Gradienttitehostetut päätöspuut

Aleksi Korpua

Pro gradu -tutkielma
Joulukuu 2019

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Aleksi Korpua: Gradienttitehostetut päätöspuut
Pro gradu -tutkielma, 42 s., 6 liites.
Sovellettu matematiikka
Joulukuu 2019

Tässä Pro Gradu -tutkielmassa esitetään XGBoost -koneoppimisalgoritmi ja sovelletaan sitä autovahinko- ja elinajanodoteaineistoihin. Lisäksi algoritmille tehdään suorituskykytesti asuntomyynti -aineistolla. Algoritmi luokittelee aineiston gradienttitehostetuilla päätöspuilla ja tekee ennusteen tämän perusteella. Tutkielman tavoitteena on selvittää, voiko XGBoost -algoritmia käyttää ennustamiseen.

Tutkielmassa käytetään Kagglen "car insurance claim" ja "Brooklynhomes2003to2017" -aineistoja sekä WHO:n "life-expectancy-who" -aineistoa. Kaikki aineistot ovat avoimia ja saatavana Kagglen internetsivuilla. Aineistojen käsittely ja mallin sovitus tehdään Python-ohjelman avulla.

Tutkielman perusteella algoritmi soveltuu ennustamiseen tyydyttävällä luotettavuudella. Algoritmin suorituskyky oli erinomainen jopa isolla aineistolla.

Asiasanat: xgboost, gradienttitehostaminen, päätöspuu, koneoppiminen, regressio, luokittelu, kvantiililuonnos, suorituskyky, ennustaminen, LaTeX-ladontajärjestelmä.

Sisältö

1 Johdanto

Vakuutusalan yksi suurimmista kysymyksistä on ajoneuvovakuutusten hinnoittelu, joka koostuu kahdesta vaikuttavasta tekijästä. Nämä hintaan vaikuttavat tekijät ovat vahingon todennäköisyys ja vahingon suuruus. Vahingon todennäköisyys kertoo, kuinka monta vahinkoa henkilölle tapahtuu vuoden aikana. Vahingon suuruus kertoo kuinka monta euroa nämä vahingot maksavat korjauttaa. Monesti näitä pyritään ennustamaan jakaumiin perustuvilla tilastollisilla malleilla, kuten Tweedie-mallilla. Tässä tutkielmassa lähestytään ongelmaa koneoppimismallilla ja yritetään ennustaa vahingon suuruutta.

Tämän Pro gradu -tutkielman tavoitteena on ennustaa autovakuutuksista koituva vahinkomeno koneoppimisalgoritmilla. Koneoppimisella pyritään selittämään ilmiöitä, joiden ratkaisemiseen ei ole suoraa algoritmia, tai ilmiöitä mistä on saatavilla paljon vaikeasti tulkittavaa dataa. Autovakuutuksissa saadaan paljon dataa autosta, sekä jos asiakas on ollut pitkään vakuutuksenottajana myös historiaa vanhoista vahingoista. Vaikka tässä tutkielmassa pyritään ennustamaan vain tulevaa vahinkomenoa, niin silti taustalla algoritmi pyrkii myös ennustamaan vahingon todennäköisyyttä.

Tutkielman menetelmät painottuvat gradienttitehostettuihin päätöspuihin ja etenkin XGBoost -algoritmiin. Päätöspuut jaottelevat datan ennalta määrätyn jakokriteerin mukaan samantyyppisiin ryhmiin. Nämä ryhmät pisteytetään jollain häviöfunktiolla ja näistä pisteytyksistä tehdään ennuste. Tässä tutkielmassa käytetään XGBoost -algoritmin Python implementointia ja tutkitaan myös algoritmin tehokkuutta eri asetuksilla. Kaikki tutkielmassa käytetyt aineistot ovat Kagglen avoimia ja ilmaisia aineistoja. Kaggle on Googlen omistama palvelu joka tarjoaa avointa dataa, kilpailuja ja oppimismateriaaleja koneoppimisen harrastajille.

Tässä tutkielmassa on viisi osiota. Kappaleissa kaksi ja kolme perehdytään koneoppimisen ja päätöspuiden teoriaan ja sanastoon. Kappale neljä keskittyy XGBoost -algoritmin teoriaan ja toteutukseen. Kappaleissa viisi ja kuusi esitetään kaksi esimerkkiä Pythonin Xgboost paketilla toteutettuna. Kuudennessa kappaleessa tutkitaan Pythonin Xgboost implementoinnin tehokkuutta kahdella eri puunjakoalgoritmilla. Viimeisessä kappaleessa vedetään yhteen tutkielmasta saadut tulokset.

2 Koneoppiminen

Koneoppimisessa tietokone ohjelmoidaan optimoimaan *suorituskriteeri* käyttämällä esimerkki *dataa* tai menneisyyden esimerkkiä. *Oppimista* käytetään silloin kun ongelman ratkaisemiseen ei ole suoraa algoritmia, mutta ilmiöstä saatavilla on dataa tai vanhoja kokemuksia. Oppiminen on välttämätöntä kun ilmiöstä ei ole ihmisasiantuntemusta tai jos asiantuntemuksen selittäminen on mahdotonta. Esimerkiksi *puheentunnistamisessa* akustinen puhesignaali muutetaan *ASCII*-tekstiksi. Ihmiset pystyvät tekemään tämän tehtävän helposti, mutta eivät pysty selittämään miten sen tekee. Koneoppimisella siis pyritään selittämään monimutkaisia ilmiöitä, joihin ei ole selvää ratkaisualgoritmia [1].

2.1 Luokittelu

Luotto on finanssilaitoksen tai pankin lainaamaa rahaa, joka maksetaan takaisin korolla. On tärkeää, että pankki pystyy ennustamaan lainaan liittyvän *riskin*, joka on todennäköisyys asiakkaan kyvyttömyydelle maksaa koko laina takaisin. Tämä on luokitteluongelma, jossa pankki haluaa varmistaa, ettei anna maksukyvyttömälle lainanottajalle lainaa.

Luottopistesysteemi on pankkien käyttämä luokittelumenetelmä, joka laskee asiakkaan aikaisemmasta luotosta ja muusta informaatiosta lainan riskin. Asiakkaan informaatio sisältää kaiken oleellisen tiedon mitä asiakkaasta on saatavilla maksukyvyn määrittämiseen. Näitä tietoja voivat olla esimerkiksi tulot, säästöt, takaukset, ammatti, ikä, aikaisempi maksuhistoria ja niin edelleen. Näistä tiedoista tavoitteena on tehdä yleispätevä ohjeistus, jolla koneoppimisalgoritmi sovittaa mallin aikaisempien asiakkaiden dataan ja pyrkii luokittelemaan uudet lainaa hakevat asiakkaat maksukyvyn mukaan.

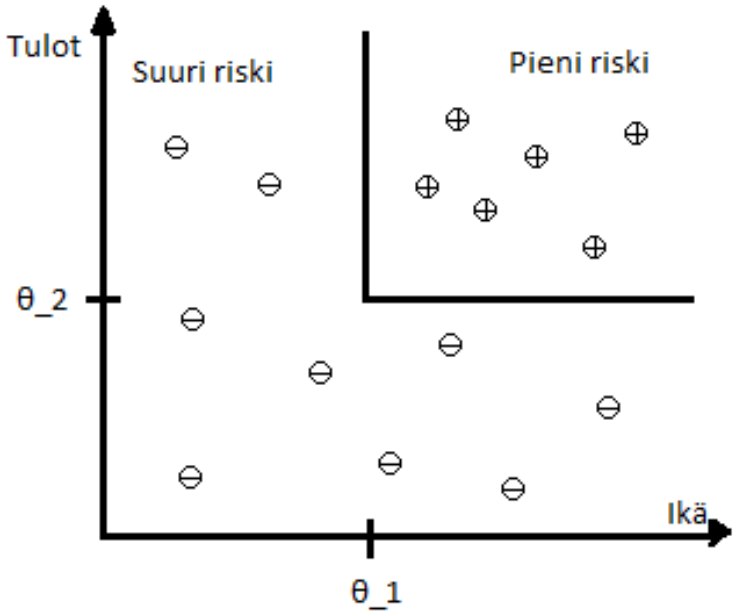
Tässä esimerkissä on kyse *luokitteluongelmasta*, jossa on kaksi luokkaa: pienen riskin ja suuren riskin asiakkaat. Asiakkaan informaatio on *luokittimen* syöte, jonka tehtävä on määrittää asiakas jompaankumpaan luokkaan syötteen perusteella.

Algoritmin *opetuksen* jälkeen luokittelusääntö on

$$IF \text{ ikä} > \theta_1 AND \text{ tulot} > \theta_2 THEN \text{ pieni riski } ELSE \text{ suuri riski.}$$

Luokkien raja *diskriminantti* muodostuu θ_1 ja θ_2 avulla (kuva 1).

Kun diskriminantti on muodostettu, voidaan *ennustaa* asiakkaiden riskiluokkaa. Kun luokittelualgoritmille on opetettu sääntö, jonka perusteella se luokittelee aikaisemman datan, niin voidaan ennustaa tulevia asiakkaita, jos tulevaisuus on samanlainen kuin menneisyys. Jos tulee uusi tietyn ikäinen asiakas tietyillä tuloilla, niin voidaan ennustaa, kuuluuko hän pienen vai suuren riskin luokkaan.



Kuva 1: Luokittelu

2.2 Regressio

Jos halutaan tehdä algoritmi, joka ennustaa käytetyn auton hinnan sen hintaan vaikuttavista ominaisuuksista, voidaan tehdä ennuste regressiolla.

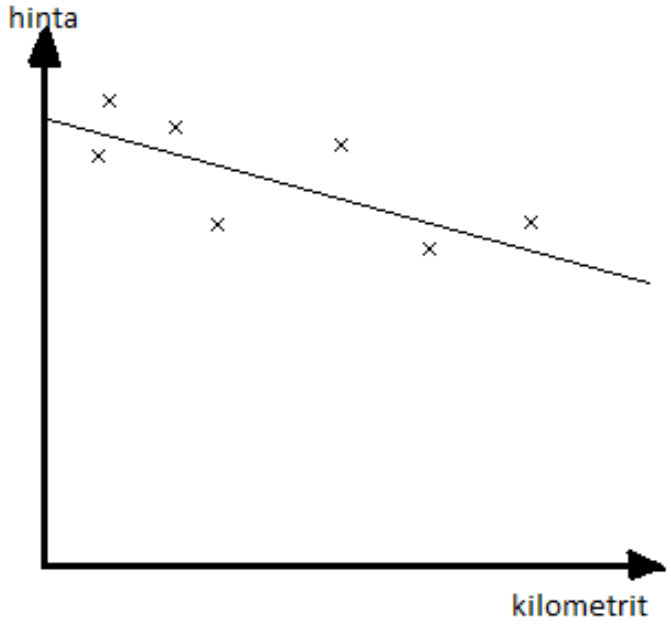
Olkoon X auton hintaan vaikuttavat muuttujat ja Y auton hinta. Käytetään vanhoja autokauppoja *opetusdatana* mallille. Koneoppimismalli sovittaa datan oppimaan auton hinnan X:n funktiona. Esimerkiksi sovitettu funktio voisi näyttää tältä

$$y = wx + w_0,$$

jollekin w ja w_0. Regressio ja luokittelu ovat molemmat *ohjattuja oppimisongelmia*, eli mallissa on syöte X ja tuloste Y ja tehtävänä on oppia kuvaus syötteeltä tulosteelle. Koneoppimisessa malli on yleisesti muotoa

$$y = g(x|\theta),$$

missä $g(\cdot)$ on malli ja θ sen parametrit. Y on regressiossa reaaliluku ja luokittelussa luokan tunnus (esim 1/0). Regression tapauksessa $g(\cdot)$ on regressiosuora ja luokittelussa se on diskriminanttifunktio, joka jakaa datan eri luokkiin. Koneoppimisalgoritmi optimoi parametrin θ siten, että approksimaatiovirhe minimoituu, eli estimaatit ovat mahdollisimman lähellä opetusdatan oikeita arvoja (kuva 2).



Kuva 2: Regressio

2.3 Mallin valinta

Mallin valinta koneoppimisessa aiheuttaa aina vinoumaa, ja tämän vuoksi mallin validointi on tärkeää. Jos malli noudattaa liikaa opetusdataa, se *ylisovittuu* eikä välttämättä ole hyvä uusien havaintojen ennustamiseen. Jos malli on liian yksinkertainen, esimerkiksi jos kolmannen asteen polynomiin yritetään sovittaa suoraa, se *alisovittuu*.

Mallin yleistämisessä valitaan sopivin kompleksisuus hypoteesiluokasta $\mathcal{H}$ vastaamaan opetusdatan selittävää funktiota. Mallin kompleksisuus valitaan niin, että malli ei opi koko opetusdataa, jotta malli pystyisi ennustamaan myös uudet havainnot. Jos $\mathcal{H}$ on vähemmän kompleksi kuin datan selittävä funktio, malli alisovittuu. Jos $\mathcal{H}$ on liian kompleksi voi olla, että ei ole tarpeeksi dataa muodostamaan funktiota, jolloin päädytään huonoon hypoteesiin $h \in \mathcal{H}$. Jos hypoteesi on liian kompleksinen, niin se voi oppia myös datan selittävän funktion lisäksi satunnaisen kohinan.

Mallin valinnassa pitää tasapainottaa kolme tekijää: hypoteesin kompleksisuus, opetusdatan määrä ja *yleistysvirhe* uusille havainnoille. Kun opetusdatan määrä kasvaa, niin yleistysvirhe laskee. Kun malliluokan $\mathcal{H}$ kompleksisuus kasvaa, niin yleistysvirhe laskee aluksi, mutta alkaa myöhemmin kasvaa. Ylikompleksin malliluokan $\mathcal{H}$ yleistysvirhe voidaan hallita isommalla opetusdatalla, mutta vain tiettyyn pisteeseen asti.

Mallin yleistysvirhettä voidaan mitata jakamalla aikaisempi data opetus- ja validointidataan. Tällä siis simuloidaan sitä, että opetusdatalla opetettuun malliin syötetään uusia havaintoja validointidatasta ja katsotaan kuinka hyvin malli ennustaa ne.

2.4 Ennusteen suorituskyvyn mittaaminen

Kahden luokan tapauksessa ennustetut tulokset voidaan jakaa neljään ryhmään: *oikein luokiteltu positiivinen* (TP), *oikein luokiteltu negatiivinen* (TN), *väärin luokiteltu positiivinen* (FP) ja *väärin luokiteltu negatiivinen* (FN). Jos oikea arvo on positiivinen ja ennuste on positiivinen, on ryhmä tällöin oikein luokiteltu positiivinen. Jos oikea arvo on positiivinen ja ennuste on negatiivinen, on ryhmä tällöin väärin luokiteltu negatiivinen. Näistä voidaan johtaa seuraavanlaisia tarkkuuden mittareita.

Määritelmä 1. (Virhe) Olkoon validointidatan havaintojen lukumäärä N. Tällöin ennustuksen virhe on

$$virhe = \frac{fp + fn}{N},$$

missä fp on väärin luokiteltujen havaintojen positiivisten lukumäärä ja fn on väärin luokiteltujen negatiivisten havaintojen lukumäärä.

Määritelmä 2. (Tarkkuus) Olkoon validointidatan havaintojen lukumäärä N. Tällöin ennustuksen tarkkuus on

$$tarkkuus = \frac{tp + tn}{N} = 1 - virhe,$$

missä tp on oikein luokiteltujen positiivisten havaintojen lukumäärä ja tn on oikein luokiteltujen negatiivisten havaintojen lukumäärä.

Jos koneoppimisalgoritmi ennustaa jatkuvaa muuttujaa, niin edellä mainitut määritelmät eivät toimi. Regressiolle yleisin mittari on *keskineliövirhe*.

Määritelmä 3. (Keskineliövirhe) Olkoon $\hat{y}_i$ mallin ennusteet, y_i havaintojen oikeat arvot ja havaintojen lukumäärä on N. Tällöin keskineliövirhe (Mean Squared Error) on

$$MSE(y_i, \hat{y}_i) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{1}$$

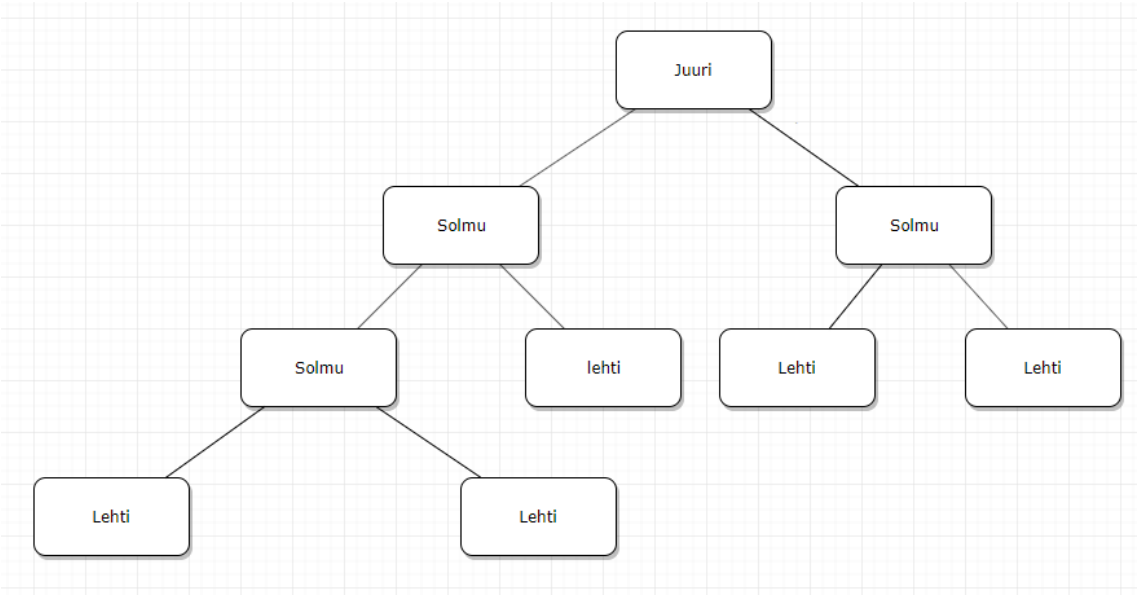
missä $i = 1, ..., N$. Keskineliövirheestä käytetään myös usein neliöjuuriversiota, eli keskihajontaa (Root Mean Squared Error).

$$RMSE(y_i, \hat{y}_i) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2}$$

3 Päästöspuut

Tässä luvussa käydään läpi *päätöspuita* ja niiden rakentamista. Päätöspuu pyrkii selittämään järjestelmän tai ilmiön rakennetta luokittelemalla *datan* puurakenteen avulla. Tässä luvussa käsitellään *ylhäältä alaspäin kasvattavia algoritmeja* (TDIDT). Algoritmeja jotka kuuluvat tähän perheeseen kutsutaan *TDIDT-oppimisperheeksi*. [2].

Puu on *tietorakenne* joka koostuu hierarkkisesti toisiinsa linkitetyistä *solmuista*. ylintä solmua kutsutaan puun *juureksi*, keskimmäisiä solmuiksi ja alimpia *lehdiksi* Kuva (3).



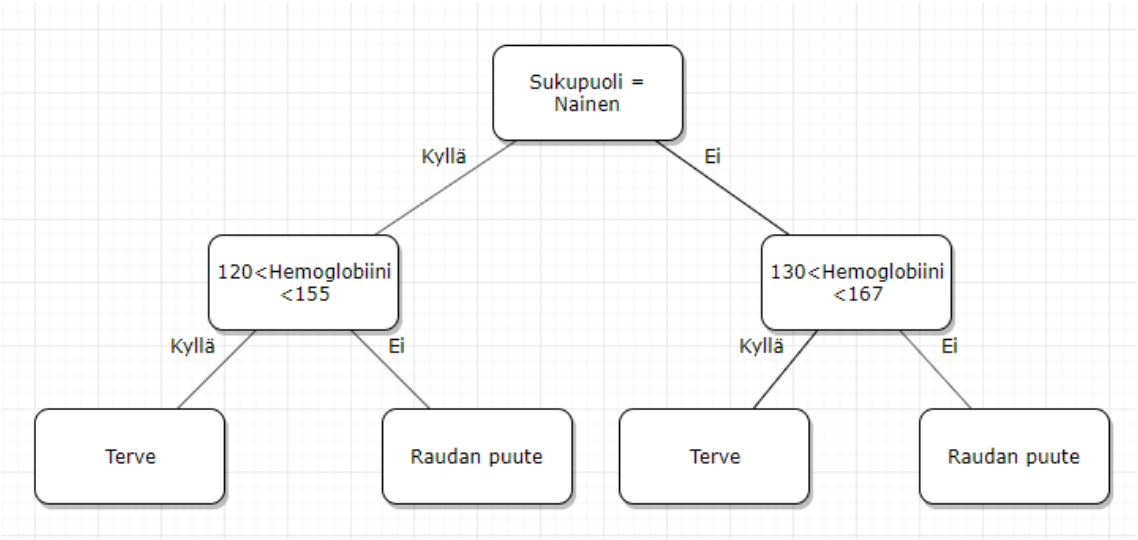
Kuva 3: Puu

Päätöspuussa data luokitellaan puurakenteen läpi siten, että lähdetään juuresta ja sen arvon perusteella laitetaan joko oikeaan tai vasempaan solmuun. Tätä jatketaan niin kauan, että päädytään lehteen. Tämä jako voidaan tehdä joko binäärisesti kyllä-ei valinnalla tai vertaamalla jatkuvaa numeerista arvoa määrättyyn jakopisteeseen tai jakopisteisiin.

Esimerkki 1. Tehdään päätöspuu, joka luokittelee, onko henkilöllä raudan puute vai ei. Käytetään puuta seuraavaan dataan.

Id	Hemoglobiini	Sukupuoli
1	110	Nainen
2	150	Mies

Raudan viitearvo on miehille 130-167 ja naisille 120-155. Jotta puu toimisi oikein, pitää päätös tehdä ensin miehen ja naisen välillä. Tämän jälkeen seuraavat solmut kysyvät onko hemoglobiini viitearvon välillä. Lehdet luokittelevat henkilöt joko terveiksi tai ei terveiksi.



Kuva 4: Päätöspuu

Ajetaan aiemmin määritelty taulukko puun läpi kuva (4). Ensimmäinen rivi menee vasemman haaran Raudan puute lehteen ja toinen rivi menee oikean haaran Terve lehteen. Tässä erimerkissä määriteltiin päätöspuu päättelemällä, mutta jos dataa on paljon tai kriteeri, jonka mukaan päätös tehdään, on epäselvä päättely voi olla vaikeaa tai työlästä. Seuraavissa kappaleissa käydään erilaisia metodeja, joiden avulla voidaan tehdä optimaalisia päätöspuita

3.1 Ginikerroin

Ginikerroin mittaa muuttujan epäpuhtautta. Se kertoo, kuinka paljon *informaatiota* säilytetään, jos tehdään satunnainen päätöspuun jako. Jos kaikki muuttujan arvot kuuluvat samaan luokkaan, on muuttuja tällöin puhdas ja ginikerroin on 0. Jos ginikerroin on 1, niin muuttujan arvot ovat satunnaisesti jakautuneet eri muuttujan luokille [3].

Määritelmä 4. (Ginikerroin)[3] Olkoon havainnot x_i, missä $i = 1, 2, .., n$. Tällöin Ginikerroin on

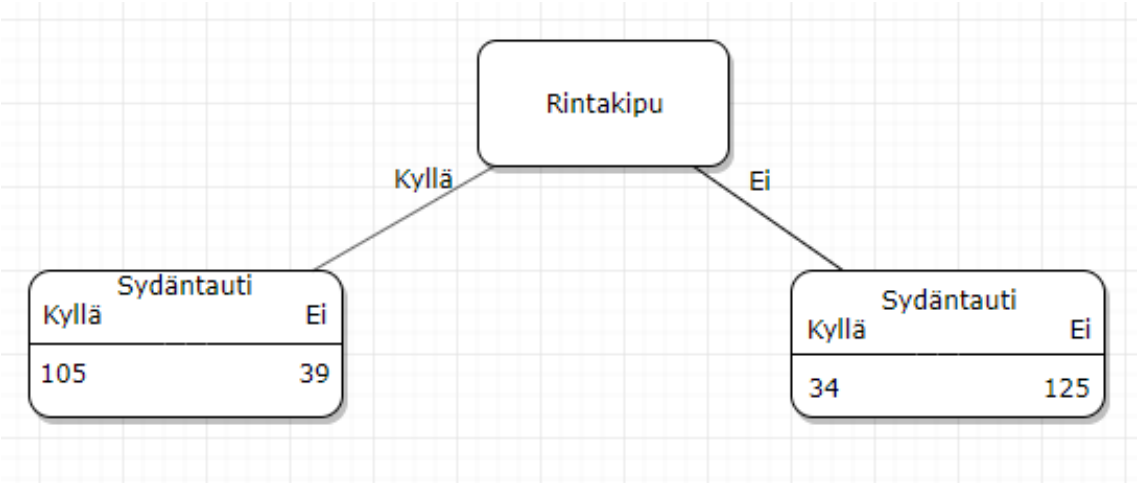
$$Gini(S) = 1 - \sum_i p_i^2, \tag{3}$$

missä p_i on todennäköisyys luokitella muuttujan arvo tiettyyn ryhmään. Päätöspuun rakentamisessa valitaan muuttujaksi siis aina se muuttuja joka jakaa aineiston parhaiten, eli muuttujan jonka ginikerroin on pienin.

Esimerkki 2. Tehdään päätöspuu Ginikertoimen avulla seuraavasta aineistosta.

Rintakipu	Hyvä verenkierto	Verisuonet tukossa	Sydäntauti
Ei	Ei	Ei	Ei
Kyllä	Ei	Kyllä	Kyllä
Ei	Kyllä	Kyllä	Kyllä
Ei	Ei	Kyllä	Kyllä
Kyllä	Ei	Kyllä	Ei
...	...	...	...

Tarkastellaan ensin, miten rintakipu ennustaa sydäntautia. Tehdään tätä varten yhden solmun kokoinen päätöspuu (kuva 5).



Kuva 5: Päätöspuu

Näin ollen rintakivun ginikerroin lasketaan käyttämällä kaavaa (3) seuraavasti vasemmalle lehdelle

$$Gini(\text{Rintakipu} = \text{Kyllä}) = 1 - ((\frac{105}{105+39})^2 + (\frac{39}{39+105})^2) \approx 0,395,$$

ja oikealle lehdelle

$$Gini(\text{Rintakipu} = \text{Ei}) = 1 - ((\frac{34}{34+125})^2 + (\frac{125}{34+125})^2) \approx 0,336.$$

Tämän jälkeen lasketaan vielä edellä laskettujen ginikertoimien painotettu keskiarvo

$$Gini(\text{Rintakipu}) = (\frac{144}{144+159})0,395 + (\frac{159}{159+144})0,336 \approx .0,364$$

Lasketaan hyvän verenkierron ja tukossa olevien verisuonien ginikertoimet.

$$Gini(\text{Hyvä verenkierto}) = 0,360,$$

$$Gini(\text{Verisuonet tukossa}) = 0,381.$$

Valitaan siis puun juureksi hyvä verenkierto, koska sen ginikerroin on pienin. Kun data jaetaan hyvän verenkierron mukaan, vasempaan solmuun päätyy 37/127 ja oikeaan 100/33 (Kyllä/Ei). Tämän jälkeen ensimmäiseksi lasketaan vasemmalle solmulle Ginikerroin rintakivulle ja tukossa oleville verisuonille

$$Gini(\text{Rintakipu}) = 0,3$$

$$Gini(\text{Verisuonet tukossa}) = 0,29.$$

Joten nyt valitaan vasemman solmun päätöskriteeriksi tukossa olevat verisuonet. Tämä jakaa vasempaan solmuun alun perin valikoituneet 37/127, siten että jos verisuonet ovat tukossa vasempaan lehteen päätyy 24/25 ja jos ne eivät ole niin oikeaan lehteen päätyy 13/102. Eli jos on hyvä verenkierto ja verisuonet eivät ole tukossa päätöspuu luokittelee 13 sydäntautia ja 102 ei sydäntautia. Vielä pitää tarkastaa kuinka hyvin rintakipu jakaa tukkeutuneet verisuonet muuttujan. Rintakipu jakaa oikean lehden 13/102 henkilöä rintakivullisiin 7/26 ja rintakivuttomiin 6/76. Selvitetään onko alkuperäisellä jaolla parempi ginikerroin kuin jakamalla rintakivun mukaan.

$$Gini(\text{Verisuonet tukossa} = \text{Ei}) = 1 - (\frac{13}{13+102})^2 - (\frac{102}{13+102})^2 \approx 0,2$$

$$\begin{aligned} Gini(\text{Rintakipu}) = &(1 - (\frac{7}{7+26})^2 - (\frac{26}{7+26})^2)(\frac{33}{33+82}) \\ &+ (1 - (\frac{6}{6+76})^2 - (\frac{76}{6+76})^2)(\frac{82}{82+33}) \approx 0,2 \end{aligned}$$

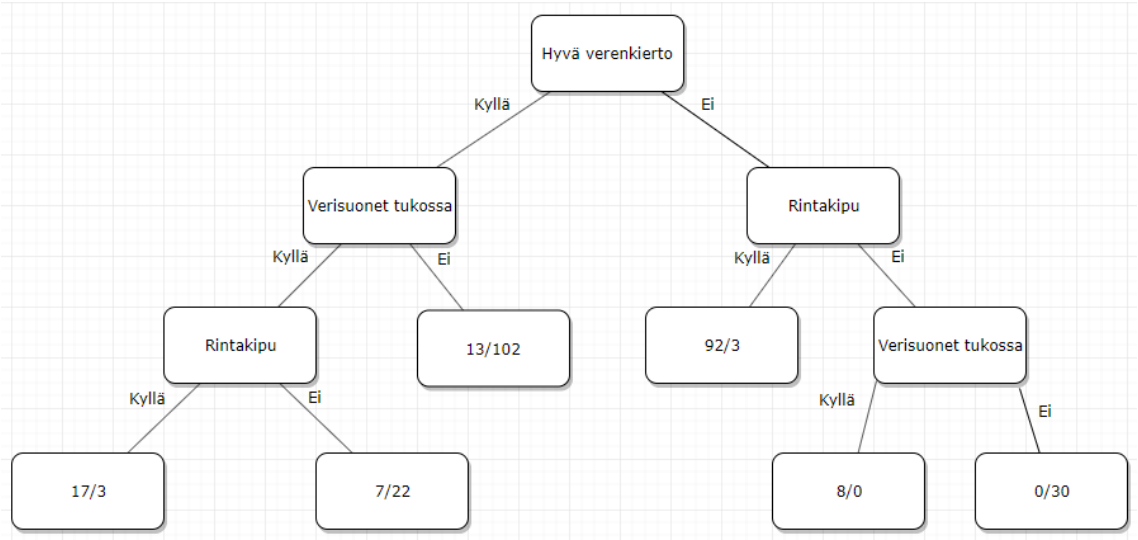
Ginikertoimet molemmille on hyvin lähellä toisiaan, eli solmua ei kannata jakaa rintakivun mukaan. Tehdään vielä sama verisuonet tukossa lehdelle

$$Gini(\text{Verisuonet tukossa} = \text{Kyllä}) = 1 - (\frac{24}{24+25})^2 - (\frac{25}{24+25})^2 \approx 0,5$$

$$\begin{aligned} Gini(\text{Rintakipu}) = &(1 - (\frac{17}{17+3})^2 - (\frac{3}{17+3})^2)(\frac{20}{20+29}) \\ &+ (1 - (\frac{7}{7+22})^2 - (\frac{22}{7+22})^2)(\frac{29}{20+29}) \approx 0,32. \end{aligned}$$

Rintakivun Ginikerroin on selvästi suurempi, joten jaetaan tukossa olevat verisuonet lehti vielä rintakivun mukaan. Nyt puun koko vasen puoli on jaettu, joten siirrytään puun oikealle puolelle.

Huono verenkierto jakaa puun oikealle puolelle 100 sydäntautista ja 33 sydäntauditonta. Rintakipu jakaa nämä vasempaan lehteen 92/3 ja oikeaan lehteen 8/30. Vasen lehti ei enää jakaudu. Tukkiutuneet verisuonet jakaa vielä oikean lehden 8/0 ja 0/30. Valmis puu näyttää tältä (kuva 6).



Kuva 6: Päätöspuu

3.2 Gradienttitehostettu päätöspuu

Gradienttitehostettu päätöspuu yhdistää monta pientä päätöspuuta ennusteen luomiseen. Jokainen päätöspuu lisää malliin tarkkuutta siihen valikoituneiden muuttujien perusteella. Mallin opettaminen tapahtuu iteratiivisesti siten, että jokainen puu pyrkii minimoimaan tämän iteraation regressiovirheen [4].

Määritellään $T(x_i)$ olemaan tämänhetkinen ennuste muuttujalle x_i. Oletetaan, että häviöfunktio $\mathcal{L}(T(x_i), ..., T(x_n))$ on on konveksi, differentioituva ja se saavuttaa miniminsä jos $T(x_i) = y_i$ kaikille x_i. Jokaisella iteraatiolla lisätään uusi puu $h(\cdot)$ aikaisempaan luokittelijaan $T(\cdot)$. Paras $h(\cdot)$ löydetään *ensimmäisen asteen Taylorin sarjakehitelmällä* $\mathcal{L}(T + h_t)$, joka minimoidaan funktion h_t suhteen. Gradienttitehostamisessa käytetään *gradienttimenetelmää* muuttujille $x_1, ..., x_i$ ja jokaisella iteraatiolla tämänhetkinen luokittelija $T(x_i)$ päivitetään *gradienttiaskeleella*

$$T(x_i) \leftarrow T(x_i) - \alpha \frac{\mathcal{L}}{T(x_i)},$$

missä $\alpha > 0$ on mallin oppimisvauhti. Negatiivinen gradientti $-\frac{\partial \mathcal{L}}{\partial T(x_i)}$ approksimoidaan regressiopuun ennusteella $h_t(x_i)$, joka toteuttaa seuraavat ehdot

$$h_t \approx -\min_{h \in \mathcal{T}_d} \sum_{i=1}^{n} (h_t(x_i) - r_i)^2, \text{ missä } r_i = \frac{\partial \mathcal{L}}{\partial T(x_i)}.$$

Tässä siis häviöfunktio $\mathcal{L}$ on neliövirhe ja gradientti muuttujalle x_i on aikaisemman iteraation residuaali $r_i = y_i - T(x_i)$. $\mathcal{T}_d$ on lehti mihin havainnot kuuluvat.

Esimerkki 3. Tehdään seuraavalle datalle yllä mainittu algoritmi.

Pituus	Paino	Sukupuoli	Punnerruksia
1,7	60	Nainen	20
1,65	70	Mies	7
1,5	55	Mies	16

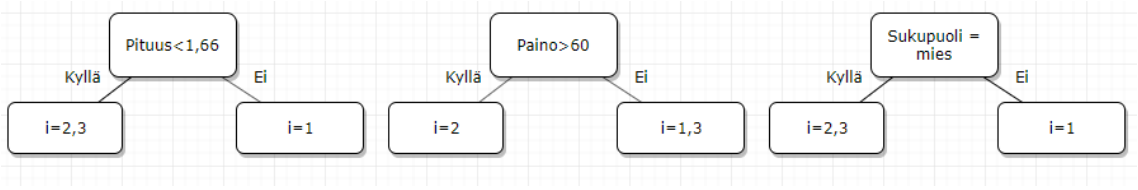
Käytetään häviöfunktiona $\mathcal{L} = \frac{1}{2}(y_i - \hat{y})^2$, eli neliövirhettä. Ensimmäinen ennuste lasketaan koko datasta.

$$\begin{aligned} T(x) &= \min_{\hat{y}} \frac{1}{2} \sum_{i=1}^{3} (y_i - \hat{y})^2 \\ &= \frac{\partial}{2\partial \hat{y}} \sum (y_i - \hat{y})^2 = 0 \\ &= -(20 - \hat{y}) - (7 - \hat{y}) - (16 - \hat{y}) = 0 \\ \hat{y} &= 14,3 \end{aligned}$$

Ensimmäinen ennuste on siis punnerrusten keskiarvo. Seuraavaksi lasketaan r, joka on aikaisemman iteraation residuaalit

$$r = y - T(x) = (5,7; -7,3; 1,7).$$

Luodaan ennalta määrätty määrä satunnaisia päätöspuita (kuva 7).



Kuva 7: Satunnaiset päätöspuut

Ensimmäisessä puussa havainnot kaksi ja kolme menevät vasempaan lehteen ja ensimmäinen havainto oikeaan. Nyt lasketaan molemmille lehdille h_t

$$\begin{aligned} h_t &\approx -\min_{h \in \mathcal{T}_d} \frac{1}{2} \sum_{i=1}^{3} (h_t(x) - r_i)^2 \\ &= \frac{\partial}{2\partial h_t} \sum_{i=1}^{3} (h_t(x) - r_i)^2 = 0 \\ &= \sum_{i=1}^{3} (h_t(x) - r_i) = 0 \\ &= \begin{cases} (h_1(x_i) - (-7,3)) + (h_1(x_i) - 1,7) = 2h_1(x_i) + 5,6, \text{ kun } i = 2,3 \\ (h_1(x_i) - 5,7), \text{ kun } i = 1 \end{cases} \\ &= \begin{cases} h_1(x_i) = -2,8 \\ h_1(x_i) = 5,7 \end{cases} \end{aligned}$$

Tästä saadaan ennuste kun oppimisvauhti $\alpha = 0,3$

$$
\begin{aligned}
T(x_1) &= 14,3 + 0,3 \cdot 5,7 = 16,01, \\
T(x_2) &= 14,3 + 0,3 \cdot (-2,8) = 13,46, \\
T(x_3) &= 14,3 + 0,3 \cdot (-2,8) = 13,46.
\end{aligned}
$$

Lasketaan taas ennusteen ja havaintojen väliset residuaalit r

$$r = y - T(x) = (3,99; -6,46; 2,54).$$

Lasketaan h_t seuraavalle puulle, missä vasemmassa lehdessä on yksi havainto ja oikeassa kaksi.

$$
\begin{aligned}
h_t &\approx -\min_{h \in \mathcal{T}_d} \frac{1}{2} \sum_{i=1}^{3} (h_t(x) - r_i)^2 \\
&= \frac{\partial}{2\partial h_t} \sum_{i=1}^{3} (h_t(x) - r_i)^2 = 0 \\
&= \sum_{i=1}^{3} (h_t(x) - r_i) = 0 \\
&= \begin{cases} (h_2(x_i) - 3,99) + (h_2(x_i) - 2,54) = 2h_2(x_i) - 6,53, \text{ kun } i = 1,3 \\ (h_2(x_i) - (-6,46)), \text{ kun } i = 2 \end{cases} \\
&= \begin{cases} h_2(x_i) = 3,28 \\ h_2(x_i) = -6,46. \end{cases}
\end{aligned}
$$

Ja seuraava ennuste on

$$
\begin{aligned}
T(x_1) &= 16,01 + 0,3 \cdot 3,28 = 16,994, \\
T(x_2) &= 13,46 + 0,3 \cdot (-6,46) = 11,522, \\
T(x_3) &= 13,46 + 0,3 \cdot 3,28 = 14,444.
\end{aligned}
$$

Lasketaan vielä viimeiselle puulle r

$$r = y - T(x) = (3,006; -4,522; 1,556).$$

Seuraavassa puussa ensimmäisessä lehdessä on kaksi havaintoa ja oikeassa yksi, joten

h_t on

$$
\begin{aligned}
h_t &\approx -\min_{h \in \mathcal{T}_d} \frac{1}{2} \sum_{i=1}^{3} (h_t(x) - r_i)^2 \\
&= \frac{\partial}{2 \partial h_t} \sum_{i=1}^{3} (h_t(x) - r_i)^2 = 0 \\
&= \sum_{i=1}^{3} (h_t(x) - r_i) = 0 \\
&= \begin{cases} (h_3(x_i) - (-4,522)) + (h_2(x_i) - 1,556) = 2h_3(x_i) + 2,966, \text{ kun } i = 2,3 \\ (h_3(x_i) - 3,006), \text{ kun } i = 1 \end{cases} \\
&= \begin{cases} h_3(x_i) = -1,438 \\ h_3(x_i) = 3,006. \end{cases}
\end{aligned}
$$

Ja viimeinen ennuste on

$$
\begin{aligned}
T(x_1) &= 16,994 + 0,3 \cdot 3,006 \approx 17,90, \\
T(x_2) &= 11,552 + 0,3 \cdot -1,438 \approx 11,12, \\
T(x_3) &= 14,444 + 0,3 \cdot -1,438 \approx 14,01.
\end{aligned}
$$

Lasketaan vielä jokaiselle ennuste kierrokselle keskineliövirhe kaavalla (1). Alkuperäisten punnerrusten lukumäärä $y = (20, 7, 16)$ ja ensimmäinen ennuste $T(\cdot) = (16,01; 13,46; 13,46)$, joten keskineliövirhe on

$$
\begin{aligned}
MSE(y,T) &= \frac{1}{N} \sum_{i=1}^{N} (y_i - T(x_i))^2 \\
&= \frac{1}{3}((20 - 16,01)^2 + (7 - 13,46)^2 + (16 - 13,46)^2) \\
&= \frac{1}{3}(15,9201 + 41,7316 + 6,4516) \approx 21,37.
\end{aligned}
$$

Seuraava ennuste on $T(\cdot) \approx (16,99; 11,52; 14,44)$, joten keskineliövirhe on

$$
\begin{aligned}
MSE(y,T) &= \frac{1}{N} \sum_{i=1}^{N} (y_i - T(x_i))^2 \\
&= \frac{1}{3}((20 - 16,99)^2 + (7 - 11,52)^2 + (16 - 14,44)^2) \\
&= \frac{1}{3}(9,0601 + 20,4304 + 2,4336) \approx 10,64.
\end{aligned}
$$

Viimeinen ennuste on $T(\cdot) \approx (17,90; 11,12; 14,01)$, joten keskineliövirhe on

$$
\begin{aligned}
MSE(y,T) &= \frac{1}{N} \sum_{i=1}^{N} (y_i - T(x_i))^2 \\
&= \frac{1}{3}((20 - 17,90)^2 + (7 - 11,12)^2 + (16 - 14,01)^2) \\
&= \frac{1}{3}(4,41 + 16,9744 + 3,9601) \approx 8,45.
\end{aligned}
$$

Huomataan siis, että jokainen ennuste parantaa keskineliövirhettä, vaikka alussa määritellyt puut olivat täysin satunnaisia. Seuraavassa kappaleessa tutustutaan edistyneenpään gradienttitehostamisalgoritmiin, joka pyrkii ratkaisemaan optimaalisen päätöspuun jokaiselle ennusteelle.

4 XGBoost

Puun tehostaminen on erittäin tehokasta ja laajasti käytettyä *koneoppimisessa*. Tämä kappale käsittelee erityisesti *XGBoost* -algoritmia joka perustuu artikkeliin [5]. *XGBoost* -algoritmi yhdistää monta päätöspuuta, joita kutsutaan *heikoiksi oppijoiksi*. Nämä heikot oppijat parantavat yksitellen ennusteen arvoa. Seuraavassa kappaleessa käydään läpi ennustavaa kohdefunktiota.

4.1 Kohdefunktio

Tässä kappaleessa esitellään XGBoost -algoritmin kohdefunktio.

Määritelmä 5. (Kohdefunktio)[5] Olkoon data missä on n kpl havaintoja ja m kpl muuttujia ja $\mathcal{D} = (x_i, y_i)$, missä $|\mathcal{D}| = n, x_i \in \mathbb{R}^m$ ja $y_i \in \mathbb{R}$. Puun kokonaismalli käyttää K:n funktion summaa mallin ennusteen luomiseksi

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i),\ f_k \in \mathcal{F}, \tag{4}$$

missä $\mathcal{F} = \{f(x) = \omega_{q(x)}\}(q : \mathbb{R}^m \to T, \omega \in \mathbb{R}^T)$ on *regressiopuuavaruus* (CART). Funktio q kuvaa jokaisen puun rakenteen, joka yhdistää havainnon sitä vastaavaan lehteen. T on puun lehtien lukumäärä. Jokainen f_k vastaa riippumatonta puurakennetta q ja lehden painoa ω. Toisin kuin päätöspuissa jokaisen regressiopuun lehdillä on oma jatkuva pisteytys. Merkitään ω_i esittämään i:nnen lehden pistettä. Jokainen havaintorivi ajetaan puiden läpi lehtiin q:n määräämien jakosääntöjen mukaan. Summaamalla nämä lehtien pisteet päädytään lopulliseen luokitukseen.

Esimerkki 4. Olkoon regressiopuuavaruus $\mathcal{F}$ (kuva 8). Ennustetaan, kuinka monta punnerrusta henkilö pystyy tekemään seuraavan datan perusteella

Ikä	Sukupuoli	Urheilee?
60	Mies	Kyllä
15	Nainen	Kyllä
40	Nainen	Kyllä
90	Mies	Kyllä
50	Nainen	Ei

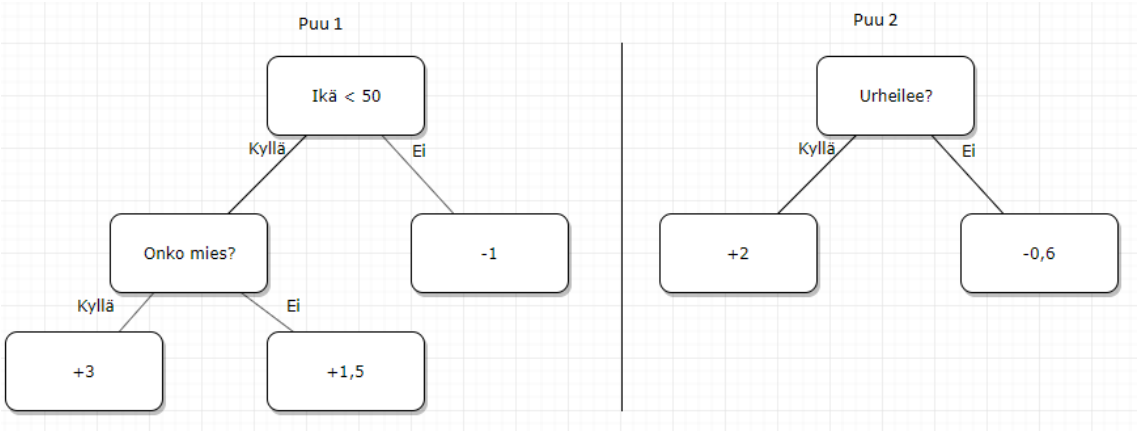
Nyt $f_k(x_i)$ tarkoittaa havainnon i puussa k olevaa arvoa. Eli ensimmäinen havainto lasketaan seuraavasti

$$\hat{y}_1 = \sum_{k=1}^{2} f_k(x_1) = -1 + 2 = 1,$$

eli ensimmäinen henkilö saa tehtyä yhden punnerruksen. Samalla lailla saadaan lopuille havainnoille seuraavat ennusteet $\hat{y} = (1, 3.5, 3.5, 1, 0.9)$.

Saadaksemme malliin tarvittavan joukon funktioita minimoidaan ensin seuraava funktio

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \tag{5}$$



Kuva 8: Regressiopuuavaruus

$$\text{missä } \Omega(f) = \gamma T + \frac{1}{2}\lambda||\omega||^2.$$

l on differentioituva konveksi häviöfunktio, joka mittaa ennusteen $\hat{y}_i$ ja tavoitteen y_i eroa ja Ω sakottaa mallin monimutkaisuutta. Toinen säännöllistämistermi tekee lopullisista painoista sileitä, jotta malli ei ylisovittuisi.

4.2 Gradienttipuun tehostaminen

Kaavan (4) puurakennemalli sisältää funktioita muuttujina, joten sitä ei voida optimoida tavallisilla optimointimetodeilla euklidisessa avaruudessa. Tämän sijasta malli opetetaan additiivisella tavalla. Olkoon $\hat{y}_i^{(t)}$ i:s ennuste t:nnellä iteraatiolla, tähän täytyy lisätä f_t jotta voidaan minimoida seuraava funktio

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}^{(t-1)} + f_t(x_i)) + \Omega(f_t). \tag{6}$$

Tämä tarkoittaa, että malliin lisätään f_t joka parantaa mallia eniten kaavan (5) mukaan.

Kaava (6) voidaan optimoida *Taylorin toisen kertaluvun approksimaatiolla* avulla yleisesti differentioituville häviöfunktioille. Funktiolle $f(x)$ yksinkertaisin lineaarinen approksimaatio on

$$f(x) \simeq f(a) + f'(a)(x-a) = f(a) + f'(a)\Delta x.$$

Sovelletaan tätä kaavaan (6) siten, että $f(x)$ on häviöfunktio l ja a on aikaisemman askeleen $(t-1)$ ennustettu arvo ja Δx on uusi oppija, joka pitää lisätä askeleella t. Näin ollen, kun valitaan toisen asteen Taylorin approksimaatio tällöin funktion $f(x)$ approksimaatio on

$$f(x) \simeq f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2.$$

Tästä saadaan johdettua Taylorin toisen asteen approksimaatio funktiolle $\mathcal{L}$

$$\mathcal{L}^t \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t),$$

missä g_i ja h_i määritellään seuraavasti

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

$$h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}).$$

Nämä ovat häviöfunktion ensimmäisen ja toisen kertaluvun gradienttistatistiikat. Poistetaan vakiotermit, jotta saadaan seuraava yksinkertaistettu funktio

$$\tilde{\mathcal{L}}^t = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t). \tag{7}$$

Tämä on siis uuden puun optimointitavoite. Tämän määrittelyn etu on se, että funktio on riippuvainen vain muuttujista g_i ja h_i. Tämän ansiosta XGBoost -algoritmi tukee kaikkia differentioituvia häviöfunktioita.

Esimerkki 5. Ratkaistaan ensimmäisen ja toisen kertaluvun gradienttistatistiikat binääriselle logaritmihäviöfunktiolle

$$f(x) = yln(p) + (1-y)ln(1-p), \text{ missä } p = \frac{1}{(1+e^{-x})}.$$

Tässä y saa arvoja $\{0, 1\}$ ja p on todennäköisyyspisteytys. Ratkaistaan siis funktion ensimmäinen ja toinen derivaatta.

$$\frac{\partial f}{\partial p} = \frac{y}{p} - \frac{1-y}{1-p} = \frac{y(1-p) - p(1-y)}{p(1-p)} = \frac{y-p}{(1-p)p},$$

$$\frac{\partial p}{\partial x} = p(1-p).$$

Joten ensimmäinen derivaatta on

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial p} \frac{\partial p}{\partial x} = y - p,$$

ja toinen derivaatta

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}(\frac{\partial f}{\partial x}) = \frac{\partial}{\partial x}(y-p) = \frac{\partial p}{\partial x} = p(1-p).$$

Määritellään lehden j jäsenten ryhmä $I_j = \{i | q(x_i) = j\}$. Avataan yhtälö (7) avaamalla Ω

$$\tilde{\mathcal{L}}^t = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} \omega_j^2$$

$$= \sum_{i=1}^{n} [(\sum_{i \in I_j} g_i)\omega_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda)\omega_j^2] + \lambda T.$$

Jos puun rakenne $q(x)$ on kiinteä, voidaan ratkaista optimaalinen j:nnen lehden paino ω_j^*

$$\omega_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{8}$$

ja tätä vastaava optimiarvo

$$\tilde{\mathcal{L}}^t(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{9}$$

Kaavaa (9) voidaan käyttää pisteytysfunktiona puurakenteen q laadun mittaamiseksi. Tämä pisteytysfunktio on niin kuin ginikerroin (3), joka mittaa päätöspuun epäpuhtautta, mutta se on johdettu eri kohdefunktiosta.

Yleensä on mahdotonta määrittää kaikki mahdolliset puurakenteet q. Käytetään *ahnetta algoritmia*, joka aloittaa yhdestä lehdestä ja iteratiivisesti lisää uusia lehtiä puuhun. Määritellään I_L ja I_R vasemman ja oikean lehden jäseniksi. Eli $I = I_L \cup I_R$ ja tällöin häviöfunktio voidaan kirjoittaa seuraavasti

$$\mathcal{L}_{jako} = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma. \tag{10}$$

Tätä kaavaa (10) käytetään yleensä puukokelaitten solmujen jakojen arviointiin. Kaavassa siis on vasemmalta oikealle uuden oikean lehden pisteytys, Oikean uuden lehden pisteytys ja alkuperäisen lehden pisteytys. Jos hyöty on pienempi kuin γ tätä jakoa ei kannata tehdä.

Esimerkki 6. Lasketaan XGBoost -algoritmilla ennustukset henkilöiden painosta seuraavalle yksinkertaiselle datalle.

Pituus	Lempi väri	Sukupuoli	Paino
1,7	Sininen	Nainen	89
1,65	Vihreä	Mies	75
1,5	Sininen	Mies	63

Käytetään datasta seuraavaa merkintää $\{(x_i, y_i)\}_{i=1}^3$, eli y_i on rivin i paino muuttujan arvo ja x_i on vektori, missä on rivin i pituus, lempi väri ja sukupuoli arvot. Valitaan häviöfunktioksi l neliövirhe, eli

$$l(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2.$$

Opetetaan malli additiivisella tavalla, eli ratkaistaan ensin $\mathcal{L}(\phi)$. Helpottaaksemme

erimerkkiä päätetään, että $\Omega(f_k) = 0$

$$\mathcal{L}(\phi) = \min_{\hat{y}} \sum_i l(y_i, \hat{y}^0) = \frac{d}{d\hat{y}} \frac{1}{2}(89 - \hat{y}^0)^2 + \frac{1}{2}(75 - \hat{y}^0)^2 + \frac{1}{2}(63 - \hat{y}^0)^2 = 0$$
$$3\hat{y}^0 = 227$$
$$\hat{y}^0 = 75,7.$$

Eli ensimmäinen paras ennuste on siis havaintojen keskiarvo ja se on ensimmäisessä puussa, joka on vain yhden lehden kokoinen eli se ennustaa kaikille havainnoille keskiarvon. Ratkaistaan seuraavaksi vektorit **g** ja **h**. Ratkaistaan esimerkiksi g_1

$$g_1 = \frac{\partial}{\partial \hat{y}^{(0)}} l(y_1, \hat{y}^{(0)}) = -(89 - \hat{y}^0) = 75,7 - 89 = -13,3,$$

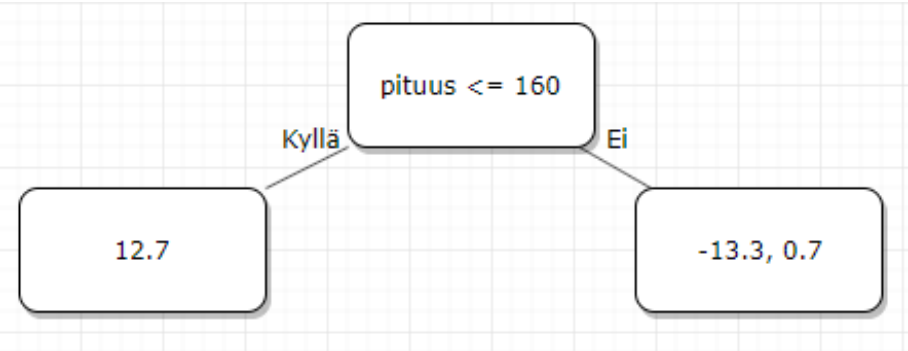
joten tällöin vektori $\mathbf{g} = (-13.3, 0.7, 12, 7)$, on alkuperäisten arvojen ja keskiarvon residuaalit. Ratkaistaan vielä samalla lailla h_1

$$h_1 = \frac{\partial^2}{\partial \hat{y}^{(0)2}} l(y_1, \hat{y}^{(0)}) = 1,$$

joten tällöin vektori **h** on $(1, 1, 1)$. Eli taulukko näyttää nyt tältä

Pituus	Lempi väri	Sukupuoli	Paino	$\hat{y}^0$	g	h
1,7	Sininen	Nainen	89	75,7	-13,3	1
1,65	Vihreä	Mies	75	75,7	0,7	1
1,5	Sininen	Mies	63	75,7	12,7	1

Luodaan uusi optimaalinen päätöspuu vektoreiden g ja h perusteella (kuva 9).



Kuva 9: päätöspuu

Määritellään lehtien jäsenten ryhmät siten, että vasen lehti on I_1 ja se sisältää yhden jäsenen ja oikea lehti I_2 sisältää kaksi jäsentä. Seuraavaksi ratkaistaan funktio

f

$$\begin{aligned}
\mathcal{L}^1 &= \min_f \sum_{i \in I_j} l(y_i, \hat{y}^0 + f_1(x_i)) \\
&= \begin{cases} \frac{\partial}{\partial f} \frac{1}{2}(y_3 - (\hat{y}^0 + f_1(x_i))^2 = 0, \text{ kun } i \in I_1 \\ \frac{\partial}{\partial f} \frac{1}{2}((y_1 - (\hat{y}^0 + f_1(x_i))^2 + (y_2 - (\hat{y}^0 + f_1(x_i))^2) = 0, \text{ kun } i \in I_2 \end{cases} \\
&= \begin{cases} -((63 - 75,7) - f_1(x_i)) = 12,7 + f_1(x_i) = 0 \\ -((89 - 75,7) - f_1(x_i)) - ((75 - 75,7) - f_1(x_i)) = -13,3 + 0,7 + 2f_1(x_i) \end{cases} \\
&= \begin{cases} f_1(x_i) = -12,7 \\ f_1(x_i) = 6,3. \end{cases}
\end{aligned}$$

Joten nyt ensimmäisen askeleen jälkeen painon ennuste näyttää seuraavalta

$$\begin{aligned}
\hat{y}_1 &= \sum_{k=1}^{n} f_k(x_1) = y^0 + f_1(x_1) = 75,7 + 6,3 = 82 \\
\hat{y}_2 &= 75,7 + 7 = 82 \\
\hat{y}_3 &= 75,7 - 12,7 = 63
\end{aligned}$$

Tässä oli ensimmäinen kierros XGBoost -algoritmia. Kuten tuloksesta huomataan, $\hat{y}_3$ sai saman arvon ennusteeksi kuin alkuperäinen arvo oli. Voidaan siis sanoa, että malli ylisovittuu ja tämän takia mallissa on mukana $\Omega(f_k)$. Jos $\lambda = 8$ saadaan lehdille optimaaliset painot kaavalla (8)

$$\omega_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} = \begin{cases} -\frac{12,7}{1+8} = -1,41 \text{ , kun } i \in I_1 \\ -\frac{-13,3 + 0,7}{(1+8)+(1+8)} = 0,62 \text{ , kun } i \in I_2. \end{cases}$$

Seuraavaksi minimoidaan $\mathcal{L}^1$

$$\begin{aligned}
\mathcal{L}^1 &= \min_f \sum_{i \in I_j} l(y_i, \hat{y}^0 + f_1(x_i)) + \frac{1}{2}\lambda\omega_i^2 \\
&= \begin{cases} \frac{\partial}{\partial f} \frac{1}{2}(y_3 - (\hat{y}^0 + f_1(x_i))^2 + 7,95 = 0, \text{ kun } i \in I_1 \\ \frac{\partial}{\partial f} \frac{1}{2}((y_1 - (\hat{y}^0 + f_1(x_i))^2 + (y_2 - (\hat{y}^0 + f_1(x_i))^2) + 1,54 = 0, \text{ kun } i \in I_2 \end{cases} \\
&= \begin{cases} -((63 - 75,7) - f_1(x_i)) + 7,95 = 12,7 + 7,95 + f_1(x_i) = 0 \\ -((89 - 75,7) - f_1(x_i)) - ((75 - 75,7) - f_1(x_i)) + 1,54 = -11,06 + 2f_1(x_i) \end{cases} \\
&= \begin{cases} f_1(x_i) = -20,65 \\ f_1(x_i) = 5,53. \end{cases}
\end{aligned}$$

Eli kun $\lambda > 1$ otetaan pienempiä askelia oikean ennusteen suuntaan. Nyt ensimmäi-

sen askeleen jälkeinen painon ennuste on

$$\hat{y}_1 = \sum_{k=1}^{n} f_k(x_1) = y^0 + f_1(x_1) = 75,7 + 5,53 = 81,23$$
$$\hat{y}_2 = 75,7 + 5,53 = 81,23$$
$$\hat{y}_3 = 75,7 - 20,65 = 55,05$$

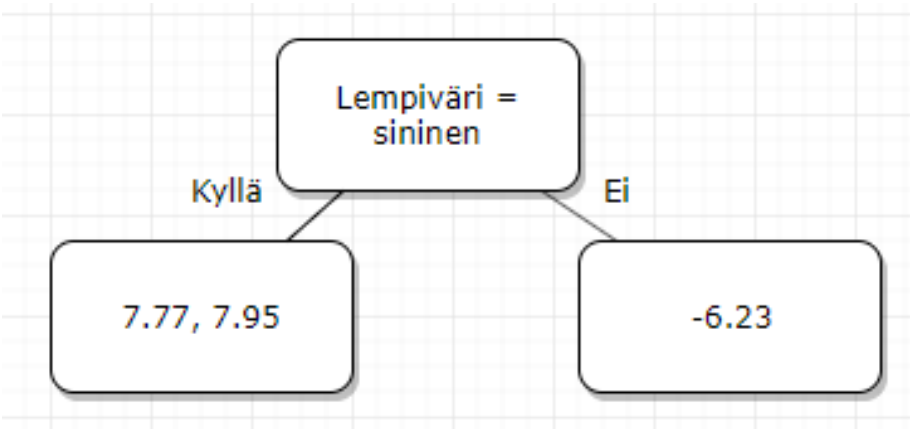
Merkitään seuraavaa algoritmin kierrosta varten $\hat{y}^{(1)} = \hat{y}$, eli seuraavan kierroksen $\hat{y}$ on edellisen kierroksen ennuste. Suoritetaan vielä seuraava kierros algoritmia. Lasketaan ensin vektorin **g** ja **h**

$$\begin{aligned}\mathbf{g} = \frac{\partial}{\partial \hat{y}^{(1)}} l(y_1, \hat{y}^{(1)}) &= ((89 - \hat{y}^{(1)}), (75 - \hat{y}^{(1)}), (55 - \hat{y}^{(1)})) \\ &= ((89 - 81,23), (75 - 81,23), (63 - 55,05)) = (7.77, -6.23, 7.95)\end{aligned}$$

joten tällöin vektori **g** on alkuperäisen arvon ja viimeisen ennusteen residuaalit. Ratkaistaan vielä samalla lailla vektori **h**

$$\mathbf{h} = \frac{\partial^2}{\partial \hat{y}^{(1)2}} l(y_1, \hat{y}^{(1)}) = (1, 1, 1).$$

Seuraavaksi tehdään vektoreiden **g** ja **h** avulla optimaalinen päätöspuu (kuva 10).



Kuva 10: Päätöspuu

Nyt lehti I_1 sisältää kaksi jäsentä ja lehti I_2 yhden jäsenen. Ratkaistaan painot ω

$$\omega_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} = \begin{cases} -\dfrac{7,77 + 7,95}{9 + 9} = -0,83 \text{ , kun } i \in I_1 \\ -\dfrac{-6,23}{9} = 0,69 \text{ , kun } i \in I_2. \end{cases}$$

Seuraavaksi minimoidaan $\mathcal{L}^2$

$$\mathcal{L}^2 = \min_f \sum_{i \in I_j} l(y_i, \hat{y}^{(1)} + f_2(x_i)) + \frac{1}{2}\lambda\omega_i^1$$

$$= \begin{cases} \frac{\partial}{\partial f}\frac{1}{2}((y_1 - (\hat{y}^{(1)} + f_2(x_i))^2 + (y_3 - (\hat{y}^{(1)} + f_2(x_i))^2) + 2,73 = 0, \text{ kun } i \in I_1 \\ \frac{\partial}{\partial f}\frac{1}{2}(y_2 - (\hat{y}^{(1)} + f_2(x_i))^2 + 1,9 = 0, \text{ kun } i \in I_2 \end{cases}$$

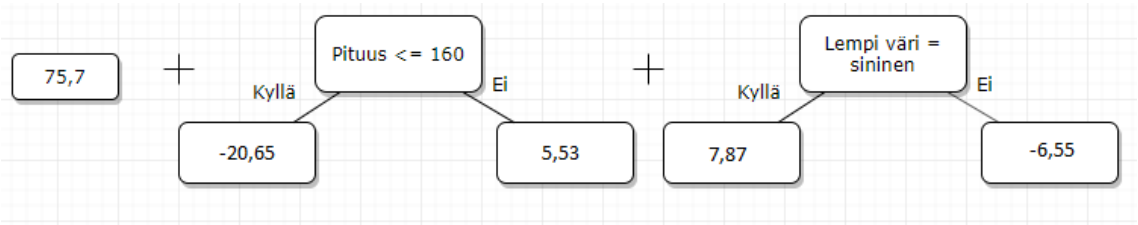
$$= \begin{cases} -((89 - 81,23) - f_1(x_i)) - ((63 - 55,05) - f_1(x_i)) + 2,73 = & -13,09 \\ & +2f_1(x_i) \\ -((75 - 81,23) - f_1(x_i)) + 1,64 = 6,23 + 1,64 + f_1(x_i) = 0 \end{cases}$$

$$= \begin{cases} f_1(x_i) = 7,87 \\ f_1(x_i) = -6,55. \end{cases}$$

Nyt uudet ennusteet ovat

$$\hat{y}_1 = \sum_{k=1}^{n} f_k(x_1) = y^0 + y^1 + f_2(x_2) = 75,7 + 5,53 + 7,87 = 89,1$$
$$\hat{y}_2 = 75,7 + 5,53 - 6,55 = 74,68$$
$$\hat{y}_3 = 75,7 - 20,65 + 7,87 = 62,92.$$

Tästä huomataa, että uusien painon ennusteiden ja oikeiden painojen residuaalit pienenee, eli ennusteet lähestyvät oikeata arvoa. Päätetään, että tämä kierros oli viimeinen mallin opettamiseen, joten mallin puurakenne näyttää tältä (kuva 11).



Kuva 11: Mallin puurakenne

Mallin opettamisen jälkeen saadaan seuraavat uudet havainnot, joille halutaan saada ennuste.

Pituus	Lempi väri	Sukupuoli
1,75	Vihreä	Nainen
1,55	Vihreä	Mies

Malli ennustaa nämä havainnot seuraavasti

$$\hat{y}_1 = \sum_{k=1}^{n} f_k(x_1) = y^0 + y^1 + f_2(x_2) = 75,7 + 5,53 - 6,55 = 74,68$$
$$\hat{y}_2 = 75,7 + -20,65 - 6,55 = 48,5.$$

Esimerkin (6) mallissa oli vain kaksi kahden lehden puuta, mutta yleensä XGBoost -algoritmi tekee 100 puuta joissa on keskimäärin 10 lehteä. Esimerkissä ei myöskään esitetty, miten algoritmi valitsee optimaaliset puut vektoreiden **g** ja **h** avulla. Optimaalisten puiden muodostamista käsitellään seuraavassa kappaleessa.

Esimerkki 7. Lasketaan esimerkin (6) mallille kappaleessa 1 käsiteltyjä mittareita. Ensimmäiseksi lasketaan opetusdatalle opetusvirhe. Havainnot ja ennusteet ovat

Pituus	Lempi väri	Sukupuoli	paino	$\hat{y}$
1,7	Sininen	Nainen	89	89,1
1,65	Vihreä	Mies	75	74,68
1,5	Sininen	Mies	63	62,92

Käytetään opetusvirheen mittarina keskineliövirhettä kaava (1)

$$\begin{aligned} MSE(y, \hat{y}) &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \\ &= \frac{1}{3}((89 - 89,1)^2 + (75 - 74,68)^2 + (63 - 62,92)^2) \\ &= \frac{1}{3}(0,01 + 0,1024 + 0,0064) = 0,04 \end{aligned}$$

Aikaisemmasta datasta jätettiin seuraavat kolme riviä pois validointia varten.

Pituus	Lempi väri	Sukupuoli	paino	$\hat{y}$
1,75	Vihreä	Nainen	76	74,68
1,55	Vihreä	Nainen	50	48,5
1,71	Sininen	Mies	81	89,1

Lasketaan tästä testivirhe kaavalla (1)

$$\begin{aligned} MSE(y, \hat{y}) &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \\ &= \frac{1}{3}((76 - 74,68)^2 + (50 - 48,5)^2 + (81 - 89,1)^2) \\ &= \frac{1}{3}(1,7424 + 2,25 + 65,61) = 23,2 \end{aligned}$$

4.3 Jakoalgoritmi

Tässä kappaleessa käydään läpi XGBoost -algoritmin käyttämät kaksi puunjakoalgoritmia. Ahne algoritmi laskee optimaalisen jaon jokaiselle puulle. Approksimaatioalgoritmi ehdottaa jonkinlaisen jaon ja tekee optimaalisen puun tämän mukaan.

Algorithm 1: Ahne algoritmi puun jakamiseen

Input: I, tämän hetkisen solmun jäsenet
Input: d, muuttujen dimensio

hyöty $\leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i, H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ *to* m **do**
  $G_L \leftarrow 0, H_L \leftarrow 0$
  for j *in sorted*$(I, by\ \boldsymbol{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j, H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L, H_R \leftarrow H - H_R$
    pisteet $\leftarrow max(pisteet, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Jako parhailla pisteillä

Ahneen algoritmin suorittaminen on mahdotonta isolla datalla, koska se laskee kaikki jakopisteet ahneesti ja ottaa niistä parhaimman. Esimerkiksi jatkuvalla muuttujalla ahne algoritmi harkitsee jokaista mahdollista jakopistettä.

Esimerkki 8. Esimerkissä (6) optimaaliset puut vain oletettiin, mutta oikeasti algoritmi ratkaisee puun jokaisella askeleella. Käytetään samaa dataa puun ratkaisemiseen.

Pituus	Lempi väri	Sukupuoli	Paino	$\hat{y}^0$	g	h
1,7	Sininen	Nainen	89	75,7	-13,3	1
1,65	Vihreä	Mies	75	75,7	0,7	1
1,5	Sininen	Mies	63	75,7	12,7	1

Ensimmäisellä askeleella puun jäsenet ovat siis kaikki havainnot, joten aloitetaan asettamalla hyöty $= 0$, $G = \sum g_i = 0,1$ ja $H = \sum h_i = 3$. Nyt ensimmäisellä kierroksella asetetaan $G_L = 0$ ja $H_L = 0$ ja järjestetään ensimmäinen muuttuja pituus $\{1.5, 1.65, 1.7\}$ ja lasketaan ensimmäisen jaon pisteet, siten että vasempaan haaraan menee $1,5$ ja oikeaan $1.65, 1.75$.

$$
\begin{aligned}
G_L &= 0 + 12,7 = 12,7,\ H_L = 1 \\
G_R &= 0,1 - 12,7 = -12,6,\ H_R = 3 - 1 = 2 \\
pisteet &= max\left(0, \frac{12,7^2}{1+8} + \frac{(-12,6)^2}{2+8} - \frac{0,1^2}{3+8}\right) \\
&= max(0, 17,92 + 15,88 - 0,00091) = 33,80.
\end{aligned}
$$

Eli jako on siis parempi kuin ei jakoa ollenkaan. Seuraavaksi lasketaan jako, missä

vasempaan lehteen menee $\{1.5, 1.65\}$ ja oikeaan lehteen menee 1,7.

$$\begin{aligned} G_L &= 12,7 + 0,7 = 13,4,\ H_L = 1 + 1 = 2 \\ G_R &= 0,1 - 13,4 = -13,3,\ H_R = 3 - 2 = 1 \\ pisteet &= max\left(0, \frac{13,4^2}{2+8} + \frac{(-13,3)^2}{1+8} - \frac{0,1^2}{3+8}\right) \\ &= max(33,8, 17,96 + 19,65 - 0,00091) = 37,61. \end{aligned}$$

Eli jako jossa $pituus < 170$ on optimaalisin pituudelle. Seuraavaksi lasketaan samat laskelmat kahdelle muulle muuttujalle. Molemmissa on vain kaksi ryhmää, joten on vain yksi tapa jakaa havainnot. Olkoon vasemmassa lehdessä havainnot, jonka lempi väri on sininen ja oikeassa vihreä.

$$\begin{aligned} G_L &= 12,7 - 13,3 = -0,6,\ H_L = 1 + 1 = 2 \\ G_R &= 0,1 - (-0,6) = 0,7,\ H_R = 3 - 2 = 1 \\ pisteet &= max\left(0, \frac{(-0,6)^2}{2+8} + \frac{0,7^2}{1+8} - \frac{0,1^2}{3+8}\right) \\ &= max(37,61, 0,036 + 0,054 - 0,00091) = 37,61. \end{aligned}$$

Eli lempi väri ei jaa havaintoja paremmin kuin pituus. Lasketaan sama vielä sukupuolelle.

$$\begin{aligned} G_L &= 0 - 13,3 = -13,3,\ H_L = 1 \\ G_R &= 0,1 - (-13,3) = 13,4,\ H_R = 3 - 1 = 2 \\ pisteet &= max\left(37,61, \frac{(-13,3)^2}{1+8} + \frac{13,4^2}{2+8} - \frac{0,1^2}{3+8}\right) \\ &= max(37,61, 19,654 + 17,956 - 0,00091) = 37,61. \end{aligned}$$

Saadaan samat pisteet kuin pituuden parhaalla jaolla. Valitaan muuttujaksi pituus siten, että vasempaan lehteen menee havainnot kaksi ja kolme ja oikeaan lehteen ensimmäinen havainto. Vasemman lehden voi vielä jakaa, eli tehdään algoritmi vielä uudestaan. Asetetaan taas hyöty $= 0$, $G = \sum g_i = 0,7 + 12,7 = 13,4$ ja $H = \sum h_i = 2$. Ainoa muuttuja jonka mukaan nämä rivit voidaan jakaa on lempi väri, joten saadaan seuraavat arvot

$$\begin{aligned} G_L &= 0 + 12,7 = 12,7,\ H_L = 1 \\ G_R &= 13,4 - 12,7 = 0,7,\ H_R = 2 - 1 = 1 \\ pisteet &= max\left(0, \frac{12,7^2}{1+8} + \frac{0,7^2}{1+8} - \frac{13,4^2}{2+8}\right) \\ &= max(0, 17,921 + 0,0544 - 17,956) = 0. \end{aligned}$$

Eli ei valita uutta jakoa ja ensimmäinen puu on luotu.

Seuraavaksi esitetään ahneen algoritmin approksimaatio.

Algorithm 2: Approksimaatioalgoritmi puun jakamiseen

for $k = 1$ *to* m **do**
  Ehdotetaan jakoa $S_k = \{s_{k1}, s_{k2}, ..., s_{kl}\}$ prosenttiosuuksittain muuttujasta k.
  Ehdotus voidaan tehdä per puu (globaali) tai per jako (lokaali).
end
for $k = 1$ *to* m **do**
  $G_{kv} \leftarrow \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$
  $G_{kv} \leftarrow \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$
end
Noudata samaa toimenpidetä kuin aikaisemmassa osassa löytääksesi pisteet vain ehdotettujen jakojen kesken.

Algoritmi ehdottaa ensin jäsenten jakopisteitä muuttujan jakauman prosenttiosuuksien mukaan. Seuraavaksi algoritmi jaottelee jatkuvat muuttujat laatikoihin jakopisteiden mukaan ja laskee näistä tilastotiedot. Tämän jälkeen valitaan paras jakoehdotus tilastotietojen perusteella.

Approksimaatio algoritmista on kaksi versiota riippuen siitä, milloin ehdotus annetaan. Globaali versio ehdottaa kaikkia jakopisteitä kerralla puun rakentamisvaiheessa ja käyttää samoja jakopisteitä kaikilla puun tasoilla. Lokaali versio uudelleen ehdottaa jokaisen solmun jakautumisen jälkeen. Globaali versio tarvitsee vähemmän jakopiste ehdotuksia kuin lokaali versio, mutta toisaalta globaali versio yleensä tarvitsee enemmän ehdotettuja jakopisteitä kuin lokaali versio, koska ehdotuksia ei uudisteta jakojen jälkeen.

4.4 Painotettu kvantiililuonnos

Approksimaatio algoritmin tärkein osuus on löytää ehdokas jakopisteet. Usein muuttujien prosenttiosuuksia käyttäen ehdokas jakopisteet jaetaan tasaisesti koko dataan.

Määritelmä 6. (Järjestysfunktio)[5] Joukko $\mathcal{D}_k = \{(x_{1k}, h_1), (x_{2k}, h_2), ..., (x_{nk}, h_n)\}$ esittää k:nnen muuttujan arvoja ja toisen kertaluvun gradienttistatistiikkaa kaikista opetushavainnoista. Olkoon järjestysfunktio $r_k : \mathbb{R} \to [0, +\infty)$ siten, että

$$r_k(z) = \frac{1}{\sum_{(x,h) \in \mathcal{D}_k} h} \sum_{(x,h) \in \mathcal{D}_k, x < z} h.$$

Tämä on osuus niistä tapauksista, joissa muuttujan arvo k on pienempi kuin z. Tavoite on löytää jakoehdotuspisteet $\{s_{k1}, s_{k2}, ..., s_{kl}\}$, jotka toteuttavat

$$|r_k(s_{k,j}) - r_k(s_{k,j+1})| < \epsilon, s_{k1} = \min_i \mathbf{x}_{ik}, s_{kl} = \max_i \mathbf{x}_{ik}.$$

Tässä ϵ on approksimaatiotekijä. Intuitiivisesti tämä tarkoittaa sitä, että jaossa on noin $\frac{1}{\epsilon}$ ehdotusjakopistettä. Järjestysfunktion jokainen havainto painotetaan h_i:llä.

Jotta olisi selvää, miksi h_i kuvaa painoa muutetaan kaava (7) seuraavaan muotoon

$$\sum_{i=1}^{n} \frac{1}{2} h_i (f_t(\mathbf{x}_i) - \frac{g_i}{h_i})^2 + \Omega(f_t) + vakio,$$

mikä on painotettu neliöhäviö muuttujalla $\frac{g_i}{h_i}$ ja painolla h_i. Isolle datalle ehdotusjaon löytäminen, joka täyttää yllä mainitun kriteerin on epätriviaalia. Kun jokaisella havainnolla on sama paino, niin kvantiilluonnosalgoritmi ratkaisee ongelman. Jos painot eivät ole samoja niin tällöin ei ole olemassa algoritmia, joka ratkaisee ongelman.

Tämän ongelman ratkaisemiseksi, esitetään uusi jakautumapainotettu kvantiililuonnosalgoritmi, joka pystyy käsittelemään painotettua dataa teoreettisesti todistettavalla takuulla. Yleinen ajatus on tehdä datarakenne joka tukee *yhdistämis-* ja *karsimis*operaatioita, jotka todistetusti pitää tietyn tarkkuustason.

Määritelmä 7. (Yhdistämisoperaatio)[5] Yhdistämisoperaatio yhdistää kaksi ehdotusjakopistettä, millä on approksimaatiovirheet ϵ_1 ja ϵ_2. Yhdistetyn pisteen approksimaatiovirhe on $\max(\epsilon_1, \epsilon_2)$

Määritelmä 8. (Karsimisoperaatio)[5] Karsimisoperaatio vähentää ehdotusjakopisteitä $b+1$ kappaleeseen, jolloin uusi approksimaatiovirhe on $\epsilon + \frac{1}{b}$.

Jotta voidaan käyttää kvantiililaskentaa approksimatiiviseen puuntehostamiseen täytyy ensin ratkaista painotetun datan kvantiilit. Tätä yleistä ongelmaa ei ole ratkaistu aikaisemmin ennen artikkelia [5].

Olkoon joukko $\mathcal{D} = \{(x_1, w_1), (x_2, w_2), ..., (x_n, w_n)\}$ siten, että $w_i \in [0, \infty)$ ja $x_i \in \mathcal{X}$. Jokainen x_i vastaa havainnon arvoa ja w_i sen painoa. Oletetaan, että kaikki havainnot $\mathcal{X}$ ovat suuruusjärjestyksessä pienimmästä suurimpaan. Määritellään kaksi järjestysfunktiota $r^-_{\mathcal{D}}, r^+_{\mathcal{D}} : \mathcal{X} \to [0, \infty)$

$$r^-_{\mathcal{D}}(y) = \sum_{(x,w)\in\mathcal{D}, x<y} w,$$

$$r^+_{\mathcal{D}}(y) = \sum_{(x,w)\in\mathcal{D}, x\leq y} w.$$

Joukko $\mathcal{D}$ voi sisältää monia havaintoja samalla arvolla x ja painolla w. Määritellään painofunktio $\omega_{\mathcal{D}} : \mathcal{X} \to [0, \infty)$

$$\omega_{\mathcal{D}}(y) = r^+_{\mathcal{D}}(y) - r^-_{\mathcal{D}}(y) = \sum_{(x,w)\in\mathcal{D}, x=y} w.$$

Lopulta määritellään koko multijoukon $\mathcal{D}$ paino summaamalla kaikkien pisteiden painot yhteen

$$\omega(\mathcal{D}) = \sum_{(x,y)\in\mathcal{D}} w.$$

Tehtävänä on siis löytää joukosta $\mathcal{D}$ estimaatit $r^-_{\mathcal{D}}(y)$ ja $r^+_{\mathcal{D}}(y)$ y:lle sekä järjestysfunktion arvot, missä $y \in \mathcal{X}$. Näillä merkinnöillä määritellään seuraavaksi *painotettujen havaintojen kvantiiliyhteenveto.*

Määritelmä 9. (Painotettujen havaintojen kvantiiliyhteenveto)[5] Joukon $\mathcal{D}$ kvantiiliyhteenveto on $\mathcal{Q}(\mathcal{D}) = (S, \overline{r}^+_{\mathcal{D}}, \overline{r}^-_{\mathcal{D}}, \overline{\omega}_{\mathcal{D}})$, missä $S = \{x_1, x_2, ..., x_k\}$ valitaan joukon $\mathcal{D}$ pisteistä seuraavilla ominaisuuksilla.

1) $x_i < x_{i+1}$ kaikille i ja x_1 sekä x_k ovat minimi- ja maksimipisteet joukossa $\mathcal{D}$

$$x_1 = \min_{(x,w)\in\mathcal{D}} x,\ x_k = \max_{(x,w)\in\mathcal{D}} x.$$

2) $\overline{r}^+_{\mathcal{D}}, \overline{r}^-_{\mathcal{D}}$, ja $\overline{\omega}_{\mathcal{D}}$ ovat S:n funktioita $S \to [0, +\infty)$, jotka toteuttavat ehdot

$$\overline{r}^-_{\mathcal{D}}(x_i) \le r^-_{\mathcal{D}}(x_i),\ \overline{r}^+_{\mathcal{D}}(x_i) \ge r^+_{\mathcal{D}}(x_i),\ \overline{\omega}_{\mathcal{D}}(x_i) \le \omega_{\mathcal{D}}(x_i).$$

Epäyhtälöiden yhtäsuuruus pätee vain maksimi- ja minimipisteille.

3) Funktio toteuttaa seuraavat ehdot

$$\overline{r}^-_{\mathcal{D}}(x_i) + \overline{\omega}_{\mathcal{D}}(x_i) \le \overline{r}^-_{\mathcal{D}}(x_{i+1}),\ \overline{r}^+_{\mathcal{D}}(x_i) \le \overline{r}^+_{\mathcal{D}}(x_{i+1}) - \overline{\omega}_{\mathcal{D}}(x_i)$$

Määritelmä 10. (Kuvauksen määrittelyjoukon laajentaminen)[5] Laajennetaan aikaisemman määritelmän kvantiiliyhteenvedon $\mathcal{Q}(\mathcal{D}) = (S, \overline{r}^+_{\mathcal{D}}, \overline{r}^-_{\mathcal{D}}, \overline{\omega}_{\mathcal{D}})$ määrittelyjoukko S funktioihin $\mathcal{X} \to [0, +\infty)$. Tällöin
Jos $y < x_1$:

$$\overline{r}^-_{\mathcal{D}}(y) = 0,\ \overline{r}^+_{\mathcal{D}}(y) = 0,\ \overline{\omega}_{\mathcal{D}}(y) = 0$$

Jos $y > x_k$:

$$\overline{r}^-_{\mathcal{D}}(y) = \overline{r}^+_{\mathcal{D}}(x_k),\ \overline{r}^+_{\mathcal{D}}(y) = \overline{r}^+_{\mathcal{D}}(x_k),\ \overline{\omega}_{\mathcal{D}}(y) = 0$$

Jos $y \in (x_i, x_{i+1})$ jollekin i :

$$\begin{aligned} \overline{r}^-_{\mathcal{D}}(y) &= \overline{r}^-_{\mathcal{D}}(x_i) + \overline{\omega}_{\mathcal{D}}(x_i) \\ \overline{r}^+_{\mathcal{D}}(y) &= \overline{r}^+_{\mathcal{D}}(x_{i+1}) - \overline{\omega}_{\mathcal{D}}(x_{i+1}) \\ \overline{\omega}_{\mathcal{D}}(x_y) &= 0 \end{aligned}$$

Lemma 1. *(Laajennettu rajoitus)[5] Laajennettu määritelmä funktioille $\overline{r}^-_{\mathcal{D}}$, $\overline{r}^+_{\mathcal{D}}$ ja $\overline{\omega}_{\mathcal{D}}$ täyttää seuraavat rajoitukset*

$$\overline{r}^-_{\mathcal{D}}(y) \le r^-_{\mathcal{D}}(y),\ \overline{r}^+_{\mathcal{D}}(y) \ge r^+_{\mathcal{D}}(y),\ \overline{\omega}_{\mathcal{D}}(y) \le \omega_{\mathcal{D}}(y)$$

$$\overline{r}^-_{\mathcal{D}}(y) + \overline{\omega}_{\mathcal{D}}(y) \le \overline{r}^-_{\mathcal{D}}(y),\ \overline{r}^+_{\mathcal{D}}(y) \le \overline{r}^+_{\mathcal{D}}(x) - \overline{\omega}_{\mathcal{D}}(x),\ \textit{kaikilla } y < x.$$

Tämän todistus on esitetty artikkelissa [5].

Esitetään seuraavaksi ϵ-approksimaatio kvantiiliyhteenveto, jolla voidaan approksimoida $r^+(y)$ ja $r^-(y)$ tietyllä tarkkuudella.

Määritelmä 11. (ϵ-approksimaatio kvantiiliyhteenveto)[5] Olkoon kvanttiiliyhteenveto $\mathcal{Q}(\mathcal{D}) = (S, \overline{r}^+_{\mathcal{D}}, \overline{r}^-_{\mathcal{D}}, \overline{\omega}_{\mathcal{D}})$. Jos jokin $y \in \mathcal{X}$, tällöin kyseessä on ϵ-approksimaatio kvantiiliyhteenveto.

$$\overline{r}^+_{\mathcal{D}}(y) - \overline{r}^-_{\mathcal{D}}(y) - \overline{\omega}_{\mathcal{D}}(y) \le \epsilon\omega(\mathcal{D}) \tag{11}$$

Tätä määritelmää voidaan käyttää, koska tiedetään, että $r^-(y) \in [\overline{r}^-_{\mathcal{D}}(y), \overline{r}^+_{\mathcal{D}}(y) - \overline{\omega}_{\mathcal{D}}(y)]$ ja $r^+(y) \in [\overline{r}^-_{\mathcal{D}}(y) + \overline{\omega}_{\mathcal{D}}(y), \overline{r}^+_{\mathcal{D}}(y)]$. Yhtälön (11) avulla voidaan estimoida $r^+(y)$ ja $r^-(y)$ virheellä $\epsilon\omega(\mathcal{D})$.

Lemma 2. *[5] Kvantiiliyhteenveto* $\mathcal{Q}(\mathcal{D}) = (S, \overline{r}^+_{\mathcal{D}}, \overline{r}^-_{\mathcal{D}}, \overline{\omega}_{\mathcal{D}})$ *on* ϵ*-approksimaatio yhteenveto, jos ja vain jos seuraavat kaksi ehtoa täyttyvät.*

$$\overline{r}^+_{\mathcal{D}}(x_i) - \overline{r}^-_{\mathcal{D}}(x_i) - \overline{\omega}_{\mathcal{D}}(x_i) \leq \epsilon\omega(\mathcal{D}) \tag{12}$$

$$\overline{r}^+_{\mathcal{D}}(x_{i+1}) - \overline{r}^-_{\mathcal{D}}(x_i) - \overline{\omega}_{\mathcal{D}}(x_{i+1}) - \overline{\omega}_{\mathcal{D}}(x_i) \leq \epsilon\omega(\mathcal{D}) \tag{13}$$

Eli yhtälöt (12) *ja* (13) *antavat yhtälön* (11). *Tämän todistus on esitetty artikkelissa [5].*

Nyt ollaan siis esitetty määrittelyjoukon laajennus funktioille $\overline{r}^+_{\mathcal{D}}$, $\overline{r}^-_{\mathcal{D}}$, $\overline{\omega}_{\mathcal{D}}$, $\mathcal{X} \to [0, +\infty)$. Tässä kappaleessa siis yleistettiin alkuperäinen funktion rajoitukset koskemaan myös laajennettua funktiota. Lemma 1 ja 2 osoittaa, että alkuperäisen funktion rajoitukset johtavat yleisempiin rajoituksiin laajennetulla funktiolla.

4.5 Yhdistämisoperaatio

Tässä kappaleessa määritellään, miten kaksi yhteenvetoa yhdistetään. Olkoon $\mathcal{Q}(\mathcal{D}_1)$ $= (S_1, \overline{r}^+_{\mathcal{D}_1}, \overline{r}^-_{\mathcal{D}_1}, \overline{\omega}_{\mathcal{D}_1})$ ja $\mathcal{Q}(\mathcal{D}_2) = (S_2, \overline{r}^+_{\mathcal{D}_2}, \overline{r}^-_{\mathcal{D}_2}, \overline{\omega}_{\mathcal{D}_2})$ kahden datan kvantiiliyhteenvedot. Olkoon $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ ja määritellään yhdistetty yhteenveto $\mathcal{Q}(\mathcal{D}) = (S, \overline{r}^+_{\mathcal{D}}, \overline{r}^-_{\mathcal{D}}, \overline{\omega}_{\mathcal{D}})$ seuraavasti

$$S = \{x_1, x_2, ..., x_k\},\ x \in S_1 \text{ tai } x_i \in S_2.$$

Joukon S pisteet ovat joukkojen S_1 ja S_2 pisteiden kombinaatio ja funktiot $\overline{r}^+_{\mathcal{D}}$, $\overline{r}^-_{\mathcal{D}}$, $\overline{\omega}_{\mathcal{D}}$ määritellään seuraavasti

$$\overline{r}^-_{\mathcal{D}}(x_i) = \overline{r}^-_{\mathcal{D}_1}(x_i) + \overline{r}^-_{\mathcal{D}_2}(x_i),$$

$$\overline{r}^+_{\mathcal{D}}(x_i) = \overline{r}^+_{\mathcal{D}_1}(x_i) + \overline{r}^+_{\mathcal{D}_2}(x_i),$$

$$\overline{\omega}_{\mathcal{D}}(x_i) = \overline{\omega}_{\mathcal{D}_1}(x_i) + \overline{\omega}_{\mathcal{D}_2}(x_i).$$

Tässä funktiot ovat siis määritelty $S \to [0, +\infty)$ vasemmalla puolella yhtälöitä ja laajennettuina oikealla puolella yhtälöitä. Koska funktiot r^+, r^- ja ω ovat additiivisia ominaisuuksiltaan, voidaan yhtälöt kirjoittaa seuraavasti

$$r^-_{\mathcal{D}}(x_i) = r^-_{\mathcal{D}_1}(x_i) + r^-_{\mathcal{D}_2}(x_i),$$

$$r^+_{\mathcal{D}}(x_i) = r^+_{\mathcal{D}_1}(x_i) + r^+_{\mathcal{D}_2}(x_i),$$

$$\omega_{\mathcal{D}}(x_i) = \omega_{\mathcal{D}_1}(x_i) + \omega_{\mathcal{D}_2}(x_i),$$

ja lemman 1 avulla voidaan varmistaa, että $\mathcal{Q}(\mathcal{D})$ yhtälöt täyttää määritelmän (9) ehdot, ja näin ollen on kvantiiliyhteenveto.

4.6 Karsimisoperaatio

Ennen kuin määritellään karsimisoperaatio, määritellään kyselyfunktio $g(\mathcal{Q}, d)$.

Algorithm 3: Kyselyfunktio

Input: d: $0 \leq d \leq \omega(\mathcal{D})$
Input: $\mathcal{Q}(\mathcal{D}) = (S, \overline{r}^+_{\mathcal{D}}, \overline{r}^-_{\mathcal{D}}, \overline{\omega}_{\mathcal{D}})$, missä $S = x_1, x_2, ..., x_k$

if $d < \frac{1}{2}[\overline{r}^-_{\mathcal{D}}(x_1) + \overline{r}^+_{\mathcal{D}}(x_1)]$ **then return** x_1;
if $d \geq \frac{1}{2}[\overline{r}^-_{\mathcal{D}}(x_k) + \overline{r}^+_{\mathcal{D}}(x_k)]$ **then return** x_k;
Etsitään i jolle pätee
$\frac{1}{2}[\overline{r}^-_{\mathcal{D}}(x_i) + \overline{r}^+_{\mathcal{D}}(x_i)] \leq d \leq \frac{1}{2}[\overline{r}^-_{\mathcal{D}}(x_{i+1}) + \overline{r}^+_{\mathcal{D}}(x_{i+1})]$
if $2d < \overline{r}^-_{\mathcal{D}}(x_i) + \overline{\omega}_{\mathcal{D}}(x_i) + \overline{r}^+_{\mathcal{D}}(x_{i+1}) - \overline{\omega}_{\mathcal{D}}(x_{i+1})$ **then**
    return x_i
else
    return x_{i+1}
end

Annetulle järjestykselle d funktio palauttaa pisteen x joka on lähimpänä järjestystä d. Esitetään tämä ominaisuus seuraavassa lemmassa.

Lemma 3. *Annetulle ϵ-approksimaatio yhteenvedolle $\mathcal{Q}(\mathcal{D}) = (S, \overline{r}^+_{\mathcal{D}}, \overline{r}^-_{\mathcal{D}}, \overline{\omega}_{\mathcal{D}})$, $x^* = g(\mathcal{Q}, d)$ täyttää seuraavat ominaisuudet*

$$d \geq \overline{r}^+_{\mathcal{D}}(x^*) - \overline{\omega}_{\mathcal{D}}(x^*) - \frac{\epsilon}{2}\omega(\mathcal{D}),$$

$$d \leq \overline{r}^-_{\mathcal{D}}(x^*) + \overline{\omega}_{\mathcal{D}}(x^*) + \frac{\epsilon}{2}\omega(\mathcal{D}).$$

Tämän todistus on esitetty artikkelissa [5].

Esitetään seuraavaksi vielä karsimisoperaatio. Olkoon $\mathcal{Q}(\mathcal{D}) = (S, \overline{r}^+_{\mathcal{D}}, \overline{r}^-_{\mathcal{D}}, \overline{\omega}_{\mathcal{D}})$, $x^* = g(\mathcal{Q}, d)$ kvantiiliyhteenveto, missä $S = \{x_1, x_2, ..., x_k\}$ ja *muistibudjetti* on b. Karsimisoperaatio luo uuden yhteenvedon $\mathcal{Q}'(\mathcal{D}) = (S', \overline{r}^+_{\mathcal{D}}, \overline{r}^-_{\mathcal{D}}, \overline{\omega}_{\mathcal{D}})$, $S' = \{x'_1, x'_2, ... , x'_{b+1}\}$, missä x'_i valitaan kyselyfunktiolla seuraavasti

$$x'_i = g\big(\mathcal{Q}, \frac{i-1}{b}\omega(\mathcal{D})\big).$$

Funktioiden $\overline{r}^+_{\mathcal{D}}, \overline{r}^-_{\mathcal{D}}, \overline{\omega}_{\mathcal{D}}$ alkuperäinen määritelmä säilyy ennallaan yhteenvedolle $\mathcal{Q}'$ pienentämällä lähtöjoukon S joukkoon S'. Joukossa S' voi olla duplikaattialkioita, jotka voidaan poistaa *muistikustannuksen* laskemiseksi. Kaikki elementit siirtyvät yhteenvedosta $\mathcal{Q}$ yhteenvetoon $\mathcal{Q}'$, joten voidaan todeta, että määritelmän (9) ehdot toteutuvat ja uusi yhteenveto on kvantiiliyhteenveto.

5 Vahinkomenon ennustaminen XGBoost -algoritmilla

Tässä luvussa sovelletaan XGBoost -algoritmia Kagglen ”car insurance claim” -aineistoon. Tavoitteena on ennustaa aineistosta vakuutusyhtiön asiakkaiden vahinkomenoa. Tässä sovelluksessa käytetään Pythonin versiota 3 ja taulukossa esiteltyjä paketteja (taulukko 5.1)

Taulukko 1: Python paketit

Paketti	Selitys	Dokumentaatio
Pandas	NumFocuksen rahoittama avoimen lähdekoodin paketti datankäsittelyyn	https://pandas.pydata.org/pandas-docs/stable/
Numpy	NumFocuksen rahoittama avoimen lähdekoodin paketti vektori laskentaan	https://numpy.org/devdocs/reference/
Sklearn	Monen teknologia yrityksen rahoittama koneoppimiskirjasto. Käytetään pakettia "train test split"tekemään satunnainen opetus- ja testidata	https://scikit-learn.org/dev/_downloads/scikit-learn-docs.pdf
Eli5	Koneoppimisalgoritmeja selittävä paketti	https://eli5.readthedocs.io/en/latest/
XGBoost	Avoimen lähdekoodin XGBoost -algoritmin implementaatio	https://xgboost.readthedocs.io/en/latest/
Shap	Koneoppimisalgoritmeja peliteorialla selittävä paketti	https://shap.readthedocs.io/en/latest/
Math	Pythonin alkuperäinen matemaattisten funktioiden paketti	https://docs.python.org/3.0/library/math.html

5.1 Ainesto

Aineistona käytetään Kagglen ”car insurance claim” -aineistoa. Aineistossa on 10 302 havaintoa ja 27 muuttujaa. Seuraavaksi käydään läpi, mitä muuttujia aineisto sisältää (taulukko 5.1).

Ennustettava muuttuja on vahingon suuruus. Jätetään aineistosta pois vakuutuksenottajan syntymäpäivä, koska sen avulla pystytään laskemaan vakuutuksenottajan ikä. Poistetaan myös muuttuja CLAIM_FLAG, koska se kertoo, onko vahinko tapahtunut ja sitä ei tiedetä etukäteen normaalissa tilanteessa. Muutetaan myös luokalliset muuttujat omiksi binäärisiksi muuttujiksi. Jaetaan tämä aineisto opetus- ja validointidataan siten, että opetusdatan osuus on 70 % ja validointidatan osuus on loput 30 %.

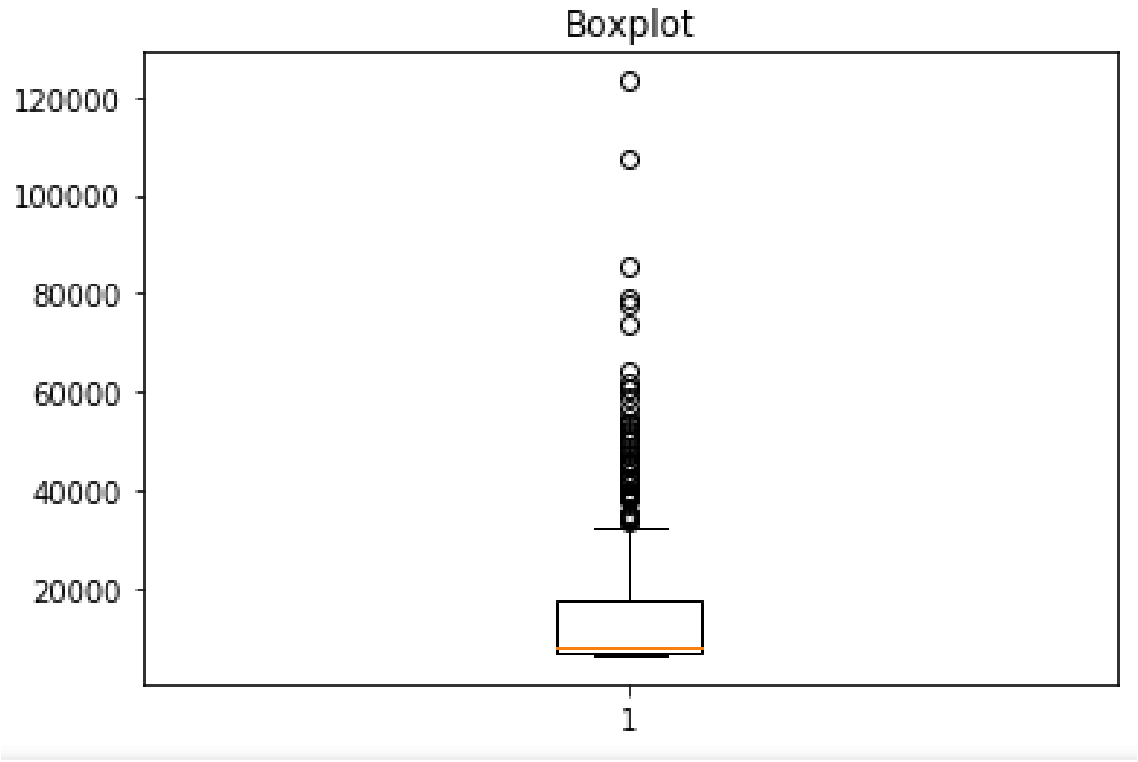
Taulukko 2: Muutujat

Muuttuja	Selitys	Tyyppi
ID	Havainnon yksilöivä tunnus	Merkkijono
KIDSDRIV	Vakuutuksenottajan lapsien lukumäärä,jotka ajavat vakuutettua autoa	Kokonaisluku
BIRTH	Vakuutuksenottajan syntymäaika	Pvm
AGE	Vakuutuksenottajan ikä	Kokonaisluku
HOMEKIDS	Kotitalouden lapsien lukumäärä	Kokonaisluku
YOJ	Asiakkuus vuosien lukumäärä	Kokonaisluku
INCOME	Vakuutuksen ottajan tulot	USD
PARENT1	Onko vakuutuksenottajalla lapsia	True/False
HOME_VAL	Omakotitalon arvo	USD
MSTATUS	Siviilisääty	Naimisissa/ei naimisissa
GENDER	Sukupuoli	Nainen/mies
EDUCATION	Koulutus	4 luokkaa
OCCUPATION	Työn kuvaus	9 luokkaa
TRAVTIME	Matkustusaika autolla	Kokonaisluku
CAR_USE	Auton käyttötarkoitus	Private/ commercial
BLUEBOOK	Auton markkinahinta	USD
TIF	Autovakuutuksen vakuutusvuodet	Kokonaisluku
CAR_TYPE	Auton tyyppi	6 luokkaa
RED_CAR	Punainen auto	True/False
OLDCLAIM	Vanhat vahingot	USD
CLM_FREQ	Vahinkofrekvenssi	Kokonaisluku
REVOKED	Peruttu vakuutussopimus	True/False
MVR_PTS	Vakuutuksenottajan MVR (Motor Vehicle Report) pisteet	Kokonaisluku
CLM_AMT	Vahingon suuruus	USD
CAR_AGE	Auton ikä	Kokonaisluku
CLAIM_FLAG	Vahinko	True/False
URBANICITY	Auton käyttösijainti	Kaupunki/ maaseutu

5.2 Ennustaminen

Ensimmäisessä mallissa häviöfunktiona käytetään neliövirhettä. Mallin hyvyyden mittarina käytetään keskineliöhajontaa eli $\sqrt{MSE}$. Tässä mallissa ennustetaan vahinkojen suuruutta dollareissa.

Ensimmäisen mallin opetusvirhe on 3 679 ja testivirhe on 4 786. Tämä ei ole mikään erityisen hyvä tulos, koska jos käytetään ennusteena havaintojen keskiarvoa, niin keskineliövirhe on 4 725, eli parempi kuin XGBoostin antama ennuste. Tutkitaan ennustettavaa muuttujaa ja katsotaan sisältääkö se poikkeavia havaintoja. Alla olevassa laatikkojanakuviossa (kuva 12) on piirrettynä 95 % kvantiilin ylittävät ha-



Kuva 12: Laatikkojanakuvio

vainnot. Poistetaan nämä poikkeavat havainnot, koska ne nostavat ennusteen liian suureksi ja luovat virhettä.

Seuraavan mallin opetusvirhe on 1358 ja testivirhe on 1 467. Tämä on jo huomattavasti parempi verrattuna aikaisempaan tulokseen, mutta jos vertaa vahinkojen keskiarvon neliövirheeseen 1 651, ei tulos ole merkitsevä. Tarkastellaan seuraavassa kappaleessa mallia tarkemmin.

5.3 Mallin tarkastelu

Aikaisemman luvun mallin opetusvirhe on 1 358 ja testivirhe on 1 467. Tarkastellaan mallia tarkemmin XGBoostiin tehtyjen mallinselitystyökalujen avulla. Tässä kappaleessa käyttetään seuraavia Python paketteja:

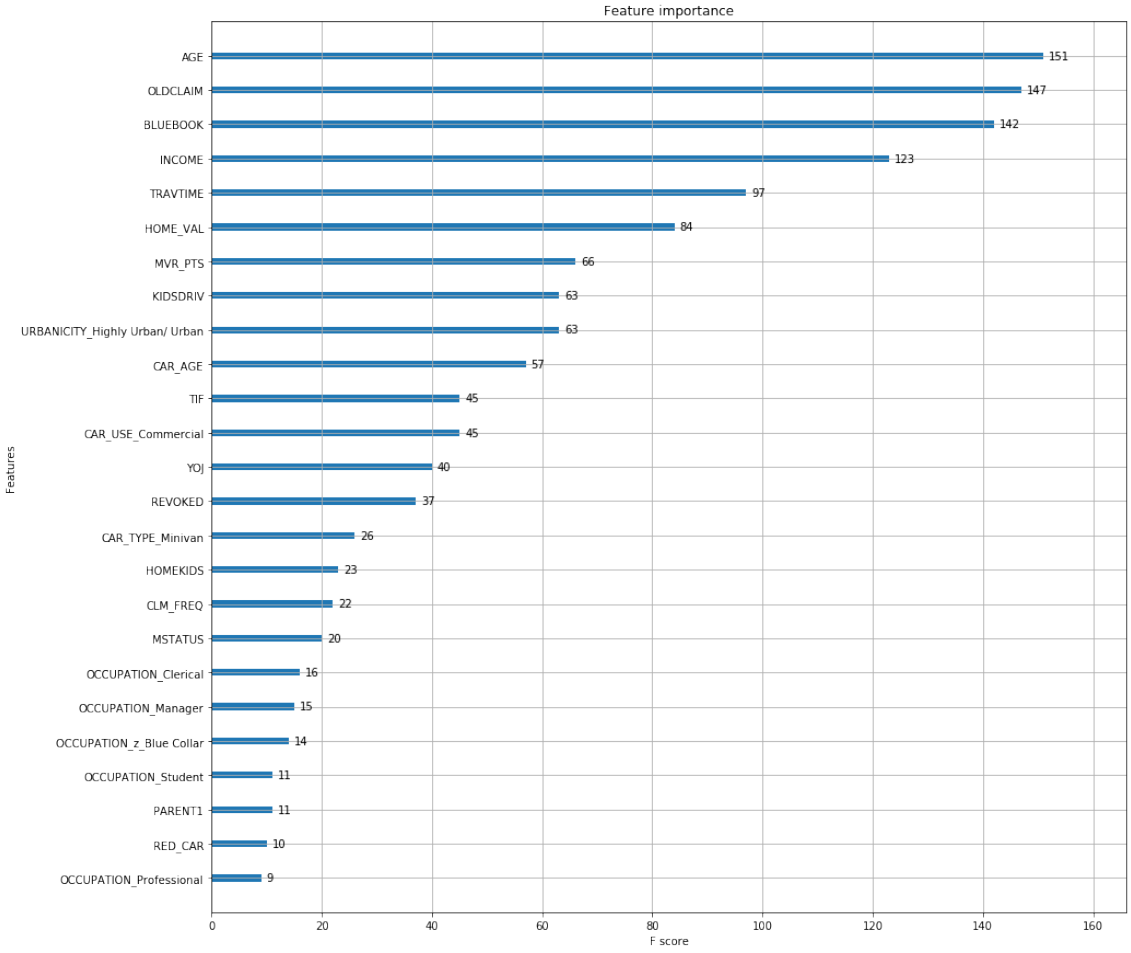
Eli5: Selittää miten malli painottaa eri muuttujia ja kuinka paljon muuttuja vaikuttaa mallin ennusteeseen.

Shap: Havainnollistaa eri muuttujien arvojen vaikutusta ennusteeseen.

Tarkastellaan ensin kuvasta 13 kuinka monta kertaa eri muuttujat esiintyvät puussa. Kuvan perusteella malliin vaikuttaa eniten ikä, vahinkohistoria ja auton markkina-arvo. Myös tuloilla, matkustusajalla ja kodin arvolla on huomattavaa vaikutusta. Työpaikalla ei näytä olevan suurta vaikutusta malliin. Tarkastellaan seuraavaksi kuvasta 14 kuinka seuraavia muuttujia painotetaan mallissa. Kuvan perusteella malli painottaa eniten yläasteen käyneitä ja sitä, onko vakuutuksenottajalla lapsia. Myös kaupunkiajossa olevat autot painottuvat enemmän. Ikää ja matkustusaikaa taas painotetaan vähän. Tämä ei vielä kerro, että vaikuttavatko nämä positiivisesti vai negatiivisesti mallin ennusteeseen. Tarkastellaan vielä tarkemmin mallia Shap -kuvaajalla.

Kuvaajassa (kuva 15) muuttujan arvo high (punainen) tarkoittaa joko 1 eli tosi tai jatkuvalla muuttujalla korkeaa arvoa, arvo low (sininen) tarkoittaa joko 0 eli epätosi tai jatkuvalla muuttujalla matalaa arvoa. Y-akselilla on muuttujat ja x-akselilla jokaisen rivin vaikutus ennusteeseen. Ensimmäisestä muuttujasta nähdään, että kaupunkiajossa olevat autot nostavat ennustetta ja ei-kaupunkiajossa olevat autot laskevat ennustetta. Vanhalla vahingolla on epäselvä vaikutus ennusteeseen. Liiketoimintakäytössä olevat autot nostavat ennustetta vähän. Matkustusaika vaikuttaa suoraan ennusteen suuruuteen, mitä enemmän autoa käyttää sen suurempi vaikutus ennusteeseen on. Auton markkinahinta vaikuttaa päinvastaisesti ennusteeseen, mitä kalliimpi auto sen pienempi ennustettu vahinko on. Nuoret ja vanhat kuskit aiheuttavat ennusteeseen kasvua ja keski-ikäiset ihmiset aiheuttavat ennusteeseen laskua.

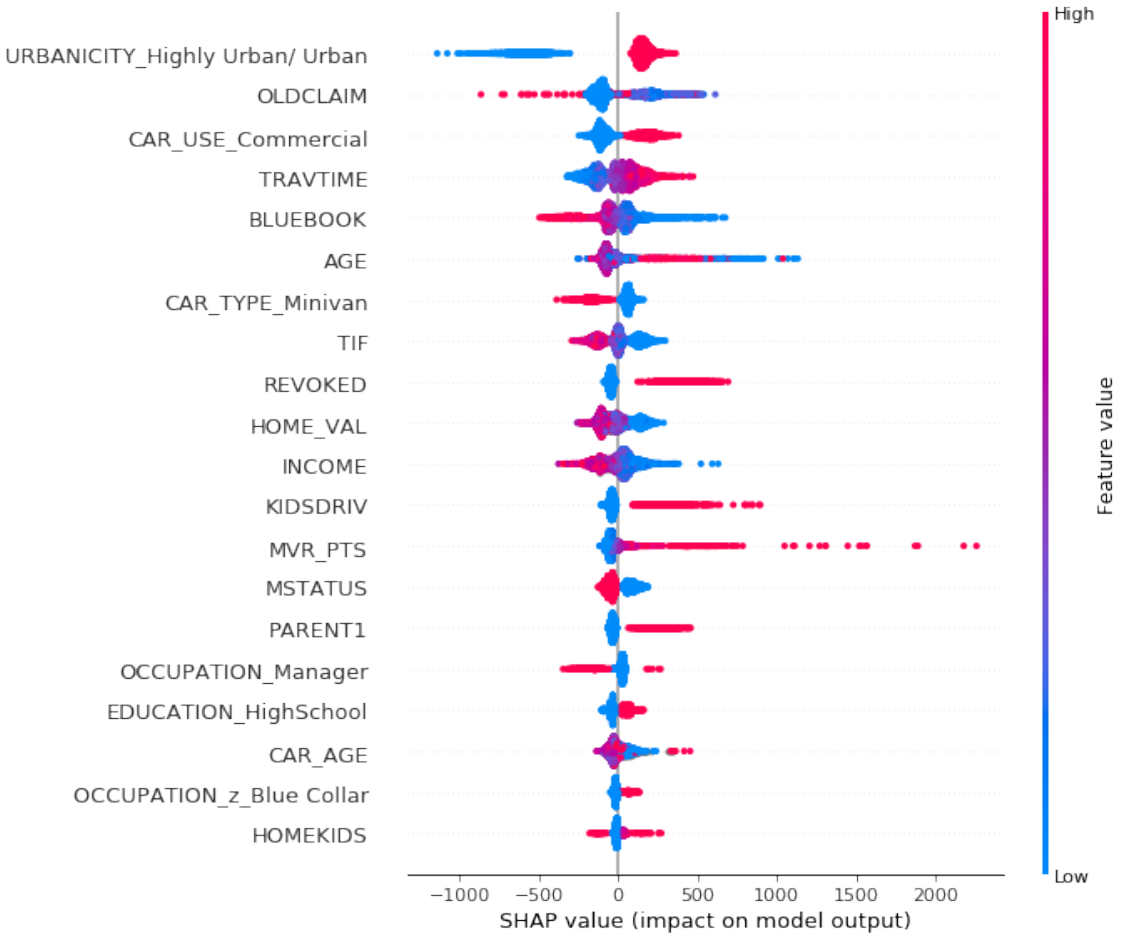
Yleisesti malliin vaikuttavat muuttujat näyttävät loogisilta, mutta esimerkiksi voisi olettaa, että auton markkinahinta vaikuttaisi positiivisesti ennusteen arvoon. Tässä tulos on päinvastainen, mutta toisaalta havainnoista poistettiin isoja vahinkoja, jotka oletettavasti ovat aiheutuneet kalliista autoista. Mallin testivirhe 1 467 on melko huono, koska jos tällä mallilla ennustettaisiin vahinkomenoa 10 000 asiakkaalle, pahimmassa tapauksessa menoa ennustettaisiin 14 670 000 dollaria liian vähän. Mallin opettamiseen käytettiin 5 871 havaintoa ja testaamiseen 3 915 havaintoa. Tämä on selvästi liian vähän ja suuremmalla datalla päädyttäisiin parempiin tuloksiin. Varsinkin tällöin pystyttäisiin ennustamaan myös suuret vahingot paremmin.



Kuva 13: Feature importance

Weight	Feature
0.1505	EDUCATION_HighSchool
0.1433	PARENT1
0.0717	URBANICITY_Highly Urban/ Urban
0.0431	REVOKED
0.0386	OLDCLAIM
0.0368	OCCUPATION_Manager
0.0367	MVR_PTS
0.0345	CAR_TYPE_Minivan
0.0318	OCCUPATION_z_Blue Collar
0.0302	HOME_VAL
0.0292	CAR_USE_Commercial
0.0207	INCOME
0.0207	KIDSDRIV
0.0198	BLUEBOOK
0.0196	TIF
0.0186	MSTATUS
0.0185	OCCUPATION_Student
0.0177	CAR_TYPE_Sports Car
0.0176	TRAVTIME
0.0175	AGE
	... 21 more ...

Kuva 14: Painot



Kuva 15: Shap kuvaaja

6 Elinajanodotteen ennustaminen XGBoost -algoritmilla

Elinajanodotteeseen vaikuttaa monta tekijää ja niiden yhteisvaikutuksen tulkitseminen on vaikeaa. Tässä luvussa sovelletaan XGBoost -algoritmia WHO:n "elinajanodote" -aineistoon.

6.1 Aineisto

Aineistona käytetään Maailman terveysjärjestö WHO:n "elinajanodote" -aineistoa vuosille $2000 - 2015$ (*https* : *//www.kaggle.com/kumarajarshi/life−expectancy−who*). Aineistossa on 1 649 riviä ja 22 muuttujaa. Aineiston muuttujat sisältävät tietoa erilaisista sairauksista, aikuis- ja lapsikuolleisuudesta, maasta ja maan taloudellisesta tilanteesta. Tarkemmat tiedot muuttujista löytyvät yllä mainitusta linkistä.

6.2 Ennustaminen

Ensimmäisessä mallissa häviöfunktiona käytetään neliövirhettä. Mallin hyvyyden mittarina käytetään keskineliöhajontaa eli $\sqrt{MSE}$. Mallissa ennustetaan elinajanodotetta vuosina.

Ensimmäisen mallin opetusvirhe on 0,097 vuotta ja testivirhe on 2,65 vuotta. Tämä on melko hyvä tulos näin pitkän aikavälin ennusteeksi. Opetusvirhe on niin pieni, että malli vaikuttaisi ylisovittuvan, eli se oppii opetusdatan liian tarkasti. Pienennetään mallin kompleksisuutta muuttamalla puun maksimisyvyyttä kolmeen solmuun. Aikaisemmin maksimisyvyys oli seitsemän solmua. Tämä kolmen solmun syvyys saadaan yrittämällä eri syvyyksiä, kunnes opetusvirhe kasvaa liikaa tai testivirhe ei enää laske. Tämä vähentää muuttujien lukumäärää jokaiselle puulle ja näin ollen vähentää mallin kompleksisuutta. Nyt opetusvirheeksi saadaan 1,59 ja testivirheeksi 2,42.

6.3 Mallin tarkastelu

Aikaisemman luvun mallin opetusvirhe on 1,59 ja testivirhe on 2,42. Tarkastellaan mallia tarkemmin XGBoostiin tehtyjen mallinselitystyökalujen avulla.

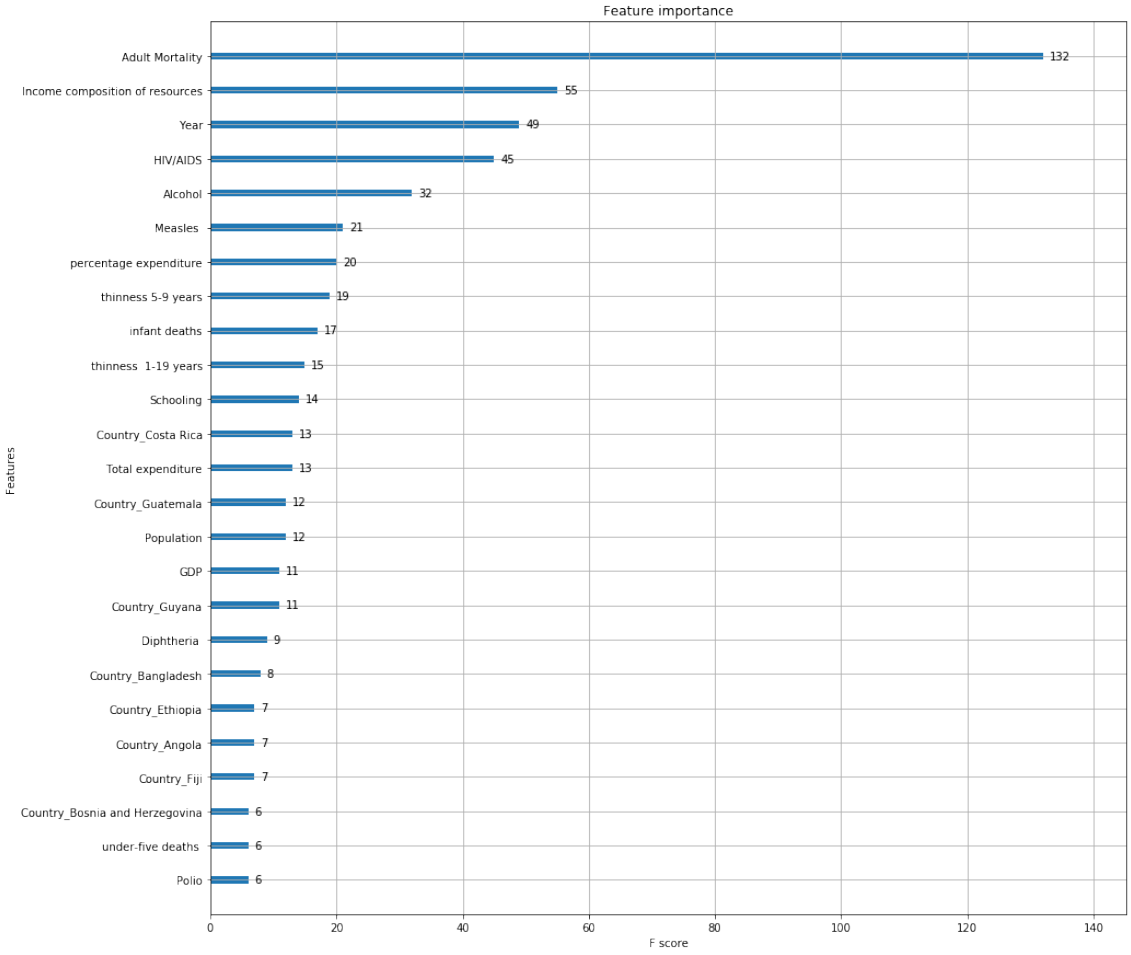
Tarkastellaan ensin kuvasta 16, kuinka monta kertaa eri muuttujat esiintyvät puussa. Kuvaajan perusteella malliin vaikuttavat eniten aikuiskuolleisuus, tulon ja resurssien jakautuminen, vuosi, alkoholi ja sairaudet. Myös lapsikuolleisuudella, koulutuksella, väkiluvulla ja GDP:llä on huomattavaa vaikutusta.

Tarkastellaan seuraavaksi kuvasta 17, kuinka seuraavia muuttujia painotetaan mallissa. Tämän perusteella malli painottaa eniten tulon ja resurssien jakaumaa ja aikuiskuolleisuutta. Myös HIV/AIDS -sairautta painotetaan paljon. Alkoholin painotus on melko alhainen. Tämä ei vielä kerro, että vaikuttavatko nämä muuttujat positiivisesti vai negatiivisesti mallin ennusteeseen.

Tarkastellaan vielä tarkemmin mallia Shap-kuvaajalla (kuva 18). Ensimmäisestä muuttujasta nähdään, että suuri aikuiskuolleisuus vähentävät elinajanodotetta ja pieni aikuiskuolleisuus nostaa sitä. Pienet tulot ja resurssit vähentävät ennustetta ja

suuret nostavat. Maissa, joissa esiintyy AID:sia, elinajanodote ennustetaan pienemmäksi kuin maissa, joissa sitä ei esiinny. Alkoholin vaikutus näyttäisi olevan pieni eikä tulkittavissa kuvaajasta.

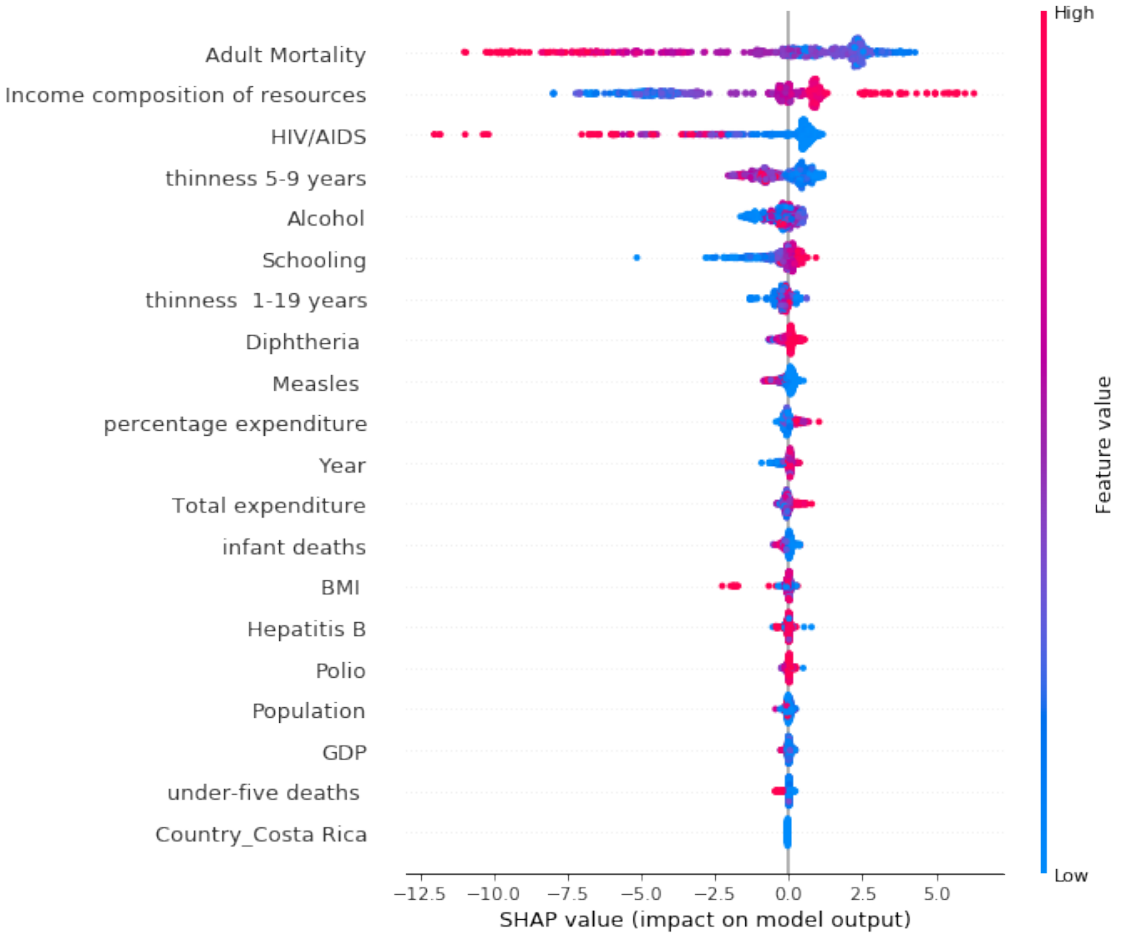
Yleisesti malliin vaikuttavat muuttujat näyttävät loogisilta, mutta alkoholin vaikutuksen luulisi olevan suurempi. Mallin testivirhe 2,42 on todella hyvä verrattuna siihen, kuinka monta havaintoa aineistossa on. Esimerkiksi WHO:n ennusteiden keskiarvo samalle aikavälille on 69,76 vuotta. Ennusteen keskiarvo on 67,89 vuotta, eli vähän alle kaksi vuotta vähemmän.



Kuva 16: Feature importance

Weight	Feature
0.4063	Income composition of resources
0.1450	Adult Mortality
0.1328	HIV/AIDS
0.0578	thinness 5-9 years
0.0466	Country_Burundi
0.0247	Country_Eritrea
0.0161	Country_Belgium
0.0154	Country_Fiji
0.0150	Country_Angola
0.0128	Schooling
0.0088	Country_Costa Rica
0.0077	Country_Indonesia
0.0061	thinness 1-19 years
0.0059	Country_Mali
0.0055	Diphtheria
0.0055	Country_Guatemala
0.0054	Alcohol
0.0045	Country_Chad
0.0044	Country_Guyana
0.0040	Country_Belize

Kuva 17: Painot

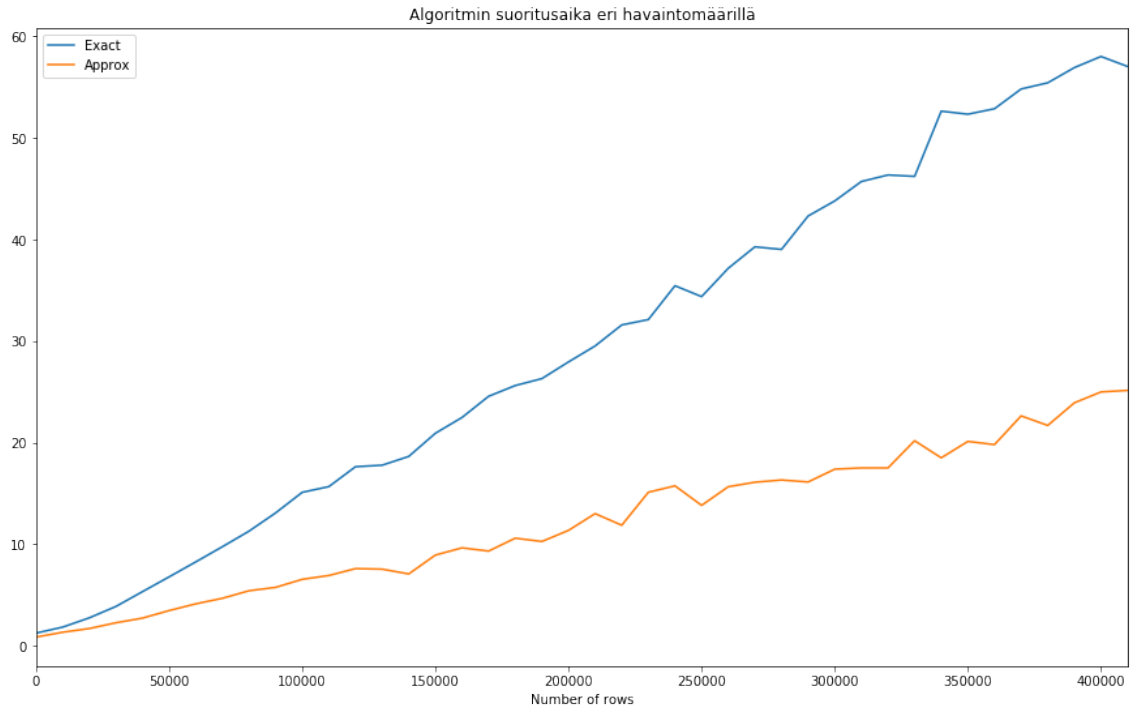


Kuva 18: Shap kuvaaja

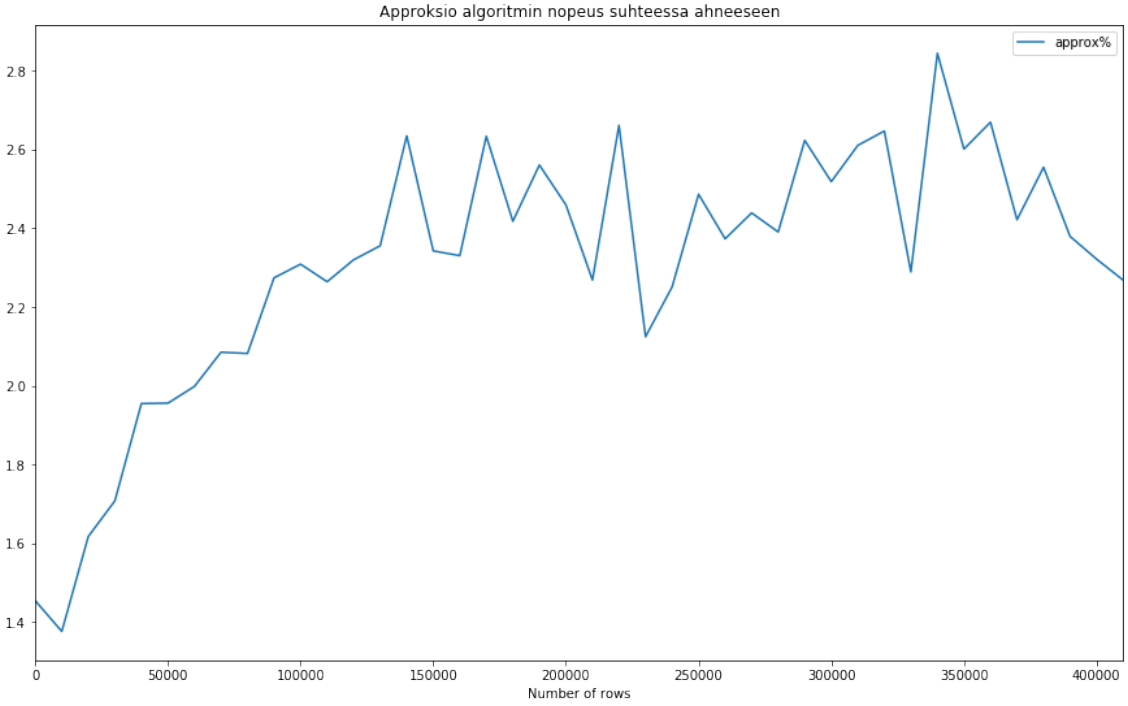
7 XGBoost suorituskykytesti

Tarkastellaan vielä, kuinka tehokkaasti XGBoost -algoritmi opettaa mallin. Käytetään tarkasteluun Brooklynin asuntomyyntidataa (https://www.kaggle.com /tianhwu/brooklynhomes2003to2017/data#). Datassa on 390 883 havaintoa ja 111 muuttujaa. Otetaan suorituskykytestiin 75 muuttujaa ja otetaan datasta otoksia 10 000:n rivin välein ja opetetaan malli sekä ahneella algoritmilla että approksimaatio algoritmilla. Tulokset ovat kuvassa 19. x-akselilla on havaintojen lukumäärä ja y-akselilla on mallin opetusaika sekunteina. Testi tehtiin Intelin i7-8550U CPU @ 1,80GHz -suorittimella.

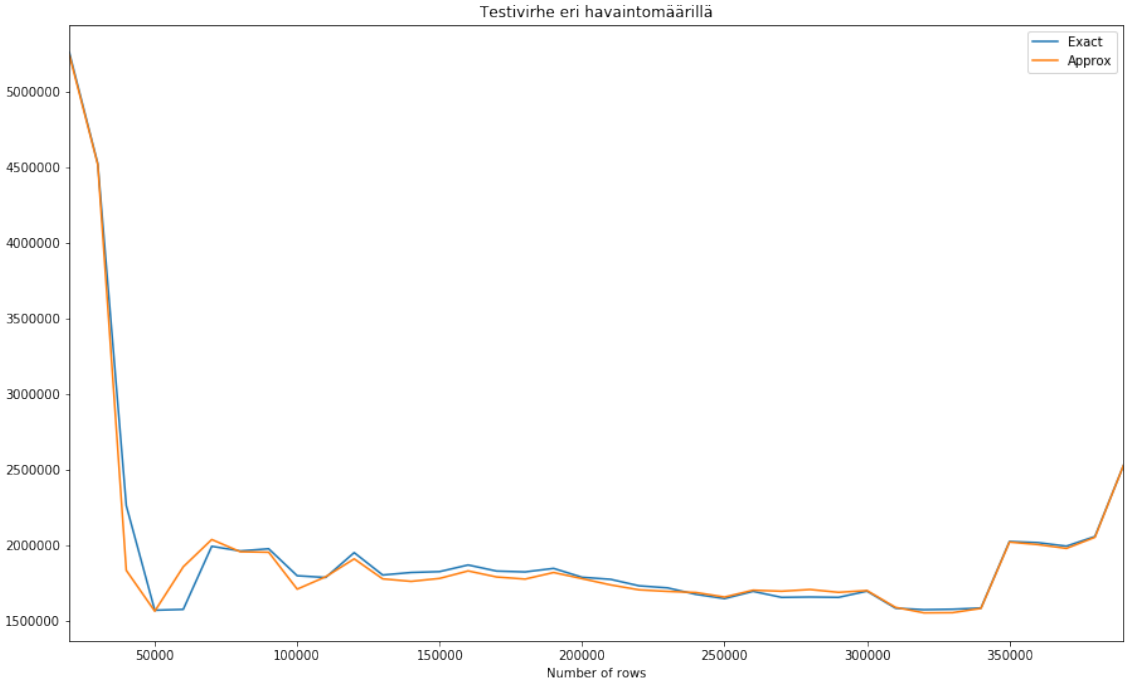
Molemmat algoritmit näyttävät pärjäävän hyvin testissä, mutta approksimaatio algoritmi on selvästi nopeampi. 100 000 rivin jälkeen ero näyttää tasaantuvan ja approksimaatio algoritmi on noin 2,5 kertaa nopeampi (kuva 20). 400 000 rivin opettamiseen meni approksimaatio algoritmilla noin 22 sekuntia ja ahneella algoritmilla 57 sekuntia. Algoritmien testivirheissäkään ei näyttäisi olevan eroa (kuva 21), varsinkin loppupäässä ne näyttäisi olevan täysin samat. Tämän perusteella XGBoost -algoritmi ei vaadi suurta suoritustehoa mallin opettamiseen ja on hyvä valinta myös tavallisella tietokoneella.



Kuva 19: Algoritmin suoritusajan vertailu



Kuva 20: Approksimaatio algoritmin nopeus suhteessa ahneeseen algoritmiin



Kuva 21: Algoritmin testivirheen vertailu

8 Johtopäätökset

Tässä tutkielmassa esitettiin gradienttitehostettu päätöspuu -ennustemalli yleisesti ja syvennyttiin XGBoost -algoritmin teoriaan. Malli opetetaan additiivisesti iteroiden yksi päätöspuu kerrallaan. Algoritmi ratkaisee iteraation optimaalisen päätöspuun joko ahneella- tai approksimaatioalgoritmilla. XGBoost -algoritmi tukee kaikkia differentioituvia häviöfunktioita.

Tämän tutkielman tavoitteena oli ennustaa gradienttitehostetulla päätöspuulla autovahinkojen vahinkomenoa. Tämän lisäksi ennustettiin myös elinajanodotetta. Sovelluksissa käytettiin XGBoostin Python implementointia ja häviöfunktiona neliövirhettä. Tulosten selittämiseen käytettiin Pythonin eli5 ja shap paketteja.

Autovahinkojen vahinkomenon ennustamisessa päästiin kohtuullisiin tuloksiin. Ennusteen keskineliövirhe oli 1467 dollaria ja opetusvirhe oli 1358 dollaria. Malliin eniten vaikuttavia muuttujia olivat kaupunkiajo, vahinkohistoria ja matkustusaika. Auton markkinahinnalla oli alkuperäistä intuitiota päinvastainen vaikutus ennusteeseen, mutta toisaalta kalliimmilla autoilla ajetaan yleensä huolellisemmin. Parempiin tuloksiin voitaisiin päästä laadukkaammalla ja isommalla aineistolla, tai häviöfunktiolla, joka soveltuu paremmin autovahinkoihin. Mallin yksi suurimmista eduista on sitä selittävä shap kuvaaja. Tästä kuvaajasta voitaisiin esimerkiksi määritellä tariffiluokkia muuttujien arvojen perusteella.

Elinajanodotteen ennustamisessa päästiin parempiin tuloksiin. Ennusteen keskineliövirhe oli 2,42 vuotta ja opetusvirhe 1,59 vuotta. Malliin eniten vaikuttavia muuttujia olivat aikuiskuolleisuus, tulot, sairaudet. Alkoholin vaikutus oli epäselvää, vaikka alkoholin ajateltiin olevan yksi suurimmista elinajanodotteeseen vaikuttavista tekijöistä. Tämän mallin kohdalla kyseessä on aikasarjan ennustaminen, joka onnistuu vielä kohtalaisesti. XGBoostilla voisi siis olla sovellusmahdollisuuksia aikasarja-analyysissa.

Suorituskykytestissä tutkittiin algoritmin tehokkuutta suurelle aineistolle. Approksimaatioalgoritmi oli hieman nopeampi kuin ahnealgoritmi tämän aineiston ennustamisessa. Kummallakaan algoritmilla ei kuitenkaan mennyt mallin opettamisessa minuuttia kauempaa. XGBoost -algoritmi soveltuu siis hyvin myös tavallisella tietokoneella tehtäväksi.

9 Liitteet

Python-koodit:

```
#Valitaan käytettävät paketit
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import xgboost as xgb
import shap
from eli5 import show_weights
from eli5 import show_prediction
import math

#Ajetaan data sisään
data = pd.read_csv("car_insurance_claim.csv", header=0)

#Muutetaan muuttujat dollareista liukuluvuiksi
data["INCOME"] = data["INCOME"].replace('[\$,]', '', regex=True)
.astype(float)
data["HOME_VAL"] = data["HOME_VAL"].replace('[\$,]', '', regex=True)
.astype(float)
data["BLUEBOOK"] = data["BLUEBOOK"].replace('[\$,]', '', regex=True)
.astype(float)
data["OLDCLAIM"] = data["OLDCLAIM"].replace('[\$,]', '', regex=True)
.astype(float)
data["CLM_AMT"] = data["CLM_AMT"].replace('[\$,]', '', regex=True)
.astype(float)

#Poistetaan poikkeavat havainnot
q = data["CLM_AMT"].quantile(0.95)
data = data[data["CLM_AMT"] < q]

#Valitaan mallin muuttujat
columns = ["BIRTH", "CLM_AMT", "ID", "CLAIM_FLAG"]
X = data.drop(columns, 1)
y = data["CLM_AMT"]

#Muokataan osaa muuttujista
X.loc[X["PARENT1"]=='No', "PARENT1"] = 0
X.loc[X["PARENT1"]=='Yes', "PARENT1"] = 1

X.loc[X["MSTATUS"]=='z_No', "MSTATUS"] = 0
X.loc[X["MSTATUS"]=='Yes', "MSTATUS"] = 1

X.loc[X["GENDER"]=='z_F', "GENDER"] = "Female"
X.loc[X["GENDER"]=='M', "GENDER"] = "Male"
```

```
X.loc[X["RED_CAR"]=='no', "RED_CAR"] = 0
X.loc[X["RED_CAR"]=='yes', "RED_CAR"] = 1

X.loc[X["REVOKED"]=='No', "REVOKED"] = 0
X.loc[X["REVOKED"]=='Yes', "REVOKED"] = 1

X.loc[X["EDUCATION"]=='<High School', "EDUCATION"] = "HighSchool"
X.loc[X["EDUCATION"]=='z_High School', "EDUCATION"] = "HighSchool"

#Tehdään opetus- ja testidatan jako
X = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(
        X, y, train_size=0.7, test_size=0.3, random_state=53,
         shuffle=False)

#Ajetaan data XGBoost DMatrix matriiseihin
xg_train = xgb.DMatrix(X_train, label=y_train, feature_names=X.columns)
xg_test = xgb.DMatrix(X_test, label=y_test, feature_names=X.columns)
xg_train.save_binary('train.buffer')
xg_test.save_binary('train.buffer')

#Parametrit
param = {}
param['objective'] = "reg:squarederror"
#param['tree_method'] = 'exact'
param['silent'] = 1
param["max_depth"] = 3
param['eta']=0.08
param["eval_metric"] = 'rmse'
watchlist = [(xg_train, 'train'), (xg_test, 'test')]
num_round = 200

#Opetetaan malli ja tehdään ennusteet
bst = xgb.train(param, xg_train, num_round, watchlist)
y_pred1 = bst.predict(xg_train)
y_pred2 = bst.predict(xg_test)

#RMSE
print(math.sqrt(1/len(y_train)*sum((y_train-y_pred1)**2)))
print(math.sqrt(1/len(y_test)*sum((y_test-y_pred2)**2)))
print(math.sqrt(1/len(y)*sum((y-y.mean())**2)))

#Muuttujan vaikutus malliin
ax = xgb.plot_importance(bst, max_num_features=25)
fig = ax.figure
```

```
fig.set_size_inches(15, 15)

#Painot
show_weights(bst)

#Shap
explainerXGB = shap.TreeExplainer(bst)
shap_values_XGB_test = explainerXGB.shap_values(X_test)
shap_values_XGB_train = explainerXGB.shap_values(X_train)

#Testidatan selitys
shap.summary_plot(shap_values_XGB_test, X_test, X.columns)

#Elinajanodote data
data = pd.read_csv("life Expectancy Data.csv", header=0)
data = data.dropna()

data.loc[data["Status"] == "Developing", "Status"] = 0
data.loc[data["Status"] != 0, "Status"] = 1

columns = ["Life expectancy "]
X = data.drop(columns, 1)
y = data["Life expectancy "]
X = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(
        X, y, train_size=0.6, test_size=0.4, random_state=42,
         shuffle=False)

xg_train = xgb.DMatrix(X_train, label=y_train, feature_names=X.columns)
 #feature_names=X_df.columns
xg_test = xgb.DMatrix(X_test, label=y_test, feature_names=X.columns)
#, feature_names=X_df.columns
xg_train.save_binary('train.buffer')
xg_test.save_binary('train.buffer')
param = {}
param['objective'] = "reg:squarederror"
param['tree_method'] = 'exact'
param['silent'] = 0 # cleans up the output
param["max_depth"] = 3
param['eta']=0.08
param["eval_metric"] = 'rmse'
watchlist = [(xg_train, 'train'), (xg_test, 'test')]
num_round = 100
bst = xgb.train(param, xg_train, num_round, watchlist)

y_pred1 = bst.predict(xg_train)
```

```
y_pred2 = bst.predict(xg_test)
y_pred2.mean()

ax = xgb.plot_importance(bst, max_num_features=25)
fig = ax.figure
fig.set_size_inches(15, 15)
show_weights(bst)

explainerXGB = shap.TreeExplainer(bst)
shap_values_XGB_test = explainerXGB.shap_values(X_test)
shap_values_XGB_train = explainerXGB.shap_values(X_train)

shap.summary_plot(shap_values_XGB_test, X_test, X.columns)


#Algoritmien tehokkuuden vertailu
data = pd.read_csv("brooklyn_sales_map.csv", header=0)
df = pd.DataFrame({"Exact": [0], "Approx": [0], "Number of rows": [0]})
for i in range(0,49):
    data1 = data.sample(n=(i+1)*10000, random_state=1, replace = True)

    columns = ["sale_date", "Unnamed: 0", "sale_price", "OwnerType",
    "Ext", "IrrLotCode", "HistDist", "Landmark", "ZoneMap", "ZMCode",
    "EDesigNum", "APPDate", "Version", "address", "apartment_number",
    "neighborhood", "building_class_category",
    "building_class_at_sale", "Borough", "FireComp", "SanitSub", "Address",
    "ZoneDist1", "ZoneDist2", "ZoneDist3", "ZoneDist4", "Overlay1",
    "Overlay2", "SPDist1", "SPDist2", "SPDist3", "LtdHeight", "SplitZone",
    "BldgClass", "OwnerName", "Sanborn"]
    X = data1.drop(columns, 1)
    y = data1["sale_price"]
    X = pd.get_dummies(X)
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, train_size=0.6, test_size=0.4, random_state=53,
        shuffle=False)


    start = timeit.default_timer()
    xg_train = xgb.DMatrix(X_train, label=y_train, feature_names=X.columns)
    xg_test = xgb.DMatrix(X_test, label=y_test, feature_names=X.columns)
    xg_train.save_binary('train.buffer')
    xg_test.save_binary('train.buffer')
    # setup parameters for xgboost
    param = {}
    param['objective'] = "reg:squarederror"
    param['tree_method'] = 'exact'
```

```
    param['silent'] = 0 # cleans up the output
    param["max_depth"] = 5
    param['eta']=0.08
    param["eval_metric"] = 'rmse'
    param['gamma'] = 0
    #param["scale_pos_weight"] = 0.12
    watchlist = [(xg_train, 'train'), (xg_test, 'test')]
    num_round = 50
    bst = xgb.train(param, xg_train, num_round, watchlist)

    stop = timeit.default_timer()
    execution_time = stop - start
    df.loc[i, 'Exact'] = execution_time

    start = timeit.default_timer()
    xg_train = xgb.DMatrix(X_train, label=y_train, feature_names=X.columns)
    xg_test = xgb.DMatrix(X_test, label=y_test, feature_names=X.columns)
    xg_train.save_binary('train.buffer')
    xg_test.save_binary('train.buffer')
    # setup parameters for xgboost
    param = {}
    param['objective'] = "reg:squarederror"
    param['tree_method'] = 'hist'
    param['silent'] = 0 # cleans up the output
    param["max_depth"] = 5
    param['eta']=0.08
    param["eval_metric"] = 'rmse'
    param['gamma'] = 0
    watchlist = [(xg_train, 'train'), (xg_test, 'test')]
    num_round = 50
    bst = xgb.train(param, xg_train, num_round, watchlist)

    stop = timeit.default_timer()
    execution_time = stop - start
    df.loc[i, 'Approx'] = execution_time

    df.loc[i, 'Number of rows'] = i*10000

    print(i)

ax = df.plot(x ='Number of rows', y=['Exact', "Approx"], kind = 'line',
figsize=(15, 9), title = 'Algoritmin suoritusaika eri havaintomäärillä')

df["approx%"] = df["Exact"]/df["Approx"]

ax = df.plot(x ='Number of rows', y="approx%", kind = 'line',
```

```
figsize=(15, 9),
title = 'Approksio algoritmin nopeus suhteessa ahneeseen')
```

Viitteet

[1] E. Alpaydin, *Introduction to machine learning*. MIT press, 2009.

[2] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[3] P. Gulati, A. Sharma, and M. Gupta, "Theoretical study of decision tree algorithms to identify pivotal factors for performance improvement: a review," *International Journal of Computer Applications*, vol. 975, p. 8887, 2016.

[4] A. Mohan, Z. Chen, and K. Weinberger, "Web-search ranking with initialized gradient boosted regression trees," in *Proceedings of the learning to rank challenge*, pp. 77–89, 2011.

[5] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, ACM, 2016.