Analytical Computing

Lecture 6: Sklearn for regression and classification learning

hogeschool Windesheim

Radboud University

Agenda

- Machine learning
- Regressie: recap
- Classificatie
- Sklearn
- Model validatie
- Geavanceerde visualisaties
- Hands-on Sklearn

Planning

20-4-2021 **Git**

22-4-2021 **Array Handling**

28-4-2021 **Pandas data manipulation**

11-5-2021 **Visualization with MatplotLib**

12-5-2021 **StatsModels for Python statistics**

21-4-2021 **Kick-off opdracht**

27-5-2021 **Sklearn for regression and classification learning**

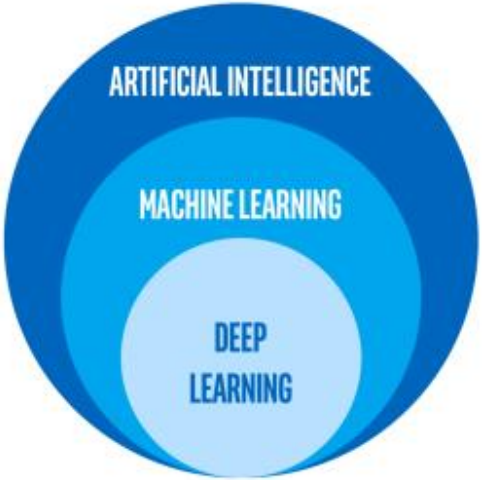
18-6-2021 **Eindpresentaties**

Machine learning

Machine learning

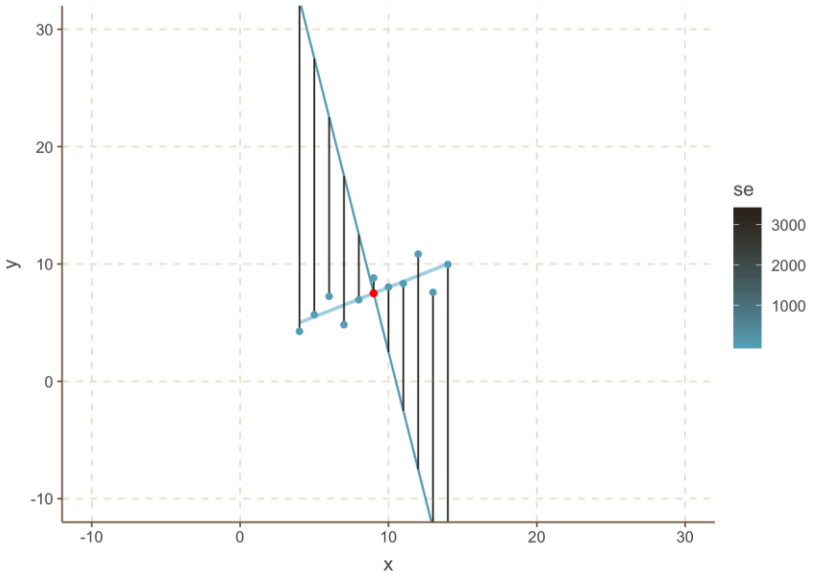
- *"Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed"*

- Three 'layers' of AI
  - AI is simply an overarching term for everything related to the automatic learning of systems, there is no 'real' clear definition
  - Machine learning is a more specific subset of AI
  - Deep learning is even more specific and focusses on training artificial neural networks (just like neurons in our brain for example). This way deep learning methods can find way more underlying structure in data



Machine learning

- Machine learning is the art of **updating weights**
- Every algorithm start with certain weights, these weights decide how *strong* the algorithm should learn new relationships
- Weights too low? Algorithm never reaches ‘optimum’ performance
- Weights too high? Algorithm overshoots optimum performance (and might never reach it at all)



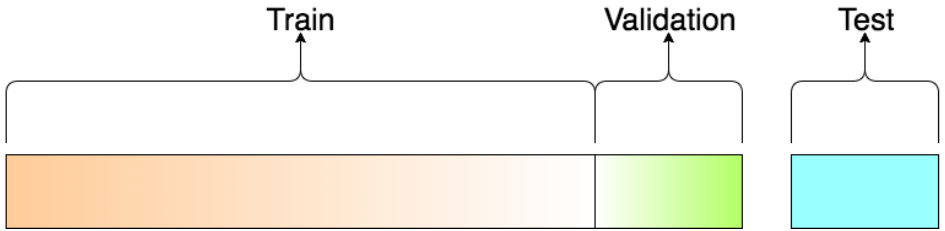
Machine learning

- Number of weights change based on your situation
  - Dataset choice
  - Task
  - Model infrastructure

- Weights can range from dozens to millions per algorithm
- Because trained machine learning models are simply sets of weights, we can easily and efficiently save and reuse these weights for future use!

Machine learning

- How do you learn a machine learning algorithm?
- Split your dataset into a *train, test* and *validation* split
  - In ‘most’ situations, 10% of your data is test, 10% validation and 80% is training data
- You train your model on your *training* data
- You validate the performance of your model on your *validation* data
- You test the final performance of your model on your *test* data



Machine learning

- Machine learning problems can be categorized within two domains
  - Predictive – we actually want to predict a value from our dataset
  - Descriptive – We want the algorithm to show us patterns in the data that we can not find in first glance

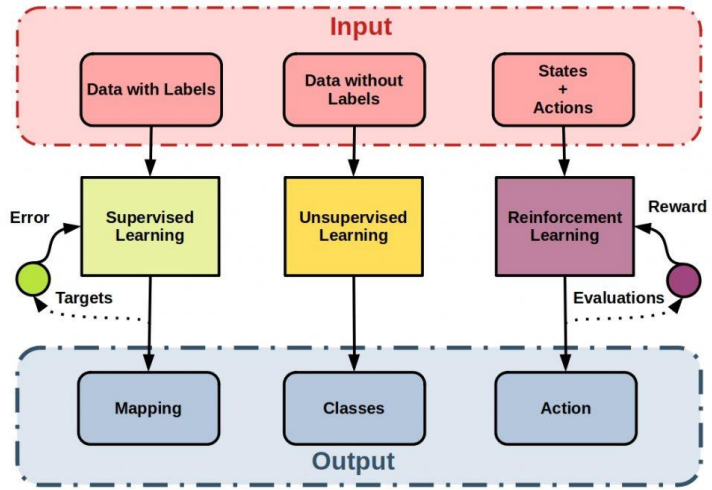
Machine learning

- Each domain has its own set of potential tasks
- **Predictive**
  - Classification
  - Regression
  - Detection
- **Descriptive**
  - Clustering
  - Association rule discovery
- We will only focus on *predictive* tasks

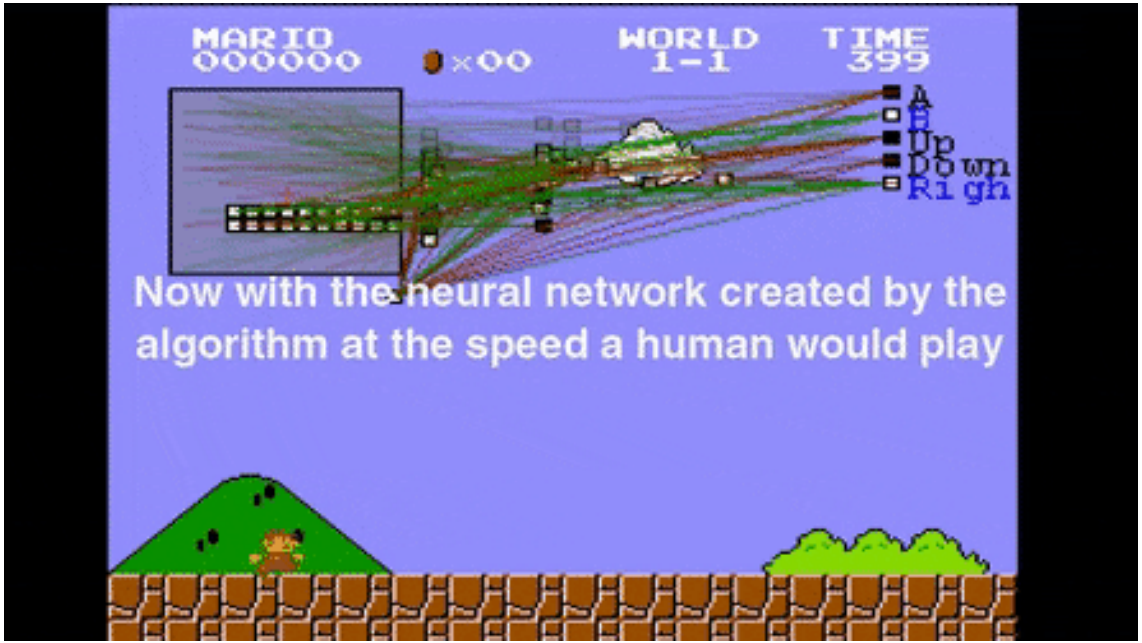
Machine learning

- These tasks can be further categorized between three distinct learning techniques

- **Supervised learning**
  - We have labels (we know the value we want to predict)
- **Unsupervised learning**
  - We only want to find underlying relationships between data, we have no distinct value to predict
- **Reinforcement learning**
  - Train a model based on a reward systems (e.g) an AI when playing your favorite shooter game will probably receive points for killing you



Machine learning



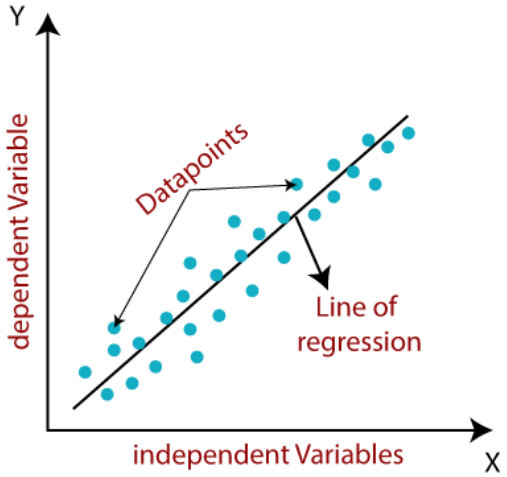
Machine learning

- Each domain has its own set of potential tasks
- **Predictive**
  - Classification – supervised
  - Regression – supervised
  - Detection – supervised/unsupervised
- **Descriptive**
  - Clustering – unsupervised
  - Association rule discovery – supervised
- We will only focus on *predictive* tasks

Regressie: recap

Regressie

- Predict a value of a given **continuous** valued variable based on the values of other variables
- Simply said, we want to identify a “relationship” between two ‘things’ (variables in our data)
  - For example: when you climb a mountain, the temperature drops
  - As a result, we could conclude that height influences temperature
  - In statistics, this is termed “regression”
  - Temperature is *dependent* variable, height is *independent* variable
- Famous in the world of statistics and neural networks



Regressie

- Applications of regression learning

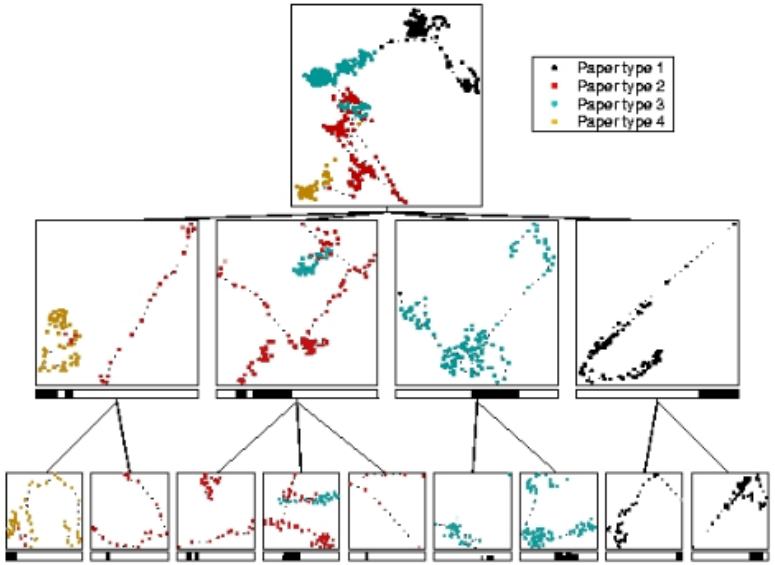
- Predict newspaper sales (better weather = more sales?)
- Temperature prediction
- Anomaly detection
  - Credit card fraud detection
  - Cyber security (network intrusion)

Height above sea level in meters	Pressure in hpa	Humidity in percentage	Temperature in Fahrenheit
0	1015	67	84
500	1000	60	73
700	850	40	65

Regressie

- Example of anomaly detection
- Monitoring paper mills

- Goal: alert operators when the paper mill start doing weird stuff
- How? Project measurements of sensors and visualize dynamics between them and check for anomalies

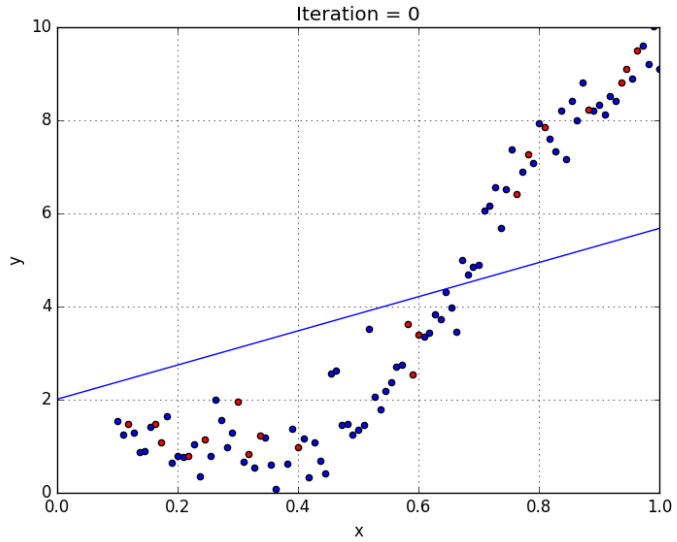


Regressie

- Popular regression learning models:

- **Linear regression**
- **Lasso**
- **Ridge**
- **ElasticNet**
- …and many more

- Above regression models are the most basic. The main difference is the way in which they penalize ‘wrong’ predictions



Classificatie

Classificatie

- Given a collection of records (a dataset), each record contains a set of **attributes** (columns), one of these attributes is called the **class**
- Machine learning models exist which learn underlying relationships between attributes and the class to predict
- Goal of classification: **previously unsees** records should be assigned a class ('Yes'/'No') as accurately as possible

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

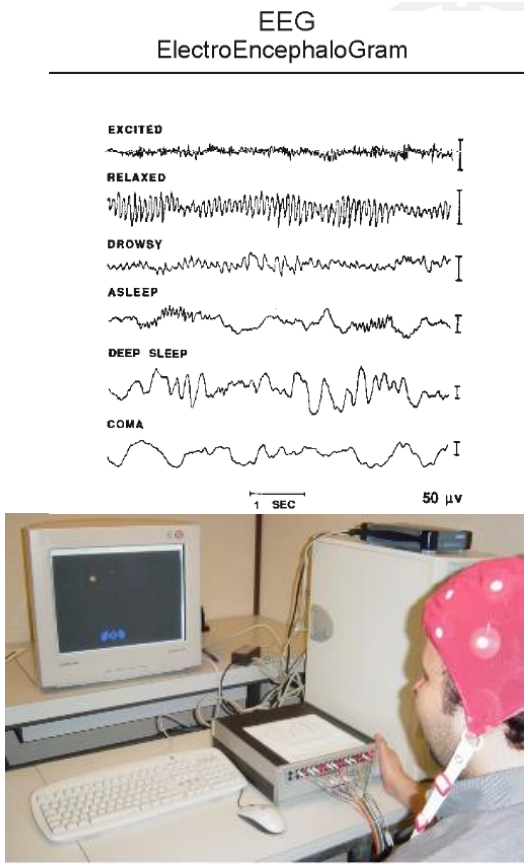
- We use a **test** set to determine the accuracy of our model.

Refund	Marital Status	Taxable Income	Cheat
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?

Classificatie

- Applications of classification learning

- Predict whether someone has breast cancer or not (‘Yes’/’No’)
- Predict fraudulent cases in credit card transactions
- Predict whether a star belongs to a certain group of stars
- Read a person’s mind
  - Measure EEG signals and classify them based on known behaviour



Classificatie

- Applications of classification learning
- Predict how likely a user might prefer an item on Amazon
- Classify handwritten digits and letters

Welke letters las u zonet? De MRI-scanner weet het

Een team in Nijmegen is er voor het eerst in geslaagd om bij iemand die een woord ziet, te achterhalen welke letters hij heeft gelezen, gegeven welke stukjes hersenschors er oplichten. De crux zit hem in een wiskundig model.

Van onze verslaggever
Bard van de Weijer

AMSTERDAM Derek Ogilvie zal zijn vingers erbij aflikken: onderzoekers van de Radboud Universiteit hebben een methode ontwikkeld waarmee uit iemands hersenactiviteit afgeleid kan worden welke letters hij ziet.

Het aflezen gebeurt met behulp van een MRI-scanner die kijkt naar de visuele cortex, het hersengebied waar beeldinformatie wordt verwerkt. Daarbij worden kubusjes brein van 2×2×2 millimeter in de visuele cortex geanalyseerd. Deze kubusjes, zogenoemde voxels, lichten op als ze worden geactiveerd door visuele informatie.

Als een proefpersoon de letter G ziet, lichten andere voxels op dan bij de letter T. De MRI-scanner meet dus voor elke letter een ander activatiepatroon. Een algoritme kan uit deze patronen de letters reconstrueren die de proefpersoon in de scanner ziet. Het gaat om handgeschreven letters, in allerlei variaties, die alle door het systeem worden herkend.

'Het is geen gedachten lezen', zegt cognitief neurowetenschapper Marcel van Gerven van het Donders Instituut van de Radboud Universiteit. 'We reconstrueren perceptie, dus wat iemand ziet, niet wat hij denkt.' Een belangrijk verschil, omdat gedachten over 'alles' kunnen gaan en het analyseren van visuele informatie – letters in dit geval – het aantal mogelijkheden beperkt. Het algoritme is getraind op het herkennen van letters. Als een proefpersoon een afbeelding van een vliegtuig wordt voorgehouden, zal dat niet herkend worden.

Een selectie van de oorspronkelijke handgeschreven letters ...

... wat de MRI-scanner ziet oplichten in de visuele cortex ...

... en de reconstructie van de letters door het algoritme.

Illustraties Radboud Universiteit

We vermoeden dat het brein ook op deze manier werkt

Tot zover is er volgens Van Gerven nog niet veel nieuws onder de zon. 'We zijn niet de eersten die met MRI-scans beeldpatronen in de visuele cortex kunnen herkennen. Het is wel voor het eerst gelukt om met een wiskundig model het oorspronkelijke beeld met hoge kwaliteit te reconstrueren.'

Dit gebeurt door twee bronnen te combineren: de onderzoekers kijken in een gebiedje van duizend voxels hoe deze reageren op externe stimuli. Deze gegevens – de wat gruizige afbeeldingen hierboven – worden gecombineerd met voorkennis over de eigenschappen van letters. Door de data van de MRI-scan te vergelijken met deze 'kennis' kan worden herleid welke letters de proefpersoon waarneemt.

'We vermoeden dat het brein ook op deze manier werkt', zegt Van Gerven. 'Je kunt al die lijntjes en hoekjes niet begrijpen voor je hebt leren lezen. Pas als sprake is van een zekere context kun je letters onderscheiden.' De onderzoekers hopen met hun onderzoek meer te weten te komen over de werking van het brein. Hoewel het bedenken van praktische toepassingen niet het eerste doel is, ziet de onderzoeker wel mogelijkheden. 'Er is een relatie tussen perceptie en verbeelding. Je zou wellicht een reconstructie kunnen maken van een beeld dat iemand zich in gedachten voorstelt. Denk aan een getuige die zich de verdachte inbeeldt en dat je dat beeld dan kunt visualiseren. Maar dat is echt de verre toekomst.'

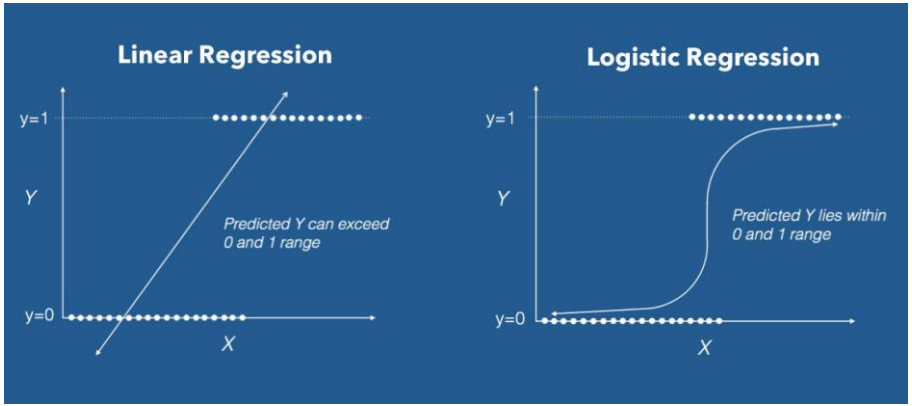
Classificatie

- Popular classification learning models:

- **Logistic regression**
- **K-Nearest Neighbours**
- **Decision Trees**
- **Random Forest**
- …and many more

Logistic regression

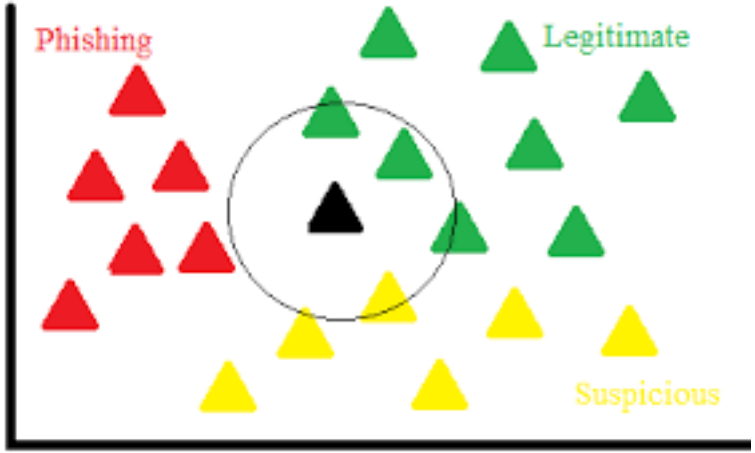
- The name is confusing, but it makes sense
- In stead of wanting to plot a straight line through our data, we want to set a *threshold* after which the algorithm prefers to predict one class above the other
- Most simple classification algorithm



K-Nearest Neighbours

- Train a KNN classifier on your training data
- When confronted with a new point, compare the new point to existing points and check which point is closer to our current point
- If *Nearest Neighbours* = 1, our new point will be assigned to the class of the nearest neighbour
- If *Nearest Neighbours = 3,* our new point will be assigned to the class to which the majority of our 3 nearest neighbours is assigned to

Question. To which class will our new (black) point be assigned when *Nearest Neighbours = 2 ?*

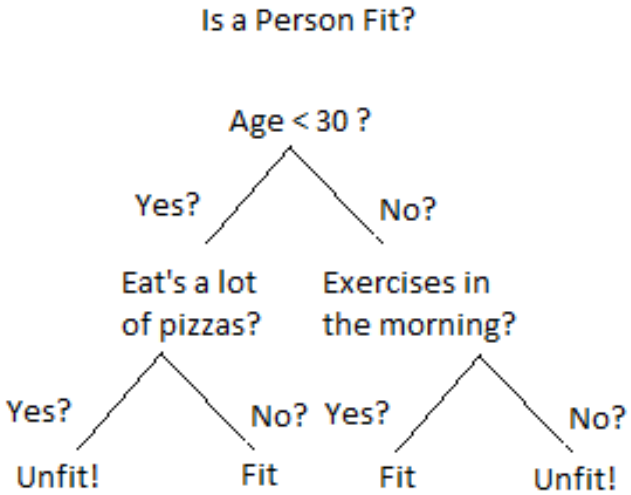


Decision trees

- Most popular classification algorithm
- Basically a huge (upside-down) tree consisting of 'if' statements

- Example if person is fit or not
- Check records in your dataset and match those against the (learned) branches of our tree
- Keep going down the tree until you read the end

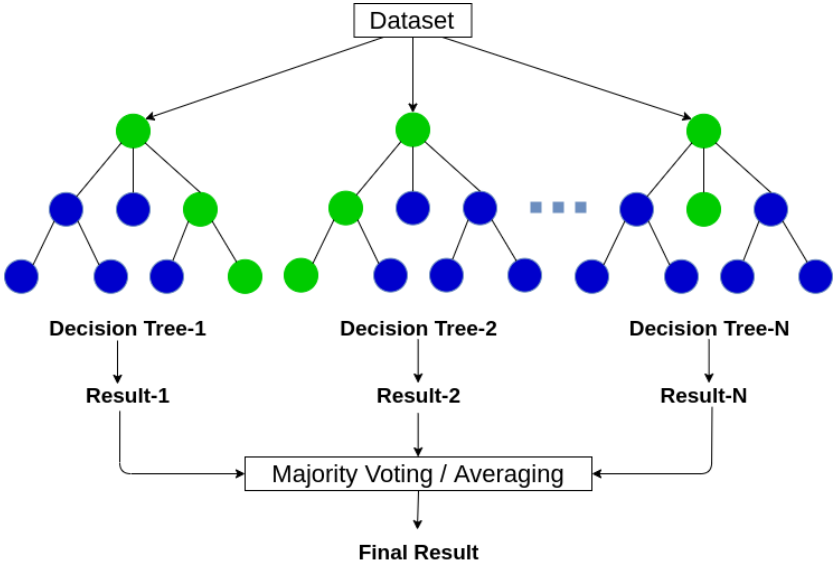
Question. If I'm 24 years old and I eat a lot of pizza's, will this model classify me as being fit?



Random forest

- Simply a collection of decision trees
- Each tree is trained separately
- Results of all the trees will be compared and majority voting will decide which decision is the (supposed) right one

- Random forest in practice performs better than decision trees. However, the runtime increases drastically!



Regressie vs. classificatie

- Note that models like K-Nearest Neighbours, Decision Trees and Random Forest can all be used for regression problems as well

Classification

Predict a discrete class label

Evaluate based on accuracy, f1, recall, etc.

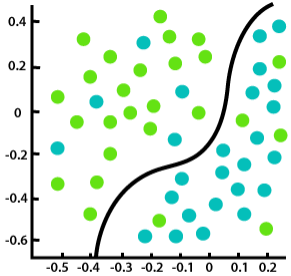
Try to find a *decision boundary* to divide classes

Regression

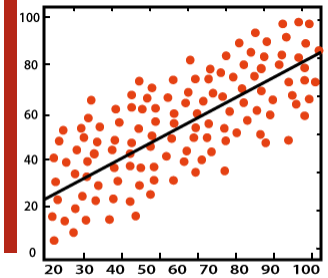
Predict a continuous quantity

Evaluate on (root) mean squared error, etc.

Try to fit the best line



Classification



Regression

Pauze

Sklearn

Sklearn

- Also known as Scikit-learn
- Open-source machine learning library for Python
- Features classification, regression and clustering algorithms and many more
- Used to be part of the SciPy toolkit (huge Python science library)
- Used NumPy for mathematical operations
- Integrates well with MatplotLib and Pandas
- Current version: 0.24.2

Sklearn

- Sklearn is huge
- Documentation: https://scikit-learn.org/stable/
- We will cover some of the basics and more important parts of the library to get you started with your project
- Experiment for yourself!

Sklearn

- Digits dataset
- Can be loaded directly using sklearn
- The `shape` attribute tells us that the dataset consists of 1797 digits with each 64 features
- If we print the `target` of the digits we can see, for each digit in our data, which 'real' digit it represents
- So: the first row of our data represents the digit '0', the second row the digit '1', etc.
- ..but, what are the values in the `digits.data` array exactly?

```
from sklearn import datasets

digits = datasets.load_digits()

print(digits.data)
print("---")
print(digits.data.shape)
```

```
[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0. 10. ... 12.  1.  0.]]
---
(1797, 64)
```

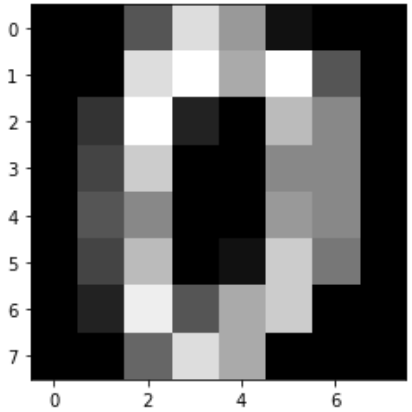
```
print(digits.target)
```

```
[0 1 2 ... 8 9 8]
```

Sklearn

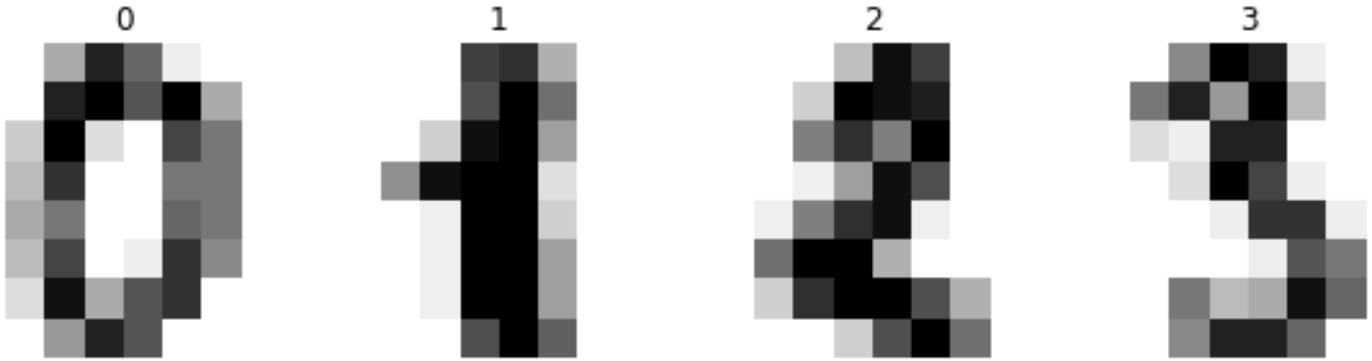
- In the previous slide, we have seen that each digit has 64 features
- If we print the content of a single digit, we can actually see that the 64 features are represented in an 8x8 grid
- This grid represents the **image** of the digit
- If you snap a colored picture with your phone, each pixel has a certain value which represents the red, green and blue **intensity** of the pixel
- Value 0 is always black, higher values are either red, green, blue or white depending on your settings
- So: each value in our 8x8 array represents the **intensity of a pixel** which, in turn, represents the digit

```
[[ 0.  0.  5. 13.  9.  1.  0.  0.]
 [ 0.  0. 13. 15. 10. 15.  5.  0.]
 [ 0.  3. 15.  2.  0. 11.  8.  0.]
 [ 0.  4. 12.  0.  0.  8.  8.  0.]
 [ 0.  5.  8.  0.  0.  9.  8.  0.]
 [ 0.  4. 11.  0.  1. 12.  7.  0.]
 [ 0.  2. 14.  5. 10. 12.  0.  0.]
 [ 0.  0.  6. 13. 10.  0.  0.  0.]]
```



Sklearn

- We can continue this technique for all of our digits



Question. Would you consider predicting digits from images a classification or regression problem and why?

Cijfers voorspellen

- **Always** make sure you clean your data before attempting to train any model!

- For supervised learning tasks, you always need 2 variables
- **X**
  - The data with the features you want the machine learning model to learn
- **y**
  - The class/continuous variable to predict

- In the digits dataset, these two are already separated into `digits.data` and `digits.target`. However, for most datasets, you must extract the target from the dataset yourself.

```
X = digits.data
y = digits.target

print(X.shape)
print(y.shape)
```

```
(1797, 64)
(1797,)
```

Cijfers voorspellen

- One Sklearn function which is critical for training is `train_test_split()`.
- This function splits your **X** (data) and **y** (labels/classes) into training and testing sets to train and verify the performance of your model
- The function returns 4 variables: X train set, X test set, y train set and y test set

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y)

print("Data train shape: " + str(X_train.shape))
print("Data test shape: " + str(X_test.shape))
print("Class train shape: " + str(y_train.shape))
print("Class test shape: " + str(y_test.shape))
```

```
Data train shape: (1257, 64)
Data test shape: (540, 64)
Class train shape: (1257,)
Class test shape: (540,)
```

Cijfers voorspellen

- The following parameters are important
  - `test_size`
    - How many % of your data is used for testing (recommend 70% training and 30% testing)
  - `random_state`
    - How much ‘randomness’ is used to select test and train indices (recommended since it removes possible patterns)
  - `stratify`
    - When given the target array **y**, Sklearn makes sure to balance the classes as much as possible
    - Why is this important?
    - If, for example, you only have the digits 0 and 1 in your train set and the rest of the digits in your test set, your model will only know how to predict 0 and 1

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y)

print("Data train shape: " + str(X_train.shape))
print("Data test shape: " + str(X_test.shape))
print("Class train shape: " + str(y_train.shape))
print("Class test shape: " + str(y_test.shape))
```

```
Data train shape: (1257, 64)
Data test shape: (540, 64)
Class train shape: (1257,)
Class test shape: (540,)
```

Cijfers voorspellen

- Now that we have our data, we can train our model and predict previously unsees digits
- For this example, we use a K-Nearest Neighbors model

- Initiate the model using the Sklearn function (don't forget to import) and call it using `KNeighboursClassifier()`
- We want to predict the class of our digit based on the 3 nearest neighbors of the new digit (`n_neighbors=3`)
- Once we have the specifications of our model set, we call `clf.fit()` to *fit* our train data (X and y train) to our model. This tells our model that we are done configuring and that we are ready to train

```
from sklearn.neighbors import KNeighborsClassifier

clf = KNeighborsClassifier(n_neighbors=3)

clf.fit(X_train, y_train)
```

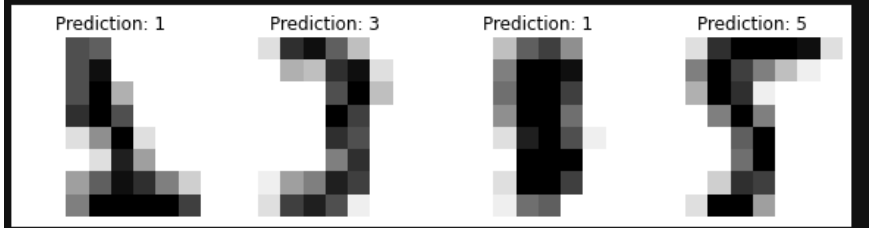
```
KNeighborsClassifier(n_neighbors=3)
```

Cijfers voorspellen

- Once we fitted the model, we can call the `.predict()` function and input our X test data to make predictions to unseen digits
- The plot below shows the true digit in the image and the predicted value of the model above it
- Our KNN classifier has learnt the relationship between the values in the array of our image to check to which class a certain digit belongs

```
preds = clf.predict(X_test)
```

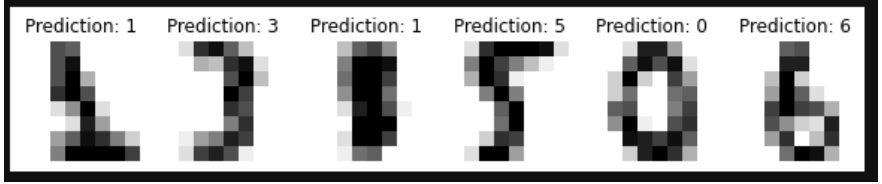
```
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, preds in zip(axes, X_test, preds):
    ax.set_axis_off()
    image = image.reshape(8, 8)
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title(f'Prediction: {preds}')
```



Cijfers voorspellen

- We can repeat the process for a Decision Tree classifier to see the same results

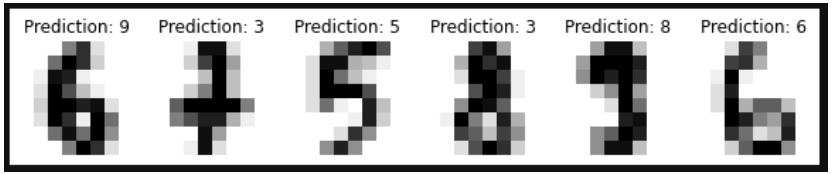
```python
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()

clf.fit(X_train, y_train)

preds = clf.predict(X_test)

_, axes = plt.subplots(nrows=1, ncols=6, figsize=(10, 3))
for ax, image, preds in zip(axes, X_test, preds):
    ax.set_axis_off()
    image = image.reshape(8, 8)
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title(f'Prediction: {preds}')
```



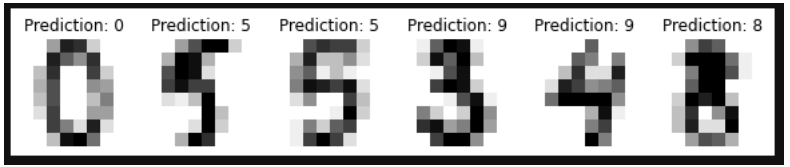
Belang van juiste verdeling

- Using the right amount of data for the test and train split is critical
- Too few test images and you model cannot accurately correct its mistakes
- Too much test images and your model is not able to fully learn relationships between features

```
Data train shape: (17, 64)
Data test shape: (1780, 64)
Class train shape: (17,)
Class test shape: (1780,)
```




```
Data train shape: (1779, 64)
Data test shape: (18, 64)
Class train shape: (1779,)
Class test shape: (18,)
```



Model validatie

Model validatie

- A crucial task in machine learning is evaluating the performance of your model
- For classification this can often be expressed in accuracy, recall or precision
- For regression this is often (root) mean squared error or mean absolute error

- Each metric can easily be imported and used using the `sklearn.metrics` library

Model validatie

- A good benchmark for your model is the `classification_report()` function
- This built-in Sklearn function shows you all of your important performance metrics like precision, recall, f1 score and support

- Accuracy: how well can my model predict a previously unsees class?
- Recall: What fraction of the relevant digit classes are retrieved in training?
- F1: A combined metric between precision and recall
- Support: Number of samples of the true response that lies in that class

```
from sklearn import metrics

print(f"Classification report for classifier {clf}:\n"
      f"{metrics.classification_report(y_test, preds)}\n")
```

```
Classification report for classifier KNeighborsClassifier(n_neighbors=3):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        53
           1       0.98      1.00      0.99        50
           2       1.00      1.00      1.00        47
           3       0.98      1.00      0.99        54
           4       0.98      1.00      0.99        60
           5       0.99      1.00      0.99        66
           6       1.00      1.00      1.00        53
           7       1.00      0.98      0.99        55
           8       0.98      0.98      0.98        43
           9       0.98      0.93      0.96        59

    accuracy                           0.99       540
   macro avg       0.99      0.99      0.99       540
weighted avg       0.99      0.99      0.99       540
```

Model validatie

- **Always** use more than one performance metric to assess the performance of your model.
- Why? You can have 100% accuracy, but if your recall (nr. of relevant class digits) is low, what does that mean for your performance?
- Example: you have 100 digits of 0 and 1 digit of 1
- As a result, your model will have an accuracy of 99%, good right?
- However, your model predicted only 0 and never 1, which means that, although your accuracy is high, your dataset is unbalanced so your support and recall will be low

```
from sklearn import metrics

print(f"Classification report for classifier {clf}:\n"
      f"{metrics.classification_report(y_test, preds)}\n")
```

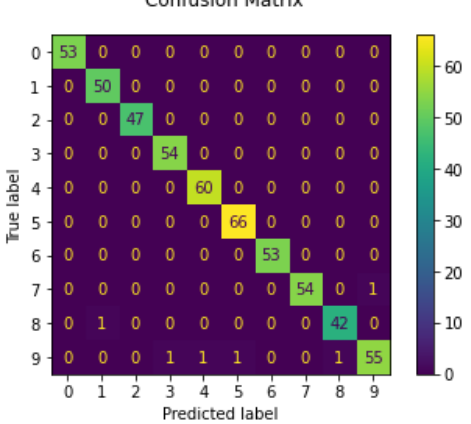
```
Classification report for classifier KNeighborsClassifier(n_neighbors=3):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        53
           1       0.98      1.00      0.99        50
           2       1.00      1.00      1.00        47
           3       0.98      1.00      0.99        54
           4       0.98      1.00      0.99        60
           5       0.99      1.00      0.99        66
           6       1.00      1.00      1.00        53
           7       1.00      0.98      0.99        55
           8       0.98      0.98      0.98        43
           9       0.98      0.93      0.96        59

    accuracy                           0.99       540
   macro avg       0.99      0.99      0.99       540
weighted avg       0.99      0.99      0.99       540
```

Confusion matrix

- A great way to see if the classes in your dataset are balanced is by plotting a **confusion matrix** after training
- Since we use classes, this matrix can only be used on classification problems
- Also part of the Sklearn metrics library

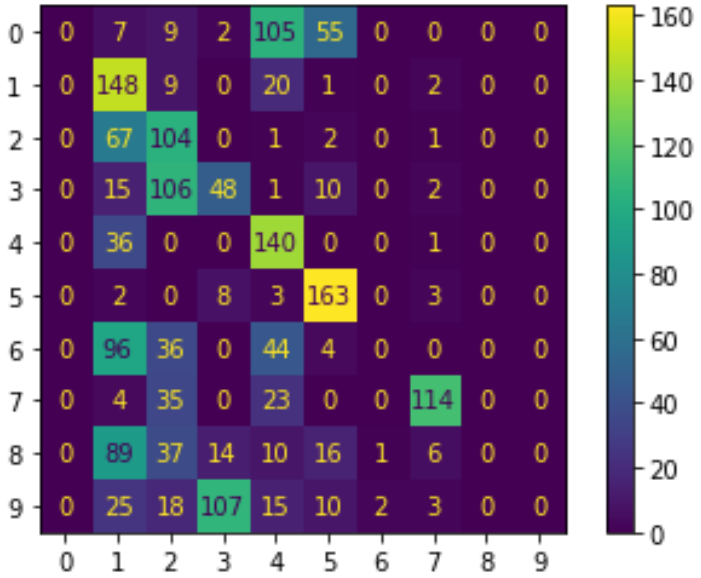
```
from sklearn import metrics

disp = metrics.plot_confusion_matrix(clf, X_test, y_test)
disp.figure_.suptitle("Confusion Matrix")

plt.show()
```

- The confusion matrix shows you the **spread** and **intensity** of predicted values and if those values have been predicted correct or not
- The x-axis shows the label that was predicted by your model while the y-axis shows the 'true' label
- Your goal is to have all (or at least most of) the values on the **diagonal line**



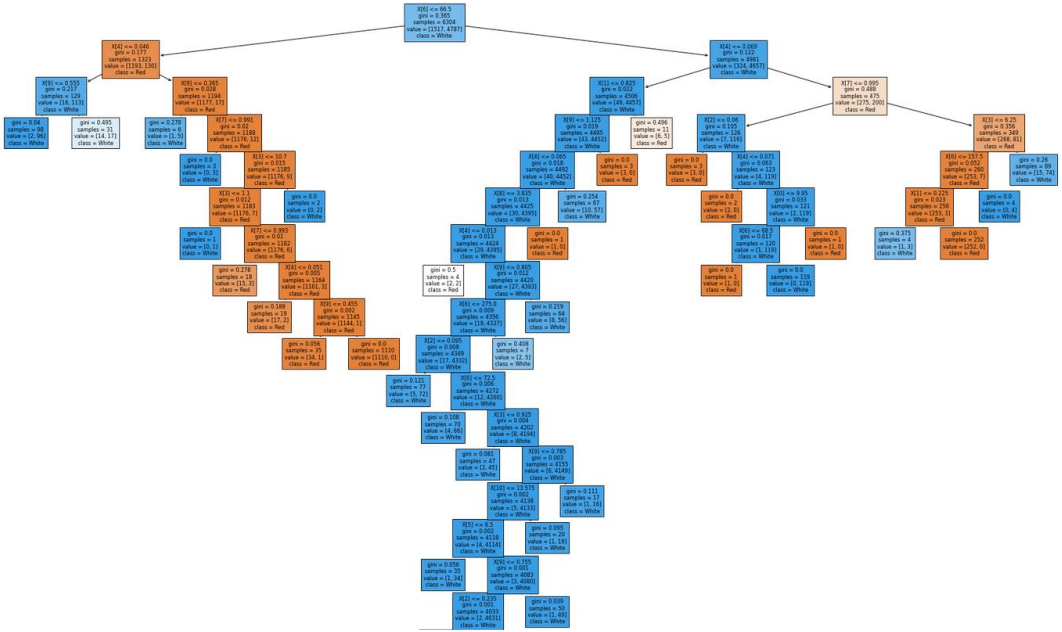
Confusion matrix

- Class inbalance is a big problem since it can mess with your predictions
- If you have a class inbalance problem, you can easily spot it in your confusion matrix
- Plotting a confusion matrix can also greatly help with determining the actual performance of your model
- For example, the given accuracy of the model on the right is 30%, we conclude that this is correct because the predicted values are all over the place



Geavanceerde visualisaties

Visualisatie van een Decision Tree



Decision Tree Classifier performance plot

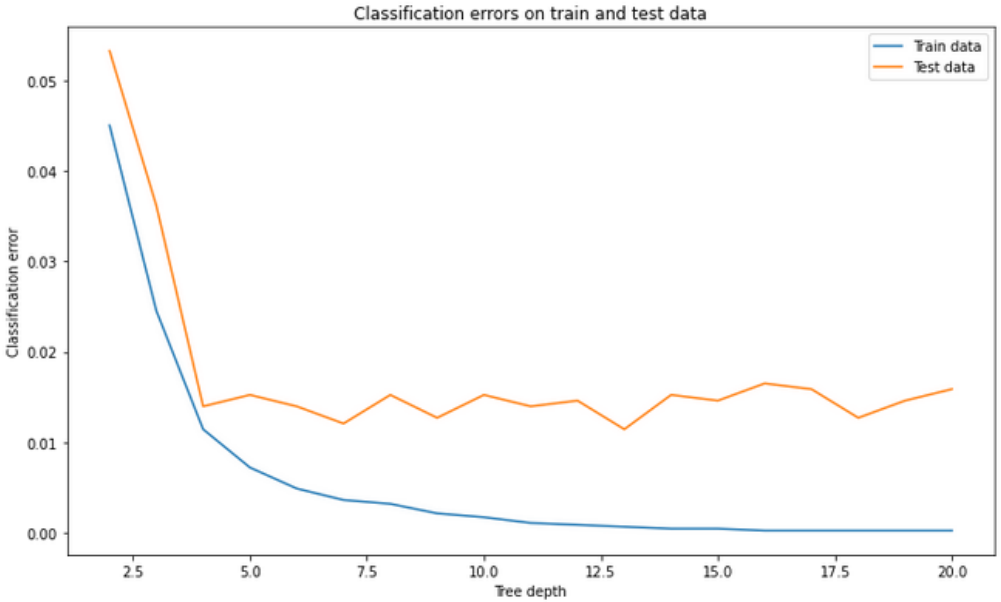


Figure 1: Simple plot showing the classification error of predictions on stratified training and testing set on a Decision Tree using multiple tree depths

Hands-on Sklearn

Hands-on Sklearnss

1) Clone the repository: https://github.com/PeaceDucko/zero-to-mastery-ml.git onto your local pc
2) Try to complete the Sklearn exercises under the 'section 2' folder

- The 'slides' folder contains some additional (more visual) theory about Sklearn

Vragen?

hogeschool Windesheim

Radboud University