6. Benchmarks und Anwendungsbeispiele

We are happy to share our materials openly:
The content of these Open Educational Resources by Lehrstuhl für Psychologische Methodenlehre und Diagnostik, Ludwig-Maximilians-Universität München is licensed under CC BY-SA 4.0. The CC Attribution-ShareAlike 4.0 International license means that you can reuse or transform the content of our materials for any purpose as long as you cite our original materials and share your derivatives under the same license.

Agenda heute

- Benchmarking
- 4 Beispiele (2 Regression, 2 Klassifikation)
  - Benchmark
  - Ergebnisse
  - Variablenwichtigkeit
- Generelle Strategien beim Resampling

Prädiktive Modellierung als Experimenteller Prozess

- Ziel prädiktiver Modellierung: Möglichst präzise Vorhersage einer Kriteriumsvariable, basierend auf einer Reihe von Prädiktorvariablen
- Um dieses Ziel zu erreichen, stehen unterschiedliche Modellklassen zur Verfügung (z.B. lineare vs. nonlineare Modellklassen)
- Da man nie sicher weiß, welche Modellklasse für eine konkrete Fragestellung die präzisesten Vorhersagen liefert, sollten prinzipiell immer mehrere prädiktive Modelle ausprobiert und hinsichtlich ihrer geschätzten Vorhersagegüte miteinander verglichen werden → **Benchmarking**
- Damit ähnelt prädiktive Modellierung einem wissenschaftlichen Experiment:
  - Vergleich verschiedener experimenteller Bedingungen (hier: Modellklassen)
  - Für jede Bedingung wird die optimale Operationalisierung identifiziert (z.B. Anzahl von Prädiktorvariablen, Hyperparameter-Einstellungen)
  - Gleiche äußere Bedingungen (z.B. Performancemaße, Resampling-Strategie), um die experimentellen Bedingungen besser vergleichen zu können

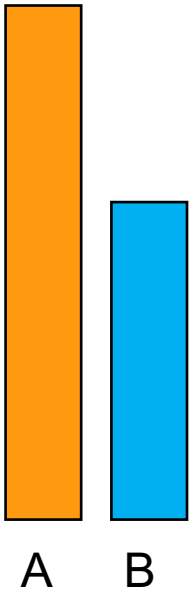
Benchmarking Details

- Erhöhung der Vergleichbarkeit:
  - Um die Vorhersagegüte der unterschiedlichen prädiktiven Modelklassen bestmöglich vergleichen zu können, werden alle Modellklassen auf den gleichen Cross-Validation Folds trainiert.
    → Alle Modellklassen „sehen“ genau die gleichen Trainings- und Testsets
- Interpretation der Ergebnisse:
  - Ergebnis eines Benchmarks ist ein Schätzwert für den erwarteten Vorhersagefehler jeder der untersuchten Modellklassen. Dabei sollte man beachten, dass diese Schätzwerte lediglich Punktschätzer sind.
  - Geht es darum zu entscheiden, welches der prädiktiven Modelle man in der praktischen Anwendung konkret einsetzen soll, möchte man auch die Varianz des Punktschätzers mit berücksichtigen.
  - Leider ist es nicht möglich, für den Cross-Validation Schätzer exakte Konfidenzintervalle oder Hypothesentests abzuleiten.
    → Betrachte deskriptiv die Varianz der Schätzwerte aus den einzelnen Folds!

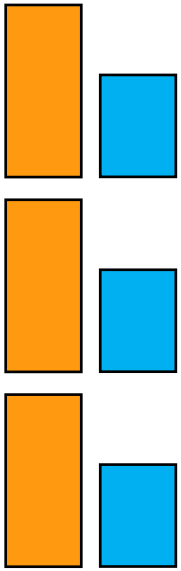
Stratifizierung bei Klassifikation

Für Cross-Validation bei Klassifikation wird typischerweise der Anteil der Klassen innerhalb der Folds an den Anteil im Gesamtdatensatz angepasst.

Gesamtdatensatz



3 - Fold Cross-Validation



$MMCE_{test}^{(1)}$

$MMCE_{test}^{(2)}$

$MMCE_{test}^{(3)}$

$$MMCE_{CV} = \frac{1}{3}\left(MMCE_{test}^{(1)} + MMCE_{test}^{(2)} + MMCE_{test}^{(3)}\right)$$

Stratifizierung schadet nie (kann man also immer machen);
Besonders sinnvoll bei:

- kleinen Stichproben
- unbalancierten Klassen

Idee:

- Präzisere Abschätzung des erwarteten Vorhersage-fehlers innerhalb der Folds
- Verhindern, dass die kleinere Klasse durch die zufällige Aufteilung in einem Fold zu stark unterrepräsentiert ist (schlechte Vorhersagegüte)

Repeated Cross-Validation

- Bei kleinen Stichproben kann sich der Cross-Validation Schätzwert des erwarteten Vorhersagefehlers relativ stark ändern, wenn eine andere zufällige Aufteilung des Datensatzes in die *K* Folds gewählt wird.
- Problem beim Benchmarking, falls sich bei Wiederholung des Resamplings plötzlich die Rangreihe der verschiedenen prädiktiven Modelle ändert ...
- **Lösung:** „Repeated Cross-Validation“
  - Wiederhole *K*–Fold Cross-Validation *W* mal
  - Für jede Wiederholung *W*, wähle eine neue zufällige Aufteilung des Gesamtdatensatzes in *K* Folds
  - Berechne den Mittelwert der *W* Cross-Validation Schätzwerte:

$$MSE_{RepCV} = \frac{1}{W}\sum_{w=1}^{W} MSE_{CV}^{(w)}$$

- Je mehr Wiederholungen *W*, desto „stabiler“ wird der Repeated Cross-Validation Schätzer (In der Praxis wird häufig *K* = 5 oder 10 und *W* = 10 verwendet).

Dummy-Modell als Baselinemodell

- Die meisten Performancemaße (z.B. *MSE*, *MMCE*) können schlecht absolut interpretiert werden, z.B. ist es nicht offensichtlich, ob ein prädiktives Modell eine höhere Vorhersagegüte erzielt als ein triviales Modell, das alle Prädiktoren ignoriert und nur „schlau rät".
  - Beispiel: Wenn in der Population die Punktprävalenz einer Erkrankung 1% beträgt, dann kann ein erwarteter *MMCE* von 0.01 (und analog eine *Accuracy* von 1 – *MMCE* = 99%) bereits dadurch erzielt werden, dass alle Personen automatisch als gesund klassifiziert werden
- **Idee:** Vergleiche ein prädiktives Modell immer mit einem „Dummy-Modell", dessen Vorhersagegüte in jedem Fall übertroffen werden sollte
  - Regression:
    Konstante Vorhersage des mittleren Kriteriumswerts im Trainingsdatensatz
  - Klassifikation:
    Konstante Vorhersage des häufigsten Kriteriumswerts im Trainingsdatensatz
- Die Vorhersagegüte des Dummy-Modells wird im Benchmark mit der gleichen Resampling-Strategie evaluiert wie die anderen prädiktiven Modelle.

Exkurs: Machine Learning in R

- Verwendung des „mlr3“ Pakets in R:

  - Standardisiertes Interface zur Verwendung (fast) aller in R implementierten prädiktiven Modellklassen („Machine Learning Verfahren“)
  - Einfache Durchführung der einzelnen Analyseschritte:
    z.B. Training, Vorhersagen, Evaluation mit Resampling

- Link zum mlr3 e-Book:
  - https://mlr3book.mlr-org.com
- Link zur mlr3 Homepage:
  - https://mlr3.mlr-org.com

Anwendung Benchmarks

- Es werden Benchmarks für die folgenden Beispiele betrachtet: Boston Housing, Titanic, Depression, PhoneStudy
- Die Benchmarks sind alle gleich aufgebaut:
  - Prädiktive Modelle
    - Dummy-Modell (konstante Vorhersagen)
    - Multiple lineare Regression oder multiple logistische Regression
    - Regularisiertes lineares oder logistisches Modell (LASSO)
    - Entscheidungsbaum (CART mit Standardabbruchkriterien)
    - Random Forest (Standardeinstellungen, 500 Bäume, *mtry* = $\sqrt{p}$)
  - Resampling Strategie
    - Repeated Cross-Validation mit *K* = 10 Folds und *W* = 10 Wiederholungen
    - Verwendung von Stratifizierung im Klassifikationsfall
  - Performancemaße
    - Regression: *MSE*, R^2
    - Klassifikation: *MMCE*, *SENS*, *SPEC*

Exkurs: R Code Benchmark

(Interpretationen und Outputs sind prüfungsrelevant, nur der R Code nicht)

Beispiel 1: Benchmark (Regression: Boston Housing)


```r
library(mlr3verse)

BostonHousing_task <- TaskRegr$new(id = "boston", backend = BostonHousing, target = "Wohnungspreis")

set.seed(1)

### create benchmark experiment
BostonHousing_design <- benchmark_grid(
  tasks = BostonHousing_task,
  learners = list(lrn("regr.featureless"), lrn("regr.lm"),
    po("encode", method = "treatment") %>>% lrn("regr.cv_glmnet"),
    lrn("regr.rpart"), lrn("regr.ranger")),
  resamplings = rsmp("repeated_cv", folds = 10, repeats = 10)
)
BostonHousing_bm <- benchmark(BostonHousing_design)
```

Verwendeter Datensatz:
Boston Housing
Kriteriumsvariable:
Wohnungspreis

Modelle, die verglichen werden:
Dummy Modell, Multiple lineare Regression, LASSO, Entscheidungsbaum, Random Forest

Verwendete Resampling-Strategie:
Repeated Cross-Validation: 10 Folds, 10 Wiederholungen

Beispiel 1: Benchmark (Regression: Boston Housing)

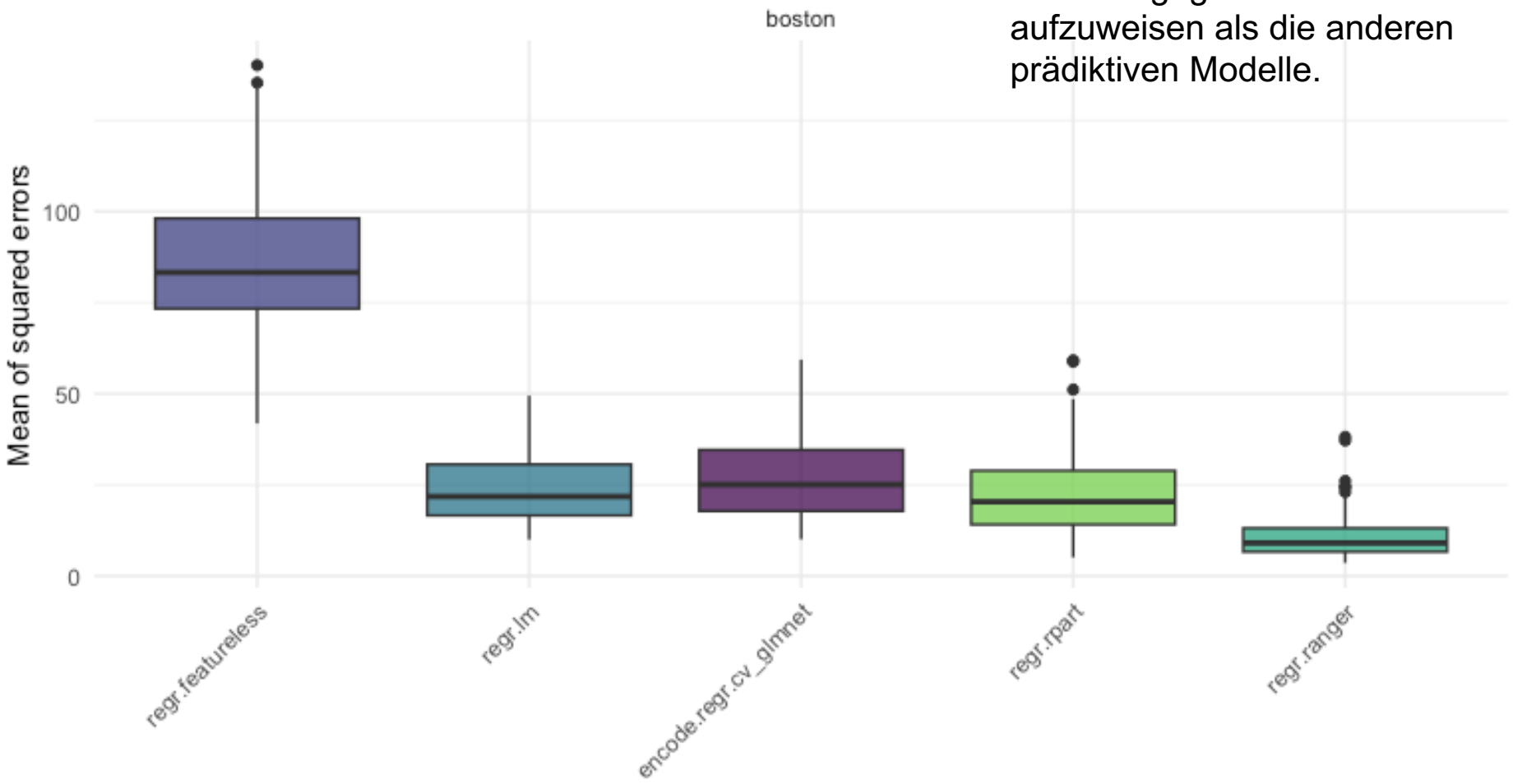

```
measures <- list(msr("regr.mse", id = "mse_test"),
                 msr("regr.rsq", id = "rsq_test"))
BostonHousing_bm$aggregate(measures)
```

Performance-Maße: MSE, R^2

learner_id	resampling_id	iters	mse_test	rsq_test
regr.featureless	repeated_cv	100	84.78651	-0.02731298
regr.lm	repeated_cv	100	23.83422	0.70970755
encode.regr.cv_glmnet	repeated_cv	100	26.81132	0.67705403
regr.rpart	repeated_cv	100	22.81299	0.71621797
regr.ranger	repeated_cv	100	11.07547	0.86686380

Beispiel 1: Benchmark (Regression: Boston Housing)


```
mlr3viz::autoplot(BostonHousing_bm)
```

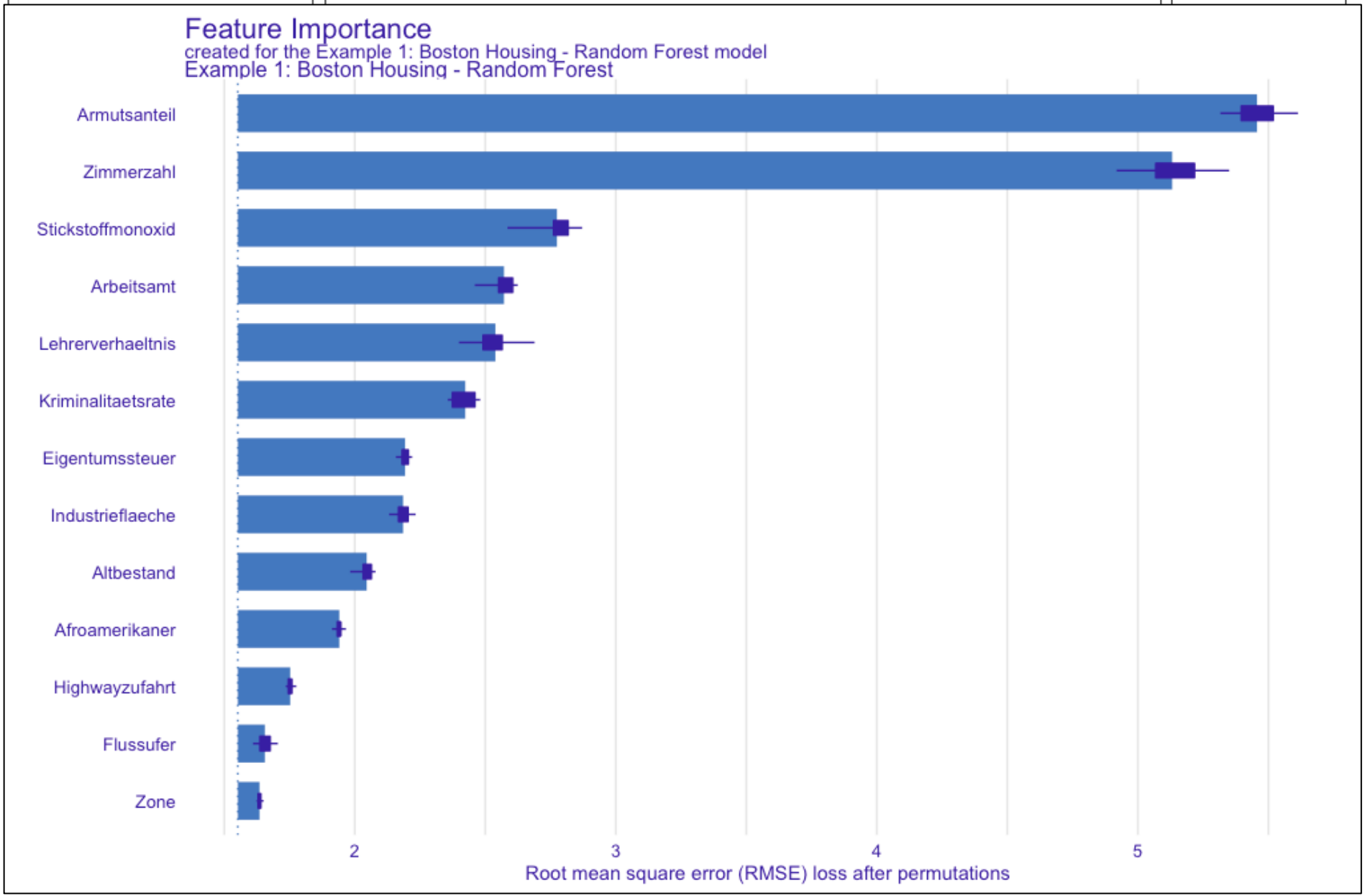


Der Random Forest scheint eine bessere Vorhersagegüte aufzuweisen als die anderen prädiktiven Modelle.

Lehrstuhl für Psychologische Methodenlehre und Diagnostik der Ludwig-Maximilians-Universität München

Beispiel 1: Boston Housing
Variable Importance Random Forest

Vorlesung Fortgeschrittene Statistische Methoden 1 WS 24/25



Beispiel 2: Benchmark (Klassifikation: Titanic)


```
library(mlr3verse)

Titanic_task <- TaskClassif$new(id = "titanic", backend = Titanic, target = "Ueberlebt", positive = "Leben")
prop.table(table(Titanic_task$truth()))

# Stratification
Titanic_task$col_roles$stratum <- Titanic_task$target_names

set.seed(1)

### create benchmark experiment
Titanic_design <- benchmark_grid(
  tasks = Titanic_task,
  learners = list(lrn("classif.featureless"), lrn("classif.log_reg"),
    po("encode", method = "treatment") %>>% lrn("classif.cv_glmnet"),
    lrn("classif.rpart"), lrn("classif.ranger")),
  resamplings = rsmp("repeated_cv", folds = 10, repeats = 10)
)
Titanic_bm <- benchmark(Titanic_design)
```

Verwendeter Datensatz: Titanic
Kriteriumsvariable: Überlebt

Stratifizierung, weil:

```
> prop.table(table(Titanic_task$truth()))

    Leben       Tod
0.4082218 0.5917782
```

Modelle, die verglichen werden:
Dummy Modell, Multiple logistische Regression, LASSO, Entscheidungsbaum, Random Forest

Verwendete Resampling-Strategie:
Repeated Cross-Validation: 10 Folds, 10 Wiederholungen (mit Stratifizierung → siehe oben)

Beispiel 2: Benchmark (Klassifikation: Titanic)

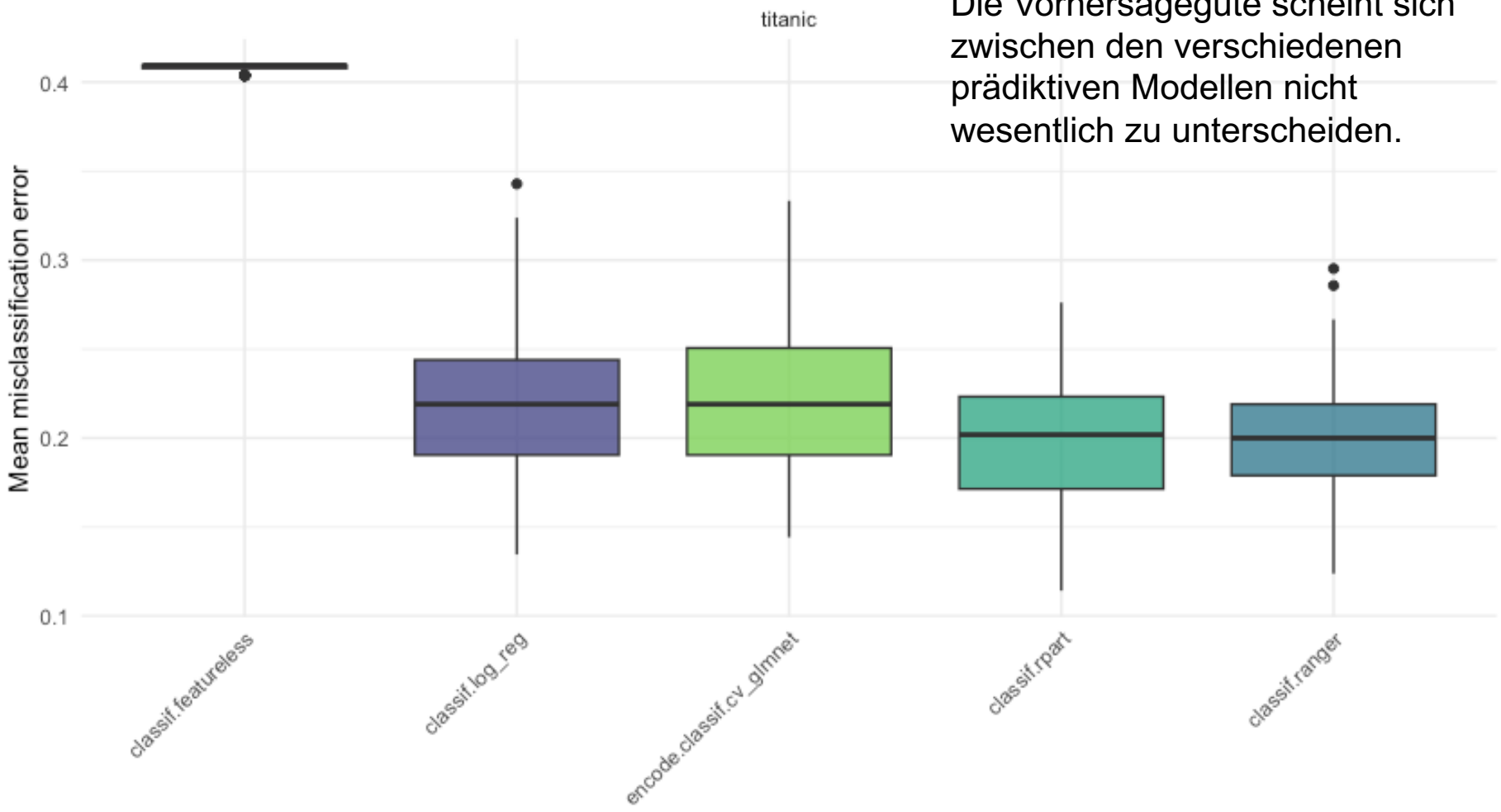

```
measures <- list(msr("classif.ce", id = "MMCE_test"),
                 msr("classif.tpr", id = "SENS_test"),
                 msr("classif.tnr", id = "SPEC_test"))
Titanic_bm$aggregate(measures)
```

Performance-Maße: MMCE, Sensitivität, Spezifität

learner_id	resampling_id	iters	MMCE_test	SENS_test	SPEC_test
classif.featureless	repeated_cv	100	0.4082126	0.0000000	1.0000000
classif.log_reg	repeated_cv	100	0.2183161	0.7092359	0.8316631
encode.classif.cv_glmnet	repeated_cv	100	0.2210141	0.6813234	0.8463591
classif.rpart	repeated_cv	100	0.1985680	0.6413289	0.9119381
classif.ranger	repeated_cv	100	0.1998875	0.6483278	0.9048493

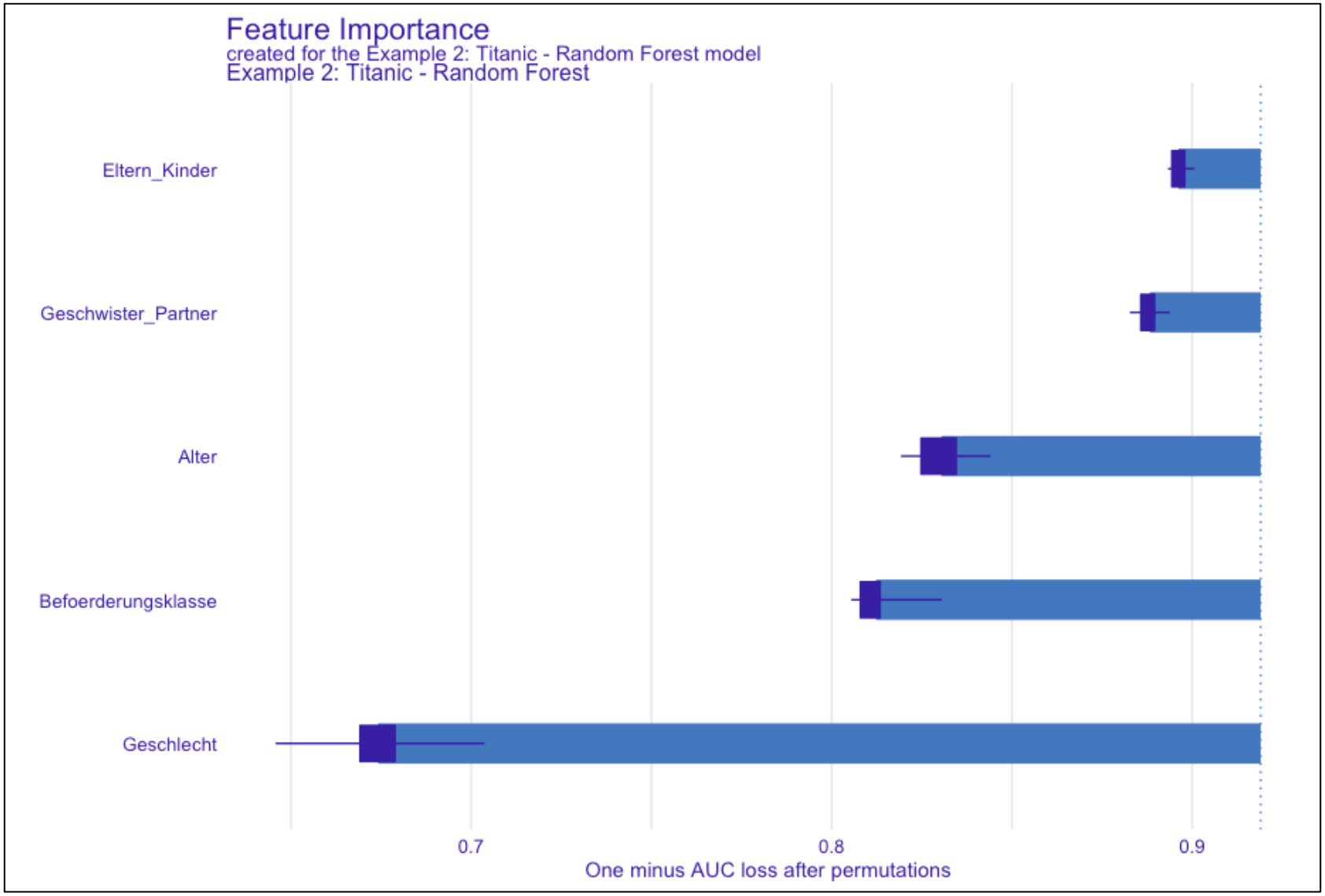
Beispiel 2: Benchmark (Klassifikation: Titanic)


```
mlr3viz::autoplot(Titanic_bm)
```



Die Vorhersagegüte scheint sich zwischen den verschiedenen prädiktiven Modellen nicht wesentlich zu unterscheiden.

Beispiel 2: Titanic
Variable Importance Random Forest



Beispiel 2: Titanic
Schätzwerte Logistische Regression


```
Titanic_logreg = glm(Ueberlebt ~ ., data = Titanic, family = "binomial")
summary(Titanic_logreg)
```

```
> summary(Titanic_logreg)

Call:
glm(formula = Ueberlebt ~ ., family = "binomial", data = Titanic)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7018  -0.6615  -0.4199   0.6660   2.5197

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)             3.906791   0.361918  10.795  < 2e-16 ***
Befoerderungsklasse2nd -1.366757   0.230041  -5.941 2.83e-09 ***
Befoerderungsklasse3rd -2.352022   0.228803 -10.280  < 2e-16 ***
Geschlechtmaennlich    -2.556860   0.173270 -14.756  < 2e-16 ***
Alter                  -0.039489   0.006634  -5.952 2.64e-09 ***
Geschwister_Partner    -0.352915   0.105349  -3.350 0.000808 ***
Eltern_Kinder           0.074361   0.099907   0.744 0.456695
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1414.62  on 1045  degrees of freedom
Residual deviance:  970.12  on 1039  degrees of freedom
AIC: 984.12

Number of Fisher Scoring iterations: 4
```

Dummy Kodierung für Beförderungsklasse: Referenzkategorie 1. Klasse

Dummy Kodierung für Geschlecht: Referenzkategorie weiblich

Prädiktorvariablen wurden nicht standardisiert.
Damit kann die Höhe der Schätzwerte nicht direkt zwischen den Variablen verglichen werden

Beispiel 2: Titanic
Schätzwerte LASSO


```
> set.seed(1)
> lasso <- as_learner(po("encode", method = "treatment") %>>% lrn("classif.cv_glmnet"))
> lasso$train(Titanic_task)
> coef(lasso$model$classif.cv_glmnet$model)
7 x 1 sparse Matrix of class "dgCMatrix"
                                s1
(Intercept)              2.04553707
Alter                   -0.01304894
Eltern_Kinder            .
Geschwister_Partner     -0.05057149
Befoerderungsklasse.2nd -0.39573433
Befoerderungsklasse.3rd -1.29497168
Geschlecht.maennlich    -2.17232093
> lasso$model$classif.cv_glmnet$model$lambda.1se
[1] 0.0214487
```

Exkurs: Im Gegensatz zur glm Funktion in R, kann das glmnet Paket nicht automatisch aus Faktorvariablen Dummyvariablen erstellen. Mithilfe des Pakets mlr3pipelines ist dies jedoch relativ einfach möglich.

Optimaler Wert für λ, vor dem Training automatisch bestimmt mithilfe von 10-fold CV.

Wie zu erwarten sind die Schätzwerte im Betrag etwas kleiner als bei der logistischen Regression ohne Regularisierung.

Beispiel 3: Benchmark (Klassifikation: Depression)


```r
library(mlr3verse)

# create Tasks
## Scale Level
Depression_scalelevel_task <- TaskClassif$new(id = "Depression Skalen Level", backend = Depression_scalelevel, target = "Depression", positive = "Diagnose vorhanden")
prop.table(table(Depression_scalelevel_task$truth()))
Depression_scalelevel_task$col_roles$stratum <- Depression_scalelevel_task$target_names

## Item level
Depression_itemlevel_task <- TaskClassif$new(id = "Depression Item Level", backend = Depression_itemlevel, target = "Depression", positive = "Diagnose vorhanden")
prop.table(table(Depression_itemlevel_task$truth()))
Depression_itemlevel_task$col_roles$stratum <- Depression_itemlevel_task$target_names

# create Benchmark Experiment
set.seed(1)

Depression_design <- benchmark_grid(
  tasks = list(Depression_scalelevel_task, Depression_itemlevel_task),
  learners = list(lrn("classif.featureless"), lrn("classif.log_reg"),
    lrn("classif.cv_glmnet"), lrn("classif.rpart"), lrn("classif.ranger")),
  resamplings = rsmp("repeated_cv", folds = 10, repeats = 10)
)
Depression_bm <- benchmark(Depression_design)
```

1

2

Verwendeter Datensatz: Depression
Kriteriumsvariable: ICD10 Diagnose vorhanden

Modelle, die verglichen werden:
Dummy Modell, Multiple logistische Regression, LASSO, Entscheidungsbaum, Random Forest

Verwendete Resampling-Strategie:
Repeated Cross-Validation: 10 Folds, 10 Wiederholungen (mit Stratifizierung → siehe oben)

1. Prädiktoren auf Skalen Level:
Alter, Geschlecht, BDI Gesamtwert, FIE Gesamtwert

2. Prädiktoren auf Item Level:
Alter, Geschlecht, 21 BDI Items, 30 FIE Items

Beispiel 3: Benchmark (Klassifikation: Depression)

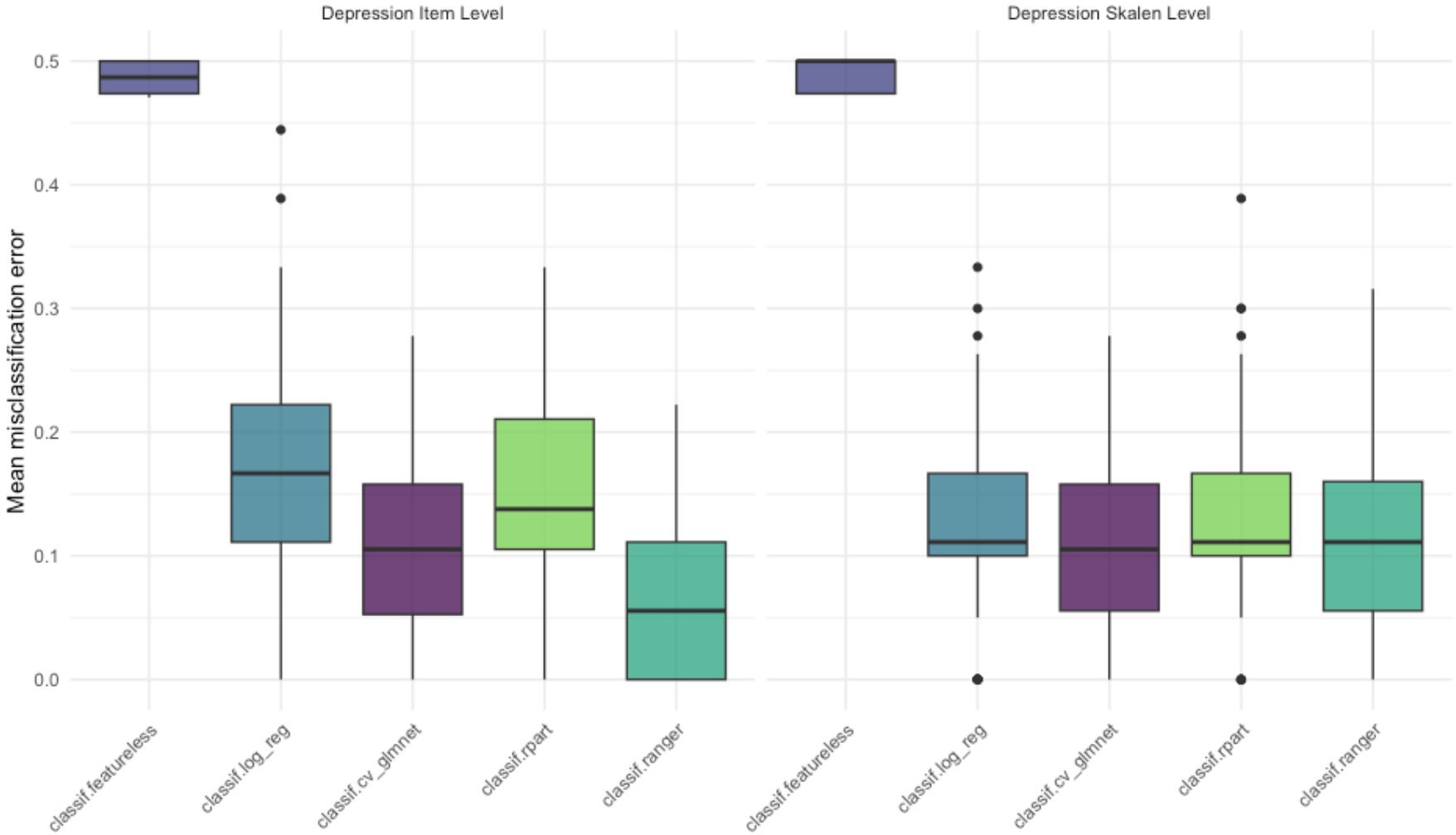

```
measures <- list(msr("classif.ce", id = "MMCE_test"),
                 msr("classif.tpr", id = "SENS_test"),
                 msr("classif.tnr", id = "SPEC_test"))
Depression_bm$aggregate(measures)
```

Performance-Maße:
MMCE, Sensitivität, Spezifität

task_id	learner_id	resampling_id	iters	MMCE_test	SENS_test	SPEC_test
Depression Skalen Level	classif.featureless	repeated_cv	100	0.48947368	0.0000000	1.0000000
Depression Skalen Level	classif.log_reg	repeated_cv	100	0.13096491	0.8492222	0.8881111
Depression Skalen Level	classif.cv_glmnet	repeated_cv	100	0.11539474	0.8330000	0.9343333
Depression Skalen Level	classif.rpart	repeated_cv	100	0.13056140	0.8473333	0.8907778
Depression Skalen Level	classif.ranger	repeated_cv	100	0.12729825	0.8407778	0.9030000
Depression Item Level	classif.featureless	repeated_cv	100	0.48653251	0.0000000	1.0000000
Depression Item Level	classif.log_reg	repeated_cv	100	0.17276574	0.8106944	0.8432222
Depression Item Level	classif.cv_glmnet	repeated_cv	100	0.09508256	0.8626389	0.9454444
Depression Item Level	classif.rpart	repeated_cv	100	0.14706398	0.8575000	0.8484444
Depression Item Level	classif.ranger	repeated_cv	100	0.07113863	0.9037500	0.9526667

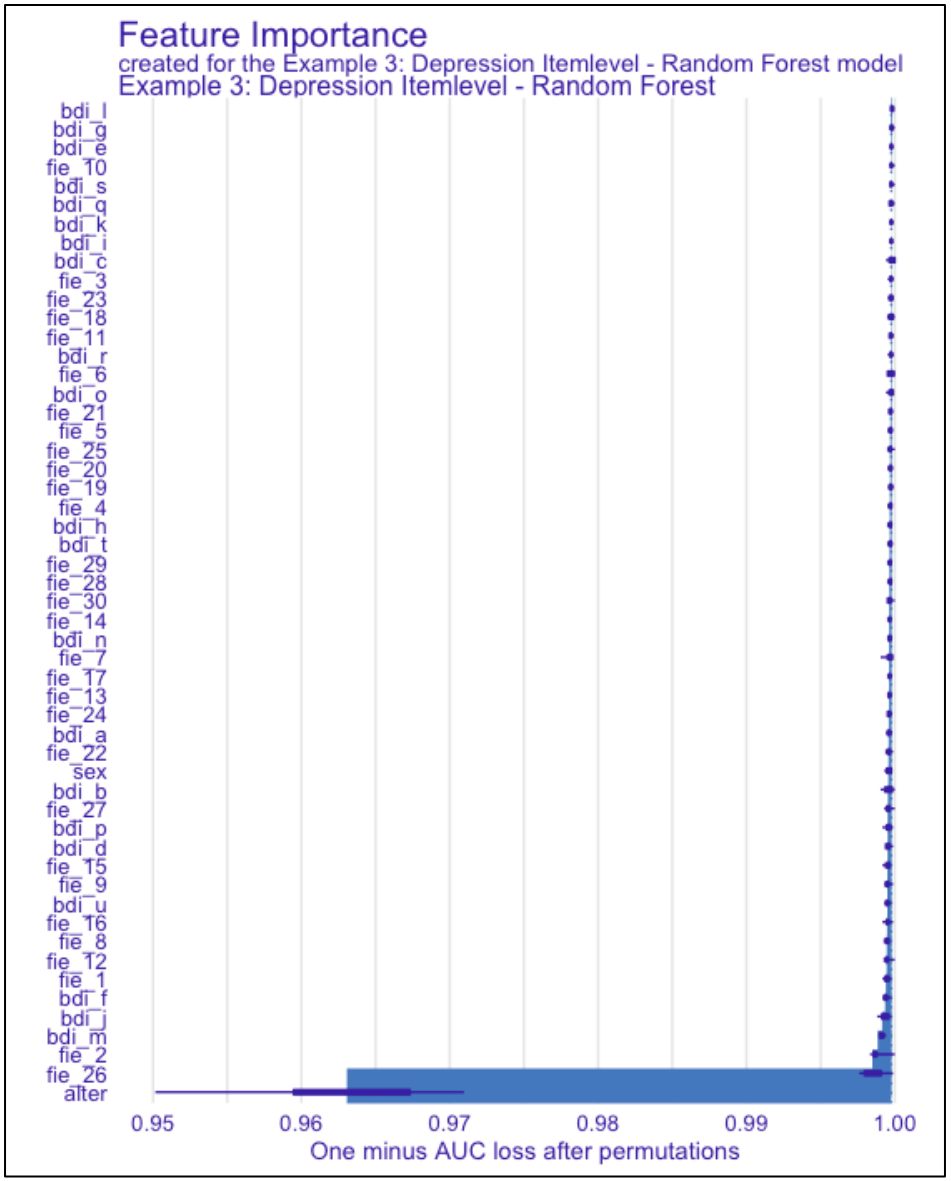
Beispiel 3: Benchmark (Klassifikation: Depression)


```
mlr3viz::autoplot(Depression_bm)
```

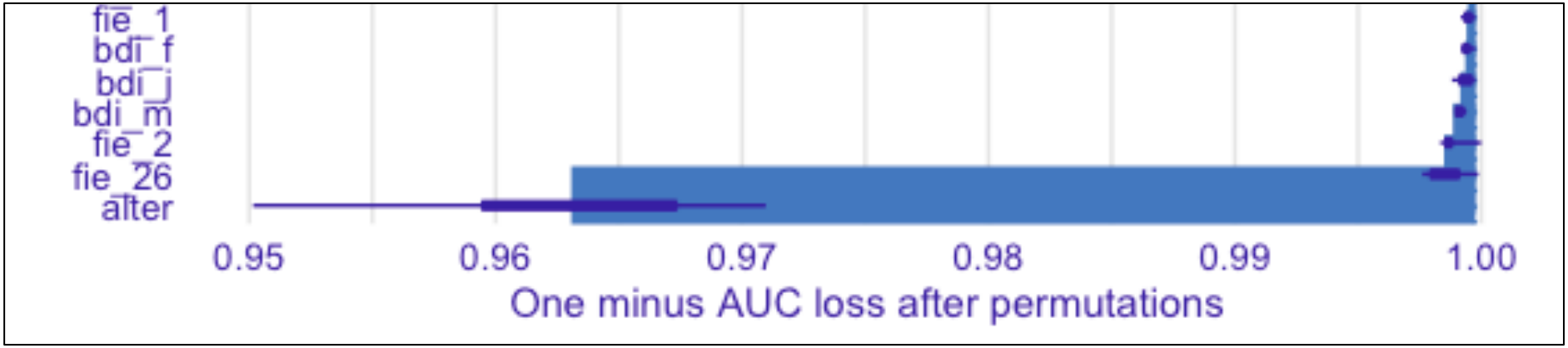


Die Vorhersagegüte scheint am besten mit dem Random Forest, bei Verwendung der einzelnen Items als Prädiktoren.

Beispiel 3: Depression
Variable Importance Random Forest



Beispiel 3: Depression
Variable Importance Random Forest



Beispiel 4: Benchmark (Regression: PhoneStudy)

- Regression: **PhoneStudy**
  - Kommunikationsverhalten von 609 Probanden, aggregiert über 30 Tage mithilfe einer App auf den Smartphones der Studienteilnehmer*innen des „PhoneStudy“ Projekts (Lehrstuhl für Psychologische Methodenlehre und Diagnostik der LMU)
  - y_i: Summenwert in der Extraversionsfacette Geselligkeit des Big-Five Struktur Inventars (BFSI)
  - 95 Prädiktoren (ein kleiner Ausschnitt von über 10.000 Sensorvariablen):
    Mittlere Anzahl ausgehender Anrufe pro Tag, mittlere Länge ausgehender SMS pro Tag, Antwortrate von Anrufen, Entropiemaß bzgl. der Kontakte eingehender Anrufe, mittlere Nutzungsdauer von Whatsapp pro Tag, mittlere Nutzungsdauer von Facebook pro Tag, ...

Beispiel 4: Benchmark (Regression: PhoneStudy)


```
PhoneStudy_task <- TaskRegr$new(id = "phonestudy", backend = PhoneStudy, target = "Geselligkeit")

set.seed(1)

### create benchmark experiment
PhoneStudy_design <- benchmark_grid(
  tasks = PhoneStudy_task,
  learners = list(lrn("regr.featureless"), lrn("regr.lm"), lrn("regr.cv_glmnet"),
    lrn("regr.rpart"), lrn("regr.ranger")),
  resamplings = rsmp("repeated_cv", folds = 10, repeats = 10)
)
PhoneStudy_bm <- benchmark(PhoneStudy_design)
```

Verwendeter Datensatz: PhoneStudy
Kriteriumsvariable: Geselligkeit

Modelle, die verglichen werden:
Dummy Modell, Multiple lineare Regression, LASSO, Entscheidungsbaum, Random Forest

Verwendete Resampling-Strategie:
Repeated Cross-Validation: 10 Folds, 10 Wiederholungen

Beispiel 4: Benchmark (Regression: PhoneStudy)

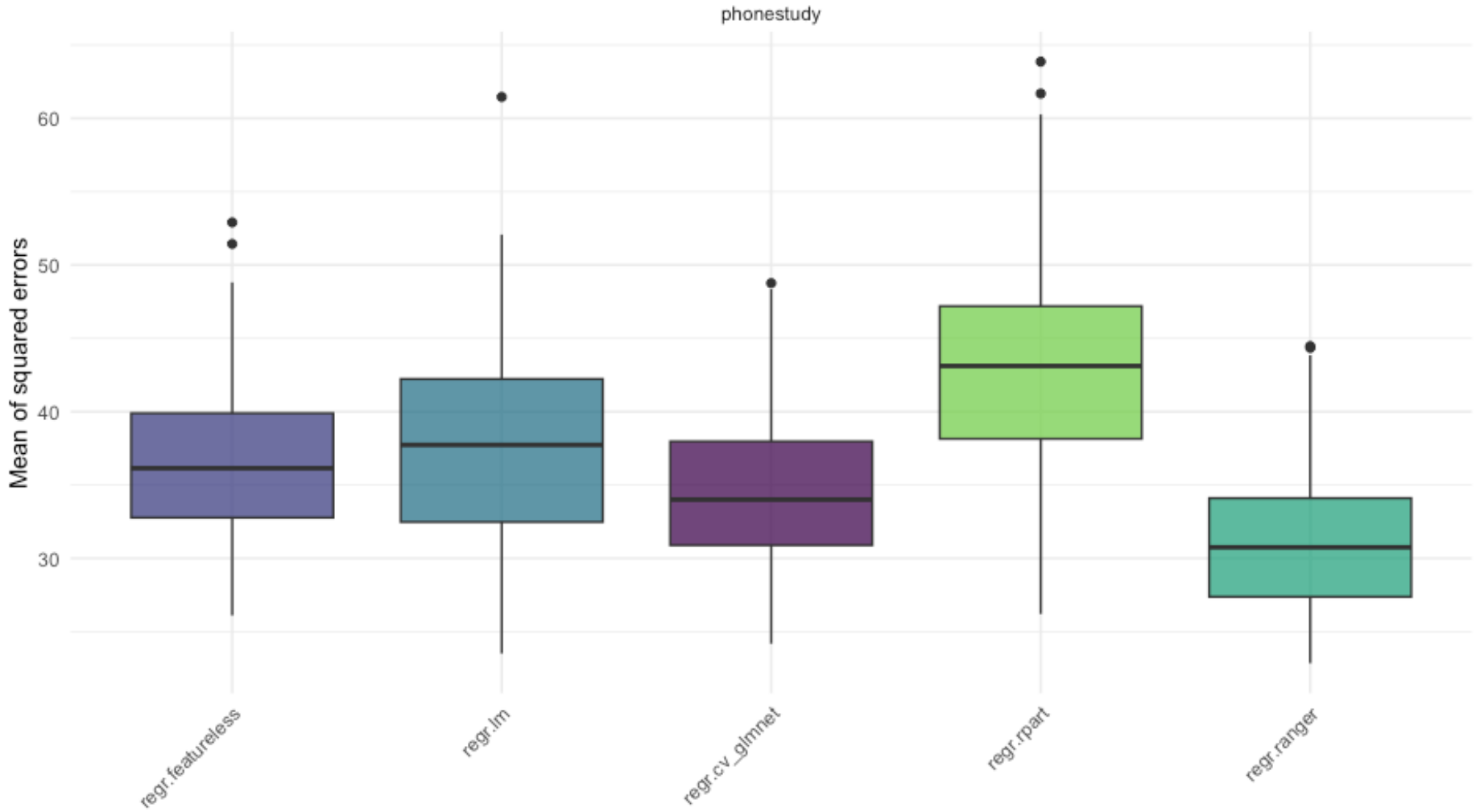

```
measures <- list(msr("regr.mse", id = "mse_test"),
                 msr("regr.rsq", id = "rsq_test")
)
PhoneStudy_bm$aggregate(measures)
```

→ Performance-Maße: MSE, R^2

learner_id	resampling_id	iters	mse_test	rsq_test
regr.featureless	repeated_cv	100	36.47667	-0.01928183
regr.lm	repeated_cv	100	38.03521	-0.06792817
regr.cv_glmnet	repeated_cv	100	34.47424	0.03656905
regr.rpart	repeated_cv	100	43.06719	-0.21397876
regr.ranger	repeated_cv	100	31.16785	0.12696329

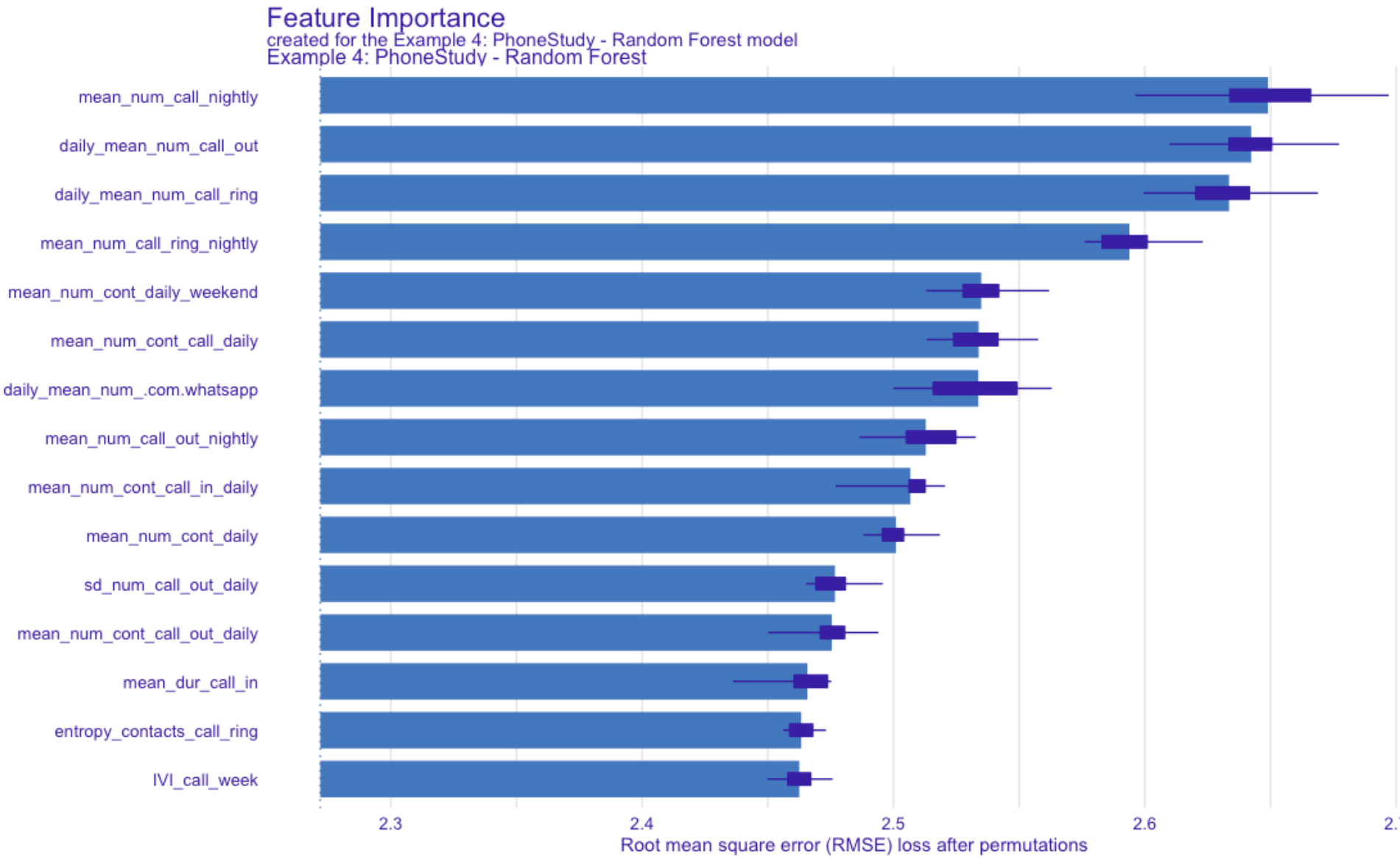
Beispiel 4: Benchmark (Regression: PhoneStudy)


```
mlr3viz::autoplot(PhoneStudy_bm)
```



Der Random Forest scheint als einzige Modellklasse ein wenig besser zu funktionieren als das Dummy-Modell

Beispiel 4: PhoneStudy
Variable Importance Random Forest



Anwendungsfazit

- Für alle Datensätze gelingt eine bessere Vorhersage als mit dem Dummy Modell
- **Titanic**: Vorhersagegüte verschiedener Modele ist bezüglich des MMCE sehr ähnlich, dabei unterscheiden sich die Modelle jedoch leicht bezüglich SENS und SPEC
  → Welches Modell am besten ist, hängt von der praktischen Anwendung ab
- **Boston Housing, Depression, PhoneStudy**: Random Forest scheint bessere Vorhersagen zu liefern als lineare Modelle
  → Vermutlich nicht lineare Zusammenhänge und/oder Interaktionen
- **Depression**: Beste Vorhersagegüte wenn die BDI und FIE-Items nicht zu Summenwerten zusammengefasst werden, sondern als einzelne Prädiktor-variablen in den Random Forest eingehen
  → Vermutlich Informationsverlust bei Verwendung der Summenwerte
- **Depression**: Die Prädiktorvariable Alter scheint für die Vorhersagen des Random Forests besonders wichtig zu sein
  → Bias in Datensatz: Probanden ohne depressive Diagnose waren alles Studierende

Leakage

- Ziel der prädiktiven Modellierung ist es, ein Modell zu entwickeln, das genaue Vorhersagen für neue, *beim Training nicht gesehene Daten* trifft.
- Manchmal „sickert“ (leakage) Information vom Trainingsset in das Testset, ohne dass wir uns dessen bewusst sind.
- Dann sind die Daten im Testset nicht komplett neu, sondern enthalten Informationen aus dem Trainingsset und haben daher das Problem des Overfittings.
- Tatsächlich neue Daten – die nicht den Vorteil haben, dass schon (Teil)Informationen im Trainingsset gelernt wurde – werden schlechter vorhergesagt. Die tatsächliche Performance ist also geringer (und wird durch die Leakage im Benchmarking-Experiment überschätzt).

„Leakage and the reproducibility crisis in machine-learning-based science“

Field	Paper	Number of papers reviewed	Number of papers with pitfalls	[L1.1] No test set	[L1.2] Pre-proc. on train-test	[L1.3] Feature sel. on train-test	[L1.4] Duplicates	[L2] Illegitimate features	[L3.1] Temporal leakage	[L3.2] Non-ind. b/w train-test	[L3.3] Sampling bias	Comput. reproducibility issues	Data quality issues	Metric choice issues	Standard dataset used?
Medicine	Bouwmeester et al. (2012)	71	27	○								○			
Neuroimaging	Whelan & Garavan (2014)	–	14	○		○									
Bioinformatics	Blagus & Lusa (2015)	–	6		○										
Autism Diagnostics	Bone et al. (2015)	–	3				○				○		○	○	○
Nutrition Research	Ivanescu et al. (2016)	–	4	○								○	○		
Software Eng.	Tu et al. (2018)	58	11					○			○	○		○	
Toxicology	Alves et al. (2019)	–	1				○				○	○			
Clinical Epidem.	Christodoulou et al. (2019)	71	48			○						○			
Satellite Imaging	Nalepa et al. (2019)	17	17						○			○		○	
Tractography	Poulin et al. (2019)	4	2	○							○	○	○	○	
Brain-computer Int.	Nakanishi et al. (2020)	–	1	○										○	
Histopathology	Oner et al. (2020)	–	1						○						
Neuropsychiatry	Poldrack et al. (2020)	100	53	○	○							○	○		
Neuroimaging	Ahmed et al. (2021)	–	1						○						
Neuroimaging	Li et al. (2021)	122	18						○						
IT Operations	Lyu et al. (2021)	9	3					○						○	
Medicine	Filho et al. (2021)	–	1				○								
Radiology	Roberts et al. (2021)	62	16	○			○				○	○			○
Neuropsychiatry	Shim et al. (2021)	–	1			○						○			
Medicine	Vandewiele et al. (2021)	24	21			○			○	○	○	○			○
Computer Security	Arp et al. (2022)	30	22	○	○	○		○	○	○	○		○	○	
Genomics	Barnett et al. (2022)	41	23			○							○		

Paper under CC-BY 4.0

Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. *Patterns, 4(9)*, 100804. https://doi.org/10.1016/j.patter.2023.100804

Leakage: Gefahren bei abhängigen Beobachtungen

- Prädiktive Modelle können mit abhängigen Beobachtungen verwendet werden.
  - Beispiele: Schüler*innen geschachtelt in Schulen, Längsschnittdaten, …
- Die meisten Modellklassen (z.B. der normale Random Forest) können bekannte Abhängigkeiten nicht explizit berücksichtigen, aber diese eventuell selbst lernen (z.B. Schule als Prädiktor) und somit trotzdem für bessere Vorhersagen nutzen.
- Auch bei der Verwendung von Modellklassen, die extra für abhängige Beobachtungen konstruiert wurden (z.B. Gemischte Lineare Modelle), kann es sinnvoll sein, die erwartete Vorhersageleistung bei neuen Daten zu schätzen.
- Möchte man durch Resampling die Vorhersagegüte eines prädiktiven Modells evaluieren das mit abhängigen Daten trainiert wurde, muss man aber **aufpassen**! (siehe nächste Folie)

Leakage: Gefahren bei abhängigen Beobachtungen

- Bei abhängigen Beobachtungen gibt es typischerweise verschiedene Vorhersageszenarien, die unterschiedlich evaluiert werden müssen, z.B.:
  1. Vorhersage für neue Schüler*in aus einer **bekannten** Schule
  2. Vorhersage für neue Schüler*in aus einer **neuen** Schule
- Normale *k*-fold CV ist nur angemessen für Szenario 1.
  - Bei normaler *k*-fold CV wird die Gruppe zu der eine Beobachtung gehört nicht berücksichtigt: Jede Schüler*in kann im Trainingsset oder im Testset landen.
- Für Szenario 2 ist normale *k*-fold CV **nicht** angemessen:
  - Das Trainingsset enthält Beobachtungen von Schüler*innen die zu der gleichen Schule gehören wie Schüler*innen im Testset.
  - Dies entspricht nicht der Anwendung des finalen Modells in der Realität: Hier würde eine *neue* Schule niemals im Trainingsset vorkommen.

  → **Unterschätzung des erwarteten Vorhersagefehlers!**

Leakage: Gefahren bei abhängigen Beobachtungen

- Lösung: *k*-fold CV mit **Blocking**
  - Bei der zufälligen Aufteilung der Beobachtungen in die Trainings- und Testsets wird sichergestellt, dass alle Beobachtungen die zu einer Gruppe gehören alle entweder im Trainingsset oder im Testset landen.
  - Damit ist sichergestellt, dass für jede Kombination aus Trainings- und Testset das Modell niemals auf Daten aus den gleichen Gruppen trainiert wurde, für die im Testset Vorhersagen berechnet werden.
- Spezialfall: **Leave-one-Group-out CV** (hier: Leave-one-School-Out CV)
  - Genau so viele Testsets wie Gruppen im Datensatz
  - Jedes Testset besteht aus allen Beobachtungen nur genau einer Gruppe

Beispiel 5: Benchmark (Regression: Matheleistung)


```r
library(mlr3verse)

HSB_task <- TaskRegr$new(id = "HSB", backend = HSB, target = "MathAch")
HSB_task$set_col_roles("School", add_to = "group")

set.seed(1)

### create benchmark experiment
HSB_design <- benchmark_grid(
  tasks = HSB_task,
  learners = list(lrn("regr.featureless"),
    lrn("regr.lmer", allow.new.levels = TRUE,
      formula = MathAch ~ 1 + SES + Sex + Minority + MeanSES + Size + Sector + (1 + SES + Sex| School)),
    lrn("regr.ranger")),
  resamplings = rsmp("repeated_cv", folds = 10, repeats = 10)
)
HSB_bm <- benchmark(HSB_design)
```

Verwendeter Datensatz: HSB
Kriteriumsvariable: Matheleistung

Blocking:
Jede Schule ist immer entweder im Trainingsset oder im Testset

Modelle, die verglichen werden:
Dummy Modell, Gemischtes Lineares Modell (theoriebasiert!), Random Forest

Verwendete Resampling-Strategie:
Repeated Cross-Validation: 10 Folds, 10 Wiederholungen (mit Blocking → siehe oben)

Beispiel 5: Benchmark (Regression: Matheleistung)


```
measures <- list(msr("regr.mse", id = "mse_test"),
                 msr("regr.rsq", id = "rsq_test")
)
HSB_bm$aggregate(measures)
```

→ Performance-Maße: MSE, R^2

learner_id	resampling_id	iters	mse_test	rsq_test
regr.featureless	repeated_cv	100	47.45274	-0.01446799
regr.lmer	repeated_cv	100	37.64164	0.19361781
regr.ranger	repeated_cv	100	38.14136	0.18311655

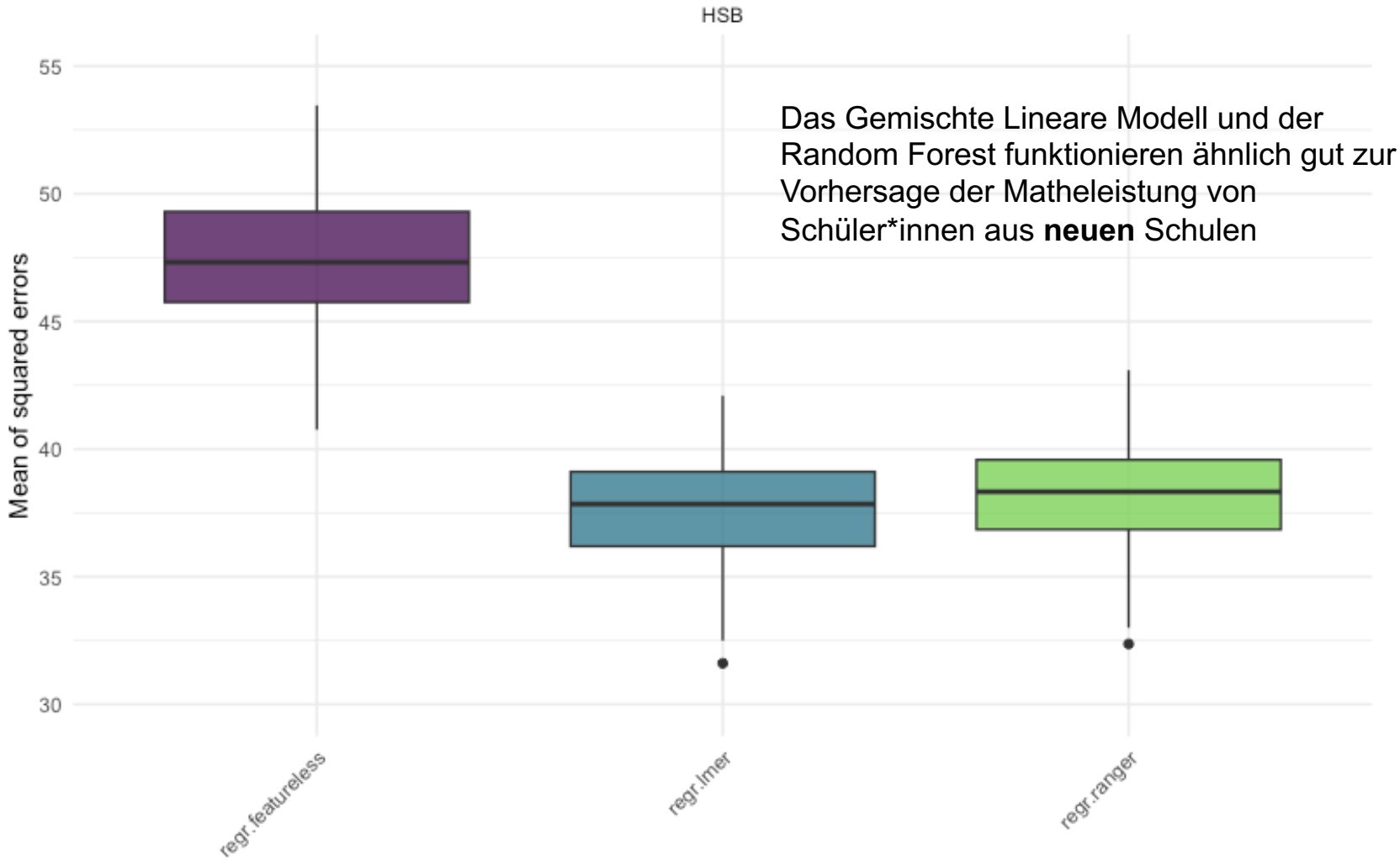
Vergleich mit dem geschätzten Vorhersagefehler ohne Resampling ($d.h., die\ in\ sample\ performance$):

Modell	MSE_{train}	R^2_{train}
LMM	34.88	0.26
Random Forest	24.98	0.47

→ Dieses R^2 (als überschätzte in-sample-performance) haben wir im ersten Teil der VL berechnet!

Beispiel 5: Benchmark (Regression: Matheleistung)

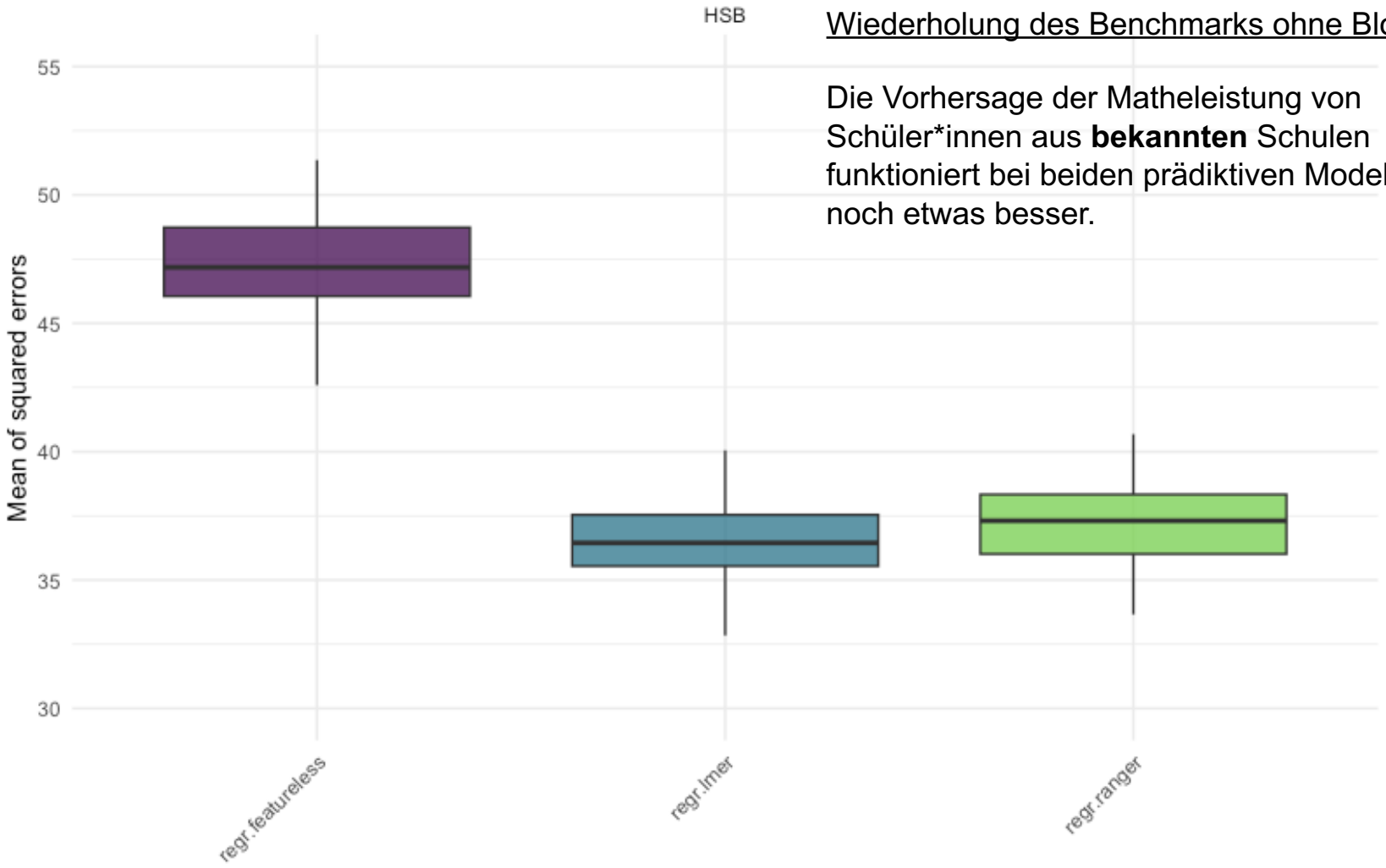

```
mlr3viz::autoplot(HSB_bm)
```



Das Gemischte Lineare Modell und der Random Forest funktionieren ähnlich gut zur Vorhersage der Matheleistung von Schüler*innen aus **neuen** Schulen

Beispiel 5: Benchmark (Regression: Matheleistung)


```
mlr3viz::autoplot(HSB_bm_no_block)
```



<u>Wiederholung des Benchmarks ohne Blocking:</u>

Die Vorhersage der Matheleistung von Schüler*innen aus **bekannten** Schulen funktioniert bei beiden prädiktiven Modellen noch etwas besser.

Leakage: Gefahren bei Variablenselektion

- In der Praxis möchte man vor der prädiktiven Modellierung manchmal eine Vorauswahl an Prädiktorvariablen treffen, die auf dem Zusammenhang mit der Kriteriumsvariable basiert („Nur die besten Kandidaten verwenden“)
  - Beispiel: Verwende die *M* Variablen als Prädiktorvariablen, die im Gesamtdatensatz am höchsten mit der Kriteriumsvariable korrelieren
- Möchte man durch Resampling die Vorhersagegüte eines prädiktiven Modells evaluieren, das basierend auf den vorausgewählten Prädiktoren trainiert wurde, muss man **aufpassen**!
- *Falsches* Vorgehen: Verwende innerhalb jedes Folds genau die Prädiktor-variablen, die zuvor anhand des gesamten Datensatzes (also inklusive des Testsets) ausgewählt wurden
  → **Unterschätzung des erwarteten Vorhersagefehlers!**
- *Richtiges* Vorgehen: Wiederhole die Variablenselektion auf dem *Trainingsdatensatz* von jedem Fold (z.B. wähle immer im Trainingsdatensatz jedes Folds die *M* Prädiktorvariablen aus, die am höchsten mit der Kriteriumsvariable korrelieren).

Leakage: Gefahren bei Hyperparameter-Tuning

- Um die Vorhersagegüte eines prädiktiven Modells zu erhöhen, möchte man manche Hyperparameter („Zusatzeinstellungen“ der entsprechenden Modellklasse) für das vorliegende Problem optimal wählen.
  - Beispiel: Finde beim Random Forest den optimalen Wert für *mtry* durch automatisiertes „Hyperparameter-Tuning“ (bei Interesse, siehe mlr3 Manual)
- Möchte man durch Resampling die Vorhersagegüte des prädiktiven Modells mit den optimalen Hyperparameter-Einstellungen evaluieren, muss man **aufpassen**!
- Falsches Vorgehen: Trainiere das Modell in jedem Fold mit den Hyperparameter Einstellungen, die zuvor anhand des *gesamten* Datensatzes (also inklusive des Testsets) ausgewählt wurden
  → **Unterschätzung des erwarteten Vorhersagefehlers!**
- Richtiges Vorgehen: Wiederhole das Hyperparameter-Tuning im *Trainingsdatensatz* von jedem Fold.

Leakage verhindern: Generelle Strategie beim Resampling

Um eine Unterschätzung des erwarteten Vorhersagefehlers bei der Modellevaluation mit Resampling zu verhindern, gibt es folgende einfache Lösung:

„Simuliere den kompletten Modellierungsprozess innerhalb des Resamplings“

- Alle Schritte die im gesamten Datensatz notwendig sind um mit dem prädiktiven Modell für neue Beobachtungen Vorhersagen treffen zu können, müssen bei der Evaluation des Modells in jedem Cross-Validation Fold durchgeführt werden.
  - Variablenselektion (Vorauswahl der besten Prädiktorvariablen)
  - Hyperparameter Tuning (Finden der besten Hyperparameter-Einstellungen)
  - Andere Vorverarbeitungsschritte (z.B. Imputation fehlender Werte)
- Generell ist es also notwendig, möglichst alle Verarbeitungsschritte vor dem Training des prädiktiven Modells zu automatisieren, da man diese sonst händisch in jedem Fold durchführen müsste ...
- Zusammengefasst: Der Testdatensatz darf *nie* (weder für sich genommen, noch als Teil des Gesamtdatensatzes) von irgendeinem Schritt des Modelltrainings gesehen werden.

Für Interessierte: Erweitertes Tutorial am Beispiel des vollständigen Phonestudy Datensatzes in Pargent, Schödel & Stachl (2022) https://doi.org/10.31234/osf.io/89snd