

Odissee
DE CO-HOGESCHOOL

Data Science – week 7



Jens Baetens

How to participate?



 [Copy participation link](#)



1

Go to
wooclap.com

2

Enter the
event code in
the top
banner

Event code

QURRLI



1

Send **@QURRLI** to **0460 200 711**

2

You can participate

Go to **wooclap.com** and use the code **QURRLI**



Een supervised learning model maakt gebruik van



1

Enkel van bestaande data

0%

0



2

De bestaande data en een ground truth

100%

10


3

Enkel van het resultaat van een actie

0%

0



 You cannot vote anymore



Welke vraag wordt opgelost door welke soort technieken



Hoeveel?

1 ✓ 91% E

Regression

Welk type?

2 ✓ 91% D

Classification

Is dit gelijkaardig aan iets anders?

3 ✓ 82% B

Clustering

Is dit resultaat vreemd?

4 ✓ 82% A

Anomaly detection

Wat zijn de opties?

5 ✓ 91% C

Recommendation



Go to **wooclap.com** and use the code **QURRLI**



Wat is het verschil tussen parameters en hyperparameters?



1

Parameters kies je zelf, hyperparameters worden aangepast tijdens het trainen

0%

0



2

Hyperparameters kies je zelf, parameters worden aangepast tijdens het trainen

69%

9



3

Hyperparameters beïnvloeden de evaluatie van een model, parameters het train-proces

8%

1

4

Parameters beïnvloeden de evaluatie van een model, hyperparameters het train-proces

23%

3



Go to **wooclap.com** and use the code **QURRLI**



Waarom is er een verschillende aanpak van regressie en classificatie?



1

De technieken van regressie kunnen niet gebruikt worden voor classificatie

73%

8



2

De loss-functie van regressie en classificatie is verschillend

9%

1



3

Classificatie gebruikt geen labels

0%

0

4

Regressie gebruikt geen labels

18%

2



🔒 You cannot vote anymore



In welk geval kies je de volgende metrieken om een classificatiemodel te evalueren



Accuraatheid

1 ✓ 45% D

Algemeen beeld met gebalanceerde klassen

Recall

2 ✓ 73% B

Vermijden van false-negatives

Precisie

3 ✓ 64% C

Vermijden van false-positives

F1-score

4 ✓ 64% A

Algemeen beeld met ongebalanceerde klassen



Go to **wooclap.com** and use the code **QURRLI**



Wat betekent gradient descent?



minimum
vinden door
kleine
stappen te

Gebruikt om
richtingscoëf
ficiënt van
de functie te

?

Laagste
waarde
zoeken

Je zoekt het
resultaat
stap voor
stap

zoeken naar
het punt
waar de
datapunten

weet het
niet

Ben
vergeten

IDK

Algoritme

stapsgewijs

Een kleur





Feature engineering



Wat is feature engineering?

- ▣ Proces om data om te zetten naar bruikbare features
 - ▬ Vervolgstep van EDA
- ▣ Doel:
 - ▬ Duidelijkere verbanden
 - ▬ Gemakkelijker te interpreteren
 - ▬ Verbetert performantie van ML-modellen



Normalisation en scaling

- ▣ Belangrijk omdat

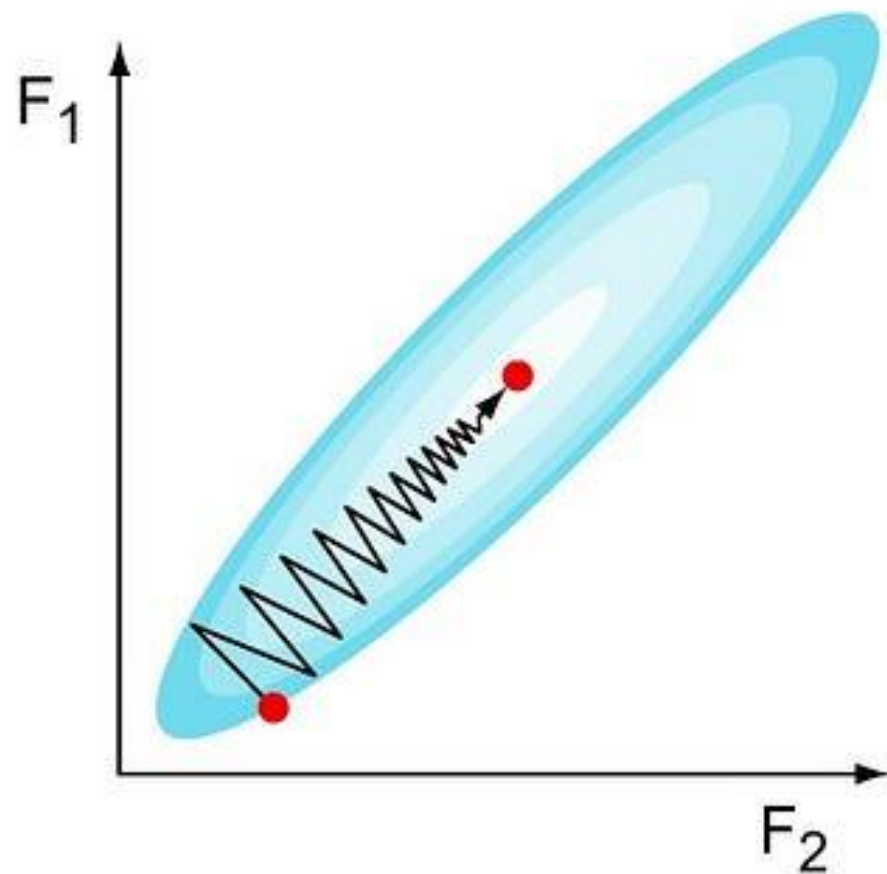
- Loss-functie gebruikt afstanden tussen features en target
- Hierdoor kunnen features andere domineren als ze grotere schaal hebben

- ▣ Opties:

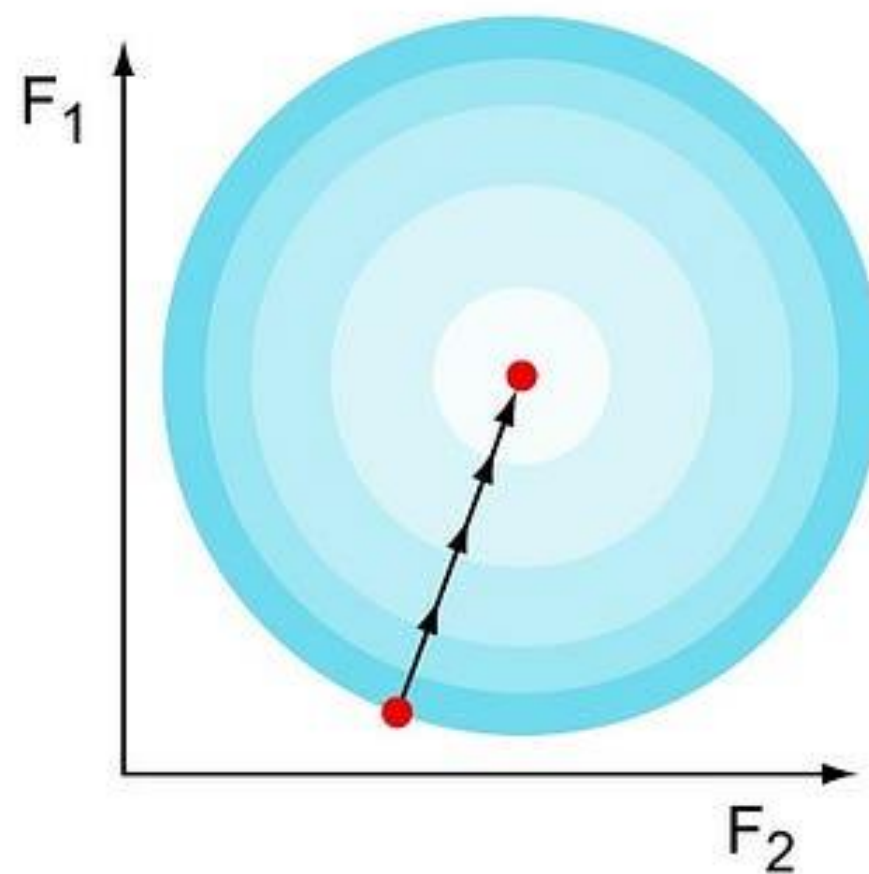
- Normalisatie: gemiddelde 0, std 1
- Min-Max scaling: range van 0 tot 1
- Eigen keuze

Gradient descent with and without feature scaling

Non-normalized features

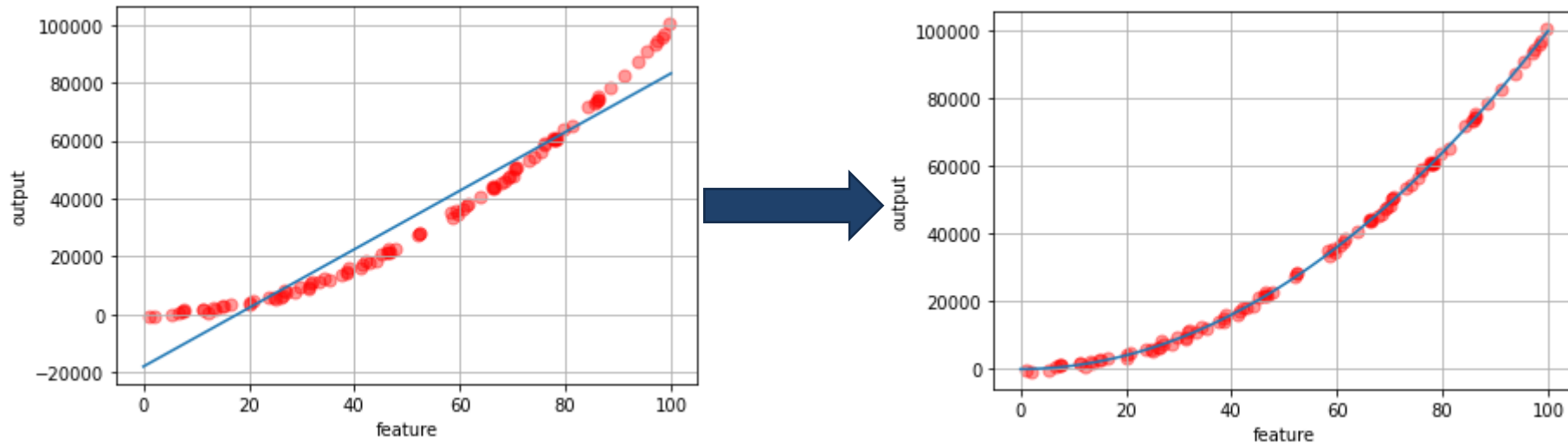


Normalized features



Higher order features

- Enkel scheidingsrechten bij lineaire/logistische regressie
 - Complexere verbanden vereisen hogere orde features
 - Machten en combinaties van verschillende kolommen

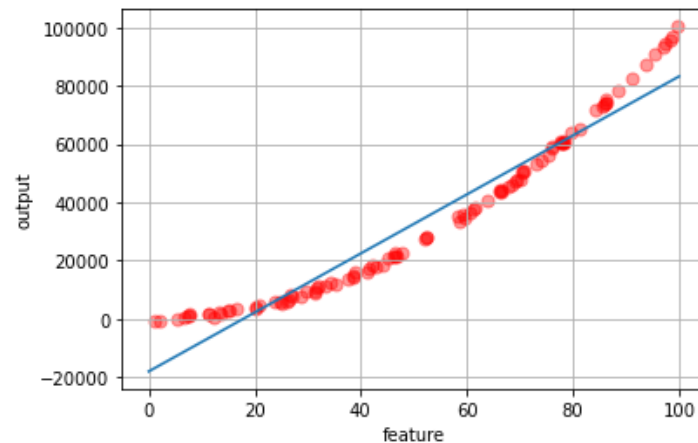




Underfitting vs overfitting

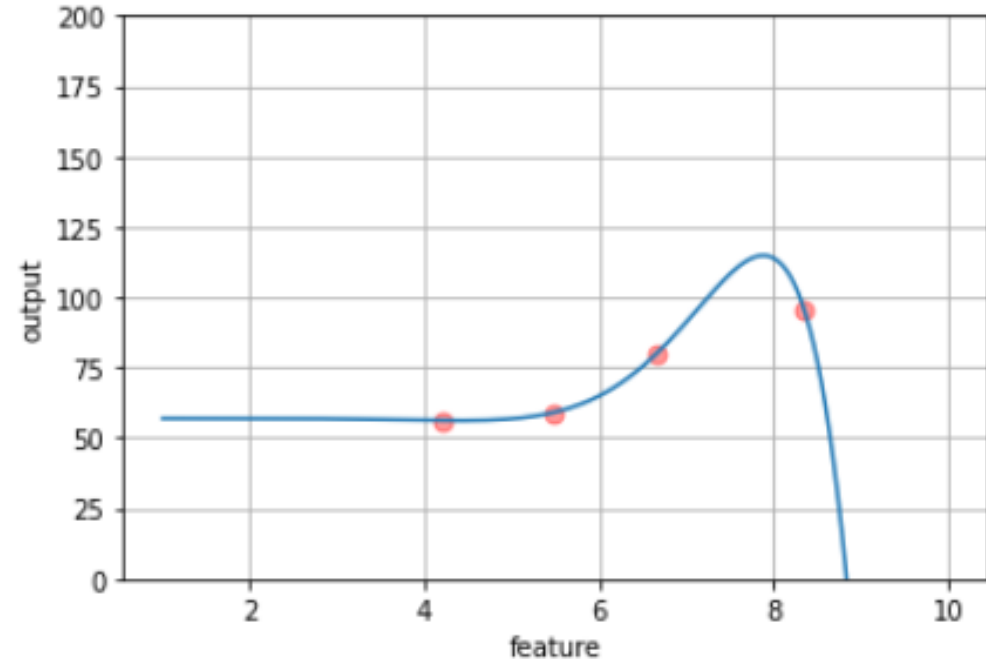
Underfitting

- Fit = hoe goed het model aangepast is aan de trainingsdata
 - Underfitting = niet goed
- Model is te eenvoudig om de trainingsdata te capteren
 - Probeer het op te lossen met feature engineering, andere hyperparameters, ...



Overfitting

- ▣ Model is te goed om de trainingsdata te capteren
 - Hierdoor presteert het niet goed op nieuwe/ongeziene data
 - Generaliseert onvoldoende
- ▣ Op te lossen door:
 - Meer data
 - Regularisatie





Hoe zie je het verschil?

- ▣ Underfitting:
 - ▬ Slechte evaluatie voor de trainingsdata
- ▣ Overfitting:
 - ▬ Goede evaluatie voor de trainingsdata
 - ▬ Slechte evaluatie voor de testdata
- ▣ Goed getraind model:
 - ▬ Goede evaluatie op trainings- en testdata

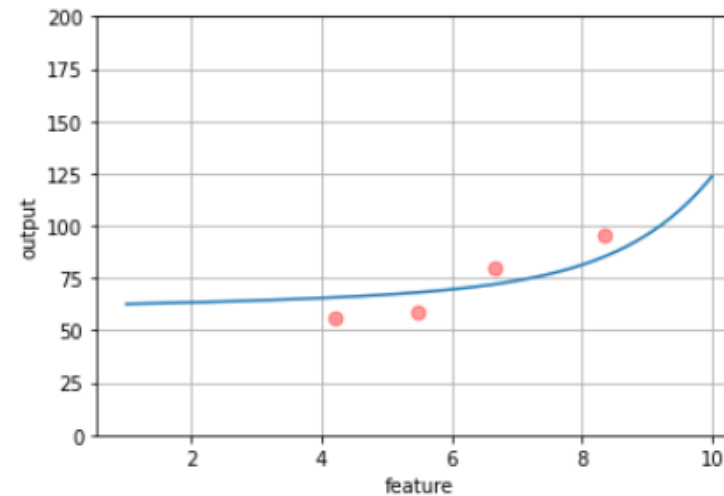
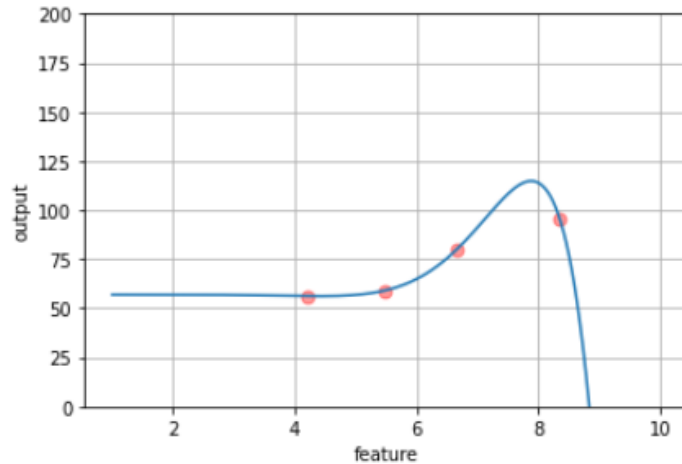


Regularisatie



Regularisatie

- ▣ Techniek om overfitting tegen te gaan
- ▣ Voeg extra kost toe aan loss-functie om te sterke verbanden af te remmen



Wat is deze kost?

$$L(\mathbf{w}) = \frac{1}{2N} \sum_{i=1}^N (f_{\mathbf{w}}(x^i) - y^i)^2 + \lambda R(\mathbf{w})$$

▣ L2-norm $\sum_{i=1}^N w_i^2$

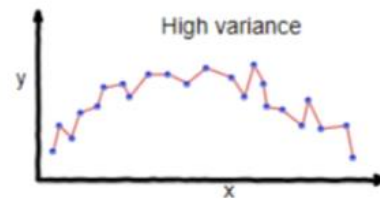
- Ridge regression bij lineaire regressie

▣ L1-norm $\sum_{i=1}^N |w_i|$

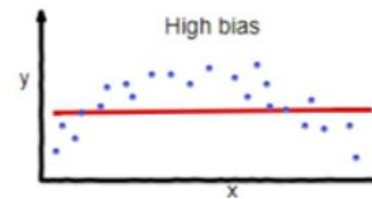
- Lasso regression

Regularisatie bij andere technieken

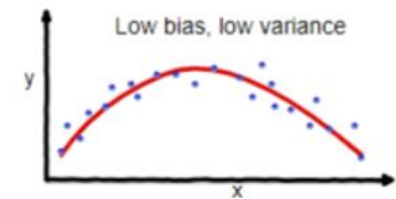
- Decision trees / Random forest
 - Max_depth, splitting parameters, ...
- SVM
 - C-parameter = L2-norm
- Bayes
 - Alpha-parameter



overfitting



underfitting



Good balance



Pipelines



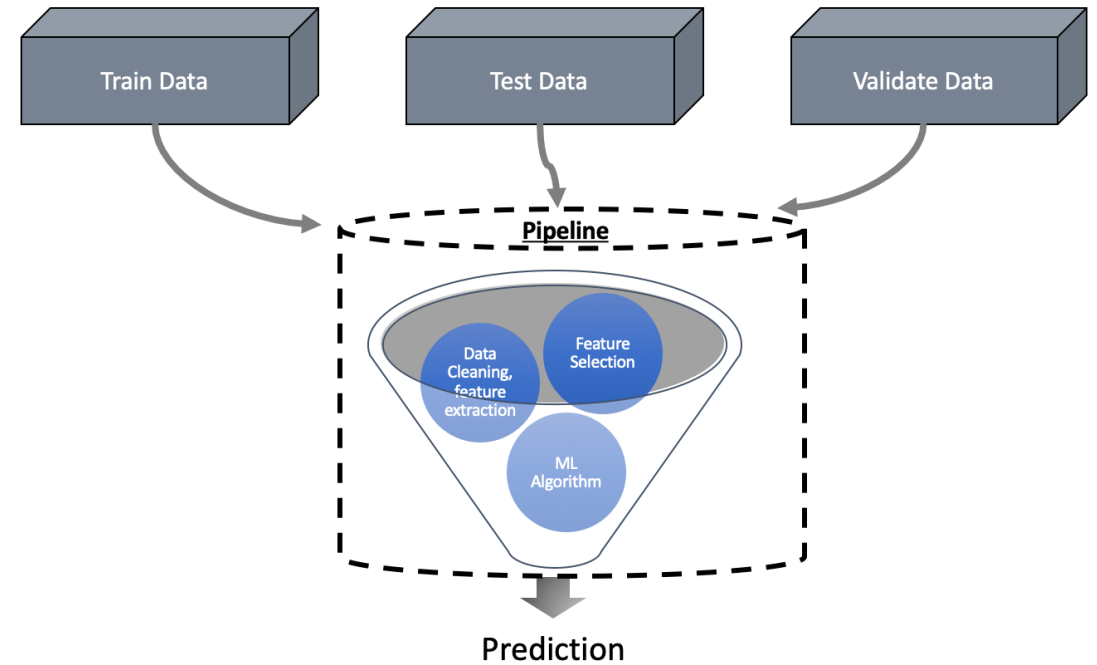
Wat is een pipeline

■ Een te volgen stappenplan

- ▬ Bevat preprocessing stappen
 - Zoals scaling, normalisatie, encoding, ...
- ▬ Bevat model-training stappen

■ Reden

- ▬ Vereenvoudigd en automatiseert het trainen en gebruiken van modellen
- ▬ Zorgt voor consistentie in preprocessing stappen
- ▬ Verbeterd reproduceerbaarheid
- ▬ Kleinere kans op human error
- ▬ Vereenvoudigd het experimenteren met verschillende parameters en modellen




```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OrdinalEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

```
# Sample data
```

```
X, y = load_data()
```

```
# Split data into training and test sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# ColumnTransformer for preprocessing
```

```
preprocessor = ColumnTransformer(
```

```
    transformers=[
```

```
        ('num', StandardScaler(), numeric_features), # Scaling numerical features
```

```
        ('ord', OrdinalEncoder(), ordinal_features), # Ordinal encoding categorical features
```

```
    ])
```

```
# Create a pipeline
```

```
pipeline = Pipeline([
```

```
    ('preprocessor', preprocessor),
```

```
    ('model', RandomForestClassifier()) # Example model, replace with your choice
```

```
])
```

```
# Fit pipeline to training data
```

```
pipeline.fit(X_train, y_train)
```

```
# Evaluate pipeline on test data
```

```
accuracy = pipeline.score(X_test, y_test)
```

Hoe elke kolom te behandelen

Scaling voor numerieke kolommen

Ordinal encoding voor categorieke

Combineer preprocessor met
ML-techniek

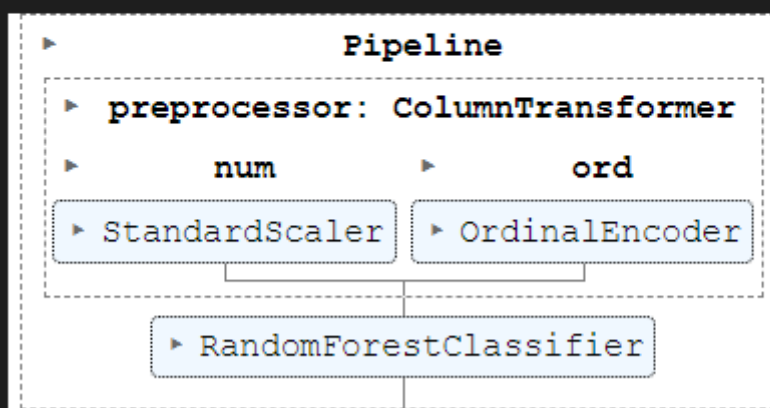
Train de ML-techniek

Evalueer het model

Visualisatie van de pipeline

```
from sklearn import set_config  
  
set_config(display="diagram")  
pipeline
```

✓ 0.0s



Beschikbare preprocessing

■ Sklearn.preprocessing

■ Custom steps

■ klasse met

- fit()
- transform()

<code>preprocessing.Binarizer(*[, threshold, copy])</code>	Binarize data (set feature values to 0 or 1) according to a threshold.
<code>preprocessing.FunctionTransformer([func, ...])</code>	Constructs a transformer from an arbitrary callable.
<code>preprocessing.KBinsDiscretizer([n_bins, ...])</code>	Bin continuous data into intervals.
<code>preprocessing.KernelCenterer()</code>	Center an arbitrary kernel matrix K .
<code>preprocessing.LabelBinarizer(*[, neg_label, ...])</code>	Binarize labels in a one-vs-all fashion.
<code>preprocessing.LabelEncoder()</code>	Encode target labels with value between 0 and n_classes-1.
<code>preprocessing.MultiLabelBinarizer(*[, ...])</code>	Transform between iterable of iterables and a multilabel format.
<code>preprocessing.MaxAbsScaler(*[, copy])</code>	Scale each feature by its maximum absolute value.
<code>preprocessing.MinMaxScaler([feature_range, ...])</code>	Transform features by scaling each feature to a given range.
<code>preprocessing.Normalizer([norm, copy])</code>	Normalize samples individually to unit norm.
<code>preprocessing.OneHotEncoder(*[, categories, ...])</code>	Encode categorical features as a one-hot numeric array.
<code>preprocessing.OrdinalEncoder(*[, ...])</code>	Encode categorical features as an integer array.
<code>preprocessing.PolynomialFeatures([degree, ...])</code>	Generate polynomial and interaction features.
<code>preprocessing.PowerTransformer([method, ...])</code>	Apply a power transform featurewise to make data more Gaussian-like.
<code>preprocessing.QuantileTransformer(*[, ...])</code>	Transform features using quantiles information.
<code>preprocessing.RobustScaler(*[, ...])</code>	Scale features using statistics that are robust to outliers.
<code>preprocessing.SplineTransformer([n_knots, ...])</code>	Generate univariate B-spline bases for features.
<code>preprocessing.StandardScaler(*[, copy, ...])</code>	Standardize features by removing the mean and scaling to unit variance.
<code>preprocessing.TargetEncoder([categories, ...])</code>	Target Encoder for regression and classification targets.
⏮	
<code>preprocessing.add_dummy_feature(X[, value])</code>	Augment dataset with an additional dummy feature.
<code>preprocessing.binarize(X, *[, threshold, copy])</code>	Boolean thresholding of array-like or scipy.sparse matrix.
<code>preprocessing.label_binarize(y, *, classes)</code>	Binarize labels in a one-vs-all fashion.
<code>preprocessing.maxabs_scale(X, *[, axis, copy])</code>	Scale each feature to the [-1, 1] range without breaking the sparsity.
<code>preprocessing.minmax_scale(X[, ...])</code>	Transform features by scaling each feature to a given range.
<code>preprocessing.normalize(X[, norm, axis, ...])</code>	Scale input vectors individually to unit norm (vector length).
<code>preprocessing.quantile_transform(X, *[, ...])</code>	Transform features using quantiles information.
<code>preprocessing.robust_scale(X, *[, axis, ...])</code>	Standardize a dataset along any axis.
<code>preprocessing.scale(X, *[, axis, with_mean, ...])</code>	Standardize a dataset along any axis.
<code>preprocessing.power_transform(X[, method, ...])</code>	Parametric, monotonic transformation to make data more Gaussian-like.
⏭	



Hyperparameter tuning





Hyperparameter tuning

- ▣ Hyperparameters zijn parameters van het model
 - ▬ Kies je bij aanmaak van het model
 - ▬ Worden niet aangepast tijdens training maar beïnvloeden het leerproces
- ▣ Hoe kies je de beste
 - ▬ Manueel
 - Heel tijdrovend
 - ▬ Automatisch via hyperparameter tuning
 - Betere resultaten/generalisatie/...



Verschillende methoden voor hyperparameter tuning

▣ Grid search


- ▬ Ga alle parameters af in een grid

▣ Random search

- ▬ Neem een aantal willekeurige combinaties van hyperparameters

▣ Bayesiaanse optimalisatie

- ▬ Bekijk de impact van hyperparameters en maak een slimme keuze
- ▬ Niet aanwezig in sklearn maar een variant ervan wel
 - HalvingGridSearch of HalvingRandomSearch



```
from sklearn.model_selection import GridSearchCV

# Define hyperparameter grid
param_grid = {
    'model__n_estimators': [50, 100, 200],
    'model__max_depth': [None, 10, 20],
    # Add more hyperparameters
}

# Create GridSearchCV
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy')

# Fit to data
grid_search.fit(X_train, y_train)

# Access best hyperparameters and model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

# Evaluate best model
best_accuracy = best_model.score(X_test, y_test)
```

Cross validation – welke data gebruiken om beste model te kiezen?

- Tot nu: data gesplitst in training en test
 - ▬ Als we testdata gebruiken voor beste hyperparameters te kiezen dan optimaliseren we het voor geziene data
 - Dit willen we niet



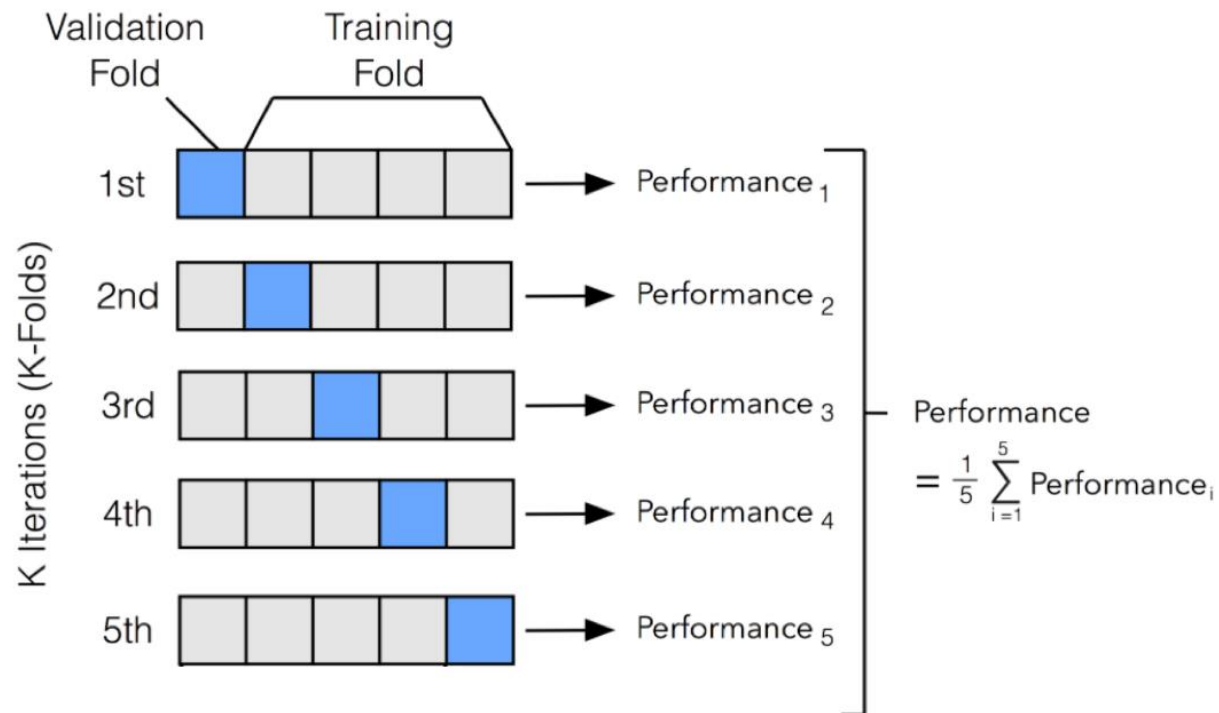
Cross validation – welke data gebruiken om beste model te kiezen?

- ▣ Oplossing: Splits trainingsdata in trainingsdata en validatiedata
 - ▬ Validatiedata om beste hyperparameters te kiezen
 - ▬ Dan nog test-data beschikbaar voor finale evaluatie



Cross validation

- Wat als je toevallig een gelukkige verdeling hebt waarvoor het goed werkt?
 - Gebruik cross-validation voor betrouwbaardere resultaten



CROSS VALIDATION

Index	Train	Test	Score	Score
1	1	2	0.8	0.7

k



Data leakage





Data leakage

- ▣ Testdata **MOET** ongezien zijn
 - ▬ D.w.z. mag niet gebruikt zijn voor training of validatie
- ▣ Let dus op voor:
 - ▬ Duplicaten
 - ▬ Data die pas op een later tijdstip beschikbaar is
 - Bvb: een test of operatie die pas uitgevoerd wordt nadat de diagnose gesteld is
 - ▬ ...



Voorbeeld van data leakage

- Competitie om walvissen te detecteren op basis van sonars
 - ▬ Model deed het heel goed op de data maar niet in de praktijk
 - ▬ <https://www.kaggle.com/c/the-icml-2013-whale-challenge-right-whale-redux/discussion/4865#25839>
 - ▬ Probleem: Bestandsgrootte en timestamps in de opnames gebruikt voor de voorspelling



Hoe data leakage minimaliseren

- ▣ Geen features die pas later ingevuld worden
- ▣ Maak gebruik van pipelines
 - Scalers/imputers werken op de correcte dataset
 - Bvb geen testdata gebruiken voor de scaling van trainingsdata
- ▣ Pas op met oversampling
- ▣ Testdata zo snel mogelijk afscheiden en apart houden!



Best practices





Best practices

- ▣ Plaats zoveel mogelijk in een pipeline
- ▣ Gebruik hyperparameter tuning
- ▣ Gebruik verschillende ML-technieken
- ▣ Let op over/underfitting en pas regularisatie toe indien nodig
- ▣ Indien de beste parameter op de rand van je bereik ligt
 - Pas het bereik waarin je hyperparameter tuning uitvoert aan



Huiswerk





Belangrijke termen

- ▣ Feature engineering
- ▣ Normalisation
- ▣ Scaling
- ▣ Overfitting
- ▣ Underfitting
- ▣ Regularisatie
- ▣ L2- vs L1 norm
- ▣ Pipelines
- ▣ Columntransformer
- ▣ Hyperparameter tuning
- ▣ Cross validation
- ▣ Data leakage
- ▣ Imputer



Data visualization tutorial

▣ Ga naar:

- <https://www.kaggle.com/learn/intermediate-machine-learning>
- Volg hoofdstuk 4-7 van de tutorial
- De informatie in de tutorials is te kennen leerstof en helpt bij het maken van de oefeningen



Data modelling: oefening

▣ Data modelling oefening

- Opgave: <https://classroom.github.com/a/FoxRPK1f>
- Op punten