//VALDKONNA
ÜLEVAADE

KRATI TEHNOLOOGIA VALDKONNA ÜLEVAADE PROJEKTIJUHTIDELE

Sisukord

//Terminoloogia ... 3
//Andmetüübid ... 4
//Mis info on andmetes? ... 4
//Suurandmed ... 5
//MapReduce vs Spark kokkuvõtvalt ... 8
//Veel tehnoloogilisi komponente ... 8
//Masinõpe ... 9
//Masinõpe ja traditsiooniline moodne tarkvaraarendus ... 9
//Enim kasutatavad programmeerimiskeeled masinõppes ... 9
//Peamised masinõppe tüübid ... 10
//Juhendatud õpe ... 10
//Juhendamata õpe ... 11
//Stiimulõpe ... 11
//Projektijuhtimine masinõppes ... 12
//Projekti läbiviimise metoodika ... 12
//Masinõppe projekti rollid ... 13
//Masinõppe kasulikkuse hindamine ... 14
//Eksimise Maatriks ... 15

//Terminoloogia

Olulisemaid masinõppe ja andmeteaduse valdkonna termineid:

Algoritm – kogum reegleid ja taasesitamist võimaldavaid samme lahendamaks probleemi või tegemaks kalkulatsiooni (tavaliselt arvuti poolt)

Andmetunnus (*Feature*) – sisendatribuut, mida kasutatakse ennustamiseks

Hajusarvutus - suurandmete töötlemiseks kohandatud arvutusarhitektuur

Masinnägemine – masinatele õpetamine, kuidas visuaalset maailma tõlgendada (nö masinate ’nägema’ õpetamine)

Masinõpe - arvutiteaduse haru, mis annab arvutitele õppimisvõime ilma neid otseselt selleks programmeerimata

Neurovõrk – algoritm, mis üritab imiteerida inimaju toimetusi kasutades selleks omavahel ühendatud neuroneid, mis asuvad kihiti ja saadavad üksteisele infot

Suurandmed (*Big Data*) - andmed, mis on väga suure mahuga või piisavalt keerulised, et nende töötlemiseks on vaja eraldi tehnilist lahendust.

Süvaõpe (*Deep Learning)* – masinõppe valdkond, mis kasutab kunstlikke neurovõrke, mis asetsevad mitmetes kihtides.

Tehisintellekt (Kratt) - populaarteaduslik termin

- Kitsas tehisintellekt – algoritm, mis on mõeldud lahendama ühte konkreetset ülesannet
- Üldine tehisintellekt – algoritm, mis võimaldab lahendada ükskõik millist probleemi. Üldine tehisintellekt on antud dokumendi kirjutamise hetkel veel suund, kuhu teadus liigub ning praktilisi lahendusi on siiski vähe

Treenimise andmed - andmed masinõppe mudeli treenimiseks

Valideerimise andmed - andmed masinõppe mudeli valideerimiseks

//Andmetüübid

Kõik andmeteadus- ja masinõppe projektid vajavad sisendiks andmeid. Selleks on vajalik teada peamisi andmete tüüpe:

- Kvantitatiivsed andmed ehk numbrilised andmed:
    - Diskreetsed andmed: 0, 1, 2, 3, 4
    - Pidevad andmed: väärtused, mida ei saa lugeda, vaid ainult kirjeldada teatud määratud intervallide kaudu
- Kvalitatiivsed andmed ehk kategoorilised andmed: andmed, mis on defineeritud kategooriatena

Kiire viis, kuidas otsustada, kas andmed on numbrilised või kategoorilised on küsida endalt küsimus: “Kas nendest andmetest on võimalik välja arvutada keskmine väärtus?”. Keskmine väärtus postiindeksist või sünniaastast ei oma tähendust – seega need andmed on kategoorilised, isegi kui neid kuvatakse numbritega.

//Mis info on andmetes?

Masinõppe ülesande lahendamiseks on vaja andmeid, kuid ainult suvalistest andmetest ei piisa; neis peab olema infot või signaali.

Näide: selleks, et tuvastada krediitkaardi pettust on vajalik teada tehingu summat, asukohta, kaardiomaniku asukohta jne (kaardiomaniku nimi ei ole antud hetkel oluline).

Juhendamata masinõppe puhul on võimalik kasutada märgendamata andmeid, juhendatud masinõppe puhul aga märgendatud andmeid. Kui märgendatud andmeid ei ole, siis tuleb neid tekitada.

● Kui ennustuse sihtmärk on kategooriline, on vaja märgendatud andmeid iga kategooria kohta;

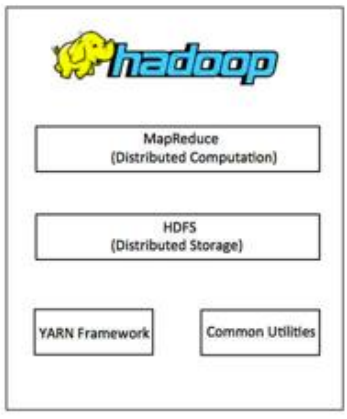
● Kui ennustuse sihtmärk on numbriline, on vaja võimalikult palju andmeid ennustuse vahemiku piires.

Selleks, et aru saada, kas tunnus annab üldse ennustatavat signaali funktsiooni, on hea aru saada, kuidas tunnus käitub väljaspool funktsiooni.

//Suurandmed

Suurandmed on üheks peamiseks tehnoloogiaks, mis võimaldab tehisintellekti kasutuselevõttu. Miks meil on vaja suuri andmeid käsitleda tavalistest andmetest erinevalt? Põhjus on peamiselt seotud arvutite tehnoloogilise võimekusega. Kõvaketaste lugemise kiirus on arenenud aeglasemalt kui arvutite protsessorite kiirus. Näiteks kui 1990. aastal võttis 1370 MB kõvaketta lugemine (4.4 MB/s) aega umbes 5 minutit, siis tänapäeval, peaaegu 30 aastat hiljem, võtab 1 TB kõvaketta lugemine (~100 MB/s) aega 2.5 tundi. Suurandmete peamine tehnoloogiline idee seisneb suurte andmete töötlemise ülesande jaotamises mitme erineva protsessori vahel. Üheks peamiseks tarkvaraliseks lahenduseks on Hadoop.

Apache Hadoop on avatud lähtekoodiga tarkvaralahenduste teek, mille eesmärgiks on suuremahuliste andmehulkade informatsiooni töötlemine ning analüüsimine, kasutades sealjuures lihtsaid programmeerimismudeleid. Hadoop on disainitud töötamaks nii üksikul serveril kui ka tuhandetel masinatel, kasutades tarkvarasse sisse ehitatud lahendusi, et tagada riistvaraliste tõrgete automaatne käsitlemine õigeaegselt ilma arvutusprotsessi häirimata. Enamjaolt kasutatakse süsteemis Java programmeerimiskeelt.

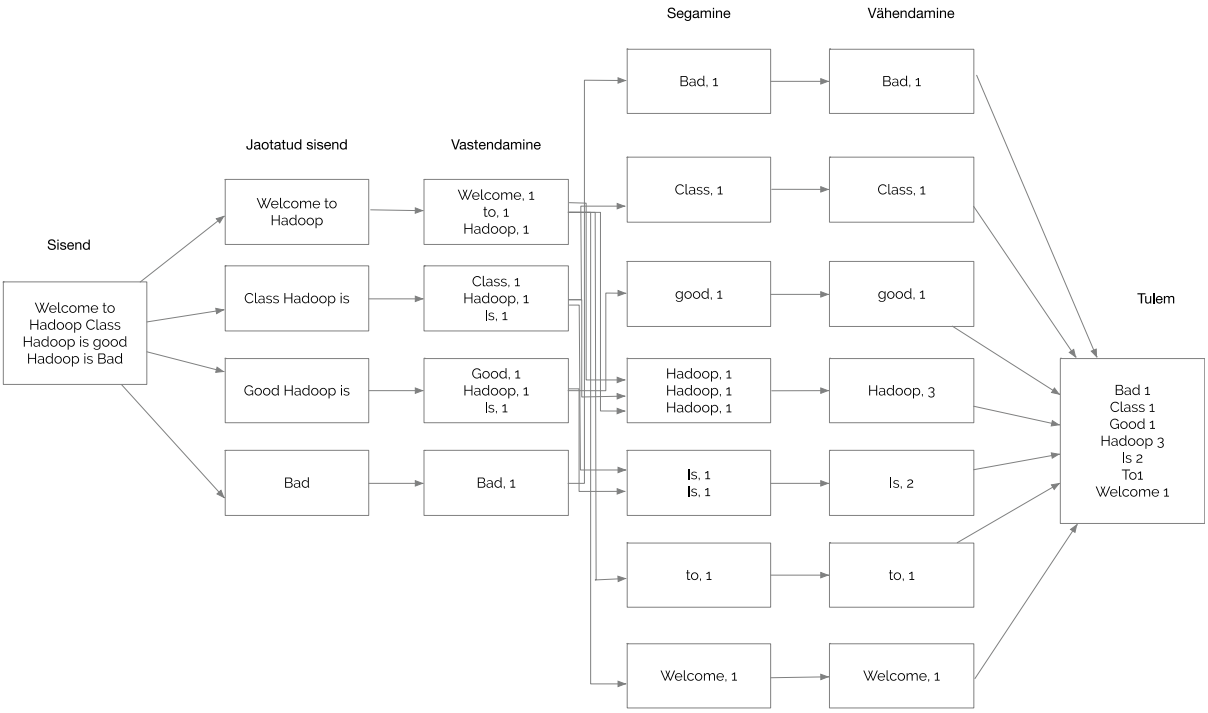


Joonis 1. Hadoop moodulid

Apache Hadoop koosneb peamiselt järgmistest moodulitest:

- Hadoop Common – kesksed tööriistad, mis toetavad ülejäänud moodulite tööd
- Hadoop Distributed File System (HDFS) – serverite vahel ära jagatud failisüsteem
- Hadoop Yarn – tööülesannete planeerija ning arvutusressursside jagaja
- Hadoop MapReduce – MapReduce programmeerimismudelit kasutav süsteem suuremahuliste andmete töötlemiseks

Hadoopi üks olulisim komponent on andmetöötlemise moodul MapReduce. Mapreduce-i moodul koosneb kahest põhilisest sammust: *map* (eesti keeles vastendama) ning *reduce* (eesti keeles kahandama). Esimeses faasis jaotatakse sisendfail, mis koosneb võti-väärtus paaridest, tükkideks ning saadetakse erinevatele masinatele süsteemis. Selle protsessi eest hoolitseb ülemtööline - spetsiaalne masin, mille ülesandeks ongi teiste, alammasinate töö juhtimine. Kui alammasinad on oma tööülesanded saanud, algab *mapping* protsess: igale võti-väärtus paarile rakendatakse sama algoritmi, mis tagastab null või enam võti-väärtus paari. *Reducer* faasis võetakse arvutatud võti-väärtus paarid ning pannakse need omavahel kokku (kahandatakse üheks). *Map* ja *reduce* faaside vahel on veel *shuffle and sort* (eesti keeles sega ja sorteeri) faas, kus *map*-funktsiooni väljund sorteeritakse enne *reduce*-funktsioonile saatmist nende võtmete järgi.



Joonis 2. MapReduce töönäide

Apache Spark on edasiarendus Hadoopist. See on paralleelarvutuste tehnoloogia, mis muudab arvutused kordades kiiremaks kui ilma klastrita lahendustes võimalik oleks. Apache Spark pakub mitmeid erinevaid võimalusi rakenduste kirjutamiseks nagu näiteks Java, Scala ja Python. Spark on avatud lähtekoodiga klasterarvutuse raamistik, mis on arendamise algusest peale disainitud suunaga kiirusele ning kasutab erinevaid optimeerimistehnikaid nagu näiteks mälusisene arvutamine. Tänu sellele võib Spark suurte andmekogude töötlemisel olla kuni 100 korda kiirem kui Hadoop. Näiteks 100 terabaidi andmete sorteerimiseks kulub Sparkil võrdluses Hadoopiga 3 korda vähem aega kasutades samal ajal kõigest ühte kümnendikku jõudlusest. Sparkil on samuti erinevaid mooduleid:

- Spark SQL – võimaldab töödelda andmeid sarnaselt relatsioonilistele andmebaasidele.
- Spark Streaming – võimaldab töödelda andmete vooge.
- MLlib – võimaldab masinõpet, pakkudes selleks mitmeid lihtsaid algoritme.
- GraphX – võimaldab Sparki kasutada graafi andmemudelite põhjal tehtud arvutuste jaoks.

//MapReduce vs Spark kokkuvõtvalt

Hadoop MapReduce on hea:

- Suurte andmete lineaarseks töötlemiseks
- Odavaks töötlemiseks kui kiireid tulemusi ei ole vaja

Apache Spark on hea:

- Kiireks andmetöötluseks
- Iteratiivseks töötluseks
- Graafide töötlemiseks
- Masinõppeks, andmekogude ühendamiseks

//Veel tehnoloogilisi komponente

Block Storage - HDFS, Ceph

Tiered Storage - Alluxio

Structured Memory - Apache Arrow

Messaging - Kafka

MPP SQL - Impala, Drill, SparkSQL

NoSQL Databases

- Columnar: Cassandra, HBase
- Document: MongoDB
- Key-Value: Redis
- Graph: Blazegraph

Computation Engine – Spark

Stream Processing – Flink, Spark Streaming

Search – Solr, Elasticsearch

Data Pipeline – StreamSets, Cask

Data Governance - Cloudera Navigator

General Purpose Resource Management – Mesos,

Job Scheduler - Aurora, Chronos

App Initialization - Marathon, Swarm

Scientific Programming – Julia (also my favorite)

Data Warehouse Migration - Cloudera Navigator Optimizer

Machine Orchestration - Chef, Puppet

//Masinõpe

Masinõpe on arvutiteaduse haru, mis annab arvutitele õppimisvõime ilma neid otseselt selleks programmeerimata. Kõige lihtsam viis antud kontseptsiooni seletada on läbi järeldamise: A'st B järeldamine. Kui tavalise IT-arenduse puhul teame nii soovitud tulemust kui ka viisi selle saavutamiseks, siis masinõppega lahendatakse tavaliselt selliseid probleeme, kus eesmärk on hästi teada, kuid isegi valdkonna eksperdid ei oska väga hästi öelda, kuidas sinna jõuda.

//Masinõpe ja traditsiooniline moodne tarkvaraarendus

Esimene ülesanne masinõppe projekti alustamisel on aru saada, millisel tasemel andmetega on tegu ning kuidas on antud kasutusjuhul lahendatud:

- Andmete kättesaadavus
- Andmete kvaliteet
- Andmete digitaliseerimine

//Enim kasutatavad programmeerimiskeeled masinõppes

- Microsoft R Open
    - https://mran.microsoft.com/
- Python:
    - https://www.python.org/

    - Jupyter + IRkernel
        - https://jupyter.org/
        - https://github.com/IRkernel/IRkernel
- Julia:
    - https://julialang.org/

//Peamised masinõppe tüübid

Klassikaliselt jaotatakse masinõppe ülesanded kolmeks:

- juhendatud õpe (supervised learning)
- juhendamata õpe (unsupervised learning)
- stiimulõpe (reinforcement learning)

//Juhendatud õpe

Kirjeldame kõik õppemeetodid lahti, alustades juhendatud õppest. Iga masinõppe ülesanne vajab treenimiseks andmeid. Juhendatud õppe eesmärk on luua mudel, mis võtab sisse andmepunkti ja ennustab selle kohta midagi. Näiteks:

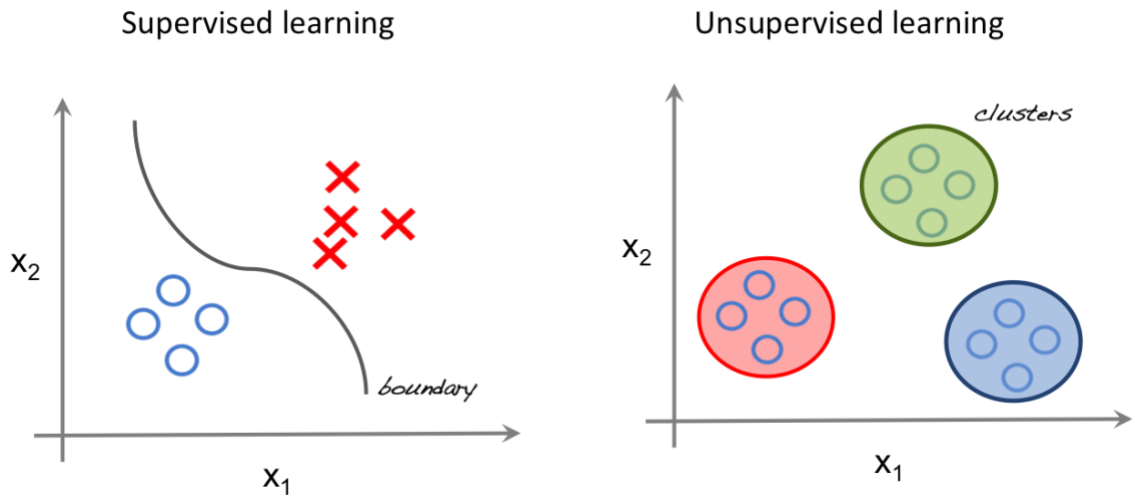
Sisend	**Väljund**	**Kasutusjuht**
E-mail →	spam?	-spämmi filter
Audio →	text	-kõnetuvastus
Eesti keel →	Hiina keel	-masintõlge
Online reklaam →		-kas kasutaja vajutab reklaamile?
Radari informatsioon →		-autode asukohad

Võime öelda, et 99% ärilisest väärtusest luuakse täna ühte tüüpi juhendatud masinõppega.

//Juhendamata õpe

Juhendamata õpe erineb juhendatud õppest selle poolest, et juhendajat (treeningandmeid) ei ole. See tähendab, et me ei ürita enam ennustada mingit konkreetset muutujat.

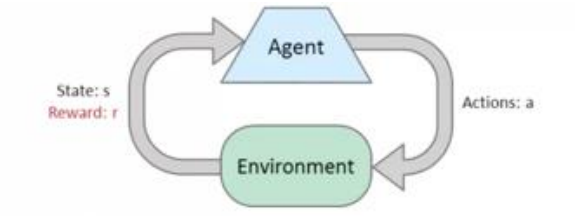
Juhendamata õppe eesmärk on leida andmetest struktuuri. Juhendamata õpet kasutakse andmetest mustrite või muude parameetrite leidmiseks. Seda kasutatakse tihti siis, kui me ei tea andmete kohta peaaegu mitte midagi.



Joonis 3. Juhendatud ja juhendamata õppe vahe

//Stiimulõpe

Stiimulõpe on tagasisidestatud õppimise meetod. Sisuliselt me anname algoritmile ülesande ning ta peab õppima seda täitma. Lihtne näide on näiteks malemäng. Stiimulõppe põhikomponendid on keskkond (malemäng), agent (mängija – see võib olla inimene või arvutiprogramm), tagasiside (*reward* – võit/kaotus) ja tegevused (käigud).



Joonis 4. Stiimulõpe

Peamine idee:

- *Saada võidu/kaotuse parameetri käest tagasisidet*
- *Peab õppima võitma*
- *Kogu õppimine baseerub võitmisel/kaotusel*

//Projektijuhtimine masinõppes

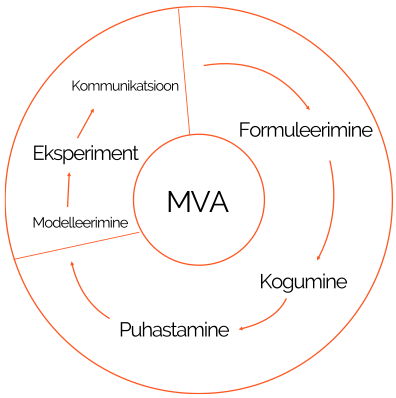
Kuigi esialgu võib arvata, et masinõppe projekte võib juhtida samamoodi nagu traditsioonilisi tarkvaraarenduse projekte, siis tegelikult see nii ei ole. Eduka masinõppe projekti läbiviimine eeldab tavaliselt organisatsiooni protsessilist muutmist. Masinõppe projektide edu võib üles ehitada kolmele peamisele sambale:

- Strateegiline andmete kogumine – andmeid tuleb koguda pidevalt ning eesmärgiga
- Ühtlustatud andmelaod – kõik andmed peavad olema kaasajastatud ning ühtlaselt kättesaadavad
- Automatiseerimise pidev otsimine

//Projekti läbiviimise metoodika

Tavaline masinõppe projekt algab ettevalmistusfaasiga, kus analüüsitakse läbi probleem, määratakse käideldav andmehulk (sh treenimiseks piisava andmekoguse hankimine ja ettevalmistus), teostatakse esmane andmeanalüüs ja andmete puhastamine. Seejärel valitakse sobivad masinõppemudelid, seatakse mudelitele mõõdikud ja valideerimistingimused ning seejärel alustatakse 1-2 nädalaste iteratsioonidega sobiva(te) mudeli(te) treenimiseks ja väljatöötamiseks.

Projekt viiakse läbi iteratiivselt. See tähendab, et esialgne prototüüp ei oma täisfunktsionaalsust, kuid kirjeldab toote iseloomu ning seda on võimalik edasi arendada.



Joonis 5. Minimaalse vajaliku algoritmi meetod

Peamised sammud andmeteadusprojektis on:

- Koguda andmeid
- Analüüsida andmeid (iteratiivselt)
- Tekitada hüpotees
    - Analüüsida andmeid uuesti
- Lansseerida toode

//Masinõppe projekti rollid

Üldiselt on masinõppe projekti koosseisu pigem keeruline määratleda, kuna projektid vajavad väga erinevaid ressursse. Küll aga on vajalik teada peamisi rolle ning nende funktsioone:

- Tarkvarainsener
    - kirjutab tarkvara koodi nagu näiteks funktsioon / alamprogramm
- Masinõppeinsener:
    - vastutab mudelite loomise eest
- Rakendatud masinõppe (ML) teadlane
    - roll masinõppeinseneri ja teadlase vahel
- Andmeteadlane:
    - uurib andmeid ja annab teadmisi äriotsuste tegemiseks. Vajalikud on sügav teadmine kõrgemast matemaatikast, Pythoni keele oskus, arusaam enim kasutatavatest andmeteaduse ja masinõppe raamistikest (Tensorflow, Keras, etc)

- Andmete insener:
    - veendub, et andmed oleksid turvaliselt ja kulutõhusalt kergesti ligipääsetavad. Eeldab Java/Scala keele teadmisi, arusaamine Hadoop'ist, Spark'ist ja muudest suurandmete arhitektuuridest
- Tehisintellekti / Krati (AI) tootejuht: defineerib, mida ehitada, mis on väärtuslik ja teostatav

//Masinõppe kasulikkuse hindamine

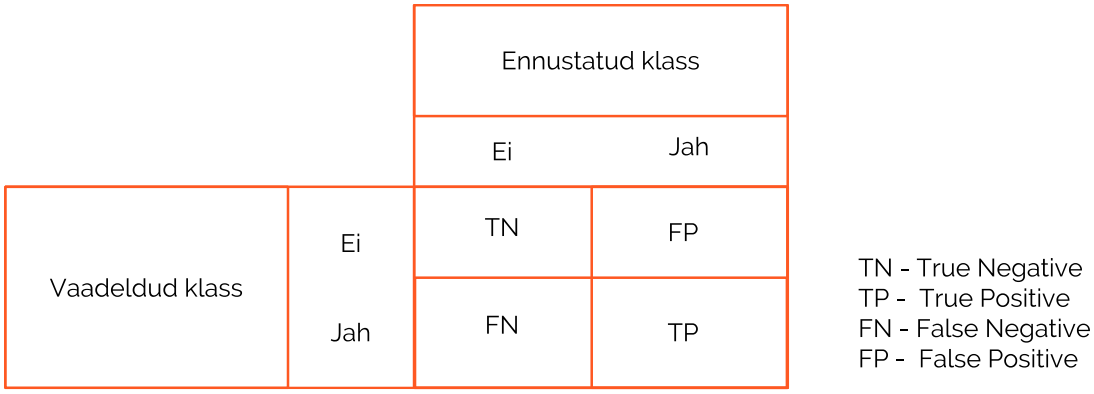
Masinõpe ei ole nn hõbedast kuulike, mis suudab teha kõike ning mida võib kasutada igas olukorras. Seetõttu on väga oluline adekvaatselt hinnata, mida masinõpe võimaldab teha ja mida mitte.

Masinõpe on väga hea:

- Funktsiooni õppimiseks, kus on selgesti defineeritud sisendid ja väljundid (klassifikatsioon, regressioon, ennustamine)
- Ülesannete jaoks, kus on võimalik tagada selge tagasiside mõõdetavate tulemustega
- Olukordades, kus ei ole pikki loogikaahelaid ning kus ei ole vaja taustateadmist
- Olukordades, kus ei ole vaja täpset detailset seletust, kuidas otsus tehti.
- Olukordades, kus on taluvus otsuse vea tolerantsi osas
- Olukordades, kus funktsioon või nähtus ei muutu ajas väga kiiresti.

//Eksimise Maatriks

Selleks, et hinnata masinõppe algoritmi täpsust, on võimalik kasutada eksimise maatriksit ning järgnevaid parameetreid.



Joonis 6. Eksimise Maatriks

Õigsus (*Accuracy*) mõõdab üldist mudeli klassifikatsiooni täpsust.

$$Accuracy = (TP+TN)/(TP+FP+TN+FN)$$

Täpsus (*Precision*) mõõdab ühe klassi täpsust kui ennustus on "Jah", siis kui tihti see on õige.

$$Precision = TP/(TP+FP) = TP/ (Ennustas\ Jah)$$

***Sensitivity* or *Recall* (Saagis)**: kui vastus on "Jah", siis kui tihti algoritm ennustab "Jah"

$$Recall = Sensitivity = TP/(TP+FN) = TP/ (Actual\ Yes)$$

Specificity mõõdab tõest negatiivset. Kui tegelikkus on "Ei", siis kui tihti on ennustus "Ei"

$$Specificity= TN/(TN+FP) = TN/(Actual\ No)$$