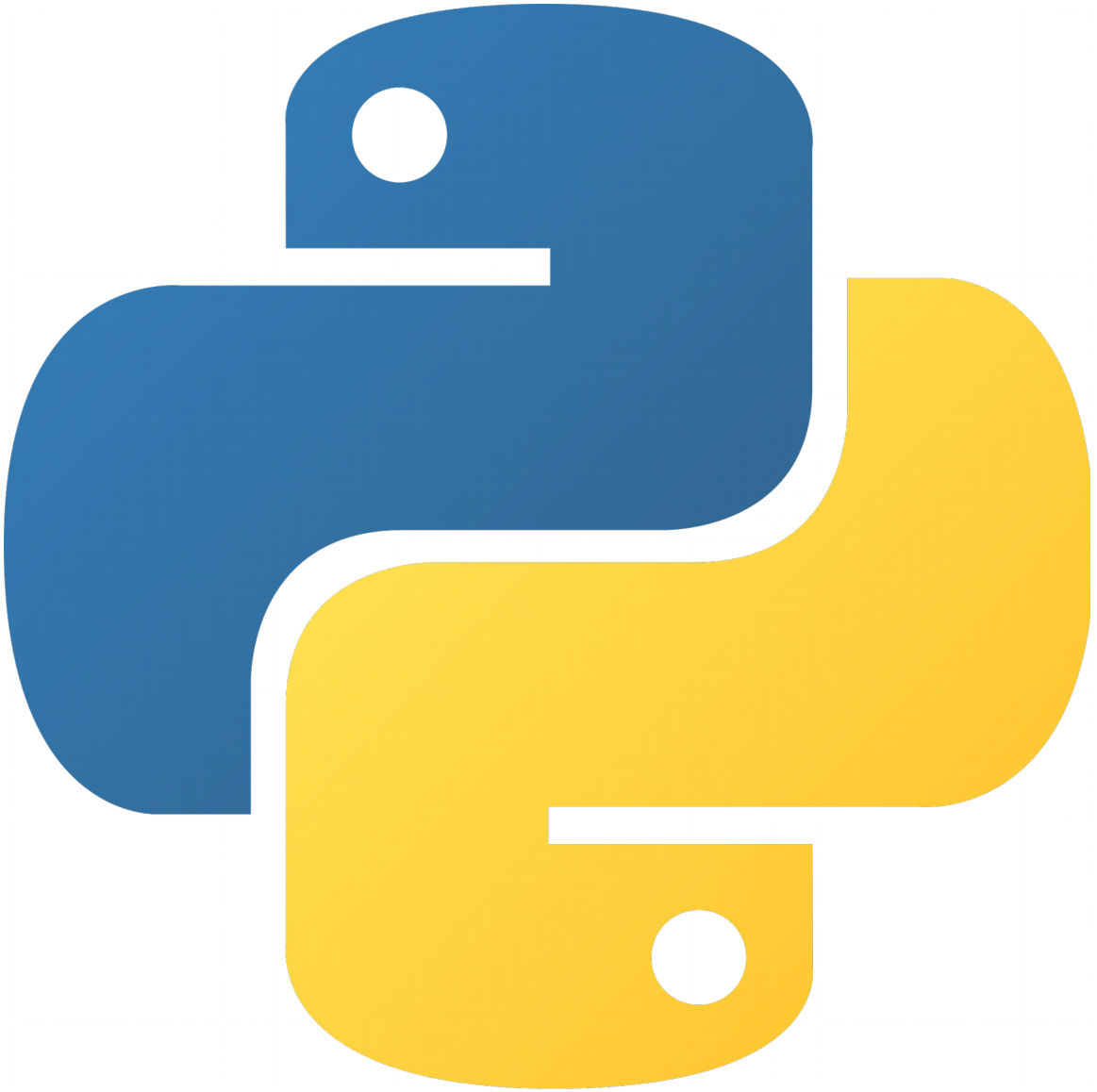
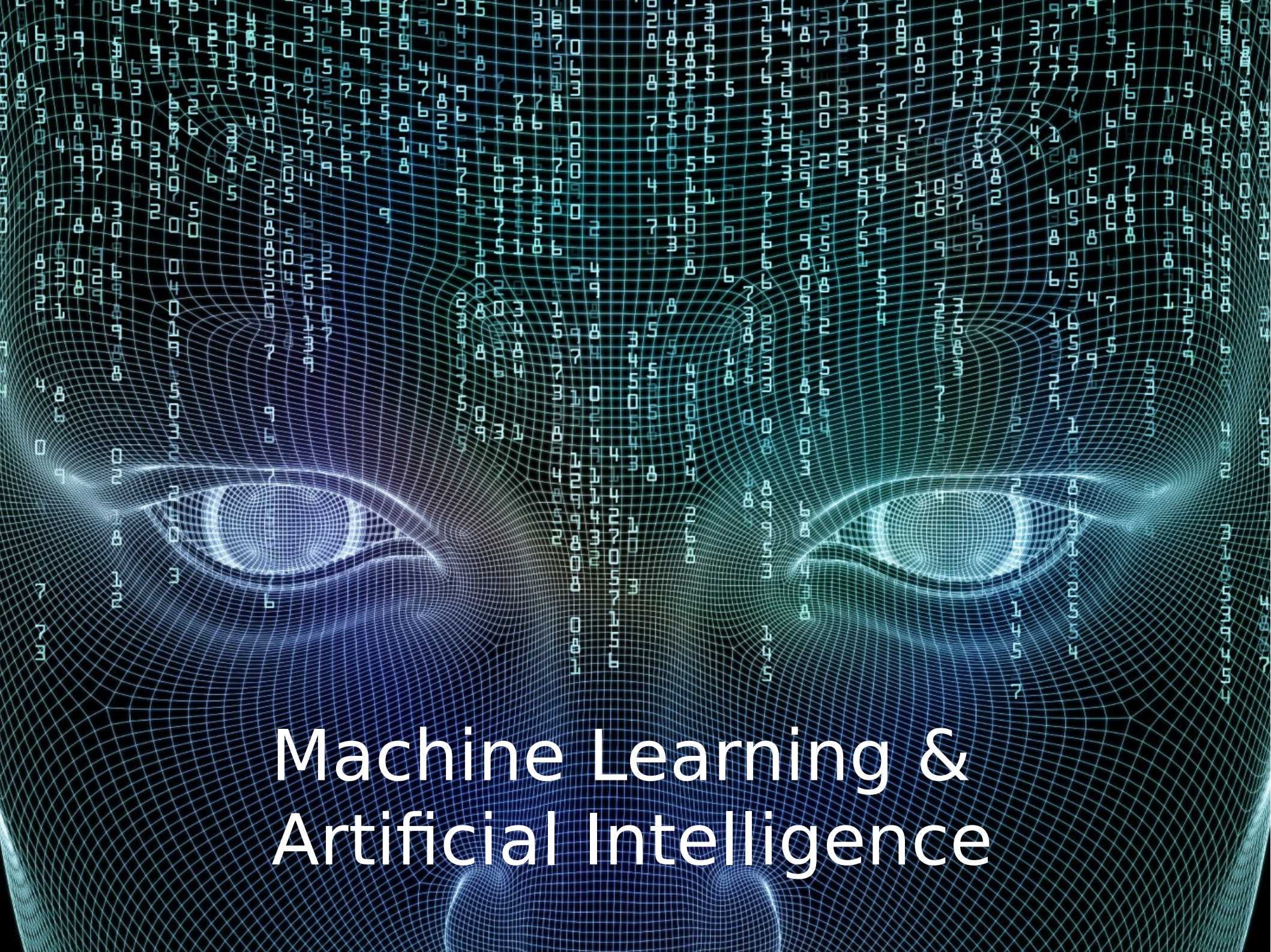
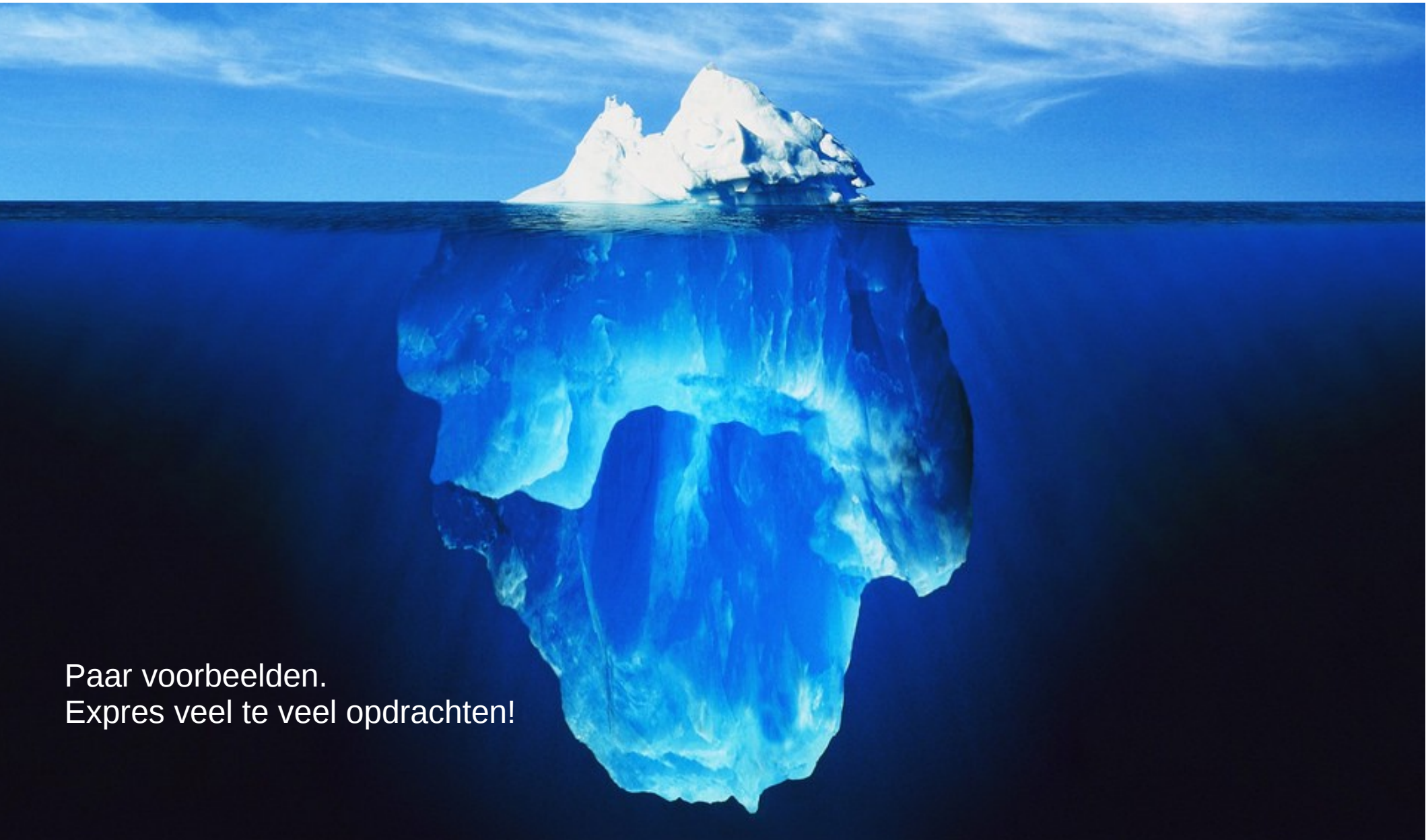
Intro Python voor Data Science



Jansen en Van der Zee
Marcel Haas, 2018


Machine Learning &
Artificial Intelligence

Machine Learning



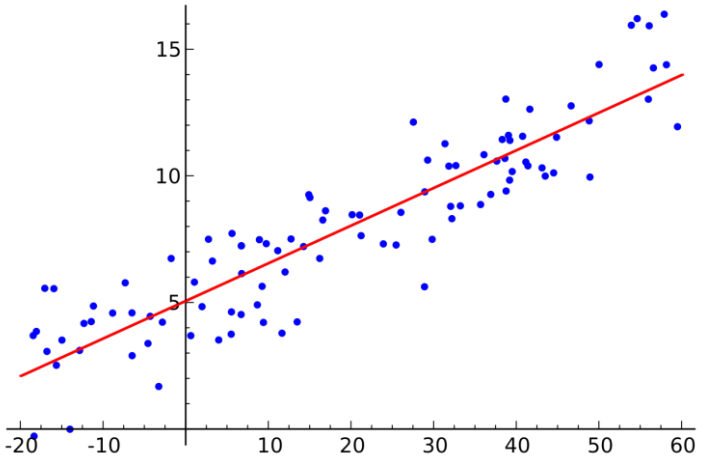
Paar voorbeelden.
Expres veel te veel opdrachten!

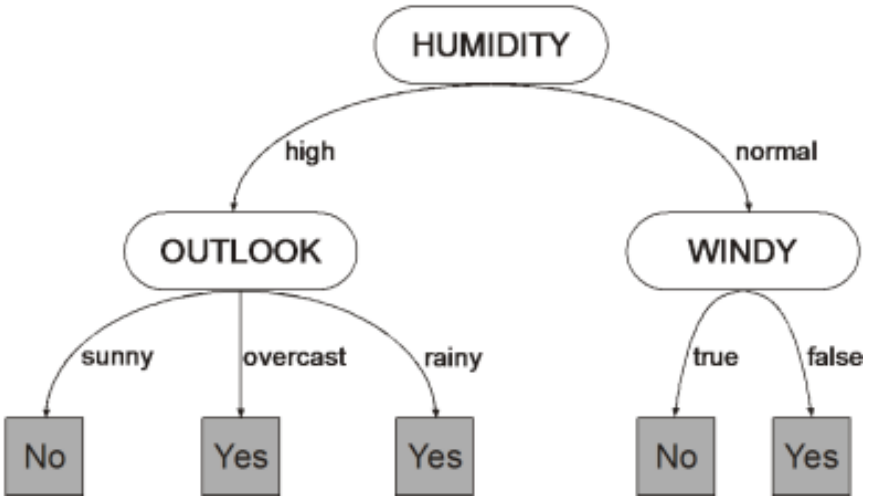
3 smaken

- Supervised Learning
  - Gewenst resultaat bekend
  - Classificatie, voorspelling
- Unsupervised Learning
  - “Zoeken naar het onbekende”
  - Clustering, dimensiereductie, patroonidentificatie
- Reinforcement Learning
  - Einddoel duidelijk, door middel van straf/beloning strategie ontwikkelen

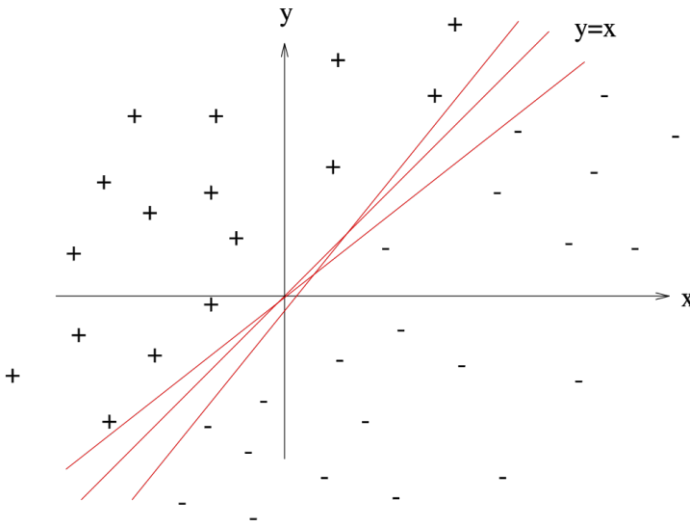
Smaak 1: Supervised Learning

Regressie



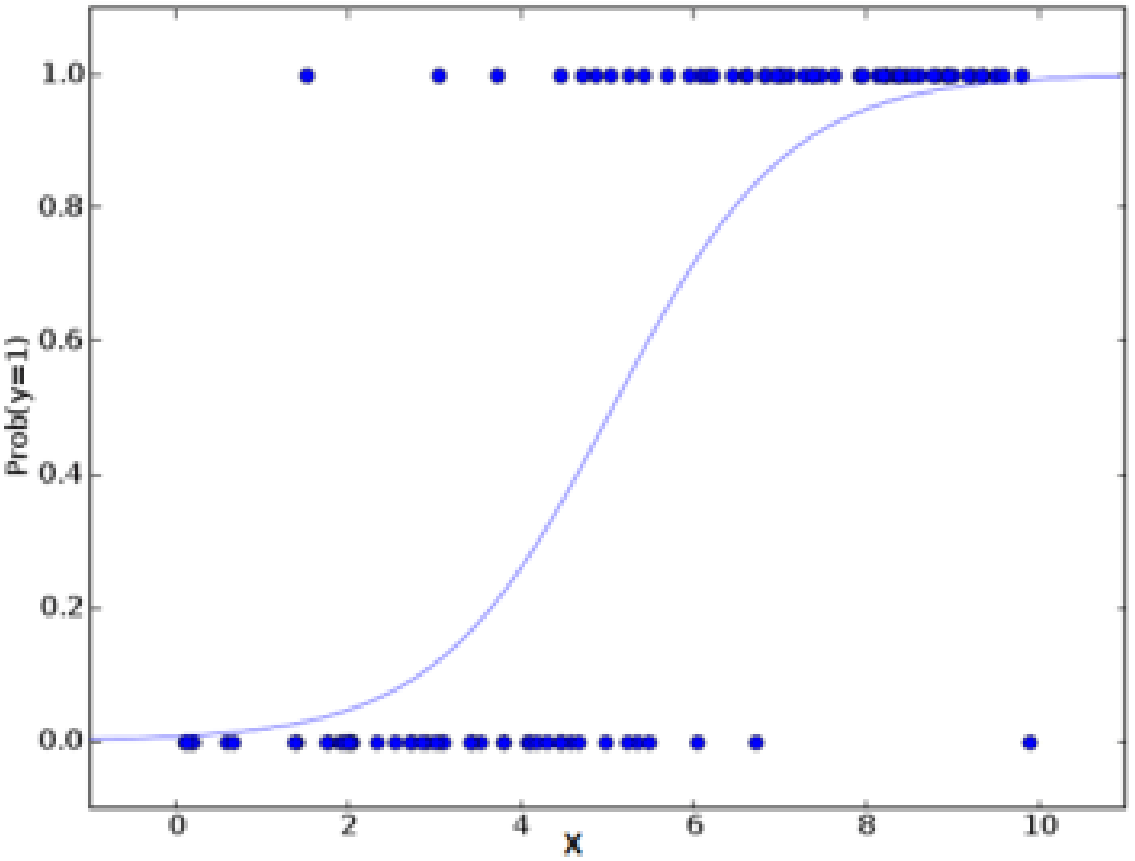


Beslissingen



Classificatie

Voorbeeld: Logistische regressie



Python en de scikit-learn API

- from sklearn.module_naam import functionaliteit
- Modules gegroepeerd per “soort”:
  - datasets: het verkrijgen van data
  - linear_models: bevat lineaire modellen
  - neighbors: nearest neighbor modellen
  - svm: Support Vector Machines
  - model_selection: functionaliteit om modellen te beoordelen
  - tree: beslisbomen etc.
  - preprocessing: voorbereiden van data

Verschillende modellen, zelfde attributen/methoden

- De modellen hebben namen met hoofdletters:

```
In [ ]: from sklearn.linear_model import LinearRegression
```

- Je moet ze initialiseren:

```
In [ ]: regressor = LinearRegression()
```

- Daarna zien alle attributen en methoden van verschillende modellen er hetzelfde uit:

```
In [ ]: regressor.fit(X, y)
        regressor.predict(newX)
        regressor.coef_
        regressor.get_params()
```

Het “scoren” van modellen - R^2

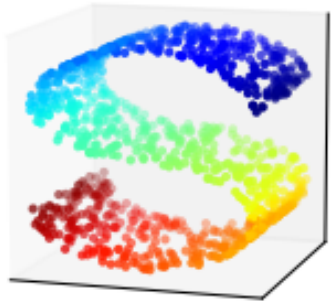
“De fractie van de variantie in de data die wordt verklaard door het model”

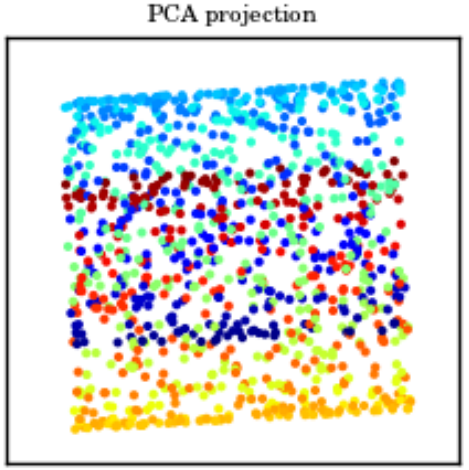
$$R^2 = 1 - \frac{\Sigma(y - y_{\text{fit}})^2}{\sigma^2}$$

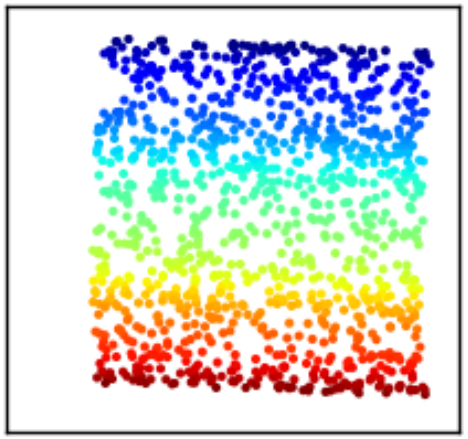
In scikit-learn: model.score(x, y)

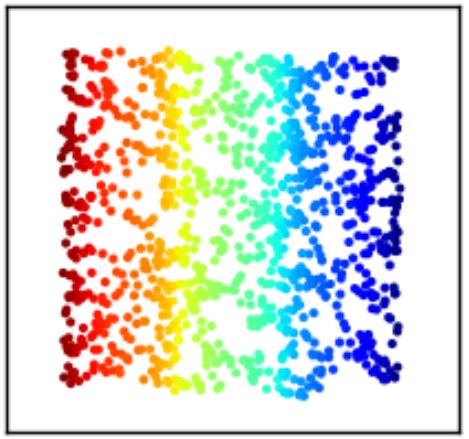
Smaak 2: Unsupervised Learning

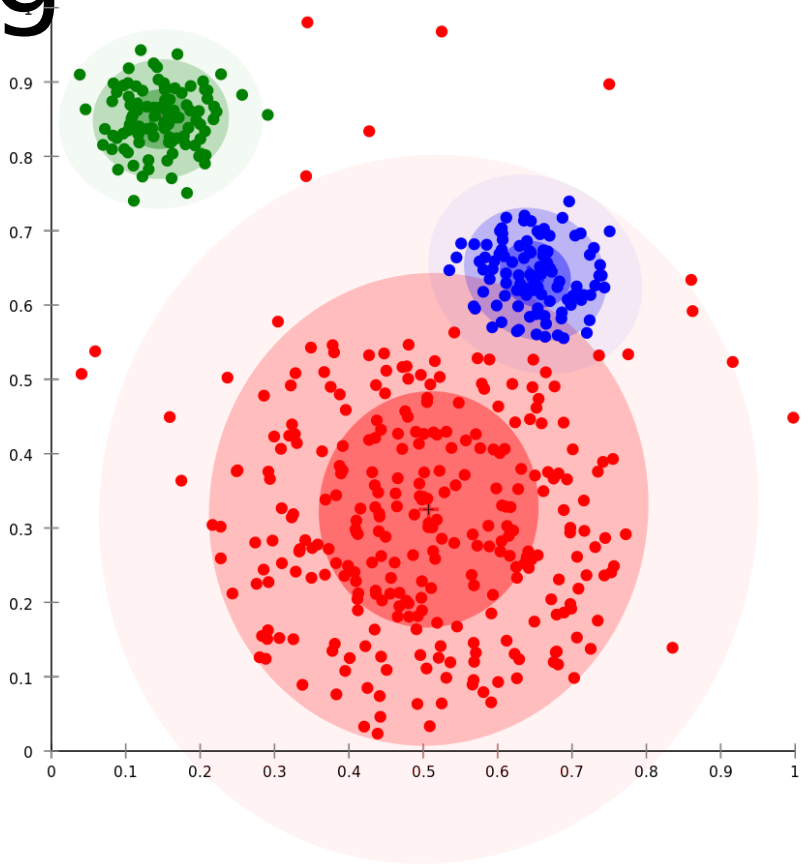
Structuur vinden en dimensiereductie





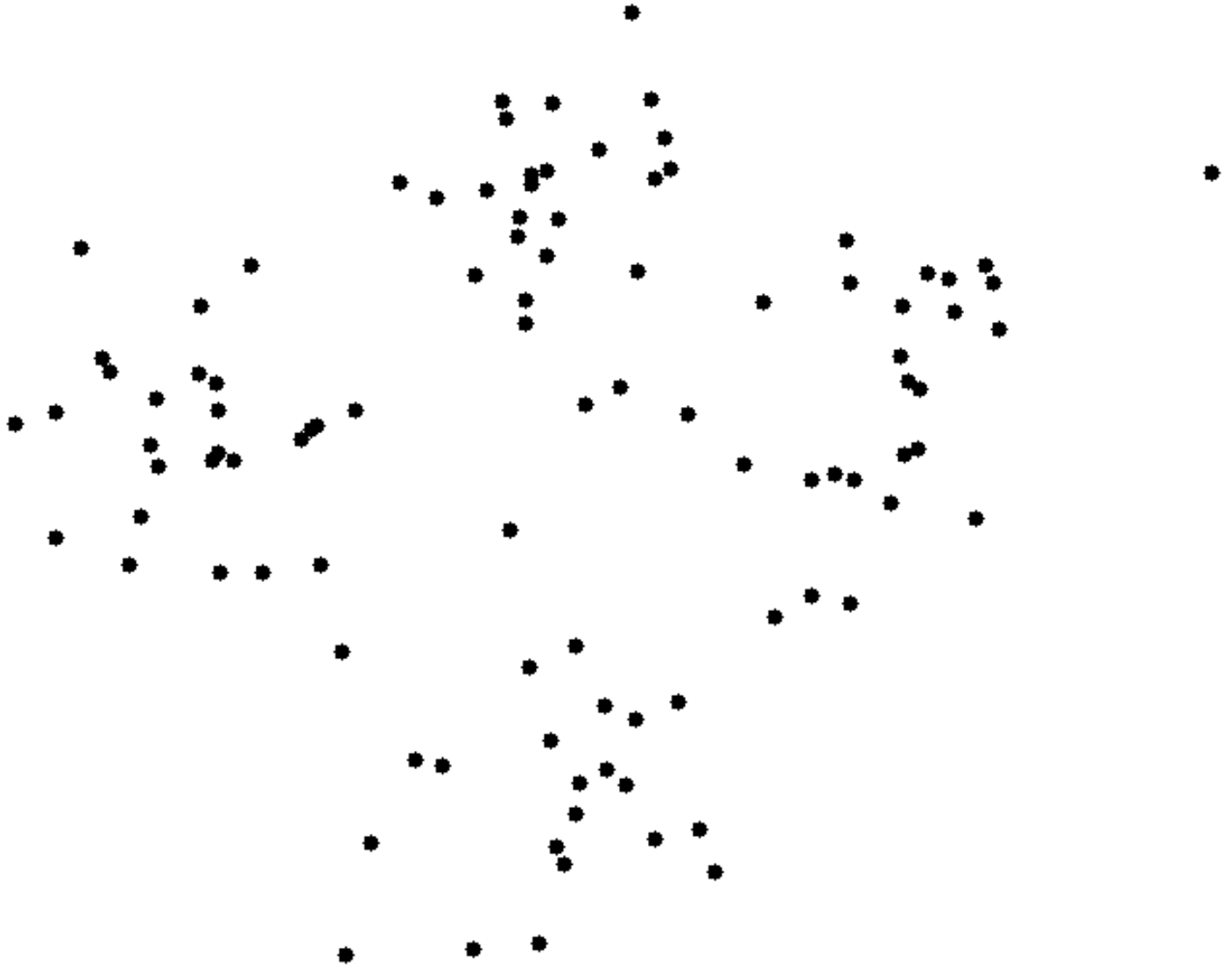






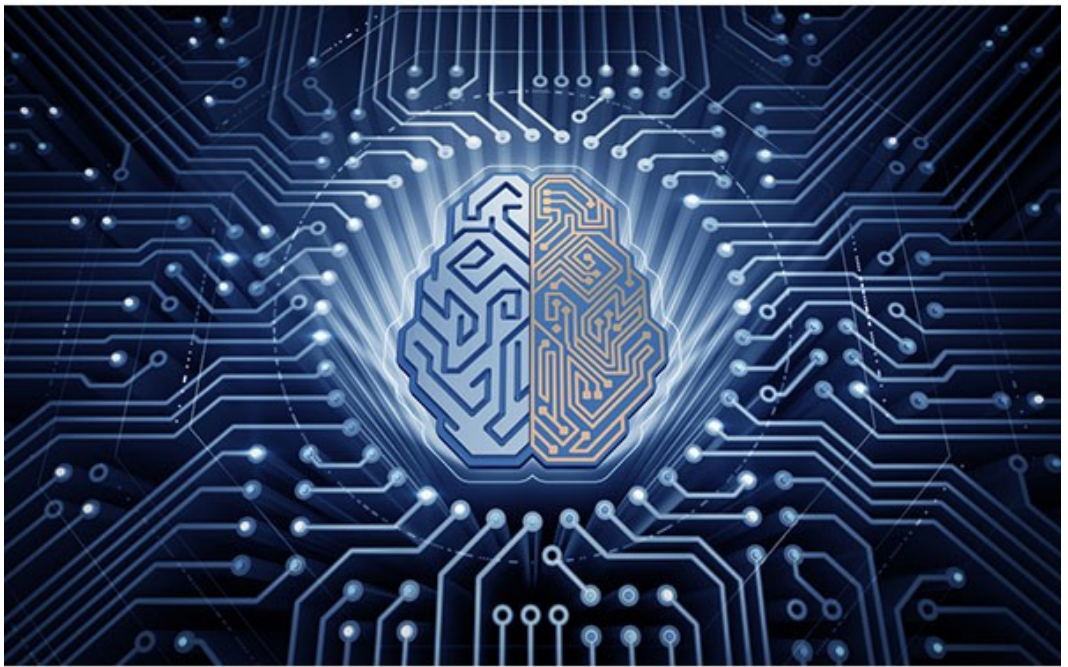
Clustering

Voorbeeld: K-Means Clustering



Neurale netwerken

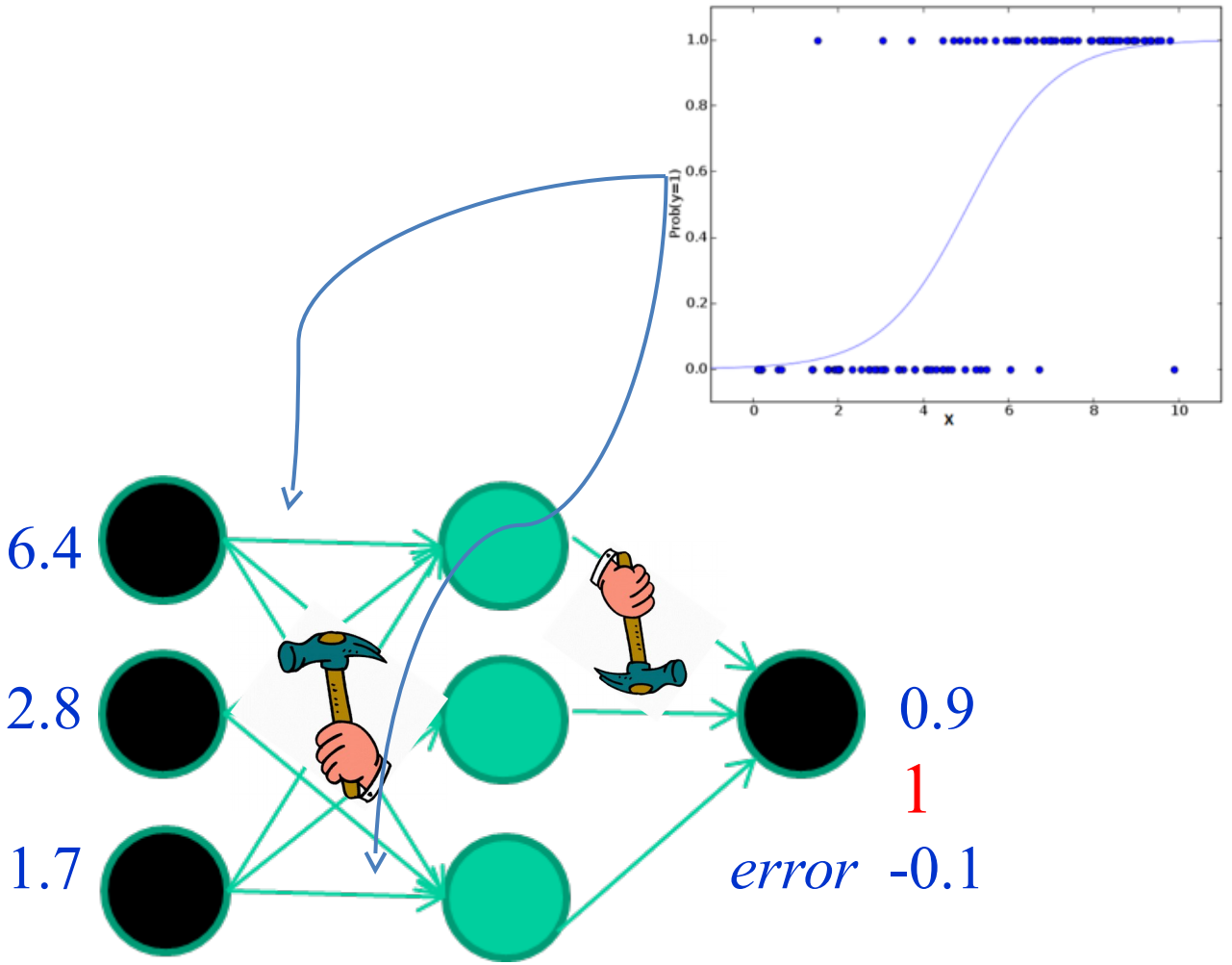
- Nonlineariteit
- Multimodaliteit
- Hoge dimensionaliteit
- De wereld is ingewikkeld.



Neurale netwerken

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			



Adjust weights based on error

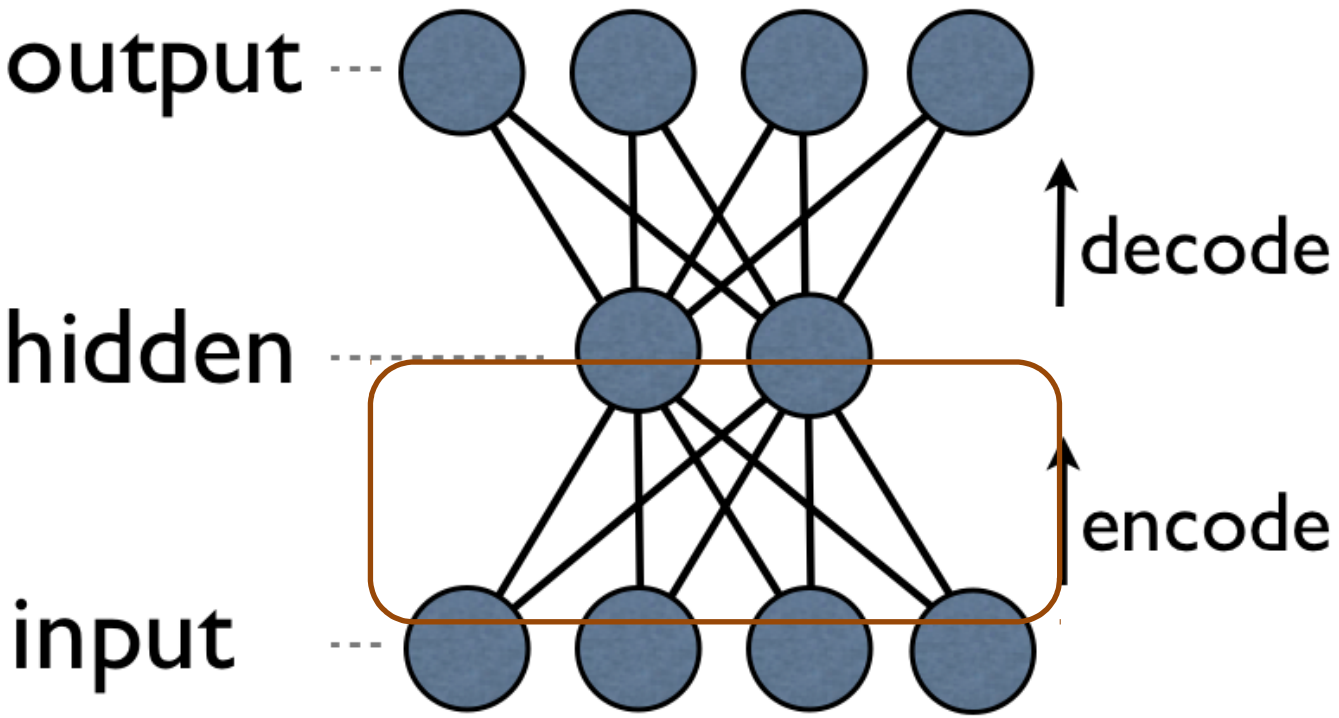
(Deep) neural nets in sklearn

- Only the multi-layer perceptron
  - Forward-fed
  - Fully connected
  - Choice of activation function
  - Number of hidden layers up to you
  - Number of neurons in each layer up to you

```
[ ]: from sklearn.neural_network import MLPClassifier, MLPRegressor
     clf = MLPClassifier(hidden_layer_sizes=(10, 10, 2), activation='relu', ....)
```

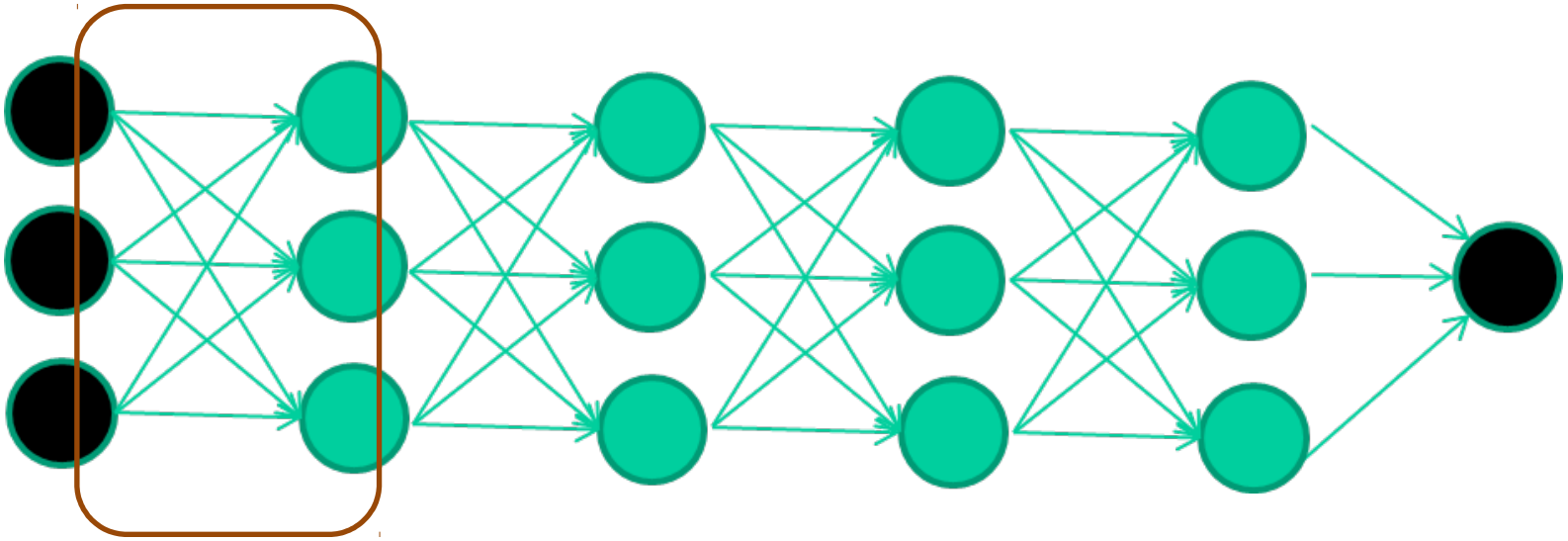
- Fully consistent with other models

Auto-encoders



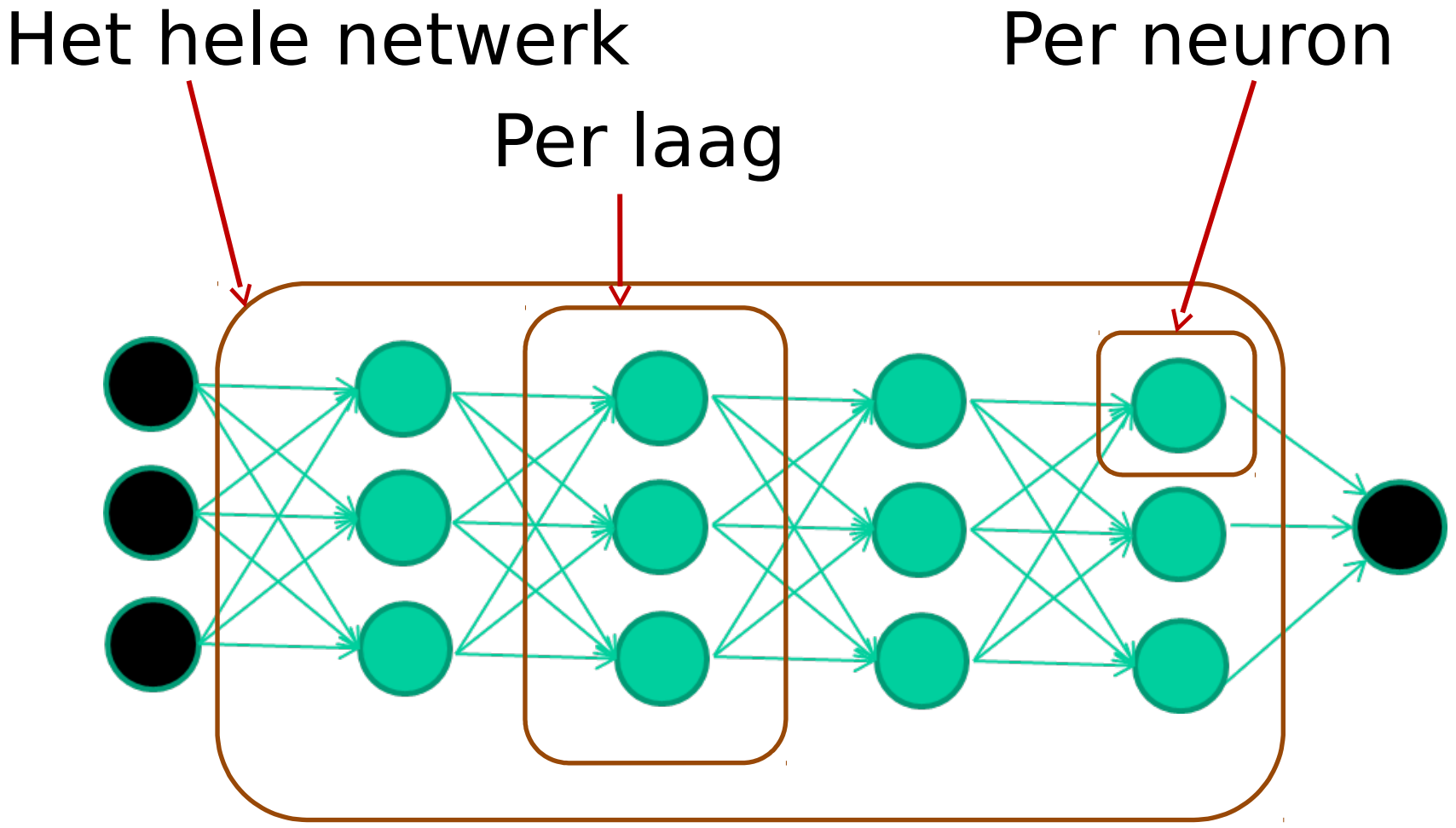
Output = Input!

The new way to train multi-layer NNs...



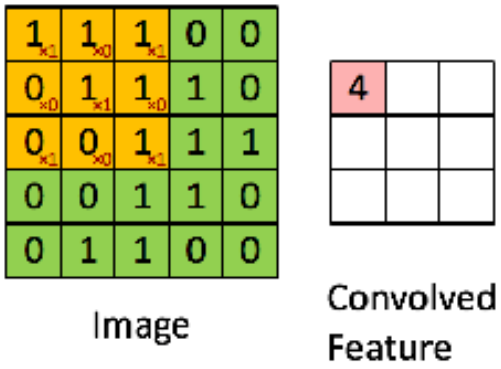
Elke laag wordt getraind als auto-encoder.
Daarmee wordt elke laag gedwongen goede features uit de vorige laag te destilleren.

High vs Low level implementatie

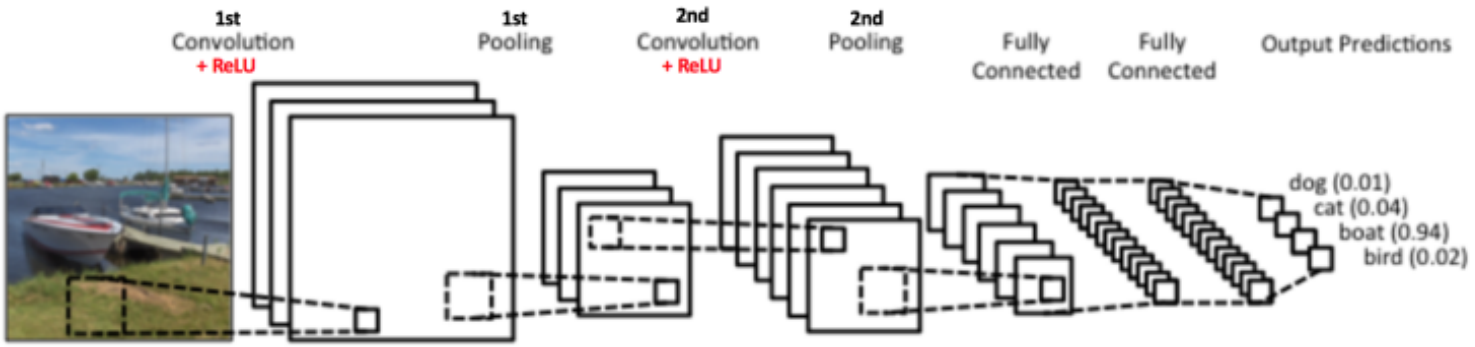


Verschillende architecturen

- Convolutional Neural Networks (CNN)

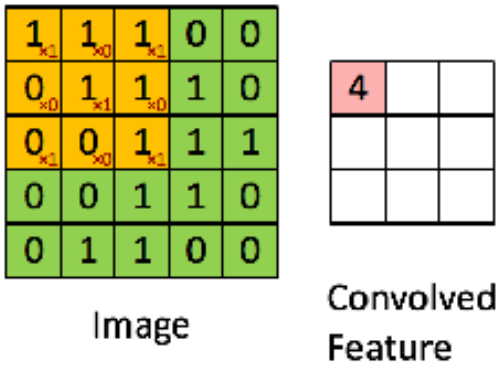


Voor pooling zijn allerlei filters mogelijk: sum(), max(), median(), ...

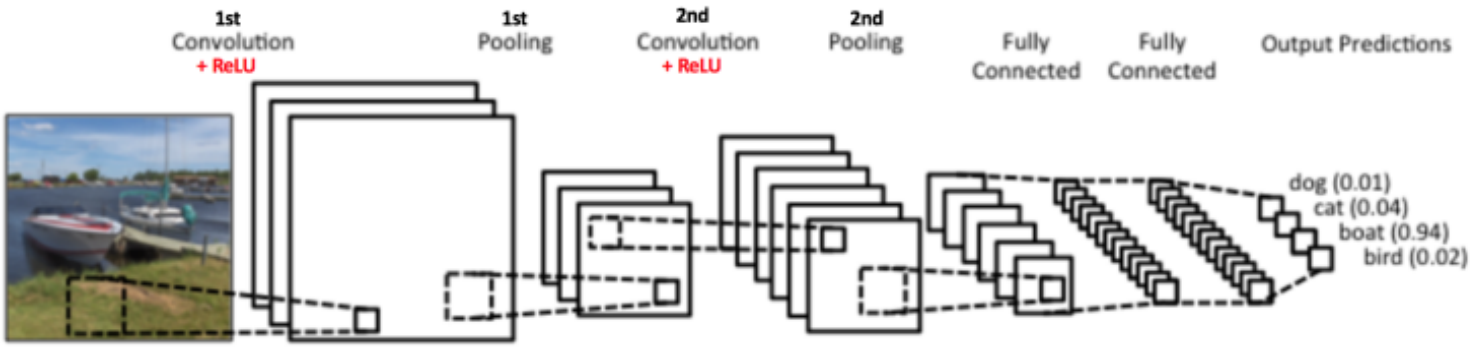


Verschillende architecturen

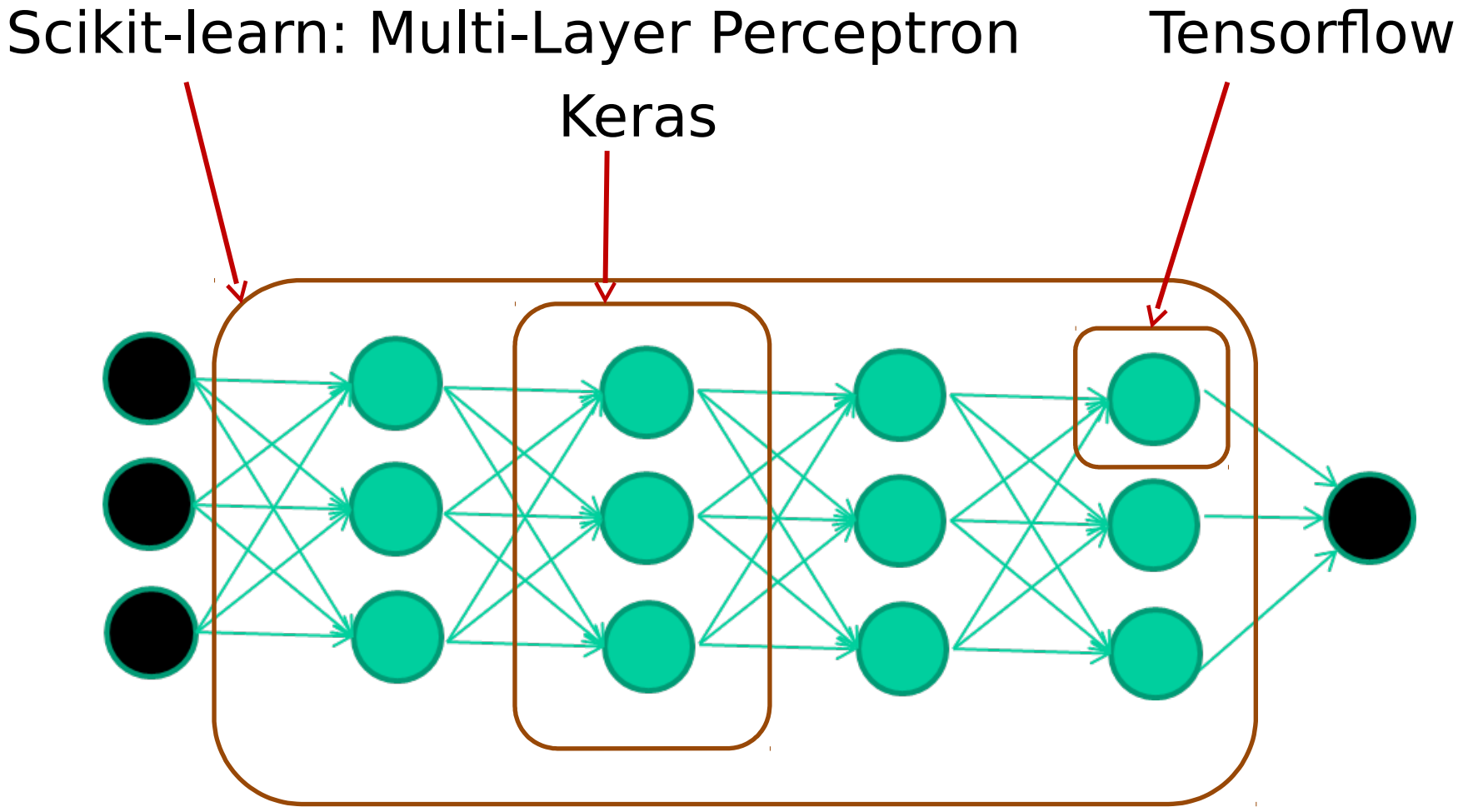
- Convolutional Neural Networks (CNN)



Voor pooling zijn allerlei filters mogelijk: sum(), max(), median(), ...



High vs Low level implementatie



Successen op de wereld

[25]:
```
# Met Sequential zet je al je lagen achter elkaar
model = tf.keras.models.Sequential([
  tf.keras.layers.Dense(200, activation=tf.nn.relu),
  tf.keras.layers.Dense(50, activation=tf.nn.relu),
  tf.keras.layers.Dense(50, activation=tf.nn.relu),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
# Door een model te compileren definieer je hoe het netwerk gefit moet gaan worden.
# Dat gebeurt nog niet.
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Pas met de fit() methode gebeurt het fitten, waarna het model ook geevalueerd kan worden.
model.fit(xtr, ytr, epochs=10)
print(model.evaluate(xtr, ytr))
model.evaluate(x, y)
```

Referenties

- Een Machine Learning workshop voor twee dagen in het Engels: https://github.com/harcel/ML_workshop
- Veel meer referenties daarin.