TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

VILLE NIITTUNEN

DIAGNOSOINNIN AVUSTAMINEN TEKSTIMUOTOISISTA TERVEYDENHUOLLON AINEISTOISTA

Diplomityö

Tarkastajat: professori Hannu-Matti Järvinen ja professori Heikki Huttunen

Tarkastajat ja aihe hyväksytty 29. maaliskuuta 2018

TIIVISTELMÄ

VILLE NIITTUNEN: Diagnosoinnin avustaminen tekstimuotoisista terveydenhuollon aineistoista
Tampereen teknillinen yliopisto
Diplomityö, 62 sivua, 1 liitesivu
Elokuu 2018
Tietotekniikan DI-tutkinto-ohjelma
Pääaine: Pervasive Systems
Tarkastajat: professori Hannu-Matti Järvinen ja professori Heikki Huttunen

Avainsanat: opinnäytetyö, tietotekniikka, terveydenhuolto, diagnoosi, koneoppiminen, tf-idf, multilabel

Tässä työssä tutkittiin erilaisia menetelmiä diagnoosien ennustamiselle tekstimuotoisesta potilasdatasta koneoppimisen periaatteiden avulla. Terveydenhuollossa potilaasta tuotetaan hoitojaksottain paljon tekstimuotoista dataa ja hänelle määritellään erilaisia diagnooseja. Näitä muistiinpanoja ja diagnooseja voidaan käyttää koneoppimista hyödyntävän diagnoosiluokittelijan opettamisessa. Tekstimuotoisesta datasta tuotettiin sanatiheyksiin pohjautuvia piirteitä ja niitä käytettiin erilaisten koneoppimisalgoritmien avulla tehtävään opetustyöhön. Näiden opetettujen mallien suorituskykyä vertailtiin erilasten tarkkuusarvojen avulla.

Työssä käytettiin anonymisoitua MIMIC-III-potilastietokantaa, johon on mahdollista saada tutkimuskäyttöä varten käyttöoikeus. Hoitojaksojen tekstimuotoisista muistiinpanoista kerättiin niiden sisältämien sanojen perusmuodoista koostettuja TF-IDF-vektoreita ja ICD-9-muotoisia diagnoosikoodeja luokittelijoiden opettamista varten. Työssä osoitetaan, että koneoppimisen avulla tuotettu luokittelija pystyy luokittelemaan potilaiden hoitojaksoja diagnooseittain ICD-9-koodiston ylemmän hierarkiatason mukaan vaihtelevalla menestyksellä. Luokittelun tuloksiin vaikuttaa se kuinka spesifinen tietty luokka on ja onko luokkaa käytetty pääsäntöisesti ensisijaisena vai toissijaisena diagnoosina.

Ratkaisua voitaisiin kehittää tutkimalla syvien neuroverkkojen käyttöä ja tarkemmin takaisinkytkettyjen verkkojen käyttämistä hoitomerkintöjen aikariippuvuuksien hyödyntämiseksi. Toinen vaihtoehto jatkokehitykseen olisi miettiä sääntöpohjaisia ratkaisuja sekä merkintöjen metatietojen parempaa hyödyntämistä.

ABSTRACT

VILLE NIITTUNEN: Assisting diagnosis using medical textual records
Tampere University of Technology
Master's thesis, 62 pages, 1 Appendix page
August 2018
Tietotekniikan DI-tutkinto-ohjelma
Major: Pervasive Systems
Examiners: Professor Hannu-Matti Järvinen and Professor Heikki Huttunen

Keywords: thesis, machine learning, ehr, diagnosis, multilabel, tf-idf

Various methods for predicting the diagnosis from textual patient data were studied using the principles of machine learning. In healthcare, a large amount of textual data and a number of diagnoses are produced for one treatment period. These notes and diagnoses can be used to teach a group of diagnostic classifiers utilizing machine learning. Word frequency based features were generated from the original data to teach several models with different machine learning algorithms. The performance of these models were compared with the help several different precision metrics.

The study in this thesis utilizes an anonymized medical record database, MIMIC-III, which is available for research purposes. TF-IDF feature vectors and diagnostic codes in ICD-9-format are collected from the medical notes for teaching classifiers. The work demonstrates that the classifier produced by machine learning is able to categorically diagnose patients according to the hierarchy of ICD-9 coding with varying successes. The classification results are influenced by how specific a particular category is and whether the class is used primarily as a primary or a secondary diagnosis.

The solution could be developed further by studying the use of deep neural networks and more specifically the use of recurrent neural networks to utilize the time dependencies of the notes and reports in medical records. Another option for further development would be to consider rule-based solutions as well as the better use of the metadata of the records.

ALKUSANAT

Tampereen teknillinen yliopisto tarjosi paljon tiedonnälkäiselle opiskelijalle. Vaikka opinnot eivät aina menneet suunitelmien mukaan, kokemus oli silti erittäin positiivinen ja ihmistä kasvattava. Koneoppiminen ja tekoäly ovat olleet mielenkiinnon kohteena aina opintojen alusta lähtien. Oli mahtavaa päästä tekemään opinnäytetyö tähän aihepiiriin liittyen.

Haluaan kiittää Atostekia ja Juho Leppämäkeä hyvästä ohjauksesta, innostamisesta sekä mahdollisuudesta tehdä tämä työ. Suuri kiitos myös professori Heikki Huttuselle sekä professori Hannu-Matti Järviselle hyvästä palautteesta ja neuvoista. Isoin kiitos kuuluu kuitenkin vaimolleni ja kahdelle ihanalle lapselleni, ilman teitä mistään ei olisi tullut mitään.

Tampereella, 31.7.2018

Ville Niittunen

SISÄLLYSLUETTELO

KUVALUETTELO

LYHENTEET JA MERKINNÄT

ML	Machine Learning (koneoppiminen)
Big Data	jatkuvasti kasvava suuri järjestelemätön tietomäärä
tf	term frequency (sanan esiintymistiheys dokumentissa)
idf	inverse document frequency (sanan käänteinen esiintymistiheys kaikissa dokumenteissa)
tf-idf	tf ja idf arvoja yhdistävä arvo
k-NN	k nearest neighbors (lähimmän naapurin luokittelualgoritmi)
SVM	support vector machine (tukivektorikone)
loss	optimointialgoritmille määritelty häviö
loss function	tietty tapa laskea häviö
SGD	stochastic gradient descent (stokastinen gradienttimenetelmä)
RF	random forest (satunnaismetsä)
accuracy	tarkkuus
precision	täsmällisyys
recall	tunnistuskyky
f1-score	f1-arvo
CPT	Current Procedural Terminology (CPT-koodisto)
DRG	Diagnose Related Groups (diagnooseihin liittyvät ryhmät)
ICD	International Statistical Classification of Diseases and Related Health Problems (tautiluokituskoodisto)
ICD-9	ICD-koodiston yhdeksäs versio
ICU	Intensive Care Unit (teho-osasto)
multiclass	moniluokkaisuus
multilabel	monilukkaisuus tulosluokissa
Python	ohjelmointikieli
NLTK	Natural Language ToolKit
CV	Cross Validation (ristivalidointi)
false positive	väärä positiivinen ennustus
false negative	väärä negatiivinen ennustus
true positive	oikea positiivinen ennustus
true negative	oikea negatiivinen ennustus
RNN	Recurrent Neural Network (takaisinkytketty neuroverkko)
FIR	Finite Impulse Response
IIR	Infinite Impulse Response

1. JOHDANTO

Ihmisen tärkeät teknologiset edistysaskeleet ovat tulleet sykäyksittäin ja ne ovat muuttaneet tapaamme toimia täysin niiden vakiintuessa. Tästä syystä niitä on kutsuttu teknologisiksi vallankumouksiksi. Elämme parhaillaan digitaalisen vallankumouksen aikaa, jonka mahdollisti tiedon siirtäminen digitaaliseen muotoon mekaanisen ja analogisen sijaan. Digitaalinen esitystapa, nollat ja ykköset, voivat sähköisesti kuvata mitä vain tietoa ja siihen perustuu niin tietokoneiden kuin internetinkin toimintaperiaate. Se mahdollistaa valtavien datamäärien tuottamisen ja kanavoimisen ympäri maailmaa silmänräpäyksessä. Tämän ansiosta kuka vain, jolla on esteetön internetyhteys on päässyt käsiksi ennennäkemättömiin tietomääriin. Ihmisen liittyminen laajaan tietoverkkoon on tehostanut teknologista kehitystä sekä yhteistyötä merkittävästi.

Digitaalisen vallankumouksen ansiosta lääketieteen ammattilaisten ei tarvitse muistaa yhtä paljon tietoa kuin analogisella aikakaudella. Digitaalinen tekniikka on vapauttanut lääkäreitä, sairaanhoitajia ja tutkijoita keskittymään enemmän korkeamman tason kognitiivisiin tehtäviin ja potilaiden hoitoon. Digitaalivallankumous on osaltaan auttanut tuomaan tärkeää tietoa lääkäreiden ja muun terveydenhuollon saataville, mutta myös lisännyt tiedon määrää eksponentiaalisesti. Tiedon määrä onkin räjähtänyt käsiin ja sitä on nykyään niin paljon ja niin eritasoista, ettemme tiedä mitä tehdä sillä.

Terveydenhuolto kärsii toisaalta monista ongelmista, kun yhä pienemmillä resursseilla yritetään hoitaa alati kasvavaa potilasmäärää. Monissa maissa painitaan saman ongelman ympärillä; jatkuvasti suureneva osa väestöstä on työiän ylittäneitä. Vanhemmalle väestönosalle on myös ominaista monisairaus, jossa potilas kärsii useammasta kroonisesta sairaudesta samaan aikaan. Nämä potilasryhmät kuluttavat suuren osan terveydenhuollon resursseista [29, s. 3] Monisairautta hoidettaessa on tärkeää, että potilaan kokonaistilanne kartoitetaan ja lääkitys sekä hoidot valitaan niin että ne eivät ole konfliktissa keskenään. Kun väestön ikä kasvaa, julkisen terveydenhuollon kustannukset kasvavat, minkä seurauksena lääkäreiden ajankäyttöä tehostetaan ja heille jää yhä vähemmän aikaa käsitellä yksittäistä potilasta. Asiakkaita ei todennäköisesti kuulla tarpeeksi, jolloin tehdään kiireessä vääriä diagnooseja ja määrätään vääriä lääkkeitä.

Sen lisäksi, että hoitosuhde heikkenee, hoitohenkilökunnalta vaaditaan yhä enemmän dokumentoinnin tuottamista ja tiedon käsittelyä. Pääasiallinen lääkärin työ on nykyään potilasmerkintöjen kirjaamista tai muuta tiedon kirjaamista, joka on jo noin 2/3 ajasta [2, abstrakti]. Tästä johtuen yhä pienempi osa lääkärien työstä käytetään tärkeimpään lääkärien työhön, potilaan tapaamiseen. Tämä osaltaan liittyy kasvavaan tiedon määrään ja sen hallinnan ongelmiin. Monissa elämänaloissa on herätty viime vuosikymmeninä tähän ongelmaan, *big dataan*. Se on ongelman lisäksi mahdollisuus. Mahdollisuus tuottaa uutta

lisätietoa vanhan pohjalta. Sen ympärille on kehittynyt uusia tieteenaloja, joiden tarkoituksena on hyödyntää ja kontrolloida sitä.

Terveydenhuollon tuottamaa tietoa ei tällä hetkellä hyödynnetä riittävän kattavasti ja joissakin tilanteissa sen määrä tuottaa ongelmia, kun terveydenhuollon asiantuntijan täytyy etsiä tietoa valatavista tietokannoista. On myös mahdollista että yhä kiireisemmät lääkärit ja hoitajat eivät ehdi kirjata kaikkea olellista ja kerätty data jää joiltakin osin vajavaiseksi. Sähköinen hoitotietojen systemaattinen ja älykäs hallinta ovat selvästi *big data* -ongelma ja sitä voidaankin valjastaa yllä olevien ongelmien ratkaisemiseen.

Koneoppiminen, *Machine Learning (ML)*, on yksi tieteenala, jolla voidaan käsitellä ja analysoida tätä suurta datamäärää. ML:n tarkoituksena on muodostaa datan perusteella malleja, jotka voivat ennustaa jonkin näkymättömissä olevan tiedon. Esimerkkinä tälläisestä järjestelmästä on ihmisen iän tunnistava malli, joka arvioi iän kasvoista otetun kuvan perusteella. Tässä mallissa käytetään kuvan pikseleiden arvoja ja niistä johdettuja ominaisuuksia päättelemään kuvatun henkilön ikä. Monet ML-algoritmit hyötyvät siitä, että tietoa on paljon, kun niistä muodostetaan ennusteita tekeviä malleja. Näiden avulla käsillä olevasta datasta on saatu paljon enemmän irti kuin aikaisemmin ja aikaisemmin ehkä hieman hyödyttömältä vaikuttanut data on saatu tuottamaan lisäarvoa.

Koneoppiminen voisi automatisoida terveydenhuollossa dokumentoitiin liittyviä töitä ja näin vapauttaa lääkärit tekemään heidän ydinosaamiseensa liittyviä tehtäviä. Koneoppiminen voisi mahdollistaa täysin uusia analysointimahdollisuudet ja koneoppiva järjestelmä voisi tehdä myös suoraan lääkäreitä tukevaa työtä, diagnoosien päättelyä ja sairausepisodien ennustamista. ML ei voi korvata täysin lääkäreitä, sillä heidän työtään ei voida täysin antaa koneoppimisen vastuulle. Se toimii sen sijaan toisena silmäparina lääkärille ja lisää tehtävien diagnoosien varmuuttaa ja vähentää vääränlaisen lääkityksen määrämistä potilaalle. Tämä taas johtaa harvempiin potilaskäynteihin ja lisääntyneeseen asiakastyytyväisyyteen. Diagnooseja ennustava ML-mallin ongelmakohtana on standardoimaton tiedon esitysmuoto varsinkin vapaata tekstiä sisältävissä hoitomuistiinpanoissa. Niihin usein kuitenkin sisältyy paljon arvokasta tietoa, josta terveydenhuollon ammatilaisten on vaikea päätellä suoraan kaikkea, mitä siitä voisi analysoimalla saada.

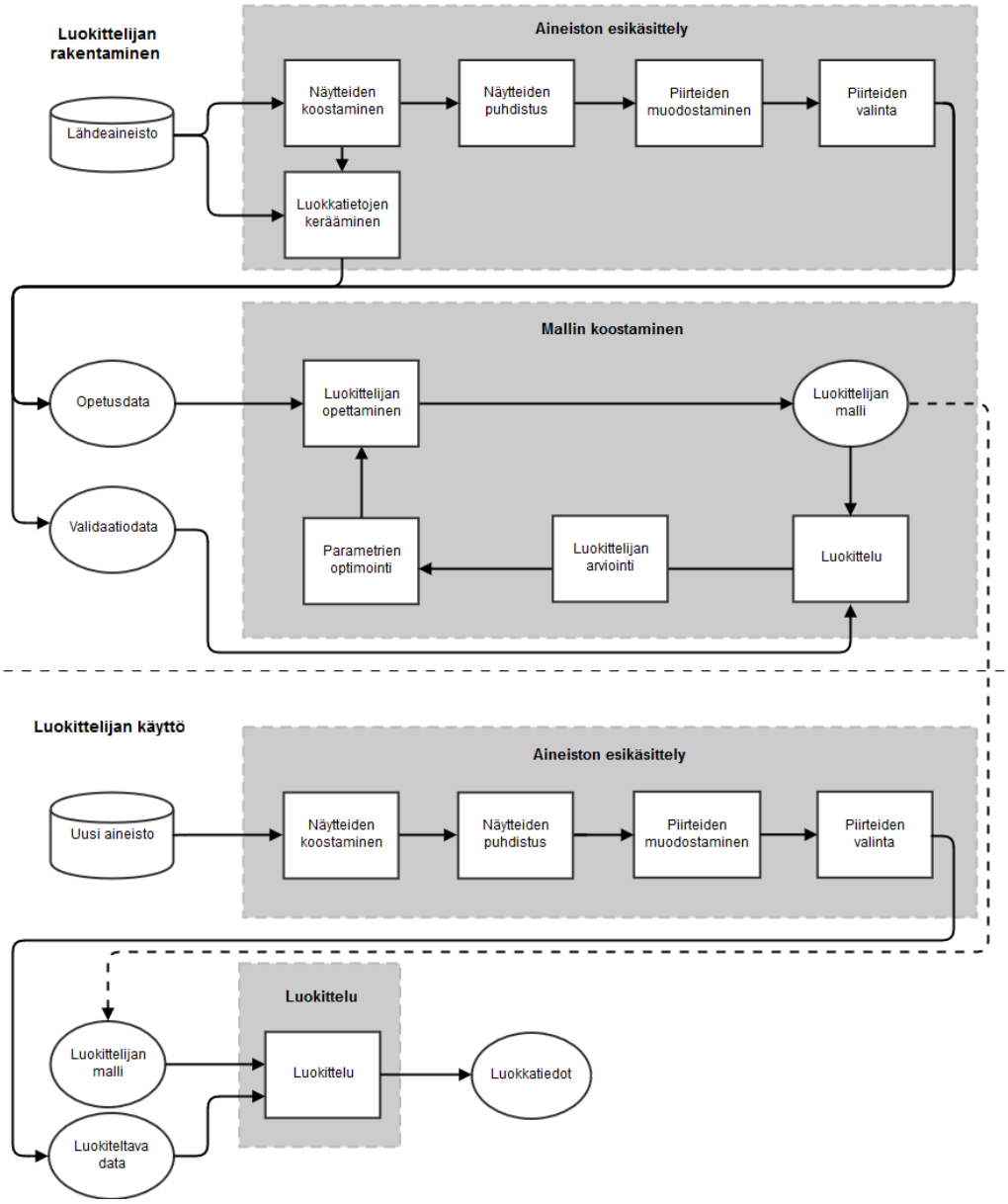
Tässä työssä käsitellään koneoppimisen ratkaisuja potilaiden diagnoosien selvityksessä tekstimuotoisesta potilasdatasta. Tavoitteena on vapauttaa lääkäreiden aikaa ja lisäämällä toinen näkökulma tai mielipide jokaiseen lääkärin tekemään diagnoosiin. Potilasdatan sanoista muodostettuja esiintymistiheyksiä analysoimalla voidaan saada jokseenkin luotettavia numeerisia arvoja, joita voidaan hyödyntää koneoppimisessa. Työssä arvioidaan parhaita koneoppimisen luokittelijoita tähän tehtävään.

Seuraavissa kappaleissa käsitellään mahdollisia koneoppimisen tarjoamia ratkaisuja diagnoosien luokitteluongelmaan liittyen. Koneoppimiseen liittyvää teoriaa käsitellään käytettyjen luokittelijoiden ja muiden tekniikoiden suhteen kappaleessa 2. Tärkeänä osana työssä oli käytetty data-aineisto, josta kerrotaan tarkemmin kappaleessa 3. Tämän jälkeen käsitellään tehtyä koneoppimistutkimuksen totetusta ja siitä saatuja tuloksia luokittelijoiden vertailussa kappaleessa 4. Kappaleessa 5 kerrotaan mahdollisista jatkokehitysideoista ja tavoista, miten voitaisiin kehittää luokittelukykyisempi malli. Lopuksi tiivistetään työn eri vaiheet ja sen tulokset kokonaisuudessaan kappaleeseen 6.

2. KONEOPPIMINEN

Tässä kappaleessa kuvataan toteutetun tekstiluokituksen kannalta tärkeimmät teoreettiset koneoppimisen periaatteet ja menetelmät. Koneoppiminen perustuu matemaattisen mallin tai sääntöjoukon muodostamiselle datan perusteella. Tarkoituksena on muodostaa oppiva järjestelmä, joka pystyy suoritumaan yhä parempaan tulokseen jossakin ongelmassa tai tehtävässä. Koneoppimista käytetään yleensä tuomaan esille tai arvioimaan jotain näkymättömissä olevaa tietoa olemassa olevan datan pohjalta ja löytämään uusia riippuvuuksia eri asioiden välillä. Tämä totetutetaan tietynlaisten oppivien algoritmien avulla, jotka koostavat datasta lainalaisuuksia tai muodostavat mallin datan rakenteesta. Valmiilla mallilla tehtävä uuden datajoukon analysointi vie vähemmän suoritusaikaa kuin kyseisen mallin koostaminen.

Koneoppisen tärkeimpiä sovelluskohteita ovat luokittelu, regressio ja klusterointi. Regressiossa tavoitteena on datan lainalaisuuksien selvittäminen ja mahdollisesti matemaattisen mallin koostaminen. Klusteroinnissa datasta pyritään rajamaan eriytyviä joukkoja, niin että joukkojen väliset eroavaisuudet olisivat mahdollisimman suuret, kun taas joukkojen sisäinen eroavaisuus mahdollisimman pientä. Tätä menettelyä käytetään, kun pyritää löytämään datasta täysin uusia eriteltäviä joukkoja tai verifioimaan aikaisempia käsityksiä kyseisestä datasta. Tässä työssä keskitytään luokitteluun, joten perehdytään siihen enemmän. Koneoppimisessa luokittelulla tarkoitetaan lähdeaineiston näytteiden jakamista ennalta tiedettyihin luokkiin näytteiden ominaisuuksien perusteella. Tätä luokittelua tekee oppivalla algoritmilla lähdeaineiston avulla kehitetty malli. Koneoppimisessa luokittelijan rakentaminen voidaan jakaa kahteen eri päävaiheeseen; aineiston esikäsittelyyn ja luokittelijan mallin koostamiseen. Luokittelijan rakentaminen tuottaa kaksi tärkeää rakennetta, joita hyödynnetään varsinaista luokittelua tehdessä. Rakentamisen yhteydessä kehitetyn luokittelijan mallin lisäksi aineiston esikäsittely uudelleenkäytetään uusia näytteitä luokiteltaessa. Kuva 1 kuvaa luokittelijan rakentamisen ja käytön toimintaperiaatetta.



Kuva 1. *Luokittelijan rakentamisprosessi*

Aineiston esikäsittely riippuu täysin lähdeaineiston datan rakenteesta ja käsiteltävän ongelman reunaehdoista. Se sisältää kuitenkin yleensä erilliset työvaiheet näytteiden muodostamiselle, niiden luokittelulle sekä esikäsittelylle, piirteiden muodostamiselle ja luokittelussa käytettävien piirteiden valinnalle. Piirteillä tarkoitetaan koneoppimisen yhteydessä lähdeaineistosta koottuja tai kehitettyjä ominaisuusjoukkoja, joita luokittelija käyttää luokittelutuloksen tuottamiseen. Reaalimaailman luokitteluongelmissa näytteiden koostamisen ja luokittelun tarve aineiston pohjalta vaihtelee suuresti, mutta usein käytössä oleva aineisto ei sisällä näitä rakenteita suoraan vaan ne pitää generoida tietokantatyökaluilla tai skripteillä. Usein näytteiden luokkatiedot täytyy myös tuottaa käsin, mutta ne voivat myös löytyä suoraan tai välillisesti lähdeaineistosta. Datan näytteistäminen on kuitenkin rakenteltaan hyvin vaihteleva prosessi, joten sitä käsitellään tarkemmin työn toteutuksen läpikäynnin yhteydessä kappaleessa 4. Kun näytteet on koostettu, ne esikäsitellään. Näytteiden esikäsittely, on niiden yhdenmukaistamista, merkityksettömän tiedon poistamista sekä äärimmäisyyksien tasoittamista. Yksi tekstimuotoiselle datalle tehtävä esikäsittely on kaikkien luokittelun kannalta merkityksettömien kirjoitusmerkkien, kuten rivinvaihtojen, poistaminen. Näytteiden yhdenmukaistamisen jälkeen niistä voidaan kehittää luokittelussa käytettävät piirteet.

Piirteiden tulisi olla sellaisia, että ne kuvaavat luokittelun kannalta tärkeän tiedon tiiviissä muodossa. Piirteitä voidaan kehittää monilla eri tavoilla. Niitä voidaan poimia lähdeaineistosta suoraan tai niitä voidaan laskea aineiston datan perusteella. Tekstipohjaiselle datalle se yleensä tarkoittaa sanatiheyksien tai -lukumäärien laskemista. Tarkoituksena on muodostaa jokaisesta dokumentissa esiintyvästä sanasta piirre, jonka arvona on kyseisen sanan lukumäärä tai esiintymistiheys yksittäisessä dokumentissa. Kun piirteet on muodostettu, pitää niistä valita ne, jotka kuvaavat luokkajakoja parhaiten ja tekevät luokittelijan mallista tarkimman. Piirteiden valintaa tehdään, koska aineisto saattaa sisältää luokittelua sekoittavia piirteitä ja ne halutaan karsia pois. Jotkin luokittelija-algoritmit, kuten päätöspuut (*eng: decision tree*), tekevät piirteiden valintaa osana mallinmuodostamisprosessiaan, mutta monille luokittelualgoritmeille se täytyy sisällyttää erillisenä vaiheenaan luokittelijan rakentamisessa. Piirteiden valinnan jälkeen aineiston esikäsittely on valmis ja näytteet jaetaan opetus-, validointi- ja testidataksi. Tämän jaon avulla opetusdatalla kehiteltäviä luokittelijakandidaatteja voidaan arvioida kehityksen aikana validointidatalla ja lopuksi testidatalla.

Luokittelijan mallin koostaminen sisältää luokittelijan opetusvaiheen jollakin koneoppimiseen sopivalla algortimilla sekä opetetun luokittelijan arvioinnin ja optimoinnin. Kuvan 1 *"Mallin koostaminen"*-laatikko kuvaa tämän vaiheen rakenteen. Tämä on iteratiivinen prosessi, jonka tarkoituksena on tehdä luokittelijasta mahdollisimman tarkka. Prosessi lopetetaan, kun koostetun mallin tarkkuuten ollaan tyytyväisiä. Tämän jälkeen malli on valmis käytettäväksi varsinaisessa luokittelukäytössä. Luokitteluun sopivia oppivia algoritmeja ovat esimerkiksi aiemmin mainittujen päätöspuiden lisäksi, tukivektorikoneet, lähimmän naapurin algoritmit ja neuroverkot. Kun mallista kehitetään erilaisilla parametriyhdistelmillä alustettuja ja erilaisilla piirrekombinaatioilla harjoitettuja luokittelijoita,

voidaan niiden suorituskykyä vertailla keskenään arviointitulosten perusteella ja säätää siten parametrit ja käytettävät piirteet optimaalisiksi. Myös eri oppimisalgoritmien tarkkuutta voidaan vertailla tällä tavalla ja valita niistä parhaiten suoriutuva algoritmi luokittelijalle. Esimerkkinä tekstimuotoista dataa käyttävästä koneoppivasta luokittelijasta on vuonna 2014 Tampereen teknillisessä yliopistossa tehty diplomityö [14]. Viitatussa työssä käsiteltiin Twitter-viestejä ja pyrittiin rakentamaan luokittelija koneoppimisen avulla, joka pystyisi arvioimaan viestien positiivisuutta ja negatiivisuutta. Seuraavissa aliluvuissa käsitellään tarkemmin tekstipohjaista dataa luokittelevan koneoppivan luokittelijan rakentamisen vaiheita.

2.1 Koneoppimisen lähestymistavat

Koneoppimisen avulla tehtävään luokitteluun on monia erilaisia lähestymistapoja. Yksi jako eilaisten metodien välille on se, käytetäänkö opetusainoston luokittelutietoja hyväksi mallia harjoitettaessa. Jos luokkatietoja käytetään, kutsutaan lähestymistapaa ohjatuksi oppimiseksi ja ohjaamattomaksi oppimiseksi, jos luokkatietoja ei hyödynnetä. Kun halutaan jakaa lähdeaineiston näytteet tässä työssä määritellyllä tavalla ennaltamäärättyihin luokkiin käytetään valvottua oppimista. Tällä tavalla muodostetaan luokittelija, jonka tiedetään luokittelevan uudet näytteet haluttujen luokkien kesken [34, s. 4–5].

Ohjaamattomassa oppimisessa voidaan myös tehdä luokittelua, mutta tässä yhteydessä sitä kutsutaan klusteroinniksi. Klusterointi eroaa luokittelusta siinä, että malli muodostaa itsenäisesti lopulliset luokat syötedatan perusteella. Klusterointimenetelmä pyrkii maksimoimaan kehitettyjen luokkajakojen väliset etäisyydet ja minimoimaan luokan sisäiset eroavaisuudet. Ohjaamatonta oppimista hyödynnetään varsinkin, kun halutaan löytää uusia lainalaisuuksia datasta. Sitä voidaan myös käyttää valvotun oppimisen luokittelutehtävissä lähdeaineiston esikäsittelyn yhteydessä. Ohjaamattoman oppimisen avulla voidaan havaita ne piirteet, joissa on eniten informaatiota ja valita niistä luokittelussa käytettävät piirteet [34, s. 2–3].

Vahvistettu oppiminen on myös eräs tapa lähestyä koneoppimista. Sitä hyödynnettäessä dataa ei jaeta suoraan opetus-, validointi- ja testausjoukkoihin, vaan jokaista datanäytettä käytetään ensin olemassa olevan mallin arviointiin, jonka jälkeen sitä käytetään mallin parantamiseen. Arvioinnista saatava virhetermi määrittelee kuinka paljon ja miten mallia muutetaan, jolloin tapahtuu varsinainen oppiminen. Vahvistettua oppimista käytetään paljon, ympäristöissä, joissa opeteltavan datan luonne saattaa muuttua ja siihen pyritään sopeutumaan [20, s. 7–8] Tässä työssä keskitytään ohjattuun oppimiseen, sillä se sopii rajattuun ongelmaan paremmin kuin kaksi jälkimmäistä tapaa.

2.2 Näytteiden esikäsittely

Näytteiden esikäsittelyn (*data preparation*) tarkoituksena on muuttaa aineistojoukkoja siten, että niiden oleellinen tietosisältö olisi mahdollisimman selvästi eroteltavissa piirteiden muodostamista varten. Kattavan esikäsittelyn seurauksena lopulliselle luokittelijalle

syötetään mahdollisimman tarkkaa ja merkityksellistä dataa, minkä seurauksena luokittelijan malli tuottaa parempia tuloksia ja luokittelu on nopeampaa. Ohjenuorana voidaan pitää, että mitä laadukkaampaa tietoa luokittelija saa, sitä laadukkaampia ovat sen tuottamat luokkatiedot.

Tekstidatan esikäsittely on yleensä hyvin suoraviivaista. Ensin tulisi miettiä mikä tekstissä olevasta tiedosta on merkityksellistä. Useinkaan ei ole merkitystä luokittelun kannalta onko tekstissä suuraakkosia tai mitä välimerkkejä dokumenteissa esiintyy. Välimerkeillä tarkoitetaan sanoista erillisiä merkkejä, joilla viestitään kirjoitetussa kielessä asioiden yhteyksiä. Näitä ovat esimerkiksi pilkku, ajatusviiva ja piste. Kirjoitetun kielen lisätiedon käyttäminen koneoppimisen yhteydessä riippuu kontekstista, joskus on myös syytä analysoida esimerkiksi tiettyjen välimerkkien käyttöä. Yleensä, kun varsinainen ongelmalle tärkeä tieto löytyy tekstistä, tulisi yllä olevan kaltainen luokittelun kannalta merkityksetön tieto poistaa. Kun keskitytään sanojen sisältämään informaatioon, tieto kirjasinkoosta ja välimerkit ovat merkityksettömiä. Käsiteltävän aineiston kieli vaikuttaa myös datan esikäsittelyyn. Työssä käytettiin tietokantaa, johon tekstidata oli kirjattu englannin kielellä, joten käsitellään tekstin esikäsittelyä vielä sen suhteen. Englannin kielessä on paljon lyhyitä apusanoja, jotka välimerkkien lailla, lisäävät tietoa tekstimuotoisen datan käsitteiden välisistä riippuvuuksista. Nämä sanat ovat peräisin kolmesta eri sanaluokasta; artikkeleista, prepositioista ja pronomineista. Ne ovat luokittelun kannalta tarpeettomia ja ne voidaan poistaa aineiston käsittelystä.

Datan normalisointi on yksi esikäsittelyvaiheen toimenpiteistä. Se perustuu tiedon yhdenmukaistamiseen ja datassa esiintyvien virheiden korjaamiseen, jolloin piirteiden muodostaminen onnistuisi mahdollisimman kattavasti ja tarkasti. Tekstimuotoiselle datalle normalisointi tarkoittaa kirjoitusvirheiden korjaamista ja sanojen kirjoitusmuotojen yhdenmukaistamista. Sanojen yhdenmukaistaminen voi olla eri murteiden kirjoitusasujen yhdenmukaistamista tai jopa sanojen muuttamista perusmuotoonsa eli stemmausta (*stemming*). Erityisen tärkeää sanojen stemmaus on esimerkiksi suomen kielessä, jossa sanoille löytyy lukemattomia eri taivutusmuotoja. Jotta eri taivutusmuodoista ei muodostettaisi erillisiä piirteitä, ne tulisi muuttaa ensin perusmuotoon ja generoida näin saaduista perusmuodoista lopulliset piirteet. Toinen tapa karsia tarpeettomia piirteitä on yhdenmukaistaa eri sanat, jotka tarkoittavat samaa asiaa. Esimerkiksi brittienglannissa ja amerikanenglannissa on sanoja, joilla tarkoitetaan jotain samaa käsitettä, mutta ne kirjoitetaan eri tavalla. Joskus tätä tapaa on vaikea käyttää ja se voi korjata sanoja virheellisesti, kuten esimerkiksi työn kontekstissa hoitomuistiinpanoja käsitellessä.

Työssä käytetyt hoitomuistiinpanot ovat lääkärien, hoitajien ja sosiaalityöntekijöiden kirjaamia muistiinpanoja potilastapaamisista, tehdyistä toimenpiteistä, kokeista sekä yleiseen hoitotyöhön liittyvistä huomioista. Niissä käytetään paljon lyhenteitä ja niissä esiintyy myös huomattava määrä kirjoitusvirheitä. Kirjoitusvirheiden esiintyminen on täysin ymmärrettävää sillä hoitohenkilökunta on kiireistä, joten kirjoitusvirheitä väistämättä sattuu paljon. Teksteissä esiintyvät lyhenteet ovat myös seurausta hoitohenkilökunnan kiireisestä työstä, niiden kirjoittaminen säästää varmasti paljon aikaa. Eräs tekstin normalisoin-

timetodi, josta saattaa olla paljon hyötyä on näiden kirjoitusvirheiden korjaaminen. Kun sanojen virheet korjataan, ei luoda virheellisiä piirteitä myöhemmin käsiteltäviä algoritmeja varten. Virheiden korjaaminen voi tulla kuitenkin hyvin vaikeaksi yllä mainittujen standardoimattomien lyhenteiden takia. Virheiden korjausta tekevälle algortimille voi olla hyvin vaikeaa erottaa oikeaa lyhennettä väärin kirjoitetusta sanasta.

Dataan saattaa myös liittyä muuta metatietoa, joita ei tarvita sanojen ominaisuuksiin keskittyvässä luokittelijassa, niinpä ne tulisi poistaa myös datasta. Kun aineisto on käsitelty tavoitellulle tasolle asti, siitä voidaan alkaa koostamaan luokittelussa käytettäviä piirteitä.

2.3 Piirteiden muodostaminen

Koneoppimisessa piirteiden muodostaminen (*feature extraction*) aloitetaan käsittelemällä edellisessä vaiheessa, datan esikäsittelyssä, normalisoitua dataa ja muodostamalla siitä johdettuja arvoja eli piirteitä. Piirteet kootaan, jokaista näytettä vastaavaksi piirrevektoriksi, jolloin se on helposti luokittelualgoritmin käytettävissä. Piirteiden muodostamisessa mielivaltaisesta datasta lasketaan joukko numeerisia ominaisuuksia jokaiselle aineiston näytteelle. Koneoppiville algoritmeille ei voida syöttää suoraan kokonaisia ääninäytteitä tai kuvatiedostoja, vaan alkuperäisestä datasta täytyy kehittää sellaisia arvoja, joita voidaan helposti käyttää laskennallisesti. Piirteiden muodostamisessa työstettävän ongelman kannalta tärkeä tieto pyritään esittämään paljon tiiviimmin ja korostetummin kuin mitä se oli alkuperäisessä data-aineistossa.

Koneoppimisen luokitteluongelma, jossa on suuri määrä muuttujia, vaatii yleensä suuren määrän muistia ja laskentatehoa. Se myös voi aiheuttaa myöhemmin mallin opetusvaiheessa harjoiteltavan datan ylioppimista, jolloin siitä johdettu koneoppimisen malli luokittelee heikosti uutta näytedataa. Piirteiden muodostamisella taistellaan osittain tätä ongelmaa vastaan, sillä se tiivistää datan sisältämää informaatiota ja poistaa siitä merkityksetöntä toisteisuutta. Kun näytteiden informaatio on esitetty mahdollisimman pelkistetyssä muodossa se helpottaa myöhempiä oppimis- ja luokitteluvaiheita ja joissakin tapauksissa johtaa myös parempaan inhimilliseen tulkintaan kyseisen aineiston riippuvuussuhteista. Piirteiden muodostaminen liittyy myös vahvasti seuraavaan vaiheeseen, piirteiden valintaan (*feature selection*), jossa ominaisuusjoukkoa voidaan karsia vielä edelleen pienemmäksi, jos sen epäillään vielä sisältävän toisteisuutta tai merkityksetöntä informaatiota.

Seuraavassa aliluvussa käsitellään kuitenkin ensin tarkemmin erilaisia piirteiden muodostustapoja, kun käsiteltävä data on tekstimuotoista.

2.3.1 Sanojen lukumäärä

Sanapussimalli (*Bag-of-words*) on yksinkertainen esitys, jota käytetään luonnollisen kielen analysoinnissa ja tiedonhaussa. Tässä mallissa käsiteltävien näytteiden, kuten lauseiden tai asiakirjojen, sanoista koostetaan näytekohtaiset sanastot, joissa jokainen sana saa

numeroarvon, jolla se voidaan identifioida. Tämän jälkeen sanastoista lasketaan niissä esiintyvien sanojen esiintymislukumäärät. Lopulliset piirrevektorit sisältävät tällöin kussakin näytteessä esiintyvien sanojen identifikaatioarvot ja niiden esiintymismäärät. Tämä malli jättää huomiotta kieliopin ja sanajärjestyksen ja käsittelee vain yksittäisiä sanoja.

Esimerkiksi teksteistä

“Koira juoksee ulos ovesta. Ovesta kuuluu narahdus.”

ja

“Kissa juoksee ovesta sisään. Sisään tullessaan kissa pysähtyy.”

koottaisiin seuraavanlaiset sanapussimallin sanastot:

["koira", "juoksee", "ulos", "ovesta", "ovesta", "kuuluu", "narahdus"],

["kissa", "juoksee", "ovesta", "sisään", "sisään", "tullessaan", "kissa", "pysähtyy"]

ja piirrevektorit:

["koira": 1, "juoksee": 1, "ulos": 1, "ovesta": 2, "kuuluu": 1, "narahdus": 1 "kissa": 0, "sisään": 0, "tullessaan": 0, "pysähtyy": 0],

["koira": 0, "juoksee": 1, "ulos": 0, "ovesta": 1, "kuuluu": 0, "narahdus": 0 "kissa": 2, "sisään": 2, "tullessaan": 1, "pysähtyy": 1]

Yksinkertaisuutensa vuoksi sanapussi-mallia käytetään yleisesti tekstidokumenttien luokittelutehtävissä, joissa kunkin sanan esiintymismäärää käytetään piirteiden tuottamiselle luokittelijaa varten. Käytännössä sanapussi-mallia käytetään lähinnä piirteiden tuottamisen esivaiheena ja sen tuottamista esiintymislukumääristä lasketaan tämän jälkeen muita edistyneempiä ominaisuuksia kuvaamaan näytetekstiä. Yleisimpiä ominaispiirteitä tai piirteitä, jotka lasketaan sanapussi-mallista, on termien taajuus eli kuinka usein termi esiintyy tekstissä.

Sanapussi-malli on järjestämätön esitys jonkin dokumentin sanasisällöstä. Sanapussi-esitys ei paljasta sitä, että ensimmäisen lauseen tekijää seuraa aina näissä dokumenteissa verbi "juoksee". Vaihtoehtoisesti n-gram-mallilla voidaan muodostaa piirteitä, joista ilmenee sanojen välisiä riippuvuuksia kokoamalla vierekkäisten sanojen joukkoja. N-gram-malli koostetaan kokoamalla dokumenteista kaikki niissä esiintyvät n-pituiset merkki- tai sanajonot. Soveltaen ylläolevan esimerkin ensimmäisiä lauseita, 2-gram, eli bigram-malli jäsentää tekstin seuraaviin yksiköihin ja tallentaa kunkin yksikön termitaajuuden kuten aiemmin.

["koira juoksee": 1, "juoksee ulos": 1, "ulos ovesta": 1, "kissa juoksee": 0, "juoksee ovesta": 0, "ovesta sisään": 0],

["koira juoksee": 0, "juoksee ulos": 0, "ulos ovesta": 0, "kissa juoksee": 1, "juoksee ovesta": 1, "ovesta sisään": 1]

Yleistä n-gram-mallien käytössä on määritellä arvoväli, joiden mukaan muodostetaan n-gram-jäsennyksiä. Määritellessä väli (x, y) n-gram-malli muodostaa kaikki n-pituiset sanajonot niin, että n on jotain x:n ja y:n välillä. Esimerkiksi välin (1, 2) n-gram-malli muodostaisi kaikki yhden ja kahden sanan mittaiset sanajonot ja olisi yllä olevalle esimerkille seuraavanlainen:

["koira": 1, "juoksee": 1, "ulos": 1, "ovesta": 1, "kissa": 0, "sisään": 0, "koira juoksee": 1, "juoksee ulos": 1, "ulos ovesta": 1, "kissa juoksee": 0, "juoksee ovesta": 0, "ovesta sisään": 0],

["koira": 0, "juoksee": 1, "ulos": 0, "ovesta": 1, "kissa": 1, "sisään": 1, "koira juoksee": 0, "juoksee ulos": 0, "ulos ovesta": 0, "kissa juoksee": 1, "juoksee ovesta": 1, "ovesta sisään": 1]

Tällä tavalla tuotettujen piirteiden määrä kasvaa nopeasti suunnattoman suureksi, jolloin täytyy valita hyvin tarkasti sopiva raja-arvo, kuinka paljon n-gram-yksiköitä kannattaa tuottaa, jotta luokittelijan tarkkuus olisi optimaalinen. Järkevä maksimiarvo on yleensä $N = 4..5$.

2.3.2 TF-IDF

Tiedonhaussa tf-idf tai TFIDF (*term frequency - inverse document frequency*) on numeerinen tilasto, jonka tarkoituksena on kuvata, kuinka tärkeä jokin aineiston sana on. Tf-idf-arvo lasketaan jokaiselle aineistoista valitulle sanalle ja se koostuu kahdesta osasta.

Ensimmäinen osa on termin taajuus tai tf (*term frequency*), joka kuvaa tietyn sanan esiintymistaajuutta. Se lasketaan jakamalla sanan esiintymismäärä yksittäisessä näytteessä saman asiakirjanäytteen kakkien sanojen kokonaislukumäärällä. Koska käsiteltävät näytteet voivat olla pituudeltaan erikokoisia, on mahdollista, että sana esiintyisi paljon useammin pitkissä näytteissä kuin lyhyissä. Tämän tasapainottamiseksi sanan esiintymismäärä jaetaan asiakirjan kaikkien sanojen lukumäärällä. Tf voidaan laskea seuraavalla tavalla.

$$TF(s,a) = \frac{\text{sanan s esiintymismäärä asiakirjassa a}}{\text{asiakirjan a sisältämien sanojen kokonaismäärä}}$$

jossa s on sana ja a on asiakirja, joille TF lasketaan.

Tf:n laskennassa kaikkia sanoja pidetään yhtä tärkeinä. Tietyt sanat eivät kuitenkaan sisällä paljon informaatiota, kuten "on", "joka"ja "sana". Monet sanat saattavat esiintyä usein dokumenteissa, mutta eivät ole kovinkaan tärkeitä. Siksi näiden sanojen merkitystä pitäisi painottaa vähemmän kuin harvinaisempia sanoja. Idf:llä eli käänteisellä asiakirjan taajuudella (*inverse document frequency*) voidaan tehdä tämänkaltaista painotusta. Se lasketaan

ottamalla logaritmi jakolaskusta, jossa näytteiden lukumäärä jaetaan niiden asiakirjojen määrällä, joissa käsiteltävä sana esiintyy. Idf lasketaan allaolevan kaavan mukaisesti.

$$IDF(s,A) = \log \frac{\text{asiakirjojen A kokonaismäärä}}{\text{asiakirjojen A määrä, joissa sana s esiintyy}}$$

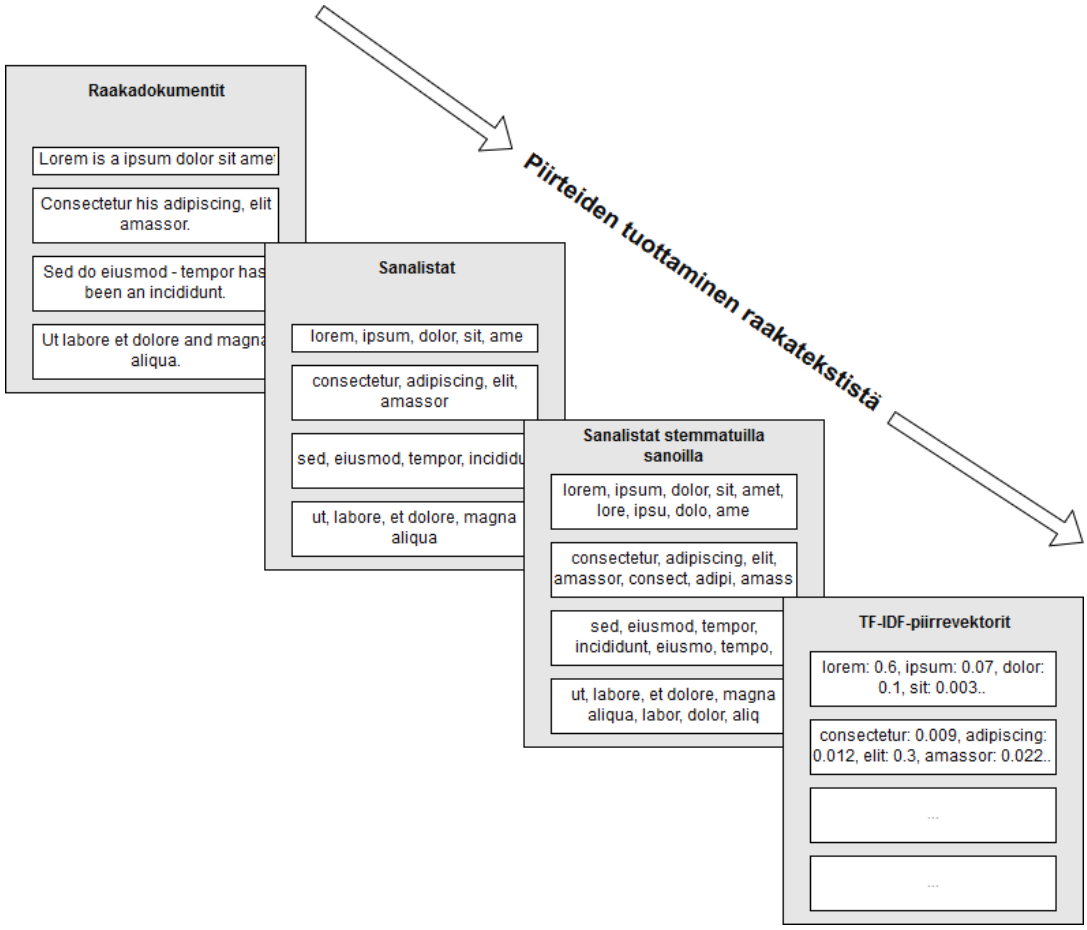
jossa s on sana ja A on kaikkien asiakirjojen joukko, joille IDF lasketaan.

Lopullinen tf-idf:n arvo lasketaan näiden osien kertolaskuna.

$$TF\text{-}IDF(s,a,A) = TF(s,a) \cdot IDF(s,A)$$

Tf-idf-arvo kasvaa suhteessa siihen, kuinka monta kertaa sana esiintyy yksittäisessä asiakirjassa, ja pienenee, kun sanan esiintyvyys kaikissa dokumenteissa kasvaa. Nykyään tf-idf on yksi suosituimmista sanojen merkitysten painotustavoista. Hakukoneet käyttävät usein tf-idf-painotusjärjestelmää keskeisenä työkaluna asiakirja-aineiston pisteytykseen ja tätä kautta sijoituksen muodostamiseen hakutulokselle.

Käsitellessä kieliä kuten englanti, jossa ei ole paljon taivutusmuotoja, tf-idf korostaa usein kuvaavampia piirteitä kuin yksinkertainen sanapussimalli. Esimerkiksi teksteistä "patient has no headaches", "patient has an anemia"ja "patient has smoked heavily"pienenisivät arvot termeillä "patient"ja "has", koska ne esiintyvät jokaisessa dokumentissa. Tärkeät termit kuten "headaches", "anemia"ja "smoked"saavat taas huomattavasti isommat arvot sillä ne esiintyvät vain yksitäisissä dokumenteissa.



Kuva 2. *Tf-idf-piirteiden tuottaminen*

Kielissä, joissa sanoilla voi olla paljon erilaisia taivutusmuotoja aikaisemmin mainitut piirteiden generointitavat, tf, idf ja tf-idf, eivät toimi tehokkaasti, koska eri taivutusmuodot mielletään eri sanoina ja ne saavat siten erilliset arvot näille algoritmeille. Tämä voi antaa virheellisiä painotuksia sanoille ja haitata siten myöhempää luokittelua. Aikaisemmin mainittu sanojen stemmaus eli niiden muuttaminen perusmuotoonsa voi auttaa tässä ongelmassa. Stemmaus perustuu sanan taivutusmuodon pelkistämiseen. Esimerkiksi sanat "koiran", "koiraksi", "koirien"muunnettaisiin kaikki sanan perusmuotoon "koira". Kun sanat on ensin pelkistetty ne ovat yhteneväisiä keskenään ja niistä pystytään muodostamaan luotettavampia tf-idf-arvoja. Kuvasta 2 voidaan nähdä tekstille tehtävä piirteidentuottamisprosessi tf-idf-arvojen saamiseksi.

2.4 Piirteiden valinta

Varsinaisessa luokittelutehtävässä hyödynnettävän ominaisuusjoukon määrittämistä alkuperäisten ominaisuuksien pohjalta kutsutaan piirteiden valinnaksi. Luokitteluun vaikuttavan tiedon odotetaan sisältyvän valittuun alijoukkoon, jotta haluttu luokittelutehtävä voidaan suorittaa käyttämällä tätä pienennettyä edustusta koko alkuperäisen joukon sijaan. Alkuperäisen syötedatan suuri piirrejoukko voi vähentää monien mallien tarkkuutta, sillä

se todennäköisesti sisältää merkityksetöntä informaatiota, joka vaikuttaa häiriönä lopullisessa luokittelijassa.

Piirteiden valinta on yleensä viimeinen vaihe luokiteltavan datan esikäsittelyssä eikä sitä tule sekoittaa piirteiden muodostamiseen. Piirteiden muodostaminen luo uusia ominaisuuksia alkuperäisen tiedon pohjalta, kun taas piirteiden valinta karsii käytössä olevista ominaisuuksista osan pois. Piirteiden valintamenetelmiä käytetään varsinkin ongelmissa, joissa on monia piirteitä ja suhteellisen vähän näytteitä. Sillä voidaan nähdä olevan kolme suurta hyötyä koneoppimisen kannalta. Sen avulla pystytään tunnistamaan tärkeimmät ominaisuudet, se lisää opetetun mallin tarkkuutta ja se vähentää harjoitteluaikaa.

Piirteiden valinta perustuu näytteiden ominaisuuksien osajoukon soveltuvuuden mittaamiseen ja maksimoimiseen luokittelutehtävässä. Valinta-algoritmit kuuluvat yleensä johonkin kolmesta ryhmästä; kääreisiin (*wrappers*), suodattimiin (*filters*) tai sulautettuihin algoritmeihin (*embedded algorithms*). Kääreet käyttävät hakualgoritmia etsimään mahdollisia piirreyhdistelmiä ja arvioi niistä kehiteltyjä malleja toisiaan vasten. Kääreet voivat olla laskennallisesti kalliita ja niillä on vaara ylioppia opetusdata. Suodattimet ovat samankaltaisia kuin kääreet, mutta mallin arvioinnin sijasta piirrekombinaatioita yksinkertaisesti suodatetaan. Sulautetut tekniikat on taas rakennettu suoraan koneoppimisen mallin sisään ja ne ovat niille spesifisiä.

Monet lähestymistavat ovat iteratiivisia optimointimenetelmiä, jotka kehittävät piirrejoukon, arvioivat tätä ehdokasjoukkoa, muokkaavat sitä ja arvioivat sitten uudestaan onko uusi osajoukko parempi kuin vanha. Paremmin toimiva osajoukko säilytetään ja optimointiprosessi alkaa alusta. Kaikkien mahdollisten piirrekombinaatioiden arvioiminen on yleensä epäkäytännöllistä, joten joissakin metodeissa määritellään pysäytysparametri, jonka määrittämän hakumäärän jälkeen haku lopetetaan ja siihen astisista kombinaatioista valitaan korkeimman pisteytyksen saanut ehdokasjoukko. Pysäytyskriteeri vaihtelee algoritmeittain; mahdollisia suureita ovat esimerkiksi se, että alijoukon saama pistetys ylittää kynnysarvon tai kombinaatioyritysten lukumäärä on saavuttanut maksiminsa.

Eri piirreosajoukkojen arviointitapojen valinta on hankalaa, koska valittujen piirteiden joukossa pyritään saavuttamaan monia eri tavoitteita. Samaan aikaan yleensä pyritään maksimoimaan kehitettävän mallin tarkkuuden mittaria sekä minimoimaan valittujen piirteiden lukumäärää, sillä se vaikuttaa mallin opetusnopeuteen ja yleistyskykyyn uuden datan luokittelussa. Monissa valintamenetelmissä käytetäänkin tarkkuusarvoa, jota sakotetaan valittujen ominaisuuksien määrällä.

2.5 Luokittelijoiden opetusmallit

Koneoppimisessa ja tilastotieteessä luokittelu on ongelma, jossa tarkoitusta varten kehitetyn tietomallin avulla määritetään, mihin luokkiin jokin ei-tunnettu havainto kuuluu. Luokittelun tietomalli kehitetään näytteiden avulla, joiden luokkatiedot tiedetään. Tämänkaltaisia näytteitä kutsutaan mallin opetusdataksi. Esimerkki luokitteluongelmasta voisi olla

sähköpostin luokitelu "roskapostiksi"tai "ei-roskapostiksi"tai diagnoosin antaminen tietylle potilaalle, kun potilaasta havaitaan tiettyjä ominaisuuksia kuten verenpaine, potilaan kokemat oireet ja potilaan aikaisemmat sairaudet.

Koneoppimisen terminologiassa luokittelua pidetään valvotun oppimisen ilmentymänä eli oppimisena, jossa on oikeasti tunnistettujen havaintojen opetusjoukko. Vastaava valvomatta tehty opettaminen tunnetaan klusterointina, ja se käsittää ryhmien yhdistämisen luokkiin, jotka perustuvat johonkin luontaiseen samankaltaisuuteen tai etäisyyteen.

Usein yksittäiset havainnot analysoidaan joukoksi kvantitatiivisia ominaisuuksia, jotka tunnetaan eri tavoin selittävinä muuttujina tai ominaisuuksina. Nämä ominaisuudet voivat olla kategorisia, kokonaislukuarvoja (esim. tietyn sanan esiintymistiheys sähköpostissa) tai reaaliaikaiset (esim. verenpaineen mittaus). Muut luokittelijat toimivat vertaamalla havaintoja edellisiin havaintoihin samankaltaisuuden tai etäisyyden funktion avulla.

Luokituksen toteuttava algoritmi, erityisesti konkreettisessa toteutuksessa, tunnetaan luokittelijana. Terminologia eri alojen välillä on melko vaihteleva. Tilastotieteessä, joissa luokittelu tehdään usein logistisella regressiolla tai vastaavalla menettelyllä, havaintojen ominaisuuksia kutsutaan muuttujiiksi ja ennustettavat luokat tunnetaan tuloksina, jotka katsotaan olevan riippuvaisia muuttujien mahdollisista arvoista. Koneoppimisessa havaintoja kutsutaan usein näytteiksi, selittäviä muuttujia kutsutaan piirteiksi, jotka on ryhmitelty piirrevektoreiksi ja mahdolliset luokitukset, jotka ovat ennustettavissa, ovat luokkia.

2.5.1 Naiivit Bayes -luokittelijat

Naiivi Bayes tai NB on yksinkertainen todennäköisyysjakaumiin perustuva tekniikka luokittelijoiden rakentamiseen. NB:stä puhuttaessa ei tarkoiteta suoraan mitään yksittäistä algoritmia, vaan algoritmien perhe, joka perustuu yhteiseen periaatteeseen: kaikki naiivit Bayes -luokittelijat olettavat, että tietyn piirteen arvo on riippumaton minkä tahansa muun piirteen arvosta jonkin tietyn luokan suhteen. Tästä piirteiden riippumattomuusoletuksesta tuleekin luokittelijaperheen etuliite, "naiivi", koska tämä olettamus käy toteen harvoin reaalimaailman ongelmissa.

Naiivi Bayes on ehdolliseen todennäköisyyteen perustuva malli. Jokaiselle näytteelle, jota edustaa piirrevektori $\overline{\mathbf{x}} = (x_1, x_2, ..., x_n)$, jossa on n kappaletta riippumattomia ominaisuuksia, saadaan näytteen todennäköisyys kuulua johonkin luokkaan L_m seuraavalla kaavalla.

$$p(L_m \mid \overline{\mathbf{x}}) = \frac{p(L_m)p(\overline{\mathbf{x}} \mid L_m)}{p(\overline{\mathbf{x}})} \tag{1}$$

Merkintä $p(x \mid y)$ tarkoittaa ehdollista todennäköisyyttä sille, että x pitää paikkanasa, suhteessa siihen, että y tapahtuu varmasti. Koska piirteiden oletetaan olevan riippumattomia ja jakolaskun nimittäjän ollessa vakioarvoinen voidaan kaava esittää edelleen seuraavalla

tavalla.

$$p(L_m \mid \overline{\mathbf{x}}) \propto p(L_m) \prod_{i=1}^{n} p(x_i \mid L_m) \tag{2}$$

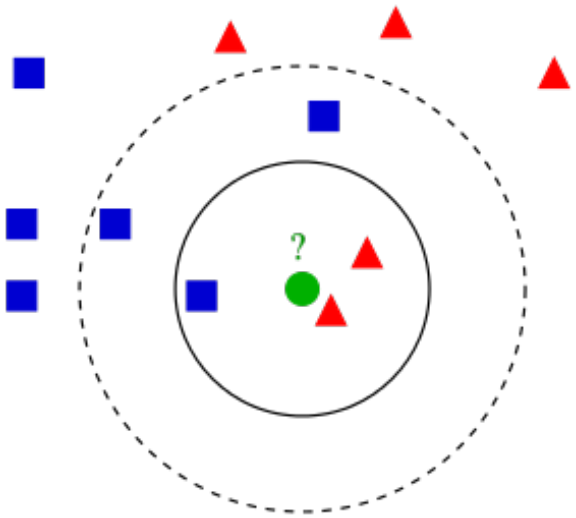
Kaavassa x_i tarkoittaa jotakin yksittäistä piirrevektroin arvoa. Tämän jälkeen saatua kaavaa 2 voidaan käyttää laskemaan todennäköisyydet näytteen kuulumisesta jokaiseen mahdolliseen luokkaan. Näistä suurimman todennäköisyyden saanut luokka valitaan lopulliseksi tulosluokaksi näytteelle.

Näennäisesti yksinkertaistetusta toimintaperiaatteestaan huolimatta naiivit Bayes-luokittelijat toimivat melko hyvin monissa monimutkaisissa todellisissa ongelmissa. Naiivin Bayesin etuna on se, että se vaatii vain vähän opetusdataa luokituksen edellyttämien piirteiden arvioimiseksi. Lisäksi naiivit Bayes-luokittelijat voivat olla yksinkertaisuutensa takia erittäin nopeita verrattuna kehittyneempiin menetelmiin.

Työssä käytettiin kahta erilaista naiivia Bayes-luokittelijaa. Multinomiaalinen naiivi Bayes (MNB) ja Bernoullin naiivi Bayes (BNB) ovat erikoistettuja versioita aikaisemmin esitetystä luokittelijasta. MNB on sopiva luokitteluongelmiin, jossa piirteet ovat diskreettejä arvoja. Kuitenkin myös reaalilukuarvot, kuten TF-IDF-vektorit, toimivan hyvin käytännössä. TF-IDF: n tapauksessa todennäköisyysjakauma parametrisoidaan vektoreilla jokaiselle luokalle, jossa piirteiden määrä on sanaston koko. BNB eroaa multonomiaalisesta naiivista Bayesistä siinä, että siinä piirteiden arvojen oletetaan olevan binäärisiä totuusarvoja. Molempia näistä malleista käytetään paljon tekstimuotoisten asikirjojen luokitteluun.

2.5.2 Lähimmän naapurin algoritmi

Lähimmän naapurin -algoritmi, eli k-NN (*k nearest neighbors*) on ei-parametrinen menetelmä, jota käytetään yksinkertaisuutensa vuoksi paljon luokitteluun ja regressioon. k-NN toimintaperiaate nojaa näytepohjaiseen oppimiseen, jossa uusia näytteitä arvioidaan opetusdataa vasten vain paikallisesti valitun kokonaislukuarvo k:n määrittämän joukon kesken. K-NN-algoritmissa ei ole varsinaista opetusvaihetta, vaan kaikki laskenta tapahtuu luokittelu- tai regressiovaiheessa. Kummassakin, regressiossa ja luokittelussa, algoritmin tuottama arvo koostetaan lähimpien harjoitteluesimerkkien mukaan. Tuloksen tyyppi riippuu siitä, kumpaan näistä k-NN-algoritmia käytetään. K-NN-luokittelussa tulos on luokan jäsen. Näyte luokitellaan k:n lähimmän naapurin enemmistöäänestyksellä sen mukaan mitä näiden lähimpien naapurien luokat ovat. Esimerkki k:n arvon vaikutuksesta uuden näytteen luokittelutulokseen voi nähdä kuvasta 3. Jos $k = 3$, uusi näyte määritetään kuuluvaksi punaisten kolmioiden luokkaan, kun taas jos $k = 5$, uusi näyte määritetään kuuluvaksi sinisten neliöiden luokkaan. Edistyneemmissä versioissa naapurien etäisyyksiä käytetään painottamaan niiden merkitystä äänestyksessä. K-NN-regressioissa tuotettu tulos on näytteelle naapurien perusteella laskettu keskiarvo.



Kuva 3. *k:n arvon vaikutus k-NN-algoritmille [1]*

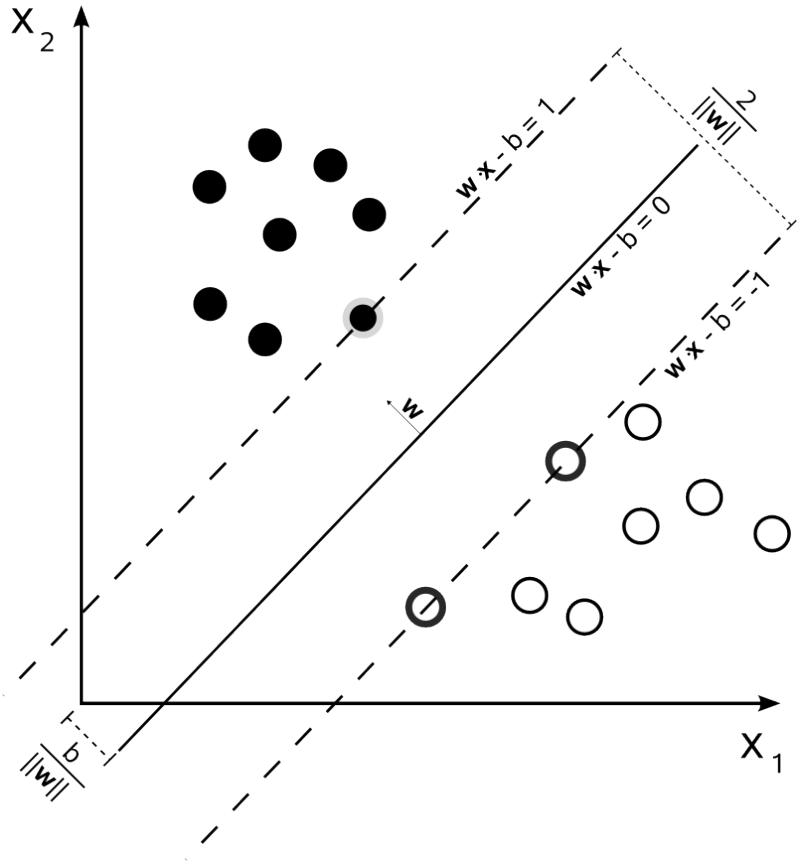
Esimerkkinä äänestysvaiheessa käytetystä painotustavasta on, että jokaiselle naapurille annetaan paino 1 / d, missä d on naapurin etäisyys käsiteltävästä näytteestä. K-NN-algoritmin erityispiirteenä on se, että se on herkkä datan paikalliselle rakenteelle, jolloin se voi ylioppia opetusdatan herkästi, kun sitä on vähän suhteessa piirteisiin. Herkkyydestään johtuen K-NN-algoritmin tarkkuuteen vaikuttaa myös paljon merkityksettömien ja häiriötä sisältävien piirteiden läsnäolo. Luokittelussa *k*:n arvona on perinteisesti käytetty paritonta kokonaislukua, sillä tällöin vältytään siltä, että äänestyksen äänet menisivät tasan kahden eri luokan välillä.

2.5.3 Tukivektorikoneet

Tukivektorikoneet tai SVM:t (*support vector machines*) ovat kahta edeltävää algoritmityyppiä edistyneempi tekniikka luokittelijan rakentamiselle. Ne perustuvat lineaariselle luokittelulle, jota tehdään rakentamalla tukivektorit, joiden tulisi erotella luokat toisistaan mahdollisimman hyvin. Joskus luokat eivät ole eroteltavissa toisistaan lineaarisesti. Tällöin on mahdollista siirtää ongelma korkeampiulotteiseen ominaisuusavaruuteen ja ratkaista tukivektorit lineaarisesti siellä.

Luokittelussa pyritään luokittelemaan uusi näyte kahdesta eri luokkaehdokkaasta. Tukiavektorikoneiden tapauksessa näytettä mallinnetaan piirreavaruudessa sijaitsevalla datapisteellä. Haluamme tietää, voidaanko eri luokkien datapisteet erotella jollain $(n-1)$-ulotteisella hypertasolla, jos piirteitä on n kappaletta. Jos voidaan, luokittelu on lineaarista tässä tapauksessa. Datapisteiden luokittelun mahdollistavia erilaisia hypertasoja on yleensä monia. Paras hypertaso on se, jolla on suurin etäisyys tai marginaali kahden luokan välillä. Kuva 4 kuvaa kahden luokan näytteiden avulla muodostettuja hypertasoja ja niiden väliin jäävää marginaalia. Täten hypertaso valitaan siten, että etäisyys siitä lähimpiin eri luokkien datapisteisiin molemmilla puolilla maksimoidaan.

Ongelma voidaan ilmaista äärellisessä piirteiden määrittämässä n-ulotteisessa avaruudes-



***Kuva 4**. Tukivektorikoneen hypertasot [4]*

sa, mutta usein on, että luokkien datapisteet eivät ole tällöin lineaarisesti eroteltavissa kyseisessä avaruudessa. Ratkaisuna on, että alkuperäinen äärellinen piirreavaruus heijastetaan paljon suurempaan ulottuvuuteen, minkä pitäisi tehdä luokkien erottelusta mahdollista. Tämä tehdään generoimalla alkuperäisten piirteiden pohjalta uusia piirteitä niin kutsutulla kernelifunktiolla. Tässä korkeampiulotteisessa avaruudessa ongelman luokat saattavat olla lineaarisesti eroteltavissa, jolloin niille voidaan rakentaa aikaisemman mallin mukaiset tukivektorit normaalisti. Korkeampiulotteisessa avaruudessa ratkaistut tukivektorit eivät ole enää lineaarisia, kun ne heijastetaan takaisin alkuperäisen piirrejoukon määrittämään avaruuteen.

2.5.4 Lineaarinen luokittelija

Lineaarinen luokittelija yhdistää lineaarisesti luokiteltavan näyteen ominaisuuksia tiettyjen painoarvojen avulla ja vertailee näin saatua tätä tulosta johonkin kynnysarvoon muodostaakseen luokittelutuloksen. Lineaarisen luokittelijan opettamisesta tehdään optimointiongelma, joka pyritään ratkaisemaan iteratiivisesti. Tällöin luokittelijaa käytetään yllä mainitulla tavalla, mutta saatua tulosta verrataan oikeaan arvoon, jolloin saadaan nykyisen mallin virhe. Tätä virhettä kututaan yleisesti häviöksi (*loss*) ja se tuotetaan jonkin häviöfunktion avulla, joka määrittää, mitä koneoppimisen mallia käytetään oppimiseen. Yhdistettynä vastavirta-algoritmiin se on vakiintunut algoritmi keinotekoisten hermoverk-

kojen opettamiseen.

Häviöllisellä oppimisella tarkoitetaan häviöfunktiolla tehtävää optimointia koneoppimisen mallin opettamisessa. Häviöfunktio antaa optimoinnin aikana opetettavan mallin tulokselle arvion häviönä eli kuinka paljon mallin tulos oli väärässä oikeasta arvosta. Tämän tuloksen avulla uudessa iteraatiossa voidaan mallin parametreja eli painoja säätää kohti ihanteellista mallia ja arvioida tätä uutta mallia taas häviöfunktion avulla. Eri häviöfunktiota hyödynnettäessä sitä voidaan käyttää esimerkiksi lineaaristen tukivektorikoneiden opettamiseen tai logistiseen regressioon.

Stokastinen gradienttimenetelmä eli SGD, (*stochastic gradient descent*) on yksinkertainen mutta erittäin tehokas lähestymistapa lineaaristen luokittelijoiden häviölliseen oppimiseen. SGD on suosittu iteratiivinen optimointialgoritmi monien koneoppimismallien opettamiseen. Se ei ole itsessään suoranainen koneoppimismalli, vaan se on tapa opettaa jotakin mallia häviöfunktion avulla, löytämällä pienin häviö jollekin mallille. Vaikka SGD on ollut jo pitkään koneenoppimisympäristössä, sitä on vasta äskettäin alettu hyödyntää enemmän suuren datajoukon koneoppimisongelmissa. Pseudokoodiesimerkissä 1 on kuvattu SGD:n toimintaperiaate [8, s. 8].

Algorithm 1 Stokastinen gradienttimenetelmä

```
1: Initialisoi θ_0
2: Asetetaan iteraatioiden määrä L
3: Asetetaan oppimisen aste α
4: for k = 1 ... L do
5:     θ_0 ← θ_m
6:     for i = 0 ... m do
7:         Arvioi g_k = ∇C_i(θ_k)
8:         θ_{k+1} ← θ_k − α g_k
9:     end for
10: end for
```

Vektori θ kuvaa mallin optimoitavien parametrien joukkoa, joka ensin asetetaan mielivaltaiseksi. Funktio C kuvaa optimointitehtävää ja g_k yhden iteraation näytteelle laskettua optimointitehtävän virhettä. Huomaa, että algoritmissa m tarkoittaa opetusesimerkkien lukumäärää. Näin ollen yhdessä iteraatiossa käydään läpi kaikki nämä opetusesimerkit.

SGD:tä on sovellettu menestyksekkäästi koneoppimisongelmiin, joissa käytettyjen piirteiden määrä on suuri ja ne ovat esiintymismääriltään harvalukuisia. Tämänkaltaisia ongelmia usein esiintyy tekstimuotoisen datan luokituksessa ja luonnollisen kielen käsittelyssä. SGD:n etuja ovat sen tehokkuus varsinkin isolle näytemäärälle sekä lukuisat optimointimahdollisuudet. Toisaalta suuri parametrimäärä tarkoittaa myös sitä, että kattava optimointityö voi viedä paljon aikaa. SGD on myös herkkä ei-normalisoidulle datalle, eli sille syötettävät piirteet tulisi aina normalisoida toisiinsa nähden.

2.5.5 Päätöspuut

Päätöspuihin perustuva koneoppiminen on yleisesti käytetty menetelmä luokittelussa ja regressiossa. Kuten aikaisemmin käsitelty kNN-algoritmi, päätöspuut voivat tuottaa sekä reaali- että diskreettiarvoja riippuen sitä kummassa käyttötarkoituksessa niitä hyödynnetään. Päätöspuiden periaatteena on luoda hierarkinen puumainen malli, joka pystyy ennustamaan näytteen piirrevektorin ilmentämien ominaisuuksien perusteella sille luokan tai regressiossa tavoitellun ominaisuuden arvon. Puu koostuu haaroista, solmuista ja lehdistä. Jokainen sisäinen solmu vastaa yhtä piirteelle tehtävää analyysiä tai vertailua, jolloin solmusta lähtevät alahaarat vastaavat tämän vertailun tuloksia. Puun ylintä solmua, josta suoritus aloitetaan kutsutaan juurisolmuksi. Jokainen lehti edustaa jotakin luokkaa tai regression arvoa.

Kun päätöspuu on rakennettu, uuden näytteen luokittelu on yksinkertaista. Lähtemällä juurisolmusta, solmussa olevaa vertailua arvioidaan näytteelle ja sen tuloksen mukaan seurataan jotakin solmusta lähtevää haaraa syvemmälle alasolmuihin. Haara johtaa joko toiseen sisäiseen solmuun, jonka vertailua taas arvioidaan tai lehtisolmuun. Kun saavutaan lehtisolmuun, näytteelle määritellään se luokka, joka liittyy tähän kyseiseen lehtisolmuun.

Kun päätöspuun toimintaa ajatellaan abstraktimmin se jakaa näytteitä osajoukkoihin niiden ominaisuuksien perusteella solmukohdissa. Se tekee tätä rekursiivisella tavalla, jota kutsutaan rekursiiviseksi osioinniksi. Kun jako ei lisää arvoa ennusteeseen tai kun solmun osajoukolla on sama tavoitemuuttujan arvo, rekursio on valmis. Jako solmukohdassa pyritään tekemään mahdollisimman optimaalisesti. Tähän on eri tapoja, mutta tärkeimmät niistä liittyvät entropiaan ja tarkemmin informaation saannin maksimoimiseen tehdyssä jaossa tai niin kutsuttuun gini-epäpuhtauden arvoon. Gini-arvo kertoo, kuinka usein satunnaisesti osajoukosta valittu alkio luokiteltaisiin väärin osajoukon luokkien todennäköisyysjakaumien perusteella.

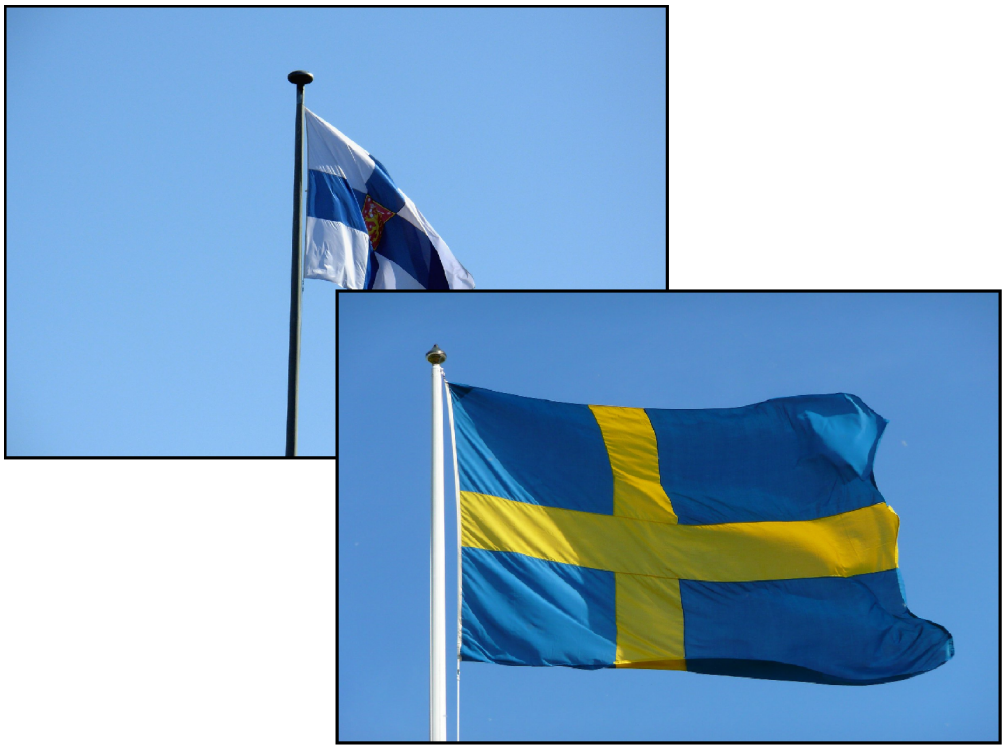
Yksittäisiä päätöspuita ei käytetä useinkaan todellisissa ongelmissa, koska ne usein ylioppivat opetusdatansa. Tehokaampia ratkaisuja ovat monista päätöspuista kootut kokoonpanomenetelmät. Kokoonpanomenetelmissä (*ensemble methods*) kootaan monia erilaisia päätöspuita yhteen, jotka äänestävät lopullisen tulosluokan arvon yhdessä. Eri päätöspuille jaetaan vain osa alkuperäisitä piirteistä, jolloin kaikki kokooonpanossa olevat puut ovat epätäydellisiä erikseen. Ne kuitenkin yhdessä muodostavat tarkan luokittelijan. Tällä edesautetaan opittavan asian yleistyskykyä ja vähennetään ylioppimista. Tämänkaltaisia ratkaisuja ovat satunnaismetsät sekä AdaBoost-tyyppiset tehostetut kokoonpanot.

Satunnaismetsät, eli RF-kokoonpanot (*random forest*) toimivat rakentamalla lukuisia satunnaisia päätöspuita luokittelijan harjoitteluaikana. RF:n harjoittelualgoritmi soveltaa niin kutsuttua pussitustekniikkaa yksittäisille puille. Pussituksessa opetusdatasta valitaan satunnainen osajoukko määrittämättömän monta kertaa ja opettaa aina yksittäisen puun tällä satunnaisella otannalla.

AdaBoost, eli Adaptive Boosting on toinen yleisesti käytössä oleva kokoonpanomenetelmä. Se on koneoppimisessa käytetty meta-algoritmi ja sitä käytetään rakentamaan joukko heikosti oppivia koneoppimisen malleja, yleensä päätöspuita, jotka yhdessä pystyvät oppimaan tehokaasti. AdaBoost on nimensä mukaisesti sopeutuva algoritmi. Luokitteluongelman yhteydessä se luo vaiheittain uuden joukon heikkoja luokittelijoita ja pyrkii oppimaan ongelman mukaisen luokkajaon niiden avulla. Tämän luokittelijajoukon tuloksia arvioidaan ja seuraavassa vaiheessa luotavat uudet heikot luokittelijat pyritään mukauttamaan aikaisempaan virheeseen siten, että ne luokittelisivat oikein ne tapaukset, jotka aiemmat luokittelijat ovat luokitelleet väärin. Kun ollaan tyytyväisiä, saatuun luokittelutulokseen tai ollaan saavutettu jokin asetettu lopetuskriteeri, opetus voidaan lopettaa. Koko luokittelijan luokittelutulos syntyy siten, että heikkojen oppijoiden tulos yhdistetään painotetulla summalla luokka- tai regressioarvoksi.

2.6 Moniluokkaisuus

Perustapaus koneoppivassa luokitteluongelmassa on näytteen luokittelu kahden eri luokan välillä. Tästä esimerkkinä kuvan 5 luokittelu Ruotsin tai Suomen lipuksi aineistossa, jossa esiintyy kuvia vain Suomen ja Ruotsin lipuista. Useimmin ongelmassa on kuitenkin mukana useampia luokkia ja tällöin ongelmasta tulee monimutkaisempi. Moniluokkaisuudella voidaan tarkoittaa kahta eri asiaa luokittelussa. Joko sillä tarkoitetaan sitä, että mahdollisia luokkakandidaatteja yhdelle näytteelle on useita tai jokaista näytettä kohden voidaan ennustaa useampi luokka-arvo. Englanniksi näitä kutsutaan vastaavasti multiclass ja multilabel -ongelmiksi.

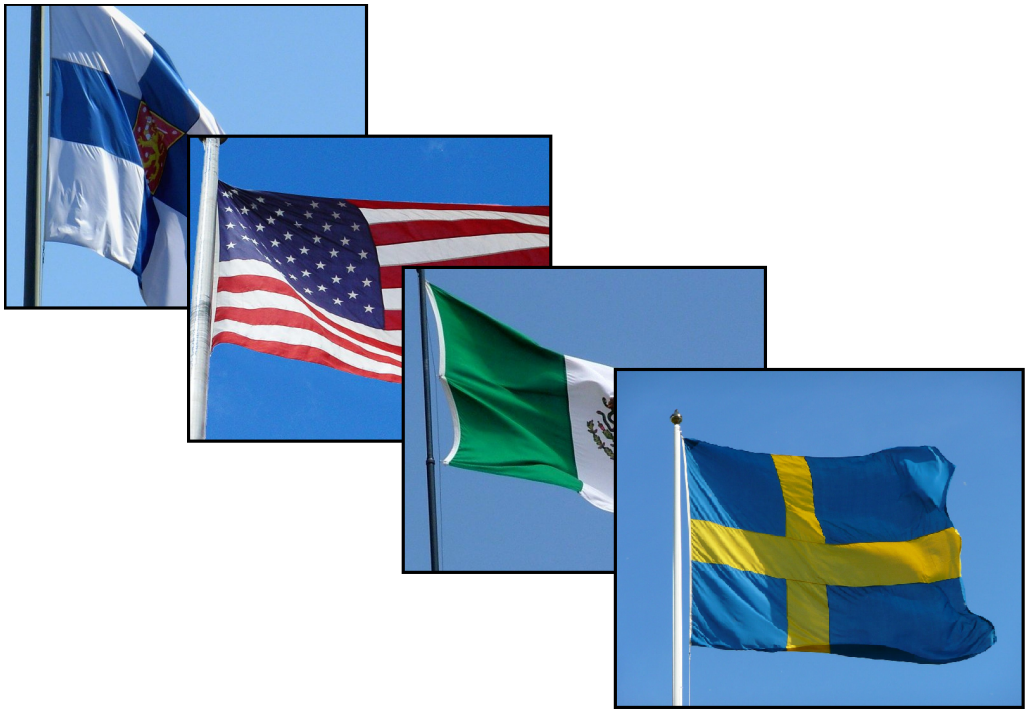


Kuva 5. *Kahden luokkakandidaatin ongelma [26], [18]*

Kaikki käytetyn Scikit-learn-kirjaston algoritmit tukevat multiclass-luokittelutehtäviä [21], mutta kaikki niistä eivät tue multilabel-tehtäviä suoraan. Käyttämällä esimerkiksi OneVsRestClassifier-wrapperia saadaan myös kaikki luokittelijat tukemaan multilabel-ongelmia.

2.6.1 Luokkakandidaattien määrä

Monissa luokittelutehtävissä arvioitavien luokkien määrä on enemmän kuin kaksi. Tällöin tehtävä luokittelu on multiclass-ongelma. Esimerkkinä tälläisestä tehtävästä voidaan käyttää edellistä kansallislippuesimerkkiä sillä erotuksella, että tehtävänä on luokitella useiden eri maiden lippuja. Tästä esimerkkinä kuva 6.



Kuva 6. *Useamman luokkakandidaatin ongelma [26], [18], [15], [12]*

Kaikki scikit-learn-kirjaston luokittelija-algortimit tukevat multiclass-tehtäviä havaitsemalla opetusdatassa esiintyvat luokat ja muodostavat luokittelijat havaitun luokkamäärän mukaan. Monet algortimit tukevat jo perusperiaatteeltaan moniluokkaisuutta, kuten päätöspuut ja lähimmän naapurin algoritmit.

2.6.2 Tulosluokkien määrä

Jos yksi näyte voi vastata useampaa eri luokkaa, tehtävää luokittelua kutsutaan multilabel-ongelmaksi. Sitä voidaan havainnollistaa lippuesimerkillä, joka on esitetty kuvassa 7. Aikaisemman esimerkin tapaan kuvissa on eri maiden lippuja ja tarkoituksena on luokitella lippukuvan lippu jollekin maalle. Erona aikaisempaan on se, että yksittäisessä kuvassa saattaa esiintyä useita eri lippuja.

Kuva 7. *Useamman tulosluokan ongelma [30]*

Tällöin ulostuloarvioina ei olekaan vain yksi maa, vaan joukko maita. Yleensä luokkien esiintymistä näytteissä esitetään esiintyvyysmatriisina, jossa rivi vastaa yhtä näytettä ja sarake luokkaa. Tällöin yksittäiset luokanesiintymisarvot ovat joko yksi tai nolla sen mukaan onko luokka edustettuna näytteessä.

2.7 Luokittelijoiden optimointi

Koneoppimismallit sisältävät monenlaisia sisäisiä parametreja, jotka ohjaavat sitä miten kyseessä olevasta mallista rakennetaan luokittelijoita tai regressiotyökaluja. Samantyyppinen koneoppimismalli voi vaatia täysin erilaisten parametrien käyttämistä erityyppisille syötedata- ja ongelmajoukoille. Näitä ominaisuuksia kutsutaan hyperparametereiksi, ja ne tulisi valita niin, että malli voisi ratkaista koneoppimisongelman mahdollisimman optimaalisesti.

Luokittelijoiden optimointi tai hyperparametrien optimointi on iteratiivinen kaksiosainen prosessi, jossa koostetaan mahdollisimman hyvä joukko erilaisia luokittelijakandidaatte-

ja erilaisilla parametreilla, jotka sen jälkeen arvioidaan parhaasta huonoimpaan jonkin arviointiparametrin tai tappiofunktion mukaan. Tätä iteratiivista prosessia on kuvattu kuvassa 1 tarkemmin. Tästä prosessista saadun arvojärjestyksen perusteella pystytään valitsemaan tarkemmin uudet parametrit ja niistä uusi joukko luokittelijoita, jolloin prosessi alkaa alusta. Näin saadaan kohdistettua haarukoimalla optimaalisten parametrien etsintä ja löytää ne mahdollisimman nopeasti. Tässä osiossa käsitellaan eri parametrien kartoitustapoja ja luokittelijakandidaattien arviontimenetelmiä.

2.7.1 Parametrien kartoitus

Hyperparametrien optimointi löytää hyperparametrien joukon, joka tuottaa optimaalisen mallin ja minimoi ennalta määritellyn häviöfunktion (*loss function*) käytetyllä syötedatalla. Hyperparametrien kartoittamiseen on monenlaisia työkaluja, käytetyin niistä on grid search. Grid search toimintaperiaate on yksinkertainen; sille määritellään joukko parametreja ja niiden mahdollisia arvoja, joiden eri yhdistelmät käydään systemaattisesti läpi. Kaikista mahdollisista parametriyhdistelmistä muodostetaan sitten luokittelijoita, jotka ovat valmiita koulutettaviksi ja arvioitaviksi.

Hieman grid search -menetelmää edistyneempi työkalu on satunnainen haku (*random search*). Satunnaiselle joukolle ei voi määritellä diskreettejä arvoja, kuten ruudukko haulle, mutta sille voi myös antaa jatkuvia arvovälejä. Sille määritellään myös kuinka monta parametriyhdistelmää, ja täten luokittelijakandidaattia, se muodostaa. Tämän jälkeen satunnainen haku muodostaa nimensä mukaisesti satunnaisia parametriyhdistelmiä määrittelyn lukumäärän. Tämän tavan etuna on sen helppokäyttöisyys ja laajempi parametriavaruus. Siinä missä ruudukkohaku etsi vain määriteltyjen parametriarvoja, satunnaishaku pystyy hyödyntämään niiden väliin jääviä arvoja. Todennäköisyyksien puolesta tämä on nopeampi tapa löytää optimaaliset arvot hyperparametreille.

2.7.2 Luokittelijoiden arviointi

Kun eri luokittelijakandidaatit on generoitu, ne pitäisi arvioida, jotta optimaaliset parametrit voidaan selvittää. Luokittelijoiden arvioinnissa täytyy huolehtia siitä, että arviot ovat mahdollisimman vertailukelpoisia ja syötedatasta riippumattomia sekä hyvin yleistäviä. Jos valikoidusta parametrijoukosta kehitetty luokittelija arvioidaan hyväksi yhden opetusdatajoukon pohjalta, ei voida olla varmoja yleistääkö luokittelija hyvin uudelle syötedatalle. Eräs arvioinnin tarkkuutta lisäävä ja tämän huomioonottava tekniikka on käyttää ristivalidaatiota.

Ristivalidaatiossa opetusdata jaetaan N:n yhtä suureen joukkoon ja arvioitava luokittelija opetetaan ja arvioidaan eri opetusdatan osajoukkokombinaatioilla siten, että N - 1 joukkoa käytetään harjoitudatana ja jäljelle jäänyttä osajoukkoa käytetään luokittelijan arvioinnissa. Näin tehdään jokaiselle osajoukkokombinaatiolle, jolloin luokittelijan lopullinen arvio

on eri kombinaatioiden arvioiden kesiarvo. Tällöin nähdään paremmin, kuinka hyvin luokittelija yleistää datalle, jota ei ole nähty aikaisemmin. Ristivalidaatiossa pystytään hyödyntämään kattavammin opetusdata ja se on erittäin tärkeää, kun opetusdataa ei ole paljon ja ylioppimisen riski on suuri.

2.7.3 Luokittelijan tarkkuus

Luokittelijoiden arviontitavan valinta on tärkeä, koska sillä on suuri merkitys optimoinnin toimivuudesta kyseiselle luokitteluongelmalle. Tärkeimpiä käsitteitä arvionnin kannalta ovat tarkkuus (*accuracy*) sekä täsmällisyys (*precision*) ja tunnistuskyky (*recall*). Tarkkuus on yleisesti käytetty yksinkertainen tapa mitata luokittelijan tarkkuutta. Se lasketaan yksinkertaisesti jakamalla oikeiden ennustusten määrä kaikkien ennustusten määrillä.

$$tarkkuus = \frac{\text{oikeiden ennustusten lkm}}{\text{kaikkien ennustusten lkm}}$$

Tarkkuus ei pysty kuitenkaan mittaamaan luokittelijan luokittelukykyä kaikissa tilanteissa. Jos mahdollisten luokkien välillä on epätasapainoa, eli jotkin luokat ovat yliedustettuina näytteissä muita luokkia enemmän, tarkkuus kuvaa huonosti luokittelukykyä. Tämä johtuu siitä, että luokittelija voi yksinkertaisesti luokitella kaikki näytteet kuuluvaksi paremmin edustettuun luokkaan ja saada korkean tarkkuusarvion. Tällaisia tilanteita varten tarvitaan parempia mittareita uokittelijoiden arviointiin.

2.7.4 Täsmällisyys ja tunnistuskyky

Täsmällisyys ja tunnistuskyky ovat edistyneempiä arviointitapoja, jotka pystyvät kertomaan luokittelijan luokittelukyvystä eri puolia kuin pelkkä tarkkuus. Niitä käytetään usein yhdessä kuvaamaan kattavammin luokittelukykyä ottaen myös huomioon luokkien välisen epätasapainon. Nämä arvot lasketaan samankaltaisilla kaavoilla kuin tarkkuus. Alla olevassa kuvassa 8 havainnollistetaan niiden laskentatapoja.

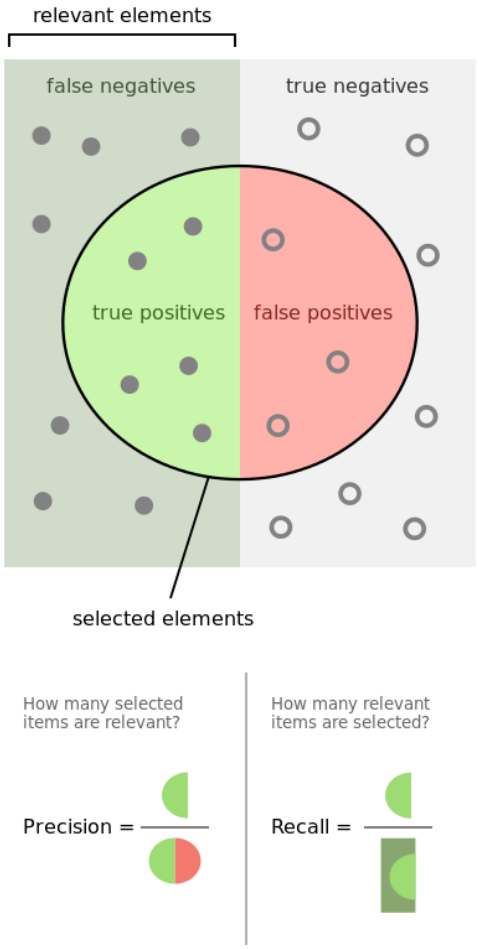
$$tsmllisyys = \frac{\text{oikeat positiiviset ennustukset}}{\text{kaikki positiiviset ennustukset}}$$

$$tunnistuskyky = \frac{\text{oikeat positiiviset ennustukset}}{\text{kaikki positiiviset oikeat}}$$

Täsmällisyydellä nähdään kuinka monta kertaa luokkaa ennustettiin oikein suhteessa kaikkiin kyseisen luokan positiivisiin ennustuksiin. Tunnistuskyvyllä nähdään, kuinka hyvin luokka oli tunnistettavissa muista luokista eli, kuinka usein oikea luokkaa löydettiin kaikista luokan esiintymisistä. Täsmällisyydestä ja tunnistuskyvystä on johdettu käsite nimeltä f1-arvo (*f1-score*). Sitä käytetään paljon luokitteluongelmissa, varsinkin kun luokkien välillä on epätasapainoa. Se on hyvin käyttökelpoinen yksittäinen mittari, sillä se ottaa huomioon sekä täsmällisyyden ja tunnistuskyvyn, jolloin se tunnistaa hyvin, jos luokittelija luokittelee yleisempiä luokkia paremmin kuin harvinaisempia. f1-arvon laskemiseen käytetään Scikit-learn-kirjaston sisäistä toteutusta, joka laskee sen alla olevalla tavalla.

$$f1 = 2 \cdot \frac{tasmallisyys \cdot tunnistuskyky}{tasmallisyys + tunnistuskyky}$$

f1-arvo voidaan tulkita täsmällisyyden ja tunnistuskyvyn keskiarvoksi, jossa f1-arvo saavuttaa maksiminsa arvossa 1 ja miniminsä pisteessä 0. Täsmällisyyden ja tunnistuskyvyn suhteellinen vaikutus f1-arvoon ovat yhtä suuret.



Kuva 8. *Täsmällisyyden ja tunnistuskyvyn periaate [31]*

3. OPETUSAINEISTO

Hyvät tietoaineistot ovat olennainen osa koneoppimisen tehokkaassa hyödyntämisessä. Tärkeät edistysaskeleet koneoppimisen saralla ajatellaan yleensä johtuvan oppimisalgoritmien tai käytettävien laitteistojen kehityksestä ja huomiotta yleensä jää sopivan dataaineiston saatavuus ja laatu. Korkealaatuinen luokiteltu opetusdata on yleensä vaikeata ja kallista tuottaa ohjattua koneoppimista varten, sillä suuren näytejoukon siistiminen ja ryhmittely vie paljon aikaa. Tähän työhön tarvitaan yleensä kyseisen ongelmakentän ammattilaisten käsin tekemää luokittelua sekä datan yhdenmukaistamista ja valikointia. Vaikka opetusdataa tarvittaisiinkin vain ohjaamattomassa koneoppimisessa, jolloin käsintehtävää luokittelua ei tarvitse tehdä, saattaa datan tuottaminen silti olla hankalaa tuottaa.

Tässä työssä käytetään terveydenhuollon tuottamaa tekstuaalista potilasdataa, joka koostuu hoitohenkilökunnan tekemistä hoitotapahtumamerkinnöistä. Tällaista dataa ei voida itse tuottaa vaan se tulisi hankkia oikeata hoitodataa sisältävästä aineistosta. Potilasdataa tuotetaan jatkuvasti valtavia määriä erilaisissa terveydenhuollon instituuteissa, mutta julkisesti sitä ei ole juurikaan saatavilla johtuen potilaan tietosuojasta Suomessa [24] ja ulkomailla. Tähän vaikuttaa suuresti myös erilaisten potilastietoa sisältävien tietokantojen pirstaleisuus ja yhteensopimattomuus.

3.1 Terveydenhuollon potilastieto

Terveydenhuollon tiedot voivat olla peräisin monista eri lähteistä, kuten geenikartoituksista tai sähköisistä potilasrekistereistä, jotka sisältävät muun muassa laboratoriotuloksia, kuvantamistutkimuksia ja diagnoositietoja. Terveydenhuollon tuottama tieto on hyvin hajanaista ja se on levinnyt maantieteellisesti hyvin laajalle alueelle. Ne koostuvat usein monenlaisista suljetuista ja koordinoimattomista arkistoista. Yhdessä arkistossa saattaa olla kuvantamistutkimuksien kuvamateriaalit ja toisessa samojen kuvantamistutkimusten raportit. Vaikka terveydenhuolto ja sen organisaatiot maailmalla toimivat monilla eri tavoilla, eri potilastietojärjestelmiin kerätty tieto on useasti sisällöltään samankaltaista. Relaatiotietokantoihin tallennettu potilastieto on kuitenkin useimmiten jonkin yleisesti käytetyn standardin mukaista (esimerkiksi CDA:n eri versiot, DICOM ja FHIR). Ei ole kuitenkaan olemassa mitään "plug and play-tyyppistä työkalua näiden datatyyppien yhteiskäyttöön vaan ne täytyy rakentaa itse tai käyttää valmiita työkaluja yksittäisten datatyyppien hyödyntämiselle [17].

Yhteistä näille arkistoille on se, että ne ovat suljettuja ja erinäiset potilastietojärjestelmien tietovarastoratkaisujen toimittajat ja ylläpitäjät varjelevat tätä dataa tarkkaan laillisista ja omaan etuun liittyvistä syistä. Yksityiset terveyspalveluiden tuottajat voivat katsoa, että

heidän kilpailuasemansa heikkenee ja tuotto vähenee, jos analytiikkaa ja koneoppimista käytettäisiin tehostamaan terveydenhuoltoa. Tästä syystä he eivät ole välttämättä halukkaita kehittämään tämänkaltaisia ratkaisuja tai jakamaan näitä menetelmiä mahdollistavaa dataa muille osapuolille [6]. Tietojen yhdistäminen suuriin, integroituihin tietokantoihin, jotka tukisivat syvällistä tautien ymmärtämistä ja parannuskeinojen kehittämistä koneoppimisen avulla, on vaikeaa. Tämä olisi kuitenkin erittäin tärkeää terveydenhuollolle uusien innovaatioiden ja palveluiden kehittämisessä koneoppimisen näkökulmasta. Sillä olisi myös tieteellistä arvoa, koska datan vapautuminen mahdollistaisi uusien ja laajojen tutkimusten tekoa terveydenhuollon piirissä.

Tiedonjaon pelisäännöt eivät ole tällä hetkellä selkeitä eikä tiedon jakamiseen ole kehitetty kunnollisia toimintamalleja. Tähän liittyy myös pelko yksityisyydensuojan vaarantumisesta sekä mahdolliset tietoturvaongelmat [7]. Tärkeää on muistaa, että potilasdata on erittäin arkaluontoista alkuperäisessä muodossaan ja se tulisi pystyä anonysoimaan täysin ennen kuin sitä voitaisiin hyödyntää. Kattavasta tietoturvallisuudesta pitäisi siten varmistua sen välittämisessä ja säilyttämisessä. Suomessa on mietitty potilasdatan laajempaa hyödyntämistä, jolloin voisi tulla mahdolliseksi hankkia anonymisoitua terveydenhuollon tuottamaa tietoa potilaiden hoidosta [28]. Käyttämällä anonymisoitua dataa pystyttäisiin osaltaan välttämään tietoturvaan ja yksityisyydensuojaan kohdistuvia riskejä.

Työssä päädyttiin käyttämään MIMIC-III-tietokantaa, joka sisältää aikaisemmin kuvatun kaltaista anonymisoitua dataa. MIMIC-III:n käyttöoikeutta pitää myös erikseen anoa hakemuksella, jonka yhtenä vaatimuksena on potilasdatan käyttöä koskevan verkkokurssin suorittamista. MIMIC-III on hyvin harvinainen siinä, että se sisältää erittäin kattavasti erityyppistä potilaan hoidosta kerättyä tietoa ja tärkeimpänä suuren määrän tekstimuotoisia hoitomerkintöjä. Seuraavassa aliluvussa käsitellään tarkemmin MIMIC-III:n käyttöoikeuden hakemista ja rakennetta.

3.2 MIMIC-III

Koneoppimistutkimuksessa päädyttiin käyttämään opetus- ja testiaineistona MIMIC-III-tietokantaa (Medical Information Mart for Intensive Care III) [5], joka kokoaa vuosina 2001-2012 kerättyä potilaiden terveydenhuoltoon liittyvää dataa Beth Israel Deaconess Medical Center-sairaalan teho-osastolta. MIMIC-III sisältää yli 400 000 terveystietuetta erilaisina tietotyyppeinä, mitatuista sydänkäyristä hoitohenkilökunnan kirjoittamiin muistiinpanoihin. Tiedoista on poistettu kaikki mahdolliset henkilötiedot, joten sitä pystyy käyttämään melko vapaasti tutkimuskäytössä. Tietokanta sisältää hyvin kattavasti erityyppistä tietoa, kuten esimerkiksi elintoimintojen mittaustietoja, laboratoriotuloksia, hoitohenkilökunnan hoitoon liittyviä kirjauksia sekä kuvantamisraportteja potilaan hoidon eri vaiheilta.

MIMIC on usealla tapaa merkittävä tietokantahanke. MIMIC-tietokantaa on lähdetty keräämään tutkijoita varten, joille on usein vaikeata saada käyttöönsä laadukkaita tietoläh-

teitä. Se on myös erittäin laaja kokonaisuus sisältäen yli kymmenen vuoden ajalta mittavan kokonaisuuden erityyppisiä tietomalleja. Se kattaa suurimman osan oleellisesta tiedosta, joka syntyy potilaasta hänen sairaalakäyntinsä aikana. MIMIC tukee kattavuudensa puolesta hyvin uusia sairaanhoidon, kliinisen päätöksenteon ja sähköisten työkalujen kehittämistä sekä monipuolista analyyttistä tutkimusta.

Työn kirjoitushetkellä uusin MIMIC-versio oli MIMIC-III v1.4, joka julkaistiin 2.6.2016. Siihen kuuluu yli 58 000 sairaalahoitoa 38 465 aikuiselle ja 7 875 vastasyntyneelle. Tiedot ulottuvat vuoden 2001 kesäkuusta aina vuoden 2012 lokakuuhun asti. Vaikka tietokanta on riisuttu henkilötiedoista, se sisältää edelleen yksityiskohtaisia tietoja potilaiden kliinisestä hoidosta, joten on varmistuttava siitä että sen sisältämää tietoa käsitellään asianmukaisella huolellisuudella ja kunnioituksella.

Tietokannan käyttöoikeutta täytyy erikseen hakea PhysioNet-sivuston kautta (https://mimic.physionet.org). Ennen kuin hakemusta voidaan käsitellä täytyy suorittaa tutkimustiedon käsittelyyn liittyvä kurssi Citi Program -hankkeen sivuston kautta (https://about.citiprogram.org/en/homepage/). Kun yksinkertainen web-kurssi on suoritettu voidaan hakemus lähettää kurssisuoritustodistuksen kanssa PhysioNet-sivuston kautta (https://physionet.org/works/MIMICIIIClinicalDatabase/access.shtml). Hakemuksen hyväksyminen kestää noin viikon ja vastauksena lähetetään sähköposti, jossa annetaan ohjeet tietokannan laatamisesta ja käytöstä.

3.2.1 Tietokannan rakenne

MIMIC on relaatiotietokanta, joka sisältää taulukkomuotoista hoitotietoa, jota on kerätty teho-osastolta. Tietokanta on saatavilla suurina UTF-8-merkistökoodattuina csv-tiedostoina, jotka sisältävät aina yhden tietokannan taulukon. Taulukot ovat keskenään linkitettyjä yksilöivien tunnisteiden avulla, joilla on yleensä pääte "ID". Taulukot, jotka alkavat etuliitteellä "D_"ovat sanakirjoja ja antavat määritelmiä tunnisteille. Esimerkiksi jokainen DIAGNOSES_ICD-taulun rivi sisltää yhden diagnoosin, mutta se ei sisällä kyseisen vaivan selkokielistä nimeä. Varsinainen diagnoosin nimi löytyy D_ICD_DIAGNOSES-taulusta, jonka kautta on mahdollista tunnistaa mikä diagnoosi vastaa DIAGNOSES_ICD-taulusta löytyvää ICD9_CODE-arvoa.

Seuraavia taulukoita käytetään potilaiden sairaalaviipymien seurantaan:

- ADMISSIONS: Jokainen yksittäinen sairaalahoitoviipymä kullekin potilaalle
- CALLOUT: Tiedot siitä, milloin potilas lähetettiin pois teho-osastolta
- ICUSTAYS: Jokainen yksittäinen teho-osastoviipymä ja sen tiedot
- PATIENTS: Jokainen yksilöllinen potilas
- SERVICES: Kliininen palvelu, jota potilas on käyttänyt
- TRANSFERS: Potilaiden siirrot sairaalan sisällä, mukaan lukien osaston sisäänpääsy ja poislähtetys

Yhdelle potilaalle kuuluu yksi tai useampi sairaalaviipymä ja yhdelle sairaalaviipymälle voi kuulua yksi tai useampia osastoviipymiä.

Seuraaviin taulukoihin on koottuu tehohoidon aikana kerättyjä tietoja potilaasta.

- CAREGIVERS: Jokainen hoitohenkilö, joka on tallentanut tietoja tietokantaan
- CHARTEVENTS: Kaikki potilaiden kaaviomuotoiset havainnot
- DATETIMEEVENTS: Kaikki tallennetut havainnot, jotka ovat päivämääriä, esimerkiksi näytteidenottoaika
- INPUTEVENTS_CV: Potilaille annostellut lääkeaineet, joita seurataan Philips CareVue -järjestelmällä
- INPUTEVENTS_MV: Potilaille annostellut lääkeaineet, joita seurataan iMDSoft Metavision -järjestelmällä
- NOTEEVENTS: Hoitohenkilökunnan tekemät potilaskirjaukset. Näihin kuuluu yleiset hoitoon liittyvät raportit sekä kirjalliset EKG- ja kuvantamistulokset.
- OUTPUTEVENTS: Potilaille tehtyjen mittausten tiedot
- PROCEDUREEVENTS_MV: Potilaille suoritetut hoitotoimenpiteet, joita seurattiin käyttäen iMDSoft MetaVision -järjestelmää.

Seuraavissa taulukoissa on kerätty tietoja sairaalan tietojärjestelmästä:

- CPTEVENTS: Hoitotoimenpiteet, jotka on kirjattu CPT-koodeiksi (*Current Procedural Terminology*)
- DIAGNOSES_ICD: Sairaalassa annetut diagnoosit, koodattu käyttäen kansainvälistä tautiluokitusjärjestelmää (ICD-9)
- DRGCODES: Diagnoosiin liittyvät ryhmät (DRG), joita sairaala käyttää laskutustarkoituksessa
- LABEVENTS: Laboratoriomittaukset potilaille sekä sairaalassa että sen ulkopuolella
- MICROBIOLOGYEVENTS: Mikrobiologiset mittaukset
- PRESCRIPTIONS: Lääkkeet, jotka on määrätty tietylle potilaalle
- PROCEDURES_ICD: Potilaille tehdyt hoitomenetelmät, koodattu käyttäen kansainvälistä tautiluokitusjärjestelmää (ICD-9)

Seuraavat taulukot ovat avustavia sanakirjoja:

- D_CPT: Nykyisen menettelytapasäännöstön koodit
- D_ICD_DIAGNOSES: Diagnoosien tiedot kansainvälisen tautiluokitusjärjestelmän (ICD-9) koodeille

- D_ICD_PROCEDURES: Hoitotoimenpiteiden tiedot kansainvälisen tautiluokitusjärjestelmän (ICD-9) koodeille
- D_ITEMS: MIMIC-tietokannassa esiintyvät mittaustoimenpiteiden termit
- D_LABITEMS: Laboratoriotutkimuksen liittyvät termit

Koska työssä käytetty data päätettiin rajata tekstimuotoisiin terveydenhuollon ammattilaisten kirjaamiin muistiinpanoihin ainoa varteenotettava taulukko tietokannassa oli NOTEEVENTS, joka sisälsi potilaskäyntiin liittyviä muistiinpanoja. Toinen taulukko, jota käytettiin koultusdatan luokittelutietona oli DIAGNOSES_ICD, joka sisälsi sairaalakäyntien lopulliset diagnoosit.

3.2.2 ICD-9-koodisto

Kuten ylläolevassa MIMIC-III rakenteen kuvauksessa todettiin diagnoosit kuvataan ja luokitellaan hierarkisella ICD-9-koodistolla. ICD (International Classification of Diseases, ICD) eli Kansainvälinen tautiluokitus on vakiomuotoinen luokittelujärjestelmä terveydenhuollossa tehtäville diagnooseille.

Maailman terveysjärjestö (WHO), joka on Yhdistyneiden Kansakuntien terveydenhuoltoa ohjaava ja koordinoiva alijärjestö, ylläpitää ja kehittää ICD:tä. ICD on suunniteltu terveydenhuollon diagnosointia avustavaksi luokitusjärjestelmäksi, joka tarjoaa hierarkisen koodistojärjestelmän sairauksien ja niihin liittyvän lisätiedon luokittelemiseksi. Tämä järjestelmä on suunniteltu kuvaamaan potilaan terveydelliset olosuhteet kattavasti enintään kuudesta merkistä koostuvan kaksiosaisen koodin avulla. Koodin ensimmäinen osa, niin kutsuttu pääluokka, kuvaa taudin tai oireen karkealla tasolla ja loppuosa tarkentaa sitä ja antaa lisätietoa potilaan tilasta. Kaikki taudit tai oireet saman pääluokan koodin alla ovat samankaltaisia.

ICD on tärkeä projekti, jolla tilastollisesti luokitellaan kaikki terveysongelmat ja annetaan diagnostista apua. Maailman terveysjärjestö julkaisee ICD:tä, jota käytetään maailmanlaajuisesti sairastavuus- ja kuolleisuuslukuihin. Tämä järjestelmä sai alkunsa kainsainvälisten kuolleisuustilastojen keräämistarpeesta ja se myöhemmin kehittyi sisältämään tiedot myös muistakin kuin kuolleisuutta aiheuttavista sairauksista. Siitä on kehitetty työkalu, joka helpottaa kansainvälistä vertailukelpoisuutta näiden tilastojen keräämisessä ja tutkimisessa. ICD:tä tarkistetaan ja uusitaan määräajoin kasvavan terveyden ja lääketieteen tiedon myötä. Uudistustyö on parhaillaan käynnissä ICD-11-version julkaisemiseksi, joka luultavasti tapahtuu vuoden 2018 aikana. Uusin käytössä oleva versio, ICD-10, on peräisin vuodelta 1992. Siitä julkaistaan säännöllisin määräajoin pieniä päivityksiä ja kolmen vuoden välein isompia päivityksiä [11].

MIMIC-III-tietokannan käyttämä ICD-9-koodisto on julkaistu 1978. Kuten tuoreempi ICD-10, se on rakenteeltaan hierarkinen ja sen koodit ovat maksimissaan kuusimerkkisiä.

ICD-9:n koodit voidaan ajatella koostuvan kahdesta pisteen jakamasta osasta; pääluokasta ja tarkennusosasta. Esimerkkinä aikuisiän diabetes ilman mainittavia komplikaatioita, jonka ICD-9-koodi on 250.00. Pääosan voidaan ajatella kuuluvan yhteen kolmesta pääryhmästä, jotka ovat.

- 001 - 999, jotka käsittelevät normaaleja sairauksien, tapaturmien ja vaivojen diagnooseja
- E001 - E999, jotka käsittelevät ulkopuolisten vammojen syitä
- V01 - V91, jotka ovat terveydentilaan ja terveydenhoitoon liittyviä täydentäviä luokkia

Niin kuin aikaisemmin käsiteltiin, koodin loppuosa kuvaa yleensä lisäoreita tai pääluokan määrittelemän vaivan erityispiirteitä. Diabeteksen 250-koodin tapauksessa, tarkennusosa kuvaa esiintyykö Diabeteksen kanssa siihen liittyviä komplikaatioita. Esimerkiksi koodit välillä 250.60 - 250.63 kuvaavat diabetestä, jonka yhteydessä esiintyy hermostollisia oireita.

3.3 Tietolähteiden haasteita

Iso osa koneoppimisen soveltamisesta johonkin ongelmaan on datan käytön ja sen esikäsittelyn suunnittelua [27, s. 318]. Koneoppimista varten hankittu data on harvoin käyttökelpoista sinällään ja sen esikäsittelyä on käsitelty jo luvussa 2.2. Kyseisessä luvussa käsiteltiin datan esikäsittelyä yleisesti riippumatta käytetystä datasta. Datan valikointia ja esikäsittelyä tulisi myös tehdä käytetyn datan näkökulmasta ja räätälöidä se käsillä olevan ongelman mukaan sopivaksi. Monista datalähteistä saatu materiaali sisältää paljon epäolennaista tietoa ja varsinainen käytettävä data täytyy usein parsia kokooon olennaisen datan hajallaan olevista osasista. Käytettyyn datalähteeseen saattaa liittyä myöds muita ongelmia, jotka tulee ottaa huomioon opetusdataa koostettaessa. Seuraavissa aliluvuissa käsitellään terveydenhuollon dataan ja käytettyyn datalähteeseen littyviä ongelmia koneoppimisen näkökulmasta.

3.3.1 Terveydenhuollon hoitomerkintöjen haasteet

Terveydenhuollon tietolähteitä on varsin heikosti saatavilla yleiseen käyttöön. Ne joita on, sisältävät lähinnä kuvamuotoista tai numeerista dataa. Tämä on täysin ymmärrettävää, sillä niissä olevat tiedot ovat tietosuojalain piirissä. Tutkijoiden täytyy erikseen anoa lupaa potilasdatan keräämiseen ja tutkimiseen. Tämän jälkeen koostettua dataa ei voi julkaista vapaasti vaan se täytyy varastoida tai hävittää tietoturvallisesti, jotta vältyttäisiin tiedon väärinkäytöltä. Henkilötiedot siis muodostuvat ongelmaksi medikaalidatan keräämisessä ja käytössä. Onneksi on olemassa joitakin tietokantoja, joista on poistettu henkilötiedot ja joita voi siten käyttää melko vapaasti. On myös mahdollista käyttää keinotekoista potilasdataa generoivia työkaluja työssä, kuten Syntheaa [32]. Se ei kuitenkaan sopisi työn

koneoppimistavoitteiseen kovin hyvin, sillä generoidulla datalla opetettu koneoppimisen malli mallintaisi vain generaattorin sisäisiä tautimalleja.

Saatavilla olevat hoitomerkintöjä sisältävät tietokannat eivät ole sinällään kovin käyttökelpoisia koneoppimisessa. Tämän vuoksi niistä on johdettu piirteitä luvun 2.3 mukaisilla tavoilla, jotka ovat numeerisia ominaisuuksia datasta. Näitä ovat esimerkiksi sanoista kootut termien esiintymismäärät ja esiintymistiheydet. Tätä tietoa voidaan käyttää perinteisissä koneoppimisalgoritmeissa paremmin, sillä se on laskennallisesti ymmärrettävässä muodossa. Piirteiden johtamisen lisäksi täytyy tehdä muita operaatioita datalle, jotta se olisi mahdollisimman kuvaavaa diagnosoinnin kannalta. Hoitotyöstä tehtävät merkinnät ovat suurelta osin ymmärrettäviä muistiinpanoja ja raportteja, mutta sen seassa vilisee myös joukko vieraita termejä ja lyhenteitä. Kun samasta käsitteestä käytetään montaa eri ilmaisumuotoa niistä kaikista muodostetaan erilliset piirteet, kun ne voitaisiin yhdistää yhden piirteen alle. Esimerkiksi diabetes voi esiintyä muistiinpanoissa muodoissa "diabetes", "diabetes mellitus", "dm", "type 2 diabetes"ja "t2dm". Nämä olisi hyvä pystyä normalisoimaan yhden termin alle ja tähän löytyykin joitakin eri työkaluja, kuten esimerkiksi DNorm [13].

Yksi terveydenhuollon muistiinpanojen ongelmista on se, että ne ovat riippuvaisia toisistaan ja niiden keskinäisistä esiintymisajoista. Merkityksen kannalta on oleellista, että terveyteen liittyvillä tapahtumilla on jokin järjestys toisiinsa nähden. Esimerkkinä on tapaus, jossa potilas kärsii huimauksesta ja käy lääkärin vastaanotolla:

1. Hän saa reseptilääkkeen, joka on tarkoitettu huimauksen hoitoon.
2. Potilas ei kuitenkaan saa mitään apua lääkkeestä.
3. Potilas menee uudestaan lääkäriin ja saa toisenlaista reseptilääkettä.
4. Tämä lääke tehoaa potilaan vaivaan ja huimaus loppuu.

Ilman järjestystä ylläolevassa esimerkissä ei olisi selvä kumpi lääke auttoi potilasta vaivassaan. Tämä on hyvin yksinkertaistettu esimerkki mahdollisesta tapahtumaketjusta, mutta ilman järjestystä asioiden merkitys saattaisi muuttua. Tätä ongelmaa olisi hyvä kompensoida valitsemalla dataa, jossa ei ole järjestystä tai sellaista dataa, jossa sillä ei ole merkitystä. Valitsemalla datasta johdetut piirteet tietyllä tavalla voidaan myös kompensoida järjestystiedon puuttumista. On myös olemassa joukko tapoja hyödyntää datan aikariippuvuutta koneoppimisessa ja niistä kerrotaan lisää luvussa 5.1.

3.3.2 MIMIC III -tietokannan ongelmat

Data, joka on peräisin oikeasta tuotantokäytöstä sisältää lähes aina epäpuhtauksia eikä ole suoraan sovellettavissa koneoppimisessa. MIMIC ei ollut poikkeus ja se sisälsi muutamia haasteteita, jotka piti ratkaista ennen kuin sitä voitaisiin hyödyntää opetusdatan generoimiseen. Ihanteellinen data olisi ollut peräisin monipuolisesti erilaisista hoitoympäristöistä, joissa hoidetaan kattavasti kaikentyyppisiä potilaita. MIMIC-data oli kuitenkin

peräisin teho-osasto-ympäristöstä (*intensive care unit*). Tämä tarkoitti sitä, että suurin osa potilaista oli melko vanhoja sekä monisairaita ja sairauksista olivat yliedustettuina siten sydän- ja verisuonitaudit ja syövät. Joukkoon mahtui myös synnyttäjiä ja tapaturmaan joutuneita potilaita, mutta vanhemmat ihmiset olivat silti selvästi suurin potilasryhmittymä [5, s. 3] Tällä datalla kehitettävät luokittelijat mitä luultavammin luokittelevat hyvin uutta dataa, joka on peräisin myös teho-osastolta. Näille luokittelijoille on kuitenkin luultavasti paljon haastavampaa luokitella terveyskeskuskäynneiltä saatuja muistiinpanoja. Tätä ongelmaa on vaikeaa kiertää ja täytyy vain hyväksyä, että kehitetyt luokittelijat toimivat paremmin ICU-ympäristöstä kerättyjä muistiinpanoja käytettäessä.

Verisuonitaudit olivat yliedustettuina tautien joukossa, jolloin ne painottuivat myös opetusdatassa. Jotkin luokittelijat ovat alttiimpia luokkien epätasapainolle ja niistä tulee huonompia luokittelijoita, kun opetusdatan luokkien suhteessa on epätasapainoa. Epätasapainoinen data johtaa herkästi luokittelijoihin, jotka ehdottavat suoraan yleisempää luokkaa luokittelun tuloksena. Toisin kuin aikaisemman ongelman kanssa, tätä ongelmaa on mahdollista kiertää eri tavoilla. Kuten aliluvussa 2.7.2 todettiin, eräs tapa on käyttää luokittelijoiden arvosteluparametrina jotain muuta kuin suoranaista luokittelun ulkoista tarkkuutta (*accuracy*). Tämänkaltainen arviointitapa on esimerkiksi sisäisen tarkkuuden eli täsmällisyyden (*precision*) ja löytökyvyn (*recall*) käyttäminen yhdessä. Arvosteluparametrina f1-arvo (*f1-score*) on täsmällisyyttä ja löytökykyä helppokäyttöisempi, sillä se lasketaan niiden avulla ja se ottaa siten ne molemmat huomioon. Jos luokittelija arvaa suoraan yleisempää luokkaa, harvinaisempi luokka saa hyvin pienen löytökyvyn arvon, jolloin sille laskettu f1-arvo on myös pieni. Monille käytetyn koneoppimissovelluskirjaston scikit-learnin luokittelija-algoritmeista tukevat niin sanottua *sample balancing*-tekniikkaa, jossa luokittelija pyrkii analysoimaan opetusdatan luokkien esiintymisuhteet ja tasapainottaa niiden vaikutusta luokittelijan opettamisen aikana. On myös mahdollista käyttää luokittelijoita, jotka eivät ole niin alttiita luokkaepätasapainolle. Tällaisia luokittelijoita ovat ainakin päätöspuut, jotka kestävät hyvin tätä ilmiötä.

Työn yhtenä ongelmana on lääkäreiden tekemät diagnoosit, joita käytettiin opetusdatan tiedettyinä luokkina. Lääkärit tekevät usein virheellisiä diagnooseja ja käytetty aineisto sisälsi paljon merkintöjä joiden tarkoituksena oli korjata aikaisemmissa merkinnöissä esitettyjä diagnooseja. Monet potilaalle annetuista diagnooseista olivat vain vanhojen diagnoosien kertausta, eikä uusille oireille annettu mitään uutta diagnoosia. Tästä voidaan päätellä että lääkärit diagnosoivat potilaat oikein ja uudet oireet todella liittyivät aikaisemmin diagnosoituihin tauteihin. Toinen vaihtoehto on, että lääkäri ei osannut diagnosoida potilaan uutta sairautta. Tämä on täysin varteenotettava vaihtoehto, sillä lääkärit tekevät usein diagnosointivirheitä ICU-ympäristöissä [33, abstrakti].

Merkinnöistä iso osa saattoi myös olla osittain toistensa kopioita [25]. Merkintöjen samankaltaisuus saattaa johtua vallalla olevasta epävirallisten merkintäpohjien käyttämistä potilaiden uusien merkintöjen kirjaamisessa. Siinä terveydenhuollon ammattilainen kopioi hänen hallussaan olevan aikaisemmin tekemänsä merkinnän uutta merkintää varten. Tämän seurauksena mahdolliset koneoppimisen luokittelijat kärsivät tästä merkintöjen

samankaltaisuudesta eikä pystytä erottamaan datan perusteella kunnolla merkityksellisiä piirteitä. On mahdollista että osa lopullisista diagnooseista voidaan löytää suoraan opetusdatan piirrevektoreista, sillä diagnoosien nimiä ja ICD-9-koodeja esiintyy joissakin MIMIC-III:n muistiinpanoissa. Tältä osin data ei ole ihanteellista diagnoosien luokittelun opettelussa, kun luokittelija voi saada siltä kysytyn diagnoosin nimen suoraan syötetekstin seassa. Sellainen piirre on hieman liian tarkka indikaattori lopulliselle diagnoosille.

4. TUTKIMUKSEN TOTEUTUS

Työssä toteutettiin tutkimus, jonka tarkoituksena oli koostaa koneoppimisen avulla potilaiden diagnosointia avustavia malleja. Opetusdatana käytettiin MIMIC-III-tietokantaa, jossa oli hoitohenkilökunnan tekemiä tekstipohjaisia hoitomerkintöjä ja raportteja, joita oli kirjattu potilaiden hoitojaksojen aikana. Näistä hoitomerkinnöistä mallien tulisi pystyä ennustamaan potilaille kirjatut lopulliset diagnoosit. Tutkimus toteutettiin Python-ohjelmointikielen ja Scikit-learn-kirjaston avulla rakennetulla sovelluksella, jonka lähdekoodi on saatavilla osoitteessa:

https://github.com/verdantred/Assisting-diagnosis-using-medical-textual-records

MIMIC-III-tietokannan sisältämä data oli hyvin monipuolista ja työssä päädyttiin käyttämään potilaan hoidosta tehtyjä tekstimuotoisia merkintöjä ja raportteja. Yksi sairaalakäynti sisältää yhden potilaan, jolla on N määrä hoitomerkintöjä, jotka voivat vaihdella hoitajan tilannekatsauksesta kuvantamislääkärin tekemään raporttiin. Potilas saa aina viimeisenä merkintänään lääkärin loppuarvion, jossa käydään läpi potilaan tila ja saama hoito sairaalakäynnin aikana. Siihen merkitään myös muita terveystietoja kuten potilaan sekä tämän lähipiirin sairaushistoria ja potilaan elintavat. Loppuarvion loppuun merkitään käynnin diagnoosit ja määrätyt lääkkeet.

Koneoppimisen näkökulmasta voidaan ajatella, että opetusdatasyötteenä käytetään tekstidokumentteja, jossa on luokkatietoina joukko diagnooseja. Mahdollisia luokkia eli ICD-9-tyyppisiä diagnoosikoodeja on paljon [9] ja jokaista näytettä kohden on yksi tai useampi luokka. Työ on siten 2.6 -luvun mukaisesti multilabel-ongelma. Scikit-learn-kirjasto sisälsi tarvittavat työkalut tämän kaltaisen ongelman ratkaisemiseen. Se tarjosi suuren määrän erilaisia luokittelija-algoritmeja sekä datan esikäsittelyyn vaadittavaa toiminnallisuutta. Tämän kirjaston lisäksi työssä hyödynettiin monia muita Python-kielen kirjastoja auttamaan datan käsittelyssä ja tulosten visualisoinnissa.

Työssä päädyttiin vertailemaan seitsemää eri luokittelijatoteutusta *multilabel*-näkökulmasta, jolloin käytettiin Scikit-learn-kirjaston *OneVsRestClassifier*-toteutusta yhdessä perusluokittelijoiden kanssa muodostamaan luokittelijoita jokaiselle mahdolliselle luokalle ja yhdistämään niiden tulos. Toinen vaihtoehto, jota tutkittiin oli erottaa tulosluokat toisistaan, jolloin jokainen näyte monine diagnooseineen jaettiin diagnoosien mukaan moneksi samakaltaiseksi näytteeksi. Tämän seurauksena yhtä näytettä vastasi aina yksi diagnoosi ja sama näyte esiintyi eri luokittelijoilla erilaisen tulosluokan kanssa. Näin ongelma saatiin *multiclass*-tyyppiseksi ja se oli helpommin lähestyttävissä. Kaikille 150:lle muodostetulle luokalle rakennettiin erikseen omat luokittelijakandidaattinsa, joita optimoitiin sekä arvioitiin. Nämä luokittelijat eivät tarvinneet *OneVsRestClassifier*-lisärakennetta.

Kun kahta eri toteutustapaa vertailtiin huomattiin, että ensimmäinen ratkaisu toimi paremmin, sillä se oli huomattavasti tarkempi. Kumpikin ratkaisuvaihtoehto oli f1-arvoon perustuvaan tarkkuusarvioltaan suunnilleen samaa luokkaa; noin $0.37 - 0.46$. Kummassakin olisi siis vielä paljon potentiaalia nostaa tarkkuutta.

Seuraavissa aliluvuissa käsitellään tarkemmin työssä käytettyjä työkaluja sekä luokittelijoiden ja niiden esikäsittelyketjujen toteuttamista eri ratkaisuvaihtoehdoille. Lopuksi tutkitaan vertailun tuloksia ja arvioidaan, mitä haasteita työn toteuttamisessa oli sekä, miten työtä pystysi kehittämään tulevaisuudessa.

4.1 Käytetyt työkalut

Koneoppimisen työkaluja on mahdollista käyttää monen erilaisen kehitysympäristön kautta. Suosituimmat ympäristöt ovat R-kielen työkalut sekä Python-kielen scikit-learn-kirjasto. Myös MatLab-tuotteen työkaluja käytetään laajasti, mutta R ja Python-pohjaiset kehitysympäristöt ovat avoimen lähdekoodinsa ja edullisuutensa takia kasvattaneet suosiota. Pythonia hyödynnetään tieteellisen laskemisen lisäksi kattavasti muissakin käyttötarkoituksissa toisin kuin R, joka on käytössä lähinnä tilastotieteen piirissä [3].

Python ja scikit-learn valittiin päätyökaluiksi tutkimuksen vaatiman koneoppimistoteutuksen kehittämiseksi ja arvioimiseksi avoimuutensa ja siihen omatun vankan osaamis- ja kokemuspohjansa takia. Python on kattava ja monikäyttöinen kieli, johon on kehitetty vapaaehtoisvoimin paljon erilaisia työkaluja ja kirjastoja. Näistä eniten kiinnostusta herättää työn kannalta SciPy-kirjastoperhe, jota on kehitetty tieteellistä laskemista varten. Se sisältää niin simulaatio-, hahmontunnistus- tilastotiede- ja koneoppimisen työkaluja erilaisissa aihekokonaisuuksittain järjestetyissä lapsikirjastoissa. Yksi näistä on scikit-learn, joka sisältää työssä käytetyt koneoppimisen algoritmit, datan esikäsittelyn, luokittelijoiden koostamiseen ja niiden tuottamien luokittelutulosten arviointiin sekä vertailuun.

Muita tärkeitä työkaluja olivat Pythonin kirjastot; Nltk, Numpy, Pandas ja Matplotlib, jotka tukivat kirjoitetun kielen käsittelyä, datan lataamista tiedostoista, matriisioperaatioita ja tulosten visualisointia. Seuraavissa aliluvuissa käsitellään tarkemmin Scikit-learn-ohjelmistokirjaston tarjoamia työkaluja sekä miten sitä ja aputyökaluja käytettiin työssä.

4.1.1 Scikit-learn

On olemassa useita Python-kirjastoja, jotka tarjoavat runsaasti koneoppimisalgoritmeja. Yksi tunnetuimmista on Scikit-Learn [23], joka tarjoaa kattavan joukon erilaisia yleisesti käytössä olevia koneoppimisen työkaluja. Scikit-learnille on ominaista puhdas, yhtenäinen ja virtaviivainen sovellusrajapinta sekä erittäin kattava ja helppolukuinen dokumentaatio. Tämän yhdenmukaisuuden etuna on, että kun ymmärrtää Scikit-Learnin peruskäytön ja syntaksin yhdelle mallityypille, siirtyminen uuteen malliin tai algoritmiin on hyvin yksinkertaista.

Scikit-learn on avoimen lähdekoodin projekti, joka on rakennettu SciPyn päälle. Se on julkaistu BSD-lisenssillä, joka asettaa vain minimaaliset rajoitukset katetun ohjelmistokokonaisuuden käyttöön ja uudelleenjakamiseen. David Cournapeau aloitti projektin vuonna 2007 Google Summer of Code -projektina, ja siitä lähtien monet vapaaehtoiset ovat osallistuneet sen kehittämiseen. Sitä ylläpitää tällä hetkellä iso joukkoja vapaaehtoisia ohjelmistokehittäjiä.

Se sisältää kattavan valikoiman erilaisia koneoppimiseen liittyviä algoritmeja. Se tarjoaa kattavat työkalut datan esikäsittelyyn, piirteiden tuottamiseen ja luokittelijoiden sekä regressiotyökalujen tuottamiseen. Sen sisältämät arviointityökalut sekä työnkulkua helpottavat algoritmit ovat myös korvaamattomia kehitettäessä ja optimoitaessa koneoppimisen ratkaisuja.

Scikit-learn on yksi kattavimmista työkalukokonaisuuksista koneoppimisen saralla ja se sisältääkin erilaisia toteutuksia kaikista kappaleen 2 luokittelualgotimeista. Tämän lisäksi tärkeimmäksi kokonaisuudeksi scikit-learnistä nousee sen aputyökalut. Niiden avulla voidaan rakentaa olio-ohjelmoinnin avulla monimutkaisia datankäsittelyn työlinjastoja. Näitä kutsutaan kirjaston sisällä *pipline*-nimellä ja ne mahdollistavat näytteiden, koostamisen, esikäsittelyn, luokittelun sekä luokittelijoiden arvionnin sekä optimoinnin saman rakenteen avulla. Tämä nopeuttaa ja helpottaa huomattavasti toteutuksen kehittämistä, koska eri datan prossesointivaiheita voidaan helposti lisätä tai poistaa isommasta prosessointiketjusta tarpeen mukaan. Se auttaa rakentamaan kunnollisen hierarkian koko tarvittavalle kokonaisuudelle.

4.1.2 Muut työkalut

Yksi tärkeä työssä käytetty työkalu oli Pythonin Nltk-kirjasto (*Natural Language ToolKit*) [16]. Nltk tarjoaa kattavan määrän eilaisia työkaluja tekstidokumenttien luokitteluun, semanttiseen analysointiin, parsimiseen ja monenlaiseen muuhun käsittelyyn. Se on laajasti käytössä koneoppimisen yhteydessä, kun käsiteltävä aineisto on tekstipohjaista. Tässä työssä sitä käytettiin TF-IDF-piirteiden tuottamiseen sanojen ja termien erotteluvaiheessa

Numpy on avoimen lähdekoodin tieteelliseen laskemiseen erikoistunut Python-kirjasto [22]. Numpy on osa SciPy-kirjastoperhettä ja se on varsinkin matriisitietorakenteidensa takia paljon käytetty työkalu. Se on hyvin laajasti käytössä ja monet SciPy-perheen kirjastot käyttävätkin sitä sisäisesti. Se on myös erittäin helppokäyttöinen ja nopea, joten sitä pystyy myös vaivatta käyttämään, kun luo omaa toteutustaan. Numpy-kirjaston käyttäminen on melkein väistämätöntä kun käytetään jotain SciPy-kirjastoa ja Scikit-learn ei ollut poikkeus. Se oli matriisityökalujensa takia korvaamaton väline datan esikäsittelyssä.

Pandas on toinen tieteelliseen laskentaan erikoistunut Python-ohjelmakirjasto [19]. Myös Pandas on osa SciPy-kirjastoperhettä ja se on rakennettu matalamman tason työkalun, Numpyn, pohjalta. Sillä toteutettiin työssä aineiston lataaminen csv-tiedostoista. Pandaksen avulla lataaminen pystyttiin tekemään muistiystävällisemmin vaiheissa. Pandas tarjo-

aa pitkälti samankaltaisia työkaluja kuin Numpykin sillä eroavaisuudella, että se tarjoaa edistyneempiä datan analysointityökaluja.

Neljäs varteenotettava työssä käytetty työkalu oli Matplotlib, joka toimi työn visualisointityökaluna [10]. Sekin kuuluu mainioon SciPy-kirjastoperheeseen. Sillä pystyttiin visualisoimaan itse työn kannalta oleellista analysointityötä ja muodostamaan ratkaisujen tarkkuuksien vertailujen kuvaajat.

4.2 Opetusdatan tuottaminen

Opetusdatan tuottaminen on erittäin tärkeä kysymys koneoppimisen saralla ja siihen täytyy kiinnittää suurta huomiota. Se voi vaikuttaa koko koneoppivan järjestelmän käytön kannattavuuteen. Sopivasti valittu opetusdata luo pohjan koko muulle koneoppimisen prosessointiketjulle. Työssä keskityttiin tuottamaan mahdollisimman laadukasta piirredataa luokittelijoita varten mahdollisimman yksinkertaisilla tavoilla. Näin päädyttiin käyttämään potilaiden hoitojaksojen loppuraportteja kaikkien potilaan saamien hoitomerkintöjen sijaan.

Luokittelijoita varten tuotettiin kahdenlaisia piirteitä. Normaalien tekstistä löytyvien sanojen lisäksi niistä muodostettiin pelkistettyjä versioita nltk-kirjaston stemmaustyökalulla. Näistä tuotetuista sanoista kehitettiin sen jälkeen n-gram-mallin mukaisesti 1 - 3 sanaa pitkiä termiketjuja. Näille termiketjuille pystyttiin sitten laskemaan TF-IDF-arvot. Tällä tavalla tuotetut piirteet sisälsivät kuitenkin paljon merkityksettömiä sanajonoja, joten niistä täytyi vielä karsia suurin osa pois ennen kuin niitä voidaan syöttää luokittelijalle.

4.2.1 Näytteiden valinta

Tässä työssä oli tärkeää saada potilaasta mahdollisimman kattavasti tietoja nykyisistä oireista ja mahdollisesta sairaushistoriasta. Poilaiden tiedot koostuivat sekalaisesti kokoelmasta potilaan hoidosta ja tilasta kertovista hoitomuistiinpanoista. Niitä olisi ollut hankala liittää yhteen sopivasti, koska osassa niistä paljastettiin potilaan saama diagnoosi ennen kuin hoitojakso oli valmis. Tavoitteena ei ole, että koneoppiva järjestelmä pystyisi poimimaan halutun diagnoosin suoraan tekstistä. Paras keino välttää tämä onkin käyttää kaikkien hoitomerkintöjen sijasta potilaan hoidon loppuraporttia. Siinä on käyty läpi kaikki potilaalle annettu hoito hänen sairaalakäyntinsä aikana. Sen lopussa on myös mainittu hänelle annetut diagnoosit ja uudet lääkkeet tai olemassaolevan lääkityksen muutokset. Mikä meitä todella kiinnostaa on raportin alkuosa, joka kokoaa yhteen tiivistetyssä muodossa potilaan sairaushistorian, elintavat sekä nykyiset oireet, jonka takia hän on haketunut hoitoon.

Potilaan loppuraportin rakenne on yksinkertainen ja yhdenmukainen, joten siitä on helppo valita vain haluttu alkuosa. Kun näiden raporttien alkuosat eristetään, saadaan vaivattomasti kattavat kuvaukset potilaiden hoitojaksoista, joita voimme käyttää koneopppimsessa näytedatana. Näytteet tarvitsevat tekstimuotoisten hoitomerkintöjen lisäksi niitä

vastaavien hoitojaksojen diagnoosit. Nämä löytyvät myös raportista, mutta ne on kirjoitettu yleensä vapaamuotoisesti ja niissä on paljon lääkärien käyttämiä lyhenteitä. Yhdenmukaiset ja kattavat diagnoositiedot löytyvät DIAGNOSES_ICD-taulusta, jossa jokainen hoitojakso ja niihin liityvät diagnoosit on listattu järjestyksessä. Kun meillä on molemmat, hoitomerkinnät ja diagnoosit, raakaversio näytedatasta on valmis. Saeuraavaksi sille täytyy tehdä esikäsittely- ja normalisointitoimenpiteitä, jotta siitä saadaan johdettua mahdollisimman hyvät piirteet. Näistä toimenpiteistä on kerrottu tarkemmin aikaisemmassa 2 -kappaleessa tarkemmin. Seuraavassa aliluvussa käydään tarkemmin läpi juuri ne esikäsittelytoimepiteet, joita työssä käytettiin.

4.2.2 Näytedatan esikäsittely

Näytedata on syytä esikäsitellä hyvin, jotta siitä johdettavat piirteet ovat mahdollisimman laadukkaita. Tekstipohjaisesta materiaalia esikäsitellessä on syytä huomioida, että mitään tarpeettomia merkkejä ei säilytetä. Työssä käytetyssä tekstidatassa oli paljon aikasemmasta tiedon prosessoinnista jäljelle jääneitä korvaustermejä. Tämä prosessointi on luultavasti koskenut tiedon anonymisointia, sillä niillä on korvattu erisnimiä, jotka viittavaat henkilöihin tai paikkoihin, kuten sairaaloihin. Niiden tilalla on alkuperäisessä datassa olleista erisnimistä generoituja tunnisteita. Esimerkkinä tämän kaltaisesta tunnisteesta on "[**Hospital 6**]". Nämä ovat täysin tarpeettomia luokittelun kannalta, joten ne poistetaan.

Muuta poistettavaa dataa ovat turhat välimerkit. Tästä johtuen tekstistä poistetaan mm. pisteet, pilkut, ajatusviivat sekä muut tämän kaltaiset merkit. Myös suuraakkoset ja pienaakkoset tulisi yhdenmukaistaa, joten kaikki merkit muutetaan pienaakkosiksi. Tämänkaltaiset operaatiot tehdään tekstidatalle termien parsimisvaiheessa. Parsimisvaihe on toteutettu osittain itse, mutta se tukeutuu suurimmaksi osaksi NLTK-kirjaston *word_tokenize*-nimiseen termin parsintafunktioon. Toteutuksessa dokumenteista karsitaan välimerkit Pythonin oman merkkijonojen käsittelyoperaatioiden avulla ja sitten siitä erotellaan kaikki sanat edellä mainitun *word_tokenize*-funktion avulla.

Viimeinen operaatio datan esikäsittelyssä on turhien sanojen poistaminen. Englannin kieli on täynnä artikkeleita, prepositioita ja pronomineja, jotka eivät tuo mitään lisätietoa tekstiin koneoppimisen kannalta. Sekä NLTK- että Scikit-learn-kirjasto sisältävät kattavan listan tämänkaltaisia sanoja, joista käytetään nimitystä pysäytyssanat *stop-words*. Toteutuksessa päädyttiin käyttämään Scikit-learnin pysäytyssanalistaa. Kun listan määrittämät sanat on poistettu, data on varsin hyvälaatuista seuraavaa vaihetta, piirteiden muodostamista, varten.

4.2.3 Piirteiden tuottaminen

Esikäsitellystä datasta olisi tarkoitus tuottaa mahdollisimman kuvaavat piirteet käsiteltävää ongelmaa varten. Tällaisia piirteitä tekstidatasta voisivat olla tiettyjen termien esiintyminen näytteessä. Esimerkiksi termi "unresponsive"voi olla erittäin tärkeä potilaan tilaa

kuvaava termi. Kun näitä yhdistetään muihin kuvaaviin termeihin saadaan hyvin tietoa potilaan tilasta. Ongelmana tässä on se, että sama käsite voi esiintyä monin eri ilmauksin tekstinäytteessä. Sen takia olisi hyvä huomioida erilaiset esitysmuodot ja yhdistää ne yhden termin alle. Esimerkkinä "unconscious", "comatose", "catatonic"ja "coma". Tämä on työssä huomioitu käyttämällä nltk-kirjaston stemmaustyökalua.

Stemmaus on sanojen muuntamista niiden perusmuotoon. Esimerkiksi sanat "comatose", "coma"muunnettaisiin termiksi "coma". Tämä menetelmä ei ota kuitenkaan huomioon täysin toisenlaisia kirjoitusasuja kuten "unconcious"ja "coma". Näiden termien yhteneväisyyttä on erittäin hankala havaita. Tähän ongelmaan on olemassa jotain työkaluja kuten aikaisemmin mainittu DNorm, joka on erikoistunut tautien nimien normalisointiin. Koska kaikkia lääketieteen termejä normalisoivaa työkalua ei löydetty, työssä päädyttiin käyttämään vain nltk-kirjaston stemmausta termien normalisoinnissa.

Työssä päädyttiin käyttämään stemmattujen sanojen lisäksi myös stemmaamattomia sanoja, jolloin piirrevektori muodostettiin tekstistä suoraan poimittujen sanojen ja niistä stemmattujen perusmuotojen yhdistelmällä. Tämä tehtiin siksi, koska stemmattujen sanojen käyttö yhdessä alkuperäisten sanojen kanssa vaikutti positiivisesti lopullisten luokittelijoiden arvostelussa. Nähtävästi stemmauksessa menetetään jotain tärkeää informaatiota luokittelun kannalta siinä, missä se poistaa luokittelua vaikeuttavaa häiriötä. Sanojen stemmaus tehtiin jo esikäsittelyä tekevässä parsimisvaiheessa. Samalla kun dokumenteista eroteltiin sanoja, näistä samoista sanoista muodostettiin myös niiden perusmuotoja, jolloin parsimista tekevä funktio palautti listan alkuperäisistä sanoista, jonka loppuun oli lisätty myös samojen sanojen perusmuodot.

Kun parsintavaihe oli tuottanut näytedokumenteista sanalistoja, niitä voitiin käyttää muodostamaan n-gram sanaketjuja. Tämän tarkoituksena oli rakentaa piirteitä, joissa olisi myös korkeamman abstraktiotason tietoa. Esimerkikisi sanajonojen "koira ei juokse"ja "koira juoksee"merkitys on täysin eri kuin yksittäisten sanojen "koira", "ei", "juokse"ja "juoksee". Sanojen väliset riippuvuudet saadaan paremmin selville, kun muodostetaan piirteitä myös peräkkäisten sanojen yhdistelmistä sen sijaan, että käytettäisiin pelkästään yksittäisiä sanoja. Tähän käytettiin työssä Scikit-learn-kirjaston *TfidfVectorizer*-rakenteessa olevaa sisäistä ngram-sanajonojen generointia. Sille syötettiin kokonaislukuväli, joka määritteli kuinka pitkiä sanajonoja se tuottaisi. Työn kannalta sopiva arvo oli väli $(1,3)$, jolla muodostettiin yhden, kahden ja kolmen peräkkäisen sanan mittaiset sanajonot.

Sanajonoista pystyttiin seuraavaksi laskemaan helposti varsinaisia piirteiden arvoja, kuten esimerkiksi esiintymislukumääriä. Sanojen esiintymislukumääriä kehittyneempi piirre on 2 -kappaleessakin kuvattu sanojen esiintymistiheyksiä ja dokumenttitiheyksiä käyttävä TF-IDF-arvo. TF-IDF (*term frequency - inverse document frequency*), ottaa huomioon sen, jos jokin termi on yleinen kaikissa dokumenteissa ja osaa vähentää tällaisten termien merkitystä. Dokumenteissa harvakseltaan esiintyvä termi saa taas toisaalta enemmän painoarvoa TF-IDF-arvoja laskettaessa. TF-IDF-arvojen laskeminen oli seuraava osa työn näytedatan esikäsittelyketjua ja siihen käytettiin edellä maninittua *Tfidf-*

Vectorizer-luokkaa. Tämä työkalu sisälsi paljon muuta toiminnallisuutta pelkän TF-IDF-arvojen muodostamisen lisäksi. Kuten aiemmin todettiin sillä pystyttiin myös muodostamaan ngram-sanaketjuja. Tämän lisäksi sille pystyttii määrittelemään piirteiden valintaa tekeviä raja-arvoja. Näillä raja-arvoilla pystyttiin generoitujen TF-IDF-arvojen suhteen rajamaan piirteistä pois pieniä tai suuria arvoja. Työssä käytettiin raja-arvoja $max_{df} = 0.9$ ja $min_{df} = 0.001$, joiden avulla lopullisiin piirrejoukkoon valikoitui vain sellaiset piirteet, jotka olivat välillä $(0.001, 0.9)$ df-arvojensa suhteen. Se karsi piirteistä todella harvinaiset ja todella yleiset sanat sekä sanajonot. *TfidfVectorizer*-rakennetta käyttämällä teksti saatiin myös muutettua käyttämään pelkästään pienaakkosia, mutta tämä tehtiin jo aikaisemmin NLTK-kirjaston *word_tokenize*-funktion avulla.

Lopuksi piirteistä arvioitiin ja valittiin sellainen osajoukko, jonka avulla luokittelu tehtäisiin mahdollismman tarkasti. Tähän käytettiin Scikit-learn-kirjaston tarjoamaa *SelectKBest*-valitsijaa. Se on yksinkertainen piirteiden valintaa tekevä algoritmi, joka arvioi näytteiden piirteitä niitä vastaavia luokkatietoja vasten ja pyrkii valitsemaan parhaan mahdollisen piirrekombinaation näiden tietojen perusteella. Se perustuu tilastotieteen keinoihin piirteiden soveltuvuuden arvioimiseen ja se valitsee k parasta piirrettä, jotka on selvitetty sisäisen arviointifunktion avulla. *SelectKBest*:lle pitää määrittää kaksi parametria; kokonaislukuarvo k, joka määrittää kuinka monta piirrettä alkuperäisestä piirrejoukosta valitaan sekä arviointifunktio, jotka käytetään piirteiden paremmuusjärjestyksen selvittämiseen. *SelectKBest* on yksi yksinkertaisimmista piirteidenvalinta-algoritmeista, joka on sisällytetty Scikit-learn-kirjastoon.

4.2.4 ICD-9-luokkatietojen esikäsittely

Näytedatan lisäksi luokkatiedot kerättiin PhysioNet:stä saaduista csv-tiedostoista. DIAGNOSES_ICD.csv sisälsi tiedot hoitojaksojen lopullisista diagnooseista. Yhdelle hoitojaksolle oli määritelty useita diagnooseja sisältäen järjestysnumeron, joka kuvasi diagnoosin tärkeyttä kyseiselle hoitokerralle. Järjestyksessä ensimmäinen diagnoosi oli aina niin kutsuttu ensisijainen diagnoosi ja loput olivat toissijaisia diagnooseja. Työssä päädyttiin käyttämään kaikkia hoitojaksolle annettuja diagnooseja ja loppullisten luokittelijoiden tuli ennustaa nämä kaikki samat diagnoosit yhdelle hoitojaksolle. Tämä teki luokittelusta *multi-label*-ongelman. ICD-9 -typpisissä diagnoosikoodeilla voidaan kuvata yli 13 600 erilaista diagnoosia [9]. Tämä on todella suuri määrä luokittelijalle, joten työssä päädyttiin karsimaan mahdollisten luokkien määrää. Koska ICD-9 on hierarkinen koodisto, joka koostuu kahdesta eri osasta; pääluokasta ja aliluokasta, voidaan yksittäinen koodi yksinkertaistaa pääluokkamuotoonsa. Tämä tehtiin yksinkertaisesti katkaisemalla koodin merkkijono luokkia erottavan pisteen kohdalta ja käyttämällä näin saatua koodin alkuosaa näytteen luokka-arvoina.

Tämän lisäksi diagnooseista haluttiin karsia pienen edustuksen omaavat diagnoosiluokat pois. Jos mahdolliset luokat ovat esiintymismääriltään paljon epätasapainossa, kehitetyistä luokittelijoista ei saada kovinkaan tarkkoja. Tämän takia ei ole mielekästä sisällyttää

harvinaisempia luokkia mukaan luokittelijoiden rakennukseen. Yksinkertaisella diagnoosien lukumäärien analysoinnilla ja suodattamisella luokkatiedoista poistettiin kaikki diagnoosikoodit, jotka esiintyivät vähemmässä kuin 500 eri näytteessä. Tämän seurauksena osassa näytteistä ei ollut lainkaan diagnoosikoodeja, joten ne täytyi poistaa lopullisesta näytejoukosta. Kun nämä operaatiot oli tehty mahdolliset luokat karsiutuivat 166 eri pääluokkaan. Tämä joukko sisälsi kuitenkin vielä 16 V-alkuisia ICD-9-koodeja, joiden luokittelu on erittäin hankalaa. Nämä koodit olivat terveydentilaan ja terveydenhoitoon liittyviä täydentäviä luokkia, joiden esiintymistä hoitojakson lopullisissa diagnooseissa oli vaikea ennustaa, koska ne olivat vain täsmentäviä apuluokkia. Ne päätettiin karsia myös pois, jotta varsinaisten diagnoosien luokittelun tarkkuuten pystyttiin keskittymään paremmin. Lopulta näiden operaatioiden jälkeen mahdollisia luokkia oli enää 150, jotka koostuivat pelkästään diagnoosien pääluokista.

4.3 Muut toteutusyksityiskohdat

Tiettyjä asioita pitää ottaa huomioon piirrevektoreita muodostettaessa. On esimerkiksi mahdollista, että samalla potilaalla on useita hoitojaksoja samasta vaivasta ja niitä on kuvattu hyvin samankaltaisilla tekstimerkinnöillä jokaisessa hoitojakson loppuraportissa. Kun näitä hoitojaksoja sattuu päätymään sekä opetus- että testidataan, ei voida olla varmoja luokittelijan objektiivisesta suorituskyvystä ja tarkkuudesta. Tätä ratkaisemaan käytettiin Scikit-learn-kirjaston *GroupKFold*-algoritmia, joka jakoi koostettuja näyte- ja luokkatietoja opetus- ja testiryhmiin. Sille pystyttiin määrittämään ryhmätieto, joka merkitsi samaan ryhmään kuuluvat näytteet. Tässä tapauksessa ryhmätiedot määräytyivät potilaiden mukaan, jolloin yhden potilaan hoitojaksot olivat samassa ryhmässä. Näiden ryhmätietojen avulla se pystyi rakentamaan sellaiset opetus- ja testijaot, jossa yksi potilas esiintyi vain toisella puolella jakoa.

Toinen tärkeä asia, mikä tulisi huomioida on lähdeaineiston näytteiden mahdollinen järjestys. Hoitomerkinnät saattavat olla järjestetty jonkin taudin mukaan tarjotuissa tiedostoissa. Kun opetus- ja testausdata jaetaan tiedostoissa esiintymisjärjestyksen mukaan saadaan epätasapainoinen jako. Tällä datajaolla harjoitetuista luokittelijoista tulee epätarkkoja luokille, mitä se ei löytänyt opetusdatasta. Lisäksi luokittelijoita ei testata kattavasti, jos joitakin luokkia ei löydy testidatasta. Koska lähdeaineiston kaikkia hoitojaksoja ei käytetty luokittelijoiden rakentamisessa, pitää varmistua siitä, että rakentamiseen käytetyt näytteet ovat tilastollisesti samankaltaisia testaus- ja opetusdatan välillä. Tähän käytettiin Scikit-learn-kirjaston työkalua *shuffle*, jonka avulla näytteitä pystyttiin sekoittamaan ennen kuin niistä valittiin opetus- ja testijoukkojaot.

4.4 Luokittelijoiden vertailu

Luokittelijoiden opettamisessa päädyttiin vertailemaan kahta erilaista ratkaisumahdollisuutta diagnoosikoodien ennustamiseksi. Ensimmäisessä vaihtoehdossa luokittelu tehtäisiin *OneVsRestClassifier*-kääreiden avulla, jolloin sillä ratkaistaan näytteiden *multilabel-*

ongelma. Toisessa vaihtoehdossa näytteiden eri tulosluokat erotellaan siten, että jokaista näytettä vastaa vain yksi luokkatieto. Tämän avulla *multilabel*-ongelmaa voidaan kiertää muuttamalla se yleisemmäksi *multiclass*-oneglmaksi. Lisää näiden käsitteiden erosta voi lukea luvusta 2.6.

Molemmissa vertailun ratkaisuvaihtoehdossa opetettiin ja harjoitettiin seitsemää erityyppistä luokittelija-algoritmia. Kaikki näistä olivat peräisin Scikit-learn-kirjastosta ja ne kaikki käyttivät aikaiksemmassa luvussa kuvattua esiprosessointiketjua ja sen avulla kehitettyjä piirteitä. Pieniä eroavaisuuksia oli siinä, kuinka monta piirrettä luokittelijakandidaatille valittiin *SelectKBest*-algoritmin toimesta. Nämä arvot valittiin eri luokittelijoille optimointivaiheessa ja käytettyjen piirteiden määrä vaihteli kuuden ja kymmenen tuhannen välillä *OneVsRestClassifier*-rakennetta käytettäessä ja yksittäisten luokkien luokittelijoille 50 ja 500 välillä.

Luokittelijoiden algoritmit Scikit-learn-kirjastossa tukivat hyvin suoraan moniluokkaisuutta, mutta monet niistä eivät tukeneet sitä, jos yhtä näytettä kohden oli useampi ennustettava luokka. Ne eivät olleet niin kutsuttaja *multilabel*-luokittelijoita suoraan. Onneksi kirasto tarjosi tähän työkalun kääreen muodossa. *OneVsRestClassifier* sovittaa minkä tahansa luokittelijan tukemaan *multilabel*-ongelmakenttää. Se toimii kääreen tavoin eli sille syötetään parametrina käytettävä luokittelija-algoritmi, jolloin siitä muodostetaan monta luokittelijaa, jotka pyrkivät kaikki erottelemaan yhden tietyn luokan muista luokista. *OneVsRestClassifier* on käytettävän rajapintansa suhteen täsmälleen samanlainen, kuin muutkin luokittelijat. Tätä rakennetta käytettiin kaikkien muiden algoritmien paitsi päätöspuiden kanssa.

Luokittelijoiden optimointia ja vertailua varten rakennettiin Scikit-learn-kirjaston tarjoama *Pipeline*-rakenne, joka helpotti huomattavasti erilaisten parametrikombinaatioiden määrittelyä sekä tulosten keräämistä eri luokittelijavaihtoehdoille. *Pipeline* mahdollistaa eri datankäsittelyoperaatioiden yhdistämistä saman rakenteen alle. Siihen voi esimerkiksi liittää osaksi piirteiden muodostuksen, piirteiden valinnan ja luokittelijoiden opettamisen. Tällöin data käsitellään prosessointiketjuna *Pipeline*-rakenteen määrittäämän suoritusjärjestyksen mukaan. Tämän mahdollistaa Scikit-learn-kirjaston datankäsittelyalgoritmien rakenteellinen ja käyttötapojen yhteneväisyys. Kaikki sopivat algoritmit sisältävät *fit_transform*-nimisen funktion, joka muuntaa sille syötetyn datan kyseisen työkalun määrittämällä tavalla.

Kun *Pipeline* yhdistetään *cross validation*-algoritmeihin saadaan tehokas optimointi- ja vertailujärjestelmä luokittelijoille. Yksi parhaista tähän tarkoitukseen rakennetuista työkaluista on *GridSearchCV*, jolle määritellään luokittelija sekä mahdolliset testattavat parametrit. *GridSearchCV* käy systemaattisesti läpi sille annetut parametrit, koostaa niiden avulla luokittelijan, jakaa sille annetun piirredatan erilaisiin opetus- ja testijoukkokombinaatioihin ja vertailee näin koottuja luokittelijoita. Se on erittäin tehokas rakenne optimoinnin ja luokittelijoiden vertailun avuksi, koska parametreina voi antaa myös eri luokittelijoita. Sille voi helposti määritellä työssä tehdyllä tavalla useita eri luokittelijoita,

joista jokaisella on erilaiset parametrit ja koota näistä vertailtavat luokittelijat. Kun *GridSearchCV*-algoritmia käytetään yhdessä *Pipeline*:n kanssa voidaan määritellä vertailtavia parametrijoukkoja koko datankäsittelyketjulle. Näin voidaan esimerkiksi helposti vertailla millaisia vaikutuksia on piirteitä tuottavan algoritmin toimannan muuttamisella tai datan esikäsittelyyn tehtävillä korjauksilla. *GridSearchCV* palauttaa suorituksensa jälkeen rakenteen, josta pystyy helposti vertailemaan eri parametrikombinaatioiden tuloksia toisiinsa nähden niin tarkkuuden kuin suoritusajankin osalta.

Työn ensimmäisessä ratkaisussa käytettiin seuraavia luokittelija-algoritmeja optimoiduilla parametreilla:

- OneVsRestClassifier(SGDClassifier(penalty='elasticnet', max_iter=1000, class_weight='balanced', alpha=0.000085, tol=0.00001)), 11000 piirteellä
- RandomForestClassifier(class_weight='balanced_subsample', max_depth=8, max_features=0.8, n_estimators=100), 6000 piirteellä
- ExtraTreesClassifier(class_weight='balanced', max_depth=5, max_features=0.0016, n_estimators=1000), 6000 piirteellä
- OneVsRestClassifier(KNeighborsClassifier(n_neighbors=3, leaf_size=5, weights='distance')), 6000 piirteellä
- OneVsRestClassifier(MultinomialNB(alpha=0.0046)) 6000 piirteellä
- OneVsRestClassifier(BernoulliNB(alpha=0.0171, binarize=0.0198)), 10000 piirteellä
- OneVsRestClassifier(LinearSVC(C=25, tol=0.0005)), 11000 piirteellä

Toisessa ratkaisussa taas käytettiin luokittelijakandidaatteina seuraavia vaihtoehtoja:

- SGDClassifier(penalty='elasticnet', max_iter=1000, class_weight='balanced', alpha=0.000038, tol=0.0000055), 50 piirteellä
- RandomForestClassifier(class_weight='balanced', max_depth=13, max_features=0.0077, n_estimators=1500), 430 piirteellä
- ExtraTreesClassifier(class_weight='balanced', max_depth=15, max_features=0.016, n_estimators=1700), 500 piirteellä
- OneVsRestClassifier(KNeighborsClassifier(n_neighbors=3, leaf_size=5, weights='distance')), 50 piirteellä
- OneVsRestClassifier(MultinomialNB(alpha=0.1)), 50 piirteellä
- OneVsRestClassifier(BernoulliNB(alpha=0.1, binarize=0.00168)), 50 piirteellä
- OneVsRestClassifier(LinearSVC(C=5, tol=0.6)), 50 piirteellä

4.4.1 Luokittelijoiden arvosteluperusteet

Luokittelijoita arvioitiin *GridSearchCV*-työkalun avulla. Sitä ajettiin eri parametrikombinaatioilla iteratiivisesti, jolloin pystyttiin optimoimaan parametreja haarukoimalla niitä vähitellen paremmiksi. Luokittelijoiden arviointifunktio määriteltiin *GridSearchCV*-työkalulle, jolloin se käytti tätä tiettyä arvoa järjestäessään eri luokittelija- ja parametriyhdistelmät tarkkuusjärjestykseen. Työssä käytettiin f1-arvoja kehitettyjen mallien tarkkuuden arviointiin.

F1 osoittautui hyväksi arviointiperusteeksi, sillä se otti hyvin huomioon luokkien epätasapainosta johtuvat luokitteluvirheet. Diagnoosit olivat hyvin epätasa-arvoisia esiintymissuhteissaan. Varsinkin verisuonitaudit olivat varsin runsaslukuisia muihin tautiryhmiin verrattuna parsitussa aineistossa. Kuten luvussa 2.7.2 käsiteltiin, f1-arvo koostetaan täsmällisyyden ja löytökyvyn avulla.

Työssä käytetty f1-arvo oli hieman muunneltu versio normaalista f1-arvosta. Siinä käytetty versio toimi paremmin moniluokkaiselle ongelmalle, sillä siinä laskettiin eri luokkien saamien f1-arvojen painotettu keskiarvo. Scikit-learn tarjosi funktion *f1_score*, jolla tämä pystyttiin laskemaan antamalla sille parametri $average =' weighted'$. Yksittäisen luokan painotus keskiarvon laskemisessa määräytyi sen esiintymismäärän mukaan. Tämä tarkoitti sitä, että vaikka harvinainen luokka saisikin kehnon f1-arvon, se ei vaikuttaisi suuresti laskettuun f1-keskiarvoon.

4.4.2 Luokittelijoiden tulokset

Erilaisia arvioituja luokittelijoita oli seitsemän. Ne vaihtelivat paljon toteutukseltaan ja niissä oli edustajia jokaisesta kappaleen 2 oppimisalgorimista. Arvostelujoukko sisälsi kaksi naiivia Bayes -luokittelijaa, kaksi päätöspuuta, lineaarisen tukivektorikoneluokittelijan, SGD-tyyppisen luokittelijan ja lähimmän naapurin luokittelijan. Kaikki näistä luokittelualgoritmeista oli peräisin Scikit-learn-kirjastosta, jossa olisi ollut erilaisia luokittelijoita vielä monta lisää. Näihin edellä mainittuihin luokittelijoihin päädyttiin kuitenkin, sillä ne olivat tutkimusta ennen entuudestaan tuttuja ja siten helppokäyttöisiä. Näitä luokittelijoita harjoitettiin ja arvioitiin 20 000 hoitomerkinnän avulla käyttäen *GridSearchCV*:tä ja *f1_score*

Ensimmäisen ratkaisun mukaisen toteutuksen tarkkuusarviot on listattu taulukossa 1.

Kuten huomataan parhaimpaan tulokseen ylsi SGD-luokittelija, joka oli f1-arvoltaan melkein 10 prosenttiyksikköä tarkempi kuin seuraavaksi tarkin luokittelija lineaarinen tukivektorikone.

Luokkakohtaiset arviot on esitetty taulukoissa 2 ja 3.

Luokkakohtaisista tarkkuusarvioista voidaan huomata, että luokkien tarkkuuksissa on suuria eroja. Alimmillaan f1-arvo (0.07) oli luokalla, jonka ICD-9-koodi oli *782*, joka liittyi

Luokittelijan Scikit-learn-nimi	Järjestys	F1-keskiarvo	Piirteiden määrä	Opetusaika
SGDClassifier	**1**	**0.466**	**11000**	**20.0s**
LinearSVC	2	0.369	6000	46.4s
BernoulliNB	3	0.367	10000	6.2s
MultinomialNB	4	0.247	6000	3.1s
KNeighborsClassifier	5	0.153	6000	3.2s
RandomForestClassifier	6	0.089	6000	442.3s
ExtraTreesClassifier	7	0.089	6000	62.9s

Taulukko 1. *Multilabel-ratkaisun luokkittelijoiden tarkkuusarvioita*

ihokudoksen oireiluun tai epänormaaliin tilaan. Maksimissaan f1-arvo (0.90) oli taas ICD-9 koodilla 765, joka taas kuvasi raskauden lyhytkestoisuuden tai alahaisen syntymäpainon oireita. On helppo ymmärtää, miksi jälkimmäinen luokka on helpommin luokiteltavissa kuin ensimmäinen. Siinä missä raskauden komplikaatioista puhutaan täsmällisesti ja rajoitetuissa tapauksissa, ihon oireilua saattaa esiintyä laajasti, mutta sitä ei diagnosoida usein, mistä kielii luokan *782* alhainen esiintymismäärä (84 tapausta). Ihon oireilu on myös käsitteenä liian laaja, jolloin se sisältää todella suuren määrän ja vaihtelevalla kuvauksella olevia tapauksia. Taulukon 4 mukaan keskiarvo sijoittuu tarkalleen näiden ääriarvojen keskelle arvoon 0.46. Kehitetyllä SGD-luokittelijalla on keskiarvollisesti hyvä löytökyky, mutta kehno täsmällisyyden arvo. Se tarkoittaa, että kyseinen luokittelija tekee paljon *false positive* -ennustuksia datasta.

Seuraavaksi arvioidaan yksittäisiin luokkiin erikoistuneita luokittelijoita ja parhaimpien luokittelijoiden yhdistämistä kakkien diagnoosien arvioimiseksi näytteille. Tässä ratkaisussa käytettiin samoja luokittelija-algoritmeja sillä erotuksella, että niitä ei upotettu *OneVsRestClassifier*-luokittelijan sisään vaan perusluokittelijoita pystyttiin käyttämään suoraan. Taulukoissa 5 ja 6 luetellaan näiden luokittelijoiden tarkkuus luokittain.

ICD-9	prec.	recall	f1	support	**ICD-9**	prec.	recall	f1	support
008	0.12	0.65	0.21	158	**345**	0.28	0.71	0.40	150
038	0.37	0.79	0.50	573	**348**	0.28	0.70	0.40	338
041	0.23	0.66	0.35	446	**349**	0.07	0.55	0.12	82
070	0.41	0.77	0.53	247	**357**	0.23	0.77	0.36	184
112	0.09	0.62	0.15	176	**362**	0.17	0.71	0.28	115
162	0.34	0.85	0.48	117	**365**	0.24	0.83	0.37	77
196	0.13	0.75	0.22	64	**396**	0.12	0.75	0.21	92
197	0.39	0.85	0.53	170	**397**	0.08	0.68	0.14	75
198	0.39	0.84	0.54	176	**401**	0.61	0.68	0.65	2086
238	0.08	0.45	0.13	71	**403**	0.47	0.76	0.58	613
244	0.60	0.77	0.67	536	**410**	0.52	0.74	0.61	536
250	0.73	0.68	0.71	1436	**411**	0.17	0.77	0.28	182
253	0.10	0.40	0.16	75	**413**	0.13	0.78	0.23	111
263	0.08	0.54	0.14	146	**414**	0.72	0.72	0.72	1402
272	0.61	0.71	0.66	1473	**415**	0.21	0.70	0.33	111
274	0.52	0.81	0.64	236	**416**	0.20	0.68	0.31	270
275	0.08	0.68	0.14	136	**423**	0.16	0.59	0.25	76
276	0.50	0.71	0.59	1483	**424**	0.45	0.68	0.54	583
278	0.30	0.77	0.44	272	**425**	0.29	0.68	0.41	223
280	0.12	0.64	0.20	212	**426**	0.11	0.75	0.19	160
284	0.12	0.70	0.21	97	**427**	0.74	0.64	0.69	1717
285	0.42	0.66	0.51	1268	**428**	0.64	0.71	0.67	1346
286	0.10	0.60	0.17	207	**433**	0.20	0.71	0.31	147
287	0.18	0.69	0.29	397	**434**	0.25	0.78	0.37	144
288	0.09	0.63	0.16	175	**437**	0.10	0.53	0.17	72
289	0.07	0.45	0.11	69	**438**	0.09	0.68	0.15	122
291	0.19	0.79	0.30	106	**440**	0.17	0.75	0.28	179
293	0.08	0.45	0.13	148	**441**	0.41	0.79	0.54	188
294	0.33	0.82	0.47	198	**443**	0.12	0.67	0.21	173
295	0.58	0.82	0.68	73	**453**	0.16	0.61	0.26	159
296	0.38	0.80	0.52	125	**456**	0.43	0.85	0.57	101
300	0.17	0.63	0.27	303	**458**	0.14	0.56	0.23	455
303	0.36	0.72	0.48	211	**459**	0.10	0.63	0.18	87
304	0.18	0.67	0.28	66	**482**	0.14	0.61	0.22	214
305	0.27	0.69	0.39	566	**491**	0.25	0.77	0.38	140
327	0.29	0.79	0.42	246	**492**	0.13	0.42	0.20	67
331	0.29	0.75	0.41	148	**493**	0.64	0.70	0.67	359
338	0.08	0.61	0.14	104	**507**	0.20	0.71	0.31	327
342	0.17	0.77	0.28	78	**511**	0.22	0.72	0.34	347

Taulukko 2. *Multilabel-ratkaisun SGD-luokan luokkakohtaisia tarkkuusarvioita 1*

ICD-9	prec.	recall	f1	support	**ICD-9**	prec.	recall	f1	support
512	0.05	0.48	0.09	97	**745**	0.11	0.68	0.18	68
518	0.54	0.67	0.60	1312	**765**	0.81	1.00	0.90	78
519	0.18	0.71	0.29	122	**770**	0.61	0.98	0.76	60
530	0.58	0.75	0.65	756	**774**	0.65	1.00	0.78	62
535	0.12	0.60	0.20	107	**780**	0.23	0.58	0.33	508
536	0.14	0.60	0.23	55	**781**	0.05	0.57	0.10	53
537	0.15	0.72	0.25	78	**782**	0.04	0.35	0.07	84
553	0.14	0.61	0.22	92	**784**	0.09	0.69	0.16	148
557	0.09	0.64	0.15	69	**785**	0.27	0.65	0.39	475
560	0.17	0.62	0.26	189	**786**	0.09	0.47	0.15	144
562	0.24	0.74	0.37	137	**787**	0.11	0.64	0.18	255
564	0.07	0.56	0.12	128	**788**	0.09	0.46	0.15	213
567	0.19	0.68	0.29	117	**789**	0.25	0.71	0.37	202
568	0.09	0.65	0.16	71	**790**	0.14	0.51	0.22	340
569	0.14	0.71	0.23	116	**799**	0.09	0.52	0.15	164
571	0.57	0.84	0.68	284	**801**	0.26	0.82	0.39	72
572	0.49	0.88	0.63	192	**802**	0.25	0.75	0.38	60
574	0.22	0.71	0.33	116	**805**	0.27	0.89	0.42	103
576	0.31	0.80	0.44	118	**807**	0.32	0.88	0.47	120
577	0.25	0.78	0.38	138	**852**	0.24	0.89	0.38	105
578	0.19	0.67	0.30	245	**860**	0.23	0.90	0.36	67
583	0.08	0.57	0.13	70	**873**	0.22	0.82	0.35	96
584	0.51	0.76	0.61	1101	**995**	0.39	0.77	0.52	554
585	0.48	0.78	0.59	654	**996**	0.30	0.67	0.41	471
593	0.05	0.43	0.09	93	**997**	0.22	0.74	0.33	557
599	0.31	0.64	0.41	755	**998**	0.21	0.64	0.31	443
600	0.11	0.61	0.18	190	**999**	0.05	0.37	0.08	94
682	0.15	0.65	0.24	156	**E849**	0.08	0.53	0.13	200
707	0.27	0.66	0.38	314	**E878**	0.21	0.70	0.32	440
714	0.36	0.63	0.46	70	**E879**	0.11	0.62	0.18	249
715	0.11	0.54	0.19	169	**E885**	0.12	0.68	0.20	71
724	0.09	0.59	0.16	130	**E888**	0.14	0.70	0.23	107
728	0.06	0.52	0.12	79	**E932**	0.07	0.44	0.12	75
729	0.05	0.48	0.09	97	**E933**	0.12	0.75	0.21	59
730	0.09	0.84	0.17	61	**E934**	0.05	0.44	0.08	79
733	0.31	0.72	0.44	266	**E950**	0.28	0.93	0.43	42

Taulukko 3. *Multilabel-ratkaisun SGD-luokittelijan luokkakohtaisia tarkkuusarvioita 2*

	prec.	recall	f1	support
keskiarvo	0.39	0.69	0.46	42260

Taulukko 4. *Multilabel-ratkaisun SGD-luokittelijan keskiarvo*

ICD-9	prec.	recall	f1	support	**ICD-9**	prec.	recall	f1	support
008	0.34	0.30	0.32	158	**345**	0.47	0.62	0.53	150
038	0.00	0.00	0.00	573	**348**	0.22	0.70	0.33	338
041	0.40	0.28	0.33	446	**349**	0.09	0.18	0.12	82
070	0.00	0.00	0.00	247	**357**	0.55	0.52	0.54	184
112	0.12	0.33	0.17	176	**362**	0.42	0.49	0.45	115
162	0.31	0.67	0.43	117	**365**	0.11	0.17	0.14	77
196	0.09	0.58	0.15	64	**396**	0.07	0.64	0.13	92
197	0.42	0.76	0.54	170	**397**	0.08	0.65	0.15	75
198	0.00	0.00	0.00	176	**401**	0.00	0.00	0.00	2086
238	0.11	0.01	0.03	71	**403**	0.00	0.00	0.00	613
244	0.68	0.25	0.36	536	**410**	0.23	0.90	0.36	536
250	0.77	0.53	0.63	1436	**411**	0.13	0.64	0.22	182
253	1.00	0.01	0.03	75	**413**	0.16	0.38	0.22	111
263	0.05	0.23	0.08	146	**414**	0.46	0.88	0.60	1402
272	0.41	0.71	0.52	1473	**415**	0.51	0.45	0.48	111
274	0.34	0.27	0.30	236	**416**	0.11	0.81	0.19	270
275	0.10	0.11	0.10	136	**423**	0.30	0.58	0.39	76
276	0.00	0.00	0.00	1483	**424**	1.00	0.00	0.01	583
278	0.33	0.54	0.41	272	**425**	0.10	0.72	0.18	223
280	0.45	0.20	0.28	212	**426**	0.35	0.29	0.32	160
284	0.13	0.57	0.21	97	**427**	0.55	0.75	0.64	1717
285	0.00	0.00	0.00	1268	**428**	0.00	0.00	0.00	1346
286	0.23	0.17	0.20	207	**433**	0.33	0.41	0.37	147
287	0.26	0.42	0.32	397	**434**	0.35	0.57	0.43	144
288	0.27	0.11	0.16	175	**437**	0.13	0.35	0.19	72
289	0.00	0.00	0.00	69	**438**	0.22	0.37	0.28	122
291	0.30	0.74	0.43	106	**440**	0.34	0.24	0.28	179
293	0.04	0.03	0.04	148	**441**	0.41	0.72	0.52	188
294	0.40	0.52	0.45	198	**443**	0.30	0.29	0.29	173
295	0.22	0.71	0.34	73	**453**	0.34	0.38	0.36	159
296	0.54	0.66	0.59	125	**456**	0.35	0.93	0.51	101
300	0.47	0.09	0.15	303	**458**	0.15	0.06	0.08	455
303	0.31	0.74	0.44	211	**459**	0.42	0.06	0.10	87
304	0.27	0.39	0.32	66	**482**	0.16	0.37	0.22	214
305	0.32	0.38	0.34	566	**491**	0.31	0.56	0.40	140
327	0.49	0.44	0.47	246	**492**	0.05	0.07	0.06	67
331	0.31	0.53	0.39	148	**493**	0.53	0.39	0.45	359
338	0.06	0.17	0.09	104	**507**	0.23	0.51	0.32	327
342	0.13	0.83	0.22	78	**511**	0.21	0.62	0.31	347

Taulukko 5. *Multiclass-ratkaisun luokittelijoiden luokkakohtaisia tarkkuusarvioita 1*

ICD-9	prec.	recall	f1	support	**ICD-9**	prec.	recall	f1	support
512	0.07	0.46	0.12	97	**745**	0.29	0.26	0.28	68
518	0.00	0.00	0.00	1312	**765**	0.85	0.88	0.87	78
519	0.33	0.66	0.44	122	**770**	0.57	0.98	0.72	60
530	0.57	0.43	0.49	756	**774**	0.61	1.00	0.76	62
535	0.23	0.37	0.28	107	**780**	0.24	0.49	0.32	508
536	0.34	0.25	0.29	55	**781**	0.08	0.36	0.13	53
537	0.28	0.58	0.38	78	**782**	0.02	0.07	0.03	84
553	0.40	0.18	0.25	92	**784**	0.41	0.14	0.20	148
557	0.11	0.64	0.19	69	**785**	0.23	0.64	0.34	475
560	0.23	0.44	0.30	189	**786**	0.17	0.31	0.22	144
562	0.33	0.55	0.41	137	**787**	0.13	0.16	0.14	255
564	0.11	0.11	0.11	128	**788**	0.38	0.11	0.17	213
567	0.13	0.54	0.21	117	**789**	0.29	0.56	0.38	202
568	0.16	0.06	0.08	71	**790**	0.15	0.26	0.19	340
569	0.20	0.23	0.22	116	**799**	0.05	0.05	0.05	164
571	0.00	0.00	0.00	284	**801**	0.24	0.75	0.36	72
572	0.00	0.00	0.00	192	**802**	0.27	0.67	0.39	60
574	0.22	0.77	0.34	116	**805**	0.21	0.67	0.32	103
576	0.28	0.77	0.41	118	**807**	0.33	0.75	0.46	120
577	0.27	0.75	0.40	138	**852**	0.19	0.77	0.31	105
578	0.28	0.47	0.35	245	**860**	0.22	0.70	0.33	67
583	0.32	0.26	0.28	70	**873**	0.22	0.74	0.34	96
584	0.00	0.00	0.00	1101	**995**	0.30	0.68	0.42	554
585	0.00	0.00	0.00	654	**996**	0.34	0.36	0.35	471
593	0.18	0.03	0.05	93	**997**	0.17	0.50	0.25	557
599	0.31	0.37	0.34	755	**998**	0.19	0.33	0.24	443
600	0.41	0.15	0.22	190	**999**	0.08	0.23	0.12	94
682	0.34	0.22	0.26	156	**E849**	0.10	0.26	0.15	200
707	0.70	0.29	0.41	314	**E878**	0.14	0.53	0.22	440
714	0.51	0.60	0.55	70	**E879**	0.09	0.45	0.15	249
715	0.47	0.05	0.09	169	**E885**	0.11	0.55	0.18	71
724	0.26	0.15	0.19	130	**E888**	0.14	0.50	0.22	107
728	0.75	0.08	0.14	79	**E932**	0.10	0.24	0.14	75
729	0.07	0.09	0.08	97	**E933**	0.24	0.31	0.27	59
730	0.47	0.39	0.43	61	**E934**	0.00	0.00	0.00	79
733	0.66	0.20	0.31	266	**E950**	0.39	0.88	0.54	42

Taulukko 6. *Multiclass-ratkaisun luokittelijoiden luokkakohtaisia tarkkuusarvioita 2*

	prec.	recall	f1	support
keskiarvo	0.26	0.35	0.26	42260

Taulukko 7. *Multiclass-ratkaisun luokittelijoiden keskiarvo kaikille luokille*

Myös *multiclass*-tyyppisissä luokittelijoiden luokkakohtaisissa tarkkuusarvioissa oli erittäin suuria eroja. Alimmillaan f1-arvo (0.00) oli 15 eri luokalla. Useat näistä huonon tarkkuusarvion saaneista luokista olivat myös varsin yleisiä, jotkin niistä esiintyen yli tuhannessa eri näytteessä. Tästä voidaan jo huomata, että tämä ratkaisutapa häviää tarkkuudessa huomattavasti *multilabel*-ratkaisulle. Maksimissaan f1-arvo (0.87) oli taas ICD-9 koodilla *765*, sama kuin *multilabel*-ratkaisulla. Myös koodin *765* lähellä olevat koodit *770* ja *774* saivat korkeat tarkkuusarviot, molempien ollessa yli 0.7. Nämä kaikki kolme koodia liityivät raskauteen ja sen mahdollisiin komplikaatioihin. Kuten *multilabel*-ratkaisun yhetydessä todettiin raskaudesta kirjoitettaessa käytetään tarkkoja ilmauksia ja potilaat ovat yleensä juuri näiden komplikaatioiden takia teho-osastolla. Silloin kysessä oleva vaiva on helposti poimittavissa näytteiden piirrevektoreista. Taulukosta 7 voidaan nähdä näistä luokittelijoista laskettu kokonaiskeskiarvo, joka on 0.26 ja se jää kauas aikaisemmin esitetyn ratkaisun vastaavasta arvosta 0.46. Yksittäisten luokittelijoiden kombinaatiolla on hieman matalampi täsmällisyyden arvo kuin aikaisemmallakin ratkaisuvaihtoehdolla, mutta löytökyky on vielä sitäkin paljon alhaisempi sen ollessa noin puolet *multilabel*-ratkaisun vastaavasta arvosta. Tämä ratkaisu on huomattavasti epätarkempi malli hoitojaksojen luokittelussa.

Kyseisten ratkaisumalleja voidaan vertailla toisiinsa vielä selvemmin generoitujen *true positive*-, *false positive*- ja *false negative*- sekä f1-arvojen kuvaajista kuvasta 9.

Kuten kuvasta voidaan nähdä *multilabel*-ratkaisun luokittelija ei ole suoraan parempi kaikkien luokkien tapauksessa. Usein sillä huonosti luokittuva luokka luokittuu vielä huonommin *multiclass*-ratkaisun luokittelijalla, mutta useat hyvin luokittuvat luokat taas luokittuvat vielä hieman paremmin *multilabel*-ratkaisun luokittelijalla.

4.5 Haasteet

Tulosten pohjalta nähdään että parannettavaa löytyy. ICD-9-koodiston koodit ovat eriarvoisia siten, että yksi koodi voi kuvata jotain yleistä potilaan vaivaa ja se saattaa olla potilaan muun tilan kannalta niin vähäpätöinen asia, että sitä ei merkitä lopullisiin diagnooseihin. Tästä on esimerkkinä aikaisemmassa luvussa esiintynyt ICD-9-koodi, *782*. Koska lääkärit keskittyvät vakavimpiin oireisiin ja sairauksien ilmentymiin tämänkaltaiset luokat eivät esiinny johdonmukaisesti näytteiden tulosdiagnooseissa, vaikka niihin viittavia piirteitä esiintyisikin näytedatassa.

Jotkin diagnoosi-koodit olivat myös selvästi vaikeasti luokiteltavia ominaisuuksia kokoavia luokkia, jolloin myös näitä oli vaikeata diagnosoida. Yhtenä esimerkkinä on ICD-9-koodi *999*, joka kuvasi lääketieteellisiä komplikaatioita, joita ei ole luokiteltu muiden koodien avulla. Kuvauksesta voi jo päätellä, että tähän luokkaan kuuluvia näytteitä olisi lähes mahdotonta luokitella tyydyttävästi.

Näiden asioiden lisäksi mahdollisten luokkien suuri määrä teki luokittelusta yksinkertaisesti haastavaa. Yleensä luokittelutehtävissä keskitytyään huomattavasti suppeampaan

luokkakattaukseen. Seuraavassa kappaleessa käsitellään tapoja, miten näitä haasteita ja ongelmia voitaisiin kiertää, jotta tutkimuksen kaltaista hoitomerkintöjen luokittelua voitaisiin tehdä tehokkaammin ja tarkemmin.

Kuva 9. *Luokittelijaratkaisujen vertailu*

5. JATKOKEHITYS

Työtä tehdessä tuli ilmeiseksi, että sitä voitaisiin kehittää paremmaksi monella eri tavalla. Nyt tehdyssä työssä jätettiin paljon huomiotta metadataa, jota voisi yrittää käyttää luokittelijan syötepiirteinä normaalien TF-IDF-piirteiden lisäksi. Esimerkiksi potilaan hoidon pituutta ja hoitojaksonsa aikana käyttämiä palveluita voitaisiin käyttää hyödyksi. Potilaiden merkinnät sisälsivät myös monentyyppistä tekstimuotoista dataa tekstimuotoisten muistiinpanojen ja merkintöjen sisällä. Niissä oli esimerkiksi erilaisten laboratorio-, kuvantamis- ja yleistutkimusten tuloksia. Nämä rakenteet olisi hyvä pystyä erottamaan muusta tekstistä ja käsittelemään eri tavalla. Nyt ei myöskään hyödynnetty hoitomerkintöjen aikariippuvuutta eli sitä missä järjestyksessä potilaan hoitotapahtumat tapahtuivat. Seuraavissa kappaleissa käsitellään tarkemmin erilaisia päätelmiä ja ajatuksia, miten diagnoosien luokittelijoiden rakentamista saadaan tehostettua ja koostettuja malleja tarkemmiksi.

5.1 Aikariippuvainen data

Datan aikariippuvuutta ei työssä käytetyillä koneoppimisen malleilla pystytty käsittelemään. Ajan huomioonottavia oppimismalleja on olemassa ja iso osa niistä käyttää hermoverkkoja toiintamallinaan. Takaisinkytketty hermoverkko (RNN, *recurrent neural network*) on tietynlainen hermoverkko, jossa solmut voivat kytkeytyä takaisin aikaisempiin solmuihin. Se pystyy näiden takaisinkytkettyjen syötteiden avulla käsittelemään datan aikariippuvuutta, koska aikaisemmin tullut syöte syötetään tämänhetkisen syöteen kanssa takaisinkytketyille solmuille. RNN:t voivat käyttää sisäistä tilaansa kuten muistia uuden syötedatan käsittelemisessä. Tämä eroaa suuresti tavallisesta hermoverkosta, joka sisältää vain eteenpäin kytkettyjä syöteyhteyksiä. RNN:t voidaan soveltaa tämän työn kaltaisiin ongelmiin, jossa aikajärjestyksellä ja riippuvuudella on merkitystä saamaan lisäarvoa luokittelua varten.

Takaisinkytketyillä hermoverkoilla voidaan tarkoittaa kahta erilaista verkkotyyppiin; FIR- (*finite impulse response*) ja IIR-tyyppisiin (*infinite impulse response*) verkkoihin. Rakenne näiden välillä ei eroa paljon toisistaan ja molemmat pystyvät hyödyntämään aikariippuvaista dataa, mutta FIR-verkko voidaan korvata tarvittaessa myös eteenpäin kytketyllä hermoverkolla, kun taas IIR-verkkoa ei voida. Tämä johtuu siitä, että IIR-verkko käyttää takaisinkytkennän syötteenä koko aikaisempaa syötehistoriaa jossain muodossa, jolloin eteenpäin kytketystä verkosta tulisi koko syötehistorian suuruinen. FIR-verkko taas käyttää vain jotain tiettyä ajanhetken syötettä takaisinkytkennöissään. FIR-verkko sanotaan asykliseksi kaavioksi ja IIR-verkkoa sykliseksi kasvioksi näiden ominaisuuksien takia.

5.2 Target drift ja online-oppiminen

Jos otetaan huomioon terveydenhuollon tapahtumien aikariippuvuus, törmätään *target drift* -ilmiöön. Tämä ilmenee terveydenhuollossa sillä, että ensin potilasta vaivaa jokin sairaus ja vähitellen uusi sairaus nousee vallitsevaksi. Ennustavaan analyysiin ja koneoppimisen piirissä target drift -ilmiö tarkoittaa, että ennustettavien luokkien tilastolliset ominaisuudet, joita malli yrittää ennustaa, muuttuvat ajan myötä ennakoimattomilla tavoilla. Tämä aiheuttaa ongelmia, sillä luokittelu menettää tarkkuutta ajan kuluessa. *Target* viittaa useimmin ennustettavaan luokkaan, mutta se voi viitata myös muihin kiinnostaviin ominaisuuksiin kohdeluokkien lisäksi, kuten syötteen piirteisiin.

Oppimista voitaisiin tehdä jatkuvasti. Aina kun tehdään diagnoosiluokitus jostakin uudesta syötedatasta lääkäri antaa potilaalle lopullisen diagnoosin. Tätä lopullista diagnoosia voidaan taas käyttää mallin verifioimiseen. Näin aina kun mallia käytetään luokitteleemaan uusi näyte, tullaan myös mallia kehitettyä hieman paremmaksi. Tämän kaltaista jatkuvaa oppimista kutsutaan online-oppimiseksi. Online-oppiminen on mahdollista, kun uutta oppimisongelmalle oleellista dataa saadaan järjestyksessä luokittelijalle. Tätä dataa voidaan hyödyntää ylläolevan kuvauksen mukaisesti parantamaan luokittelua tekevää mallia. Online-oppimista käytetään varsinkin kun luokittelijan on sopeuduttava uuteen tilanteeseen nopeasti ja se pystyy siten kiertämään osittain *target drift* -ilmiötä. Sitä käytetään myös tilanteissa, joissa opetusaineistoa on suuri määrä, jolloin sitä kaikkea ei voida kerralla opettaa luokittelijalle.

5.3 Sääntöpohjaiset menetelmät koneoppimisen tukena

Luvussa 4.5 mainitut ongelmat voitaisiin ehkä ottaa paremmin huomioon käyttämällä jonkinlaista tätä tarkoitusta varten rakennettua analyysityökalua. ICD-koodistoja tulisi käyttää kuten oikea lääkäri niitä käyttää, sillä pelkällä koneoppimisella tehty luokittelu ei pysty hyödyntämään koodiston sääntöpohjaisuutta kunnolla. Kehitettävän metarakenteen tulisikin hyödyntää sekä koneoppimista, että sääntöpohjaisia luokittelumetodeja, jotta tehty luokittelu saataisiin vietyä tarkkuudeltaan lähemmäs ihmislääkärin tasoa.

6. YHTEENVETO

Tämän työn tarkoitus oli selvittää, miten koneoppimista käyttäen voitaisiin ennustaa tekstimuotoisesta potilasdatasta potilaan saamat diagnoosit. Työssä toteutettiin yksinkertainen koneoppimisen sovellus käyttäen Python-ohjelmointikieltä ja erityisesti koneoppimiseen erikoistunutta Scikit-learn-ohjelmointikirjastoa. Sovelluksen tarkoituksena oli erityisesti vertailla koneoppimisen eri algoritmeja ja kehitellä niillä luokittelukyvyltään tarkka diagnoosien luokittelua tekevä malli.

Digitalisaation myötä terveydenhuolto tuottaa kasvavan määrän tietoa potilaista. Suuri osa tästä tiedosta on tekstimuotoista. Kerättyä tietoa hyödynnetään pääsääntöisesti vain potilaiden normaalissa hoitotyössä ja sitä voitaisiin käyttää myös automaattisissa analyyttisissä työkaluissa tehostamaan terveydenhuollon eri prosesseja. Varsinkin lääkärien aika on hyvin arvokasta ja koneoppimisen avulla voitaisiin tuottaa potilaalle mahdollisia diagnooseja potilaan kuvailemien oireiden, tutkimustulosten ja aikaisempien sairauksien pohjalta automaattisesti. Lääkäri voisi käydä nämä tuotetut diagnoosit läpi ja valmistella itse niiden pohjalta lopulliset diagnoosit, jolloin diagnosointityö voisi olla nopeampaa ja lääkärien antamat diagnoosit luotettavampia.

Tekstimuotoista dataa käsitellessä yleisesti toimitaan sanatasolla, jolloin on tärkeää, että vain luokittelun kannalta oleelliset sanat huomioidaan ja merkityksettömät välimerkit ja kirjasintiedot riisutaan datasta. Sanoja ei käytetä suoraan koneoppimisessa, vaan niistä generoidaan numeerisia ominaisuuksia, kuten esiintymismääriä ja tiheyksiä. Piirteiden johtamisen lisäksi dataa käsitellään siten, että se olisi mahdollisimman kuvaavaa juuri diagnosoinnin kannalta. Tällaisia tapoja ovat terveydenhuollon termien yhdenmukaistaminen ja diagnoosikoodien käsittely.

Terveydenhuollon tekstimuotoista dataa ei ole paljon yleisen käytön piirissä. Tämä on ymmärrettävää, sillä ne sisältävät arkaluontoista dataa, jota on vaikea muuntaa tietoturvalliseksi. Työtä varten löytyi lähes ihanteellinen MIMIC-III-potilastietokanta [5], joka sisälsi oikeaa tekstimuotoista anaonymisoitua hoitotietoa sekä ICD-9-muotoisia diagnoosikoodeja. Se sisälsi kuitenkin vain teho-osastoympäristöstä kerättyä dataa, jolloin potilaiden keskimääräinen ikä oli melko korkea ja sairauksista verisuonitaudit ja syövät olivat paljon muita yleisempiä diagnoosien joukossa. MIMIC-III sisältämä luokkien suhteen epätasapainoinen moniluokkainen data johtaa helposti luokittelijoihin, jotka ennustavat yksinkertaisesti yleisimpiä luokkia. Tähän voidaan vaikuttaa käyttämällä luokittelijoiden luokittelukykymittarina täsmällisyyttä (*precision*) ja löytökykyä (*recall* pelkän tarkkuuden sijaan (*accuracy*). Arvosteluparametrina f1-arvo (*f1-score*) on täsmällisyyttä ja löytökykyä helppokäyttöisempi, sillä se lasketaan niiden avulla ja se ottaa ne molemmat huomioon.

Työssä toteutettiin tutkimus, jonka tarkoituksena oli koostaa koneoppimisen avulla potilaiden diagnosointia avustavia malleja. Opetusdatana käytetystä MIMIC-III-tietokannasta etsitiin hoitojaksojen lääkärin loppuarvio, joka kokoaa kattavasti potilaasta kerätyn tiedon. Näistä tekstidokumenteista kerättiin piirrevektoreita ja hoitojaksojen diagnoositaulukon avulla luokkatiedot. Mahdollisia luokkia oli paljon ja jokaista näytettä kohden on yksi tai useampi luokka, jolloin puhutaan multilabel-ongelmasta. Luokkien määriä päätettiin rajata ICD-9 koodien ylemmän tason mukaan ja karsimalla vähiten lähdeaineistossa esiintyneet diagnoosikoodit, jolloin päädytiin lopulliseen 150 eri luokkaan. Toteutetulla sovelluksella päädytiin vertailemaan seitsemää luokittelijaa, joita varten löytyi toteutukset käytetystä Scikit-learn-kirjastosta. Jotkin luokittelijat käärittiin *OneVsRestClassifier*-toteutukseen, jotta ne tukisivat *multilabel*-tyyppistä ongelmaa. Toisena vaihtoehtona kaikille 150:lle luokalle rakennettiin omat luokittelijakandidaattinsa, jonka tarkoituksena oli kiertää *multilabel*-näkökulmaa.

Ratkaisuja arvioitiin Scikit-learn-kirjaston tarjoaman *Pipeline*-rakenteen ja *GridSearchCV*-ristivalidaation avulla, jolloin pystyttiin kartoittamaan eri luokittelijoille suuri parametriavaruus ja näin haarukoimaan optimaaliset parametrit. *Pipeline*-rakenteen avulla pystyttiin linkittämään lähdeaineiston esikäsittely- ja piirteidenmuodostusprosessit yhteen, jolloin iterointi oli nopeaa ja helppokäyttöistä. Käytetyssä arviointimittarissa, muunnellussa F1-arvossa, yksittäisen luokan painotus keskiarvon laskemisessa määräytyi sen esiintymismäärän mukaan. Kaksi ratkaisumallia suorituivat ongelmasta vaihtelevasti. Huonommin suoritui yksittäisille tulosluokkadiagnooseille rakennetut luokittelijat joiden f1-arvojen keskiarvo oli noin 0.26. Paremmin pärjäsi *multilabel*-ongelmaan tähtäävä ratkaisu, jonka paras luokittelija ylsi f1-arvoon 0.46.

Työn sovelluksen avulla osoitettiin, että kehitetyt luokittelijat pystyvät luokittelemaan potilaiden hoitojaksoja diagnooseittain ICD-9-koodiston ylemmän hierarkiatason mukaan vaihtelevalla menestyksellä. Kehnoa luokittelukykyä joidenkin diagnoosien osalta selittää se, että ICD-9-koodiston koodit ovat eriarvoisia siinä miten niitä käytetään. Jotkin diagnoosikoodit kokosivat vaikeasti muuten luokiteltavia tauteja ja toiset kuvasivat tehohoosastohoidon kannalta vähäpätöisempiä oireita. Luokittelun tuloksiin vaikuttaa myös lopuulisten luokkien suuri määrä ja se onko luokkia käytetty pääsääntöisesti ensisijaisena vai toissijaisena diagnoosina.

Työn kehittämiselle on paljon mahdollisuuksia ja luokittelukyvyssä on paljon parannettavaa. Jatkotyötä ajatellen pitäisi kokeilla syvien neuroverkkojen sekä takaiskytkettyjen verkkojen soveltuvuutta ongelmaan. Takaisinkytketty verkko pystyy hyödyntämään datassa olevaa aikariippuvuutta, jota ei tehdyssä työssä huomioitu ollenkaan. Metadatan ja ei-tekstimuotoisen datan hyödyntäminen voisi olla myös tulosta parantava tekijä. Numeeristen laboratoriotulosten ja kuvantamisdatan hyödyntäminen pitäisi kuitenkin käsitellä eri tavlla kuin tekstimuotoinen data. Eräs vaihtoehto jatkokehitykseen olisi miettiä sääntöpohjaisten ratkaisujen hyödyntämistä tavanomaisen koneoppimisen kanssa rinnan.

LÄHTEET

[1] Antti Ajanki AnAj, Example of k-NN classification, Wikipedia, the free encyclopedia, verkkosivu, 2007. Saatavissa (viitattu 15.5.2018): https://en.wikipedia.org/wiki/File:KnnClassification.svg

[2] S. C, C. L, L. L, et al, Allocation of physician time in ambulatory practice: A time and motion study in 4 specialties, Annals of Internal Medicine, vsk. 165, nro 11, 2016, s. 753–760.

[3] T. Colliau, G. Rogers, Z. Hughes, C. Ozgur, MatLab vs. Python vs. R, huh. 2016.

[4] Cyc, Graphic showing the maximum separating hyperplane and the margin, Wikipedia, the free encyclopedia, verkkosivu, 2008. Saatavissa (viitattu 15.5.2018): https://en.wikipedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png

[5] A. Edward William Johnson, T. Joseph Pollard, L. Shen, L. w. Lehman, M. Feng, M. Ghassemi, B. Edward Moody, P. Szolovits, L. Anthony G. Celi, R. G. Mark, MIMIC-III, a freely accessible critical care database, vsk. 3, tou. 2016.

[6] H. EELKE, Y. KEVIN, T. F. W., C. BRUCE, G. B. JAVIER, H. L. F., W. W. KINDRED, P. VLADIMIR, L. L. AUDREY, The promise and perils of using big data in the study of corporate networks: problems, diagnostics and fixes, Global Networks, vsk. 18, nro 1, jou. 2017, s. 3–32.

[7] P. Groves, B. Kayyali, D. Knott, S. Van Kuiken, The "big data"revolution in healthcare. Accelerating value and innovation, tam. 2013.

[8] R. Hallén, A Study of Gradient-Based Algorithms, 2017. Student Paper.

[9] HCUP Facts and Figures: Statistics on Hospital-Based Care in the United States, National Center for Biotechnology Information, U.S. National Library of Medicine, verkkosivu, 2009. Saatavissa (viitattu 15.5.2018): https://www.ncbi.nlm.nih.gov/books/NBK91980/

[10] J. D. Hunter, Matplotlib: A 2D graphics environment, Computing In Science & Engineering, vsk. 9, nro 3, 2007, s. 90–95.

[11] ICD, World Health Organization, verkkosivu, 2018. Saatavissa (viitattu 15.5.2018): http://www.who.int/classifications/icd/en/

[12] juan carlos arellano, Free Mexican Flag Stock Photo, 2006. Saatavissa (viitattu 15.5.2018): https://www.freeimages.com/photo/mexican-flag-1419731

[13] R. Leaman, R. Dogan, Z. lu, DNorm: Disease Name Normalization with Pairwise Learning to Rank, vsk. 29, elo. 2013.

[14] T. Leuhu, SENTIMENT ANALYSIS USING MACHINE LEARNING, diplomityö, Tampereen teknillinen yliopisto, Signaalinkäsittelyn laitos, Tampere, 2014, 54 p.

[15] Lisa Setrini-Espinosa, Free American Flag Stock Photo, 2005. Saatavissa (viitattu 15.5.2018): https://www.freeimages.com/photo/american-flag-1314998

[16] E. Loper, S. Bird, NLTK: The Natural Language Toolkit, teoksessa: Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1, Stroudsburg, PA, USA, 2002, Association for Computational Linguistics, ETMTNLP '02, Philadelphia, Pennsylvania, s. 63–70.

[17] P. Marcheschi, Relevance of eHealth standards for big data interoperability in radiology and beyond, La radiologia medica, vsk. 122, nro 6, Jun, 2017, s. 437–443.

[18] marmit, Free Swedish flag Stock Photo, 2007. Saatavissa (viitattu 15.5.2018): https://www.freeimages.com/photo/swedish-flag-1443423

[19] W. McKinney, Data Structures for Statistical Computing in Python, teoksessa: Walt, S. van der, Millman, J. (toim.), Proceedings of the 9th Python in Science Conference, 2010, s. 51 – 56.

[20] M. Mohri, A. Rostamizadeh, A. Talwalkar, Foundations of Machine Learning, The MIT Press, 2012.

[21] Multiclass and multilabel algorithms, scikit-learn developers, verkkosivu, 2018. Saatavissa (viitattu 15.5.2018): http://scikit-learn.org/stable/modules/multiclass.html

[22] T. E. Oliphant, Guide to NumPy, 2nd p., CreateSpace Independent Publishing Platform, USA, 2015.

[23] 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, E. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, vsk. 12, 2011, s. 2825–2830.

[24] Potilasasiakirjoihin sisältyvien tietojen salassapito, Finlex, verkkosivu, 2000. Saatavissa (viitattu 15.5.2018): https://www.finlex.fi/fi/laki/ajantasa/1992/19920785#L4P13

[25] The Presence of Highly Similar Notes Within the MIMIC-III Dataset, AMIA 2017 Annual Symposium, verkkosivu, 2017. Saatavissa (viitattu 15.5.2018): https://amia2017.zerista.com/event/member/389641

[26] Päivi Tiittanen, Free Finnish Flag Stock Photo, 2007. Saatavissa (viitattu 15.5.2018): https://www.freeimages.com/photo/finnish-flag-1172680

[27] C. Sammut, G. I. Webb, Encyclopedia of Machine Learning and Data Mining, 2nd p., Springer Publishing Company, Incorporated, 2017.

[28] Sosiaali- ja terveystietojen toissijainen käyttö, Sosiaali- ja terveysministeriö, verkkosivu, 2017. Saatavissa (viitattu 15.5.2018): https://stm.fi/sote-tiedon-hyodyntaminen

[29] M. Sutinen, BIG DATA JA ANALYTIIKKA TERVEYDENHUOLLOSSA, 2016, 36 s.

[30] Tibor Fazakas, Free flags Stock Photo, 2006. Saatavissa (viitattu 15.5.2018): https://www.freeimages.com/photo/flags-1445227

[31] Walber, Precision and recall, Wikipedia, the free encyclopedia, verkkosivu, 2014. Saatavissa (viitattu 15.5.2018): https://en.wikipedia.org/wiki/File:Precisionrecall.svg

[32] J. Walonoski, M. Kramer, J. Nichols, A. Quina, C. Moesel, D. Hall, C. Duffett, K. Dube, T. Gallagher, S. Mclachlan, Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record, vsk. 0, syy. 2017, s. 1–9.

[33] B. Winters, J. Custer, S. M. Galvagno, E. Colantuoni, S. G. Kapoor, H. Lee, V. Goode, K. Robinson, A. Nakhasi, P. Pronovost, D. Newman-Toker, Diagnostic errors in the intensive care unit: a systematic review of autopsy studies, BMJ Quality & Safety, vsk. 21, nro 11, 2012, s. 894–902.

[34] X. Zhu, A. B. Goldberg, R. Brachman, T. Dietterich, Introduction to Semi-Supervised Learning, Morgan and Claypool Publishers, 2009.

LIITE A: TOTEUTUKSEN LÄHDEKOODI

Saatavilla osoitteessa:

https://github.com/verdantred/Assisting-diagnosis-using-medical-textual-records