Ohjelmiston kehittäminen puutteellisen ja virheellisen havaintoaineiston käsittelyyn

Ismo Horppu

Tietotekniikan pro gradu -tutkielma
Ohjelmistotekniikan linja
11. helmikuuta 2002

Jyväskylän yliopisto
Tietotekniikan laitos

Tekijä: Ismo Horppu
Yhteystiedot: ishorppu@cc.jyu.fi
Työn nimi: Ohjelmiston kehittäminen puutteellisen ja virheellisen havaintoaineiston käsittelyyn
Title in English: Software development for missing and erroneous data
Työ: Pro Gradu- tutkielma
Sivumäärä: 121
Linja: Ohjelmistotekniikka
Teettäjä: Jyväskylän yliopisto, tietotekniikan laitos
Avainsanat: editointi, imputointi, puutteellinen ja virheellinen data, mallintaminen, säännöt, TS-SOM, editointi- ja imputointisovellus
Keywords: editing, imputation, missing and erroneous data, modelling, rules, TS-SOM, editing and imputing application

Esipuhe

Virheellisyys ja puutteellisuus havaintoaineistossa on keskeinen ongelma useissa reaalimaailman mallinnustehtävissä. Ongelman käytännön merkitys on suuri analysoitaessa yhteiskunnallisesti merkittäviä tilastoaineistoja, joissa esimerkiksi määrätään kansantaloutta tai yrityselämää kuvaavia indeksejä. Kerätty raaka-aineisto voi sisältää jopa 20%:n osalta epätäydellistä tietoa, jonka pohjalta tulisi pystyä määräämään alle yhden prosentin tarkkuudella määriteltäviä tunnuslukuja kaupan, työelämän ja valtiontalouden tilasta. Virheiden havainnointia sekä korjausta kutsutaan alan kirjallisuudessa editoinniksi, ja puutteellisten havaintojen täydentämistä imputoinniksi.

Tämän työn lähtökohtana ovat

- tilastokeskuksen data (puuttuvan ja virheellisen datan käsittely on tilastokeskuksille erittäin tärkeää),
- EurEdit-projekti (joka pyrkii datan editointi- ja imputointimenetelmien arvioimiseen ja kehittämiseen),
- NDA (laskennallisesti älykkäistä menetelmistä koostuva ohjelmisto, jota tehty sovellus käyttää) ja
- Jyväskylän yliopistossa tehty tutkimustyö (mm. Erkki Häkkisen tekemä väitöskirja [15] ja edellä mainittu NDA)

Haluan kiittää Pasi Koikkalaista gradun hyvästä ohjauksesta. Kiitokset on myös syytä antaa Anssi Lensulle, joka auttoi huomattavasti NDA:han tutustumisessa ja sen ohjelmoinnissa. Haluan myös kiittää koko tutkimusryhmäämme erinomaisesta työviihtyvyydestä ja auttamisesta ongelmatilanteissa. Lopuksi kiitän vielä Pasi Pielaa mm. NDA sovelluksen ongelmien ilmoittamisesta ja imputointitulosten vertailusta.

Tiivistelmä

Pro gradu- tutkielma käsittelee virheellisen ja puuttuvan tiedon editointia ja imputointia. Editointi ja imputointi ovat erittäin tärkeitä tehtäviä kun halutaan hyödyntää virheellistä ja/tai puutteellista havaintoaineistoa. Ilman havaintojen täydentämistä ja korjausta kyseistä havaintoaineistoa ei voida käyttää yleensä kovin hyvin. Mitä enemmän virheitä ja puuttuvuuksia on, sitä tärkeämmäksi editointi- ja imputointitehtävät tulevat. Tutkielmaan sisältyy editointi- ja imputointisovelluksen määritteleminen ja toteuttaminen sekä sen soveltaminen EurEdit datoihin.

Luvussa 1 kuvataan ongelmaa ja kerrotaan aiheesta taustatietoja. Mallintamista on käsitelty luvussa 2. Sitä hyödynnetään tehdyssä sovelluksessa havaintoaineiston jakauman- ja imputointimallin/-mallien luomiseen. Luvussa esitellään myös mallin validointi ja Bootstrap-validointimenetelmä. Virheiden havaitsemisesta ja korjaamisesta syntaktisilla- ja mallipohjaisilla menetelmillä on kerrottu luvussa 3, jossa esitellään myös robustit menetelmät. Luvussa 4 käsitellään puuttuvuusmekanismeja, yksikkö- ja moniimputointia, imputointimenetelmiä ja puuttuvan tiedon täydentämistä. Tämän lisäksi kerrotaan ryvästelymenetelmien ja itseorganisoituvan piirrekartan käyttämisestä imputointiin. Luvussa 5 esitellään tehty ohjelmisto, joka mahdollistaa sääntöpohjaisen editoinnin ja imputoinnin eri menetelmillä. Ohjelmiston tavoitteet, käyttöliittymän spesifikaatio, arkkitehtuuri, käyttöliittymä ja toiminnallisuus ovat kuvattu luvussa. Näiden lisäksi on myös esitetty testitulokset, arviointi sovelluksesta ja sen tulevaisuudesta, eli jatkokehityksestä. Johtopäätökset on tehty luvussa 6.

Abstract

The thesis contains information about editing and imputing of erroneous and missing data. Edit and imputation tasks are very important when one wants to use such data. Without imputing and correcting errors the data cannot usually be used well. The more the data has missingness and errors the more important become the edit and imputation tasks. The thesis includes specifying and implementing of edit and imputation application and using of EurEdit datas with it.

In chapter 1 the problem is described and background information is given. Chapter 2 is about modelling which is used in application when doing modelling of data distribution and building imputation model(s). In addition to these validation and Bootstrap-method are described. Chapter 3 is about detecting and correcting errors with syntactical and model based methods. The chapter also includes robust methods. Imputation, including missingness mechanisms, single and multiple imputation and different imputation methods are covered in chapter 4. Using clustering methods and self-organizing map for imputation is described in the chapter too. The application made is presented in chapter 5. The application allows syntactical editing and imputation with various methods. The requirements, user interface specification, architecture, user interface and funtionality of the application are presented in the chapter. Test results, future (further development) and evaluation of the application are also told. Conclusions are drawn in chapter 6.

Sisältö

TERMILUETTELO

Populaatio
Äärellinen tai ääretön joukko havainnoitavia kohteita. Voi olla havaintoavaruuden osajoukko.

Otos
Osajoukko havaintoavaruudesta tai populaatiosta poimittuja havaintoja. Satunnaisessa poiminnassa pyritään valitsemaan näytteet toisistaan riippumattomasti samalla satunnaisperiaatteella, jolloin otosta voidaan käyttää kuvaamaan todellista populaatiota.

Data
Yleisessä mielessä mielivaltainen havaintojoukko. Usein data on sama kuin otos.

Havainto
Yhden yksilön ilmentymä. Tietojenkäsittelyn näkökulmasta tietue, joka voi sisältää useita kenttiä eli havaintomuuttujia.

Virheellinen havainto
Havainto, josta jokin osa on virheellinen, esimerkiksi mahdoton, tai totuuden vastainen.

Puutteellinen havainto
Havainto, josta puuttuu jokin osa, esimerkiksi muuttuja.

Muuttuja
Jokainen mitattava ominaisuus voidaan tulkita muuttujana. Tällainen on esimerkiksi ihmisen pituus.

Riippuva muuttuja
Jos muuttujan vaikutus tutkittavaan kohteeseen riippuu myös muista muuttujista, niin kyseessä on riippuvainen muuttuja.

Riippumaton muuttuja
Riippumattomassa tapauksessa muuttujan vaikutus tutkittavaan kohteeseen ei riipu muiden muuttujien arvoista.

Jakauma
Todennäköisyysjakauma, jota satunnaismuuttuja noudattaa.

Malli
Tiivistetty, usein matemaattinen kuvaus jonkin tutkittavan ilmiön käyttäytymisestä. Tilastollinen malli on yleensä matemaattinen esitystapa tutkittavan ilmiön jakaumalle.

Mallin sovittaminen
Mallin sovittaminen havaintoaineistoon voidaan tehdä monella tavalla. Jos malli on tilastollinen ja parametrinen, niin sovitus yksinkertaistuu sopivien parametrien määräämiseen siten, että malli selittää havaintoaineiston.

Neuroverkko
Eräs tilastollisten mallien perhe, jolla alunperin pyrittiin mallintamaan biologisten hermoverkkojen toimintaa.

Neuroni
Neuroverkon perusyksikkö.

Ylioppiminen
Ylioppimisella tarkoitetaan, sitä että neuroverkko oppii opetusaineiston liian hyvin (oppii datan sisältämän kohinan) ja sen yleistyskyky heikkenee.

Itseorganisoituva piirrekartta
Itseorganisoituva piirrekartta on neuroverkkomenetelmä, jolla voidaan kuvata alkuperäisen havaintojoukon näytteet alempiulotteiseen avaruuteen. Alunperin lähellä toisiaan olevat näytteet ovat lähellä toisiaan myös itseorganisoituvan piirrekartan muodostamassa kuvauksessa.

Editointi
Havaintoaineiston virheiden tunnistamista ja korjaamista.

Validisuus eli ristiriidattomuus
Tarkoittaa yleisesti hyvin muodostettua loogista järjestelmää. Se on laadun looginen käsite sekä mallien että havaintoaineiston hyvyyttä tarkasteltaessa.

Säännöt
Sääntöpohjaisilla järjestelmillä tarkoitetaan yleensä symbolisiin lauseisiin tai loogisiin epäyhtälöihin perustuvia järjestelmiä, joiden pohjalta voidaan tehdä päättelytehtäviä. Editoinnin yhteydessä säännöillä voidaan pyrkiä havainnoimaan havaintojoukossa olevia virheitä.

Imputointi
Puuttuvien havaintojen täydentämistä.

Robustisuus
Tämä tarkoittaa yleensä mallin kykyä sietää virheitä.

Singulaarinen
Tarkoittaa yleensä, että mallille ei ole yksikäsitteistä ratkaisua. Esimerkiksi matriisi on singulaarinen, kun sen determinantti on nolla.

Yksikkömatriisi
Yksikkömatriisin diagonaalialkiot ovat ykkösiä ja kaikki muut alkiot nollia.

Työssä käytettävät symbolit

Symboli	**Merkitys**
X	Satunnaismuuttuja.
x	Muuttuja, joka voi saada reaaliarvoja eli $x \in \mathbf{R}$.
$\mathbf{X}$	Satunnaisvektori, joka koostuu useasta satunnaismuuttujasta.
$\mathbf{x}$	Vektorimuuttuja, joka voi saada vektoriarvoja, eli $\mathbf{x} \in \mathbf{R}^d$.
$\mathbf{A}$	Matriisi $\mathbf{A}$ on taulukko, jossa on i riviä ja j saraketta eli yhteensä $i \times j$ alkiota.
$\mathbf{A}^{\mathrm{T}}$	Matriisin $\mathbf{A}$ transpoosi on matriisi, jossa rivit ja sarakkeet on vaihdettu keskenään.
$\mathbf{A}^{-1}$	Matriisin käänteisoperaatio, jolle pätee että $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$, jossa $\mathbf{I}$ on yksikkömatriisi. Matriisin $\mathbf{A}$ on oltava ei-singulaarinen, jotta käänteismatriisi $\mathbf{A}^{-1}$ on olemassa.
$X\|Y$	Muuttuja X ehdolla Y, jossa Y on tunnettu taustatieto.
$E(X)$	Muuttujan X odotusarvo.
$\mathbf{X}_{\text{missing}}$	Osa vektorista $\mathbf{X}$, joka voi sisältää puuttuvuutta.
$\mathbf{X}_{\text{observed}}$	Osa vektorista $\mathbf{X}$, joka ei sisällä puuttuvutta.
D	Havaintojoukko, joka koostuu vektorihavainnoista $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ tai skalaarihavainnoista $x_1, x_2, \ldots, x_n$ datan dimension ollessa 1.
$\mathbf{D}$	Datamatriisi.
$\mathbf{D}_{\text{observed}}$	Kokonaan havaittu data.
$\mathbf{D}_{\text{missing}}$	Datan osa, jossa esiintyy puutteellisuutta.
$\mathbf{M}$	Datan puuttuvuuden indikaattori matriisi.
Pr(A)	Tapahtuman A todennäköisyys.
$p_x(x)$	Diskreetin satunnaismuuttujan X tiheysfunktio.
$f_x(x)$	Jatkuvan skalaarimuuttujan X tiheysfunktio.
$f_{x_i}(x_i)$	Jatkuvan vektorimuuttujan $\mathbf{X}$ i:n komponentin tiheysfunktio.
M_d	Mediaani eli kohta, jonka molemmin puolin on 50%:a datan massasta.
$\mathcal{Q}_1$	Alakvartiilin alapuolella on 25%:a datan massasta.
$\mathcal{Q}_2$	Sama kuin mediaani eli M_d.
$\mathcal{Q}_3$	Yläkvartiili. Tämän yläpuolella on 25%:a datan massasta.
$\mathcal{Q}I$	Kvartiiliväli on alakvartiilin ja yläkvartiilin väli eli $\mathcal{Q}I = (\mathcal{Q}_1, \mathcal{Q}_3)$.

Symboli	Merkitys
$\mathcal{Q}_d$	Kvartiilipoikkeama on puolet ylä- ja alakvartiilin välisestä erotuksesta eli $\mathcal{Q}_d = (\mathcal{Q}_3 - \mathcal{Q}_1)/2$.
μ	Keskiarvo eli μ ($=\overline{x}$ satunnaismuuttujalle X) määrittelee havaintojen keskiarvon.
σ^2	Varianssi eli σ^2 kuvaa datan jakauman levittäytyneisyyttä, σ on keskihajonta.
s^2	Otosvarianssi on havaintoaineistosta laskettu varianssi, s on keskihajonta.
Q	Tilastollisen tunnusluvun satunnaismuuttuja.
q	Tilastollinen tunnusluvun muuttuja q (esimerkiksi keskiarvo μ).
$\hat{q}$	$\hat{q}$ on tilastollisen tunnusluvun q estimaatti eli arvio.
$\hat{q}^{(i)}$	Tilastollisen tunnusluvun q i:s estimaatti.
ϵ	Satunnainen virhe datassa, jonka oletetaan yleensä olevan normaalijakautunutta ja odotusarvon $E(\epsilon)$ olevan nolla.
$\mathbf{\Delta}_\mathrm{c}$	Vakiovirhematriisi.
$\mathbf{\Delta}_\epsilon$	Satunnaisvirhematriisi.
$\in$	Merkinnällä $x \in D$ tarkoitetaan että havainto x kuuluu joukkoon D. Merkinnän vastakohta on $\notin$, joka tarkoittaa joukkoon kuulumattomuutta.
$\subset$	$D' \subset D$ tarkoittaa, että D' on joukon D osajoukko.
θ	Parametrijoukko.
$\sim$	Satunnaismuuttujan X noudattaessa jakaumaa $D(\theta)$ merkitään $X \sim D(\theta)$, jossa θ on jakauman parametrit.
x^{-1}	Käänteisoperaatio, jolle pätee että $xx^{-1} = 1$, jossa x on skalaarimuuttuja ja $x \neq 0$.
$\|\|$	Normi eli etäisyyden mitta: esimerkiksi $\|\|\mathbf{a} - \mathbf{b}\|\|$ on vektorin **a-b** euklidinen pituus.
$\cap$	Joukkojen A ja B leikkausta merkitään $A \cap B$:llä. Leikkaukseen kuuluvat kaikki alkiot, jotka kuuluvat sekä A että B joukkoihin.
$\#$	Joukon mahtavuus eli kardinaliteetti, joka on siis joukon alkioiden lukumäärä.
$\forall x$	Kaikilla muuttujan x arvoilla.
$[a, b]$	Suljettu väli, jonka päätepisteet kuuluvat välille.

Symboli	Merkitys
(a, b)	Avoin väli, jonka päätepisteet eivät kuulu välille.
$[a, b)$	Puoliavoin väli, jossa a kuuluu ja b ei kuulu välille.
$(a, b]$	Puoliavoin väli, jossa a ei kuulu ja b kuuluu välille.
$\exp(x)$	Exponenttifunktio e^x, jossa e on Neperin luku.
$f(x)$	$f(x)$ on funktio, joka kuvaa lähtöjoukon arvon x maalijoukon arvoksi $y = f(x)$.
$\arg\min_x f(x)$	x:n arvo, jolla $f(x)$ minimoituu.
$\min(x_1, x_2, \ldots, x_n)$	Joukon minimi eli palauttaa minimiarvon annetusta joukosta.
m	Moni-imputointien lukumäärä.
$\overline{Q}$	m:n imputoinnin keskiarvo tutkittavan suureen arvolle.
$L(x)$	Uskottavuusfunktio, jota käytetään etsittäessä uskottavimpia eli todennäköisimpiä mallin parametrien arvoja.
δ	Raja-arvo parametrien muutoksen merkittävyyden testaamiseen.

NDA tietomallin symbolit

Symboli	Merkitys
f	Kenttä eli muuttuja, jossa on n havaintoa.
f_{int}	Kokonaislukukenttä.
f_{float}	Liukulukukenttä.
f_{string}	Merkkijonokenttä.
d	Data, joka koostuu i:stä kentästä.
c	Luokkaindeksit, jotka luokittelevat havaintoaineiston ryppäisiin.
s_{SOM}	SOM-tietorakenne.
s_{G}	Grafiikkarakenne.

Luku 1

Ongelman kuvaus

Tämän työn aiheena on virheellisen ja puutteellisen havaintoaineiston käsittely laskennallisesti älykkäiden järjestelmien avulla.

Reaalimaailman havainnot ovat usein virheellisiä tai puutteellisia, mikä aiheuttaa paljon ongelmia havaintojen pohjalta tehtävissä sovelluksissa. Editoinnilla pyritään tunnistamaan ja korjaamaan virheellistä osaa havainnoista. Vaikka virheet onnistuttaisiinkin havaitsemaan, niitä ei välttämättä osata korjata vaan havainnot joudutaan poistamaan. Pahimmassa tapauksessa virheitä ei pystytä havaitsemaan, vaan ne jäävät havaintoaineistoon ja aiheuttavat tuntemattoman virhelähteen aineistosta tehtäviin johtopäätöksiin. Imputoinnilla pyritään täydentämään puutteellista havaintoaineistoa. Imputoidessa on myös otettava huomioon, ettei prosessi tuota virheellistä dataa. Editointi ja imputointi edellyttävät tilastotieteen ja matematiikan osaamista. Niiden lisäksi tarvitaan usein myös tietotekniikan taitoja, esimerkiksi jos tavoitteena on tehdä automatisoituja ohjelmia suurten tietomassojen automaattiseen puhdistamiseen. Manuaalistakin, ihmisen suorittamaa, editointia ja imputointia on mahdollista soveltaa rajoitetuissa sovelluksissa. Se on tosin erittäin hidasta, jos käsiteltävä havaintoaineisto on suuri tai monimutkainen.

Työn ensimmäisessä vaiheessa editoitavat ja imputoitavat kentät on tunnistettava. Jos data vaatii sekä editointia että imputointia niin yleensä editointi on suoritettava ensin. Yleisesti ottaen tämä ei ole helppo ongelma. Virheellisen datan havaitsemista auttaa

jos tiedetään taustatietoja, kuten että

- havainnot ovat tietyllä alueella havaintoavaruudessa, tai
- havaintojen komponentit muodostavat helposti tulkittavan, toisistaan riippuvan kokonaisuuden (esimerkiksi perheen tulot ovat perheenjäsenien tulojen summa).

Havaintoaineisto voidaan jakaa neljään täydellisyysluokkaan, jotka ovat:

- täydellinen, missä ei ole virheellisyyksiä tai puuttuvuuksia,
- virheellinen, jossa osa havainnoista on virheellisiä,
- puutteellinen, jossa osa havainnoista on puutteellisia, sekä
- virheellinen ja puutteellinen, jossa esiintyy molempia edellämainittuja ongelmia.

Oletetaan esimerkin vuoksi, että havaintoaineisto koostuu kyselytutkimuksesta, jossa neljältä henkilöltä on kysytty vastausta neljään kysymykseen X_1, X_2, X_3, X_4. Kysymyksiin vastataan positiivisella kokonaisluvulla. Ideaalitilanteessa datassa ei ole virheitä eikä puutteita (oletetaan, että kuvan 1. tilanne edustaa tätä tapausta).

Muuttuja		X_1	X_2	X_3	X_4
Havainto	1	1	2	5	3
	2	9	11	17	7
	3	24	6	4	14
	4	18	19	22	33

Kuva. 1: Täydellinen data, jossa ei ole virheitä tai puutteita

Havaintoaineisto on virheellinen, kun siinä on havaintoja, jotka eivät ole sallittuja tai ovat totuuden vastaisia. Jos esimerkissämme tiedettäisiin, että vain positiiviset arvot ovat mahdollisia, niin kuvan 2. tilanne edustaa helposti tunnistettavia virheitä.

Muuttuja		X_1	X_2	X_3	X_4
Havainto	1	1	**-2000**	5	3
	2	9	11	**-1700**	7
	3	**-2400**	6	4	14
	4	18	19	22	**-3300**

Kuva. 2: Virheellinen data

Haastattelututkimuksen tapauksessa puutteellista dataa syntyy esimerkiksi silloin, kun henkilö jättää vastaamatta kyselyn joihinkin kohtiin. Tätä on havainnollistettu kuvassa 3. Havaintoaineistoa kerättäessä puutteellinen data on yleensä merkitty, esimerkiksi sen arvoa ei ole (se on tyhjä) tai se on -9 (olettaen että -9 ei esiinny itse datassa).

Muuttuja		X_1	X_2	X_3	X_4
Havainto	1	1	?	5	?
	2	9	11	17	7
	3	24	?	4	14
	4	?	19	22	33

Kuva. 3: Puutteellinen data

Pahimmassa tapauksessa datassa on virheitä sekä puutteita (kuva 4).

Muuttuja		X_1	X_2	X_3	X_4
Havainto	1	?	2	5	3
	2	**-900**	11	?	**-700**
	3	?	6	4	14
	4	18	**-190**	?	33

Kuva. 4: Virheellinen ja puutteellinen data

1.1 Virheiden ja puutteellisuuksien tunnistaminen ja korjaaminen

Käytännössä virheiden havainnointia helpottaa, jos on mahdollista muodostaa sääntöjä havaintojen oikeellisuuden tarkistamiseksi. Tällainen sääntö on esimerkiksi, että kentässä "sukupuoli" arvo saa olla vain "N" tai "M". Käytännössä sääntöjä ei välttämättä ole saatavilla. Tällöin säännöt on muodostettava itse datasta tai sitten korjattavat havainnot on tunnistettava muilla keinoin. Vieraiden havaintojen tunnistaminen käyttäen nk. robusteja menetelmiä on usein varsin käytännöllinen menetelmä. Sen avulla virheellisiksi määrätään ne havainnot, jotka poikkeavat "liian paljon" havaintojen tyypillisistä arvoista.

Tämäkään lähestymistapa ei ole ongelmaton, sillä monissa reaalimaailman havainnoissa on erikoisia, mutta täysin totuudenmukaisia havaintoja. Esimerkiksi Suomen yritysrekisterissä Nokian kaltainen suuryritys poikkeaa varmuudella kaikista tilastollisista malleista, jotka rakennetaan ilman erikseen määrättävää taustatietämystä. Virheiden ja puutteiden tunnistamisen jälkeen data on korjattava. Korjaaminen ei ole välttämättä helppoa, varsinkin jos data on monimuuttujaista, rakenteista ja jos sen määrä on suuri. Menetelmiä datan korjaamiseen on useita. Useimmat menetelmät perustuvat jonkinasteiseen taustatietoon havaintoaineiston luonteesta. Tieto voi olla rakenteista (symbolista) tai se voi perustua tilastolliseen malliin havaintojen jakaumista. Korjaamisen jälkeen uusi data tai sen tuottamiseen käytettävä menetelmä on validoitava, jotta varmistutaan sen käyttökelpoisuudesta.

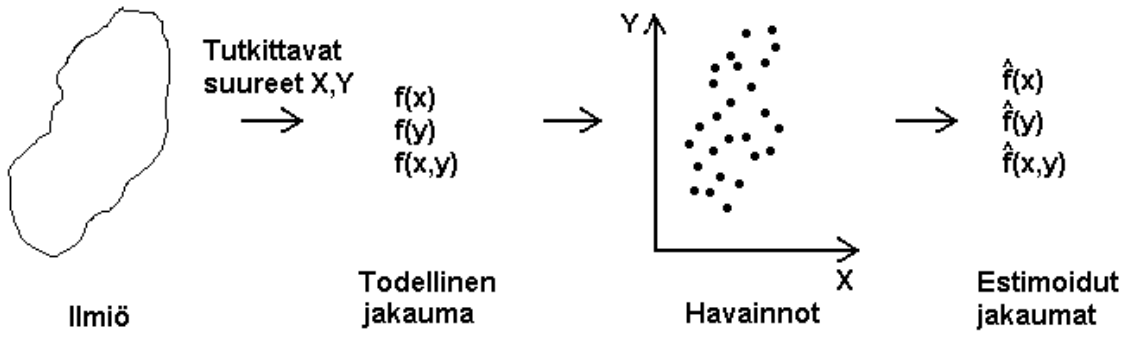
1.2 Havainnot ja jakaumat

Havaintoaineistoja voidaan hankkia usealla tavalla. Esimerkkejä tyypillisistä lähteistä ovat

- tilastokeskuksen kyselytutkimukset sekä rekisteritiedot kuten vero- ja väestörekisterit,

- erilaiset mittaukset, kuten teollisuusprosesseista instrumentoinnin avulla saadut havainnot, tai
- simuloinnin avulla generoidut datajoukot.

Usein havaintoja käyttäen pyritään muodostamaan malli tutkittavasta ilmiöstä. Tilastollisessa tarkastelussa tätä kutsutaan estimoinniksi, jota on havainnollistettu kuvassa 5. Malli pystyy usein selittämään vain osan ilmiöstä.



Kuva. 5: Ilmiön mallintaminen jakaumilla, jotka saadaan havainnoista

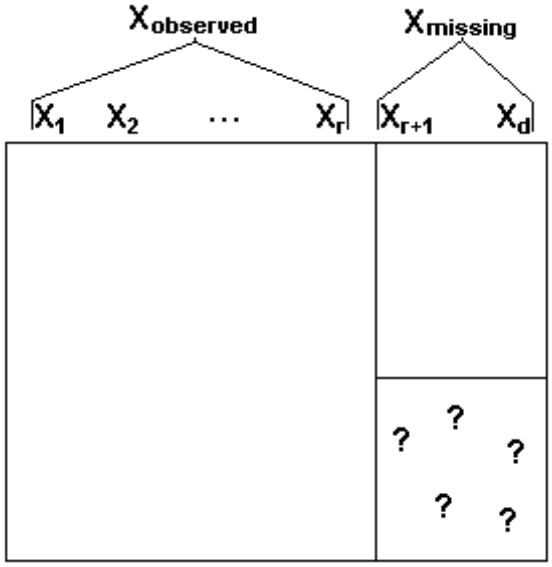
Havaintojoukosta D voidaan muodostaa satunnaismuuttujien $X_1, X_2, \ldots, X_r$ tiheysjakaumat $f_{X_j}(x_j), j = 1 \ldots r$. Käytännössä tiheysjakauman approksimaatio voidaan muodostaa esimerkiksi histogrammina. Sitä muodostattaessa tutkitaan alue millä data on, minkä jälkeen alue jaetaan keskenään samankokoisiin osa-alueisiin. Prosessin jälkeen lasketaan kuinka monta havaintoa osuu kuhunkin osa-alueeseen. Lukumäärät ovat frekvenssejä ja niiden lukumäärän summalla normeerataan histogrammitiheysfunktion integraali ykköseksi. Tiheysjakaumien $f_{X_j}(x_j)$ muodostamisen jälkeen tutkitaan ovatko ne yleisiä, hyvin tunnettuja tiheysjakaumia, kuten normaali-, Poisson-jakaumia jne. Vertaamisen tekemiseen on olemassa tilastollisia testejä. Kolmogorov-Smirnovin Goodness-Of-Fit [24] ja mm. sen muunnelmalla Anderson-Darling testeillä [25] voidaan tutkia noudattaako otos oletettua jakaumaa. Perusjoukon normaalisuuden testaamiseen on olemassa useita testejä [4]. Testeillä on omat heikkoutensa ja vahvuutensa. Esimerkiksi Kolmogorov-Smirnovin Goodness-Of-Fit testi toimii vain jatkuville jakaumille. Toisaalta sen etuna on että testisuureen jakauma ei riipu testattavasta kertymäfunktiosta.

Muuttujien arvojen todennäköisyyden ennustamiseen voidaan käyttää yhteisjakaumaa $f(X_1, X_2, \ldots, X_r)$. Se muodostetaan ehdollistamalla seuraavasti

$$f(X_1, X_2, \ldots, X_r) = \prod_{j=1}^{r} f(X_j | X_{j-1}, \ldots, X_1).$$

Käytännössä yhteisjakauman muodostaminen on vaikeaa. Tehtävää voidaan helpottaa esimerkiksi taustaoletuksella että jakauman funktiomuoto tunnetaan. Usein havaintoaineisto oletetaan normaalijakautuneeksi.

Tässä työssä käsiteltävät aineistot ovat monimuuttujaisia, eli data koostuu havaintovektoreista, joissa on useampia arvoja. Kuten tilastotieteessä yleensä dimensioltaan d-ulotteisen havaintovektorin (havaintopisteen) $\mathbf{x} \in \mathbf{R}^d$ yhteys jakaumiin määräytyy satunnaisvektorin $\mathbf{X}$ kautta siten, että havainnon (numero i) arvo on $\mathbf{x}$, eli $\mathbf{X}(i) = \mathbf{x}$. Siis i edustaa erästä näytettä $\mathbf{X} \sim f_{\mathbf{X}}(\mathbf{x})$, joka on poimittu jakaumasta $f_{\mathbf{X}}(\mathbf{x})$. Puuttuvuus ilmenee tällöin vektorin $\mathbf{x}$ komponenteissa, jotka voidaan jakaa oheisen kuvan 6 tavoin täysin havaittuihin $\mathbf{X}_{\text{observed}}$ ja niihin kenttiin, joissa voi esiintyä puutteellisuutta $\mathbf{X}_{\text{missing}}$. Imputointitehtävä tarkoittaa tällöin puutteellisen osan täydentämistä havaitulla osalla.



Kuva. 6: Datan täysin ja osittain havaitut muuttujat

Editointi ja imputointitehtäviä ajatellen olisi tärkeää tietää muuttujien ominaisuuksia kuten:

- muuttujien sallitut arvoalueet
- muuttujien väliset riippuvuudet: riippuvia vai riippumattomia
- muuttujien tyypit: diskreettejä vai jatkuvia
- muuttujien noudattama yleinen todennäköisyysjakauma, jos sellainen on olemassa

Käytännössä näitä kaikkia tietoja ei välttämättä saada datan mukana, vaan tiedot on pääteltävä datasta, mikä voi olla vaikeaa.

Havaintoarvot voivat olla joko jatkuvia tai diskreettejä. Diskreetti data jakautuu edelleen neljään luokkaan, jotka ovat:

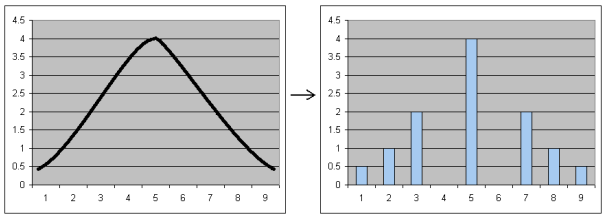
1. Luokittelu- eli nominaaliasteikollinen
   Luokitteluasteikollisen, eli kategorisen, muuttujan arvot edustavat tilanteita tai tapahtumia, joita ei voida asettaa järjestykseen. Tyypillinen luokittelumuuttuja on sukupuoli(mies tai nainen).
2. Järjestys- eli ordinaaliasteikollinen
   Järjestysasteikolliset, eli ordinaaliset, muuttujat voidaan asettaa järjestykseen. Tällainen mitattava muuttuja voi olla esimerkiksi mielipide: 0=täysin eri mieltä, 1=eri mieltä, 2=samaa mieltä ja 3=täysin samaa mieltä. Lukujen erotuksilla ei ole tulkintaa ja ne eivät ole tasavälisiä.
3. Välimatka- eli intervalliasteikollinen
   Välimatka-asteikolliset muuttujat voidaan järjestellä ja niiden välisillä erotuksilla on tulkinta. Niillä ei ole absoluuttista nollakohtaa. Esimerkiksi syntymävuosi on tällainen muuttuja.
4. Suhdeasteikollinen
   Suhdeasteikollisillta muuttujilla on absoluuttinen nollakohta, jossa sen määre häviää. Myös muuttujien arvojen suhteella on merkitys. Tällainen muuttuja on esimerkiksi ihmisen pituus.

Asteikot ovat tärkeitä mm. korjattaessa tai imputoidessa dataa koska editointi- tai imputointimalli saattaa käsitellä esimerkiksi diskreettiä muuttujaa jatkuvana. Tästä voi

seurata se, että puutteelliselle muuttujalle täydennetään arvoja jotka ovat mahdottomia. Esimerkissämme on tällöin mietittävä miten jatkuva arvo muutetaan diskreetiksi. Editointia ja imputointia ajatellen välimatka- ja suhdeasteikolliset muuttujat käsitellään usein jatkuvina ja tarvittaessa katkaistaan lopulta kokonaislukuarvoisiksi. Luokittelu- ja järjestysasteikollisten muuttujien jokaista kategoriaa varten on muodostettava oma todennäköisyysmuuttuja. Toinen vaihtoehto on määrätä jokin koodaustapa, joka ei tee riippuvuuksia eri luokkien välille.

1.2.1 Jatkuvan muuttujan diskretisointi

Joissakin menetelmissä on tarpeen muuntaa jatkuva-arvoiset havainnot diskreeteiksi. Muunnos voidaan tehdä jakamalla jatkuvan muuttujan arvoalue diskreetteihin väleihin, eli luokkiin kuvan 7 esittämällä tavalla. Yleensä jako on tasavälinen eli välien pituus on vakio. Esimerkiksi väli $[a, b]$ jaetaan n:ään luokkaan, joiden jokaisen pituus on $(b-a)/n$.



Kuva. 7: Diskretisointi

Esimerkki 1.2.1 *Jos jatkuvan muuttujan X arvot ovat välillä [0,100] voidaan luokittelu tehdä esimerkiksi seuraavasti:*

$$y = \begin{cases} 1, & jos\ x < 10 \\ 2, & jos\ x \in [10, 30) \\ 3, & jos\ x \in [30, 50) \\ 4, & jos\ x \in [50, 80) \\ 5, & jos\ x \geq 80 \end{cases}$$

Näin muodostuu siis uusi muuttuja Y, joka on diskreetti versio muuttujasta X.

Toinen tapa jatkuvan muuttujan diskretisointiin on sen katkaiseminen kokonaisluvuksi (algoritmi 1). Tämä toimii usein hyvin kun muuttuja on välimatka- tai suhdeasteikollinen.

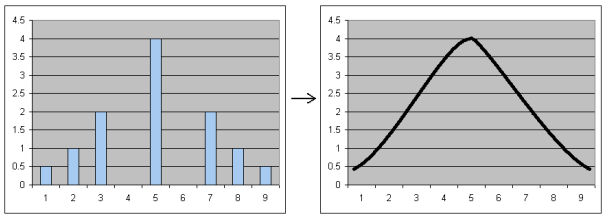
Algoritmi 1 *(Diskretisointi)*
Oletuksena on että kaikille muuttujan X arvoille x pätee: $x = n + r$, jossa $n \in \mathbf{N}$ ja $r \in [0, 1) \subset \mathbf{R}$. Diskretisointi on kuvaus:

$$y = \begin{cases} n, & jos\ r < 0.5 \\ n + 1, & jos\ r \geq 0.5, \end{cases}$$

missä $y \in \mathbf{N}$.

1.2.2 Diskreetin muuttujan muuntaminen jatkuvaksi

Vastaavasti diskreetti muuttuja voidaan muuttaa jatkuvaksi sovittamalla siihen jatkuva malli, mitä on havainnollistettu kuvassa 8. On tosin otettava huomioon, että jatkuva mallin oletukset ovat mahdollisimman hyvin voimassa. Muunnoksen etuna on, että käytettävä menetelmä voi hyödyntää myös diskreettien arvojen välistä aluetta.



Kuva. 8: Diskreetin muuttujan muuttaminen jatkuvaksi

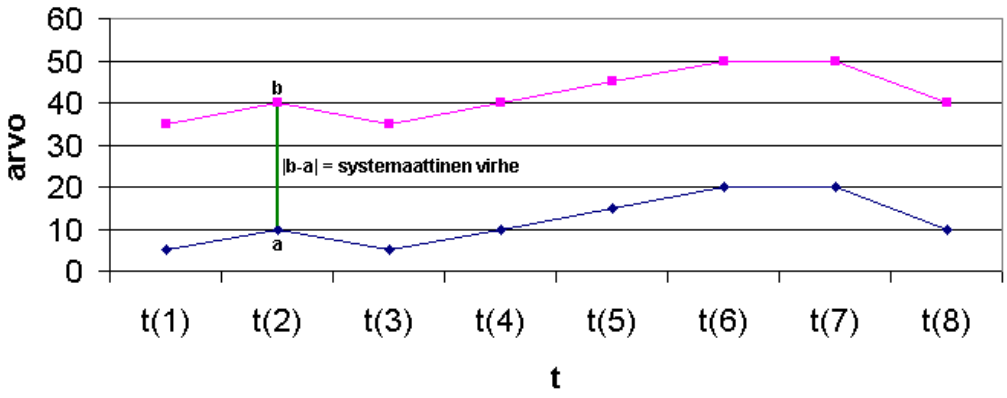
1.3 Datan hankinnan ongelmat

Johdatuksena virheellisen ja puuttuvien havaintojen käsittelyyn tarkastelemme seuraavassa muutamia tyypillisiä virheiden ja puuttuvuuksien syntymekanismeja.

Virhelähteet voidaan pääsääntöisesti jakaa kahteen ryhmään:

- systemaattisiin virheisiin ja
- satunnaisiin virheisiin.

Systemaattisen virheen taustalla on jokin tiedonhankinnasta tai tietojenkäsittelystä johtuva virhe tai väärinkäsitys. Esimerkiksi vastaus annetaan eurojen sijasta markkoina. Usein systemaattinen virhe on toistuva eli jokaiseen havaintoon tulee vakiovirhe, joka johtuu samoista tekijöistä (kuva 9). Se on monesti myös "samansuuntainen", ollen muotoa: $\mathbf{D}_{\text{observed}} = \mathbf{D} + \mathbf{\Delta}_{\text{c}}$, jossa $\mathbf{D}$ on alkuperäinen data ja $\mathbf{\Delta}_{\text{c}}$ on systemaattinen vakiovirhematriisi. Systemaattiset virheet on yleensä helppo havaita ja ne ovat monesti poistettavissa. Esimerkiksi auton nopeusmittarin lukemat voivat olla systemaattisesti 30 yksikköä liian suuria. Aina ongelma ei ole näin helppo, sillä systemaattinen virhe voi olla riippuvainen datan arvoista tai jopa aiemmista systemaattisista virheistä, jolloin virheen tunnistaminen ja korjaaminen voi olla erittäin vaikeaa tai jopa mahdotonta.

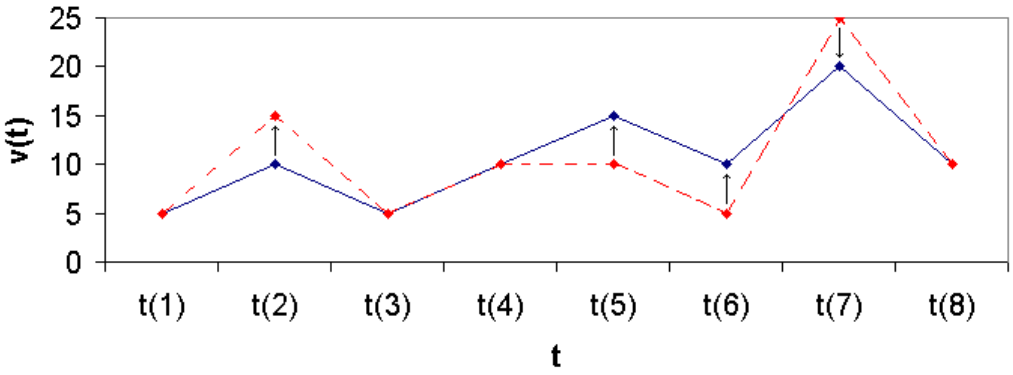


Kuva. 9: Systemaattinen virhe datassa (alempi kuvaaja on alkuperäinen data)

Satunnaisvirheellinen data (kuva 10) on usein muotoa $\mathbf{D}_{\text{observed}} = \mathbf{D} + \mathbf{\Delta}_{\epsilon}$, jossa $\mathbf{D}$ on alkuperäinen data ja $\mathbf{\Delta}_{\epsilon}$ on satunnainen virhematriisi. Monesti tilannetta helpottaa, jos satunnaisvirheiden oletusarvo voidaan olettaa nollaksi eli $E(\Delta_{\epsilon ij}) = 0 \ \forall i, j$. Tällöin mallinnuksessa voidaan käyttää suosittuja regressiomenetelmiä. Satunnaisia virheitä voi olla vaikeampi havaita kuin systemaattisia virheitä, joskin tämä on tapauskohtaista. Tyypillisiä satunnaisvirheiden tuottajia ovat häiriölliset mittaukset, satunnaisin väliajoin vikautuvat koneet ja muut tiedon tuotantoprosessiin vaikuttavat epätarkkuudet.

Esimerkiksi tilastokeskusten haastattelututkimuksien yhteydessä syntyy usein satunnaisia virheitä siirrettäessä vastauslomakkeiden tietokenttiä digitaaliseen muotoon.

Kun virhe poikkeaa selvästi muun datan käyttäytymisestä sanotaan, että se on "outlier" eli havaintopoikkeama. Tämä on periaatteessa mahdollista tunnistaa jopa yksittäisen havainnon osalta. Vaikeampaa on, jos virhe tuottaa sinänsä kelvollisen, mutta väärän havainnon. Tämäntyyppisiä "inlier" havaintoja ei voida tunnistaa yksilötasolla, mutta niiden esiintymiä saattaa näkyä muutoksena populaation jakautumisessa.



Kuva. 10: Satunnaisia virheitä datassa, alkuperäinen data on merkitty katkoviivalla

Havaintoaineiston puutteellisuuden takana voi olla useita tekijöitä. Havaintoaineiston keräysvaiheessa ei saada kaikkea tietoa. Esimerkiksi osa henkilöistä jättää kokonaan vastaamatta kyselyyn tai vastaa kyselyyn ainoastaan osittain. Ensimmäistä tapausta kutsutaan yksikköpuuttuvuudeksi ja toista muuttujapuuttuvuudeksi. Puuttuvuus voi syntyä myös tiedon myöhemmässä käsittelyvaiheessa virheiden tai laitehäiriöiden seurauksena. Kolmas tilanne syntyy virheiden tunnistamisen yhteydessä, kun jokin havainto tai sen osa tunnistetaan virheelliseksi, mutta sitä ei osata korjata. Tällöin on tapana merkitä virheellinen havainto puuttuvaksi.

Luku 2

Mallintaminen

Tämä luku on lyhyt johdatus tilastollisiin mallinnusmenetelmiin, joihin työn taustalla oleva imputointi- ja editointimetologia perustuu. Mallintamisella on keskeinen rooli kaikessa tilastollisen datan pohjalta tehtävässä analysoinnissa ja päättelyssä. Mallintamisen tavoitteena on tiivistää tutkittavan kohteen käyttäytyminen mahdollisimman yksinkertaiseen, mutta tehtävän kannalta riittävän informatiiviseen esitykseen. Tilastollisessa mallintamisessa tehtävä on perinteisesti tulkittu parametrisoidun jakauman sovittamiseksi havaintoaineistoon. Fisheriläisen ajattelutavan mukaisesti tämä pelkistyy mallia kuvaavien parametrien estimoimiseksi havaintoaineiston avulla.

Puutteellisen ja virheellisen tiedon käsittelyssä tilastollisten mallien sovellusmahdollisuudet ovat laajat. Tavoitteena on yleensä laatia malli, joka selittää puuttuvat tai virheelliseksi oletettavat havainnot taustatiedon pohjalta. Tällä kuten millään muullakaan tavalla ei ole mahdollista kuvata puuttuvia arvoja yksikkötasolla oikeiksi. Sen sijaan tärkein tavoite, koko populaatiojakauman yli laskettava informaatio, on useissa tapauksissa täydennettävissä siten, että tärkeät tunnusluvut kuten keskiarvot, summa ym. ovat riittävässä määrin luotettavia.

Mallinnusmenetelmät voidaan karkeasti ottaen jaotella kahdella tavalla:
1) Parametriset vs. epäparametriset menetelmät
2) Ohjatun oppimisen (ennustamis) vs. ohjaamattoman oppimisen (ryvästely ja projektio) menetelmät

2.1 Parametriset mallit

Kaikkein yleisimmällä tasolla tilastollinen mallinnus tarkoittaa jakaumien estimointia. Näin siksi, että tutkittavan ilmiön käyttäytymisen selittävien muuttujien yhteisjakauma sisältää kaiken saatavissa olevan tiedon kohteesta. Toisin sanoen jos ilmiön taustalla ovat muuttujat $X_1, X_2, \ldots, X_p$ niin paras mahdollinen malli on yhteisjakauma (esim. tiheys) $f_X(\mathbf{x}) = f_{X_1,X_2,\ldots,X_p}(x_1, x_2, \ldots, x_p)$. Käytännössä yhteisjakauman vaativuus havaintojen määrän sekä toteutusteknisten ongelmien vuoksi yleensä estää täydellisen mallin muodostamisen. Tämän vuoksi tehtävä jaetaan osatehtäviin sopivilla taustaoletuksilla. Parametrisissa malleissa tämä taustaoletus on, että jakauman funktiomuoto tunnetaan. Esimerkiksi parametrisoiduissa jakaumissa voidaan olettaa, että jakauma on normaalijakauman muotoa $f_X(\mathbf{x}) = N(\overline{\mathbf{x}},\mathbf{C})$, missä muodon määräävät parametrit $\overline{\mathbf{x}}$ (keskiarvovektori) ja $\mathbf{C}$ (kovarianssimatriisi). Toinen mahdollisuus on olettaa, että satunnaismuuttujien välinen relaatio tunnetaan. Esimerkiksi regressiomalleissa oletetaan, että ennustettava(t) muuttuja(t) Y ovat jonkin satunnaisen funktion $g(x, \theta)$ odotusarvoja $g(x, \theta) = E[Y|x]$, missä x on taustatietoa edustava satunnaismuuttujan arvo ja θ on mallin parametrien joukko. Ghahramanin ja Jordanin mukaan parametristen mallien mahdollinen haittapuoli on, että ne eivät ole niin joustavia kuin epäparametriset mallit [12].

2.1.1 Parametrisoidut jakaumat

Yksinkertaisissa tehtävissä käytettäviä parametrisoituja jakaumia ovat mm. normaali-, binomi- ja Poisson-jakauma. Jakaumat ovat joko jatkuville tai diskreeteille muuttujille. Parametrisoitujen jakaumien etuina ovat helppo laskettavuus ja parametrien nopea estimointi datasta. Haittana on, että jos datan jakauma on monihuippuinen, niin sovittamalla siihen yksihuippuinen jakauma saadaan vääriä tuloksia. Esimerkkinä voidaan tarkastella normaalijakaumaa, jonka määräävät täysin sen parametrit. Yksiulotteisessa tapauksessa parametrit ovat jakauman oletusarvo ja varianssi (keskihajonnan neliö).

Odotusarvon määritelmä on

$$E(X) = \int_{-\infty}^{\infty} x f(x)\, dx,$$

mitä estimoidaan havaintojoukosta empiirisen keskiarvon μ avulla.

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x(i),$$

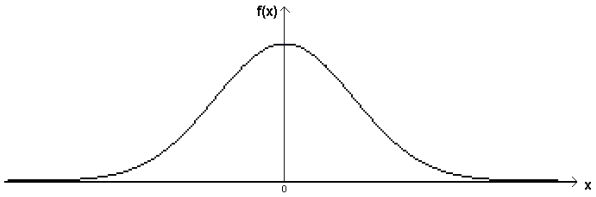
missä n on havaintojen lukumäärä. Vastaavasti muuttujan X varianssi voidaan estimoida havainnoista kaavalla

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x(i) - \overline{x})^2.$$

Jakauman muoto on tällöin

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp[-\frac{1}{2\sigma^2}(x-\mu)^2],$$

mitä on myös havainnollistettu kuvassa 11.



Kuva. 11: Yksiulotteinen normaalijakauma ($\mu = 0$ ja $\sigma = 1$)

Useampiulotteisessa tapauksessa normaalijakauman tiheysfunktio yleistyy muotoon

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\mathbf{C}|^{1/2}} \exp[-\frac{1}{2}(\mu - \mathbf{x})^{\mathrm{T}} \mathbf{C}^{-1} (\mu - \mathbf{x})],$$

missä d on dimensio, $\mathbf{C}$ on $d \times d$ kovarianssimatriisi, $|\mathbf{C}|$ on kovarianssimatriisin determinantti ja $\mathbf{C}^{-1}$ on kovarianssimatriisin käänteismatriisi.

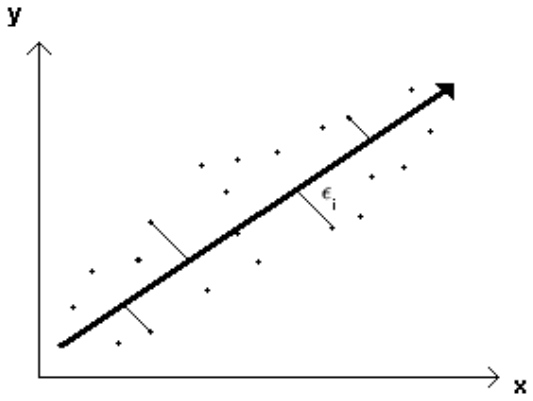
2.2 Regressiomenetelmät ja ohjattu oppiminen

Regressiomenetelmissä pyritään löytämään odotusarvoinen ennustusmalli, joka on muotoa $g(\mathbf{x}) = E[Y|\mathbf{x}]$. Kyseessä on siten satunnaisfunktiosta $f_{Y|\mathbf{x}}(\mathbf{y}|\mathbf{x})$ laskettu tunnuslu-

ku, joka tietyin oletuksin on helpompi määrätä, kuin itse jakauma. Esimerkiksi lineaarisessa regressiossa etsitään odotusarvoinen hypertaso, joka on muotoa

$$g(\mathbf{x}, \theta) = E[Y|\mathbf{x}] = \sum_{i=1}^{d} \theta_i x_i + \theta_0 = \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d + \theta_0.$$

Tätä on havainnollistettu kuvassa 12. Sovitus onnistuu, jos todellinen funktion jakauma on satunnainen siten, että $\mathbf{y} = g(\mathbf{x}, \theta) + \epsilon$, missä ϵ on nollakeskiarvoinen satunnaistekijä. Epälineaariset regressiomenetelmät ovat usein tarpeen kun mallinnetaan reaalimaailman ilmiöitä, jotka ovat harvoin lineaarisia. Epälineaarisessa regressiossa funktio $g(\mathbf{x}, \theta)$ on epälineaarinen. Yleisen teorian näkökulmasta tämä ei muuta tilannetta, mutta regressiotehtävän ratkaisu tulee vaikeammaksi, joissain tapauksissa jopa huonosti määritellyksi tehtäväksi. Perinteisissä tilastollisissa malleissa käytetäänkin yleensä varsin konservatiivisia valintoja regressiofunktion $g(\mathbf{x}, \theta)$ muodolle.



Kuva. 12: Lineaarinen regressio, jossa etsitään suora s.e. $\sum \epsilon_i = 0$

Eräs hyödylliseksi osoittautunut epälineaarinen regressiofunktio on nk. logistinen regressio, joka on muotoa $g(\mathbf{x}, \theta) = h(\sum_{i=1}^{d} \theta_i x_i + \theta_0)$, missä $h(a)$ on logistinen aktivaatiofunktio $h(a) = \frac{1}{1+\exp^{-a}}$ (kuva 13). Tässä muuttuja Y voi saada arvoja ainoastaan välillä $[0 \ldots 1]$, jolloin logistinen regressio soveltuu luontevasti mm. luokkatodennäköisyyksien esittämiseen. Esimerkiksi kirjoittamalla

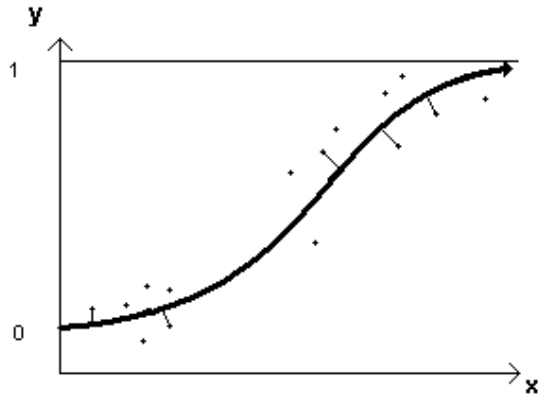
$$a = \ln \frac{f_{\mathrm{X}}(\mathrm{x}|y=1)\Pr(y=1)}{f_{\mathrm{X}}(\mathrm{x}|y=0)\Pr(y=0)}$$

huomataan, että logistinen regressio pystyy määräämään luokittelijan posterioritodennäköisyyden, sillä Bayesin kaavan mukaan

$$\Pr(y=1|\mathrm{x}) = \frac{f_{\mathrm{X}}(\mathrm{x}|y=1)\Pr(y=1)}{f_{\mathrm{X}}(\mathrm{x}|y=1)\Pr(y=1) + f_{\mathrm{X}}(\mathrm{x}|y=0)\Pr(y=0)} =$$

$$\frac{1}{1+\frac{f_{\mathrm{X}}(\mathrm{x}|y=1)\Pr(y=1)}{f_{\mathrm{X}}(\mathrm{x}|y=0)Pr(y=0)}} = g(a).$$

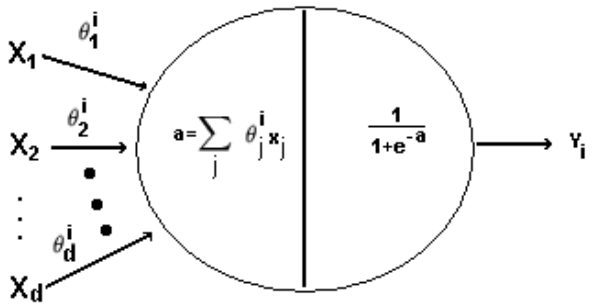
Lähde [18] sisältää taustatietoa logistisesta regressiosta.



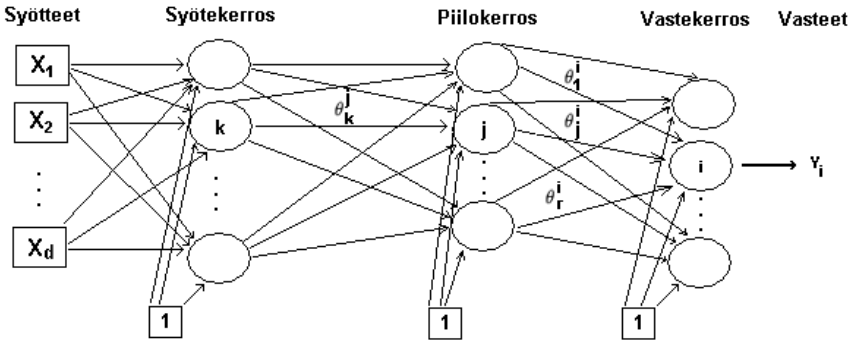
Kuva. 13: Logistinen regressio

2.3 Neuroverkot ja regressio

Monet tässä työssä käytettävät mallinnusmenetelmät kuuluvat nk. neuroverkkomenetelmien luokkaan. Nämä ovat epälineaarisia ja varsin joustavia menetelmiä, joille kuitenkin löytyy tehokkaita ratkaisumenetelmiä. Ohjatun oppimisen regressiomenetelmien puolella menetelmistä tunnetuin lienee nk. monikerroksinen perceptronverkko MLP, joka voidaan nähdä logistisen regression yleistyksenä. MLP-verkko koostuu kuvan 14a esittämistä yksiköistä, neuroneista, jotka ovat kytketty kuvan 14b esittämällä tavalla verkoksi.



Kuva. 14a: Neuroni, eli logistinen yksikkö



Kuva. 14b: Neuroverkko

Jokainen yksikkö, neuroni, on logistinen aktivaatiofunktio, joka saa syötteensä edellisen kerroksen neuronien vasteista. Syötekerros saa syötteensä koko verkon syötteistä. Koko verkon regressiofunktio on siten vahvasti epälineaarinen. Esimerkiksi yhden verkon vasteen taustalla oleva regressiofunktio on muotoa

$$Y_i = h(\sum_j \theta_j^i h(\sum_k \theta_k^j h(\ldots h(\sum_l \theta_l^r x_l)\ldots))).$$

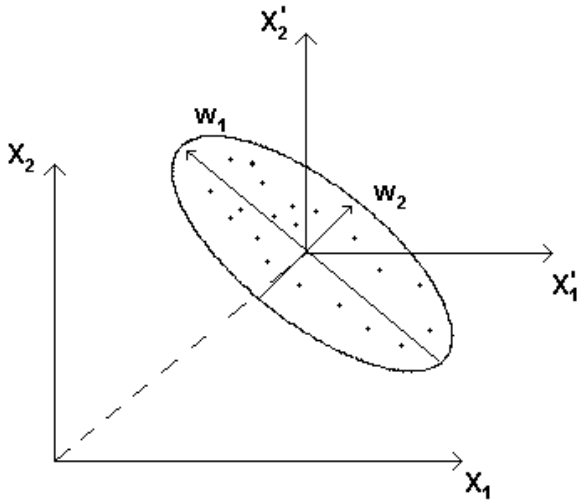
Mallinnuksen näkökulmasta MLP-neuroverkolla on se olennainen ero perinteisiin tilastollisiin menetelmiin, että verkolla on periaatteessa mahdollista muodostaa mikä tahansa kuvaus lähtö- ja maalijoukon välille. Neuroverkkokirjallisuudessa tätä ominaisuutta kutsutaan universaalisuudeksi. Toisaalta mallin joustavuus tekee sen parametrien estimointitehtävästä erityisen ongelmallisen. Tässä yhteydessä MLP-verkon ominaisuuksien tarkempi tarkastelu sivuutetaan ja asiasta kiinnostunutta lukijaa kehotetaan tutustumaan mm. lähteeseen [6]. Neurolaskentaan voi lukija perehtyä tarkemmin tutkimalla lähdettä [20].

2.4 Ohjaamattoman oppimisen menetelmät

Ohjaamattoman oppimisen menetelmissä ei pyritä muodostamaan ennustusmallia, eli funktiokuvausta $y = g(\mathrm{x})$, vaan tavoitteena on rakentaa epäsuora, latentti, malli, joka tiivistää havainnoissa $\mathbf{x}(j)$ olevan informaation. Siis ennusteosaa Y ei tiedetä tai hyödynnetä. Karkeasti jaoteltuna ohjaamattoman oppimisen menetelmät voidaan jakaa projektiomenetelmiin ja ryvästelymenetelmiin [31], mutta välimuotojakin löytyy.

2.4.1 Pääkomponenttianalyysi

Projektiomenetelmistä tunnetuimpia lienee pääkomponenttianalyysi. Pääkomponenttianalyysissa (engl. *Principle Component Analysis*) etsitään komponentit, jotka selittävät havainnot pienemmällä dimensiolla kuin mitä datan alkuperäinen dimensio oli. Jos oletetaan että alkuperäinen data koostuu vektoreista $\mathbf{x}(j) = (x_1(j), x_2(j), \ldots, x_n(j))$ niin pääkomponenttianalyysissa etsitään d-ulotteiset ortogonaaliset vektorit $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_l$, jossa $l \leq d$. Dimension pudottaminen tapahtuu siten että oletetaan että alkuperäiset vektorit ovat muotoa $\mathbf{x}(j) = s_{j1}\mathbf{w}_1 + s_{j2}\mathbf{w}_2 + \ldots + s_{jl}\mathbf{w}_l$, jossa $s_{j1}, s_{j2}, \ldots, s_{jl}$ ovat havaintovektorin uudet, pienempidimensioiset komponentit. Oletuksena on siis että alkuperäinen havaintoaineisto voidaan esittää pääkomponenttien lineaarikombinaationa. Pääkomponenttianalyysi on käytännöllinen mallinnettaessa dataa sillä sen avulla saadaan lineaarisessa mielessä selville mikä osa informaatiosta on olennaista. Sen avulla voidaan määrätä tiheysfunktion pääsuunnat. Menetelmällä saadaan myös selville pääkomponenttien selitysasteet, olettaen että havaintoavaruus on normeerattu. Selitysaste on arvo väliltä [0,1] ja se kertoo kuinka monta prosenttia pääkomponentti selittää datajoukosta. Selitysastetta voidaan käyttää mm. kohinan poistamiseen datasta, joka tapahtuu usein poistamalla vähiten merkitsevä pääkomponentti. Pääkomponenttianalyysin voi tulkita myös moniulotteisen normaalijakauman sovittamiseksi havaintoaineistoon. Tällöin pääkomponenttiakselit, eli vektorit $\mathbf{w}_l, l \leq d$ ovat tulkittavissa normaalijakauman muodon määräämän varianssiellipsoidin pääakseleiksi (kuva 15). Suurimman pääkomponentin suunta siis maksimoi havaintoaineiston varianssin.



Kuva. 15: Pääkomponenttianalyysi

2.4.2 Ryvästelymenetelmät

Ryvästelyllä, eli klusteroinnilla pyritään jakamaan havaintoaineisto mielekkäisiin, samanlaista käyttäytymistä edustaviin ryhmiin. Staattisissa ryvästelymenetelmissä ryppäiden lukumäärä oletetaan ennalta tunnetuksi. Dynaamisissa menetelmissä ryppäitä luodaan tarpeen mukaan, esim. jakamalla suurempia ryppäitä osiin. Ryvästelymenetelmien etuna on se että ne ovat yksinkertaisia toteuttaa ja ne usein säilyttävät datan monimutkaisen jakauman (olettaen että ryppäiden määrä ei ole rajattu). Ryvästelymenetelmistä tunnetuimpia ovat nk. K-means klusterointi ja erityyppiset hierarkiset klusterointimenetelmät. K-means klusteroinnissa datajoukko jaetaan osiin siten, että jokainen havainto kuuluu vain yhteen ryppääseen (kuva 16). Valittu rypäs on K-means algoritmissa se, jonka Euklidinen etäisyys rypäskeskuksesta on lyhyin. Vastaavasti rypäskeskukset sijoitetaan niihin kuuluvien havaintojoukkojen keskipisteisiin. Tämä johtaa seuraavan kaltaiseen iteratiiviseen algoritmiin.

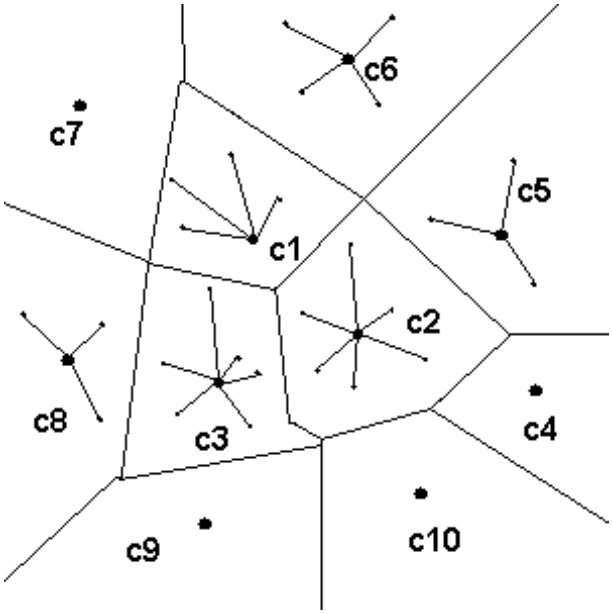
Algoritmi 2 (K-means klusterointi)

1. *Alustetaan klustereiden keskipisteet* $\overline{\mathbf{x}}^{(0)}(i), i = 1 \ldots k$ *satunnaisesti datajoukosta ja asetetaan iteraatioaskelta kuvaava indeksi* t *nollaksi*

2. *Tehdään Voronoi-jako eli etsitään jokaiselle ryppäälle i ne havainnot* $\mathbf{x}(l), l = 1 \ldots n$ *joille pätee* $||\overline{\mathbf{x}}(i) - \mathbf{x}(l)||^2 < ||\overline{\mathbf{x}}(j) - \mathbf{x}(l)||^2, \forall j \neq i$

3. *Lasketaan ryppäiden keskipisteet* $\overline{\mathbf{x}}^{(t+1)}(i) = \frac{1}{\#V_i} \sum_{l \in V_i} \mathbf{x}(l)$

4. *Jos keskipisteiden muutos on suuri eli* $|\overline{\mathbf{x}}^{(t+1)}(i) - \overline{\mathbf{x}}^{(t)}(i)| > \delta$ *yhdelläkin ryppäällä i niin kasvatetaan t:tä yhdellä ja palataan takaisin kohtaan 2, missä* δ *on positiivinen pieni luku lähellä nollaa.*

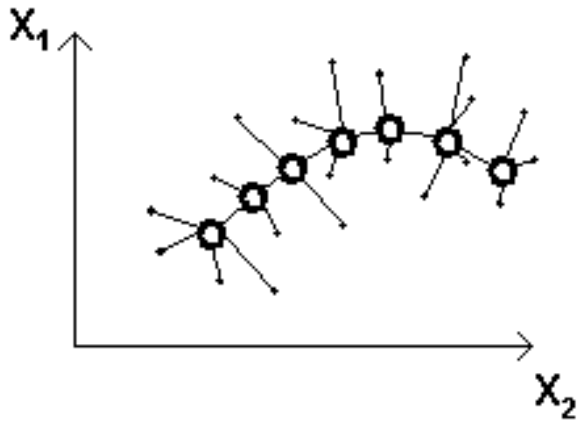


Kuva. 16: K-means rypästelty data, c1-c10=rypäskeskukset

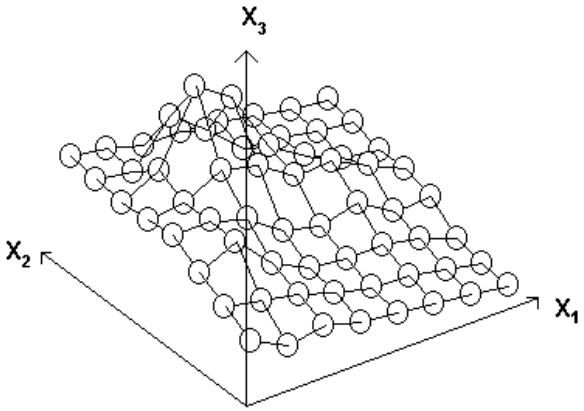
2.4.3 Itseorganisoituva piirrekartta (SOM)

Itseorganisoituva piirrekartta SOM on akateemikko Teuvo Kohosen vuonna 1982 keksimä menetelmä, jota voi luonnehtia projektiomenetelmien ja ryvästelymenetelmien välimuodoksi. Alkuperäinen vuonna 1982 [*Kohonen*] esitetty algoritmi perustuu stokastiseen opetusalgoritmiin, mutta nykyään käytetään useimmiten K-means algoritmin tyyppistä Batch-opetusta. Itseorganisoituvan piirrekartan ideana on sovittaa havaintoaineistoon toisiinsa kytketyistä ryppäistä muodostuva hila, jonka voi ajatella edustavan diskretisoitua pintaa (tai käyrää yksiulotteisessa tapauksessa). Projisoimalla useampiulotteinen havaintoaineisto SOM-hilan muodostamalle pinnalle, voidaan moni-

muuttujaista havaintoaineistoa tarkastella alkuperäistä yksinkertaisemmassa representaatiossa. Itseorganisoituvan piirrekartan ideaa on havainnollistettu kuvissa 17 ja 18. Kartta koostuu ryppäistä (kuten K-means algoritmi), mutta itseorganisoituvan piirrekartan tapauksessa ryppäät kytkeytyvät toisiinsa, muodostaen jatkumon.



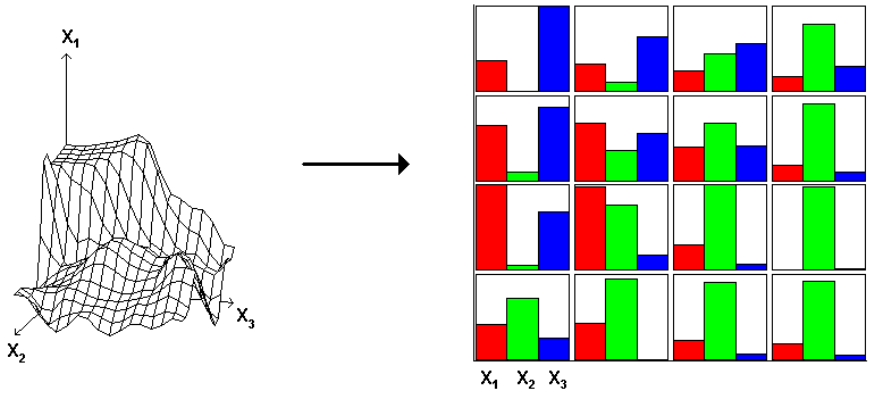
Kuva. 17: Yksiulotteinen SOM sovitettuna 2-D havaintoaineistoon



Kuva. 18: 2-D SOM sovitettuna 3-D havaintoaineistoon

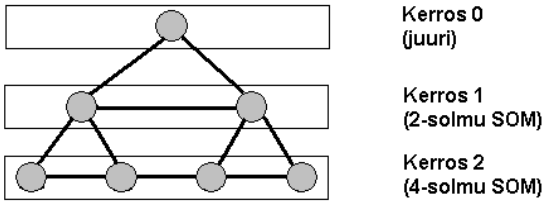
Itseorganisoituvan kartan opetusalgoritmissa ryppäiden muodostama verkkotopologia on huomioitu siten, että siirrettäessä yhtä rypästä, niin myös sen lähinaapurustoa siirretään samaan suuntaan. Tarkempia tietoja itseorganisoituvasta kartasta ja sen taustalla olevasta teoriasta löytyy mm. lähteistä [19] ja [26]. Lyhyenä johdatuksena aiheeseen voi tutustua myös lähteisiin [34] ja [13].

Visuaalisuutensa vuoksi itseorganisoituva karttaa sovelletaan usein data-analyysiin. Toimintatapa on tällöin rypästellä moniulotteinen havaintoaineisto SOM-pinnalle ja tarkastella tulosta ryppäisiin laskettujen tunnuslukujen avulla. Tätä on havainnollistettu kuvassa 19.



Kuva. 19: Datan visualisointi SOM:lla

Eräs tunnettu itseorganisoituvan kartan muunnelma on TkT Pasi Koikkalaisen kehittämä puurakenteinen itseorganisoituva piirrekartta TS-SOM. TS-SOM (engl. *Tree-Structured Self-Organizing Map*) rakentuu useasta SOM kerroksesta (engl. *SOM layer*), jotka on kytketty toisiinsa puurakenteen avulla (kuva 20).



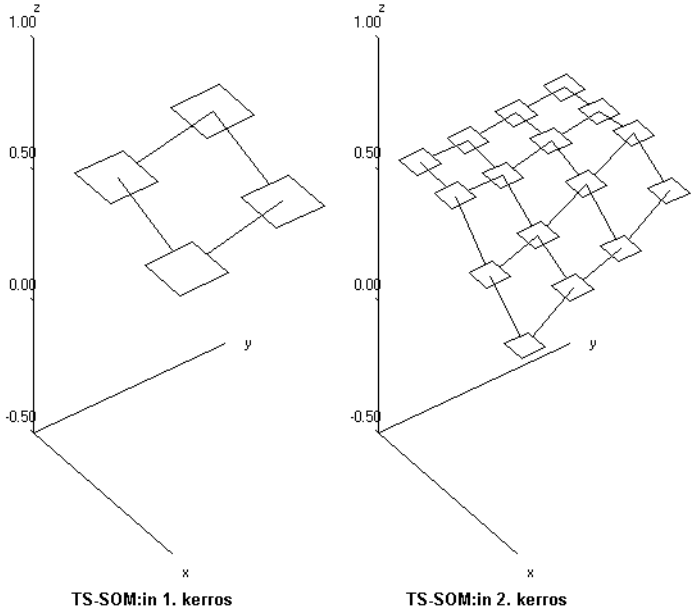
Kuva. 20: 1-d TS-SOM

Puurakenteella saavutetaan perinteiseen SOM algoritmiin nähden useita etuja:

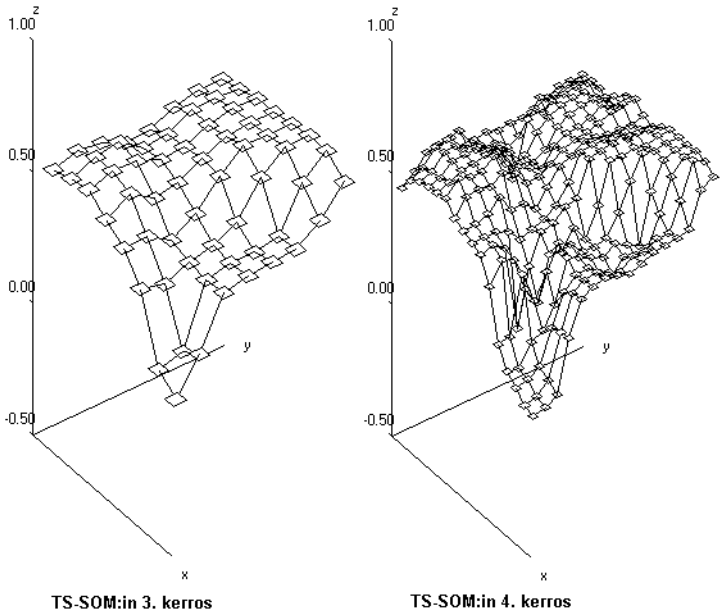
- Algoritmin opetusaika voidaan tehdä huomattavan nopeaksi hyödyntämällä puuhakumenetelmiä
- Algoritmi tuottaa moniresoluutioisen esitystavan, joka koostuu useasta SOM kartasta. Näitä voidaan monipuolisesti hyödyntää etsittäessä sopivaa kartan kompleksisuutta dataan nähden

- Algoritmin avulla voidaan eliminoida perinteisessä SOM algoritmissa tarvittavia vaikeasti määrättäviä opetusparametreja.

Algoritmin sovitusta erääseen moniulotteiseen havaintoaineistoon on havainnollistettu oheisissa kuvissa 21 ja 22. Mitä korkeampaa SOM kerrosta käytetään sitä tarkemmin datan jakauma mallintuu.



Kuva. 21: TS-SOM kerrokset 1 ja 2



Kuva. 22: TS-SOM kerrokset 3 ja 4

2.5 Validointi

Havaintoaineistosta muodostettu malli on validoitava, jotta voidaan tietää kuinka hyvin siihen voidaan luottaa. Validoinnilla selvitetään mallin hyvyys. Validointimenetelmillä saadaan selville mallin mahdollisia puutteita. Havaintoaineistosta laskettavien mallin parametrien luotettavuutta voidaan estimoida validoinnilla. Tätä varten lasketaan $f(\theta|D)$, missä θ on mallin parametrit ja D on havaintoaineisto. Validoinnilla saadaan myös mitattua korjatusta ja/tai täydennetystä havaintoaineistosta laskettavien tunnuslukujen vaihteluvälit. Validointi voidaan tehdä mm. Bootstrap-menetelmällä.

Luvussa 5 esitellyssä editointi- ja imputointiohjelmistossa koko tiedon tuotantoprosessi validoidaan. Ohjelmisto tuottaa virheellisestä ja/tai puutteellisesta havaintojoukosta virheistä korjatun ja täydennetyn havaintojoukon. Ohjelman aikana käytetään useita malleja, kuten editointimallia (kts. luku 3), havaintoaineiston jakauman mallia ja imputointimalleja (kts. luku 4). Nämä kaikki mallit voivat tuottaa vaihtelua tuotettuun havaintojoukkoon. Editointimallina eli virheiden korjausmallina on sääntöpohjainen editointijärjestelmä, jossa käytetään satunnaista otantaa, jos korjausarvoja on useita. Havaintoaineiston jakaumasta tehty malli voi vaihdella. Esimerkiksi käytettäessä

TS-SOM-algoritmia samaan havaintoaineistoon useita kertoja neuronien painovektorit voivat vaihdella. Jakauman mallin vaihtelu voi aiheuttaa vaihtelua imputointimalleihin, koska niiden käyttämä havaintoaineiston osajoukko saattaa muuttua. Imputointimallit voivat myös sisältää täydennettyihin arvoihin vaihtelua tuottavia tekijöitä. Tästä esimerkkinä on normaalijakauman avulla tehty puuttuvien arvojen täydentäminen, missä havaintojoukkoon sovitetaan normaalijakauma, josta poimitaan estimaatit puuttuville arvoille.

2.5.1 Bootstrap

Bootstrap-menetelmässä ideana on että havaintojoukosta D, jossa on n havaintoa, muodostetaan N uutta havaintojoukkoa satunnaisella otannalla. Muodostetuissa toisistaan riippumattomissa havaintojoukoissa $D_1, D_2, \ldots, D_N$ on myös n havaintoa. Tämän jälkeen lasketaan tunnusluvut, jotka ovat esimerkiksi keskiarvot

$$\mu_l = \frac{1}{n} \sum_{x(i) \in D_l} x(i), \tag{2.1}$$

missä $l = 1 \ldots N$ ja $n = \#\{D_l\}$.

Seuraavaksi lasketaan tunnuslukujen keskiarvo, joka on esimerkissämme keskiarvojen keskiarvo

$$\mu = \frac{1}{N} \sum_{l=1}^{N} \mu_l \tag{2.2}$$

Lopulta määritellään virhevarianssin estimaatti

$$s^2 = \frac{1}{N-1} \sum_{l=1}^{N} (\mu_l - \mu)^2 \tag{2.3}$$

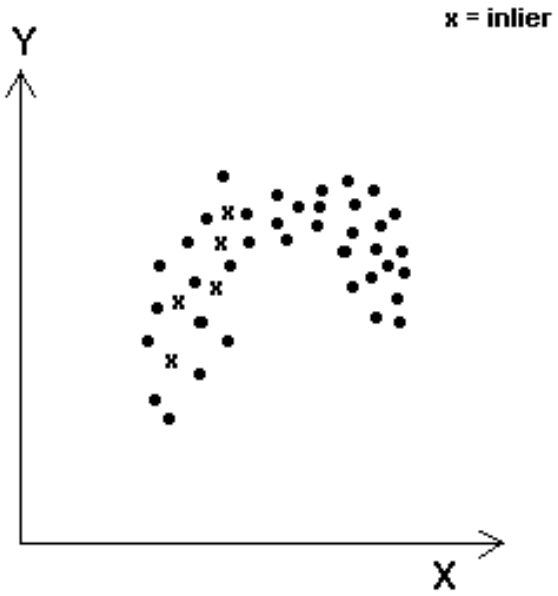
Näin saadaan selville luottamusväli havaintojoukkojen tunnusluvuille (tässä keskiarvolle). Lasketuilla tunnusluvuilla voidaan arvioida mallin luotettavuutta.

Luku 3

Virheiden havaitseminen ja korjaaminen

Tämä luku on lyhyt johdatus yleisempiin menetelmiin, joita käytetään havaintoaineistossa olevien virheiden löytämiseen ja korjaamiseen. Kuten luvussa 1 tuotiin esille, voi virheiden taustalla olla hyvinkin monenlaisia syitä, mistä myös seuraa, ettei virheiden tunnistamiseen ja korjaamiseen ole olemassa mitään täysin yleispätevää menetelmää.

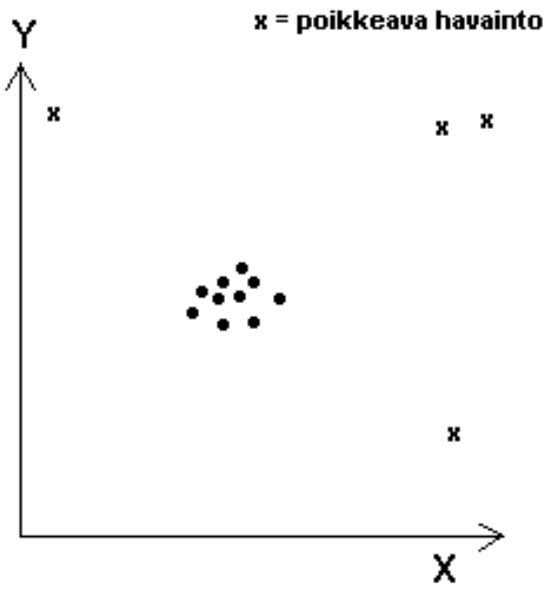
Virheiden tunnistaminen voidaan karkeasti jakaa syntaktisiin (sääntöpohjaisiin) ja tilastollisiin menetelmiin. Tilastollisen tarkastellun näkökulmasta virhe voi olla tilastollisesti mahdollinen, inlayer, joka nimensä mukaisesti esiintyy samalla arvoalueella kuin virheettömät havainnot. Inlier tyyppiset havainnot ovat siis ikäänkuin datan jakauman sisällä (kuva 23), jolloin niitä on yleensä erittäin vaikea havaita yksikkötasolla. Ero virheellisen ja virheettömän havaintoaineiston välillä näkyykin ainoastaan jakauman muuttumisena.



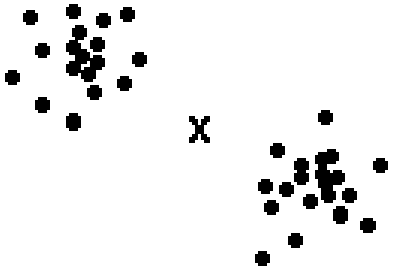
Kuva. 23: Inlier havaintoja

Toinen, yleensä helpompana pidetty vaihtoehto on, että kyseessä on selvästi poikkeava käyttäytyminen (engl. *outlier*), jonka voi tulkita olevan datan luonnollisen jakauman ulkopuolella (kuva 24). Käytännön moniulotteisissa havaintoaineistoissa outliertyyppisten havaintojen löytäminen on myös erittäin hankalaa, sillä havaintojakauman rajojen määrittäminen on vaativa tehtävä. Rajojen määrittämistä on tutkittu artikkelissa [7].

Useat virheentunnistuksen menetelmät perustuvat vahvoille oletuksille havaintoaineiston rakenteesta. Tämän vuoksi ennen virheiden korjaamista on syytä tunnistaa sen luonne (mm. datan jakauman huippujen määrä, kuva 25). Tämä helpottaa myös virheiden tunnistuksen jälkeen tehtävää havaintojoukon korjaamista. Sillä voidaan myös usein välttää uusien virheiden synnyttämistä. Esimerkiksi on tärkeää tunnistaa onko kyseessä 2-huippuinen jakauma kuten kuvassa 25. Olettamalla jakauma yksihuippuiseksi vieraat havainnot tunnistetaan vasta molempien huippujen reunojen ulkopuolella. Tällöin ei havaita huippujen välissä olevia vieraita havaintoja, minkä vuoksi datan korjaaminen voi epäonnistua.



Kuva. 24: Poikkeavia havaintoja



Kuva. 25: 2-huippuinen jakauma

Tilastolliseen otantaan perustuvassa havaintoaineiston hankinnassa outlier-tyyppisten virheiden lähteitä ovat [10]

1. tallennus- ja mittausvirheet,
2. virheellinen oletus datan jakaumasta,
3. tuntematon datan rakenne ja
4. tutkittavan ilmiön ainutlaatuisuus.

Tallennus- ja mittausvirheet voivat aiheuttaa havaintoihin virheitä, jotka näkyvät havaintojoukossa havaintopoikkeamina. Toisaalta virheellinen oletus datan jakaumasta aiheuttaa dataan helposti havaintopoikkeamilta "näyttäviä" havaintoja. Tuntematon datan rakenne voi aiheuttaa helposti havaintopoikkeamien havaitsemista. Esimerkiksi oletetaan että data on oikeasti jakautunut kolmeen toisistaan riippumattomaan alijoukkoon. Jos data käsitellään yhtenä joukkona niin tunnistetaan havaintopoikkeamia. Tutkittavan ilmiön ainutlaatuisuutta havaintopoikkeamien lähteenä kuvaa hyvin se kun mittauksia, jotka indikoivat että otsonikerroksessa on reikä, pidettiin havaintopoikkeamina ja ne hylättiin automaattisesti. Tämä huomaamattomuus viivästytti ilmiön havaitsemista useilla vuosilla [5].

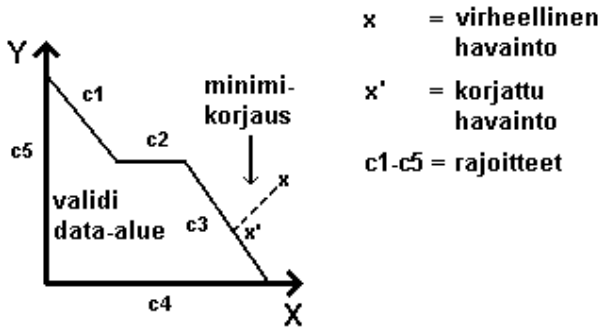
Virheiden havaitsemiseen on useita menetelmiä. Eräs, mutta ei ainoa mahdollinen, luokittelu on jakaa menetelmät

1) Syntaktisiin eli sääntöpohjaisiin menetelmiin, joissa käytetään sääntöjä virheiden havaitsemiseen.
2) Mallipohjaisiin menetelmiin, joissa sekä ehjälle että virheelliselle datan osalle on olemassa malli.
3) Robusteihin menetelmiin, joissa pyritään mallintamaan ainoastaan havaintoaineiston ehjää osaa, hyödyntäen estimointimenetelmiä, jotka ovat virhesietoisia eli robusteja.

Varsinainen korjaus eli editointi jakautuu kahteen pääkategoriaan, jotka ovat yhden- ja usean vaiheen editointi. Yhden vaiheen editoinnissa (engl. *single stage editing*) kaikkia muuttujia käsitellään samanaikaisesti. Tällöin dataa tarvitaan kaikista muuttujista yhtäaikaa, mikä on heikkous. Tämän ansiosta editointijärjestelmästä tulee todennäköisesti monimutkainen. Datan rakenne (yhdistelmäjakaumat ja muuttujien väliset riippuvuudet) säilyvät todennäköisemmin kuin usean vaiheen editoinnissa. Usean vaiheen editoinnissa (engl. *multiple stage editing*) muuttujia käsitellään pieninä ryhminä tai yksittäin. Tällöin vahvuutena on editointijärjestelmän yksinkertaisuus verrattuna yhden vaiheen menetelmiin. Datan rakenne ei välttämättä säily kovin hyvin, mikä on heikkous.

3.1 Sääntöpohjainen virheiden tunnistus ja korjaus

Datan editointi (engl. *editing*) on virheellisen datan korjaamista ja mahdollisesti poistamista. Virheet korjataan yleensä siten että havainto poistetaan ja sen tilalle imputoidaan eli täydennetään estimoitu arvo. Esimerkiksi LP-malleissa minimikorjaus voidaan tehdä lineaaristen rajoitteiden avulla (kuva 26).



Kuva. 26: Minimikorjaus LP-mallin lineaarisilla rajoitteilla

Datan korjaaminen voidaan tehdä myös eksplisiittisten sääntöjen avulla. Esimerkiksi jos oletetaan että on olemassa sääntö $X = Y + Z$ ja on havaittu arvot x=100, y=100 ja z=-50000 ja tiedetään z:n olevan virheellinen havainto niin se voidaan korjata. Z:n arvo ratkaistaan yhtälöstä $X - Y = Z$ (esimerkissä z=100-100=0). Oletetaan että havainto on esimerkiksi vektori $\mathbf{x} = (0, 0, 2, 2, 0, 0)$, jossa 2:t merkitsevät virheellistä tietoa. Sallittujen korjauksien määrälle voidaan määritellä kynnysarvo l. Jos havaintovektorissa on enemmän kuin l virheellistä arvoa, niin se hylätään kokonaan. Olkoon kynnysarvo l esimerkissämme 1. Esimerkin vektori $\mathbf{x}$ hylättäisiin, koska siinä on kaksi virheellistä arvoa.

3.2 Robustit menetelmät

Normaalit estimaattorit, kuten keskiarvo ovat erittäin herkkiä virheille (vieraat havainnot). Robustit estimaattorit (engl. *robust estimators*) taas sietävät suhteellisen hyvin virheitä. Laskettaessa keskiarvoa yksikin havaintopoikkeama saattaa aiheuttaa jo huo-

mattavan virheen. Keskiarvon laskemisen sijaan voidaan laskea Turkey'in 3-keskiarvo (engl. *Turkey's three-mean*), joka on

$$C_t = \frac{\mathcal{Q}_1}{4} + \frac{\mathcal{Q}_2}{2} + \frac{\mathcal{Q}_3}{4}. \tag{3.1}$$

Datajoukon "leveyden" määrittelemiseen käytetään tällöin usein kvartiilipoikkeamaa Q_d (kts. 3.6) koska se on robusti. Kyseessä on siis tapa muodostaa virhesietoinen malli, josta poikkeavat havainnot on (toivottavasti) helppo tunnistaa.

Triviaali tapa robustin mallin tekemiseen on datan karsiminen. Datan karsimisessa (engl. *data trimming*) datasta jätetään virheelliset havainnot pois. Poisjätettävät havainnot ovat datan "alku-" ja "loppupäässä". Esimerkiksi jos havaintojoukko on {100, 1000, 1003, 1007, 1005, 50000}, niin karsimalla se muuttuu joukoksi {1000, 1003, 1007, 1005}. Menetelmästä voidaan kehittää edelleen nk. ikkunointimenetelmä. Häiriöiden poistaminen (engl. *winsorisation*) on menetelmä, jossa järjestellään havaintoaineisto kasvavaan järjestykseen. Ääripäille annetaan viereisten arvot. Menetelmää voidaan joutua iteroimaan, jotta datasta saadaan "häiriötön". Sitä käytetään usein symmetrisesti, jolloin datan alku- ja loppupää käsitellään vastaavasti. Esimerkiksi jos datajoukko on {1, 5, 6, 7, 8, 20}, niin ääripäille annetaan viereisten arvot. Näin alkuperäisestä datasta saadaan joukko {5, 5, 6, 7, 8, 8}.

Esityksen tiivistämiseksi tarkastelemme vain seuraavaa esimerkkiä. Eräs karkea mutta usein käytetty robusti menetelmä virheellisten havaintojen tunnistamiseen on kvartiilivälin käyttäminen. Se ei ole kuitenkaan yleispätevä ja saattaa johtaa validin datan hylkäämiseen. Kvartiilit ovat nimensä mukaisesti jakauman massan neljänneksiä, jotka voidaan määritellä mm. tiheysfunktion $f_X(x)$:n avulla seuraavasti
alakvartiili $\mathcal{Q}_1$:

$$\int_{-\infty}^{\mathcal{Q}_1} f(x)\,dx \ = 0.25 \tag{3.2}$$

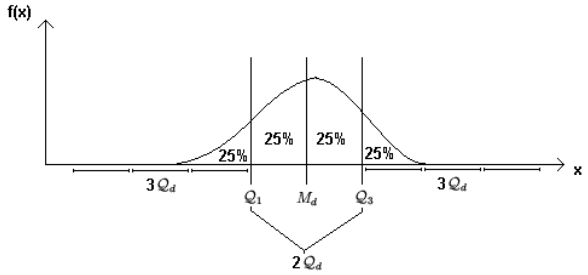
mediaani $\mathcal{Q}_2 = M_d$:

$$\int_{-\infty}^{M_d} f(x)\,dx \ = 0.5. \tag{3.3}$$

yläkvartiili $\mathcal{Q}_3$:

$$\int_{-\infty}^{\mathcal{Q}_3} f(x)\,dx \ = \int_{-\infty}^{\mathcal{Q}_1} f(x)\,dx + \int_{\mathcal{Q}_1}^{M_d} f(x)\,dx + \int_{M_d}^{\mathcal{Q}_3} f(x)\,dx \ = 0.25+0.25+0.25 = 0.75. \tag{3.4}$$

Tätä on havainnollistettu kuvassa 27.



Kuva. 27: Kvartiilit

Menetelmä toimii seuraavasti: oletetaan, että $f_X(x)$ on tiheysfunktio jonka kvartiiliväli on

$$\mathcal{Q}I = (\mathcal{Q}_1, \mathcal{Q}_3) \tag{3.5}$$

ja kvartiilipoikkeama on

$$\mathcal{Q}_d = \frac{(\mathcal{Q}_3 - \mathcal{Q}_1)}{2}. \tag{3.6}$$

Jos havainto x on havaintopoikkeama, niin on voimassa

$$\min(x - (\mathcal{Q}_1 - k\mathcal{Q}_d), (\mathcal{Q}_3 + k\mathcal{Q}_d) - x) < 0, \tag{3.7}$$

missä k:n arvona käytetään yleensä kolmosta.

On muistettava, että yhtälön 3.7 ollessa tosi, on käytettävä myös järkeä. Esimerkiksi 2-huippuisessa jakaumassa yhtälö antaa turhan helposti virheellisen vastauksen. Tämä voi tapahtua esimerkiksi kun yhtälöä hyödynnetään oletuksella että havaintoaineistossa on vain yksi huippu vaikka siinä oikeasti olisikin kolme.

Luku 4

Imputointi

Tässä luvussa esitetään yleisellä tasolla mitä puuttuvuusmekanismeja epätäydellisen havaintoaineiston taustalla voi olla, annetaan taksonomia erityyppisistä imputointimenetelmistä, sekä luodaan katsaus muutamiin yleisimpiin imputointimenetelmiin puutteellisen havaintoaineiston täydentämiseksi. Menetelmistä on keskitytty selittämään tarkemmin ne, joita käytetään luvussa 5 esitellyssä sovelluksessa.

Usein imputoitavana on vain yksi havaintojoukko, mutta joskus on saatavilla useita havaintojoukkoja samasta aiheesta, esimerkiksi samasta kyselystä eri ajankohtina kerättyjä vastauksia. Tällöin voidaan joko yhdistää eri havaintojoukkojen sisältämä informaatio tai hyödyntää toistoja menetelmien luotettavuuden testaamiseksi [11]. Kummassakin menetelmässä on ongelmia. Ensimmäisen lähestymistavan ongelmana on, että eri havaintojoukkojen, esimerkiksi kyselyiden, välisiä eroja ei osata käsitellä, minkä taustalla voi olla mm. ajankohta, otantamenetelmä tai otannantekijä. Jälkimmäisessä ongelmana on, että jos puuttuvuutta on paljon, niin imputointi voi olla vaikeaa.

4.1 Puuttuvuusmekanismit

Little ja Rubin määrittelevät kolme erilaista tapaa havaintoaineiston puuttuvuudelle (kts. [23], s. 14-15), jotka ovat

- täysin satunnainen puuttuvuus,
- satunnainen puuttuvuus ja
- ei-satunnainen puuttuvuus.

Olkoon havaintoaineisto kirjoitettu matriisimuotoon $\mathbf{D}$ siten, että matriisiin sarakkeet edustavat muuttujia $X_1, X_2, \ldots, X_r$ ja rivit edustavat havaintoja $j = 1, 2, \ldots, n$ eli $X_i(j) = d_{ji}$. Havaintoaineisto voidaan jakaa kahteen osaan $\mathbf{D} = \{D_{\text{observed}}, D_{\text{missing}}\}$, missä D_{observed} edustaa kaikkia havaittuja havaintojen arvoja d_{ji} ja D_{missing} taas puuttuvia. Puuttuva osa voidaan edelleen esittää käyttäen erillistä indikaattorimatriisia $\mathbf{M}$, joka voidaan määritellä siten, että alkio

$$m_{ji} = \begin{cases} 0, & \text{kun } d_{ji} \in D_{\text{observed}} \\ 1, & \text{kun } d_{ji} \in D_{\text{missing}} \end{cases}$$

Kun puuttuvuus on täysin satunnaista (engl. *missing completely at random* - MCAR) niin havaintoaineiston $\mathbf{D}$ alkioiden d_{ji} puuttuvuutta ei voida ennustaa havaintoaineistolla, joten on voimassa yhtälö

$$\Pr(\mathbf{M}|\mathbf{D}) = \Pr(\mathbf{M}). \tag{4.1}$$

Esimerkiksi tutkijat voivat pyytää satunnaisesti kolikkoa heittämällä osaa haastateltavista täyttämään 2/3 kyselystä [41]. Tällöin puuttuminen on täysin satunnaista.

Satunnaisessa puuttuvuudessa (engl. *missing at random* - MAR) epätäydelliset havainnot poikkeavat täydellisistä havainnoista, mutta puuttuvien muuttujien arvot ovat jäljitettävissä tai ennustettavissa muista muuttujista. Puuttuvuus on tällöin seurausta jostakin ulkopuolisesta tekijästä, eli se on satunnaista. Todennäköisyys havainnon puutteellisuudelle riippuu havaituista mutta ei havaitsemattomista muuttujista. Se voidaan kirjoittaa muotoon

$$\Pr(\mathbf{M}|\mathbf{D}) = \Pr(\mathbf{M}|D_{\text{observed}}). \tag{4.2}$$

Esimerkiksi tutkijat voivat kyselyn alussa järjestää lukutaitotestin [41]. On todennäköisempää että lukutaidoltaan huonommat henkilöt eivät saa täytettyä kyselyä kokonaan kuin lukutaidoltaan paremmat henkilöt. Puuttuvat havainnot ovat nyt seurausta ulkopuolisesta tekijästä eli lukutaidottomuudesta.

Ei-satunnaisessa puuttumisessa (engl. *nonignorable* - NI) puuttuvat muuttujat eivät ole ennustettavissa muista muuttujista. Tällöin myös puuttuvat havaintovektorin arvot riippuvat muista havaitsemattomista arvoista. Tässä tapauksessa $\Pr(\mathbf{M}|\mathbf{D})$ ei yksinkertaistu vaan

$$\Pr(\mathbf{M}|\mathbf{D}) = \Pr(\mathbf{M}|\{D_{\text{observed}}, D_{\text{missing}}\}) = \Pr(\mathbf{M}|\mathbf{D}). \tag{4.3}$$

Vastoin kuin MAR tapauksessa, jossa havaintoaineiston puutteellinen osa voitiin ennustaa muista muuttujista, ei-satunnaisessa tapauksessa puuttuvien havaintojen ennustamiseen tarvitaan myös tietoa niistä muuttujista, joissa on puuttuvuutta. Esimerkiksi suurituloiset ihmiset jättävät todennäköisemmin vastaamatta tuloja koskeviin kysymyksiin kuin muut, ja jos muut muuttujat eivät selitä ketkä ovat suurituloisia niin kyseessä on ei-satunnainen datan puuttuvuus [16].

Käytännössä havaintoaineisto täsmää harvoin MCAR-oletuksen kanssa, mutta geneerisempi MAR-oletus pitää usein paikkansa [41]. Toisaalta havaintojen puuttuvuus voidaan luokitella myös geometrisesti (engl. *geometric missingness*) tutkimalla havaintomatriisia $\mathbf{D}$, jolloin puuttuvuustyyppejä on kaksi, jotka ovat

- satunnainen puuttuvuus
- lohkopuuttuvuus.

Jos puuttuvuus havaintomatriisissa on satunnaista, kuten kuvassa 28, niin kyseessä on satunnainen puuttuvuus.

?	2	?	3
9	11	17	7
24	?	4	?
18	19	22	33

Kuva. 28: Satunnainen puuttuvuus

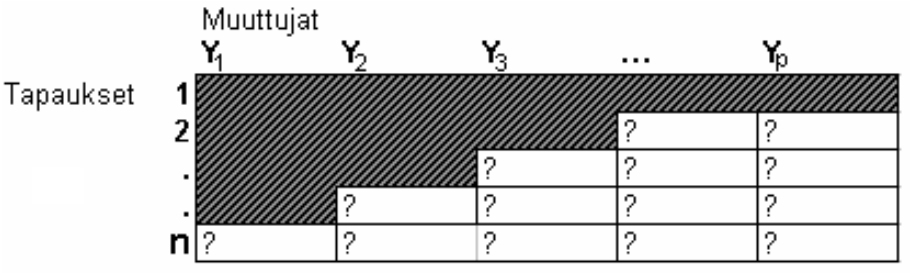
Jos puuttuvuus on ”säännöllistä” eli havaintomatriisissa on selviä puuttuvuusalueita, kuten kuvassa 29, niin kyseessä on lohkopuuttuminen (engl. *pattern missingness*).

1	2	8	3
9	11	17	7
24	6	?	?
18	19	?	?

Kuva. 29: Lohkopuuttuvuus

Monotoninen lohkopuuttuvuus

Monotonisella lohkopuuttuvuudella (engl. *monotone missingness pattern*) tarkoitetaan sitä että arvoa $Y_i(j)$ seuraavat arvot $Y_{i+1}(j), Y_{i+2}(j), \ldots, Y_p(j)$ ovat myös puutteellisia, edellyttäen että muuttujat on järjestelty puuttuvuusosuuksiensa suhteen pienimmästä suurimpaan (lähde [37], s. 218-219) kuten kuvassa 30.



Kuva. 30: Monotoninen lohkopuuttuvuus

Monotonisen lohkopuuttuvuuden avulla voidaan nk. IP-menetelmällä (kts. s. 41) tehtävää imputointia nopeuttaa jättämällä puuttuvasta havaintoaineistosta osa täyttämättä I- eli imputointivaiheessa (lähde [37], s. 218). I-vaiheessa simuloidaan puuttuvia arvoja edellisen havaintojoukon ja laskettujen mallin parametrien avulla. Nopeutuksen ideana on täydentää I-vaiheessa havaintoaineistoa vain sen verran että siihen muodostuu monotoninen puuttuvuuslohko. Tällöin imputointi nopeutuu koska puuttuvista arvoista täydennetään vain osaa. Schaferin mukaan tämä menetelmä stationarisoituu vähemmällä iteraatioiden määrällä verrattuna siihen, että simuloitaisiin kaikkia puuttuvia arvoja (lähde [37], s. 227).

4.2 Imputointimenetelmien taksonomia

Imputointimenetelmät voidaan jakaa kahteen pääkategoriaan, jotka ovat yksikkö- ja moni-imputointi. Varsinainen imputointi eli puuttuvien arvojen täydentäminen voidaan tehdä kummassakin pääkategoriassa käyttäen joko mallipohjaisia- tai luovuttajamenetelmiä. Yksikköimputoinnissa (engl. *single imputation*) jokainen puuttuva tieto korvataan täsmälleen yhdellä arvolla, jolloin prosessin jälkeen muodostuu yksi ehjä datajoukko. Yksikköimputoinnin etuja ovat [14]

- täydennettyyn havaintoaineistoon voidaan käyttää standardeja käsittelymenetelmiä,
- imputointi täytyy tehdä vain kerran, joten se on nopeaa ja
- imputoinnissa voidaan ottaa huomioon taustatietämys (esimerkiksi muuttujien väliset riippuvuudet).

Yksikköimputoinnin haittana on se, että imputoiduista havaintojoukoista lasketut tunnusluvut (esimerkiksi keskiarvo, varianssi, jne) eivät sisällä oikeiden arvojen vaihtelevuutta. Moni-imputoinnissa (engl. *multiple imputation*) puuttuvat tiedot korvataan useilla imputoiduilla arvoilla. Näin muodostuu M-kappaletta imputoituja havaintojoukkoja kuvan 31 mukaisesti.

X	1	2	3				
1	1	?	6	X12(1)	X12(2)	...	X12(M)
2	?	2	7	X21(1)	X21(2)	...	X21(M)
3	9	8	?	X33(1)	X33(2)	...	X33(M)
4	?	4	2	X41(1)	X41(2)	...	X41(M)
5	11	13	?	X53(1)	X53(2)	...	X53(M)

Kuva. 31: Moni-imputoinnin havainnollistaminen

Moni-imputoinnin vaiheet ovat [22]

1. imputoitujen havaintojoukkojen luominen,
2. luotujen havaintojoukkojen analysointi ja

3. tulosten yhdistäminen.

Moni-imputoitujen arvojen yhdistäminen

Moni-imputoinnin keskeisimpänä etuna pidetään, että imputoitujen arvojen luotettavuutta voidaan arvioida perinteisten tilastollisten menetelmien avulla. Olkoon $\mathcal{Q}$ jokin tutkittava tunnusluku, esimerkiksi keskiarvo, ja olkoon $\hat{\mathcal{Q}}^j$ j:nnestä $(j = 1, 2, \ldots, m)$ täydennetystä 1-ulotteisesta havaintojoukosta $\{x^j(i)\}_{i=1}^n$ laskettu tunnusluvun estimaatti. Havaintojoukoista lasketut tunnusluvun estimaatit yhdistetään lopulliseksi estimaatiksi laskemalla niiden keskiarvo [17]

$$\overline{\mathcal{Q}} = \frac{1}{m} \sum_{j=1}^{m} \hat{\mathcal{Q}}^j.$$

Keskiarvon laskemisen oletuksena on, että tunnusluvun ja estimaatin erotus on normaalijakautunut (lähde [32], s.75). Lähteessä [40] sivuilla 47-48 esitetään menetelmä estimaatin $\overline{\mathcal{Q}}$ luotettavuuden tarkistamiseen.

Moni-imputoinnin tehokkuudella tarkoitetaan sitä kuinka luotettava m:n imputoinnin tuottama estimaatti on verrattuna siihen jos tehtäisiin ääretön määrä imputointeja. Lähteen [38] mukaan moni-imputoinnin saavuttaa tehokkuutensa jo pienellä imputointien lukumäärällä m. Imputointeja ei tarvita kuin 3–5 kappaletta ja saavutetaan suhteellisen hyvä tehokkuus laskettavalle estimaatille. Enempää imputointia ei yleensä tarvita ellei puuttuvuus ole erittäin suurta. Moni-imputointiin voi tutustua tarkemmin perehtymällä mm. lähteisiin [32] ja [37].

4.3 Mallipohjaiset menetelmät

Mallipohjaisessa imputoinnissa (engl. *model based imputation*) käytetään mallia, joka muodostaa relaation havaittujen ja puuttuvien arvojen välille. Malli käyttää taustatietona havaittuja arvoja ja ennustaa niiden avulla puuttuvia arvoja.

Keskiarvoistaminen

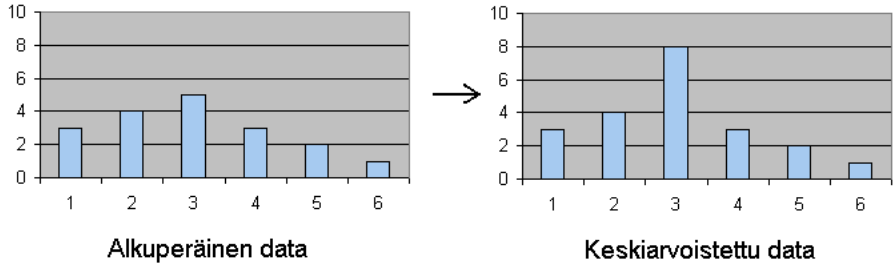
Keskiarvoistamisessa (engl. *mean substitution*) puuttuvat havainnot korvataan havain-

tojoukon keskiarvolla. Muuttujan X puuttuva arvo korvataan keskiarvolla

$$\overline{x} = \frac{1}{n_{\text{observed}}} \sum_{i=1}^{n_{\text{observed}}} x(i), \tag{4.4}$$

missä $\{x(i)\}_{i=1}^{n_{\text{observed}}}$ ovat havaitut muuttujan X arvot ja n_{observed} on niiden lukumäärä.

Menetelmä piikittää jakaumaa eli vähentää varianssia [29] ja tuo siten virhettä havaintoaineiston jakaumaan (kuva 32).



Kuva. 32: Keskiarvoistaminen

Menetelmää voidaan parantaa käyttämällä ryhmäkeskiarvoja (engl. *group mean substitution*) [1]. Puuttuvaa arvoa ei korvata koko muuttujan keskiarvolla vaan se korvataan aliryhmän keskiarvolla. Esimerkiksi jos kahden henkilön perhe ei ole kyselyssä ilmoittanut tulojaan niin heidän tuloikseen imputoidaan kaikkien kaksi-jäsenisten perheiden tulojen keskiarvo.

Arvojen ennustaminen normaalijakaumalla

Yksinkertainen menetelmä on ennustaa puuttuvia arvoja normaalijakauman avulla. Tällöin puuttuvien arvojen oletusarvoksi lasketaan havaintojoukon keskiarvo. Varianssi lasketaan myös havaintojoukosta. Tämän jälkeen puuttuvat arvot poimitaan satunnaisesti normaalijakaumasta sen tiheyksien suhteen.

Keskiarvoistamista voidaan myös parantaa lisäämällä sen tuottamaan vakioarvoon varianssia tuottava termi, joka on esim. normaalijakautunut, s.e. sen oletusarvo on nolla ja varianssi lasketaan havaintojoukosta. Keskiarvoistamisen virheensietokykyä (mm. havaintopoikkeamat) voidaan parantaa määrittelemällä robusti keskiarvo, joka on Turkey'n kolmois-keskiarvo (kts. kaava 3.1). Robustisesta keskiarvostakin voidaan määri-

tellä versio, joka tuottaa varianssia lisäämällä siihen kaavan 3.6 avulla laskettu varianssitermi, joka on myös normaalijakautunut ja oletusarvoltaan nolla.

Normaalijakaumaa voi käyttää sekä skalaari että monimuuttujaiseen imputointiin. Erona on ainoastaan, että monimuuttujaisessa tapauksessa havaintoaineistoon on sovitettava monimuuttujainen jakauma, josta imputoiva monimuuttujainen vektori $\mathbf{x}_{\text{imp}}$ poimitaan.

Regressiomenetelmien käyttö imputointiin

Keskiarvon käyttäminen on yksinkertainen esimerkki regressiomenetelmien käyttämisestä imputointiin. Yleisemmin regression käyttö imputoinnissa voidaan esittää muodossa

$$x_{\text{imp}} = E[X_{\text{mis}}|X_{\text{obs}}],$$

missä x_{imp} on (imputoitu-) muuttujan x arvo ja X on havainto-osan X_{obs} määräämä jakauma (satunnaismuuttuja) muuttujan x vaihtelulle. Yleensä taustalla on malli

$$X_{\text{mis}} = g(X_{\text{obs}}) + \epsilon,$$

missä $g(x)$ on jokin X_{obs}:n funktio ja ϵ on nollakeskiarvoinen satunnaistekijä. Esimerkiksi lineaarisessa regressiossa $g(x)$ on lineaarinen, eli $x_{\text{imp}} = \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_r x_r + \theta_0 + \epsilon$.

Koska MLP-tyyppinen neuroverkko on pohjimmiltaan regressiomenetelmä, soveltuu se hyvin regressioimputointiin. MLP neuroverkko opetetaan havaitulla aineistolla, missä syötteenä on täysin havaitut muuttujat ja ulostulona on puutteelliset muuttujat. Tämän jälkeen MLP:n avulla voidaan tehdä joko yksi- tai monimuuttujainen regressio. Warren S. Sarlen suorittamissa regressiotesteissä neuroverkot pärjäsivät aika hyvin [36].

EM-algoritmi

EM-algoritmin (engl. *Expectation Maximization algorithm*) perusideana on maksimoida todennäköisyyttä $\Pr(\mathbf{X}|\theta)$, jossa θ ovat havaintoihin sovitettavan mallin parametrit [33]. Ramonin ja Sebastianin mukaan EM-algoritmi tuottaa luotettavia estimaatteja ja sitä pidetään toimivana ratkaisuna puuttuvan datan ongelmaan [28]. Kirjoittamalla havaintovektori puuttuvan ja havaitun osan yhteisjakaumaksi saa todennäköisyys

muodon

$$\Pr(\mathbf{X}_{\text{missing}}, \mathbf{X}_{\text{observed}}|\theta) = \Pr(\mathbf{X}_{\text{missing}}|\mathbf{X}_{\text{observed}}, \theta)\Pr(\mathbf{X}_{\text{observed}}|\theta).$$

Tämän uskottavuusfunktio on muotoa

$$L(\theta|\mathbf{X}) = \prod_{j=1}^{n} \Pr(\mathbf{X}_{\text{observed}}(j)|\theta)\Pr(\mathbf{X}_{\text{missing}}(j)|\mathbf{X}_{\text{observed}}(j), \theta).$$

Yhtälön viimeinen termi $\Pr(\mathbf{X}_{\text{missing}}(j)|\mathbf{X}_{\text{observed}}(j), \theta)$ on ennustaja puuttuvan havaintoaineiston jakaumalle taustatiedoilla, joita ovat havaittu havaintoaineisto ja parametrit. Olettaen että parametreja ei tunneta, tätä termiä ei voida laskea. Se voidaan kuitenkin estimoida korvaamalla termi θ termillä $\hat{\theta}^{(i)}$, joka i:n arvolla 0 on alkuestimaatti parametrien arvoille. Yhtälöä käytetään yleensä regressioimputoinnin tavoin, jolloin puuttuvat arvot korvataan niiden odotusarvoilla $\mathbf{X}_{\text{imp}} = E[\mathbf{X}_{\text{missing}}(j)|\mathbf{X}_{\text{observed}}(j), \theta]$, mistä seuraa, että uskottavuusfunktio $L(\theta|\mathbf{X})$ voidaan tulkita odotusarvoiseksi uskottavuudeksi parametreille θ edellisten parametrien ja havaintojen avulla

$$L(\theta^i|\mathbf{X}) \approx E[L(\theta^i|\mathbf{X}_{\text{observed}}, \theta^{i-1})].$$

Algoritmi on iteratiivinen ja koostuu kahdesta eri vaiheesta. Jokaisella iteraatiolla parannetaan estimaattia $\theta^{(i)}$. Ensimmäisessä vaiheessa (E) saadaan jakauma uskottavuusfunktioita, joista valitaan odotusarvoisin. Toisessa vaiheessa (M) etsitään todennäköisimmät estimaatit parametreille maksimoimalla uskottavuusfunktiota. Menetelmä pyörii vaiheissa E ja M kunnes se konvergoituu eli $\theta^{(i)} \approx \theta^{(i-1)}$. Konvergointia voidaan nopeuttaa toteuttamalla algoritmista muunnettu versio [39]. Algoritmi löytää yleensä vain lokaalin maksimin. EM-algoritmia voidaan soveltaa kaikkiin mallipohjaisiin menetelmiin, mutta se on luontevin suhteellisen yksinkertaisten regressiomenetelmien yhteydessä joissa odotusarvo $E[\mathbf{X}_{\text{missing}}|\mathbf{X}_{\text{observed}}, \theta]$ on helppo laskea. EM-algoritmin idea on helppo yleistää tai varioida korvaamalla odotusarvo eli regressioimputointi jollain toisella menetelmällä. Yleisellä tasolla tätä kutsutaan joskus IP-menetelmäksi. Kuten EM-algoritmi niin myös IP-menetelmä (engl. *imputation posterior method*) konvergoi eli se löytää jonkun ratkaisun, joka ei välttämättä ole globaali optimi. IP-menetelmän alussa valitaan imputointimallin aloitusparametrit $\hat{\theta}^{(0)}$. Imputointivaiheessa (I) puuttuvan havaintovektorin i:s estimaatti $\hat{\mathbf{X}}^{(i)}_{\text{missing}}$ saadaan jakauman $p(\mathbf{X}_{\text{missing}}|\mathbf{X}_{\text{observed}}, \hat{\theta}^{(i-1)})$ avulla, jossa i on iteraatiokierros (i=1 alussa). Jälkivaiheessa (P) mallin parametrit $\hat{\theta}^{(i)}$ otetaan jakaumasta $p(\theta|\mathbf{X}_{\text{observed}}, \hat{\mathbf{X}}^{(i)}_{\text{missing}})$.

Muut, ei-tilastolliset mallipohjaiset imputointimenetelmät

Ei-tilastollisia mallipohjaisia imputointimenetelmiä ovat deterministinen imputointi, asiantuntijajärjestelmät ja viimeisen havainnon vieminen eteenpäin. Näistä kaksi ensimmäistä olettavat että imputoitaville muuttujille on olemassa säännöt, joiden avulla puuttuvien muuttujien arvot voidaan laskea. Viimeinen menetelmä on aikasarjahavaintoja varten. Se olettaa että muutos havaitun ja havaitsemattoman arvon aikavälillä on pieni. Deterministisellä säännöllä täydentäminen on determinististä imputointia (engl. *deterministic imputation*). Esimerkiksi puuttuva havainto saadaan muista havainnoista laskemalla numeroiden summa, jos kyseessä on summasääntö. Sääntöpohjaisten menetelmien ongelmatilanteeksi voi muodostua se, jos puuttuvia muuttujia on enemmän kuin yksi ja ne kaikki esiintyvät vain samassa säännössä. Asiantuntijajärjestelmät (engl. *expert systems*) ovat älykkäitä tietokoneohjelmia, jotka käyttävät tietämystä ja päättelymenetelmiä ongelmien ratkaisuun. Ne koostuvat aina kahdesta perusosasta, jotka ovat tietämyskanta (engl. *knowledge base*) ja päättelyjärjestelmä (engl. *inference engine*). Tietämyskanta koostuu faktoista ja heuristisista tiedoista. Tietämyskanta sisältää säännöstön, joka voi olla esimerkiksi seuraavanlainen:

IF X_1 THEN Y_1
IF X_2 THEN Y_2
$\vdots$
IF X_n THEN Y_n,
missä X_i ja Y_i ovat loogisia lauseita.

Yhdistämällä asiantuntijajärjestelmä ja esimerkiksi deterministinen imputointi voidaan rakentaa tehokkaita imputointijärjestelmiä. Täydennettäessä aikasarjahavaintoja voidaan joskus käyttää erittäin yksinkertaista menetelmää, jossa viimeinen havainto viedään eteenpäin (engl. *Last Value Carried Forward*). Pieni parannus menetelmään on käyttää lineaarista interpolointia. Voidaan myös käyttää menetelmiä, jotka ottavat huomioon useamman kuin kaksi peräkkäistä havaintoa. N:n asteen interpolointipolynomi ja MLP ovat eräitä tällaisiä menetelmiä.

4.4 Luovuttajamenetelmät

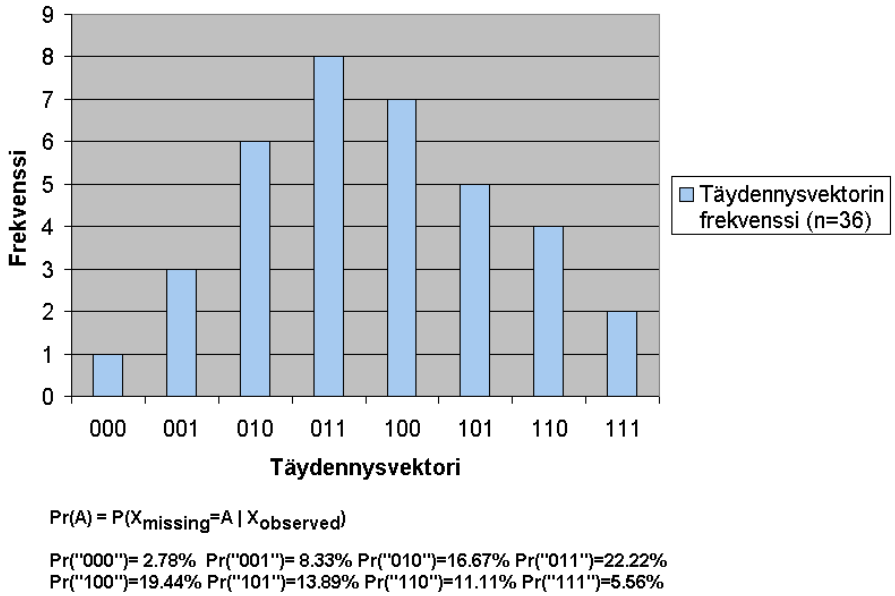
Luovuttajamenetelmissä puuttuvalle havainnolle etsitään lähin vastaava eli samankaltainen havainto ja täydennetään puuttuvat kohdat sen arvoilla.

Tasoimputointi

Tasoimputointi (engl. *deck imputation*) on menetelmä, jossa luovuttajaa käytetään toimittamaan puuttuvat arvot. Tasoimputointikategorioita on kaksi, jotka ovat hot-deck ja cold-deck. Hot-deck imputoinnissa puuttuva tieto etsitään samasta havaintojoukosta. Hot-deck menetelmän etuna on, että se säilyttää kategoriset muuttujat kategorisina ja jatkuvat jatkuvina. Hot-deck imputoinnin jälkeen havaintojoukko on täydellinen ja sitä voidaan käyttää. Haittapuolena menetelmässä on se, että samankaltaisuuden määrittäminen on vaikeaa. Ongelma tulee vastaan kun luovuttajaehdokkaita on useampia. Cold-deck imputointi on samanlainen kuin hot-deck paitsi että puuttuva tieto haetaan aiemmin tehdystä vastaavasta havaintojoukosta. Tasoimputointimenetelmiä ovat lähimmän naapurin menetelmä ja satunnainen luovuttaja. Lähimmän naapurin menetelmässä imputointi tehdään lähimmän luovuttajan havaitulla osalla, joka puuttuu imputoitavasta havainnosta. Etäisyys skaalataan havaittujen ja havaitsemattomien komponenttien erotuksella. Jos luovuttajaehdokkaita on useampi niin niistä poimitaan satunnaisesti yksi. Satunnaisessa luovuttajassa havaintojoukosta etsitään satunnainen luovuttaja, jolla on puuttuvat komponentit.

Ehdollinen tiheysfunktio

Käyttämällä ehdollista tiheysfunktiota $f(\mathbf{X}_{\text{missing}}|\mathbf{X}_{\text{observed}})$ voidaan imputoinnin tuottaman havaintoaineiston varianssi säilyttää. Keskiarvoimputointiin verrattuna menetelmä on mahdollisesti parempi koska taustatieto, eli havaittu havainnon osa, käytetään hyödyksi (kts. [9], s. 54). Lähteen [3] mukaan minkä tahansa yhden kiinteän täydennysarvon käyttäminen johtaa aina vääriin tuloksiin. Menetelmässä etsitään kaikki havaintovektorit, joissa on puuttuvan havainnon havaittu osa. Näitä vektoreita kutsutaan täydennysvektoreiksi. Tämän jälkeen lasketaan todennäköisyydet havaituille täydennysehdokkaille. Kuvassa 33 on havainnollistettu täydennysvektoreita ja niiden todennäköisyyksiä.



Kuva. 33: Täydennysvektoreiden frekvenssit

Esimerkki 4.4.1 (Täydennysvektori-imputointi [35])
Oletetaan että on tehty kysely 1140:ltä ihmiseltä, jossa jokaisen havainnon pituus on 12. Yksittäinen havainto $\mathbf{x}$ *on vektori:* $\mathbf{x} = (x_1, x_2, \ldots, x_{12})$. *Vektorin* $\mathbf{x}$ *komponenteille* $x_i, i \in [1, 12]$ *pätee:* $x_i \in [0, 2] \cap \mathbf{N}$. *Arvot 0, 1 ja 2 vastaavat tapauksia: 0="vastaus on myönteinen", 1="vastaus on kielteinen" ja 2="vastaus puuttuu". Estimoitava ehdollinen todennäköisyys on*
$\Pr(X_8 = x_8, X_9 = x_9, X_{10} = x_{10} | X_1 = 0, \ldots, X_7 = 0, X_{11} = 0, X_{12} = 0)$, *joka on (alla olevat on saatu havaintoaineistosta):*
$\frac{1120}{1140} = 0.9823$, *jos* $x_8 = 0, x_9 = 0, x_{10} = 0$,
$\frac{10}{1140} = 0.0088$, *jos* $x_8 = 1, x_9 = 0, x_{10} = 0$,
$\frac{4}{1140} = 0.0035$, *jos* $x_8 = 0, x_9 = 0, x_{10} = 1$,
$\frac{3}{1140} = 0.0026$, *jos* $x_8 = 1, x_9 = 1, x_{10} = 0$,
$\frac{2}{1140} = 0.0018$, *jos* $x_8 = 0, x_9 = 1, x_{10} = 0$,
$\frac{1}{1140} = 0.0009$, *jos* $x_8 = 1, x_9 = 0, x_{10} = 1$,
$\frac{0}{1140} = 0$, *jos* $x_8 = 0, x_9 = 1, x_{10} = 1$,
ja $\frac{0}{1140} = 0$, *jos* $x_8 = 1, x_9 = 1, x_{10} = 1$.

Generoidaan satunnainen numero s nollan ja ykkösen väliltä. Tämän jälkeen jaetaan reaaliakselin väli $[0, 1]$ *kahdeksaan osaan, jotka vastaavat edellä mainittuja ehdollisia*

todennäköisyyksiä. Osien pituudet ovat siis, 0.9823, 0.0088, 0.0035, 0.0026, 0.0018, 0.0009, 0 ja 0. Lopulta tutkitaan mihin väliin s osuu, väli määrittelee havainnon $\mathbf{x}$ *imputoitavien komponenttien* x_8*,*x_9 *ja* x_{10} *arvot. Esimerkiksi jos* $s \in [0, 0.9823]$ *niin* $x_8 = 0, x_9 = 0$ *ja* $x_{10} = 0$*.*

4.4.1 Imputointivirheiden todennäköisyys

Luovuttajamenetelmissä imputointivirheen todennäköisyyttä voidaan arvioida. Todennäköisyyden avulla voidaan estimoida kuinka monta virheellistä havaintoa imputoinnissa tehdään. Olkoon imputoinnissa käytettävä luovuttajajoukko $\{\mathbf{x}(i)\}_{i=1}^{n}$ ja puutteellinen täydennettävä havainto $\mathbf{x} \in \{\mathbf{x}(i)\}_{i=1}^{n}$. Estimaatti yksittäisen havainnon imputointivirheen todennäköisyydellä on (lähde [35],The Expected Number Of Incorrect Imputations)

$$\Pr(\text{"Imputointivirhe"}) = 1 - \sum_{j=1}^{n} \Pr\{\mathbf{x}_{\text{missing}} = \mathbf{x}_{\text{observed}}(j) | \mathbf{x}_{\text{observed}}\}^2, \tag{4.5}$$

missä $\Pr\{\mathbf{x}_{\text{missing}} = \mathbf{x}_{\text{observed}}(j) | \mathbf{x}_{\text{observed}}\}$ on todennäköisyys että havainnon $\mathbf{x}$ puuttuva osa on $\mathbf{x}_{\text{observed}}(j)$. Todennäköisyydet saadaan muodostamalla havaintoaineistosta jakauma $p_{\text{X}}(\mathbf{X}|\mathbf{x}_{\text{observed}})$. Yhtälöstä 4.5 saadaan estimaatti n_s:n samanlaisen imputointitapauksen virheiden lukumäärälle

$$n_{\text{ie}} = n_s \Pr(\text{"Imputointivirhe"}). \tag{4.6}$$

4.5 Ryvästelymenetelmien ja itseorganisoituvan piirrekartan käyttö imputointiin

Havaintoaineiston ryvästelyllä imputointi yleensä tehostuu. Ryvästelyllä saadaan imputoituihin arvoihin varianssia vaikka imputointimenetelmä ei sitä itse tuottaisikaan. Tämä johtuu siitä, että imputointi tehdään jokaisessa ryppäässä joissa on eri osa havaintojoukosta, ja siten niissä laskettavat tunnusluvut, esimerkiksi keskiarvo, poikkeavat usein toisistaan. Ryvästelyllä voidaan myös joskus nopeuttaa imputointia, koska

käsiteltävä havaintojoukko jaetaan osiin, jolloin koko tehtävän laskennallinen kompleksisuus saattaa vähentyä. Ryvästelyssä on käytössä luokitus ryppäisiin c_i eli joukot $V(i)$, missä $i = 1 \ldots k$ ja k on ryppäiden lukumäärä. Ryppäät koostuvat d-ulotteisista havaintovektoreista. Joukko $V_n(i)$ koostuu ryppään i havaintovektoreiden n.:istä komponenteista.

Olkoon $\mathbf{w}(i)$ ryppään i keskipiste eli $\mathbf{w}(i) = (w_1(i), w_2(i), \ldots, w_d(i))$, missä $w_n(i), n = 1 \ldots d$ on

$$\frac{1}{\#\{V_{\text{obs}(n)}(i)\}} \sum_{x_n(j) \in V_{\text{obs}(n)}(i)} x_n(j),$$

missä $V_{\text{obs}(n)}(i)$ joukon $V_n(i)$ täysin havaitut havainnot.

Kuulukoon puutteellinen havainto $\mathbf{x}$ ryppääseen c_i eli $\mathbf{x} \in V(i)$. Imputointi voitaisiin suorittaa ryppään c_i avulla, jos jokaisessa ryppässsä olisi täysin havaittuja havaintoja. Käytännössä näin ei ole vaan on täysin mahdollista, että ryppäässä on pelkästään puutteellisia havaintoja. Jos ryppään havainnot yhdistämällä ei saada edes yhtä kokonaista havaintoa, niin on käytettävä sopivia havaintojoukkoja ryppään naapurustosta. Sopiva havaintojoukko $V(l)$ eli rypäs c_l voidaan etsiä havainnolle $\mathbf{x} \in V(i)$ seuraavan kriteerin avulla

$$l = \arg\min_m \frac{d}{d - d_{\text{missing}}} ||\mathbf{x}_{obs} - \mathbf{w}_{obs}(m)||, \text{ ehdolla } \#\{V_{\text{obs}}(m)\} \geq p \qquad (4.7)$$

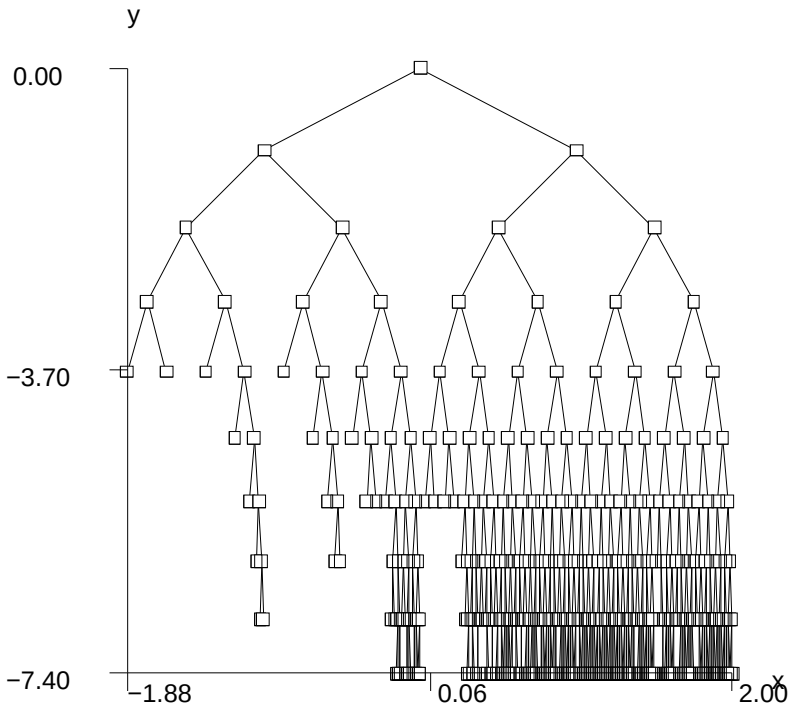
missä $m = 1 \ldots k$, d on havaintojen dimensio, d_{missing} on havainnon $\mathbf{x}$ puuttuvien komponenttien lukumäärä, $\mathbf{x}_{obs}$ on havaintovektorin $\mathbf{x}$ havaitut komponentit, $\mathbf{w}_{obs}$ on niitä vastaavat ryppään komponentit ja p määrittelee kuinka monta täysin havaittua havaintoa ryppäässä l on vähintään oltava.

Jos ryppään keskipisteen kaikki komponentit puuttuvat ($d = d_{\text{missing}}$), niin tällöin etäisyydeksi määritellään ∞ eli käytännössä tietokoneen liukuluvun maksimiarvo. Kriteeri 4.7 varmistaa, että imputoitava arvo on täysin havaittu ja toisaalta se takaa että käytettävän ryppään c_l havainnot ovat mahdollisimman samankaltaisia kuin alkuperäisen ryppään c_i. Optimaalisimmassa tilanteessa $c_l = c_i$. Käytettävän ryppään l määrittämisen jälkeen voidaan havaintovektori imputoida mm. edellä esitetyillä mallipohjaisilla- ja luovuttajamenetelmillä muodostamalla lokaali imputointimalli ryppäässä c_l. Tämän jälkeen ryppäässä tehdään imputointi "normaaliin" tapaan.

Puurakenteiset ryvästelymenetelmät imputoinnissa

Puurakenteisissa ryvästelymenetelmissä rakennetaan hierarkkinen malli havaintoaineiston jakaumasta (kuva 34). Tutkitaan esimerkkinä yksinkertaista 1-ulotteista binaaripuuta, jossa jokainen solmu jakautuu kahtia mentäessä tarkemmalle tasolle. Jakauman mallin muodostaminen voidaan tehdä esimerkiksi käyttämällä K-Means ryvästelyalgoritmia. Ensin muodostetaan yksi rypäs, josta tulee puun juurisolmu. Solmu sisältää keskipistearvon ja siihen luokittuvat havaintojen indeksit. Tämän jälkeen yhteen ryppääseen luokittuvat havainnot eli koko havaintoaineisto jaetaan kahteen ryppääseen. Tätä jatketaan rekursiivisesti kunnes päästään haluttuun tarkkuuteen. Puurakenteen etuna on, että sen rakentaminen on myös tehokasta, koska edellistä hierarkiatasoa voidaan käyttää apuna tehtäessä tarkempaa tasoa. Tällöin ryvästely nopeutuu huomattavasti. Pohjatasona on taso, jossa jokainen havainto luokittuu omaan ryppääseen. Tosin niin tarkkaan hierarkiaan ei ole välttämättä järkevää mennä, koska kyseinen esitys vaatii paljon muistia. Toisena vaarana on havaintoaineiston ylioppiminen.

Käytettäessä puurakenteista ryvästelymenetelmää imputointiin tutkitaan ensin mihin valitun hierarkiatason solmuun eli ryppääseen havaintovektori luokittuu sen havaitun osan $\mathbf{x}_{\text{obs}}$ avulla. Solmu etsitään laskemalla etäisyys havaitun osan ja ryppään keskipisteen välillä. Näin käsiteltävä havaintojoukko vähenee huomattavasti koska puurakenne jakaa sen pienempiin paloihin. Käytettävä hiearkiataso vaikuttaa usein imputoinnin laskennalliseen aikavaativuuteen ja imputointimalli(e)n kompleksisuuteen. Puurakenteen etuna on, että imputointia saadaan nopeutettua huomattavasti koska käsiteltävä havaintojoukko saadaan pienemmäksi. Toisaalta kriteerissä 4.7 etsittävä havaintojoukko voidaan mahdollisesti löytää nopeasti ylemmältä eli epätarkemmalta hierarkiatasolta. Kaikki hakualgoritmit saadaan toimimaan huomattavasti tehokkaammin, koska puurakenne mahdollistaa tehokkaan hakuavaruuden rajoittamisen.

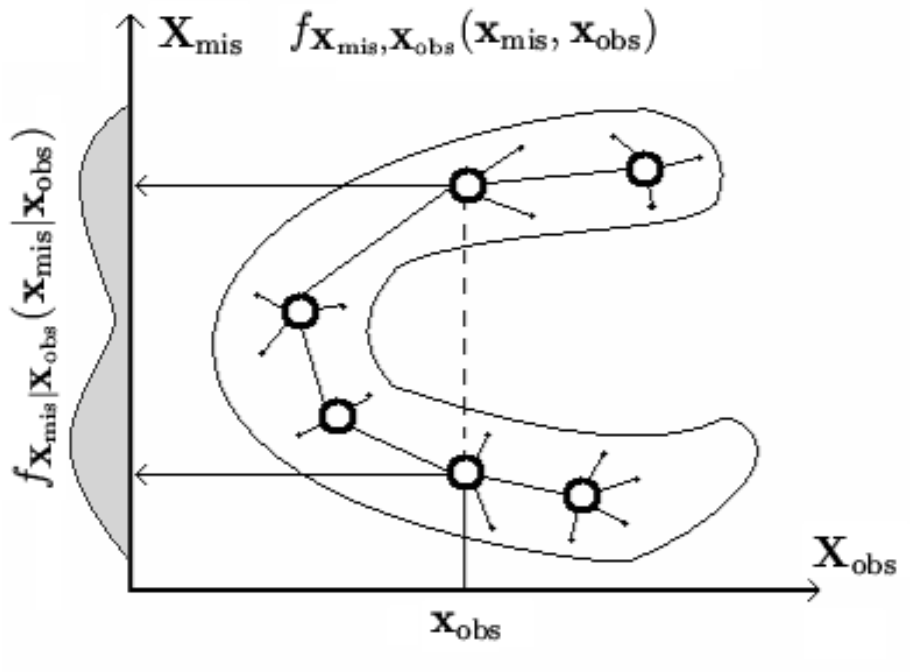


Kuva. 34: Puurakenteinen ryvästely

Itseorganisoituva piirrekartta (SOM) ja sen puurakenteinen versio (TS-SOM) imputointimenetelmänä

Kuten luvussa 2 esitettiin, voidaan itseorganisoituva piirrekartta eli SOM tulkita projektiomenetelmien ja ryvästelymenetelmien välimuodoksi. Näin sitä voi käyttää imputoinnissa kuten ryvästelymenetelmiä. Vastaavasti puurakenteinen itseorganisoituva piirrekartta TS-SOM tuo SOM:n käyttöön samat edut kuin puurakenteiset ryvästelymenetelmät yleisemmin. Menetelmän käyttöä on havainnollistettu oheisessa kuvassa 35. Puutteellisen havainnon havaitulla osalla $\mathbf{x}_{\text{obs}}$ etsitään jakauma samankaltaisia neuroneita (engl. *BMU - Best-Matching-Unit*), joita on kaksi kuvassa 35. Tämän jälkeen puuttuva osa $\mathbf{x}_{\text{imp}}$ täydennetään jonkin BMU:n vastaavalla osalla (joka on havaittu) tai neuroniin luokittuvien havaintojen avulla. BMU:n valinnassa käytetään jakaumaa $f_{\mathbf{X}_{\text{missing}}|\mathbf{X}_{\text{observed}}}(\mathbf{x}_{\text{missing}}|\mathbf{x}_{\text{observed}})$ jolloin imputoituihin arvoihin saadaan alkuperäisen havaintojoukon mukainen varianssi. Kun käytettävä neuroni on löydetty, niin impu-

tointi tehdään siinä normaaliin tapaan.



Kuva. 35: Imputointi SOM:in avulla

TS-SOM rakentuu useasta SOM:ista, joten se soveltuu myös varsin hyvin imputointiin. TS-SOM imputoinnin etuna on että sen avulla laskennallinen kompleksisuus saadaan yleensä melko alhaiseksi. Tämä johtuu TS-SOM:in puurakenteesta, jonka avulla saadaan nopeutettua useita algoritmeja paljon. Esimerkiksi normaalin lähimmän naapurin menetelmän kompleksisuus N:lle havainnolle on $O(N^2)$. Olkoon TS-SOM:issa neuronien lasten lukumäärä p. Jos havaintoaineisto on tasaisesti jakautunut neuroneihin, niin TS-SOM:illa tehtynä menetelmän kompleksisuus on (lähde [15], s. 66)

$$O(N\log_p(M) + N^2/M),$$

missä M on käytettävän TS-SOM:in kerroksen neuronien lukumäärä, $O(N\log_p(M))$ on TS-SOM algoritmin kompleksisuus ja $O(N^2/M)$ on M:n ryppään imputoinnin kompleksisuus.

Luku 5

Editointi- ja imputointiohjelmisto

Tässä luvussa esitetään työn keskeinen sisältö, joka on työssä suunniteltu ja toteutettu editointi- ja imputointiohjelmisto.

Kuten luvussa 1 todettiin, on havaintoaineiston virheet ja puutteellisuudet syytä tunnistaa ja korjata ennen aineiston varsinaista hyödyntämistä. Esimerkiksi havaintoaineiston yli laskettavia tunnuslukuja, agregaatteja määrättäessä, voi virheellinen havaintoaineisto vaikuttaa tulosten totuudenmukaisuuteen tavalla, mikä ei ilmene perinteisillä luotettavuutta mittaavilla testeillä. Tämän työn alkuperäisenä tavoitteena oli tehdä ohjelmisto puutteellisuuksien korjaamiseen eli imputointiin. Tehtäessä ohjelmistoa kävi ilmi, että samalla työllä, käyttäen NDA-ohjelmiston työkaluja, voidaan toteuttaa myös sääntöpohjainen editointi suhteellisen helposti.

Luvussa 2 kuvattua mallintamista on hyödynnetty ohjelmistossa mm. havaintojoukon jakauman kuvaamiseen sekä havaittujen ja havaitsemattomien muuttujien välisen relaation muodostamiseen (regressio). Ohjelmistolla voidaan muodostaa ensin malli havaintoaineiston jakaumasta erilaisilla ryvästelymenetelmillä, joita ovat esimerkiksi TS-SOM, K-Means, Fuzzy C-Means ja hierarkkiset ryvästelymenetelmät. Tämän jälkeen ryppäissä voidaan mallintaa relaatiota havaintojoukon havaitun, eli taustatiedon, ja havaitsemattoman osan välillä esimerkiksi MLP-verkolla. Toisaalta tätä relaatiota ei ole pakko muodostaa eksplisiittisesti esimerkiksi hyödynnettäessä luovuttajamenetelmiä, joissa relaatio saadaan havaintojoukon itsensä avulla.

Työ on tehty osana yhteiseurooppalaista EurEdit tutkimushanketta, jonka rahoittaja on Euroopan komission alainen EUROSTAT -tilastovirasto. Projektin osapuolet, mm. Britannian tilastokeskus (ONS, Office for National Statistics), Tanskan tilastokeskus (StatDK, Statistics Denmark) ja Sveitsin tilastokeskus (SFSO, Swiss Federal Statistical Office) ovat antaneet tutkimushankkeen käyttöön tyypillisiä virheitä ja puutteellisuuksia sisältäviä havaintoaineistoja. Tässä työssä käytetyt aineistot ovat:

- Danish Labour Force Survey (DLS)
  DLS on synteettinen otos Tanskan vuoden 1996 väestörekisteristä, jossa on mm. tiedot ihmisten sukupuolesta, iästä, siviilisäädystä ja koulutuksesta. Aineiston ensisijainen tarkoitus on antaa kuva Tanskan työmarkkinoiden tilanteesta. Vastaavia kyseleitä tehdään useissa maissa samalla konseptilla, minkä ansiosta kyselyt ovat vertailukelpoisia. Alkuperäisessä havaintoaineistossa ei ollut puuttuvuuksia. Aineistossa on puuttuvuutta vain INCOME-muuttujassa.
- UK Annual Business Inquiry (ABI)
  ABI testiaineisto sisältää osan vuonna 1997-1998 yrityksiltä tehdystä kyselystä. Testiaineistossa on kaksi teollisuuden toimialasektoria. Kysely sisältää tietoa työllisyydestä, energiasta, taloudesta ja ostetuista palveluista. Havaintoaineisto on hierarkkinen ja siinä on muuttuja, joka ilmaisee havainnon hierarkialuokan. Hierarkialuokitus on tosin anonymisoitu. Otospainot ovat mukana havaintokohtaisesti. Havaintoaineisto sisältää lyhyen ja pitkän kyselyn, ja siitä tehdään johtopäätöksia mm. työllisyydestä.
- UK Household SARS
  Household SARS on 1%:n otos vuoden 1991 UK Census -havaintoaineistosta. UK Census -kysely tehdään kymmenen vuoden välein, mistä johtuu että aineiston sisältö on vanhaa. Kysely kattaa koko populaation lukuunottamatta vastaamattomien osaa. Kyselyn tarkoituksena on antaa tietoa työllisyys- ja työttömyystilanteesta (International Labour Organisation -määritys). 1991 vuoden kysely ei noudattanut täysin tätä määritystä. Se sisältää tietoja ihmisten talouksista, kuten huoneiden määrän, lämmityksen, autojen määrän ja siviilisäädyn. Testiaineiston kaikissa muuttujissa on puuttuvuutta.

Luvussa 3 todettiin että virheitä voidaan tunnistaa joko mallipohjaisilla- tai sääntöpoh-

jaisilla menetelmillä. Ohjelmistoon toteutettiin jälkimmäinen menetelmä, koska Britannian tilastokeskuksen käyttämät loogiset editointisäännöt UK Household SARS- ja UK ABI- testiaineistoille olivat käytettävissä. Editointisääntöjen avulla voidaan havaita osa loogisista virheistä ja parhaassa tapauksessa korjata ne (poistamalla virheellinen havainto ja merkitsemällä imputoitaviksi). On hyvä huomata, että editointisäännöillä voidaan myös validoida imputointimenetelmiä. Imputoinnin jälkeen tehtävällä nk. post-edit toiminnolla voidaan tarkistaa, ettei imputointi ole tuottanut uusia virheitä. Optimaalisimmassa tapauksessa virheitä ei ole yhtään puuttuvien havaintojen täydentämisen jälkeen. Tehokkaimmaksi editointisääntöjen käytön saisi yhdistämällä ne imputointimenetelmiin, jolloin imputoinnissa voitaisiin ainakin osittain välttyä uusien virheiden tuottamiselta. Tämä on käytännössä hankala toteuttaa, ohjelmistossa editointivaihetta ei ole yhdistetty imputointivaiheeseen, vaan ne tehdään erikseen. Editointi- ja imputointi suoritetaan usein erikseen [8]. Ideaalista, joskin käytännössä mahdotonta olisi, että havaintoaineistosta tehtävät imputointimallit oppisivat esimerkeistä tekemään loogisesti järkeviä imputointeja, esimerkiksi muodostettaessa MLP-menetelmällä editointisääntöjen suhteen virheetön relaatio havaintojoukon havaittujen ja havaitsemattomien osien välillä.

Työssä kehitetyllä ohjelmistolla voidaan tehdä sekä yksikkö- että moni-imputointia. Sensijaan luvussa 4 esiteltyjä aikasarja-, EM-, IP- ja deterministisiä imputointimenetelmiä ei ole toteutettu ohjelmistoon. Imputointivaiheessa ei ole käytössä asiantuntijajärjestelmää, mutta editointivaihetta voidaan pitää asiantuntijajärjestelmän jonkinasteisena päättelyosana: ihmisen on annettava sille tietämys eli editointisäännöt. Ryvästelymenetelmien ja TS-SOM:in hyödyntäminen imputoitaessa on ohjelmistossa myös mahdollista ja suositeltavaa, koska niillä päästään usein suhteellisen tarkkaan havaintoaineiston jakauman mallintamiseen.

Ohjelmiston tuotantoprosessia on kuvattu kappaleissa: analyysi (5.1), suunnittelu (5.2), toteutus (5.3) ja testaus ja arviointi (5.4). Sovellus-alikappaleessa (5.3.2) on esitelty toteutunut sovellus edeten sivulla 62 olevan kuvan 38 tiedon tuotantoprosessin vuokaavion mukaisesti. Alikappale sisältää myös suunnitteluvaiheen määritelmiin tehtyjen lisäyksien kuvaukset.

5.1 Analyysi

Olio-ohjelmoinnin terminologiasta lainattu analyysi käsittää ohjelmiston tavoitteiden määräämisen ja toiminnallisten vaatimusten spesifioinnin. Tässä yhteydessä analyysi on jaettu erikseen ohjelman yleisiin vaatimuksiin ja käyttöliittymän spesifikaatioon.

5.1.1 Tavoitteet

Tiivistettynä editointi- ja imputointiohjelmiston keskeiset tavoitteet ovat

i) Ohjelmistolla voidaan kokeilla puurakenteisen itseorganisoituvan piirrekartan (TS-SOM) soveltuvuutta erityyppisten havaintoaineistojen editointiin ja imputointiin.

ii) Ohjelma toimii sekä Windows että Linux käyttöjärjestelmissä.

iii) Ohjelma sopii sekä asiantuntijoiden käyttöön, että satunnaisille käyttäjille.

iv) Ohjelman tulokset voidaan analysoida sekä suhteessa imputoinnin onnistumiseen että laskennalliseen tehokkuuteen.

v) TS-SOM tyyppisen imputoinnin lisäksi ohjelmisto mahdollistaa vertailun vuoksi myös muuntyyppisten imputointimallien käytön.

Ensimmäinen vaatimus tulee EurEdit tutkimushankkeen tavoitteista, missä puurakenteisen itseorganisoituvan piirrekartan (TS-SOM) käyttö on nimenomaan Jyväskylän ryhmän keskeinen tavoite. Osoittautuu, että TS-SOM-algoritmin käyttäminen on hyödyllistä koska menetelmä on nopea ja varsin hyvä mallintamaan datan jakaumaa. Algoritmi tukee hyvin myös isojen datamassojen käsittelyä. Sen multiresoluutio-ominaisuuden avulla voidaan isoihin datamassoihin sovittaa nopeasti halutun tarkkuustason malli.

Kyky isojen datamassojen käsittelyyn on ehdoton vaatimus sovellukselle, koska mm. kyselytutkimuksen havaintoaineistot saattavat olla valtavia. Käytännössä manuaalinen imputointi on suurien datamassojen kanssa mahdotonta. Siksi prosessin päävaiheet

onkin voitava automatisoida ja näin voidaan muodostaa "linjasto", joka korjaa ja täydentää sisääntulevat havaintoaineistot.

Käytännössä vaatimukset i) ja ii) edellyttävät tilastollisen tiedon tuotantoprosessin ohjelmallisen toteutuksen suunnittelua. Tilastollinen tiedon tuotantoprosessi on tarpeellinen koska havaintoaineisto on voitava "jalostaa" muotoon, josta voidaan tehdä mahdollisimman virheettömät johtopäätökset (esimerkiksi vähittäiskaupan tuotteiden tilaus). Koko prosessi on syytä rakentaa yhteen ohjelmaan sillä muutoin sen tekeminen on työlästä. Jos dataa joudutaan käsittelemään eri ohjelmilla, niin työn tehokkuus putoaa usein huomattavasti. Kaiken tarpeellisen toiminnallisuuden integroiminen yhteen sovellukseen on täten varsin perusteltua. Samalla on mahdollista ottaa huomioon satunnaisten käyttäjien tarpeet rakentamalla käyttöliittymä siten, että perustoiminnot voidaan suorittaa ilman suurempia taustatietoja.

Projektiin osallistuneilta tilastokeskuksilta saatujen reaalimaailman esimerkkiaineistojen avulla voidaan testata sovelluksen ja menetelmien toimivuutta. Esimerkkiaineistojen käsittely sisältää mm. tarpeettomien taustamuuttujien poistamisen, ordinaali- ja nominaaliasteikollisten muuttujien koodaamisen ja muuttujien vaihteluvälien yhdenmukaistamisen. Testauksessa tutkitaan miten tehty sovellus toimii käytännössä ja kuinka luotettavia sen tulokset ovat.

Sovelluksen on hallittava NDA-ohjelmiston nimiavaruuden tietoja automaattisesti. Tällöin normaalikäyttäjän ei tarvitse tietää ohjelman käyttöön liittyviä teknisiä yksityiskohtia. Tämä ominaisuus helpottaa huomattavasti sovelluksen käyttämistä.

5.1.2 Käyttöliittymä

Käyttöliittymä on määritelty siten, että käyttäjän on pystyttävä tekemään editointia ja imputointia ilman NDA:n makrokielen (kts. 5.2.1) käyttämistä. Tähän määrittelyyn päädyttiin sen vuoksi, että kohderyhmän (tilastokeskuksen väki) ei yleisesti ottaen voi olettaa osaavan NDA:n makrokieltä ennestään ja toisaalta siksi että usein toistettava toiminnallisuus voidaan automatisoida. Tärkeää oli myös tehdä sovelluksen ulkoasusta selkeä ja yksinkertainen. Toisaalta haluttiin kuitenkin sallia ammattilaiselle mahdolli-

suus prosessiparametrien asettamiseen. Käyttöliittymän on visualisoitava prosessi siten että käyttäjä tietää missä vaiheessa hän on. Prosessi on oltava keskeytettävissä kaikissa vaiheissa lukuunottamatta vaiheita, joissa suoritaan NDA:n komentoja. Yleisenä toiminnallisuutena jokaisessa sovelluksen näkymässä on, että ensin toiminnolle asetetaan taustatieto (esimerkiksi imputointimenetelmä ja sen parametrit). Tämän jälkeen painetaan toimintapainiketta, joka suorittaa valitun toiminnan. Näin prosessi etenee vaiheittain. Prosessin vaiheiden näkymät keskitetään keskelle näyttöä. Tämä sen vuoksi, että kiinteillä koordinaateilla näkymä voi jäädä kuvan ulkopuolelle tai sen sisältö on vaikeasti luettavissa jostain kuvaruudun "kulmasta".

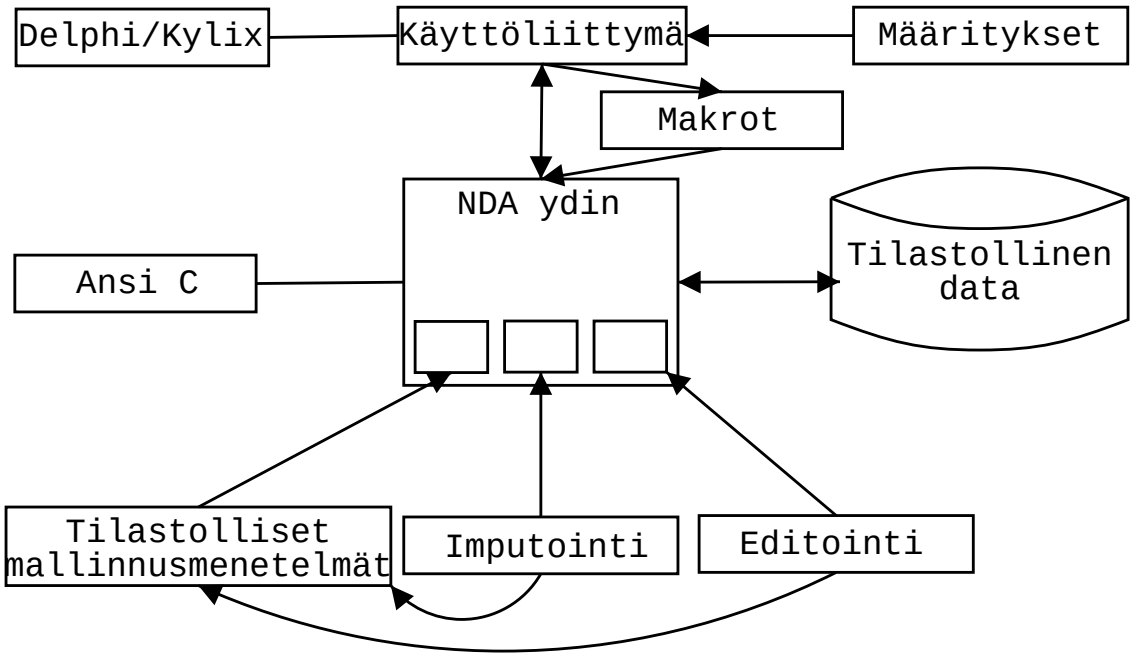
5.2 Suunnittelu

Ohjelmiston suunnitteluvaiheen keskeinen tehtävä on määrätä sovelluksen ohjelmistoarkkitehtuuri ja toteutustapa. Tämän lisäksi suunnitelma sisältää kattavamman spesifikaation siitä minkätyyppisiä toimintoja tarvitsee ohjelman tukea. Toimintojen määrääminen ja niiden muodostaman toimintaketjun kuvaus on tehdyn suunnitelman konkreettinen lopputulos.

5.2.1 Valittu ohjelmistoarkkitehtuuri

Ohjelmointiympäristönä NDA-ohjelmiston lisäksi päätettiin käyttää Borlandin Delphi 5, Delphi 6 ja Kylix kehitystyökaluja. Kyseinen ohjelmointiympäristö valittiin, koska ohjelman haluttiin olevan saatavilla Linux- ja Windows-käyttöjärjestelmiin. Borlandin sovellusten tulevaisuus näytti suhteellisen valoisalta, koska Kylix ja Delphi 6 olivat juuri julkaistu. Delphin ensimmäinen versio ilmestyi vuonna 1995. Valinnan taustalla on myös kesän 2001 aikana kyseisellä ohjelmointiympäristöllä (lukuunottamatta Delphi 6:sta, joka ei ollut vielä saatavissa) tehty kokeiluluonteinen graafinen käyttöliittymä NDA-ohjelmistolle. Ohjelmisto käyttää kaikkeen laskentaan ja mallinnukseen Neural Data Analysis -ohjelmistoa, joka on suunniteltu sulautettavaksi tämäntyyppisiin sovelluksiin. NDA:n käyttäminen on perusteltua koska siinä on ennestään käsittelymenetelmiä puuttuvalle datalle. Sen valintaa puoltaa myös seikka,

että se on tehty Ansi C:llä, joka on varsin siirrettävää. NDA tarjoaa myös TS-SOM-algoritmin lisäksi joukon muita datan mallinnusmenetelmiä ja ominaisuuksia, joiden avulla ohjelmiston voi toteuttaa. NDA:n etuna on myös se, jos tässä työssä tehty käyttöliittymä halutaan joskus korvata toisentyyppisellä toteutuksella, niin koko ohjelmistoa ei tarvitse tehdä alusta alkaen. Käyttöliittymän vaihtamista helpottaa, että kaikki toiminnallisuus on NDA:ssa ja sen makroissa. Täten NDA ja ohjelmiston makrot tukevat editointi- ja imputointitoimintojen uudelleenkäytettävyyttä. Varsinainen ohjelman suoritus tehdään NDA-ytimessä, mitä ohjataan käyttöliittymästä lähetettävillä makrokutsuilla. Makrokutsut välittyvät editointi- ja imputointiohjelmistolta pääohjelmalle, joka välittää ne edelleen NDA-ytimelle. Tämän jälkeen tietokoneen laskentateho siirtyy NDA-kirjastolle. Ohjelman käyttöliittymän päivitys keskeytyy laskennan ajaksi koska sovellukseen ei ole toteutettu erillistä säiettä tätä varten. Toiminnon jälkeen käyttöliittymän päivitys jatkuu. Pääohjelma välittää paluuarvo(t) ohjelmalle. Sovelluksen arkkitehtuurikaavio on kuvassa 36.

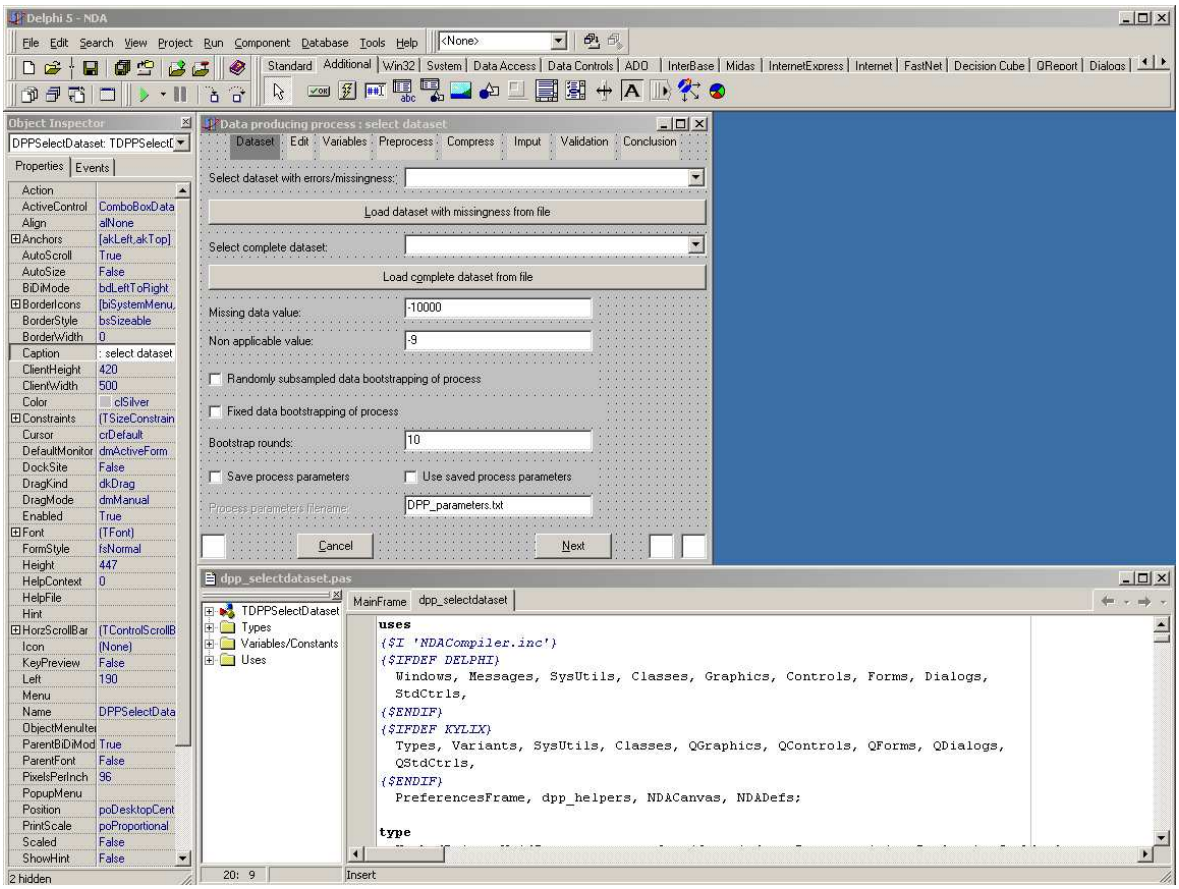


Kuva. 36: Sovelluksen arkkitehtuuri

Ohjelmaan voidaan ladata havaintoaineistoja NDA:n formaatissa.

Borland Delphi 5, Delphi 6 ja Kylix

Delphi 5, Delphi 6 ja Kylix ovat oliopohjaisia ohjelmiston kehitysympäristöjä. Delphissä ja Kylixissä on käytössä Object Pascal-ohjelmointikieli. Niiden suosio perustuu mm. siihen, että ne ovat komponenttipohjaisia, joka tarkoittaa, että sovellusta tehtäessä on käytössä valmiiksi tehtyjä ja uudelleenkäytettäviä olioita. Tämän ansiosta erityisesti käyttöliittymien tekeminen on erittäin nopeaa. Käytetyissä kehitysympäristöissä on työkalurivi, joka sisältää graafisen, drag-and-drop käyttöliittymän komponenttien sijoitteluun kuvaruudelle (kuvassa 37: Standard, Additional, jne). Näitä komponentteja ovat mm. painikkeet ja tekstit. Ympäristöt sisältävät aktiivisen olion muokkaamiseen tarkoitetun työkalun, Object Inspector (kuva 37). Työkalulla voidaan muokata olion ominaisuuksia, kuten tekstin sijoittelua, fonttia, sijaintia ja kokoa. Aktiivinen olio esimerkiksi työn alla oleva Data Producing Process : select dataset lomake näytetään käyttäjälle ruudulla. Tämän lisäksi on myös mahdollista nähdä olion toiminnallisuus ja tapahtumakäsittelijöiden koodi, esimerkissämme dpp_selectdataset.pas.



Kuva. 37: Borland Delphi 5:n graafinen käyttöliittymä

Oliopohjaisuus

Oliopohjaisuus tarkoittaa, että kokonaisuudesta voidaan muodostaa rakenne, joka on helposti uudelleenkäytettävissä. Olion rakenne sisältää ominaisuudet ja toiminnallisuuden (sisältäen tapahtumakäsittelijät). Tällainen rakenne voi olla esimerkiksi graafinen painike, joka osaa piirtää itsensä graafisesti (toiminnallisuus), sisältää sijainnin, tekstin ja värin (ominaisuudet). Se voi myös esimerkiksi käyttäjän painaessa painiketta ilmoittaa siitä sovellukselle (tapahtumakäsittelijä). Komponentti on olio, joka on peritty yhdestä tai useammasta oliosta. Periminen tarkoittaa sitä että olio saa perittävän olion rakenteen.

Neural Data Analysis -ohjelmisto

Neural Data Analysis, lyhyemmin NDA, on ohjelmistoympäristö, joka sisältää useita laskennallisesti älykkäitä työkaluja mm. sumeaan logiikkaan, ryvästelymenetelmiin ja puuttuvien havaintojen käsittelyyn. NDA on toteutettu C-ohjelmakirjastoston ympärille, joten sitä voidaan käyttää useassa eri ympäristössä (Windows, Linux, Solaris, jne). Näin myös uudet käskyt ja algoritmit ohjelmoidaan C-kielellä. NDA-ohjelmisto sisältää oman komentokielen ja tulkin, joten suurin osa ohjelmankehityksestä voidaan tehdä NDA:n omalla makrokielellä. NDA:n toteutuksen ja käytön kannalta keskeinen rooli on nimiavaruudella, joka ylläpitää ohjelman suorituksen aikana ylläpidettäviä tietorakenteita. Nimiavaruuden rakenteita ovat kentät (muuttujat), tietokehykset (sisältävät useita kenttiä), luokituskehykset (ryvästelymenetelmien luokitukset). Tämän lisäksi on olemassa erityisrakenteita, kuten matriisi-, kuva-, Fuzzy-, SOM-, MLP- ja grafiikkarakenteet.

NDA:n makrokieli

Makrokieli koostuu peräkkäin suoritettavista komennoista. NDA suorittaa makroja käsky kerrallaan. Käytännössä kätevintä on kirjoittaa käskyt makrotiedostoksi (esimerkki 5.2.1), joka voidaan suorittaa viittaamalla sen nimeen. Makrotiedosto voi sisältää alimakroja, jotka määritellään lohkolla

```
BeginMacro <Makron nimi>
NDA makrokoodia
...
EndMacro
```

Makrolle voidaan välittää parametreja, esimerkiksi jos makrotiedosto on oma.cmd niin kutsuttaessa sitä tyyliin "oma.cmd boston crim" välittyy makrolle parametrit $1=boston ja $2=crim. NDA:n komennot ovat muunnoksia syöte (operandi) tietorakenteesta vaste-tietorakenteiksi. Tyypillinen komento, esimerkiksi ryvästelyalgoritmi ottaa syötteenään datajoukon ja palauttaa vasteenaan uuden datajoukon, joka koostuu ryppäiden prototyyppivektoreista. Vastetta voi edelleen käsitellä muilla NDA:n komennoilla. NDA:n komentokieli on periaatteessa universaali laskentamalli, joka perustuu osin lausekekieleen ja osin relaatioalgebraan. Relaatioalgrebra ja eräät komennot, kuten expr, while ja if laajentavat ohjelmiston ominaisuuksia valmiiden operaatioiden ja algoritmien tarjoamien toimintojen ulkopuolelle. NDA makrokielen aritmeettisella lausekekomennolla voidaan tehdä lasku- ja vertailutoimituksia. Lausekkeiden suorittamiskäsky on
`expr -fout <field> -expr <expr>; [-dout <dataout>] [-v]`,
missä `field` on kenttä johon tulos sijoitetaan, `expr` on laskettava lauseke, `dataout` on tietokehys johon tulos sijoitetaan, `v` lipulla tulostetaan tarkempaa tietoa annetun lausekkeen laskennasta. Laskennan operandeina ja tuloksena voi olla kokonaislukuja, liukulukuja tai merkkijonoja.

Makrokieli sisältää myös ehto- eli if-rakenteen, joka on muotoa
`if -exist <nameent> | -notexist <nameent> | -expr <expression>; -cmd <cmdline>:`,
missä `nameent` on nimiavaruuden rakenne, `expression` on looginen lauseke ja `cmdline` on suoritettava käsky. Määrittelyt `-exist`, `-notexist` ja `-expr` ovat toistensa poissulkevia. Käskyllä `if -exist <nameent> -cmd <cmdline>:` suoritetaan komento jos nimiavaruudesta on rakenne `nameent`. Edellisen negaatio on `if -notexist <nameent> -cmd <cmdline>:`. Käskyllä `if -expr <expression> -cmd <cmdline>:` suoritetaan komento jos lauseke on tosi.

If-rakenteen lisäksi makrokielelessä on myös toisto- eli while-rakenne
`while -expr <expression>; -cmd <cmdline>: [-loop <loopcmd>:]`,
missä `expression` on toistorakenteen suoritusehto, `cmdline` on toistossa suoritettava komento ja `loopcmd` on komento, joka suoritetaan jokaisen suoritetun komennon jälkeen.

Kokonaisuudessaan makrokieli on tehokas ja sillä voidaan suorittaa monimutkaisia toimintoketkuja. Lukijan halutessa tarkempia tietoja NDA:sta voi hän tutustua sen käyttöoppaaseen [21].

Esimerkki 5.2.1 (NDA:n makrokieli)

```
# Aineiston lataaminen ja esikäsittely
load boston.dat
prepro -d boston -dout predata -e -n
# Mallintaminen: SOM:in opettaminen ja
# aineiston luokittelu neuroneihin
somtr -d predata -sout som -l 5
somcl -d predata -s som -cout cld
# Mallin esittäminen: ryppäiden statistiikan laskeminen ja
# mallin visualisointi
clstat -d boston -c cld -dout stadata -all
mkgrp grp1 -s som
setgdat grp1 -d stadata
layer grp1 -l 3
bar grp1 -f stadata.crim_avg -co red
bar grp1 -f stadata.indus_avg -co green
show grp1 -hide
```

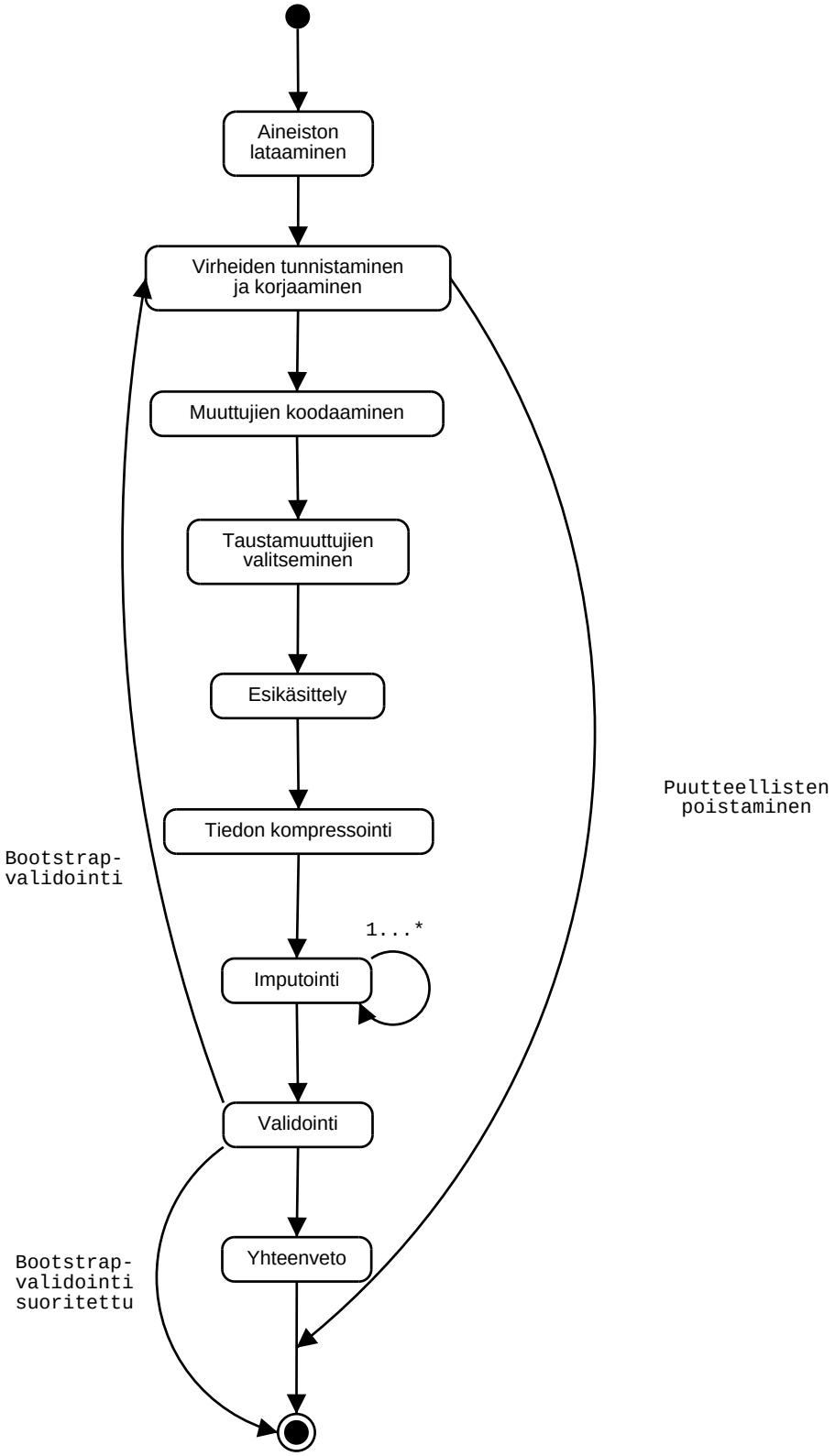
Esimerkki 5.2.1 lataa aluksi The Boston Housing Dataset havaintoaineiston, minkä jälkeen se esikäsittelee muuttujat. Esikäsittelyn jälkeen opetetaan 5-kerroksinen TS-SOM. Tämän jälkeen havaintoaineisto luokitellaan ja lasketaan statistiikka (stadata). Seuraavaksi luodaan grafiikka (grp1) ja valitaan TS-SOM kerros 3 näytettäväksi. Neuroneille määritellään grafiikkapylväiksi stadata.crim_avg (punainen) ja stadata.indus_avg (vihreä) eli rikollisuuden- ja teollisuuden keskiarvot. Lopulta näytetään grafiikka ilman työkalupaneelia.

5.2.2 Käyttäjälle suunnitellut toiminnot

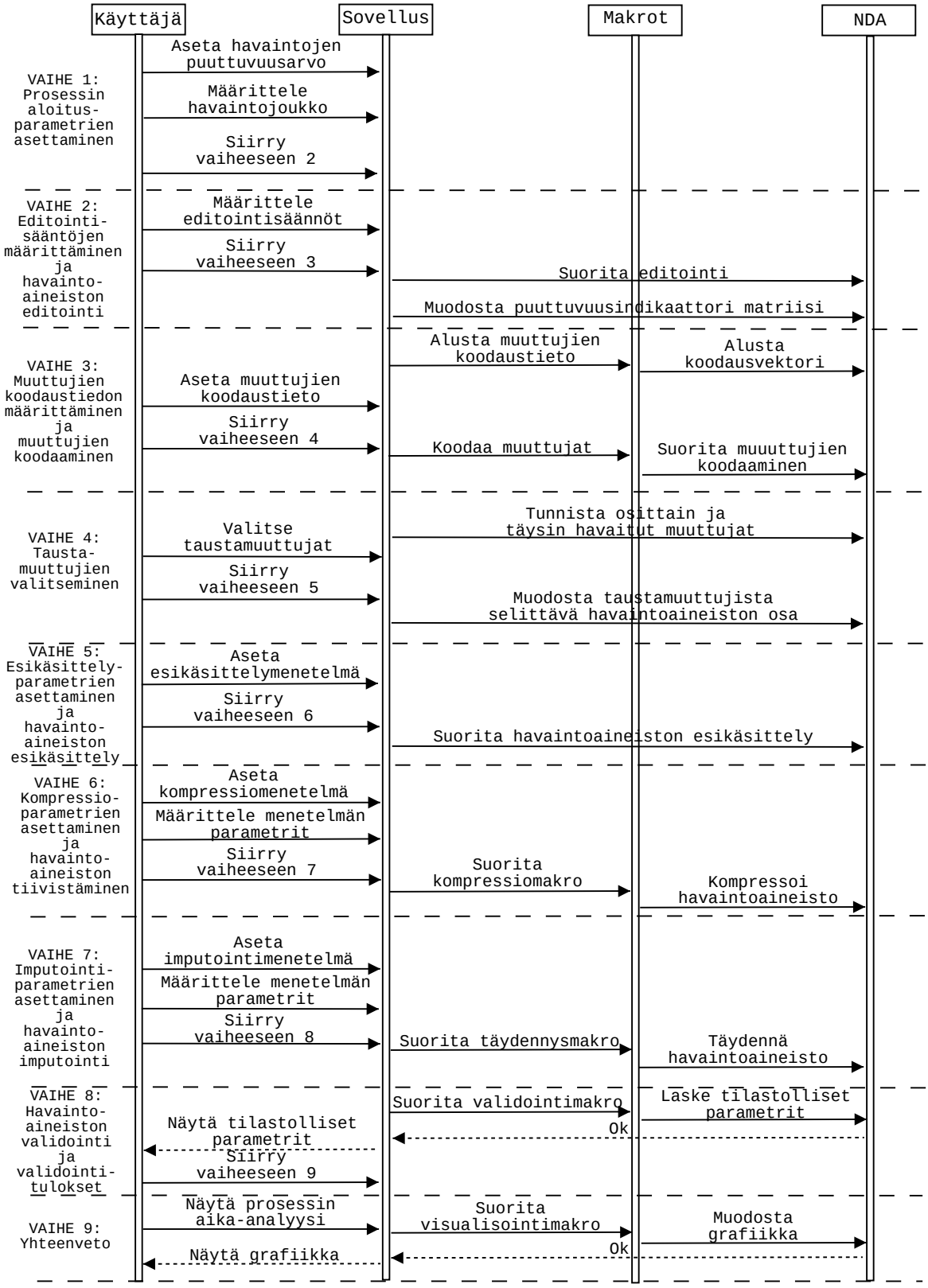
Tässä työssä toteutettu editointi- ja imputointisovellus on ohjelmistollinen toteutus tiedon tuotantoprosessille, joka koostuu yhdeksästä eri vaiheesta, jotka ovat

1. yhden tai useamman aineiston lataaminen,
2. loogisten virheiden tunnistaminen ja mahdollinen korjaaminen,
3. muuttujien koodaaminen imputointialgoritmien edellyttämään muotoon,
4. taustamuuttujien valitseminen,
5. esikäsittely,
6. tiedon kompressointi, eli tiivistäminen mallinnusmenetelmillä,
7. imputointi mallinnusmenetelmiä hyödyntäen,
8. tulosten validointi käyttäen uudelleen näytteistystekniikkoja ja
9. tulosten esittäminen.

Kuvassa 38 on havainnollistettu sovelluksen tiedon tuotantoprosessia yleisellä tasolla ja kuvassa 39 toteutetun ohjelmiston vaiheita. Käyttäjän on määriteltävä prosessin aloitusparametrit ja editointisäännöt ennen kuin editointi voidaan suorittaa. Editoinnin jälkeen luodaan puuttuvuusmatriisi. Käyttäjän on seuraavaksi määriteltävä muuttujien koodaustieto, minkä jälkeen muuttujat voidaan koodata. Ennen taustamuuttujien valitsemista taustamuuttujat tunnistetaan. Havaintoaineiston esikäsittely edellyttää esikäsittelyparametrien asettamista. Esikäsittelyn jälkeen käyttäjän on asetettava kompressointiparametrit, minkä jälkeen tieto voidaan kompressoida. Seuraavaksi käyttäjän on asetettava imputointiparametrit, minkä jälkeen tehdään imputointi. Tämän jälkeen tulokset validoidaan ja lopulta esitetään tulokset. Tiedon tuotantoprosessin aikana tehdään prosessianalyysi. Analyysissa mitataan aika seuraavista vaiheista: puuttuvuusmatriisin luominen, editointi, muuttujien koodaus, taustamuuttujien tunnistus, esikäsittely, tiedon kompressointi, imputointi ja validointi. Huom: prosessin aloitusparametrien asettamisvaihe sisältää aineiston lataamisvaiheen.



Kuva. 38: Vuokaavio tiedon tuotantoprosessista



Kuva. 39: Sekvenssidiagrammi ohjelmiston vaiheista

Prosessin aloitusparametrien asettamisvaiheessa käyttäjän on valittava käsiteltävä (puutteellinen ja/tai virheellinen) havaintoaineisto ja määrättävä tunnus, luku, jota käytetään puuttuvan havainnon muuttujan merkitsemiseen. Tämän lisäksi ohjelma tarjoaa mahdollisuuden määritellä validointitavan. Validointiin käytetään bootstrap-algoritmia, jossa voi varioida prosessin suorituskertojen lukumäärää. Lisäksi käyttäjän on mahdollista verrata editoinnin ja imputoinnin tuloksia oikeaan eli ehjään havaintoaineistoon, mikäli sellainen on saatavilla. Koska EurEdit-projektissa käytetyt havaintoaineistot sisälsivät puuttuvien havaintojen lisäksi luonnollista puuttuvuutta eli "ei-merkitystä/ei-saatavilla" olevia havaintoja, niin parametrien määrittelyssä voidaan asettaa myös näille oma luokka-arvo. Tämän lisäksi prosessiparametritiedoston nimen asettaminen, prosessiparametrien tallentaminen ja tallennettujen parametrien käyttäminen on mahdollista. Jos käyttäjä tallentaa prosessiparametrit ja valitsee seuraavalla ajokerralla prosessiparametrien käytön niin ohjelmisto valitsee automaattisesti jokaiseen näkymään edellisen kerran valinnat.

Editointisääntöjen määrittämisvaiheessa sääntöjen on oltava muokattavissa, tallennettavissa, tyhjättävissä ja ladattavissa. Sääntöihin voidaan määritellä editointiehto, tarkistusehto, editointitoiminnot ja imputointitoiminnot. Sääntöjen alussa määritellään editointiehdon totuusarvo virheelliselle havainnolle. Käyttäjällä on myös mahdollisuus määrittää kaikkien puutteellisten havaintojen poistaminen, mikä tehdään editoinnin jälkeen. Sääntöjen määrittämisen jälkeen ohjelmisto käy havaintoaineiston läpi sääntö kerrallaan ja luo virheindikaattorimatriisin.

Muuttujien koodaustiedon määrittämisvaiheessa osoitetaan mitkä muuttujat koodataan kategorisiksi muuttujiksi. Käyttäjän valitessa muuttujia, näytetään muuttujien tietotyypit (kokonaisluku, liukuluku tai merkkijono) ja havaintojen puutteellisuusindikaattori. Koodaustiedolla ohjelmisto muuttaa valitut kokonaisluvut omiksi muuttujiksiin siten, että jokaista havaintoaineistossa esiintyvää erilaista lukua kohden tuotetaan yksi uusi luokka muuttuja (kategoria). Samalla puuttuvalle muuttujan arvolle merkitään kaikki kategoriat puuttuviksi.

Taustamuuttujien valitsemisvaiheessa käyttäjälle näytetään osittain ja täysin havaitut muuttujat. Tämän jälkeen käyttäjä määrittelee mitä täysin havaittuja muuttujia käytetään taustatietämyksenä. Taustamuuttujista luodaan selittävä havaintoaineiston

osa, jota käytetään mm. MLP-verkon opetuksessa. Lukijan on syytä huomata, että tämä osa ohjelmistosta on edelleen kehityksen alla, sillä sopivan taustamuuttujien joukon valinta on tärkeä tehtävä kaikissa mallinnusmenetelmissä. Ohjelman nykyinen versio käyttää monissa malleissa kaikkia mahdollisia muuttujia mallinnustehtävään, mikä on yleisellä tasolla järjetöntä.

Esikäsittelyparametrien asettamisvaiheessa havaintojoukon muuttujat voidaan yhdenmukaistaa minimi-maksimi vaihteluvälien tai varianssien perusteella. Parametrien asettamisen jälkeen havaintoaineisto esikäsitellään valitulla tavalla. Esikäsittelyn merkitys korostuu erityisesti monimuuttujaisissa menetelmissä, joissa kustannusfunktion minimointi tehdään pienimmän neliösumman tai varianssin suhteen. Tällöin lopputulos on suuresti riippuvainen muuttujien vaihteluvälien, skaalan, pituudesta.

Tiedon kompressoinnin parametrien asettamisvaiheessa käyttäjä valitsee kompressiomenetelmän ja asettaa sen parametrit. Valittavissa olevat menetelmät riippuvat siitä onko havaintoaineistossa täysin havaittuja muuttujia vai ei. Jos niitä ei ole, niin valittavissa olevia vaihtoehtoja on vähemmän. Käytännössä kompressio tehdään joko ryvästely tai regressiomenetelmillä. Kompression voi myös jättää tekemättä, jolloin imputointimalli käyttää koko aineistoa jokaisen puuttuvan arvon täydentämiseen.

Imputointiparametrien asettamisvaiheessa valitaan käytetäänkö yksikkö- vai moni-imputointia. Moni-imputoinnille määritellään imputointien lukumäärä. Osa imputointimenetelmistä tuottaa aina saman täydennetyn havaintoaineiston samalla selitysaineistolla. Näillä menetelmillä ei ole "järkevää" tehdä moni-imputointia. Ohjelmisto tiedottaa käyttäjälle jos hän valitsee tällaisen menetelmän. Tämän jälkeen käyttäjä valitsee käytettävän imputointimenetelmän ja asettaa sen parametrit. Se mitä imputointimenetelmiä on valittavissa riippuu valitusta kompressiomenetelmästä. Jos havaintoaineistolle ei ole tehty kompressiota niin valittavissa imputointimenetelmien määrä on pienempi; useat menetelmät hyödyntävät kompressiossa mahdollisesti tehtävää klusterointia osana imputointimenetelmää. Imputointiparametrien asettamisen jälkeen havaintoaineistoon sovitetaan imputointimalli(t), jonka jälkeen suoritetaan havaintoaineiston täydentäminen, eli varsinainen imputointi.

Validointitulosvaiheessa käyttäjälle näytetään validointitulokset. Validointitulokset sisältävät kahdentyyppisiä tilastollisia tunnuslukuja. Ensimmäisen tyypin tunnusluvut mittaavat korjatun data-aineiston laadukkuutta suhteessa tunnettuun ja oikeanapidettyyn data-aineistoon, edellyttäen, että tämä on olemassa. Tunnuslukuja ovat keskivirhe, keskineliövirhe ja toisen momentin neliöjuuri (s. 77). Ne lasketaan sekä populaatioettä muuttujatasolla. Toisen tyypin tunnusluvut ovat arvioita syntyvistä eroavaisuuksista todellisen ja korjatun välillä, kun oikea havaintoaineisto ei ole tiedossa. Näitä tunnuslukuja ovat keskiarvo ja hajonta. Näistä tunnusluvuista lasketaan myös niiden erotus ja prosentuaalinen muutos. Laskenta tehdään muuttujittain täydennettävän ja täydennetyn havaintoaineiston välillä. Jokaisesta muuttujasta käytetään kaikki havaitut havainnot.

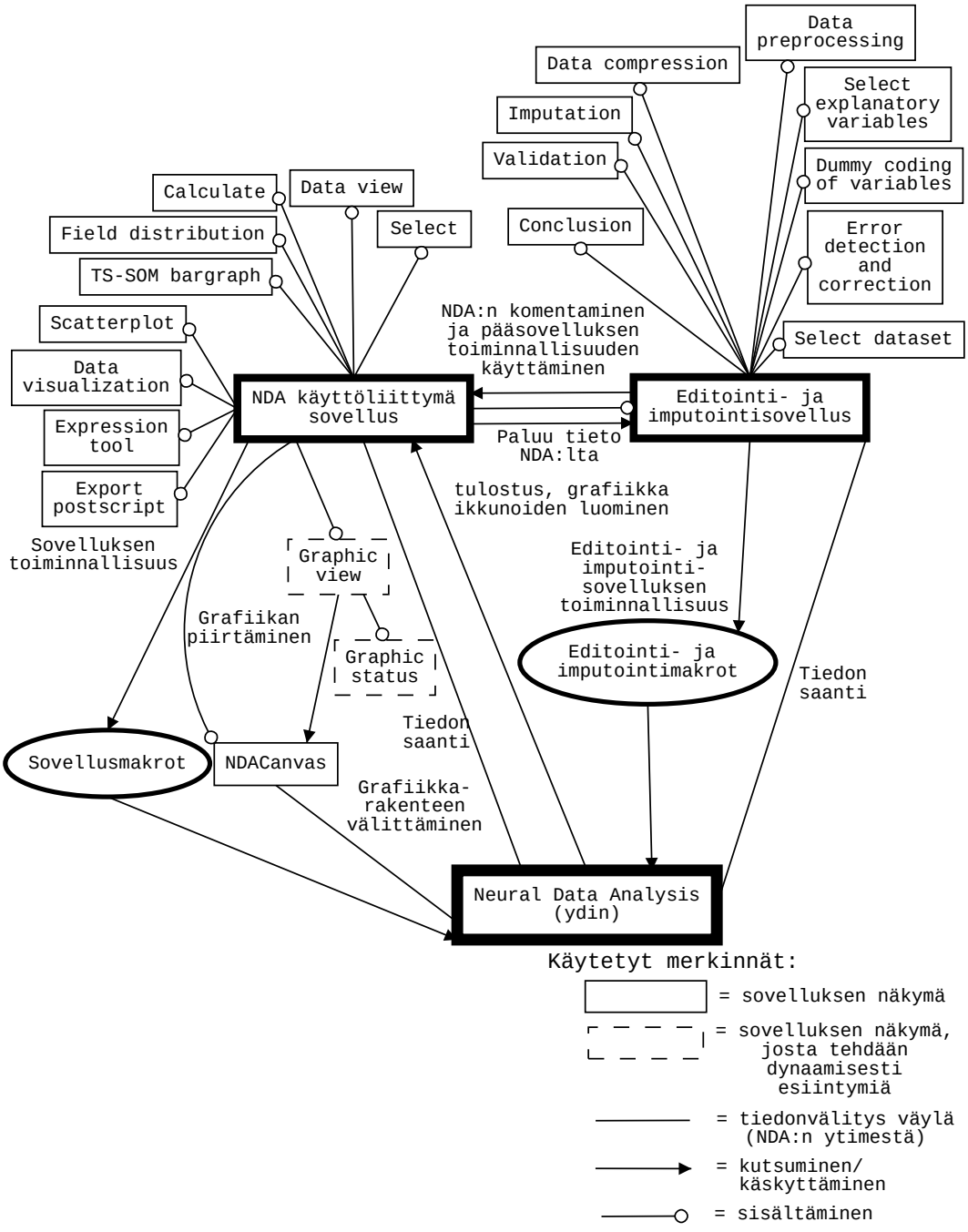
Yhteenvetovaiheessa käyttäjällä on mahdollisuus nähdä prosessin ajankulutus ja tallentaa tuotettu havaintojoukko. Ajankulutuksen avulla voidaan vertailla eri menetelmien laskennallista vaativuutta.

5.3 Toteutus

Toteutusvaiheessa ohjelmoitiin ohjelmistoarkkitehtuurin ja tavoitteiden mukainen sovellus. Ohjelmointikielinä olivat Object Pascal (ohjelmiston ja sen asennusohjelman tekemiseen), Ansi-C (NDA:n makrokielen käskyjen tekemiseen) ja NDA:n makrokieli (sovelluksen makroja varten). Sovelluksen laskennallinen toiminnallisuus tehtiin NDA:n makrokielellä, mikä mahdollisti käyttöliittymän ja imputoinnin toiminnallisuuden erottamisen toisistaan. Ohjelmiston tiedon tuotantoprosessi voidaan suorittaa lähes kokonaan ilman käyttöliittymää. Lähes siksi, että validointivaihe vaatii käyttöliittymän tukea, joskin sekin on toteutettu NDA:n makroilla, mutta sitä ei ehditty liittämään tiedon tuotantoprosessin suoritusmakroon. Sovelluksen tiedonvälitys ja rakenne on kuvattu kuvassa 40.

Editointi- ja imputointiohjelmisto rakennettiin kesän 2001 aikana tehdyn NDA sovelluksen päälle. NDA sovelluksella voidaan suorittaa makroja. Tämän lisäksi siihen on tehty työkaluja, joita ovat datan valinta, datan näkymä, laskenta, muuttujan jakauma,

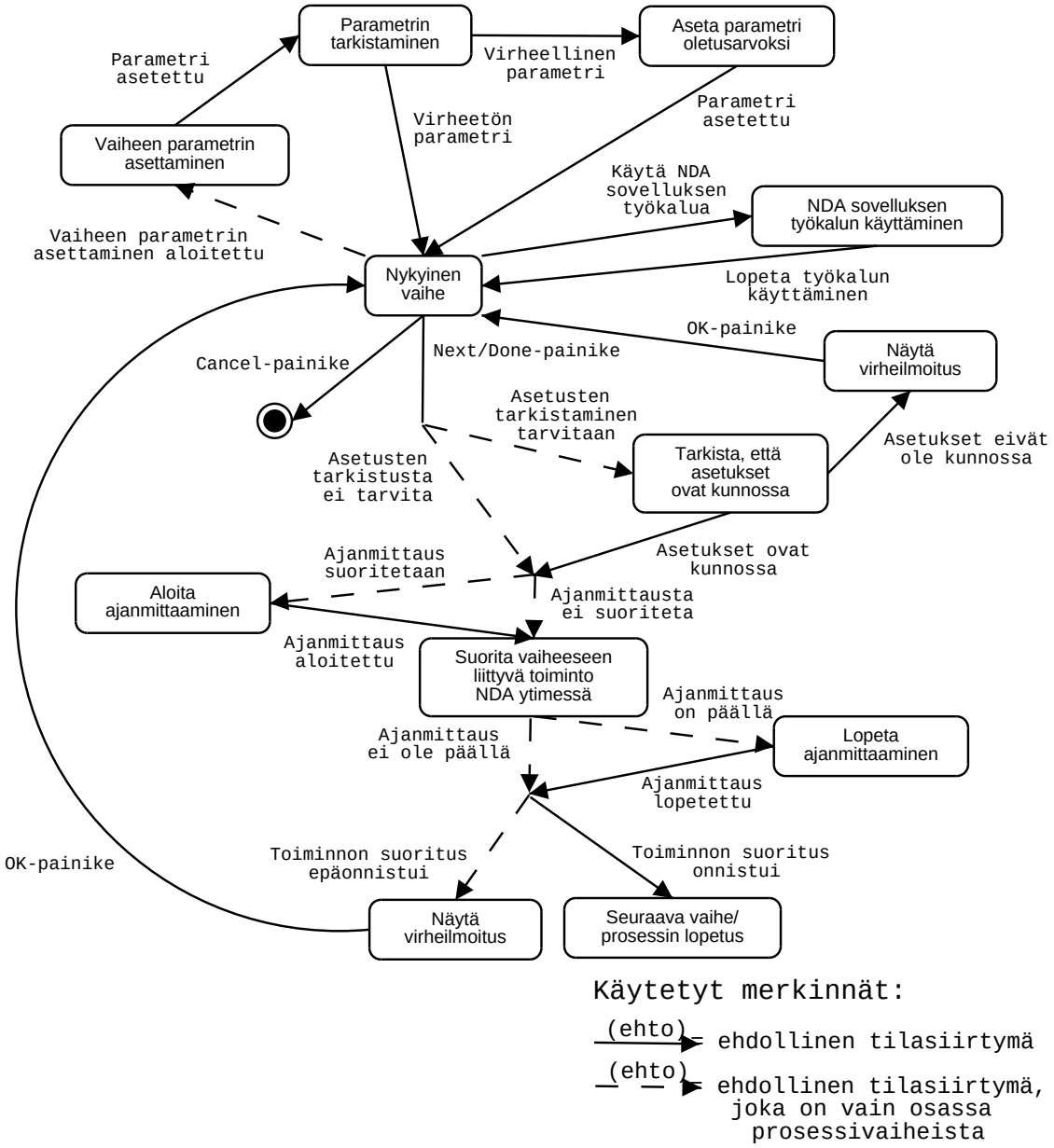
TS-SOM pylväsgrafiikka, hajontadiagrammi, datan visualisointi, expression työkalu ja PostScript:in tallentaminen. Nämä kaikki työkalut on tehty NDA:n avulla. Niiden tarkoitus on helpottaa usein toistuvia tehtäviä sekä mahdollistaa muutama perusvisualisointi. Tehty sovellus käyttää osaa näistä työkaluista hyödykseen.



Kuva. 40: Sovelluksen tiedonvälitys- ja rakennekaavio

Editointi- ja imputointiohjelmisto rakentuu yhdestätoista Object Pascal lähdetiedostosta ja 69:stä makrotiedostosta. Sovelluksen toiminnallisuus on jaettu eri tiedostoihin, jotta sen hallitseminen olisi helpompaa. Object Pascal lähdetiedostoja on yksi kullekin tiedon tuotantoprosessin vaiheelle ja editointia varten yksi aputiedosto. Tämän lisäksi on lähdetiedosto, joka sisältää yleisiä toimintoja, kuten graafisten komponenttien luomisen. Prosessin vaiheisiin on rakennettu mekanismi, jolla voidaan vaihtaa aktiivista vaihetta. Jos käyttäjä on valinnut Bootstrap-validoinnin ja prosessi on suoritettu kerran niin prosessin vaiheet vaihtavat automaattisesti aktiivista vaihetta. Tämän ansiosta käyttäjän ei tarvitse painella painikkeita. Bootstrap-validointi voi kestää kauan, minkä vuoksi on hyvä että tällöin prosessin suorituskerrat ovat automatisoituja. NDA käyttöliittymä sovelluksen pääikkunasta voidaan syöttää käskyjä. Tämä on mahdollista jopa tiedon tuotantoprosessin aikana. Prosessi nimeää NDA:n nimiavaruudessa käyttämät rakenteet DPP_-alkuisiksi. Täten on käyttäjän vastuulla että hän ei muokkaa (ja sotke) kyseisiä rakenteita. On suotavaa että käskyjä ei syötetä tiedon tuotantoprosessin aikana ellei käyttäjä tiedä tarkalleen mitä on tekemässä.

Kuvassa 40.1 on yleinen kaavio sovelluksen vaiheiden tilasiirtymistä. Sovellus on aina joko käyttöliittymän tilassa tai suorittamassa NDA makroa. Nykyinen- ja seuraava vaihe ovat jotkut yhdeksästä tiedon tuotantoprosessin vaiheista. Jos vaiheessa on asetettavia parametreja niin käyttäjä voi siirtyä niiden asettamiseen. Parametrin asettamisen jälkeen tarkistetaan onko sen arvo virheellinen. Jos se oli virheellinen niin se korjataan automaattisesti oletusarvoksi. Tämän jälkeen siirrytään odottamaan seuraavaa toimintoa. Editointi- ja imputointiohjelmiston rinnalla voidaan käyttää NDA sovelluksen työkaluja. Painettaessa Cancel-painiketta keskeytetään tiedon tuotantoprosessi. Tällöin prosessin tulokset tuhotaan. Next-painikkeen painamisen jälkeen tarkistetaan aloitus- ja editointivaiheissa parametrit. Jos ne eivät ole kunnossa niin käyttäjälle näytetään virheilmoitus, minkä jälkeen odotetaan seuraavaa toimintoa. Jos parametreissä ei ollut virheitä tai niitä ei ole tarkistettu, niin aloitetaan osassa vaiheita toimintoon kuluvan ajan mittaaminen. Sen jälkeen suoritetaan toimintomakro, minkä jälkeen lopetetaan ajan mittaaminen. Imputointivaiheessa tarkistetaan toiminnon onnistuminen. Jos toiminto ei onnistunut, niin käyttäjälle ilmoitetaan siitä ja palataan takaisin imputointivaiheen alkuun. Toiminnon onnistuessa siirrytään seuraavaan vaiheeseen tai lopetetaan prosessi.



Kuva. 40.1: Kaavio sovelluksen vaiheiden tilasiirtymistä

NDA-ohjelmistossa on ennestään jonkin verran puuttuvan datan käsittelymenetelmiä. Ohjelmistoa tehtäessä huomattiin, että ne eivät riittäneet, vaan työssä tarvitaan myös uusia komentoja. Tämän vuoksi NDA:han lisättiin 19 uutta makrokomentoa. Tehdyt käskyt jakautuvat seitsemään luokkaan, jotka ovat yleiset, parametritiedoston lukeminen, luokittelu, puuttuvan datan käsittely, editointi, moni-imputointi ja jälkikäsittely.

Yleiset

NDA:ssa ei ollut ennestään komentoa, joka palauttaa muuttujatyypin. Tehtäessä makroja huomattiin että kyseiselle komennolle on tarvetta, mm. tutkittaessa mitkä jatkuvina käsitellyt muuttujat ovat alunperin olleet kokonaislukuja. Komennon muoto on:

$$getfieldtype : f \rightarrow f_{\text{int}},$$

missä f on kenttä, jonka tyyppi tallennetaan kokonaislukukenttään f_{int}. Arvo 0 vastaa kokonaisluku-, 1 liukuluku- ja 2 merkkijonokenttää.

Verrattessa kahden havaintojoukon eroja on käytännöllistä laskea esimerkiksi havaintojoukkojen havaintojen keskimääräisen etäisyys

$$\frac{1}{n}\sum_{j=1}^{n}|\mathbf{x}_1(j) - \mathbf{x}_2(j)|,$$

missä n on havaintojen määrä ja $\{\mathbf{x}^1(j)\}_{j=1}^n$ sekä $\{\mathbf{x}_2(j)\}_{j=1}^n$ ovat havaintojoukot. Etäisyys lasketaan käskyllä

$$datadist : d_1 \times d_2 \rightarrow f_{\text{float}},$$

missä d_1 ja d_2 ovat havaintoaineistot joiden välillä etäisyys lasketaan. Etäisyys tallennetaan liukulukukenttään f_{float}.

Parametritiedoston lukeminen

Ohjelma on suunniteltu siten, että se voidaan suorittaa automaattisesti ilman käyttäjän asetuksia, tai käyttäen jotain aiempia asetusparametreja. Tämä edellyttää, että ohjelman ohjausparametrit voidaan tallentaa tekstitiedostoon ja lukea sieltä uutta käyttökertaa varten. Prosessin parametritiedosto koostuu rakenteista, jotka ovat muotoa:

```
[LOHKO]
PARAMETRI(1)=ARVO(1)
PARAMETRI(2)=ARVO(2)
...
PARAMETRI(N)=ARVO(N)
```

Lohkoja ovat havaintoaineiston valinta, editointi-, esikäsittely-, kompressio- ja imputointi. Niiden parametrien arvojen kenttiin siirtämistä varten tarvittiin kolme uutta käskyä. Parametrien arvot voivat olla kokonaislukuja, liukulukuja, merkkijonoja

tai binaariarvoja (true/false). Seuraavissa kolmessa parametrien arvojen kenttään siirtokäskyssä vaaditaan että data on prosessitiedoston parametrirakenteen määrityksen kaltainen. Huom: merkkijonoja, lukuunottamatta arvoja "false" ja "true", voidaan lukea vain käskyllä $getprofilestring$. Parametrin arvo voidaan siirtää kenttään käskyillä:

$$getprofilestring : d \times block \times parameter \rightarrow f_{\text{string}}$$
$$getprofileint : d \times block \times parameter \rightarrow f_{\text{int}}$$
$$getprofilefloat : d \times block \times parameter \rightarrow f_{\text{float}},$$

missä d on data, $block$ on etsittävä lohko, $parameter$ on etsittävä parametri. f_{string} on merkkijonokenttä, f_{int} kokonaislukukenttä ja f_{float} on liukulukukenttä johon parametrin arvo sijoitetaan.

Luokittelu

Tässä työssä käytetyt kompressio eli ryvästelymenetelmät tukevat puutteellisen tiedon käsittelyä vain osittain tai ei ollenkaan. Tämän vuoksi ryvästely on tehty pääsääntöisesti käyttäen niitä havaintoaineiston muuttujia ja havaintovektoreita, joiden osalla ei ollut puuttuvuutta. Käytännössä tämä tarkoittaa puutteellisten havaintovektorien hylkäämistä opetusvaiheessa. Varsinaisen mallinmuodostuksen jälkeen ryppäisiin on sijoitettava myös puutteelliset havaintovektorit. Tämä on mahdollista uuden *bmpcl*-komennon avulla

$$bmpcl : d_{\text{explanatory}} \times d_{\text{weights}'} \rightarrow c,$$

joka valitsee ryppääksi sen, jonka Euklidinen etäisyys havaittujen muuttujien suhteen on pienin havaintovektoriin nähden, rypäs k $= \arg\min_k \sum_{i=1}^{d} (x_i(j) - w_i(k))^2$. Painovektoreissa $d_{\text{weights}'}$ on selittävään dataan $d_{\text{explanatory}}$ valitut komponentit.

Teoria osuudessa esiteltiin ryväsimputoinnin ongelmatilanne, joka aiheutuu kun ryppäissä ei ole yhtään kokonaista havaintoa. Tällöin jokaiselle havainnolle on etsittävä lähin rypäs, jossa on tarpeeksi kokonaisia havaintoja. Etsiminen tehdään käskyllä

$$bmniv : d \times d_{\text{weights}} \times d_{\text{hits}} \times f_{\text{min hits}} \rightarrow f_{\text{int}},$$

missä d on ryppään havaintoaineisto, d_{weights} on painovektorit, d_{hits} on ryppääseen luokittuvien kokonaisten havaintojen lukumäärä, $f_{\text{min hits}}$ on etsintäkriteeri (ryppäässä

olevien kokonaisten havaintojen lukumäärä) ja f_{int} on rypäsindeksien havaintovektori. Näiden lisäksi operaatio tarvitsee arvon ilmaisemaan havainnon puuttuvuutta ja kriteerin, joka on kokonaisten havaintojen minimimäärä.

Puuttuvan datan käsittely

Puuttuvan datan käsittelyä varten tarvittiin muutama uusi käsky. Puuttuvuuslohkojen visualisointia varten tarvitaan puuttuvuusindikaattori data. Se voidaan muodostaa käskyllä

$$bmim : d \rightarrow d_{\text{mim}}.$$

Käsky merkitsee havaintoaineiston d puuttuvien havaintojen kohdalle d_{mim}:iin ykkösen ja havaittujen kohdalle nollan. d_{mim}:n voidaan opettaa TS-SOM rakenteelle, minkä jälkeen puuttuvuuslohkoja voidaan visualisoida pylväsgrafiikkana. Grafiikassa esitetään valittujen muuttujien keskimääräiset puuttuvuudet jokaisessa ryppäässä (kuva 44.3).

Puuttuvien havaintoaineiston arvojen korvaamista ennustetuilla arvoilla tarvitaan tehdyssä työssä imputointivaiheissa. Tällöin imputointimenetelmällä on laskettu "arviot" jokaista komponenttia varten. Näistä tarvitaan vain ne, joiden kohdalla alkuperäisessä havaintoaineistossa on puuttuvuutta. Korvaamisoperaatio tehdään käskyllä

$$fmd : d \times d_{\text{supplemental}} \rightarrow d_{\text{supplemented}}.$$

Puutteellisten muuttujien muuntamista (esim. logaritmimuunnos) tarvitaan sovelluksessa. Käskyllä fmv sijoitetaan muunnetut arvot puutteelliseen muuttujaan.

$$fmv : f \times f_{\text{transform}} \rightarrow f_{\text{transformed}}.$$

Käsky korvaa muuttujan f kaikki havaitut havainnot muuttujan $f_{\text{transform}}$ arvoilla.

Imputoitujen ja alkuperäisten havaintojen välisen keskimääräisen virheen laskemiskäsky on

$$mdatadist : d_{\text{imputed}} \times d_{\text{compare}} \times d_{\text{missing}} \rightarrow f_{\text{float}},$$

missä $d_{imputed}$ on täydennetty aineisto, d_{compare} on alkuperäinen aineisto, d_{missing} on puutteellinen aineisto ja f_{float} on muuttuja johon etäisyys lasketaan. Näiden lisäksi käsky tarvitsee arvon ilmaisemaan havaintojen puuttuvuutta. Etäisyys lasketaan kuten *datadist* käskyssä eli laskemalla keskiarvo havaintojen absoluuttisten erojen summasta. Etäisyys normeerataan vain puuttuvien havaintojen määrällä, kun taas *datadist* käskyssä se normeerataan kaikkien havaintojen määrällä.

Editointi

Ohjelmistoon tehtiin sääntöpohjainen editointi. Virhelohkojen visualisointia varten tarvittiin virheindikaattorimatriisi. Näiden tekemistä varten ohjelmoitiin käsky

$$editrules : d \times d_{\text{editrules}} \rightarrow < d_{\text{edited}}, d_{\text{eim}} >,$$

missä d on virheellinen havaintoaineisto, $d_{\text{editrules}}$ on editointisäännöt (kts. esimerkki 5.3.1), d_{edited} on editoitu havaintoaineisto ja d_{eim} on virheindikaattorimatriisi. Huom: käsky tekee editoinnin suoraan aineistoon d muistin säästämisen vuoksi. Käsky suorittaa sääntöpohjaisen editoinnin sääntö kerrallaan. Virheindikaattorimatriisi on rakentaaltaan täysin samanlainen kuin vastaava puuttuvuusmatriisi. Matriisissa yksi tarkoittaa virheellistä havaintoa ja nolla virheetöntä.

Moni-imputointi

Moni-imputoinnissa kahden havaintoaineiston yhteenlaskeminen tehdään käskyllä

$$aftf : d_1 \times d_2 \rightarrow d_{\text{added}}.$$

Käsky lisää havaintoaineistojen d_1 ja d_2 havainnot komponenteittain yhteen ja sijoittaa tuloksen d_{added}:iin. Käskyä voidaan käyttää useita kertoja jos halutaan yhdistää m kappaletta havaintoaineistoja. Käskystä tehtiin myös muunnelma

$$maftf : d_1 \times d_2 \times d_{\text{missing}} \rightarrow d_{\text{added}}.$$

Tämän avulla havaintoaineistot d_1 ja d_2 lasketaan yhteen vain d_{missing} aineiston puuttuvien havaintojen kohdalta ja tulos sijoitetaan d_{added}:iin. Havaitut arvot kopioidaan d_{missing}:istä d_{added}:iin.

Imputoitujen havaintoaineistojen yhdistettyjen arvojen keskiarvo lasketaan käskyllä

$$mfbs : d_{\text{added}} \rightarrow d_{\text{multiplied}}.$$

Käsky kertoo $aftf$:n luodun d_{added}:n havainnot annetulla vakiolla ja sijoittaa tuloksen d_{average}:iin. Jos vakiona käytetään $\frac{1}{m}$:ää niin tällöin saadaan keskiarvoaineisto, olettaen että m kappaletta havaintoja on aiemmin yhdistetty. Käskystä tehtiin puutteellisen aineiston käsittelyä varten versio

$$mmfbs : d_{\text{added}} \times d_{\text{missing}} \rightarrow d_{\text{multiplied}}.$$

Käsky kertoo vakiolla vain ne d_{added}:n havainnot, jotka ovat puutteellisia d_{missing}:ssä. Sitä on käytettävä jos d_{added} on luotu $maftf$-käskyllä.

Jälkikäsittely

Tiedon tuotantoprosessin esikäsittelyyn sisältyy kokonaislukuna esiteltyjen kategoristen muuttujien koodaus totuusarvoisiksi luokkamuuttujiksi siten, että jokaista luokkaa vastaa oma muuttuja. Editoinnin ja imputoinnin suorittamisen jälkeen tämä muuttuja on koodattava alkuperäisellä tavalla, mitä tarkoitusta varten on toteutettu $debin$-käsky

$$debin : d_{\text{original}} \times d_{\text{binarized}} \times f_{\text{binvec}} \rightarrow d_{\text{debinarized}}.$$

Käsky koodaa muuttujan f_{binvec} osoittamat $d_{\text{binarized}}$:n muuttujat alkuperäisellä tavalla, mihin käytetään avuksi d_{original}:n havaintoaineistoa. f_{binvec} on muuttujien koodausvektori. Tulos sijoitetaan $d_{\text{debinarized}}$:een.

Koska useassa imputointimenetelmässä käytetään mm. normaalijakautunutta satunnaistermiä niin lopulliset havaintoarvot oli järkevintä rajoittaa. Rajoittamista varten tehtiin $clamp$-käsky

$$clamp : d \times f_{\text{min}} \times f_{\text{max}} \rightarrow d_{\text{clamped}}.$$

Käsky leikkaa havaintoaineiston d alkuperäisen havaintoaineiston minimi-maksimi vaihteluväleille, jotka on tallennettu kenttiin f_{min} ja f_{max}. Leikattu havaintoaineisto sijoitetaan d_{clamped}:een. Käskyllä estetään minimi-maksimi vaihteluvälien ulkopuoleiset havaintopoikkeamat, joita imputointivaihe voi tehdä.

Usein reaalimaailman tai NDA:n käsittelemä kokonaislukumuotoinen informaatio on esitetty liukulukuformaatissa. Editoinnin ja imputoinnin jälkeisessä jälkikäsittelyssä tällainen esitysmuoto on muutettava alkuperäiseen muotoon. Muuttaminen tehdään käskyllä

$$cuttoint : f_{\text{float}} \rightarrow f_{\text{cut}}.$$

Käsky leikkaa liukulukukentän f_{float} kokonaislukukentäksi f_{cut}. Operaatio voidaan myös suorittaa NDA:n expression kielellä. Tehdyn käskyn tulkitseminen on nopeampaa kuin expression kielen vastaavan toiminnon.

Editointi- ja imputointiohjelmiston toiminta rakentuu NDA:n makrokielen varaan, joita komennetaan käyttöliittymästä. Tämän lisäksi ohjelma voidaan suorittaa automaattisesti ilman käyttöliittymää, jota varten kaikki toiminnallisuus on kirjoitettu $DoDPP$

makroon ja sen tarvitsemiin apumakroihin. Apumakrot jakautuvat neljään luokka, jotka ovat tiedon kompressio-, imputointi-, validointi- ja yleiset. Makrolista on liitteessä A sivulla 118.

Kompressiomakrot

Kompressiomakrot luovat valitulla algoritmilla havaintoaineiston jakauman painovektorit. Tämän jälkeen havaintoaineisto luokitellaan ryppäisiin ja lasketaan niihin luokittuvien kokonaisten havaintojen lukumäärä. Näiden lisäksi tehdään mahdollisesti grafiikkarakenne. TS-SOM-algoritmi osaa hyödyntää opetuksessa ja ryvästelyssä puutteellisia havaintoja. Sen makro on $DoTSSOMFullCompression$:

$$d_{\text{all}} \times d_{\text{fully observed}} \rightarrow < d_{\text{weights}}, d_{\text{hits}}, c_{\text{all}}, c_{\text{fully observed}}, (s_{\text{G}}) > .$$

Ryvästely voidaan tehdä myös hierarkkisilla-, K-Means ja Fuzzy C-Means ryvästelymenetelmillä, jotka osaavat hyödyntää vain täysin havaittuja havaintoja. TS-SOM-algoritmia voidaan myös käyttää hyödyntäen vain havaittua havaintoaineiston osaa. Menetelmiä varten ohjelmoitiin makrot $DoTSSOMCompression$, $DoHClustCompression$, $DoKMeansCompression$ ja $DoFCMCompression$. Niiden suorittamien toimintojen formaalimuoto on

$$d_{\text{all}} \times d_{\text{fully observed}} \times d_{\text{explanatory}} \rightarrow < d_{\text{weights}}, d_{\text{hits}}, c_{\text{all}}, c_{\text{fully observed}}, (s_{\text{G}}) > .$$

Makrot poikkeavat $DoTSSOMFullCompression$:ista, siten että ne luovat aluksi osaluokituksen $c_{\text{fully observed}}$, jonka ne täydentävät kokonaiseksi luokitukseksi c_{all}. Täydentäminen tehdään luokittelemalla puutteelliset havainnot ryppäisiin etsimälle jokaiselle havainnolle lähin rypäs. Selittävä havaintoaineisto $d_{\text{explanatory}}$ sisältää käyttäjän valitsemat täysin havaitut muuttujat. Ryppään etsiminen tehdään laskemalla euklidinen etäisyys $d_{\text{explanatory}}$:n ja d_{weights}:n välillä. K-Means- ja Fuzzy C-Means eivät luo grafiikkarakennetta s_{G}. Sovellukseen toteutettiin myös iteratiivinen menetelmä, jonka makro on

$$DoEMTSSOMCentroidImputation : d_{\text{all}} \rightarrow < d_{\text{imputed}}, s_{\text{G}} > .$$

Se käyttää TS-SOM-algoritmia ja keskipisteimputointia. Menetelmä luo grafiikkarakenteeseen s_{G} kaksi viivadiagrammia. Ensimmäinen diagrammi on kovarianssimatriisin alkioiden keskimääräinen muutos iteraatioiden välillä. Muutos lasketaan *datadist* käskyllä. Toinen diagrammi on puutteellisten havaintojen keskimääräinen muutos. Diagrammien avulla voidaan mahdollisesti arvioida algoritmin konvergoimista.

Imputointimakrot

Imputointia varten muodostettiin imputointimakrot. Klusteroimattoman havaintoaineiston makrot tarvitsevat havaintoaineiston, puuttuvuusarvon ja täydennetyn havaintoaineiston, jonka ne muodostavat imputoinnissa. Osa makroista tarvitsee myös havaintoaineiston täysin havaitun osan, jotta imputointia voidaan nopeuttaa.

Imputointimenetelmiä ovat keskiarvo, lähin naapuri, ennustaminen normaalijakaumalla, robusti keskiarvo, stokastinen keskiarvo, robusti stokastinen keskiarvo, satunnainen luovuttaja, MLP monimuuttujaregressio ja MLP regressio muuttujittain. MLP:n opetusalgoritmeja on seitsemän, joista jokaista varten on oma makro. Imputointimakrojen suorittaman toiminnallisuus on $d \rightarrow d_{\text{imputed}}$, lukuunottamatta satunnaista luovuttajaa ja MLP regressiomenetelmiä. Ensimmäisen toiminto on $d \times d_{\text{fully observed}} \rightarrow d_{\text{imputed}}$ ja jälkimmäisen

$$d_{\text{fully observed}} \times d_{\text{explanatory}} \times d_{\text{partially observed}} \times d \times d_{\text{MLP parameters}} \rightarrow < d_{\text{imputed}}, d_{\text{MLP errors}} >,$$

missä $d_{\text{fully observed}}$:ssa on täysin havaitut havainnot, $d_{\text{explanatory}}$ on selittävät muuttujat, $d_{\text{partially observed}}$ on ennustettavat muuttujat, d on havaintoaineisto, $d_{\text{MLP parameters}}$ on MLP menetelmän parametrit ja $d_{\text{MLP errors}}$ opetusvirheet. MLP parametreista tärkeimmät ovat mm. iteraatioiden ja piilokerrosten määrä sekä piilokerrosten neuronien lukumäärä. Kaksi jälkimmäistä parametria määrittelevät MLP-verkon kompleksisuuden.

Koska sovellus tukee tiedon kompressointia, oli makroista tehtävä versiot jotka osaavat käyttää havaintoaineiston luokitusta ryppäisiin. Ryväsimputointimakrot tarvitsevat havaintoaineiston, prototyypit (d_{weights}), luokituksen (c), puuttuvuusarvon ja täydennetyn havaintoaineiston, jonka ne muodostavat imputoinnissa. Makrojen toiminnallisuus formaalisti on $d \times d_{\text{weights}} \times c \rightarrow d_{\text{imputed}}$. Poikkeuksena ovat ennustaminen normaalijakaumalla, satunnainen luovuttaja ja MLP-menetelmät. Ennustaminen normaalijakaumalla ei tarvitse painovektoreita d_{weights}. Satunnainen luovuttaja tarvitsee ne sekä täysin havaitun osan $d_{\text{fully observed}}$. MLP regressiomakrojen toiminnallisuus on

$$c \times d_{\text{weights}} \times d_{\text{fully observed}} \times d_{\text{explanatory}} \times d_{\text{partially observed}} \times d \times d_{\text{MLP parameters}} \rightarrow$$
$$< d_{\text{imputed}}, d_{\text{MLP errors}} >.$$

Moni-imputointia varten oli muodostettava omat versiot makroista. Sitä varten tarvi-

taan myös imputointien lukumäärä. Moni-imputointi on mahdollista vain menetelmillä, jotka ovat ennustaminen normaalijakaumalla, satunnainen luovuttaja, stokastinen keskiarvo ja robusti stokastinen keskiarvo. Moni-imputointimakrot suorittavat imputoinnin m kertaa ja yhdistävät imputoidut havaintoaineistot lopulta yhdeksi havaintoaineistoksi.

Validointimakrot

Kahden havaintoaineiston muuttujien tilastollisten tunnuslukujen laskeminen tehdään makrolla

$$CalculateStatisticalParameters2 : d_{\text{imputed}} \times d_{\text{compare}} \rightarrow d_{\text{statistical parameters}}.$$

Makro laskee muuttujakohtaisesti tunnusluvut $d_{\text{statistical parameters}}$:iin, joita ovat keskiarvo ja hajonta. Makroon voidaan lisätä mitä tahansa NDA:lla laskettavia muuttujien tunnuslukuja. On huomattava että d_{imputed}:ssa ja d_{compare}:ssa on havaintoja muuttujakohtaisesti eri määrät. Tämä johtuu siitä, että d_{compare}:ssa on puutteellisia havaintoja, jotka ohitetaan tunnuslukuja laskettaessa. Tunnusluvuista lasketaan myös havaintoaineistojen välillä erotus ja prosentuaalinen ero. Näin voidaan tutkia prosessin vaikutusta tunnuslukuihin ja nähdä niiden aliarviointi ja/tai yliarviointi.

Toinen käytännällinen validointimenetelmä on laskea havaintojen keskimääräinen virhe. Tämä on mahdollista, jos oikeat arvot ovat saatavilla. Tätä varten ohjelmoitiin makro

$$CalculateStatisticalParameters3 : d_{\text{imputed}} \times d_{\text{compare}} \rightarrow d_{\text{statistical parameters}}.$$

Keskimääräinen virhe lasketaan kolmella eri tunnusluvulla, jotka ovat keskivirhe (engl. *Mean Error*):

$$\text{ME} = \frac{1}{n_{\text{missing}}} \sum_{j=1}^{n} |x_1(j) - x_2(j)|,$$

keskineliövirhe (engl. *Mean Square Error*):

$$MSE = \frac{1}{n_{\text{missing}}} \sum_{j=1}^{n} (x_1(j) - x_2(j))^2,$$

ja toisen momentin neliöjuuri (engl. *Root Mean Square Error*):

$$RMSE = \frac{1}{n_{\text{missing}}} \sqrt{\sum_{j=1}^{n} (x_1(j) - x_2(j))^2},$$

ME:n, MSE:n ja $RMSE$:n laskukaavoissa n_{missing} on puuttuvien havaintojen lukumäärä. Virheluvut lasketaan populaatio- ja muuttujatasolla.

Jos tiedontuotanto prosessi validoidaan niin tunnusluvuille lasketaan vaihteluvälit. Laskenta tehdään makrolla

$$CalculateFinalStatisticalParameters : d_{\text{statistical parameters}} \rightarrow d_{\text{statistical parameters}'}.$$

Makro laskee tilastollisista tunnusluvuista $d_{\text{statistical parameters}}$ keskiarvon ja vaihteluvälin (hajonnan). Tulos sijoitetaan $d_{\text{statistical parameters}'}$:iin.

Yleiset makrot

Yleisten makrojen tarkoituksena oli vähentää muiden makrojen sisältöä korvaamalla usein tehtävät asiat niillä. Makrolla $DummyCodeDiscreteVariables$ diskreetit muuttujat koodataan luokkamuuttujiksi.

$$DummyCodeDiscreteVariables : d \times f_{\text{codvec}} \rightarrow d_{\text{binarized}},$$

missä d on koodattava data, f_{codvec} on koodausvektori, $d_{\text{binarized}}$:een sijoitetaan koodattu data. Koodausvektorissa on 1, jos kyseinen muuttuja halutaan koodata ja arvo 0 muulloin. Koodausvektorin alustamista varten tehtiin $InitDummyVariableCoding$ makro. Se lataa koodaustiedon tiedostosta tai alustaa sen nollaksi.

Havaintoaineiston täysin havaittu, osittain havaittu ja selittävä data valitaan makrolla

$SeparateFullPartialExplanatory$:
$d \rightarrow < d_{\text{fully observed}} \times d_{\text{partially observed}} \times d_{\text{explanatory}} \times d_{\text{mim}}.$

Makro luo myös puuttuvuusindikaattorimatriisin d_{mim}. Käskyä tarvitaan $DoDPP$ makrossa valitsemaan mm. selittävät ja selitettävät muuttujat.

Täysin havaitun havaintoaineiston luokitus täydennetään koko havaintoaineiston luokitukseksi makrolla

$$BuildCompleteClassification : d_{\text{fully observed}} \times d_{\text{weights}} \times c_{\text{fully observed}} \times d_{\text{explanatory}} \rightarrow c_{\text{all}}.$$

Makro laajentaa luokituksen $c_{\text{fully observed}}$ luokitukseksi c_{all}. Yhdistämällä puutteelliset havainnot $d_{\text{explanatory}}$:n avulla d_{weights}:iin. $d_{\text{explanatory}}$:ssa on käyttäjän valitsemat se-

littävät muuttujat. Sovelluksessa makroa tarvitaan kun käytetään jakauman mallinnusmenetelmiä, jotka eivät osaa hyödyntää puuttuvia havaintoja.

MLP regressiomenetelmiä varten tehtiin neljä apumakroa. MLP-verkon alustusmenetelmäparametri asetetaan makrolla *BuildMLPInitializationParameter*, rakenne *BuildMLPNetParameter*:lla ja aktivaatiofunktiot *BuildMLPTypesParameter*:lla. MLP-verkon opetus-, testi- ja validointijoukko luodaan makrolla

$$CreateTrainTestValidationSets : d \rightarrow < d_{\text{train}}, d_{\text{test}}, d_{\text{validation}} > .$$

Näiden joukkojen luomista varten on kaksi makroa, jotka ovat *create_bootstrap_group* ja *create_bootstrap_group*2. Niiden toiminto on

$$create\_bootstrap\_group2 : d \rightarrow d_{\text{subsampled}}.$$

Niistä ensimmäinen luo annetun havaintoaineiston d kokoisen aineiston $d_{\text{subsampled}}$. Se käyttää avukseen makroa *random_subsample*, joka poimii satunnaisesti yhden havainnon aineistosta. Makrolle *create_bootstrap_group*2 voidaan määrittää satunnaisen otoksen koko. Makroja käytetään tehdessä imputointi satunnaisella luovuttajamenetelmällä.

Havaintoaineiston d puutteelliset havainnot poistetaan makrolla

$$DoListwiseDeletion : d \rightarrow d_{\text{listwise deleted}}.$$

Jäljelle jääneet havainnot laitetaan $d_{\text{listwise deleted}}$:een. Sovelluksessa voidaan suorittaa tämä operaatio editoinnin jälkeen.

Havaintoaineiston jakauman mallia tehdessä makro *SetupCompressionData* valitsee mallin datan. Malli voidaan tehdä sekä esikäsitellystä että esikäsittelemättömästä havaintoaineistosta. TS-SOM-algoritmin yhden SOM kerroksen valitsemista varten tehtiin makro *PickHighestResolutionSOM*:

$$s_{\text{SOM}} \times d_{\text{weights all}} \times c_{\text{all}} \rightarrow < d_{\text{weights selected}} \times c_{\text{selected}} > .$$

Makro poimii hierarkkisesta luokituksesta halutun tarkkuisen osaluokituksen, joka luokittelee koko havaintoaineiston. Mallintamisen jälkeen lasketaan ryppäiden keskipisteet makrolla

$$CalculateClustersMeans : d \times c \rightarrow d_{\text{centroids}}.$$

Imputoinnissa satunnainen termi tehdään makrolla

$$CreateNormallyDistributedNoise : d \times f_{\text{variance}} \rightarrow d_{\text{noise}}.$$

Makro luo havaintoaineiston d kokoisen "häiriöllisen" aineiston d_{noise}. Häiriön suuruuden määrittelee varianssikenttä f_{variance}. Häiriö tehdään luomalla normaalijakautunutta havaintoaineistoa, jonka keskiarvo on nolla ja varianssi f_{variance}. Luominen tehdään satunnaisesti näytteistämällä normaalijakaumasta havaintoja. Poimintatodennäköisyyksinä käytetään normaalijakauman tiheysfunktiota.

Jälkikäsittelyssä jatkuvina muuttujina käsitellyt kokonaislukumuuttujat leikataan kokonaislukumuuttujiksi. Leikkaaminen tehdään makrolla

$$CutToInt : d_{\text{original}} \times d_{\text{imputed}} \rightarrow d_{\text{cut}}.$$

Makro muuttaa kokonaisluvuiksi d_{imputed}:n kaikki muuttujat, jotka ovat d_{original}:ssa kokonaislukuja.

Makro $VerifyCompleteData$ varmistaa, että kahden aineiston dimensiot vastaavat eli formaalimmin

$$VerifyCompleteData : d_1 \times d_2 \rightarrow f_{\text{status}}.$$

Kenttään f_{status} laitetaan arvo 0, jos aineistot d_1 ja d_2 eivät ole samankokoiset. Muussa tapauksessa kenttään sijoitetaan arvo 1. Makroa käytetään varmistamaan että virheellinen ja/tai puutteellinen havaintoaineisto ja validointiaineisto ovat dimensioiltaan samanlaiset.

$SelectCompleteRecords$ makro valitsee havaintoaineistosta täysin havaitun osan.

$$SelectCompleteRecords : d \rightarrow d_{\text{fully observed}}.$$

$MergeDataframes$ liittää havaintoaineistoon kaikki toisen aineiston muuttujat, jotka siitä puuttuvat.

$$MergeDataframes : d_1 \times d_2 \rightarrow d_{\text{merged}}.$$

Validointivaiheessa näytettävä aika-analyysin grafiikka tehdään makrolla $ShowTimeConsumption$, joka luo grafiikan s_{G}. Grafiikassa näytetään pylväinä tiedon tuotantoprosessi vaiheisiin kuluneet ajat.

5.3.1 Ongelmat

Seuraavat asiat aiheuttivat ongelmia sovellusta kehittäessä

- editointisääntöjen tallennusformaatti,
- ohjelmakoodin siirrettävyys,
- komponenttien nopeus,
- komponenttien resurssienkulutus,
- käyttöjärjestelmien väliset erot,
- NDA-kirjaston osittainen testaamattomuus,
- Dynaaminen NDA-kirjasto Linuxissa ja
- liukulukujen desimaalierotin.

Editointisääntöjen tallennusformaatti
Editointisäännöt olivat alunperin Microsoft Excelin CSV-tallennusformaatista, joka on tekstitiedostoformaatti, jossa muuttujat (sarakkeet) ovat sarkaimin ja havainnot (rivit) rivinvaihdoin eroteltuina. Säännöt olivat myös osittain sanallisessa muodossa vaikka muuttujien arvot olivat numeroita. Editointisäännöt muutettiin CSV-formaatista NDA:n expression-muotoon, jotta niiden käyttäminen sovelluksessa olisi nopeampaa. Ongelmia aiheuttivat merkkijonot joiden numeeriset arvot olivat selitetty toisessa tiedostossa. Toisen ongelman muodosti UK ABI -aineiston säännöt, jotka erosivat rakenteeltaan UK Household SARS:in säännöistä. ABI aineistossa oli ehto, jonka on oltava tosi editointisäännön tarkistamista varten. Loppujen lopuksi tallennusformaatin aiheuttamat ongelmat eivät olleet vakavia.

Ohjelmakoodin siirrettävyys
Koska Delphi 5, Delphi 6 ja Kylix eivät ole lähdekoodiltaan 100% yhteensopivia, niin siirrettävyyden takia oli tehtävä kääntäjäkohtaisia määrityksiä lähdetiedostoihin. Delphi 5:n lähdekoodi ei ole aivan Kylix yhteensopivaa, mutta pienin kääntäjäkohtaisin määritelmin ohjelma on mahdollista kääntää. Delphi 5:n ja Kylixin välisiä eroja ovat mm.:

- kirjastojen nimet (esim. Delphi 5:StdCtrls - Kylix:QStdCtrls),
- resurssitiedostojen nimeämisessä ja käyttämisessä (Delphi 5:*.dfm - Kylix:*.xfm),
- komponenttien ominaisuuksissa (esim. Delphi 5:n TForm:in alignment-ominaisuus puuttuu Kylixistä),
- komponenttien tapahtumakäsittelijät ovat eri nimisiä, toiminnaltaan eroavia tai kokonaan puuttuvia (esim. Delphi 5:n TStringGrid-komponentin OnSetEditText-käsittelijä puuttuu Kylixistä) ja
- näppäimien nimet (esim. Delphi 5:VK_RETURN - Kylix:Key_Return).

Kylixissä ja Delphi 6:ssa on CLX-kirjasto, joka on kehitetty siirrettävyyttä varten. Delphi 6:n sovellukset eivät kuitenkaan ole lähdekoodiltaan aivan 100% Kylix yhteensopivia koska mm. merkkijonojen tyypityksessä ja tietueiden pakkaamisessa on pieniä eroja.

Komponenttien nopeus

Komponentit eivät ole kovinkaan nopeita sillä ne periytyvät usein monesta oliosta ja niiden luominen voi kestää kohtuullisen kauan. Nopeusongelma ei tule vastaan vähäisellä määrällä komponentteja, mutta suurella määrällä sen huomaa. Kylixin CLX-sovelluksessa merkkijonojen lisääminen mm. yhdistelmälistaan (engl. *combobox*) on hidasta. Delphi 6:lla käännetty normaali versio sovelluksesta on selvästi nopeampi kuin sillä käännetty CLX versio. Ero näkyy mm. graafisten näkymien alustuksessa eli graafisissa komponeneteissa. CLX versio kuluttaa myös enemmän muistia kuin normaali versio.

Komponenttien resurssienkulutus

Komponentit kuluttavat paljon resursseja. Esimerkiksi merkkijonoja taulukkona näyttävän TStringGrid-komponentin solut vievät huomattavasti muistia. Testattaessa sitä n. 18 miljoonalla liukulukusolulla muistin varaaminen keskeytyi yhteen gigatavuun koska koneen virtuaalimuisti loppui. Havaintoaineisto ei vienyt kuin n. 70 megatavua. Tämän takia oli toteutettava 1000-rivinen näkymä havaintoaineistoon, joka vie vähemmän muistia.

Käyttöjärjestelmien väliset erot

Ohjelmaa tehtäessä havaittiin, että Windows-käyttöjärjestelmissä systeemiresurssien (GDI-grafiikkarakenne, jne) hallinta on erilaista. Sovellusta ajettaessa Windows 98- tai Windows Millennium Edition -käyttöjärjestelmissä systeemiresurssit loppuvat melkein heti kun yksikin NDA:n grafiikkaikkuna näytetään. Windows 95- ja Windows NT -käyttöjärjestelmiä ei ollut saatavilla testattavaksi. On erittäin todennäköistä että ainakin Windows 95:ssa on sama ongelma kuin Windows 98- ja Windows ME -käyttöjärjestelmissä, koska pohjana on sama ydin. Ainakin vanhemmissa Windowseissa GDI-resursseja on kiinteä määrä, mikä aiheuttaa ongelmia koska sovellus tarvitsee niitä paljon luodessaan dynaamisesti komponentteja.

Linux- ja Windows-käyttöjärjestelmissä tiedostojen poluissa käytettävä hakemistojen erotinmerkki on erilainen. Linuxissa se on / ja Windowissa \. Sovelluksen joissakin kohdissa joudutaan muuttamaan polun esitysmuotoa, jotta viittaus toimisi käytettävässä käyttöjärjestelmässä. On myös huomattava että Linuxissa isot ja pienet kirjaimet tiedostojen nimissä tarkoittavat eri tiedostoa, kun Windowissa ne tarkoittavat samaa tiedostoa.

NDA-kirjaston osittainen testaamattomuus

NDA-kirjasto on valtava ohjelmistopaketti ja siinä on joitakin osittain testattuja käskyjä. Sovellus keskeytyy jos NDA-kirjaston jokin käsky kaatuu. Ratkaisu ongelmaan on joko käyttää jotain toista menetelmää, jos sellainen on olemassa, tai sitten korjata kyseinen käsky. Suurimmaksi osaksi NDA:n käskyt toimivat erittäin hyvin, koska ne ovat hyvin testattuja.

Dynaaminen NDA-kirjasto Linuxissa

Linuxissa dynaaminen NDA-kirjasto on shared object (.so) -muodossa koska Kylixillä ei voi linkittää staattista kirjastoa sovellukseen. Staattinen kirjasto on kokonaisuus joka liitetään sovelluksen binaarikoodiin sen linkityksen yhteydessä. Dynaaminen kirjasto, joka on erillinen tiedosto, luetaan muistiin kun sovellus käynnistetään. Kirjastosta on viittauksia ulkopuolisiin kirjastoihin, kuten libc:hen, joka on osa C-kielen peruskirjastoa. Viittaus on tiettyyn kirjaston versioon, joten sovelluksen siirrettävyys Linuxin eri levitysversioiden välillä on hankalaa. Yksi mahdollisuus olisi kääntää NDA-kirjastosta binaariversioita eri Linux-järjestelmille. Ongelmaa ei tutkittu tarkemmin vaan NDA-

kirjastosta käännettiin versio testikoneella, mikä varmisti että kirjastoviittaukset toimivat siinä.

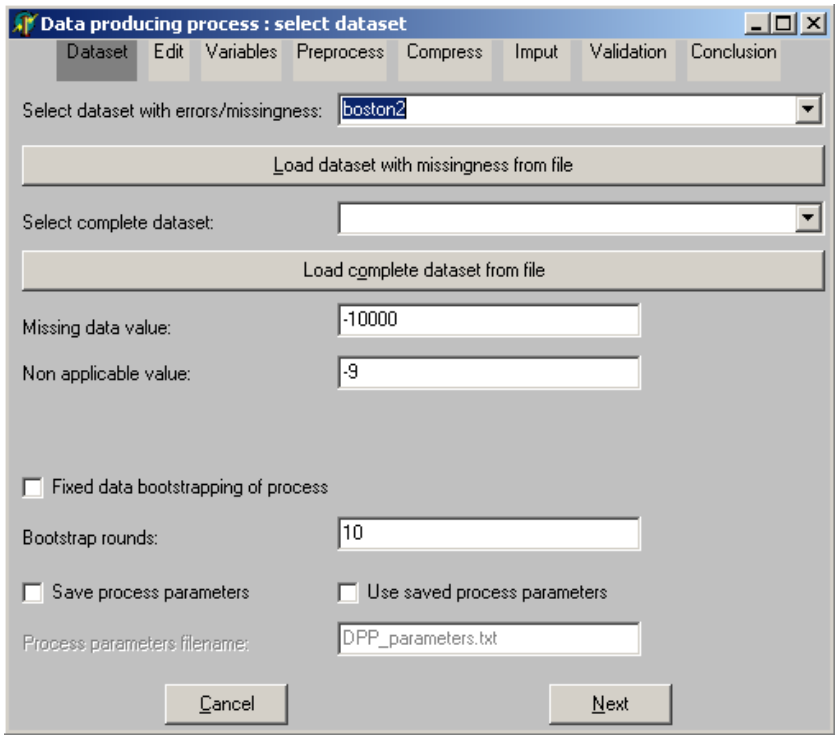
Liukulukujen desimaalierotin
Windows-käyttöjärjestelmässä liukulukujen desimaalierotin aiheutti ongelmia koska oletuksena Windows käyttää erottimena maavyöhykkeen (engl. *regional settings*) määrittelemää erotinta. Ongelma havaittiin muunnettaessa liukulukuja merkkijonoiksi ja päinvastoin. Ongelman ratkaisuksi liukulukujen desimaalierotin asetetaan sovelluksen alussa pisteeksi.

5.3.2 Sovellus

Seuraavissa kappaleissa kuvaillaan sovelluksen toiminnallisuus (kts. 5.2.1). Mukana on myös kuvia joiden toivotaan helpottavan sovelluksen hahmottamista lukijalle. Kappaleissa **taustamuuttujien valitseminen** ja **validointi** käy ilmi, että sovellukseen lisättiin toimintoja, joita ei ollut alkuperäisessä suunnitelmassa. Aliotsikot etenevät sovelluksen käyttöliittymän vaiheiden mukaisesti.

Prosessin aloitusparametrien asettaminen
Ensimmäisessä vaiheessa (kuva 41) käyttäjän on valittava käsiteltävä puutteellinen ja/tai virheellinen havaintojoukko, josta halutaan tuottaa ehjä havaintoaineisto. Ohjelma mahdollistaa havaintoaineiston lukemisen sisään NDA:n tiedostoformaatissa (tiedostopääte .dat). Käyttäjän on määriteltävä havaintojen puuttuvuutta ilmaisema arvo, joka oletuksena on -10000. Käyttäjä voi myös määritellä luonnollista puuttuvuutta ilmaiseman arvon, joka oletuksena on -9. Arvoa käytetään visualisoitaessa muuttujien jakaumia validointivaiheessa. Nämä arvot poistetaan tällöin, jotta jakaumat eivät mahdollisesti piikity siihen. Täydellinen havaintoaineisto voidaan myös valita tässä vaiheessa, jos sellainen on saatavilla. Prosessin tuottamia tuloksia voidaan sen avulla verrata validointivaiheessa alkuperäisiin (kts. alikappale 5.4.8). Prosessi voidaan myös validoida Bootstrap-menetelmällä ja vakiohavaintoaineistolla, mikä tarkoittaa, että validointi mittaa prosessin menetelmän tuottamaa vaihtelua. Vaihtelu voi syntyä virheiden tunnistamis- ja korjaamis-, tiedon kompressointi- ja imputointivaiheessa. Prosessin parametrit voidaan määritellä tallennattavaksi tiedostoon ja tallennettuja parametreja

voidaan käyttää jälkeenpäin. Oletuksena parametrit tallennetaan DPP_parameters.txt tiedostoon. Painettaessa Next-painiketta ohjelma tarkistaa, että havaintoaineisto on määritelty. Jos täydellinen havaintoaineisto on asetettu niin tarkistetaan, että sen dimensio vastaa käsiteltävän havaintoaineiston dimensiota. Cancel-painikkeella voidaan keskeyttää prosessi. Valitsemalla tallennettujen prosessiparametrien käyttämisen sovellus lataa tiedoston määrittelemän käsiteltävän havaintoaineiston. Tämän lisäksi ladataan validointiaineisto jos sellainen on määritelty. Lopuksi asetetaan prosessiparametrit tiedostosta. Jos käyttäjä asettaa prosessiparametrien tallentamisen niin tällöin ennen kuin siirrytään seuraavaan vaiheeseen tallennetaan aloitusparametrit prosessitiedostoon. Prosessiparametrien tallentaminen ja tallennettujen parametrien käyttäminen ovat toistensa poissulkevia. Oletusarvona on, että prosessiparametreja ei tallenneta eikä tallennettuja parametreja käytetä.



Kuva. 41: Prosessin aloitusparametrien asettaminen

Editointisääntöjen määrittäminen

Ensin virheet tunnistetaan editointisääntöjen avulla, minkä jälkeen ne korjataan. Virheet voidaan merkitä imputoitaviksi tai korvata eksplisiittisella arvolla. Editointisäännöt ladataan automaattisesti .er-päätteisestä tiedostosta, jos sellainen on olemassa. Tämän

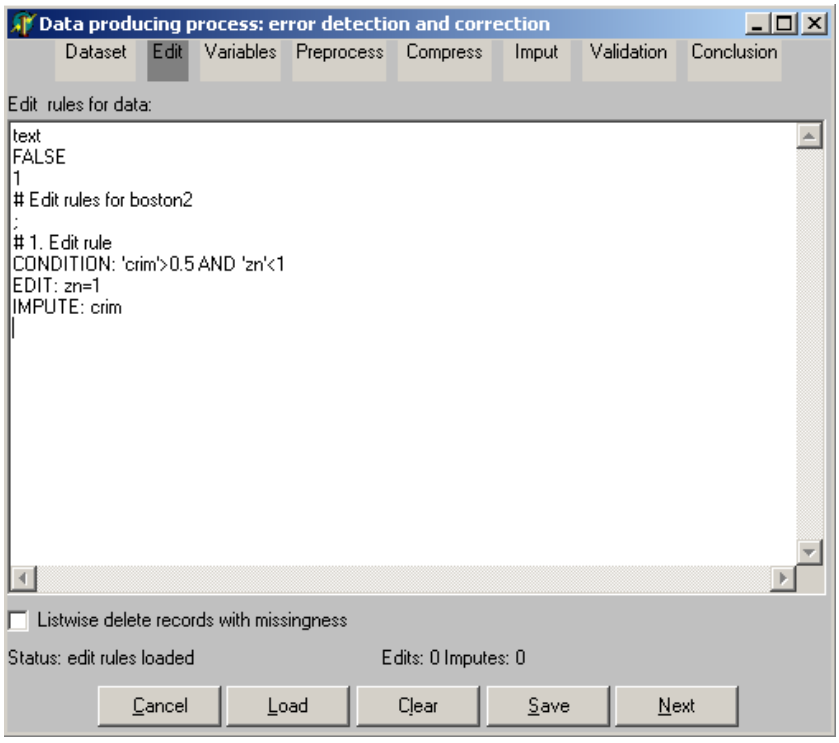
ohittaa prosessiparametritiedoston määritys, jos tallennettuja arvoja käytetään. Sovelluksen editointisäännöt ovat määrittelyltään esimerkin 5.3.1 mukaiset. Sääntöjä voidaan myös muokata ohjelmalla (kuva 42). Sääntöjen tyhjentäminen tapahtuu Clear-painikkeella. Load-painikkeella ladataan editointisäännöt tiedostosta. Näkymässä olevien editointisääntöjen tiedostoon tallentaminen tehdään Save-painikkeella. Prosessin keskeyttämien tapahtuu Cancel-painikkeella. Next-painikkeen painamisen jälkeen suoritetaan virheiden havaitseminen ja korjaaminen säännöillä. Tämän lisäksi luodaan virheindikaattorimatriisi ja mitataan editointiin kuluva aika. Editoinnin jälkeen luodaan puuttuvuusindikaattorimatriisi ja sen tekemiseen kuluva aika mitataan. Lopulta siirrytään seuraavaan vaiheeseen. Jos puutteellisten poistaminen oli valittu, niin editoinnin jälkeen puutteelliset havainnot poistetaan havaintoaineistosta. Tällöin siirrytään suoraan prosessin loppuun.

Esimerkki 5.3.1 (Esimerkki säännöstö)

```
text
FALSE
1
# Editointisäännöt virheelliselle Boston Household -aineistolle
;
# Tarkistetaan onko muuttujien crim arvo positiivinen, zn:n arvo
# pienempi kuin yksi ja indus erisuuri kuin nolla. Jos näin on niin
# havainto on virheellinen. Tällöin zn:lle editoidaan arvo 6 ja
# crim muuttuja merkitään imputoitavaksi.
CONDITION: 'crim' > 0.0 AND 'zn' < 1
CHECK: 'indus' != 0
EDIT: zn=6
IMPUTE: crim
```

Editointisäännöissä toinen rivi määrittelee totuusarvon, jolla editointi tehdään. Editointisäännön tulee sisältää ehto (`CONDITION`), editointitoiminnot (`EDIT`) ja imputointitoiminnot (`IMPUTE`). Näiden lisäksi siinä voi olla myös tarkistusehto (`CHECK`). `CONDITION` määrittelee ehdon, jonka totuusarvon täytyy olla sääntöjen 2. rivin totuusarvon negaatio. Jos näin ei ole, niin havainto ei läpäise sääntöä ja vaatii mahdolliset editointi- ja imputointitoiminnot. Ehto saa olla mikä tahansa mielivaltainen NDA:n lausekekie-

len tukema looginen lauseke, kunhan tulos on 0/1-arvoinen eli epätosi tai tosi. `CHECK` määrittelee tarkistusehdon, jonka on oltava tosi jokaiselle editoitavalla havainnolle. Tämän avulla voidaan tehdä mm. editointisäännöt yhdistetylle lyhyen ja pitkän kyselyn havaintoaineistolle. Tällöin jonkun muuttujan on määriteltävä mihin kyselyyn havainnot kuuluvat. `EDIT` määrittelee editointitoiminnan, joka kuvaa muuttujiin tehtävät korjaukset. Korjauslausekkeessa muuttujille voidaan määritellä explisiittisesti yksi arvo, joka annetaan sille korjauksen tapahtuessa. Vaihtoehtoisesti voidaan määritellä joukko arvoja korjaukselle, joista poimitaan satunnaisesti yksi. Yleinen muoto editointirivil-le on: `EDIT: kenttä1=arvo1|...|arvoK1,...,kenttäN=arvo1|...|arvoKN`. Jos korjausarvojen poimintatodennäköisyyksiä halutaan painottaa on korjausarvot syötettävä todennäköisyyksien suhteessa eli esimerkiksi: `EDIT: status=1|1|1|2|2` määrittelee 60%:n todennäköisyyden 1:lle ja 40%:n todennäköisyyden 2:lle. `IMPUTE` määrittelee kentät, jotka halutaan imputoida kun editointisääntö astuu voimaan. Imputointirivin on oltava muotoa: `IMPUTE: kenttä1,...,kenttäN`.

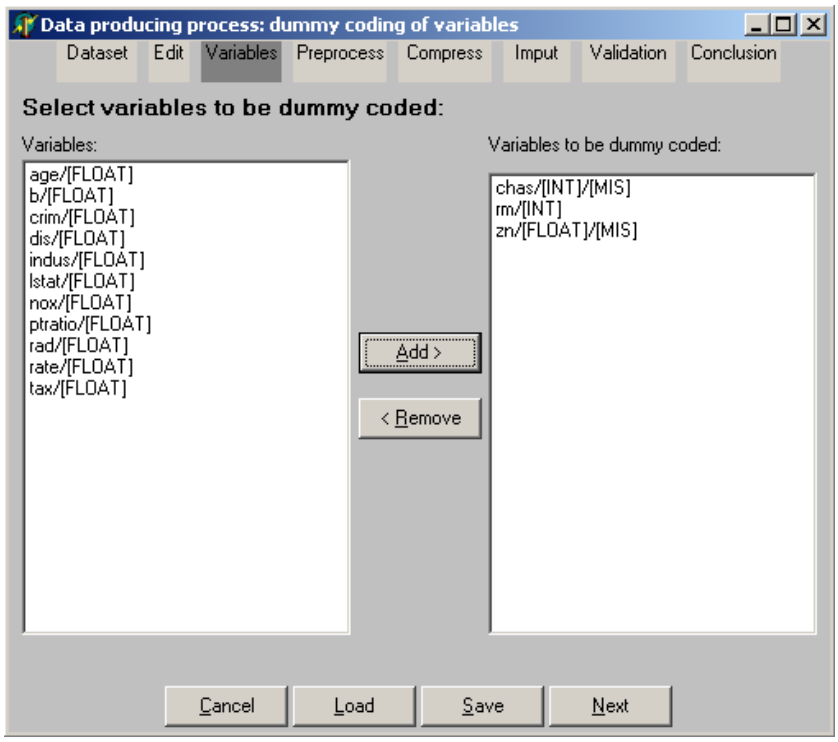


Kuva. 42: Editointisääntöjen määrittäminen

Muuttujien koodaustiedon määrittäminen

Diskreetit muuttujat, jotka eivät ole jatkuvia (kuten sukupuoli, ammatti, jne), täytyy

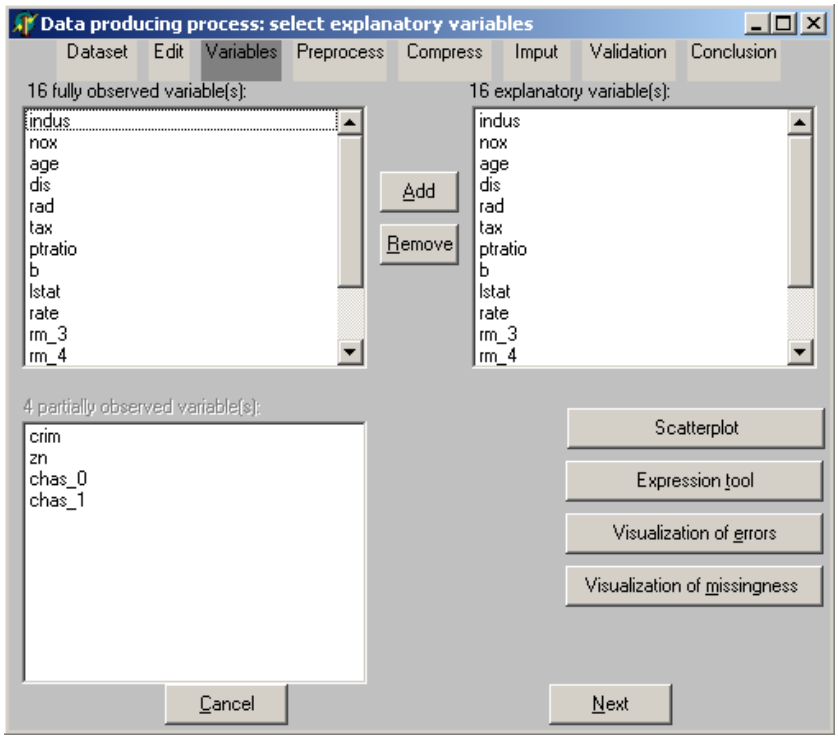
koodata siten ettei niiden luokkien välille muodostu koodauksesta johtuvia riippuvuuksia. Yksinkertaisin tapa on koodata niiden luokat luokkamuuttujiksi. Ilman kyseistä käsittelyä lopputuloksiin voi tulla mahdottomia arvoja, joita alkuperäisessä havaintoaineistossa ei ollut. Diskreetit "jatkuvat" muuttujat, kuten ikä, voidaan usein prosessoida ilman koodaamista. Tällöin ohjelma käsittelee ne jatkuvina muuttujina ja lopputulos katkaistaan kokonaisluvuksi. Koodaustiedon määrittämisvaiheessa käyttäjä valitsee koodattavat muuttujat listalta (kuva 43). Listalla näkyy muuttujatyyppi ([INT],[FLOAT] tai [STRING]) ja mahdollinen puuttuvuusindikaattori (teksti [MIS]). Add-painikkeella lisätään muuttuja koodattavaksi. Vain kokonaislukumuuttujia voidaan koodata. Remove-painikkeella poistetaan muuttuja koodattavien listalta. Ohjelma lataa koodaustiedon automaattisesti .cod-päätteisestä tiedostosta, jos sellainen on olemassa. Tämän ohitse menee prosessiparametritiedostossa mahdollisesti määritelty koodaustiedosto. Load-painikkeella voidaan koodaustieto ladata mistä tahansa tiedostosta. Vastaavasti Save-painikkeella voidaan määritelty koodaustieto tallentaa haluttuun tiedostoon. Next-painikkeen painamisen jälkeen tehdään valittujen muuttujien koodaaminen. Tähän kuluva aika mitataan. Tämän jälkeen tutkitaan havaintoaineiston puuttuvuus. Jos puuttuvuutta ei ole, niin siitä ilmoitetaan käyttäjälle, minkä jälkeen prosessi keskeytetään.



Kuva. 43: Muuttujien koodaustiedon määrittäminen

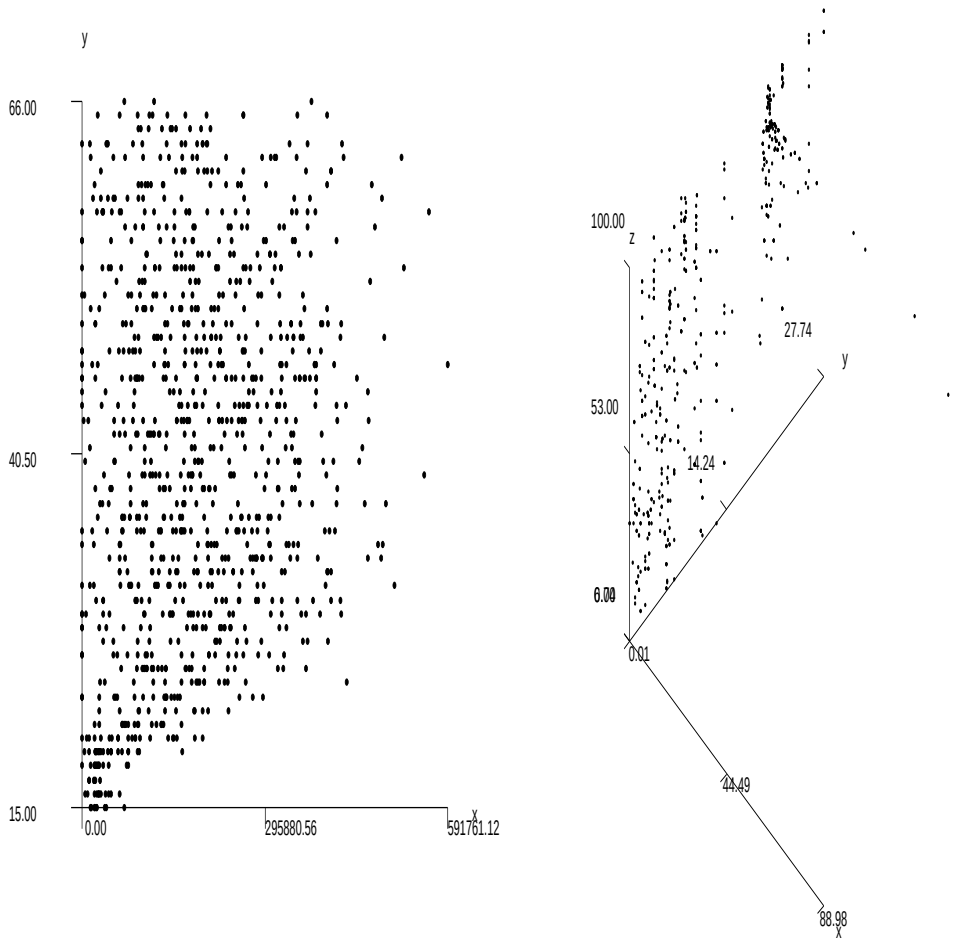
Taustamuuttujien valitseminen

Seuraavaksi on valittava taustamuuttujat (kuva 44), joilla yhdistetään puutteellinen havaintoaineisto havaittuun tai selitetään puuttuvia arvoja MLP regressiossa. Muuttujien puuttuvuuden tunnistamiseen kuluva aika mitataan. Add-painikkeella lisätään muuttuja selittäväksi muuttujaksi. Remove-painikkeella poistetaan valittu muuttuja selittävien muuttujien listalta. Merkkijonokentät siirretään suoraan täydennettyyn havaintoaineistoon, oletuksena on että merkkijonokenttää vastaa kokonais- tai liukulukukenttä. Osittain havaitut muuttujat otetaan automaattisesti mukaan havaintoaineiston jakauman malliin (TS-SOM, K-Means, jne). Jos havaintoaineistossa ei ole yhtään täysin havaittua muuttujaa niin MLP regressiomenetelmät eivät ole käytettävissä. K-Means-, Fuzzy C-Means-, kokonaan havaitusta havaintoaineiston osasta rakennettu TS-SOM- ja hierarkiset klusterointimenetelmät eivät ole myöskään tällöin käytettävissä. Cancel-painikkeella voidaan prosessi keskeyttää. Seuraavaan vaiheeseen siirtyminen tapahtuu Next-painikkeella. Huom: selittävien muuttujien listaa ei tallenneta prosessiparametritiedostoon.



Kuva. 44: Taustamuuttujien valitseminen

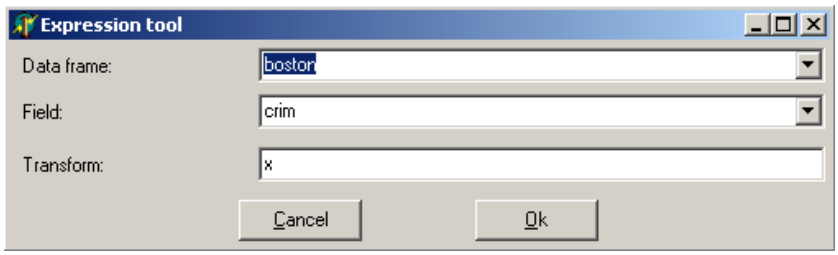
Tässä vaiheessa lisättiin mahdollisuus muuttujien hajontadiagrammin katselemiseen, muuttujan asteikon muuttamiseen sekä puuttuvuuden ja virheiden visualisoimiseen. Nämä toiminnot tehdään Scatterplot-, Expression tool-, Visualization of errors- ja Visualization of missingness -painikkeilla. Näitä ei ollut sovelluksen suunnitelmassa. Hajontadiagrammin muuttujien valitsemisen lisäksi voidaan tutkia muuttujan asteikon muuttamista, esimerkiksi logaritmiseksi. Jos havaintoaineistossa on havaintoja yli 1000 kappaletta, niin niistä poimitaan satunnaisesti 1000, jotta diagrammin visualisointi sujuu nopeasti. Muuttujien hajontadiagrammeja voidaan visualisoida kuvan 44.1 mukaisesti.



a) b)

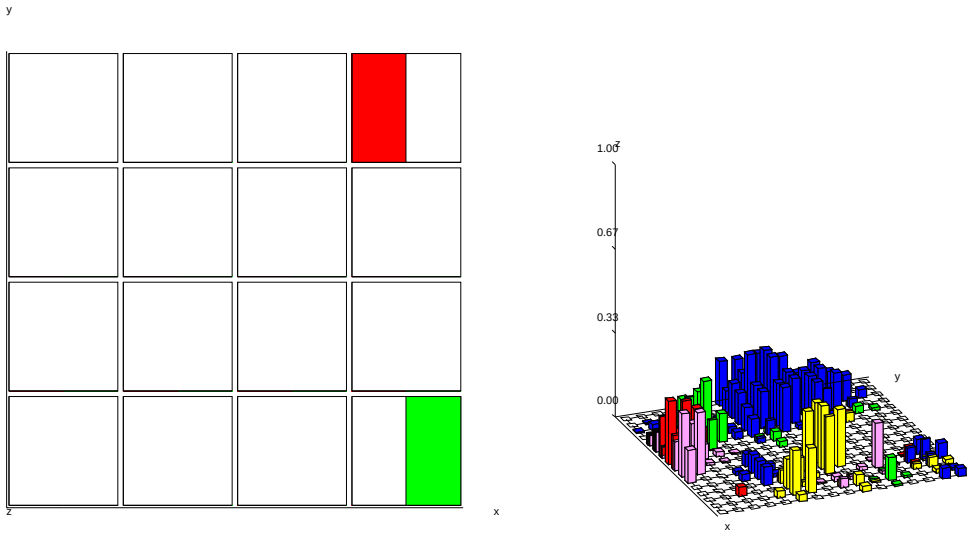
Kuva. 44.1: Hajontadiagrammit: a) 2-ulotteinen; b) 3-ulotteinen

2-ulotteisella hajontadiagrammilla voidaan tutkia kahden muuttujan ja 3-ulotteisella kolmen muuttujan välistä korrelaatiota. Näiden lisäksi sovelluksessa on mahdollisuus myös visualisoida 1-ulotteinen hajontadiagrammi. Se on hyödyllinen mm. tutkittaessa muuttujan asteikkoa. Hajontadiagrammien tutkimisen jälkeen saattaa olla tarpeen muuttaa jonkun muuttujan asteikkoa. Tämä voidaan tehdä expression-työkalulla (kuva 44.2).



Kuva. 44.2: Expression-työkalu

Havaintoaineistossa olleita virheitä, olettaen että editointisäännöt olivat aiemmin käytössä, ja puuttuvuuksia voidaan visualisoida 2- tai 3-ulotteisesti kuvan 44.3 mukaisesti. Näin voidaan tutkia yhteisjakaumia ja mahdollisesti tunnistaa virhe- ja puuttuvuuslohkoja.

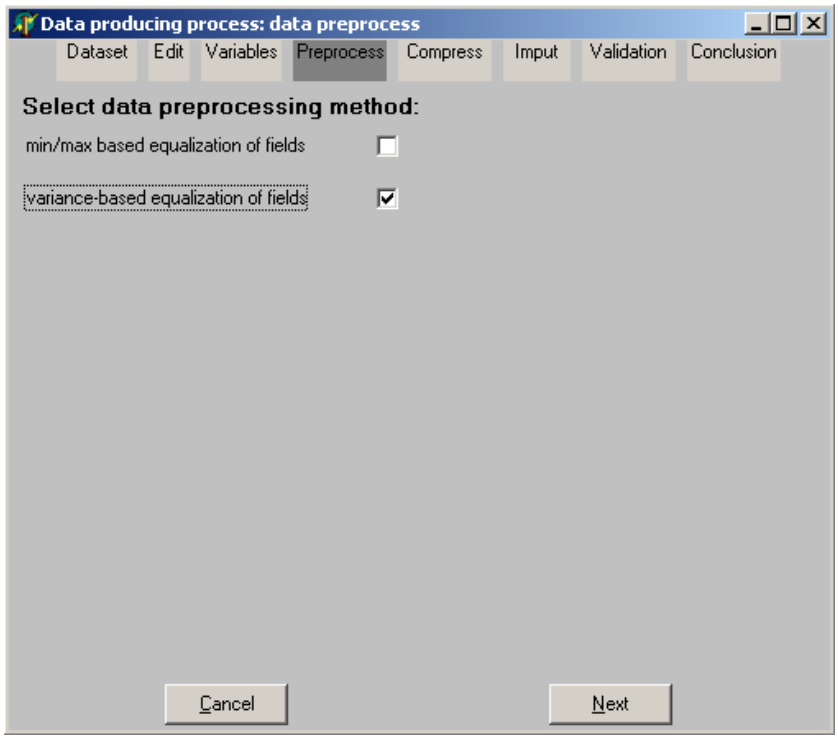


a) b)

Kuva. 44.3: Virheiden ja puuttuvuuksien visualisointi:
a) 2-ulotteinen visualisointi; b) 3-ulotteinen visualisointi

Esikäsittelyparametrien asettaminen

Esikäsittelyssä havaintoaineiston muuttujat on mahdollista yhdenmukaistaa (kuva 45). Muuttujien asteikkojen yhdenmukaistaminen tehdään niiden minimi- ja maksimiarvojen avulla (min/max based equalization of fields -painike). Vaihtoehtoisesti muuttujat voidaan yhdenmukaistaa varianssien avulla (variance-based equalization of fields -painike). Yhdenmukaistaminen on tarpeen sillä kaikki käytetyt menetelmät ovat varianssipohjaisia. Jos esikäsittelyä ei tehdä niin muuttuja, jolla on suurin varianssi, dominoi kompressiovaiheessa tehtävää jakauman mallintamisprosessia. Oletusasetuksena on että esikäsittelyä ei tehdä. Cancel-painike keskeyttää prosessin. Next-painikkeen painamisen jälkeen suoritetaan mahdollisesti esikäsittely. Esikäsittelyyn kuluva aika mitaan. Esikäsittelyparametrit tallennetaan prosessiparametritiedostoon, jos käyttäjä on niin määritellyt.



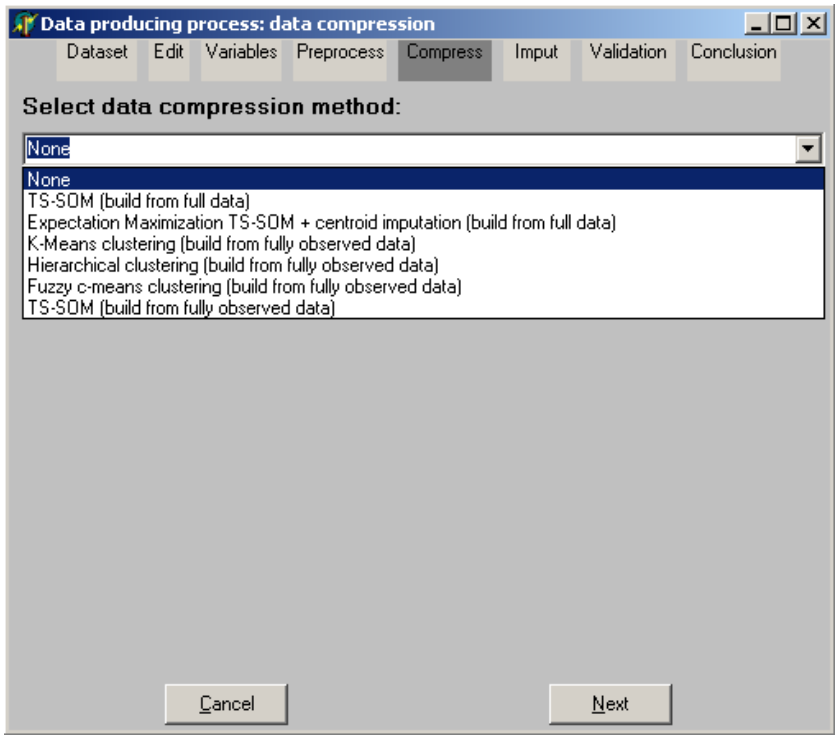
Kuva. 45: Esikäsittelyparametrien asettaminen

Kompressiovaiheen parametrien asettaminen

Tiedon kompressiolla voidaan mallintaa havaintoaineiston jakaumaa, mikä nopeuttaa usein imputointia. Jakauman mallinnuksessa muodostetaan sen prototyyppejä (neuroneita tai ryppäitä). Käyttäjän on valittava menetelmä ja sen parametrit (kuva 46).

Havaintoaineisto voidaan myös jättää kompressoimatta, mikä on oletusasetus. Vaihdettaessa menetelmää vaihtuu myös graafiset kontrollit, joilla asetetaan menetelmän parametrit. Cancel-painikkeella on mahdollista keskeyttää prosessi. Next-painiketta painettaessa suoritetaan mahdollisesti havaintoaineiston kompressointi, johon käytettävä aika mitataan. Kompressioparametrit tallennetaan tiedostoon - jos käyttäjä on niin valinnut.

NDA:ssa on kahdenlaisia ryvästelymenetelmiä, joista ensimmäiset ryvästelevät havaintoaineiston täysin havaitun osajoukon avulla. Ne siis muodostavat osaluokituksen, joka ei sisällä puutteellisia havaintoja. Tämä luokitus täydennetään koko havaintoaineiston luokitukseksi etsimällä havainnoille lähimmät prototyypit selittävän havaintoaineiston avulla. Jälkimmäiset ryvästelymenetelmät luokittelevat havaintoaineiston käyttäen hyödykseen myös puutteellista osaa havaintoaineistosta. Täten muodostettua luokitusta ei tarvitse täydentää koska se on kokonainen. Jos havaintoaineiston täysin havaittu osa on tyhjä joukko niin tällöin suurinta osaa ryvästelymenetelmistä ei voida käyttää ja kyseiset ryvästelyvaihtoehdot eivät ole valittavissa. Ryvästelyn suorittamista varten ohjelmisto tukee viittä NDA:n mahdollistamaa ryvästelymenetelmää, jotka ovat TS-SOM-, hierarkkinen-, K-Means-, Fuzzy C-Means- ja EM TS-SOM- (keskipisteimputoinnilla) ryvästely. TS-SOM:illa ja EM TS-SOM:illa ryvästely voidaan tehdä hyödyntäen myös puutteellista havaintoaineistosta. Muilla menetelmillä ryvästely tehdään täysin havaitusta havaintoaineistosta. EM TS-SOM algoritmi ei ole oikea Expectation Maximization -algoritmi. Algoritmi tekee iteraativisesti parametrien (TS-SOM:n painovektorit) uudelleen laskemista ja keskipisteimputointia. Algoritmin konvergointia voidaan arvioida kovarianssimatriisin ja puuttuvien arvojen muutoksen kuvaajilla. Hierarkista ryvästelyä voidaan tehdä yhdistämis- ja hajautusmenetelmillä. Yhdistämismenetelmiä on kuusi ja hajautusmenetelmiä kaksi. Hierarkkinen ryvästely yhdistämismenetelmillä on käytännöllinen vain kun havaintoaineisto on pieni, koska menetelmät käyttävät paljon muistia.

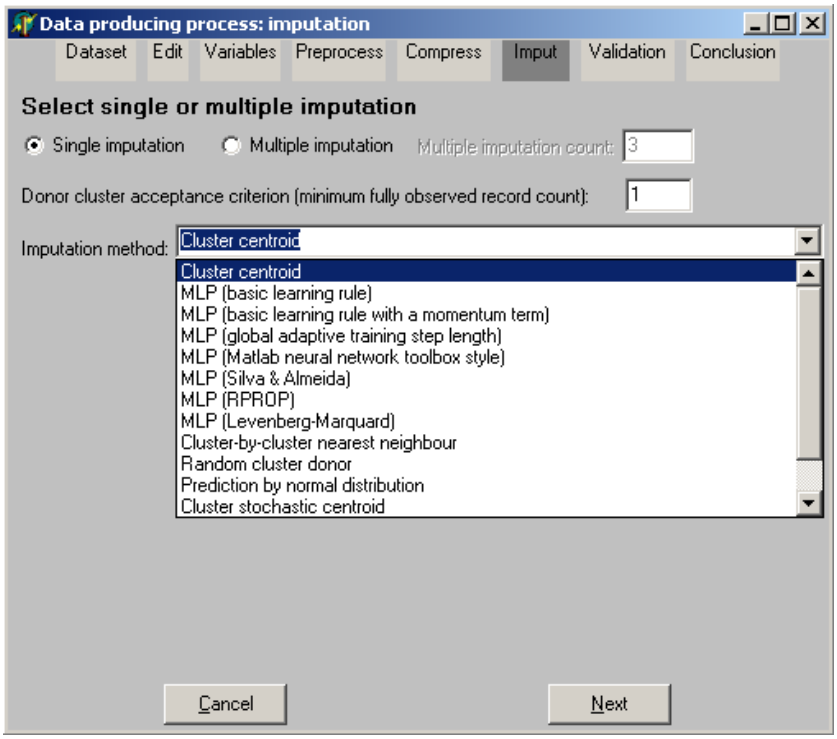


Kuva. 46: Kompressioparametrien asettaminen

Imputointiparametrien asettaminen

Imputointivaiheessa (kuva 47) valitaan halutaanko suorittaa yksikkö- vai moni-imputointi (Single- ja Multiple imputation -painikkeet). Jälkimmäisessä tapauksessa valitaan moni-imputointien lukumäärä m (Multiple imputation count), jolloin muodostetaan m kappaletta kokonaisia havaintojoukkoja. Moni-imputoinnin loppuvaiheessa lasketaan yksi kokonainen havaintojoukko m:stä imputoidusta havaintojoukosta. Valitulle imputointimenetelmälle voidaan asettaa parametrit, joilla on olemassa oletusarvot. Oletuksena on yksikköimputointi ja menetelmänä lähin naapuri ryvästelemättömälle havaintoaineistolle ja keskipisteimputointi ryvästellylle havaintoaineistolle. Moni-imputointien oletusmäärä on kolme. Imputointimenetelmää vaihdettaessa vaihtuu myös kontrollit, joilla määritellään menetelmän parametrit. Prosessin keskeyttäminen voidaan suorittaa Cancel-painikkeella. Next-painiketta painettaessa suoritetaan havaintoaineiston täydentäminen, johon kuluva aika mitataan. Jos prosessiparametrien tallentaminen on valittu prosessin alussa niin imputointiparametrit tallennetaan tiedostoon. Imputointi voi epäonnistua MLP regressiomenetelmillä jos jossakin ryppäässä ei ole täysin havaittuja havaintoja. Tällöin näytetään käyttäjälle virheilmoitus. Tämän jälkeen käyttäjä voi valita jonkun muun imputointimenetelmän. Vaihtoehtoisesti hän

voi aloittaa prosessin alusta ja vähentää jakauman mallin kompleksisuutta.



Kuva. 47: Imputointiparametrien asettaminen

NDA:n avulla pystyttiin tekemään seuraavat imputointimenetelmät:

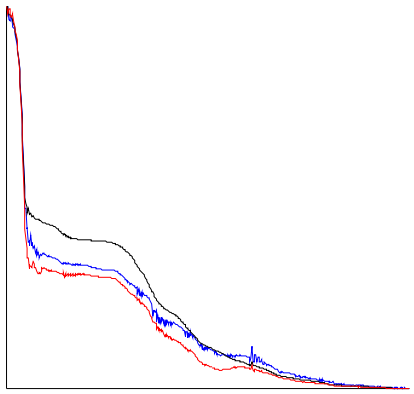
1. lähimmän naapurin menetelmä,
2. arvojen ennustaminen normaalijakauman avulla,
3. satunnainen luovuttajaimputointi,
4. keskiarvoimputointi,
5. stokastinen keskiarvoimputointi,
6. robusti keskiarvoimputointi,
7. stokastinen robusti keskiarvoimputointi,
8. MLP monimuuttujaregressio,

9. MLP regressio muuttujittain ja
10. (puutteellisten poistaminen).

Keskiarvo tarkoittaa ryväsimputoinnissa ryppään keskipistettä. Menetelmät 2, 3, 5 ja 7 tuottaa varianssia moni-imputoituihin arvoihin. Lähimmän naapurin menetelmä tuottaa varianssia imputointien välillä (jos luovuttajia on useampi) mutta moni-imputointia ei ole tuettu. Imputointimenetelmät on selitetty luvuissa 2 ja 4. MLP menetelmissä Backpropagation-opetusalgoritmista on käytettävissä seitsemän versiota, jotka ovat

- perus BP-algoritmi (engl. *Basic learning rule*),
- perus BP-algoritmi momenttitermillä (engl. *Basic learning rule with a momentum term*),
- globaali adaptiivinen opetusaskeleen pituus algoritmi (engl. *Global adaptive training step length*),
- Matlabin neuroverkko työkalun tyylinen algoritmi (engl. *Matlab neural network toolbox style*),
- Silva ja Almeida -algoritmi,
- RPROP-algoritmi ja
- Levenberg-Marquardt -algoritmi.

On suositeltavaa käyttää RPROP-menetelmää koska se oppii nopeammin kuin muut. Klusteroimattomalle havaintoaineistolle MLP:n opetus-, testi- ja validointijoukon virheitä voidaan visualisoida. Kuvassa 47.1 kuvaajan keskivaiheessa alin viiva on testijoukon-, keskimmäinen viiva on validointijoukon- ja ylin viiva on opetusjoukon virhe. Kuvaajien avulla voidaan tapauskohtaisesti päätellä onko MLP-verkko oppinut havaintoaineiston.

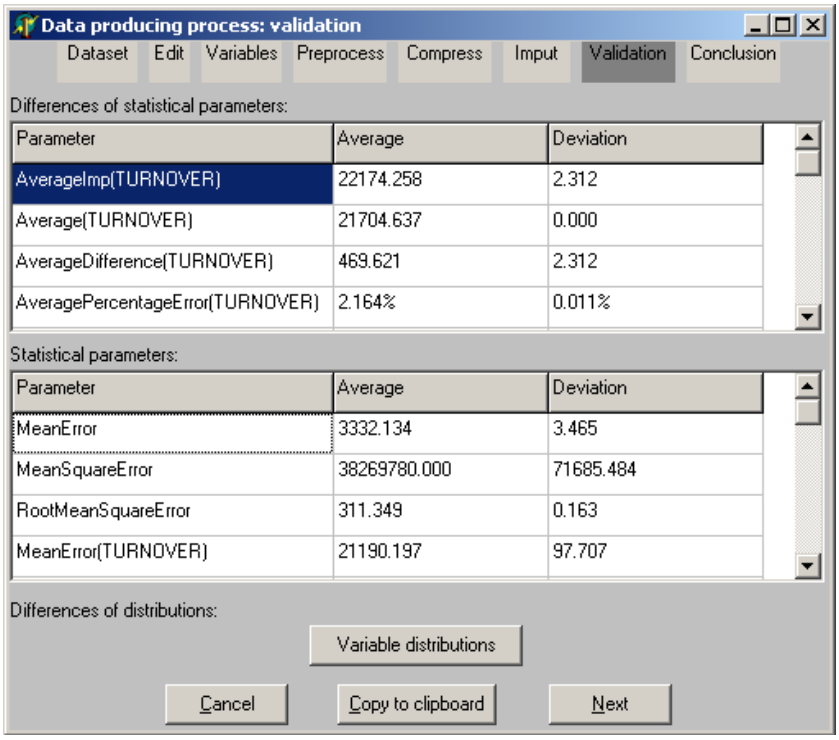


Kuva. 47.1: MLP:n virheet

Ryvästellyn havaintoaineiston imputointimenetelminä voidaan käyttää samoja menetelmiä kuin ryvästelemättömän havaintoaineiston kanssa. Keskiarvomenetelmissä käytetään keskiarvona ryppäiden keskipistettä. Satunnainen luovuttaja ja robusti keskiarvoimputointi käyttävät klusterin hyväksymiskriteeriä (donor cluster acceptance criterion). Kriteeri määrittelee imputoinnissa käytettävät havaintojoukot. MLP imputointimenetelmät vaativat, että jokaisessa klusterissa on vähintään yksi täysin havaittu havainto. Jos täysin havaittu osa on tyhjä, niin kyseisillä menetelmillä imputointiprosessi keskeytyy, minkä jälkeen voidaan valita toinen menetelmä. Tosin jos halutaan käyttää MLP menetelmää niin kompressointimallin kompleksisuutta (= klusterien/neuronien määrää) on vähennettävä. Klusteroidun havaintoaineiston imputointimenetelmät toimivat täysin samalla tavalla kuin klusteroimattoman havaintoaineiston menetelmät paitsi, että ne käyttävät havaintoaineiston osajoukkoa imputointimallin muodostamiseen. Ryväsimputointimenetelmien etuna on, että ne mallintavat monihuippuisen havaintojoukon.

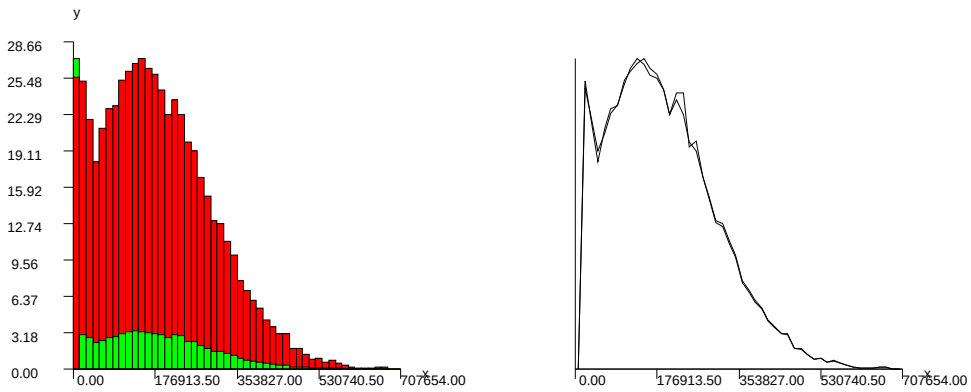
Validointitulokset

Ennen validointivaihetta havaintoaineisto leikataan minimi-maksimi hyperkuutiolla, mikä estää alueen ulkopuolisten havaintopoikkeamien tuottamisen. Hyperkuutio on laskettu alkuperäisestä havaintoaineistosta. Validointiin kuluva aika mitataan.



Kuva. 48: Validointitulokset

Validointivaiheessa tutkitaan tuloksien validisuus laskemalla tunnuslukuja (kuva 48) ja vertaamalla jakaumia eli tiheysfunktion kuvaajia (kuva 48.1). Suunnitteluvaiheessa kaavailtuihin ominaisuuksiin lisättiin mahdollisuus tallentaa tulokset leikepöydälle. Tallentaminen tapahtuu Copy to clipboard -painikkeella. Tiheysfunktioiden vertailu tehdään Variable distributions -painikkeella. Cancel-painikkeella prosessi voidaan keskeyttää. Next-painikkeella siirrytään seuraavaan vaiheeseen. Tiheysfunktion estimointimenetelmiä ovat histogrammi (kuva 48.1 vasen), Parzenin ikkuna (kuva 48.1 oikea), Parzenin ikkuna Gaussisella painotuksella, K-lähimmän naapurin menetelmä, K-lähimmän naapurin menetelmä Gaussisella painotuksella ja MeanV Gaussisella painotuksella. Menetelmien yleinen parametri on poimintatarkkuus. Muita menetelmäkohtaisia parametreja ovat mm. Parzenin ikkunan leveys, Gaussisen jakauman varianssi ja lähimpien naapurien määrä.



Kuva. 48.1: Tiheysfunktion kuvaajat, imputointimenetelmänä ryppään keskipiste: histogrammi (vasen); Parzenin ikkuna (oikea)

Validoinnin avulla voidaan verrata eri mallinnus- ja imputointimenetelmien tuottamia tuloksia. Validointi voi tapahtua sovelluksessa kahdella eri tavalla, riippuen siitä onko virheetön vertailu havaintoaineisto saatavilla. Jos se on saatavilla, niin tällöin imputoitua havaintoaineistoa voidaan verrata alkuperäiseen. Vertailu havaintoaineiston puuttuessa vertauskohteena käytetään jakaumaa, joka muodostuu muuttujien havaituista arvoista. Edellisten lisäksi validointi voidaan tehdä myös Bootstrap-menetelmällä. Validoinnissa lasketaan tilastolliset tunnusluvut sekä niiden välinen erotus ja ero prosentteina. Jälkimmäinen lasketaan kaavalla

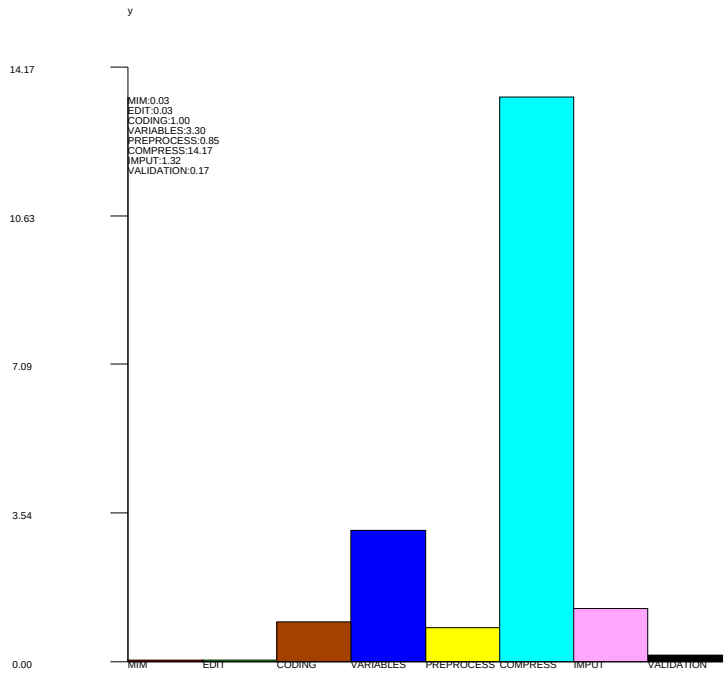
$$\text{Error}_{\text{percentage}} = \frac{100\% * (x - \hat{x})}{x}.$$

Kaavan ongelma on että kun alkuperäinen tunnusluku x on nolla (tai riittävän lähellä sitä) niin tulos ei mahdu enää tietokoneen liukulukuarvoon. Ongelman ratkaisuna on jättää prosentuaalinen ero laskematta kun x on riittävän lähellä nollaa.

Yhteenveto

Yhteenvetovaiheessa imputoitu aineisto voidaan tallentaa NDA:n tukemaan tiedostoformaattiin Save dataset -painikkeella. Validointituloksiin on myös mahdollista palata painamalla Validation-painiketta. Näiden lisäksi prosessin aikana tehty aika-analyysi voidaan visualisoida kuvan 49 mukaisena pylväsdiagrammina, jolloin käyttäjä näkee eri vaiheisiin kuluneen ajan. Visualisointi nähdään kun painetaan Time consumption -painiketta. Aika-analyysin avulla voidaan verrata eri vaihtoehtojen laskennal-

lista kompleksisuutta. Ajankulutus grafiikan Y-akseli on sekunteina. X-akselilla olevat pylväät ovat mitatut prosessin eri vaiheet. Prosessi päätetään Done-painikkeella. Tämän jälkeen täydennetty havaintoaineisto on käytettävissä.



Kuva. 49: Yhteenveto - ajankulutus

5.4 Testaus ja arviointi

Testaaminen koostui sovelluksen yleisen toimivuuden- ja editointi- ja imputointimenetelmien validoinnista. Sovelluksen yleisen toimivuuden testaus suoritettiin Windows 2000, Windows 98 ja Windows ME -ympäristöissä. Kuten toteutusvaiheen ongelma alikappaleessa (5.3.1) todettiin, aiheutti graafisen näkymän dynaamiset komponentit GDI-systeemiresurssien loppumisen Windows 98 ja -Millennium käyttöjärjestelmissä. Ratkaisuksi ongelmaan päätettiin poistaa dynaamisten komponenttien luominen kyseisissä järjestelmissä, minkä jälkeen ohjelma toimi. Tosin grafiikkanäkymien toimin-

nallisuus hävisi käytännössä kokonaan, mutta tiedon tuotantoprosessia se ei haittaa. Sovelluksen yleistä toimivuutta testattiin käsittelemällä sillä EurEdit-projektin ohella saatuja testiaineistoja eri menetelmin. Sovellusta testattaessa löytyi myös NDA ohjelmistosta joitakin virheitä, esimerkiksi TS-SOM:in havaintojen luokittelualgoritmi kaatui jos havainnon kaikki komponentit olivat puutteellisia.

5.4.1 Tavoitteiden toteutuminen

Ohjelmistossa TS-SOM-algoritmia voidaan käyttää havaintoaineiston jakauman mallintamiseen. TS-SOM on käytettävistä jakauman mallinnusmenetelmistä nopein. Isoja datamassoja pystytään käsittelemään ohjelmiston avulla. Rajoitteeksi muodostuu diskreettien muuttujien koodaaminen kategorisiksi muuttujiksi. Isoilla datamassoilla yhden kategorian muuttuja voi viedä paljon muistia. Jos kategorioita on esimerkiksi satoja, niin muistinkulutus voi kasvaa huomattavasti. Ohjelmisto toimii fyysisen muistin loppuessa, mutta laskennallinen toiminnallisuus hidastuu huomattavasti käytettäessä virtuaalista muistia. Ennakko-oletuksena oli, että kategoriset muuttujat vievät paljon muistia suurien datamassojen kanssa. Niiden muistinkulutusta voitaisiin vähentää 32-kertaisesti lisäämällä NDA:han 1-bittinen muuttujatyyppi. Lisääminen merkitsisi sitä, että useisiin käskyihin täytyisi ohjelmoida tuki uudelle tietotyypille. Tiedon tuotantoprosessi mahdollistaa havaintoaineistojen loogisten virheiden korjaamisen, olettaen että editointisäännöt on määritelty, ja puutteellisuuksien täydentämisen. Prosessin luotettavuus voidaan arvioida validoimalla, mikä auttaa käyttäjää. EurEdit-testiaineistot onnistuttiin käsittelemään ohjelmistolla. Editointi- ja imputointimenetelmiä testattiin kolmella havaintoaineistolla, jotka olivat Danish Labour Force Survey-, UK Annual Business Inquiry (sektori 2/1998)- ja UK Household SARS -havaintoaineisto. Testikoneena käytettiin Celeron 700 MHz konetta, jossa oli 256 megatavua muistia ja Windows 2000 -käyttöjärjestelmä. Testauksessa käytettiin mallintamiseen TS-SOM:ia, muuttujat yhdenmukaistettiin varianssien perusteella ja prosessin tulokset validoitiin Bootstrap-menetelmällä. Kokonaisuudessaan ohjelmisto toteuttaa sille asetetut tavoitteet.

5.4.2 Testitulokset

Danish Labour Force Survey tulokset

Testattu havaintoaineisto oli synteettinen ja siinä oli 20000 havaintoa, joista 14633 (= 73,2%) oli täysin havaittu. Editointisääntöjä ei ollut saatavilla. Muuttujia oli yhteensä 14 kappaletta, joista vain yksi (INCOME) oli puutteellinen. TS-SOM:issa käytettiin neljää kerrosta lukuunottamatta MLP regressiomenetelmää jonka kanssa käytettiin vain kahta. MLP-verkossa oli viisi piilokerrosta, joissa jokaisessa oli 25 neuronia. MLP:n opetusmenetelmänä käytettiin RPROP-algoritmia. Validointi tehtiin suorittamalla prosessi 5 kertaa. Taulukossa 5.1 on havainnollistettu imputointimenetelmien tuottamia

Imputointimenetelmä	**k.a.(imp)**	**k.a.virhe**
Keskipiste	180381.156±249.935	0.990%±0.137%
Lähin naapuri	176749.344±342.015	2.983%±0.188%
Satunnainen luovuttaja	182420.750±595.243	0.251%±0.221%
Ennustaminen normaalijakaumalla	180088.969±315.011	1.150%±0.173%
Stokastinen keskipiste	180456.063±498.423	0.949%±0.274%
Robusti keskiarvo	178225.453±266.964	2.173%±0.147%
Stokastinen robusti keskiarvo	177939.406±485.909	2.330%±0.267%
MLP regressio	178457.625±618.109	2.046%±0.339%

Taulukko 5.1: Danish Labour Force Survey -havaintoaineiston testitulokset 1/2

tuloksia INCOME muuttujan keskiarvolle, keskiarvon prosentuaalisille virheille ja tunnuslukujen vaihteluväleille. Imputoitavan muuttujan oikea keskiarvo oli 182184.453. Taulukosta 5.1 voidaan päätellä että imputointimenetelmät aliarvioivat keskiarvoa. Satunnainen luovuttaja tuottaa keskiarvon, joka on erittäin lähelle oikeaa arvoa.

Imputointimenetelmä	**hajonta(imp)**	**hajontavirhe**
Keskipiste	104150.648±141.154	11.904%±0.119%
Lähin naapuri	116579.852±176.859	1.390%±0.150%
Satunnainen luovuttaja	118482.734±615.455	0.469%±0.236%
Ennustaminen normaalijakaumalla	104326.625±195.480	11.755%±0.165%
Stokastinen keskipiste	116454.773±303.609	1.496%±0.257%
Robusti keskiarvo	104686.625±120.095	11.450%±0.102%
Stokastinen robusti keskiarvo	110989.273±266.094	6.119%±0.225%
MLP regressio	106748.750±407.709	9.706%±0.345%

Taulukko 5.2: Danish Labour Force Survey -havaintoaineiston testitulokset 2/2

Taulukossa 5.2 on havainnollistettu imputointimenetelmien tuottamia tuloksia INCOME muuttujan hajonnalle, hajonnan prosentuaalisille virheille ja tunnuslukujen vaihteluväleille. INCOME muuttujan oikea keskihajonta on 118223.641. Satunnainen luovuttaja tuottaa parhaan tuloksen hajontaa testattaessa. Muut menetelmät aliarvioivat hajontaa.

UK Annual Business Inquiry tulokset

Havaintoaineiston puuttuville arvoille oli saatavilla oikeat arvot validointia varten. Havaintoaineistosta oli olemassa editointisäännöt, tosin ne eivät sisältäneet editointi- tai imputointitoimintoja, joten niillä pystyi vain validoimaan havaintoaineistoa. Muuttujia on yhteensä 38 kappaletta. Havaintoaineistossa on puuttuvuutta kaikissa muissa muuttujissa paitsi kuudessa. Havaintoja oli 5594 kappaletta joista 3976 (71,1%) oli täysin havaittu. Jakauman mallia tehtäessä ei käytetty REF (juokseva laskuri) ja WEIGHT (otospainot) muuttujia koska ne ovat siihen tarkoitukseen "hyödyttömiä".

Testauksessa mitattiin imputoidun havaintoaineiston etäisyyttä alkuperäisestä ja havaittujen virheiden määrää. TS-SOM:issa käytettiin kahta kerrosta MLP-regressioiden ja lähimmän naapurin kanssa. Kolmea kerrosta käytettiin robustin keskiarvon, stokastisen robustin keskiarvon ja stokastisen keskipisteen kanssa. Neljää kerrosta käytettiin muiden menetelmien kanssa. MLP neuroverkossa oli kolme piilokerrosta, joissa jokaisessa 15 neuronia, ja opetusmenetelmänä oli RPROP. Keskimääräisen havaintovirheen validoimiseksi prosessi ajatettiin 10 kertaa. Taulukoissa 5.3, 5.4 ja 5.5 tähdellä (*) merkit-

ty menetelmä (lähin naapuri) on tehty ilman havaintoaineiston ryvästelyä. Taulukosta

Imputointimenetelmä	**keskimääräinen havaintovirhe**
Keskipiste	115.255±3.622
Lähin naapuri	48.138±0.133
Satunnainen luovuttaja	135.334±13.84
Ennustaminen normaalijakaumalla	120.488±10.743
Stokastinen keskipiste	324.274±10.858
Robusti keskiarvo	89.067±0.0
Stokastinen robusti keskiarvo	90.727±0.338
MLP regressio	147.534±58.495
MLP monimuuttujaregressio	121.6±0.041
EM TS-SOM + keskipiste	29.710±0.834
Lähin naapuri(*)	47.638±0.246

Taulukko 5.3: UK Annual Business Inquiry sektori 2/1998 -havaintoaineiston testitulokset 1/2

5.3 havaitaan että EM TS-SOM (keskipiste imputoinnilla) tuottaa parhaimman tuloksen. Tosin varmaa johtopäätöstä menetelmien paremmuudesta ei voida päätellä, koska muuttamalla havaintoaineiston jakauman ja imputointimallien kompleksisuutta saadaan erilaisia tuloksia.

Imputointimenetelmä	**virheitä**	**virheiden muutos**
Keskipiste	36790	-722
Lähin naapuri	36320	-1192
Satunnainen luovuttaja	36741	-771
Ennustaminen normaalijakaumalla	36712	-800
Stokastinen keskipiste	37095	-417
Robusti keskiarvo	36440	-1072
Stokastinen robusti keskiarvo	36708	-804
MLP regressio	37469	-43
MLP monimuuttujaregressio	36492	-1020
EM TS-SOM + keskipiste	36350	-1162
Lähin naapuri(*)	36309	-1203

Taulukko 5.4: UK Annual Business Inquiry sektori 2/1998 -havaintoaineiston testitulokset 2/2

Ennen imputointia havaintoaineistossa oli 37512 editointisääntöjen havaitsemaa virhettä. Havaittujen virheiden lukumäärä on laskettu satunnaisista prosessin tuottamista havaintojoukoista, koska editointi- ja imputointiohjelmistossa ei ole vielä automaattista virheiden lukumäärän seurantaominaisuutta. Tämän vuoksi virheiden muutoksen luotettavuudesta ei ole varmuutta. Kaikki testatut menetelmät näyttävät kuitenkin vähentävän testiaineistosta virheitä. Virheiden väheneminen on kuitenkin erittäin vähäistä.

Tiedon tuotantoprosessin vaiheisiin keskimääräisesti käytetty aika oli seuraavanlainen

- puuttuvuuksien havaitseminen: 0.03s,
- virheiden havaitseminen ja korjaaminen: 0.88s,
- muuttujien koodaaminen: 1.00s,
- täysin ja osittain havaittujen muuttujien tunnistaminen: 0.96s,
- esikäsittely: 0.22s,
- kompressointi: 4.9s ja
- validointi: 0.92s.

Huom: keskiarvon laskemisesta on jätetty pois lähimmän naapurin- ja EM TS-SOM keskipisteimputointi. MLP menetelmissä datan jakauman malli oli huomattavasti yk-

Imputointimenetelmä	**käytetty aika**
Keskipiste	0.09s
Lähin naapuri	14.99s
Satunnainen luovuttaja	1.44s
Ennustaminen normaalijakaumalla	2.04s
Stokastinen keskipiste	3.41s
Robusti keskiarvo	2.98s
Stokastinen robusti keskiarvo	3.72s
MLP regressio	1809.10s
MLP monimuuttujaregressio	100.19s
Lähin naapuri(*)	45.75s

Taulukko 5.5: UK Annual Business Inquiry sektori 2/1998 -havaintoaineiston imputoinnin aikavaativuus (satunnainen otos prosessista)

sinkertaisempi kuin muissa, joten tulokset MLP:n ja muiden menetelmien välillä eivät ole suoraan vertailtavia. MLP:n laskennallisen aikavaativuuden selittää, se että MLP-verkkoja rakennettiin 32:ta ulostulomuuttujaa kohti. Yhteensä MLP-verkkoja tehtiin 96 kappaletta.

UK Household SARS tulokset

UK Household SARS -havaintoaineistossa on 47704 havaintoa, joista 15026 (31,5%) on täysin havaittu. Havaintoaineistossa on 31 muuttujaa, jotka ovat kaikki puutteellisia. Käytettävissä oli editointisäännöt, jotka sisälsivät eksplisiittisiä editointitoimintoja.

TS-SOM:issa käytettiin kuutta kerrosta, paitsi satunnaisella luovuttajalla ja robustilla keskiarvolla vain viittä. MLP regressio ei ollut käytettävissä koska kaikki muuttujat olivat osittain havaittuja. Tuloksissa mitattiin virheiden määrää. Ennen imputointia havaintoaineistossa oli 809 korjattavaa virhettä ja 79531 editointisääntöjen havaitsemaa virhettä. Säännöissä oli korjaustoiminnot vain osalle virheistä. Imputointia ei suoritettu lähimmän naapurin menetelmällä koska se olisi kestänyt kauan. Havaittujen virhei-

Imputointimenetelmä	**imputoitavia**	**virheitä/muutos**
Keskipiste	1110	89436/+9905
Satunnainen luovuttaja	1900	89700/+10169
Ennustaminen normaalijakaumalla	1253	89511/+9980
Stokastinen keskipiste	1666	89534/+10003
Robusti keskiarvo	1121	89698/+10167
Stokastinen robusti keskiarvo	1361	8970/+10169

Taulukko 5.6: UK Household SARS -datan testitulokset

den lukumäärä on satunnainen otos prosessista. Taulukosta 5.6 havaitaan että kaikki testatut imputointimenetelmät tuottivat virheitä havaintoaineistoon lisää. Havaintoaineiston editointisäännöt olivat huomattavasti laajemmat kuin UK ABI -aineiston. Editointisäännöissä oli 174 sääntöä kun taas ABI aineiston säännöissä oli vain 22. Tämän vuoksi editointisäännöt havaitsevat tarkemmin virheitä.

5.4.3 Arvio imputointimenetelmistä

Seuraavaksi tarkastellaan sovelluksen tukemien imputointimenetelmien etuja ja haittoja, joista yhteenveto on koottu taulukkoon 5.7. Lähimman naapurin menetelmä tuottaa arvoja, jotka ovat usein "realistisia". Tämä johtuu siitä, että luovuttaja havainto etsitään imputoitavan havainnon havaitulla osalla. Menetelmä on kuitenkin varsin hi-

das ilman ryvästelyä, koska jokaiselle havainnolle joudutaan käymään lävitse kaikki muut havainnot ja etsimään minimietäisyys. Normaalijakauman avulla ennustettaessa täydentäminen on nopeaa ja arvoihin tulee varianssia. Haittapuolena on että oletuksena on normaalijakauma, joka on herkkä havaintopoikkeamille. Toisena haittapuolena on, että reaalimaailman havaintoaineisto ei ole välttämättä normaalijakautunutta. Sameer Agarwal:in mukaan täysin satunnainen todennäköisyyspoimintaan perustuva imputointi on käytännöllinen menetelmä vain aineiston alijoukoissa [2]. Satunnainen luovuttaja tuottaa varianssia imputoituihin arvoihin. Toisaalta luovuttajan etsinnässä ei käytetä havaittua osaa puutteellisesta havainnosta. Ryvästelyn käyttäminen satunnaiseen luovuttajamenetelmän kanssa tehostaa sitä huomattavasti koska jakauman muoto säilyy. Keskiarvoimputointi on erittäin nopeaa, koska havaintoaineistosta tarvitsee laskea vain keskiarvo, jolla puuttuvat havainnot täydennetään. Menetelmän haittapuolina on, että jakauma piikittyy ja keskiarvon herkkyys havaintopoikkeamille. Ryvästelyn kanssa keskiarvoimputointi toimii hyvin, koska havaintoaineiston jakauman malli on käytettävissä ja havaintoaineiston piikittymistä voidaan vähentää huomattavasti. Tällöin imputoituihin arvoihin tulee varianssia. Stokastisen keskiarvoimputoinnin haittana on normaalijakauma oletus, joten se on herkkä havaintopoikkeamille. Toisaalta se tuottaa varianssia. Robusti keskiarvoimputointi sietää havaintopoikkeamia. Sen haittapuolena on jakauman piikittyminen. Käytettäessä ryvästelyä piikittymistä saadaan vähennettyä ja tuotettuihin arvoihin muodostettua varianssia. Robustisen keskiarvoimputoinnin stokastinen versio tuottaa varianssia imputoituihin arvoihin. Tuotetuissa arvoissa on varianssia, koska ne poimitaan normaalijakaumasta. Menetelmä ei ole herkkä havaintopoikkeamille, koska keskiarvo ja varianssi ovat määritelty robustisesti. MLP monimuuttujaregression etuna on, että havaintoaineiston jakaumasta ei tehdä oletusta. Haittana on opettamisen hitaus. MLP muuttujittaisessa regressiossa ei ole myöskään oletusta jakaumasta. Menetelmä on hidas, koska jokaista puutteellista muuttujaa varten on opetettava MLP-verkko. Ryvästelyn kanssa MLP menetelmät ovat usein erittäin hitaita, koska MLP-verkkoja on opetettava paljon. Puutteellisten poistaminen ei ole varsinainen imputointimenetelmä. Se on menetelmä, jota käytetään puutteellisten havaintojen käsittelyyn. Sen etuna on, että se on nopea. Haittana on että informaatiota hylätään ja havaintojen lukumäärä ei säily alkuperäisenä. Se ei käytä tehokkaasti havaittua dataa [43]. Tämän lisäksi johtopäätökset saattavat vääristyä [42]. Ryvästelyllä imputointimenetelmien toiminta usein paranee koska havaintoaineiston jakauma on mallinnettu, minkä ansiosta imputoiduissa arvoissa on varianssia.

Imputointimenetelmä	**Edut ja haitat**
Lähimman naapurin menetelmä	+Arvot ovat ”realistisia” –Hidas
Ennustaminen normaalijakauman avulla	+Nopea +Arvoissa on varianssia –Oletuksena normaalijakauma –Herkkä havaintopoikkeamille
Satunnainen luovuttaja	+Arvoissa on varianssia +Ryvästelyn kanssa hyvin toimiva –Ei huomioi havaittuja arvoja
Keskiarvoimputointi	+Nopea +Toimiva ryvästelyn kanssa –Jakauma piikittyy (ilman ryvästelyä) –Herkkä havaintopoikkeamille
Stokastinen keskiarvoimputointi	+Tuottaa varianssia –Oletuksena normaalijakauma –Herkkä havaintopoikkeamille
Robusti keskiarvoimputointi	+Sietää havaintopoikkeamia +Toimiva ryvästelyn kanssa –Jakauma piikittyy (ilman ryvästelyä)
Stokastinen robusti keskiarvoimputointi	+Sietää havaintopoikkeamia +Arvoissa on varianssia –Oletuksena normaalijakauma
MLP monimuuttujaregressio	+Ei oletusta jakaumasta –Hidas
MLP regressio muuttujittainen	+Ei oletusta jakaumasta –Hidas –Saattaa hylätä informaatiota
Puutteellisten poistaminen	+Nopea –Informaatiota hylätään –Havaintojen lukumäärä ei säily

Taulukko 5.7: Imputointimenetelmien edut ja haitat

5.4.4 Yhteenveto

Kokonaisuudessaan tehty ohjelmisto toimii suhteellisen hyvin ja se toteuttaa sille asetetut tavoitteet. Ohjelmistoa tultaneen kehittämään tulevaisuudessa. Prosessin alkuun lisätään toiminto, jonka avulla koko prosessi voidaan suorittaa yhdestä painikkeesta. Käytännössä tämä tarkoittaa sitä, että suoritetaan DoDPP makro joka tulkkaa aiemmin tallennetun prosessitiedoston. Koko prosessiin tarvitaan todennäköisesti painike joka vaiheeseen, jolla voidaan palata aiempaan vaiheeseen. Kategoristen muuttujien muistinkulutusta tultaneen vähentämään. Esikäsittelymenetelmiin lisätään todennäköisesti normeeraminen ja whitening-menetelmä, jolla poistetaan havaintoaineistosta korrelaatio. Havaintoaineiston mallin rakentamista muutetaan siten, että osittain havaittuja muuttujia ei ole pakko käyttää opettamisessa. Selittävien muuttujien valinta siirrettäneen MLP regressiomenetelmän parametriksi. Menetelmien ajankulutuksen validointi Bootstrap-menetelmällä lisätään myös ohjelmistoon. Validointivaiheeseen lisätään mahdollisuus visualisoida alkuperäinen muuttuja ja imputoitu muuttuja. Tämä on yksinkertainen mutta suhteellisen tehokas tapa visualisoida imputoinnin tulosta. Virheiden lukumäärän validointi on myös tarpeen. Tällöin prosessi suorittaa imputoidun havaintoaineiston editoinnin automaattisesti ja ottaa ylös virhemäärät. Mallipohjainen editointimenetelmä kehitetään todennäköisesti. Monet havaintoaineistot ovat hierarkisia rakenteeltaan, joten niitä varten on kehitettävä omat menetelmänsä. Otospainojen käyttäminen on myös järkevää lisätä ohjelmaan, jotta tunnusluvut saadaan suoraan oikeaan muotoon.

Luku 6

Johtopäätökset

On muistettava että empiiristen mallien, joita mm. tilastotieteilijät usein käyttävät, luominen regressiomallinnetuista havaintojoukoista ilman riittävän vahvaa varmuutta/tietämystä tuloksista ei ole suotavaa. Regressio tekee usein harhaa kahden riippumattoman muuttujien väliseen kovarianssiin [30]. Tämä johtuu siitä, että regressiomalleissa oletetaan taustatietona käytetyn muuttujan selittävän puutteellisen muuttujan. Jos muuttujat ovat riippumattomia ja silti käytetään kyseistä mallia, niin se tuottaa harhaa tuloksiin. Lähteen [30] mukaan onkin suositeltavampaa, että empiiriset mallit luodaan täysin havaituista havaintojoukoista. Tosin jos johtopäätöksien tekijä on varma että regressiomalli ei ole tuottanut "harhaa" tuloksiin, niin hän voi tehdä johtopäätöksiä täydennetystä havaintoaineistosta.

Tiedon tuotantoprosessin ongelmana on mm. imputointimallin ja havaintoaineiston jakauman mallin kompleksisuuden määrittäminen. Lisättäessä jakauman mallin kompleksisuutta vähenee imputointimallin kompleksisuus. Tämän seurauksena ongelmaksi voi muodostua esimerkiksi se, että lähimmät luovuttajat saattavat luokittua johonkin viereiseen ryppääseen. Tällöin lokaalissa imputointimallissa ei välttämättä ole käytettävissä ryppään jokaisen puuttuvan havainnon lähintä havaintoa. Taulukosta 5.3 nähdään mm. että luovuttajamenetelmän kanssa käytetty havaintoaineiston jakauman malli (2-kerroksinen TS-SOM) saattaa olla liian kompleksinen. Tämän näkee siitä, että ilman ryvästelyä (tähdellä merkitty menetelmä) tuotettu tulos on parempi. Toinen ongelma

on se, että havaintoaineiston jakauman mallia rakennettaessa käytetään kaikkia osittain havaittuja muuttujia. Järkevämpää olisi rakentaa jakauman malli vain joistakin tärkeistä muuttujista. Normaalijakauman avulla suoritettavan imputoinnin ongelmaksi muodostuu se, että täydennetyksi arvoksi voi tulla mikä tahansa arvo havaintoaineiston minimi-maksimi vaihteluväliltä. Tosin sitä pienemmällä todennäköisyydellä mitä kauempana keskiarvoa se sijaitsee.

Testituloksista (taulukko 5.3) nähdään, että yksilötasolla ei voida taata virheettömyyttä. Populaatiotasolla, kuten tunnusluvussa keskiarvo, voidaan saavuttaa suhteellisen hyviä tuloksia (taulukko 5.1), riippuen havaintoaineistosta. Samaan johtopäätökseen päätyi myös Pasi Piela suorittamassaan lähimmän naapurin TS-SOM imputoinnissa [27]. Editointisääntöjen antamista virhemääristä (taulukko 5.6) voidaan päätellä, että yksikään testattu imputointimenetelmä ei ole toistansa juurikaan parempi korjattujen virheiden määrää mitattaessa. Virheiden suhteen voi käydä kuten UK Household SARS havaintoaineiston kanssa (taulukko 5.4) eli imputoinnissa voi syntyä uusia virheitä. Tämä on jossain määrin itsestäänselvyys, koska ohjelmiston imputointijärjestelmä ei käytä editointisääntöjä. Tämän vuoksi se ei voi toimia tehokkaasti virheiden korjaamista ajateltaessa. Jos editointisäännöt sisältäisivät eksplisiittisen editoinnin jokaiselle havaitulle virheelle, niin imputoinnista ei välttämättä tarvitsisi olla yhteyttä editointijärjestelmään. Yhdistämällä editointisääntöjen käyttö imputointivaiheeseen saataisiin virheitä vähennettyä enemmän. Toisaalta imputointivaihe saattaisi hidastua huomattavasti, riippuen mm. editointisääntöjen määrästä.

Kirjallisuutta

[1] Acock Alan C.: *Working with Missing Data*, saatavilla WWW-muodossa <URL: http://www.orst.edu/instruct/hdfs632/MissingData.html>, viitattu 1.8.2001

[2] Agarwal S. (2001): *Learning from Incomplete Data*, saatavilla PDF-muodossa <URL: http://www.cs.ucsd.edu/users/elkan/254/sagarwalrep.pdf>, viitattu 6.2.2002

[3] Ahmad Subutai and Tresp Volker (1993): *Some Solutions to the Missing Feature Problem in Vision*, Advances in Neural Information Processing Systems, Volume 5, pp. 393-400

[4] BBN Corporation (1997): *PROPHET StatGuide: Examining normality test results*, saatavilla WWW-muodossa <URL: http://www.basic.nwu.edu/statguidefiles/n-dist_exam_res.html>, viitattu 7.12.2002

[5] Berthouex, P.M. and Brown L.C. (1994): *Statistics for Environmental Engineers*, CRC Press, London

[6] Bishop Christopher M. (1995): *Neural Networks for Pattern Recognition*, Oxford University Press

[7] Breckling Jens, Kokic Philip, Lübke Oliver (2001): *A note on multivariate M-quantiles*, Statistics & Probability Letters, Volume 55(1), pp. 39-44

[8] Della Rocca G., Luzi O. Pagliuca D. and Riccini E. (2001): *The integration of algorithms for outlier detection in the generalised editing and imputation system Concord*, saatavilla PDF-muodossa: <URL: http://webfarm.jrc.cec.eu.int/etk-ntts/Papers/final_papers/67.pdf>, viitattu 6.2.2002

[9] Duda Richard O., Hart Peter E. and Stork David G. (2001): *Pattern Classification (2nd edition)*, Wiley-Interscience

[10] Fallon Agata and Spada Christine: *Environment Sampling and Monitoring Primer: Detection and Accommodation of Outliers in Normally Distributed Data Sets*, saatavilla WWW-muodossa <URL: http://www.ce.vt.edu/program_areas/environmental/teach/smprimer/outlier/outlier.html>, viitattu 23.12.2001

[11] Gelman Andrew, King Gary, Liu Chuanhai (1998): *Not asked or not answered: multiple imputation for multiple surveys*, Journal of the American Statistical Association, Volume 93, No. 443, pp. 846-857

[12] Ghahramani Z. and Jordan M. (1994): *Supervised learning from incomplete data via an EM approach*, Advances in Neural Information Processing Systems 6, Morgan Kaufmann Publishers, Inc., pp. 120-127

[13] Himberg Johan (1997): *Itseorganisoituvaan karttaan perustuva työkalu ja sen soveltaminen puheludatan analyysiin (diplomityö)*, saatavilla WWW-muodossa <URL: http://www.cis.hut.fi/ jhimberg/dippa/>, viitattu 10.1.2002

[14] Hopke Philip K., Liu Chuanhai, Rubin Donald B. (2001): *Multiple Imputation for Multivariate Data with Missing and Below-Threshold Measurements: Time-Series Concentrations of Pollutants in the Arctic*, Biometrics, Volume 57(1), pp. 22-33

[15] Häkkinen Erkki (2001): *Design, Implementation and Evaluation of Neural Data Analysis Environment (väitöskirja)*, Jyväskylän yliopiston kirjasto

[16] King Gary, Honaker James, Joseph Anne, Scheve Kenneth (1998): *Listwise Deletion is Evil: What to Do About Missing Data in Political Science*, saatavilla PDF-muodossa <URL: http://web.polmeth.ufl.edu/papers/98/king98e.zip>, viitattu 10.1.2002

[17] King Gary, Honaker James, Joseph Anne and Scheve Kenneth (2001): *Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation*, American Political Science Review, Volume 95 Issue 1, pp. 49-69

[18] King Gary and Zeng Langche (2001): *Logistic Regression in Rare Events Data*, Political Analysis, Volume 9, No. 2, pp. 137-163

[19] Kohonen Teuvo (2001): *Self-Organizing Maps*, Springer

[20] Koikkalainen Pasi (1994): *Neurolaskennan mahdollisuudet*, TEKES - julkaisu 43/94

[21] Koikkalainen Pasi, Häkkinen Erkki, Lensu Anssi ja Rekkilä Mika: *Neural Data Analysis Environment, User's Guide*, saatavilla WWW-muodossa <URL: http://erin.mit.jyu.fi/projects/nda/usrman/usrman.html>, viitattu 4.1.2002

[22] Laiho Johanna (2001): *Mitä tehdä puuttuville arvoille? - Moni-imputointimenenetelmät ja SAS (seminaariesitelmä)*, saatavilla PDF-muodossa <URL: http://www.sas.com/offices/europe/finland/tapahtumat/seminaarit/sugif2001/Sali3/Day2/Laiho_tilastokeskus_SAS_Multiple_Imputatio.pdf>, viitattu 10.1.2002

[23] Little Roderick J.A., Rubin Donald B. (1987): *Statistical Analysis With Missing Data*, New York: Wiley

[24] National Institute of Standards and Technology: *Engineering Statistics Handbook : Kolmogorov-Smirnov Goodness-of-Fit Test*, saatavilla WWW-muodossa <URL: http://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm>, viitattu 7.1.2002

[25] National Institute of Standards and Technology: *Engineering Statistics Handbook : Anderson-Darling Test*, saatavilla WWW-muodossa <URL: http://www.itl.nist.gov/div898/handbook/eda/section3/eda35e.htm>, viitattu 7.1.2002

[26] Oja E. and Kaski S. Eds. (1999): *Kohonen Maps*, Elsevier

[27] Piela Pasi (2001): *Introduction to Self-Organizing Maps Modelling for Imputation - Techniques & Technology*, saatavilla PDF-muodossa <URL: http://webfarm.jrc.cec.eu.int/ETK-NTTS/Papers/final_papers/69.pdf>, viitattu 10.1.2002

[28] Ramoni M. and Sebastiani P. (1999): *Learning Conditional Probabilities from Incomplete Data: An Experimental Comparison*, Proceedings of Uncertainty 99, Seventh International Workshop on Artificial Intelligence and Statistics, Morgan Kaufmann, San Mateo, CA

[29] Reams Gregory A., McCollum Joseph (1998): *The use of multiple imputation in the Southern Annual Forest Inventory System Integrated tools for natural resources inventories in the 21st century: an international conference on the inventory and monitoring of forested ecosystems*, 1998 August 16-19, Boise, ID. Gen. Tech. Rep. NCRS-212. St. Paul, MN: U.S. Department of Agriculture, Forest Service, North Central Research Station, pp. 228-233.

[30] Reams Gregory A., McCollum Joseph (1999): *Evaluating Multiple Imputation Models for the Southern Annual Forest Inventory*, Proceedings of the Section on Statistics and the Environment of the American Statistical Association, saatavilla PDF-muodossa <URL: http://www.srs.fs.fed.us/pubs/ja/ja_reams005.pdf>, viitattu 10.1.2002

[31] Ripley Brian D., B. D. Ripley with N. L. Hjort (1995): *Pattern Recognition and Neural Networks*, University Press

[32] Rubin Donald B. (1987): *Multiple Imputation For Nonresponse In Surveys*, John Wiley & Sons

[33] Russell Stuart (1998): *The EM Algorithm*, saatavilla PostScript-muodossa <URL: http://www.cs.berkeley.edu/~russell/classes/cs281/s98/em.ps>, viitattu 10.1.2002

[34] Ryhänen Timo (1999): *Itseorganisoituva kartta (seminaarityö)*, saatavilla WWW-muodossa <URL: http://www.it.lut.fi/opetus/98-99/1591/seminars/Ryhanen/Ryhanen.html>, viitattu 10.1.2002

[35] Samuhel Michael E. and Huggins Vicki (1984): *Longitudinal Item Imputation In A Complex Survey (research report)*, U.S. Census Bureau, SRD/RR-84/20, saatavilla PDF-muodossa <URL: http://www.census.gov/srd/papers/pdf/rr84-20.pdf>, viitattu 10.1.2002

[36] Sarle Warren S. (1998): *Prediction with Missing Inputs*, JCIS '98 Proceedings, Vol II, Research Triangle Park, NC, pp. 399-402

[37] Schafer Joseph L. (1997): *Analysis of Incomplete Multivariate Data*, Chapman & Hall, London

[38] Schafer Joseph L. and Olsen Maren K. (1998): *Multiple imputation for multivariate missing-data problems: a data analyst's perspective*, Multivariate Behavioral Research. 33, pp. 545-571

[39] Schafer Joseph L. and Yucel Recai M. (1997): *Multivariate linear mixed-effects models with missing values*, saatavilla PDF-muodossa <URL: http://www.stat.psu.edu/~jls/mglmm.pdf>, viitattu 10.1.2002

[40] Statistical Solutions, Ltd. (1999): *Solas 2.0 Manual*

[41] UT-Austin Statistical Services (2000): *General FAQ 25: Handling missing or incomplete data*, saatavilla WWW-muodossa <URL: http://www.utexas.edu/cc/faqs/stat/general/gen25.html>, viitattu 10.1.2002

[42] Van Buuren S., Van Mulligen E.M. and Brand J.P.L. (1994): *Routine multiple imputation in statistical databases*, Proceedings of the 7th International Working Conference on Scientific and Statistical Database Management, Los Alamitos, IEEE Computer Society Press, 28-30 Sept. 1994, pp. 74-78

[43] Wothke (1998): *Longitudinal and multi-group modeling with missing data*, *Modeling longitudinal and multiple group data: Practical issues, applied approaches and specific examples*, Mahwah, NJ: Lawrence Erlbaum Associates

Liite A

Editointi- ja imputointiohjelmiston makrot

Tiedon tiivästämismakrot

Makro	**Toiminto**
DoTSSOMCompression	Tiedon kompressointi TS-SOM menetelmällä
DoTSSOMFullCompression	Tiedon kompressointi TS-SOM menetelmällä
DoHClustCompression	Tiedon kompressointi hierarkisella menetelmällä
DoKMeansCompression	Tiedon kompressointi K-Means menetelmällä
DoFCMCompression	Tiedon kompressointi Fuzzy C-Means menetelmällä
DoEMTSSOMCentroidImputation	TS-SOM + keskipiste imputointi

Imputointimakrot

Makro	**Toiminto**
DoMeanImputation	Keskiarvoimputointi
DoNNImputation	Lähimmän naapurin menetelmä
DoPredictionByND	Ennustaminen normaalijakaumalla
DoRobustMeanImputation	Robusti keskiarvoimputointi
DoStochasticMeanImputation	Stokastinen keskiarvoimputointi
DoStochasticRobustMeanImputation	Stokastinen robusti keskiarvoimputointi
DoRandomDonorImputation	Satunnainen luovuttaja imputointi
DoPredictionByND_MI	Ennustaminen normaalijakaumalla moni-imputointi
DoRandomDonor_MI	Satunnainen luovuttaja moni-imputointi
DoStochasticMean_MI	Stokastinen keskiarvo moni-imputointi
DoStochasticRobustMean_MI	Stokastinen robusti keskiarvo moni-imputointi
DoMLPRegressionMultiPhaseABP	MLP regressio muuttujittain
DoMLPRegressionMultiPhaseBP	MLP regressio muuttujittain
DoMLPRegressionMultiPhaseLMBP	MLP regressio muuttujittain
DoMLPRegressionMultiPhaseMBP	MLP regressio muuttujittain
DoMLPRegressionMultiPhaseMLBP	MLP regressio muuttujittain
DoMLPRegressionMultiPhaseRPROP	MLP regressio muuttujittain
DoMLPRegressionMultiPhaseSABP	MLP regressio muuttujittain
DoMLPRegressionSinglePhaseABP	MLP monimuuttujaregressio
DoMLPRegressionSinglePhaseBP	MLP monimuuttujaregressio
DoMLPRegressionSinglePhaseLMBP	MLP monimuuttujaregressio
DoMLPRegressionSinglePhaseMBP	MLP monimuuttujaregressio
DoMLPRegressionSinglePhaseMLBP	MLP monimuuttujaregressio
DoMLPRegressionSinglePhaseRPROP	MLP monimuuttujaregressio
DoMLPRegressionSinglePhaseSABP	MLP monimuuttujaregressio

Ryväsimputointimakrot

Makro	Toiminto
DoClusterCentroidImputation	Keskipiste imputointi
DoClusterMultiPhaseMLPImputation	MLP imputointi muuttujittain
DoClusterNNImputation	Lähimmän naapurin imputointi
DoClusterPredictionByND	Ennustaminen normaali-jakaumalla
DoClusterRandomDonorImputation	Satunnainen luovuttaja imputointi
DoClusterRobustMeanImputation	Keskiarvoimputointi
DoClusterSinglePhaseMLPImputation	Monimuuttuja MLP imputointi
DoClusterStochasticCentroidImputation	Stokastinen keskipiste imputointi
DoClusterStochasticRobustMeanImputation	Robusti keskiarvoimputointi
DoClusterPredictionByND_MI	Ennustaminen normaali-jakaumalla (moni-imputointi)
DoClusterRandomDonor_MI	Satunnainen luovuttaja (moni-imputointi)
DoClusterStochasticRobustMean_MI	Stokastinen robusti keskiarvo (moni-imputointi)
DoClusterStochasticCentroid_MI	Stokastinen keskipiste (moni-imputointi)

Validointimakrot

Makro	Toiminto
CalculateFinalStatisticalParameters	Laskee valitoitujen parametrien keskiarvon ja hajonnan
CalculateStatisticalParameters2	Laskee kahdelle havaintojoukolle tilastolliset parametrit ja niiden erot
CalculateStatisticalParameters3	Laskee kahden havaintojoukon välisiä "etäisyysparametreja"

Apumakrot

Makro	**Toiminto**
DummyCodeDiscreteVariables	Koodaa valittujen muuttujien jokaisen kategorian muuttujaksi.
SeparateFullPartialExplanatory	Erottaa datasta täysin havaitun, osittain havaitun ja selittävän datan.
BuildCompleteClassification	Rakentaa luokituksen koko datalle
BuildMLPInitializationParameter	Rakentaa MLP:n initialization-parametrin
BuildMLPNetParameter	Rakentaa MLP:n net-parametrin
BuildMLPTypesParameter	Rakentaa MLP:n types-parametrin
CalculateClustersMeans	Laskee ryppäiden keskiarvot
CreateNormallyDistributedNoise	Luo normaalijakautunutta kohinaa
CreateTrainTestValidationSets	Luo (MLP:n) opetus-, testi- ja validointijoukot
create_bootstrap_group	Luo Bootstrap havaintojoukon
create_bootstrap_group2	Luo Bootstrap havaintojoukon
CutToInt	Katkaisee liukuluvut kokonaisluvuiksi
SetupCompressionData	Alustaa tiedon kompressointi
MergeDataframes	Yhdistää havaintojoukot
PickHighestResolutionSOM	Poimii tarkimman SOM kerroksen TS-SOM:sta
random_subsample	Poimii satunnaisen havainnon datasta
SelectCompleteRecords	Poimii täysin havaitut havainnot
VerifyCompleteData	Varmistaa että datojen dimensiot täsmäävät
InitDummyVariableCoding	Alustaa muuttujien luokkien koodauksen
DoListwiseDeletion	Poistaa datasta puutteelliset havainnot
ShowTimeConsumption	Näyttää prosessin ajankulutus graafin