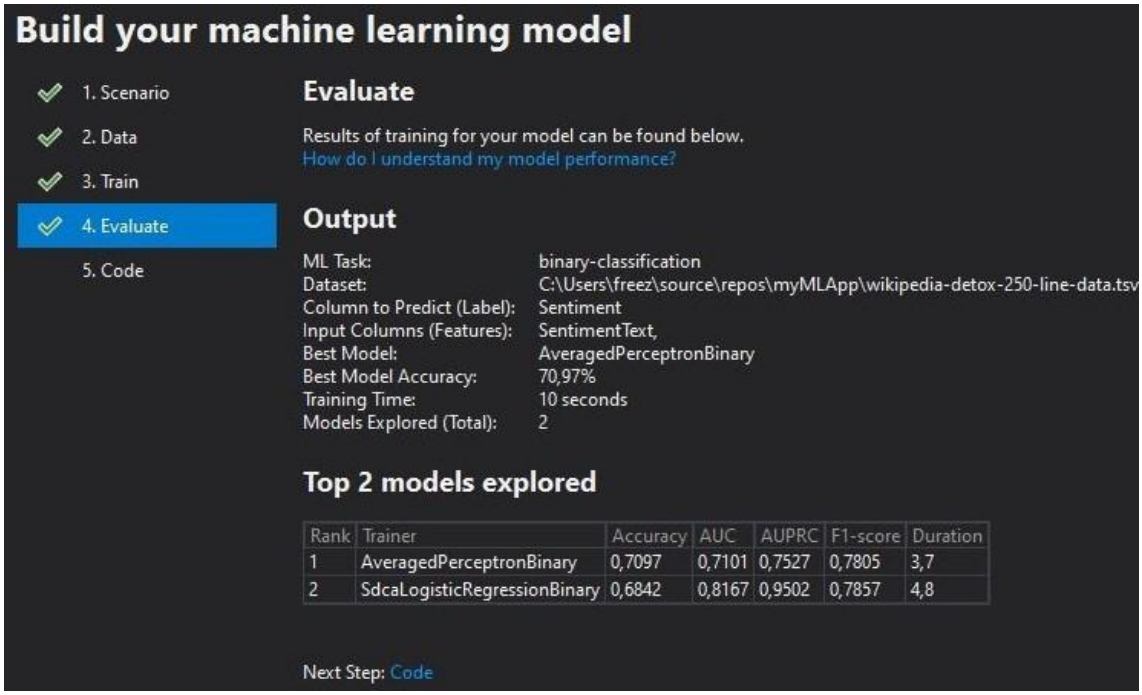


Kuva 5. Kuvakaappaus ohjelman opettamisesta (Kuva: Mika Salmi).



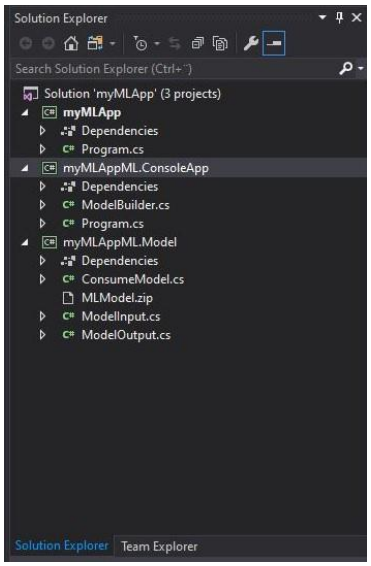
Kuva 6. Kuvakaappaus ohjelman opetuksesta (Kuva: Mika Salmi).

Mallin arvioinnissa kerrataan ja kootaan yhteen jo aikaisemmin nähdyt tulokset. Tässä näkymässä voi vielä tarkistaa, onko kaikki asetukset oikein ja mikäli ei ole tyytyväinen johonkin tuloksiin voi käyttäjä palata aina takaisiin edellisiin vaiheisiin ja muuttaa asetuksia (kuva 7).



Kuva 7. Kuvakaappaus ohjelman arvioinnista (Kuva: Mika Salmi).

Seuraavassa näkymässä käyttäjä painaa Add Projects -nappia, jolloin Model Builder lisää projektiin koneoppimisen mallin ja projektit, joilla sitä voidaan opettaa ja käyttää valitussa sovelluksessa. Solution Explorer valikkoon tulee tiedostot ja koodi, jonka ohjelma on tehnyt käyttäjälle valmiiksi (kuva 8).



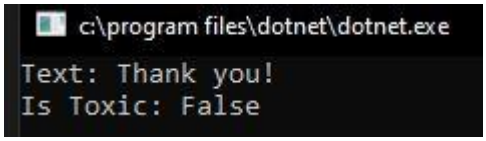
Kuva 8. Kuvakaappaus Solution Explorerista Model Builderin automaattisen koodin lisäämisen jälkeen (Kuva: Mika Salmi).

Automaattisen koodin lisäämisen jälkeen voidaan ruveta kirjoittamaan koodia itse ja testaamaan miten ohjelma toimii. Kaikki muutokset tehdään Program.cs-tiedostoon myMLApp-projektiin. Ensimmäisenä otetaan pois automaattisesti luotu Hello World! ja korvataan se omalla koodilla (kuva 9).

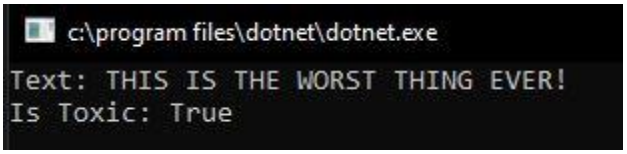
```
using System;
using MyMLAppML.Model;

namespace myMLApp
{
    class Program
    {
        static void Main(string[] args)
        {
            // lisää input data
            var input = new ModelInput();
            // input.SentimentText arvoa muuttamalla voi testata koneoppimisen mallia
            input.SentimentText = "THIS IS THE WORST THING EVER!";

            // Ladataan malli ja ennustetaan syöte
            ModelOutput result = ConsumeModel.Predict(input);
            Console.WriteLine($"Text: {input.SentimentText}\nIs Toxic:
{result.Prediction}");

            Console.ReadLine();
        }
    }

}
```

Kuva 9. Valmis koodi ohjelmasta (Kuva: Mika Salmi).

Kun koodi on valmis, voidaan mallia testata. Testaaminen on tässä tapauksessa helppoa muuttamalla input.SentimentText arvoa haluttuun arvoon. Koska opetettava materiaali oli englanniksi, on myös arvo oltava englanniksi. Muuten ohjelma ei osaa arvioida syötettä oikein. Kun on laitettu haluttu arvo, ohjelma antaa arvion onko arvo negatiivinen vai positiivinen (kuva 10 ja kuva 11).



Kuva 10. Kuvakaappaus ohjelman arviosta positiivisesta syötteestä (Kuva: Mika Salmi).



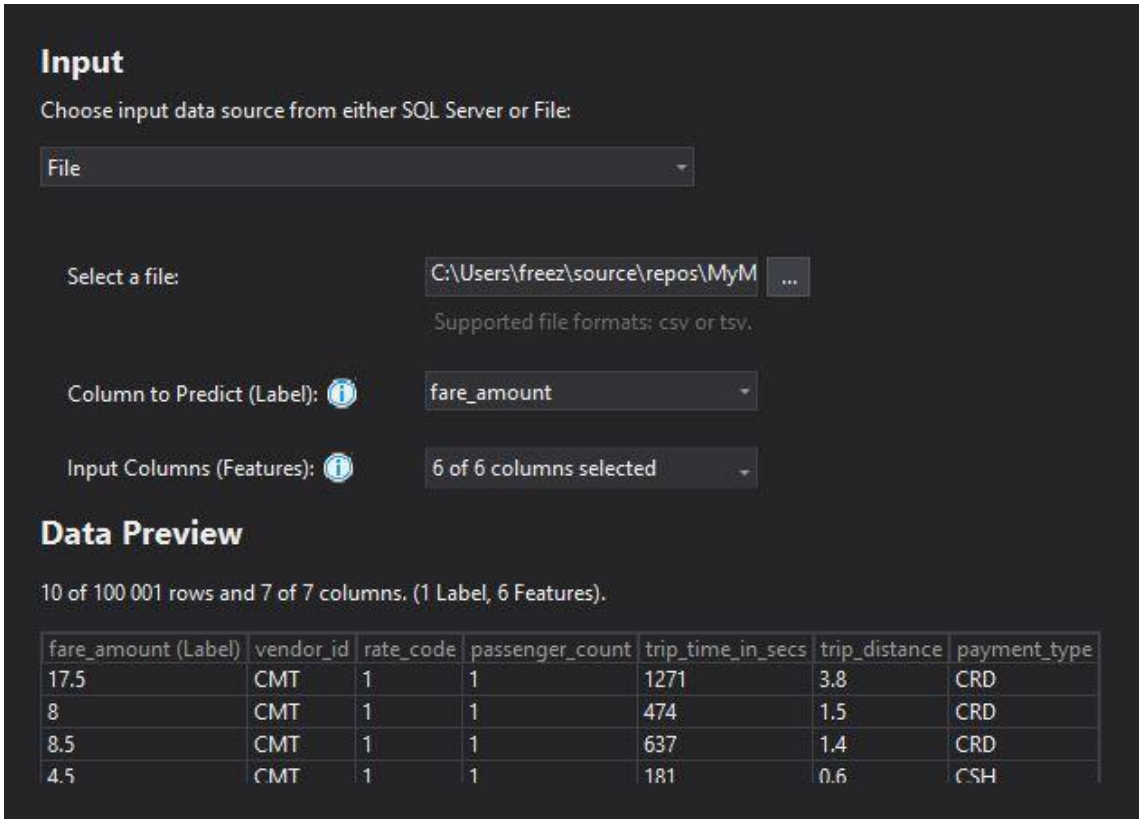
Kuva 11. Kuvakaappaus ohjelman arviosta negatiivisesta syötteestä (Kuva: Mika Salmi).

7.4 Demo 2: Hinnan ennustaminen

Tässä demossa oli tarkoitus saada ennuste taksimatkan hinnasta ja sitä varten luodaan koneoppimisen malli, joka ennustaa sen. Tällä kertaa käytössä on huomattavasti suurempi määrä dataa, joten tulokset voivat olla paremmat kuin edellisessä demossa.

Koneoppimisen lisääminen projektiin toimii samalla tavalla kuin edellisessäkin demossa. Klikataan hiiren oikealla painikkeella oman projektin nimeä, minun tapauksessani MyMLApp2 > Add > Machine learning.

ML.NET Model Builder -opastuksen avulla valitaan tällä kertaa malliksi Price prediction, jolla voidaan helposti ennustaa hintoja. Datan tyypiksi otetaan taas tiedosto paikallisesta lähteestä, etsitään kyseinen tiedosto, jonka olen ladannut omalle tietokoneelleni ML.NET sivustolta. Column to Predict (Label) kohtaan valitaan fare_amount koska se on arvo, jonka ennusteen haluamme. Input Column sisältää loput sarakkeet, ja ne ovat muuttujia tässä koneoppimisen mallissa kuten muissakin malleissa (kuva 12).

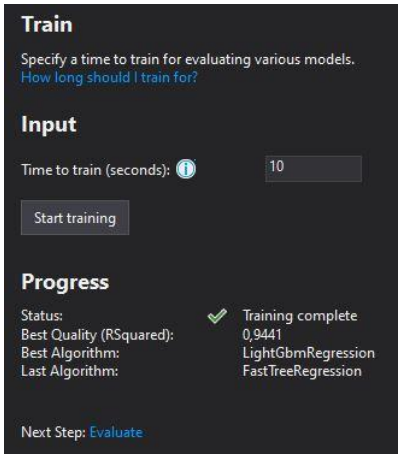


Kuva 12. Kuvakaappaus opeteltavan datan lisäämisestä ja ennustettavan arvon valinta (Kuva: Mika Salmi).

Opeteltavan datan lisäämisen jälkeen opetin ohjelmaa käyttämällä dataa 10 sekuntia. Koska dataa oli paljon, sain ennusteen arvoksi 94,4 % tuloksen (kuva 13).

Ohjelma siis pystyy ennustamaan taksimatkan hinnan 94,4 % tarkkuudella. Mikäli en ole tyytyväinen tulokseen voin opettaa mallia uudestaan lisäämällä tai vähentämällä datan opetukseen käytettävää aikaa.

Seuraavassa vaiheessa voidaan tarkistaa kaikki asetukset ja sen jälkeen voidaankin lisätä valmis koodi ohjelmaan klikkaamalla Add Projects -painiketta.



Kuva 13. kuvakaappaus valitun datan avulla kouluttamisen tuloksesta (Kuva: Mika Salmi).

Model Builderin koodin lisäämisen jälkeen voidaan ruveta tekemään itse koodia. Ohjelman koodi kirjoitetaan MyMLApp2 valikon alla olevaan Program.cs-tiedostoon. Kun koodi on valmis, voidaan ennustetta testata vaihtamalla arvoja (kuva 14).

```
using System;
using MyMLApp2ML.Model;

namespace MyMLApp2
{
    class Program
    {
        static void Main(string[] args)
        {

            // Lisätään syötettävä data joita muuttamalla voidaan muuttaa ennustetta
            var input = new ModelInput();
            //Matkan pituus
            input.Trip_distance = 100;
            //Matkan kesto
            input.Trip_time_in_secs = 1000;
            //Matkustajien määrä
            input.Passenger_count = 1;
            //Maksutapa CRD = Kortti, CSH = käteinen
            input.Payment_type = "CRD";
            //Hinnoittelun koodi
            input.Rate_code = 1;
            //Firman id
            input.Vendor_id = "CMT";

            // Ladataan malli ja annetaan ennuste matkan kustannuksesta
            ModelOutput result = ConsumeModel.Predict(input);
            Console.WriteLine(result.Score);

            Console.ReadLine();
        }
    }
}
```

Kuva 14. Valmis koodi taksimatkan hinnan arviosta (Kuva: Mika Salmi).

8 Koneoppimisen tulevaisuus

Koneoppiminen on ollut pitkään tutkittu aihe ja todennäköisesti tulee olemaan myös tulevaisuudessakin. Missä muodossa koneoppiminen on käytössä tulevaisuudessa, sitä on vaikea vielä ennustaa. Vuoden 2019 Gartnerin hypekäyrässä näkyy, miten tilalle on tullut uusi koneoppimisen muoto mukautuva koneoppiminen. Miten paljon tämä tulee vaikuttamaan muihin koneoppimisen malleihin, on vielä tässä vaiheessa vaikea ennustaa. Mukautuvan koneoppimisen malli on vielä Gartnerin hypekäyrän mukaan noin 5 -10 vuoden päästä tuottavuudesta, jolloin siitä saadaan kaikki hyöty käyttöön. (Kotecki 2018.)

On kuitenkin selvää, että koneoppimisesta ei tulla luopumaan, sillä koneoppiminen on kuitenkin käytössä niin monessa elämän osa-alueessa ja siitä on tullut korvaamaton työväline joillekin aloille kuten lääketieteelle, liiketaloudelle ja ICT-alalle. Mitä olisikaan maailma ilman koneoppimisen suosituksia, ennakkosyöttöjä, suuren datamäärän tai muiden isojen ongelmien tutkimista? Uskon, että koneoppimiselle on valoisa tulevaisuus sillä, siitä on tullut iso osa arkipäivää, että sen korvaajaa on vaikea löytää. (Columbus 2019.)

ML.NETin tulevaisuus näyttää sekin valoisalta, sillä siihen tehdään tasaiseen tahtiin päivityksiä, joista viimeisin iso päivitys ML.NET 1.4 julkaistiin 6.11.2019 Tämä päivitys toi mukanaan paljon uusia ominaisuuksia ja parannuksia ML.NETin käyttöön. Uskon myös, että avoimen lähdekoodin sovelluskehyksenä sille riittää kysyntää ja ihmiset tulevat käyttämään ML.NET sovelluskehystä osana omia projektejaan. (Torre 2019.)

9 Tuloksien pohdinta ja analysointi

Ohjelman tekeminen oli helppoa hyvien ohjeiden ansiosta ja ML.NET Model Builderin. Itselleni tuli yllätyksenä, miten nopeasti sain tehtyä koneoppimisen mallin käyttökuntoon, vaikka aikaisempaa kokemusta ei minulla ole koneoppimisesta muuta kuin teoriasta.

Vaikka ensimmäinen demo oli hyvin yksinkertainen, voisi sitä soveltaa sellaisenaan johonkin tarkoitukseen Lemonsoft Oy:n toiminnanohjausjärjestelmässä. Koneoppimisen mallia voisi soveltaa laittamalla sen suodattamaan asiakaspalautetta ja järjestelemään ne negatiiviseen ja positiiviseen palautteeseen. Näin näkisi nopeasti, mikäli negatiivista palautetta alkaa tulla normaalia enemmän, jolloin asioihin puuttuminen olisi nopeampaa.

Toisessa demossa nähtiin selkeästi, miten paljon datan määrä vaikuttaa ennusteen tuloksen paranemiseen. 94,4 %:n todennäköisyys saada ennuste oikein on huikea lukema. Tämä saatiin vain 10 sekunnin mallin opettamisen avulla, joten pidän tulosta erinomaisena suorituksena. Kyseistä demoa voisi soveltaa tai muuttaa toimimaan jo aikaisemmin pohtimallani tavalla. Dataa tutkimalla voisi ennustaa, millaisella tarjouksella asiakas on hyväksynyt uusien lisenssien oston, jos hänellä on ollut jo tietyt lisenssit käytössään. Tämän lisäksi muuttujina voi käyttää firman kokoa, liikevaihdon suuruutta tai muuta vastaavaa tietoa. (Columbus 2019.)

Ilman demojakin voisi kuvitella, miten laskutukseen liitettävä koneoppiminen voisi ennakoida käyttäjän syötteitä. Tämä voisi tapahtua esimerkiksi antamalla ohjelmiston käyttäjälle maksunsaajan tilinumero pelkän nimen perusteella. Tietenkin tässä pitää ottaa huomioon, ettei koneoppimisen malli ole täysin varma vaan loppujen lopuksi vastuu tarkistukseen jää aina käyttäjälle. (Kozyrkov 2018.)

Tuloksien vertailun ja opinnäytetyön tekemisen jälkeen olen tullut siihen tulokseen, että Lemonsoft Oy hyötyisi ottamalla käyttöönsä koneoppimisen joko valmistamaansa ohjelmistoon tai omiin sisäisesti käytettäviin järjestelmiin. Sovelluskehykseksi valitsisin kolmesta tutkitusta vaihtoehdosta ehdottomasti ML.NETin, sillä se oli ainut vaihtoehto, joka täytti kaikki vaatimukset, joita sovelluskehykseltä vaadittiin.

Koneoppimisen mallin opettaminen vaatii paljon laadukasta dataa. Mikäli dataa ei ole tarpeeksi, voi tulokset jäädä huonoiksi ja koneoppimisen kautta saatu ennusteen todennäköisyysprosentti jää alhaiseksi (Venkatesh 2017). Tämä oli selkeästi havaittavissa, kun opetettavan datan välissä oli suuri ero. Ensimmäisessä demossa minulla oli käytössä tiedosto, jossa oli 250 riviä opetettavaa dataa, kun taas toisessa demossa opetettavaa dataa oli 100 001 riviä. Kumpaakin demoa opetettiin ajallisesti saman verran eli 10 sekuntia. Demojen tulokset olivat selkeitä, sillä ensimmäisessä demossa todennäköisyysprosentti saada oikea tulos oli pyöristettynä 71 % kun taas toisessa demossa, jossa opetettavaa dataa oli enemmän, todennäköisyys saada oikea tulos oli paljon suurempi: 94,4 %. Onkin siis äärimmäisen tärkeää, että dataa on paljon ja se on laadukasta, mikäli koneoppimisesta halutaan ottaa kaikki mahdollinen hyöty (Venkatesh 2017).

On kuitenkin muistettava, että vaikka tulokset olivat hyviä, kaikki koneoppimisella saatu data on kuitenkin hyvä tarkistaa, sillä koneoppiminen ei ole mikään ihmeiden tekijä, vaikka se nopeuttaa ja helpottaa asioita. Tämän lisäksi en suosittele koneoppimisen käyttöä, jos asian voi ratkaista helposti yksinkertaisella koodilla mutta mikäli muuttujia on paljon, kannattaa ehdottomasti miettiä koneoppimisen ottamista osaksi ohjelmistoa. (Kozyrkov 2018.)

Lähteet

Alexander, J., Kershaw, N., Kulikov, P., Mohona, M., Schonning, N. & Xu, C. 2019. What is ML.NET and how does it work? Microsoft. https://docs.microsoft.com/en-us/dotnet/machine-learning/how-does-mldotnet-work 13.12.2019

Amazon Web Services. 2019. Amazon SageMaker developer resources. Amazon Web Services, Inc. https://aws.amazon.com/sagemaker/developer-resources/ 10.9.2019

Amazon Web Services. 2019. AWS Customer Success. Amazon Web Services, Inc. https://aws.amazon.com/solutions/case-studies/all/ 5.9.2019

Amazon Web Services. 2019. Machine Learning – customers. Amazon Web Services, Inc. https://aws.amazon.com/machine-learning/customers/ 5.9.2019

Bala, R., Gill, B., Smith, D. & Wright, D. 2019. Magic Quadrant for Cloud Infrastructure as a Service, Worldwide. Gartner, Inc. https://www.gartner.com/doc/reprints?id=1-1CMAPXNO&ct=190709&st=sb 5.9.2019

Bal, M. 2019. NimbusML. Microsoft. https://github.com/microsoft/NimbusML 10.9.2019

BBC. 2019. The History of Machine Learning. BBC. https://www.bbc.com/timelines/zypd97h 5.9.2019

Bean, R. 2018.The State of Machine Learning in Business Today. Forbes Media LLC. https://www.forbes.com/sites/ciocentral/2018/09/17/the-state-of-machine-learning-in-business-today/#1c9cc1fd3b1d 23.2.2020

Brownlee, J. 2019. Supervised and Unsupervised Machine Learning Algorithms. Machine Learning Mastery Pty. Ltd. https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ 5.11.2019

Buck, A., Ericson, G., Petersen, T., Lee, C., Martens, J., Schonning, N., Martens, J., Urnun, M., Wasson, M. & Wilson, M. 2019. What are the machine learning products at Microsoft. Microsoft. https://docs.microsoft.com/en-us/azure/architecture/data-guide/technology-choices/data-science-and-machine-learning 4.9.2019

Columbus, L. 2019. 10 Ways to Improve Cloud ERP With AI and Machine Learning. Forbes Media LLC. https://www.forbes.com/sites/louiscolumbus/2018/07/29/10-ways-to-improve-cloud-erp-with-ai-machine-learning/#4703e5a619b7 5.11.2019

Dave, A. 2018. Regression in Machine Learning. Medium Corporation. https://medium.com/datadriveninvestor/regression-in-machine-learning-296caae933ec 4.11.2019

Expert System S.p.A. 2020. What is Machine Learning? A definition. https://expertsystem.com/machine-learning-definition/ 23.2.2020

Gartner. 2020. Gartner Hype Cycle. Gartner, Inc. https://www.gartner.com/en/research/methodologies/gartner-hype-cycle 27.1.2020

Gleb, K., Pagnoni, A., Shariq, S., Siddiqui, Z., Tienhoang, V. & Yamada, Y. 2019. The ML.NET Roadmap. Microsoft. https://github.com/dotnet/machinelearning/blob/master/ROADMAP.md 13.12.2019

Grossfeld, B. 2019. How is machine learning being used in customer service. Zendesk. https://www.zendesk.com/blog/machine-learning-used-customer-service/ 23.2.2020

Gupta, A. 2019. Machine Learning for Anomaly Detection. GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning-for-anomaly-detection/ 4.11.2019

Gupta, M. 2019. ML | Linear Regression. GeeksforGeeks. https://www.geeksforgeeks.org/ml-linear-regression/ 4.11.2019

Kapsi, R. & Murtaugh, B. 2019. ML.NET Model Builder Guide. Microsoft. https://github.com/dotnet/machinelearning-samples/tree/master/modelbuilder 8.11.2019

Kordík, P. 2018. Machine Learning for Recommender systems — Part 1 (algorithms, evaluation and cold start). Medium Corporation. https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed 4.11.2019

Kotecki, J. 2018. Deep learning's 'Permanent Peak' On Gartner's Hype Cycle. Medium Corporation. https://medium.com/machine-learning-in-practice/deep-learnings-permanent-peak-on-gartner-s-hype-cycle-96157a1736e 1.12.2019

Kozyrkov, C. 2018. Whose fault is it when AI makes mistakes? Medium Corporation. https://towardsdatascience.com/dont-trust-ai-10a7df520925 23.2.2020

Lee, T., Hogenson, G., Robertson, C. & Warren, G. 2019. Welcome to the Visual Studio IDE. Microsoft. https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2017 5.11.2019

Merilehto, A. 2018 Tekoäly – matkaopas johtajalle. Helsinki: Alma Talent Oy.

Microsoft. 2019. Azure Machine Learning Pricing. Microsoft. https://azure.microsoft.com/en-in/pricing/details/machine-learning-service/ 5.9.2019

Microsoft. 2019. Machine Learning for .Net. Microsoft. https://github.com/dotnet/machinelearning/blob/master/README.md 13.9.2019

Microsoft. 2019. Web Languages. Microsoft. https://visualstudio.microsoft.com/vs/features/web/languages/ 13.12.2019

Microsoft. 2019. What is ML.NET? Microsoft. https://dotnet.microsoft.com/learn/ml-dotnet/what-is-mldotnet 26.8.2019

Morgan, H. 2019. ML.NET: Machine Learning framework by Microsoft for .NET developers. Hacker Noon. https://hackernoon.com/ml-net-machine-learning-framework-by-microsoft-for-net-developers-3c6f46bcb59f 22.8.2019

Neittaanmäki, P. & Siukonen T. 2019. Mitä tulisi tietää tekoälystä. Jyväskylä: Docendo Oy.

Pernaa, J. 2015. Hypekäyrä TVT-suunnittelussa. AJATUKSIA. https://peda.net/p/johannespernaa/ajatuksia/2015/hypekayra 5.11.2019

Priya, S. 2019. Clustering in Machine Learning. GeeksforGeeks. https://www.geeksforgeeks.org/clustering-in-machine-learning/ 4.11.2019

Sidana, M. 2017. Types of classification algorithms in Machine Learning. Medium Corporation. https://medium.com/@Mandysidana/machine-learning-types-of-classification-9497bd4f2e14 4.11.2019

Technopedia. 2019. Reinforcement learning. Techopedia Inc. https://www.techopedia.com/definition/32055/reinforcement-learning 4.11.2019

Torre, C. 2019. Announcing ML.NET 1.4 general availability (Machine Learning for .NET). Microsoft. https://devblogs.microsoft.com/dotnet/announcing-ml-net-1-4-global-availability-machine-learning-for-net/ 2.12.2019

Torre, C. 2019. What is ML.NET 1.0 – Machine Learning for .NET. Microsoft. https://devblogs.microsoft.com/cesardelatorre/what-is-ml-net-1-0-machine-learning-for-net/ 5.11.2019

Venkatesh, M. 2017. Why Does Machine Learning Require So Much Training Data? Hacker Noon. https://hackernoon.com/why-does-machine-learning-require-so-much-training-data-cc839cd62fa5 23.2.2020