Kokonaan itse tehty tekoäly

Lappeenrannan–Lahden teknillinen yliopisto LUT

Tietotekniikan kandidaatintyö

2024

Aleksi Levanen

Tarkastaja(t): Apulaisprofessori Jussi Kasurinen

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUTin insinööritieteiden tiedekunta

Tietotekniikan koulutusohjelma

Aleksi Levanen

Kokonaan itse tehty tekoäly

Tietotekniikan kandityö 2024

24 sivua, 6 kuvaa ja 1 liite

Tarkastaja: Apulaisprofessori Jussi Kasurinen

Avainsanat: Tekoäly, Koneoppiminen, Ohjattu oppiminen, Regressio, Lineaarinen regressio

Tiivistelmä

Tekoäly on kiinnostava ja nopeasti kehittyvä teknologia. Tekoäly perustuu mahdollisuuteen antaa koneille kognitiivisia kykyjä. Tässä työssä tutkitaan mitä tekoäly on ja miten sellainen ohjelmoidaan kokonaan itse ilman eri koneoppimisen kirjastoja.

Tutkimus aloitettiin tekoälyn ja koneoppimisen historialla ja nykytekniikan perusteiden esittelyllä. Tutkimuksessa rajattiin jo alussa käytettävä teknologia ohjattuun oppimismalliin ja regressio algoritmiin. Tutkimus toteutettiin empiirisenä tapaustutkimuksena, jossa rakennetaan tekoäly malli, joka ennustaa asuntojen neliöhintaa.

Tutkimuksessa rakennetaan onnistuneesti tekoäly malli, joka saavuttaa kognitiivisia kykyjä ja hyvän tarkkuuden. Tutkimuksen lopussa nostetaan asioita ylös, jotka voisivat parantaa mallin kognitiivisia kykyjä ja mietitään mitä tekoälyn rakentamiseen vaaditaan ja mitä haasteita prosessissa oli.

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

School of Engineering Science

Degree Programming in Software Engineering

Aleksi Levanen

100% Self-made AI

Bachelor's thesis 2024

24 pages, 6 figures and 1 appendix

Examiner: Associate Professor Jussi Kasurinen

Keywords: AI, Machine learning, Supervised learning, Regression, Linear regression

Abstract

Artificial intelligence is an interesting and quickly developing technology. AI is based on the possibility of giving machines cognitive abilities. In this thesis, we will research what AI is and how you program one without machine learning libraries.

The thesis started by introducing AI and machine learning history and modern technologies foundation. In the beginning, research was delimited to only use supervised learning and regression. The research was conducted as an empirical case study where an AI model is built that can predict house prices per square.

In the thesis, the AI model is built successfully, which achieves cognitive functions and good accuracy. At the end of the research, we raised issues that could improve the model's cognitive abilities and thought about what is required to build artificial intelligence and what challenges there were in the process.

Sisällysluettelo

1. Johdanto

Tekoälyä (Artificial Intelligence) on kiinnostava ja nopeasti kehittyvä teknologia. Tekoälyn eri sovellukset voivat määritellä eri kiinnostuksemme kohteet hyvin tarkasti ja antaa mahdollisia muita kiinnostuksemme herättäviä kohteita. Erilaiset tekoälyn sovellukset pystyvät ennustamaan parhaita siirtoja shakissa tai arvioimaan potilaalle hoitotarpeen. Tekoäly on teknologiaa, joka antaa koneelle mahdollisuuden kognitiiviselle toiminnalle. Se voi esimerkiksi hahmottaa eri kuvioita tai oppia erilaisista tietojoukoista tiedon ominaisuuksia tai malleja. Tekoälyä on edistänyt huomattavasti lähiaikoina seuraavat kolme asiaa: Tiedon (Data) määrän valtava kasvu yhteiskunnassa, kehitykset algoritmeissa, laskentateho on kasvanut jatkuvasti, ja tiedon tallennustilan hinta on laskenut. Tekoälystä on tullut merkittävä apuri tiedon käsittelyssä. Se pystyy käsittelemään suuria määriä tietoa ja tekemään ennustuksia ja päätöksiä tiedon perusteella. [1], [2]

Eri yritykset tai organisaatiot ovat alkaneet tutkimaan ja käyttämään tekoälyä voiton tai hyödyn tavoittelemiseksi. Yrityksien tai organisaatioiden osa-alueella ei ole väliä, sillä tekoälyä on käytetty melkein kaikilla aloilla kuten lääketieteessä, teollisuuden eri aloilla, taloudessa. Tekoäly on helpottanut monien ihmisten elämää vähentämällä työtehtäviä missä on toistavia askeleita, tai analysoimalla tehokkaasti eri tietoja. Tekoäly ei tuo pelkästään hyötyjä vaan sisältää myös riskejä, kuten työpaikkojen vähentymistä. Tekoäly myös katsoo asioita täysin objektiivisesti numeroista, joka voidaan nähdä tai kokea huonona asiana päätöksen teossa. Tekoälyyn liittyy myös väärinkäsityksiä, kuten henkilöt, jotka ovat tekemisissä tekoälyn kanssa menettävät kosketuksensa tunteisiin, koska työskentelevät ympäristössä, jossa olisi vähemmän mahdollisuuksia kokea tai näyttää tunteita. [3]

1.1. Tavoitteet ja rajaukset

Tämän työn tavoitteena on tutkia, kuinka tekoälysovellus ohjelmoidaan ja mitä haasteita sen ohjelmointiin liittyy. Tavoitteena on saada tekoäly toimimaan ainakin ajatustasolla. Lisäksi tekoäly yritetään rakentaa toimimaan myös käytännössä. Toteutuksessa käytetään Python-ohjelmointikieltä. Python on korkean tason tulkittu

ohjelmointikieli, joka pystyy hyödyntämään systeemi-tason kirjastoja (Library), jotka lyhentävät laskenta aikaa huomattavasti [4, s. 2]. Työssä käytetään NumPy-kirjastoa, koska se yksinkertaistaa, helpottaa ja nopeuttaa matriisien käsittelyä [5]. Työssä tekoäly oppii ohjatulla opetuksella. Tietopaketit, joilla opetetaan ohjatun oppimisen malleja, on helposti saatavaa. Ohjatun oppimisen käyttäminen on myös helpompaa asian opiskelijalle. Työssä käytetään valmista tietopakettia, koska sellaisen rakentaminen itse veisi liikaa aikaa ja resursseja ja voisi vaikuttaa tekoälyn tarkkuuteen, jos tiedossa esiintyisi liikaa virheitä. Lineaarinen regressio ja oppimismalli valittiin työn alussa, koska muuten työstä tulisi liian laaja tutkiessa eri rakennemahdollisuuksia.

Työn tarkoituksena ei ole ohjelmoida parasta mahdollista tekoälyä niin nopeuden kuin tarkkuuden kannalta, vaan saada toimiva ratkaisu. Työssä pyritään välttämään muita Pythonin koneoppimisen integroituja ohjelmoinnin tukiympäristöjä (Framework) tai kirjastoja.

Työn tutkimuskysymykset ovat seuraavat:

1. Miten tekoäly ohjelmoidaan?
2. Mitä haasteita prosessissa oli?

1.2. Työn rakenne

Työ koostuu kokonaisuudessa 7 kappaleesta. Ensimmäisessä kappaleessa kerrotaan työstä ja miksi työ tehdään. Kappaleessa myös esitellään tutkimuskysymykset ja rajataan tutkimusta. Työn toisessa kappaleessa esitellään tutkimukseen liittyvää kirjallisuutta, teoriaa ja termistöä. Kolmannessa kappaleessa esitellään työn tutkimusmenetelmä ja miten tutkimus suoritetaan. Seuraavassa kappaleessa käsitellään, että miten tekoälymalli rakennetaan, kerrotaan mitä koodi tekee ja tarkastellaan, että miten tekoälymalli suoriutuu sille annetusta tehtävästä. Viidennessä kappaleessa keskustellaan tutkimuksesta ja sen tuloksista. Kuudennessa kappaleessa tiivistetään tutkimuksen eri vaiheet, jonka jälkeen viimeisenä kappaleena on lähdeluettelo.

2. Kirjallisuuskatsaus

Tässä kappaleessa käydään läpi mitä tekoäly ja koneoppiminen pitävät sisällään. Ensin käydään läpi mistä termi tekoäly on tullut ja mitä tekoäly on. Toisessa kappaleessa kerrotaan koneoppimisen yleisesti ja käydään läpi mihin alueisiin koneoppiminen jakautuu. Kolmannessa kappaleessa koneoppimisen algoritmeista ohjatulle oppimiselle, kuten luokittelusta ja regressiosta. Neljännessä kappaleessa avataan yleisiä termejä tutkimukseen liittyen.

2.1. Tekoäly

Ensimmäinen tunnettu tieteellinen artikkeli tekoälystä on Alan Turingin julkaisema “Computing Machinery and Intelligence” 1950-luvulla, jonka myötä tekoäly siirtyi tieteisfiktiosta todellisuuteen [2, s. 6]. Artikkelissa ehdotetaan keinoa kysymykseen ”voivatko koneet ajatella?”, jonka esittää työnsä alussa. Vaikka tekoälyä, joka onnistuisi läpäisemään A. Turingin ehdottaman testin ei ole vielä tehty, tutkimus on laajentanut käsitystä mitä tekoäly voi olla ja johtanut tutkimusta eteenpäin nykypäivänäkin. Termi tekoäly tuli käytäntöön vuonna 1956 John McCarthyn ehdottamana [1, s. 2].

Yleensä tekoäly jaetaan kahteen eri ryhmään, heikkoon ja vahvaan tekoälyyn. Heikko tekoäly kykenee esimerkiksi kyetä ennustamaan yhden pienen ongelman ratkaisun tai ehdotuksen ratkaisuun. Heikko tekoäly kykenee ratkaisemaan ongelman saamalla sensori dataa kyseisestä ongelmasta. Suuri osa ellei kaikki nykyajan tekoäly ratkaisut ovat heikkoja tekoälyjä. Vahva tekoäly taas vastaisi ihmisen tasoisia kognitiivisia kykyjä ja tietenkin varustettu tietokoneen laskentateholla. Tämän takia vahvan tekoälyn kehittämiseen liittyy esimerkiksi eettisiä ongelmia. Vahvaa tekoälyä ei ole vielä saavutettu. [6, s. 62]

Tekoäly on laaja käsite, jota on vaikea tiivistää. Tekoälytutkimus perustuu kognitiotieteisiin, joka on tullut yrityksestä ymmärtää ihmisälyä laskennallisen näkökulman kautta, kun taas tekoäly on tullut yrityksestä ymmärtää ihmisen ajattelua. Yleensä tekoälyn älykkyyttä ei testata psykologisilla älykkyyden kokeilla vaan, miten suoriutuu älykkyyttä vaativissa tehtävissä kuten, esimerkiksi musiikki listojen ennakoinnissa, eri liikennemerkkien luokittelussa tai ajaessa autoa

itsenäisesti. Näihin tehtäviin harjautuvat tekoälyt ovat heikkoja tekoälyjä. Tekoälytutkimus keskittyy vahvasti koneellisten tekoälyjärjestelmiin, joissa keskitytään psykologisen älykkyyden ulkopuolelle. Kyseiset ratkaisut voivat vaatia laajemman valikoiman kognitiivisia kykyjä esimerkiksi konenäkö ja sen sensoridatan luokitteluun ja prosessointiin vaadittavaa päättelykykyä. [7, ss. 42–43]

2.2. Koneoppiminen

Koneoppimisen termi tuli käytäntöön vuonna 1959 Arthur Samuelin ehdottamana. Koneoppiminen tavoittelee saavuttamaan tarvittavan älykkyyden aiemmin mainittuun testiälykkyyden saavuttamiseksi. Milloin ohjelma pystyy ratkaisemaan sille annetun ongelman aiemman oppimisen pohjalta ja tehdä älykkyyteen pohjautuvan ratkaisun. Koneoppiminen perustuu matemaattisiin algoritmeihin, joiden pohjalta kyseisen ohjelma oppii sille annetusta tiedosta. [1, ss. 7–8, 69–70] Suurin osa tekoäly ratkaisuista pohjautuu koneoppimiselle [2, s. 6]. Koneoppiminen voidaan jakaa kolmeen yleisimpään osa-alueeseen: ohjattuun koneoppimiseen, ohjaamattomaan koneoppimiseen sekä vahvistusoppimiseen. Seuraavassa kappaleissa käymme läpi lyhyesti miten kyseiset oppimismallit toimivat. [8, ss. 106–107]

2.2.1. Ohjattu oppiminen

Ohjattu oppiminen (supervised learning) on valvottua oppimista. Ohjatussa oppimisessa algoritmi opetetaan tiedolla, jossa tiedetään halutut arvot. Algoritmi oppii miten tiedon eri muuttujat korreloivat haluttuun arvoon, jonka jälkeen ohjelmaa ohjaa ennustukseen vaikuttavia arvoja. Algoritmi opettelee, kunnes saavuttaa halutun tarkkuuden, jonka jälkeen algoritmille voidaan ajaa tietoa, josta ohjelma sitten tekee ennustuksia. [1, ss. 72–73], [8, ss. 105–107], [9, s. 33]

Ohjatusta oppimista hieman tarkemmin. Tieto ohjatussa oppimisessa yleensä jakautuu ennustukseen vaikuttaviin tekijöihin x_i. Halutusta ennustettavasta tiedosta voitaisiin käyttää termiä y_i. Lopputuloksena mallia harjoitetaan (x_i , y_i), jossa x_i on vektori eri muuttujista ja i kuvastaa jokaista eri tieto riviä kyseisen muuttujan kolumnista. [1, ss. 90–91]

2.2.2. Ohjaamaton oppiminen

Ohjaamaton oppiminen (Unsupervised learning) on oppimista missä kone oikeasti opettelee itse. Ohjattuun oppimiseen eroten ohjaamattomassa oppimisessa tietoa ei merkitä vaan koneen pitää itse päätellä tiedosta eri samankaltaisuudet tai yhteydet. Suosittuja käytännön ratkaisuja ohjaamattomalla oppimisella ovat suositteluratkaisut ja kuluttajien käyttäytymistä selvittävät data-analyysit. [1, ss. 74–75], [8, ss. 105–107]

2.2.3. Vahvistusoppiminen

Vahvistusoppiminen (Reinforced learning) on oppimista, jossa kone oppii positiivisen ja negatiivisen palautteen kautta. Ohjelma kokeilee eri ratkaisuja opiskelu ympäristössä, jonka jälkeen oppii palautteesta halutut päätökset. Vahvistusoppimista hyödyntävät koneoppimisen ratkaisut ovat suosittuja tekoälyissä, jotka esimerkiksi laitetaan pelaamaan videopelejä. [8, ss. 105–107]

2.3. Algoritmit ohjatulle oppimiselle

Koneoppimisen algoritmeja on runsaasti eri oppimismalleille kuten henkilöiden Panesar [1] ja Mahesh [10] töissä tuodaan esille. Seuraavissa alakappaleissa kerrotaan vain yleisimmistä ohjatulle oppimiselle olemassa olevia algoritmeja.

2.3.1. Luokittelu

Luokittelu on prosessi, jolla tiedon eri jäsenet luokitellaan eri ominaisuuksien mukaan eri luokkiin. Luokittelu prosessi voi muuttua hieman riippuen, miten luokitellaan ja mihin. Luokittelu algoritmeja ovat: päätöspuut (decision trees), naïve Bayesian, logistinen regressio (logistic regression), k:n lähin naapuri (kNNs) ja tukeva vektorikone (suppot vector machine). Luokittelu algoritmit ovat hyviä lääketieteessä esimerkiksi tapauksissa: kuvien lajitellussa, potilaiden riskien tai hoitotarpeiden luokittelussa ja profiloinnissa. Panesar [1, ss. 91–92] kertoo myös, että kyseisissä esimerkeissä tekoäly ratkaisut osaavat myös tehdä johtopäätöksiä luokittelun perusteella. Tästä huomataan, että yhteiseksi tekijäksi eri käyttöratkaisuille tulee luokittelu, jonka pohjalta voidaan tehdä ratkaisuja, joita on käytetty ryhmän muihin henkilöihin ja koettu toimiviksi.

2.3.2. Regressio

Regressio on prosessi, jolla numeerisen tiedon eri jäsenistä havaitaan matemaattisia yhtäläisyyksiä tai suhteita. Näistä kuvioista saadaan asiaan selkeyttä ja voidaan ennustaa numeerisella tiedolla tulevia tapahtumia. Regressio keskittyy ennustettavan arvon ja ennustukseen vaikuttavien arvon yhteyden täsmentämiseen. [11, ss. 167–168] Regressio algoritmeihin kuuluu: lineaarinen regressio, regressio puut (regression trees), tukeva vektorikone (suppot vector machine), k:n lähin naapuri (kNNs) ja synapsi (perceptrons) [1, s. 92]. Regressio algoritmit ovat hyviä ratkaisemaan ongelmia, jossa ennustetaan jotakin aiemman tiedon perusteella. Esimerkiksi: milloin seuraava henkilö käyttää toimiston tulostinta, arvosanaa opiskelutunneista ja ennustamaan hoitohistoriasta erilaisia asioita kuten tulevia terveydellisiä komplikaatioita tai kuolleisuusriskiä. [1, ss. 92–93]

2.4. Yleiset Termit

Seuraavissa kappaleissa avataan yleisiä termejä, jotka ovat tärkeitä tekoälyä suunnitellessa ja ohjelmoidessa.

Artefakti

Artefakti on termi ohjelmistokehityksen kontekstissa sen sivutuote, joka sisältää artefaktin määrittelevät asiat. Artefakti tässä työssä on synonyymi tekoälymallille. Gillis [12] kertoo artikkelissaan myös, että artefaktit jaetaan kolmeen eri ryhmään: ohjelmointi keskeisiin, projektin hallinta keskeisiin tai dokumentaatio keskeisiin.

Kohina

Kohina (Noise) on mitattujen arvojen satunnaisvarianssivirhe. Oikean maailman esimerkeissä kohinaa on enemmän. Yleensä harjoittelua varten tehdyissä tietojoukoissa on pienempää kohinaa. Työssä käytetyssä tietopaketista ei ole kerrottu mistä tiedot ovat peräisin, eikä tietopaketissa esiinny kaikki asunnon hintaan vaikuttavat tekijät, jolloin tiedossa esiintyy enemmän kohinaa mikä heikentää ennustuksen tarkkuutta. [13, ss. 1–2]

Korrelaatio

Korrelaatio analyysin avulla voidaan tutkia kuinka vahva lineaarinen tai ei lineaarinen vaikutus tiedon eri tekijöillä on lopputulokseen tai toisiinsa. Täydellisesti korreloivat arvot ovat korrelaatiokertoimet -1 tai 1, kun arvot, joilla ei ole korrelaatiota on korrelaatiokerroin 0. Pearsonin korrelaationkerroin voi olla [-1,1] välistä, jossa tuloksen ollessa lähempänä raja-arvoja kertoo suuremmasta korrelaatiosta. Suurempi korrelaatio tarkoittaa, että ennustettu arvo tiedon perusteella on tarkempi, jolloin tuloksessa on vähemmän ääntä. Pearsonin korrelaation kaava on $r = \frac{n(\sum xy)-(\sum x)(\sum y)}{\sqrt{(n\sum x^2-(\sum x)^2)(n\sum y^2-(\sum y)^2)}}$. Kaavassa r on korrelaatio kerroin, n on rivien määrä, x ja y ovat tarkasteltavat sarakkeet. [14]

Epäkeskisyys

Epäkeskisyys (Bias) viittaa tekoälymallin virheeseen, jossa mitataan kuinka kaukana oletetut ennustusarvot olivat oikeista arvoista. Matala epäkeskisyys viittaa mallin parempaan ymmärrykseen tietopaketista ja korkea epäkeskisyys mallin huonompaan ymmärrykseen tietopaketista. [1, s. 157]

Varianssi

Varianssi (Variance) viittaa tekoälymallin tarkkuus virheeseen, jossa katsotaan kuinka paljon ennustukset eroavat toisistaan. Matalalla varianssilla ennustukset ovat lähellä toisiaan ja korkealla varianssilla ennustukset ovat kaukana toisistaan. [1, s. 157]

Ylisovittaminen

Ylisovittaminen (Overfitting) viittaa kun algoritmissa esiintyy virhettä kohinan vaikuttaessa liikaa useisiin ennustuskertoimiin, jolloin algoritmi ei opi tiedon eri muuttujien välisiä suhteita tai signaaleja. Ylisovittamista voidaan välttää esimerkiksi vähentämällä muuttujia tiedosta. [1, s. 139]

3. Tutkimusmenetelmä

Tässä työssä tutkitaan tekoälyn rakentamista, ilman tekoäly kirjastoja tai integroituja ohjelmoinnin tukiympäristöjä. Tutkimuksessa rakennetaan toimiva tekoäly artefakti tutkitun teorian pohjalta, joten tutkimusmenetelmänä voidaan käyttää empiiristä tapaustutkimusta.

Runeson [15, ss. 11–13] tiivistää empiirisen tapaustutkimuksen tutkimukseksi ohjelmistotekniikassa, joka käyttää monipuolisesti todisteita tutkiakseen tarkasti yhtä tapausta tai pientä määrää esiintymiä nykyaikaisessa ohjelmistotekniikassa. Tutkimus sijoittuu tapahtumaan sen todellisessa elinympäristössä, erityisesti silloin, kun ilmiön ja kontekstin välistä rajaa ei voida määritellä selkeästi.

Tapaustutkimukset voidaan toteuttaa ilman kontrollia, jolloin tutkittavaa asiaa voidaan tarkastella oikeassa ympäristössä ja kontekstissa. Tapaustutkimukset ovat myös oikea menetelmä tutkimuskysymyksille, jotka yrittävät vastata ’miksi’ ja ’miten’ alkaviin kysymyksiin, kuten tämän työn ensimmäiseen kysymykseen ”miten tekoäly ohjelmoidaan”.

Robson [16, s. 61] mukaan tutkimusmenetelmällä voi olla 3 erilaista tarkoitusta: tutkiva (exploratory), selittävä (explanatory) ja kuvaileva (descriptive). Runeson [15, s. 13] tiivistää tutkimuksen, joka on tutkivaa seuraavasti: Tutkimuksessa etsitään mitä tapahtuu, miksi ja kehitetään uusia ideoita tai hypoteeseja tulevia tutkimuksia varten. Tutkiva tutkimus on hyvä valinta tälle tutkimukselle, sillä työssä tutkitaan tekoälyn ohjelmointia ja mitä prosessista nousee esille. Työn lopussa myös keskustellaan mahdollisesti paremmista ratkaisuista tai prosessin vaiheista.

Tutkimuksessa käytetään ohjattua oppimista, koska työssä lähdettiin rakentamaan tekoälyä oppimiskokemuksena, valittiin työhön hyvä ja helppo oppimismalli opettelulle. Ohjatun oppimisenmallilla suurimpana etuna on merkitty tietopaketti (Labeled data), joka auttaa mallin opettamisessa ja tarkastelussa. [10, s. 381] Työhän koneoppimisen algoritmiksi valittiin lineaarinen regressio, koska on ohjatun oppimisen algoritmeista yleisin ja pystyy ennustamaan melkein mistä vain ilmiöstä. Lineaarinen regressio nousi regressio algoritmeista, sen monipuolisuuden takia [1, s.

150]. Heikkouksia algoritmilla on, että tiedon pitää olla numeerisessa muodossa, se ei pysty käsittelemään puuttuvia tieto kohtia ja tekee isoja oletuksia. [11, s. 177] Ohjattu oppiminen tarvitsee tietopaketin mikä sisältää merkittyä tietoa eli ennustettavat arvot ovat mukana tietopaketissa, eli käytettävän tietopaketin pitää sisältää ennustettavat arvot. Tietopaketin valinnassa vaikutti myös, että tieto olisi puhdasta eli ei sisältäisi tyhjiä kohtia ja kohina olisi pientä. Tietopaketista oli tehty aiempia tekoäly malleja, joissa esimerkiksi pythonin SciPy kirjastoa [17] hyödyntäen oltiin päästy hyviin tuloksiin.

Tutkimuksessa siis yritetään rakentaa tekoäly artefakti, joka saavuttaa tekoälyn määritelmän. Tekoälyn määritelmä on hyvinkin yksinkertaisesti ohjelma, joka pystyy ratkaisemaan ongelman, joka vaatii älykkyyttä. Esimerkiksi tunnistamaan kuvioita tai pelaamaan shakkia. [7, s. 42] Seuraavassa kappaleessa aletaan kehittämään tekoälymallia ja perehdytään tietopakettiin paremmin.

4. Tekoälyn kehitys

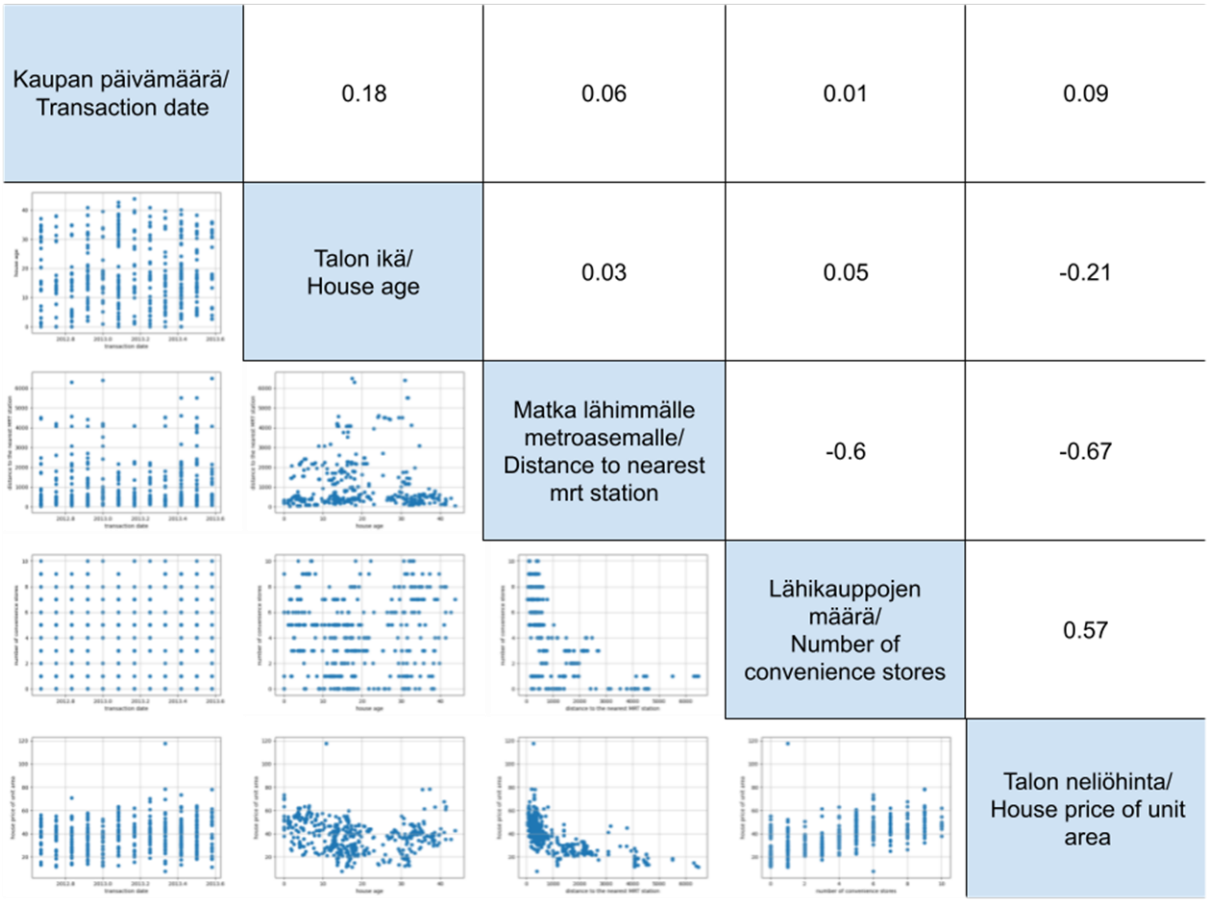
Tässä kappaleessa käydään läpi ensin tietopaketin tiedon eri suhteita, jonka jälkeen kerrotaan mitä tekoäly tekee kyseisellä tiedolla ja miten tekoäly toimii teoriassa. Toisessa kappaleessa käydään läpi, miten suunnitellut vaiheet toteutetaan koodissa. Kolmannessa kappaleessa tarkastellaan tekoälymallin tuloksia.

4.1. Tekoälyn suunnittelu

Työssä käytetään valmista listaa asunnon hintaan vaikuttavista tekijöistä [18]. Tietopaketti on avointa dataa sivustolla, jossa jaetaan harjoittelu dataa koneoppimisen algoritmeille. Kyseisellä tiedolla on hyvät arvostelut ja data on valmiiksi täysin numeerisessa muodossa, mutta kyseiseltä tietopaketilta puuttuu lähde. Tietopaketti valittiin kumminkin hyvien arvostelujen takia ja aiempien käyttäjien perusteella, sillä tietopaketista ei ollut tyhjiä kohtia, eikä kohinaa. Pitää kuitenkin ottaa huomioon, että oikean maailman tapauksissa pitää osata ottaa myös muita asioita huomioon, kuten huoneiden määrä tai asunnon tyyppi. Tietopaketti koostuu seuraavista sarakkeista: Indeksi, kauppapäivämäärä, talon ikä, matka lähimmälle metroasemalle, lähikauppojen määrä, korkeusaste (koordinaatisto), leveysaste (koordinaatisto) ja talon neliön hinta. Ennustettava arvo on talon neliön

hinta. Arvoista leveys- ja korkeusaste eivät vaikuta hintaan lineaarisesti, joten nämä kaksi saraketta voidaan jättää huomioimatta tekoälyn ennustukseen vaikuttavista arvoista.

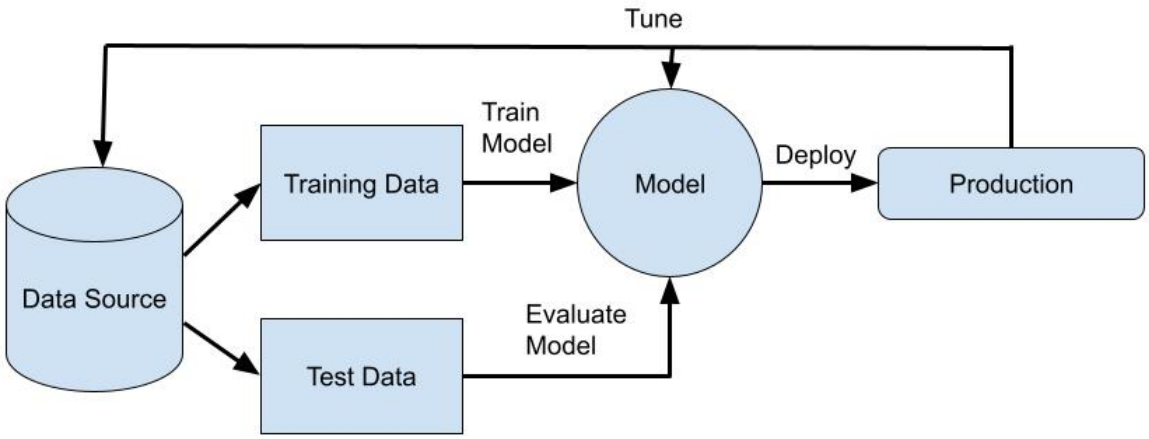
Tietopaketista rakennetaan sirontakaaviomatriisi, jossa tehdään visuaaliset mallinnukset tiedon eri sarakkeista. Diagonaali sisältää nyt histogrammit, jotka kuvaavat kullekin ominaisuudelle kulkevien kulkuarvojen jakautumista. Diagonaalin yläpuolella olisi vain kuvaajat käänteisenä, joten sinne voidaan laittaa korrelaatio arvot. Alempana rakennettu esimerkki sirontakaaviomatriisista kuva 1. [11, s. 188]



Kuva 1, Sirontakaaviomatriisi

Sirontakaaviomatriisista (kuva 1) nähdään että matka metroasemalle ja lähikauppojen määrä korreloi maltillisesti talon neliömetrin hintaa eli ennustettavaa arvoa. Talon ikä korreloi heikosti ja kaupan päivämäärä ei korreloi ollenkaan ennustettavan arvon kanssa. Tässä kohtaa huomataan, että korrelaatio ei ole

merkittävä tietopaketissa. Oikean maailman tapauksissa tässä kohtaa tietopakettiin voitaisiin lisätä muita asiaan vaikuttuvia tekijöitä, kuten huoneiden määrää tai onko asunto kalustettu tai matka lähimpään ostoskeskukseen. Tässä tutkimuksessa jatketaan kumminkin tietopaketin käyttämistä tutkimuksessa.



Kuva 2, Ohjatun oppimisen toimintakaavio [11, s. 381]

Tietopaketti jaetaan harjoittelu- ja testauslistoihin kuvan 2 mukaan. Panesar [1, ss. 153–154] lajittelee jakoalgoritmit kolmeen eri vaihtoehtoon seuraavasti: Itse määrittelemä osuus siirretään tietopaketista testauslistaan ja loput toimivat harjoituslistana (hold-back method), tietopaketti taitetaan samankokoisiin listoihin, joista yksi lista jätetään testaamista varten ja muilla harjoitutetaan mallia (n-fold cross-validation) ja viimeisä on tietopaketin jako satunnaislukugeneraattorilla (Monte Carlo cross-validation). Harjoittelu lista on mallin opettamista varten ja testaamiseen tarkoitetulla listalla nähdään, kuinka hyvin malli suoriutuu sille annetusta tehtävästä [10, s. 381]. Työssä valittiin tietopaketin jako samankokoisiin listoihin, joista yksi jätetään sivuun testaamista varten. Metodi valittiin, koska määrää kuinka monta kertaa alkuperäinen lista jaetaan, on tällöin helppo muuttaa, jolloin mallia voidaan tarkastella pienellä ja suurella määrällä opetus tietoa.

Lineaarinen regression tavoite on yksinkertaisesti katsottuna sijoittaa lineaarinen suora datapisteiden mukaisesti niin että jokaisen pisteen etäisyys suoraan on summattuna mahdollisimman pieni. Puhutaan keskiarvoisesta neliö virheestä. Lineaarisessa regressiossa puhutaan yleensä yksinkertaisesta ja moninkertaisesta. Yksinkertaisella on vain yksi tulokseen vaikuttava arvo x ja yksi tulos y.

Moninkertaisella voi olla 2 tai useampi x:n arvo ja yksi tulos y. [11, ss. 177–179], [19, ss. 57–58]

Tekoälymallin lineaarinen regressio toteutetaan seuraavasti kaavalla $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_i x_i + \varepsilon$, jossa y on eri x ja β tulojen summa. x_i on [transaction date, house age, distance to nearest MRT station, number of convenience stores]. β_i on ennustettu kerroin. α on y:n oletettu arvo kun kaikki x:n arvot ovat nolla. y voidaan myös kuvata β_0 ja sille annetaan termi x_0 ja sen arvoksi 1 jokaiselle matriisin riville. Kaavaksi muodostuu $y = \beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_i x_i + \varepsilon$. Kaava voidaan esittää simppelisti $Y = \beta X + \varepsilon$, jossa nyt Y on matriisi y:n arvoista, X on matriisi x_i:n jokaisesta arvosta, ja β on matriisi kaikista ennustuskertoimista. β voidaan laskea kaavalla $\beta = (X^T X)^{-1} X^T Y$. [11, ss. 177–179] Kyseinen kaava voidaan muotoilla seuraavasti $(X^T X)\beta = X^T Y$. Kyseisestä kaavasta voidaan ratkaista ennustuskerroin matriisi NumPy-kirjaston avulla joka laskee ”tarkimman” mahdollisen tulos matriisin, jolla yhtälö pitää paikkaansa [20]. Kun ensimmäinen versio valmistuu tekoälymallista, voidaan mallia tarkastella esimerkiksi seuraavin keinoin: Varianssin, epäkeskisyyden ja virheprosentin mukaan.

4.2. Tekninen toteutus

Kappaleen koodi on saatavilla liitteestä 1. Koodissa tietopaketin lukeminen ja jakaminen tapahtuu samassa aliohjelmassa. Aliohjelma ottaa vastaan tiedoston nimen ja tietopaketin jakoa kuvaavan numeron. Ohjelma jakaa jokaisen rivin joko testaus- tai opetusmatriisiin. Jako tapahtuu tarkastelemalla jaon aikana numeroa, joka kasvaa joka tiedosto rivin jälkeen. Kyseisestä numerosta otetaan jakojäännös jaettaessa tietopaketin jakavalla numerolla. Jos jakojäännös on nolla, silloin rivi menee testausmatriisiin, muuten rivi menee opetusmatriisiin.

Koodissa on myös tehty aliohjelma, joka ottaa vastaan matriisin ja jakaa sen ennustukseen vaikuttaviin arvoihin eli X matriisiin ja Y matriisiin, joka on ennustettava arvo.

Ennustuskertoimet laskeva aliohjelma jakaa aiemmassa kappaleessa mainitulla aliohjelmalla opetusmatriisin X ja Y matriisiin. Ennustuskertoimia laskettaessa

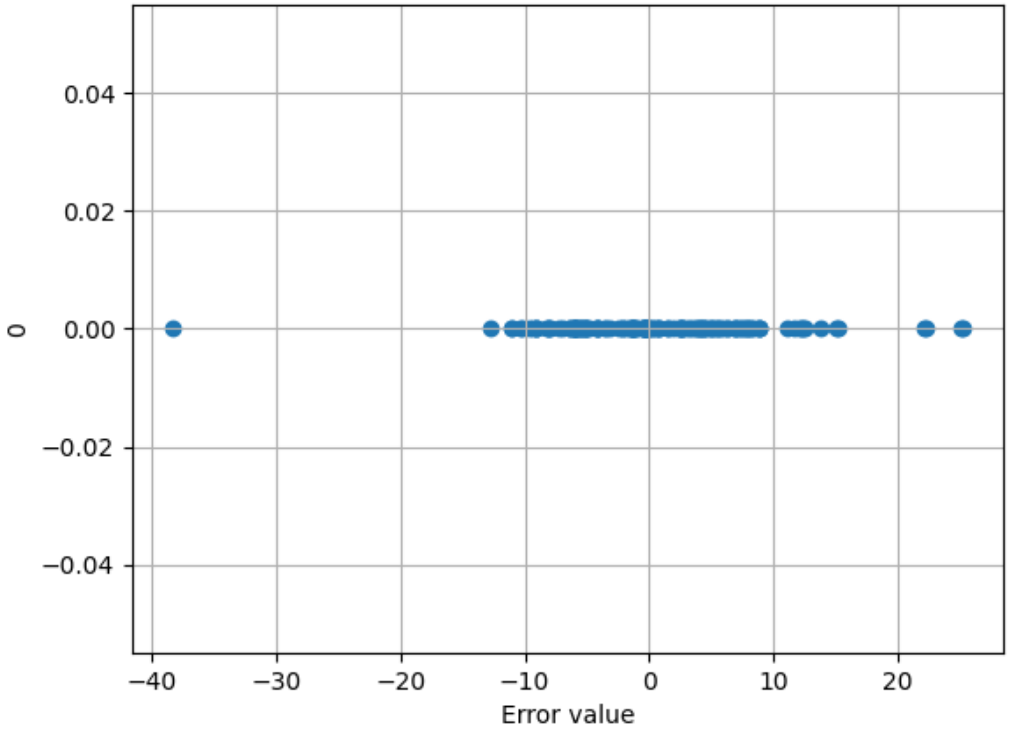
käytetään NumPy kirjaston lineaarisen algebran ratkaisijaa. Kirjaston aliohjelma ottaa vastaan 2 matriisia vaikka a ja b. Ratkaistavan lineaarisen matriisi yhtälön pitää olla ax = b muodossa. Koodissa a matriisi on tulomatriisi $(X^T X)$ ja b on tulomatriisi $X^T Y$. tällöin voidaan ratkaista ennustuskerroin matriisi x aiemmin mainitun kaavan mukaan $(X^T X)\beta = X^T Y$. Kyseisessä kaavassa matriisi x, joka saadaan lineaarisen algebran ratkaisijasta on ennustuskerroin matriisi β. Koodissa aliohjelma palauttaa pääohjelmalla ennustuskerroin matriisiin.

Mallin tarkastamista varten tehtiin aliohjelma, joka ottaa ennustuskertoimet matriisin muodossa ja testattavan matriisin. Testattava matriisi jaetaan X ja Y matriisiin. X matriisi kerrotaan ennustuskerroin matriisilla, jonka jälkeen tulomatriisi on ennustetut arvot. Ennustettua matriisia voidaan vertailla Y matriisiin. Aliohjelma laskee virheprosentin ja tallentaa listaan virheen määrän vähentämällä ennustetun arvon oikeasta arvosta, jonka jälkeen listan virhearvot toimivat kuvaajassa x arvoina ja y arvo on aina 0. Kuvaajasta voidaan tarkastella varianssia ja epäkeskisyyttä.

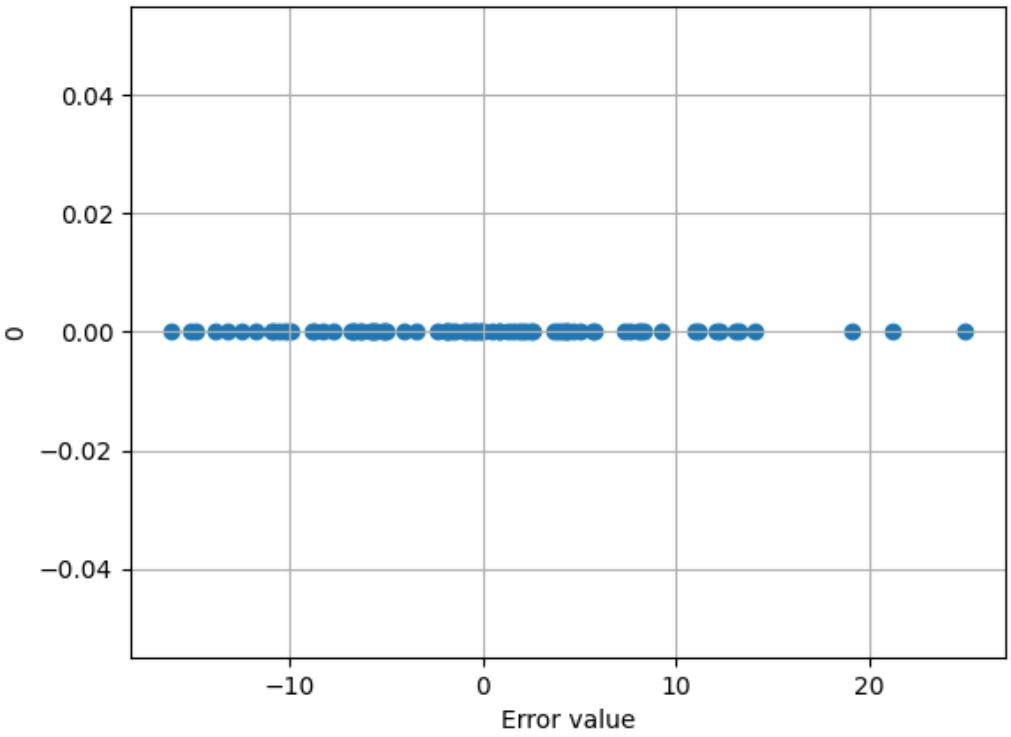
4.3. Tekoälymallin tulokset

Tässä kappaleessa esitellään lyhyesti mallin tuloksia kirjallisesti ja kuvaajilla. Tekoälymallia ajettiin eri testaus- ja opetuslistojen suhteilla. Jaosta voidaan puhua taitoksilla. Tekoälymallia ajettiin 3, 5, 10 ja 20 taitoksella. Käytännössä siis testauslistan koko oli opetuslistaan suhteessa 1:3, 1:5, 1:10, 1:20 eri taitoksilla. [1, s. 154]

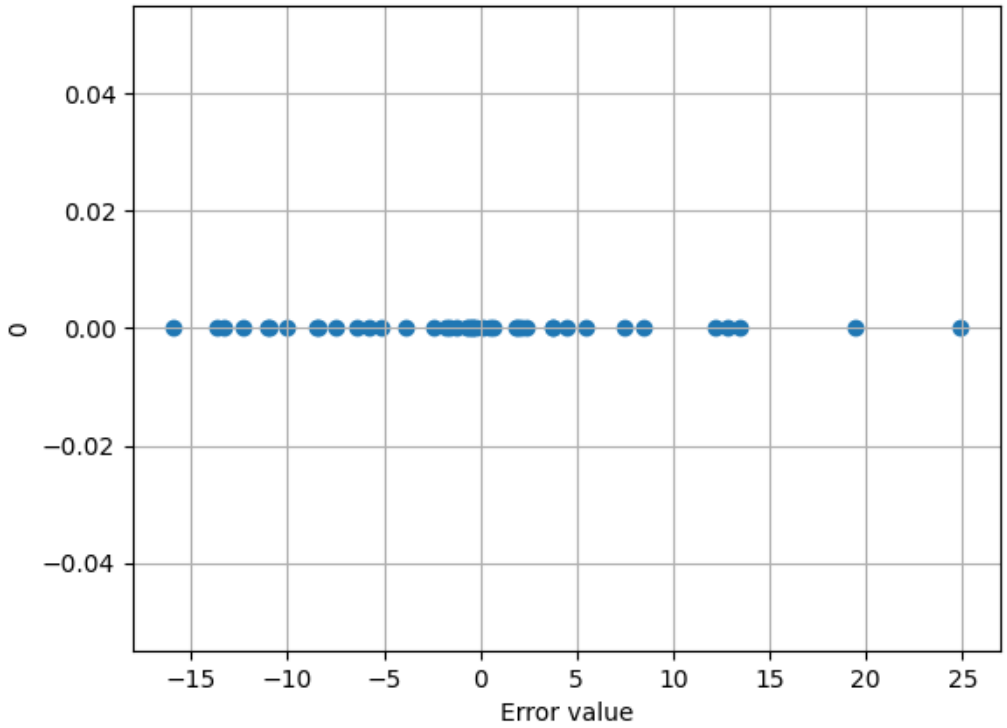
Kuvista 3, 4, 5 ja 6 huomataan että pienemmillä taitos määrillä epäkeskisyys vaikuttaisi olevan pienempi kuin suuremmilla taitosmäärillä. Varianssi on todella hyvä pienillä ja isoilla taitoksilla. Virheprosentti on suurimmillaan 5 ja 10 taitoksen välissä ja vähenee taitos määrien kasvaessa tai pienentyessä.



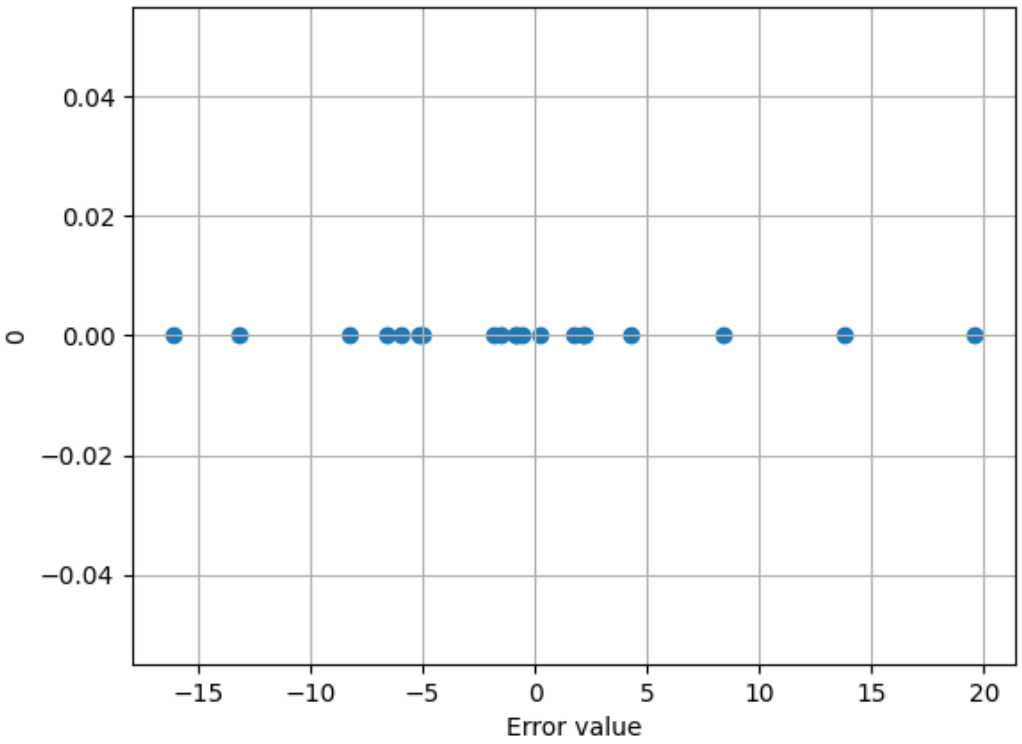
Kuva 3, 3 Taitoista Virheprosentti 20.2 %



Kuva 4, 5 Taitosta Virheprosentti 21.5 %



Kuva 5, 10 Taitosta Virheprosentti 22.3 %



Kuva 6, 20 Taitosta Virheprosentti 15.6 %

5. Keskustelu

Tekoäly saatiin kehitettyä ja toimimaan. NumPy kirjaston käyttäminen nousi työssä tärkeämpään rooliin, kuin aiemmassa vaiheessa tutkimusta odotin. Kyseinen kirjasto oli suunniteltu helpottamaan matriisien laskutoimituksia, kuten matriisien tuloja. Työssä päädyttiin kuitenkin ratkaisemaan ennustuskertoimien laskeminen NumPy kirjaston lineaarisen algebran ratkaisijalla. Tähän ratkaisuun päädyttiin, koska oli tehokkain ja helpompi ratkaisu kuin opetella rakentamaan kyseinen funktio, jolloin säästettiin huomattavasti aikaa ja resursseja.

Tekoälyn tarkkuudessa oli parannettavaa, sillä NumPy-kirjaston lineaarisen algebran ratkaisija laskee parhaimman mahdollisen ennustuskerroin matriisin, jolloin jos tieto sisältää kohinaa saattaa tekoälymalli ylisovittaa ennustuskertoimet [1, s. 139]. Tämä on yleinen ongelma lineaarisissa regressio algoritmeissa ja sen vähentäminen on resursseja kuluttava prosessi, joka kuluttaa enemmän kuin antaa tuloksia näin pienessä tutkimuksessa [1, s. 103].

Tekoäly laskee ennustuskertoimet ja sen jälkeen kokeilee, miten onnistui ennustuskertoimien laskemisessa, mutta ei osaa muuttaa ennustuskertoimia testituloksien mukaan. Tekoäly ei siis osaa muuttaa ennustuskertoimia testituloksien mukaan, jolloin kyseinen ohjelma ei omista kykyä oppia omista virheistä, jolloin malli ei saavuta niin sanottua täydellistä oppimista. Jatkokehitysmahdollisuutena olisi suhteellisen pienin resurssien avulla olisi voitu kehittää mallille tietopaketin jakoon taitosten rajat ja antaa mallin itse valita paras mahdollinen taitosten määrä. Tällöin malli olisi saavuttanut tietystä näkökulmasta paremman älykkyyden. [8, s. 43]

Varianssin ja epäkeskisyyttä voitaisiin parantaa, joko vertailemalla eli koneoppimisen algoritmeja samassa ongelmassa tai laajentamalla ohjelmaa siten, että se ottaa huomioon enemmän ennustukseen vaikuttavia tekijöitä, kuten talon huoneiden määrän, asuinalueen tai etäisyys keskustasta. Tämä kuitenkin lisäisi ylisovittamisen ongelmaa entuudestaan. Paras ratkaisu olisi tähän tilanteeseen valita suurimmat korrelaatio kertoimet keskenään omaavat muuttujat ja opettaa malli niillä tarkastelemaan asuntojen hintoja. [1, ss. 158–161]

Vaikka tutkimus saatiin onnistuneesti valmiiksi, silti ei kaikkiin tavoitteisiin päästy. Tutkimuksessa tekoälylle olisi voinut rakentaa oman lineaarisen algebran ratkaisijan matriiseille. NumPy otettiin työssä käyttöön pääosin matriisien kertomista varten, mutta tutkimuksessa päädyttiin käyttämään kirjastoa myös matriisien lineaarisen algebran ratkaisemiseen. Ohjelma ottaa vastaan tiedon ja menetelmän ja palauttaa kaavan millä ratkaista ongelma mutta oppii joka kerta samalla tavalla, algoritmi tai kaava ei kehity pidemmällä aikavälillä. Aiemmassa kappaleessa myös mainittiin, että tekoälymalli ei saavuttanut niin sanottua täydellistä oppimista, minkä olisin halunnut saavuttaa tutkimuksessa, mutta sen implementointi olisi ollut hyvinkin vaikea ja resursseja kuluttava ominaisuus näin pienessä tutkimuksessa, joten tämä tavoite voidaan tavoittaa isommassa jatkotutkimuksessa. Eli tutkimukseen jäi parantamisen varaa tavoitteiden saavuttamisessa.

Tälle tutkimukselle on myös muita jatkotutkimus mahdollisuuksia. Mallin kehitystä voitaisiin lähteä rakentamaan eri suuntiin. Mahdolliset jatkokehitys olisivat seuraavat: sensoreiden lisääminen, eri algoritmien testaaminen ja vertailu, tai ohjelman muokkaaminen kaupalliseen tarkoitukseen.

Ensimmäisenä tutkimuskysymyksenä oli ”Miten tekoäly ohjelmoidaan?”. Tämän tutkimuksen perusteella tekoälyn ohjelmoimiseksi pitää kyseisin ohjelman saavuttaa älykkyyttä. Tämä yleensä tuotetaan erilaisilla koneoppimisen oppimismalleilla, jotka yleensä perustuvat erilaisiin algoritmeihin. Algoritmi pitää sitten vielä kirjoittaa toimivaksi käytetyllä ohjelmointikielellä. Tutkimuksen aikana huomasin, että suurin osa tekoäly ratkaisuista lähdettiin kehittämään ongelmalle, jolloin eri valinnat kehitysvaiheessa helpottuvat, sillä eri ratkaisut sopivat tietyn tyyppisille ongelmille.

Toisena tutkimuskysymyksenä oli ”Mitä haasteita prosessi oli?”. Tutkimuksessa oli haasteena löytää tietoa, jolla ymmärtäisi hyvin suoraan mitä tekoäly on ja mikä on pienin mahdollinen malli, joka täyttää tekoälyn määritelmän. Tutkimuksen aikana piti etsiä tietoa monesta eri lähteestä, joista jokainen näki asian omista näkökulmistaan mikä välillä helpotti ja toisin haastoi asian oppimista ja tutkimista. Prosessia helpotti valmiiksi rajatut menetelmät. Tutkimuksen aikana nousi selvästi osa käytetyistä lähteistä ylös, koska niissä oli selitetty asiat selvemmin ja paremmin kuin toisissa

artikkeleissa tai kirjoissa. Myös tekoälyn ohjelmointi prosessi vaatii hyvää ymmärrystä algoritmeista, matematiikasta, tiedon käsittelystä ja analysoinnista.

6. Yhteenveto

Tässä tutkimuksessa kehitettiin tekoälymalli, jotta ohjelmointi prosessista voitaisiin oppia ja nostaa eri huomioita. Työssä tutkimusmenetelmäksi valittiin empiirinen tapaustutkimus. Tutkimusta rajattiin käyttämällä lineaarista regressiota ja matriisien käsittelyä varten otettiin NumPy kirjasto mukaan. Tutkimuksen tavoitteena oli saada toimiva ratkaisu

Tutkimusta jatkettiin tutkimuksen avaamisen jälkeen tutkimalla aiempaa tietoa tekoälystä ja koneoppimista. Jonka jälkeen tutkimuksessa siirryttiin kertomaan koneoppimisen oppimistyyppejä ja esiteltiin laajemmin ohjattua oppimista ja sen yleisimpiä algoritmeja. Tutkimuksessa myös avattiin aiheeseen liittyvää termistöä.

Tutkimuksen käytännön vaihe aloitettiin tutkimalla valittua tietopakettia. Tutkimuksessa tietopaketin analysoinnin jälkeen esitellään teoria tekoälymallin takana ja miten tietopakettia meinataan käsitellä koodissa ja miten saadusta tiedosta ohjelma tekee ennustuksia.

Lopuksi työssä tarkastelleen tuloksia ja nostetaan erilaisia havaintoja ylös sekä mahdollisia tulevaisuuden tutkimus- tai kehitysmahdollisuuksia. Tutkimuksessa rajoitti huomattavasti resurssit, mutta työ saatiin silti onnistuneesti loppuun ja tekoälymalli saatiin saavuttamaan matala älykkyys. Tutkimuskysymyksiin vastattiin tutkimuksen avulla onnistuneesti.

7. Lähteet

[1] A. Panesar, *Machine Learning and AI for Healthcare: Big Data for Improved Health Outcomes*. Berkeley, CA: Apress, 2021. doi: 10.1007/978-1-4842-6537-6.

[2] M. Ergen, ”What is Artificial Intelligence? Technical Considerations and Future Perception”, *Anatol. J. Cardiol.*, 2019, doi: 10.14744/AnatolJCardiol.2019.79091.

[3] M. Nadimpalli, ”Artificial Intelligence Risks and Benefits”, vsk. 6, nro 6, 2007.

[4] S. Raschka, J. Patterson, ja C. Nolet, ”Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence”, *Information*, vsk. 11, nro 4, s. 193, huhti 2020, doi: 10.3390/info11040193.

[5] ”What is NumPy? — NumPy v1.26 Manual”. Viitattu: 3. lokakuuta 2023. [Verkossa]. Saatavissa: https://numpy.org/doc/stable/user/whatisnumpy.html

[6] W. Wang ja K. Siau, ”Artificial Intelligence, Machine Learning, Automation, Robotics, Future of Work and Future of Humanity: A Review and Research Agenda”, *J. Database Manag.*, vsk. 30, nro 1, ss. 61–79, tammi 2019, doi: 10.4018/JDM.2019010104.

[7] O. Lappi, ”TEKOÄLY JA IHMISKO GNITIO”.

[8] H. Vartiainen, M. Tedre, I. Jormanainen, J. Kahila, T. Valtonen, ja T. Toivonen, ”Tekoäly, koneoppiminen ja teknologinen murros:: Kohti datatoimijuutta ja tulevaisuuden design-taitoja”, *Ainedidaktiikka*, vsk. 5, nro 2, joulu 2021, doi: 10.23988/ad.90776.

[9] S. Das, A. Dey, A. Pal, ja N. Roy, ”Applications of Artificial Intelligence in Machine Learning: Review and Prospect”, *Int. J. Comput. Appl.*, vsk. 115, nro 9, ss. 31–41, huhti 2015, doi: 10.5120/20182-2402.

[10] B. Mahesh, ”Machine Learning Algorithms - A Review”, vsk. 9, nro 1, 2018.

[11] ”Machine Learning with R: Expert techniques for predictive modeling - Brett Lantz - Google Books”. Viitattu: 2. lokakuuta 2023. [Verkossa]. Saatavissa: https://books.google.fi/books?hl=en&lr=&id=iNuSDwAAQBAJ&oi=fnd&pg=PP1&dq=Machine+Learning+with+R+by+Brett+Lantz&ots=O8aVse6yR2&sig=ORWOZQ7Vsy0O8xsMSXJ233htD6A&redir_esc=y#v=onepage&q&f=false

[12] Alexander S. GIllis, ”What is an Artifact in Software Development?”, Software Quality. Viitattu: 12. tammikuuta 2024. [Verkossa]. Saatavissa: https://www.techtarget.com/searchsoftwarequality/definition/artifact-software-development

[13] E. Kalapanidas, N. Avouris, M. Craciun, ja D. Neagu, ”Machine Learning algorithms: a study on noise sensitivity”.

[14] K. H. Zou, K. Tuncali, ja S. G. Silverman, ”Correlation and Simple Linear Regression”, *Radiology*, vsk. 227, nro 3, ss. 617–628, kesä 2003, doi: 10.1148/radiol.2273011499.

[15] Per Runeson, Martin Host, Austen Rainer, ja Bjorn Regnell, *Case Study Research in Software Engineering Guidelines and Examples*, 1. p. Hoboken, N.J. : Wiley, 2012. [Verkossa]. Saatavissa: https://lut.primo.exlibrisgroup.com/permalink/358FIN_LUT/1js2888/alma991953812206254

[16] C. Robson ja K. McCartan, *Real world research*, 4. p. John Wiley & Sons Ltd.

[17] ”SciPy”. Viitattu: 11. joulukuuta 2023. [Verkossa]. Saatavissa: https://scipy.org/

[18] ”Real estate price prediction”. Viitattu: 30. lokakuuta 2023. [Verkossa]. Saatavissa: https://www.kaggle.com/datasets/quantbruce/real-estate-price-prediction/data

[19] A. Jung, *Machine Learning: The Basics*. teoksessa Machine Learning: Foundations, Methodologies, and Applications. Singapore: Springer Nature Singapore, 2022. doi: 10.1007/978-981-16-8193-6.

[20] ”numpy.linalg.solve — NumPy v1.26 Manual”. Viitattu: 4. joulukuuta 2023. [Verkossa]. Saatavissa: https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html

8. Liitteet

Liite 1. GitHub lähdekoodi

https://github.com/AleksiLe/SelfmadeAi