Lösungen zu den Aufgaben

1. **Aufgabe**

   Melden Sie sich an für die Kaggle Competition TMDB Box Office Prediction - Can you predict a movie's worldwide box office revenue?.

   Sie benötigen dazu ein Konto; es ist auch möglich, sich mit seinem Google-Konto anzumelden.

   Bei diesem Prognosewettbewerb geht es darum, vorherzusagen, wieviel Umsatz wohl einige Filme machen werden. Als Prädiktoren stehen einige Infos wie Budget, Genre, Titel etc. zur Verfügung. Eine klassische "predictive Competition" also :-) Allerdings können immer ein paar Schwierigkeiten auftreten ;-)

   Aufgabe

   Erstellen Sie ein Random-Forest-Modell mit Tidymodels! Reichen Sie es bei Kaggle ein un berichten Sie den Score!

   Hinweise

   - Verzichten Sie auf Vorverarbeitung.
   - Tunen Sie die typischen Parameter.
   - Begrenzen Sie sich auf folgende Prädiktoren.

```
preds_chosen <-
  c("id", "budget", "popularity", "runtime")
```

   Lösung

Pakete starten

```
library(tidyverse)
library(tidymodels)
library(tictoc)
```

Daten importieren

```
d_train_path <- "https://raw.githubusercontent.com/sebastiansauer/Lehre/main/data/tmdb-box-office-prediction/train.csv"
d_test_path <- "https://raw.githubusercontent.com/sebastiansauer/Lehre/main/data/tmdb-box-office-prediction/test.csv"

d_train <- read_csv(d_train_path)
d_test <- read_csv(d_test_path)
```

Werfen wir einen Blick in die Daten:

```
glimpse(d_train)
```

```
## Rows: 3,000
## Columns: 23
## $ id                    <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12…
## $ belongs_to_collection <chr> "[{'id': 313576, 'name': 'Hot Tub Tim…
## $ budget                <dbl> 1.40e+07, 4.00e+07, 3.30e+06, 1.20e+0…
## $ genres                <chr> "[{'id': 35, 'name': 'Comedy'}]", "[{…
## $ homepage              <chr> NA, NA, "http://sonyclassics.com/whip…
## $ imdb_id               <chr> "tt2637294", "tt0368933", "tt2582802"…
## $ original_language     <chr> "en", "en", "en", "hi", "ko", "en", "…
## $ original_title        <chr> "Hot Tub Time Machine 2", "The Prince…
## $ overview              <chr> "When Lou, who has become the \"fathe…
## $ popularity            <dbl> 6.575393, 8.248895, 64.299990, 3.1749…
## $ poster_path           <chr> "/tQtWuwvMf0hCc2QR2tkolwl7c3c.jpg", "…
## $ production_companies  <chr> "[{'name': 'Paramount Pictures', 'id'…
## $ production_countries  <chr> "[{'iso_3166_1': 'US', 'name': 'Unite…
## $ release_date          <chr> "2/20/15", "8/6/04", "10/10/14", "3/9…
## $ runtime               <dbl> 93, 113, 105, 122, 118, 83, 92, 84, 1…
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'Englis…
## $ status                <chr> "Released", "Released", "Released", "…
## $ tagline               <chr> "The Laws of Space and Time are About…
## $ title                 <chr> "Hot Tub Time Machine 2", "The Prince…
## $ Keywords              <chr> "[{'id': 4379, 'name': 'time travel'}…
## $ cast                  <chr> "[{'cast_id': 4, 'character': 'Lou', …
## $ crew                  <chr> "[{'credit_id': '59ac067c92514107af02…
## $ revenue               <dbl> 12314651, 95149435, 13092000, 1600000…
```

```
glimpse(d_test)
```

```
## Rows: 4,398
## Columns: 22
## $ id                    <dbl> 3001, 3002, 3003, 3004, 3005, 3006, 3…
## $ belongs_to_collection <chr> "[{'id': 34055, 'name': 'Pokémon Coll…
## $ budget                <dbl> 0.00e+00, 8.80e+04, 0.00e+00, 6.80e+0…
## $ genres                <chr> "[{'id': 12, 'name': 'Adventure'}, {'…
## $ homepage              <chr> "http://www.pokemon.com/us/movies/mov…
## $ imdb_id               <chr> "tt1226251", "tt0051380", "tt0118556"…
## $ original_language     <chr> "ja", "en", "en", "fr", "en", "en", "…
## $ original_title        <chr> "ディアルガVSパルキアVSダークライ", "…
## $ overview              <chr> "Ash and friends (this time accompani…
## $ popularity            <dbl> 3.851534, 3.559789, 8.085194, 8.59601…
```

```
## $ poster_path           <chr> "/tnftmLMemPLduW6MRyZE0ZUD19z.jpg", "…
## $ production_companies  <chr> NA, "[{'name': 'Woolner Brothers Pict…
## $ production_countries  <chr> "[{'iso_3166_1': 'JP', 'name': 'Japan…
## $ release_date          <chr> "7/14/07", "5/19/58", "5/23/97", "9/4…
## $ runtime               <dbl> 90, 65, 100, 130, 92, 121, 119, 77, 1…
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'Englis…
## $ status                <chr> "Released", "Released", "Released", "…
## $ tagline               <chr> "Somewhere Between Time & Space... A …
## $ title                 <chr> "Pokémon: The Rise of Darkrai", "Atta…
## $ Keywords              <chr> "[{'id': 11451, 'name': 'pok√©mon'}, …
## $ cast                  <chr> "[{'cast_id': 3, 'character': 'Tonio'…
## $ crew                  <chr> "[{'credit_id': '52fe44e7c3a368484e03…
```

`preds_chosen` sind alle Prädiktoren im Datensatz, oder nicht? Das prüfen wir mal kurz:

```
preds_chosen %in% names(d_train) %>%
  all()
```

```
## [1] TRUE
```

Ja, alle Elemente von `preds_chosen` sind Prädiktoren im (Train-)Datensatz.

CV

```
cv_scheme <- vfold_cv(d_train)
```

Rezept 1

```
rec1 <-
  recipe(revenue ~ budget + popularity + runtime, data = d_train) %>%
  step_impute_bag(all_predictors()) %>%
  step_naomit(all_predictors())
rec1
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          3
##
## Operations:
##
## Bagged tree imputation for all_predictors()
## Removing rows with NA values in all_predictors()
```

Man beachte, dass noch 21 Prädiktoren angezeigt werden, da das Rezept noch nicht auf den Datensatz angewandt (“gebacken”) wurde.

```
tidy(rec1)
```

```
## # A tibble: 2 × 6
##   number operation type       trained skip  id
##    <int> <chr>     <chr>      <lgl>   <lgl> <chr>
## 1      1 step      impute_bag FALSE   FALSE impute_bag_YCfi7
## 2      2 step      naomit     FALSE   FALSE naomit_XSBmv
```

Rezept checken:

```
prep(rec1)
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          3
##
## Training data contained 3000 data points and 2 incomplete rows.
##
## Operations:
##
## Bagged tree imputation for budget, popularity, runtime [trained]
## Removing rows with NA values in budget, popularity, runtime [trained]
```

```
d_train_baked <-
  rec1 %>%
  prep() %>%
  bake(new_data = NULL)

glimpse(d_train_baked)
```

```
## Rows: 3,000
## Columns: 4
## $ budget     <dbl> 1.40e+07, 4.00e+07, 3.30e+06, 1.20e+06, 0.00e+00…
## $ popularity <dbl> 6.575393, 8.248895, 64.299990, 3.174936, 1.14807…
## $ runtime    <dbl> 93, 113, 105, 122, 118, 83, 92, 84, 100, 91, 119…
## $ revenue    <dbl> 12314651, 95149435, 13092000, 16000000, 3923970,…
```

Fehlende Werte noch übrig?

```
library(easystats)
describe_distribution(d_train_baked) %>%
  select(Variable, n_Missing)
```

```
## Variable   | n_Missing
## ----------------------
## budget     |         0
## popularity |         0
## runtime    |         0
## revenue    |         0
```

Modell 1: RF

```
model1 <- rand_forest(mtry = tune(),
                      trees = tune(),
                      min_n = tune()) %>%
           set_engine('ranger') %>%
           set_mode('regression')
```

Workflow 1

```
wf1 <-
  workflow() %>%
  add_model(model1) %>%
  add_recipe(rec1)
```

Modell fitten (und tunen)

```
doParallel::registerDoParallel(4)
tic()
rf_fit1 <-
  wf1 %>%
  tune_grid(resamples = cv_scheme)
toc()
```

```
## 69.079 sec elapsed
```

```
rf_fit1[[".notes"]][1]
```

```
## [[1]]
## # A tibble: 0 × 3
## # … with 3 variables: location <chr>, type <chr>, note <chr>
```

Bester Kandidat

```
select_best(rf_fit1)
```

```
## Warning: No value of `metric` was given; metric 'rmse' will be used.
```

```
## # A tibble: 1 × 4
##    mtry trees min_n .config
##   <int> <int> <int> <chr>
## 1     1  1124     4 Preprocessor1_Model10
```

Workflow Finalisieren

```
wf_best <-
  wf1 %>%
  finalize_workflow(parameters = select_best(rf_fit1))
```

```
## Warning: No value of `metric` was given; metric 'rmse' will be used.
```

Final Fit

```
fit1_final <-
  wf_best %>%
  fit(d_train)

fit1_final
```

```
## == Workflow [trained] ==========================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------
## 2 Recipe Steps
##
## • step_impute_bag()
## • step_naomit()
##
## -- Model -----------------------------------------------------
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~1L,      x), num.trees = ~1124L, min.node.size = min_rows(~4L, x),
```

```
##
## Type:                             Regression
## Number of trees:                  1124
## Sample size:                      3000
## Number of independent variables:  3
## Mtry:                             1
## Target node size:                 4
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       6.572476e+15
## R squared (OOB):                  0.6525283
```

```
preds <-
  fit1_final %>%
  predict(d_test)
```

Submission df

```
submission_df <-
  d_test %>%
  select(id) %>%
  bind_cols(preds) %>%
  rename(revenue = .pred)

head(submission_df)
```

```
## # A tibble: 6 × 2
##      id   revenue
##   <dbl>     <dbl>
## 1  3001  4417241.
## 2  3002  5509047.
## 3  3003 13764032.
## 4  3004 38653947.
## 5  3005  3940693.
## 6  3006 22111594.
```

Abspeichern und einreichen:

```
#write_csv(submission_df, file = "submission.csv")
```

Kaggle Score

Diese Submission erzielte einen Score von **Score: 2.76961** (RMSLE).

```
sol <-  2.76961
```

2. **Aufgabe**

Wir bearbeiten hier die Fallstudie TMDB Box Office Prediction - Can you predict a movie's worldwide box office revenue?, ein Kaggle-Prognosewettbewerb.

Ziel ist es, genaue Vorhersagen zu machen, in diesem Fall für Filme.

Die Daten können Sie von der Kaggle-Projektseite beziehen oder so:

```
d_train_path <- "https://raw.githubusercontent.com/sebastiansauer/Lehre/main/data/tmdb-box-office-prediction/train.csv"
d_test_path <- "https://raw.githubusercontent.com/sebastiansauer/Lehre/main/data/tmdb-box-office-prediction/test.csv"
```

Aufgabe

Reichen Sie bei Kaggle eine Submission für die Fallstudie ein! Berichten Sie den Kaggle-Score

Hinweise:

- Sie müssen sich bei Kaggle ein Konto anlegen (kostenlos und anonym möglich); alternativ können Sie sich mit einem Google-Konto anmelden.
- Berechnen Sie einen Entscheidungsbaum und einen Random-Forest.
- Tunen Sie nach Bedarf; verwenden Sie aber Default-Werte.
- Verwenden Sie Tidymodels.

Lösung

Vorbereitung

```
library(tidyverse)
library(tidymodels)
library(tictoc)

d_train <- read_csv(d_train_path)
d_test <- read_csv(d_test_path)

glimpse(d_train)
```

```
## Rows: 3,000
## Columns: 23
## $ id                    <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12…
## $ belongs_to_collection <chr> "[{'id': 313576, 'name': 'Hot Tub Tim…
## $ budget                <dbl> 1.40e+07, 4.00e+07, 3.30e+06, 1.20e+0…
## $ genres                <chr> "[{'id': 35, 'name': 'Comedy'}]", "[{…
## $ homepage              <chr> NA, NA, "http://sonyclassics.com/whip…
## $ imdb_id               <chr> "tt2637294", "tt0368933", "tt2582802"…
## $ original_language     <chr> "en", "en", "en", "hi", "ko", "en", "…
## $ original_title        <chr> "Hot Tub Time Machine 2", "The Prince…
## $ overview              <chr> "When Lou, who has become the \"fathe…
## $ popularity            <dbl> 6.575393, 8.248895, 64.299990, 3.1749…
## $ poster_path           <chr> "/tQtWuwvMf0hCc2QR2tkolwl7c3c.jpg", "…
## $ production_companies  <chr> "[{'name': 'Paramount Pictures', 'id'…
## $ production_countries  <chr> "[{'iso_3166_1': 'US', 'name': 'Unite…
## $ release_date          <chr> "2/20/15", "8/6/04", "10/10/14", "3/9…
## $ runtime               <dbl> 93, 113, 105, 122, 118, 83, 92, 84, 1…
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'Englis…
## $ status                <chr> "Released", "Released", "Released", "…
## $ tagline               <chr> "The Laws of Space and Time are About…
## $ title                 <chr> "Hot Tub Time Machine 2", "The Prince…
## $ Keywords              <chr> "[{'id': 4379, 'name': 'time travel'}…
## $ cast                  <chr> "[{'cast_id': 4, 'character': 'Lou', …
## $ crew                  <chr> "[{'credit_id': '59ac067c92514107af02…
## $ revenue               <dbl> 12314651, 95149435, 13092000, 1600000…
```

```
glimpse(d_test)
```

```
## Rows: 4,398
## Columns: 22
## $ id                    <dbl> 3001, 3002, 3003, 3004, 3005, 3006, 3…
## $ belongs_to_collection <chr> "[{'id': 34055, 'name': 'Pokémon Coll…
## $ budget                <dbl> 0.00e+00, 8.80e+04, 0.00e+00, 6.80e+0…
## $ genres                <chr> "[{'id': 12, 'name': 'Adventure'}, {'…
## $ homepage              <chr> "http://www.pokemon.com/us/movies/mov…
## $ imdb_id               <chr> "tt1226251", "tt0051380", "tt0118556"…
## $ original_language     <chr> "ja", "en", "en", "fr", "en", "en", "…
## $ original_title        <chr> "ディアルガVSパルキアVSダークライ", "…
## $ overview              <chr> "Ash and friends (this time accompani…
## $ popularity            <dbl> 3.851534, 3.559789, 8.085194, 8.59601…
## $ poster_path           <chr> "/tnftmLMemPLduW6MRyZE0ZUD19z.jpg", "…
## $ production_companies  <chr> NA, "[{'name': 'Woolner Brothers Pict…
## $ production_countries  <chr> "[{'iso_3166_1': 'JP', 'name': 'Japan…
## $ release_date          <chr> "7/14/07", "5/19/58", "5/23/97", "9/4…
## $ runtime               <dbl> 90, 65, 100, 130, 92, 121, 119, 77, 1…
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'Englis…
## $ status                <chr> "Released", "Released", "Released", "…
## $ tagline               <chr> "Somewhere Between Time & Space... A …
## $ title                 <chr> "Pokémon: The Rise of Darkrai", "Atta…
## $ Keywords              <chr> "[{'id': 11451, 'name': 'pok√©mon'}, …
## $ cast                  <chr> "[{'cast_id': 3, 'character': 'Tonio'…
## $ crew                  <chr> "[{'credit_id': '52fe44e7c3a368484e03…
```

Rezpet

Rezept definieren

```
rec1 <-
  recipe(revenue ~ ., data = d_train) %>%
  update_role(all_predictors(), new_role = "id") %>%
  update_role(popularity, runtime, revenue, budget) %>%
  update_role(revenue, new_role = "outcome") %>%
  step_mutate(budget = ifelse(budget < 10, 10, budget)) %>%
  step_log(budget) %>%
  step_impute_knn(all_predictors())
```

```
rec1
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##         id         19
##    outcome          1
##  predictor          3
##
## Operations:
##
## Variable mutation for ifelse(budget < 10, 10, budget)
## Log transformation on budget
## K-nearest neighbor imputation for all_predictors()
```

Check das Rezept

```
rec1_prepped <-
  prep(rec1, verbose = TRUE)
```

```
## oper 1 step mutate [training]
## oper 2 step log [training]
## oper 3 step impute knn [training]
## The retained training set is ~ 28.71 Mb  in memory.
```

```
rec1_prepped
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##         id         19
##    outcome          1
##  predictor          3
##
## Training data contained 3000 data points and 2793 incomplete rows.
##
## Operations:
##
## Variable mutation for ~ifelse(budget < 10, 10, budget) [trained]
## Log transformation on budget [trained]
## K-nearest neighbor imputation for popularity, runtime, budget [trained]
```

```
d_train_baked <-
  rec1_prepped %>%
  bake(new_data = NULL)

head(d_train_baked)
```

```
## # A tibble: 6 × 23
##      id belongs_to_collection          budget genres homepage imdb_id
##   <dbl> <fct>                          <dbl> <fct>  <fct>    <fct>
## 1     1 [{'id': 313576, 'name': 'Hot …  16.5  [{'id… <NA>     tt2637…
## 2     2 [{'id': 107674, 'name': 'The …  17.5  [{'id… <NA>     tt0368…
## 3     3 <NA>                            15.0  [{'id… http://… tt2582…
## 4     4 <NA>                            14.0  [{'id… http://… tt1821…
## 5     5 <NA>                             2.30 [{'id… <NA>     tt1380…
## 6     6 <NA>                            15.9  [{'id… <NA>     tt0093…
## # … with 17 more variables: original_language <fct>,
## #   original_title <fct>, overview <fct>, popularity <dbl>,
## #   poster_path <fct>, production_companies <fct>,
## #   production_countries <fct>, release_date <fct>, runtime <dbl>,
## #   spoken_languages <fct>, status <fct>, tagline <fct>,
## #   title <fct>, Keywords <fct>, cast <fct>, crew <fct>,
## #   revenue <dbl>
```

Die AV-Spalte sollte leer sein:

```
bake(rec1_prepped, new_data = head(d_test), all_outcomes())
```

```
## # A tibble: 6 × 0
```

```
d_train_baked %>%
  map_df(~ sum(is.na(.)))
```

```
## # A tibble: 1 × 23
##      id belongs_to_collection budget genres homepage imdb_id
##   <int>                <int>  <int>  <int>    <int>   <int>
## 1     0                 2396      0      7     2054       0
## # … with 17 more variables: original_language <int>,
## #   original_title <int>, overview <int>, popularity <int>,
## #   poster_path <int>, production_companies <int>,
## #   production_countries <int>, release_date <int>, runtime <int>,
## #   spoken_languages <int>, status <int>, tagline <int>,
## #   title <int>, Keywords <int>, cast <int>, crew <int>,
## #   revenue <int>
```

Keine fehlenden Werte mehr *in den Prädiktoren*.

Nach fehlenden Werten könnte man z.B. auch so suchen:

```
datawizard::describe_distribution(d_train_baked)
```

variable | mean | sd | iqr | range | skewness | kurtosis | n | n_missing

id | 1500.50 | 866.17 | 1500.50 | [1.00, 3000.00] | 0.00 | -1.20 | 3000 | 0 budget | 12.51 | 6.44 | 14.88 | [2.30, 19.76] | -0.87 | -1.09 | 3000 | 0 popularity | 8.46 | 12.10 | 6.88 | [1.00e-06, 294.34] | 14.38 | 280.10 | 3000 | 0 runtime | 107.85 | 22.08 | 24.00 | [0.00, 338.00] | 1.02 | 8.20 | 3000 | 0 revenue | 6.67e+07 | 1.38e+08 | 6.66e+07 | [1.00, 1.52e+09] | 4.54 | 27.78 | 3000 | 0

So bekommt man gleich noch ein paar Infos über die Verteilung der Variablen. Praktische Sache.

Das Test-Sample backen wir auch mal:

```
d_test_baked <-
  bake(rec1_prepped, new_data = d_test)

d_test_baked %>%
  head()
```

```
## # A tibble: 6 × 22
##      id belongs_to_collection          budget genres homepage imdb_id
##   <dbl> <fct>                          <dbl> <fct>  <fct>    <fct>
## 1  3001 [{'id': 34055, 'name': 'Pokém…  2.30 [{'id… <NA>     <NA>
## 2  3002 <NA>                           11.4  [{'id… <NA>     <NA>
## 3  3003 <NA>                            2.30 [{'id… <NA>     <NA>
## 4  3004 <NA>                           15.7  <NA>   <NA>     <NA>
## 5  3005 <NA>                           14.5  [{'id… <NA>     <NA>
## 6  3006 <NA>                            2.30 [{'id… <NA>     <NA>
## # … with 16 more variables: original_language <fct>,
```

```
## #   original_title <fct>, overview <fct>, popularity <dbl>,
## #   poster_path <fct>, production_companies <fct>,
## #   production_countries <fct>, release_date <fct>, runtime <dbl>,
## #   spoken_languages <fct>, status <fct>, tagline <fct>,
## #   title <fct>, Keywords <fct>, cast <fct>, crew <fct>
```

Kreuzvalidierung

```
cv_scheme <- vfold_cv(d_train,
                      v = 5,
                      repeats = 3)
```

Modelle

Baum

```
mod_tree <-
  decision_tree(cost_complexity = tune(),
                tree_depth = tune(),
                mode = "regression")
```

Random Forest

```
doParallel::registerDoParallel()

mod_rf <-
  rand_forest(mtry = tune(),
              min_n = tune(),
              trees = 1000,
              mode = "regression") %>%
  set_engine("ranger", num.threads = 4)
```

Workflows

```
wf_tree <-
  workflow() %>%
  add_model(mod_tree) %>%
  add_recipe(rec1)

wf_rf <-
  workflow() %>%
  add_model(mod_rf) %>%
  add_recipe(rec1)
```

Fitten und tunen

Tree

```
tic()
tree_fit <-
  wf_tree %>%
  tune_grid(
    resamples = cv_scheme,
    grid = 2
  )
toc()
```

```
## 7.227 sec elapsed
```

Hilfe zu `tune_grid()` bekommt man hier.

```
tree_fit
```

```
## # Tuning results
## # 5-fold cross-validation repeated 3 times
## # A tibble: 15 × 5
##    splits             id      id2   .metrics          .notes
##    <list>             <chr>   <chr> <list>           <list>
##  1 <split [2400/600]> Repeat1 Fold1 <tibble [4 × 6]> <tibble [0 × 3]>
##  2 <split [2400/600]> Repeat1 Fold2 <tibble [4 × 6]> <tibble [0 × 3]>
##  3 <split [2400/600]> Repeat1 Fold3 <tibble [4 × 6]> <tibble [0 × 3]>
##  4 <split [2400/600]> Repeat1 Fold4 <tibble [4 × 6]> <tibble [0 × 3]>
##  5 <split [2400/600]> Repeat1 Fold5 <tibble [4 × 6]> <tibble [0 × 3]>
##  6 <split [2400/600]> Repeat2 Fold1 <tibble [4 × 6]> <tibble [0 × 3]>
##  7 <split [2400/600]> Repeat2 Fold2 <tibble [4 × 6]> <tibble [0 × 3]>
##  8 <split [2400/600]> Repeat2 Fold3 <tibble [4 × 6]> <tibble [0 × 3]>
##  9 <split [2400/600]> Repeat2 Fold4 <tibble [4 × 6]> <tibble [0 × 3]>
## 10 <split [2400/600]> Repeat2 Fold5 <tibble [4 × 6]> <tibble [0 × 3]>
## 11 <split [2400/600]> Repeat3 Fold1 <tibble [4 × 6]> <tibble [0 × 3]>
## 12 <split [2400/600]> Repeat3 Fold2 <tibble [4 × 6]> <tibble [0 × 3]>
## 13 <split [2400/600]> Repeat3 Fold3 <tibble [4 × 6]> <tibble [0 × 3]>
## 14 <split [2400/600]> Repeat3 Fold4 <tibble [4 × 6]> <tibble [0 × 3]>
## 15 <split [2400/600]> Repeat3 Fold5 <tibble [4 × 6]> <tibble [0 × 3]>
```

Steht was in den `.notes`?

```
tree_fit[[".notes"]][[2]]
```

```
## # A tibble: 0 × 3
## # … with 3 variables: location <chr>, type <chr>, note <chr>
```

Nein.

```
collect_metrics(tree_fit)
```

```
## # A tibble: 4 × 8
##   cost_complexity tree_depth .metric .estimator    mean     n std_err
##             <dbl>      <int> <chr>   <chr>        <dbl> <int>   <dbl>
## 1        5.38e-10         13 rmse    standard   8.93e+7    15 1.59e+6
## 2        5.38e-10         13 rsq     standard   5.82e-1    15 1.41e-2
## 3        5.23e- 6          3 rmse    standard   9.09e+7    15 1.58e+6
## 4        5.23e- 6          3 rsq     standard   5.58e-1    15 1.77e-2
## # … with 1 more variable: .config <chr>
```

```
show_best(tree_fit)
```

```
## Warning: No value of `metric` was given; metric 'rmse' will be used.
```

```
## # A tibble: 2 × 8
##   cost_complexity tree_depth .metric .estimator    mean     n std_err
##             <dbl>      <int> <chr>   <chr>        <dbl> <int>   <dbl>
## 1        5.38e-10         13 rmse    standard    8.93e7    15  1.59e6
## 2        5.23e- 6          3 rmse    standard    9.09e7    15  1.58e6
## # … with 1 more variable: .config <chr>
```

Finalisieren

```
best_tree_wf <-
  wf_tree %>%
  finalize_workflow(select_best(tree_fit))
```

```
## Warning: No value of `metric` was given; metric 'rmse' will be used.
```

```
best_tree_wf
```

```
## == Workflow ====================================================
## Preprocessor: Recipe
## Model: decision_tree()
##
## -- Preprocessor ----------------------------------------------------
## 3 Recipe Steps
##
## • step_mutate()
## • step_log()
## • step_impute_knn()
##
## -- Model -----------------------------------------------------------
## Decision Tree Model Specification (regression)
##
## Main Arguments:
##   cost_complexity = 5.37967376130334e-10
##   tree_depth = 13
##
## Computational engine: rpart
```

```
tree_last_fit <-
  fit(best_tree_wf, data = d_train)
```

```
tree_last_fit
```

```
## == Workflow [trained] ==========================================
## Preprocessor: Recipe
## Model: decision_tree()
##
## -- Preprocessor ----------------------------------------------------
## 3 Recipe Steps
##
## • step_mutate()
## • step_log()
## • step_impute_knn()
##
## -- Model -----------------------------------------------------------
## n= 3000
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##     1) root 3000 5.672651e+19   66725850.0
##       2) budget< 18.32631 2845 1.958584e+19   46935270.0
##         4) budget< 17.19976 2252 5.443953e+18   25901120.0
##           8) popularity< 9.734966 1745 1.665118e+18   17076460.0
##            16) popularity< 5.761331 1019 3.184962e+17    8793730.0
##              32) budget< 15.44456 782 1.408243e+17    6074563.0
##                64) popularity< 1.517383 293 1.907705e+16    3025921.0
##                 128) runtime>=46.5 284 6.431604e+15    2487837.0
##                   256) budget< 14.54573 248 5.220631e+15    2103707.0
##                     512) runtime< 96.5 118 6.571642e+14    1203847.0
##                      1024) popularity< 1.140496 99 3.624224e+14     960819.7
##                        2048) popularity< 0.18234 26 2.613956e+12     283857.7
##                          4096) budget< 10.99014 19 1.656689e+12     212028.9 *
##                          4097) budget>=10.99014 7 5.931626e+11     478821.6 *
```

```
##                         2049) popularity>=0.18234 73 3.436495e+14    1201929.0
##                           4098) runtime< 92 48 1.563662e+14     942088.9
##                             8196) popularity< 0.698595 23 1.197191e+13     472543.1 *
##                             8197) popularity>=0.698595 25 1.346582e+14    1374071.0 *
##                           4099) runtime>=92 25 1.778200e+14    1700823.0
##                             8198) popularity>=0.319976 17 3.705079e+13    1197195.0 *
##                             8199) popularity< 0.319976 8 1.272946e+14    2771033.0 *
##                       1025) popularity>=1.140496 19 2.584278e+14    2470148.0 *
##                      513) runtime>=96.5 130 4.381186e+15    2920503.0
##                       1026) popularity< 0.4028185 33 1.196999e+14    1118933.0
##                         2052) runtime< 104.5 13 2.411497e+11     102104.2 *
##                         2053) runtime>=104.5 20 9.728066e+13    1779872.0
##                           4106) runtime>=121.5 10 9.005154e+12     960192.6 *
##                           4107) runtime< 121.5 10 7.483801e+13    2599552.0 *
##                       1027) popularity>=0.4028185 97 4.117942e+15    3533408.0
##                         2054) popularity>=0.4693685 90 3.010532e+15    3196752.0
##                           4108) popularity< 0.886855 40 4.025034e+14    1971924.0
##                             8216) popularity< 0.565074 9 3.732993e+11     254432.2 *
##                             8217) popularity>=0.565074 31 3.678747e+14    2470550.0 *
##                           4109) popularity>=0.886855 50 2.500013e+15    4176615.0
##                             8218) popularity>=1.016719 40 1.691925e+15    3486101.0 *
##                             8219) popularity< 1.016719 10 7.127265e+14    6938669.0 *
##                         2055) popularity< 0.4693685 7 9.660623e+14    7861843.0 *
##                     257) budget>=14.54573 36 9.222880e+14    5134069.0
##                       514) budget>=14.95414 19 1.808518e+14    2918740.0 *
##                       515) budget< 14.95414 17 5.439742e+14    7610024.0 *
##                  129) runtime< 46.5 9 9.968486e+15   20005440.0 *
##                 65) popularity>=1.517383 489 1.173924e+17    7901255.0
##                  130) runtime< 102.5 275 2.547281e+16    5299779.0
##                    260) popularity< 4.655744 202 1.162337e+16    3865514.0
##                      520) runtime>=71.5 195 8.658935e+15    3508301.0
##
## ...
## and 420 more lines.
```

Vorhersage Test-Sample

```
predict(tree_last_fit, new_data = d_test)
```

```
## # A tibble: 4,398 × 1
##         .pred
##         <dbl>
##  1   2031852
##  2  13816447.
##  3   2447879.
##  4  48403371.
##  5  10710904.
##  6   8044510.
##  7  10296873.
##  8  63399160.
##  9  33470076.
## 10 372118475.
## # … with 4,388 more rows
```

RF

Fitten und Tunen

Um Rechenzeit zu sparen, kann man das Objekt, wenn einmal berechnet, abspeichern unter `result_obj_path` auf der Festplatte und beim nächsten Mal importieren, das geht schneller als neu berechnen.

In diesem Fall hat `result_obj_path` den Inhalt tmbd_rf_fit1.rds.

```
if (file.exists(result_obj_path)) {
  rf_fit <- read_rds(result_obj_path)
} else {
  tic()
  rf_fit <-
    wf_rf %>%
    tune_grid(
      resamples = cv_scheme)
  toc()
}
```

Achtung Ein Ergebnisobjekt von der Festplatte zu laden ist *gefährlich*. Wenn Sie Ihr Modell verändern, aber vergessen, das Objekt auf der Festplatte zu aktualisieren, werden Ihre Ergebnisse falsch sein (da auf dem veralteten Objekt beruhend), ohne dass Sie durch eine Fehlermeldung von R gewarnt würden!

So kann man das Ergebnisobjekt auf die Festplatte schreiben:

```
#write_rds(rf_fit, file = "objects/tmbd_rf_fit1.rds")
```

```
collect_metrics(rf_fit)
```

```
## # A tibble: 20 × 8
##     mtry min_n .metric .estimator         mean     n  std_err .config
##    <int> <int> <chr>   <chr>             <dbl> <int>    <dbl> <chr>
##  1     2    15 rmse    standard   82814784.       15  1.71e+6 Prepro…
##  2     2    15 rsq     standard          0.643    15  1.15e-2 Prepro…
##  3     1    34 rmse    standard   82884640.       15  1.82e+6 Prepro…
##  4     1    34 rsq     standard          0.646    15  1.15e-2 Prepro…
##  5     1    23 rmse    standard   82457030.       15  1.78e+6 Prepro…
##  6     1    23 rsq     standard          0.648    15  1.15e-2 Prepro…
```

```
##  7     1    29 rmse    standard   82726287.       15  1.78e+6 Prepro…
##  8     1    29 rsq     standard          0.646    15  1.13e-2 Prepro…
##  9     2    27 rmse    standard   82386320.       15  1.74e+6 Prepro…
## 10     2    27 rsq     standard          0.645    15  1.21e-2 Prepro…
## 11     3    20 rmse    standard   83010493.       15  1.75e+6 Prepro…
## 12     3    20 rsq     standard          0.641    15  1.23e-2 Prepro…
## 13     3    10 rmse    standard   83920729.       15  1.72e+6 Prepro…
## 14     3    10 rsq     standard          0.634    15  1.22e-2 Prepro…
## 15     2    40 rmse    standard   82786794.       15  1.78e+6 Prepro…
## 16     2    40 rsq     standard          0.642    15  1.22e-2 Prepro…
## 17     2     9 rmse    standard   83237809.       15  1.71e+6 Prepro…
## 18     2     9 rsq     standard          0.640    15  1.14e-2 Prepro…
## 19     2     3 rmse    standard   83861944.       15  1.64e+6 Prepro…
## 20     2     3 rsq     standard          0.635    15  1.12e-2 Prepro…
```

```
select_best(rf_fit)
```

```
## Warning: No value of `metric` was given; metric 'rmse' will be used.
```

```
## # A tibble: 1 × 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     2    27 Preprocessor1_Model05
```

Finalisieren

```
final_wf <-
  wf_rf %>%
  finalize_workflow(select_best(rf_fit))
```

```
## Warning: No value of `metric` was given; metric 'rmse' will be used.
```

```
final_fit <-
  fit(final_wf, data = d_train)

final_preds <-
  final_fit %>%
  predict(new_data = d_test) %>%
  bind_cols(d_test)

submission <-
  final_preds %>%
  select(id, revenue = .pred)
```

Abspeichern und einreichen:

```
write_csv(submission, file = "submission.csv")
```

Kaggle Score

Diese Submission erzielte einen Score von **2.7664** (RMSLE).

```
sol <- 2.7664
```

3. **Aufgabe**

Wir bearbeiten hier die Fallstudie TMDB Box Office Prediction - Can you predict a movie's worldwide box office revenue?, ein Kaggle-Prognosewettbewerb.

Ziel ist es, genaue Vorhersagen zu machen, in diesem Fall für Filme.

Die Daten können Sie von der Kaggle-Projektseite beziehen oder so:

```
d_train_path <- "https://raw.githubusercontent.com/sebastiansauer/Lehre/main/data/tmdb-box-office-prediction/train.csv"
d_test_path <- "https://raw.githubusercontent.com/sebastiansauer/Lehre/main/data/tmdb-box-office-prediction/test.csv"
```

Aufgabe

Reichen Sie bei Kaggle eine Submission für die Fallstudie ein! Berichten Sie den Score!

Hinweise:

- Sie müssen sich bei Kaggle ein Konto anlegen (kostenlos und anonym möglich); alternativ können Sie sich mit einem Google-Konto anmelden.
- Verwenden Sie *mehrere, und zwar folgende Algorithmen*: Random Forest, Boosting, lineare Regression. Tipp: Ein Workflow-Set ist hilfreich.
- Logarithmieren Sie `budget`.
- Betreiben Sie Feature Engineering, zumindest etwas. Insbesondere sollten Sie den Monat und das Jahr aus dem Datum extrahieren und als Features (Prädiktoren) nutzen.
- Verwenden Sie `tidymodels`.
- Die Zielgröße ist `revenue` in Dollars; nicht in "Log-Dollars". Sie müssen also rücktransformieren, falls Sie `revenue` logarithmiert haben.

Lösung

Vorbereitung

```
library(tidyverse)
library(tidymodels)
library(tictoc)  # Rechenzeit messen
#library(Metrics)
library(lubridate)  # Datumsangaben
library(VIM)  # fehlende Werte
library(visdat)  # Datensatz visualisieren

d_train_raw <- read_csv(d_train_path)
d_test <- read_csv(d_test_path)
```

Mal einen Blick werfen:

```
glimpse(d_train_raw)
```

```
## Rows: 3,000
## Columns: 23
## $ id                    <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12…
## $ belongs_to_collection <chr> "[{'id': 313576, 'name': 'Hot Tub Tim…
## $ budget                <dbl> 1.40e+07, 4.00e+07, 3.30e+06, 1.20e+0…
## $ genres                <chr> "[{'id': 35, 'name': 'Comedy'}]", "[{…
## $ homepage              <chr> NA, NA, "http://sonyclassics.com/whip…
## $ imdb_id               <chr> "tt2637294", "tt0368933", "tt2582802"…
## $ original_language     <chr> "en", "en", "en", "hi", "ko", "en", "…
## $ original_title        <chr> "Hot Tub Time Machine 2", "The Prince…
## $ overview              <chr> "When Lou, who has become the \"fathe…
## $ popularity            <dbl> 6.575393, 8.248895, 64.299990, 3.1749…
## $ poster_path           <chr> "/tQtWuwvMf0hCc2QR2tkolwl7c3c.jpg", "…
## $ production_companies  <chr> "[{'name': 'Paramount Pictures', 'id'…
## $ production_countries  <chr> "[{'iso_3166_1': 'US', 'name': 'Unite…
## $ release_date          <chr> "2/20/15", "8/6/04", "10/10/14", "3/9…
## $ runtime               <dbl> 93, 113, 105, 122, 118, 83, 92, 84, 1…
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'Englis…
## $ status                <chr> "Released", "Released", "Released", "…
## $ tagline               <chr> "The Laws of Space and Time are About…
## $ title                 <chr> "Hot Tub Time Machine 2", "The Prince…
## $ Keywords              <chr> "[{'id': 4379, 'name': 'time travel'}…
## $ cast                  <chr> "[{'cast_id': 4, 'character': 'Lou', …
## $ crew                  <chr> "[{'credit_id': '59ac067c92514107af02…
## $ revenue               <dbl> 12314651, 95149435, 13092000, 1600000…
```

```
glimpse(d_test)
```

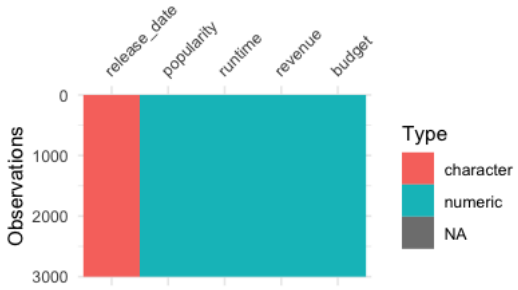
```
## Rows: 4,398
## Columns: 22
## $ id                    <dbl> 3001, 3002, 3003, 3004, 3005, 3006, 3…
## $ belongs_to_collection <chr> "[{'id': 34055, 'name': 'Pokémon Coll…
## $ budget                <dbl> 0.00e+00, 8.80e+04, 0.00e+00, 6.80e+0…
## $ genres                <chr> "[{'id': 12, 'name': 'Adventure'}, {'…
## $ homepage              <chr> "http://www.pokemon.com/us/movies/mov…
## $ imdb_id               <chr> "tt1226251", "tt0051380", "tt0118556"…
## $ original_language     <chr> "ja", "en", "en", "fr", "en", "en", "…
## $ original_title        <chr> "ディアルガVSパルキアVSダークライ", "…
## $ overview              <chr> "Ash and friends (this time accompani…
## $ popularity            <dbl> 3.851534, 3.559789, 8.085194, 8.59601…
## $ poster_path           <chr> "/tnftmLMemPLduW6MRyZE0ZUD19z.jpg", "…
## $ production_companies  <chr> NA, "[{'name': 'Woolner Brothers Pict…
## $ production_countries  <chr> "[{'iso_3166_1': 'JP', 'name': 'Japan…
## $ release_date          <chr> "7/14/07", "5/19/58", "5/23/97", "9/4…
## $ runtime               <dbl> 90, 65, 100, 130, 92, 121, 119, 77, 1…
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'Englis…
## $ status                <chr> "Released", "Released", "Released", "…
## $ tagline               <chr> "Somewhere Between Time & Space... A …
## $ title                 <chr> "Pokémon: The Rise of Darkrai", "Atta…
## $ Keywords              <chr> "[{'id': 11451, 'name': 'pok√©mon'}, …
## $ cast                  <chr> "[{'cast_id': 3, 'character': 'Tonio'…
## $ crew                  <chr> "[{'credit_id': '52fe44e7c3a368484e03…
```

Train-Set verschlanken

```
d_train <-
  d_train_raw %>%
  select(popularity, runtime, revenue, budget, release_date)
```

Datensatz kennenlernen

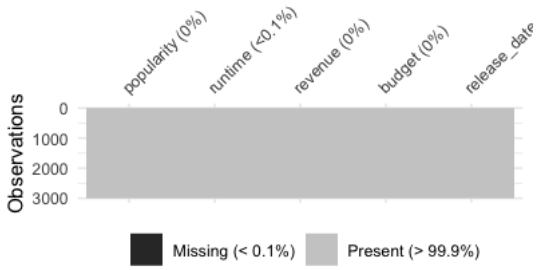
```
library(visdat)
vis_dat(d_train)
```



Fehlende Werte prüfen

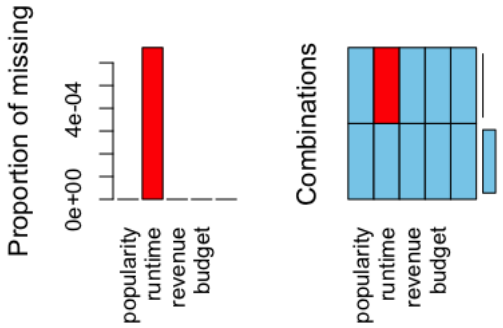
Welche Spalten haben viele fehlende Werte?

```
vis_miss(d_train)
```



Mit `{VIM}` kann man einen Datensatz gut auf fehlende Werte hin untersuchen:

```
aggr(d_train)
```



Rezept

Rezept definieren

```
rec1 <-
  recipe(revenue ~ ., data = d_train) %>%
  #update_role(all_predictors(), new_role = "id") %>%
  #update_role(popularity, runtime, revenue, budget, original_language) %>%
  #update_role(revenue, new_role = "outcome") %>%
  step_mutate(budget = if_else(budget < 10, 10, budget)) %>%
  step_log(budget) %>%
  step_mutate(release_date = mdy(release_date)) %>%
  step_date(release_date, features = c("year", "month"), keep_original_cols = FALSE) %>%
  step_impute_knn(all_predictors()) %>%
  step_dummy(all_nominal())

rec1
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          4
##
## Operations:
##
## Variable mutation for if_else(budget < 10, 10, budget)
```

```
## Log transformation on budget
## Variable mutation for mdy(release_date)
## Date features from release_date
## K-nearest neighbor imputation for all_predictors()
## Dummy variables from all_nominal()
```

```
tidy(rec1)
```

```
## # A tibble: 6 × 6
##   number operation type       trained skip  id
##    <int> <chr>     <chr>      <lgl>   <lgl> <chr>
## 1      1 step      mutate     FALSE   FALSE mutate_6ju5z
## 2      2 step      log        FALSE   FALSE log_KAivB
## 3      3 step      mutate     FALSE   FALSE mutate_U0lpz
## 4      4 step      date       FALSE   FALSE date_2scL5
## 5      5 step      impute_knn FALSE   FALSE impute_knn_4OuDg
## 6      6 step      dummy      FALSE   FALSE dummy_bQ4ok
```

Check das Rezept

```
prep(rec1, verbose = TRUE)
```

```
## oper 1 step mutate [training]
## oper 2 step log [training]
## oper 3 step mutate [training]
## oper 4 step date [training]
## oper 5 step impute knn [training]
## oper 6 step dummy [training]
## The retained training set is ~ 0.38 Mb  in memory.
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          4
##
## Training data contained 3000 data points and 2 incomplete rows.
##
## Operations:
##
## Variable mutation for ~if_else(budget < 10, 10, budget) [trained]
## Log transformation on budget [trained]
## Variable mutation for ~mdy(release_date) [trained]
## Date features from release_date [trained]
## K-nearest neighbor imputation for runtime, budget, release_date_year,... [trained]
## Dummy variables from release_date_month [trained]
```

```
d_train_baked <-
prep(rec1) %>%
  bake(new_data = NULL)
```

```
d_train_baked
```

```
## # A tibble: 3,000 × 16
##    popularity runtime budget  revenue release_date_year
##         <dbl>   <dbl>  <dbl>    <dbl>             <dbl>
##  1      6.58       93  16.5  12314651              2015
##  2      8.25      113  17.5  95149435              2004
##  3     64.3       105  15.0  13092000              2014
##  4      3.17      122  14.0  16000000              2012
##  5      1.15      118   2.30  3923970              2009
##  6      0.743      83  15.9   3261638              1987
##  7      7.29       92  16.5  85446075              2012
##  8      1.95       84   2.30  2586511              2004
##  9      6.90      100   2.30 34327391              1996
## 10      4.67       91  15.6  18750246              2003
## # … with 2,990 more rows, and 11 more variables:
## #   release_date_month_Feb <dbl>, release_date_month_Mar <dbl>,
## #   release_date_month_Apr <dbl>, release_date_month_May <dbl>,
## #   release_date_month_Jun <dbl>, release_date_month_Jul <dbl>,
## #   release_date_month_Aug <dbl>, release_date_month_Sep <dbl>,
## #   release_date_month_Oct <dbl>, release_date_month_Nov <dbl>,
## #   release_date_month_Dec <dbl>
```

```
d_train_baked %>%
  map_df(~ sum(is.na(.)))
```

```
## # A tibble: 1 × 16
##   popularity runtime budget revenue release_date_ye… release_date_mo…
##        <int>   <int>  <int>   <int>            <int>            <int>
## 1          0       0      0       0                0                0
## # … with 10 more variables: release_date_month_Mar <int>,
## #   release_date_month_Apr <int>, release_date_month_May <int>,
## #   release_date_month_Jun <int>, release_date_month_Jul <int>,
## #   release_date_month_Aug <int>, release_date_month_Sep <int>,
## #   release_date_month_Oct <int>, release_date_month_Nov <int>,
## #   release_date_month_Dec <int>
```

Keine fehlenden Werte mehr *in den Prädiktoren*.

Nach fehlenden Werten könnte man z.B. auch so suchen:

```
datawizard::describe_distribution(d_train_baked)
```

```
## Variable                |     Mean |       SD |      IQR |              Range | Skewness | Kurtosis |    n | n_Missing
## ------------------------------------------------------------------------------------------------------------------------
## popularity              |     8.46 |    12.10 |     6.88 | [1.00e-06, 294.34] |    14.38 |   280.10 | 3000 |         0
## runtime                 |   107.84 |    22.09 |    24.00 |     [0.00, 338.00] |     1.02 |     8.19 | 3000 |         0
## budget                  |    12.51 |     6.44 |    14.88 |      [2.30, 19.76] |    -0.87 |    -1.09 | 3000 |         0
## revenue                 | 6.67e+07 | 1.38e+08 | 6.66e+07 |   [1.00, 1.52e+09] |     4.54 |    27.78 | 3000 |         0
## release_date_year       |  2004.58 |    15.48 |    17.00 | [1969.00, 2068.00] |     1.22 |     3.94 | 3000 |         0
## release_date_month_Feb  |     0.08 |     0.26 |     0.00 |       [0.00, 1.00] |     3.22 |     8.37 | 3000 |         0
## release_date_month_Mar  |     0.08 |     0.27 |     0.00 |       [0.00, 1.00] |     3.11 |     7.71 | 3000 |         0
## release_date_month_Apr  |     0.08 |     0.27 |     0.00 |       [0.00, 1.00] |     3.06 |     7.35 | 3000 |         0
## release_date_month_May  |     0.07 |     0.26 |     0.00 |       [0.00, 1.00] |     3.24 |     8.49 | 3000 |         0
## release_date_month_Jun  |     0.08 |     0.27 |     0.00 |       [0.00, 1.00] |     3.12 |     7.76 | 3000 |         0
## release_date_month_Jul  |     0.07 |     0.25 |     0.00 |       [0.00, 1.00] |     3.38 |     9.45 | 3000 |         0
## release_date_month_Aug  |     0.09 |     0.28 |     0.00 |       [0.00, 1.00] |     2.97 |     6.83 | 3000 |         0
## release_date_month_Sep  |     0.12 |     0.33 |     0.00 |       [0.00, 1.00] |     2.33 |     3.43 | 3000 |         0
## release_date_month_Oct  |     0.10 |     0.30 |     0.00 |       [0.00, 1.00] |     2.63 |     4.90 | 3000 |         0
## release_date_month_Nov  |     0.07 |     0.26 |     0.00 |       [0.00, 1.00] |     3.27 |     8.67 | 3000 |         0
## release_date_month_Dec  |     0.09 |     0.28 |     0.00 |       [0.00, 1.00] |     2.92 |     6.52 | 3000 |         0
```

So bekommt man gleich noch ein paar Infos über die Verteilung der Variablen. Praktische Sache.

Check Test-Sample

Das Test-Sample backen wir auch mal. Das hat *nur* den Zwecke, zu prüfen, ob unser Rezept auch richtig funktioniert. Das Preppen und Backen des Test-Samples wir *automatisch* von `predict()` bzw. `last_fit()` erledigt.

Wichtig: Wir preppen den Datensatz mit dem *Train-Sample*, auch wenn wir das Test-Sample backen wollen.

```
d_test_baked <-
  bake(rec1_prepped, new_data = d_test)
```

```
## Error in bake(rec1_prepped, new_data = d_test): object 'rec1_prepped' not found
```

```
d_test_baked %>%
  head()
```

```
## Error in head(.): object 'd_test_baked' not found
```

Kreuzvalidierung

```
cv_scheme <- vfold_cv(d_train,
                      v = 5,
                      repeats = 3)
```

Modelle

Baum

```
mod_tree <-
  decision_tree(cost_complexity = tune(),
                tree_depth = tune(),
                mode = "regression")
```

Random Forest

```
doParallel::registerDoParallel()
```

```
mod_rf <-
  rand_forest(mtry = tune(),
              min_n = tune(),
              trees = 1000,
              mode = "regression") %>%
  set_engine("ranger", num.threads = 4)
```

XGBoost

```
mod_boost <- boost_tree(mtry = tune(),
                        min_n = tune(),
                        trees = tune()) %>%
  set_engine("xgboost", nthreads = parallel::detectCores()) %>%
  set_mode("regression")
```

LM

```
mod_lm <-
  linear_reg()
```

Workflow-Set

```
preproc <- list(rec1 = rec1)
models <- list(tree1 = mod_tree, rf1 = mod_rf, boost1 = mod_boost, lm1 = mod_lm)
```

```
all_workflows <- workflow_set(preproc, models)
```

Fitten und tunen

Wenn man das Ergebnis-Objekt abgespeichert hat, dann kann man es einfach laden, spart Rechenzeit (der Tag ist kurz):

```
result_obj_file <- "tmdb_model_set.rds"
```

(Davon ausgehend, dass die Datei im Arbeitsverzeichnis liegt.)

```
if (file.exists(result_obj_file)) {
  tmdb_model_set <- read_rds(result_obj_file)
} else {
  tic()
  tmdb_model_set <-
    all_workflows %>%
    workflow_map(
      resamples = cv_scheme,
      grid = 10,
    #  metrics = metric_set(rmse),
      seed = 42,  # reproducibility
      verbose = TRUE)
  toc()
}
```

Um Rechenzeit zu sparen, kann man das Ergebnisobjekt abspeichern, dann muss man beim nächsten Mal nicht wieder von Neuem berechnen:

```
#write_rds(tmdb_model_set, "objects/tmdb_model_set.rds")
```

Finalisieren

Welcher Algorithmus schneidet am besten ab?

Genauer geagt, welches Modell, denn es ist ja nicht nur ein Algorithmus, sondern ein Algorithmus plus ein Rezept plus die Parameterinstatiierung plus ein spezifischer Datensatz.

```
tune::autoplot(tmdb_model_set) +
  theme(legend.position = "bottom")
```

R-Quadrat ist nicht entscheidend; `rmse` ist wichtiger.

Die Ergebnislage ist nicht ganz klar, aber einiges spricht für das Boosting-Modell, `rec1_boost1`.

```
tmdb_model_set %>%
  collect_metrics() %>%
  arrange(-mean) %>%
  head(10)
```

```
## # A tibble: 10 × 9
##    wflow_id    .config  preproc model .metric .estimator   mean     n
##    <chr>       <chr>    <chr>   <chr> <chr>   <chr>       <dbl> <int>
##  1 rec1_lm1    Preproc… recipe  line… rmse    standard   1.15e8    15
##  2 rec1_tree1  Preproc… recipe  deci… rmse    standard   1.12e8    15
##  3 rec1_rf1    Preproc… recipe  rand… rmse    standard   1.10e8    15
##  4 rec1_tree1  Preproc… recipe  deci… rmse    standard   9.46e7    15
##  5 rec1_tree1  Preproc… recipe  deci… rmse    standard   9.33e7    15
##  6 rec1_boost1 Preproc… recipe  boos… rmse    standard   9.30e7    15
##  7 rec1_boost1 Preproc… recipe  boos… rmse    standard   9.27e7    15
##  8 rec1_tree1  Preproc… recipe  deci… rmse    standard   9.21e7    15
##  9 rec1_tree1  Preproc… recipe  deci… rmse    standard   9.21e7    15
## 10 rec1_boost1 Preproc… recipe  boos… rmse    standard   9.21e7    15
## # … with 1 more variable: std_err <dbl>
```

```
best_model_params <-
extract_workflow_set_result(tmdb_model_set, "rec1_boost1") %>%
  select_best()
```

```
## Warning: No value of `metric` was given; metric 'rmse' will be used.
```

```
best_model_params
```

```
## # A tibble: 1 × 4
##    mtry trees min_n .config
##   <int> <int> <int> <chr>
## 1     6   100     4 Preprocessor1_Model04
```

Finalisieren

```
best_wf <-
all_workflows %>%
  extract_workflow("rec1_boost1")

best_wf
```

```
## == Workflow ====================================================
## Preprocessor: Recipe
## Model: boost_tree()
##
## -- Preprocessor ----------------------------------------------
## 6 Recipe Steps
##
## • step_mutate()
## • step_log()
## • step_mutate()
## • step_date()
## • step_impute_knn()
## • step_dummy()
##
## -- Model -----------------------------------------------------
## Boosted Tree Model Specification (regression)
##
## Main Arguments:
##   mtry = tune()
##   trees = tune()
##   min_n = tune()
##
## Engine-Specific Arguments:
##   nthreads = parallel::detectCores()
##
## Computational engine: xgboost
```

```
best_wf_finalized <-
  best_wf %>%
  finalize_workflow(best_model_params)

best_wf_finalized
```

```
## == Workflow ====================================================
## Preprocessor: Recipe
## Model: boost_tree()
##
## -- Preprocessor ----------------------------------------------
## 6 Recipe Steps
##
## • step_mutate()
## • step_log()
## • step_mutate()
## • step_date()
## • step_impute_knn()
## • step_dummy()
##
## -- Model -----------------------------------------------------
## Boosted Tree Model Specification (regression)
##
## Main Arguments:
##   mtry = 6
##   trees = 100
##   min_n = 4
##
## Engine-Specific Arguments:
##   nthreads = parallel::detectCores()
##
## Computational engine: xgboost
```

Final Fit

```
fit_final <-
  best_wf_finalized %>%
  fit(d_train)
```

```
## [00:20:51] WARNING: amalgamation/../src/learner.cc:576:
## Parameters: { "nthreads" } might not be used.
##
##   This could be a false alarm, with some parameters getting used by language bindings but
##   then being mistakenly passed down to XGBoost core, or some parameter actually being used
##   but getting flagged wrongly here. Please open an issue if you find any such cases.
```

```
fit_final
```

```
## == Workflow [trained] ==========================================
## Preprocessor: Recipe
## Model: boost_tree()
##
## -- Preprocessor ----------------------------------------------
## 6 Recipe Steps
##
```

```
## • step_mutate()
## • step_log()
## • step_mutate()
## • step_date()
## • step_impute_knn()
## • step_dummy()
##
## ── Model ───────────────────────────────────────────────────────
## ##### xgb.Booster
## raw: 340.4 Kb
## call:
##   xgboost::xgb.train(params = list(eta = 0.3, max_depth = 6, gamma = 0,
##     colsample_bytree = 1, colsample_bynode = 0.4, min_child_weight = 4L,
##     subsample = 1, objective = "reg:squarederror"), data = x$data,
##     nrounds = 100L, watchlist = x$watchlist, verbose = 0, nthreads = 8L,
##     nthread = 1)
## params (as set within xgb.train):
##   eta = "0.3", max_depth = "6", gamma = "0", colsample_bytree = "1", colsample_bynode = "0.4", min_child_weight = "4", subsample = "1'
## xgb.attributes:
##   niter
## callbacks:
##   cb.evaluation.log()
## # of features: 15
## niter: 100
## nfeatures : 15
## evaluation_log:
##     iter training_rmse
##        1     122301056
##        2     103897424
## ---
##       99      28913574
##      100      28644486
```

```
d_test$revenue <- NA

final_preds <-
  fit_final %>%
  predict(new_data = d_test) %>%
  bind_cols(d_test)
```

Submission

```
submission_df <-
  final_preds %>%
  select(id, revenue = .pred)
```

Abspeichern und einreichen:

```
write_csv(submission_df, file = "submission.csv")
```

Kaggle Score

Diese Submission erzielte einen Score von **4.79227** (RMSLE).

```
sol <- 4.79227
```

4. **Aufgabe**

   Wir bearbeiten hier die Fallstudie TMDB Box Office Prediction - Can you predict a movie's worldwide box office revenue?, ein Kaggle-Prognosewettbewerb.

   Ziel ist es, genaue Vorhersagen zu machen, in diesem Fall für Filme.

   Die Daten können Sie von der Kaggle-Projektseite beziehen oder so:

```
d_train_path <- "https://raw.githubusercontent.com/sebastiansauer/Lehre/main/data/tmdb-box-office-prediction/train.csv"
d_test_path <- "https://raw.githubusercontent.com/sebastiansauer/Lehre/main/data/tmdb-box-office-prediction/test.csv"
```

Aufgabe

Reichen Sie bei Kaggle eine Submission für die Fallstudie ein! Berichten Sie den Score!

Hinweise:

- Sie müssen sich bei Kaggle ein Konto anlegen (kostenlos und anonym möglich); alternativ können Sie sich mit einem Google-Konto anmelden.
- Halten Sie das Modell so *einfach* wie möglich. Verwenden Sie als Algorithmus die *lineare Regression* ohne weitere Schnörkel.
- Logarithmieren Sie `budget` und `revenue`.
- Minimieren Sie die Vorverarbeitung (`steps`) so weit als möglich.
- Verwenden Sie `tidymodels`.
- Die Zielgröße ist `revenue` in Dollars; nicht in "Log-Dollars". Sie müssen also rücktransformieren, wenn Sie `revenue` logarithmiert haben, bevor Sie Ihre Prognose einreichen.

Lösung

Vorbereitung

```
library(tidyverse)
library(tidymodels)

d_train_raw <- read_csv(d_train_path)
d_test_raw <- read_csv(d_test_path)

# d_test$revenue <- NA

d_train_backup <- d_train_raw
```

Mal einen Blick werfen:

```
glimpse(d_train_raw)
```

```
## Rows: 3,000
## Columns: 23
## $ id                    <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12…
## $ belongs_to_collection <chr> "[{'id': 313576, 'name': 'Hot Tub Tim…
## $ budget                <dbl> 1.40e+07, 4.00e+07, 3.30e+06, 1.20e+0…
## $ genres                <chr> "[{'id': 35, 'name': 'Comedy'}]", "[{…
## $ homepage              <chr> NA, NA, "http://sonyclassics.com/whip…
## $ imdb_id               <chr> "tt2637294", "tt0368933", "tt2582802"…
## $ original_language     <chr> "en", "en", "en", "hi", "ko", "en", "…
## $ original_title        <chr> "Hot Tub Time Machine 2", "The Prince…
## $ overview              <chr> "When Lou, who has become the \"fathe…
## $ popularity            <dbl> 6.575393, 8.248895, 64.299990, 3.1749…
## $ poster_path           <chr> "/tQtWuwvMf0hCc2QR2tkolwl7c3c.jpg", "…
## $ production_companies  <chr> "[{'name': 'Paramount Pictures', 'id'…
## $ production_countries  <chr> "[{'iso_3166_1': 'US', 'name': 'Unite…
## $ release_date          <chr> "2/20/15", "8/6/04", "10/10/14", "3/9…
## $ runtime               <dbl> 93, 113, 105, 122, 118, 83, 92, 84, 1…
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'Englis…
## $ status                <chr> "Released", "Released", "Released", "…
## $ tagline               <chr> "The Laws of Space and Time are About…
## $ title                 <chr> "Hot Tub Time Machine 2", "The Prince…
## $ Keywords              <chr> "[{'id': 4379, 'name': 'time travel'}…
## $ cast                  <chr> "[{'cast_id': 4, 'character': 'Lou', …
## $ crew                  <chr> "[{'credit_id': '59ac067c92514107af02…
## $ revenue               <dbl> 12314651, 95149435, 13092000, 1600000…
```

Train-Set verschlanken

```
d_train_raw_reduced <-
  d_train_raw %>%
  select(id, popularity, runtime, revenue, budget)
```

Test-Set verschlanken

```
d_test <-
  d_test_raw %>%
  select(id,popularity, runtime, budget)
```

Outcome logarithmieren

Der Outcome sollte *nicht* im Rezept transformiert werden (vgl. Part 3, S. 30, in dieser Unterlage).

```
d_train <-
  d_train_raw_reduced %>%
  mutate(revenue = if_else(revenue < 10, 10, revenue)) %>%
  mutate(revenue = log(revenue))
```

Prüfen, ob das funktioniert hat:

```
d_train$revenue %>% is.infinite() %>% any()
```

```
## [1] FALSE
```

Keine unendlichen Werte mehr

Fehlende Werte prüfen

Welche Spalten haben viele fehlende Werte?

```
sum_isna <- function(x) {sum(is.na(x))}

d_train %>%
  summarise(across(everything(), sum_isna))
```

```
## # A tibble: 1 × 5
##      id popularity runtime revenue budget
##   <int>      <int>   <int>   <int>  <int>
## 1     0          0       2       0      0
```

Rezept

Rezept definieren

```
rec2 <-
  recipe(revenue ~ ., data = d_train) %>%
  step_mutate(budget = ifelse(budget == 0, NA, budget)) %>%  # log mag keine 0
  step_log(budget) %>%
  step_impute_knn(all_predictors()) %>%
  step_dummy(all_nominal_predictors())  %>%
  update_role(id, new_role = "id")

rec2
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##         id          1
##    outcome          1
##  predictor          3
##
## Operations:
##
## Variable mutation for ifelse(budget == 0, NA, budget)
## Log transformation on budget
## K-nearest neighbor imputation for all_predictors()
## Dummy variables from all_nominal_predictors()
```

Schauen Sie mal, der Log mag keine Nullen:

```
x <- c(1,2, NA, 0)

log(x)
```

```
## [1] 0.0000000 0.6931472        NA      -Inf
```

Da $log\,(0) = -\infty$. Aus dem Grund wandeln wir 0 lieber in `NA` um.

```
tidy(rec2)
```

```
## # A tibble: 4 × 6
##   number operation type       trained skip  id
##    <int> <chr>     <chr>      <lgl>   <lgl> <chr>
## 1      1 step      mutate     FALSE   FALSE mutate_Uh6zl
## 2      2 step      log        FALSE   FALSE log_3cxNv
## 3      3 step      impute_knn FALSE   FALSE impute_knn_boz1h
## 4      4 step      dummy      FALSE   FALSE dummy_PsEBM
```

Check das Rezept

Wir berechnen das Rezept:

```
rec2_prepped <-
  prep(rec2, verbose = TRUE)
```

```
## oper 1 step mutate [training]
## oper 2 step log [training]
## oper 3 step impute knn [training]
## oper 4 step dummy [training]
## The retained training set is ~ 0.12 Mb  in memory.
```

```
rec2_prepped
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##         id          1
##    outcome          1
##  predictor          3
##
## Training data contained 3000 data points and 2 incomplete rows.
##
## Operations:
##
## Variable mutation for ~ifelse(budget == 0, NA, budget) [trained]
## Log transformation on budget [trained]
## K-nearest neighbor imputation for runtime, budget, popularity [trained]
## Dummy variables from <none> [trained]
```

Das ist noch *nicht* auf einen Datensatz angewendet! Lediglich die `steps` wurden *vorbereitet*, “präpariert”: z.B. “Diese Dummy-Variablen impliziert das Rezept”.

So sieht das dann aus, wenn man das *präparierte* Rezept auf das Train-Sample anwendet:

```
d_train_baked2 <-
  rec2_prepped %>%
  bake(new_data = NULL)

head(d_train_baked2)
```

```
## # A tibble: 6 × 5
##      id popularity runtime budget revenue
```

```
##   <dbl>      <dbl>   <dbl>  <dbl>   <dbl>
## 1     1      6.58       93   16.5    16.3
## 2     2      8.25      113   17.5    18.4
## 3     3     64.3       105   15.0    16.4
## 4     4      3.17      122   14.0    16.6
## 5     5      1.15      118   15.8    15.2
## 6     6      0.743      83   15.9    15.0
```

```
d_train_baked2 %>%
  map_df(sum_isna)
```

```
## # A tibble: 1 × 5
##      id popularity runtime budget revenue
##   <int>      <int>   <int>  <int>   <int>
## 1     0          0       0      0       0
```

Keine fehlenden Werte mehr *in den Prädiktoren*.

Nach fehlenden Werten könnte man z.B. auch so suchen:

```
datawizard::describe_distribution(d_train_baked2)
```

```
## Variable   |    Mean |     SD |     IQR |              Range | Skewness | Kurtosis |    n | n_Missing
## ------------------------------------------------------------------------------------------------------
## id         | 1500.50 | 866.17 | 1500.50 |    [1.00, 3000.00] |     0.00 |    -1.20 | 3000 |         0
## popularity |    8.46 |  12.10 |    6.88 | [1.00e-06, 294.34] |    14.38 |   280.10 | 3000 |         0
## runtime    |  107.85 |  22.08 |   24.00 |     [0.00, 338.00] |     1.02 |     8.20 | 3000 |         0
## budget     |   16.09 |   1.89 |    1.90 |      [0.00, 19.76] |    -2.93 |    18.71 | 3000 |         0
## revenue    |   15.97 |   3.04 |    3.37 |      [2.30, 21.14] |    -1.60 |     3.82 | 3000 |         0
```

So bekommt man gleich noch ein paar Infos über die Verteilung der Variablen. Praktische Sache.

Check Test-Sample

Das Test-Sample backen wir auch mal, um zu prüfen, das alles läuft:

```
d_test_baked2 <-
  bake(rec2_prepped, new_data = d_test)
```

```
d_test_baked2 %>%
  head()
```

```
## # A tibble: 6 × 4
##      id popularity runtime budget
##   <dbl>      <dbl>   <dbl>  <dbl>
## 1  3001       3.85      90   15.8
## 2  3002       3.56      65   11.4
## 3  3003       8.09     100   16.4
## 4  3004       8.60     130   15.7
## 5  3005       3.22      92   14.5
## 6  3006       8.68     121   16.1
```

Sieht soweit gut aus.

Kreuzvalidierung

```
cv_scheme <- vfold_cv(d_train,
                      v = 5,
                      repeats = 3)
```

Modelle

LM

```
mod_lm <-
  linear_reg()
```

Workflow-Set

Hier nur ein sehr kleiner Workflow-Set.

Das ist übrigens eine gute Strategie: Erstmal mit einem kleinen Prozess anfangen, und dann sukzessive erweitern.

```
preproc2 <- list(rec1 = rec2)
models2 <- list(lm1 = mod_lm)


all_workflows2 <- workflow_set(preproc2, models2)
```

Fitten und tunen

```
tmdb_model_set2 <-
    all_workflows2 %>%
    workflow_map(resamples = cv_scheme)
```

Finalisieren

```
tmdb_model_set2 %>%
  collect_metrics() %>%
  arrange(-mean) %>%
  head(10)
```

```
## # A tibble: 2 × 9
##   wflow_id .config       preproc model .metric .estimator  mean     n
##   <chr>    <chr>         <chr>   <chr> <chr>   <chr>      <dbl> <int>
## 1 rec1_lm1 Preprocessor… recipe  line… rmse    standard   2.48     15
## 2 rec1_lm1 Preprocessor… recipe  line… rsq     standard   0.338    15
## # … with 1 more variable: std_err <dbl>
```

```
best_model_params2 <-
extract_workflow_set_result(tmdb_model_set2, "rec1_lm1") %>%
  select_best()
```

```
## Warning: No value of `metric` was given; metric 'rmse' will be used.
```

```
best_model_params2
```

```
## # A tibble: 1 × 1
##   .config
##   <chr>
## 1 Preprocessor1_Model1
```

Finalisieren

Finalisieren bedeutet:

- Besten Workflow identifizieren (zur Erinnerung: Workflow = Rezept + Modell)
- Den besten Workflow mit den optimalen Modell-Parametern ausstatten
- Damit dann den ganzen Train-Datensatz fitten
- Auf dieser Basis das Test-Sample vorhersagen

```
best_wf2 <-
all_workflows2 %>%
  extract_workflow("rec1_lm1")
```

```
best_wf2
```

```
## == Workflow ====================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------
## 4 Recipe Steps
##
## • step_mutate()
## • step_log()
## • step_impute_knn()
## • step_dummy()
##
## -- Model -----------------------------------------------------
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

```
best_wf_finalized2 <-
  best_wf2 %>%
  finalize_workflow(best_model_params2)
```

```
best_wf_finalized2
```

```
## == Workflow ====================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------
## 4 Recipe Steps
##
## • step_mutate()
## • step_log()
## • step_impute_knn()
## • step_dummy()
##
## -- Model -----------------------------------------------------
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

Final Fit

```
fit_final2 <-
  best_wf_finalized2 %>%
  fit(d_train)
```

```
fit_final2
```

```
## == Workflow [trained] ==========================================
## Preprocessor: Recipe
## Model: linear_reg()
```

```
##
## ── Preprocessor ────────────────────────────────────────────────
## 4 Recipe Steps
##
## • step_mutate()
## • step_log()
## • step_impute_knn()
## • step_dummy()
##
## ── Model ───────────────────────────────────────────────────────
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Coefficients:
## (Intercept)   popularity      runtime       budget
##     1.26186      0.03755      0.01289      0.80752
```

```
preds <-
fit_final2 %>%
  predict(new_data = d_test)

head(preds)
```

```
## # A tibble: 6 × 1
##   .pred
##   <dbl>
## 1  15.3
## 2  11.4
## 3  16.1
## 4  16.0
## 5  14.3
## 6  16.1
```

Achtung, wenn die Outcome-Variable im Rezept verändert wurde, dann würde obiger Code *nicht* durchlaufen.

Grund ist hier beschrieben:

> When predict() is used, it only has access to the predictors (mirroring how this would work with new samples). Even if the outcome column is present, it is not exposed to the recipe. This is generally a good idea so that we can avoid information leakage.
>
> One approach is the use the skip = TRUE option in step_log() so that it will avoid that step during predict() and/or bake(). However, if you are using this recipe with the tune package, there will still be an issue because the metric function(s) would get the predictions in log units and the observed outcome in the original units.
>
> The better approach is, for simple transformations like yours, to log the outcome outside of the recipe (before data analysis and the initial split).

Submission df

```
submission_df <-
  d_test %>%
  select(id) %>%
  bind_cols(preds) %>%
  rename(revenue = .pred)

head(submission_df)
```

```
## # A tibble: 6 × 2
##      id revenue
##   <dbl>   <dbl>
## 1  3001    15.3
## 2  3002    11.4
## 3  3003    16.1
## 4  3004    16.0
## 5  3005    14.3
## 6  3006    16.1
```

Zurücktransformieren

```
submission_df <-
  submission_df %>%
  mutate(revenue = exp(revenue)-1)

head(submission_df)
```

```
## # A tibble: 6 × 2
##      id   revenue
##   <dbl>     <dbl>
## 1  3001  4435143.
## 2  3002    91755.
## 3  3003  9782986.
## 4  3004  8573795.
## 5  3005  1598106.
## 6  3006 10061439.
```

Hier ein Beispiel, warum $e^x - 1$ genauer ist für kleine Zahlen als e^x.

Abspeichern und einreichen:

```
write_csv(submission_df, file = "submission.csv")
```

Kaggle Score

Diese Submission erzielte einen Score von **Score: 2.46249** (RMSLE).

```
sol <- 2.5
```

5. **Aufgabe**

Melden Sie sich an für die Kaggle Competition TMDB Box Office Prediction - Can you predict a movie's worldwide box office revenue?.

Sie benötigen dazu ein Konto; es ist auch möglich, sich mit seinem Google-Konto anzumelden.

Bei diesem Prognosewettbewerb geht es darum, vorherzusagen, wieviel Umsatz wohl einige Filme machen werden. Als Prädiktoren stehen einige Infos wie Budget, Genre, Titel etc. zur Verfügung. Eine klassische "predictive Competition" also :-) Allerdings können immer ein paar Schwierigkeiten auftreten ;-)

Aufgabe

Erstellen Sie ein Random-Forest-Modell mit Tidymodels!

Hinweise

- Verzichten Sie auf Vorverarbeitung.
- Tunen Sie die typischen Parameter.
- Reichen Sie das Modell ein und berichten Sie Ihren Score.
- Begrenzen Sie sich auf folgende Prädiktoren.
- Verwenden Sie (langweiligerweise) nur ein lineares Modell.

```
preds_chosen <-
  c("id", "budget", "popularity", "runtime")
```

Lösung

Pakete starten

```
library(tidyverse)
library(tidymodels)
library(tictoc)
```

Daten importieren

```
d_train_path <- "https://raw.githubusercontent.com/sebastiansauer/Lehre/main/data/tmdb-box-office-prediction/train.csv"
d_test_path <- "https://raw.githubusercontent.com/sebastiansauer/Lehre/main/data/tmdb-box-office-prediction/test.csv"

d_train <- read_csv(d_train_path)
d_test <- read_csv(d_test_path)
```

Werfen wir einen Blick in die Daten:

```
glimpse(d_train)
```

```
## Rows: 3,000
## Columns: 23
## $ id                    <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12…
## $ belongs_to_collection <chr> "[{'id': 313576, 'name': 'Hot Tub Tim…
## $ budget                <dbl> 1.40e+07, 4.00e+07, 3.30e+06, 1.20e+0…
## $ genres                <chr> "[{'id': 35, 'name': 'Comedy'}]", "[{…
## $ homepage              <chr> NA, NA, "http://sonyclassics.com/whip…
## $ imdb_id               <chr> "tt2637294", "tt0368933", "tt2582802"…
## $ original_language     <chr> "en", "en", "en", "hi", "ko", "en", "…
## $ original_title        <chr> "Hot Tub Time Machine 2", "The Prince…
## $ overview              <chr> "When Lou, who has become the \"fathe…
## $ popularity            <dbl> 6.575393, 8.248895, 64.299990, 3.1749…
## $ poster_path           <chr> "/tQtWuwvMf0hCc2QR2tkolwl7c3c.jpg", "…
## $ production_companies  <chr> "[{'name': 'Paramount Pictures', 'id'…
## $ production_countries  <chr> "[{'iso_3166_1': 'US', 'name': 'Unite…
## $ release_date          <chr> "2/20/15", "8/6/04", "10/10/14", "3/9…
## $ runtime               <dbl> 93, 113, 105, 122, 118, 83, 92, 84, 1…
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'Englis…
## $ status                <chr> "Released", "Released", "Released", "…
## $ tagline               <chr> "The Laws of Space and Time are About…
## $ title                 <chr> "Hot Tub Time Machine 2", "The Prince…
## $ Keywords              <chr> "[{'id': 4379, 'name': 'time travel'}…
## $ cast                  <chr> "[{'cast_id': 4, 'character': 'Lou', …
## $ crew                  <chr> "[{'credit_id': '59ac067c92514107af02…
## $ revenue               <dbl> 12314651, 95149435, 13092000, 1600000…
```

```
glimpse(d_test)
```

```
## Rows: 4,398
## Columns: 22
## $ id                    <dbl> 3001, 3002, 3003, 3004, 3005, 3006, 3…
```

```
## $ belongs_to_collection <chr> "[{'id': 34055, 'name': 'Pokémon Coll…
## $ budget                <dbl> 0.00e+00, 8.80e+04, 0.00e+00, 6.80e+0…
## $ genres                <chr> "[{'id': 12, 'name': 'Adventure'}, {'…
## $ homepage              <chr> "http://www.pokemon.com/us/movies/mov…
## $ imdb_id               <chr> "tt1226251", "tt0051380", "tt0118556"…
## $ original_language     <chr> "ja", "en", "en", "fr", "en", "en", "…
## $ original_title        <chr> "ディアルガVSパルキアVSダークライ", "…
## $ overview              <chr> "Ash and friends (this time accompani…
## $ popularity            <dbl> 3.851534, 3.559789, 8.085194, 8.59601…
## $ poster_path           <chr> "/tnftmLMemPLduW6MRyZE0ZUD19z.jpg", "…
## $ production_companies  <chr> NA, "[{'name': 'Woolner Brothers Pict…
## $ production_countries  <chr> "[{'iso_3166_1': 'JP', 'name': 'Japan…
## $ release_date          <chr> "7/14/07", "5/19/58", "5/23/97", "9/4…
## $ runtime               <dbl> 90, 65, 100, 130, 92, 121, 119, 77, 1…
## $ spoken_languages      <chr> "[{'iso_639_1': 'en', 'name': 'Englis…
## $ status                <chr> "Released", "Released", "Released", "…
## $ tagline               <chr> "Somewhere Between Time & Space... A …
## $ title                 <chr> "Pokémon: The Rise of Darkrai", "Atta…
## $ Keywords              <chr> "[{'id': 11451, 'name': 'pok√©mon'}, …
## $ cast                  <chr> "[{'cast_id': 3, 'character': 'Tonio'…
## $ crew                  <chr> "[{'credit_id': '52fe44e7c3a368484e03…
```

`preds_chosen` sind alle Prädiktoren im Datensatz, oder nicht? Das prüfen wir mal kurz:

```
preds_chosen %in% names(d_train) %>%
  all()
```

```
## [1] TRUE
```

Ja, alle Elemente von `preds_chosen` sind Prädiktoren im (Train-)Datensatz.

CV

```
cv_scheme <- vfold_cv(d_train)
```

Rezept 1

```
rec1 <-
  recipe(revenue ~ budget + popularity + runtime, data = d_train) %>%
  step_impute_bag(all_predictors()) %>%
  step_naomit(all_predictors())
rec1
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          3
##
## Operations:
##
## Bagged tree imputation for all_predictors()
## Removing rows with NA values in all_predictors()
```

Man beachte, dass noch 21 Prädiktoren angezeigt werden, da das Rezept noch nicht auf den Datensatz angewandt (“gebacken”) wurde.

```
tidy(rec1)
```

```
## # A tibble: 2 × 6
##   number operation type       trained skip  id
##    <int> <chr>     <chr>      <lgl>   <lgl> <chr>
## 1      1 step      impute_bag FALSE   FALSE impute_bag_dQpex
## 2      2 step      naomit     FALSE   FALSE naomit_Rkvrr
```

Rezept checken:

```
prep(rec1)
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          3
##
## Training data contained 3000 data points and 2 incomplete rows.
##
## Operations:
##
## Bagged tree imputation for budget, popularity, runtime [trained]
## Removing rows with NA values in budget, popularity, runtime [trained]
```

```
d_train_baked <-
  rec1 %>%
  prep() %>%
  bake(new_data = NULL)

glimpse(d_train_baked)
```

```
## Rows: 3,000
## Columns: 4
## $ budget     <dbl> 1.40e+07, 4.00e+07, 3.30e+06, 1.20e+06, 0.00e+00…
## $ popularity <dbl> 6.575393, 8.248895, 64.299990, 3.174936, 1.14807…
## $ runtime    <dbl> 93, 113, 105, 122, 118, 83, 92, 84, 100, 91, 119…
## $ revenue    <dbl> 12314651, 95149435, 13092000, 16000000, 3923970,…
```

Fehlende Werte noch übrig?

```
library(easystats)
describe_distribution(d_train_baked) %>%
  select(Variable, n_Missing)
```

```
## Variable   | n_Missing
## ----------------------
## budget     |         0
## popularity |         0
## runtime    |         0
## revenue    |         0
```

Modell 1

```
model_lm <- linear_reg()
```

Workflow 1

```
wf1 <-
  workflow() %>%
  add_model(model_lm) %>%
  add_recipe(rec1)
```

Modell fitten (und tunen)

```
doParallel::registerDoParallel(4)
tic()
lm_fit1 <-
  wf1 %>%
  tune_grid(resamples = cv_scheme)
```

```
## Warning: No tuning parameters have been detected, performance will
## be evaluated using the resamples with no tuning. Did you want to
## [tune()] parameters?
```

```
toc()
```

```
## 2.734 sec elapsed
```

```
lm_fit1[[".notes"]][1]
```

```
## [[1]]
## # A tibble: 0 × 3
## # … with 3 variables: location <chr>, type <chr>, note <chr>
```

Final Fit

```
fit1_final <-
  wf1 %>%
  fit(d_train)

fit1_final
```

```
## == Workflow [trained] ==========================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## ── Preprocessor ────────────────────────────────────────────────
## 2 Recipe Steps
##
## • step_impute_bag()
## • step_naomit()
##
## ── Model ───────────────────────────────────────────────────────
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Coefficients:
## (Intercept)       budget   popularity      runtime
##  -2.901e+07    2.482e+00    2.604e+06    1.648e+05
```

```
preds <-
  fit1_final %>%
  predict(d_test)
```

Submission df

```
submission_df <-
  d_test %>%
  select(id) %>%
  bind_cols(preds) %>%
  rename(revenue = .pred)

head(submission_df)
```

```
## # A tibble: 6 × 2
##      id   revenue
##   <dbl>     <dbl>
## 1  3001 -4147868.
## 2  3002 -8809025.
## 3  3003  8523824.
## 4  3004 31675556.
## 5  3005  -504679.
## 6  3006 13531638.
```

Abspeichern und einreichen:

```
#write_csv(submission_df, file = "submission.csv")
```

Kaggle Score

Diese Submission erzielte einen Score von **Score: 6.14787** (RMSLE).

```
sol <- 6.14787
```

6. **Aufgabe**

Ein merkwürdiger Fehler bzw. eine merkwürdige Fehlermeldung in Tidymodels - das untersuchen wir hier genauer und versuchen das Phänomen zu erklären.

Aufgabe

Erläutern Sie die Ursachen des Fehlers! Schalten Sie den Fehler an und ab, um zu zeigen, dass Sie Ihn verstehen.

Startup

```
library(tidyverse)
library(tidymodels)
```

Data import

```
data("mtcars")

d_train <- mtcars %>% slice(1:20)
d_test <- mtcars %>% slice(21:nrow(mtcars))
```

Recipe

```
preds_chosen <- c("hp", "disp", "am")

rec1 <-
  recipe( ~ ., data = d_train) %>%
  update_role(all_predictors(), new_role = "id") %>%
  update_role(all_of(preds_chosen), new_role = "predictor") %>%
  update_role(mpg, new_role = "outcome")
rec1
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##         id          7
##    outcome          1
##  predictor          3
```

```
d_train_baked <-
  rec1 %>%
  prep() %>%
  bake(new_data = NULL)

glimpse(d_train_baked)
```

```
## Rows: 20
## Columns: 11
## $ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, …
## $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 4, …
## $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7…
## $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 18…
## $ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, …
## $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190…
```

```
## $ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00…
## $ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, …
## $ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, …
## $ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, …
## $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, …
```

Model 1

```
model_lm <- linear_reg()
```

Workflow 1

```
wf1 <-
  workflow() %>%
  add_model(model_lm) %>%
  add_recipe(rec1)
```

Fit

```
lm_fit1 <-
  wf1 %>%
  fit(d_train)

preds <-
  lm_fit1 %>%
  predict(d_test)

head(preds)
```

```
## # A tibble: 6 × 1
##   .pred
##   <dbl>
## 1  22.6
## 2  17.2
## 3  17.4
## 4  12.1
## 5  14.9
## 6  28.2
```

Aus Gründen der Reproduzierbarkeit bietet es sich an, eine `SessionInfo` anzugeben:

```
sessionInfo()
```

```
## R version 4.1.3 (2022-03-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Monterey 12.3.1
##
## Matrix products: default
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets
## [7] methods   base
##
## other attached packages:
##  [1] visdat_0.5.3        VIM_6.1.1           colorspace_2.0-3
##  [4] lubridate_1.8.0     rpart_4.1.16        ranger_0.13.1
##  [7] report_0.5.1.1      see_0.7.0.1         correlation_0.8.0.1
## [10] modelbased_0.8.0    effectsize_0.6.0.1  parameters_0.17.0.9
## [13] performance_0.9.0.2 bayestestR_0.12.1   datawizard_0.4.0.17
## [16] insight_0.17.0.6    easystats_0.4.3     tictoc_1.0.1
## [19] yardstick_0.0.9     workflowsets_0.1.0  workflows_0.2.6
## [22] tune_0.2.0.9000     rsample_0.1.1       recipes_0.2.0
## [25] parsnip_0.2.1       modeldata_0.1.1     infer_1.0.0
## [28] dials_0.1.1         scales_1.2.0        broom_0.8.0
## [31] tidymodels_0.1.4    forcats_0.5.1       stringr_1.4.0
## [34] purrr_0.3.4         readr_2.1.2         tidyr_1.2.0
## [37] tibble_3.1.7        ggplot2_3.3.6       tidyverse_1.3.1
## [40] exams_2.3-6         dplyr_1.0.9         colorout_1.2-2
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.3.1        backports_1.4.1    primes_1.1.0
##   [4] plyr_1.8.7          sp_1.4-6           splines_4.1.3
##   [7] listenv_0.8.0       TH.data_1.1-1      digest_0.6.29
##  [10] foreach_1.5.2       htmltools_0.5.2    fansi_1.0.3
##  [13] magrittr_2.0.3      doParallel_1.0.17  openxlsx_4.2.5
##  [16] tzdb_0.1.2          globals_0.14.0     modelr_0.1.8
##  [19] gower_1.0.0         vroom_1.5.7        sandwich_3.0-1
##  [22] hardhat_0.2.0       rvest_1.0.2        haven_2.4.3
##  [25] xfun_0.30           crayon_1.5.1       jsonlite_1.8.0
##  [28] survival_3.2-13     zoo_1.8-9          iterators_1.0.14
##  [31] glue_1.6.2          gtable_0.3.0       ipred_0.9-12
##  [34] emmeans_1.7.3       car_3.0-11         future.apply_1.8.1
##  [37] DEoptimR_1.0-10     abind_1.4-5        mvtnorm_1.1-3
##  [40] DBI_1.1.2           Rcpp_1.0.8.3       laeken_0.5.2
##  [43] xtable_1.8-4        foreign_0.8-82     proxy_0.4-26
##  [46] GPfit_1.0-8         bit_4.0.4          lava_1.6.10
##  [49] prodlim_2019.11.13 vcd_1.4-9          httr_1.4.3
```

```
##  [52] ellipsis_0.3.2     farver_2.1.0       pkgconfig_2.0.3
##  [55] nnet_7.3-17        dbplyr_2.1.1       utf8_1.2.2
##  [58] labeling_0.4.2     tidyselect_1.1.2   rlang_1.0.2
##  [61] DiceDesign_1.9     munsell_0.5.0      cellranger_1.1.0
##  [64] tools_4.1.3        xgboost_1.5.2.1    cli_3.3.0
##  [67] generics_0.1.2     evaluate_0.15      fastmap_1.1.0
##  [70] knitr_1.39         bit64_4.0.5        fs_1.5.2
##  [73] zip_2.2.0          robustbase_0.93-9  future_1.24.0
##  [76] xml2_1.3.3         compiler_4.1.3     rstudioapi_0.13
##  [79] curl_4.3.2         e1071_1.7-9        reprex_2.0.1
##  [82] lhs_1.1.5          stringi_1.7.6      highr_0.9
##  [85] lattice_0.20-45    Matrix_1.4-0       vctrs_0.4.1
##  [88] pillar_1.7.0       lifecycle_1.0.1    furrr_0.2.3
##  [91] lmtest_0.9-39      estimability_1.3   data.table_1.14.2
##  [94] R6_2.5.1           rio_0.5.29         parallelly_1.31.0
##  [97] codetools_0.2-18   boot_1.3-28        MASS_7.3-55
## [100] assertthat_0.2.1   withr_2.5.0        multcomp_1.4-19
## [103] parallel_4.1.3     hms_1.1.1          timeDate_3043.102
## [106] coda_0.19-4        class_7.3-20       rmarkdown_2.14
## [109] carData_3.0-4      pROC_1.18.0        base64enc_0.1-3
```

Lösung

Definiert man das Rezept so:

```
rec2 <- recipe(mpg ~ hp + disp + am, data = d_train)
```

Dann läuft `predict()` brav durch.

Auch dieser Code funktioniert:

```
rec3 <-
  recipe(mpg ~ ., data = d_train) %>%
  update_role(all_predictors(), new_role = "id") %>%
  update_role(all_of(preds_chosen), new_role = "predictor") %>%
  update_role(mpg, new_role = "outcome")
```

Das Problem von `rec1` scheint darin zu legen, dass die *Rollen* der Variablen nicht richtig gelöscht werden, was `predict()` verwirrt:

```
rec1 <-
  recipe(mpg ~ ., data = d_train) %>%
  update_role(all_predictors(), new_role = "id") %>%
  update_role(all_of(preds_chosen), new_role = "predictor") %>%
  update_role(mpg, new_role = "outcome")
rec1
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##         id          7
##    outcome          1
##  predictor          3
```

Daher läuft das Rezept `rec3` durch, wenn man zunächst alle Prädiktoren in ID-Variablen umwandelt: Damit sind alle Rollen wieder sauber.