Fahrradvermietungen

Prognosemodell

Marcel Albers, Koen Loogman, Jacques Peluso und Steffen Seegler

2024-08-25

Inhaltsverzeichnis

1 Einleitung

1.1 Aufgabenstellung

Für die Vermietung von Fahrrädern sind verschiedene Aspekte ausschlaggebend. In dem Fall des vorliegenden Szenarios sind dies die potentiell erklärenden Variablen Einführungsphase, Jahreszeit, Wetter, Arbeitstag, Temperatur, Windgeschwindigkeit und Luftfeuchtigkeit. Die Ausprägung dieser wurden von früheren Vermietungen in einem Trainingsdatensatz festgehalten. Um künftige Fahrradvermietungen vorhersagen zu können, müssen die Zusammenhänge zwischen den genannten Variablen ermittelt und in einem Modell festgehalten werden. Ein zweiter Datensatz, der Anwendungsdatensatz, dient als Grundlage für den Performancetest.

1.2 Zielsetzung

Ziel der Arbeit ist es, ein Modell zu entwickeln, welches die Zielvariable $\hat{y}_n$ für die gegeben Anwendungsdaten x_n voraussagt. Die Zielvariable beschreibt hierbei die Anzahl an täglichen Fahrradvermietungen. Basis der Modellierung bilden die Trainingsdaten x, y. Die Bewertung von diesem Wert findet mithilfe des mittleren absoluten Fehlers - MAE (mean absolute error) statt. Der MAETest ist möglichst klein zu halten. Wie häufig wird eine Fahrradvermietung täglich stattfinden?

1.3 Aufbau der Arbeit

Das nachfolgende Kapitel *Explorative Datenanalyse und Datenverarbeitung* befasst sich mit der detaillierten Beschreibung des vorliegenden Datensatzes und der Analyse von diesem. Hierbei werden die Zusammenhänge von verschiedenen Konstellationen der Attribute betrachtet. Des Weiteren findet eine Datenaufbereitung statt, die für die spätere Verarbeitung der Daten in den Modellen notwendig ist. Im darauf folgenden Kapitel, der *Methodenbeschreibung*, wird die Auswahl der Modelle und die Funktionsweise dieser erläutert. Nach dem Training des Modells, wird dieses angewendet, die Ergebnisse aufgeführt, betrachtet und eine Vorhersage getroffen. Die *Zusammenfassung* der Arbeit stellt die Ergebnisse dar und führt diese als abschließendes Kapitel zusammen.

2 Explorative Datenanalyse und Datenvorverarbeitung

Einlesen der relevanten Datensätze, wobei die entsprechenden CSV-Dateien aus dem festgelegten Verzeichnis data/raw geladen werden.

```r
# Load data
raw.application <- here("data", "raw", "anwendung.csv") |> read.csv2()
raw.dataset <- here("data", "raw", "train.csv") |> read.csv2()

# Show the structure of the data
str(raw.application)
```

```
'data.frame':   200 obs. of  7 variables:
 $ einfuehrungsphase  : chr  "Nein" "Ja" "Nein" "Ja" ...
 $ jahreszeit         : chr  "Frühling" "Sommer" "Sommer" "Herbst" ...
 $ wetter             : chr  "Gut" "Gut" "Nicht gut" "Nicht gut" ...
 $ arbeitstag         : chr  "Nein" "Nein" "Ja" "Ja" ...
 $ temperatur         : num  5.39 24.9 25.51 15.3 16.72 ...
 $ windgeschwindigkeit: num  15.57 13.55 6.43 12.56 9.64 ...
 $ luftfeuchtigkeit   : num  46.1 55.6 60.3 83.2 52.6 ...
```

```r
str(raw.dataset)
```

```
'data.frame':   500 obs. of  8 variables:
 $ einfuehrungsphase  : chr  "Nein" "Nein" "Nein" "Nein" ...
 $ jahreszeit         : chr  "Sommer" "Sommer" "Winter" "Frühling" ...
 $ wetter             : chr  "Nicht gut" "Gut" "Gut" "Gut" ...
 $ arbeitstag         : chr  "Ja" "Ja" "Ja" "Ja" ...
 $ temperatur         : num  23.7 26.5 4.6 12.9 23.8 ...
 $ windgeschwindigkeit: num  15.2 10.2 10.2 25.8 11.9 ...
 $ luftfeuchtigkeit   : num  73.6 44 58.5 50 82.3 ...
 $ vermietungen       : int  4127 8173 3783 5557 3350 4709 4609 5117 1851 2494 ...
```

Der vorliegende Datensatz ist in zwei Teildatenmengen ohne fehlende Werte aufgeteilt. Diese umfassen meteorologische und temporale Informationen, die in einem strukturierten Format bereitgestellt wurden. Die Gesamtdatenmenge bildet sich aus aus den Anwendungsdaten, bestehend aus 200 Beobachtungen und 7 Variablen und den Trainingsdaten, bestehend aus 500 Beobachtungen und 8 Variablen. Die ersten 7 Variablen der Datensätze sind formal deckungsgleich. Die zusätzliche Variable des Trainingsdatensatzes enthält die Zielvariable, die durch die Modellierung angenähert wird.

2.1 Variablenübersicht

- *Einführungsphase*: Nominalskalierte, kategorische Variable mit den Ausprägungen “Ja” und “Nein”.
- *Jahreszeit*: Ordinalskalierte, kategorische Variable mit vier Ausprägungen Frühling, Sommer, Herbst und Winter.

- *Wetter*: Ordinalskalierte, kategorische Variable mit den Hauptausprägungen “Gut” und “Nicht gut”, sowie “Schlecht”.
- *Arbeitstag*: Nominalskalierte, kategorische Variable mit den Ausprägungen “Ja” und “Nein”.
- *Temperatur*: Intervallskallierte, stetig, metrische Variable, die die Temperatur in Grad Celsius angibt.
- *Windgeschwindigkeit*: Verhältnisskalierte, stetig, metrische Variable, die die Windgeschwindigkeit in km/h misst.
- *Luftfeuchtigkeit*: Verhältnisskalierte, stetig, metrische Variable, die die relative Luftfeuchtigkeit in Prozent angibt.
- *Vermietungen*: Verhältnisskalierte, diskret, metrische Variable und gibt die Anzahl der Vermietungen als Ganzzahl an.

2.2 Vermietungen

```
stats.vermietungen <- fav_stats(raw.dataset$vermietungen)
stats.vermietungen
```

```
 min      Q1 median      Q3  max     mean       sd   n missing
  23 2825.25 4514.5 5871.75 8715 4416.606 1989.573 500       0
```

```
gf_dhistogram(~ vermietungen, data=raw.dataset) |>
  gf_dens(linewidth = 1, color = "blue")
```

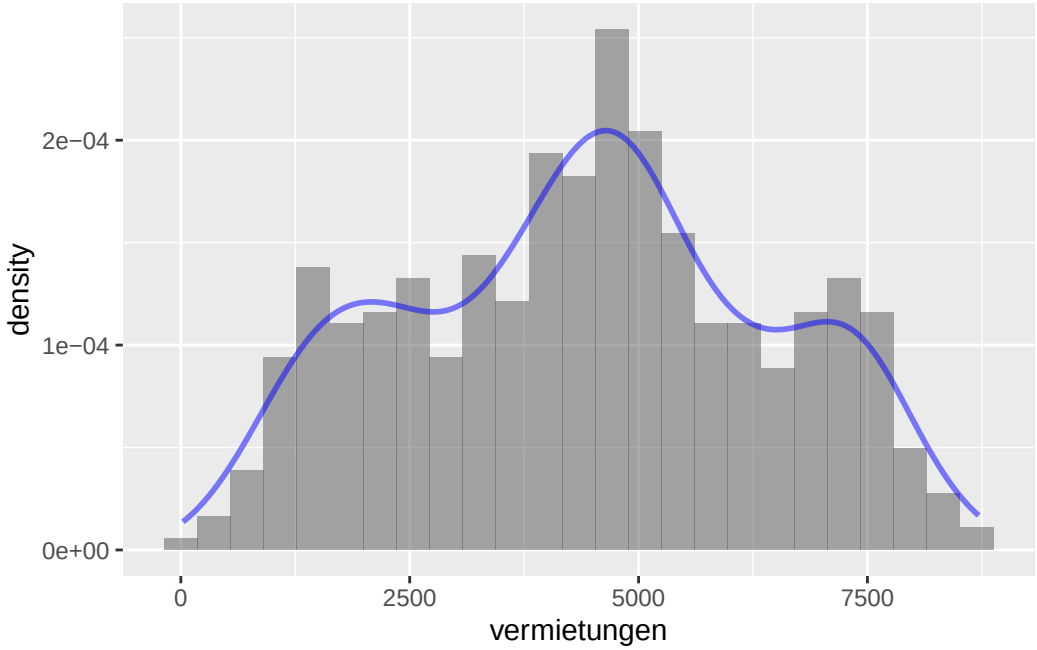


Abbildung 1: Häufigkeitsverteilung der Vermietungen

Die Analyse der Häufigkeitsverteilung der Vermietungen in Abbildung 1 zeigt eine Verteilung mit einem Hauptpeak und zwei kleineren Peaks, die links und rechts von diesem liegen. Diese Verteilung weist eine unimodale Struktur auf, wobei die kleineren Peaks auf Variationen innerhalb der Daten hinweisen. Dieses Muster kann durch die unterschiedlichen Einflüsse der Variablen im Datensatz verursacht werden, welche die Vermietungszahlen beeinflussen. Diese werden in dem folgenden Absatz näher betrachtet.

2.3 Einfuehrungsphase

```
counts.einfuehrungsphase <- tally(~ einfuehrungsphase, data=raw.dataset)
gf_bar(~ einfuehrungsphase, data=raw.dataset)
```

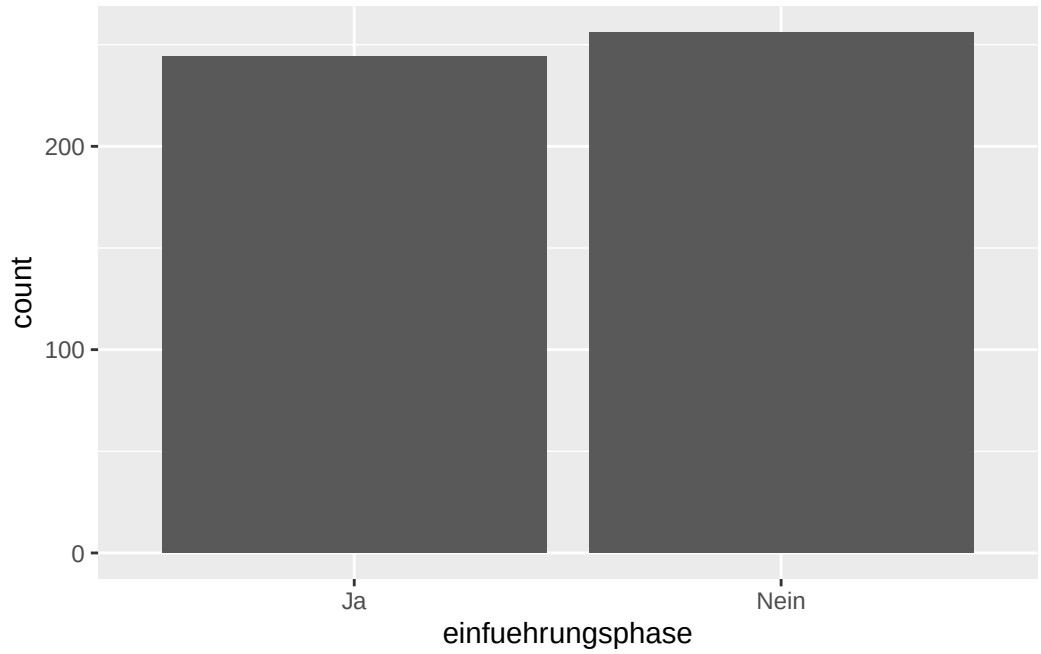


Abbildung 2: Häufigkeitsverteilung der Einfuehrungsphase

Der erste zu untersuchende Einflussfaktor ist die kategoriale Variable “Einführungsphase” und deren Verteilung. Beide Kategorien “Ja” und “Nein” haben eine ähnliche Anzahl von Einträgen (Ja=244, Nein=256). Insgesamt zeigt die Verteilung in Abbildung 2, dass es keinen signifikanten Unterschied in der Anzahl der Einträge zwischen den beiden Kategorien gibt.

```
gf_boxplot(vermietungen ~ einfuehrungsphase, data=raw.dataset)
```

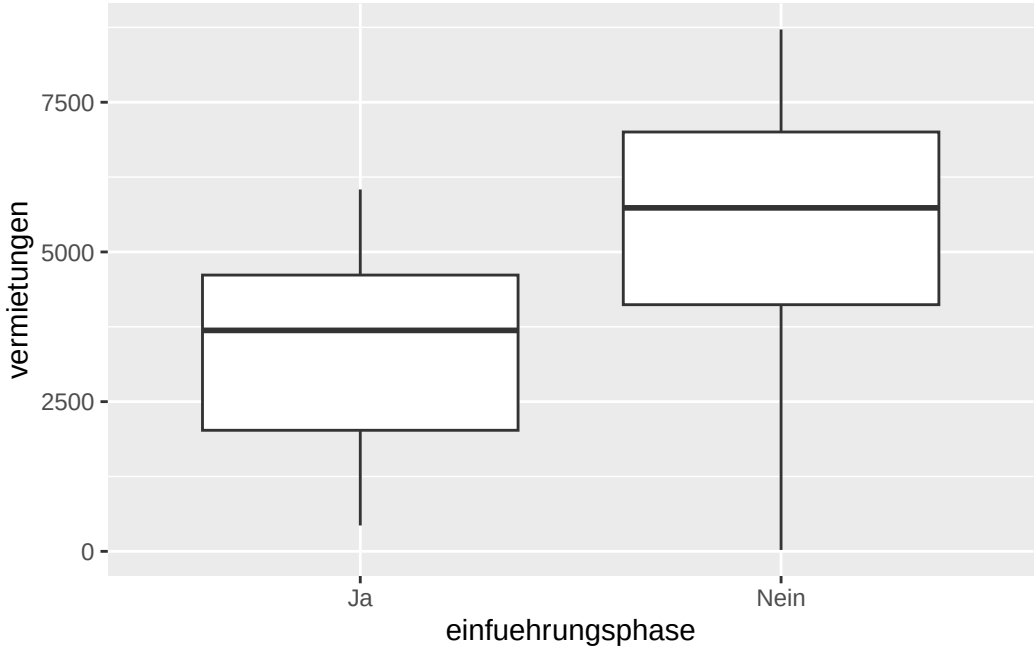


Abbildung 3: Vermietungen in Abhängigkeit zur Einfuehrungsphase

Der Boxplot in Abbildung 3 veranschaulicht die Verteilung der Vermietungen im Kontext der Einführungsphase und zeigt mehrere wesentliche Aspekte auf. Der Median der Vermietungen ist in der Gruppe ohne Einführungsphase („Nein“) höher als in der Gruppe mit Einführungsphase („Ja“). Dies weist auf eine größere zentrale Tendenz der Vermietungen hin, wenn keine Einführungsphase vorliegt. Zudem ist der Interquartilsabstand (IQR) in der Gruppe ohne Einführungsphase ebenfalls größer, was auf eine erhöhte Streuung der Daten in dieser Gruppe hindeutet. Beide Gruppen zeigen keine auffälligen Ausreißer, da alle Datenpunkte innerhalb der Whisker liegen. Insgesamt sind die Vermietungen in der Gruppe ohne Einführungsphase höher, was durch die Position der Box und der Whisker im Boxplot deutlich wird.

2.4 Jahreszeit

```
counts.jahreszeit <- tally(~ jahreszeit, data=raw.dataset)
gf_bar(~ jahreszeit, data=raw.dataset)
```

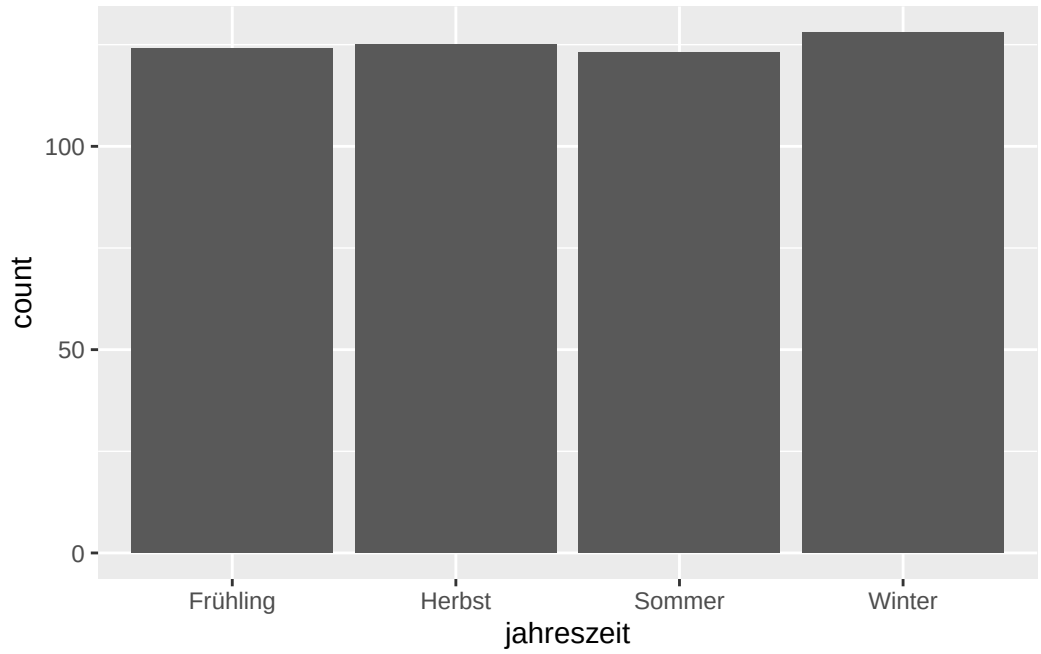


Abbildung 4: Häufigkeitsverteilung der Jahreszeit

Die Häufigkeit der Vorkommen über die verschiedenen Jahreszeiten ist hinweg weitgehend gleichmäßig verteilt, wie in Abbildung 4 zu sehen ist. Dies bedeutet, dass es keine signifikanten Unterschiede oder ausgeprägte Schwankungen in der Anzahl der Vorkommen zu verschiedenen Zeiten des Jahres gibt.

```
gf_boxplot(vermietungen ~ jahreszeit, data=raw.dataset)
```

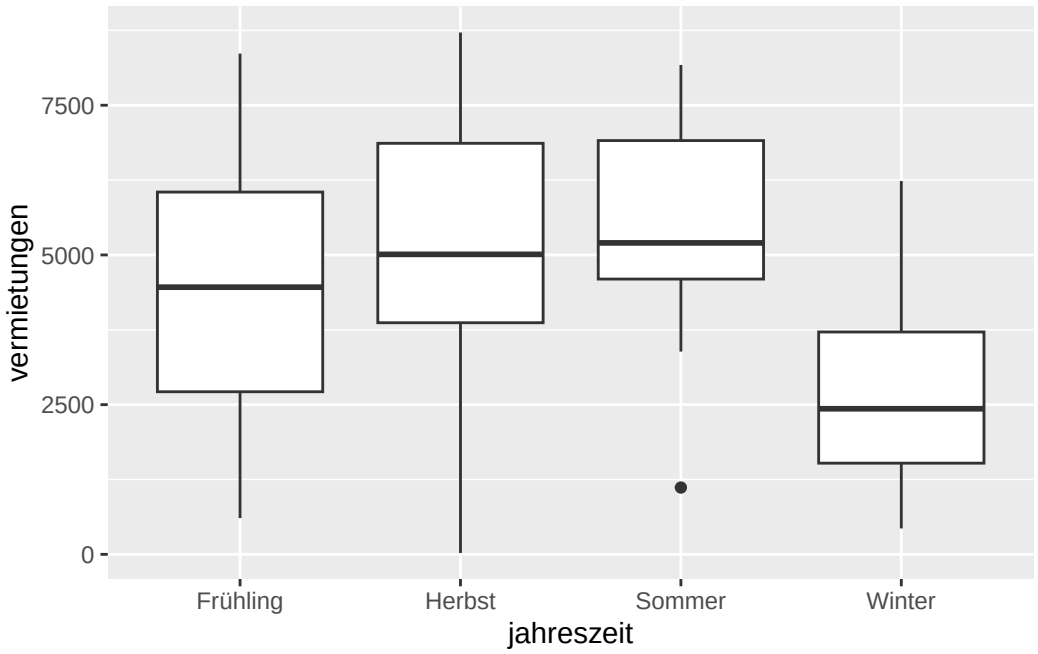


Abbildung 5: Vermietungen in Abhängigkeit zur Jahreszeit

Der Boxplot in Abbildung 5 zeigt, dass im Frühling und Herbst die Verteilungen der Vermietungen ähnlich sind, wobei beide Jahreszeiten relativ große Interquartilsabstände aufweisen. Dies deutet auf eine breite Streuung der Daten in diesen Perioden hin. Die Medianwerte sind in beiden Fällen hoch, sodass die Vermietungsaktivität in diesen Jahreszeiten stark vertreten ist. Im Sommer bleibt die Verteilung ähnlich, aber es gibt einen Anstieg im Medianwert, was darauf hinweist, dass die Vermietungen in dieser Jahreszeit tendenziell etwas höher sind. Der Winter hingegen zeigt eine deutliche Verringerung der Vermietungszahlen, wie durch den niedrigeren Median und den geringeren Interquartilsabstand ersichtlich ist. Die geringere Variabilität im Winter deutet darauf hin, dass die Vermietungsaktivitäten in dieser Jahreszeit relativ konstant, aber auf einem niedrigen Niveau sind.

Der Boxplot in Abbildung 5 zeigt auch, dass es für Sommer einen auffälligen Eintrag mit einer Anzahl Vermietungen von unter 2500 gibt, der potenziell als Ausreiser gilt.

2.5 Wetter

```
counts.wetter <- tally(~ wetter, data=raw.dataset)
gf_bar(~ wetter, data=raw.dataset)
```

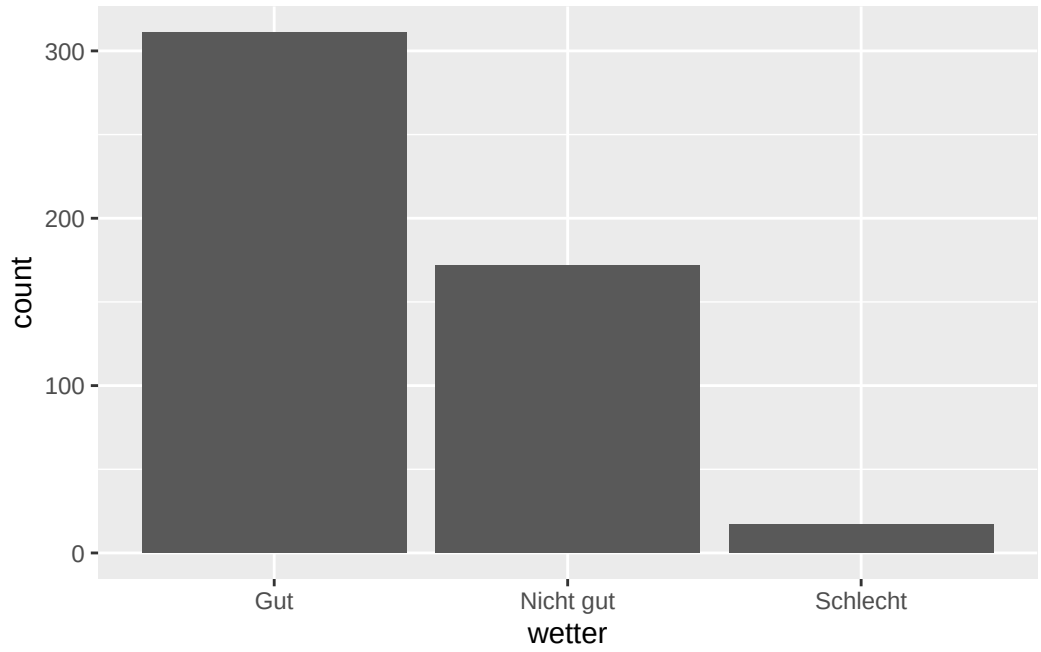


Abbildung 6: Häufigkeitsverteilung vom Wetter

Das Balkendiagramm in Abbildung 6 illustriert die Häufigkeit der Wetterkategorien “Gut”, “Nicht gut” und “Schlecht”. Die Kategorie “Gut” weist mit 311 die höchste Anzahl von Beobachtungen auf. Die Kategorie “Nicht gut” liegt mit 172 in der Mitte, mit etwa der Hälfte der Zählungen von “Gut”, was zeigt, dass solche Wetterbedingungen weniger häufig sind, aber dennoch häufiger als “Schlecht” vorkommen. Die Kategorie “Schlecht” verzeichnet mit 17 die geringste Anzahl von Beobachtungen und tritt somit am seltensten auf.

Dies zeigt eine deutliche Dominanz guten Wetters, während schlechtes Wetter selten ist. “Nicht gut” liegt in einer mittleren Häufigkeitsposition, was die Variabilität der Wetterbedingungen verdeutlicht, jedoch mit einer klaren Präferenz für gutes Wetter.

```
gf_boxplot(vermietungen ~ wetter, data=raw.dataset)
```

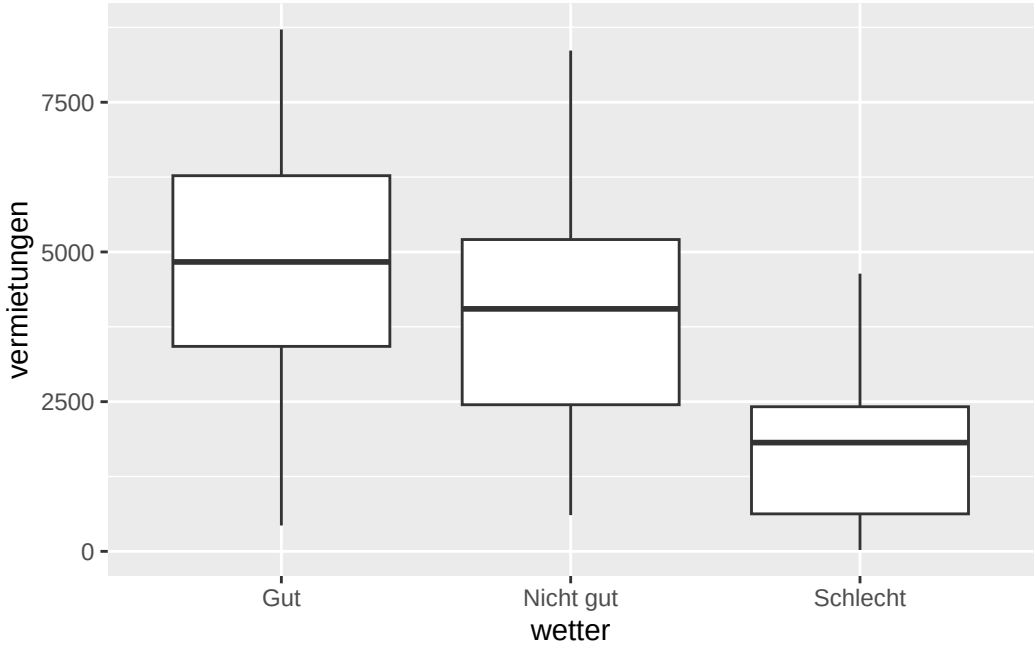


Abbildung 7: Vermietungen in Abhängigkeit zum Wetter

Die Boxplots in Abbildung 7 zeigen, dass die Vermietungszahlen bei gutem Wetter am höchsten sind, mit einem Median knapp unter 5000 und einer breiten Streuung. Bei "Nicht gut" liegt der Median etwas niedriger, aber ähnlich, mit geringerer Streuung. Die Vermietungszahlen sind bei schlechtem Wetter deutlich niedriger, mit einem Median unter 2500 und einer kleineren Streuung. Gutes Wetter führt demnach zu den höchsten und variabelsten Vermietungszahlen, während schlechtes Wetter mit deutlich geringerer Nachfrage verbunden ist.

2.6 Arbeitstag

```
counts.arbeitstag <- tally(~ arbeitstag, data=raw.dataset)
gf_bar(~ arbeitstag, data=raw.dataset)
```

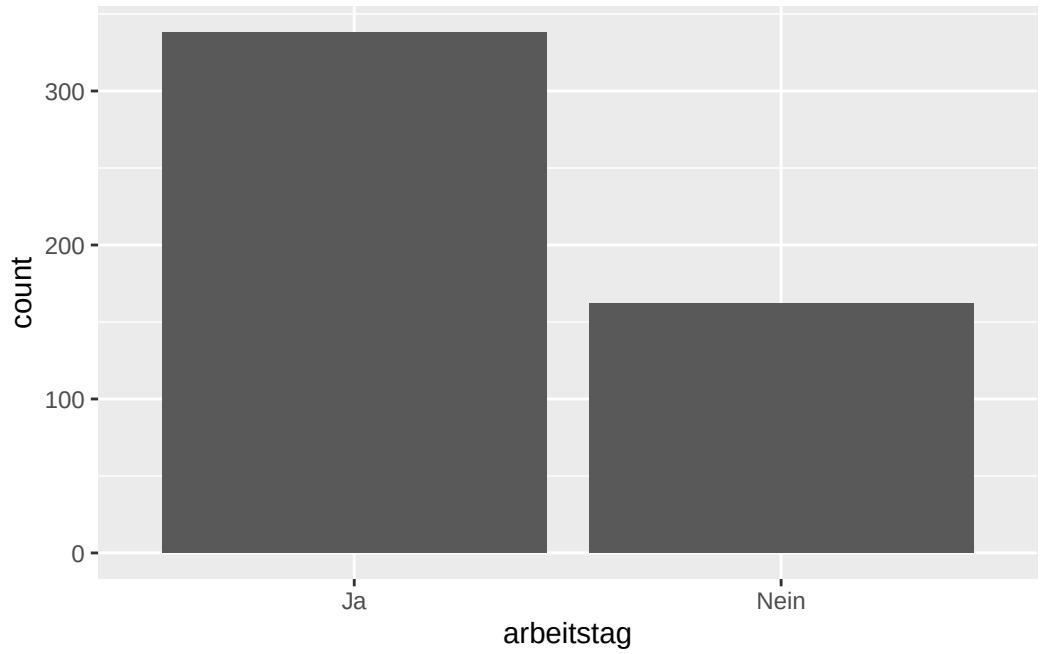


Abbildung 8: Häufigkeitsverteilung vom Arbeitstag

Das Diagramm in Abbildung 8 zeigt, dass die meisten Beobachtungen (338) an Arbeitstagen gesammelt wurden, was darauf hindeutet, dass diese Tage in der zugrunde liegenden Datensatz häufiger beobachtet wurden als Nicht-Arbeitstage. Vermietungen werden demnach häufiger an Arbeitstagen als an Nicht-Arbeitstagen wie Wochenenden, Feiertagen oder in den Ferien durchgeführt.

```
gf_boxplot(vermietungen ~ arbeitstag, data=raw.dataset)
```

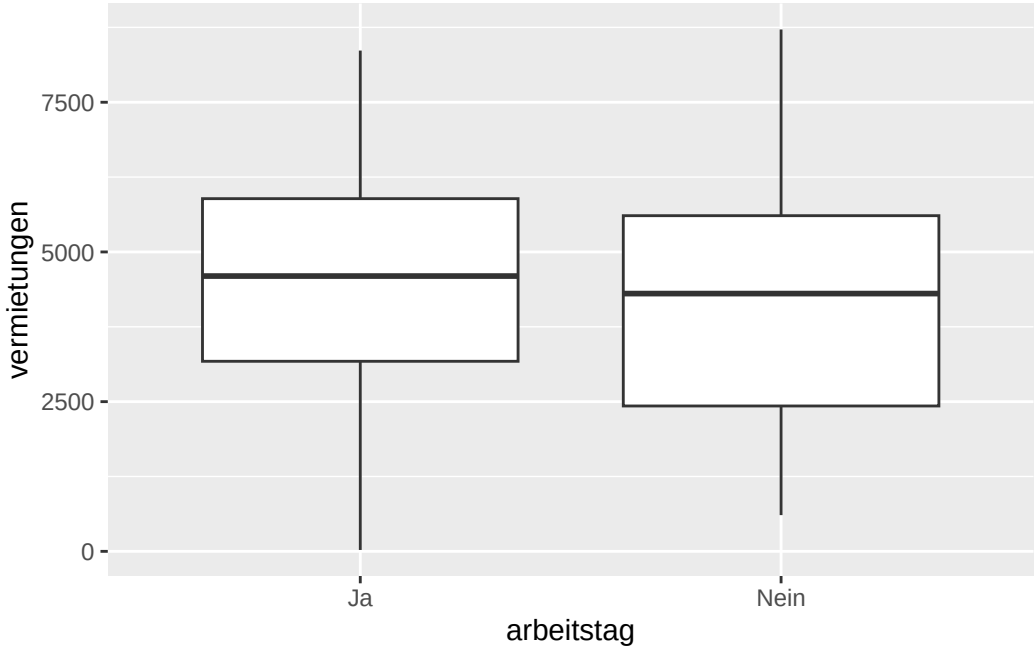


Abbildung 9: Vermietungen in Abhängigkeit zum Arbeitstag

Der Boxplot in Abbildung 9 zeigt, dass der Median der Vermietungen ist an Arbeitstagen etwas niedriger als an Nicht-Arbeitstagen. Der Interquartilsabstand (IQR) ist in beiden Fällen ähnlich, was darauf hindeutet, dass die Streuung der mittleren 50% der Daten an beiden Tagen vergleichbar ist. Die Whisker erstrecken sich ebenfalls in ähnlicher Weise, was auf eine vergleichbare Spannweite der Daten hinweist. Die Verteilungen sind relativ symmetrisch, da die Mediane etwa in der Mitte der Boxen liegen. Zudem gibt es keine offensichtlichen Ausreißer, da keine Punkte außerhalb der Whisker zu sehen sind. Schlussfolgernd lässt sich dazu sagen, dass die Vermietungen an Arbeitstagen und Nicht-Arbeitstagen ähnliche Verteilungen aufweisen, wobei die Medianwerte an Nicht-Arbeitstagen geringfügig höher sind.

2.7 Temperatur

```
stats.temperatur <- fav_stats(raw.dataset$temperatur)
stats.temperatur
```

```
   min     Q1 median    Q3   max     mean       sd   n missing
 -3.59 7.4825  15.21 23.02 33.83 15.03484 8.778967 500       0
```

```
gf_dhistogram(~ temperatur, data=raw.dataset) |>
  gf_dens(linewidth = 1, color = "blue")
```

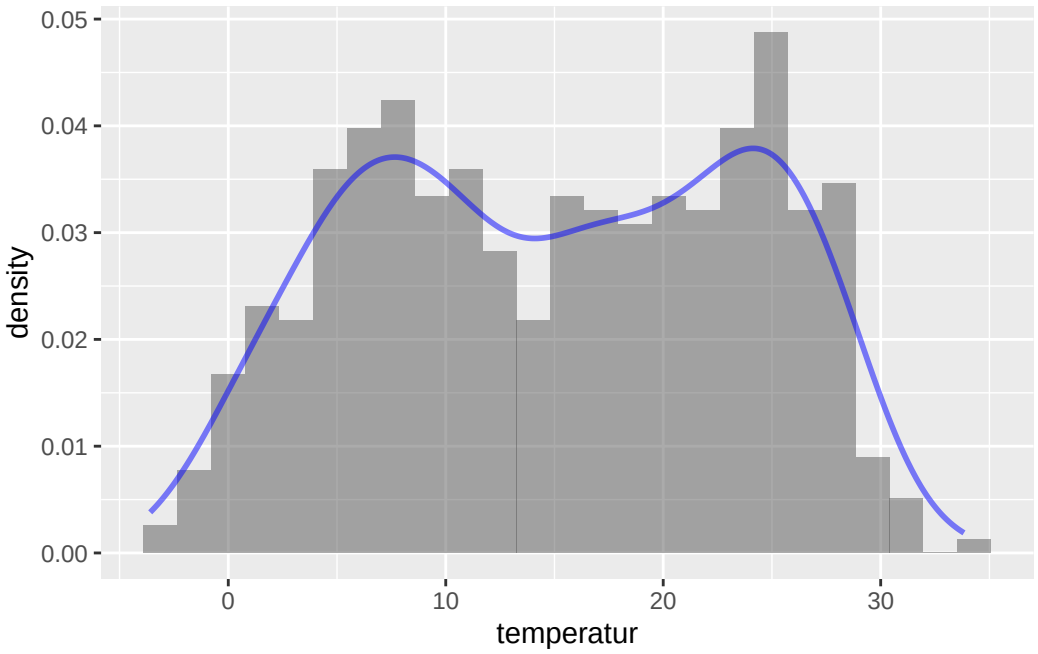


Abbildung 10: Häufigkeitsverteilung der Temperatur

Das Histogramm in Abbildung 10 zeigt, dass die Temperatur eine bimodale Verteilung der Daten hat, mit zwei Häufigkeitsspitzen bei etwa 8°C und 24°C. Dies deutet darauf hin, dass diese Temperaturbereiche am häufigsten auftreten. Die Dichte nimmt an den Rändern, insbesondere bei Temperaturen unter 5°C und über 24°C, deutlich ab, was auf seltenere extreme Temperaturen hinweist. Insgesamt zeigt die Verteilung eine Konzentration der Temperaturen in zwei unterschiedlichen Bereichen, was saisonalen Schwankungen entspricht.

```
gf_point(vermietungen ~ temperatur, data=raw.dataset) |>
  gf_smooth(method = "loess")
```

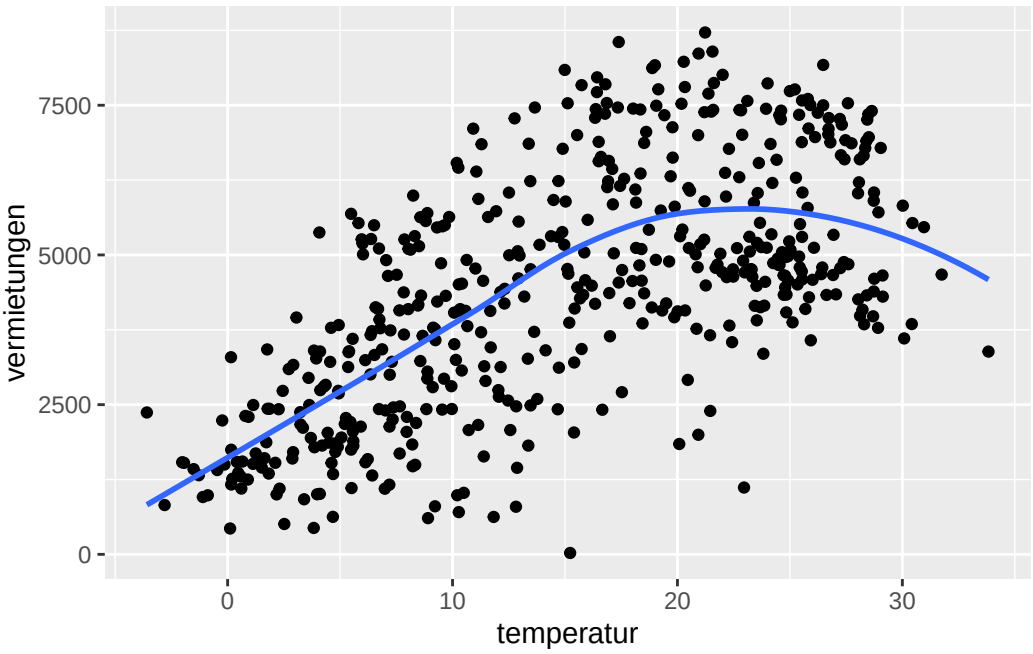


Abbildung 11: Vermietungen in Abhängigkeit zur Temperatur

Die Abhängigkeit zwischen der Temperatur und der Anzahl der Vermietungen in Abbildung 11 zeigt einen anfänglichen Anstieg der Vermietungszahlen mit zunehmender Temperatur, wobei ein Maximum bei etwa 20°- 25°C erreicht wird. Jenseits dieses Temperaturwertes nimmt die Anzahl der Vermietungen trotz weiter steigender Temperaturen ab. Dies legt nahe, dass gemäßigte Temperaturen um 20°C optimal für Vermietungen sind, während höhere Temperaturen tendenziell eine negative Auswirkung auf die Nachfrage haben könnten.

2.8 Windgeschwindigkeit

```
stats.windgeschwindigkeit <- fav_stats(raw.dataset$windgeschwindigkeit)
stats.windgeschwindigkeit
```

```
  min     Q1 median      Q3   max     mean       sd   n missing
 1.49 8.9575  11.91 15.6925 34.41 12.86882 5.358726 500       0
```

```
gf_dhistogram(~ windgeschwindigkeit, data=raw.dataset) |>
  gf_dens(linewidth = 1, color = "blue")
```

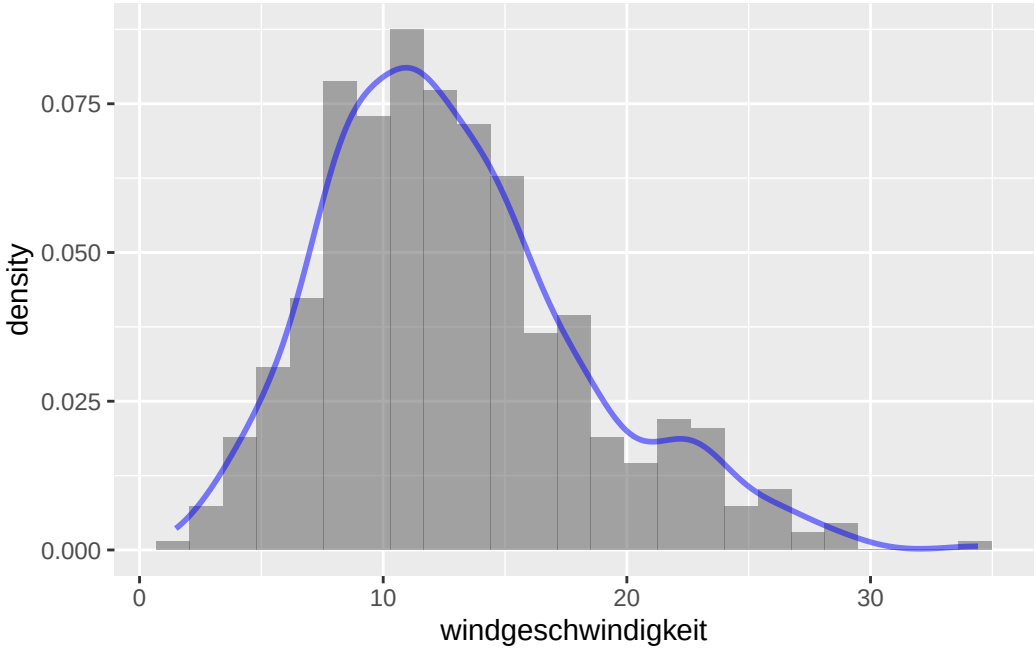


Abbildung 12: Häufigkeitsverteilung der Windgeschwindigkeit

Die Verteilung der Windgeschwindigkeiten in Abbildung 12 weist einen deutlichen Gipfel bei etwa 11 km/h auf und stellt damit die häufigste Windgeschwindigkeit in den Daten dar. Abseits dieser Wertes nimmt die Dichte der Verteilung ab. Höhere Windgeschwindigkeiten treten seltener auf. Die Verteilung ist leicht rechtsschief, was darauf hindeutet, dass extrem hohe Windgeschwindigkeiten weniger häufig sind, aber dennoch vorkommen.

```
gf_point(vermietungen ~ windgeschwindigkeit, data=raw.dataset) |>
  gf_smooth(method = "loess")
```

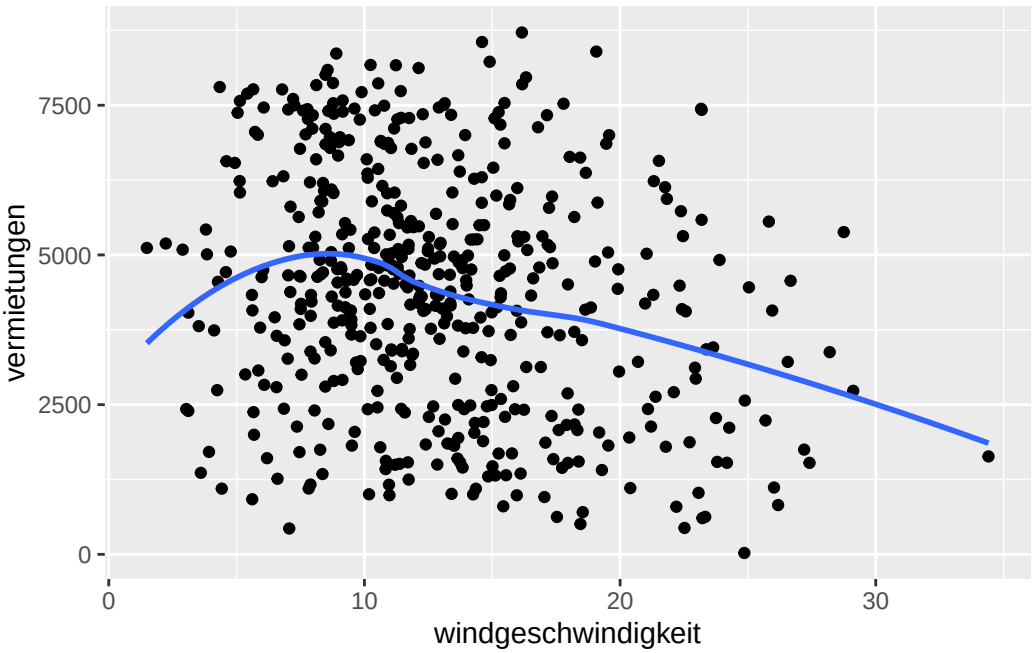


Abbildung 13: Vermietungen in Abhängigkeit zur Windgeschwindigkeit

Die Glättungslinie im Punktdiagramm zur Abhängigkeit der Vermietungen zur Windgeschwindigkeit in Abbildung 13 verdeutlicht den Trend der Daten und zeigt, dass die Anzahl der Vermietungen zunächst mit zunehmender Windgeschwindigkeit steigt. Ab einem bestimmten Punkt beginnt sie jedoch wieder zu sinken.

2.9 Luftfeuchtigkeit

```
stats.luftfeuchtigkeit <- fav_stats(raw.dataset$luftfeuchtigkeit)
stats.luftfeuchtigkeit
```

```
  min     Q1 median     Q3    max     mean       sd   n missing
 3.88 50.935 62.345 74.015 103.61 62.38542 15.46771 500       0
```

```
gf_dhistogram(~ luftfeuchtigkeit, data=raw.dataset) |>
  gf_dens(linewidth = 1, color = "blue")
```

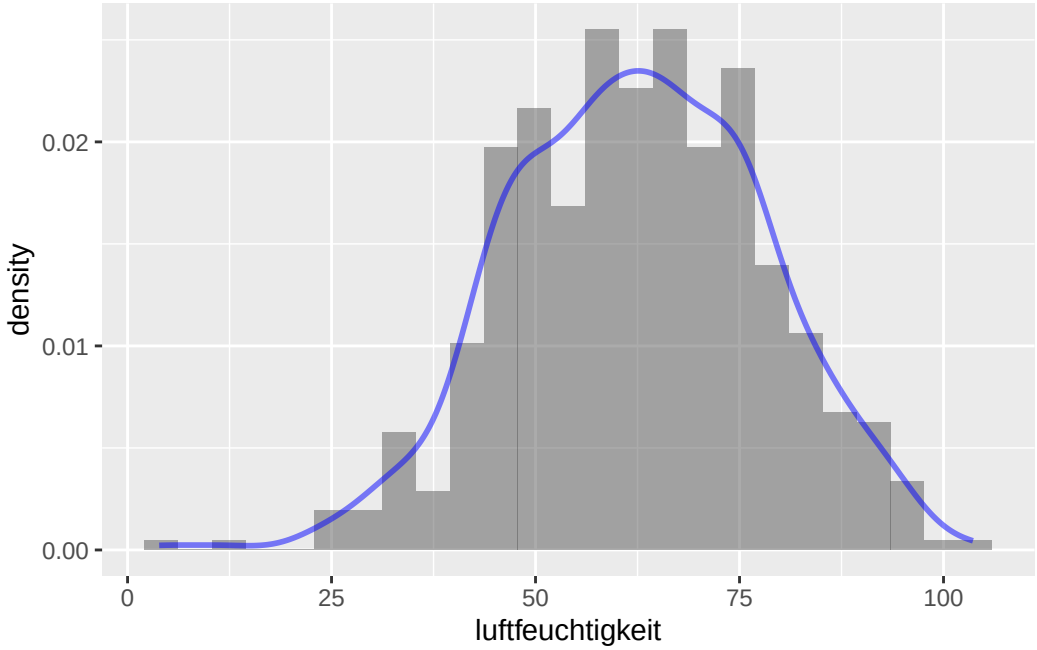


Abbildung 14: Häufigkeitsverteilung der Luftfeuchtigkeit

Das Histogramm in Abbildung 14 zeigt die Verteilung der Luftfeuchtigkeitsdaten in %, mit einem Minimum von 3,88 und einem Maximum von 103,61. Das Maximum von 103,61 scheint hierbei ein fehlerhafter Wert zu sein, da Luftfeuchtigkeitswerte ein Höchstwert von 100 erreichen können. Der Mittelwert und der Median liegen nahe bei 62,39 bzw. 62,35. Im Gesamten ergibt sich hierbei eine Normalverteilung mit einer moderaten Streuung um den Mittelwert.

```
gf_point(vermietungen ~ luftfeuchtigkeit, data=raw.dataset) |>
  gf_smooth(method = "loess")
```

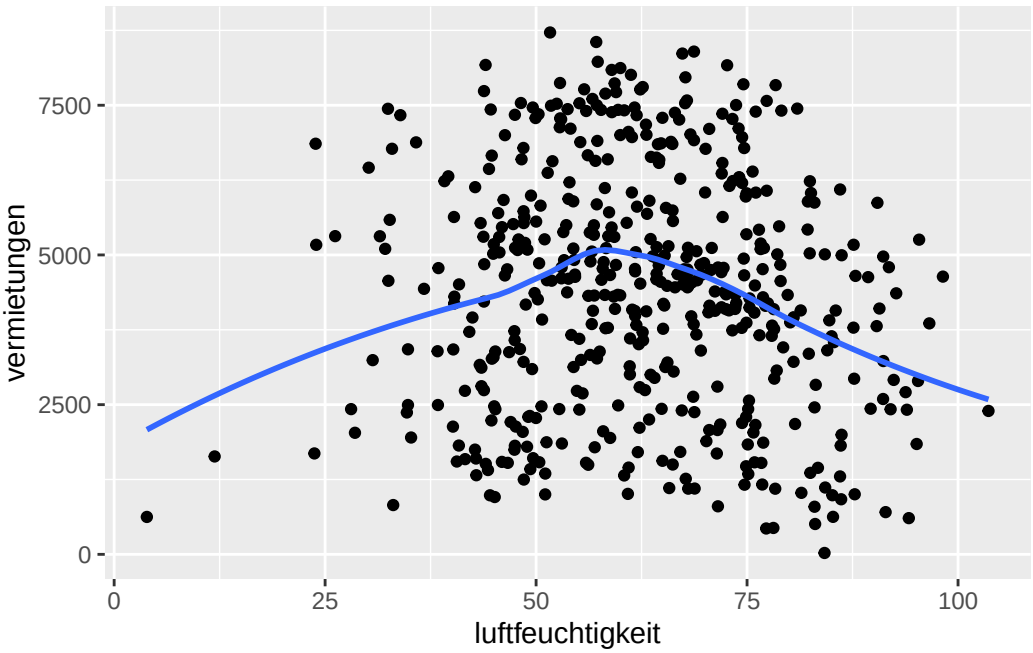


Abbildung 15: Vermietungen in Abhängigkeit zur Luftfeuchtigkeit

Der höchste Punkt der Trendlinie im Punktdiagramm in Abbildung 15 befindet sich ungefähr im Bereich einer Luftfeuchtigkeit von 50 bis 60, was darauf hindeutet, dass dieser Bereich optimal für die meisten Vermietungen ist.

2.10 Entfernen von Ausreißern in den Daten

In Kapitel 2.9 und Kapitel 2.4 wurde festgestellt, dass der Eintrag mit einer Luftfeuchtigkeit von 103.61% fehlerhaft ist und ein Eintrag für die Jahreszeit Sommer mit einer Anzahl Vermietungen von unter 2500 potenziell als Ausreiser gilt. Diese Einträge werden im folgenden aus den Daten entfernt.

```
# Remove wrong data
clean.raw.dataset <- raw.dataset[raw.dataset$luftfeuchtigkeit <= 100, ]
clean.raw.dataset <- clean.raw.dataset[
  !(
    clean.raw.dataset$jahreszeit == "Sommer" &
    clean.raw.dataset$vermietungen <= 2500
  ),
]

# Update affected stats / counts
stats.luftfeuchtigkeit <- fav_stats(clean.raw.dataset$luftfeuchtigkeit)
counts.jahreszeit <- tally(~ jahreszeit, data=clean.raw.dataset)
```

2.11 Preprocessing der Daten

Quelle: https://www.geeksforgeeks.org/data-preprocessing-in-r/

Folgend wird die Aufbereitung des Datensatz systematisch in mehreren Schritten durchgeführt. Diese sind notwendig für die Optimierung der Daten, um anschließend das neuronale Netz zu traineren.

Zunächst wird eine Kopie des ursprünglichen Datensatzes erstellt. Dies dient dem Zweck, die Originaldaten unverändert zu lassen und eine nicht-destruktive Bearbeitung zu ermöglichen. Diese Vorgehensweise stellt sicher, dass die Integrität der Ausgangsdaten während der Verarbeitung erhalten bleibt und bei Bedarf auf die Originaldaten zurückgegriffen werden kann.

Im nächsten Schritt erfolgt die binäre Kodierung der Variablen „einfuehrungsphase“ und „arbeitstag“. Hierzu wird ein Mapping von „Nein“ zu 0 und „Ja“ zu 1 definiert.

Anschließend wird die Variable „wetter“ ordinal kodiert. Die Werte „Schlecht“, „Nicht gut“ und „Gut“ werden in die numerischen Werte 0, 1 und 2 überführt. Diese Kodierung reflektiert die natürliche Reihenfolge der Kategorien, um die hierarchische Beziehung zwischen den Kategorien zu berücksichtigen.

Für die Variable „jahreszeit“ wird die One-Hot-Kodierung verwendet. Mithilfe der Funktion model.matrix werden für jede Kategorie der „jahreszeit“-Variable separate binäre Spalten erstellt. Hierdruch wird ermöglicht, dass jede Kategorie der Variable in einer eigenen Spalte dargestellt wird, wobei der Wert 1 angibt, dass die Kategorie zutrifft, und 0, dass sie nicht zutrifft. Die ursprüngliche „jahreszeit“-Spalte wird nach der Kodierung entfernt, um Redundanz zu vermeiden und den Datensatz zu bereinigen.

```r
preprocess <- function(dataset) {
  # Make a copy of the data
  dataset.copy <- dataset

  # Encode einfuehrungsphase and arbeitstag (binary encoding)
  yn_mapping <- c("Nein" = 0, "Ja" = 1)
  dataset.copy$einfuehrungsphase <- yn_mapping[
    as.character(dataset$einfuehrungsphase)
  ]
  dataset.copy$arbeitstag <- yn_mapping[
    as.character(dataset$arbeitstag)
  ]

  # Encode wetter (ordinal encoding)
  wetter_mapping <- c("Schlecht" = 0, "Nicht gut" = 1, "Gut" = 2)
  dataset.copy$wetter <- wetter_mapping[
    as.character(dataset$wetter)
  ]
```

```r
  # Encode jahreszeit (one-hot encoding)
  encoded_jahreszeit <- as.data.frame(
    model.matrix(~ jahreszeit - 1, dataset)
  )

  # Add the new columns to the encoded data frame and remove the old column
  dataset.copy <- dataset.copy |>
    select(-jahreszeit)
  dataset.copy <- cbind(encoded_jahreszeit, dataset.copy)

  # Min/max scale temperatur
  dataset.copy$temperatur = (
    dataset.copy$temperatur - stats.temperatur$min
  ) / (
    stats.temperatur$max - stats.temperatur$min
  )

  # Min/max scale windgeschwindigkeit
  dataset.copy$windgeschwindigkeit = (
    dataset.copy$windgeschwindigkeit - stats.windgeschwindigkeit$min
  ) / (
    stats.windgeschwindigkeit$max - stats.windgeschwindigkeit$min
  )

  # Scale luftfeuchtigkeit
  dataset.copy$luftfeuchtigkeit <- dataset.copy$luftfeuchtigkeit / 100

  return(dataset.copy)
}
```

2.11.1 Preprocessing Ergebnisse

In der Vorverarbeitung der Ergebnisse wird zunächst ein Zufallszahlengenerator mit dem Wert 69 initialisiert, um die Reproduzierbarkeit der Ergebnisse sicherzustellen. Anschließend werden die Rohdaten sowohl für Anwendungen als auch für den Datensatz vorverarbeitet. Die vorverarbeiteten Daten werden in den Variablen *application* und *dataset* gespeichert.

Um die Auswirkungen der Vorverarbeitung zu veranschaulichen, wird der Anfang des ursprünglichen Datensatzes *raw.dataset* und des vorverarbeiteten Datensatzes dataset mit der Funktion *head()* angezeigt. Dies ermöglicht einen direkten Vergleich der Daten vor und nach der Vorverarbeitung. Der Datensatz wird folgend für das Training vorbereitet. Die Features werden aus dem vorverarbeiteten Datensatz *dataset* extrahiert, indem die Spalte vermietungen ausgeschlossen wird, während die Zielvariable *vermietungen* separat ausgewählt wird. Der Datensatz wird dann in einen Trainings- und einen Testsatz aufgeteilt. Diese Aufteilung erfolgt zufällig, wobei 80 % der Daten für das Training und 20 % für die Test verwendet

werden. Dies wird durch die Funktion *sample()* erreicht, die eine logische Vektorreihe erzeugt, die angibt, welche Zeilen in den Trainingssatz (*train.raw.dataset* und *train.dataset*) und welche in den Testsatz (*test.raw.dataset* und *test.dataset*) aufgenommen werden sollen. Schließlich werden die Features und Labels sowohl für den Trainings- als auch für den Testsatz getrennt gespeichert, indem die entsprechenden Zeilen aus den vorverarbeiteten Feature- und Labeldatensätzen ausgewählt werden.

```
# Make this example reproducible
set.seed(69)

# Preprocess the data
application <- preprocess(raw.application)
dataset <- preprocess(clean.raw.dataset)

# Data before preprocessing
head(clean.raw.dataset)
```

```
  einfuehrungsphase jahreszeit    wetter arbeitstag temperatur
1              Nein     Sommer Nicht gut         Ja      23.69
2              Nein     Sommer       Gut         Ja      26.48
3              Nein     Winter       Gut         Ja       4.60
4              Nein   Frühling       Gut         Ja      12.94
5                Ja     Herbst Nicht gut       Nein      23.82
6                Ja     Sommer Nicht gut         Ja      22.99
  windgeschwindigkeit luftfeuchtigkeit vermietungen
1               15.19            73.61         4127
2               10.24            44.02         8173
3               10.24            58.51         3783
4               25.82            50.01         5557
5               11.90            82.32         3350
6                8.38            68.01         4709
```

```
# Data after preprocessing
head(dataset)
```

```
  jahreszeitFrühling jahreszeitHerbst jahreszeitSommer jahreszeitWinter
1                  0                0                1                0
2                  0                0                1                0
3                  0                0                0                1
4                  1                0                0                0
5                  0                1                0                0
6                  0                0                1                0
  einfuehrungsphase wetter arbeitstag temperatur windgeschwindigkeit
1                 0      1          1  0.7290219           0.4161604
2                 0      2          1  0.8035810           0.2657959
```

```
3                 0      2          1  0.2188669           0.2657959
4                 0      2          1  0.4417424           0.7390644
5                 1      1          0  0.7324960           0.3162211
6                 1      1          1  0.7103153           0.2092953
  luftfeuchtigkeit vermietungen
1           0.7361         4127
2           0.4402         8173
3           0.5851         3783
4           0.5001         5557
5           0.8232         3350
6           0.6801         4709
```

```r
# Prepare the data for training
dataset_features <- dataset |> select(-vermietungen)
dataset_labels <- dataset |> select(vermietungen)

# Use 80% of dataset for training and 20% for testing
dataset.split <- sample(
  c(TRUE, FALSE),
  nrow(dataset),
  replace=TRUE,
  prob=c(0.8, 0.2)
)

train.raw.dataset = clean.raw.dataset[dataset.split, ]
test.raw.dataset = clean.raw.dataset[!dataset.split, ]

train.dataset = dataset[dataset.split, ]
test.dataset = dataset[!dataset.split, ]

train.dataset_features = dataset_features[dataset.split, ]
train.dataset_labels = dataset_labels[dataset.split, ]
test.dataset_features = dataset_features[!dataset.split, ]
test.dataset_labels = dataset_labels[!dataset.split, ]
```

3 Methodenbeschreibung

https://www.datacamp.com/tutorial/neural-network-models-r

Gehen Sie hier auf die verwendete Methode zur Modellierung, Variablen, und Modellauswahl ein. Zitieren Sie hier auch die methodische Literatur.

3.1 Modellauswahl

Damit die Performance, zur Beurteilung hier der MAE, des Vorhersage-Modells bewertet werden kann, wird zuvor ein weiteres, zweites Modell erstellt. Hierbei wird ein lineares Modell als Vergleichs-Modell gewählt. Durch das simple Base-Line Modell, steht für den Vergleich ein einfaches und schnell zu trainierendes Modell zur Verfügung, welches auf Grund des Aufbaus einen geringen Ressourcen Bedarf hat. Dargestellt wird hierbei jedoch lediglich ein linearer Zusammenhang, der gegebenenfalls nicht ausreichend die Realität abbildet, wodurch es zu einem erhöhten MAE kommen kann. (Buch: Modellieren in R; Seite 322ff.) Dies wird in den nachfolgenden Kapiteln bezüglich der Anwendung der Modelle betrachtet.

Das Vorhersage-Modell basiert, im Gegensatz zum Vergleichs-Modell, auf einem neuronalen Netz. Dieses Netz wird aufgrund seiner höheren Komplexität eingesetzt, um den MAE signifikant zu verringern. Dies wird vor allem dadurch ermöglicht, dass das neuronale Netz in der Lage ist, von dem linearen Zusammenhang abzuweichen. In realen Szenarien existieren oft keine einfachen linearen Beziehungen zwischen den Variablen, weshalb das neuronale Netz, diese Beziehung erkennen und somit präzisere Vorhersagen liefern kann.

Ein weiterer Aspekt für die Auswahl des neuronalen Netzes, ist die Gewichtung der Variablen im Modell. Während des trainings passt das Netz die Gewichte der einzelnen Ergebnisse so an, dass die relevanten Zusammenhänge aus dem Trainingsdatensatz effektiv in das Modell eingebunden werden. Auf diese Weise trägt das Netz dazu bei, dass wichtige Korrelationen zwischen den Variablen erkannt und festgehalten werden. (https://www.ibm.com/de-de/topics/neural-networks)

3.2 Modellierung

Durch die Modellierung eines Vorhersage-Modells, für die Vermietung von Fahrrädern, kann ein Ausschnitt der Wirklichkeit betrachtet und nach Ermittlung von Regelmäßigkeiten auf andere Daten angewendet werden. Hierbei ist zu beachten, dass die Annäherung durch das Modell und somit der Zielvariable an die Wirklichkeit verzerrt oder falsch sein kann. Dies kann vorliegen, da nicht die ganze Gesamtheit zum trainieren des Modells bekannt ist. (Buch: Modellieren in R; Seite 246). Wie bereits im vorherigen Abschnitt beschrieben, wird bei der nachfolgenden Ausarbeitung ein lineares Regressions-Modell und ein Neuronales Netz trainiert und angewendet.

Das lineare Regressions-Modell stellt den Zusammenhang zwischen der abhängigen Variable „Vermietung“ und den im Datensatz vorhandenen unabhängigen Variablen dar. Dieses Zusammenhang ist, sofern vorhanden linear. Das Modell wird genutzt, um einen Trend abzubilden, der es ermöglicht, Vorhersagewerte für Kombinationen von Variablen aus dem Trainingsdatensatz zu ermitteln. Um die Genauigkeit des Modells zu bewerten, wird der MAE berechnet. (Buch: Modellieren in R; Seite 322ff.)

Die Modellierung des neuronalen Netzes ist komplexer als das Base-Line Modell. Die einzelnen Neuronen in den nachfolgend erklärten Schichten, haben immer jeweils eine Verbindung zu allen Neuronen der folgenden Schicht. Es besteht aus einer Eingabeschicht, der so genannten Inputlayer. Hier werden die zu betrachtenden Werte in das Netz geladen. Bei dem

vorliegenden Datensatz sind dies die zehn Variablen. Nachfolgende Schichten im neuronalen Netz sind die versteckten Schichten, die Hiddenlayer. Diese verarbeiten die Werte aus der vorherigen Schicht. Von der Schicht der Hiddenlayer liegen zwei vor. Nach der Verarbeitung werden die Werte in die letzte Schicht, der Outputlayer, übertragen. In dem vorliegenden Anwendungsfall, gibt es für diese Schicht ein Neuron. Es wird ausgegeben, ob eine Vermietung stattfindet oder nicht. Die Verarbeitung in diesem Netzwerk findet von der Inputlayer zum Outputlayer statt und die Ergebnisse werden lediglich an die nächste Ebene weitergegeben. Dadurch liegt ein Feedforward neuronales Netz vor, da die Daten in Vorwärtsrichtung fließen. (Awan 2023) Das zu erstellende Modell wird 5000 Mal trainiert um die Gewichtungen und somit die Ergebnisse zu optimieren. Damit eine Optimierung möglich ist, werden die besten Gewichte der einzelnen Durchgänge gespeichert. Um das Modell vor Overfitting, also der Überanpassung auf den Trainingsdatensatz, zu schützen, wird eine Dropout-Rate gesetzt. Diese sperrt je Durchlauf zufällig einen angegebenen Prozentsatz an Neuronen je Layer, die somit temporär nicht trainiert werden. Somit wird ein gleichmäßiges Training gewährleistet. (QUELLE?)

3.3 Variablen

Die Grundlage für die Modellierung bilden die zur Verfügung gestellten Daten. Diese Daten liegen, wie im vorherigen Kapitel zur *Explorativen Datenanalyse und Datenvorverarbeitung* beschrieben, in Form verschiedener Attribute mit unterschiedlichen Ausprägungen vor. Um diese Daten für die Modellierung nutzbar zu machen, wurde ein Preprocessing durchgeführt, bei dem die Daten für die weitere Verarbeitung angepasst und bereinigt wurden. Diese vorverarbeiteten Daten wurden anschließend separat von den Originaldaten gespeichert, um sicherzustellen, dass nur die bereinigten und angepassten Variablen in der Modellierung verwendet werden. In der nachfolgenden Modellierungsphase werden ausschließlich diese verarbeitenden Datensätze verwendet. Für beide Modelle, dem Base-Line Modell und dem neuronalen Netz, wurden sämtliche verfügbaren Attribute verwendet, um die Modelle zu trainieren. Das neuronale Netzwerk gewichtet hierbei die Relevanz der einzelnen Variablen um die bestmögliche Annäherung an die Beziehungen in den vorliegenden Trainingsdaten zu finden. (Buch: Neuronale Netze kompakt; Seite 71ff.)

Die Auswahl der Variablen innerhalb des Codes der zu trainierenden Modelle ist dynamisch gestaltet. Das bedeutet, dass das Hinzufügen neuer oder das Entfernen bestehender Attribute keine Anpassung des Codes erfordert. Dadurch bleibt die Modellierung und das Training flexibel gegenüber Änderungen im Datensatz.

4 Anwendung, Ergebnis und Vorhersage

Wenden Sie hier Ihr Modell an und Interpretieren Sie Ihr Ergebnis. Bei Einzelarbeiten sollte der reine Text (ohne Code, Abbildungen etc.) einen Umfang von ca. 1–2 Seiten haben, bei Gruppenarbeiten einen von ca. 2–4 Seiten.

4.1 Modell mit den Daten trainieren

4.1.1 Base-Line Modell

Um die Vorhersagegüte des neuronalen Netzes bestimmten zu können, wird zunächst das Base-Line Modell in Form einer linearen Regression erstellt, bei dem die Zielvariable von allen verfügbaren Prädiktoren im Trainingsdatensatz abhängt. Die Vorhersagegüte des Modells kann folgend anhand des Mean Absolute Error (MAE) bewertet und mit dem des neuronalen Netzes ins Verhältnis gesetzt werden.

```
# Train a simple lm on all the data
lm.model <- lm(vermietungen ~ ., data = train.raw.dataset)
summary(lm.model)
```

```

Call:
lm(formula = vermietungen ~ ., data = train.raw.dataset)

Residuals:
     Min       1Q   Median       3Q      Max
-3062.66  -474.18    25.49   537.77  2175.18

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)            3019.996    302.751   9.975  < 2e-16 ***
einfuehrungsphaseNein  2061.837     89.891  22.937  < 2e-16 ***
jahreszeitHerbst        637.492    132.829   4.799 2.28e-06 ***
jahreszeitSommer       -257.758    170.373  -1.513 0.131120
jahreszeitWinter       -693.963    153.715  -4.515 8.43e-06 ***
wetterNicht gut        -450.416    119.801  -3.760 0.000196 ***
wetterSchlecht        -1821.848    285.051  -6.391 4.73e-10 ***
arbeitstagNein         -134.127     95.627  -1.403 0.161535
temperatur              123.055      9.836  12.511  < 2e-16 ***
windgeschwindigkeit     -39.740      9.030  -4.401 1.40e-05 ***
luftfeuchtigkeit         -9.764      4.084  -2.391 0.017276 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 875.6 on 387 degrees of freedom
Multiple R-squared:  0.8053,    Adjusted R-squared:  0.8003
F-statistic: 160.1 on 10 and 387 DF,  p-value: < 2.2e-16
```

```
# Calculate the MAE to compare models
mae(
  test.raw.dataset$vermietungen,
```

```
  predict(lm.model, newdata = test.raw.dataset)
)
```

```
[1] 623.8114
```

Die Residuen reichen von -3641,3 bis 2384,8, wobei der Median nahe bei 0 liegt (-3,4). Dies deutet darauf hin, dass das Modell die Daten gut mittelt, obwohl einige Abweichungen auftreten. Die Variable *einfuehrungsphaseNein* hat den stärksten positiven Effekt auf die Vermietungen (2057,883), was darauf schließen lässt, dass die Vermietungen signifikant höher sind, wenn keine Einführungsphase vorliegt. Die Jahreszeiten zeigen unterschiedliche Effekte. Im Herbst steigen die Vermietungen signifikant (+562,431), während sie im Winter stark sinken (-765,179). Der Sommer zeigt einen leichten, aber nicht stark signifikanten Rückgang (-297,132). Wetterbedingungen spielen ebenfalls eine wichtige Rolle: Bei „Nicht gutem“ und „Schlechtem“ Wetter sinken die Vermietungen signifikant um -390,531 bzw. -2115,621. Die Temperatur hat einen positiven Effekt: Mit jedem Grad Anstieg erhöhen sich die Vermietungen um 120,920. Windgeschwindigkeit und Luftfeuchtigkeit haben negative Effekte auf die Vermietungen (-43,602 bzw. -10,445).

Der Multiple R-squared-Wert von 0,7975 zeigt, dass fast 80% der Variabilität in den Vermietungen durch die erklärenden Variablen im Modell erfasst wird und weist somit eine solide Anpassung des Modells auf.

Der Residual Standard Error von 903,3 zeigt, dass die durchschnittliche Abweichung der Vorhersagen vom tatsächlichen Wert etwa 903 Vermietungen beträgt.

Der F-statistic-Wert von 152,8 mit einem sehr niedrigen p-Wert ($< 2.2e\text{-}16$) betont, dass das Modell insgesamt signifikant ist und die unabhängigen Variablen eine hohe Erklärungskraft für die Variation in den Vermietungen haben.

Die lineare Regression stellt dar, dass insbesondere die Einfuehrungsphase und extreme Wetterbedingungen wie schlechtes Wetter sich stark auf die Vermietungen auswirken. Die Jahreszeiten haben ebenfalls einen signifikanten Einfluss, wobei der Winter einen deutlichen Rückgang und der Herbst einen Anstieg der Vermietungen bewirkt. Das Modell erklärt einen großen Teil der Variabilität der Vermietungen und ist damit eine solide Vorhersage.

4.1.2 Unser Modell

Zunächst wird, wie in dem Base-Line Modell, ein zufälliger Seed gesetzt (*set_random_seed(69)*), um die Reproduzierbarkeit der Ergebnisse zu gewährleisten. Zusätzlich die Nutzung der GPU ermöglicht, sofern diese verfügbar ist. Im Anschluss wird der Pfad für die Speicherung der Modellgewichte definiert (*checkpoint_path*), sodass das Modell während des Trainings regelmäßig gesichert werden kann.

Um eine optimale Skalierung der Eingabedaten zu gewährleisten, wird ein Normalisierungslayer (*layer_normalization*) erstellt und an die Verteilung der Features im Datensatz (*dataset_features*) angepasst. Hierdurch wird sichergestellt, dass die Eingabedaten vor dem Durchlaufen des neuronalen Netzes auf eine vergleichebare Skala gebracht werden.

Das Modell wird mit dann mit einer Eingabeschicht definiert, deren Dimensionen den Merkmalen des Datensatzes entsprechen (*input <- layer_input(shape = dim(dataset_features))*).

Das neuronale Netz selbst wird durch ein Eingabelayer definiert, dessen Form durch die Anzahl der Features im Datensatz bestimmt wird. Anschließend folgt eine Sequenz von weiteren Layern, beginnend mit der Normalisierung der Eingaben. Ein Dropout-Layer mit einer Rate von 5% wird hinzugefügt, um das Modell vor Überanpassung zu schützen. Darauf folgen zwei voll verbundene Layer (*dense layers*) mit jeweils 32 Neuronen, welche die ReLU-Aktivierungsfunktion verwenden, um nicht-lineare Zusammenhänge im Datensatz zu modellieren. Die Ausgabeschicht besteht aus einem einzigen Neuron, welches die Vorhersage der Zielvariable liefert.

Das Modell wird anschließend kompiliert, wobei der Adam-Optimierer (*optimizer = 'adam'*) zur Optimierung der Gewichte verwendet wird. Als Verlustfunktion wird der mittlere quadratische Fehler (*loss = 'mse'*) eingesetzt, während der mittlere absolute Fehler (*mae*) als Metrik zur Bewertung der Modellleistung herangezogen wird. Ein Callback (*callback_model_checkpoint*) sorgt dafür, dass die Gewichte des Modells nach jeder Epoche gespeichert werden, wobei nur die besten Gewichte (gemessen an der Testleistung) gesichert werden.

Das Training des Modells erfolgt über 5000 Epochen (*epochs = 5000*) mit einer Batch-Größe von 64. Die Trainingsdaten werden in Form von Matrizen (*as.matrix(train.dataset_features)*) bereitgestellt, und der Testdatensatz wird verwendet, um die Leistung des Modells während des Trainings zu überwachen. Nach Abschluss des Trainings werden die besten während des Trainings gespeicherten Gewichte geladen (*load_model_weights*), um sicherzustellen, dass die anschließende Evaluation auf der bestmöglichen Version des Modells basiert.

Die abschließende Bewertung des Modells erfolgt anhand der Testdaten, wobei die zuvor definierten Metriken, insbesondere der mittlere absolute Fehler, zur Beurteilung der Vorhersagegenauigkeit des Modells herangezogen werden.

```
# Adapt a normalizer on the data as we didn't normalize it yet
normalizer <- layer_normalization(axis = -1L)
normalizer |> adapt(as.matrix(dataset_features))

# Define training for neural networks
train.nn <- function(name, layers_hidden, epochs) {
  # Set random seed
  set_random_seed(69, disable_gpu = FALSE)

  # Define paths for the states
  chache_path <- str_interp("cache/nn.training.${name}.rds")
  chache_dir <- fs::path_dir(chache_path)
  if (!dir.exists(file.path(chache_dir))) {
    dir.create(file.path(chache_dir))
  }
```

```
checkpoint_path <- str_interp("checkpoints/${name}.weights.h5")
checkpoint_dir <- fs::path_dir(checkpoint_path)
if (!dir.exists(file.path(checkpoint_dir))) {
  dir.create(file.path(checkpoint_dir))
}

# Build model
inputs <- layer_input(
  shape = dim(dataset_features)[2],
  name = "inputs"
)
outputs <- inputs |>
  layers_hidden() |>
  layer_dense(
    units = 1,
    name = "output"
  )
nn.model <- keras_model(inputs, outputs)
nn.model$name <- name

# Assign the optimizer and the metrics to use (mse and mae for regression)
nn.model |> compile(
  optimizer = 'adam',
  loss = 'mse',
  metrics = list('mae')
)

# Train the model with a 20% validation split
if (file.exists(chache_path) & file.exists(checkpoint_path)) {
  nn.training <- readRDS(chache_path)
} else {
  # Create a callback that saves the model weights
  cp_callback <- callback_model_checkpoint(
    filepath = checkpoint_path,
    save_weights_only = TRUE,
    save_best_only = TRUE,
  )

  # Train model and use test data to pick the best performing one
  nn.training <- nn.model |>
    fit(
      as.matrix(train.dataset_features),
      as.matrix(train.dataset_labels),
      epochs = epochs,
      batch_size = 64,
```

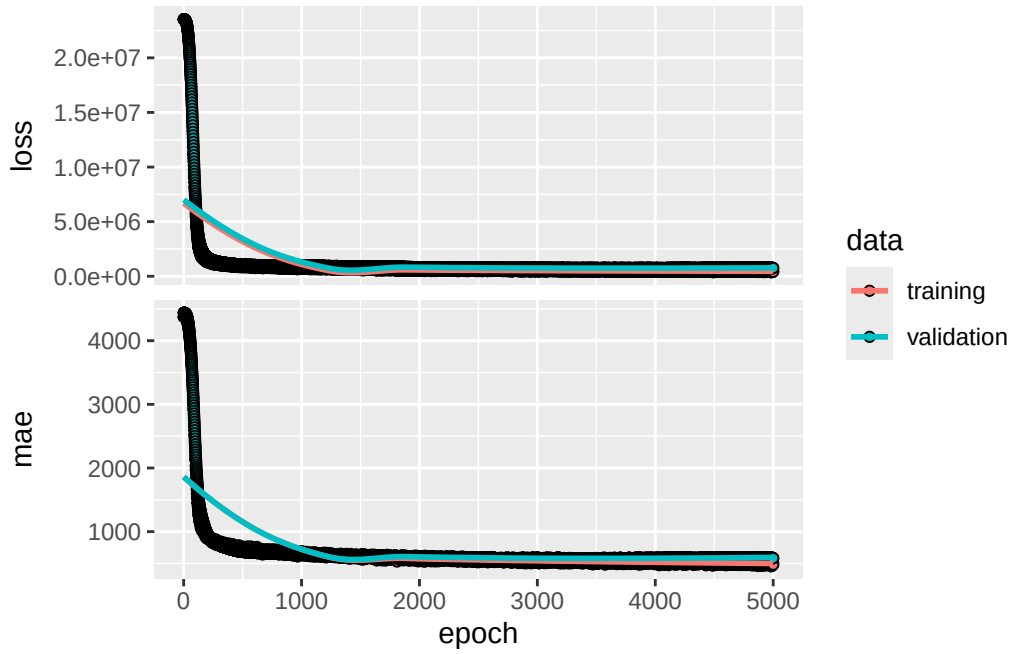
```
        verbose = FALSE,
        validation_data = list(
          as.matrix(test.dataset_features),
          as.matrix(test.dataset_labels)
        ),
        callbacks = list(cp_callback),
      )
    saveRDS(nn.training, chache_path)
  }

  # Loads the weights
  load_model_weights(nn.model, checkpoint_path)

  nn.eval <- nn.model |> evaluate(
    as.matrix(test.dataset_features),
    as.matrix(test.dataset_labels),
    batch_size = 64,
    verbose = 0
  )

  results <- list(
    model = nn.model,
    training = nn.training,
    mae = nn.eval$mae
  )
  return(results)
}
```

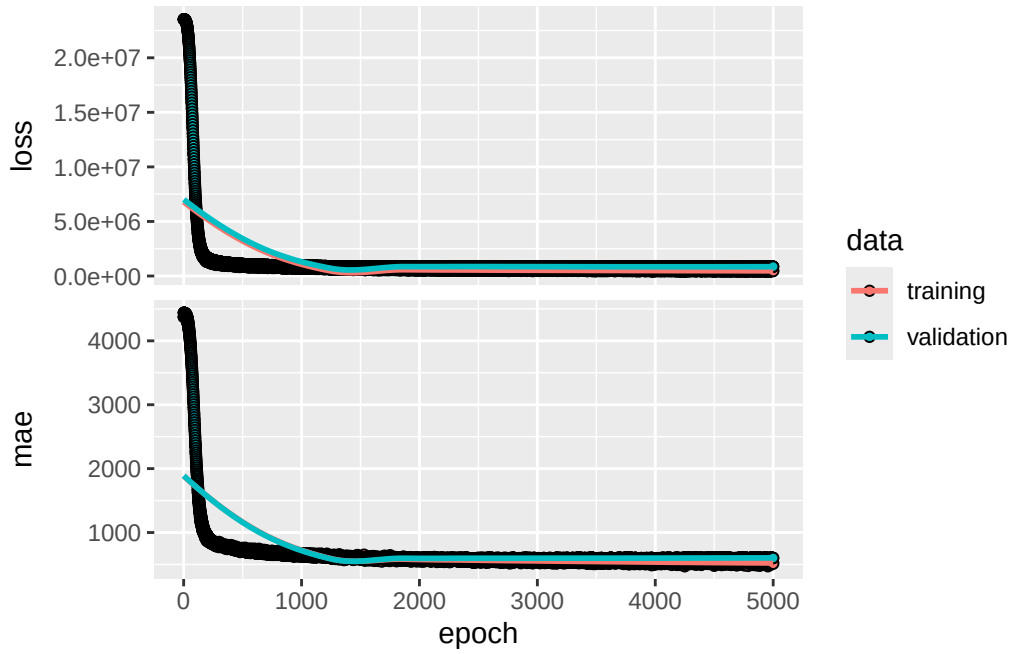
```
# Train model A
results.A <- train.nn(
  name = "Model.A",
  layers_hidden = \(inputs) inputs |>
    normalizer() |>
    layer_dropout(rate = 0.05) |>
    layer_dense(units = 32, activation = 'relu') |>
    layer_dense(units = 32, activation = 'relu'),
  epochs = 5000
)
results.A$training |> plot()
```




```
results.A$mae
```

```
[1] 583.1541
```

```
# Train model B
results.B <- train.nn(
  name = "Model.B",
  layers_hidden = \(inputs) inputs |>
    normalizer() |>
    layer_dense(units = 32, activation = 'relu') |>
    layer_dense(units = 32, activation = 'relu') |>
    layer_dropout(rate = 0.1),
  epochs = 5000
)
results.B$training |> plot()
```




```
results.B$mae
```

```
[1] 601.0692
```

4.2 Modell zur Vorhersage anwenden

Um die Anwendung des trainierten neuronalen Netzes zur Vorhersage der Vermietungen durchzuführen, werden zunächst die Vorhersagen mit dem bereits trainierten Modell getroffen. Hierzu wird der Eingabedatensatz (*application*) in eine Matrix umgewandelt, sodass das Modell die Daten im erforderlichen Format erhält. Das neuronale Netz (*nn.model*) verarbeitet die Eingaben und generiert Vorhersagen, die in der Variablen *predictions* gespeichert werden.

Anschließend werden diese Vorhersagen in den bestehenden Datensatz (*raw.application*) integriert. Zuerst wird eine Kopie des ursprünglichen Datensatzes (*raw.application*) erstellt, um die originalen Daten unverändert zu lassen. In der Kopie (*pred.application*) wird eine neue Spalte 'vermietungen' hinzugefügt, die die berechneten Vorhersagen enthält. Dabei werden die Vorhersagewerte in ganze Zahlen (*as.integer(predictions)*) umgewandelt, bevor sie in die Spalte eingefügt werden. Hierdruch wird ein erweiterten Datensatz erstellt, der sowohl die Originaldaten als auch die vom Modell vorhergesagten Werte beinhaltet.

```
# Predict with the model
predictions <- results.A$model(as.matrix(application))

# Add predictions to the application
```

```r
pred.application <- raw.application
pred.application$vermietungen <-as.integer(predictions)
```

Abschließend wird der erweiterte Datensatz (*pred.application*), der die Vorhersagen des Modells enthält, in eine CSV-Datei exportiert.

```r
# Save results
write.csv2(pred.application, file = "Prognose_IhrName.csv")
```

5 Zusammenfassung

Fassen Sie kurz die zentralen Ergebnisse zusammen (0,5–1 Seite). Gehen Sie auch auf die Grenzen Ihrer Analyse ein.

Hinweis: Laden Sie das aus der qmd-Datei gerenderte pdf-Dokument als Prüfungsleistung im OC hoch (keine gedruckte Abgabe erforderlich). Als Zusatzmaterial laden Sie die qmd-Datei und die csv-Datei mit den Daten Ihrer Prognose (`Prognose_IhrName.csv`) hoch.

6 Literatur

Hier stehen die im Text verwendeten Quellen:

- Nachname Autor1, Anfangsbuchstabe Vorname Autor1, Nachname Autor2, Anfangsbuchstabe Vorname Autor2 1 & Nachname Autor3, Anfangsbuchstabe Vorname Autor3 (Jahr der Veröffentlichung). Titel des Beitrags. Weitere Publikationsinformationen.

Im Text werden die Quellen nach der Harvard-Intext-Zitation in Klammern angegeben (siehe Leitfaden).

ChatGPT & Co: Beachten Sie die Hinweise dazu im Leitfaden zur formalen Gestaltung von Seminar- und Abschlussarbeiten, Stand 24/01.

Awan, Abid Ali. 2023. „Building Neural Network (NN) Models in R“. 2023. https://www.datacamp.com/tutorial/neural-network-models-r.