Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

2024 FS CAS PML
1 Feature Engineering
1.4 Konstruktion

Werner Dähler 2024

1 Feature Engineering - AGENDA

11. Einführung
12. Exploration
13. Transformation
14. Konstruktion
    141. Ableiten aus bestehenden Variablen
    142. Dimensionsreduktion (mit PCA)
15. Selektion
16. Implementation
17. Nachträge

1.4 Feature Engineering - Konstruktion

- darunter versteht man das Erstellen neuer Variablen durch Ableiten aus bestehenden
- Ziele:
  - reduzieren der Komplexität
  - vermeiden von Korrelationen

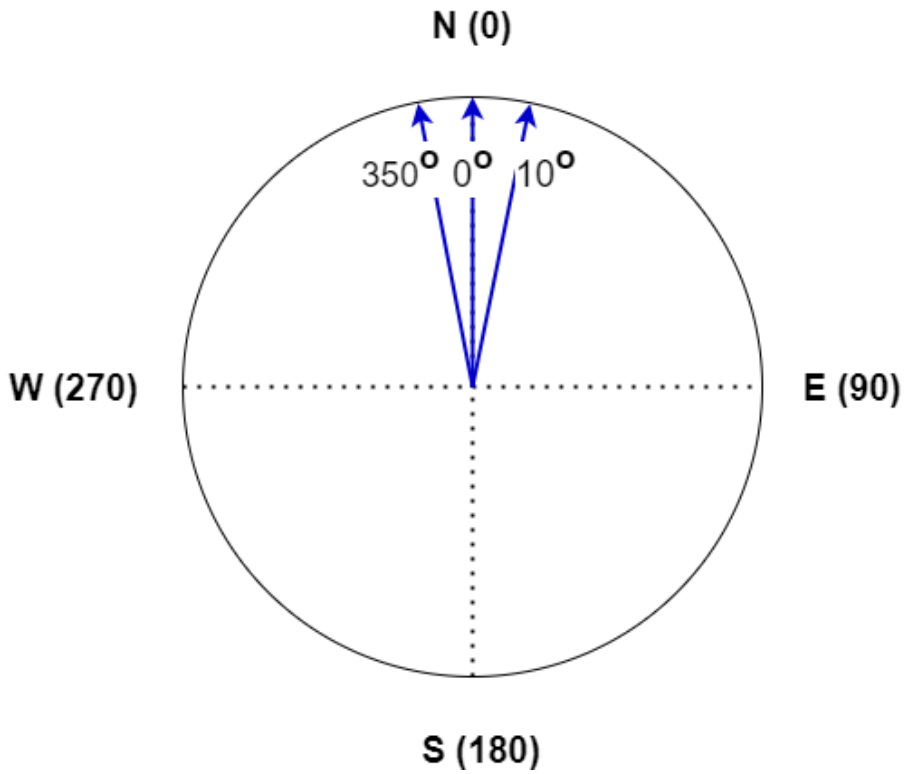
- zwei Techniken stehen hier im Vordergrund
  - gezieltes Konstruieren von neuen Variablen aus bestehenden und ersetzen der Ausgangsvariablen
  - Dimensionsreduktion mit Methoden des Nichtüberwachten Lernens (dies als Ausblick)

1.4 Feature Engineering - Konstruktion

1.4.1 Ableiten aus bestehenden Variablen

Beispiel Wetterdaten

- die quantitative Beschreibung von Wettersituationen besteht meist aus einer Anzahl unterschiedlicher Merkmale wie
  - Temperatur, Luftdruck, Luftfeuchtigkeit, Niederschlagsmenge, Bewölkungsgrad etc.
  - ausserdem Windgeschwindigkeit und Windrichtung
- während die meisten metrisch skaliert und damit für Machine Learning unproblematisch sind, trifft dies für die Windrichtung nicht zu
- diese kann in unterschiedlicher Form vorliegen
  - nominal oder ordinal (N, E, S, W oder N, NE, E, SE, ...)
  - metrisch als Abweichung in Grad gegenüber Norden im Uhrzeigersinn
- letzteres ist insofern problematisch, als z.B. eine kleine Abweichung von N in Richtung W zu einem sehr grossen Wert führt (< 360º) während N selber 0º beträgt



1.4 Feature Engineering - Konstruktion

1.4.1 Ableiten aus bestehenden Variablen

Abhilfe

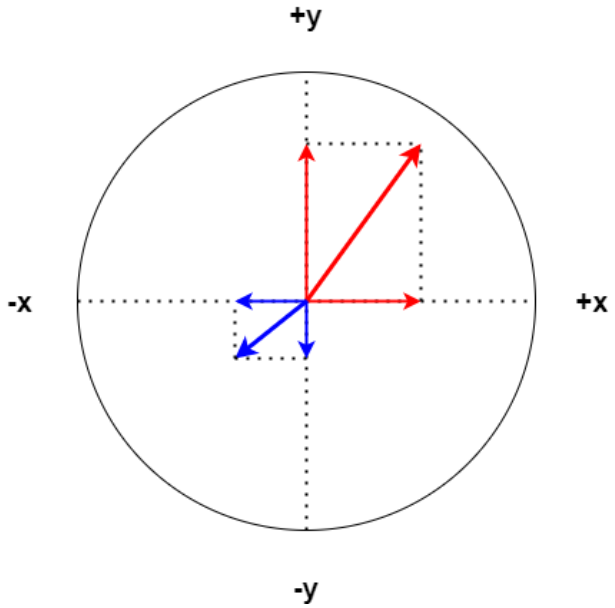
- zerlegen der Richtung in eine x- und y-Komponente mit Hilfe trigonometrischer Funktionen, z.B. die W-E Achse als x, die N-S Achse als y
- multiplizieren der neuen Komponenten mit Geschwindigkeit, formal:

$$x = sin\left(\frac{direction \cdot \pi}{180}\right) \cdot speed \qquad y = cos\left(\frac{direction \cdot \pi}{180}\right) \cdot speed$$

- oder als Code

```
import numpy as np
data['x'] = np.sin(data.direction * np.pi / 180) * data.speed
data['y'] = np.cos(data.direction * np.pi / 180) * data.speed
```

- die Ausgangsvariablen direction und speed können danach aus dem Data Frame entfernt werden



1.4 Feature Engineering - Konstruktion

1.4.1 Ableiten aus bestehenden Variablen

Beispiel Datum Typ

- im Melbourne Housing Dataset hat es die Variable "Date", welche ein Kalenderdatum enthält

```
:
print(data.Date.head())
```

```
0    3/12/2016
1    4/02/2016
2    4/03/2017
3    4/03/2017
4    4/06/2016
```

- ohne besondere Vorkehrungen werden Timestamps als String (object) Type in Pandas Dataframes eingelesen
- im Folgenden wird dargestellt, wie solche Daten in den Typ `datetime` umgewandelt und dann einzelne Komponenten wie Jahr, Monat, Tag etc. extrahiert werden können

1.4 Feature Engineering - Konstruktion

1.4.1 Ableiten aus bestehenden Variablen

- in einem ersten Schritt wird die Variable in den Typ `datetime` konvertiert, zu Demonstrationszwecken gleich in eine neue Variable "date_dt"

```
data['date_dt'] = pd.to_datetime(data.Date, format="%d/%m/%Y")
```

- daraus können anschliessend einzelne Komponenten wie Tag, Monat und Jahr extrahiert werden, auch hier als neue Variablen im Data Frame

```
data['year'] = data.date_dt.dt.year
data['month'] = data.date_dt.dt.month
data['day'] = data.date_dt.dt.day
```

- als weitere Möglichkeit wird hier noch die Differenz zu einem Startdatum (1.1.2016) ermittelt und im Data Frame hinterlegt

```
start_date = pd.to_datetime('1/1/2016', format="%d/%m/%Y")
data['daydiff'] = (data.date_dt - start_date).dt.days
```

1.4 Feature Engineering - Konstruktion

1.4.1 Ableiten aus bestehenden Variablen

- das Ergebnis:

```
print(data[['Date', 'date_dt', 'day', 'month', 'year', 'daydiff']].head())
```

```
        Date    date_dt  day  month  year  daydiff
0  3/12/2016 2016-12-03    3     12  2016      337
1  4/02/2016 2016-02-04    4      2  2016       34
2  4/03/2017 2017-03-04    4      3  2017      428
3  4/03/2017 2017-03-04    4      3  2017      428
4  4/06/2016 2016-06-04    4      6  2016      155
```

- neben den hier gezeigten können weitere Komponenten extrahiert werden (falls vorhanden) wie hour, minute, second, etc.
  vgl. Online Ref: https://pandas.pydata.org/docs/reference/api/pandas.Series.dt.date.html
- ausserdem können mit der entsprechenden Parametrisierung Variablen mit Datum bereits beim Einlesen mit `.read_csv()` konvertiert werden (vgl. [ipynb])

1.4 Feature Engineering - Konstruktion

1.4.2 Dimensionsreduktion mit PCA

- bei mehrdimensionalen Datensätzen bestehen oft Korrelationen zwischen Variablen, d.h. Informationen einer Variable können auch durch andere ausgedrückt werden
- Dekompositionsmethoden bilden mehrdimensionale Datensätze durch Transformationen in solche mit weniger Dimensionen ab, dies unter Wahrung eines Maximums der Ausgangsinformation (im Sinne von Varianz), daher auch Dimensionsreduktion genannt
- können die Daten dabei auf zwei Dimensionen reduziert werden, wird es möglich, mit Hilfe einer Streugrafik (Scatterplot) die Ähnlichkeitsbeziehungen der Ausgangsdaten zu visualisieren, d.h. Datenpunkte, welche näher beieinander liegen sind sich generell ähnlicher als weiter voneinander entfernte

1.4 Feature Engineering - Konstruktion

1.4.2 Dimensionsreduktion mit PCA

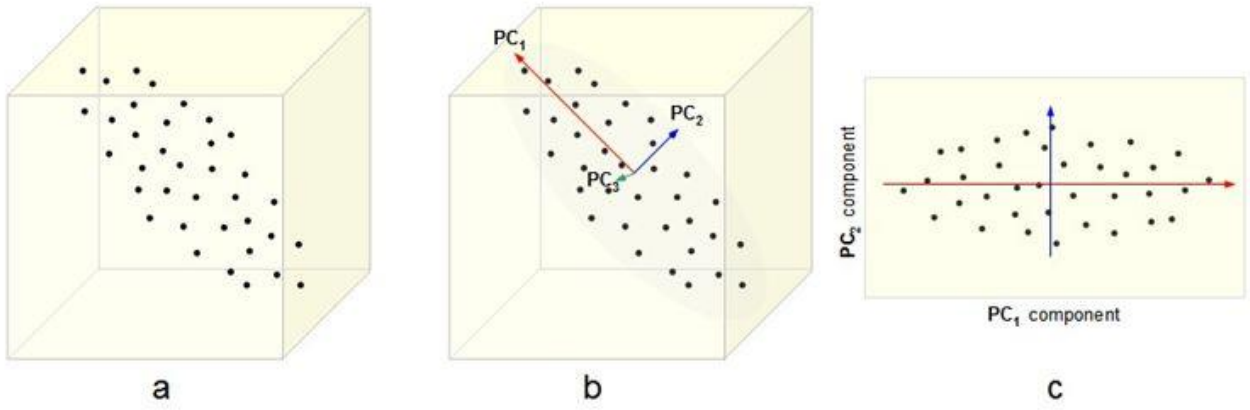
- exemplarisch soll hier das populärste Verfahren dazu vorgestellt werden: Hauptkomponentenanalyse (Principal components analyses, PCA)
  - strukturiert umfangreiche Datensätze durch Benutzung der Eigenvektoren der Kovarianzmatrix
  - eine Vielzahl von Variablen wird dabei durch eine geringere Zahl möglichst aussagekräftiger Linearkombinationen (die Hauptkomponenten) genähert
  - die Hauptkomponenten sind "Richtungen" im mehrdimensionalen Raum mit der (jeweils noch) höchsten Varianz
  - weitere Eigenschaften der ermittelten Hauptkomponenten
    - die Varianzen nehmen von der ersten an kontinuierlich ab
    - sie sind untereinander unkorreliert
  - es werden zwar gleich viele Hauptkomponenten ermittelt, wie Ausgangsvariablen vorliegen, aber nur die ersten sind tatsächlich Informationsträger

1.4 Feature Engineering - Konstruktion

1.4.2 Dimensionsreduktion mit PCA

Visualisierung des oben genannten Prinzips:

- a: Ausgangslage: dreidimensionales Dataset in perspektivischer Darstellung
- b: erste Hauptkomponente (PC1, roter Pfeil): maximale Ausdehnung des Punkteschwarms
- zweite Hauptkomponente (PC2, blauer Pfeil): maximale Ausdehnung senkrecht auf PC1
- dritte Hauptkomponente (PC3): Rest, senkrecht auf PC1 und PC2
- c: Rotation: PC1 wird zur neuen x-Achse, PC2 zur neuen y-Achse



- im Hintergrund wirken Matrixoperationen, welche geschlossene Lösungen darstellen und recht schnell in der Ausführung sind

[Quelle]

1.4 Feature Engineering - Konstruktion

1.4.2 Dimensionsreduktion mit PCA

- Anwendung unter Anderem in der Bildverarbeitung
  - jedes Pixel eines Bildes bildet eine Variable mit einem Helligkeitswert
  - benachbarte Pixel sind meist hoch korreliert
  - durch PCA lässt sich für anschliessende ML-Analysen die Anzahl der Variablen reduzieren, was Korrelationen reduziert sowie Rechenzeit und Speicherplatz sparen hilft

- Zuordnungen der Methoden
  - ursprünglich: multivariater Statistik / Datenanalyse
  - später: Data Mining
  - noch später: Unüberwachtes Lernen
  - heute zunehmend (und in diesem Kurs): Feature Engineering

1.4 Feature Engineering - Konstruktion

1.4.2 Dimensionsreduktion mit PCA

- zur Anwendung von PCA mit Python wird ebenfalls eine Methode aus scikit-learn eingesetzt: `sklearn.decomposition.PCA`
- da es sich dabei um eine Machine Learning Methode handelt, müssen die Daten entsprechend aufbereitet sein, d.h.
  - keine Missing Values
  - nur numerische Daten
  - Bereinigung anderer möglicher Anomalien
- ausserdem wird das Target "y" vom Rest der Daten abgetrennt: "features - target - split"
  - Features $\rightarrow$ "X"
  - Target $\rightarrow$ "y"
- und die Features müssen standardisiert werden
- Code: vgl. [ipynb]

1.4 Feature Engineering - Konstruktion

1.4.2 Dimensionsreduktion mit PCA

Vorgehen mit `sklearn.decomposition.PCA`

- der Umgang mit Trainerklassen erfolgt immer nach demselben Muster
  1. importieren der Klasse aus sklern
  2. instanziieren eines Trainer-Objektes
  3. anwenden des Trainer-Objektes auf die Daten

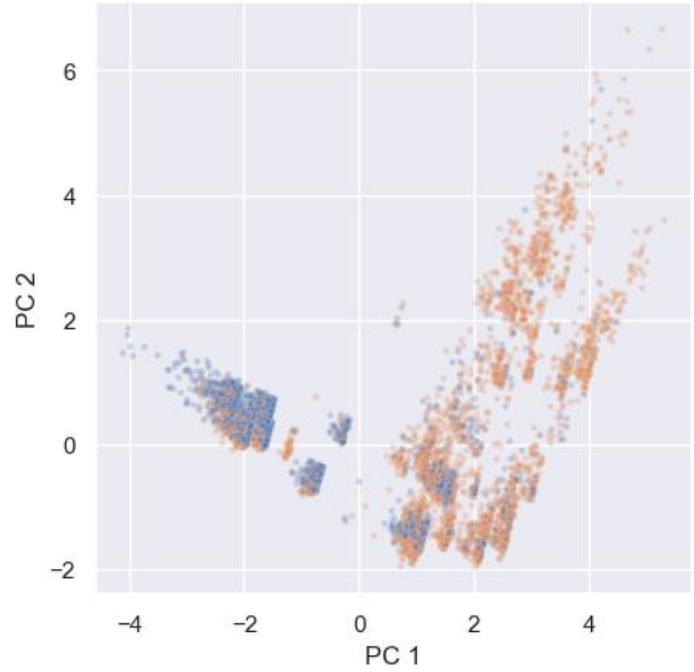
```
from sklearn.decomposition import PCA ## import trainer class
model = PCA()                         ## instantiate trainer object
pred = model.fit_transform(X)         ## train and apply trainer on data
print(pred[:3, :])                    ## check prediction (result, optional)
```

```
[[ 0.82454003 -1.45559011 -1.0976675   0.91524634  0.08531455 -0.15409804
   0.04340854 -0.21781905  0.01493519 -0.01620926]
 [-0.7627286  -0.72786538 -0.32087404 -0.13931963 -0.77314314  0.13132969
   0.59687245 -0.0447055   0.04996993  0.14532455]
 :
```

1.4 Feature Engineering - Konstruktion

1.4.2 Dimensionsreduktion mit PCA

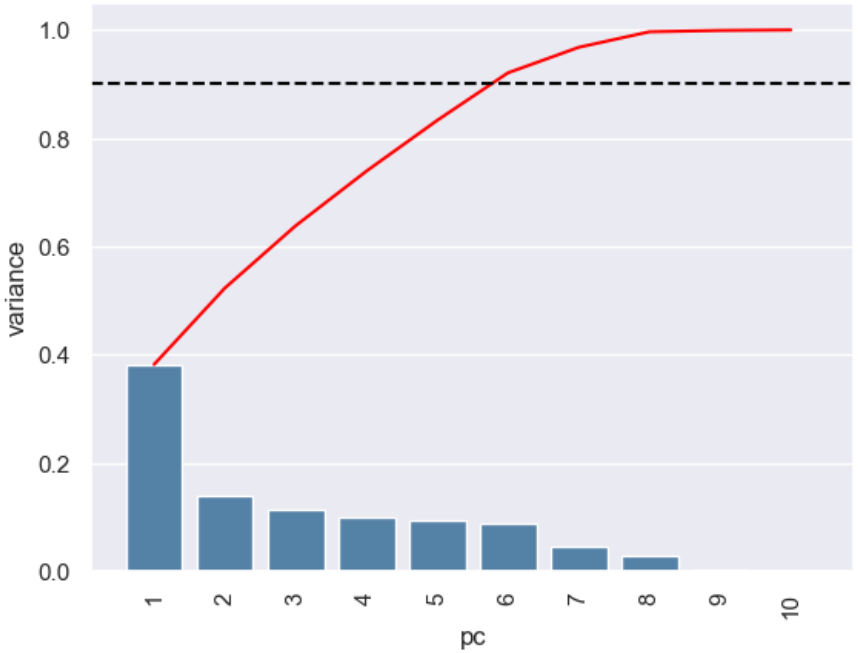
- das Ergebnis ("pred") ist eine Matrix (numpy.ndarray) mit denselben Dimensionen wie die Feature-Matrix und enthält die Koordinaten (Hauptkomponenten) der rotierten Daten
- übliche Visualisierung
  - Scatterplot von "PC 1" vs "PC 2"
  - optional: einfärben nach einem kategorialen Merkmal (hier "y")
- eine vertieftere Analyse der erkennbaren Muster ist allerdings eine Tätigkeit der Datenanalyse und unterbleibt an dieser Stelle



1.4 Feature Engineering - Konstruktion

1.4.2 Dimensionsreduktion mit PCA

- für Feature Engineering ist dagegen interessanter, wie sich die Varianzen der Gesamtdaten in den ermittelten Komponenten wederspiegeln
- das trainierte Modell enthält eine Vielzahl von Metainformationen, z.B.
  - `.n_components_`: Anzahl Hauptkomponenten, entspricht normalerweise der Anzahl Features in den Ausgangsdaten
  - `.explained_variance_ratio_`: relative (normierte) erklärte Varianz pro Hauptkomponente
- letzteres kann in einer sogenannten Paretodiagramm visualisiert werden (vgl. [ipynb])
  - Balken: erklärte Varianz pro Hauptkomponente (kontinuierlich abnehmend)
  - Linie: kumulierte Summen der Varianzen
- dadurch wird beispielsweise erkennbar, dass bereits mit 6 Hauptkomponenten 90% der Gesamtvarianz der Daten abgebildet werden kann (horizontale Linie)



1.4 Feature Engineering - Konstruktion

1.4.2 Dimensionsreduktion mit PCA

- schliesslich soll noch überprüft werden, ob die ermittelten Hauptkomponenten untereinander tatsächlich unkorreliert sind, wie dies bei den Anforderungen skizziert worden war (vgl. [ipynb])

	PC 1	PC 2	PC 3	PC 4	PC 5	PC 6	PC 7	PC 8	PC 9	PC 10
PC 1	1.000000	-0.000000	0.000000	0.000000	-0.000000	-0.000000	-0.000000	-0.000000	0.000000	0.000000
PC 2	-0.000000	1.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	-0.000000
PC 3	0.000000	0.000000	1.000000	-0.000000	0.000000	-0.000000	-0.000000	-0.000000	0.000000	-0.000000
PC 4	0.000000	-0.000000	-0.000000	1.000000	-0.000000	0.000000	-0.000000	0.000000	0.000000	0.000000
PC 5	-0.000000	0.000000	0.000000	-0.000000	1.000000	-0.000000	0.000000	0.000000	0.000000	0.000000
PC 6	-0.000000	0.000000	-0.000000	0.000000	-0.000000	1.000000	-0.000000	0.000000	0.000000	0.000000
PC 7	-0.000000	0.000000	-0.000000	-0.000000	0.000000	-0.000000	1.000000	-0.000000	0.000000	-0.000000
PC 8	-0.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	-0.000000	1.000000	0.000000	-0.000000
PC 9	0.000000	-0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	-0.000000
PC 10	0.000000	-0.000000	-0.000000	0.000000	0.000000	0.000000	-0.000000	-0.000000	-0.000000	1.000000