TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

AINO KOSKIMIES

KIELEN TUNNISTUS KONEOPPIMISMENETELMILLÄ

Kandidaatintyö

Tarkastaja: Prof. Heikki Huttunen
Jätetty tarkastettavaksi 21.10.2017

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma, Signaalinkäsittelyn laitos
KOSKIMIES, AINO: Kielen tunnistus koneoppimismenetelmillä
Kandidaatintyö, 21 sivua
Lokakuu 2017
Pääaine: Machine learning
Tarkastaja: Prof. Heikki Huttunen
Avainsanat: Koneoppiminen, kielen tunnistus, TF-IDF, n-gram, Random forest, logistinen regressio

Koneoppimismenetelmien avulla aineistosta voidaan irrottaa piirteitä ja luokitella se eri luokkiin näiden piirteiden perusteella. Erilaisia piirteenirrotus- ja luokittelumenetelmiä hyödyntämällä on mahdollista esimerkiksi tunnistaa kirjoitetusta tekstistä sen kieli. Tässä työssä esitellään erilaisia ohjattua oppimista käyttäviä koneoppimismenetelmiä ja kokeillaan niiden soveltuvuutta kielen tunnistamiseen. Työssä käytettyjä piirteenirrotusmenetelmiä ovat TF-IDF ja n-gram, kun taas luokitteluun käytettyihin menetelmiin kuuluvat Random forest, logistinen regressio ja tukivektorikone.

Algoritmien esittelemisen lisäksi käydään vaihe kerrallaan läpi kielen tunnistuksessa tapahtuva koneoppimisprosessi. Työssä esitetään testauksen tulokset ja niiden pohjalta jokaisen menetelmän suoriutumista arvioidaan. Tutkimuksen perusteella nähdään, että kaikki tässä työssä käsitellyt piirteenirrotus- ja luokittelumenetelmät soveltuvat hyvin tekstin luokitteluun kirjoituskielen perusteella.

SISÄLTÖ

1. Johdanto 1

2. Teoria 3

2.1 Piirteenirrotus 4

2.1.1 TF-IDF 4

2.1.2 N-gram 5

2.2 Luokittelu 6

2.2.1 Random Forest 7

2.2.2 Tukivektorikone 8

2.2.3 Logistinen regressio 9

2.3 Virheen arviointi 9

3. Testaus ja tulokset 11

3.1 Työkalut 11

3.2 Aineisto 12

3.3 Toteutus 13

3.4 Tulokset 14

4. Johtopäätökset 17

Lähteet 18

TERMIT JA LYHENTEET

C	Kaikkien luokitteluongelman kohdekategorioiden joukko
cURL	Komentoriviohjelma, jonka avulla luetaan dataa URL-lähteestä
D	Kaikkien aineiston dokumenttien joukko
n-gram	Piirteenirrotusmenetelmä, jossa teksti jaetaan määritellyn mittaisiin osiin ja osille määritetään painokertoimet
OOB	*Out-Of-Bag* menetelmä, jolla arvioidaan Random forest -algoritmin virhettä luokittelijapuissa käyttämättä jääneiden piirteiden avulla
TF-IDF	*Term Frequency-Inverse Document Frequency*, piirteenirrotusmenetelmä, joka laskee sanoille painokertoimia ja määrittää näin olennaiset piirteet kullekin luokalle
tsv	*Tab-Separated Values*, tiedostomuoto, jossa tekstistä koostuvan aineiston osat on eroteltu toisistaan sarkainmerkillä
Twiitti	Yhteisöpalvelu Twitterissä julkaistu korkeintaan 140 merkkiä pitkä kirjoitus

1. JOHDANTO

Modernisoituvassa maailmassa koneoppimisen merkitys kasvaa jatkuvasti. Tekstiä tunnistavia koneoppimisalgoritmeja käytetään esimerkiksi roskapostin tunnistamisessa ja ne voivat auttaa löytämään keskustelupalstoille ja uutissivustojen kommenttipalstoille kirjoitettujen kommenttien joukosta sellaiset, jotka eivät noudata palstan sääntöjä. Tällaisilla algoritmeilla voidaan toteuttaa myös ohjelma, joka kerää sosiaalisesta mediasta tiettyyn yritykseen, kilpailuun tai henkilöön liittyvät julkaisut ja määrittää niiden sävyn (negatiivinen–positiivinen). Kyseistä menetelmää on kokeiltu muun muassa vuoden 2016 Yhdysvaltain presidentinvaalien tuloksen ennustamisessa [1]. Tutkimuksessa analysoitiin yhteisöpalvelu Twitterissä julkaistujen kirjoitusten mielipidetrendejä ja ennustettiin niiden avulla presidenttiehdokkaihen kannatusta.

Koneoppimisalgoritmeja käytetään nykypäivänä myös tunnistamaan tekstin kieli. Käyttäjille on jo pitkään tarjottu Facebookin kaltaisilla verkkosivustoilla ja joissakin verkkoselaimissa kuten Google Chrome:ssa automaattista kääntämistä vieraasta kielestä käyttäjän omaan kieleen. Näissä tapauksissa algoritmi tunnistaa tekstin kielen ja tekee käännöksen tämän tiedon avulla. Käännöksestä saadaan sitä parempi mitä paremmin käännösalgoritmi oppii tunnistamaan sanaston ja tämän avulla kääntämään sanan kontekstissaan [2].

Tässä tutkielmassa pyritään vertailemaan erilaisten koneoppimismenetelmien soveltuvuutta tekstin kielen tunnistamiseen. Tarkasteltavat menetelmät ovat arkkitehtuuriltaan niin kutsuttuja perinteisiä koneoppimismenetelmiä, eli ne eivät perustu neuroverkkoihin, toisin kuin esimerkiksi syväoppiminen (engl. *deep learning*). Käytännön testeissä käytetään materiaalina yhteisöpalvelu Twitterin twiittejä (yhteisöpalvelu Twitterissä julkaistu korkeintaan 140 merkkiä pitkä kirjoitus), joiden kieli tunnistetaan Pythonin scikit-learn-kirjastoa hyödyntävällä ohjelmalla. Tarkasteltavia piirteenirrotusmenetelmiä ovat TF-IDF (engl. *Term Frequency-Inverse Document Frequency*) ja n-gram, ja luokittelumenetelmiä Random forest, tukivektorikone ja logistinen regressio. Nämä menetelmät valittiin, sillä ne ovat alan julkaisujen perusteella yleisesti käytettyjä menetelmiä myös kielentunnistuksen saralla. Edel-

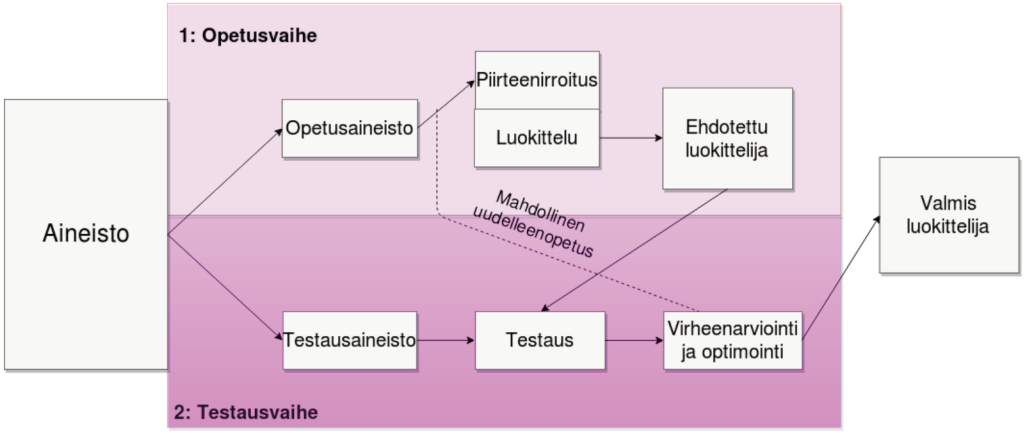
lämainittujen seikkojen pohjalta tutkimuskysymys muotoillaan seuraavasti: *Kuinka hyvin tarkastellut koneoppimismenetelmät soveltuvat tekstin kielen tunnistamiseen?*

Seuraavaksi esitellään tutkittavien koneoppimismenetelmien teoreettista taustaa luvussa kaksi. Selkeyden vuoksi kyseisessä luvussa avataan ensin tekstintunnistusprosessin kulkua ja tämän jälkeen käsitellään erikseen piirteenirrotusmenetelmät ja luokittelumenetelmät. Luvun lopussa kerrotaan vielä virheenarvioinnista ristiinvalidointimenetelmällä ja pohditaan milloin saatua tulosta voidaan pitää luotettavana. Luku kolme käsittelee tarkemmin menetelmien käytännön toteutusta ja eri menetelmien suoritusten vertailua. Sen alkupuolella kerrotaan pääseikat käytetyistä työkaluista ja aineistosta, loppupuolella taas arvioidaan saatuja tuloksia. Viimeisessä, eli neljännessä luvussa kerrotaan työstä syntyneet johtopäätökset.

2. TEORIA

Tämän luvun tarkoitus on käsitellä luokitteluongelmia ratkaisevien koneoppimismenetelmien teoriaa ja esitellä tarkemmin työhön liittyvät algoritmit. Koneoppimisongelman ratkaisu alkaa aina aineiston keruusta ja kyseisen käsittelemättömän aineiston muuttamisesta sellaiseen muotoon, että sitä pystytään hyödyntämään algoritmissa [3]. Kun aineisto on sopivassa muodossa käsiteltäväksi, jaetaan se osiin algoritmin opetus- ja testausvaihetta varten. Kuvasta 1 voidaan nähdä yhdenlaisen luokitteluongelmia ratkaisevan koneoppimisalgoritmin toiminnan vaiheet. Nämä vaiheet esitellään paremmin seuraavaksi.

Opetusvaiheessa käytettävä aineisto on nimeltään opetusaineisto. Tekstien kieliä luokiteltaessa se koostuu eri kielillä kirjoitetuista tekstinpätkistä $T_1, T_2, ..., T_n$. Opetusaineisto käsittää yli puolet, usein noin 80%, alkuperäisestä käsitellystä aineistosta. Tarvittavan aineiston koko päätetään aina tehtävän mukaan, mutta vähimmäismäärä hyvän oppimistuloksen saavuttamiseksi on useita satoja näytteitä kutakin luokittelun kohderyhmää kohti [4]. Tämän jälkeen opetusaineiston näytteistä lasketaan p-ulotteiset piirrevektorit $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}$, jotka sisältävät tunnistuksen kannalta olennaisia piirteitä [5]. Piirteenirrotuksen jälkeen algoritmi etenee luokitteluun. Luokittelussa määritetään piirrevektoreita vastaavat luokat $y_1, y_2, ..., y_m$, missä $y_k \in \{1, 2, ..., m\}$. Esimerkiksi tekstin kielen luokittelussa luokat voisivat olla $(1, 2, 3)$,



Kuva 1 *Luokittelevan koneoppimisalgoritmin toiminta.*

missä 1 vastaa englantia, 2 ranskaa ja 3 saksaa. Näiden vaiheiden jälkeen koneoppimismenetelmillä on saatu aikaan jonkinlainen ehdotus luokittelija-algoritmista.

Saadun algoritmin laatu tulee vielä testata testausvaiheessa. Tämän vaiheen aineisto voidaan jakaa kahteen osaan: testausaineistoon ja validointiaineistoon. Jälkimmäisenä mainittua hyödynnetään arvioitaessa opetusvaiheessa muodostetun luokittelijan suoritusta. Tulosta verrataan näytteiden tunnettuihin luokkiin ja näin pystytään arvioimaan opetusvaiheessa luodun lajittelualgoritmin virhettä. Tämän perusteella voidaan haluttaessa vielä optimoida luokittelijaa, mutta tämä johtaa helposti algoritmin ylioppimiseen eli siihen, että algoritmi oppii testiaineiston [6]. Tämän vuoksi tulosten perusteella optimointi on melko riskialtis menetelmä optimointiin. Lopuksi testausaineistoa käytetään lopullisen mallin virheen arviointiin [7, s. 222].

Seuraavaksi käydään yksityiskohtaisesti läpi edellä mainitut vaiheet ja esitellään työhön liittyvässä käytännön toteutuksessa tutkitut menetelmät. Näihin menetelmiin kuuluu piirteenirrotuksen osalta TF-IDF ja n-gram, sekä luokittelun osalta Random forest, tukivektorikone ja logistinen regressio.

2.1 Piirteenirrotus

Piirteenirrotus (engl. *feature extraction*) tarkoittaa käsiteltävän datan jakamista sopiviin osiin ja muuntamista lukuarvoiksi, joita koneoppimisalgoritmin on helpompi käsitellä kuin sanoja tai valokuvan pikseleitä. Teksti voidaan esimerkiksi jakaa lauseisiin, sanoihin tai 1–3 kirjainmerkin mittaisiin pätkiin. Koska kieli rakentuu siten, että sanojen ja kirjainten järjestyksellä on merkitystä, toimii kielen tunnistuksessa myös muutaman kirjainmerkin mittaisiin pätkiin perustuva piirteenirrotus. Jakamisen jälkeen piirteille annetaan lukuarvo esimerkiksi kyseisen piirteen esiintymistiheyden mukaan. [8]

On useita erilaisia tapoja suorittaa piirteenirrotus. Optimaalisinta piirteenirrotusmenetelmää ei voida yksilöidä, vaan eri tarkoituksiin sopivat erilaiset menetelmät. Irrotettavat piirteet voidaan myös valita usealla tavalla. Tässä työssä tarkemmin käsitellyt TF-IDF ja n-gram käyttävät piirteenvalintaan filteröintimenetelmää. [9]

2.1.1 TF-IDF

TF-IDF-piirteenirrotusmenetelmä (engl. *term frequency-inverse document frequency*) on tarkoitettu erityisesti piirteiden irrottamiseen tekstistä. Tämän vuoksi se soveltuu hyvin esimerkiksi tekstin sävyn, kielen tai aiheen tunnistamiseen [8, 10, 11]. TF-

IDF-menetelmässä käsiteltävää tekstiä arvioidaan laskemalla termeille painokertoimia, mikä tapahtuu laskemalla termin esiintymistiheyksien summa S_{TF} ja termin esiintymisestä kaikissa dokumenttinäytteissä kertova luku S_{IDF}. Yksinkertaisimmillaan summa S_{TF} on vain luku joka kertoo montako kertaa kyseinen termi esiintyy dokumentissa, eli $S_{TF}(t,d) = f_{t,d}$, jossa $f_{t,d}$ kuvaa termin esiintymistiheyttä, eli montako kertaa termi t esiintyy dokumentissa d. Useimmissa sovelluksissa käytetään kuitenkin seuraavaa logaritmisesti skaalattua painokerrointa S_{TF}:lle:

$$S_{TF}(t,d) = 1 + log(1 + f_{t,d}) \tag{1}$$

Lopullisen TF-IDF:n laskemiseksi edellisellä kaavalla laskettu luku kerrotaan painokertoimella S_{IDF}, joka kertoo, kuinka moni koko aineiston dokumenteista sisältää kyseisen termin. Kuten S_{TF} myös painokerroin skaalautuu logaritmisesti, ja se saadaan laskettua kaavasta

$$S_{IDF}(N, n_t) = log\frac{N}{n_t}, \tag{2}$$

jossa N = dokumenttien kokonaismäärä. Termi n_t taas määrittää moniko dokumenteista sisältää termin t, matemaattisesti ilmaistuna $n_t = |(d \in D : t \in d)|$.

Näin ollen saadaan TF-IDF:n kaavaksi

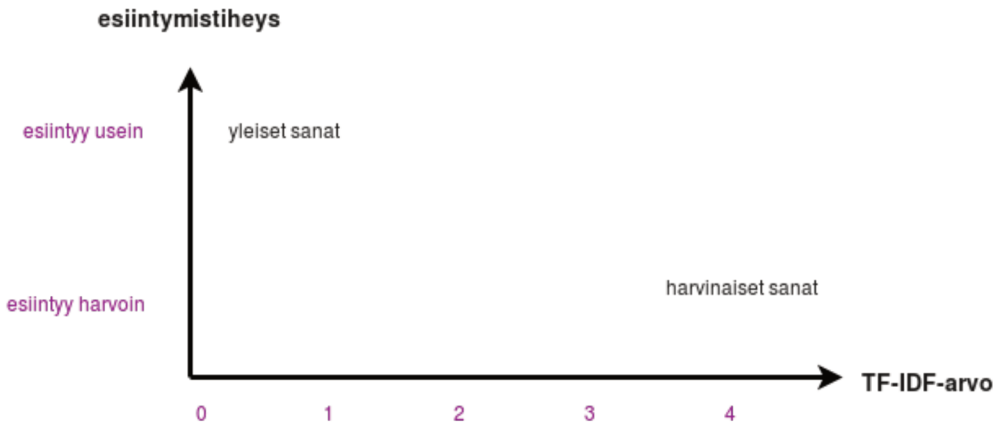
$$D(t) = S_{TF} \cdot S_{IDF} = (1 + log(1 + f_{t,d})) \cdot log\frac{N}{n_t}. \tag{3}$$

jossa D kuvaa kaikkien dokumenttien joukkoa termien t funktiona. Termien esiintymistiheys on $f_{t,d}$, N dokumenttien kokonaismäärä ja n_t termin sisältävien dokumenttien määrä.

TF-IDF perustuu siihen, että saadun painokertoimen arvo on sitä suurempi, mitä harvinaisempi sana on. Toisin sanoen kielessä harvinaisemmin esiintyvillä sanoilla on suurempi merkitys. Tämä voidaan havainnollistaa kaaviokuvan 2 avulla.

2.1.2 N-gram

Kuten Häkkinen ja Tian tutkimuksessaan muotoilevat: "The n-gram method uses letter n-grams, representing the frequency of occurrence of various n-letter combina-



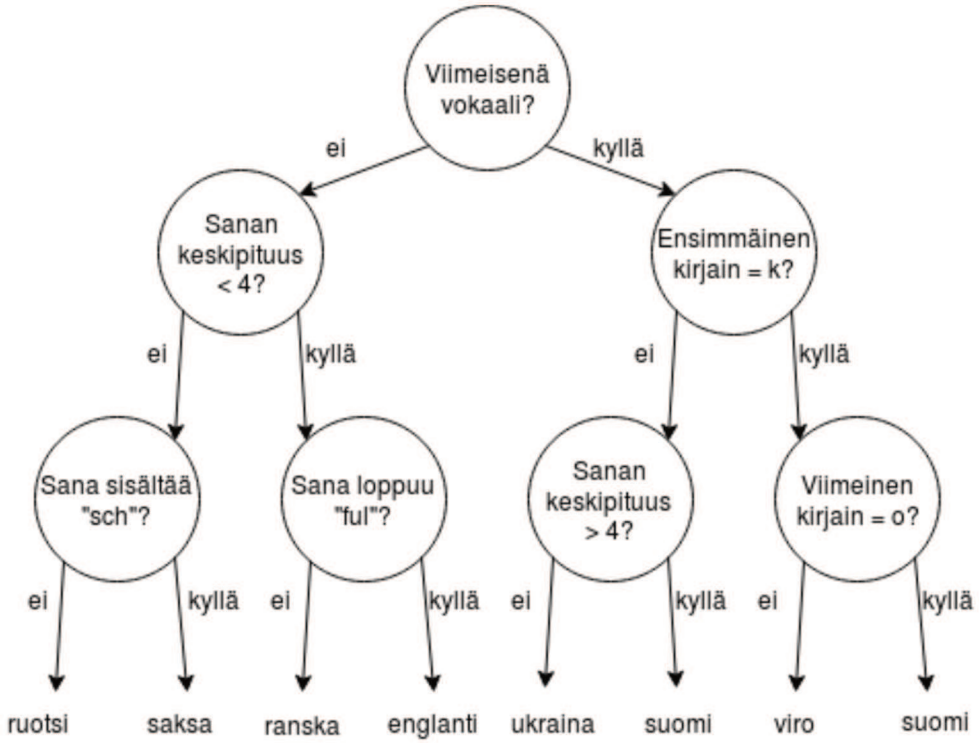
Kuva 2 *Sanojen TF-IDF-arvon määräytyminen.*

tions in a particular language." [12] Tämä merkitsee sitä, että n-gramin käyttämät pätkät voivat olla kirjain- tai sanayhdistelmiä, sanoja, yksittäisiä kirjaimia tai merkkejä ja niin edelleen. Jos käyttää esimerkiksi 2-gramia eli niin kutsuttua bigramia, ohjelma jakaa tekstin "ohjelmointi on hauskaa" seuraavanlaisiin osiin: ("ohjelmointi", "on", "hauskaa", "ohjelmointi on", "on hauskaa") [8]. Sana- ja kirjainyhdistelmät luodaan aina viereisistä sanoista tai kirjaimista, tämän vuoksi tavanomaisessa bigramissa esimerkkilauseesta ei muodosteta sanaparia "ohjelmointi hauskaa" [13]. Toisaalta myös sanojen viimeiset kirjaimet voivat olla hyvä keino kielen päättelemiseksi. Jos useampi sana loppuu esimerkiksi päätteisiin "ness", "tion" tai "ful" tekstin kieli on todennäköisimmin englanti, kun taas "nen"-päätteisten sanojen esiintyminen viittaa suomen kieleen [12].

Kun n-gramin avulla tunnistetaan tekstin kieltä, lasketaan muodostettujen osasten perusteella todennäköisyys, jolla kukin osa kuuluu tiettyyn kohdekieleen. Se kohdekieli, joka kyseisen osan kohdalla osoittautuu todennäköisimmäksi, päätetään tekstin kieleksi. [12] Jos n-gramin tulosta halutaan optimoida, voidaan tekstistä poistaa merkityksen kannalta epäolennaiset sanat (engl. *stop words*). Tällaisia ovat esimerkiksi artikkeli, kuten saksan *der*, *die* ja *das*, sekä prepositiot, kuten englannin *with*, *for* ja *in* [13]. Epäolennaiset sanat poistamalla voidaan esimerkiksi bigrameista tehdä mielekkäämpiä tekstintunnistuksen kannalta.

2.2 Luokittelu

Luokitteluvaiheessa koneoppimisalgoritmi suorittaa varsinaisen luokittelun. Esimerkiksi tekstintunnistusongelmaa ratkaistessa tämä tarkoittaa sitä, että algoritmi päättelee kullekin luokalle, tässä tapauksessa kielelle, ominaiset piirteet oppimisvaiheessa



Kuva 3 *Esimerkki yksittäisestä kieliä luokittelevasta puusta.*

rakennetun luokittelijan avulla. Kyseinen ongelma voidaan myös ilmaista Boolen logiikan avulla seuraavasti: määritetään kuuluuko dokumenttien joukosta D tarkasteltava yksittäinen dokumentti d kategoriaan c. Kategoria c on kaikkien kategerioiden C joukkoon kuuluva yksittäinen kategoria, tässä tapauksessa siis yksi kieli. Yksittäinen dokumentti ja sen kategoria kuuluvat siis aina kaikkien dokumenttien ja kaikkien kategorioiden joukkoon, matemaattisesti ilmaistuna dokumentti $(d_i, c_j) \in D \times C$. [14]

2.2.1 Random Forest

Random forest -algoritmi muodostaa opetusdatan avulla erilaisia puurakenteita (engl. *decision trees*), joiden avulla tunnistetaan aineiston luokka. Nimensä tämä algoritmi on saanut siitä, että puurakenteisia luokittelijoita on useita ja ne muodostavat ikäänkuin satunnaisesti luotujen luokittelijoiden metsän. Kun Random forest -menetelmään perustuva ohjelma saa syötteen, jonka luokka pitää päätellä, antaa jokainen puu erikseen arvauksen syötteen luokasta. Toisin sanoen jokainen puu ikäänkuin äänestää syötteen luokkaa ja eniten ääniä saanut luokka päätellään syötteen luokaksi, kuten kuvassa 3. [15]

Yksityiskohtaisemmin kuvattuna Random forest -algoritmi muodostetaan tekemällä satunnaisia, toisistaan riippumattomia piirrevektoreita ja rakentamalla jokainen

luokittelijapuu näistä vektoreista opetusaineiston avustuksella [16]. Puut ovat binäärisiä, eli jokaisella puun solmulla on vain kaksi juurisolmua tai -lehteä [17]. Algoritmin suoritustarkkuus kasvaa puiden määrän kasvaessa, mutta jo noin viidestä puusta koostuvalla Random forest:lla saa melko luotettavan tuloksen [18]. Suoritustarkkuuden kasvu alkaa hidastua puiden määrän kasvaessa ja samalla kasvaa algoritmin opetusvaiheeseen käytetty aika, joten hyvänä Random forest:n kokona voidaan pitää esimerkiksi sataa luokittelijapuuta [18][19].

Random forest on taipuvainen ylioppimiseen, jos luokittelijapuut sisältävät liikaa samoja piirrevektoreita, toisin sanoen liian hyviä puita ei kannata rakentaa. Tämän vuoksi algoritmin virhemarginaalia ja toisaalta myös yhden piirteen merkitystä luokittelussa voidaan arvioida niin kutsutun OOB-menetelmän (*Out-Of-Bag*) avulla. Yksi OOB:n näyte kertoo montako sellaista piirrettä on, joita ei ole käytetty tarkasteltavan puun rakennukseen. [17] [19]

2.2.2 Tukivektorikone

Tukivektorikone (engl. *Support Vector Machine, SVM*) on usein käytetty koneoppimismenetelmä muun muassa hahmontunnistukseen, käsin kirjoitetun tekstin tunnistukseen ja tekstin kielen määrittämiseen [20]. Vladimir Vapnik esitteli menetelmän teorian jo 1960-luvulla, mutta parannuksia, kuten kernel trickin, hän lisäsi siihen kollegoidensa kanssa vasta 1990-luvun keskivaiheilla [21]. Kernel trick on menetelmä, jonka avulla luokittelu voidaan tehdä korkeammissa dimensioissa ilman, että näytteiden koordinaatteja pitää laskea[22]. Alkujaan tukivektorikoneet kehitettiin lineaarisiksi menetelmiksi, mutta myöhemmin niitä laajennettiin siten, että myös useamman luokan ongelmat olivat ratkaistavissa kyseiseen menetelmään pohjautuvan algoritmin avulla [23]. Funktiona ilmaistuna tukivektorikoneen pääsääntö on seuraava:

$$f(x) = \mathbf{w}^{Tx} + \mathbf{w_0}, \tag{4}$$

jossa luokka on 1 kun $f(x) > 0$, muutoin luokka on nolla.

Kahden luokan tapauksessa tukivektorikone toimii siten, että se yrittää löytää luokkia erottavan hypertason. Tätä tasoa kuvaa vektori $\mathbf{w} \in \mathbf{R}^P$. Vektori erottelee luokat toisistaan siten, että marginaali luokkien välillä on mahdollisimman suuri. Yksinkertaisimmissa luokittelutapauksissa luokittelu suoritetaan vain ratkaisemalla kummalla puolella hypertasoa kukin näyte on, toisin sanoen kernel trickiä ei käytetä. [24, 25] [7, pp. 417–421]

2.2.3 Logistinen regressio

Logistinen regressio kuuluu lineaarisiin luokittelijoihin. Se soveltuu erittäin hyvin binääriseen luokitteluun eli tapauksiin joissa kohdeluokkia on vain kaksi, mutta myös k:n luokan ongelmiin [25]. Logistinen regressio on saanut nimensä siitä, että se luokittelee piirrevektorista saamiensa piirteiden avulla käyttäen logistisia funktioita [26].

Opetusvaiheessa logistinen regressio käyttää aineiston oppimiseen iteratiivisesti painotettua pienimmän neliösumman menetelmää [26]. Kahden kohdeluokan tapauksessa logistinen regressio päättää luokan sen perusteella saako logistinen funktio arvon 1 vai 0. Tämä merkitsee kuitenkin sitä, että useamman luokan tapauksessa logistinen regressio vaatii niin kutsutun one-vs-all-yleistyksen, eli kukin luokka opitaan kaikkien muiden luokkien kokonaisuutta vastaan. [27].

2.3 Virheen arviointi

Virheen arvioiminen on luonnollisesti tärkeää minkä tahansa tieteellisen tutkimuksen tulosten arvioimisessa. Koneoppimisessa käytettyjen algoritmien suoritustarkkuuden arviointiin on kehitetty useita menetelmiä. Nämä menetelmät vertaavat opetusvaiheessa saatuja luokittelutuloksia tunnettuihin todellisiin luokkiin ja esittävät tuloksen esimerkiksi taulukon, käyrän tai prosenttilukujen avulla [28, 29, 30]. Myöhemmin esitellyistä menetelmistä confusion matrix:ssa tulokset esitetään taulukkomuodossa, jossa on esimerkiksi ristiinvalidoinnin prosenttilukuja tai absoluuttisia lukuja.

On huomattava, että virhettä saattaa olla jo aineistossa. Tällöin tulee muistaa, että vaikka algoritmin ennustustarkkuus olisi yli 90%, aineiston virhe tulee myös ottaa huomioon tutkimustulosta arvioitaessa. Toisaalta monet koneoppimisalgoritmit ovat robusteja, eli niiden tulos ei merkittävästi muutu yksittäisistä aineistossa olevista virheistä [31]. Seuraavaksi esitellään tarkemmin koneoppimisalgoritmien virheenarvioimiseen tehdyistä työkaluista confusion matrix, sekä virheenarvioimismenetelmä nimeltä ristiinvalidointi.

Ristiinvalidoinnissa (engl. *k-fold cross-validation*) aineisto jaetaan k:hon yhtäsuureen osaan ja tämän jälkeen koneoppimisalgoritmi ajetaan k kertaa. Jokaisella ajokerralla käytetään yhtä joukkoa testausjoukkona ja muita joukkoja opetusjoukkona. Termi tarkkuudelle saadaan laskemalla keskiarvo jokaisen ajon tarkkuuksista kulloisessakin testausjoukossa. Jos k on sama kuin luokiteltavien luokkien määrä, voidaan

tätä menetelmää kutsua *leave-one-out cross-validation* tai *n-fold cross-validation* [7, pp. 241–249]. Tulokset esitetään prosentteina ja ne kertovat tekstin kieltä tunnistettaessa montako prosenttia luokittelijan luokittelemista näytteistä on oikeassa luokkassa. [32, 33]

Toinen tässä työssä tutkittu virheenarviointitapa on confusion matrix. Se on visuaalinen tapa esittää algoritmin suoriutuminen tehtävästään. Confusion matrix käyttää suoriutumistarkkuuden esittämiseen matriisia, jonka vaakariveillä on näytteiden todelliset luokat ja pystyriveillä algoritmin testausvaiheessa ennustamat luokat [7, p. 301]. Täydellisesti suoriutuneen algoritmin confusion matrix muodostaisi näin ollen matriisin lävistäjälle kunkin luokan sisältämien näytteiden määrän ja muut matriisin alkiot olisivat nollia [34, p. 93].

3. TESTAUS JA TULOKSET

Tämä luku käsittelee koneoppimisalgoritmien testausta kielen tunnistukseen ja kertoo testausten tuloksista. Algoritmien testauksessa käytettiin sitä varten kirjoitettua kielen tunnistavaa ohjelmaa. Ohjelmassa kielet tunnistetaan eri koneoppimismenetelmillä ja vertaillaan keskenään niistä saatuja tuloksia. Näiden tulosten pohjalta arvioidaan menetelmien sopivuutta kielen tunnistuksen kaltaisiin koneoppimistehtäviin. Saman kaltaisia koneoppimistehtäviä ovat esimerkiksi kirjoituksen sävyn (negatiivinen–positiivinen) tunnistus ja roskapostin erottelu tavallisesta sähköpostista [8, 15].

On huomioitavaa, että piirteenirrotusmenetelmän valinta todennäköisesti vaikuttaa luokittelevan algoritmin suoritustarkkuuteen. Tämän vuoksi valittiin toteutukseen kaksi eri piirteenirrotusmenetelmää, TF-IDF ja n-gram, sekä kolme eri luokittelumenetelmää, logistinen regressio, Random forest ja tukivektorikone. Näin ollen saadaan tarkasteltavaksi kuusi erilaista yhdistelmää.

3.1 Työkalut

Tässä työssä esiteltyjen menetelmien tarkasteluun käytännössä hyödynnettiin Python-ohjelmointikieltä [1]. Python on yksi maailman käytetyimmistä ohjelmointikielistä [35] ja koneoppimiseen liittyvissä sovelluksissa eräiden lähteiden mukaan jopa käytetyin ohjelmointikieli [36, 37]. Koneoppimisessa työn osalta hyödynnettiin scikit-learn-kirjastoa. Tämä kirjasto on suunniteltu käytettäväksi koneoppimiseen ja se on avoimen lähdekoodin työkalu[2]. Koska scikit-learn on suunniteltu myös luokitteluun, sisälsi se tässä työssä käsitellyt algoritmit ja soveltui siksi käytettäväksi työn toteutukseen.

Aineiston jakamiseen käytettiin sitä varten kirjoitettua Python-ohjelmaa. Tämän ohjelman tarkemmasta toiminnasta kerrotaan seuraavassa luvussa 3.2. Lisäksi twiitit tuli ladata Twitter API:n avulla Twitteristä ohjelman käytettäviksi. Käytännössä

[1] https://www.python.org
[2] http://scikit-learn.org/stable

data-aineisto ladattiin tekstimuotoiseksi käyttäen erityisesti Twitterin datan käsittelyyn suunniteltua Twurl-ohjelmaa. Tämä ohjelma perustuu cURL-komentorivityökaluun, jonka tarkoituksena on tiedostojen käsittely ja lataaminen URL-pohjaista lähdettä käyttäen. [38]

3.2 Aineisto

Kun jotakin ongelmaa ratkaistaan koneoppimismenetelmällä, tarvitaan ohjelman opettamiseen ja testaamiseen aineistoa. Oppimisvaiheessa algoritmi oppii sille annetun aineiston avulla ja testausaineistossa arvioidaan kuinka hyvä tulos oppimisessa saavutettiin. Kerätty aineisto jaetaan opetus- ja testausaineistoon siten, että opetusaineistossa on esimerkiksi 80% ja testausaineistossa vähintään 20% näytteistä. On huomattava, että testausaineistossa ei saa olla yhtäkään näytettä, joka on esiintynyt opetusaineistossa. Jos aineisto on päällekkäistä, ei voida olla varmoja onko algoritmi oppinut opetusaineiston ulkoa eli ylioppinut (engl. *over-fitted*). Suositeltu vähimmäiskoko koneoppimisalgoritmin opettamista ja testaamista varten on muutaman sadan näytteen kokoinen aineisto. Suurimmat aineistot taas voivat sisältää miljoonia näytteitä. Algoritmin oppimisaika riippuu vahvasti opetusaineiston koosta. [39, s. 30–31]

Työssä käytettäväksi aineistoksi valittiin mikroblogipalvelu Twitteristä kerätty kokoelma twiittejä. Twitteriä käytti vuonna 2016 kuukausittain arviolta 328 miljoonaa käyttäjää ja sen käyttöliittymä oli tajolla yli neljälläkymmenellä eri kielellä [40]. Kerätty aineisto koostui vuonna 2014 kerätyistä twiiteistä, joita oli yhteensä seitsemälläkymmenellä (70) eri kielellä [41]. Tämä aineisto oli pakattu tsv-tiedostoksi (*tab-separated values*), joka sisälsi ID-arvoja. Nämä arvot linkitettiin Twitter API:n kautta tekstimuotoisiin twiitteihin, jolloin aineistoa saatettiin käyttää tekstin kielen tunnistamiseen.

Luodussa koneoppimismenetelmiä testaavassa ohjelmassa ei kuitenkaan käytetty kaikkia aineiston seitsemääkymmentä kieltä, vaan työtä varten aineistosta erotettiin viisi kieltä (englanti, japani, ranska, suomi ja viro) omaksi tsv-muotoiseksi tiedostokseen. Erottelu tehtiin tarkoitusta varten kirjoitetun Python-koodin avulla. Kyseiset kielet valittiin niiden ominaisuuksien vuoksi. Haluttiin, että mukana olisi erilaisia kirjainmerkistöjä käyttäviä kieliä, kuten japani ja englanti, sekä mahdollisesti huonosti toisistaan erottuvia kielipareja (suomi–viro) ja hyvin erottuvia kielipareja (ranska–japani).

Kuva 4 *Esimerkki toteutuksessa käytetyn aineiston sisällöstä.*

3.3 Toteutus

Varsinainen koneoppimiesohjelma alkaa edellä mainitun datan sisältämien kielten esittämisellä ohjelmalle sopivalla tavalla. Tämä merkitsee sitä, että jokaista kohdekieltä vastaa ohjelmassa numero. Data ladataan numpy-kirjaston taulukkoon (engl. *numpy array*), jonka jokainen avainarvopari sisältää itse twiitin, sekä sen kieltä kuvaavan numeron. Lataamisen jälkeen data jaetaan opetus- ja testausaineistoihin *train_test_split()*-funktion avulla ja päätetään opetusaineiston kooksi 80% ja testiaineiston kooksi 20%.

Ohjelman seuraavassa vaiheessa tekstistä irrotetaan piirteet. Ohjelmaa ajettiin sekä TF-IDF-, että n-gram-algoritmia suorittavalla piirteenirrotusmenetelmällä. Näistä TF-IDF-menetelmä löytyy scikit-learn-kirjastosta funktiolla *TfidfVectorizer()*, kun taas n-gram-menetelmän saa käyttöönsä *CountVectorizer()*-funktiolla. N-gramin tapauksessa skaalana käytettään ohjelmassa arvoa $(2, 2)$, eli tekstistä poimitaan kahden sanan pareja.

Komentojen putkittaminen merkitsee komentojen ketjuttamista. Putki eli *pipeline* muodostuu komennoista, joissa toisen komennon ulostulo on toisen komennon sisäänmeno. Scikit-learnista löytyy valmis *pipeline*-funktio, jonka hyöty ohjelman toteutuksessa on esimerkiksi se, että ristiinvalidoinnin pystyy toteuttamaan suoraan. Koska komentojen putkittaminen selkeyttää koodia jonkin verran, toteutetaan täs-

sä työssä opetusaineiston opettaminen putken kautta. Luokittelijana käytetään eri ajoissa random forestia, logistista regressiota ja tukivektorikonetta. Tukivektorikoneen käyttämisessä ilmeni kuitenkin ongelmia, kun luokittelija luokitteli kaikki näytteet vain yhteen tai kahteen luokkaan viidestä kohdeluokasta. Näin ollen tukivektorikoneen virheestä tuli TF-IDF:n tapauksessa 0,21 ja n-gramin kanssa 0,65. Kyseinen luokittelija olisi luultavasti vaatinut jonkinlaisen normalisoinnin ennen luokitteluoperaatiota, jotta oltaisiin päästy luotettavaan lopputulokseen tukivektorikonetta käyttäen. Random forest:n luokittelukykyä taas testattiin sekä 10, 100 että 1000 puulla.

Opetusvaiheen jälkeen ohjelmassa ennustetaan testausaineiston avulla luokittelun tulokset *predict()*-funktion avulla. Tätä tulosta käytetään ristiinvalidoinnissa, toisin sanoen lasketaan montako prosenttia twiiteistä on luokiteltu oikein. Ristiinvalidoinnissa käytetään *5-fold cross-valitation*:a Näin kyetään vertailemaan luokittelijoiden suoriutumista tehtävästä. Lisäksi tuloksia arvioidaan confusion matrixin avulla, jolloin saadaan visuaalisesti havainnollistettua luokittelijan toimivuutta.

Ensimmäisessä aineistossa kielten ryhmät oli suhteutettu kyseisen kielen esiintymiseen Twitterissä. Tämä merkitsee sitä, että "englanti" -luokan twiittejä oli huomattavan paljon muihin kieliin verrattuna, mikä johti ohjelman ylioppimiseen. Pienempiä kieliryhmiä, kuten viroa ohjelma ei oppinut lainkaan, sillä opetus- ja testausaineistoihin jakamisen jälkeen vironkielisiä twiittejä oli opetusaineistossa alle kymmenen kappaletta. Näin pienellä opetusaineistolla oppimista ei tapahtunut. Ongelma korjaantui, kun otettiin aineistoksi tiedosto, jossa kutakin kieltä kohden oli suurinpiirtein sama määrä twiittejä, noin kaksisataa. Epätasapainoisen aineiston ongelman voisi myös saada korjattua pienillä muutoksilla ohjelmaan, jos aineistoa vain on tarpeeksi, kuten Fernandez et al. esittää tutkimuksessaan [42].

3.4 Tulokset

Ohjelman tuloksena saatiin eri algoritmeilla taulukoiden 1 ja 2 mukaisia tuloksia. Tästä voidaan päätellä, että parhaaseen tulokseen päästiin logistisella regressiolla, kun käytettiin piirteenirrotukseen TF-IDF-menetelmää. Testatuista luokittelualgoritmeista Random forest ja logistinen regressio saavuttivat tavoitellun, yli 90% tarkkuuden, ja näin ollen testituloksia voidaan pitää onnistuneina. Tukivektorikone sen sijaan tuotti n-gram-menetelmällä noin 62% ja TF-IDF:llä 21% tarkkuuden. Syyn arveltiin olevan ohjelman toteutustavassa ja näin ollen kyseistä algoritmia ei otettu mukaan vertailuun. Random forest:a testattiin 20, 100 ja 1000 puun luokitteluilla, joista paras tulos saatiin odotusten mukaisesti 1000 puulla.

	TF-IDF	n-gram
Logistinen regressio	$[0.95 \pm 0.0047]$	$[0.94 \pm 0.0061]$
Random forest 20 puuta	$[0.90 \pm 0.0054]$	$[0.88 \pm 0.0077]$
Random forest 100 puuta	$[0.94 \pm 0.0054]$	$[0.92 \pm 0.0109]$
Random forest 1000 puuta	$[0.94 \pm 0.0055]$	$[0.93 \pm 0.0084]$
Tukivektorikone	$[0.21 \pm 0.00003]$	$[0.65 \pm 0.101]$

Taulukko 1 *5-fold cross-validation tulokset, [keskiarvo, keskihajonta]*

	TF-IDF					n-gram				
	en	fr	ja	fi	et	en	fr	ja	fi	et
Logistinen regressio										
en	169	3	1	1	0	204	9	0	1	1
fr	5	199	1	0	6	6	182	1	0	5
ja	1	0	185	1	0	1	0	172	0	0
fi	2	1	0	205	6	0	1	0	186	9
et	3	2	2	4	178	2	3	0	4	188
Random forest 1000 puuta										
en	191	7	1	0	2	194	12	1	0	2
fr	7	209	1	0	5	9	186	1	0	8
ja	0	0	178	0	0	1	0	171	0	1
fi	1	2	0	163	18	1	3	0	164	23
et	1	2	0	1	186	1	1	0	1	195

Taulukko 2 *Confusion matrix kahdelle parhaalle luokittelijalle*

Ristiinvalidoinnissa parhaat tulosvektorit saatiin, kun käytettiin piirteenirrotuksessa TF-IDF-menetelmää ja luokittelussa logistista regressiota. Logistisen regression ja 1000-puisen Random forest:n tulokset eivät kuitenkaan eroa toisistaan juurikaan, joten ei voida sanoa että logistinen regressio olisi ollut selkeästi parempi. Sen sijaan TF-IDF sai kaikkien luokittelijoiden kanssa käytettynä paremman tuloksen kuin n-gram, joten sitä voidaan pitää n-gramia parempana luokittelijana kielenoppimisongelmiin.

Confusion matrix:sta käy ilmi, että Random forest -algoritmi sekoitti suomenkieliset twiitit helpommin vironkielisiksi kuin logistinen regressio. Toisaalta piirteenirrotusmenetelmistä TF-IDF näyttäisi kyseisten confusion matrix:en perusteella soveltuvan paremmin annettuun tehtävään kuin n-gram. Merkittäviä eroja ei kuitenkaan havaittu ja varsinkin japaninkieliset twiitit tunnistettiin oletusarvon mukaisesti erittäin hyvin. Tämä johtuu luultavasti siitä, että luokittelijan on helppo tunnistaa japani omaksi kielekseen jo muihin verrattuna erilaisen merkistön perusteella.

4. JOHTOPÄÄTÖKSET

Tässä työssä esiteltiin perinteisiä koneoppimismenetelmiä ja kokeiltiin niitä käytännössä kielentunnistusongelmaan. Selvitettiin myös luokitteluongelmaan käytetyn koneoppimisalgoritmin toiminta esittelemällä aineiston keruu ja käsittely, opetusvaihe, sekä testivaihe. Kielen tunnistusongelman ratkaisuun todettiin tarvittavan kahdenlaisia koneoppimismenetelmiä: piirteenirrotusta ja luokittelijoita. Piirteenirrotuksen osalta esiteltiin TF-IDF ja n-gram, sekä luokittelijoista tukivektorikone, logistinen regressio ja Random forest. Lähteiden perusteella tehdyn taustatutkimuksen nojalla todettiin, että kaikkia tutkittuja menetelmiä voidaan soveltaa kielen tunnistukseen.

Tulosten perusteella voidaan päätellä, että parhaiten tekstin kielen tunnistamiseen sopivat luokittelijoista 1000-puinen Random forest -luokittelija ja logistinen regressio, piirteenirrotusmenetelmistä taas TF-IDF oli kaikilla suorituskerroilla n-gramia parempi. Todettiin myös, että tukivektorikone vaatii toimiakseen erilaiset puitteet, kuin logistinen regressio ja Random forest -luokittelija.

Tutkimusta voisi jatkaa ottamalla tarkasteluun useampia luokittelu- ja piirteenirrotusmenetelmiä. Lisäksi voitaisiin vertailla neuroverkkopohjaisia menetelmiä perinteisiin koneoppimismenetelmiin. Myös tutkimusaineistoa voisi laajentaa käyttämällä aineistona erilaisia tekstidokumentteja lyhyistä twiittien kaltaisista teksteistä satasivuisiin teoksiin. Ongelmanratkaisuun saisi haastavuutta lisäämällä aineistoon pseudokielisiä tai kahta kieltä sisältäviä tekstejä, jolloin luokkia pitäisi muuntaa näiden mukaisiksi.

LÄHTEET

[1] A. Bovet, F. Morone, and H. A. Makse, "Validation of Twitter opinion trends with national polling aggregates: Hillary Clinton vs Donald Trump," *arXiv:1610.01587 [physics]*, Oct. 2016, arXiv: 1610.01587.

[2] J. Carbonell, S. Klein, D. Miller, M. Steinbaum, T. Grassiany, and J. Frey, "Context-Based Machine Translation," *Proceedings of the Association for Machine Translation of the Americas (AMTA-2006)*, pp. 19–28, Jul. 2006.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[4] S. Shalev-Shwartz and N. Srebro, "SVM Optimization: Inverse Dependence on Training Set Size," ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 928–935.

[5] J.-G. Ko, Y.-H. Gil, J.-H. Yoo, and K.-I. Chung, "A Novel and Efficient Feature Extraction Method for Iris Recognition," *ETRI Journal*, vol. 29, no. 3, pp. 399–401, Jun. 2007.

[6] G. Panchal, A. Ganatra, P. Shah, and D. Panchal, "Determination of over-learning and over-fitting problem in back propagation neural network," *International Journal on Soft Computing*, vol. 2, no. 2, pp. 40–51, 2011.

[7] R. Tibshirani, T. Hastie, and J. Friedman, *The Elements of Statistical Learning*, 12th ed. Springer, 2009.

[8] T. Leuhu, "Sentiment analysis using machine learning," Ph.D. dissertation, Jun. 2015.

[9] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, Jan. 2014.

[10] B. G. Gebre, M. Zampieri, P. Wittenburg, and T. Heskes, "Improving Native Language Identification with TF-IDF weighting," in *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, Atlanta, GA, 2013, pp. 216–223.

[11] L. H. Wang, "An Improved Method of Short Text Feature Extraction Based on Words Co-Occurrence," *Applied Mechanics and Materials*, vol. 519-520, pp. 842–845, Feb. 2014.

[12] J. Hakkinen and J. Tian, "n-gram and decision tree based language identification for written words," in *IEEE Workshop on Automatic Speech Recognition and Understanding, 2001. ASRU '01*, 2001, pp. 335–338.

[13] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, and L. Chanona-Hernandez, "Syntactic N-grams as machine learning features for natural language processing," *Expert Systems with Applications*, vol. 41, no. 3, pp. 853–860, 2014.

[14] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, Mar. 2002.

[15] A. A. Akinyelu and A. O. Adewumi, "Classification of Phishing Email Using Random Forest Machine Learning Technique," *Journal of Applied Mathematics; New York*, 2014.

[16] Z. Masetic and A. Subasi, "Congestive heart failure detection using random forest classifier," *Computer Methods and Programs in Biomedicine*, vol. 130, no. Supplement C, pp. 54–64, Jul. 2016.

[17] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, "Variable selection using random forests," *Pattern Recognition Letters*, vol. 31, no. 14, pp. 2225–2236, Oct. 2010.

[18] T. K. Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, Aug. 1995, pp. 278–282 vol.1.

[19] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[20] S. Tong and D. Koller, "Support Vector Machine Active Learning with Applications to Text Classification," *Journal of Machine Learning Research*, vol. 2, no. Nov, pp. 45–66, 2001.

[21] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, Sep. 1999.

[22] B. Schölkopf, "The kernel trick for distances," in *Advances in neural information processing systems*, 2001, pp. 301–307.

[23] V. Rodriguez-Galiano, M. Sanchez-Castillo, M. Chica-Olmo, and M. Chica-Rivas, "Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector

machines," *Ore Geology Reviews*, vol. 71, no. Supplement C, pp. 804–818, Dec. 2015.

[24] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs Up?: Sentiment Classification Using Machine Learning Techniques," ser. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 79–86.

[25] A. B. Musa, "Comparative study on classification performance between support vector machine and logistic regression," *International Journal of Machine Learning and Cybernetics*, vol. 4, no. 1, pp. 13–24, Feb. 2013.

[26] D. B. Springer, L. Tarassenko, and G. D. Clifford, "Logistic Regression-HSMM-Based Heart Sound Segmentation," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 4, pp. 822–832, Apr. 2016.

[27] F. Y. Hsieh, D. A. Bloch, and M. D. Larsen, "A simple method of sample size calculation for linear and logistic regression," *Statistics in medicine*, vol. 17, no. 14, pp. 1623–1634, 1998.

[28] H. G. Lewis and M. Brown, "A generalized confusion matrix for assessing area estimates from remotely sensed data," *International Journal of Remote Sensing*, vol. 22, no. 16, pp. 3223–3235, Jan. 2001.

[29] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

[30] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," vol. 14. Stanford, CA, 1995, pp. 1137–1145.

[31] T. Joachims, "Text categorization with Support Vector Machines: Learning with many relevant features," in *Machine Learning: ECML-98*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Apr. 1998, pp. 137–142.

[32] Y. Reich and S. V. Barai, "Evaluating machine learning models for engineering problems," *Artificial Intelligence in Engineering*, vol. 13, no. 3, pp. 257–272, Jul. 1999.

[33] N. Williams and S. Zander, "Evaluating machine learning algorithms for automated network application identification," Swinburne University of Technology. Centre for Advanced Internet Architectures, Melbourne, VIC, Report, 2006. [Online]. Available: http://researchrepository.murdoch.edu.au/id/eprint/36413/

[34] L. A. Zadeh, I. Guyon, S. Gunn, and M. Nikravesh, *Feature Extraction Foundations and Applications*, ser. Studies in Fuzziness and Soft Computing. Springer, 2006, no. 207.

[35] Anonymous, "TIOBE Index | TIOBE - The Software Quality Company." [Online]. Available: https://www.tiobe.com/tiobe-index/

[36] C. Voskoglou, "What is the best programming language for Machine Learning?" May 2017.

[37] A. Verma, "Most Popular Programming Languages For Machine Learning And Data Science," Dec. 2016.

[38] M. Gibbs, "Updating Twitter with cURL and Wget," no. 25.21, May 2008.

[39] G. Hackeling, *Mastering Machine Learning with scikit-learn*. Packt, Oct. 2014.

[40] Twitter, "About Company." [Online]. Available: https://about.twitter.com/company

[41] Anonymous, "Evaluating language identification performance." [Online]. Available: https://blog.twitter.com/engineering/en_us/a/2015/evaluating-language-identification-performance.html

[42] A. Fernández, V. López, M. Galar, M. J. del Jesus, and F. Herrera, "Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches," *Knowledge-Based Systems*, vol. 42, no. Supplement C, pp. 97–110, Apr. 2013.