Pro gradu -tutkielma
Datatieteen maisteriohjelma

SQL-tehtäväyritysten oikeellisuuden ennustaminen koneoppimismenetelmin

Matti Räty

11.6.2020

Ohjaaja(t): Tohtori Juho Leinonen

Tarkastaja(t): Apulaisprofessori Petri Ihantola
Tohtori Juho Leinonen

Helsingin Yliopisto
Matemaattis-luonnontieteellinen tiedekunta

PL 68 (Pietari Kalmin katu 5)
00014 Helsingin yliopisto

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Matemaattis-luonnontieteellinen tiedekunta	Datatieteen maisteriohjelma

Tekijä — Författare — Author

Matti Räty

Työn nimi — Arbetets titel — Title

SQL-tehtäväyritysten oikeellisuuden ennustaminen koneoppimismenetelmin

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Pro gradu -tutkielma	11.6.2020	45

Tiivistelmä — Referat — Abstract

SQL kuuluu suositeltujen oppiaineiden joukkoon tietojenkäsittelytieteestä. Se on tehokas tapa varastoida dataa kontekstista riippumatta. SQL on kuitenkin opittavana aiheena opiskelijoilleen vaikea, ja tämän vuoksi SQL-opetuksen rinnalla käytetään opetusohjelmistoja. Opetusohjelmistojen avulla SQL:ää päästään opettelemaan käytännössä, paikataan suurta oppilaiden määrää opettajien määrään nähden, ja kerätään aineistoa opiskelijoiden suoriutumisesta.

Oppimisohjelmistojen keräämä aineisto oppilaiden suoriutumisesta tarjoaa mahdollisuuden ennustaa opiskelijoiden suoriutumista kurssilla koneoppimismenetelmin. Tämä tutkielma kouluttaa SQL-opetusohjelmiston aineistoilla hyväksi todettuja koneoppimisalgoritmeja malleiksi, jotka osaavat ennustaa osaako opiskelija seuraavalla yrityksellään SQL-harjoitustehtävän oikein. Kyseessä ei ole tehdä mallia joka osaisi tarkastaa SQL-tehtäviä, vaan tarkoituksena on antaa koneoppimisalgoritmien tarkkailla opiskelijoilta muita kerättyjä tilastoja tehtäväyrityksen oikeellisuuden arvioimiseen ilman itse oppilaan antamaa ratkaisua.

Tutkielmassa huomataan useiden koneoppimismallien olevan toimivia tämän tavoitteen saavuttamiseksi. Vastaavia koneoppimismalleja voidaan hyödyntää oppilaiden löytämisessä, joilla on vaikeuksia tehtävien tekemisessä. Tämä tieto on arvokasta esimerkiksi opetusohjelmistoille, jotka pyrkivät antamaan SQL-tehtävien tekijöille vihjeitä hyödylliseen aikaan.

ACM Computing Classification System (CCS):
Applied computing Education → E-learning
Computing methodologies → Machine learning → Learning paradigms → Supervised learning → Supervised learning by classification
Computing methodologies → Machine learning → Learning settings → Batch learning

Avainsanat — Nyckelord — Keywords

SQL, education, supervised learning, classification, support vector machine, random forest, naive bayes

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

Termistö

Suomeksi	Engalnniksi	Selitys
(tietokanta)taulu	(database) table	Tietokannan tai SQL:n tietorakenne, johon on säilötty tietokannan yksittäinen looginen kokonaisuus. Taulu koostuu riveistä ja sarakkeista.
rivi	row	Kuvaa tietokantataulussa yksittäistä säilöttyä data-objektia.
sarake	column	Kuvailee tietokantataulun rivin ominaisuuksia.
(SQL-)kysely	query	Tietokantakyselyllä haetaan tietokannasta rivejä kyselyn määrittämillä rajoitteilla.
(SQL-)lauseke	(SQL) clause	Osa SQL-kyselyä. Esimerkiksi JOIN, ORDER BY tai DISTINCT.
ohjattu oppiminen	supervised learning	Koneoppimisen laji jossa saadut havainnot yritetään luokitella omaan luokkaansa.
aineisto	data	Kokoelma havaintoja.
havainto	observation	Ohjatussa oppimisessa kukin havainto pyritään luokittelemaan johonkin luokkaan. Yksittäiseen havaintoon viitataan vektorilla X_i, jossa i on jokin kokonaisluku yhden ja havaintojen koon välillä.
ennuste, piirre, ominaisuus	feature	Havainnon ominaisuus. Esimerkiksi ihmisen ikä tai pituus.
luokka, vaste	class, response	Ominaisuus joka pyritään ennustamaan havainnon X_i perusteella. Yksittiseen luokkaan viitataan merkinnällä y_i, jossa i on sama kuin havainnolla X_i. Kokoelmaan luokkia viitataan merkinnällä Y.

Suomeksi	Engalnniksi	Selitys
luokittelu, ennustus	classification, prediction	Koneoppimismallin tekemä ennustus tai arvio havainnon X_i arvosta y_i. Ilmaistaan merkinnällä $\hat{f}(X_i)$ tai $\hat{y}_i$.
tosi-positiivinen-luokittelu	true-positive classification	Binäärisen luokittelijan tekemä ennustus havaintoon, joka on oikein luokiteltu positiiviseen luokkaan.
tosi-negatiivinen-luokittelu	true-negative classification	Binäärisen luokittelijan tekemä ennustus havaintoon, joka on oikein luokiteltu negatiiviseen luokkaan.
vale-positiivinen-luokittelu	false-positive classification	Binäärisen luokittelijan tekemä ennustus havaintoon, joka on vihreellisesti luokiteltu positiiviseksi.
vale-negatiivinen-luokittelu	false-negative classification	Binäärisen luokittelijan tekemä ennustus havaintoon, joka on virheellisesti luokiteltu negatiiviseksi.
Matthewsin korrelaatiokerroin	Matthews correlation coefficient	Binääristen koneoppimisluokittelijoiden suorioutumisen arvioimiseen suunniteltu pisteytys [23].
monimuuttuja regressio	multivariate regression	Menetelmä jolla tarkastellaan yhden tai useamman piirteen lineaarista suhdetta vasteeseen.
verrokkiluokittelija	dummy classifier	”Luokittelija” joka esimerkiksi arvaa aina yleisintä luokkaa. Koulutettujen mallien on oltava vähintään yhtä hyviä kuin verrokkiluokittelija.
ylisovittaminen	overfitting	Tilanne jossa koulutettu koneoppimismalli seuraa aineistoa niin tarksti, että uutta aineistoa ennustaessa sen ennustustarkkuus heikkenee.
siirto-oppiminen	transfer learning	Ihmisen oppimista jäljittelevä koneoppimismenetelmä, jossa koulutetun mallin osia voidaan hyödyntää uuden oppimisessa.

1. Johdanto

Relaatiotietokantakieli Structured Query Language eli SQL kuuluu perusopintojen suositeltuihin oppiaiheisiin tietojenkäsittelytieteessä [1] ja ohjelmistoinsinöörien koulutuksessa [5]. SQL:n käyttö on opintojen ulkopuolella työelämässä on yleistä, ja on yksi suosituista tiedon tallentamisen muodoista IT-alalla. SQL-kielet mahdollistavat tiedon rakenteen ylläpitämisen, tiedon noudon ja hallinnoivat tietokannan turvallisuutta [39].

Ohjelmointikielenä SQL on vaikea oppiaine opiskelijoilleen. Vaikeuden selitykseksi on ehdotettu SQL-kielen muista ohjelmointikielistä poikkeavaa muotoilua, ja sitä että SQL näyttää päältäkatsoen yksinkertaiselta [29]. Tutkimuskohteena SQL:n oppiminen ei ole ollut suosittu tieteellisissä piireissä [33], ja on keskittynyt enimmäkseen muutamaan osa-alueeseen. Perinteisesti SQL:n oppimista käsittelevät tutkimukset ovat tarkastelleet opiskelijoiden tekemiä virheitä ja opetustyökaluja. Muun muassa SQL:n opiskelijoille vaikeat osa-alueet ovat antaneet enintään tulkinnanvaraisia tuloksia [33].

SQL:n oppimisen on todettu onnistuvan parhaiten ympäristössä jossa opiskelijat pääsevät itse kokeilemaan SQL:n kirjoittamista. Heidän on saatava ohjausta, päästävä huomaamaan omia virheitään tekemistään tehtävistä ja saatava palautetta [29]. Yksinkertaisin ratkaisu tämän saavuttamiseksi olisi, että opettaja tai muu SQL:n asiantuntija tarkastaisi opiskelijoiden tehtävät. Ihmisten tarkastamat tehtävät eivät tue suurta määrää opiskelijoita, koska yksittäiset tehtävät ovat työläitä tarkistaa: kuhunkin tehtävään voi olla useita kirjoitustapoja ja virheet eivät aina ole ilmeisiä. Opiskelija-opettaja mittasuhteiden ollessa epätasapainoiset, opiskelijoiden tärkeä yksilöllinen palaute vaarantuu ja viivästyy.

Opiskelijoiden yksilöllistä SQL-opetusta on automatisoitu tämän vuoksi erilaisien ohjelmistojen voimin [29]. Näiden ohjelmistojen avulla voidaan muun muassa vähentää opettajien työtuntien määrää SQL:n opetuksessa ja tarjota opiskelijoille välitöntä palautetta tehtävistään. Koska ohjelmistoilla voidaan haluta käyttää myös opiskelijoiden arvosanojen osana, ohjelmistojen käytön sivuvaikutuksena ne voivat tarjota aineistoa tehtävien tekemisestä. Tämä kerätty aineisto tarjoaa mahdollisuuden tutkia tarkemmin opiskelijoiden oppimista ja suoriutumista tietokantakursseilla.

Tämä tutkielma hyödyntää juurikin tällaista aineistoa. Aineisto on kerätty Helsingin yliopiston järjestämältä ”tietokantojen perusteet” [17]-kurssilta. Kyseinen kurssi

kuuluu Helsingin tietojenkäsittelytieteen perusopintoihin, ja sen tarkoitus on olla opiskelijoiden ensikosketus tietokantoihin. Itse aineisto koostuu opiskelijoiden tekemistä tehtäväyrityksistä kurssin omaan harjoitteluohjelmistoon, ja yrityksistä koostetuista tilastoista.

Hyödynnän tätä aineistoa tutkiakseni mahdollisuuksia ennustaa opiskelijoiden suoriutumista tietokantatehtävien parissa koneoppimisen menetelmillä. Poimin tehokkaaksi todettuja koneoppimisalgortmeja aineiston yleistämiseksi ja koulutan näistä koneoppimismalleja ennustamaan, että onko opiskelijan seuraava tehtäväyritysoikein. Tarkoituksena ei ole tehdä koneoppimismallia joka olisi SQL-tarkastaja. Tämä siis tarkoittaa että koneoppimismalleja kouluttaessa ei käytetä tehtäväyritysten vastausta eikä tehtävän oikeaa ratkaisua.

Koska kyse on luokitteluongelmasta eli onko annettu tehtäväyritys oikein vai ei, ovat valitut koneoppimisalgoritmit ovat ohjattua koneoppimista. Tutkielman käyttämät koneoppimisalgoritmeihin lukeuteuu naive Bayes-luokittelijan variaatioita, tukivektorikone-luokittelijoita eri ydinfunktioilla ja random forest-luokittelija, sekä linear discriminant analysis-luokittelija. Käytetty aineisto suodatuksen päätteeksi on lähes 187 000 riviä, joka ei anna perusteita neuroverkon kouluttamiseen. Neuroverkot hyötyvät suuremmasta aineistomäärästä [13].

Kappaleeseen on koottu tutkielmassa esiintyviä termejä. Kullekin termille on annettu englanninkielinen sana, josta kukin termi on käännetty suomeksi, ja lyhyt kuvaus termistä. Kappale 2 aloittaa selittämällä aiemmasta tietokantojen opetukseen kohdistuneesta tutkimuksesta ja esittelemällä SQL-tietokantakielen. Kappaleessa lähdetään avaamaan koneoppimista ja koneoppimismallien arvioimista. Kappale lopettaa selittämällä tutkimuksesta, jota on aiemmin tehty kurssimenstyksen ennustamisesta. Kappaleessa 3 kuvaillaan tutkielman aineistoa, ja annetaan tarkempi kuvaus tutkielmassa käytetyistä menetelmistä. Tämän tutkielman tutkimusta käsitellään kappaleessa 4. Kappaleessa kuvaillaan aineiston esikäsittelyä, ja esittelee koneoppimisalgoritmien toteutusta aineistoon. Kappele viimeistelee kertomalla tuloksista ja koulutettujen koneoppimismallien rajoituksista.

2. Taustaa

2.1 Tietokannat ja hallinnointijärjestelmät

Tietokanta on kokoelma yhtenäistä tietoa joka on tallennettu siten, että se on monen käyttäjän saatavissa erilaisiin tarkoituksiin [25]. Tietokannan luomiseen liittyy jonkin yhteisön tarpeeseen tallentaa ja hakea tietoa [17]. Esimerkisi ohjelmointikurssilla on syytä tallentaa tietoa opiskelijoiden tehtävien etenemisestä. Tärkein osuus tietokantojen suunnittelussa on oleellisten käsitteiden tunnistaminen. Näin tunnistetaan tiedon säilömistä tarvittavat osa-alueet [17]. Tietokannoissa näitä käsitteistä ilmaistaan kentillä (field), tietueilla (record), tiedostoilla (file) sekä avainkentillä (key field) [25].

Kenttä on tietokannan pienin mahdollinen tietoa sisältävä kokonaisuus. Se voi sisältää esimerkiksi merkkijonoja tai numeroita. Esimerkiksi verkkosivun käyttäjiä sisältävässä tietokannassa yksittäinen kenttä voisi sisältää käyttäjän nimen.

Tietue on kokoelma toisiinsa liittyviä kenttiä, jotka yhdessä muodostavat yhden tietokokonaisuuden. Esimerkiksi tietue verkkosivun käyttäjästä voi sisältää käyttäjänimen, sähköpostin ja nimen sisältävät kentät. Kullakin tietueella on uniikki tunniste eli avainkenttä. Avainkentän avulla manipuloidaan tietokantaa ja noudetaan tietueita tietokannasta.

Tietueet jotka ovat samanmuotoisia ryhmitellään tiedostoihin. Esimerkiksi tiedosto käyttäjistä sisältää kaikki käyttäjiä kuvaavat tietueet. Saman järjestelmän tiedostot yhdessä muodostavat tietokannan. Samaan tietokantaan voi kuulua esimekriksi tiedostot eri käyttäjistä, rooleista sekä käyttäjien lähettämistä viesteistä.

Tietokantojen ylläpitoon käytetään tietokantahallintajärjestelmiä (database management system). Tietokantahallintajärjestelmien ominaisuuksiin kuuluu muun muassa tiedon eheyden valvominen ja käyttöoikeuksien valvominen [17]. Esimerkkejä tietokantahallintajärjestelmistä ovat muun muassa PostgreSQL ja MySQL, jotka käyttävät omaa murrettaan SQL-kielestä.

Tämä opinnäytetyö keskittyy relaatiotietokantoihin. Relaatiotietokannoissa tieto esitetään relaatioina, jotka voidaan ilmaista taulukkoina [17]. Yleisemmin relaatioita kutsutaan tietokantatauluiksi, ja edustavat tietokannan termistössä tiedostoa. Relaatiotietokannassa rivit edustavat tietokannan tietueita. Kullakin rivillä on attribuutte-

ja, joita edustaa taulukon sarakkeet. Taulukko eli yleisemmin tietokantataulu edustaa tietokannan tiedostoa.

2.2 Structured Query Language

Structured Query Language eli lyhyemmin SQL on relaatiotietokantojen [15] hallinnoimiseen tehty ohjelmointikieli [17]. Sen avulla tietokantahallintajärjestelmissä voi luoda ja määritellä tietokantojen rakennetta, noutaa tietoa tietokannasta ja pyrkiä takaamaan tietokannan turvallisuus [39].

Tietokantataulujen (eli tiedostojen) luominen, tiedon syöttämisen säännöt ja rivien (eli tietueiden) noutaminen tehdään SQL-ohjelmointikielellä. SQL-kieli on luonteeltaan deklaritiivinen ohjelmointikieli [29], eli kirjoitetuilla kyselyillä pyritään kuvailemaan haluttua lopputulosta. Seuraavana on esimerkki yksinkertaisesta tietokantakyselystä.

```
SELECT name, breed, age
FROM Dogs
WHERE age BETWEEN 2 AND 5;
```

Tässä esimerkissä tietokantataulusta *Dogs*, etsitään rivejä joiden sarakkeiden *age*-arvot ovat lukujen 2 ja 5 välillä. Näistä riveistä pyydetään tuloksena saatavaan tauluun sarakket *name*, *breed*, sekä *age*.

SQL-kielen näennäisestä yksinkertaisesta syntaksista huolimatta se on voimakas kyselykieli, joka mahdollistaa monimutkaisiakin kyselyjä tietokantaan.

2.3 SQL:n oppiminen

SQL on kirjattuna suositelluksi opeteltavaksi aiheeksi tietojenkäsittelytieteen- [1] ja ohjelmistoinsinöörien [5] perusopinnoissa. SQL on laajan käyttönsä vuoksi helposti sovellettavissa opintojen ulkopuolella IT-alalla.

Laajasta käytöstä huolimatta SQL vaikuttaa olevan vaikea oppia. Vaikeuksien on arvioitu johtuvan SQL:n deklaratiivisesta luonteesta, ja SELECT-kyselyjen näennäisestä yksinkertaisuudesta [29].

SQL:n opetus on saanut varsin vähäistä huomiota tieteellisissä piireissä [33]: esimerkiksi SQL:n vaikeiden osa-alueiden tutkimuksen tulokset ovat tulkinnanvaraisia [33]. Perinteisesti SQL:n opetuksen tutkimus on kohdistunut opetustyökaluihin ja opiskelijoiden tekemiin virheisiin [33]. Opetustyökaluihin lukeutuvat esimerkiksi erilaiset e-oppimisohjelmistot.

2.3.1 E-oppiminen

E-oppiminen on ollut tutkimusaiheena kasvavassa suosiossa ainakin vuodesta 2000 alkaen. Se on herättänyt kiinnostusta niin työmaailmassa kuin akateemisissa piireissä. E-oppimisen käyttöä perustellaan mahdollisuudella oppia aktiivisesti tekemällä, sen sijaan että opiskellaan asioita passiivisesti lukemalla [11].

E-oppiminen on jaoteltavissa kolmeen luokkaan [29]: hallintajärjestelmiin (management system), yhteisölliseen oppimiseen (collaborative learning) ja kohdennettuihin työkaluihin. Hallintajärjestelmiin lukeutuvat esimerkiksi sovellukset Moodle* ja Blackboard†. Yhteisölliseen oppimiseen kuuluu erilaiset kommunikointitavat, esimerkiksi foorumit, video-konferenssit tai sähköposti. Kohdennetut työkalut ovat puolestaan esimerkiksi simulaattoreita, interaktiivisia työkaluja sekö monikysymys-tietopankkeja (Multiple Choice Question bank).

Kun harkitaan e-oppimisen käyttöönottoa on huomioitava muun muassa onko teknologian käyttö pedagogisesti ja opetuksellisesti arvokasta käyttökohteessa [29]. Olennaista itse työkalun kehityksessä on käyttökohteen huomioiminen suunnittelussa siten, että se tukee oppimista mahdollisimman hyvin. Esimerkiksi luonteva navigointi opetusympäristössä parantaa merkittävästi opetustyökalun käyttöastetta ja sen tarjoamaa hyötyä [11]. Koska e-oppimistyökalut korostavat itseopiskelua, korostuu opettajan läsnäolon puutteessa tarve ymmärtää opiskelijan käsitystä oppimisesta [29].

Onnistuneesti toteutettuna e-oppimistyökalut tukevat opiskelijoittensa syväoppimista opettamastaan aiheestaan [29]. Syväoppimisessa opiskelija lähestyy ongelmaa tai tehtävää tarkoituksenmukaisesti siten, että opiskelija työstää itse konseptia sen sijaan, että tarkastelisi vain aiheen pinnallisia yksityiskohtia [10]. Syväoppiminen e-oppimistyökaluissa onnistuu esimerkiksi siten, että aloitetaan selittämällä käsite, jonka avulla opiskelija joutuu ratkaisemaan tehtävän e-oppimistyökalussa.

2.3.2 SQL opetustyökalut

Parhaiten SQL:n oppimisen on havaittu toteutuvan ohjatuissa opetussessioissa ja siten, että opiskelijoilla on mahdollisuus huomata virheitään ja oppia virheistään [29]. Yksi tapa näiden saavuttamiseen ovat tehtäväkokoelmat, jotka tarkistaa SQL:n ammattilainen.

Ratkaisuna tämä on työläs muun muassa suuren opiskelijamäärän ja itse tuloksen tarkistamisen vaikeuden vuoksi. Kyselyjen tarkastaminen voi olla hidasta, sillä saman taulun saamiseen voidaan kirjoittaa erilaisia kyselyitä. Ratkaisun ollessa työläs vaarantuu opiskelijoiden saama yksilöllinen palaute [29].

*https://docs.moodle.org/38/en/Features

†https://www.blackboard.com/about-us

Kurssien ohjaajien työmäärän vähentämiseen e-oppimistyökalut ovat luonteva ratkaisu. SQL e-oppimistyökalut, kuten esimerkiksi SQLator [29] tai AsseSQL [27] kuuluvat e-oppimistyökalujen kolmanteen luokkaan eli kohdennettuihin työkaluihin [29].

Kehitettyjen SQL e-oppimistyökalujen on todettu parantavan opiskelijoiden kurssiarvosanaa, ja saavat käyttäjiltään hyvää palautetta [11]. Suuntaa antava vertailu SQL:n kontekstissa saatiin vertailemalla kahden perättäisen vuoden tenttiarvosanajakaumaa, jossa jälkimmäisenä vuotena käyttöön otettiin SQLator [29]. Tenttiarvosanajakauma parani vuonna jona SQLator otettiin käyttöön.

Kohdennetut e-oppimistyökalut jotka on toteutettu SQL:n oppimiseen, vähentävät ihmisen työn tekemää määrää vaarantamatta oppijoiden saamaa palautetta. Koska tarkistuksen tekee oppimistyökalu, tapahtuu kyselyn oikeellisuuden tarkistaminen nopeasti antaen välitöntä palautetta oppijalle.

SQLator on esimerkki kohdennetusta e-oppimistyökalusta [29]. Sen kuvaillaan olevan verkko-oppimis työpöytä (online learning workbench). Käytännössä se on interaktiivinen oppiympäristö verkko-pohjaisella käyttöliittymällä, joka mahdollistaa pääsyn oppimisympäristöön kaikkialta internet-yhteyden päästä. SQLatorin kehityksen motivaationa on SQL-kielen laaja käyttö, ja SQL kyselyjen arvioinnin laskennallinen monimutkaisuus.

Kyselyjen tarkastamisen ollessa SQLator:n tärkein ominaisuus tarjoaa se lisäksi oppijalle oppimateriaalia, turvallisen ympäristön vapaalle harjoittelulle sekä yhteyden opettajiin. Opettajat saavat mahdollisuuden luoda ja muokata tietokantoja ja tietokantakohtaisia tehtäviä. SQLator kerää opiskelijoiden oppimisprosessin dataa, esimerkiksi lokeja etenemisestä ja tehtävien suoritusyrityksistä. Tämä kerätty data mahdollistaa tilastojen keräämisen ja opiskelijoiden tarkkailun sekä plagiarismin tunnistamisen.

SQLatorin lisäksi SQL-kyselykielen opetteluun on olemassa useita e-oppimistyökaluja. Yksittäinen opetustyökalu ei yksinään välttämättä tarjoa opetuksen vaatimia ominaisuuksia. Yksi mahdollisuus on integroida olemassa olevia työkaluja, joissa tuodaan useampi opetusympäristö yhteen käyttöliittymään. Tutkimus olemassa olevien opetustyökalujen yhdistämiseen on vähäistä, ilmeisesti integroitujen työkalujen teknisestä vaikeudesta johtuen [11].

Integroitujen työkalujen helppokäyttöisyyden varmistamiseksi on Brusilovskyn ym. [11] mukaan mahdollistettava kertakirjautuminen (single sign-on) palveluun, opiskelijoiden toimintojen seuraaminen ja tallennettujen tietojen saatavuus. Tallennetun tiedon on tärkeää olla muodossa, josta opiskelijoiden osaamista on mahdollista päätellä.

2.3.3 Tutkimusta SQL:n oppimisesta / Opiskelijoiden virheet SQL:ssä

Taipalus ym. [33] tekivät tutkimusta opiskelijoiden tekemistä virheistä SQL-opetuskurssilla. Data kerättiin heidän tekemässä verkkosovelluksessa, jossa tavoitetaulu oli näkyvillä koko tehtävän tekemisen ajan. Kun tehtävän tekijä oli yritysten jälkeen tyytyväinen kyselyynsä, hän palautti vastauksen. Eniten heitä kiinnosti pysyvät virheet (persistent error), eli virheet jotka esiintyivät palautetuissa tehtävissä.

Taipalus ym. [33] jakoivat tehdyt virheet neljään yläluokkaan: sekaannuksiin (complication), loogisiin-, semanttisiin- sekä syntaktisiin virheisiin.

Kyselyt joissa on syntaksisia virheitä palauttavat virheilmoituksen ilman tietokantataulua. Syntaktiset virheet ovat käytännössä kirjoitusvirheitä jotka SQL-kielen ympäristö itse havaitsee. Ne ilmenevät muun muassa määrittelemättönä tietokantaobjekteina, väärään järjestykseen asetettuina kyselyinä tai yksinkertaisesti puuttuvana puolipisteenä.

Semanttiset virheet ovat syntaktsiesti oikein eli ne palauttavat tietokantataulun. Niiden tunnistaminen on ilmeistä, vaikka ei kyselyn tavoitetta tietäisikään. Usein ne ilmenevät joko puuttuvina tai toistuvina tietokanta-riveinä. Semanttisesti virheellisillä kyselyillä ei ole käyttökohdetta.

Taipalus ym. [34] esitti esimerkin semanttisesta virheestä, jossa kyselyssä etsittiin työntekijöitä, joilla kullakin on yksi omaan työtehtävään liittyvä titteli. Tehtävänä oli etsiä managerin ja toimistovirkailijan asemassa olevat työntekijät. Kyselyllä joka täyttää ehdon ”toimistovirkailija AND manageri”, saadaan tulokseksi aina tyhjä taulu. Tämä johtuu kyseisen tietokannan työntekijä tietokantataulun rajoitteesta, jossa kullakin työntekijällä on yksi titteli, eli yksikään tietokantarivi ei täytä ehtoa jossa joku työntekijä olisi molempien toimistovirkailijan ja managerin asemassa. Tämä on esimerkki semanttisesta virheestä, jossa ilmeisesti sekoitetaan puhutun kielen ja logiikan ”ja”. Kyselyllä ”toimistovirkailija OR manageri”, tässä tapauksessa saataisiin haluttu tulos.

Kuten semnattiset virheet, myös loogiset virheet ovat syntaktisesti oikeita. Niiden tunnistamiseen vaaditaan käsitys ja ymmärrys kyselyn taivoitteesta. Virheet joissa on loogisia virheitä voivat olla hyödyllisiä johonkin toiseen käyttötarkoitukseen. Jatkaen edellistä esimerkkiä [34], jos tavoitteena on etsiä tietokannan toimistovirkalijoita ja managereita rajoituksella ”toimistovirkalija OR manageri”, niin virhettä ei ole. Jos kuitenkin tämä ei ole tavoitteena, kyseessä on looginen virhe.

Sekaannukset ovat hieman eri asia loogisiin virheisiin nähden. Sekaannukset eivät vaikuta itse tuloksena saatavaan vastaustauluun. Samalla tavalla kuin semanttiset virheet, ovat sekaannukset ilmeisiä huomata: vastauksena saatava taulu on oikein, mutta

itse kysely voitaisiin muodostaa jollakin yksinkertaisemmalla tavalla. Esimerkkejä sekaannuksista ovat esimerkiksi ylimääräinen SELECT-kysely, tai alikyselyssä sijaitseva ORDER BY-lauseke [34].

Yleisin virhe kaikissa SQL kyselyissä ovat syntaktiset virheet [33, 6]. Luonnollisesti syntaktisten virheiden määrä oli suurempi kurssin alussa, ja vähenivät kurssin edetessä. Ahadi ym. [6] huomauttivat syntaktisten olevan yleisin virhe, jossa opiskelija päättää luovuttaa jos hän ei saa virhettä ratkaistua. Yleisimmät pysyvien virheiden luokat ovat puolestaan loogiset virheet ja sekaannukset [33]. Loogiset virheet ovat vaikeita ratkaista, ja tämä oli tapaus myös Taipaluksen ym. [33] mainitsemalla SQL kurssilla. Loogisten virheiden vaikeuden on ehdotettu juontuvat ihmisen työmuistin luonteeseen, ja tapaan jolla ihminen kääntää ajatuksiaan SQL:ään [31].

Tehtävät joiden ratkaisut vaativat jokaista kuutta SQL-kielen lauseketta tuottivat vaikeuksia [33]. Tavallisesti tehtävissä käytettiin kolmesta neljää eri lauseketta. Jotkin käsitteet ja konseptit toivat mukaansa itsellensä tyypillisiä virheitä mukaan tilastoihin. Esimerkiksi monitauluisissa kyselyissä esiintyi JOIN-kyselyille tyypillisiä syntaktisia- ja semanttisia virheitä. Yksittäinen yleisin ja pysyvin virhetyyppi nousi kokoamisfunktioiden (aggregation function) käytöstä, aina kun tehtävässä niitä tarvittiin.

2.4 Koneoppiminen

Koneoppiminen keskittyy kolmeen käsitteeseen, joita ovat havainnot X, vasteet Y ja funktio f [19].

Havainnot ovat käytännössä dataa, josta halutaan päätellä jotakin. Havaintoja X kutsutaan myös opetusdataksi. Tässä tutkielmassa havainnot voivat olla esimerkiksi opiskelijoidan yrityksiä johonkin tehtävään. Kullakin havainnolla $X = (X_1, X_2, \ldots, X_p)$ on p ennustetta. Tehtäväyrityksissä näitä piirteitä voisivat olla esimerkiksi tehtävän aiheen nimi, tai yrityksen ajankohta.

Vaste Y edustaa arvoa, jota pyritään koneoppimisella ennustaa. Tässä tutkielmassa yritetään ennustaa onko seuraava tehtäväyritys onnistunut vai ei. Koneoppimista soveltaessa oletetaan että kullakin havainnon ennusteella on jonkinlainen suhde vasteeseen Y. Tätä suhdetta ilmaistaan seuraavalla yhtälöllä [19].

$$Y = f(X) + \epsilon \tag{2.1}$$

Yhtälössä ϵ ilmaisee virhettä tai kohinaa vasteen arvioimisessa, jota ei onnistuta sisältämään funktioon f. Funktio f puolestaan edustaa systemaattista tietoa, jota havainto X tarjoaa vasteesta Y. Koneoppimisprosessin alussa funktio f on tuntematon, ja prosessin tavoitteena on se selvittää.

Käytännössä suurin osa koneoppimisesta on menetelmiä arvioida funktiota $f(X)$. Sen arvioiminen on mielenkiintoista muun muassa vasteiden määrän ollessa pieni havaintojen määrään nähden, tai ennusteiden ja vasteen suhteen luonteen selvittämiseksi [19]. Esimerkki tilanteesta jossa ollaan kiinnostuttu vasteen arvioimisesta, voisi olla tilanne jossa halutaan ohjelmointikurssin aikana löytää heikosti menestyvät opiskelijat.

Vastaavasti kun ollaan enemmän kiinnostuneita ennusteiden ja vasteiden suhteesta, tarkasteltaisiin tilannetta jossa ohjelmointikurssi on jo päättynyt. Nyt vasteiden arvioimisen on toissijaista, ja tarkastellaan ennusteita p jotka vaikuttavat vasteeseen Y. Esimerkiksi mitkä tekijät vaikuttavat opiskelijoiden arvosanoihin.

Koneoppimista voidaan jaotella usein ohjattuun- ja ohjaamattomaan oppimiseen. Joissakin koneoppimismenetelmissä on molempia piirteitä [19]. Ohjatussa oppimisessa kullakin havainnolla x_i oletetaan olevan jokin vaste y_i [19]. Koneoppimismallin opetuksen aikana ainakin osalla opetusdatalla on jo oma vasteensa. Ohjatun oppimisen lopputavoitteena on pystyä antamaan tuleville, vielä tuntemattomille havainnoille asian mukainen vaste. Tämä opinnäytetyö tulee keskittymään tähän koneoppimisen koulukuntaan.

Ohjaamattomassa oppimisessa opetusdatassa ei ole vasteita, eikä vasteisiin kiinnitetä huomiota algoritmin opetuksen aikana [19]. Yksi esimerkki ohjaamattomasta oppimisesta on klusteroiminen, jossa annetaan koneoppimis algoritmin ryhmitellä data annettujen havaintojen perusteella.

2.4.1 Koneoppimismallien hyvyyden arvioiminen

Koneoppimismallin menestystä mitataan mallin ennusteiden tarkkuudella. Kuinka hyvin ennustetut vasteet vastaa aitoa vastetta? Riippuen vasteen tyypistä, tähän vastataan hieman eri tavalla.

Jos ennustettava vaste on kvantitatiivinen eli jokin numeroarvo, käytetään usein Mean Squared Error-funktiota [19].

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{f}(X_i))^2 \tag{2.2}$$

MSE vertailee jokaisen havainnon kohdalla ennustetta $\hat{f}(X_i)$ todelliseen arvoon y_i. Koska MSE on kiinnostunut näiden kahden arvon etäisyydestä eikä siitä, että onko erotus negatiivinen, jokainen erotus nostetaan toiseen potenssiin. Mitä suurempi MSE on, sitä epätarkempi koulutettu mallin todennäköisesti on.

Jos vaste on puolestaa kvalitatiivinen eli vaste on esimerkiksi jokin luokka, summataan ennustetun vasteen ja aidon vasteen erotuksen sijasta yksinkertaisesti, että

oliko ennustettu luokka oikea [19].

$$CV_n = \frac{1}{n}\Sigma_{i=1}^{n} Err_i \tag{2.3}$$

$$Err_i = I(\hat{f}(x)_i \neq y_i) \tag{2.4}$$

Tässä jälleen validaatiovirheen ollessa suuri, sitä todennäköisemmin koulutettu malli on suoritukseltaan heikko.

Mallien tarkkuutta voidaan tarkastella mallin koulutuksen ja testaamisen aikana. Koulutuksen aikana tehty validointi tehdään samalla datalla, jolla malli on koulutettu. Testaamisen aikana tehty validointi puolestaan tehdään koulutetulle mallille datalla, jota malli ei ole vielä nähnyt.

Harjoituksen aikana tehty validaatio on enintään suuntaa antava, ja parhaimmillaan voi antaa hyvin tarkkoja ennusteita vasteeksi: antaen siis varsin matalia arvoja. Harjoitusdatan validaatio on suuntaa antava metriikka, koska malli voi oppia myötäilemään harjoitusdataa liian tarkasti, jonka seurauksena on ylisovittaminen. Ylisovittamista avataan kappaleessa 2.4.2.

Tarkkuus ei sellaisenaan ole riittävä kriteeri mallin hyvyyden tarkastelemiseen. Tämä johtuu siitä että tarkkuus pitää kaikkia virhe-arvioita saman arvoisina. Tämä voi koitua ongelmaksi esimerkiksi tilanteessa, jossa käytössä on enemmistöä arvaava luokittelija jolle on annettu todella epätasainen aineisto, jossa esimerkiksi 90% aineistosta luokitellaan toiseen luokkaan.

Binääristä luokkaa ennustaessa ennusteet kuuluvat yhteen neljästä luokasta: tosi-positiivinent (true-positive), tosi-negatiivinen (true-negative), vale-positiivinen (false-positive), vale-negatiivinen (false-negative. Luokan nimen alkuosa viittaa siihen, osuiko ennustettu luokka oikeaan. Nimen jälkimmäinen osa viittaa koneoppimismallin ennustamaan luokkaan. Esimerkiksi tosi-positiivinen kuvaa ennustetta, joka on oikein luokiteltu positiiviseksi. Vastaavasti vale-positiivinen on positiiviseksi ennustettu havainto, joka on todellisuudessa luokaltaan negatiivinen.

Ennustuksen aiheesta riippuen väärin ennustetuilla luokilla voivat olla vaarallisia. Esimerkiksi jos ennustetaan että onko potilas sairastunut vaaralliseen tautiin, on vale-negatiivinen vaarallinen väärin-luokittelu. Jos puolestaan taudin hoito on haitallinen potilaalle, voi vale-positiivinen olla vaarallisempi väärin-luokittelu.

Tämän tutkielman kontekstissa väärin-luokittelut eivät varsinaisesti ole vaarallisia. Opetuksen kannalta haitallisempi olisi olla löytämättä opiskelijoita, joilla on vaikeuksia tehtävien tekemisessä. Eli vale-positiivinen on tässä tapauksessa haitallisempi.

Matthewsin korrelaatiokerroin (Matthews correlation coefficient, MCC) [23] kiinnittää huomiota juuri väärin luokiteltujen ennusteiden tyyppiin. Tämä pistearvo las-

ketaan virheiden luokkien perusteella seuraavaasti.

$$N = TN + TP + FN + FP \tag{2.5}$$

Tässä TN on tosi-negatiivisten tapausten lukumäärä, TP on tosi-positiivinen, FN vale-negatiivinen ja FP on vale-positiivinen. N on siis kaikkien tapausten summa, joka asetetaan laskukaavan muissa osissa jakolaskun nimittäjäksi. Tämän perusteella saadaan laskettua osuuksia aineiston ennusteiden eri luokista.

$$S = \frac{TP + FN}{N} \tag{2.6}$$

S on oikein luokiteltujen positiivisten tapausten ja väärin luokiteltujen negatiivisten luokittelujen osuus kaikista ennusteista. Se siis edustaa ennusteiden määrää, joiden todellinen luokka on positiivinen.

$$P = \frac{TP + FP}{N} \tag{2.7}$$

P on oikein luokiteltujen positiivisten tapausten ja väärin luokiteltujen positiivisten luokittelujen osuus kaikista ennusteista. P edustaa ennusteita, joita luokittelija ennusti positiiviseksi.

Ja lopulta koko kaava on seuraava:

$$MCC = \frac{\frac{TP}{N} - S \times P}{\sqrt{P \times S(1-S)(1-P)}} \tag{2.8}$$

Osoittajaksi asetetaan oikein positiiviseksi luokiteltujen havaintojen osuus, josta erotetaan positiivisten luokkien osuus ja positiiviseksi ennustettujen havaintojen osuuden kertolasku. Nimittäjäksi asetetaan kertolasku, jossa on mukana positiivisesti ennustettujen havaintojen osuus, todellisten positiivisten osuus, todellisten negatiivisten osuus sekä negatiiviseksi ennustettujen havaintojen osuus. Tästä kertolaskusta otetaan juuri.

MCC:n antama pisteytys on +1:n ja -1 välillä. MCC arvo +1 tarkoittaa täydellistä ennustusta, ja vastaavasti -1 tarkoittaa täydellisen vastakkaista ennustusta. Heikoin arvo minkä MCC voi antaa on nolla. Käytännössä luokittelija joka saa arvokseen nolla, pelkästään arvaa ennustamaansa luokkaa.

F_1 pisteytys (F_1 score) on toinen tapa arvioida binääriluokittelijan suoriutumista [30]. F_1 hyödyntää myös vale-positiivisia, vale-negatiivisia ja muita vastaavia arvoja. F_1 pisteytys lasketaan täsmällisyyden (precision) ja takaisinkutsun (recall) perusteella.

Jos binääriluokittelussa arvot jotka ovat positiivisia, pidetään ”olennaisina” arvoina, täsmällisyys edustaa osuutta siitä että kuinka moni koneoppimismallin valitsemista arvoista on olennaisia. Toisin sanoen täsmällisyys on tosi-positiivisten arvojen

osuus kaikista positiivisiksi luokitelluista arvoista.

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

Takaisinkutsu edustaa osuutta luokitelluista arvoista sitä että montako oleellista arvoa valittiin. Eli se on tosi-positiivisten arvojen osuus kaikista positiivisista arvoista.

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

Perinteinen F_1 pisteytys eli painottamaton F_1 pisteytys on täsmällisyyden ja takaisinkutsun harmoninen keskiarvo [30]. Koska kyse on osuuksista, F_1 pisteytyksen korkein mahdollinen arvo on 1.

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.11}$$

MCC:tä ja F_1 pisteytykstä verrattaessa F_1 pisteytystä pidetään vähemmän totuudenmukaisena binäärisiä luokittelijoita arvioitaessa [14].

2.4.2 Ylisovittaminen

Ylisovittamisessa (overfitting) malli myötäilee harjoitusdataa niin uskollisesti, että se ei osaa ennustaa tarkasti harjoitusdatan ulkupuolelta tulevia syötteitä.

Ylisovittamista silmällä pitäen mallia validoidaan datalla, jota malli ei ole aikaisemmin nähnyt: testidatalla. Testidataan tehdyn validoinnit ovat harjoitusdataan nähden usein heikompia. Jos malli suoriutuu hyvin testidatalla valdoimisesta, pidetään sitä hyvänä merkkinä.

Testidatalla voidaan helposti validoida, jos on saatavilla suuria määriä harjoitusdatan ulkopuolella olevaa dataa. Useimmissa tapauksissa tämä ei ole kuitenkaan mahdollista. Suurta testidataa voidaan yrittää korvata ottamalla koulutusdatasta validointikokoelma (validation-set).

Validointikokoelma (tai validointidata) on käytännössä koulutusdatasta otettu satunnainen otos, jota ei laisinkaan anneta mallin käyttöön koulutuksen aikana. Tätä kokoelmaa käytetään mallin kouluttamisen lopussa mallin tarkkuuden arvioimiseen.

Ristiinvalidointi (cross-validation) vie tätä ajatusta pidemmälle. Harjoitusdata jaetaan k osaan, esimerkiksi $k = 5$ osaan. Nyt koulutetaan koneoppimismalli samalla algoritmilla k kertaa siten, että kullakin kerralla jätetään yksi osa pois harjoitusdatasta, ja ulos jätettyä osaa käytetään validaatiodatana.

Ristiinvalidoinnilla voidaan selvittää kuinka hyvin koneoppimisalgoritmi selviää itsenäisestä datasta. Lisäksi on mielenkiintoista selvittää vaihe, missä validointivirhe on matalimmillaan [19].

Ristiinvalidointi on usein vakio tapa tarkastella koneoppimisalgoritmien suoriutumista [19].

2.5 Kurssimenestyksen ennustaminen

Oli kurssin oppiaiheena mikä tahansa, on kurssin järjestäjien pyrittävä arvioimaan opiskelijoiden suoriutumisesta myös kurssin aikana. Tämä on haastavaa opiskelijoiden tullessa eri taustoilta ja eri opintolinjoilta.

Yksi ehdotettu ratkaisu on ennakkokyselyjen tekeminen kurssin alussa [37]. Tämä potentiaalisesti tarjoaa yleisnäkymän kurssin osallistujista.

Kyselyt ovat kuitenkin hyödyttömiä opiskelijamäärien ollessa yli sata jokaista kurssiohjaajaa kohden: lomakkeiden läpikäyminen vie liikaa aikaa, ja näin apua tarvitsevat löydetään liian myöhään. Tukea tarvitaan mahdollisimman varhaisessa vaiheessa jotta arvosanoihin voidaan vaikuttaa merkittävällä tavalla [8].

2.5.1 Kurssimenestysken ennustaminen ohjelmointikursseilla

Ohjelmointikurssit joiden tarkoitus on olla opiskelijoiden ensimmäinen kosketus ohjelmointiin ovat vaikeita huomattavalle osuudelle opiskelijoista. Tämä johtaa kursseilta pois jäämiseen ja hylättyihin arvosanoihin sekä heikkoihin arvosanoihin [8].

Aloittelijoiden ohjelmointikurssien läpipääsymäärä tutkimusaiheena on kasvattanut kasvattanut suosiotaan vuodesta 2009 alkaen [16]. Kiinnostuksen kasvusta huolimatta maailmanlaajuinen arvioitu läpipääsyjen määrä ohjelmoinnin aloituskursseilla oli noin 68% vuonna 2014 [35].

Ohjelmointikurssien ennustamisen kohteena ovat usein kurssin loppuarvosana, tenttiarvosana sekä tehtävien kurssiarvosana. Yleisimpiä ennustamismenetelmiä ovat tilastolliset menetelmät [16]. Muita ohjelmointikurssien yhteydessä käytettäviä menetelmiä ovat muun muassa datalouhimis- ja koneoppimismenetelmät. Ohjelmointikursseilla data on usein kerätty ohjelmointiympäristön (IDE, Integrated Development Environment) keräämästä tiedosta. Tähän voi sisältyä ohjelmakoodin kääntämisen aikana kerätyt kirjaukset (ts. lokitukset), tai tilannekuvat (snapshot) koko ohjelmakoodista [36, 18, 21].

Tilastollisten menetelmien yhteydessä ennusteiden mallintamiseen on muutamassa tapauksessa käytetty tilakoneita. Tilakone on järjestelmä, jota kuvataan tiloina ja sen tilojen muutoksina, jonka tila riippuu tilakoneen nykyisestä tilasta ja sen saamasta syötteestä. Viitatuimpia tilakoneita jotka pyrkivät mallintamaan ohjelmoivan opiskelijan tehtävien tilaa ovat, Error Quotient [18], Watson Score [36] sekä Normalized Programming State Model (NPSM) [12].

Error Quotient ja Watson Score rakentavat mallinsa oman ohjelmointikielen kääntäjän kääntämis-lokeista, ja näin pyrkivät ymmärtämään aloittelevien ohjelmoijien ohjelmointikääntämiskäyttäytymistä. Nämä mallit tarkastelevat opiskelijoiden virheellisiä sovelluksien kääntämisyrityksiä, ja vetävät johtopäätöksiä perättäisten kääntämiskertojen välillä. Error quotient on näistä kahdesta yksinkertaisempi. Se lähinnä tarkastelee perättäisten virheiden tyyppiä.

Perättäisten virheiden lisäksi Watson Score ottaa käyttöönsä useampia parametreja. Watson Scoren:n käyttämä pääpiirre on virheiden korjaamiseen kuluva aika. Tätä aikaa verrataan samalla kurssilla oleviin opiskelijoihin, ja pisteytetään tämän mukaan. Jos aikaa kului enemmän kuin vertaisilla, lasketaan kyseisen opiskelijan pisteytystä.

NPSM lienee ottanut innoitusta Watson Score:sta. Kuten Watson Score, myös NPSM tarkkailee kulunutta aikaa. NPSM käyttää kuitenkin tarkempaa mallia ja käyttää ajankohtaisempaa dataa. Mallissa otetaan huomioon, oliko ohjelmakoodissa semanttisia- tai syntaktisia virheitä. Näin NPSM muodostaa holistisemman kuvan kunkin opiskelijan ohjelmakoodin tilasta. NPSM hyödynsi kääntämislokien lisäksi striimattuja otoksia opiskelijoiden koodista, joita opiskelijoiden ohjelmointiympäristät lähettivät, antaen tarkempaa tietoa esimerkiksi ohjelmakoodin syntaktisesta tilasta.

Ennustavina menetelminä heikoin on Error Quotient, ja selkeästi vahvin on NPSM huomioitaessa selittävää varianssia [12]. Error Quotient-mallista puuttuu liikaa tietoa ollakseen enemmän kuin suuntaa antava pisteytys [18]. Vaikka Watson Score on tarkempi Error Quotient:iin nähden, on sen käyttämä data paljon suppeampaa verrattuna NPSM:iin [12]. Carter kuitenkin huomautti mahdollisista eroista kurssiasetelmassa, ohjelmistoympäristöissä ja ohjelmointikielissä [12].

Kullakin tilakoneella rakennettiin ennustava malli lineaarisella regressiolla. NPSM:a sovellettiin myös monimuuttuja regressioon (multivariate regression). Mielenkiintoinen tutkimisen aihe olisi soveltaa näiden tilakoneiden tuottamia parametreja monimutkaisempiin koneoppimismalleihin.

Opiskelijoiden ohjelmointikurssien menestyksen kontekstiin sovellettuja koneoppimistekniikoita on laaja kirjo. Kun kyseessä on akateemisen menestyksen ennustaminen, ovat suosituimpia tekniikoita lineaarinen regressio, ja luokittelualgoritmit [16].

Tarkimpien koneoppimismenetelmien joukkoon kuuluvat muun muassa Naive Bayes [8], tukivektorikoneet [8] sekä random forest [21]. Näistä jokainen on luokittelukoneoppmis-tekniikka. Myös neuroverkot esiintyvät ohjelmoinnin oppimisen ennustamisessa [13].

Koneoppimisalgoritmien ongelmana on usein niiden joustamattomuus: ne on koulutettava datasta, joka on samanlaista kuin itse käytössä oleva data. Tämä pätee erityisesti klassisiin koneoppimisalgoritmeihin kuten esimerkiksi tukivektorikoneisiin ja logistiseen regressioon. Jos tätä ehtoa ei noudateta, ovat ennusteiden tarkkuudet huo-

mattavan heikkoja.

Jos koneoppimisalgoritmi koulutetaan kurssia varten ja kurssin aikana, todennäköisesti ongelmaa ei ole. Tavoiteltavaa on kuitenkin saada aikaiseksi malli, jolla on kohtuullinen tarkkuus edes saman kurssin eri toistokerroilla. Vaikka ohjelmointikurssilla kurssimateriaali, tavoitteet ja opeteltavat asiat pysyisivät samana vuodesta toiseen, on muuttuvia tekijöitä jokaisella iteraatiokerralla useita: muun muassa osallistujien lähtötaso, opettajat sekä opetusmenetelmät. Muuttuvat tekijät pudottavat ennusteiden tarkkuutta merkittävästi.

Saman kurssin eri toistoihin sovellettuja algoritmeja on tutkittu muun muassa neuroverkoilla [13] ja siirto-oppimisella (transfer learning) [21]. Nämä muistuttavat toisiansa rakenteeltaansa, joka jäljittelee aivojen neuronien muodostamaa rakennetta.

2.5.2 Kurssimenestyksen ennustaminen SQL-kursseilla

Kurssimenestyksen ennustamisesta SQL-kurssien kontekstissa oli tausta-artikkelien keräämisen hetkellä varsin vähän. Ainoa artikkeli, jossa suoraan puhuttiin koneoppimisen soveltamisesta oppilaiden menestyksen ennustamiseen SQL-kurssilla on Ahadin ym. kirjoittama ”Students” Syntactic Mistakes in Writing Seven Differente Types of SQL Queries and its Application to Predicting Students” Success” [6]. Tutkimuksen lähestymistapana on luokitella opiskelijat onnistuneisiin ja epäonnistuneisiin heidän kääntämisyritysten perusteella PART-luokittelijalla.

PART luokittelija on sääntöpohjainen luokittelija joka perustuu C4.5 algoritmiin [6]. PART valittiin luokittelijaksi, koska se hallitsee puuttuvat arvot, ja se tarjoaa selkeän esityksen generoiduista säännöistä. Itse kysymys johon luokittelijalla haettiin vastausta, oli selvittää mikä määrä työtä on hyvä ennuste sille, että onnistuuko oppilas tuottamaan oikean kyselyn. Luokittelija koulutettiin ennustamaan, että onnistuuko oppilas tuottamaan GROUP BY-kyselyn oikein.

Luokittelija sai syötteekseen 480 oppilaan yritykset yhteen tehtävään. Puolet syötteeksi valituista oppilaista onnistui vastaamaan kysymykseen ja puolet eivät. Aineistona PART-luokittelijalle annettiin oppilaitten suorittamat kyselyt, sekä virheelliset että oikein menneet. Ennusteita valittiin (feature selection) korrelaatio-pohjaisilla menetelmillä ylisovittamisen välttämiseksi, ennusteiden päällekkäisyyksien vähentämiseksi ja ennustustarkkuuden parantamiseksi. Koulutettu luokittelija validoitiin kymmenkertaisella cross-validation-menetelmällä. GROUP BY-kyselyn ennustamisessa saavutettiin 77% tarkkuus.

Kuten edellissä kappaleessa todettiinkin, Ahadi ym. [6] muistuttavat eri koneopimisalgoritmien olevan voimakkaasti kontekstiriippuvaisia. Samalla tavoitteella koulutetut koneoppimismallit saavuttivat tarkkuuden 60%:n ja 79%:n väliltä.

3. Valitut menetelmät

Tutkielman rakennetaan ja tarkastellaan koulutettuja koneoppimismalleja, jotka ennustavat onko opiskelijan antama tehtäväyritys oikein. Kyseessä on siis binääriset luokittelijat. Koulutetut mallit eivät ole kuitenkaan vastausten tarkastajia. Mallien kouluttamisessa on esimerkiksi jätetty pois itse opiskelija antama SQL-kysely, ja mallit yrittävät muiden piirteiden perusteella ennustaa, että onko tehtäväyritys oikein. Koulutan useamman koneoppimismallin aineiston perusteella ja vertailen näiden suoriutumista keskenään. Eri koneoppimismallien suoriutuminen kertoo myös jotain aineiston luonteesta, koska kullakin algoritmilla on oma lähestymistapansa aineiston yleistämiseen.

3.1 Aineisto

Tämän opinnäytetyön aineisto on kerätty Helsingin yliopiston tietokantojen perusteetkurssilta [17]. Kurssi on tarkoitettu olemaan tietojenkäsittelytieteen kandidaatin tutkielman opiskelijoiden ensimmäinen kosketus muun muassa tietokantoihin ja SQL:ään [17].

Aineisto on kerätty opiskelijoilta jotka kävivät tietokantakurssia vuoden 2019 kevät- ja kesälukukautena. Aineistossa on myös hajanaisia yrityksiä jotka sijoittuvat syyslukukaudelle.

Aineistossa on mukana opiskelijoiden saamat arvosanat ja tietokantakurssin aikana tehtyjen tehtävien yritykset. Kurssin tehtävät perustuvat kurssimateriaaliin, joka puolestaan on jaettu eri aiheisiin. Aiheet voivat koostua esimerkiksi tietokantojen käsitteistä, tai SQL-kielen ominaisuuksista. Kurssiin kuului osuus, jossa opiskelijat loivat uusia harjoitustehtäviä muiden opiskelijoiden tehtäväksi.

Kun opiskelija lähti tekemään johonkin kurssin aiheeseen liittyviä tehtäviä, hänelle valittiin satunnaisesti aiheeseen liittyviä tehtäviä. Osa näistä arvotuista tehtävistä voi olla muiden opiskelijoiden luomia. Jos opiskelija kohtasi tehtävän, jota hän ei pystynyt ratkaisemaan, oli hänellä mahdollisuus arpoa uusi tehtävä samasta aiheesta. Tämä esti tapauksen, jossa opiskelija jää jumiin esimerkiksi virheellisen tehtävän kanssa.

Eniten tehtäviä yritettiin vuoden 2019 periodilla 3, yhteensä 103 149 yrityksen

edestä. Tämä on periodi jolla tietojenkäsittelytieteen kandidaatintutkielman luentokurssi järjestetään. Muilla ajankohdilla todennäköisesti on kyse kurssin verkkoversiosta tai avoimen yliopiston versiosta, joissa ei ole erillistä aikataulua. Tehtäväyrityksistä 31.59% on luokiteltu olevan oikein.

Opiskelijat yrittivät tehtäviä 1399 uniikilla opiskelija-id:llä. Tehtäviä yritettiin yhteensä 197 193 kertaa. Kukin opiskelija yritti tehtäviä keskimäärin 140,9 kertaa. Kukin opiskelija yritti yksittäistä tehtävää keskimäärin 2,8 kertaa.

3.2 Koneoppimisalgoritmit

3.2.1 Bayes luokittelija

Bayes luokittelija perustuu nimensä mukaisesti Bayesin kaavaan [19]. Se laskee todennäköisyyden sille että havainnot kuuluvat johinkin luokkaan. Olettaen että Bayesin luokittelijan ehdot on määritelty oikein, olisi sillä luokittelijoista matalin virheaste [19].

$$Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \tag{3.1}$$

Tämä yhtälön ongelma kuvaillaan yhtäsuuruusmerkin vasemmalla puolella: lasketaan ehdollinen todennäköisyys sille, että havainto $X = x$ kuuluu luokkaan $Y = k$. Tämä on Bayesin kaavan posteriori, ja tähän viitataan jatkossa merkinnällä $p_k(X)$. Osoittajassa lasketaan kyseessä olevan luokkaan k liittyviä arvoja. π_k on Bayesin kaavan priori, jolla kuvataan aineistosta tunnettua ennakkotietoa. Tiheysfunktio $f_k(x)$ ilmaisee kuvailee aineiston muotoa. Nimittäjään summataan vastaavaa tietoa kaikista luokista.

Jotta arvio luokkaan kuulumisesta voidaan tehdä, Bayes luokittelija pyrkii etsimään luokan k joka maksimoi posteriorin $p_k(X)$.

$$p^*(X) = \arg\max_k p_k(X) \tag{3.2}$$

Merkintä $p^*(X)$ on tässä korkein mahdollinen todennäköisyys, joka voidaan savuttaa. Tämä tapahtuu etsimällä luokka k joka antaa suurimman todennäköisyyden Bayesin kaavan posteriorista $p_k(X)$.

Bayes luokittelijan ennakkotietoa luokkien todennäköisyyksistä kuvaa priori π_k. Jos tarkkaa ennakkotietoa todellisesta luokkien todennäköisyydestä ei ole, voidaan se helposti arvioida satunnaisotoksen esiintyvistä luokista: ottamalla kunkin luokan lukumäärän osuus kaikista havainnoista.

Bayesin kaavan tiheysfunktio $f_k(x)$ on käytännössä ehdollinen todennäköisyys siitä että havinto on jokin x ehdolla, että luokka $Y = k$. Siihen viitataan myös merkinnäl-

lä $Pr(X = x|Y = k)$. Se on huomattavasti hankalampaa ratkaista suoraan verrattuna yhtälön muihin osiin. Se on laskennallisesti raskasta ja sen todelliset parametrit ovat todellisuudessa saavuttamattomia ilman kaikkea aineistoa. Todellisuudessa siis tiheysfunktiota ei voi ikinä laskea tarkasti, joten todellista arvoa ei käytännössä edes lasketa. Tästä syystä Bayes luokittelijaa pyritään tavalla tai toisella approksimoimaan [19].

3.2.2 Naive Bayes

Naive Bayes nimensä mukaisesti perustuu Bayesin kaavaan. Se on yksinkertaistettu versio Bayes-optimoidusta luokittelijasta [28].

Koska kyse on ehdollisesta todennäköisyydestä, on havaintojen ennusteiden määrän kasvaessa tiheysfunktion $Pr(X = x|Y = k)$ ratkaiseminen suoraan laskennallisesti raskasta. Tämän vuoksi naive Bayes tekee rajun yksinkertaistuksen: se olettaa että kaikki havaintojen ennusteet ovat toisistaan riippumattomia. Tämä yksinkertaistaa kyseisen ehdollisen todennäköisyyden muotoon $P(\mathbf{X}|C) = \Pi_{i=1}^{n} P(X_i|\ C)$, jolloin saadaan naive Bayesin funktio diskriminantissa muodossa:

$$f_k(X) = Pr(X = x|\ Y = k) = \Pi_{n=1}^{n} Pr(X = x_i|\ Y = k) \tag{3.3}$$

$$f_i^{NB}(\mathbf{x}) = \Pi_{j=1}^{n} P(X_j = x_j|C = i)P(C = i) \tag{3.4}$$

Tässä j on havainnon indeksi, i on luokan indeksi ja n on havaintojen määrä.

Rajusta yksinkertaistamisestaan huolimatta naive Bayes kilpailee suorituskyvyllään hienostuneempien koneoppimismenetelmien kanssa [28]. Parhaiten se toimii havaintojen kanssa joiden ennusteet ovat toisistaan riippumattomia, ja kun ennusteet ovat toisistaan funktionaalisesti riippuvaisia. Funktionaalisessa riippuvuudessa kahden tai useamman ennusteen suhde voidaan ilmaista jollakin funktiolla.

Naive Bayesin rajoituksen ovat nominaaliset koneoppimisennusteet, eli ennusteet joilla on rajallinen määrä mahdollisia arvoja. Binääristen ennusteiden kohdalla naive Bayes voi oppia vain lineaarisia suhteita, ja kun ennusteiden mahdollisten arvojen määrä ylittää kaksi, on havaintoavaruden oltava polynomisesti jakautunut [28].

Opetuksessa naive Bayesia on käytetty esimerkiksi koulutuksellisen datan louhintaan [9], ja opiskelijoiden soveltuvuuden arviointiin kirjallisuuden opinnoissa [16].

3.2.3 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) on approksimaatio Bayes luokittelijasta joka pyrkii yksinkertaistamaan havaintojen ennusteavaruuden lineaariseksi. LDA saavuttaa tä-

män olettamalla tiheysfunktion olevan normaalisti jakautunut, ja estimoimalla havaintojen keskiarvoa μ_k,varianssia σ^2 sekä prioria π_k.

Kun havaintojen ennusteista käytetään yhtä ennustetta p, käytetään tiheysfunktiona normaalijakauman tiheysfunktiota.

$$f_k(x) = \frac{1}{\sqrt{2\pi\sigma_k}} \exp\left(-\frac{1}{2\sigma_k^2(x-\mu_k)^2}\right) \tag{3.5}$$

Normaalijaakuman tiheysfunktio sijoitetaan Bayesin kaavaan 3.1 ja josta otetaan logaritmi. Näin saadaan funktion lineaarinen diskriminantti funktio.

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \tag{3.6}$$

Koska todelliset arvot keskiarvolle μ_k, yhteiselle varianssille σ^2 ja priorille π_k ovat tuntemattomia, approksimoidaan ne seuraavasti ja sijoitetaan ne vastaaviin paikkoihin funktiossa.

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i \tag{3.7}$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 \tag{3.8}$$

$$\hat{\pi}_k = n_k/n \tag{3.9}$$

Ja näillä arvoilla saadaan todellisen diskriminantin funktion estimaatti $\hat{\delta}_k(x)$. Havainto x luokitellaan luokkaan k jolla $\hat{\delta}_k(x)$ on suurin.

Tapauksessa jossa kullakin havainnolla X on p ennustetta, eli $X = (X_1, X_2, \ldots, X_p)$ oletetaan, ettähavainnot riippuvat moniarvo normaalijakaumasta $X \sim N(\mu_k, \mathbf{\Sigma})$. Tässä μ_k on luokan k yhteinen odotusarvo $E(X)$, ja $\mathbf{\Sigma}$ on havainnon X $p \times p$ kokoinen kovarianssimatriisi $Cov(X)$. Kovarianssimatriisi on kaikille havainnoille yhteinen arvo. Moniarvo normaalijakauman tiheysfunktio saa seuraavan muodon.

$$f(k) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} \exp -\frac{1}{2}(x-\mu)^T \mathbf{\Sigma}^{-1}(x-\mu) \tag{3.10}$$

Samalla tavalla kuin yksittäisellä ennusteella p, sijoitetaan kaava 3.10 tiheysfunktion paikalle $f_k = (X = x)$ Bayesin kaavaan 3.1, ja otetaan tästä kaavasta logaritmi, jolloin saavutetaan luokittelun päättävä kaava.

$$\delta_k(x) = x^T \mathbf{\Sigma}^{-1} \mu_k - \frac{1}{2} \mu_k^T \mathbf{\Sigma}^{-1} \mu_k + \log \pi_k \tag{3.11}$$

Arvot $\mu_1, \ldots, \mu_K$, $\pi_1, \ldots, \pi_K$ korvataan arvioilla 3.7 ja 3.9 edellisessä kaavassa ja kovarianssin $\mathbf{\Sigma}$ löytämiseen, kaavan arvion $\hat{\delta}_k$ selvittämiseksi. Päälle näkyvästä moni-

mutkaisuudestaan huolimatta LDA:n vektori-versiokin riippuu vain arvojensa lineaarisesta suhteesta.

LDA:n on toimiva approksimaatio Bayesin luokittelijasta [19]. Sen suurimpana rajoituksena on oletus havaintojen tiheyden funktiosta. Jos havainnot eivät seuraa normaalijakaumaa ovat LDA:n antamat ennusteet tarkkuudeltaan varsin rajallisia.

Opetuksessa on LDA:ta käytetty opiskelijoiden suoriutumisen ennustamiseen tenttikysymyksien perusteella [32] IT-aiheisella kurssilla.

3.2.4 Linear regression

Lineaarinen regressio on yksinkertainen ohjatun oppimisen () sovellus [19]. Se on ollut käytössä pitkään ja se on edelleen laajasti käytetty tilastollinen menetelmä. Sen katsotaan olevan hyvä lähtökohta muiden koneoppimismenetelmien ymmärtämiseen. Iästään huolimatta lineaarinen regressio pystyy vastaamaan varsin joustavasti ilmiön yksittäisten ominaisuuksien välisiin kysymyksiin.

Yksinkertainen lineaarinen regressio olettaa luokan Y ja havainnon X välillä olevan lineaarinen suhde. Tämä suhde voidaan ilmaista seuraavasti.

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{3.12}$$

Yhtälössä ϵ kuvaa virhettä, jota ei yhtälöstä saada poistettua. Se syntyy arvoista, joita ei ole otettu tai pystytä ottamaan yhtälössä huomioon. Sitä voidaan ajatella satunnaisena kohinana, jota aineistosta löytyy. Yhtälössä β_0 edustavaa leikkauspistettä, ja β_1 puolestaan on yhtälön kerroin. Ne ovat arvoja jotka yritetään opettaa funktiolle. Kun aineisto on saatavilla, voidaan yhtälöstä laskea arviot $\hat{\beta}_0$ ja $\hat{\beta}_1$. Näillä arvoilla voidaan antaa ennuste luokasta $\hat{y}$ annetule havainnolle x. Lineaarisessa regressiossa luokka y on nominaalisen luokan sijasta ennuste mahdollisesta arvosta, eli se ei välttämättä ole kokonaisluku.

Yksinkertaisella lineaarisella regressiolla voidaan yrittää ennustaa luokkaa yksittäisen ennusteen perusteella. Usein ennusteita on useita, ja näin yksinkertaista lineaarista regressiota laajennetaan moni-lineaariseksi regressioksi (multiple linear regression) seuraavasti.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon \tag{3.13}$$

Tässä p kuvaa valittujen ennusteiden määrää, numeroidut syötteet $X_1 \ldots p$ kuvaavat kutakin valittua ennustetta, joilla jokaisella on oma kertoimensa $\beta_1 \ldots p$.

Lineaarisella regressiolla saadaan nopeasti aikaiseksi suuntaa antavaa tietoa aineiston ominaisuuksista. Onko esimerkiksi havainnon ja luokan välinen suhde lineaarinen, tai kuinka vahva havainnon ja luokan välinen suhde on. Vastaukset ovat kuitenkin

vain suuntaa antavia. Tarkempina heikkouksina lineaariselle regressiolle mainitaan että sille ei ole luontevaa tapaa käsitellä havaintojen kvalitatiivisia, eli laadullisia ennusteita [19].

Akateemisen suoritumisen ennustamisessa lineaariset mallit, joihin lineaarinen regressio kuuluu, ovat käytetyimpiä. Lineaaristen mallien osuus käytetyistä ennustamisen menetelmistä oli vuonna 2018 31% [16]. Lineaarista regressiota käyttivät muun muassa Carter ym. [12] sekä Watson ym. [36].

Valitsin lineaarisen regression yhdeksi menetelmistä sen yksinkertaisuuden ja laajan käytön vuoksi. Yksinkertaisena mallina se tarjonnee jonkinlaisen vertailukohdan muita koneoppimismalleja soveltaessa.

3.2.5 Support Vector Machine

Tukivektorikone on ohjatun oppimisen menetelmä joka pyrkii jakamaan havaintoavaruuden kahteen osaan. Se laajentaa tukivektoriluokittelijaa mahdollistamalla epälineaarisen luokittelun kahden luokan välillä [19].

Tukivektorikone saa syötteekseen $n \times p$ kokoisen matriisin $\mathbf{X}$, jossa on p-ennustetta, ja n-syötettä. Havaintojen luokat $y_1, \ldots, y_n \in \{-1, 1\}$ ovat binäärisiä, joissa -1 ja 1 ovat mahdollisia ennustettavia luokkia.

Tukivektoriluokittelijan tavoin tukivektorikone pyrkii etsimään kahden luokan välisen rajan eli hypertason (hyperplane), ja saamaan mahdollisimman leveän marginaalin M eri luokkiin kuuluvien havaintojen välille. Kullakin havainnolla x_i on löysäysparametri (slack variable) ϵ_i, joka kertoo havainnon sijainnin marginaaliin ja hypertasoon nähden seuraavasti.

$$\epsilon_i = 0, \text{ niin havainto } x_i \text{ on oikealla puolella marginaalia} \tag{3.14}$$

$$\epsilon_i > 0, \text{ niin havainto } x_i \text{ on väärällä puolella marginaalia} \tag{3.15}$$

$$\epsilon_i > 1, \text{ niin havainto } x_i \text{ on väärällä puolella hypertasoa} \tag{3.16}$$

Vektorit (eli havainnot) jotka ovat marginaalin sisällä tai ovat hypertason väärällä puolella, ovat nimeltään tukivektoreita. Sallittujen löysäysparametrien summaa määrää viritysparametri C, joka käytännössä valitaan ristivalidoinnilla (cross-validation). Löysäysparametrien summa $\sum_{i=1}^{n} \epsilon_i$ ei ylitä viritysparametria C. Viritysparametri säätelee tukivektoriluokittelijan ja tukivektorikoneen vinouma-varianssi tasapainoa (bias-variance trade-off). Siinä missä tukivektoriluokittelijaa käytetään kun havaintoavaruus on lineaarisesti jaettavissa, on tukivektorikone laajennettavissa muunkin muotoisilla

funktioilla. Yleisesti tukivektorikone on ilmaistavissa seuraavasti.

$$f(x) = \beta_0 + \sum_{i \in \mathbf{S}} \alpha_i K(x, x_i) \tag{3.17}$$

Jossa β_0 ilmaisee leikkauspistettä, $\mathbf{S}$ sisältää havaintojen indeksit jotka ovat tukivektoreita, α_i on havainnon x_i kerroin. Funktio K on tukivektorikoneen ydinfunktio (kernel function). Tukivektorikoneen ydinfunktioksi voi asetella muun muassa polynomisen funktion, radiaalisen funktion tai tukivektoriluokittelijan lineaarisen funktion.

$$K(x_i, x_{i''}) = \sum_{j=1}^{p} x_{i,j} x_{i'',j} \text{ Lineaarinen ydinfunktio} \tag{3.18}$$

$$K(x_i, x_{i''}) = (1 + \sum_{j=1}^{p} x_{i,j} x_{i'',j})^d \text{ Polynominen funktio} \tag{3.19}$$

$$K(x_i, x_{i''}) = \exp -\gamma \sum_{j=1}^{p} (x_{i,j} - x_{i'',j})^2 \text{ Radiaalinen funktio} \tag{3.20}$$

Parametri p kertoo havaintoavaruuden ennusteiden määrän. Polynomisessa funktiossa parametri d on polynomisen funktion valittu aste. Radiaalisen funktion γ on positiivinen vakio.

Ydin-funktioiden voimin tukivektorikoneet ovat laskennallisesti yksinkertaisia, ja pystyvät käsittelemään moniulotteista (tai moniennusteista) dataa. Lisäksi tukivektorikoneen kouluttaminen on varsin muistitehokasta, koska ainoastaan tukivektorit, eli havainnot jotka ovat marginaalin sisäpuolella tai hypertason väärällä puolella osallistuvat tukivektorikoneen oppimiseen.

Kuten lineaariregressiossa, tukivektorikone ei tarjoa todennäköisyyksiä luokkien ennustamisen tueksi. Datasta riippuen ydin-funktion valitseminen voi olla haastavaa, varsinkin datassa jossa on rajusti kohinaa. Suuren kohinan määrän läsnäollessa tukivektorikone saattaa ylisovittua kohinaa myötäillen, rampauttaen tukivektorikoneen suoriutumista.

Opetuksen kontekstissa Kentli ym. [20] pyrkivät ennustamaan tekniikan alan kurssin opiskelijoiden suoriutumista tukivektorikoneen avulla. Lagus ym. [21] tekivät vastaavia ennusteita ohjelmointikurssilla. Artikkeli [21] tutki transfer learning-koneoppimismallien suoritumista, jossa tukivektorikone oli verrokkina. Tukivektorikoneet luokittelivat kunnioitettavalla tarkkuudella opiskelijat, jotka pääsivät kurssista läpi. Ennustetarkkuuksien keskiarvot vaihtelivat kurssien edetessä aina 82% ja 87% prosentin väliltä.

Tukivektorikone on lineaariregression jälkeen luonteva jatke. Se on hyvin toteutettavissa ja siitä on olemassa olevia toteutuksia. Tukivektorikone on hyvä yleistämään aineistoa ja pystyy antamaan hyviä ennusteita yksinkertaisuuteensa nähden.

3.2.6 Random Forest

Random forest-menetelmä on kokoelma päättelypuita, ja on laajennus pussittamisesta (bagging) [19] joka puolestaan perustuu bootstrapping-menetelmälle.

Pussittamisen ajatuksena on tehdä useita satunnisia otoksia aineistosta. Tuotettu otos on alkuperäisen aineiston kokoinen. Kun aineistosta tehdään satunnaisia otoksia, voi syöte esiintyä useamman kerran pussitetussa aineistossa, ja osa taas voi jäädä otoksen ulkopuolelle. Aineiston ulkopuolisia syötteitä, eli ”pussin ulkuopuolisia” (out-of-bag) havaintoja hyödynnetään tuotetun mallin virheen arvioimiseen. Pussittamisessa tämä satunnainen otos tehdään B-kertaa, ja jokaiselle aineisto-otokselle tehdään oma päättelypuu. Regressio-ongelmassa päättelypuut antavat oman arvionsa, ja näistä arvioista lasketaan keskiarvo ennusteeksi.

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x) \tag{3.21}$$

Jos kyseessä on luokitteluongelma, saadaan luokan ennuste puiden enemmistöäänellä. [19]

Pussien määrä B:n kasvattaminen ei lisää ennusteiden ylisovittamista [19]. Usein pussien määrän B valinta tapahtuu siten, että sitä kasvatetaan kunnes luokitteluvirhe tasaantuu.

Verrattuna yksittäiseen päättelypuuhun pussittaminen ennustusmenetelmänä on paljon tarkempi. Pussittamisen ongelmana on sama kuin yksittäisellä päätteylypuulla, eli syötteen korreloivat- ja vahvemmat ennusteet. Jos yksittäisellä ennusteella on vahva painoarvo luokittelussa, näyttävät pussittamisen päättelypuut keskenään samanlaisilta. Vastaavanlainen ongelma ilmenee korreloivissa ennusteissa luoden toistuvia rakenteita päättelypuissa.

Random forest ratkaisee ongelmaa vaikuttamalla logiikkaan, että millä valitaan kunkin puun haaran merkitsevä syötteen ennuste. Sen sijaan että syötteen jokaista ennustetta harkitaan kunkin haaran jakavaksi tekijäksi, valitaan jakava ennuste vain satunnaisen m-kappaleen joukosta. Lukumäärän m määrittelee usein $m \approx \sqrt{p}$, eli syötteiden ennusteiden määrän määrän neliöjuuri. Näin random forest dekorreloi syötteiden ennusteita, ja saadaan suurempaa variaatiota luotujen päättelypuiden välille [19].

Ytimessään random forest on kokoelma päättelypuita. Sen ennusteet ovat merkittävästi parempia verrattuna yksittäiseen päättelypuuhun [19]. Koska päättelypuita on kokoelmassa useita ja satunnaisuus on olennainen osa mallin rakentamista, välttää random forest hyvin ylisovittamista [19].

Usean päättelypuun käytön vuoksi random forest kuitenkin menettää yhden päättelypuun suurimmista eduista, joka on selkeä tulkittavuus [19]. Päättelypuussa tuloksen tulkitseminen on yksinkertaista, sillä jokaista tulosta edustaa yksittäinen puun

”lehti”. Jokaisella lehdellä on selkeä reitti ja solmut, joita kautta kyseiseen lehteen on päädytty. Tämän sijasta random forest käyttää jokaisen puunsa antamaa ennustetta, joten ei tulos ole enään yhtä helposti luettavissa.

Opetukessa random forest on ollut mukana ennustamassa muun muassa hyvin ja heikosti suoriutuvia ohjelmointi-opiskelijoita. Yleisesti sitä on tarkasteltu muiden koneoppimismenetelmien rinnalla [7]. Lisäksi se on ollut vertailussa tutkimuksessa transfer learning-menetelmän soveltuvuudesta samaan kontekstiin [21]. Molemmissa artikkeleissa random forest toimii hyvin muihin koneoppimismenetelmiin nähden.

4. Koneoppimismallien valmistelu ja toteutus

4.1 Aineiston esikäsittely

Koska osa tehtävistä on opiskelijoiden tekemiä, on tarkasteltavia tehtäviä suodatettava siten että ne ovat varmasti mahdollista tehdä oikein tehtävän antamilla ohjeilla. Datasta suodatettiin tehtävät, joissa on vähintään viisi onnistunutta yritystä, joka pienensi yritysten määrää datassa.

Päivämäärä saraketta ei saada hyödynnettyä sellaisenaan. Se on purettava merkkijonosta, tai päivämäärä-merkinnästä konkreettiseksi numeroarvoksi. Tässä tapauksessa päivämäärästä saadaan kaksi oleellista: kurssin viikko jolloin yritys on tehty, ja opiskelijan yrityksen järjestysluku tehtävään.

Kurssin viikolla tarkoitetaan ensimmäistä viikon numeroa, laskettuna yksittäisen kurssi-iteraation alkamisesta lähtien. Viikun numero on valittu viikottaisten yritysmäärien perusteella. Ensimmäinen viikko on valittu Helsingin yliopiston periodikalenterista [3, 4].

Opiskelijan yrityksen indeksillä viitataan siihen että monesko kyseinen yritys on opiskelijan yrityksistä samaan tehtävään.

Datassa on merkintä tehtävän luoneesta opiskelijasta, jos opiskelija on luonut annetun tehtävän. Jos asia ei ole näin, id:n tilalla ei ole arvoa. Tämä sarake on muutettu totuusarvoksi, jossa ”False” ilmaisee, että tehtävän on tehnyt opettaja, ja päinvastaisella arvolla joku opiskelijoista.

Koska yhtenä tavoitteena on tarkastella aiheiden merkitystä tehtävien onnistumiseen, on otettava aiemmat tehtävät jollain tavalla huomioon. Jokaiseen yritykseen on lisätty edellisten onnistuneiden tehtävien summa jaoteltuna aiheittain.

Sarakkeeseen ”time_spent” asetettiin kuhunkin tehtävään arvo, joka on tähän tehtäväyritykseen mennessä kulunut aika laskettuna ensimmäisestä yrityksestä samaan tehtävään.

Lopullinen esikäsitelty aineisto sisältää siis 23 saraketta. Koneoppimismalleille

Sarake	Kuvaus
aiheen id	Aiheen tunniste, johon tehtäväyrityksen tehtävä kuulu.
tehtävän id	Tehtävän tunniste, johon tehtäväyritys kuuluu.
tehtävän on luonut käyttäjä	Totuusarvo siitä, onko opiskelija luonut tehtävän.
viikko	Viikon numero, jona tehtäväyritys on tehty.
periodi	Opintoperiodin numero, jona tehtäväyritys on tehty. Kesäperiodit poislukien periodeja on neljä.
periodin viikko	Opintoperiodin viikon numero, jona tehtäväyritys on tehty. Kullakin periodilla on seitsemän viikkoa.
aikaa kulunut	Aika jonka opiskelija on käyttänyt tehtävän ratkaisemiseen tähän yritykseen mennessä.
yritysten järjestysluku	Kuinka mones opiskelijan yritys on tämän tehtävän ratkaisemiseen.
aiheiden summat 1-14	Kuinka monta onnistunutta yritystä opiskelija on tehnyt tähän yritykseen mennessä. Summat on jaoteltuna aiheittain aiheen tunnisteiden mukaan.

Taulukko 4.1: Datan piirteet

annetaan taulukossa 4.1 listatut sarakket. Näistä on jätetty pois käyttäjän tunniste, tehtäväyrityksen tunniste sekä tieto siitä että onko tehtävä oikein.

Mallien validoimiseksi aineisto jaettiin siten, että koulutusaineisto sisältää 70% datasta, ja validointi aineisto sisältää 30% datasta. Koulutusaineiston tehtäväyrityksistä 29.61% on luokiteltu onnistuneiksi. Validointiaineistosta puolestaan 30.21% yrityksistä on luokiteltu onnistuneiksi.

4.2 Toteutus

Toteutettuihin malleihin lukeutuu siis tukivektori-luokittelijat, naive Bayes-luokittelijat, random forest-luokittelija sekä LDA-luokittelija. Luokittelijat koulutetaan ennustamaan, että onko annettu opiskelijan tehtäväyritys oikein.

Kaikkien luokittelijoiden kouluttamisessa käytetään viisinkertaista ristiinvalidointia. Ennen mallien kouluttamista ja ristiinvalidointia, luokittelijoiden suoriutumisen parantamiseksi kokeiltiin ristiinvalidoinnilla eri hyperparametreja. Hyperparametrien tarkat määritelmät menevät tämän opinnäytetyön aihealueen ulkopuolelle.

4.2.1 Verrokkiluokittelija

Verrokkiluokittelijaksi valitsin yksinkertaisen mallin, joka arvaa aina aineiston yleisintä luokkaa annetusta syötteestä riippumatta. Tällainen löytyi scikit-learn [26]-kirjastosta.

4.2.2 Tukivektorikoneet

Koska aineisto on moniuloitteinen, sisältäen 23 eri piirrettä, on tukivektorikoneiden kouluttaminen yhdellä säikeellä hidas prosessi. Tämän vuoksi valitsin mallien kouluttamiseen ThunderSVM [38] kirjaston. Kirjasto on Singaporen yliopiston Xtra Computing-tutkimusryhmän kehittämä tuote.

ThunderSVM tarjoaa mahdollisuuden kouluttaa tukivektorikoneita GPU:lla usealla säikeellä. Ongelmana kohtasin että mallin kouluttaminen saattoi kestää ikuisesti. Kirjaston keskusteluissa ilmeni, että kyse voi olla siitä että malli ei ikinä yhdy (converge), jos mallille annettuja hyperparemtereja ei ole optimoitu.

Hyperparametrit oli mahdollista löytää ristiinvalidoimalla eri arvoilla käyttäen pientä osaa aineistosta. Ristiinvalidointi ajettiin scikit-learn-kirjastolla [26].

Hyperparametrien löydyttyä ThunderSVM-kirjaston mallit saivat koulutuksen päätökseen. Yhden mallin kouluttaminen kesti kahdesta neljään minuuttia, riippuen Google Colab:n [2] tarjoamista resursseista.

Tukivektorikoneet koulutettiin lineaarisella, polynomisella, radiaalisella sekä sigmoidisella ydinfunktioilla.

Lineaarinen ydinfunktio sai löysäysparametrin C arvon 9.5.

Radiaalisen ydinfunktiolla malli koulutettiin löysäysparametrilla $C = 6$, ja gamma-arvolla $\gamma = 0.026$.

Polynominen funktio koulutettiin yksi-asteisena funktiona, joka on lähes samanlainen kuin lineaarinen funktio. Löysäysparametriksi ristiinvalidoitiin $C = 7.08$, ja gamma-arvoksi $\gamma = 0.087$.

Sigmoidisella ydinfunktio koulutettiin 3-asteisena. Löysäysparametriksi valittiin $C = 8.32$, ja gamma-arvo 0.0023.

4.2.3 Naive Bayes

Scikit-learn [26] tarjoaa useita toteutuksia naive Bayes luokittelijasta. Tarjolla olevista luokittelijoista kokeilin komplementti-, Bernoullin-, gaussian- sekä multinomial naive Bayes luokittelijoita.

Yksittäisen mallin kouluttamisessa meni enintään sekunti. Kukin toteutus tekee oman oletuksensa Bayesin kaavan tiheysfunktion muodosta, jota aineisto seuraa. Esimerkiksi gaussian naive Bayes olettaa aineiston tiheysfunktion seuraavan gaussian-

tiheysfunktiota. Kaikki Naive Bayes luokittelijat eivät välttämättä ota vastaan montaa hyperparametria.

Komplementti naive Bayes sai alpha-arvokseen $\alpha = 0.035$. Gaussian naive Bayes koulutettiin *var_smoothing*-parametrin arvolla 2.56. Multinomial naive Bayes malli koulutettiin arvolla $\alpha = 0.132$. Bernoulli naive Bayes koulutettiin parametrilla $\alpha = 0.132$.

4.2.4 Linear Discriminant Analysis

Linear discrimant analysis-luokittelija löytyi scikit-learn-kirjastosta [26]. Mallin kouluttamisessa kesti vajaa kolme sekuntia.

Linear discriminant analysis-luokittelija koulutettiin least-squares-ratkaisijalla. *shrinkage*-parametri jätettiin automaattiseksi, jolloin se seuraa scikit-learn-kirjaston dokumentaation mukaan ”Ledoit Wolfin lemmaa”.

4.2.5 Random Forest

Random forest luokittelija valittiin scikit-learn [26] kirjastosta.

Kunkin random-forestin puun ”puhtauden” kriteerinä käytettiin Gini-epäpuhtaus-funktiota. Puiden määräksi valittiin 779. Out-of-bag arvot eivät parantaneet luokittelijan suoritusta ristiinvalidointia tehtäessä, joten niitä ei hyödynnetty mallin kouluttamisessa.

4.3 Tulokset

Mallien validoimiseen käytettiin 30% aineistosta, jota yksikään malleista ei ole nähnyt koulutuksen aikana. Kaikki mallit antoivat omat ennisteensa validaatio-aineistolle, ja ennusteiden perusteella tehdyt pisteytykset on merkitty taulukkoon 4.2. Taulukkoon merkittiin ennustusten tarkkuus, MCC sekä F1-pisteytys.

Taulukossa 4.2 ”Majority classifier” on käytössä oleva verrokkiluokittelija, joka arvaa aina yleisintä luokkaa.

4.3.1 Pisteytysten nolla-arvot

Verrokkiluokittelija (majority classifier) ja gaussian naive Bayes-luokittelijaa arvioitaessa Scikit-kirjastolla [26], saivat ne arvokseen tasan nolla MCC:llä ja F1-pisteytyksellä laskiessa. Tämä on virhearvo seurausta epätasaisista ennusteiden arvoista jotka johtavat tavalla tai toisella siihen, että arvo nolla vaikuttaa lopputulokseen. Arvo nolla voi esimerkiksi päätyä jakolaskun nimittäjäksi, tai osaksi kertolaskua.

Aineiston ollessa suuri ja luokittelija ennustaa luokkaa heikosti, voi pisteytysalgoritmeihin päätyä todella pieniä negatiivisia arvoja. Näissä tapauksissa scikit-kirjaston [26] käyttämä numpy-kirjasto [24] voi pyöristää näitä hyvin pieniä parametreja nollaan.

Vaihtoehtoisesti nollia löytyy ennusteiden luokkien ollessa epätasaisia. Esimerkiksi verrokkiluokittelijan ennusteet ovat kaikki joko luokkaa negative tai positive.

MCC:n (2.8) kohdalla nolla nimittäjäksi saavutetaan tapauksessa, jossa luokittelija arvaa kaikkia positiivisekis tai negatiiviseksi. Jos kaikki on luokiteltu positiiviseksi, saa kaavan P-arvo (2.7) arvokseen yksi, ja tuo nimittäjään MCC-kaavaan (2.8) arvon nolla. Vastaavasti jos kaikki on luokiteltu negatiiviseksi, sama p-arvo on nolla ja tuo molempiin osoittajaan ja nimittäjään arvon nolla.

F1-pisteytyksessä (2.11) epätasaisissa ennusteissa joko tarkkuus tai takaisinkutsu voi saada arvokseen nolla, antaen pisteytyksen arvoksi tasan nolla.

Koska aineiston yleisin luokka on että tehtävä ei onnistunut, arvaa verrokkiluokittelija aina negatiivista arvoa. Näin F1-pisteytystä laskiessa molemmat tarkkuus (2.9) ja takaisinkutsu (2.10) ovat arvoltaan nollia. MCC:n kohdalla kaavan P-arvo (2.7) on nolla.

Gaussian naive Bayes olettaa aineiston luokkien seuraavan gaussian-jakaumaa. Tämä oletus vaikuttanee tämän luokittelijan antamiin arvioihin siten, että sen ennusteet eivät poikkea suuresti tai ollenkaan verrokkiluokittelijan antamista ennusteista.

4.3.2 Tulokset tarkkuutta käyttäessä

Taulukko 4.2 havainnollistaa miksi tarkkuus antaa enintään suuntaa antavan arvion koneoppimismallien suoriutumisesta. Verrokkiluokittelija arvaamalla aina että tehtäväyritys on väärin, saa tarkkuudekseen lähes 70% ennusteista oikein.

Parhaan tarkkuuden antoi random forest-luokittelija 95.78%:lla. Tämä tulos vastaa koneoppimisvertailussa [7] opiskelu-kontekstissa ja transfer learning-vertailussa [21] esiintyneitä tuloksia, joissa random forest antoi hyviä tuloksia verrokkeihinsa nähden.

Bernoulli naive Bayes-luokittelija ja lineaarinen tukiverkotikoneluokittelija antoivat lähes samoja tarkkuuksia. Ne olivat jäljessä random forest-luokittelijasta muutamalla kymmenellä promille-yksiköllä.

Pienimmän tarkkuuden verrokkiluokittelijan lisäksi antoi gaussian naive Bayes-luokittelija: se antoi täysin saman arvon kuin verrokkiluokittelija eli tarkkuuden 69.79%. Koska tarkkuuden arvo, MCC ja F1-pisteytys olivat gaussian naive Bayes luokittelijalla samat kuin verrokkiluokittelijalla, antaa tämä viitteitä siitä että se loukitteli validaatioaineiston samalla tavalla kuin verrokkiluokittelija. Gaussian naive Bayes-luokittelijan suoriutumisesta voinee ainakin olettaa etteivät käytetyn aineiston luokat

seuraa gaussian-jakaumaa.

Kaksi seuraavaa pienintä tarkkuutta antoivat multinomial- sekä complement naive Bayes-luokittelijat. Molemmat antoivat tarkkuudekseen hieman yli 70%.

Linear discriminant analysis sai tarkkuudekseen 73.14%. LDA:n heikko tarkkuus on osittain selitettävissä sillä, että se tekee oman oletuksensa aineiston tiheysfunktiosta. Samaan tapaan kuin naive Bayes-luokittelijat.

Suuri vaihtelu naive Bayes-sukuisten luokittelijoiden kanssa johtuu kunkin luokittelijan tekemistä oletuksista liittyen aineistoon. Kukin naive Bayes-luokittelija olettaa aineiston luokkien seuraavan nimensä mukaista tiheysfunktiota. Odotettavaa siis on, että yksi naive Bayes variaatioista antaa huomattavasti suuremman tarkkuuden kuin muut. Tässä tapauksessa Bernoulli naive Bayes-luokittelija sai parhaan tarkkuuden, 95.75%, ja siitä seuraavaksi tarkin naive Bayes luokittelija oli multinomial naive Bayes-luokittelija tarkkuudella 70.19%. Suurin osa naive Bayes luokittelijoista olivat siis enintään kahdella prosentilla tarkempia kuin verrokkiluokittelija.

Tukivektorikoneluokittelijoiden välillä on myös odotettavissa vaihtelua. Kunkin tukivektorikoneluokittelijan tarkkuus riippuu siitä onko olemassa jakavaa hypertasoa, joka jakaa aineistoa ydinfunktion muodon mukaan. Jos aineisto ei ole jaoteltavissa ydinfunktion ehdottamalla tavalla, sen tarkkuus kärsii. Verrattuna naive Bayes-luokittelijoihin, vaihtelu tukivektorikoneluokittelijoiden tarkkuuksien välillä on vähemmän eroja. Lineaarinen tukivektoriluokittelija antoi tukivektorikoneista parhaan tuloksen tarkkuudella 95.2%. Seuraavaksi tarkimman antoi polynominen tukivektoriluokittelija tarkkuudella 91.02%. Pienimmän tarkkuuden antoi sigmoidisella ydinfunktiolla varustettu tukivektorikone, tarkkuudella 76.85%.

4.3.3 Tulokset Matthewsin korrelaatiokerrointa käyttäessä

Koneoppimismalleja arvioitaessa tarkkuus (accuracy) jättää huomiotta sen, että minkä tyyppisiä virheitä koulutettu malli tekee ennusteita tehdessään. Tätä piirrettä pyrkivät paikkaamaan Matthewsin korrelaatio kerroin (Matthews correlation coeffcient, MCC), sekä F1-pisteytys (F1 pisteytys). Molemmat pyrkivät vastaamaan samankalatiseen ongelmaan. MCC:tä usein suositaan F1-pistytyksen sijasta [14] binäärisen luokittelijan arvioimisessa. Verratessa MCC:tä F1-pisteytykseen, MCC ei ota huomioon että kumpi luokista on ”positiivinen” tai ”negatiivinen”. Vaikka näiden luokkien tyyppiä vaihtaisi, ei MCC-arvo muutu toisin kuin F1-pisteytyksen kohdalla.

Kuten jo aiemmin havaittiin, verrokkiluokittelija ja gaussian naive Bayes-luokittelija antoivat MCC-arvokseen tasan nolla. Järjestäessä malleja MCC:n mukaan verrattuna tarkkuuden mukaan järjestämiseen, ei suuria muutoksia tapahdu. Enintään järjestyksessä vierekkäiset mallit vaihtoivat paikkaa. Suurin poikkeus tähän oli LDA,

Classifier	Accuracy	MCC	F1 score
Majority classifier	69.79%	0.0	0.0
LDA	73.14%	0.2626	0.3099
Random forest	95.78%	0.9023	0.9321
Naive Bayes			
Bernoulli NB	95.76%	0.9058	0.9338
Complement NB	70.18%	0.3655	0.5793
Gaussian NB	69.79%	0.0	0.0
Multinomial NB	70.19%	0.3651	0.5790
Support vector classifier			
Linear SVC	95.20%	0.8957	0.9262
Radial SVC	87.91%	0.7253	0.8113
Polynomial SVC	91.02%	0.8149	0.8688
Sigmoid SVC	76.85%	0.4131	0.5536

Taulukko 4.2: Mallien tulokset

joka on MCC:n mukaan järjestäessä yhdeksäs, siinä missä se oli tarkkuuden mukaan järjestäessä seitsemäs. MCC:n mukaan järjestäessä Bernoulli naive Bayes-luokittelija on hieman random forest-luokittelijaa tarkempi. Bernoulli naive Bayes-luokittelija sai MCC-arvokseen 0.9058, ja random forest-luokittelija 0.9023. Ne ovat tarkkuuden mukaan järjestäessä toisin päin. Vastaava kävi multinomial naive Bayes- ja complement naive Bayes-luokittelijoiden kohdalla. Tarkkuudessa complement naive Bayes oli tarkempi, ja MCC:n mukaan järjestäessä multinomial naive Bayes oli tarkempi. Ero on näiden välillä on kuitenkin vähäinen.

MCC nostaa esille mallit, jotka eivät pelkästään arvanneet yleisintä luokkaa. Esimerkiksi tarkastellessa multinomial naive Bayes-luokittelijaa ja verrokkiluokittelijaa huomataan, että niiden tarkkuudet ovat lähellä toisiansa. Kuitenkin MCC-arvoa tarkastellessa käy ilmi, että toisin kuin verrokkuluokittelija, multinomial naive Bayes-luokittelija oppi jotain aineistosta eikä pelkästään arvaa annetun syötteen luokkaa.

Vastaavasti kun verrataan random forest-luokittelijaa tarkkuudella 95.78% ja Bernoulli naive Bayes-luokittelijaa 95.76%, voidaan MCC-arvoa tarkastellessa olettaa, että Bernoulli naive Bayes-luokittelia osaa ilmaista jotain sääntöä hieman paremmin kuin random forest-luokittelija. Ehkä random forest-luokittelija teki jonkin yleistyksen, joka nostaa sen omaa tarkkuutta samalla altistaen sen jollekin toiselle virheluokittelulle, jota Bernoulli naive Bayes-luokittelija ei tee.

Naive Bayes luokittelijoiden MCC-arvoissa toistuu sama teema kuin tarkkuudessa: yksi malleista on huomattavasti tarkempi kuin muut. Kuten jo aiemmin mainit-

sin, complement naive Bayes-luokittelija on MCC-arvoltaan multinomial naive Bayes-luokittelijaan nähden. Tilanne on päinvastainen tarkkuutta tarkastellessa. Bernoulli naive Bayes-luokittelijan hyvä tarkkuus viittaa siihen, että aineisto on Bernoulli-jakautunut, eli binäärisesti jakautunutta.

Tukivektorikone-luokittelijoiden MCC-arvoissa järjestys säilyi samana kuin tarkkuuden perusteella järjestäessä: lineaarinen tukivektorikone-luokittelija kaikkein tarkimpana arvolla 0.8957, ja sigmoid tukivektorikone-luokittelija heikoimpana MCC-arvolla 0.4131.

4.4 Mallien rajoitukset

Koulutetut mallit osaavat luokitella tehtävän oiken tehdyksi ja päinvastoin melko tarkasti tämän aineiston puitteissa. Uuden datan myötä mallin tarkkuudet kuitenkin laskevat. Tämä johtuu mallien tekemästä ylisovittamisesta joita mallit väistämättä tekevät niiden saaman aineiston myötä. Esimerkiksi muutokset opiskelijoissa, opiskelijoiden esitiedoissa, tai opetusmenetelmissä heijastuvat aineistoon siten etteivät koneoppimismallit osaa yleistää havaintoja tutkielmassa ilmoitetulla tavalla. Myös yliopiston oma pedagogiikka vaikuttaa dataan. Vaikka kurssin rakenne säilyisi samana kahden yliopiston välillä, on opetustavoissa väistämättä niin suuria eroja, että mallin tarkkuus ei säilyisi samana.

Ylisovittamisen vaikutusta voi tietysti vähentää kouluttamalla mallit uudestaan silloin kun uutta aineistoa on saatavilla, esimerkiksi ensi vuodelta. Vaikka mallit koulutettaisiin uudestaan, ajan myötä voi koneoppimismallien tarkkuus heiketä. Esimerkiksi verratessa yksittäisien vuosien tehtäväyrityksiä, kullakin koulutetulla mallilla voi olla oletuksia aineistoista, jotka eivät päde muiden vuosien oletuksiin. Aineiston kasvaessa mallit luopuvat näistä yksittäisten vuosien potentiaalisesti luotettavien oletuksien käytöstä, vaihtaen ne yleisempiin ja mahdollisesti vähemmän luotettaviin oletuksiin. Aineiston kasvaessa voi olla järkevämpää tarkastella koneoppimismalleja, jotka suoriutuvat paremmin suurilla datamäärillä. Yksi koneoppimislagoritmien luokka, joka hyötyy juurikin suuresta aineistomäärästä ovat neuroverkot.

Koska tehtäviin lukeutui muiden opiskelijoiden tekemiä tehtäviä, katsoin tärkeäksi suodattaa tehtäväyrityksistä tehtävät, jotka vaikuttavat liian vaikeilta tai mahdottomilta. Tämä tehtiin suodattamalla tehtävät, joissa on alle viisi onnistunutta yritystä. Jos tutkielmassa koulutettuja käytettyjä malleja käytettäisiin näihin tehtäviin kohdistuvien yritysten ennustamiseen, eivät tulokset ole yhtä hyviä verrattuna tutkielmassa ilmoitettuihin tulokseen.

4.5 Mallien mahdollisuudet

Tutkielman saamien tulosten perusteella huomataan, että opiskelijoiden tehtäväyritysten ennustamiseen riittää koneoppimismallit, joiden kouluttamiseen ei tarvita huomattavaa määrää laskentatehoa. Yritysten oikeellisuuden ennustamisen ollessa melko tarkkoja, voitaisiin vastaavia tekniikoita soveltaa jossakin käytännöllisessä kohteessa. Jos käyttökohde vaatii suurempaa tarkkuutta, mallien kouluttamisen pienten laskentakustannusten puitteissa voitaisiin malleja kouluttaa uudelleen melko lyhyinkin väliajoin.

Koska tutkielmassa käytettyjen mallien ennusteet ovat melko tarkkoja, voitaisiin näitä menetelmiä soveltaa esimerkiksi SQL-harjoitteluohjelmiston vihjeen antajaan. Vihjeen antaja tai generoija pyrkii antamaan automaattista apua käyttäjillensä oppimiskokemuksen parantamiseksi [22]. Sovellus jonka tarkoituksena on tarjota vihjeitä käyttäjälleen hyötyisi arviosta siitä, että osaako käyttäjä seuraavan tehtävän. Tutkielman malleja voisi hyödyntää tilanteen päättelemiseen, jossa käyttäjä saattaa tarvita lisää vihjetä tehtävän ratkaisemiseksi.

5. Yhteenveto

Tutkielman tavoitteena oli tarkastella tietokantakurssin opiskelijoiden tehtäväyritysten ennustettavuutta koneoppimismenetelmillä. Aineistosta suodatettiin mahdottomat tehtävät, jonka jälkeen aineiston päivämäärää ilmaisevat sarakkeet jaettiin erilaista tietoa merkitseviin kenttiin. Aineiston perusteella koulutettiin tukivektori-luokittelijoita, naive Bayes-luokittelijoita, random forest-luokittelija sekä LDA-luokittelija.

Tukivektorikoneet koulutettiin ThunderSVM-kirjaston [38] toteutuksella, ja loput koulutettiin scikit-learn-kirjaston [26] toteutuksella. Verrokkiluokittelijaksi valittiin yleisintä luokkaa arvaava ”luokittelija”. Parhaimmat koneoppimismallit MCC:n ja tarkkuuden perusteella olivat random forest-, lineaarinen tukivektorikone- sekä Bernoulli naive Bayes luokittelija. Validoidessa kukin malleista sai yli 95% ennustuksista oikein, ja mallien MCC oli kullakin lähellä 0.9:ää. Koneoppimismallien ryhmien sisällä oli suurta vaihtelua ennustustarkkuuksissa ja MCC:ssä. Tämän todettiin johtuvan kunkin mallin tekemistä oletuksista aineistossa. Esimerkiksi naive Bayes-luokittelijat tekevät oletuksia syötteiden luokan tiheysfunktioista. Tutkielmassa käytettyjen koneoppimismallien suoriutuminen vastasi aiempia tutkimuksia samankaltaisessa kontektissa [16, 7, 8].

Tästä tutkielmasta voisi jatkaa esimerkiksi seuraavilla aiheilla: aineiston ryhmittelyllä, mallien tekemien oletusten purkamisella, neuroverkoilla ja tutkielman mallien sovelluksien tutkiminen. Aineistoa voisi esimerkiksi ryhmitellä siten, että vertailee luentokurssin kävijöitä verkkokurssin käyneisiin opiskelijoihin. Tämä jako voisi esimerkiksi tarjota luentokurssin tehosta, tai kurssimateriaalin puutteita. Tämän tutkielman koulutetut mallit reagoisivat näihin ryhmittelyihin aineistoihin eri tavoin, ja päinvastoin näillä ryhmitellyillä koulutetut mallit saattavat reagoida eri tavoin koko aineistoon.

Tutkielman mallien tulokset antoivat viitteitä aineiston luonteesta. Käsittelemättä jäi kutenkin yleistykset, joita mallit tekivät aineiston perusteella. Tämänkaltainen mallien tulosten tarkastelu voisi tarjota pedagogisia oivalluksia siitä, miten opiskelijat tekevät tehtäviään.

Koska aineiston koko ei antanut perusteita neuroverkkojen kouluttamiseen [13], jäivät ne tämän tutkielman ulkopuolelle. Aineiston kasvaessa on mielenkiintoinen aihe tarkastella miten tämän tutkielman mallit suoriutuvat isommalla datamäärillä, ja

miten ne vertautuisivat neuroverkon ennustustarkkuudelle.

Tutkielman mallit antoi hyviä ennusteita siitä, että ovatko opiskelijoiden tehtäväyritykset oikein. Tämän perusteella voisi olla arvokasta tutkia tällaisten koneoppimismenetelmien soveltamista. Yksi käyttökohde voisi olla SQL:ää opettavat e-oppimisympäristöt, jotka yrittävät parantaa opiskelijoiden käyttökokemusta antamalla automaattisesti vihjeitä opiskelijoille [22]. Arvio siitä että onko opiskelijan yritys oikea voisi tarjota tukea tämänkaltaisille järjestelmille.

Kirjallisuutta

[1] Curriculum guidelines for undergraduate programs in computer science. URL: https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf.

[2] Google colab. https://colab.research.google.com.

[3] Helsingin yliopisto: Viikot ja päivämäärät 2018-2019. https://blogs.helsinki.fi/optime-apu/files/2014/10/Viikot-ja-p%C3%A4iv%C3%A4m%C3%A4%C3%A4r%C3%A4t-2018-2019.pdf.

[4] Helsingin yliopisto: Viikot ja päivämäärät 2018-2019. https://blogs.helsinki.fi/optime-apu/files/2019/01/Viikot-ja-p%C3%A4iv%C3%A4m%C3%A4%C3%A4r%C3%A4t-2019-2020.pdf.

[5] Swebok v3.0 - guide to software engineering body of knowledge. URL: https://www.computer.org/education/bodies-of-knowledge/software-engineering.

[6] Alireza Ahadi, Vahid Behbood, Arto Vihavainen, Julia Prior, and Raymond Lister. Students' syntactic mistakes in writing seven different types of sql queries and its application to predicting students' success. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 401–406, New York, NY, USA, 2016. ACM. URL: http://doi.acm.org/10.1145/2839509.2844640, doi:10.1145/2839509.2844640.

[7] Alireza Ahadi, Raymond Lister, Heikki Haapala, and Arto Vihavainen. Exploring machine learning methods to automatically identify students in need of assistance. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, pages 121–130, New York, NY, USA, 2015. ACM. URL: http://doi.acm.org/10.1145/2787622.2787717, doi:10.1145/2787622.2787717.

[8] Susan Bergin, Aidan Mooney, John Ghent, and Keith Quille. Using machine learning techniques to predict introductory programming performance. *International*

Journal of Computer Science and Software Engineering (IJCSSE), 4(12):323–328, 2015.

[9] Brijesh Kumar Bhardwaj and Saurabh Pal. Data mining: A prediction for performance improvement using classification. *arXiv preprint arXiv:1201.3418*, 2012.

[10] John B. Biggs, Catherine So-kum Tang, and Education Society for Research into Higher. *Teaching for Quality Learning at University : What the Student Does.*, volume 4th ed. McGraw-Hill Education, 2011. URL: http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=405333&site=ehost-live&scope=site.

[11] Peter Brusilovsky, Sergey Sosnovsky, Michael V. Yudelson, Danielle H. Lee, Vladimir Zadorozhny, and Xin Zhou. Learning sql programming with interactive tools: From integration to personalization. *ACM Trans. Comput. Educ.*, 9(4):19:1–19:15, January 2010. URL: http://doi.acm.org/10.1145/1656255.1656257, doi:10.1145/1656255.1656257.

[12] Adam S Carter, Christopher D Hundhausen, and Olusola Adesope. The normalized programming state model: Predicting student performance in computing courses based on programming behavior. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*, pages 141–150. ACM, 2015.

[13] Karo Castro-Wunsch, Alireza Ahadi, and Andrew Petersen. Evaluating neural networks as a method for identifying students in need of assistance. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, pages 111–116, New York, NY, USA, 2017. ACM. URL: http://doi.acm.org/10.1145/3017680.3017792, doi:10.1145/3017680.3017792.

[14] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):6, 2020.

[15] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June 1970. URL: http://doi.acm.org/10.1145/362384.362685, doi:10.1145/362384.362685.

[16] Arto Hellas, Petri Ihantola, Andrew Petersen, Vangel V Ajanovski, Mirela Gutica, Timo Hynninen, Antti Knutas, Juho Leinonen, Chris Messom, and Soohyun Nam Liao. Predicting academic performance: a systematic literature review. In

Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, pages 175–199. ACM, 2018.

[17] Arto Hellas and Matti Luukkainen. Tietokantojen perusteet 2019: Johdanto tietokantoihin. https://tietokantojen-perusteet-19.mooc.fi/osa-1/1-johdanto.

[18] Matthew C Jadud. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the second international workshop on Computing education research*, pages 73–84. ACM, 2006.

[19] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibishirani. *An Introduction to Statistical Learning*. Springer, 2013. URL: http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf.

[20] Fulya Damla Kentli and Yusuf Sahin. An svm approach to predict student performance in manufacturing processes course. *Energy, Edu., Sci. Technol. B*, 3(4):535–544, 2011.

[21] Jarkko Lagus, Krista Longi, Arto Klami, and Arto Hellas. Transfer-learning methods in programming course outcome prediction. *ACM Transactions on Computing Education (TOCE)*, 18(4):19, 2018.

[22] Dejan Lavbič, Tadej Matek, and Aljaž Zrnec. Recommender system for learning sql using hints. *Interactive Learning Environments*, 25(8):1048–1064, 2017.

[23] B.W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442 – 451, 1975. URL: http://www.sciencedirect.com/science/article/pii/0005279575901099, doi:https://doi.org/10.1016/0005-2795(75)90109-9.

[24] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[25] Nirupma Pathak. *Database Management System*. Global Media, 2007.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[27] Julia R Prior. Assesql: an online, browser-based sql skills assessment tool. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 327–327, 2014.

[28] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[29] Shazia Sadiq, Maria Orlowska, Wasim Sadiq, and Joe Lin. Sqlator: an online sql learning workbench. In *ACM SIGCSE Bulletin*, volume 36, pages 223–227. ACM, 2004.

[30] Yutaka Sasaki et al. The truth of the f-measure. 2007, 2007.

[31] John B. Smelcer. User errors in database query composition. *International Journal of Human-Computer Studies*, 42(4):353 – 381, 1995. URL: http://www.sciencedirect.com/science/article/pii/S1071581985710178, doi:https://doi.org/10.1006/ijhc.1995.1017.

[32] S. E. Sorour, J. Luo, K. Goda, and T. Mine. Correlation of grade prediction performance with characteristics of lesson subject. In *2015 IEEE 15th International Conference on Advanced Learning Technologies*, pages 247–249, July 2015. doi:10.1109/ICALT.2015.24.

[33] Toni Taipalus and Piia Perälä. What to expect and what to focus on in sql query teaching. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pages 198–203, New York, NY, USA, 2019. ACM. URL: http://doi.acm.org/10.1145/3287324.3287359, doi:10.1145/3287324.3287359.

[34] Toni Taipalus, Mikko Siponen, and Tero Vartiainen. Errors and complications in sql query formulation. *ACM Transactions on Computing Education (TOCE)*, 18(3):15, 2018.

[35] Christopher Watson and Frederick WB Li. Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 39–44. ACM, 2014.

[36] Christopher Watson, Frederick WB Li, and Jamie L Godwin. Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*, pages 319–323. IEEE, 2013.

[37] Christopher Watson, Frederick WB Li, and Jamie L Godwin. No tests required: comparing traditional and dynamic predictors of programming success. In *SIGCSE*, volume 14, pages 469–474, 2014.

[38] Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. ThunderSVM: A fast SVM library on GPUs and CPUs. *Journal of Machine Learning Research*, 19:797–801, 2018.

[39] Paul Wilton and John W. Colby. *Beginning SQL*. Indianopolis.