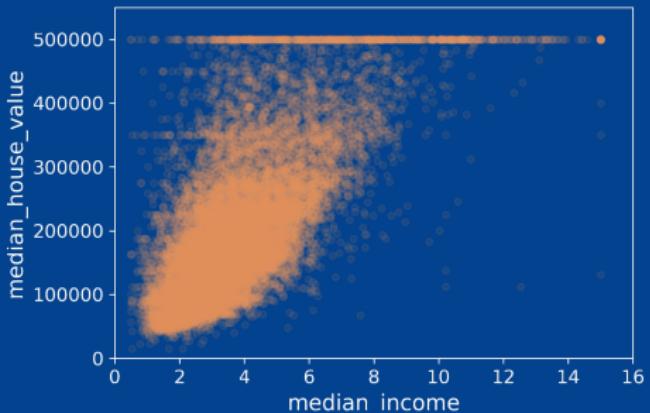




LESSON 4: Data analyse

CARSTEN EIE FRIGAARD

SPRING 2023



Agenda

Data analyse

1. Admin
 - ▶ feedback på O1.
2. Datasæt og Jeres O4 projekt.
 - ▶ Opgave: L04/[dataanalyse.ipynb](#)
3. Statestik og visualisering.
4. Pipelines
 - ▶ Opgave: L04/[pipelines.ipynb](#)

'Demo' datasæt

MNIST, Iris og Moon

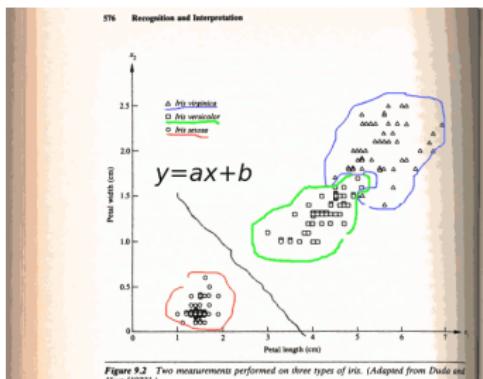
Iris:

Sepal/petal længde/bredde,

Mr. Fisher, 1936,

"Anderson's Iris data set"

```
sklearn.datasets.load_iris(..)
```



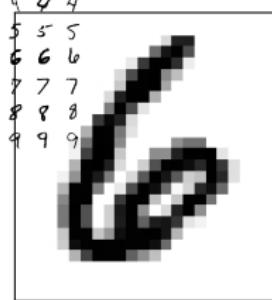
MNIST:

Håndskrevne tal,

preprocesseret, centrerede,

```
sklearn.datasets.fetch_openml('mnist_784'..
```

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

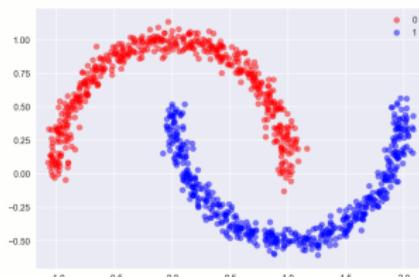


Moon:

'XOR' lign.,

non-linear decision boundary,

```
sklearn.datasets.make_moons(..)
```



Opg. L04 Beskrivelse af eget slutprojekt.pdf

Dit datasæt fra f.eks. [https://www.kaggle.com...](https://www.kaggle.com)

(brug min login: user=cef@ase.au.dk, password=test123)

The screenshot shows a web browser window with the URL [https://www.kaggle.co...](https://www.kaggle.com) in the address bar. The page content is a dataset titled "Beer Consumption - Sao Paulo". The title is displayed prominently at the top left, followed by the subtitle "Predict beer consumption". Below the title, it says "Don George · updated 3 months ago (Version 2)". The main content area features a blurred background image of a city street at night. At the bottom of the page, there are navigation links for "Data", "Overview", "Kernels (8)", "Discussion (1)", and "Activity". On the right side, there are buttons for "Download (5 KB)" and "New Kernel". A sidebar on the left contains sections for "Data Sources" (listing "Consumo_cerveja.csv" as 941x7), "About this file" (describing beer as a democratic drink), and "Columns" (listing "Data" and "Temperatura Media (C)").

We use cookies on kaggle to deliver our services, analyze web traffic, and improve your experience on the site. By using kaggle, you agree to our use of cookies.

Got it Learn more

kaggle Search Competitions Datasets Kernels Discussion Learn ... Sign in

Dataset

Beer Consumption - Sao Paulo

Predict beer consumption

Don George · updated 3 months ago (Version 2)

Data Overview Kernels (8) Discussion (1) Activity Download (5 KB) New Kernel

Data (5 KB)

Data Sources

Consumo_cerveja.csv 941x7

About this file

Beer is one of the most democratic and consumed drinks in the world. Not

Columns

Data Temperatura Media (C)

4/24

Opg. L04 Beskrivelse af eget slutprojekt.pdf

..eller UCI [https://archive.ics.uci.edu/ml/index.php..](https://archive.ics.uci.edu/ml/index.php)



[About](#) [Citation Policy](#) [Donate a Data Set](#) [Contact](#)

Repository Web [Search](#) [Google](#)

[View ALL Data Sets](#)

Machine Learning Repository

[Center for Machine Learning and Intelligent Systems](#)

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 557 data sets as a service to the machine learning community. You may [view all data sets](#) through our searchable interface. For a general overview of the Repository, please visit our [About page](#). For information about citing data sets in publications, please read our [citation policy](#). If you wish to donate a data set, please consult our [donation policy](#). For any other questions, feel free to [contact the Repository librarians](#).

Supported By:



In Collaboration With:



Latest News:

- 09-24-2018: Welcome to the new Repository admins Dheeru Dua and Efi Karra Taniskidou!
- 04-04-2013: Welcome to the new Repository admins Kevin

Newest Data Sets:

- 07-22-2020:  [Facebook Large Page-Page Network](#)
- 07-17-2020:  [Amphibians](#)
- 07-12-2020:  [Early stage diabetes risk prediction dataset.](#)

Most Popular Data Sets (hits since 2007):

- 3521507:  [Iris](#)

- 1917226:  [Adult](#)

- 4476074:  [Vowel](#)

Intro til YOLOV5..

YOLOV3/4/5 Family and Convolutional Neural Networks (CNN)

PyTorch Get Started Ecosystem Mobile Blog Tutorials Docs

YOLOV5

[View on Github](#) > [Open on Google Colab](#)



BEFORE YOU START

Start from a **Python>=3.8** environment with <https://pytorch.org/get-started/locally/>. To install YOLOv5, run:

```
pip install -qr https://raw.githubusercontent.com/ultralytics/yolov5/master/requirements.txt
```

MODEL DESCRIPTION

	Nano	Small	Medium	Large	XLarge
YOLOv5n					
4 MB _{FP16}	14 MB _{FP16}	41 MB _{FP16}	89 MB _{FP16}	166 MB _{FP16}	
6.3 ms _{COCO}	6.4 ms _{COCO}	8.2 ms _{COCO}	10.1 ms _{COCO}	12.1 ms _{COCO}	
28.4 mAP _{COCO}	37.2 mAP _{COCO}	45.2 mAP _{COCO}	48.8 mAP _{COCO}	50.7 mAP _{COCO}	



YOLOv5
你只看一次v5
by 



 Download on the App Store

YOL Better YOLOv5n YOLOv5s YOLOv5m YOLOv5l YOLOv5x Efficiency

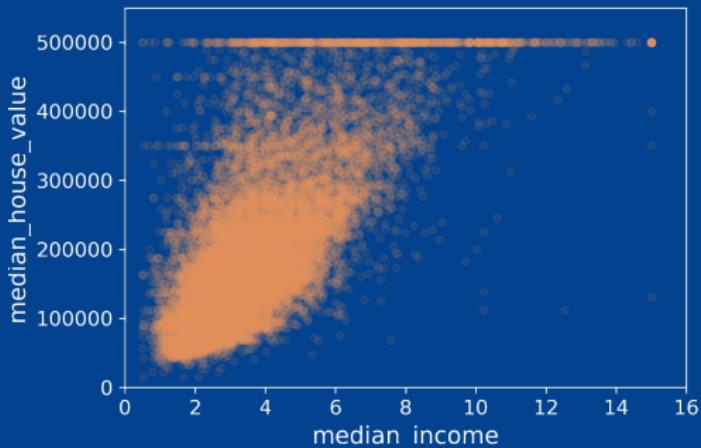
GPU Speed (ms)

NOTE: YOLOV5 [https://pytorch.org/hub/ultralytics_yolov5/]

Demo video [https://www.youtube.com/watch?v=1_SiUOYUoOI]

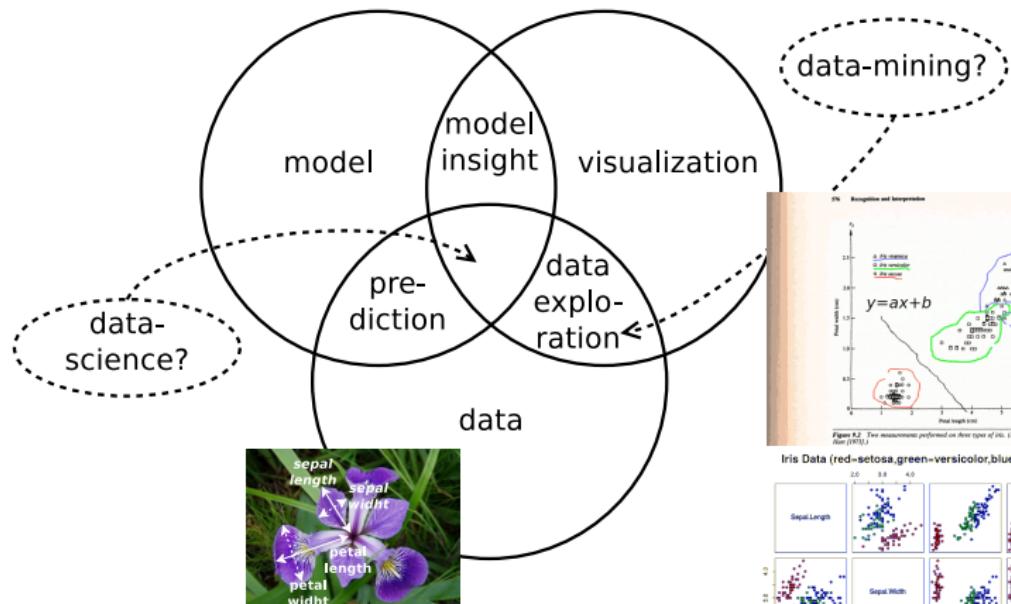
STATISTICS AND VISUALIZATION

μ, σ and being a Data Science Expert..



Machine learning baggrund

Machine learning: data science ekspert (fra L01)



Mean and Variance

The mean and variance (and hence the standard deviation) for a random variable X can for a population of N samples be estimated as

$$E[X] = \frac{1}{N} \sum_{i=1}^N X_i = \mu_X$$

$$\begin{aligned} V[X] &= E[(X_i - E[X])(X_i - E[X])] \\ &= E[X_i X_i] - E[X]^2 \\ &= \left(\frac{1}{N} \sum_{i=1}^N X_i X_i \right) - \mu_X^2 = \sigma_X^2 \end{aligned}$$

$$\sigma = \sqrt{V}$$

Notice that the $1/N$ factor most often appears as $1/(N - 1)$ for the variance estimation.

When using the factor $1/(N - 1)$, \hat{V} is said to be the best unbiased estimator, when it is $1/N$ it will be biased, both assuming an underlying normal distribution.

Auto- and Cross-covariance

Now let's go to full matrix notation for the covariance. For a data matrix \mathbf{X}

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \vdots & & & \\ x_1^{(n)} & x_2^{(n)} & \dots & x_d^{(n)} \end{bmatrix}$$

the previous one-dimensional random variable X can now be seen as a d -dimensional vector $\mathbf{x}^{(i)}$, that is one of the data rows in the full data matrix

$$X_i \rightarrow \mathbf{x}^{(i)} = \left[x_1^{(i)} \ x_2^{(i)} \ \dots \ x_d^{(i)} \right]^T$$

Auto- and Cross-covariance

The (biased) auto-covariance matrix can be estimated as

$$\begin{aligned} E[\mathbf{X}] &= \mu_{\mathbf{X}} \\ &= \frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)} \end{aligned}$$

$$\begin{aligned} \Sigma(\mathbf{X}) &= \text{cov}(\mathbf{X}, \mathbf{X}) \\ &= \frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)} \mathbf{x}^{(i)T} - \mu_{\mathbf{X}} \mu_{\mathbf{X}}^T \end{aligned}$$

with implicit $1/N$ factor here (inside the first $E[\cdot]$), so the definition is a biased covariance. You may opt to use the factor $1/(N - 1)$ instead.

For yet another data matrix \mathbf{Z} the (biased) cross-covariance matrix is given by

$$\begin{aligned} \Sigma(\mathbf{X}, \mathbf{Z}) &= \text{cov}(\mathbf{X}, \mathbf{Z}) \\ &= \frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)} \mathbf{z}^{(i)T} - \mu_{\mathbf{X}} \mu_{\mathbf{Z}}^T \end{aligned}$$

Covariance Matrix, binary classifier (SKIP slide!)

For a dataset, \mathbf{X} ; features cat and dog; classifying cat/dog

Covariance matrix (dim = features x features)

$$\Sigma(\mathbf{X}) = \begin{bmatrix} \sigma_{\text{cat,cat}} & \sigma_{\text{cat,dog}} \\ \sigma_{\text{dog,cat}} & \sigma_{\text{dog,dog}} \end{bmatrix} = \begin{array}{c|cc} & \text{cat} & \text{dog} \\ \text{cat} & \sigma_{\text{cat}}^2 & \sigma_{\text{cat,dog}} \\ \text{dog} & \sigma_{\text{dog,cat}} & \sigma_{\text{dog}}^2 \end{array}$$

co-variance

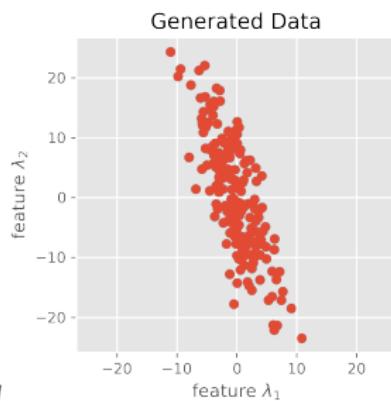
$$\sigma_{\text{cat,dog}} = \frac{1}{n} \sum_{i=1}^n (x_{\text{cat}}^{(i)} - \mu_{\text{cat}})(x_{\text{dog}}^{(i)} - \mu_{\text{dog}})$$

Example $\Sigma(\mathbf{X}) = \begin{bmatrix} 13.2 & -28.8 \\ -28.8 & 93.3 \end{bmatrix}$

Confusion matrix, cats only, cat/non-cat

(dim = classes x classes)

$$\mathbf{M} = \begin{bmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{bmatrix} = \begin{array}{c|cc} & \text{actual} \\ & \text{cat} & \text{non-cat} \\ \text{cat} & \text{T, cat} & \text{F, cat} \\ \text{dog} & \text{F, dog} & \text{T, non-cat} \end{array}$$



Looking for Correlations

..using the **auto-covariance matrix**,

	Grade 3	Grade 4	Grade 5	Grade 6	Grade 7	Grade 8
Grade 3	1.0000	0.7598	0.7199	0.6940	0.6869	0.6432
Grade 4	0.7598	1.004	0.7975	0.7675	0.7574	0.7189
Grade 5	0.7198	0.7975	0.9933	0.7813	0.7639	0.7218
Grade 6	0.6940	0.7675	0.7813	0.9899	0.7958	0.7579
Grade 7	0.6869	0.7574	0.7639	0.7958	0.9820	0.7884
Grade 8	0.6432	0.7189	0.7218	0.7579	0.7884	0.9826

..for regression: the standard correlation coefficient,
Pearson's r,

Looking for Correlations

Since the dataset is not too large, you can easily compute the standard correlation coefficient (also called Pearson's r) between every pair of attributes using the `corr()` method:

```
corr_matrix = housing.corr()
```

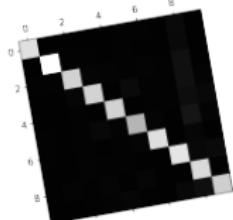
Now let's look at how much each attribute correlates with the median house value:

```
>>> corr_matrix["median_house_value"].sort_values(ascending=False)
```

median_house_value	1.000000
median_income	0.687170
total_rooms	0.135231
housing_median_age	0.114220
households	0.064702
total_bedrooms	0.047865
population	-0.026699
longitude	-0.047279
latitude	-0.142826

Name: median_house_value, dtype: float64

The correlation coefficient ranges from -1 to 1. When it is close to 1, it means that there is a strong positive correlation; for example, the median house value tends to go up when the median income goes up. When the coefficient is close to -1, it means that there is a strong negative correlation; you can see a small negative correlation

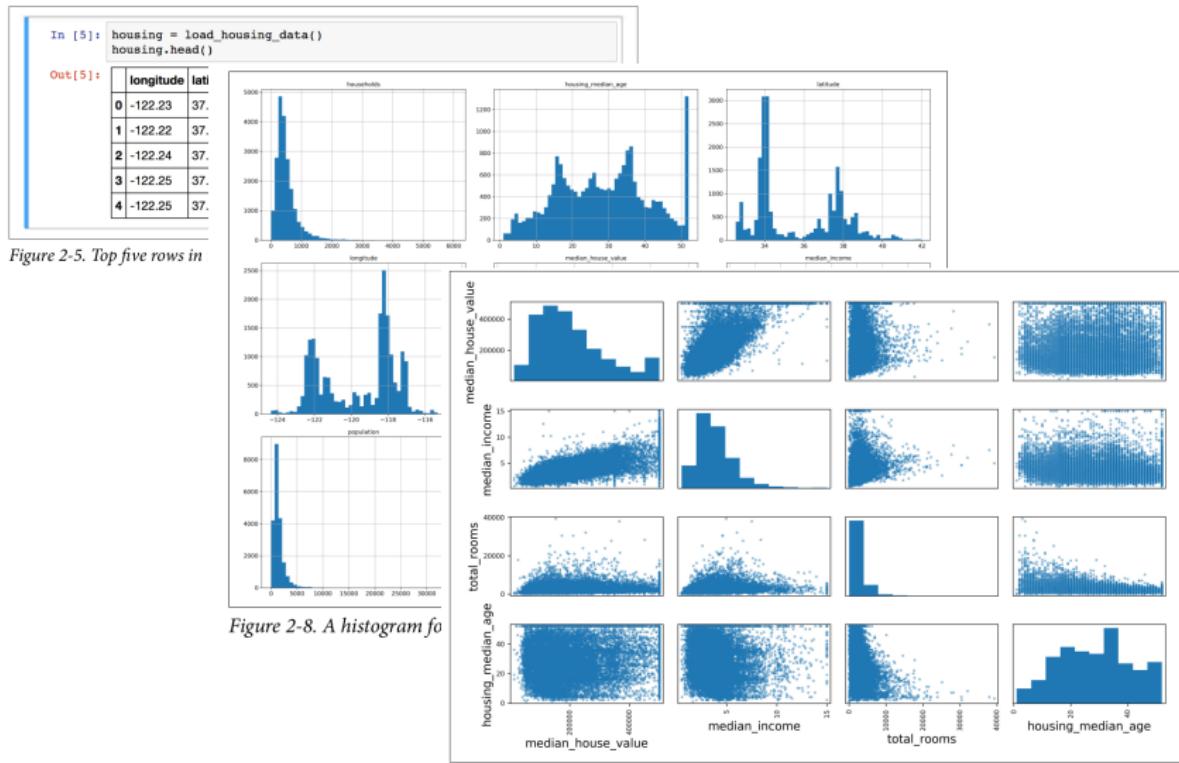


..or for classification:
the **confusion matrix**

Data Science: Visualization

Take a Quick Look at the Data Structure

Let's take a look at the top five rows using the DataFrame's `head()` method (see Figure 2-5).



Data Science: Visualization

..why so many sales at median_house_value = 350 K\$ and 500 K\$??

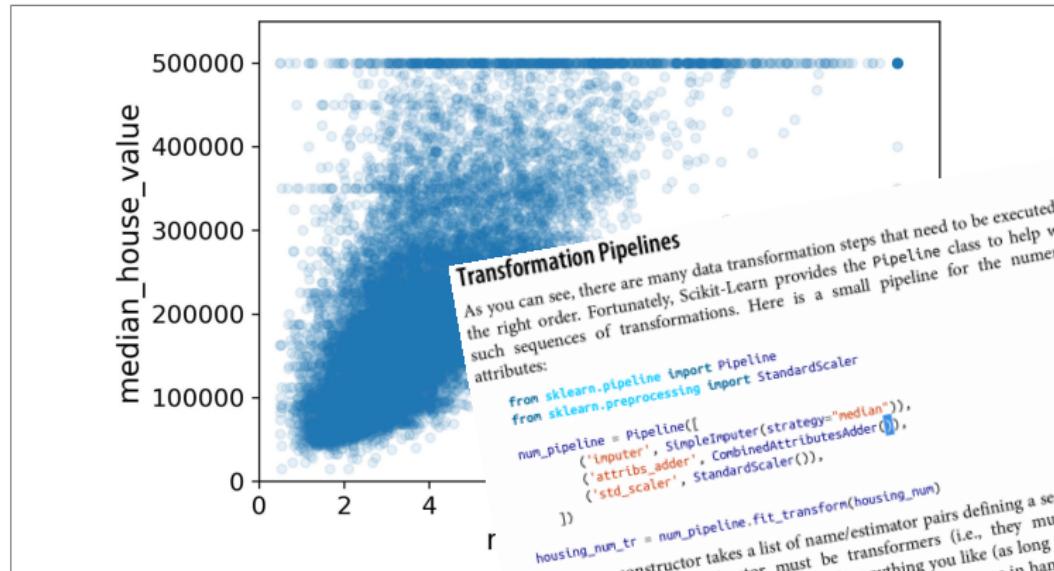


Figure 2-16. Median income versus media

..fixing data via pipelines..

As you can see, there are many data transformation steps that need to be executed in the right order. Fortunately, Scikit-Learn provides the Pipeline class to help with such sequences of transformations. Here is a small pipeline for the numerical attributes:

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy="median")),
    ('attribs_adder', CombinedAttributesAdder()),
    ('std_scaler', StandardScaler())
])
housing_num_tr = num_pipeline.fit_transform(housing_num)
```

The Pipeline constructor takes a list of name/estimator pairs defining a sequence of steps. All but the last estimator must be transformers (i.e., they must have a fit_transform() method). The names can be anything you like (as long as they are unique and don't contain double underscores " __ "): they will come in handy later for hyperparameter tuning.

When you call the pipeline's fit() method, it calls fit_transform() sequentially on all transformers, passing the output of each call as the parameter to the next call, until it reaches the final estimator, for which it just calls the fit() method.

PIPELINES

Putting it all together in Python code...



Pipelines and Preprocessing of Data

Normalization via Scaling or Standardization

Why the need for preprocessing?

*Standardization of datasets is a **common requirement for many machine learning estimators** [...] they might **behave badly** if the individual features do not more or less look like standard normally distributed data. [...]*

[<https://scikit-learn.org/stable/modules/preprocessing.html>]

Standardization of a feature vector \mathbf{x} , giving \mathbf{x}' mean zero, and standard deviation one

$$\mathbf{x}' = \frac{\mathbf{x} - \mu_{\mathbf{x}}}{\sigma_{\mathbf{x}}}$$

What kind of estimators needs preprocessing?

→ **Neural networks (NNs) in particular!**

What is the difference between **Standardization** and **Scaling**?

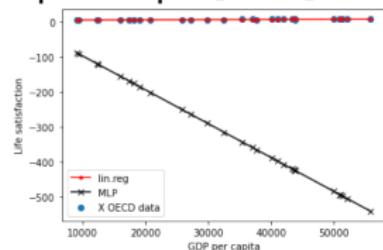
Pipelines and Polynomial Regression

Exercise: pipelines.ipynb: Revisit the OECD data in OIs for MLPs

Feature: GDP per capita feature in range 10K to 50K \$.

But MLP expects input in the range [0;1] or perhaps [-1;1].

```
1 # Manual scaling..
2 X_min = np.min(X)
3 X_max = np.max(X)
4 s = X_max - X_min
5
6 print(f"X_min={X_min:.0f}, X_max={X_max:.0f}, s={s:.0f}")
7
8 X_scaled = (X-X_min)/s
9 print(f"X_scaled.shape={X_scaled.shape}")
10 print(f"np.min(X_scaled)={np.min(X_scaled)}")
11 print(f"np.max(X_scaled)={np.max(X_scaled)}")
12
13 mlp.fit(X_scaled ,y.ravel())
14 y_pred_mlp = mlp.predict((M-X_min)/s)
15
16 plt.plot(m, y_pred_lin, "r")
17 plt.plot(m, y_pred_mlp, "k")
18
19 print(f"mpl.score={mlp.score(X_scaled, y.ravel()):0.2f}")
```



Prints:
X_min=9055,
X_max=55805, s=46750
X_scaled.shape=(29, 1)
np.min(X_scaled)=0.0
np.max(X_scaled)=1.0
mpl.score=0.70

Pipelines

OECD Data and MLPs: introducing a MinMaxScaler

```
1 # Now, do the same but via a pipeline..
2 from sklearn.preprocessing import MinMaxScaler
3
4 scaler = MinMaxScaler()
5 scaler.fit(X)
6 X_scaled = scaler.transform(X)
7 M_scaled = scaler.transform(M)
8
9 mlp.fit(X_scaled, y)
10 y_pred_mlp = mlp.predict(M_scaled)
11
12 print(f"mpl.score={mlp.score(X_scaled, y):0.2f}")
13
14 # PRINTS: mpl.score=0.71
```

Pipelines

OECD Data and MLPs: putting everything in a Full Pipeline

```
1 # Or even better, in a full pipeline..
2 from sklearn.pipeline import Pipeline
3
4 pipe = Pipeline( # indent pipeline as VHDL port mappings!
5     [
6         ('scaler', MinMaxScaler()),
7         ('mlp', mlp)
8     ]
9 )
10
11 pipe.fit(X, y)
12
13 print(f"pipe.score(..)={pipe.score(X, y):0.2f}")
14
15 # PRINTS: pipe.score(..)=0.68
```

Pipelines and Polynomial Regression

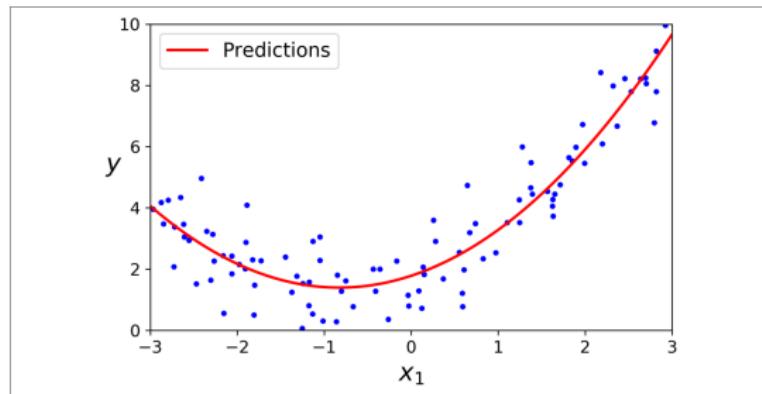


Figure 4-13. Polynomial Regression model predictions

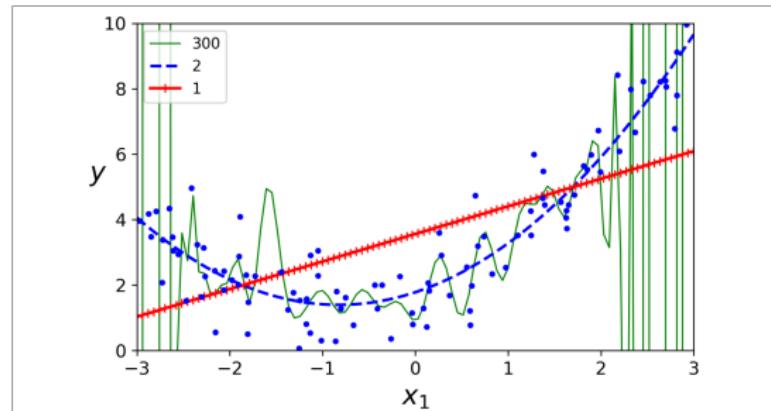


Figure 4-14. High-degree Polynomial Regression

Pipelines and Polynomial Regression

Python code from Opgave: L07/capacity_under_overfitting.ipynb

```
1  from sklearn.pipeline import Pipeline
2  from sklearn.preprocessing import PolynomialFeatures
3  from sklearn.linear_model import LinearRegression
4  from sklearn.model_selection import cross_val_score
5
6  polynomial_features = PolynomialFeatures(degree=degrees[i], ..
7  linear_regression = LinearRegression()
8
9  pipeline = Pipeline(
10    [
11      ("polynomial_features", polynomial_features),
12      ("linear_regression", linear_regression)
13    ]
14 )
15
16 pipeline.fit(X[:, np.newaxis], y)
17
18 scores = cross_val_score(
19   pipeline, X[:, np.newaxis], y,
20   scoring="neg_mean_squared_error", cv=10
21 )
22 score_mean = -scores.mean()
```

Pipelines and K-fold CV

Evaluate a score by cross-validation: `cross_val_score(..)`



Install User Guide API Examples More ▾

Prev Up Next

scikit-learn 0.24.1
Other versions

Please cite us if you use the software.

sklearn.model_selection.cross_val_score
Examples using sklearn.model_selection.c

sklearn.model_selection.cross_val_score

sklearn.model_selection.cross_val_score(*estimator*, *X*, *y=None*, *, *groups=None*, *scoring=None*, *cv=None*, *n_jobs=None*, *verbose=0*, *fit_params=None*, *pre_dispatch='2*n_jobs'*, *error_score=nan*) [source]

Evaluate a score by cross-validation

Read more in the [User Guide](#).

Parameters:

- estimator* : estimator object implementing 'fit'**
The object to use to fit the data.
- X* : array-like of shape (*n_samples*, *n_features*)**
The data to fit. Can be for example a list, or an array.
- y* : array-like of shape (*n_samples*,) or (*n_samples*, *n_outputs*), default=None**
The target variable to try to predict in the case of supervised learning.
- groups* : array-like of shape (*n_samples*,), default=None**
Group labels for the samples used while splitting the dataset into train/test set. Only used in conjunction with a "Group" *cv* instance (e.g., `GroupKFold`).
- scoring* : str or callable, default=None**
A str (see model evaluation documentation) or a scorer callable object / function with signature `scorer(estimator, X, y)` which should return only a single value.
- Similar to `cross_validate` but only a single metric is permitted.**
- If None, the estimator's default scorer (if available) is used.

***cv* : int, cross-validation generator or an iterable, default=None**