STATS 7022 - Data Science PG

Assignment 2

Possakorn A1873765

August 07, 2023

Read Dataset

Read the data set from providing sources and preview data summary using **skimr**.

```
diamonds_df <- read_rds('diamonds.rds')

diamonds_df %>%
  skim_without_charts()
```

Table 1: Data summary

Name	Piped data
Number of rows	51513
Number of columns	10
_____	
Column type frequency:	
factor	3
numeric	7
_____	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cut	0	1	FALSE	5	ide: 20598, pre: 13169, ver: 11527, goo: 4683
colour	0	1	FALSE	7	D: 10771, B: 9337, C: 9090, E: 7950
clarity	0	1	FALSE	7	SI1: 12508, VS2: 11722, SI2: 8793, VS1: 7780

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
carat	0	1	0.80	0.47	0.20	0.40	0.70	1.04	5.01
depth	0	1	61.75	1.43	43.00	61.00	61.80	62.50	79.00
table	0	1	57.46	2.24	43.00	56.00	57.00	59.00	95.00
price	0	1	3934.72	3990.54	326.00	950.00	2402.00	5329.00	18823.00

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
x	0	1	5.73	1.12	3.73	4.71	5.70	6.54	10.74
y	0	1	5.74	1.14	3.68	4.72	5.71	6.54	58.90
z	0	1	3.54	0.70	1.07	2.91	3.53	4.04	31.80

```
head(diamonds_df) %>%
  knitr::kable(caption = "The first 6 rows of the dataset, diamonds_clean")
```

Table 4: The first 6 rows of the dataset, diamonds_clean

carat	depth	table	price	x	y	cut	colour	clarity	z
0.23	61.5	55	326	3.95	3.98	ideal	B	SI2	2.43
0.21	59.8	61	326	3.89	3.84	premium	B	SI1	2.31
0.23	56.9	65	327	4.05	4.07	good	B	VS1	2.31
0.31	63.3	58	335	4.34	4.35	good	G	SI2	2.75
0.24	62.8	57	336	3.94	3.96	very good	G	VVS2	2.48
0.24	62.3	57	336	3.95	3.98	very good	F	VVS1	2.47

Output manipulation

create the log(price) as column: log_price as a response variable and drop the previous column: price.

```
diamonds_df <- read_rds('diamonds.rds') %>%
  mutate(log_price = log(price)) %>%
  select(-price)
```

Data splitting

You can split your data into training and testing sets using the initial_split function. And, perform the cross validation folds as k equal to 10 folds.

```
# perform the data splitting
diamonds_split <- initial_split(diamonds_df)
diamonds_train <- training(diamonds_split)
diamonds_test <- testing(diamonds_split)
diamonds_split
```

```
## <Training/Testing/Total>
## <38634/12879/51513>
```

```
# 10 folds - crossvalidation
set.seed(1311)
diamonds_folds <- vfold_cv(diamonds_train, v = 10)
```

Cleaning and Pre-processing

Following the results from table above, most of diamonds data set are already cleaned. However, there are some necessary data cleaning and pre-processing including normalization, outlier handling, and categorical encoding.

Pre-processing

```
diamonds_recipe <- recipe(log_price ~ ., data = diamonds_train) %>%
  step_dummy(all_factor_predictors()) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_YeoJohnson(all_predictors())
```

Model Tuning

model specifications

Create the model specification with tuning the penalty with lasso regression.

```
lasso_model <- linear_reg(
  penalty = tune(),
  mixture = 1) %>%
  set_engine("glmnet") %>%
  set_mode("regression")
lasso_model
```

```
## Linear Regression Model Specification (regression)
##
## Main Arguments:
##   penalty = tune()
##   mixture = 1
##
## Computational engine: glmnet
```

Tuning model by Grid

```
# Create Grid for tuning
lasso_grid <- grid_regular(penalty(), levels = 50)

# create the workflow
lasso_workflow <- workflow() %>%
  add_recipe(diamonds_recipe) %>%
  add_model(lasso_model)

# Tuning Grid
lasso_tune <- tune_grid(
  lasso_workflow,
  resamples = diamonds_folds,
  grid = lasso_grid,
  metrics = metric_set(rmse, mae, rsq)
)
lasso_tune %>% autoplot()
```

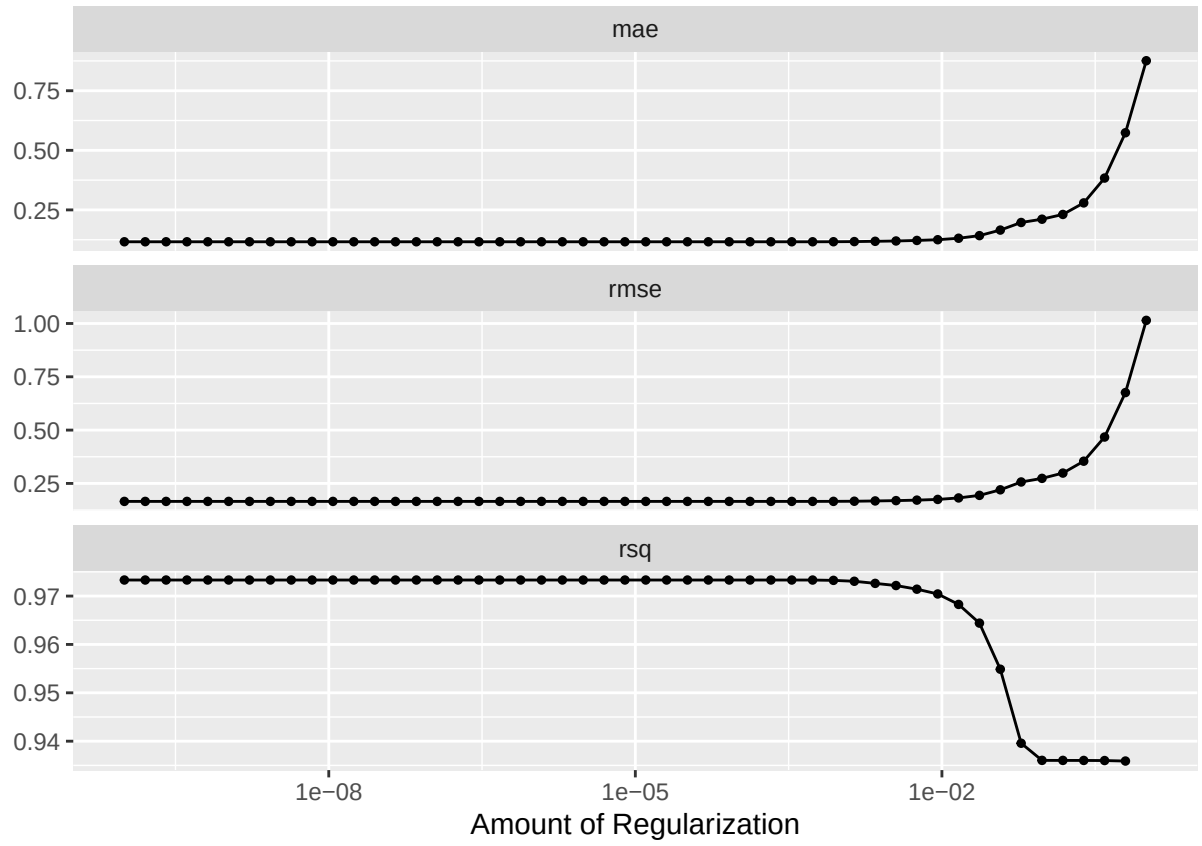

Figure 1: The plot show the hyperparameter tuning following the setting grid

Show the best tuned hyper parameter.

```
lasso_tune_hyper <- select_best(lasso_tune, metric = "rmse")
lasso_tune_hyper %>%
  mutate(penalty = as.character(penalty)) %>%
  knitr::kable(caption = "Show the best hyperparameter output after tuning")
```

Table 5: Show the best hyperparameter output after tuning

penalty	.config
1e-10	Preprocessor1_Model01

Fit with best hyperparameter

```
lasso_workflow_tune <- lasso_workflow %>%
  finalize_workflow(lasso_tune_hyper)
lasso_workflow_tune
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_dummy()
## * step_normalize()
## * step_YeoJohnson()
##
## -- Model -----------------------------------------------------------------------
## Linear Regression Model Specification (regression)
##
## Main Arguments:
##   penalty = 1e-10
##   mixture = 1
##
## Computational engine: glmnet
```

Model Fit

Fit the tuned model from previous part.

```
lasso_fit <- lasso_workflow_tune %>%
  last_fit(split = diamonds_split)

lasso_fit %>%
  collect_metrics() %>%
  knitr::kable(caption = "Show the performance of the tuned model")
```

Table 6: Show the performance of the tuned model

.metric	.estimator	.estimate	.config
rmse	standard	0.1698715	Preprocessor1_Model1
rsq	standard	0.9720502	Preprocessor1_Model1

test the tuned model from previous part.

```
lasso_fit %>% collect_predictions() %>%
  ggplot(aes(log_price, .pred)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_abline(intercept = 0, slope = 1) +
  labs(
    x = "Truth",
    y = "Prediction"
  )
```

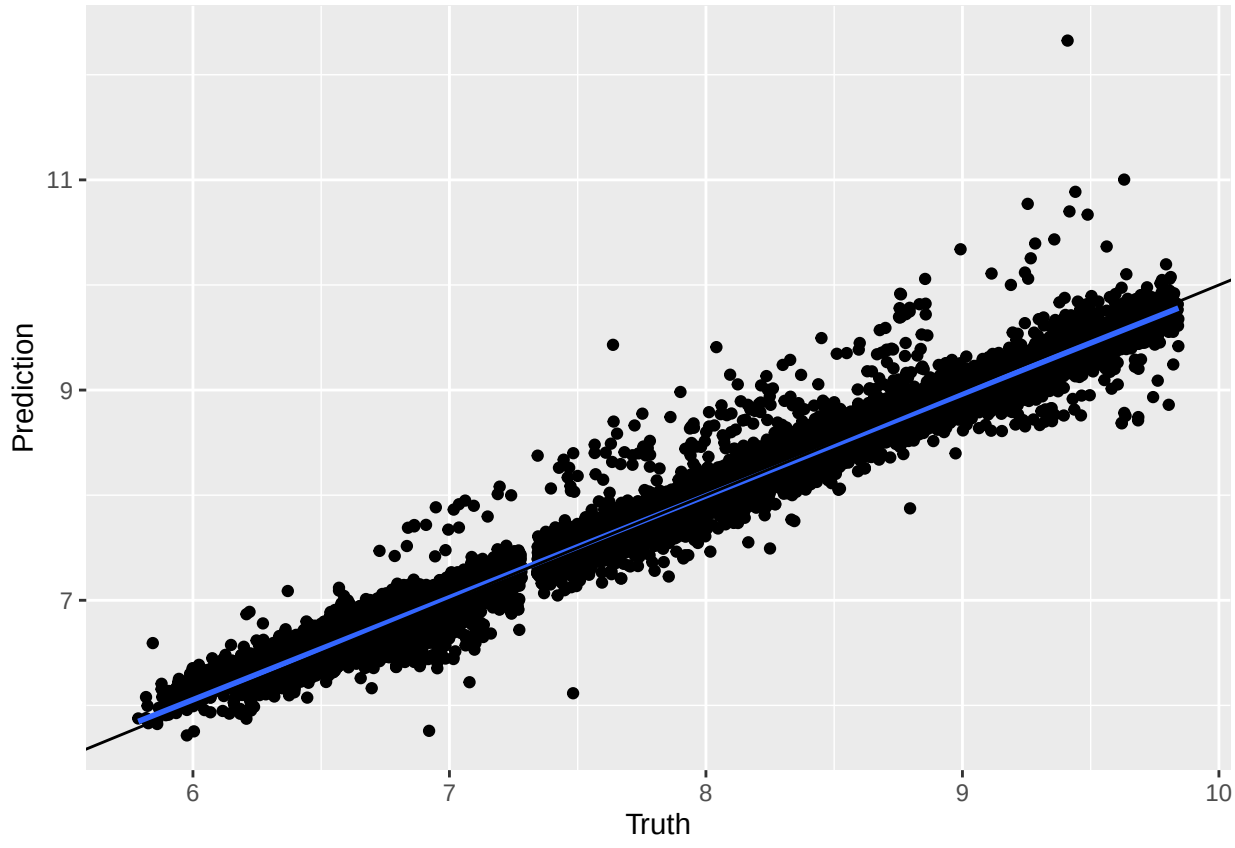

Figure 2: Chart represent the model performance between prediction and truth in log(price)