OULUN YLIOPISTO
UNIVERSITY of OULU

OULUN YLIOPISTON KAUPPAKORKEAKOULU

Jukka-Pekka Jauhiainen

KONEOPPIMINEN – OHJATTU OPPIMINEN TALOUSTIETEELLISESSÄ TUTKIMUKSESSA

Pro gradu -tutkielma

Taloustiede

Toukokuu 2019

OULUN YLIOPISTO TIIVISTELMÄ OPINNÄYTETYÖSTÄ
Oulun yliopiston kauppakorkeakoulu

Yksikkö Oulun yliopiston kauppakorkeakoulu			
Tekijä Jauhiainen, Jukka-Pekka		Työn valvoja Huikari, S., Tutkijatohtori	
Työn nimi Koneoppiminen – Ohjattu oppiminen taloustieteellisessä tutkimuksessa			
Oppiaine Taloustiede	Työn laji Pro Gradu	Aika Toukokuu 2019	Sivumäärä 81

Tiivistelmä

Tämän tutkielman tarkoituksena on selvittää, voitaisiinko koneoppimista hyödyntää taloustieteellisessä tutkimuksessa. Koneoppiminen on saavuttanut viime vuosina suurta suosiota ja koneoppimista on sovellettu lukuisten erilaisten ongelmien ratkaisemiseksi. Viime vuosina myös taloustieteilijöiden keskuudessa on alkanut herätä mielenkiintoa koneoppimista kohtaan. Koneoppiminen voidaan jakaa kahteen osa-alueeseen: ohjattuun oppimiseen ja ohjaamattomaan oppimiseen. Ohjatun oppimisen ja perinteisten ekonometrian menetelmien tavoite on periaatteessa sama ja ohjatun oppimisen menetelmiä voitaisiinkin mahdollisesti soveltaa tutkimuskäytössä perinteisten ekonometrian menetelmien tapaan.

Tutkielman teoria osuudessa käsitellään koneoppimisen keskeisiä perusperiaatteita sekä useiden koneoppimismenetelmien toimintaperiaatteita. Teoria osuudessa käsitellään lisäksi koneoppimisen ja perinteisten ekonometrian menetelmien eroja sekä mitä ongelmia koneoppimismenetelmiin liittyy.

Koneoppimisen paremmuutta perinteisiin tilastollisiin menetelmiin verrattuna perustellaan niiden paremmalla ennustamiskyvyllä. Koneoppimismenetelmien avulla voidaan sovittaa aineistoihin hyvinkin monimutkaisia malleja ja niiden avulla voidaan aineistosta löytää rakenteita, joita ei perinteisten menetelmien avulla voida löytää. Koneoppimisen soveltamisessa tutkimuskäyttöön liittyy kuitenkin joitain ongelmia. Yksi keskeinen ongelma on se, että koneoppimismenetelmiä ei ole suunniteltu syy-seuraussuhteiden selvittämiseen vaan ne on suunniteltu puhtaasti ennustamistarkoitukseen. Viime aikoina on kuitenkin pyritty kehittämään uusia koneoppimismenetelmiä, joiden avulla voidaan estimoida myös syy-seuraussuhteita.

Tutkielman empiirisessä osuudessa vertaillaan koneoppimismenetelmiä sekä ekonometriassa perinteisesti käytettävää logistista regressiota. Empiirisessä osuudessa pyrittiin ennustamaan portugalilaisen pankin asiakkaiden käytöstä. Empiirisen osuuden lopputulokseksi saatiin se, että koneoppimismallit eivät olleet merkittävästi parempia ennustamaan kuin logistinen regressiokaan.

Tutkielman tuloksena voidaan sanoa, että koneoppimista voidaan hyödyntää taloustieteellisessä tutkimuksessa. Mutta kuten tutkielman empiirisestä osuudesta nähdään, niin koneoppimismenetelmät eivät välttämättä ole joka tilanteessa perinteisiä regressiomalleja parempia. Suurin potentiaalinen käyttökohde koneoppimiselle onkin suurten tietoaineistojen eli niin sanotun big datan analysointi.

Asiasanat
Koneoppiminen, ekonometria, big data

Muita tietoja

SISÄLLYS

KUVIOT

TAULUKOT

1 JOHDANTO

Taloustieteellisessä tutkimuksessa on alettu yhä enemmän painottaa empiirisen tutkimuksen tärkeyttä. Suurin osa 1980-luvun jälkeen julkaistuista vertaisarvioiduista taloustieteellisistä artikkeleista ovat empiirisiä tutkimuksia, kun aikaisemmin julkaistiin enemmän teoreettisia artikkeleita. Vielä 15-20 vuotta sitten näissä tutkimuksissa käytetyt aineistot olivat tutkijoiden itse laatimia tai joidenkin virallisten tahojen keräämiä. Viime vuosina taloustieteellisessä tutkimuksessa on alettu yhä enemmän käyttää uuden tyyppisiä aineistoja, joita ei aiemmin ole ollut saatavilla. Tietokoneet, matkapuhelimet, satelliitit, internet ja erilaiset sensorit keräävät valtavia määriä informaatiota ihmisten käyttäytymisestä. Näitä aineistoja voidaan hyödyntää tutkimuskäytössä ja niitä on jo hyödynnetty useissa tutkimuksissa. Esimerkiksi MIT:n ”billion prices” -projektissa kerätään internetistä satojen tuhansien tuotteiden hinnat päivittäin. Näiden hintojen perusteella voidaan sitten muodostaa inflaatiota mittaava hintaindeksi. Tämän hintaindeksin on osoitettu noudattavan kohtalaisen tarkasti virallisia kuluttajahintaindeksejä. BPP-indeksin etuna on se, että se voidaan laatia lähes reaaliaikaisesti, kun viralliset hintaindeksit julkaistaan usean viikon viiveellä. (Einav & Levin, 2014.)

Toinen esimerkki internet aineiston hyödyntämisestä on työttömyyden ennustaminen Google -hakujen avulla (Einav & Levin, 2014). Tällaisista aineistoista, joissa on suuri määrä muuttujia ja havaintoja, sekä havaintoja kerätään korkealla frekvenssillä, käytetään englanninkielistä nimitystä big data. Big data tarjoaa tutkijoille mahdollisuuksia, mutta sen analysointiin tarvitaan mahdollisesti myös uusia työkaluja. Esimerkiksi kun muuttujia on todella paljon, niin tarvitaan työkaluja, joilla voidaan valita potentiaalisimmat muuttujat. Myös jos muuttujia on enemmän kuin havaintoja, niin perinteisten regressiomallien käyttäminen on mahdotonta. Koneoppimismenetelmät kuitenkin toimivat myös tällaisissa tilanteissa. Koneoppimista hyödynnetään nykyään lukemattomissa sovelluksissa ja kohteissa. Esimerkiksi internetin hakukoneet ja palvelut kuten Netflix ja Spotify hyödyntävät koneoppimista, mutta koneoppimista käytetään jopa lääketieteellisten diagnoosien tekemiseen. Koneoppimista voidaan myös käyttää tieteellisessä tutkimuksessa. Pedro Domingoksen mukaan koneoppiminen on kuin tieteellinen metodi sterodeissa (Domingos, 2015). Tällä hän tarkoittaa sitä, että koneoppimisalgoritmit voivat testata

tuhansia hypoteeseja muutamassa sekunnissa. Viime vuosina taloustieteilijöiden keskuudessa onkin herännyt kiinnostus koneoppimisen hyödyntämiseen taloustieteellisessä tutkimuksessa. Googlen pääekonomisti ja Berkleyn yliopiston emeritusprofessori Hal Varianin mielestä koneoppimisen menetelmiä voitaisiin hyödyntää taloustieteellisessä tutkimuksessa paljon nykyistä laajemmin (Varian, 2014). Varian suositteleekin kaikille taloustieteen opiskelijoille jonkinlaisen koneoppimiskurssin suorittamista. Stanfordin yliopiston professori Susan Athey uskoo, että koneoppimisen menetelmiä tullaan tulevaisuudessa hyödyntämään laajamittaisesti tutkimuskäytössä (Athey, 2018).

Tässä tutkimuksessa perehdytään koneoppimisen menetelmiin ja tarkastellaan koneoppimismenetelmiä ekonometrian ja taloustieteellisen tutkimuksen näkökulmasta. Koneoppimisen menetelmistä keskitytään niin sanottuihin ohjatun oppimisen menetelmiin. Ohjatun oppimisen menetelmissä tavoitteena on yleensä ennustaminen. Tutkimuksen tarkoituksena on selvittää mahdollisuutta käyttää koneoppimista taloustieteellisessä tutkimuksessa sekä tarkastella mitä hyviä ja huonoja puolia koneoppimismenetelmissä on perinteisimpiin ekonometrian menetelmiin verrattuna. Tutkimuksessa myös esitellään useiden ohjatun oppimisen menetelmien toimintaperiaatteita sekä koneoppimismenetelmiin liittyviä keskeisiä konsepteja.

Tutkielman empiirisessä osiossa demonstroidaan koneoppimismenetelmien käyttöä käytännössä sekä vertaillaan koneoppimismenetelmiä ekonometriassa yleisesti käytettyyn logistiseen regressioon. Tutkimuksen aineistona käytetään portugalilaisen pankin puhelinmarkkinointiaineistoa.

Tutkimuksen tuloksena voidaan sanoa, että koneoppimismenetelmiä voidaan hyödyntää taloustieteellisessä tutkimuksessa. Useita tutkimuksia, joissa koneoppimismenetelmiä hyödynnetään, on jo julkaistu. Koneoppimismenetelmien suurin heikkous on siinä, että niitä ei ole suunniteltu syy-seuraussuhteiden selvittämiseen vaan puhtaasti ennustamiseen. Jotta koneoppimismenetelmiä voidaan käyttää laajamittaisesti taloustieteellisessä tutkimuksessa, niitä täytyy kehittää niin että niiden avulla syy-seuraussuhteita voidaan selvittää.

Tutkimus etenee siten että luvussa 2 käydään läpi koneoppimisen keskeisiä perusperiaatteita ja useita ohjatun oppimisen menetelmien toimintaperiaatteita. Luvussa 3 käsitellään koneoppimista ekonometrian ja taloustieteellisen tutkimuksen näkökulmasta. Luvussa 4 kuvaillaan empiirisessä osiossa käytetty aineisto ja tutkimusmenetelmät. Luvussa 5 kuvataan empiirisen osuuden vaiheet ja tulokset. Luvussa 6 esitellään tutkimuksen johtopäätökset.

2 KONEOPPIMINEN

2.1 Mitä koneoppiminen on?

Koneoppimisen määritelmä ei ole täysin yksiselitteinen. Usein koneoppiminen määritellään esimerkiksi siten, että sen tarkoituksena on saada tietokoneet oppimaan ilman että, niitä tarvitsee eksplisiittisesti ohjelmoida. Voidaan myös sanoa, että koneoppimisessa tarkoituksena on luoda algoritmeja, jotka oppivat empiirisesti aineistoon tai dataan perustuen. Koneoppimisen tarkasta määritelmästä ei kuitenkaan ole laajaa yksimielisyyttä. Käytännössä koneoppimisen menetelmät ovat tilastollisia menetelmiä ja koneoppimisen sijaan näistä menetelmistä joskus käytetäänkin termiä tilastollinen oppiminen. Koneoppiminen voidaan jakaa pääpiirteissään kahteen osaan: ohjattuun oppimiseen ja ohjaamattomaan oppimiseen.

Ohjatussa oppimisessa tarkoituksena on luoda malli, joka ennustaa vastemuuttujan arvoja selittävien muuttujien avulla mahdollisimman hyvin. Koneoppimissanastossa selittäviä muuttujia kutsutaan usein myös ominaisuuksiksi tai piirteiksi (features) ja vastemuuttujaa kutsutaan kohdemuuttujaksi (target). Tarkoituksena ohjatussa oppimisessa on siis löytää funktio, joka kuvaa hyvin syötemuuttujien ja vastemuuttujien välistä suhdetta. Perinteisesti ekonometriassa tällaisen funktion estimoimiseen on käytetty lineaarisia regressiomalleja. Ohjatun oppimisen ja perinteisten ekonometrian työkalujen tarkoitus on siis pitkälti sama, mutta niiden välillä on hyvin oleellisia eroja. Tarkemmin näitä eroja käsitellään luvussa kolme. (Boelaert & Ollion, 2018; Athey, 2018; Mullainathan & Spiess, 2017.)

Ohjaamatonta oppimista käytetään, kun käytettävässä aineistossa ei ole syöte-vaste – pareja (unlabeled data). Toisin sanoen sitä käytetään, kun aineistossa ei ole kohdemuuttujaa, jonka arvot ovat tiedossa. Ohjaamattoman oppimisen avulla pyritään löytämään yhteyksiä muuttujien välillä, niin että aineiston havainnot voidaan jakaa jonkinlaisiin ryhmiin. Ohjaamattoman oppimisen menetelmiin kuuluvat esimerkiksi klusterointi ja pääkomponenttianalyysi (Principal component analysis). Tässä tutkielmassa ohjaamatonta oppimista ei käsitellä tarkemmin vaan keskitytään ohjatun oppimisen menetelmiin. (Boelaert & Ollion, 2018; Athey, 2018; Mullainathan & Spiess, 2017.)

2.1.1 Opetus-testaus -jako

Koneoppimisessa on siis tarkoitus luoda malli, joka ennustaa mahdollisimman hyvin. Oleellista kuitenkin on, että malli ennustaa hyvin uusia havaintoja, eli sellaisia havaintoja, joita ei ole käytetty mallin sovittamiseen/opettamiseen. Koneoppimisessa mallin hyvyyttä arvioidaan pelkästään sen ennustamistarkkuudella. Mallin ennustamistarkkuudesta saadaan arvio käyttämällä mallia uusien havaintojen ennustamiseen. Jotta mallin ennustetarkkuutta voidaan testata, koneoppimisprojekteissa aineisto jaetaan kahteen osaan: opetus- ja testausosaan. Opetusosaa käytetään mallin opettamiseen ja testausosaa mallin testaamiseen. Näin toimitaan, koska jos mallin ennustetarkkuutta testattaisiin aineistolla, jota on käytetty mallin opettamiseen, mallin ennustetarkkuudesta saataisiin turhan optimistinen arvio. Useat koneoppimismenetelmät ovat myös erittäin joustavia ja niiden avulla voidaan luoda erittäin monimutkaisia, epälineaarisia malleja. Tästä johtuen koneoppimismenetelmillä voidaan helposti sovittaa malleja, jotka ennustavat opettamiseen käytettyjä havaintoja hyvin, mutta joiden ennustetarkkuus on huono uusia havaintoja ennustettaessa. Tätä ongelmaa kutsutaan ylisovittamiseksi ja sitä käsitellään tarkemmin luvussa 2.2. (Boelaert & Ollion, 2018; Athey, 2018; Mullainathan & Spiess, 2017.)

2.1.2 Hyperparametrit

Aineisto voidaan jakaa myös kolmeen osaan: opetus-, testaus- ja validointiosaan. Validointiosaa voidaan käyttää muun muassa sen varmistamiseksi, että malli ei ylisovita sekä optimaalisten hyperparametrien etsimiseen. Hyperparametrit ovat koneoppimisalgoritmien parametreja, joilla voidaan vaikuttaa mallin kompleksisuuteen/monimutkaisuuteen. Hyperparametrit ovat siis nimenomaan algoritmien parametreja, eivätkä mallien parametreja. Eri algoritmeilla on eri hyperparametreja ja useissa algoritmeissa on useita hyperparametreja, joita voidaan muuttaa. Esimerkiksi päätöspuiden hyperparametreilla voidaan rajoittaa päätöspuun maksimisyvyyttä tai lehtisolmuissa olevien havaintojen määrää. Yleensä vain muutaman hyperparametrin muuttaminen on tarpeellista tarpeeksi hyvän mallin

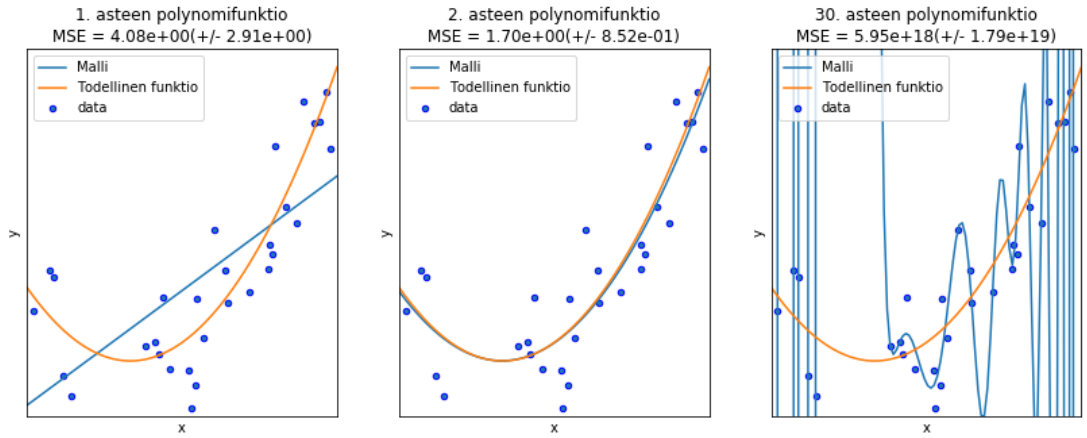
luomiseen. Optimaalisten hyperparametrien etsiminen on oleellinen osa koneoppimisprojektia. Optimaaliset hyperparametrit etsitään käytännössä kokeilemalla useita arvoja valituille hyperparametreille, ja laskemalla eri parametrien arvoilla luotujen mallien ennustevirheet käyttäen validointiaineistoa tai ristiinvalidointia. Optimaaliset hyperparametrien arvot ovat ne, jotka johtavat pienimpään ennustevirheeseen. Hyperparametrien arvoja etsiessä on tärkeää käyttää juuri validointiaineistoa (tai ristiinvalidointia) ennustevirheiden laskemiseen, eikä testausaineistoa. Testausaineistoa käytetään pelkästään testaamiseen, ja vasta kun lopullinen malli on luotu ja hyperparametrien arvot valittu. Näin toimitaan, jotta mallin ennustevirheestä saadaan mahdollisimman luotettava arvio. (Boelaert & Ollion, 2018; Athey, 2018; Mullainathan & Spiess, 2017.)

2.2 Ylisovittaminen

Ylisovittaminen (overfitting) tarkoittaa sitä, että koneoppimisalgoritmin luoma malli ennustaa opetusaineistoon kuuluvat havainnot liian hyvin. Mallit voidaan sovittaa jopa siten, että malli ennustaa opetusaineiston havainnot täydellisesti. Tämä ei ole kuitenkaan hyvä ratkaisu, sillä aineistossa on aina hieman satunnaisvaihtelua. Jos mallista luodaan liian monimutkainen, niin se alkaa ennustamaan myös tätä aineiston havaintojen satunnaisvaihtelua. Tämä taas johtaa siihen, että malli ei yleistä hyvin, ja kun mallia käytetään ennustamiseen uuden aineiston avulla, niin mallin ennusteet ovat huonoja. (James, Witten, Hastie & Tibshirani, 2013, s.30-36.)

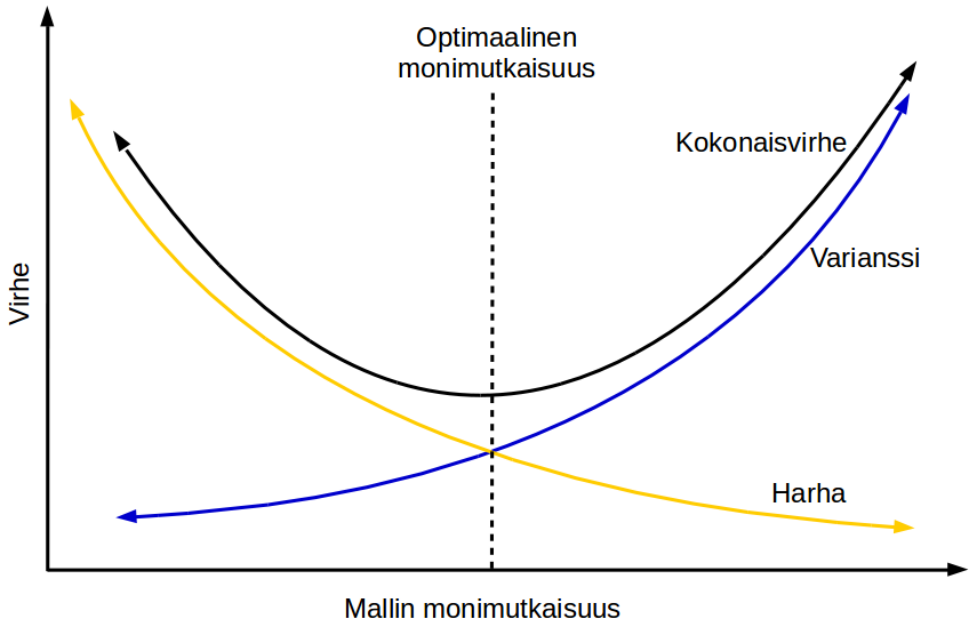
Ylisovittamista on havainnollistettu kuviossa 1. Kuviossa olevat siniset pisteet ovat havaintoja, jotka on luotu lisäämällä toisen asteen polynomifunktioon hieman satunnaisvaihtelua. Kuvion keltainen viiva kuvaa tätä toisen asteen polynomifunktiota, jota on käytetty aineiston luomiseen. Kuvion vasemmassa paneelissa aineistoon on sovitettu lineaarinen, ensimmäisen asteen polynomifunktio. Selvästikin tämä malli alisovittaa, eikä sovi aineistoon hyvin. Kuvion keskimmäisessä paneelissa aineistoon on sovitettu toisen asteen polynomifunktio. Luonnollisesti tämä malli sopii aineistoon hyvin. Kuvion oikean puolimmaisessa paneelissa aineistoon on sovitettu 30. asteen polynomifunktio. Tämä malli ennustaa aineiston pisteet paremmin kuin aikaisemmat mallit, mutta se selvästikin ylisovittaa. Se ennustaa myös aineistossa olevaa satunnaisvaihtelua. Ylisovittaminen siis johtuu

siitä, että luotu malli on liian monimutkainen. Malli ennustaa hyvin opetusaineiston havaintoja, mutta jos sitä testattaisiin uudella aineistolla, niin sen ennustevirhe olisi todennäköisesti suuri. Voidaan myös sanoa, että kun malli ylisovittaa, se ei yleistä hyvin. Oleellinen osa jokaista koneoppimisprojektia onkin sen varmistaminen, että valittu malli ei ylisovita. Ylisovittamisen estämiseksi koneoppimisalgoritmeissa mallien monimutkaisuutta voidaan rajoittaa eri hyperparametreilla.



Kuvio 1. Esimerkki ylisovittamisesta (Kirjoittajan omat laskelmat).

Ylisovittamisen ongelma liittyy vahvasti harhaan ja varianssiin. Ennustavan mallin ennustevirhe voidaan jakaa kahteen komponenttiin: harhaan ja varianssiin. Harhalla tarkoitetaan virhettä, joka syntyy siitä, että malli on liian yksinkertainen. Jos esimerkiksi todellinen funktio on epälineaarinen, on lineaarisen mallin harha suuri. Varianssilla taas tarkoitetaan virhettä, joka syntyy siitä, että malli reagoi liian herkästi opetusaineistossa olevaan vaihteluun. Jos mallin varianssi on suuri, se todennäköisesti ylisovittaa. Harhasta ja varianssista aiheutuvaa virhettä on kuvattu kuviossa 2. Kun mallin monimutkaisuus kasvaa, niin harha vähenee, mutta varianssi kasvaa. Mallin monimutkaisuutta säädettäessä pyritään valitsemaan malli, jonka kokonaisvirhe on mahdollisimman pieni. Koska aineistossa on aina hieman kohinaa, kokonaisvirhettä ei kyetä koskaan minimoimaan täysin pois. (Domingos, 2012.)



Kuvio 2. Harha, varianssi ja mallin monimutkaisuus (mukaillen Hall, 2018).

2.3 Ristiinvalidointi

Ristiinvalidointi (cross-validation) on menetelmä, jota voidaan käyttää mallin ennustevirheen arviointiin tai mallin parametrien valintaan. Varianin (2014) mukaan ristiinvalidointi on yksi koneoppimisen menetelmistä, jota pitäisi käyttää enemmän taloustieteessä. Ristiinvalidointi tehdään tyypillisesti siten, että aineisto jaetaan useaan yhtä suureen osajoukkoon, minkä jälkeen malli sovitetaan ja testataan yhtä monta kertaa kuin aineistossa on osajoukkoja. Jos esimerkiksi suoritettaisiin 10-kertainen ristiinvalidointi (10-fold cross validation), aineisto jaettaisiin ensin 10 osaan. Sitten malli sovitettaisiin käyttäen kaikkia paitsi ensimmäistä osajoukkoa opetukseen, ja ensimmäistä osajoukkoa käytettäisiin ennustevirheen laskemiseen. Seuraavassa vaiheessa sama toistettaisiin siten, että toisena olevaa osajoukkoa käytettäisiin ennustevirheen laskemiseen. Tätä jatketaan, kunnes kaikkia kymmentä osajoukkoa on käytetty ennustevirheen laskemiseen. Näin koko aineistoa käytetään sekä opettamiseen että ennustevirheen laskemiseen. Lopullinen ennustevirheen arvio saadaan näiden kymmenen ennustevirheen keskiarvosta. Ristiinvalidoinnin avulla laskettu ennustevirhe on luotettava, koska ennustevirhettä laskettaessa käytetään aina sellaista aineiston osaa, jota ei ole käytetty mallin sovittamisessa ja ristiinvalidoinnin avulla laskettu

ennustevirhe on usein hyvin lähellä todellista ennustevirhettä. Ristiinvalidointi on myös hyvä työkalu mahdollisen ylisovittamisen havaitsemiseen ja sitä voidaan käyttää silloinkin, kun aineisto on niin pieni, että sitä ei voida jakaa erillisiin opetus- ja validointiosiin. (Hastie, Tibshirani & Friedman, 2009, s.241-247.)

2.4 Päätöspuut

Päätöspuut (Classification and regression trees) ovat monipuolisia koneoppimisalgoritmeja, joita voidaan käyttää sekä regressio- että luokittelutehtäviin. Päätöspuut ovat erittäin tehokkaita ja niiden avulla voidaan sovittaa malleja hyvin monimutkaisiin aineistoihin. Päätöspuiden toimintaperiaate on kohtalaisen yksinkertainen. Päätöspuu jakaa aineiston useita kertoja kahteen osaan. Jakamisesta syntyviä aineiston osia jaetaan yhä uudelleen, kunnes aineiston havaintojen puhtautta ei voida parantaa, tai saavutetaan jokin ennalta määritetty pysäytyskriteeri. Aineiston jakaminen tapahtuu selittävien muuttujien avulla. Päätöspuu muodostuu solmuista, oksista ja lehdistä. Ensimmäistä solmua, jossa ovat aineiston kaikki havainnot kutsutaan juurisolmuksi. Kun aineisto jaetaan, niin syntyy oksia. Oksat yhdistävät juurisolmun, sisäsolmut ja lehtisolmut, joihin aineiston osat jaetaan. Jokaisessa solmussa aineisto jaetaan jonkin muuttujan perusteella, kunnes aineiston puhtautta ei voida enää parantaa. Se minkä muuttujan perusteella aineisto jaetaan kussakin solmussa, määrittää niin sanottu jakokriteeri. Jakokriteereitä on useita, mutta yleisimmät jakokriteerit luokitteluasteikon muuttujille ovat gini-epäpuhtaus ja Entropia. Regressiopuiden jakokriteerinä käytetään yleensä keskimääräistä ennustevirheen neliötä. (James, Witten, Hastie & Tibshirani, 2013, s.303-315.)

Gini-epäpuhtaus solmulle i lasketaan seuraavalla kaavalla:

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2 \tag{1}$$

Edellä G_i on siis gini-epäpuhtaus, jota päätöspuu-algoritmi pyrkii minimoimaan. Kaavassa k on diskreetin vastemuuttujan luokka ja $p_{i,k}$ on luokkaan k kuuluvien

havaintojen osuus i:nnessä solmussa. Kun solmun kaikki havainnot kuuluvat samaan luokkaan, niin gini-epäpuhtaus saa arvon 0, ja koska puhtautta ei voida lisätä, niin kyseisestä solmusta ei kasvateta lisää oksia ja solmusta tulee lehtisolmu.

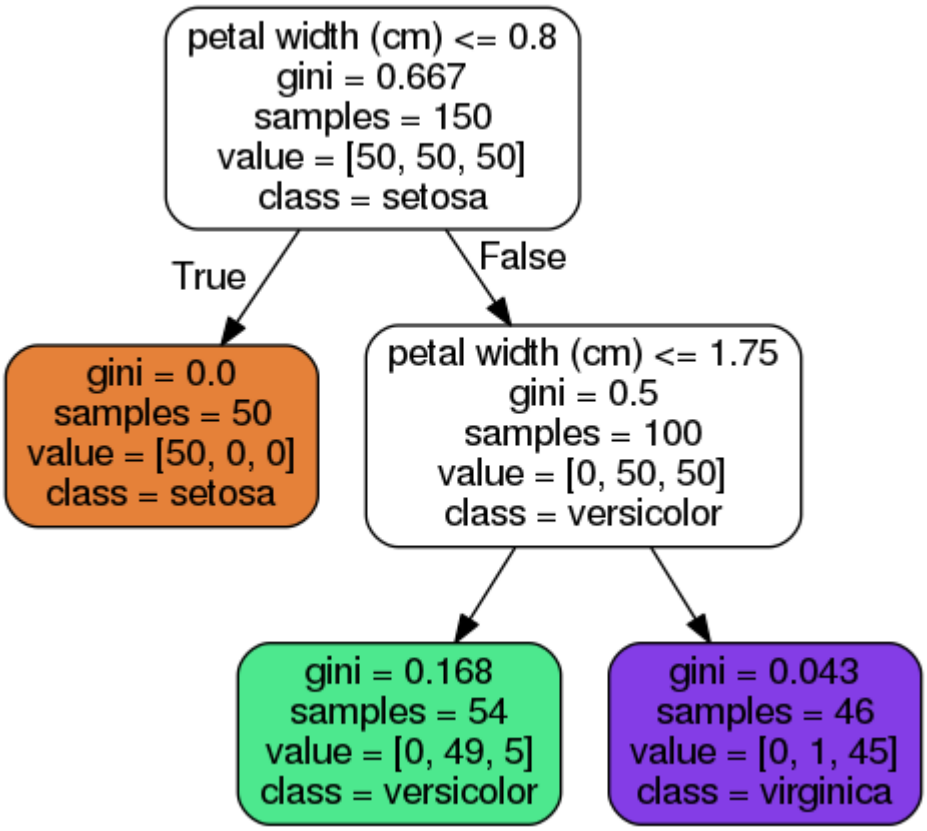
Entropia poikkeaa hieman gini-epäpuhtaudesta, mutta yleensä molemmat johtavat suurin piirtein samanlaiseen lopputulokseen. Entropia-epäpuhtaus lasketaan seuraavalla kaavalla:

$$H_i = -\sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log(p_{i,k}) \quad (2)$$

Edellä H_i on entropia, ja kuten kaavassa 1, $p_{i,k}$ on luokkaan k kuuluvien havaintojen osuus i:nnessä solmussa. Algoritmi valitsee muuttujan x ja sen arvon siten, että syntyvien solmujen epäpuhtauden painotettu keskiarvo on mahdollisimman pieni. Jos muuttujan arvoa kuvataan muuttujalla t, niin algoritmi pyrkii löytämään parin (x, t_x), joka minimoi syntyvien aineiston osien epäpuhtauden.

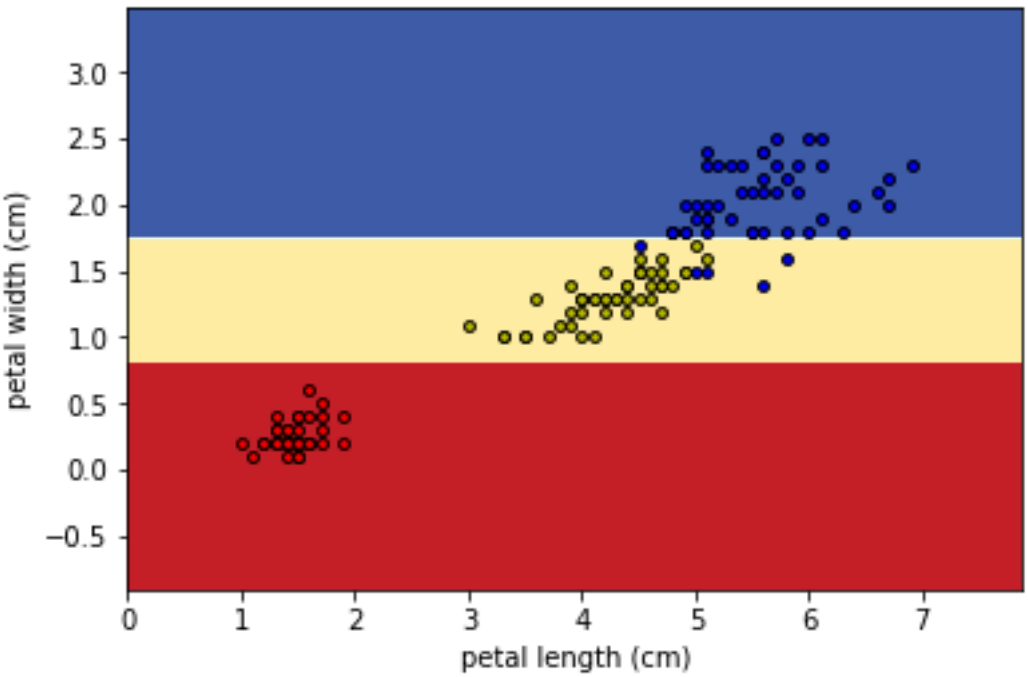
Yksi päätöspuiden hyvistä puolista on se, että niiden toimintaa voidaan kuvata graafisesti. Kuviossa 3 on päätöspuu, joka on luotu Pythonin scikit-learn paketin iris-aineistoa hyväksi käyttäen. Aineistossa on 150 havaintoa, 2 selittävää muuttujaa (Petal width ja Petal length) sekä vastemuuttuja (class). Aineiston havainnot koostuvat kasveista ja niiden ominaisuuksista. Päätöspuun tarkoituksena on ennustaa mihin lajiin (setosa, versicolor, virginica) kukin havainto kuuluu. Päätöspuuta luetaan ylhäältä alaspäin siten, että mikäli valittu jakokriteeri pitää paikkansa kyseisen havainnon osalta, liikutaan vasenta oksaa pitkin alaspäin. Mikäli jakokriteeri ei pidä paikkaansa liikutaan oikeaa oksaa pitkin. Kuvion 3 päätöspuu on luotu gini-epäpuhtautta käyttäen. Kuviossa esimerkiksi vihreällä väritetyn lehtisolmun gini-epäpuhtaus on 0.168. Tämä voidaan laskea kaavan 1 avulla:

$$1 - \left(\frac{0}{54}\right)^2 - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 \approx 0.168$$



Kuvio 3. Päätöspuu iris-aineistoa käyttäen (Kirjoittajan omat laskelmat).

Kuviossa 4 on kuvattu kuvion 3 päätöspuun päätösrajoja. Päätöspuu jakaa aineiston ensin kohdasta $petal\ width \leq 0.8$, ja sen jälkeen kohdasta $petal\ width \leq 1.75$. Tässä tapauksessa toista muuttujaa (petal length) ei käytetty aineiston jakamiseen lainkaan. Kuviossa 4 punaiset pisteet ovat aineiston setosa-luokkaan kuuluvia havaintoja. Kuviossa 3 oranssilla värjätyn lehtisolmun gini-epäpuhtaus on 0, ja kuviosta 4 nähdään myös, että aineiston puhtautta ei voida tämän luokan osalta parantaa, koska punaisella alueella on vain setosa-luokan havaintoja.

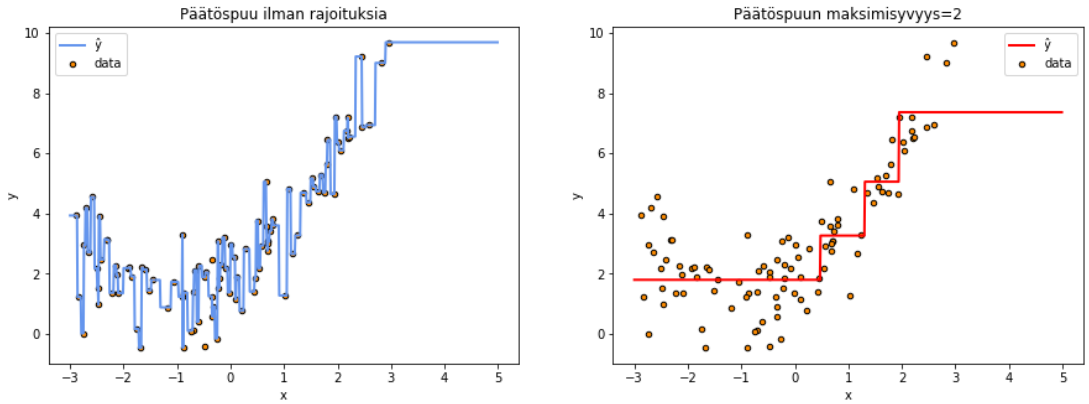


Kuvio 4. Päätöspuun päätösrajat (kirjoittajan omat laskelmat).

Edellä on kuvattu päätöspuiden toimintaa, kun vaste on luokitteluasteikollinen muuttuja. Päätöspuita voidaan käyttää myös regressiotehtäviin, kun vaste on jatkuva muuttuja. Regressiopuut toimivat pitkälti samalla logiikalla kuin luokitteluaineistoon käytetyt päätöspuut, ainoastaan aineiston jakokriteeri muuttuu. Jakokriteerinä käytetään yleensä pienimmän virheneliösumman menetelmää. Käytännössä regressiopuut jakavat aineiston siis valitsemalla muuttujan, ja muuttujan arvon, jonka perusteella jaetun aineiston virheneliösumma on pienin. Solmun ennustama vaste on siihen kuuluvien havaintojen vasteiden keskiarvo.

Päätöspuut ovat siis hyvin monipuolisia algoritmeja, joiden avulla aineistosta voidaan löytää monimutkaisia suhteita ja luoda epälineaarisia malleja, joita olisi vaikea luoda esimerkiksi lineaaristen regressiomallien avulla. Päätöspuiden tulkinta on myös helppoa. Päätöspuiden ongelmia ovat suuri varianssi ja alttius ylisovittamiselle. Varianssilla tarkoitetaan sitä, että päätöspuun rakenne voi muuttua huomattavasti pienestäkin muutoksesta opetusaineistossa. Tämä johtuu osittain siitä, että päätöspuun solmujen muutokset vaikuttavat mahdollisiin kyseisen solmun alapuolella oleviin solmuihin. Näin pienetkin muutokset päätöspuun rakenteessa leviävät alaspäin ja voivat muuttaa lehtisolmujen ennusteita huomattavasti.

Kuviossa 5 on havainnollistettu päätöspuiden taipumusta ylisovittaa. Kun päätöspuun annetaan kasvaa rajatta, se sovittaa lähes täydellisesti opetusaineistoon. Kuviossa 5 oikeanpuoleisessa paneelissa päätöspuun maksimisyvyydeksi on määritetty 2, mikä johtaa huomattavasti yksinkertaisempaan malliin.



Kuvio 5. Päätöspuu ja ylisovittaminen (kirjoittajan omat laskelmat).

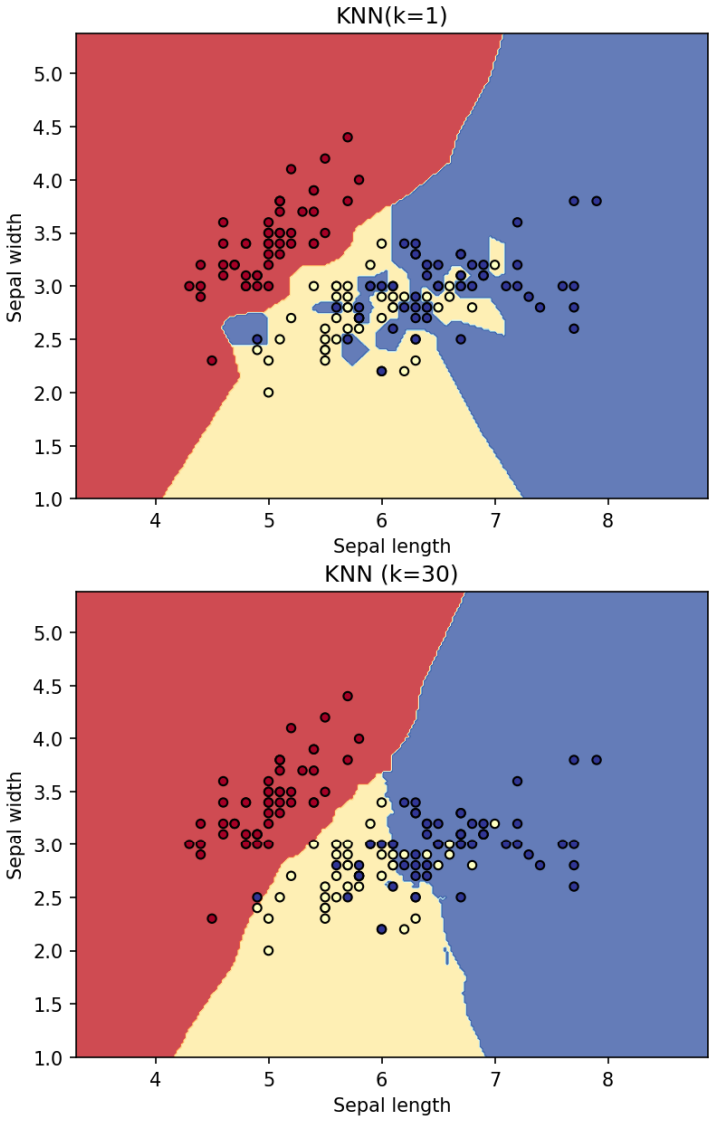
2.5 K:n lähimmän naapurin luokittelija

K:n lähimmän naapurin menetelmä on erittäin yksinkertainen, mutta tehokas luokittelumenetelmä. K:n lähimmän naapurin luokittelija ennustaa uuden havainnon/instanssin luokan siten, että se laskee kaikkien opetusaineiston näytteiden etäisyyden uuteen havaintoon, jonka jälkeen valitaan k kappaletta näytteitä, joiden etäisyys on pienin uuteen instanssiin. Uuden instanssin luokka määräytyy sen mukaan, mikä on näiden k:n lähimmän naapurin yleisin luokka. Etäisyys voidaan mitata useilla eri tavoilla. Yleisin käytetty mitta on euklidinen etäisyys. Euklidinen etäisyys on kahden pisteen välinen etäisyys. Kaksiulotteisessa avaruudessa kahden pisteen euklidinen etäisyys voidaan laskea Pythagoraan lauseella. (Wu ym., 2008.)

K:n lähimmän naapurin luokittelija on niin sanottu laiska oppija. Laiskat oppijat eroavat ahkerista oppijoista, kuten päätöspuista ja tukivektorikoneista siten, että niiden malleja ei rakenneta eksplisiittisesti opetusvaiheessa. Laiskojen oppijoiden opetusvaihe koostuu käytännössä aineiston tallentamisesta muistiin ja varsinainen laskenta tapahtuu vasta uusia instansseja luokiteltaessa. (Wu ym., 2008.)

K:n lähimmän naapurin luokittelijan luokittelukykyyn vaikuttaa usea tekijä. Keskeinen valinta luokittelijaa muodostettaessa on sopivan k:n arvon valinta. Liian pieni k:n arvo voi johtaa ylisovittamiseen ja liian suuri k:n arvo voi johtaa alisovittamiseen. K:n arvoksi kannattaa myös valita sellainen luku, että uusien instanssien luokittelussa ei päädytä tasatulokseen. Esimerkiksi binäärisen kohdemuuttujan tapauksessa k:n arvoksi kannattaa valita pariton luku. Koska k:n lähimmän naapurin menetelmä käyttää etäisyyksiä, yleensä muuttujat kannattaa normalisoida. Normalisointi on tarpeellista varsinkin, jos muuttujien arvoalueiden vaihtelut ovat suuria. Jos jonkin muuttujan arvoalue on huomattavasti laajempi kuin muiden muuttujien arvoalue, tällainen muuttuja voi dominoida luokitusta. (Wu ym., 2008.)

Kuviossa 6 on havainnollistettu k:n lähimmän naapurin luokittelijan päätösrajoja k:n arvoilla 1 ja 30. Kun k:n arvo on 1, niin päätösrajat määräytyvät vain yhden lähimmän naapurin mukaan ja päätösrajoista tulee kiemuraisia. Kun k:n arvo on suurempi, niin päätösrajoista tulee lineaarisempia.



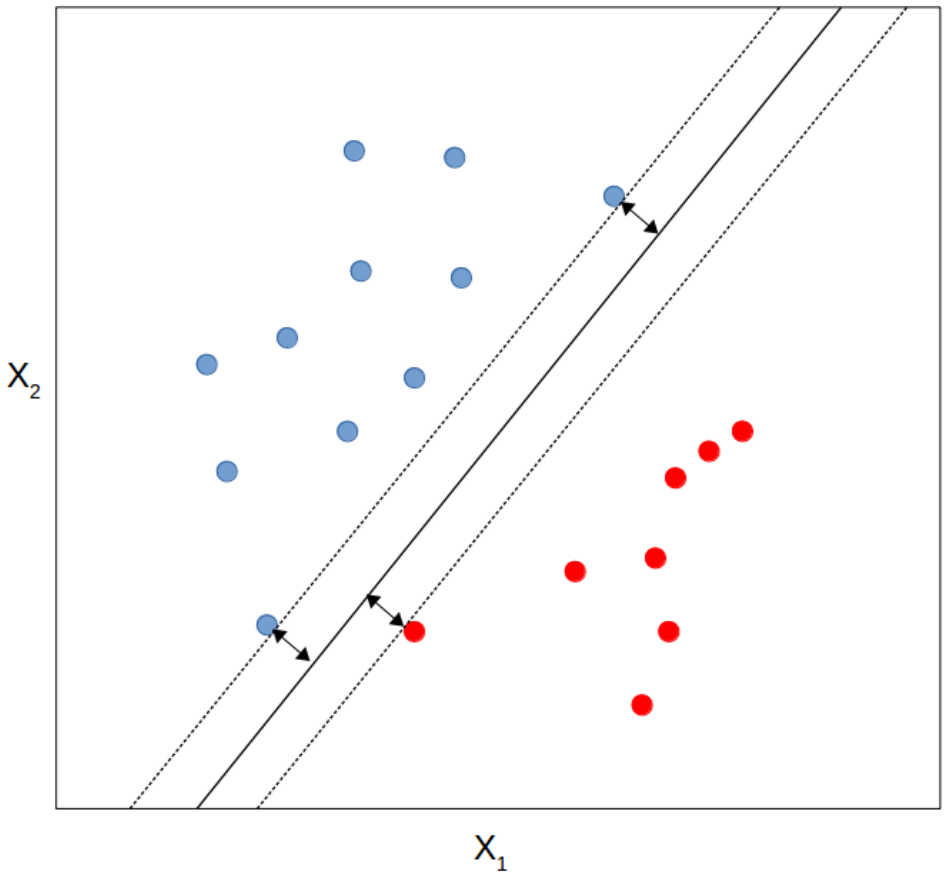
Kuvio 6. K:n lähimmän naapurin luokittelija (kirjoittajan omat laskelmat).

2.6 Tukivektorikone

Tukivektorikone on yksi suosituimmista luokittelumalleista. Tukivektorikoneita pidetäänkin yksinä parhaimmista ”out of the box”-luokittelijoista. Toisin sanoen tukivektorikoneet toimivat yleensä kohtalaisen hyvin ilman, että niiden hyperparametreja tarvitsee säätää. Tukivektorikone perustuu hyvin yksinkertaiseen ja intuitiiviseen luokittelijaan, jota kutsutaan maksimaalisen marginaalin luokittelijaksi (Maximal margin classifier). Maksimaalisen marginaalin luokittelijan periaatteena on asettaa aineiston näytteiden väliin päätöstaso, joka erottelee näytteet niiden vastemuuttujan arvon mukaan. Esimerkiksi jos vastemuuttuja on binäärinen

muuttuja, niin näytteet eroteltaisiin siten, että päätöstason toisella puolella olevien näytteiden vastemuuttujan arvo on 1 ja tason toisella puolella olevien näytteiden vasteen arvo on 0. Jotta tällainen päätöstaso voidaan asettaa, niin aineiston täytyy olla lineaarisesti eroteltavissa, eli eri luokkien yksikäsitteinen erottaminen tasolla täytyy olla mahdollista. Jos aineisto on lineaarisesti eroteltavissa, niin silloin on olemassa loputon määrä tasoja, joilla aineisto voidaan erotella. Maksimaalisen marginaalin luokittelijan periaatteena on sovittaa näytteiden välille päätöstaso siten, että päätöstason ja näytteiden väliin jäävien marginaalitasojen välimatka on mahdollisimman suuri. (James, Witten, Hastie & Tibshirani, 2013, s. 337-354.)

Maksimaalisen marginaalin luokittelijan toimintaa on havainnollistettu kuviossa 7. Kuviossa marginaalitasot on piirretty katkoviivalla ja päätöstaso on piirretty yhtenäisellä viivalla. Marginaali on katkoviivalla merkittyjen tasojen ja yhtenäisellä viivalla merkityn päätöstason välinen etäisyys. Marginaalitasot kulkevat lähimpänä päätöstasoa kulkevien näytteiden kautta. Näitä näytteitä kutsutaan tukivektoreiksi. Nimi tukivektori tulee siitä, että ne tukevat maksimaalisen marginaalin päätöstasoa. Jos näitä näytteitä siirrettäisiin, niin myös päätöstaso liikkuisi. Merkille pantavaa on se, että maksimaalisen marginaalin tason sijainti riippuu täysin tukivektoreista, mutta ei lainkaan muista näytteistä. Mikäli kauempana olevat näytteet liikkuisivat, päätöstaso pysyisi entisellä paikallaan. Kun maksimaalisen marginaalin luokittelijaa käytetään luokittelemaan uusia havaintoja, ne luokitellaan sen mukaan kummalla puolella päätöstasoa ne ovat. (James, Witten, Hastie & Tibshirani, 2013, s. 337-354.)



Kuvio 7. Tukivektorikone (mukaillen James, Witten, Hastie & Tibshirani, 2013, s. 342)

Mikäli aineisto ei ole lineaarisesti eroteltavissa, niin voidaan käyttää niin sanottua joustavan marginaalin luokittelijaa, missä joidenkin näytteiden sallitaan olla marginaalin väärällä puolella tai jopa päätöstason väärällä puolella. Päätöstaso sovitetaan niin, että se luokittelee suurimman osan opetukseen käytetyistä näytteistä oikein.

On myös muita tapoja käyttää kyseistä luokittelijaa, vaikka aineisto ei ole lineaarisesti separoituva. Syötemuuttujiin voidaan lisätä syötemuuttujien muunnoksia. Muuttujat voidaan esimerkiksi korottaa toiseen tai kolmanteen potenssiin. Samankaltaista toimenpidettä käytetään usein lineaaristen regressiomallien suhteen, kun syötemuuttujien ja vasteen välinen suhde on epälineaarista. Syötemuuttujien joukkoa voidaan kasvattaa hyvin monilla tavoilla, ja vaarana onkin, että muuttujien määrä kasvaa niin suureksi, että päätöstason määrittämiseksi vaadittavat laskutoimitukset on mahdoton suorittaa. Tämän

ongelman ratkaisemiseksi käytetään usein niin sanottuja kernel-funktioita. Kernel-funktioiden avulla syötemuuttujien joukkoa voidaan laajentaa siten että, lineaarisesti erottamattomat näytejoukot voidaan erottaa. Kernel-funktioiden etuna on myös se, että niiden käyttämiseen ei vaadita yhtä paljon laskutoimituksia kuin jos muuttujajoukkoa kasvatettaisiin lisäämällä alkuperäisten syötemuuttujien funktioita. (James, Witten, Hastie & Tibshirani, 2013, s. 337-354.)

Kernel-funktioiden ja tukivektorikoneiden laskutoimitukset ja matemaattiset esitysmuodot ovat kohtalaisen monimutkaisia, eikä niitä käsitellä tässä tarkemmin. Edellä on kuvattu tukivektorikoneen toiminnan periaate. Tarkoituksena on sovittaa eri luokkiin kuuluvien havaintojen väliin päätöstaso, niin että marginaalitasojen ja päätöstason välinen etäisyys on mahdollisimman suuri. Tukivektorikone ennustaa uusien havaintojen Y-muuttujan luokan sen mukaan millä puolella päätöstasoa ne ovat. Tukivektorikone on, useiden muiden algoritmien tapaan, altis ylisovittamiselle. Ylisovittamisen ongelma ratkaistaan käyttämällä joustavaa marginaalia, missä joidenkin näytteiden sallitaan olla marginaalien väärällä puolella. Marginaalien joustavuutta säädetään hyperparametrilla C. Mitä pienempi C:n arvo on sitä, joustavampi on marginaali. Liian suuret C:n arvot voivat johtaa ylisovittamiseen ja liian pienet C:n arvot voivat johtaa alisovittamiseen. (James, Witten, Hastie & Tibshirani, 2013, s. 337-354.)

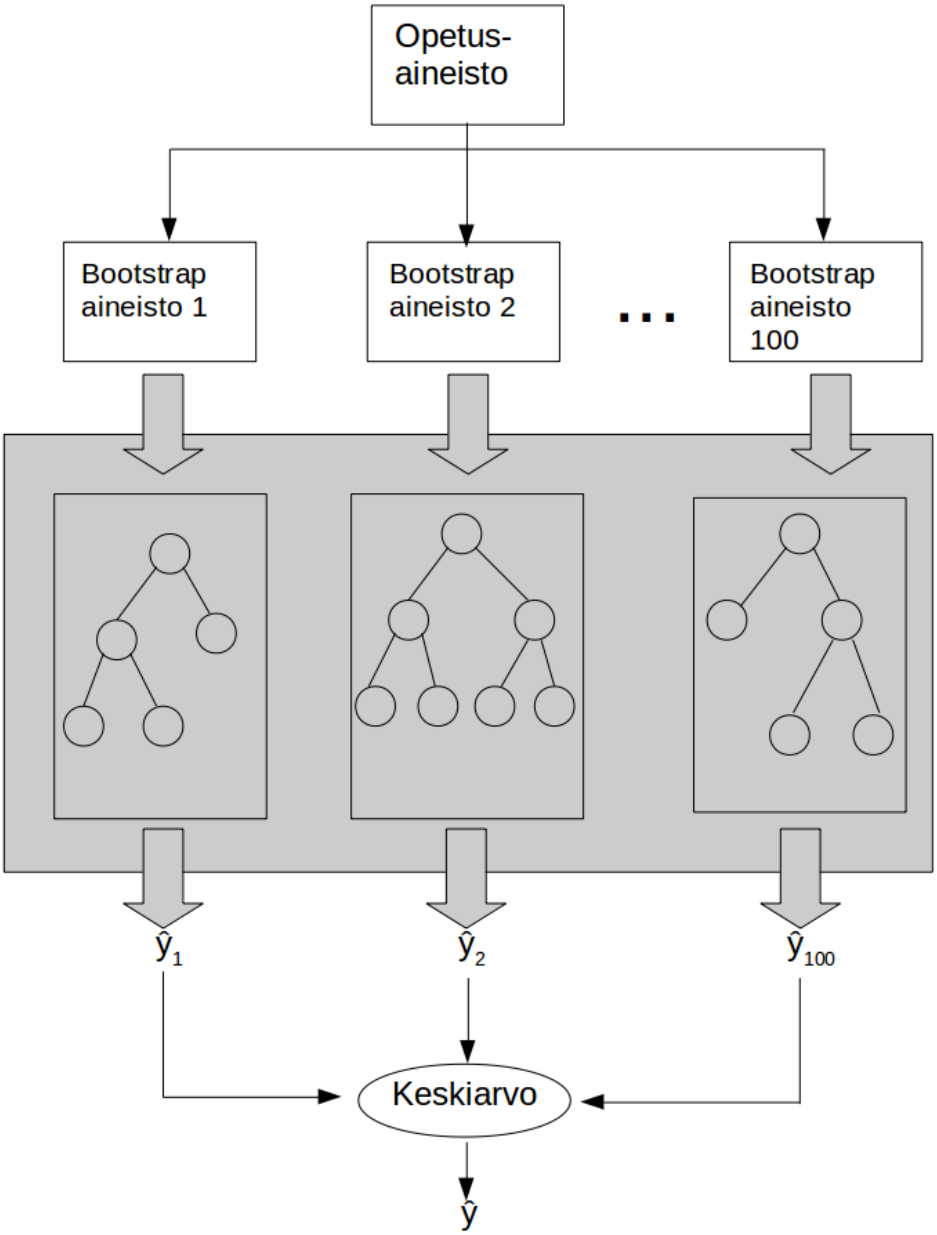
2.7 Bagging-menetelmä

Bagging tulee englanninkielen sanoista bootstrap aggregating. Bootstrap-menetelmällä tarkoitetaan satunnaisotoksen ottamista takaisinpanolla. Tämä tarkoittaa sitä, että opetusaineistosta otetaan satunnaisesti näytteitä, ja samoja näytteitä voidaan ottaa useaan kertaan. Tyypillisesti näytteitä otetaan yhtä monta kuin alkuperäisessä aineistossakin on havaintoja. Näihin opetusaineistosta otettuihin otoksiin sitten sovitetaan jokin koneoppimismalli, esimerkiksi päätöspuu. Mallin ennuste saadaan yksinkertaisesti näiden useiden eri mallien ennusteiden keskiarvosta. Luokitteluasteen muuttujaa ennustettaessa ennusteen luokka on se, mikä saa eniten ääniä eri luokittelijoilta. Bagging-menetelmän tehokkuus perustuu siihen, että se vähentää varianssia, mutta ei lisää harhaa. Tästä syystä bagging-malli tuottaa parempia ennusteita kuin yksittäinen malli. Bagging toimii erityisen hyvin, kun

perusoppijana on epävakaa malli, jonka varianssi on suuri mutta harha pieni. (Breiman, 1996; Hastie, Tibshirani & Friedman, 2009, s.282-288.)

2.7.1 Satunnainen metsä

Satunnainen metsä on Breimanin (2001) kehittämä malli, joka perustuu bagging-menetelmään. Satunnaisen metsän tarkoituksena on kasvattaa suuri määrä päätöspuita, jotka eivät korreloi keskenään ja ottaa näiden puiden keskiarvo. Kuten aikaisemmin mainittiin, bagging toimii hyvin korkean varianssin ja vähäisen harhan malleille. Tästä syystä päätöspuut soveltuvat hyvin bagging-menetelmään. Päätöspuut voivat mallintaa hyvin monimutkaisia funktioita ja vuorovaikutuksia aineistossa. Jos päätöspuiden annetaan kasvaa tarpeeksi suuriksi, niin niiden harha on kohtalaisen vähäistä. Merkittävin ero satunnaisen metsän ja bagging-menetelmän välillä on se, että kun päätöspuut kasvatetaan satunnaisessa metsässä, niin syötemuuttujista valitaan satunnaisesti vain osa, joita käytetään. Eli kun päätöspuun kasvatuksessa normaalisti aineisto jaetaan jokaisessa solmussa sen muuttujan perusteella mikä parhaiten lisää aineiston puhtautta, niin satunnaisessa metsässä valitaan satunnaisesti m kappaletta muuttujia, joiden perusteella jako tehdään. Jos p on kaikkien syötemuuttujien määrä, niin tyypillisesti m:n arvo on $\sqrt{p}$, tai jopa 1. Tämän toimenpiteen seurauksena kasvatettujen päätöspuiden välinen korrelaatio vähenee. Satunnainen metsä on erittäin suosittu algoritmi. Satunnaisen metsän hyviä puolia on se, että sen hyperparametrejä ei yleensä tarvitse juurikaan virittää hyvien ennusteiden saamiseksi. Tyypillisesti tehostamismenetelmät toimivat paremmin ennustamiseen kuin satunnainen metsä, mutta niiden hyperparametreja joudutaan yleensä säätämään enemmän kuin satunnaisen metsän hyperparametrejä, optimaalisen mallin luomiseksi. Satunnainen metsä-algoritmin toimintaperiaatetta on havainnollistettu kuviossa 8. (Hastie, Tibshirani & Friedman, 2009, s.587-595; Ho, 1995.)



Kuvio 8. Satunnainen metsä (kirjoittajan omat laskelmat).

Yksi satunnaisen metsän hyvistä puolista on myös se, että mallin ennustevirheelle voidaan laskea estimaatti samalla kun malli sovitetaan. Koska bagging-menetelmässä aineistosta otetaan satunnaisotoksia takaisinpanolla, niin koko opetusaineistoa ei käytetä yksittäisen päätöspuun kasvattamiseen. Tätä osaa opetusaineistosta voidaan sitten käyttää ennustevirheen laskemiseksi yksittäiselle puulle. Näin laskettujen ennustevirheiden keskiarvosta saadaan sitten arvio mallin ennustevirheelle. Tätä

ennustevirheen arviota kutsutaan englanniksi Out-of-Bag-virheeksi (OOB error) ja yleensä se antaa kohtalaisen hyvän kuvan mallin suorituskyvystä.

Satunnaisen metsän käyttämien syötemuuttujien tärkeyttä vastemuuttujan ennustamiselle voidaan myös helposti arvioida. Muuttujien tärkeyden arvioimiseen on useita keinoja, mutta yksinkertaisin keino on laskea muuttujien tärkeyttä esimerkiksi gini-epäpuhtauden keskimääräisenä vähenemisenä. Yleensä tämä lasketaan siten, että katsotaan kuinka paljon epäpuhtaus on keskimäärin vähentynyt niissä päätöspuiden solmuissa, joissa kutakin muuttujaa on käytetty aineiston jakamiseen.

2.8 Tehostaminen

Tehostaminen on yksi tehokkaimpia oppimismetodeja mitä on kehitetty viimeisen 20 vuoden aikana. Tehostaminen kehitettiin alun perin luokittelutehtäviä varten, mutta sitä voidaan käyttää myös regressiotehtäviin. Tehostamisen ideana on yhdistää useita ”heikkoja” oppijoita yhdeksi vahvaksi oppijaksi. Heikolla oppijalla tarkoitetaan oppijaa, joka on vain hieman parempi kuin arvaus. Tehostamisessa sovitetaan tällainen heikko oppija/luokittelija useita kertoja peräkkäin aineistoon, jota on muokattu. Tuloksena saatavien heikkojen luokittelijoiden ennusteet yhdistetään lopulliseksi ennusteeksi. Suosituimmat laajasti käytetyt tehostusmenetelmät ovat Adaboost ja gradientti tehostaminen. (Hastie, Tibshirani & Friedman, 2009, s.337-342.)

2.8.1 Adaboost-algoritmi

Adaboost (adaptive boosting) on tehostusalgoritmi, jonka ideana on keskittyä niihin havaintoihin, joita edellinen oppija ei kyennyt ennustamaan hyvin. Tämä tapahtuu siten, että aikaisempien oppijoiden virheellisesti ennustettuja havaintoja painotetaan enemmän kuin oikein ennustettuja havaintoja. Seuraavassa on selostettu adaboost-algoritmin toimintaa luokitteluasteikon muuttujan ennustamiseen.

Jos opetusaineistossa on N kappaletta havaintoja (x_i, y_i), niin aluksi jokaiselle havainnolle asetetaan paino $w_i = 1/N$. Eli ensimmäinen luokittelija sovitetaan aineistoon normaaliin tapaan. Seuraavat luokittelijat sovitetaan aineistoon, jossa havaintojen painoja on muokattu. Edellisen luokittelijan virheellisesti luokittelemien havaintojen painoja lisätään ja oikein luokiteltujen havaintojen painoja vähennetään. Adaboost-luokittelualgoritmin tarkempi toiminta on kuvattu taulukossa 1.

Taulukko 1. Adaboost-luokittelualgoritmi. (Hastie, Tibshirani & Friedman, 2009, s.339.)

1. Aseta havaintojen painot $w_i = \frac{1}{N}, i = 1,2, \dots, N.$
2. Tee kaikille $m = 1, \dots, M$:
   a. Sovita luokittelija $G_m(x)$ opetusaineistoon käyttäen painoja w_i.
   b. Laske $r_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_i(x_i))}{\sum_{i=1}^{N} w_i}$
   c. Laske $\alpha_m = \log((1 - r_m)/r_m)$
   d. Aseta $w_i \leftarrow w_i \cdot \exp\left[\alpha_m \cdot I\left(y_i \neq G_m(x_i)\right)\right], i = 1,2, \dots, N$
3. Vaste: $G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$

Algoritmissa rivillä 2a sovitetaan luokittelija $G_m(x)$ painotettuihin havaintoihin. Tämän luokittelijan painotettu ennustevirhe on laskettu rivillä 2b. Rivillä 2c on laskettu luokittelijalle $G_m(x)$ annettu painoarvo α_m, lopullisessa ennusteessa $G(x)$. Jokaisen havainnon painot seuraavalle iteraatiolle on päivitetty rivillä 2d. Havaintojen, jotka $G_m(x)$ on luokitellut väärin, painot skaalataan tekijällä $exp(\alpha_m)$. Lopullinen vasteen ennuste saadaan kaikkien luokittelijoiden painotettuna äänestystuloksena. (Freund & Schapire, 1997; Hastie, Tibshirani & Friedman, 2009, s.337-339.)

2.8.2 Gradientti tehostaminen

Gradientti tehostaminen on samantapainen vaiheittainen tehostamisalgoritmi kuin adaboost. Toisin kuin adaboostissa, gradientti tehostamisessa ei säädetä havaintojen painoja, vaan gradientti tehostamisessa sovitetaan vaiheittain uusia oppijoita edellisen oppijan ennusteiden residuaaleihin. Yleisellä tasolla gradientti-tehostamisalgoritmi sovittaa joka iteraatiokierroksella oppijan häviöfunktion negatiiviseen gradienttiin, jota kutsutaan myös pseudo-residuaaliksi.

Koneoppimisessa yleisesti käytetty häviöfunktio on niin sanottu neliövirhe-häviöfunktio, joka on muotoa: $\frac{1}{2}[y_i - f(x_i)]^2$. Tämän häviöfunktion negatiivinen gradientti on: $y_i - f_{m-1}(x)$, eli normaali residuaali. Gradientti-tehostamisessa siis sovitetaan aina uusi malli, joka pyrkii parantamaan mallia niiden havaintojen osalta, joita edellinen malli ei kykene ennustamaan hyvin. Lopullinen malli saadaan näiden useiden heikkojen oppijoiden yhdistelmänä. (Breiman, 1997; Hastie, Tibshirani & Friedman, 2009, s.359-361.)

Gradientti-tehostamisalgoritmin toimintaa on kuvattu taulukossa 2. Gradientti-tehostamisessa siis sovitetaan pieniä päätöspuita residuaaleihin. Näin tekemällä parannetaan funktiota $\hat{f}$ siltä osin, kun se ei suoriudu hyvin. Kutistusparametri λ hidastaa opetusprosessia entisestään. Yleisesti ottaen hitaasti oppivat menetelmät toimivat hyvin. Tässä on kuvailtu tehostaminen regressio-ongelman yhteydessä, luokittelutilanteessa tehostaminen toimii samankaltaisesti, mutta sitä ei nyt käydä tässä läpi. Tehostamisessa säädettäviä parametreja ovat päätöspuiden määrä B, kutistusparametri λ ja päätöspuiden kokoon vaikuttava parametri d. Yleensä tehostamisessa käytetään päätöspuiden tynkiä, joissa aineisto jaetaan vain kerran, eli $d = 1$. Parametrien B ja λ arvot selvitetään totuttuun tapaan esimerkiksi ristiinvalidoinnin avulla. Kutistusparametrin λ arvo on tyypillisesti 0.01 tai 0.001. Kutistusparametri vaikuttaa siihen, kuinka nopeasti malli oppii ja siten myös siihen, mikä on optimaalinen päätöspuiden määrä. Eli mitä pienempi kutistusparametri on, sitä enemmän tarvitaan päätöspuita. Gradientti-tehostamisessa ylisovittaminen on mahdollista, jos B on liian suuri. (James, Witten, Hastie & Tibshirani, 2013, s.321-324.)

Taulukko 2. Gradientti tehostaminen algoritmi (James, Witten, Hastie & Tibshirani, 2013, s.323)

1. Aseta $\hat{f}(x) = 0$ ja $r_i = y_i$ kaikille instansseille i opetusaineistossa
2. Tee kaikille $b = 1,2, \dots, B$:
    a. Sovita päätöspuu $\hat{f}^b(x)$ opetusaineistoon (X, r), jakaen aineisto d kertaa ($d + 1$ lehtisolmua)
    b. Päivitä $\hat{f}$ lisäämällä kutistettu versio uudesta päätöspuusta: $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$
    c. Päivitä residuaalit: $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$

3. Lopullinen malli:

$$\hat{f}^b(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

2.9 Osittaisen riippuvuuden kuvaajat

Edellä kuvattujen yhdistelmämenetelmien (tehostaminen, bagging) tulkinta on haasteellista. Perinteisesti näitä menetelmiä on pidetty niin sanottuina ”black box”-malleina. Olisi kuitenkin hyvä, jos näiden mallien tuottamia ennusteita voitaisiin jotenkin tulkita. Osittaisen riippuvuuden kuvaajilla pyritään avaamaan näiden mallien toimintaa, ja sitä miten jotkin tietyt muuttujat vaikuttavat vasteeseen. Osittaisen riippuvuuden kuvaajia voidaan käyttää selvittämään, onko jonkin syötemuuttujan ja vasteen välinen suhde esimerkiksi lineaarinen, monotoninen vai onko kyseessä monimutkaisempi suhde. Osittaisen riippuvuuden funktio on määritelty seuraavalla tavalla:

$$f_s(X_s) = E_{X_C} f(X_S, X_C) \tag{3}$$

Edellä X_S on niiden syötemuuttujien joukko, joiden osittaista riippuvuutta on tarkoitus kuvata, ja X_C sisältää loput mallissa käytetyistä ennustemuuttujista. Tyypillisesti joukossa X_S muuttujia on 1 tai 2. Osittaista riippuvuutta voidaan estimoida seuraavalla kaavalla:

$$\overline{f_s}(X_s) = \frac{1}{N} \sum_{i=1}^{N} f(X_S, x_{iC}) \tag{4}$$

Tässä kaavassa x_{iC} kuvaa todellisia syötemuuttujan arvoja opetusaineistossa. N on havaintojen/instanssien määrä opetusaineistossa. Mallin ennusteiden muutoksia muuttujan X_s eri arvoilla voidaan estimoida edellisestä kaavasta käyttämällä eri X_s:n arvoja. Käytännössä siis $\overline{f_s}(X_s)$:n arvo lasketaan käyttäen sovitettua mallia ja opetusaineiston muuttujien X_C arvoja, pitäen muuttujien X_s arvot vakiona. Jokaiselle käytetylle X_s:n arvolle saadaan siis N kappaletta arvoja. Lopullinen osittaisen

riippuvuuden estimaatti saadaan laskemalla näiden arvojen keskiarvo. Osittaisen riippuvuuden funktio kertoo siis mallin keskimääräisen ennusteen tietyille muuttujan X_s arvoille. (Hastie, Tibshirani & Friedman, 2009, s.369-370.)

Osittaisen riippuvuuden funktioita kuvataan yleensä graafisesti. Kun kuvataan yhden muuttujan osittaista riippuvuutta, graafisesta kuvaajasta tulee kaksiulotteinen, ja kun muuttujia on kaksi kuvaajasta tulee komiulotteinen. Kolmiulotteisesta kuvaajasta on mahdollista havaita myös mahdolliset interaktiot eri muuttujien välillä. Osittaisen riippuvuuden kuvaajien yksi heikkous on kuitenkin siinä, että tarkasteltavien muuttujien määrä on hyvin rajoittunut käytännössä kahteen. Suurin ongelma osittaisen riippuvuuden kuvaajissa on se, että muuttujien X_s arvot eivät välttämättä ole riippumattomia muuttujien X_C arvoista. Osittaisen riippuvuuden funktiota estimoidessa oletetaan, että näiden muuttujien välillä ei ole korrelaatioita. Tämä ei välttämättä pidä paikkaansa. Osittaisen riippuvuuden funktiossa ei myöskään oteta huomioon mahdollisia heterogeenisiä vaikutuksia. Kaavassa 4 osittainen riippuvuus lasketaan kaikkien havaintojen keskiarvona. Jos havainnoissa on suurta hajontaa, osittaisen riippuvuuden kuvaaja voi antaa virheellisen kuvan muuttujan X_s vaikutuksesta vasteeseen. (Hastie, Tibshirani & Friedman, 2009, s.369-370.)

2.10 Lasso- ja ridge-regressiot

Lineaarinen regressio on yksi käytetyimpiä malleja ekonometriassa. Lineaarinen regressio ei kuitenkaan toimi välttämättä tilanteissa, joissa syötemuuttujia on paljon. Eri syötemuuttujat voivat korreloida keskenään ja liian monimutkainen malli voi johtaa ylisovittamiseen. Yksi keino estää lineaarisen regression ylisovittaminen on rajoittaa muuttujien määrää. Tämä voidaan tehdä muokkaamalla häviöfunktiota lisäämällä niin sanottu rangaistustermi, joka estää liian monimutkaisen mallin sovittamisen. Tällaisia lineaarisen regression regularisointimenetelmiä ovat ridge- ja lasso-regressiot. (Hastie, Tibshirani & Friedman, 2009, s.61-73.)

Ridge-regressio kutistaa regressiokertoimet asettamalla rangaistuksen niiden koolle. Useimmiten lineaarisen regression regressiokertoimet estimoidaan pienimmän neliösumman menetelmällä. Regressioyhtälö, josta lineaarisen regression regressiokertoimet ratkaistaan, voidaan esittää kompaktisti matriisimuodossa:

$$Y = X\beta + U \quad . \tag{5}$$

Edellä Y on $n \times 1$ vektori n:stä vastemuuttujan havainnosta. X on $n \times (k+1)$ matriisi syötemuuttujien havainnoista k:lle syötemuuttujalle. β on $(k+1) \times 1$ tuntemattomien regressiokertoimien vektori. U on $n \times 1$ virhetermien vektori. Pienimmän neliösumman estimaattori minimoi ennustevirheiden (residuaalien) neliöiden summan. Ennustevirhe voidaan esittää matriisimuodossa seuraavalla tavalla:

$$RSS(\beta) = (y - X\beta)^T(y - X\beta). \tag{6}$$

Tästä yhtälöstä β , joka minimoi residuaalit, voidaan ratkaista derivoimalla:

$$\frac{\partial RSS}{\partial \beta} = -2X^T(y - X\beta) = 0$$
$$\leftrightarrow\ X^T(y - X\beta) = 0$$
$$\leftrightarrow\ \hat{\beta} = (X^TX)^{-1}X^Ty \quad .$$

Ridge-regressiossa lineaariseen regressioon lisätään rangaistustermi $\lambda\beta^2$, missä λ on kerroin, jolla säädetään rangaistustermin voimakkuutta. Matriisimuodossa ridge-regression residuaalit ovat:

$$RSS(\lambda) = (y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta \, . \tag{7}$$

Ridge-regression regressiokertoimet puolestaan ovat:

$$\hat{\beta}^{ridge} = (X^TX + \lambda I\,)^{-1}X^Ty \, , \tag{8}$$

missä I on yksikkömatriisi. Kun $\lambda = 0$, ridge-regressio on yhtä kuin tavallinen lineaarinen regressio. Jos taas $\lambda = \infty$, kaikki regressiokertoimet kutistuvat nollaan.

Lasso-regressiossa lineaariseen regressioon lisätään rangaistustermi $\lambda|\beta|$. Lasso- ja ridge-ressioiden välinen ero on se, että lasso-regressiossa merkityksettömien

muuttujien regressiokertoimet kutistuvat nollaan, kun taas ridge-regressiossa regressiokertoimet ovat aina suurempia kuin nolla. Lasso- ja ridge-regressioissa rangaistustermin vaikutusta voidaan säätää muuttamalla parametrin λ arvoa. Sopiva lambdan arvo etsitään esimerkiksi ristiinvalidoinnin avulla.

2.11 ROC-käyrä ja AUC-arvo

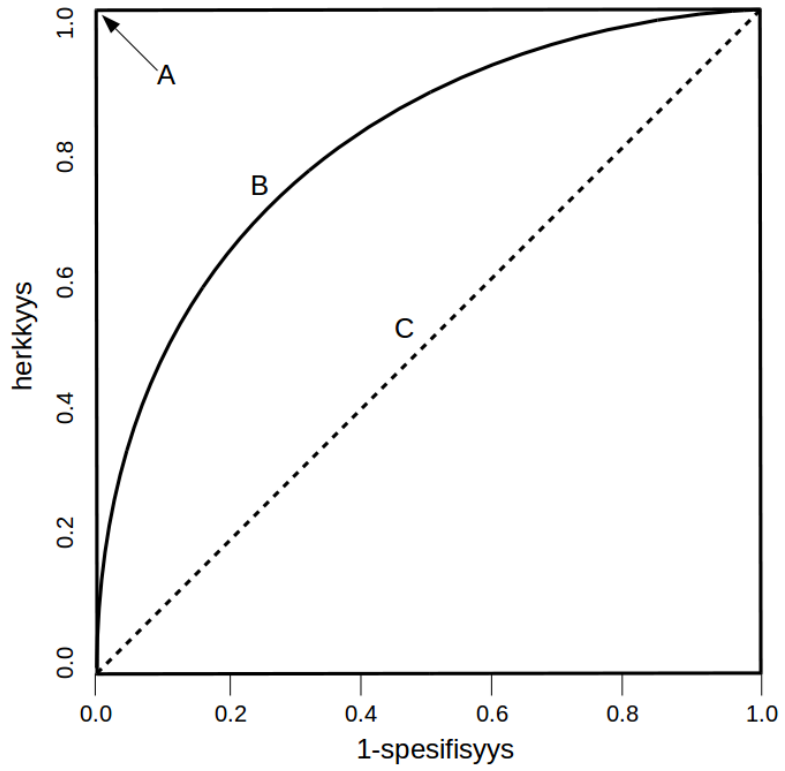
Eri koneoppimismallien ennustamistarkkuutta voidaan arvioida muutamilla eri tavoilla. Jatkuvan vastemuuttujan tapauksessa eri mallien ennustetarkkuutta yleensä arvioidaan ennustevirheen neliösummalla tai keskineliövirheen neliöjuurella.

Eri luokittelijoiden ennustamiskyvyn arvioiminen ei ole välttämättä yhtä suoraviivaista. Luonnollinen lähtökohta luokittelijan ennustuskyvyn arvioinnille on sen osumatarkkuus (accuracy), eli kuinka monta havaintoa luokittelija kykenee ennustamaan oikeaan luokkaan kaikista ennustettavista instansseista. Ongelmia voi kuitenkin aiheutua, jos ennustettavan muuttujan luokat ovat epätasapainossa. Jos esimerkiksi binäärisen muuttujan tapauksessa toiseen luokkaan kuuluu 90% havainnosta, niin luokittelijan, joka luokittelee kaikki instanssit tähän luokkaan, osumatarkkuus on 90%. Jotta eri luokittelijoiden suorituskyvystä saataisiin luotettavampi arvio, käytetään usein ROC (Receiving operator characteristics) -analyysiä luokittelijoiden vertailuun.

Kaksi oleellista tunnuslukua, joilla luokittelijan tarkkuutta voidaan arvioida ovat herkkyys (sensitivity) ja spesifisyys (specifity). Herkkyydellä tarkoitetaan oikeiden positiivisten ennusteiden osuutta ja spesifisyydellä tarkoitetaan oikeiden negatiivisten ennusteiden osuutta. ROC-käyrä muodostamiseksi määritetään herkkyys ja spesifisyys eri kynnysarvoilla. (Zou, Malley & Mauri, 2007.)

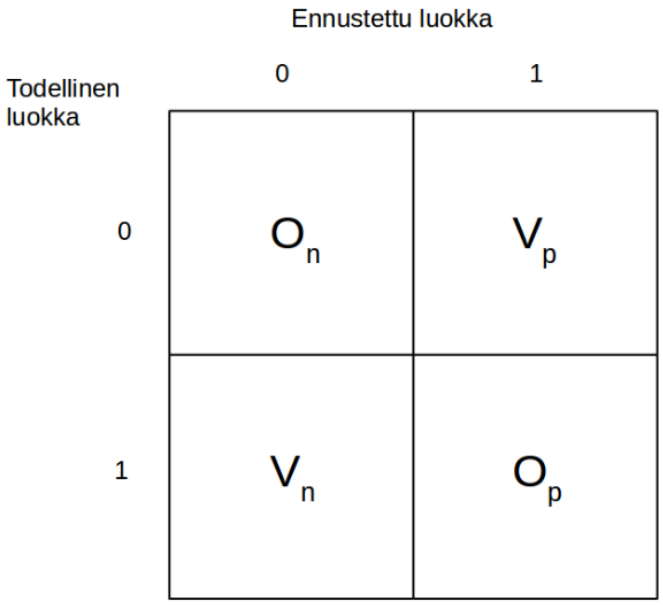
Kynnysarvo on arvo, joka määrittää mihin luokkaan instanssit luokitellaan luokittelijan estimoimien todennäköisyyksien perusteella. Jos esimerkiksi kynnysarvo on 0.5, niin kaikki instanssit, joille luokittelija ennustaa todennäköisyyden, joka on yli 0.5, luokitellaan kuuluvaksi positiiviseen luokkaan.

Kun herkkyys ja spesifisyys on määritetty eri kynnysarvoilla, voidaan muodostaa kuvaaja, missä vaaka-akselilla on $1 - spesifisyys$ eli väärien positiivisten osuus ja pystyakselilla herkkyys eli oikeiden positiivisten osuus. ROC-käyrää on havainnollistettu kuviossa 9. ROC-käyrän alle jäävää pinta-alaa voidaan käyttää tunnuslukuna, joka kuvastaa luokittelijan kykyä erotella eri luokkiin kuuluvat havainnot. Kun luokittelija kykenee erottelemaan havainnot täydellisesti, ROC-käyrä kulkee kuvaajan akseleita pitkin ja käyrän alle jäävän alueen pinta-ala on 1. Kuvioon 9 piirretty 45 asteen suora C kuvaa luokittelijaa, joka on yhtä huono kuin satunnainen arvaus. Suoran alapuolelle jäävän alueen pinta-ala on 0.5. ROC-käyrän alle jäävästä pinta-alasta käytetään lyhennettä AUC (area under the curve). Tyypillisesti luokittelijoiden AUC-arvot ovat 0.5 ja 1 välillä. Mitä korkeampi AUC-arvo on, sitä paremmin luokittelija kykenee erottelemaan havainnot eri luokkiin. ROC-analyysi on alun perin kehitetty toisen maailmansodan aikaan tutkasignaalien luotettavuuden arviointiin, mutta sitä käytetään hyvin yleisesti koneoppimisalgoritmien arvioinnissa. (Bradley, 1997; Zou, Malley & Mauri, 2007.)



Kuvio 9. ROC-käyrä (mukaillen Zou, Malley & Mauri, 2007)

Luokittelijan tuottamat ennusteet voidaan myös esittää taulukkomuodossa. Tällaista taulukkoa on havainnollistettu kuviossa 10. Taulukon solut muodostuvat luokittelijan eri luokkien oikein ja väärin luokiteltujen instanssien lukumäärästä. Taulukon riveillä ovat instanssien todelliset luokat ja taulukon sarakkeilla ovat luokittelijan ennustamat luokat. Kuviossa 10 O_n on siis oikeiden negatiivisten lukumäärä, V_n on väärien negatiivisten lukumäärä, V_p on väärien positiivisten lukumäärä ja O_p on oikeiden positiivisten lukumäärä. (Bradley, 1997.)



Kuvio 10. Luokittelumatriisi (mukaillen Bradley, 1997)

3 KONEOPPIMINEN JA TALOUSTIEDE

3.1 Koneoppimismenetelmien ja ekonometrian erot

Perinteisissä ekonometrian menetelmissä, estimoitavan funktion muoto täytyy määritellä ennen kuin se sovitetaan (Boelaert & Ollion, 2018). Yleensä tutkijat käyttävät teoriaa tai asiantuntemusta funktion muodon ja siihen valittavien muuttujien valinnassa. Koneoppimisessa puolestaan etsittävän funktion muotoa ei tarvitse etukäteen määritellä. Oikeastaan koneoppimisessa ajatellaan, että todellinen funktio on niin monimutkainen, että sen muotoa ei edes voida tietää etukäteen (Breiman, 2001). Siitä, että mallinnettavan funktion muotoa ei tarvitse etukäteen määritellä, on useita etuja. Ensinnäkin koneoppimisalgoritmit käyvät läpi lukuisia eri malleja, joista valitaan paras. Käytännössä tutkijan on mahdotonta edes teoriassa harkita niin montaa erilaista mallia, joita koneoppimisalgoritmit harkitsevat. Tämä koneoppimismenetelmien ominaisuus voisi myös viedä taloustieteellistä tutkimusta enemmän empiriaan pohjautuvaksi, kun malli rakennetaan aineiston pohjalta eikä teoreettisten mallien perusteella. Koneoppimismenetelmiä käytettäessä tutkijat voivat myös kuvata yksiselitteisesti mallin valintaan johtaneen prosessin. Perinteisissä regressiomalleissa tutkijan täytyy tehdä useita valintoja mallin suhteen, esimerkiksi mitkä muuttujat valitaan ja lisätäänkö malliin esimerkiksi interaktiotermejä tai muuttujien muunnoksia. Usein muuttujia valitaan esimerkiksi aineistosta muodostettujen kuvaajien avulla. Tutkijoiden täytyy siis tehdä useita valintoja tutkimusprosessissa, ja ne kaikki pitäisi pystyä perustelemaan ja dokumentoimaan. Jokainen valinta on myös potentiaalinen kritiikin kohde (Athey, 2018). Koneoppimismenetelmiä käytettäessä ei tarvitse tehdä yhtä paljon valintoja mallin suhteen, ja mallin valintaprosessi voidaan myös dokumentoida siten, että se on toistettavissa (Athey, 2018).

Myös koneoppimismenetelmien käyttöön liittyy joitain valintoja. Esimerkiksi päätöspuiden käytössä täytyy päätöspuiden kokoa rajoittaa hyperparametrien avulla. Päätöspuilla on kuitenkin useita hyperparametreja, kuten maksimisyvyys, instanssien minimimäärä lehtisolmuissa ja niin edelleen. Se, mitä hyperparametreja käytetään vaikuttaa lopullisen päätöspuun rakenteeseen, mutta siihen mitä hyperparametreja tulisi käyttää ei ole mitään selkeää vastausta. Simulaatiot ja asiantuntijoiden intuitio

käytännössä usein määrittävät käytetyt hyperparametrit. Myös se, käytetäänkö esimerkiksi ristiinvalidointia vai erillistä validointiaineistoa hyperparametrien säätämiseen on tutkijan itse päätettävä. Myöskään näihin koneoppimismenetelmiin liittyviin valintoihin ei ole usein olemassa mitään selkeitä oikeita ratkaisuja. (Mullainathan & Spiess, 2017.)

Toinen merkittävä ero koneoppimismenetelmien ja perinteisten ekonometrian menetelmien välillä on se, että esimerkiksi lineaarisilla regressiomalleilla on vahva teoreettinen pohja, kun taas koneoppimismenetelmien suosio perustuu ennen kaikkea niiden empiiriseen tehokkuuteen (Boelaert & Ollion, 2018). Koneoppimisalgoritmitkin yleensä pohjautuvat jonkinlaiseen teoriaan, mutta käytännössä käytetään niitä menetelmiä jotka tuottavat parhaita tuloksia, eikä koneoppimismenetelmien käyttöä pyritä perustelemaan teorioiden avulla. Lineaaristen regressiomallien ratkaisu saadaan myös yksiselitteisesti matemaattisen kaavan avulla, kun taas koneoppimismenetelmien tulokset ovat aina arvioita, eikä koskaan voida olla täysin varmoja, että niiden tulokset ovat optimaalisia. On aina olemassa mahdollisuus, että joitain hyperparametreja muuttamalla saataisiin muodostettua entistä parempi malli. Joihinkin algoritmeihin liittyy myös hieman satunnaisuutta (esim. satunnainen metsä), mikä tarkoittaa, että sama algoritmi voi tuottaa hieman erilaisia tuloksia, vaikka käytettäisiin samaa aineistoa ja samoja hyperparametreja. (Boelaert & Ollion, 2018.)

Koneoppimismenetelmien ja regressiomallien ”hyvyyttä” arvioidaan myös eri tavalla. Regressiomallin hyvyyttä arvioidaan determinaatiokertoimen avulla. Determinaatiokerroin kertoo kuinka paljon selittävä muuttuja selittää selitettävän muuttujan kokonaisvaihtelusta. Koneoppimismallien hyvyyttä puolestaan arvioidaan puhtaasti niiden kyvyllä ennustaa uusia havaintoja (Boelaert & Ollion, 2018). Tämä mitataan joko erillistä testiaineistoa hyväksi käyttäen tai ristiinvalidoinnin avulla. Tätä perustellaan sillä, että jos malli kykenee ennustamaan hyvin, niin silloin se myös kuvaa hyvin sitä todellista funktiota, jota sillä pyritään mallintamaan (Breiman, 2001). Breimanin (2001) mukaan tilastollisen analyysin tarkoituksena on saada mahdollisimman luotettavaa informaatiota, ja malli, jonka ennustamistarkkuus on hyvä, tuottaa luotettavampaa informaatiota kuin malli, jonka ennustamistarkkuus on

huono. Koneoppimismallit ovat usein parempia ennustamaan kuin lineaariset mallit, ja siten myös niiden tuottamaa informaatioita voidaan pitää luotettavampana.

Mallin hyvyyden arviointi esimerkiksi determinaatiokertoimen avulla on ongelmallista myös siksi, että sen avulla ei välttämättä voida tehdä eroa eri mallien välillä. Jos esimerkiksi aineistossa on paljon muuttujia, niin voidaan sovittaa useita yhtä hyviä malleja, joissa käytetään eri muuttujia. Usein eri mallien determinaatiokertoimet voivat olla yhtä hyvät, mutta eri malleissa muuttujien väliset suhteet ovat erilaisia. Eli eri mallit voivat tuottaa hyvin erilaista informaatioita, vaikka ne istuvat aineistoon yhtä hyvin. Sitä, miten hyvin malli istuu aineistoon, voidaan arvioida myös esimerkiksi AIC- ja BIC-informaatiokriteerien avulla, mutta jos useiden eri mallien R^2, AIC- ja BIC-informaatiokriteerit ovat samankaltaisia, on hyvin vaikea sanoa mikä niistä on paras. (Breiman, 2001.)

Myös joihinkin koneoppimismenetelmiin liittyy vastaanvanlaisia ongelmia. Mullainathan ja Spiess osoittavat, että myös esimerkiksi lasso-regression avulla voidaan sovittaa useita malleja saman aineiston eri osiin, niin että eri malleissa eri muuttujien regressiokertoimet kutistuvat nollaan. Näiden eri mallien ”hyvyys” voi kuitenkin olla yhtä hyvä. Toisin sanoen myös koneoppimismenetelmiä vaivaa epävakaus. Pohjimmiltaan tämä epävakaus johtuu siitä, että muuttujat korreloivat keskenään. Eri malleissa muuttujat voidaan korvata toisella muuttujalla, joka korreloi kyseisen muuttujan kanssa, ja mallin hyvyys voi olla yhtä hyvä. Koneoppimismenetelmien suuri houkutus on juuri siinä, että ne voivat sovittaa useita eri funktioita. Tämä kuitenkin voi johtaa juuri siihen, että useita eri funktioita voidaan sovittaa aineistoon käyttäen eri muuttujia, ja niiden ennustamistarkkuus voi olla yhtä hyvä. Se minkä funktion algoritmi valitsee, voi olla täysin sattuman varassa. (Mullainathan & Spiess, 2017.)

Lineaaristen regressiomallien vahvuus on siinä, että niiden avulla saadaan helposti tietoa eri muuttujien välisistä yhteyksistä (Breiman, 2001). Taloustieteellisessä tutkimuksessa pyritään usein selvittämään miten jokin selittävä muuttuja vaikuttaa vastemuuttujaan. Perinteisten regressiomallien tuloksista saatavien regressiokertoimien/parametrien avulla muuttujien välisiä suhteita on kohtalaisen helppo tulkita. Koneoppimismenetelmien tulokset ovat vaikeammin tulkittavissa.

Ensinnäkään koneoppimismallien tuloksena ei saada muuttujien parametreja. Toiseksi, eri koneoppimismallien tulkittavuuden välillä on eroja. Yleisesti ottaen mallit, joita on helpompi tulkita, eivät ole yhtä tehokkaita ennustettaessa kuin monimutkaisemmat mallit, joiden tulkinta on vaikeampaa (Breiman, 2001). Esimerkiksi päätöspuita on kohtalaisen helppo tulkita, ja niiden tuloksista voidaan selvittää muuttujien välisiä yhteyksiä. Valitettavasti päätöspuut eivät ole kovin hyviä ennustamaan verrattuna tehokkaampiin menetelmiin. Yleensä parhaita ennusteita tuottavat menetelmät, kuten esimerkiksi satunnainen metsä ja tehostaminen, ovat niin sanottuja black box-malleja, joiden toimintaa on vaikea tulkita.

Näiden black box-mallien tulkintaan on kuitenkin kehitetty joitain menetelmiä. Esimerkiksi satunnainen metsä-malleista voidaan selvittää ennustamisen kannalta tärkeimmät muuttujat. Yleensä tärkeimmiksi muuttujiksi tulkitaan ne, jotka parhaiten parantavat ennustamistarkkuutta. Tämän lisäksi koneoppimismallien tulkintaan voidaan käyttää luvussa 2.9 kuvattuja osittaisen riippuvuuden kuvaajia. Näiden kuvaajien avulla voidaan selvittää ainakin kvalitatiivisesti eri selittävien muuttujien vaikutusta vasteeseen. Lisäksi viime aikoina on kehitetty useita muita menetelmiä, joiden avulla black box-malleja voidaan tulkita (Ribeiro, Singh ja Guestrin 2016; Lundberg ja Lee 2017; Biecek 2018). Black box-mallien tulkintaan on siis useita menetelmiä, mutta mitään varsinaisia standardimenetelmiä tulkintaan ei ole. Yleensä malleja kuitenkin tulkitaan lähinnä muuttujien tärkeyden ja osittaisen riippuvuuden kuvaajien perusteella. Koneoppimismenetelmät ovat vielä suhteellisen tuoreita, eikä tulosten tulkintaa ole perinteisesti pidetty tärkeänä. Kun koneoppimismenetelmiä aletaan soveltamaan uusilla alueilla ja tulkinnan tärkeys korostuu, niin todennäköisesti myös kehitetään parempia menetelmiä, joilla koneoppimismallien tuloksia voidaan tulkita.

3.2 Koneoppiminen taloustieteellisessä tutkimuksessa

Koneoppimismenetelmiä on käytetty vielä kohtalaisen vähän taloustieteellisessä tutkimuksessa. Ehkä merkittävin ongelma koneoppimismenetelmien käytössä on se, että ne on kehitetty puhtaasti ennustamiseen. Taloustieteellisessä tutkimuksessa tavoitteena on usein selvittää jonkinlaisia syy-seuraussuhteita, eikä niinkään ennustaa uusia havaintoja. Luotettavimpana tapana selvittää syy-seuraussuhteita tieteellisessä

tutkimuksessa pidetään yleisesti satunnaistettuja kontrolloituja tutkimuksia. Taloustieteellisessä tutkimuksessa tällaisten tutkimusten tekeminen on vaikeaa tai jopa mahdotonta. Usein taloustieteilijät joutuvat turvautumaan aineistoihin, joita ei ole kerätty kokeellisesti. Ekonometriassa on kehitetty useita menetelmiä syy-seuraussuhteiden selvittämiseen tällaisista aineistoista. Koneoppimismenetelmiä taas ei ole alun perin tarkoitettu syy-seuraussuhteiden selvittämiseen. Koneoppimiskirjallisuudessa kausaalisuhteiden selvittämistä ei ole perinteisesti käsitelty millään tavalla. Koneoppimismenetelmät käytännössä etsivät aineistosta muuttujien välisiä korrelaatioita, eikä korrelaatio implikoi kausaliteettia (Domingos, 2012). Kausaliteetti kuitenkin implikoi korrelaatiota, ja korrelaatioita voidaan käyttää apuna potentiaalisten kausaalisuhteiden löytämiseen. Ennustamista voidaan myös käyttää joissain tapauksissa suoraan kausaalivaikutusten estimointiin. Joissain ekonometrian menetelmissä voidaan myös implisiittisesti käyttää ennustamista kausaalivaikutusten estimointiin. Viime vuosina koneoppimismenetelmiä on myös pyritty kehittämään niin, että ne soveltuisivat kausaalivaikutusten estimointiin.

Kuten aiemmin todettiin, usein tutkimuksessa halutaan selvittää syy-seuraussuhteita. Tavoitteena on siis selvittää mitä tapahtuisi, jos jotain muutettaisiin. Paras ja luotettavin tieteellinen tapa selvittää syy-seuraussuhteita on tehdä kontrolloitu satunnaistettu koe. Kontrolloiduissa kokeissa kohdistetaan jokin ”hoito” osalle ”subjekteista” ja sitten mitataan vastemuuttujan arvo. Näitä hoidettujen subjektien vasteiden arvoja sitten verrataan kontrollisubjektien vasteisiin, ja näin voidaan päätellä kausaalivaikutus. Usein tällaisten kokeiden tekeminen on kuitenkin mahdotonta tai kallista. Varian (2016) esittää yksinkertaisen tavan estimoida esimerkiksi mainoskampanjan vaikutusta myyntiin ennustamisen avulla. Mainoskampanjan vaikutusta myyntiin voidaan estimoida tekemällä lyhytaikainen koe, missä mainostukseen käytettyjä menoja lisätään, ja kontrolli, eli se kuinka paljon myynti olisi ollut ilman mainoskampanjaa, ennustetaan. Ennustamiseen voidaan käyttää koneoppimismenetelmiä ja aineistoa, joka on kerätty ennen kokeen aloittamista. Näin voidaan verrata ”hoitoryhmää” ja kontrolliryhmää. Hoitoryhmän ja kontrolliryhmän myyntien erotuksesta saadaan arvio mainoskampanjan vaikutuksesta ilman todellista kontrolliryhmää. Kontrollin ennustaminen voi olla jopa parempi tapa selvittää mainoskampanjan vaikutusta, koska siinä voidaan pitää kaikki muut tekijät vakioina. Jos mainoskampanjan vaikutusta estimoitaisiin todellisen kontrollin avulla,

esimerkiksi jos jollain toisella alueella mitattaisiin myynti ilman mainoskampanjaa, niin tällä alueella jotkin muut muuttujat, kuten esimerkiksi sää, voisivat vaikuttaa myyntiin, eikä kontrolli olisi vertailukelpoinen. (Varian, 2016; Varian 2014.)

Joissain tapauksissa kausaalivaikutusten estimointiin käytetään ekonometrian menetelmiä, joihin implisiittisesti sisältyy ennustamista. Eräs tällainen menetelmä on instrumenttimuuttujaregressio (IM-regressio). IM-regressiota käytetään, kun halutaan estimoida jonkin muuttujan x vaikutusta muuttujaan y regression avulla, mutta muuttuja x korreloi regressioyhtälön virhetermin kanssa. Gauss-Markov -teoreeman mukaan, jotta muuttujan x regressiokerroin β on harhaton, x ei saa korreloida virhetermin kanssa. Periaatteessa tämä korrelaatio johtuu siitä, että regressiosta puuttuu jokin muuttuja. Se, että regressioyhtälöstä puuttuu muuttuja, ei sinänsä ole ongelma. Regressioyhtälöstä puuttuu käytännössä aina muuttujia. Ongelma johtuu siitä, että puuttuva muuttuja vaikuttaa sekä selitettävään muuttujaan että selittävään muuttujaan. β:n harhaton estimaatti voidaan kuitenkin estimoida, jos löydetään jokin muuttuja, joka vaikuttaa selitettävään muuttujaan y, mutta pelkästään selittävän muuttujan x kautta. Tällaisen instrumenttimuuttujan täytyy siis korreloida muuttujan x kanssa, mutta se ei saa korreloida regression virhetermin kanssa. Mikäli tällainen instrumenttimuuttuja Z löydetään, regressiokerroin β voidaan estimoida kaksivaiheisen proseduurin avulla. (Varian, 2016; Stock & Watson, 2012.)

Kaksivaiheisen proseduurin ensimmäisessä vaiheessa regressoidaan x:ää instrumenttimuuttujaan z. Tästä regressiosta lasketaan x:n sovitetut arvot $\hat{x}$. Toisessa vaiheessa regressoidaan y:tä näihin endogeenisen muuttujan sovitettuihin arvoihin $\hat{x}$ pienimmän neliösumman menetelmällä. Kaksivaiheisen regression estimaattorit ovat tästä regressiosta saatavat regressiokertoimet. (Varian, 2016.)

Tyypillisesti tämän kaksivaiheisen menetelmän ensimmäistä vaihetta on käsitelty estimointitehtävänä, mutta tosiasiassa kyseessä on ennustamistehtävä. Ensimmäisessä vaiheessa estimoituja regressiokertoimia käytetään ainoastaan keinona sovitettujen arvojen hankkimiseksi. Koneoppimismenetelmiä voidaan hyödyntää tässä IM-regression ensimmäisessä vaiheessa. Koneoppimisessa käytettävät menetelmät, joilla estetään ylisovittamista voivat parantaa optimaalisten

instrumenttien löytämistä ensimmäisessä vaiheessa. Esimerkiksi Belloni ja Chernukov (2012) käyttivät luvussa 2.10 esitettyä Lasso-menetelmää IM-regression ensimmäisen vaiheessa sovitettujen arvojen muodostamiseen ja optimaalisten instrumenttien löytämiseen tilanteessa missä instrumenttien määrä on suuri. (Mullainathan & Spiess, 2017.)

3.3 Koneoppiminen ja kausaliteetti

Koneoppimismenetelmiä on myös viime vuosina pyritty kehittämään, niin että niitä voitaisiin käyttää kausaalisuhteiden tutkimiseen. Menetelmiä on kehitetty esimerkiksi keskimääräisen hoitovaikutuksen (average treatment effect/ average causal effect) arviointiin (Athey & Imbens 2018; Chernozhukov, Chetverikov, Demirer, Duflo, Hansen, & Newey, 2016). Keskimääräisellä hoitovaikutuksella tarkoitetaan esimerkiksi lääkkeen vaikutusta populaatiossa keskimäärin. Kun halutaan selvittää hoitovaikutusta jollekin erilliselle ryhmälle, puhutaan heterogeenisestä hoitovaikutuksesta. Esimerkiksi jonkin lääkkeen vaikutus voi riippua lääkkeen saajan ominaisuuksista. Myös heterogeenisiä hoitovaikuksia voidaan arvioida koneoppimismenetelmillä (Athey & Imbens, 2015; Green & Kern, 2012).

Hoitovaikutuksella tarkoitetaan siis sitä eroa vastemuuttujassa Y, kun hoidettava altistetaan jollekin hoidolle W. Hoitoa kuvataan binäärisellä muuttujalla W. Hoitovaikutus on potentiaalisten vastemuuttujien arvojen välinen erotus (Athey & Imbens, 2015):

$$\tau_i = Y_i(1) - Y_i(0) \quad , \tag{9}$$

missä τ_i on hoitovaikutus, $Y_i(1)$ on vaste, mikäli potilas on saanut hoidon ja $Y_i(0)$ on vaste, jos potilas ei ole saanut lääkettä. Athey ja Imbens pyrkivät estimoimaan heterogeenistä hoitovaikutusta, eli hoitovaikutusta jollekin tietylle yksilölle tai osalle populaatiosta. Ehdollinen (heterogeeninen) keskimääräinen hoitovaikutus jollekin osapopulaatiolle x ilmaistaan seuraavalla tavalla:

$$\tau(x) = E[Y_i(1) - Y_i(0)|X_i = x]. \quad (10)$$

Ongelma $\tau(x)$:n selvittämisessä on se, että kullekin yksilölle havaitaan aina joko $Y(1)$ tai $Y(0)$, mutta ei molempia. Sama henkilö ei voi sekä saada hoitoa että olla saamatta hoitoa yhtä aikaa. Tästä syystä $\tau(x)$ täytyy estimoida.

Athey ja Imbens (2015) esittävät neljä eri päätöspuihin perustuvaa algoritmia heterogeenisten hoitovaikutusten estimoimiseen. Ensimmäinen yksinkertainen algoritmi on nimeltään yhden puun algoritmi. Yhden puun algoritmissa hoitomuuttujaa W käytetään mallissa yhtenä selittävänä muuttujana. Eli algoritmissa W ja X_i ovat selittäviä muuttujia, ja Y on selitettävä muuttuja. Y:n odotusarvo μ osapopulaatiolle x estimoidaan päätöspuun avulla:

$$\mu(w, x) = E[Y_i^{obs}|W_i = w, X_i = x]. \quad (11)$$

Eli kun päätöspuu on kasvatettu, sitä käytetään kahdesti μ:n estimoimiseen. Ensin siten, että hoitomuuttujan arvoksi on asetettu 1, ja sitten siten että hoitomuuttujan arvoksi on asetettu 0. Ehdollinen hoitovaikutus saadaan estimoitua odotusarvojen erotuksena (Athey & Imbens, 2015):

$$\hat{\tau}(x) = \hat{\mu}(1, x) - \hat{\mu}(0, x). \quad (12)$$

Toinen menetelmä, jonka Athey ja Imbens esittävät on niin sanottu kahden puun algoritmi. Kahden puun algoritmissa hoitovaikutuksen estimointi jaetaan kahdeksi erilliseksi ongelmaksi. Käytännössä aineisto jaetaan kahteen osaan, ja luodaan kaksi eri mallia, joita käytetään $\hat{\mu}(1, x)$:n ja $\hat{\mu}(0, x)$:n estimoimiseen. Menetelmässä kasvatetaan kaksi päätöspuuta. Ensimmäinen päätöspuu kasvatetaan käyttäen sitä aineiston osaa missä $W = 1$, ja toinen päätöspuu käyttäen sitä aineiston osaa missä $W = 0$. Molemmissa päätöspuissa käytetään X_i:tä selittävinä muuttujina ja Y_i:tä selitettävänä muuttujana. Tässä menetelmässä hoitomuuttujaa W ei siis käytetä mallien selittävänä muuttujana. Odotusarvot $\hat{\mu}(1, x)$ ja $\hat{\mu}(1, x)$ estimoidaan

käyttämällä näitä eri päätöspuita ja $\tau(x)$ voidaan jälleen estimoida näiden kahden odotusarvon erotuksena:

$$\hat{\tau}(x) = \hat{\mu}(1, x) - \hat{\mu}(0, x). \tag{13}$$

Kolmas menetelmä, jonka Athey ja Imbens esittävät on menetelmä, jossa vasteelle suoritetaan muunnos, mikä mahdollistaa hoitovaikutuksen estimoimisen suoraan, ilman odotusarvojen estimoimista. Tätä algoritmia kutsutaan muunnetun vasteen päätöspuu-menetelmäksi (Transformed outcome tree method). Vaste muunnetaan seuraavalla tavalla:

$$Y_i^* = Y_i^{obs} \cdot \frac{W_i - e(X_i)}{e(X_i) \cdot (1 - e(X_i))}, \tag{14}$$

missä $e(x) = P(W_i = 1 | X_i = x)$ eli niin sanottu propensity score. Kun Y_i^{obs} sijoitetaan yllä olevaan kaavaan, saadaan:

$$Y_i^* = Y_i(1) \cdot \frac{W_i}{e(X_i)} - Y_i(0) \cdot \frac{1 - W_i}{1 - e(X_i)}. \tag{15}$$

Kun kyseessä on täysin satunnaistettu tilanne, eli hoito on määräytynyt täysin satunnaisesti kohteille, propensity score on vakio p ja muunnos yksinkertaistuu muotoon $Y_i^* = Y_i^{obs}/p$ hoidetuille havainnoille ja $Y_i^* = -Y_i^{obs}/(1 - p)$ kontrollihavainnoille. Tästä seuraa että:

$$E[Y_i^* | X_i = x] = \tau(x). \tag{16}$$

Edelleen seuraa se, että hoitovaikutus voidaan estimoida suoraan millä tahansa koneoppimismenetelmällä, ilman että algoritmiin tarvitsee tehdä muutoksia. Päätöspuu kasvatetaan käyttämällä muunnettua vastetta Y_i^* selitettävänä muuttujana ja X_i:tä selittävinä muuttujina. Hoitomuuttuja jätetään pois selittävien muuttujien joukosta. Hoitovaikutus uudelle havainnolle x estimoidaan paikallistamalla

päätöspuun lehtisolmu, mihin x kuuluu, ja laskemalla muunnetun vasteen Y_i^* keskiarvo kyseisessä lehtisolmussa.

Neljäs menetelmä, jonka Athey ja Imbens esittävät on kausaalisen puun menetelmä (causal tree method). Kausaalisen puun menetelmä muistuttaa hieman edellistä muunnetun vasteen menetelmää. Kausaalisen puun menetelmässä kasvatetaan ensin päätöspuu, missä X_i ovat selittäviä muuttujia ja Y_i on selitettävä Hoitovaikutus osapopulaatiolle x estimoidaan seuraavalla tavalla. Ensin paikallistetaan lehtisolmu, johon x kuuluu. Hoitovaikutus estimoidaan laskemalla tähän lehtisolmuun kuuluvien opetusaineiston havaintojen vasteen Y_i keskiarvo niille havainnoille missä $W = 1$ sekä niiden havaintojen vasteen Y_i keskiarvo missä $W = 0$. Hoitovaikutuksen estimaatti saadaan näiden keskiarvojen erotuksena. (Athey & Imbens, 2015)

Wager ja Athey (2018) esittävät, että kausaalisista puista voidaan kasvattaa kausaalinen metsä. Kausaalinen metsä perustuu Breimanin satunnainen metsä-algoritmiin ja sen toimintaperiaate on hyvin samankaltainen. Kun kausaalinen metsä menetelmää käytetään hoitovaikutuksen estimointiin, kasvatetaan useita kausaalisia puita edellä esitetyllä kausaalisen puun menetelmällä, ja lopullinen hoitovaikutus $\tau(x)$ saadaan näiden useiden kausaalisten puiden hoitovaikutusten keskiarvona.

Atheyn (2018) mukaan koneoppimismenetelmien laajamittainen käyttö taloustieteellisessä tutkimuksessa vaatii, että niitä muokataan paremmin sopivaksi kausaalisuhteiden estimoimiseen. Perinteisessä koneoppimiskirjallisuudessa kausaalisuutta ei ole juuri käsitelty, vaan algoritmeja on kehitetty puhtaasti ennustamistarkoitukseen. Athey kuitenkin argumentoi, että koneoppimismenetelmiä ja ekonometrian menetelmiä voidaan yhdistellä siten, että kausaalisuhteiden estimoiminen on mahdollista. Ideana on siis kehittää menetelmiä, missä yhdistetään koneoppimismenetelmien hyvät puolet ja mahdollisuus estimoida kausaalivaikutuksia. Ja kuten edellä on nähty, tällaisia menetelmiä kausaalisuhteiden estimoimiseen koneoppimismenetelmillä on jo olemassa. (Athey, 2018.)

3.4 Big data

Aineistoa ihmisten toiminnasta ja taloudellisista aktiviteeteista kerätään nykyään esimerkiksi erilaisten elektronisten laitteiden avulla valtavia määriä. Tällaisia aineistoja on alettu hyödyntämään myös taloustieteellisessä tutkimuksessa. Usein näissä aineistoissa on suuri määrä havaintoja, muuttujia sekä havaintojen välinen frekvenssi on tiheä. Tällaisista aineistoista käytetään englanninkielistä termiä big data. Big dataa on siis alettu hyödyntämään viime vuosina tutkimuskäytössä. Yksi esimerkki big datan hyödyntämisestä on MIT:n ”billion prices project”. ”Billion prices project”:issa kerätään internetistä satojen tuhansien tuotteiden hinnat päivittäin, ja niiden avulla muodostetaan inflaatiota kuvaava päivittäinen hintaindeksi. Tämän hintaindeksin etuna on se, että se muodostetaan käytännössä reaaliaikaisesti, kun taas virallinen kuluttajahintaindeksi laaditaan usean viikon viiveellä. Lisäksi, tällä menetelmällä voidaan mitata inflaatiota maissa, joissa viralliset tilastot eivät ole korruption takia luotettavia. Internet-aineistoja on myös hyödynnetty esimerkiksi ennustamalla työttömyyttä google-hakujen avulla ja verkkokauppa Ebayn keräämää dataa on käytetty erilaisten hinnoittelustrategioiden ja verotuksen vaikutuksen tutkimiseen internet huutokaupoissa. Big datan analysointi ei kuitenkaan välttämättä onnistu perinteisillä ekonometrian menetelmillä. Suuri muuttujien määrä voi esimerkiksi aiheuttaa ongelmia perinteisiä regressiomalleja käytettäessä. (Varian 2014; Mullainathan, 2017.; Einav & Levin, 2014.)

Koneoppimismenetelmiä käytetään lukuisissa kuvantunnistusta hyödyntävissä sovelluksissa. Kun kuvat muutetaan algoritmien ymmärtämään numeeriseen muotoon, aineiston muuttujien määrä voi jopa ylittää aineiston havaintojen määrän. Tällaisissa tilanteissa perinteisten regressiomallien käyttö on mahdotonta, mutta koneoppimismenetelmät kykenevät käsittelemään myös tällaisia aineistoja. Kun muuttujien määrä on suuri, malliin valittavien muuttujien valinta on myös vaikeaa. Koneoppimismenetelmissä tutkijan ei tarvitse itse valita malliin sisältyviä muuttujia, mikä helpottaa big datan analysointia. Yleisesti voidaan sanoa, että koneoppimismenetelmät toimivat jopa sitä paremmin, mitä enemmän aineistossa on havaintoja ja muuttujia. Big datassa on myös usein muuttujien välillä kompleksisia yhteyksiä, joiden mallintamiseen tarvitaan perinteisiä regressiomalleja joustavampia menetelmiä. Koneoppimismenetelmät soveltuvat lisäksi esimerkiksi tekstiaineistojen

analysointiin. Koneoppimismenetelmät mahdollistavat siis uuden tyyppisten aineistojen hyväksikäytön taloustieteellisessä tutkimuksessa ja big datan avulla taloustiede voisi mennä yhä enemmän empiiriseen suuntaan. (Varian 2014; Mullainathan, 2017.; Einav & Levin, 2014.)

3.5 Aiemmat tutkimukset

Koneoppimismenetelmiä on hyödynnetty vielä kohtalaisen vähän tutkimuskäytössä. Tässä osiossa käydään läpi joitain viime vuosina tehtyjä tutkimuksia, joissa koneoppimismenetelmiä on käytetty. Tarkoituksena tässä alaluvussa on havainnollistaa, minkälaisissa tutkimuksissa koneoppimismenetelmiä on käytetty, ja millaisia tuloksia on saatu. Useimmissa tutkimuksissa on pyritty ennustamaan jotain muuttujaa koneoppimismenetelmien avulla sekä perinteisten ekonometrian menetelmien avulla ja vertailtu koneoppimismenetelmien ja perinteisten menetelmien ennustamistarkkuutta. Jean ym. (2016) käyttivät neuroverkko-algoritmia (Artificial neural network) kotitalouksien käytettävissä olevien tulojen arvioimiseen satelliittikuvista. Kang, Kuznetsova, Luca, ja Choi (2013) käyttivät Yelp.com-sivuston ravintola-arvosteluja ja tukivektorikonetta ennustamaan terveystarkastusten lopputuloksia. Antweiler ja Frank (2004) käyttivät internetin keskustelupalstojen viestejä ja koneoppimisalgoritmia osakamarkkinoiden volatiliteetin ennustamiseen. Tuloksissaan he toteavat, että keskustelupalstojen viesteistä saatu informaatio helpottaa volatiliteetin ennustamisessa. Vomfell, Härdle ja Lessamann (2018) käyttivät koneoppimismenetelmiä ja twitter-dataa rikollisuuden ennustamiseen New Yorkin kaupungissa. Verrattaessa koneoppimismenetelmiä lineaarisiin malleihin koneoppimismallit ennustivat paremmin rikollisuuden määrää.

Koneoppimismenetelmiä voidaan myös käyttää päätöksenteon apuvälineenä ja politiikkasuositusten tekemiseen. Kleinberg, Lakkaraju, Leskovec, Ludwig ja Mullainathan (2017) esittävät että koneoppimista voitaisiin käyttää päätettäessä, pitäisikö rikollinen pitää vangittuna oikeudenkäyntiä odotettaessa vai päästää vapaaksi takuita vastaan. Tutkimuksen mukaan, jos näitä päätöksiä tehtäessä hyödynnettäisiin koneoppimismenetelmiä, rikollisuutta voitaisiin vähentää ilman että vangittujen rikollisten määrää tarvitsisi nostaa. Andini, Ciani, Blasio, D'Ognazio ja Salvestrini (2018) osoittivat että koneoppimismenetelmiä voidaan käyttää

päätöksenteon tehostamisessa. Tutkimuksessaan he tarkastelevat Italiassa vuonna 2014 suoritettua verojen huojennusta, jonka tavoitteena oli lisätä kotitalouksien kulutusta. Tutkimuksen lopputuloksissa todetaan, että mikäli veronalennusten kohdistamisessa olisi käytetty päätöspuuta, sen tehokkuus olisi ollut huomattavasti parempi ja veronalennus olisi voitu kohdistaa kotitalouksille, joita verojen alennus olisi hyödyttänyt eniten.

Koneoppimismenetelmiä on käytetty kohtalaisen vähän tutkimuksissa, mutta tutkimuksia joissa kyseisiä menetelmiä hyödynnetään, julkaistaan yhä enemmän. Ohjatun oppimisen menetelmiä on viime vuosina hyödynnetty esimerkiksi konkurssien, kysynnän ja lukuisten muiden ilmiöiden tutkimiseen.

Gogas, Papadimitriou ja Agrapetidou (2018) käyttivät tukivektorikonetta yhdysvaltalaisten pankkien konkurssien ennustamiseen. Tukivektorikoneen ennustamistarkkuus oli 99,22% ja se päihitti perinteisemmät konkurssien ennustamiseen käytetyt menetelmät. Myös Le ja Viviani (2018) käyttivät koneoppimismenetelmiä pankkien konkurssien ennustamiseen. Useita koneoppimismenetelmiä (neuroverkko, tukivektorikone ja k:n lähimmän naapurin menetelmä) verrattiin perinteisiin ekonometrian menetelmiin kuten logistiseen regressioon. Parhaaseen ennustamistarkkuuteen päästiin neuroverkkoa käyttämällä. Behr ja Weinblat (2017) käyttivät satunnainen metsä-algoritmia seitsemän eurooppalaisen maan yritysten konkurssien ennustamiseen kirjanpitoaineiston perusteella. Satunnainen metsä-algoritmin avulla pystyttiin selvittämään konkurssia ennakoivia tekijöitä. Tanaka, Kinkyo ja Hamori (2016) käyttivät satunnainen metsä-algoritmia pankkien konkurssien ennustamiseen. Satunnainen metsä ennusti konkursseja huomattavasti paremmalla tarkkuudella kuin logistinen regressio.

Claveria ja Torra (2016) käyttivät tukivektorikonetta ja neuroverkkoa kysynnän ennustamiseen espanjalaisissa turistikohteissa. Koneoppimismallien ennustamistarkkuutta verrattiin lineaaristen mallien ennustamistarkkuuteen. Vertailluista menetelmistä tukivektorikone ennusti parhaiten. Koneoppimismenetelmät olivat erityisen hyviä pitkän aikavälin kysyntää ennustettaessa, lineaarisiin malleihin verrattuna. Bajari, Nekipelov, Ryan ja Yang (2015) käyttävät koneoppimismenetelmiä ruokakauppojen kysynnän ennustamiseen.

Koneoppimismenetelmät ennustivat ruokakauppojen kysyntää lineaarisia regressiomalleja paremmin. Parhaiten kysyntää ennusti satunnainen metsä-algoritmi. Muchlinski, Siroky, He ja Kocher (2016) vertasivat satunnaista metsää logistiseen regressioon sisällissodan syttymisen ennustamisessa. Satunnainen metsä oli huomattavasti parempi ennustamaan sisällissodan syttymistä kuin logistinen regressio. Akay, Astar ja Topal (2018) käyttivät koneoppimismenetelmiä koulutuksen ja palkkatason välisen yhteyden selvittämiseen. Koneoppimismenetelmistä tukivektorikone oli paras palkkatason ennustamiseen ja se oli myös parempi kuin lineaarinen regressiomalli. Sohnesen ja Stender (2016) käyttivät satunnainen metsä-algoritmia köyhyyden ennustamiseen kuudessa eri maassa. Satunnainen metsä ennusti köyhyyttä paremmin kuin lineaariset mallit.

Ohjatun oppimisen menetelmiä voidaan siis hyödyntää moniin taloustieteilijöitä askarruttaviin kysymyksiin, mutta niitä voidaan käyttää myös uusien aineistojen analysointiin ja aivan uuden tyyppisten tutkimuskysymysten tutkimiseen. Esimerkiksi sosiaalisen median aineistoja, kuva- ja tekstiaineistoja voitaisiin mahdollisesti hyödyntää taloustieteellisessä tutkimuksessa koneoppimismenetelmien avulla. Mullhainathan ja Spiess (2017) nimittävät ongelmia, joiden ratkaisemiseksi voidaan käyttää koneoppimista, $\hat{y}$-ongelmiksi, ja perinteisiä ekonometrian ongelmia $\hat{\beta}$-ongelmiksi. Useat koneoppimismenetelmät eivät tuota minkäänlaisia parametreja ja vaikka koneoppimismenetelmä tuottaisikin parametreja, ne eivät ole vakaita. Jotta koneoppimista voidaan hyödyntää, täytyy löytää sopivia $\hat{y}$-ongelmia. Kuten nähdään, koneoppimismenetelmiä on jo käytetty useissa tutkimuksissa menestyksekkäästi. Useimmissa tutkimuksissa koneoppimismenetelmät ennustavat paremmin kuin perinteiset ekonometriset mallit.

4 AINEISTO JA TUTKIMUSMENETELMÄT

Tutkielman empiirisessä osuudessa demonstroidaan koneoppimismenetelmien käyttöä käytännössä, sekä vertaillaan koneoppimismenetelmiä ekonometrian perinteisiin menetelmiin. Empiirisen osuuden aineistona käytetään portugalilaisen pankin puhelinmarkkinointikampanjasta kerättyä aineistoa. Aineisto on kerätty toukokuun 2008 ja kesäkuun 2013 välisenä aikana. Pankin markkinointikampanjan tarkoituksena on ollut saada myytyä asiakkaille pitkäaikaisia talletuksia. Aineiston selitettävä muuttuja on binäärinen muuttuja, joka kertoo, onko asiakas suostunut tekemään pitkäaikaisen talletuksen vai ei. Selittäviä muuttujia on yhteensä 20 kappaletta. Selittäviä muuttujia on neljää tyyppiä: asiakkaan ominaisuudet, yhteydenoton ominaisuudet, talouden tilasta kertovat tekijät ja muut muuttujat. Asiakkaan ominaisuuksista kertovia muuttujia ovat esimerkiksi asiakkaan ikä, siviilisääty ja koulutustaso. Kyseessä olevaan yhteydenottoon liittyviä ominaisuuksia ovat: yhteydenottotapa, yhteydenottokuukausi, viikonpäivä ja puhelun kesto. Talouden tilasta kertovia muuttujia ovat esimerkiksi kolmen kuukauden euribor korko ja kuluttajahintaindeksi. Muita selittäviä muuttujia ovat esimerkiksi edellisen markkinointikampanjan lopputulos ja kuinka monta kertaa asiakkaaseen on otettu aikaisemmin yhteyttä. Kaikki aineiston muuttujat ja niiden selitteet on listattu taulukossa 3. Aineistossa on yhteensä 41188 havaintoa.

Taulukko 3. Tutkimusaineiston muuttujat.

nro	Muuttuja	Selite	Tyyppi
1	age	Asiakkaan ikä	Numeerinen
2	job	Asiakkaan työn tyyppi	Kategorinen
3	marital	Siviilisääty	Kategorinen
4	education	Koulutustaso	Kategorinen
5	default	Onko asiakkaalla maksuhäiriömerkintöjä	Kategorinen
6	housing	Onko asiakkaalla asuntolainaa	Kategorinen
7	loan	Onko asiakkaalla muuta lainaa	Kategorinen
8	contact	Yhteydenottotapa (cellular/telephone)	Kategorinen
9	month	Yhteydenottokuukausi	Kategorinen
10	day_of_week	Viikonpäivä	Kategorinen
11	duration	Puhelun kesto	Numeerinen
12	campaign	Kuinka monta kertaa asiakkaaseen on oltu yhteydessä tässä kampanjassa	Numeerinen
13	pdays	Kuinka monta päivää	Numeerinen

		edellisestä yhteydenotosta tässä kampanjassa	
14	previous	Kuinka monta kertaa asiakkaaseen on oltu yhteydessä edellisessä kampanjassa	Numeerinen
15	poutcome	Edellisen kampanjan lopputulos	Kategorinen
16	emp.var.rate	Työvoiman variaatio (neljännesvuosittainen indikaattori)	Numeerinen
17	cons.price.idx	Kuluttajahintaindeksi	Numeerinen
18	cons.conf.idx	Kuluttajaluottamusindeksi	Numeerinen
19	euribor3m	3kk euribor korko	Numeerinen
20	nr.employed	Työntekijöiden lkm (neljännesvuosittainen indikaattori)	Numeerinen
21	y	Syntyikö kauppa vai ei	Kategorinen

Aineisto on hankittu UCI Machine learning repository -sivustolta (Dua & Graff, 2019). UCI Machine learning repository on sivusto, jonne on kerätty aineistoja, joita voidaan käyttää koneoppimisalgoritmien testaamiseen. Suuri osa sivuston aineistoista on tutkijoiden sinne luovuttamia. Tässä tutkimuksessa käytettyä aineistoa ovat alun perin käyttäneet Moro, Cortez ja Rita (2014).

4.1 Tutkimusmenetelmät

Tutkimuksessa pyritään vertailemaan koneoppimismenetelmiä perinteisiin ekonometrian menetelmiin. Tutkimuksessa käytetään kaikkia teoria-osuudessa käsiteltyjä koneoppimismenetelmiä soveltuvin osin. Koska selitettävä muuttuja on binäärinen, lasso- ja ridge-regressioita ei voida käyttää tutkimuksessa. Koneoppimismenetelmien avulla on tarkoitus luoda malli, joka ennustaa mahdollisimman hyvin suostuuko asiakas tekemään pitkäaikaistalletuksen vai ei. Parhaan mallin tuottamaa informaatiota pyritään tulkitsemaan ja selvittämään mitkä ovat tärkeimpiä tekijöitä, jotka vaikuttavat binääriseen vastemuuttujaan.

Koska vastemuuttuja on binäärinen, vertailtavana mallina käytetään logit-mallia. Logit-mallia käytetään ekonometriassa hyvin yleisesti binäärisen vastemuuttujan yhteydessä. Logit-mallia arvioidaan muuttujien merkitsevyystasojen eli p-arvojen sekä regressiokertoimien perusteella. Logit-malli voidaan esittää yleisessä muodossa binääriselle muuttujalle Y, seuraavalla tavalla:

$$\begin{aligned}\Pr(Y = 1)\,|X_1, X_2, \ldots, X_k) &= F(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k) \\ &= \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k)}}\end{aligned} \tag{17}$$

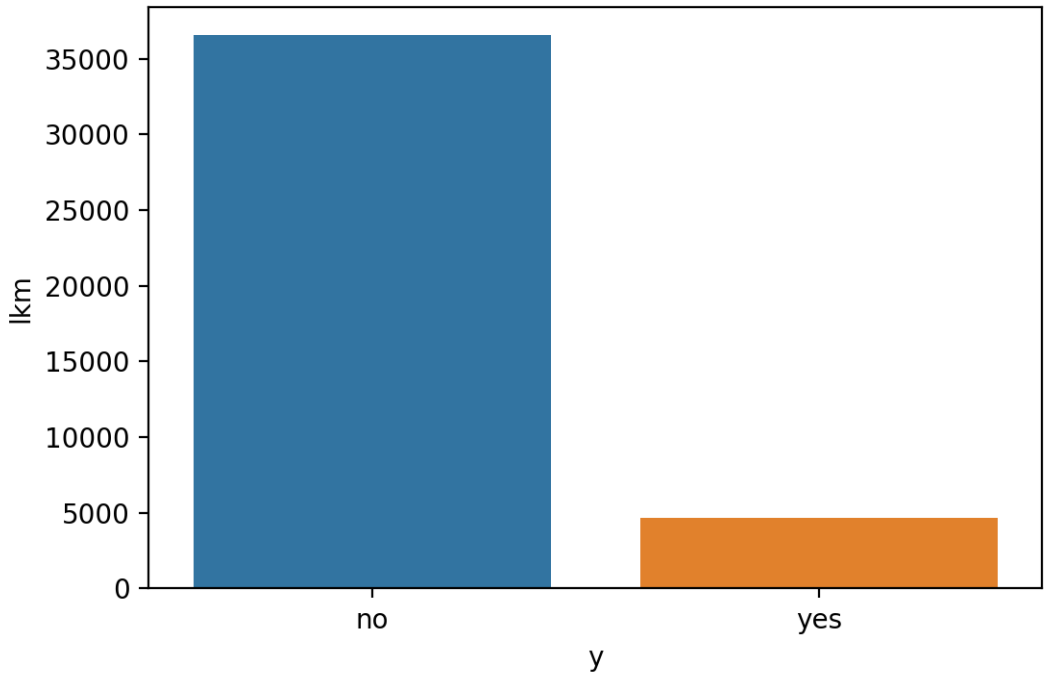
(Stock & Watson, 2012.)

4.2 Käytetyt työkalut

Aineiston analysointi ja mallien sovittaminen suoritettiin Python-ohjelmointikielellä. Koneoppimismallien sovittamiseen käytettiin Pythonin scikit-learn kirjastoa (Pedregosa ym., 2011). Useita muitakin kirjastoja käytettiin tutkimuksen eri vaiheissa, nämä kirjastot on listattu liitteisiin. Python ohjelmointi suoritettiin Jupyter notebook -ohjelmistoa käyttäen.

5 TUTKIMUKSEN TULOKSET

Tutkimuksen tarkoituksena on demonstroida koneoppimismenetelmien käyttöä ja vertailla koneoppimismenetelmiä logistiseen regressioon. Ennen varsinaisen analyysin aloittamista on usein suotavaa tarkastella analysoitavaa aineistoa tarkemmin esimerkiksi kuvaajien avulla. Kuviossa 11 on kuvattu aineiston muuttujan y luokkajakaumaa. Kuten kuviosta nähdään, aineiston jakauma on erittäin epätasainen. Vain noin 11% asiakkaista on tehnyt pitkäaikaistalletuksen. Aineiston epätasapainosta johtuen analyysissä päätettiin käyttää luvussa 3.2 kuvattua AUC tunnuslukua pääasiallisena mittarina mallien vertailuun. Tarkkuutta ei yleisesti ottaen pidetä hyvänä tunnuslukuna epätasapainoisen aineiston ollessa kyseessä. Tarkkuus-tunnusluku kuitenkin pidettiin mukana analyysissä, sekundäärisenä tunnuslukuna.



Kuvio 11. Selitettävän muuttujan y luokkajakauma.

Aineiston epätasapainosta johtuen päätettiin myös kokeilla alinäytteistämistä (undersampling) ja ylinäytteistämistä (oversampling). Ali- ja ylinäytteistäminen ovat koneoppimisessa yleisesti käytettäviä menetelmiä epätasapainoisten aineistojen ollessa kyseessä. Epätasapainoinen aineisto voi johtaa siihen, että enemmistöluokka dominoi luokittelua. Ali- ja ylinäytteistämisellä pyritään siihen, että luokittelijan

avulla saataisiin paremmin informaatiota vähemmistöluokkaan kuuluvista havainnoista, koska ne ovat yleensä juuri mielenkiinnon kohteena. Näissä menetelmissä luodaan uudet aineistot, joissa luokat ovat tasapainossa. Alinäytteistämisessä enemmistöluokkaan kuuluvista havainnoista otetaan satunnaisesti näytteitä yhtä paljon kuin aineistossa on vähemmistöluokkaan kuuluvia havaintoja. Ylinäytteistämisessä taas vähemmistöluokkaan kuuluvia havaintoja monistetaan niin paljon, että luokat ovat tasapainossa. Alinäytteistämisen ongelmana on, että menetetään suuri määrä informaatiota, kun koko aineistoa ei käytetä mallin opettamisessa. Ylinäytteistäminen taas voi helposti johtaa ylisovittamiseen. (Chawla, 2009, s. 879-883)

Ennen koneoppimismallien ja logistisen regression sovittamista aineiston muuttujia jouduttiin muokkaamaan. Kaikki kategoriset muuttujat täytyi muuttaa numeeriseen muotoon. Osa kategorisista ovat sellaisia, että niillä voidaan sanoa olevan jokin luonnollinen järjestys. Tällaiset muuttujat, kuten esimerkiksi koulutustasoa kuvaava muuttuja education, muutettiin järjestysasteikollisiksi muuttujiksi, missä jokainen eri luokka sai oman numeroarvon. Kategoriset muuttujat, joilla ei nähty olevan mitään loogista järjestystä, kuten esimerkiksi month ja day_of_week, muunnettiin dummy-muuttujiksi. Kun muuttuja muutetaan dummy-muuttujiksi, niin jokaisesta muuttujan arvosta tehdään uusi muuttuja, joka voi saada joko arvon nolla tai yksi. Eli esimerkiksi day_of_week muuttuja muutettiin viideksi eri viikonpäivää kuvaavaksi binääriseksi muuttujaksi. Dummy-muuttujia luodessa, dummy-muuttujia täytyy kuitenkin luoda aina yksi vähemmän kuin alkuperäisessä muuttujassa on vaihtoehtoja. Näin ollen esimerkiksi viikonpäivää kuvaavista dummy-muuttujista yksi pudotettiin pois, ja jäljelle jäi 4 viikonpäivää kuvaavaa dummy-muuttujaa. Joidenkin kategoristen muuttujien arvot puuttuivat osasta havaintoja. Puuttuvat arvot korvataan joskus mediaanilla tai moodilla, tai sellaiset havainnot, joissa on puuttuvia arvoja, voidaan poistaa aineistosta. Tässä tutkimuksessa päätettiin pitää puuttuvat arvot aineistossa, ja käsitellä niitä kuten ne olisivat kategorisen muuttujan yksi kategoria.

Koska jotkut koneoppimisalgoritmit perustuvat etäisyyksille, numeeriset muuttujat päätettiin normalisoida. Normalisointi tehtiin siten, että kaikkien numeeristen muuttujien arvot skaalattiin välille 0-1. Numeerisista muuttujista muuttuja duration

jätettiin kokonaan pois analyysista. Tämä muuttuja kertoo puhelun keston, mutta koska puhelun kestoa ei voida tietää etukäteen sitä ei haluttu ottaa mukaan analyysiin. Tarkoituksena on siis pyrkiä ennustamaan myyntipuhelun lopputulosta käyttäen muuttujia, jotka ovat käytössä ennen myyntipuhelua. Numeerinen muuttuja pdays päätettiin muuttaa binääriseksi muuttujaksi. Tämä muuttuja kertoo, kuinka monta päivää on kulunut siitä, kun asiakkaaseen on oltu yhteydessä edellisen kampanjan tiimoilta. Jos asiakkaaseen ei ollut oltu yhteydessä edellisen kampanjan tiimoilta tämä muuttuja oli koodattu arvolla 999. Koska aineistossa oli vain vähän havaintoja, joissa asiakkaaseen oli oltu aikaisemman kampanjan tiimoilta yhteydessä, tämä muuttuja nähtiin parhaimmaksi muuttaa binääriseksi muuttujaksi.

Kun aineiston muuttujat oli saatu valmiiksi analyysiä varten, aineistossa selittävien muuttujien määrä oli 32 ja havaintojen määrä oli 41188. Seuraavaksi aineisto jaettiin kolmeen osaan: opetus-, validointi ja testausosaan. Aineisto jaettiin siten, että opetukseen käytettiin 50% aineistosta, validointiin 25% ja testaukseen 35%. Tässä siis päätettiin käyttää hyperparametrien virittämiseen validointiaineistoa eikä ristiinvalidointia. Kun aineisto oli jaettu, opetusosaan sovitettiin logistinen regressio. Logistisen regression tulokset on esitetty taulukossa 4.

Taulukko 4. Logistisen regressiomallin tulokset.

	Selitettävä muuttuja
Selittävä muuttuja	Kaupan toteutuminen(y)
Vakiotermi	-2.3331***
age	0.2475
campaign	-2.8784***
previous	-0.4677
emp.var.rate	-7.4622***
cons.price.idx	5.3173***
cons.conf.idx	0.5883**
euribor3m	1.3964*
nr.employed	1.5107
job	0.0051
education	0.0295**
default	-0.2613***
housing	-0.0189
loan	0.0222
contact	-0.8186***
poutcome	0.4541***
divorced	0.3608
married	0.3223
single	0.4023
apr	-0.2566
aug	0.1614
dec	0.0166

jul	-0.1820
jun	-0.8443***
mar	1.1970***
may	-0.6727***
nov	-0.7742***
oct	-0.1647
fri	-0.1156
mon	-0.3377***
thu	0.0012
tue	-0.0487
previous_campaign	0.9661***
N	20595
pseudo R^2	0.213

N: havaintojen lukumäärä
pseudo R^2: determinaatiokerroin
*** Tilastollisesti merkitsevä 1 %:n merkitsevyystasolla.
** Tilastollisesti merkitsevä 5 %:n merkitsevyystasolla.
* Tilastollisesti merkitsevä 10 %:n merkitsevyystasolla.

Logistisen regression tuloksista nähdään, että useat muuttujat eivät ole tilastollisesti merkitseviä. Regressiokertoimien tulkinta logistisessa regressiossa ei ole yhtä helppoa kuin esimerkiksi lineaarisen regression tapauksessa. Käytännössä regressiokertoimia voidaan tulkita lähinnä etumerkkien perusteella. Negatiivinen regressiokerroin tarkoittaa, että muuttuja vaikuttaa negatiivisesti kaupan syntymisen todennäköisyyteen ja positiivinen regressiokerroin vaikuttaa positiivisesti. Taulukosta 4 nähdään esimerkiksi, että kuluttajahintaindeksiä kuvaavan muuttujan cons.price.idx regressiokerroin on positiivinen, eli kaupan syntyminen on todennäköisempää korkeammilla kuluttajahintaindeksin arvoilla.

Seuraavaksi opetusaineistoon sovitettiin useita eri koneoppimismalleja vakioasetuksilla. Tarkoituksena tässä vaiheessa oli testata useita eri menetelmiä, joista valitaan 1-2, joiden hyperparametrit sitten viritetään. Näin toimittiin, koska usein ei voida etukäteen tietää mikä algoritmeista tuottaa parhaimman tuloksen. Käytetyt algoritmit olivat k-lähimmän naapurin menetelmä (KNN), päätöspuu (Päätöspuu), satunnainen metsä (RF), lineaarinen tukivektorikone (SVC Linear), tukivektorikone toisen asteen kernel-funktiolla (SVC Poly2), tukivektorikone kolmannen asteen kernel-funktiolla (SVC Poly3) sekä gradientti tehostaminen (GT). Kaikille sovitetuille malleille, sekä logistiselle regressiolle laskettiin opetusaineiston tarkkuus, opetusaineiston AUC, validointiaineiston tarkkuus ja validointiaineiston AUC. Edellä mainitut tunnusluvut laskettiin myös malleille, jotka sovitettiin alinäytteistettyyn ja ylinäytteistettyyn aineistoon. Vakioasetuksilla sovitettujen mallien tulokset on esitetty taulukossa 5.

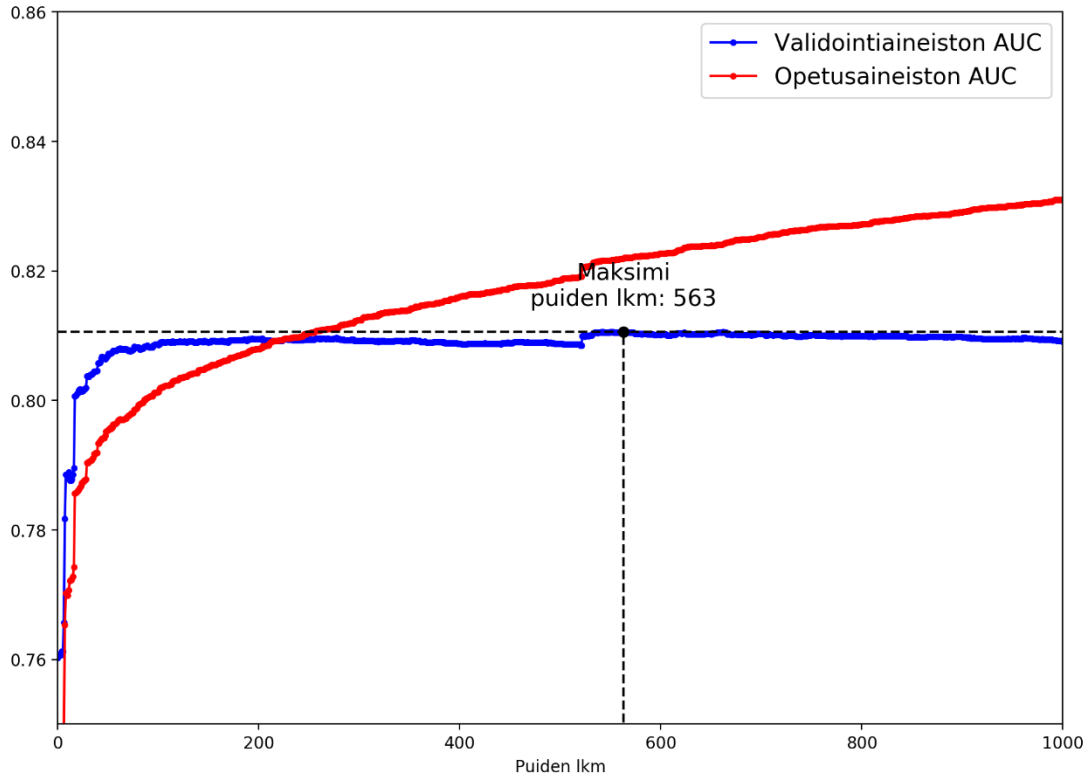
Taulukko 5. Vakioasetuksilla sovitettujen mallien tulokset.

Malli	Opetusaineiston tarkkuus	Opetusaineiston AUC	Validointiaineiston tarkkuus	Validointiaineiston AUC
	Paneeli A: Alkuperäinen aineisto			
Logit	0.9007	0.7929	0.8990	0.8055
KNN	0.9089	0.9205	0.8877	0.7044
Päätöspuu	0.9966	0.9999	0.8365	0.6316
RF	0.9831	0.9989	0.8892	0.7585
SVC Linear	0.8980	0.7055	0.8970	0.7094
SVC Poly2	0.8984	0.7014	0.8970	0.7104
SVC Poly3	0.8982	0.7429	0.8953	0.7151
GT	0.9076	0.8100	**0.9024**	**0.8093**
	Paneeli B: Alinäytteistetty aineisto			
Logit	0.7376	0.7922	0.8337	**0.8036**
KNN	0.7860	0.8665	0.6971	0.7297
Päätöspuu	0.9978	1.00	0.6386	0.6529
RF	0.9797	0.9986	0.7739	0.7745
SVC Linear	0.7310	0.7677	0.8361	0.7914
SVC Poly2	0.7313	0.7832	0.7976	0.7918
SVC Poly3	0.7326	0.7950	**0.8463**	0.7918
GT	0.7565	0.8325	0.8281	0.8010
	Paneeli C: Ylinäytteistetty aineisto			
Logit	0.7403	0.7953	0.8253	0.8072
KNN	0.9234	0.9968	0.7639	0.6924
Päätöspuu	0.9978	1.00	0.8417	0.6132
RF	0.9972	0.9999	**0.8835**	0.7673
SVC Linear	0.7339	0.7726	0.8361	0.7888
SVC Poly2	0.7394	0.7953	0.8254	0.7957
SVC Poly3	0.7471	0.8110	0.8382	0.7911
GT	0.7562	0.8298	0.8326	**0.8128**

Tuloksista nähdään miten esimerkiksi päätöspuu ylisovittaa huomattavasti. Sen opetuaineiston tarkkuus ja AUC ovet todella hyviä, mutta validointiaineiston tunnusluvut ovat huonoja. Hieman yllättäen myös satunnainen metsä näyttäisi kärsivän ylisovittamisesta. Usein satunnaisen metsän vahvuuksina pidetään juuri sitä, että se ei ylisovita kovin herkästi. Ali- ja ylinäytteistäminen eivät näyttäneet tuovan huomattavaa parannusta mallien ennustamistarkkuuteen. Ainoastaan tukivektorikoneiden kohdalla AUC-arvot paranivat merkittävästi ali- ja ylinäytteistysten avulla. Tästä johtuen päätettiin lopullisten mallien luomisessa käyttää alkuperäistä aineistoa. Koska gradientti tehostaminen tuotti parhaat tarkkuus- ja AUC-arvot vakioasetuksilla, se valittiin menetelmäksi, jota käytetään lopullisen koneoppimismallin luomiseen, eli sen hyperparametreja päätettiin virittää ja tätä lopullista mallia sitten verrataan logistiseen regressioon. Lisäksi päätettiin virittää päätöspuun hyperparametreja. Päätöspuut eivät yleisesti ole yhtä hyviä ennustamaan kuin monimutkaisemmat menetelmät, mutta niiden tulkinta on monimutkaisempia

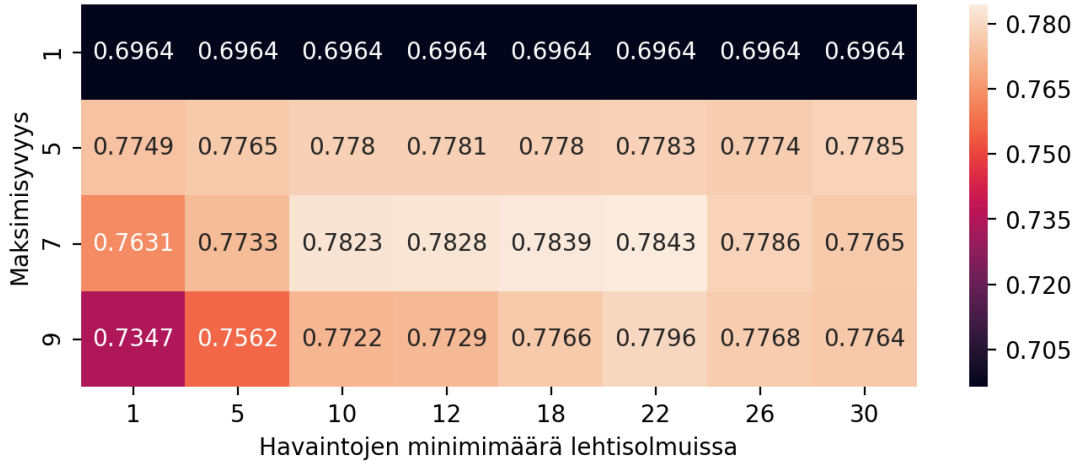
malleja helpompaa, joten niiden avulla voidaan saada lisäinformaatiota esimerkiksi siitä, mitkä ovat tärkeimpiä muuttujia.

Gradientti tehostamisessa valittavat hyperparametrit ovat päätöspuiden lukumäärä ja kutistustermi. Hyperparametrit viritettiin valitsemalla muutama eri arvo kutistustermille ja kaikkien näiden kutistustermien arvoja käyttäen sovitettiin malli, jossa on 1000 päätöspuuta. Näiden mallien joka iteraatiokierroksilla laskettiin validointiaineiston AUC ja opetusaineiston AUC. Optimaalinen kutistustermi ja päätöspuiden lukumäärä saatiin siitä kombinaatiosta missä validointiaineiston AUC oli korkein. Optimaaliseksi kutistustermiksi saatiin 0.1. Kuviossa 12 on kuvattu miten validointiaineiston AUC ja opetusaineiston AUC muuttuvat eri iteraatiokierroksilla. Validointiaineiston AUC saavutettiin, kun päätöspuiden lukumäärä oli 563, joten lopulliseksi malliksi saatiin malli, jonka kutistustermi on 0.1 ja päätöspuiden lukumäärä on 563. Kuviosta 12 nähdään myös, miten opetusaineiston AUC paranee yhä, vaikka validointiaineiston AUC on jo saavuttanut maksimin. Selvästikin siis malli alkaa ylisovittamaan, kun siitä tulee liian monimutkainen.



Kuvio 12. Gradientti tehostaminen -mallin hyperparametrien virittäminen

Päätöspuun hyperparametrien virittämiseen käytettiin edellisestä poiketen ristiinvalidiontia. Näin toimittiin, koska ristiinvalidointi oli päätöspuun tapauksessa helpompi ohjelmoida scikit-learn:n valmiita funktioita käyttäen. Yleisesti ottaen ristiinvalidoinnin avulla saadut tulokset voivat olla jopa parempia kuin validointiaineiston avulla saadut tulokset. Viritettäviksi hyperparametreiksi valittiin päätöspuun syvyys sekä havaintojen minimimäärä lehtisolmuissa. Maksimisyvyydelle valittiin 4 arvoa ja havaintojen minimimäärälle lehtisolmuissa valittiin 8 arvoa. Näin ollen saatiin 32 eri yhdistelmää näille hyperparametreille. Kaikille näille yhdistelmille laskettiin AUC arvo ristiinvalidoinnin avulla. Ristiinvalidoinnin tulokset ovat nähtävillä kuviossa 13. Paras AUC-arvo saatiin mallilla, jossa päätöspuun maksimisyvyys on 7 ja havaintojen minimimäärä lehtisolmuissa on 22.



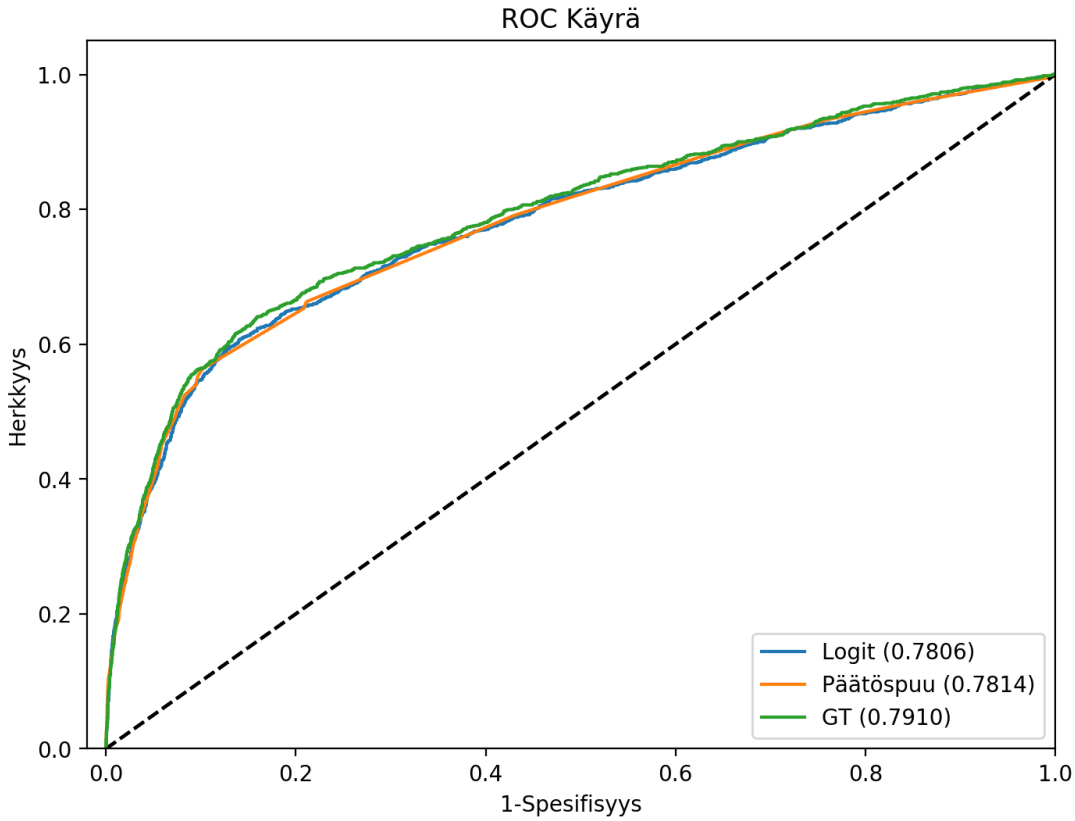
Kuvio 13. Päätöspuun hyperparametrien virittäminen.

Hyperparametrien virittämisen jälkeen voitiin laskea lopullisten mallien tarkkuus ja AUC-tunnuslukujen arvot testausaineistoa käyttäen. Gradientti tehostamis -mallin, päätöspuun ja logistisen regression tarkkuus ja AUC testiaineistoa käyttäen on esitetty taulukossa 6.

Taulukko 6. Testausaineiston tulokset.

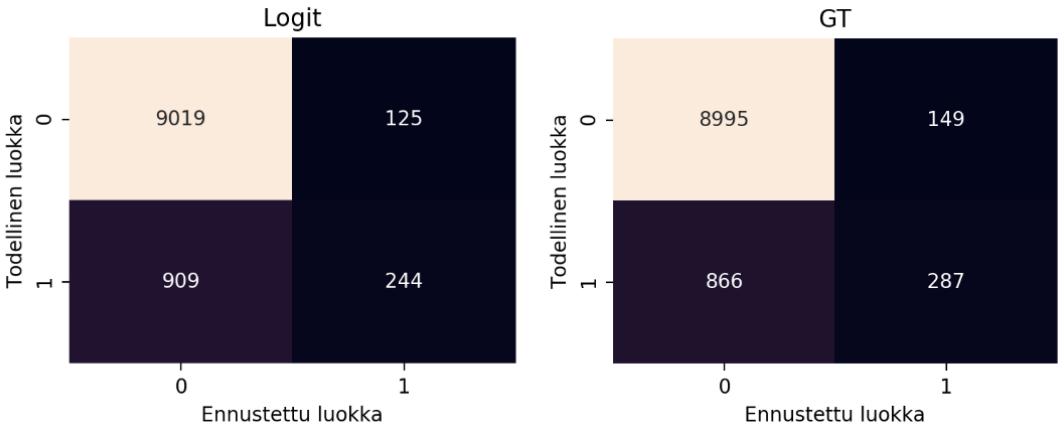
Malli	Testausaineiston tarkkuus	Testausaineiston AUC
Logit	0.8996	0.7806
Päätöspuu	0.8983	0.7814
GT	**0.9014**	**0.7910**

Sekä AUC:n että tarkkuuden perusteella gradientti tehostaminen oli näistä malleista paras. Erot eri mallien välillä ovat kuitenkin pieniä. Eri mallien suorituskykyä on edelleen havainnollistettu ROC-käyrien avulla kuviossa 14.



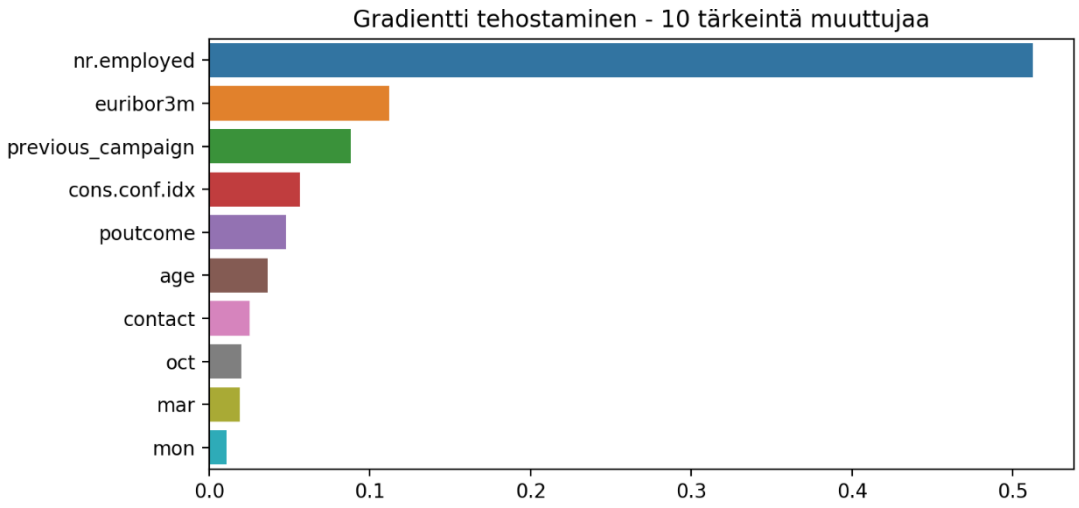
Kuvio 14. Lopullisten mallien ROC-käyrät.

Eri mallien ROC-käyrät ovat lähes identtiset. Näyttäisi siis siltä, että tässä tapauksessa koneoppimismenetelmät eivät ole ylivertaisia ennustamaan verrattuna perinteiseen logistiseen regressioon. Tämä voi johtua siitä, että käytetty aineisto oli suhteellisen pieni. Vaikka koneoppimismallit ovatkin yleensä parempia ennustamaan kuin perinteiset regressiomallit, ne tarvitsevat paljon havaintoja ennen kuin niiden vahvuudet tulevat kunnolla esille. Kuviossa 15 on kuvattu logistisen regressiomallin sekä GT-mallin luokittelumatriisit. GT-malli ennustaa paremmin positiiviseen luokkaan kuuluvia havaintoja, mutta myös sen väärien positiivisten määrä on suurempi kuin logistisessa regressiomallissa. Logistinen regressiomalli ennustaa paremmin negatiiviseen luokkaan kuuluvia havaintoja, mutta sen väärien negatiivisten määrä on suurempi kuin GT-mallissa. Usein ennustamisessa tärkeämpää on juuri oikeiden positiivisten löytäminen. Esimerkiksi tehtäessä lääketieteellistä diagnoosia, saatetaan mieluummin sietää suurempaa väärien positiivisten määrää, kunhan oikeita positiivisia löydetään mahdollisimman paljon.



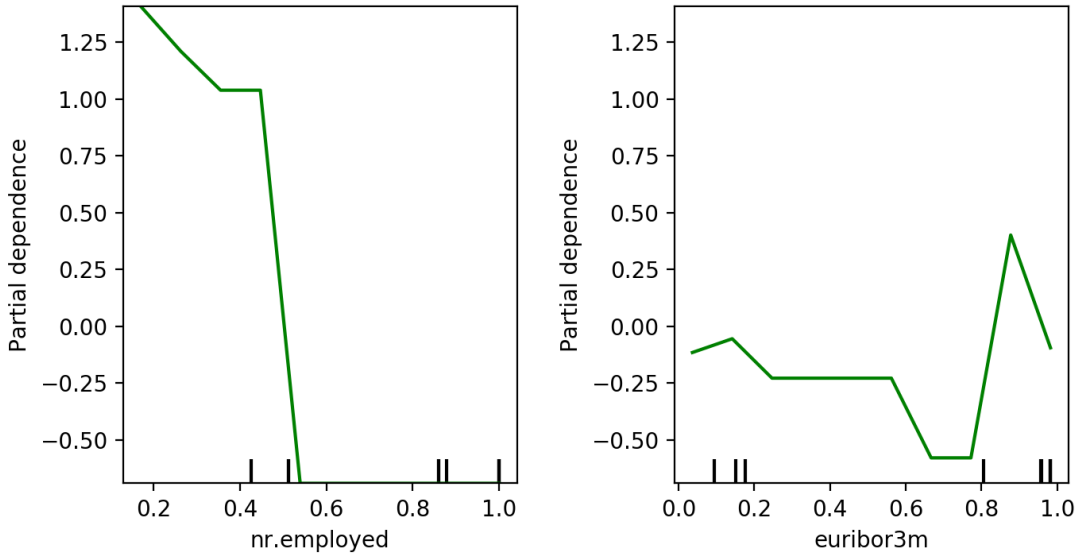
Kuvio 15. GT-mallin ja logit-mallin luokittelumatriisit.

Lopullisten tunnuslukujen laskemisen jälkeen siirryttiin mallien tulkintaan. GT-mallista selvitettiin tärkeimmät muuttujat, sekä piirrettiin luvussa 2.9 kuvattuja osittaisen riippuvuuden kuvaajia muutamille tärkeimmille muuttujille. Muuttujien tärkeys laskettiin käyttäen gini-epäpuhtauden keskimääräistä vähenemistä niissä päätöspuiden solmuissa, missä kutakin muuttujaa käytettiin aineiston jakamiseen. Suhteellisesti tärkeimmät muuttujat on esitetty kuviossa 16. GT-mallissa ylivoimaisesti tärkeimmäksi muuttujaksi nousee työllisyyttä kuvaava neljännesvuosittainen indikaattori nr.employed, sen suhteellinen tärkeys on yli 0.5. Seuraavaksi tärkein muuttuja on kolmen kuukauden euribor korko. Kolmanneksi tärkein muuttuja on binäärinen muuttuja previous_campaign, joka kertoo, onko asiakkaaseen oltu yhteydessä edellisen kampanjan yhteydessä vai ei. Hieman yllättäen, nr.employed ei ollut tilastollisesti merkitsevä logistisessa regressiomallissa. Kolmen kuukauden euribor oli logistisessa regressiomallissa tilastollisesti merkitsevä 10%:n merkitsevyystasolla ja previous_campaign oli tilastollisesti merkitsevä 1%:n merkitsevyystasolla.



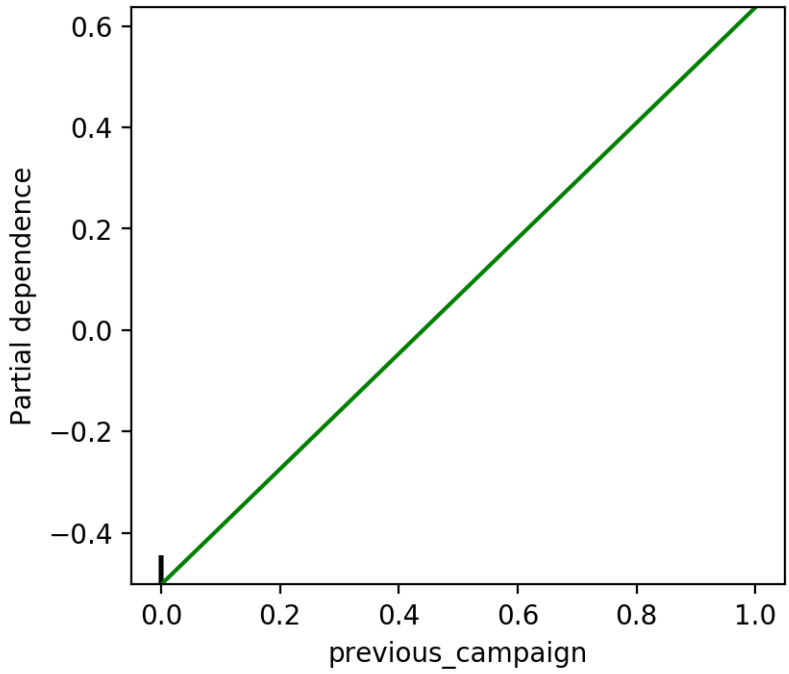
Kuvio 16. GT-mallin tärkeimmät muuttujat.

Kuviossa 17 on kuvattu kahden tärkeimmän muuttujan osittaisen riippuvuuden kuvaajat. Kuviosta nähdään, että kun nr.employed kasvaa, todennäköisyys että asiakas tekee pitkäaikaistalletuksen laskee. Muuttujan nr.employed suhde ei ole lineaarista, vaan noin puolivälissä kuviota nähdään jyrkkä pudotus. Kolmen kuukauden euriborin vaikutus y-muuttujaan on mielenkiintoinen. Ensin näyttäisi että euriborin noustessa kauppojen syntymisen todennäköisyys laskee, mutta aivan korkeimpien euriborin arvojen kohdalla nähdään piikki, eli todennäköisyys kauppojen syntymiselle on korkeimmillaan korkeimmilla euriborin arvoilla.



Kuvio 17. Osittaisen riippuvuuden kuvaajat muuttujille nr.employed ja euribor3m.

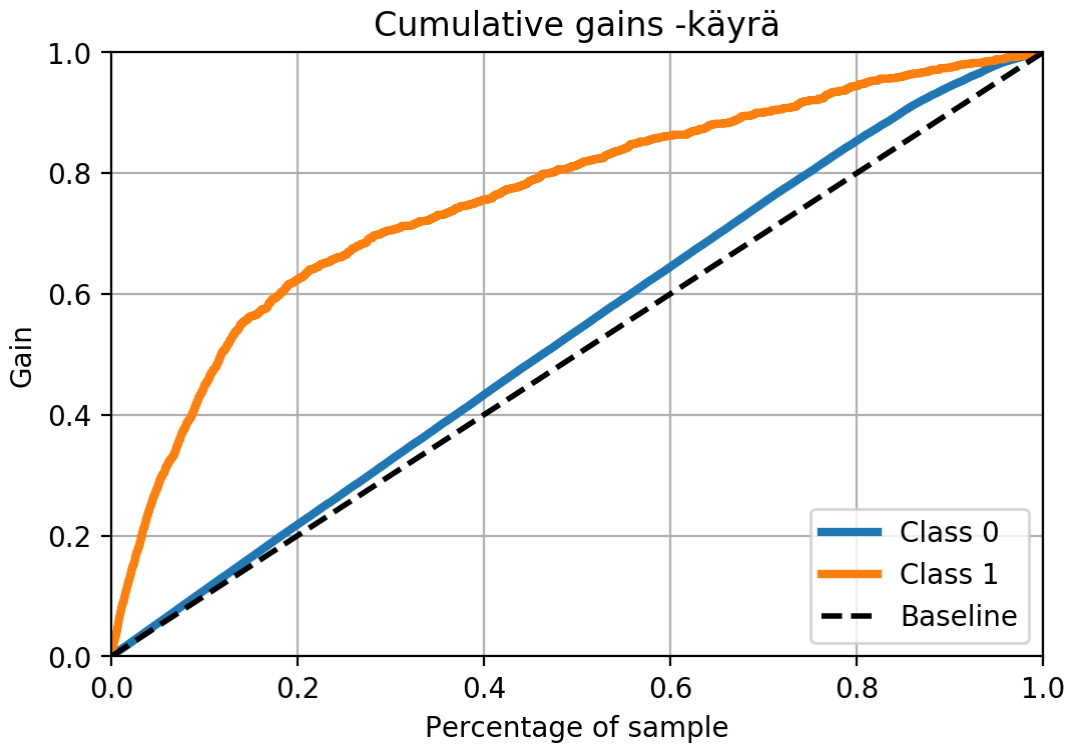
Kolmanneksi tärkeimmän muuttujan eli previous_campaign muuttujan osittaisen riippuvuuden kuvaaja on esitetty kuviossa 18. Muuttujan previous_campaign vaikutus on suoraviivainen. Kun asiakkaaseen on oltu yhteydessä aikaisemman kampanjan yhteydessä, on kauppojen syntyminen todennäköisempää. Mallin tuloksista ei voida kuitenkaan vetää kovin pitkälle meneviä johtopäätöksiä. Käytetty aineisto on kerätty vajaan kahden vuoden aikana, joten esimerkiksi neljännesvuosittaisen nr.employed-indikaattorin eri arvoja on aineistossa vähän. Mielenkiintoista tuloksissa on kuitenkin se, että merkittävimmät kaupan syntymiseen vaikuttavat muuttujat, nr.employed ja kolmen kuukauden euribor, ovat sellaisia, joihin pankki ei itse voi vaikuttaa millään tavalla. Jos esimerkiksi jotkin asiakkaiden ominaisuuksia kuvaavat muuttujat olisivat olleet merkittäviä ennustamisen kannalta, myyntipuhelut voitaisiin kohdistaa asiakkaille näiden ominaisuuksien perusteella.



Kuvio 18. Muuttujan previous_campaign osittaisen riippuvuuden kuvaaja.

Lopullisesta päätöspuusta kasvoi kohtalaisen suuri, eikä sen tulkinta ole aivan yksinkertaista. Koska päätöspuu on suuri sitä ei ole esitetty graafisesti. Lopullisessa päätöspuussakin kuitenkin selkeästi tärkein muuttuja oli nr.employed; se on ensimmäinen muuttuja jonka perusteella päätöspuu jakaa aineiston. Myös kolmen kuukauden euribor on useissa päätöspuun solmuissa merkittävänä muuttujana ennustettaessa tekeekö asiakas talletuksen vai ei.

Koska ROC-käyrä ja AUC-tunnusluku ovat hieman abstrakteja, päätettiin demonstroida GT-mallin ennustamiskykyä niin sanotun cumulative gains -käyrän avulla. Tämä käyrä on kuvattu kuviossa 19. Kuviossa vaaka-akselilla on testausaineiston osuus ja pystyakselilla on eri luokkiin kuuluvien havaintojen osuus. Cumulative gains -käyrästä nähdään esimerkiksi, että jos haluttaisiin saavuttaa 80% positiiviseen luokkaan kuuluvista havainnoista, riittäisi että oltaisiin yhteydessä noin 50% kaikista asiakkaista. Näin voitaisiin esimerkiksi säästää markkinointikuluissa ja kohdistaa markkinointitoimenpiteet potentiaalisimpiin asiakkaisiin. Logistisen regression cumulative gains -käyrä on hyvin samankaltainen kuin GT-mallin käyrä, joten sitä ei ole erikseen tässä kuvattu.



Kuvio 19. GT-mallin cumulative gains-käyrä.

Tässä osiossa tarkoituksena oli vertailla logistista regressiota koneoppimismenetelmiin. Tarkoituksena oli ennen kaikkea vertailla koneoppimismenetelmien ja logistisen regression ennustamistarkkuuksia. Vaikka koneoppimisen vahvuutena yleensä pidetään juuri parempaa ennustamiskykyä, tässä tapauksessa koneoppimismallit eivät ennustaneet juuri sen paremmin kuin logistinen regressiokaan. Koneoppimismalleista gradientti tehostaminen -mallia myös tulkittiin. Vaikka gradientti tehostamisen ja logistisen regression ennustamistarkkuudet olivat hyvin samankaltaiset, niin mallien toiminnassa on kuitenkin joitain eroja. Gradientti tehostamisessa selkeästi tärkein muuttuja nr.employed ei ollut logit-mallissa merkitsevä edes 10%:n merkitsevyystasolla. Tämä mahdollisesti tarkoittaa, että gradientti tehostaminen löysi aineistosta jonkinlaisen rakenteen, jota logit-mallista ei voida päätellä. Mikäli käytössä olisi ollut vielä enemmän havaintoja on mahdollista, että koneoppimismenetelmien avulla olisi saavutettu logit-mallia selkeästi parempi ennustamistarkkuus. Tämä empiirinen osuus kuitenkin demonstroi hyvin sitä, että koneoppimismenetelmät eivät välttämättä ole joka tilanteessa parempia kuin perinteiset regressiomallit.

6 JOHTOPÄÄTÖKSET

Tämän tutkielman tavoitteena oli selvittää, voitaisiinko koneoppimista hyödyntää taloustieteellisessä tutkimuksessa, ja verrata koneoppimismenetelmiä ekonometrian perinteisiin menetelmiin. Tutkimuksen tuloksena voidaan sanoa, että koneoppimista voidaan hyödyntää taloustieteellisessä tutkimuksessa. Näin voidaan sanoa sillä perusteella, että koneoppimista on jo hyödynnetty useissa tutkimuksissa. Koneoppimismenetelmien käyttämiseen tutkimuskäytössä liittyy kuitenkin joitain ongelmia.

Ensinnäkin, tehokkaimpien koneoppimismallien tulkinta voi olla haastavaa. Mallien tulkinta on tarpeellista, koska tutkimuksessa ei ole usein hyötyä pelkästä ennustamisesta, vaan kiinnostuksen kohteena on eri muuttujien väliset suhteet. Toiseksi, taloustieteellisessä tutkimuksessa yleensä pyritään selvittämään jonkinlaisia syy-seuraussuhteita. Koneoppimismenetelmät on kuitenkin tarkoitettu ennustamiseen, eikä syy-seuraussuhteiden selvittämiseen. Viime vuosina on kuitenkin pyritty kehittämään uusia menetelmiä, joissa yhdistyvät koneoppimismenetelmien ja ekonometrian hyvät puolet. Näiden uusien koneoppimismenetelmien avulla voidaan myös estimoida syy-seuraussuhteita. Tällainen koneoppimismenetelmien muokkaaminen tutkimuskäyttöön soveltuviksi on välttämätöntä, jotta koneoppimista voidaan käyttää laajamittaisesti tutkimuskäytössä.

Koneoppimismenetelmät vaativat myös huomattavan paljon havaintoja mallien opettamiseen. Kun havaintoja on vain vähän saatavilla, niin koneoppimismenetelmät eivät ennusta sen paremmin kuin perinteisetkään ekonometrian menetelmät. Tämän tutkielman empiirisessä osuudessa käytettiin koneoppimismenetelmiä sekä logistista regressiota ennustamaan pankin asiakkaiden käytöstä. Logistisen regression ennustamistarkkuutta verrattiin koneoppimismallien ennustamistarkkuuksiin. Empiirisen osuuden lopputulos oli, että tässä tapauksessa koneoppimismallit eivät ennustaneet juuri sen paremmin kuin logistinen regressiokaan. Empiirisen osuuden tulokset demonstroivat hyvin sitä tosiasiaa, että koneoppimismallit eivät välttämättä ole joka tilanteessa parempia ennustamaan kuin perinteiset regressiomallit.

Potentiaalisin käyttökohde koneoppimiselle tutkimuskäytössä onkin niiden hyödyntäminen suurten tietomassojen eli niin sanotun big datan analysoinnissa.

LÄHTEET

Akay, E. Ç., Astar, M., & Topal, K. H. (2018). Reexamining the returns to education and wages in Turkey: Supervised machine learning methods. *Finance & econometrics*, 97.

Andini, M., Ciani, E., de Blasio, G., D'Ignazio, A. & Salvestrini, V. (2018). Targeting with machine learning: An application to a tax rebate program in Italy. *Journal of Economic Behavior & Organization, 156*, 86-102.

Athey, S. (2018). The impact of machine learning on economics. *The Economics of Artificial Intelligence: An Agenda.* University of Chicago Press.

Athey, S., Imbens, G., & Wager, S. (2018). Approximate residual balancing: debiased inference of average treatment effects in high dimensions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 80*(4), 597-623.

Athey, S., & Imbens, G. (2015). Machine learning methods for estimating heterogeneous causal effects. *stat*, 1050(5).

Bajari, P., Nekipelov, D., Ryan, S. P., & Yang, M. (2015). Machine learning methods for demand estimation. *American Economic Review*, 105(5), 481-85.

Behr, A., & Weinblat, J. (2017). Default patterns in seven eu countries: A random forest approach. *International Journal of the Economics of Business*, *24*(2), 181-222. doi: 10.1080/13571516.2016.1252532

Biecek, P. (2018). DALEX: explainers for complex predictive models in R. *The Journal of Machine Learning Research, 19*(1), 3245-3249.

Boelaert, J., Ollion, E. (2018) The Great Regression. Machine Learning, Econometrics, and the and the Future of Quantitative Social Sciences. *Revue française de sociologie, Centre National de la Recherche Scientifique,* (painossa)

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition, 30*(7), 1145-1159.

Breiman, L. (1996). Bagging predictors. *Machine learning,* 24(2), 123-140.

Breiman, L. (1997). Arcing the edge. *Technical Report 486*, Statistics Department, University of California at Berkeley.

Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical science*, 16(3), 199-231.

Chawla, N., (2009). *Data mining for imbalanced datasets: An overview. In Data mining and knowledge discovery handbook*. Springer, Boston, MA.

Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., & Newey, W. K. (2016). Double machine learning for treatment and causal parameters (No. CWP49/16). cemmap working paper, *Centre for Microdata Methods and Practice*.

Clavería González, Ó., Monte Moreno, E., & Torra Porras, S. (2016). Modelling tourism demand to Spain with machine learning techniques. The impact of forecast horizon on model selection. *Revista de Economia Aplicada*, *72*(24), 109-132.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78-87.

Domingos, P. (2015). The master algorithm: How the quest for the ultimate learning machine will remake our world. Basic Books, New York.

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Einav, L., & Levin, J. (2014). Economics in the age of big data. *Science*, 346(6210), 1243089.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences,* 55(1), 119-139.

Gogas, P., Papadimitriou, T., & Agrapetidou, A. (2018). Forecasting bank failures and stress testing: A machine learning approach. *International Journal of Forecasting*, *34*(3), 440-455.

Green, D. P., & Kern, H. L. (2012). Modeling heterogeneous treatment effects in survey experiments with Bayesian additive regression trees. *Public Opinion Quarterly*, *76*(3), 491-511.

Hall, A. S. (2018). Machine Learning Approaches to Macroeconomic Forecasting. *Federal Reserve Bank of Kansas City Economic Review (USA),* (4), 5-23.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction.* Haettu osoitteesta: http://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf

Ho, T. K. (1995). Random decision forests. *In Document analysis and recognition, 1995., proceedings of the third international conference on* (Vol. 1, pp. 278-282). IEEE.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning.* Haettu osoitteesta: https://www-bcf.usc.edu/~gareth/ISL/ISLR%20First%20Printing.pdf

Kleinberg, J., Lakkaraju, H., Leskovec, J., Ludwig, J., & Mullainathan, S. (2017). Human decisions and machine predictions. *The Quarterly Journal of Economics, 133*(1), 237-293.

Le, H., & Viviani, J. (2018). Predicting bank failure: An improvement by implementing a machine-learning approach to classical financial ratios. *Research in International Business and Finance, 44*, 16-25.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems* 4765-4774.

Moro, S., Cortez, P., & Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62, 22-31.

Muchlinski, D., Siroky, D., He, J., & Kocher, M. (2016). Comparing random forest with logistic regression for predicting class-imbalanced civil war onset data. *Political Analysis,* 24(1), 87-103.

Mullainathan, S., & Spiess, J. (2017). Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, 31(2), 87-106.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.

Ribeiro, T., Singh, S., & Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (s. 1135-1144). ACM.

Sohnesen, T., & Stender, N. (2016). Is random forest a superior methodology for predicting poverty? an empirical assessment. *The World Bank*. Policy Research Working Paper 7612

Stock, J. H., & Watson, M. W. (2012). *Introduction to econometrics*. Edinburgh Gate. Pearson Education Limited.

Tanaka, K., Kinkyo, T., & Hamori, S. (2016). Random forests-based early warning system for bank failures. *Economics Letters*, 148, 118-121.

Varian, H. (2014). Big data: New tricks for econometrics. *Journal of Economic Perspectives*, 28(2), 3-28.

Varian, H. R. (2016). Causal inference in economics and marketing. *Proceedings of the National Academy of Sciences, 113*(27), 7310-7315.

Vomfell, L., Härdle, W. K., & Lessmann, S. (2018). Improving Crime Count Forecasts Using Twitter and Taxi Data. *IRTG 1792 Discussion Paper* 2018-013

Wager, S., & Athey, S. (2018). Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association, 113(523)*, 1228-1242.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., & Zhou, Z. H. (2008). Top 10 algorithms in data mining. *Knowledge and information systems, 14*(1), 1-37.

Zou, K. H., O'Malley, A. J., & Mauri, L. (2007). Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models. *Circulation, 115*(5), 654-657.

Liite 1

Python ohjelmointi

```
import pandas as pd
df = pd.read_csv("Downloads/bank-additional/bank-additional-
full.csv", sep=";")

import seaborn as sns
import matplotlib.pyplot as plt

ax = sns.countplot(df['y'], label='lkm')
ax.set_ylabel('lkm')
plt.savefig('countplot.png', dpi=200, bbox_inches='tight')
plt.show()

df['y'] = df['y'].map({'no':0, 'yes':1})

import numpy as np
bins = [0, 998, np.inf]
names = ['previous_campaign', 'no_previous_campaign']

df['pdays'] = pd.cut(df['pdays'], bins, labels=names)

num_attribs = ['age', 'campaign', 'previous', 'emp.var.rate',
'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']
cat_attribs = ['marital', 'month', 'day_of_week', 'pdays']
ord_attribs = ['job', 'education', 'default', 'housing', 'loan',
'contact', 'poutcome']

from sklearn.preprocessing import OrdinalEncoder

ordinal_encoder = OrdinalEncoder()
df_ordinal = df[ord_attribs]
df_ordinal_encoded =
ordinal_encoder.fit_transform(df_ordinal.astype('str'))

ordinal_df = pd.DataFrame(df_ordinal_encoded, columns=ord_attribs)

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler, StandardScaler

num_pipeline = Pipeline([
        ('minmax_scaler', MinMaxScaler()),
    ])

df_num_tr = num_pipeline.fit_transform(df[num_attribs])

numer_df = pd.DataFrame(df_num_tr, columns=num_attribs)

full_df = pd.concat([numer_df.reset_index(drop=True),
ordinal_df.reset_index(drop=True)], axis=1)

for col in cat_attribs:
    dummies = pd.get_dummies(df[col])
    dummies.drop(dummies.columns[len(dummies.columns)-1], axis=1,
inplace=True)
    full_df = pd.concat([full_df.reset_index(drop=True),
dummies.reset_index(drop=True)], axis=1)
```

```
# Aineiston jakaminen opetus-, validointi- ja testausosiin

from sklearn.model_selection import train_test_split

X = full_df
y = df['y']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42)

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
test_size=0.3333, random_state=42)


import statsmodels.api as sm
#Vakiotermin lisääminen
X_train_copy = X_train.copy()
X_train_copy['intercept'] = 1

#Logit-Mallin sovittaminen
model = sm.Logit(list(y_train) , X_train_copy, method='ncg')
results = model.fit()

with open('summary.txt', 'w') as fh:
    fh.write(results.summary2().as_text())

from sklearn.metrics import accuracy_score, roc_auc_score,
confusion_matrix
def get_results(classifiers, names, X=X_train, y=y_train,
X_val=X_val, y_val=y_val, val_scores=True):

    results_dict = {}
    i=0

    for clf in classifiers:
        clf.fit(X, y)
        y_pred = clf.predict(X_val)

        train_set_acc = round(clf.score(X, y), 4)
        train_set_auc = round(roc_auc_score(y,
clf.predict_proba(X)[:,1]), 4)
        val_set_acc = round(accuracy_score(y_val, y_pred), 4)
        val_set_auc = round(roc_auc_score(y_val,
clf.predict_proba(X_val)[:,1]), 4)

        results_dict[names[i]] = [train_set_acc, train_set_auc,
val_set_acc, val_set_auc]
        i += 1

    results_df = pd.DataFrame.from_dict(results_dict,
orient='index')
    if val_scores:
        results_df.columns = ['train acc','train auc', 'val acc',
'val auc']
    else:
        results_df.columns = ['train acc','train auc', 'test acc',
'test auc']


    return results_df
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.svm import SVC

classifiers = [LogisticRegression(C=1e9, solver='newton-cg'),
KNeighborsClassifier(), DecisionTreeClassifier(random_state=42),
               RandomForestClassifier(random_state=42),
               SVC(kernel='linear', probability=True),
SVC(kernel='poly', degree=2, probability=True, gamma='auto'),
               SVC(kernel='poly', degree=3, probability=True,
gamma='auto'),
              GradientBoostingClassifier(random_state=42)]
clf_names = ['Logit', 'KNN', 'Decision tree', 'Random Forest',
             'SVC linear', 'SVC poly2', 'SVC poly3',
             'GBM', ]
print("Fitting models")
results1 = get_results(classifiers, clf_names)

from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler(random_state=42)
X_undersampled, y_undersampled = rus.fit_resample(X_train, y_train)
results_rus = get_results(classifiers, clf_names, X_undersampled,
y_undersampled)
print("Fitting models ROS")
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=42)
X_oversampled, y_oversampled = ros.fit_resample(X_train, y_train)
results_ros = get_results(classifiers, clf_names, X_oversampled,
y_oversampled)

#Vakioasetuksilla sovitettujen mallien tulokset
with pd.ExcelWriter('results.xlsx') as writer:
    results1.to_excel(writer, sheet_name='Original data')
    results_rus.to_excel(writer, sheet_name='Undersampled data')
    results_ros.to_excel(writer, sheet_name='Oversampled data')

rates = [0.05, 0.1, 0.15]
val_aucs = []
train_aucs = []
all_aucs = []
n_estimators1 = []
aucs1 = []


for lrate in rates:
    gbrt = GradientBoostingClassifier(max_depth=2,
n_estimators=1500, random_state=42, learning_rate=lrate)
    gbrt.fit(X_train , y_train)

    aucs = [roc_auc_score(y_val, y_pred[:,1])
              for y_pred in gbrt.staged_predict_proba(X_val)]

    train_set_aucs = [roc_auc_score(y_train, y_pred[:,1])
          for y_pred in gbrt.staged_predict_proba(X_train)]

    bst_n_estimators = np.argmax(aucs)
    max_auc = np.max(aucs)
```

```
        aucs1.append(max_auc)
        n_estimators1.append(bst_n_estimators)
        val_aucs.append(aucs)
        train_aucs.append(train_set_aucs)

best_lrate = rates[np.argmax(aucs1)]
print("Optimaalinen kutistustermi: {}".format(best_lrate))
max_auc = aucs1[np.argmax(aucs1)]
bst_n_estimators = n_estimators1[np.argmax(aucs1)]

gbrt_best =
GradientBoostingClassifier(max_depth=2,n_estimators=bst_n_estimators
, random_state=42, learning_rate=best_lrate)


plt.figure(figsize=(11, 8), dpi=200)
plt.plot(val_aucs[np.argmax(aucs1)], "b.-",
label="Validointiaineiston AUC")
plt.plot(train_aucs[np.argmax(aucs1)], "r.-", label="Opetusaineiston
AUC")
plt.plot([bst_n_estimators, bst_n_estimators], [0, max_auc], "k--")
plt.plot([0, 1000], [max_auc, max_auc], "k--")
plt.plot(bst_n_estimators, max_auc, "ko")
plt.text(bst_n_estimators, max_auc*1.002, "Maksimi\npuiden lkm:
"+str(bst_n_estimators), ha="center", fontsize=14)
plt.axis([0, 1000, 0.750, 0.86])
plt.xlabel("Puiden lkm")
plt.ylabel("AUC")
plt.legend(fontsize=14)

plt.savefig('gbm_earlystopping.png', dpi=200, bbox_inches='tight')
plt.show()

gbrt_best.fit(X_train, y_train)


y_pred = gbrt_best.predict(X_test)
cmat = confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(cmat)
plt.figure(figsize = (4,3), dpi=200)
ax = plt.axes()
sns.heatmap(df_cm, annot=True, fmt="d", cbar=False)
ax.set_xlabel("Ennustettu luokka")
ax.set_ylabel("Todellinen luokka")
ax.set_title("GT")
plt.savefig('CM_gbm.png', dpi=200, bbox_inches='tight')

from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(C=1e9, solver='newton-cg')
log_reg.fit(X_train, y_train)
y_pred = log_reg.predict(X_test)
cmat = confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(cmat)
plt.figure(figsize = (4,3), dpi=200)
ax = plt.axes()
sns.heatmap(df_cm, annot=True, fmt="d", cbar=False)
ax.set_xlabel("Ennustettu luokka")
ax.set_ylabel("Todellinen luokka")
ax.set_title("Logit")
plt.savefig('CM_logit.png', dpi=200, bbox_inches='tight')
```

```
from sklearn.model_selection import GridSearchCV
param_grid = {'max_depth':[1,5,7,9],
'min_samples_leaf':[1,5,10,12,18,22,26,30]}
dt_clf = DecisionTreeClassifier(random_state=42)
grid_search = GridSearchCV(dt_clf, param_grid=param_grid, cv=5,
scoring='roc_auc')
grid_search.fit(X_train, y_train)

df_gridsearch =
pd.DataFrame(grid_search.cv_results_)[['param_max_depth','param_min_
samples_leaf','mean_test_score']]
max_scores =
df_gridsearch.groupby(['param_max_depth','param_min_samples_leaf']).
max().unstack()
fmt = 'png'
plt.figure(figsize = (8,3), dpi=200)
ax = plt.axes()
sns.heatmap(max_scores.mean_test_score, annot=True, fmt='.4g')
ax.set_xlabel("Havaintojen minimimäärä lehtisolmuissa")
ax.set_ylabel("Maksimisyvyys")
plt.savefig('DT_gridsearch.png', dpi=200, bbox_inches='tight')

dt_best = grid_search.best_estimator_
dt_best.fit(X_train, y_train)

y_pred = dt_best.predict(X_test)
cmat = confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(cmat)
plt.figure(figsize = (4,3), dpi=200)
ax = plt.axes()
sns.heatmap(df_cm, annot=True, fmt="d", cbar=False)
ax.set_xlabel("Ennustettu luokka")
ax.set_ylabel("Todellinen luokka")
ax.set_title("Päätöspuu")
plt.savefig('CM_päätöspuu.png', dpi=200, bbox_inches='tight')

#Lopulliset tulokset
final_results = get_results(classifiers=[log_reg, dt_best,
gbrt_best],
                            X_val=X_test, y_val=y_test,
names=['logit','DT','GT'], val_scores=False)

final_results.to_excel('final_results.xlsx')

from sklearn.tree import export_graphviz
export_graphviz(
            dt_best,
            out_file="bank_tree.dot",
            feature_names=list(X.columns),
            class_names=['0','1'],
            rounded=True,
            filled=True
)


classifiers = {'Logit':log_reg, 'Päätöspuu': dt_best,
'GT':gbrt_best}

#ROC-käyrä
from sklearn.metrics import roc_curve
plt.figure(figsize=(8,6), dpi=200)
```

```
for name, model in classifiers.items():
    y_pred_prob = model.predict_proba(X_test)[:,1]

    fpr, tpr, tresholds = roc_curve(y_test, y_pred_prob)
    roc_auc = roc_auc_score(y_test, y_pred_prob)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.plot(fpr, tpr, label=name+' (%0.4f)'  % roc_auc)
    plt.xlabel('1-Spesifisyys')
    plt.ylabel('Herkkyys')
    plt.title('ROC Käyrä')

plt.legend(loc="lower right")
plt.xlim([-0.02, 1.0])
plt.ylim([0.0, 1.05])
plt.savefig('ROC_kkäyrä.png', dpi=200, bbox_inches='tight')
plt.show()

#Tärkeimmät muuttujat
feature_importances = gbrt_best.feature_importances_
attributes = X_train.columns.astype(str)
feat_imp = sorted(zip(feature_importances, attributes),
reverse=True)

x_value = [x[1] for x in feat_imp[:10]]
y_value = [x[0] for x in feat_imp[:10]]

plt.figure(figsize=(8,4), dpi=200)
sns.barplot(y_value,x_value)
plt.title("Gradientti tehostaminen - 10 tärkeintä muuttujaa")
plt.savefig('fimp.png', dpi=200, bbox_inches='tight')
plt.show()

#Osittaisen riippuvuuden kuvaajat
from sklearn.ensemble.partial_dependence import
plot_partial_dependence
def part_plot(a,b):
    fig, axs = plot_partial_dependence(gbrt_best,
X_train,feature_names=list(X_train.columns),

features=[list(X_train.columns).index(a),list(X_train.columns).index
(b)], grid_resolution=10)

    fig.set_dpi(200)
    plt.subplots_adjust(top=0.9)
    fig.set_size_inches(7, 4)

part_plot('nr.employed','euribor3m')
plt.savefig('pdp1.png', dpi=200, bbox_inches='tight')

fig, axs = plot_partial_dependence(gbrt_best,
X_train,feature_names=list(X_train.columns),

features=[list(X_train.columns).index('previous_campaign')],
grid_resolution=10)
fig.set_dpi(200)
plt.subplots_adjust(top=0.9)
fig.set_size_inches(4, 4)
plt.savefig('pdp2.png', dpi=200, bbox_inches='tight')

#Cumulative gains-käyrä
import scikitplot as skplt
```

```
y_probas = gbrt_best.predict_proba(X_test)
skplt.metrics.plot_cumulative_gain(y_test, y_probas,
title="Cumulative gains -käyrä")
plt.savefig('cumulative_gains.png', dpi=200, bbox_inches='tight')
plt.show()
```