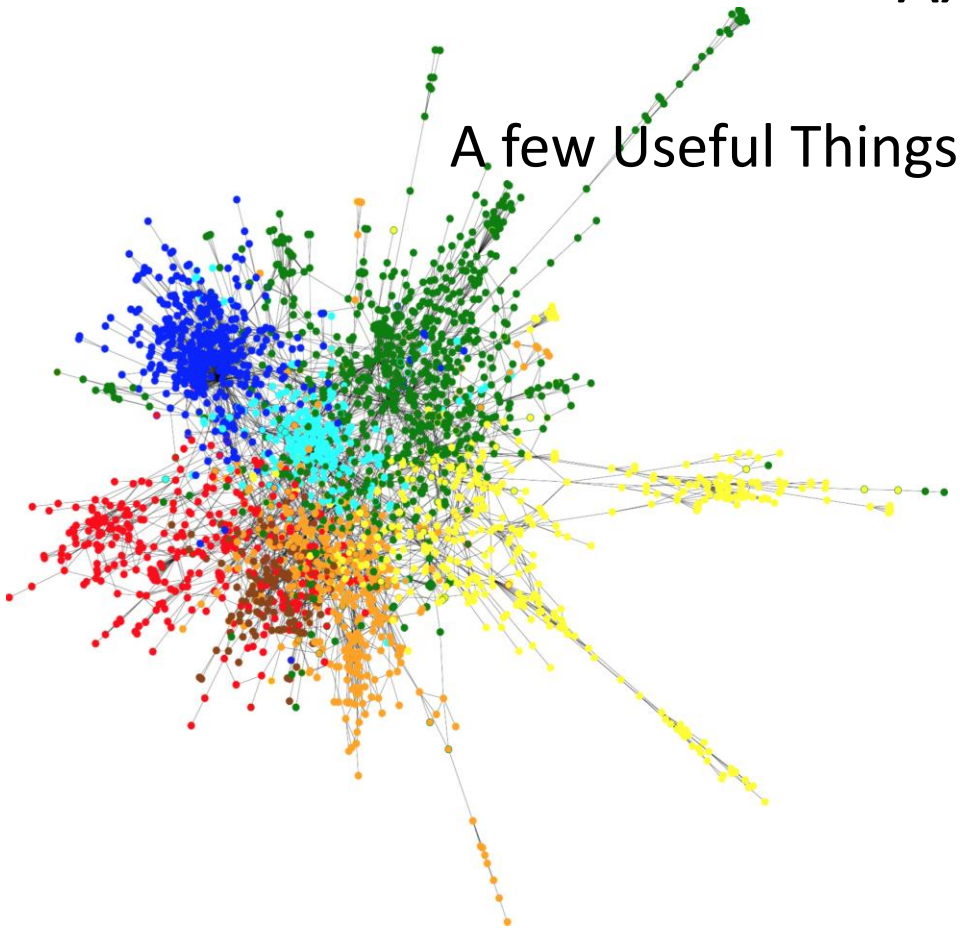
Axians Summer Academy 2021



A few Useful Things to know about Machine Learning and visualization

Merelbeke, September 21, 2021

A few Useful Things to know about Machine Learning and visualization

What do you want to take away from this session ?

Who are you ?

Ideas conveyed

Provide a fast track / basic concepts / fundamentals/ toolchest

- Data analysis
- Data Science Methods (these differ from application development methods)
- Data visualization using pre-build Python / Plotly scripts
- Extracting data from Excel and Flat Files using Python / Pandas.
- Unsupervised Machine Learning - Data Clustering(K-Means)
- Supervised Machine Learning - Linear regression
- Supervised Machine Learning - Classification (Logistic Regression)

BTW. Next week's session is about MS Power BI, notorious for not being able to export diagrams.

Ideas conveyed (cont’d)

Concepts of Machine Learning (ML)

There is ample information available on machine learning algorithms. The majority of the documentation focuses either on the **information theory** of machine learning or on the development of machine learning applications by relying on readily available machine learning **libraries**.

The major objective of the presentation is to convey the notion that writing your own machine learning source code is a **viable** exercise and also a **gratifying** experience, i.e. **fun**.

Collateral

Handouts, sample data and Java source code at http://github.com/koenberton/AxiansSummerSchool2021

- Presentation (this.)
- Handouts (reference purposes)
- Lab Notes (installation, scripts, etc)
- Data

Introduction

ML

Definition

One perspective

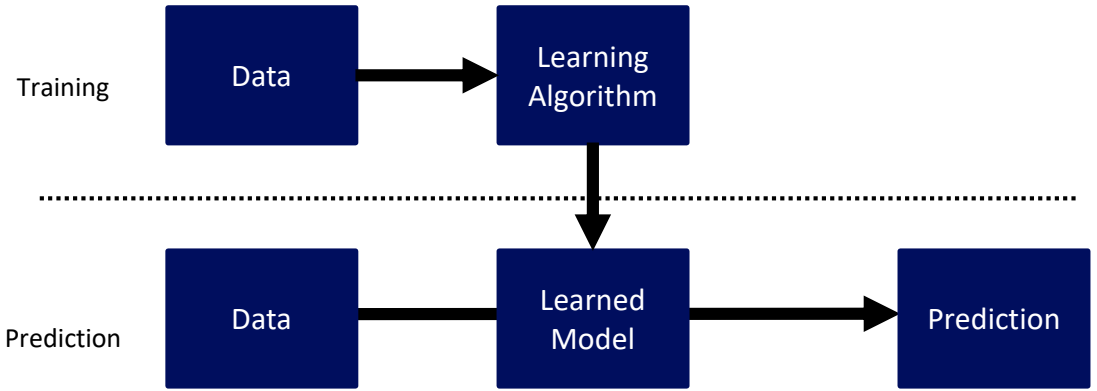
Trends

Definition

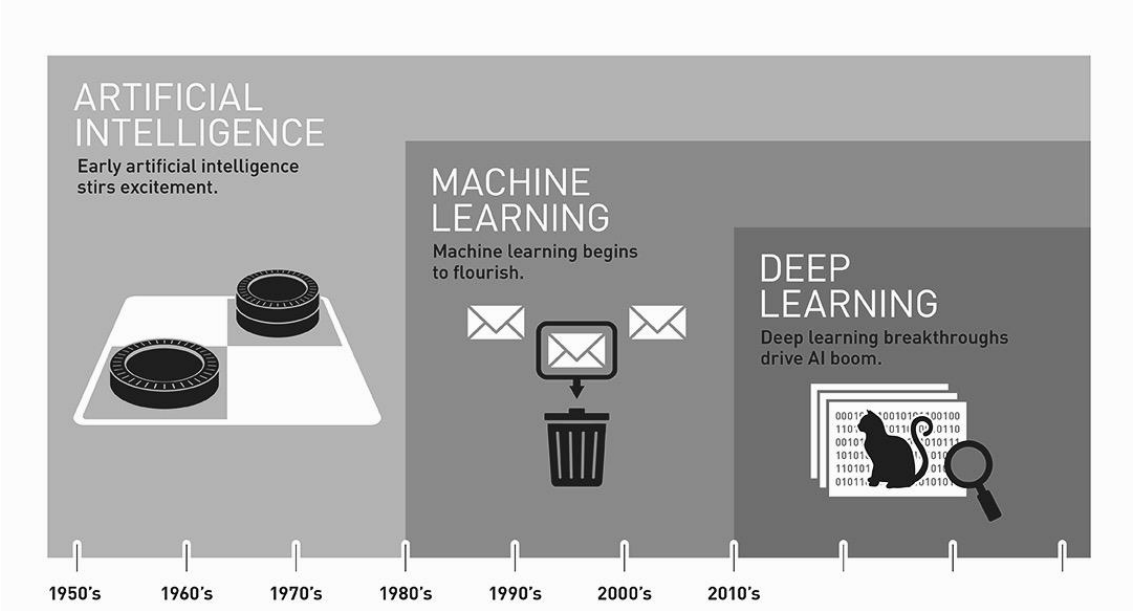
Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.
[Wikipedia]

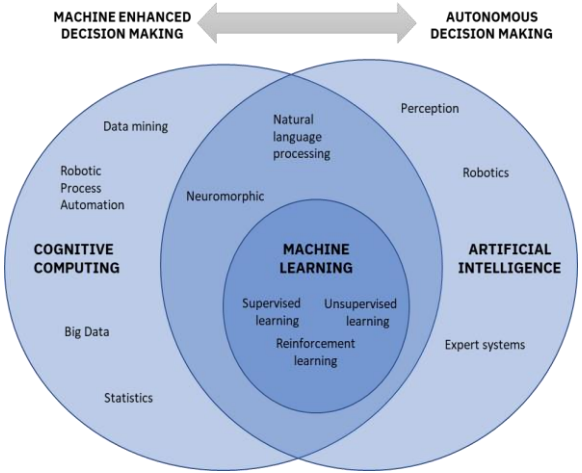
Machine Learning is a type of AI that provides computers with the ability to learn **without being explicitly programmed.**

The core machine learning process (or workflow) is the following:



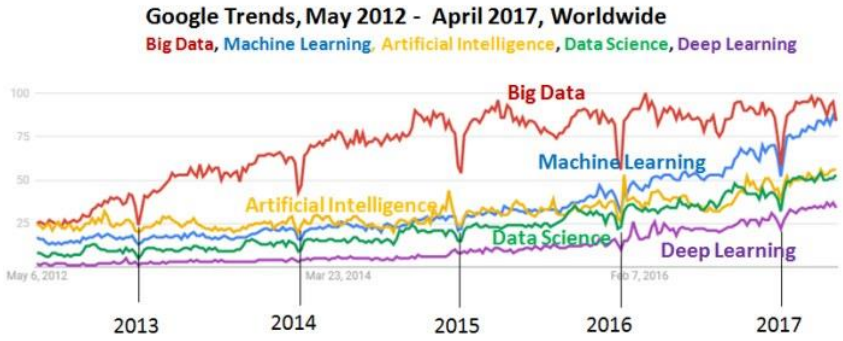
One perspective : AI, ML and Deep learning



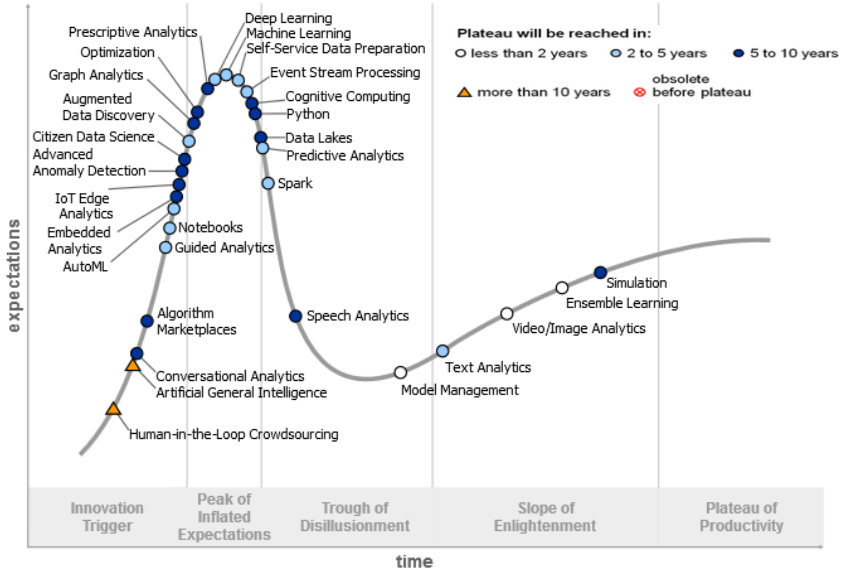


Just buzzwords?

Evolution



Hype cycle



Source : Gartner 2017

The learning Process

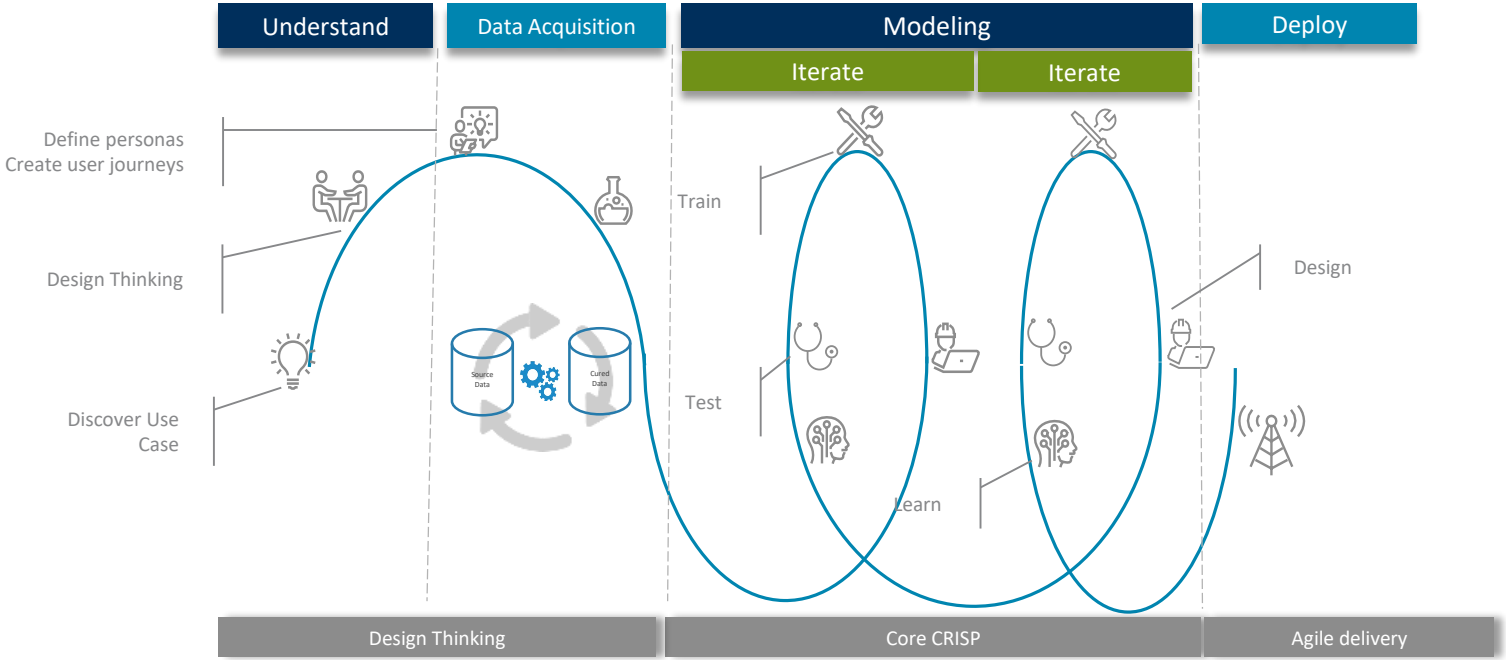
Understand

Data

Model

The learning process

- Defining and understanding a problem to be addressed
- Data acquisition, pre-processing and feature engineering
- Definition, training and evaluating of a learned model
- Understand the results and metrics produced by a model and gather feedback
- (if applicable) select between various models
- Deploy the model that has been chosen.



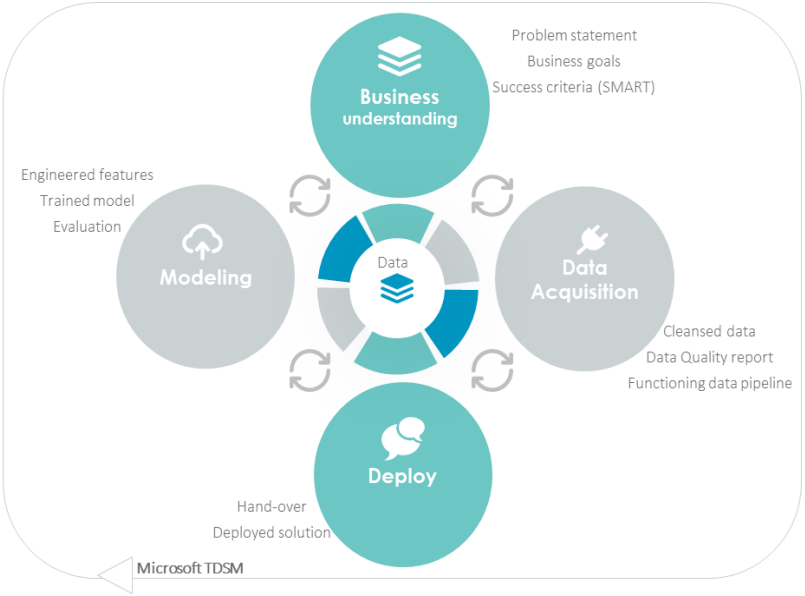
ML Methods

CRISP-DM

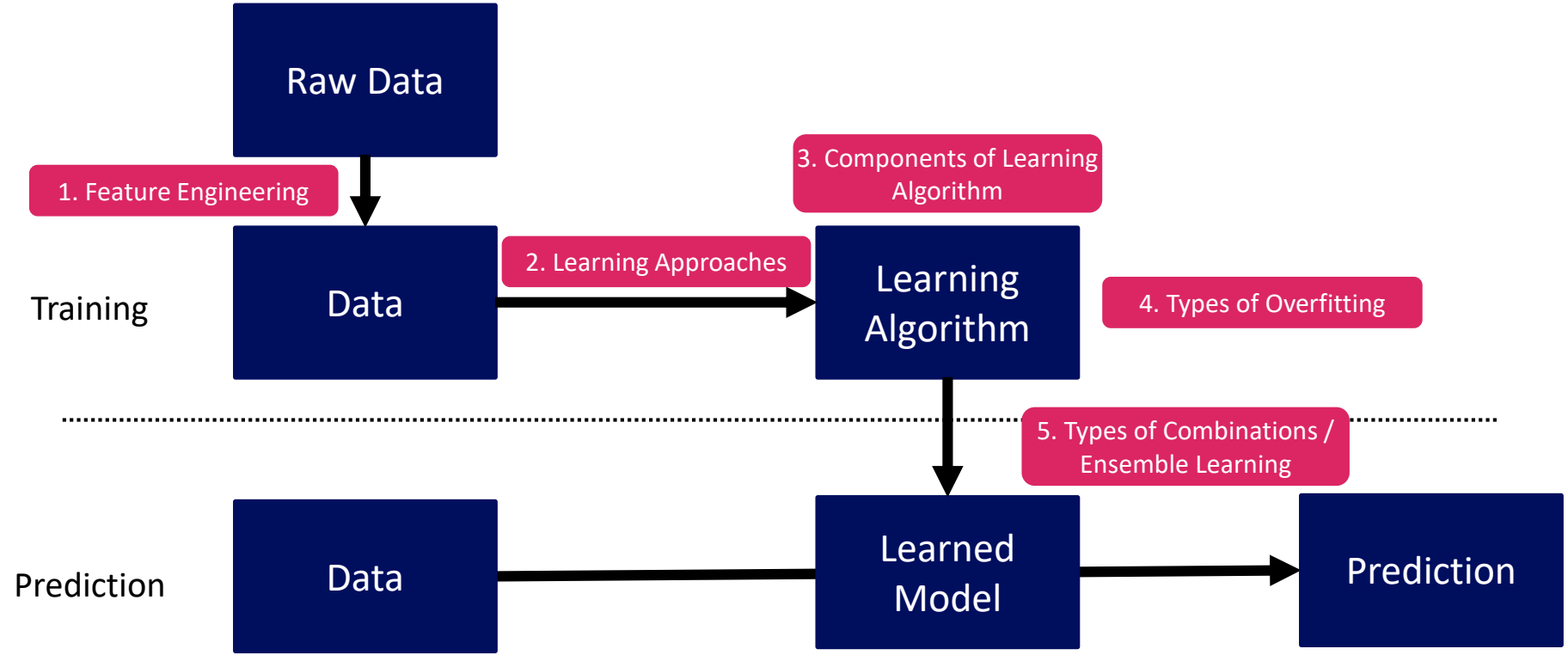
Cross-industry standard process for data mining

TDSM

Team Data Science Method



Crucial Components of Machine Learning



Understand

To gather an understanding of the (business) problem and (business) variable to predict.

This is achieved through interviews, workshops, reading documents or any other technique that is deemed appropriate.

NOTE - An important objective of this phase is to define the success (or acceptance) criterions

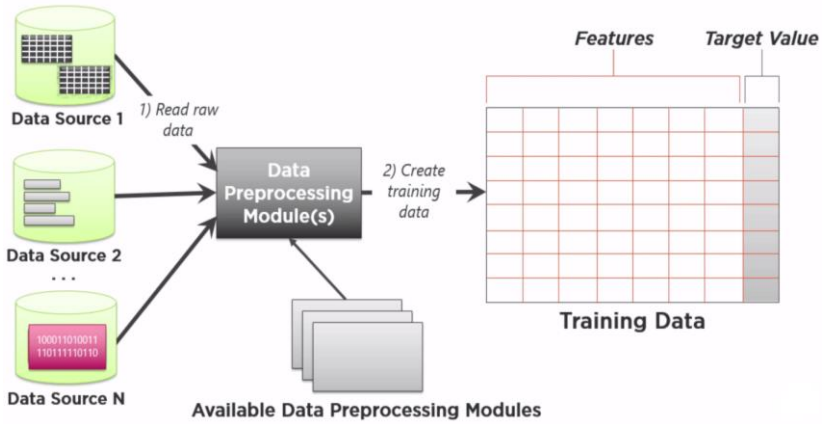
It is the least formal phase of the learning process

Data retrieval

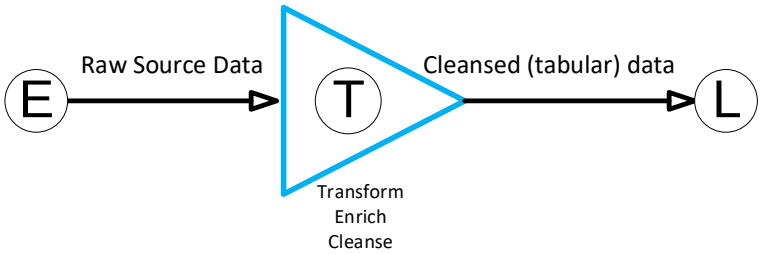
To gather insight on the structure of the data that will be used for training, testing and running a ML model.

Relies on well-accepted Information Management disciplines : data profiling, ETL, feature engineering.

Data Retrieval Process



Extraction – Transform - Load



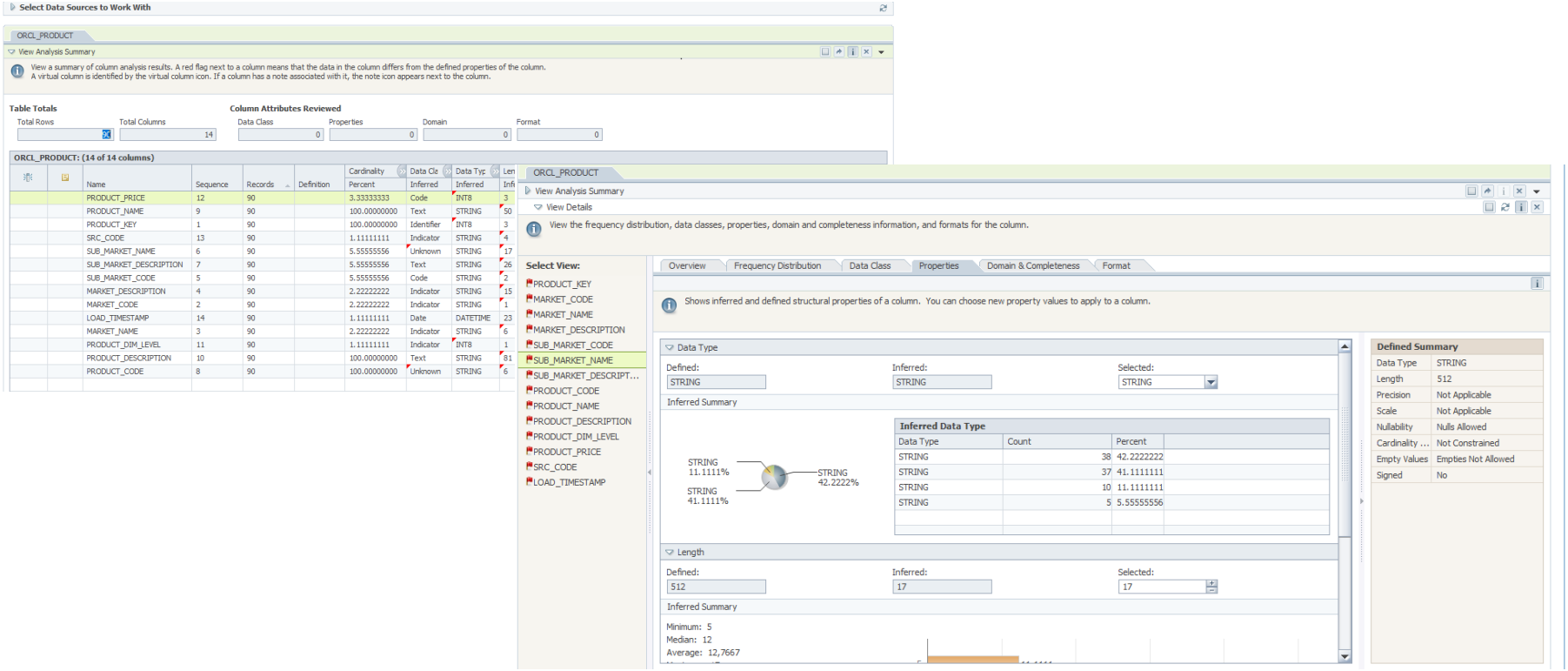
It is good practice to just "look" at the data

Data retrieval – Class exercise

```
CUST_ID|KUNDEN_NO|REGION_LONGITUDE|LAT|CUST_NAME|CUST_LOCATION|CREA_DT|AOB
13254|02-45554|4.330605696|50.873.874,11|Brooks|"Merelbeke"|01-Jan-2010
18426|08-XC-7878|4.350452707|50.876.494,73|Les établissements des Frères Mérode|"Sint-
Amandsberg"|14-Jul-1789|Gesloten op zondag
21552|00-1234|4.70125841|50.878.494,81|DAKWERKEN COSTERMANS|"Deinze"|20-Sep-2021
24154|789-FRANCE-01|4.963176672|50.970.438,82||"Eeklo"|20-Sep-2021|TODO: opzoeken rechtsvorm in Staatsblad
48447|TX-78C|5.969730949|50.612.245,40|"SIMAR"|"Zelzate"|18-Oct-2025
1639|Je ne sais pas|4.378546286|51.209.895,65|New Homes|"Oostakker"|15-Feb-2020|Voorheen : Novel
Homes
44327||4.187360991|50.39835884|S.S.P.|"De Pinte"|99-Apr-2019
27727|00-1234|3.286103516|51.184.527,78|ATENOR REAL ESTATE|"St-Maartens Latem"|08-Mai-2018|일욜에
4.475582066|4.475.582,66|51.02989317|Mehrzweckhalle Herbesthal, G.o.E., Centre Educatif polyvalent
Herbestahl|"Herbestahl"|19-Mar-2020|Fout bij printen factuur
35509|99-99999|4.040356698|50.939.342,33|Cisco|"Heusden"|10-Dec-15
24597|AZ-7878|4.958247855|50.817.557,03|Comat Conter et Cie SARL|"Melle"|11-Nov-2014
21553|AZ-7879|4.70125841|50.878.494,81|Daniel PIRON et Cie|"Wetteren"||Cie neemt de telefoon niet op
als ik bel
```

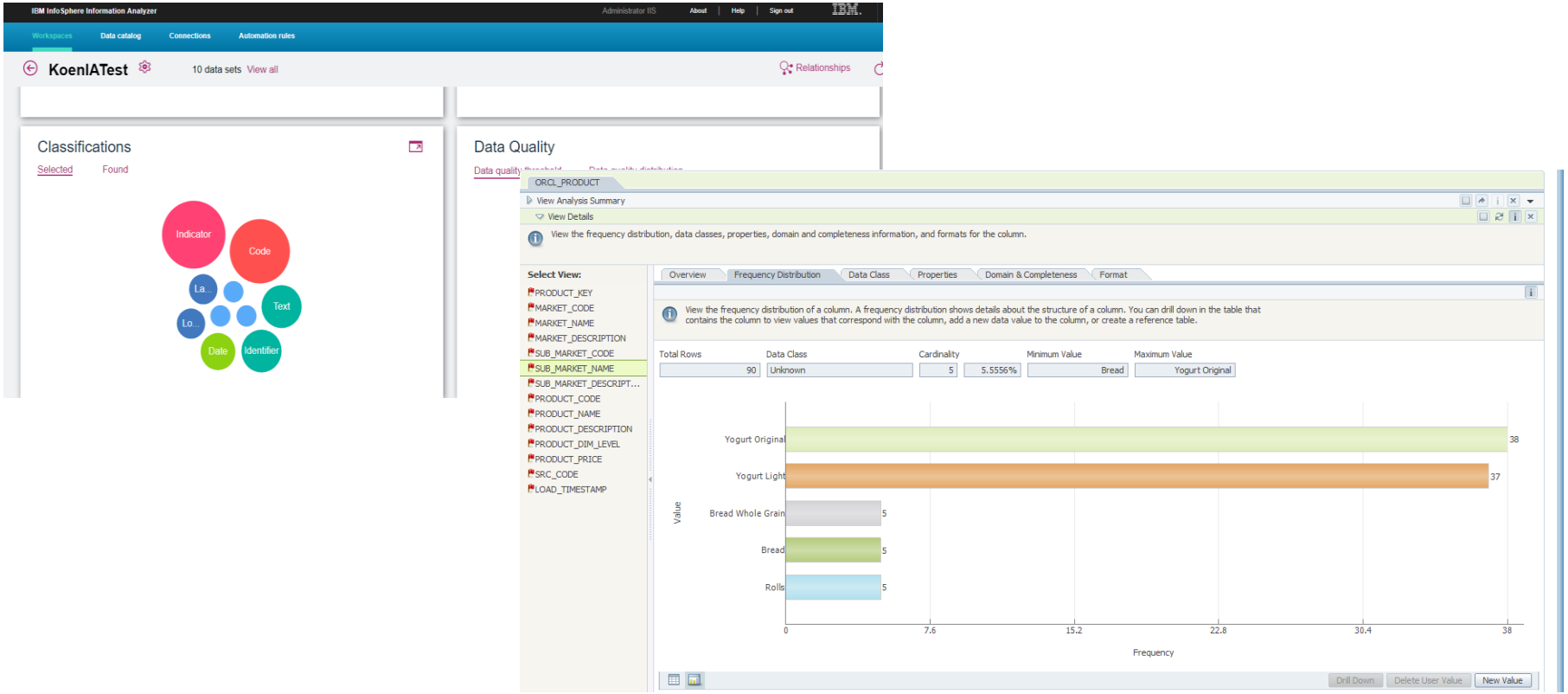
What are your thoughts?

Data retrieval – Profiling



See : IBM WebSphere Information Analyzer and Data Quality Assessment" Redbook and Axians Demo Environment

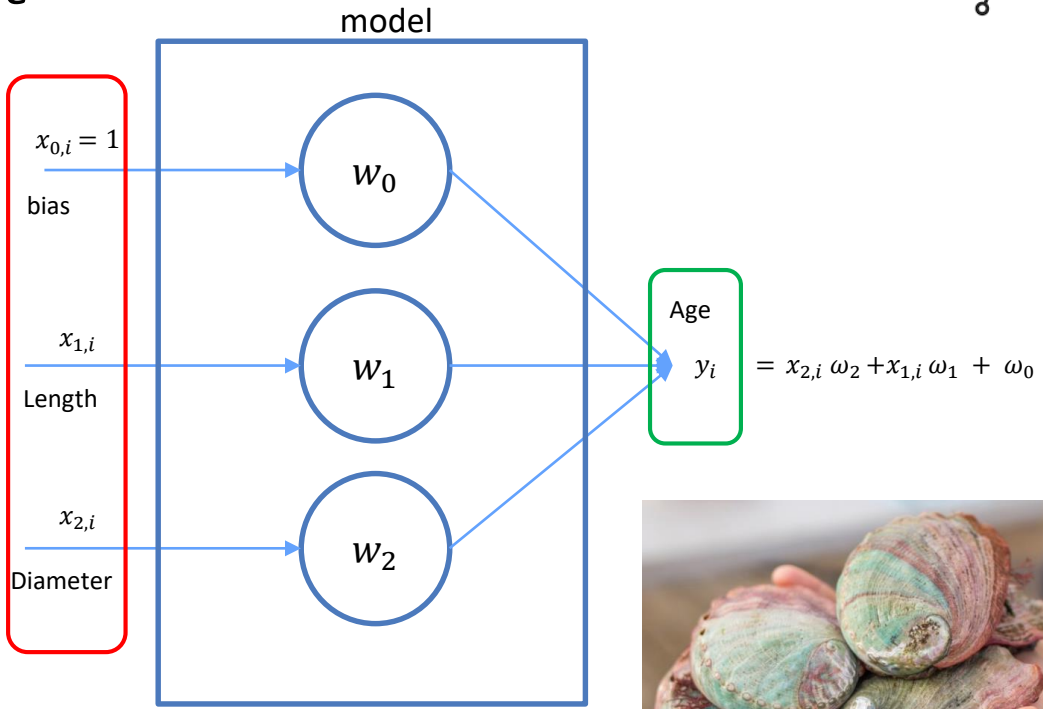
Data retrieval – Profiling



Features – Brining it all together

Predicting continuous target value

```
@RELATION abalone

@ATTRIBUTE Length          NUMERIC
@ATTRIBUTE Diameter        NUMERIC

@ATTRIBUTE Age             NUMERIC

@DATA
0.455,0.365,1
0.35,0.265,2
0.53,0.42,1
0.44,0.365,3
0.33,0.255,4
0.425,0.3,6
0.53,0.415,6
0.545,0.425,7
```

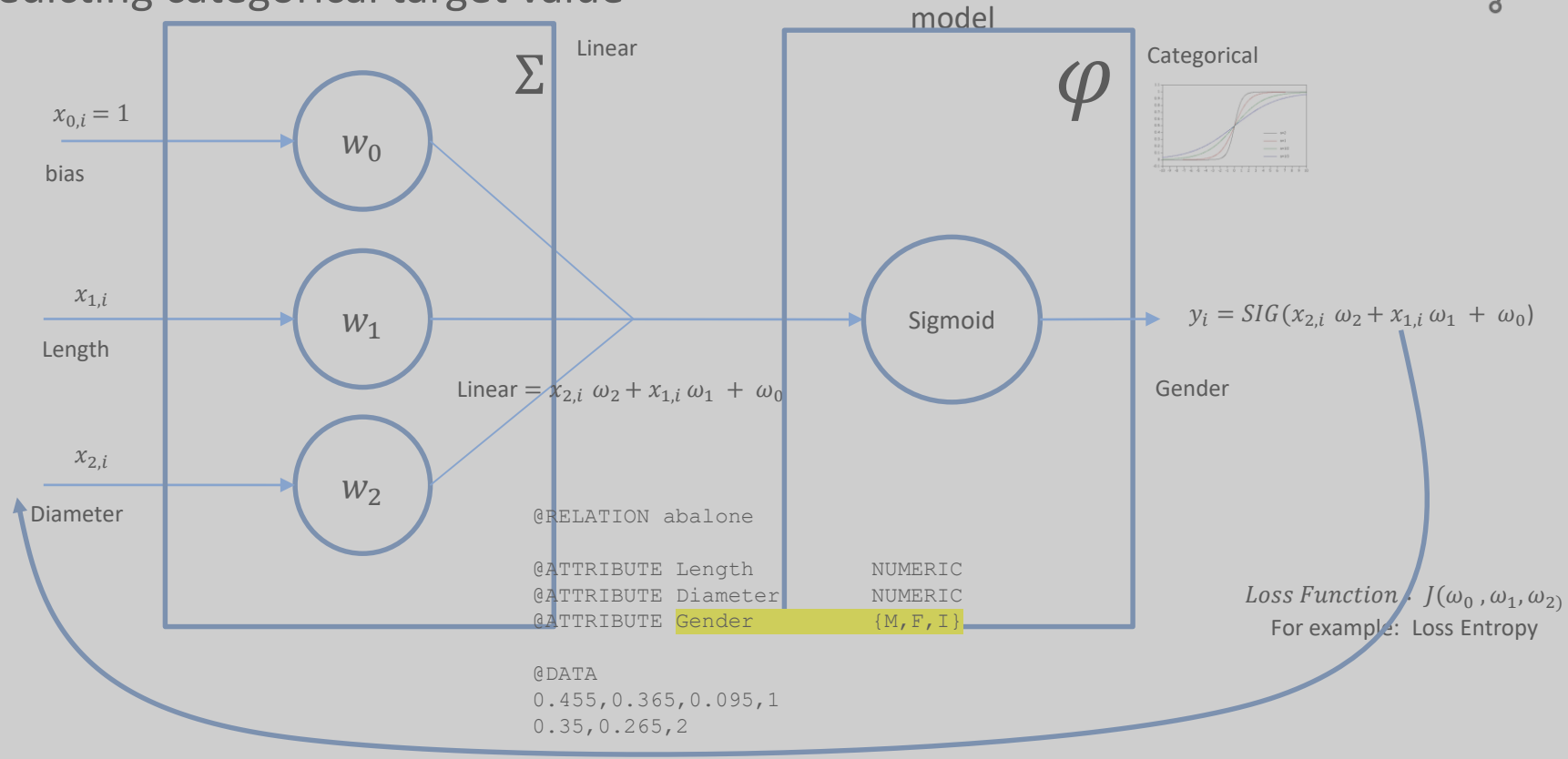


Example ARFF data format

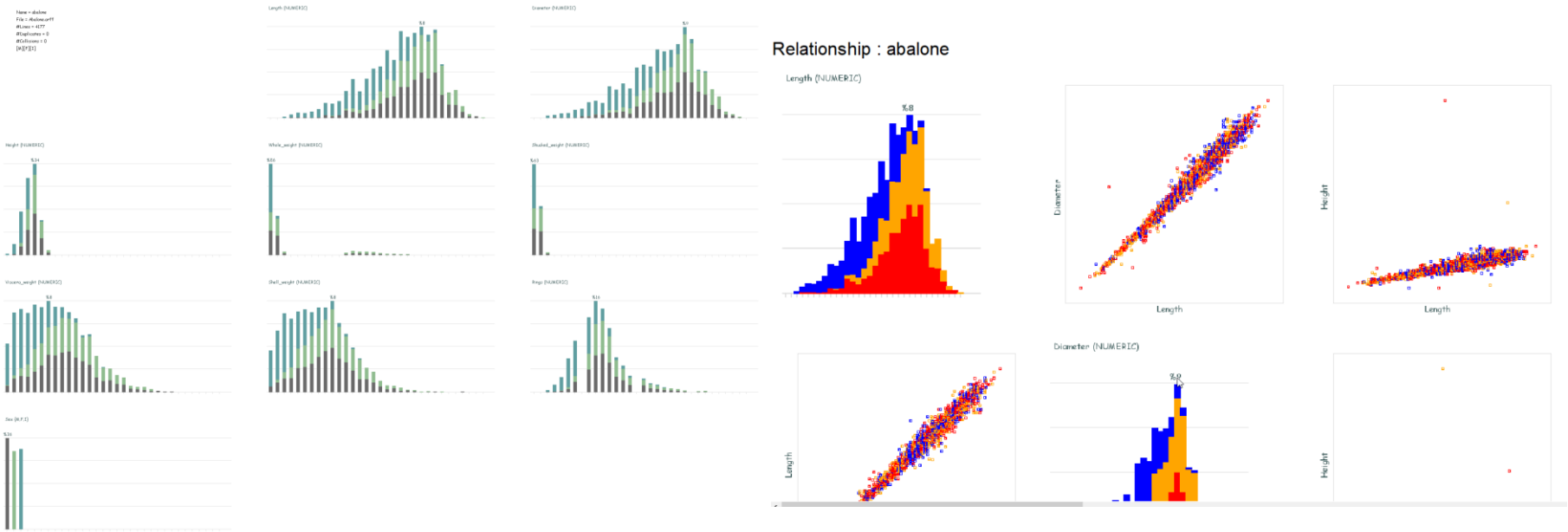
NOTE – If we want to predict "Gender" we need a classification, not a regression algoritm

Features - labels – model

Predicting categorical target value



Features – All histograms ande scatterplots



See : DataSample folder for the full-scale images
NOTE – scatterplots have been abbreviated

NOTE => Kurstosis and Skew

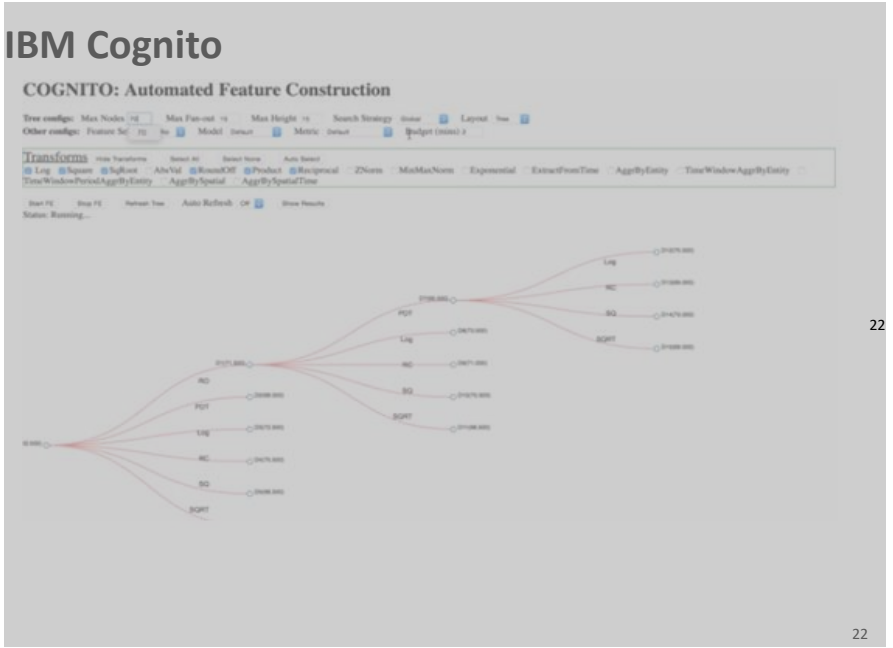
Data Retrieval : Feature Engineering

Feature engineering is the process of using domain knowledge, expertise or creative insights to create features (attributes, classes, data) that make machine-learning algorithms work.

Feature engineering is fundamental to machine learning and is both difficult and expensive. [Wikipedia]

Feature Engineering is foremost a manual approach, it is therefore worthwhile automating it.

IBM Cognito



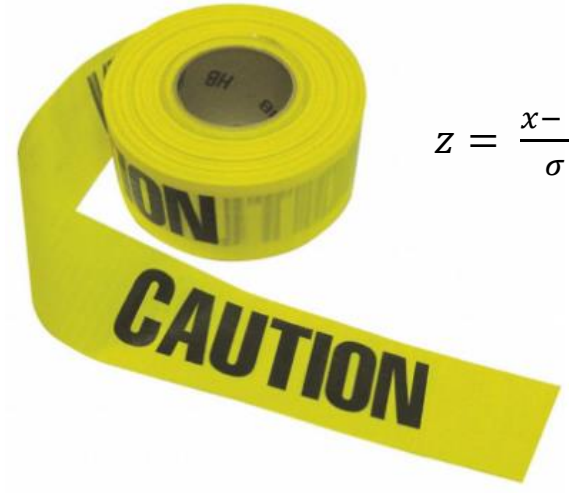
22

Data Retrieval – Feature Engineering

Life saver : <u>Normalization</u>

The integrity (coherence) of the data is maintained, but the stochastic properties are improved.

It's easy as rain :

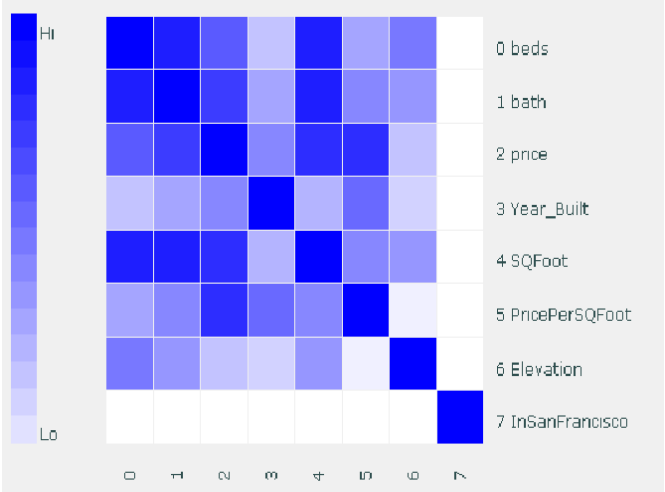


$$z = \frac{x - \mu}{\sigma}$$

Pearson Correlation

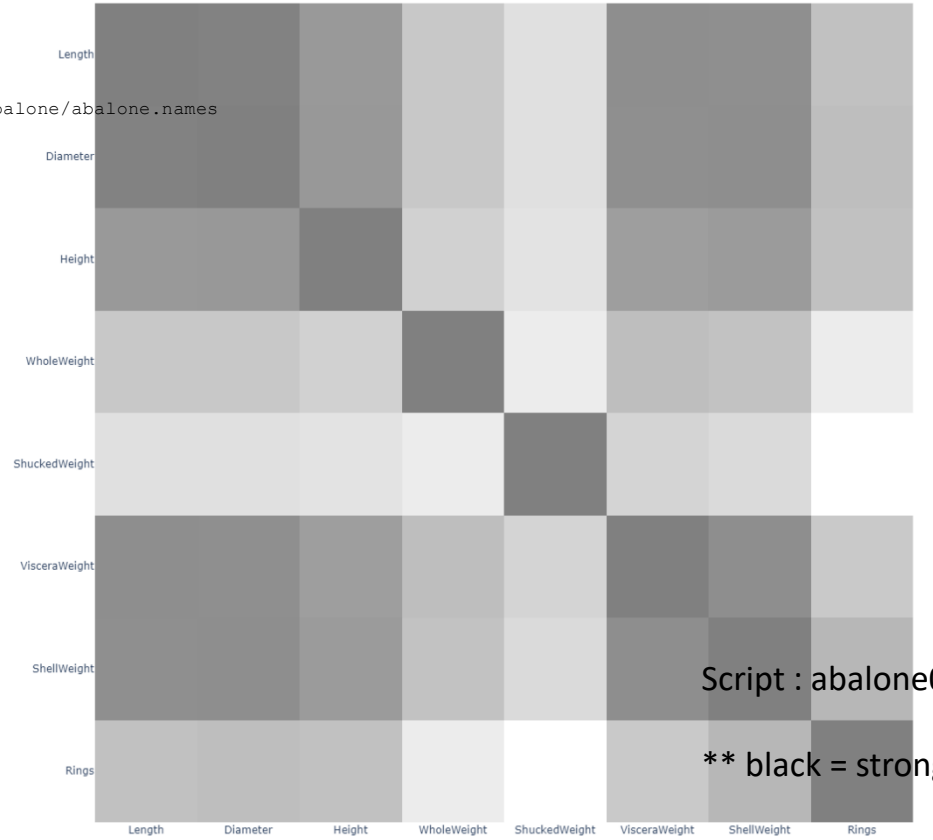
$$R_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

1 = strong correlation



Data Retrieval – Feature Engineering - Correlation matrix

```
%https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.names

@RELATION abalone


@ATTRIBUTE Length          NUMERIC
@ATTRIBUTE Diameter        NUMERIC
@ATTRIBUTE Height          NUMERIC
@ATTRIBUTE Whole_weight     NUMERIC
@ATTRIBUTE Shucked_weight NUMERIC
@ATTRIBUTE Viscera_weight NUMERIC
@ATTRIBUTE Shell_weight    NUMERIC
@ATTRIBUTE Rings           NUMERIC
@ATTRIBUTE Gender          {M,F,I}


@DATA
0.455,0.365,0.095,0.514,0.2245,0.101,0.15,15,M
0.35,0.265,0.09,0.2255,0.0995,0.0485,0.07,7,M
0.53,0.42,0.135,0.677,0.2565,0.1415,0.21,9,F
0.44,0.365,0.125,0.516,0.2155,0.114,0.155,10,M
0.33,0.255,0.08,0.205,0.0895,0.0395,0.055,7,I
0.425,0.3,0.095,0.3515,0.141,0.0775,0.12,8,I
0.53,0.415,0.15,0.7775,0.237,0.1415,0.33,20,F
0.545,0.425,0.125,0.768,0.294,0.1495,0.26,16,F
0.475,0.37,0.125,0.5095,0.2165,0.1125,0.165,9,M
0.55,0.44,0.15,0.8945,0.3145,0.151,0.32,19,F
0.525,0.38,0.14,0.6065,0.194,0.1475,0.21,14,F
0.43,0.35,0.11,0.406,0.1675,0.081,0.135,10,M
```



Script : abalone04.txt

** black = strong correlation

Data Retrieval – Feature Engineering

Information entropy

Entropy defines “how much information is present in a set of data”.

$$entropy\ (S) = -\sum_{c=1}^{K} p_c\ LN(\ p_c)$$

Where

- S is the sample of data, comprising K different classes, e.g. sunny, overcast, etc.
- p_c is the proportion of times the category c (e.g. sunny) occurs in the sample S

Examples

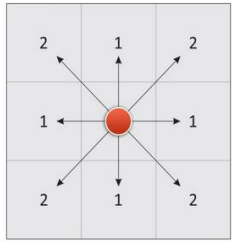
- Entropy 0 : all samples have the same category. So the lower the entropy the less chaos (just as in thermodynamics)
- Higher entropy means more random (highest value is LN (p_c)

25

Vector distances

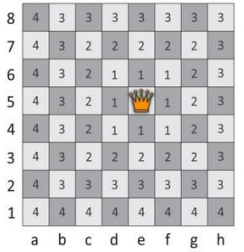
ML relies extensively on distance metrics

Euclidean Distance



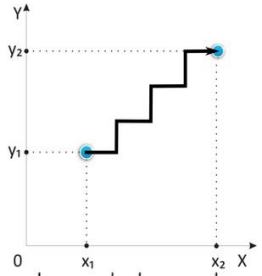
$$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

Chebyshev Distance



$$\max\left(|x_1-x_2|,|y_1-y_2|\right)$$

Manhattan Distance

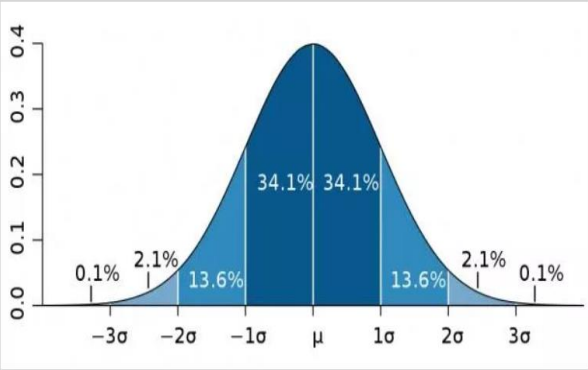


$$|x_1-x_2|+|y_1-y_2|$$

Data Retrieval – Feature Engineering - Tips

Outlier detection

Almost 70% (68.2) of the data is situated between the boundaries of the mean and plus and minus the standard deviation.
99% is situated beyond the mean plus or minus 3 times the standard deviation.



CAVEAT - only for a normal distribution

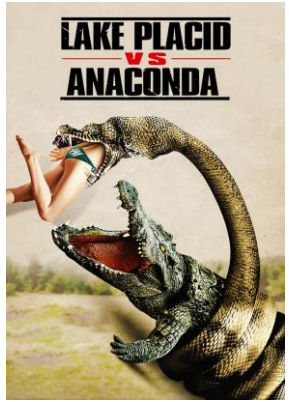
Dimensionality reduction

Principal component analysis

26

Hands-on

1. Installation of Anaconda on your laptop
   - Anaconda Individual Edition is the world's most popular Python distribution, it is the platform of choice for Python based data science developments.
2. Installation of Python and required libraries
3. Data Visualization examples
   - Sample scripts have been created to demonstrate the visualization capabilities of plotly which are not available in MS PowerBI (come & see next week). A script development template/framework is proposed for extracting data from Excel into Pandas Dataframes. The idea is that you can leverage (i.e. copy/paste/adapt) these scripts.



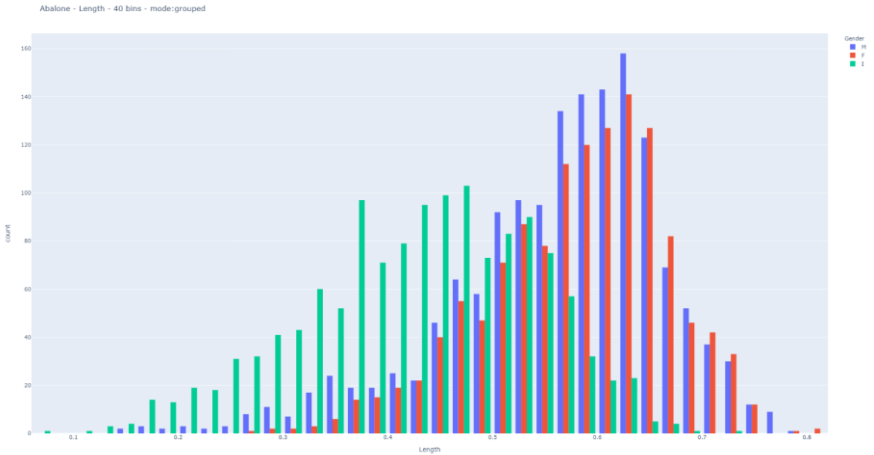
Hands-on – script : abalone01.txt - scatterplot

```
import pandas as pd
import numpy as np
import plotly.express as px
import os
#
WorkingDirCDrive='C:/temp/summerschool'
if os.path.exists( WorkingDirCDrive ):
  WorkingDir = WorkingDirCDrive
#
DiagramFileName= WorkingDir + '/diagrams/abalone-01.png'
ExcelFileName=WorkingDir + '/sampledata/ML_samples.xlsx'
print ( DiagramFileName )

df0 = pd.read_excel( ExcelFileName  , sheet_name='Abalone' ,
header=0 , engine='openpyxl'  )
print( df0.head() )

fig = px.scatter(df0, x="Length", y="ShellWeight"  ,
color="Gender", size_max=50 , title="Abalone-Length-
Shellweight")

fig.write_image( DiagramFileName , width=1980, height=1080 )

quit()
```


```
(base) C:\temp\SummerSchool\Scripts>activate summerschool

(summerschool) C:\temp\SummerSchool\Scripts>python abalone01.txt
C:/temp/summerschool/diagrams/abalone-01.png
   Length  Diameter  Height  WholeWeight  ShuckedWeight  VisceraWeight  ShellWeight  Rings Gender
0   0.455     0.365   0.095       0.5140         0.2245         0.1010        0.150     15      M
1   0.350     0.265   0.090       0.2255         0.0995         0.0485        0.070      7      M
2   0.530     0.420   0.135       0.6770         0.2565         0.1415        0.210      9      F
3   0.440     0.365   0.125       0.5160         0.2155         0.1140        0.155     10      M
4   0.330     0.255   0.080       0.2050         0.0895         0.0395        0.055      7      I

(summerschool) C:\temp\SummerSchool\Scripts>
```

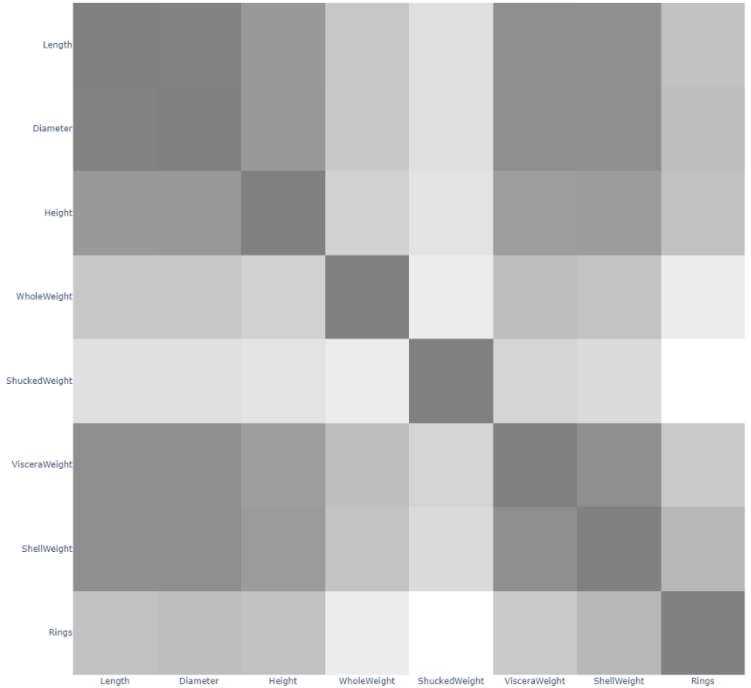
Hands-on – abalone03.txt - histogram

```
import pandas as pd
import numpy as np
import plotly.express as px
import os
#
WorkingDirCDrive='C:/temp/summerschool'
if os.path.exists( WorkingDirCDrive ):
  WorkingDir = WorkingDirCDrive
#
DiagramFileName= WorkingDir + '/diagrams/abalone-03.png'
ExcelFileName=WorkingDir + '/sampledata/ML_samples.xlsx'
print ( DiagramFileName )
#
df0 = pd.read_excel( ExcelFileName  , sheet_name='Abalone' ,
header=0 , engine='openpyxl'  )
print( df0.head() )
#
fig = px.histogram(df0, x="Length", color="Gender" ,
nbins=40 , title="Abalone - Length - 40 bins - mode:grouped"
, barmode="group" )
#
fig.write_image( DiagramFileName , width=1980, height=1080 )

quit()
```



Hands-on – abalone04.txt – pearson covariance matrix

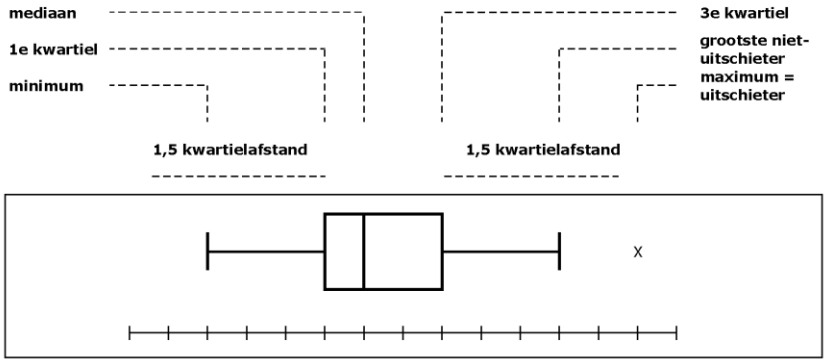
```
import pandas as pd
import numpy as np
import plotly.express as px
import os
#
WorkingDirCDrive='C:/temp/summerschool'
if os.path.exists( WorkingDirCDrive ):
  WorkingDir = WorkingDirCDrive
#
DiagramFileName= WorkingDir + '/diagrams/abalone-
correlation.png'
ExcelFileName=WorkingDir + '/sampledata/ML_samples.xlsx'
#
df0 = pd.read_excel( ExcelFileName  , sheet_name='Abalone' ,
header=0 , engine='openpyxl'  )
#
#mean01 = df0["Length"].mean()
covariancemat = df0.cov()
#
correlationmat = df0.corr(method="pearson")
print ( correlationmat )
#
fig = fig = px.imshow( correlationmat ,
color_continuous_scale=["white","gray"] )
fig.write_image( DiagramFileName , width=1980, height=1080 )
#
quit()
```



Hands-on – boxplots

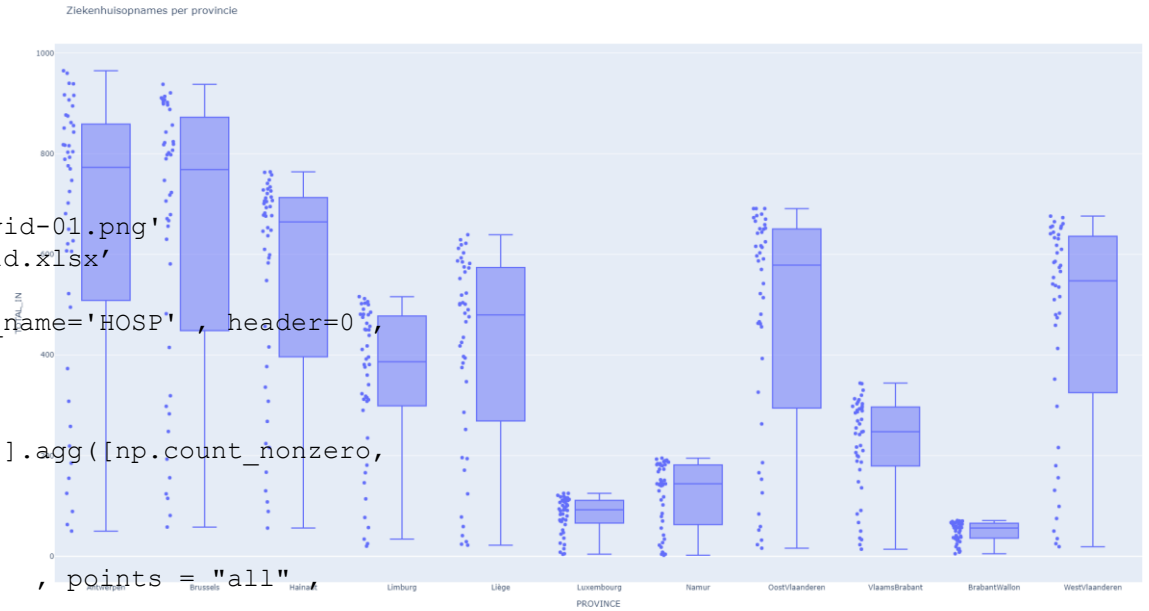
[Wikipedia] A boxplot is a standardized way of displaying the dataset based on a five-number summary:

- Minimum (Q0 or 0th percentile): the lowest data point excluding any outliers.
- Maximum (Q4 or 100th percentile): the largest data point excluding any outliers.
- Median (Q2 or 50th percentile): the middle value of the dataset.
- First quartile (Q1 or 25th percentile): also known as the lower quartile qn(0.25), is the median of the lower half of the dataset.
- Third quartile (Q3 or 75th percentile): also known as the upper quartile qn(0.75), is the median of the upper half of the dataset.



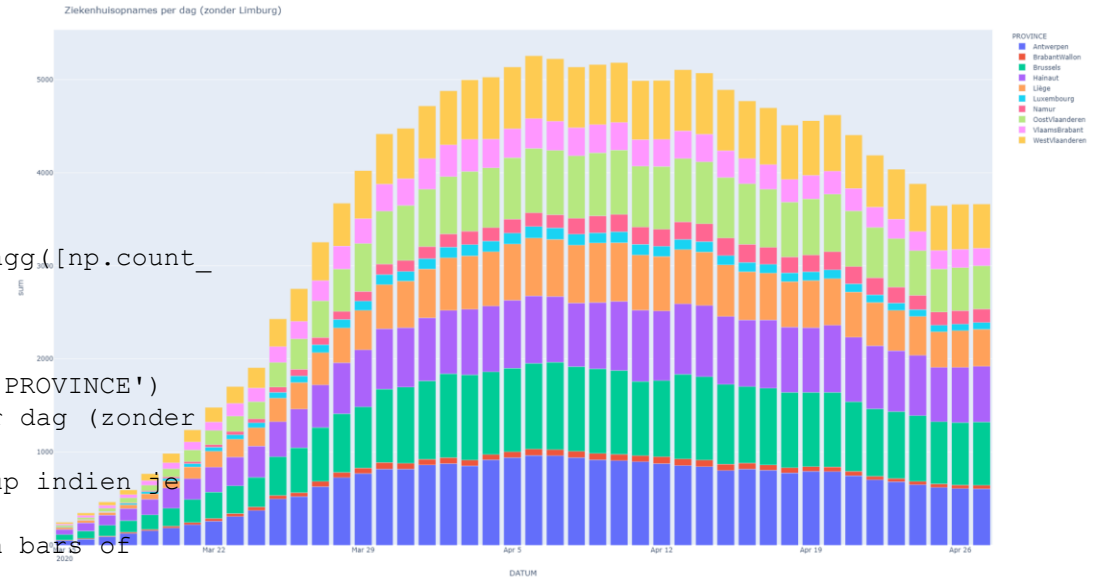
Hands-on – covid04.txt - boxplots

```
import pandas as pd
import numpy as np
import plotly.express as px
import os
#
WorkingDirCDrive='C:/temp/summerschool'
if os.path.exists( WorkingDirCDrive ):
  WorkingDir = WorkingDirCDrive
#
DiagramFileName= WorkingDir + '/diagrams/Covid-01.png'
ExcelFileName=WorkingDir + '/sampledata/covid.xlsx'
#
df0 = pd.read_excel( ExcelFileName  , sheet_name='HOSP' , header=0 ,
engine='openpyxl'  )
#
grouped =
df0.groupby(['DATUM','PROVINCE'])['TOTAL_IN'].agg([np.count_nonzero,
np.mean, np.std , np.sum])
df1 = grouped.reset_index()
# gebruik de waarde van count_nonzero
fig = px.box(df0, x="PROVINCE", y="TOTAL_IN"  , points = "all" ,
title="Ziekenhuisopnames per provincie" )
#
fig.write_image( DiagramFileName , width=1980, height=1080 )
quit()
```



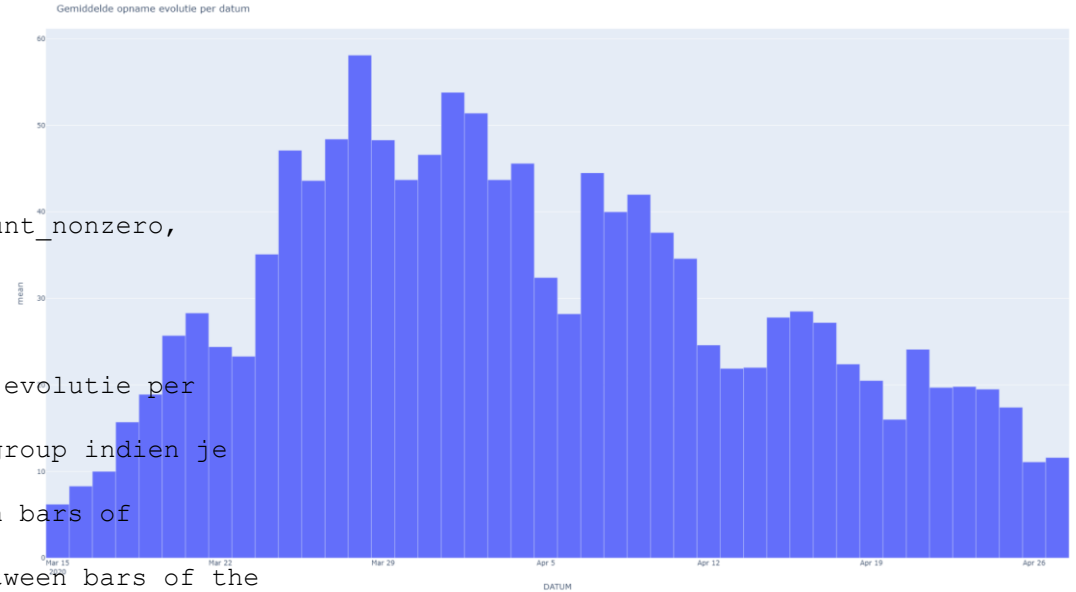
Hands-on – covid04.txt – barcharts

```
..
#
df0 = pd.read_excel( ExcelFileName  , sheet_name='HOSP' ,
header=0 , engine='openpyxl'  )
print( df0.head() )
# overschrijf de blanks
df1 = df0.replace( np.nan, '0', regex=True)
# voorbeeld van een filter
df2 = df1[ df0["PROVINCE"] != "Limburg" ]
#
grouped =
df2.groupby(['DATUM','PROVINCE'])['TOTAL_IN'].agg([np.count_
nonzero, np.mean, np.std , np.sum])
df3 = grouped.reset_index()
#
fig = px.bar( df3, y='sum', x='DATUM',  color='PROVINCE')
fig.update_layout( title='Ziekenhuisopnames per dag (zonder
Limburg)')
fig.update_layout( barmode='stack')   # or group indien je
naast elkaar wil
fig.update_layout( bargap=0.05 )  # gap between bars of
adjacent location coordinates.
fig.update_layout( bargroupgap=0.15 ) # gap between bars of
the same location coordinate
#
fig.write_image( DiagramFileName , width=1980, height=1080 )
quit()
```



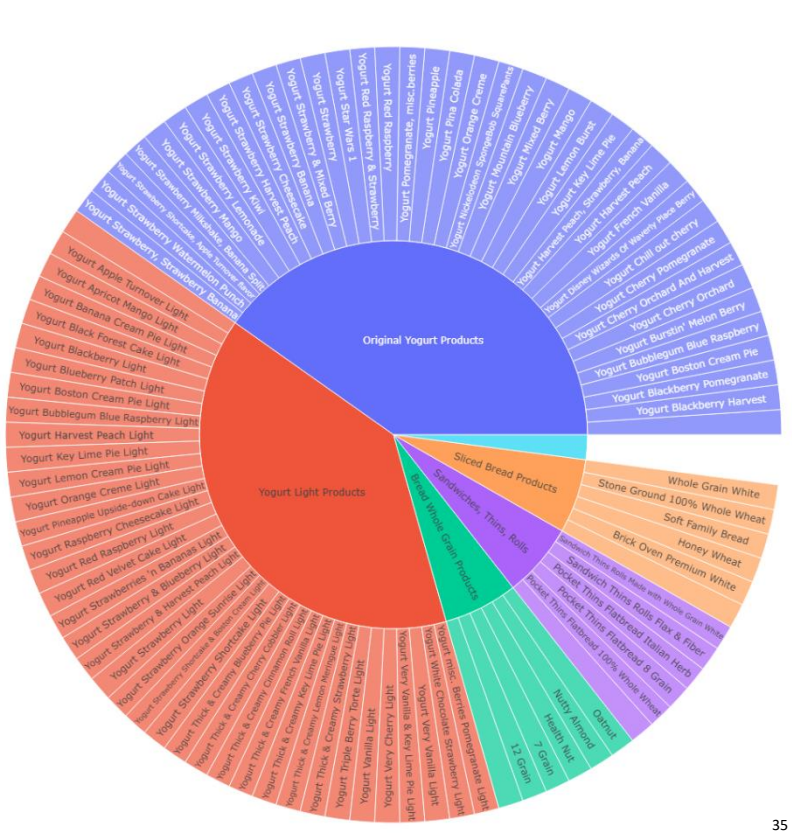
Hands-on – covid05.txt – barcharts

```
# toont de ketting van dataframes
#
df0 = pd.read_excel( ExcelFileName  , sheet_name='HOSP' ,
header=0 , engine='openpyxl'  )
# overschrijf de blanks
df1 = df0.replace( np.nan, '0', regex=True)
# voorbeeld van een filter
df2 = df1[ df0["PROVINCE"] != "Limburg" ]
#
grouped =
df2.groupby(['DATUM'])['NEW_IN'].agg([np.count_nonzero,
np.mean, np.std , np.sum])
df3 = grouped.reset_index()
#
fig = px.bar( df3, y='mean', x='DATUM' )
fig.update_layout( title='Gemiddelde opname evolutie per
datum')
fig.update_layout( barmode='stack')   # or group indien je
naast elkaar wil
fig.update_layout( bargap=0 )  # gap between bars of
adjacent location coordinates.
fig.update_layout( bargroupgap=0 ) # gap between bars of the
same location coordinate

fig.write_image( DiagramFileName , width=1980, height=1080 )

quit()
```



Hands-on – dairy02.txt - sunburst

```
#
df0 = pd.read_excel( ExcelFileName  , sheet_name='Product' ,
header=0 , engine='openpyxl'  )
#
df1 = df0[ df0["MARKET_NAME"] != "NEGEER" ]
# sunburst does not support nodes which are NULL or NaN
df2 = df1.replace( np.nan, '', regex=True)
#
grouped =
df2.groupby(['SUB_MARKET_DESCRIPTION','PRODUCT_NAME'])['PROD
UCT_ID'].agg([np.count_nonzero, np.mean, np.std , np.sum ,
np.min])
df3 = grouped.reset_index()
print ( df3 );
# indien je express ebruikt
fig = px.sunburst( df3,
path=["SUB_MARKET_DESCRIPTION","PRODUCT_NAME"],
values="count_nonzero" )
#test
fig.update_layout(  title='Yoghurt product gamma')

fig.write_image( DiagramFileName , width=1980, height=1080 )

quit()
```

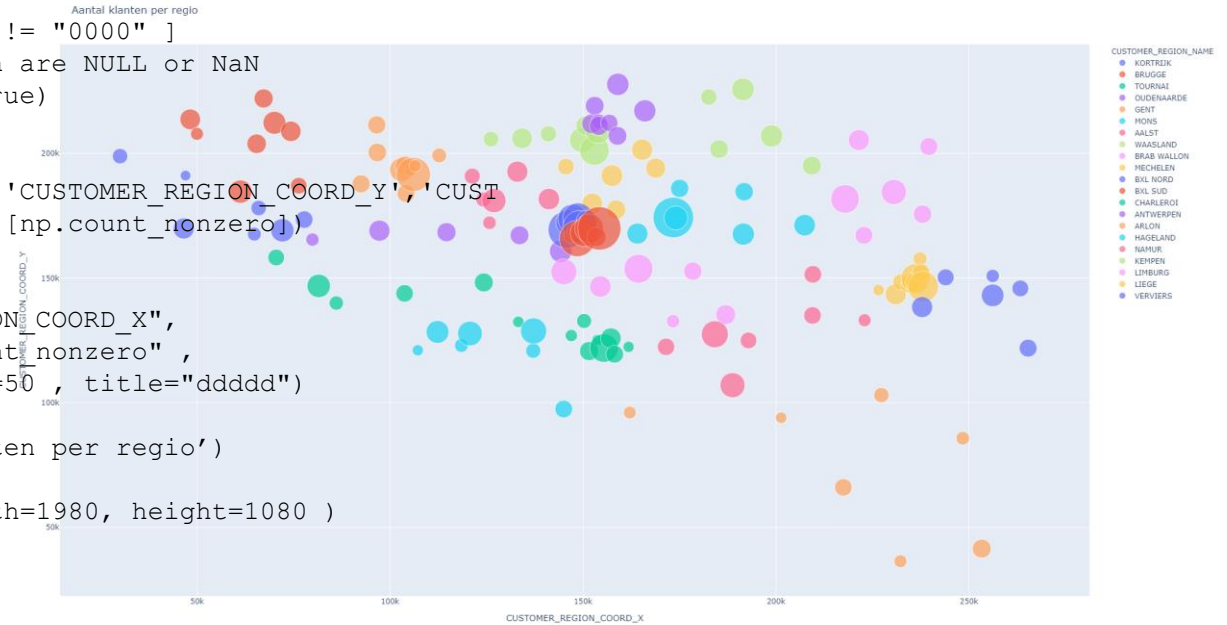


Hands-on – dairy03.txt – MERGE/JOIN/Sunburst

```
df1 = df0[ df0["MARKET_NAME"] != "NEGEER" ]
# sunburst does not support nodes which are NULL or NaN
df2 = df1.replace( np.nan, '', regex=True)
df3 = df2[["PRODUCT_ID","PRODUCT_NAME","SUB_MARKET_DESCRIPTION"]]
#
df10 = pd.read_excel( ExcelFileName  , sheet_name='Fact' , header=0 ,
engine='openpyxl'  )
df12 = df10.replace( np.nan, '', regex=True)
df14 =
df12.groupby(['PRODUCT_ID','CUSTOMER_ID'])['SALES_VALUE'].agg([np.coun
t_nonzero, np.sum ])
df15 = df14.reset_index()
#
df30 = pd.read_excel( ExcelFileName  , sheet_name='Customer' ,
header=0 , engine='openpyxl'  )
df31 = df30.replace( np.nan, '', regex=True)
df32 =
df31[["CUSTOMER_ID","CUSTOMER_REGION_NAME","CUSTOMER_REGION_COORD_X","
CUSTOMER_REGION_COORD_Y"]]
#
df20 = pd.merge( df3 , df15 , on="PRODUCT_ID" )
df21 = pd.merge( df20 , df32 , on="CUSTOMER_ID" )
#
df21.to_csv( OutputFileName, index=False)
# indien je express gebruikt
fig = px.sunburst( df21,
path=["CUSTOMER_REGION_NAME","SUB_MARKET_DESCRIPTION"], values="sum" )
```

Hands-on – dairy05.txt – scatterdiagrams

```
#
df0 = pd.read_excel( ExcelFileName  , sheet_name='Customer' , header=0
, engine='openpyxl'  )
print( df0.head() )
# alles
df1 = df0[ df0["CUSTOMER_POSTAL_CODE"] != "0000" ]
# sunburst does not support nodes which are NULL or NaN
df2 = df1.replace( np.nan, '', regex=True)
#
grouped =
df2.groupby(['CUSTOMER_REGION_COORD_X','CUSTOMER_REGION_COORD_Y','CUST
OMER_REGION_NAME'])['CUSTOMER_ID'].agg([np.count_nonzero])
df3 = grouped.reset_index()
#
fig = px.scatter(df3, x="CUSTOMER_REGION_COORD_X",
y="CUSTOMER_REGION_COORD_Y", size="count_nonzero" ,
color="CUSTOMER_REGION_NAME", size_max=50 , title="ddddd")
#test
fig.update_layout(  title='Aantal klanten per regio')

fig.write_image( DiagramFileName , width=1980, height=1080 )

quit()
```



You should be able to detect the shape of Belgium

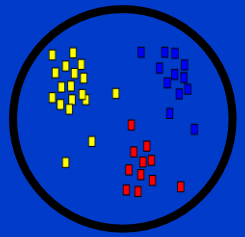
Model - Learning Approaches

Supervised Learning

Learning with a labeled training set

Task-driven

Predictive model

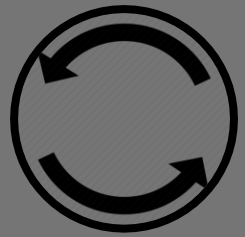


Unsupervised Learning

Discovering patterns in unlabeled data

Data-driven

Descriptive model

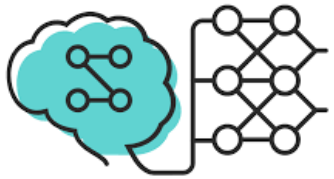


Reinforcement Learning

Learning based on feedback or reward

Reaction to an environment

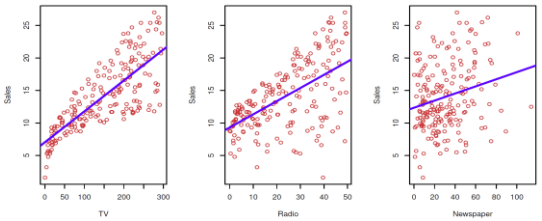
Types of learning (Supervised)

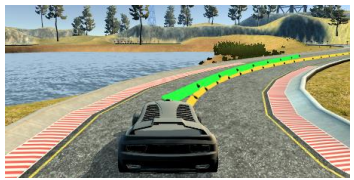


Supervised (inductive) learning. Training data includes <u>desired outputs</u>.

Regression

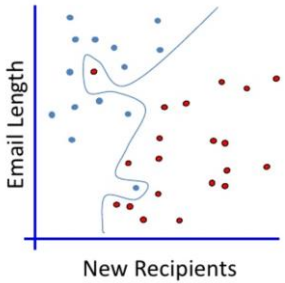
E.g. Price, blood pressure, temperature, steering wheel control, etc.



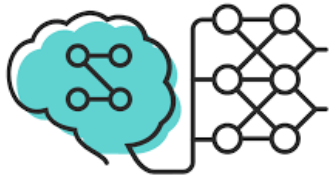


Classification problem

Which traffic sign? Spam or not?

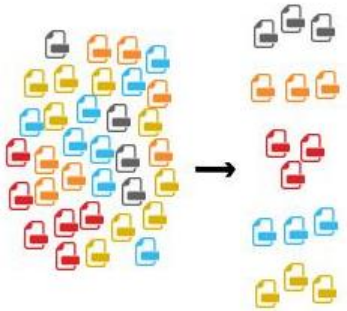


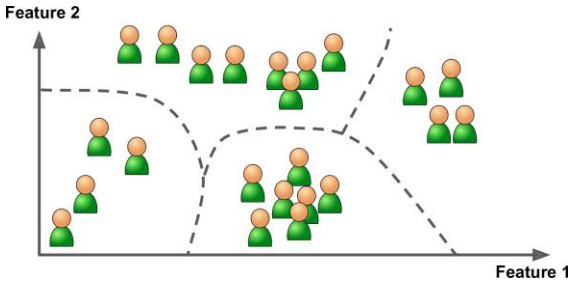
Types of learning (Unsupervised)

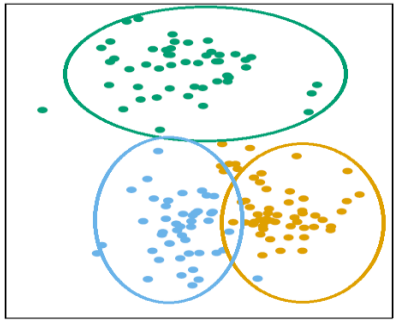


Unsupervised learning

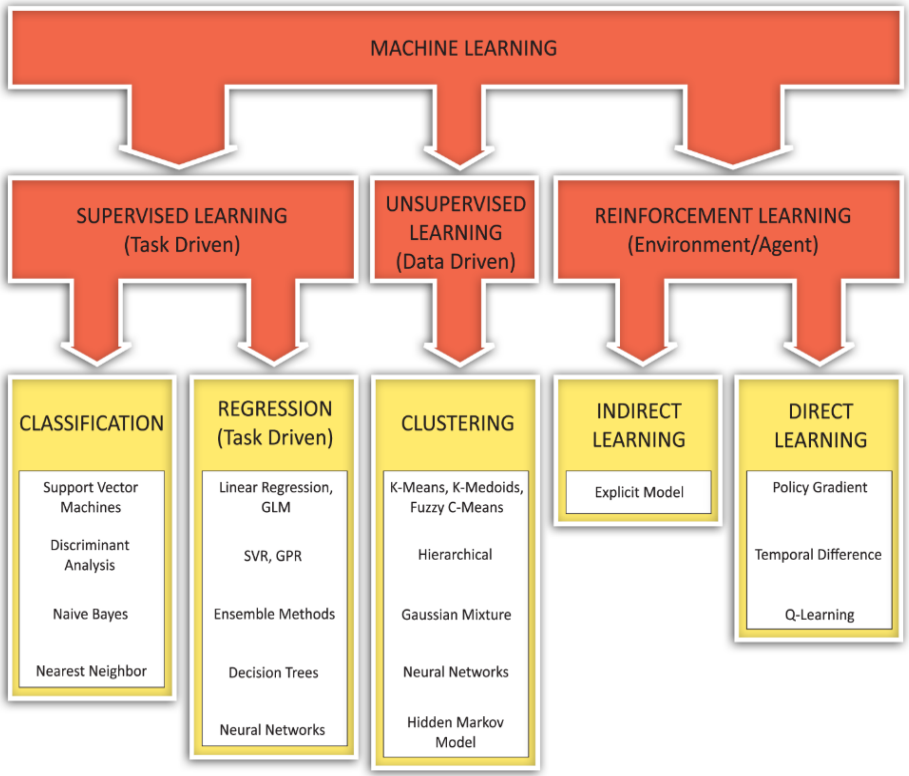
– Training data does not include desired outputs (Customer segmentation, text categorization)



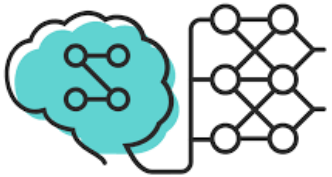




Categories

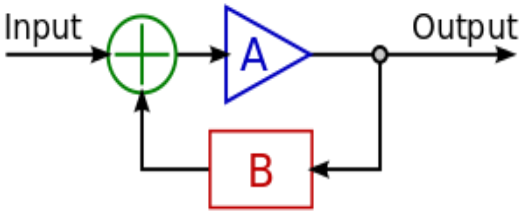

MACHINE LEARNING
SUPERVISED LEARNING (Task Driven)
UNSUPERVISED LEARNING (Data Driven)
REINFORCEMENT LEARNING (Environment/Agent)
CLASSIFICATION
Support Vector Machines
Discriminant Analysis
Naive Bayes
Nearest Neighbor
REGRESSION (Task Driven)
Linear Regression, GLM
SVR, GPR
Ensemble Methods
Decision Trees
Neural Networks
CLUSTERING
K-Means, K-Medoids, Fuzzy C-Means
Hierarchical
Gaussian Mixture
Neural Networks
Hidden Markov Model
INDIRECT LEARNING
Explicit Model
DIRECT LEARNING
Policy Gradient
Temporal Difference
Q-Learning

Three Components of Learning Algorithms

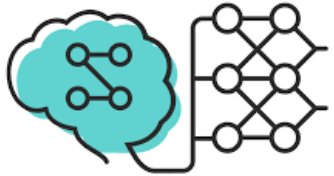


LEARNING = REPRESENTATION + EVALUATION + OPTIMIZATION

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		



Model (Representation)



*A **model** is an abstraction representing and simplifying reality, allowing us to solve real-life problems.*

Building a Machine Learning model requires 3 major activities

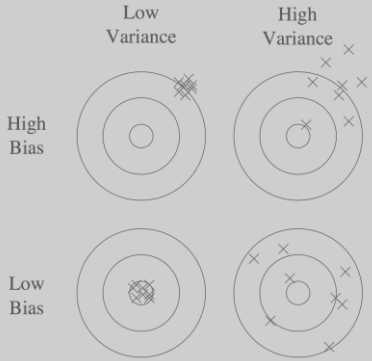
- Modeling - “a model must be represented in some formal language that the computer can handle”
- Evaluation - “an evaluation function is needed to distinguish good classifiers from bad ones”
- Optimization - “we need a method to search among the classifiers in the language for the highest-scoring one”

Evaluation (Accuracy, errors, loss)

Evaluation is essentially how you judge or prefer one model vs. another;

- how do we make sure and are getting better with each step, and not worse?
- The answer lies in our "measurement of wrongness".
- The wrongness measure is known as the **loss (cost) function.**

Types of overfitting



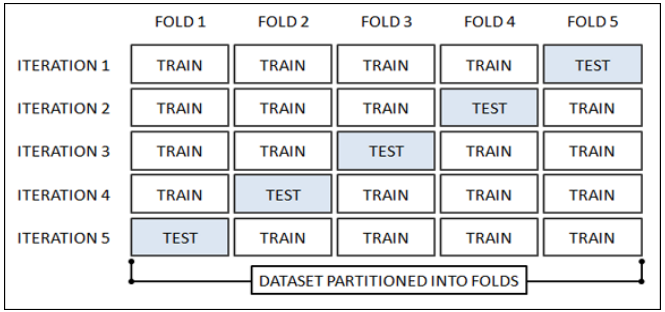
Bias is the tendency to consistently learn the same wrong thing

Variance is the tendency to learn random things irrespective of the real signal

Evaluation – Data sets

Training – testing – evaluating data

Rule : 70% - 20% - 10%



It is more important to determine the error on the test data set than the error on the training set.

The error on the test set shows how the model reacts to new or previously not submitted data.

ARF File

```
@RELATION abalone

@ATTRIBUTE Length          NUMERIC
@ATTRIBUTE Diameter        NUMERIC
@ATTRIBUTE Height          NUMERIC
@ATTRIBUTE Whole weight                NUMERIC
@ATTRIBUTE Shucked weight NUMERIC
@ATTRIBUTE Viscera weight NUMERIC
@ATTRIBUTE Shell weight    NUMERIC
@ATTRIBUTE Rings           NUMERIC
@ATTRIBUTE Gender          {M,F,I}

@DATA
0.455,0.365,0.095,0.514,0.2245,0.101,0.15,15,M
0.35,0.265,0.09,0.2255,0.0995,0.0485,0.07,7,M
0.53,0.42,0.135,0.677,0.2565,0.1415,0.21,9,F
0.44,0.365,0.125,0.516,0.2155,0.114,0.155,10,M
0.33,0.255,0.08,0.205,0.0895,0.0395,0.055,7,I
0.425,0.3,0.095,0.3515,0.141,0.0775,0.12,8,I
0.53,0.415,0.15,0.7775,0.237,0.1415,0.33,20,F
0.545,0.425,0.125,0.768,0.294,0.1495,0.26,16,F
0.475,0.37,0.125,0.5095,0.2165,0.1125,0.165,9,M
0.55,0.44,0.15,0.8945,0.3145,0.151,0.32,19,F
< abridged for legibility purposes>
```

Evaluation - Metrics

Regression

Mean Square Error

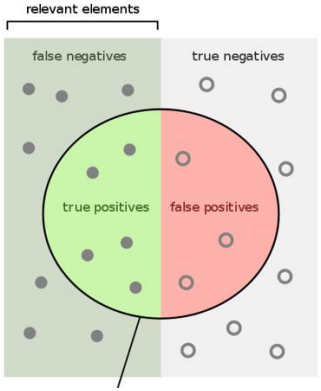
$$MSE = \frac{1}{N}\sum_{i=0}^{N-1}(y_i - predicted(y_i))^2$$

Mean Absolute Error

$$MAE = \frac{1}{N}\sum_{i=0}^{N-1}|y_i - predicted(y_i)|$$

See handouts for more details

Classifiers



True positive (TP) is a positive sample which is correctly classified

True negative: (TN) a negative sample correctly classified

False positive (FP). A negative sample classified as positive (aka. Type 1) - inconvenient

False negative (FN) a positive sample classified as negative. (aka. Type 2) - unwanted

Evaluation - Metrics

Confusion matrix

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**

Accuracy – Precision – Sensitivity

$$Accuracy = \frac{Number\ of\ correctly\ classified\ samples}{Total\ number\ of\ samples}$$

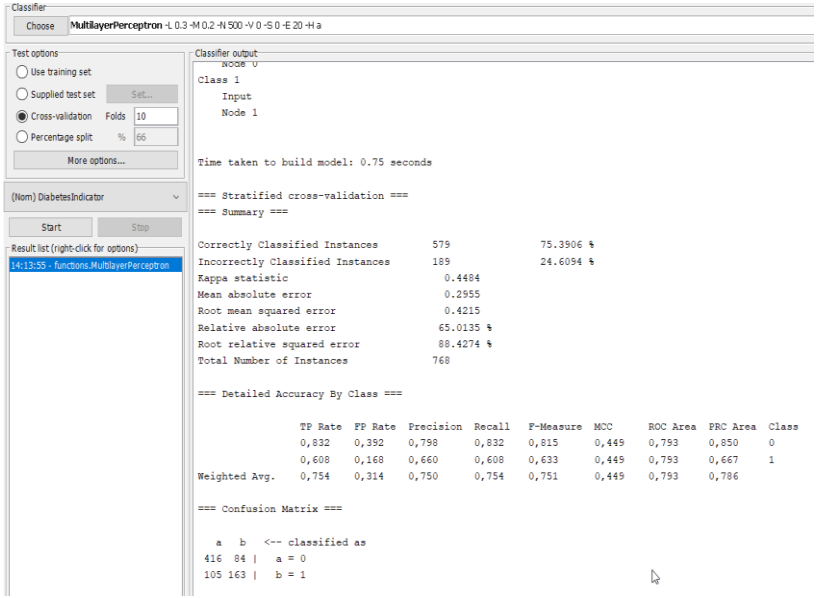
$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F\_Measure = \frac{Precison\ .\ Recall}{Precison + Recall}$$

Evaluation - Global Classification report

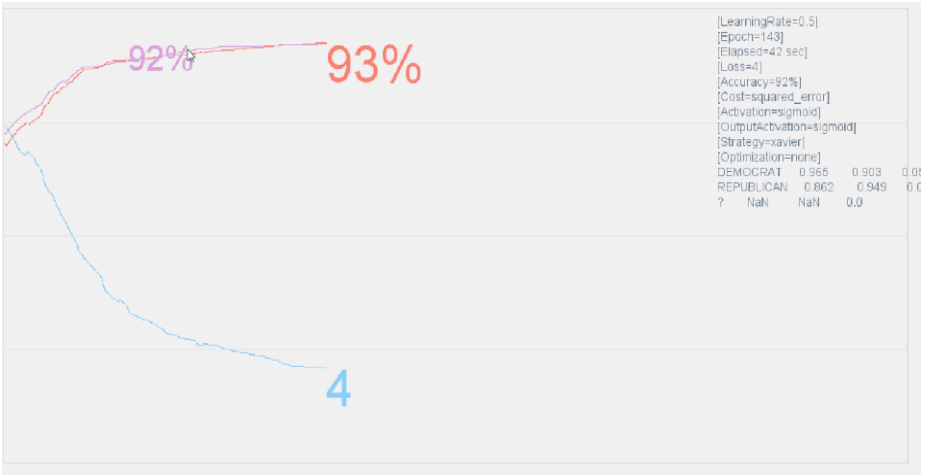
WEKA

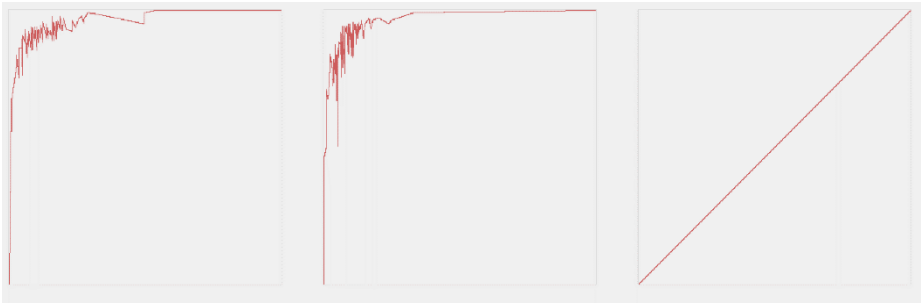


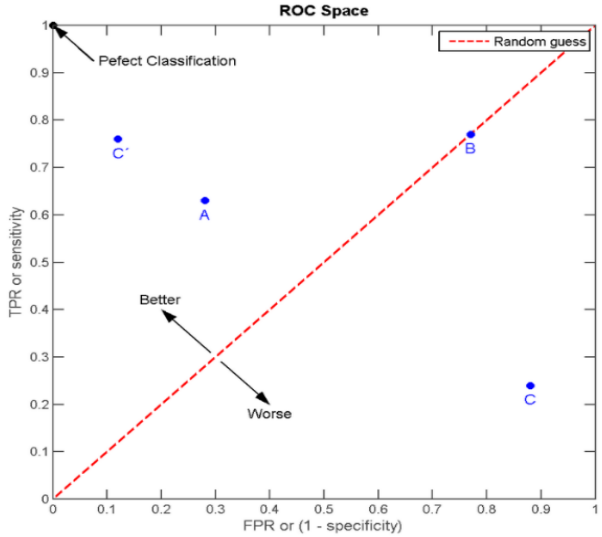
Home brewn

```
- <SMOModelGeneral>
    <ModelApplicationDomain>General</ModelApplicationDomain>
    <ARFFFileName>c:\temp\cbrTekStraktor\Tutorial\MachineLearning\SandBox\SanFranciscoHousing.arff</ARFFFileName>
    <KernelTrickType>GAUSSIAN</KernelTrickType>
    <NumberOfFeatures>8</NumberOfFeatures>
    <MaxCycles>40</MaxCycles>
    <Tolerance>0.001</Tolerance>
    <SoftMargin>0.6</SoftMargin>
    <NumberOfSupportVectors>226</NumberOfSupportVectors>
  </SMOModelGeneral>
- <SMOModelEvaluation>
  - <TrainedModel>
      <TrainedModelAcurracy>0.8691588785046729</TrainedModelAcurracy>
      <TrainedModelEntries>428</TrainedModelEntries>
    - <![CDATA[

                      0        1 ---> Actual
                ---------- --------- ---------
          0     [      186][       9]
          1     [       47][      186]

                 Precision  Sensitivity   FMeasure
                ---------- ---------- ---------
          0     [ 0,953846][ 0,798283][ 0,869159]
          1     [ 0,798283][ 0,953846][ 0,869159]
      ]]>
    </TrainedModel>
  - <TestedModel>
      <TestedModelAcurracy>0.78125</TestedModelAcurracy>
      <TestedModelEntries>64</TestedModelEntries>
      <TestSetPercentage>13</TestSetPercentage>
    - <![CDATA[

                      0        1 ---> Actual
                ---------- --------- ---------
          0     [       23][       2]
          1     [       12][       27]

                 Precision  Sensitivity   FMeasure
                ---------- ---------- ---------
          0     [ 0,920000][ 0,657143][ 0,766667]
          1     [ 0,692308][ 0,931034][ 0,794118]
      ]]>
    </TestedModel>
```

Evaluation – Accuracy progress and ROC



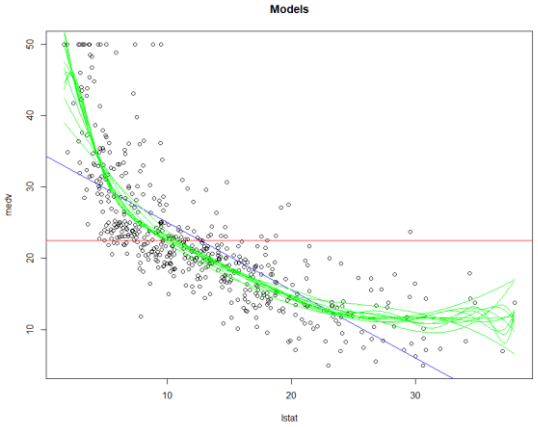


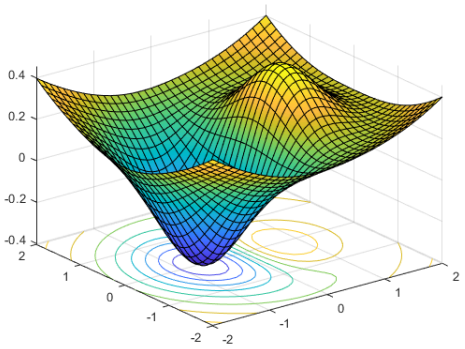


- The X axis is the False Positive Rate: FP / (FP + TN) (or specificity)
- The Y axis is the True Positive Rate: TP / (TP + FN) (the recall/sensitivity)

Optimization

Optimization is how you search the space of ***represented*** models to obtain better ***evaluations***. This is the way you expect to traverse the landscape to find the most optimal model.





Optimization

Loss function

The Loss function defines the difference of the results of a model and the expected or real value.

A ML algorithm aims to find a “local minimum”.

Jump of faith - Loss functions are chosen in such a manner that they have **a convex** shape and are **differentiable**.

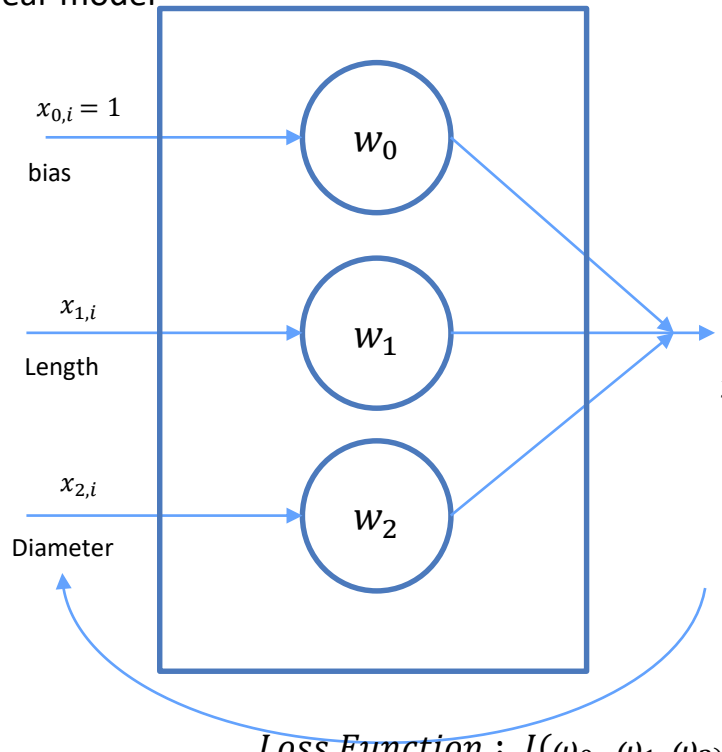
Mean Squared Error (regression – neural networks)

Cross Entropy (logistic regression)

Optimization – gradient descent

How does it fit together? (Continuous target value prediction)

e.g. linear model



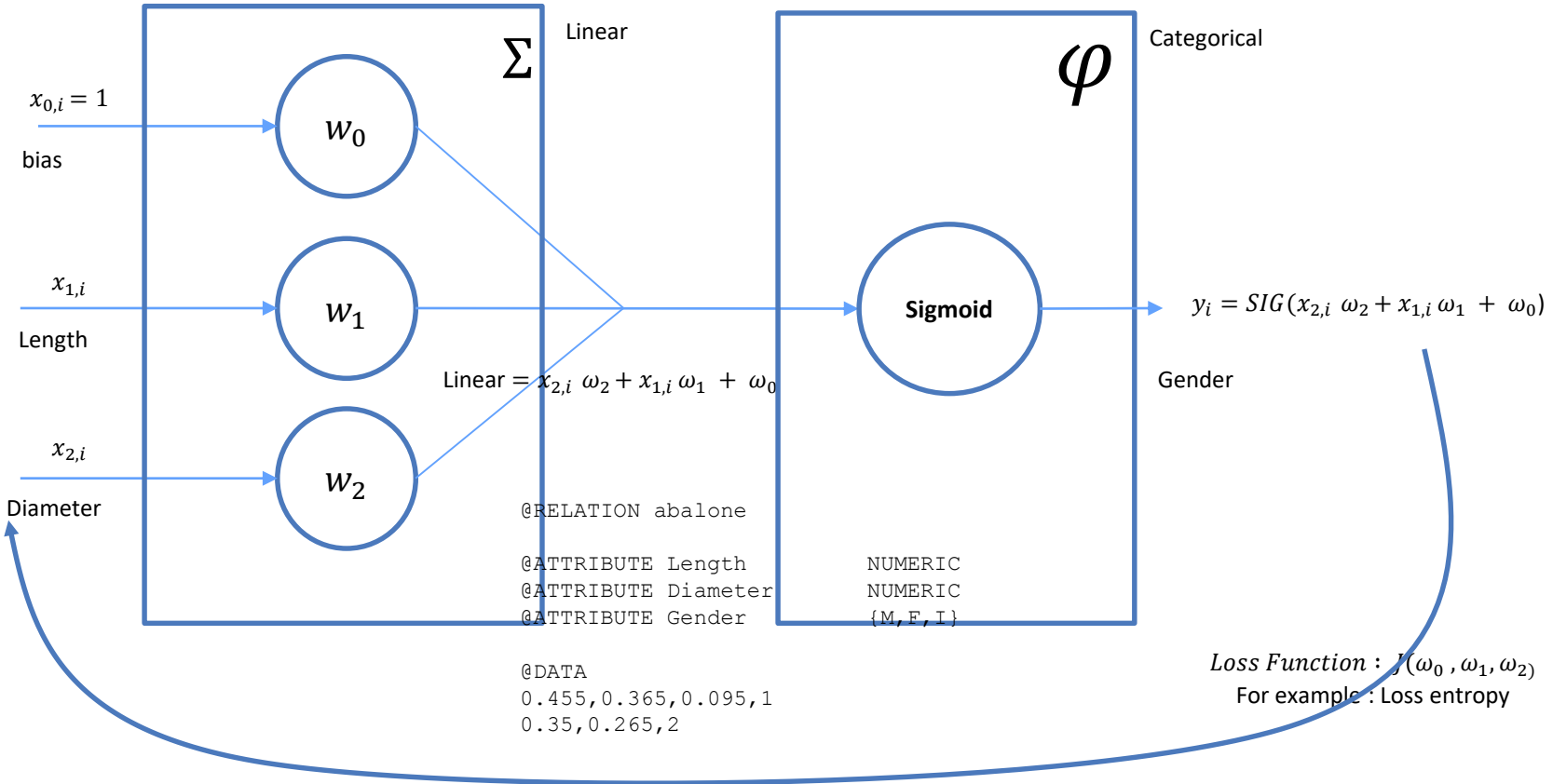
$$y_i = x_{2,i}\,\omega_2 + x_{1,i}\,\omega_1 + \omega_0$$

```
@RELATION abalone

@ATTRIBUTE Length          NUMERIC
@ATTRIBUTE Diameter        NUMERIC
@ATTRIBUTE Age             NUMERIC

@DATA
0.455,0.365,0.095,1
0.35,0.265,2
0.53,0.42,4
0.44,0.3655M
0.33,0.2556I
0.425,0.3,7
0.53,0.415,1
0.545,0.425,2
0.475,0.37,3
0.55,0.44,4
0.525,0.38,2
```

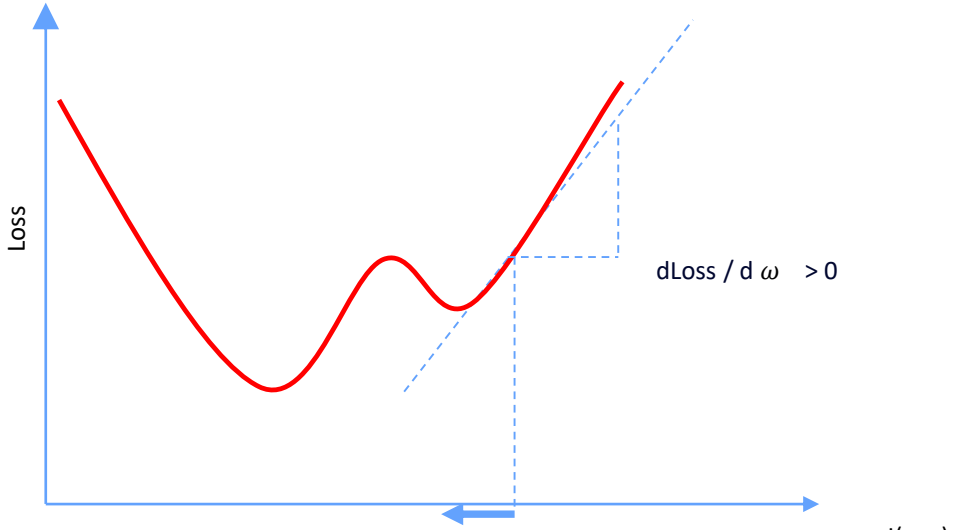
$$Loss\ Function: J(\omega_0, \omega_1, \omega_2) = \frac{1}{2N}\sum_{i=1}^{N}\left(y_i - \left(x_{2,i}\,\omega_2 + x_{1,i}\,\omega_1 + \omega_0\right)\right)^2$$

Optimization – gradient descent

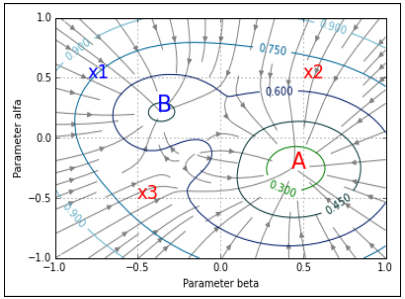
How does it fit together? (Categorical target value prediction)



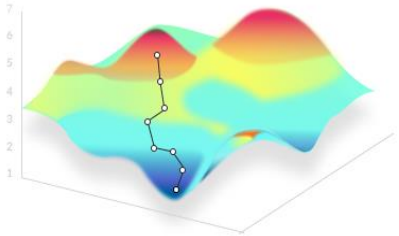
Optimization – gradient descent



Gradient descent in neural networks



Direction = - dL/ dw
Stepsize : k or alpha



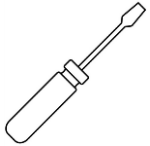
Tools

Open Source

IBM Open Source

Watson Knowledge Studio

Tools



Client tools

- WEKA
- Java and libraries (e.g. libSVM.jar)
- Python and its numerous libraries: scikit, pandas, seaborn, numPhy, etc.
- **Jupyter** notebook
- *Watson Studio & ML*

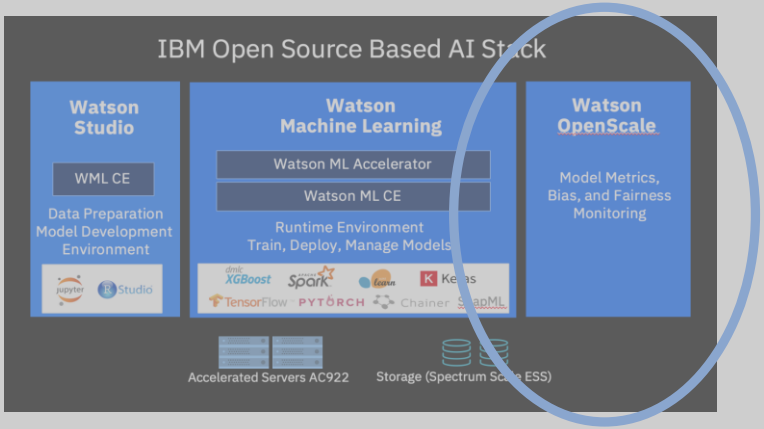
Platforms (distributions)

- Keras
- Google Tensorflow

*”TensorFlow is quickly becoming a viable option for companies interested in deploying deep learning for tasks ranging from computer vision, to speech recognition, to text analytics,” said **Rajat Monga, engineering leader for TensorFlow**. “IBM’s enterprise offering of TensorFlow will help organizations deploy this framework — we’re glad to see this support.”*

Tools – IBM AI stack & Power AI

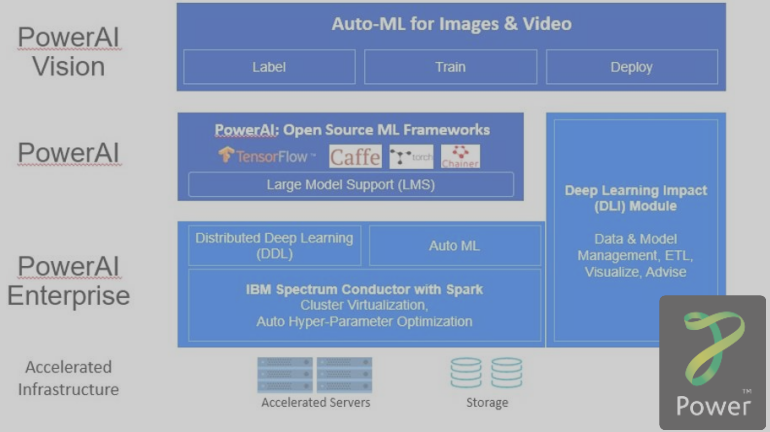
Open source AI Stack



[NYT 01 June 2018] Google, hoping to head off a rebellion by employees upset that the technology they were working on could be used for lethal purposes, will not renew a contract with the Pentagon for artificial intelligence work when a current deal expires next year.

[Datanews 10 April 2019] Bedrijven zoals Barco, IBM en Microsoft in België gaan de volgende maanden de ethische richtlijnen die de Europese Commissie wil opleggen rond artificiële intelligentie (AI) uittesten op hun praktische haalbaarheid.

Power AI

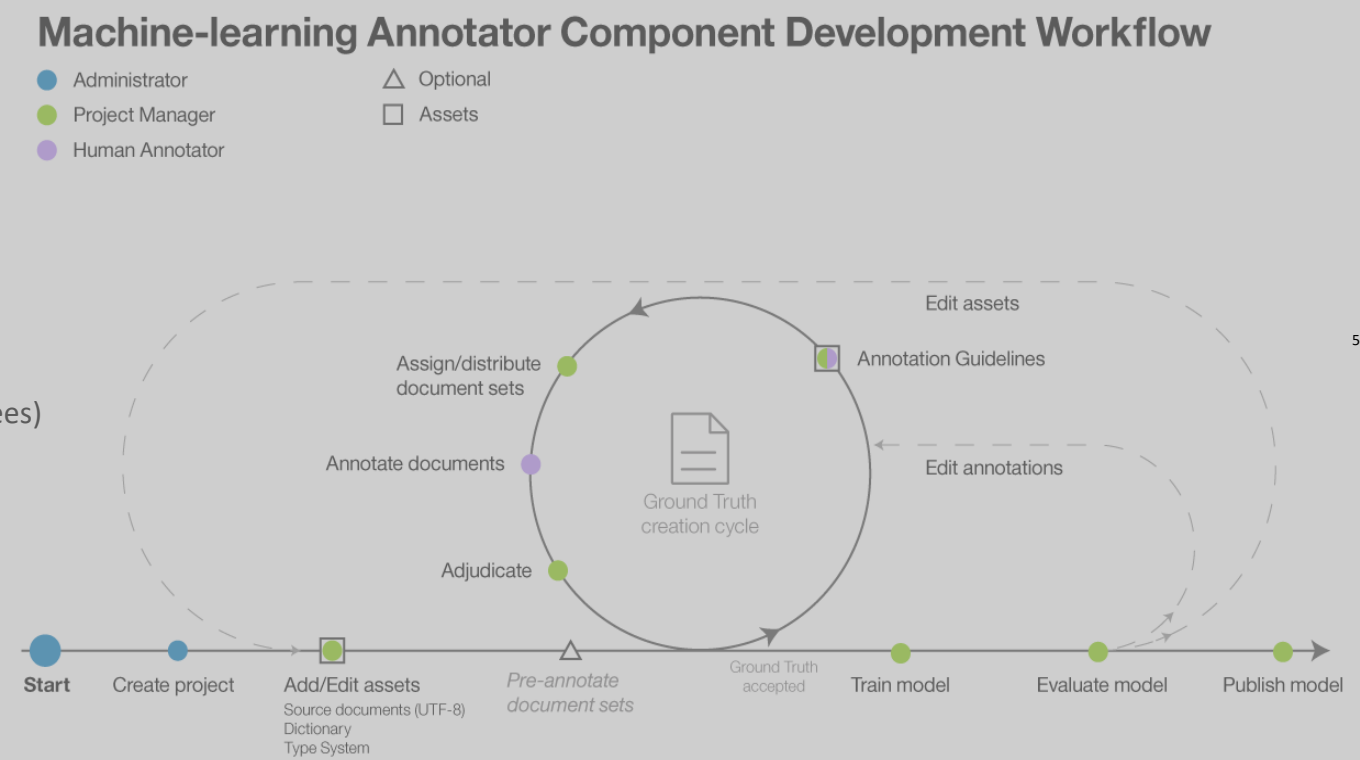


PowerAI is a distribution for popular open source Machine Learning and Deep Learning frameworks on the Power8 architecture

Tools – Watson Knowledge Studio

ML Components used

- Natural Language processing
- Machine Learning (Decision trees)
- N-grams



Clustering

K-means

Hierarchical K-Means

DBSCAN

What is Data Clustering

Clustering or Data Clustering is a way to organize data into groups based on the existing undiscovered natural patterns in a data set

It is an unsupervised (no dependent variable) data mining methodology.

The goals are to find similarities within groups and maximize differences between groups.

It is most beneficial in a multi-dimensional space, typically 5 to 10 dimensions that can be used to describe the data where visual perception fails.

When would you use clustering

Finding similarities in marketing segments – Determining patterns in customer cohort groups to maximize marketing efforts.

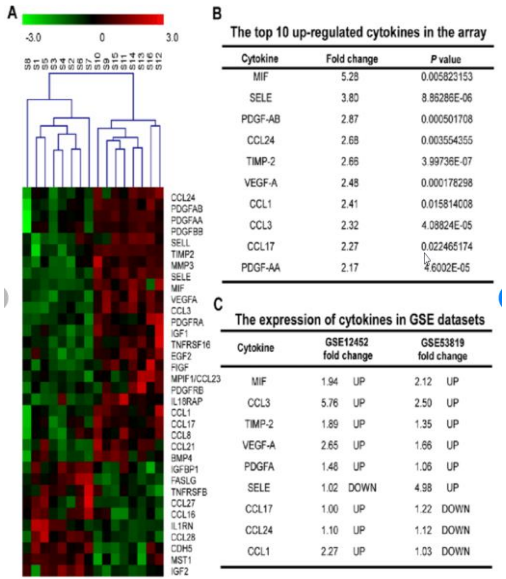
Product groupings – using product features and sales patterns to segment for store positioning or online sales recommendations.

Image segmentation - for example using RGB Coordinates to cluster together tokens with high similarity in the feature space.

Recommender Systems – finds recommendations for like minded entities and returns recommendations based on those similarities.

Missing values analysis and handling – mean and mode are used from resulting member cluster.

Detect Outliers – measures the deviation of a record from its peer group and if it exceeds thresholds is considered an outlier.



Cytokine : a small protein produced by cells in the nervous and immune systems that affects what happens between cells.

K-means

Step 1. Specify K number of clusters.

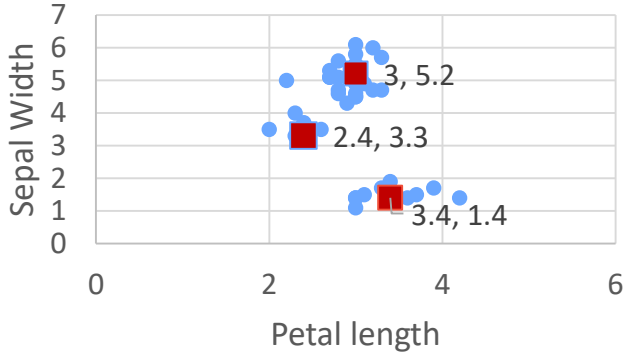
Step 2. Initial centers (centroids) are selected at random.

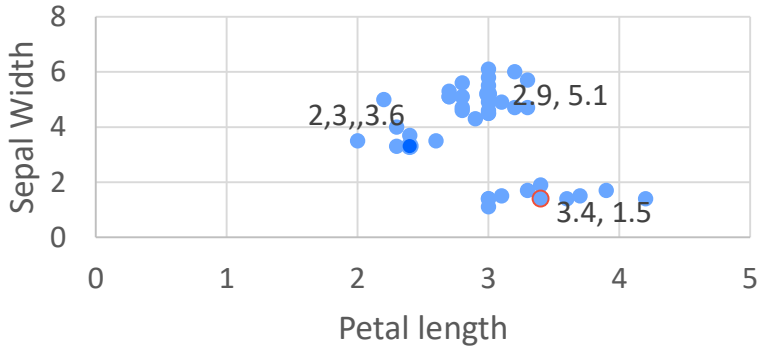
Step 3. For each sample (point) calculate the distance to the K centroids and assign a sample to the cluster of which the distance is the smallest.

Step 4. Centroids are recalculated (find the point in a set which is nearest to all other) once all points have been assigned.

Step 5. Re-estimate the distance of each point to each centroid and move a point to the cluster which is nearest.

Step 6. If there are no points being transferred to another cluster stop, else goto step 4





Euclidian distance

Measuring the distance between Customers A and B

Customer	Age	Weight	Height	Sex
A	29	120	66	0
B	54	220	72	1

Step 1: Subtracts each of the coordinate values (i.e. Age of Customer A – Age of Customer B and Weight of Customer A and Weight of Customer B etc…

Age	Weight	Height	Sex
-25	-100	-6	-1

Step 2: Square the differences to get the absolute difference

Age	Weight	Height	Sex
625	10000	36	1

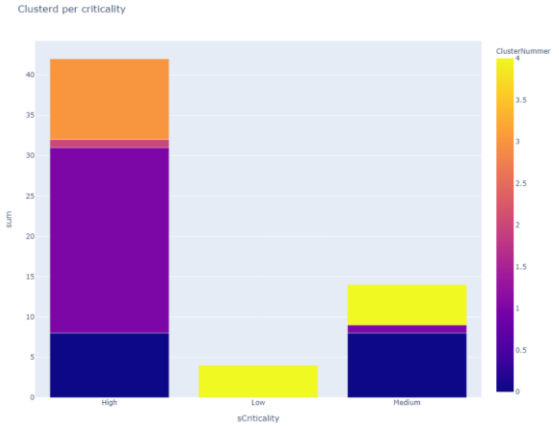
Step 3: Sum the squares

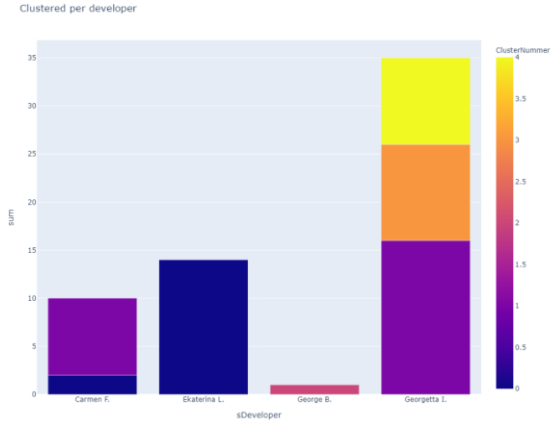
Age	Weight	Height	Sex	Sum of Squares
625	10000	36	1	10662

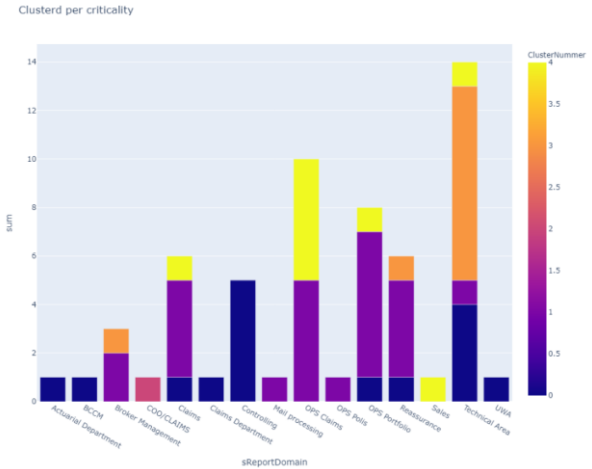
Step 4: Obtain the Square Root to get your distance metric

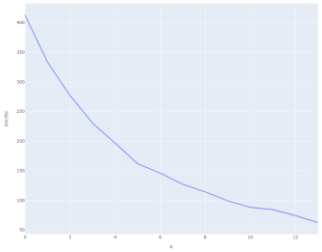
103.26

Testimonial : Development effort estimation







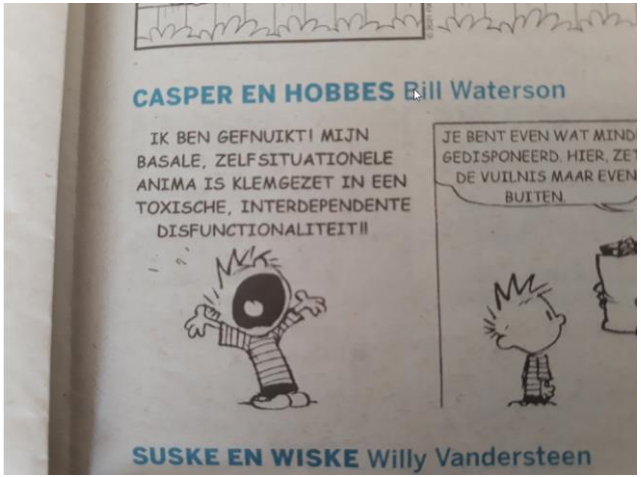


OID	Report	ReportDomain	Criticality	Type	Developer	DataVaultCoverage	NbrOfEndTables	NbrOfKPIs	NbrOfUnassignedTables	NbrOfDWHGTables	NbrOfStagingT	NbrOfDerive	NbrOfComposite	NbrOfReport
R0006		Claims	High	Extract	Carmen F.	0.55	2	16	0	40	42	9	15	0
R0007		Claims	High	Extract	Carmen F.	0.567567568	1	14	0	37	39	7	13	0
R0008		OPS Portfolio	High	Report	Carmen F.	0.534883721	2	7	1	43	46	8	25	2
R0011		OPS Portfolio	High	Report	Carmen F.	0.534883721	1	8	1	43	46	8	24	2
R0012		OPS Portfolio	High	Report	Carmen F.	0.542857143	1	9	0	35	37	5	19	2
R0015		OPS Portfolio	High	Report	Carmen F.	0.534883721	2	7	1	43	46	8	25	2
R0016		OPS Portfolio	High	Report	Carmen F.	0.542857143	1	15	0	35	37	5	19	2
R0009		Reassurance	High	Extract	Carmen F.	0.531914894	3	15	2	47	50	9	24	3
R0141		Actuarial Departme	High	Extract	Carmen F.	0.555555556	1	0	0	45	48	9	21	3

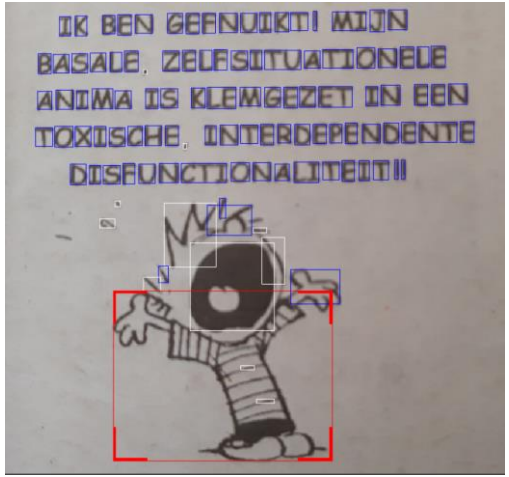
Hands-on : Yohurt Sample Data
Clustering of customers on geolocation

script : k-means.txt

Hands-on : Casper clustering



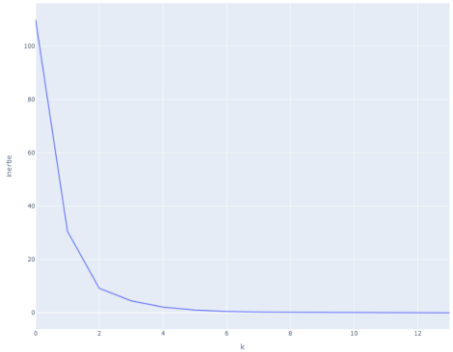
First step
connected components

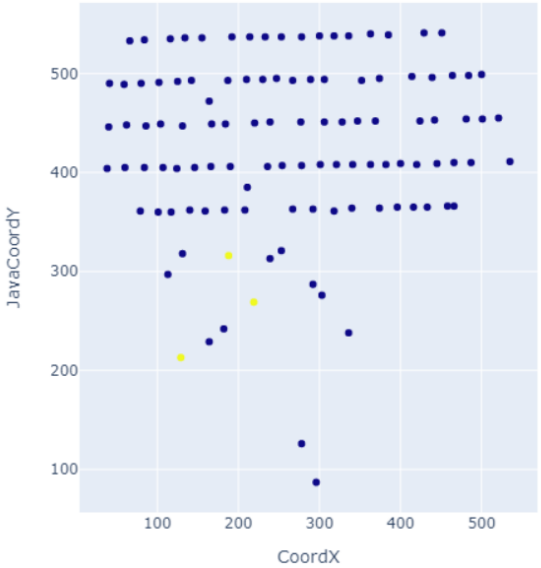


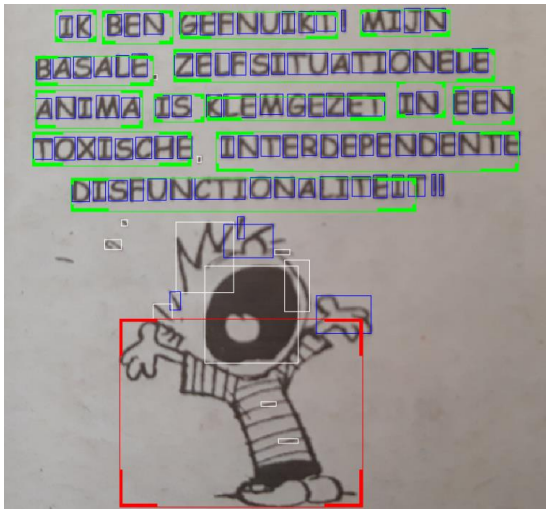
UID	ParentUID	CoordX	CoordY	Width	Height	Content	Tipe
5277563950925	-1	429	13	17	25	?	ConCom
5277563950926	-1	451	13	21	23	?	ConCom
5277563950927	-1	363	14	4	23	?	ConCom
5277563950928	111	385	15	39	22	mijn	ConCom
5277563950929	-1	300	16	13	23	?	ConCom
5277563950930	-1	318	16	16	22	?	ConCom
5277563950931	-1	336	16	20	21	?	ConCom
5277563950932	-1	192	17	18	23	?	ConCom
5277563950933	110	214	17	15	24	g	ConCom
5277563950934	110	233	17	15	23	e	ConCom
5277563950935	110	253	17	21	22	f	ConCom
5277563950936	110	278	17	19	22	nuikt	ConCom
5277563950937	109	134	18	16	24	b	ConCom
5277563950938	109	155	18	20	22	e	ConCom
5277563950939	109	116	19	15	22	n	ConCom
5277563950940	108	84	20	14	23	i	ConCom
5277563950941	108	66	21	13	22	k	ConCom
5277563950942	-1	500	55	17	25	?	ConCom
5277563950943	113	464	56	16	25	z	ConCom

Hands-on : Casper clustering

The major features of the Connected Components have been copied in tabsheet “casper” the ML_Samples.xlsx
Connected Components have a Tipe = “ConCom” (.. filter)
By clustering on the Height you will find the characters, use K = 2
Script : casper01.txt

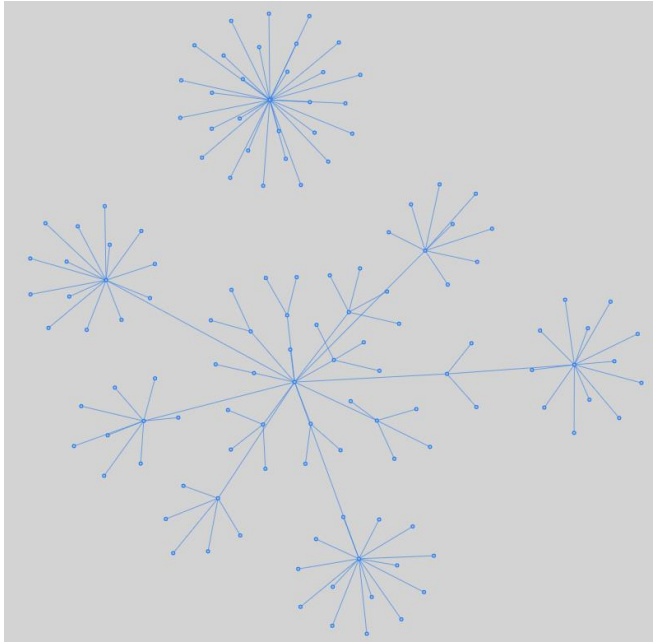






Hands-on : Casper Interactive dependency graph

Script : casper03.txt demonstrates how to use pyvis
Just click on the HTML file casper03.html

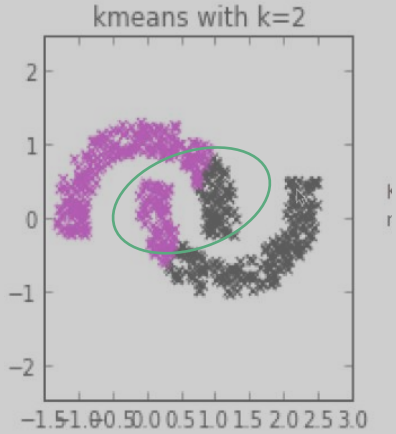


Other clustering algorithms

Hierachical K-Means

Determine K-Means cluster but always for K=2. Then pick the one with the smallest distances and continue.

Density Based Scan



Based on the idea of a minimum number of data points a cluster must have within a certain area, i.e. density.

Regression

Concepts

Lineair Regression

Ordinary Least Square Regression

Regression

Regression models are used for prediction. There are basically two types of regression

- Quantitative (linear or continuous). Real values are predicted (or extrapolated). Examples are the statistical formulas for first and second-degree regression fitting.
- Qualitative (discrete or nominative). Nominative of classes are predicted. This type of regression is a **classifier**.

Remind me to talk about augmented vectors/matrices

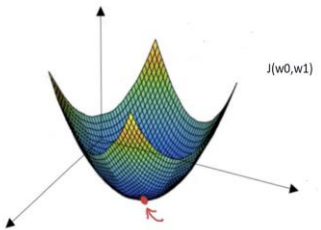
Linear regression

Easy as rain (*at least when we stick to first degree regression curves*)

Rationale : When you grasp the linear regression concept you understand most of the other ML algorithms.

- Objective is to find the weight factors in the regression equation $y = w_1 x_1 + w_0 x_0 + \xi$
- We use the Least Mean Square as a Loss function $J(w_1, w_0) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - (w_1 x_{i1} + w_0))^2$
- LMS is 2nd power has a local minimum (convex), so we can perform a gradient descent
- The partial derivatives against **w0** and **w1**

» $\begin{cases} \frac{\partial J}{\partial w_0} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_1 x_{1i} + w_0))x^0 \\ \frac{\partial J}{\partial w_1} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_1 x_{1i} + w_0))\, x_{1i}{}^1 \end{cases}$



Linear regression

The iterative nature is expressed as follows

$$\begin{cases} w_{0_{k+1}} \leftarrow w_{0_k} - \alpha \frac{1}{N} \sum_{i=1}^{N} (w_1 x_{1i} + w_0 - y_i) \\ w_{1_{k+1}} \leftarrow w_{1_k} - \alpha \frac{1}{N} \sum_{i=1}^{N} (w_1 x_{1i} + w_0 - y_i)\, x_{1i} \end{cases}$$

This is the commonly accepted notation.

$$\overline{w_{k+1}} = \overline{w_k} - \alpha \sum \nabla J$$

=> In a nutshell : a gradient descent (with stepsize alpha) on a loss function encompassing an activation function

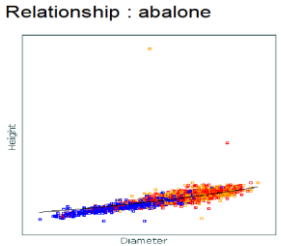
Linear regression

Initialize the regression weights

Perform the following steps for a predefined number of cycles

- For each sample
  - Calculate the result of H using the current weights (one per variable – degree)
  - Calculate the gradient
- Aggregate the individual gradients and divide by N (number of samples)
- Update each weight (using the gradient and step size)
- Calculate the L2 Loss (so you can assess the convergence, i.e. verify whether the delta drops below a predefined threshold).

```
for(int cycle=0;cycle<NbrOfCycles;cycle++)
{
	// Gradient
	double[] gradient = new double[ weights.length ];
	for( int i=0;i<gradient.length;i++) gradient[i]=0;
	for(int i=0;i<data.getNbrOfRows();i++)
	{
		double y = data.getValues()[i][1];
		//  CURVE => y = w2.(x2^2)  + w1.(x1^1) + w0.(x0^0)  etc
		double yi = 0; // the calculated value of y using the current
weights
		for(int j=0;j<weights.length;j++)
		{
			yi += xn[i][j] * weights[j];
		}
		// calculate the gradient    (y - calculatedY).x power
		for(int j=0;j<weights.length;j++)
		{
			gradient[j] += (y - yi) * xn[i][j];    // postpone 1/N
		}

	}
	for( int i=0;i<gradient.length;i++) gradient[i]=gradient[i] /
data.getNbrOfRows();
	// update weights
	for( int i=0;i<gradient.length;i++)
	{
		weights[i] = weights[i] + (step * gradient[i] );
	}
	// stukken weggelaten die de LMS berekenen
}
```



Degree : 3
Max Number of Cycles : 10000
Mean Squared Error : 2.8955372316731 21E-4
Regression fitted curve : 0,06886 (x^3) + 0,11760 (x^2) + 0,16551 x + 0,04613

In all fairness, Excel does the job faster and more accurate

Ordinary Least Square (OLS) algorithm

Another approach to determine a regression equation is via matrix calculation. See your notes for the maths.

The Ordinary Least Square is not an iterative ML algorithm per se; it is a **formula** that uses training data.

$$W = (X^T X)^{-1} X^T Y$$

Caution - Only works if $(X^T X)^{-1}$ is an invertable matrix.

Ordinary Least Square – hands-on

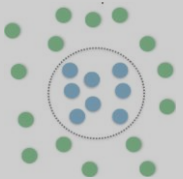
Classifiers

Decision Trees

K Nearest Neigbors

Naieve Bayes

Decision trees



The ID3 algorithm is a multi-category classifier it relies on splitting data along boundaries that produce subsets resulting in the highest information gain.

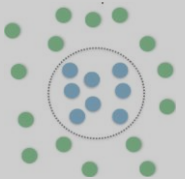
Information Entropy $entropy\ (S) = -\sum_{c=1}^{K} p_c\ LN(\ p_c)$ *(Entropy 0 has the highest information, pc is portion)*

Information gain $Gain(\ S_v\ , V) =\ entropy(S) - \sum(\frac{|S_v|}{|S|}\ \ entropy(S_v))$

Where

- S is the overall sample
- S_v is a subset of sample S. it contains all records which have a target category v
- |S| is the number of elements in the overall sample
- $|S_v|$ is the number of elements in subset S_v

Decision trees



Calculate Entropy for the sample

For each potential split in the sample

- Calculate Entropy in each potential bin
- Find the net entropy for your split
- Calculate entropy gain

Select the split with the highest entropy gain

Recursively (or iteratively in some cases) perform the partition on each split until a termination criteria is met

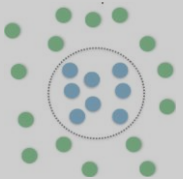
- Terminate once you reach a specified number of bins

Terminate once entropy gain falls below a certain threshold

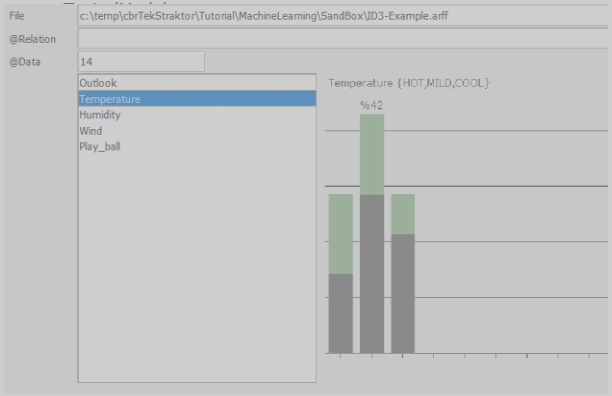
```
@RELATION   ID3_Example

@ATTRIBUTE  Outlook          {Sunny,Overcast,Rain}
@ATTRIBUTE  Temperature      {Hot,Mild,Cool}
@ATTRIBUTE  Humidity         {High,Normal}
@ATTRIBUTE  Wind             {Weak,Strong}
@ATTRIBUTE  Play_ball        {Yes,No}

@DATA
Sunny,Hot,High,Weak,No
Sunny,Hot,High,Strong,No
Overcast,Hot,High,Weak,Yes
Rain,Mild,High,Weak,Yes
Rain,Cool,Normal,Weak,Yes
Rain,Cool,Normal,Strong,No
Overcast,Cool,Normal,Strong,Yes
Sunny,Mild,High,Weak,No
Sunny,Cool,Normal,Weak,Yes
Rain,Mild,Normal,Weak,Yes
Sunny,Mild,Normal,Strong,Yes
Overcast,Mild,High,Strong,Yes
Overcast,Hot,Normal,Weak,Yes
Rain,Mild,High,Strong,No
```

80

Decision trees



```
- <DecisionTreeModelEvaluation>
  - <TrainedModel>
      <TrainedModelAcurracy>1.0</TrainedModelAcurracy>
      <TrainedModelEntries>12</TrainedModelEntries>
    - <![CDATA[

                        YES         NO ---> Actual
                    ----------  ----------  ----------
            YES     [       9][       0]
            NO      [       0][       3]

                     Precision  Sensitivity    FMeasure
                    ----------  ----------  ----------
            YES     [ 1,000000][  1,000000][  1,000000]
            NO      [ 1,000000][  1,000000][  1,000000]
    ]]>
  </TrainedModel>
```



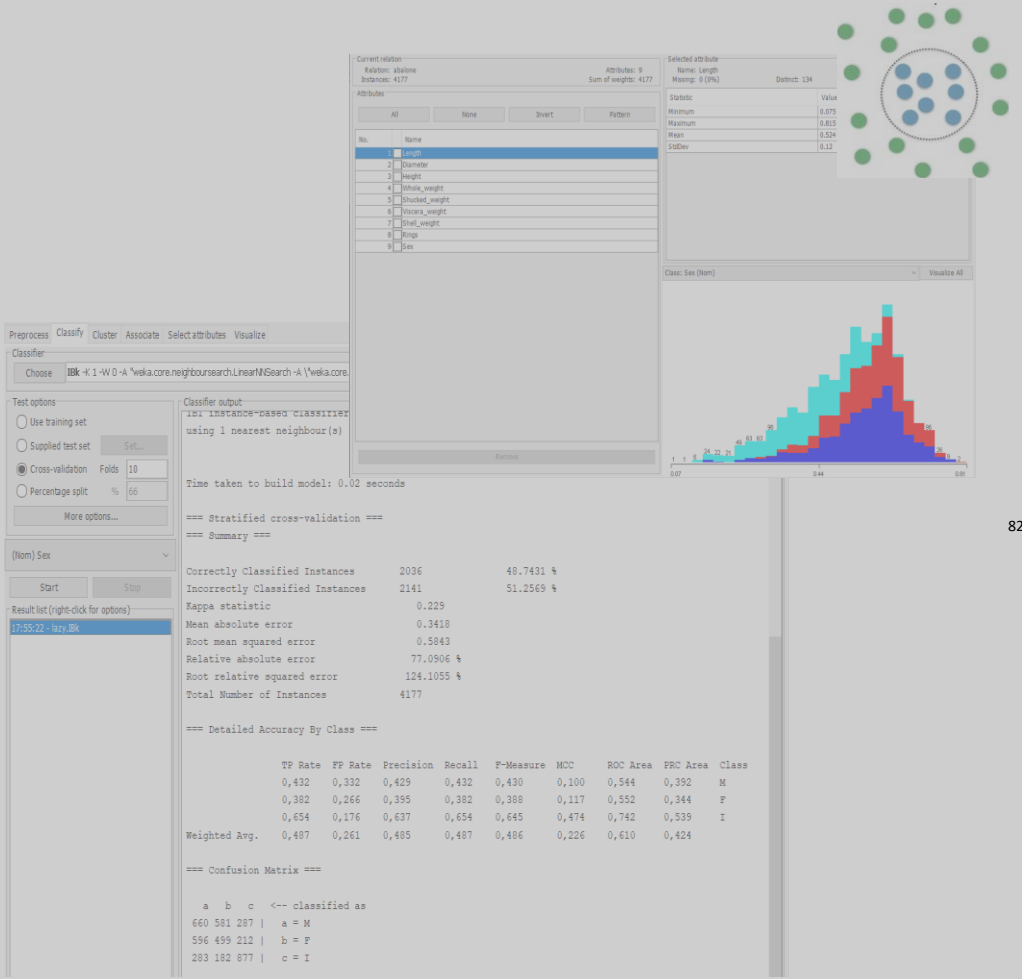

```
- <DecisionTreeModel>
  - <DecisionTreeModelGeneral>
      <ModelApplicationDomain>General</ModelApplicationDomain>
      <NumberOfNodes>9</NumberOfNodes>
      <NumberOfLevels>4</NumberOfLevels>
      <NumberOfCategories>5</NumberOfCategories>
      <NumberOfClasses>2</NumberOfClasses>
      <NumberOfCoinFlips>0</NumberOfCoinFlips>
      <Classes>[YES][NO]</Classes>
    </DecisionTreeModelGeneral>
  + <DecisionTreeModelEvaluation>
  - <DecisionTreeModelDisplay>
    - <![CDATA[

        .[UID=0 Val=HIGH PUID=-1 #=12 Humidity UNKNOWN]
        .+--[UID=1 Val=SUNNY PUID=0 #=6 Outlook LEFT]
        .   +--[LEAF=NO UID=3 PUID=1 #=2 LEFT]
        .   +--[UID=4 Val=WEAK PUID=1 #=4 Wind RIGHT]
        .      +--[LEAF=YES UID=5 PUID=4 #=2 LEFT]
        .      +--[UID=6 Val=OVERCAST PUID=4 #=2 Outlook RIGHT]
        .         +--[LEAF=YES UID=7 PUID=6 #=1 LEFT]
        .         +--[LEAF=NO UID=8 PUID=6 #=1 RIGHT]
        .+--[LEAF=YES UID=2 PUID=0 #=6 RIGHT]
    ]]>
  </DecisionTreeModelDisplay>
```


81

K Nearest Neighbors

A surprisingly easy algorithm for multi-category classification

- Step 1. Initialize the data samples. It is **crucial** to normalize all features
- Step 2. Initialize the label or target set, i.e. the label related to each sample.
- Step 3. Pick a single sample and determine the distance to all other samples (using Euclidian distance)
- Step 4. Determine the K samples which have the smallest distance to the sample (these are the K Nearest Neighbors).
- Step 5. Perform a **vote**, i.e. determine the class which occurs most frequently in the K Nearest Neighbors. This is the predicted class for the sample.
- Step 6. Add the predicted class to the confusion matrix and go to step 2.

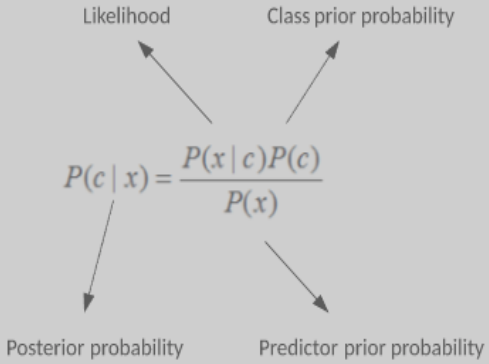


$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)}$$

Naive Bayes

Multi-category classifier

Based on conditional probabilities theorem




```
@RELATION  Howest-Bayes-Voorbeeld

@ATTRIBUTE weather {Sunny,Snowing,Windy}
@ATTRIBUTE Temperature {Cold,Freezing}
@ATTRIBUTE Ski  {Yes,No}

@DATA
Sunny,Cold,Yes
Snowing,Cold,Yes
Sunny,Freezing,Yes
Windy,Cold,No
Snowing,Cold,No
Sunny,Cold,Yes
Windy,Cold,No
Snowing,Cold,Yes
Snowing,Cold,Yes
Windy,Freezing,No
Sunny,Cold,No
Snowing,Cold,Yes
Windy,Freezing,No
Windy,Cold,Yes
Snowing,Cold,Yes
```

P(Ski | Snow and Freezing) = (P(Snow|Ski).P(Freeze|Ski)) . P(Ski) / P(Snow).P(Freeze)

83

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)}$$

Naive Bayes

```
- <BayesModelEvaluation>
  - <TrainedModel>
      <TrainedModelAcurracy>0.8461538461538461</TrainedModelAcurracy>
      <TrainedModelEntries>13</TrainedModelEntries>
    - <![CDATA[

                    YES         NO ---> Actual
                ----------  ----------  ----------
     YES        [        8][         1]
     NO         [        1][         3]

                 Precision  Sensitivity   FMeasure
                ----------  ----------  ----------
     YES        [  0,888889][  0,888889][  0,888889]
     NO         [  0,750000][  0,750000][  0,750000]
      ]]>
    </TrainedModel>
  - <TestedModel>
      <TestedModelAcurracy>0.5</TestedModelAcurracy>
      <TestedModelEntries>2</TestedModelEntries>
      <TestSetPercentage>13</TestSetPercentage>
    - <![CDATA[

                    YES         NO ---> Actual
                ----------  ----------  ----------
     YES        [        0][         1]
     NO         [        0][         1]

                 Precision  Sensitivity   FMeasure
                ----------  ----------  ----------
     YES        [  0,000000][       NaN][       NaN]
     NO         [  1,000000][  0,500000][  0,666667]
      ]]>
    </TestedModel>
  </BayesModelEvaluation>
```

```
- <BayesModelGeneral>
    <ModelApplicationDomain>General</ModelApplicationDomain>
    <NumberOfEntries>13</NumberOfEntries>
    <NumberOfBins>16</NumberOfBins>
    <NumberOfModels>3</NumberOfModels>
    <NumberOfClasses>2</NumberOfClasses>
    <Classes>[YES][NO]</Classes>
    <ClassProbabilities>[0.7142857142857143][0.35714285714285715]</ClassProbabilities>
  </BayesModelGeneral>
- <BayesModelEvaluation>
  + <TrainedModel>
    + <![CDATA[]]>
  + <TestedModel>
    + <![CDATA[]]>
  </BayesModelEvaluation>
- <Model>
    <ModelIndex>0</ModelIndex>
    <CategoryName>weather</CategoryName>
    <CategoryType>NOMINAL</CategoryType>
    <NominalValues>[SUNNY][SNOWING][WINDY]</NominalValues>
    <BinSegmentBegins>[0.0][0.0][0.0][0.0][0.0][0.0][0.0][0.0][0.0][0.0][0.0][0.0][0.0][0.0][0.0][0.0]</Bin
    <ClassCounts>[9][4]</ClassCounts>
  - <BABinStats>
    + <BACountsPerBin>
    - <BAProbabilitiesPerBin>
        <BAProbabilityPerBin>0.23076923076923078</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.46153846153846156</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.3076923076923077</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
        <BAProbabilityPerBin>0.0</BAProbabilityPerBin>
      </BAProbabilitiesPerBin>
    </BABinStats>
```

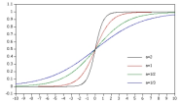
84

Classifiers

Logistic Regression

Stochastic Gradient Descent

Support Vector Machines & SMO



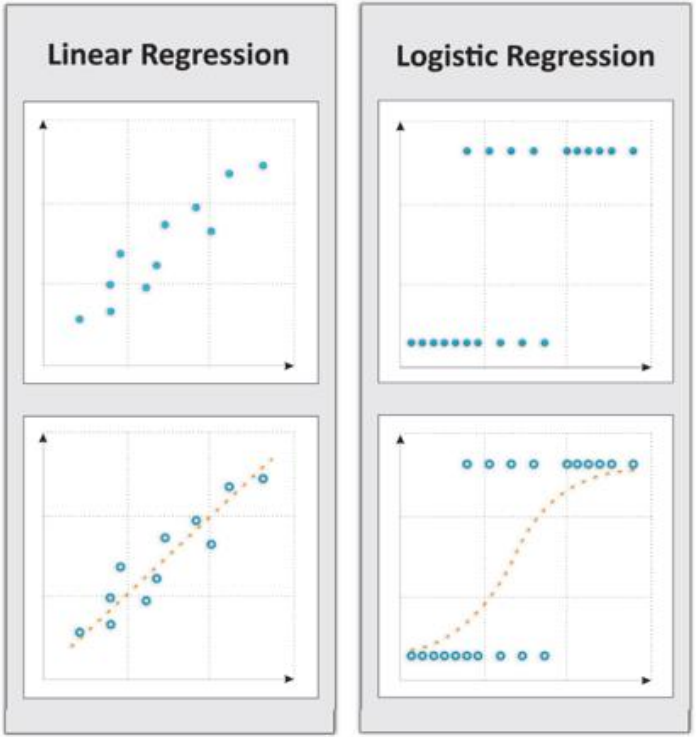
Logistic regression

Classification algorithm

Works on dichotomous (binary) result categories between 0 and 1 (*) (and data must be linearly separable)

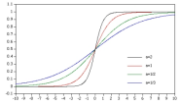
It is based on maximizing the likelihood that a results belongs to either 0 or 1 *(see top right diagram).*

Obviously such a curve is not linear. So it is about finding the weights of linear regression equation that in their turn determine the steepness of an S-shaped curve (called a Sigmoid)



**See your handouts for the full story*

**Softmax algorithm is for proportional values between 0 an 1*



Logistic regression

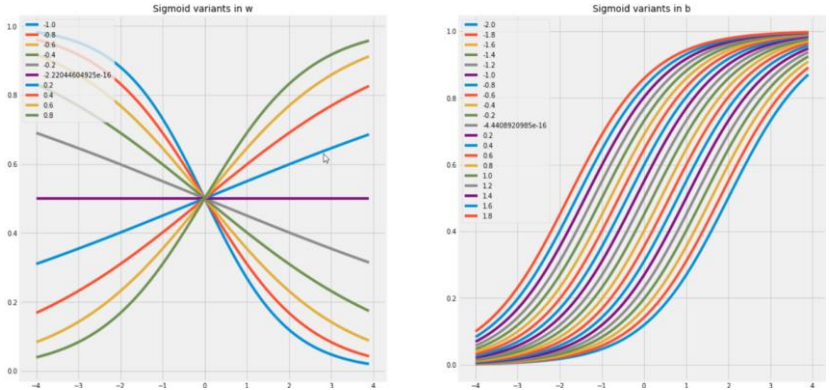
Activation function

We use the LOGIT (logaritmic on an odd) function to define the function to be maximized.

$$z_i = \text{LOGIT} = ln\left(\frac{\mathrm{p}(\bar{x}_i)}{1-\mathrm{p}(\bar{x}_i)}\right)$$

The inverse LOGIT is the Sigmoid

$$LOGIT^{-1} = \frac{1}{1+e^{-z_i}}$$



Sigmoid zero is always 0.5

Loss function

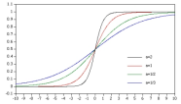
Entropy Loss function is used *(clever guys once determined that this function delivers a maximum when applied to a sigmoid hypothesis function)*

P(C=y|x) = $H(x)^t(1-H(x))^{1-t}$

Where

- If y=0 then $H(x)^0(1-H(x))^1$ or $1-H(x)$
- If y=1 then $H(x)^1(1-H(x))^0$ or $H(x)$

The above is multiplicative, so problematic to optimize. However the same wise guys cracked this and rewrote it to minimization problem.



Logistic regression

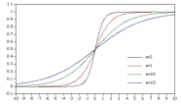
The function for which the minimum must be found is (without going into any detail)

Bear in mind that we use a simple linear equation to drive the sigmoid (delta)

$$E(w) = -\sum_{m=1}^{m} y_m \ln(\sigma(w_1 x_{1m} + w_0) + (1 - y_m)\ln(1 - (\sigma(w_1 x_{1m} + w_0))$$

This results in the following partial derivatives which we will use for the gradient descent

$$\begin{cases} \dfrac{\delta E(W)}{\delta w_1} = \displaystyle\sum_{m=1}^{M} (\sigma(x_m) - y_m)\, x_m^1 \\ \dfrac{\delta E(W)}{\delta w_0} = \displaystyle\sum_{m=1}^{M} (\sigma(x_m) - y_m)\, x_m^0 \end{cases}$$

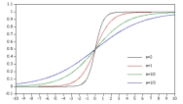


Logistic regression

Eventually one arrives to this definition for the iterative Logistic Regression

$$\bar{W} \dashleftarrow \bar{W} - \alpha \nabla \sigma(\bar{W}^T \bar{X})$$

- W is the weight vector (we might need to transpose it so that the matrix multiplications work out alright)
- X are vectors of with the input variables (of N dimensionality or columns and there are M samples or rows)
- $\bar{W}^T \bar{X}$ is our linear equation, our actual model.
- $\sigma(\bar{W}^T \bar{X})$ is the sigmoid of our model
- $\nabla\sigma(\bar{W}^T \bar{X})$ is the gradient, i.e. the partial derivative of the difference between the actual categories (Y) and the result of the sigmoid function.$\alpha \nabla\sigma(\bar{W}^T \bar{X})$
- $\bar{W} + \alpha \nabla\sigma(\bar{W}^T \bar{X})$ is the current W vector minus the stepsize times gradient, so the updated weight vector.



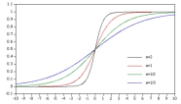
Logistic regression

I prefer to rewrite the formula in matrix notation to make the product of X plainy visible and also to make the coding easier and to check the alignment of the dimensions when multiplying. The below notation is formalistic heresy.

$$W - \alpha \left(\left(Y - \sigma(W\, X^T) \right) X \right)$$

Where

- M is the number of samples or rows in the data set
- N the dimension or number or columns in the data set
- X are the input variables of the sample data, so (M,N)
- Y are the categories or classes or results of the sample (M,1)
- W are the weight factors (N,1) or (1,N) according to the way you like to code this vector.



Logistic regression

Read data M rows of N columns and create the augmented matrix. This is the DataMatrix of dimension (M,N)

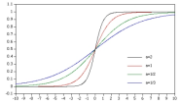
Read M rows of labeled categories. This is the LabelMatrix of dimension (M,1)

Initialize the WeightMatrix of dimension (N,1), for example set all weights to 0

Iterate (for a predefined number of cycles or until the gradient does not change any more)

- Calculate the sigmoid of the multiplication of WeightMatrix and DataMatrix (TempMatrix)
- LabelMatrix minus TempMatrix is Temp2Matix
- Temp2Matrix multiply with DataMatrix is Temp3Matrix (DO NOT FORGET THIS)
- multiply Temp3Matrix with the stepsize (GradientMatrix)
- Previous WeightMatrix minus GradientMatrix

```
//  FORMULE = W - alfa ( (y - sigmoid( w XT ))X )
double minLoss = -1;
double prevloss = 0;
double  maxAccuracy = Double.NaN;
int minLossCycle=-1;
for(int cycle=0;cycle<NbrOfCycles;cycle++)
{
 cmcMatrix h1 = vrout.multiplyMatrix( weightMatrix , dataMatrixTransposed ); //
(1,n) * (n,m) =>  (1,m)

 cmcMatrix h2b = vrout.sigmoidMatrix( h1 ); // (1,m)

 cmcMatrix errorMatrix = vrout.substractMatrix( targetMatrixTransposed , h2b );
// 1,m - 1,m = 1,m

  // (y - sigmoid( w XT )) X     (1,m) (m,n) => 1,n    => number of columns
  cmcMatrix gradientMatrix = vrout.multiplyMatrix( errorMatrix , dataMatrix );

  //  apply stepsize
  cmcMatrix t2 = vrout.scalarMultiplyMatrix( StepSize , gradientMatrix);  // alfa
* 1,n

  //
  prevWeight = weightMatrix;  // 1,n
  weightMatrix = vrout.addMatrix( weightMatrix , t2 );   // (n,1)

}
```

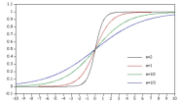


Logistic regression

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Application : cbrTekStraktor V0.4 (18-Apr-2019) -->
<!-- Start : 25-APR.-2019 16:47:21 -->
<LogisticRegressionModel>
  - <LRModelGeneral>
        <ModelApplicationDomain>General</ModelApplicationDomain>
        <ARFFFileName>c:\temp\cbrTekStraktor\Tutorial\MachineLearning\SandBox\Iris - Tweaked.txt</ARFFFileName>
        <MaxCycles>2500</MaxCycles>
        <OptimalCycles>9</OptimalCycles>
        <StepSize>0.001</StepSize>
        <NumberOfFeatures>5</NumberOfFeatures>
        <Weights>[0.9284436250965923][0.2908568403179856][-0.26510555827249416][0.3882196042221724][0.37344737984393694]</Weights>
    </LRModelGeneral>
  + <LogisticRegressionModelEvaluation>
  + <Categories>
  - <Normalisation>
        <NormalisationMean>[5.819230769230772][3.0792307692307666][3.704615384615386][1.188461538461539]</NormalisationMean>
        <NormalisationStdDev>[0.8120312509591537][0.4469875314921896][1.7774147910086608][0.7728568381710111]</NormalisationStdDev>
    </Normalisation>
</LogisticRegressionModel>
```

```
@relation IRIS-One against all

@attribute sepallength NUMERIC
@attribute sepalwidth NUMERIC
@attribute petallength NUMERIC
@attribute petalwidth NUMERIC
@attribute class {IRIS-SETOSA,OTHER}

@data
5.0,3.4,1.5,0.2,IRIS-SETOSA
5.7,4.4,1.5,0.4,IRIS-SETOSA
5.4,3.9,1.3,0.4,IRIS-SETOSA
5.1,3.3,1.7,0.5,IRIS-SETOSA
5.2,3.5,1.5,0.2,IRIS-SETOSA
4.5,2.3,1.3,0.3,IRIS-SETOSA
4.4,3.2,1.3,0.2,IRIS-SETOSA
7.0,3.2,4.7,1.4,OTHER
6.0,2.2,4.0,1.0,OTHER
5.6,2.9,3.6,1.3,OTHER
```

Observe the weights and normalization parameters

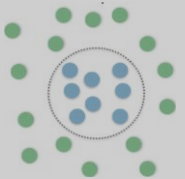


Stochastic gradient descent

Identical to logistic regression

However instead of optimizing on the entire sample data set “random” (stochastic) samples are taken.

Support Vector Machines



Support Vector Machine (SVM) is a supervised learning classification algorithm.

Sequential Minimal Optimization is an optimized SVM

SVM/SMO work on binary result classes -1 and 1 and also on **NON**-linearly separable data

Data scientists consider SMO to be the best “out of the box” classifier, i.e. it delivers good accuracy, low bias and has good training performance.

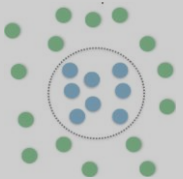
94

Invented by Vapnik (1970-1990) and elaborated by Platt (SMO is as recent as of 1998)

You might need a bit of background information before tackling SVM-SMO.

- Hyperplanes
- Kernel trick
- Lagrange multiplier

SVM - SMO



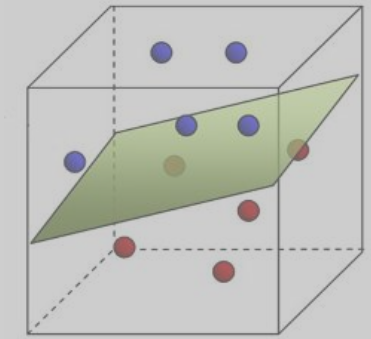
It is based on the idea of finding the hyperplane that separates a set of sample data the best.

A hyperplane is a “subspace of one dimension less than the ambient space”. Wow!

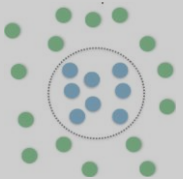
The equation of a hyperplane can be written as $\overline{w} \,.\, \bar{x}$

$\overline{w}$ is the support vector which is perpendicular to the hyperplane.

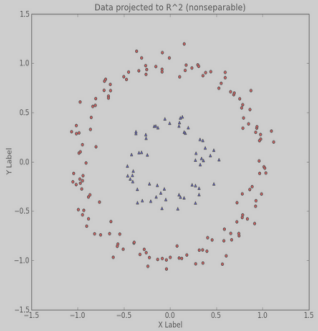
95

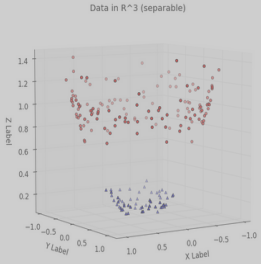


SVM - SMO



Sometimes a hyperplane (or hyper line) cannot be found



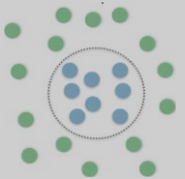


96

In those cases a **Kernel Trick** is performed. This increases the dimensionality of the variables in the sample data. In the above picture you cannot fine a line to separate the data, however you can find a plane if you move into 3D.

Luckily (well Mr.Vapnik probably contributed a bit) in SVM-SMO all operations on the N++ data are “inner products”, i.e. result in a single scaler (a number). You are not facing the complexities of the N++ dimensions.

SVM-SMO



Optimization problem

$$minimize_x \quad f(x)$$
$$subject\ to \quad g(x) = 0$$

Lagrange (1736-1813) discovered that to find the minimum x (depending on g(x)=0) it appears that when that minimum of f is found the partial derivative of f and g are pointing in the same direction (vectors have the same slope or inclination).

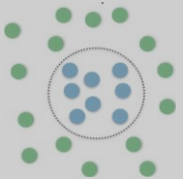
97

$$\nabla f(x) = \propto \nabla\, g(x)$$

So if you want to find a minimum you 'just' solve the following, i.e. you go an find the alphas (the Lagrange multipliers).

$$\nabla \mathcal{L}(x, \propto) \;=\; \nabla f(x) - \propto \nabla\, g(x) = 0$$

SMO



The SVM algorithm requires “quadratic solvers” to find the Lagrange multipliers. I have therefore left SVM out of the handouts and presentation. John Platt simplified / optimized the SVM into the SMO algorithm.

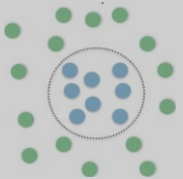
You can visualize SMO as follows: we want to find a fitting hyperplane; which is located the furthest of the closest data point. So a maximization problem.

“Given a linearly separable set of training data find the optimal separating hyperplane by defining a normal vector W and bias b for which the geometric margin is the largest”.

$$maximimize_{w,b}\ (minimum_{i=1..m}\ \gamma_i)$$
$$subject\ to\ \gamma_i \geq M \quad For\ each\ i = 1\ ..m$$

- M is the geometric margin $= \ minimum_{i=1..m}\ \gamma_i$
- $\gamma_i = \ y_i(\frac{w}{\|W\|}.x_i + \frac{b}{\|W\|})$ (normalized hyperplane function)

SMO



Without going into too much detail, ‘it happens’ that the SMO algorithm solves the following optimization problem. The objective function is

$$minimize_{\propto} \quad \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \propto_i \propto_j y_i y_j K(x_i . x_j) - \sum_{i=1}^{m} \propto_i$$

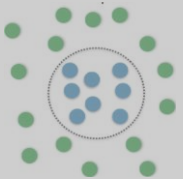
Subject to the following inequality and equality constraints

$$\begin{cases} C \geq \propto_i \geq 0 \quad for\ any\ i = 1..m \\ \sum_{i=1}^{m} \propto_i y_i = 0 \end{cases}$$

99

- $\propto$ is the Lagrange multiplier. There is one Lagrange multiplier for each row. Rows with a Lagrange multiplier different from zero are in fact the data points out of which the support vectors start.
- C is the Soft Margin constant. A small C gives a wider margin at the cost of misclassification, a very large C does not accommodate for any tolerance at all. C is set via trial and error, but remember that C is proper to a certain type of data set.
- K is the Kernel trick function. It is used to move from N dimension to N+1 dimension in order to solve non-linearly separable distributions.
- M is the number of rows in the sample data set
- Y is the labeled result (the category defined to be correct in the training data).

SMO



The idea : SMO breaks down the SVM into small size optimization problems.

We initialize all multipliers to zero and we take two multipliers ($\propto_1, \propto_2$) which we willstart tweaking (actually we apply gradients changes to $\propto_1^{new}$ and $\propto_2^{new}$). We keep on changing the two chosen Lagrange multipliers until the objective function reaches a minimum.

We then take two other multipliers and keep on changing those multipliers until the objective function again reaches a minimum. We continue doing this until there are no more changes occurring on the multipliers.

The second Lagrange multiplier is calculated as follows.

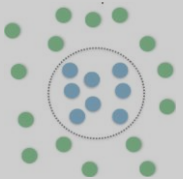
$$\propto_2^{new} = \propto_2^{previous} + \frac{y_2(E_1 - E_2)}{K(x_1.x_1) + K(x_2.x_2) - 2K(x_1.x_2)}$$

The first Lagrange multiplier can be calculated as follows

$$\propto_1^{new} = \propto_1^{previous} + y_1 y_2 (\propto_2^{previous} - \propto_2^{previous})$$

K is the kernel trick. E is the difference between actual and result calculated using the hypothesis function

SMO



Platt defined a smart approach for finding the best performing multipliers. The so called Full Platt SMO algorithm consists of an outer loop for choosing the first Lagrange multiplier and an inner loop for choosing the second multiplier (or alpha).

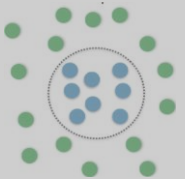
The outer loop switches between single passes over the entire dataset and single passes over non-bound multipliers (the alphas which are not bound to the 0 and C limit). The run over the entire dataset is as easy as it sounds, i.e. just loop over every alpha in the set. The pass over the non-bound set starts by creating an exclusion list of multipliers that won't change.

The second Lagrange multiplier is selected once we have the first one and is chosen in such a way that it will maximize the step size during optimization. We keep track of the error values and choose the set of alphas that maximize the step size, stated differently we pick the set of alphas with the largest difference between E1 and E2.

101

(Just mention bounded and unbounded multipliers and KKT constraints)

SMO



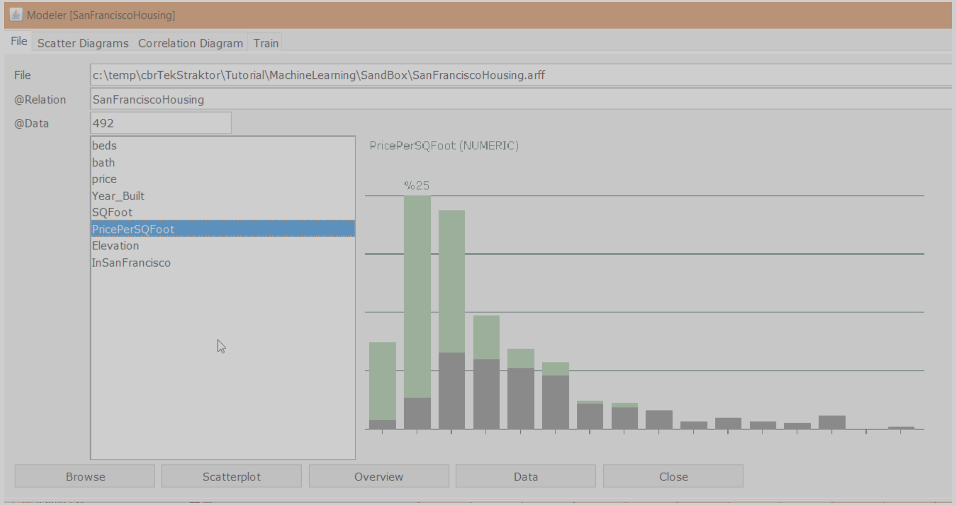
```
double Ei = calculateError( i , dto );
// see whether Lagrange multipliers can be changed
if( (((resultMatrix.getValues()[i][0]*Ei) < (0-dto.getTolerance())) && (alphaMatrix.getValues()[i][0] < dto.getC()))  ||
(((resultMatrix.getValues()[i][0]*Ei) > dto.getTolerance()) && (alphaMatrix.getValues()[i][0] > 0)) )
{
  int j = getNextIndexForMultiplier( i , dto );
  double Ej = calculateError( j , dto );
  if( Double.isNaN(Ej) ) return -1;
  //
  double alphaIOld = alphaMatrix.getValues()[i][0];
  double alphaJOld = alphaMatrix.getValues()[j][0];
  double labelI = resultMatrix.getValues()[i][0];
  double labelJ = resultMatrix.getValues()[j][0];
    //...alphas must always between 0 and C
  if( labelI != labelJ ) {
   L = Math.max(  0 , (alphaJOld - alphaIOld) );
   H = Math.min(  dto.getC() , dto.getC() + (alphaJOld - alphaIOld) );
  }
  else {
   L = Math.max(  0 , (alphaJOld + alphaIOld) - dto.getC() );
   H = Math.min(  dto.getC() , (alphaJOld + alphaIOld) );
  }
  if( L==H ) return 0;
  // 2nd lagrange multiplier - begin met noemer
  double dKxixi = getKernelTrickedXiXj( dataMatrix , i , i , dto);
  double dKxixj = getKernelTrickedXiXj( dataMatrix , j , j , dto);
  double dKxixj = getKernelTrickedXiXj( dataMatrix , i , j , dto);
  double eta = dKxixi + dKxjxj - ((double)2 * dKxixj);
  if( eta <= 0 )  return 0;

 //...2nd Lagrange multiplier - alpha2
  double alphaJ = alphaJOld + ((labelJ * (Ei - Ej)) / eta);
  alphaJ = clipAlpha( alphaJ , H , L );
  dto.getAlphaMatrix().setValue( j , 0 , alphaJ );
  if( recalculateAndStoreError( j , dto ) == false ) return -1;
  //
  if( Math.abs( alphaJ - alphaJOld ) < 0.00001 ) return 0;

 //...1st Lagrange multiplier   alpha + yiyj(alph2old ) alpha2)
  double alphaI = alphaIOld + (labelJ * labelJ * ( alphaJOld - alphaJ));
  dto.getAlphaMatrix().setValue( i , 0 , alphaI );
  if( recalculateAndStoreError( i , dto ) == false ) return -1;
 //...b
  double b1 = dto.getBias() - Ei - (labelI * (alphaI - alphaIOld) * dKxixi) - (labelJ * (alphaJ - alphaJOld) * dKxixj);
  double b2 = dto.getBias() - Ej - (labelI * (alphaI - alphaIOld) * dKxixj) - (labelJ * (alphaJ - alphaJOld) * dKxjxj);
  //
  if( (0 < alphaI) && (dto.getC() > alphaI) ) dto.setBias(b1);
  else
  if( (0 < alphaJ) && (dto.getC() > alphaJ) ) dto.setBias(b2);
  else {
    dto.setBias( ( b1 + b2) / (double)2 );
  }
   return 1;
 }
 else return 0;
```

```
// FULL Platt SMO version
int alphaPairChanges=0;
boolean entireSet=true;
int cycle = 0;
while ( (cycle <dto.getMaxCycles()) || (alphaPairChanges>0) || (entireSet==true) )
{
      alphaPairChanges=0;
      if( entireSet ) {
            for(int i=0;i<dto.getNbrOfRows();i++)
            {
            int change = innerLoop( i , dto );
            if( change < 0 ) { do_error( "innerloop error I"); return false; }
            alphaPairChanges += change;
            }
            cycle++;
      }
      else {
            boolean[] nonBoundList = new boolean[ dto.getNbrOfRows() ];
            for(int i=0;i<nonBoundList.length ; i++ ) nonBoundList[i]=false;
                  cmcMatrix alphaMatrix = dto.getLagrangeMultiplierMatrix();
                  for(int i=0;i<alphaMatrix.getNbrOfRows();i++)
                  {
                  if( (alphaMatrix.getValues()[i][0] > 0) &&
(alphaMatrix.getValues()[i][0] < dto.getC()) ) nonBoundList[i]=true;
                  }
                        //
                  for(int i=0;i<nonBoundList.length;i++)
                  {
                  int change = innerLoop( i , dto );
                  if( change < 0 ) { do_error( "innerloop error II"); return
false; }
                  alphaPairChanges += change;
                  }
                  cycle++;
      }
      if( entireSet == true ) entireSet = false;
      else
      if( alphaPairChanges == 0 ) entireSet = true;
}
```


102

SMO


```
- <SMOModelGeneral>
    <ModelApplicationDomain>General</ModelApplicationDomain>
    <ARFFFileName>c:\temp\cbrTekStraktor\Tutorial\MachineLearning\SandBox\SanFranciscoHousing.arff</ARFFFileName>
    <KernelTrickType>GAUSSIAN</KernelTrickType>
    <NumberOfFeatures>8</NumberOfFeatures>
    <MaxCycles>40</MaxCycles>
    <Tolerance>0.001</Tolerance>
    <SoftMargin>0.6</SoftMargin>
    <NumberOfSupportVectors>226</NumberOfSupportVectors>
  </SMOModelGeneral>
- <SMOModelEvaluation>
  - <TrainedModel>
      <TrainedModelAcurracy>0.8691588785046729</TrainedModelAcurracy>
      <TrainedModelEntries>428</TrainedModelEntries>
    - <![CDATA[

                 0          1 ---> Actual
             ----------  ----------  ----------
      0      [    186][       9]
      1      [     47][     186]

              Precision  Sensitivity   FMeasure
             ----------  ----------  ----------
      0      [ 0,953846][  0,798283][  0,869159]
      1      [ 0,798283][  0,953846][  0,869159]
      ]]>
    </TrainedModel>
  - <TestedModel>
      <TestedModelAcurracy>0.78125</TestedModelAcurracy>
      <TestedModelEntries>64</TestedModelEntries>
      <TestSetPercentage>13</TestSetPercentage>
    - <![CDATA[

                 0          1 ---> Actual
             ----------  ----------  ----------
      0      [     23][       2]
      1      [     12][      27]

              Precision  Sensitivity   FMeasure
             ----------  ----------  ----------
      0      [ 0,920000][  0,657143][  0,766667]
      1      [ 0,692308][  0,931034][  0,794118]
      ]]>
    </TestedModel>
  - <Normalisation>
      <NormalisationMean>[2.1320093457943927][1.871728971962617][1953181.1939252336][1957.978971962617][1509.3551401869158][1178.8084112149534][40.427570093457945]
        </NormalisationMean>
      <NormalisationStdDev>[1.2608233731838812][0.9932888627077568][2712124.804688644][40.760303814993186][1016.9112207843528][708.3869334741485][44.54150380594308]
        </NormalisationStdDev>
    </Normalisation>
  </SMOModelEvaluation>
- <LinearEquation>
    <Weights>[7.617108497486934][-0.5531834443758502][-11.225451102807273][8.347143766934021][4.18722306933347][-24.789087759355283][26.088099501702263]</Weights>
    <Bias>0.6564768440738586</Bias>
  </LinearEquation>
```

103

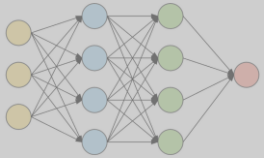
Neural Networks

Perceptron

ADALINE

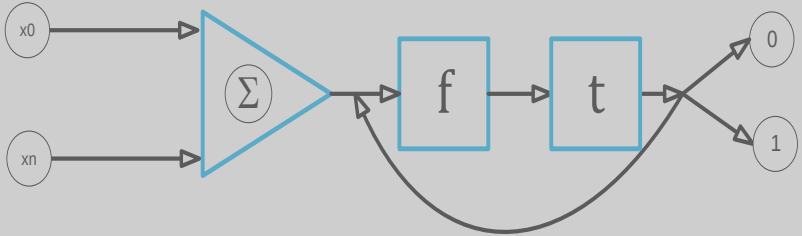
Deep Learning

Perceptron



A perceptron is a classifier for 2 classes {C1, C2} or {-1, 1}.

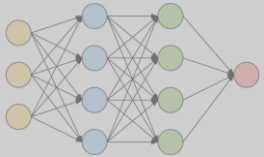
It relies on feedback provided by a loss function (mean squared error)



105

- An aggregator (sigma), which aggregates the inputs from various input variables {x0, x1, .. , xn}
- An activation or transfer function f($\sum \bar{w}^T \bar{x}$).
- A perceptron also limits the regression equation to the first degree so limited to (w_0, w_1)
- A step function (t) which transforms the output of f into {-1,1}

Perceptron



$$\overline{w_{k+1}} = \overline{w_k} + \nabla \sum_{i=1}^{M} w^T x_i t_i$$

Where

M is the number of samples

N is the number of dimensions (features, columns)

M is the dimensionality, in the case of perceptron 2, so M=1

w_1 is the slope and w_0 is the bias (or intercept)

w^T is the transposed weight matrix

Step 1. Initiate the weights randomly

Step 2. Compute y using the current weights (keep in mind that y I either -1 or 1)

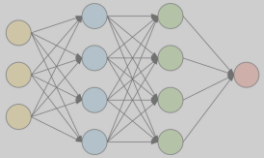
106

Step 3. If all calculated (predicted) classes are equal to the actual class or category then stop

Step 4. For all samples where the calculated y differs from the actual class or category perform step 5

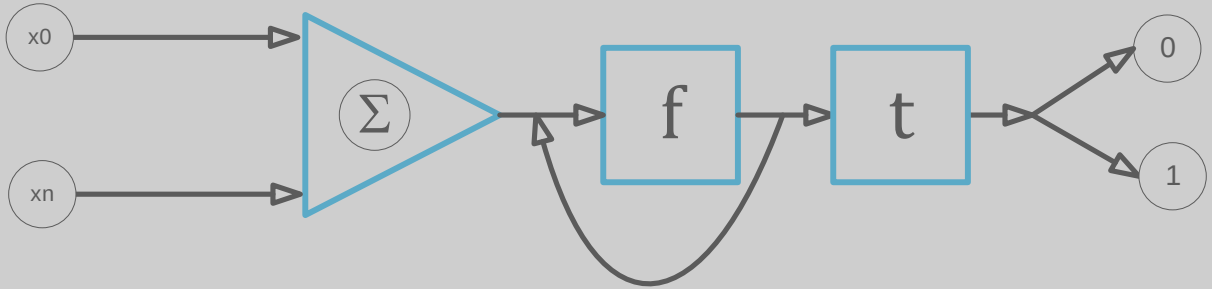
Step 5. Adjust the weights whereby the adjustment: Δ weight = actual label . x_i. This means that you pick a random sample which misclassified (i), fetch the actual class y_i and substract the calculated value of y to get the updated set of weights and go back to Step 2

Adaline



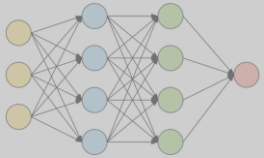
The Adaline (Adaptive Linear Neuron or later Adaptive Linear Element) algorithm is a modification of the Perceptron.

The perceptron algoritm adjusts the weight vector by performing a Mean Squared Loss function on the results of the T function (-1,1). Adaline performs the adjustment on the values of the activation function (f), so the values prior to be being submitted to the step function T.



107

Deep learning



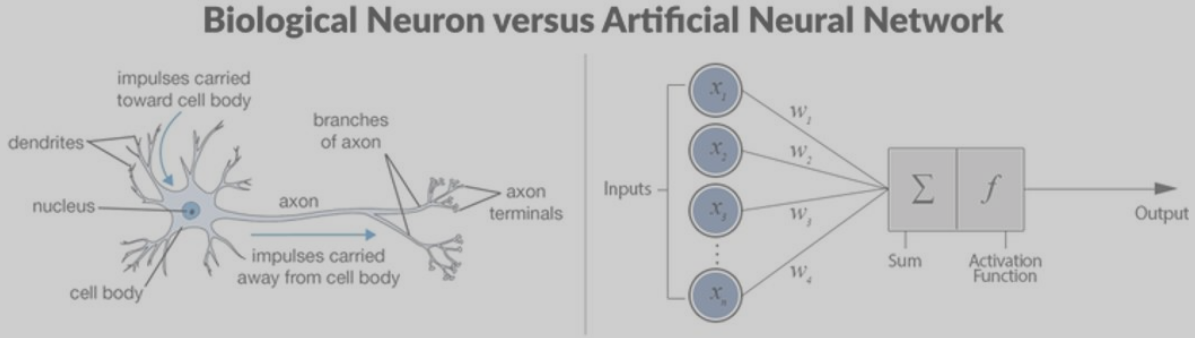
Part of machine learning field of learning representations of data

Exceptionally effective at learning patterns

Utilizes learning algorithms that derive meaning out of data by using hierarchy of multiple layers that mimic the neural networks of our brain

If you provide the system tons of information, it begins to understand it and respond in useful ways

No feature engineering required!



108

Back-up slides

Types of Combinations / Ensemble Learning

All three are so-called "meta-algorithms": approaches to combine several machine learning techniques into one predictive model in order to decrease the variance (*bagging*), bias (*boosting*), or improving the predictive force (*stacking* alias *ensemble*).

Bagging (Bootstrap Aggregation)	Boosting
parallel ensemble: each model is built independently	**sequential** ensemble: try to add new models that do well where previous models lack
aim to **decrease variance**, not bias suitable for high variance low bias models (complex models)	aim to **decrease bias**, not variance suitable for low variance high bias models
example of a tree-based method is **random forest**	example of a tree-based method is **gradient boosting**

Algorithms Comparison - Classification