Miika Myötänen

Ohjelmistovirheiden ennustaminen koneoppimismenetelmien avulla

Tekniikan ja innovaatiojohtamisen yksikkö
Tietojärjestelmätiede kandidaatintutkielma
Tietotekniikan ja tuotantotalouden ohjelma

Vaasa 2026

VAASAN YLIOPISTO
Tekniikan ja innovaatiojohtamisen yksikkö
Tekijä: Miika Myötänen
Tutkielman nimi: Ohjelmistovirheiden ennustaminen koneoppimismenetelmien avulla
Tutkinto: Kauppatieteiden kandidaatti
Koulutusohjelma: Tietotekniikan ja tuotantotalouden ohjelma
Opintosuunta: Tietojärjestelmätiede
Työn ohjaaja: Juho-Pekka Mäkipää
Valmistumisvuosi: 2026 **Sivumäärä:** 35

TIIVISTELMÄ:
Tämän kandidaatintutkielman tavoitteena on tarkastella ohjelmistovirheiden ennustamista koneoppimismenetelmien avulla kirjallisuuskatsauksen muodossa. Ohjelmistokehityksessä virheiden aikainen tunnistaminen on tärkeää, koska virheiden korjaaminen tulee yleensä kalliimmaksi projektin myöhemmissä vaiheissa. Tässä tutkielmassa selvitetään, millaisia koneoppimismenetelmiä ohjelmistovirheiden ennustamiseen on käytetty, millaisia tuloksia niillä on saavutettu ja millaisia haasteita niiden käyttöön liittyy.
Kirjallisuuskatsauksessa tarkastellaan sekä perinteisiä valvottuja koneoppimismalleja että syväoppimismenetelmiä. Lisäksi käsitellään luokkaepätasapainon vaikutusta ennustemallien toimintaan sekä sitä, miten menetelmiä voidaan hyödyntää teollisissa ympäristöissä. Kirjallisuuden perusteella voidaan todeta, että menetelmien suorituskyky riippuu paljon datan laadusta, ominaisuuksien valinnasta ja käytetyistä arviointimittareista. Syväoppimismallit ovat viime vuosina yleistyneet, mutta niiden käyttöön liittyy myös haasteita, kuten suuri datantarve ja heikompi selitettävyys.
Yhteenvetona voidaan todeta, että ohjelmistovirheiden ennustaminen koneoppimisen avulla on kehittyvä tutkimusalue, jossa eri menetelmien toimivuus vaihtelee tilanteen ja aineiston mukaan. Erityisesti datan esikäsittelyllä ja epätasapainoisen datan hallinnalla on merkittävä vaikutus ennustustuloksiin.

AVAINSANAT: ohjelmistovirheet, koneoppiminen, syväoppiminen, ohjelmistotekniikka, neuroverkot, ennustaminen, datan esikäsittely

Sisällys

1 Johdanto 4

1.1 Tutkimuksen tavoite 5

1.2 Työn rakenne 5

2 Kirjallisuuskatsaus tutkimusmenetelmänä 7

2.1 Kirjallisuuskatsauksen tavoite ja tarkoitus 7

2.2 Kirjallisuuskatsauksen tyypit 7

2.3 Narratiivinen kirjallisuuskatsaus 8

2.4 Kirjallisuuskatsauksen toteuttaminen 9

3 Aihepiirin tausta ja keskeiset käsitteet 13

3.1 Ohjelmistomittarit ja metriikat 14

3.2 Ennustemallien arviointimittarit 15

3.3 Datasetit ja projektien välinen yleistettävyys 17

4 Aiempi tutkimus ohjelmistovirheiden ennustamisesta 19

4.1 Valvotut koneoppimismenetelmät 20

4.2 Neuroverkot ja syväoppiminen 21

4.3 Epätasapainoisen datan ongelma 22

4.4 Teolliset sovellukset 23

5 Diskussio 25

5.1 Vastaukset tutkimuskysymyksiin 25

5.2 Menetelmien vertailu 27

5.3 Tutkimusaukot ja jatkotutkimus 29

5.4 Yhteenveto diskussiosta 30

6 Johtopäätökset ja pohdinta 31

Lähteet 34

1 Johdanto

Ohjelmistojen laatua ja virheettömyyttä pidetään tärkeänä osana ohjelmistokehitystä, koska ohjelmistovirheet voivat aiheuttaa merkittäviä toiminnallisia ja taloudellisia haittoja. Virheiden löytäminen mahdollisimman varhaisessa vaiheessa on usein kustannustehokkaampaa kuin myöhemmässä vaiheessa tehtävät korjaukset, mikä on yksi syy siihen, miksi ohjelmistovirheiden ennustaminen on noussut aktiiviseksi tutkimusalueeksi. (Malhotra, 2015)

Koneoppimismenetelmät tarjoavat lupaavia työkaluja ohjelmistovirheiden ennustamiseen, koska ne voivat tunnistaa piirteitä ja malleja aiemmista ohjelmistoprojekteista. Systemaattisissa kirjallisuuskatsauksissa on raportoitu, että erilaisia koneoppimistekniikoita käytetään laajasti virheiden ennustamisessa (Malhotra, 2015). Toisaalta uusia lähestymistapoja, kuten syväoppimista, on otettu käyttöön, koska ne pystyvät käsittelemään monimutkaisempia piirteitä ja riippuvuuksia datassa (Wang ym., 2018; Zhang ym., 2021).

Vaikka syväoppimismallit ovat herättäneet kiinnostusta, niiden käyttöön liittyy myös haasteita. Esimerkiksi niiden vaatima suuri datamäärä ja tulkittavuuden puute voivat rajoittaa soveltamista käytännön ympäristöissä. Lisäksi eri projektien välinen vaihtelu, datan laatu ja ominaisuuksien valinta voivat vaikuttaa merkittävästi koneoppimismallien toimintaan. Näistä näkökohdista on raportoitu myös teollisiin sovelluksiin keskittyvissä katsauksissa (Madeyski & Stradowski, 2023).

Tässä tutkielmassa tarkastellaan sekä perinteisiä koneoppimismalleja että syväoppimistekniikoita ja niiden soveltamista ohjelmistovirheiden ennustamisessa.
Tämän työn tavoitteena on tarkastella kirjallisuuskatsauksen avulla ja vastata seuraaviin tutkimuskysymyksiin:

- Millaisia koneoppimismenetelmiä ohjelmistovirheiden ennustamisessa käytetään?
- Millaisia tuloksia eri menetelmillä on saavutettu?

- Mitkä tekijät vaikuttavat ennustemallien suorituskykyyn?
- Mitkä ovat menetelmiin liittyviä haasteita?
- Miten menetelmiä voidaan hyödyntää käytännön ohjelmistokehityksessä?

1.1 Tutkimuksen tavoite

Tämän kandidaatintutkielman tavoitteena on muodostaa narratiivinen kirjallisuuskatsaus ohjelmistovirheiden ennustamisesta koneoppimismenetelmien avulla. Työssä pyritään kokoamaan ja jäsentämään aiempaa tutkimustietoa siten, että muodostuu selkeä kokonaiskuva siitä, millaisia menetelmiä alalla on käytetty ja millaisia tuloksia niillä on saavutettu. Aiemmassa tutkimuksessa on tarkasteltu sekä perinteisiä koneoppimismenetelmiä että syväoppimismalleja ohjelmistovirheiden ennustamisessa (Malhotra, 2015; Wang ym., 2018; Zhang ym., 2021). Lisäksi on tuotu esiin, että menetelmien toimivuus riippuu usein sovellusympäristöstä ja käytännön kontekstista (Madeyski & Stradowski, 2023).

Tutkielman tarkoituksena on selvittää, millaisia koneoppimismenetelmiä ohjelmistovirheiden ennustamiseen on käytetty, millaisia etuja ja haasteita eri menetelmiin liittyy sekä millaisissa tilanteissa menetelmät vaikuttavat kirjallisuuden perusteella toimivan parhaiten. Työ ei sisällä empiiristä tutkimusosuutta, vaan keskittyy olemassa olevan tutkimustiedon analysointiin ja vertailuun, jotta tutkimusalueesta saadaan jäsennelty ja perusteltu kokonaiskuva.

1.2 Työn rakenne

Tutkielma etenee johdonmukaisesti siten, että aluksi esitellään tutkimuksen tausta, tavoite ja keskeiset käsitteet. Johdannon jälkeen käsitellään kirjallisuuskatsaus tutkimusmenetelmänä ja kuvataan, miten katsaus on toteutettu tässä työssä. Tässä yhteydessä

avataan myös narratiivisen kirjallisuuskatsauksen periaatteita (Salminen, 2011) sekä perustellaan menetelmävalinta.

Tämän jälkeen työssä tarkastellaan aiempaa tutkimusta ohjelmistovirheiden ennustamisesta koneoppimismenetelmien avulla. Tarkastelu on jaettu teemoittain, ja siinä käsitellään erikseen valvottuja koneoppimismenetelmiä, syväoppimista sekä epätasapainoisen datan ongelmaa. Lisäksi huomioidaan tutkimusten esiin tuomia teollisia sovelluksia ja käytännön näkökulmia (Madeyski & Stradowski, 2023). Lopussa työssä kootaan yhteen kirjallisuuskatsauksen keskeiset havainnot diskussion sekä johtopäätösten muodossa.

2 Kirjallisuuskatsaus tutkimusmenetelmänä

2.1 Kirjallisuuskatsauksen tavoite ja tarkoitus

Kirjallisuuskatsaus on tutkimusmenetelmä, jonka tavoitteena on koota ja analysoida aiempaa tutkimustietoa systemaattisesti ja kriittisesti. Se ei ole pelkkä referointi, vaan tutkimuskentän jäsentämistä ja kokonaiskuvan muodostamista. Kirjallisuuskatsauksen avulla tutkija tunnistaa keskeiset käsitteet, menetelmät ja tutkimussuunnat sekä arvioi niiden välisiä suhteita (Vaasan yliopisto, 2023).

Kirjallisuuskatsauksen keskeinen tehtävä on osoittaa, mitä aiheesta jo tiedetään ja millaisia tutkimusaukkoja on olemassa. Se toimii tutkimuksen teoreettisena perustana, rajaa tutkimusaiheen ja perustelee tutkimuskysymysten merkityksen. Toisin kuin yksittäinen tutkimusartikkeli, kirjallisuuskatsaus tarkastelee ilmiötä laajemmasta näkökulmasta ja yhdistää useiden tutkimusten tuloksia.

Tässä kandidaatintutkielmassa kirjallisuuskatsauksen tarkoituksena on muodostaa jäsennelty kokonaiskuva ohjelmistovirheiden ennustamisesta koneoppimismenetelmien avulla sekä tunnistaa keskeiset menetelmät ja niiden rajoitteet.

2.2 Kirjallisuuskatsauksen tyypit

Kirjallisuuskatsaukset voidaan jakaa eri tyyppeihin niiden tavoitteen ja toteutustavan perusteella. Salmisen (2011) mukaan keskeiset muodot ovat kuvaileva kirjallisuuskatsaus, systemaattinen kirjallisuuskatsaus ja meta-analyysi.

Systemaattinen kirjallisuuskatsaus perustuu ennalta määriteltyyn ja tarkkaan kuvattuun prosessiin. Siinä tutkimuskysymys, hakustrategia ja aineiston valintakriteerit määritellään selkeästi, jotta tutkimuksen läpinäkyvyys ja toistettavuus paranevat. Meta-analyysi

puolestaan yhdistää aiempien tutkimusten tuloksia tilastollisesti tai laadullisesti ja pyrkii tuottamaan yleistettävän kokonaiskuvan. (Vaasan yliopisto, 2023)

Kuvaileva kirjallisuuskatsaus on joustavampi lähestymistapa, jonka tavoitteena on muodostaa laaja yleiskuva tutkimusalueesta. Se voidaan jakaa narratiiviseen ja integroivaan katsaukseen. Narratiivinen katsaus painottuu tutkimuskentän jäsentämiseen ja keskeisten teemojen esittämiseen. (Vaasan yliopisto, 2023)

Tässä tutkielmassa toteutetaan kuvaileva, tarkemmin narratiivinen kirjallisuuskatsaus, koska tavoitteena on muodostaa kokonaiskuva ohjelmistovirheiden ennustamisesta koneoppimismenetelmien avulla.

2.3 Narratiivinen kirjallisuuskatsaus

Tässä tutkielmassa tutkimusmenetelmäksi on valittu narratiivinen kirjallisuuskatsaus (Salminen, 2011). Sen tavoitteena on muodostaa selkeä ja jäsennelty kokonaiskuva tutkimusaiheesta kokoamalla keskeiset tutkimukset yhteen ja tarkastelemalla niitä temaattisesti.

Narratiivinen katsaus ei perustu yhtä tiukkaan seulontaprotokollaan kuin systemaattinen katsaus, vaan tutkimusten valinta pohjautuu niiden relevanssiin suhteessa tutkimuskysymyksiin. Keskeistä on tutkimuskentän jäsentäminen, keskeisten suuntausten tunnistaminen sekä tulosten tulkinta. (Vaasan yliopisto, 2023)

Narratiivinen lähestymistapa soveltuu erityisesti laajoihin ja metodologisesti monimuotoisiin tutkimusalueisiin. Ohjelmistovirheiden ennustaminen koneoppimismenetelmien avulla sisältää erilaisia algoritmeja, aineistoja ja arviointitapoja, minkä vuoksi narratiivinen katsaus mahdollistaa näiden rinnakkaisen tarkastelun ja vertailun. Tavoitteena on muodostaa tiivis synteesi, joka tuo esiin alan keskeiset kehityssuunnat ja haasteet.

2.4 Kirjallisuuskatsauksen toteuttaminen

Tässä tutkimuksessa kirjallisuuskatsaus toteutettiin vaiheittain siten, että prosessi oli mahdollisimman selkeä ja läpinäkyvä. Vaikka kyseessä on narratiivinen kirjallisuuskatsaus, aineiston valinta ja käsittely tehtiin systemaattisesti, jotta katsaus ei perustuisi satunnaiseen lähdevalintaan.

Ensimmäisessä vaiheessa määriteltiin tutkimuksen rajaus ja keskeiset käsitteet. Tarkastelun kohteeksi rajattiin ohjelmistovirheiden ennustaminen koneoppimismenetelmien avulla. Aihetta lähestyttiin teknisestä näkökulmasta, jolloin painopiste oli algoritmeissa, datassa ja arviointimenetelmissä. Hakuprosessissa käytettiin useita kansainvälisiä tieteellisiä tietokantoja, kuten IEEE Xplore, Scopus ja ScienceDirect, koska ne sisältävät laajasti ohjelmistotekniikan ja tietojenkäsittelytieteen vertaisarvioituja julkaisuja.

Hakusanoina käytettiin muun muassa yhdistelmiä “software defect prediction”, “software fault prediction”, “machine learning” ja “deep learning”. Hakulausekkeissa hyödynnettiin AND- ja OR-operaattoreita, jotta tulokset rajautuisivat ohjelmistovirheiden ennustamiseen ja koneoppimismenetelmiin. Hakua rajattiin pääasiassa vuosille 2015–2024, jotta katsaus keskittyisi ajankohtaiseen tutkimukseen ja erityisesti syväoppimisen yleistymiseen alalla. Mukaan otettiin kuitenkin myös muutamia hieman vanhempia tutkimuksia, mikäli ne olivat tutkimusalueen kannalta keskeisiä.

Toisessa vaiheessa hakutuloksia arvioitiin otsikon ja tiivistelmän perusteella. Mukaan valittiin tutkimukset, jotka käsittelivät nimenomaan ohjelmistovirheiden ennustamista koneoppimismenetelmillä. Poissulkukriteereinä olivat esimerkiksi tutkimukset, jotka eivät olleet vertaisarvioituja, eivät keskittyneet ohjelmistovirheisiin tai joissa koneoppiminen ei ollut keskeisessä roolissa. Lisäksi tutkimusten tuli sisältää empiirinen osuus tai selkeä metodologinen tarkastelu, jotta niitä voitiin hyödyntää analyysissa.

Valituista artikkeleista koottiin tutkimusaineisto, joka esitetään taulukkomuodossa. Taulukkoon kirjattiin kunkin artikkelin tekijät, julkaisuvuosi, käytetty menetelmä, aineisto sekä keskeiset tulokset. Tämän tarkoituksena oli helpottaa tutkimusten keskinäistä vertailua ja tunnistaa toistuvia piirteitä. Taulukointi myös lisäsi katsauksen läpinäkyvyyttä, koska se teki aineiston käsittelystä systemaattisempaa.

Kolmannessa vaiheessa aineisto analysoitiin teemoittelun avulla. Teemoittelu tarkoittaa sitä, että tutkimuksista tunnistetaan toistuvia aihealueita ja ne ryhmitellään laajemmiksi kokonaisuuksiksi. Tässä työssä tutkimukset jaettiin esimerkiksi valvottuja koneoppimismenetelmiä, syväoppimismalleja ja epätasapainoista dataa käsitteleviin teemoihin. Teemat muodostuivat iteratiivisesti aineiston lukemisen aikana, eli niitä tarkennettiin useaan otteeseen, kun tutkimuksista löytyi uusia näkökulmia. (Vaasan yliopisto, 2023)

Kokonaisuutena kirjallisuuskatsauksen toteuttaminen eteni rajauksesta ja hausta aineiston valintaan, taulukointiin ja teemoitteluun. Vaikka lähestymistapa ei ollut systemaattinen meta-analyysi, prosessi pyrittiin tekemään mahdollisimman johdonmukaiseksi ja perustelluksi, jotta katsaus tarjoaa luotettavan ja jäsennellyn kokonaiskuvan tutkimusalueesta.

Valituista tutkimuksista koottiin taulukko, jossa esitetään keskeiset tiedot tutkimuksista, kuten käytetyt menetelmät, aineisto ja keskeiset tulokset. Taulukko helpottaa tutkimusten vertailua ja aineiston jäsentämistä.

Taulukko 1. Kirjallisuuskatsaukseen valitut tutkimukset.

Tekijä(t)	Vuosi	Menetelmä	Aineisto	Keskeiset tulokset
Malhotra	2015	Systemaattinen kirjallisuuskatsaus, useita ML-menetelmiä	Useita dataset-kokoelmia	Ei yhtä parasta menetelmää, suorituskyky riippuu

Tekijä(t)	Vuosi	Menetelmä	Aineisto	Keskeiset tulokset
				datasta ja mittareista
Wang ym.	2018	Syväoppiminen	Ohjelmistoprojektien data	Syväoppiminen saavuttaa kilpailukykyisiä tuloksia, vaatii paljon dataa
Zhang ym.	2021	Katsaus syväoppimismenetelmistä	Useita tutkimuksia	Tulokset vaihtelevat aineiston ja menetelmien mukaan
Huda ym.	2018	Yhdistelmämalli ja uudelleensamplaus	Epätasapainoinen dataset	Epätasapainon käsittely parantaa virheiden tunnistamista
Laradji ym.	2015	Ensemble learning ja piirrevalinta	Ohjelmistodata	Piirrevalinta parantaa mallien suorituskykyä
Nam ym.	2013	Transfer learning	Useita ohjelmistoprojekteja	Projektien välinen oppiminen voi parantaa ennusteita
Jiang ym.	2020	Transfer learning epätasapainoisessa datassa	Heterogeeninen dataset	Yleistettävyys riippuu projektien samankaltaisuudesta
Madeyski & Stradowski	2023	Systemaattinen kirjallisuuskatsaus (teollinen näkökulma)	Useita tutkimuksia	Käytännön sovelluksissa tarvitaan tulkittavia ja käytännöllisiä malleja

Tekijä(t)	Vuosi	Menetelmä	Aineisto	Keskeiset tulokset
Tan ym.	2015	Online defect prediction	Epätasapainoinen data	Reaaliaikainen ennustaminen mahdollista epätasapainossa
Qiao & Wang	2019	Neuroverkot, just-in-time ennustaminen	Ohjelmistoprojektien data	Mahdollistaa virheiden ennustamisen kehityksen aikana
Albattah	2024	Useiden koneoppimismenetelmien vertailu	Useita datasetteja	Menetelmien suorituskyky vaihtelee datan ja piirteiden mukaan

Taulukosta voidaan havaita, että ohjelmistovirheiden ennustamisessa käytetään monipuolisesti erilaisia menetelmiä perinteisistä koneoppimismalleista syväoppimiseen. Lisäksi tutkimukset korostavat datan laadun, epätasapainon ja projektien välisten erojen merkitystä ennustemallien suorituskykyyn. Taulukko toimii pohjana aineiston teemoittelulle, jossa tutkimukset ryhmiteltiin keskeisten aihealueiden perusteella.

3 Aihepiirin tausta ja keskeiset käsitteet

Ohjelmistovirheiden ennustaminen tarkoittaa menetelmiä, joiden avulla pyritään tunnistamaan ohjelmiston osia, jotka todennäköisesti sisältävät virheitä. Tavoitteena on kohdistaa testaus- ja laadunvarmistusresursseja tehokkaammin niihin moduuleihin, joissa virheriski on suurin. Ennustaminen perustuu yleensä ohjelmistoprojekteista kerättyihin historiallisiin tietoihin, kuten koodimetrikoihin, muutoshistoriaan tai kehitysprosessin mittareihin. (Malhotra, 2015; Nam ym., 2013; Wang ym., 2018)

Koneoppimismenetelmien käyttö perustuu ajatukseen, että aiemmista projekteista kerätty data sisältää piirteitä, joiden avulla voidaan arvioida uusien ohjelmistoversioiden virhealttiutta. Malhotra (2015) toteaa systemaattisessa katsauksessaan, että ohjelmistovirheiden ennustamisessa on käytetty laajasti erilaisia valvottuja koneoppimismenetelmiä, kuten päätöspuita, tukivektorikoneita ja logistista regressiota. Näiden mallien toimivuus riippuu kuitenkin usein käytetystä aineistosta ja ominaisuuksien valinnasta (Malhotra, 2015).

Viime vuosina syväoppimismenetelmät ovat yleistyneet ohjelmistovirheiden ennustamisessa. Wang ym. (2018) esittävät, että syväoppimismallit voivat oppia ohjelmistodatan piirteistä useita abstraktiotasoja, mikä mahdollistaa monimutkaisempien riippuvuuksien tunnistamisen. Myös Zhang ym. (2021) korostavat katsauksessaan, että syväoppiminen tarjoaa uusia mahdollisuuksia erityisesti silloin, kun käytettävissä on suuria määriä dataa ja monipuolisia piirteitä.

Ohjelmistovirheiden ennustamiseen liittyy myös haasteita, joista yksi keskeisimmistä on luokkaepätasapaino. Käytännössä virheellisiä moduuleja on usein huomattavasti vähemmän kuin virheettömiä, mikä voi heikentää mallien kykyä tunnistaa virheitä (Huda ym., 2018; Tan ym., 2015). Tämän ongelman ratkaisemiseksi on kehitetty erilaisia uudelleensamplaus- ja yhdistelmämalleja (Huda ym., 2018). Lisäksi projektien välinen vaihtelu voi vaikeuttaa mallien yleistettävyyttä, mikä on tuotu esiin myös transfer learning -lähestymistavoissa (Nam ym., 2013; Jiang ym., 2020).

Näin ollen ohjelmistovirheiden ennustaminen on tutkimusalue, jossa yhdistyvät ohjelmistotekniikan mittarit, koneoppimismenetelmät sekä käytännön sovellusympäristön vaatimukset. Menetelmien toimivuus ei riipu pelkästään algoritmista, vaan myös datan laadusta, esikäsittelystä ja arviointitavasta.

3.1 Ohjelmistomittarit ja metriikat

Ohjelmistovirheiden ennustaminen perustuu tyypillisesti erilaisiin ohjelmistomittareihin eli metriikkoihin, joiden avulla pyritään kuvaamaan ohjelmiston rakennetta, monimutkaisuutta ja kehitysprosessia. Ennustemallit eivät suoraan analysoi koko ohjelmistoa, vaan ne käyttävät numeerisia piirteitä, jotka on johdettu lähdekoodista tai projektin historiasta. (Malhotra, 2015; Wang ym., 2018)

Malhotra (2015) toteaa, että ohjelmistovirheiden ennustamisessa hyödynnetään usein staattisia koodimetrikoita, kuten rivimäärää (Lines of Code), syklomaattista kompleksisuutta sekä olio-ohjelmointiin liittyviä CK-metriikoita. Näiden mittareiden avulla pyritään arvioimaan koodin rakennetta ja monimutkaisuutta, koska monimutkaisempi koodi voi olla alttiimpi virheille. Mittareiden valinta vaikuttaa kuitenkin merkittävästi mallin suorituskykyyn, eikä yksittäinen metriikka yksinään yleensä riitä selittämään virhealttiutta (Malhotra, 2015).

Staattisten metrikoiden lisäksi tutkimuksessa on hyödynnetty myös kehitysprosessiin liittyviä mittareita, kuten muutosten määrää, kehittäjien lukumäärää tai commit-historiaa. Wang ym. (2018) korostavat, että erityisesti syväoppimismenetelmät voivat hyödyntää laajempaa ja monimuotoisempaa piirrejoukkoa kuin perinteiset mallit. Syväoppimismallit voivat oppia piirteiden välisiä monimutkaisia riippuvuuksia ilman, että kaikkia ominaisuuksia täytyy määritellä käsin.

Ominaisuuksien valinta ja esikäsittely ovat keskeisessä roolissa ennustemallien kehittämisessä. Liian suuri määrä piirteitä voi lisätä mallin monimutkaisuutta ja heikentää

yleistettävyyttä, kun taas liian rajattu piirrejoukko voi jättää olennaisia tekijöitä huomioimatta. Laradji ym. (2015) osoittavat, että piirteiden valinta yhdistettynä yhdistelmämalleihin voi parantaa ennustustuloksia verrattuna tilanteeseen, jossa kaikkia saatavilla olevia piirteitä käytetään ilman valintaa.

Metriikoiden käyttöön liittyy myös haasteita. Eri projektien välillä käytetyt mittarit ja niiden jakaumat voivat vaihdella huomattavasti, mikä voi vaikeuttaa mallien yleistämistä uusiin ympäristöihin. Nam ym. (2013) tuovat esiin, että projektien välinen vaihtelu on yksi syy siihen, miksi siirto-oppimista on alettu hyödyntää ohjelmistovirheiden ennustamisessa. Tällöin pyritään siirtämään opittua tietoa projektista toiseen, vaikka käytettävät mittarit tai niiden jakaumat poikkeaisivat toisistaan.

Yhteenvetona voidaan todeta, että ohjelmistomittarit muodostavat perustan ennustemallien toiminnalle. Mallien suorituskyky ei riipu pelkästään valitusta algoritmista, vaan myös siitä, millaisia piirteitä käytetään ja miten ne on esikäsitelty.

3.2 Ennustemallien arviointimittarit

Ohjelmistovirheiden ennustamisessa pelkkä mallin rakentaminen ei riitä, vaan yhtä tärkeää on arvioida mallin suorituskykyä asianmukaisilla mittareilla. Arviointimittareiden valinta vaikuttaa siihen, millaisia johtopäätöksiä mallin toimivuudesta voidaan tehdä. Erityisesti luokkaepätasapainon vuoksi kaikkia mittareita ei voida pitää yhtä informatiivisina.

Malhotra (2015) tuo esiin, että ohjelmistovirheiden ennustamisessa käytetään usein mittareita kuten tarkkuus (accuracy), herkkyys (recall), tarkkuusmitta (precision) ja F1-luku. Vaikka tarkkuus on yleisesti käytetty mittari, se voi antaa harhaanjohtavan kuvan mallin suorituskyvystä silloin, kun virheellisiä tapauksia on huomattavasti vähemmän kuin virheettömiä. Tällöin malli voi saavuttaa korkean tarkkuuden ennustamalla suurimman osan tapauksista virheettömiksi, vaikka se ei tunnistaisi virheitä tehokkaasti.

Luokkaepätasapainon vuoksi tutkimuksessa on korostettu erityisesti herkkyyden ja tarkkuusmitan merkitystä sekä niiden yhdistävää F1-lukua (Huda ym., 2018; Malhotra, 2015). Huda ym. (2018) osoittavat, että kun aineisto tasapainotetaan ennen mallin kouluttamista, mallin kyky tunnistaa virheellisiä tapauksia paranee. Tämä korostaa sitä, että arviointimittarit ja datan esikäsittely liittyvät tiiviisti toisiinsa.

Lisäksi ROC-käyrä ja sen alle jäävä pinta-ala (AUC) ovat yleisesti käytettyjä mittareita mallien vertailussa (Malhotra, 2015). AUC mittaa mallin kykyä erottaa virheelliset ja virheettömät tapaukset toisistaan eri kynnysarvoilla. Se tarjoaa usein tasapainoisemman kuvan mallin suorituskyvystä kuin pelkkä tarkkuus.

Arviointimittareiden valinta on erityisen tärkeää myös teollisissa sovelluksissa. Madeyski ja Stradowski (2023) korostavat, että mallien hyödyllisyyttä tulisi tarkastella suhteessa käytännön tarpeisiin. Esimerkiksi tilanteessa, jossa virheen havaitsematta jättäminen aiheuttaa merkittäviä kustannuksia, herkkyys voi olla tärkeämpi mittari kuin kokonais-tarkkuus.

Näin ollen ennustemallien arviointi ei ole pelkästään tekninen vaihe, vaan se vaikuttaa siihen, miten mallin toimivuutta tulkitaan ja miten sitä voidaan hyödyntää käytännössä. Sopivien arviointimittareiden valinta on keskeinen osa luotettavan ohjelmistovirheiden ennustemallin kehittämistä.

Arviointimittareiden merkitys korostuu erityisesti tilanteissa, joissa aineisto on epätasapainoinen. Tällöin pelkkä tarkkuus (accuracy) voi antaa harhaanjohtavan kuvan mallin suorituskyvystä. Esimerkiksi tilanteessa, jossa virheelliset tapaukset muodostavat vain pienen osan aineistosta, malli voi saavuttaa korkean tarkkuuden ennustamalla suurimman osan tapauksista oikein ilman, että se tunnistaa virheitä tehokkaasti. Tämän vuoksi tarkkuuden lisäksi on tärkeää tarkastella muita mittareita, kuten precisionia, recallia ja F1-arvoa. (Malhotra, 2015)

Precision kuvaa sitä, kuinka suuri osa mallin ennustamista virheellisistä tapauksista on todellisuudessa virheellisiä. Recall puolestaan mittaa sitä, kuinka hyvin malli tunnistaa kaikki todelliset virheelliset tapaukset. Näiden mittareiden välillä on usein tasapainottelua, sillä mallin optimointi toisen suhteen voi heikentää toista. F1-arvo yhdistää precisionin ja recallin yhdeksi mittariksi ja antaa näin tasapainoisemman kuvan mallin suorituskyvystä erityisesti epätasapainoisissa aineistoissa. (Huda ym., 2018)

Lisäksi voidaan tarkastella myös muita mittareita, kuten ROC-käyrää ja AUC-arvoa, jotka kuvaavat mallin kykyä erotella eri luokkia toisistaan. Nämä mittarit voivat olla hyödyllisiä erityisesti silloin, kun halutaan vertailla eri mallien suorituskykyä kokonaisvaltaisesti. On kuitenkin tärkeää huomata, että mikään yksittäinen mittari ei yksin riitä kuvaamaan mallin toimivuutta, vaan arviointi tulisi tehdä useiden mittareiden perusteella. (Malhotra, 2015)

Kokonaisuutena voidaan todeta, että arviointimittareiden valinta on keskeinen osa ohjelmistovirheiden ennustamista. Oikeiden mittareiden avulla voidaan saada realistisempi kuva mallin suorituskyvystä ja välttää virheellisiä johtopäätöksiä. Tämä korostaa sitä, että mallien vertailussa ei tulisi keskittyä pelkästään yhteen mittariin, vaan tarkastella tuloksia laajemmin eri näkökulmista.

3.3 Datasetit ja projektien välinen yleistettävyys

Ohjelmistovirheiden ennustamisessa käytettävät aineistot eli datasetit vaikuttavat merkittävästi tutkimustuloksiin. Mallien suorituskyky riippuu paitsi valituista algoritmeista ja mittareista myös siitä, millaiseen dataan ne on koulutettu ja testattu. Tutkimuksessa on yleisesti käytetty avoimia ohjelmistoprojekteihin perustuvia aineistoja, joiden avulla eri menetelmiä voidaan vertailla keskenään. (Malhotra, 2015; Zhang ym., 2021)

Malhotra (2015) tuo esiin, että monet tutkimukset perustuvat julkisiin dataset-kokoelmiin, mikä mahdollistaa menetelmien vertailun ja toistettavuuden. Samalla hän

kuitenkin huomauttaa, että eri aineistojen ominaisuudet voivat poiketa toisistaan merkittävästi, mikä vaikuttaa mallien tuloksiin. Esimerkiksi projektin koko, kehittäjien määrä ja koodin rakenne voivat vaihdella, mikä tekee tulosten yleistämisestä haastavaa.

Projektien välinen yleistettävyys on ollut keskeinen tutkimuskysymys. Nam ym. (2013) tarkastelevat transfer learning -lähestymistapaa, jossa mallia pyritään soveltamaan projektiin, josta ei ole riittävästi omaa historiallista dataa. Tällöin hyödynnetään toisen projektin tietoa ja pyritään siirtämään opittuja piirteitä uuteen kontekstiin. Tämä lähestymistapa voi parantaa mallien toimivuutta tilanteissa, joissa uuden projektin aineisto on rajallinen.

Myös Jiang ym. (2020) käsittelevät projektien välistä ennustamista erityisesti epätasapainoisen datan näkökulmasta. Heidän tuloksensa osoittavat, että siirto-oppiminen voi auttaa parantamaan mallien suorituskykyä, mutta onnistuminen riippuu siitä, kuinka samankaltaisia lähde- ja kohdeprojektit ovat keskenään. Jos projektit poikkeavat merkittävästi toisistaan, mallin suorituskyky voi heikentyä.

Datasetien käyttöön liittyy myös haasteita, kuten datan laatu ja puutteellisuus. Wang ym. (2018) korostavat, että syväoppimismallit hyötyvät suurista ja laadukkaista aineistoista, mutta kaikissa projekteissa tällaista dataa ei ole saatavilla. Tämä rajoittaa menetelmien sovellettavuutta erityisesti pienissä tai uusissa projekteissa.

Yhteenvetona voidaan todeta, että datasetit ja niiden ominaisuudet vaikuttavat ratkaisevasti ennustemallien toimivuuteen. Mallien yleistettävyys eri projektien välillä ei ole itsestäänselvyys, ja siksi projektien välisten erojen huomioiminen on tärkeä osa ohjelmistovirheiden ennustamiseen liittyvää tutkimusta.

4 Aiempi tutkimus ohjelmistovirheiden ennustamisesta

Ohjelmistovirheiden ennustamista koneoppimismenetelmien avulla on tutkittu laajasti jo useiden vuosien ajan. Tutkimus on kehittynyt perinteisistä luokittelumenetelmistä kohti monimutkaisempia ja datalähtöisempiä lähestymistapoja. Malhotra (2015) esittää systemaattisessa kirjallisuuskatsauksessaan, että ohjelmistovirheiden ennustamisessa on käytetty monipuolisesti erilaisia koneoppimistekniikoita, ja että yksittäisen menetelmän paremmuus riippuu usein käytetystä aineistosta ja arviointitavasta.

Myöhemmässä tutkimuksessa on kiinnitetty huomiota erityisesti syväoppimismenetelmiin ja niiden mahdollisuuksiin käsitellä monimutkaisempaa ohjelmistodataa. Wang ym. (2018) osoittavat, että syväoppimiseen perustuvat mallit voivat saavuttaa kilpailukykyisiä tai jopa parempia tuloksia verrattuna perinteisiin koneoppimismenetelmiin. Zhang ym. (2021) puolestaan kokoavat yhteen syväoppimista käsittelevää tutkimusta ja toteavat, että menetelmien kehitys on ollut nopeaa, mutta tulokset vaihtelevat huomattavasti aineistojen ja arviointimenetelmien mukaan.

Tutkimuskentässä on myös tarkasteltu mallien soveltamista käytännön ympäristöissä. Madeyski ja Stradowski (2023) korostavat, että teollisissa sovelluksissa mallien käyttö edellyttää huomioimaan organisaation omat prosessit, käytössä olevat metriikat ja resurssit. Näin ollen pelkkä korkea ennustustarkkuus ei riitä, vaan mallien tulee olla myös käytännöllisiä ja sovellettavissa todellisissa kehitysympäristöissä.

Aiempi tutkimus osoittaa, että ohjelmistovirheiden ennustaminen on monimuotoinen tutkimusalue, jossa yhdistyvät algoritmien kehittäminen, datan esikäsittely, arviointimenetelmät sekä käytännön soveltaminen. Seuraavissa alaluvuissa tarkastellaan tarkemmin eri menetelmäryhmiä ja niiden keskeisiä piirteitä. Myös uudemmassa tutkimuksessa on todettu, että koneoppimismenetelmät tarjoavat tehokkaita ratkaisuja ohjelmistovirheiden ennustamiseen, mutta niiden toimivuus riippuu vahvasti käytettävissä olevasta datasta ja valituista piirteistä (Albattah, 2024).

4.1 Valvotut koneoppimismenetelmät

Valvotut koneoppimismenetelmät ovat olleet pitkään keskeisessä roolissa ohjelmistovirheiden ennustamisessa. Näissä menetelmissä malli opetetaan käyttämällä aineistoa, jossa tiedetään etukäteen, mitkä moduulit sisältävät virheitä ja mitkä eivät. Tavoitteena on oppia tunnistamaan sellaisia piirteitä, joiden perusteella uusia moduuleja voidaan luokitella virheellisiksi tai virheettömiksi. (Malhotra, 2015; Nam ym., 2013)

Malhotra (2015) toteaa systemaattisessa katsauksessaan, että ohjelmistovirheiden ennustamisessa on käytetty laajasti muun muassa päätöspuita, logistista regressiota, Naive Bayes -menetelmää sekä tukivektorikoneita. Tutkimusten perusteella mikään yksittäinen menetelmä ei ole selvästi paras kaikissa tilanteissa, vaan mallien suorituskyky riippuu käytetystä aineistosta, ominaisuuksien valinnasta ja arviointimittareista. Tämä viittaa siihen, että menetelmän valinta tulisi tehdä tapauskohtaisesti. (Malhotra, 2015; Zhang ym., 2021)

Valvottujen mallien etuna on usein niiden suhteellinen yksinkertaisuus ja tulkittavuus verrattuna monimutkaisempiin syväoppimismalleihin. Toisaalta niiden suorituskyky voi heikentyä, jos data on epätasapainoista tai sisältää paljon kohinaa (Malhotra, 2015; Huda ym., 2018). Luokkaepätasapainon ongelmaa on pyritty ratkaisemaan esimerkiksi yhdistelmämalleilla ja uudelleensamplausmenetelmillä (Huda ym., 2018). Näiden lähestymistapojen tavoitteena on parantaa mallin kykyä tunnistaa harvinaisempia virheellisiä tapauksia.

Lisäksi projektien välinen vaihtelu voi vaikuttaa valvottujen mallien toimivuuteen. Nam ym. (2013) tarkastelevat transfer learning -lähestymistapaa, jossa tietoa siirretään projektista toiseen, jotta ennustemallit toimisivat paremmin uusissa ympäristöissä. Myös Jiang ym. (2020) osoittavat, että siirtämällä opittuja piirteitä eri projektien välillä voidaan parantaa mallien suorituskykyä erityisesti tilanteissa, joissa data on epätasapainoista.

Yhteenvetona voidaan todeta, että valvotut koneoppimismenetelmät muodostavat edelleen perustan ohjelmistovirheiden ennustamiselle. Niiden toimivuus riippuu kuitenkin vahvasti aineiston laadusta, ominaisuuksien valinnasta ja siitä, miten mallien suorituskykyä arvioidaan.

4.2 Neuroverkot ja syväoppiminen

Neuroverkot ja syväoppimismenetelmät ovat viime vuosina nousseet merkittäväksi tutkimussuunnaksi ohjelmistovirheiden ennustamisessa. Toisin kuin perinteiset koneoppimismallit, syväoppimismallit pystyvät oppimaan useita abstraktiotasoja suoraan datasta ilman laajaa manuaalista ominaisuuksien suunnittelua. Tämä voi olla hyödyllistä erityisesti silloin, kun käsitellään monimutkaista ohjelmistodataa, kuten lähdekoodin rakenteita tai muutoshistoriaa. (Wang ym., 2018; Zhang ym., 2021)

Wang ym. (2018) esittävät, että syväoppimismallit voivat muodostaa semanttisia esityksiä ohjelmistodatan piirteistä ja siten tunnistaa monimutkaisempia riippuvuuksia kuin perinteiset menetelmät. Heidän tutkimuksensa mukaan syväoppimiseen perustuvat mallit saavuttivat kilpailukykyisiä tuloksia ohjelmistovirheiden ennustamisessa. Myös Zhang ym. (2021) toteavat katsauksessaan, että syväoppimismenetelmien käyttö on lisääntynyt nopeasti ja että niitä on sovellettu monenlaisiin ohjelmistodatan muotoihin.

Syväoppimismallien etuna pidetään niiden kykyä käsitellä suuria datamääriä ja oppia automaattisesti olennaisia piirteitä. Toisaalta niiden käyttöön liittyy myös haasteita. Mallit vaativat usein paljon opetusdataa ja laskentatehoa, ja niiden tulkittavuus on heikompi verrattuna yksinkertaisempiin malleihin (Wang ym., 2018). Tämä voi vaikeuttaa niiden käyttöönottoa teollisissa ympäristöissä, joissa mallien ymmärrettävyys ja läpinäkyvyys ovat tärkeitä. Myös Qiao ja Wang (2019) osoittavat, että neuroverkkoja voidaan hyödyntää ohjelmistovirheiden ennustamisessa kehityksen aikana niin sanotussa just-in-time-lähestymistavassa, jossa virheitä pyritään ennustamaan heti muutosten yhteydessä.

Lisäksi syväoppimismenetelmien suorituskyky ei ole aina selvästi parempi kuin perinteisten mallien, vaan tulokset vaihtelevat aineistosta ja arviointimenetelmistä riippuen (Zhang ym., 2021). Näin ollen syväoppiminen tarjoaa uusia mahdollisuuksia ohjelmistovirheiden ennustamiseen, mutta sen soveltuvuus riippuu vahvasti käytettävästä datasta ja kontekstista.

4.3 Epätasapainoisen datan ongelma

Yksi keskeisimmistä haasteista ohjelmistovirheiden ennustamisessa on luokkaepätasapaino. Käytännössä virheellisiä moduuleja on usein huomattavasti vähemmän kuin virheettömiä, mikä voi johtaa siihen, että koneoppimismalli oppii painottamaan enemmistöluokkaa. Tällöin malli voi saavuttaa korkean kokonais-tarkkuuden, vaikka sen kyky tunnistaa virheellisiä tapauksia olisi heikko. (Huda ym., 2018; Tan ym., 2015)

Luokkaepätasapainon vaikutuksia on pyritty vähentämään erilaisilla menetelmillä. Huda ym. (2018) esittävät yhdistelmämallin, jossa käytetään uudelleensamplausmenetelmiä tasapainottamaan aineistoa ennen mallin kouluttamista. Heidän tulostensa mukaan epätasapainon huomioiminen paransi mallien kykyä tunnistaa virheellisiä tapauksia verrattuna tilanteeseen, jossa dataa ei tasapainotettu. Tan ym. (2015) tarkastelevat epätasapainoisen datan käsittelyä erityisesti reaaliaikaisessa ennustamisessa ja osoittavat, että luokkaepätasapaino vaikuttaa merkittävästi mallien suorituskykyyn myös dynaamisissa ympäristöissä.

Epätasapaino liittyy myös projektien välisiin eroihin. Nam ym. (2013) tarkastelevat transfer learning -lähestymistapaa, jossa mallia mukautetaan uuteen projektiin hyödyntämällä aiemmin opittua tietoa. Tällainen lähestymistapa voi auttaa tilanteissa, joissa uuden projektin aineisto on pieni tai epätasapainoinen. Myös Jiang ym. (2020) osoittavat, että siirtämällä tietoa projektien välillä voidaan parantaa ennustemallien suorituskykyä erityisesti silloin, kun positiivisia tapauksia on vähän.

Luokkaepätasapainon vuoksi pelkkä tarkkuus ei aina ole riittävä arviointimittari, vaan mallien suorituskykyä tulisi tarkastella useamman mittarin avulla. Tämän vuoksi aineiston esikäsittely, sopivien arviointimittareiden valinta ja mallin säätäminen ovat keskeisessä roolissa, kun pyritään kehittämään luotettavia ennustemalleja.

4.4 Teolliset sovellukset

Vaikka ohjelmistovirheiden ennustamista on tutkittu laajasti akateemisessa ympäristössä, mallien soveltaminen käytännön teollisiin projekteihin tuo mukanaan omia haasteitaan. Tutkimustulokset eivät aina siirry sellaisenaan todellisiin kehitysympäristöihin, joissa projektien koko, käytettävät työkalut ja kehitysprosessit voivat vaihdella merkittävästi.

Madeyski ja Stradowski (2023) tarkastelevat systemaattisessa kirjallisuuskatsauksessaan koneoppimisen käyttöä teollisissa ohjelmistovirheiden ennustamisen sovelluksissa. Heidän mukaansa mallien käyttöönotto vaatii muutakin kuin hyvää ennustustarkkuutta. Mallien tulee olla yhteensopivia organisaation omien prosessien ja käytössä olevien metriikoiden kanssa, ja niiden tuottamien tulosten tulee olla ymmärrettäviä ja hyödynnettävissä päätöksenteossa.

Teollisissa ympäristöissä korostuu myös resurssien rajallisuus. Syväoppimismallit voivat tarjota hyviä tuloksia, mutta niiden vaatima laskentateho ja datamäärä voivat rajoittaa niiden käyttöä käytännössä (Wang ym., 2018). Lisäksi mallien tulkittavuus on tärkeä tekijä, koska kehittäjien ja projektipäälliköiden on voitava luottaa mallien antamiin ennusteisiin.

Näin ollen teollisten sovellusten näkökulmasta ohjelmistovirheiden ennustaminen ei ole pelkästään tekninen ongelma, vaan siihen liittyy myös organisatorisia ja käytännöllisiä kysymyksiä. Mallien onnistunut hyödyntäminen edellyttää tasapainoa ennustustarkkuuden, tulkittavuuden ja käytännön sovellettavuuden välillä.

Teollisissa sovelluksissa koneoppimismallien hyödyntäminen voi näkyä konkreettisesti esimerkiksi testauksen kohdentamisessa ja resurssien optimoinnissa. Ennustemallien avulla voidaan tunnistaa ohjelmiston osia, joissa virheiden todennäköisyys on suurin, jolloin testaus voidaan kohdistaa näihin kriittisiin komponentteihin. Tämä voi parantaa testauksen tehokkuutta ja vähentää tarpeettoman työn määrää verrattuna tilanteeseen, jossa testaus kohdistetaan tasaisesti kaikkiin moduuleihin (Madeyski & Stradowski, 2023).

Lisäksi ennustemalleja voidaan hyödyntää ohjelmistokehityksen eri vaiheissa. Esimerkiksi kehitysvaiheessa mallit voivat auttaa tunnistamaan riskialttiita muutoksia jo varhaisessa vaiheessa, jolloin virheiden korjaaminen on vielä edullista. Myös ylläpitovaiheessa ennustemallit voivat tukea päätöksentekoa, esimerkiksi priorisoimalla korjattavat virheet niiden todennäköisen vaikutuksen perusteella (Qiao & Wang, 2019).

Käytännön sovelluksissa mallien hyödyntämiseen liittyy kuitenkin myös haasteita. Organisaatioiden käytössä oleva data voi olla hajanaista tai epätäydellistä, mikä vaikeuttaa mallien kouluttamista ja käyttöönottoa. Lisäksi mallien integrointi osaksi olemassa olevia kehitysprosesseja voi vaatia muutoksia toimintatapoihin. Tämä voi hidastaa uusien menetelmien käyttöönottoa, vaikka niiden potentiaali olisi suuri (Wang ym., 2018; Madeyski & Stradowski, 2023).

Näin ollen voidaan todeta, että koneoppimismallien hyödyntäminen teollisissa ohjelmistoprojekteissa tarjoaa merkittäviä mahdollisuuksia, mutta niiden onnistunut käyttöönotto edellyttää sekä teknisten että organisatoristen tekijöiden huomioimista. Pelkkä mallin kehittäminen ei riitä, vaan sen on sovelluttava käytännön kehitysympäristöön ja tuotettava lisäarvoa ohjelmistokehityksen tueksi (Madeyski & Stradowski, 2023).

5 Diskussio

5.1 Vastaukset tutkimuskysymyksiin

Tässä luvussa tarkastellaan kirjallisuuskatsauksen keskeisiä havaintoja suhteessa tutkimuskysymyksiin. Tarkoituksena on koota yhteen aiemmissa luvuissa esitettyjä tuloksia ja arvioida niitä kokonaisuutena. Lisäksi pyritään tunnistamaan keskeisiä yhtäläisyyksiä ja eroja eri tutkimusten välillä sekä pohtimaan menetelmien soveltuvuutta käytännön ohjelmistokehitykseen.

Ensimmäinen tutkimuskysymys koski sitä, millaisia koneoppimismenetelmiä ohjelmistovirheiden ennustamisessa käytetään. Kirjallisuuden perusteella voidaan todeta, että menetelmävalikoima on laaja ja sisältää sekä perinteisiä valvottuja koneoppimismenetelmiä että uudempiin lähestymistapoihin kuuluvia syväoppimismalleja. Perinteisistä menetelmistä yleisimpiä ovat muun muassa päätöspuut, logistinen regressio, Naive Bayes ja tukivektorikoneet (Malhotra, 2015). Näitä menetelmiä on käytetty pitkään, ja niiden etuna pidetään usein yksinkertaisuutta ja suhteellisen hyvää tulkittavuutta. Viime vuosina syväoppimismenetelmät ovat kuitenkin yleistyneet, koska ne pystyvät käsittelemään monimutkaisempia piirteitä ja oppimaan datasta useita abstraktiotasoja (Wang ym., 2018; Zhang ym., 2021). Näin ollen voidaan todeta, että tutkimuskenttä on siirtymässä kohti monimutkaisempia ja datalähtöisempiä menetelmiä, vaikka perinteiset mallit ovat edelleen keskeisessä roolissa.

Toinen tutkimuskysymys käsitteli sitä, millaisia tuloksia eri menetelmillä on saavutettu. Kirjallisuuden perusteella ei voida yksiselitteisesti osoittaa yhtä menetelmää, joka olisi kaikissa tilanteissa paras. Malhotra (2015) korostaa, että mallien suorituskyky riippuu vahvasti käytetystä aineistosta ja arviointimittareista. Myös syväoppimismenetelmien osalta tulokset ovat vaihtelevia: Wang ym. (2018) esittävät, että syväoppiminen voi saavuttaa kilpailukykyisiä tuloksia, kun taas Zhang ym. (2021) huomauttavat, että menetelmien tehokkuus vaihtelee huomattavasti eri tutkimuksissa. Tämä viittaa siihen, että

menetelmän valinta ei ole universaali ratkaisu, vaan se tulisi tehdä tapauskohtaisesti ottaen huomioon käytettävissä oleva data ja sovellusympäristö. Lisäksi yhdistelmämallit ja uudelleensamplausmenetelmät voivat parantaa tuloksia erityisesti epätasapainoisissa aineistoissa (Huda ym., 2018; Laradji ym., 2015).

Kolmas tutkimuskysymys liittyi siihen, mitkä tekijät vaikuttavat ennustemallien suorituskykyyn. Kirjallisuuskatsauksen perusteella keskeisiksi tekijöiksi nousevat datan laatu, ominaisuuksien valinta, käytetty dataset sekä arviointimittarit. Useat tutkimukset korostavat, että mallin suorituskyky ei riipu pelkästään algoritmista, vaan myös siitä, millaisia piirteitä käytetään ja miten data on esikäsitelty (Malhotra, 2015; Wang ym., 2018). Lisäksi luokkaepätasapaino on merkittävä tekijä, joka voi heikentää mallien kykyä tunnistaa virheellisiä tapauksia (Huda ym., 2018). Datasettien osalta projektien välinen vaihtelu vaikuttaa siihen, kuinka hyvin mallit yleistyvät uusiin ympäristöihin (Nam ym., 2013; Jiang ym., 2020). Näin ollen ennustemallien kehittäminen on kokonaisuus, jossa useat tekijät vaikuttavat lopputulokseen samanaikaisesti.

Neljäs tutkimuskysymys koski menetelmiin liittyviä haasteita. Yksi keskeisimmistä haasteista on luokkaepätasapaino, joka voi johtaa siihen, että mallit painottavat enemmistöluokkaa ja jättävät virheelliset tapaukset tunnistamatta (Huda ym., 2018). Toinen merkittävä haaste liittyy datan saatavuuteen ja laatuun, erityisesti silloin kun tarkastellaan uusia tai pieniä ohjelmistoprojekteja, joissa historiallista dataa on vähän (Wang ym., 2018). Lisäksi syväoppimismenetelmien käyttöön liittyy tulkittavuuden ongelma, mikä voi rajoittaa niiden hyödyntämistä käytännön ympäristöissä. Myös projektien välinen yleistettävyys on haaste, koska mallit eivät välttämättä toimi yhtä hyvin eri projekteissa ilman mukauttamista (Nam ym., 2013; Jiang ym., 2020). Näiden haasteiden perusteella voidaan todeta, että teknisesti toimiva malli ei välttämättä ole käytännössä helposti hyödynnettävissä.

Viides tutkimuskysymys käsitteli sitä, miten menetelmiä voidaan hyödyntää käytännön ohjelmistokehityksessä. Kirjallisuuden perusteella koneoppimismallit voivat auttaa

kohdistamaan testaus- ja laadunvarmistusresursseja tehokkaammin, mutta niiden käyttöönotto edellyttää useiden tekijöiden huomioimista. Madeyski ja Stradowski (2023) korostavat, että teollisissa sovelluksissa mallien tulee olla yhteensopivia organisaation prosessien kanssa ja niiden tulosten tulee olla ymmärrettäviä. Pelkkä korkea ennustustarkkuus ei riitä, vaan mallien on oltava myös käytännöllisiä ja luotettavia. Tämä tarkoittaa, että yritysten on huomioitava esimerkiksi käytettävissä oleva data, resurssit sekä se, miten mallien tuottamaa tietoa voidaan hyödyntää päätöksenteossa.

Kokonaisuutena kirjallisuuskatsaus osoittaa, että ohjelmistovirheiden ennustaminen koneoppimismenetelmien avulla on kehittyvä ja monimuotoinen tutkimusalue. Eri menetelmien välillä ei ole selkeää paremmuusjärjestystä, vaan niiden toimivuus riippuu kontekstista. Lisäksi tutkimus osoittaa, että datan laatu, esikäsittely ja arviointimenetelmät ovat yhtä tärkeitä kuin itse algoritmi. Tämä korostaa sitä, että ohjelmistovirheiden ennustaminen ei ole pelkästään tekninen ongelma, vaan siihen liittyy myös käytännön ja organisatorisia näkökulmia.

5.2 Menetelmien vertailu

Kirjallisuuden perusteella voidaan tarkastella tarkemmin myös eri menetelmien välisiä eroja ja niiden soveltuvuutta erilaisiin tilanteisiin. Perinteiset valvotut koneoppimismenetelmät ja syväoppimismallit eroavat toisistaan erityisesti monimutkaisuuden, datavaatimusten ja tulkittavuuden osalta. Perinteiset menetelmät, kuten logistinen regressio ja päätöspuut, ovat usein helpommin tulkittavia ja vaativat vähemmän dataa, minkä vuoksi ne voivat soveltua paremmin tilanteisiin, joissa aineisto on rajallinen tai mallin toiminnan ymmärtäminen on tärkeää (Malhotra, 2015). Sen sijaan syväoppimismallit pystyvät hyödyntämään suuria ja monimuotoisia aineistoja tehokkaammin, mutta niiden käyttö edellyttää usein enemmän laskentatehoa ja suurempaa datamäärää (Wang ym., 2018; Zhang ym., 2021).

Menetelmien vertailussa voidaan myös huomata, että korkea ennustustarkkuus ei välttämättä tarkoita parasta mahdollista ratkaisua käytännön kannalta. Esimerkiksi syväoppimismallit voivat saavuttaa hyviä tuloksia, mutta niiden heikompi tulkittavuus voi vaikeuttaa niiden käyttöönottoa teollisissa ympäristöissä. Toisaalta yksinkertaisemmat mallit voivat olla hieman vähemmän tarkkoja, mutta niiden tuottamat tulokset voivat olla helpommin ymmärrettäviä ja siten hyödyllisempiä päätöksenteossa. Tämä korostaa sitä, että menetelmän valinnassa tulisi huomioida paitsi suorituskyky myös käytännön sovellettavuus.

Menetelmien vertailua voidaan tarkastella myös siitä näkökulmasta, millaisiin ongelmiin eri lähestymistavat soveltuvat parhaiten. Perinteiset koneoppimismenetelmät voivat olla erityisen hyödyllisiä tilanteissa, joissa aineisto on rakenteeltaan yksinkertaista ja piirteet ovat selkeästi määriteltyjä. Tällöin mallien tulkittavuus ja kevyempi laskennallinen kuormitus voivat olla merkittäviä etuja, erityisesti käytännön sovelluksissa, joissa resurssit ovat rajallisia (Malhotra, 2015).

Syväoppimismenetelmät puolestaan soveltuvat paremmin tilanteisiin, joissa data on monimutkaisempaa tai sisältää epälineaarisia riippuvuuksia. Ne voivat hyödyntää laajempaa piirrejoukkoa ja oppia automaattisesti uusia esitystapoja datasta ilman manuaalista ominaisuuksien suunnittelua (Wang ym., 2018; Zhang ym., 2021). Tämä voi johtaa parempiin ennustustuloksiin erityisesti suurissa ja monimuotoisissa aineistoissa. Toisaalta tällainen lähestymistapa voi heikentää mallien tulkittavuutta, mikä voi olla ongelma käytännön ohjelmistokehityksessä.

Menetelmien vertailussa on myös huomioitava, että ennustustulosten parantaminen ei aina edellytä yksittäisen algoritmin vaihtamista, vaan usein keskeisessä roolissa ovat datan esikäsittely ja piirteiden valinta. Esimerkiksi epätasapainoisen datan käsittely uudelleensamplausmenetelmillä tai yhdistelmämalleilla voi parantaa mallien suorituskykyä merkittävästi riippumatta siitä, käytetäänkö perinteisiä vai syväoppimismenetelmiä

(Huda ym., 2018; Laradji ym., 2015). Tämä viittaa siihen, että algoritmien vertailun lisäksi tulisi kiinnittää huomiota koko ennustusprosessiin.

Lisäksi projektien välinen vaihtelu vaikuttaa siihen, kuinka hyvin eri menetelmät toimivat eri ympäristöissä. Siirto-oppimisen avulla voidaan pyrkiä hyödyntämään aiemmista projekteista saatua tietoa, mutta sen toimivuus ei ole itsestään selvää kaikissa tilanteissa (Nam ym., 2013; Jiang ym., 2020). Tämä korostaa sitä, että menetelmien vertailussa tulisi huomioida myös niiden yleistettävyys eikä pelkästään suorituskyky yksittäisessä aineistossa.

Näin ollen voidaan todeta, että menetelmien välinen vertailu ei ole yksiselitteinen, vaan eri lähestymistavat tarjoavat etuja eri tilanteissa. Perinteiset koneoppimismenetelmät voivat olla käytännöllisempiä ja helpommin sovellettavia, kun taas syväoppiminen tarjoaa mahdollisuuksia käsitellä monimutkaisempaa dataa. Lopullinen valinta riippuu käytettävissä olevasta datasta, projektin vaatimuksista sekä siitä, kuinka tärkeää mallin tulkittavuus on käytännön sovelluksissa.

5.3 Tutkimusaukot ja jatkotutkimus

Kirjallisuuskatsauksen perusteella voidaan tunnistaa myös useita tutkimusaukkoja, jotka tarjoavat mahdollisuuksia jatkotutkimukselle. Yksi keskeinen tutkimusaukko liittyy mallien yleistettävyyteen eri ohjelmistoprojektien välillä. Vaikka siirto-oppimista on tutkittu (Nam ym., 2013; Jiang ym., 2020), sen toimivuus riippuu edelleen vahvasti projektien samankaltaisuudesta. Tämä tarkoittaa, että tarvitaan lisää tutkimusta siitä, miten malleja voidaan soveltaa luotettavasti erilaisissa ympäristöissä.

Toinen tutkimusaukko liittyy datan laatuun ja saatavuuteen. Useat tutkimukset perustuvat julkisiin dataset-kokoelmiin, mutta teollisissa projekteissa data voi olla puutteellista tai vaikeasti saatavilla. Tämä rajoittaa erityisesti syväoppimismenetelmien käyttöä, koska ne vaativat usein suuria aineistoja toimiakseen tehokkaasti (Wang ym., 2018). Jatkossa

olisi tärkeää tutkia menetelmiä, jotka toimivat luotettavasti myös pienemmillä tai epätäydellisillä aineistoilla.

Lisäksi tulkittavuus nousee esiin keskeisenä kehityskohteena. Vaikka syväoppimismallit voivat tarjota hyviä ennustustuloksia, niiden “musta laatikko” -luonne voi vaikeuttaa niiden hyväksymistä käytännön käyttöön. Madeyski ja Stradowski (2023) korostavat, että teollisissa sovelluksissa mallien tulee olla ymmärrettäviä ja luotettavia. Tämä viittaa siihen, että tulevaisuudessa tarvitaan enemmän tutkimusta selitettävistä koneoppimismenetelmistä, jotka yhdistävät hyvän suorituskyvyn ja tulkittavuuden.

5.4 Yhteenveto diskussiosta

Kokonaisuutena tarkasteltuna kirjallisuus osoittaa, että ohjelmistovirheiden ennustaminen on moniulotteinen ongelma, jossa ei ole yhtä yksinkertaista ratkaisua. Eri menetelmät tarjoavat erilaisia etuja ja rajoitteita, ja niiden soveltuvuus riippuu käytettävästä datasta, projektin ominaisuuksista sekä käytännön vaatimuksista. Tämä korostaa sitä, että menetelmien valinnassa tulisi ottaa huomioon koko kehitysympäristö eikä pelkästään yksittäinen suorituskykymittari.

6 Johtopäätökset ja pohdinta

Tämän tutkimuksen tavoitteena oli tarkastella ohjelmistovirheiden ennustamista koneoppimismenetelmien avulla kirjallisuuskatsauksen muodossa. Työssä selvitettiin, millaisia menetelmiä alalla käytetään, millaisia tuloksia niillä on saavutettu sekä millaisia haasteita menetelmien käyttöön liittyy. Kirjallisuuden perusteella voidaan todeta, että ohjelmistovirheiden ennustaminen on monipuolinen ja kehittyvä tutkimusalue, jossa hyödynnetään sekä perinteisiä koneoppimismenetelmiä että uudempiin lähestymistapoihin kuuluvia syväoppimismalleja (Malhotra, 2015; Wang ym., 2018; Zhang ym., 2021).

Tulosten perusteella ei voida nimetä yhtä selkeästi parasta menetelmää, vaan mallien toimivuus riippuu vahvasti käytettävästä aineistosta, ominaisuuksien valinnasta ja arviointimittareista (Malhotra, 2015). Perinteiset koneoppimismenetelmät ovat edelleen käyttökelpoisia erityisesti tilanteissa, joissa dataa on rajallisesti tai mallin tulkittavuus on tärkeää. Syväoppimismenetelmät puolestaan tarjoavat mahdollisuuksia käsitellä monimutkaisempaa dataa ja saavuttaa hyviä ennustustuloksia, mutta niiden käyttö edellyttää usein suurempia aineistoja ja enemmän laskentatehoa (Wang ym., 2018; Zhang ym., 2021). Näin ollen menetelmän valinta tulisi tehdä tapauskohtaisesti ottaen huomioon sekä tekniset että käytännön näkökulmat.

Keskeisiksi ennustemallien suorituskykyyn vaikuttaviksi tekijöiksi nousivat datan laatu, piirteiden valinta sekä aineiston esikäsittely. Erityisesti luokkaepätasapaino osoittautui merkittäväksi haasteeksi, joka voi heikentää mallien kykyä tunnistaa virheellisiä tapauksia (Huda ym., 2018; Tan ym., 2015). Lisäksi projektien välinen vaihtelu vaikuttaa mallien yleistettävyyteen, mikä tekee ennustamisesta haastavaa tilanteissa, joissa käytettävissä oleva data on rajallista tai poikkeaa aiemmista projekteista (Nam ym., 2013; Jiang ym., 2020). Näiden tekijöiden perusteella voidaan todeta, että ohjelmistovirheiden ennustaminen ei ole pelkästään algoritmien vertailua, vaan kokonaisuus, jossa data ja konteksti ovat keskeisessä roolissa.

Kirjallisuuden perusteella voidaan myös todeta, että koneoppimismallien hyödyntäminen käytännön ohjelmistokehityksessä edellyttää muutakin kuin hyvää ennustustarkkuutta. Mallien tulee olla ymmärrettäviä, luotettavia ja sovellettavissa organisaation käytäntöihin (Madeyski & Stradowski, 2023). Tämä korostaa sitä, että teollisissa sovelluksissa on huomioitava sekä tekniset että organisatoriset tekijät. Esimerkiksi mallien tulkittavuus voi olla yhtä tärkeää kuin niiden suorituskyky, koska kehittäjien on voitava luottaa mallien antamiin ennusteisiin.

Tämän tutkielman perusteella voidaan todeta, että ohjelmistovirheiden ennustaminen koneoppimismenetelmien avulla tarjoaa merkittäviä mahdollisuuksia ohjelmistokehityksen laadun parantamiseen. Samalla voidaan kuitenkin todeta, että tutkimusalue on edelleen kehittyvä, eikä menetelmien käyttö ole täysin vakiintunutta. Tämä näkyy erityisesti siinä, että tutkimustulokset vaihtelevat eri aineistojen ja menetelmien välillä, eikä selkeää yleispätevää ratkaisua ole löydetty (Malhotra, 2015; Zhang ym., 2021).

Tutkielmaan liittyy myös rajoitteita, jotka on hyvä huomioida. Koska kyseessä on narratiivinen kirjallisuuskatsaus, aineiston valinta ei ole täysin systemaattista, mikä voi vaikuttaa tulosten kattavuuteen (Salminen, 2011). Lisäksi tarkasteltujen tutkimusten määrä on rajallinen, eikä kaikkia mahdollisia menetelmiä tai lähestymistapoja ole voitu käsitellä. Näin ollen tuloksia tulee tulkita suuntaa antavina kokonaiskuvan muodostamiseksi.

Jatkossa tutkimusta voitaisiin suunnata erityisesti mallien yleistettävyyden parantamiseen eri ohjelmistoprojektien välillä sekä menetelmien soveltamiseen tilanteissa, joissa dataa on rajallisesti (Nam ym., 2013; Jiang ym., 2020). Lisäksi selitettävien koneoppimismenetelmien kehittäminen voisi parantaa mallien käytettävyyttä käytännön ympäristöissä (Madeyski & Stradowski, 2023). Myös erilaisten yhdistelmämallien ja hybridimenetelmien tutkiminen voi tarjota uusia mahdollisuuksia ohjelmistovirheiden ennustamiseen (Huda ym., 2018).

Kokonaisuutena voidaan todeta, että ohjelmistovirheiden ennustaminen koneoppimisen avulla on lupaava mutta haastava tutkimusalue, jossa eri menetelmien vahvuudet ja rajoitteet tulee huomioida kokonaisvaltaisesti. Menetelmien onnistunut hyödyntäminen edellyttää tasapainoa ennustustarkkuuden, tulkittavuuden ja käytännön sovellettavuuden välillä (Malhotra, 2015; Madeyski & Stradowski, 2023).

Lähteet

Albattah, W. (2024). *Software defect prediction based on machine learning approaches*. Data, 5(4), 86. https://www.mdpi.com/2673-2688/5/4/86

Huda, S., Liu, K., Abdelrazek, M., Ibrahim, A., Alyahya, S., Al-Dossari, H., & Ahmad, S. (2018). *An ensemble oversampling model for class imbalance problem in software defect prediction*. IEEE Access, 6, 24184–24195. https://doi.org/10.1109/ACCESS.2018.2817572

Jiang, K., Zhang, Y., Wu, H., Wang, A., & Iwahori, Y. (2020). *Heterogeneous defect prediction based on transfer learning to handle extreme imbalance*. Applied Sciences, 10(1), 396. https://doi.org/10.3390/app10010396

Laradji, I. H., Alshayeb, M., & Ghouti, L. (2015). *Software defect prediction using ensemble learning on selected features*. Information and Software Technology, 58, 388–402. https://doi.org/10.1016/j.infsof.2014.07.005

Madeyski, L., & Stradowski, S. (2023). *Industrial applications of software defect prediction using machine learning: A business-driven systematic literature review*. Information and Software Technology, 159, 107192. https://doi.org/10.1016/j.infsof.2023.107192

Malhotra, R. (2015). *A systematic review of machine learning techniques for software fault prediction*. Applied Soft Computing, 27, 504–518. https://doi.org/10.1016/j.asoc.2014.11.023

Nam, J., Pan, S. J., & Kim, S. (2013). *Transfer defect learning*. In Proceedings of the 35th International Conference on Software Engineering (pp. 382–391). https://doi.org/10.1109/ICSE.2013.6606584

Qiao, L., & Wang, Y. (2019). *Effort-aware and just-in-time defect prediction with neural network*. PLOS ONE, 14(2), e0211359. https://doi.org/10.1371/journal.pone.0211359

Salminen, A. (2011). *Mikä kirjallisuuskatsaus? Johdatus kirjallisuuskatsauksen tyyppeihin ja hallintotieteellisiin sovelluksiin*. Vaasan yliopisto.

Tan, M., Tan, L., Dara, S., & Mayeux, C. (2015). *Online defect prediction for imbalanced data*. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE) (Vol. 2, pp. 99–108). https://doi.org/10.1109/ICSE.2015.139

Vaasan yliopisto. (2023). *Kandidaattityön tutkimusmetodi: Kirjallisuuskatsaus* [Kurssimateriaali].

Wang, S., Liu, T., Nam, J., & Tan, L. (2018). *Deep semantic feature learning for software defect prediction*. IEEE Transactions on Software Engineering, 44(12), 1267–1293. https://doi.org/10.1109/TSE.2018.2877612

Zhang, J., Zhang, Z., & Chen, G. (2021). *A survey on software defect prediction using deep learning*. Mathematics, 9(11), 1180. https://doi.org/10.3390/math9111180