Shapley Values Masterarbeit

Stephan Baartz

2023-08-29

Shapley Values Masterarbeit

Diese Arbeit untersucht die Anwendung von Shapley Values – einem Konzept aus der kooperativen Spieltheorie – zur Interpretation und Erklärung von Vorhersagen dreier gängiger maschineller Lernmodelle: Logistische Regression, XGBoost und Neuronale Netze mittels Keras/TensorFlow. Die Analyse der Shapley Values dieser Modelle vermittelt ein tieferes Verständnis der jeweiligen Vorhersageentscheidungen und bietet eine vergleichende Perspektive hinsichtlich ihrer Erklärbarkeit.

Inhaltsverzeichnis

- Abhängigkeiten
- Installation
- Aufbau des R-Dokuments
- Modellierung
- Shapley Values

Abhängigkeiten

Dieses Projekt verwendet die folgenden R-Pakete:

- fastDummies
- tidyverse
- kableExtra
- visNetwork
- keras
- tensorflow
- gridExtra
- ggplot2
- xgboost
- SHAPforxgboost

```
install.packages(
  c("fastDummies",
    "tidyverse",
    "kableExtra",
    "visNetwork",
    "keras",
    "tensorflow",
    "gridExtra",
    "ggplot2",
    "xgboost")
  )
# Für SHAPforxgboost:
```

```r
# Installieren Sie zuerst devtools, wenn Sie es noch nicht haben
# install.packages("devtools")
devtools::install_github("liuyanguu/SHAPforxgboost")
```

Versionen

Verwendete R-Version:

```r
R.version.string
```

```
## [1] "R version 4.2.2 (2022-10-31 ucrt)"
```

Verwendete RStudio-Version:

```r
rstudioapi::versionInfo()$long_version
```

```
## [1] "2023.06.1+524"
```

Versionen der verwendeten Pakete:

```r
packages <- c("fastDummies",
    "tidyverse",
    "kableExtra",
    "visNetwork",
    "keras",
    "tensorflow",
    "gridExtra",
    "ggplot2",
    "xgboost")

versions <- sapply(packages, function(pkg) {
  as.character(packageVersion(pkg))
})

data <- data.frame( Version = versions)

kable(data, format = "pipe", col.names = "Version")
```

	Version
fastDummies	1.7.3
tidyverse	2.0.0
kableExtra	1.3.4
visNetwork	2.1.2
keras	2.13.0
tensorflow	2.13.0
gridExtra	2.3
ggplot2	3.4.3
xgboost	1.7.5.1

Aufbau des R-Dokuments

Zu Beginn werden die Daten geladen und mittels Dummy-Codierung umgewandelt. Anschließend erfolgt die Aufteilung in Trainings- und Testdaten, und die Daten werden auf ein Intervall von 0 bis 1 normiert.

Modellierung

logistische Regression

Die Modellierung der logistischen Regression beinhaltet die manuelle Berechnung der Beta-Koeffizienten und die Erstellung eines Netzwerkdiagramms. Zudem wird eine Konfusionsplot-Funktion definiert.

Neuronaler Netze

Die Modellierung des neuronalen Netzes umfasst eine Netzgrafik, die mittels visNetwork erstellt wurde. Zudem werden die Lossfunktion und die Genauigkeit der Validierungsdaten analysiert.

Modellierung XgBoost

Der Trainingsprozess des Modells wird analysiert. Anstelle einer Netzgrafik wird der plot_tree verwendet, um einen der modellierten Bäume darzustellen.

Shapley Values

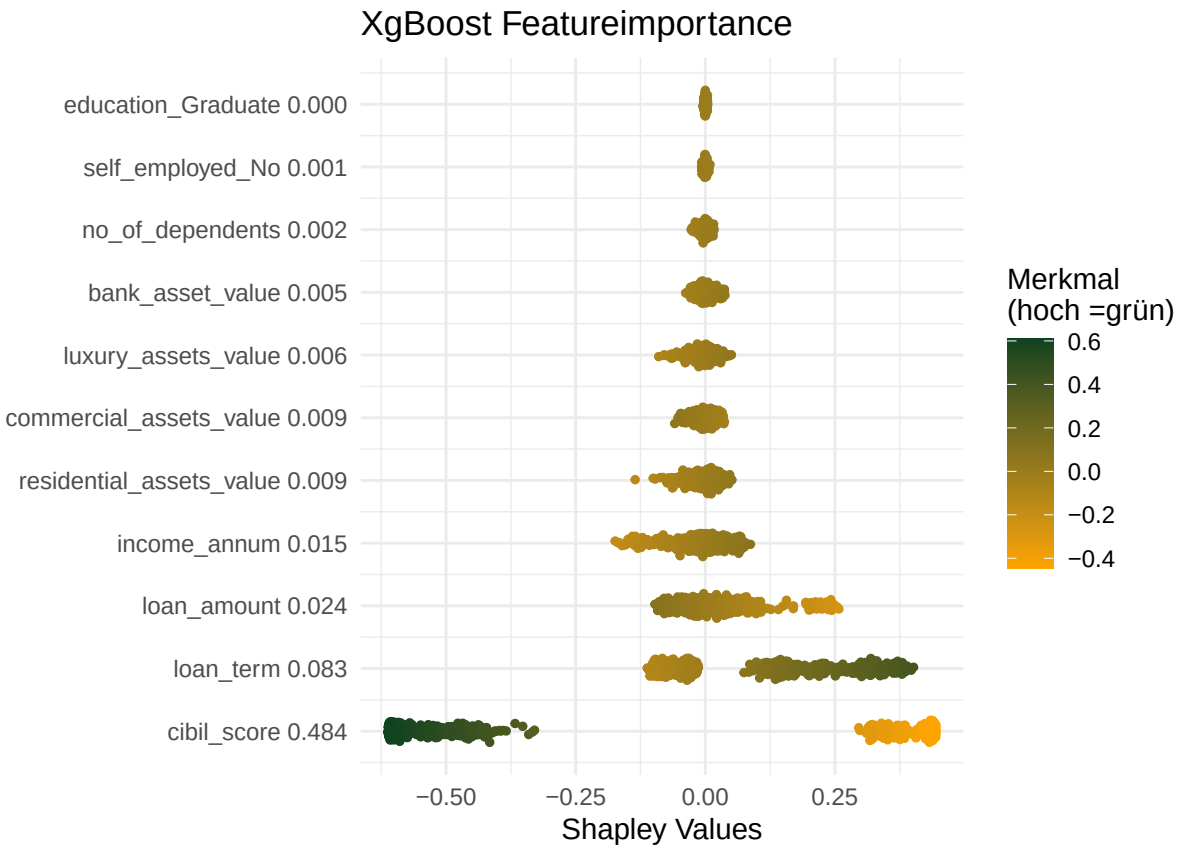
Für das xgBoost Modell wird über das Package der BIAS mitgeliefert. Für die anderen beiden Modelle wird der Bias aus den Daten ermittelt. Nachdem alle drei Modelle die gleiche Struktur in den Shapley Values aufweisen, wurde deren Summe mit den Prognosen vergleichen.

Für das xgBoost-Modell wird der Bias direkt über das zugehörige Paket bereitgestellt. Bei den anderen beiden Modellen, der logistischen Regression und dem neuronalen Netz, wird der Bias direkt aus den Daten abgeleitet, sodass alle drei Modelle eine konsistente Struktur in ihren Shapley Values aufweisen.

Featureimportance Plot

Die Funktion `featureImportance.Plot` wurde entwickelt, um die Bedeutung von insgesamt 11 Merkmalen zu berechnen und zu visualisieren. Hier ist ein Beispiel für xgBoost:

```
featureImportance.Plot(
  data = shap.train.xg,
  titel = "XgBoost Featureimportance")
```



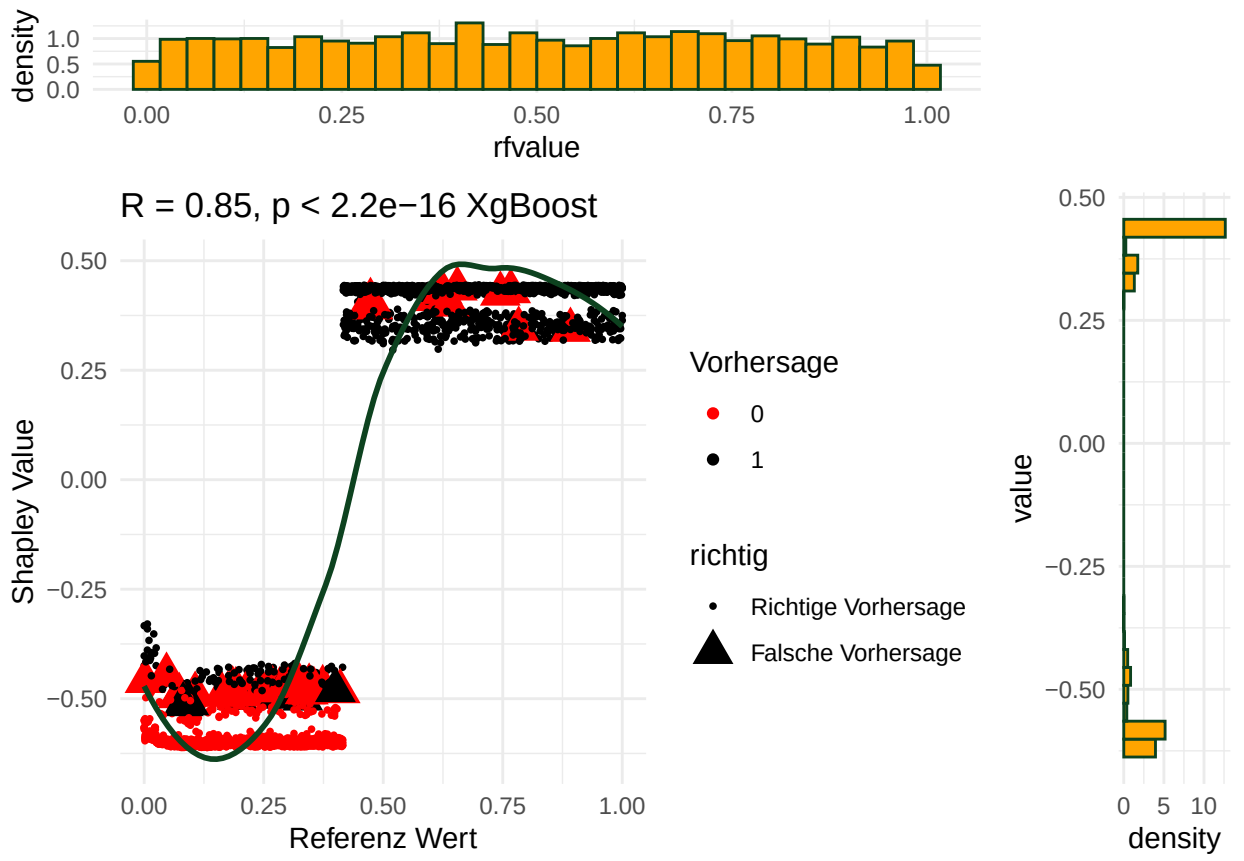
Ähnliche Darstellungen können auch für die anderen Modelle erstellt werden:

```
featureImportance.Plot(
  data = shap.train.lr,
  titel = "Logistische Regression Featureimportance")
featureImportance.Plot(
  data = shap.train.nn,
  titel = "Neuronales Netz Featureimportance")
```

Abhängigkeitsplot

Mit der Funktion `depend.cibil.plot` können die Shapley Values eines bestimmten Merkmals gegen die Referenzdaten geplottet werden. Zusätzlich werden Histogramme des jeweiligen Merkmals, die Korrelation und ein Hypothesentest durchgeführt. Hier ein Beispiel für xgBoost:

```
depend.cibil.plot(shap.train.xg,pred.round.train.xg,"cibil_score","XgBoost")
```



Weitere Beispiele:

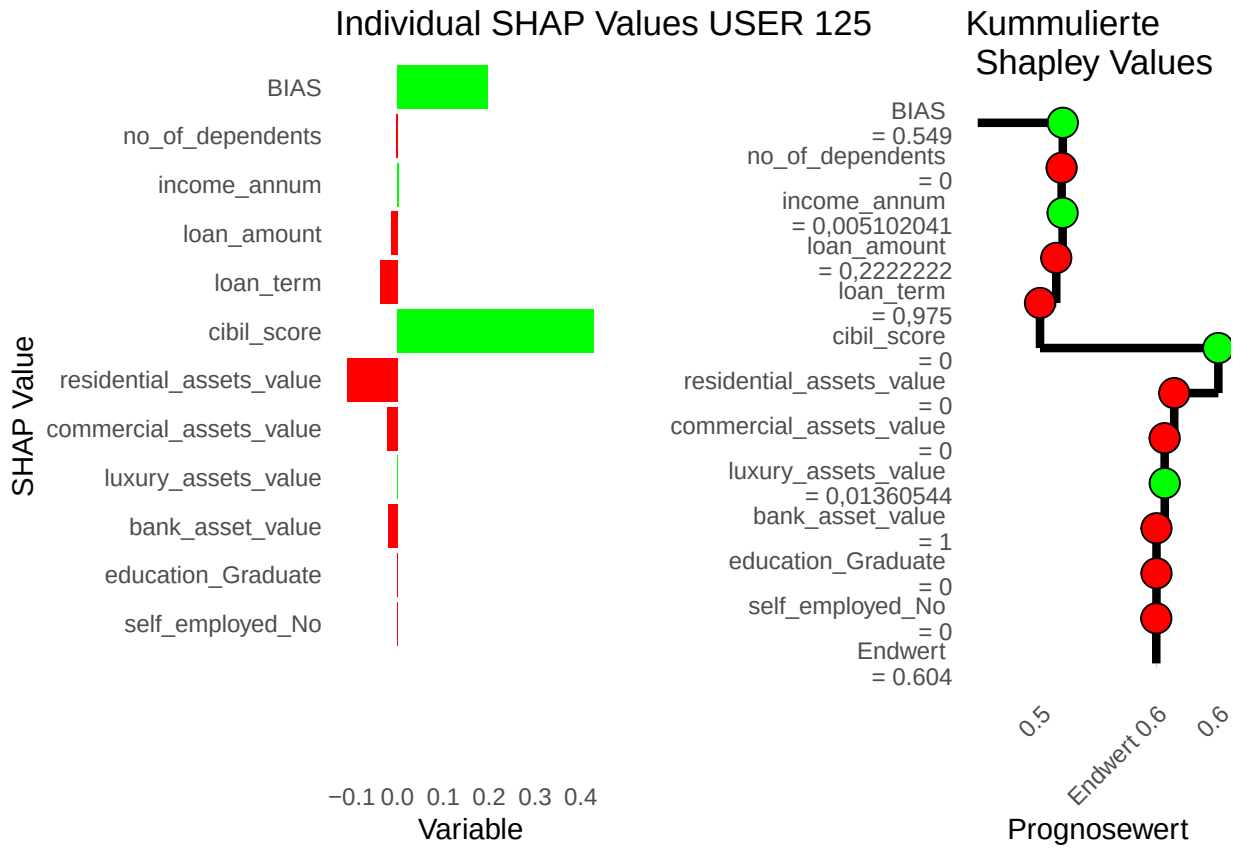
```
depend.cibil.plot(shap.train.lr,pred.round.train.lr,"cibil_score","Logistische Regression")
depend.cibil.plot(shap.train.nn,pred.round.train.nn,"cibil_score","Neuronales Netz")
```

Individueller Plot

Die Funktion `waterBalken` ermöglicht die detaillierte Analyse der Shapley Values für einen einzelnen Benutzer. Sie stellt die Werte in Form von Balken- und Wasserfall-Diagrammen dar und bezieht sich dabei auf die ursprünglichen Daten. Hier ein Beispiel für xgBoost:

```
waterBalken(shapData = shap.test.xg,id = 125,referenzData = norm.test)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Weitere Beispiele:

```
waterBalken(shapData = shap.test.lr,id = 125,referenzData = norm.test)
waterBalken(shapData = shap.test.nn,id = 125,referenzData = norm.test)
```