Introduction to Machine Learning in R

Lab 1: Introduction to Machine Learning in R

Table of contents

Load required packages

```r
library(tidyverse)
library(tidymodels)
library(DataExplorer)
library(modelsummary)
library(visdat)
library(naniar)
library(patchwork)
```

1. Machine Learning Intro: Regression using a Linear Model

Learning outcomes/objective: Learn…

- …how to predict using a regression model relying on tidymodels.

Sources: #TidyTuesday and tidymodels

The data

Below we'll use the European Social Survey (ESS) [Round 10 - 2020. Democracy, Digital social contacts] to illustrate how to use linear models for machine learning. The ESS contains different outcomes amenable to both classification and regression as well as a lot of variables that could be used as features (~580 variables). And we'll focus on the french survey respondents.

The variables were named not so well, so we have to rely on the codebook to understand their meaning. You can find it here or on the website of the ESS.

- `life_satisfaction = stflife`: measures life satisfaction (How satisfied with life as a whole).
- `unemployed_active = uempla`: measures unemployment (Doing last 7 days: unemployed, actively looking for job).
- `unemployed = uempli`: measures life satisfaction (Doing last 7 days: unemployed, not actively looking for job).
- `education = eisced`: measures education (Highest level of education, ES - ISCED).
- `country = cntry`: measures a respondent's country of origin (here held constant for France).
- etc.

	Unique	Missing Pct.	Mean	SD	Min	Median	Max	Histogram
life_satisfaction	12	10	7.0	2.2	0.0	8.0	10.0	
unemployed_active	2	0	0.0	0.2	0.0	0.0	1.0	
unemployed	2	0	0.0	0.1	0.0	0.0	1.0	
education	8	1	3.1	1.9	0.0	3.0	6.0	
age	76	0	49.5	18.7	16.0	50.0	90.0	

We first import the data into R:

```
load(file = here::here("data/data_ess.Rdata"))
```

Inspecting the dataset

First we should make sure to really explore/unterstand our data. How many observations are there? How many different variables (features) are there? What is the scale of the outcome (here we focus on life satisfaction)? What are the averages etc.? What kind of units are in your dataset?

```
#nrow(data)
#ncol(data)
dim(data)
```

```
[1] 1977  346
```

```
# str(data)
# glimpse(data)
# skimr::skim(data)
```

Also always inspect summary statistics for both numeric and categorical variables to get a better understanding of the data. Often such summary statistics will also reveal (coding) errors in the data. Here we take a subset because the are too many variables (>250).

Q: Does anything strike you as interesting the two tables below?

```
data_summary <- data %>%
  select(life_satisfaction, unemployed_active, unemployed, education, age)
datasummary_skim(data_summary, type = "numeric", output = "latex")
```

```r
# datasummary_skim(data_summary, type = "categorical", output = "html")
```

The `table()` function is also useful to get an overview of variables. Use the argument `useNA = "always"` to display potential missings.

```r
table(data$life_satisfaction, useNA = "always")
```

```
   0    1    2    3    4    5    6    7    8    9   10 <NA> 
  34   14   43   50   76  170  170  324  465  213  213  205 
```

```r
table(data$education, data$life_satisfaction, useNA = "always")
```

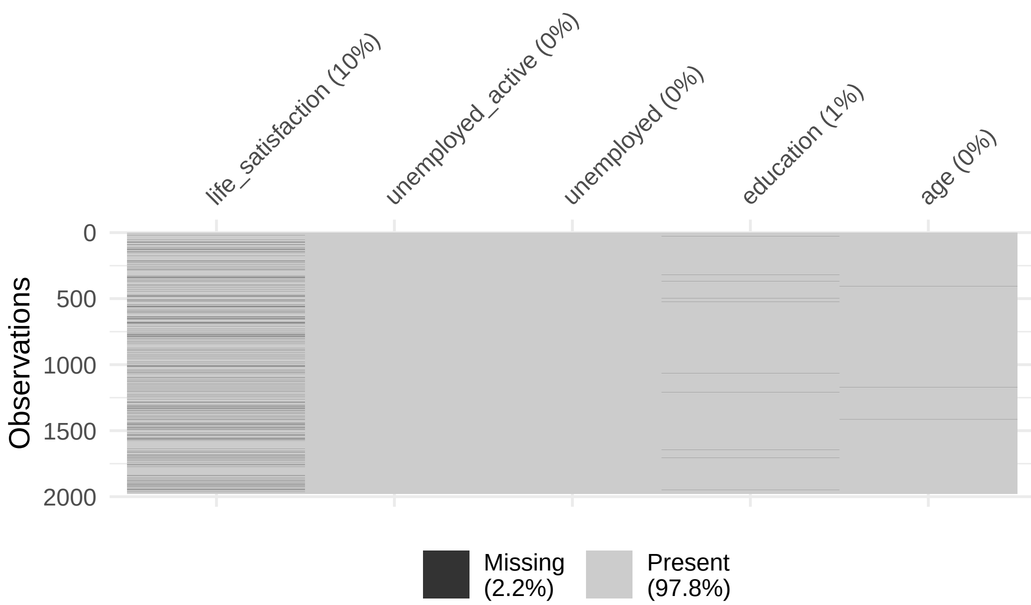
```
      
         0   1   2   3   4   5   6   7   8   9  10 <NA>
  0      8   5   8   3   5  24  23  28  33  12  23   26
  1      2   0   3   3  11  17  16  35  32  15  22   19
  2     16   5  13  16  25  54  46  56  99  40  41   43
  3      4   3  12  12  20  26  30  59  96  47  45   42
  4      1   1   3   6   6  25  21  50  69  32  28   29
  5      1   0   1   4   1   9  12  32  42  16  14   11
  6      2   0   2   5   8  15  21  63  91  50  40   33
  <NA>   0   0   1   1   0   0   1   1   3   1   0    2
```

```r
round(prop.table(table(data$education,
                       data$life_satisfaction, useNA = "always")),2)
```

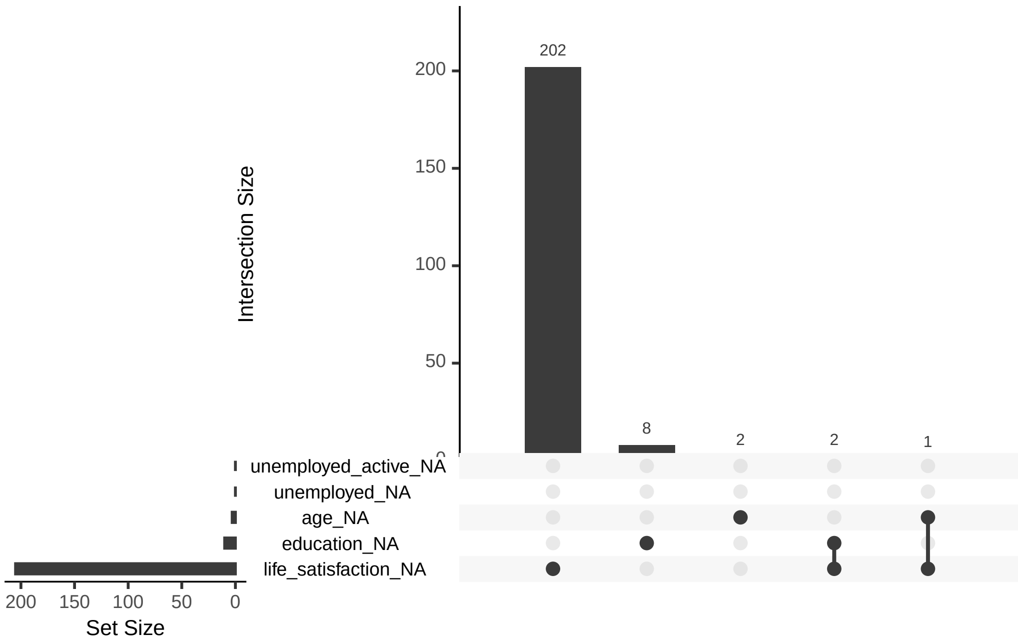
```
      
          0    1    2    3    4    5    6    7    8    9   10 <NA>
  0    0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.01 0.02 0.01 0.01 0.01
  1    0.00 0.00 0.00 0.00 0.01 0.01 0.01 0.02 0.02 0.01 0.01 0.01
  2    0.01 0.00 0.01 0.01 0.01 0.03 0.02 0.03 0.05 0.02 0.02 0.02
  3    0.00 0.00 0.01 0.01 0.01 0.01 0.02 0.03 0.05 0.02 0.02 0.02
  4    0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.03 0.03 0.02 0.01 0.01
  5    0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.02 0.02 0.01 0.01 0.01
  6    0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.03 0.05 0.03 0.02 0.02
  <NA> 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

Finally, there are some helpful functions to explore missing data included in the `naniar` package. Here we do so for a subset of variables. Can you decode those graphs? What do they show? (for publications the design would need to be improved)

```
vis_miss(data %>%
                select(life_satisfaction,
                       unemployed_active,
                       unemployed,
                       education,
                       age))
```




```
gg_miss_upset(data %>%
                select(life_satisfaction,
                       unemployed_active,
                       unemployed,
                       education,
                       age), nsets = 10, nintersects = 10)
```

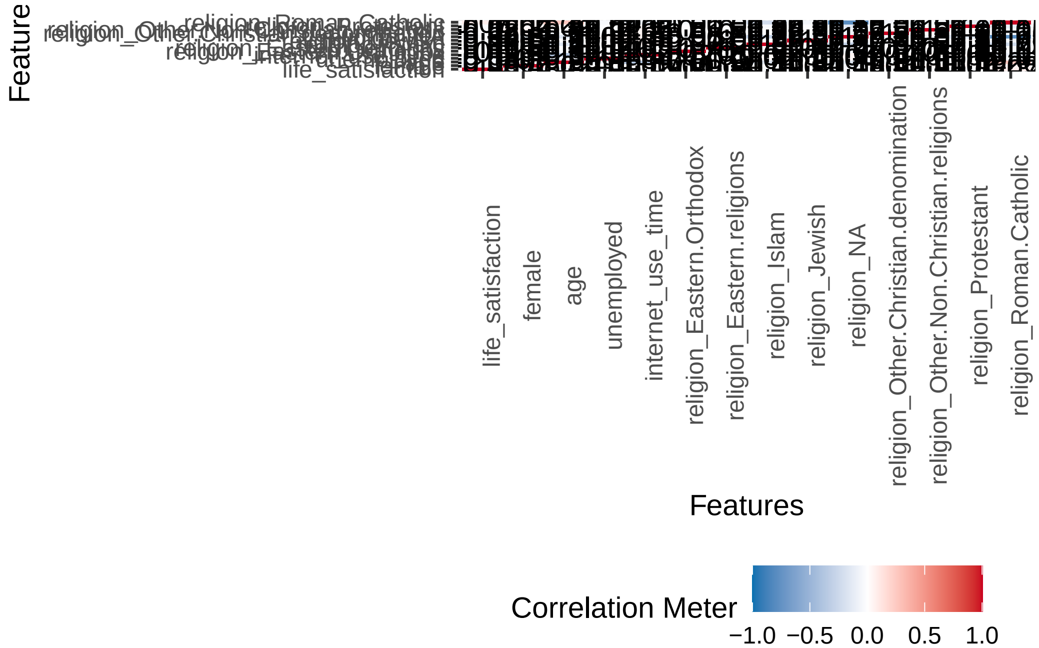


Exploring potential predictors

A correlation matrix can give us first hints regarding important predictors.

- Q: Can we identify anything interesting?

```
plot_correlation(data %>% dplyr::select(life_satisfaction, female,
                                        age, unemployed,
                                        internet_use_time, religion),
                 cor_args = list("use" = "pairwise.complete.obs"))
```



Building a first linear ML model

Below we estimate a simple linear machine learning model only using one split into training and test data. Beforehand we extract the subset of individuals for whom our outcome `life_satisfaction` is missing, store them `data_missing_outcome` and delete those individuals from the actual dataset `data`.

```
# Extract data with missing outcome
  data_missing_outcome <- data %>% filter(is.na(life_satisfaction))
  dim(data_missing_outcome)
```

```
[1] 205 346
```

```
# Omit individuals with missing outcome from data
  data <- data %>% drop_na(life_satisfaction) # ?drop_na
  dim(data)
```

```
[1] 1772  346
```

Then we split the data into training and test data.

```r
# Split the data into training and test data
  data_split <- initial_split(data, prop = 0.80)
  data_split # Inspect
```

```
<Training/Testing/Total>
<1417/355/1772>
```

```r
# Extract the two datasets
  data_train <- training(data_split)
  data_test <- testing(data_split) # Do not touch until the end!
```

Subsequently, we estimate our linear model based on the training data. Below we just use 3 predictors:

```r
# Fit the model
  fit1 <- linear_reg() %>% # linear model
        set_engine("lm") %>% # define lm package/function
        set_mode("regression") %>%# define mode
        fit(life_satisfaction ~ unemployed + age + education, # fit the model
        data = data_train) # based on training data
  fit1
```

```
parsnip model object


Call:
stats::lm(formula = life_satisfaction ~ unemployed + age + education,
    data = data)

Coefficients:
(Intercept)   unemployed          age    education
   6.759429    -1.665729    -0.006379     0.198953
```

```r
  # summary(fit1$fit) # Access fit within the object
```

Then, we predict our outcome in the training data and evaluate the accuracy in the training data.

```r
# Training data: Add predictions
  data_train <- augment(fit1, data_train)

  head(data_train %>%
      select(life_satisfaction, unemployed, age, education, .pred))
```

```
# A tibble: 6 x 5
  life_satisfaction unemployed   age education .pred
              <dbl>      <dbl> <dbl>     <dbl> <dbl>
1                 9          0    32         6  7.75
2                 8          0    37         2  6.92
3                 8          0    86         0  6.21
4                 7          0    17         1  6.85
5                10          0    53         6  7.62
6                 5          0    41         3  7.09
```

```r
# Training data: Metrics
  data_train %>%
      metrics(truth = life_satisfaction, estimate = .pred)
```

```
# A tibble: 3 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      2.18
2 rsq     standard      0.0417
3 mae     standard      1.69
```

- Q: How can we interpret the accuracy metrics? Are we happy? Or should we improve the model for the training data?

Finally, we can also predict data for the test data and evaluate the accuracy in the test data.

```r
# Test data: Add predictions
  data_test <- augment(fit1, data_test)

  head(data_train %>%
      select(life_satisfaction, unemployed, age, education, .pred))
```

```
# A tibble: 6 x 5
  life_satisfaction unemployed   age education .pred
              <dbl>      <dbl> <dbl>     <dbl> <dbl>
```

```
1                    9          0    32           6  7.75
2                    8          0    37           2  6.92
3                    8          0    86           0  6.21
4                    7          0    17           1  6.85
5                   10          0    53           6  7.62
6                    5          0    41           3  7.09
```

```
# Test data: Metrics
  data_test %>%
      metrics(truth = life_satisfaction, estimate = .pred)
```

```
# A tibble: 3 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      2.20
2 rsq     standard      0.0240
3 mae     standard      1.67
```

Q: The accuracy seems similar to that in the training data. What could be the reasons?

- Answer: The split training data/test data was random and both datasets are "relatively" large. And we use a very inflexible model with few features that does not adapt a lot to the training data. With a more flexible model and smaller datasets, more adaption would happen leading to better accuracy in the trainning data (but potentially worse accuracy in the test data).

If we are happy with the accuracy in the test data (the ultimate test for our predictive model) we could then use our model to predict the outcomes for those individuals for which we did not observe the outcome which we stored in `data_missing`.

```
# Missing outcome data predictions
  data_missing_outcome <- augment(fit1, data_missing_outcome)

  head(data_missing_outcome %>%
      select(life_satisfaction, unemployed, age, education, .pred))
```

```
# A tibble: 6 x 5
  life_satisfaction unemployed   age education .pred
              <dbl>      <dbl> <dbl>     <dbl> <dbl>
1                NA          0    62         2  6.76
2                NA          0    48         0  6.45
```

```
3                  NA          0    78           3  6.86
4                  NA          1    53           2  5.15
5                  NA          0    60           6  7.57
6                  NA          0    68           6  7.52
```

```r
# Replace missing outcome variable with the predictions
data_missing_outcome <- data_missing_outcome %>% mutate(life_satisfaction = .pred)
```

Visualizing predictions & errors

It is often insightful to visualize a MLM's predictions, e..g, exploring whether our predictions are better or worse for certain population subsets (e.g., the young). In other words, whether the model works better/worse across groups. Below we take data_test from above (which includes the predictions) and calculate the errors and the absolute errors.

```r
data_test <- data_test %>%
  mutate(errors = life_satisfaction - .pred, # calculate errors
         errors_abs = abs(errors)) %>% # calculate absolute errors
  select(life_satisfaction, unemployed, age, education, .pred, errors, errors_abs) # only kee
head(data_test)
```

```
# A tibble: 6 x 7
  life_satisfaction unemployed   age education .pred errors errors_abs
              <dbl>      <dbl> <dbl>     <dbl> <dbl>  <dbl>      <dbl>
1                10          0    59         2  6.78  3.22       3.22
2                 8          0    63         2  6.76  1.24       1.24
3                 7          0    31         6  7.76 -0.755      0.755
4                 3          0    62         2  6.76 -3.76       3.76
5                 8          0    28         2  6.98  1.02       1.02
6                 8          0    74         0  6.29  1.71       1.71
```

Figure 1 visualizes the variation of errors in a histogram. What can we see?[1]

```r
# Visualize errors and predictors
ggplot(data = data_test,
       aes(x = errors)) +
  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

[1] Life satisfaction mostly underestimated -> positive errors.

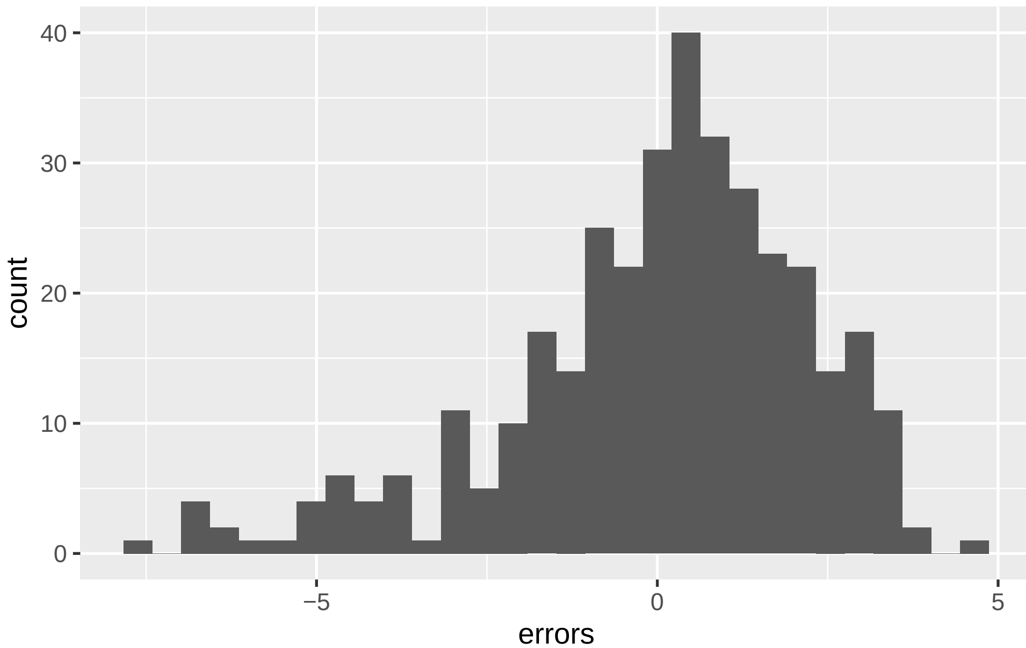


Figure 1: Histogram of errors/residuals

In Figure 2 we visualize the errors as a function of covariates/predictors after discretizing and factorizing the numeric variables.

Q: What can we observe? Why is the prediction error seemingly higher for the unemployed (= 1)?

```
# Visualize errors and predictors
data_plot <- data_test %>%
  select(errors, errors_abs, unemployed, age, education) %>%
  mutate(unemployed = factor(unemployed, ordered = FALSE),
         education = factor(education, ordered = TRUE),
         age = cut_interval(age, 8))


p1 <- ggplot(data = data_plot, aes(y = errors, x = unemployed)) +
  geom_boxplot()
p2 <- ggplot(data = data_plot, aes(y = errors, x = education)) +
  geom_boxplot()
p3 <- ggplot(data = data_plot, aes(y = errors, x = age)) +
      geom_boxplot() +
      theme(axis.text.x = element_text(angle = 30, hjust = 1))
```

```
p1 + p2 + p3
```

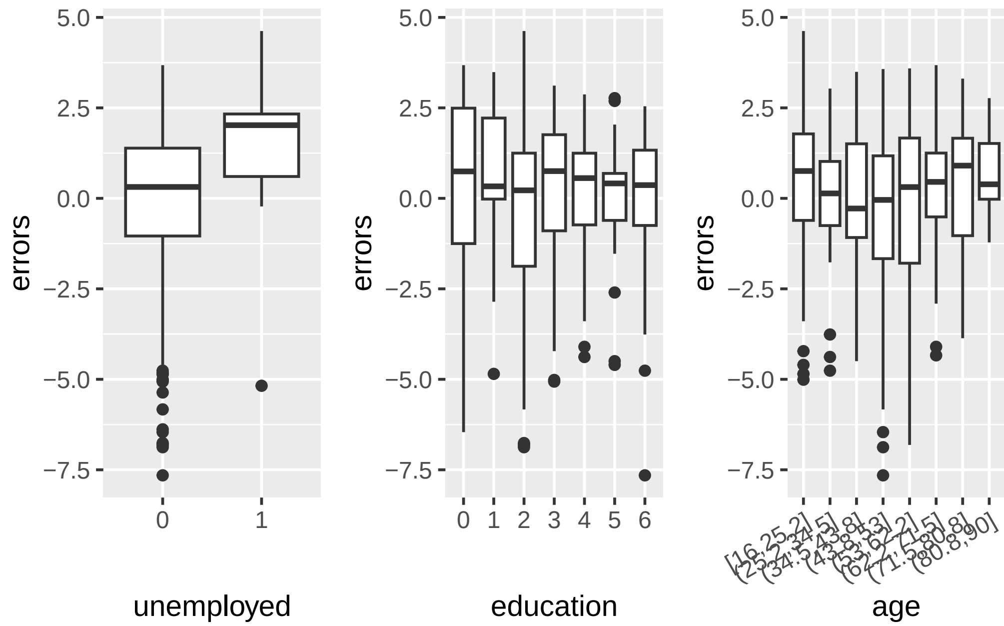


Figure 2: Visualizing prediction errors as a function of predictors/covariates

Exercise: Enhance simple linear model

1. Use the code below to load the data.
2. In the next chunk you find the code we used above to built our first predictive model for our outcome `life_satisfaction`. Please use the code and add further predictors to the model (maybe even `age^2`). Can you find a model with better accuracy in the training data (and better or worse accuracy in the test data?

```
# Extract data with missing outcome
  data_missing_outcome <- data %>% filter(is.na(life_satisfaction))
  dim(data_missing_outcome)

# Omit individuals with missing outcome from data
  data <- data %>% drop_na(life_satisfaction) # ?drop_na
  dim(data)
```

```r
# Split the data into training and test data
  data_split <- initial_split(data, prop = 0.80)
  data_split # Inspect

# Extract the two datasets
  data_train <- training(data_split)
  data_test <- testing(data_split) # Do not touch until the end!

# Fit the model
  fit1 <- linear_reg() %>% # linear model
        set_engine("lm") %>% # define lm package/function
        set_mode("regression") %>%# define mode
        fit(life_satisfaction ~ unemployed + age + education + religion, # fit the model
        data = data_train) # based on training data
  fit1
  # summary(fit1$fit) # Access fit within the object


# Training data: Add predictions
  data_train <- augment(fit1, data_train)

  data_train %>%
      select(life_satisfaction, unemployed, age, education, .pred) %>%
          head()

# Training data: Metrics
   data_train %>%
        metrics(truth = life_satisfaction, estimate = .pred)



# Test data: Add predictions
  data_test <- augment(fit1, data_test)

  data_test %>%
      select(life_satisfaction, unemployed, age, education, .pred) %>%
          head()

# Test data: Metrics
   data_test %>%
        metrics(truth = life_satisfaction, estimate = .pred)
```

Appendix: Same but trying to avoid tidymodels

```r
# Extract data with missing outcome
  data_missing_outcome <- data %>% filter(is.na(life_satisfaction))
  dim(data_missing_outcome)

# Omit individuals with missing outcome from data
  data <- data %>% drop_na(life_satisfaction) # ?drop_na
  dim(data)



# Split the data into training and test data
  randomized_vector <- as.logical(rbinom(n = nrow(data), size = 1, prob = 0.2))
  table(randomized_vector)


  data_split <- initial_split(data, prop = 0.80)
  data_split # Inspect

# Extract the two datasets
  data_train <- data[!randomized_vector,]
  data_test <- data[randomized_vector,]
  dim(data_train)
  dim(data_test)

# Fit the model
  fit1 <- lm(life_satisfaction ~ unemployed + age + education + religion,
             data = data_train)


# Training data: Add predictions
  data_train$.pred <- predict(fit1, data_train)

  head(data_train %>%
         select(life_satisfaction, unemployed, age, education, .pred))

# Training data: Metrics
 data_train %>%
    metrics(truth = life_satisfaction, estimate = .pred)
```

```r
# Test data: Add predictions
  data_test$.pred <- predict(fit1, data_test)

  head(data_test %>%
         select(life_satisfaction, unemployed, age, education, .pred))



  # After that calculate metrics for both training and test data!
  # e.g., RMSE
  sqrt(mean((data_train$life_satisfaction - data_train$.pred)^2, na.rm = TRUE))
  sqrt(mean((data_test$life_satisfaction - data_test$.pred)^2, na.rm = TRUE))
```

2. Machine Learning Intro: Classification using a Logistic Model

Learning outcomes/objective: Learn…

- …how to use trainingset and validation dataset for ML in R.
- …how to predict binary outcomes in R (using a simple logistic regression).
- …how to assess accuracy in R (logistic regression).

Predicting Recidvism: Background story

- Background story by ProPublica: Machine Bias
  - Methodology: How We Analyzed the COMPAS Recidivism Algorithm
- Replication and extension by Dressel and Farid (2018): The Accuracy, Fairness, and Limits of Predicting Recidivism
  - Abstract: "Algorithms for predicting recidivism are commonly used to assess a criminal defendant's likelihood of committing a crime. […] used in pretrial, parole, and sentencing decisions. […] **We show, however, that the widely used commercial risk assessment software COMPAS is no more accurate or fair than predictions made by people with little or no criminal justice expertise**. In addition, despite **COMPAS's collection of 137 features**, the **same accuracy can be achieved with a simple linear classifier with only two features**."
- Very nice lab by Lee, Du, and Guerzhoy (2020): Auditing the COMPAS Score: Predictive Modeling and Algorithmic Fairness
- We will work with the corresponding data and use it to grasp various concepts underlying statistical/machine learning

The data

Our lab is based on Lee, Du, and Guerzhoy (2020) and on James et al. (2013, chap. 4.6.2) with various modifications. We will be using the dataset at LINK that is described by Angwin et al. (2016). - It's data based on the COMPAS risk assessment tools (RAT). RATs are increasingly being used to assess a criminal defendant's probability of re-offending. While COMPAS seemingly uses a larger number of features/variables for the prediction, Dressel and Farid (2018) showed that a model that includes only a defendant's sex, age, and number of priors (prior offences) can be used to arrive at predictions of equivalent quality.

Overview of Compas dataset variables

- `id`: ID of prisoner, numeric
- `name`: Name of prisoner, factor
- `compas_screening_date`: Date of compass screening, date
- `decile_score`: the decile of the COMPAS score, numeric
- `is_recid`: whether somone reoffended/recidivated (=1) or not (=0), numeric
- `is_recid_factor`: same but factor variable
- `age`: a continuous variable containing the age (in years) of the person, numeric
- `age_cat`: age categorized
- `priors_count`: number of prior crimes committed, numeric
- `sex`: gender with levels "Female" and "Male", factor
- `race`: race of the person, factor
- `juv_fel_count`: number of juvenile felonies, numeric
- `juv_misd_count`: number of juvenile misdemeanors, numeric
- `juv_other_count`: number of prior juvenile convictions that are not considered either felonies or misdemeanors, numeric

We first import the data into R:

```r
load(file = here::here("data/data_compas.Rdata"))
```

Inspecting the dataset

The variables were named quite well, so that they are often self-explanatory:

- `decile_score` is the COMPAS score
- `is_recid` wether someone reoffended (1 = recidividate = reoffend, 0 = NOT)
- `race` contains the race
- `age` contains age.
- `priors_count` contains the number of prior offenses

- etc.

First we should make sure to really explore/unterstand our data. How many observations are there? How many different variables (features) are there? What is the scale of the outcome? What are the averages etc.? What kind of units are in your dataset?

```r
nrow(data)
```

```
[1] 7214
```

```r
ncol(data)
```

```
[1] 14
```

```r
dim(data)
```

```
[1] 7214   14
```

```r
str(data) # Better use glimpse()
```

```
tibble [7,214 x 14] (S3: tbl_df/tbl/data.frame)
 $ id                   : num [1:7214] 1 3 4 5 6 7 8 9 10 13 ...
 $ name                 : Factor w/ 7158 levels "aajah herrington",..: 4922 4016 1989 4474 69
 $ compas_screening_date: Date[1:7214], format: "2013-08-14" "2013-01-27" ...
 $ decile_score         : num [1:7214] 1 3 4 8 1 1 6 4 1 3 ...
 $ is_recid             : num [1:7214] 0 1 1 0 0 0 1 NA 0 1 ...
 $ is_recid_factor      : Factor w/ 2 levels "no","yes": 1 2 2 1 1 1 2 NA 1 2 ...
 $ age                  : num [1:7214] 69 34 24 23 43 44 41 43 39 21 ...
 $ age_cat              : Factor w/ 3 levels "25 - 45","Greater than 45",..: 2 1 3 3 1 1 1 1
 $ priors_count         : num [1:7214] 0 0 4 1 2 0 14 3 0 1 ...
 $ sex                  : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 1 2 ...
 $ race                 : Factor w/ 6 levels "African-American",..: 6 1 1 1 6 6 3 6 3 3 ...
 $ juv_fel_count        : num [1:7214] 0 0 0 0 0 0 0 0 0 0 ...
 $ juv_misd_count       : num [1:7214] 0 0 0 1 0 0 0 0 0 0 ...
 $ juv_other_count      : num [1:7214] 0 0 1 0 0 0 0 0 0 0 ...
```

```r
# glimpse(data)
# skimr::skim(data)
```

	Unique	Missing Pct.	Mean	SD	Min	Median	Max	Histogram
id	7214	0	5501.3	3175.7	1.0	5509.5	11 001.0	
decile_score	10	0	4.5	2.9	1.0	4.0	10.0	
is_recid	3	9	0.5	0.5	0.0	0.0	1.0	
age	65	0	34.8	11.9	18.0	31.0	96.0	
priors_count	37	0	3.5	4.9	0.0	2.0	38.0	
juv_fel_count	11	0	0.1	0.5	0.0	0.0	20.0	
juv_misd_count	10	0	0.1	0.5	0.0	0.0	13.0	
juv_other_count	10	0	0.1	0.5	0.0	0.0	17.0	

Also always inspect summary statistics for both numeric and categorical variables to get a better understanding of the data. Often such summary statistics will also reveal errors in the data.

Q: Does anything strike you as interesting the two tables below?

```
datasummary_skim(data, type = "numeric", output = "latex")
```

```
datasummary_skim(data, type = "categorical", output = "latex")
```

```
Warning: These variables were omitted because they include more than 50 levels:
name.
```

The `table()` function is also useful to get an overview of variables. Use the argument `useNA = "always"` to display potential missings.

```
table(data$race, useNA = "always")
```

```

African-American            Asian        Caucasian         Hispanic 
            3696               32             2454              637 
 Native American            Other             <NA> 
              18              377                0 
```

		N	%
is_recid_factor	no	3422	47.4
	yes	3178	44.1
age_cat	25 - 45	4109	57.0
	Greater than 45	1576	21.8
	Less than 25	1529	21.2
sex	Female	1395	19.3
	Male	5819	80.7
race	African-American	3696	51.2
	Asian	32	0.4
	Caucasian	2454	34.0
	Hispanic	637	8.8
	Native American	18	0.2
	Other	377	5.2

```
table(data$is_recid, data$is_recid_factor)
```

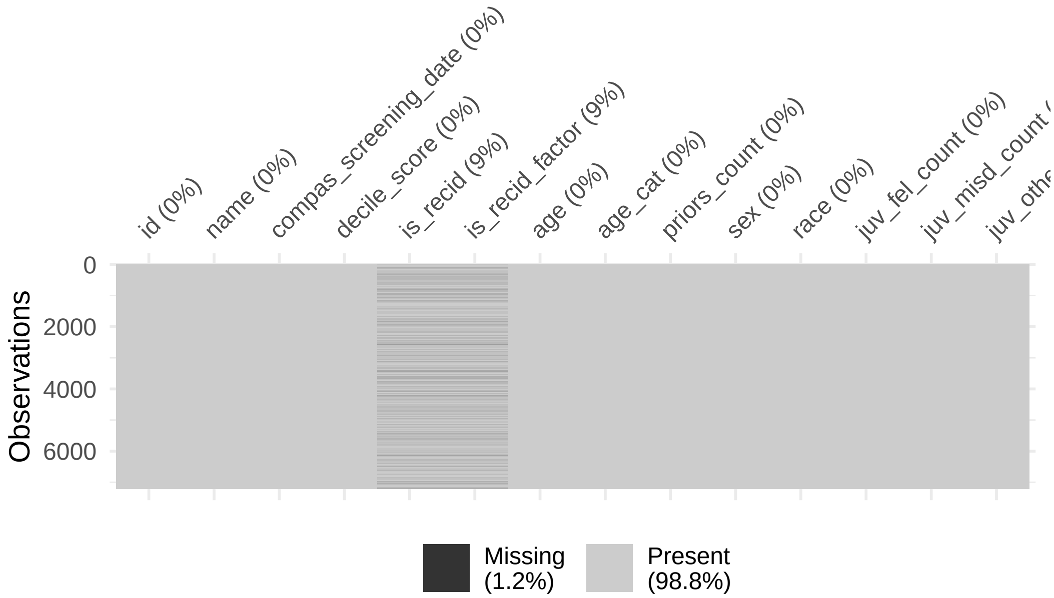
```
      no  yes
  0 3422    0
  1    0 3178
```

```
table(data$decile_score)
```

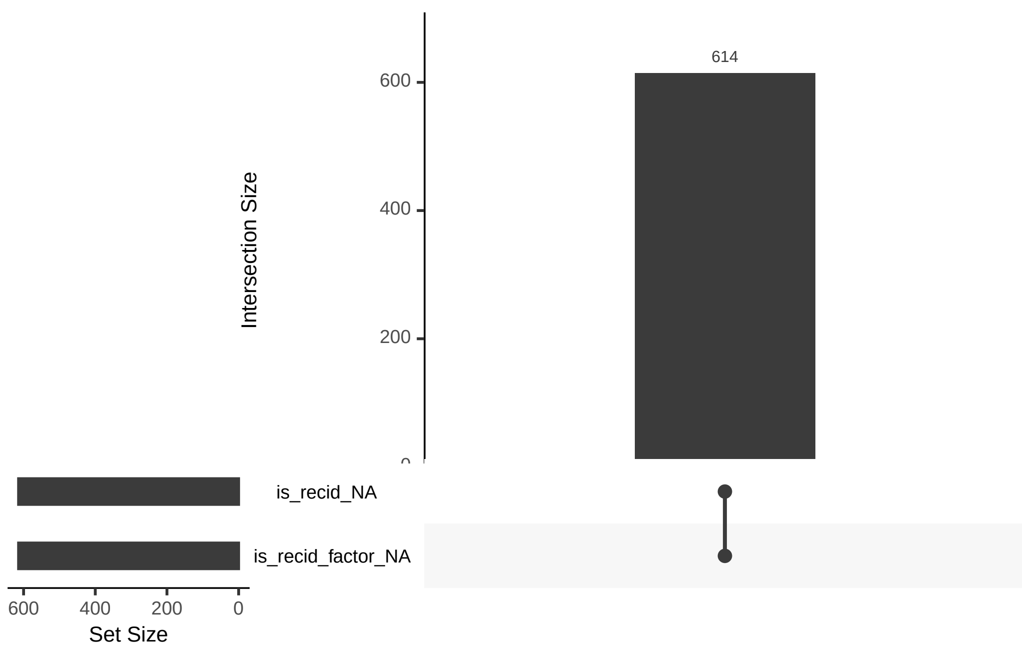
```
   1    2    3    4    5    6    7    8    9   10 
1440  941  747  769  681  641  592  512  508  383 
```

Finally, there are some helpful functions to explore missing data included in the `naniar` package. Can you decode those graphs? What do they show? (for publications the design would need to be improved)

```
vis_miss(data)
```




```r
gg_miss_upset(data, nsets = 2, nintersects = 10)
```



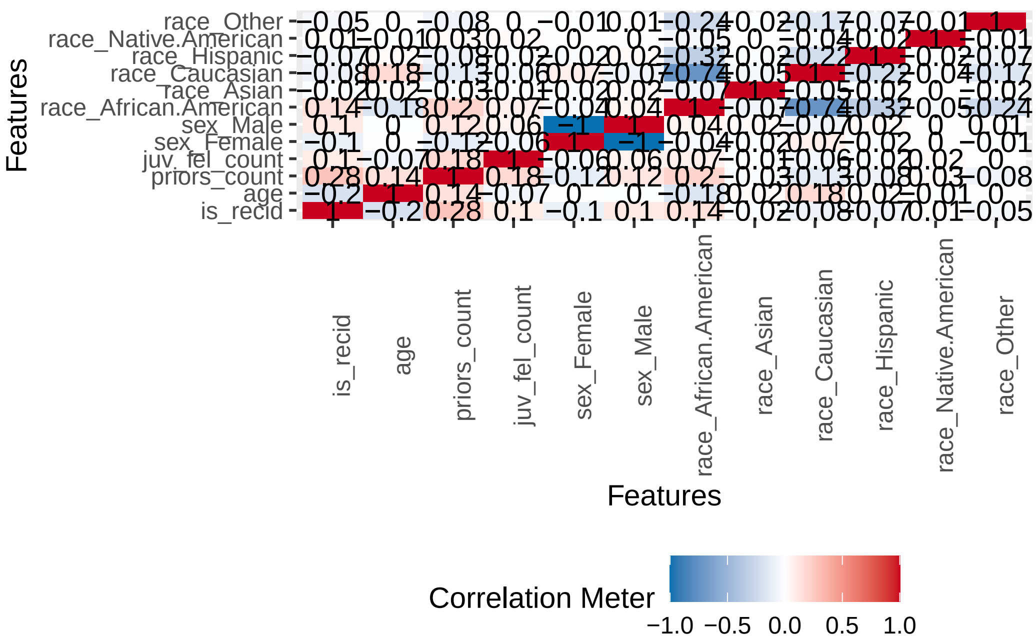

```r
# Ideally, use higher number of nsets/nintersects
# with more screen space
```

Exploring potential predictors

A correlation matrix can give us first hints regarding important predictors.

- Q: Can we identify anything interesting?

```
plot_correlation(data %>% dplyr::select(is_recid, age,
                                        priors_count,sex,
                                        race,
                                        juv_fel_count),
                 cor_args = list("use" = "pairwise.complete.obs"))
```



Building a first logistic ML model

Below we estimate a simple logistic regression machine learning model only using one split into training and test data. To start, we check whether there are any missings on our outcome variable `is_recid_factor` (we use the factor version of our outcome variable). We extract the subset of individuals for whom our outcome `is_recid_factor` is missing, store them `data_missing_outcome` and delete those individuals from the actual dataset `data`.

```
# Extract data with missing outcome
  data_missing_outcome <- data %>% filter(is.na(is_recid_factor))
  dim(data_missing_outcome)
```

```
[1] 614  14
```

```
# Omit individuals with missing outcome from data
  data <- data %>% drop_na(is_recid_factor) # ?drop_na
  dim(data)
```

```
[1] 6600   14
```

Then we split the data into training and test data.

```
# Split the data into training and test data
  data_split <- initial_split(data, prop = 0.80)
  data_split # Inspect
```

```
<Training/Testing/Total>
<5280/1320/6600>
```

```
# Extract the two datasets
  data_train <- training(data_split)
  data_test <- testing(data_split) # Do not touch until the end!
  dim(data_train)
```

```
[1] 5280   14
```

```
  dim(data_test)
```

```
[1] 1320   14
```

Subsequently, we estimate our linear model based on the training data. Below we just use 1 predictor:

```
# Fit the model
  fit1 <- logistic_reg() %>% # logistic model
        set_engine("glm") %>% # define lm package/function
        set_mode("classification") %>%# define mode
        fit(is_recid_factor ~ age, # fit the model
        data = data_train) # based on training data
  fit1 # Class model output with summary(fit1$fit)
```

```
parsnip model object


Call:  stats::glm(formula = is_recid_factor ~ age, family = stats::binomial, 
    data = data)

Coefficients:
(Intercept)          age  
    1.16335     -0.03543  

Degrees of Freedom: 5279 Total (i.e. Null);  5278 Residual
Null Deviance:      7314 
Residual Deviance: 7092     AIC: 7096
```

Then, we predict our outcome in the training data and evaluate the accuracy in the training data.

- Q: How can we interpret the accuracy metrics? Are we happy?

```
# Training data: Add predictions
data_train %>%
  augment(x = fit1, type.predict = "response") %>%
    select(is_recid_factor, age, .pred_class, .pred_no, .pred_yes) %>%
      head()
```

```
# A tibble: 6 x 5
  is_recid_factor   age .pred_class .pred_no .pred_yes
  <fct>           <dbl> <fct>          <dbl>     <dbl>
1 no                 25 yes            0.431     0.569
2 no                 57 no             0.702     0.298
3 yes                23 yes            0.414     0.586
4 yes                20 yes            0.388     0.612
5 no                 55 no             0.687     0.313
6 no                 61 no             0.731     0.269
```

```
# Cross-classification table (Columns = Truth, Rows = Predicted)
data_train %>%
  augment(x = fit1, type.predict = "response") %>%
      conf_mat(truth = is_recid_factor, estimate = .pred_class)
```

```
          Truth
Prediction   no  yes
       no  1496  958
       yes 1228 1598
```

```r
# Training data: Metrics
data_train %>%
  augment(x = fit1, type.predict = "response") %>%
      metrics(truth = is_recid_factor, estimate = .pred_class)
```

```
# A tibble: 2 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.586
2 kap      binary         0.174
```

```r
# F-1 Score
data_train %>%
  augment(x = fit1, type.predict = "response") %>%
      f_meas(truth = is_recid_factor, estimate = .pred_class)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 f_meas  binary         0.578
```

Note: Kappa is a similar measure to accuracy(), but is normalized by the accuracy that would be expected by chance alone and is very useful when one or more classes have large frequency distributions.

Finally, we can also predict data for the test data and evaluate the accuracy in the test data.

```r
# Test data: Add predictions
  data_test %>%
    augment(x = fit1, type.predict = "response") %>%
    select(is_recid_factor, age, .pred_class, .pred_no, .pred_yes) %>%
      head()
```

```
# A tibble: 6 x 5
  is_recid_factor   age .pred_class .pred_no .pred_yes
  <fct>           <dbl> <fct>          <dbl>     <dbl>
```

```
1 no                     69 no              0.783     0.217
2 yes                    47 no              0.623     0.377
3 yes                    31 yes             0.484     0.516
4 yes                    64 no              0.751     0.249
5 no                     32 yes             0.493     0.507
6 yes                    49 no              0.639     0.361
```

```r
# Cross-classification table (Columns = Truth, Rows = Predicted)
  data_test %>%
    augment(x = fit1, type.predict = "response") %>%
      conf_mat(truth = is_recid_factor, estimate = .pred_class)
```

```
          Truth
Prediction  no yes
       no  388 245
       yes 310 377
```

```r
# Test data: Metrics
  data_test %>%
    augment(x = fit1, type.predict = "response") %>%
      metrics(truth = is_recid_factor, estimate = .pred_class)
```

```
# A tibble: 2 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.580
2 kap      binary         0.161
```

Possible reasons if accuracy is higher on test data than training data:

- **Bad training accuracy**: Already bad accuracy in training data is easy to beat.
- **Small Dataset**: Test set may contain easier examples due to small dataset size.
- **Overfitting to Test Data**: Repeated tweaking against the same test set can lead to overfitting.
- **Data Leakage**: Information from the test set influencing the model during training.
- **Strong Regularization**: Techniques like dropout can make the model generalize better but underperform on training data.
- **Evaluation Methodology**: The splitting method can affect results, e.g., stratified splits.
- **Random Variation**: Small test sets can lead to non-representative results.
- **Improper Training**: Inadequate training epochs or improper learning rates.

Below code to visualize the ROC-curve. The function `roc_curve()` calculates the data for the ROC curve.

```
# Calculate data for ROC curve - threshold, specificity, sensitivity
  data_test %>%
    augment(x = fit1, type.predict = "response") %>%
      roc_curve(truth = is_recid_factor, .pred_no) %>%
  head() %>% knitr::kable()
```

Table 1: Data: ROC curve - threshold, specificity, sensitivity

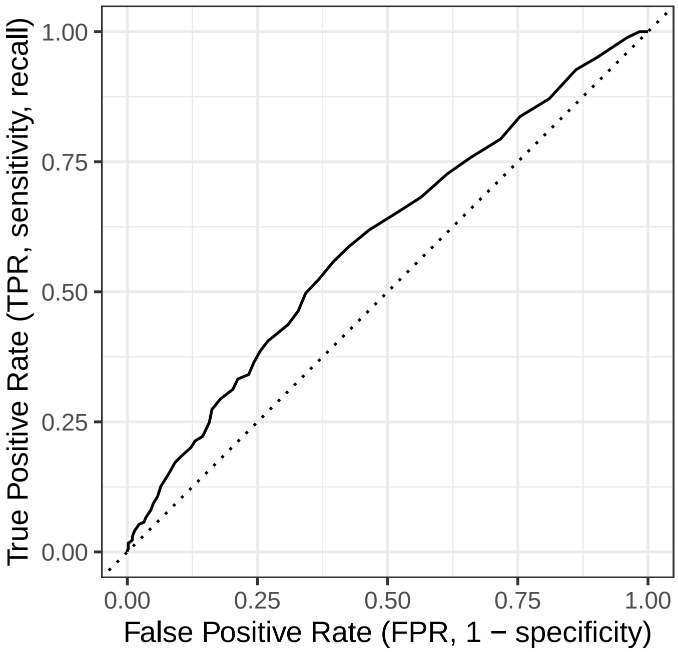
.threshold	specificity	sensitivity
-Inf	0.0000000	1.0000000
0.3715377	0.0000000	1.0000000
0.3798472	0.0016077	1.0000000
0.3882278	0.0160772	1.0000000
0.3966749	0.0401929	0.9885387
0.4051842	0.0964630	0.9512894

We can then visualize is using `autoplot()`. Since it's a ggplot we can make change labels etc. with `+`. Subsequently, we can use `roc_auc()` to calculate the area under the curve.

```
# Calculate data for ROC curve - threshold, specificity, sensitivity
  data_test %>%
    augment(x = fit1, type.predict = "response") %>%
      roc_curve(truth = is_recid_factor, .pred_no) %>%
  head()
```

```
# A tibble: 6 x 3
  .threshold specificity sensitivity
       <dbl>       <dbl>       <dbl>
1   -Inf         0             1
2      0.372     0             1
3      0.380     0.00161       1
4      0.388     0.0161        1
5      0.397     0.0402        0.989
6      0.405     0.0965        0.951
```

```r
# Calculate data for ROC curve and visualize
  data_test %>%
    augment(x = fit1, type.predict = "response") %>%
      roc_curve(truth = is_recid_factor, .pred_no)  %>% # Default: Uses first class (=0=no)
  autoplot() +
  xlab("False Positive Rate (FPR, 1 - specificity)") +
  ylab("True Positive Rate (TPR, sensitivity, recall)")
```




```r
# Calculate are under the curve
  data_test %>%
    augment(x = fit1, type.predict = "response") %>%
      roc_auc(truth = is_recid_factor, .pred_no)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.596
```

If we are happy with the accuracy in the training data we could then use our model to predict the outcomes for those individuals for which we did not observe the outcome which we stored in `data_missing`.

```r
# Missing outcome data predictions
  data_missing_outcome <- data_missing_outcome %>%
                              augment(x = fit1, type.predict = "response")

  data_missing_outcome %>%
    select(is_recid_factor, age, .pred_class, .pred_no, .pred_yes) %>%
        head()
```

```
# A tibble: 6 x 5
  is_recid_factor   age .pred_class .pred_no .pred_yes
  <fct>           <dbl> <fct>          <dbl>     <dbl>
1 <NA>               43 no             0.589     0.411
2 <NA>               31 yes            0.484     0.516
3 <NA>               21 yes            0.397     0.603
4 <NA>               32 yes            0.493     0.507
5 <NA>               30 yes            0.475     0.525
6 <NA>               21 yes            0.397     0.603
```

Visualizing predictions

It is often insightful to visualize a MLM's predictions, e..g, exploring whether our predictions are better or worse for certain population subsets (e.g., the young). In other words, whether the model works better/worse across groups. Below we take data_test from above (which includes the predictions) and calculate the errors and the absolute errors.

```r
data_test %>%
    augment(x = fit1, type.predict = "response") %>%
        select(is_recid_factor, .pred_class, .pred_no, .pred_yes, age, sex, race, priors_cou
```

```
# A tibble: 1,320 x 8
   is_recid_factor .pred_class .pred_no .pred_yes   age sex   race  priors_count
   <fct>           <fct>          <dbl>     <dbl> <dbl> <fct> <fct>        <dbl>
 1 no              no             0.783     0.217    69 Male  Other            0
 2 yes             no             0.623     0.377    47 Fema~ Cauc~            1
 3 yes             yes            0.484     0.516    31 Male  Afri~            7
 4 yes             no             0.751     0.249    64 Male  Afri~           13
 5 no              yes            0.493     0.507    32 Male  Other            0
 6 yes             no             0.639     0.361    49 Male  Other            7
 7 yes             yes            0.466     0.534    29 Male  Afri~            0
 8 no              no             0.639     0.361    49 Male  Cauc~            0
```

```
 9 yes             yes             0.466     0.534    29 Male  Cauc~            1
10 no              yes             0.457     0.543    28 Fema~ Other            0
# i 1,310 more rows
```

Figure 3 visualizes the variation of the predicted probabilites. What can we see?

```r
# Visualize errors and predictors
data_test %>%
    augment(x = fit1, type.predict = "response") %>%
ggplot(aes(x = .pred_yes)) +
  geom_histogram() +
  xlim(0,1)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```

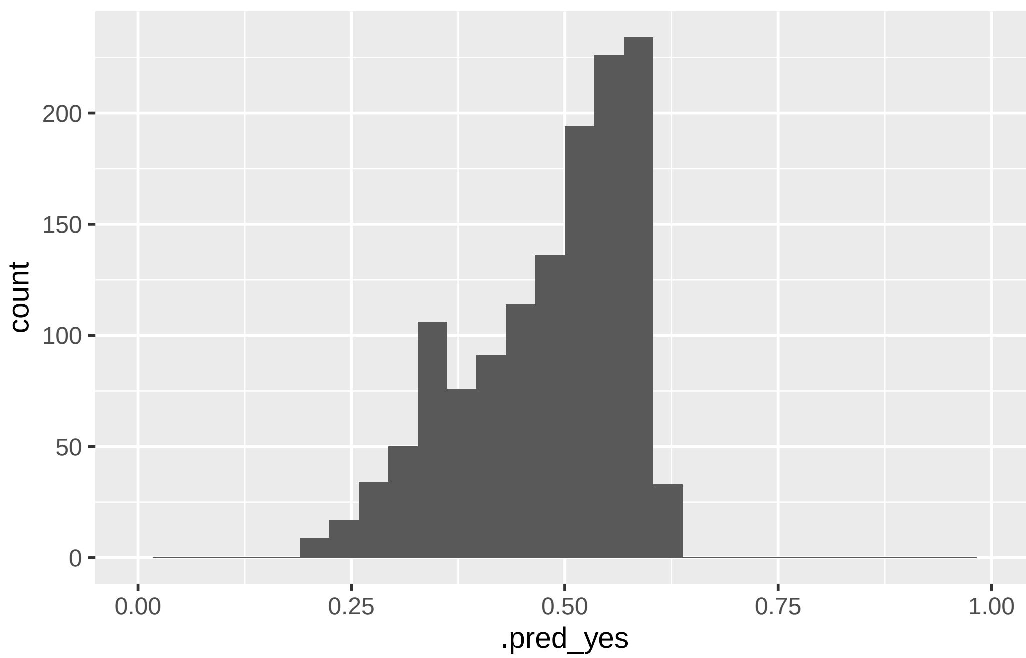


Figure 3: Histogram of errors/residuals

In Figure 4 we visualize the predicted probability of recidivating as a function of covariates/predictors after discretizing and factorizing some variables. Imporantly, the ML model is only based on one of those variables namely `age`, hence, why the predictions do not vary that strongly with the other variables.

Q: What can we observe? What problem does that point to?

```
# Visualize errors and predictors
library(patchwork)
library(ggplot2)
data_plot <- data_test %>%
    augment(x = fit1, type.predict = "response") %>%
  select(.pred_yes, age, sex, race, priors_count) %>%
  mutate(age = cut_interval(age, 8),
         priors_count = as.factor(priors_count))


p1 <- ggplot(data = data_plot, aes(y = .pred_yes, x = sex)) +
  geom_boxplot()
p2 <- ggplot(data = data_plot, aes(y = .pred_yes, x =  age)) +
  geom_boxplot() +
      theme(axis.text.x = element_text(angle = 30, hjust = 1))
p3 <- ggplot(data = data_plot, aes(y = .pred_yes, x = race)) +
      geom_boxplot() +
      theme(axis.text.x = element_text(angle = 30, hjust = 1))
p4 <- ggplot(data = data_plot, aes(y = .pred_yes, x = priors_count)) +
      geom_boxplot() +
      theme(axis.text.x = element_text(angle = 30, hjust = 1))
p1 + p2 + p3 + p4
```

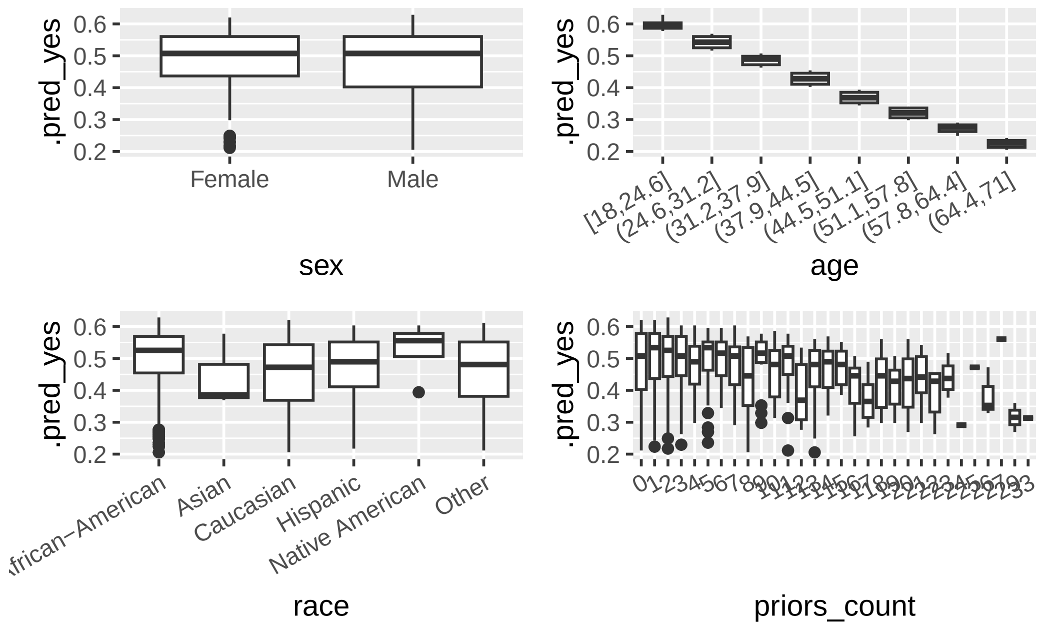


Figure 4: Visualizing predicted probability (for recidvism = yes) as a function of predictors/covariates

Exercise: Enhance simple logistic model

1. Use the code below to load the data.
2. In the next chunk you find the code we used above to built our first predictive model for our outcome `is_recid_factor`. Please use the code and add further predictors to the model. Can you find a model with better accuracy picking further predictors?

```
# Extract data with missing outcome
  data_missing_outcome <- data %>% filter(is.na(is_recid_factor))
  dim(data_missing_outcome)

# Omit individuals with missing outcome from data
  data <- data %>% drop_na(is_recid_factor) # ?drop_na
  dim(data)

# Split the data into training and test data
  data_split <- initial_split(data, prop = 0.80)
  data_split # Inspect

# Extract the two datasets
  data_train <- training(data_split)
```

```r
  data_test <- testing(data_split) # Do not touch until the end!


# Fit the model
  fit1 <- logistic_reg() %>% # logistic model
        set_engine("glm") %>% # define lm package/function
        set_mode("classification") %>%# define mode
        fit(is_recid_factor ~ age, # fit the model
        data = data_train) # based on training data
  fit1

# Training data: Add predictions
data_train %>%
  augment(x = fit1, type.predict = "response") %>%
    select(is_recid_factor, age, .pred_class, .pred_no, .pred_yes) %>%
      head()

# Cross-classification table (Columns = Truth, Rows = Predicted)
data_train %>%
  augment(x = fit1, type.predict = "response") %>%
      conf_mat(truth = is_recid_factor, estimate = .pred_class)

# Training data: Metrics
data_train %>%
  augment(x = fit1, type.predict = "response") %>%
      metrics(truth = is_recid_factor, estimate = .pred_class)



# Test data: Add predictions
  data_test %>%
    augment(x = fit1, type.predict = "response") %>%
    select(is_recid_factor, age, .pred_class, .pred_no, .pred_yes) %>%
      head()

# Cross-classification table (Columns = Truth, Rows = Predicted)
  data_test %>%
    augment(x = fit1, type.predict = "response") %>%
      conf_mat(truth = is_recid_factor, estimate = .pred_class)

# Test data: Metrics
  data_test %>%
```

```
    augment(x = fit1, type.predict = "response") %>%
      metrics(truth = is_recid_factor, estimate = .pred_class)
```

Homework/Exercise:

Above we used a logistic regression model to predict recidivism. In principle, we could also use a linear probability model, i.e., estimate a linear regression and convert the predicted probabilities to a predicted binary outcome variable later on.

1. What might be a problem when we use a linear probability model to obtain predictions (see James et al. (2013), Figure, 4.2, p. 131)?
2. Please use the code above (see next section below) but now change the model to a linear probability model using the same variables. How is the accuracy of the lp-model as compared to the logistic model? Did you expect that?

- Tips
  - The linear probability model is defined through `linear_reg() %>% set_engine('lm') %>% set_mode('regression')`
  - The linear probability model provides a predicted probability that needs to be converted to a binary class variable at the end.
  - The linear probability model requires a numeric outcome, i.e., use `is_recid` as outcome and only convert `is_recid` to a factor at the end (as well as the predicted class).

Solution

```
# Extract data with missing outcome
  data_missing_outcome <- data %>% filter(is.na(is_recid))
  dim(data_missing_outcome)

# Omit individuals with missing outcome from data
  data <- data %>% drop_na(is_recid) # ?drop_na
  dim(data)

# Split the data into training and test data
  data_split <- initial_split(data, prop = 0.80)
  data_split # Inspect

# Extract the two datasets
```

```r
  data_train <- training(data_split)
  data_test <- testing(data_split) # Do not touch until the end!


# Fit the model
  fit1 <- linear_reg() %>% # logistic model
        set_engine("lm") %>% # define lm package/function
        set_mode("regression") %>%# define mode
        fit(is_recid ~ age, # fit the model
        data = data_train) # based on training data
  fit1

# Training data: Add predictions
  data_train <- augment(x = fit1, data_train) %>%
  mutate(.pred_class = as.factor(ifelse(.pred>=0.5, 1, 0)),
         is_recid = factor(is_recid))

  head(data_train %>%
      select(is_recid, is_recid_factor, age, .pred, .resid, .pred_class))

# Training data: Metrics
  data_train %>%
      metrics(truth = is_recid, estimate = .pred_class)



# Test data: Add predictions
  data_test <- augment(x = fit1, data_test) %>%
  mutate(.pred_class = as.factor(ifelse(.pred>=0.5, 1, 0)),
         is_recid = factor(is_recid))

  head(data_test %>%
      select(is_recid, is_recid_factor, age, .pred, .resid, .pred_class))

# Test data: Metrics
  data_test %>%
      metrics(truth = is_recid, estimate = .pred_class)
```


Angwin, Julia, Jeff Larson, Lauren Kirchner, and Surya Mattu. 2016. "Machine Bias." https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

Dressel, Julia, and Hany Farid. 2018. "The Accuracy, Fairness, and Limits of Predicting Recidivism." *Sci Adv* 4 (1): eaao5580.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics. Springer.

Lee, Claire S, Jeremy Du, and Michael Guerzhoy. 2020. "Auditing the COMPAS Recidivism Risk Assessment Tool: Predictive Modelling and Algorithmic Fairness in CS1." In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 535–36. ITiCSE '20. New York, NY, USA: Association for Computing Machinery.