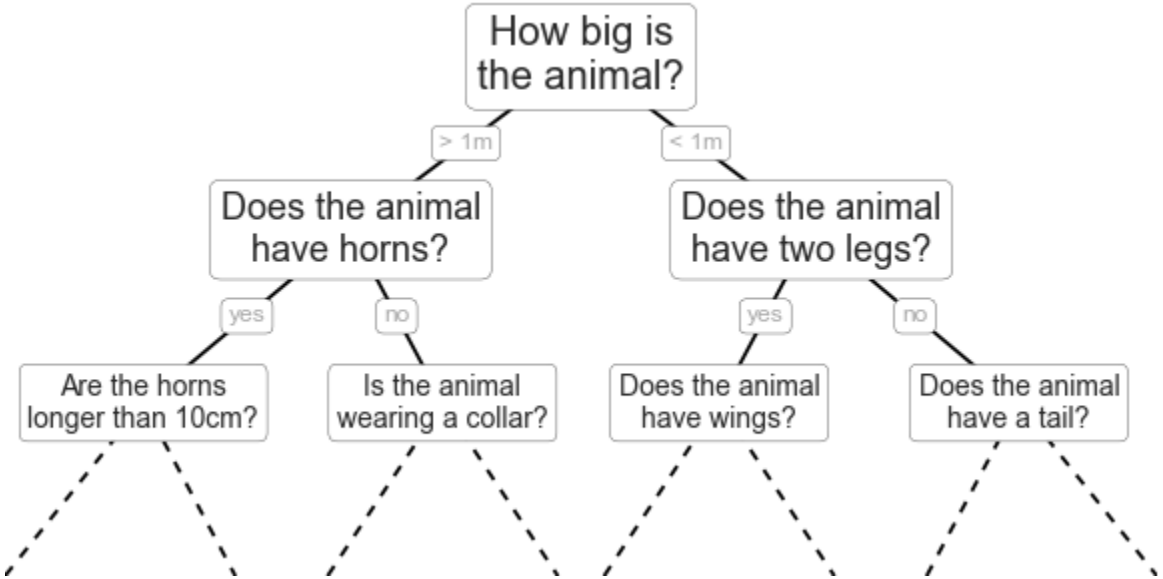
Decision trees

Supervised algoritme voor classificatie en regressie

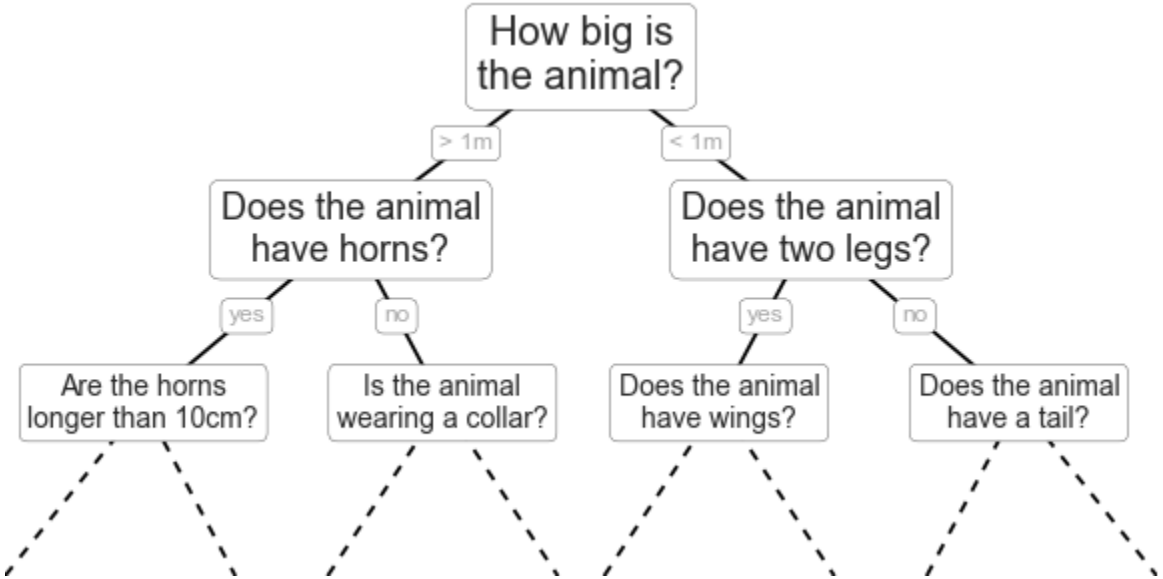
Decision trees introductie

- Intuïtieve manier voor classificatie
  - Feitelijk het stellen van een serie van vragen
  - (Bijna altijd) binair antwoord
- Bestaande uit
  - Nodes - vragen
  - Leaves - classificatie



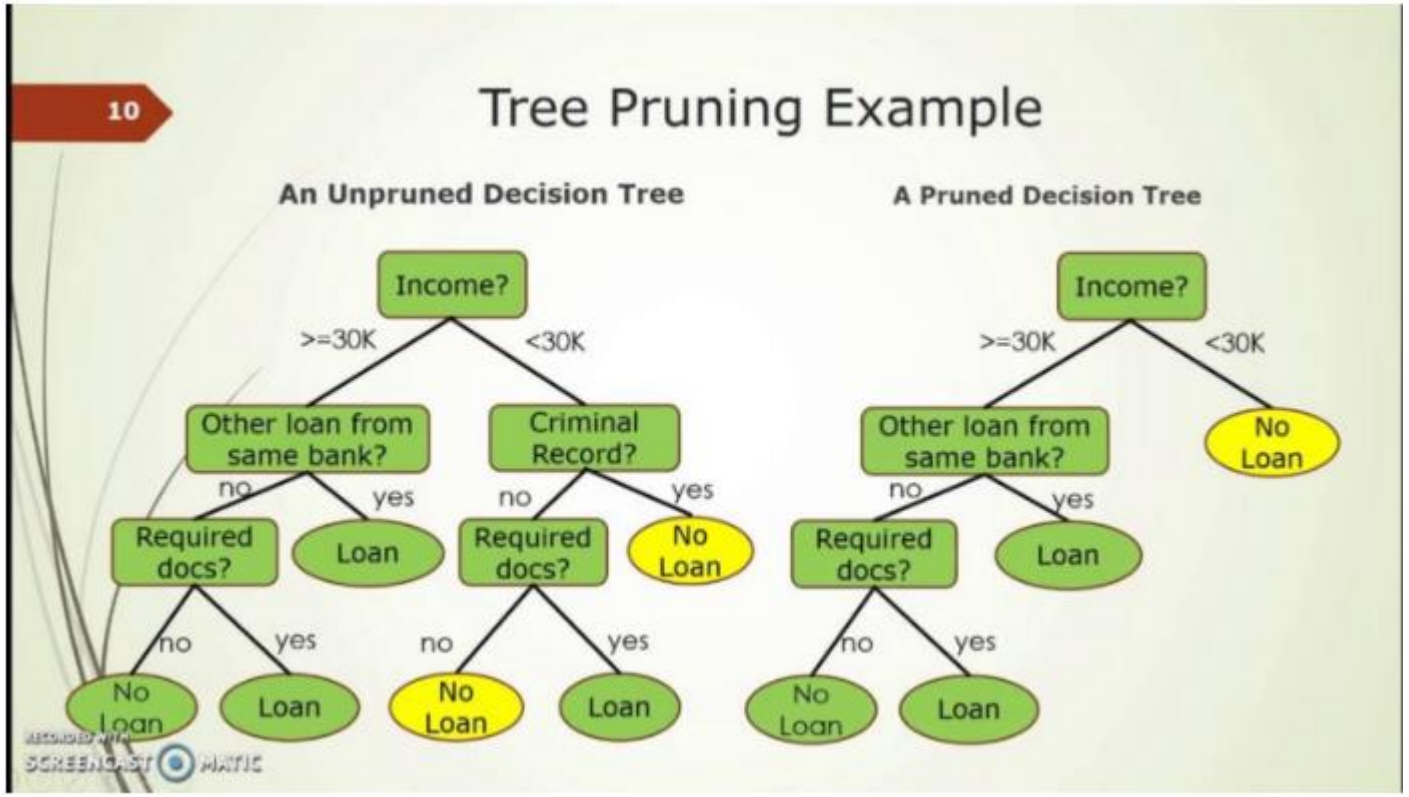
Decision trees effectiviteit

- Per “node” in de boom wordt geprobeerd het aantal oplossingen door 2 te delen (splitting)
  - Per node dus slim de juiste vragen stellen
- Doel is om een zo klein mogelijke boom te vinden

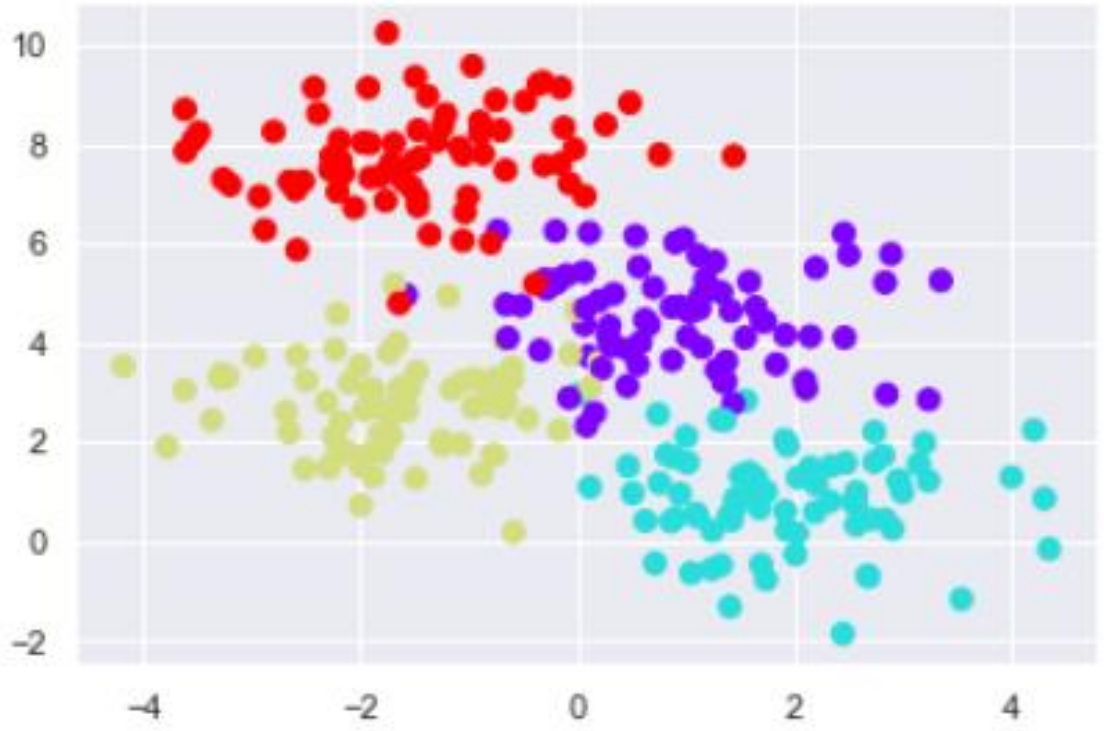


Decision trees - Pruning

- Het inkorten van de boom door het verwijderen van nodes welke een lage bijdrage leveren aan de oplossing

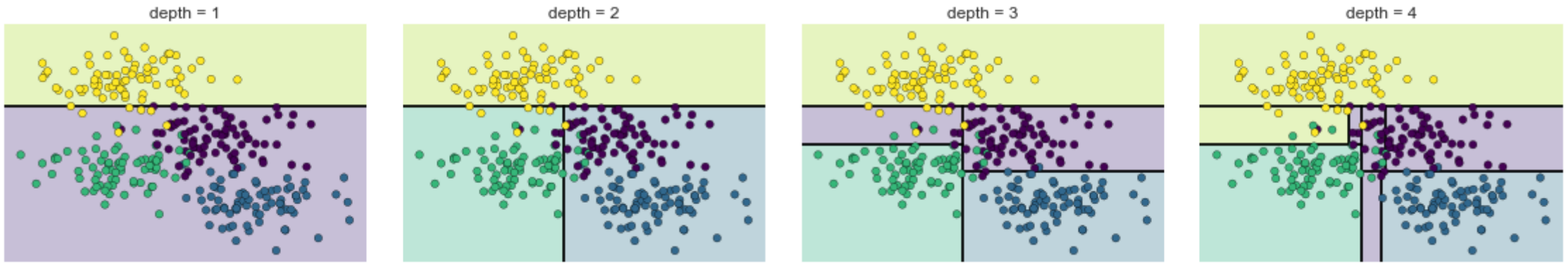


Voorbeeld

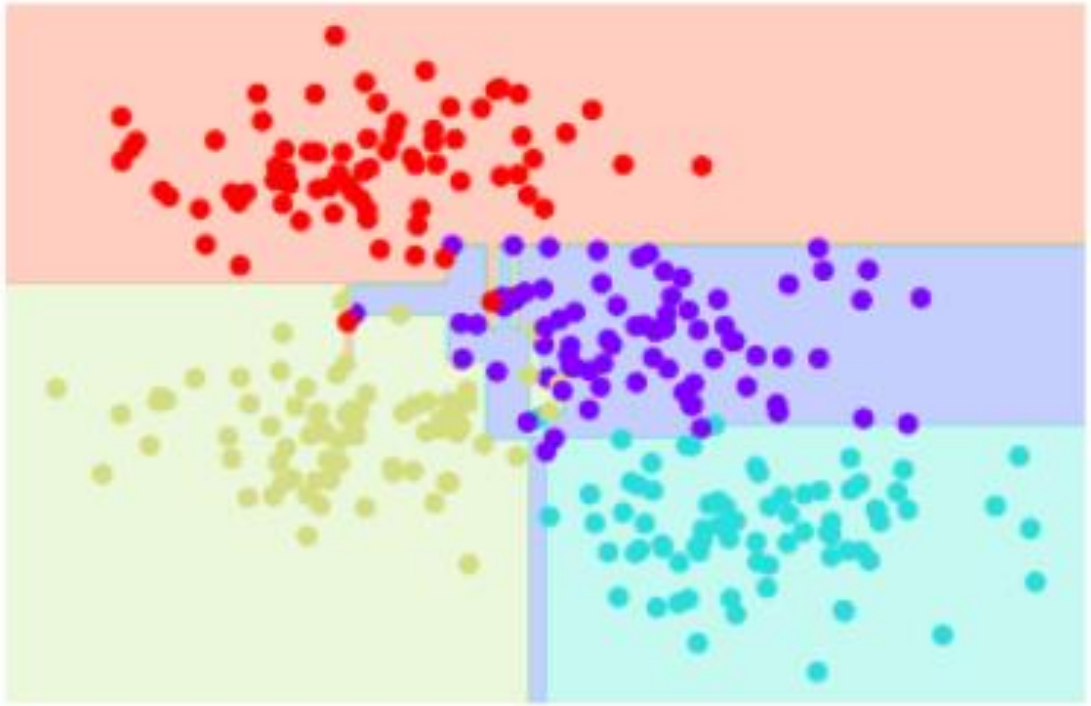



```
%matplotlib inline
import numpy as np
from helpers import helper
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()

from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples=300, centers=4,
                  random_state=0, cluster_std=1.0)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='rainbow');
```

Stap voor stap



Decision trees oplossing



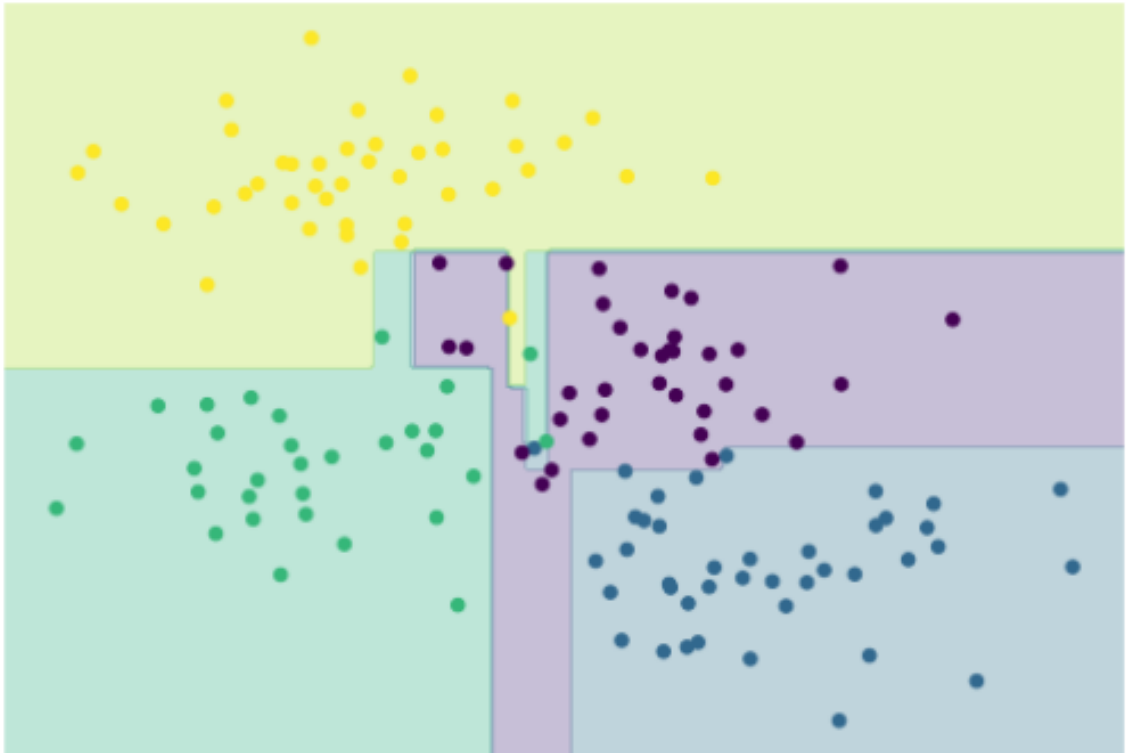
```python
from sklearn.tree import DecisionTreeClassifier

# Ook hier weer een hulp functie:
def visualize_classifier(model, X, y, ax=None, cmap='rainbow'):
    ax = ax or plt.gca()

    # Plot the training points
    ax.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=cmap,
               clim=(y.min(), y.max()), zorder=3)
    ax.axis('tight')
    ax.axis('off')
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # fit the estimator
    model.fit(X, y)
    xx, yy = np.meshgrid(np.linspace(*xlim, num=200),
                         np.linspace(*ylim, num=200))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

    # Create a color plot with the results
    n_classes = len(np.unique(y))
    contours = ax.contourf(xx, yy, Z, alpha=0.3,
                           levels=np.arange(n_classes + 1) - 0.5,
                           cmap=cmap, clim=(y.min(), y.max()),
                           zorder=1)

    ax.set(xlim=xlim, ylim=ylim)

visualize_classifier(DecisionTreeClassifier(), X, y)
```

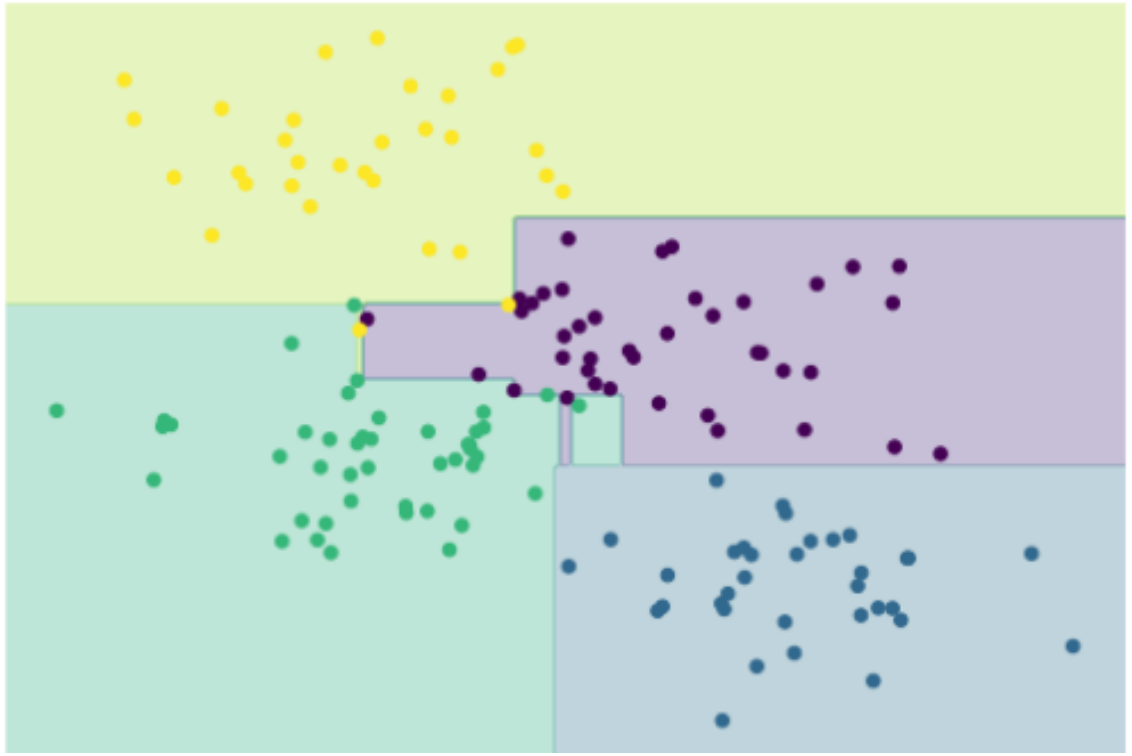
Overfitting

- Als de boom te veel wordt gestuurd op de details van de specifieke data punten en niet op de details behorende bij de labels in het algemeen
  - De details zijn dan feitelijk ruis

Overfitting - voorbeeld

```
model = DecisionTreeClassifier()

fig, ax = plt.subplots(1, 2, figsize=(16, 6))
fig.subplots_adjust(left=0.0625, right=0.95, wspace=0.1)
helper.visualize_tree(model, X[::2], y[::2], boundaries=False, ax=ax[0])
helper.visualize_tree(model, X[1::2], y[1::2], boundaries=False, ax=ax[1])
```

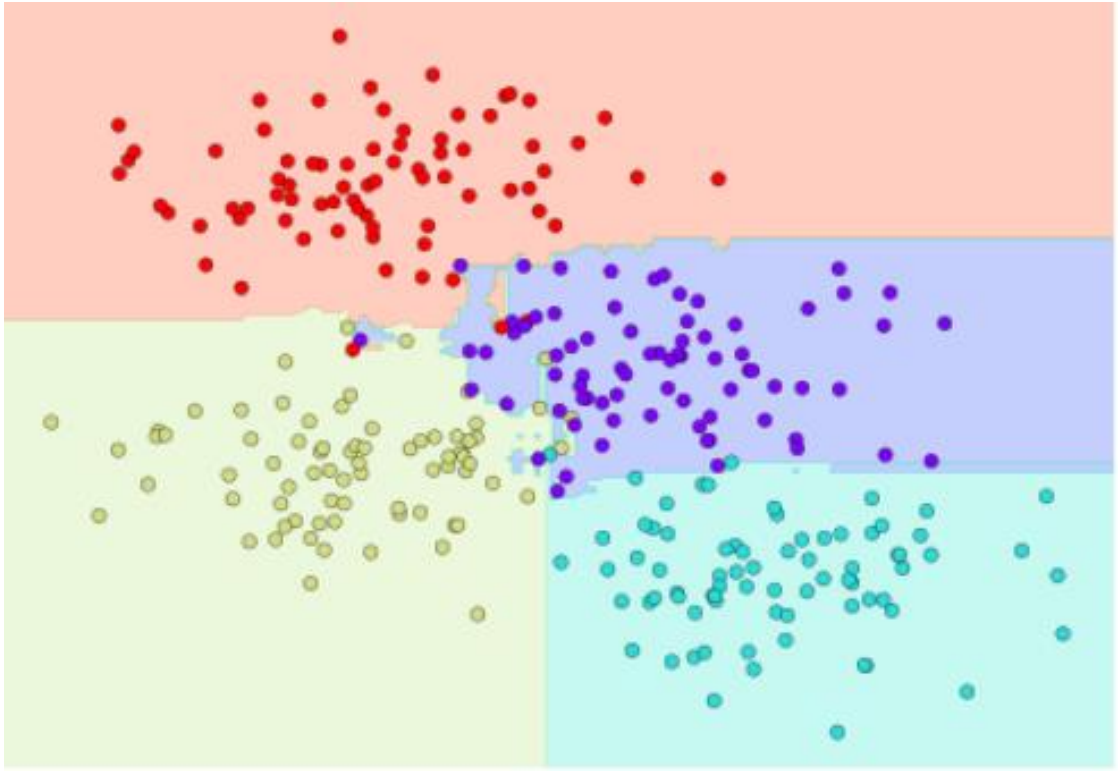




Ensembles of estimators - Bagging

- Combineren van verschillende overfitting estimators
  - Effect van overfitting reduceren
  - Verschillende opgebouwde modellen, gecombineerd om tot het gemiddelde van alle uitkomsten te komen
  - Elke individuele estimator wordt opgebouwd met een subset van de gegeven data set
- Random forest is het combineren van verschillende decision trees oplossingen

Random forest

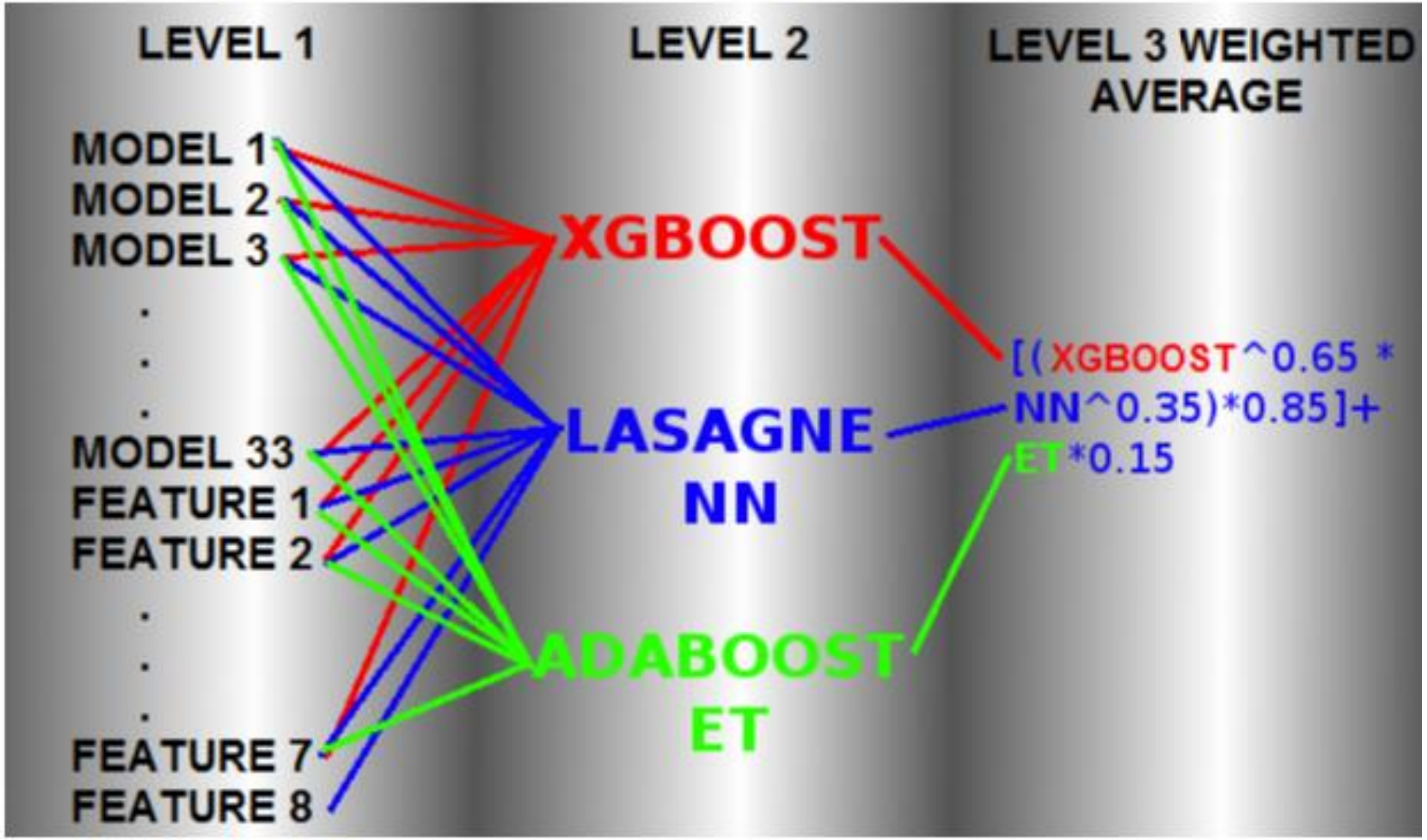


```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, random_state=0)
visualize_classifier(model, X, y);
```

Code voorbeeld

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.datasets import make_classification
```

```
>>> X, y = make_classification(n_samples=1000, n_features=4,
...                            n_informative=2, n_redundant=0,
...                            random_state=0, shuffle=False)
>>> clf = RandomForestClassifier(n_estimators=100, max_depth=2,
...                              random_state=0)
>>> clf.fit(X, y)
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=2, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
>>> print(clf.feature_importances_)
[0.14205973 0.76664038 0.0282433  0.06305659]
>>> print(clf.predict([[0, 0, 0, 0]]))
[1]
```


LEVEL 1
LEVEL 2
LEVEL 3 WEIGHTED AVERAGE
MODEL 1
MODEL 2
MODEL 3
.
.
.
MODEL 33
FEATURE 1
FEATURE 2
.
.
.
FEATURE 7
FEATURE 8
XGBOOST
LASAGNE NN
ADABOOST ET
[(XGBOOST^0.65 * NN^0.35)*0.85]+ ET*0.15

Oefeningen

- Pas titanic op de volgende punten aan:
  - Toevoegen RandomForestClassifier
  - Voor zowel SVM als RandomForest:
    - Speel met verschillende parameters en zie wat de invloed is op de score
    - Welke van de 2 heeft een hogere score?
  - Cross_Validation
  - Grid_Search