Supervised Learning

Achmad Wildan Al Aziz

Universitas Airlangga, 6 November 2018

Outline

1. Introduction
2. Linear Regression
3. K-Nearest Neighbour
4. Support Vector Regression
5. Evaluation Metrics
6. Cross Validation

Machine Learning

Supervised

Clasification

Parametric

- Logistic Regression
- Discriminant Analysis

Non Parametric

- SVM
- Deep Learning
- KNN
- etc

Regression

Parametric

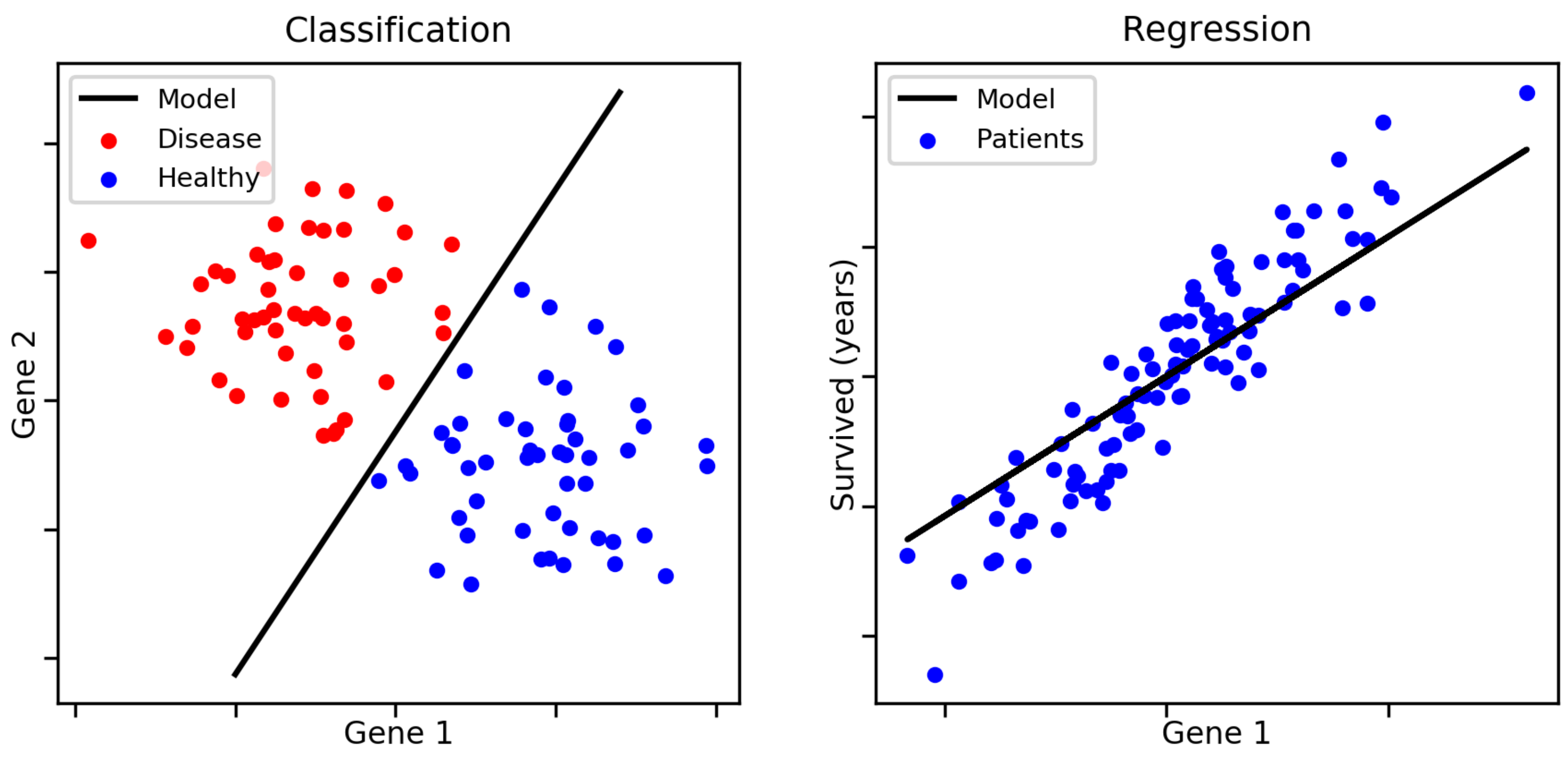
- Multiple Linear Regression

Non Parametric

- SVR
- Deep Learning
- KNN
- etc

Unsupervised

Introduction



https://aldro61.github.io/microbiome-summer-school-2017/figures/figure.classification.vs.regression.png

Data Structure

Observations	Y	X_1	X_2	.	.	.	X_k
1	Y_1	X_{11}	X_{12}	.	.	.	X_{1k}
2	Y_2	X_{21}	X_{22}	.	.	.	X_{2k}
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
n	Y_n	X_{n1}	X_{n2}	.	.	.	X_{nk}

Classificatin Case

Y: Nominal / Ordinal

X: Nominal / Ordinal / Interval / Rasio

Regression Case

Y: Interval / Rasio

X: Nominal / Ordinal / Interval / Rasio

Parametric vs. nonparametric algorithms

III DATA MODELING

	Pros	Cons
Parametric algorithms	**Simpler** Easier to understand and to interpret **Faster** Very fast to fit your data **Less data** Require "few" data to yield good perf.	**Limited complexity** Because of the specified form, parametric algorithms are more suited for "simple" problems where you can guess the structure in the data
Nonparametric algorithms	**Flexibility** Can fit a large number of functional forms, which doesn't need to be assumed **Performance** Performance will likely be higher than parametric algorithms as soon as data structures get complex	**Slower** Computations will be significantly longer **More data** Require large amount of data to learn **Overfitting** We'll see in a bit what this is, but it affects model performance

https://www.slideshare.net/CharlesVestur/building-a-performing-machine-learning-model-from-a-to-z

Linear Regression

- Model the relationship between dependant variable y and independent variable(s) X_1, X_2, …, X_p

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_p + \varepsilon_i$$

$$y_i = X_i^T \boldsymbol{\beta} + \varepsilon_i$$

$$y = \boldsymbol{X\beta} + \varepsilon$$

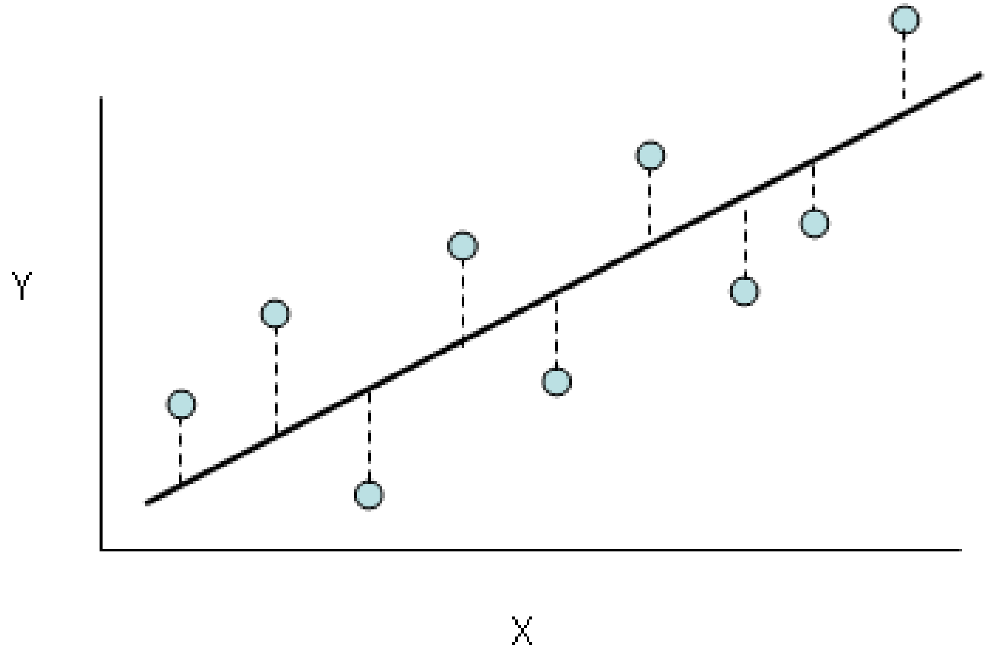
$$\begin{pmatrix} y_1 \\ y_1 \\ \vdots \\ y_\mathrm{n} \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots x_{1p} \\ 1 & x_{21} & x_{22} & \ldots x_{2p} \\ \vdots & \vdots & \vdots & \ddots \quad \vdots \\ 1 & x_{\mathrm{n}1} & x_{\mathrm{n}2} & \ldots x_{np} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

Linear Regression

Goals

- Estimate β using least squares ⇒ $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T y$
- find predictor variable(s) influencing the response
- Prediction

Regression works by minimizing sum square error



Source :*www.spcforexcel.com*

Linear Regression

Model the relationship between fuel consumption (mpg) to engine power (hp) and car weight (wt)

```
data("mtcars")

head(mtcars)
```

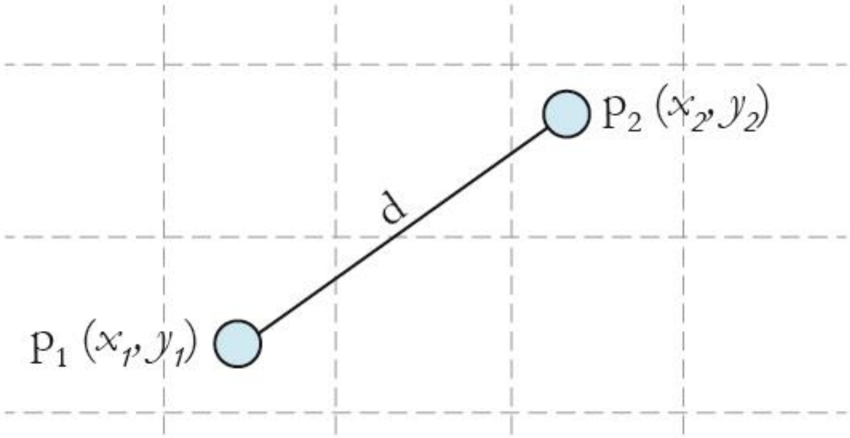
	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

```
model=lm(mpg ~ hp + wt, data=mtcars)
summary(model)
```

```
Call:
lm(formula = mpg ~ hp + wt, data = mtcars)

Residuals:
   Min     1Q Median     3Q    Max
-3.941 -1.600 -0.182  1.050  5.854

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
hp          -0.03177    0.00903  -3.519  0.00145 **
wt          -3.87783    0.63273  -6.129 1.12e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.593 on 29 degrees of freedom
Multiple R-squared:  0.8268,  Adjusted R-squared:  0.8148
F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

K Nearest Neighbours

- KNN can be used for classification and regression
- Algorithm :
  1. Calculate distance each observation
  2. Sort the calculated distances in ascending order based on distance values
  3. Get k smallest distances from the sorted array, and calculate the average (regression) or count the majority vote (classification)

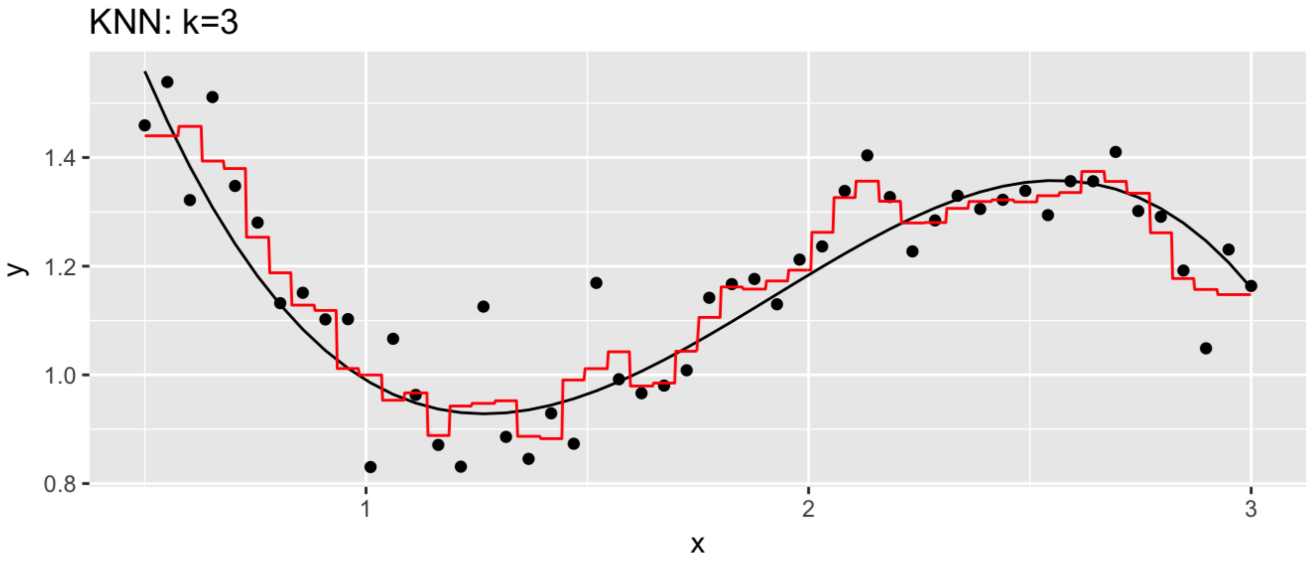


$$\text{Euclidean distance (d)} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Source : *Gamesetmap*

K Nearest Neighbours

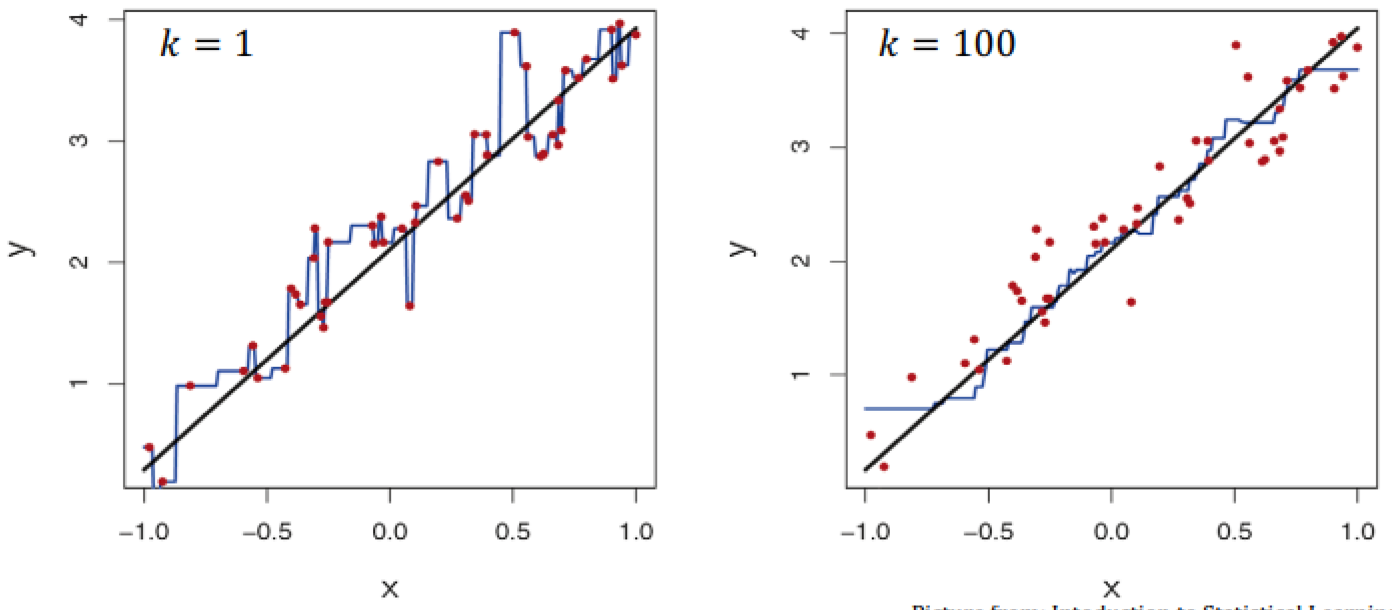
KNN Regression



Source : *dereksonderegger.github.io*

K Nearest Neighbours

- Example of KNN with different K



Picture from: Intoduction to Statistical Learning

Distance functions

Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan $$\sum_{i=1}^{k}\left|x_i - y_i\right|$$

Minkowski $$\left(\sum_{i=1}^{k}\left(\left|x_i - y_i\right|\right)^q\right)^{1/q}$$

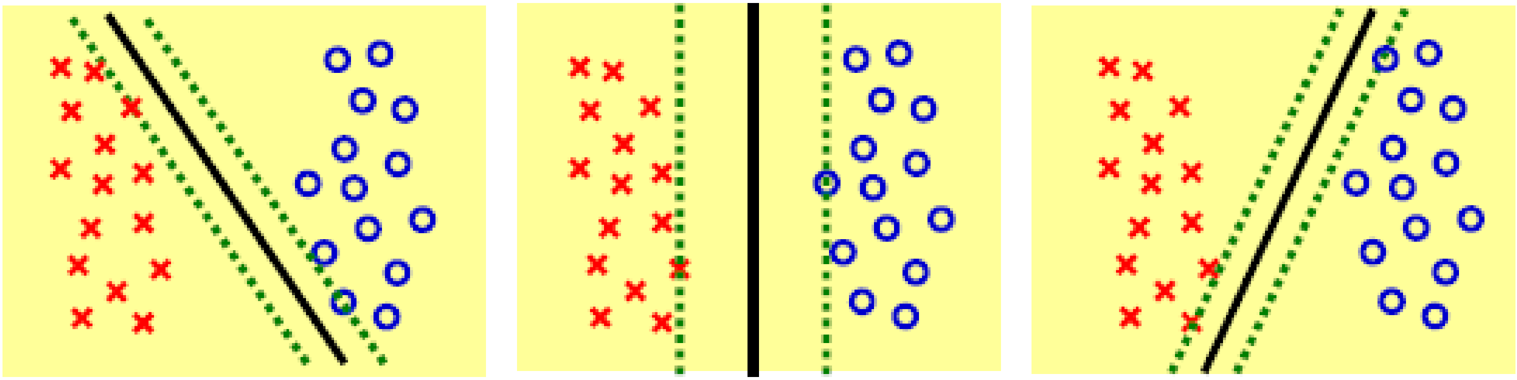
Source : *http://www.saedsayad.com*

KNN Regression

```
library(caret)
data(BloodBrain)

inTrain <- createDataPartition(logBBB, p = .8)[[1]]

trainX <- bbbDescr[inTrain,]
trainY <- logBBB[inTrain]

testX <- bbbDescr[-inTrain,]
testY <- logBBB[-inTrain]

fit <- knnreg(trainX, trainY, k = 3)
prediksi = predict(fit, testX)

RMSE(testY,prediksi)
```

Support Vector Machine

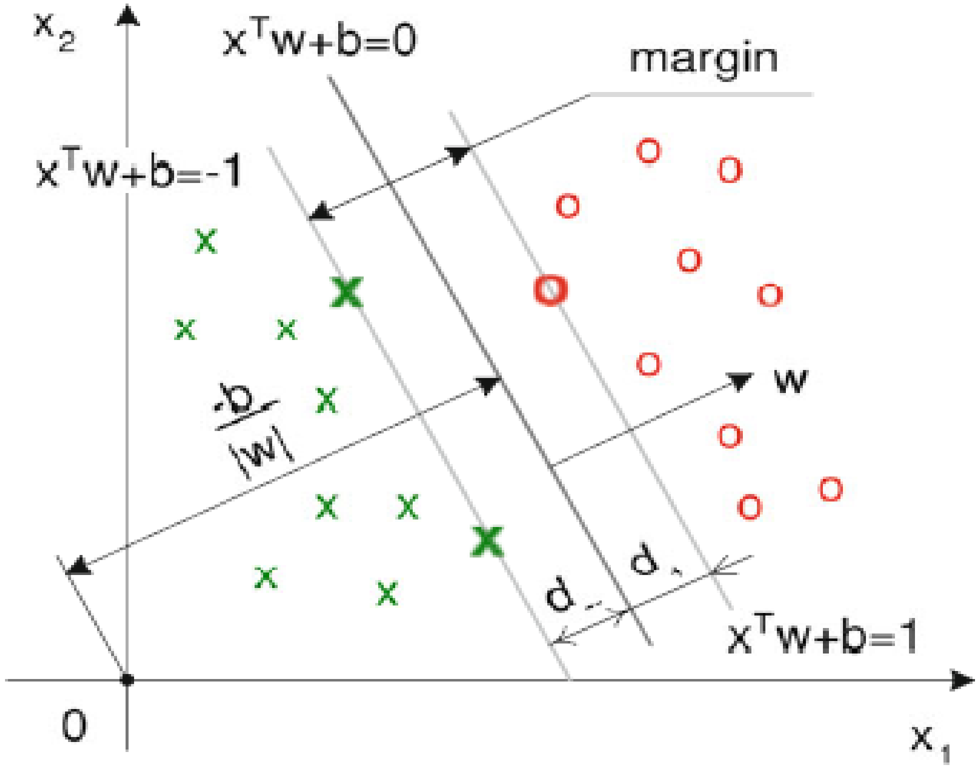
- Main idea SVM is maximize hyperplane



(a) Small margin (b) Large margin (c) Small margin

Source : Introduction to Statistical Machine Learning (Mashashi Sugiyama)

Support Vector Machine



Hyperplane concept (Haerdle, *et.al.*, 2011)

Dapat direpresentasikan dalam pertidaksamaan

$$y_i(x_i^T w + b) - 1 \geq 0, \quad i = 1, 2, \ldots, n$$

Secara matematis, formulasi problem optimasi SVM untuk klasifikasi linier dalam primal space adalah

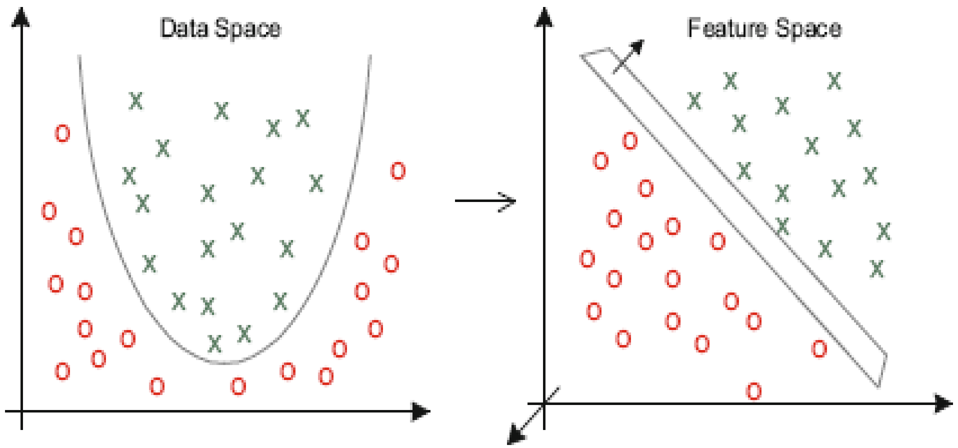
$$\min \frac{1}{2} \|w\|^2$$

Dengan demikian permasalahan optimasi dengan konstrain dapat dirumuskan menjadi

$$L_{pri}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{n} \alpha_i \{ y_i(x_i^T w + b) - 1 \}$$

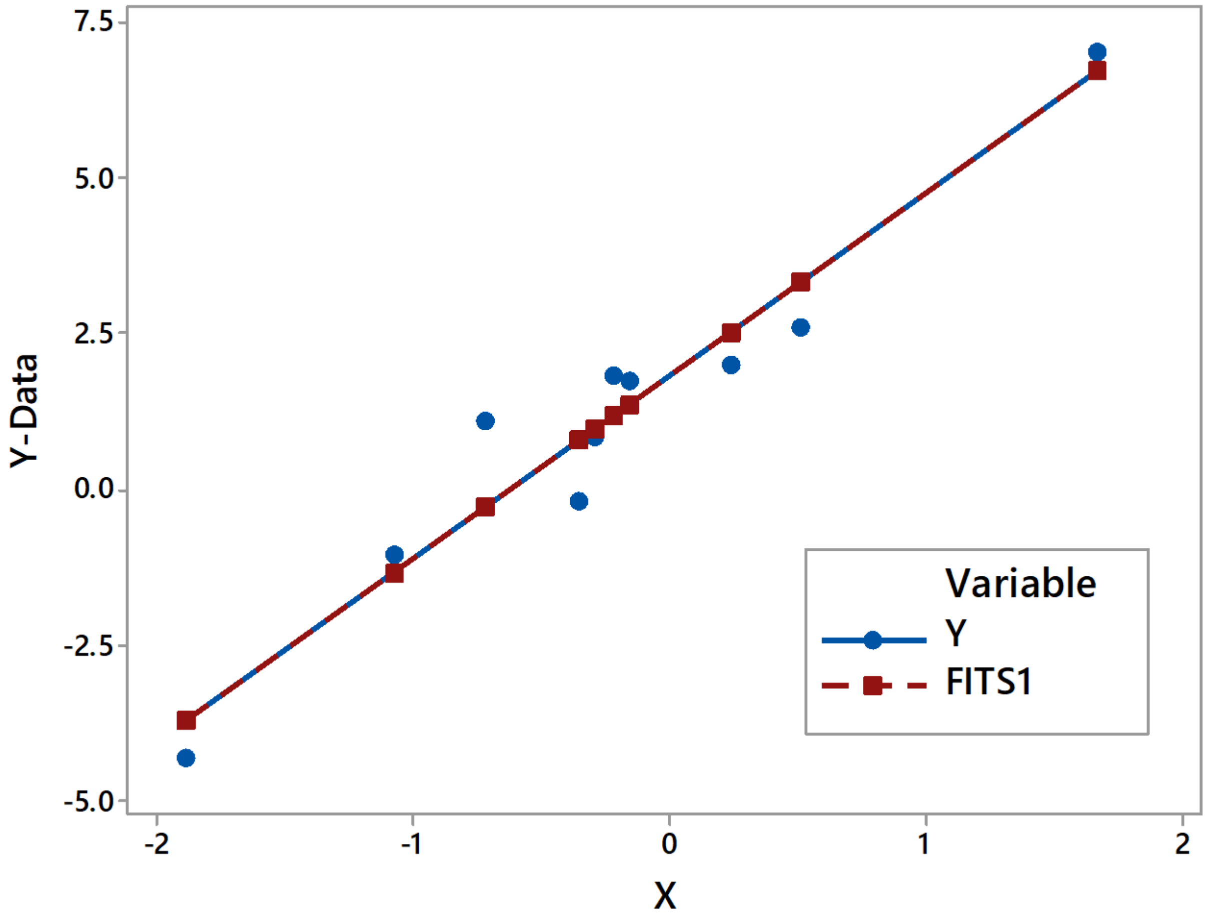
Support Vector Machine

Dalam mencari solusi masalah nonlinier digunakan “kernel trick” yaitu menambahkan fungsi kernel ke dalam persamaan SVM



1. Kernel Linier
2. Kernel Polynomial
3. Fungsi Kernel Radial Basis Function (RBF)
4. Kernel Eksponensial

Evaluation Metrics (Regression Case)



Evaluation Metrics (Regression Case)

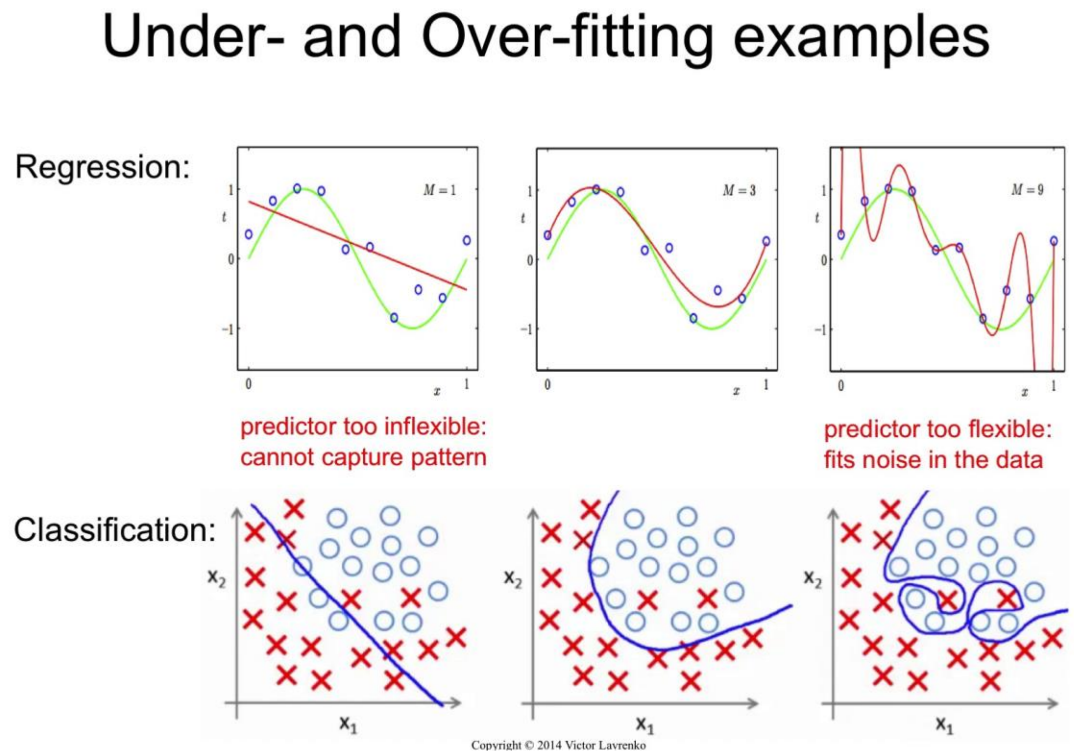
$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2}{n}}$$

$$MAPE = \frac{1}{n}\frac{\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|}{Y_i} \times 100\%$$

$$Median\ APE = median\left(\frac{\left|Y - \hat{Y}\right|}{Y}\right) \times 100\%$$

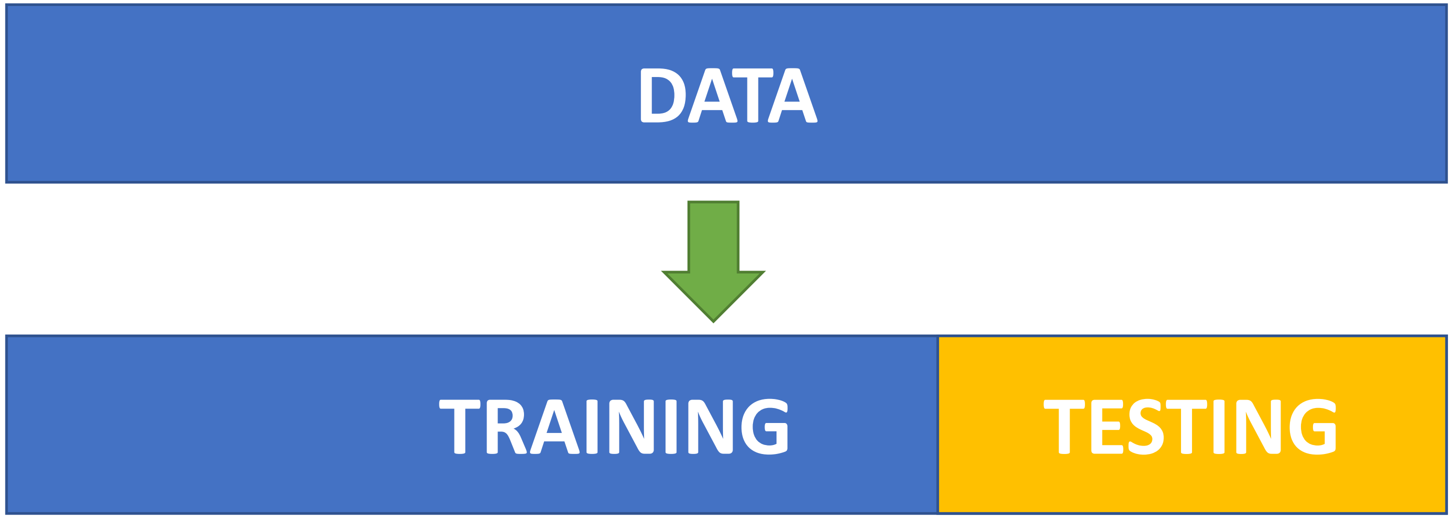
Cross Validation

- Split data into several partitions
- Measure model performance
- Prevent Overfitting
- Make predictive model more robust



Holdout

- Simplest method
- Split data into two parts, train data and test data
- Test data usually smaller than the train data



1

```
data("mtcars")
n=nrow(mtcars)
n
ntrain=sample(n,round(0.7*n))
ntrain

datatrain=mtcars[ntrain,]
datatest=mtcars[-ntrain,]
```

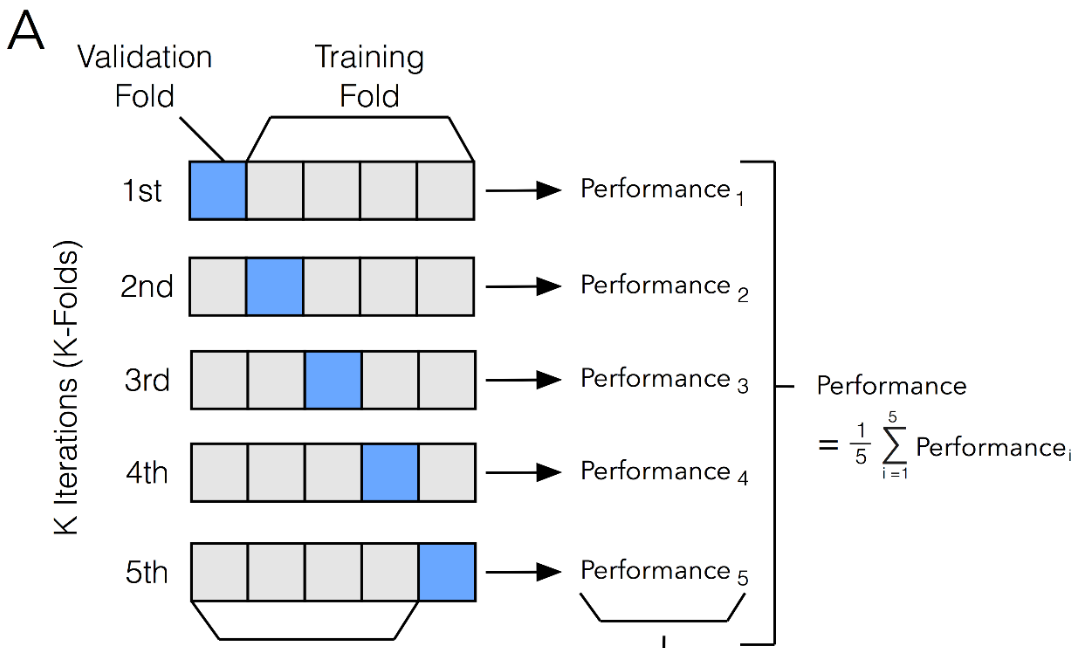
2

```
library(caret)

ntrain2 = createDataPartition (y=
        mtcars$mpg , p=0.7)

training = mtcars[ ntrain2 , ]

testing   mtcars[- ntrain2 , ]
```

K-Fold

- Split data into K parts
- K-1 parts are used as training data, and the rest is used as testing data

Split data into 5 part



DATA SCIENCE INDONESIA

1

```
folds = cut(seq(1,nrow(mtcars)),
        breaks=5,labels=FALSE)
for(i in 1:10){
        index=which(folds==i,arr.ind=TRUE)
        testData = mtcars[index, ]
        trainData = mtcars[- index, ]
}
```

2

```
library(caret)

fold=createFolds(y=mtcars$mpg, k = 5)
```

Thank You