Koneoppimismenetelmien vertailu lainan takaisinmaksun ennakoinnissa: Logistinen regressio, Random Forest ja Extreme Gradient Boosting

LuK Rasmus Harmanen

Pro gradu-tutkielma
Marraskuu 2025

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

Tarkastajat:
Prof. Ion Petre
Dos. Yury Nikulin

TURUN YLIOPISTO, Matematiikan ja tilastotieteen laitos

Pro gradu-tutkielma
Pääaine: Sovellettu Matematiikka
Tekijä: Rasmus Harmanen
Otsikko: Koneoppimismenetelmien vertailu lainan takaisinmaksun ennakoinnissa: Logistinen regressio, Random Forest ja Extreme Gradient Boosting
Ohjaaja: Prof. Ion Petre
Sivumäärä: 46 sivua
Aika: Marraskuu 2025

Tutkielman tavoitteena on vertailla kolmen yleisesti käytetyn koneoppimismallin logistinen regressiomalli, Random Forest -mallin ja Extreme Gradient Boosting (XGBoost) -mallin suorituskykyä ja tarkkuutta lainahakemuksista saatavien muuttujien vaikutukseen maksetaanko laina takaisin ajallaan vai ei. Tutkimuksen pohjautuu Yhdysvaltalaisen LendingClub-pankin vuosina 2007-2018 kerättyyn aineistoon, jossa on yli kaksi miljoonaa havaintoa. Aineisto on epätasapaoinoine, joten tuloksia arvioidaa seuraavilla mittareilla PR-AUC, ROC-AUC, F1 ja Brier Loss.

Tulokset osoittavat logistisen regressiomallin olevan paras tulkittavuudeltaan ja nopeudeltaan kouluttaa, mikä mahdollistaa läpinäkyvimmän pohjan lainapäätösten analyysiin. Random Forest pystyy hyvin yleistämään uusiin havaintoihin ja se pystyy oppimaan ei-lineaarisia yhteyksiä, mutta mallin tulkittavuus on heikompi. XGBoost-malli suoriutui PR-AUC ja ROC-AUC-mittareilla parhaiten, mutta mallin kalibrointitarve ja monimutkaisuus tekevät mallista vaativamman käytännön sovelluksissa.

Tutkielma vahvistaa käyttötarpeen määrittävän koneoppimismallin valintaa. Logistinen regressiomalli on parhaiten auditoitavissa ja XGBoost-malli tarjoaa parhaat tarkkuudet, jos halutaan parhaat ennusteet. Jatkotutkimuksena suositellaan vertailemaan syväoppimismenetelmiä aikasarja-aineistoissa.

Asiasanat: Koneoppiminen, Logistinen regressio, XGBoost, Random Forest, Luottoriski, LendingClub, Lainahakemus, Ennustemallit

Sisällys

1 Johdanto

Rahoitussektori on datan hyödyntämisen edelläkävijä. Laskentatehon kasvu on mahdollistanut entistä monimutkaisempien mallien kouluttamisen ja käyttämisen erityisesti riskienhallinnassa. Lainahakemuksien arvioinnin parantaminen voi mahdollistaa paremman kannattavuuden, vakavaraisuuden ja pienentää luottotappioita. Lainanmyöntäjille datan hyödyntäminen ja mallien soveltaminen on merkittävä kilpailuetu.

Koneoppimismallit mahdollistavat monimutkaisten ei-lineaarisia riippuvuuksien havaitsemisen ja mallintamisen. Tässä tutkielmassa tarkastellaan kolmea hyvin erilaista koneoppimismallia luokittelutehtävissä. Tutkielman tarkoituksena on selvittää kolmen yleisesti käytetyn koneoppimismallin logistisen regressio, Random Forest ja Extreme Gradient Boosting (XGBoost) eroavaisuudet ja tarvitaanko monimutkaisempia malleja, jotta voidaan saavuttaa parempi ennustetarkkuus kuin perinteisellä lineaarisella mallilla. Logistinen regressiomalli toimii edustaen perinteistä tilastollista mallia, kun taas XGBoost- ja Random Forest-malli edustavat päätöspuumallien ensemble-menetelmiä, joissa yhdistyy satunnaisuus ja iteratiivinen mallienparannus. Malleilla on tarkaituksena ennustaa lainan takaisinmaksuun liittyvää riskiä.

Tutkielmassa käytetään vuonna 2006 perustetun Yhdysvaltalaisen LendingClub pankin aineistoa vuosilta 2007-2018, jossa on 2 267 001 lainahakemusta. LengingClubin tarkoituksen oli tarjota lainoja yksityishenkilöltä yksityishenkilölle. Koska käytettävä aineisto on epätasapainoinen, luokittelumalleja arvioidaan testiaineistoon useilla mittareillä, kuten PR-AUC-, ROC-AUC-, Brier-Loss- ja F1-arvoilla.

Tutkielma rakentuu siten, että johdantoa seuraa aineiston esittelyyn ja aineiston esikäsittelyyn. Tämän jälkeen jokainen luokittelumalli määritetään ja malleja kouluttaessa käytettävät hyperparametrit esitetään. Mallien määrittämisen jälkeen mallien arviointi mittarit määritetään. Seuraavaksi esitellään mallien baseline-tulokset, josta seuraa mallien kouluttaminen, kalibrointi ja mallien keskenäinen vertailu. Lopuksi on esitetty yhteenveto ja loppupäätelmät.

Tutkielman valmistelussa on hyödynnetty avustavaa kielimallia (ChatGPT) lähinnä kielenhuoltoon ja koodiesimerkkien tarkennuksiin.

2 Aineisto

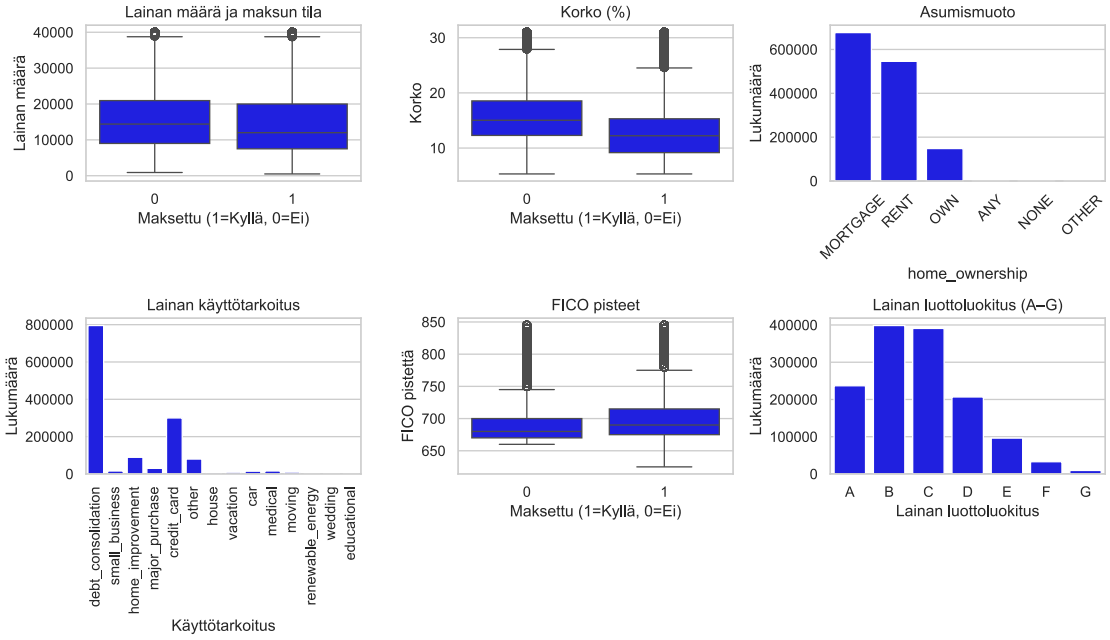
Vuonna 2006 Yhdysvalloissa perustettu digitaalisten lainojen välittäjä (Fintech-yritys) LendingClub tarjoaa yksityisasiakkaille mahdollisuuden hakea lainaa yksityisiltä sijoittajilta ilman perinteistä pankkia välikätenä. LendingClubin tarkoituksena on tarjota mahdollisimman helppo ja yksinkertainen lainanhaku- ja lainanantopalvelu ilman pankin luomaa kitkaa. Yksityishenkilöt pystyvät hakemaan kulutustai yhdistelmälainoja, joita yksityishenkilöt voivat rahoittaa osittain. LendingClub itse arvioi lainahakemuksien perusteella lainojen luottoriskiä lainanhakijoiden luottoluokituksen perusteella. Vuonna 2013 LendingClub ryhtyi tarjoamaan avointa aineistoa osana heidän läpinäkyvyysstrategiaansa. Radius Bank osti LendingClubin vuonna 2020 minkä takia vertaislainapalvelu päättyi eikä vanhoja aineistoja ole sen jälkeen päivitetty. Tässä tutkielmassa käytetään LendingClubin alustalta kerättyä anonymisoitua aineistoa, joka on lainanhakijoiden tekemästä hakemuksesta sijoittajille. Aineistossa on miljoonia lainahakemuksia ja se sisältää lainanhakijoiden taustatietoja, kuten aikaisempien lainojen takaisinmaksuhistorian, luottoluokituksen ja tiedot haettavan sekä jo olevien lainojen ominaisuuksista.

Tutkielman aineistona toimii LendingClubin hyväksyttyjen lainahakemuksien aineistoa vuodesta 2007 vuoteen 2018, joka on ladattu Kagglesta ja on siellä kaikkien saatavilla. Kaggle tarjoaa aineistoja oppimis- ja tutkimuskäyttöön. LendingClubin data on harvinaislaatuinen, koska yleisesti yritykset suojelevat omia aineistojaan ja myöskin niistä tehtyjä päätelmiä. Tämän takia LendingClub ei ole enää päivittänyt avoimia aineistojaan lainoistaan. Tämä kyseinen versio aineistosta on hyvin siivottu ja standardisoitu jo valmiiksi. LendingClubin aineistoa on hyödynnetty useissa rahoituksen ja matemaattisen mallinnuksen tutkielmissa, koska se on rakeenteeltaan ainutlaatuinen ja helposti saatavilla. Näin ollen tästä samasta aineistosta tehdyt tutkimukset ovat hyvin vertailukelpoisia keskenään.

Kagglesta ladatussa CSV-aineistossa on kymmenessä vuodessa kertynyt 2 267 001 hyväksyttyä lainahakemusta, joissa on 151 muuttujaa. Näistä 113 muuttujaa on numeerisia ja 38 on tekstimuotoisia. Aineisto jakautuu lainanhakijoiden itse antamiin tietoihin ennen lainatarjouksen hyväksymistä ja LendingClubin määrittelemiin muuttujiin lainahakemuksen hyväksymisen jälkeen. Aineistosta on etukäteen jo poistettu kaikki henkilötietoja sisältävät muuttujat, joten aineistoa on tieto- ja yksityisyyden suojan puolesta turvallista käsitellä. LendingClubin aineisto sopii tutkimuskysymykseeni, koska aineistossa on suuri määrä muuttujia lainanhakijoiden taustoista ja sosioekonomisesta asemasta.

Kuvassa 1 on koulutettavan aineiston keskeisimmistä muuttujista tehdyt kuvaajat. Maksettujen lainojen- ja maksamattomien lainojen luokissa mediaani asettuu noin 15 000 ja 20 000 dollarin tasolle. Myöskin molemmissa luokissa kuvaajan viikset ulottuvat 40 000 dollariin asti. Näin ollen lainan suuruudella ei vaikuttaisi olevan yksisteen merkitystä maksetaanko laina takaisin ajallaan vai ei.

Lainojen korot vaihtelevat yllättävän paljon, ollen 5% ja 25% välillä. Erityisesti juuri maksamattomien ja maksettujen lainojen korkojen erot ovat selvät, maksamattomien lainojen korkojen ollessa suuremmat. Tämä haijastaa lainanhakijoiden vaihtelevaa riskiprofiilia, kun suureman riskin lainanhakijat saavat suurempi korkoisia lainatarjouksia.



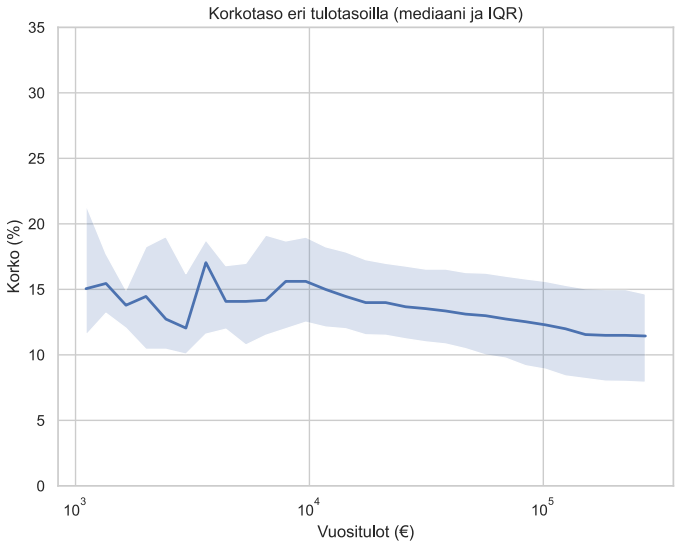
Kuva 1: Aineiston keskeisimmät muuttujat ja niiden yhteydet lainantakaisin maksamiseen.

Suurin osa lainanhakijoista on asuntovelallisia, toisiksi suurin osa asuu vuokralla ja kolmantena on omistusasunnossa asuvat. Tämä jakauma on hyvin yleinen keskiluokkaisten keskuudessa. Suurin osa lainoista on haettu yhdistääkseen useampia pienempiä lainoja. Seuraavana listalla on luottokorttilainan takaisin maksu, joka on myös yhden tai useamman korkea korkoisen lainan yhdistämistä. Kolmanneksi suurin on kodin parantaminen. Koska useimmat lainanhakemisen syistä ovat vanhojen jo voimassa olevien lainojen helpottamista varten, voidaan lainanhakijoita pitää hyvin korkean riskin lainanhakijoina.

FICO pisteistä lainan takaisin maksaneiden ja ei maksaneiden välillä oleva ero on selvästi havaittavissa. Näin ollen luottopistejärjestelmän käyttökelpoisuus on selvästi merkitsevässä roolissa.

Viimeisessä kuvassa esitetään haettavien lainojen luottoluokitus (A-G). Suurin osa lainoista on luokiteltu B ja C lainoiksi ja arvoa D huonompia lainoja on aineistossa huomattavasti vähemmän, joka on linjassa muiden muuttujien kanssa. Nämä lainojen luokitukset ovat linjassa muiden aineiston kuvaajien kanssa.

Lähtökohtaisesti suuremmilla vuosituloilla saa suurempia lainoja. Lainalle myönnettävä korko on viite lainaan sijoittamisen riskistä. Kuvaajassa 2 on esitetty korkotason muutos eri vuositulojen mediaaneilla. Kuvaajassa tulot kasvavat oikealle x-akselia pitkin ja korko prosentti esitetään y-akselilla. Kuvaajasta nähdään, että yli 10 000 dollarin vuosituloilla koron suuruus on laskeva. Kun taas alle 10 000 dollarin vuosituloilla korot vaihtelevat todella suuresti 11-18% välillä.



Kuva 2: Vuositulojen vaikutukset lainan korkotasoon.

2.1 Datan esikäsittely

Datan esikäsittelyllä tarkoitetaan datan saattamista valmiiksi sen analysoimista varten. Tämä vaihe on erittäin tärkeä koneoppimismallien kouluttamisessa, sillä se vaikuttaa merkittävästi malleista saatavaan analyyseihin [1]. Datan esikäsittely voidaan jakaa karkeasti kolmeen vaiheeseen. Ensimmäisessä vaiheessa kategoriset muuttujat muunnetaan mallille sopivaan muotoon. Toisessa vaiheessa päätetään, mitä mahdollisesti puuttuville arvoille tehdään, ja viimeisessä vaiheessa muuttujien arvot skaalataan malleille sopiviksi.

Ennen muita datan esikäsittelyvaiheita aineistoon luotiin uusi päätösmuuttuja *loan_prepaid*. Päätösmuuttuja saa arvon 0, jos laina ei ole maksettu takaisin ajallaan tai lainanmaksamisessa on ongelmia, ja arvon 1, jos laina on maksettu ajallaan onnistuneesti. Muuttuja *loan_prepaid* johdettiin alkuperäisestä muuttujasta *loan_status*, jolla on kahdeksan mahdollista arvoa *Fully Paid, Charged Off, Default, In Grace Period, Late (31-120 days), Late (16-30 days), Does not meet the credit policy* ja *Current*. Mikäli *loan_status* on *Fully Paid*, saa *loan_prepaid* arvon 1. Kaikki lainat poistettiin, joiden *loan_status* on *Current*, koska näiden osalta takaisinmaksusta ei ole vielä tietoa. Loput tilat saivat arvon 0, koska tilat *Late, Default, Charged Off* ja *In Grace Period* viittaavat maksuongelmiin.

Muuttujalla *loan_status* on lisäksi kaksi erikoisarvoa: "Does not meet the credit policy. Status: Fully Paid" ja "Does not meet the credit policy. Status: Charged Off". Kumpikaan näistä arvoista ei täyttänyt lainan myöntämiskriteerejä, mutta myönnettiin silti. Nämä molemmat arvot poistettiin aineistosta, koska tarkoituksena oli tutkia lainan myöntämisen vähimmäiskriteerit täyttävien lainanhakijoita, mutta esiintyi maksuongelmia. Näin saatu binäärinen päätösmuuttujan *loan_prepaid* saa arvon 1, jolloin laina maksettiin ajallaan takaisin $78,5\%$ tapauksista ja arvon 0, jolloin lainan takaisinmaksun kanssa ilmeni ongelmia $21,5\%$. Koneoppimismalleja kouluttaessa muuttuja y käännetään ympäri $y \leftarrow y - 1$, jolloin positiivinen luokka

1 on kaikki, joiden lainojen maksamisessa havaittiin ongelmia. Tämä kääntäminen tehdään, koska tarkoituksena on mallintaa positiivista luokkaa eli lainan takaisinmaksun kanssa ilmenneitä ongelmia. Tämä on jakauma ei ole äärimmäisen vino, mutta tulee ottaa huomioon myöhemmin malleja koulutettaessa.

Alkuperäisistä 151 muuttujasta poistettiin ensin tietovuotoriskin sisältävät muuttujat, jotka LendingClub on kirjannut vasta lainan myöntämisen jälkeen, kuten *total_pymnt*. Tällaiset muuttujat häiritsisivät mallien koulutusta ja vääristäisivät malleja. Tämän jälkeen poistettiin kaikki muuttujat, jotka olivat vakioita koko aineistossa kuten *policy_code*, *pymnt_plan* ja *application_type*.

Aineistosta poistettiin myös kaikki muuttujat, joiden arvoista yli 50% puuttui, koska tällaiset muuttujat eivät tuota luotettavaa lisäarvoa mallille. Tämän jälkeen muuttujia jäi jäljelle 93 kappaletta. Jäljelle jääneistä muuttujista poistettiin vielä sellaiset, jotka sisältävät pelkkää tekstiä eikä niitä voitu helposti kategorisoida kuten ammatti *emp_title*.

Muuttujat *grade* ja *subgrade* jätettiin aineistoon, koska ne ovat LendingClubin itse antamia luottoluokituksia hakemuksille. Luokitukset tulevat heti lainahakemuksen yhteydessä eikä niihin ole päässyt vaikuttamaan vielä yksikään maksuerä tai muu tapahtuma. Näin ollen voidaan luokittelumalleissa huomioida niin sanottu sisäisenluokituksen merkitys myöskin. Mikäli tutkielmassa arvioitaisiin yleistä lainojenmaksuihin liittyvää riskiä, olisi perusteltua poistaa kyseiset muuttujat.

Näiden rajauksien jälkeen aineistoon jäi 74 muuttujaa, jotka kuvaavat lainanhakijoiden taustatietoja ja lainaehtoja. Jäljelle jääneet muuttujat sisältävät muun muassa hakijoiden työhistoriaa, lainojen yhteismäärää, luottoluokitusta, tulotietoja ja asumismuotoa. Muuttuja *emp_length* muunnettiin numeeriseen muotoon vuosiksi ja puuttuvat arvot täydennettiin mediaaniarvolla.

2.1.1 Puuttuvat arvot

Puuttuvien arvojen ongelma on yleinen kaikissa aineistoissa, joissa käsitellään dataa. Puuttuvat arvot voivat johtaa vääristymiin ja siten huonoihin koneoppimismallien tuloksiin sekä virheellisiin päätelmiin aineiston pohjalta [2]. Arvojen puuttumisen vakavuus ja laajuus riippuvat siitä, kuinka paljon arvoja puuttuu ja mitkä ovat niiden taustalla olevat syyt [3].

Seuraavien mekanismien määrittämisessä on käytetty lähteitä [3] ja [4]. Yleisesti erotetaan kolme erilaista mekanismia puuttuville arvoille. Ensimmäinen mekanismi on, että arvoja puuttuu täysin sattumanvaraisesti (*Missing Completely At Random*). Tällöin puuttuvat arvot eivät ole mitenkään riippuvaisia muuttujista tai havainnoista. Tämä on paras mahdollinen tilanne, sillä puuttuvat arvot eivät vääristä aineistoa.

Toinen mekanismi on arvojen puuttuminen siten, että se voidaan selittää jollain toisella muuttujalla (*Missing At Random*). Esimerkiksi jos muuttuja *annual_income* sisältää puuttuvia arvoja, näitä voidaan selittää *emp_title* muuttujalla. Tietyt ihmiset voivat esimerkiksi jättää ilmoittamatta vuositulonsa, mutta heidän ammattinsa antaa viitteen vuosituloista. Tämä voi esiintyä esimerkiksi matalapalkka-aloilla. Tällöin puuttuvat arvot voidaan täyttää mallintamalla muiden muuttujien avulla, mikä mahdollistaa analyysin jatkamisen ilman merkittävää vääristymistä.

Kolmantena ja hankalimpana mekanismina on arvojen puuttumisen johtuminen juuri kyseisestä arvosta itsestään (*Missing Not At Random*). Esimerkiksi jos *an-*

nual_ income arvoja puuttuu juuri niiltä, joilla tulot ovat poikkeuksellisen pienet. Tämä on hankala tilanne, sillä puuttumisen syy liittyy suoraan mitattavaan ilmiöön. Tällöin harhattoman aineiston saamiseksi täytyy mallintaa puuttuvat arvot jollakin muulla tavalla kuin arvojen täyttämisellä, kuten k-lähimmät naapureiden menetelmällä tai k-means, jotta saataisiin mahdollisimman harhaton aineisto. Lopputulos jää väistämättä epävarmemmaksi.

Puuttuvia arvoja arvioitaessa saadaan lopputulema, että suurin osa puuttuvista arvoista johtuu siitä, kun maksuaikajärjestelyä ei ole syntynyt. Tämä selittää suurimman osan *hardship-* ja *settlement-* muuttujien puuttumisen. Tämä viittaa *missing not at random* -tilanteeseen, kun maksuaikaa tai erityisjärjestelyjä ei ole tarvittu. Muuttujassa *emp_ length* havaittiin myös puuttuvia arvoja. Tämä näkyi erityisesti tietyissä hakijaryhmissä, jolloin tilanne olisi *missing at random*. Muuttuja postinumerosta sisälsi vain harvoja puuttuvia arvoja, eikä loogisuutta puuttuvien arvojen osalta löytynyt, joten tässä arvioitiin tilanteen olevan *missing completely at random*.

Tämän jälkeen aineisto jaettiin selitettävään muutujaan y ja selittäviin muuttujiin X. Aineisto jaettiin ajallisesti harjoitusaineistoon (2007-2015), jossa on 828 654 havaintoa ja arvon 1 saa $18,6\%$ havainnoista. Validointiaineistossa (2016-2017) on 477 370 havaintoa, joista arvoja 1 on $25,6\%$ ja testiaineistossa on 65 142 havaintoa, joiden arvo on 1 $27,2\%$ tapauksista. Näin ollen arvon 1 todennäköisyys on kasvanut aineiston seuranta jaksolla. Tällöin ajallisen jaon merkitys on suuri malleja arvioitaessa.

2.1.2 Kategoriset muuttujat

Kategoristen muuttujien muuntaminen numeeriseen muotoon on kriittinen vaihe datan esikäsittelyssä, sillä koneoppimismallit tarvitsevat erilailla käsitellyt kategoriset muuttujat. Tästä syystä aineistosta tehdään kaksi erilaista versiota malleja varten.

Lineaariset mallit tarvitsevat yksiluokkaista binääriesitystä (one-hot encoding) kategorisille muuttujille. Yksiluokkainen binääriesitys toimii siten, että jokainen muuttujan kategoria saa oman binäärisen arvon 1 tai 0, joka ilmaisee luokkaan kuulumisen. Menetelmän etuna on erityinen selkeys ja tarkkuus sekä se, ettei se perustu järjestysasteikkoon. Esimerkiksi, jos kategorisena muuttujana on koulutusaste, jossa on vaihtoehtoina peruskoulu, lukio tai ammattikoulu, luodaan kolme uutta saraketta, joista jokainen saa arvon 1, mikäli lainanhakija kuuluu kyseiseen kategoriaan.

	Peruskoulu	Lukio	Yliopisto
Lainanhakija 1	1	0	0
Lainanhakija 2	0	1	0
Lainanhakija 3	0	0	1

Taulukko 1: Yksiluokkainen binääriesitys.

Vaikka lineaarisille malleille yksiluokkainen binääriesitys on suotavaa, se ei yleensä ole optimaalisin menetelmä muihin koneoppimismalleihin, kuten puumallien, kannalta. Tämä johtuu siitä, että sarakkeiden lukumäärää kasvaa nopeasti kategorioiden lisääntyessä, mikä kasvattaa muistin käyttöä ja mallien monimutkaisuutta mer-

kittävästi. Tämä puolestaan lisää laskentatehon tarvetta sekä mallin koulutusaika pitenee [5]. Tutkielmassa tällä tavalla muunnetaan kategoriset muuttujat logistiselle regressioanalyysimallille.

Toinen yleinen tapa käsitellä kategorisia muuttujia on laskea kullekin kategorialle päätösmuuttujan keskiarvo (target encoding). Tätä menetelmää suositellaan erityisesti puumalleille. Menetelmää sovelletaan vain harjoitusdataan, sillä mallin kouluttaminen päätösmuuttujasta johdetuilla arvoilla aiheuttaisi tietovuodon ja johtaisi ylioppimiseen. Harjoitusvaiheessa koulutetaan koneoppimismallia, koska menetelmä hyödyntää tietoa päätösmuuttujasta, mikä on aina suuri riski tietovuodolle, joka yliopettaisi luokittelumallin. Target-enkoodauksessa jokaiselle kategoriselle muuttujalle lasketaan päätösmuuttujan keskiarvo [2].

	Koulutustaso	*loan_prepaid*
Lainanhakija 1	Lukio	0
Lainanhakija 2	Yliopisto	0
Lainanhakija 3	Lukio	1
Lainanhakija 4	Yliopisto	1
Lainanhakija 5	Yliopisto	1

Taulukko 2: Päätösmuuttujan mukaan muuntaminen lähtötilanne.

Kuvitellaan taulukon 2 mukainen tilanne. Tästä saadaan laskettua keskiarvot. Lukio saa keskiarvoksi $\frac{0+1}{2} = 0.5$ ja yliopisto saa keskiarvoksi $\frac{0+1+1}{3} = \frac{2}{3}$. Kun alkuperäiset kategoriat korvataan saaduilla keskiarvoilla, saadaan puumalleille sopivat muuttujat.

	Koulutustaso	*loan_prepaid*
Lainanhakija 1	0.5	0
Lainanhakija 2	1	0
Lainanhakija 3	0.5	1
Lainanhakija 4	1	1
Lainanhakija 5	1	1

Taulukko 3: Päätösmuuttujan mukaan muuntaminen lopputilanne.

Kuten taulukossa 3 huomataan, mallit saavat tästä jatkuvat muuttujat. Haasteena target-enkoodaamisessa on, että jos harjoitusaineistossa jokin kategoria esiintyy vain kerran, mallilla ei ole mahdollisuutta oppia sen käyttäytymistä luotettavasti. Esimerkiksi jos yksi kategoriaryhmä on maksanut onnistuneesti lainansa pois, tällöin testidatassa malli käyttää tätä ryhmää niin sanottuna päätösmuuttujana. Ongelma voidaan ratkaista käyttämällä ristiinvalidointia (cross-validation), jossa keskiarvot lasketaan ainoastaan harjoitusaineistossa olevien havaintojen perusteella, ja koulutusaineiston ulkopuolisille kategorioille annetaan yhtenäiset arvot [6]. Mikäli testiaineistoon ilmestyy kategoria, jota ei harjoitusaineistossa ollut, tällöin sille annetaan arvo Nan eli puuttuva arvo. Kaikki ei arvoa sisältävät muuttujat käsitellään seuraavassa vaiheessa.

Puumalleille esikäsiteltäessä kategorisia muuttujia voidaan käyttää oridinaalenkoodausta. Kun kategoriset muuttujat voivat olla nominaaliset, joilla ei ole si-

säistä järjestystä tai ordinaalisia, joilla on selvä sisäinen järjestys. Label encoding muuttaa kategoriat kirjallisesta muodosta numeeriseen muotoon [7]. Esimerkiksi lukio=0 ja yliopisto=1. Tämä onnistuu erityisen hyvin puumalleille, koska ne eivät ajattele numeroita paremmuusjärjestyksessä, vaan käyttävät numeroita ainoastaan haarojen jakamisessa.

Logistiselle Regressiomallille ja Random Forestille käytettiin one-hot-koodausta ja Extreme Gradient Boosting-mallille käytettiin oridinali-enkoodausta.

2.1.3 Arvojen skaalaus

Arvojen skaalauksella tarkoitetaan muuttujien arvojen muuntamista vertailukelpoiselle välille. Esimerkiksi vuositulojen muuttujan *annual_ income* arvot vaihtelevat välillä 0$ - 10 999 200 dollaria, kun taas muuttujan *num_ tl_ op_ past_ 12m*, joka kertoo montako avointa luottosuhdetta lainanhakijalla on, arvot vaihtelevat 0:n ja 32:n välillä. Tällöin muuttujien arvot voidaan skaalata välille 0-1 siten, että niiden keskiarvo on 0 ja hajonta 1.

Skaalauksen merkitys riippuu koneoppimismallista. Esimerkiksi puumallit eivät yleensä tarvitse arvojen skaalausta, mutta lineaariset mallit sitä vastoin vaativat [8]. Lisäksi eri malleille sopivat paremmin erilaiset skaalamenetelmät. Logistisessa Regressiossa skaalauksella voi olla merkittävä vaikutus, kun verrataan z-standardoinnilla skaalatuilla muuttujilla saadaan huomattavasti parempia tuloksia verrattuna skaalaamattomiin arvoihin [8]. Z-standardoinnissa jokaisen muuttujan arvo muunnetaan kaavalla

$$x = \frac{x - \mu}{\sigma},$$

missä x on alkuperäinen arvo, μ on muuttujan arvojen keskiarvo ja σ on muuttujan arvojen keskihajonta. Tällöin saadaan muuttujalle keskiarvoksi 0 ja keskihajonnaksi 1.

Arvojen skaalaus estää myös yksittäisiä muuttujia hallitsemasta mallin koulutusta liikaa. Tämä on erityisen tärkeää lineaarisissa malleissa, joissa suuret arvovälit voivat vääristää tuloksia [8]. Aineistoa esikäsiteltäessä Logistiselle Regressiomallille käytettiin tätä menetelmää. Random Forest-ja Extreme Gradien Boosting-mallit eivät tarvitse skaalausta etukäteen.

3 Menetelmät

Koneoppimismalleja on lukuisia erilaisia, ja ne voidaan luokitella oppimistyylin, malliperheen tai päätösluonteen mukaan. Oppimistyylin perusteella mallit jaetaan valvottuun oppimiseen, valvomattomaan oppimiseen, puolivalvottuun oppimiseen sekä vahvistusoppimiseen. Valvotussa oppimisessa malli oppii aineiston syötteistä ja niihin liittyvistä tunnetuista lopputuloksista. Valvomattomassa oppimisessa malli etsii rakenteita tai klustereita aineistosta ilman etukäteen määriteltyjä lopputuloksia ja tekee omat päätelmänsä näiden perusteella luokkajaosta. Puolivalvottu oppiminen on näiden edellä mainittujen välimalli, jossa osa aineiston lopputuloksista jo tunnetaan ja osa ei. Vahvistusoppimisessa malli oppii palkkioiden ja rangaistusten avulla arvioimaan omaa toimintaansa ja tätä kautta oppii aineistosta.

Malliperheiden mukaan koneoppimismallit voidaan jakaa lineaarisiin malleihin, puu- ja enseblemalleihin, probabilistisiin malleihin, neuroverkkoihin sekä dimensioiden pienennysmenetelmiin.

Koneopppimismallit voidaan päätösluonteen perusteella jakaa luokittelumalleihin, regressiomalleihin, klusterointimalleihin, dimensioiden vähentämismalleihin tai anomalianhavaitsemismalleihin.

Tämän tutkielman tavoitteena on kehittää mahdollisimman hyvä luokittelumalli, joka ennustaa lainanhakijan maksujen toteutumista lainan maksun aikana hakija ominaisuuksiensa perusteella. Luokittelumallin tarkoituksena on arvioida, maksaako asiakas lainansa takaisin ongelmitta vai ei. Päätösmuuttuja *loan_prepaid* saa arvon 0, jos laina maksetaan takaisin ajallaan ilman ongelmia, ja arvon 1, jos lainan maksussa esiintyy ongelmia tai se jää maksamatta.

Jokaisen lainanmaksajan kohdalla tehtävässä luokittelussa on ongelmana, mikä todennäköisyys riittää siihen, että laina voidaan katsoa maksetuksi. Koneoppimismalli laskee jokaiselle lainanhakijalle todennäköisyyden $P(y = 1|x)$, jolla kuvataan ongelmien ilmaantumista lainan takaisinmaksun aikana. Yksi keskeisimpiä asioita koneoppimismallille on löytää todennäköisyys, jonka jälkeen laina luokitellaan maksetuksi. Tätä arvoa kutsutaan kynnysarvoksi. Esimerkiksi, jos kynnysarvo on 0,7 ja saatu todennäköisyys olisi alle tämän, laina luokitellaan maksamattomaksi, kun taas yli laina luokitellaan maksetuksi.

3.1 Logistinen regressiomalli

Logistinen regressiomalli on valvotun oppimisen malli, joka kuuluu lineaaristen mallien perheeseen. Tässä tutkielmassa sitä käytetään luokittelumallina, joka hyödyntää todennäköisyyspohjaista lähestymistapaa. Maksetaanko laina takaisin vai ei on binäärinen luokitteluongelma, jonka ratkaisemiseen logistinen regressiomalli soveltuu hyvin [9].

Logistisen regressiomallin taustalla oleva logistinen funktio kehitettiin jo 1800-luvun puolivälissä mallintamaan populaatioiden kasvua ja kuvaamaan kemiallisia reaktioita [10]. Varsinainen logistinen regressio kehitettiin 1940-luvun loppupuolella, jolloin sitä alettiin käyttää erityisesti tilastotieteessä binääristen luokitteluongelmien ratkaisemiseen. Viimeisten vuosikymmenten aikana logistista regressiota on sovellettu laajasti sekä tilastotieteessä että koneoppimisessa, ja se on osoittautunut

erittäin tehokkaaksi malliksi monilla sovellusalueilla [11].

Lineaarinen regressioanalyysi on tilastollinen menetelmä, jonka tavoitteena on sovittaa selittävien muuttujien X ja selitettävän muuttujan Y välille mahdollisimman hyvin sopiva suora. Mallin perusmuoto on

$$Y = X\beta + \epsilon,$$

missä $Y \in \mathbb{R}$, β on parametrivektori ja ϵ on mallin virhetermi, jonka oletetaan olevan normaalijakautunut [12]. Logistisessa regressioanalyysissä lineaarista mallia sovelletaan siten, että lopputulos tulkitaan todennäköisyydeksi kuulua tiettyyn luokkaan [13].

Logistisen regressioanalyysin päätavoitteena on löytää funktio, joka muuntaa syötemuuttujien lineaarisen yhdistelmän todennäköisyydeksi [14]. Tämä toteutetaan sigmoidisella eli logistisella funktiolla, jonka muoto on

$$p(X) = \frac{1}{1 + e^{(-X\beta)}},$$

missä $p(X)$ on todennäköisyys sille, että havainto kuuluu luokkaan $y = 1$. Logistisen funktion arvo on aina välillä $(0, 1)$, minkä vuoksi se on luonnollinen tapa mallintaa todennäköisyyksiä [14].

3.1.1 Määritelmä

Seuraavassa määritelmässä noudatetaan lähteitä [14] ja [15] logistisen regressioanalyysifunktion mallintamiseen.

Olkoon jokaiselle havainnolle eli lainalle $i \in \{1, 2, 3, ..., n\}$ ominaisuusvektori $\mathbf{x_i^T} \in \mathbb{R}^m$ ja jokaiselle lainalle olemassa $Y_i \in \{0, 1\}$ päätösmuuttuja, joka saa arvon 1, jos laina maksetaan ajallaan ja $Y_i = 0$ muuten.

Näin ollen saadaan aineisto $D = \{\{x_i, y_i\}\}_{1 \leq i \leq n}$, joka voidaan esittää matriisimuodossa

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^{n \cdot 1}, \ \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & ... & x_{1m} \\ x_{21} & x_{22} & ... & x_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & ... & x_{mn} \end{bmatrix} = \begin{bmatrix} x_{[1]}^T \\ x_{[2]}^T \\ \vdots \\ x_{[n]}^T \end{bmatrix} \in \mathbb{R}^{n \cdot m}.$$

Jokainen $\mathbf{x}_{[i]}$ rivi esittää yhden lainanhakijan ominaisuuksia, esimerkiksi $\mathbf{x}_{i1}$ voisi olla lainanhakijan vuositulot ja $\mathbf{x}_{i2}$ lainanhakijan FICO-luottoluokitus. Tällä matriisirakenteella voidaan käyttää logistista regressiota, jonka tarkoituksena on määrittää jokaisen lainan takaisinmaksun todennäköisyys.

Päätösmuuttuja Y_i noudattaa Bernoulli-jakaumaa logistisessa regressiossa ominaisuusvektorin x_i perusteella

$$Y_i | x_i \sim Bernoulli(p_i), \quad p_i = P(Y_i = 1 | x_i),$$

missä p_i kuuluu lineaariseen yhdistelmään $x_i^T \beta$ logit-linkin

$$log(\frac{p_i}{1 - p_i}) = x_i^T \beta$$

kautta, jossa todennäköisyys määritetään $\Theta(x_i) = \frac{p_i}{1-p_i}$.

Logistisessa regressiossa teoreettisesti oletetaan päätösmuuttujan $\mathbf{Y} \in (0, 1)$ noudattavan ehdollisesti binomijakaumaa jokaisen yksittäisen lainan takaisinmaksun kohdalla. Tällöin koko aineiston mallintamisessa voidaan käyttää binomijakaumaa. Kun lainan takaisinmaksuun liittyvät muuttujat ovat annettuna ominaisuusvektorissa $\mathbf{X}$, ehdollista todennäköisyyttä lainan takaisinmaksusta merkitään $P(\mathbf{Y} = 1|\mathbf{X})$. Mikäli lainaa ei makseta ajallaan takaisin, merkitään $P(\mathbf{Y} = 0|\mathbf{X})$. Tällöin takaisinmaksun eli onnistuneen ja epäonnistuneen takaisinmaksun välistä suhdetta annetulla ominaisuusvektorilla määritetään kaavalla

$$\Theta(\mathbf{X}) = \frac{P(\mathbf{Y} = 1|\mathbf{X})}{P(\mathbf{Y} = 0|\mathbf{X})} = \frac{P(\mathbf{Y} = 1|\mathbf{X})}{1 - P(\mathbf{Y} = 1|\mathbf{X})}.$$

Funktion $p(\mathbf{X})$ arvot kuuluvat välille $(0, 1)$, koska sigmoidifunktion arvot lähestyvät 0 tai 1, mutta eivät koskaan saavuta niitä. Tämän takia funktion $\Theta(\mathbf{X})$ arvot kuuluvat välille $(0, \infty)$. Tällöin funktion $ln(\Theta(\mathbf{X}))$ arvot taas kuuluvat välille $(-\infty, \infty)$. Edellisen funktion muunnos antaa mallille seuraavan lineaarisen muodon

$$ln(\Theta(\mathbf{X})) = ln(\frac{p(\beta, \mathbf{X})}{1 - p(\beta, \mathbf{X})}) = \beta\mathbf{X}.$$

β on mallin parametrivektori, joka kertoo paljonko muuttujavektorin alkio $\mathbf{x_i}$ vaikuttaa logit-muotoiseen todennäköisyyteen. Ratkaisemalla edellisestä kohdasta $p(\beta, \mathbf{X})$, saadaan

$$p(\beta, \mathbf{X}) = \frac{e^{(\mathbf{X}\beta)}}{1 + e^{(\mathbf{X}\beta)}}.$$

Tämä on aikaisemmin mainittu sigmoidifunktio, joka rajoittaa mallin antamat todennäköisyydet välille $(0, 1)$. Seuraavaksi ratkaistaan parametrivektorin β kertoimet kaavalla

$$\max_{\beta} L(\beta, \mathbf{Y}, \mathbf{X}), \tag{1}$$

jossa L on uskottavuusfunktio. Kuten aikaisemmin sanottiin, jokainen havainto on Bernoulli-jakautunut ja ehdollisesti riippumaton. Näin ollen saadaan uskottavuusfunktio

$$L(\beta, \mathbf{Y}, \mathbf{X}) = \prod_{i=1}^{n} p(\beta, \mathbf{x}_{(i)})^{y_i}(1 - p(\beta, \mathbf{x}_{(i)}))^{1-y_i}.$$

Logaritmin laskeminen on mielekkäämpää kuin juuri esitetyn yhtälön ratkaiseminen, koska logaritmi muuntaa tulolausekkeet summiksi. Tällöin voidaan ratkaista yhtälön 1 sijasta yhtälöä

$$\max_{\beta} l(\beta, \mathbf{Y}, \mathbf{X}), \tag{2}$$

jonka uskottavuusfunktion on

$$l(\beta, \mathbf{Y}, \mathbf{X}) = \sum_{i=1}^{n} (y_i \mathbf{x}_{(i)} \beta - ln(1 + e^{(\mathbf{x}_{(i)}\beta)}). \tag{3}$$

Tuntemattomien parametrien β ratkaisemiseksi käytetään maksimoinnissa Newton-Raphson-algoritmia. Algoritmin käyttäminen perustuu parametrien β ratkaisemiseen iteratiivisin askelin. Askeleen $j+1$ parametrit saadaan kaavasta

$$\beta_{j+1} = \beta_j + (\frac{\partial^2 l}{\partial \beta \partial^T}(\beta_j))^{-1} \frac{\partial l}{\partial \beta}(\beta_j), \tag{4}$$

jossa $\frac{\partial l}{\partial \beta}$, $\frac{\partial^2 l}{\partial \beta \partial \beta^T}(\beta_j)$ ovat uskottavuusfunktion l ensimmäinen ja toinen osittaisderivaatta.

3.1.2 Hyperparametrit

Hyperparametrit ovat luokittelumalleille annettavia parametreja eli arvoja, jotka ovat yleensä numeroita tai mallin tietämiä arvoja, joilla säädellään mallin toimintaa. Hyperparametreillä ohjataan luokittelumallien yleistämistä, muuttujien valintaa ja mallien laskennalista tehokkuutta ja suoritusaikaa. Sopivien hyperparametrien löytäminen on haastavaa, ja jokaisella mallilla on niille omat erityispiirteet. Tässä tutkielmassa käytettävät hyperparametrit perustuvat Pythonin Scikit-learn -kirjaston malleihin, ja niiden määrittämisessä on hyödynnetty kirjaston dokumentaatiota, joka löytyy kirjaston kotisivulta `https://scikit-learn.org/stable/index.html`.

Logistisella regressiomallilla on olemassa erilaisia ratkaisijoita. Ratkaisijoilla tarkoitetaan mallin parametriestimoinnissa käytettyjä optimointialgoritmeja. Edellisessä kappaleessa esitettiin näistä ratkaisijoista newton-cg. Muita yleisesti käytettyjä ratkaisijoita ovat liblinear, lbfgs, sag ja saga. Liblinear toimii hyvin pienille ja keskisuurille aineistoille, sekä se tukee L1- ja L2-rangaistuksia. Lbfgs on yleiskäyttöinen ja nopea suurille aineistoille ja se tukee L2-rangaistusta. Sag soveltuu erityisen suurille aineistoille, ja saga on sen uudempi ja paranneltu versio.

Yleisesti eri koneoppimismallien keskeiset hyperparametrit hallitsevat mallin monimutkaisuutta ja ylisovittamista. Seuraavien logistisen regressiomallin keskeisimpien hyperparametrien analysointiin ja esittämiseen on hyödynnetty lähdettä [16].

Logistisessa regressiomallissa yleisiä ongelmia ovat ylisovittaminen ja monikollineaarisuus, erityisesti kun kyse on parametriarvioinnissa. Näin ollen apuna käytetään L2-rangaistusta, jolloin 2 lauseke saa uuden rangaistusparametrin

$$l_{rangaistus}(\beta, Y, X) = l(\beta, Y, X) - \lambda \sum_{j=1}^{p} \beta_j^2. \tag{5}$$

Kyseisessä lausekkeessa λ on rangaistusparametri, jolla pystytään kontrolloimaan lausekkeen kutistuksen voimakkuutta. Toisin sanoen, jos λ on suuri, tällöin kertoimet lähestyvät nollaa, mutta eivät koskaan saavuta sitä kuitenkaan. Näin ollen L2-rangaistus ei tee muuttujavalintaa eli ei poista yhtäkään muuttujaa mallista, vaan vakauttaa mallin parametriarviot pienentämällä muuttujien painoja. Tästä

seuraa myös se, että kaavaan 4 vaihdetaan tilalle edellä esitetty penalisoitu log-uskottavuuden gradientti ja Hessian.

Hyperparametrilla *dual* määritetään, että ratkaistaanko alkuperäisen tehtävän sijasta siitä johdettu duaalimuotoinen tehtävä. Se voi saada arvon True tai oletuksena olevan False. Duaalitehtävä valittaessa täytyy käyttää liblinear ratkaisijaa. Hyperparametri *tol* määrittää optimoinnin pysäyttämiskriteerin, eli kuinka pieni parannus riittää, jottei algoritmi jatka seuraavalle iterointikierrokselle. Tämä tarkoittaa, että kun kahden peräkkäisen Newton-Raphson 4 iteraatiokierroksen välinen ero on alle pysäyttämiskriteerin, malli pysähtyy.

Hyperparametri C määrittää voimakkuuden regularisoinnille, joka on lausekkeessa 5 esitetyn parametrin λ käänteisluku. Mitä suuremman arvon hyperparametri C saa, sitä heikompi on L2 rangaistus, koska

$$\lambda = \frac{1}{C}.$$

Hyperparametri *fit_ intercept* poistaa vakiotermin β_0, jolloin malli pakotetaan kulkemaan origon kautta. Tästä voi olla hyötyä, jos aineiston muuttujat eivät ole mitenkään keskitetty, jolloin jokin perusviitepiste aineistoon kannattaa lisätä.

Hyperparametri *class_ weight* muuttaa log-uskottavuusfunktion painotuksen muotoa. Parametri *class_ weight* on erityisen tärkeä kun aineiston luokat ovat epätasapainoisia. Tämä hyperparametri muuttaa kaavan 3 muotoon

$$l_w(\beta, Y, X) = \sum_{i=1}^{n} w_i(y_i X_{(i)}\beta - ln(1 + e^{(X_{(i)}\beta)})).$$

Mahdolliset arvot ovat None, jolloin jokainen luokka saa saman painon tai balanced, jolloin painot asetetaan automaattisesti kaavalla $w_i = \frac{n}{k \cdot n_j}$, jossa n on kaikkien havaintojen lukumäärä, k on kaikkien luokkien lukumäärä ja n_j on luokan j havaintojen lukumäärä. Tällöin myös Newton-Raphson algoritmin gradientti muuttuu muodosta

$$\frac{\partial l}{\partial \beta}(\beta_j) = \sum_{i=1}^{n}(y_i - p_i)X_{(i)}$$

muotoon

$$(\frac{\partial l}{\partial \beta})_w(\beta_j) = \sum_{i=1}^{n} w_i(y_i - p_i)X_{(i)}.$$

Tämä sama tapahtuu myös Hessianille. Näin ollen harvinaisten luokkien havainnot saavat suuremman painon, joten malli huomioi ne paremmin. Yksi vaihtoehto on myös määrittää itse painot jokaiselle luokalle esimerkiksi $\{0 : 1, 1 : 3\}$, jolloin luokan 1 havainnot painavat kolme kertaa enemmän.

Seuraavat hyperparametrit ovat hyvin yleisiä ja tekevät lähes samat asiat oli koneoppimismalli mikä tahansa scikit-learn kirjastosta, joten tulevien koneoppimismallien kanssa näitä ei määritellä erikseen. Hyperparametri *max_ iter* määrittää, jokaiselle mallille suurimman sallitun iteraatioiden lukumäärän, mikäli algoritmi ei ole

konvergoinut eli saanut lopetusarvoansa vielä tässä vaiheessa. Lähtökohtaisesti tätä hyperparametria voidaan käyttää kaikissa tässä tutkielmassa esiteltävissä luokittelumalleissa. Lineaarisessa regressiomallissa *max_iter* rajoitin rajoittaa iteraatioiden määrää kaavan 4 kohdassa. Hyperparametri *verbose* määrittää kuinka paljon ratkaisija kertoo etenemisestään käyttäjälle. Mikäli sille antaa arvon 0, ratkaisija ei kerro mitään ja arvolla 1 tai suurempi se tulostaa tietoja iteraatiokierroksien etenemisestä. Mikäli sille annetaan arvo -1 se kertoo lähtökohtaisesti kaiken mallin koulutuksen edistymisestä. Hyperparametrilla *n_jobs* määritetään kuinka montaa CPU-ydintä tai prosessoria käytetään rinnakkaislaskentaan mallin sovituksen aikana. Oletuksena se on None eli käytetään vain yhtä. Mikäli sille antaa arvon -1, tällöin käytetään kaikkia saatavilla olevia ytimiä.

Viimeisenä esitellään hyperparametri L1, eli *LASSO* parametri, jota ei voida hyödyntää esittämässäni tavassa ratkaista lineaarista regressiomallia, mutta se on erittäin käytetty hyperparametri muiden lineaaristen regressiomallien ratkaisijoiden kanssa. Kuten 2 esitetään logistisen regression maksimoivan uskottavuutta. L1 hyperparametrin lisääminen muokkaa tavoitteeksi

$$\max_{\beta}(l(\beta, \mathbf{Y}, \mathbf{X}) - \lambda \sum_{j=1}^{p} |\beta_j|),$$

jossa rangaistusparametri $\lambda > 0$. Scikit-learn kirjastossa tämä vastaava on $\frac{1}{C}$. Tähän asti kyseistä hyperparametria voidaan käyttää, mutta seuraava vaihe estää käytön. Tällä on vaikutus myös Newton-Raphsonin kaavaan 4

$$l_{rangaistus}(\beta) = l(\beta) - \lambda \sum_{j=1}^{p} |\beta_j|.$$

Tällöin joudutaan kuitenkin siirtymään käyttämään liblinear tai saga ratkaisijaa, jotka pystyvät käyttämään koordinaattilaskeutumista.

3.2 Random Forest

Random Forest on valvotun koneoppimisen malli, joka kuuluu puumalleihin. Tässä tutkielmassa sitä käytetään luokittelumallina, joka pystyy havaitsemaan aineistosa ei-lineaarisia riippuvuuksia. Vuonna 2001 Breiman kehitti Random Forest -algoritmin, joka yhdistää useita päätöspuita yhdeksi malliksi ja tuottaa vakaita ja tarkkoja ennusteita [17]. Random Forest on yksi menestyneimmistä ja laajimmin käytetyistä koneoppimismalleista. Se skaalautuu hyvin suurillekin aineistoille ja säilyttää samalla tilastollisen tehokkuuden luotettavien päätelmien tekemiseen [18].

Random Forest hyödyntää Breimanin vuonna 1996 esittelemää bootstrap-näytteenottoa, jossa yhdestä mallista koulutetaan useaan kertaan eri versioita alkuperäisestä aineistosta saatavalla osa-aineistolla [19]. Mallissa käytetään lisäksi satunnaisotantaa päätöspuun lehtien muodostamiseen solmuissa. Yksittäisissä päätöspuissa solmujen jakopisteiksi valitaan ne muuttujat, jotka tuottavat parhaimman jaon. Random Forestissa muuttujat jaetaan sattumanvaraisesti, mikä vähentää päätöspuiden keskenäistä korrelaatiota. Alkuperäisestä muuttujajoukosta, jonka koko on p valitaan yleensä luokittelutehtäviin $\sqrt{p}$ muuttujaa ja regressiotehtäviin $\frac{p}{3}$ muuttujaa

jokaiseen solmuun. Breimanin analyysit osoittavat mallin säilyttävän riittävän suuren yleistämisvirheen, vaikka yksittäiset päätöspuut voivat ylioppia, kokonaisuutena malli ei kuitenkaan siihen taivu [17].

3.2.1 Määritelmä

Tässä määritelmässä esitellään Random Forestista yleistetty malli luokitteluongelmalle, jossa voidaan muuttaa jokaisen yksittäisen puun merkitystä eli painoarvoa mallille. Määritelmä pohjautuu Chenin, Yun ja Zhangin esittämään muunnokseen koneoppimismallista [18].

Oletetaan ominaisuusvektorin $\mathbf{x}_i \in \mathbb{R}^m$ ja päätösmuuttujan $y_i \in \{0, 1\}$ olevan samat kuin määritelmässä 3.1.1. Random Forest generoi päätösmuuttujan arvon

$$y_i = \mu_i + e_i,$$

jossa $\mu = \mathbb{E}(y_i|\mathbf{x}_i)$ on mallin odotusarvo, jota pyritään mallilla ennustamaan. Kaikille mallin havaintopareille $\{e_i, \mathbf{x}_i\}$ pätee perusoletus, että ne ovat itsenäisesti ja identtisesti jakautuneita. Mallin virhetermin ensimmäinen ominaisuus $\mathbb{E}(e_i|\mathbf{x}_i) = 0$ tarkoittaa, että mallilla ei ole systemaattista harhaa, vaan virhetermin odotusarvo on 0. Toinen ominaisuus $\mathbb{E}(e_i^2|\mathbf{x_i}) = \sigma_i^2$ tarkoittaa, että mallin virhetermi saa vaihdella jokaisen eri havainnon $\mathbf{x}_i$ kohdalla. Tällöin virheen hajonta riippuu ominaisuusvektorista, mikä tunnetaan heteroskedastisuutena.

Olkoon aineisto $D = \{\{\mathbf{x}_i, y_i\}\}_{1 \leq i \leq n}$ samat kuin kohdassa 3.1.1 esitetty aineisto. Kun tunnetaan ominaisuusvektorin $\mathbf{x}_i$ päätösmuuttuja y_i, voidaan esittää yksi puumalli BL kaikista Random Forestin puista

$$\hat{y}_i = \mathbf{P}_{BL}^T(\mathbf{x}_i, \mathbf{X}, \mathbf{y}, \mathcal{B}, \Theta)\mathbf{y},$$

jossa $\mathbf{y} = (y_1, y_2, ..., y_n)$ on päätösmuuttujien vektori, $\mathbf{P}_{BL}^T(\mathbf{x}_i, \mathbf{X}, \mathbf{y}, \mathcal{B}, \Theta)$ on nulotteinen vektori, joka määrittää yksittäisen puun rakenteen, ja $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n)$ on ominaisuuksien matriisi. Yksittäisen puun rakenteessa $\mathbf{x}_i$ on uusi havaintovektori, jota ennustetaan, $\mathbf{X}$ on koulutusdatan piirrejoukko, $\mathbf{y}$ on koulutusdatan päätösmuuttujat ja $\mathcal{B}$ on koulutukseen käytetty bootstrap-osa ja Θ kuvastaa tämän yksittäisen puun satunnaisuutta.

Muuttujat $\mathcal{B}$ ja Θ tuovat malliin yksittäisten puiden kautta satunnaisuutta. Muuttuja $\mathcal{B}$ määrittää havaintojen valintaan satunnaisuutta, kun tehdään jokaiselle puulle erikseen bootstrap-otantaa. Tämä on bagging-vaihe [19]. Koska kaikissa solmuissa ei tarkastella kaikkia ominaisuuksia, määrittää Θ jokaiselle puun solmulle tulevien ominaisuuksien satunnaisuutta. Näin ollen puista tulee keskenään erilaisia ja niiden väliset korrelaatiot pienenevät, mistä seuraa kokonaisuutena paremmin toimiva malli.

Oletetaan olevan M_n bootstrap-aineistoa, josta on koulutettu Random Forest -malli valmiiksi. Tässä huomiona on, että jokainen puu on kasvatettu eri sattumanvaraisesti valitulla aineistolla alkuperäisestä aineistosta. Lähtökohtaisesti jokainen puu on erilainen. Tarkastellaan yksittäistä puuta m. Syötetään havainto $(y_0, \mathbf{x}_0)$ valitsemaan m puuhun. Havainto käy puun läpi päätyen johonkin puun lehteen l, jossa on jokin määrä n_l koulutusdatan pistettä

$$\mathcal{D}_l = \{(y_{i1}, \mathbf{x}_{i1}), ..., (y_{i_{n_l}}, \mathbf{x}_{i_{n_l}})\}$$

joiden mukaan luodaan ennuste ominaisuusvektorille $\mathbf{x}_0$. Koska bootstrap-näytteessä sama havainto voi esiintyä useammin, oletetaan havainnon $(\mathbf{x}_i, y_i)$ esiintyvän puun lehdessä h_i kertaa. Näin ollen vektori $\mathbf{P}_{BL}(\mathbf{x}_0, \mathbf{X}, \mathbf{y}, \mathcal{B}, \Theta)$ on harva ja suurin osa sen havainnoista on $\frac{h_i}{n_l}$ tai nollia. Kuitenkin kyseisessä vektorissa suurin osa arvoista on nollia johtuen sen harvuudesta. Eri suurta kuin nolla arvot tulevat pisteistä, jotka ovat samassa lehdessä l kuin ominaisuusvektori $\mathbf{x}_0$. Toisin sanoen vektorissa $\mathbf{P}_{BL}(\mathbf{x}_0, \mathbf{X}, \mathbf{y}, \mathcal{B}, \Theta)$ elementti i on 0 jos $(y_i, \mathbf{x}_i) \notin \mathcal{D}$ ja $(y_0, \mathbf{x}_0) \in \mathcal{D}_l$. Vektorin $\mathbf{P}_{BL}$ alkiot ovat painokertoimia muodostettaessa ennustetta päätösmuuttujalle y_0 vektorista $\mathbf{y}$. Bootstrap-aineistoa käytetään kasvattamaan puita alkuperäisen koulutusaineiston sijasta. Bootstrap lisää malliin satunnaisuutta, koska havainnot voivat esiintyä useammin kuin alkuperäisessä aineistossa.

Normaalissa päätöspuussa käytetään jokaista muuttujaa p puun haarottamiseen. Random Forest -menetelmässä sen sijaan valitaan satunnaisesti esimerkiksi $q(q < \sqrt{p})$ muuttujaa jakoehtojen löytämiseen. Ilman bootstrapia jokainen havainto i olisi mukana jokaisessa puussa täsmälleen kerran. Tästä seuraisi myös se, että vektori $\mathbf{P}_{BL}$ olisi vähemmän harvempi, koska kaikki havainnot olisivat aina mukana.

Random Forest -mallin puun m ennuste päätösmuuttujalle f_m on

$$f_m(\mathbf{x}_i) = \mathbf{P}_{BL(m)}(\mathbf{x}_i, \mathbf{X}, \mathbf{y}, \mathcal{B}_{(m)}, \Theta_{(m)})\mathbf{y},$$

jossa $\mathbf{P}_{BL(m)}(\mathbf{x}_i, \mathbf{X}, \mathbf{y}, \mathcal{B}_{(m)}, \Theta_{(m)})\mathbf{y}$ on kyseisen puun n-ulotteinen konfiguraatiovektori. Mallin lopullinen tulos on muotoa

$$\hat{y}_i(\mathbf{w}) = \sum_{m=1}^{M_n} w_{(m)} f_m(\mathbf{x}_i), \tag{6}$$

jossa w kuvastaa puun m painoa. Puiden painot voivat olla tasaisesti jaettuja, tai niiden painoja voidaan muuttaa saatujen tulosten perusteella. Perinteinen Random Forest -malli käyttää tasapainoisia painoja.

$$w_{(m)} = \frac{1}{M_n},$$

jolloin ennuste saadaan laskemalla kaikkien puiden ennusteiden aritmeettinen keskiarvo. Edellä esitetty formaali rakenne toimii siltana seuraavaksi esiteltävään koneoppimismalliin.

3.2.2 Hyperparametrit

Random Forest -mallissa käyttäjän on määritettävä useita hyperparametrejä, jotta malli toimisi tehokkaasti. Seuraavien hyperparametrien määrittämisessä on käytetty

lähdettä [20] ja scikit-learn kirjastoa https://scikit-learn.org/stable/index.html.

Yksi keskeisistä hyperparametreistä on metsän puiden lukumäärä. Puiden määrän kasvattaminen ei lisää ylisovittamista, vaan pienentää varianssia ja vakauttaa mallin ennusteita [17]. Puiden määrää lisättäessä on huomioitava, että jokaisen puun lisääminen malliin kasvattaa sen tarvitsemaa laskenta-aikaa. Näin ollen puiden lisääminen merkitsee kompromissia laskenta-ajan ja mallin vakauden välillä. Puiden määrää voi säätää hyperparametrilla *n_estimators*, joka on usein oletuksena 500 tai 1000 puuta. Säätämällä *n_estimators* hyperparametria säädetään kaavaa 6, jolloin saadaan arvo $M_n = n\_estimators$, joka kertoo puiden määrän mallissa. Mikäli M_n on pieni malli saa suurella varianssilla ennusteita eri ajokerroilta ja arvon ollessa suuri ennuste pysyy enemmän samanlaisena ajokerroista riippumatta.

Max_features on toinen keskeisimmistä Random Forest-mallin hyperparametreistä, jolla määritetään montako ominaisuutta jokaisessa solmussa on valittavana. Luokittelumallissa yleensä oletuksena on $max\_features = \sqrt{p}$, mutta se voi myös olla osassa tapauksista $log_2(p)$. Mikäli *max_features* arvo on suuri, sitä vahvempia yksittäiset puut ovat, mutta samalla puut korreloivat toistensa kanssa enemmän. Liian suuri korrelaatio tekee puista liian samanlaisia keskenään, jolloin mallin ennusteet eivät monipuolistu tarpeeksi. Mikäli *max_features* on liian pieni puista tulee heikkoja, jolloin ennusteista tulee samanlaisia. *Max_features* ei suoranaisesti vaikuta Random Forest-mallin määritelmään, vaan se määritää ainoastaan lausekkeeseen 6 solmujen valinta-avaruuden.

Jokainen Random Forest-mallin puu rakentuu omalle otosjoukolleen alkuperäisestä aineistosta. Breimanin esittelemässä Random Forest-algoritmissa otoskoko otetaan bootstrapilla, jossa puusta arvotaan n havaintoa takaisin panolla [17]. Hyperparametri *sample_size* määrittää, otoskoon suhteessa alkuperäiseen aineistoon. Oletuksena *sample_size* on samankokoinen kuin alkuperäinen aineisto. Suuri otoskoko lisää puiden samankaltaisuutta, kun taas pieni otoskoko heikentää yksittäisten puiden tarkkuutta. Mikäli otoskokoa pienennetään se johtaa ennusteiden varianssin vähenemiseen, jolloin puista tulee vähemmän samanlaisia, mutta mallin tuottamat ennustee ovat enemmän harhaanjohtavia [20]. Tämä hyperparametri ei myöskään näy mallin määritelmässä, vaan se määrittää monestako aineiston esimerkkitapauksesta malli voi oppia. *Sample_size* hyperparamtriin liittyy vahavasti replacement hyperparametri, jolla määritetään mallille annettavan esimerkkiaineiston poimintamenetelmästä. Mikäli *replacement* saa arvon True, käytetään bootstrap menetelmää, jolloin jokainen mallille jo aineistosta valittu esimerkki palautetaan takaisin aineistoon seuraavan esimerkin valintaa varten. Näin ollen sama havainto voi esiintyä useita kertoja bootstrap-näytteessä, mistä voi seurata muuttujavalintavinouma. Tästä voi seurata puiden vahvempaa korrelaatiota keskenään. Jos *replacement* on False, kukin havainto voi esiintyä näytteessä vain kerran.

Lähteessä [20] käytetään node size, joka on hyperparametrina *min_samples_leaf*. Tällä hyperparametrilla määritetään maksimimäärä lehtisolmuissa oleville havainnoille. Oletusarvoisesti luokitteluun käytettäessä maksimimäärä on yksi havainto. Tällöin yksittäistä puuta jatketaan niin kauan kunnes lehtisolmussa on jäljellä ainoastaan yksi havainto. Näin ollen *min_samples_leaf* määrittää puun syvyyttä siten, että suuremmilla arvoilla puista tulee lyhyempiä ja eivät seuraa yksittäisten

havaintojen erityispiirteitä, koska lehtisolmuun voi jäädä useampi havainto. Mikäli *min_ samples_ leaf* on pieni arvo, malli oppii jokaisen havainnon erityispiirteet todella tarkasti. Tähän vahvasti liittyvä toinen hyperparametri on *min_ samples_ leaf*, jolla voidaan määrittää minimi määrä havaintoja jokaisen jaon jälkeen, jotta kyseinen jako voidaan tehdä.

Random Forest-luokittelumallissa yleisesti käytetään *Gini*- tai entropia -arvoa jakokriteerinä. Hyperparametrilla *criterion* voi valita sopivamman jakokriteerin, jolla valitaan millä ehdoilla havaintojoukko jaetaan kahteen tai useampaan osaan. Hyperparametrin *ginin* -lauseessa

$$Gini(t) = 1 - \sum_{k=1}^{K} p_{k,t}^2,$$

jossa $p_{k,t}$ on luokan k osuus solmussa t. Mikäli *gini* saa arvokseen 0, tällöin kyseisessä solmussa on vain ja ainoastaan havaintoja yhdestä luokasta. Mikäli *gini* saa arvokseen 0.5 on luokkajako tasan, mutta jos on kolme luokkaa ja luokkajako on tasan saadaan arvoksi 0.67. Esimerkiksi kuvitellaan tilanne, jossa on 5 havaintoa luokasta A ja 4 havaintoa luokasta B. Tästä saadaan ennen jakoa:

$$Gini = 1 - (\frac{5}{10})^2 - (\frac{4}{10})^2 = 1 - 0,25 - 0,16 = 0,59.$$

Nyt harkitaan jotain jakoa. Ensimmäiseen haaraan saadaan 4 havaintoa luokasta A ja 3 havaintoa luokasta B. Tästä saadaan

$$Gini_1 = 1 - (\frac{4}{7})^2 - (\frac{3}{7})^2 = 1 - 0,3249 - 0,1764 = 0,4987,$$

sekä toiseen haaraan tulee 2 luokasta A ja 1 luokasta B.

$$Gini_2 = 1 - (\frac{2}{3})^2 - (\frac{1}{3})^2 = 1 - 0,44435 - 0,110889 = 0,444761.$$

Näin ollen yhdistetty Gini on jaon jälkeen näiden tuloksien painoitettukeskiarvo

$$Gini_{painoitettu} = \frac{7}{10} \cdot 0,4987 + \frac{3}{10} \cdot 0,444761 = 0,482518.$$

Suoritettu jako paransi solmun puhtautta.

Toinen yleisesti käytetty vaihtoehto *ginille* on entropia, joka mittaa solmun epäpuhtautta. Entropia lause

$$H(t) = -\sum_{k=1}^{K} p_{k,t} log(p_{k.t}),$$

jossa t on solmu, K on luokka ja $p_{k,t}$ on luokan k osuus solmussa. Luokka on taas täysin puhdas mikäli jokin $p_{k.t} = 1$ ja kaikki muut ovat 0, ja tästä seuraa, että $H(t) = 0$. Mikäli kaikki havainnot jakautuisivat tasaisesti luokkien kesken entropia saa maksimaalisen arvonsa. Esimerkiksi kuvitellaan tilanne, jossa on 50% luokkaa A ja 50% luokkaa B, tällöin $p_A = 0.5$ ja $p_B = 0.5$. Nyt kaavasta saadaan

$$H = -(0.5log0.5 + 0.5log0.5) = log2 = 0,693...,$$

joten tässä on suuri epävarmuus. Random Forest-luokittelumalli on saanut myös uuden jakokriteerin *log_ loss*, joka käyttää todennäköisyyksiä mittaamaan jaon onnistuneisuutta. Siinä missä *Gini* ja Entropia huomioi ainoastaan solmujen epäpuhtauksia *log_ loss* huomioi myös sen, miten hyvät todenäköisyyden malli antaa. *Log_ loss* määritetään seuraavasti

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i ln(\hat{p}_i) + (1 - y_i) ln(1 - \hat{p}_i)],$$

jossa $y_i \in \{0, 1\}$ ja $\hat{p}_i$ on mallin arvioima todennäköisyys luokalle 1.

Yksittäisten puiden syvyyttä voidaan rajoittaa hyperparametri *max_ depth*, jolla määritetään kuinka monta jakoa voidaan tehdä juuresta lehteen. Oletusarvoisesti tämä on yleensä rajoittamattomana eli puut voivat kasvaa juuri niin pitkiksi kuin tarvitsee. Toinen hyperparametri rajoite on *max_ nodes*, jolla voidaan säätää solmujen määrää koko puussa. Hyperparametrilla *class_ weight* voidaan määrittää aineistosssa pienemmän luokan painoarvo, jotta saadaan tasapainoitettua epätasaisia luokkajakaumia. Tällä saadaan mallin suorituskykyä paremmaksi vähemmistöluokkaan. Hyperparametri *sample_ weight* mahdollistaa yksittäisten havaintojen painottamisen, esimerkiksi jos joillakin havainnoilla katsotaan olevan erityistä merkitystä. Näin malli saadaan luottamaan joihinkin havaintoihin enemmän.

3.3 Extreme Gradient Boosting (XGBoost)

Extreme gradient boosting -malli (XGBoost) on valvotun oppimisen ensamble-malli, joka perustuu gradienttivahvistettuihin päätöspuihin ja sitä voidaan käyttää sekä regressio- että luokittelutehtävissä. Tässä tutkielmassa sitä käytetään luokittelumallina. Nimensä mukaisesti malli perustuu gradientin tehostamiseen, eli aikaisempien mallien virheiden korjaamiseen ja ominaisuuksien parantamiseen. XGBoost rakentaa mallinsa päätöspuista, niin kuin aikaisemmin esitelty Random Forest, mutta hyödyntäen gradienttipohjaista oppimista puiden yhdistämisessä. XGBoost minimoi säännöllistettyä kohdefunktiota, jossa se hyödyntää toisen kertaluvun gradienttitietoa.

XGBoostin kehittäjä Tianqi Chen julkaisi mallin GitHubissa vuonna 2014, ja varsinainen tutkimusartikkelin vuonna 2016 [21]. Jo artikkelin julkistamisajankohtana malli oli voittanut useita koneoppimiskilpailuja.

3.3.1 Määritelmä

Tässä määritelmässä hyödynnetään lähteitä [21] ja [22] sekä käytetään tutkielmassa aiemmin esiteltyä Random Forest -mallia. Yksinkertaisuuden vuoksi aikaisemmin käytetyn yksittäisen päätöspuun ennusteen $\mathbf{P}_{BL(m)}$ mallintamisen sijasta esitetään se muodossa $f_t(\mathbf{x}_i) = w_{q(\mathbf{x}_i)}$, jossa $w_j \in \mathbb{R}$ on yksittäisen lehtisolmun arvo.

Kuten aiemmin kappaleessa 3.3 ja kaavassa 6 esiteltiin, malli voidaan kirjoittaa seuraavasti

$$\hat{y}_i(\mathbf{w}) = \sum_{m=1}^{M_n} w_{(m)} f_m(\mathbf{x}_i). \tag{7}$$

Random Forest-mallissa jokaisen yksittäisen puun paino on tasainen,

$$w_{(m)} = \frac{1}{M_n}.$$

Tämä kuvastaa satunnaismetsän toimintaperiaatetta. Mutta XGBoostissa sen sijaan puiden painot eivät ole ennalta määrättyjä, vaan ne johdetaan optimointiprosessin kautta tavoitefunktion minimoinnissa. Yksi keskeinen ero XGBoostini ja Random Forestin välillä on puiden painojen määrittäminen ja se, miten päätöspuut rakentuvat. XGBoost-mallissa päätöspuut rakentuvat iteratiivisesti aina edellisen niin sanotun heikon puun päälle ja siinä tavoitteena on minimoida tavoitefunktiota. Kun taas Random Forestissa jokainen päätöspuista rakentuu toisistaan riippumattomina, jonka jälkeen niiden keskiarvot yhdistetään.

XGBoostissa tavoitefunktiolla on kaksi roolia. Sen tarkoituksena on toisaalta ohjata mallia rakentamaan mahdollisimman tarkkoja päätöspuita, mutta samalla sen tarkoituksena on rajoittaa liian monimutkaisia rakenteita rankaisemalla niitä. Tällöin tarvitaan funktio

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \tag{8}$$

jossa $l(\hat{y}_i, y_i)$ on virhefunktio, joka voi olla logistinen tappiofunktio tai neliösumma. Mallissa f_k on tarkastelussa oleva yksittäinen päätöspuu ja $\Omega(f_k)$ on päätöspuun kompleksisuutta mittaava regularisaatiotermi, joka on muotoa $\Omega(f) = \gamma T + \frac{1}{2}\lambda||w||^2$. Tässä T on puun lehtien kokonaismäärä, jonka rangaistuskerroin on γ. Termi $||w||^2$ on lehdistä saatavien arvojen neliösumma ja λ on sen L2-säännöllistämisen kerroin tai toisin sanoen painokerroin. Näiden funktioiden ansiosta malli kykenee hallitsemaan suuria määriä muuttujia ja kohinaa.

Kohdan 8 funktiota ei voida käsitellä euklidisessa avaruudessa, koska se sisältää funktioita parametreinaan. Tästä seuraa se, ettei funktiota voida optimoida perinteisiä optimointimenetelmiä käyttäen, tällöin malli opetetaan additiivisesti. Mallia opetetaan additiivisesti: jokaisella iteraatiokierroksella tavoitteena on lisätä uusi puu f_t, joka pienentää virhettä mahdollisimman paljon ja samalla pienentää regularisointia. Olkoon havaintopisteen i ennusteena $\hat{y}_i^{(t)}$ iteraatiokierroksella t. Nyt lisätään funktio f_t siten, että sen lisääminen minimoi tavoitefunktion

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t), \tag{9}$$

jossa $\hat{y}_i^{(t-1)}$ on edellisen mallin ennuste ja $f_t(\mathbf{x}_i)$ on päätöspuu, joka niin sanotusti ahneesti lisätään malliin. Ahneeksi lisäykseksi sanotaan sellaista seuraavan funktion lisäystä, jossa malli hyötyy vain ja ainoastaan sillä hetkellä eniten. Ja kuten edellä mainittu, funktio $l(y_i,)$ on mallissa käytettävä virhefunktio ja funktio $\Omega(f_t)$ on rangaistusfunktio. Kaavaa 9 ei optimoida perinteisesti, vaan Taylorin sarjan avulla.

Mallin virhefunktio on usein vaikea tai jopa mahdoton optimoida tehokkaasti. Näin ollen käytetään toisen kertaluvun Taylorin sarjaa funktion approksimointiin. Tästä saadaan laajennettu kaava

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n}[l(y_i, \hat{y}_i^{(t-1)} + g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t), \tag{10}$$

jossa $l(y_i, \hat{y}_i^{(t-1)})$ on virhe ennen uutta päätöspuuta ja virhe on vakio tässä vaiheessa. Kaavassa $g_i f_t(\mathbf{x}_i)$ on gradienttiosuus, joka kertoo, miten paljon uusi päätöspuu muuttaa mallin virhettä. Kohta $\frac{1}{2}h_i f_t^2(\mathbf{x}_i)$ kertoo, kuinka paljon virheen vähenemisen nopeus muuttuu, koska Hessian korjaa liian suuria muutoksia. Viimeinen funktio $\Omega(f_t)$ edelleen rankaisee mallin monimutkaisuudesta. Mallissa funktio $g_i = \frac{\partial}{\partial \hat{y}^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ kertoo virheen muutoksen suuntaan $f_t(\mathbf{x}_i)$ ja funktio $h_i = \frac{\partial^2}{\partial \hat{y}^{(t-1)2}} l(y_i, y_i^{(t-1)})$ on toinen derivaatta eli se kertoo, kuinka nopeasti gradientti g_i muuttuu ennusteen $\hat{y}^{(t-1)}$ suhteen. Koska kohdan 10 funktiossa olevat vakiotermit eivät riipu seuraavaksi lisättävästä päätöspuusta, voidaan ne poistaa. Näin ollen saadaan yksinkertaisempi optimoitava funktio askeleelle t

$$\hat{\mathcal{L}}^{(t)} = \sum_{n=1}^{n}[g_i f_t(\mathbf{x}_i) + \frac{1}{2}f_t^2(\mathbf{x}_i)] + \Omega(f_t), \tag{11}$$

josta seuraa varsinainen XGBoost-malli.

Olkoon $I_j = \{i|q(x) = j\}$ kaikki havaintoja i, jotka päätyivät lehteen j päätöspuun solmuissa tehtyjen päätösten perusteella. Tällöin voidaan summat ryhmitellä lehtien mukaan, minkä seurauksena kaava 11 voidaan kirjoittaa avaamalla Ω, josta saadaan

$$\begin{aligned}\hat{\mathcal{L}}^{(t)} &= \sum_{i=1}[g_i f_t(\mathbf{x}_i + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T.\end{aligned}$$

Nyt nähdään, että γT rankaisee mallia, jos on T määrä lehtiä. Muuttuja λ on mallin L2-regularisaatiotermi, joka pyrkii pitämään mallin parametrien arvot riittävän pieninä.

Puun rakenteen $q(x)$, joka ohjaa jokaisen havaintopisteen i oikeaan lehteen, kiinnittämisen jälkeen lasketaan lehden j optimaalisin paino w_j^* kaavasta

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda} \tag{12}$$

ja vastaavasti optimaalisin tavoitefunktion arvo saadaan kaavasta

$$\hat{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in g_i} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T, \tag{13}$$

joissa g_i on gradientti eli kertoo kuinka väärässä malli on havainnosta i.

Vaikkakin kaava 13 johdetaan useammasta joukosta tavoitefunktioita, voidaan sitä käyttää mittaamaan puun rakenteen q laatua. Todellisuudessa on kuitenkin lähes mahdotonta luetella kaikkia mahdollisia päätöspuiden rakenteita q. Tästä syystä ahneita algoritmeja käytetään tässä. Olkoon I_v ja I_o vasemman ja oikean puunhaaran jakamat havaintojoukot. Näin ollen on selvää, että $I = I_v \cup I_o$ ja häviön minimi jakamisen jälkeen saadaan kaava

$$\mathcal{L}_{jako} = \frac{1}{2}[\frac{(\sum_{i \in I_v} g_i)^2}{\sum_{i \in I_v} h_i + \lambda} + \frac{(\sum_{i \in I_o} g_i)^2}{\sum_{i \in I_o} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda}],$$

jota voidaan käyttää jakoehdokkaiden arviointiin.

3.3.2 Hyperparametrit

Extreme gradient boosting -luokittelumalli ei ole sellaissenaan valmis, vaan sen toimivuus edellyttää useiden hyperparametrien säätämistä. Hyperparametrien avulla voidaan ohjata mallin rakennetta, oppimisprosessin satunnaisotantaa sekä datan jakaumaa. XGBoostin hyvä ennustuskyky ja mallin yleistämiskyky ovat vahvasti yhteydessä hyperparametreihin, sillä niillä kontrolloidaan mallin harhan ja varianssin välistä tasapainoa [23]. Seuraavien hyperparametrien määrittely perustuu lähteeseen [23] sekä Pythonin Scikit-learn kirjastoa `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html`.

XGBoostin *n_ estimators* hyperparametrilla määritetään yläraja puiden lukumäärälle. Jos puita on liian vähän, malli ei opi aineistosta riittävästi. Jos puita on liikaa, malli ylisovittuu aineistoon ja sen laskenta muuttuu hitaaksi. Käytännössä sopiva puumäärä on usein 20-120, riippuen aineiston koosta ja luonteesta [23]. Parametri *n_ estimators* antaa aikaisemmin määriteltyyn lauseeseen 6 summan ylärajan M. Näin ollen mallille kerrotaan montako kertaa lisätään uusi puu, jota optimoidaan.

Hyperparametri *n_ estimators* liittyy läheisesti hyperparametriin *learning_ rate*, joka määrittää kullekkin puulle annetavan painokertoimen. Pieni *learning_ rate* tekee oppimisesta varovaisemman ja vaatii suuremman määrän puita. Kun taas suuri *learning_ rate* tekee mallista agressiivisempi oppimaan, mutta ylisovitusriski kasvaa huomattavasti. Käytännön sovelluksissa *learnin_ rate* 0.05-0.3 on suositeltu vaihteluväli [23].

Hyperparametrilla *max_ depth* rajoitetaan jokaisen puun syvyyttä. Mikäli *max_ depth* on pieni, puista tulee matalia ja malli on yksinkertaisempi. Kun taas suurilla arvoilla malli oppii monimutkaisempia sääntöjä, mutta riski ylisovittamiseen kasvaa. Oletusarvo *max_ depth*-hyperparametrille on 3. On suositeltavaa käyttää arvoja 2-6, sillä tätä suuremmilla arvoilla mallin paraneminen on harvinaista ja varianssi kasvaa huomattavasti [23].

Hyperparametri *min_ child_ weight* on oletuksena 1. Tämä on hyvin samankaltainen hyperparametri kuin Random Forest-mallissa käytetty *min_ child_ weight*, mutta XGBoostissa otetaan lisäksi huomioon Hessian-gradientin antama varmuustieto. Kyseinen hyperparametri määrittää, mikä on pienin sallittu gradienttipainoitettu havaintomäärä lehden sisällä. Arvon ollessa suurempi, pieniä lehtiä ei synny, mikä johtaa ylisovituksen vähenemiseen.

Seuraava hyperparametri on *gamma*, joka on lisärangaistus jokaisen uuden jaon tekemiselle. Toisin sanoen, uutta jakoa ei tehdä, ellei tavoitefuntkiota parane vähintään arvolla *gamma*. Pieni *gamma* sallii myös hyvin vähäiset parannukset, jolloin puut kasvavat helposti monimutkaisemmiksi. Hyperparametrit *gamma*, *max_depth* ja *min_child_weight* toimivat erityisen hyvin yhdessä, koska ne rajoittavat puiden kasvua. Parametrin *gamman* arvoksi käytännönsovelluksissa suositellaan arvoa väliltä 0-5 [23].

Hyperparametrilla *subsample* säädetään koulutusaineiston otannan suuruutta varsinaisesta aineistosta. Jos *subsample*=1, koko aineistoa käytetään jokaisen puun rakentamiseen. Mikäli taas *subsample*<1, jokainen puu saa vain satunnaisen osan alkuperäisestä aineistosta. Käytännönsovelluksissa suositaan yleisesti arvoja 0.5-1 väliltä [23].

Random Forest-mallin *max_features* hyperparametri vastaa XGBoostissa *colsample_bytree*, joilla määritetään ominaisuuksien määrää puiden rakentamiseen. XGBoostissa on lisäksi hyperparametri *colsample_bylevel* ja *colsample_bynode*, joilla säädetään ominaisuuksien määrää taso- ja solmukohtaisesti. Näistä kolmesta parametrista arvolla 1 käytetään kaikki ominaisuudet, ja arvo < 1 käytettävissä olevia ominaisuuksia rajoitetaan. Yleisesti käytettynä arvo haetaan 0.6-1 väliltä ristiin validoimalla [23].

XGBoost-luokittelumallissa on olemassa sisäänrakennettu regularisaatio lehtisolmujen painoille [23]. Hyperparametrilla λ voidaan säätää L2-rangaistusta lehtien painoille. Mitä suuremman arvon λ saa sen enemmän yksittäisiä puita rajoitetaan tekemään suuria muutoksia edelliseen. Tyypillisesti λ saa arvoja väliltä 1-10. L2-rangaistustermillä määritetään arvo muuttujalle λ rangaistuslausekeessa 12 ja itse rangaistustermille $\Omega(f) = \gamma T + \frac{1}{2}\lambda||w||^2$. Lisäksi voidaan käyttää α-hyperparametria, joka lisää malliin L1-rangaistuksen. Arvot muuttujalle α ovat väliltä 0-1, mutta joissakin sovelluksissa voidaan arvoa kasvattaa suuremmaksikin [23]. Kun otetaan käyttöön α, rangaistustermi saa seuraavan muodon

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_j w_j^2 + \alpha \sum_j |w_j|,$$

joka on L2-rangaistustermi ja lisäämällä muuttuja α saadaan siihen myös L1-rangaistustermi, joka on tuo jälkimmäinen osa lauseketta.

Toinen keskeinen ongelma liittyy epätasapainoon positiivisten ja negatiivisten havaintojen välillä, jolloin luokittelumalli voi oppia positiiviset havainnot paremmin kuin negatiiviset. Tähän ongelmaan apuna XGBoost-luokittelumallissa on hyperparametri *scale_pos_weight*, jolla pystytään skaalaamaan luokittelumallin huomioimat havaintojen painoarvot positiivisen ja negatiivisen luokan välillä. Tällöin malli ottaa paremmin huomioon aineiston harvinaisemman luokan virheet. Oletusarvoisesti kaikkien luokkien paino on yksi. Painot voidaan määrittää kaavalla

$$\text{scale\_pos\_weight} = \frac{\text{negatiivisten\_lukumäärä}}{\text{positiivisten\_lukumäärä}},$$

mikäli käytännönsovellus niin vaatii [23].

Lisäksi epätasapainoisissa aineistoissa voidaan käyttää hyperparametria *max_delta_step*, jolla rajoitetaan yksittäisen boosting-askeleen painojen maksimipäivi-

tyksiä. Käytännössä arvo rajataan usein välille 1-20 [23]. Hyperparametrilla rajataan lauseen 12 termin w_j^* absoluuttista arvoa.

Ei varsinainen mallin oma hyperparametri, mutta voidaan säätää mallin turhaa kouluttamista on parametri *early_stopping_rounds*, jolla saadaan XGBoost-luokittelumallin oppiminen pysäytettyä, jos se ei ole oppinut enää jonkin ennalta määrätyn kierroslukumäärän jälkeen. Tällä tavalla estetään turhien puiden kasvattaminen ja säästetään laskenta-aikaa huomattavasti. Parametri *early_stopping_rounds* ei muuta itse XGBoost-luokittelumallia yhtään. Tämän on todettu olevan yksi tehokkaimmista tavoista hallita mallin kompleksisuutta [23].

4 Koneoppimismallien arviointi

Koneoppimismallien suorituskykyä arvioidaan koulutusvaiheessa erilaisilla mittareilla, jotta ne toimisivat mahdollisimman hyvin varsinaisessa aineistossa. Keskeisiä kysymyksiä ovat mallin koulutuksen onnistuminen ja sen soveltuvuus tehtävään, joihin voidaan vaikuttaa mallin koulutusvaiheessa. Joissakin tapauksissa koneoppimismalli voidaan hylätä kokonaan. Yleensä kuitenkin koulutusvaiheessa löydetään parannettavat hyperparametrit, joita säätämällä malli saadaan oppimaan koulutusaineistosta paremmin.

Koneoppimismalleja kouluttaessa ei voida käyttää vain yhtä mittaria, kun arvioidaan mallin suorituskykyä. Esimerkiksi tutkielman aineistossa 78% lainanhakijoista ovat maksaneet lainansa ongelmitta takaisin, joten aineisto on epätasapainossa. Tämä vastaa suurinta osaa reaalimaailman aineistoista.

Jotta luokittelumallien keskeiset mittarit voidaan esitellä, seuraavaksi määritetään yleiset käsitteet, joita mittarit hyödyntävät. Seuraavat määritelmät erilaisistä luokittelutilanteista perustuvat lähteeseen [24]. Oikea positiivinen (TP) tarkoittaa tilannetta, jossa havainto luokitellaan oikein positiiviseen luokkaan. Esimerkiksi lainanhakija A ei maksaa lainaansa takaisin ajallaan, ja mallikin luokitteli hakijan ongelmatapaukseksi. Väärä positiivinen (FP) on tilanne, jossa lainanhakija B maksaa lainansa ajallaan, mutta malli luokitteli hänet epäonnistuneesti takaisin maksavien joukkoon. Väärä negatiivinen (FN) on tilanne, jossa A luokiteltiin onnistuneesti maksavien joukkoon, vaikka hän ei maksanutkaan lainaa ajallaan. Oikea negatiivinen (TN) tarkoittaa tilannetta, jossa B luokitellaan lainansa onnistuneesti takaisin maksavien joukkoon.

Oikeiden positiivisten suhde (TPR) saadaan kaavalla $\frac{TP}{TP+FN}$. Väärien positiivisten suhde saadaan kaavasta (FPR) $\frac{FP}{TN+FP}$. Näin ollen saadaan taulukon 4 mukainen luokittelumatriisi.

	Ennustettu positiivinen	**Ennustettu negatiivinen**
Oikea positiivinen	TP	FN
Oikea negatiivinen	FP	TN

Taulukko 4: Luokittelumatriisi.

Tästä taulukosta saadaan laskettua luokittelumalleja arvioitaessa paljon käytettävät käsitteet herkkyys ja tarkkuus.

4.1 Tarkkuus

Lähtökohtaisesti koneoppimismalleja arvioitaessa tarkastellaan erityisesti niiden tarkkuutta, joka kuvaa oikein luokiteltujen havaintojen osuutta kaikista havainnoista. Tarkkuuden rinnalla käytetään usein myös positiivista ennustetarkkuutta (taulukoissa lyhenteenä ennustetark.), joka kuvaa oikein luokiteltujen positiivisten osuutta kaikista mallin positiivisiksi luokittelemista havainnoista. Kuitenkaan luokittelumallin koulutuksen tuloksia ei voida arvioida pelkästään tarkkuudella, varsinkaan epäsymmetrisessä datassa [24]. Tämä johtuu siitä, että arvaamalla kaikki tulokset positiiviseksi datasta, jossa suurin osa on positiivisia, saadaan jo varsin hyvä tulos.

Tarkkuus saadaan kaavasta

$$Tarkkuus = \frac{TP + TN}{TP + TN + FP + FN},$$

jossa on hyödynnetty matriisin 4 arvoja. Positiivinen ennustetarkkuus taas saadaan kaavasta

$$positiivinen\, ennustetarkkuus = \frac{TP}{TP + FP}. \tag{14}$$

Mikäli positiivinen ennustetarkkuus on pieni, tällöin mallin positiiviseksi luokittelemista havainnoista suurin osa ovat vääriä positiivisia. Näin ollen malli ylireagoi. Jos positiivinen ennustetarkkuus on taas suuri, tällöin kaikki mallin luokittelemat positiiviset ovat myös oikeasti positiivisia.

Mallin ollessa positiivisten ennusteiden kanssa tarkka saatetaan puhua mallin olevan varovainen, vaikka se löytääkin oikeat positiiviset hyvin. Haittapuolena mallin ollessa positiivisesti ennustetarkka on se, että herkkyys saattaa laskea huomattavasti.

4.2 Herkkyys

Luokittelumallin herkkyys tarkoittaa kykyä löytää raja positiivisten ja negatiivisten tuloksien väliltä. Mikäli luokittelumallin herkkyysarvo on pieni, tällöin mallilta jää suuri määrä positiivisia tuloksia löytämättä. Näin ollen usein puhutaan varovaisesta luokittelumallista. Tämä voi johtaa ongelmallisiin tuloksiin luokittelumallia käyttäessä. Esimerkiksi lainanhakijoista saattaa jäädä huomaamatta erittäin huonosti lainansa takaisin maksavia. Jos luokittelumallin herkkyys on suuri, tällöin malli löytää lähes kaikki positiiviset. Tällöin FN on pieni ja puhutaan herkästä mallista. Mitä herkempi malli on, sitä paremmin se pystyy löytämään lainahakija, jotka ovat riski. Luokittelumallin herkkyys saadaan kaavasta

$$herkkyys = \frac{TP}{TP + FN},$$

jossa käytetään matriisia 4 hyödyksi.

4.3 ROC-AUC

Seuraava määritelmä perustuu lähteeseen [24], missä esitellään binääriluokittelun suorituskykymittari ROC-AUC, jonka avulla voidaan arvioida luokittelumallin suorituskykyä. Kyseinen mittari muodostuu kahdesta eri elementistä, ROC-käyrästä ja sen alle jäävästä alueesta.

ROC-käyrä on kaksiulotteinen käyrä, joka kuvastaa oikeiden positiivisten hyödyn ja väärien positiivisten haitan välistä vaihtokauppaa luokittelumallin tietyllä kynnysarvolla. Piste (0,0) tarkoittaa, että malli ei luokittele yhtään tapausta positiiviseksi. Käyrän ihanteellinen piste on (0,1), jolloin malli luokittelee ainoastaan oikeita positiivisia eikä yhtäkään väärää positiivista. Jos verrataan kahta eri pistettä ROC-käyrällä, parempi malli vastaa käyrää, joka kulkee mahdollisimman lähellä vasenta

reunaa ja yläosaa. Tällaisessa pisteessä malli luokittelee tulokset siten, että oikeiden positiivisten suhde on suurempi kuin väärien positiivisten suhde. ROC-käyrän yksi eduista on sen suhteellinen riippumattomuus luokkajakauman muutoksista, koska se perustuu suhteellisiin osuuksiin (TPR ja FPR). Tämä johtuu siitä, että ROC-käyrä lasketaan luokittelumatriisin perusteella saaduista suhteista 4. ROC-käyrä muodostuu, kun väärien positiivisten ja oikeiden positiivisten lukumäärät lasketaan luokittelumallille eri kynnysarvoilla.

AUC tarkoittaa aluetta käyrän alla. Tässä tapauksessa ROC-käyrän alle jäävää aluetta. Koska alue on pinta-alaa 1x1 neliössä, se saa minkä tahansa arvon nollan ja yhden väliltä. Mikäli luokittelumalli saa arvon 0.25, se on yhtä huono kuin satunnainen arvaus tutkielman aineistossa, jossa 25% on positiivisia. Kun taas AUC = 1, luokittelumalli onnistuu täydellisesti erottamaan positiiviset ja negatiiviset toisistaan. Huomioitava asia ROC-AUC-mittaria käyttäessä on se, että mittari kuvastaa mallin suoriutumista kokonaisuutena. Tästä seuraa se, että tietyllä kynnysarvolla toinen malli voi suoriutua huomattavasti paremmin vaikka ROC-AUC arvo olisi sama molemmille malleille. Näin ollen luokittelumallista kannattaa aina saada näkyviin tulokset jokaiselle kynnysarvolle luokittelumallin suoriutumisen tuloksista.

4.4 PR-AUC

Tässä määritelmässä on käytetty lähdettä [25]. Precision Recall Area Under The Curve (PR-AUC) mittari kuvastaa mallin kykyä löytää positiiviset havainnot eri luokittelukynnyksillä luokkajakauman ollessa epätasapainoinen. Mallin positiivinen ennustetarkkuus saadaan kaavasta $\frac{TP}{TP+FP}$ ja herkkyys kaavasta $\frac{TP}{TP+FN}$, nämä kaksi arvoa yhdistämällä saadaan lukuparit X ja Y koordinaatistoon samalla tavalla kuin ROC-käyrää muodostaessa. Näitä lukupareja saadaan lisää, kun mallin herkkyys ja tarkkuus lasketaan eri kynnysarvoilla. PR-AUC mittarin arvo satunnaiselle mallille ei ole 0,25 kuten ROC-AUC mittarille, vaan se saadaan positiivisten osuudesta aineistosta kaavalla

$$Baseline_{PR} = \pi = \frac{P}{P+N}.$$

Näin ollen PR-AUC voi saada todella matalia arvoja, vaikka malli olisikin erityisen hyvä, koska positiivisten osuus aineistosta π voi olla jo valmiiksi todella pieni. PR-AUC mittaria käytettäessä on huomioitava sen erityinen herkkyys luokkajakaumalle.

PR-AUC on ROC-AUC mittaria huomattavasti parempi vinoutuneessa datassa, koska PR-AUC korostaa mallin suorituskykyä datan pienemmässä luokassa. Tämä johtuu siitä, että ROC-AUC käyttää suhdelukuja, kun taas PR-AUC käyttää mallin tarkkuutta, jossa luokittelumallin antamat väärät positiiviset tai väärät negatiiviset korostuvat.

4.5 F1 ja F2

F1 on yksi kaikkein käytetyimmistä mittareista arvioimaan luokittelumalleja. Tässä määritelmässä käytetään lähdettä [26]. F1 mittari perustuu tarkkuuden ja herkkyyden harmoniselle keskiarvolle yhdellä kynnysarvolla. F1 tulos saadaan laskettua

kaavalla

$$F1 = 2 \cdot \frac{Tarkkuus \cdot Herkyys}{Tarkuus + Herkkyys} = \frac{2TP}{2TP + FP + FN}.$$

Kuten kaavasta huomataan, malli yhdistää kaksi olennaista luokittelun onnistumisen mittaria yhdeksi mittariksi. Näin ollen on tärkeää kuvata herkkyyden ja tarkkuuden tulokset myös omina lukuinaan F1 mittaria käytettäessä. F1 ei ole herkkä luokkajakaumalle, joten mittaria on erityisen hyvä käyttää tilanteissa, joissa niin negatiiviset tulokset kuin positiiviset tulokset ovat tuloksina merkityksellisiä.

F1-tulos ei ota ollenkaan huomioon mallin antamia oikeita negatiivisia. Tästä seuraa se, että F1 onnistuu paljon paremmin tilanteissa, joissa on tärkeämpää tunnistaa nimen omaan positiivisen alkiot verrattuna PR-AUC-mittariin. Suurimpana erona PR-AUC ja ROC-AUC mittareihin verrattuna F1-mittari ei muodosta tulosta pinta-alan mukaan, koska se laskee ainoastaan yhdellä kynnysarvolla tuloksen. Tästä johtuen F1-tulos voi vaihdella huomattavasti kynnysarvoa vaihtamalla. Näin ollen luokittelumallista saa kokonaisemman kuvan PR-AUC ja ROC-AUC mittareita käytettäessä.

F2-mittari on lähestulkoon sama kuin F1-mittari. F2-mittaria käytetään enemmän luokittelumalleissa, jotka koulutetaan epätasapainoisissa aineistoissa. Kaavaan tulee seuraavanlainen muutos

$$F2 = 5 \cdot \frac{Tarkkuus \cdot Herkyys}{4 \cdot Tarkuus + Herkkyys},$$

jolloin mittarin tarkoituksena on korostaa positiivisten havaintojen luokittelutodennäköisyyttä. Näin ollen F2 korostaa herkkyyttä nelinkertaisesti F1 verrattuna.

F2-mittaria tullaan käyttämään Logistisen Regressiomallin kanssa, koska se tuottaa suoraan kalibroituja todennäköisyys arvioita. Kun taas Random Forest- ja XGBoostluokittelumallit tuottavat mallien sisäisistä puista heijastuneita pistearvoja ja todennäköisyyksiä ilman kiinteää päätöskynnysarvoa.

4.6 Brier

Yksi ehdottomasti tunnetuimpia todennäköisyysennusteissa käytettävistä mittareista on Brier-mittari. Tämän mittarin esittämisessä on käytetty lähdettä [27]. Brier-mittarin kaava on muotoa

$$Brier = \frac{1}{n} \sum_{i=1}^{n} (p_i - y_i)^2,$$

jossa p_i on luokittelumallin antama todennäköisyys havainnon positiivisuudelle ja y_i on todellinen havainnon tulos. Tämä mittari mittaa siis keskimääräistä neliöityä eroa mallin antaman todennäköisyyden ja todellisen tuloksen välille. Näin ollen mitä pienemmän arvon malli saa, sen paremmin malli on onnistunut ja arvot vaihtelevat välille nolla ja yksi. Edes täydellinen malli ei voi antaa arvoa 0, koska harvoin alkuperäisessä aineistossa kaikki viitteet viittaisivat siihen, että todennäköisyydet olisivat vain ja ainoastaan ääripäissä. Brier-mittari lasketaan samalla kaavalla kuin

keskimääräinen neliövirhe (MSE), mutta Brier-mittaria käytetään nimen omaan binäärisiin ennusteisiin. Siitä on myös olemassa moniluokkaisiin luokitteluennusteisiin käytettävä malli.

Brier-mittari ei ole päätöskynnys riippuvainen samalla tavalla kuin F1, koska Brier kertoo suoraan todennäköisyyden luokittelumallin antamalle tulokselle. Luonteeltaan F1 ja Brier eroavat todella paljon. Brier kertoo mallin antaman todennäköisyyden ja oikeasti toteutuneen tilanteen välillä. Kun taas F1 katsoo suoraan luokittelun tulokset ja vertaa niiden välisiä eroja tietyillä kynnysarvolla.

5 Tulokset

Koneoppimismallien koulutus alkaa kun aineiston esikäsittely on saatu valmiiksi. Seuraavien mallien koulutuksessa on käytetty apuna ChatGPT kielimallia ratkaisemaan ongelmia, mutta kaikki ideat, ongelmien ratkaisut ja koulutuksen eteneminen on täysin kirjoittajan omia johtopäätöksiä.

5.1 Baseline-mallit

Aluksi koneoppimismalleista (Logistinen Regressio, Random Forest ja XGBoost) koulutetaan baseline-mallit, joista saadaan tärkeä lähtökohta lopullisia malleja varten tehtävistä hyperparametrien ja lopullisen päätöskynnyksen säätämiseen. Aineisto jaettiin ajallisesti harjoitusaineistoon (2007-2015), validointiaineistoon (2016-2017) ja testiaineistoon (2018) koulutusta varten. Testiaineisto pidetään koulutuksen ajan koskemattomana, kunnes lopulliset mallit ovat valmiit esittelyä varten. Kuten aikaisemmin tutkielmassa todetttin päätösmuuttujan harvinainen tapahtuma käännettiin positiiviseksi $y \leftarrow 1 - y$. Näin ollen positiivinen luokka on lainan takaisinmaksun laiminlyönti, mistä tutkielmassa ollaan kiinnostuneita.

Mallien harjoitusaineistojen esikäsittelyssä puuttuville numeerisille muuttujille tehtiin mediaani-imputointi ja Logistiselle Regressiomallille arvojen standardiskaalaaminen. Kategorisille muuttujille Random Forest- ja Logistiselle Regressiomalleille tehtiin yksiluokkainen binääriesitys harvinaisten luokkien yhdistämisellä ja XGBoost-mallille oridinaali-enkoodaus. Aineistojen luokkaepätasapaino huomioitiin Logistiselle Regressio- ja Random Forest-mallille muuttujalla $class\_weight$ ja XGBoost-mallissa muuttujalla $scale\_pos\_weight = \frac{N\_neg}{N\_pos}$.

Ensisijaisesti mallien todennäköisyydet arvioitiin validointiaineistolla käyttäen PR-AUC, Brier-loss ja ROC-AUC. Luokittelykynnys valittiin validointiaineiston perusteella maksimoimaan F1-tulosta.

Malleista XGBoost menestyi parhaiten koulutusajan ollessa 27 sekuntia. Logistisen Regressio-mallin koulutus kesti 21 sekuntia ja Random Forest-mallin koulutus 270 sekuntia.

Taulukossa 5 on esitetty baseline-mallien tulokset, joiden pohjalta jokaista mallia parannetaan. Nämä ovat tärkeät tulokset, jotta tiedetään mallien käyttökelpoisuus sekä voidaan arvioida mallien mahdollisesti haitallista ylioppimista.

Taulukossa 6 on esitetty validointiaineiston tulokset matriisi-muodossa.

	PR-AUC	π	ROC-AUC	Brier	Kynnys	Precision
LR	0,4399	0,256	0,7096	0,2313	0,500	0,3743
RF	0,4379	0,256	0,7068	0,1727	0,200	0,3697
XGBoost	0,4598	0,256	0,7187	0,2118	0,470	0,460

Taulukko 5: Baseline-mallien tulokset.

	TP	FP	FN	TN
LR	87 593	146 426	34 772	208 579
RF	88 615	151 068	33 750	203 937
XGBoost	87 708	142 611	34 657	212 394

Taulukko 6: Baseline-mallien luokittelumatriisi.

5.2 Mallien suorituskyvyn parantaminen

Baseline-mallien kouluttamisen jälkeen jokaiselle mallille haetaan parhaat hyperparametrit 20 prosentilla (165 873 lainahakemusta) harjoitusaineistosta. Jokaiselle mallille luotiin hyperparametrien hakuavaruus, joista parhaimpien hyperparametrien valitsemisessa käytettiin PR-AUC mittaria.

5.2.1 Logistinen Regressio

Logistiselle Regressiomallille suoritettiin satunnainen hyperparametrien haku (RandomizedSearchCV), jonka tavoitteena oli löytää parhaat mahdolliset hyperparametrit käyttäen kolminkertaista ristiinvalidointia, joka tarkoittaa 120 erilaista sovitusta 25% harjoitusaineistosta kerrallaan. Näin ollen malli saatiin arvioitua luotettavasti epätasapainoisessa aineistossa. Hyperparametrien arvioimisessa käytettiin epätasapainoisille aineistoille soveltuvaa PR-AUC-mittaria.

Logistiselle regressiomallille on olemassa erilaisia ratkaisualgoritmeja kuten *liblinear*, *lbfgs* ja *saga*. Näistä ratkaisualgoritmeista käytettiin *lbfgs*, *saga L2* ja *saga elasticnet* parhaan mallin hakemisessa siten, että jokaiselle ratkaisualgoritmille määritettiin omat hyperparametriavaruus, joista valittiin satunnaisesti 40 yhdistelmää. Ratkaisualgoritmia *liblinear* ei käytetty, koska se soveltuu erityisen pienille aineistoille, missä aineisto on tiheää.

Jokaiselle ratkaisualgoritmille annettin samat hyperparametrien arvojen hakuvälit, joista valittiin parhaat mahdolliset lopulliseen malliin. Regularisointivakion C arvo valittiin käyttäen log-uniform -jakaumaa, jolloin arvo saadaan logaritmisesti jakautuneelta väliltä 0,001 ja 100, jolloin hakualue kattaa heikon sekä voimakkaan säännöllistämisen alueet. Algoritmin pysäytysehto *tol* arvo, mikä määrittää pienimmän muutoksen, joka tappiofunktiossa voi olla jolloin iteraatiot lopetetaan. Kyseinen arvo haettiin logiritmisesti jakautuneelta väliltä 0,001 ja 0,1, koska optimoinnin tarkkuutta haluttiin säätää, jolloin myös laskenta-aikaa saatiin rajattua. Ratkaisualgoritmi *saga* käytti myös elasticnet-ratkaisussa l1-rangaistusta, jolloin hyperparametrin *l1_ratio* jatkuva arvo valittiin väliltä 0 ja 1, koska L1- ja L2-rangaistuksen yhdistelmän haluttiin määräytyvän vapaasti. Myös hakuavaruudessa käytettiin hyperparametreille *fit_intercept* arvoja True tai None. Lisäksi hyperparametrille *class_weight* käytettiin arvoa "balanced".

Edellä mainituista ratkaisualgoritmeistä parhaaksi valikoitui *saga*, joka tukee L1- ja elastic net -säännöllistämistä. Kyseinen ratkaisualgoritmi skaalaa myös hyvin isoille ja harvoille aineistoille. Regularisaatiovakion C arvoksi valikoitui 0,0635, joka on suhteellisen pieni arvo, mikä tarkoittaa vahvaa säännöllistämistä. Tämä oli ennakoitavissa kyseiselle aineistolle, koska se on korkea dimensioinen, suuri ja muutujat ovat suhteellisen korreloituneita keskenään. Hyperparametrin *l1_ratio* arvoksi valikoitui 0,8245, joka tarkoittaa L1-rangaistuksen säännöllistävän noin 82 prosenttia ja L2-rangaistus loput 18 prosenttia. L1-rangaistus tekee tässä tapauksessa aineistosa vielä harvemman, kun osa epäolennaisista piirteistä tiputetaan pois mallin kouluttamisessa. Hyperparametrin *tol* arvoksi valikoitui 0,00125, mikä on tiukka. Näin ollen saadaan lasketa-aikaa rajattua riittävästi säilyttäen edelleen riittävän hyvän optimoinnin tarkkuuden.

	PR-AUC	**ROC-AUC**	**Brier**	**F1**	**Kynnysarvo**
LR	0,4405	0,7097	0,2306	0,4915	0,51

Taulukko 7: Logistinen regressiomallin tulokset ennen kalibrointia.

Optimaalisten hyperparametrien ja parhaan ratkaisijan valitsemisen jälkeen mallin tulokset ovat hieman paremmat kuin baseline-mallin. Taulukosta 7 nähdään, että PR-AUC tulos 0,4405 on selvästi satunnaista luokittelijaa parempi. Tämä tarkoittaa, että malli kykenee erottelemaan riskiluokan paljon paremmin kuin sattuma. ROC-AUC tulos 0,7097 kertoo, että mallilla on yli 70% todennäköisyys antaa korkeampi todennäköisyys positiiviselle havainnolle kuin negatiiviselle havainnolle. F1-arvon ollessa korkea 0,4915 malli herkemmin painottaa riskiluokan havaitsemista, vaikkakin väärien positiivisten määrä on edelleen huomattava. Tämä tulos sopii kuitenkin aineistoon ja tutkielman tarkoitukseen paremmin, kun on tärkeämpää hallita kokonaisriskiä kuin yksittäisiä lainahakemuksia. Brier-tulos 0,2306 on kohtalainen, mikä kuvastaa realistisesti tutkielman aineistoa.

	Ennustetark. 1%	**Ennustetark. 5%**	**Ennustetark. 10%**
LR	0,6357	0,5765	0,5295

Taulukko 8: Logistisen regressiomallin ennustetarkkuus ennen kalibrointia.

Taulukosta 8 nähdään, että mallin ennustamista 1 prosentista korkeimman todennäköisyyden saaneista havainnoista 63,6% ovat oikeasti positiivisia. Mallin ennustamat 5 prosentin ryhmästä 57,7% ovat oikeasti positiivisa havaintoja ja korkeimmasta 10% yli puolet ovat positiivisa havaintoja. Tämä on hyvä tulos ja osoittaa, että malli kykenee järjestelemään havainnot oikeaan järjestykseen. Tutkielman aineistosta noin 25% on oikeasti positiivisia havaintoja, joten näin korkeat arvot ovat realistisen hyviä.

Tämän jälkeen malli kalibroitiin isotonic regressionin avulla validointiaineistolle, missä mallin antamien todennäköisyyksien ja todellisten havaintojen välille sovitetaan monotonisesti kasvava funktio. Isotonic regressiossa optimoidaan neliötappiota $\sum_i (y_i - g(s_i))^2$, jossa y_i on todellinen havainto, s_i on mallin antama pistemäärä ennen kalibrointia ja $g(s_i)$ kalibroitu todennäköisyys [28]. Kalibroinnin menestymisen mittarina käytettiin Brier-tulosta sekä odotettua kalibroinnin virhettä.

	PR-AUC	ROC-AUC	Brier	F2	Kynnysarvo
LR	0,4386	0,7099	0,17064	0,6590	0,133

Taulukko 9: Logistinen regressiomalli kalibroinnin jälkeen.

Logistisen regressiomallin kalibroinnin jälkeen saatiin taulukon 9 mukaiset tulokset. Mallin erottelukykyä mittaavat mittarit PR-AUC ja ROC-AUC pysyivät lähes ennallaan. Tämä oli oletettua, koska kalibrointi ei muuta mallin erottelu kykyä vaan mallin luotettavuutta, jolloin malli järjestää havainnot lähes samaan järjestykseen. Kalibroinnissa käytetty mittari Brier taas parani huimasti saaden 0,17 arvon, mikä osoittaa mallin antamien todennäköisyyksien vastaavan nyt paljon paremmin havaittujen tapahtumien suhteita. Optimaaliseksi luokittelukynnykseksi asettui 0,133, mikä on paljon matalampi kuin alkuperäisessä mallissa. Tämä tarkoittaa, että malli on erityisen herkkä luokittelemaan positiiviset havainnot. Mallin F2-tulos 0,6590 osoittaa mallin olevan hyvin virittynyt epätasapaoinoiseen aineistoon. Tämä on tärkeä tulos, kun väärän negatiivisen tuloksen kustannus on korkea.

	Ennustetark. 1%	Ennustetark. 5%	Ennustetark. 10%
LR	0,6344	0,5766	0,5295

Taulukko 10: Logistisen regressionin kalibroinnin jälkeiset positiiviset ennustetarkkuudet.

Taulukosta 10 nähdään, että mallin antama järjestys havainnoille pysyi vahvana. Tämä oli oletettavaa mallin antaman havaintojen järjestyksen pysyessä samana.

	PR-AUC	ROC-AUC	Brier	F1	Kynnysarvo
LR	0,4224	0,6915	0,1811	0,4766	0,1333

Taulukko 11: Logistisen regressiomallin testiaineiston tulokset.

Täysin valmiin ja viritetyn logistisen regressiomallin tulokset testiaineistolle ovat taulukossa 11. Kuten taulukossa PR-AUC tulos 0,4224 ja ROC-AUC tulos 0,6915 kertovat, että malli säilyttää hyvän kohtuullisen erottelukyvyn myös erillisessä datassa. Koska arvot ovat hieman alhaisempia kuin validointi vaiheessa, voidaan päätellä ettei malli ole ylisovittanut harjoitusaineistoa tai validointiaineistoa. Brier-tulos 0,1811 osoittaa mallin hyvää kalibroitumista ja arvo on realistinen uudessa datassa. Testiaineistolle otettiin F1-arvo, jotta tuloksia voidaan arvioida Random Forest- ja XGBoostmalliin. F1-tulos on lähestulkoon sama kuin validointivaiheen tulos. Tämä osoittaa mallin vakautta ja luotettavuutta näkemättömään aineistoon. Kynnysarvon ollessa näinkin matala 0,1333 tämä tarkoittaa, että mallilla pyritään saamaan eroteltua aineisto siten, että väärän negatiivisen kustannus olisi liian suuri. Näin ollen taulusta 13 nähdään, että malli löytää suurimman osan todellisista positiivisista (16 291) vaikka väärien positiivisten määrä on myös korke (34 377). Tämä kuitenkin on hyvä strategia tilanteisiin, joissa väärien negatiivisten määrä halutaan minimiin ja riskiennuste mahdollisimman tarkaksi.

Logistisen regressiomalli luokittelemat 1% korkeimmista todennäköisyyksistä omaavista havainnoista 52,7% on oikeasti positiivisia. Tämä on yli kaksikertaa aineiston

	Ennustetark. 1%	Ennustetark. 5%	Ennustetark. 10%
LR	0,5269	0,5192	0,4945

Taulukko 12: Logistisen regressiomallin testiaineiston positiiviset ennustetarkkuudet.

todellisen luokkajakauman ja näin ollen voidaan pitää hyvänä tuloksena. Yli puolet (52%) mallin ylimpään 5% luokittelemista havainnoista on positiivisa, joten malli kykenee säilyttämään tarkkuuden laajemminkin. Näin ollen mallin tuottamat riskiennusteet ja järjestykset ovat hyödyllisiä.

	TP	FP	FN	TN
LR	16 291	34 377	1 407	13 067

Taulukko 13: Valmiin logistisen regressiomallin luokittelumatriisi.

Testitulokset pysyivät lähestulkoon samoina validointiaineiston kanssa, mikä tarkoittaa mallin pystyvän säilyttämään hyvän luokittelun myös uusille aineistoille. Myös mallin antamat tulokset ovat hyvin luotettavia. Näin ollen mallia voidaan pitää käyttökelpoisena ja realistisena verrokkina XGBoost- ja Random Forestluokittelumalleille.

5.2.2 Random Forest

Random Forest -mallille suoritettiin samanlainen satunnainen hyperparametrien haku (RandomizedSearchCV), jota käytettiin logistiselle regressiomallille. Sen tavoitteena oli löytää parhaat mahdolliset hyperparametrit käyttäen viiden osan ristiinvalidointia. Hakeminen toteutettiin tekemällä satunnaisesti 15 erilaista hyperparametrien kombinaatiota, joista valittiin paras yhdistelmä validointituloksen PR-AUC -mittarin antamalla tuloksella.

Hakuavaruuteen hyperparametrille *n_estimators* valittiin kokonaislukuja satunnaisesti väliltä 200-600. Parhaassa mallissa puiden lukumääräksi valikoitui 492. Seuraavana haettiin hyperparametrin *max_depth* arvo väliltä 5-20, jonka arvo parhaassa mallissa oli 19. Pienimmän määrän havaintoja solmussa, jotta se voi jakautua määrittävä hyperparametri *min_samples_split* arvo haettiin väliltä 23-15 ja parhaassa mallissa arvoksi valikoitui 11. Maksimimäärää lehtisolmuissa oleville havainnoille rajaava hyperparametri *min_samples_leaf* arvo haettiin väliltä 2-10 ja parhaassa mallissa arvona oli 9. Hyperparametrille *max_samples* arvo haettiin liukulukuarvona väliltä 0,5-0,4 ja parhaaseen malliin valikoitui arvoksi 0,6736. Hyperparametrille *max_features* annettiin kolme mahdollista arvoa $\sqrt{p}$, $log_2(p)$ tai None, joista parhaaseen malliin valikoitui $log_2(p)$. Jakokriteerin *criterion* vaihtoehtoina olivat gini, entropy tai log_loss, joista valikoitui log_loss parhaaseen malliin. Näin ollen Random Forest -mallin parhaat hyperparametrit on saatu valittua.

	PR-AUC	ROC-AUC	Brier	F1	Kynnysarvo
RF	0,4428	0,7077	0,1888	0,4890	0,39

Taulukko 14: Random Forest-mallin tulokset ennen kalibrointia.

Hyperparametrien hakemisen jälkeen Random Forest antaa jo kohtalaisen hyvät tulokset, jotka on esitetty taulukossa 14 ja 15. PR-AUC 0,4428 on kohtalaisen hyvä tulos. Tällöin malli löytää epätasapainoisesta aineistosta positiiviset tapaukset paremmin kuin pelkästään arvaamalla noin 21,5% havainnoista positiiviseksi, mikä on positiivisten havaintojen osuus aineistossa. Tavoitteena olisi saada PR-AUC yli 0,5. ROC-AUC kertoo kuinka hyvin malli järjestää havainnot positiivisten saaden korkeammat pisteet kuin negatiivisten. Tulos 0,7077 on hyvä taso, kun 0,5 olisi sattumaa ja 1,0 olisi täydellinen tulos. Brier-loss tulos mittaa todennäköisyyksien tarkkuutta. Tuloksena 0,1888 on todella hyvä tulos, kun 0,0 olisi täydellinen malli. Kaikki arvot alle 0,2 ovat hyviä ja alle 0,15 olisi jo erinomainen. Taulukossa paras F1-tulos 0,4890 tarkkuuden ja herkkyyden kompromissin välillä saatiin kynnysarvolla 0,39. Tämä tulos on ok, mutta tässä ja seuraavissa mittareissa tärkeintä on selkeys luokittelumallin tarkoitusperässä. Mikäli mallilla halutaan löytää enemmän positiivisia havaintoja, voidaan kynnysarvoa laskea, mutta tällöin mallin tarkkuus laskee.

	Ennustetark. 1%	**Ennustetark. 5%**	**Ennustetark. 10%**
RF	0,6662	0,5796	0,5313

Taulukko 15: Random Forest-mallin positiiviset ennustetarkkuudet ennen kalibrointia.

Taulukossa 15 on esitetty Random Forest-mallin varmimmat prosenttiosuuksien tulokset. Tulos 0,6662 1% tarkoittaa, että mallin luokittelema 1% korkeimmista riski lainanhakijoista 2/3 ovat oikeasti positiivisia. Korkeimman 5% luokkaan kuuluvista 6/10 on oikeasti positiivisia havaintoja ja 10% 5/10. Tämä on hyvä tulos reaalimaailman ongelmaan, kun positiivisa havaintoja aineistosta on ainoastaan noin 25%.

	PR-AUC	**ROC-AUC**	**Brier**	**F1**	**Kynnysarvo**
RF	0,4397	0,7079	0,17084	0,4893	0,240

Taulukko 16: Random Forest-mallin tulokset kalibroinnin jälkeen.

Kalibroinnin jälkeinen Random Forest-malli on lopullinen, jonka tulokset on esitetty taulukossa 16. Brier-tulos laskeminen tarkoittaa, että mallin antamat todennäköisyydet vastaavat paremmin esiintymistiheyksiä todennäköisyysluokissa, koska luokittelumallin antamista 70% todennäköisyyksistä noin 70% on oikeita positiivisia. PR-AUC ja ROC-AUC pysyivät ennallaan, mikä on tarkoituskin, koska kalibrointi ei muuta havaintojen keskenäistä järjestystä. F1-mittarin opitmi pysyi ennallaa, mikä on hyvä merkki. Kynnysarvon laskeminen tarkoittaa, että malli pystyy ennustamaan positiivisen tuloksen pienemmällä todennäiköisyydellä. Tällä on yhteys Brier-tulokseen, koska mallin antamat tulokset ovat näin ollen luotettavampia.

	Ennustetark. 1%	**Ennustetark. 5%**	**Ennustetark. 10%**
RF	0,6646	0,5806	0,5316

Taulukko 17: Random Forest-mallin kalibroinnin jälkeiset positiiviset ennustetarkkuudet.

Tarkkuuksien taulukosta 19 huomataan, että positiiviset ennustetarkkuudet ovat pysyneet lähestulkoon samoina, mikä oli oletettavaa validointi aineistolle. Tulokset ovat edelleen hyvät.

	PR-AUC	**ROC-AUC**	**Brier**	**F1**	**Kynnysarvo**
RF	0,4260	0,6921	0,1815	0,4949	0,240

Taulukko 18: Random Forest-mallin tulokset testiaineistolle.

Random Forest-mallin antamat tulokset testiaineistolle taulukossa 18 ovat hyvät. PR-AUC pysyi lähestulkoon samana, mikä tarkoitaa mallin pystyvän erinomaiseen yleistettävyyteen. Tämä myös osoittaa, että malli ei yliopi harjoitus- tai validointiaineistoa. ROC-AUC tulos laski vähäsen validointiaineistoon verrattuna, mutta säilyttää hyvän tason. Tässä täytyy myös huomoida, että testidata saattaa sisältää hankalampia havaintoja. Brier-loss arvo on suurinpiirtein sama kuin validointiaineistolle, joten tämäkin on hyvä tulos. F1-tulos on myös lähestulkoon sama kuin validointiaineiston kanssa kynnysarvon ollessa 0.24. Nämä tulokset vahvistavat mallin hyvän yleistettävyyden.

	Ennustetark. 1%	**Ennustetark. 5%**	**Ennustetark. 10%**
RF	0,5791	0,5281	0,4988

Taulukko 19: Random Forest-mallin testiaineiston positiiviset ennustetarkkuudet.

Testiaineiston tarkkuuksien arvoissa on hiukan laskua validointiaineistoon verrattuna, mutta malli säilyttää edelleen saman vahvan priorisointikyvyn. Tämä tarkoittaa, että malli ei ainakaan ylioppinut harjoitus- tai validointiaineistoa ja pystyy yleistämään hyvin täysin uuteen aineistoon.

	TP	**FP**	**FN**	**TN**
RF	29 321	18 123	5 921	11 777

Taulukko 20: Valmiin Random Forest-mallin luokittelumatriisi.

Random Forest-malli löytää todellisita positiivisistä noin 2/3, mutta silti säilyttäen hienosti tasapainon herkkyyden ja positiivisen ennustetarkkuuden välillä.

5.2.3 Extreme Gradient Boosting

Extreme Gradient Boosting mallille tehtiin samanlainen satunnainen hyperparametrien haku, jonka tavoitteena oli löytää mahdollisimman hyvät hyperparametrit lopulliselle mallille. Hyperparametrien hakemisessa käytettiin harjoitusaineistoa ja vertailukohtana pidettiin baseline-mallia.

Hakuavaruus määritettiin seuraavalla tavalla. Hyperparametrilla *n_estimators* määritetään yläraja mallissa käytettävien puiden lukumäärälle ja siihen valittiin kokonaisluku 300 ja 900 väliltä. Parhaassa mallissa *n_estimators* sai arvon 754. Puun maksimisyvyyttä säätelevän hyperparametrin kokonaisluku *max_depth* valittiin 3 ja 8 väliltä, josta paras malli sai arvon 7. Puiden painokertoimen *learning_rate* arvo haettiin välilä 0,005 ja 0,2 ja parhaassa mallissa se sai arvon 0,01671. Jokaiselle

puulle annettavan koulutusaineistojen suuruutta määrittävä hyperparametrin *subsample* liukulukuarvo haettiin väliltä 0,6 ja 1,0, joka parhaassa mallissa on 0,8933. Puiden rakentamisessa käytettäviä ominaisuuksia rajoittavat hyperparametrien *colsample_bytree*, *colsample_bylevel* ja *colsample_bynode* liukulukuarvot haettiin väliltä 0,4 ja 1,0. Parhaassa mallissa arvot olivat *colsample_bytree* sai arvon 0,799, *colsample_bynode* sai arvon 0,625 ja *colsample_bylevel* sai arvon 0,938. Lehdissä olevien havaintojen lukumäärää rajoittava *min_child_weight* arvo haettiin välilät 0,1 ja 5, josta arvoksi valikoitui 1,899. Lisärangaistuksena uusien jakojen tekemiselle toimivan hyperparametrin *gamma* arvo haettiin väliltä 0,001-1,0, josta arvoksi valikoitui 0,0055. Rangaistusparametri L2 *lambda* ja L1 rangaistusparametri *alpha* arvot haettiin väliltä 0,001 ja 5, joista ne saivat arvoikseen 0.00898 ja 8,8625.

	PR-AUC	**ROC-AUC**	**Brier**	**F1**	**Kynnysarvo**
XGBoost	0,4582	0,7179	0,2141	0,4965	0,49

Taulukko 21: XGBoost-malli ennen kalibrointia.

Taulukosta 21 nähdään XGBoost -mallin ennen kalibrointia saamista tuloksista. PR-AUC-tulos 0,4582 on kohtalaisen hyvä tulos ja epätasapainoisessa aineistossa hyväksyttävä. PR-AUC tulos määrittää, kuinka hyvin malli tunnistaaa positiiviset tapaukset aineistosta. Malli pystyy selvästi erottelemaan paremmin kuin pelkällä sattumalla tehty erottelu. ROC-AUC tulos 0,7179 on vielä kaukana täydellisestä, mutta sitä pidetään hyvänä yleistasona. Tulos 0,7179 tarkoittaa, että keskimäärin malli asettaa satunnaisesti valitun positiivisen havainnon 72% korkeammalle, kuin satunnaisesti valitun negatiivisen havainnon. Brier -tulos 0,2141 on hyvä ja tarkoittaa, että mallin antamat arvot ovat kohtalaisen lähellä oikeita arvoja. Taulukossa F1-tulos ja kynnysarvo ovat toisiinsa liitännäiset. F1-tulos kertoo positiivisen ennustetarkkuuden (precision) ja herkkyyden välisestä tasapainosta. Näiden tuloksien pohjalta tämä XGBoost-malli toimii parhaiten keskellä eli ei liian agressiivisesti luokittele havaintoja positiiviseksi eikä liikaa jätä huomioimatta havaintojen ominaisuuksista saatavia viitteitä.

	Ennustetark. 1%	**Ennustetark. 5%**	**Ennustetark. 10%**
XGBoost	0,6998	0,6044	0,5474

Taulukko 22: XGBoost-mallin positiiviset ennustetarkkuudet ennen kalibrointia.

Taulukossa 22 nähdään, että mallin mukaan 1% eniten positiivisia havaintoja sisältää noin 70% oikeita positiivisia havaintoja. Tämä on hyvä tulos. Kuten taulukosta huomataan, mallin jatkavan hyvää luokittelua viiden prosentin kanssa, luokitellen siitä 60% oikein. Malli onnistuu säilyttämään hyvän luokittelun myös kymmenen prosentin luokkaan.

	PR-AUC	**ROC-AUC**	**Brier**	**F1**	**Kynnysarvo**
XGBoost	0,4553	0,7181	0,16869	0,4966	0,2460

Taulukko 23: XGBoost-malli kalibroinnin jälkeiset tulokset.

XGBoost-malli kalibroitiin samalla tyylillä kuin Random Forest -malli käyttäen isotonic regressiota ja tulokset ovat taulussa 23. Kalibroinnin jälkeen Brier-tulos koki huiman parannuksen, kun se tippui 0,16869, mikä tarkoittaa mallin ennustamien todennäköisyyksien vastaavan paljon paremmin todellisia havaintoja. Tämä on loistava osoitus mallin tarvitsevan kalibrointia. PR-AUC ja ROC-AUC pysyivät lähestulkoon samoina, joka on hyvä asia.

	Ennustetark. 1%	**Ennustetark. 5%**	**Ennustetark. 10%**
XGBoost	0,7006	0,6054	0,5477

Taulukko 24: XGBoost-mallin kalibroinnin jälkeiset positiiviset ennustetarkkuudet validointiaineistolle.

Taulukosta 24 nähdään, kun PR-AUC ja ROC-AUC pysyvät lähestulkoon samoina, pysyvät samoina myös mallin ennustamien ylimpien prosenttien ennusteet, koska kalibrointi ei vaihda mallin tekemien ennusteiden järjestystä.

	PR-AUC	**ROC-AUC**	**Brier**	**F1**	**Kynnysarvo**
XGBoost	0,4464	0,7054	0,1784	0,5041	0,260

Taulukko 25: XGBoost-mallin testiaineiston tulokset.

Testiaineistoon malli toimii loistavasti, kuten taulukosta 25 nähdään. PR-AUC pysyy lähestulkoon identtisenä, joten malli ei ole ylisovittanut validointiaineistoa. ROC-AUC laskee hieman, mikä on normaalia mallin yleistysvaiheessa. Brier pysyy käytännössä paikallaan, joten luokittelumallin antamat tulokset ovat luotettavia.

	Ennustetark. 1%	**Ennustetark. 5%**	**Ennustetark. 10%**
XGBoost	0,6267	0,5649	0,5226

Taulukko 26: XGBoost-mallin testiaineiston positiiviset ennustetarkkuudet.

XGBoost-luokittelumallin testiaineiston tarkkuudet ovat taulukossa 26. Taulukosta nähdään, että mallin antamat tarkkuudet laskevat hyvin vähän, joka on normaalia mallin yleistämisessä. Kuitenkaan tarkkuuksien lasku on täysin hyväksyttävissä rajoissa.

	TP	**FP**	**FN**	**TN**
XGBoost	30 718	16 726	6 096	11 602

Taulukko 27: Valmiin XGBoost-mallin luokittelumatriisi testiaineistolle.

Taulukosta 27 nähdään, että mallin todellisten positiivisten suhde on noin 65% ja vielä tärkeämpää, että väärien positiivisten suhde on ainoastaan 35%. Epätasapainoisessa aineistossa tämä on todella hyvä kompromissi, kunhan tarkkuus saa korkeita arvoja.

5.3 Mallien vertailu

Kuten edellisestä kappaleesta huomataan mallien oppineen harjoitusaineistosta hyvin sekä validointiaineiston kalibroinnista saadut lisähyödyt olivat tarpeelliset. Jokaisella mallilla on omat vahvuutensa ja kuten oli oletettavaa, että XGBoost- ja Random Forestmallit olivat aika samanlaisia, kun taas logistinen regressiomalli oli eroavat.

Kokonaisuutena XGBoostmalli tarjoaa parhaan kokonaisluokittelutason PR-AUC ja ROC-AUC mittareilla. Logistinen regressiomalli taas onnistuu parhaiten kaliborinnissa, mistä seuraa realistisimmat todennäköisyydet. Random Forest jää kaikissa muissa mittareissa XGBoosti- ja Logistisen regressiomallin jälkeen paitsi F1-tulos oli hieman parempi kuin Logistisen Regressiomallin.

XGBoostmallin luokittelemat precisionit 1% ja 5% ovat selvästi paremmat kuin Random Forest- tai Logistisen Regressiomallin. Tämä osoittaa XGBoostmallille ylivoimaisuutta, kun pitää erotella riskisimmät lainanhakijat. Mallin ehdoton hyvä puoli on, että se kykenee löytämään muuttujien välisiä monimutkaisiakin riippuvuuksia. Näin ollen XGBoostmalli soveltuisi tällaissenaan käytettäväksi ennaltaehkäisevään riskienhallintaan ja petosten torjuntaan, mutta vaatii vielä tarkemman kalibroinnin juuri kyseiseen käyttötarkoitukseen.

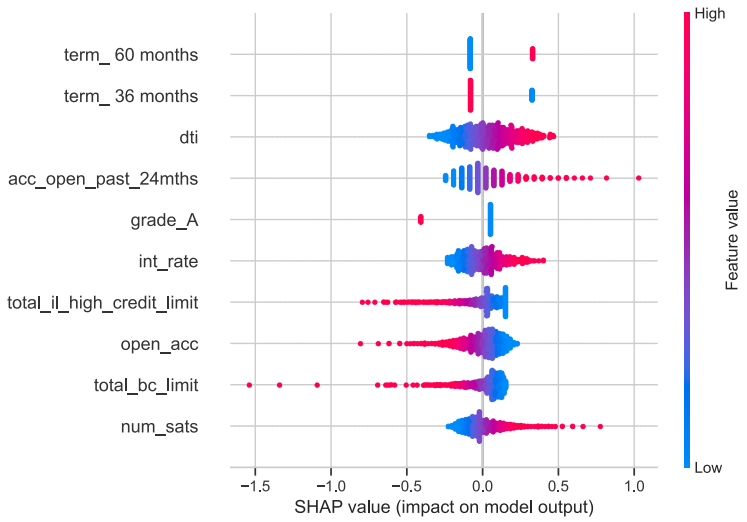
Logistinen regressiomalli on vakaa, helposti kalibroitava ja tulkittava luokittelumalli. Tällaisenaan kyseinen malli sopisi hyvin riskienhallintaan ja päätöksen tekoon, jossa tarvittaisiin läpinäkyviä sovelluksia ja ymmärrystä miten luokittelu tapahtui.

Random Forest-malli on raskas ja vakaa. Tällaissenaan mallia voitaisiin käyttää alustaviin lainanhakijoiden luokitteluun, joissa saaduilla todennäköisyyksillä ei tehdä suoraan päätöstä.

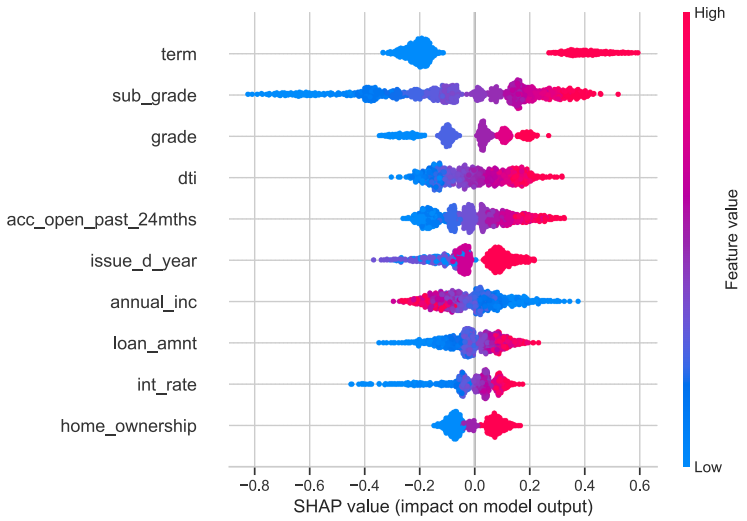
5.3.1 SHAP-Analyysi

Malleja arvioitaessa niiden tulkittavuus eli muuttujien vaikutukset päätökseen ovat avainasemassa. Tätä varten malleille tehtiin SHAP-analyysi, jossa selviää selittävien muuttujien vaikutus mallin tekemään ennusteeseen. SHAP-kuvaajissa x-akselin arvo kertoo, kuinka paljon muuttuja siirtää mallin ennustetta positiiviseen tai negatiiviseen suuntaan. Kuvaajalla positiiviset arvot kasvattavat takaisin maksun ennustetta, kun taas negatiivinen arvo laskee lainan takaisin maksun onnistumisen todennäköisyyttä. Kuvaajalla olevien pisteiden värit kertovat muuttujien todellisen arvon suuruudesta datassa. Y-akselilla on ainoastaan lueteltuna kymmenen tärkeintä muuttujaa satunnaisessa järjestyksessä.

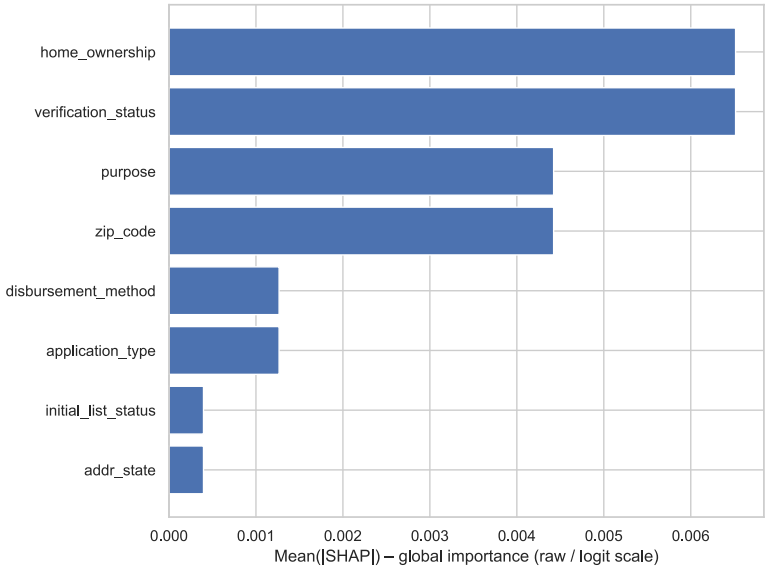
Kuvaajassa 5.3.1 on Random Forest -mallin tärkeimmät muuttujat. X-akselin arvot kertovat absoluuttisen arvon jokaiselle muuttujalle, mikä tarkoittaa sitä, että arvo ei ota kantaa vaikuttaako kyseinen muuttuja positiivisesti vai negatiivisesti ennusteeseen. Random Forest -mallin tekijät ovat kuten oletettua linjassa LendingClubin jo tekemän arvion kanssa. Tämän seurauksen muuttuja *grade* on selvästi yksi määrittävistä tekijöistä. Shap-analyysin pohjalta luokka A on pienemmän riskin laina, kuten LendingClub oletti. Seuraavaksi esiin nousee laina-aikaan liittyvät muuttujat *term_ 60* ja *term_ 36*. Mallin mukaan pidemmän laina-ajan myötä tulee suurempi riski ettei lainaa makseta ajallaan takaisin, kun taas lyhyemmällä laina-ajalla laina tulee takaisin maksetuksi. Velkaantumisasteesta kertova muuttuja *dti* esiintyy



Kuva 3: Logistisen regressiomallin tärkeimmät muuttujat.



Kuva 4: XGBoost-mallin tärkeimmät muuttujat.



Kuva 5: Random Forest-mallin tärkeimmät muuttujat.

keskeisenä tekijänä. Muuttujan *dti* saadessa korkeita arvoja lainantakaisin maksuun liittyvä riski on selvästi korkeampi.

Mallissa korkotaso *int_rate* nousee myös yhdeksi päätekijöistä, jolloin korkea korkotaso lisäsi riskiä. Myöskin luottorajoihin liittyvät muuttujat *total_il_high_credit_limit* ja *total_bc_limit* ovat molemmat negatiivisesti vaikuttaneet riskiin, jolloin suurempi luottoraja viittaa vahvemmasta maksukyvystä.

Myös aikaisempi luottohistoria nousee merkittäväksi tekijäksi. Mitä suurempi määrä oli hoidettuja tilejä *num_sats* pienensi takaisinmaksuun liittyvää riskiä. Muuttuja *acc_open_past_24mths* kuvastaa avattujen luottojen määrää viimeiseltä kahdelta vuodelta, mikä selvästi nostatti riskiä.

Kuvassa 5.3.1 nähdään, että XGBoost-malli antaa samanlaiset tulokset kuin logistinen regressiomalli, mutta havainnot ovat syvemmälle ulottuvia ja monipuolisemmat. XGBoost-mallin mukaan lainan keston, velkaantumisasteen ja luottoluokituksen muuttujat ovat tärkeimpiä tekijöitä. XGBoost-mallin ei-lineaarinen rakenne mahdollistaa muuttujien monimutkaisempien vuorovaikutuksien havaitsemisen. Muuttujat *grade* on korostunut myös tässä tuloksessa. Samalla tavalla kuin logistisessa regressiossa koron suuruus *int_rate* nostaa myös riskiä. XGBoost-mallissa korkeat vuositulot *annual_income* ja omistusasuminen *home_ownership* liittyivät selvästi pienempään riskiin.

Vaikka velkaantumisasteen muuttuja *dti* on edelleen esillä, mutta XGBoost-mallin analyysissä sen vaikutus ei ole täysin lineaarinen. Kuvasta nähdään, että vähäinen velkaantuminen ei vielä automaattisesti kasvata riskiä, kun taas korkeat arvot tekevät sen jyrkästi. Samanlaisen havainnon voi tehdä myös lainasumman muuttujan *loan_amnt* kanssa, kun suuret lainasummat nostattavat riskiä suuresti. Lisäksi on havaittavissa lainan myöntämisvuodella *issue_d_year* olevan merkitystä, joka voi liittyä talouden eri sykleihin.

Kuvassa 5.3.1 nähdään Random Forest -mallin merkittävimmät muuttujat. Tu-

loksesta nähdään, että muuttujat *home_ ownership* ja *verification_ status* ovat suurin osa koko mallin selittäjistä. Näiden jälkeen merkittävimmät muuttujat ovat *purpose* sekä *zip_ code*. Eroavaisuutena huomataan, että Random Forest -mallin tärkeimpinä muuttujina ei esiinny LendingClubin valmiita lainariskiluokkia.

Edellä esitetyissä analyyseissä pitää ottaa huomioon, että numeeriset arvot logistisessa regressiomallissa skaalattiin StandardScalerilla, mutta binäärisiä muuttujia ei. Näin ollen kertoimien suuruus ei ole niiden tärkeys suoraan. Myöskin kaikkien mallien antamissa tuloksissa muuttujat *grade* ja *sub_ grade* voivat peittää alleensa muiden muuttujien itsenäisiä vaikutuksia, kun kyseiset muuttujat ovat koostettu muista muuttujista.

Puumalleille SHAP-analyysi toteutettiin TreeExplainer-menetelmällä käyttäen testiaineistosta 1000 havainnon otosta, jotta laskenta-aika ei kasvaisi liian suureksi.

6 Yhteenveto ja johtopäätökset

Tutkielman päätavoitteena oli selvittää kolmen koneoppimismallin eroja ja soveltuvuutta LendingClub pankin lainahakemusten analysointiin ja päätöksentekoon. Malleina käytettiin logistinen regressiomallia, Random Forest -mallia ja Extreme Gradient Boosting -mallia.

Tulokset, jotka saatiin jokaiselle luokittelumallille, osoittivat, että jokainen malli saatiin koulutettua ilman ylikouluttamista harjoitus- tai validointiaineistolle tapahtunut. Jokaisen mallin vahvuudet ja heikkoudet saatiin selvästi esiin.

Odotetusti logistinen regressiomalli osoittautui yksinkertaisimmaksi ja selkeimmäksi malliksi tulkita. Se osoittautui hyväksi lähtökohdaksi tarjoten parhaan läpinäkyvyyden yksittäisten muuttujien vaikutuksista lainapäätöksiin. Tämä tarkoittaa logistisen regressiomallin olevan paras näistä kolmesta mallista, mikäli päätöksen teossa tarvitaan tiukkoja sääntelyvaatimuksia tai mallin tulee olla helposti auditoitavissa.

Random Forest-malli tarjoaa paremman kyvyn mallintaa ei-lineaarisia riippuvuuksia. Mallista saadut tulokset olivat selvästi parempia kuin Logistisen Regressiomallin, mutta mallin tulkittavuus väheni suuresti. Tulokset osoittivat Random Forest-mallin yleistävän hyvin, johtuen mallin päätöspuuryhmien keskiarvoistamisesta. Näin ollen Random Forest on erinomainen vaihtoehto, jos tarvitaan tasapainoa tarkkuuden ja yleistettävyyden välillä.

XGBoost-malli oli näistä kolmesta tarkin PR-AUC- ja ROC-AUC-mittareilla mitattuna. Mallin iteratiivinen gradienttivahvistus teki mallista kykeneväisen havaitsemaan monimutkaisia yhteyksiä eri muuttujien välillä, mikä tarkoittaa mallin olevan erinomainen vaihtoehto kustannusherkkiin tilanteisiin tai epätasapainoiseen aineistoon. Mallin haittapuolena on sen monimutkaisuus ja huomattava kalibrointitarve. Malli vaatii tarkkaa analyysia sen tuloksista.

Tutkielman tulokset vahvistivat ennakkokäsitystä siitä, että jos tarkoituksena on saada nopea ja selkeä malli, jonka tulokset ovat luotettavia, paras valinta on logistinen regressiomalli. Mikäli halutaan saada erityisen tarkka malli joka kykenee monimutkaisiin yhtäläisyyksiin ja mallin läpinäkyvyydestä voidaan karsia, tällöin valinnaksi kannattaa ottaa XGBoost-luokittelumalli.

Tutkielmasta huomattiin myös mallin valinnan olevan ei vain tekninen vaan myös strateginen. Tämä johtuu siitä, että jokaisella mallilla on erityispiirteensä ja heikkoutensa. Kuitenkin mallin valintaa tehdessä täytyy löytää tasapaino riskinsietokyvyn ja läpinäkyvyyden välille.

Rahoitussektorilla koneoppimismallien käyttäminen on kasvanut eksponentaalisesti viimeisten vuosien aikana [29]. Erityisesti mallien käyttäminen luottoriskien analysoimisessa on ollut yksi merkittävimmistä sovellusalueista, jossa käyttäminen on yleistynyt. Koneoppimismallien ovat osoittaneet ylivertaisuuttaan massiivisista ja monimutkaisista aineistoista tehtävillä päätelmillä, joihin perinteiset tilastolliset menetelmät eivät olisi pystyneet yhtä tehokkaasti.

Lisäksi kasvavan laskentatehon myötä yhä useampi malli perustuu syväoppimiseen. Koneoppimismallien yleistyminen vaatii niihin luotettavuuden lisääntymistä, joka tulee mallien käyttämisen ja ajan myötä, sekä riittävän hyvää datan laatua, joka ei ole itsestään selvyys. Artikkelissa [29] mainitaan myös, että pääsääntöisesti

rahoitussektorilla halutaan mahdollisimman selittäviä malleja, jotka kykenevät arvoimaan riskejä läpinäkyvästi. Näin ollen yksinkertaisemmat ja selitettävät mallit, kuten logistinen regressiomalli, tulevat säilyttämään paikkansa vielä pitkään.

Jatkotutkimuksissa olisi hyödyllistä vertailla XGBoost-luokittelumallin, logistisen regressiomallin ja syväoppimismenetlmien eroja aikasarja-aineistoissa, kun laskentatehot kasvavat entisestään yrityksillä on varaa kouluttaa monimutkaisempia malleja. Tämä mahdollistaisi enemmän dynaamisia malleja ja yliajan vaihtuvien trendien helpompaa tunnistamista. Toinen jatkotutkimus voisi liittyä samoille malleille ja samankaltaiselle aineistolle, mutta ilman tietyn yrityksen näkökulmaa ja arviota lainahakemuksista. Toinen näkökulma samasta aineistosta ja samoilla malleilla olisi luokitella lainahakemukset riskiluokkiin lainahakemuksesta saatavien tietojen perusteella.

Viitteet

[1] B. Berendt and S. Schulze, "Five Steps to Text Mining in Biomedical Literature," in *Proceedings of the 15th Euro- pean Conference on Machine Learning and the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, vol. 2168. Springer, Berlin, Heidelberg, 2001, pp. 47–50.

[2] O. F. Ayilara, L. Zhang, T. T. Sajobi, R. Sawatzky, E. Bohm, and L. M. Lix, "Impact of missing data on bias and precision when estimating change in patient-reported outcomes from a clinical registry," *Health and Quality of Life Outcomes*, vol. 17, no. 1, p. 106, Jun. 2019. [Online]. Available: https://doi.org/10.1186/s12955-019-1181-2

[3] H. Kang, "The prevention and handling of the missing data," *kja*, vol. 64, no. 5, pp. 402–406, May 2013, publisher: Korean Society of Anesthesiologists. [Online]. Available: http://dx.doi.org/10.4097/kjae.2013.64.5.402

[4] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, B. Mphago, and O. Tabona, "A survey on missing data in machine learning," *Journal of Big Data*, vol. 8, no. 1, p. 140, Oct. 2021. [Online]. Available: https://doi.org/10.1186/s40537-021-00516-9

[5] P. Rodríguez, M. A. Bautista, J. Gonzàlez, and S. Escalera, "Beyond one-hot encoding: Lower dimensional target embedding," *Image and Vision Computing*, vol. 75, pp. 21–31, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885618300623

[6] F. Pargent, F. Pfisterer, J. Thomas, and B. Bischl, "Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features," *Computational Statistics*, vol. 37, no. 5, pp. 2671–2692, Nov. 2022. [Online]. Available: https://doi.org/10.1007/s00180-022-01207-6

[7] T. Johnson, A. J. Liu, S. Raza, and A. McGuire, "A Comparison of Modeling Preprocessing Techniques," 2023, version Number: 2. [Online]. Available: https://arxiv.org/abs/2302.12042

[8] J. M. H. Pinheiro, S. V. B. de Oliveira, T. H. S. Silva, P. A. R. Saraiva, E. F. de Souza, L. A. Ambrosio, and M. Becker, "The Impact of Feature Scaling In Machine Learning: Effects on Regression and Classification Tasks," 2025, version Number: 2. [Online]. Available: https://arxiv.org/abs/2506.08274

[9] V. Carlo, C. Francesco, G. Niccoló, and B. Michele, "Can credit scoring models effectively predict small enterprise default? Statistical evidence from Italian firms," Tinbergen Institute, Amsterdam and Rotterdam, Tech. Rep., Oct. 2008.

[10] J. S. Cramer, "The Origins of Logistic Regression," Tinbergen Institute, Amsterdam and Rotterdam, Tinbergen Institute Discussion Paper 02-119/4, 2002. [Online]. Available: https://hdl.handle.net/10419/86100

[11] E. Boateng and D. A. Abaye, "A Review of the Logistic Regression Model with Emphasis on Medical Research," *Journal of Data Analysis and Information Processing*, vol. 7, no. 4, pp. 190–207, Oct. 2019. [Online]. Available: https://doi.org/10.4236/jdaip.2019.74012

[12] D. Pregibon, "Logistic Regression Diagnostics," *The Annals of Statistics*, vol. 9, no. 4, Jul. 1981. [Online]. Available: https://projecteuclid.org/journals/annals-of-statistics/volume-9/issue-4/Logistic-Regression-Diagnostics/10.1214/aos/1176345513.full

[13] D. Dey, M. S. Haque, M. M. Islam, U. I. Aishi, S. S. Shammy, M. S. A. Mayen, S. T. A. Noor, and M. J. Uddin, "The proper application of logistic regression model in complex survey data: a systematic review," *BMC Medical Research Methodology*, vol. 25, no. 1, p. 15, Jan. 2025. [Online]. Available: https://doi.org/10.1186/s12874-024-02454-5

[14] X. Zou, Y. Hu, Z. Tian, and K. Shen, "Logistic Regression Model Optimization and Case Analysis," in *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, 2019, pp. 135–139.

[15] T. Rymarczyk, E. Kozłowski, G. Kłosowski, and K. Niderla, "Logistic Regression for Machine Learning in Process Tomography," *Sensors*, vol. 19, no. 15, p. 3400, Aug. 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/15/3400

[16] C. J. Greenwood, G. J. Youssef, P. Letcher, J. A. Macdonald, L. J. Hagg, A. Sanson, J. Mcintosh, D. M. Hutchinson, J. W. Toumbourou, M. Fuller-Tyszkiewicz, and C. A. Olsson, "A comparison of penalised regression methods for informing the selection of predictive markers," *PLOS ONE*, vol. 15, no. 11, p. e0242730, Nov. 2020. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0242730

[17] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[18] X. Chen, D. Yu, and X. Zhang, "Optimal Weighted Random Forests," *Journal of Machine Learning Research*, vol. 25, no. 320, pp. 1–81, 2024. [Online]. Available: http://jmlr.org/papers/v25/23-0607.html

[19] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug. 1996. [Online]. Available: https://doi.org/10.1007/BF00058655

[20] P. Probst, M. N. Wright, and A. Boulesteix, "Hyperparameters and tuning strategies for random forest," *WIREs Data Mining and Knowledge Discovery*, vol. 9, no. 3, p. e1301, May 2019. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1301

[21] 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*, ser. KDD '16. New York, NY, USA: Association

for Computing Machinery, 2016, pp. 785–794, event-place: San Francisco, California, USA. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[22] J. Mushava and M. Murray, "Flexible loss functions for binary classification in gradient-boosted decision trees: An application to credit scoring," *Expert Systems with Applications*, vol. 238, p. 121876, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423023783

[23] V. Verma, "Exploring Key XGBoost Hyperparameters: A Study on Optimal Search Spaces and Practical Recommendations for Regression and Classification," *International Journal of All Research Education and Scientific Methods*, vol. 12, no. 10, pp. 2455–6211, 2024. [Online]. Available: https://www.ijaresm.com/uploaded_files/document_file/Vibhu_VermanPWN.pdf

[24] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786550500303X

[25] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," *PLOS ONE*, vol. 10, no. 3, pp. 1–21, Mar. 2015, publisher: Public Library of Science. [Online]. Available: https://doi.org/10.1371/journal.pone.0118432

[26] P. Christen, D. J. Hand, and N. Kirielle, "A Review of the F-Measure: Its History, Properties, Criticism, and Alternatives," *ACM Comput. Surv.*, vol. 56, no. 3, Oct. 2023, place: New York, NY, USA Publisher: Association for Computing Machinery. [Online]. Available: https://doi.org/10.1145/3606367

[27] L. Hoessly, "On misconceptions about the Brier score in binary prediction models," Jul. 2025, arXiv:2504.04906 [stat]. [Online]. Available: http://arxiv.org/abs/2504.04906

[28] U. Johansson, T. Löfström, H. Linusson, and H. Boström, "Efficient Venn predictors using random forests," *Machine Learning*, vol. 108, no. 3, pp. 535–550, Mar. 2019. [Online]. Available: http://link.springer.com/10.1007/s10994-018-5753-x

[29] H. Gao, G. Kou, H. Liang, H. Zhang, X. Chao, C.-C. Li, and Y. Dong, "Machine learning in business and finance: a literature review and research opportunities," *Financial Innovation*, vol. 10, no. 1, p. 86, Sep. 2024. [Online]. Available: https://doi.org/10.1186/s40854-024-00629-z