

# Machine learning

## Inleiding & Regressie

---

Wouter Gevaert

# Inhoud

AI in context

Hoe leren uit data?

Enkelvoudige lineaire regressie

Meervoudige lineaire regressie

Feature engineering

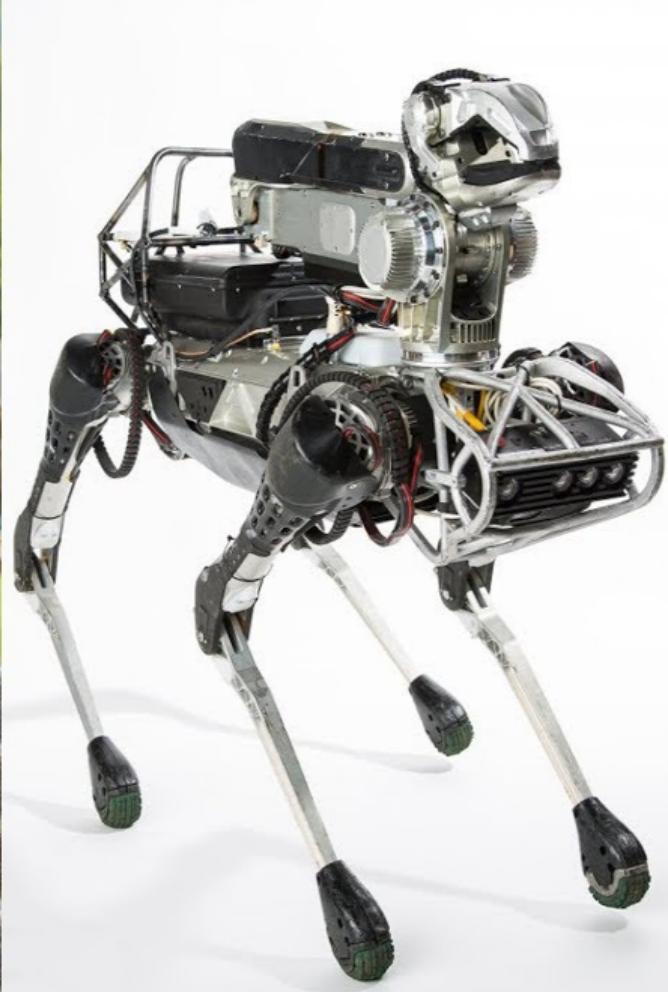
Underfitting en overfitting

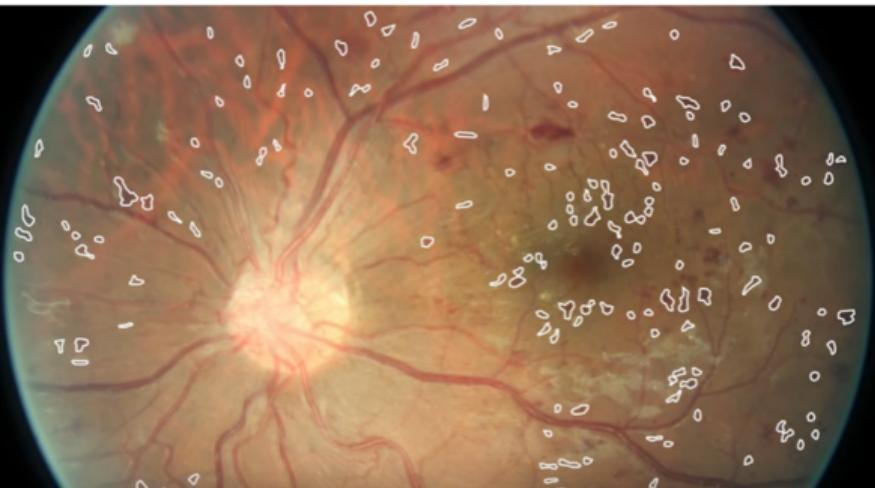
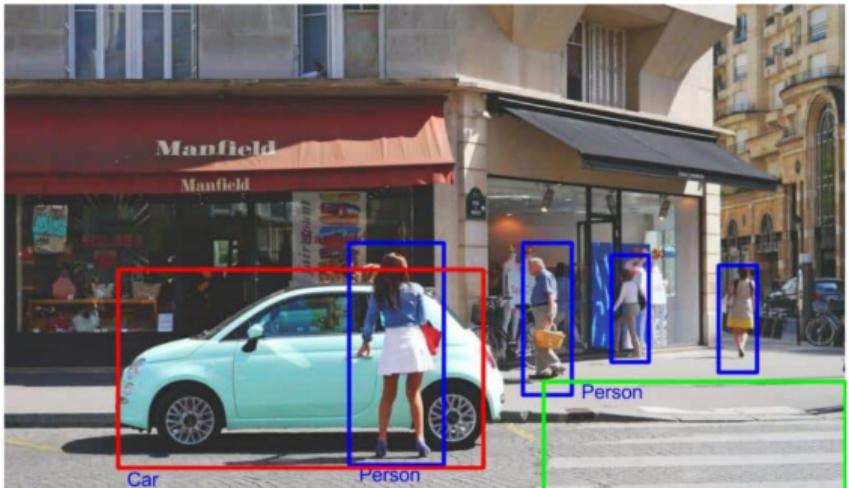
## AI in context

---

# **Facebook legt experiment rond artificiële intelligentie stil nadat robots eigen geheime taal ontwikkelen**

Koen Van De Sype | 01 augustus 2017 | 11u39 | Bron: Next Web, The New York Post, Gizmodo, The Sun





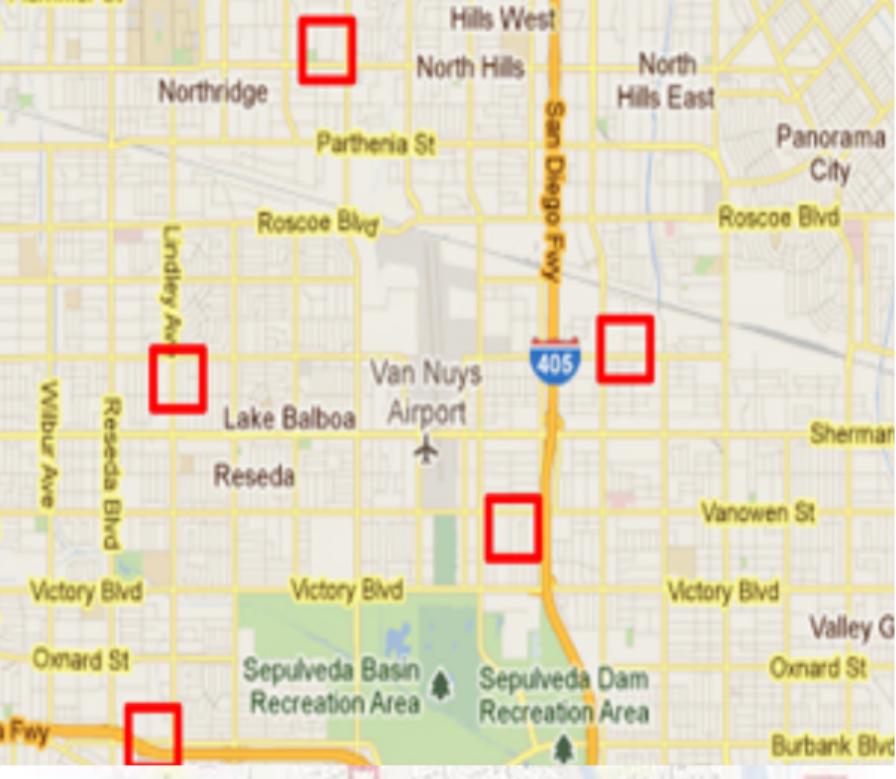
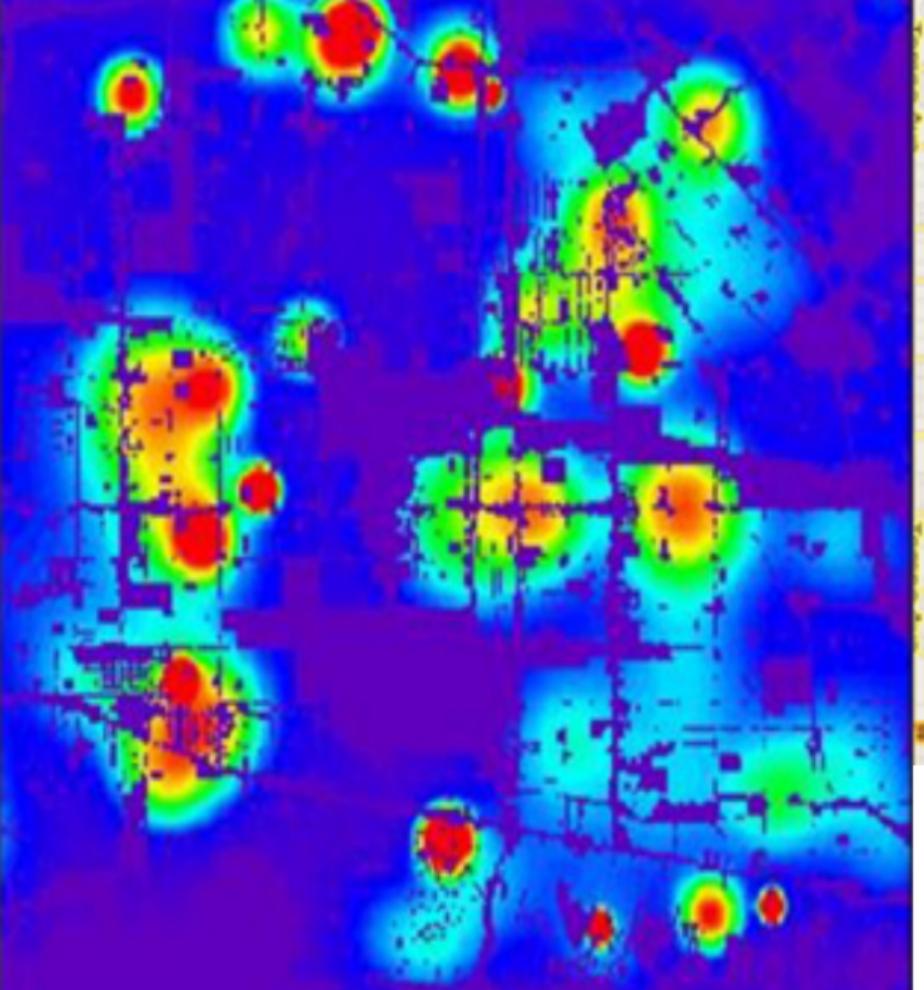
"girl in pink dress is jumping in air."



"black and white dog jumps over bar."



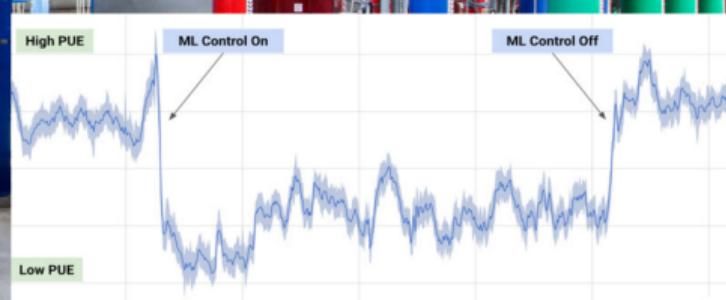


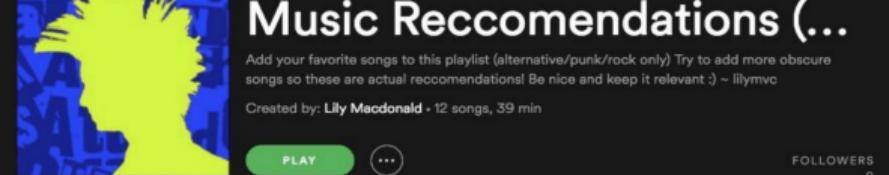


**PredPol**  
Predict Crime in **Real Time**™

PredPol provides targeted, real-time crime prediction  
designed for and successfully tested by officers in the field.



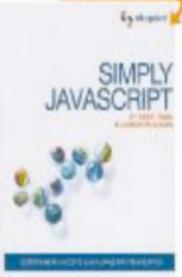
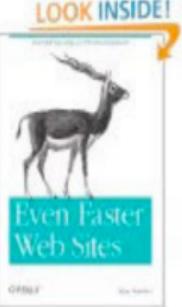




SONG	ARTIST	ALBUM	USER	
+ Shade	Boston Manor	Saudade - EP	Lily Macdonald	32 minutes ago
+ Constant Headache	Joyce Manor	S/T	Lily Macdonald	32 minutes ago
+ Watercolor Daydream	Fossil Youth	Watercolor D...	Lily Macdonald	31 minutes ago
+ Nosey	Sorority Noise	Joy, Departed	Lily Macdonald	31 minutes ago
+ Can't Deny It	Turnstile	Nonstop Feel...	Lily Macdonald	31 minutes ago

## Today's Recommendations For You

Here's a daily sample of items recommended for you. Click here to



[Even Faster Web Sites: Performance Optimizations for Web Developers \(Paperback\)](#) by Steve Souders

★★★★★ (7) \$23.10

[Fix this recommendation](#)

[Simply JavaScript \(Paperback\)](#) by Kevin Yank

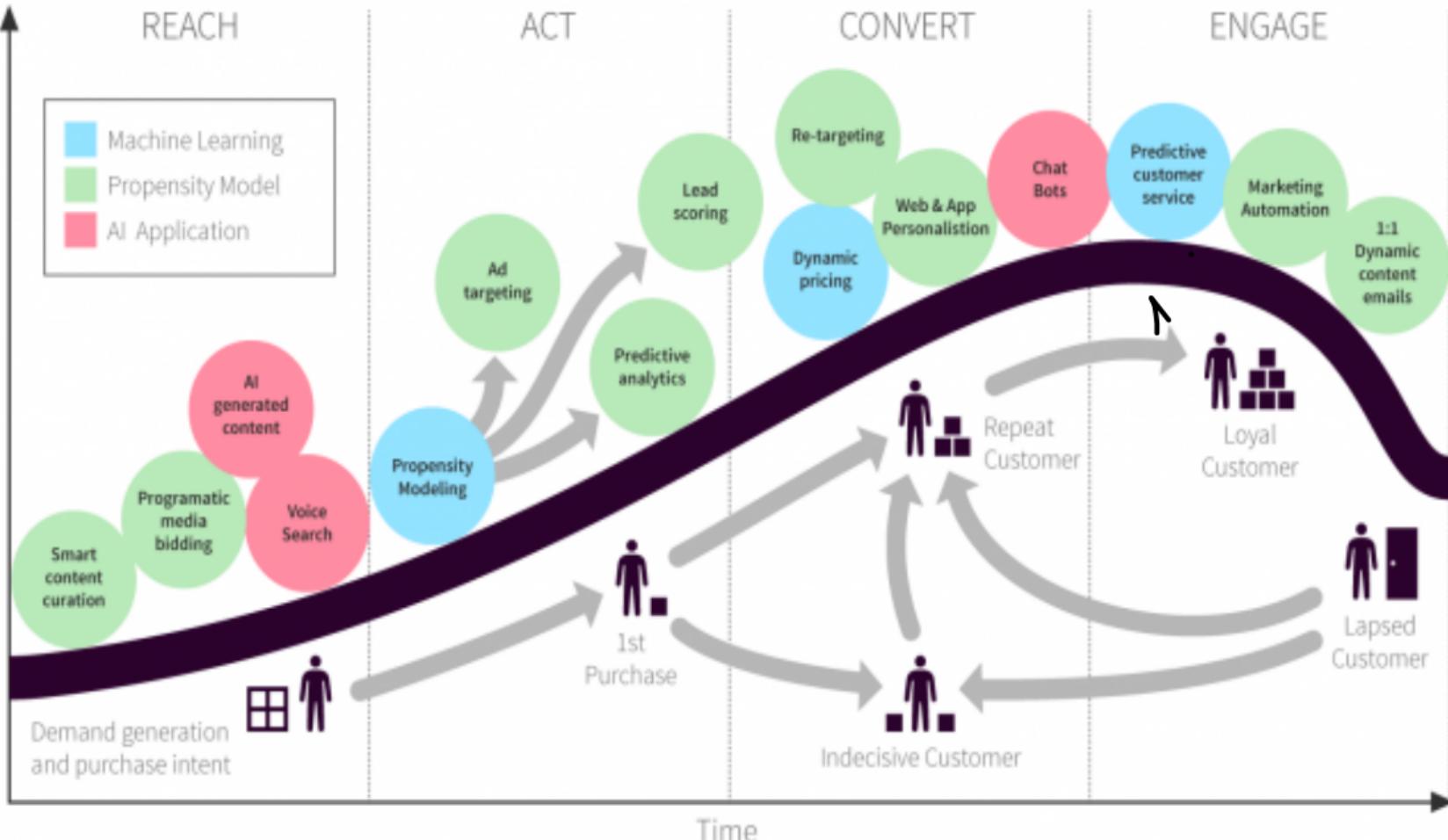
★★★★★ (19) \$26.37

[Fix this recommendation](#)

[Any Category](#)   [Algorithms](#)   [Boxed Sets](#)   [Business & Commercial](#)

[Graphic Design](#)   [Microsoft](#)   [Networking](#)   [Networks, Telecommunications](#)

Customer interactions and value

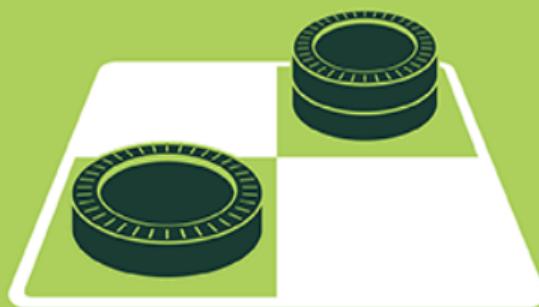






# ARTIFICIAL INTELLIGENCE

Early artificial intelligence  
stirs excitement.



## MACHINE LEARNING

Machine learning begins  
to flourish.

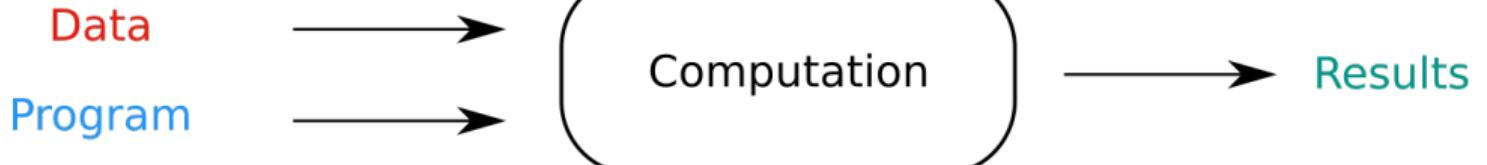


## DEEP LEARNING

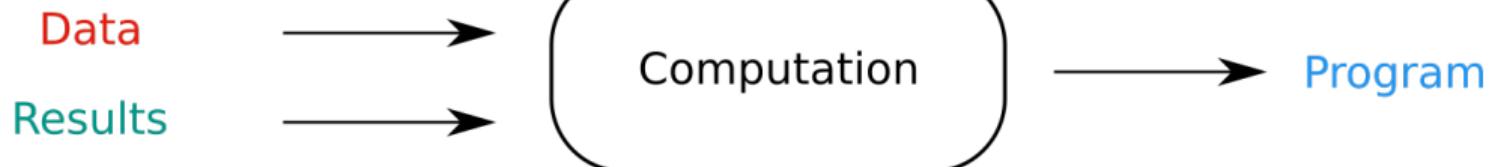
Deep learning breakthroughs  
drive AI boom.

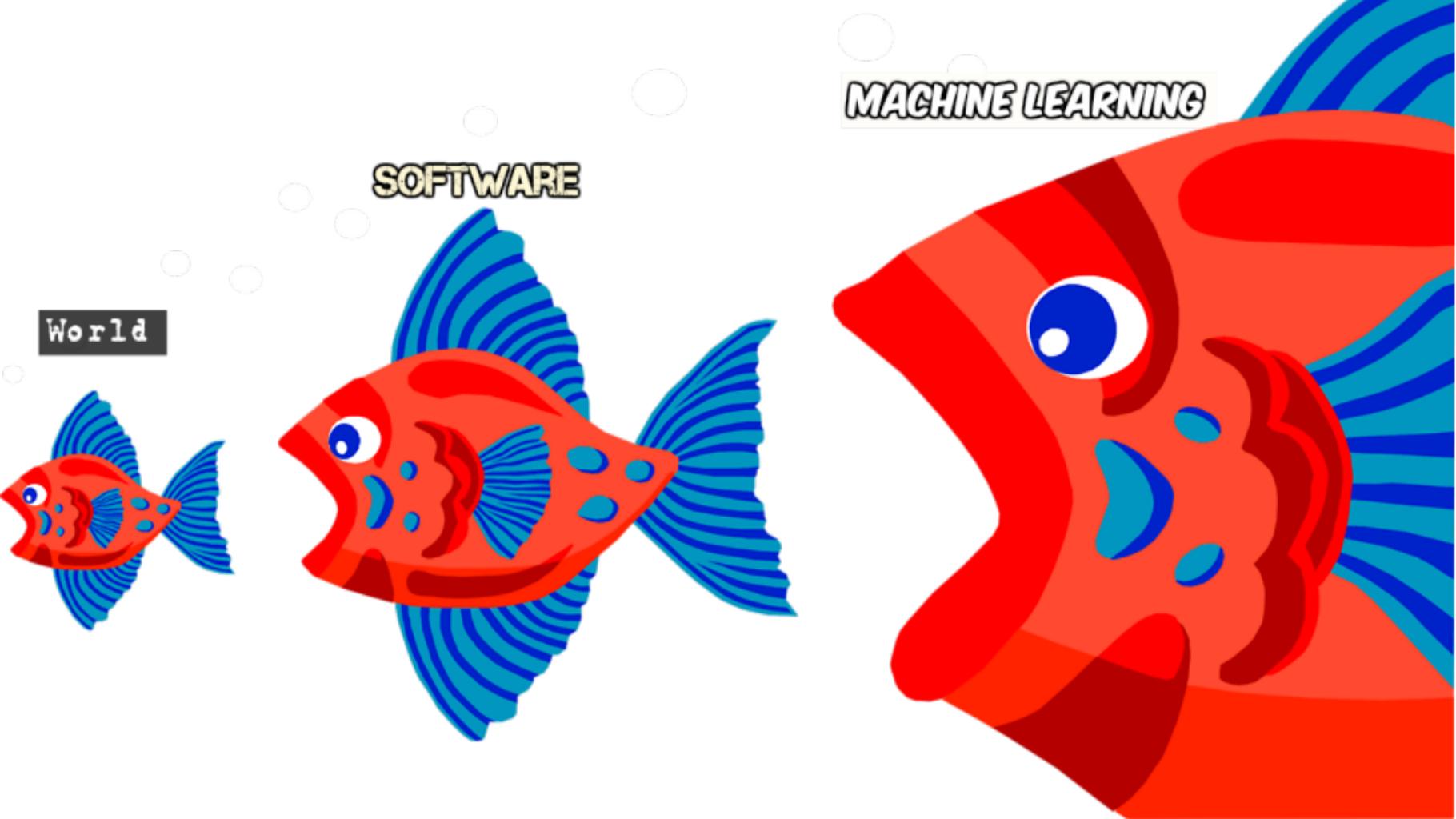


## **Traditional programming**



## **Machine Learning Approach**





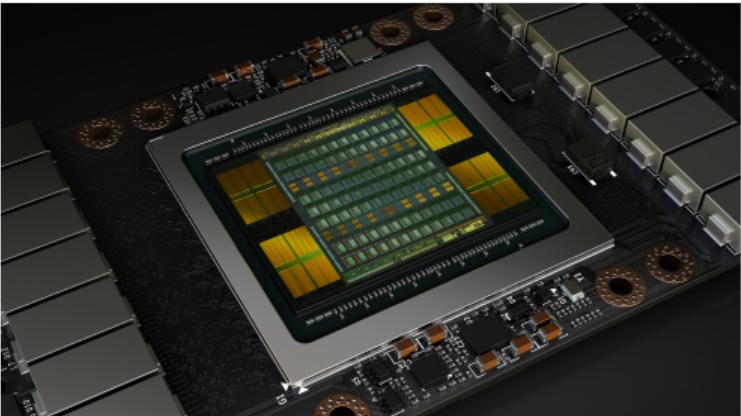
World

SOFTWARE

MACHINE LEARNING

# Waarom nu?

1. Snellere hardware
2. Betere algoritmes
3. Meer data
4. (Open source) frameworks



## Tesla V100 - NVIDIA

15,3 miljard transistoren

5120 CUDA cores

672 tensor cores

120 TFLOPS peak performance

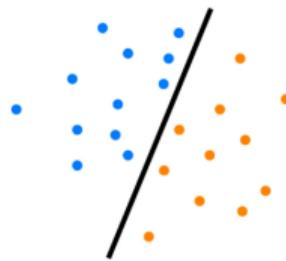
## Hoe leren uit data?

---

# Leeralgoritmes

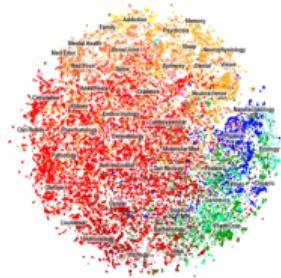
## Supervised

- Inputs met gewenste outputs zijn gegeven.
  - Task driven.



## Unsupervised

- De gewenste outputs zijn niet gegeven.
  - Data driven (clustering).



## Reinforcement

- Beslissingsproces op basis van beloningen.
  - Algoritme leert te reageren op zijn omgeving.



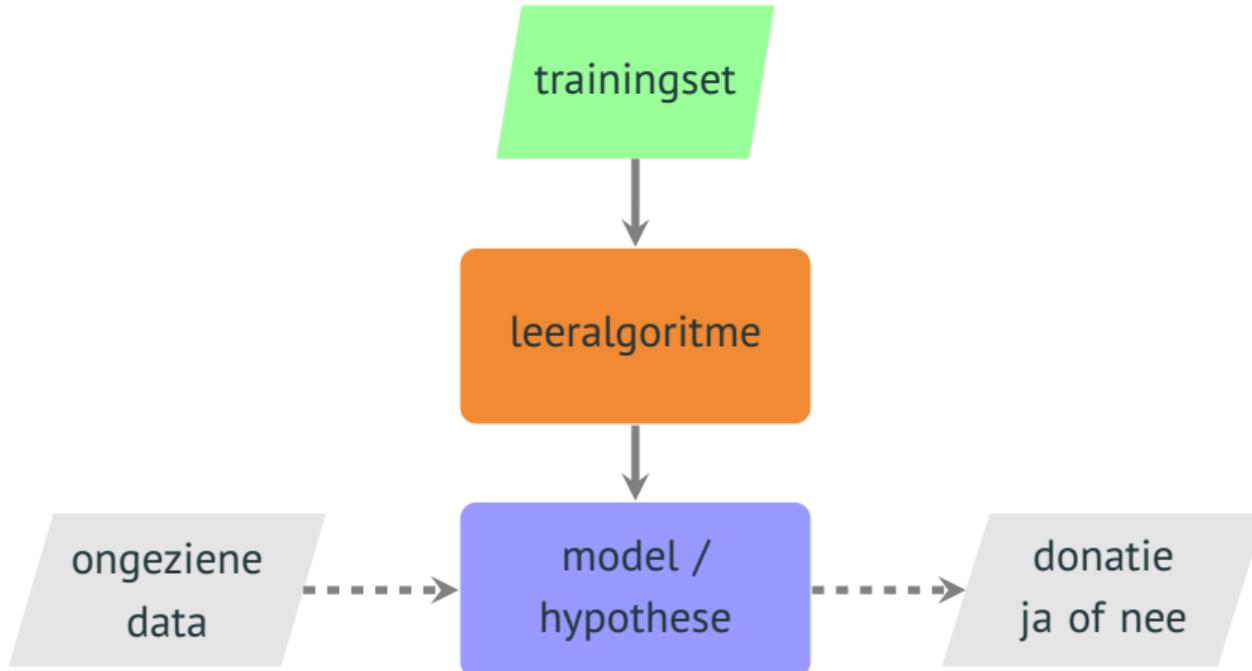
# Supervised learning

Is het leren op basis van een gelabelde trainingset. Het algoritme krijgt een **dataset voorzien van de juiste antwoorden**. Na training probeert het algoritme het juiste antwoord te vinden.



Person ID	Months since Last Donation	Number of Donations	Total Volume Donated (c.c.)	Months since First Donation	Donated the next month
0	619	2	50	12500	98
1	664	0	13	3250	28
2	441	1	16	4000	35
3	160	2	20	5000	45
4	358	1	24	6000	77
5	335	4	4	1000	4
6	47	2	7	1750	14
7	164	1	12	3000	35
8	736	5	46	11500	98
9	436	0	3	750	4

# Supervised learning



# Supervised learning - voorbeeld

Hoe stuurhoek bepalen bij een self-driving car?



# Supervised learning - voorbeeld

Hoe stuurhoek bepalen bij een self-driving car?

(infrarood) Camera's

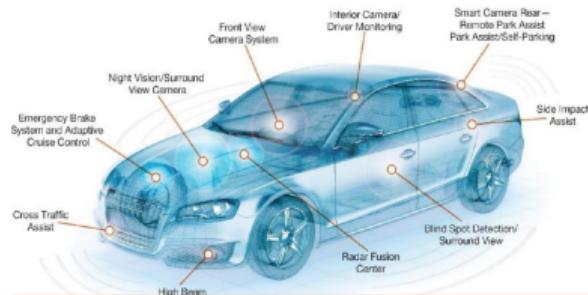
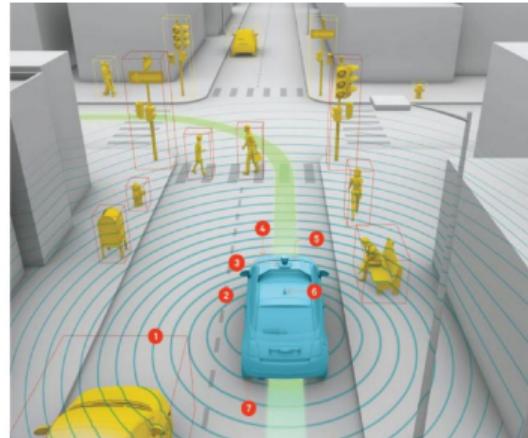
Stereo vision

Radar

LIDAR

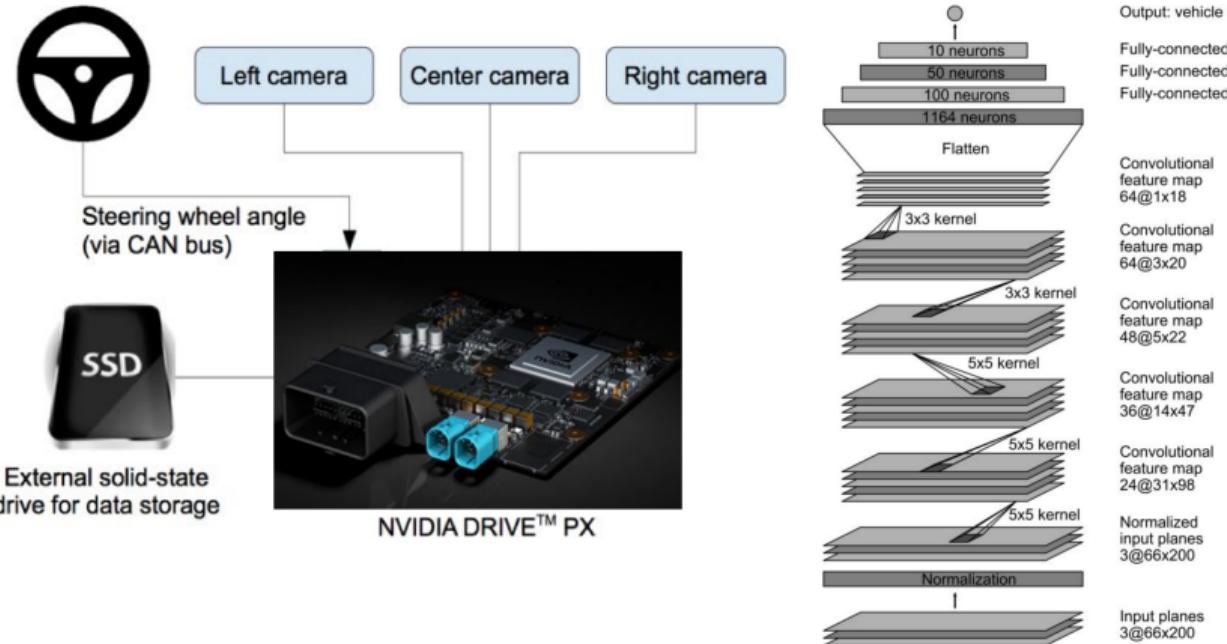
GPS

Audio



# Supervised learning - voorbeeld

Hoe stuurhoek bepalen bij een self-driving car?

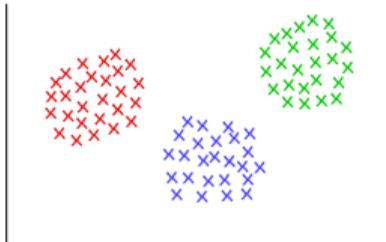


<https://selfdrivingcars.mit.edu/deeptesla/>

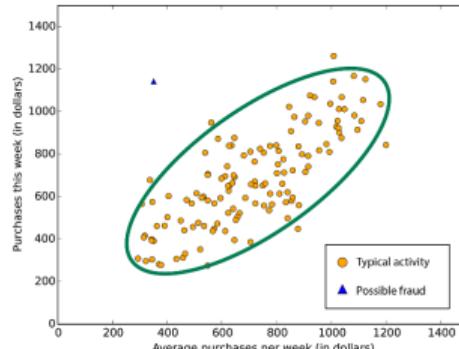
# Unsupervised learning

## Unsupervised learning toepassingen

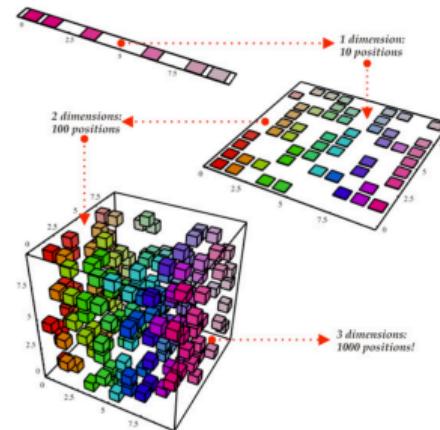
Clustering



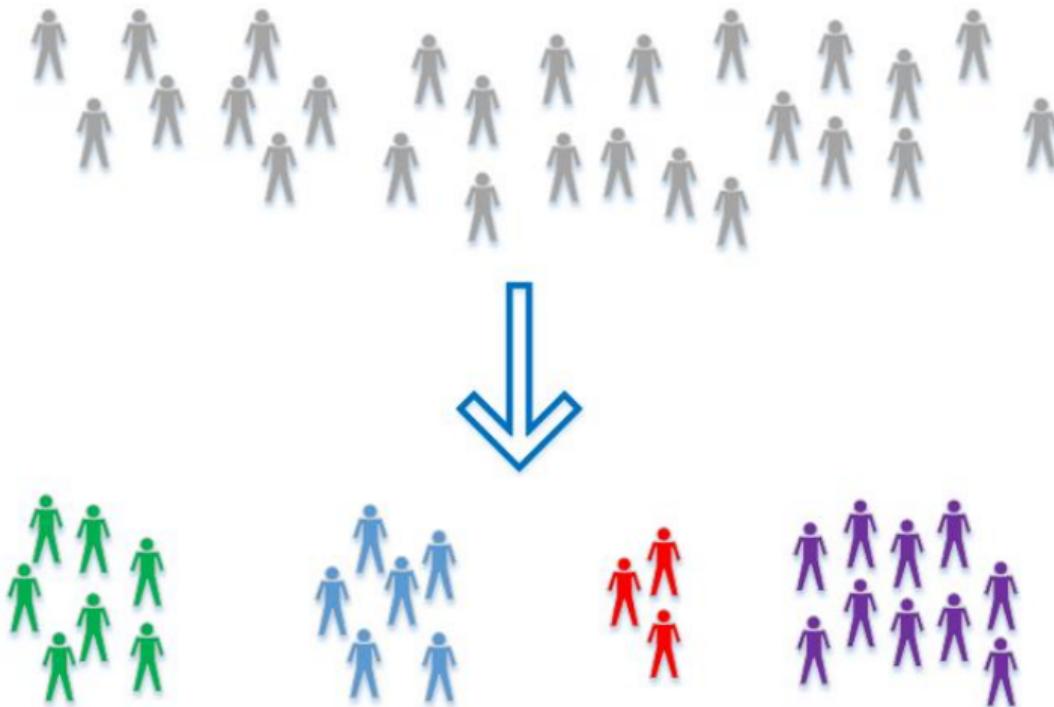
Anomaly detection



Dimensionality reduction



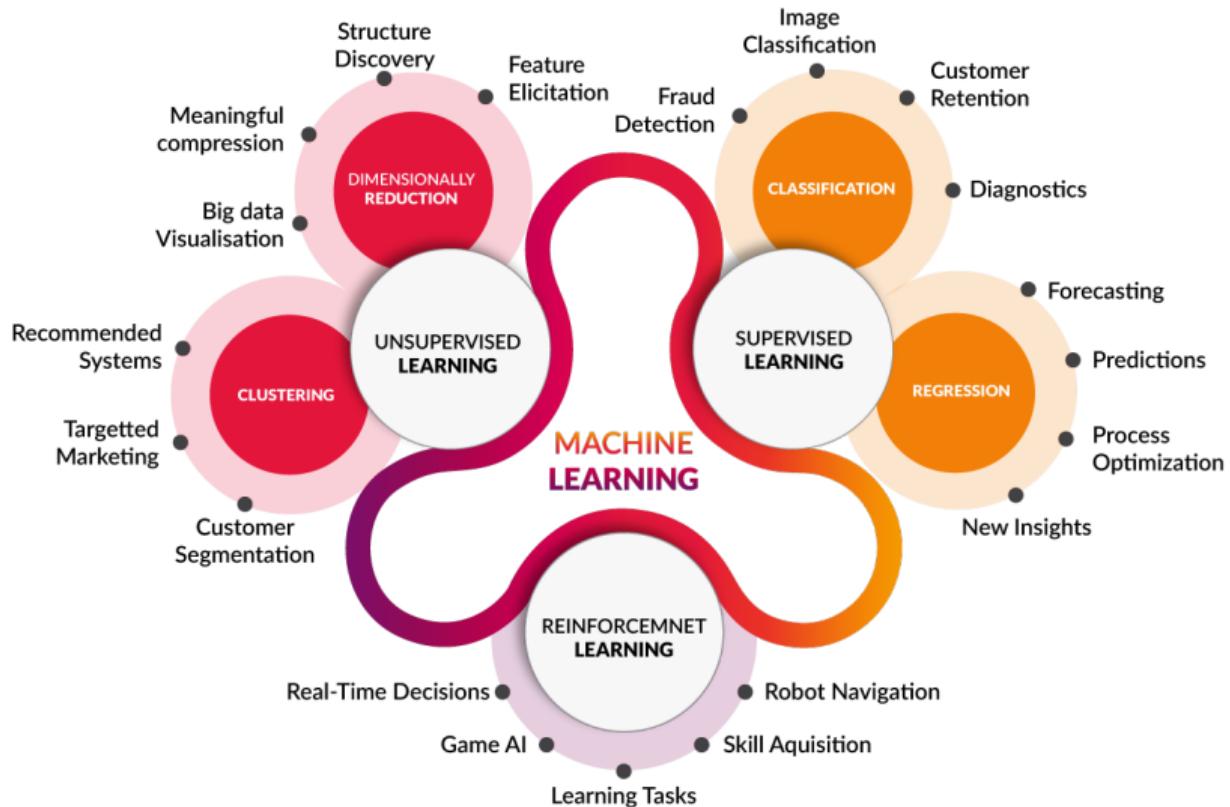
# Unsupervised learning



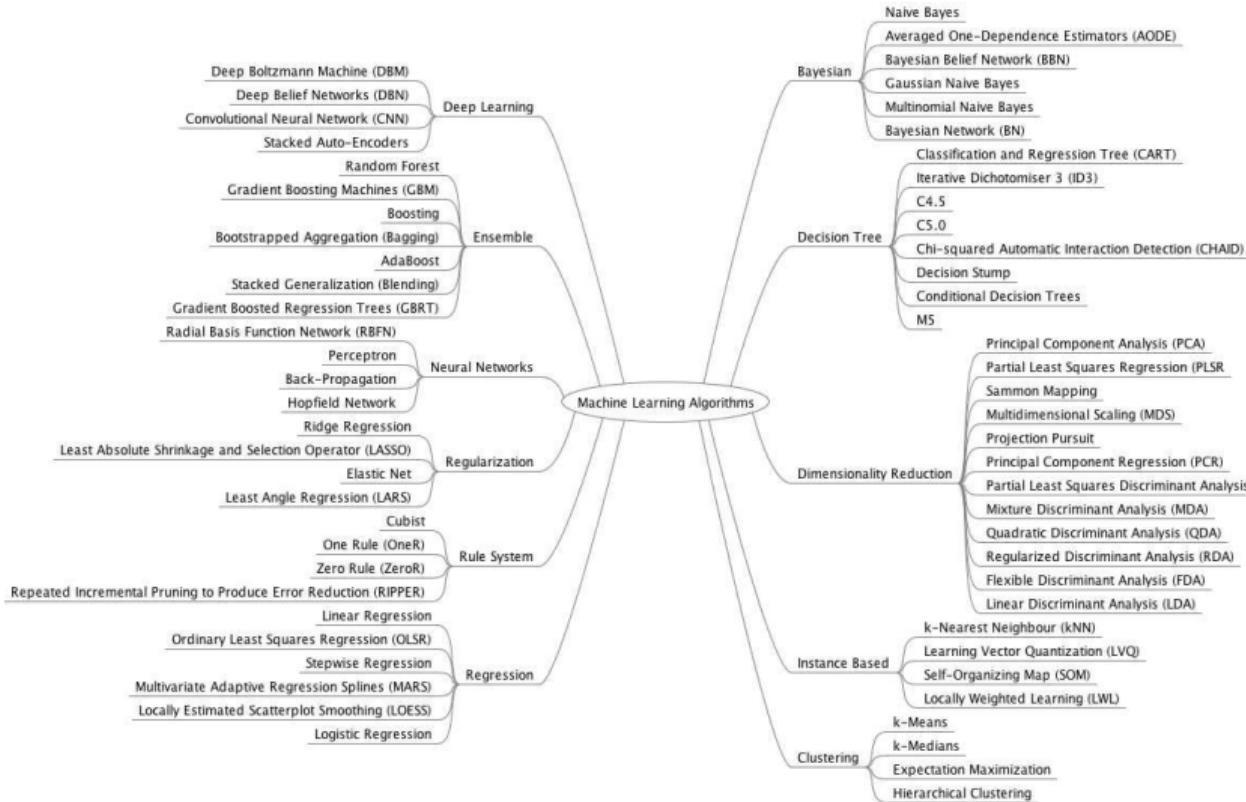
# Reinforcement learning



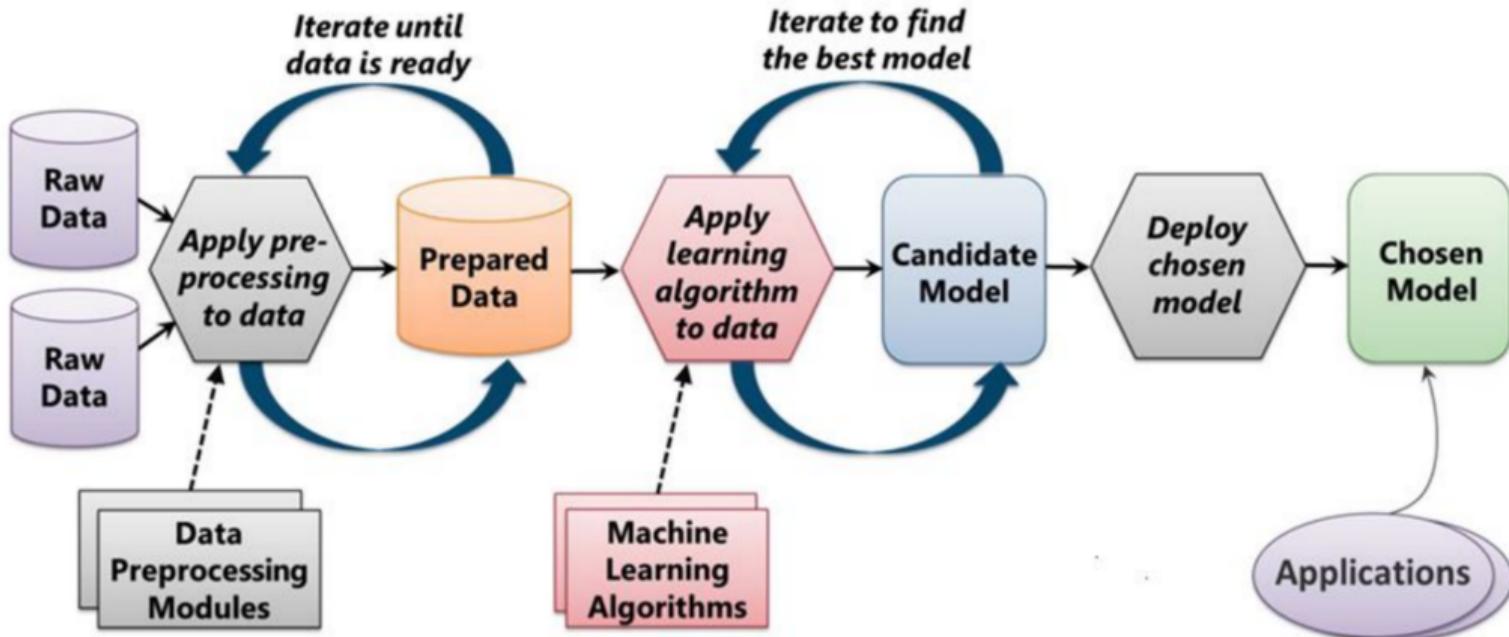
# Overzicht leeralgoritmes



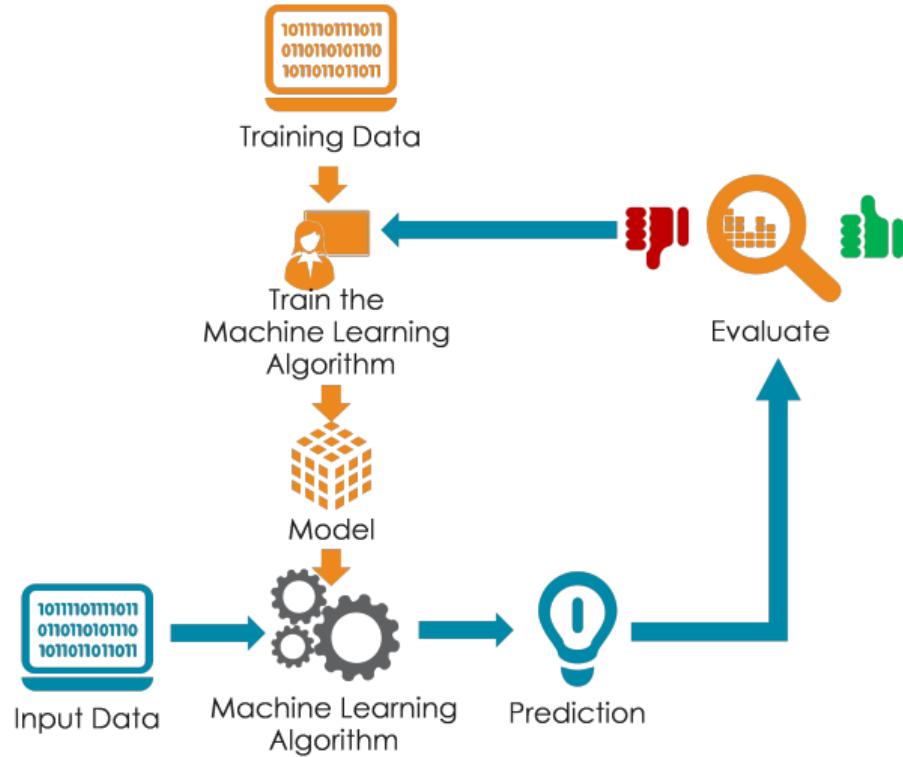
# Overzicht leeralgoritmes



# Aanpak van een machine learning project



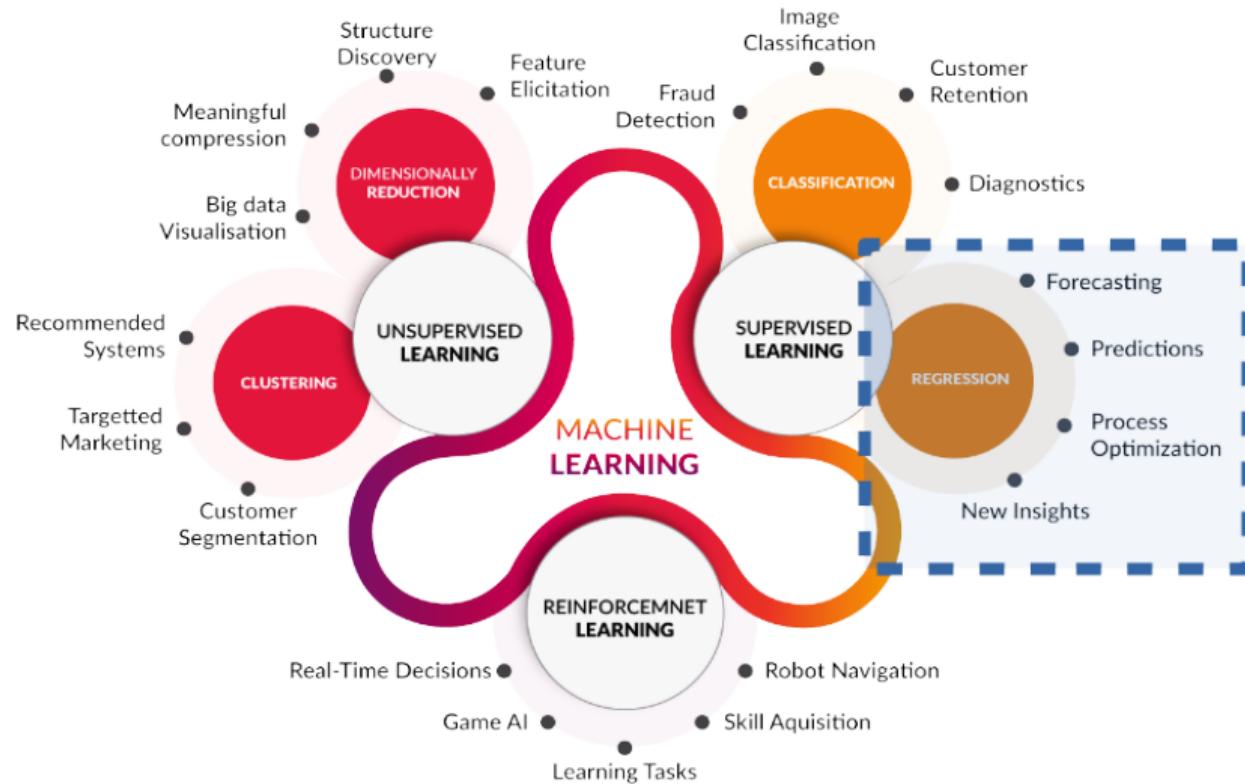
# Aanpak van een machine learning project



## Enkelvoudige lineaire regressie

---

# Lineaire regressie



# Lineaire regressie

	leeftijd	gewicht	bloeddruk
0	52	78	132
1	59	83	143
2	67	88	153
3	73	96	162
4	64	89	154
5	74	100	168
6	54	85	137
7	61	85	149
8	65	94	159
9	46	76	128
10	72	98	166

Voor spel de bloeddruk op basis van leeftijd en gewicht.

**features:** leeftijd en gewicht.

**target:** bloeddruk.

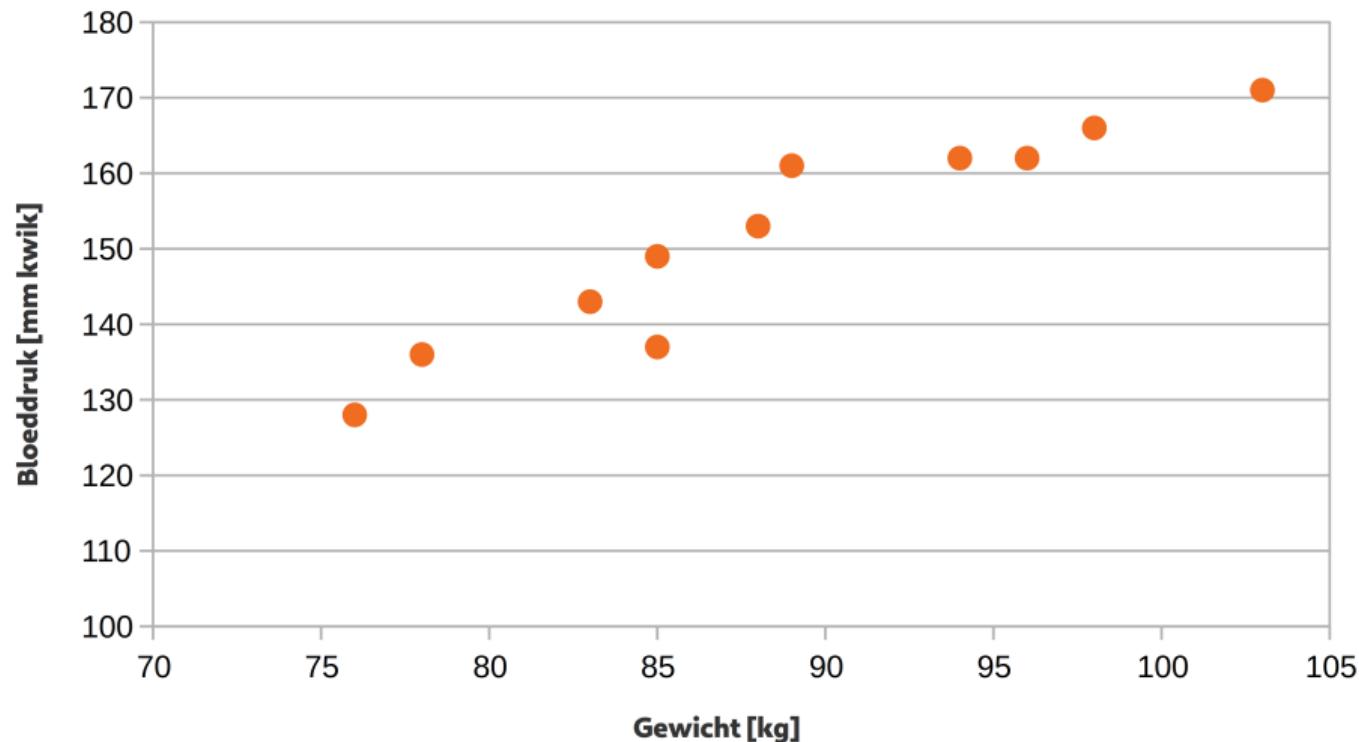
**trainingset** met 11 training examples.

## Regressie

Bij regressie is de output/target een (continue) variabele.

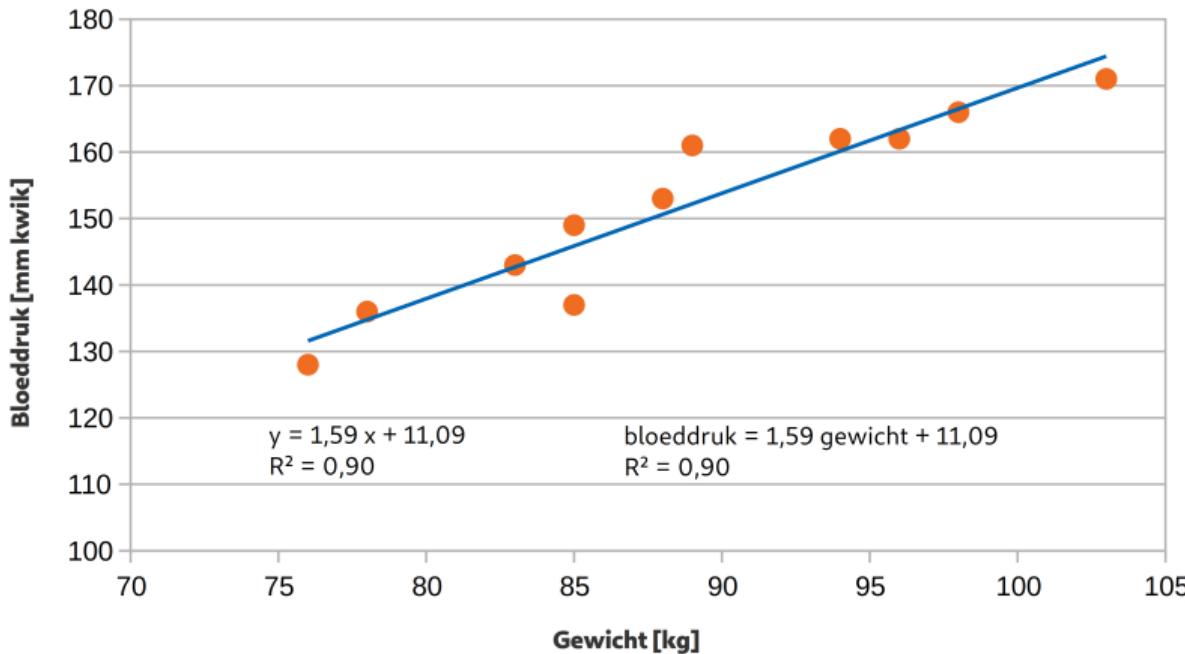
# Enkelvoudige lineaire regressie

Scatterplot



# Enkelvoudige lineaire regressie

Op zoek naar het verband tussen gewicht en bloeddruk



Lineaire trendlijn als model  $h_{\theta}(x)$  (= hypothese)

# Enkelvoudige lineaire regressie

Op zoek naar het verband tussen gewicht en bloeddruk

Het verband (model of hypothese)  $h_\theta(x)$  is van de vorm:

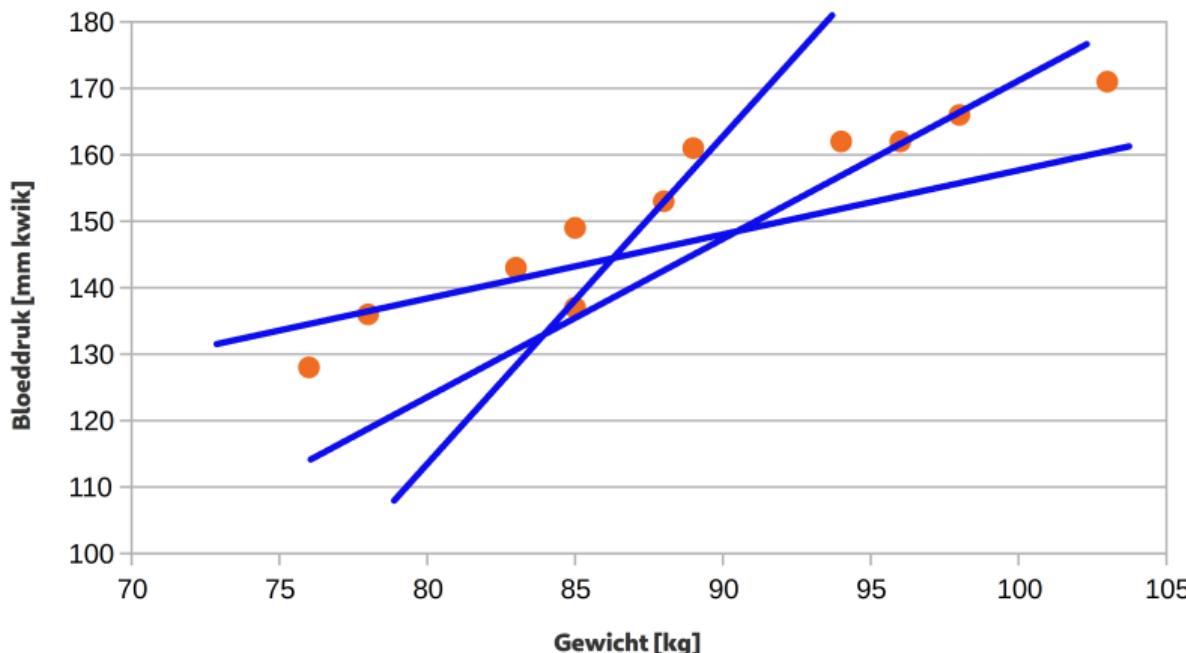
$$h_\theta(x) = \theta_0 + \theta_1 x$$

- Bepalen van de optimale waarden voor  $\theta_0$  en  $\theta_1$ .
  - $\theta_0$  = snijpunt met de y-as.
  - $\theta_1$  = helling van de rechte.
- De parameters  $\theta_i$  = **weights**.
- Het zoeken van het model / hypothese = **training / learning**.

# Enkelvoudige lineaire regressie

Op zoek naar het verband tussen gewicht en bloeddruk

Welke zijn de optimale waarden voor  $\theta_0$  en  $\theta_1$ ?

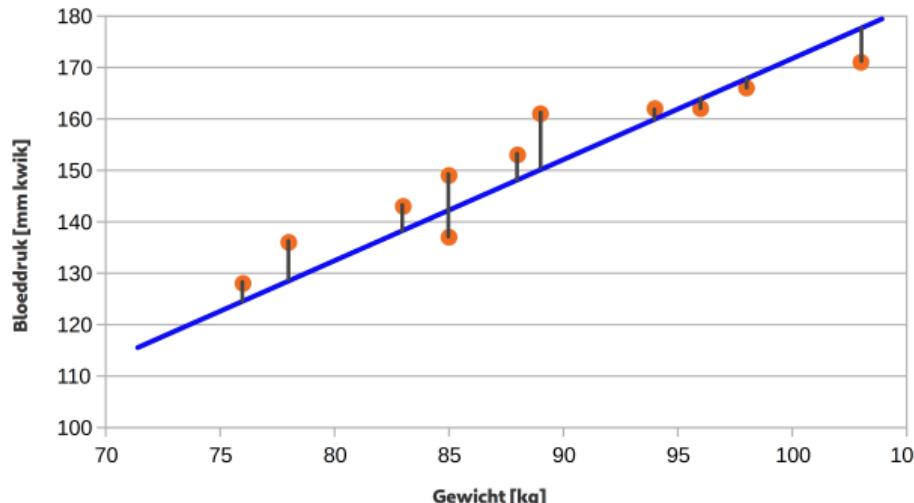


# Enkelvoudige lineaire regressie

Op zoek naar het verband tussen gewicht en bloeddruk

Minimaliseer de kostenfunctie  $J(\theta)$  via Least Mean Squares methode (LMS).

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

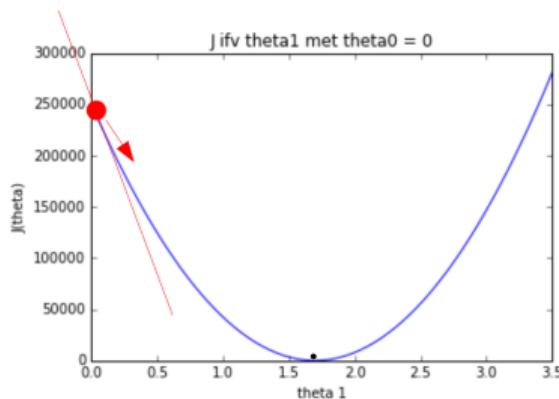


# Enkelvoudige lineaire regressie

## Gradient Descent (GDS)

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m \left( (\theta_1 x^{(i)} + \theta_0) - y^{(i)} \right)^2$$

Stel de parameters  $\theta_0$  en  $\theta_1$  voortdurend bij in een iteratief proces tot je de waarden voor  $\theta_0$  en  $\theta_1$  hebt gevonden die de kleinst mogelijke waarde voor  $J(\theta_0, \theta_1)$  opleveren.



$$J(\theta_0 = 0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m \left( (\theta_1 x^{(i)}) - y^{(i)} \right)^2$$

# Enkelvoudige lineaire regressie

## Gradient Descent (GDS) - algoritmisch

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m \left( (\theta_1 x^{(i)} + \theta_0) - y^{(i)} \right)^2$$

Bepaal de gradient naar  $\theta_0$  en  $\theta_1$

$$\frac{dJ(\theta)}{d\theta_0} = \frac{2}{2m} \sum_{i=1}^m \left( (\theta_1 x^{(i)} + \theta_0) - y^{(i)} \right)$$

$$\frac{dJ(\theta)}{d\theta_1} = \frac{2}{2m} \sum_{i=1}^m \left( (\theta_1 x^{(i)} + \theta_0) - y^{(i)} \right) x^{(i)}$$

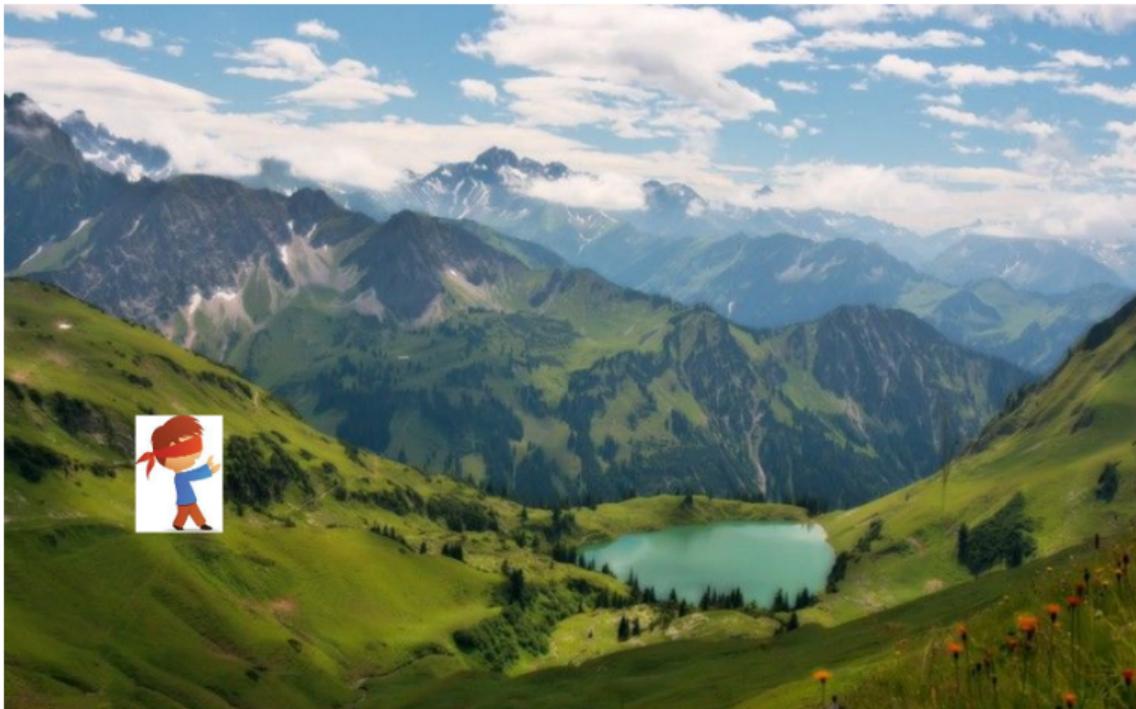
Update de parameters  $\theta_0$  en  $\theta_1$  volgens:

$$\theta_0 := \theta_0 - \eta \frac{dJ(\theta)}{d\theta_0} \quad \text{en} \quad \theta_1 := \theta_1 - \eta \frac{dJ(\theta)}{d\theta_1}$$

$\mu$  is de learning rate

# Enkelvoudige lineaire regressie

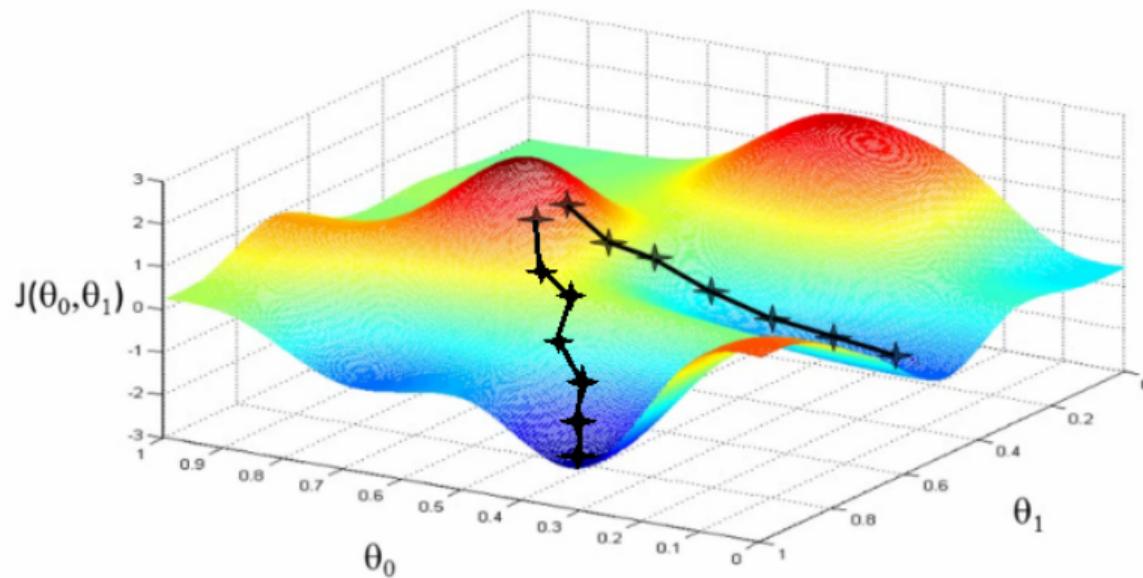
Gradient Descent (GDS)



# Enkelvoudige lineaire regressie

## Gradient Descent (GDS)

Lokaal versus globaal optimum

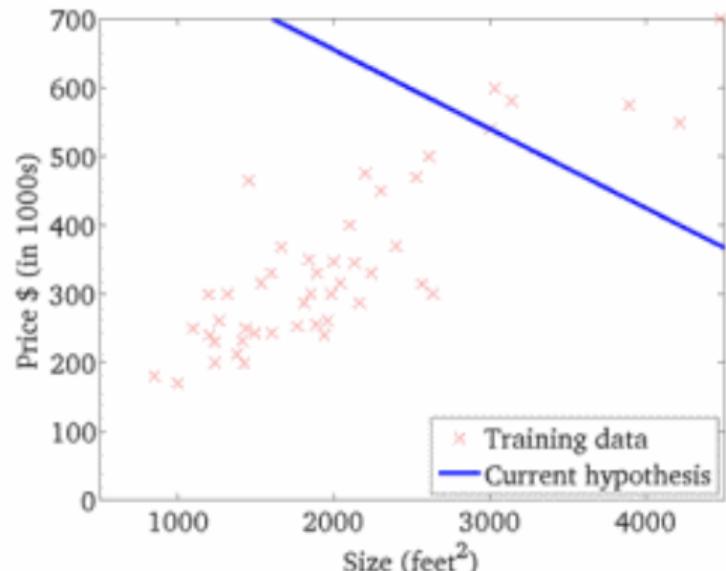


# Enkelvoudige lineaire regressie

## Gradient Descent (GDS)

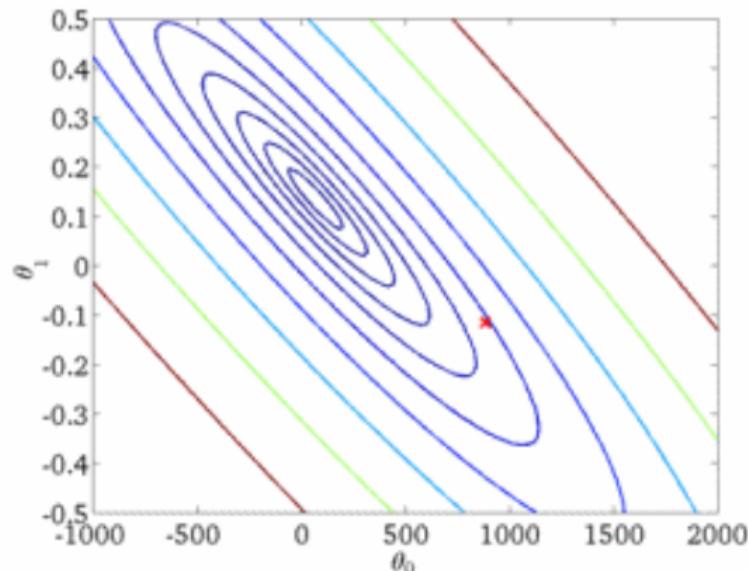
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )

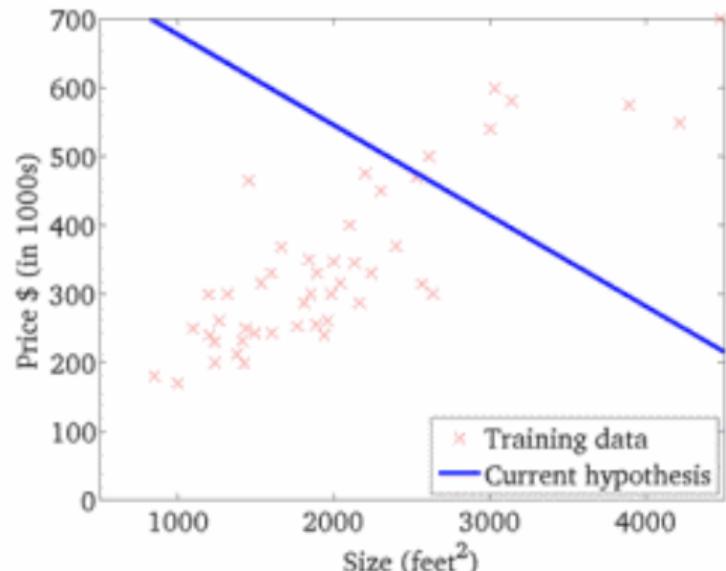


# Enkelvoudige lineaire regressie

## Gradient Descent (GDS)

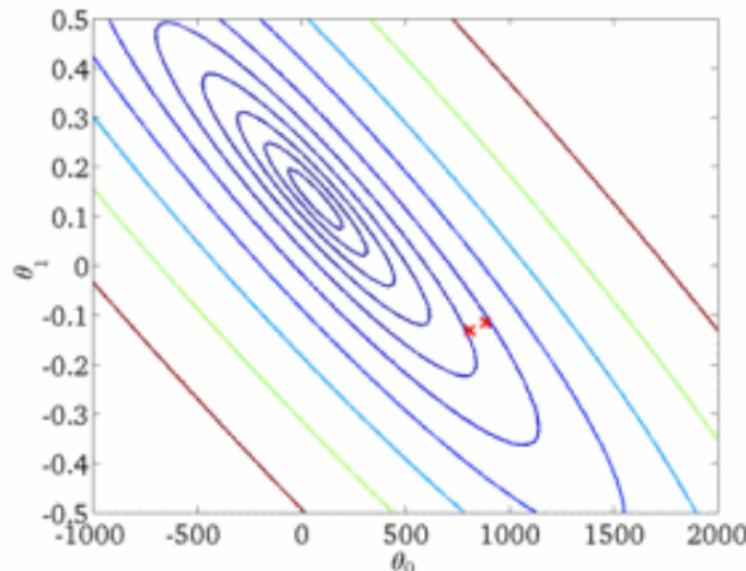
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )

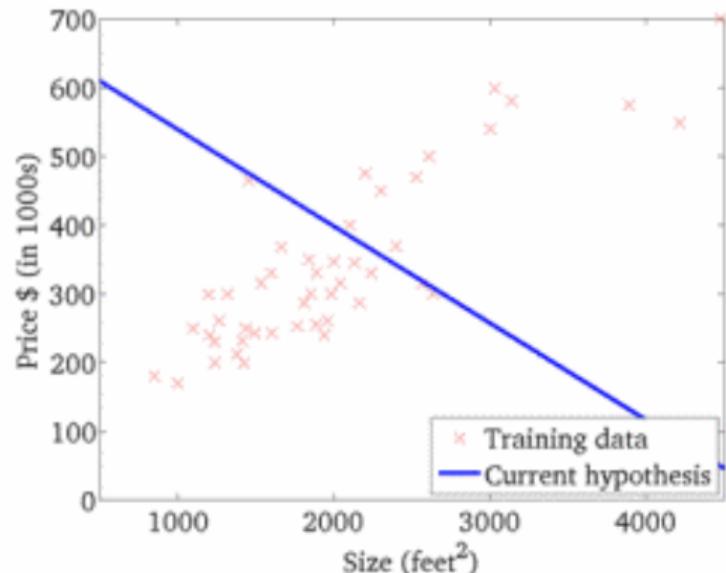


# Enkelvoudige lineaire regressie

## Gradient Descent (GDS)

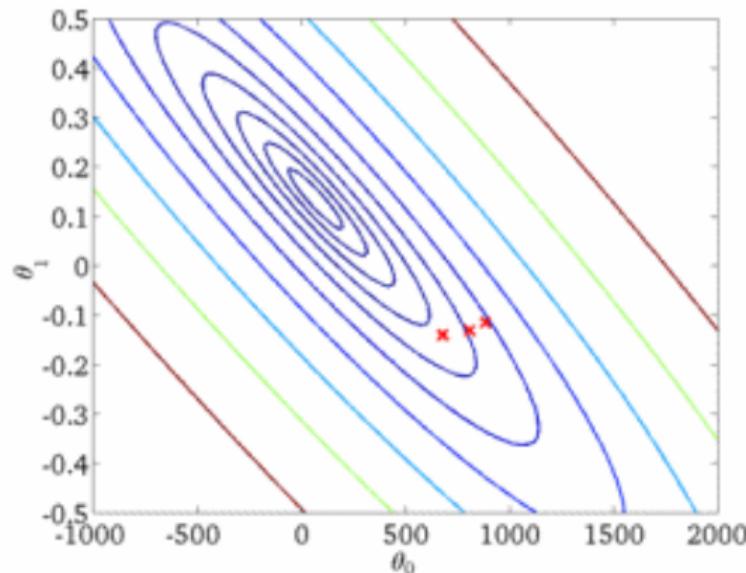
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )

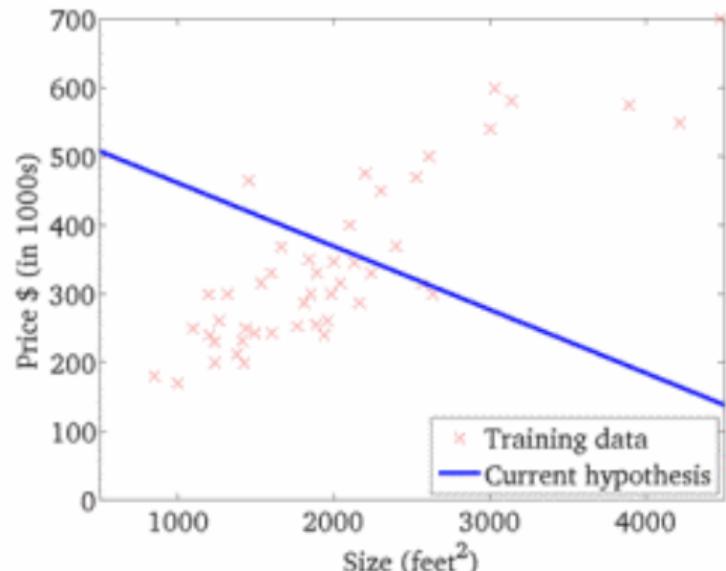


# Enkelvoudige lineaire regressie

## Gradient Descent (GDS)

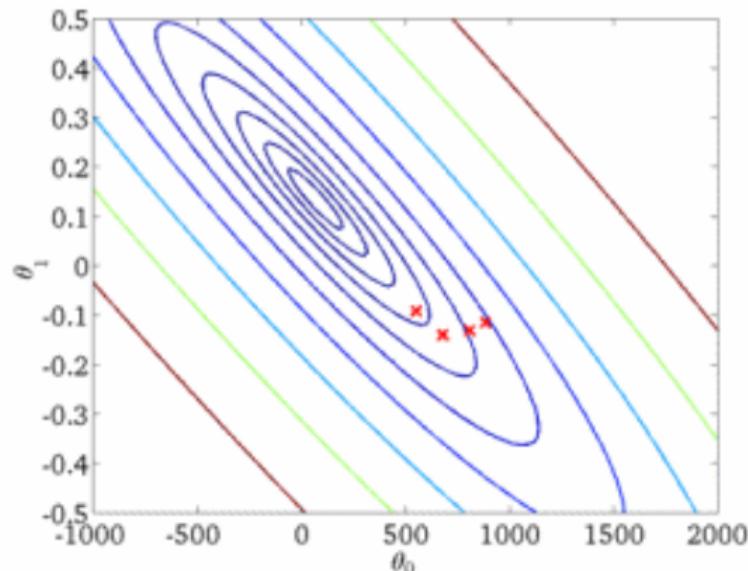
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )

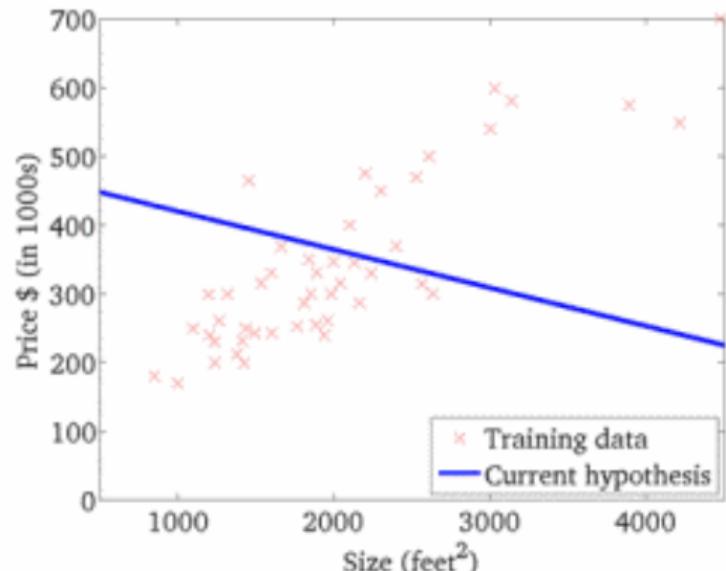


# Enkelvoudige lineaire regressie

## Gradient Descent (GDS)

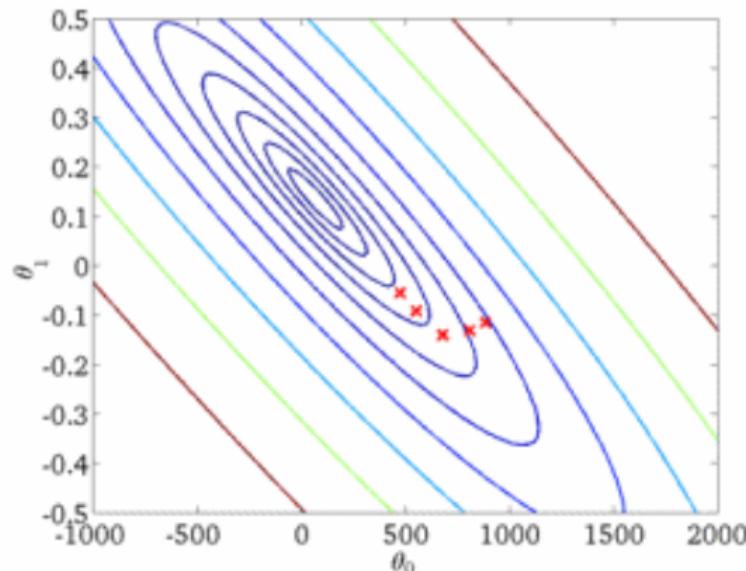
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )

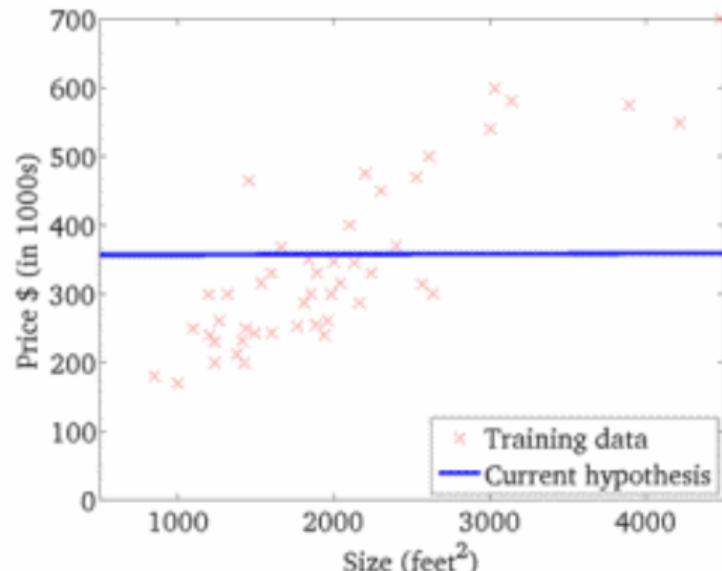


# Enkelvoudige lineaire regressie

## Gradient Descent (GDS)

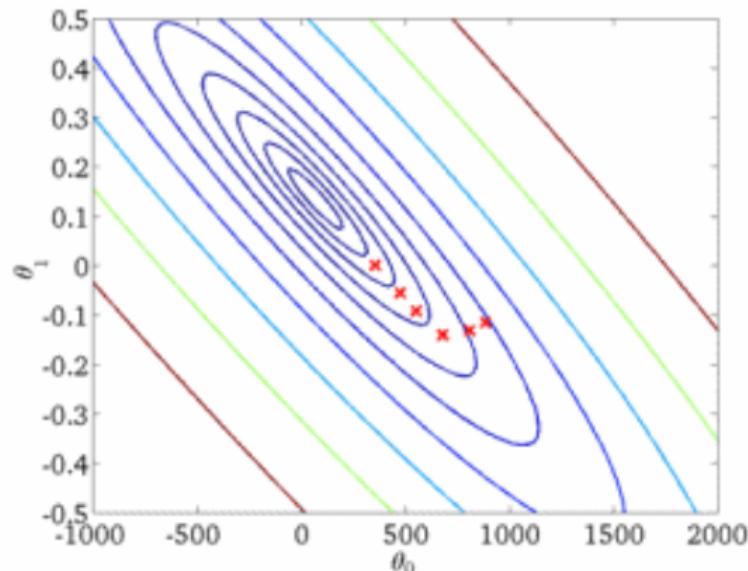
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )

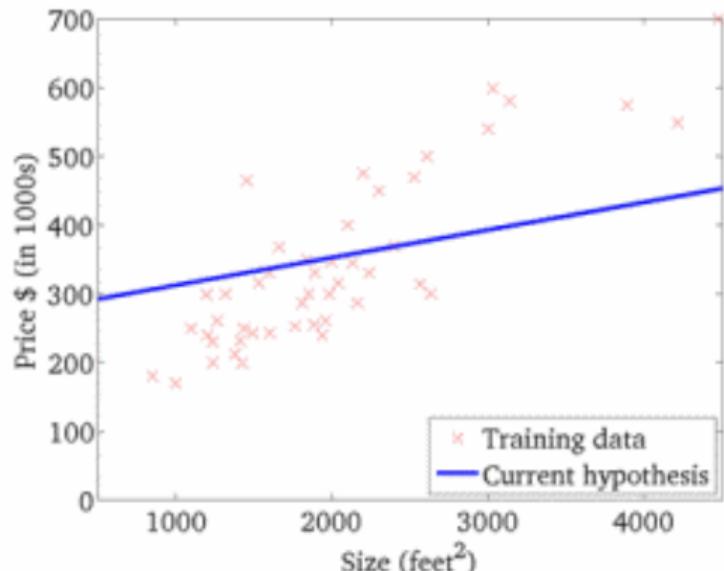


# Enkelvoudige lineaire regressie

## Gradient Descent (GDS)

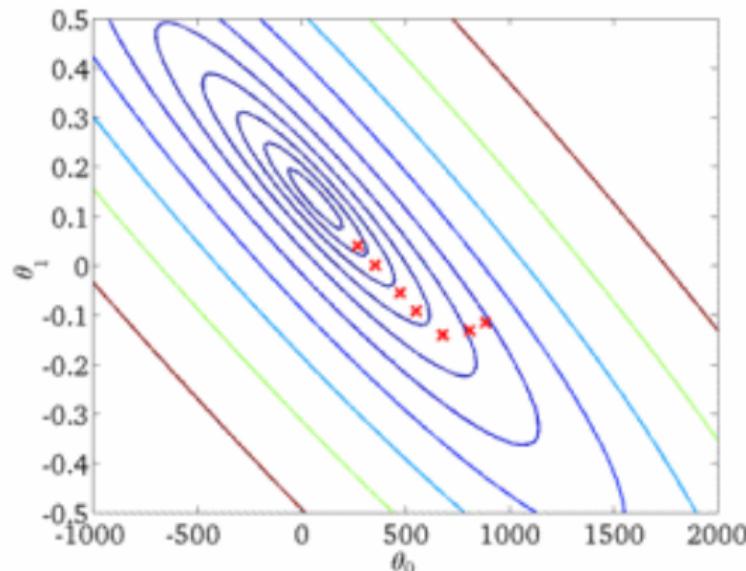
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )

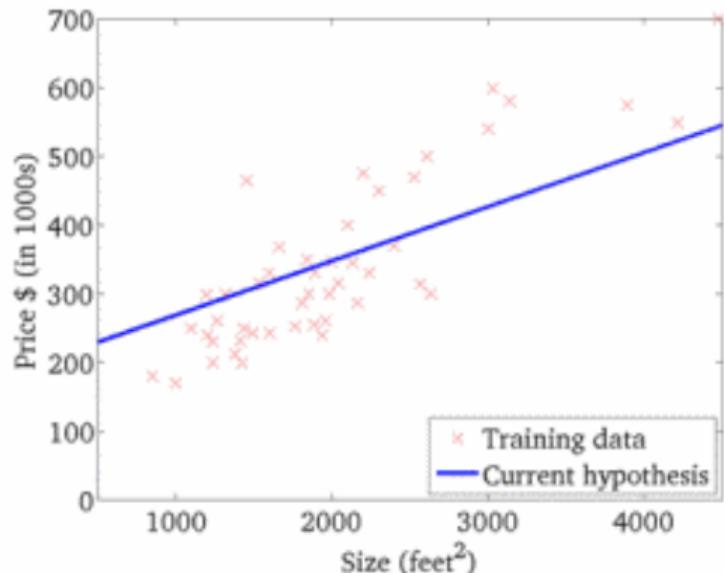


# Enkelvoudige lineaire regressie

## Gradient Descent (GDS)

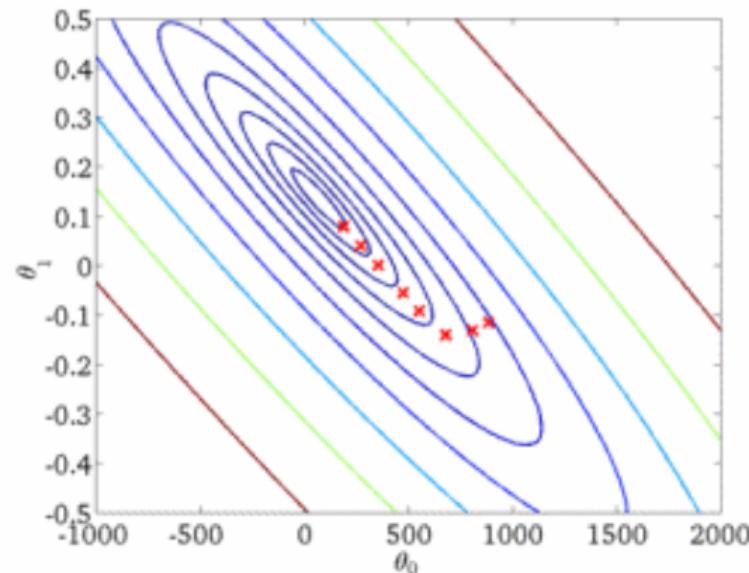
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )

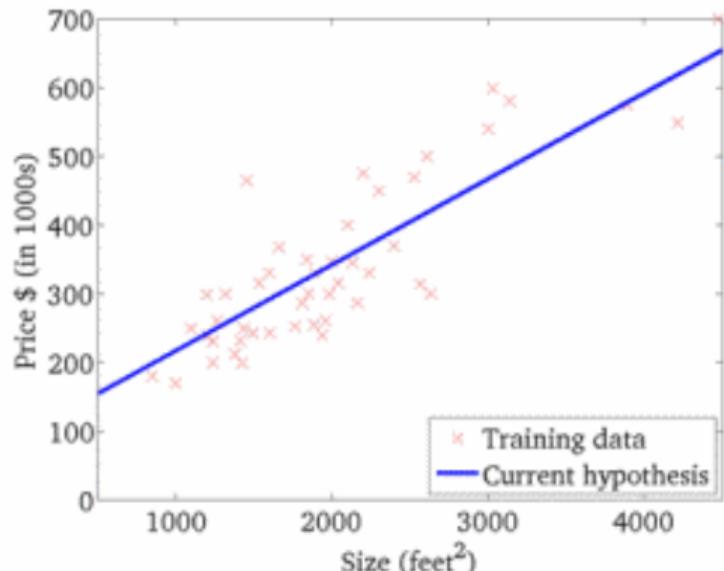


# Enkelvoudige lineaire regressie

## Gradient Descent (GDS)

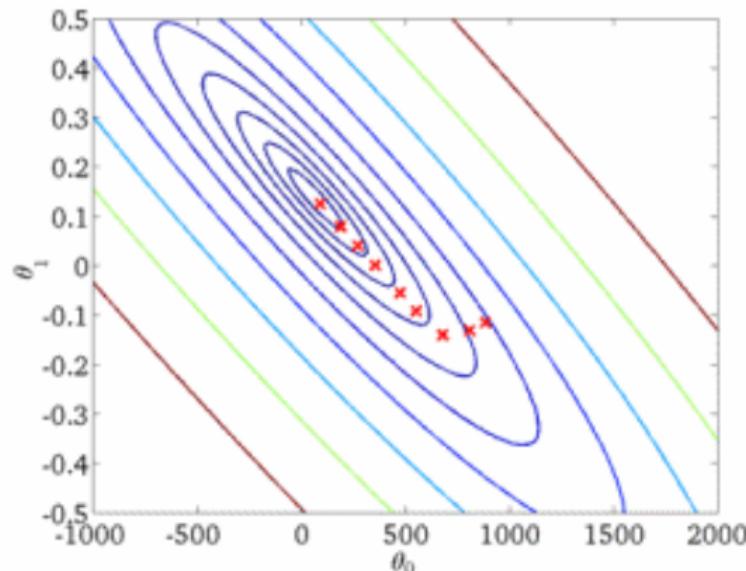
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )

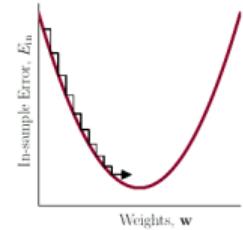


# Enkelvoudige lineaire regressie

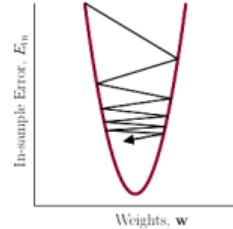
## Gradient Descent (GDS)

Invloed van de learning rate

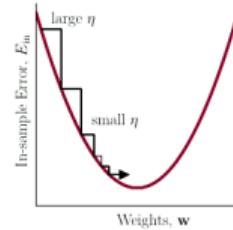
$\eta$  too small



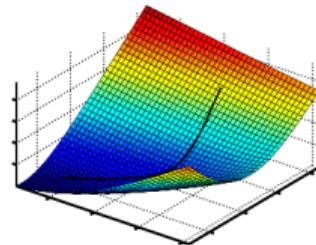
$\eta$  too large



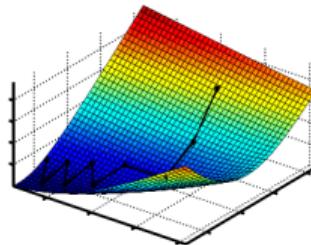
variable  $\eta_t$  – just right



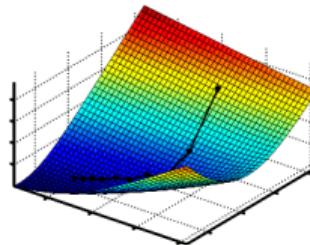
$\eta = 0.1$ ; 75 steps



$\eta = 2$ ; 10 steps



variable  $\eta_t$ ; 10 steps



## Meervoudige lineaire regressie

---

# Meervoudige lineaire regressie

## Definitie en voorbeelden

Bij meervoudige regressie (multiple regression) wordt het model/hypothese bepaald aan de hand van een trainingset met **meerdere features**.

- Bloeddruk wordt bepaald aan de hand van gewicht en leeftijd.
- Voorspel de kwaliteit van wijn op basis van de zuurtegraad, suikergehalte, chloriden, dichtheid, sulfaten, hoeveelheid alcohol, ...
- Voorspel het warmteverlies van een huis op basis van het type glas, muurisolatie, oriëntatie van het huis,...

$$h_{\theta}(x) = \theta_0 + \theta_1x_1 + \theta_2x_2 + \dots + \theta_nx_n$$

# Meervoudige lineaire regressie

## Voorbeeld - Voorspel de huisprijs op basis van onderstaande features

- CRIM - per capita crime rate by town.
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise).
- NOX - nitric oxides concentration (parts per 10 million).
- RM - average number of rooms per dwelling.
- AGE - proportion of owner-occupied units built prior to 1940.
- DIS - weighted distances to five Boston employment centres.
- RAD - index of accessibility to radial highways.
- TAX - full-value property-tax rate per \$10000.
- PTRATIO - pupil-teacher ratio by town.
- B -  $1000(B_k - 0.63)^2$  where  $B_k$  is the proportion of blacks by town.
- LSTAT % lower status of the population.
- Price - Median value of owner-occupied homes in 1000's.

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PT	B	LSTAT	Price
0	0.00632	18.0	2.31	0	0.538	6.575	65.199997	4.0900	1	296	15.300000	396.899994	4.98	504.000000
1	0.02731	0.0	7.07	0	0.469	6.421	78.900002	4.9671	2	242	17.799999	396.899994	9.14	453.600008
2	0.02729	0.0	7.07	0	0.469	7.185	61.099998	4.9671	2	242	17.799999	392.829987	4.03	728.700016
3	0.03237	0.0	2.18	0	0.458	6.998	45.799999	6.0622	3	222	18.700001	394.630005	2.94	701.400032
4	0.06905	0.0	2.18	0	0.458	7.147	54.200001	6.0622	3	222	18.700001	396.899994	5.33	760.200016

# Statistische vooranalyse

## Consistentie van de dataset

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PT	B	LSTAT	Price
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	4.673727	11.363636	11.136779	0.069170	25.892968	87.231275	68.574901	11.445487	9.549407	408.237154	18.455534	356.674030	12.653063	473.188934
std	25.223207	23.322453	6.860353	0.253994	132.093200	747.124382	28.148862	172.108941	8.707259	168.537116	2.164946	91.294863	7.141062	193.139186
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	105.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.888000	45.025000	2.100175	4.000000	279.000000	17.400000	375.377487	6.950000	357.525002
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.210000	77.500000	3.207450	5.000000	330.000000	19.050000	391.440002	11.360000	445.200016
75%	3.689388	12.500000	18.100000	0.000000	0.624000	6.630750	94.074999	5.213925	24.000000	666.000000	20.200001	396.225006	16.954999	525.000000
max	537.000000	100.000000	27.740000	1.000000	713.000000	8375.000000	100.000000	3875.000000	24.000000	711.000000	22.000000	396.899994	37.970001	1050.000000

- Volledigheid van de dataset.
- Inconsistenties.
- Spreiding van de gegevens.

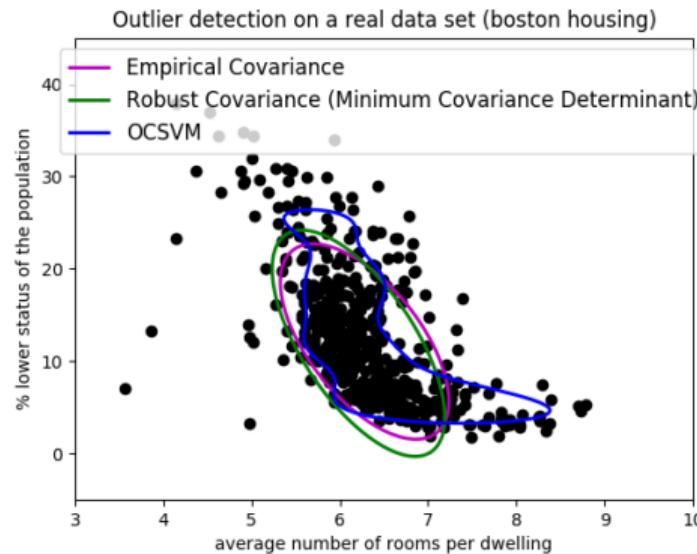
Verwijder de CHAS kolom:

```
dataset.drop( 'CHAS' , axis=1, inplace=True )
```

# Statistische vooranalyse

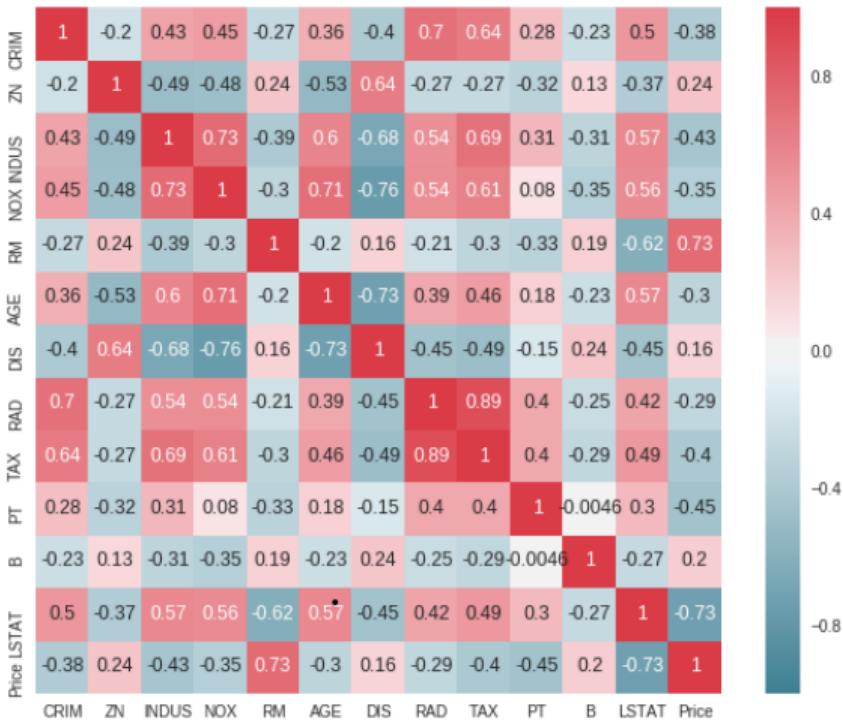
## Uitschieters

- Vinden en verwijderen van extreme waarden/samples.
- Geavanceerde technieken (zie later bij clustering).



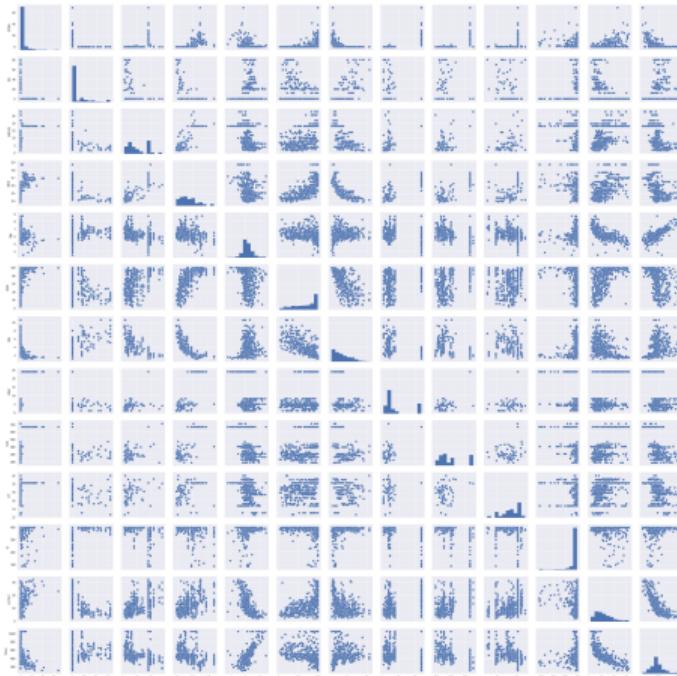
# Statistische vooranalyse

## Onderlinge correlaties - heatmap



# Statistische vooranalyse

## Onderlinge correlaties - pairplot



# Features en targets

## Dataset opsplitsen in features en targets

CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PT	B	LSTAT	Price
0.00632	18.0	2.31	0.538	6.575	65.199997	4.0900	1	296	15.300000	396.899994	4.98	504.000000
0.02731	0.0	7.07	0.469	6.421	78.900002	4.9671	2	242	17.799999	396.899994	9.14	453.600008
0.02729	0.0	7.07	0.469	7.185	61.099998	4.9671	2	242	17.799999	392.829987	4.03	728.700016
0.03237	0.0	2.18	0.458	6.998	45.799999	6.0622	3	222	18.700001	394.630005	2.94	701.400032
0.06905	0.0	2.18	0.458	7.147	54.200001	6.0622	3	222	18.700001	396.899994	5.33	760.200016

features X

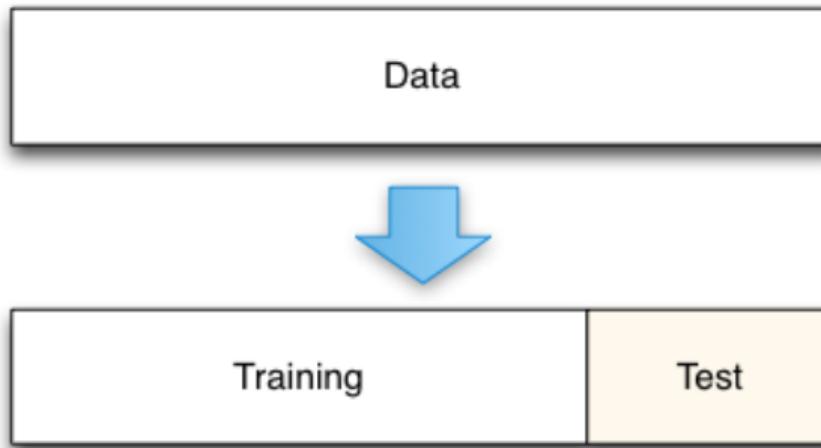
targets y

```
features = list(dataset.columns[:dataset.columns.size - 1])
X = dataset[features].values
y= dataset['Price'].values

# Een alternatief voor het opsplitsen in training en test set:
X = dataset.ix[:,0:dataset.columns.size - 1].copy()
y = dataset.ix[:,dataset.columns.size - 1:dataset.columns.size ].copy()
```

# Trainen van het model

Dataset opsplitsen in training en test set



Belangrijk om eerst te randomiseren.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,  
random_state=0)
```

# Trainen van het model

## Initialiseren en trainen van het regressiemodel

```
lregmodel = linear_model.LinearRegression()  
lregmodel.fit(X_train,y_train)
```

Model:

```
print('coeffs:',lregmodel.coef_)  
print('intercept', lregmodel.intercept_)  
  
>> coeffs: [-3.56141289e+00 4.05479295e-01 8.14080284e-01 -2.70514977e+02  
             8.96450415e+01 -3.02997261e-01 -2.77339444e+01  
             7.47151897e+00 -2.92233040e-01 -1.61741146e+01 7.62044683e-02  
             -1.17962045e+01]  
  
>> intercept: 650.652022517
```

$$\begin{aligned} Price = & -3.56 \times CRIM + 0.41 \times ZN + 0.81 \times INDUS - 270.51 \times NOX + 89.65 \times RM \\ & - 0.30 \times AGE - 27.74 \times DIS + 7.47 \times RAD - 0.29 \times TAX - 16,17 \times PT \\ & + 0.08 \times B - 11.80 \times LSTAT + 650.65 \end{aligned}$$

# Testen van het model

## Voorspellingen maken

Voorspel de prijs van een huis waarbij

CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PT	B	LSTAT
0.11	0	12.03	0.57	6.80	89.30	2.39	1	273	21.00	393.45	6.48

```
house = np.array([0.11, 0, 12.03, 0.57, 6.80, 89.30, 2.39, 1,
                  273, 21.00, 393.45, 6.48])

price = lregmodel.predict(house.reshape(1,-1))

print('De prijs van het huis bedraagt: ', price)

>> De prijs van het huis bedraagt: 563.68335073
```

reshape(1,-1) maakt een rijvector

Werkelijke prijs = 562.00

# Testen van het model

Performantie en scores van het model: MAE (Mean Absolute Error)

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Is het **gemiddelde van de absolute waarden** van het verschil tussen de werkelijke waarden  $y_i$  en de voorspelde waarden  $\hat{y}_i$ .

```
from sklearn.metrics import mean_absolute_error
y_predicted = lregmodel.predict(X_test)
MAE = mean_absolute_error(y_test, y_predicted)
print('MAE = ', MAE)
```

```
>> MAE = 64.0090867586
```

# Testen van het model

## Performantie en scores van het model: MSE (Mean Squared Error)

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Is het **gemiddelde van de gekwadrateerde waarden** van het verschil tussen de werkelijke waarden  $y_i$  en de voorspelde waarden  $\hat{y}_i$ .

```
from sklearn.metrics import mean_squared_error
y_predicted = lregmodel.predict(X_test)
MSE = mean_squared_error(y_test, y_predicted)
print('MSE = ', MSE)

>> MSE = 7803.89332739
```

# Testen van het model

Performantie en scores van het model: Determinatiecoëfficiënt ( $R^2$ )

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Zegt hoeveel van de variabiliteit verklaard wordt door het model.

Bij perfecte voorspellingen is  $R^2 = 1$

Een negatieve waarde voor  $R^2$  betekent dat het model slechter scoort dan een horizontale lijn ( $y_i = \bar{y}$ ).

```
from sklearn.metrics import r2_score
y_predicted = lregmodel.predict(X_test)
r2 = r2_score(y_test, y_predicted)
print('r2 score = ', r2)
### alternatieve manier voor het bepalen van de r2 score
r2 = lregmodel.score(X_test, y_test)

>> r2 score =  0.754254234917
```

## Feature engineering

---

# Normalisatie

## Doel van normalisatie

Normalisatie zorgt ervoor dat de features op **dezelfde schaalverdeling** staan.

In ons voorbeeld van de huizenprijs:

	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PT	B	LSTAT
count	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000
mean	2.649013	9.988662	10.759479	0.547575	6.265785	67.632200	3.851895	8.478458	389.045351	18.387302	375.393853	12.165329
std	6.273766	19.941189	6.749778	0.112896	0.685393	27.997824	2.054524	8.000859	158.293650	2.164533	49.296266	6.632866
min	0.006320	0.000000	1.250000	0.385000	3.561000	2.900000	1.169100	1.000000	188.000000	12.600000	83.449997	1.730000
25%	0.081870	0.000000	5.190000	0.449000	5.877000	45.000000	2.122200	4.000000	277.000000	17.000000	377.730011	6.920000
50%	0.217190	0.000000	8.140000	0.524000	6.172000	74.500000	3.375100	5.000000	311.000000	18.700001	392.200012	10.740000
75%	1.656600	12.500000	18.100000	0.609000	6.590000	93.599998	5.231100	8.000000	432.000000	20.200001	396.899994	15.940000
max	67.920799	80.000000	27.740000	0.871000	8.780000	100.000000	10.710300	24.000000	711.000000	22.000000	396.899994	31.990000

NOX: 0.385  $\rightarrow$  0.871

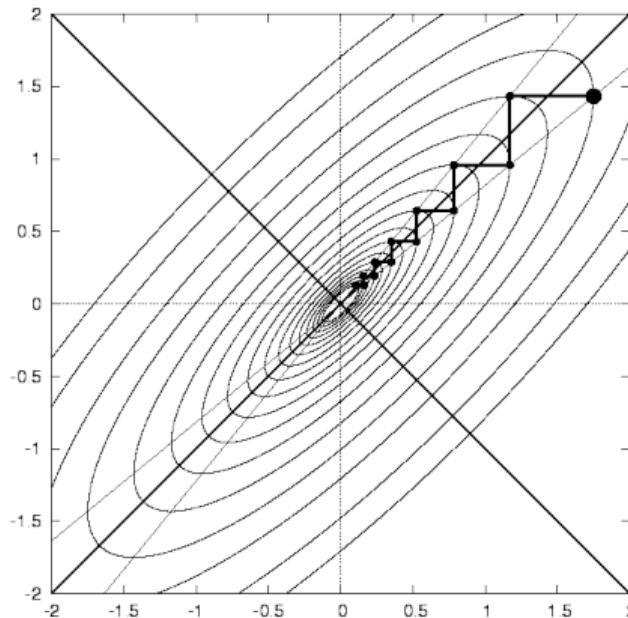
terwijl

TAX: 188  $\rightarrow$  711

# Normalisatie

## Doel van normalisatie

Gradient Descent convergeert minder snel als features op een verschillende schaalgrootte staan.

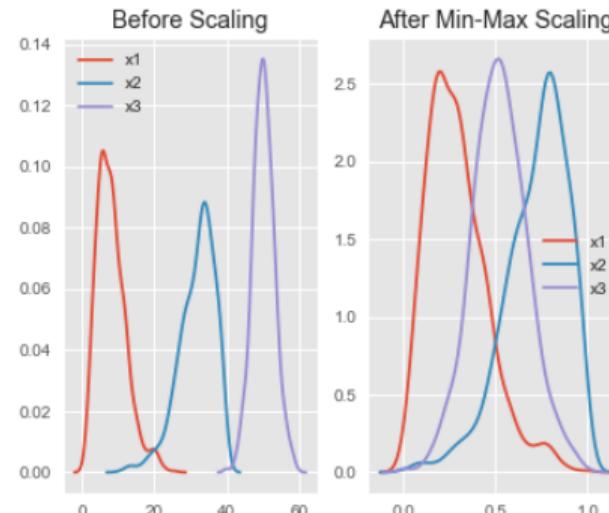


# Normalisatie

## MIN-MAX scaling

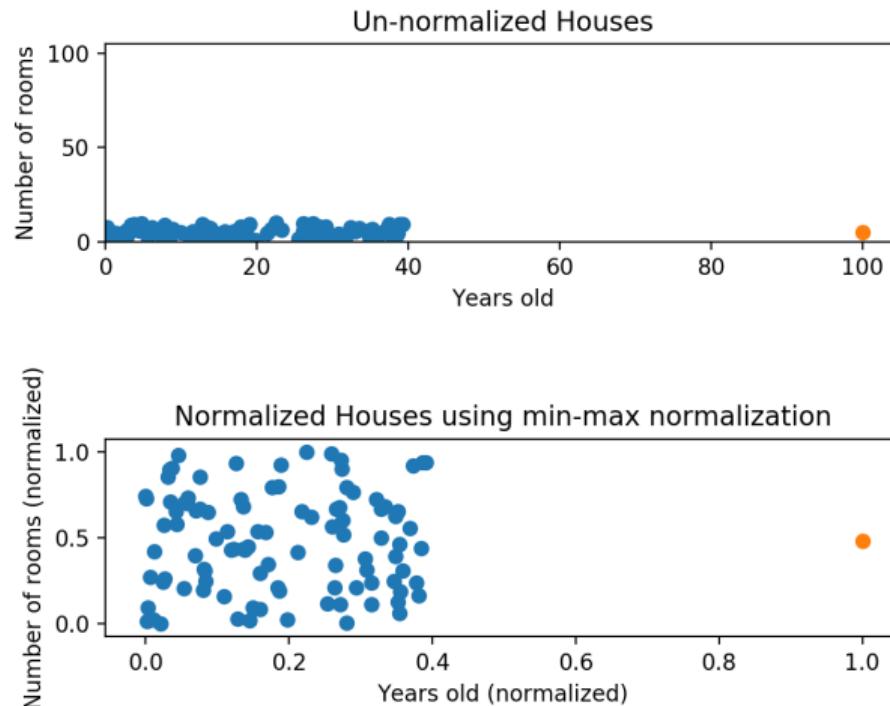
$$x_{S_i} = \frac{x_i - \text{Min}(x)}{\text{Max}(x) - \text{Min}(x)}$$

- Schaalt alle features tussen 0 en 1.
- Werkt goed bij niet Gaussiaanse distributies en bij kleine variantie.
- De scheefheid (skew) blijft bewaard.
- Gevoelig voor uitschieters.



# Normalisatie

## MIN-MAX scaling



# Normalisatie

## MIN-MAX scaling

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

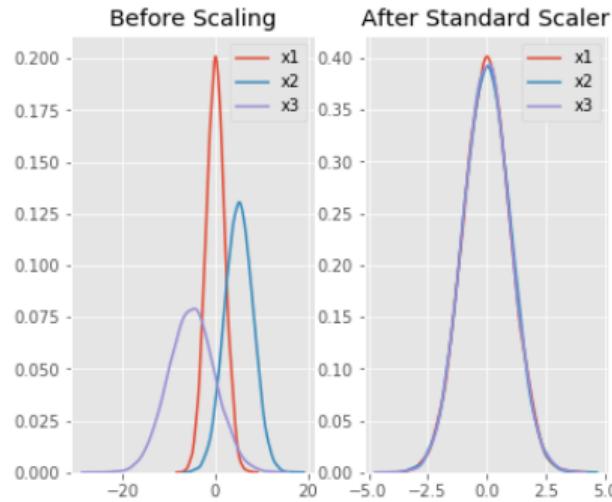
# alternatief
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

# Normalisatie

## Standard scaling

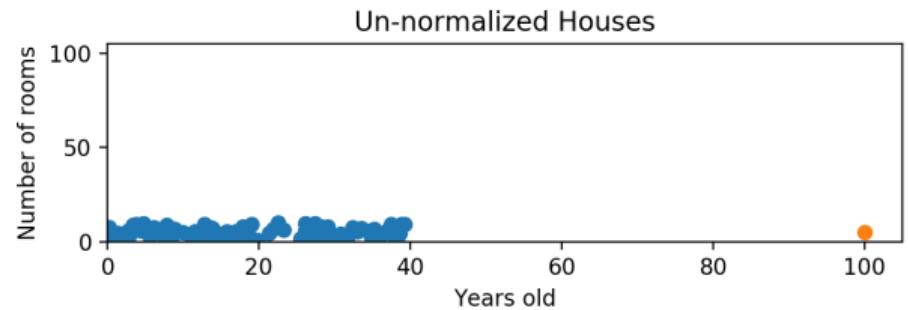
$$x_{S_i} = \frac{x_i - \text{mean}(x)}{\text{stdev}(x)}$$

- Geschaalde features: gemiddelde = 0 en standaardafwijking = 1.
- Geschaalde features schommelen rond 0 (soms nodig bij deep learning).
- Vervormt geen relatieve afstanden tussen de feature waarden.
- Kan beter overweg met uitschieters.
- Garandeert geen genormaliseerde data op exact dezelfde schaal.



# Normalisatie

## Standard scaling



# Normalisatie

## Standard scaling

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

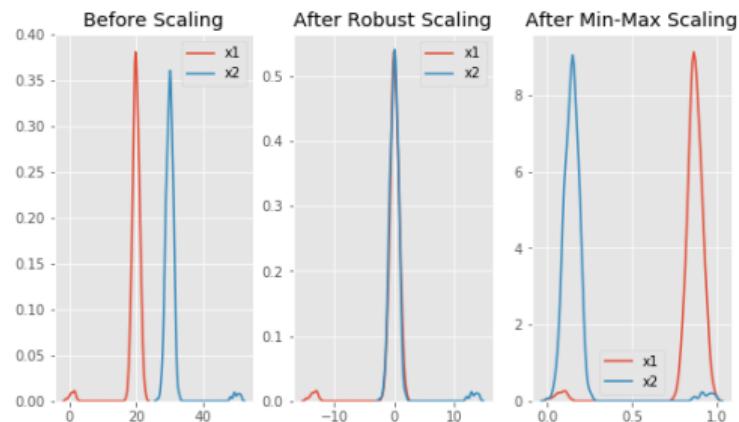
# alternatief
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

# Normalisatie

## Robust scaling

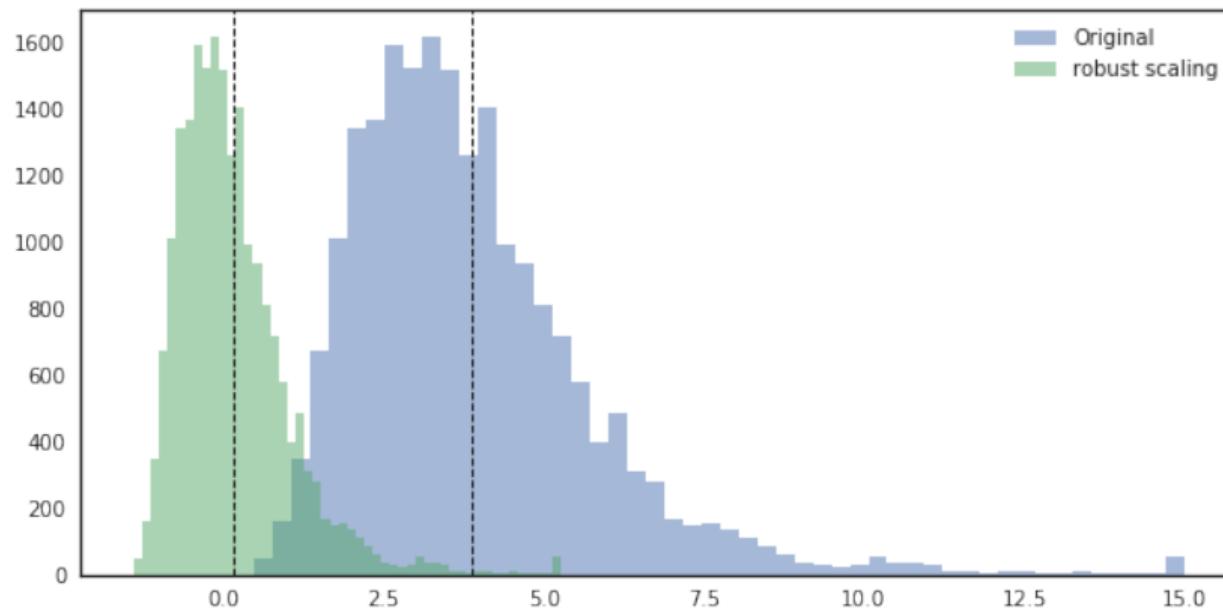
$$x_{S_i} = \frac{x_i - Q_2(x)}{Q_3(x) - Q_1(x)}$$

- Lijkt op MIN-MAX scaler maar gebruikt de interkwartielafstand ipv range.
- Houdt geen rekening met uitschieters.
- Gebruikt minder data bij het bepalen van de schaal.
- Range van de genormaliseerde data is groter dan bij MIN-MAX scaling.
- Garandeert geen genormaliseerde data op exact dezelfde schaal.



# Normalisatie

## Robust scaling



# Normalisatie

## Robust scaling

```
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# alternatief
scaler = RobustScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

# Feature expansion

## Nieuwe features

Bedenken van nieuwe features

Voorbeelden:

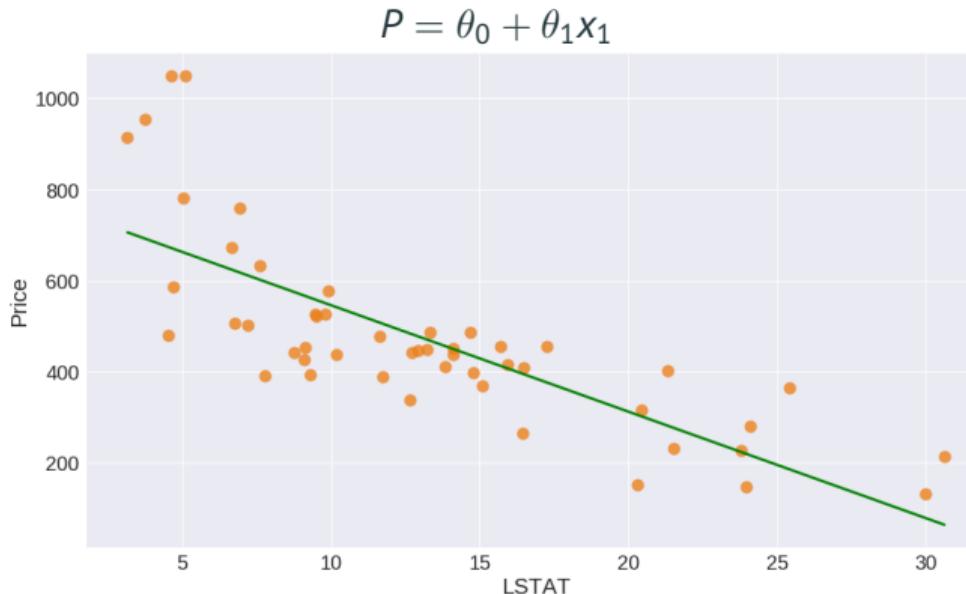
- Uit de lengte en breedte van een huis de oppervlakte halen als nieuwe feature.
- Uit een start en eindpunt de afstand halen als nieuwe feature.
- Uit een datum afleiden welke dag van de week het is.
- Veranderingen in de features.
- Nieuwe opgemeten parameters.

# Feature expansion

## Hogere-orde features

Het verband tussen features en de target(s) is niet altijd linear.

Voorbeeld: samenhang tussen LSTAT ( $x_1$ ) en de huizenprijs ( $P$ ).

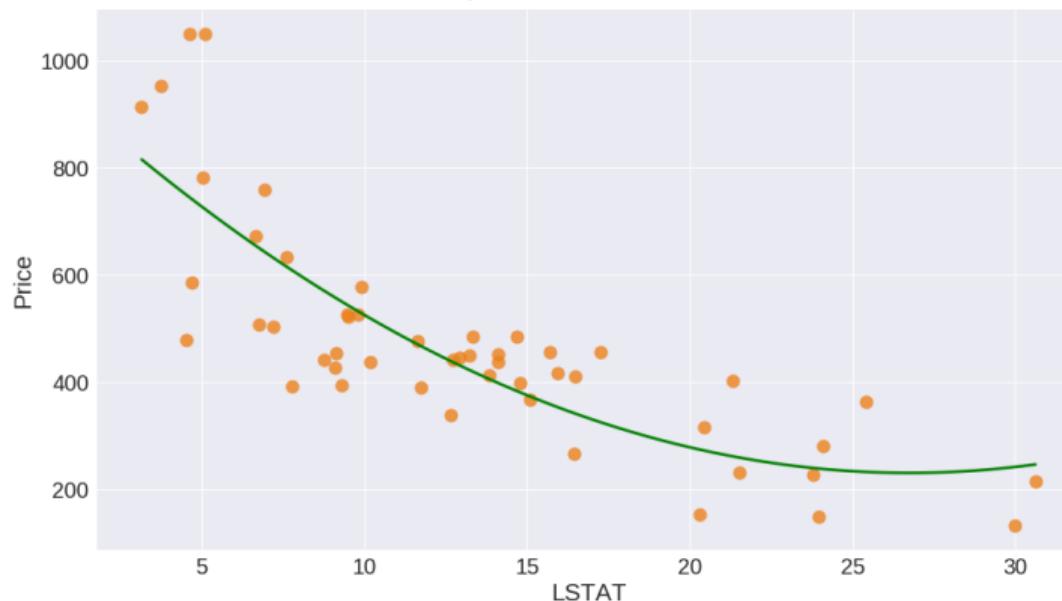


# Feature expansion

## Hogere-orde features

Toevoegen van een extra hogere-orde feature  $x_2 = x_1^2$

$$P = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

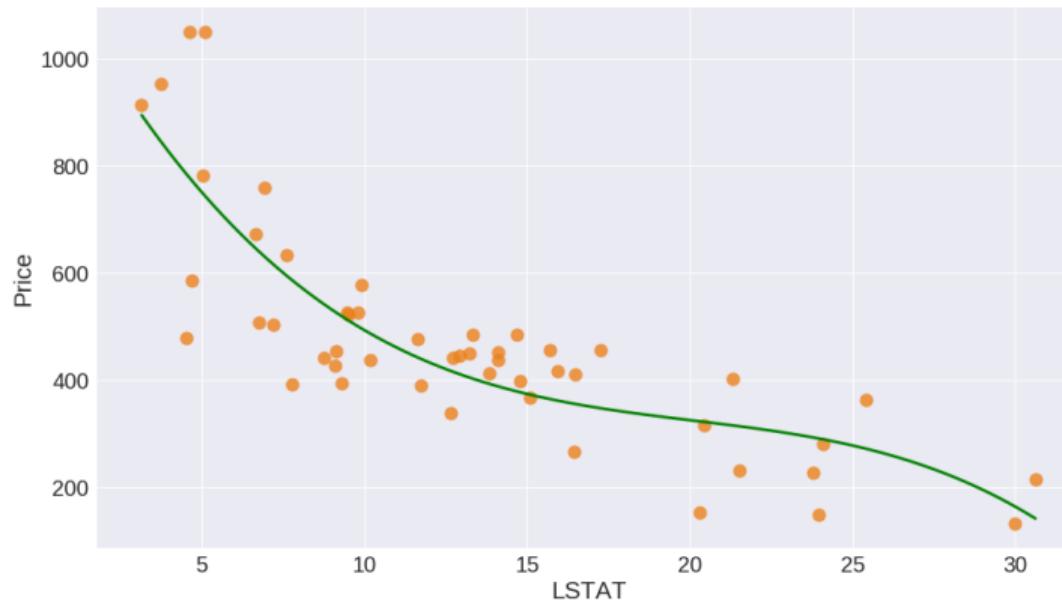


# Feature expansion

## Hogere-orde features

Toevoegen van een extra hogere-orde feature  $x_3 = x_1^3$

$$P = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

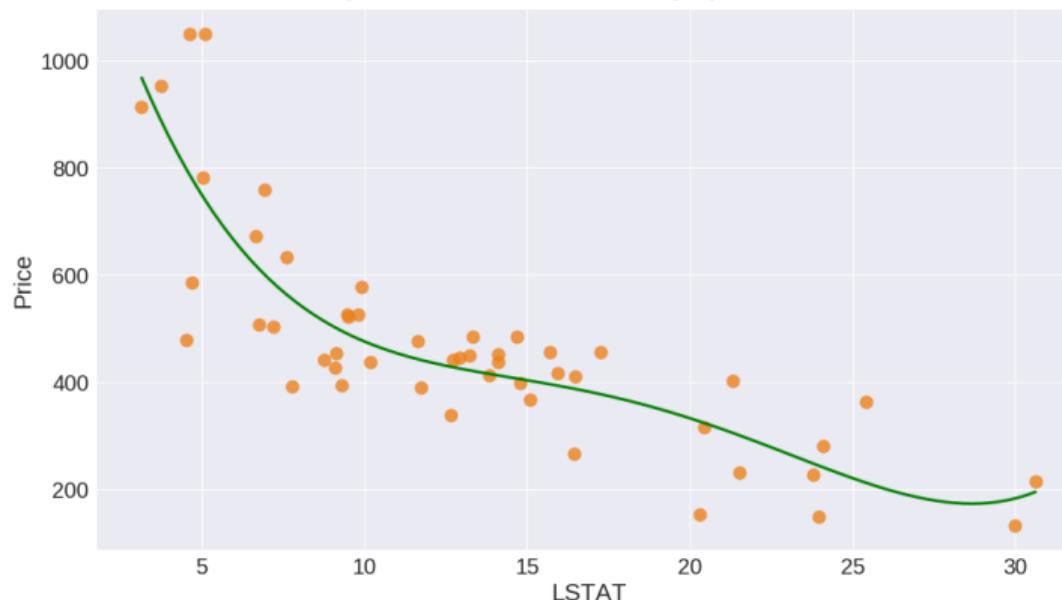


# Feature expansion

## Hogere-orde features

Toevoegen van een extra hogere-orde feature  $x_4 = x_1^4$

$$P = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$



# Feature expansion

## Hogere-orde features

```
# toevoegen van extra feature: LSTAT^2 LSTAT^3
dataset.insert(dataset.columns.size - 1, 'LSTAT^2', dataset.LSTAT**2)
dataset.insert(dataset.columns.size - 1, 'LSTAT^3', dataset.LSTAT**3)
```

	CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PT	B	LSTAT	LSTAT^2	LSTAT^3	Price
0	0.00632	18.0	2.31	0.538	6.575	65.199997	4.0900	1	296	15.300000	396.899994	4.98	24.800400	123.505993	504.000000
1	0.02731	0.0	7.07	0.469	6.421	78.900002	4.9671	2	242	17.799999	396.899994	9.14	83.539606	763.552030	453.600008
2	0.02729	0.0	7.07	0.469	7.185	61.099998	4.9671	2	242	17.799999	392.829987	4.03	16.240902	65.450837	728.700016
3	0.03237	0.0	2.18	0.458	6.998	45.799999	6.0622	3	222	18.700001	394.630005	2.94	8.643600	25.412185	701.400032
4	0.06905	0.0	2.18	0.458	7.147	54.200001	6.0622	3	222	18.700001	396.899994	5.33	28.408899	151.419431	760.200016

Model met extra features trainen en nadien evalueren op test set.

**Opgepast: bij de test set moet je ook dezelfde features toevoegen.**

# One-hot encoding

## Doel van one-hot encoding

Omzetten van categorische variabelen naar meerde aparte features.

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50



One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

# One-hot encoding

## One-hot encoding in Sklearn

```
dataset = pd.concat([dataset, pd.get_dummies(dataset['food_name'], prefix='food')], axis=1)
dataset.drop(['food_name'], axis=1, inplace=True)
dataset.head()
```

	food_name	Calories	Calories	food_Apple	food_Broccoli	food_Chicken	food_Chocolat
0	Apple	95	0	95	1	0	0
1	Chicken	231	1	231	0	0	1
2	Broccoli	50	2	50	0	1	0
3	Chocolat	549	3	549	0	0	1

## **Underfitting en overfitting**

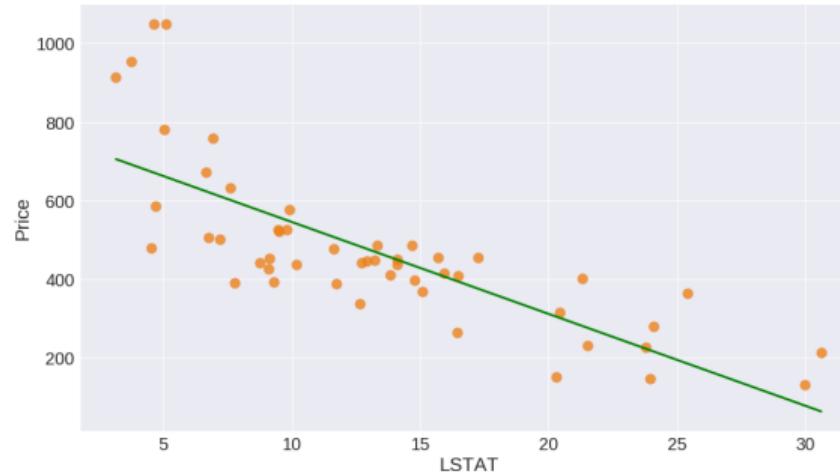
---

# Underfitting en overfitting

## Underfitting

Underfitting treedt op wanneer een model de training data niet kan modelleren en ook niet kan generaliseren op nieuwe data.

- Het model is te 'simpel'.
- Model met een hoge bias.

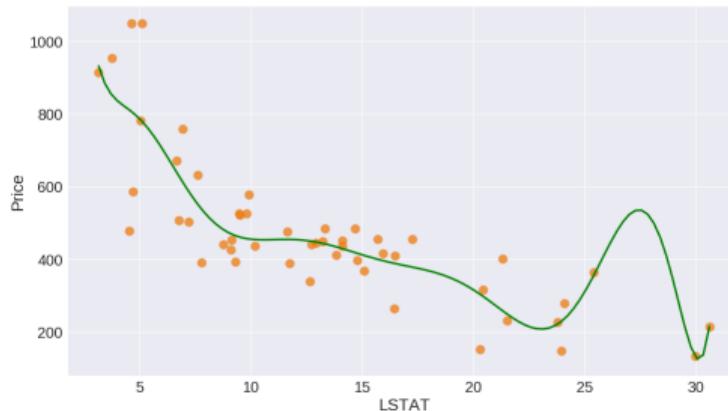


# Underfitting en overfitting

## Overfitting

Overfitting treedt op wanneer een model de training data te goed modelleert en niet kan generaliseren op nieuwe data.

- Het model is te 'complex'.
- De ruis van willekeurige fluctuaties in de data wordt opgepikt.
- Model met een hoge variance.



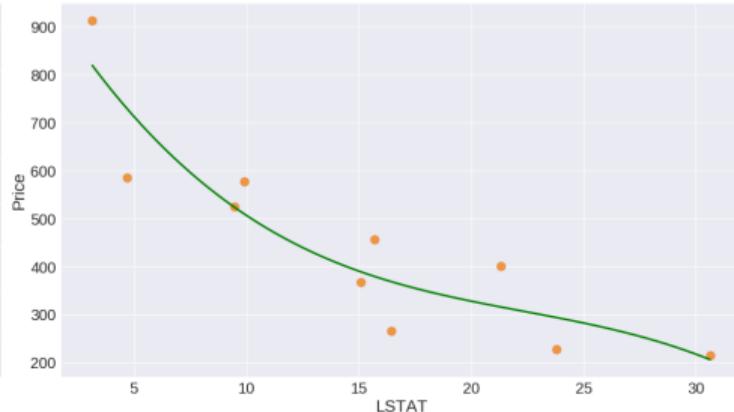
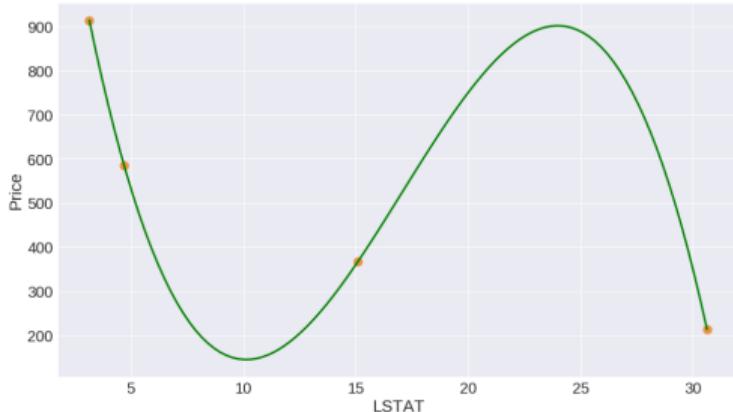
# Underfitting en overfitting

## Overfitting

Afhankelijkheid van grootte van de trainingset (aantal observaties m).

- Bij weinig observaties: snel overfitting bij complexer model.
- Bij veel observaties: minder snel overfitting bij complexer model.

Voorbeeld: Derde orde polynoom



# Underfitting en overfitting

## Regularisatie (regularization)

Methode om de mate van bias van een hypothese te regelen en een goed evenwicht te vinden tussen underfitting en overfitting.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left( h_{\theta}(x^{(i)}) - y^{(i)} \right)^2 + R(\theta)$$

$R(\theta)$  is de regularisatie-term. Dit is een extra kostenterm die het gebruik van hogere orde features afstraft tenzij ze de globale kostenfunctie doen dalen.

# Underfitting en overfitting

## Regularisatie (regularization)

Voorbeeld van een regularisatie

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left( h_{\theta}(x^{(i)}) - y^{(i)} \right)^2 + \lambda \theta \theta^T$$

$\lambda$  is een tuning parameter (hyper parameter)       $\theta = \{\theta_1, \dots, \theta_n\}$

- $\lambda = 0 \Rightarrow$  geen regularisatie.
  - $\lambda = \infty \Rightarrow \theta = 0$ .
  - $\lambda$  tussenin regelt de mate van regularisatie.
- intercept  $\theta_0$  wordt meestal niet geregulariseerd.

# Underfitting en overfitting

## Regularisatie (regularization)

Voorbeeld van een regularisatie

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left( h_{\theta}(x^{(i)}) - y^{(i)} \right)^2 + \lambda \theta \theta^T$$

De tuningparameter  $\lambda$  regelt de complexiteit van de hypothese

- Kleine waarde voor  $\lambda$ : lage bias, hoge variantie (overfitting).
- Grote waarde voor  $\lambda$ : hoge bias, lage variantie (underfitting).

Afhankelijk van hoe R gedefinieerd wordt is er andere benaming voor de regularisatie:

- Ridge regression (L2 regularisatie).
- Lasso regression (L1 regularisatie).

# Underfitting en overfitting

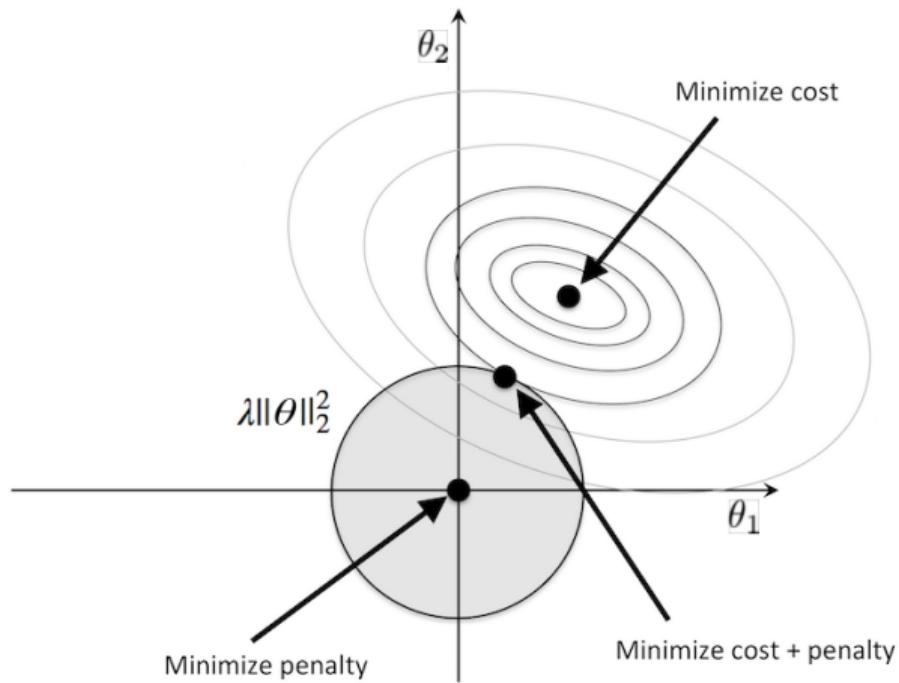
Regularisatie via L2 norm

$$J_{L2} = J + \lambda_2 \sum_{j=1}^m \theta_j^2$$

$$J_{L2} = \sum_{i=1}^n (\text{target}^{(i)} - \text{output}^{(i)}) + \lambda_2 \sum_{j=1}^m \theta_j^2$$

# Underfitting en overfitting

Regularisatie via L2 norm



# Underfitting en overfitting

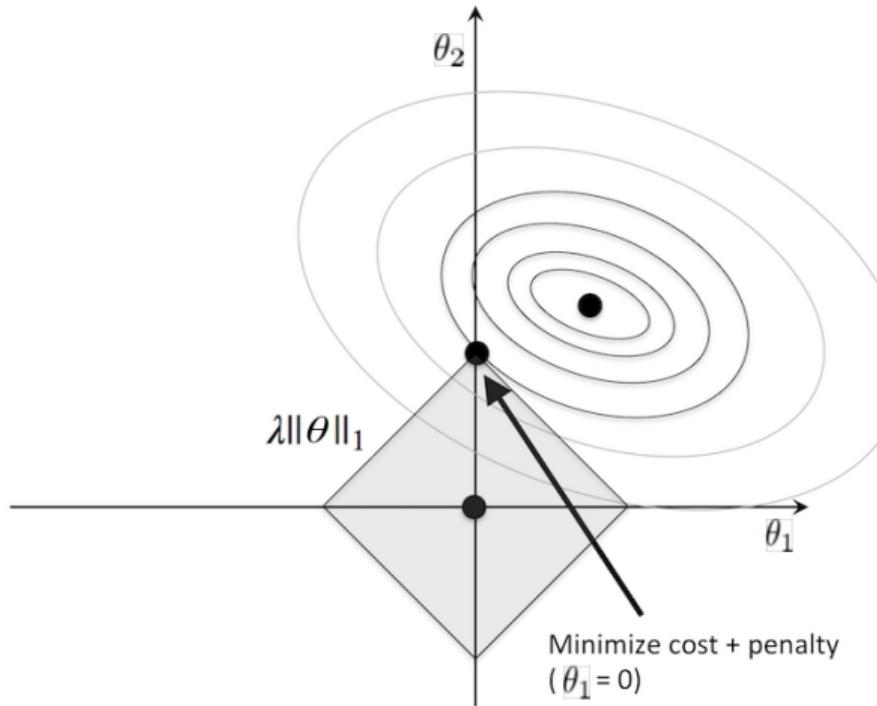
Regularisatie via L1 norm

$$J_{L1} = J + \lambda_1 \sum_{j=1}^m |\theta_j|$$

$$J_{L1} = \sum_{i=1}^n (\text{target}^{(i)} - \text{output}^{(i)}) + \lambda_1 \sum_{j=1}^m |\theta_j|$$

# Underfitting en overfitting

Regularisatie via L1 norm



# Underfitting en overfitting

## Voorbeeld regularisatie op huizenprijzen

Via Ridge of Lasso regressie met regularisatieparameter  $\alpha$ .

- hoe groter  $\alpha$ , hoe sterker de regularisatie en dus hoe minder complex het model.
- hoe kleiner  $\alpha$ , hoe zwakker de regularisatie en dus hoe complexer het model.

```
regmodel = Ridge(alpha=0.14,tol=0.0001,fit_intercept=True)
regmodel.fit(X_train,y_train)
regmodel.score(X_test,y_test)
>> 0.79834480089914472
```

```
lregmodel = Lasso(alpha=0.5,tol=0.0001,fit_intercept=True)
lregmodel.fit(X_train,y_train)
lregmodel.score(X_test,y_test)
>> 0.8437113338085345
```