Tampereen yliopisto

Kasper Styrman

NAIIVIN KONEOPPIMISEN VÄLTTÄMINEN KOULUTUSDATAN ESIKÄSITTELYLLÄ

Eturauhassyövän hoitotuloksen ennustaminen koneoppimismenetelmillä

Kandidaatintyö
Lääketieteen ja terveysteknologian tiedekunta
Tarkastajat: Matti Nykter
Huhtikuu 2022

TIIVISTELMÄ

Kasper Styrman: Naiivin koneoppimisen välttäminen koulutusdatan esikäsittelyllä
Kandidaatintyö
Tampereen yliopisto
Bioteknologian ja biolääketieteen tekniikan tutkinto-ohjelma TkK
Huhtikuu 2022

Tässä työssä tutkittiin, miten koneoppimisella saatuja ennustustuloksia pystytään parantamaan esikäsittelemällä dataa. Työssä koulutettiin *Support Vector Machine* -koneoppimisalgoritmi (SVM) ennustamaan kastraatioresistenttien eturauhassyöpäpotilaiden hoidon vastetta. Koulutuksessa haluttiin saada kone ennustamaan potilaalle oikea vaste potilaan AR-, TP53- sekä RB1-geenien kopiolukumuutosten pohjalta. Vaste luokiteltiin joko hyvään tai huonoon vasteeseen sen pohjalta, kuinka pitkään potilas eli hoidon aloittamisen jälkeen. Käytetty potilasdata oli epätasapainoista. Tässä tapauksessa hyvän vasteen pisteitä oli huomattavasti enemmän huonon vasteen pisteisiin verrattaessa. Tästä johtuen jakauman epätasapainoa kompensoitiin yli- ja alinäytteistämällä datasettiä. Työn pääpaino olikin vertailla erityyppisten datan esikäsittelyjen vaikutusta saatavaan mallin hyvyyteen.
Työssä ohjelmoitiin Pythonin `sklearn.svm` -kirjaston SVM-työkaluilla. Datan esikäsittelyä varten toteutettiin myös omat funktiot. Potilasdatasta haettiin potilaiden kopiolukumuutokset sekä elinkuukaudet. Datasta otettiin erilleen pienempi setti pisteitä mallin testaamista varten. Tämän jälkeen data ajettiin esikäsittelyn läpi, jonka jälkeen sillä koulutettiin SVM-algoritmi. Lopuksi mallin ennustamia tuloksia vertailtiin tunnettuihin vastearvoihin laskemalla erinäisiä avainlukuja, joiden pohjalta eri datan saaneiden mallien hyvyyttä vertailtiin keskenään. Datan muokkaaminen ennen käyttöä paransi ennustustuloksia, kun kyseisiä malleja vertailtiin esikäsittelemättömällä datalla koulutettuihin malleihin. Tulokset paranivat esikäsittelyllä ja yleisesti parhaimmat tulokset saatiin, kun eri luokkien jakauma oli $50/50$ ja testidataksi oli otettu sivuun 20% pisteistä. Tulokset tukevat yleisesti tunnettua ongelmaa koneoppimisessa liittyen datajakauman tasapainoon ja mallin testaamiseen.

Avainsanat: koneoppiminen, datan epätasapaino, eturauhassyöpä, biotekniikka, genetiikka

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

ABSTRACT

Kasper Styrman: Avoiding Naive Machine Learning by Preprocessing Training Data
Bachelor of Science
Tampere University
B.Sc. in biotechnology and biomedical engineering
April 2022

The main focus of this bachelor's thesis was to study how data preprocessing could be used to improve results gotten from machine learning algorithm predictions. A Support Vector Machine learning algorithm (SVM) was trained to predict the treatment outcome of patients with castration resistant prostate cancer. The aim was to get the machine to predict to correct treatment outcome based on a patient's copy number alteration value in the AR, TP53 and RB1 genes. The treatment outcome was classified as being either good or bad based on how long the patient lived after the treatment was started. The patient data used was imbalanced. In this case there were considerably more data points with a 'good' outcome value compared to the amount of 'bad' outcomes. Therefore, the data distribution imbalance was compensated by over- and undersampling the dataset. Emphasis in the thesis was put on comparing the effect different data preprocessing procedures had on the gotten predictive model's goodness. The created program used tools from the Python sklear.svm library. Data preprocessing was also implemented as separate Python functions. The patients' copy number alteration values as well as overall survival month amounts were fetched from the patient data files. A small portion of data points was taken aside for performing model testing later on. Afterwards, the data was preprocessed after which it was used to train an SVM algorithm. Finally, the treatment outcome values predicted by the model were compared with already known outcome values by calculating different index values. The Indices were used for comparing the results gotten from the different models; each trained with a differently preprocessed dataset. Data preprocessing did improve the prediction results when compared to those of models trained with non-preprocessed data. Generally, the best results were gotten when the class distribution was $50/50$ and the amount of test datapoints was 20% of the original. These results support the common notion of data imbalance as a problem in the field of machine learning, as well as the difficulty of model testing.

Keywords: machine learning, data imbalance, prostate cancer, bioengineering, genetics

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

ALKUSANAT

Eturauhassyöpä on yleisimmin esiintyvä syöpä miehillä. Suomessa sairastuneiden keski-ikä on 70 vuotta. Useimmat potilaat kuolevat syövän sijasta iän tuomiin komplikaatioihin.

Tässä kandidaatintyössä esitellään yleisesti koneoppimista ja eturauhassyöpää, sekä tutkitaan, kuinka koneoppimista voitaisiin mahdollisesti hyödyntää kyseisen syövän hoidossa. Työ sisältää koodeja, jotka on toteutettu Pythonilla.

Tämän työn pohjana on vuonna 2019 julkaistu tutkimus potilaiden genetiikan vaikutuksesta hoitotulokseen. Käytetty data julkaistiin Githubissa ja hyödynnänkin sitä tutkimuksessani.

Kiitos kaikille, jotka auttoivat minua tämän työn kirjoittamisessa!

Tampereella, 16. huhtikuuta 2022

Kasper Styrman

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

Gleason	Syövän vaikeusastetta kuvaava pisteytysasteikko. Perustuu kudosnäytteissä havaittavaan syövän erilaistumistapaan.
GnRH	(engl. *Gonadotropin Releasing Hormone*) Hormoni, joka saa aivolisäkkeen etulohkon vapauttamaan lutenisoivaa (LH) ja follikkeleja stimuloivaa hormonia (FSH). Nämä gonadotropiinit säätelevät sukupuolirauhasten toimintaa.
ML	koneoppiminen (engl. *Machine Learning*)
PSA	eturauhaselle spesifi merkkiaine, jonka pitoisuus veressä kasvaa muun muassa eturauhassyövässä (engl. *Prostate-Specific Antigen*)
SVC	luokitteleva tukivektoreihin pohjautuva koneoppimisalgoritmi (engl. *Support Vector Classifier*)
SVM	koneoppimisalgoritmi, pohjautuu tukivektoreihin (engl. *Support Vector Machine*)
TNM	syöpäluokitus; kuvaa syövän kokoa (T), syöpäsoluja sisältävien imusolmukkeiden määrää (N) sekä etäpesäkkeiden määrää (M) (engl. *Tumor, Nodes, Metastasis*)

1 JOHDANTO

Tämän kandidaatintyön tarkoitus on tutustua tarkemmin koneoppimisen periaatteisiin ja käytännön sovelluksiin sekä riittävästi eturauhassyöpään ja sen hoitoon lääketieteellisestä näkökulmasta. Tarkoitus on perehtyä aihetta koskevaan kirjallisuuteen sekä julkaistuun geneettiseen ja kliiniseen dataan, rakentaa Python-ohjelmointikielelle luotua scikit-learn -koneoppimiskirjastoa hyödyntäen erilaisia tukivektorikone-malleja (eng. *support vector machine*, SVM), testata hoidon vasteen ennustamista, sekä lopuksi vertailla ja arvioida luotuja malleja. Merkittävä osa työssä on myös datan jakautuneisuudella ja sen tasapainottamisella.

Teknologian kehittyessä on luonnollista, että löydetään uusia ja innovatiivisia sovelluksia. Muun muassa tekoäly ja koneoppiminen (eng. *artificial intelligence*, AI; *machine learning*, ML) ovat yleisiä puheenaiheita tekniikan piireissä tällä hetkellä. Tämä niin kutsuttu keinoäly tuo paljon mahdollisuuksia esimerkiksi optimoinnin ja automatisoinnin suhteen monella alalla, mukaan lukien lääketieteessä. Lääketiede on monimutkaista, eikä ihmiskehosta ja sen hoitamisesta tiedetä läheskään kaikkea. Onkin kiinnostavaa pohtia, tarjoaisiko tekoälyn hyödyntäminen uusia kehityspolkuja niin lääketieteellisen tutkimuksen kuin hoitotyönkin puolelle.

Nature Materials -lehdessä julkaistiin vuonna 2019 tällaista tekoälyn ja lääketieteen yhteistyötä visioiva teksti [1]. Tekstissä kuvailtiin, kuinka koneoppimista voitaisiin hyödyntää ja hyödynnetäänkin muun muassa lääkekehityksessä, diagnostiikassa ja hoitoennusteissa. Lääketieteellinen data on usein melko kompleksia ja konealgoritmi saattaa pystyä löytämään yhtäläisyyksiä siellä, missä ihminen ei. Koneoppimisella voidaan esimerkiksi parantaa kuvantamisen tuloksekkuutta tai löytää ja spekuloida uuden mahdollisen lääkeaineen toimivuutta jonkin tietyn taudin hoidossa. [1] Tulevaisuus toki näyttää, mihin suuntaan kehitys vie. On myös syytä suhtautua kriittisesti kehityksen mukana tuleviin uusiin riskeihin; syötetyn datan huomaamaton ja pahantahtoinen manipulointi voisi johtaa esimerkiksi täysin vääriin diagnooseihin ja siten vaikuttaa hoidon turvallisuuteen automatisoidummassa terveydenhuollossa.

Eturauhassyöpä on yleisin syöpä miehillä. Suomessa eturauhassyöpiä diagnosoidaan 5000 kappaletta vuodessa. Tyypillisesti se ilmaantuu vasta myöhemmällä iällä. Eturauhassyöpä on harvemmin potilaan kuolinsyy. [2]

Vuonna 2019 W. Abida et al. julkaisivat tutkimuksen [3], jossa tutkimusryhmä selvitti yli 400 myöhäisen vaiheen eturauhassyöpäpotilaan genetiikan ja hoidon toimivuuden yhteyttä. Potilaiden syöpä oli edennyt niin pitkälle, että se oli muodostanut etäpesäkkeitä ja kehittänyt resistenssin kastroiville hoitokeinoille. Annettava hoito siis pyrkii tässä vaiheessa lieventämään oireita ja pidentämään elinikää pääasiassa hormonaalisilla hoidoilla, muttei enää pyri parantamaan tautia. Tutkimuksessa keskityttiin enemmän tilastolliseen ja biologiseen analyysiin, ja siinä löydettiinkin yhteys muun muassa RB1-geenin mutaation ja huonon hoitovasteen välillä. Tällainen analyysi lienee kuitenkin mahdollista toteuttaa myös koneoppimismenetelmillä.

Tässä tutkimuksessa katsotaan koneoppimista sekä tarkemmin SVM-algoritmeja. Työn kannalta on myös olennaista ymmärtää perusteet eturauhassyövästä. Käytettävän potilasdatan jakauma on epätasapainoinen. Tästä johtuen työssä selvitetään, onnistutaanko mallin laatua parantamaan esikäsittelemällä dataa yli- ja alinäytteistysmenetelmillä. Malleja arvioidaan ja vertaillaan niiden ennustamien arvojen pohjalta laskettujen avainlukujen avulla. Työssä arvioidaan myös, pystyykö hoidon vastetta ennustamaan geneettisen datan pohjalta käyttäen SVM-mallia.

2 KONEOPPIMINEN

2.1 Yleistä

Koneoppiminen on tekoälyn osa-alue. Tekoäly pyrkii imitoimaan ihmisen älykkyyttä koneiden ja ohjelmistojen toiminnassa. Kone voidaan esimerkiksi koodata siirtämään nappulaa shakkipelissä tietyllä tavalla riippuen vastustajan siirrosta. Koneoppiminen pyrkii lisäämään koneen autonomisuutta luomalla ohjelmiston, jolla kone päätyy haluttuun tulokseen ilman ulkoista ennaltamäärättyä ohjeistusta.[4]

Tekoälyä on tutkittu vuodesta 1956 lähtien, kun ensimmäinen tutkimus tehtiin Darthmouthin yliopistossa. Siitä lähtien tekoälyn tutkimus ja rahoitus on kokenut ylä- ja alamäkiä, kun sijoittajat vuorotellen kiinnostuivat tekoälyn mahdollisuuksista, mutta pettyivät tuloksiin. 2000-luvulla kiinnostus on kuitenkin taas lähtenyt kasvuun juurikin koneoppimisen onnistuneiden sovellusten myötä.[4]

Oppaassaan [5, s. 7] A. Jung erittelee koneoppimisen koostuvan kolmesta pääkomponentista: datasta, hypoteesiavaruudesta ja tappiofunktiosta. Data merkitsee mallinnettavasta systeemistä kerättävää informaatiota, hypoteesiavaruus erilaisia mahdollisia mallinnusfunktioita ja tappiofunktiolla pyritään optimoidaan ja arvioimaan määritetyn mallin hyvyyttä.

2.2 Ominaisuudet ja nimikkeet

Ominaisuudet ovat joukko datasta erotettavia muuttujia. Näitä voivat olla periaatteessa mitkä tahansa datapisteestä havaittavissa olevat piirteet, kuten esimerkiksi väri, paino ja muoto. Näiden parametrien pohjalta tekoäly pyrkii ennustamaan jonkin nimikearvon datapisteelle. Nimikearvo voi olla esimerkiksi kategorinen luokka, johon piste kuuluu tai jonkin havainnollistavan suureen lukuarvo. Koneoppimisen hyöty tuleekin esiin tilanteissa, joissa ominaisuudet ovat helposti mitattavissa, mutta nimikearvo ei. Tällöin koneelle voidaan syöttää mitatut parametrit ja se hoitaa lopun päättelytyön. [5, s. 10]

Tekstissään A. Jung huomauttaa, että valittavat ominaisuudet riippuvat paljolti sovelluksen tarkoitusperästä, ja näitä valittaessa tulisi pyrkiä maksimoimaan muuttujista saatavan informaation määrä ja hyöty. A. Jung nimittääkin kaikkia mahdollisia ominaisuuksia

ominaisuusavaruudeksi $\mathbf{X}$, josta rajataan sovelluksen kannalta merkityksellisten ominaisuuksien joukko $\overline{x}$. Oikeiden ominaisuuksien valitseminen voi usein olla jopa vaikein vaihe koneoppimisprosessia.

Syöpänäytteiden kohdalla näytteen ominaisuuksia voisivat siis olla muun muassa kudostyyppi, josta näyte on otettu tai näytteen genomissa tapahtuneet mutaatiot, sillä näiden voidaan olettaa olevan merkityksellisiä tutkimuskohteen kannalta. On hyvä mainita myös, että on mahdollista siirtää vastuu ominaisuuksien valitsemisesta tekoälylle niin kutsutuilla syväoppimisen menetelmillä. [5, s. 10]

Nimikkeet ovat arvoja, joita datapisteen voidaan arvioida edustavan [5, s. 11-12]. Esimerkiksi jos ohjelmalla tutkitaan hedelmiä, se voisi datan ominaisuuksien perusteella luokitella ne joko omenoiksi, banaaneiksi tai päärynöiksi. Nämäkin arvot edustaisivat vain osaa kaikista mahdollisista nimikeavaruuden $\mathbf{Y}$ joukoista. Mikäli ennustettava arvo on diskreetti, on kyseessä luokitteluongelma. Jos taas ennustetaan jatkuvaa reaaliarvoa, kuten painoa tai elinikää, on kyseessä regressio-ongelma. Sovelluksesta riippuen toinen ennustustyyppi voi olla helpompi toteuttaa kuin toinen. Tämän tutkimuksen kannalta lienee helpompi tehdä diskreetti luokittelija muun muassa datapisteiden vähyydestä johtuen.

Koneoppimismenetelmät tähtäävät luomaan mallin, jolla pystytään mahdollisimman tarkasti ennustamaan haluttu arvo. Jos koetetaan ennustaa pisteen ominaisuuksien pohjalta sen jo tunnettua arvoa $\mathbf{y}$, tulisi varsinaisen arvon $\mathbf{y}$ ja mallin ennustaman arvon $\hat{y}$ välinen etäisyys olla keskiverrosti mahdollisimman pieni luku ε.

Malleista saatavat tulokset voidaan T. Hastie et al. [6, s. 9–10] mukaan jakaa kvantitatiivisiin ja kvalitatiivisiin. Kvantitatiiviset tulokset ovat reaalilukuarvoja (esimerkiksi lämpötila tai ilmanpaine), kun taas kvalitatiiviset arvot ovat diskreettejä kuvailevia arvoja (kuten koira tai kissa), joilla ei ole selvää järjestystä. Malli, joka ennustaa kvalitatiivisia arvoja on luokittelija (eng. *classifier*).

2.3 Hypoteesit

On mahdollista luoda useita eri malleja, jotka päätyvät samoja ominaisuusparametrejä käyttäen samoihin nimiketuloksiin, mutta eri lasku- ja valintamenetelmiä hyödyntäen. Koko tätä mahdollisten mallien joukkoa A. Jung [5, s. 15-16, 18] nimittää hypoteesiavaruudeksi $\mathbf{H}$, josta yksi valittava malli on laskentafunktio h. Matemaattisesti ilmaistuna [5, s. 16]:

$$\mathbf{H}^{(n)} := \{h^{(w)} : \mathbb{R}^n \longrightarrow \mathbb{R} : h^{(w)}(x) = x^T w\}, \tag{2.1}$$

jossa n kuvaa ominaisuuksien määrää, $h^{(w)}$ valittua mallia painotusvektorilla w ja x ominaisuusvektoria.

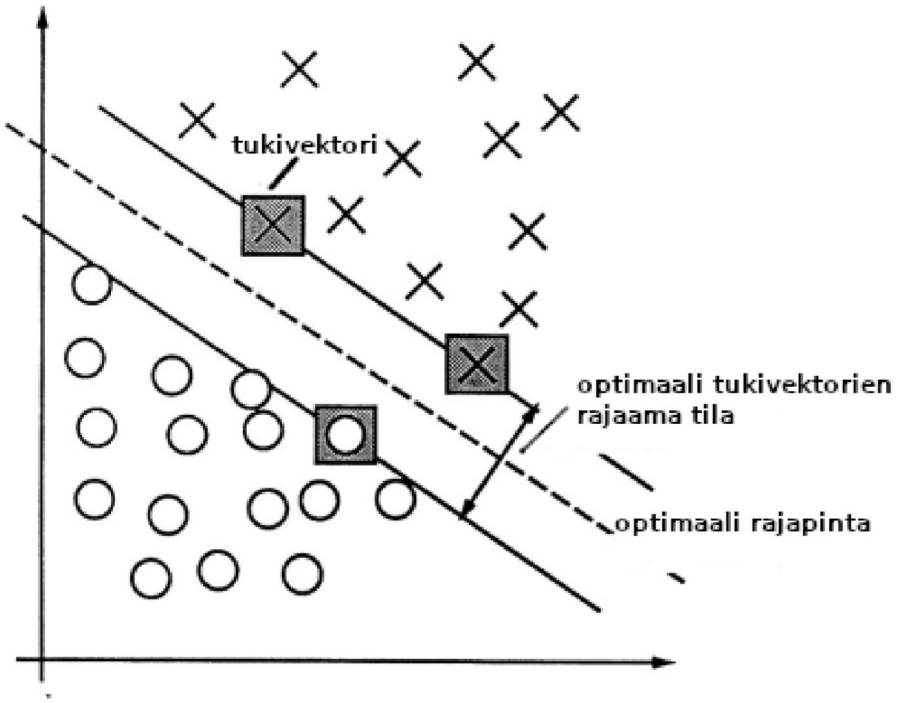
Yllä olevaa kaavaa käytetään usein, kun on mahdollista käyttää lineaarialgebraa ja ominaisuusvektori koostuu reaaliluvuista. Tämä on myös keskiverto tietokoneen kannalta laskennallisesti toteutettavissa.

Toisin sanoen etsitään painovektori w, jolla kerrottuna ominaisuusvektorin x transpoosi saadaan muunnettua nimikearvoennusteeksi $\hat{y}$. Mitä optimaalimpi sovitus mallissa on, sitä pienempi ero ε on ennustetun ja varsinaisen arvon välillä eli $y \approx \hat{y}$.

Vaikka vektorilaskenta on tavallisin tapa esittää ja analysoida dataa, on silti mainittava, että valittava menetelmä on sovelluskohtaista. [5, s. 20-23] Muita mahdollisia mallinnusmenetelmiä lineaarifunktioiden lisäksi ovat muun muassa polynomifunktiot.

2.4 Support Vector Machine eli SVM

SVM on eräs koneoppimisalgoritmi, jota C. Cortes ja V. Vapnik esittelivät tarkemmin artikkelissaan *Support-Vector Networks* (1995) [7]. Algoritmin periaatteena on siirtää ominaisuusvektorin arvot korkeampaan dimensioon epälineaarisesti, jolloin eri luokkien välillä voidaan muodostaa lineaariset rajapinnat.[7]. Luokittelu pyritään tekemään niin, että muutama piste eli tukivektori (eng. *support vector*) määrittää erottavan rajapinnan luokkien välille (kuva 2.1). Rajapinta on optimaali, kun tukivektorien rajaaman tilan väliin ei jää pisteitä ja pisteiden etäisyys on suurin mahdollinen.



Kuva 2.1. *SVM-periaatetta havainnollistava kuva (muokattu alkuperäisestä) [7]*

Laskentaprosessia voidaan optimoida käyttämällä niin kutsuttua *kernel trickiä*. Kernel on funktio, joka palauttaa saman arvon, joka saataisiin laskemalla korkeampaan dimensioon muunnettujen ominaisuusvektorien pistetulo [8]. Funktio ei kuitenkaan laske korkeamman dimension vektoreita eikä niiden pistetuloa, vaan hyödyntää laskemismenetelmiä, jotka ovat usein huomattavasti tehokkaampia. Voidaan tarkastella seuraavaa:

$$x = (x_1, x_2)$$
$$z = (z_1, z_2) \tag{2.2}$$
$$\Phi(x) = x^2 \tag{2.3}$$
$$K(x, z) = \langle \Phi(x), \Phi(z) \rangle \iff \langle x, z \rangle^2 \tag{2.4}$$

Esimerkkikaavassa (2.2) ovat ominaisuusvektorit ja (2.3) on ominaisuusvektorien muunnosfunktio. Kohta (2.4) havainnollistaa, kuinka kernel-funktiossa muunnettujen vektorien pistetulo voidaan korvata vektorien pistetulon muunnoksella; tässä tapauksessa neliöimällä. [8]

Molemmissa prosesseissa siis saadaan lopulta sama skalaariarvo tulokseksi, jota voidaan käyttää ominaisuusvektoreita vastaavien datapisteiden luokitteluun. *Kernel trick* tulee hyödyllisemmäksi, kun ominaisuusvektoreiden dimensio kasvaa. Kerneleitä voidaan skaalata hyperparametreilla, jotta niiden antamat tulokset olisivat paremmin sovellettavissa käytännön luokitteluongelmiin.

3 ETURAUHASSYÖPÄ

3.1 Yleistä

Eturauhassyöpä on yleisimmin esiintyvä syöpä miehillä. Suomessa tapauksia todetaan 5000 vuosittain. Eturauhassyöpään sairastuneiden keski-ikä Suomessa on 70 vuotta eli se esiintyy useimmiten vasta vanhemmalla iällä. Tästä johtuen potilaat eivät välttämättä menehdy itse syöpään vaan muihin ikään liittyviin komplikaatioihin. [2] Eturauhassyöpä on kuitenkin toiseksi yleisin kuolemaan johtava syöpä Suomessa [9].

Terveyskirjaston sivuilla [2] kerrotaan syövän oireilevan yleisesti virtsausongelmina ja verijääminä virtsassa tai siemennesteessä eli niin kutsuttuina urologisina oireina. Mikäli syöpä muodostaa etäpesäkkeitä eli metastaaseja esimerkiksi luustoon, se voi oireilla luustokipuna ja luiden haurautena.

Vaikka kaikkia vaikuttavia tekijöitä eturauhassyövän kehittymisessä ei tunnetakaan, androgeeneillä eli miessukupuolihormoneilla tiedetään olevan suuri merkitys siinä [10, Anatomiaa ja fysiologiaa]. Androgeenit ovat muutenkin merkittäviä eturauhasen toiminnan kannalta. Ne sitoutuvat andgrogeenireseptoreihin ja aktivoivat spesifejä geenejä. [9]

Taudinehkäisystä ei olla täysin varmoja, mutta muun muassa runsas rasvan ja lihan kulutus [9] sekä ylipaino lisäävät riskiä. On myös mahdollista, että soijan ja tomaatin kulutus taas vähentäisivät riskiä. Arviolta 2-5 % eturauhassyövistä on perinnöllisten tekijöiden aiheuttamia. [2]

Yleensä eturauhassyöpä diagnosoidaan vasta etäpesäkevaiheessa, sillä leviämättömän syövän oireilu on vähäistä. Diagnostisten jatkotutkimusten tarvetta arvioitaessa huomioidaan muun muassa potilaan ikä ja PSA-arvo (eng. *Prostate Specific Antigen*), sillä näiden avulla voidaan arvioida potilaan riskiryhmä. [11]

3.2 Hoito

Eturauhassyöpä ja sen hoito vaihtelee yksilöittäin. Yksi merkittävä tekijä hoitoa suunniteltaessa on syövän levinneisyys. Tyypillisesti päätöksenteossa hyödynnetään syövän TNM-luokitusta (engl. *tumor, node, metastasis*), joka kuvaa syövän levinneisyyttä eturauhasessa, imusolmukkeissa ja muualla kehossa. Valittuun hoitoon vaikuttavat myös potilaan eliniän ennuste diagnosointivaiheessa, oireet, PSA-arvo sekä Gleason-arvo.
[12][Eturauhassyöpä > Hoito]

Esimerkiksi lievässä tapauksessa voidaan päätyä pelkästään seuraamaan syövän kehitystä, mutta vakavammissa tapauksissa voidaan suorittaa leikkaus tai antaa sädehoitoa ulkoisesti tai sisäisesti. Iäkkäillä ja huonokuntoisilla potilailla voidaan antaa muun muassa GnRH-agonistihoitoa tai kirurginen kastraatio, jotta oireet lievenisivät. Tällöin annettava palliatiivinen hoito ei paranna syöpää, mutta tähtää pidentämään potilaan elinikää ja parantamaan elämänlaatua. Tässä tapauksessa syöpä on jo metastasoitunut.
[12][Eturauhassyöpä > Hoito][13]

Kastraatio eli androgeenideprivaatio voidaan tehdä poistamalla kivekset (orkiektomia), sillä eturauhassyöpä tarvitsee kasvaakseen testosteronia, jota kivekset tuottavat. Orkiektomialle ei kuitenkaan ole aina tarvetta, sillä nykyään lääkkeillä voidaan tehdä kemiallinen kastraatio.[11][Eturauhassyövän endokriininen hoito]

Kastraatio lieventää oireita tyypillisesti 2-3 vuotta, kunnes syöpä kehittyy kastraatioresistentiksi. Tällöin aiemmat hoidot eivät enää estä eturauhassyövän kasvua, vaikka testosteronitasot olisivatkin matalat (< 1.73 nmol/l)[12][Eturauhassyöpä > Hoito > Kastraatioresistentin eturauhassyövän hoito]. Tässä vaiheessa annetun hoidon päätarkoitus on lievittää oireita sekä parantaa elämänlaatua. Potilaiden haitallisin oire on kipu, jota voidaan lievittää muun muassa kipulääkkeillä ja sädehoidolla. Hoitona voidaan myös antaa solunsalpaajia, mutta niiden käytössä voi esiintyä usein vaikea-asteisia haittavaikutuksia.

Hoidossa voidaan ottaa käyttöön esimerkiksi entsalutamidia tai abirateronia sisältäviä lääkkeitä. Niiden on osoitettu olevan toimivia ja pidentävän elinikää 3-5 kuukautta.[13] Entsalutamidi estää eturauhassolujen androgeenireseptorien viestinnän [14][Hae: XTANDI] (engl. *androgen receptor signaling inhibitor*; ARSI) ja abirateroni vähentää testosteronin tuotantoa [15][Hae: ZYTIGA]. Kasvainsoluissa käynnistyy ohjelmoitu solukuolema eli apoptoosi, kun sen dihydrotestosteronin (androgeeni) määrä soluissa pienenee 80%[12][Eturauhassyöpä > Hoito]

W. Abida et al. tutkivat artikkelissaan [3] nimenomaan edellä esiteltyjen hormonihoitojen vaikutusta potilaiden veriarvoihin ja elinikään, kun syöpä oli metastaattinen ja resistentti kastraatiohoidoille. Tutkimuksessa keskityttiin tilastollisiin yhteyksiin potilaan genomin ja hoidosta saadun vasteen välillä. Tutkimuksessa huomattiin muun muassa, että SPOP-geenissä tapahtunut muutos pidensi ja RB1-muutos taas lyhensi aikaväliä, jolla hoidot olivat toimivia potilaan kohdalla.

3.3 Genetiikan merkitys

Syöpä on geneettinen sairaus. Solu alkaa muuntua syöpäsoluksi, kun solun kasvua edistävissä (onkogeenit) tai sitä rajoittavissa (kasvunrajoitegeenit) geeneissä tapahtuu merkittäviä mutaatioita. Mutaatioita voi ilmaantua henkilön genomiin esimerkiksi ympäristötekijöiden, virusten tai periytymisen kautta. Mutatoitunut solu ei enää käyttäydy normaalisti, vaan monistuu rajattomasti, tunkeutuu ympäröivään kudokseen, ei reagoi normaalin kudoksen signalointiin ja jatkaa leviämistä. [16]. On siis luonnollista olettaa, että tutkimalla potilaan genetiikkaa voidaan saada tarkempaa tietoa syövän laadusta.

W. Abida et al. huomasivat tutkimuksessaan [3] RB1-, AR- sekä TP53-geenien vaikuttaneen potilaiden hoidon tulokseen.

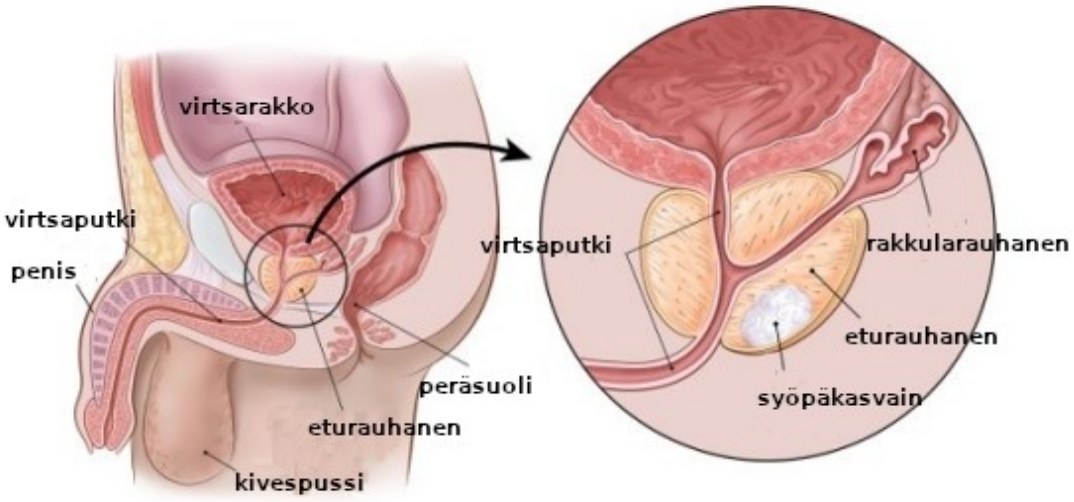
Ensimmäinen geeni, RB1, on ensimmäisenä löydetty kasvainten kehitystä estävä geeni (eng. *tumor suppressor gene*). Sen koodaama proteiini säätelee negatiivisesti solusykliä ja auttaa ylläpitämään kromatiinirakennetta. Se on tunnettu syöpägeeni.[17][Gene > Hae: RB1 Homo sapiens]

TP53 on myös kasvainten kehitystä estävä geeni. Geenin koodaama proteiini säätelee solun eri geenien ekspressiota ja siten vaikuttaa muun muassa solusykliin ja apoptoosiin. Kyseisen geenin mutaatiot tunnetusti liittyvät moniin syöpiin. [17][Gene > Hae: TP53 Homo sapiens]

Kolmas geeni, AR, on androgeenireseptoreita koodaava geeni. Sitoutuessaan hormoni, esimerkiksi testosteroni, saa reseptorin kulkeutumaan tumaan, jossa sen osat aktivoivat androgeeneille herkkien geenien transkriptiota aiheuttaen näin vasteen. Mutaatiot tässä geenissä voivat tähän vasteeseen henkilön elimistössä. [17][Gene > Hae: AR Homo sapiens]

3.4 Eturauhasen anatomia ja fysiologia

Eturauhanen on lantion pohjassa virtsarakon alapuolella sijaitseva elin, jonka päätehtävä on tuottaa ja varastoida siemennestettä. Siemennestettä tarvitaan ravitsemaan siittiöitä sekä parantamaan niiden liikkuvuutta. Rauhanen kehittyy ja kasvaa miehen eliniän aikana androgeenien vaikutuksesta, mutta samaiset hormonit vaikuttavat myös eturauhassyövän muodostumiseen. [10, Anatomiaa ja fysiologiaa]



Kuva 3.1. *Eturauhasen (eng. prostate, lat. prostata) sijainti kehossa. Nähtävissä myös perifeerinen vyöhyke, johon syöpä useimmiten kehittyy [10, Anatomiaa ja fysiologiaa]. (alkuperäistä kuvaa muokattu [18])*

Eturauhanen erittää myös prostataspesifistä antigeeniä eli PSA:ta. Tämän entsyymin tehtävä on saada siemenneste juoksevammaksi, mutta sen määrää seerumissa käytetään diagnosoidessa eturauhassyöpää. [10, Anatomiaa ja fysiologiaa] [11, s. 4]

4 TUTKIMUS

4.1 Aineisto

W. Abida et al. tutkivat ja taltioivat artikkelissaan [3] geneettisen profiilin, histologian ja kliiniset tulokset 429 potilaalta, joiden eturauhassyöpä oli edennyt metastaattiseen kastraatioresistenttiin vaiheeseen. 128 potilasta hoidettiin ARSI-lääkityksellä ja heiltä tutkittiin 18 toistuvan geenimuutoksen vaikutusta kliinisiin tuloksiin. Ryhmä löysi yhteyden RB1-geenin ja huonon hoitovasteen välillä. Myös TP53 ja AR olivat mahdollisesti vasteeseen vaikuttavia geneettisiä tekijöitä. Tutkimuksen data on saatavilla githubissa [19]. Tässä työssä hyödynnetyt datatiedostot ovat: `data_clinical_patient.txt`, `data_clinical_sample.txt` ja `data_CNA.txt`. Ensimmäinen tiedosto sisältää tutkittujen potilaiden kliinisiä tietoja, kuten diagnoosi-ikä, hoitomuoto, elinkuukaudet (engl. *overall survival months*) hoidolla sekä yksilöivä tunnus. Tiedosto `data_CNA.txt` sisältää tiedon eri näytteiden geenianalyysissä löytyneet kopiolukumuutokset. Tässä työssä tutkittiin vain kolmea geeniä: RB1, AR sekä TP53. Tiedostosta `data_clinical_sample.txt` poimittiin vain näytteet potilaisiin liittävät tunnukset, mutta kyseisessä tiedostossa olisi ollut myös muun muassa näytteiden Gleason-arvot. Lääkkeiden potilaissa aiheuttama hoitovaste luokiteltiin joko hyvään tai huonoon vasteeseen hoidon aloituksen jälkeisten elinkuukausien määrän mukaan. Tässä työssä hoitovastetta pidettiin hyvänä, mikäli elinkuukaudet ylittivät 6 kuukautta, muussa tapauksessa vaste oli huono.

	Ominaisuusavaruus $\mathbf{X}$	**Nimikeavaruus** $\mathbf{Y}$
Arvot	$\langle$ RB1, TP53, AR $\rangle$	$\langle$ -1, 1 $\rangle$
Selite	Potilaalla ilmenevä kyseisen geenin kopioluku.	ARSI-hoidon hoitovaste: huono (-1) mikäli elinikä oli alle tai yhtäsuuri kuin 6 kuukautta, muulloin hyvä (1).
Dimensiot	3	1
Potilastietueesta siis kerätään 3-ulotteinen ominaisuusvektori geenien kopioluvuista, jonka pohjalta määritetään potilasta vastaava 1-ulotteinen nimike. Tutkimuksella pyritään arvioimaan näiden pohjalta muodostettavan koneoppimismallin laatua sekä mahdollisuutta kehittää mallia käsittelemällä dataa ennen mallin koulutusta.		

Taulukko 4.1. *Havainnollistava taulukko, jossa on nähtävissä potilasdatasta poimittavat ominaisuudet ja näiden pohjalta ennustettava nimikearvo.*

4.2 Naiivin oppimisen välttäminen

Tarkasteltaessa käytössä olevaa dataa tarkemmin huomataan nopeasti yleinen ongelma koneoppimisessa: epätasapainoinen datajakauma [20]. Datasetistä vain noin 13% ($\mathrm{n}_{\mathrm{positiiviset}} = 111, \mathrm{n}_{\mathrm{negatiiviset}} = 17$), alittaa 6 kuukauden rajan siten lukeutuen huonoihin vasteisiin. Toisin sanoen, datassa on huomattavasti enemmän hyvään vasteluokkaan kuuluvia pisteitä suhteessa huonoon vasteeseen kuuluviin. Tällaisen datasetin pohjalta koulutettu malli oppii naiiviksi. Tämä tarkoittaa, että malli ennustaa aina samaa arvoa, joka tässä tapauksessa on hyvän vasteen arvo eli 1. Huonoa vastetta vastavasti merkattaisiin arvolla -1. Mikäli tarkkuus lasketaan suhteuttamalla oikeat ennusteet kaikkiin ennusteisiin, malli on näennäisesti hyvä. Koska vain murto-osa pisteistä kuuluu toiseen luokkaan, saa malli kuitenkin noin 87% tarkkuuden ennustaessaan aina vain 1. Usein tällaisissa tapauksissa huomataankin, että enemmistöluokan ennustukset ovat lähes 100% oikein, kun taas vähemmistöluokan ennusteista vain 0-10% on oikein ennustettuja [20]. Täten yleinen tarkkuus ei annakaan todellista tarkkuusarvoa.

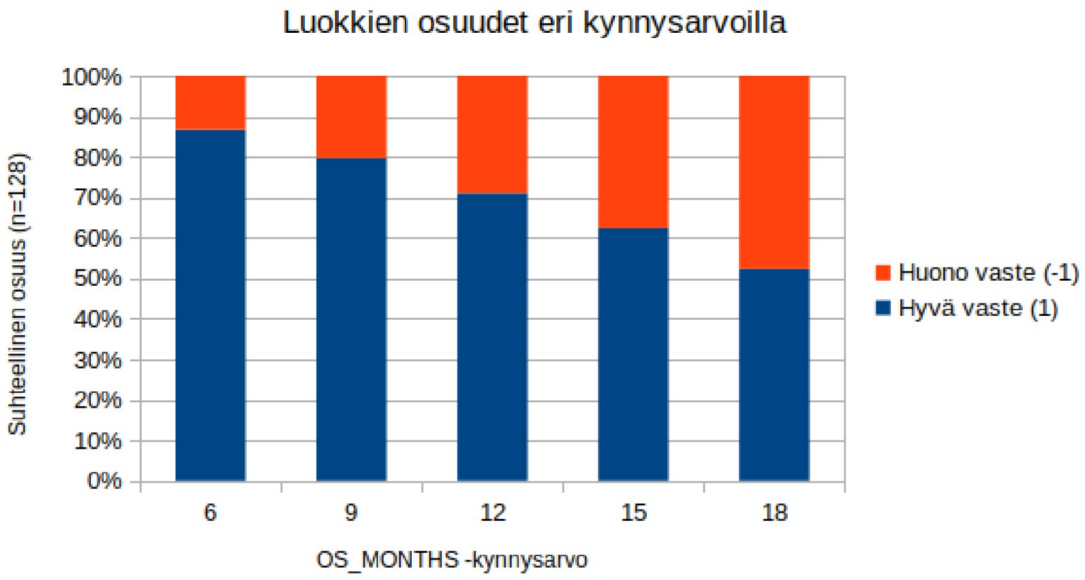
SVM-algoritmit eivät ole niin herkkiä datan epätasapainoisuudelle johtuen niiden tavasta määritellä luokittelurajat. Kuitenkin epätasapainoa ei täysin saada kompensoitua. Mallin oppiminen pohjautuu totaalivirheen minimointiin. [21][20]

H. Haibo et al. mainitsevat artikkelissaan [20] useita tapoja pyrkiä poistamaan epätasapainon vaikutus. Näitä ovat muun muassa satunnainen alinäytteistys, jossa poistetaan x-määrä enemmistöluokan $\mathrm{S}_{\mathrm{major}}$ edustajista, satunnainen ylinäytteistys, jossa replikoidaan x-määrä vähemmistöluokan $\mathrm{S}_{\mathrm{minor}}$ edustajista, SMOT (eng. *Synthetic Minority Over-*

sampling Technique), jossa generoidaan synteettisesti uusia datapisteitä $\mathrm{S}_{\mathrm{minor}}$ pohjalta, sekä eriävä painotus luokitteluvirheille eli esimerkiksi väärinluokiteltu vähemmistöluokan edustaja ohjaa koulutuksen suuntaa enemmän kuin enemmistöluokan edustajan luokan väärinluokittelu.

Tässä tutkimuksessa keskityttiin satunnaiseen ali- ja ylinäytteistykseen. Alinäytteistyksen haittana on, että silloin mahdollisesti tärkeää tietoa sisältävä osuus enemmistöluokan datasta jää käyttämättä. Ylinäytteistys taas voi laittaa liikaa painoa tiettyjen pisteiden tietyille ominaisuuksille, mikä voi johtaa ylisovitukseen. [20] On aina syytä huomioida mahdolliset seuraukset, kun lähdetään manuaalisesti muokkaamaan mitattua dataa ennen algoritmille syöttöä.

Kuvasta 4.1 huomataan, kuinka kynnysarvon valinta vaikuttaa luokkien jakaumaan. Esimerkiksi 6 kuukauden arvolla vain noin 13% datasta kuuluu huonoon vasteeseen.



Kuva 4.1. *Luokkien jakauma riippuen vasteluokittelun kynnysarvosta*

On selvää, että epätasapainon suuruus vaikuttaa merkittävästi mallin naiiviuteen. Googlen kehittäjäsivuilla [22] epätasapainoisuudelle listataan kolme eri luokkaa pohjautuen $\mathrm{S}_{\mathrm{minor}}$ osuuteen koko datasetistä S:

- kevyt (20-40%)
- kohtalainen (1-20%)
- suuri (<1%)

Tässä tutkimuksessa päädyttiin mittaamaan SVM-mallin tarkkuutta, kun datajakaumaa muokataan yli- ja alinäytteistyksellä.

4.3 Menetelmät

Tutkimusta varten hyödynnettiin Pythonille saatavaa scikit-learn -kirjastoa (kirjastoa esittelevä artikkeli: [23], kirjaston dokumentaatio: [24]), joka tarjoaa monta valmiiksi implementoitua koneoppimismallia, työkaluja datan käsittelyyn sekä runsaasti dokumentaatiota. Tutkimusmenetelmät ovat sovellusta G. Raulin et G. Moncecchin scikit-learn -käsikirjasta [25], koneoppimisen teoriasta sekä tilastomatematiikasta.

Kirjasto `sklearn.svm` tarjoaa luokittelutehtäviä varten erillisen kirjaston: `sklearn.svm.svc` [24][User Guide > Supervised Learning > Support Vector Machines > Classification] (eng. *support vector classifier*). SVM-metodeilla on hyperparametri `kernel`, jonka avulla voidaan säätää, millä tavalla algoritmi siirtää datapisteiden ominaisuusvektorit korkeampaan dimensioon. Tämä käsittelytapa mahdollistaa pisteiden erottelun lineaarisella rajalla, vaikka alemmissa dimensioissa tämä ei olisikaan mahdollista.[5, s. 38–39] Valinta vaikuttaa saataviin rajapintoihin sekä laskentatehoon, mutta tässä tutkimuksessa ei muokattu oletusarvoja.

Tässä tutkimuksessa kirjoitettu koodi pyrkii opettelemaan ennustamaan hoitovastetta potilaalle kolmen geenin (RB1, TP53, AR) kopiolukujen avulla. Tutkimuksessa keskityttiin enimmäkseen datan esikäsittelyn (ali- ja ylinäytteistys) kykyyn parantaa ennusteen tuloksia. Tästä johtuen koulutettava malli oli ainoastaan SVM-luokittelija-algoritmi (engl. *support vector classifier*, SVC).

On syytä huomioida, että dataa ei suoraan voitu käyttää sellaisenaan. Tutkimukseen alun perin osallistuneesta 429 potilaasta vain 128:lle oli merkitty OS_MONTHS-arvo (elinkuukaudet). Käyttöä varten tiedostojen datasta suodatettiin pois tietueet, joissa oli tämän työn kannalta puutteellista tietoa. Myöskin elinkuukausiarvo muutettiin jokaisen potilastietueen kohdalla joko arvoon 1 tai -1 hoitovasteen laadun mukaisesti.

Tässä työssä osuus datasta, joka jätetään sivuun mallin testausta varten määritetään luvun `fold_ratio` ($0 < fold\_ratio < 1$) mukaiseksi.

Testausprosessi oli seuraavanlainen (koodit on kuvattu tarkemmin luvussa 4.3.2):

1. Ladataan ja esikäsitellään data `read_data.py`-tiedoston funktioilla.
2. Yli- ja alinäytteistetään $1 - fold\_ratio$ datasta (`machine_learning_models.py`), jätetään $fold\_ratio$ testidataksi.
3. Koulutetaan SVC, ajetaan testidata sen läpi, lasketaan indeksit, tallennetaan tulos (`machine_learning_models.py`).

4.3.1 Laskettavat indeksit

Haibo H. et al. kuvaavat artikkelissaan [20] kahta indeksiä, tarkkuus (eng. *precision*) ja jäljitys (eng. *recall*), kvantifioidessaan koneoppimismallin toimivuutta.

Tarkkuus voidaan laskea kaavalla:

$$Precision = \frac{TP}{TP + FP}, \tag{4.1}$$

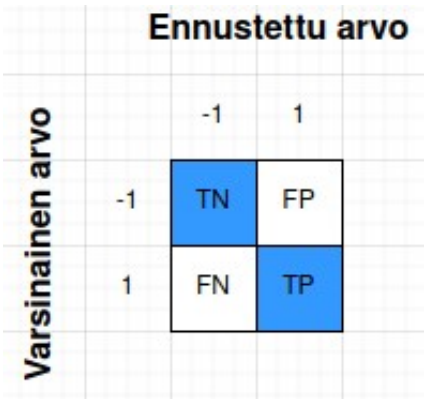
jossa TP on todelliset positiiviset arvot ja FP väärin ennustetut positiiviset arvot. Näin laskettu suhde kuvaa, kuinka luotettava mallin antama positiivinen ennuste on. Suuri arvo merkitsee, että suuri osa ennustetuista positiivisista arvoista todella vastaa positiivista datapistettä, kun taas pieni arvo merkitsisi, että suurin osa ennustetuista positiivisista arvoista onkin todellisuudessa negatiivisia. Vastaavassa indeksissä negatiiviselle arvolle TP korvautuu TN-arvolla (todelliset negatiiviset) ja FP FN-arvolla (väärät negatiiviset).

Jäljitystä kuvaa kaava:

$$Recall = \frac{TP}{TP + FN}, \tag{4.2}$$

joka kuvaa mallin kykyä löytää kaikki tietyn luokan edustajat luokitellessaan datapisteitä. Oikein menneet positiiviset ennusteet suhteutetaan todellisten positiivisten ja väärien negatiivisten (eli todellisuudessa positiivisten) ennusteiden lukumäärien summaan. Suuri arvo kuvaa, että malli seuloo tarkasteltavat luokat hyvin datan joukosta, kun taas pieni arvo viittaisi huonoon seulontakykyyn. Indeksi ei siis varsinaisesti ota kantaa itse luokittelijan tarkkuuteen.

Visualisoinnissa käytettiin konfuusiomatriisia. Konfuusiomatriisista on helppo katsoa $TP,\ FP,\ TN,\ FN$ -arvojen lukumäärää mallin tekemistä luokituksissa. Matriisin koko on $N \times N$, jossa N on eri luokkien lukumäärä. Matriisin alkio $C_{i,j}$ kuvaa luokkaan i kuuluvien, luokaksi j ennustettujen pisteiden lukumäärää. Tässä mallissa -1 vastaa negatiivisen vasteen luokkaa ja 1 positiivisen vasteen luokkaa. Esimerkiksi alkio $C_{-1,1}$ kuvaa oikeasti negatiivisten, mutta positiiviseksi ennustettujen pisteiden lukumäärää eli toisin sanoen FP-arvoa. [24, sklearn.metrics.confusion_matrix]



Kuva 4.2. *Kaksiluokkaisen luokitteluongelman* 2×2 *-konfuusiomatriisi*

4.3.2 Koodit

Tutkimusta varten kirjoitettiin kaksi Python-kooditiedostoa. Tiedostot sisältävät funktioita, joiden avulla kliininen ja geneettinen data esikäsiteltiin. Päätehtävät koodilla olivat eritellä kiinnostavien geenien (RB1, TP53 ja AR) CNA-arvot ominaisuusvektoriin ja potilaiden selviytymiskuukausien lukumäärä nimikevektoriin, yli- ja alinäytteistää dataa samalla luoden kompensoidut datavektorit sekä kouluttaa koneoppimismallit (SVC).

Ohjelma koostuu kahdesta kooditiedostosta: `read_data.py` (liite 6) ja `machine_learning_models.py` (liite 6). Alla tiivistetty selitys koodien toiminnasta:

Ensimmäinen tiedosto, `read_data.py`, sisältää datan lukemiseen ja käsittelyyn liittyvät funktiomääritelmät. Funktio `read_data_into_matrix` (rivi 24) lukee läpi datatiedostot , poimii olennaiset arvot ja koostaa ne yhdeksi matriisiksi. Funktiot `select_arsi_patients` ja `os_months_to_binary` (rivit 67 ja 86) poimivat tarkasteltavat geenit ja elinkuukaudet (`OS_MONTHS`) riveiltä, joissa kyseinen data on saatavilla. Rivien nimike muutetaan binääriarvoiseksi siten, että `TRESHOLD` vakion arvon ylittävät `OS_MONTHS`-arvot merkitään positiiviseksi eli hyväksi (1) ja muut negatiiviseksi eli huonoksi (-1). Funktio `init_osm_data` (rivi 102) koostaa edelliset yhdeksi kutsuksi ja palauttaa datan yhdessä ominaisuus- ja yhdessä nimikevektorissa.

Jälkimmäinen tiedosto, `machine_learning_models.py`, sisältää funktiot koneoppimisalgoritmeille, ali- ja ylinäytteistykselle, tulosten laskemiselle ja tallentamiselle.

Funktio `undersample` (rivi 23) alinäytteistää ominaisuus- ja nimikevektorit siten, että uusien vektoreiden koosta parametrin `ratio` kokoinen osuus on majoriteettiluokan pisteitä. Tämä saadaan johdetulla kaavalla:

$$\begin{aligned} ratio * S_2 &= ratio * (S_1 - x) = S_{maj} - x \\ x &= \frac{S_{maj} - ratio * S_1}{1 - ratio}, \end{aligned} \tag{4.3}$$

jossa x on poistettavien majoriteettipisteiden määrä, S_1 kaikkien pisteiden alkuperäinen määrä ja S_2 niiden uusi määrä. Funktio valitsee satunnaisesti majoriteettipisteitä vektoreista kunnes uusi lukumäärä saavutetaan.

Funktio `oversample` (rivi 54) ylinäytteistää vektorit (`x,y`) siten, että palautettujen vektoreiden uudesta koosta parametrin `ratio` kokoinen osuus on minoriteettiluokkaa. Tämä saadaan johdetulla kaavalla:

$$\begin{aligned} ratio * S_2 &= ratio * (S_1 + x) = S_{min} + x \\ x &= \frac{ratio * S_1 - S_{min}}{1 - ratio}, \end{aligned} \tag{4.4}$$

jossa x on lisättävien minoriteettipisteiden määrä ja S_1 ja S_2 ovat vastaavasti alkuperäinen ja uusi minoriteettipisteiden kokonaismäärä vektorissa. Funktion toteutuksessa pisteitä replikoidaan satunnaisesti, kunnes haluttu lukumäärä saavutetaan.

Funktio `split_data` (rivi 132) jakaa datan koulutus- ja testisetteihin. Parametri `ratio_fold` määrittää, kuinka suuri osuus datasta otetaan sivuun testisetiksi (sama kuin vakio `FOLD_RATIO`). Funktio määrittää datan luonnollisen jakauman pohjalta yksittäisen testisetin minoriteetti- ja majoriteettipisteiden määrän. Näin varmistetaan, että testisetin datajakauma noudattaa alkuperäistä jakaumaa.

Funktiot `get_scores` ja `record_scores` (rivit 88 ja 112) keräävät konfuusiomatriisin senhetkisestä testistä, laskee TN, FP, FN, TP -arvojen pohjalta indeksiluvut ja tallentaa ne tiedostoon myöhempää tarkastelua varten.

Funktio `test_svc` (rivi 180) sitoo edellä esitellyt funktiot yhdeksi kokonaisuudeksi. Sille syötetään ominaisuus- ja nimikevektorit. Testi-koulutus -jakoa voidaan muokata muuttamalla vakion `FOLD_RATIO` arvoa. Funktio kouluttaa ja testaa luokittelija-algoritmia ite-

roiden eri suhdelukuarvoilla (`ratio`). Pääohjelmaa ei ole, vaan koulutus ja testaaminen suoritettiin Python-komentotulkissa ladaten esitellyt tiedostot, lukemalla data vektoreihin, testaamalla mallia kolmella eri `FOLD_RATIO`-arvolla $(0.2, 0.3, 0.4)$.

4.4 Tulosten keräys

Opittujen mallien hyvyyttä ja datan kompensoinnin onnistuneisuutta pyrittiin arvioimaan laskemalla tarkkuus- ja jäljitettävyysindeksit (kaavat 4.1, 4.2) mallin ennustamille arvoille, kun sille syötettiin testidataa. Tulosten hahmotusta varten saadut indeksit taulukoitiin ja ennusteiden pohjalta määritettiin erinäiset konfuusiomatriisit.

Aluksi koko datasetti jaettiin kahteen osaan: koulutusdataan ja testidataan. Testidataa ei muokattu. Koulutusdataa vastaavasti yli- ja alinäytteistettiin eri suhdelukuparametreilla {0.4, 0.5, 0.6}, minkä jälkeen SVC koulutettiin sillä. Testaus sisälsi myös yhden kierroksen, jossa dataa ei muokattu lainkaan etukäteen (luonnollinen jakauma). Tämän jälkeen testidata syötettiin koulutetulle mallille ja sen ennustamia tuloksia vertailtiin datapisteiden todellisiin arvoihin. Tämä prosessi toistettiin kolmesti eri *fold_ratio* -parametrein {0.2, 0.3, 0.4}. Kyseinen parametri määrittää, kuinka suuri osuus datasetistä otetaan sivuun testidataksi.

4.5 Luotettavuus

Edellä kuvatut menetelmät mittaavat ali- ja ylinäytteistysmuokkausten toimivuuden datan käsittelyssä ja epätasapainon kompensoinnissa. On kuitenkin syytä huomioida, että tämä menetelmä ei kata läheskään kaikkia mahdollisia tapoja lähestyä tätä ongelmaa. Ensinnäkin, ali- ja ylinäytteistys ovat vain murto-osa eri menetelmistä kompensoida datan epätasapainoa. Muita menetelmiä ovat muun muassa synteettinen datan generointi, isomman datasetin kerääminen kentältä, eri luokkien edustajien erilaiset painotuskertoimet, hyperparametrien muokkaus (esimerkiksi SVM-mallin kohdalla oletusarvoinen `kernel` on "rbf") sekä ylipäänsä toisen algoritmin valitseminen. Toiseksi, datasetin melko pieni koko ja merkittävä epätasapaino korostavat satunnaisuuden roolia prosessissa. Olisi myös mahdollista säädellä tiukemmin esimerkiksi, mitkä pisteet datasta otetaan mihinkin settiin mukaan, mutta tämä ei välttämättä enää kuvaa reaalimaailman tilannetta yhtä tarkasti. Viimeiseksi, on syytä pohtia käytettyjen geenien CNA-tietojen suoraviivaista vaikutusta tutkittuun arvoon, eli syöpähoidon tehoon. W. Abida et al. eivät löytäneet suurta vaikutusta muiden kuin RB1-geenin kohdalla [3].

5 TULOKSET

5.1 Tutkimustulokset

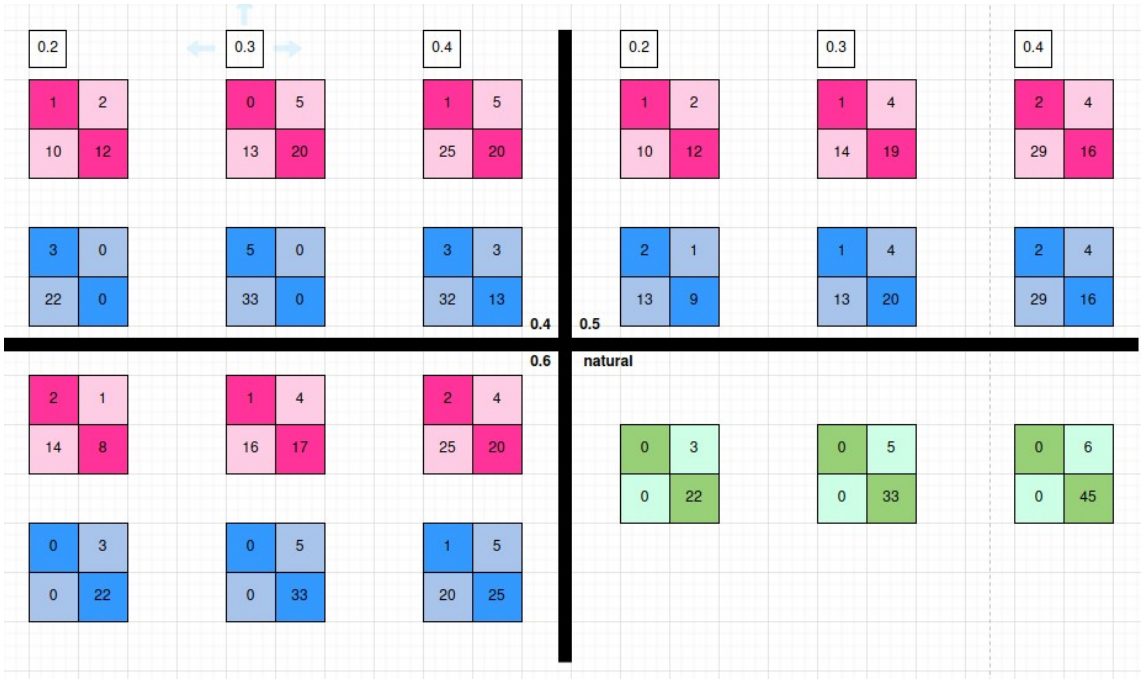
Taulukossa 5.1 on ilmaistuna kolmesta mittauskierroksesta peräisin olevat tunnusluvut. Fold ratio kuvaa, kuinka suuri osuus koko datasetistä otettiin erilleen testidataksi. Ratio kuvaa muokattavan dataluokan osuutta lopullisessa setissä, jolla SVM-malli koulutettiin, ja sampling kuvaa menetelmää, jolla datasettiä muokattiin. Loput neljä saraketta kuvaavat luokille saatuja avainlukuja tietyssä koetilanteessa. Eli kokoavasti esimerkiksi ensimmäisen fold ratio -lohkon viimeinen rivi kuvaa tilannetta, jossa datasetista 20% on otettu erilleen testidataksi, ja koulutusdatan majoriteettiluokkia on alinäytteistetty, kunnes niiden osuus lopullisessa setissä on ollut 60%. Testattaessa näin koulutettua mallia sivuun otetulla datasetillä positiivisten ennusteiden tarkkuus oli 88% ja jäljitysosuus 100%, kun taas negatiivisille ei voitu määrittää tarkkuutta ja jäljitysosuus oli 0%.

Fold ratio	Ratio	Sampling	Precision P	Precision N	Recall P	Recall N
0.2	natural	-	0.88	-	1.00	0.00
	0.4	over	0.86	0.09	0.55	0.33
		under	-	0.12	0.00	1.00
	0.5	over	0.86	0.09	0.55	0.33
		under	0.90	0.13	0.41	0.67
	0.6	over	0.89	0.13	0.36	0.67
		under	0.88	-	1.00	0.00
0.3	natural	-	0.87	-	1.00	0.00
	0.4	over	0.80	0.00	0.61	0.00
		under	-	0.13	0.00	1.00
	0.5	over	0.83	0.07	0.58	0.20
		under	0.83	0.07	0.61	0.20
	0.6	over	0.81	0.06	0.52	0.20
		under	0.87	-	1.00	0.00
0.4	natural	-	0.88	-	1.00	0.00
	0.4	over	0.80	0.04	0.44	0.17
		under	0.81	0.09	0.29	0.50
	0.5	over	0.80	0.06	0.36	0.33
		under	0.80	0.06	0.36	0.33
	0.6	over	0.83	0.07	0.44	0.33
		under	0.83	0.05	0.56	0.17

Kuva 5.1. *Precision- ja recall-tulokset eri over- ja undersampling-ratioilla*

Luvut, joiden pohjalta indeksit on laskettu, on koottu konfuusiomatriiseiksi kuvaan 5.2.

Ylimmät luvut kuvaavat sen alapuolella olevien matriisien fold ratiota, ja sisempien nurkkien luvut kuvaavat ratioita. Punaiset ja siniset matriisit ovat vastaavasti yli- ja alinäytteistettyjä, vihreät ovat muokkaamattomista seteistä. Matriisien lukujen merkitykset on kuvattu tarkemmin kohdassa Laskettavat indeksit.



Kuva 5.2. *Confusion matrix fold ratioilla ja ratioilla*

5.2 Johtopäätelmät

Tuloksista huomataan, että luonnollisella jakaumalla koulutettu malli ennusti aina vain positiivisia arvoja. Vaikka näin saadut tarkkuus- ja jäljitysarvot ovat näennäisesti hyviä, malli ei kuitenkaan ole kokonaisuudessaan hyvä naiiviuden vuoksi.

Muokattujen datasettien antamat tulokset ovat yleisesti ottaen parempia. Ennusteet ovat jakautuneet tasaisemmin, eikä malli ole jatkuvasti biasoitunut ennustamaan vain yhtä luokkaa. Kuitenkin alinäytteistys näyttäisi useammin aiheuttavan naiiviutta mallissa. Tuloksia tarkasteltaessa huomataan niin fold ration kuin rationkin vaikuttaneen saatuihin arvoihin.

Fold ratioista arvo 0.4 antoi tasaisimmat tulokset. Arvot 0.2 ja 0.3 paransivat myös tuloksia, mutta näissä tapauksissa ratioilla 0.4 ja 0.6 esiintyi biasoituvuutta alinäytteistetyissä seteissä. Fold ratio 0.4 oli ainoa, jolla mikään avainluvuista muokatuissa seteissä ei saanut arvoa 0.

Ratioista tasaisimpia tuloksia antoi arvo 0.5. Täydellistä optimaaliarvoa ei löytynyt, sillä parannettaessa jonkin avainluvun arvoa saadan alennettua toista. Kuitenkin 0.5 oli arvo, jolla jokaiselle avainluvulle löytyi jokaisessa fold ratio -lohkossa siedettävä positiivinen lukuarvo.

Tarkasteltaessa ainoastaan 0.5 ration tuottamia rivejä eri fold ratioilla huomataan kuitenkin, että arvoa 0.4 pienemmät fold ratiot tuottavat suurimmaksi osaksi parempia tuloksia avainluvuille tällä kyseisellä ratiolla. Fold ratio 0.2 antaa enimmäkseen paremmat tulokset kuin toiset fold ratiot.

Ei siis voida absoluuttisesti todeta tiettyjen jakaumien olevan toisia parempia, mutta tulokset tukevat ajatusta, että koulutusdatan tulisi olla mahdollisimman tasajakoinen luokkien välillä, ja kyseisen datasetin koon tulisi olla verrattain suuri testausdatasettiin nähden.

5.3 Pohdinta

Koulutusdatan muokkauksella pystyttiin selkeästi vaikuttamaan saatuun malliin. Malli ei enää yksipuoleisesti ennustanut vain positiivista vasteluokkaa eli siitä ei kehittynyt yhtä naiivia ennustinta. Parhaimmat tulokset saatiin, kun jakaumat tasattiin puoliksi kahden ennustettavan luokan suhteen.Tässä tutkimuksessa tutkittiin vain kahta datamanipulaatiomenetelmää. Muita mahdollisia menetelmiä on lukuisia. Esimerkiksi synteettinen datan generointi (SMOTE) on mielenkiintoinen vaihtoehto [20].

Yleisesti koneoppimismallien optimointi tapahtuu teoreettisen analyysin ja käytännön kokeilun kautta. Tutkimalla dataa voidaan päätellä ja mallintaa eri algoritmien sopivuutta kyseiseen ongelmaan. Myöskin testaamalla voidaan säätää algoritmin hyperparametreja kuten käytettävää kernel-metodia SVM-algoritmien tapauksessa. Kiinnostavaa olisikin lähteä tutkimaan, kuinka eri algoritmit eri parametreilla käyttäytyisivät, kun niille syötetään manipuloitua dataa. Vaihtoehtoisesti voitaisiin keskittyä vain yhteen malliin ja pyrkiä optimoimaan hyperparametrit. Tässä työssä pääongelma oli datan epätasapainon kompensointi. Tämä toteutettiin manipuloimalla dataa, mutta yhtä käypä ratkaisu olisi saattanut olla suuremman datasetin hankkiminen koulutusta varten. Vielä yksi mahdollinen lähestymistapa olisi ollut säätää `sklearn.svm`-kirjaston tarjoamaa luokkien painotusta [24]][User Guide > Supervised Learning > Support Vector Machines > Unbalanced Problems]. Näin voitaisiin korostaa minoriteettiluokan edustajien painoarvoa mallin koulutuksessa.

6 YHTEENVETO

Mallit koulutettiin eri tavoin ja eri suhdeluvuin muokatuilla dataseteillä. Mallien hyvyyttä vertailtaessa ei löydetty absoluuttisesti parasta jakauman suhdetta, mutta datan esikäsittely antoi selvästi parempia tuloksia verrattuna luonnollisella jakaumalla koulutettuun malliin. Tulokset siis viittaavat siihen, että tasajakoinen datasetti antaa parempia tuloksia koulutettaessa mallia. Myöskin koulutusdataa tulisi olla testidataa enemmän. Nämä tulokset tukevat aiempia havaintoja koneoppimisalgoritmeihin liittyen.

Työssä perehdyttiin tarkemmin koneoppimiseen, SVM-algoritmeihin sekä eturauhassyöpään.

Työn tuloksista ei pitäisi tehdä liikaa johtopäätöksiä siitä, onko koneoppiminen sopiva tapa ennustaa hoitovastetta. Valittu malli ja valittu data ovat vain murto-osa kaikista mahdollisista yhdistelmistä. Esimerkiksi eri algoritmin tai geenijoukon valinta olisi voinut antaa erilaiset tulokset. Kuitenkin tutkimuksen tulos auttaa alleviivaamaan haasteita ja epävarmuutta koneoppimisen soveltamisessa data-analyysissä. Työ voikin toimia kannustimena lääketieteellisten koneoppimissovellusten tutkimiselle ja kehittämiselle.

LÄHTEET

[1] Ascent of machine learning in medicine. *Nature Materials* 407.18 (2019). URL: https://doi.org/10.1038/s41563-019-0360-1.

[2] Terveyskirjasto : Eturauhassyöpä. *Duodecim* (2022). Artikkelitunnus: dlk00210 (008.011). URL: https://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=dlk00210 (viitattu 14. 04. 2022).

[3] W. Abida et al. Genomic correlates of clinical outcome in advanced prostate cancer. *Proceedings of the National Academy of Sciences* 116.23 (2019), 11428–11436. URL: https://doi.org/10.1073/pnas.1902651116 (viitattu 14. 04. 2022).

[4] Skycode Oy. *Mitä tekoäly on*. URL: https://tekoaly.info/mita_tekoaly_on (viitattu 14. 04. 2022).

[5] A. Jung. *"Machine Learning: The Basics"*. Springer, Singapore, 2022. URL: https://doi.org/10.48550/arXiv.1805.05052.

[6] T. Hastie et al. *The elements of statistical learning : data mining, inference, and prediction*. eng. 2. ed. Springer series in statistics. New York: Springer, 2009. ISBN: 978-0-387-84857-0.

[7] C. Cortes & V. Vapnik. Support-Vector Networks. *Machine Learning* 20.3 (1995), 273–297. DOI: 10.1023/A:1022627411411.

[8] G. Hackeling. *Mastering machine learning with scikit-learn : learning to implement and evaluate machine learning solutions with scikit-learn*. Second edition. Birmingham, England ; Packt, 2017, 168. ISBN: 1-78829-987-6.

[9] P. Taimen & P. Kujala. *Eturauhasen syöpä*. Hae: Duodecim Oppiportti > Patologia > Elinpatologia > Miehen sukuelimet > Eturauhanen. (2021). URL: https://www.oppiportti.fi/op/pat00540/do (viitattu 14. 04. 2022).

[10] Rannikko, A. *Suomalainen Eturauhassyöpä*. URL: https://www.suomalaineneturauhassyopa.fi/.

[11] Suomalaisen Lääkäriseuran Duodecimin ja Suomen Urologiyhdistys ry:n asettama työryhmä. Käypä hoito -suositus : Eturauhassyöpä. *Käypä hoito* (2014). URL: https://www.kaypahoito.fi/xmedia/hoi/hoi11060.pdf.

[12] T. Tammela. *Eturauhassyöpä*. Hae: Duodecim Oppiportti > Oppikirjat > Urologia > Aikuisten urologiset sairaudet > Eturauhassyöpä. (2019). URL: https://www.oppiportti.fi/op/uro01900/do (viitattu 14. 04. 2022).

[13] P. Hervonen & T. Utriainen. Levinneen eturauhassyövän lääkehoito. *Duodecim* 135.2 (2019), 183–6. URL: https://www.duodecimlehti.fi/lehti/2019/2/duo14742?keyword=entsalutamidi.

[14] Lääketietokeskus Oy. *Xtandi-lääkepakkaus*. Hae: Lääkeinfo > Nimihaku > xtandi. (2021). URL: https://laakeinfo.fi.

[15] Lääketietokeskus Oy. *Zytiga-lääkepakkaus*. Hae: Lääkeinfo > Nimihaku > zytiga. (2021). URL: https://laakeinfo.fi.

[16] K. Aittomäki & P. Peltomäki. Lääketieteellinen genetiikka - Syövän genetiikka ja Mikä tekee solusta syöpäsolun? *Duodecim* (2016). Artikkelien tunnukset (ltg01800 (018.000)) sekä (ltg01801 (018.001)). Saatavilla: https://www.oppiportti.fi/op/ltg01800/do sekä https://www.oppiportti.fi/op/ltg01801/do.

[17] National Library of Medicine. *National Center for Biotechnology Information. Gene*. URL: https://www.ncbi.nlm.nih.gov/gene/.

[18] The American Cancer Society medical and editorial content team. *What is prostate cancer?* URL: https://www.cancer.org/cancer/prostate-cancer/about/what-is-prostate-cancer.html (viitattu 14. 04. 2022).

[19] *Datasetti on saatavilla cBioPortalin Github-repositoriossa*. URL: https://github.com/cBioPortal/datahub/tree/master/public/prad_su2c_2019.

[20] H. He & E. A. Garcia. Learning from Imbalanced Data. *IEEE Transactions on knowledge and data engineering* 21.9 (2009), 1263–1284. DOI: 10.1109/TKDE.2008.239.

[21] R. C. Prati et al. Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. *Knowledge and Information Systems* 45 (2015), 247–270. DOI: https://doi.org/10.1007/s10115-014-0794-3.

[22] Google Developers. *Imbalanced Data*. 2021. URL: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data (viitattu 14. 04. 2022).

[23] F. Pedregosa. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[24] *Scikit-learn : User guide*. URL: https://scikit-learn.org/stable/user_guide.html (viitattu 14. 04. 2022).

[25] R. Garreta, G. Moncecchi. *Learning scikit-learn : machine learning in Python*. eng. 1st edition. Community experience distilled. Birmingham: Packt Publishing, 2013. ISBN: 1-78328-194-4.

LIITE A

```
"""
This file contains functions used for reading patient data from files.
It also has functions for converting real number values to binary (-1,1).


author: Kasper Styrman, Tampereen yliopisto, BMT
student id: 284971

"""


import numpy as np
import pandas as pd
import statistics
import io


SEP = '\t'
# treshold for bad and good treatment outcome
TRESHOLD = 6



def read_data_into_matrix():
    """
    Read patient data from different files and
    combine them into one pandas dataframe.
    The row structure is as follows:
    [SAMPLE_ID, PATIENT_ID, CHEMO_REGIMEN_CATEGORY, AGE_AT_DIAGNOSIS,
     PSA, OS_STATUS, OS_MONTHS, CNA DATA -->]
    :return: <pandas.DataFrame> M, contains relevant data for model training
    """
    # load raw data into pd.DataFrames
    # rows are genes, columns are samples -> one col = one sample's data
    cna = pd.read_csv('../../data_CNA.txt',sep=SEP)
    sample = pd.read_csv('../../data_clinical_sample.txt', sep=SEP, skiprows=4)
```

```
    patient = pd.read_csv('../../data_clinical_patient.txt',sep=SEP, skiprows=4)

    M = []
    # create column names
    col = ['SAMPLE_ID','PATIENT_ID']
    col.extend(patient.columns[1:])
    col.extend(cna['Hugo_Symbol']) # each gene

    # build matrix rows containing needed data
    for index, row in sample.iterrows():
        sampleID = row['SAMPLE_ID']
        patID = row['PATIENT_ID']
        # fetch row by patientID
        patientData = patient.loc[patient['PATIENT_ID'] == patID,'CHEMO_REGIMEN_CATEGORY':]
         # column contains gene data for sample
        cnaData = cna[sampleID]
        arr = [sampleID, patID]
        # unpack numpy array
        arr.extend(patientData.values[0])
        arr.extend(cnaData)
        M.append(arr)

    M = pd.DataFrame(data=M, columns = col)

    return M




def select_arsi_patients(M):
    """
    Create matrix consisting of feature vector (RB1, TP53 and AR genes)
    and label (survival months).
    Many of the entries don't have an os_month (weren't treated with an arsi),
    so those rows are skipped.
    :param: <pandas.DataFrame> M, matrix containing all read patient data
    :return: <pandas.DataFrame> N, filtered out matrix consisting of features and label
    """

    N = M.loc[pd.notnull(M['OS_MONTHS']), :]
```

```
78      # remove duplicate patient data (some had more than one sample
        taken)
79      N = N.drop_duplicates(subset = 'PATIENT_ID')
80      # select genes to be studied (features x) and survival months (
        label y)
81      N = N[['RB1','TP53','AR', 'OS_MONTHS']]
82
83      return N
84
85
86  def os_months_to_binary(N, treshold = TRESHOLD):
87      """
88      Classify continues label values (os_months) into either one of
89      the binary values -1 or 1 (bad or good response).
90      The classification is determined by the amount of survival
        months when
91      compared to the treshold value (bad if less or equal to
        treshold).
92      :param: <pandas.DataFrame> N, matrix containing features and
        labels
93      :param: <numeric> treshold, value for line between bad and good
        response classes
94      :return: <pandas.DataFrame> A, matrix containing features and
        labels
95      """
96
97      A = N.copy()
98      A.loc[N['OS_MONTHS'] <= treshold, 'OS_MONTHS'] = -1
99      A.loc[N['OS_MONTHS'] > treshold, 'OS_MONTHS'] = 1
100     return A
101
102 def init_osm_data(treshold = TRESHOLD):
103     """
104     Initialise the data for classification model training.
105     Data is read into a matrix, from which only the needed and
        valid data is kept.
106     The resulting matrix is separated into a feature vector x and
        class label vector y.
107     :return: <numpy.Array> x, array containing genetic features (
        CNA data)
108     :return: <numpy.Array> y, array containing classification
        labels (-1,1)
109     """
110
111     M = read_data_into_matrix()
112     M = select_arsi_patients(M)
113     M = os_months_to_binary(M, treshold)
114     x = M.loc[:, ['RB1', 'TP53', 'AR']].to_numpy()
```

```
115     y = M.loc[:, 'OS_MONTHS'].to_numpy()
116 
117     return x, y
```

LIITE B

```
"""
Code file containing functions for creating, training and testing machine learning models.

author: Kasper Styrman, Tampereen yliopisto, BMT
student id: 284971

"""

import read_data

from sklearn.svm import SVC as svc
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import math
import random
import numpy


RATIOS = [0.4, 0.5, 0.6]
FOLD_RATIO = 0.1
FILENAME = "test_results_new_program/data-sampling-modified"

def undersample(x,y,ratio):
    """
    Randomly undersample the majority class from feature and label vectors
    so that the ratio of major/total is reached.
    :param x:<numpy.Array>, feature vector
    :param y:<numpy.Array>, label vector
    :param ratio:<float>, amount of majority class points = ratio*total
    :return:<numpy.Array,numpy.Array>, undersampled feature and label vector
    """

    # majority samples amount
    smaj = len(y[y==1])
    s = len(y)
    amount_of_remaining_samples = smaj - math.floor((smaj-ratio*s)/(1-ratio))

```

```
38      # pick minority arrays
39      minx = x[y==-1]
40      miny = y[y==-1]
41
42      # pick majority arrays
43      majx = x[y==1]
44      majy = y[y==1]
45
46      # randomly choose which indices to keep from total range
47      index_of_remaining = random.sample(range(len(majy)),
        amount_of_remaining_samples)
48      majx = majx[index_of_remaining]
49      majy = majy[index_of_remaining]
50      # add remaining majority class to minority class among the same
        axis
51      return numpy.append(minx,majx, axis=0),numpy.append(miny,majy,
        axis=0)
52
53
54  def oversample(x,y,ratio):
55      """
56      Randomly oversample the minority class from feature and label
        vectors
57      so that the ratio of minor/total is reached.
58      :param x:<numpy.Array>, feature vector
59      :param y:<numpy.Array>, label vector
60      :param ratio:<float>, amount of minor class points = ratio*
        total
61      :return:<numpy.Array,numpy.Array>, oversampled feature and
        label vector
62      """
63
64      smin = len(y[y==-1])
65      s = len(y)
66
67      # pick minority arrays
68      minx = x[y==-1]
69      miny = y[y==-1]
70
71      # pick majority arrays
72      majx = x[y==1]
73      majy = y[y==1]
74
75      amount_of_samples_to_add = math.floor((ratio*s-smin)/(1-ratio))
76      # randomly generate the points to replicate
77      index_to_replicate = [random.randrange(len(miny)) for i in
        range(amount_of_samples_to_add)]
78
```

```
79      replicated_x = minx[index_to_replicate]
80      replicated_y = miny[index_to_replicate]
81      x = numpy.append(minx,majx,axis=0)
82      y = numpy.append(miny,majy,axis=0)
83
84      return numpy.append(x,replicated_x, axis=0), numpy.append(y,
        replicated_y, axis=0)
85
86
87
88  def get_scores(y,ypred):
89      """
90      Calculate the trained models' performance based on the actual
91      data point labels and the predicted labels. The extracted
        parameters are
92      true negatives, false positives, false negatives and true
        positives.
93      :param y:<numpy.Array>, actual labels
94      :param ypred:<numpy.Array>, labels predicted by the model
95      """
96      tn, fp, fn, tp = confusion_matrix(y, ypred).ravel()
97
98      precision_good_response =  float(tp/(tp+fp)) if (tp+fp) > 0
        else 'none'
99      precision_bad_response = float(tn/(tn+fn)) if (tn+fn) > 0 else
        'none'
100
101     recall_good_response = float(tp/(tp+fn)) if (tp+fn) > 0  else '
        none'
102     recall_bad_response = float(tn/(tn+fp)) if (tn+fp) > 0 else '
        none'
103
104     return (tn, fp, fn, tp,
105             precision_good_response,
106             precision_bad_response,
107             recall_good_response,
108             recall_bad_response)
109
110
111
112 def record_scores(scores, ratio, sampling_type, fold_ratio =
        FOLD_RATIO):
113     """
114     Record the results for an individual program cycle (model
        training
115     and testing). The format in the output file is:
116     sampling;ratio;tn,fp,fn,tp,precision_positive;
        precision_negative;recall_positive;recall_negative
```

```
117     The foldRatio index in the filename indicates the test-training
        data split ratio.
118     :param scores: <float Tuple>, the numbers from model tests
119     :param ratio: <float>, the ratio by which the data was modified
        (sampling)
120     :param sampling_type: <string> indicates how the data was
    modified
121     (either over, under or natural sampling)
122     :param fold_ratio: <float> the ratio for the test-train data
    split
123     """
124     with open(f'{FILENAME}-foldRatio-{fold_ratio}.txt', 'a') as
    file:
125         file.write(f'{sampling_type};{ratio}')
126         for data in scores:
127             file.write(f';{data}')
128         file.write('\n')
129     file.close()
130
131
132 def split_data(x,y,ratio_fold):
133     """
134     Divide data into training and test sets according to required
    fold ratio
135     (how big of a part of the whole set is set aside as testing
    data), and
136     natural ratio, which is used to keep the minority and majority
    datapoint
137     ratios stable.
138     :param x:<numpy.Array>, the dataset feature vector,
139     :param y:<numpy.Array>, the dataset label vector,
140     :param ratio_fold:<float>, 0 < r < 1, the amount of the dataset
        that is set aside as test set,
141     :return:<numpy.Array>(4), feature_train, feature_test,
    label_train, label_test
142     """
143
144     # All datapoints amount in dataset (size)
145     s = len(y)
146
147     # determine dataset natural minority datapoint ratio
148     # assuming that the dataset contains only two labels: -1 and 1
149     ratio_natural = float(len(y[y==-1])/s)
150
151
152     # calculate test set amounts based on fold ratio
153     test_set_size = math.floor(s*ratio_fold)
154     minority_in_test = math.floor(test_set_size*ratio_natural)
```

```
155     majority_in_test = test_set_size - minority_in_test
156 
157     # fetch random datapoints into test set, then delete them.
158     minx = x[y==-1]
159     miny = y[y==-1]
160     majx = x[y==1]
161     majy = y[y==1]
162 
163     index_of_min_in_test = [random.randrange(len(miny)) for i in
    range(minority_in_test)]
164     index_of_maj_in_test = [random.randrange(len(majy)) for i in
    range(majority_in_test)]
165 
166     test_x = numpy.append(minx[index_of_min_in_test],
167                           majx[index_of_maj_in_test], axis = 0)
168     test_y = numpy.append(miny[index_of_min_in_test],
169                           majy[index_of_maj_in_test], axis = 0)
170     train_x = numpy.append(numpy.delete(minx,index_of_min_in_test),
171                            numpy.delete(majx,index_of_maj_in_test),
    axis = 0)
172     train_y = numpy.append(numpy.delete(miny,index_of_min_in_test),
173                            numpy.delete(majy,index_of_maj_in_test),
    axis = 0)
174 
175     return test_x, test_y, train_x, train_y
176 
177 
178 
179 
180 def test_svc(x, y, fold_ratio = FOLD_RATIO):
181     """
182     Executes program cycles by using the predefined functions.
183     The general flow is as follows:
184         The data is split into training and testing data (
    fold_ratio).
185         A SVC model is trained using the training split (natural
    sampling).
186         The model is tested and the scores are recorded into a txt
    file.
187         The function loops through the same cycle utilising either
188         oversampling or undersampling according to one of the given
     ratios.
189 
190 
191     :param x: <numpy.Array> feature vectors for the dataset
192     :param y: <numpy.Array> label data for the dataset
193     :param fold_ratio: <float> indicates the relative amount of
    test samples
```

```
194                                 in relation to all datapoints ->
    used to
195                                 determine split ratio between test
    and training data
     """
196
197
198
199     xtrain, xtest, ytrain, ytest = split_data(x,y, fold_ratio)
200     model = svc()
201
202     model.fit(xtrain, ytrain)
203     ypred = model.predict(xtest)
204
205     record_scores(get_scores(ytest, ypred), 'none','none',
    fold_ratio)
206
207     for ratio in RATIOS:
208         xover, yover = oversample(xtrain, ytrain, ratio)
209         xunder, yunder = undersample(xtrain, ytrain, ratio)
210
211         model.fit(xover,yover)
212         ypred = model.predict(xtest)
213
214         record_scores(get_scores(ytest, ypred), ratio, 'over',
    fold_ratio)
215
216         model.fit(xunder,yunder)
217         ypred = model.predict(xtest)
218
219         record_scores(get_scores(ytest, ypred), ratio, 'under',
    fold_ratio)
```