OPITZ CONSULTING

Überraschend mehr Möglichkeiten

KI-Boom – Fallstricke im Maschinellen Lernen – Vermeidung von Fehlern und Imageschäden

Milen Großmann, Developer

Inhalte

1 Grundlagen

2 Fallstricke

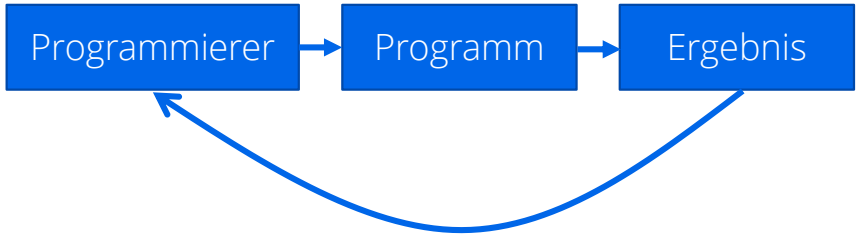
Grundlagen

- Was ist Maschinelles Lernen?
- Arten des Maschinellen Lernens
- Wie bewertet man Modelle?

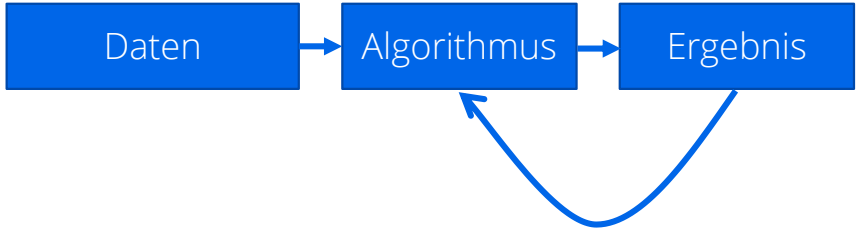
Was ist Maschinelles Lernen?

- = Lehre von Algorithmen, die die Fähigkeit haben, selbstständig aus Daten zu lernen ohne explizit dazu programmiert worden zu sein
- Anwendungsfälle:
  - Intrusion Detection in Computersystemen
  - Fraud Detection im Finanzwesen
  - Gesichtserkennung
  - Predictive Maintenance
  - Chatbots, Spam Filter

- “Früher”:

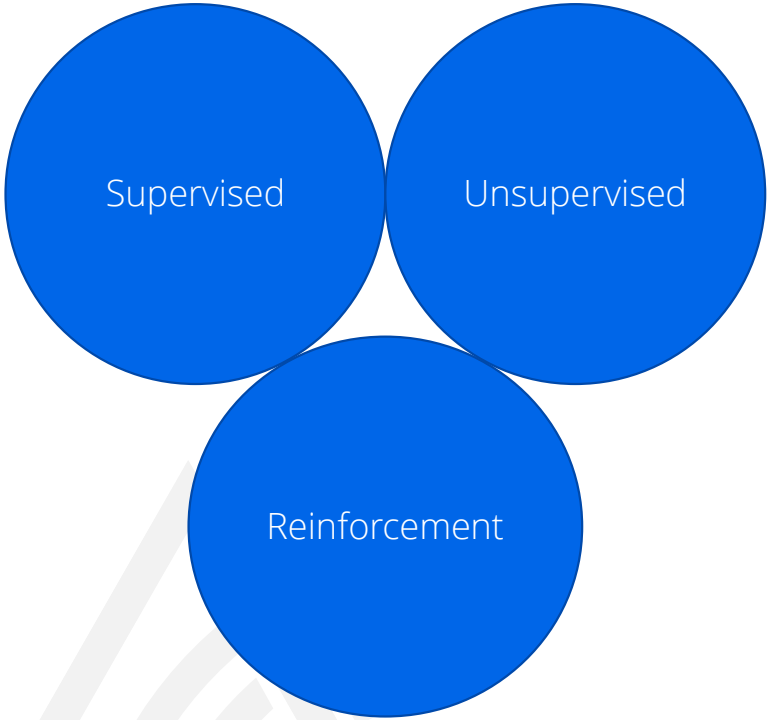


- “Heute”:



Was ist Maschinelles Lernen?

Arten



Was ist Maschinelles Lernen?

Arten

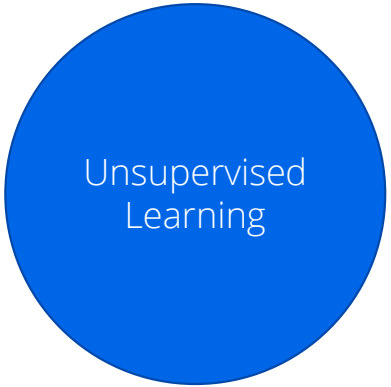
Supervised Learning

- Daten und zugehörige Ergebnisvariablen sind bekannt
- Algorithmus lernt Struktur

→Verallgemeinerung auf ungesehene Daten

Was ist Maschinelles Lernen?

Arten



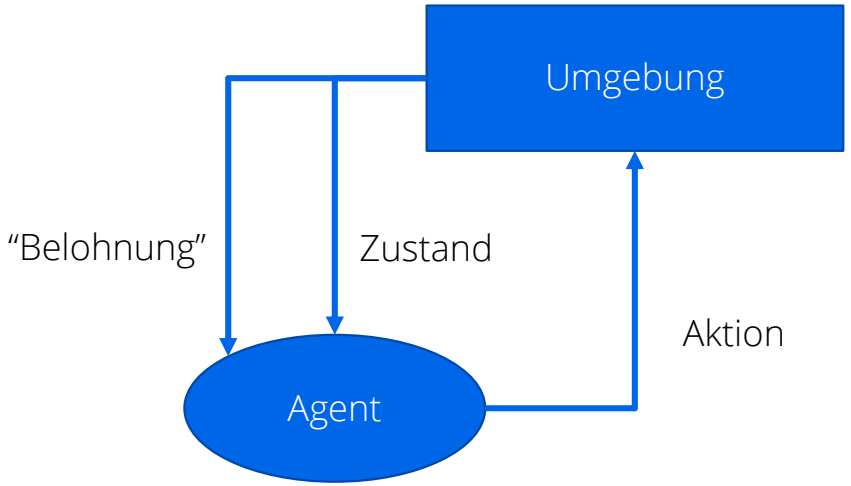
Unsupervised Learning

- Ergebnisvariablen sind unbekannt
- Algorithmus findet selbstständig Struktur in Daten

Was ist Maschinelles Lernen?

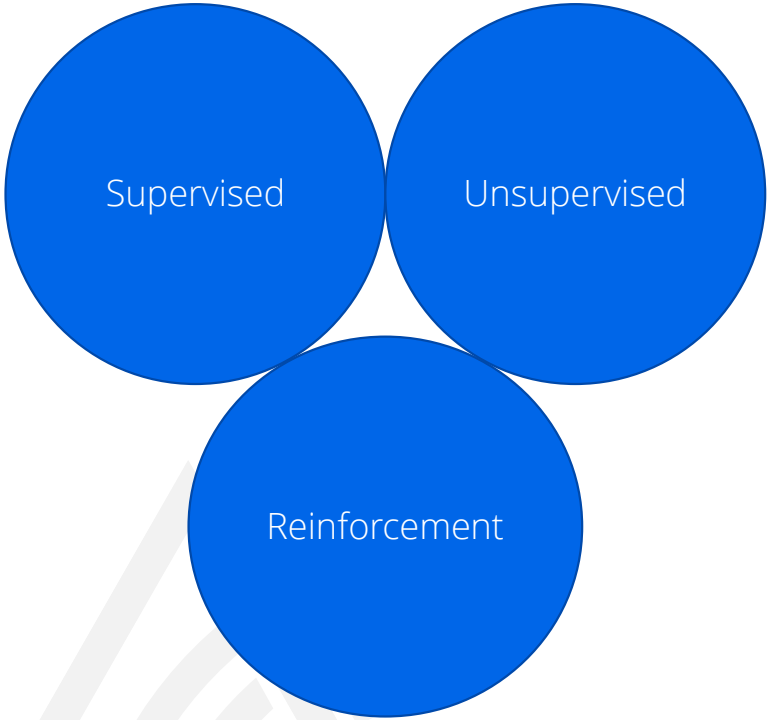
Arten

Reinforcement Learning



Was ist Maschinelles Lernen?

Arten

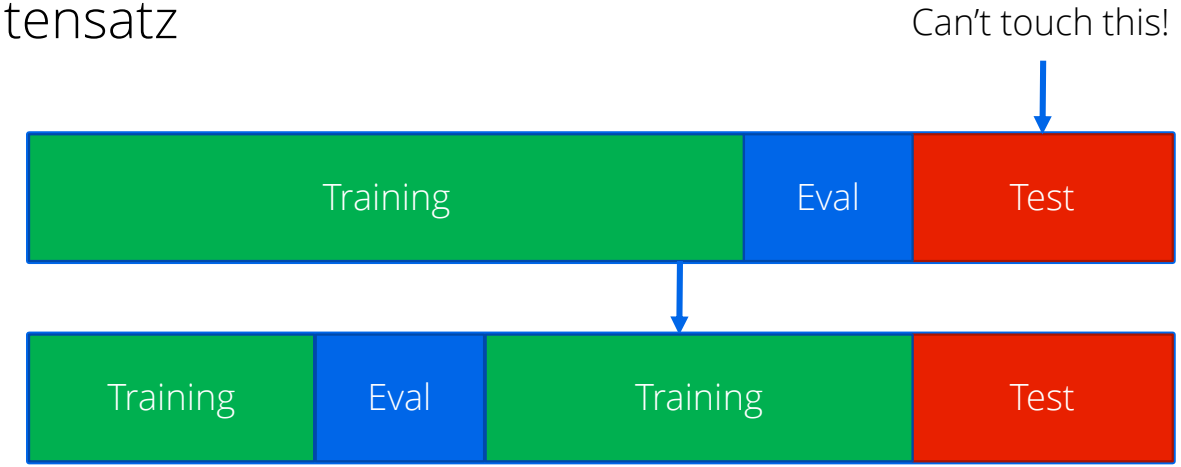


Potentielle Gefahren

- Finanzielle Einbußen
- Imageschäden
- Sicherheitslücken
- Systemausfälle oder Systemfehlverhalten

Wie bewertet man ML-Modelle?

- Erzeugung Testdatensatz für finale Bewertung
- Unterteilung des Datensatzes in Trainings- und Evaluationsdaten
- Trainieren des Algorithmus mit versch. Einteilungen
- Final: Test mit Testdatensatz



Fallstricke

- Missachtung von Klassengewichtungen
- Overfitting
- Underfitting
- Data Snooping
- Misinterpretation von Daten

Missachtung von Klassengewichtungen

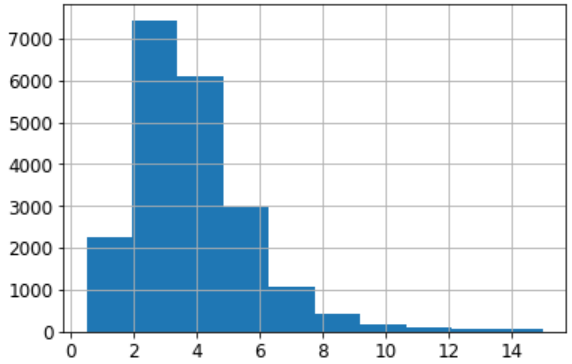
- Beispiel:
  - Umfrage unter 1000 Personen
  - Einteilung in 500 Männer & 500 Frauen → Fehler!
  - 51.3% Männer, 48.7% Frauen → korrekte Einteilung: 513 Männer & 487 Frauen
- Abhilfe: Stratified Sampling
  - = Einteilung von Datensätzen in homogene Untergruppen (Strata)
  - Repräsentative Menge an Samples wird aus Daten gezogen
  - Strata sollten nicht zu zahlreich sein

Missachtung von Klassengewichtungen

```
1. # Plotte Einkommenskategorien in 1000$
2. housing["median_income"].hist()
```

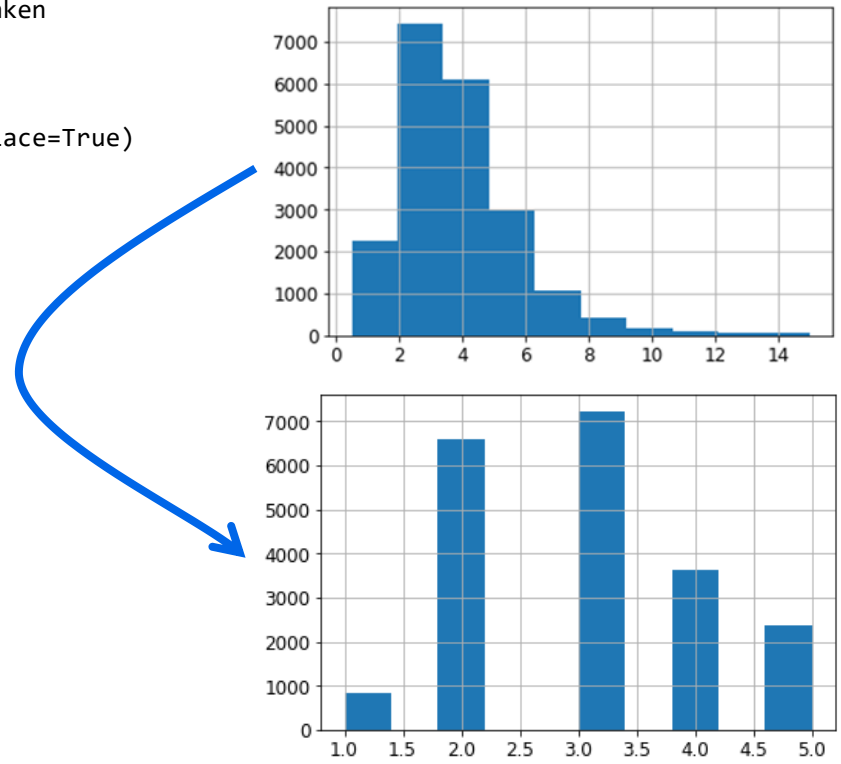
- Zu viele Untergruppen
- Nicht genügend Datensätze pro Gruppe → inhomogen

→Voreingenommenheit



Missachtung von Klassengewichtungen

```
1. # Teile durch 1.5 um Anzahl der Einkommenskategorien zu beschränken
2. housing["income_cat"] = np.ceil(housing["median_income"] / 1.5)

3. # Fasse alle Einkommenskategorien >5 als EK 5 zusammen
4. housing["income_cat"].where(housing["income_cat"] < 5, 5.0, inplace=True)

5. # Plot
6. housing["income_cat"].hist()
```



Missachtung von Klassengewichtungen

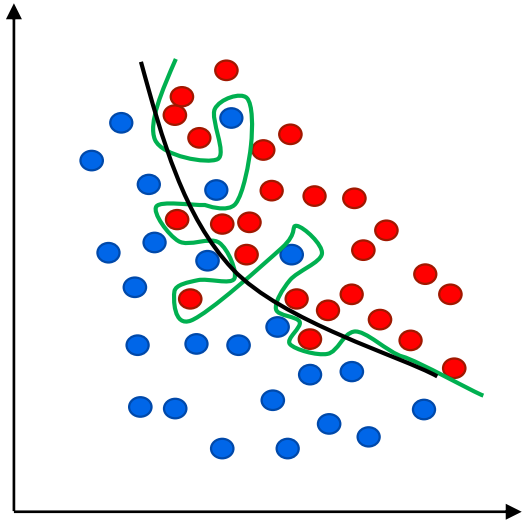
```
1.  def income_cat_proportions(data):
2.      return data["income_cat"].value_counts() / len(data)

3.  train_set, test_set = train_test_split(housing, test_size=0.2,
    random_state=42)

4.  compare_props = pd.DataFrame({
5.      "Overall": income_cat_proportions(housing),
6.      "Stratified": income_cat_proportions(strat_test_set),
7.      "Random": income_cat_proportions(test_set),
8.  }).sort_index()
9.  compare_props["Rand. %error"] = 100 * compare_props["Random"] /
    compare_props["Overall"] - 100
10. compare_props["Strat. %error"] = 100 * compare_props["Stratified"] /
    compare_props["Overall"] – 100

11. print(compare_props)
```

- Deutlich geringere Abweichung mit Stratified Sampling

	Overall	Random	Stratified	Rand. %error	Strat. %error
1.0	0.039826	0.040213	0.039729	0.973236	-0.243309
2.0	0.318847	0.324370	0.318798	1.732260	-0.015195
3.0	0.350581	0.358527	0.350533	2.266446	-0.013820
4.0	0.176308	0.167393	0.176357	-5.056334	0.027480
5.0	0.114438	0.109496	0.114583	-4.318374	0.127011

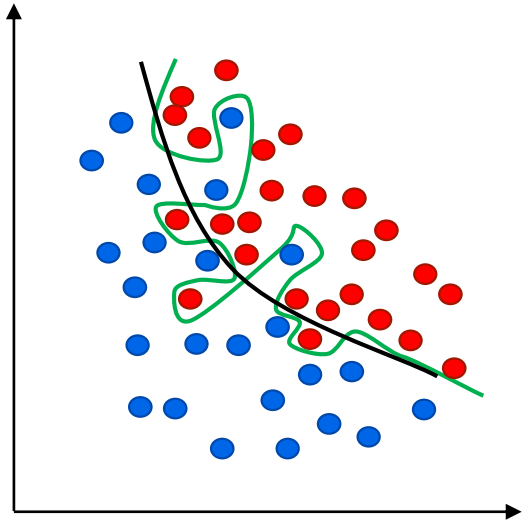
Overfitting

- Modellkomplexität zu hoch
- Modell lernt Rauschen statt Datenmuster
- Schlechte Verallgemeinerung auf unbekannte Daten
- Analogien:
  - Klischees
  - Kenntnis von Prüfungsfragen vor der Prüfung
- Häufiger Hinweis: zu hohe Modellgenauigkeit



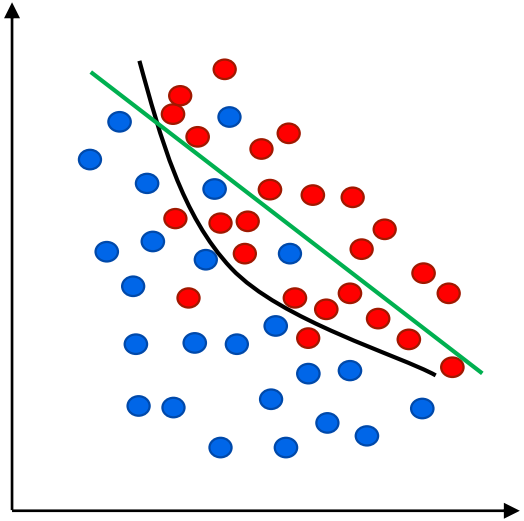
Overfitting

- Mögliche Gegenmaßnahmen:
  - Regularisierung
  - mehr Daten (kaufen, messen, generieren)
  - Rauschen reduzieren



Underfitting

- Modellkomplexität zu gering
- Schlechte Modellperformance bei Testdaten und neuen Daten
- Mögliche Gegenmaßnahmen:
  - Wahl eines komplexeren Modells
  - Verwendung von Features höherer Qualität (feature engineering)
  - Modelleinschränkungen reduzieren



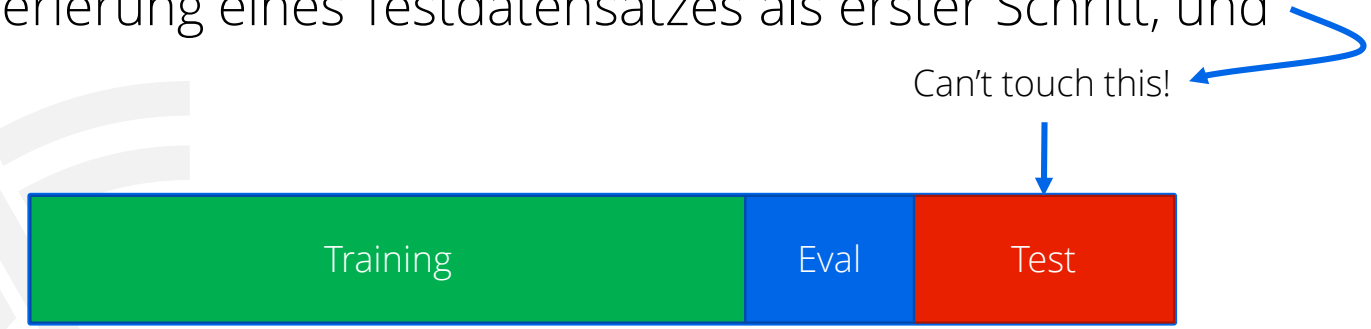
Data Snooping
“Menschliches Overfitting”

- Betrachten von Testdaten

→“Zurechtbiegen” der Realität

→Falsche Modelle

- Auch durch Hyperparametertuning **und** Modellevaluation durch Validation mit gleichen Datensätzen
- Abhilfe: Generierung eines Testdatensatzes als erster Schritt, und

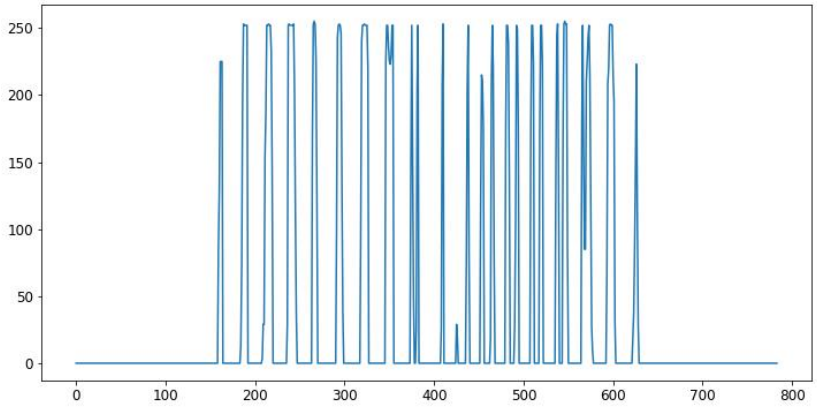


“Know your Data”!

```
1. from sklearn.datasets import fetch_mldata

2. data = fetch_mldata(‘big surprise’)
3. X, y = data[‘data’], data[‘target’]

4. print(X.shape)
5. # Output: (70000, 784)

6. plt.figure(figsize=(12, 6))
7. plt.plot(X[36000])
8. plt.show()
```

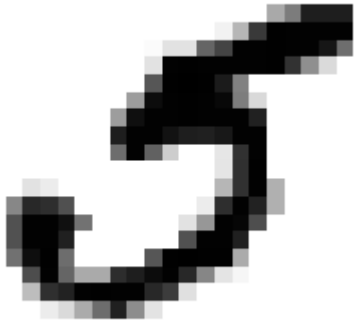
- Sensordaten?
- Energieverbrauch, Schaltfrequenz?

→Unsinn!



“Know your Data”!

```
1.  from sklearn.datasets import fetch_mldata
2.  mnist = fetch_mldata('MNIST original')

3.  X, y = mnist["data"], mnist["target"]

4.  print(28*28)  # output: 784

5.  some_digit = X[36000]
6.  some_digit_image = some_digit.reshape(28, 28)
7.  plt.imshow(some_digit_image, cmap = matplotlib.cm.binary,
8.             interpolation="nearest")
9.  plt.axis("off")

10. save_fig("some_digit_plot")
11. plt.show()
```

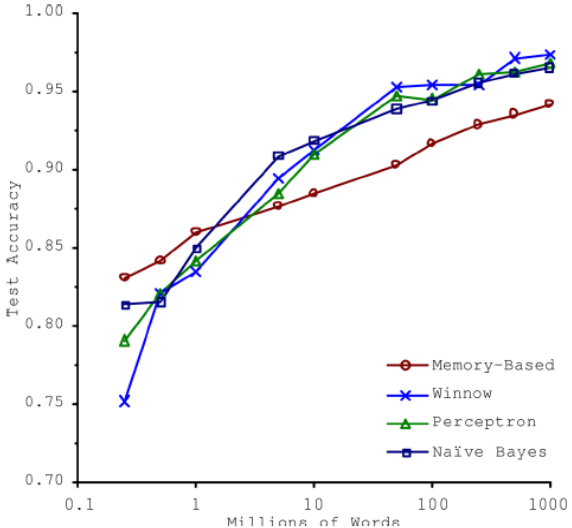


“Know your Data”!

```
1.  def plot_digit(data):
2.      image = data.reshape(28, 28)
3.      plt.imshow(image, cmap = matplotlib.cm.binary,
4.                 interpolation="nearest")
5.      plt.axis("off")

6.  def plot_digits(instances, images_per_row=10, **options):
7.      size = 28
8.      images_per_row = min(len(instances), images_per_row)
9.      images = [instance.reshape(size,size) for instance in instances]
10.     n_rows = (len(instances) - 1) // images_per_row + 1
11.     row_images = []
12.     n_empty = n_rows * images_per_row - len(instances)
13.     images.append(np.zeros((size, size * n_empty)))
14.     for row in range(n_rows):
15.         rimages = images[row * images_per_row : (row + 1) * images_per_row]
16.         row_images.append(np.concatenate(rimages, axis=1))
17.     image = np.concatenate(row_images, axis=0)
18.     plt.imshow(image, cmap = matplotlib.cm.binary, **options)
19.     plt.axis("off")

20. plt.figure(figsize=(9,9))
21. example_images = np.r_[X[:12000:600], X[13000:30600:600], X[30600:60000:590]]
22. plot_digits(example_images, images_per_row=10)
23. save_fig("more_digits_plot")
24. plt.show()
```

“Know your Data”!

- Seien Sie sich bewusst wo Ihre Daten herkommen und was sie aussagen!
- Wissen Sie um die Qualität der Daten!
- Algorithmenwahl ohne Annahmen ist sinnlos (“No Free Lunch Theorem”)
- Je mehr Daten, desto unwichtiger der Algorithmus



Quelle: “The Unreasonable Effectiveness of Data” Peter Norvig et al. (2009)

Vielen Dank!

Milen Großmann

Developer

Landwehr 2
22087 Hamburg

milen.grossmann@opitz-consulting.com
+49 40 741122-1357

WWW.OPITZ-CONSULTING.COM

@OC_WIRE

OPITZCONSULTING

opitzconsulting

opitz-consulting-bcb8-1009116