DataCraft

3.5 Machine Learning

Ausreißererkennung

DataCraft

Was sind Ausreißer?

DataCraft

Was sind Ausreißer?

Definition aus der Statistik: Messwerte, die sich extrem von anderen Werten unterscheiden und nicht zu den erwarteten Werten passen

Doch Ausreißer ist nicht gleich Ausreißer, es muss unterschieden werden zwischen:

- „Echte“ Ausreißer
- „Falsche“ Ausreißer

DataCraft

Was sind „echte“ Ausreißer?

- Messungen, die mit richtigen Instrumenten vorgenommen worden sind und richtig übertragen worden, sich jedoch extrem von den anderen gemessenen Werten unterscheiden. Beispiel in der Psychologie aus Krankheitstests (z.B.: Depression, Stress, Burnout…): Im Normalfall haben die meisten Testpersonen sehr niedrige Werte, außer die, die wirklich krank sind. Durch die Datenmenge sind diese Personen in der Messung dann Ausreißer
- Diese Ausreißer sind zumeist besonders interessant für die Auswertung und sollten in jedem Fall beibehalten werden

Was sind „falsche“ Ausreißer?

- Messungen die abweichend sind, weil:
  - Fehlerhaftes Messinstrument (weil Messinstrument kaputt / ungeeignet (analoge Waage vs digitale Waage), oder Messung in falschen Einheiten mit Angaben in Fuß und Inch statt Zentimeter)
  - Übertragungsfehler der Messungen in die Datenbank (ganz klassisch: Das Datumsproblem von Excel)

- Diese Messungen gilt es bestmöglich zu korrigieren oder aus den Daten zu entfernen / neu zu erheben.

DataCraft

Wie erkenne ich Ausreißer?

DataCraft

Erste Option: Händisches Vorgehen mit Sortieren in den Daten

Person	Einkommen
1	55000
2	52000
3	50000
4	110000
5	0
6	58000
7	56000

Person	Sortiertes Einkommen
4	110000
6	58000
7	56000
1	55000
2	52000
3	50000
5	0

Erste Option: Händisches Vorgehen mit Sortieren in den Daten in Python

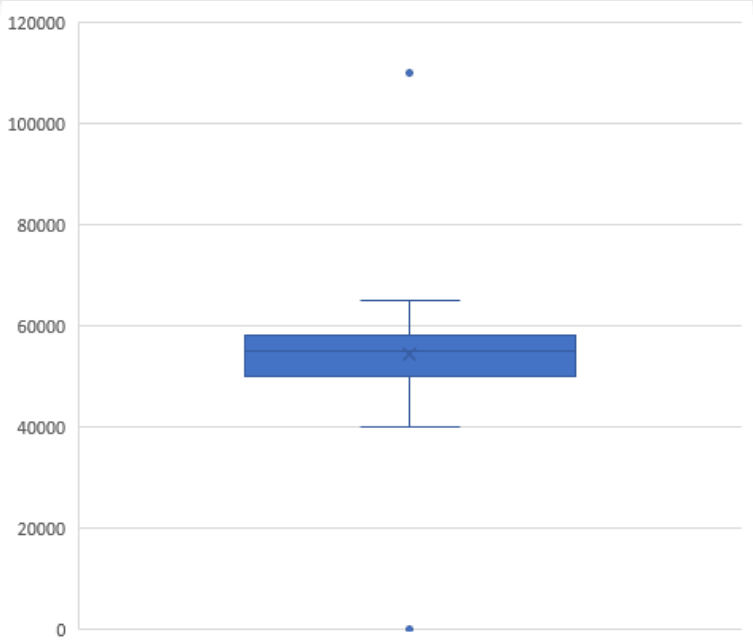
Daten Sortieren:

- sorted_df = df.sort_values('column_name’)

Top und Bottom 10 anzeigen lassen:

- top_10 = sorted_df.tail(10)
- bottom_10 = sorted_df.head(10)

Zweite Option: Grafisch über Boxplots



- Im Boxplot erkennbar: 50% Quantil (Median), Antennen/Whisker (Abweichungsbereiche zu den mittleren 50%) und Ausreißerpunkte

Zweite Option: Grafisch über Boxplots in Python

Nötige Pakete:

- import pandas as pd
- import matplotlib.pyplot as plt
- import seaborn as sns

Boxplot erstellen und Layout einstellen

- sns.boxplot(data=df, y='column_name')
- plt.title('Boxplot of column_name')
- plt.xlabel('Column')
- plt.ylabel('Values')
- plt.show()

Dritte Option: Über die Berechnung des IQR

- In der Statistik gilt die Regel, das alles, was 2*IQR über dem 75%-Quantil und unterhalb des 25%-Quantils liegt, ein Ausreißer darstellt.
- 50% Quantil = Median, der Wert in der Mitte der Verteilung
- Gleiches Prinzip bei 25% und 75%
- InterQuartilsRange = 75%-Quantil – 25%-Quantil

Dritte Option: Über die Berechnung des IQR

Person	Sortiertes Einkommen
4	110000
6	58000
7	56000
1	55000
2	52000
3	50000
5	0

- 25%-Wert: 50000
- 75%-Wert: 58000
- IQR = 58000-50000 = 8000

- Also Ausreißer bei:
- 25%-2*IQR = 50000-2*8000 = < 34000
- 75%+2*IQR = 58000+2*8000 = > 74000

Dritte Option: Über die Berechnung des IQR in Python

Spalte extrahieren:

- column = df['column_name‘]

Quantile und IQR berechnen:

- quantiles = column.quantile([0.25, 0.5, 0.75])
- iqr = quantiles[0.75] - quantiles[0.25]

DataCraft

Umgang mit Ausreißern

Umgang mit Ausreißern

- Als erste Frage ist dabei immer entscheidend: Welchen Ausreißertyp habe ich in meinen Daten?
  - „Echte Ausreißer“ -> Möglichst behalten und damit umgehen, zum Beispiel durch Variablentransformation
  - „Falsche Ausreißer“ -> Möglichst korrigieren oder aus den Daten entfernen

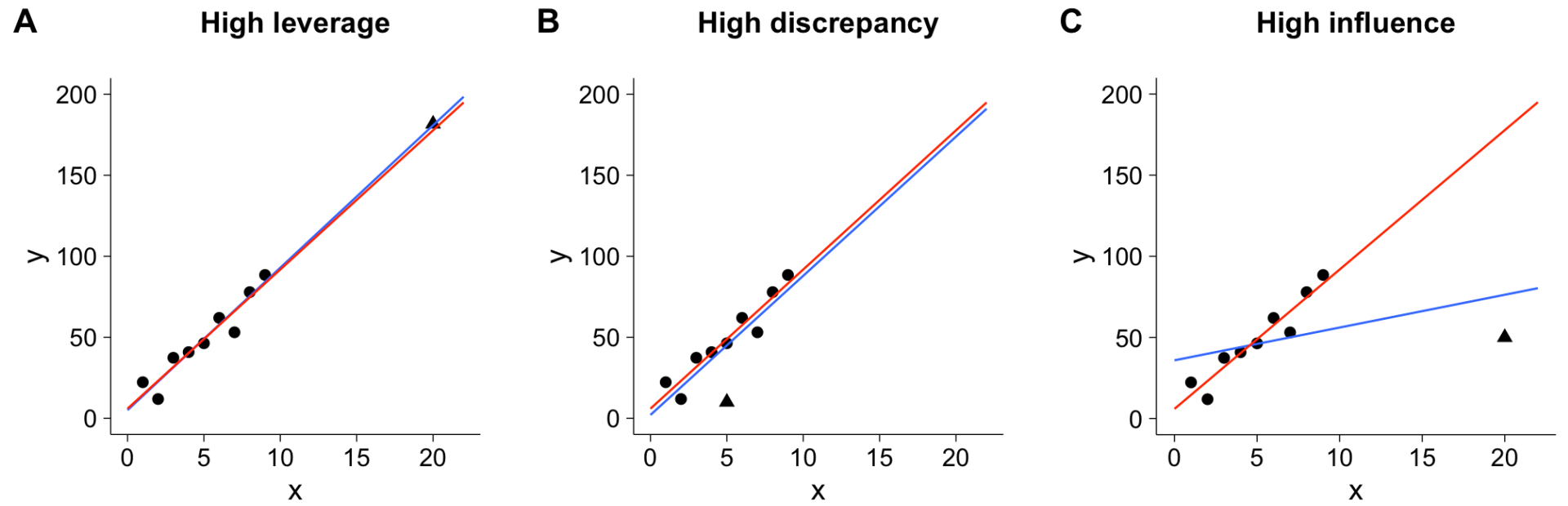
DataCraft

Hebelwirkung / Leverage von Ausreißern

- Um zu überprüfen, ob „echte Ausreißer“ nicht einfach direkt in den Daten verbleiben können, muss die Hebelwirkung geprüft werden
- Daher muss geprüft werden: Hat ein Beobachtungspunkt eine Auswirkung auf zum Beispiel die erstellte Regressionsgleichung / die berechnete Teststatistik?

DataCraft

Hebelwirkung / Leverage von Ausreißern in der grafischen Prüfung



Bilderquelle: https://tillbe.github.io/outlier-influence-identification.html

Hebelwirkung / Leverage von Ausreißern in der grafischen Prüfung in Python

Pakete laden:

- import pandas as pd
- import matplotlib.pyplot as plt

Spalten extrahieren:

- x = df['x_column']
- y = df['y_column‘]

Scatterplot erstellen:

- plt.scatter(x, y)
- plt.title('Scatter Plot of x_column vs y_column')
- plt.xlabel('x_column‘)
- plt.ylabel('y_column')
- plt.show()

DataCraft

Hebelwirkung / Leverage von Ausreißern in der statistischen Prüfung in Python

Pakete laden:

- import pandas as pd
- import statsmodels.api as sm

Abhängig (Y) und Unabhängig (X) deklarieren:

- X = df[['independent_var1', 'independent_var2']]
- y = df['dependent_var‘]

Modell erstellen und Einfluss jeder Beobachtung berechnen:

- model = sm.OLS(y, sm.add_constant(X)).fit() # Fit the regression model
- influence = model.get_influence().hat_matrix_diag
- print(leverage)

Kompromiss zwischen Behalten und Entfernen

Was ist, wenn Daten „echte“ Ausreißer sind, diese unbedingt in den Daten gehalten werden sollen, jedoch im jetzigen Zustand nicht berechenbar sind?

- Transformation der ganzen Variable in Gruppen / Ränge statt der originalen Werte
- Anpassung der Extrempunkte auf nicht extreme Werte

Ränge

Person	Sortiertes Einkommen	Rang
4	110000	1
6	58000	2
7	56000	3
1	55000	4
2	52000	5
3	50000	6
5	0	7

- Durch erstellte Ränge werden Ausreißer direkt entfernt und eine viel gröbere Auswertung der Daten ist direkt möglich
- In Python:

```
df['rank_column'] =
df['column_name'].rank()
```

Winsorisierung

Person	Sortiertes Einkommen	Winsorisierte Variable
4	110000	78800
6	58000	58000
7	56000	56000
1	55000	55000
2	52000	52000
3	50000	50000
5	0	30000

- Durch Winsorisierung wird um den Mittelwert der Variable ein 90%-Konfidenzintervall gebildet. Alle Werte außerhalb dieses Intervalls werden durch die Intervallgrenzen ersetzt
- In Python:

```
from scipy.stats import mstats

winsorized_column = mstats.winsorize(column, limits=(0.05, 0.95))
```