AI-lähettiläs-hankkeen prosessikuvaus

AI-lähettiläs -tiimi, Tampereen ammattikorkeakoulu, Tampere 19.5.2021

AI-lähettiläs -hanke on Tampereen ammattikorkeakoulun ja Turun yliopiston kauppakorkeakoulun yhteinen ESR-hanke, joka auttaa yritysten avainhenkilöiden kehittymistä tekoälyn ja data-analytiikan hyödyntämisessä. Hankkeen rahoittajana toimii Suomen rakennerahasto-ohjelma Kestävää kasvua ja työtä 2014–2020.

Tämä dokumentti kuvaa hankkeen keskeiset toiminnot ja tarjoaa vinkkejä kehitystehtävän tekemiseen.

Sisällysluettelo

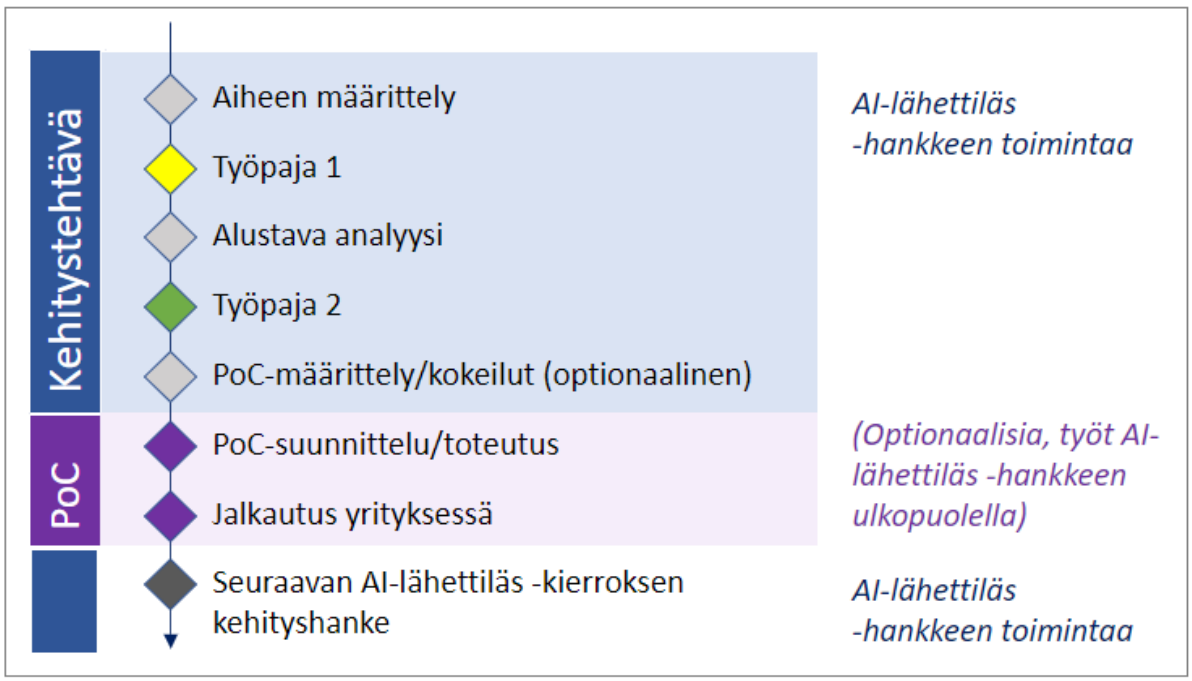
1. Johdanto: AI-lähettiläät, kehitystehtävät ja hankeprosessi

Tekoäly ja data-analytiikka auttavat yrityksiä ymmärtämään asiakkaitaan, kehittämään omia prosessejaan sekä tuottamaan uusia palveluita ja tuotteita. AI-lähettiläs -hanke tukee yritysten avainhenkilöiden kehittymistä tekoälyn ja data-analytiikan hyödyntämisessä (ks. hankkeen kotisivut [1]).

Hankkeen näkökulmasta AI-lähettilään edistyminen tapahtuu pitkälti yrityksen kontekstissa suoritettavan **kehitystehtävän** työstämisen kautta, jonka tuloksia esitellään hanketyöpajoissa. Kehitystehtävän määrittelyä ja etenemistä tuetaan hankkeessa aktiivisesti.

Kehitystehtävän ensisijaisena tavoitteena on tukea yritysten avainhenkilöiden kehittymistä, mutta sopivan aihevalinnan myötä tietenkin myös tarjota eväitä uusien avausten kehittämiseen yrityksissä. Kehitystehtävää voidaan siten AI-lähettiläs -hankkeen toimintojen ulkopuolella jatkaa **yritysten Proof of Concept (PoC) –projekteissa**. (Tämä toiminta ei kuitenkaan kuulu AI-lähettiläs-hankerahoituksen piiriin.)

Oheinen kuvio osoittaa hankeprosessin pääkohdat:



Kuva 1: AI-lähettiläs-hankeprosessi. (Kuvan saavutettava tekstivastine: AI-lähettiläs -hankkeen toiminta käsittää kehitystehtävän, jonka vaiheina aiheen määrittely, työpaja 1, alustava analyysi, työpaja 2 ja valinnaiset PoC-määrittelyt/kokeilut. Varsinainen PoC-kehitys on valinnaista ja tehdään AI-lähettiläs -hankkeen ulkopuolella. Tämän jälkeen on mahdollista lähteä (uuden) kehitystehtävän kanssa hankkeen seuraavalle kierrokselle.)

Kukin AI-lähettiläs omistaa yhden kehitystehtävän. Kehitystehtävän tavoitteena on ideoida ja määritellä konkreettinen, data-analytiikkaa ja/tai tekoälyä hyödyntävä **kehityshanke**. Kehitystehtävän keskeisenä osana tyypillisesti analysoidaan yrityksen liiketoimintamahdollisuuksia, sekä kerätään ja alustavasti analysoidaan kehityshankkeessa tarvittavaa **dataa**.

Datan analysointi ja ymmärtäminen on työn alkuvaiheessa ensisijaisen tärkeää, sillä huomattava osa tekoälyn ja erityisesti koneoppimisen sovelluksista pohjautuu sovellusdatan oivaltavaan hyödyntämiseen: Lähes poikkeuksetta, ennen kuin dataa voidaan osoittaa esim. koneoppimisen algoritmin syötteeksi, tulee ihmissuunnittelijan ensin ymmärtää missä datassa oikein on kyse – ja tarvittaessa esikäsitellä dataa sopivasti.

Kehitystehtävä voi olla luonteeltaan **konseptuaalinen** tai **tekninen**.

Konseptuaalinen kehitystehtävä voi painottua esim. data-analytiikan ja tekoälyn **yrityksen tavoitteen sekä liiketoimintapotentiaalin tunnistamiseen ja kirkastamiseen** yrityksessä, ilman kehityshankkeen puitteissa tehtävää teknistä datan keruu- ja analyysityötä.

Teknisen kehitystehtävän tyypillisenä tavoitteena on puolestaan luoda perusta kehityshanketta seuraavan, teknisen **Proof of Concept (PoC) -projektin määrittelyn ja kokeiluiden** pohjaksi.

AI-lähettiläs-hankkeen ESR-rahoitusohjeista johtuen, varsinainen **yrityksessä tapahtuva yksityiskohtaisempi PoC-suunnittelu ja toteutustyö** tulee kuitenkin rajata AI-lähettiläs-hankkeen ulkopuolelle. (Toteutustyölle on toki mahdollista yhteistuumin tunnistaa muita rahoitus- ja tukimuotoja.)

AI-lähettiläs-hankkeen vuosikellon näkökulmasta toimintojen vaiheistus on seuraava:

1. **Yritysyhteydenotot (Y1Q1-Q2).** Yritysten AI-lähettilään liittyvät hankkeeseen alustavan kehitystehtävän aihepiirin kera.
2. **Kokoontumisajo (Y1Q2).** Esitellään/kerrataan AI-lähettiläs-hankkeen toiminnot, valitut AI-lähettiläät tutustuvat toisiinsa, keskustellaan kehityshankkeiden prosessista ja alustavista aihepiireistä.
3. **Sparrauskeskustelut (Y1Q2-Q3).** AI-lähettiläät täsmentävät kehitystehtävän tarkan kuvauksen ja tarvittaessa käyvät sparrauskeskusteluja hankkeen asiantuntijoiden kanssa.
4. **Ensimmäinen työpaja (Y1Q3).** AI-lähettiläät esittelevät omat kehitystehtävänsä (suunnilleen hankkeen ohjeiden mukaan; ks. kohta ***Kehitystehtävän aiheen määrittely***). Kehitystehtävän data-aineistojen tekninen, alustava analyysityö käynnistyy työpajan jälkeen. Työn tukena on AI-lähettiläs-hankkeen aineistoja ja muita työpajaesityksiä.
5. **Toinen työpaja (Y1Q4).** AI-lähettiläät esittelevät alustavien analyysiensä tuloksia ja täsmennettyjä kehityshankesuunnitelmia sekä mahdollisesti myös jo PoC-jatkokehitysprojektien ideoita (hankeohjeistus tukee alustavien analyysien tekemistä; ks. kohta ***Alustava datan kerääminen ja analysointi***). Työn tukena on AI-lähettiläs-hankkeen aineistoja ja muita työpajaesityksiä.
6. **Proof of Concept (PoC) -määrittelyt (Y2Q1).** AI-lähettiläät tekevät kehitystehtävän tulosten pohjalta PoC-suunnitelmia yrityksissä ja suorittavan aihepiirin kokeiluita teknisillä esimerkeillä. Teknisten esimerkkien kehitystyön tukena on AI-lähettiläs-hankkeen aineistoja ja sparrauskeskusteluita. Varsinainen PoC-työ rajataan kuitenkin AI-lähettiläs-hankkeen ulkopuolelle.
7. **Tulosten jalkautus (Y2Q2).** AI-lähettiläät jatkavat PoC-projekteja yrityksissään. AI-lähettiläs-hankeen seuraava kierros alkaa ja AI-lähettiläitä kutsutaan tutustumaan uusiin AI-lähettiläisiin ja esittelemään omaa työtään. Viimeistään nyt on mahdollisesta lähteä tunnistamaan uusia kehitystehtäviä, joita kehittää tutun AI-lähettiläs-hankeprosessin puitteissa.

Päivitetty versio ja hanke-etappien ja työpajojen tarkat päivämäärät selviävät hankkeen verkkosivuilta [1]. Käytännössä eri AI-lähettiläiden työt etenevät tietenkin hieman eri tahdissa ja esim. työpajojen esitysten aiheita sovitetaan vastaavasti.

2. Ensimmäinen työvaihe: Kehitystehtävän aiheen määrittely

Kehitystehtävän ensimmäisen vaiheen tavoitteena on tunnistaa ja määritellä tärkeä, mielenkiintoinen ja kohtuudella ratkeavaksi arvioitu kehitystehtävä. (Kehitystehtäviä myös esitellään ja niistä keskustellaan ensimmäisessä työpajassa.) Tässä vaiheessa tarkkaa teknistä suunnittelua ei vielä tehdä, vaan ideana on vertailla useita mahdollisia kehitystehtäviä: Alustavan arvion perusteella, näistä ehdokkaista sitten valitaan sopivin ja määritellään se tarkemmin AI-lähettilään kehitystehtäväksi.

Kehitystehtävän aihe on syytä valita ja määritellä riittävän konkreettisesti siten, että kehitystehtävän tavoite on selkeästi tunnistettavissa ja se on ainakin periaatteessa mahdollista saavuttaa. Kehitystehtävän aihetta tulee tietenkin matkan varrella täsmentää, tai ääritapauksessa aiheen voi myös vaihtaa.

2.1. Kehitystehtävän lyhyt määrittely

Kehitystehtävän aihepiirin valintaa ja määrittelyä kannattaa pohtia seuraavan systemaattisen kuvauksen kautta:

"""

Kehitystehtävä: X

Kehitystehtävän työnimi: X

Kehitystehtävän omistavan yrityksen ja sen AI-lähettilään nimi:

Lyhyt kuvaus yrityksestä:

Kehitystehtävän tiivistelmä ja tehtävän rajaus tai ala:

Kehitystehtävän tarve tai liiketoiminnallinen tavoite:

Lyhyt kuvaus valmiin kehitystehtävän tuotteista tai tuloksista (arvio/tavoite):

Mihin liiketoiminnan prosesseihin tai perusjärjestelmiin kehitystehtävän valmiin tuloksen tulisi käyttöönotossa integroitua:

Kehitystehtävän käyttöön tällä hetkellä saatavilla olevat lähdeaineistot ja osaaminen:

Alustava arvio kehitystehtävään liittyvästä teknisestä kehitystarpeesta ja myöhemmin tarvittavasta datasta:

Esimerkkejä kehitystehtävän kanssa samantyyppisistä (analytiikan/tekoälyn/koneoppimisen) sovelluksista maailmalta (jos tiedossa):

Kehitystehtävän alustava aikataulu:

Kehitystehtävän keskeiset sidosryhmät:

Kehitystehtävän keskeisiä haasteita tai riskejä:

Arvioi siitä, miksi tämä kehitystehtävä voisi olisi työhön varattujen resurssien ja osaamisen puitteissa kohtuudella ratkaistavissa:

Kehitystehtävän lyhyt muokkaushistoria (esim. rivejä muodossa pvm/muokkaaja/muutos):

"""

Kehitystehtävän määrittelyn ei tarvitse heti alkuun olla pitkä, ja 1-2 tekstinkäsittelyohjelman sivua riittää hyvin. Mahdolliset tarkemmat tiedot kannattaa koota määrittelyn liitteiksi, joiden kaikkien ei tarvitse olla julkisia. Tärkeintä on työn konkreettisen aiheen ja työn tavoitteen selkä esitys. (Itse asiassa, jos alustava määrittely on liian pitkä, saattaa se jopa haitata kehitystehtävän alkuvaiheen ketteryyttä.)

Kaikkiin määrittelyn kysymyksiin ei työn alkumetreillä ole tietenkään olemassa tarkkoja tai yksityiskohtaisia vastauksia. Tärkeä osa ongelmaa on siten kehitystehtävän täsmentäminen ja sen pohtiminen osa työn isompia liiketoiminnallisia tavoitteita: Kehitystehtävä alustavaa kuvausta tulee tarkentaa konkreettiseksi työsuunnitelmaksi hankkeen edetessä. Viimeistään tämä voi tapahtua Proof of Concept (PoC) -projektin määrittelyn yhteydessä kehitystehtävän lopuksi.

Huomaa, että kehitystehtävän kuvauksen tulisi aluksi keskittyä *mitä* ja *miksi* –kysymyksiin vastaamiseen, ja ottaa *miten*-kysymykseen kantaa vain tarvittavan datan osalta. Teknisiä ideoita kannattaa tietenkin myös kirjata muistiin, mutta jos heti työn alkumetreillä päättää ratkaista ongelman teknologialla se-ja-se, ohjaa tämä helposti ongelmanratkaisua liikaa ja kehitystehtävän varsinainen tavoite on vaarassa hämärtyä.

Kehitystehtävän tavoitteen tai tuloksen tulee joka tapauksessa olla selkeästi ja todennettavissa olevalla tavalla määritelty. Kannattaa muistaa, että ongelmaratkaisu on ihmisasiantuntijallekin mahdollista vain, jos ongelma on ymmärretty oikein ja ratkaisuehdotusten toimivuutta on mahdollista järkevästi testata.

Erityisesti, jos tehtävän ymmärtäminen vaikeaa ihmiselle, voi se koneelle --- tai analyysia tekevälle ja konetta ohjelmoivalle ihmiselle --- olla täysin mahdotonta. Tietokoneet ovat nopeita, niillä on hyvä muisti, ja ne eivät väsy eivätkä kyllästy puuduttavankaan (oppimis)tehtävän suorittamiseen. Toisin kuin ihminen, tietokone ei kuitenkaan tiedä sovelluksesta mitään muuta, kuin mitä datassa on kuvattu. Hieman kärjistäen, (useimmilla nykyisillä) tietokoneilla ei ole lainkaan maalaisjärkeä, hiljaista tietoa, eikä juuri kykyä omaksua luonnollisella kielellä kuvattua monimutkaista informaatiota. Käytännössä tämä siten korostaa täsmällisen tehtävän määrittelyn tärkeyttä --- ja numeerisen datan roolia sovelluksissa.

2.2. Tyypillisiä kehitystoiminnan tavoitteita

Kehitystehtävän tarkka aihe ja tavoite riippuvat tietenkin AI-lähettilään harkinnasta ja kohdeyrityksen tarpeista.

On kuitenkin hyödyllistä tunnistaa yleisiä data-analytiikan ja tekoälyn käyttötapauksia. Tässä tyypillisiä kehitystoiminnan tavoitteita ovat esimerkiksi:

1. (Pitkän aikavälin) historiallisen datan hyödyntäminen organisaation menneen toiminnan ymmärtämiseksi tai tulevan suunnittelun pohjaksi;
2. lähitulevaisuuden tapahtumien ennusteiden tekeminen (lyhyen aikavälin) historiallisen datan pohjalta;
3. organisaation keräämän tiedon organisointi säännönmukaisuuksien tai poikkeamien tunnistamiseksi;
4. suunnitteluratkaisuiden tai mallien optimointi;
5. merkityksellisen uuden tiedon löytäminen; sekä
6. valmiin tekoälyratkaisun liittäminen osaksi organisaation toimintaa.

Näiden lisäksi voidaan tunnistaa myös erilaisia valmistelevia tai tukevia käyttötapauksia, esimerkiksi tarkempaa tuotantodataa keräävän tai eri datalähteitä integroivan "data warehouse" -tyyppisen järjestelmän kehittäminen tai henkilöstön kouluttaminen analyysityökalujen käyttöön.

Huomaa että suunnilleen samantyyppisiä käyttötapauksia löytyy yritysten eri toiminnoista, niin esim. myynnistä ja markkinoinnista, asiakkuuksien hallinnasta, tuotannosta, tuotekehityksestä kuin yrityshallinnostakin. Tyypillisesti menetelmätasolla "vaikeimpia" tehtäviä ovat erilaiset optimointitehtävät ja monimutkaisten aikasarjojen perusteella tehtävät ennusteet, koska näihin liittyy lisämausteena usein haastavia mallinnustehtäviä.

Pitkän aikavälin historiallista dataa voidaan hyödyntää esim. trendien ja ison mittakaavan lainalaisuuksien tunnistamiseen. Tyypillinen esimerkki on vuosittaisten kausivaihteluiden tai kustannusten yleiskorotusten tunnistaminen yksikön toiminnan suunnittelussa. (Esim. regressiosuoran sovitus kustannustrendin tunnistamiseksi.)

Lyhyen aikavälin kerättyä dataa voidaan hyödyntää esim. asiakasprojektien onnistumisen ennustamiseen, vakuutuksen myönnettävän riskin arvioimiseen tai virheellisen tuotteen tunnistamiseen. Tässä erona edellisen käyttötapaukseen on se, että vaikka taustalla voidaankin hyödyntää aineistoa pitkältä ajalta, esim. luokittelu täytyy usein pystyä tekemään nopeasti, lyhyen aikaikkunan sisällä kerätyn kuvaustiedon varassa. (Esim. ohjatun oppimisen avulla havainnoista "reaaliajassa" tehtävät ennusteet.)

Säännönmukaisuuksien tai poikkeamien tunnistamisen tyypillisiä tehtäviä ovat esim. asiakkaiden (tarpeiden) segmentointi ja prosessien pullonkaulojen tunnistaminen. Tunnistamisessa voidaan käyttää myös ennusteita, esim. kuten esim. vaikeiden, asiantuntija-apua tai lisäresurssia tarvitsevien asiakastukipyyntöjen tunnistamisessa. (Esim. klusterointi tai erilaisten sääntöjen ja luokittimien avulla rakennetut "hälytykset".)

Suunnitteluratkaisuiden optimointi voidaan yleisessä tapauksessa käsittää hakutehtävänä. Tässä suunnitteluratkaisun tulee täyttää tietyt laatukriteerit ja suurin haaste on suunnitteluratkaisun ja sen testaamisen sopiva mallinnus (automatisoitua mallien muuntelua ja testausta varten). Tyypillinen esimerkki on jonkin fyysisen komponentin muodon optimointi esim. sen kulutuskestävyyden parantamiseksi, tai painon tai ilmanvastuksen vähentämiseksi. (Esim. geneettiset algoritmit.)

Merkityksellisen uuden tiedon löytäminen perustuu yleensä organisaation datan systemaattiseen ja pitkäkestoiseen puoliautomatisoituun analysointiin, ja sitä saattaa myös tapahtua muiden käyttötapausten "sivutuotteena". Yksinkertainen esimerkki on tilanne, jossa jonkin monimutkaisen

ilmiön, kuten projektin keston tai asiakastilauksen tuottavuuden havaitaan korreloivan jonkin helposti hallittavaa toimintoa kuvailevan attribuutin kanssa. (Esim. datalle kausiluonteisesti automaattisesti ajettavat korrelaatiotestit tai ennusteet.)

Valmiiden tekoälyratkaisuiden liittäminen osaksi organisaation toimintaa on kyseessä silloin, kun kyse on pikemminkin integraatio- kuin analytiikka- tai tekoälyprojektista. Tyypillisiä esimerkkejä ovat valmiiden roskapostisuodattimien, GDPR-dataa tunnistavien komponenttien tai kuvasta tekstiä tunnistavien konenäkösovellusten käyttöönotto. Työ vaatii tällöinkin huolellista testaamista ja vähintäänkin datan esikäsittelyä ja sovelluksen parametrien valintaa, mutta pääpaino on valmiin ratkaisun konfiguroinnissa ja käyttöönotossa (ja ratkaisutoimittajan toteutuksen yksityiskohdat pysyvät usein piilossa).

Tulosten hyväksyttävyyttä arvioitaessa kannattaa myös pitää mielessä, että esimerkiksi data-analytiikan johtopäätökset tai koneoppimisen mallit toimivat vain tietyllä tilastollisella tarkkuudella. Hyvä ennuste- tai luokittelumalli voi esimerkiksi tuhannen ajon tapauksessa antaa keskimäärin yhdeksässä tapauksessa kymmenestä oikean tuloksen --- mutta siten yhdessä tapauksessa kymmenestä (täysin) väärän. Kriittisissä sovelluksissa koneoppimisen väistämättömien virheiden vaikutusten minimointi on aivan yhtä tärkeää kuin ihmistenkin väistämättä tekemien virheiden vaikutusten minimointi.

Lopuksi on syytä todeta, että yritysten sovelluskehitysprojektit sisältävät tietenkin paljon muutakin kuin vain analytiikan tai tekoälyn sovelluksia; esim. heuristisia algoritmeja ja ohjelmistokehitystä, (data-analytiikan ja koneoppimisen) ohjelmistokomponenttien integrointia, arkkitehtuuri- ja systeemityötä, datan varastointia sekä järjestelmien ajonaikaisen laadunvarmistuksen ja jatkuvan kehitystyön ja integroinnin tehtäviä. AI-lähettiläs-hankkeen näkökulmasta nämä aktiviteetit kuitenkin rajataan pääsääntöisesti AI-lähettilään kehitystehtävän ulkopuolelle.

2.3. Tähtää oikean ongelman ratkaisemiseen

Mikäli kehittämistehtävän kohdesovellus ei ole entuudestaan tuttu, on viimeistään tässä vaiheessa syytä ottaa lusikka kauniiseen käteen ja tutustua sovellukseen pintapuolista katselmointia tarkemmin:

1. Millainen prosessi tosiasiallisesti on kyseessä ja minkä prosessin vaiheen tulisi hyötyä kehitystehtävän ratkaisusta? Miten?
2. Mitkä työntekijät, työkalut ja teknologiat ovat prosessissa osallisina?
3. Miten sovellusalueen ammattilaiset, asiantuntijat, tai asiakkaat ratkaisisivat kehitystehtävän? Onko heiltä kysytty ja mitä he vastasivat?
4. Mihin tietoon ja dataan ratkaisu voisi perustua?
5. Millaisia ratkaisuyrityksiä aikaisemmin on tehty? Miksi ne eivät ole onnistuneet riittävän hyvin --- mikä on "uuden" kehitystehtävän keskeinen haaste?

Huomaa, että organisaatioiden de facto –prosessit joskus hieman poikkeavat virallisista prosessikaavioista. Vastaavasti, näkemykset ja tietämys yrityksen eri toiminnoista ovat usein rajallisia ja toisinaan hieman siiloutuneitakin: Paras tapa saada varmaa tietoa jostakin toiminnosta, on siten kysyä siitä suoraan kyseisiltä työntekijöiltä eikä luottaa epäsuoraan tietoon tai yleiseen käsitykseen asiasta.

Vallitsevia, usein pitkän ajan kuluessa kehittyneitä hyviä käytänteitä ei tietenkään kannata heppoisin perustein haastaa tai yrittää "ratkaista" negatiivisessa mielessä. Selvitysten ja keskusteluiden tavoitteena tulisi yksinkertaisesti olla hakea tietoa ja yrittää rakentavasti löytää oman kehitystehtävän

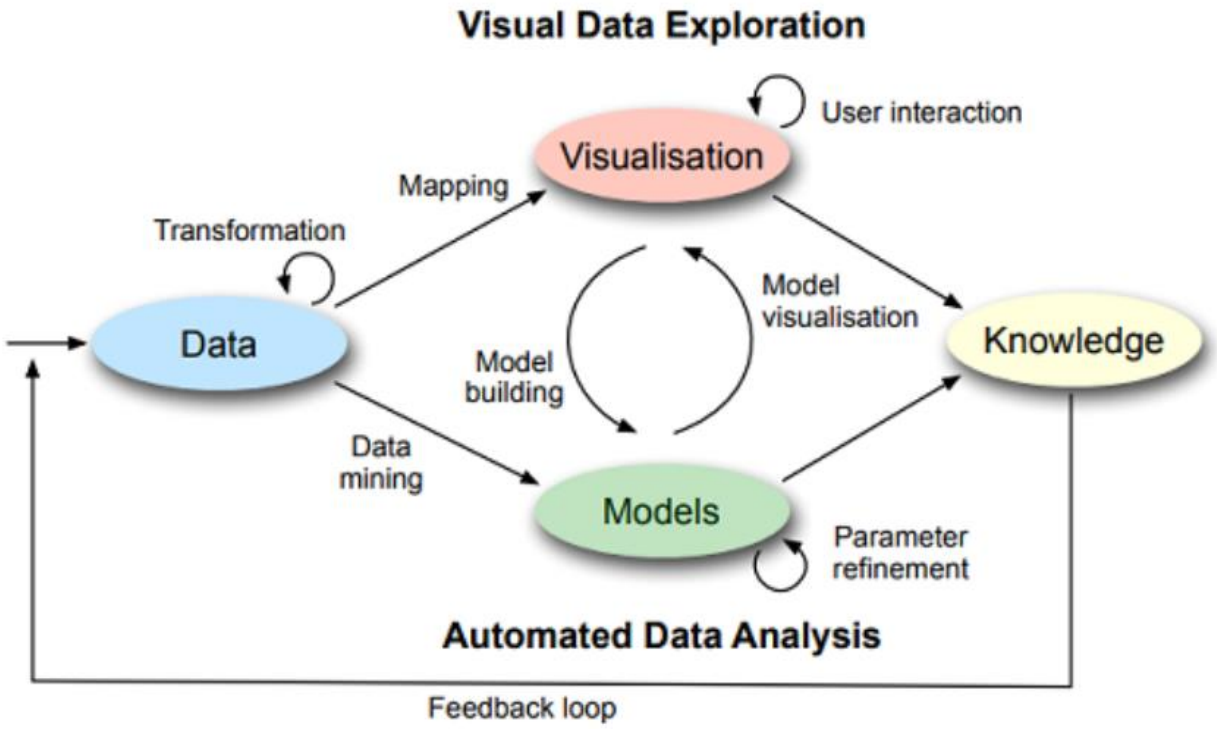
ratkaisuun eväitä. (Vakiintuneita toimintoja kannattaa vakavissaan arvioida vasta kehitystehtävän ja mahdollisen Proof of Concept-projektin tulosten myötä.)

Kehitystehtävän keskeisen tavoitteen analyysiä ja ennen kaikkea siitä keskustelemista tyypillisesti helpottaa, jos piirtää työn tueksi esimerkiksi visuaalisesti intuitiivisen mielle- tai käsitekartan (aihepiirin vapaa assosiointi) tai kalanruotodiagrammin (ratkaisuun ensisijaiset ja toissijaiset vaikuttavat tekijät) [2, 3].

Olipa analyysin tarkka muoto sitten mikä tahansa, ei sovelluksen tavoitteiden ja vallitsevan nykytilan ymmärtämistä tule koskaan ylenkatsoa. Oikeastaan kehitystehtävän kaikki seuraavat vaiheet perustuvat sovelluksen ja sen tarpeiden ymmärtämiseen. Jos kohdesovellus tunnetaan huonosti, on vaarana, että kehitystyössä jätetään huomiotta jokin kriittisen tärkeä osa-alue, (liiketoiminnan) reunaehto tai "jo kaikkien tietämä" parannusehdotus, tai että kehitettävät tulokset eivät ole käytännössä hyväksyttäviä tai merkittäviä.

3. Toinen työvaihe: Alustava datan kerääminen ja analysointi

Teknisen kehitystehtävän toisessa vaiheessa on aika mennä sanoista tekoihin ja kerätä kehitystehtävään perustana tarvittavaa dataa ja alustavasti analysoida sitä. (Alustavan analyysin tuloksia myös esitellään ja niistä keskustellaan toisessa työpajassa.)



Kuva 2: Visuaalinen data-analytiikkaprosessi (kuvalähde [4]). (Kuvan saavutettava tekstivastine: Visual data exploration and automated data analysis includes transforming data and mapping it into visualisations and mining it into models. Visualisations may include user interactions and contribute into understanding the domain. Models require parameter refinement and also contribute into

understanding the domain. Visualisation and modeling activities are also related, including a feeback loop between model building and model visualisation.)

Datan keräämisellä, analytiikalla ja datasta rakennettavilla visualisoinneilla pyritään lisäämään ymmärrystä sovelluksesta ja kehitystehtävän tarpeista. Lisääntynyt ymmärrys puolestaan kertoo myös, minkälaista dataa tulisi kerätä (lisää) ja kuinka sitä kannattaisi kukaties vielä paremmin analysoida. Tämä prosessi osoittaa luonnollisen, eksploratiivisen ja iteraatiivisen rakenteen analyysien tekemiselle (vrt. visuaalisen data-analytiikan yksinkertaistettu prosessikuvio yllä).

Visualisoinneilla on datan hahmottamisessa keskeinen rooli, jonka taustalla vaikuttaa ihmisen kognitioon liittyviä tekijöitä. Ihmisten on usein paljon helpompaa tunnistaa asioista kuvista kuin esim. numeerisista taulukoista. Tämän perustana on ihmiselle luontainen taipumus esim. erottaa kuvista tai äänistä (ym.) "objekteja" tilanteesta riippuvaa "taustaa" vasten, tunnistaa samankaltaisuuksia, kuvioiden läheisyyttä, alueita, jatkuvuutta, nähdä kokonaisuuksia yksityiskohtien sijaan sekä huomata poikkeamia --- kuva todellakin kertoo enemmän kuin tuhat sanaa!

Käytännössä osa visualisoinneista nähtävistä mielenkiintoisista piirteistä voi toki olla myös näennäisiä tai osoittaa virheitä datassa (so. kuvassa näkyy jotain, mitä ilmiö ei tosiasiallisesti sisällä, esim. koristekuvioihin tai mittaustulosten pyöristyksiin liittyen). Tärkeät potentiaaliset löydökset tulee siten aina varmistaa useammalla kuin yhdellä tavalla.

Kun alustava analysointi ja kerätty data ovat riittävällä tasolla, on datan avulla mahdollista myös jatkaa tiedonlouhinnan ja koneoppimisen sovelluksiin. Näissä keskeistä on sopivan algoritmin ja sen parametrien valinta, sekä mahdollisuus arvioida tulosten paikkansapitävyyttä. Myös tiedonlouhinnan ja koneoppimisen tulokset kaipaavat usein visualisointia ja lisäävät aihepiirin ymmärrystä, minkä ansiosta esim. koneoppimisen tulokset saattavat myös nostaa esille ideoita tai tarpeen lisädatan keräämiselle.

Tyypillinen analytiikan ja koneoppimisen virhe on pyrkiä etenemään alustavissa kokeiluissa liian nopeasti. Eksoottisen näköisestä kuviosta tai mallinnuksesta ei kuitenkaan ole juuri hyötyä, ellei ymmärretä mitä se tarkkaan ottaen tarkoittaa.

Alustavan analytiikan ja visualisointien tavoitteena on rakentaa ymmärrykselle vakaa ja objektiivinen perusta, lähtien liikkeelle yksinkertaisista tunnusluvuista ja visualisoinneista. Vasta kun tämä perusta on kunnolla valettu, kannattaa siirtyä monimutkaisempien menetelmien pariin.

3.1. Alustavan datan kerääminen

Data-analytiikan ja tekoälyn sovellukset perustuvat koneluettavaan dataan. Datan tulee olla teknisesti sopivassa muodossa ja dataa tulee olla riittävästi.

Vaikka kyse on vielä alustavasta datasta, kannattaa heti alkuun pyrkiä siihen, että data olisi mahdollisimman edustavaa, eli että haluttu ilmiö olisi mahdollisimman hyvin läsnä datassa. Hienosti sanottuna, data otannan tulisi olla hyvä ja sopivasti edustaa ilmiön "koko" havaintoyksikköpopulaatiota [5].

Intuitiivisesti alustavan datan hyvyys tarkoittaa sitä, että valitusta datasta on mahdollista piirtää taulukkolaskentaohjelmista tuttuja kaavioita, joilla on mielekäs tulkinta --- ja jonka voidaan tietyissä rajoissa ajatella yleistyvän koko ilmiön kuvailemiseen. Työhön ryhdyttäessä ei kuitenkaan ole yleensä

selvillä, minkä tyyppistä dataa tarvitaan. Kehitystyössä tulee siis varautua siihen, että dataa joudutaan keräämään matkan varrella sekä enemmän että eri näkökulmista käsin.

Alustavan datan tarkka sisältö, rakenne ja koko riippuvat tietenkin kehitystehtävästä, mutta useimmissa tapauksissa keskeinen osa datasta kannattaa aluksi puhdistaa seuraavaan taulukkomuotoon, jossa on N riviä ja M saraketta (esim. MS Excelissä tai LibreOffice Calc -taulukkolaskentaohjelmassa, jotka päättävät esim. sarakkeiden erotinmerkit, lukujen ja päivämäärien esitystavan, ym.):

attribuutti1, attribuutti2, attribuutti3, …, attribuuttiM

arvo11, arvo12, arvo13, …, arvo1M

arvo21, arvo22, arvo23, …, arvo2M

…

arvoN1, arvoN2, arvoN3, …, arvoNM

Taulukossa yksittäinen rivi kuvaa aina täsmälleen yhden havaintoalkion tai instanssin, käyttäen kuvaukseen M kpl oivaltavasti valittuja attribuutteja. Tapauksesta riippuen, datarivit voivat esim. kuvata asiakkaiden ostotapahtumia, tarjouksia, liukuhihnalta kuvattuja tuotteita tai työkoneesta tietyn aikavälin tehtyjä mittauksia ja muutoksia. Koko aineisto voi siten kuvata esim. kaupan myyntihistorian, asiakkaille lähetettyjen tarjousten versiohistorian, valmistusprosessin laadunvalvonta-aineiston tai työkoneen käyttö- ja huoltohistorian.

Datan osana voidaan kuvata paitsi mittauksia, myös esim. tietoja prosessin eri vaiheista ja siihen osallistuneista toimijoista aikaleimoineen. Tämän avulla data voidaan tarvittaessa kytkeä osaksi suurempaa toimintokokonaisuutta ja erityisesti muita, vastaavantyyppisiä datataulukoita. Analytiikkaohjelmistot osaavat käsitellä myös relaatiotietokannoista tuttuja taulukoiden yhdisteitä käyttämällä tiettyjä attribuutteja (vieras)avaimina, mutta on usein aluksi paljon helpompaa tarkastella vain yksittäisiä ei-normalisoituja taulukoita (esim. taulukkolaskentaohjelman tallennustiedoston tai Unicode UTF-8 -koodatun csv-tiedoston muodossa). Helppouden hinta tässä on tietenkin ei-normalisoitujen taulukoiden toisteisuus ja suuri koko.

Huom. Aineiston havaintoyksiköitä kuvaavien attribuuttien valintaan ja koodaukseen kannattaa pureutua erityisen huolella, sillä yleensä juuri tällä työ- ja mallinnusvaiheella on ratkaisevan tärkeä rooli kaikissa seuraavissa työvaiheissa!

Kehitystyössä tulee yleensä varautua siihen, että dataa joudutaan keräämään matkan varrella sekä enemmän (enemmän datarivejä) että eri näkökulmista käsin (erilaisia attribuutteja, kukaties niitäkin enemmän). Kannattaa myös muistaa, että tietokoneohjelmat usein esittävät murto- ja reaalilukuja vain likimääräisesti ns. liukulukuina, esim. 15 desimaalin tarkkuudella. Useimmissa tapauksissa tämä riittää sovellusten tarpeisiin mainiosti. Näennäisestä tarkkuudesta huolimatta, lukujen likiarvoihin perustuva laskenta saattaa silti useiden välivaiheiden myötä johtaa epäintuitiivisiin pyöristysvirheisiin jopa varsin yksinkertaisilta vaikuttavissa laskutehtävissä. (Siten esim. tuttu taulukkolaskentaohjelma ei vaativassa käytössä välttämättä lainkaan sovellu suurta tarkkuutta vaativiin fysiikan tai finanssilaskelman iteratiivisiin laskutehtäviin.)

Käytännössä datan tekninen sopiva muoto riippuu työkalusta, mutta nyrkkisäännön tasalla data tulisi olla luettavissa taulukkolaskentaohjelmaan siten, että luvut ja päivämäärät on esitetty oikein.

Peukalosäännön tasolla, alle 30 datarivin otosaineistot ovat yleensä aivan liian pieniä määrälliseen (kvantitatiivisen) tai tilastollisen analyysin perustaksi. Tämä selittyy sillä, että jos ja kun pienen otoksen havaintoyksiköt valitaan asianmukaisen satunnaisesti, on varsin mahdollista, että myös otoksen perusteella tehdyt sinänsä pätevät päätelmät pätevät myös vain sattumalta. Toisaalta, mikäli tarkoitus on laskea otosaineiston perusteella "vain keskiarvoja" (esim. mielipidekyselyt), satunnaisotantaa ei yleensä ole tarpeen suorittaa noin tuhatta havaintoyksikköä enempää; tämän jälkeen tutkimuksen tarkkuus ei enää sovellusten muiden virheiden --- ja otosaineiston keräämisen kustannusten --- suhteen enää järkevästi parane.

Yksinkertaisetkin koneoppimisen sovellukset sen sijaan vaativat yleensä vähintään satoja tai tuhansia datarivejä. Intuitiivisesti sanottuna, näissä on usein kyse jonkin keskiarvoja monimutkaisemman säännönmukaisuuden tai hahmon tunnistamisesta, ja mitä hienosyisempi ilmiö on kyseessä, sitä enemmän dataa tarvitaan. Tarvittavien datarivien lukumäärään vaikuttaa suoraan myös kuvailevien attribuuttien lukumäärä ja laatu: Edelleen nyrkkisäännön tasolla, dataa mallintavassa taulukossa tulisi olla useita kertaluokkia enemmän rivejä kuin sarakkeita. Tästä syystä esimerkiksi kuvantunnistuksen tehtävissä dataa tarvitaan usein kymmeniä- tai satojatuhansia datarivejä --- koska pientenkin kuvien koodaamiseen datariveiksi käytetään helposti satoja attribuutteja.

Mikäli valmista dataa ei ole helposti saatavilla, datan kerääminen, muokkaaminen sopivaan muotoon ja puhdistaminen, voivat olla hyvinkin hankalia ja työläitä vaiheita. Tyypillisiä datalähteitä ovat erilaisten tuotantojärjestelmien export-tiedostot ja lokitiedot, järjestelmärajapintojen tietokantakyselyiden tulokset ja tiedostojärjestelmistä eri tavoin kerätty data (esim. pienen ryömijä-apuohjelman tuottama tuloste). Hankalimmissa tapauksissa dataa joudutaan koodaamaan ja siivoamaan "käsin". Tässä kannattaa pitää mielessä se, missä määrin esikäsittelytehtävää on mahdollista automatisoida kehitystehtävän tuotantokäytössä myöhemmin --- tai onko se edes realistista. Erityisestä huomiota tulee kiinnittää asiantuntijan harkintaa vaativien esikäsittelyaskelten toistettavuuteen tuotantokäytössä. Riskinä on se, että ihmisasiantuntija suorittaa esikäsittelyn osana alitajuisesti esim. luokittelua (kuvailevissa attribuuteissa), jonka automatisointi ei tuotannossa tietokoneohjelmalta sitten onnistukaan.

Alla esimerkkinä klassinen, pieni ja siisti iiris-aineisto, jota käytetään usein helppona opiskelun, analytiikan ja luokittelevan koneoppimisen esimerkkinä [6].

Iiris-aineistossa on kuvattu 150x5 –kokoiseen taulukkoon 150 eri mitatun yksittäisen kukkasen verho- ja terälehtien pituudet (sepal=verholehti, petal=terälehti) sekä asiantuntijan tunnistama kukkalaji (class). Tässä tyypillisen kehitystehtävän tavoite on käyttää ensimmäisiä "mekaanisesti kerättyjä", kuvailevia attribuutteja viidennen luokka-attribuutin ennustamiseen. Taulukkoa edeltävä, tiedoston ensimmäinen rivi on metatietoa, joka nimeää attribuutit niin että ihmisten on helpompaa keskustella datasta.

Iiris-datataulukko näyttää seuraavalta:

sepalLengthCm, sepalWidthCm, petalLengthCm, petalWidthCm, class

5.1,3.5,1.4,0.2,Iris-setosa

4.9,3.0,1.4,0.2,Iris-setosa

4.7,3.2,1.3,0.2,Iris-setosa

…

7.0,3.2,4.7,1.4,Iris-versicolor

6.4,3.2,4.5,1.5,Iris-versicolor

6.9,3.1,4.9,1.5,Iris-versicolor

…

6.5,3.0,5.2,2.0,Iris-virginica

6.2,3.4,5.4,2.3,Iris-virginica

5.9,3.0,5.1,1.8,Iris-virginica

Huomaa että esimerkissä datarivien (toiselta riviltä alkaen) viimeisen sarakkeen arvo (class-attribuutin arvo, esim. "Iris-setosa") ei ole numeerista tietoa. Käytännössä data-analytiikan sovelluksissa tämäntyyppiset tekstinpätkät kuitenkin nekin korvataan sisäisesti usein kokonaisluvuilla (tässä esim. 0, 1 ja 2). Luonnollista kieltä sisältäviä attribuuttiarvoja voidaan toki myös analysoida, mutta tähän käytetään yleensä säännönmukaisiin lausekkeisiin tai valmiisiin NLP-komponentteihin perustuvia esikäsittelyratkaisuita esim. englanninkielisen tekstin analysointiin (NLP=Natural Language Processing.)

Ts. tässä tapauksessa yksinkertaisen tietokoneohjelman näkökulmasta esim. "Iris-setosa" ei sinänsä tarkoita mitään muuta kuin että se on *eri* merkkijono kuin "Iris-versicolor". Ensimmäinen datarivi tulkitaan siten esim. seuraavasti:

5.1,3.5,1.4,0.2,**0**

Tästä huomataan, että numeerisellakin datalla voi olla perustavaa laatua olevia tulkintaeroja, liittyen asteikkoihin, joiden avulla erityyppisiä lukuja on mielekästä tulkita. Erityisesti, on tärkeää pysyä kärryillä siitä, onko numeerisessa datassa kyse luokittelu-, järjestys-, välimatka-, tai suhdeasteikollisista [7] attribuuteista tai muuttujista. Suhdeasteikon perusteella on esimerkiksi perusteltua sanoa, että 7 cm pitkä verholehti on kaksi kertaa niin pitkä kuin 3.5 cm pitkä verholehti. Ei kuitenkaan ole järkevää ajatella, että luokitteluasteikon arvoon 2 sattumalta koodattu Iris-virginica -havaintoyksikkö olisi mielekkäästi "kahden mittayksikön päässä" arvoon 0 koodatusta Iris-setosa -havaintoyksiköstä (vaikka tällainen vähennyslasku 2-0=2 sinänsä on toki helposti määriteltävissä).

Aikasarja-aineistoissa yksi kuvailevista attribuuteista on tyypillisesti aikaleima, jonka mukaan aineisto voidaan luonnollisesti sijoittaa aikajanalle ja tutkia esim. kausivaihteluita. Aikaleimojen koodauksissa on eri maissa ja ohjelmistoissa huomattavia eroja, joten kannattaa varmistaa, että tietojen kirjoittaja ja lukija ovat koodauksesta yhtä mieltä. Standarditapa on kirjoittaa tiedot ns. DateTime-muodossa, esim. 2002-05-30T09:30:10Z, missä merkkijonon viimeinen merkki Z ("Zulu") kertoo että aika(vyöhyke) on ilmoitettu UTC-muodossa.

Oman datan suunnitteluun on tyypillisesti hyödyllistä hakea ideoita ja aineiston koodausvinkkejä myös valmiista esimerkkiaineistoista, vrt. esim. [8, 9].

3.2. Alustavia, koeluonteisia visualisointeja

Kun säällinen otos alustavaa dataa on saatu kerättyä ja puhdistettua, kannattaa siitä systemaattisesti tuottaa datan ja sitä kautta ongelman ymmärtämistä helpottavia alustavia tai eksploratiivisia visualisointeja.

Työn tekemiseen soveltuu aluksi tuttu taulukkolaskentaohjelma. Tehokkaampi, mutta aluksi lisäopiskelua edellyttävä ratkaisu, on käyttää Tableau (Public) [11] tai MS Power BI [12] –tyyppistä analytiikkaohjelmaa, tai sitten voi hypätä suoraan syvään päähän ja ohjelmointiin esim. Anaconda (Individual)/JupyterLab/Python [13] -tyyppisen lähestymistavan avulla. Tämän luvun esimerkkivisualisoinnit on tehty Tableau Public ja Python -työkaluilla, sopivia apukirjastoja hyödyntäen (esim. pandas, sckikit-learn ja matplotlib). AI-lähettiläs-hanke tarjoaa aloitusohjeen Tableau-työkalun käytöön.

Alustavissa analyyseissä on tyypillisesti hyödyllistä tuottaa seuraavia tietoja (attribuuttien tyypit huomioiden, vrt. esim. luokittelu- vs. suhdeasteikko):

- Datarivien ja attribuuttien lukumäärät (jo senkin takia, että nähdään, että data ladattiin työkaluohjelmaan oikein);
- Attribuuttien keskiarvot, mediaanit, moodit ja kvartiilit (sijaintiluvut);
- Attribuuttien pienimmät ja suurimmat arvot sekä keskihajonnat (standardipoikkeamat); sekä
- Luokitteluasteikollisista attribuuteista on yleensä mielekästä selvittää ainakin moodit eli tyyppiarvot, sekä se, mitä kaikkia eri arvoja aineisto oikeasti sisältää.

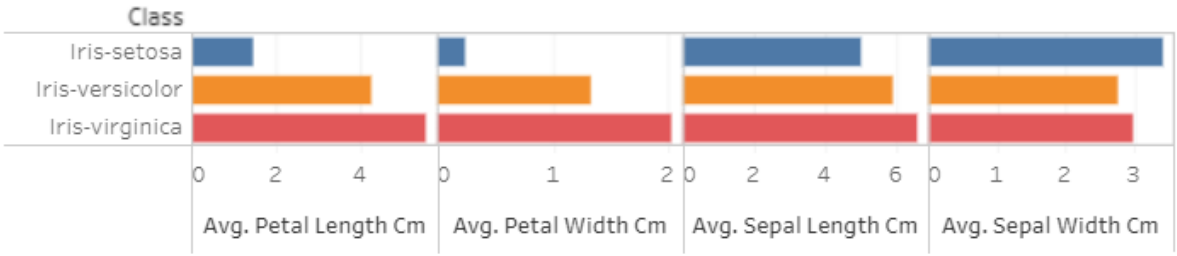
Seuraavassa esimerkki iiris-aineiston "perustiedoista". Metatietorivi poislukien, aineisto sisältää siis 150 datariviä ja neljä kuvailevaa attribuuttia (sepal=verholehti, petal=terälehti). Ao. kuvat ovat lähteestä [10]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Kuva 3: Aineiston tunnuslukuja (kuvalähde [10]). (Kuvan saavutettava tekstivastine: Esimerkki 8*4 taulukosta, jossa kuvattu iiris-aineiston kuvailevien attribuuttien sepal_length, sepal_width, petal_length ja petal_width tunnusluvut count, mean, std, min, 2%, 50%, 75% ja max. Tunnuslukujen tarkat numeeriset arvot eivät ole tässä oleellisia.)

Huomaa, että luokitteleva attribuutti class on tässä jätetty pois (arvot Iris-setosa, Iris-versicolor ja Iris-virginica --- tai 0, 1 ja 2).

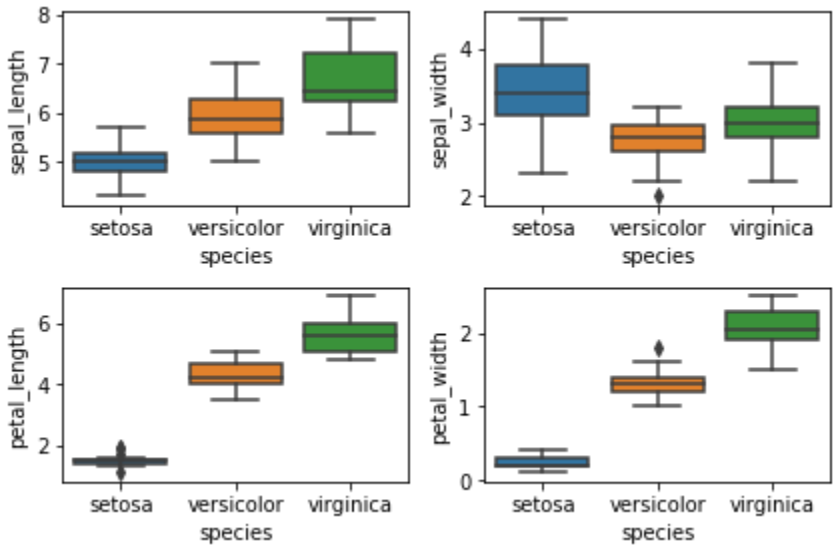
Numeerinen tieto on analyysin perustana, mutta ihmisten voi olla hyvin vaikeaa saada siitä kokonaiskuvaa. Tulkinta helpottuu esim. pylväsdiagrammien avulla, joilla voidaan periaatteessa havainnollisesti esittää mitä tahansa numeerista tietoa, esim. tässä keskiarvoja:



Kuva 4: Pylväsdiagrammeja. (Kuvan saavutettava tekstivastine: Neljä pylväsdiagrammiryhmää, joissa kuvattuna luokkien Iris-setosa, Iris-versicolor ja Iris-virginica kuvailevien attribuuttien Petal Length Cm, Petal Width Cm, Sepal Length Cm ja Sepal Witdh Cm keskiarvot. Kuvasta erityisesti nähdään, että attribuutin Petal Length Cm keskiarvo luokalle Iris-setosa on selvästi pienempi kuin luokalle Iris-versicolor, joka puolestaan on hieman pienempi kuin luokan Iris-virginica attribuutin Petal Length Cm keskiarvo.)

Huomaa että edellä pylväiden suunta on valittu vaakatasoon siten, että tekstin lukeminen on helppoa. Kuvaajien asteikot alkavat nollasta ja värejä on käytetty havainnollistamaan eri luokkia.

Monessa tapauksessa jakaumien keskiarvoja ja kvartiileja (sijaintilukuja) voi vielä mukavammin havainnollistaa esim. boxplot-kuvioiden muodossa:

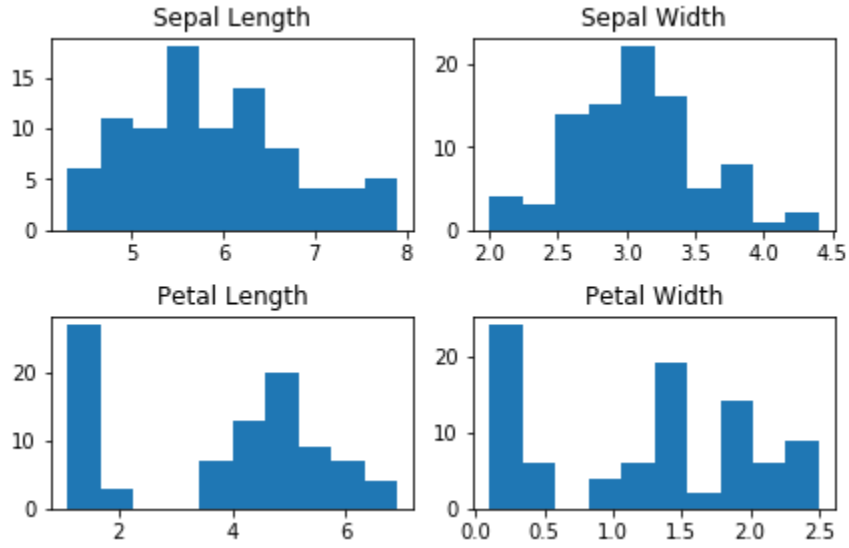


Kuva 5: Boxplot-diagrammeja (kuvalähde [10]). (Kuvan saavutettava tekstivastine: Esimerkkejä Iiris-aineistoa kuvailevien attribuuttien sepal_length, sepal_width, petal_length ja petal_width tunnuslukuja havainnollistavista boxplot-kuvioista. Boxplot-kuvio havainnollistaa graafisesti jakauman viisi tunnuslukua: pienimmän ja suurimman arvot viivoina ja kolme kvartiilia (25%, mediaani eli 50% ja 75%) laatikkona. Tunnuslukujen summittaiset numeeriset arvot eivät ole tässä oleellisia.)

Em. kuvaajista käy esim. ilmi, että attribuutti petal length jakaa aineiston kauniisti kolmeen luokkaan, eli se sopii sellaisenaan esim. luokan ennustamiseen. Ikävä kyllä, aineistot eivät yleensä toki ole näin "helppoja"!

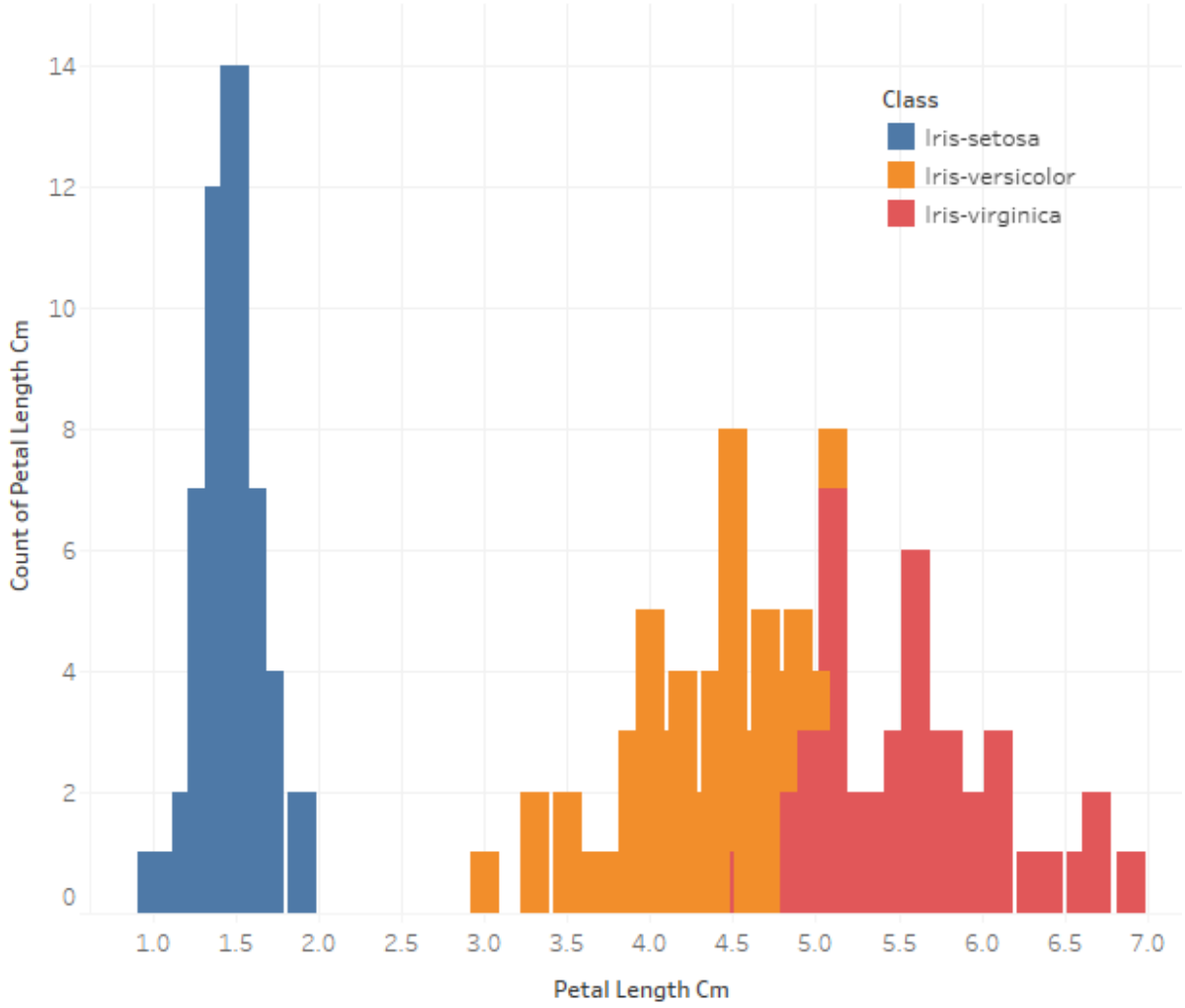
Huomaa että aineiston nimilappujen tai värikoodauksen vaihtaminen kesken analyysin tyypillisesti hankaloittaa intuitiivisten tulkintojen tekemistä. (Edellä iris-virginica -luokan värikoodaus vaihtui punaisesta vihreään.)

Sijaintiluvut ovat täsmällisiä, mutta aineiston attribuuttijakaumista saa intuitiivisesti havainnollisemman kuvan erilaisten attribuuttihistogrammien avulla:



Kuva 6: Histogrammeja (kuvalähde [10]). (Kuvan saavutettava tekstivastine: Iiris-aineistoa kuvailevien attribuuttien Sepal Length, Sepal Width, Petal Length ja Petal Width jakaumia havainnollistavat histogrammit. Kuvioista havaitaan, että histogrammien muodot vaihtelevat aineiston mukaan yksi-, kaksi- tai kolmi-"kyttyräisinä". Pylväiden summittaiset numeeriset arvot eivät ole tässä oleellisia.))

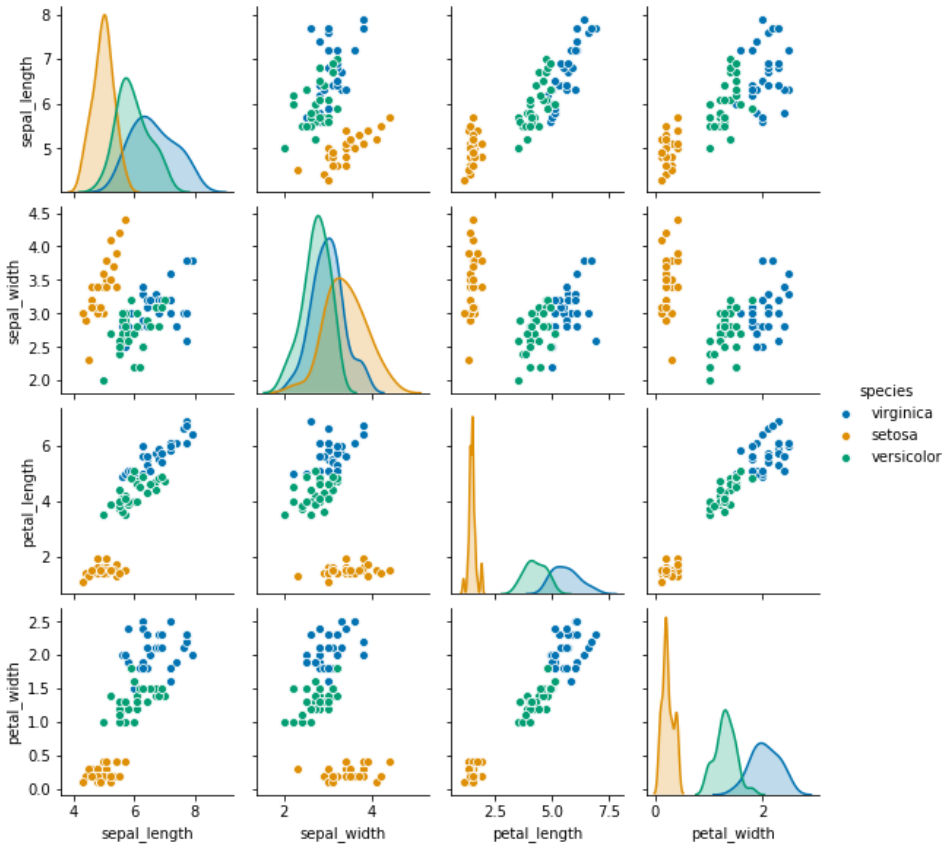
Histogrammeja voi myös värittää esim. luokittelutiedolla; seuraavassa iiris-aineiston petal length – attribuutin jakauma tarkemmin uudelleen, tällä kertaa väritettynä luokka-attribuutin avulla:



Kuva 7: Histogrammi sekoittuneiden jakaumien havainnollistamisessa. (Kuvan saavutettava tekstivastine: Iiris-aineiston Petal Length Cm –attribuutin histogrammi, jonka pylväät on merkattu värikoodein luokkien Iris-setosa, Iris-versicolor ja Iris-virginica mukaisesti. Ilman värikoodausta histogrammi sisältäisi vain kaksi komponenttia Iris-setosa vs. muut luokat, mutta värikoodauksen ansiosta myös toisiinsa sekoittuneet luokat Iris-versicolor ja Iris-virginica erottuvat melko terävärajaisesti.)

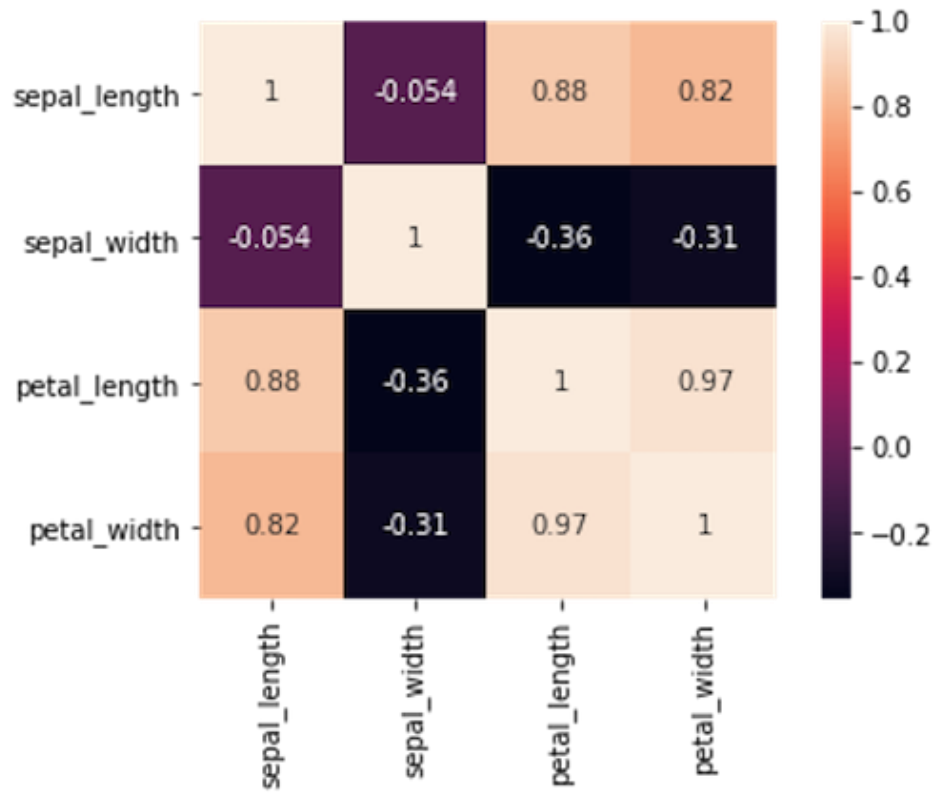
Yo. kuvasta nähdään, että jakauma sisältää kolme komponenttia, joista kaksi esiintyy datassa toisiinsa sekoittuneena. Tämä on tyypillistä luokittelutehtävissä ja analysoinnin tavoitteena on usein juurikin luokkien "löytäminen toisiinsa sekoittuneista aineistoista".

Attribuuttien korrelaatioiden systemaattinen analyysi ja visualisointi on myös hyödyllistä, sillä se helpottaa tärkeiden attribuuttien tunnistamista. Seuraavasta scatterplot- ja histogrammikuviosta käy ilmi useita säännönmukaisuuksia iiris-aineistossa (värikoodaus kannattaisi taaskin valita systemaattisesti):



Kuva 8: Systemaattinen taulukko scatterplot- ja jakaumakuvioita. Kuviot (16 kpl) havainnollistavat aineistoa kaikkien kuvailevien attribuuttiparien avulla (4x4 kpl) laskettuna (kuvalähde [10]). (Kuvan saavutettava tekstivastine: 4*4 kappaletta luokittain värikoodattuja scatterplot- tai jakaumakuvioita Iris-aineiston kuvailevien attribuuttien sepal_length, sepal_width, petal_length ja petal_width kaikilla kombinaatioilla. Kuvion perusteella voidaan epäillä, että esim. attribuuttien petal_length ja petal_width välillä vallitsee voimakas positiivinen lineaarinen korrelaatio.)

Korrelaatioita voidaan tarkemmin tutkia myös laskennallisten korrelaatiokertoimien avulla (positiivinen vs. negatiivinen korrelaatio):

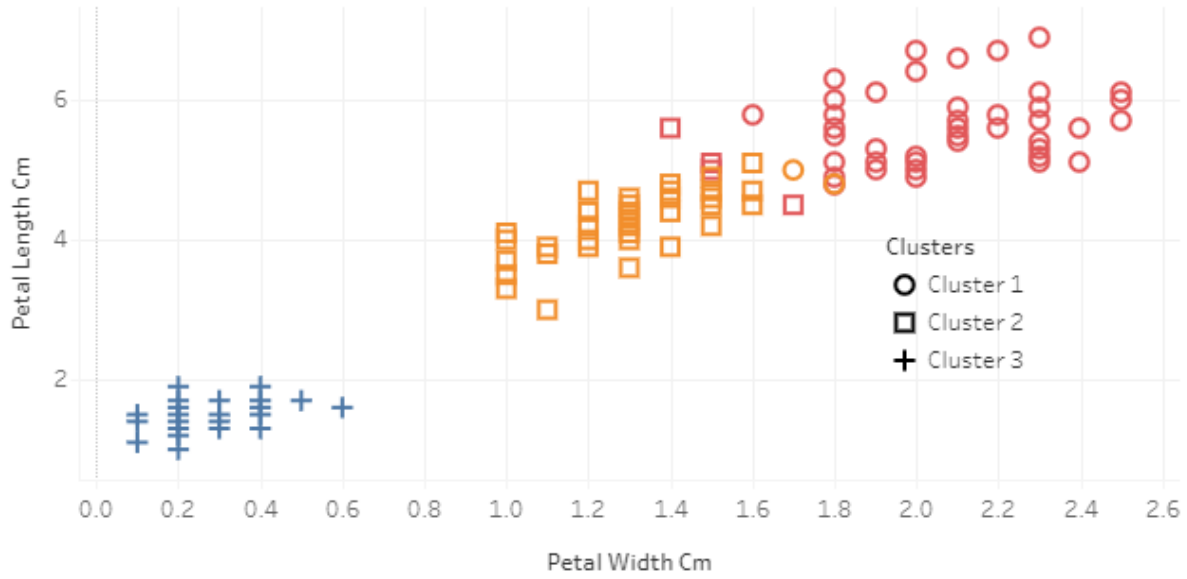


Kuva 9: Korrelaatiomatriisi (kuvalähde [10]). (Kuvan saavutettava tekstivastine: Iiris-aineiston kuvailevien attribuuttien sepal_length, sepal_width, petal_length ja petal_width lineaariset korrelaatiokertoimet kaikilla attribuuttikombinaatioilla. Korrelaatiokertoimien tarkat numeeriset arvot eivät ole tässä oleellisia, mutta esimerkiksi attribuuttien petal_length ja petal_width välinen korrelaatiokerroin on 0.97, mikä osoittaa merkittävän lineaarisen korrelaation.)

Tässä on syytä muistaa, että esim. lineaarisen korrelaation mittari (esim. Pearson) ei ole herkkä epälineaariselle korrelaatiolle: Lineaarisen korrelaation puute ei siten ole riittävä merkki korrelaation puutteesta sinänsä.

Korrelaatio ei myöskään automaattisesti esitä syy-seuraussuhdetta, vaan tällaisten tulkintojen tekeminen edellyttää sovelluksen kausaalisen mallin määrittelyä ja analysointia. Esimerkiksi asianmukaisesti toimivan auton lisääntynyt polttoaineen kulutus on usein pääosin seurausta kuljettajan ajotyylistä --- viime kädessä auton hallintalaitteiden käytöstä ja reittivalinnoista --- muttei päinvastoin. Mikäli kuljettajan käyttäytymisessä ja reittivalinnoissa ei ole oletettavaa isoa muutosta, voi juurisyyksi epäillä esim. merkittävää auton kuormauksessa tapahtunutta muutosta tai jonkinlaista vikatilannetta. Vika voi tietenkin piillä myös polttoaineen kulutuksen mittauksessa tai itse analyysissä (esim. systemaattiset muunnosvirheet eri lähteistä kerätyn data yhdistämisessä).

Scatterplot-tyyppisten kuvaajien käytössä seuraava askel on usein datan klustereiden ja poikkeamien, so. outlier-tyyppisten attribuuttiarvojen tutkiminen (koska näidenkin visualisointien tekeminen on analytiikkatyökaluissa melko helppoa). Seuraavassa kuviossa havainnollistetaan kolmen klusterin "löytymistä" iiris-aineistosta K-Means –klusterointialgoritmin avulla siten, että havaintoyksikoiden värit on koodattu luokka-attribuutin avulla:



Kuva 10: Scatterplot-visualisointi, jonka datapisteet on väritetty luokkien avulla. Kunkin datapisteen automaattisesti määritelty klusteri on merkitty erityyppisellä kuvakkeella. (Kuvan saavutettava tekstivastine: Attribuuttien Petal Width Cm ja Petal Length Cm suhteen määritelty scatterplot-kuvio, jossa datapisteet on värikoodattu tunnettujen luokkien Iris-setosa, Iris-versicolor ja Iris-virginica mukaisesti. Samaan kuvioon on merkitty klusterointialgoritmin laskennallisesti tuottamat klusterit, siten että datapisteiden luokkia ja niille laskettuja klustereita voidaan visuaalisesti vertailla.)

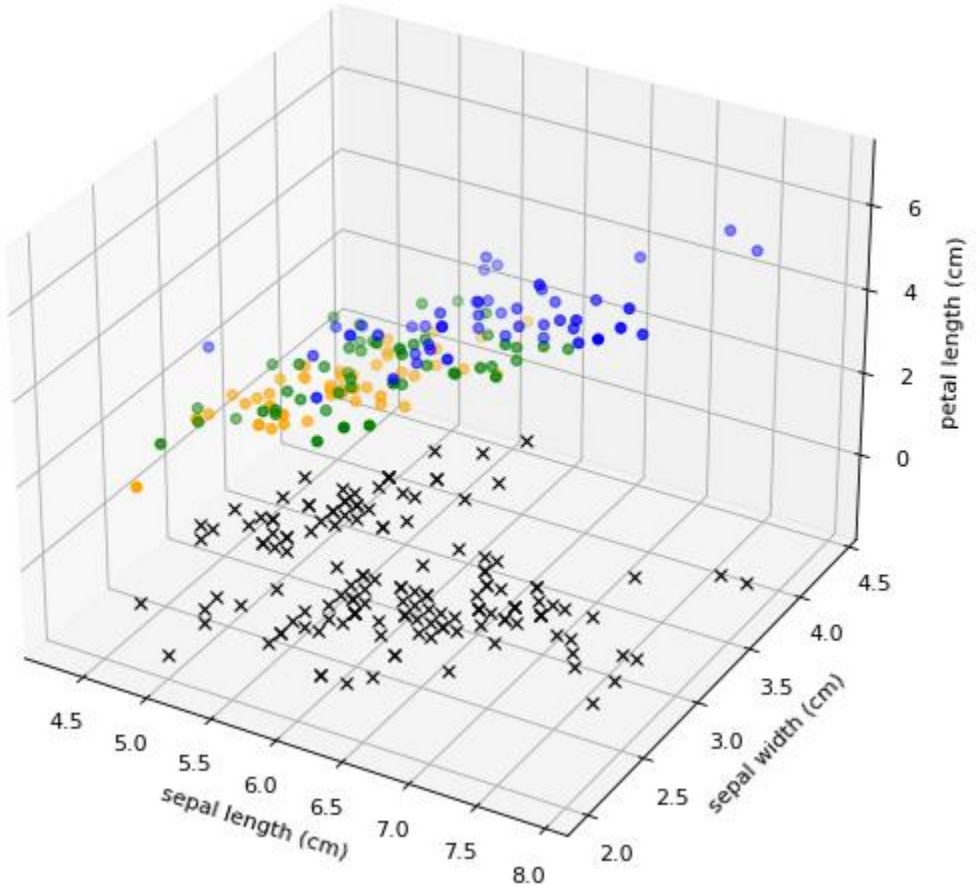
Koska iiris-aineisto on niin yksinkertainen, huomataan, että klusterit vastaavat lähes suoraan luokka-attribuuttien arvoja (mutta klusterointialgoritmi ei perusmuodossaan tiedä saati osaa selittää tätä). Toisin sanoen, klusterointia voitaisiin hyvin suurella tarkkuudella käyttää myös luokittelun tekemiseen, sopimalla että Cluster 1 = Iris virginica, Cluster 2 = Iris versicolor ja Cluster 3 = Iris setosa.

Huomaa että klusteroinnin visualisointi on nyt esitetty kahden attribuutin suhteen scatterplot-tasokuviona, mutta itse klusterointialgoritmi voidaan toki suorittaa neljäulotteisessa avaruudessa kaikkien neljän attribuutin suhteen. Tämä on tyypillistä analytiikan ja koneoppimisen sovelluksissa: Algoritmit sinänsä usein yleistyvät moniulotteisiin avaruuksiin, mutta ihmisten on vaikea hahmottaa muita kuin kaksi- tai (tasoprojisoituja) kolmiulotteisia visualisointeja. Tästä syystä dataa ja malleja usein projisoidaan matalampiulotteisiin visuaalisiin kuvioihin. "Lisäulottuvuuksia" voidaan toki pyrkiä visualisoimaan eri tavoin, esim. kuvaamalla tiettyjä attribuuttiarvoja väreiksi tai kuvioiksi, tai havainnollistamalla aika-attribuuttia animaatioiden tai viipalekuvioiden avulla.

Toisaalta jo verraten yksinkertaisten kolmiulotteisten visualisointien hahmottaminen on käytännössä usein huomattavasti hankalampaa kuin joukon "vastaavia kaksiulotteisia" visualisointeja. Hahmottamisen helpottamiseksi, visualisointeihin liitetäänkin usein koristeellisia apukuvioita (esim. tasojen kohtisuorat tai pistepilven muotoa kuvaava varjostettu kappale) ja vuorovaikutteisuutta (esim. kameran kuvakulman vaihtaminen). Koristekuvioiden käyttöön sisältyy tietenkin myös riski näennäisten säännönmukaisuuksien ilmestymisestä visualisointeihin ja siten väärien tulkintojen tekemisestä.

Oheisessa kuvassa iiris-aineiston kolme ensimmäistä attribuuttia on kuvattu kolmiulotteiseen koordinaatistoon ja väritetty viidennen, luokittelevan attribuutin avulla (neljäs attribuutti ei näy kuvassa

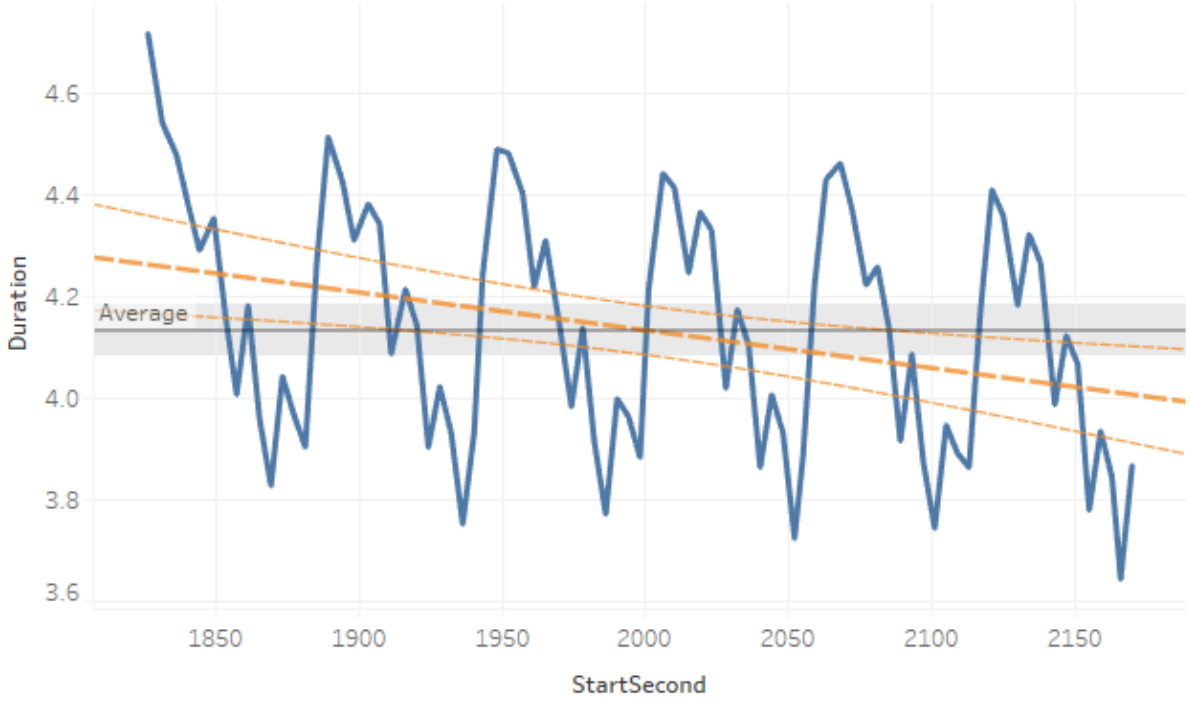
lainkaan). Pistepilven muodon havainnollistamiseksi, kuvaan on vielä lisätty koristeellisia apukuvioita: kukin (värillinen) datapiste (o) heittää mustan varjon (x) alla olevaan koordinaattitasoon:



Kuva 11: Iiris-aineiston kolmen ensimmäisen attribuutin avulla määritellyn datapilven visualisointi (tasoprojisoituna) 3D-kuviona. Datapisteet (o) on väritetty luokka-attribuutin mukaan. Huomaa visuaalisena vihjeenä lisätyt koristeelliset varjopisteet (x). (Kuvan saavutettava tekstivastine: Paralleeliprojisioitu kuvaus, jossa Iiris-aineiston luokittain värikoodatut datapisteet on kuvattu kolmiulotteisessa avaruudessa, käyttäen pisteiden koordinaatteina kuvailevia attribuutteja sepal length (cm), sepal width (cm) ja petal_length (cm). Kolmiulotteiden vaikutelman korostamiseksi, kukin datapiste heittää kuvitteellisen varjon kahden ensimmäisen attribuutin määrittelemään tasoon. Kuviosta saa summittaisen käsityksen aineiston muodosta.)

3D-kuvat auttavat yleensä numeerisen aineiston karkean muodon havainnollistamisessa, mutta ilman visualisoinnin vuorivaikutteisuutta (esim. valitun luokan/datapisteen korostus, filtteröinti ja kameran liikkeet), yksityiskohdista on hankalaa saada selvää. Käytännössä visualisoinneissa kannattaa siten suosia yksinkertaisten, sopivasti rajattujen ja pelkistettyjen tasokuvioiden käyttöä. (Vertaa esim. yllä olevaa (tasoprojisoitua) 3D-kuviota edellisten esimerkkien 4x4-taulukon 2D-scatterplot-kuvioihin, jotka itse asiassa sisältävät luokittain värikoodattuna kaikki iiris-aineiston tasoprojisiot, myös yllä kuvatun "varjon".)

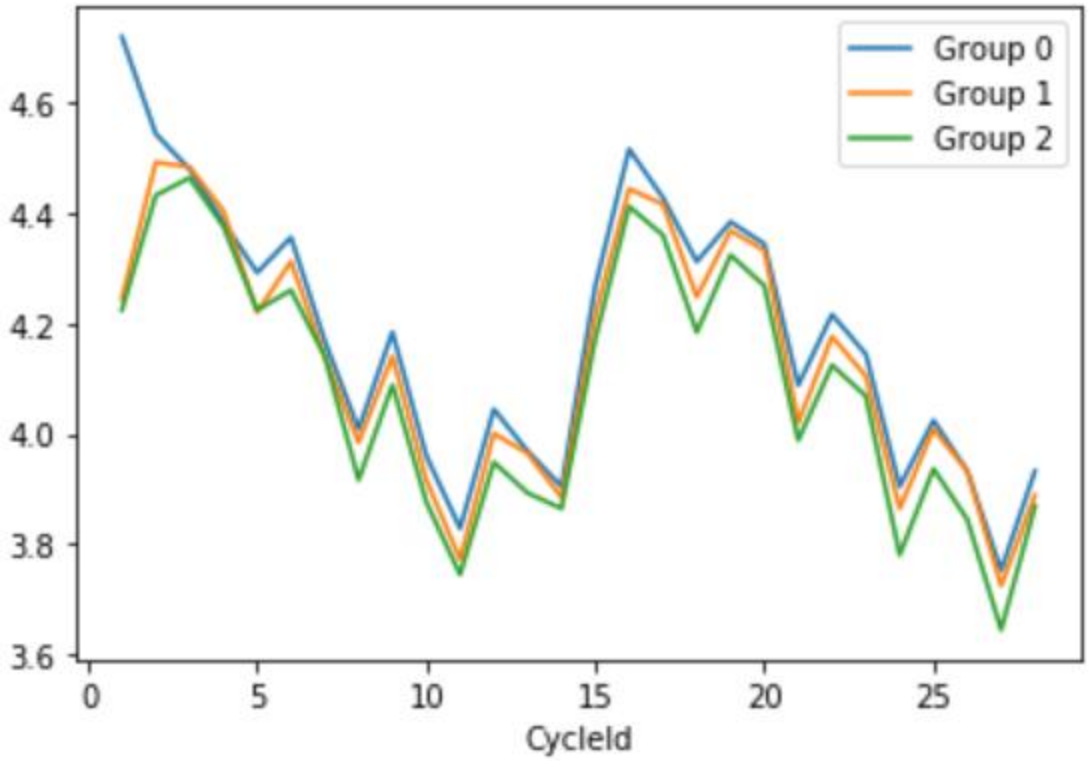
Aikasarjatyyppisen aineiston tapauksessa luonnollisia visualisointeja ovat valitun attribuutin arvon kuvaaminen ajan suhteen, syklisyyden havainnollistaminen ja trendikehitystä esittävien (usein lineaaristen) regressiosuorien esittäminen:



Kuva 12: Prosessin eri vaiheiden ajalliseen kestoon liittyvän aikasarjan visualisointi. Kuvioon on merkitty myös keskiarvo ja automaattisesti sovitettu trendisuora arvioituine virhemarginaaleineen. (Kuvan saavutettava tekstivastine: Attribuutin Duration arvoa kuvaava viivadiagrammi aikamuuttujan StartSecond suhteen. Kuvio muodostaa sahalaitaisen hammaskuvion, jossa kuusi kappaletta merkittäviä lokaaleita maksimi- ja minimikohtia. Trendisuorasta käy ilmi, että aikasarja on laskeva, minkä havainnon tekeminen voisi olla hankalaa ilman kuvioon lisättyä trendisuoraa.)

Huomaa, että myös yllä oleva kuvio perustuu oleellisesti datapisteiden esittämiseen. Pisteiden välille vedetyt viivat auttavat tässä hahmottamaan prosessin etenemistä --- mutta saattavat muodostaa myös osin harhaanjohtavan kuvan siitä, mitä tapahtuu datapisteiden välissä. Erityisesti, mikäli datan keruun näytteenottotaajuus on oleellisesti pienempi kuin prosessin eteneminen, saattaa tuloksena olla erittäin harhaanjohtavia visualisointeja. (Tätä voi testata kokeilemalla, miten visualisointi muuttuu, jos aineistosta jätetään satunnaisesti osa datapisteistä pois. Sovelluksissa näytteenottotaajuuden tulisi olla vähintään kaksinkertainen ilmiön etenemisen tai taajuuden suhteen.)

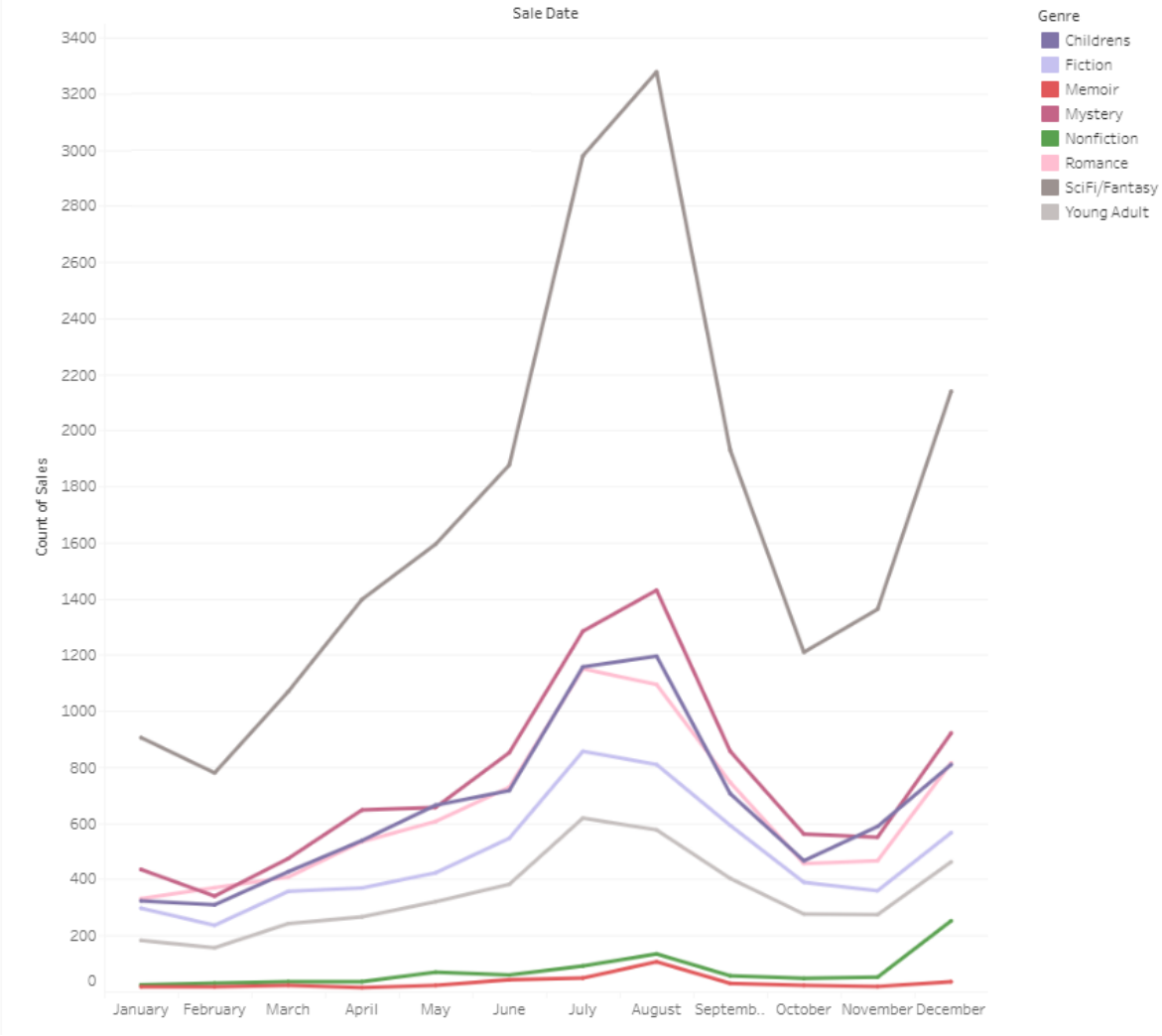
Visualisointien avulla voidaan paitsi esittää attribuuttien absoluuttisia arvoja, myös korostaa poikkeamia tai eri lähteistä kerättyjen aineistojen eroavaisuuksia. Esimerkiksi seuraavassa kuviossa vertaillaan edellisen kuvion kolmen peräkkäin toistuvan aikasarjan sykliryhmien eroavaisuuksia --- ja huomataan että viimeisen sykliryhmän 2 mittaukset sijaitsevat pääsääntöisesti ensimmäisen sykliryhmän 0 mittausten alapuolella (tämän havaitseminen yllä olevasta kuviosta on paljon hankalampaa). Toisin sanoen, paitsi että prosessi nopeutuu kokonaisuutena (vrt. trendisuora edellä), myös kaikki prosessin yksittäiset vaiheet vaikuttavat nopeutuvan:



Kuva 13: Syklisten aikasarja-aineistojen vertailua, aineistona edellisen esimerkin mittaukset prosessin eri vaiheiden ajallisesta kestosta. (Kuvan saavutettava tekstivastine: Edellisen kuvaajaesimerkin aineiston esitys toisessa muodossa. Tällä kertaa aineiston kolmen syklin (CycleId) StartSecond-Duration –asteikon datapisteet on piirretty yhteen kaavioon, viivakuviona toistensa päälle. Viivakuvio on värikoodattu tunnettujen sykliryhmien (CycleId) 0, 1 ja 2 mukaisesti. Kuviosta voidaan havaita, että ryhmän 2 kuvio sijaitsee ryhmän 1 kuvion alapuolella, joka puolestaan sijaitsee ryhmän 0 kuvion alapuolella.)

Pitkäkestoista aikasarjadataa ja sen kausivaihteluita esitetään tyypillisesti myös suhteutettuna esim. kalenterikuukausiin tai vuosineljänneksiin. Alla oleva kuva esittää eri kaunokirjallisuuden lajien menekkiä

vuoden ympäri tietyn aineiston pohjalta:

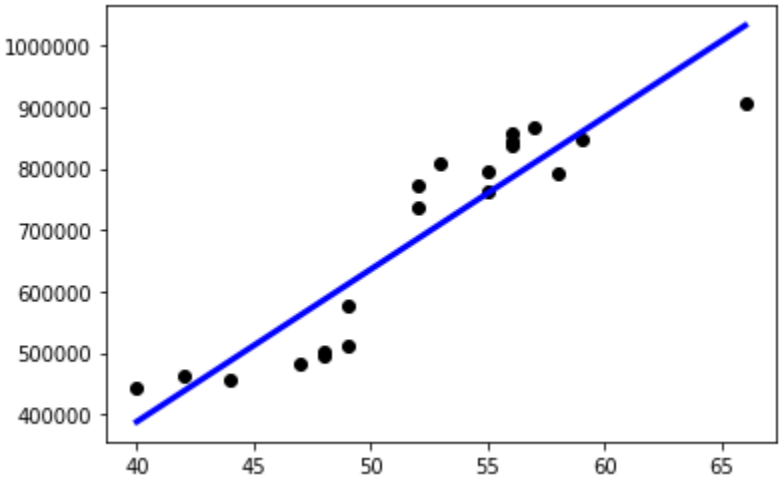


Kuva 14: Kausivaihteluiden visualisointia. (Kuvan saavutettava tekstivastine: Viivadiagrammeja kirjojen myyntiaineistosta akselilla Sale Date – Count of Sales. Kuvio muodostuu kahdeksasta päällekkäisestä viivadiagrammista, jotka kuvaavat kirjakategorioiden Childrens, Fiction, Memoir, Mystery, Nonfiction, Romance, SciFi/Fantasy ja Young Adult kuukausittaista myyntiä valittuna tarkasteluvuotena. Käytännössä kaikkia kirjoja myydään eniten kesäkuukausina ja joulukuussa. Kuviosta käy myös ilmi, että SciFi/Fantasy-tyyppisten kirjojen vuosittainen myynti on lähes kaksi kertaa suurempaa kuin seuraavaksi suurimman kategorian, Mystery-tyyppisten kirjojen myynti. Memoir-kirjat myyvät kaikkein vähiten, mutta niissäkin on selvä myyntipiikki kuukausien July-September –ajanjaksona.)

Kuvasta nähdään, että kaikentyyppiset kirjat myyvät parhaiten kesäkuukausina --- ja että scifi ja fantasia myyvät kaikkein eniten ympäri vuoden.

3.3. Huomioita (otoksen) edustavuudesta kehitystehtävän ilmiön kuvailussa

Kuten alussa todettiin, aineistoa käsiteltäessä on yleensä syytä kiinnittää huomiota attribuuttien arvojen vaihteluväleihin. Esimerkiksi ikä-palkka –aineistossa palkka-attribuutti on tyypillisesti paljon suurempi kuin ikä, mikä saattaa (tahattomasti) korostaa palkka-attribuuttia tietyissä laskutoimituksissa:

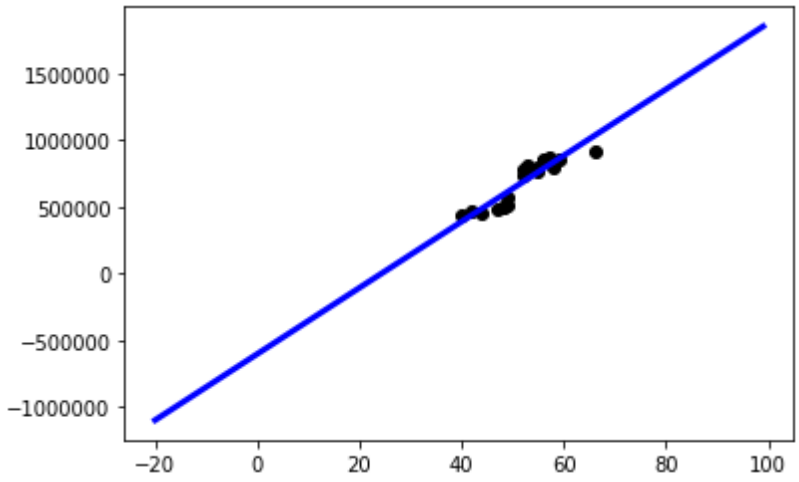


Kuva 15: Ikä-palkka –aineiston visualisointi aineistoon automaattisesti sovitetun regressiosuoran kera. Huomaa kuviossa automaattisesti tapahtuva skaalaus. (Kuvan saavutettava tekstivastine: Ikä-palkka-aineiston scatterplot-kuvio ja aineiston pohjalta sovitettu lineaarinen regressiosuora. Kuviosta käy ilmi, että vaikka regressiosuora mallintaa aineistoa hyvin, kulkee se tarkkaan ottaen vain yhden datapisteen kautta. Kuviosta havaitaan myös yksi potentiaalinen ulkopuolinen, yli 65-vuotiasta datapistettä kuvaava havainto. Aineiston ikä-attribuutit vaihtelevat noin välillä 40-66 ja palkat noin välillä 45000-90000, eli palkkojen vaihteluväli on huomattavasti suurempi. Kuvion automaattisen skaalauksen ansiosta nousevan regressiosuoran kulma näyttää kuitenkin silmämääräisesti olevan noin 45 astetta.)

Jos esimerkiksi lasketaan – tai katsotaan kuvasta -- kahden havaintoyksikön suorakulmaista (Euklidista) etäisyyttä, korostuu palkka-attribuutin osuus laskuissa. Kuvittele miltä yo. kuva näyttäisi, jos kuvion X- ja Y-asteikkojen skaalat olisivat 1:1 --- arvot sijoittuisivat lähes pystysuoralle viivalle! Monet visualisointikomponentit itse asiassa skaalaavat kuvioita automaattisesti, mistä hieman harhaanjohtavasti saattaa muodostua sellainen virheellinen käsitys, että itse datakin olisi skaalattua.

Eksplisiittinen attribuuttien arvojen skaalaus tai normalisointi on siten joskus tarpeen. Normalisointia tarvitaan esim. otosaineiston normaalijakaumaolettamukseen perustuvissa tilastollisissa analyyseissa, tai jos analyysimenetelmä on herkkä muuttujien arvon absoluuttiselle vaihtelulle (myöhemmin esim. neuroverkot). Normalisointi tehdään usein ns. Z-arvon tai normitetun arvon avulla, jossa normitettu arvo saadaan jakamalla mitatun arvon ja aineiston keskiarvon erotus aineiston keskihajonnalla.

Toisaalta mikään analyysi ei välttämättä selitä ilmiön käyttäytymistä attribuuttien havaitun vaihteluvälin ulkopuolella. Esimerkiksi ikä-palkka -tyyppisessä aineistossa yksinkertaista regressiosuoraa voidaan hyvin käyttää suunnilleen ennustamaan henkilön palkka iän suhteen. Kuitenkaan sille, miksi malli toimisi "kaukana" lähdeaineiston vaihteluvälin ulkopuolella, ei välttämättä ole hyviä perusteluja:



Kuva 16: Aineisto ja sen perusteella sovitettu regressiosuora. Huomaa että regressiosuoran tulkinta ei ole mielekäs "kaukana" aineistosta. (Kuvan saavutettava tekstivastine: Ikä-palkka-aineisto ja aineistoon sovitettu regressiosuora kuvattuna siten, että tällä kertaa kuviossa on esitettynä ikä-attribuuttia vastaava akseli arvojen –20 ja 100 välillä, Kuviosta havaitaan, että varsinaiset datapisteet sijaitsevat tietenkin edelleen välillä 40-66, mutta abstrakti trendisuora kulkee läpi koko tason, käsittäen siis mielivaltaisen pieniä ja suuria ikä-muuttujan arvoja.)

Erityisesti, koska regressiomalli on yksinkertaisesti vain aineistoon sovitetun suoran yhtälö, johon voi sijoittaa mitä tahansa lukuja, voi sen avulla "ennustaa" palkan vaikkapa negatiivisen iän perusteella! Tämä ei tietenkään ole järkevää ja erityyppisten mallien tuloksiin kannattaa aina suhtautua terveen kriittisesti ja maalaisjärjellä.

Lopuksi kannattaa vielä muistaa, että kehitystehtävän alustava analyysi perustuu yleensä tietyn *otosaineiston* analysointiin. Koska sekä otoksen valintaan että sen edustavuuteen voidaan ajatella liittyvän satunnaisuutta, tarkoittaa tämä sitä, että myös analyysien taustalla voi vaikuttaa osin sattuma. Mitä pienempi aineisto on (ja mitä suurempi on aineiston hajonta), sitä suurempi on todennäköisyys sille, että analyysin tulokseen voidaankin päätyä sattumalta!

Tilastotieteessä täsmällisesti määritellyille testeille voidaan laskea ns. p-arvo, joka kuvaa tuloksen tilastollisen merkitsevyyttä: Intuitiivisesti sanottuna p-arvo kuvaa, kuinka todennäköistä on, että testin tulokseen on päädytty sattumalta --- mitä pienempi testin p-arvo, sitä uskottavampi tai merkitsevämpi testitulos on. Alustavissa analyyseissä tällaiseen täsmällisyyteen ei yleensä päästä, joten on tärkeää, että alustavien tulosten perusteella ei tehdä liian kauaskantoisia johtopäätöksiä. Kyse on siis ennen kaikkea kehitystehtävän ilmiön suuntaa-antavasta opiskelusta.

3.4. Miten tästä eteenpäin?

Alustavan datan keräämisen ja alustavien visualisointien tarkoituksena on siis lisätä ymmärrystä kehitystehtävän ilmiöstä ja ennen kaikkea siitä saatavilla olevasta datasta. Tietyssä mielessä tavoite on oppia näkemään kehitystehtävän ilmiö datalähtöisesti, tietokoneen "näkökulmasta" --- joka ei ymmärrä ilmiöstä muuta, kuin mitä datassa on kuvattu.

Tarkka rajanveto milloin alustava analyysi loppuu ja varsinainen analyysi ja edelleen varsinainen sovelluskehitys alkavat, on tietenkin häilyvä. Nyrkkisäännön tasolla, alustava analyysi on valmis, kun dataa on saatu kerätyksi siinä määrin, että datasta on muodostunut kohtuullisen hyvä kokonaiskuva, ja yksinkertaisten tunnuslukujen ja helppojen visualisointien hyödyt on ulosmitattu.

Tässä kohtaa kehitystehtävän tekijälle tulisi myös olla muodostunut kohtuullisen hyvä käsitys siitä, millaista dataa on saatavilla ja mistä data kertoo. Samalla ymmärrys itse kehitystehtävän ilmiöstä tulisi olla selkiytynyt: "Aivan, tämähän on se kehitystehtäväni pihvi". Viimeistään nyt tulisi myös täsmentää ja päivittää kehitystehtävän määrittelyä.

Työn seuraavassa vaiheessa on vuorossa Proof of Concept (PoC) -projektin määrittely ja siihen liittyviä kokeiluita. Tämä edellyttää monimutkaisempiin menetelmiin ja uusiin työkaluihin perehtymistä ja eteneminen on tyypillisesti hieman hitaampaa. (Huomaa, että kaikkien kehitystehtävien tavoitteena ei välttämättä ole päästä PoC-vaiheeseen AI-lähettiläs-hankkeen puitteissa.)

Seuraavaksi suositeltavia AI-lähettiläs -dokumentteja [1]:

- AI-lähettiläshanke: Tableau (Public) –aloitusohje data-analytiikan ja visualisointien tekoon ilman ohjelmointia
- AI-lähettiläshanke: Weka –aloitusohje koneoppimisen kokeiluiden tekoon ilman ohjelmointia
- AI-lähettiläshanke: Scikit-learn –aloitusohje koneoppimisen kokeiluiden tekoon Python-ohjelmointikielellä

Em. aloitusohjeiden avulla pääsee hyvin kärryille ensimmäisten aihepiirin oikeiden työkalujen käyttöönotossa.

Lähteitä

[1] AI-lähettiläs -hankkeen verkkosivut. TAMK. Saatavilla https://projects.tuni.fi/ai-lahettilas/

[2] Mind map. Wikipedia. Saatavilla https://en.wikipedia.org/wiki/Mind_map

[3] Ishikawa diagram. Wikipedia. Saatavilla https://en.wikipedia.org/wiki/Ishikawa_diagram

[4] Keim, D., Kohlhammer, J. Ellis, G., & Mansmann, F. (Toim.) (2010). Mastering the Information Age: Solving Problems with Visual Analytics. Eurographics Association. Saatavilla https://www.vismaster.eu/wp-content/uploads/2010/11/VisMaster-book-lowres.pdf

[5] Otantamenetelmät. KvantiMOTV, Yhteiskuntatieteellinen tietoarkisto. Saatavilla https://www.fsd.tuni.fi/menetelmaopetus/otos/otantamenetelmat.html

[6] Iris Data Set. UCI Machine Learning Repository, University of California, School of Information and Computer Science. Saatavilla https://archive.ics.uci.edu/ml/datasets/Iris

[7] Muuttujien ominaisuudet. KvantiMOTV, Yhteiskuntatieteellinen tietoarkisto. Saatavilla https://www.fsd.tuni.fi/menetelmaopetus/mittaaminen/ominaisuudet.html

[8] Resources. Tableau Public. Tableau Software, LLC, a Salesforce Company. Saatavilla https://public.tableau.com/en-us/s/resources

[9] UC Irvine Machine Learning Repository. University of California, School of Information and Computer Science. Saatavilla https://archive.ics.uci.edu/ml/index.php

[10] Zhang, D. (2020). Exploring Classifiers with Python Scikit-learn — Iris Dataset. Towards Data Science. Medium. Saatavilla https://towardsdatascience.com/exploring-classifiers-with-python-scikit-learn-iris-dataset-2bcb490d2e1b

[11] Tableau Public. Tableau Software, LLC, a Salesforce Company. Saatavilla https://public.tableau.com/en-us/s/

[12] Data Visualization | Microsoft Power BI. Microsoft. Saatavilla https://powerbi.microsoft.com/en-us/

[13] Anaconda | Individual Edition. Anaconda Inc. Saatavilla https://www.anaconda.com/products/individual