

# Logistic Regression

Dominik Glandorf

2023-01-31

## Data preparation

### Load data

```
setwd("~/EWS")
source("read_data.R")
students = get_student_sub()
```

### See available columns

```
attr(students, "spec")

## cols(
##   mellon_id = col_double(),
##   first_code = col_double(),
##   last_code = col_double(),
##   num_terms = col_double(),
##   first_year = col_double(),
##   first_term = col_double(),
##   first_month = col_double(),
##   first_term_desc = col_character(),
##   last_year = col_double(),
##   last_term = col_character(),
##   last_term_desc = col_character(),
##   last_month = col_double(),
##   number_of_years = col_double(),
##   birth_year = col_double(),
##   birth_month = col_double(),
##   age_at_enrolment = col_double(),
##   start_as_freshman = col_logical(),
##   appl_status = col_character(),
##   uc_total_score = col_double(),
##   uc_read_score = col_double(),
##   uc_writing_score = col_double(),
##   uc_math_score = col_double(),
##   number_ap = col_double(),
##   passed_ap = col_double(),
```

```
## best_ap = col_double(),
## avg_ap = col_double(),
## graduated = col_logical(),
## dropout = col_logical(),
## admitdate = col_character(),
## int_student = col_double(),
## ethnicity = col_character(),
## first_generation = col_double(),
## low_income = col_logical(),
## father_edu_level_code = col_double(),
## mother_edu_level_code = col_double(),
## ell = col_double(),
## single_parent = col_double(),
## foster_care = col_double(),
## household_size_app = col_double(),
## distance_from_home = col_double(),
## sport_at_admission = col_character(),
## cal_res_at_app = col_character(),
## hs_gpa = col_double(),
## toefl_score = col_double(),
## ielts_score = col_double()
## )
```

## Create dataframe for model

```
predictors = c("age_at_enrolment",
               "uc_total_score",
               "uc_math_score",
               "uc_read_score",
               "uc_writing_score",
               "number_ap",
               "passed_ap",
               "best_ap",
               "avg_ap",
               "int_student",
               "ethnicity",
               "first_generation",
               "low_income",
               "father_edu_level_code",
               "mother_edu_level_code",
               "ell",
               "single_parent",
               "foster_care",
               "household_size_app",
               "distance_from_home",
               "sport_at_admission",
               "cal_res_at_app",
               "hs_gpa")
dat = students %>% select(all_of(c(predictors, "dropout")))
```

Strategy for missing categorical values: Create a missing category. Also, merge/exclude categories that are rare.

```

dat$low_income[is.na(dat$low_income)]=F
dat$household_size_app[dat$household_size_app>6] = 6
dat$father_edu_level_code[dat$father_edu_level_code==4]=NA
saa = table(dat$sport_at_admission)
dat$sport_at_admission[dat$sport_at_admission %in% names(saa[saa<nrow(dat)/1000])] = "other"
categorical_cols <- c("int_student", "ethnicity", "first_generation", "father_edu_level_code", "mother_edu_
dat[categorical_cols] <- lapply(dat[categorical_cols], as.character)
dat[categorical_cols][is.na(dat[categorical_cols])] = "unknown"

dat = dat[complete.cases(dat),]
dat[categorical_cols] <- lapply(dat[categorical_cols], as.factor)

#for (c in categorical_cols) {
#  print(table(dat[,c]))
#}

```

We now include 0.355025 of the dataset.

## Model estimation

First, split into training and test data

```

sample <- sample(c(TRUE, FALSE), nrow(dat), replace=TRUE, prob=c(0.7,0.3))
train <- dat[sample, ]
test <- dat[!sample, ]

```

```

log.reg.fit = glm(dropout ~ ., train, family = 'binomial')
#summary(log.reg.fit)

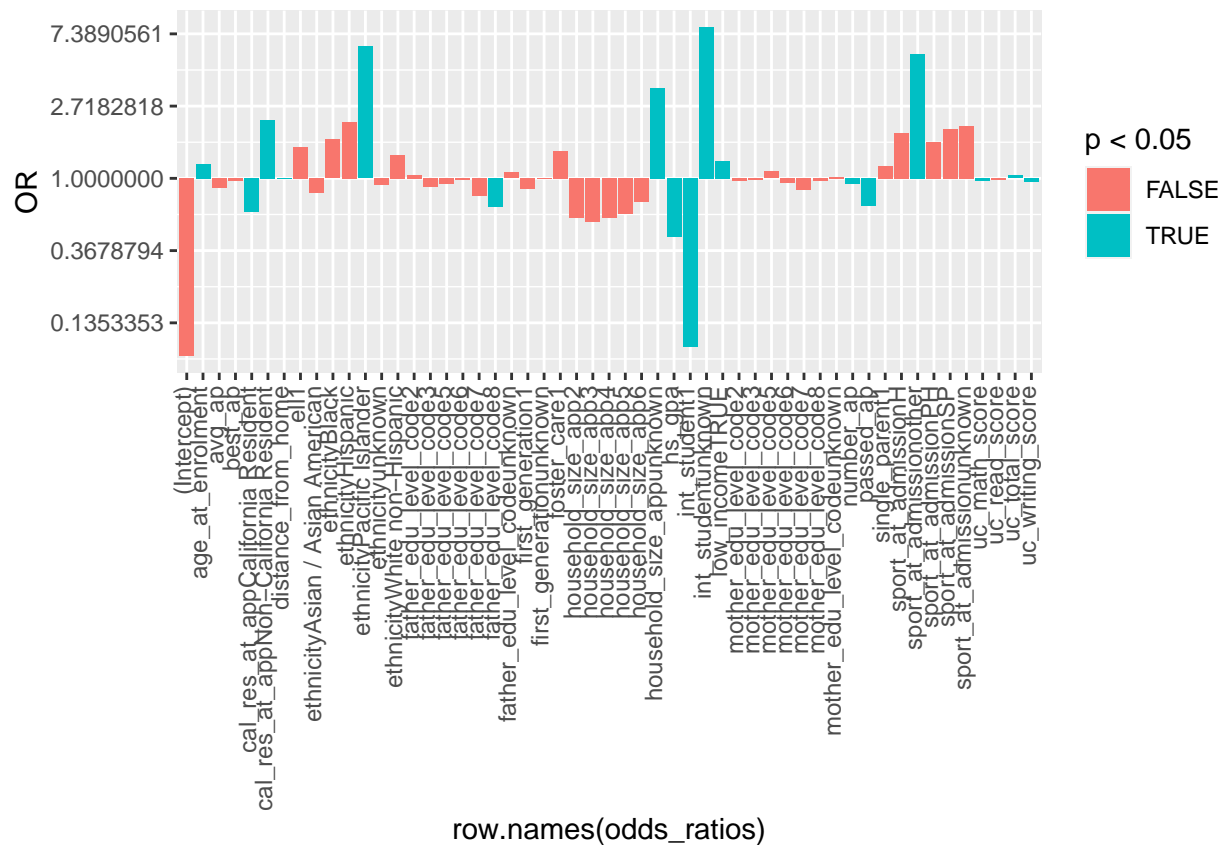
```

## Effects

```

odds_ratios = data.frame(OR=exp(log.reg.fit$coefficients),
                        p=coef(summary(log.reg.fit))[,4])
ggplot(odds_ratios, aes(x=row.names(odds_ratios), y=OR, fill=p<0.05))+
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.25, hjust=1)) +
  scale_y_continuous(trans='log')

```



# Model evaluation

```
dropout_prob = predict(log.reg.fit, test, type = "response") # response outputs the prob instead of log

rocobj <- roc(test$dropout, dropout_prob)
auc <- round(auc(test$dropout, dropout_prob),4)
ggroc(rocobj) +
  ggtitle(paste0('ROC Curve ', '(AUC = ', auc, ')')) +
  geom_abline(intercept = 1, slope = 1, color = "red", linetype = "dashed")
```

