HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

MATEMAATTIS-LUONNONTIETEELLINEN TIEDEKUNTA
MATEMATISK-NATURVETENSKAPLIGA FAKULTETEN
FACULTY OF SCIENCE

Tiedekunta – Fakultet – Faculty Matemaattis-luonnontieteellinen tiedekunta		Koulutusohjelma – Utbildningsprogram – Degree programme Filosofian maisteri
Tekijä – Författare – Author Suvi Telivuo		
Työn nimi – Arbetets titel – Title Satunnaismetsä-koneoppimismenetelmä, teoria ja soveltaminen		
Työn laji – Arbetets art – Level Pro Gradu-tutkielma	Aika – Datum – Month and year Toukokuu 2018	Sivumäärä – Sidoantal – Number of pages 44

Tiivistelmä – Referat – Abstract

Taloudellisten, institutionaalisten ja teknologisten ympäristöjen kiihtyvä muutostahti on luonut tarpeen tehdä oikeita valintoja menestymisen ja kehityksen takeeksi. Parhaimman valinnan tekee henkilö, jolla on eniten tietoa ja varmuutta tiedon paikkansapitävyydestä. Vaihtoehtoisesti päätöksenteon epävarmuustekijöitä voidaan hallita eliminointimenetelmillä, joiden hyödyntäminen voi myös johtaa parempiin päätöksiin. Epävarmuuden minimoimisen edellytyksenä on niin ikään pohjatietojen parantaminen. Tästä tarpeesta ovat nousseet tiedonlouhintamenetelmät.
Tiedonlouhintamenetelmiä on kehittynyt valtava määrä vastaamaan kysynnän luomia tarpeita. Päätöksentekopuu on eräs tällainen analysointimenetelmä ja päätöksentekopuun pohjalta on luotu koneoppimismenetelmä satunnaismetsä. Satunnaismetsä on tutkimusten mukaan tällä hetkellä paras saatavilla oleva luokittelumenetelmä ja valikoitunut tämän tutkielman aiheeksi.
Luvussa 2 luomme pohjaa satunnaismetsä-menetelmän ymmärtämiseksi. Lähdemme liikkeelle koneoppimisesta ja datalouhintamenetelmistä, joilla alustamme päätöksentekopuutyökalun. Käy ilmi, että on olemassa luokittelu-, regressio- ja luokitteluregressiopuita, ja että tässä tutkielmassa keskitymme luokittelupuihin. Tämän jälkeen esittelemme päätöksentekopuun metodologiaa.
Luvussa 3 esittelemme tutkielman kannalta päätöksentekopuiden tärkeimmät validointimenetelmät, sillä datalouhinnassa analysointimenetelmien validoiminen on yhtä tärkeää kuin itse analysoiminen. Esittelemme mallien validointiin liittyviä käsitteitä, kuten tarkkuus, yleistysvirhe ja ylisovittuminen. Käymme läpi yleisimpiä tapoja validoida malleja, sekä näytämme esimerkkien kautta työkalut, joita käytämme tutkielmassa. Näitä ovat tarkkuus, väärinluokittelumatriisi, ROC-, kumulatiivinen saanti- ja nostokäyrä.
Luvussa 4 esittelemme satunnaismetsä-koneoppimismenetelmän. Käymme ensiksi läpi joukko-oppimisen metodologiaa, jonka jälkeen käsittelemme satunnaismetsän algoritmin. Osoitamme teoreettisesti, miksi satunnaismetsä on parempi luokittelija kuin esimerkiksi päätöksentekopuu näyttämällä, että satunnaismetsän puumäärän kasvaessa satunnaismetsän yleistysvirhe suppenee kohti nollaa. Analysoimme teoreettisesti satunnaismetsän hyötyjä ja haittoja. Satunnaismetsän hyötyjä ovat sen tarkkuus, nopeus, ymmärrettävyys, toimivuus valtavilla datamäärillä, sekä kykeneväisyys analysoida tietojoukon merkittävimpiä muuttujia. Haittoja ovat, että satunnaismetsä ei suoriudu yhtä hyvin regressio-ongelmissa kuin esimerkiksi logistinen regressiomalli, sekä huono sovellettavuus pieniin tietojoukkoihin.
Luvussa 5 sovellamme opittuja taitoja suppeaan tietojoukkoon. Tarkoituksenamme on arvioida RStudion ja SAS Enterprise Minerin satunnaismetsä-pakettien toimivuutta tunnetulla syötejoukolla. Analysoimme satunnaismetsäin suoriutumista luokittelutehtävässä ja vertailemme tuloksia päätöksentekopuuhun ja regressiomalliin. Hyödynnämme luvussa 3 opittuja validointimenetelmiä. Käy ilmi, että RStudion ja SAS Enterprise Minerin satunnaismetsä-paketit toimivat hyvin, ja että satunnaismetsä suoriutuu pienenkin tietojoukon luokittelussa malleista parhaiten.
Luvussa 6 sovellamme satunnaismetsää yrityksen tarjoamaan haasteeseen, jossa tarkoitus on selittää ja ennustaa uusien asiakkaiden tietyn asiakassegmentin asiakasvaihtuvuutta. Käytämme yhtiön tarjoamia tietokantoja ja SAS Enterprise Miner-työkalua. Suoritamme vertailun satunnaismetsä- ja päätöksentekopuu-mallien välillä käyttämällä luvussa 3 esiteltyjä validointimenetelmiä ja analysoimme tulokset. Käy ilmi, että ROC-käyrien ja tarkkuuden perusteella satunnaismetsä suoriutuu sekä luokittelussa että ennustamisessa paremmin kuin päätöksentekopuu.
Luvussa 7 pohdimme, millaisissa puitteissa satunnaismetsä soveltuu yrityksen liiketoimintaprosessiin. Käymme läpi vaatimuksia, joita satunnaismetsän soveltaminen asettaa, sekä mitä lisäarvoa satunnaismetsä menetelmänä tuo yritykselle. Tulos on, että satunnaismetsä soveltuu hyvin yrityksille, jotka hyödyntävät SAS-työkaluja ja tuo lisäarvoa analysointitehtäviin olemalla ymmärrettävä malli, mutta kuitenkin monipuolinen, nopea ja tarkka.

Avainsanat – Nyckelord – Keywords
tiedonlouhinta, koneoppimismenetelmä, joukko-oppimismenetelmä, luokitteluongelma, päätöksentekopuu, Satunnaismetsä

Säilytyspaikka – Förvaringställe – Where deposited
Kumpulan tiedekirjasto

Muita tietoja – Övriga uppgifter – Additional information

Satunnaismetsä-koneoppimismenetelmä, teoria ja soveltaminen

Suvi Telivuo
Matematiikan ja tilastotieteen laitos
Helsingin Yliopisto

14. toukokuuta 2018

Sisältö

Luku 1

Johdanto

Taloudellisten, institutionaalisten ja teknologisten ympäristöjen kiihtyvä muutostahti on luonut tarpeen tehdä oikeita valintoja menestymisen ja kehityksen takeeksi. Parhaimman valinnan tekee henkilö, jolla on eniten tietoa ja varmuutta tiedon paikkansapitävyydestä. [1] Vaihtoehtoisesti päätöksenteon tukena voidaan käyttää eliminointimenetelmiä, jotka minimoivat epävarmuustekijöiden määrää. Epävarmuuden minimoimisen edellytyksenä on niin ikään pohjatietojen parantaminen. Tätä tarkoitusta varten on kehitetty tiedonlouhintamenetelmät.

Internetin välityksellä tietoa jaetaan yhä nopeammin, tallennuskapasiteettien kasvaessa tietoa on mahdollista säilöä yhä suurempia määriä ja laskentatehon kasvaessa säilömisen ohella tietoa kyetään myös hyödyntämään tehokkaammin. Niin sanottu *hard data* eli numeerinen tieto ei sellaisenaan kuitenkaan ole hyödynnettävissä, sillä tietokantataulussa oleva binäärivaste tuottaa käyttöarvoa vasta tulkintojen kautta. Tietyllä tapaa data-analyysi on sofistikoitunutta tulkitsemista, jonka tarpeellisuuteen markkinoinnissa ja toiminnan optimoinnissa yritykset ovat hiljattain heränneet.

Manuaalinen malli on osattu erottaa tietokannasta jo hyvän aikaa, kuten varhaiset mallin tunnistamismetodit Bayesin teoreema 1700-luvulta ja regressioanalyysi 1800-luvulta osoittavat. Tietokone mullisti nämä menetelmät kasvattamalla eksponentiaalisesti käytettävän tiedon määrää. Nykypäivänä data-analyysiksi kutsuttua käytäntöä esiintyi sellaisenaan 1950-luvulla ja menetelmää harjoittivat lähinnä tilastotieteilijät. Vuonna 1989 Gregory Piatetsky-Shapiro kehitti termin "knowledge discovery in databases"(myöhemmin KDD-yhteisö), josta tuli suosittu tekoälyn ja koneoppimisen ammattilaisten parissa. Yritysten ja lehdistön vaikutuksesta termi lyhentyi *tiedonlouhinnaksi*.

1700-luvulla varhaisten data-analyytikoiden osaaminen kohdistui sopivan

lähtöjoukon valitsemiseen ja kyseisen joukon tulkitsemiseen. 1950-luvulla tiedon säilömiskapasiteetti kasvoi niin suureksi, että data-analyytikoiden toimenkuvassa alkoi painottua kyky manipuloida valtavia tietomääriä. Tietopankkien kasvaessa tämä kyseinen taito korostuu entisestään. Tietoa on paljon, joten analysointitarpeita on monenlaisia ja tarpeen luoman kysynnän seurauksena data-analysointitekniikoita on kehitetty valtava määrä. Luokittelumallia edustava päätöksentekopuu on yksi tällainen analysointimenetelmä, ja päätöksentekopuun pohjalta on kehitetty myös koneoppimismenetelmä satunnaismetsä. Satunnaismetsä on tutkimusten mukaan [2] tällä hetkellä paras saatavilla oleva luokittelumalli sen suoman tarkkuuden vuoksi ja on valikoitunut tämän tutkielman aiheeksi.

Esittelemme tutkielmassa ensiksi päätöksentekopuun ja koneoppimismenetelmiä yleisesti, sekä analysointimenetelmien validointimenetelmiä. Näillä kartutamme perustietoja satunnaismetsä-menetelmän ymmärtämiseksi ja arvioimiseksi. Tämän jälkeen luvussa 4 tutustutaan satunnaismetsän algoritmiin. Luvussa 5 sovellamme menetelmää luokittelutehtävään, jossa on käytössä suppea tietokanta. Luvussa 6 sovellamme menetelmää yrityksen tarjoamaan reaalimaailman tapaukseen, jossa tarkoituksena on luokittelemalla selittää ja ennustaa yrityksen asiakasvaihtuvuutta. Tehtävän yhteydessä suoritamme vertailun satunnaismetsä- ja päätöksentekopuu-mallin välillä. Hyödynnämme validointimenetelmiä selvittääksemme, suoriutuuko satunnaismetsä tehtävästä paremmin kuin päätöksentekopuu. Lopuksi tutkielmassa analysoidaan millä ehdoilla satunnaismetsä soveltuu yrityksen liiketoimintaprosessiin.

Luku 2

Päätöksentekopuu

Tässä kappaleessa esittelemme päätöksentekopuutyökalua. Lähdemme liikkeelle koneoppimisesta ja tiedonlouhintamenetelmistä, joilla selvennämme päätöksentekopuutyökalun taustoja. Tämän jälkeen esittelemme päätöksentekopuun metodologiaa.

2.1 Koneoppiminen

Koneoppiminen on tietojenkäsittelytieteen haara, jossa tavoitteena on kehittää algoritmeja, jotka mahdollistavat tietokoneen kyvyn oppia ilman erityistä, tehtäväorientoitunutta ohjelmointia [3]. Koneoppiminen on kehittynyt tekoälytutkimuksissa kaavantunnistamisen ja oppimisteorian ohessa.

Koneoppiminen tutkii ja kehittää algoritmeja, jotka voivat oppia ja tehdä havaintoja datasta ja strukturoidun ohjelman sijasta valinnat ovat datavetoisia. Koneoppimista harjoitetaan tehtävissä, joissa eksplisiittisen algoritmin luominen on hankalaa tai jopa mahdotonta, esimerkiksi lääketieteessä diagnosoinnin apuna tapauksissa, joissa jos-niin-ehtolauseet eivät riitä taudinkuvan tunnistamiseen.

Koneoppiminen jaetaan kahteen laajaan kategoriaan: ohjattu oppiminen ja ohjaamaton oppiminen (engl. *supervised learning* ja *unsupervised learning*). Ohjatussa oppimisessa tietokoneelle esitellään esimerkkijoukko, jossa jokaiselle syötemuuttujalle on määritelty toivottu lopputulos. Koneen tehtävänä on löytää funktio, jolla kuvata lähtöjoukko maalijoukoksi. Ohjaamattomassa oppimisessa koneelle ei anneta osviittaa halutusta lopputuloksesta, vaan koneen on tarkoitus löytää lähtöjoukon alkioiden välinen rakenne omin neuvoin. Tämä voi olla itsessään jo tavoite, esimerkiksi mikäli tarkoitus on löytää satunnaisesta lähtöjoukosta jokin yhdistävä tekijä.

Tässä tutkielmassa käsittelemme luokittelumalleja, jotka kuuluvat ohjatun

oppimisen piiriin.

2.2 Datalouhinta

Tietotekniikan ja teknologian kehittyminen on merkinnyt mahdollisuutta säilöä yhä enemmän tietoa ja käsitellä yhä suurempia tietomääriä. Tämä on edellyttänyt laadukkaampien tiedonhallintamenetelmien kehittämistä, kun perinteinen manuaalinen käsittely on osoittautunut riittämättömäksi valtavien tietopankkien käsittelemiseen. Vallitseva trendi on sisällyttää käsiteltäväksi mahdollisimman paljon dataa, josta on mahdollista ammentaa lisää tietoa ja ymmärrystä.

Datalouhinta on tieteenala, joka pyrkii löytämään annetusta datasta aiemmin tunnistamattomia ja potentiaalisesti hyödyllisiä lainalaisuuksia [4]. Tieteenala on osa tietokantojen tiedonhakuprosesseja. Uuden teknologian myötä tiedon säilöminen on tullut entistä halvemmaksi ja siten kustannustehokkaammaksi. Tämä on johtanut siihen, että tietoa on tarjolla runsaasti ja helposti. Näin ollen kyvystä analysoida datan sisältöä on tullut merkittävä voimavara.

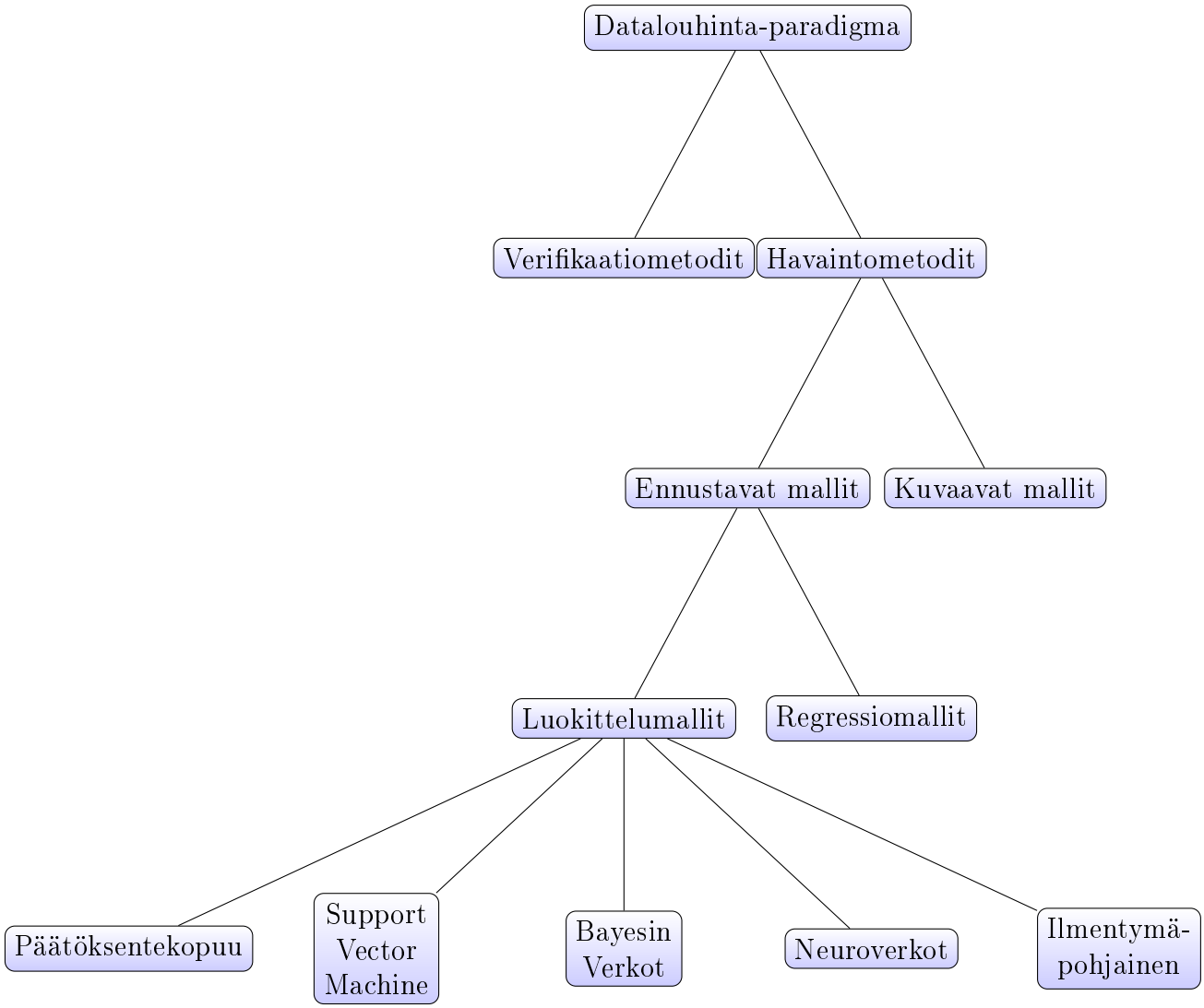
Datalouhintamenetelmät perustuvat pitkälti induktiivisiin operaatioihin, joissa malli on luotu yleistämällä kohdennettuun opetusdataan harjoitetut eksplisiittiset tai implisiittiset mallit. Tällaisen mallin luomisen tarkoituksena on, että opetusmalli on sovellettavissa ennalta tuntemattomaan dataan, jolloin pyritään muun muassa tekemään arvioita tulevaisuudesta. Oletuksena on, että samankaltaiset tapahtumat muistuttavat toisiaan attribuutti- eli muuttujatasolla. Toisin sanoen, kun tiedetään tietyn asiakasryhmän ominaispiirteet ja käyttäytyminen, voidaan ennustaa toisen asiakkasryhmän käyttäytymistä, mikäli näillä ryhmillä on yhteneviä ominaispiirteitä.

2.2.1 Menetelmiä

Datalouhinta-nimikkeen alla esiintyy monenlaista tekniikkaa, mutta Rokach ja Maimon jakavat kirjassaan [4] datalouhinnan kahteen alakategoriaan; verifikaatio-orientoituneet datalouhinnat, joissa järjestelmä verifioi käyttäjän hypoteesit, ja havainto-orientoituneet, joissa järjestelmä autonomisesti kehittää uusia lainalaisuuksia ja metodeita, kuva 2.1. Havainto-orientoituneet metodit etsivät annetusta datasta säännönmukaisuuksia autonomisesti. Nämä menetelmät sisältävät kuvaavia ja ennustavia metodeita. Kuvaavat metodit pyrkivät selittämään datan sisällä ilmeneviä samankaltaisuuksia, kun taas ennustavien metodien tarkoitus on luoda uutta ja rakentaa ennalta näkemätöntä dataa varten käytösmalli, jonka perusteella voi ennustaa näytejouk-

koon kuuluvia muuttujia. Kategorioiden välillä esiintyy päällekkäisyyksiä ja esimerkiksi jotkut ennustavat metodit toimivat myös kuvaavina metodeina.

Ennustavia metodeja pidetään koneoppimisen piirissä ohjattuna oppimisena ja näin ollen ohjattuina metodeina. Ohjatut metodit pyrkivät löytämään yhteyksiä syötettyjen muuttujien ja kohdemuuttujien välillä. Löydettyä yhteyttä kutsutaan malliksi. Mallit toimivat yleensä selittävinä tekijöinä ilmiöille, jotka ovat löydettävissä datasta. Mallin avulla voidaan ennustaa kohdemuuttuja, kun syötemuuttujat tunnetaan. Ohjattuja metodeita sovelletaan monipuolisesti markkina-, finanssi- ja tekniikan aloilla.



Kuva 2.1: Datalouhintamenetelmien luokittelu.

Ohjatut metodit, joihin ennustavat metodit kuuluvat, voidaan jakaa kahteen suurempaan alakategoriaan; luokittelumalleihin ja regressiomalleihin (Kuva 2.1). Regressiomallit kuvaavat lähtöjoukon reaaliluvuiksi ja luokittelumallit kuvaavat lähtöjoukon ennalta määrättyihin luokkiin.

2.3 Päätöksentekopuu

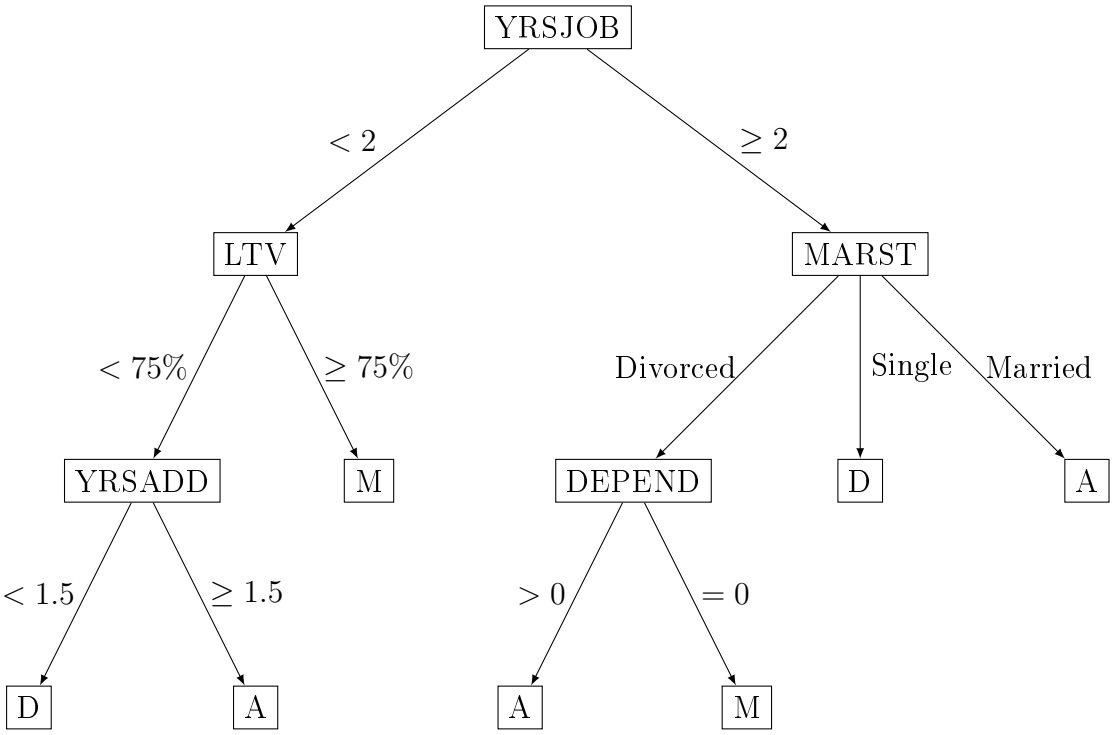
Päätöksentekopuu on ennustemalli, joka edustaa sekä regressio- että luokittelumalleja. Riippuen halutusta tehtävästä, päätöksentekopuuhun viitatataan joko luokittelupuuna, regressiopuuna tai luokitteluregressiopuuna eli CART:ina (*Classification And Regression Tree*). Regressiopuu eroaa luokittelupuusta siten, että regressiopuu kuvaa joukon jatkuvalle jakaumalle, kuten esimerkiksi reaaliluvuiksi, kun taas luokittelupuu jakaa kohteen tai tapahtuman attribuuttien perusteella ennalta määrättyihin luokkiin. Tässä tutkielmassa keskitymme luokittelupuiden metodologiaan.

Päätöksentekopuu on luokittelumalli, joka esitetään esimerkkijoukon rekursiivisena osituksena [4]. Tarkoitamme tällä sitä, että puulla on kohdistettu solmu, johon viittaamme "juurena", ja tähän juureen ei ole saapuvia oksia. Muilla solmuilla on yksi saapuva oksa. Lähteviä juuria kutsutaan testisolmuiksi ja kaikkia muita solmuja kutsutaan lehdiksi. Päätöksentekopuussa lähtöjoukko jaotellaan kahdeksi tai useammaksi alajoukoksi riippuen määritellystä jaottelufunktiosta. Jokainen lehti on määritelty yhdeksi luokaksi ja tämä luokka edustaa soveltuvinta kohdemuuttujaa. Vaihtoehtoisesti lehdessä voi esiintyä kohdemuuttujan tietyn arvon esiintymistodennäköisyyttä kuvaavia todennäköisyysarvoja.

Kuvassa 2.2 näemme tyypillisen esimerkin luokittelupuusta. Puu on malli pankin myöntämistä asuntolainoista. Lainanhakija täyttää hakemuksen, jossa kysytään seuraavia tietoja: elätettävien lukumäärä (DEPEND), laina-arvo (LTV), siviilisääty (MARST), palkka (PAYINC), korko (RATE), nykyisessä osoitteessa asutut vuodet (YRSADD) ja nykyisessä työpaikassa työskennellyt vuodet (YRSJOB). Näiden tietojen perusteella virkailija päättää saako hakija lainaa vai ei. Tarkemmin sanottuna kyseinen puumalli jakaa hakemukset hyväksynnän perusteella kolmeen luokkaan:

- A, hakemus hyväksytään.
- D, hakemus hylätään.
- M, hakemus vaatii manuaalikäsittelyn. Esimerkiksi tarvitaan lisätietoja.

Edellä kuvattu esimerkki havainnollistaa, kuinka päätöksentekopuuta voidaan käyttää luokittelumallina. Päätöksentekopuumalli on rakennettu opetusjoukon avulla, joka perustuu asiantuntijan tekemiin aiempiin päätöksiin. Tämän jälkeen opetettua algoritmia voidaan käyttää uuden hakemuksen luokittelemisen automatisoimiseksi.



Kuva 2.2: Päätöksentekopuu annettavista lainoista.

Päätöksentekopuu on hyvin läheistä sukua sääntöinduktiolle. Jokainen polku juuresta lehteen voidaan muuntaa säännöksi yhdistämällä polun eri vaiheet ennakkotapaukseksi ja muuntamalla lehden luokkaennuste luokan arvoksi.

2.3.1 Päätöksentekopuuoppiminen

Kuten aiemmin totesimme, ohjattu koneoppimismetodi on konstruoitu siten, että algoritmille syötetään opetusta varten lähtöjoukko, jonka perusteella muodostetaan funktio lähtöjoukon ja maalijoukon eli kohdemuuttujan välil-

le. Päämääränä on soveltaa tätä funktiota uuteen lähtöjoukkoon ja kyetä ennustamaan ennalta tuntemattomia tapahtumia.

Lähtöjoukon kuvaamiseen on useita eri tapoja. Eräs tapa kuvata lähtöjoukkoa on tietyn pyydysmallin (engl. *bag schema*) pyydysilmentymänä (engl. *bag instance*). Pyydysilmentymä on kokoelma monikkoja, joissa voi esiintyä kopioita. Kuvan 2.2 esimerkissä kyseessä olisi lainaa hakevien joukko. Jokainen monikko kuvataan vektorilla, joka edustaa muuttujien arvoja, kuten sukupuolta, elätettävien määrää ja niin edelleen. Pyydysmalli on kuvaus muuttujista ja muuttujien määrittelyalueista, esimerkiksi ikä, jonka määrittelyalue on 18-150 vuotta.

Tässä tutkielmassa käytämme notaatiota, että A on syötettyjen muuttujien joukko $A = \{a_1, ..., a_i, ..., a_n\}$, $i, n \in \mathbb{N}$ ja y edustaa luokkamuuttujaa tai kohdemuuttujaa. Syötetyt muuttujat ovat tyypillisesti nominaalisia tai numeerisia arvoja. Kuvan 2.2 esimerkissä joukko A olisi siis asiakkaista kerättyjen tietojen joukko, kuten ikä, sukupuoli, siviilisääty ja niin edelleen, ja y olisi päätös, saako henkilö lainan.

Ilmentymäavaruus X kuvataan syötettyjen muuttujien määrittelyalueiden karteesisena tulona

$$X = dom(a_1) \times dom(a_2) \times ... \times dom(a_n), \tag{2.1}$$

missä $dom(a_i)$ edustaa muuttujan määrittelyjoukon mahdollisia arvoja. Esimerkiksi olkoon syötemuuttuja a_1 lainaa hakevan henkilön ikä. Määrittelyalue $dom(a_1)$ on näin ollen ikävuodet 18-150 vuotta. Universaali ilmentymäavaruus U määritellään taas ilmentymäavaruuden X ja kohdemuuttujan määrittelyalueen $dom(y)$ karteesisena tulona:

$$U = X \times dom(y) \tag{2.2}$$

Lähtöjoukko on pyydysilmentymä, joka sisältää monikkojoukon. Formaalisti lähtöjoukot määritellään $S(U) = (\langle x_1, y_1 \rangle, ..., \langle x_n, y_n \rangle)$, missä $x_q \in X$ ja $y_q \in dom(y)$, $q \in \mathbb{N}$. Yleisesti oletetaan, että lähtöjoukon monikot muodostetaan sattumanvaraisesti ja itsenäisesti kiinnitetyllä todennäköisyydellä D yli joukon U. Tämä on yleistys deterministisestä tapauksesta, jossa monikko määritellään funktiolla $y = f(x)$.

Konseptioppiminen on metodi, jolla esimerkkijoukon jäsenten perusteella päätellään ryhmän yleinen määritelmä. KDD-yhteisö (*Knowledge Discovery in Databases*) laajentaa tämän käsitteen luokitteluongelmaksi. Tässä tapauksessa etsimme funktiota, joka kuvaa kaikkien mahdollisten tapausten joukon

ennalta määrättyihin luokkiin, jotka eivät ole binäärisiä. Algoritmia, joka suorittaa tämän toimenpiteen, kutsumme luokkaindusoijaksi.

Luokkaindusoija on itsenäinen kokonaisuus, joka hallinnoi opetusjoukkoa ja muodostaa funktion syötemuuttujien ja kohdemuuttujien välille. Oletetaan, että meillä on joukko A, joka on syötemuuttujien määrittelyalueiden karteesinen tulo eli $A = dom(a_1) \times dom(a_2) \times ... \times dom(a_n)$. Olkoon T kohdemuuttujan y määrittelyalue, $T = dom(y)$. Olkoon nyt joukko S opetusjoukko, joka määritellään joukkojen A ja T karteesisena tulona, $S = A \times T$.

Olkoon D tuntematon kiinnitetty todennäköisyysjakauma opetusjoukon S yli. Tavoitteena on määritellä minimiyleistysvirhe optimaaliselle luokittelumetodille.

Yleistysvirhe (engl. *generalization error*) määritellään luokitteluvirheasteena yli todennäköisyysjakauman D. Nominaalisen muuttujan tapauksessa voimme määritellä virheen seuraavasti:

$$\epsilon(DT(S), D) = \sum_{\langle x,y \rangle \in U} D(x, y) \cdot L(y, DT(S(x))), \tag{2.3}$$

missä $L(y, DT(S)(x))$ on määritelty seuraavasti:

$$L(y, DT(S)(x)) = \begin{cases} 0, \text{ jos } y = DT(S)(x) \\ 1, \text{ jos } y \neq DT(S)(x) \end{cases} \tag{2.4}$$

Notaatio DT kuvaa päätöksentekopuun indusoijaa ja $DT(S)$ kuvaa luokittelupuuta, joka muodostuu, kun päätöksentekopuu DT opetetaan opetusjoukon S yli. Luokittelupuun $DT(S)$ avulla voimme ennustaa kohdemuuttujan arvon monikolle x_q, $q \in \mathbb{N}$, joka on siis $DT(S)(x_q)$. Numeeristen muuttujien tapauksessa summafunktio korvataan integraalilla.

Luokittelupuuta voidaan käyttää joko määräämällä luokittelupuu eksplisiittiseen lähtöjoukkoon tai varustamalla luokittelupuu todennäköisyysvektorilla, joka edustaa jokaisen luokan ehdollisia todennäköisyyksiä. Tälläisiä indusoijia kutsutaan todennäköisyysindusoijiksi. .

2.3.2 Satunnaismuuttujista

Seuraavaksi käsittelemme todennäköisyysindusoijien ja kappaleen 4.3 kannalta olennaisia määritelmiä liittyen satunnaismuuttujiin.

Olkoon X satunnaismuuttuja. Diskreetillä satunnaismuuttujalla X on arvojoukko $\mathcal{X} = x_0, x_1, ...$, joka edustaa satunnaismuuttujan X mahdollisia arvoja. Arvojoukko voi olla äärellinen tai ääretön.

Merkitsemme $(X = x_i)$, kun diskreetillä satunnaismuuttujalla ilmenee jokin arvo. Tällaisella tapahtumalla on todennäköisyys $P(\mathcal{X} = x_i) = p_i$. Nämä todennäköisyydet määräävät diskreetin satunnaismuuttujan pistetodennäköisyysfunktion, joka kuvaa satunnaismuuttujan jakaumaa. Olettaen, että diskreetin satunnaismuuttujan X arvojoukko on K, todennäköisyysfunktiolla $p(k) = P(X = k)$ on seuraavat ominaisuudet:

$$0 \leq p(k) \leq 1, k \in K \qquad \text{ja} \qquad \sum_{k \in K} p(k) = 1. \tag{2.5}$$

Diskreetin satunnaismuuttujan X kertymäfunktio saadaan summaamalla

$$F(x) = \sum_{k \leq x} p(k) \tag{2.6}$$

ja tälle pätee $\lim_{x \to -\infty} F(x) = 0$, $\lim_{x \to \infty} F(x) = 1$ ja $F(x)$ on ei-vähenevä.

Määritelmä 2.1. Diskreetin satunnaismuuttujan *odotusarvo* on

$$E(X) = \sum_{k \in K} k \cdot p(k), \tag{2.7}$$

jos summa suppenee itseisesti eli $\sum_{k \in K} |k| \cdot p(k) < \infty$.

Jos satunnaismuuttujia on kaksi, on kyse satunnaismuuttujaparista X, Y. Useampiulotteisessa tilanteessa puhutaan satunnaisvektorista $(X_0, X_1, ..., X_n)$. Tapahtumat $(X = x)$ ja $(Y = y)$ voivat vaikuttaa toisiinsa tavalla tai toisella. Tätä yhteisvaikutusta voidaan kuvata ehdollisella todennäköisyydellä ja riippumattomuudella.

Määritelmä 2.2. Oletetaan, että pätee $P(X) > 0$. Tapauksen Y *ehdollinen todennäköisyys* ehdolla X on

$$P(Y|X) = \frac{P(X \cap Y)}{P(X)}. \tag{2.8}$$

Määritelmä 2.3. Satunnaismuuttujat X ja Y ovat toisistaan riippumattomia jos ja vain jos $P(X = x \cap Y = y) = P(X = x)P(Y = y)$ kaikilla $x \in \mathcal{X}$ ja $y \in \mathcal{Y}$.

Näin ollen riippumattomien satunnaismuuttujien tapauksessa $P(Y = y|X = x) = P(Y = y)$ kaikille $x \in X$.

Määrittelemme vielä kaksi ehdollisiin jakaumiin liittyvää käsitettä.

Määritelmä 2.4. Satunnaismuuttujan Y *ehdollinen odotusarvo* ehdolla $X = x$ on

$$E(Y|X = x) = \sum_{y \in \mathcal{Y}} yP(Y = y|X = x). \tag{2.9}$$

Määritelmä 2.5. Satunnaismuuttujan Y *marginaalijakauma* on

$$P(Y = y) = \sum_{x \in X} P(Y = y \cap X = x).$$

Päätöksentekopuiden tapauksessa kaikille lehdille on mahdollista arvioida ehdollisia todennäköisyyksiä laskemalla lehden luokan esiintyvyys kaikissa opetusjoukoissa. Tällä tavalla arvioitu todennäköisyysvektori yleensä arvioi esiintyvyyden liian suureksi. Tämä voi muodostua ongelmaksi erityisesti silloin, kun annetun luokan esiintyvyys tietyssä lehdessä on nolla. Tällöin saamme tulokseksi, että kyseisessä puussa kokonaistodennäköisyys luokan esiintyvyydelle on nolla. Kyseistä ilmiötä voi korjata joko Laplace- tai No Match-korjauksella, joista voi lukea lisätietoa Rokachin ja Maimonin kirjasta [4].

2.3.3 Päätöksentekopuualgoritmi

Päätöksentekopuuindusoijat ovat algoritmeja, jotka konstruoivat annetusta datajoukosta päätöksentekopuun. Tyypillisesti päämääränä on löytää optimaalinen puu minimoimalla yleistysvirhe, kappale 3.1. Toinen tapa on minimoida solmujen määrä tai lehtien keskimääräinen syvyys. Kyseessä olevaa induktiivista menetelmää pidetään yleisesti ottaen haastavana tehtävänä. Käymme lyhyesti läpi laskennallisesti haastavien ongelmien eli NP-luokan (*Nondeterministic Polynomial time*) terminologiaa.

NP on kompleksisuusluokka, joka sisältää ongelmia, jotka ratkeavat epädeterministisellä Turingin koneella polynominaalisessa ajassa. NP-kovat ongelmat ovat ongelmia, joihin kaikki NP-ongelmat voidaan tarvittaessa redusoi-

da polynomiajassa, mutta jotka eivät välttämättä kuulu NP-luokkaan. NP-täydelliset ongelmat ovat NP-kovia ongelmia, jotka kuuluvat NP-luokkaan. [5]

Hancock ym. [6] ovat osoittaneet, että minimipuun löytäminen annetusta datajoukosta, joka olisi konsistentti opetusjoukon kanssa, on NP-kova ongelma. Hyavil ja Rivest [7] ovat puolestaan osoittaneet, että binäärisen minimaalisen puun konstruoiminen, joka pystyisi riittävän tarkasti luokittelemaan ennalta tuntemattoman joukon, on NP-täydellinen ongelma. Tiedetään myös, että annetun puun minimiekvivalenssipuun löytäminen tai päätöstauluista optimaalisen puun rakentaminen ovat NP-kovia ongelmia. Nämä tulokset viittaavat siihen, että optimaalisten päätöksentekopuun algoritmien käyttö on mielekästä vain suhteellisen pienten joukkojen kohdalla. Tämän vuoksi ongelman ratkaisemiseen käytetään heuristisia menetelmiä. Karkeasti ottaen näitä algoritmeja on kahta tyyppiä, ylhäältä alas ja alhaalta ylös, joista ensin mainittua on kommentoitu kirjallisuudessa enemmän.

Ylhäältä alas kasvavat algoritmit ovat luonnostaan ahneita (engl. *greedy*) ja kasvattavat puuta ylhäältä-alas rekursiivisesti. Jokaisessa iteraatiossa algoritmi tutkii lähtöjoukon osallisuutta diskreettien syötemuuttujien arvojen avulla. Merkittävimmät muuttujat valitaan jakamismetodin tulosten perusteella. Validin muuttujan valinnan jälkeen algoritmi toistaa jakamismetodin. Algoritmi jakaa joukkoa alajoukkoihin, kunnes pysähtymiskriteeri täyttyy. Seuraavat kriteerit ovat tyypillisimpiä pysähtymiskriteereitä:

(1) Harjoitusjoukon kaikki alkiot toistavat samaa arvoa y.
(2) Puun maksimisyvyys on saavutettu.
(3) Tapausten lukumäärä viimeisessä solmussa on vähemmän kuin isäsolmun minimivaatimus.
(4) Jos solmu pitäisi jakaa, tulevien lapsisolmujen tapausten lukumäärä olisi pienempi kuin minimivaatimus.
(5) Paras jakamiskriteeri ei täytä kynnysehtoa.

Olemme käsitelleet päätöksentekopuumallin oppimista, rakentamista ja algoritmia. Seuraavaksi esittelemme, miten tiedonlouhintamenetelmien suorituskykyä on mahdollista arvioida.

Luku 3

Analysointimenetelmien validointi

Empiirinen havainnointi on osoittanut, että jokainen tiedonlouhintalähestymistapa tuottaa parhaan tuloksen joissakin, muttei kaikissa tapauksissa [4]. Tarkkuuden arvioiminen ei ole aina triviaalia ja tarkkuus ei ole ainoa kriteeri kuvaamaan mallin paremmuutta. Esimerkiksi neuroverkko saattaa tuottaa tarkempia tuloksia kuin päätöksentekopuu, mutta päätöksentekopuiden tuloksia pidetään yleisesti ottaen ymmärrettävämpinä. Lisäksi *Accuracy paradox* sanoo, että ennustavilla malleilla, joilla on tietty tarkkuusarvo, saattaa olla suurempi ennustuskyky kuin malleilla, joilla on suurempi tarkkuus. Näin ollen tiedonlouhinnassa analysointimenetelmien validoiminen on yhtä tärkeää kuin itse analysoiminen.

Tässä kappaleessa esittelemme tutkielman kannalta päätöksentekopuiden tärkeimmät validointimenetelmät. Päätöksentekopuun toimivuuden analysoiminen on koneoppimisen keskeinen osa-alue. Kuten aiemmin on todettu, päätöksentekopuuindusoija saa syötteenä lähtöjoukon, jonka perusteella algoritmi rakentaa puun, jolla voidaan luokitella uusi syötejoukko. Sekä indusoija että itse puu tulee arvioida. Validointi on tärkeää puun laadun ymmärtämisen ja parametrien hienosäädön kannalta.

3.1 Yleistysvirhe

Oletetaan, että $DT(S)$ edustaa luokittelupuuta, joka on kasvatettu lähtöjoukon S perusteella. Luokittelupuun $DT(S)$ yleistysvirhe on todennäköisyys, jolla jakaumaa D edustava puu luokittelee joukon jäseniä väärin. Luokittelutarkkuus on arvo, joka saadaan vähentämällä yleistysvirhe luvusta yksi. Myöhemmin tässä tutkielmassa viittaamme luokittelutarkkuuteen lyhyesti termillä *tarkkuus*.

Opetusvirhe (engl. *training error*) määritellään oikein luokiteltujen alkioi-

den osuutena koko joukosta, formaalisti ilmaistuna:

$$\hat{\epsilon}(DT(S), S) = \sum_{\langle x,y\rangle \in U} L(y, DT(S(x))), \tag{3.1}$$

missä $L(y, DT(S)(x)$ on kohdassa 2.4 määritelty nolla-yksi menetys.

Vaikka yleistysvirhe on luonnollinen kriteeri arvioida mallin tarkkuutta, yleensä sen todellinen arvo tunnetaan vain synteettisissä tapauksissa. Tämä johtuu siitä, että ilmentymäavaruuden jakaumaa D ei yleisesti ottaen ole tiedossa. Joissain tapauksissa on mielekästä arvioida yleistysvirhe opetusvirheen avulla, mutta yleensä tämä arvio on ylisovittamisesta johtuen vääristynyt ja liian optimistinen arvio mallin tarkkuudesta. Yleistysvirhettä voidaan arvioida teoreettisesti tai empiirisesti.

3.1.1 Yli- ja alisovittaminen

Ylisovittamisella (engl. *overfitting*) viitataan tilanteeseen, jossa induktioalgoritmi kehittää luokittelijan, joka sopii täydellisesti opetusjoukkoon ja luokittelee kyseisen joukon onnistuneesti, mutta joka ei kykene yleistämään luokittelusääntöä tapauksiin, jotka eivät esiinny opetusjoukossa. Toisin sanoen oppimisen sijaan algoritmi opettelee ulkoa opetusalkioiden suhteen kohdemuuttujaan. Alisovittamisella (engl. *underfitting*) viitataan tilanteisiin, joissa algoritmi ei selitä opetusjoukkoa riittävällä tarkkuudella. Tämä yleensä ratkaistaan parantamalla luokittelijaa.

Päätöksentekopuissa ylisovittumista tapahtuu yleensä, kun puulla on liian monta solmua suhteessa opetusjoukkoon ja lehtiin kerääntyvät joukot ovat hyvin pieniä. Solmujen lukumäärää kasvattamalla opetusvirhe yleensä pienenee, mutta sitä vastoin yleistysvirhe kasvaa. Ylisovittamisen on osoitettu vähentävän puun ennustuskyvykkyyttä [8]. Ylisovittamisen välttämiseksi on mahdollista joko välttää tilastollisesti riittämättömiä jakamissääntöjä tai karsimalla tarpeettomat solmut jälkikäteen.

3.1.2 Yleistysvirheen teoreettinen arvioiminen

Pieni opetusvirhe ei takaa pientä yleistysvirhettä. Yleistysvirheissä on huomattu poikkeamaa, kun yleistysvirhe on ennustettu opetusvirheestä lisäämällä luottamusvälit verrattuna todelliseen yleistysvirheeseen. Indusoijan kapasiteetti määrittää opetusvirheen luottamusvälin asteen. Yleisesti indusoijan kapasiteetti ilmaisee sen kykyä muodostaa moninaisia luokittelijoita. Päätöksentekopuut, joissa on monta solmua suhteessa lähtöjoukkoon, luokittelevat

todennäköisesti niin, että muodostuva opetusvirhe on pieni. Tämä yleensä johtuu algoritmin kyvystä oppia attribuutit ulkoa, eikä kyvystä muodostaa funktio syöte- ja kohdemuuttujien välillä, jolloin sen yleistämiskyky on heikko. Tällaisissa tapauksissa pieni opetusvirhe merkitsee suurta yleistysvirhettä. Kun päinvastainen tapahtuu eli indusoijan kapasiteetti on liian pieni suhteessa joukon kokoon, indusoija saattaa alisovittua, jolloin sekä opetus- että yleistysvirhe ovat suuria.

3.1.3 Yleistysvirheen empiirinen arvioiminen

Toinen tapa arvioida yleistysvirhettä on jakaa käytetty joukko satunnaisesti kahdeksi joukoksi; opetus- ja testijoukoksi. Yleensä 2/3 joukosta määritetään opetusjoukoksi ja loput allokoidaan testijoukoksi. Tässä indusoija luo luokittelijan opetusjoukon avulla, jonka väärinluokitteluvirhe (engl. *misclassification error*) lasketaan testijoukossa. Testijoukosta saatu väärinluokitteluvirhe on yleensä tarkempi arvio yleistysvirheestä kuin mitä opetusvirheen avulla olisi voinut arvioida. Toisaalta koska mallin luomiseen käytetään vain osaa tietojoukosta, tarkkuusarvio saattaa olla turhan pessimistinen.

Joukon ollessa pieni sitä voidaan jakaa usealla eri metodilla. On hyvin tyypillistä, että joukko jaetaan opetus- ja testijoukoiksi useammalla eri tavalla ja indusoijat harjoitetaan ja testataan jokaisen joukolla ja tulokset keskiarvoistetaan. Tällä tavalla saadaan luotettavampi arvio yleistyvirheestä.

Yleisimmät tavat ottaa uudet näytteet käytetystä joukosta ovat satunnaisnäytteistäminen ja n-kertainen ristiinvalidointi. Satunnaisessa näytteistämisessä joukko jaetaan useampaan kertaan epäjatkuviksi opetus- ja testijoukoksi. Jokaisesta osituksesta saadut virheet keskiarvoistetaan. N-kertaisessa ristiinvalidoinnissa käytetty joukko jaetaan satunnaisesti osapuilleen n:ksi yhtä suuriksi ja yhtä eksklusiivisiksi alajoukoiksi. Indusoijaa harjoitetaan ja arvioidaan n kertaa: joka kerta indusoija harjoitetaan jollakin alajoukolla indeksillä k ja arvioidaan lopuilla alajoukoilla, joiden lukumäärä on n-1. Tämä toimenpide toistetaan, kunnes indusoijaa on harjoitettu jokaisella eksklusiivisella alajoukolla.

Satunnaisen näytteistämisen etu on siinä, että se voidaan toistaa mielivaltaisen monta kertaa, mutta menetelmän haittapuoli on, että testijoukot eivät muodostu itsenäisesti suhteessa alkuperäisen joukon jakaumaan. Tästä johtuen I-tyypin virheen esiintyvyys kasvaa. I-tyypin virheellä viitataan tilanteeseen, jossa algoritmi saattaa löytää merkittävän eron muuttujien välillä, vaikka eroa ei todellisuudessa ole. Toistamalla metodia I-tyypin virheen muodostumista voidaan hillitä, mutta näin ei välttämättä saada stabiilia arviota yleistysvirheestä.

Ristiinvalidointimetodilla saatu yleistyvirhe saadaan jakamalla väärin luo-

kiteltujen alkioiden määrä koko joukon lukumäärällä. Stabiilin yleistyvirheen tuottamiseksi on hyvin tyypillistä, että ristiinvalidoinnissa validointi toistetaan n kertaa. Tässä toistuu sama virhe eli testijoukoista tulee epäitsenäisiä ja I-tyypin virheen esiintyvyys kasvaa. Tähän ongelmaan ei valitettavasti olla löydetty vielä tyydyttävää ratkaisua. Vaihtelevat testit [9] ovat osoittaneet, että sitä mukaan kun I-tyypin virheet saadaan minimoitua, II-tyypin virheiden määrä kasvaa. II-tyypin virheellä viitataan tilanteeseen, jossa algoritmi ei tunnista merkittävää eroa alkioiden välillä, vaikka joukossa esiintyy sellainen.

Sekä n-kertainen ristiinvalidointi ja satunnainen näytteistäminen osittavat otannan. Ositus varmistaa, että alkuperäisen joukon jakauma säilyy kaikissa opetus- ja testijoukoissa. Osituksen on osoitettu vähentävän arvioidun virheen varianssia, erityisesti monen luokan joukoissa. [4]

3.2 Muita validointimenetelmiä

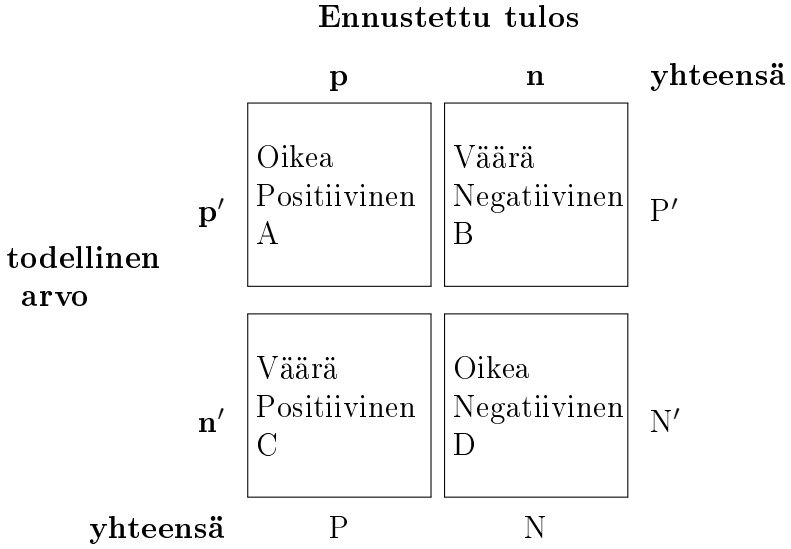
Yleistysvirheen suuruus ja mallin tarkkuus eivät ole välttämättä riittäviä arvioimaan mallin suorituskykyä, esimerkiksi mikäli käytetyn joukon jakauma on epätasapainoinen, käytettävissä oleva joukko on rajoittunut tai tarkkuuden sijasta mallin ymmärrettävyys on korkeampi prioriteetti. Tämän vuoksi arvioimismenetelmiä on kehitetty valtava määrä vastaamaan erilaisia tarpeita, ja näistä käymme läpi tässä tutkielmassa väärinluokittelumatriisin (engl. *misclassification matrix*), ROC-käyrän ja nostokäyrän (engl. *lift curve*).

3.2.1 Väärinluokittelumatriisi

Väärinluokittelumatriisi kuvaa luokittelijan kykyä luokitella joukon alkiot oikeisiin luokkiin. Oletetaan, että meillä on lähtöjoukko, jonka alkiot on tarkoitus jakaa kahteen luokkaan, toisin sanoen kohdemuuttuja on binäärinen. Toinen luokista määritellään halutuksi luokaksi. Oikea positiivinen (engl. *true positive*) kuvaa haluttuun luokkaan kuuluvia alkioita, jotka on luokiteltu oikein; oikea negatiivinen (engl. *true negative*) luokkaan kuulumattomia alkioita, jotka on luokiteltu oikein; väärä negatiivinen (engl. *false negative*) luokkaan kuuluvia, jotka on luokiteltu virheellisesti; väärä positiivinen (engl. *false positive*) luokkaan kuulumattomia, jotka on luokiteltu virheellisesti.

Taulukossa 3.1 nähtävien arvojen perusteella voimme laskea:
-Tarkkuuden: $A + D/(A + B + C + D)$
-Väärinluokitteluasteen: $B + C/(A + B + C + D)$
-Täsmällisyyden: $D/(B + D)$

-Oikean positiivisuusasteen eli herkkyyden (engl. *sensitivity*): $A/(A+B)$
-Väärän positiivisuusasteen eli pudonneet (engl. *fall-out*, 1-spesifisyys): $C/(C+D)$
-Oikean negatiivisuusasteen eli spesifisyyden (engl. *specificity*): $D/(C+D)$
-Väärän negatiivisuusasteen:$B/(A+B)$



Kuva 3.1: Väärinluokittelumatriisi.

Se, mikä arvo on olennaisinta, riippuu analysointitehtävästä. Esimerkiksi mainoskampanjassa, jossa pyritään maksimoimaan ostettujen tuotteiden määrä eli positiivisten vastausten määrän, oikea positiivisuusaste on olennaisin mitta kuvaamaan mallin paremmuutta. Mikäli kustannukset ovat merkittävät tai virheiden välttäminen tärkeää, väärä positiivisuusaste on olennaisempi mitta.

Hämmennysmatriisi (engl. *confusion matrix*) kuvaa väärinluokittelumatriisin tavoin oikein ja väärin luokiteltuja alkioita silloin, kun luokkia on enemmän kuin kaksi. Esimerkiksi oletetaan, että meillä on algoritmi, joka on harjoitettu luokittelemaan pankin 27 asiakkaan joukosta toivotut, neutraalit ja epätoivotut asiakkaat oikeisiin luokkiin. Toivottuja asiakkaita on kahdeksan, epätoivottuja 13 ja neutraaleja kuusi. Tuloksena syntyvä hämmennysmatriisi voisi näyttää kuvan 3.2 mukaiselta.

Tässä esimerkissä algoritmi olisi luokitellut viisi toivottua, kolme neutraalia ja 11 epätoivottua oikein, kaksi epätoivottua neutraaliksi, kaksi neutraalia toivotuksi ja yksi epätoivotuksi ja kolme toivottua neutraaliksi. Esimerkistä näkee, että algoritmi luokittelee epätoivotut asiakkaat onnistuneesti oikean

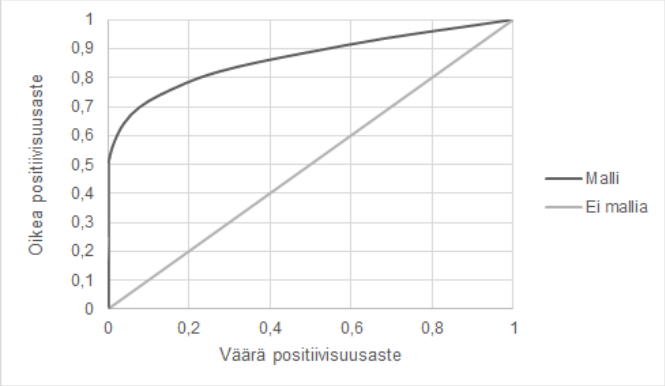
		Ennustettu		
		Toivottu	Neutraali	Epätoivottu
Todellinen	Toivottu	5	2	0
	Neutraali	3	3	2
	Epätoivottu	0	1	11

Kuva 3.2: Esimerkki hämmennysmatriisista.

positiivisuusasteen ollessa 0.85, kun taas toivottujen ja neutraalien asiakkaiden erottaminen on haasteellisempaa. (Toivottujen asiakkaiden oikea positiivisuusaste on 0.625 ja neutraalien 0.5.)

3.2.2 ROC-käyrä ja AUC

ROC-käyrä (*Receiver Operating Characteristic*) on analysointimenetelmä, joka visualisoi binääriluokittelijan oikea positiivisuus- ja väärä positiivisuusasteiden välistä vaihtuvuutta [10]. ROC-käyrä lasketaan tutkimalla mallin oikea positiivisuus- ja väärä positiivisuusasteiden arvoja erilaisilla kynnysehdoilla.



Kuva 3.3: Esimerkki ROC-käyrästä.

ROC-käyrän alle jäävää pinta-alaa kutsutaan AUC:ksi (engl. *Area Under the ROC curve*) ja tämän pinta-alan suuruus implikoi mallin tarkkuutta. Täydellisen mallin pinta-ala on yksi ja hyödyttömän mallin pinta-ala puolikas.

3.2.3 Nostokäyrä

Nosto (engl. *lift*) on mitta, joka kuvaa mallin ennustuskykyä. Nosto lasketaan mallin ja ilman mallia saatavien tulosten suhteena. Näiden suhteiden avulla voimme laskea kumulatiiviset saanti-(engl.*cumulative gain*) ja nostokäyrät, jotka visualisoivat mallin suorituskykyä. Otetaan havainnollistava esimerkki.[1]

Yritys haluaa tehdä suoramarkkintointikampanjan kirjeitse. Jokainen kirje maksaa yritykselle euron ja yrityksellä on tietoja 100 000:sta asiakkaasta. Aiemmin on havaittu, että lähettämällä kaikille asiakkaille mainoskirje, 20 000 on ostanut tuotteen eli positiiviseksi vastausprosentiksi saadaan 20%.

Oletetaan, että olemme rakentaneet mallin, joka ennustaa ketkä asiakkaista

Hinta(€)	Asiakkaiden määrä	Positiivisia vastauksia
100 000	100 000	20 000

Taulukko 3.1: Esimerkki.

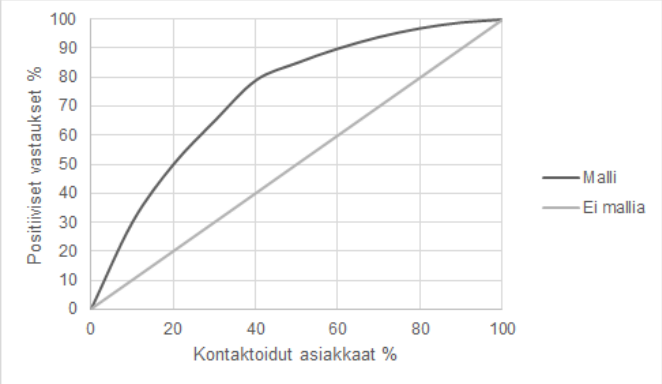
mainoksen ansiosta todennäköisimmin ostavat tuotteen. Mallin avulla voimme laskea ennustetut positiiviset vastaukset, kun otamme yhteyttä 10 000:n ensimmäiseen asiakkaaseen, 20 000:n ja niin edelleen. Tämän jälkeen voim-

Hinta(€)	Asiakkaiden määrä	Positiivisia vastauksia
10 000	10 000	6 000
20 000	20 000	10 000
30 000	30 000	13 000
40 000	40 000	15 800
50 000	50 000	17 000
60 000	60 000	18 000
70 000	70 000	18 800
80 000	80 000	19 400
90 000	90 000	19 800
100 000	100 000	20 000

Taulukko 3.2: Esimerkki.

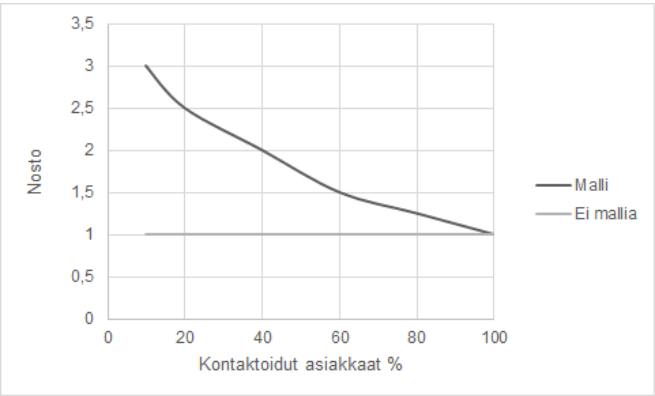
[1] http://www2.cs.uregina.ca/

me laskea kumulatiivisen saantikäyrän. Kuvassa 3.4 näkyvä käyrä on laskettu siten, että x-akselilla on positiivisten vastausten prosentuaalinen osuus ja y-akselilla mainoksen saaneiden asiakkaiden prosentuaalinen osuus. Vertailulinjana on arvio, että jos lähetämme X%:lle asiakkaista mainoksen, niin saamme X% osuuden positiivisista vastauksista.



Kuva 3.4: Esimerkki kumulatiivisesta saantikäyrästä.

Nostokäyrästä näemme, kuinka moninkertaisen positiivisen vastauksen saamme mallin avulla verrattuna siihen, että emme käytä mallia. Tämä lasketaan kumulatiivisen saantikäyrän avulla. Kun kontaktoimme 10% asiakasjoukosta ilman mallia, tavoitamme 10% positiivisista vastauksista. Mallin avulla lähettämällä mainoksen valikoidulle 10%:lle asiakasjoukosta saavutamme 30% positiivisista vastauksista. Nosto lasketaan näiden lukujen suhteena $30/10 = 3$ (Kuva 3.5).



Kuva 3.5: Esimerkki nostokäyrästä.

Luku 4

Satunnaismetsä

Satunnaismetsä on joukko-oppimista soveltava metodi, jossa yksinkertaisena algoritmikomponenttina on päätöksentekopuu. Satunnaismetsä on olennaisesti yhdistelmä bagging-menetelmää, jonka esittelemme myöhemmin, ja satunnaisen attribuuttijoukon valintaa. Sen perustava idea on luoda suuri joukko päätöksentekopuita, joiden varianssi on kontrolloitu. [11] Satunnaismetsä ideana on suosittu koneoppimismenetelmä luokittelu- ja regressio-ongelmiin sen tarkkuuden ja kohtuullisen yksinkertaisen algoritmin ansiosta. Satunnaismetsää sovelletaan myös muunlaisiin tehtäviin osana datalouhintaa, esimerkiksi datajoukon merkittävien muuttujien analysoimiseen.

Tässä kappaleessa aloitamme joukko-oppimisesta ja joukko-oppimismetodeista, jonka jälkeen käsittelemme satunnaismetsän algoritmia.

4.1 Joukko-oppiminen

Joukko-oppiminen (engl. *ensemble learning*) on tilastotieteessä ja koneoppimisessa käytetty menetelmä, jonka ideana on yhdistää monta yksinkertaista ja potentiaalisesti heikompaa algoritmia yhdeksi metodiksi [9]. Näin useasta heikosti ennustavasta metodista tehdään kokonaisuus, jonka suorituskyky on parempi kuin yksittäisen komponentin.

Joukko-oppimista toteuttavia algoritmeja on useita erilaisia, mutta algoritmin rakentamiseen on pääasiallisesti kaksi lähestymistapaa. Ensimmäinen lähestymistapa on luoda joukko, joka rakentaa hypoteesipareja. Äänestettäessä tällainen painotus antaa mallille hyvän sovituksen käytettävään dataan. Toinen lähestymistapa on konstruoida yksittäinen hypoteesi itsenäisesti siten, että lopullinen hypoteesien joukko on täsmällinen ja monimuotoinen. Näin ollen uusia ennustuksia tehtäessä yksittäisten hypoteesien virhemargi-

naali on kohtalaisen pieni ja lopullisessa hypoteesijoukossa vaihtelevaisuus on kohtalaisen suurta. Tällaisen joukon tarkkuus on parempi kuin yksittäisen komponentin, koska äänestettäessä mahdolliset erimielisyydet kumoavat toisensa.

4.1.1 Joukko-oppimismenetelmät

Luokittelemisen tarkkuuden parantamiseksi on luotu useita joukko-oppimismenetelmiä, jotka hyödyntävät puujoukkoja. Eräs tällainen metodi on niin kutsuttu bagging-metodi. Bagging konstruoidaan kasvattamalla joukko puita, jotka äänestävät käsillä olevan syötejoukon suosituimmasta luokasta. Tällaisen joukon kasvattamiseksi määritellään opetusjoukosta satunnaisia vektoreita, jotka ohjaavat yksittäisen puun kasvamista.

Toinen esimerkki joukko-oppimismenetelmistä on halkomissäännön satunnainen valinta. Tällaisessa metodissa jokainen solmu halkaistaan säännöllä, joka on valittu satunnaisesti joukosta K, missä joukko K on kokoelma parhaita halkomissääntöjä. Lisää esimerkkejä joukko-oppimismetodeista löytyy Dietterichin teoksesta [9].

Yhteistä näille metodeille on, että käyttämällä samaa opetusjoukkoa puulle indeksillä k, $k \in \mathbb{N}$, luodaan satunnainen vektori Θ_k, joka on riippumaton satunnaisista vektoreista $\Theta_1, ..., \Theta_{k-1}$, saman jakauman yli. Puu siis kasvaa opetusjoukon ja vektorin Θ_k perusteella, tuottaen luokittelijan $h(x, \Theta_k)$, missä x on syötevektori. Esimerkiksi bagging-metodissa satunnainen vektori Θ luodaan N määrästä osumia laatikoihin, kun N määrä tikkoja heitetään satunnaisesti, ja N on opetusjoukon muuttujien määrä. Halkomissäännön satunnaistamisessa Θ kuvaa muista riippumattomia satunnaisia kokonaislukuja välillä $[1, K]$. Vektorin Θ luonne ja ulottuvuus riippuu puukonstruktiosta. Kun on luotu riittävä määrä puita, puut äänestävät suosituimmasta luokasta.

4.2 Satunnaismetsän algoritmi

Määritelmä 4.1. Satunnaismetsä on puustruktuurisista luokittelijoista koostuva joukko $h(x, \Theta_k), k = 1, ...,$ missä Θ_k ovat itsenäisiä identtisesti luotuja satunnaisvektoreita ja jokainen puu äänestää yksikkönä suosituinta luokkaa syötteelle x. [11]

Yksittäinen puu satunnaismetsässä luodaan seuraavalla algoritmilla:

Vaatimukset: Päätöksentekopuun indusoija IDT, iteraatioiden lukumäärä T, opetusjoukko S, alajoukon koko μ ja jokaisessa solmussa olevien

attribuuttien lukumäärä N.

Varmista: $M_t; t = 1, ..., T$
$1 : t \rightarrow 1$
$2 : toista$
$3 : S_t \leftarrow$ Erottele μ määrä ilmentymiä joukosta S korvaamista varten.
$4 :$ Rakenna luokittelija M_t hyödyntäen indusoijia $IDT(N)$ joukosta S.
$5 : t++$
$6 :$ Toista, kunnes $t > T$

Satunnaismetsän menetelmä voidaan määritellä muillakin tavoin ja menetelmää voidaan soveltaa muihinkin kuin päätöksentekopuihin, mutta emme paneudu niihin tässä tutkielmassa.

Usean luokittelijan joukko on hyödyllinen vain siinä tapauksessa, että yksittäisten luokittelijoiden tulokset vaihtelevat samalla lähtöjoukolla [12]. Joukko-oppimismetodin suurin haaste onkin luoda yksittäisiä luokittelijoita, joiden lopputulosten varianssi on suuri, mutta jotka ovat opetusjoukon suhteen yhä johdonmukaisia. Empiiriset testit ovat osoittaneet, että mikäli nämä ehdot täyttyvät, tuloksena on virheitä, jotka eivät korreloi ja luokittelijajoukon tarkkuus kasvaa [13].

4.3 Satunnaismetsän yleistysvirhe

Kuten aiemmin olemme todenneet, eräs tapa arvioida mallin hvyyttä on yleistysvirheen suuruus. Päätöksentekopuiden tapauksessa vaarana on, että puut ylisovittuvat opetusjoukkoon, ja puiden lukumäärän kasvaessa ylisovitus kasvaa, jolloin yleistysvirhe suurenee. Satunnaismetsällä ei kuitenkaan käy näin, vaan puiden lukumäärän kasvaessa satunnaismetsä luo yleistysvirheelle ylärajan ja yleistysvirhe suppenee kohti nollaa. Osoitetaan seuraavaksi, että satunnaismetsän yleistysvirhe suppenee.

Oletetaan, että meillä on luokittelijoiden joukko $h_1(x), h_2(x), ..., h_k(x)$, $k \in \mathbb{N}$ ja että opetusjoukko on koostettu satunnaisesti satunnaisten vektorien $\mathbf{Y}$, $\mathbf{X}$ jakaumasta. Tällöin *marginaalifunktio* määritellään seuraavasti:

$$mg(X, Y) = av_k I(h_k(X) = Y) - \max_{j \neq Y} av_k I(h_k(X) = j), \tag{4.1}$$

missä $I(\cdot)$ on indikaattorifunktio ja av_k äänien keskiarvofunktio. Marginaali mittaa, kuinka kaukana keskimääräinen määrä ääniä joukoissa $\mathbf{X}$,$\mathbf{Y}$ oikealle luokalle eli $h_k(X) = Y$ ylittää keskimääräisen äänen mille tahansa muulle

luokalle eli $h_k(X) = j, j \neq Y$. Mitä suurempi marginaali on, sitä suurempi luottamustaso luokalla on. Yleistysvirhe määritellään seuraavasti:

$$PE^* = P_{X,Y}(mg(X,Y) < 0), \tag{4.2}$$

missä **X,Y** alaindeksi merkitsee, että todennäköisyys on yli avaruuden $X \times Y$. Tässä tutkielmassa seuraavat notaatiot ovat ekvivalentit, $h_k(\mathbf{X}) = h(\mathbf{X}, \Theta_k)$. Suurelle joukolle puustruktuurisia luokittelijoita seuraa suurien lukujen vahvan lain mukaan:

Teoreema 4.2. *Kun puiden määrä kasvaa, lähes kaikki jonot* $\Theta_1, ..., PE^*$ *suppenevat*

$$P_{X,Y}(P_\Theta(h(X,\Theta)) = Y) - \max_{j \neq Y} P_\Theta(h(X,\Theta) = j) < 0. \tag{4.3}$$

Todistus: Meidän riittää osoittaa, että jonoavaruudessa $\Theta_1, \Theta_2, ..$ on olemassa joukko C, jonka todennäköisyys on nolla ja jolle kaikilla $x \notin C$ pätee

$$\frac{1}{N}\sum_{n=1}^{N} I(h(\Theta_n, x) = j) \to P_\Theta(h(\Theta, x) = j) \tag{4.4}$$

Kiinnitetään opetusjoukko ja vektori Θ. Nyt joukko x, jolla $h(\Theta, x) = j$, on hypersuorakulmioiden yhdiste. Kaikille $h(\Theta, x)$ on vain äärellinen määrä K mahdollisia yhdisteitä $S_1, S_2, ...$ Määritellään, että on olemassa $\phi(\Theta) = k$, jos $\{x : h(\Theta, x) = j\} = S_k$. Olkoon N_k lukumäärä, jolloin on voimassa $\phi(\Theta_n) = k$ ensimmäiset N kertaa. Tällöin

$$\frac{1}{N}\sum_{n=1}^{N} I(h(\Theta_n, x) = j) = \frac{1}{N}\sum_{n=1}^{N} N_k I(x \in S_k) \tag{4.5}$$

Suurten lukujen lain mukaan

$$N_k = \frac{1}{N}\sum_{n=1}^{N} I(\phi(\Theta_n) = k) \to P_\Theta(\phi(\Theta) = k) \tag{4.6}$$

Kun otetaan yhdiste kaikista joukoista, joissa suppenemista ei tapahdu jollakin arvolla k, saadaan joukko C, jonka todennäköisyys on nolla ja jonka komplementti on muotoa

$$\frac{1}{N}\sum_{n=1}^{N} I(h(\Theta_n, x) = j) \rightarrow \sum_{k} P_\Theta(\phi(\Theta) = k) I(x \in S_k). \tag{4.7}$$

Oikea puoli on $P_\Theta(h(\Theta, x) = j)$. □

Satunnaismetsän yleistysvirheen yläraja voidaan laskea kahden parametrin avulla. Nämä parametrit ovat yksittäisen puun tarkkuus ja puiden välinen riippuvuus.

Määritelmä 4.3. Satunnaismetsän marginaalifunktio on

$$mr(X, Y) = P_\Theta(h(X, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(X, \Theta) = j) \tag{4.8}$$

ja luokittelijajoukon $h(x, \Theta)$ vahvuus on

$$s = E_{X,Y} mr(X, Y), \tag{4.9}$$

missä E_X, Y merkitsee satunnaismetsän marginaalifunktion odotusarvoa.

Oletetaan, että $s \geq 0$. Nyt Chebychevin epäyhtälö antaa meille

$$PE^* \leq var(mr)/s^2, \tag{4.10}$$

missä $var(mr)$ merkitsee satunnaismetsän marginaalifunktion varianssia. Marginaalifunktion varianssi voidaan ilmaista myös seuraavasti:
Oletetaan, että pätee

$$\hat{j}(X, Y) = arg \max_{j \neq Y} P_\Theta(h(X, \Theta) = j). \tag{4.11}$$

Näin ollen

$$mr(X, Y) = P_\Theta(h(X, \Theta) = Y) - P_\Theta(h(X, \Theta) = \hat{j}(X, Y)) \tag{4.12}$$

$$= E_\Theta[I(h(X,\Theta) = Y) - I(h(X,\Theta) = \hat{j}(X,Y))]$$

Määritelmä 4.4. Satunnaismetsän raakamarginaalifunktio on

$$rmg(\Theta, X, Y) = I(h(X,\Theta) = Y) - I(h(X,\Theta)) = \hat{j}(X,Y). \tag{4.13}$$

Näin ollen $mr(X,Y)$ on funktion $rmg(\Theta, X, Y)$ odotusarvo riippuen arvosta Θ. Kaikilla funktioilla f pätee

$$[E_\Theta f(\Theta)]^2 = E_{\Theta,\Theta'} f(\Theta) f(\Theta'), \tag{4.14}$$

kun Θ, Θ' ovat samalla jakaumalla varustettuja, mutta keskenään riippumattomia olettaen, että

$$mr(X,Y)^2 = E_{\Theta,\Theta'} rmg(\Theta, X, Y) rmg(\Theta', X, Y). \tag{4.15}$$

Tästä saamme

$$\begin{aligned} var(mr) &= E_{\Theta,\Theta'}(cov_{X,Y} rmg(\Theta, X, Y) rmg(\Theta', X, Y) \\ &= E_{\Theta,\Theta'}(\rho(\Theta,\Theta') sd(\Theta) sd(\Theta')), \end{aligned} \tag{4.16}$$

missä $\rho(\Theta,\Theta')$ on funktioiden $rmg(\Theta, X, Y)$ ja $rmg(\Theta', X, Y)$ välinen riippuvuus, kun Θ, Θ' ovat kiinnitettyjä ja $sd(\Theta)$ on standardoitu poikkeama funktiolle $rmg(\Theta, X, Y)$, kun Θ on kiinnitetty. Tällöin

$$var(mr) = \bar{\rho}(E_\Theta sd(\Theta))^2 \leq \bar{\rho} E_\Theta var(\Theta), \tag{4.17}$$

missä $\bar{\rho}$ on korrelaation keskiarvo olettaen, että

$$\bar{\rho} = E_{\Theta,\Theta'}(\bar{\rho}(\Theta,\Theta') sd(\Theta) sd(\Theta'))/E_{\Theta,\Theta'}(sd(\Theta) sd(\Theta')). \tag{4.18}$$

Kirjoitetaan

$$E_\Theta var(\Theta) \leq E_\Theta(E_{X,Y} rmg(\Theta, X, Y))^2 - s^2 \leq 1 - s^2. \tag{4.19}$$

Tämän avulla saamme seuraavan teoreeman.

Teoreema 4.5. *Yleistysvirheen yläraja on*

$$PE^* \leq \bar{\rho}(1 - s^2)/s^2. \tag{4.20}$$

Ylärajan arviosta näkee, että satunnaismetsän yleistysvirhe riippuu metsän yksittäisten luokittelijoiden vahvuudesta ja niiden välisestä korrelaatiosta. Näiden avulla voidaan laskea hyvänä ohjenuorana pidetty c/s^2 arvo, joka on korrelaatio jaettuna vahvuuden neliöllä. Mitä pienempi arvo on, sitä parempi.

Määritelmä 4.6. Satunnaismetsän arvio c/s^2 saadaan

$$c/s^2 = \bar{\rho}/s^2. \tag{4.21}$$

4.4 Satunnaismetsän hyödyt ja haitat

Satunnaismetsä on tehokas menetelmä ennustamiseen. Oikein tehdyllä satunnaistamisella satunnaismetsä on myös hyvä luokittelija ja suurten lukujen lain mukaisesti satunnaismetsä ei ylisovitu opetusjoukkoon. Se on nopeampi menetelmä kuin esimerkiksi bagging tai boosting ja satunnaismetsä tuottaa hyödyllisiä arvioita metsän luokittelijoiden virheistä, vahvuudesta, korrelaatioista ja parametrien merkittävyydestä. Se on kohtuullisen immuuni datan kohinalle ja satunnaistamisen tuloksena puuttuvat arvot eivät muodostu ongelmiksi.

Riippuu tietenkin analysointitehtävästä, mihin satunnaismetsää kannattaa soveltaa. Esimerkiksi regressiotyökaluna satunnaismetsä ei ole tuottanut aivan yhtä hyviä tuloksia kuin luokittelutehtävissä. Koska yleistysvirhe pienenee sitä mukaan, kun joukko ja puiden määrä kasvaa, satunnaismetsä toimii parhaiten suurien tietojoukkojen kanssa. Pieniin datajoukkoihin esimerkiksi yksinkertainen puu voi olla tarkempi luokittelija. Joissain tapauksissa satunnaismetsä voi olla tarkempi luokittelija, mutta tutkimukset ovat osoittaneet, että mitä suurempi syötejoukko, sitä pienempi virheaste [11].

Luku 5

Satunnaismetsän soveltaminen

Tässä kappaleessa sovellamme satunnaismetsä-menetelmää yksinkertaisen datajoukon luokittelemiseen. Käytämme UCI Machine Learning Repository-sivustolta[1] ladattua Iris-tietokantaa. Kyseistä tietokantaa on käytetty lukuisissa tutkimuksissa, joista voi lukea tarkemmin UCI Machine Learning Repository-sivustolta.

RStudio-ohjelmisto tarjoaa valmiin satunnaismetsä-paketin, jota käytämme tässä luokittelutehtävässä. Tämän jälkeen suoritamme saman luokittelutehtävän SAS Enterprise Minerilla verifioidaksemme tulokset ja tarjolla olevien satunnaismetsä-pakettien johdonmukaisuuden. Suoritamme SAS Enterprise Minerilla myös vertailun kahteen kilpailevaan malliin. Käytämme validointimenetelminä luokittelemistarkkuutta, konfuusiomatriisia ja nostokäyrää.

5.1 Satunnaismetsä-menetelmän soveltaminen Iris-tietokantaan

Käyttämämme tietokanta sisältää tietoa kurjenmiekoista. Kyseisessä tietokannassa kurjenmiekkoja on kolme alalajia ja jokaisesta lajista on tiedossa verholehden pituus ja leveys ja terälehden pituus ja leveys, kuvan 5.1 mukaisesti. Käytetyssä tietokannassa näitä havaintoja on yhteensä 150. Tehtävämme on opettaa algoritmimme luokittelemaan kukka oikein perustuen saatavilla olevien attribuuttien arvoihin.

[1] Lichman, M., http://archive.ics.uci.edu/ml, Irvine, CA: University of California, School of Information and Computer Science, 2013

Kurjenmiekka	Lehden pituus	Lehden leveys	Terälehden pituus	Terälehden leveys
Setosa	4,8	3,4	1,6	0,2
Setosa	5,1	3,5	1,4	0,2
Setosa	4,9	3,0	1,4	0,2
Versicolor	7,0	3,2	4,7	1,4
Versicolor	6,4	3,2	4,5	1,5
Versicolor	6,9	3,1	4,9	1,5
Virginica	5,7	2,5	5,0	2,0
Virginica	6,3	3,3	6,0	2,5
Virginica	5,8	2,7	5,1	1,9

Taulukko 5.1: Esimerkki käytetystä tietokannasta.

5.1.1 RStudio

Käytimme RStudion valmista kirjastoa ja sen *sample*-komentoa, jolla muodostimme 150 rivin datajoukosta satunnaisen 100 rivin joukon. Tällä 100-rivin joukolla harjoitimme algoritmia. RStudion satunnaismetsä-paketin oletuksissa on, että algoritmi kasvattaa 500 puuta ja emme tehneet tähän muutoksia.[2]

```
> library(randomForest)
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.
> s <- sample(150,100)
> iris_train <- iris[s,]
> iris_test <- iris[-s,]
```

Kuva 5.1: Joukon näytteistäminen.

Tämän jälkeen harjoitimme algoritmin syötejoukon avulla. Seuraavaksi katsoimme, kuinka hyvin algoritmimme osasi luokitella 50 rivin joukkomme, jonka jätimme pois alkuperäisestä joukosta testausta varten. *Mean*-komennolla saimme tietää, mikä oli algoritmimme tarkkuus, joka tässä tapauksessa oli 92%.

[2] http://trevorstephens.com/

```
> rfm <- randomForest(Species ~., iris_train)
> p <- predict(rfm, iris_test)
> table(iris_test[,5],p)
            p
             setosa versicolor virginica
  setosa         15          0         0
  versicolor      0         17         1
  virginica       0          3        14
> mean(iris_test[,5]==p)
[1] 0.92
```

Kuva 5.2: Algoritmin hämmennysmatriisi ja tarkkuus.

Kuvassa 5.2 näkyvä hämmennysmatriisi näyttää, että algoritmi luokitteli yhden versicolor-iriksen virginicaksi ja kolme virginicaa versicoloriksi.

Hämmennysmatriisin avulla laskimme kuvassa 5.3 näkyvän ROC-käyrän.

Kuva 5.3: ROC-käyrä.

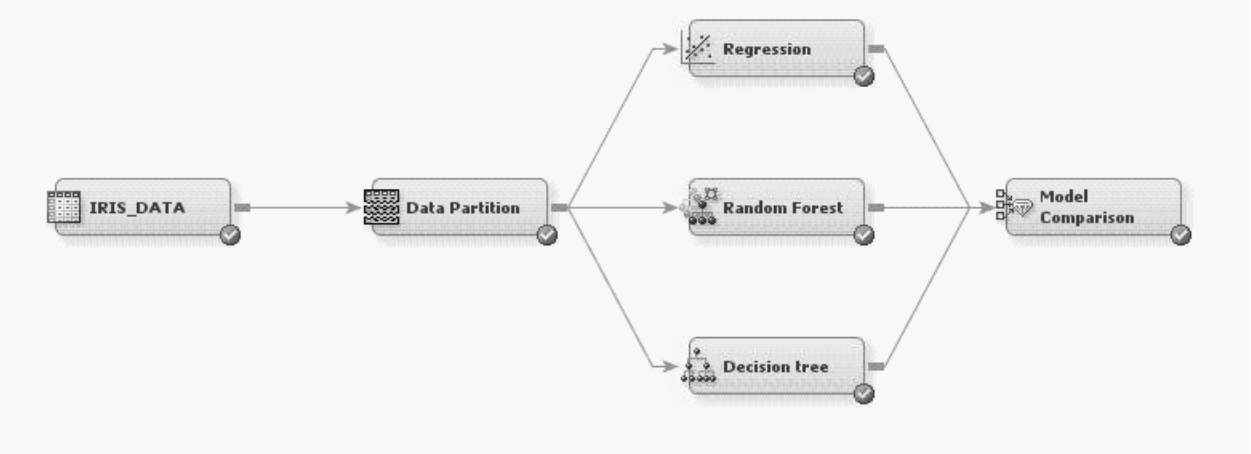
Kuvan 5.4 taulukko näyttää tehtävässä olevien muuttujien painoarvot. Tässä tapauksessa tieto ei ole olennainen, mutta muissa analysointitehtävissä satunnaismetsää voisi käyttää merkittävien muuttujien määrittämiseen.

```
> importance(rfm)
             MeanDecreaseGini
Sepal.Length         7.669485
Sepal.Width          1.563349
Petal.Length        32.071216
Petal.Width         24.655333
```

Kuva 5.4: Muuttujien painoarvoja.

5.1.2 SAS Enterprise Miner

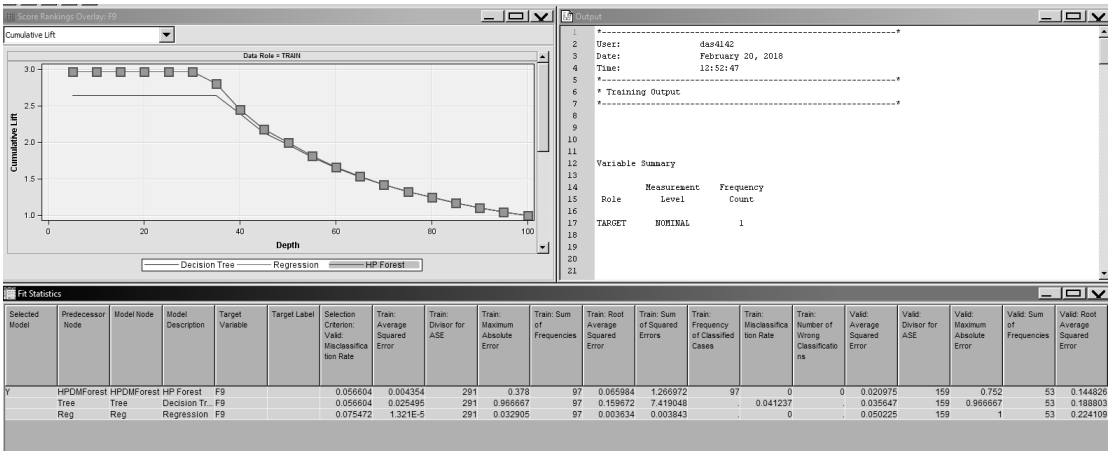
Seuraavaksi tutkimme SAS Enterprise Minerin HP Forest solmun suoriutumista luokittelutehtävässä, jossa käytämme jo aiemmin käyttämäämme Iris-tietokantaa. HP Forest solmu tuottaa satunnaismetsän algoritmin automaattisesti SAS Enterprise Minerin käyttöliittymään. Tarkoituksena on verifioida eri ohjelmien tuottamien tulosten johdonmukaisuus ja järjestelmien tarjoamien pakettien toimivuus.



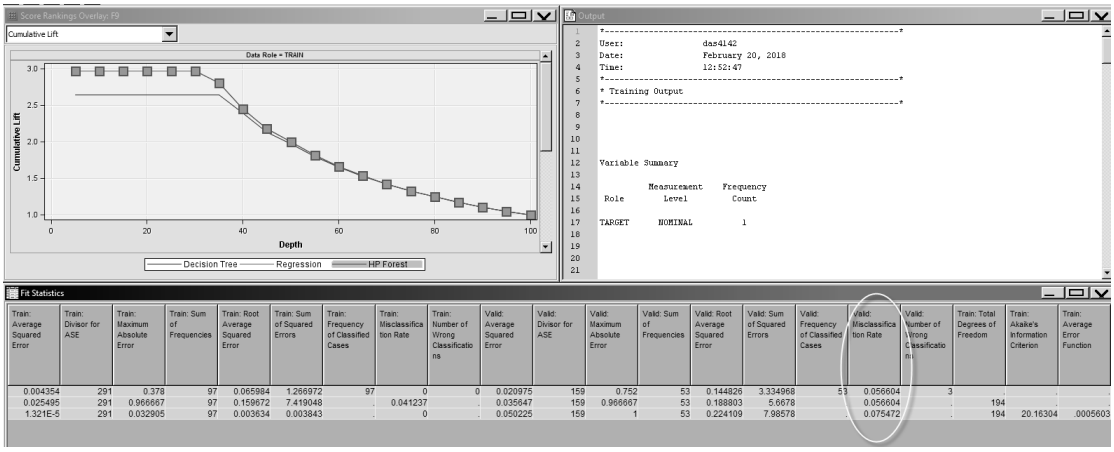
Kuva 5.5: SAS Enterprise Minerissa käytetty malli.

Teimme SAS:illa myös kilpailevat puu- ja regressiomallit, jotta näkisimme, kuinka hyvin HP Forest solmu toimii suhteessa muihin malleihin. Syötimme asetuksiin, että algoritmi kasvattaa 500 puuta, jotta vertailu RStudion tuloksiin olisi mielekästä.

Model comparison-solmun avulla voimme vertailla luotujen mallien tehtävästä suoriutumista. Kuvista 5.6 ja 5.7 näemme solmun tuottamat tulokset. Nostokäyrästä näkyy, että satunnaismetsä ja regressiomalli saivat yhtä hyvät tulokset, mutta tarkkuustuloksien mukaan satunnaismetsä ja päätöksentekopuu suoriutuivat yhtä hyvin ja paremmin tietokannan luokittelemisessa kuin logistinen regressiomalli. Satunnaismetsän validaatiodatan väärinluokitteluaste oli 5,7% eli algoritmin tarkkuus oli näin ollen n.94%.



Kuva 5.6: SAS Enterprise Minerin tulokset.



Kuva 5.7: SAS Enterprise Minerin tulokset.

5.2 Iris-tietokannan luokittelemisen tulokset ja analysointi

SAS Enterprise Minerin ja RStudion tulokset olivat johdonmukaiset keskenään, sillä tarkkuus oli yli 90%, joka on erittäin hyvä. RStudiolla tehtäessä Satunnaismetsän ROC-käyrä oli ihanteellinen. SAS Enterprise Minerilla huomattiin mallien vertailussa, että päätöksentekopuu suoriutui luokittelutehtävästä yhtä hyvin kuin satunnaismetsä ja toisaalta regressiomallin nostokäyrä oli samalla tasolla kuin satunnaismetsän nostokäyrä. Tämä selittyy sillä, että käyttämämme Iris-tietokanta oli hyvin pieni. Kokonaisuudessaan satunnaismetsä suoriutui luokittelutehtävästä parhaiten.

Luku 6

Satunnaismetsän soveltaminen reaalimaailman datan kanssa

Seuraavaksi sovellamme satunnaismetsää reaalimaailman ongelmaan. Käytämme yrityksen tarjoamaa datajoukkoa kyseisen yrityksen tietyn asiaskassegmentin asiakasvaihtuvuuden eli churn-käyttäytymisen selittämiseksi ja ennustamiseksi.

Käytössämme on yrityksen tietokantataulut, joista rakennamme tehtävää varten tietojoukon. Tähän joukkoon on kerätty tietyllä aikavälillä liittyneet uudet asiakkaat. Tutkimuskysymyksenä on uusien asiakkaiden asiakasvaihtuvuus tietyssä segmentissä. Tarkoituksena on ensiksi valikoida sopivat muuttujat selittämään vaihtuvuutta kohdemuuttujan "churn-flg"avulla. Kohdemuuttuja on binäärivaste, jossa arvo 1 kuvaa, että vaihtuvuutta on tapahtunut ja arvo 0 kuvaa, että vaihtuvuutta ei ole tapahtunut.

Rakennamme selittävän ja ennustavan päätöksentekopuu-mallin ja satunnaismetsä-menetelmää soveltavan mallin. Lopuksi vertailemme malleja olemassaoleviin asiakasvaihtuvuustietoihin arvioidaksemme, kuinka hyvin mallimme todellisuudessa ennustivat asiakasvaihtuvuutta. Käytämme mallien rakentamiseen yrityksen hyödyntämää tuotantotyökalua SAS Enterprise Mineria.

6.1 Asiakasvaihtuvuuden selittäminen ja ennustaminen

Lähdimme rakentamaan tietojoukkoa mallia varten. Joukkoon kerättiin uudet asiakkaat valikoimalla käyttäjät, jotka olivat avanneet uuden palvelun syys-, loka- tai marraskuussa vuonna 2016. Näille asiakkaille määritettiin asiakassegmentti. Asiakasnumeron yhteyteen liitettiin tietoja asiakkaan

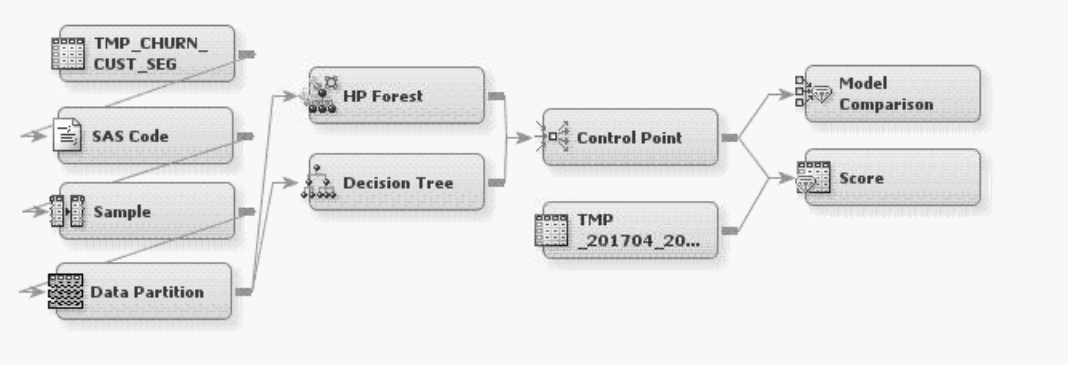
iästä, sukupuolesta, tilaamista palveluista ynnä muuta, sekä churn-lippu; mikäli asiakas oli irtisanonut edes yhden tuotteen, hän sai arvon 1, jos ei, arvon 0. Taulukossa 6.1 näkyy pieni otos tästä joukosta.

ID_COMBINED_PARTY_ID	ID_YEARMONTH	ID_BPN	ID_FIRST_DAY_OF_MONTH_ID	AREA_CUST_3G_INDOOR_ALL_FLG	AREA_CUST_4G_AVAIL_FLG	AREA_CUST_4G_PLAN_4WEEKS_F	AREA_CUST_ILEC_CLEC	AREA_CUST_IPTV_SALES_OK_FLG	CAMPC_CC_CALL_FLG	CAMPC_COMM_AGE	CUST_AGE
14195320	201611	3000961452	42673	1	1	0	CLEC1	1	.	106	30
14195399	201611	1000213786	42673	0	1	0	CLEC1	1	.	34	62
1345659	201611	1011511697	42673	0	1	0	CLEC1	0	0	147	64
14195754	201611	3000957916	42673	0	1	0	CLEC1	0	0	27	88
12888333	201611	1011470754	42673	1	1	0	ILEC1	1	.	2	89
1477115	201611	1011919451	42673	0	1	0	CLEC1	1	.	27	70
1191134	201611	1010660604	42673	0	1	0	CLEC3	0	.	27	83
1101382	201611	1009122894	42673	1	1	0	CLEC1	1	.	2	51
16485246	201611	3001289297	42673	0	1	0	ILEC1	1	.	.	21
4531891	201611	1037864551	42673	1	1	0	CLEC1	1	.	.	52
16774997	201611	1018303594	42673	0	1	0	ILEC3	0	.	27	71
16775454	201611	3001385608	42673	0	1	0	CLEC1	1	.	.	20
1114724	201611	1009548960	42673	0	1	0	CLEC3	0	0	2	66
1114816	201611	1009481797	42673	1	1	0	ILEC1	1	.	9	65
1345931	201611	1011569089	42673	0	1	0	ILEC3	0	0	13	69
1346000	201611	1011183795	42673	1	1	0	CLEC1	1	1	6	62
14195796	201611	3000956147	42673	1	1	0	CLEC1	1	.	34	69
980025	201611	1007969567	42673	1	1	0	CLEC3	0	.	16	40
13302682	201611	3000845728	42673	1	1	0	CLEC1	1	.	8	53
4001523	201611	1030803616	42673	1	1	0	CLEC1	0	1	3	66
4001614	201611	1031002953	42673	.	.	.		.	.	34	38
4514911	201611	1038016719	42673	1	1	0	ILEC2	1	0	2	50
14195398	201611	3000956014	42673	0	1	0	CLEC1	1	0	34	24
14195447	201611	3001427273	42673	0	1	0	ILEC1	1	.	146	51
14195607	201611	3000957964	42673	1	1	0	CLEC1	1	0	28	26
14195741	201611	3000956165	42673	1	1	0	CLEC1	1	.	27	25
16774949	201611	1018509592	42673	0	1	0	CLEC1	0	0	.	66
12715345	201611	1011550483	42673	1	1	0	CLEC1	1	.	71	64
16763589	201611	3000229538	42673	1	1	0	ILEC1	0	1	14	63

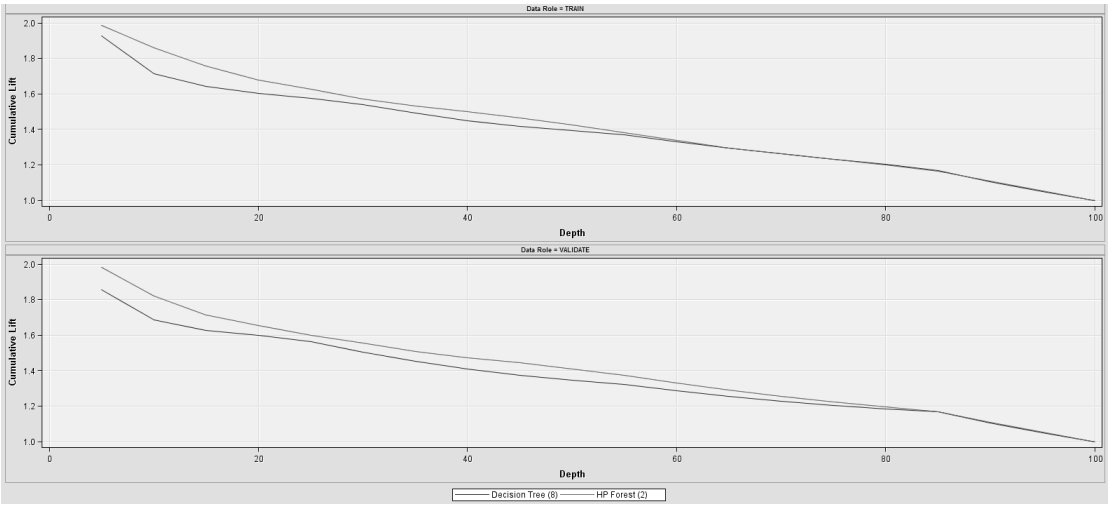
Taulukko 6.1: Pieni otos käytetystä datasta.

Kun datajoukko oli saatu muodostettua, tiedot koottiin yhdeksi tauluksi, joka siirrettiin SAS Enterprise Mineriin tietolähteeksi. Koska halusimme tietää tietyn asiakassegmentin churn-käyttäytymistä, suodatimme muihin segmentteihin kuuluvien asiakkaiden tiedot joukosta pois. Tämän jälkeen jaoimme joukon testi- ja validointijoukoiksi.

Rakensimme SAS Enterprise Minerilla kaksi kilpailevaa mallia, päätöksentekopuu- ja satunnaismetsä-mallit. Satunnaismetsä-solmuun määritettiin, että solmu rakentaisi 500 luokittelupuuta. Model Comparison-solmulla pystyimme vertailemaan, miten mallit suoriutuivat testijoukon luokittelemisesta. Kuten kuvissa 6.2 ja 6.3 näkyy, satunnaismetsä suoriutui tehtävästä hiukan paremmin. Tämä oli odotettavaa, sillä päätöksentekopuu ylisovittuu hyvin joukkoon, jolloin testijoukon nosto-käyrä näyttää kohtuullisen korkeaa arvoa, vaikka mallin ennustuskyky saattaa olla heikko ylisovituksesta johtuen. Kuvan 6.4 tuloksista näkee, että satunnaismetsän väärinluokitteluaste oli 0,295 eli tarkkuus oli 70,5%, kun taas päätöksentekopuun väärinluokitteluaste oli 0,32 eli tarkkuus oli 68%.

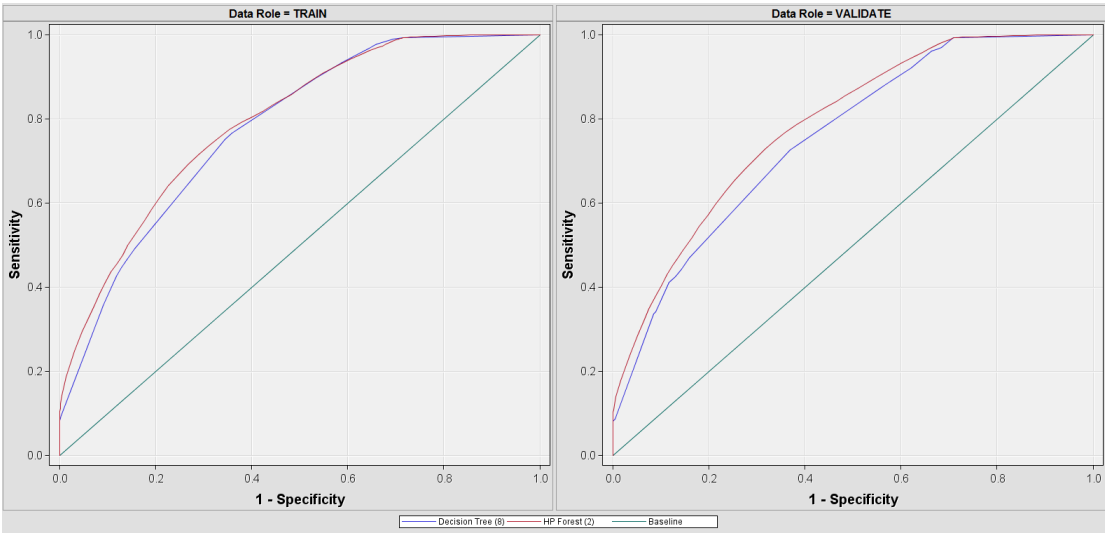


Kuva 6.1: SAS Enterprise Miner diagrammi.



Kuva 6.2: Nostokäyrät.

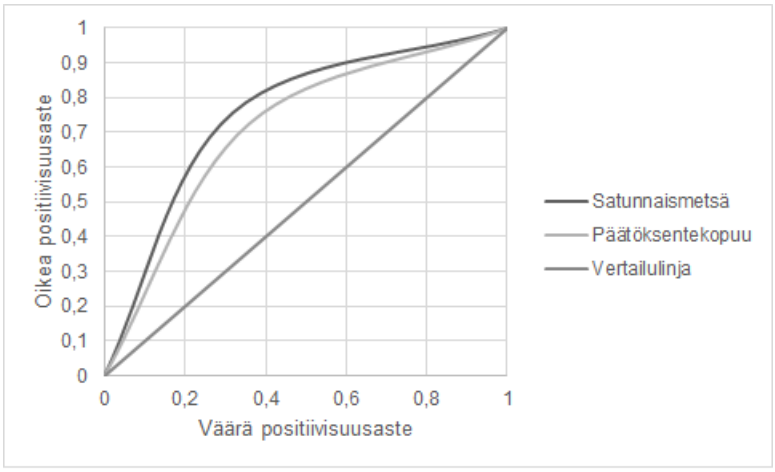
Seuraavaksi tutkimme, miten hyvin algoritmit suoriutuivat ennustustehtävästä. Tämä tapahtui tekemällä SAS Enterprise Minerin score-solmulla joukosta niin kutsuttu *scoredata*, kuva 6.1. Algoritmille syötettiin uusi joukko churn-tietoineen. Kyseinen joukko oli aikavälillä 04/2017-08/2017 liittyneet uudet asiakkaat. Score-solmu laski mallien perusteella kyseisille asiakkaille todennäköisyydet, joilla kukin asiakas irtisanoo palvelun tai ei. Tämän jälkeen score-solmu liitti tiedot, onko asiakas todellakin irtisanonut tuotteen nykyisen tiedon perusteella. Teimme tämän sekä päätöksentekopuulle että satunnaismetsälle. Siirsimme nämä tiedot SAS Enterprise Guideen ja Exceliin, joissa laskimme kyseisille scoredatoille ROC-käyrät, kuva 6.5.



Kuva 6.3: ROC-käyrät.

Model Node	Model Description	Target Variable	Target Label	Selection Criterion: Valid: Misclassification Rate	Train: Sum of Frequencies	Train: Misclassification Rate	Train: Maximum Absolute Error	Train: Sum of Squared Errors	Train: Average Squared Error	Train: Root Average Squared Error	Train: Divisor for ASE	Train: Total Degrees of Freedom	Valid: Sum of Frequencies	Valid: Misclassification Rate	Valid: Maximum Absolute Error	Valid: Sum of Squared Errors	Valid: Average Squared Error	Valid: Root Average Squared Error	Valid: Divisor for VASE	Train: Frequency of Classified Cases
HPDMFore...	HP Forest (2)	TG_CHUR...	TG_CHUR...	0.294738	82473	0.2883	0.897806	31199.16	0.189148	0.434911	164946		44409	0.294738	0.888956	16977.1	0.191145	0.437201	88818	82473
Tree8	Decision Tr...	TG_CHUR...	TG_CHUR...	0.321782	82473	0.296461	0.997058	30664.1	0.185904	0.431166	164946	82473	44409	0.321782	0.997058	17208.51	0.19375	0.440171	88818	

Kuva 6.4: Tulokset.



Kuva 6.5: Mallien score-datan ROC-käyrät.

Uuden joukon luokittelussa satunnaismetsän tarkkuus oli 70,5% ja oikea positiivisuusaste oli 75,8%. Päätöksentekopuun tarkkuus oli 67,8% ja

oikea positiivisuusaste oli 72,6%. Tulokset vastaavat luokittelun yhteydessä laskettuja arvoja. Kuvan 6.5 perusteella näkee, että satunnaismetsä suoriutui ennalta tuntemattoman joukon luokittelussa hiukan paremmin kuin päätöksentekopuu. Tutkimusten perusteella on oletettavaa, että satunnaismetsän algoritmi sietää paremmin aikaa kuin päätöksentekopuu tarkoittaen, että ajan kuluessa satunnaismetsä luultavasti kykenisi ennustamaan vastaavien joukkojen asiakasvaihtuvuutta paremmin kuin päätöksentekopuu.

6.2 Satunnaismetsän soveltuvuus yhtiön liiketoimintaprosessiin

Tiedonlouhinta on monipuolinen tieteenala, jonka ansiosta yritykset voivat optimoida toimintaansa. Datalouhinnassa tulee ottaa huomioon, miten mahdollinen menetelmä soveltuu yhtiön olemassa oleviin rakenteisiin. Toisin sanoen onko kyseinen menetelmä hyödynnettävissä yrityksessä ja tuoko se jotain lisäarvoa yhtiön liiketoimintaprosessiin. Analysoimme seuraavaksi satunnaismetsän soveltuvuutta yrityksen liiketoimintaprosessiin.

Yrityksen tiedonlouhinnan liiketoimintaprosessi on iteroituva ja koostuu olennaisesti kuvan 6.6 mukaisista osa-alueista.

'Mitä halutaan tehdä tai tietää'-vaiheessa yritys määrittelee ongelman tai kysymyksen, johon halutaan vastaus. Tämä voi olla esimerkiksi markkinointikampanjan kohderyhmän etsiminen tai asiakasvaihtuvuuden selittäminen.

'Saatavilla olevan datan selvittäminen'-vaiheessa tutkitaan, mitä tietoja yrityksellä on valmiina saatavilla tai onko mahdollista saada haluttu data aikaan yhdistelemällä olemassa olevia tietoja.

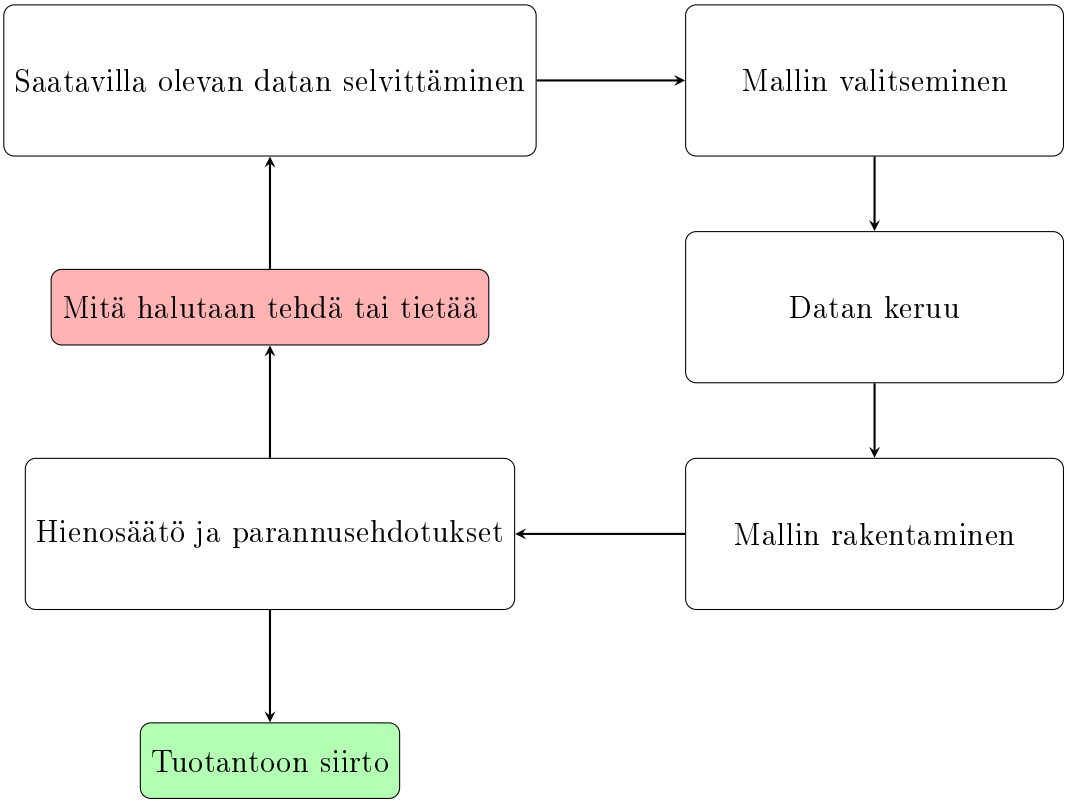
'Mallin valitseminen'-vaiheessa valitaan kyseiseen ongelmaan soveltuvin malli. Esimerkiksi mikäli halutussa datassa ilmenee paljon puuttuvia arvoja, logistinen regressio ei ole optimaalisin mallivalinta.

'Datan keruu'-vaiheessa kerätään tarvittavat tiedot valittua mallia varten.

'Mallin rakentaminen'-vaiheessa rakennetaan malli ja analysoidaan tulokset.

'Hienosäätö ja parannusehdotukset'-vaiheessa tehdään mahdolliset hienosäädöt malliin esimerkiksi valikoimalla datasta paremmat muuttujat, lisäämällä uusia tietoja dataan, mahdollisesti suunnitellaan toimenpide, jolla voidaan kerätä tietoja, jotta saadaan vastaus asetettuun kysymykseen tai muotoillaan tutkimuskysymys uudestaan.

'Tuotantoon siirto'-vaiheessa malli siirretään tuotantoon, mikäli 'Hienosäätö ja parannusehdotukset'-kohdassa mainituille toimenpiteille ei ilmene tarvetta.



Kuva 6.6: Liiketoimintaprosessi.

Satunnaismetsä vaikuttaa olennaisesti 'Mallin valitseminen'-, 'Datan keruu'-, 'Mallin rakentaminen'-, 'Hienosäätö ja parannusehdotukset'- ja 'Tuotantoon siirto'-vaiheisiin, joten käsittelemme minkälaista lisäarvoa satunnaismetsä toisi yrityksille näissä vaiheissa prosessia.

'Mallin valitseminen'-vaiheessa yleinen hyvä periaate on, että analyytikko hyödyntää malleja ja validointimenetelmiä, joita hän itse ymmärtää. Ymmärrys takaa nopeamman työskentelytahdin ja helpottaa vianhallintaa merkittävästi. Näin ollen vaikka tarjolla on monenlaisia malleja, ainoastaan ne, jotka analyytikko itse hallitsee ja ymmärtää, tuottavat kumulatiivisesti parempia tuloksia.

Satunnaismetsän toimintaperiaate on kohtalaisen helposti ymmärrettävissä, kuten päätöksentekopuu- ja regressiomalleilla. Satunnaismetsä tarjoaa nopeammat ja tarkemmat tulokset kuin kyseiset mallit yksinkertaisissakin

datajoukoissa, kuten olemme luvussa 5 osoittaneet. Mahdolliset puuttuvat arvot datassa eivät häiritse satunnaismetsän suorituskykyä.

'Datan keruu'-vaiheessa satunnaismetsää voi myös hyödyntää datajoukon merkittävimpien muuttujien selvittämiseen. Mikäli analyytikko on päätynyt käyttämään analysoimiseen jotain muuta mallia, satunnaismetsällä pystyy nopeasti selvittämään merkittävät muuttujat. Tämän avulla on mahdollista rakentaa kompakti datajoukko analysointia varten. Tämä on niin ikään hyödyllinen ominaisuus 'Hienosäätö ja parannusehdotukset'-vaiheessa.

'Mallin rakentaminen'-vaiheessa tulee ottaa huomioon, kuinka helposti mallin rakentaminen onnistuu. SAS Enterprise Miner tarjoaa valmiin HP Forest solmun, joten satunnaismetsää voi hyödyntää yrityksissä, jotka käyttävät SAS:ia analysointitehtäviin. SAS on hiirivetoinen käyttöliittymä, joten mikäli työntekijät osaavat käyttää SAS:ia valmiiksi, satunnaismetsä ei vaadi työntekijöiltä merkittävästi uuden opettelemista, sillä SAS:in HP Forest solmun oletusasetukset toimivat riittävän hyvin.

'Tuotantoon siirto'-vaiheessa tulee ottaa huomioon satunnaismetsän siirrettävyys tuotantoon. SAS tarjoaa mallin hallinnoimisiin Decision Manager-työkalua, jonne voi automaattisesti siirtää SAS Enterprise Minerilla rakennetut mallit. Decision Managerissa malleille voi laskea scoredatan ja siirtää mallit osaksi tuotantoa. Valitettavasti SAS ei kykene siirtämään automaattisesti algoritmeja, joilla on liian pitkä koodi johtuen koodirivien ylärajasta. Tämä olennaisesti koskee kaikkia joukko-oppimismenetelmiä eli myös satunnaismetsää. Algoritmien siirtäminen käsin SAS Decision Manageriin on mahdollista, mutta tässä vaiheessa helppokäytöisyys ja työ-hyötysuhde kärsivät merkittävästi. Kyseiset scoredatat voidaan laskea myös SAS Enterprise Guidella tai jopa Excelillä, kuten tässä tutkielmassa, mutta mallin saaminen tuotantoon on haastavaa, koska prosessi pitäisi näin ollen tehdä manuaalisesti. SAS on kaupallinen yritys, joten on oletettavaa, että SAS päivittää ja kehittää ominaisuuksiaan, mutta tällä hetkellä satunnaismetsää soveltavan mallin siirtäminen tuotantoon on aikaa vievää ja haasteellista.

Satunnaismetsä tuottaa lisäarvoa olemalla helppokäyttöinen, monipuolinen, ymmärrettävä, nopea ja tarkka. Satunnaismetsä on kohtuullisen immuuni datan taustahälylle ja satunnaismetsän suorituskyky paranee syötejoukon kasvaessa. Olemme tässä tutkielmassa osoittaneet, että satunnaismetsä suoriutuu pienenkin joukon luokittelemisessa paremmin kuin päätöksentekopuuja regressiomalli. Yrityksen tarjoamassa liiketoimintaongelmassa satunnaismetsä suoriutui sekä luokittelussa että ennustamisessa paremmin kuin yksinkertainen päätöksentekopuu. Tässä kappaleessa nostetut seikat huomioon ottaen satunnaismetsä on erittäin hyvä lisä osaksi yrityksen liiketoimintaprosessia.

Luku 7

Johtopäätökset

Olemme käsitelleet tutkielmassa satunnaismetsä-koneoppimismenetelmää ja tutkimuskysymyksenä oli menetelmän soveltuvuus yrityksen liiketoimintaprosessiin. Käsittelimme koneoppimista ja päätöksentekopuiden algoritmia kartuttaaksemme pohjatietoja, jotka olivat välttämättömiä satunnaismetsäkoneoppimismenetelmän ymmärtämiseksi. Niin ikään käsittelimme datalouhintaa ja validointikeinoja ymmärtääksemme kyseisten koneoppimismenetelmien sovelluskohteita, sekä keinoja validoida menetelmän suoriutumista. Sovelsimme opittuja taitoja yksinkertaiseen luokitteluongelmaan ja yrityksen tarjoamaan liiketoimintaongelmaan. Lopuksi analysoimme satunnaismetsän soveltuvuutta yrityksen liiketoimintaprosessiin.

Tutkielmassa esitettyjen validointimenetelmien, sekä liiketoimintaprosessin analysoinnin pohjalta päädyimme lopputulokseen, että satunnaismetsäkoneoppimismenetelmä soveltuisi hyvin yrityksen liiketoimintaprosessiin. Satunnaismetsä on monipuolinen menetelmä, joka pääsee oikeuksiinsa nykytrendin mukaisessa tavassa sisällyttää käsiteltäväksi mahdollisimman paljon tietoa ja on hyvä lisä analyytikoiden menetelmäarsenaaliin.

Kirjallisuutta

[1] Hilbert, M., *Big data for development: A review of promises and challenges*, Development Policy Review, Vol. 34, Issue 1, s. 135 - 174, 01/2006

[2] Fernàndez-Delgado M., Cernadas E., Barro S., *Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?*, Journal of Machine Learning Research 15 3133-3181, 2014

[3] Samuel, A., *Some Studies in Machine Learning Using the Game of Checkers*, IBM Journal of Research and Development, 1995

[4] Rokach L., Maimon O., *Data Mining with decision tree, Theory and Applications*, 04/2008

[5] Arora S., Barak B., *Computational Complexity: A Modern Approach*, 2007

[6] Hancock T. R., Jiang T., Li M., Tromp J., *Lower Bounds on Learning Decision Lists and Trees*, Information and Compuation 126(2), s. 114-122, 1996

[7] Hyafil L. and Rivest R.L., *Constructing optimal binary decision trees is NP- complete*, Information Processing Letters, 5(1), s. 15-17, 1976

[8] Schaffer, C., *When does overfitting decrease prediction accuracy in induced decision trees and rule sets?*, In Proceedings of the European Working Sessionon Learning (EWSL-91), s. 192-205, Berlin, 1991

[9] Dietterich T. D., *Ensemble Learning*, 09/2002

[10] Provost F., Fawcett T., *The case against accuracy estimation for comparing induction algorithms.* Proc. 15th Intl. Conf. On Machine Learning, s. 445-453, Madison, WI, 1998

[11] Breiman, L., *Random Forests*, 45, s. 5–32, c 2001 Kluwer Academic Publishers, Manufactured in The Netherlands, 2001

[12] Tumer, K. and Ghosh J., *Error Correlation and Error Reduction in Ensemble Classifiers, Connection Science, Special issue on combining artificial neural networks: ensemble approaches*, 8 (3-4), s. 385-404, 1996

[13] Hu, X., *Using Rough Sets Theory and Database Operations to Con-*

struct a Good Ensemble of Classifiers for Data Mining Applications. ICDM01. s. 233-240, 2001

[14] Gordon S., Linoff, Michael J., Berry A., *Data Mining Techniques For Marketing, Sales and Customer Relationship Management*, 03/2011

[15] Quinlan, J. R., *Induction of Decision trees*, Machine Learning 1: s. 81-106, © 1986 Kluwer Academic Publishers, Boston - Manufactured in The Netherlands, 1986

[16] Wolpert, D. H., *The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework. In D. H. Wolpert, editor, The Mathematics of Generalization*, The SFI Studies in the Sciences of Complexity, s. 117-214. AddisonWesley, 1995

[17] Emet, S., *Johdatus todennäköisyyslaskentaan ja tilastotieteeseen*, Matematiikan ja tilastotieteen laitos, Turun Yliopisto, 2014

[18] Grimmett, G., Stirzaker, D., *Probability and Random Processes.* Oxford University Press Inc., New York, 2001