Tilastollisia luokittelumenetelmiä koneelliseen tunnistamiseen – sovellus pohjaeläinaineistoon

Pro gradu -tutkielma
Jyväskylän yliopisto
Matematiikan ja tilastotieteen laitos
15. tammikuuta 2010
Johanna Ärje

Tiivistelmä

Johanna Ärje: *Tilastollisia luokittelumenetelmiä koneelliseen tunnistamiseen – sovellus pohjaeläinaineistoon*
Tilastotieteen pro gradu -tutkielma, Jyväskylän yliopisto, 15. tammikuuta 2010.
Sivuja 34+35.

Pohjaeläimiä käytetään biologisessa seurannassa, jolla tutkitaan ihmistoiminnan vaikutuksia vesistöjen ympäristön tilaan. Perinteisesti pohjaeläimet tunnistetaan manuaalisesti. Tässä työssä tarkastellaan, miten pohjaeläimiä tunnistetaan koneellisesti käyttäen luokittelumenetelmiä, jotka ovat tuottaneet hyviä tuloksia planktoneilla. Pohjaeläinten tapauksessa on tärkeää saavuttaa mahdollisimman tarkat estimaatit lajien suhteellisille osuuksille. Tätä varten tarkastellaan sekaannusmatriisikorjauksena tunnettua menetelmää lajiosuuksien estimaateille.

Pohjaeläimet ovat vesistöjen pohjassa eläviä selkärangattomia eläimiä, jotka reagoivat nopeasti ympäristön muutoksiin. Niiden runsaussuhteiden muutokset kertovat ympäristön tilan muutoksista. Biologinen seuranta on biologisten laatutekijöiden, kuten pohjaeläinten, havainnointia. Biologisessa seurannassa pohjaeläinten havaituista lukumääristä lasketaan useita indeksejä, joita käytetään vesistöjen vertailussa.

Koneellisessa tunnistamisessa tutkittavat kohteet kuvataan tietokoneelle, ja kuvasta segmentoidaan, eli erotellaan, jokainen hahmo muista yksilöistä ja taustasta. Yksilökuvista määritetään yksilön ominaisuuksia kuvaavia piirteitä, joiden avulla hahmot luokitellaan. Tässä työssä keskitytään luokittelumenetelmiin.

Aineistoon sovellettavat luokittelumenetelmät ovat Bayes-luokittelija, päätöspuu ja satunnainen metsä. Lisäksi tarkastellaan vähemmän käytettyä satunnaisen metsän sovellusta Bayes-luokittelijaan. Tämän odotetaan parantavan perinteisen Bayes-luokittelijan robustisuutta ja tarkkuutta.

Luokkakohtaiset luokitteluvirheet aiheuttavat harhaa luokittelun tuloksena saataviin lajiosuuksien estimaatteihin. Tämän vuoksi työssä sovelletaan sekaannusmatriisikorjauksena tunnettua menetelmää näiden estimaattien korjaamiseksi.

Tutkielmassa kaikilla luokittelumenetelmillä saavutetaan erittäin hyviä tuloksia. Bayes-luokittelijan luokitteluvirhe on pienin ja sillä saavutettuja tuloksia esitellään myös käsikirjoituksessa [10].

Erityisesti lajiosuuksien korjaus tuottaa kiinnostavia tuloksia. Bayes-luokittelija toimii jo itsessään hyvin, mutta kaikilla muilla luokittelijoilla sekaannusmatriisikorjauksella saadaan luokittelun tuloksia paremmat estimaatit lajien suhteellisille osuuksille. Etenkin satunnaisen Bayes-metsän korjatuilla estimaateilla saavutetaan kilpailukykyisiä tuloksia Bayes-luokittelijan luokittelun tuloksena saatujen lajiosuuksien estimaattien kanssa.

Avainsanat: Bayes-luokittelija, koneellinen tunnistaminen, luokittelumenetelmä, luokitteluvirhe, piirre, pohjaeläin, päätöspuu, satunnainen Bayes-metsä, satunnainen metsä, sekaannusmatriisikorjaus

Sisältö

Luku 1

Johdanto

Tässä tutkielmassa esitellään hahmontunnistuksessa käytettäviä luokittelualgoritmeja ja sovelletaan niitä Suomen ympäristökeskuksen tuottamaan pohjaeläinaineistoon. Tarkoituksena on tutkia jo olemassa olevien algoritmien toimivuutta pohjaeläinten luokittelussa ja löytää menetelmä, jota voitaisiin jatkossa käyttää pohjaeläinlajien määrittämiseen. Lisäksi tutkitaan, miten luokittelussa tapahtuva virhe vaikuttaa lajien suhteellisten osuuksien arviointiin.

Nykyisin voimassa oleva Euroopan Unionin vesipuitedirektiivi edellyttää jäsenmailta niiden pinta- ja pohjavesien luokittelua ihmistoiminnan aiheuttamien muutosten voimakkuuden perusteella. Luokittelu pohjautuu fysikaalis–kemialliseen sekä biologiseen seurantaan. Fysikaalis–kemiallisiin tekijöihin kuuluvat esimerkiksi ravinteet, kun taas biologisella seurannalla tarkoitetaan biologisten laatutekijöiden havainnointia. Näitä ovat kalat, piilevät, pohjaeläimet sekä kasviplankton. Tässä työssä tarkastellaan pohjaeläimiä. Biologinen seuranta toteutetaan ottamalla näytteitä vesistöistä ja laskemalla biologisista laatutekijöistä niiden havaittuihin lukumääriin perustuvia indeksejä. Näitä indeksejä käytetään jokien, järvien ja rannikkovesien vertailussa sekä eliöstön monimuotoisuuden arvioinnissa.

Pohjaeläimet ovat järvien ja jokien pohjassa eläviä selkärangattomia eläimiä. Niiden lajimääriin ja runsaussuhteisiin vaikuttavat mm. kasvillisuus sekä veden ravinne-, suola- ja happipitoisuus. Biologisessa seurannassa käytetään pohjaeläimiä, koska ne reagoivat nopeasti muutoksiin ja niillä on lajikohtaisia eroja herkkyydessä eri ympäristöpaineisiin. Jotkut lajit ovat herkempiä esimerkiksi ympäristömyrkyille, toiset taas rehevöitymiselle tai veden pH-arvon muutoksille. Näin ollen näytteissä esiintyvien lajien runsaussuhteiden muutokset indikoivat hyvin seurantakohteen ympäristön tilan muutoksia.

Vesistöjen pohjaeläinperustaista luokittelua käsitellään Suomen ympäristökeskuksen luokitteluoppaassa [17]. Sen mukaan jokien pohjaeläinperustainen luokittelu koostuu jokityypille ominaisten taksonien lukumäärästä, jokityypille ominaisten EPT-heimojen lukumäärästä ja yhteisöjen samankaltaisuutta kuvaavasta PMA-indeksistä. Jokityypille ominaisten taksonien lukumäärä on sille ominaisten taksonien, eli sukulaisuussuhteen mukaan nimettyjen eliöryhmien havaittu lukumäärä. Se kuvaa jokityypin taksonomista monimuotoisuutta ja on suora johdannainen maailmalla yleisesti

käytetystä havaittujen ja ennustettujen taksonien suhteesta. Tyyppiominaiset EPT-heimot ovat kullekin jokityypille ominaisia päivänkorento-, koskikorento- ja vesiperhosheimoja. Niiden havaitulla lukumäärällä voidaan kuvata tärkeiden taksonomisten ryhmien puuttumista. Lisäksi EPT-heimot ovat herkkiä erilaisille ympäristöpaineille, mikä tekee niistä keskeisen muuttujan monissa indekseissä. Yhteisöjen suhteellista samankaltaisuutta ilmentävä PMA-indeksi (Percent Model Affinity) taas kuvaa pohjaeläinlajiston koostumusta ja runsaussuhteita. Sillä voidaan myös kuvata muutoksia, joissa lajimäärä kasvaa ympäristön tilan muutoksen seurauksena.

Järvien pohjaeläinlajiston tilaa kuvataan PMA-indeksin lisäksi syvänteiden surviaissääskitoukkien esiintymiseen perustuvalla pohjanlaatu- eli BQI-indeksillä (Benthic Quality Index). BQI-indeksissä pisteytetään seitsemän surviassääskilajia niiden kuormituksen sietokyvyn mukaan, ja indeksi ilmaisee lajien runsauksilla painotetun indikaattoripisteiden keskiarvon.

Rannikkovesien pohjaeläinperustainen luokittelu tehdään BBI-indeksin (Brackish water Benthic Index) avulla. Se perustuu pehmeiden pohjien pohjaeläinnäytteisiin ja käyttää lajien lukumääriä, runsaustietoja sekä pistearvoja eri lajien ympäristöpaineiden sietokyvystä. Indeksi on sovitettu Itämeren olosuhteisiin, ja se huomioi rannikkovesiemme luonnostaankin alhaisen lajiston monimuotoisuuden.

Ympäristöhallinnon toimesta Suomen sisävesistä tehtävä pohjaeläinseuranta käsittää noin 6000 näytettä vuodessa. EU:n vesipuitedirektiivi vaatii vielä laajempaa seurantaa kalojen, levien ja pohjaeläinten osalta. Pohjaeläinten luokittelussa ongelmana on, että lajit joudutaan tunnistamaan näytteistä manuaalisesti. Tämä vaatii asiantuntijatyötä ja on hyvin aikaavievää sekä kallista. Yhden näytteen sisältämien pohjaeläinten lajien määrittämiseen asiantuntijalta kuluu aikaa yhdestä neljään tuntia. Jos lajeja pystyttäisiin tunnistamaan koneellisesti, se mahdollistaisi laajemman biologisen seurannan, jolloin lajien osuuksien arvioinnista tulisi tarkempaa.

Koneellista tunnistamista on sovellettu jo monella alalla, mm. lääketieteessä sekä biologisessa seurannassa planktoneiden luokitteluun, mistä on saatu hyviä tuloksia ([2], [12]). Pohjaeläimiin menetelmää ei ole vielä laajalti sovellettu.

Koneellisen tunnistamisen vaiheet ovat kuvaus, segmentointi, piirteiden määrittäminen ja luokittelu. Pohjaeläinten tapauksessa kuvaaminen suoritetaan siten, että kuvausvaiheessa pohjaeläimet asetetaan vesimaljaan ja skannataan tietokoneelle. Näin saadaan digitaalisessa muodossa oleva kaksiulotteinen harmaasävykuva. Kuvaustavasta johtuen saman lajin edustajista saadaan monia erilaisia kuvia. Pohjaeläimet voivat olla kuvassa hyvin erilaisissa asennoissa ja osalta voi puuttua raajoja. Kuvasta normalisoidaan tausta ja erotellaan, eli segmentoidaan, jokainen pohjaeläin omaksi kuvakseen. Segmentoinnin tarkoituksena on eristää jokainen hahmo mahdollisimman hyvin muista hahmoista ja taustasta.

Yksittäisten pohjaeläinten kuvista määritetään joukko muuttujia, joita kutsutaan piirteiksi. Havainnosta lasketut piirteiden arvot kuvaavat yksilön eri ominaisuuksia. Piirteiden avulla yhteen yksilöön liittyvä aineisto tiivistetään kokonaisesta kuvasta piirrevektoriksi. Tavoitteena on kuvailla hahmoa piirteillä, jotka saavat lähellä toisiaan olevia arvoja saman luokan havainnoilla ja kaukana toisistaan olevia arvoja eri luokkien havainnoilla. Piirteiden tulisi olla havaintoja hyvin erottelevia, robusteja ko-

hinalle, ja niiden pitäisi olla invariantteja epäolennaisten muunnosten suhteen ([4]). Tämä tarkoittaa, että esimerkiksi sillä ei saisi olla väliä, miten päin hahmo kuvassa on. Epäolennaisten muunnosten suhteen invariantteja piirteitä ovat muun muassa muotoa ja väriä kuvaavat piirteet. Mitkä piirteet ovat hyödyllisiä, riippuu tietysti luokitteluongelmasta. Alan asiantuntemuksella voidaan helpottaa piirteiden valintaa.

Luokitteluvaiheessa tavoitteena on löytää aineistolle esitystapa, jolla eri luokkien havainnot erotetaan toisistaan. Luokittelija päättää hahmon luokan piirrevektorin avulla. Luokittelun vaikeus riippuu siitä, miten suurta luokkien sisäinen vaihtelu on verrattuna luokkien väliseen vaihteluun. Kun piirteisiin yhdistetään tieto lajista, saadaan aineisto, jota voidaan käyttää luokittelualgoritmin opettamiseen ja toimivuuden arviointiin. Aineiston lajitiedot saadaan asiantuntijoilta, jotka tunnistavat yksilöt manuaalisesti. Aineiston käyttöä luokittelijan muodostamiseksi kutsutaan opettamiseksi. Luokittelija oppii esimerkkihavainnoista, jotka muodostavat opetusaineiston. Luokkien taustalla olevien mallien tuntemattomat parametrit estimoidaan opetusaineiston avulla. Arviointivaiheessa testiaineiston avulla testataan luokittelijan toimivuutta ja kehittämistarvetta. Arviointi suoritetaan luokitteluvirheen perusteella, joka saadaan esimerkiksi laskemalla testiaineistoksi kutsuttujen havaintojen väärinluokiteltujen yksilöiden osuus.

Tässä työssä sovelletaan planktoneiden tunnistamisessa hyväksi havaittuja menetelmiä pohjaeläimiin ja tutkitaan niiden toimivuutta. Aineistoon sovellettavat luokittelumenetelmät ovat Bayes-luokittelija, päätöspuu, satunnainen metsä ja satunnainen Bayes-metsä. Bayes-luokittelija luokittelee havainnot niiden piirteiden jakaumien perusteella. Päätöspuu perustuu aineiston osajoukkoihin jakamiseen. Satunnainen metsä koostuu satunnaisista päätöspuista, jotka äänestävät havainnoille luokkaa. Satunnainen Bayes-metsä on satunnaisen metsän sovellus Bayes-luokittelijaan.

Tavoitteena on myös arvioida, miten luokittelussa tapahtuva virhe vaikuttaa lajien runsaussuhteiden arviointiin. Tässä yhteydessä esitellään sekaannusmatriisikorjaus lajiosuuksien estimaateille sekä kerrotaan, miten lajien runsaussuhteiden eri estimaatteja voidaan vertailla keskenään. Tarkoituksena on vertailla eri luokittelumenetelmiä sekä perinteisellä luokitteluvirheellä että sen perusteella, miten sekaannusmatriisikorjaus lajiosuuksille toimii eri luokittelijoilla. Näin laajaa tutkimusta esitettyjen menetelmien vertaamiseksi ei tietääksemme ole pohjaeläimillä tehty. Bayes-luokittelijaa ja sillä saavutettuja hyviä tuloksia käsitellään myös julkaisua odottavassa käsikirjoituksessa Classification and retrieval on macroinvertabrate image databases [10].

Luvussa 2 esitellään käytettävissä oleva aineisto. Alustavaan tutkimukseen valittujen lajien lisäksi käydään läpi pohjaeläinten kuvista irrotetut piirteet. Luvussa 3 esitellään aineistoon sovellettavat luokittelualgoritmit. Tässä luvussa tarkastellaan myös, miten eri luokittelumenetelmiä voidaan vertailla keskenään luokitteluvirheen avulla. Luvussa 4 pohditaan, miten luokitteluvirheen vaikutusta lajiosuuksien estimointiin voidaan arvioida ja korjata. Luvussa esitellään korjatut estimaatit lajiosuuksille ja pohditaan, miten niiden toimivuutta voidaan vertailla luokittelun tuloksena saatujen osuuksien estimaattien kanssa. Luvussa 5 esitellään aineiston luokittelusta saadut tulokset ja vertaillaan eri luokittelumenetelmiä keskenään luokitteluvirheen ja lajiosuuksien estimaattien perusteella. Luvussa 6 kerrataan saavutetut tulokset ja

pohditaan, mitä kaikkea olisi syytä tutkia syvemmin.

Luku 2

Tutkimusaineisto

Tässä luvussa esitellään työssä käytettävä tutkimusaineisto. Ensin esitellään pohjaeläinten taksonomiset luokat, joihin havaintoja pyritään luokittelemaan. Tämän jälkeen käydään läpi luokittelussa käytettävät piirteet.

2.1 Lajit

Suomen Ympäristökeskuksen tuottama aineisto sisältää yhteensä 1350 pohjaeläintä. Alustavaan tutkimukseen on valittu herkimmin ympäristön tilan muutoksiin reagoivia lajeja. Aineistossa ovat seuraavat kahdeksan eri taksonomista luokkaa: *Baetis rhodani, Diura nanseni, Heptagenia sulphurea, Hydropsyche pellucidulla, Hydropsyche siltalai, Isoperla sp., Rhyacophila nubila* ja *Taeniopteryx nebulosa* (Kuva 2.1.1). Näistä *Isoperla sp.* on suku ja muut lajeja.

Kuva 2.1.1: Pohjaeläinten taksonomiset luokat. Yläreunassa ovat *Baetis rhodani, Diura nanseni, Heptagenia sulphurea* ja *Hydropsyche pellucidulla.* Alarivillä ovat *Hydropsyche siltalai, Isoperla sp., Rhyacophila nubila* ja *Taeniopteryx nebulosa.*

Asiantuntijat ovat tunnistaneet yksilöt manuaalisesti. Tämän jälkeen pohjaeläimet on kuvattu lajeittain. Kuvaaminen tapahtuu siten, että yksilöt asetetaan vesimaljaan, josta skannataan tietokoneelle kaksiulotteinen harmaasävykuva. Tästä ns.

tarjotinkuvasta segmentoidaan jokainen pohjaeläin omaksi kuvakseen. Tarkoituksena on eristää yksittäinen pohjaeläin mahdollisimman hyvin muista pohjaeläimistä sekä taustasta.

2.2 Piirteet

Piirteellä tarkoitetaan yksilön ominaisuutta kuvaavaa muuttujaa. Piirteet voivat olla jatkuvia numeerisia muuttujia tai diskreettejä luokitteluasteikollisia muuttujia. Tässä työssä pohjaeläimiin liittyvät piirteet määritetään kuhunkin pohjaeläimeen liittyvästä harmaasävykuvasta. Piirteitä voidaan laskea kuvan harmaasävyarvoista, jolloin ne kuvaavat yksilön väriä. Esimerkiksi keskiarvolla tarkoitetaan pohjaeläimeen liittyvien harmaasävyarvojen keskiarvoa. Vastaavasti geometriset piirteet kuvaavat pohjaeläimen muotoon liittyviä ominaisuuksia. Kuvankäsittelyohjelmalla voidaan irrottaa kuvista seuraavat harmaasävy- ja geometriset piirteet:

- keskiarvo (Mean)
- keskihajonta (StDev)
- moodi (Mode)
- mediaani (Median)
- summa (IntDen)
- vinous (Skewness)
- huipukkuus (Kurtosis)
- pinta-ala (Area)
- ympärysmitta (Perimeter)
- pienimmän sellaisen suorakulmion leveys, jonka sisään pohjaeläin mahtuu (Width)
- pienimmän sellaisen suorakulmion korkeus, jonka sisään pohjaeläin mahtuu (Height)
- pisin mitta kahden pisteen välillä, jotka sijaitsevat pohjaeläimen ääriviivoilla (Feret)
- pienimmän sellaisen ellipsin, jonka sisään pohjaeläin mahtuu, pidempi halkaisija (Major)
- pienimmän sellaisen ellipsin, jonka sisään pohjaeläin mahtuu, lyhyempi halkaisija (Minor)
- ympyrämäisyysarvo, joka kertoo, onko pohjaeläin ympyrän vai pitkänomaisen monikulmion mallinen (Circularity)

Näistä seitsemän ensimmäistä ovat harmaasävyarvoista määritettyjä suureita ja loput kuvaavat geometrista muotoa. Tarkempi kuvaus piirteistä löytyy ImageJ-ohjelman kotisivulta [8].

Piirteiden avulla voidaan päätellä mm., miten tumma pohjaeläin on kyseessä, miten sen väritys vaihtelee tai minkä kokoinen pohjaeläin on. Saman lajin edustajien kuvista voidaan estimoida kunkin piirteen lajikohtainen jakauma. Luokittelualgoritmille voidaan opettaa, millainen piirteen jakauma kullakin pohjaeläinlajilla on, ja tätä tietoa käytetään uusien havaintojen luokittelussa.

Luku 3

Luokittelumenetelmät

Tässä luvussa esitellään erilaisia luokittelualgoritmeja. Ensimmäisenä esitellään Bayes-luokittelija, joka perustuu piirteiden jakaumiin. Seuraavaksi käsitellään päätöspuuta, joka perustuu aineiston osajoukkoihin jakamiseen. Tämän jälkeen esitellään satunnainen metsä, joka koostuu satunnaisista päätöspuista, ja lisäksi sovelletaan satunnaisen metsän menetelmää Bayes-luokittelijaan. Lopuksi tarkastellaan tapoja luokittelumenetelmien vertailuun.

Luokittelijoiden muodostamista varten tarvitaan aineisto, jossa havaintojen piirteet ja luokat ovat tiedossa. Tätä kutsutaan opetusaineistoksi. Opetusaineistossa olevien piirteiden avulla muodostetaan luokittelusääntöjä. Näiden sääntöjen avulla voidaan jatkossa luokitella havaintoja, joiden luokat eivät ole tiedossa.

3.1 Bayes-luokittelija

Bayes-luokittelija luokittelee uuden havainnon ennalta tunnettuun luokkaan opetusaineistosta muodostetun luokittelufunktion avulla. Luokittelufunktio perustuu luokkien prioritodennäköisyyksiin ja luokkiin liittyviin tiheysfunktioihin. Seuraava teoriaesitys on kirjoitettu seuraten kirjan *Pattern Classification* [4] esitystä.

Olkoot $\mathbf{x} = (x_1, \ldots, x_p)'$ piirrevektori, $\omega_1, \ldots, \omega_k$ mahdolliset luokat ja $P(\omega_1) = \pi_1, \ldots, P(\omega_k) = \pi_k$, näiden luokkien prioritodennäköisyydet siten, että $\sum_{i=1}^{k} \pi_i = 1$. Oletetaan vielä, että luokan ω_i tiheysfunktio on $f_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)$. Tällöin posterioritodennäköisyys luokalle ω_i on

$$P(\omega_i|\mathbf{x}) = \frac{P(\omega_i)p(\mathbf{x}|\omega_i)}{p(\mathbf{x})} = \frac{\pi_i f_i(\mathbf{x})}{\sum_{l=1}^{k} \pi_l f_l(\mathbf{x})}.$$

Jos luokittelussa on käytettävissä vain tieto prioritodennäköisyyksistä, yksilö luokitellaan aina luokkaan, jonka prioritodennäköisyys on suurin. Tällöin virheellisen luokittelun todennäköisyys on pienin mahdollinen. Kun lisäksi havaitaan piirrevektori $\mathbf{x}$, päästään prioritodennäköisyydestä posterioritodennäköisyyteen $P(\omega_i|\mathbf{x})$. Bayesin päätössäännön mukaan yksilö luokitellaan luokkaan ω_i, jos

$$P(\omega_i|\mathbf{x}) > P(\omega_j|\mathbf{x}) \tag{3.1.1}$$

kaikille $j \neq i$. Tämä päätössääntö minimoi luokitteluvirheen todennäköisyyden.

3.1.1 Bayesin päätössäännön muodostaminen

Päätöstä luokitella yksilö johonkin tiettyyn luokkaan voidaan ajatella toimintona, samoin päätöstä jättää joku yksilö luokittelematta. Näin ollen toimintojen määrä voi olla suurempi kuin luokkien määrä. Merkitään tätä äärellistä toimintojen joukkoa $\{\alpha_1, \ldots, \alpha_a\}$.

Otetaan käyttöön tappiofunktio $\lambda(\alpha_i|\omega_j)$, joka ilmaisee toiminnosta α_i seuraavan tappion, kun yksilön todellinen luokka on ω_j. Näin valitsemalla toiminnon α_i, kun on havaittu luokasta ω_j peräisin oleva piirrevektori $\mathbf{x}$, aiheutamme tappion $\lambda(\alpha_i|\omega_j)$. Koska todennäköisyys, että oikea luokka on ω_j, on $P(\omega_j|\mathbf{x})$, odotettu tappio toiminnolle α_i on

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{k} \lambda(\alpha_i|\omega_j) P(\omega_j|\mathbf{x}). \tag{3.1.2}$$

Tätä kutsutaan ehdolliseksi riskiksi. Kun havaitaan piirrevektori $\mathbf{x}$, odotettu tappio voidaan minimoida valitsemalla toiminto, joka minimoi ehdollisen riskin (3.1.2). Tavoitteena on löytää päätössääntö, joka minimoi ns. kokonaisriskin huomioiden kaikki mahdolliset piirrevektorin $\mathbf{x}$ arvot.

Olkoon $\alpha(\mathbf{x})$ päätössääntö, joka antaa jonkin arvon $\alpha_i, \ldots, \alpha_a$ jokaiselle $\mathbf{x}$. Päätös riippuu piirrevektorista $\mathbf{x}$, toisin sanoen $\alpha : \mathbf{x} \mapsto \{\alpha_i\}$ on kuvaus. Kun käytetään tiettyä päätössääntöä $\alpha(\mathbf{x})$, odotettua tappiota kutsutaan kokonaisriskiksi. Koska $R(\alpha_i|\mathbf{x})$ on toimintoon α_i liittyvä ehdollinen riski, ja koska päätössääntö määrää toiminnon, kokonaisriski on

$$R = \int R(\alpha(\mathbf{x})|\mathbf{x}) p(\mathbf{x}) \mathrm{d}\mathbf{x},$$

missä $\alpha(\mathbf{x})$ on funktio, joka kertoo toiminnon kullakin $\mathbf{x}$. Kokonaisriski minimoituu, kun valitaan päätössääntö $\alpha(\mathbf{x})$, joka minimoi $R(\alpha_i(\mathbf{x}))$:n kaikille $\mathbf{x}$. Kokonaisriskin minimoimiseksi lasketaan ehdolliset riskit $R(\alpha_i|\mathbf{x})$, $i = 1, \ldots, a$ ja valitaan toiminto α_i, jolla $R(\alpha_i|\mathbf{x})$ on pienin. Näin saatavaa pienintä mahdollista kokonaisriskiä kutsutaan Bayes-riskiksi. Jos minimi saavutetaan usealla α_i, ei ole väliä, mikä näistä toiminnoista valitaan.

Etsittäessä päätössääntöä, joka minimoi luokitteluvirheen todennäköisyyden, valitaan 0–1 -tappiofunktio

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0, & \text{kun } i = j \\ 1, & \text{kun } i \neq j \end{cases}. \tag{3.1.3}$$

Ehdollinen riski on tällöin

$$\begin{aligned} R(\alpha_i|\mathbf{x}) &= \sum_{j=1}^{k} \lambda(\alpha_i|\omega_j) P(\omega_j|\mathbf{x}) \\ &= \sum_{j \neq i} P(\omega_j|\mathbf{x}) \\ &= 1 - P(\omega_i|\mathbf{x}), \end{aligned}$$

missä $P(\omega_i|\mathbf{x})$ on ehdollinen todennäköisyys, että toiminto α_i on oikea, ts. $R(\alpha_i|\mathbf{x})$ on luokitteluvirheen todennäköisyys. Vastaavasti kokonaisriski on nyt täsmälleen keskimääräinen luokitteluvirheen todennäköisyys. Jotta keskimääräinen virheen todennäköisyys minimoituisi, on valittava i, joka maksimoi posteriorin $P(\omega_i|\mathbf{x})$.

Kun luokkia on enemmän kuin kaksi, otetaan käyttöön luokittelufunktiot $g_i(\mathbf{x})$, $i = 1, \ldots, k$. Tässä tapauksessa luokittelija on koneisto, joka laskee kaikki k luokittelufunktiota ja valitsee luokan, jonka luokittelufunktio saa suurimman arvon piirrevektorilla $\mathbf{x}$. Nyt Bayes-luokittelija saadaan yleisessä tapauksessa asettamalla

$$g_i(\mathbf{x}) = -R(\alpha_i|\mathbf{x}),$$

koska silloin suurin luokittelufunktio vastaa pienintä ehdollista riskiä. Jos halutaan minimoida luokitteluvirheen todennäköisyys, tilanne yksinkertaistuu. Voidaan asettaa

$$g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}),$$

jolloin suurin luokittelufunktio vastaa suurinta posterioritodennäköisyyttä. Koska $P(\omega_i|\mathbf{x}) = \frac{P(\omega_i)p(\mathbf{x}|\omega_i)}{p(\mathbf{x})}$ ja $p(\mathbf{x})$ on sama kaikille luokille, voidaan luokittelufunktioksi asettaa

$$g_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)P(\omega_i).$$

Koska logaritmi on monotonisesti kasvava, myös asettamalla

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|\omega_i) + \ln P(\omega_i)$$

päädytään samaan tulokseen.

3.1.2 Normaalijakautuneet piirteet

Jos $\mathbf{X}|\omega_i \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, pienin luokitteluvirheen todennäköisyys saavutetaan asettamalla luokittelufunktioksi

$$\begin{aligned} g_i(\mathbf{x}) &= \ln p(\mathbf{x}|\omega_i) + \ln P(\omega_i) \\ &= -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)' \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) - \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i| + \ln P(\omega_i). \quad (3.1.4) \end{aligned}$$

Jos $\mathbf{\Sigma}_i = \sigma^2\mathbf{I}$ kaikille i, luokittelufunktio (3.1.4) saa muodon

$$\begin{aligned} g_i(\mathbf{x}) &= -\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu}_i)'(\mathbf{x}-\boldsymbol{\mu}_i) + \ln P(\omega_i) \\ &= -\frac{1}{2\sigma^2}(\mathbf{x}'\mathbf{x} - 2\boldsymbol{\mu}_i'\mathbf{x} + \boldsymbol{\mu}_i'\boldsymbol{\mu}_i) + \ln P(\omega_i). \end{aligned} \tag{3.1.5}$$

Koska $\mathbf{x}'\mathbf{x}$ on sama kaikille i, saadaan lineaariset luokittelufunktiot

$$g_i(\mathbf{x}) = \mathbf{w}_i'\mathbf{x} + w_{i0}, \tag{3.1.6}$$

missä

$$\begin{aligned} \mathbf{w}_i &= \frac{1}{\sigma^2}\boldsymbol{\mu}_i, \\ w_{i0} &= -\frac{1}{2\sigma^2}\boldsymbol{\mu}_i'\boldsymbol{\mu}_i + \ln P(\omega_i). \end{aligned}$$

Tälläista luokittelijaa, joka käyttää lineaarista luokittelufunktiota, kutsutaan lineaariseksi koneistoksi. Jos prioritodennäköisyydet ovat yhtäsuuret kaikilla luokilla, optimaalinen päätössääntö saadaan laskemalla piirrevektorin $\mathbf{x}$ Euklidiset etäisyydet luokkien keskiarvovektoreista $\boldsymbol{\mu}_i$ ja luokittelemalla näin lähimpänä olevaan luokkaan.

Jos $\mathbf{\Sigma}_i = \mathbf{\Sigma}$ kaikille i, luokittelufunktio (3.1.4) saa muodon

$$\begin{aligned} g_i(\mathbf{x}) &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)'\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + \ln P(\omega_i) \\ &= -\frac{1}{2}(\mathbf{x}'\mathbf{\Sigma}^{-1}\mathbf{x} - 2(\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_i)'\mathbf{x} + \boldsymbol{\mu}_i'\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_i) + \ln P(\omega_i). \end{aligned} \tag{3.1.7}$$

Koska termi $\mathbf{x}'\mathbf{\Sigma}^{-1}\mathbf{x}$ ei riipu i:stä, se voidaan jättää pois. Näin saadaan jälleen lineaariset luokittelufunktiot

$$g_i(\mathbf{x}) = \mathbf{w}_i'\mathbf{x} + w_{i0}, \tag{3.1.8}$$

missä

$$\begin{aligned} \mathbf{w}_i &= \mathbf{\Sigma}^{-1}\boldsymbol{\mu}_i, \\ w_{i0} &= -\frac{1}{2}\boldsymbol{\mu}_i'\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln P(\omega_i). \end{aligned}$$

Jos prioritodennäköisyydet ovat samat kaikilla luokilla, optimaalinen päätössääntö saadaan laskemalla Mahalanobis-etäisyydet $(\mathbf{x}-\boldsymbol{\mu}_i)'\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)$ ja luokittelemalla yksilö näin lähimpänä olevaan luokkaan. Tässä tilanteessa lineaarinen luokittelusääntö tunnetaan myös nimellä Fisherin luokittelusääntö ([9]).

Muissa tapauksissa, eli kun $\mathbf{\Sigma}_i$ on mielivaltainen, saadaan kvadraattiset luokittelufunktiot

$$g_i(\mathbf{x}) = \mathbf{x}'\mathbf{W}_i\mathbf{x} + \mathbf{w}_i'\mathbf{x} + w_{i0}, \tag{3.1.9}$$

missä

$$\begin{aligned} \mathbf{W}_i &= -\frac{1}{2}\mathbf{\Sigma}_i^{-1}, \\ \mathbf{w}_i &= \mathbf{\Sigma}_i^{-1}\boldsymbol{\mu}_i, \\ w_{i0} &= -\frac{1}{2}\boldsymbol{\mu}_i'\mathbf{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}\ln|\mathbf{\Sigma}_i| + \ln P(\omega_i). \end{aligned}$$

3.1.3 Parametrien estimointi

Bayes-luokittelija koostuu posterioritodennäköisyyksistä, joiden laskemiseen tarvitaan luokkakohtaiset tiheysfunktiot ja luokkien prioritodennäköisyydet. Usein nämä eivät kuitenkaan ole tiedossa. Jos tiheysfunktion muoto on tunnettu, ongelma yksinkertaistuu parametrien estimointiin. Tähän on monia tapoja, joista tässä esitellään suurimman uskottavuuden menetelmä.

Suurimman uskottavuuden menetelmä

Suurimman uskottavuuden menetelmässä estimoitavien parametrien oletetaan olevan kiinnitettyjä, mutta tuntemattomia. Estimaattori maksimoi todennäköisyyden saada havaitun kaltainen aineisto ja konvergoi melko lievin ehdoin kohti oikeaa arvoa aineiston kasvaessa ([4]).

Opetusaineisto voidaan jakaa luokkien mukaan osiin $D_1, \ldots, D_k$. Oletetaan, että parametrivektori $\boldsymbol{\theta}_j$ määrää yksikäsitteisesti luokkakohtaisen tiheysfunktion $p(\mathbf{x}|\omega_j)$ muodon. Voidaan siis kirjoittaa $p(\mathbf{x}|\omega_j) = p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$. Oletetaan myös, etteivät joukon D_i havainnot anna informaatiota $\boldsymbol{\theta}_j$:stä, kun $i \neq j$. Tällöin eri luokkien parametrit ovat funktionaalisesti toisistaan riippumattomat, ja jokaisen luokan parametrivektori voidaan estimoida erikseen. Näin ollen ratkaistavaksi jää k samanlaista ongelmaa. Seuraavaksi esitetään, miten yksi tällainen ongelma voidaan ratkaista.

Jos joukko D sisältää n riippumatonta piirrevektoria $\mathbf{x}_1, \ldots, \mathbf{x}_n$, saadaan uskottavuusfunktio

$$L(\boldsymbol{\theta}) = p(D|\boldsymbol{\theta}) = \prod_{l=1}^{n} p(\mathbf{x}_l|\boldsymbol{\theta})$$

ja logaritminen uskottavuusfunktio

$$l(\boldsymbol{\theta}) = \ln p(D|\boldsymbol{\theta}).$$

Koska logaritmi on monotonisesti kasvava ja $\hat{\boldsymbol{\theta}}$ maksimoi logaritmisen uskottavuusfunktion, se maksimoi myös uskottavuusfunktion. Suurimman uskottavuuden estimaatti $\hat{\boldsymbol{\theta}}$ on estimaatti, joka parhaiten tukee havaittua aineistoa. Logaritmista muotoa käytetään, koska sitä on usein helpompi käsitellä kuin alkuperäistä uskottavuusfunktiota.

Jos parametrivektori on muotoa $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_p]'$, merkitään differentiaalioperaattoria symbolilla $\bigtriangledown_{\boldsymbol{\theta}} = [\frac{\partial}{\partial \theta_1}, \ldots, \frac{\partial}{\partial \theta_p}]'$. Nyt suurimman uskottavuuden estimaatti on $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta})$. Tämä saadaan ratkaistua yhtälöistä

$$\bigtriangledown_{\boldsymbol{\theta}} l \;=\; \mathbf{0},$$

missä

$$\bigtriangledown_{\boldsymbol{\theta}} l \;=\; \sum_{l=1}^{n} \bigtriangledown_{\boldsymbol{\theta}} \ln p(\mathbf{x}_l|\boldsymbol{\theta}).$$

Normaalijakauman tapauksessa suurimman uskottavuuden estimaatit luokan ω_i odotusarvolle ja kovarianssimatriisille ovat

$$\hat{\boldsymbol{\mu}}_i \;=\; \frac{1}{n}\sum_{l=1}^{n} \mathbf{x}_l$$

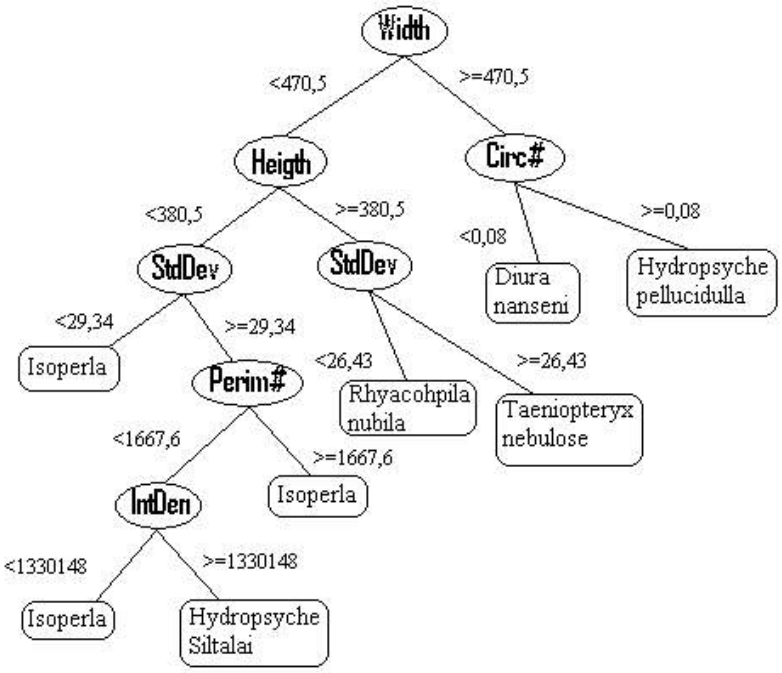
ja

$$\hat{\boldsymbol{\Sigma}}_i \;=\; \frac{1}{n}\sum_{l=1}^{n} (\mathbf{x}_l - \hat{\boldsymbol{\mu}}_i)(\mathbf{x}_l - \hat{\boldsymbol{\mu}}_i)'.$$

3.2 Päätöspuu

Päätöspuu on luokittelualgoritmi, jossa aineistoa pilkotaan piirteiden avulla osajoukkoihin, kunnes jokainen osajoukko sisältää vain samaan luokkaan kuuluvia havaintoja ([18]). Päätöspuu koostuu solmuista, oksista ja lehdistä. Solmuiksi valitaan havaintoja hyvin luokittelevia piirteitä. Oksat sisältävät aineiston osajoukkoja ja lehdet nimetään luokkamuuttujan luokkien mukaan.

Tarkastellaan ensin päätöspuun muodostamista tilanteessa, jossa luokittelussa käytetyt piirteet ovat luokitteluasteikollisia tai luokiteltuja. Päätöspuun muodostaminen alkaa juurisolmusta (root node). Juurisolmuksi valitaan piirre, joka jakaa opetusaineiston mahdollisimman hyvin tutkittavan luokkamuuttujan mukaisiin luokkiin (ks. Kuva 3.2.1). Aineisto jaetaan juurisolmun eri arvojen mukaan, ja nämä osajoukot muodostavat puun ensimmäiset oksat. Jokaisesta oksasta alkaa uusi osapuu. Menettely toistetaan jokaiselle osapuulle, kunnes uloimmilla oksilla on vain samaan luokkaan kuuluvia havaintoja. Tällaisen ns. puhtaan oksan päähän kasvatetaan lehti, joka nimetään oksalla olevien havaintojen luokan mukaan. Jos oksalla ei ole yhtään havaintoa, nimetään lehti sen mukaan, mikä on yleisin luokka aiemmalla oksalla. Kun



Kuva 3.2.1: Päätöspuu, jossa juurisolmuna leveys (Width). Diskretisointi tehty etsimällä sopiva raja jokaiselle muuttujalle erikseen.

päätöspuu on valmis, käydään tulevilla havainnoilla koko puu läpi ja luokitellaan ne sen mukaan, mille lehdelle ne päätyvät.

Päätöspuun etuna on, että siinä voidaan käyttää myös laadullisia, luokitteluasteikollisia piirteitä. Jos tutkittavassa aineistossa on jatkuvia piirteitä, kuten tämän tutkielman aineistossa, on ne joidenkin päätöspuiden tapauksessa ensin diskretisoitava. Tämän voi tehdä järjestämällä havainnot muuttujan arvojen mukaisesti ja jakamalla arvot väleiksi sen mukaan, milloin havaintojen luokka muuttuu. Jos jako väleihin tehdään ennen puun muodostamista, kyseessä on globaali diskretisointi. Jos jako tehdään vasta solmumuuttujaa valittaessa, on kyse lokaalista diskretisoinnista. Jakoa ei ole välttämätöntä tehdä luokkien määrän mukaan. Tällöin osa piirteen sisältämästä tiedosta voi jäädä hyödyntämättä ja samaa piirrettä voidaan käyttää päätöspuussa useammin kuin kerran.

Joskus voi käydä niin, että muodostettu päätöspuu sopii opetusaineistoon liian hyvin. Jos uloimmilla oksilla on vain vähän havaintoja, luokittelu perustuu liian tarkasti yksittäisiin opetusaineiston havaintoihin, eikä toimi hyvin uuden, riippumattoman aineiston kanssa. Tällöin puuta täytyy karsia. Karsiminen voidaan tehdä niin, että jotkin osapuut korvataan lehdillä. Lehden luokka määräytyy osapuun havaintojen enemmistön perusteella. Monimutkaisemmassa karsimisessa ylempi osapuu voidaan korvata sen alemmalla osapuulla. Tällöin osa päätöspuusta joudutaan rakentamaan uudestaan. Karsimisen seurauksena puussa voi olla epäpuhtaita lehtiä, joilla on useamman luokan havaintoja. Tällöin lehti nimetään sillä olevien havaintojen yleisimmän luokan mukaan.

Juurisolmun ja muiden solmujen solmumuuttujaksi valitaan se piirre, joka jakaa aineiston parhaiten luokkamuuttujan luokkiin informaatiohyödyn (Information Gain) mielessä. Informaatiohyöty perustuu entropiaan [16], jota voidaan pitää hajaannuksen tai epäjärjestyksen mittana. Sen voi myös ajatella kuvaavan multinomijakauman heterogeenisuutta eli sitä, miten tasaisesti havainnot jakautuvat eri luokkiin. Matemaattisemmin, muuttujan Y entropia on

$$H(Y) = -\mathbb{E}(\log(p(Y))),$$

missä Y on diskreetti satunnaismuuttuja ja $p(y) = P(Y = y)$. Jos $P(Y = y_i) = p_i \geq 0$ kaikilla $i = 1, \ldots, k$ ja $\sum_{i=1}^{k} p(y_i) = 1$, niin

$$H(Y) = H(p_1, \ldots, p_k) = -\sum_{i=1}^{k} p_i \log(p_i).$$

Entropian maksimiarvo on $H_{max} = \ln k$, missä k on muuttujan Y mahdollisten luokkien määrä. Havaintojen tasajakautuneisuutta kuvataan yleensä entropiasuhteella $H(Y)/H_{max}$. Jos kaikki havainnot kuuluvat samaan luokaan, on entropiasuhde 0. Jos taas kaikissa luokissa on yhtä monta havaintoa, saa entropiasuhde arvon 1. Entropia siis kasvaa sen mukaan, miten hyvin havainnot ovat hajaantuneet eri luokkiin. Jos logaritmin kanta on luokkien määrä k, entropia $H(Y) \in [0, 1]$ aina. Yleisesti käytetyt kannat ovat 2, 10 ja e.

Ehdollinen entropia määritellään seuraavasti:

$$H(Y|X) = \sum_{j=1}^{m} p_{\cdot j} H(Y|X = x_j),$$

missä $H(Y|X = x_j) = -\sum_{i=1}^{k} P(Y = y_i|X = x_j) \log(P(Y = y_i|X = x_j))$ ja $p_{\cdot j} = \sum_{i=1}^{k} p_{ij}$.

Kun tiedetään tutkittavan luokkamuuttujan Y entropia ja sen ehdollinen entropia ehdolla X, voidaan laskea mitattavan piirteen X informaatiohyöty

$$G(Y, X) = H(Y) - H(Y|X).$$

Informaatiohyötyä voidaan käyttää useilla eri tavoilla, kun päätöspuuta muodostetaan. Esimerkiksi Id3-päätöspuun ([14]) algoritmissa solmumuuttujaksi valitaan piirre, jolla on suurin informaatiohyöty. Id3-päätöspuu on tarkoitettu luokitteluasteikollisille piirteille ja edellyttää jatkuvien piirteiden diskretisointia. Sitä ei myöskään karsita, jolloin ylisovittumisongelmaa voi esiintyä. Tätä ongelmaa on korjattu Id3-päätöspuun seuraajassa, C4.5-päätöspuussa. Se osaa käsitellä myös jatkuvia piirteitä ja käyttää informaatiohyödyn sijaan suhteutettua hyötyä [14], jossa informaatiohyöty jaetaan X:n entropialla:

$$\text{Gain ratio}(X) = \frac{G(Y, X)}{H(X)}.$$

Logaritmin kanta pidetään samana koko ajan. C4.5 käyttää karsintaa, jossa osapuu voidaan korvata sen suosituimmalla oksalla. Karsinta perustuu jakojen tilastollisiin merkitsevyyksiin.

Satunnaisessa päätöspuussa jokaisen oksan kohdalla arvotaan m piirrettä, $m \leq M$, missä M on kaikkien piirteiden lukumäärä. Tästä m piirteen joukosta valitaan parhaiten oksan havaintoja luokitteleva piirre seuraavaksi solmuksi, ja m pidetään vakiona joka oksalla.

3.3 Satunnainen metsä

Satunnainen metsä on Leo Breimanin [3] kehittämä nopea ja tarkka luokittelualgoritmi, joka koostuu satunnaisten päätöspuiden kokoelmasta. Se pystyy käsittelemään hyvinkin suuria muuttujajoukkoja. Satunnainen metsä pyrkii parantamaan päätöspuun robustisuutta rakentamalla puut alkuperäisestä aineistosta poimittujen bootstrap-otosten pohjalta. Satunnaisen päätöspuun käyttäminen siten parantaa menetelmän tarkkuutta.

Yksittäisen päätöspuun muodostaminen alkaa bootstrap-otoksesta. Jos havaintojen lukumäärä alkuperäisessä aineistossa on N, siitä poimitaan N alkion otos palauttaen. Poimitaan siis otokseen datamatriisin rivejä. Tämä otos toimii opetusaineistona puun rakentamisessa. Satunnainen päätöspuu kasvatetaan bootstrap-otosta käyttäen siten, että joka solmussa m piirrettä arvotaan alkuperäisten M piirteen joukosta, $m \leq M$, ja näistä valitaan paras jako. Puu kasvatetaan mahdollisimman suureksi, jolloin jokaiselle oksalle jää vain samaan luokkaan kuuluvia havaintoja. Näin muodostetaan S puuta, kun m pysyy vakiona.

Kun päätöspuu rakennetaan bootstrap-otosta käyttäen, osa havainnoista jää otoksen ulkopuolelle. Koska otanta tehdään palauttaen, jokaisella havainnolla on joka kerta yhtä suuri todennäköisyys tulla poimituksi otokseen, $P(\mathbf{x} \text{ valitaan}) = 1/N$. Kun poimitaan N alkion otos palauttaen, todennäköisyys, ettei havainto $\mathbf{x}$ tule kertaakaan otokseen on $(1 - 1/N)^N \approx e^{-1} = 0.368$. Keskimäärin 37 % havainnoista jää siis opetusaineiston ulkopuolelle. Nämä ulkopuoliset havainnot muodostavat ns. oob-aineiston (out-of-bag), jota voidaan hyödyntää luokitteluvirheen arvioinnissa. Luokitellaan $\mathbf{x}$ kaikilla niillä puilla, joissa se on ollut oob-havainto. Havainnon $\mathbf{x}$ luokka äänestetään siis niiden puiden kesken, joiden muodostamisessa sitä ei ole käytetty. Tehdään sama kaikille havainnoille ja lasketaan väärinluokiteltujen osuus. Kun puiden määrä satunnaisessa metsässä kasvaa, oob-luokitteluvirhe e_{oob} konvergoi oikeaan luokitteluvirheeseen ([3]). Näin saadaan asymptoottisesti harhaton estimaattori luokitteluvirheelle, joten erillistä testiaineistoa tai ristiinvalidointia ei tarvita. Tämä on siis arvio uuden, metsästä riippumattoman aineiston luokitteluvirheelle, kun kaikki uuden aineiston havainnot käydään kaikkien metsän puiden läpi ja äänestetään niille luokat. Paras tulos saavutetaan, kun valitaan m siten, että e_{oob} minimoituu. Usein valinnalla $m = \sqrt{M}$ saavutetaan lähes optimaaliset tulokset

3.4 Satunnainen Bayes-metsä

Satunnaisen metsän menetelmää voidaan käyttää parantamaan myös muiden luokittelijoiden robustisuutta ja tarkkuutta ([13]). Muodostetaan satunnaisten puiden tapaan bootstrap-otosten avulla useita satunnaisia Bayes-luokittelijoita ja luokitellaan havainnot äänestämällä kaikkien luokittelijoiden kesken. Kutsutaan tätä kokonaisuutta satunnaiseksi Bayes-metsäksi.

Yksittäisen Bayes-luokittelijan muodostaminen aloitetaan poimimalla alkuperäisten N havainnon joukosta palauttaen N alkion otos. Tämän jälkeen arvotaan m piirrettä alkuperäisten M piirteen joukosta, $m \leq M$, jolloin havaintovektori saa muodon $\mathbf{x} = (\tilde{x}_1, \ldots, \tilde{x}_m)'$, missä $\tilde{x}_j \in \{x_1, \ldots, x_M\}$. Luokittelijalla lasketaan luokkakohtaiset posterioritodennäköisyydet

$$P(\omega_i|\mathbf{x}) = \frac{\pi_i f_i(\mathbf{x})}{\sum_{l=1}^{k} \pi_l f_l(\mathbf{x})}$$

kaikilla $i = 1, \ldots, k$ jokaiselle havainnolle. Muodostetaan näin S satunnaista Bayes-luokittelijaa, kun m pysyy vakiona. Jokainen luokittelija äänestää havainnolle $\mathbf{x}$ todennäköisintä luokkaa ja havainto määrätään eniten ääniä saaneeseen luokkaan. Koska Bayes-luokittelija tuottaa luokkakohtaiset posterioritodennäköisyydet, äänestyksen sijaan voitaisiin laskea keskiarvot näistä todennäköisyyksistä ja suorittaa luokittelu niiden mukaan.

Kuten perinteisessä Bayes-luokittelijassa, jos $\mathbf{X}|\omega_i \sim N_m(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ luokassa ω_i ja $\boldsymbol{\Sigma}_i = \sigma^2\mathbf{I}$ kaikille i tai $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}$ kaikille i, saadaan lineaariset luokittelufunktiot

$$g_i(\mathbf{x}) = \mathbf{w}_i'\mathbf{x} + w_{i0}.$$

Muissa tapauksissa, eli kun $\boldsymbol{\Sigma}_i$ on mielivaltainen, saadaan kvadraattiset luokittelufunktiot

$$g_i(\mathbf{x}) = \mathbf{x}'\mathbf{W}_i\mathbf{x} + \mathbf{w}_i'\mathbf{x} + w_{i0}.$$

Käytännössä luokittelusääntö muodostetaan bootstrap-otoksesta laskettujen estimaattien $\{\hat{\boldsymbol{\mu}}_i\}$ ja $\{\hat{\boldsymbol{\Sigma}}_i\}$ pohjalta.

Kun Bayes-luokittelijat muodostetaan bootstrap-otoksia käyttäen, jokainen havainto jää ulkopuolelle noin 1/3:ssa luokittelijoista. Äänestetään havainnolle luokkaa kaikilla niillä luokittelijoilla, joiden muodostamisessa se ei ole ollut mukana. Määrätään havainto siihen luokkaan, joka saa eniten ääniä ja toistetaan tämä kaikille havainnoille. Väärin luokiteltujen osuudesta e_{oob} saadaan jälleen estimaatti luokitteluvirheelle.

3.5 Luokittelumenetelmien vertailu

Yleisesti tilastollisia malleja voidaan vertailla keskenään ennustevirheen avulla. Olkoon vastemuuttuja Y, selittävät muuttujat $\mathbf{X}$ ja $\hat{f}(\mathbf{X})$ opetusaineistoon τ sovitettu

malli. Tappiofunktio, jolla mitataan virhettä Y:n ja $\hat{f}(\mathbf{X})$:n välillä on $L(Y, \hat{f}(\mathbf{X}))$. Luokittelumenetelmien tapauksessa tappiofunktioksi asetetaan

$$L(Y, \hat{f}(\mathbf{X})) = \begin{cases} 0, & \text{jos } \hat{f}(\mathbf{X}) = Y \\ 1, & \text{jos } \hat{f}(\mathbf{X}) \neq Y \end{cases} \tag{3.5.1}$$

Edellä oleva tappiofunktio $L(Y, \hat{f}(\mathbf{X}))$ vastaa luvussa 3.1.1 päätössäännön muodostamisen yhteydessä esitettyä tappiofunktiota (3.1.3). Seuraava teoriaosuus on kirjoitettu seuraten kirjan *The Elements of Statistical Learning* [6] esitystä.

Aineistoa, jonka avulla luokittelusääntö rakennetaan, kutsutaan opetusaineistoksi. Luokittelusäännön toimivuutta voidaan testata testiaineiston avulla. Testivirhe on ennustevirhe mallista riippumattomalle testiaineistolle:

$$\mathrm{Err}_\tau = \mathbb{E}[L(Y, \hat{f}(\mathbf{X}))|\tau]. \tag{3.5.2}$$

Se on tappiofunktion (3.5.1) tapauksessa testiaineiston väärinluokiteltujen havaintojen osuus. Vastaavasti odotettu testivirhe on

$$\begin{aligned} \mathrm{Err} &= \mathbb{E}[L(Y, \hat{f}(\mathbf{X}))] \\ &= \mathbb{E}[\mathrm{Err}_\tau]. \end{aligned} \tag{3.5.3}$$

Opetusvirhe taas on keskimääräinen tappio opetusaineistolle:

$$\overline{\mathrm{err}} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(\mathbf{x}_i)).$$

Se on opetusaineiston väärinluokiteltujen havaintojen osuus.

Tavoitteena on laskea ennustevirhe mallille $\hat{f}$. Opetusvirhe ei ole hyvä estimaatti tälle, sillä se käyttää ennustevirheen arviointiin samoja havaintoja, joilla malli rakennettiin. Lisäämällä mallin kompleksisuutta voidaan saada $\overline{\mathrm{err}} = 0$, jolloin malli ylisovittuu opetusaineistoon, eikä ole hyvin yleistettävissä.

Yleinen tapa on jakaa havainnot opetus- ja testiaineistoon. Kun osa havainnoista otetaan erilleen opetusaineistosta ennen mallin sovittamista, saadaan riippumaton testiaineisto. Testiaineisto voidaan poimia satunnaisotannalla koko aineistosta tai luokittain. Jos testiaineiston havainnot arvotaan luokkien sisällä, varmistetaan kaikkien luokkien edustus. Testivirhettä ei pitäisi käyttää mallinvalinnassa, koska tällöin se voi aliestimoida lopullisen mallin ennustevirhettä. Parhaassa tapauksessa aineistoa on käytettävissä niin paljon, että se voidaan jakaa kolmeen osaan: opetus-, validointi- ja testiaineistoon. Tällöin opetusaineiston avulla muodostetaan luokittelusääntö, validointiaineistolla estimoidaan ennustevirhettä luokittelumenetelmän valintaa varten ja testiaineistolla arvioidaan valitun luokittelijan ennustevirhe. Käytännössä tämä ei kuitenkaan aina ole mahdollista.

Yksinkertaisin ja yleisimmin käytetty tapa ennustevirheen arviointiin on ristiinvalidointi (cross validation). Ristiinvalidoinnissa aineisto jaetaan K:hon osaan, $N_1, \ldots, N_K$, joista jokaisessa on n havaintoa. Tätä kutsutaan K-kertaiseksi ristiinvalidoinniksi. K:nnelle osalle luokittelusääntö muodostetaan muilla $K-1$ osalla ja luokitteluvirhe lasketaan pois jätetyllä osalla. Tämä toistetaan K kertaa, kunnes jokaisella osalla on arvioitu luokitteluvirhettä. Ennustevirhe saadaan laskemalla näistä luokitteluvirheistä keskiarvo. Jos $K = N$, kyseessä on leave-one-out -ristiinvalidointi, jossa testiaineistona toimii aina yksi havainto kerrallaan. Olkoon $\kappa : \{1, \ldots, N\} \mapsto \{1, \ldots, K\}$ indeksifunktio, joka kertoo, mihin jakoon havainto i arvotaan. Merkitään $\hat{f}^{-k}(\mathbf{x})$:lla mallia, joka on sovitettu, kun k:s osa on jätetty sivuun. Näin saadaan ristiinvalidoinnilla ennustevirheelle estimaatti

$$CV = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(\mathbf{x}_i)).$$

Ennustevirhettä voidaan arvioida myös bootstrap-otosten avulla. Olkoon $\mathbf{Z} = (z_1, \ldots, z_N)$ opetusaineisto, missä $z_i = (\mathbf{x}_i, y_i)$. Poimitaan siitä palauttaen N alkion otos B kertaa, jolloin saadaan B bootstrap-aineistoa, $\mathbf{Z}^{*1}, \ldots, \mathbf{Z}^{*B}$. Yksi tapa tutkia ennustevirhettä bootstrapin avulla on rakentaa luokittelusääntö kaikilla otoksilla $\mathbf{Z}^{*i}$, $i = 1, \ldots, B$, ja laskea näille luokitteluvirhe alkuperäisellä aineistolla $\mathbf{Z}$:

$$\widehat{\mathrm{Err}}_{boot} = \frac{1}{B} \times \frac{1}{N} \sum_{b=1}^{B} \sum_{i=1}^{N} L(y_i, \hat{f}^{*b}(\mathbf{x}_i)),$$

missä $\hat{f}^{*b}$ on b:nnen bootstrap-otoksen avulla muodostetun luokittelusäännön piirrevektorille $\mathbf{x}_i$ ennustama luokka. Tämä ei ole hyvä estimaatti ennustevirheelle, koska opetus- ja testiaineistoissa on samoja havaintoja.

Kun otanta tehdään palauttaen, hieman yli puolet havainnoista päätyy otokseen:

$$P(\text{havainto } i \in \mathbf{Z}^{*b}) = 1 - (1 - \frac{1}{N})^N \approx 1 - e^{-1} = 0.632.$$

Otosten ulkopuolelle jääviä havaintoja voidaan hyödyntää nyt testiaineistona. Leave-one-out -bootstrap-estimaatti ennustevirheelle on

$$\widehat{\mathrm{Err}}^{(1)} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} L(y_i, \hat{f}^{*b}(\mathbf{x}_i)),$$

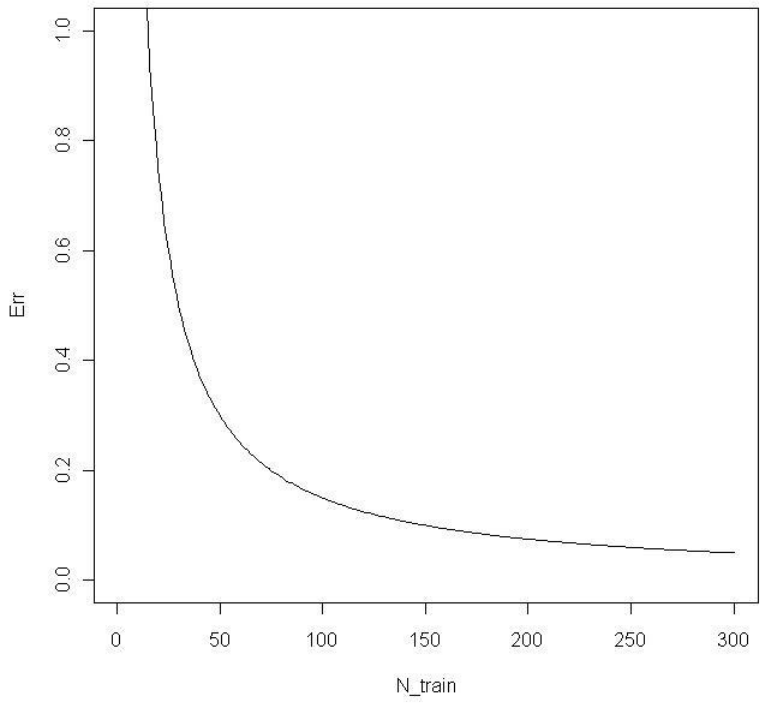
missä C^{-i} on kaikkien niiden bootstrap-otosten b joukko, jotka eivät sisällä havaintoa i, ja $|C^{-i}|$ näiden otosten lukumäärä. Satunnaisen metsän ja satunnaisen Bayesmetsän menetelmissä käytetään juuri tätä estimaattia.

Mitä suurempi opetusaineisto on, sitä parempi luokittelija saadaan. Toisaalta, mitä suurempi testiaineisto on, sitä tarkempi on ennustevirheen arvio. Siksi monesti

testivirheen laskemisen jälkeen testiaineisto yhdistetään opetusaineistoon ja luokittelusääntö rakennetaan uudestaan. Näin saadaan mahdollisimman paljon tietoa lopullisen luokittelijan käytettäväksi. Opetusaineiston koko voi myös aiheuttaa harhaa ennustevirheen estimaattiin, jos mallin toimivuus riippuu siitä liikaa. K-kertaisessa ristiinvalidoinnissa tämä vaikuttaa K:n valintaan. Bootstrap-otokseen päätyy aina hieman yli puolet havainnoista, joten $\widehat{\text{Err}}^{(1)}$:n sisältämä harha käyttäytyy samoin kuin 2-kertaisessa ristiinvalidoinnissa. Kuvasta 3.5.1 nähdään esimerkki harhasta, joka aiheutuu, jos aineisto koostuisi sadasta havainnosta ja käytettäisiin 2-kertaista ristiinvalidointia. Testivirhe, joka saadaan tässä tapauksessa 50 havainnon opetusaineistolla, on huomattavasti suurempi kuin, jos käytettäisiin suurinta osaa havainnoista opetusaineistona. Toisaalta, jos aineisto koostuisi 300 havainnosta, harha olisi paljon pienempi. Opetusaineiston koosta aiheutuvaa harhaa voidaan korjata $\widehat{\text{Err}}^{(.632)}$-estimaattorilla:

$$\widehat{\text{Err}}^{(.632)} = 0.368 \times \overline{\text{err}} + 0.632 \times \widehat{\text{Err}}^{(1)}.$$

Näin ylöspäin harhaisen leave-one-out -bootstrap-estimaatin arvo pienenee kohti opetusvirhettä.



Kuva 3.5.1: Opetusaineiston koon vaikutus. Kuvassa on testivirhe piirrettynä opetusaineiston kokoa vasten.

Luku 4

Luokitteluvirheen vaikutus

Koneellisen tunnistamisen tarkoituksena on mahdollistaa entistä tarkempi pohjaeläinlajien osuuksien arviointi näytteen perusteella. Tässä luvussa pohditaan, miten luokittelussa tapahtuva virhe vaikuttaa lajisuhteiden arviointiin. On tärkeää huomata, että luokitteluvirheet vaihtelevat lajeittain. Lajiosuuksille muodostetaan ensin estimaattorit, joissa huomioidaan luokittelussa tapahtuva virhe. Tämän jälkeen pohditaan, miten voidaan tutkia näiden estimaattoreiden hyvyyttä.

4.1 Osuuksien arviointi

Yleensä luokittelussa pyritään luokitteluvirheen minimointiin. Joissakin tapauksissa, kuten pohjaeläimiä luokitellessa tai metsän ja maanviljelyn peittoalaa arvioitaessa, ollaan kuitenkin kiinnostuneempia luokkien suhteellisista osuuksista. Tällöin pyritään mahdollisimman tarkkoihin osuuksien estimaatteihin, ei tarkkaan yksittäisen havainnon luokitteluun.

Oletetaan aluksi, että populaatiossa esiintyy k luokkaa. Tästä populaatiosta otetaan näyte, josta havainnot tunnistetaan koneellisesti. Otetaan käyttöön seuraavat merkinnät:

- $\mathbf{p} = [p_1 \cdots p_k]'$ luokkien oikeat osuudet näytteessä
- $\hat{\mathbf{p}} = [\hat{p}_1 \cdots \hat{p}_k]'$ luokittelun tuloksena saadut luokkien osuudet.

Oletetaan lisäksi, että käytettävissämme on luokittelija ja testiaineisto. Tarkastellaan testiaineiston osalta tapahtumia

- $B_i =$ yksilö luokitellaan luokkaan i, $i = 1, \ldots, k$
- $C_j =$ yksilö on peräisin luokasta j, $j = 1, \ldots, k$

ja seuraavia todennäköisyyksiä:

- $P(B_i) =$ todennäköisyys tulla luokitelluksi luokkaan i, $i = 1, \ldots, k$

- $P(C_j)$ = todennäköisyys olla peräisin luokasta j, $j = 1, \ldots, k$
- $P(B_i|C_j)$ = todennäköisyys tulla luokitelluksi luokkaan i, kun on peräisin luokasta j, $i = 1, \ldots, k$, $j = 1, \ldots, k$.

Tapahtuma B_i koostuu siis k:sta vaihtoehdosta. Joko yksilö on peräisin luokasta i ja tulee myös luokitelluksi siihen tai yksilö on peräisin luokasta j, $j \neq i$, ja tulee virheellisesti luokitelluksi luokkaan i. Kun testiaineisto luokitellaan, saadaan luokittelun tuloksena sekaannusmatriisi

	C_1	C_2	$\cdots$	C_k	
B_1	n_{11}	n_{12}	$\cdots$	n_{1k}	n_1
B_2	n_{21}	n_{22}	$\cdots$	n_{2k}	n_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
B_k	n_{k1}	n_{k2}	$\cdots$	n_{kk}	n_k
					n

,

josta todennäköisyydet $P(B_i|C_j)$ voidaan laskea. Merkitään näitä todennäköisyyksiä matriisilla $\mathbf{A} = (a_{ij})$, $a_{ij} = P(B_i|C_j)$. Matriisissa $\mathbf{A}$ diagonaalilla ovat nyt todennäköisyydet luokitella havainto oikein ja muualla todennäköisyydet luokitella se väärin. Täydelliselle luokittelijalle $\mathbf{A} = \mathbf{I}$.

Nyt ollaan kiinnostuneita luokkien osuuksista näytteessä, ei luokkien prioritodennäköisyyksistä. Näytteen luokittelun tuloksena saadut osuudet $\hat{\mathbf{p}}$ eivät ole hyvä estimaattori todellisille osuuksille $\mathbf{p}$, sillä

$$\mathbb{E}[\hat{\mathbf{p}}|\mathbf{p}] = \mathbf{A}\mathbf{p}.$$

Nämä sisältävät luokkakohtaisista luokitteluvirheistä aiheutuvaa harhaa ([7], [5]). Sen sijaan $\mathbf{p}$:n voi estimoida pns-menetelmällä mallista

$$\hat{\mathbf{p}} = \mathbf{A}\mathbf{p} + \boldsymbol{\varepsilon},$$

missä $\boldsymbol{\varepsilon}$ on satunnaisvektori, jolle pätee $\sum_i \boldsymbol{\varepsilon}_i = 0$ ja $\mathbb{E}[\boldsymbol{\varepsilon}] = 0$. Matriisi $\mathbf{A}$ on yleensä kääntyvä, jolloin

$$\hat{\mathbf{p}}^* = \mathbf{A}^{-1}\hat{\mathbf{p}} \tag{4.1.1}$$

on ainoa harhaton lineaarinen estimaattori $\mathbf{p}$:lle ([1]). Tämä tunnetaan nimellä sekaannusmatriisikorjaus.

4.2 Estimaattoreiden arviointi

Jotta voidaan arvioida luokitteluvirheestä korjattujen estimaattoreiden (4.1.1) hyvyyttä, aineisto on jaettava kolmeen osaan: opetus- ja testiaineistoon sekä näytteeseen. Opetusaineiston avulla muodostetaan luokittelija. Testiaineiston luokittelun tuloksena saadaan sekaannusmatriisi, josta todennäköisyydet a_{ij} voidaan laskea. Tämän

jälkeen näyte luokitellaan, jolloin saadaan luokittelun tuloksena luokkien osuuksien estimaatit $\hat{\mathbf{p}}$. Näiden ja matriisin $\mathbf{A}$ avulla voidaan laskea korjatut estimaatit $\hat{\mathbf{p}}^*$.

Estimaattien keskinäiseen vertailuun vaaditaan kuitenkin useita simulointeja ja tässä voidaan hyödyntää bootstrap-otoksia. Jos testiaineiston koko on N_{test}, poimitaan siitä B kertaa N_{test} alkion otos. Luokitellaan otokset N^b_{test}, jolloin niiden sekaannusmatriiseista saadaan A_b, $b = 1, \ldots, B$. Jos näytteen koko on N_{sample}, poimitaan siitä B kertaa N_{sample} alkion otos. Lasketaan kullekin N^b_{sample} luokkien oikeat osuudet $\mathbf{p}_b$, luokittelun tuloksena saadut osuuksien estimaatit $\hat{\mathbf{p}}_b$ sekä niiden luokitteluvirheen vaikutuksesta korjatut estimaatit $\hat{\mathbf{p}}^*_b = \mathbf{A}_b^{-1}\hat{\mathbf{p}}_b$, $b = 1, \ldots, B$. Tällä tavoin saadaan jakaumat osuuksille sekä molemmille estimaateille ja päästään vertaamaan, kumman estimaatin jakauma on lähempänä todellisten osuuksien jakaumaa.

Jakaumien vertailuun voidaan käyttää X^2-etäisyysmittaa ([11])

$$X^2 = \sum_{i=1}^{k} \frac{(h_i - p_i)^2}{p_i},$$

missä h_i on luokittelun tuloksena saatu osuuden estimaatti tai korjattu estimaatti. Lasketaan kullekin bootstrap-otokselle summa jokaisen estimaatin ja oikean osuuden suhteutetusta etäisyydestä. Näin saadaan X^2-etäisyysmittojen empiiriset bootstrap-jakaumat, $X^2_1, \ldots, X^2_B$ ja $X^{2*}_1, \ldots, X^{2*}_B$. Näitä voidaan vertailla keskiarvon ja vaihtelun perusteella. Mitä enemmän pieniä etäisyyksiä oikeisiin osuuksiin saadaan, sitä parempi estimaatti on kyseessä.

Luku 5

Pohjaeläinten luokittelu

Tässä luvussa pohjaeläinaineistoa pyritään luokittelemaan Bayes-luokittelijalla, C4.5-päätöspuulla, satunnaisella metsällä ja satunnaisella Bayes-metsällä. Kaikille menetelmille lasketaan myös luokitteluvirheen vaikutuksesta korjatut lajiosuuksien estimaatit ja niitä verrataan luokittelun tuloksina osuuksien estimaatteihin. Luokittelumenetelmiä vertaillaan luokitteluvirheen ja sen perusteella, millä menetelmällä saadaan lähinnä osuuksien oikeaa jakaumaa oleva estimaattien jakauma. Tulokset on saatu käyttäen R-ohjelmaa, [15].

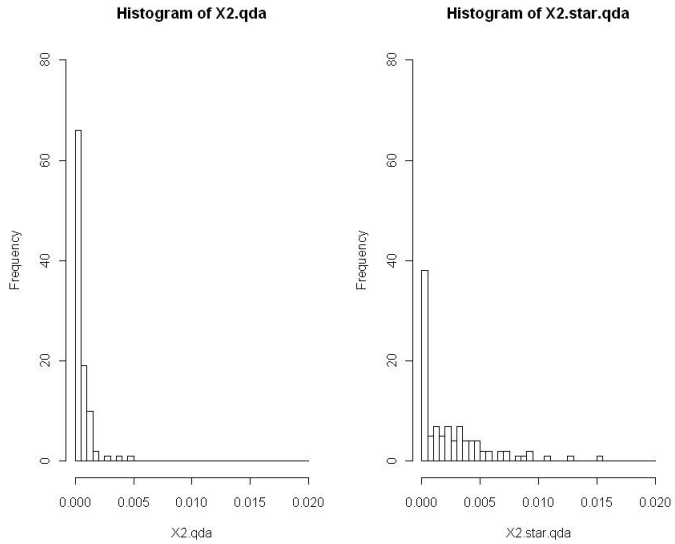
5.1 Bayes-luokittelija

Bayes-luokittelualgoritmissa käytetään kvadraattista luokittelufunktiota (3.1.6). Luokittelija muodostetaan käyttäen opetusaineistona 650 satunnaisesti arvottua havaintoa ja kaikkia käytössä olevaa 15 piirrettä. Prioritodennäköisyydet asetetaan yhtäsuuriksi kaikille luokille. Testiaineistona käytetään jäljelle jääviä 700 havaintoa. Opetus- ja testiaineistoon arvotaan sata jakoa, jolloin testivirheistä (3.5.2) saadaan sata arviota luokitteluvirheelle. Näiden keskiarvo on $\overline{\mathrm{Err}}_\tau = 0.0736$, joka on arvio odotetulle testivirheelle Err (3.5.3). Jos simuloinnit tehdään siten, että jokaisesta pohjaeläinlajista arvotaan puolet havainnoista opetusaineistoon ja loput muodostavat testiaineiston, saadaan vielä hieman parempi luokitteluvirheen estimaatti, joka on $\overline{\mathrm{Err}}_\tau = 0.0706$.

Luokitteluvirheestä korjattuja lajiosuuksien estimaatteja varten aineisto jaetaan kolmeen osaan siten, että jokaiseen osaan arvotaan kolmasosa jokaisen lajin havainnoista. Opetusaineiston avulla muodostetaan luokittelija, joka pysyy kiinteänä. Testiaineistosta poimitaan 100 bootstrap-otosta, joiden luokittelun tuloksena saadaan sekaannusmatriisit, joista luokkakohtaiset luokitteluvirheiden todennäköisyydet voidaan laskea. Kolmannesta aineistosta, näytteestä, poimitaan samoin 100 bootstrap-otosta, jotka luokitellaan ja luokittelun tuloksista lasketaan lajiosuuksien estimaatit $\hat{\mathbf{p}}_i$. Näytteestä lasketaan lisäksi testiaineistojen sekaannusmatriisien avulla luokitteluvirheistä korjatut osuuksien estimaatit $\hat{\mathbf{p}}_i^*$.

Estimaattien vertaamiseksi, jokaisesta otoksesta lasketaan X^2-etäisyysmitta molemmille estimaateille. Näin saadaan estimaattien etäisyydelle todellisista lajiosuuksista empiiriset bootstrap-jakaumat. Ne on esitetty kuvassa 5.1.1. Kuvasta nähdään,

että Bayes-luokittelijan tapauksessa korjaus ei paranna lajiosuuksien estimaatteja. Korjattujen estimaattien etäisyyksissä todellisista lajiosuuksista on huomattavasti enemmän vaihtelua ja vähemmän pieniä arvoja. Samaa päätelmää tukevat myös keskiarvot $\overline{X^2} = 0.00049$, $\overline{X^{2*}} = 0.00252$.



Kuva 5.1.1: Empiiriset bootstrap-jakaumat X^2-etäisyysmitalle Bayes-luokittelijalla sadalla simulaatiolla. Vasemmassa kuvassa simulaatioista lasketut X^2-etäisyydet näytteen luokittelun tuloksena saaduista lajiosuuksien estimaateista ja havaituista lajiosuuksista. Oikeassa kuvassa vastaavasti X^2-etäisyydet korjatuille lajiosuuksien estimaateille.

5.2 C4.5-päätöspuu

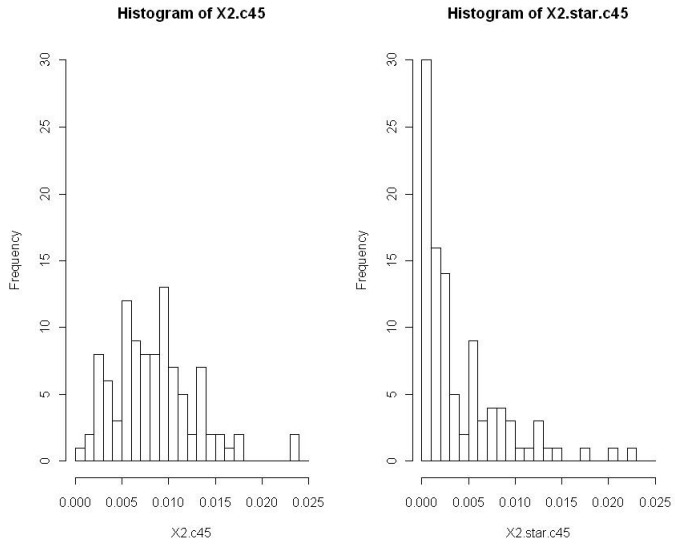
C4.5-päätöspuu muodostetaan siten, että muuttujanvalinnassa kriteerinä käytetään suhteutettua informaatiohyötyä. Puu rakennetaan opetusaineiston avulla ja testiaineiston väärinluokiteltujen havaintojen osuudesta saadaan estimaatti luokitteluvirheelle.

Luokittelu suoritetaan siten, että opetusaineistoon arvotaan satunnaiset 650 havaintoa ja testiaineistona käytetään loppuja 700 havaintoa. Tämä toistetaan 100 kertaa, jolloin saadaan $\overline{\mathrm{Err}_\tau} = 0.1731$. Kun jako opetus- ja testiaineistoon tehdään siten, että jokaisesta lajista puolet havainnoista arvotaan opetusaineistoon ja loppuja käytetään testiaineistona, saadaan aavistuksen parempi luokitteluvirheen estimaatti, $\overline{\mathrm{Err}_\tau} = 0.1708$.

Lajiosuuksien estimaatteja varten aineisto jaetaan jälleen kolmeen osaan ja testiaineistosta sekä näytteestä poimitaan 100 bootstrap-otosta. Näiden avulla saadaan lajiosuuksien estimaatit $\hat{\mathbf{p}}_i$ ja korjatut estimaatit $\hat{\mathbf{p}}_i^*$, $i = 1, \ldots, 100$. Estimaateista ja

korjatuista estimaateista lasketaan X^2-etäisyysmitat, joiden empiiristen bootstrap-jakaumien avulla voidaan vertailla, kumpien estimaattien jakauma on lähempänä todellisten lajiosuuksien jakaumaa. Estimaattien X^2-etäisyysmittojen jakaumat on esitetty kuvassa 5.2.1.

Kuvasta nähdään, että korjatut estimaatit arvioivat oikeita osuuksia paremmin kuin luokittelun tuloksena saadut estimaatit. Niiden etäisyyksissä oikeista lajiosuuksista on huomattavasti enemmän pieniä arvoja. Sama voidaan todeta etäisyyksien keskiarvoista $\overline{X^2} = 0.00843$, $\overline{X^{2*}} = 0.00400$.



Kuva 5.2.1: Empiiriset bootstrap-jakaumat X^2-etäisyysmitalle C4.5-päätöspuulla sadalla simulaatiolla. Vasemmassa kuvassa simulaatioista lasketut X^2-etäisyydet näytteen luokittelun tuloksena saaduista lajiosuuksien estimaateista ja havaituista lajiosuuksista. Oikeassa kuvassa vastaavasti X^2-etäisyydet korjatuille lajiosuuksien estimaateille.

5.3 Satunnainen metsä

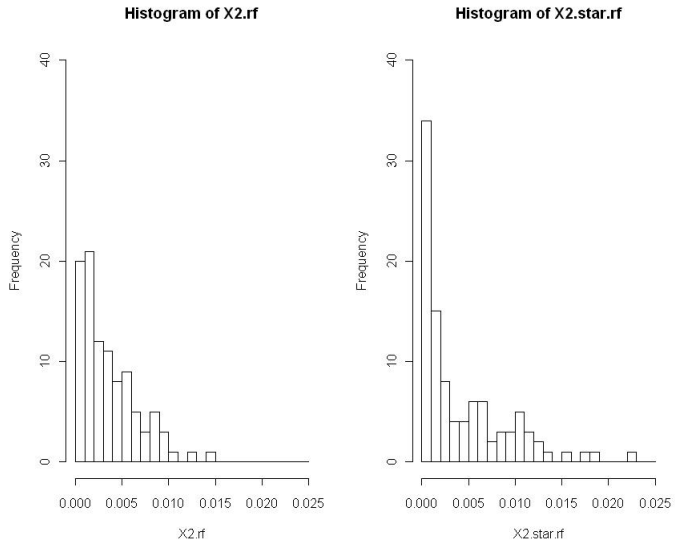
Satunnainen metsä muodostetaan rakentamalla satunnaisia päätöspuita. Menetelmä ei vaadi erillistä testiaineistoa, joten kaikkia havaintoja voidaan käyttää luokittelijan muodostamiseen. Simuloidaan 100 metsää, jotka kaikki koostuvat 70 puusta. Jokainen puu on muodostettu arpomalla kaikissa solmuissa 5 muuttujaa alkuperäisten 15 muuttujan joukosta. Luokitteluvirheelle saadaan estimaatti out-of-bag -virheiden keskiarvona, $\overline{e_{oob}} = 0.0762$. Out-of-bag -havainnoista laskettu e_{oob} vastaa opetusaineiston kanssa samankokoisen, riippumattoman testiaineiston testivirhettä.

Satunnaisen metsän tapauksessa myös estimaatit ja korjatut estimaatit lajiosuuksille saadaan ilman erillistä testiaineistoa. Poimitaan kaikista lajeista kolmannes havainnoista näytteeksi ja käytetään loput opetusaineistona. Metsä rakennetaan ope-

tusaineiston pohjalta, jolloin jokaisen puun kohdalla osa havainnoista jää käyttämättä. Lopuksi jokainen havainto luokitellaan kaikilla niillä puilla, joiden rakentamisessa niitä ei ole käytetty. Tällä tavoin out-of-bag -havainnoista saadaan estimaatti testivirheelle, e_{oob}. Samoin saadaan myös sekaannusmatriisi, jota voidaan käyttää korjattujen estimaattien laskemiseen.

Muodostetaan siis 100 satunnaista metsää. Nyt voidaan koko ajan käyttää samaa opetusaineistoa, sillä luokittelumenetelmä itse poimii aineistosta bootstrap-otoksia ja arpoo puissa käytettävät muuttujat, joten jokainen puu ja metsä on erilainen. Tämän seurauksena saadaan 100 erilaista out-of-bag -sekaannusmatriisia. Poimitaan näytteestä 100 bootstrap-otosta ja luokitellaan jokainen otos yhdellä satunnaisella metsällä. Näin saadaan luokittelun tuloksena estimaatit todellisille lajiosuuksille. Out-of-bag -sekaannusmatriisien avulla saadaan laskettua korjatut estimaatit.

Kuvassa 5.3.1 on esitetty kummastakin estimaatista ja todellisista osuuksista laskettujen X^2-etäisyysmittojen empiiriset bootstrap-jakaumat. Kuvasta katsoen korjatut estimaatit vaikuttavat hieman paremmilta, sillä niiden ja todellisten lajiosuuksien etäisyyksissä on enemmän pieniä arvoja. Keskiarvoja perusteella taas luokittelun tuloksena saadut estimaatit ovat parempia arvioita lajiosuuksille, $\overline{X^2} = 0.00353$, $\overline{X^{2*}} = 0.00429$. Ero keskiarvoissa on kuitenkin pieni.



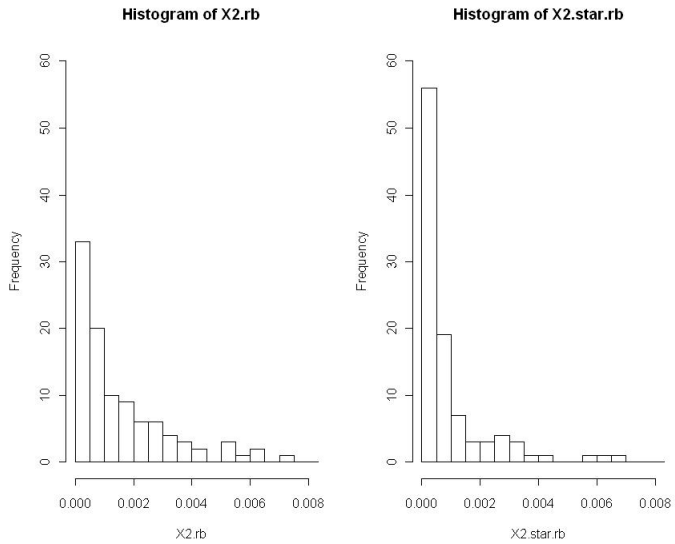
Kuva 5.3.1: Empiiriset bootstrap-jakaumat X^2-etäisyysmitalle satunnaisella metsällä sadalla simulaatiolla. Vasemmassa kuvassa simulaatioista lasketut X^2-etäisyydet näytteen luokittelun tuloksena saaduista lajiosuuksien estimaateista ja havaituista lajiosuuksista. Oikeassa kuvassa vastaavasti X^2-etäisyydet korjatuille lajiosuuksien estimaateille.

5.4 Satunnainen Bayes-metsä

Satunnainen Bayes-metsä muodostuu useista satunnaisista Bayes-luokittelijoista. Satunnaisen metsän tapaan sekään ei vaadi erillistä testiaineistoa. Yksittäinen satunnainen Bayes-luokittelija muodostetaan bootstrap-otoksen pohjalta ja siihen arvotaan 8 piirrettä alkuperäisten 15 piirteen joukosta. Simuloidaan 100 satunnaista Bayes-metsää, jotka kaikki koostuvat 90 satunnaisesta Bayes-luokittelijasta. Näin saadaan luokitteluvirheelle estimaatti out-of-bag -virheiden keskiarvona, $\overline{e_{oob}} = 0.0769$.

Estimaatit ja korjatut estimaatit lajiosuuksille saadaan samoin kuin satunnaisen metsän tapauksessa. Kolmannes kaikkien lajien havainnoista arvotaan näytteeseen ja loppujen havaintojen avulla rakennetaan 100 satunnaista Bayes-metsää. Kun jokainen havainto luokitellaan kaikilla niillä satunnaisilla Bayes-luokittelijoilla, joiden muodostamiseen sitä ei ole käytetty, saadaan kaikista Bayes-metsistä out-of-bag -sekaannusmatriisi. Niitä käytetään korjattujen estimaattien laskemiseen. Näytteestä poimitaan 100 bootstrap-otosta, joista jokainen luokitellaan yhden satunnaisen Bayes-metsän avulla. Näin saadaan estimaatit ja korjatut estimaatit lajiosuuksille.

Molemmista estimaateista lasketaan jälleen X^2-etäisyydet oikeisiin lajiosuuksiin. Kuvassa 5.4.1 on esitetty X^2-etäisyyksien empiiriset bootstrap-jakaumat. Luokitteluvirheen vaikutuksesta korjatuista estimaateista lasketuissa etäisyyksissä on enemmän pieniä arvoja, joten satunnaisen Bayes-metsän tapauksessa korjaus parantaa estimaatteja. Tämä nähdään myös etäisyyksien keskiarvoista, $\overline{X^2} = 0.00151$, $\overline{X^{2*}} = 0.00089$.

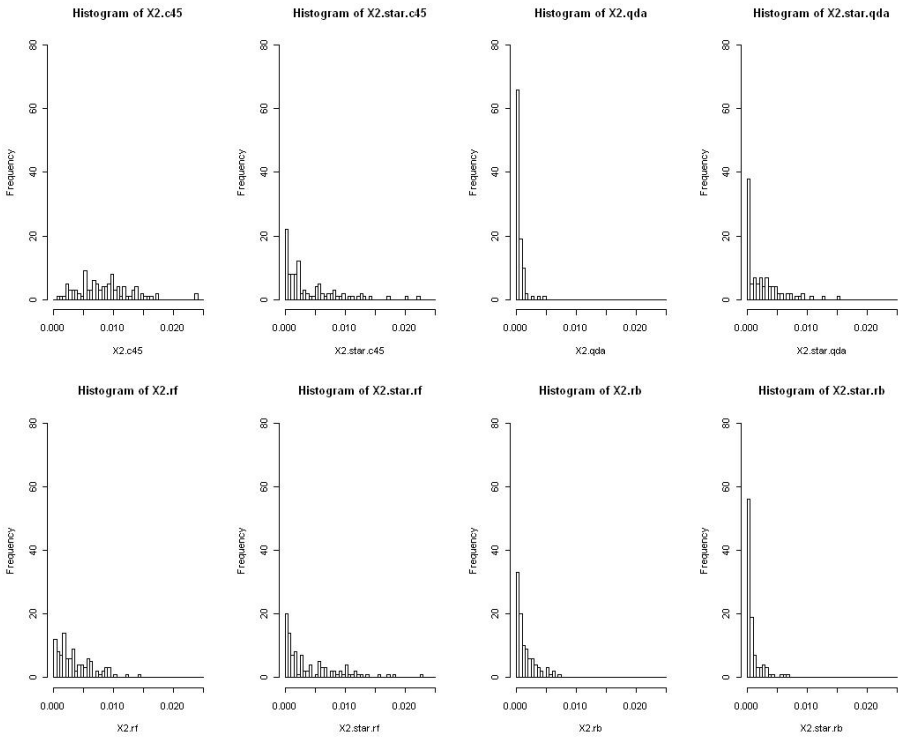


Kuva 5.4.1: Empiiriset bootstrap-jakaumat X^2-etäisyysmitalle satunnaisella Bayes-metsällä sadalla simulaatiolla. Vasemmassa kuvassa simulaatioista lasketut X^2-etäisyydet näytteen luokittelun tuloksena saaduista lajiosuuksien estimaateista ja havaituista lajiosuuksista. Oikeassa kuvassa vastaavasti X^2-etäisyydet korjatuille lajiosuuksien estimaateille.

5.5 Luokittelumenetelmien vertailu

Analyyseissa käytetyllä pohjaeläinaineistolla pienin luokitteluvirhe saadaan Bayes-luokittelijalla. Satunnaisen metsän ja satunnaisen Bayes-metsän luokitteluvirheet ovat kuitenkin lähes yhtä pieniä. Ainoastaan C4.5-päätöspuu toimii selvästi muita luokittelijoita huonommin.

Luokitteluvirheen sijaan menetelmiä voidaan myös vertailla sen mukaan, millä luokittelijalla saadaan lajiosuuksille todellisia osuuksia lähimpänä olevat estimaatit. Tätä varten kuvassa 5.5.1 on piirretty kaikkien käytettyjen luokittelumenetelmien estimaateista ja todellisista osuuksista laskettujen X^2-etäisyysmittojen empiiriset bootstrap-jakaumat samaan mittakaavaan. Kuvasta nähdään, että perinteinen Bayes-luokittelija tuottaa luokittelun tuloksena kaikkein parhaimmat estimaatit todellisille lajiosuuksille. Lähes yhtä hyviä tuloksia saavutetaan kuitenkin satunnaisen Bayes-metsän luokitteluvirheen vaikutuksesta korjatuilla estimaateilla. Heikoimmin toimivat C4.5-päätöspuun luokittelun tuloksena saadut estimaatit.



Kuva 5.5.1: Empiiriset bootstrap-jakaumat X^2-etäisyysmitalle kaikilla luokittelijoilla sadalla simulaatiolla. Ylärivillä simulaatioista lasketut X^2-etäisyydet C4.5-päätöspuulla lajiosuuksien estimaateille ja korjatuille estimaateille sekä vastaavasti Bayes-luokittelijalla. Alarivillä simulaatioista lasketut X^2-etäisyydet satunnaisella metsällä lajiosuuksien estimaateille ja korjatuille estimaateille sekä vastaavasti satunnaisella Bayes-metsällä.

Luku 6

Loppusanat

Tutkielmassa sovellettiin erilaisia luokittelumenetelmiä pohjaeläinaineistoon tarkoituksena löytää menetelmä, jota jatkossa voidaan käyttää pohjaeläinten koneelliseen tunnistamiseen. Luokittelu onnistui hyvin kaikilla menetelmillä, joista parhaaksi osoittautui Bayes-luokittelija. Sillä saavutettuja tuloksia on esitelty myös julkaistavaksi lähetetyssä käsikirjoituksessa [10]. Bayes-luokittelija oli paras menetelmä sekä luokitteluvirheen että lajiosuuksien estimaattien mielessä. Satunnaisen metsän sovellus Bayes-luokittelijaan tuotti lähes yhtä hyviä tuloksia. Sen parametreja säätämällä on mahdollista, että se voisi toimia jopa tavallista Bayes-luokittelijaa paremmin. Satunnainen Bayes-metsä toimii siten, että jokainen satunnainen Bayes-luokittelija äänestää havainnolle todennäköisintä luokkaa ja äänestyksen voittanut luokka asetetaan ennusteeksi. Koska Bayes-luokittelijat tuottavat luokkakohtaiset todennäköisyydet, voitaisiin äänestyksen sijaan kaikkien luokittelijoiden tuottamat todennäköisyydet summata ja valita sitten ennusteeksi kaikkein todennäköisin luokka. Olisikin mielenkiintoista tutkia, saavutettaisiinko tällä tavoin parempia tuloksia kuin perinteisellä äänestyksellä.

Kaikilla menetelmillä saavutettiin niin pieniä luokitteluvirheitä, että tämä herätti epäilyksiä pohjaeläinten skannausvaiheen teknisistä ongelmista. On mahdollista, että eri pohjaeläinlajien harmaasävykuvien taustat ovat tummuusasteeltaan erilaisia. Näin ollen voisi olla, että luokittelija tunnistaa kuvasta taustan eikä itse pohjaeläintä. Tämän varmistamiseksi luotiin uusi aineisto ottamalla jokaisen havainnon kuvasta pala taustaa. Taustalle annettiin luokaksi sama kuin kuvassa olleella pohjaeläimellä ja näin saatu aineisto luokiteltiin. Esimerkiksi satunnaisen metsän menetelmällä taustapalojen luokittelun luokitteluvirheeksi saatiin 0.7215. Tämä puoltaa sitä, etteivät pohjaeläinten luokittelussa saavutetut pienet luokitteluvirheet johtuneet teknisistä ongelmista.

Pohjaeläinten luokittelussa kiinnostuksen kohteena ovat lajien suhteelliset osuudet vesistöistä poimituissa näytteissä. Näytteen luokittelun tuloksena saadaan estimaatit näille lajiosuuksille, mutta ne sisältävät harhaa. Todennäköisyys tulla luokitelluksi tiettyyn luokkaan vaihtelee luokittain ja tämä vaikuttaa osuuksien estimaatteihin. Niinpä tutkielmassa sovellettiin sekaannusmatriisikorjausta lajiosuuksien estimaattoreihin. Bayes-luokittelijan kohdalla korjaus ei parantanut estimaatteja, vaan suoraan luokittelun tuloksena saadut estimaatit olivat lähempänä todellisia lajiosuuksia. Mui-

den luokittelijoiden tapauksessa korjaus kuitenkin paransi estimaatteja. Jatkon kannalta onkin mielenkiintoista, jos satunnaisella Bayes-metsällä voidaan oikeilla parametrien arvoilla saavuttaa tavallista Bayes-luokittelijaa parempia tuloksia, onko näin myös lajiosuuksien estimaattien tapauksessa. Sekaannusmatriisikorjaus ei myöskään ole ainoa osuuksien estimaattoreille kehitetty korjaus ja jatkossa voitaisiin tutkia, saavutetaanko muilla korjausmenetelmillä vielä parempia tuloksia.

Yksi mielenkiintoinen jatkotutkimuskohde on myös ulkopuolisten havaintojen tunnistaminen. Jos koko tunnistusprosessi on automatisoitu, voi osa kuvista sisältää jotain muuta kuin pohjaeläimen. Osa luokittelijoista tuottaa ennustetun luokan lisäksi todennäköisyyden kuulua kuhunkin luokkaan. Voitaisiinkin kehittää luokittelija, joka jättäisi havainnon luokittelematta, jos se ei riittävän suurella todennäköisyydellä kuulu mihinkään luokkaan. Tällaisen rajan etsiminen vaatisi toki laajaa aineistoa, jossa olisi sekä pohjaeläimiä että ulkopuolisia havaintoja.

Tässä tutkielmassa Bayes-luokittelijassa ja satunnaisen Bayes-metsän menetelmässä prioritodennäköisyydet olivat kaikilla luokilla yhtä suuret. Asiantuntijoilla on kuitenkin esittää hyvinkin tarkkoja arvioita vesistöittäin vaihteleville lajikohtaisille prioritodennäköisyyksille. Olisi kiinnostavaa päästä käyttämään tutkielmassa esitettyjä Bayes-luokittelijoita oikeaan vesistöstä poimittuun näytteeseen. Tällöin voitaisiin nähdä, miten paljon prioritodennäköisyydet luokittelun tuloksiin vaikuttavat, ja miten tämä välittyy lajiosuuksien estimaatteihin.

Kiitosmaininnat

Kiitos tutkielmani ohjaajalle Salme Kärkkäiselle. Kiitos Kristian Meissnerille ja Tuomas Turpeiselle tutkielman aineistosta sekä konsultoinnista biologisen seurannan suhteen. Kiitos Antti Penttiselle taustatuesta ja neuvoista. Kiitos Serkan Kiranyazille ja muille yhteistyökumppaneille mahdollisuudesta osallistua artikkeliin. Kiitos Johannes Ärjelle konsultoinnista.

Kirjallisuutta

[1] Bélanger, J. & Gagnon, D. (1993). Classification and proportion estimation. *Australian Journal of Statistics*, **35**(1), 19–28.

[2] Blaschko, M.B., Holness, G., Mattar, M.A., Lisin, D., Utgoff, P.E., Hanson, A.R., Schultz, H., Riseman, E.M., Sieracki, M.E., Balch, W.M. & Tupper, B. (2005). Automatic In Situ Identification of Plankton. *Proceedings of the 7th IEEE Workshops on Application of Computer Vision (WACV/MOTION '05)*, Vol. 1.

[3] Breiman, L. (2001). Random forests. *Machine Learning*, **45**(1), 5–32.

[4] Duda, R.O., Hart, P.E. & Stork, D.G. (2001). *Pattern Classification*, 2nd edition, John Wiley & Sons, Inc., New York.

[5] Fortier, J-J. (1992). Best linear corrector of classification estimates of proportions of objects in several unknown classes. *The Canadian Journal of Statistics*, **20**(1), 23–33.

[6] Hastie, T., Tibshirani, R. & Friedman, J. (2001), *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer-Verlag New York.

[7] Healy, J.D. (1981). The effects of misclassification error on the estimation of several population proportions. *Bell System Technical Journal*, **60**(5), 697–704.

[8] ImageJ-kuvankäsittelyohjelman kotisivu:
`http://rsbweb.nih.gov/ij/docs/menus/analyze.html`

[9] Johnson, R.A. & Wichern, D.W. (2002). *Applied Multivariate Statistical Analysis*, 5th edition, Prentice Hall, Upper Saddle River, New Jersey.

[10] Kiranyaz, S., Ince, T., Pulkkinen, J., Gabbouj, M., Ärje, J., Kärkkäinen S., Tirronen, V., Juhola M., Turpeinen, T. & Meissner, K. (2009). Classication and retrieval on macroinvertabrate image databases. Submitted manuscript.

[11] Krzanowski, W.J. & Marriot, F.H.C. (1994). *Multivariate Analysis, Part 1, Distributions, Ordination and Inference*, Edward Arnold, London.

[12] Oliver, S. (2008). MRes Interactive Intelligent Systems, PRII501 Project. Masters thesis. School of Computing, Communications and Electronics. Plymouth.

[13] Prinzie, A. & van den Poel, D. (2007). Random multiclassclassification: generalizing random forests to random mnl and random nb. Wagner, R., Revell, N. & Pernul, G., editors, *LNCS 4653*. Springer-Verlag Berlin Heidelberg.

[14] Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, **1**(1), 81–106.

[15] R Development Core Team (2009). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL `http://www.R-project.org`.

[16] Shannon, C.E. (1984). *Bell System Technical Journal*, **27**, 379–423, 623–656.

[17] Suomen ympäristökeskuksen Pintavesien luokittelu ekologisen ja kemiallisen tilan perusteella -sivusto:
`http://www.environment.fi/default.asp?node=22615\&lan=fi\#a0`

[18] Witten, I.H. & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition, Morgan Kaufmann Publishers, Amsterdam.

Liite A

R-koodit

A.1 Bayes-luokittelija

```
#BAYES-luokittelija
library(MASS)

#Funktio poimii pohjaeläinaineistosta N opetus- ja testiaineistoa.
#Se muodostaa kvadraattiset erottelufunktiot opetusaineiston pohjalta
#ja luokittelee niilä testiaineiston. Lopuksi se tulostaa saadut
#testivirheet.

QDA.simulointi1 <- function(data,N){
tulokset <- vector(length=N)
for(i in 1:N){
x <- sample(1:1350, 650, replace=FALSE)
opetus <- data[x,]
testi <- data[-x,]

kaava<-as.formula(paste("Laji~",paste(names(opetus[,2:16]), collapse="+")))
malli<-qda(kaava,prior=c(0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.125),
data=opetus)

ennuste.testi <- predict(malli,testi)$class

funktio<-function(dat,enn,n){
muisti<-NULL
for(i in 1:n){
if(dat$Laji[i] == ennuste.testi[i]){muisti[i]<-1}
else{muisti[i]<-0}
}
muisti
}
```

```
os.testi<-funktio(testi,ennuste.testi,700)
tulokset[i]<-1-sum(os.testi)/700
}
tulokset
}


#Sekaannusmatriisikorjaus Bayes-luokittelijalle

#Funktio, jota tarvitaan sekaannusmatriisin laskemiseksi
sekaannus <- function(tulokset, j){
N <- length(tulokset)
muisti <- NULL
for(i in 1:N){
if(tulokset[i] == j){
muisti[i] <- 1
}
else{muisti[i] <- 0}
}
sum(muisti)
}

#Funktiolle annetaan opetus- ja testiaineisto sekä näyte.
#Se tuottaa näytteen bootstrap-otoksesta lasketut oikeat lajiosuudet,
#niiden luokittelun tuloksena saadut estimaatit sekä
#sekaannusmatriisikorjatut estimaatit.

luokittelu <- function(data.op,data.te,data.na){
n1 <- length(data.na[,1])
X <- sample(1:n1,replace=TRUE)
NADAT <- data.na[X,]
n2 <- length(data.te[,1])
Y <- sample(1:n2,replace=TRUE)
TEDAT <- data.te[Y,]

opetus.oikeat <- data.op[,1]
testi.oikeat <- TEDAT[,1]
nayte.oikeat <- NADAT[,1]

muuttujat <- data.op[,2:16]
kaava <- as.formula(paste("Laji~",paste(names(muuttujat), collapse="+")))
malli<-qda(kaava,prior=c(0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.125),
data=data.op)
```

```
opetus.tulokset <- predict(malli,data.op)$class
testi.tulokset <- predict(malli,TEDAT)$class
nayte.tulokset <- predict(malli,NADAT)$class

br.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="BR"){br.lkm.te<-br.lkm.te+1}
}

dn.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="DN"){dn.lkm.te<-dn.lkm.te+1}
}
hs.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="HS"){hs.lkm.te<-hs.lkm.te+1}
}
hyp.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="HYP"){hyp.lkm.te<-hyp.lkm.te+1}
}
hys.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="HYS"){hys.lkm.te<-hys.lkm.te+1}
}
ip.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="IP"){ip.lkm.te<-ip.lkm.te+1}
}
rn.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="RN"){rn.lkm.te<-rn.lkm.te+1}
}
tn.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="TN"){tn.lkm.te<-tn.lkm.te+1}
}

BR.te<-matrix(nrow=br.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="BR"){
BR.te[k,j]<-TEDAT[i,j]}
}
```

```
if(TEDAT[i,]$Laji=="BR"){k<-k+1}
}
BR.te.lajit<-rep("BR",br.lkm.te)
Laji<-factor(BR.te.lajit)
X<-as.data.frame(BR.te)
BR.te<-cbind(Laji,X[,2:16])
BR.te<-as.data.frame(BR.te)
BR.te.oikeat<-BR.te[,1]
BR.te.tulokset<-predict(malli,BR.te)$class


DN.te<-matrix(nrow=dn.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="DN"){
DN.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="DN"){k<-k+1}
}
DN.te.lajit<-rep("DN",dn.lkm.te)
Laji<-factor(DN.te.lajit)
X<-as.data.frame(DN.te)
DN.te<-cbind(Laji,X[,2:16])
DN.te<-as.data.frame(DN.te)
DN.te.oikeat<-DN.te[,1]
DN.te.tulokset<-predict(malli,DN.te)$class



HS.te<-matrix(nrow=hs.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="HS"){
HS.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="HS"){k<-k+1}
}
HS.te.lajit<-rep("HS",hs.lkm.te)
Laji<-factor(HS.te.lajit)
X<-as.data.frame(HS.te)
HS.te<-cbind(Laji,X[,2:16])
HS.te<-as.data.frame(HS.te)
HS.te.oikeat<-HS.te[,1]
```

```
HS.te.tulokset<-predict(malli,HS.te)$class



HYP.te<-matrix(nrow=hyp.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="HYP"){
HYP.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="HYP"){k<-k+1}
}
HYP.te.lajit<-rep("HYP",hyp.lkm.te)
Laji<-factor(HYP.te.lajit)
X<-as.data.frame(HYP.te)
HYP.te<-cbind(Laji,X[,2:16])
HYP.te<-as.data.frame(HYP.te)
HYP.te.oikeat<-HYP.te[,1]
HYP.te.tulokset<-predict(malli,HYP.te)$class


HYS.te<-matrix(nrow=hys.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="HYS"){
HYS.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="HYS"){k<-k+1}
}
HYS.te.lajit<-rep("HYS",hys.lkm.te)
Laji<-factor(HYS.te.lajit)
X<-as.data.frame(HYS.te)
HYS.te<-cbind(Laji,X[,2:16])
HYS.te<-as.data.frame(HYS.te)
HYS.te.oikeat<-HYS.te[,1]
HYS.te.tulokset<-predict(malli,HYS.te)$class


IP.te<-matrix(nrow=ip.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="IP"){
```

```
IP.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="IP"){k<-k+1}
}
IP.te.lajit<-rep("IP",ip.lkm.te)
Laji<-factor(IP.te.lajit)
X<-as.data.frame(IP.te)
IP.te<-cbind(Laji,X[,2:16])
IP.te<-as.data.frame(IP.te)
IP.te.oikeat<-IP.te[,1]
IP.te.tulokset<-predict(malli,IP.te)$class


RN.te<-matrix(nrow=rn.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="RN"){
RN.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="RN"){k<-k+1}
}
RN.te.lajit<-rep("RN",rn.lkm.te)
Laji<-factor(RN.te.lajit)
X<-as.data.frame(RN.te)
RN.te<-cbind(Laji,X[,2:16])
RN.te<-as.data.frame(RN.te)
RN.te.oikeat<-RN.te[,1]
RN.te.tulokset<-predict(malli,RN.te)$class


TN.te<-matrix(nrow=tn.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="TN"){
TN.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="TN"){k<-k+1}
}
TN.te.lajit<-rep("TN",tn.lkm.te)
Laji<-factor(TN.te.lajit)
X<-as.data.frame(TN.te)
TN.te<-cbind(Laji,X[,2:16])
TN.te<-as.data.frame(TN.te)
```

```
TN.te.oikeat<-TN.te[,1]
TN.te.tulokset<-predict(malli,TN.te)$class


BR_BR <- sekaannus(BR.te.tulokset, "BR")
BR_DN <- sekaannus(BR.te.tulokset, "DN")
BR_HS <- sekaannus(BR.te.tulokset, "HS")
BR_HYP <- sekaannus(BR.te.tulokset, "HYP")
BR_HYS <- sekaannus(BR.te.tulokset, "HYS")
BR_IP <- sekaannus(BR.te.tulokset, "IP")
BR_RN <- sekaannus(BR.te.tulokset, "RN")
BR_TN <- sekaannus(BR.te.tulokset, "TN")

DN_BR <- sekaannus(DN.te.tulokset, "BR")
DN_DN <- sekaannus(DN.te.tulokset, "DN")
DN_HS <- sekaannus(DN.te.tulokset, "HS")
DN_HYP <- sekaannus(DN.te.tulokset, "HYP")
DN_HYS <- sekaannus(DN.te.tulokset, "HYS")
DN_IP <- sekaannus(DN.te.tulokset, "IP")
DN_RN <- sekaannus(DN.te.tulokset, "RN")
DN_TN <- sekaannus(DN.te.tulokset, "TN")

HS_BR <- sekaannus(HS.te.tulokset, "BR")
HS_DN <- sekaannus(HS.te.tulokset, "DN")
HS_HS <- sekaannus(HS.te.tulokset, "HS")
HS_HYP <- sekaannus(HS.te.tulokset, "HYP")
HS_HYS <- sekaannus(HS.te.tulokset, "HYS")
HS_IP <- sekaannus(HS.te.tulokset, "IP")
HS_RN <- sekaannus(HS.te.tulokset, "RN")
HS_TN <- sekaannus(HS.te.tulokset, "TN")

HYP_BR <- sekaannus(HYP.te.tulokset, "BR")
HYP_DN <- sekaannus(HYP.te.tulokset, "DN")
HYP_HS <- sekaannus(HYP.te.tulokset, "HS")
HYP_HYP <- sekaannus(HYP.te.tulokset, "HYP")
HYP_HYS <- sekaannus(HYP.te.tulokset, "HYS")
HYP_IP <- sekaannus(HYP.te.tulokset, "IP")
HYP_RN <- sekaannus(HYP.te.tulokset, "RN")
HYP_TN <- sekaannus(HYP.te.tulokset, "TN")

HYS_BR <- sekaannus(HYS.te.tulokset, "BR")
HYS_DN <- sekaannus(HYS.te.tulokset, "DN")
HYS_HS <- sekaannus(HYS.te.tulokset, "HS")
HYS_HYP <- sekaannus(HYS.te.tulokset, "HYP")
HYS_HYS <- sekaannus(HYS.te.tulokset, "HYS")
```

```
HYS_IP <- sekaannus(HYS.te.tulokset, "IP")
HYS_RN <- sekaannus(HYS.te.tulokset, "RN")
HYS_TN <- sekaannus(HYS.te.tulokset, "TN")

IP_BR <- sekaannus(IP.te.tulokset, "BR")
IP_DN <- sekaannus(IP.te.tulokset, "DN")
IP_HS <- sekaannus(IP.te.tulokset, "HS")
IP_HYP <- sekaannus(IP.te.tulokset, "HYP")
IP_HYS <- sekaannus(IP.te.tulokset, "HYS")
IP_IP <- sekaannus(IP.te.tulokset, "IP")
IP_RN <- sekaannus(IP.te.tulokset, "RN")
IP_TN <- sekaannus(IP.te.tulokset, "TN")

RN_BR <- sekaannus(RN.te.tulokset, "BR")
RN_DN <- sekaannus(RN.te.tulokset, "DN")
RN_HS <- sekaannus(RN.te.tulokset, "HS")
RN_HYP <- sekaannus(RN.te.tulokset, "HYP")
RN_HYS <- sekaannus(RN.te.tulokset, "HYS")
RN_IP <- sekaannus(RN.te.tulokset, "IP")
RN_RN <- sekaannus(RN.te.tulokset, "RN")
RN_TN <- sekaannus(RN.te.tulokset, "TN")

TN_BR <- sekaannus(TN.te.tulokset, "BR")
TN_DN <- sekaannus(TN.te.tulokset, "DN")
TN_HS <- sekaannus(TN.te.tulokset, "HS")
TN_HYP <- sekaannus(TN.te.tulokset, "HYP")
TN_HYS <- sekaannus(TN.te.tulokset, "HYS")
TN_IP <- sekaannus(TN.te.tulokset, "IP")
TN_RN <- sekaannus(TN.te.tulokset, "RN")
TN_TN <- sekaannus(TN.te.tulokset, "TN")

confusion <- matrix(c(BR_BR,BR_DN,BR_HS,BR_HYP,BR_HYS,BR_IP,BR_RN,
    BR_TN,DN_BR,DN_DN,DN_HS,DN_HYP,DN_HYS,DN_IP,DN_RN,DN_TN,
HS_BR,HS_DN,HS_HS,HS_HYP,HS_HYS,HS_IP,HS_RN,HS_TN,
HYP_BR,HYP_DN,HYP_HS,HYP_HYP,HYP_HYS,HYP_IP,HYP_RN,HYP_TN,
HYS_BR,HYS_DN,HYS_HS,HYS_HYP,HYS_HYS,HYS_IP,HYS_RN,HYS_TN,
IP_BR,IP_DN,IP_HS,IP_HYP,IP_HYS,IP_IP,IP_RN,IP_TN,
RN_BR,RN_DN,RN_HS,RN_HYP,RN_HYS,RN_IP,RN_RN,RN_TN,
TN_BR,TN_DN,TN_HS,TN_HYP,TN_HYS,TN_IP,TN_RN,TN_TN),
ncol=8, byrow=TRUE)

P.BR_BR <- confusion[1,1]/sum(confusion[1,])
P.BR_DN <- confusion[1,2]/sum(confusion[1,])
P.BR_HS <- confusion[1,3]/sum(confusion[1,])
P.BR_HYP <- confusion[1,4]/sum(confusion[1,])
```

```
P.BR_HYS <- confusion[1,5]/sum(confusion[1,])
P.BR_IP <- confusion[1,6]/sum(confusion[1,])
P.BR_RN <- confusion[1,7]/sum(confusion[1,])
P.BR_TN <- confusion[1,8]/sum(confusion[1,])
P.DN_BR <- confusion[2,1]/sum(confusion[2,])
P.DN_DN <- confusion[2,2]/sum(confusion[2,])
P.DN_HS <- confusion[2,3]/sum(confusion[2,])
P.DN_HYP <- confusion[2,4]/sum(confusion[2,])
P.DN_HYS <- confusion[2,5]/sum(confusion[2,])
P.DN_IP <- confusion[2,6]/sum(confusion[2,])
P.DN_RN <- confusion[2,7]/sum(confusion[2,])
P.DN_TN <- confusion[2,8]/sum(confusion[2,])
P.HS_BR <- confusion[3,1]/sum(confusion[3,])
P.HS_DN <- confusion[3,2]/sum(confusion[3,])
P.HS_HS <- confusion[3,3]/sum(confusion[3,])
P.HS_HYP <- confusion[3,4]/sum(confusion[3,])
P.HS_HYS <- confusion[3,5]/sum(confusion[3,])
P.HS_IP <- confusion[3,6]/sum(confusion[3,])
P.HS_RN <- confusion[3,7]/sum(confusion[3,])
P.HS_TN <- confusion[3,8]/sum(confusion[3,])
P.HYP_BR <- confusion[4,1]/sum(confusion[4,])
P.HYP_DN <- confusion[4,2]/sum(confusion[4,])
P.HYP_HS <- confusion[4,3]/sum(confusion[4,])
P.HYP_HYP <- confusion[4,4]/sum(confusion[4,])
P.HYP_HYS <- confusion[4,5]/sum(confusion[4,])
P.HYP_IP <- confusion[4,6]/sum(confusion[4,])
P.HYP_RN <- confusion[4,7]/sum(confusion[4,])
P.HYP_TN <- confusion[4,8]/sum(confusion[4,])
P.HYS_BR <- confusion[5,1]/sum(confusion[5,])
P.HYS_DN <- confusion[5,2]/sum(confusion[5,])
P.HYS_HS <- confusion[5,3]/sum(confusion[5,])
P.HYS_HYP <- confusion[5,4]/sum(confusion[5,])
P.HYS_HYS <- confusion[5,5]/sum(confusion[5,])
P.HYS_IP <- confusion[5,6]/sum(confusion[5,])
P.HYS_RN <- confusion[5,7]/sum(confusion[5,])
P.HYS_TN <- confusion[5,8]/sum(confusion[5,])
P.IP_BR <- confusion[6,1]/sum(confusion[6,])
P.IP_DN <- confusion[6,2]/sum(confusion[6,])
P.IP_HS <- confusion[6,3]/sum(confusion[6,])
P.IP_HYP <- confusion[6,4]/sum(confusion[6,])
P.IP_HYS <- confusion[6,5]/sum(confusion[6,])
P.IP_IP <- confusion[6,6]/sum(confusion[6,])
P.IP_RN <- confusion[6,7]/sum(confusion[6,])
P.IP_TN <- confusion[6,8]/sum(confusion[6,])
P.RN_BR <- confusion[7,1]/sum(confusion[7,])
```

```
P.RN_DN <- confusion[7,2]/sum(confusion[7,])
P.RN_HS <- confusion[7,3]/sum(confusion[7,])
P.RN_HYP <- confusion[7,4]/sum(confusion[7,])
P.RN_HYS <- confusion[7,5]/sum(confusion[7,])
P.RN_IP <- confusion[7,6]/sum(confusion[7,])
P.RN_RN <- confusion[7,7]/sum(confusion[7,])
P.RN_TN <- confusion[7,8]/sum(confusion[7,])
P.TN_BR <- confusion[8,1]/sum(confusion[8,])
P.TN_DN <- confusion[8,2]/sum(confusion[8,])
P.TN_HS <- confusion[8,3]/sum(confusion[8,])
P.TN_HYP <- confusion[8,4]/sum(confusion[8,])
P.TN_HYS <- confusion[8,5]/sum(confusion[8,])
P.TN_IP <- confusion[8,6]/sum(confusion[8,])
P.TN_RN <- confusion[8,7]/sum(confusion[8,])
P.TN_TN <- confusion[8,8]/sum(confusion[8,])


confusion.p <- t(matrix(c(P.BR_BR,P.BR_DN,P.BR_HS,P.BR_HYP,P.BR_HYS,P.BR_IP,
    P.BR_RN,P.BR_TN,P.DN_BR,P.DN_DN,P.DN_HS,P.DN_HYP,P.DN_HYS,P.DN_IP,P.DN_RN,
    P.DN_TN,P.HS_BR,P.HS_DN,P.HS_HS,P.HS_HYP,P.HS_HYS,P.HS_IP,P.HS_RN,P.HS_TN,
P.HYP_BR,P.HYP_DN,P.HYP_HS,P.HYP_HYP,P.HYP_HYS,P.HYP_IP,P.HYP_RN,P.HYP_TN,
P.HYS_BR,P.HYS_DN,P.HYS_HS,P.HYS_HYP,P.HYS_HYS,P.HYS_IP,P.HYS_RN,P.HYS_TN,
P.IP_BR,P.IP_DN,P.IP_HS,P.IP_HYP,P.IP_HYS,P.IP_IP,P.IP_RN,P.IP_TN,
P.RN_BR,P.RN_DN,P.RN_HS,P.RN_HYP,P.RN_HYS,P.RN_IP,P.RN_RN,P.RN_TN,
P.TN_BR,P.TN_DN,P.TN_HS,P.TN_HYP,P.TN_HYS,P.TN_IP,P.TN_RN,P.TN_TN),
ncol=8, byrow=TRUE))

nayte.BR.osuus <- sekaannus(nayte.tulokset, "BR")/450
nayte.DN.osuus <- sekaannus(nayte.tulokset, "DN")/450
nayte.HS.osuus <- sekaannus(nayte.tulokset, "HS")/450
nayte.HYP.osuus <- sekaannus(nayte.tulokset, "HYP")/450
nayte.HYS.osuus <- sekaannus(nayte.tulokset, "HYS")/450
nayte.IP.osuus <- sekaannus(nayte.tulokset, "IP")/450
nayte.RN.osuus <- sekaannus(nayte.tulokset, "RN")/450
nayte.TN.osuus <- sekaannus(nayte.tulokset, "TN")/450

P.tulokset <- matrix(c(nayte.BR.osuus,nayte.DN.osuus,nayte.HS.osuus,
      nayte.HYP.osuus,nayte.HYS.osuus,nayte.IP.osuus,nayte.RN.osuus,
      nayte.TN.osuus),ncol=1,byrow=TRUE)

PI <- solve(confusion.p)%*%P.tulokset

br.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="BR"){br.lkm.na<-br.lkm.na+1}
```

```
}
dn.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="DN"){dn.lkm.na<-dn.lkm.na+1}
}
hs.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="HS"){hs.lkm.na<-hs.lkm.na+1}
}
hyp.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="HYP"){hyp.lkm.na<-hyp.lkm.na+1}
}
hys.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="HYS"){hys.lkm.na<-hys.lkm.na+1}
}
ip.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="IP"){ip.lkm.na<-ip.lkm.na+1}
}
rn.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="RN"){rn.lkm.na<-rn.lkm.na+1}
}
tn.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="TN"){tn.lkm.na<-tn.lkm.na+1}
}

P.oikeat <- matrix(c(br.lkm.na/450,dn.lkm.na/450,hs.lkm.na/450,hyp.lkm.na/450,
hys.lkm.na/450,ip.lkm.na/450,rn.lkm.na/450,tn.lkm.na/450),ncol=1,byrow=TRUE)

G <- cbind(P.oikeat,P.tulokset,PI)

list(tulos=G)
}
```

A.2 C4.5-päätöspuu

```
#C4.5-päätöspuu
library(RWeka)

#Funktio poimii pohjaeläinaineistosta N opetus- ja testiaineistoa.
#Se muodostaa opetusaineiston pohjalta C4.5-päätöspuut
#ja luokittelee niilä testiaineiston. Lopuksi se tulostaa saadut
#testivirheet.

J48.simulointi1<-function(data,N){
tulokset<-vector(length=N)
for(i in 1:N){
x<-sample(1:1350,650,replace=FALSE)
opetus<-data[x,]
testi<-data[-x,]

kaava<-as.formula(paste("Laji~",paste(names(opetus[,2:16]), collapse="+")))
malli<-J48(kaava,data=opetus)

ennuste.testi<-predict(malli,testi)

funktio<-function(dat,enn,n){
muisti<-NULL
for(i in 1:n){
if(dat$Laji[i] == ennuste.testi[i]){muisti[i]<-1}
else{muisti[i]<-0}
}
muisti
}

os.testi<-funktio(testi,ennuste.testi,700)
tulokset[i]<-1-sum(os.testi)/700
}
tulokset
}


#Sekaannusmatriisikorjaus C4.5-päätöspuulle

#Funktio, jota tarvitaan sekaannusmatriisin laskemiseksi


sekaannus <- function(tulokset, j){
N <- length(tulokset)
```

```
muisti <- NULL
for(i in 1:N){
if(tulokset[i] == j){
muisti[i] <- 1
}
else{muisti[i] <- 0}
}
sum(muisti)
}

#Funktiolle annetaan opetus- ja testiaineisto sekä näyte.
#Se tuottaa näytteen bootstrap-otoksesta lasketut oikeat lajiosuudet,
#niiden luokittelun tuloksena saadut estimaatit sekä
#sekaannusmatriisikorjatut estimaatit.

luokittelu <- function(data.op, data.te, data.na){
n1 <- length(data.na[,1])
X <- sample(1:n1,replace=TRUE)
NADAT <- data.na[X,]
n2 <- length(data.te[,1])
Y <- sample(1:n2,replace=TRUE)
TEDAT <- data.te[Y,]

opetus.oikeat <- data.op[,1]
testi.oikeat <- TEDAT[,1]
nayte.oikeat <- NADAT[,1]

muuttujat <- data.op[,2:16]
kaava <- as.formula(paste("Laji~",paste(names(muuttujat), collapse="+")))
malli <- J48(kaava, data=data.op)

opetus.tulokset <- predict(malli,data.op)
testi.tulokset <- predict(malli,TEDAT)
nayte.tulokset <- predict(malli,NADAT)

br.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="BR"){br.lkm.te<-br.lkm.te+1}
}

dn.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="DN"){dn.lkm.te<-dn.lkm.te+1}
}
hs.lkm.te<-0
```

```
for(i in 1:450){
if(testi.oikeat[i]=="HS"){hs.lkm.te<-hs.lkm.te+1}
}
hyp.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="HYP"){hyp.lkm.te<-hyp.lkm.te+1}
}
hys.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="HYS"){hys.lkm.te<-hys.lkm.te+1}
}
ip.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="IP"){ip.lkm.te<-ip.lkm.te+1}
}
rn.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="RN"){rn.lkm.te<-rn.lkm.te+1}
}
tn.lkm.te<-0
for(i in 1:450){
if(testi.oikeat[i]=="TN"){tn.lkm.te<-tn.lkm.te+1}
}

BR.te<-matrix(nrow=br.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="BR"){
BR.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="BR"){k<-k+1}
}
BR.te.lajit<-rep("BR",br.lkm.te)
Laji<-factor(BR.te.lajit)
X<-as.data.frame(BR.te)
BR.te<-cbind(Laji,X[,2:16])
BR.te<-as.data.frame(BR.te)
BR.te.oikeat<-BR.te[,1]
BR.te.tulokset<-predict(malli,BR.te)


DN.te<-matrix(nrow=dn.lkm.te,ncol=16)
k<-1
for(i in 1:450){
```

```
for(j in 1:16){
if(TEDAT[i,]$Laji=="DN"){
DN.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="DN"){k<-k+1}
}
DN.te.lajit<-rep("DN",dn.lkm.te)
Laji<-factor(DN.te.lajit)
X<-as.data.frame(DN.te)
DN.te<-cbind(Laji,X[,2:16])
DN.te<-as.data.frame(DN.te)
DN.te.oikeat<-DN.te[,1]
DN.te.tulokset<-predict(malli,DN.te)



HS.te<-matrix(nrow=hs.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="HS"){
HS.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="HS"){k<-k+1}
}
HS.te.lajit<-rep("HS",hs.lkm.te)
Laji<-factor(HS.te.lajit)
X<-as.data.frame(HS.te)
HS.te<-cbind(Laji,X[,2:16])
HS.te<-as.data.frame(HS.te)
HS.te.oikeat<-HS.te[,1]
HS.te.tulokset<-predict(malli,HS.te)



HYP.te<-matrix(nrow=hyp.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="HYP"){
HYP.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="HYP"){k<-k+1}
}
HYP.te.lajit<-rep("HYP",hyp.lkm.te)
```

```
Laji<-factor(HYP.te.lajit)
X<-as.data.frame(HYP.te)
HYP.te<-cbind(Laji,X[,2:16])
HYP.te<-as.data.frame(HYP.te)
HYP.te.oikeat<-HYP.te[,1]
HYP.te.tulokset<-predict(malli,HYP.te)


HYS.te<-matrix(nrow=hys.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="HYS"){
HYS.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="HYS"){k<-k+1}
}
HYS.te.lajit<-rep("HYS",hys.lkm.te)
Laji<-factor(HYS.te.lajit)
X<-as.data.frame(HYS.te)
HYS.te<-cbind(Laji,X[,2:16])
HYS.te<-as.data.frame(HYS.te)
HYS.te.oikeat<-HYS.te[,1]
HYS.te.tulokset<-predict(malli,HYS.te)


IP.te<-matrix(nrow=ip.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="IP"){
IP.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="IP"){k<-k+1}
}
IP.te.lajit<-rep("IP",ip.lkm.te)
Laji<-factor(IP.te.lajit)
X<-as.data.frame(IP.te)
IP.te<-cbind(Laji,X[,2:16])
IP.te<-as.data.frame(IP.te)
IP.te.oikeat<-IP.te[,1]
IP.te.tulokset<-predict(malli,IP.te)


RN.te<-matrix(nrow=rn.lkm.te,ncol=16)
```

```
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="RN"){
RN.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="RN"){k<-k+1}
}
RN.te.lajit<-rep("RN",rn.lkm.te)
Laji<-factor(RN.te.lajit)
X<-as.data.frame(RN.te)
RN.te<-cbind(Laji,X[,2:16])
RN.te<-as.data.frame(RN.te)
RN.te.oikeat<-RN.te[,1]
RN.te.tulokset<-predict(malli,RN.te)


TN.te<-matrix(nrow=tn.lkm.te,ncol=16)
k<-1
for(i in 1:450){
for(j in 1:16){
if(TEDAT[i,]$Laji=="TN"){
TN.te[k,j]<-TEDAT[i,j]}
}
if(TEDAT[i,]$Laji=="TN"){k<-k+1}
}
TN.te.lajit<-rep("TN",tn.lkm.te)
Laji<-factor(TN.te.lajit)
X<-as.data.frame(TN.te)
TN.te<-cbind(Laji,X[,2:16])
TN.te<-as.data.frame(TN.te)
TN.te.oikeat<-TN.te[,1]
TN.te.tulokset<-predict(malli,TN.te)


BR_BR <- sekaannus(BR.te.tulokset, "BR")
BR_DN <- sekaannus(BR.te.tulokset, "DN")
BR_HS <- sekaannus(BR.te.tulokset, "HS")
BR_HYP <- sekaannus(BR.te.tulokset, "HYP")
BR_HYS <- sekaannus(BR.te.tulokset, "HYS")
BR_IP <- sekaannus(BR.te.tulokset, "IP")
BR_RN <- sekaannus(BR.te.tulokset, "RN")
BR_TN <- sekaannus(BR.te.tulokset, "TN")

DN_BR <- sekaannus(DN.te.tulokset, "BR")
```

```
DN_DN <- sekaannus(DN.te.tulokset, "DN")
DN_HS <- sekaannus(DN.te.tulokset, "HS")
DN_HYP <- sekaannus(DN.te.tulokset, "HYP")
DN_HYS <- sekaannus(DN.te.tulokset, "HYS")
DN_IP <- sekaannus(DN.te.tulokset, "IP")
DN_RN <- sekaannus(DN.te.tulokset, "RN")
DN_TN <- sekaannus(DN.te.tulokset, "TN")

HS_BR <- sekaannus(HS.te.tulokset, "BR")
HS_DN <- sekaannus(HS.te.tulokset, "DN")
HS_HS <- sekaannus(HS.te.tulokset, "HS")
HS_HYP <- sekaannus(HS.te.tulokset, "HYP")
HS_HYS <- sekaannus(HS.te.tulokset, "HYS")
HS_IP <- sekaannus(HS.te.tulokset, "IP")
HS_RN <- sekaannus(HS.te.tulokset, "RN")
HS_TN <- sekaannus(HS.te.tulokset, "TN")

HYP_BR <- sekaannus(HYP.te.tulokset, "BR")
HYP_DN <- sekaannus(HYP.te.tulokset, "DN")
HYP_HS <- sekaannus(HYP.te.tulokset, "HS")
HYP_HYP <- sekaannus(HYP.te.tulokset, "HYP")
HYP_HYS <- sekaannus(HYP.te.tulokset, "HYS")
HYP_IP <- sekaannus(HYP.te.tulokset, "IP")
HYP_RN <- sekaannus(HYP.te.tulokset, "RN")
HYP_TN <- sekaannus(HYP.te.tulokset, "TN")

HYS_BR <- sekaannus(HYS.te.tulokset, "BR")
HYS_DN <- sekaannus(HYS.te.tulokset, "DN")
HYS_HS <- sekaannus(HYS.te.tulokset, "HS")
HYS_HYP <- sekaannus(HYS.te.tulokset, "HYP")
HYS_HYS <- sekaannus(HYS.te.tulokset, "HYS")
HYS_IP <- sekaannus(HYS.te.tulokset, "IP")
HYS_RN <- sekaannus(HYS.te.tulokset, "RN")
HYS_TN <- sekaannus(HYS.te.tulokset, "TN")

IP_BR <- sekaannus(IP.te.tulokset, "BR")
IP_DN <- sekaannus(IP.te.tulokset, "DN")
IP_HS <- sekaannus(IP.te.tulokset, "HS")
IP_HYP <- sekaannus(IP.te.tulokset, "HYP")
IP_HYS <- sekaannus(IP.te.tulokset, "HYS")
IP_IP <- sekaannus(IP.te.tulokset, "IP")
IP_RN <- sekaannus(IP.te.tulokset, "RN")
IP_TN <- sekaannus(IP.te.tulokset, "TN")

RN_BR <- sekaannus(RN.te.tulokset, "BR")
```

```
RN_DN <- sekaannus(RN.te.tulokset, "DN")
RN_HS <- sekaannus(RN.te.tulokset, "HS")
RN_HYP <- sekaannus(RN.te.tulokset, "HYP")
RN_HYS <- sekaannus(RN.te.tulokset, "HYS")
RN_IP <- sekaannus(RN.te.tulokset, "IP")
RN_RN <- sekaannus(RN.te.tulokset, "RN")
RN_TN <- sekaannus(RN.te.tulokset, "TN")

TN_BR <- sekaannus(TN.te.tulokset, "BR")
TN_DN <- sekaannus(TN.te.tulokset, "DN")
TN_HS <- sekaannus(TN.te.tulokset, "HS")
TN_HYP <- sekaannus(TN.te.tulokset, "HYP")
TN_HYS <- sekaannus(TN.te.tulokset, "HYS")
TN_IP <- sekaannus(TN.te.tulokset, "IP")
TN_RN <- sekaannus(TN.te.tulokset, "RN")
TN_TN <- sekaannus(TN.te.tulokset, "TN")

confusion <- matrix(c(BR_BR,BR_DN,BR_HS,BR_HYP,BR_HYS,BR_IP,
BR_RN,BR_TN,DN_BR,DN_DN,DN_HS,DN_HYP,DN_HYS,DN_IP,DN_RN,DN_TN,
HS_BR,HS_DN,HS_HS,HS_HYP,HS_HYS,HS_IP,HS_RN,HS_TN,
HYP_BR,HYP_DN,HYP_HS,HYP_HYP,HYP_HYS,HYP_IP,HYP_RN,HYP_TN,
HYS_BR,HYS_DN,HYS_HS,HYS_HYP,HYS_HYS,HYS_IP,HYS_RN,HYS_TN,
IP_BR,IP_DN,IP_HS,IP_HYP,IP_HYS,IP_IP,IP_RN,IP_TN,
RN_BR,RN_DN,RN_HS,RN_HYP,RN_HYS,RN_IP,RN_RN,RN_TN,
TN_BR,TN_DN,TN_HS,TN_HYP,TN_HYS,TN_IP,TN_RN,TN_TN),
ncol=8, byrow=TRUE)

P.BR_BR <- confusion[1,1]/sum(confusion[1,])
P.BR_DN <- confusion[1,2]/sum(confusion[1,])
P.BR_HS <- confusion[1,3]/sum(confusion[1,])
P.BR_HYP <- confusion[1,4]/sum(confusion[1,])
P.BR_HYS <- confusion[1,5]/sum(confusion[1,])
P.BR_IP <- confusion[1,6]/sum(confusion[1,])
P.BR_RN <- confusion[1,7]/sum(confusion[1,])
P.BR_TN <- confusion[1,8]/sum(confusion[1,])
P.DN_BR <- confusion[2,1]/sum(confusion[2,])
P.DN_DN <- confusion[2,2]/sum(confusion[2,])
P.DN_HS <- confusion[2,3]/sum(confusion[2,])
P.DN_HYP <- confusion[2,4]/sum(confusion[2,])
P.DN_HYS <- confusion[2,5]/sum(confusion[2,])
P.DN_IP <- confusion[2,6]/sum(confusion[2,])
P.DN_RN <- confusion[2,7]/sum(confusion[2,])
P.DN_TN <- confusion[2,8]/sum(confusion[2,])
P.HS_BR <- confusion[3,1]/sum(confusion[3,])
P.HS_DN <- confusion[3,2]/sum(confusion[3,])
```

```
P.HS_HS <- confusion[3,3]/sum(confusion[3,])
P.HS_HYP <- confusion[3,4]/sum(confusion[3,])
P.HS_HYS <- confusion[3,5]/sum(confusion[3,])
P.HS_IP <- confusion[3,6]/sum(confusion[3,])
P.HS_RN <- confusion[3,7]/sum(confusion[3,])
P.HS_TN <- confusion[3,8]/sum(confusion[3,])
P.HYP_BR <- confusion[4,1]/sum(confusion[4,])
P.HYP_DN <- confusion[4,2]/sum(confusion[4,])
P.HYP_HS <- confusion[4,3]/sum(confusion[4,])
P.HYP_HYP <- confusion[4,4]/sum(confusion[4,])
P.HYP_HYS <- confusion[4,5]/sum(confusion[4,])
P.HYP_IP <- confusion[4,6]/sum(confusion[4,])
P.HYP_RN <- confusion[4,7]/sum(confusion[4,])
P.HYP_TN <- confusion[4,8]/sum(confusion[4,])
P.HYS_BR <- confusion[5,1]/sum(confusion[5,])
P.HYS_DN <- confusion[5,2]/sum(confusion[5,])
P.HYS_HS <- confusion[5,3]/sum(confusion[5,])
P.HYS_HYP <- confusion[5,4]/sum(confusion[5,])
P.HYS_HYS <- confusion[5,5]/sum(confusion[5,])
P.HYS_IP <- confusion[5,6]/sum(confusion[5,])
P.HYS_RN <- confusion[5,7]/sum(confusion[5,])
P.HYS_TN <- confusion[5,8]/sum(confusion[5,])
P.IP_BR <- confusion[6,1]/sum(confusion[6,])
P.IP_DN <- confusion[6,2]/sum(confusion[6,])
P.IP_HS <- confusion[6,3]/sum(confusion[6,])
P.IP_HYP <- confusion[6,4]/sum(confusion[6,])
P.IP_HYS <- confusion[6,5]/sum(confusion[6,])
P.IP_IP <- confusion[6,6]/sum(confusion[6,])
P.IP_RN <- confusion[6,7]/sum(confusion[6,])
P.IP_TN <- confusion[6,8]/sum(confusion[6,])
P.RN_BR <- confusion[7,1]/sum(confusion[7,])
P.RN_DN <- confusion[7,2]/sum(confusion[7,])
P.RN_HS <- confusion[7,3]/sum(confusion[7,])
P.RN_HYP <- confusion[7,4]/sum(confusion[7,])
P.RN_HYS <- confusion[7,5]/sum(confusion[7,])
P.RN_IP <- confusion[7,6]/sum(confusion[7,])
P.RN_RN <- confusion[7,7]/sum(confusion[7,])
P.RN_TN <- confusion[7,8]/sum(confusion[7,])
P.TN_BR <- confusion[8,1]/sum(confusion[8,])
P.TN_DN <- confusion[8,2]/sum(confusion[8,])
P.TN_HS <- confusion[8,3]/sum(confusion[8,])
P.TN_HYP <- confusion[8,4]/sum(confusion[8,])
P.TN_HYS <- confusion[8,5]/sum(confusion[8,])
P.TN_IP <- confusion[8,6]/sum(confusion[8,])
P.TN_RN <- confusion[8,7]/sum(confusion[8,])
```

```
P.TN_TN <- confusion[8,8]/sum(confusion[8,])


confusion.p <- t(matrix(c(P.BR_BR,P.BR_DN,P.BR_HS,P.BR_HYP,P.BR_HYS,
P.BR_IP,P.BR_RN,P.BR_TN,P.DN_BR,P.DN_DN,P.DN_HS,P.DN_HYP,P.DN_HYS,
P.DN_IP,P.DN_RN,P.DN_TN,P.HS_BR,P.HS_DN,P.HS_HS,P.HS_HYP,P.HS_HYS,
P.HS_IP,P.HS_RN,P.HS_TN,P.HYP_BR,P.HYP_DN,P.HYP_HS,P.HYP_HYP,P.HYP_HYS,
P.HYP_IP,P.HYP_RN,P.HYP_TN,P.HYS_BR,P.HYS_DN,P.HYS_HS,P.HYS_HYP,
P.HYS_HYS,P.HYS_IP,P.HYS_RN,P.HYS_TN,P.IP_BR,P.IP_DN,P.IP_HS,P.IP_HYP,
P.IP_HYS,P.IP_IP,P.IP_RN,P.IP_TN,P.RN_BR,P.RN_DN,P.RN_HS,P.RN_HYP,
P.RN_HYS,P.RN_IP,P.RN_RN,P.RN_TN,P.TN_BR,P.TN_DN,P.TN_HS,P.TN_HYP,
P.TN_HYS,P.TN_IP,P.TN_RN,P.TN_TN),ncol=8, byrow=TRUE))

nayte.BR.osuus <- sekaannus(nayte.tulokset, "BR")/450
nayte.DN.osuus <- sekaannus(nayte.tulokset, "DN")/450
nayte.HS.osuus <- sekaannus(nayte.tulokset, "HS")/450
nayte.HYP.osuus <- sekaannus(nayte.tulokset, "HYP")/450
nayte.HYS.osuus <- sekaannus(nayte.tulokset, "HYS")/450
nayte.IP.osuus <- sekaannus(nayte.tulokset, "IP")/450
nayte.RN.osuus <- sekaannus(nayte.tulokset, "RN")/450
nayte.TN.osuus <- sekaannus(nayte.tulokset, "TN")/450

P.tulokset <- matrix(c(nayte.BR.osuus,nayte.DN.osuus,nayte.HS.osuus,
nayte.HYP.osuus,nayte.HYS.osuus,nayte.IP.osuus,nayte.RN.osuus,
nayte.TN.osuus),ncol=1,byrow=TRUE)

PI <- solve(confusion.p)%*%P.tulokset

br.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="BR"){br.lkm.na<-br.lkm.na+1}
}
dn.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="DN"){dn.lkm.na<-dn.lkm.na+1}
}
hs.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="HS"){hs.lkm.na<-hs.lkm.na+1}
}
hyp.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="HYP"){hyp.lkm.na<-hyp.lkm.na+1}
}
hys.lkm.na<-0
```

```
for(i in 1:450){
if(nayte.oikeat[i]=="HYS"){hys.lkm.na<-hys.lkm.na+1}
}
ip.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="IP"){ip.lkm.na<-ip.lkm.na+1}
}
rn.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="RN"){rn.lkm.na<-rn.lkm.na+1}
}
tn.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="TN"){tn.lkm.na<-tn.lkm.na+1}
}

P.oikeat <- matrix(c(br.lkm.na/450,dn.lkm.na/450,hs.lkm.na/450,hyp.lkm.na/450,
hys.lkm.na/450,ip.lkm.na/450,rn.lkm.na/450,tn.lkm.na/450),ncol=1,byrow=TRUE)

G <- cbind(P.oikeat,P.tulokset,PI)

list(tulos=G)
}
```

A.3 Satunnainen metsä

```
#Satunnainen metsä
library(randomForest)

#Funktio muodostaa N satunnaista metsää, jotka kaikki koostuvat
#n päätöspuusta. Solmumuuttujat valitaan aina m satunnaisen muuttujan
#joukosta. Metsät luokittelevat kaikki havainnot puilla, joiden
#rakentamiseen niitä ei käytetty ja tulostaa out-of-bag -virheet.

RF.simulointi1<-function(data,N,n,m){
tulos<-NULL
kaava<-as.formula(paste("Laji~",paste(names(data[,2:16]), collapse="+")))
for(i in 1:N){
malli<-randomForest(formula=kaava,data=data,ntree=n,mtry=m,
classwt=c(0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.125),replace=TRUE)
tulos[i]<-malli$err.rate[n,1]
}
tulos
}

#Sekaannusmatriisikorjaus satunnaiselle metsälle

#Funktiota tarvitaan oikeiden lajiosuuksien laskemiseen

sekaannus <- function(tulokset, j){
N <- length(tulokset)
muisti <- NULL
for(i in 1:N){
if(tulokset[i] == j){
muisti[i] <- 1
}
else{muisti[i] <- 0}
}
sum(muisti)
}

#Funktiolle annetaan opetusaineisto sekä näyte.
#Se tuottaa näytteen bootstrap-otoksesta lasketut oikeat lajiosuudet,
#niiden luokittelun tuloksena saadut estimaatit sekä
#sekaannusmatriisikorjatut estimaatit. Satunnainen metsä ei tarvitse
#erillistä testiaineistoa.

luokittelu<-function(data.op,data.na){
n1<-length(data.na[,1])
```

```
X<-sample(1:n1,replace=TRUE)
NADAT<-data.na[X,]
nayte.oikeat<-NADAT[,1]

x.op<-data.op[,2:16]
y.op<-data.op[,1]
x.na<-NADAT[,2:16]
malli<-randomForest(x=x.op,y=y.op,mtry=5,ntree=70,
classwt=c(0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.125),replace=TRUE,
xtest=x.na)
oob.pred<-malli$predicted
confusion<-malli$confusion[,1:8]
nayte.tulokset<-malli$test$predicted

P.BR_BR <- confusion[1,1]/sum(confusion[1,])
P.BR_DN <- confusion[1,2]/sum(confusion[1,])
P.BR_HS <- confusion[1,3]/sum(confusion[1,])
P.BR_HYP <- confusion[1,4]/sum(confusion[1,])
P.BR_HYS <- confusion[1,5]/sum(confusion[1,])
P.BR_IP <- confusion[1,6]/sum(confusion[1,])
P.BR_RN <- confusion[1,7]/sum(confusion[1,])
P.BR_TN <- confusion[1,8]/sum(confusion[1,])
P.DN_BR <- confusion[2,1]/sum(confusion[2,])
P.DN_DN <- confusion[2,2]/sum(confusion[2,])
P.DN_HS <- confusion[2,3]/sum(confusion[2,])
P.DN_HYP <- confusion[2,4]/sum(confusion[2,])
P.DN_HYS <- confusion[2,5]/sum(confusion[2,])
P.DN_IP <- confusion[2,6]/sum(confusion[2,])
P.DN_RN <- confusion[2,7]/sum(confusion[2,])
P.DN_TN <- confusion[2,8]/sum(confusion[2,])
P.HS_BR <- confusion[3,1]/sum(confusion[3,])
P.HS_DN <- confusion[3,2]/sum(confusion[3,])
P.HS_HS <- confusion[3,3]/sum(confusion[3,])
P.HS_HYP <- confusion[3,4]/sum(confusion[3,])
P.HS_HYS <- confusion[3,5]/sum(confusion[3,])
P.HS_IP <- confusion[3,6]/sum(confusion[3,])
P.HS_RN <- confusion[3,7]/sum(confusion[3,])
P.HS_TN <- confusion[3,8]/sum(confusion[3,])
P.HYP_BR <- confusion[4,1]/sum(confusion[4,])
P.HYP_DN <- confusion[4,2]/sum(confusion[4,])
P.HYP_HS <- confusion[4,3]/sum(confusion[4,])
P.HYP_HYP <- confusion[4,4]/sum(confusion[4,])
P.HYP_HYS <- confusion[4,5]/sum(confusion[4,])
P.HYP_IP <- confusion[4,6]/sum(confusion[4,])
P.HYP_RN <- confusion[4,7]/sum(confusion[4,])
```

```
P.HYP_TN <- confusion[4,8]/sum(confusion[4,])
P.HYS_BR <- confusion[5,1]/sum(confusion[5,])
P.HYS_DN <- confusion[5,2]/sum(confusion[5,])
P.HYS_HS <- confusion[5,3]/sum(confusion[5,])
P.HYS_HYP <- confusion[5,4]/sum(confusion[5,])
P.HYS_HYS <- confusion[5,5]/sum(confusion[5,])
P.HYS_IP <- confusion[5,6]/sum(confusion[5,])
P.HYS_RN <- confusion[5,7]/sum(confusion[5,])
P.HYS_TN <- confusion[5,8]/sum(confusion[5,])
P.IP_BR <- confusion[6,1]/sum(confusion[6,])
P.IP_DN <- confusion[6,2]/sum(confusion[6,])
P.IP_HS <- confusion[6,3]/sum(confusion[6,])
P.IP_HYP <- confusion[6,4]/sum(confusion[6,])
P.IP_HYS <- confusion[6,5]/sum(confusion[6,])
P.IP_IP <- confusion[6,6]/sum(confusion[6,])
P.IP_RN <- confusion[6,7]/sum(confusion[6,])
P.IP_TN <- confusion[6,8]/sum(confusion[6,])
P.RN_BR <- confusion[7,1]/sum(confusion[7,])
P.RN_DN <- confusion[7,2]/sum(confusion[7,])
P.RN_HS <- confusion[7,3]/sum(confusion[7,])
P.RN_HYP <- confusion[7,4]/sum(confusion[7,])
P.RN_HYS <- confusion[7,5]/sum(confusion[7,])
P.RN_IP <- confusion[7,6]/sum(confusion[7,])
P.RN_RN <- confusion[7,7]/sum(confusion[7,])
P.RN_TN <- confusion[7,8]/sum(confusion[7,])
P.TN_BR <- confusion[8,1]/sum(confusion[8,])
P.TN_DN <- confusion[8,2]/sum(confusion[8,])
P.TN_HS <- confusion[8,3]/sum(confusion[8,])
P.TN_HYP <- confusion[8,4]/sum(confusion[8,])
P.TN_HYS <- confusion[8,5]/sum(confusion[8,])
P.TN_IP <- confusion[8,6]/sum(confusion[8,])
P.TN_RN <- confusion[8,7]/sum(confusion[8,])
P.TN_TN <- confusion[8,8]/sum(confusion[8,])


confusion.p <- t(matrix(c(P.BR_BR,P.BR_DN,P.BR_HS,P.BR_HYP,P.BR_HYS,P.BR_IP,
P.BR_RN,P.BR_TN,P.DN_BR,P.DN_DN,P.DN_HS,P.DN_HYP,P.DN_HYS,P.DN_IP,P.DN_RN,
P.DN_TN,P.HS_BR,P.HS_DN,P.HS_HS,P.HS_HYP,P.HS_HYS,P.HS_IP,P.HS_RN,P.HS_TN,
P.HYP_BR,P.HYP_DN,P.HYP_HS,P.HYP_HYP,P.HYP_HYS,P.HYP_IP,P.HYP_RN,P.HYP_TN,
P.HYS_BR,P.HYS_DN,P.HYS_HS,P.HYS_HYP,P.HYS_HYS,P.HYS_IP,P.HYS_RN,P.HYS_TN,
P.IP_BR,P.IP_DN,P.IP_HS,P.IP_HYP,P.IP_HYS,P.IP_IP,P.IP_RN,P.IP_TN,
P.RN_BR,P.RN_DN,P.RN_HS,P.RN_HYP,P.RN_HYS,P.RN_IP,P.RN_RN,P.RN_TN,
P.TN_BR,P.TN_DN,P.TN_HS,P.TN_HYP,P.TN_HYS,P.TN_IP,P.TN_RN,P.TN_TN),
ncol=8, byrow=TRUE))
```

```
nayte.BR.osuus <- sekaannus(nayte.tulokset, "BR")/450
nayte.DN.osuus <- sekaannus(nayte.tulokset, "DN")/450
nayte.HS.osuus <- sekaannus(nayte.tulokset, "HS")/450
nayte.HYP.osuus <- sekaannus(nayte.tulokset, "HYP")/450
nayte.HYS.osuus <- sekaannus(nayte.tulokset, "HYS")/450
nayte.IP.osuus <- sekaannus(nayte.tulokset, "IP")/450
nayte.RN.osuus <- sekaannus(nayte.tulokset, "RN")/450
nayte.TN.osuus <- sekaannus(nayte.tulokset, "TN")/450

P.tulokset <- matrix(c(nayte.BR.osuus,nayte.DN.osuus,nayte.HS.osuus,
nayte.HYP.osuus,nayte.HYS.osuus,nayte.IP.osuus,nayte.RN.osuus,nayte.TN.osuus)
,ncol=1,byrow=TRUE)

PI <- solve(confusion.p)%*%P.tulokset

br.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="BR"){br.lkm.na<-br.lkm.na+1}
}
dn.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="DN"){dn.lkm.na<-dn.lkm.na+1}
}
hs.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="HS"){hs.lkm.na<-hs.lkm.na+1}
}
hyp.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="HYP"){hyp.lkm.na<-hyp.lkm.na+1}
}
hys.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="HYS"){hys.lkm.na<-hys.lkm.na+1}
}
ip.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="IP"){ip.lkm.na<-ip.lkm.na+1}
}
rn.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="RN"){rn.lkm.na<-rn.lkm.na+1}
}
tn.lkm.na<-0
for(i in 1:450){
```

```
if(nayte.oikeat[i]=="TN"){tn.lkm.na<-tn.lkm.na+1}
}

P.oikeat <- matrix(c(br.lkm.na/450,dn.lkm.na/450,hs.lkm.na/450,hyp.lkm.na/450,
hys.lkm.na/450,ip.lkm.na/450,rn.lkm.na/450,tn.lkm.na/450),ncol=1,byrow=TRUE)

G <- cbind(P.oikeat,P.tulokset,PI)

list(tulos=G)
}
```

A.4 Satunnainen Bayes-metsä

```
#Satunnainen Bayes-metsä
library(MASS)

#Funktio laskee, kuinka moni ennustetuista luokista menee
#oikein.

osumat<-function(data,pred){
summa<-0
n<-0
N<-length(data[,1])
for(i in 1:N){
if(is.na(pred[i])==FALSE){
if(data$Laji[i]==pred[i]){
summa<-summa+1
n<-n+1
}
else{n<-n+1}
}
}
summa/n
}

muuntaja<-function(v){
n<-length(v)
for(i in 1:n){
if(is.na(v[i])==FALSE){
if(v[i]==1){v[i]<-"Baetisrhodani"}
if(v[i]==2){v[i]<-"Diurananseni"}
if(v[i]==3){v[i]<-"Heptageniasulphurea"}
if(v[i]==4){v[i]<-"Hydropsychepellucidulla"}
if(v[i]==5){v[i]<-"Hydropsychesiltalai"}
if(v[i]==6){v[i]<-"Isoperla"}
if(v[i]==7){v[i]<-"Rhyacophilanubila"}
if(v[i]==8){v[i]<-"Taeniopteryxnebulose"}
}
else{v[i]<-NA}
}
v
}

#Funktio muodostaa N satunnaista Bayes-metsää, jotka kaikki koostuvat
#n satunnaisesta Bayes-luokittelijasta. Jokaiseen luokittelijaan
#arvotaan m satunnaista piirrettä. Metsät luokittelevat kaikki havainnot
```

```
#Bayes-luokittelijoilla, joiden rakentamiseen niitä ei käytetty ja
#tulostaa out-of-bag -virheet.

RB.simulointi1<-function(data,N,m,k){
tulos<-NULL
opetus.oikeat<-data[,1]
n<-length(data[,1])
for(i in 1:N){
oob.tulokset<-matrix(nrow=n,ncol=k)
oob.ennuste<-NULL
for(j in 1:k){
x<-sample(1:n,n,replace=TRUE)
opetus<-data[x,]
oob<-data[-x,]
w<-(1:1350)
luvut<-w[-x]
muuttujat<-sample(opetus[,2:16],m)
kaava<-as.formula(paste("Laji~",paste(names(muuttujat), collapse="+")))
malli<-qda(kaava,prior=c(0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.125),
data=opetus)
for(l in 1:length(luvut)){
oob.tulokset[luvut[l],j]<-predict(malli,oob)$class[l]
}
}

for(o in 1:n){
if(is.null(names(which.max(table(oob.tulokset[o,])))==FALSE){
oob.ennuste[o]<-names(which.max(table(oob.tulokset[o,])))}
else{oob.ennuste[o]<-NA}
}

oob.ennuste<-muuntaja(oob.ennuste)
tulos[i]<-1-osumat(data,oob.ennuste)
}
tulos
}

#Sekaannusmatriisikorjaus satunnaiselle Bayes-metsälle

#Funktiota tarvitaan sekaannusmatriisien laskemiseen

sekaannus <- function(tulokset, j){
N <- length(tulokset)
summa <- 0
for(i in 1:N){
```

```
if(is.na(tulokset[i])==FALSE){
if(tulokset[i] == j){
summa <- summa+1
}
}
}
summa
}

#Funktiolle annetaan opetusaineisto sekä näyte.
#Se tuottaa näytteen bootstrap-otoksesta lasketut oikeat lajiosuudet,
#niiden luokittelun tuloksena saadut estimaatit sekä
#sekaannusmatriisikorjatut estimaatit. Satunnainen Bayes-metsä ei
#tarvitse erillistä testiaineistoa.

luokittelu<-function(data.op,data.na,m,k){
oikeat<-data.op[,1]
n<-length(data.op[,1])
n2<-length(data.na[,1])
oob.tulokset<-matrix(nrow=n,ncol=k)
oob.ennuste<-NULL
X<-sample(1:n2,n2,replace=TRUE)
NADAT<-data.na[X,]
nayte.tulokset<-matrix(nrow=n2,ncol=k)
nayte.ennuste<-NULL
nayte.oikeat<-NADAT[,1]

for(j in 1:k){
x<-sample(1:n,n,replace=TRUE)
opetus<-data.op[x,]
oob<-data.op[-x,]
w<-(1:900)
luvut<-w[-x]
muuttujat<-sample(opetus[,2:16],m)
kaava<-as.formula(paste("Laji~",paste(names(muuttujat), collapse="+")))
malli<-qda(kaava,prior=c(0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.125),
data=opetus)
for(l in 1:length(luvut)){
oob.tulokset[luvut[l],j]<-predict(malli,oob)$class[l]
}
for(i in 1:n2){
nayte.tulokset[i,j]<-predict(malli,NADAT)$class[i]
}
}
```

```
for(o in 1:n){
if(is.null(names(which.max(table(oob.tulokset[o,]))))==FALSE){
oob.ennuste[o]<-names(which.max(table(oob.tulokset[o,])))}
else{oob.ennuste[o]<-NA}
}
for(o in 1:n2){
nayte.ennuste[o]<-names(which.max(table(nayte.tulokset[o,])))
}

oob.ennuste<-muuntaja(oob.ennuste)
nayte.ennuste<-muuntaja(nayte.ennuste)

BR_BR <- sekaannus(oob.ennuste[1:77], "BR")
BR_DN <- sekaannus(oob.ennuste[1:77], "DN")
BR_HS <- sekaannus(oob.ennuste[1:77], "HS")
BR_HYP <- sekaannus(oob.ennuste[1:77], "HYP")
BR_HYS <- sekaannus(oob.ennuste[1:77], "HYS")
BR_IP <- sekaannus(oob.ennuste[1:77], "IP")
BR_RN <- sekaannus(oob.ennuste[1:77], "RN")
BR_TN <- sekaannus(oob.ennuste[1:77], "TN")

DN_BR <- sekaannus(oob.ennuste[78:163], "BR")
DN_DN <- sekaannus(oob.ennuste[78:163], "DN")
DN_HS <- sekaannus(oob.ennuste[78:163], "HS")
DN_HYP <- sekaannus(oob.ennuste[78:163], "HYP")
DN_HYS <- sekaannus(oob.ennuste[78:163], "HYS")
DN_IP <- sekaannus(oob.ennuste[78:163], "IP")
DN_RN <- sekaannus(oob.ennuste[78:163], "RN")
DN_TN <- sekaannus(oob.ennuste[78:163], "TN")

HS_BR <- sekaannus(oob.ennuste[164:278], "BR")
HS_DN <- sekaannus(oob.ennuste[164:278], "DN")
HS_HS <- sekaannus(oob.ennuste[164:278], "HS")
HS_HYP <- sekaannus(oob.ennuste[164:278], "HYP")
HS_HYS <- sekaannus(oob.ennuste[164:278], "HYS")
HS_IP <- sekaannus(oob.ennuste[164:278], "IP")
HS_RN <- sekaannus(oob.ennuste[164:278], "RN")
HS_TN <- sekaannus(oob.ennuste[164:278], "TN")

HYP_BR <- sekaannus(oob.ennuste[279:346], "BR")
HYP_DN <- sekaannus(oob.ennuste[279:346], "DN")
HYP_HS <- sekaannus(oob.ennuste[279:346], "HS")
HYP_HYP <- sekaannus(oob.ennuste[279:346], "HYP")
HYP_HYS <- sekaannus(oob.ennuste[279:346], "HYS")
HYP_IP <- sekaannus(oob.ennuste[279:346], "IP")
```

```
HYP_RN <- sekaannus(oob.ennuste[279:346], "RN")
HYP_TN <- sekaannus(oob.ennuste[279:346], "TN")

HYS_BR <- sekaannus(oob.ennuste[347:527], "BR")
HYS_DN <- sekaannus(oob.ennuste[347:527], "DN")
HYS_HS <- sekaannus(oob.ennuste[347:527], "HS")
HYS_HYP <- sekaannus(oob.ennuste[347:527], "HYP")
HYS_HYS <- sekaannus(oob.ennuste[347:527], "HYS")
HYS_IP <- sekaannus(oob.ennuste[347:527], "IP")
HYS_RN <- sekaannus(oob.ennuste[347:527], "RN")
HYS_TN <- sekaannus(oob.ennuste[347:527], "TN")

IP_BR <- sekaannus(oob.ennuste[528:734], "BR")
IP_DN <- sekaannus(oob.ennuste[528:734], "DN")
IP_HS <- sekaannus(oob.ennuste[528:734], "HS")
IP_HYP <- sekaannus(oob.ennuste[528:734], "HYP")
IP_HYS <- sekaannus(oob.ennuste[528:734], "HYS")
IP_IP <- sekaannus(oob.ennuste[528:734], "IP")
IP_RN <- sekaannus(oob.ennuste[528:734], "RN")
IP_TN <- sekaannus(oob.ennuste[528:734], "TN")

RN_BR <- sekaannus(oob.ennuste[735:789], "BR")
RN_DN <- sekaannus(oob.ennuste[735:789], "DN")
RN_HS <- sekaannus(oob.ennuste[735:789], "HS")
RN_HYP <- sekaannus(oob.ennuste[735:789], "HYP")
RN_HYS <- sekaannus(oob.ennuste[735:789], "HYS")
RN_IP <- sekaannus(oob.ennuste[735:789], "IP")
RN_RN <- sekaannus(oob.ennuste[735:789], "RN")
RN_TN <- sekaannus(oob.ennuste[735:789], "TN")

TN_BR <- sekaannus(oob.ennuste[790:900], "BR")
TN_DN <- sekaannus(oob.ennuste[790:900], "DN")
TN_HS <- sekaannus(oob.ennuste[790:900], "HS")
TN_HYP <- sekaannus(oob.ennuste[790:900], "HYP")
TN_HYS <- sekaannus(oob.ennuste[790:900], "HYS")
TN_IP <- sekaannus(oob.ennuste[790:900], "IP")
TN_RN <- sekaannus(oob.ennuste[790:900], "RN")
TN_TN <- sekaannus(oob.ennuste[790:900], "TN")

confusion <- matrix(c(BR_BR,BR_DN,BR_HS,BR_HYP,BR_HYS,BR_IP,
BR_RN,BR_TN,DN_BR,DN_DN,DN_HS,DN_HYP,DN_HYS,DN_IP,DN_RN,DN_TN,
HS_BR,HS_DN,HS_HS,HS_HYP,HS_HYS,HS_IP,HS_RN,HS_TN,
HYP_BR,HYP_DN,HYP_HS,HYP_HYP,HYP_HYS,HYP_IP,HYP_RN,HYP_TN,
HYS_BR,HYS_DN,HYS_HS,HYS_HYP,HYS_HYS,HYS_IP,HYS_RN,HYS_TN,
IP_BR,IP_DN,IP_HS,IP_HYP,IP_HYS,IP_IP,IP_RN,IP_TN,
```

```
RN_BR,RN_DN,RN_HS,RN_HYP,RN_HYS,RN_IP,RN_RN,RN_TN,
TN_BR,TN_DN,TN_HS,TN_HYP,TN_HYS,TN_IP,TN_RN,TN_TN),
ncol=8, byrow=TRUE)

P.BR_BR <- confusion[1,1]/sum(confusion[1,])
P.BR_DN <- confusion[1,2]/sum(confusion[1,])
P.BR_HS <- confusion[1,3]/sum(confusion[1,])
P.BR_HYP <- confusion[1,4]/sum(confusion[1,])
P.BR_HYS <- confusion[1,5]/sum(confusion[1,])
P.BR_IP <- confusion[1,6]/sum(confusion[1,])
P.BR_RN <- confusion[1,7]/sum(confusion[1,])
P.BR_TN <- confusion[1,8]/sum(confusion[1,])
P.DN_BR <- confusion[2,1]/sum(confusion[2,])
P.DN_DN <- confusion[2,2]/sum(confusion[2,])
P.DN_HS <- confusion[2,3]/sum(confusion[2,])
P.DN_HYP <- confusion[2,4]/sum(confusion[2,])
P.DN_HYS <- confusion[2,5]/sum(confusion[2,])
P.DN_IP <- confusion[2,6]/sum(confusion[2,])
P.DN_RN <- confusion[2,7]/sum(confusion[2,])
P.DN_TN <- confusion[2,8]/sum(confusion[2,])
P.HS_BR <- confusion[3,1]/sum(confusion[3,])
P.HS_DN <- confusion[3,2]/sum(confusion[3,])
P.HS_HS <- confusion[3,3]/sum(confusion[3,])
P.HS_HYP <- confusion[3,4]/sum(confusion[3,])
P.HS_HYS <- confusion[3,5]/sum(confusion[3,])
P.HS_IP <- confusion[3,6]/sum(confusion[3,])
P.HS_RN <- confusion[3,7]/sum(confusion[3,])
P.HS_TN <- confusion[3,8]/sum(confusion[3,])
P.HYP_BR <- confusion[4,1]/sum(confusion[4,])
P.HYP_DN <- confusion[4,2]/sum(confusion[4,])
P.HYP_HS <- confusion[4,3]/sum(confusion[4,])
P.HYP_HYP <- confusion[4,4]/sum(confusion[4,])
P.HYP_HYS <- confusion[4,5]/sum(confusion[4,])
P.HYP_IP <- confusion[4,6]/sum(confusion[4,])
P.HYP_RN <- confusion[4,7]/sum(confusion[4,])
P.HYP_TN <- confusion[4,8]/sum(confusion[4,])
P.HYS_BR <- confusion[5,1]/sum(confusion[5,])
P.HYS_DN <- confusion[5,2]/sum(confusion[5,])
P.HYS_HS <- confusion[5,3]/sum(confusion[5,])
P.HYS_HYP <- confusion[5,4]/sum(confusion[5,])
P.HYS_HYS <- confusion[5,5]/sum(confusion[5,])
P.HYS_IP <- confusion[5,6]/sum(confusion[5,])
P.HYS_RN <- confusion[5,7]/sum(confusion[5,])
P.HYS_TN <- confusion[5,8]/sum(confusion[5,])
P.IP_BR <- confusion[6,1]/sum(confusion[6,])
```

```
P.IP_DN <- confusion[6,2]/sum(confusion[6,])
P.IP_HS <- confusion[6,3]/sum(confusion[6,])
P.IP_HYP <- confusion[6,4]/sum(confusion[6,])
P.IP_HYS <- confusion[6,5]/sum(confusion[6,])
P.IP_IP <- confusion[6,6]/sum(confusion[6,])
P.IP_RN <- confusion[6,7]/sum(confusion[6,])
P.IP_TN <- confusion[6,8]/sum(confusion[6,])
P.RN_BR <- confusion[7,1]/sum(confusion[7,])
P.RN_DN <- confusion[7,2]/sum(confusion[7,])
P.RN_HS <- confusion[7,3]/sum(confusion[7,])
P.RN_HYP <- confusion[7,4]/sum(confusion[7,])
P.RN_HYS <- confusion[7,5]/sum(confusion[7,])
P.RN_IP <- confusion[7,6]/sum(confusion[7,])
P.RN_RN <- confusion[7,7]/sum(confusion[7,])
P.RN_TN <- confusion[7,8]/sum(confusion[7,])
P.TN_BR <- confusion[8,1]/sum(confusion[8,])
P.TN_DN <- confusion[8,2]/sum(confusion[8,])
P.TN_HS <- confusion[8,3]/sum(confusion[8,])
P.TN_HYP <- confusion[8,4]/sum(confusion[8,])
P.TN_HYS <- confusion[8,5]/sum(confusion[8,])
P.TN_IP <- confusion[8,6]/sum(confusion[8,])
P.TN_RN <- confusion[8,7]/sum(confusion[8,])
P.TN_TN <- confusion[8,8]/sum(confusion[8,])


confusion.p <- t(matrix(c(P.BR_BR,P.BR_DN,P.BR_HS,P.BR_HYP,P.BR_HYS,
P.BR_IP,P.BR_RN,P.BR_TN,P.DN_BR,P.DN_DN,P.DN_HS,P.DN_HYP,P.DN_HYS,
P.DN_IP,P.DN_RN,P.DN_TN,P.HS_BR,P.HS_DN,P.HS_HS,P.HS_HYP,P.HS_HYS,
P.HS_IP,P.HS_RN,P.HS_TN,P.HYP_BR,P.HYP_DN,P.HYP_HS,P.HYP_HYP,P.HYP_HYS,
P.HYP_IP,P.HYP_RN,P.HYP_TN,P.HYS_BR,P.HYS_DN,P.HYS_HS,P.HYS_HYP,
P.HYS_HYS,P.HYS_IP,P.HYS_RN,P.HYS_TN,P.IP_BR,P.IP_DN,P.IP_HS,P.IP_HYP,
P.IP_HYS,P.IP_IP,P.IP_RN,P.IP_TN,P.RN_BR,P.RN_DN,P.RN_HS,P.RN_HYP,
P.RN_HYS,P.RN_IP,P.RN_RN,P.RN_TN,P.TN_BR,P.TN_DN,P.TN_HS,P.TN_HYP,
P.TN_HYS,P.TN_IP,P.TN_RN,P.TN_TN),ncol=8, byrow=TRUE))

nayte.BR.osuus <- sekaannus(nayte.ennuste, "BR")/450
nayte.DN.osuus <- sekaannus(nayte.ennuste, "DN")/450
nayte.HS.osuus <- sekaannus(nayte.ennuste, "HS")/450
nayte.HYP.osuus <- sekaannus(nayte.ennuste, "HYP")/450
nayte.HYS.osuus <- sekaannus(nayte.ennuste, "HYS")/450
nayte.IP.osuus <- sekaannus(nayte.ennuste, "IP")/450
nayte.RN.osuus <- sekaannus(nayte.ennuste, "RN")/450
nayte.TN.osuus <- sekaannus(nayte.ennuste, "TN")/450

P.tulokset <- matrix(c(nayte.BR.osuus,nayte.DN.osuus,nayte.HS.osuus,
```

```
nayte.HYP.osuus,nayte.HYS.osuus,nayte.IP.osuus,nayte.RN.osuus,nayte.TN.osuus),
ncol=1,byrow=TRUE)

PI <- solve(confusion.p)%*%P.tulokset

br.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="BR"){br.lkm.na<-br.lkm.na+1}
}
dn.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="DN"){dn.lkm.na<-dn.lkm.na+1}
}
hs.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="HS"){hs.lkm.na<-hs.lkm.na+1}
}
hyp.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="HYP"){hyp.lkm.na<-hyp.lkm.na+1}
}
hys.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="HYS"){hys.lkm.na<-hys.lkm.na+1}
}
ip.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="IP"){ip.lkm.na<-ip.lkm.na+1}
}
rn.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="RN"){rn.lkm.na<-rn.lkm.na+1}
}
tn.lkm.na<-0
for(i in 1:450){
if(nayte.oikeat[i]=="TN"){tn.lkm.na<-tn.lkm.na+1}
}

P.oikeat <- matrix(c(br.lkm.na/450,dn.lkm.na/450,hs.lkm.na/450,hyp.lkm.na/450,
hys.lkm.na/450,ip.lkm.na/450,rn.lkm.na/450,tn.lkm.na/450),ncol=1,byrow=TRUE)

G <- cbind(P.oikeat,P.tulokset,PI)

list(tulos=G)
}
```