

Machine Learning & AI

Tuur Vanhoutte

16 februari 2021

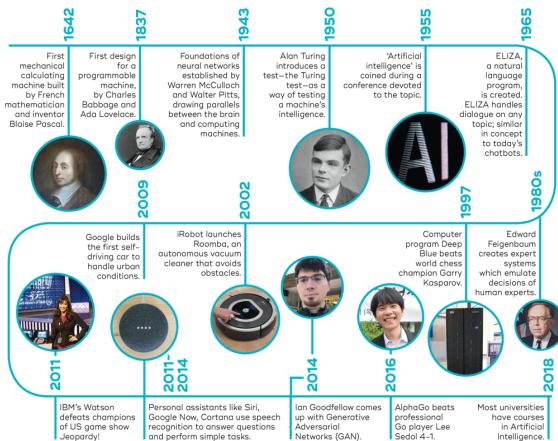
Inhoudsopgave

1 Inleiding	1
1.1 AI in context	1
1.1.1 Vormen van AI	1
1.1.2 Sectoren die de planeet verbeteren	2
1.1.3 Waarom nu?	2
2 Hoe leren uit data?	2
2.1 Leeralgoritmes	2
2.2 Supervised Learning	3
2.2.1 Regressie vs Classificatie	4
2.2.2 Voorbeeld	4
2.3 Unsupervised learning	5
2.4 Reinforcement learning	6
2.5 Overzicht leeralgoritmes	6
2.6 Werkwijze van een ML Project	7
2.6.1 Tijdverdeling	7
3 Enkelvoudige Lineaire regressie	8
3.1 Voorbeeld	8
3.1.1 Scatterplot	8
3.2 De hypothese	9
3.3 De kostenfunctie	9
3.4 Gradient Descent (GDS)	9
3.4.1 Learning rate	10
4 Meervoudige lineaire regressie	11
4.1 Statistische vooranalyse	12
4.1.1 Consistentie van de dataset	12
4.1.2 Uitschieters	12
4.1.3 Onderlinge correlatie	13
4.2 Features en targets	13
4.3 Trainen van het model	14
4.3.1 Initialiseren en trainen van het regressiemodel	14
4.4 Testen van het model	15
4.4.1 Voorspellingen maken	15
4.5 Performantie en scores van het model	15
4.5.1 Mean Absolute Error (MAE)	15
4.5.2 Mean Squared Error (MSE)	16
4.5.3 Determinatiecoëfficiënt	16
5 Feature engineering	16
5.1 Normalisatie / Scaling	17
5.1.1 Voordelen	17
5.1.2 MIN-MAX-scaling	17
5.1.3 Standard scaling (normalisatie)	18
5.1.4 Robust scaling	20
5.2 Feature expansion	20
5.2.1 Nieuwe features	20
5.2.2 Hogere-orde features	21
5.3 One-hot encoding	22

6 Underfitting & overfitting	23
6.1 Underfitting	23
6.2 Overfitting	23
6.2.1 Impact van de grootte van de dataset	24
6.3 Regularisatie (regularisation)	24
6.3.1 Voorbeeld regularisatie	24
6.3.2 Regularisatie met L2 norm	25
6.3.3 Regularisatie met L1 norm	25
6.3.4 Voorbeeld regularisatie op huizenprijzen	26

1 Inleiding

1.1 AI in context



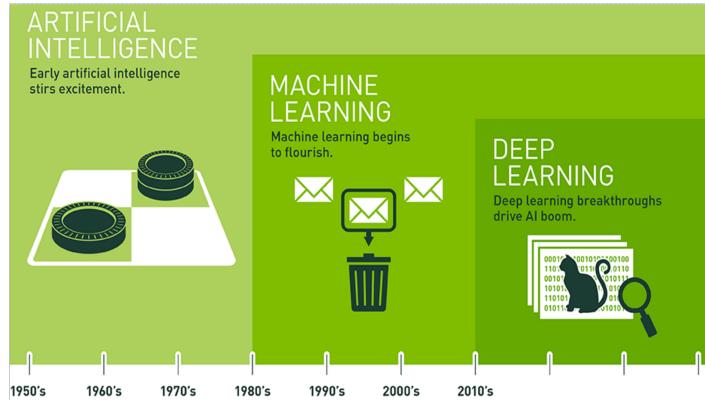
Figuur 1: Geschiedenis van AI

Belangrijkste gebeurtenissen:

- **1943:** McCulloch - Pitts: fundering van neurale netwerken
- **1950:** Alan Turing: de Turing test
- **1956:** Dartmouth workshop: bijeenkomst voor breinstorm AI
- **1997:** Garry Kasparov vs Deep Blue (IBM)
- **2011:** IBM Watson
- **2016:** AlphaGo
- **2021-:** toekomst

1.1.1 Vormen van AI

- Zwakke AI (weak AI / Artificial Narrow Intelligence)
 - Goed in een bepaalde taak maar alleen in die taak
 - **Voorbeelden:** spamfilters, schaakcomputers, gezichtsherkenning
- Sterke AI (strong AI / Artificial General Intelligence)
 - Intelligentie op menselijk niveau
 - In staat om zich aan te passen en problemen te leren oplossen in verschillende contexten
- Superintelligentie (Artificial Super Intelligence)
 - Als AI zelfbewust wordt en de mens op alle vlakken voorbij steekt



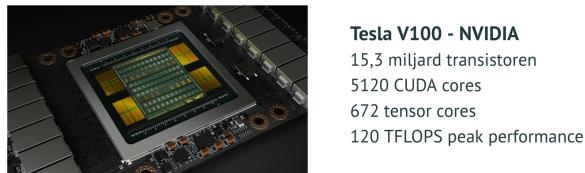
Figuur 2: AI vs ML vs DL

1.1.2 Sectoren die de planeet verbeteren

- Klimaatsverandering
- Biodiversiteit en conservatie
- Water
- Hernieuwbare energie
- Medische sector
- Weer- en rampenvoorspelling

1.1.3 Waarom nu?

- Snellere hardware
- Betere algoritmes
- Meer data
- (Open source) frameworks



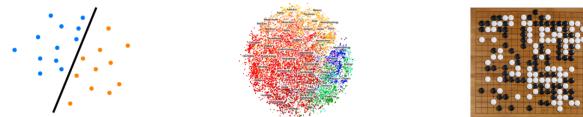
Figuur 3: Voorbeeld huidige hardware: de Tesla V100 van Nvidia

2 Hoe leren uit data?

2.1 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



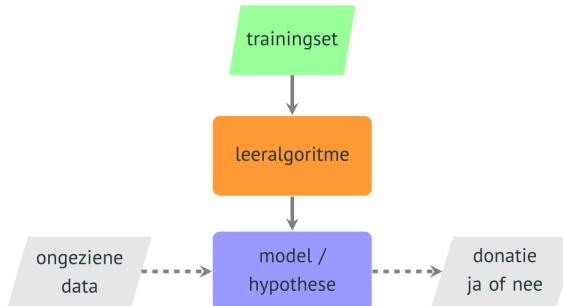
Figuur 4: Supervised / Unsupervised / Reinforcement learning

2.2 Supervised Learning

Leren uit een gelabelde dataset. Vind het verband tussen de features en de labels

		sentiment	text
0	1	1	I am going to start reading the Harry Potter series again because that is one awesome story.
1	1	1	the story of Harry Potter is a deep and profound one, and I love Harry Potter.
2	1	1	Mission Impossible 3 was excellent.
3	0	0	The Da Vinci Code sucked, but the night was great.
4	1	1	The Da Vinci Code was absolutely AWESOME!
5	0	0	Then snuck into Brokeback Mountain, which is the most depressing movie I have ever seen..
6	1	1	I love Harry Potter.
7	0	0	Ok brokeback mountain is such a horrible movie.
8	1	1	He's like,"YEAH I GOT ACNE AND I LOVE BROKEBACK MOUNTAIN".
9	0	0	Da Vinci Code sucks.

Figuur 5: Leren uit een dataset

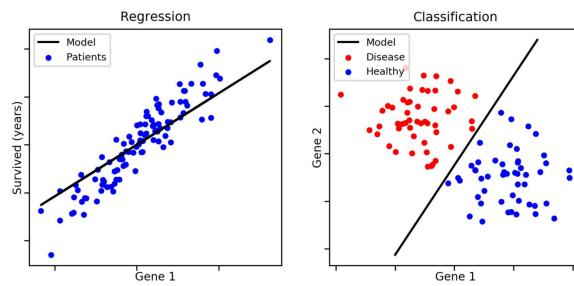


Figuur 6: Supervised learning kan uit ongeziene data een resultaat berekenen

2.2.1 Regressie vs Classificatie



Figuur 7: Regressie vs classificatie

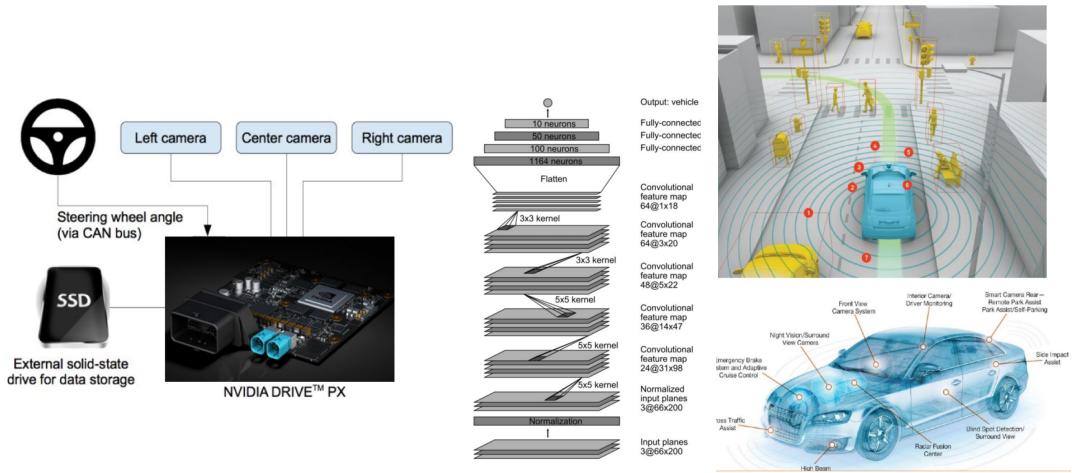


Figuur 8: Regressie vs classificatie

2.2.2 Voorbeeld

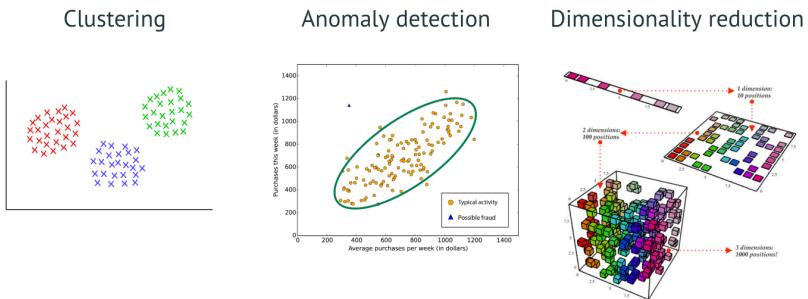
Hoe stuurhoek bepalen bij een self-driving car?

- (infrarood) camera's
- Stereo vision
- Radar
- LIDAR
- GPS
- Audio

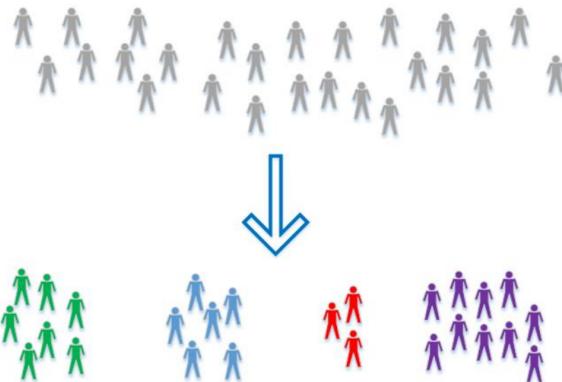


Figuur 9: Via sensoren weet de auto

2.3 Unsupervised learning

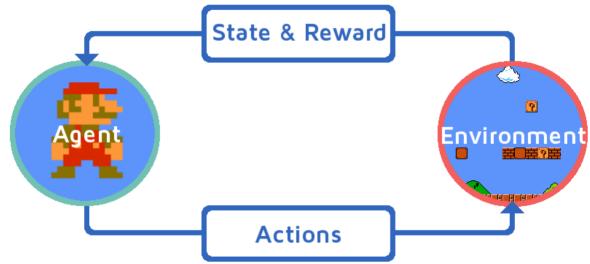


Figuur 10: Unsupervised Learning



Figuur 11: Voorbeeld Clustering: de data in groepen verdelen

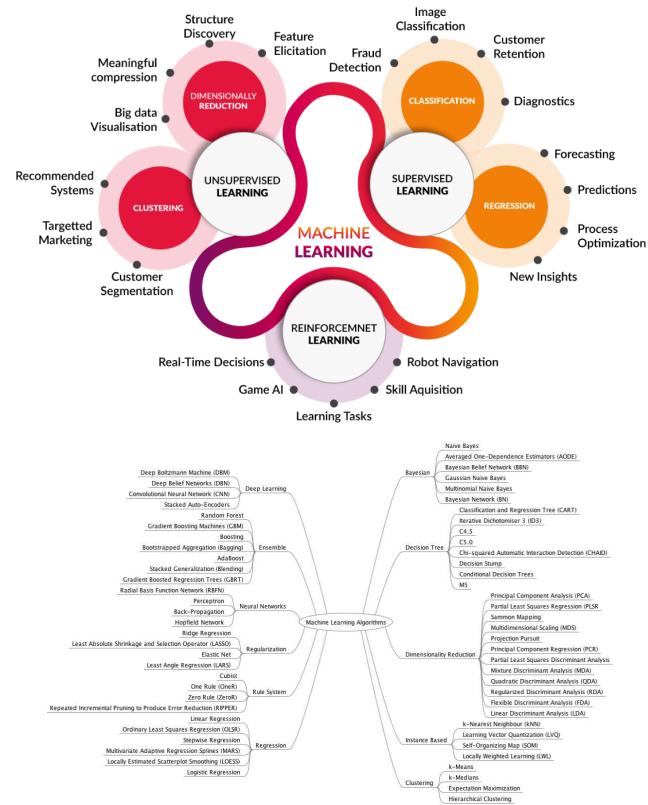
2.4 Reinforcement learning



Figuur 12: Reinforcement learning

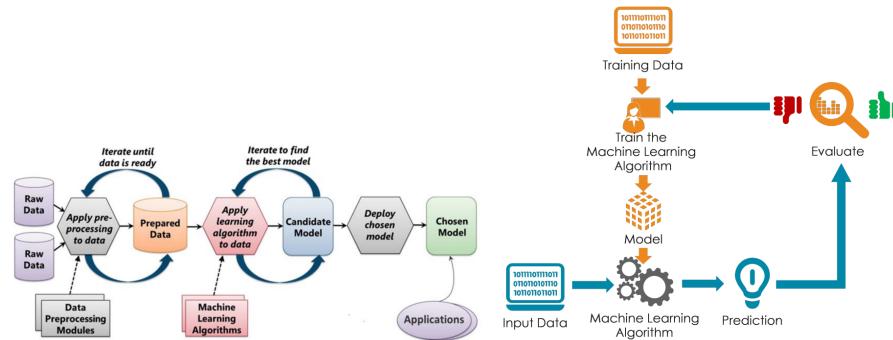
- Voor elke actie krijgt de AI feedback
- De AI leert uit de feedback
- In het begin zijn de acties heel willekeurig

2.5 Overzicht leeralgoritmes

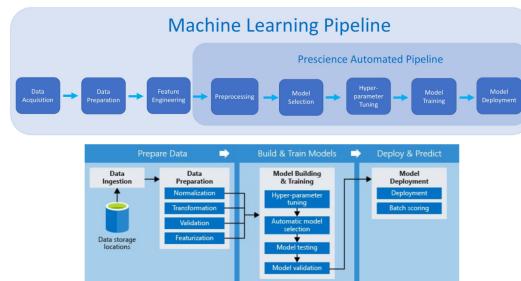


Figuur 13: Overzicht

2.6 Werkwijze van een ML Project

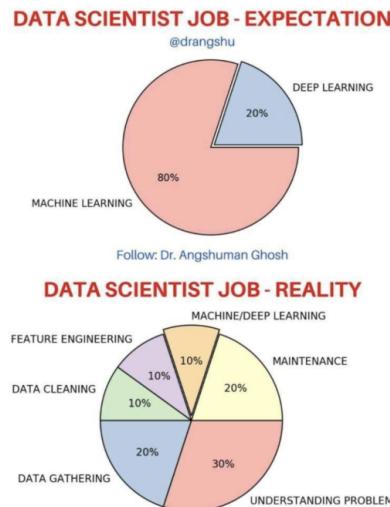


Figuur 14



Figuur 15

2.6.1 Tijdverdeling



Figuur 16: Tijdverdeling: verwachting vs realiteit

3 Enkelvoudige Lineaire regressie

3.1 Voorbeeld

	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

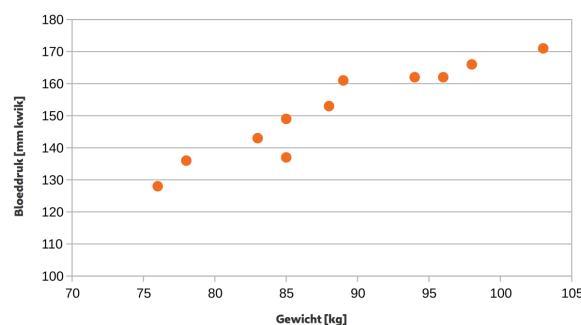
Figuur 17: Voorspel de bloeddruk op basis van leeftijd en gewicht

- **features:** leeftijd en gewicht
- **target:** bloeddruk (=wat je wil voorspellen = output = label)
- **trainingset:** 11 training examples (=samples)

Definitie 3.1 (Regressie-analyse) *Regressie-analyse is het modelleren van of het zoeken naar een verband op basis van één of meerdere variabelen.*

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

3.1.1 Scatterplot



Figuur 18: Scatterplot: de grafiek toont een positieve correlatie ⇒ een sterk verband

3.2 De hypothese

Definitie 3.2 (De hypothese) Het verband (model of hypothese) $h_\theta(x)$ is van de vorm:

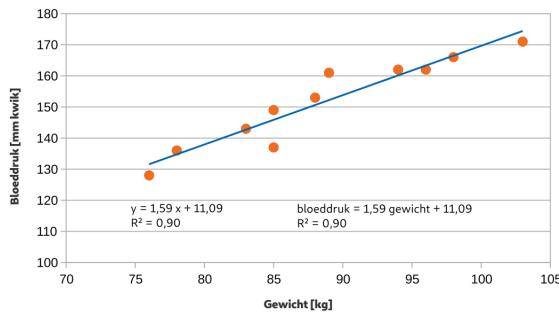
$$h_\theta(x) = \theta_0 + \theta_1 x \quad (1)$$

Bepalen van de optimale waarden voor θ_0 en θ_1 :

- θ_0 = snijpunt van de y-as (= noemen we ook de **bias**)
- θ_1 = helling van de rechte (rico)

De parameters θ_i = **weights**

Het zoeken van het model / hypothese = **training / learning**



Figuur 19: Lineaire trnedlijn met model $h_\theta(x)$

R²-waarde: determinatiecoëfficiënt

3.3 De kostenfunctie

We moeten de kostenfunctie $J(\theta)$ via de **Least Mean Squared** methode (LMS).

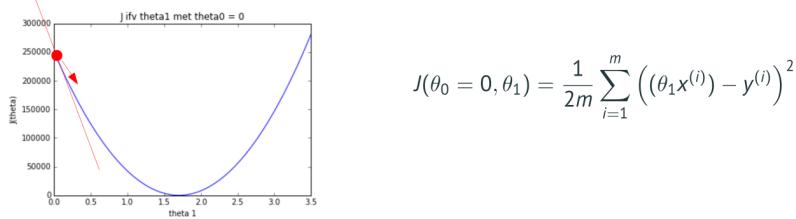
$$J(\theta) = \frac{1}{2 \cdot m} \cdot \sum_{i=1}^m (h_\theta(x_i) - y_i)^2 \quad (2)$$

- m = de bias = snijpunt met de y-as
- De kostenfunctie berekent de gemiddelde fout door alle fouten op te tellen
- Elke fout wordt gekwadrateerd om:
 - negatieve waardes positief te maken
 - de fout uit te groten

3.4 Gradient Descent (GDS)

$$J(\theta_0, \theta_1) = \frac{1}{2 \cdot m} \cdot \sum_{i=1}^m ((\theta_1 \cdot x_i + \theta_0) - y_i)^2 \quad (3)$$

Stel de parameters θ_0 en θ_1 voortdurend bij in een iteratief proces tot je de waarden voor θ_0 en θ_1 hebt gevonden die de kleinst mogelijke waarde. Start met willekeurige waarden.



Figuur 20: De GDS als $\theta_0 = 0$

- Je krijgt een dalparabool als uitkomst
- Je kan aflezen wat de parameters moeten zijn om de kleinst mogelijke waarde te vinden
- In de realiteit heb je vaak veel meer dan 2 gewichten
 - Voorbeeld: de textgenererende AI GPT-3 heeft rond de 175 miljard gewichten
 - \Rightarrow veel rekenkracht nodig om beste uitkomst te vinden

3.4.1 Learning rate

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

Bepaal de gradient naar θ_0 en θ_1

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

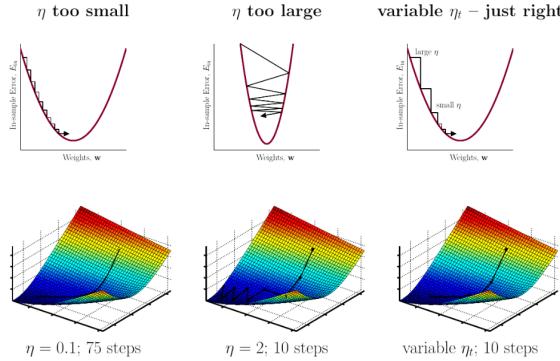
$$\frac{dJ(\theta)}{d\theta_1} = \frac{2}{2m} \sum_{i=1}^m ((\theta_1 x^{(i)} + \theta_0) - y^{(i)}) x^{(i)}$$

Update de parameters θ_0 en θ_1 volgens:

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

Figuur 21: De parameters θ_0 en θ_1 stellen we constant bij (formules niet te kennen)

- We bepalen de afgeleide (= de gradient, de helling) van θ_0 en θ_1
- We gebruiken die afgeleiden om een betere θ_0 en θ_1 te vinden.
- We vermenigvuldigen de gradient met een variable η (=de learning rate)
- Onze nieuwe θ wordt berekend met behulp van de oude θ en de afgeleide maal de learning rate.



Figuur 22: Learning rate η : bij een te kleine/te grote η hebben we te veel stappen

De learning rate η stellen we constant bij om met zo weinig aantal stappen het optimum te bereiken.

4 Meervoudige lineaire regressie

Definitie 4.1 (Meervoudige lineaire regressie) Bij meervoudige lineaire regressie (*multiple regression*) wordt het model/hypothese bepaald aan de hand van een trainingset met **meerdere features**.

- Bloeddruk wordt bepaald a.d.h.v. gewicht en leeftijd
- De kwaliteit van wijn wordt voorspeld op basis van: zuurtegraad, suikergehalte, chloriden, dichtheid, sulfaten, hoeveelheid alcohol, ...
- Het warmeverlies van een huis wordt voorspeld op basis van: het type glas, muurisolatie, oriëntatie van het huis, ...

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n + \quad (4)$$

- 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

Figuur 23: **Voorbeeld:** voorspel de huisprijs op basis van deze features

4.1 Statistische vooranalyse

4.1.1 Consistentie van de dataset

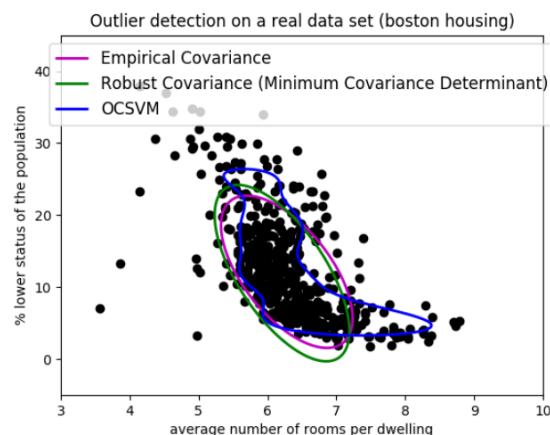
- Volledigheid van de dataset
- Inconsistenties
- Spreiding van de gegevens

We verwijderen de CHAS kolom:

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

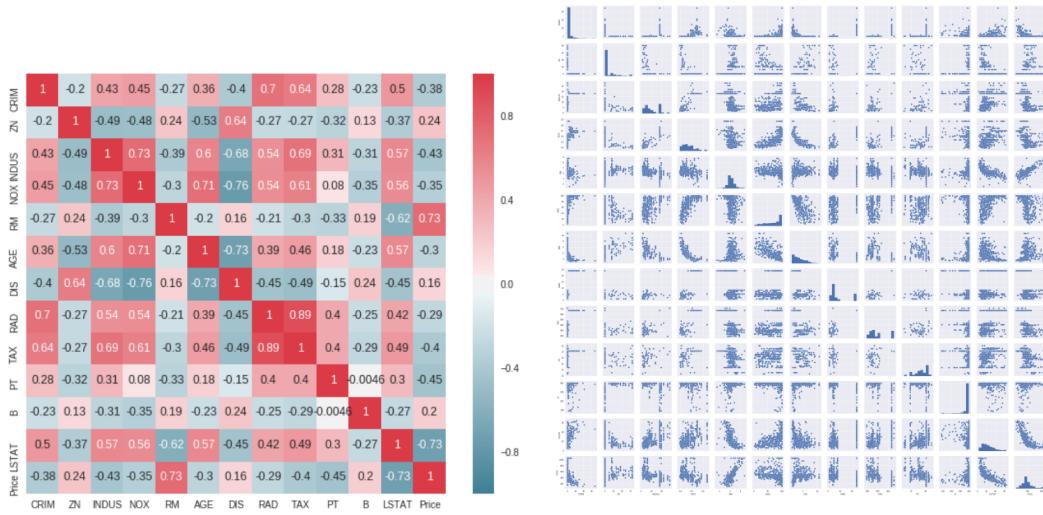
4.1.2 Uitschieters

- Vinden en verwijderen van extreme waarden/samples
- Geavanceerde technieken: zie later bij clustering



Figuur 24: Uitschieters

4.1.3 Onderlinge correlatie



Figuur 25: Heatmap en pairplot van de underline correlatie tussen de features

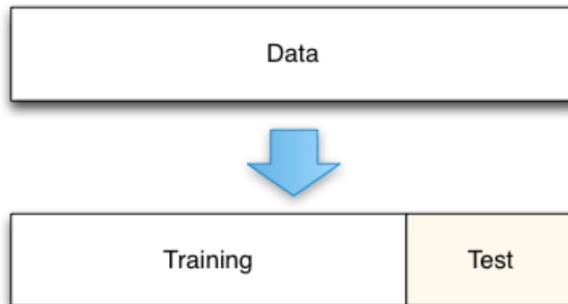
4.2 Features en targets

CRIM	ZN	INDUS	NOX	RM	AGE	DIS	RAD	TAX	PT	B	LSTAT	Price
0.00632	18.0	2.31	0.538	6.557	65.199997	4.0900	1	296	15.300000	396.899994	4.98	504.000000
0.02731	0.0	7.07	0.469	4.621	78.900002	4.9671	2	242	17.799999	396.899994	9.14	453.600000
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	45.999999	6.0622	3	22	18.700001	394.630000	2.94	701.400032
0.06903	0.0	2.18	0.458	7.147	54.200001	6.0622	3	222	18.700001	396.899994	5.33	760.200016

Figuur 26: Dataset opsplitsen in features en targets

```
1 y = dataset['target_kolom'].values
2 X = dataset.drop('target_kolom',axis=1).values
3 # Alternatief
4 features=list(dataset.columns[:dataset.columns.size-1])
5 X = dataset[features].values
6 y = dataset['Price'].values
```

4.3 Trainen van het model



Figuur 27: Dataset opsplitsen in training set en test set

- Belangrijk om eerst de data te randomiseren: te data zou gesorteerd kunnen zijn, dat willen we vermijden
- Stel dat huizenprijzen van laag naar hoog gesorteerd is, en je traaint de data op de eerste 75%, en test de laatste 25%. Resultaat: Het model zal niet getraind zijn op dure huizen.
- Ander voorbeeld: stel dat je een self-driving AI alleen traaint op de autosnelweg, en dan test in een zone-30 straat bij een school...

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

4.3.1 Initialiseren en trainen van het regressiemodel

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

Model:

```
1 print('coeffs: ', lregmodel.coef_)
2 print('intercept', lregmodel.intercept_)
3
4 >> coeffs: [-3.56141289e+00, 4.05479295 e-01, 8.14080284 e-01,
5     8.96450415 e+01, -3.02997261e-01, -2.77339444e+01,
6     7.47151897 e+00, -2.92233040e-01, -1.61741146e+01,
7     -1.17962045e +01]
8
9 >> intercept: 650.652022517
```

Price = $-3.56 \times CRIM + 0.41 \times ZN + 0.81 \times INDUS - 270.51 \times NOX + 89.65 \times RM - 0.30 \times AGE - 27.74 \times DIS + 7.47 \times RAD - 0.29 \times TAX - 16.17 \times PT + 0.08 \times B - 11.80 \times LSTAT + 650.65$

4.4 Testen van het model

4.4.1 Voorspellingen maken

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

Figuur 28: Voorspel de prijs van een huis met deze features

```
1 house = np.array([0.11, 0, 12.03, 0.57, 6.80, 89.30, 2.39, 1,
2 273, 21.00, 393.45, 6.48])
3
4 house = house.reshape(1, -1)
5
6 # met reshape wordt house:
7 # house = np.array([[0.11, 0, 12.03, 0.57, 6.80, 89.30, 2.39, 1,
8 # 273, 21.00, 393.45, 6.48]])
9
10 price = lregmodel.predict(house)
11
12 print('De prijs van het huis bedraagt: ', price)
13
14 >> De prijs van het huis bedraagt: 563.68335073
```

- `reshape(1, -1)` maakt een rijvector
- Werkelijke prijs: 562.00

4.5 Performantie en scores van het model

4.5.1 Mean Absolute Error (MAE)

Definitie 4.2 (MAE) De Mean Absolute Error (MAE) is *het gemiddelde van de absolute waarden van het verschil tussen de werkelijke waarden y_i en de voorgespelde waarden \hat{y}_i .*

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^n |y_i - \hat{y}_i| \quad (5)$$

```
1 from sklearn.metrics import mean_absolute_error
2
3 y_predicted = lregmodel.predict(X_test)
4 MAE = mean_absolute_error(y_test, y_predicted)
5
6 print('MAE= ', MAE)
7
8 >> MAE = 64.0090867586
```

4.5.2 Mean Squared Error (MSE)

Definitie 4.3 (MSE) De Mean Squared Error (MSE) is **het gemiddelde van de gekwadrateerde waarden** van het verschil tussen de werkelijke waarden y_i en de voorspelde waarden \hat{y}_i .

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (6)$$

```
1 from sklearn.metrics import mean_squared_error
2
3 y_predicted = lregmodel.predict(X_test)
4 MSE = mean_squared_error(y_test, y_predicted)
5
6 print('MSE = ', MSE)
7
8 >> MSE = 7803.89332739
```

4.5.3 Determinatiecoëfficiënt

Definitie 4.4 (De determinatiecoëfficiënt R^2) De determinatiecoëfficiënt (R^2) is de variabiliteit van het model

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (7)$$

Bij perfecte voorspellingen is $R^2 = 1$

Een negatieve waarde voor R^2 betekent dat het model slechter scoort dan een horizontale lijn (= slechter dan het gemiddelde te nemen) ($y_i = \bar{y}$, \bar{y} is het gemiddelde van y)

```
1 from sklearn.metrics import r2_score
2
3 y_predicted = lregmodel.predict(X_test)
4 r2 = r2_score(y_test, y_predicted)
5
6 print('r2_score = ', r2)
7
8 # alternatieve manier voor het bepalen van de r2 score:
9 r2 = lregmodel.score(X_test, y_test)
10
11 >> r2 score = 0.754254234917
```

5 Feature engineering

Om een beter model te verkrijgen (en zo een betere R^2 score), kunnen we verschillende dingen doen:

- Meer data toevoegen
- Ander model kiezen, hyperparameter tuning
- Feature engineering

5.1 Normalisatie / Scaling

Definitie 5.1 (Normalisatie / Scaling) *Normalisatie of Scaling zorgt ervoor dat de features op dezelfde schaalverdeling staan*

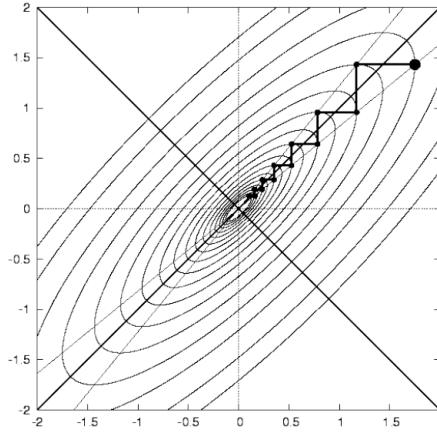
In ons voorbeeld van de huurprijs:

	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	3.85189	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	6.032866
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 \mapsto 0.871 terwijl TAX: 188 \mapsto 711

Figuur 29: NOX: min = 0.385, max = 0.871; TAX: min = 188, max = 711

5.1.1 Voordelen

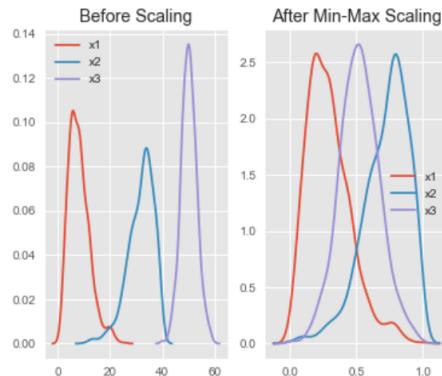


Figuur 30: Gradient Descent convergeert minder snel als features op een verschillende schaalgrootte staan. Normalisatie zorgt er dus voor dat het model sneller zal trainen.

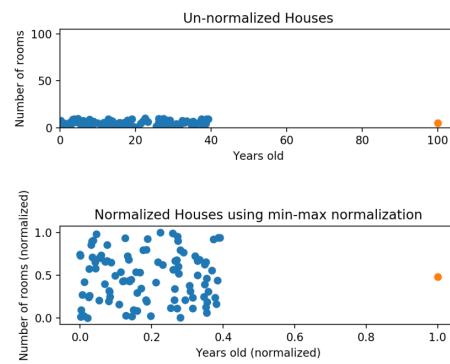
5.1.2 MIN-MAX-scaling

$$x_{s_i} = \frac{x_i - \text{Min}(x)}{\text{Max}(x) - \text{Min}(x)} \quad (8)$$

- 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



Figuur 31: Voor en na scaling



Figuur 32: Bij het voorbeeld van de huizenprijzen

```

1  from sklearn.preprocessing import MinMaxScaler
2
3  scaler = MinMaxScaler().fit(X_train)
4  X_train = scaler.transform(X_train)
5  X_test = scaler.transform(X_test)
6
7  # alternatief
8  scaler = MinMaxScaler()
9  X_train = scaler.fit_transform(X_train)
10 X_test = scaler.transform(X_test)

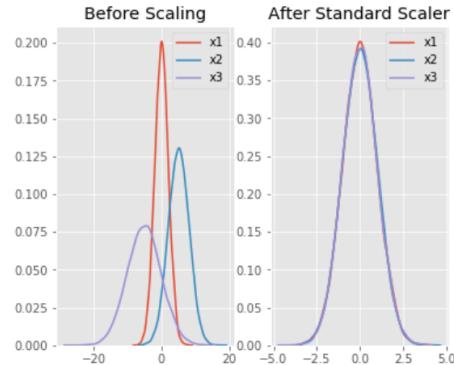
```

5.1.3 Standard scaling (normalisatie)

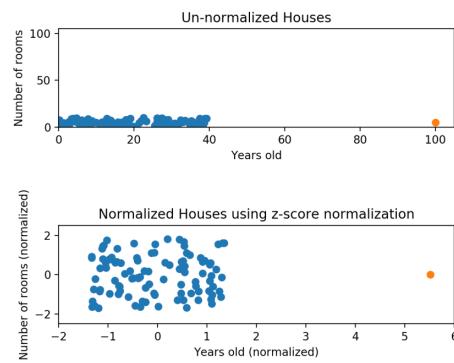
$$x_{s_i} = \frac{x_i - \text{mean}(x)}{\text{stddev}(x)} \quad (9)$$

- Geschaalde features;
 - Gemiddelde = 0
 - 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



Figuur 33: Voor en na standard scaling



Figuur 34: Bij het voorbeeld van de huizenprijzen

```

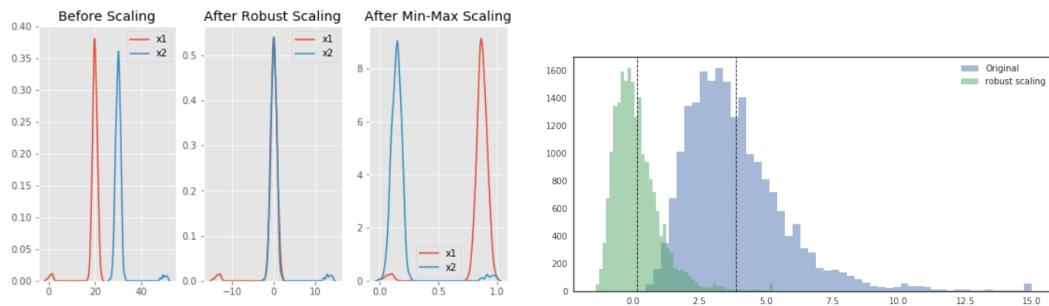
1 from sklearn.preprocessing import StandardScaler
2
3 scaler = StandardScaler().fit(X_train)
4 X_train = scaler.transform(X_train)
5 X_test = scaler.transform(X_test)
6
7 # alternatief
8 scaler = StandardScaler()
9 X_train = scaler.fit_transform(X_train)
10 X_test = scaler.transform(X_test)

```

5.1.4 Robust scaling

$$x_{s_i} = \frac{x_i - Q_2(x)}{Q_3(x) - Q_1(x)} \quad (10)$$

- 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



Figuur 35: Voor en na robust scaling

```

1 from sklearn.preprocessing import RobustScaler
2
3 scaler = RobustScaler().fit(X_train)
4 X_train = scaler.transform(X_train)
5 X_test = scaler.transform(X_test)
6
7 # alternatief
8 scaler = RobustScaler()
9 X_train = scaler.fit_transform(X_train)
10 X_test = scaler.transform(X_test)

```

5.2 Feature expansion

5.2.1 Nieuwe features

Bedenken van nieuwe features

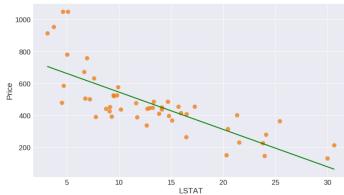
Voorbeelden:

- 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.

5.2.2 Hogere-orde features

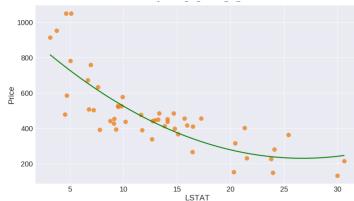
= Het verband tussen features en de target(s) is niet altijd lineair.

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



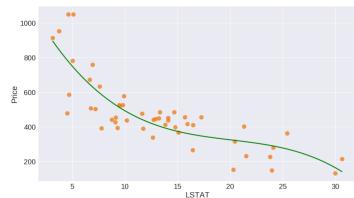
Figuur 36: $P = \theta_0 + \theta_1 x_1$

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

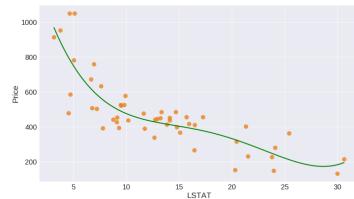


Figuur 37: $P = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

Toevoegen van een extra hogere-orde features $x_3 = x_1^3$ en $x_4 = x_1^4$:



Figuur 38: $P = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$



Figuur 39: $P = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

```

1 # toevoegen van een extra feature: LSTAT^2 LSTAT^3
2 dataset.insert(dataset.columns.size - 1, 'LSTAT^2', dataset.LSTAT**2)
3 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	4.9	24.800400	123.505993	504.000000
1	0.02731	0.0	7.07	0.469	6.421	78.900000	4.9671	2	242	17.799999	396.899994	9.14	83.539606	763.562006	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	65.460837	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	25.412181	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

Figuur 40: Resultaat toevoegen hogere-orde features

- Nu model met extra features trainen en nadien testen op de test set.
- Opgepast:** bij de test set moet je ook dezelfde features toevoegen.

5.3 One-hot encoding

- = Omzetten van categorische variabelen naar meerdere aparte features
- categorische variabelen = variabelen zonder echte waarden, de waarden stellen een categorie voor (niet altijd een nummer)
- ⇒ voor elke categorie een nieuwe kolom
- ‘Dummy Variable Trap’
 - = als een rij maar tot 1 categorie kan behoren (zie onderstaand voorbeeld, een appel is geen kip), dan zou je in principe 1 kolom kunnen schrappen en dan kan je toch dezelfde informatie krijgen.
 - $x_1 + x_2 + x_3 = 1 \Leftrightarrow x_1 = 1 - x_2 - x_3$
 - In de praktijk laat men dit gewoon staan.
 - (niet te kennen op examen)

Label Encoding			One Hot Encoding			
Food Name	Categorical #	Calories	Apple	Chicken	Broccoli	Calories
Apple	1	95	1	0	0	95
Chicken	2	231	0	1	0	231
Broccoli	3	50	0	0	1	50

Figuur 41: Voorbeeld One-hot encoding

```

1 # voeg de categorieën toe als kolommen
2 dataset = pd.concat(
3     [dataset, pd.get_dummies(dataset['food_name'], prefix='food')],
4     axis=1)
5 # verwijder de food_name kolom
6 dataset.drop(['food_name'], axis=1, inplace=True)
7 # toon de eerste 5 rijen:
8 dataset.head()

```

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

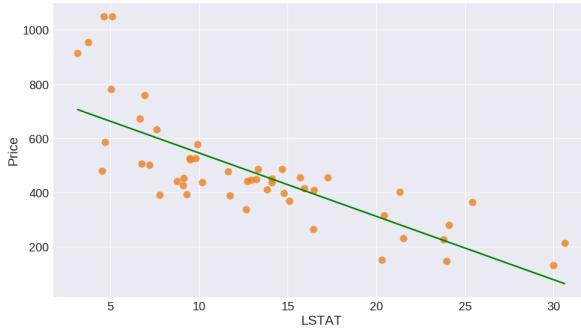
Figuur 42: Resultaat van dataset.head(), voor en na one-hot encoding

6 Underfitting & overfitting

6.1 Underfitting

Definitie 6.1 (Underfitting) *Underfitting treedt op wanneer een model de training data niet kan modeleren en ook niet kan generaliseren op nieuwe data.*

- *Het model is te 'simpel'*
- *Model met hoge bias*

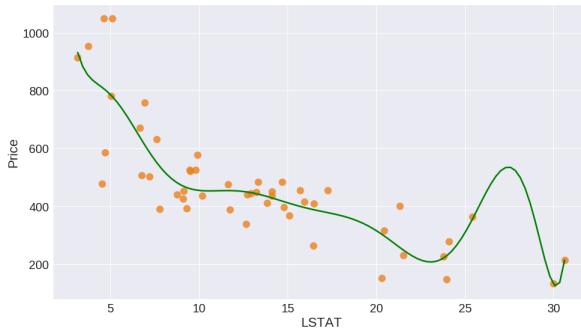


Figuur 43: Het model werkt voor sommige variabelen maar voor velen ook niet

6.2 Overfitting

Definitie 6.2 (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 data worden opgepikt*
- *Model met een hoge variance*

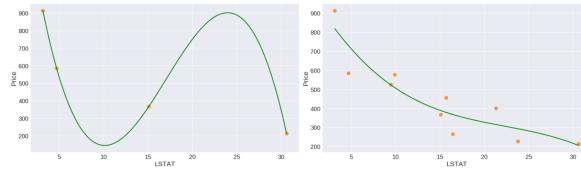


Figuur 44: Model met order ≈ 15

6.2.1 Impact van de grootte van de dataset

Afhankelijkheid van de grootte van de dataset (aantal observaties m):

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



Figuur 45: Voorbeeld: 3de orde polynoom

6.3 Regularisatie (regularisation)

Definitie 6.3 (Regularisatie) Methode om de mate van bias van een hypothese te regelen en een goed evenwicht te vinden tussen underfitting en overfitting.

$$J(\theta) = \frac{1}{2 \cdot m} \sum_{i=1}^m (h_\theta(x_i) - y_i)^2 + R(\theta) \quad (11)$$

Definitie 6.4 ($R(\theta)$) $R(\theta)$ is de regularisatie-term.

Dit is een extra kostenterm die het gebruik van hogere orde features afstrafst tenzij ze de globale kostenfunctie doen dalen.

6.3.1 Voorbeeld regularisatie

$$J(\theta) = \frac{1}{2 \cdot m} \sum_{i=1}^m (h_\theta(x_i) - y_i)^2 + \lambda \theta \theta^\top$$

- $\theta = \{\theta_1, \dots, \theta_n\}$
 - intercept θ_0 wordt meestal niet geregulariseerd
- λ is een tuning parameter (hyper parameter), we moeten die zelf vinden
 - $\lambda = 0 \Rightarrow$ geen regularisatie

- $\lambda = \inf \Rightarrow \theta = 0$
- λ tussenin regelt de mate van regularisatie.

De tuningparameter λ regelt de complexiteit van de hypothese:

- Kleine waarde voor λ : lage bias, hoge variantie (overfitting)
- Grote waarde voor λ : hoge bias, lage variantie (underfitting)

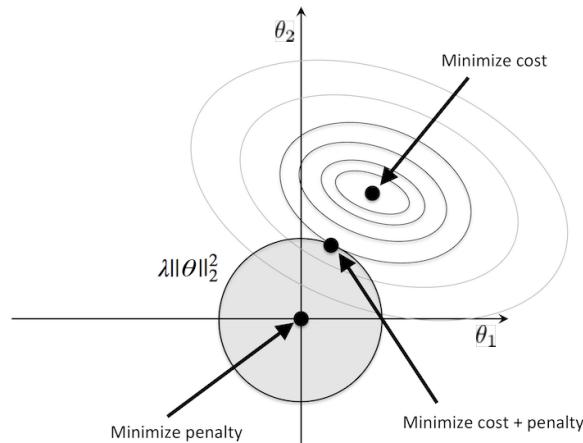
Afhankelijk van hoe R gedefinieerd wordt is er een andere benaming voor de regularisatie:

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

6.3.2 Regularisatie met L2 norm

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

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

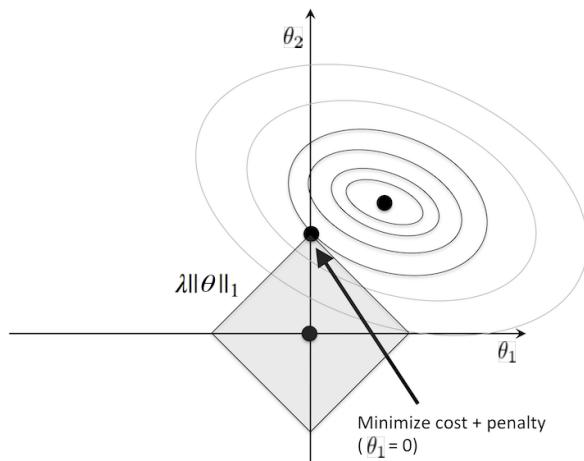


Figuur 46

6.3.3 Regularisatie met L1 norm

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

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



Figuur 47

6.3.4 Voorbeeld regularisatie op huizenprijzen

Via Ridge of Lasso regressie met regularisatieparameter α

- Hoe groter α , hoe sterker de regularisatie en dus hoe simpler het model
- Hoe kleiner α , hoe zwakker de regularisatie en dus hoe complexer het model

```

1 regmodel = Ridge(alpha=0.14, tol=0.0001, fit_intercept=True)
2 regmodel.fit(X_train, y_train)
3 regmodel.score(X_test, y_test)
4 >> 0.79834480089914472
5
6 lregmodel = Lasso(alpha=0.5, tol=0.0001, fit_intercept=True)
7 lregmodel.fit(X_train, y_train)
8 lregmodel.score(X_test, y_test)
9 >> 0.8437113338085345

```