Tekoäly- ja ohjelmistorobotiikkakokeilun loppuraportti - syksy 2020

Liikennedatan validointi ja korjaus

Helsinki

Tukea digitalisaatiokokeiluihin kaupungin työntekijöille

Liikennedatan validointi ja korjaus

Hannu Seppälä (Helsingin kaupunki)
Jussi Martikka ja Antti Heino (SAS)
Joonas Itkonen (Gofore)

Helsinki

1. Kokeilun onnistuminen

- Kokeilun alussa asetetuissa perustavoitteissa onnistuttiin.
- Kokeileminen vakuutti, että tekoälyn hyödyntäminen soveltuu puuttuvan liikennedatan täydentämiseen ja korjaamiseen.
- Tavoitteeksi ei asetettu tuotantovalmista ratkaisua ja tuotantovalmis ratkaisu vaatisi vielä jatkokehitystä.
- Kokeilussa käytetyt menetelmät osoittivat, että ne soveltuvat monipuoliseen puuttellisen datan täydentämiseen ja korjaamiseen, joita voi iteraatioiden myötä hyödyntää myös datan validoinnissa.
- Lopputulokset antavat hyvän viitteen siitä, että kokeluissa hyödynnettyjä menetelmiä kannattaisi hyödyntää vastaanvanlaisten puuttuvien datojen korjaamiseen, täydentämiseen ja prosessin automatisointiin.

Helsinki

2. Kokeilun eteneminen

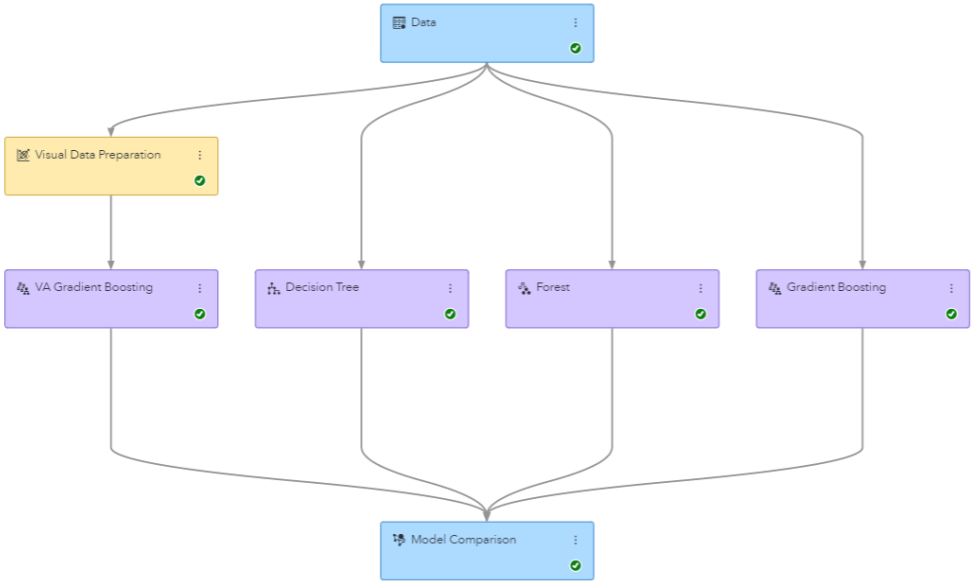
- Ratkaistavan ongelman määrittely ja kokeilun rajaaminen
- Liikennedatan ja rikastedatan kerääminen (sisäisistä ja avoimista tietolähteistä)
- Datojen siirtäminen analyysiympäristöön (SAS Viya HKI Azuressa)
- Datojen valmistelu, yhdistäminen ja laadun parantaminen
- Mallintamiselle olennaisten muuttujien luonti (mm. Edellisen päivän ja edellisen viikon arvot)
- Datan tutkiminen ja huomiot datasta
- Mallinnettavien muuttujien ja selittävien tekijöiden valitseminen
- Erilaisten ML mallien kokeileminen liikennedatan puuttuvien arvojen päättelemiseen (SAS ja Python)
- Välituloksien läpikäynti
- Mallien vertailu ja parhaiten suoriutuvien mallien valinta
- Mallien kouluttaminen ja hienosäätö hyödyntäen tekoälyä
- Tulosten visualisointi ja analyysien tuottaminen
- Tulosten tarkastelu ja jatkon ideointi

3. Kokeilun tuotokset

- Huomioitiin, että datassa voi olla puutteita, joka vaatii esikäsittelyä
  - Tuotettu koodia ja käsittelyaskelia ongelman ratkaisemiseksi
- Todettiin, että tekoälyä voidaan käyttää liikennedatan validointiin ja korjaukseen ratkaisemiseen
- Saatiin luotua menetelmiä datan aitoon puuttuvuuksien paikkaamiseen
  - Käsittelyputket SAS:ssa ja Pythonissa
- Konkreettiset mallit olemassa ja sovitettavissa SAS Viya -ympäristössä sekä Python-koodina

3. Kokeilun tuotokset

- Kokeilun aikana syntyi useampi koneopettamista hyödyntävä malli
  - SAS
  - Avoin lähdekoodi (Python)
- Malleja vertailtiin keskenään keskineliövirhe kriiterillä



Model Comparison

Champion	Name	Algorithm Name	Average Squared Error	Root Average Squared Error
	Gradient Boosting	Gradient Boosting	217,792.7995	466.6828
	Forest	Forest	310,942.9672	557.6226
	VA Gradient Boosting	Gradient Boosting	470,231.5060	685.7343
	Decision Tree	Decision Tree	536,704.5597	732.6012

Helsinki

3. Kokeilun tuotokset

- Pythonilla tehty tuotos on ohjelmakoodi, jolla voidaan luoda imputointi dataan

In [ ]:

```
# Functions to caluculate statistics and test and compare models

# A function to revert standarded imputed data to original scale
def getOriginalScaleImputed(X, model):
    XC = X.copy()
    imputer_step = [step for step in model.steps if step[0] == "imputer" or step[0] == "standardizer"]
    XC_destandardized = DataToLong().transform(imputer_step[0][1].destandardize(DataToWide().transform(XC)))
    return(XC_destandardized)

# A function to compare real data (XTest) and estimates (XPred) given by imputer fitted without values in XTest
# Requires long format, an option to limit to single variable
def imputationLoss(XPred, XTest, measure="all"):
    XTestV = XTest[["PVM","variable", "Measure"]]
    XTestV = XTestV.rename({'Measure': 'MeasureTest'}, axis=1)
    XPredV = XPred[["PVM","variable", "Measure"]]
    XPredV = XPredV.rename({'Measure': 'MeasurePred'}, axis=1)
    merged = XTestV.merge(XPredV, on=('PVM', 'variable'), how='left')
    merged = merged[~merged["MeasurePred"].isnull()]
    merged = merged[~merged["MeasureTest"].isnull()]
    mean_diff2 = None
    if measure != "all":
        merged = merged[merged['variable'] == measure]
    merged["Diff2"] = pow(merged["MeasureTest"] - merged["MeasurePred"],2)
    mean_diff2 = merged["Diff2"].mean()
    return(mean_diff2)


# A function to calculate cross validation scores for different models
def imputation_cross_validate(model, X, measure="all"):
    scores = []
    to_standard=False
    imputer_step = [step for step in model.steps if step[0] == "imputer"]
    if len(imputer_step) == 1:
        to_standard = imputer_step[0][1].to_standard
    sss = StratifiedShuffleSplit(n_splits=10, test_size=0.1, random_state=0)
    splitted = sss.split(X,X["variable"])
    for train_index, test_index in splitted:
        train = X.iloc[train_index,:]
        test = X.iloc[test_index,:]
        model.fit(train)
        imputedTest = model.transform(train)
        if to_standard:
            pipeSt = Pipeline([('data_to_wide', DataToWide()), ('standardizer', Standardizer()), ('data_to_long', DataToLong(
            testStd = pipeSt.transform(test)
        score = imputationLoss(imputedTest, testStd, measure)
        scores.append(score)
    return(scores)

# Calculate similar score as earlier but limit only to one variable and no standardization
def variable_mean_square_sum(model, X_train, X_test, measure):
    model.fit(X_train)
    imputedTest = model.transform(X_train)
    imputedTest = getOriginalScaleImputed(imputedTest, model)
    score = imputationLoss(imputedTest, X_test, measure)
    return(score)
```

Helsinki

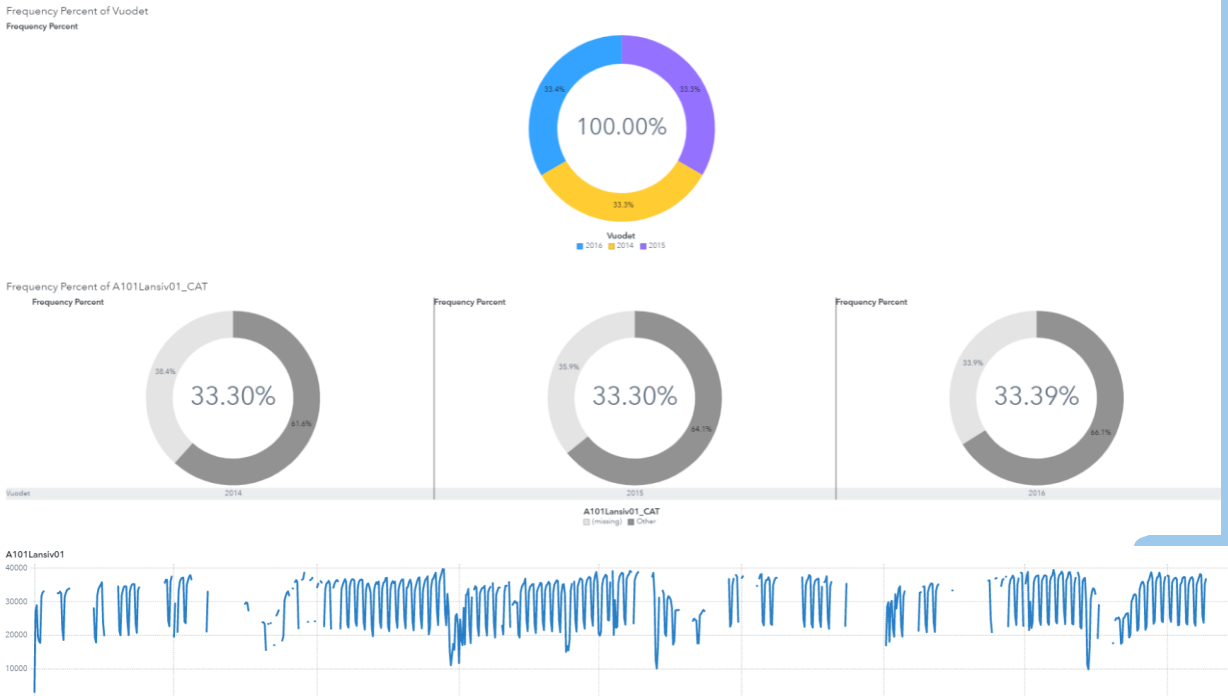
3. Kokeilun tuotokset

- Koneopetettu malli pystyy ennustamaan liikennearvot puuttuville ajanjaksoille
- Kokeilun aikana kokeiltiin eri metodeja ennusteen tuottamiseksi
  - Python puolella 6-7 eri mallia, joista valittiin paras
- Mallien tuottamia ennusteita tarkasteltiin yksittäin mallikohtaisesti
  - Python (Ridge Regressio)
  - SAS (Gradient Boosting)
- Tarkasteluun sisältyi myös SASin ja avoimen lähdekoodin mallien ennusteiden vertailu keskenään

Helsinki

3. Kokeilun tuotokset

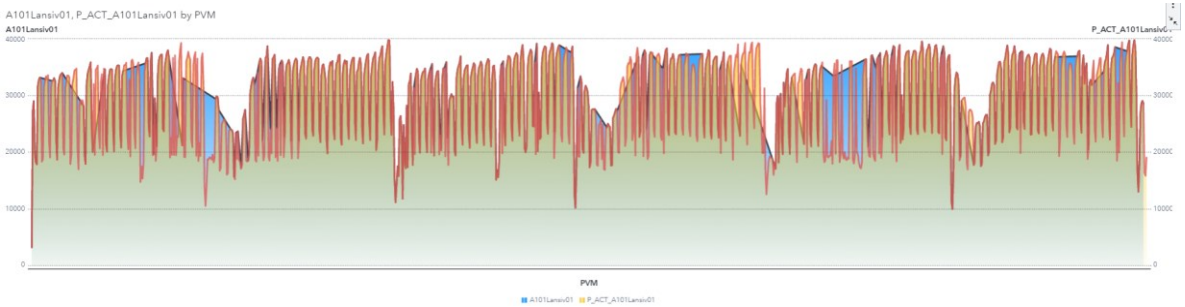
- Datan otos (3) kolmen vuoden ajanjaksolta (n. 1100riviä)
- Kokeilun datan puuttuvuutta simuloitiin vastaamaan aitoa dataa, jolloin puuttuvuus oli jakautunut tasaisesti otoksen eri ajanjaksoille
- Mallien koneopettamisen jälkeen simulointi datasta poistettiin n. 300 riviä sattumanvaraisesti poimituilta riveiltä, jolloin malleilla oli oikeat lähtökodat ennustaa oikeaa puuttuvuutta
    - Datassa oli tämän jälkeen sekä aitoa että simuloitua puuttuvuutta
    - Simuloidun puuttuvuuden perusteella mahdolisuus saada tarkkusasrvioita

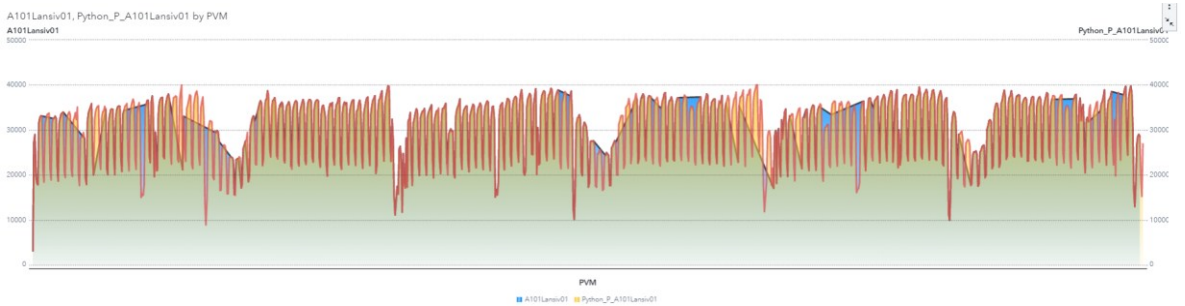


Helsinki

3. Kokeilun tuotokset

- Ennusteiden tuottamiseen hyödynnettiin eri metodeja ja teknologioita, näin ollen myös ennustetuissa arvoissa oli eroja
- Ennustettava data oli molempia malleja ajettaessa identtinen vertailukelpoisuuden vuoksi ja kohde, joita mallit selittivät oli valittu datan laadun mukaan
- Näissä vertailuissa kohde on "A101Lansiv01"





Helsinki

3. Kokeilun tuotokset

PVM	Päivät	juhlapaiva	Saalampoilma	A101Lansiv01	P_ACT_A101Lansiv01	Python_P_A101Lansiv01	Mallien erotus
01/01/2014	Keskiviikko	Uudenvuodenpäivä	3.6	3268	3268	3268	0.00
02/01/2014	Torstai		1.1	27313	27313	27313	0.00
03/01/2014	Perjantai		1.7	29027	29027	29027	0.00
04/01/2014	Lauantai		2.6	19744	19744	19744	0.00
05/01/2014	Sunnuntai		4.5	18291	18291	18291	0.00
06/01/2014	Maanantai	Loppiainen	2.4	17878	17878	17878	0.00
07/01/2014	Tiistai		3.4	31681	31681	31681	0.00
08/01/2014	Keskiviikko		5.1	32799	32799	32799	0.00
09/01/2014	Torstai		4.6	33178	33178	33178	0.00
10/01/2014	Perjantai		1.8	.	32661.293349	33096.182785	-434.89
11/01/2014	Lauantai		-3.2	.	18377.70606	21134.797781	-2,757.09
12/01/2014	Sunnuntai		-7.3	.	18842.212136	18470.197639	372.01
13/01/2014	Maanantai		-10.4	.	31470.46751	31383.971912	86.50
14/01/2014	Tiistai		-13.6	.	32547.814525	32344.039753	203.77
15/01/2014	Keskiviikko		-10.4	.	33422.931831	33461.968883	-39.04
16/01/2014	Torstai		-11.5	.	33096.083584	33648.235225	-552.15
17/01/2014	Perjantai		-12.7	.	33616.738511	33772.278272	-155.54
18/01/2014	Lauantai		-12.7	.	21859.295326	21352.979089	506.32
19/01/2014	Sunnuntai		-12.1	.	19089.594526	18950.283649	139.31
20/01/2014	Maanantai		-11	.	31497.930194	31528.367083	-30.44
21/01/2014	Tiistai		-10.4	32601	32601	32601	0.00
22/01/2014	Keskiviikko		-13.9	32619	32619	32619	0.00
23/01/2014	Torstai		-14.3	33083	33083	33083	0.00
24/01/2014	Perjantai		-15.3	32584	32584	32584	0.00
25/01/2014	Lauantai		-7.2	21995	21995	21995	0.00
26/01/2014	Sunnuntai		-7.2	18718	18718	18718	0.00
27/01/2014	Maanantai		-7	31470	31470	31470	0.00
28/01/2014	Tiistai		-6.9	32701	32701	32701	0.00
29/01/2014	Keskiviikko		-7.8	33569	33569	33569	0.00
30/01/2014	Torstai		-9.7	33804	33804	33804	0.00
31/01/2014	Perjantai		-8.3	34009	34009	34009	0.00
01/02/2014	Lauantai		-5.6	.	20750.569399	20942.011419	-191.44
02/02/2014	Sunnuntai		-2.6	.	19584.663995	19493.270267	91.39
03/02/2014	Maanantai		-0.4	.	31193.508433	31350.323186	-156.81

Helsinki

4. Opit tekoälyn mahdollisuuksista

- Tekoälyä on mahdollista käyttää virheellisen datan etsimisen ja korjaamisen ja sen korjaamisen automatisoinnin helpottamiseksi
- Käytännön oppi, millainen prosessi datan virheiden korjaus tekoälyn avulla on
- Tekoälyssä on potentiaalia kaupungin palveluiden parantamisessa

Helsinki

5. Opit tekoälyn kehittämisestä

- Datan esikäsittely vie suhteellisen paljon aikaa, vaikka data olikin jo varsin hyvin määritelty aluksi
- Mallinnus onnistui varsin sujuvasti ja kutakuinkin suunnitelmien mukaan
  - Ymmärrys ongelmasta parani vielä mallinnuksen aikana
- Mittari, jolla mallin toimivuutta mitataan, vaati harkintaa
  - Keskineliövirhe vai jokin muu kriteeri
  - Eri mittareiden merkityksen ja skaalan vaikutus
  - SAS Viyassa automatiikkaa tämän ratkaisemiseen

Helsinki

6. Opit kokeilemisesta

- Työvaiheiden suunnittelu ja läpivienti lopulta onnistui hyvin
- Kommunikaatio toimi hyvin
- Kokeilukiihdyttämö on helppo ja hyvä konsepti
- Kokeilun käytännön tekemisen vaihe ajoittui aikaikkunan loppuun, jolloin joidenkin asioiden kanssa tuli kiire
  - Datan käsittely tehtiin nopeasti
  - Nopeita ratkaisuja
- Työskentely-ympäristöjen tekniset haasteet
  - Python-kehitys SAS-ympäristössä ei heti onnistunut, vaan piti keksiä kiertoratkaisuja käytännön työskentelyyn

Helsinki

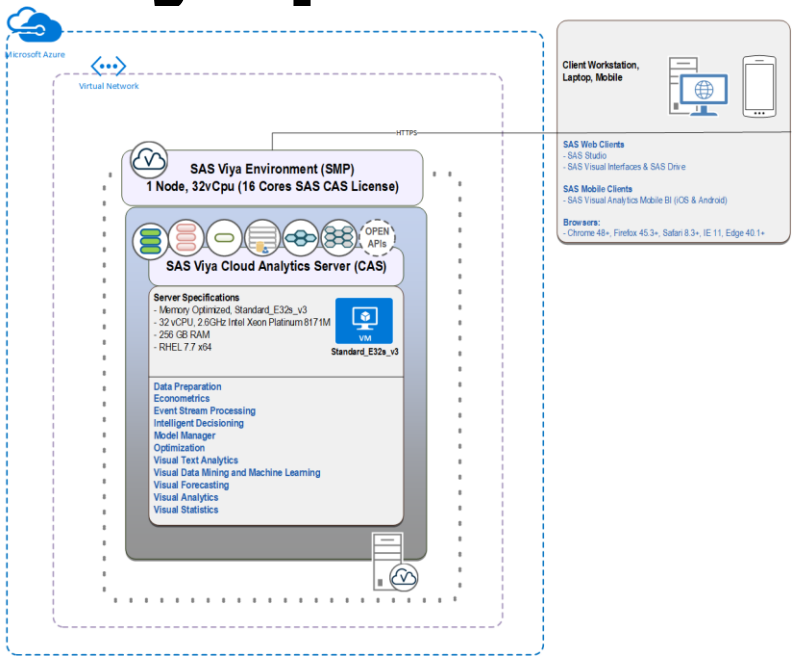
7. Opit resursoinnista

- Kesti kauan, että kokeilu päästiin aloittamaan
  - Sopimusasiat ja tekniset ympäristöt
  - Kaupungin kokeiluissa tekniset ympäristöt voivat olla haaste
- Kokeilu eteni varsin paljon odotetussa aikataulussa, kun kokeilu saatiin käyntiin
- Kokeilun ajankäyttö painottui voimakkaasti loppuvaiheeseen

Helsinki

8. Kokeilun tekninen ympäristö

- Kokeilussa käytettiin SAS Viya analytiikka- ja tekoäly alustaa
- Alusta on asennettuna Microsoft Azureen
- JupyterHub samassa ympäristössä, jossa mahdollista tehdä avoimen lähdekoodin Python-mallinnusta
  - Pythonin omat datankäsittelykirjastot (Num py, Pandas)
  - Visuaalisen tarkastelun kirjastot (MatPlotLi b)
  - Pythonin mallinnuskirjastot (Sklearn, Stats models)
- Samaa alustaa hyödynnettiin jo viime kokeilukierroksella ja se otettiin käyttöön nyt uudestaan



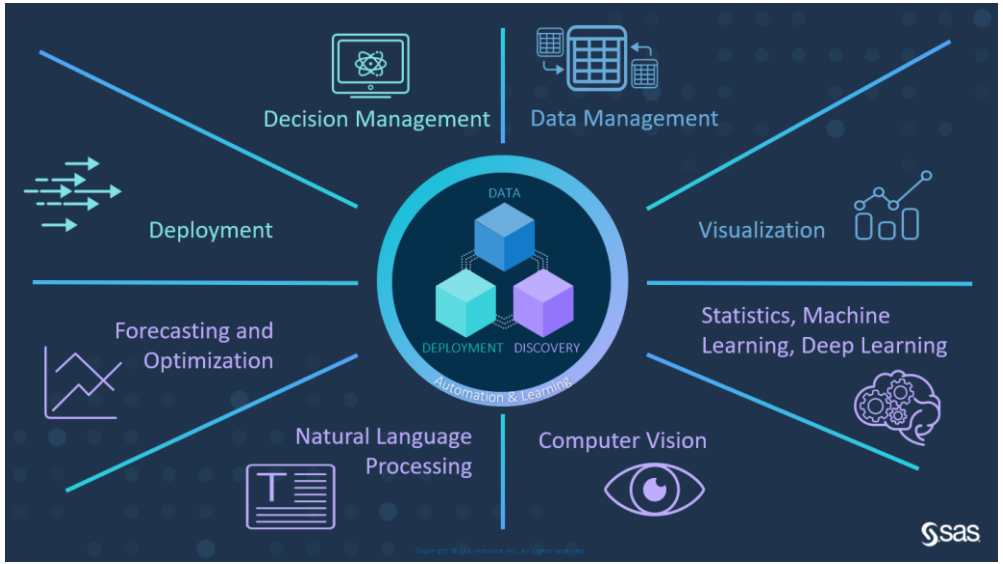
Helsinki

8. Kokeilun tekninen ympäristö

SAS Viyalla pystytään toteuttamaan valtava määrä erilaisia tekoälyn käyttötapauksia

- Samaa alustaa pystyy hyödyntämään moneen tekoälyn käyttötapaukseen
  - Koneoppiminen
  - Syväoppiminen
  - Konenäkö
  - Tekstianalytiikka
  - Aikasarjaennustaminen
  - Optimointi
  - Tilastollinen analyysi
  - Visualisointi

- Alustassa myös vahvat kyvykkyydet tuotannollistamiseen, jotta projektit eivät jää kokeiluksi ja niistä saadaan jatkuvia hyötyjä
- Vältytään vaikeasti hallittavalta joukolta pisteratkaisuja kun yhdellä alustalla voi ratkaista monta haastetta



8. Kokeilun tekninen ympäristö

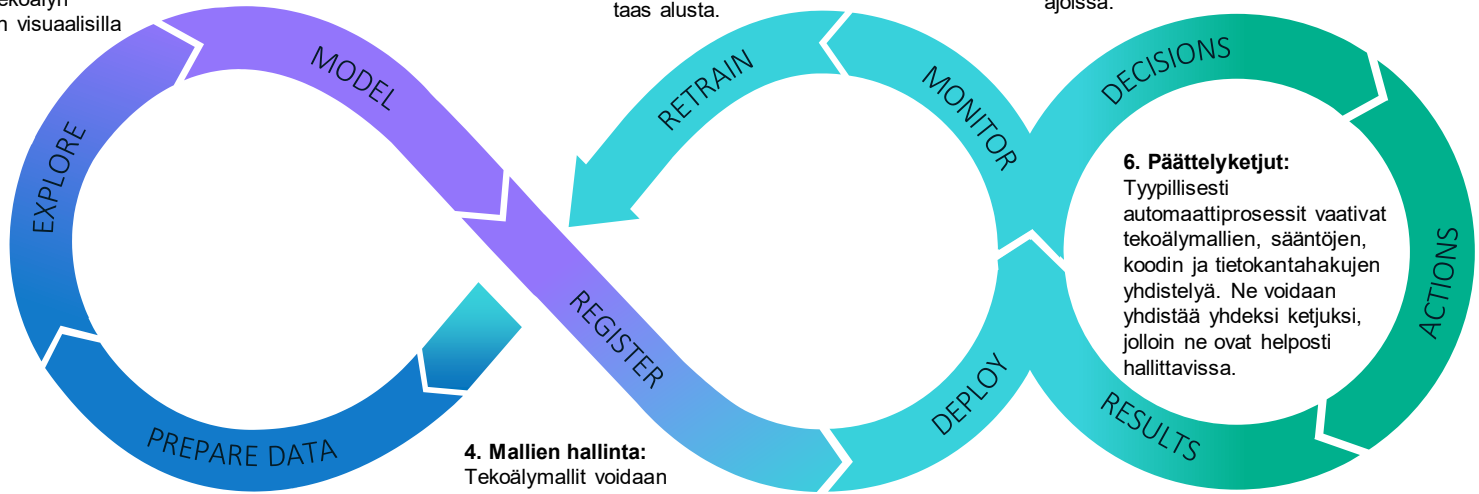
SAS Viya mahdollistaa tekoälyn elinkaaren kaikki vaiheet ja varsinkin tuotannollistamisen

2. Explorointi & visualisointi: Datan nopea tutkiminen ja tekoälyn hyödyntäminen visuaalisilla käyttöliittymillä

3. Tekoälymallien kehitys ja hienosäätö: Erilaisia algoritmeja on nopeaa verrata tosiaan vastaan ja saada paras mahdollinen malli datalle

8. Uudelleen koulutus: Data ja ymmärrys siitä harvoin pysyvät samanlaisena ikuisesti, joten jossain vaiheessa on syytä uudelleen kouluttaa mallit, jolloin elinkaari alkaa taas alusta.

7. Monitorointi: Tekoälymallien performanssia voidaan seurata, jotta mahdollinen uudelleen kouluttamisen tarve huomataan ajoissa.



6. Päättelyketjut: Tyypillisesti automaattiprosessit vaativat tekoälymallien, sääntöjen, koodin ja tietokantahakujen yhdistelyä. Ne voidaan yhdistää yhdeksi ketjuksi, jolloin ne ovat helposti hallittavissa.

1. Datan valmistelu: yhdistely, muutokset, uusien muuttujien luonti etc.

4. Mallien hallinta: Tekoälymallit voidaan rekisteröidä keskitettyyn mallien hallintaan, jotta tiedetään missä ne ovat, kuka ne on tehnyt ja mikä versio on käytössä.

5. Tuotannollistaminen: Tekoälymallit voidaan tuotannollistaa sellaisenaan (eräajo / reaaliaikainen) tai lisätä osaksi päättelyketjuja. Näin ne voivat päivittää esim. näkymiä tasaisin väliajoin tai palvella automaattiprosesseja.

Helsinki

9. Kokeilun data

- Autoliikenne-, pyöräilijä-, jalankulku sekä metromatkustajadata mitattuna päivittäisinä summina kiinteässä joukossa pisteitä
  - Automaattisesti mitattuja
  - Sisältävät merkittävän osan puuttuvuutta
- Taustadata, jota voidaan käyttää apuna edellisten mittausten korjaamisessa
  - Säädata
  - Mittauksen ajankohta (vuosi, kuukausi, viikko, viikonpäivä, juhlapävät)
- Kokeilu rajattiin vuosien 2014-2016 datan tutkimiseen
  - Datan liiallinen puuttuvuus tai liian pieni otos datasta on usein haitallinen tekijä mallien koneopettamisessa
- Data oli saatavilla Excel-muodossa ja SAS-tauluissa

10. Jatkopäätökset ja -ideat

- Manuaalisesti korjatun datan vertailu
  - Mahdollisuuksien mukaan malleja voidaan koneopettaa manuaalisesti korjatulla datalla
- SAS ja avoimen lähdekoodin mallien vertaileminen SAS Viyassa ja mallien käyttöönotto
  - Vaatii teknisen ympäristön konfiguraatiota
- Potentiaalia jatkokehitykseen
  - Jatko tarkentuu

Helsinki