Themenblock Prognosemodelle (Klassifikation und Regression):
02_02Evaluation_Validierung_01

Verfahren der (cross-) validation

Erarbeitet von
Dr. Katja Theune

Lernziele

- Du kannst erklären, was ein Trainings- und ein Testfehler ist
- Du kannst erklären, warum der Testfehler für eine Modellevaluation wichtig ist
- Du kannst verschiedene Verfahren der (cross-) validation erläutern
- Du kannst die Vor- und Nachteile der Verfahren der (cross-) validation erläutern

Projekt ai4all „KI für Alle"
HeiCAD –Heine Center for Artificial Intelligence and Data Science

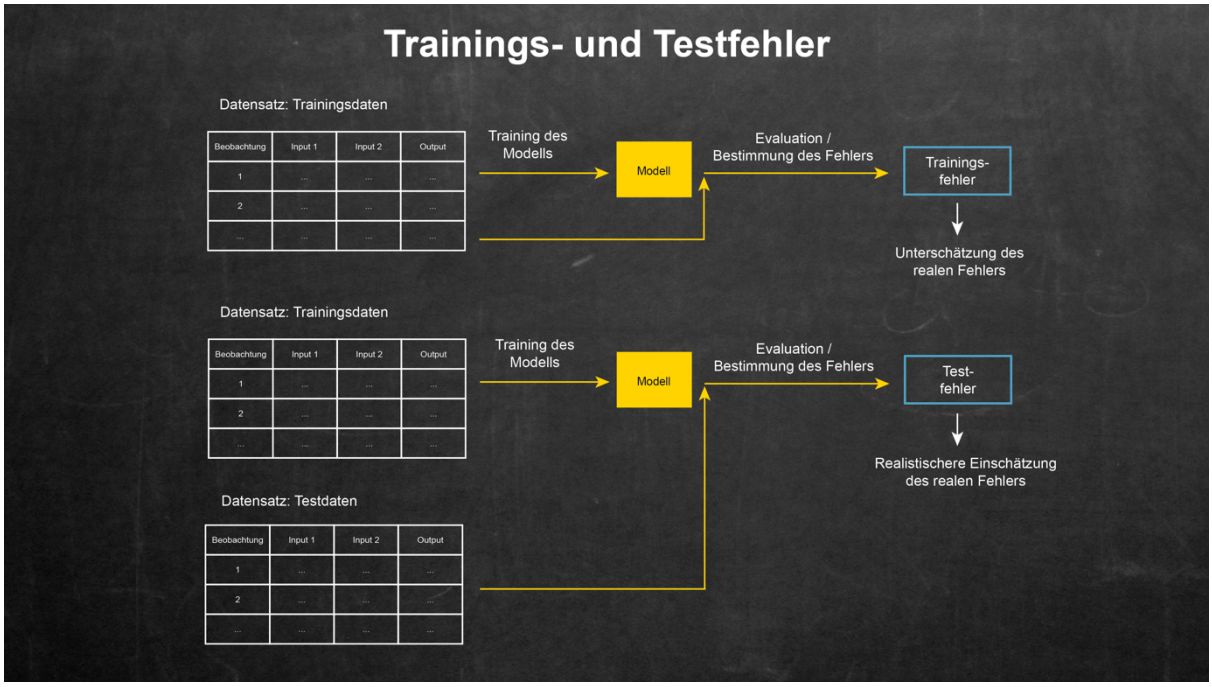
Inhalt

Einstieg

Überwachtes Lernen besteht aus den Schritten des Trainings und der Evaluation. Hier wollen wir uns jetzt intensiver mit dem Konzept der Evaluation beschäftigen. Dazu legen wir den Fokus zunächst auf eine Klassifikation.

Bei der Evaluation wird überprüft, wie gut die Prognose ist bzw. wie hoch der Fehler des Modells ist. Als gute Prognose bzw. kleinen Fehler bezeichnen wir hier zunächst der Einfachheit halber eine häufige Vorhersage des wahren Outputs. Hierfür gibt es aber ganz verschiedene Maße bzw. Metriken.

Trainings- und Testfehler



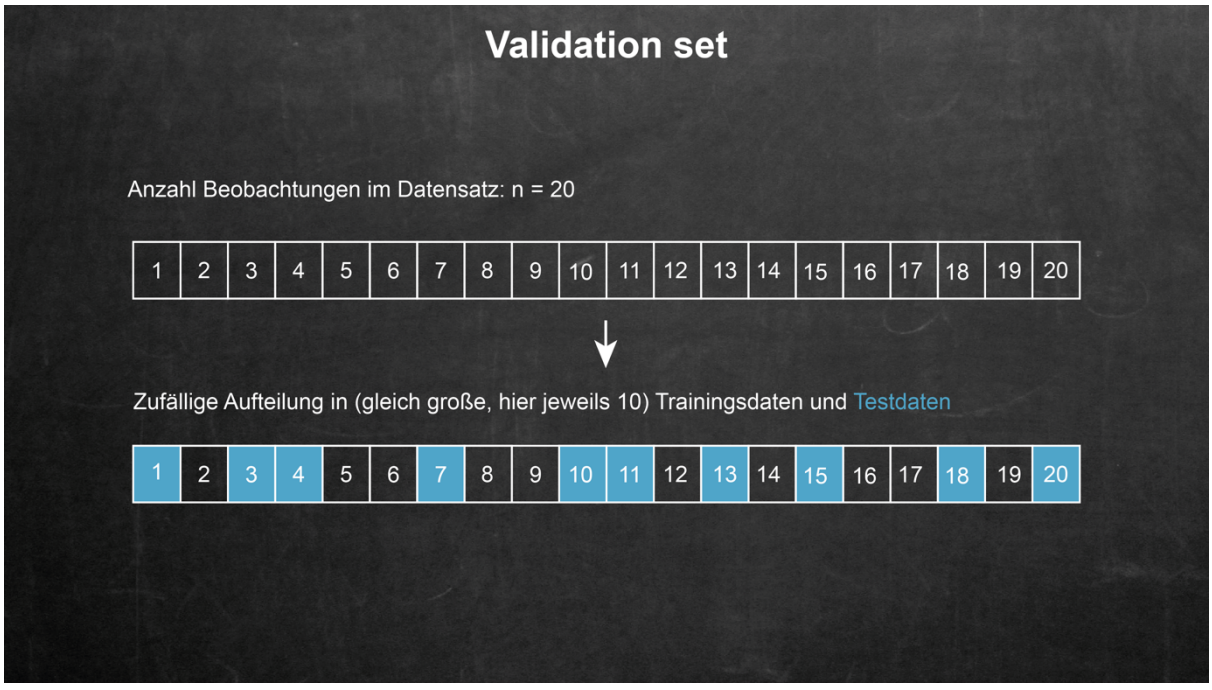
Stellen wir uns nun vor, wir haben unser Modell auf unseren Daten trainiert und wollen nun das Modell auch auf Basis dieser selben Daten evaluieren. Vergleichen wir nun die vom Modell prognostizierten und die wahren Outputs, erhalten wir den sogenannten Trainingsfehler. Allerdings ist dieser Trainingsfehler nicht sehr aussagekräftig, da er viel zu optimistisch ist. Das Modell ist ja sozusagen perfekt auf die Daten angepasst und hat seine Gesetzmäßigkeiten gelernt. Das Stichwort ist hier Überanpassung. Diese Gesetzmäßigkeiten müssen für einen anderen Datensatz aber nicht genau so gelten. Dann wäre der Fehler, der sich ergeben würde, wenn wir unser Modell auf andere Daten anwenden, an die das Modell nicht angepasst wurde, vermutlich größer. Der reale Fehler wird also mit dem Trainingsfehler unterschätzt.

Warum ist das wichtig zu wissen? In der Praxis wollen wir mit dem Prognosemodell den unbekannten Output für neue, zukünftige Beobachtungen möglichst gut prognostizieren. Wir sagen auch, dass das Modell generalisieren soll. Es soll also nicht nur für unsere Daten gut funktionieren, sondern insbesondere auch für neue Daten. Daher ist es wichtig, einen realistischen Eindruck des Fehlers des Modells zu bekommen. Für die Evaluation müssen also neue, bisher vom Modell ungesehene Daten verwendet werden. Also Daten, mit denen das Modell nicht trainiert wurde.

Beziehen wir das mal auf unser Anwendungsszenario. Unsere Daten enthalten verschiedene Inputs und den Output unserer Studierenden, also z. B., ob sie das Studium abgebrochen haben oder nicht. Mit dem auf diesen Daten trainierten Prognosemodell bzw. Frühwarnsystem wollen wir ja dann neue abbruchgefährdete Studierende identifizieren, von denen wir den Output natürlich nicht kennen. Daher muss das trainierte Modell auch für unsere neuen Studierenden gut funktionieren.

Um die Generalisierbarkeit der Ergebnisse auf für das Modell „ungesehene" Daten zu gewährleisten und realistische Fehler unseres Modells abschätzen zu können, brauchen wir also Daten, die wir nur zum Testen, aber nicht zum Trainieren verwenden. Man nennt diese dann Testdaten und den darauf basierenden Fehler Testfehler.

Verfahren der (cross-) validation: validation set

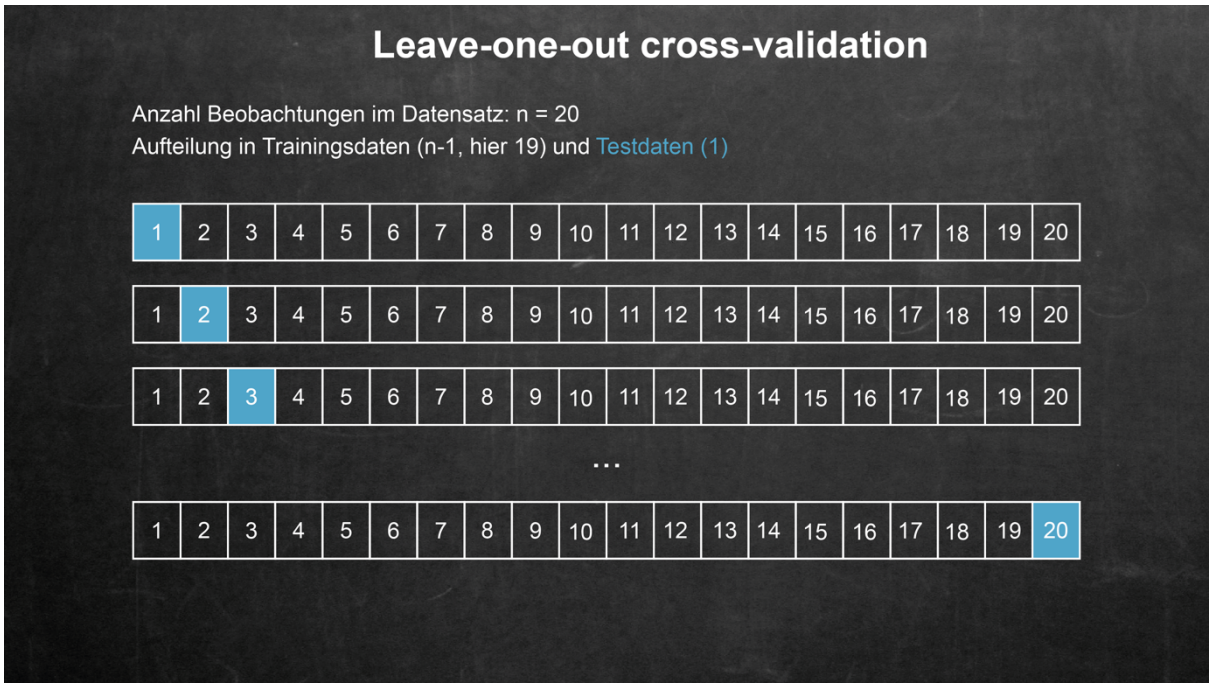


Eine Möglichkeit wäre, die vorliegenden Daten ganz zufällig in Trainingsdaten und Testdaten aufzuteilen. Man nennt das den „validation set" Ansatz.
Als Beispiel betrachten wir mal einen Datensatz mit n = 20 Beobachtungen. Wir sehen hier eine Aufteilung in zwei gleich große Teildatensätze. In Türkis sind diejenigen 10 Beobachtungen, die zufällig in die Testdaten gelangen und mit denen das Modell evaluiert

und der Testfehler bestimmt wird. Die restlichen 10 Beobachtungen kommen in den Trainingsdatensatz. Mit ihnen wird das Modell trainiert.
Ein bedeutender Nachteil ist, dass so nur eine kleine Teilmenge der Daten, von denen wir meistens eh nicht genug haben, für das Training verwendet wird. Statistische Methoden und damit eben auch unsere Modelle tendieren aber dazu, auf kleinen Datenmengen eine schlechtere Leistung zu erbringen. Das führt zu einer Überschätzung des Testfehlers, der somit nicht realistisch ist.

Wir haben also in der Praxis meistens nicht genügend Daten zur Verfügung, um einen Teil ausschließlich zum Trainieren und einen anderen nur zum Testen zu verwenden. Die Idee ist nun, das Training und Testen mehrmals durchzuführen, und zwar mit immer anderen Aufteilungen der Daten in Trainings- und Testdaten. So gelangt jede Beobachtung sowohl mal in die Trainings- als auch mal in die Testdaten. Der Gesamt-Testfehler wird dann über alle Wiederholungen gemittelt und bietet somit eine bessere Aussagekraft bzgl. des wahren Fehlers. Man nennt dies cross-validation oder auf deutsch Kreuz-Validierung. Aber wie könnte das nun genau aussehen?

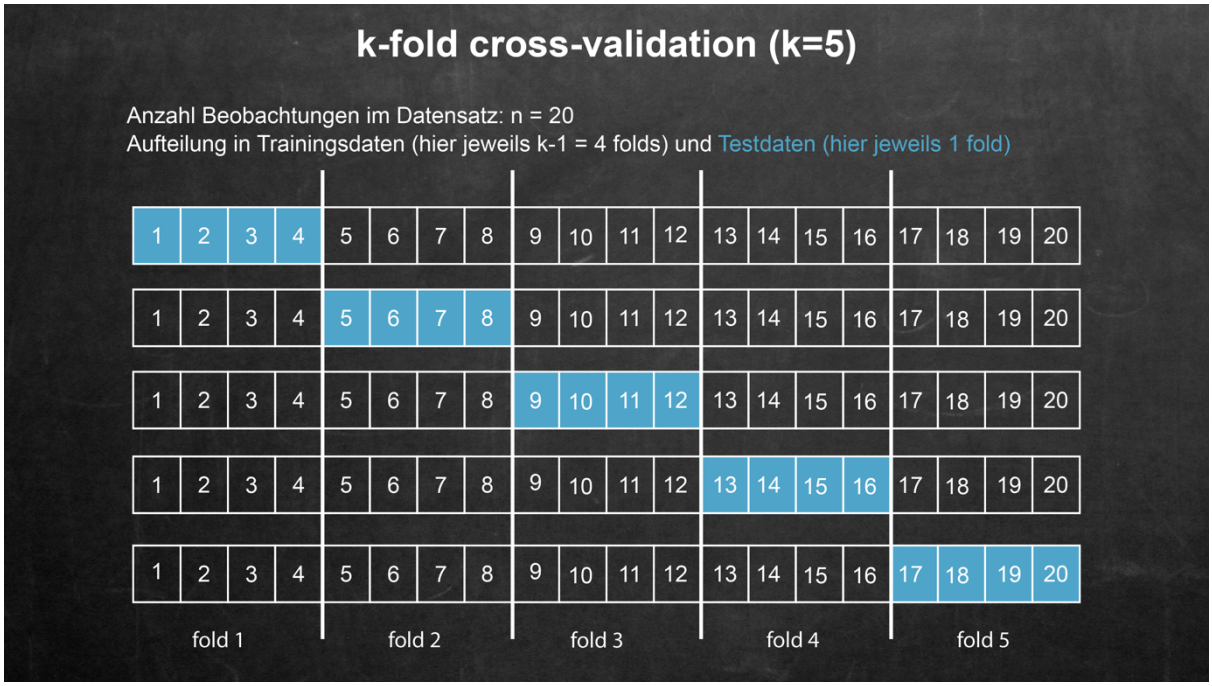
Verfahren der (cross-) validation: leave-one-out cross-validation



Bei der leave-one-out cross-validation erfolgt eine Aufteilung des Datensatzes ebenfalls in zwei Teile. In den Testdaten befindet sich aber nun nur eine einzige Beobachtung, für welche dann der Fehler berechnet wird. In den Trainingsdaten sind dann die verbliebenen n-1 Beobachtungen. Diese Aufteilung und das Training bzw. die Evaluation des Modells wird dann so oft wiederholt, bis jede Beobachtung einmal in der Testmenge war, hier also 20 Mal. Der gesamte Testfehler ergibt sich dann als Mittel über alle 20 Testfehler.

Positiv ist, dass wir hier keine so große Überschätzung des Testfehlers erhalten wie bei dem validation-set, da das Training wiederholt auf n-1 Beobachtungen stattfindet, und damit fast dem gesamten Datensatz.
Wir können uns aber vorstellen, dass bei großen Datensätzen diese Herangehensweise sehr rechenaufwändig wird. Haben wir z. B. einen Datensatz mit 1000 Beobachtungen, müsste also 1000-mal das Modell auf den Trainingsdaten trainiert und 1000-mal der Fehler auf den Testdaten berechnet werden. Daher hat sich in der Praxis eine andere Variante durchgesetzt, die sogenannte k-fold cross-validation.

Verfahren der (cross-) validation: k-fold cross-validation



Bei der k-fold cross-validation werden die Daten in k zufällige, annähernd gleich große, nicht überlappende Teilmengen, die sogenannten folds, unterteilt. Wir sehen hier ein Beispiel für k = 5. Zunächst fungiert der erste fold, bzw. die erste Teilmenge, als Testdaten und die restlichen k-1, hier also 4 Teilmengen als Trainingsdaten. Das Modell wird dann auf diesen 4 Teilmengen trainiert und auf der ersten Teilmenge getestet. Das Ganze wird dann für alle möglichen Aufteilungen in Trainings- und Testdaten, also k mal, wiederholt und der gesamte Testfehler wieder als Mittel aus allen k Testfehlern berechnet. Kurzer Hinweis: für k = n ergibt sich die leave-one-out cross-validation.

Ein bedeutender Vorteil ist, dass diese Methode weniger rechenaufwendig ist, da nicht n-mal das Modell trainiert werden muss, sondern nur k mal. Hier also 5-mal, anstatt 20-mal, wie eben. Ein Nachteil ist, dass die k-fold cross-validation zu einer größeren Überschätzung des Testfehlers tendiert als die leave-one-out cross-validation, da das Modell hier auf kleineren Trainingsmengen trainiert wird. Häufig genutzt wird ein k von 5 oder 10.

Die cross-validation dient nur der generellen Einschätzung der Leistung des Modells. Am Ende wird das Modell final noch einmal auf der gesamten Datenmenge trainiert, da hier dann ein noch kleinerer Fehler zu erwarten ist.

Abschluss

Wir haben den wichtigen Unterschied zwischen Trainings- und Testfehler kennengelernt und wissen, warum nur die Ermittlung des Testfehlers eine realistische Evaluation unseres Prognosemodells ermöglicht. Damit wissen wir auch, dass wir eine Trainings- und eine Testmenge zur Evaluation benötigen. Um das zu gewährleisten, gibt es verschiedene Verfahren, die jeweils Vor- und Nachteile mit sich bringen.

Weiterführendes Material

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques* (3. Auflage). Morgan Kaufmann.

James, G., Witten, D., Hastie, T., & Tibshirani, R., & Tylor, J. (2023). *An Introduction to Statistical Learning - with Applications in Python*. Springer.

Lantz, B. (2015). *Machine learning with R* (2. Auflage). Packt Publishing Ltd, Birmingham.

Disclaimer