Linear Regression

Thiago Pires

2022-10-25

Table of contents

Introduction

This paper will analyse the main factors associated to car consumption (Miles/(US) gallon). Therefore I will focus on the model interpretability.

Methods

I will use to analyse the R language and the library `tidymodels` to modeling.

Dataset

The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

Table 1: Variable description

Variable	Description
mpg	Miles/(US) gallon
cyl	Number of cylinders
disp	Displacement (cu.in.)
hp	Gross horsepower
drat	Rear axle ratio
wt	Weight (lb/1000)
qsec	1/4 mile time
vs	V/S
am	Transmission (automatic/manual)
gear	Number of forward gears
carb	Number of carburetors

Results

Exploratory analysis

Set the types of the variables

```
mtcars <- mtcars |>
    dplyr::mutate(
        vs = factor(vs, labels = c("V", "S")),
        am = factor(am, labels = c("automatic", "manual")),
        cyl = ordered(cyl),
        gear = ordered(gear),
        carb = ordered(carb)
        )
```

Summary of quantitative variables

```r
describe <- function(data, x) {
    table <- data |>
        dplyr::summarise(
            Min = min({{x}}),
            Max = max({{x}}),
            Mean = mean({{x}}),
            Median = median({{x}}),
            SD = sd({{x}}),
            IQR = IQR({{x}}),
            N = dplyr::n()
        )

        dplyr::tibble(Variable = dplyr::quo_name(dplyr::quo({{x}}))) |>
            dplyr::bind_cols(table)
}

c("mpg", "disp", "hp", "drat", "wt", "qsec") |>
    purrr::map_dfr(~ describe(mtcars, !! rlang::sym(.x))) |>
    knitr::kable()
```

Table 2: Summary

Variable	Min	Max	Mean	Median	SD	IQR	N
mpg	10.400	33.900	20.090625	19.200	6.0269481	7.37500	32
disp	71.100	472.000	230.721875	196.300	123.9386938	205.17500	32
hp	52.000	335.000	146.687500	123.000	68.5628685	83.50000	32
drat	2.760	4.930	3.596563	3.695	0.5346787	0.84000	32
wt	1.513	5.424	3.217250	3.325	0.9784574	1.02875	32
qsec	14.500	22.900	17.848750	17.710	1.7869432	2.00750	32

Frequency to categorical variables

The variable `carb` there are categories (6 and 8) with little counts, so in the next steps they should be aggregated in other classes.

```r
freq <- function(data, x) {

  table <-
    data |>
    dplyr::filter(!is.na({{x}})) |>
```

```
    dplyr::count({{x}}) |>
    dplyr::mutate(`%` = round(n/sum(n, na.rm = TRUE) * 100, 2)) |>
    dplyr::rename(Levels = {{x}}, N = n)

  dplyr::tibble(Variable = dplyr::quo_name(dplyr::quo({{x}}))) |>
    dplyr::bind_rows(dplyr::tibble(Variable = rep("", nrow(table) - 1))) |>
    dplyr::bind_cols(table) |>
    dplyr::mutate(Variable = ifelse(is.na(Variable), "", Variable),
    Levels = as.character(Levels))

}

c("cyl", "vs", "am", "gear", "carb") |>
    purrr::map_dfr(~ freq(mtcars, !! rlang::sym(.x))) |>
    knitr::kable()
```

Table 3: Frequency

Variable	Levels	N	%
cyl	4	11	34.38
	6	7	21.88
	8	14	43.75
vs	V	18	56.25
	S	14	43.75
am	automatic	19	59.38
	manual	13	40.62
gear	3	15	46.88
	4	12	37.50
	5	5	15.62
carb	1	7	21.88
	2	10	31.25
	3	3	9.38
	4	10	31.25
	6	1	3.12
	8	1	3.12

Normality test

The outcome that will be used in the model (`mpg`) has the nomal distribution by the shapiro test (p-value > 0.05).

```r
mtcars$mpg |>
    shapiro.test() |>
    broom::tidy() |>
    knitr::kable()
```

Table 4: Normality test

statistic	p.value	method
0.9475647	0.1228814	Shapiro-Wilk normality test

Modeling

Split in train and test

```r
library(tidymodels)
set.seed(555)

data_split <-
    initial_split(mtcars, prop = 3/4)

train_data <- training(data_split)
test_data  <- testing(data_split)
```

In the next sections we will see the process to fit three proposed models:

- Linear model
- Linear model with polynomial effect
- Linear model with lasso regularization

Fit linear model

```r
linear_mod <-
    linear_reg() |>
    set_engine("lm") |>
    set_mode("regression")

mtcars_rec <- recipe(mpg ~ ., data = train_data)

mtcars_rec <-
```

```r
    mtcars_rec |>
    step_other(carb) |>
    step_dummy(all_nominal_predictors())

mtcars_rec <-
    prep(mtcars_rec, training = train_data)

mtcars_work <- workflow() |>
    add_model(linear_mod) |>
    add_recipe(mtcars_rec)

linear_fit <- mtcars_work |>
    fit(data = train_data)

linear_fit |>
    broom::tidy() |> knitr::kable()
```

Table 5: Linear model

term	estimate	std.error	statistic	p.value
(Intercept)	31.4896922	10.2374451	3.0759327	0.0152104
disp	0.0129109	0.0218535	0.5907899	0.5709705
hp	0.0547162	0.0411630	1.3292559	0.2204198
drat	-0.0398289	1.2712870	-0.0313296	0.9757742
wt	0.5782001	2.0167016	0.2867058	0.7816228
qsec	-1.5404352	0.5962634	-2.5834811	0.0324402
cyl_1	-11.6698662	3.1709919	-3.6801943	0.0062174
cyl_2	-3.5037119	1.6638113	-2.1058349	0.0683146
vs_S	3.9749773	1.4382371	2.7637844	0.0245310
am_manual	5.4103025	2.1066134	2.5682465	0.0332180
gear_1	-4.0776510	1.9102214	-2.1346484	0.0653185
gear_2	-3.1886284	1.5934498	-2.0010850	0.0803810
carb_X2	4.4277477	1.5788774	2.8043646	0.0230423
carb_X3	4.6060004	2.6610922	1.7308684	0.1217213
carb_X4	-5.0112223	3.5921127	-1.3950627	0.2005082
carb_other	-9.6816008	7.6698174	-1.2622987	0.2423948

Evaluation

```r
linear_test_results <-
    predict(linear_fit, new_data = test_data) |>
    dplyr::bind_cols(test_data)

rmse(linear_test_results,
     truth = mpg,
     estimate = .pred) |>
     knitr::kable()
```

Table 6: Evaluation

.metric	.estimator	.estimate
rmse	standard	6.821958

Fit linear model with polynomial effects

```r
mtcars_rec_poly <-
    mtcars_rec |>
    step_poly(disp, hp, drat, wt, qsec)

mtcars_rec_poly <-
    prep(mtcars_rec_poly, training = train_data)

mtcars_work_poly <- workflow() |>
    add_model(linear_mod) |>
    add_recipe(mtcars_rec_poly)

linear_fit_poly <- mtcars_work_poly |>
    fit(data = train_data)

linear_fit_poly |>
    broom::tidy() |> knitr::kable()
```

Table 7: Linear model with polynomial effects

term	estimate	std.error	statistic	p.value
(Intercept)	22.213130	8.618731	2.5773087	0.0819710
cyl_1	-21.330757	16.564145	-1.2877669	0.2881720
cyl_2	-3.464548	8.955606	-0.3868580	0.7246711
vs_S	1.763706	6.454920	0.2732344	0.8024034

term	estimate	std.error	statistic	p.value
am_manual	7.878571	10.958141	0.7189697	0.5241016
gear_1	-4.875037	4.690012	-1.0394508	0.3750090
gear_2	-9.097857	6.923106	-1.3141293	0.2802509
carb_X2	3.236003	3.991036	0.8108177	0.4768230
carb_X3	-1.461222	6.582324	-0.2219918	0.8385737
carb_X4	-11.776031	5.740853	-2.0512685	0.1326024
carb_other	-16.723336	21.771169	-0.7681414	0.4983299
disp_poly_1	14.748998	30.421347	0.4848240	0.6609978
disp_poly_2	-11.742903	25.593375	-0.4588259	0.6775596
hp_poly_1	64.018706	28.062817	2.2812644	0.1068109
hp_poly_2	-12.758826	31.775639	-0.4015285	0.7149228
drat_poly_1	3.777697	4.116112	0.9177828	0.4264142
drat_poly_2	2.399012	9.936545	0.2414333	0.8247806
wt_poly_1	2.049539	16.655052	0.1230581	0.9098424
wt_poly_2	3.207591	8.060534	0.3979378	0.7173022
qsec_poly_1	-5.476535	12.014029	-0.4558450	0.6794745
qsec_poly_2	-12.465728	9.405067	-1.3254267	0.2769267

Evaluation

```
linear_test_results_poly <-
    predict(linear_fit_poly, new_data = test_data) |>
    dplyr::bind_cols(test_data)

rmse(linear_test_results_poly,
     truth = mpg,
     estimate = .pred) |>
     knitr::kable()
```

Table 8: Evaluation

.metric	.estimator	.estimate
rmse	standard	11.05097

Fit linear model with lasso

```r
linear_mod_lasso <-
    linear_reg(penalty = 0.1, mixture = 1) |>
    set_engine("glmnet")

mtcars_work_lasso <- workflow() |>
    add_model(linear_mod_lasso) |>
    add_recipe(mtcars_rec)

linear_fit_lasso <- mtcars_work_lasso |>
    fit(data = train_data)

linear_fit_lasso |>
    broom::tidy() |> knitr::kable()
```

Table 9: Linear model with lasso

term	estimate	penalty
(Intercept)	22.6033196	0.1
disp	0.0000000	0.1
hp	-0.0104285	0.1
drat	0.0000000	0.1
wt	-1.0303888	0.1
qsec	-0.0084777	0.1
cyl_1	-2.4172295	0.1
cyl_2	0.0000000	0.1
vs_S	1.8676995	0.1
am_manual	3.2547373	0.1
gear_1	0.0000000	0.1
gear_2	0.0000000	0.1
carb_X2	2.1401318	0.1
carb_X3	1.1459642	0.1
carb_X4	-0.7024311	0.1
carb_other	-1.3734245	0.1

Evaluation

```r
linear_test_results_lasso <-
    predict(linear_fit_lasso, new_data = test_data) |>
    dplyr::bind_cols(test_data)
```

```r
rmse(linear_test_results_lasso,
     truth = mpg,
     estimate = .pred) |>
     knitr::kable()
```

Table 10: Evaluation

.metric	.estimator	.estimate
rmse	standard	4.263524

Comparing models

Based on **rmse** the best model was the linear model with lasso. The variables with greater effect on the consumption was transmission manual, two carborators compared with one, straight engine compared with v engine, 4 cyliders decrease consuption when compared with 8 cylinders.

```r
evaluate <- function(x) {
   rmse(x,
     truth = mpg,
     estimate = .pred)
}

metrics <-
  purrr::map_dfr(list(linear_test_results,
  linear_test_results_poly,
  linear_test_results_lasso), evaluate)

dplyr::tibble(models = c("linear", "poly", "lasso")) |>
    dplyr::bind_cols(metrics) |>
    knitr::kable()
```

Table 11: Comparing models

models	.metric	.estimator	.estimate
linear	rmse	standard	6.821958
poly	rmse	standard	11.050966
lasso	rmse	standard	4.263524

Discussion

Next steps:

- Test others feature engineering
- Test others model approaches: bayesian approaches for instance
- Use grid search