Computational Statistics

Prof. Dr. Dominik Liebl

Inhaltsverzeichnis

Informationen **5**

1 Regressionsmodelle im Kontext des Maschinellen Lernens **7**

1.1 Das allgemeine Regressionsmodell 11

1.2 Das multivariate lineare Regressionsmodell 14

1.3 Modellauswahl 17

1.4 Anwendung: Vorhersage des Benzinverbrauchs 21

1.5 Ende 24

Informationen

Dies ist das Skript zur Vorlesung *Computational Statistik*

Vorlesungszeiten

RCodes

Die RCodes zu den einzelnen Kapiteln können hier heruntergeladen werden: RCodes

Leseecke

Folgende *frei zugängliche* Lehrbücher enthalten Teile dieses Kurses. In den jeweiligen Kapiteln, werde ich auf die einzelnen Bücher verweisen.

- Pattern Recognition and Machine Learning (by Christopher Bishop)
- An Introduction to Statistical Learning, with Applications in R (by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani).
- Statistical Learning with Sparsity: the Lasso and Generalizations (by Trevor Hastie, Robert Tibshirani and Martin Wainwright).
- Elements of Statistical Learning: Data mining, Inference and Prediction (by Trevor Hastie, Robert Tibshirani and Jerome Friedman).
- Computer Age Statistical Inference: Algorithms, Evidence and Data Science (by Bradley Efron and Trevor Hastie)

Wochentag	Uhrzeit	Hörsaal
Dienstag	9:15-10:45	Online-Vorlesung
Freitag	8:30-10:00	Online-Vorlesung

Florence Nightingale

Das Logo zu diesem Skript stammt von einer Briefmarke zur Erinnerung an die Krankenschwester und inspirierende Statistikerin, Florence Nightingale. Nightingale war die Begründerin der modernen westlichen Krankenpflege und Pionierin der visuellen Datenanalyse. Sie nutzte statistische Analysen, um Missstände in Kliniken zu erkennen und diese dann auch nachweislich abzustellen. Sie ist die erste Frau, die in die britische Royal Statistical Society aufgenommen wurde; später erhielt sie auch die Ehrenmitgliedschaft der American Statistical Association.

Kapitel 1

Regressionsmodelle im Kontext des Maschinellen Lernens

Lineare Regressionsmodelle gehören zu den erfolgreichsten statistischen Modellen, da sie

- vergleichsweise **einfach zu interpretieren** sind und
- zugleich **äußerst flexibel** sind.

In diesem Kapitel betrachten wir das multivariate (oder multiple) lineare Regressionsmodell als **Prädiktionsmodell** im Rahmen einer Anwendung des maschinellen Lernens.

Lernziele für dieses Kapitel

Sie können ...

- ein **Anwendungsfeld** des linearen Regressionsmodell als Prädiktionsmodell **benennen**.
- die **Probleme** der Auswahl eines geeigneten Prädiktionsmodells am Beispiel der Polynomregression **benennen und erläutern**.
- die **Grundidee** der Validierungsdaten-Methode **erläutern**.
- die **Grundidee** der k-fachen Kreuzvalidierung **erläutern**.

Begleitlektüren

Zur Vorbereitung der Klausur ist es grundsätzlich ausreichend dieses Kapitel durchzuarbeiten - aber Lesen hat ja noch nie geschadet. Empfehlenswerte weiterführende Literatur:

- Kapitel 3 in **An Introduction to Statistical Learning, with Applications in R** (James et al., 2021). Die pdf-Version des Buches ist hier frei erhältlichen: **www.statlearning.com**
- Kapitel 3 in **Pattern Recognition and Machine Learning** (Bishop, 2006). Die pdf-Version des Buches ist frei erhältlichen: **pdf-Version**
- Kapitel 6 in **Introduction to Econometrics with R** (Hanck et al., 2021). Freies Online-Buch: **www.econometrics-with-r.org**

R-Pakete und Datenbeispiel für dieses Kapitel

Folgende Pakete werden in diesem Kapitel benötigt.

- **tidyverse**: Viele nützliche Pakete zur Datenverarbeitung.
- **GGally**: Enthält die Funktion `ggpairs()` zur Erzeugung von Pairs-Plots
- **ISLR**: Enthält die `Auto` Daten

Falls noch nicht geschehen, müssen diese Pakete installiert und geladen werden:

```
## Installieren
install.packages("tidyverse")
install.packages("GGally")
install.packages("ISLR")
## Laden
library("tidyverse") # Viele nützliche Pakete zur Datenverarbeitung
library("GGally")    # Pairs-Plot
library("ISLR")      # Enthält die Auto-Daten
data(Auto)           # Auto-Daten abrufbar machen
```

Als Datenbeispiel für diese Kapitel betrachten wir den `Auto` Datensatz im R-Paket ISLR. Wir betrachten folgende Auswahl der Variablen im Datensatz `Auto`:

- **Zielvariable:**
  - **Verbrauch (km/Liter)**
- **Prädiktorvariablen:**
  - **Gewicht (kg):** Schwerere Autos verbrauchen wahrscheinlich mehr.

- **Leistung (PS):** Höhere Leistung geht wohl auch mit höherem Verbrauch einher.
- **Hubraum (ccm):** Großer Hubraum … höherer Verbrauch?

Achtung: Es gibt sicherlich noch weitere relevante Prädiktorvariablen. Obige Auswahl ist jedoch relativ einfach zu erheben und ermöglicht eventuell bereits eine **gute Prädiktion des Verbrauches** im Rahmen eines **Regressionsmodells**. Falls dem so ist, könnte unser Prädiktionsmodell dazu dienen, nach Auffälligkeiten bei den herstellerseitigen Verbrauchsangaben zu suchen. Besonders große Abweichung zwischen Modellprädiktion und Herstellerangabe wäre ein mögliches Indiz auf unlautere Zahlenschönung.

Aufbereitung der Daten:

```
## Auswahl und Aufbereitung der Variablen
Auto_df <- Auto %>%
  mutate(Verbrauch = mpg * (1.60934/3.78541), # Verbrauch (km/Liter)
         Gewicht   = weight * 0.45359,        # Gewicht (kg)
         PS        = horsepower,              # Pferdestärken (PS)
         Hubraum   = displacement * 2.54^3    # Hubraum (ccm)
         ) %>%
 select(Verbrauch, Gewicht, PS, Hubraum)

n <- nrow(Auto_df) # Stichprobenumfang
```

Insgesamt enthält der betrachtete Datensatz also fünf Variablen zu $n = 392$ verschiedenen Autos. Dies sind die ersten sechs Zeilen des Datensatzes:

Verbrauch (km/Liter)

Gewicht (kg)

Pferdestärken (PS)

Hubraum (ccm)

7.65

1589.38

130

5030.83

6.38

1675.11

165

5735.47

7.65

1558.54

150

5211.09

6.80

1557.17

150

4981.67

7.23

1564.43

140

4948.89

6.38

1969.03

198

7030.05

Um sich einen Überblick zu den Beziehungen zwischen den Variablen zu verschaffen, eignet sich ein **Pairs-Plot** sehr gut (siehe Abbildung 1.1):

```
ggpairs(Auto_df,
upper = list(continuous = "density", combo = "box_no_facet"),
lower = list(continuous = "points", combo = "dot_no_facet"))
```

Der Pairs-Plot veranschaulicht alle paarweisen Zusammenhänge zwischen den Variablen im Datensatz `Auto_df`. Uns interessieren hierbei in erster Linie die Zusammenhänge zwischen der Zielvariable **Verbrauch** und den **Prädiktorvariablen**:

- $Y =$**Verbrauch** und …
  - $G =$ **Gewicht**$_i$**:** haben einen nicht linearen, negativen Zusammenhang.
  - $P =$ **PS:** haben einen nicht linearen, negativen Zusammenhang.
  - $H =$ **Hubraum:** haben einen nicht linearen, negativen Zusammenhang.

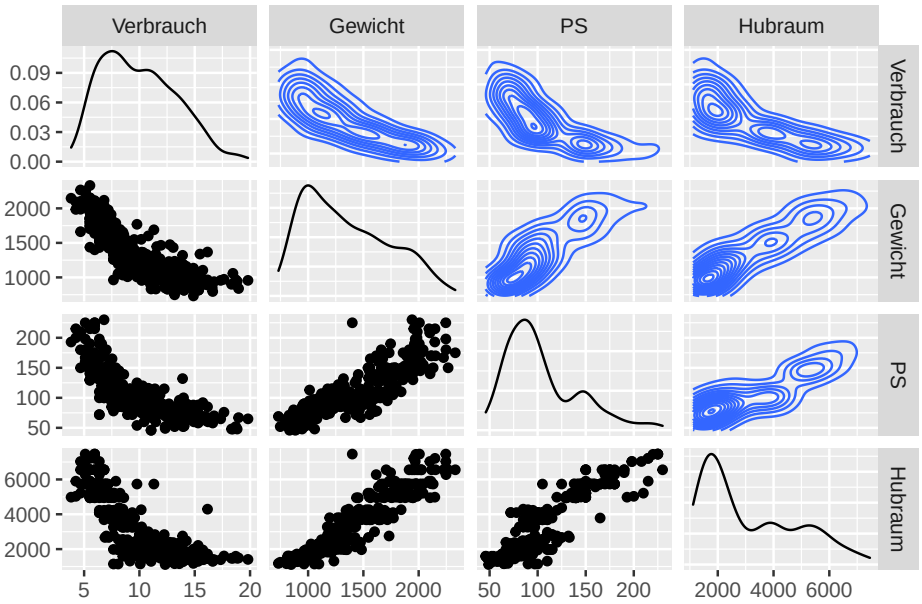


Abbildung 1.1: Pairs-Plot zur Veranschaulichung der paarweisen Zusammenhänge zwischen den Variablen.

1.1 Das allgemeine Regressionsmodell

Die einzelnen Prädiktorvariablen werden gerne kompakt zu einer multivariaten Prädiktorvariablen $X = (X_1, X_2, \ldots, X_p)$ zusammengefasst; in unserem Benzinverbrauchsbeispiel also $X = (G, P, H, B)$. So lässt sich das **allgemeines Regressionsmodell** schreiben als

$$Y = f(X) + \varepsilon$$

wobei

- f den **systematischen Zusammenhang** zwischen der Zielvariable Y und den Prädiktorvariablen X beschreibt und
- ε ein **Fehlerterm** ist, der unabhängig von X ist und Mittelwert $E(\varepsilon) = 0$ Null hat.

Daraus ergibt sich folgender Zusammenhang zwischen der **allgemeinen Regressionsfunktion** f und dem bedingten Mittelwert von Y gegeben X:

$$E(Y|X) = f(X)$$

Die Funktion f beschreibt also den bedingten Mittelwert von Y gegeben X. Ziel ist es nun, die Regressionsfunktion f aus den Daten zu schätzen (lernen).

Achtung: Die Annahme der Unabhängigkeit zwischen ε und X kann in der Praxis verletzt sein. Die Verletzung dieser Unabhängigkeitsannahme erlaubt insbesondere keine kausale Interpretation der Ergebnisse, daher betrachtet die Literatur zur Kausalinferenz viele Möglichkeiten diese Unabhängigkeitsannahme durch eine weniger strikte Annahmen zu ersetzen. In der Literatur zur prädiktiven Inferenzen wird eine Verletzung der Unabhängigkeitsannahme weniger kritisch gesehen, da eine Prädiktion trotz verletzter Unabhängigkeitsannahme sehr gut sein kann. Eine schöne und gut lesbare Übersicht zu den Unterschieden zwischen der Kausalinferenz und der prädiktiven Inferenzen findet man, z.B., im Artikel To Explain or To Predict? (Shmueli, 2010).

Abbildung 1.2 zeigt ein Beispiel von 50 simulierten Daten (künstlich erzeugte Fake-Daten). Der Plot legt nahe, dass man das Einkommen mit Hilfe der Ausbildungsjahre vorhersagen kann. Normalerweise ist die wahre Funktion f, welche die Verbindung zwischen Y und X beschreibt, unbekannt und muss aus den Daten geschätzt werden. Da es sich hier um simulierte Daten handelt, können wir den Graph der Funktion f als blaue Linie plotten. Einige der 50 Beobachtungspunkte (X, Y) liegen über der Regressionsfunktion $f(X)$, andere darunter. Im Großen und Ganzen haben die Fehlerterme einen Mittelwert von Null.

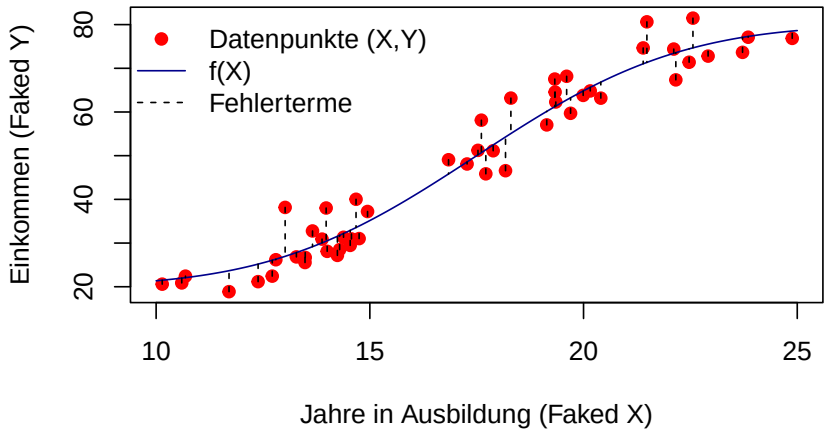


Abbildung 1.2: Simulierte (künstlich erzeugte) Daten zur Veranschaulichung einer allgemeinen, univariaten Regressionsbeziehung.

Abbildung 1.3 zeigt ein simuliertes Beispiel einer allgemeinen, bivariaten Regressionsbeziehung

$$Y = f(X) + \varepsilon \quad \text{mit} \quad X = (X_1, X_2).$$

1.1.1 Der Prädiktionsfehler (zwischen Y und $\hat{Y}$)

In vielen Datenproblemen sind zwar die Prädiktorvariablen X bekannt (z.B. Gewicht, PS und Hubraum eines neuen Autos), aber die dazugehörige Zielvariable Y ist unbekannt. Da sich der Fehlerterm zu Null mittelt, lässt sich in solch einem Fall das unbekannte Y durch

$$\hat{Y} = \hat{f}(X)$$

vorhersagen, wobei

- $\hat{f}$ für unsere Schätzung von f steht und
- $\hat{Y}$ die Vorhersage von Y für gegebene Prädiktorvariablen X ist.

Die Genauigkeit der Vorhersage von $\hat{Y}$ für Y hängt von zwei verschiedenen Prädiktionsfehlergrößen ab:

- **Reduzierbarer Prädiktionsfehler** aufgrund des Schätzfehlers in $\hat{f}$. Eine genauere Schätzung kann diesen Fehler reduzieren.
- **Nicht reduzierbarer Prädiktionsfehler** aufgrund des Fehlerterms ε. Das ist der Fehler, den wir selbst bei perfekter Schätzung von f nicht reduzieren können.

Der **nicht reduzierbare Fehler** ε enthält alle nicht messbaren und nicht gemessenen Variablen, die ebenfalls einen Einfluss auf Y haben. Und da wir diese Variablen nicht messen können, können wir sie auch nicht verwenden, um f zu schätzen.

Sei nun $\hat{f}$ eine gegebene Schätzung von f und seien X gegeben Werte der Prädiktorvariablen welche die Vorhersage $\hat{Y} = \hat{f}(X)$ ergeben. Nehmen wir nun für einen Moment an, dass $\hat{f}$ und X gegeben und fest (also nicht zufällig) sind, dann

$$\begin{aligned} E\left[(Y-\hat{Y})^2\right] &= E\Big[(\overbrace{f(X)+\varepsilon}^{=Y} - \overbrace{\hat{f}(X)}^{=\hat{Y}})^2\Big] \\ &= E\left[\left((f(X)-\hat{f}(X)\right)^2 + 2\left((f(X)-\hat{f}(X)\right)\varepsilon + \varepsilon^2\right] \\ &= \underbrace{\left((f(X)-\hat{f}(X)\right)^2}_{\text{reduzierbar}} + \underbrace{\operatorname{Var}(\varepsilon)}_{\text{nicht reduzierbar}} \end{aligned}$$

Der mittlere quadratische Prädiktionsfehler $E\left[(Y-\hat{Y})^2\right]$ lässt sich also in eine reduzierbare und eine nicht reduzierbare Fehlerkomponente zerlegen.

1.2 Das multivariate lineare Regressionsmodell

Um die allgemeine Regressionsfunktion $f(X) = E(Y|X)$ mit Hilfe der Daten zu schätzen (lernen), gibt es sehr viele verschiedenen Möglichkeiten. Eine der erfolgreichsten und am häufigsten verwendete Möglichkeit ist das **multivariaten linear Regressionsmodell**. Dieses Modell ist die **strukturelle Modellannahme**, dass sich die unbekannte Regressionsfunktion f als lineare Funktion (linear in den Modellparametern $\beta_0, \beta_1, \dots, \beta_p$) schreiben lässt:

$$f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

Unter dieser Modellannahme wird das allgemeine Regressionsmodell $Y = f(X) + \varepsilon$ zum multivariaten (multiplen) linearen Regressionsmodell

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon.$$

Zusammen mit der Annahme, dass ε unabhängig von X ist, und dass $E(\varepsilon) = 0$, können wir mit dieser Modellannahme den unbekannten bedingten Mittelwert $E(Y|X) = f(X)$ vereinfacht schreiben als

$$E(Y|X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

Vorteile des **multivariaten linearen Regressionsmodells:**

- Anstatt eine gänzlich unbekannte Funktion f schätzen (lernen) zu müssen, muss man lediglich die unbekannten Parameterwerte $\beta_0, \beta_1, \dots, \beta_p$ schätzen.
- Die Modellstruktur ist **keine Black Box**, sondern gibt Aufschluss darüber den **assoziativen Zusammenhang** zwischen den Prädiktorvariablen und der Zielvariablen.
- Die lineare Modellstruktur ist **extrem flexibel**, da Transformationen der Prädiktorvariablen grundsätzlich erlaubt sind.

  Gerade die große Flexibilität linearer Modelle werden wir nutzten müssen, um die **nicht linearen Zusammenhänge** zwischen den Prädiktorvariablen und der Zielvariablen in unserem Benzinverbrauchsbeispiel berücksichtigen zu können (siehe Abbildung 1.1).

1.2.1 Schätzung

Wir wollen nun diejenige Funktion

$$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p$$

finden, sodass $Y \approx \hat{f}(X)$ für alle Datenpunkte (Y, X).

Zur Berechnung von $\hat{f}$ können wir die **beobachteten Daten** als **Trainingsdaten** verwenden:

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \quad \text{wobei} \quad x_i = (x_{i1}, x_{i2}, \dots, x_{ip})^T.$$

Im Folgenden werden wir oft die Notation

$$x_{ij}, \quad i = 1, \dots, n, \quad j = 1, \dots, p$$

verwenden, um die $j$te Prädiktorvariable der $i$ten Beobachtung zu bezeichnen. Der Laufindex $j = 1, \dots, p$ repräsentiert die einzelnen Prädiktorvariablen (z.B. Verbrauch, Gewicht, Pferdestärken, und Hubraum im `Auto_df` Datensatz) und der Laufindex $i = 1, \dots, n$ repräsentiert die einzelnen Beobachtungen (z.B. gespeichert als Zeilen im `Auto_df` Datensatz).

> **Idee:** Die Trainingsdaten $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ enthalten Information zum unbekannten Regressionsmodell f, da (so die Grundidee) die Daten von eben diesem Modell erzeugt wurden. Ziel ist also die unbekannte Regressionsfunktion f mit Hilfe der Trainingsdaten zu schätzen (erlernen).

Für jede mögliche Schätzung $\hat{f}$ von f können wir die beobachteten Werte der Zielvariablen $y_1, \dots, y_n$ mit den vorhergesagten Werten

$$\hat{y}_i = \hat{f}(x_i) = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots + \hat{\beta}_p x_{ip}$$

vergleichen, indem wir die **Residuen**

$$e_i = y_i - \hat{y}_i \quad i = 1, \dots, n$$

betrachten.

Die gängigste Methode zur Schätzung der unbekannten Modellparameter $\beta_0, \beta_1, \dots, \beta_p$ ist die **Methode der kleinsten Quadrate**. Wir definieren die **Residuenquadratsumme** RSS (Residual Sum of Squares) als:

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2$$

oder äquivalent als

$$\text{RSS} = (y_1 - \hat{\beta}_0 + \hat{\beta}_1 x_{11} + \cdots + \hat{\beta}_p x_{1p})^2 + \cdots + (y_n - \hat{\beta}_0 + \hat{\beta}_1 x_{n1} + \cdots + \hat{\beta}_p x_{np})^2$$

Die Methode der kleinsten Quadrate bestimmt die Parameterschätzungen $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p)^T$ durch Minimierung der Residuenquadratsumme RSS. Nach ein paar Rechnungen kann man zeigen, dass

$$\begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_p \end{pmatrix} = \left(\begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ \vdots & & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix}^T \begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ \vdots & & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix} \right)^{-1} \begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ \vdots & & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix}^T \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}$$

1.2.2 Polynomregression

Die **Polynomregression** ist eine Möglichkeit, die nicht linearen Beziehungen zwischen der Zielvariablen und den Prädiktorvariablen in unserem Benzinverbrauchsproblem (siehe Abbildung 1.1) berücksichtigen zu können. So kann, zum Beispiel, der nicht lineare Zusammenhang zwischen `Verbrauch` und Leistung `PS` sehr flexibel als Polynomfunktion modelliert werden:

$$\texttt{Verbrauch} = \beta_0 + \beta_1 \texttt{Ps} + \beta_2 \texttt{PS}^2 + \cdots + \beta_p \texttt{PS}^p$$

Je höher der Grad p des Polynoms, desto flexibler ist ein Polynomregressionsmodell und ermöglicht so auch die Modellierung nicht linearen Zusammenhänge. Das Polynomregressionsmodell ist jedoch für alle Polynomgrade p ein **(multivariates) lineares Regressionsmodell**, denn es ist linear bezüglich der Modellparameter $\beta_0, \beta_1, \ldots, \beta_p$.

```
## Polynom Regressionen
polreg_1 <- lm(Verbrauch ~ poly(PS, degree = 1, raw=TRUE), data = Auto_df)
polreg_2 <- lm(Verbrauch ~ poly(PS, degree = 2, raw=TRUE), data = Auto_df)
polreg_5 <- lm(Verbrauch ~ poly(PS, degree = 5, raw=TRUE), data = Auto_df)
## Data-Frame zum Abspeichern der Prädiktionen
plot_df       <- tibble("PS" = seq(45, 250, len=50))
## Abspeichern der Prädiktionen
plot_df$fit_1 <- predict(polreg_1, newdata = plot_df)
plot_df$fit_2 <- predict(polreg_2, newdata = plot_df)
plot_df$fit_5 <- predict(polreg_5, newdata = plot_df)
## Ploten
plot(Verbrauch ~ PS, data = Auto_df, ylim=c(2,20),
     xlab="Leistung (PS)", pch=21, col="gray", bg="gray", cex=1.5)
with(plot_df, lines(x = PS, y = fit_1, lwd=2, col="orange"))
with(plot_df, lines(x = PS, y = fit_2, lwd=2, col="blue"))
with(plot_df, lines(x = PS, y = fit_5, lwd=2, col="darkgreen"))
legend("topright", lty=c(NA,1,1,1), pch=c(21,NA,NA,NA),
       col=c("gray","orange","blue","darkgreen"), pt.bg="gray", pt.cex=1.5,
       legend=c("Datenpunkte", "Grad 1", "Grad 2", "Grad 5"), bty="n")
```

Überanpassung

Zusätzlich zur Wahl der Modellparameter $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ besteht hier nun das Problem der Wahl des Grades p des Polynoms als weiteren Modellparameter

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2}^2 + \cdots + \hat{\beta}_p x_{ip}^p + e_i$$

Wenn man jedoch versucht, alle Modellparameter (also $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ **und** p) durch Minimieren der Trainingsdaten-RSS

$$\text{RSS} \equiv \text{RSS}(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p, p) = e_1^2 + e_2^2 + \cdots + e_n^2$$

zu schätzen, so ergibt sich ein Problem das als **Überanpassung** (**Overfitting**) bekannt ist (siehe Abbildung 1.5). Das Polynomregressionsmodell ist so flexibel, dass es den einzelnen Trainingsdaten (x_i, y_i) folgen kann. Eine Überangepassung an die Trainingsdaten führt jedoch notwendigerweise zu einer Verschlechterung der Vorhersagegüte bezüglich *neuer* Daten.

1.3 Modellauswahl

Maschinelles Lernen versus Strukturelle Modelle

Das oben veranschaulichte Problem der Überanpassung (Overfitting) ist eng damit verbunden, dass wir hier ein sehr flexibles Regressionsmodell (Polynomregression) betrachten. Viele der möglichen Polynomfunktionen sind unsinnig, da sie nicht die strukturellen Einschränkungen des betrachteten Datenproblems berücksichtigen. Falls ein gesichertes Wissen zu den zugrundeliegenden, strukturellen Zusammenhängen zwischen der Zielvariable Y und den Prädiktorvariablen X existiert, sollte man diese strukturellen Zusammenhängen auch im statistischen Modell berücksichtigen. (Immer mit den Expert*Innen des Faches sprechen!) Im besten Falle gibt es ein **strukturelles Modell** zu den systematischen Zusammenhängen f zwischen Y und X, welches genügend Einschränkungen bietet, sodass alle unsinnigen Modellierungen vermieden werden können. In solchen Idealfällen führt die Minimierung der Trainingsdaten-RSS zu keinem Problem der Überanpassung.

Falls jedoch kein (vertrauenswürdiges) strukturelles Modell vorliegt, ist die Verwendung von sehr flexiblen Regressionsmodellen wie der Polynomregression eine grundsätzlich sehr gute Idee, da wir so, ohne große Einschränkungen, nach den unbekannten richtigen Zusammenhängen f suchen können. Dies ist der Ansatz des **maschinellen Lernens** und die **Polynomregression mit unbekanntem Polynomgrad** p ist lediglich eine von sehr vielen Methoden, welche im Kontext des maschinellen Lernens verwendet werden.

> **Fazit:** Methoden des **maschinellen Lernens** sind typischerweise sehr flexibel und bauen nicht auf strukturellen Modellen auf. Daher benötigen diese Methoden spezielle Verfahren der **Modellauswahl**, um eine Überanpassung an die Trainingsdaten zu vermeiden. Richtig angewandt, können Methoden des maschinellen Lernens unbekannte Zusammenhänge richtig erkennen.

1.3.1 Die Validierungsdaten-Methode

Da die Minimierung der Trainingsdaten-RSS schnell zu einem Problem der Überanpassung führt, benötigen wir eine alternative Methode, um die Güte des

geschätzten Modells zu prüfen. Die einfachste Idee ist dabei die beobachteten Daten in einen Satz von Trainingsdaten

$$\text{Trainingsdaten} = \left\{(x_1^{Train}, y_1^{Train}), (x_2^{Train}, y_2^{Train}), \dots, (x_{n_{Train}}^{Train}, y_{n_{Train}}^{Train})\right\}$$

und einen **separaten** (disjunkten) Satz von Validierungsdaten

$$\text{Validierungsdaten} = \left\{(x_1^{Valid}, y_1^{Valid}), (x_2^{Valid}, y_2^{Valid}), \dots, (x_{n_{Valid}}^{Valid}, y_{n_{Valid}}^{Valid})\right\}$$

zu teilen mit

- $n = n_{Train} + n_{Valid}$
- Trainingsdaten $\cap$ Validierungsdaten $= \emptyset$

Folgender Code-Schnipsel ermöglicht solch eine (zufällige) Aufteilung der Daten in Trainings- und Validierungsdaten:

```
n        <- nrow(Auto_df) # Stichprobenumfang
n_Train  <- 200           # Stichprobenumfang der Trainingsdaten
n_Valid  <- n - n_Train   # Stichprobenumfang der Validierungsdaten

## Index-Mengen zur Auswahl der
## Trainings- und Validierungsdaten
I_Train  <- sample(x = 1:n, size = n_Train, replace = FALSE)
I_Valid  <- c(1:n)[-I_Train]

## Trainingsdaten
Auto_Train_df <- Auto_df[I_Train, ]
## Validierungsdaten
Auto_Valid_df <- Auto_df[I_Valid, ]
```

Obschon die Validierungsdaten-Methode auf alle Regressionsmodelle angewandt werden kann, veranschaulichen wir im Folgenden die Methode anhand der Polynomregression.

Die Aufteilung der Daten in Trainings- und Validierungsdaten ermöglicht uns nun ein zweistufiges Verfahren:

Schritt 1: Mit Hilfe der **Trainingsdaten** wird das Polynomregressionsmodell **geschätzt**:

$$y_i^{Train} = \hat{\beta}_0^{Train} + \hat{\beta}_1^{Train} x_i^{Train} + \hat{\beta}_2^{Train} (x_i^{Train})^2 + \dots + \hat{\beta}_p^{Train} (x_i^{Train})^p + e_i^{Train}$$

Code-Schnipsel Beispiel:

```
Train_polreg <- lm(Verbrauch ~ poly(PS, degree = p, raw=TRUE), data = Auto_Train_df)
```

Schritt 2: Mit Hilfe der **Validierungsdaten** wird das geschätzte Polynomregressionsmodell **validiert**:

$$\hat{y}_i^{Valid} = \hat{\beta}_0 + \hat{\beta}_1^{Train} x_i^{Valid} + \hat{\beta}_2^{Train} (x_i^{Valid})^2 + \cdots + \hat{\beta}_p^{Train} (x_i^{Valid})^p,$$

indem man den **mittleren quadratischen Prädiktionsfehler** (Mean Squared Prediction Error **MSPE**) berechnet:

$$\begin{aligned} \text{MSPE} &= \frac{1}{n_{Valid}} \text{RSS}_{Valid} \\ &= \frac{1}{n_{Valid}} \left((y_1^{Valid} - \hat{y}_1^{Valid})^2 + \cdots + (y_{n_{Valid}}^{Valid} - \hat{y}_{n_{Valid}}^{Valid})^2 \right) \end{aligned}$$

Code-Schnipsel Beispiel:

```
y_fit_Valid   <- predict(Train_polreg, newdata = Auto_Valid_df)
RSS_Valid     <- sum( (Auto_Valid_df$Verbrauch - y_fit_Valid)^2 )
MSPE          <- RSS_Valid / n_Valid
```

Man wiederholt obige Schritte für eine Auswahl von verschiedenen Polynomgraden $p = 1, 2, \ldots, p_{\max}$, z.B. $p_{\max} = 10$, und berechnet für jeden dieser Fälle den MSPE, also:

$$\text{MSPE} \equiv \text{MSPE}(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p, p), \quad \text{für jedes} \quad p = 1, 2, \ldots, p_{\max}$$

Der MSPE ist eine Schätzung des wahren, unbekannten mittleren quadratischen Prädiktionsfehlers $E\left[(Y - \hat{Y})^2\right]$,

$$\text{MSPE}(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p, p) \approx E\left[(Y - \hat{Y})^2\right].$$

Die Minimierung des MSPE über verschiedene Werte des Polynomgrades $p = 1, 2, \ldots$ erlaubt es uns den **reduzierbaren Prädiktions-Fehler** der Polynomregression zu minimieren.

Folgender R-Code verbindet nun alle Schritte und berechnet den MSPE für verschiedene Werte des Polynomgrades p. Dasjenige Modell, welches den MSPE minimiert, ist laut der Daten das beste Prädiktionsmodell.

```
set.seed(31)
##
n        <- nrow(Auto_df) # Stichprobenumfang
n_Train  <- 200           # Stichprobenumfang der Trainingsdaten
n_Valid  <-n - n_Train    # Stichprobenumfang der Validierungsdaten
```

```
## Index-Mengen zur Auswahl der
## Trainings- und Validierungsdaten
I_Train  <- sample(x = 1:n, size = n_Train, replace = FALSE)
I_Valid  <- c(1:n)[-I_Train]

## Trainingsdaten
Auto_Train_df <- Auto_df[I_Train, ]
## Validierungsdaten
Auto_Valid_df <- Auto_df[I_Valid, ]

p_max         <- 6
MSPE          <- numeric(p_max)
fit_plot      <- matrix(NA, 50, p_max)
for(p in 1:p_max){
  ## Schritt 1
  Train_polreg <- lm(Verbrauch ~ poly(PS, degree = p, raw=TRUE),
                     data = Auto_Train_df)
  ## Schritt 2
  y_fit_Valid   <- predict(Train_polreg, newdata = Auto_Valid_df)
  RSS_Valid     <- sum( (Auto_Valid_df$Verbrauch - y_fit_Valid)^2 )
  MSPE[p]       <- RSS_Valid / n_Valid
  ## Daten für's plotten
  fit_plot[,p] <- predict(Train_polreg, newdata = plot_df)
}
```

> **Achtung:** Auch eine Modellauswahl ist fehlerhaft und stellt lediglich eine Schätzung (mit Schätzfehlern) des besten Prädiktionsmodelles innerhalb der betrachteten Klasse von Prädiktionsmodellen (hier Polynomregressionen) dar.

Abbildung 1.7 zeigt jedoch ein Problem der Validierungsdaten-Methode. Die Trainingsdaten und die Validierungsdaten haben kleinere Stichprobenumfänge ($n_{Train} < n$ und $n_{Valid} < n$) was zu einer **erhöhten Schätzgenauigkeit in der MSPE-Schätzung** führt.

1.3.2 k-Fache Kreuzvalidierung

Die k-fache (z.B. $k = 5$ oder $k = 10$) Kreuzvalidierung ist eine Vorgehensweise zur Bewertung der Leistung einer Schätzprozedur (Algorithmus) im Kontext des maschinellen Lernens. Als Schätzprozedur verwenden wir wieder das Beispiel der Polynomregression mit unbekanntem Polynomgrad p, welcher zusammen mit den Modellparametern $\beta_0, \beta_1, \ldots, \beta_p$ aus den Daten erlernt werden muss.

Die k-fache Kreuzvalidierung stellt eine Verbesserung der Validierungsdaten-Methode dar, da sie faktisch die Stichprobenumfänge in den Trainingsdaten und Validierungsdaten erhöht. Wie bei der Validierungsdaten-Methode wird der Datensatz in Trainings- und Validierungsdaten aufgeteilt – jedoch k-fach. Abbildung **??** zeigt ein Beispiel der Datenaufteilung bei der 5-fachen Kreuzvalidierung.

Folgender Code-Schnipsel ermöglicht eine (zufällige) Aufteilung der Daten in k verschiedene Trainings- und Validierungsdaten:

```
n      <- nrow(Auto_df) # Stichprobenumfang
k      <- 5             # 5-fache Kreuzvalidierung

## Index zur Auswahl k verschiedener
## Trainings- und Validierungsdaten:
folds  <- sample(x = 1:k, size = n, replace=TRUE)

## Trainingsdaten im j-ten (j=1,2,...,k) Durchgang
Auto_df[folds != j,]
## Validierungsdaten im j-ten (j=1,2,...,k) Durchgang
Auto_df[folds == j,]
```

Für jede der k Datenaufteilungen wird der MSPE berechnet. Der Mittelwert dieser MSPE-Werte wird häufig als $\text{CV}_{(k)}$ Wert (crossvalidation score) bezeichnet

$$\text{CV}_{(k)} = \frac{1}{k}\sum_{j=1}^{k}\text{MSPE}_j$$

Der $\text{CV}_{(k)}$-Wert stellt eine im Vergleich zur Validierungsdaten-Methode verbesserte Schätzung des unbekannten mittleren quadratischen Pädiktionsfehlers $\text{CV}_{(k)} \approx E[(Y-\hat{Y})^2]$ dar. Die Modellauswahl folgt also auch hier mittels Minimierung des $\text{CV}_{(k)}$-Wertes über die verschiedene Werte des Polynomgrades $p = 1, 2, ...$

> **Wahl von k:** In der Praxis haben sich die Werte $k = 5$ und $k = 10$ etabliert, da diese Größenordnunen einen guten Kompromiss zwischen der Varianz und der Verzerrung des Schätzers $\text{CV}_{(k)}$ für $E[(Y-\hat{Y})^2]$ darstellen.

1.4 Anwendung: Vorhersage des Benzinverbrauchs

Nun haben wir das Werkzeug, um die nicht linearen Zusammenhänge zwischen der **Zielvariable** Y =`Verbrauch` und den **Prädiktorvariablen** G =`Gewicht`,

P =PS und H =Hubraum im Datensatz Auto_df zu berücksichtigen (siehe Abbildung 1.1) und allein mit Hilfe der Daten zu erlernen. Wir folgen hier der Herangehensweise des **maschinellen Lernens** und lassen die **Daten für sich selbst sprechen**.

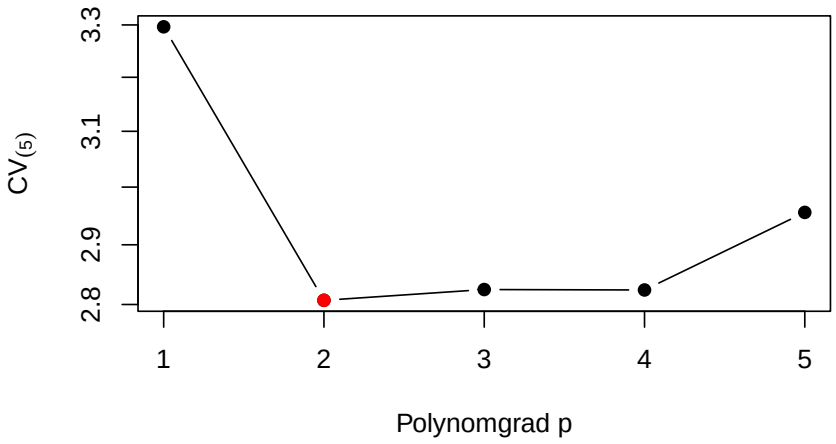
Da Abbildung 1.1 sehr ähnliche Zusammenhänge zwischen der Zielvariable Y =Verbrauch und den Prädiktorvariablen G =Gewicht, P =PS und H =Hubraum vermuten lässt, betrachten wir zunächst ein vereinfachtest Polynomregressionsmodell, bei dem für alle Prädiktorvariablen der gleiche Polynomgrad p verwendet wird.

$$\begin{aligned} Y_i = \beta_0 + \beta_1^G G_i + \beta_2^G G_i^2 + \cdots + \beta_p^G G_i^p + \\ \beta_1^P P_i + \beta_2^P P_i^2 + \cdots + \beta_p^P P_i^p + \\ \beta_1^H H_i + \beta_2^H H_i^2 + \cdots + \beta_p^H H_i^p + \varepsilon_i \end{aligned}$$

Folgender R-Code (Algorithmus) erlernt aus den Daten, mit Hilfe der 5-fachen Kreuzvalidierung $\text{CV}_{(5)} \approx E[(Y - \hat{Y})^2]$, den optimalen Polynomgrad p.

```
set.seed(8)              # Seed für den Zufallsgenerator

n      <- nrow(Auto_df) # Stichprobenumfang
k      <- 5             # 5-fache Kreuzvalidierung
p_max  <- 5             # Maximaler Polynomgrad

folds     <- sample(x = 1:k, size = n, replace=TRUE)

## Container für die MSPE-Werte
## für alle j=1,...,k Kreuzvalidierungen und
## für alle p=1,...,p_max Polynomgrade
MSPE <- matrix(NA, nrow = k, ncol = p_max,
                   dimnames=list(NULL, paste0("p=",1:p_max)))

for(p in 1:p_max){
  for(j in 1:k){
  ## Modelschätzung auf Basis j-ten Traininsdaten Auto_df[folds != j,]
  poly_fit <- lm(Verbrauch ~
                   poly(Gewicht,        degree = p, raw = TRUE) +
                   poly(PS,             degree = p, raw = TRUE) +
                   poly(Hubraum,        degree = p, raw = TRUE),
                 data=Auto_df[folds != j,])
    ## Prädiktion  auf Basis j-ten Validierungsdaten Auto_df[folds == j,]
    pred          <- predict(poly_fit, newdata = Auto_df[folds == j,])
    ##
    MSPE[j,p] <- mean( (Auto_df$Verbrauch[folds==j] - pred)^2 )
```

```
  }
}

## CV-Wert für alle p=1,...,p_max Polynomgrade
CV_k <- colMeans(MSPE)

## Plotten
plot(y = CV_k, x = 1:length(CV_k), pch=21, col="black", bg="black",
     type='b', xlab="Polynomgrad p", ylab=expression(CV[(5)]), log="y")
points(y = CV_k[which.min(CV_k)],
       x = c(1:length(CV_k))[which.min(CV_k)],
       col = "red", bg = "red", pch = 21)
```



Auch der 5-fache Kreuzvalidierungswert $\text{CV}_{(5)}$ ist lediglich eine zufallsbehaftete Schätzung des unbekannten mittleren quadratischen Prädiktionsfehlers $E[(Y - \widehat{Y})^2]$. Um eine Idee von der Präzision und Stabilität der Modellauswahl mittels der Minimierung von $\text{CV}_{(5)}$ zu bekommen, können wir die zufälligen, 5-fachen Aufteilungen der Daten in Trainins- und Validierungsdaten wiederholen und den Effekt alternativer Datenaufteilungen betrachten. Abbildung 1.8 zeigt, dass die Minimierung des Kreuzvalidierungswertes $\text{CV}_{(5)}$ auch in Wiederholungen häufig das Modell mit Polynomgrad $p = 2$ auswählt. Der Polynomgrad $p = 2$ scheint also eine vertauenswürde Modellauswahl darzustellen.

Das Polynomregressionsmodell mit $p = 2$ stellt also ein gutes Prädiktionsmodell dar. Wir verwenden nun dieses Modell, um nach auffälligen Unterschiedenen in den herstellerseitigen Verbrauchsangaben y_i und unseren Prädiktionen zu suchen. Gerade **stark negative Residuen** $y_i - \hat{y}_i$ sind verdächtig, da es auf eine Schönung der Verbrauchsangaben hindeuten könnte.

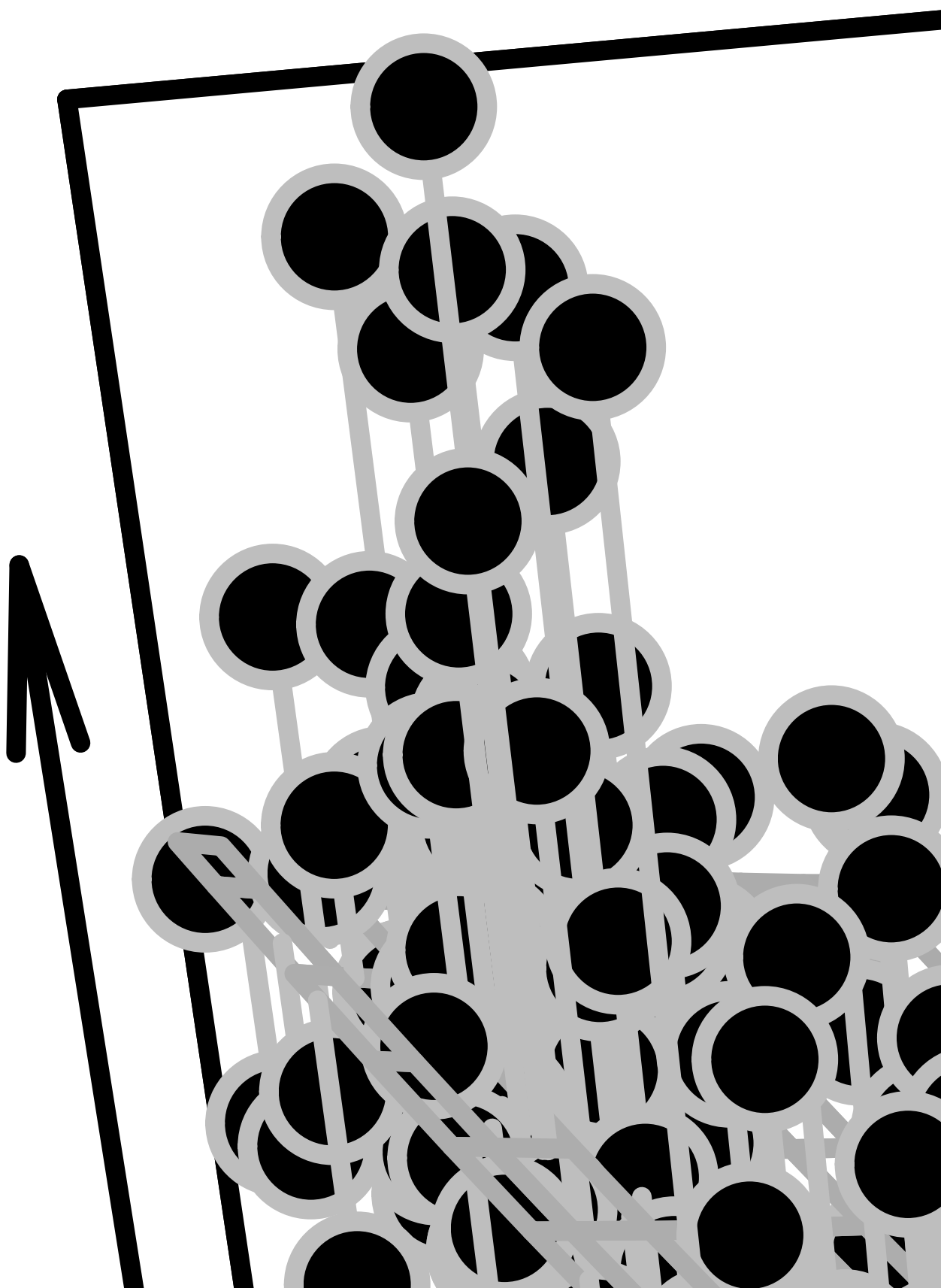
Folgender R-Code schätzt zunächst das Polynomregressionsmodell mit $p = 2$, berechnet dann die Residuen $y_i - \hat{y}_i$ und veranschaulicht die größte negative Abweichung in Abbildung 1.9.

```
p <- 2
poly_fit <- lm(Verbrauch ~
                  poly(Gewicht,        degree = p, raw = TRUE) +
                  poly(PS,             degree = p, raw = TRUE) +
                  poly(Hubraum,        degree = p, raw = TRUE),
                data=Auto_df)

par(mar=c(5.1, 5.1, 4.1, 2.1))
plot(y = resid(poly_fit), x = fitted(poly_fit),
     ylab = expression("Residuen:"~y[i] - hat(y)[i]),
     xlab = expression("Prädiktionen:"~hat(y)[i]),
     main="Größte negative Abweichung der Verbrauchsangabe")
slct <- order(abs(resid(poly_fit)), decreasing = TRUE)[4]
points(y = resid(poly_fit)[slct], x = fitted(poly_fit)[slct],
       col = "red", bg = "red", pch = 21)
text(y = resid(poly_fit)[slct], x = fitted(poly_fit)[slct],
     labels = "Mazda RX-3 (1973)", pos = 2)
```

```
par(mar=c(5.1, 4.1, 4.1, 2.1))
```

Wir haben hier tatsächlich einen besonderen Fall gefunden. Der Mazda RX-3 (1973) (Abbildung **??**) lief mit einem ungewöhnlich sparsamen Wankelmotor. Dieser Motor war so ungewöhnlich, dass es vielerlei Streitigkeiten um die Verbrauchsangaben gab.

1.5 Ende



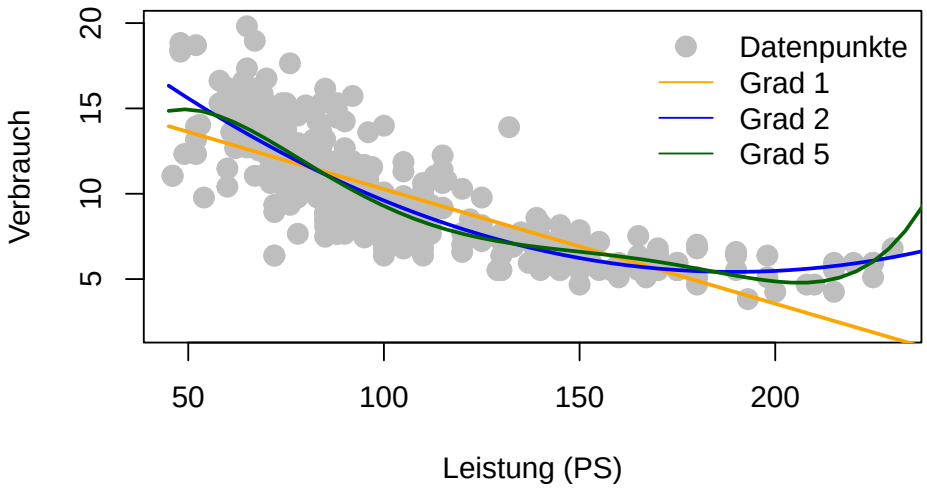


Abbildung 1.4: Polynom Regression bei verschiedenen Polynomgraden p.

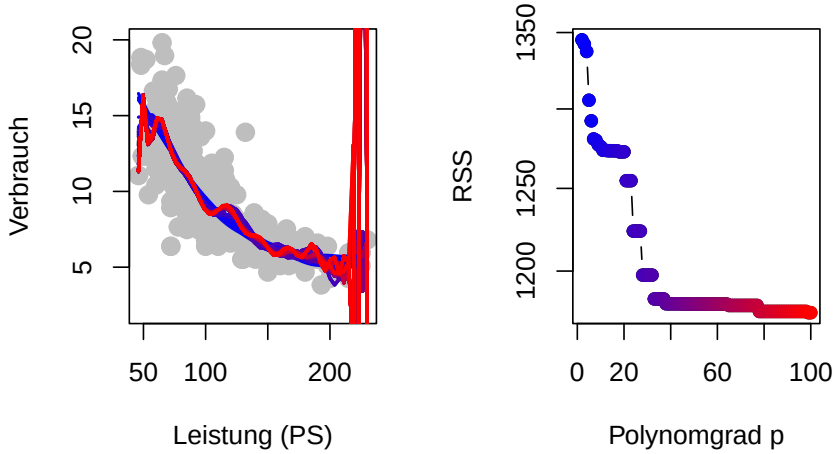


Abbildung 1.5: Polynom Regression und die Wahl des Polynomgrades p durch Minimierung der Trainingsdaten-RSS. (Eine schlechte Idee).

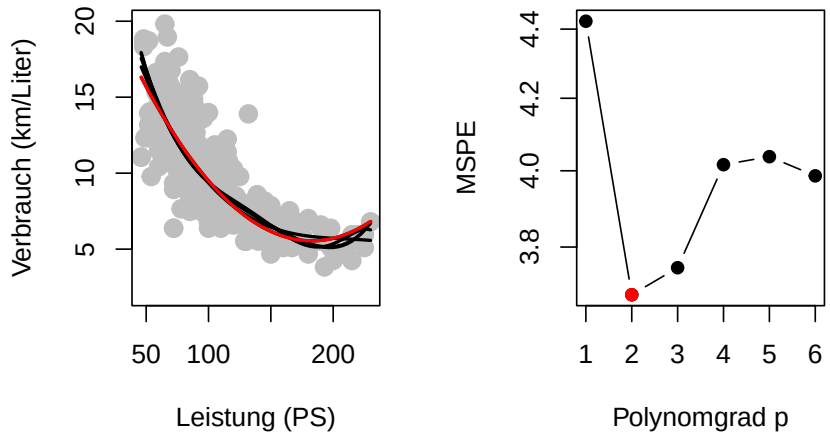


Abbildung 1.6: Polynom Regression und die Wahl des Polynomgrades p durch Minimierung des mittleren quadratischen Prädiktionsfehler MSPE.

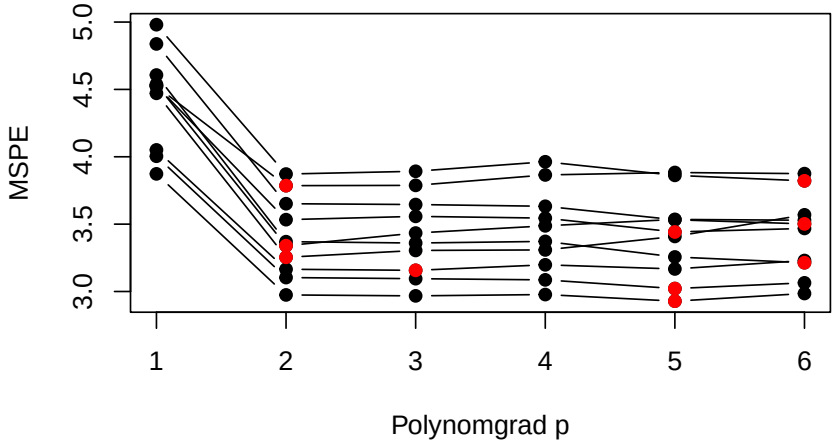


Abbildung 1.7: Zehn verschiedene MSPE-Berechnungen basierend auf zehn verschiedenen, zufälligen Aufteilungen der Daten in Trainings- und Validierungsdaten.

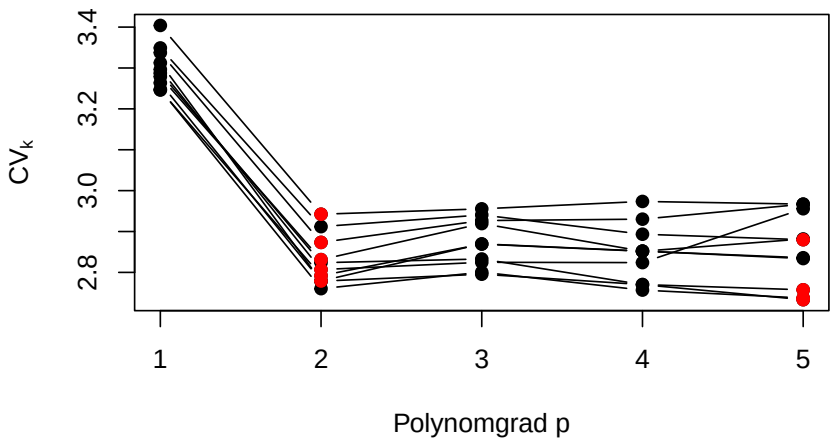


Abbildung 1.8: Zehn verschiedene $CV_{(k)}$-Berechnungen basierend auf zehn verschiedenen, zufälligen Wiederholungen der 5-fachen Kreuzvalidierung.

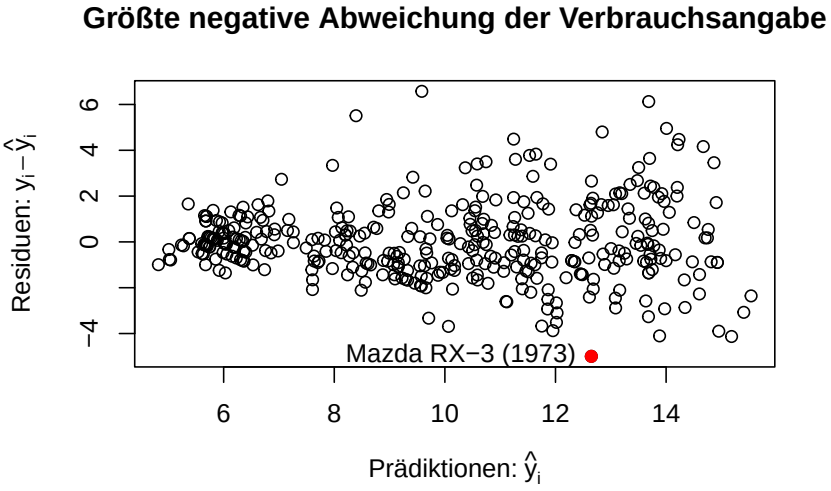


Abbildung 1.9: Polynomregression im Anwendungsbeispiel zum Benzinverbrauch. Die größte negative Abweichung ist der Mazda RX-3 von 1973.

Literaturverzeichnis

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer Science & Business Media.

Hanck, C., Arnold, M., Gerber, A., and Schmelzer, M. (2021). *Introduction to Econometrics with R*. www.econometrics-with-r.org.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2021). *An Introduction to Statistical Learning: With Applications in R*. Springer Science.

Shmueli, G. (2010). To Explain or to Predict? *Statistical Science*, 25(3):289 – 310.