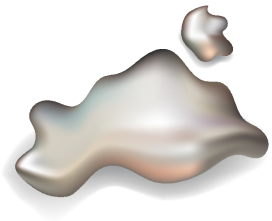

ZENTRUM FÜR
KÜNSTLICHE
INTELLIGENZ IN MV

KI
MV

HOST
Hochschule Stralsund

mlr³ Shiny:
MACHINE LEARNING OHNE PROGRAMMIERKENNTNISSE

Gero Szepannek, Laurens Tetzlaff & Alexander Frahm
Hochschule Stralsund

Whitepaper-Serie des Zentrums für Künstliche Intelligenz in MV
Ausgabe 3

Titelbild Gero Szepannek

1 Predictive Modelling

Verfahren des *Predictive Modelling* haben in der jüngeren Vergangenheit in unterschiedlichste Bereiche der Industrie Einzug gehalten. Hierbei werden datengestützt Prognosemodelle entwickelt, um zukünftige Geschäftsentscheidungen bewerten zu können. Eine der populärsten Anwendungen ist sicherlich das Empfehlungssystem von Amazon; andere verbreitete Anwendungsszenarien bilden Mailingselektion oder Churnprognose im Marketing oder die Kreditausfallprognose (*Credit Scoring*) durch Banken. Alle diese Anwendungen eint, dass die Eintretenswahrscheinlichkeit eines zukünftigen Ereignisses (z.B. Kaufentscheidung: ja/nein) vorhergesagt werden soll. Die hierfür verwendeten Verfahren aus dem Bereich des *Machine Learning* sind dabei die Gleichen.

In den vergangenen Jahren wurden zahlreiche frei verfügbare Softwarelösungen entwickelt, die es Unternehmen ermöglichen, kostengünstig Vorhersagemodelle auf Basis Ihrer eigenen Daten zu erstellen. Dies setzt dabei jedoch in der Regel Programmierkenntnisse voraus. Das hier vorgestellte, an der Hochschule Stralsund entwickelte *mlr3shiny* bietet einen einfachen Einstieg auch ohne Programmierkenntnisse. Dabei baut es auf *mlr3* [LBR+19] auf, einer Weiterentwicklung von *mlr* (Machine Learning in R) [BLK+16], dass erstmalig in [SGB+10] vorgestellt wurde und eines umfangreichsten Frameworks zur Erstellung von Machine-Learning-Modellen darstellt und sämtliche erforderlichen Entwicklungsschritte berücksichtigt.

Im Folgenden werden die erforderlichen Schritte zur Erstellung eines Machine-Learning-Modells in mlr3shiny am Use Case der Entwicklung eines Kreditrisiko-Scoring-Modells erläutert.

2 Installation und Start

Die Installation vor der erstmaligen Nutzung erfordert zunächst eine Installation der statistischen Software R[1]. Es ist üblich (aber nicht notwendig), zusätzlich RStudio[2] zu installieren. Mit dem Kommando

```
install.packages("mlr3shiny")
```

wird anschließend mlr3shiny installiert und kann über den Aufruf

```
mlr3shiny::launchMlr3Shiny()
```

gestartet werden. Abbildung 1 zeigt die Oberfläche nach dem Start. Über die Taskleiste am oberen Rand lassen sich die verschiedenen Modellierungsschritte anwählen. Die einzelnen Schritte werden im folgenden Abschnitt beschrieben. In der oberen rechten Ecke befindet sich ein eingekreistes Fragezeichen. Per Klick auf das Symbol wird ein Hilfefenster zum jeweiligen Schritt geöffnet.

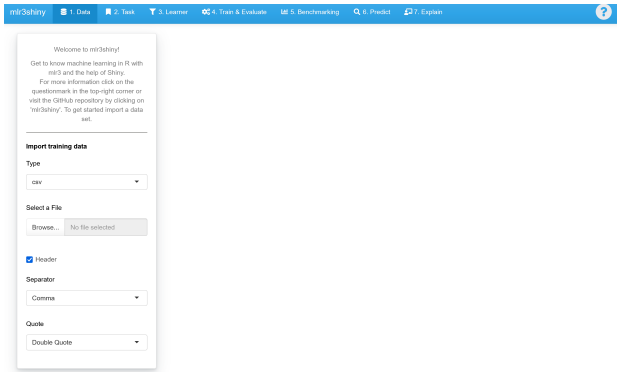

Abbildung 1: Startoberfläche von `mlr3shiny`.

mlr3shiny wird kontinuierlich erweitert und verbessert. Die jeweils neueste Version ist auf Github[3] verfügbar. Diese Version wurde auch für den im Folgenden vorgestellten Use Case verwendet.

[1] `https://cran.r-project.org/`
[2] `https://www.rstudio.com/`
[3] `https://github.com/LamaTe/mlr3shiny`

3 Use Case: Credit Scoring

1. Data Tab Abbildung 1 zeigt die Oberfläche nach dem Start. Über die Tableiste am oberen Rand lassen sich die verschiedenen Prozessschritte zur Erstellung eines Machine-Learning-Modells anwählen. Im ersten Tab können Daten unterschiedlicher Formate eingelesen werden. Für den hier beschriebenen Use Case werden stattdessen Beispieldaten verwendet, die durch das Programm bereitgestellt werden.

2. Task Tab Über *Select Data Backend* lassen sich die „German Credit"-Daten auswählen. Hierbei handelt es sich um frei verfügbare Daten zum Kreditrückzahlverhalten von 1000 Bankkunden[4]. Ziel ist es, die individuelle Kreditausfallwahrscheinlichkeit zu bestimmen. Hierfür muss zunächst unter *Task Processing, Set Positive Class* das Ausfallereignis „bad" ausgewählt werden. Anschließend können per *Drop* Variablen von der weiteren Verwendung ausgeschlossen werden, hier: *installment rate*, *number credits* und *present residence*. Im Fall eigener Daten muss zuvor zusätzlich der Name der zu prognostizierenden Größe (*Target*) festgelegt werden.

3. Learner Tab Im dritten Schritt können bis zu sieben verschiedene *Learner* erstellt werden, die auf den Daten trainiert werden sollen. Hierbei kann zwischen verschiedenen Algorithmen (*lineare und logistische Regression*, *Entscheidungsbaum*, *Random Forest*, *Support Vector Machine* sowie *xgboost*) ausgewählt werden. Im Beispiel soll zu nächst ein Entscheidungsbaum (*Decision Tree*) ausgewählt werden. Über *Go* wird der Learner angelegt und es erscheint ein entsprechender Tab auf der rechten Seite des Fensters. Mit *+Add Learner* kann ein weiterer Learner ergänzt werden.

4. Train & Evaluate Tab Unter *Basic Workflow – Select a new learner* muss zunächst einer der zuvor angelegten Learner ausgewählt werden.

[4] `https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`

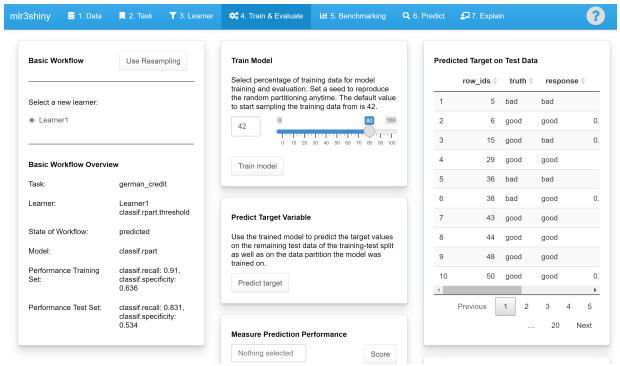

Abbildung 2: Vorhergesagte Kreditausfallwahrscheinlichkeiten.

Anschließend kann dieser anhand der Daten per *Train model* trainiert werden. Der Schieberegler definiert den Prozentsatz an Daten, der zum Erstellen des Modells verwendet wird (siehe Abbildung 2). Der verbleibende (ausgegraute) Prozentsatz wird als sogenanntes *Holdout set* (oder auch *Test set*) zurückgelegt und ermöglicht eine unabhängige Überprüfung der Vorhersagegenauigkeit des Modells, um *Overfitting* zu vermeiden (siehe unten). Per anschließendem Klick auf *Predict Target* werden anhand des trainierten Modells Vorhersagen für die Trainings- und Testdaten erstellt (siehe Abbildung 2, rechts).

Anschließend erfolgt eine Performancebewertung: Unter *Measure Prediction Performance* können unterschiedliche Kennzahlen zur Performancebewertung ausgewählt werden. Im Fall von Kreditrisikovorhersage interessiert insbesondere die *True positive rate* (auch *Recall* bzw. *Sensitivität* genannt): der Anteil durch das Modell erkannten Kreditausfälle. Gleichzeitig sollte jedoch auch der Anteil unter den Kunden möglichst hoch sein, die Ihren Kredit ordnungsgemäß zurückgezahlt haben und laut Modell einen Kredit erhalten sollen. Dieser wird durch die Kennzahl *True negative rate* (auch *Spezifität*) erfasst. Beide Kennzahlen können ausgewählt und per *Score* berechnet werden (Abbildung 2, Mitte unten). Es ist zu erkennen, dass das Modell auf den Trainingsdaten besser abschneidet als auf den Testdaten und die Daten, anhand derer es trainiert wurde, überanpasst (Overfitting, siehe

oben): 91% der korrekt zurückgezahlten Kredite werden richtig vorhergesagt (auf den Testdaten sind es nur 83.1%). Gleichzeitig würden jedoch nur 53.4% der Kreditausfälle prognostiziert – hier besteht noch Verbesserungspotenzial.

5. Benchmarking Tab Der Benchmarking Tab erlaubt den simultanen Performancevergleich mehrerer Modelle. Hierzu wird zunächst im Tab *3. Learner* über *+Add Learner* ein weiterer Learner angelegt (diesmal ein *Random Forest*) und per *Go* bestätigt. Im Benchmarking Tab können nun unter *Create a Benchmark Design* beide Learner ausgewählt und ihre Performance miteinander verglichen werden: Nach Klick auf *Start benchmarking* öffnet sich ein neues Fenster zur Spezifikation der *Resampling Parameter Settings*. Nach Klick auf *Benchmark*, Auswahl der beiden Performancekennzahlen wie im letzten Abschnitt und anschließendem Klick auf *Score* zeigt sich im Fenster rechts oben, dass der Random Forest (*ranger*) auf den Holdout-Daten sowohl hinsichtlich Sensitivität als auch hinsichtlich der Spezifität überlegen ist[5].

Die erzielte Sensitivität (Recall) liegt allerdings unterhalb von 50% – das bedeutet, dass aufgrund des Modells sehr viele Kreditausfälle nicht prognostiziert würden. Im Weiteren soll versucht werden, die Sensitivität des Modells zu erhöhen, ohne dabei zu stark an Spezifität zu verlieren.

Wenn nicht anders spezifiziert, entscheidet das Modell gegen eine Kreditvergabe, wenn die vorhergesagte Wahrscheinlichkeit eines Kreditausfalls größer ist als die eines für einen zurückgezahlten Kredit, d.h. größer als 0.5, ist. Dieser, auch als *Cut off* bezeichnete, Wert lässt sich verändern: Legt man im Learner Tab einen Learner an (siehe oben) öffnet sich rechts ein Fenster das die Spezifikation so genannter *Hyperparameter* ermöglicht (vergleiche Abbildung 3). Über den Parameter *threshold* lässt sich der Cut off verändern.

Werden zwei weitere Random Forests mit

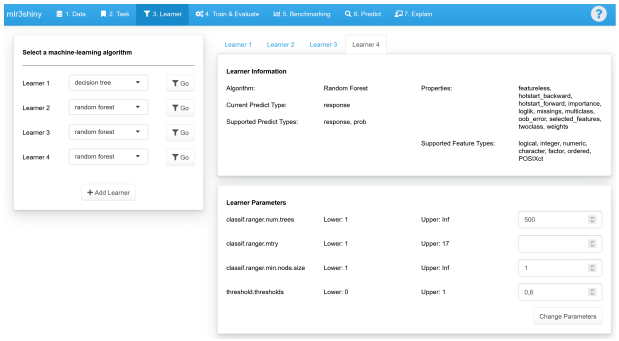

Abbildung 3: Maske zur Spezifikation von Hyperparametern.

einem geänderten threshold auf 0.45 bzw. 0.4 angelegt, lässt sich ein erneuter Performancebenchmark durchführen: Senkt man den threshold auf 0.45, so steigt die Sensitivität von 47% auf 58% – gleichzeitig sinkt die Spezifität von 90.1% auf 85.8%. Durch eine weitere Änderung des thresholds auf 0.4 erhöht sich die Sensitivität weiter auf 63%, die Spezifität sinkt dabei gleichzeitig auf einen Wert unter 80%. Wenn, wie in diesem Fall, mehrere Kennzahlen gleichzeitig betrachtet werden, gibt es keine allgemeine Regel für die Auswahl des besten Modells. Vielmehr sollten geschäftspolitische Erwägungen in die Wahl des Modells einbezogen werden[6].

6. Predict Tab Die beiden vorangegangenen Schritte dienten im Wesentlichen dazu, ein Modell auszuwählen und einen Eindruck von dessen Vorhersagekraft zu gewinnen. Hat man sich abschließend für ein Modell entschieden (hier dasjenige mit einem threshold von 0.45) sollte dies mit den entsprechenden Daten noch einmal auf der Gesamtmenge an Daten (Training + Holdout) neu trainiert werden, da davon auszugehen ist, dass die Performance hierdurch noch einmal (leicht) verbessert werden kann. Dies erfolgt durch dessen Auswahl im „*Apply best learner on new data*“-Fenster und einen Klick auf *Train Learner*.

[5] Durch die abweichende zufällige Aufteilung der Daten ergeben sich geringfügig andere Ergebnisse im Vergleich zum vergangenen Abschnitt.

[6] Neben dem *threshold* gibt es weitere, algorithmusspezifische Hyperparameter zur Modelloptimierung. Für diese sei auf die entsprechende Dokumentation verwiesen.

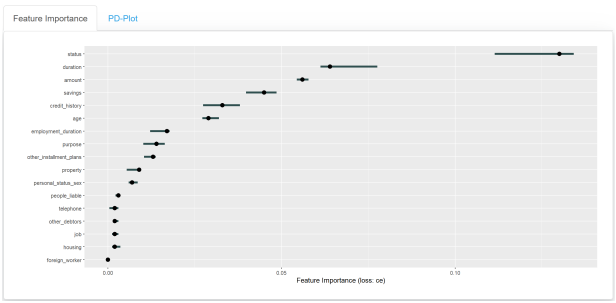

Abbildung 4: Feature importance plot.

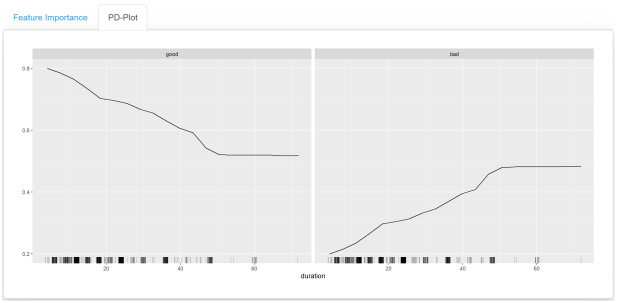

Abbildung 5: Partial dependence plot für die Variable duration.

Das so erstellte Modell lässt sich nun zur Vorhersage neuer Daten verwenden (Fenster *Import a new dataset*). Über *Export Learner* kann das Modell weiterhin exportiert werden, um es auch außerhalb von mlr3shiny zu verwenden. Über *Show the Code* lässt sich zudem der im Hintergrund generierte R-Code anzeigen, um diesen ggf. weiter zu verwenden.

7. Explain Tab Der Explain Tab ermöglicht, mit Hilfe zweier Verfahren aus dem Bereich der *Explainable artificial intelligence*, einen Eindruck von der Wirkungsweise des Modells zu gewinnen: Nach Auswahl des (zuvor im Predict Tab trainierten) Learners muss zunächst über *Select loss function for Feature Importance* eine Kennzahl zur Bestimmung der Variablenwichtigkeit (hier der Klassifikationsfehler *ce*) ausgewählt werden, sowie unter *Select Features to show in PD-Plot* Variablen, deren Effekt auf die Vorhersage analysiert werden soll (hier: *duration*).

Die Durchführung per *Start evaluating* ist sehr rechenintensiv und nimmt etwas Zeit in Anspruch. Als Ergebnis erhält man zwei Grafiken: Der *Feature importance plot* (Abbildung 4) zeigt die Bedeutung der Variablen für die Vorhersagegüte des Modells: am Wichtigsten sind die Variablen *status*, *duration* sowie *amount*.

Hinter dem Reiter *PD-Plot* (Abbildung 5) verbergen sich *Partial dependence plots*, die den (durchschnittlichen) Effekt einer Variablen auf die Vorhersage des Modells für alle Klassen beschreiben: in der rechten Grafik lässt sich gut erkennen, dass das vorhergesagte Kreditausfallrisiko mit wachsender Kreditlaufzeit ansteigt.

4 Zusammenfassung

Das frei verfügbare Open Source Tool mlr3shiny erlaubt die Entwicklung von Machine-Learning-Modellen in kurzer Zeit mit wenigen Mausklicks und ohne Programmierkenntnisse. Es basiert auf der Datenanalysesoftware R und dem State-of-the-Art-Framework mlr3 und umfasst die Analyse der Vorhersagegüte, Modelloptimierung sowie Verfahren der explainable AI. Neben der Anwendung zur Vorhersage neuer Daten ist auch der Export von Modellen oder dem zugrunde liegenden R-Code möglich.

Literatur

[BLK+16] Bischl, Bernd, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio und Zachary M. Jones: *mlr: Machine Learning in R*. Journal of Machine Learning Research, 17(170):1–5, 2016.

[LBR+19] Lang, Michel, Martin Binder, Jakob Richter, Patrick Schratz, Florian Pfisterer, Stefan Coors, Quay Au, Giuseppe Casalicchio, Lars Kotthoff und Bernd Bischl: *mlr3: A modern object-oriented machine learning framework in R*. Journal of Open Source Software, 2019.

[SGB+10] Szepannek, Gero, Matthias Gruhne, Bernd Bischl, Sebastian Krey, Tamas Harczos, Frank Klefenz, Christian Dittmar und Claus Weihs: *Perceptually Based Phoneme Recognition in Popular Music*. In: Locarek-Junge, H. und C. Weihs (Herausgeber): *Classification as a Tool for Research*, Seiten 751–758. Springer, 2010.

Kontakt KIMV

Dr.-Ing. Anne Gutschmidt
Zentrum für Künstliche Intelligenz in MV
Albert-Einstein-Straße 21
18059 Rostock
anne.gutschmidt@uni-rostock.de

Das Zentrum für Künstliche Intelligenz wird vom Wirtschaftsministerium des Landes MV und dem Europäischen Fonds für regionale Entwicklung (EFRE) gefördert.

EUROPÄISCHE UNION
Europäischer Fonds für regionale Entwicklung

MV tut gut.

Kontakt zu den Autoren

Prof. Dr. Gero Szepannek
Hochschule Stralsund
Zur Schwedenschanze 15
18435 Stralsund
gero.szepannek@hochschule-stralsund.de